Copyright Information:

Purpose of publication

When the OTN forum software was upgraded some time around September 2008 there were some performance problems related to threads with very long messages, so these threads were archived out of the public system. (See thread: http://forums.oracle.com/forums/message.jspa?messageID=2780056#2780056) Since I had already published an item on my blog that linked to several points in one such thread to highlight a few interesting topics I asked if I could publish a copy of the thread on my blog so that I could at least change the URLs in my article to page pointers in the copy.

The driving blog article is at http://jonathanlewis.wordpress.com/2008/07/19/block-sizes/ and this pdf file should be read in conjunction with that article. If you decide to link to the document, please do so indirectly by linking to the blog article as this will ensure that your link will still work in the future. If you link directly to the document your link will stop working if I update the document. (This is a feature of how Wordpress.com handles document uploads).

Despite my comment regarding copyright and the licence you granted to Oracle Corporation, if you quoted something from one of your copyrighted websites in the original thread but would like to have it removed from this document, please let me know either by email at jonathan@jlcomp.demon.co.uk or by adding a comment to the blog article – with a precise description of the material you would like removed, and the link showing the prior publication of the material.

If your submission was not relevant to any of the topics that my blog was highlighting, or even if it was relevant but didn't add any significant value, I will remove the text – it may take a few days, though, depending on my timetable.

If another user has replied by quoting and commenting ***usefully*** on the text that you submitted then I may invoke the general rule of "fair use" and the terms of the licence you granted to Oracle Corporation when considering their contribution to the thread.

In response to a request from Burleson Consulting I have deleted two entries that contained material that Don Burleson had quoted from one of the Burleson Consulting websites. Following a subsequent request from Janet Burleson I have also deleted all the other comments made by Don Burleson and all occurrences of his name. The material deleted was not relevant to any of the topics I was highlighting in my blog note.

user619401

Posts: 36
Registered: 2/10/08

Larger vs. Small data block
Posted: Jun 2, 2008 3:31 PM

Reply

Hi guys,

Why does Oracle Adminstration 10g self-study CD rom says that a database that supports data warehousing application may perform better with a larger data block and a database that supports transactional application may perform better with a smaller data block? What's the difference btwn. data warehousing and transactional application? And what does big or small data block do to them?

Many thanks,
Daniel

---

mpowel01

Posts: 2,840
Registered: 12/8/98

Re: Larger vs. Small data block
Posted: Jun 2, 2008 4:32 PM in response to: user619401

Reply

An OLTP database instance would have a large percentage of its SQL consist of single and small row select and update statements for which the data likely be retrieved by key. Lots of small transactions in other words.

A warehouse on the other hand will have heavy full table access to calculate summary amounts and will likely support large data loads. Update transactions are unlikely to compete with concurrent updaters.

HTH -- Mark D Powell --

Re: Larger vs. Small data block
Posted: Jun 2, 2008 7:28 PM in response to: user619401

Reply

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 2, 2008 8:36 PM in response to: user619401

Reply

I'm not sure what you are reading or what others are advising but a simple statement by Bryn Llewellyn (PL/SQL Product Manager, Database and Application Server Technologies Development Group, at Oracle Corporation Headquarters) should clarify it all.

The correct answer for blocks size is 8K because that is the ONLY size Oracle tests.

If you implement any block size other than 8K your benefits, if any, will be marginal and your risks greater. Jonathan Lewis has published a bit on the subject and given that he tests before commenting, unlike others in our industry, you should read his comments.

In my lab nothing we have seen with larger block sizes, except in special contrived situations has been worth the cost of a latte' at Starbucks.

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 2, 2008 9:21 PM in response to: damorgan

Reply

damorgan,

Greg Rahn, Richard Foote, as well as you contributed to this thread related to block size recommendations:
http://forums.oracle.com/forums/thread.jspa?messageID=2445936&#2445936

To avoid repeating history, the OP might find the above thread interesting.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Amardeep Sidhu

Posts: 900
From: amardeepsidhu.com
Registered: 10/27/06

Re: Larger vs. Small data block
Posted: Jun 2, 2008 10:07 PM in response to:

Reply

Hmmmm....

That poor guy just asked about the basics. How using small and large block size affects the performance in OLTP & Data warehousing environments. This whole stuff might be too heavy :(

Amardeep Sidhu

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 2, 2008 10:17 PM in response to: damorgan

Reply

> The correct answer for blocks size is 8K because that
> is the ONLY size Oracle tests.
>

So if 8K blocks is the only correct answer for block size, why Oracle has created the multple-size block buffers? And even it provides Pros/Cons of different block sizes in metalink:

Notes on Choosing an Optimal DB BLOCK SIZE
Doc ID: Note:46757.1

~ Madrid

http://hrivera99.blogspot.com/

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 2, 2008 10:19 PM in response to: Amardeep Sidhu
Reply

It all depend on who is reading now and in the future this thread.

~ Madrid

http://hrivera99.blogspot.com/

Amardeep Sidhu

Posts: 900
From: amardeepsidhu.com
Registered: 10/27/06

Re: Larger vs. Small data block
Posted: Jun 2, 2008 10:30 PM in response to: Madrid
Reply

Hmmm....i am not getting you here properly.

But what OP wanted, Mark gave a perfect answer for that :)

Amardeep Sidhu

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 2, 2008 10:43 PM in response to: Charles Hooper
Reply

Good grief. This thread has been around so long I wrote the same thing in it before.

I truly find it unfathomable that bad advice, such as rebuilding indexes, changing block sizes, etc. gets such wide currency when there is not a single shred of published evidence that, in real-world production applications, they have value.

I guess the fact that spam exists proves the world is full of gullible people.
And snake oil salesmen willing to take advantage of their ignorance.

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 2, 2008 10:45 PM in response to: Madrid
Reply

So that it is possible to use Transportable Tablespaces to move data from a database with one block size to another.

Based on Oracle's very reasonable addition of a feature some people just went off the deep end inventing possible uses for this ... not one of which has ever survived a real-world test. And, of course, we have a small group of people who believe that shouting loudly and waving your hands substitutes for metrics. Thus the mythology continues.

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 2, 2008 11:01 PM in response to: damorgan
Reply

Well, actually I am talking about the above referred metalink note 46757.1 which is a white paper published by Oracle. Is this metalink note a myth?

~ Madrid

http://hrivera99.blogspot.com/

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 2, 2008 11:13 PM in response to: Madrid
Reply

I am familiar with that note and with apologies to the author for my suspicion, I don't believe it to be the result of testing but rather just a marketing piece.

I've never met anyone from Oracle's tuning team that supported it. I've never met any Oak Table network member that supported it. I've never met an Oracle Ace that supported it. And the testing in my lab indicates that except for highly contrived situations it is, at best, a marginal influence.

Further, the note was posted 4 years ago and has never once been updated. It was written for 9i and no one has ever written anything like it for 10.1, 10.2, or 11.1.

We should always keep in mind that Oracle has also published as "fact" things we know today to have never been true such as all of the nonsense about separating tables and indexes into different tablespaces and all of the nonsense about controlling the number of extents.

If the author wishes to put up the test environment information and the test results I would be happy to reconsider my opinion. But based on my own work, and that of others I respect, I think the note is misleading and should be removed.

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 12:11 AM in response to: damorgan
Reply

Metalink notes are official Oracle statements, so they are considered the ultimate truth reference that overrules any ACE's, researcher's or whoever criteria. Just for the sake of truth, and based on your lab tests, would you mind making an official request with the metalink team for this note to be either removed or updated?

~ Madrid

http://hrivera99.blogspot.com/

Mohan Nair

Posts: 612
Registered: 7/14/00

Re: Larger vs. Small data block
Posted: Jun 3, 2008 4:29 AM in response to: user619401
Reply

See this link
http://www.myoracleguide.com/s/MultipleBlocksizes.htm

chris_c

Posts: 160
Registered: 10/17/06

Re: Larger vs. Small data block
Posted: Jun 3, 2008 4:53 AM in response to: damorgan
Reply

>> The correct answer for blocks size is 8K because that is the ONLY size Oracle tests.

do you have a link to a quote on this? Its a fairly broad statement I doubt oracle performs no testing at all on other block sizes, 8k may be tested first/more but it would be nice to see the actuall statements on this one.

Chris

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:08 AM in response to: chris_c
Reply

Over coffee with Bryn in the 300 building earlier this year. It would have been improper to bring along a tape recorder. <g>

But feel free to put the question to him yourself if you wish.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:12 AM in response to: Madrid
Reply

I understand the point you are making but I must disagree.

The fact that metalink notes are official statements does not make them correct. If you don't believe me I suggest you try to install a number of Oracle technologies using them. I can provide you with a list ... a very long list.

The fact is that, just dealing with Physical Data Guard for 10.2.x you can find at least three sets of instructions there that are mutually incompatible. And not one of them is, by itself, correct.

But taking even the most optimistic view of the note you referenced ... it was written for a version that is no longer fully supported and never once demonstrates a difference in timing based on end-user experience which is the only timing that matters.

Re: Larger vs. Small data block
Posted: Jun 3, 2008 7:43 AM in response to: Amardeep Sidhu
Reply

Re: Larger vs. Small data block
Posted: Jun 3, 2008 7:50 AM in response to: damorgan
Reply

Richard Foote

Posts: 278
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 9:04 AM in response to:
Reply

Note: it doesn't matter how often one says something, it doesn't make it any more accurate or any less misleading ...

In this one post you quote from UNISYS:

"indexes (with small index entries) that are predominantly accessed via a matching key may benefit from a smaller DB_BLOCK_SIZE."

and then from the infamous metalink note you quote:

"Indexes like big blocks because index height can be lower and more space exists within the index branch nodes."

Ummm, consistent as always ;)

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Richard Foote

Posts: 278
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 9:19 AM in response to: Madrid
Reply

> Metalink notes are official Oracle statements, so
> they are considered the ultimate truth reference that
> overrules any ACE's, researcher's or whoever
> criteria. Just for the sake of truth, and based on
> your lab tests, would you mind making an official
> request with the metalink team for this note to be
> either removed or updated?
>

Hi Madrid

I worked for Oracle Corporation for a number of years from the mid 1990's. Guess what. Despite some rumors to the contrary, most people who work for Oracle are just ordinary, everyday folk like you and me. They generally don't have any special powers or abilities, surprisingly perhaps, they generally have access to little or no additional documentation or information that isn't generally available, they can make mistakes and incorrect assumptions on how things work and often look to people such a Jonathan Lewis or a Steve Adams for insights and information.

In short, you could possibly have a much experience and insight into Oracle as many of those who write some of these metalink notes.

It would be nice to think that official Oracle statements and documentation would be error free and totally 100% accurate. Unfortunately, the real-world and Oracle specifically isn't like that. Although on the whole, information you get from Oracle is pretty damn good, it has mistakes and errors just like any other source of information.

Cheers

Richard Foote

http://richardfoote.wordpress.com/

Richard Foote

Posts: 278
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 9:31 AM in response to:

Reply

Don't confuse the issue of deciding an appropriate database block size with that of having multiple block sizes within the same database ...

Again, for the umpteenth time, let me make the point that your general advise that one of the first things an experienced DBA should do is move all indexes into a larger block size and that indexes always favour large blocks contradicts entirely with what you have just quoted from metalink:

"Large block size is not good for index blocks used in an OLTP type environment, because they increase block contention on the index leaf blocks"

You can't keep having it both ways. You can't suggest one minute that the first thing one should do is rebuild indexes into a bigger block size because indexes always favour them and then provide quotes that directly and totally contradicts such advice.

It's not just beginners you are totally confusing :(

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Billy Verreynne

Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 10:14 AM in response to: Richard Foote

Reply

> It would be nice to think that official Oracle statements and documentation would be
> error free and totally 100% accurate. Unfortunately, the real-world and Oracle specifically
> isn't like that.

Can attest to that. Some years ago had a nasty I/O problem at kernel driver level that reported data corruption. Fortunately the actual data written to SAN was not corrupted.

Took me a few days and replicating the environment on another smaller platform and testing each and every layer in turn to find the problem. Oracle Support tried to help in a fashion, but without the right h/w and s/w combinations...

Turned out to be a multi I/O path issue dealing with ASMlib and certain vendor s/w - that according to a very specific Metalink note I've consulted prior to installation, should not have been any problem at all.

I requested (in my SR) that the Metalink note to be corrected, or at least contain a reference that what the note stated was incorrect for certain s/w combinations... nothing happened.

So yeah.. one needs to be very wary simply to treat a Metalink note as gospel.

mpowel01

Posts: 2,840
Registered: 12/8/98

Re: Larger vs. Small data block
Posted: Jun 3, 2008 10:22 AM in response to: Madrid

Reply

Madrid, metalink notes vary in quality. Many of the notes contain factual misstatements or have been superceded by newer notes but not removed from the system. If the information is important you should always attempt to verify it through testing and additional documentation research.

The above statement is in general and is not a comment on the validity of the note you referenced earlier. Just do not think that because you find it in a metalink note that the contents are automatically correct. Obsolete information and errors are not uncommon.

HTH -- Mark D Powell --

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 3:49 PM in response to: damorgan

Reply

Daniel,

I don't agree nor disagree, I am just looking for the truth. You said you have some lab tests, If you don't mind I would like to take a look at your research results. Have you published them in internet? Are they available?

Regards.

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 3:59 PM in response to: Billy Verreynne

Reply

> So yeah.. one needs to be very wary simply to treat a
> Metalink note as gospel.

IMO Metalink notes should be treated seriously, and in case someone has factual scientific evidence a note is wrong then that someone must report it to the Metalink team along with the research results. I don't think Oracle is willing to publish lies. And if metalink doesn't have any credibility at all, what's left, Search The Web? Forums?, Friends? Crystal balls?

~ Madrid

http://hrivera99.blogspot.com/

Amit_DBA

Posts: 503
From: Bangalore, India
Registered: 2/2/05

Re: Larger vs. Small data block
Posted: Jun 3, 2008 4:07 PM in response to: Madrid

Reply

Atleast Metalink articles with RAV status should be verified before they are followed. This warning also comes in the Note header.

Secondly as Richard Foote has clearly mentioned that there is no special documentation which is available to oracle people

(except for unpublished bugs)
They are Notes terming good and bad tips. But if you find people like Jonathan Lewis or Richard Foote proving them wrong with Real examples , then I believe we should acknowledge it.

-Amit
http://askoracledba.wordpress.com

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 4:18 PM in response to: Amit_DBA
Reply

I am perfectly aware of the RAV kind of notes, in fact I have provided Metalink with feedback when I have found the documents have inaccurate statements. But what about those notes which are already official. We cannot consider Metalink as unofficial, otherwise metalink would end up considered as a Grimorium Verum.

~ Madrid

http://hrivera99.blogspot.com/

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:18 PM in response to: Amit_DBA
Reply

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:23 PM in response to: Amit_DBA
Reply

sybrandb

Posts: 4,036
From: Amsterdam, Netherlands
Registered: 8/4/98

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:27 PM in response to: Madrid
Reply

I have reported several errors in the past in notes you consider to be 'official' ie non-RAV.
Please also acknowledge many Metalink Notes are about 10 years old.
As Tom Kyte always says 'Always question authority'

--
Sybrand Bakker
Senior Oracle DBA

sybrandb

Posts: 4,036
From: Amsterdam, Netherlands
Registered: 8/4/98

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:29 PM in response to:
Reply

I'm quite positive the only source of information for OTN support people is Metalink.
I almost never get responses to SRs which go beyond Metalink.
If I need such a response the SR is assigned to development.

--
Sybrand Bakker
Senior Oracle DBA

Amit_DBA

Posts: 503
From: Bangalore, India
Registered: 2/2/05

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:32 PM in response to:
Reply

I believe these are only for tips and tricks for resolving corruption issues or undocumented features..and not on Performance Test Results..

Anyways I Do agree that Metalink is THE place to get the right information. At the same time for some issues, we cant depend on it fully.

Didn't Oracle docs said that ASM balances Hot spots..But if you check book on ASM (Oracle press by Nitin Vengurekar,Murali...) it says it is a Myth.

-Amit
http://askoracledba.wordpress.com

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:35 PM in response to: sybrandb
Reply

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 3, 2008 5:51 PM in response to: damorgan
Reply

<PRE>

I created two basic identical databases on same server with same configuration, except db_block_size and db_file_multiblock_read_count. You can see the result below.

```
SQL> select * from v$version
2 /

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Prod
PL/SQL Release 10.2.0.4.0 - Production
CORE 10.2.0.4.0 Production
TNS for 32-bit Windows: Version 10.2.0.4.0 - Production
```

```
NLSRTL Version 10.2.0.4.0 - Production

SQL> select name from v$database
2 /

NAME
---------
DWDB

SQL> Select Name, Value
2 From v$parameter
3 Where Name In ('db_block_size', 'db_file_multiblock_read_count')
4 /

NAME VALUE
-------------------------------------------------------------------------------- ------
db_block_size 16384
db_file_multiblock_read_count 32

SQL> Explain Plan For
2 Select count(1)
3 From employee emp, department dept
4 Where emp.dept_code = dept.dept_code
5 /

Explained.

SQL> Select plan_table_output
2 From Table (Dbms_xplan.display ())
3 /

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------------
Plan hash value: 1228034791

---------------------------------------------------------------------------------------------------
| Id | Operation | Name | Rows | Bytes |TempSpc| Cost (%CPU)| Time |
---------------------------------------------------------------------------------------------------
| 0 | SELECT STATEMENT | | 1 | 26 | | 15748 (2)| 00:03:41 |
| 1 | SORT AGGREGATE | | 1 | 26 | | | |
|* 2 | HASH JOIN | | 5472K| 135M| 130M| 15748 (2)| 00:03:41 |
| 3 | INDEX FAST FULL SCAN| DEPARTMENT_ID01 | 5472K| 67M| | 1814 (2)| 00:00:26 |
| 4 | INDEX FAST FULL SCAN| EMPLOYEE_ID01 | 6331K| 78M| | 1814 (2)| 00:00:26 |
---------------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

2 - access("EMP"."DEPT_CODE"="DEPT"."DEPT_CODE")

Note
-----
- dynamic sampling used for this statement

20 rows selected.

SQL> Exit;


****************************************************************************************************

SQL> select * from v$version
2 /

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Prod
PL/SQL Release 10.2.0.4.0 - Production
CORE 10.2.0.4.0 Production
TNS for 32-bit Windows: Version 10.2.0.4.0 - Production
NLSRTL Version 10.2.0.4.0 - Production

SQL> select name from v$database
2 /

NAME
---------
TPDB

SQL> Select Name, Value
2 From v$parameter
3 Where Name In ('db_block_size', 'db_file_multiblock_read_count')
4 /

NAME VALUE
-------------------------------------------------------------------------------- -----
db_block_size 8192
db_file_multiblock_read_count 8

SQL> Explain Plan For
2 Select count(1)
3 From employee emp, department dept
4 Where emp.dept_code = dept.dept_code
5 /

Explained.

SQL> Select plan_table_output
2 From Table (Dbms_xplan.display ())
3 /

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------
Plan hash value: 1228034791

---------------------------------------------------------------------------------------------------
| Id | Operation | Name | Rows | Bytes |TempSpc| Cost (%CPU)| Time |
----------------------------------------------------------------------------------------------------
| 0 | SELECT STATEMENT | | 1 | 26 | | 19319 (2)| 00:03:52 |
| 1 | SORT AGGREGATE | | 1 | 26 | | | |
|* 2 | HASH JOIN | | 5293K| 131M| 126M| 19319 (2)| 00:03:52 |
| 3 | INDEX FAST FULL SCAN| DEPARTMENT_ID01 | 5293K| 65M| | 3226 (2)| 00:00:39 |
| 4 | INDEX FAST FULL SCAN| EMPLOYEE_ID01 | 5475K| 67M| | 3226 (2)| 00:00:39 |
----------------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

2 - access("EMP"."DEPT_CODE"="DEPT"."DEPT_CODE")

Note
-----
- dynamic sampling used for this statement

20 rows selected.

SQL> Exit

==========================================================================================================
```

You can clearly see the database TPDB with 8k blocksize took 10% more CPU time than DWDB with 16k blocksize.

Mr.damorgan, Can you explain for this 10% cost difference.

</PRE>

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 3, 2008 6:19 PM in response to: sybrandb

Reply

> I have reported several errors in the past in notes
> you consider to be 'official' ie non-RAV.

great for you, that is what it is all about. I did say if someone has evidence the note is false or missleading then it MUST be reported to the meatalink team, so it maintains its credibility. I have also reported notes which I have found to have mistakes.

> As Tom Kyte always says 'Always question authority'

That's right, you have to question, from the research point of view. Now what happens from the practical point of view in a day by day dba work?. Let's assume you are in a consulting service, or just a professional who works for a company and wants to resolve some issue, who would you give more credibility when the business availability depends on the information you gather, Metalink or a google search? Do you think there will be enough time to 'Question Authority' in this case?

~ Madrid

http://hrivera99.blogspot.com/

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 3, 2008 6:34 PM in response to: sp009

Reply

Database DWDB:

> db_block_size 16384
> db_file_multiblock_read_count 32

Cost of 2 tablescans and hash join is 15748, Time 3:41

Database TPDB

> db_block_size 8192
> db_file_multiblock_read_count 8
>
Cost of 2 tablescans and hash join is 19319, Time 3:52

>
> You can clearly see the database TPDB with 8k blocksize took 10% more
> CPU time than DWDB with 16k blocksize.
>

Neither database took any time to run the query - what you're looking at is execution plan which is the predicted cost and time to run.

Secondly, the 3:41 vs. 3:52 is the predicted ***elapsed*** time to run, how are you turning an 11 second difference in elapsed time into a 10% difference in CPU time ?

Thirdly, given you've told Oracle that it's allowed to read 32 blocks in a single read request for the 16K block size (for a total of 512K) why should you be surprised if the predicted runtime is longer when you tell Oracle that it can only read 8 blocks of 8Kb (for a total of 64K) in a single read request.

Fourth, most of the time shown relates to the temp space I/O due to a predicted hash join spill to disc. Unfortunately the variation in the time for the hash join line is affected by the difference in estimates of the sizes of the inputs: the temp space size for the larger block size is 4M (3%) bigger, but the time due to that line is 2:49 compared to the 2:34 (9% less) for the smaller block size. So nothing conclusive from that line - which happens to be the largest contributor to the predicted elapsed time.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Re: Larger vs. Small data block
Posted: Jun 3, 2008 7:34 PM in response to: Jonathan Lewis

Reply

---

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 3, 2008 9:20 PM in response to: Jonathan Lewis

Reply

Mr. Jonathan,

I am not arguing with you since you are far more knowledgeable than me. I oversee a medium size Oracle Shop and i can see that after switching to block size 16 and read count 32 (i got the opportunity to test first using our production data during server upgrade) , me and my DB users can see noticeable performance gain in our DW application. OK, the case may be different for Oracle Shop with Servers hosting different applications other than Oracle and with limited resources. The 10% difference i mentioned in above case is 15748 Vs 19319. As far as i know, there are hundreds of like queries executing in our DW DB in every 30 minutes. That make a noticeable performance difference in overall for 8K Vs 16K

Regards,
sp009

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 3, 2008 9:41 PM in response to: sp009

Reply

That the query is faster is not being questions. What is at issue is that you are drawing an unsupported inference.

The point I think Jonathan is making is that your test case does not prove what you are claiming it does. 16x32 <> 8x8. You have no evidence that the relevant factor was the block size and not the change in multi-block reads or any one of a number of other possible factors.

The lab test should look like this:
Test 1: Run test using 8K blocks.
Test 2: Run the exact same test changing NOTHING other than the block size.

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 3, 2008 9:43 PM in response to:

Reply

The most relevant portion of Jonathan's post was this short paragraph:

"... that it's allowed to read 32 blocks in a single read request for the 16K block size (for a total of 512K) why should you be surprised if the predicted runtime is longer when you tell Oracle that it can only read 8 blocks of 8Kb (for a total of 64K) in a single read request."

Is there some reason you don't attempt to address the obvious difference that renders the "test case" meaningless? Or is it your opinion that 512K = 64K? <g>

Hemant K Chitale

Posts: 1,259
Registered: 11/6/98

Re: Larger vs. Small data block
Posted: Jun 3, 2008 10:08 PM in response to: sp009

Reply

Your 16K block database has an mbrc of 32 but the 8K block database
has an mbrc of 8 only.
The Index Fast Full Scan does a multiblock read which in your 16K database
is 512KB but is only 64KB in the 8K dadtabase. Oracle realises that it will
have to issue more read calls to the OS, taking more time to do, in the 8K
database.

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 3, 2008 10:20 PM in response to: Hemant K Chitale

Reply

Exactly. That some here are capable of tuning it out is truly amazing.

I am copying parts of this thread onto slides for my class at the university next year. There is a lot to be learned from observing people that don't or can't.

Billy Verreynne

Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 4, 2008 1:16 AM in response to: Madrid

Reply

> I don't think Oracle is willing to publish lies

You are misconstruing what some of us are saying. Metalink articles are correct.. but only correct as far as their context - such as the Oracle version (and patchset) they refer to.

Even then, other factors like o/s version, h/w and so on can have an impact on the accuracy of that note.

Over time, these notes can get out of date. Oracle is always introducing core changes in the RDBMS kernel (never mind all the other new features) as the technologies evolves and matures. The product is not stagnant. Expecting Metalink notes to always be 100% correct and 100% applicable, is a very unrealistic expectation.

This is not "*Oracle publishing lies*". No-one has said that here - or even implied it. What has been said is that one should not treat Metalink articles as the sole and only truth.

> And if metalink doesn't have any credibility at all, what's left, Search The Web? Forums?, ..

Again, no-one has said that Metalink has no credibility. It is a resource. Like searching the web, consulting forums and so on.

>.. Friends? Crystal balls?

You mean like having this some person starting a forum posting "*Dear Oracle Friends*" as if we are all part of a brotherhood of the Oracle Religion? I broke my crystal ball on such a person's head, so I'm out of crystal balls.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 4, 2008 2:02 AM in response to: damorgan

Reply

> That the query is faster is not being questioned.
>

The speed of the query is not available for questioning - as far as the post from sp009 goes the query has not been run, so there are no figures whatsoever about the actual speed of the query. All we have seen is that if you change the input statistics and optimizer parameters for a query the execution plan costs can change.

> The lab test should look like this:
> Test 1: Run test using 8K blocks.
> Test 2: Run the exact same test changing NOTHING other than the block size.

Test 2 should change the db_file_multiblock_read_count size so that the product of block_size and db_file_multiblock_read_count does not change from the value in Test 1.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Hemant K Chitale

Posts: 1,259
Registered: 11/6/98

Re: Larger vs. Small data block
Posted: Jun 4, 2008 2:33 AM in response to: Billy Verreynne

Reply

>>I broke my crystal ball on such a person's head, so I'm out of crystal balls.

Sometimes, I am tempted to stop replying to any questions on this forum.
(I never had a crystal ball to begin with, you see).

Richard Foote

Posts: 278
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 4, 2008 6:04 AM in response to: sp009

Reply

Hi sp009

Some advice.

If you're going to compare benchmarks, make sure you actually **execute** the associated statements, not just get an explain plan.

Learn the difference between determining the **actual** cpu used by differing statements vs. the cpu the CBO **thinks** it might use, as they may or may not be totally and completely different.

If you're going to claim 10% **cpu time** differences, then make sure you actually quote correct figures (15748 vs 19319 are not "cpu times") .

If you're going to claim a **10%** difference, then make sure your arithmetic is correct (15748 vs 19319 is not a 10% difference).

Note that 2 databases on the same server are not going to necessarily be identical. For example, the associated data files, log files might live on faster disks or on faster parts of the disks, the server may be at differing loads at differing times etc.

Note that running a SQL statement (when you actually get around to running it of course) that uses a multiblock read execution plan but compares a 512K max read vs. a 64K max read is not the same thing, not even close really.

You've unfortunately made the classic mistake of changing 2 things (block size and MBRC) and assuming the net change is the result of just one of those changes (block size) when in actual fact the other change (the overall MBRC) is likely to have a greater impact.

A golden rule. If you don't compare an apple with an apple but instead compare an apple with an orange, you can't really complain too much if the orange isn't crunchy enough for you :)

Thank you very much for your contribution, it's an excellent lesson/warning for us all ...

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 4, 2008 6:07 AM in response to: Jonathan Lewis

Reply

> > That the query is faster is not being questioned.
> >
> The speed of the query is not available for
> questioning - as far as the post from sp009 goes the
> query has not been run, so there are no figures
> whatsoever about the actual speed of the query. All
> we have seen is that if you change the input
> statistics and optimizer parameters for a query the
> execution plan costs can change.
>
> Test 2 should change the
> db_file_multiblock_read_count size so that the
> product of block_size and
> db_file_multiblock_read_count does not change from
> the value in Test 1.

Jonathan,

I am probably forgetting something here, but as sp009's explain plan on Oracle 10.2.0.4 only shows the estimated time for data retrieval, would not the values in sys.aux_stats$ be more relevant to the estimated time for data retrieval than db_file_multiblock_read_count? I thought that on Oracle 10.2 CPU costing values would be used for estimated time, while db_file_multiblock_read_count will be used for actual data retrieval times.

References:
http://www.oracle.com/technology/pub/articles/lewis_cbo.html
http://jonathanlewis.wordpress.com/2007/05/20/system-stats-strategy/
http://www.jlcomp.demon.co.uk/system_stats.html

sp009's experiment, while more thorough and complete than others, did not report all information necessary to build a test case (as has already been stated a couple times in this thread). I would have liked to see the DBMS_XPLAN output for the query with ALL STATS LAST specified, a list of all initialization parameters, and the values in sys.aux_stats$. It might also have been nice to see a 10046 trace to see if the effects of block buffer caching, file system caching, or read-ahead optimization had any impact on actual execution performance.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Hemant K Chitale

Posts: 1,259
Registered: 11/6/98

Re: Larger vs. Small data block
Posted: Jun 4, 2008 6:12 AM in response to: Charles Hooper

Reply

>> I thought that on Oracle 10.2 CPU costing values would be used for estimated time, while db_file_multiblock_read_count will be used for actual data retrieval times.

Good reminder.
I believe that would be the case if System Statistics have been gathered (SYS.AUX_STATS$ is populated). But, I can't be sure ....

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 4, 2008 6:18 AM in response to: Charles Hooper

Reply

Hi Charles

If the DFMBRC is specified, it will still limit the max size of a multiblock read with 10g and system stats. The CBO will use the system stats in determining the **cost**, but Oracle will still use the DFMBRC to limit the **size** of the actual associated I/Os.

System stats are yet another thing that can differ between the 2 databases, assuming of course they've been collected. And I also agree that the information you've specified is somewhat important to determine what may or may differ between the 2 databases.

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 4, 2008 7:02 AM in response to: Richard Foote

Reply

*If the DFMBRC is specified, it will still limit the max size of a multiblock read with 10g and system stats. The CBO will use the system stats in determining the **cost**, but Oracle will still use the DFMBRC to limit the **size** of the actual associated I/Os.*

Hi Richard,

Thanks for the reply. What you stated above seems to confirm what I was attempting to imply in my previous post (I did not word my previous reply as well as I would have liked). As sp009's test case reported cost and estimated time (and not actual time), it would appear that the different values of DFMBRC did not further decrease the accuracy of the test case.

Perhaps the test case posted by sp009' should have stated "with a larger default block size, the calculated estimated cost for executing a query is different if ...".

Is it possible, given what damorgan has stated in this thread, that Oracle does not consistently calculate a query's estimated cost across changes in the default database block size (if that is the only change)?:
"The correct answer for blocks size is 8K because that is the ONLY size Oracle tests."

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Removed extra word in the last sentence.
Message was edited by:
Charles Hooper

Re: Larger vs. Small data block
Posted: Jun 4, 2008 8:14 AM in response to: Jonathan Lewis

Reply

---

cd

Posts: 4,585
From: Vienna, Austria
Registered: 9/8/98

Re: Larger vs. Small data block
Posted: Jun 4, 2008 9:07 AM in response to:

Reply

Quit whining. As long as you hide your expertise behind locked down live production sites, you'll always be second. Get used to it.

C.

Message was edited by:
cd

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 4, 2008 9:57 AM in response to: Charles Hooper

Reply

Hi Charles

I'm not sure I would call sp009's post a "test case" as such as it doesn't really test anything other than listing a couple of execution plans.

I'm not entirely sure what Daniel meant by his comment. A database with a larger block size will have costs relative to the block size. With the cpu cost model , the cost can basically be summarised as No. of single block reads x single block read time plus No. of multiblock reads x multiblock read time plus cpu cycles/cpu cycles per second all divided by the single block read time.

So on the multiblock part of the costings, by having a larger MBRC, you potentially decrease the number of MBR operations but increase the associated read times (MREADTIM).

If you have multi sized blocks in a database, things get a little confusing for the CBO with regard to the single block read costs as you can vary the number of single block reads but the SREADTIM becomes an averaged figure between all the blocksizes. Can't say I tested the possible consequences here ? Multiblock reads aren't such a problem as they get treated the same regardless of the block size.

Yet another reason perhaps to avoid multi sized blocks in a database if there weren't enough already.

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Re: Larger vs. Small data block
Posted: Jun 4, 2008 9:58 AM in response to: cd
Reply

Richard Foote
Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 4, 2008 10:05 AM in response to:
Reply

For someone who makes a nice living because of the short comings of Metalink and the documentation, you sure appear to run to them for verification and confirmation of your little theories at every possible opportunity.

Not sure who is the more dependant on one than the other ...

Cheers

Richard Foote
http://richardfoote.wordpress.com/

sp009
Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 4, 2008 11:06 AM in response to: damorgan
Reply

> That the query is faster is not being questions. What
> is at issue is that you are drawing an unsupported
> inference.
>
> The point I think Jonathan is making is that your
> test case does not prove what you are claiming it
> does. 16x32 <> 8x8. You have no evidence that the
> relevant factor was the block size and not the change
> in multi-block reads or any one of a number of other
> possible factors.
>
> The lab test should look like this:
> Test 1: Run test using 8K blocks.
> Test 2: Run the exact same test changing NOTHING
> other than the block size.

Here every one is forgetting the basic question. "Can we increase the performance in DW applications by increasing db_block_size?". If you argue the difference is b'cos 16x32 <> 8x8 or 512K = 64K?, then you are virtually agreeing that, performance matters by increasing the block size and read count.

The example i give again is identical database in same server (created using same script). All parameters same except for db_block_size. I did clean restart of both database and server (no excuse for data cache or network traffic or bang on server) and executed the following sql set in the server.

```
Script Executed
---------------
Select *
  From v$version
/
Select Name
  From v$database
/
Select Name, Value
  From v$parameter
 Where Name In ('db_block_size', 'db_file_multiblock_read_count')
/
Select Current_timestamp
  From Dual
/
Select Count (1)
  From employee
/
Select Current_timestamp
  From Dual
/
Select Count (1)
  From department
/
Select Current_timestamp
  From Dual
/
```

```
Select Count (1)
  From employee emp, department dept
 Where emp.dept_code = dept.dept_code
/
Select Current_timestamp
  From Dual
/
Explain Plan For
  Select Count(1)
  From employee emp, department dept
  Where emp.dept_code = dept.dept_code
/
Select plan_table_output
  From Table (Dbms_xplan.display ())
/

===============================================================================================


SQL> autotrace OFF
linesize 80
linesize 80
wrap : lines will be wrapped
Select *
  2    From v$version
  3  /

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Prod
PL/SQL Release 10.2.0.4.0 - Production
CORE        10.2.0.4.0   Production
TNS for 32-bit Windows: Version 10.2.0.4.0 - Production
NLSRTL Version 10.2.0.4.0 - Production

SQL> Select Name
  2    From v$database
  3  /

NAME
---------
TPDB

SQL> Select Name, Value
  2    From v$parameter
  3   Where Name In ('db_block_size', 'db_file_multiblock_read_count')
  4  /

NAME                                                                             VALUE
-------------------------------------------------------------------------------- -----
db_block_size                                                                    8192
db_file_multiblock_read_count                                                    8

SQL> Select Current_timestamp
  2    From Dual
  3  /

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
04-JUN-08 09.19.22.038000 AM -05:00

SQL> Select Count (1)
  2    From employee
  3  /

  COUNT(1)
----------
   5000000

SQL> Select Current_timestamp
  2    From Dual
  3  /

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
04-JUN-08 09.19.32.678000 AM -05:00

SQL> Select Count (1)
  2    From department
  3  /

  COUNT(1)
----------
   5000000

SQL> Select Current_timestamp
  2    From Dual
  3  /

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
04-JUN-08 09.19.45.600000 AM -05:00

SQL> Select Count (1)
  2    From employee emp, department dept
  3   Where emp.dept_code = dept.dept_code
  4  /

  COUNT(1)
----------
   5000000

SQL> Select Current_timestamp
  2    From Dual
  3  /

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
04-JUN-08 09.20.42.396000 AM -05:00
```

```
SQL> Explain Plan For
  2  Select Count(1)
  3  From employee emp, department dept
  4  Where emp.dept_code = dept.dept_code
  5  /

Explained.

SQL> Select plan_table_output
  2    From Table (Dbms_xplan.display ())
  3  /

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------
Plan hash value: 4001065367

--------------------------------------------------------------------------------------------------
| Id  | Operation              | Name            | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                 |     1 |    12 |       | 15183   (2)| 00:03:03 |
|   1 |  SORT AGGREGATE        |                 |     1 |    12 |       |            |          |
|*  2 |   HASH JOIN            |                 |  5004K|    57M|    85M| 15183   (2)| 00:03:03 |
|   3 |    INDEX FAST FULL SCAN| EMPLOYEE_ID01   |  5004K|    28M|       |  3260   (2)| 00:00:40 |
|   4 |    INDEX FAST FULL SCAN| DEPARTMENT_ID01 |  5012K|    28M|       |  3271   (2)| 00:00:40 |
--------------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("EMP"."DEPT_CODE"="DEPT"."DEPT_CODE")

16 rows selected.

SQL> exit

==================================================================================================

SQL> Select *
  2    From v$version
  3  /

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Prod
PL/SQL Release 10.2.0.4.0 - Production
CORE        10.2.0.4.0   Production
TNS for 32-bit Windows: Version 10.2.0.4.0 - Production
NLSRTL Version 10.2.0.4.0 - Production

SQL> Select Name
  2    From v$database
  3  /

NAME
---------
DWDB

SQL> Select Name, Value
  2    From v$parameter
  3   Where Name In ('db_block_size', 'db_file_multiblock_read_count')
  4  /

NAME                                                                             VALUE
-------------------------------------------------------------------------------- ------
db_block_size                                                                    16384
db_file_multiblock_read_count                                                    8

SQL> Select Current_timestamp
  2    From Dual
  3  /

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
04-JUN-08 09.21.31.068000 AM -05:00

SQL> Select Count (1)
  2    From employee
  3  /

  COUNT(1)
----------
   5000000

SQL> Select Current_timestamp
  2    From Dual
  3  /

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
04-JUN-08 09.21.37.474000 AM -05:00

SQL> Select Count (1)
  2    From department
  3  /

  COUNT(1)
----------
   5000000

SQL> Select Current_timestamp
  2    From Dual
  3  /

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
```

```
04-JUN-08 09.21.47.911000 AM -05:00

SQL> Select Count (1)
  2    From employee emp, department dept
  3   Where emp.dept_code = dept.dept_code
  4  /

  COUNT(1)
----------
   5000000

SQL> Select Current_timestamp
  2    From Dual
  3  /

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
04-JUN-08 09.22.37.004000 AM -05:00

SQL> Explain Plan For
  2  Select Count(1)
  3  From employee emp, department dept
  4  Where emp.dept_code = dept.dept_code
  5  /

Explained.

SQL> Select plan_table_output
  2    From Table (Dbms_xplan.display ())
  3  /

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------
Plan hash value: 4001065367

---------------------------------------------------------------------------------------------------
| Id  | Operation              | Name             | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                  |     1 |    12 |       | 11879   (2)| 00:02:47 |
|   1 |  SORT AGGREGATE        |                  |     1 |    12 |       |            |          |
|*  2 |   HASH JOIN            |                  |  4992K|    57M|    85M| 11879   (2)| 00:02:47 |
|   3 |    INDEX FAST FULL SCAN| EMPLOYEE_ID01    |  4992K|    28M|       |  2234   (2)| 00:00:32 |
|   4 |    INDEX FAST FULL SCAN| DEPARTMENT_ID01  |  4999K|    28M|       |  2236   (2)| 00:00:32 |
---------------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("EMP"."DEPT_CODE"="DEPT"."DEPT_CODE")

16 rows selected.

SQL> exit

====================================================================================================
```

From the above result, if you compare the execution result for each query and the plan result, it's clear that DWDB shows more performance than TPDB. I don't want to compare and explain each result set. You do your math and see the truth.

Mr. Richard-

>>If you're going to claim a 10% difference, then make sure your arithmetic is correct (15748 vs. 19319
>> is not a 10% difference).

My bad math (shame on me for having masters in math and computer science). Also even if you say "read document" million times, truth won't change

Mr. damorgan-

>>That the query is faster is not being questions. What is at issue is that you are drawing an unsupported
>>inference.

Oracle supports db_block_size from 2048 to 16384, at least for Windows (We confirmed with Support). Also, refer Doc#B10752-01 page 87. As a Lab expert can you show some thing similar, like my above example, which shows nothing is going to change related to performance after increasing the block size?

Thank you,
sp009

Message was edited by:
sp009

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 4, 2008 1:01 PM in response to: damorgan

Reply

> Exactly. That some here are capable of tuning it out
> is truly amazing.
>
> I am copying parts of this thread onto slides for my
> class at the university next year. There is a lot to
> be learned from observing people that don't or can't.

Here the tkprof result on both database for same query. Now it’s up to you test in

```
your lab and decide

Database :=TPDB
###############

TKPROF: Release 10.2.0.4.0 - Production on Wed Jun 4 11:46:24 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Trace file: tpdb_ora_428.trc
Sort options: default

********************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
********************************************************************************

Select Count(1)
From employee emp, department dept
Where emp.dept_code = dept.dept_code

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     12.51      21.54      38750      23490          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     12.51      21.54      38750      23490          0           1

Misses in library cache during parse: 0
Optimizer mode: FIRST_ROWS
Parsing user id: SYS

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=23490 pr=38750 pw=15285 time=21546363 us)
5000000   HASH JOIN  (cr=23490 pr=38750 pw=15285 time=29565227 us)
5000000    INDEX FAST FULL SCAN EMPLOYEE_ID01 (cr=11745 pr=11725 pw=0 time=525 us)(object id 51779)
5000000    INDEX FAST FULL SCAN DEPARTMENT_ID01 (cr=11745 pr=11725 pw=0 time=231 us)(object id 51780)

********************************************************************************

alter session set sql_trace=false


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0
Parsing user id: SYS



********************************************************************************

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        2      0.00       0.00          0          0          0           0
Execute      2      0.00       0.00          0          0          0           0
Fetch        2     12.51      21.54      38750      23490          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        6     12.51      21.54      38750      23490          0           1

Misses in library cache during parse: 0


OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        0      0.00       0.00          0          0          0           0
Execute      0      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        0      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0

    2  user  SQL statements in session.
    0  internal SQL statements in session.
    2  SQL statements in session.
********************************************************************************
Trace file: tpdb_ora_428.trc
Trace file compatibility: 10.01.00
Sort options: default

       1  session in tracefile.
       2  user  SQL statements in trace file.
       0  internal SQL statements in trace file.
       2  SQL statements in trace file.
       2  unique SQL statements in trace file.
      45  lines in trace file.
      33  elapsed seconds in trace file.
```

```
Database :=DWDB
###############


TKPROF: Release 10.2.0.4.0 - Production on Wed Jun 4 11:50:37 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Trace file: dwdb_ora_1484.trc
Sort options: default

********************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
********************************************************************************

Select Count(1)
From employee emp, department dept
Where emp.dept_code = dept.dept_code

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     12.00      20.28      19123      11596          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     12.00      20.28      19123      11596          0           1

Misses in library cache during parse: 0
Optimizer mode: FIRST_ROWS
Parsing user id: SYS

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=11596 pr=19123 pw=7560 time=20284142 us)
5000000   HASH JOIN  (cr=11596 pr=19123 pw=7560 time=19622027 us)
5000000    INDEX FAST FULL SCAN EMPLOYEE_ID01 (cr=5798 pr=5778 pw=0 time=484 us)(object id 47749)
5000000    INDEX FAST FULL SCAN DEPARTMENT_ID01 (cr=5798 pr=5778 pw=0 time=210 us)(object id 47750)

********************************************************************************

alter session set sql_trace=false


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0
Parsing user id: SYS



********************************************************************************

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        2      0.00       0.00          0          0          0           0
Execute      2      0.00       0.00          0          0          0           0
Fetch        2     12.00      20.28      19123      11596          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        6     12.00      20.28      19123      11596          0           1

Misses in library cache during parse: 0


OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        0      0.00       0.00          0          0          0           0
Execute      0      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        0      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0

    2  user  SQL statements in session.
    0  internal SQL statements in session.
    2  SQL statements in session.
********************************************************************************
Trace file: dwdb_ora_1484.trc
Trace file compatibility: 10.01.00
Sort options: default

       1  session in tracefile.
       2  user  SQL statements in trace file.
       0  internal SQL statements in trace file.
       2  SQL statements in trace file.
       2  unique SQL statements in trace file.
      51  lines in trace file.
      25  elapsed seconds in trace file.



############################################################################################
```

>>*I am copying parts of this thread onto slides for my*
>>*class at the university next year. There is a lot to*
>>*be learned from observing people that don't or can't.*

Probably you should rethink about that

sp009

Billy Verreynne

Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 4, 2008 1:13 PM in response to: sp009

Reply

> *Select Count (1) ...*

And that is a lot better and faster than a *count(*)* I presume?

I can make INSERTs on almost any table significantly faster on any Oracle database. Do not believe my claim? Well, you do a "heavy" insert on said table. Time it. I drop all indexes, constraints and triggers on the applicable table. You then repeat your "heavy" insert and time it. Now compare the times.

I guarantee a 90% success rate of very noticeable performance increase.

Performance tuning is not about focusing on a single thing and attempting to make that thing as fast as possible. It is not the faster F1 car that wins a F1GP race. It is about the complete car, how well it was setup for the track, the choice of tires, race tactics and how good the driver is.

It's the same in Oracle (where the driver is the application).

Sure, I can make your INSERTs freakingly fast. But at what cost to data integrity and queries?

So the question is.. what is the price to pay for this "improvement" in performance you've demonstrated?

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 4, 2008 3:18 PM in response to: Jonathan Lewis

Reply

Jonathan,

```
> > That the query is faster is not being questioned.
> >
>
> The speed of the query is not available for
> questioning - as far as the post from sp009 goes the
> query has not been run, so there are no figures
> whatsoever about the actual speed of the query. All
> we have seen is that if you change the input
> statistics and optimizer parameters for a query the
> execution plan costs can change.
>
> > The lab test should look like this:
> > Test 1: Run test using 8K blocks.
> > Test 2: Run the exact same test changing NOTHING
> other than the block size.
>
> Test 2 should change the
> db_file_multiblock_read_count size so that the
> product of block_size and
> db_file_multiblock_read_count does not change from
> the value in Test 1.
>
> Regards
> Jonathan Lewis
> http://jonathanlewis.wordpress.com
> http://www.jlcomp.demon.co.uk
```

In my latest example nothing got changed between database or query execution plan except db_block_size.

>>Test 2 should change the db_file_multiblock_read_count size so that the product of block_size and
>>db_file_multiblock_read_count does not change from the value in Test 1.

Are you are saying, in order to justify the theory "db_block_size will not change performance in data warehousing applications", you should decrease db_file_multiblock_read_count, in case you increase db_block_size???

Oracle never says db_block_size * db_file_multiblock_read_count should be same across different Database and Platforms. If it does, please point documentation in that.

Regards
sp009

mpowel01

Posts: 2,840
Registered: 12/8/98

Re: Larger vs. Small data block
Posted: Jun 4, 2008 3:31 PM in response to: sp009

Reply

sp009, if I was trying to compare the results of the same query when one database had an 8k block size and another had a 32k block size I would want the db_file_multiblock_read_count X db_block_size to equal the same size IO otherwise the difference in performance may be due to the difference in IO size and not due to the difference in block size itself.

HTH -- Mark D Powell --

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 4, 2008 3:57 PM in response to: Charles Hooper

Reply

>
> I am probably forgetting something here, but as
> sp009's explain plan on Oracle 10.2.0.4 only shows
> the estimated time for data retrieval, would not the
> values in sys.aux_stats$ be more relevant to the
> estimated time for data retrieval than
> db_file_multiblock_read_count? I thought that on
> Oracle 10.2 CPU costing values would be used for
> estimated time, while db_file_multiblock_read_count
> will be used for actual data retrieval times.
>

Charles,
I didn't mention system statistics because it was clear from his example that sp009 was using the default values (10 m/s seek time and 4K per m/s transfer rate) as the conversion factor from cost to time for the 8K plan was 12m/s and the conversion factor for the 16K plan was 14m/s).

Since sp009 has set db_file_multiblock_read_count in both cases, the value supplied would have been used as the MBRC.

Updated: I forgot to comment on the 'allstats last' option with dbms_xplan.display_cursor(). It's quite useful, but it can add a huge overhead when enabled with 100% sample rate - so much so that the query runtime beomes completely meaningless. So it's one of those things that you might look at and then discard because the measurement effect outweighs the difference you are trying to measure.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 4, 2008 4:05 PM in response to: sp009

Reply

> Oracle never says db_block_size *
> db_file_multiblock_read_count should be same
> across different Database and Platforms. If it does,
> please point documentation in that.
>
> Regards
> sp009

sp009,

Try this search:
http://www.oracle.com/pls/db111/search?remark=quick_search&word=+db_file_multiblock_read_count&partno=

From:
http://download.oracle.com/docs/cd/B28359_01/server.111/b28313/usingpe.htm#sthref1646
"The recommended value for this parameter is eight for 8 KB block size, or four for 16 KB block size. The default is 8. This parameter determines how many database blocks are read with a single operating system READ call. The upper limit for this parameter is platform-dependent. If you set DB_FILE_MULTIBLOCK_READ_COUNT to an excessively high value, your operating system will lower the value to the highest allowable level when you start your database. In this case, each platform uses the highest value possible. Maximum values generally range from 64 KB to 1 MB."

From:
http://download.oracle.com/docs/cd/B28359_01/server.111/b32009/appa_aix.htm#BEHIIECG
"Set this parameter so that its value when multiplied by the value of the DB_BLOCK_SIZE parameter produces a number larger than the Logical Volume Manager stripe size. Such a setting causes more disks to be used."

From:
http://download-uk.oracle.com/docs/cd/B28359_01/server.111/b28320/initparams053.htm
"As of Oracle Database 10g release 2, the default value of this parameter is a value that corresponds to the maximum I/O size that can be performed efficiently. This value is platform-dependent and is 1MB for most platforms.Because the parameter is expressed in blocks, it will be set to a value that is equal to the maximum I/O size that can be performed efficiently divided by the standard block size. Note that if the number of sessions is extremely large the multiblock read count value is decreased to avoid the buffer cache getting flooded with too many table scan buffers."
"The maximum value is the operating system's maximum I/O size expressed as Oracle blocks ((max I/O size)/DB_BLOCK_SIZE). If you set this parameter to a value greater than the maximum, Oracle uses the maximum."

From:
http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/optimops.htm#BABDECGJ
"DB_FILE_MULTIBLOCK_READ_COUNT: This parameter specifies the number of blocks that are read in a single I/O during a full table scan or index fast full scan. The optimizer uses the value of DB_FILE_MULTIBLOCK_READ_COUNT to cost full table scans and index fast full scans. Larger values result in a cheaper cost for full table scans and can result in the optimizer choosing a full table scan over an index scan. If this parameter is not set explicitly (or is set is 0), the optimizer will use a default value of 8 when costing full table scans and index fast full scans."

From:
http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/stats.htm#sthref1191
"In release 10.2, the optimizer uses the value of mbrc when performing full table scans (FTS). The value of db_file_multiblock_read_count is set to the maximum allowed by the operating system by default. However, the optimizer uses mbrc=8 for costing. The "real" mbrc is actually somewhere in between since serial multiblock read requests are processed by the buffer cache and split in two or more requests if some blocks are already pinned in the buffer cache, or when the segment size is smaller than the read size. The mbrc value gathered as part of workload statistics is thus useful for FTS estimation.
During the gathering process of workload statistics, it is possible that mbrc and mreadtim will not be gathered if no table scans are performed during serial workloads, as is often the case with OLTP systems. On the other hand, FTS occur frequently on DSS systems but may run parallel and bypass the buffer cache. In such cases, sreadtim will still be gathered since index lookup are performed using the buffer cache. If Oracle cannot gather or validate gathered mbrc or mreadtim, but has gathered sreadtim and cpuspeed, then only sreadtim and cpuspeed will be used for costing. FTS cost will be computed using analytical algorithm implemented in previous releases. Another alternative to computing mbrc and mreadtim is to force FTS in serial mode to allow the optimizer to gather the data."

It looks like the documentation does suggest that db_block_size * db_file_multiblock_read_count should be considered.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 4, 2008 4:09 PM in response to: sp009

Reply

```
>
> Are you are saying, in order to justify the theory
> "db_block_size will not change performance
> in data warehousing applications", you should
> decrease db_file_multiblock_read_count, in case you
> increase db_block_size???
>
> Oracle never says db_block_size *
> db_file_multiblock_read_count should be same
> across different Database and Platforms. If it does,
> please point documentation in that.
>
```

If the purpose of your testing is an intelligent examination of the effects of different block sizes, then you should certainly be aware of the significance of the relationship between the block size and the multiblock read count.

If you wish to think otherwise then the logic of your argument suggests that you would advise someone to rebuild their database before suggesting that they try increasing the multiblock read count.

Having said that, though, I would like to point out that you are using 10.2.0.4 - and the suggestion from Oracle is that you don't set the ***db_file_multiblock_read_count*** at all in 10g. As it is, you've picked a fairly arbitrary value that happens to introduce an unfair bias in the 8K test.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 4, 2008 4:13 PM in response to: sp009

Reply

sp009,

Much better; however, given the interest in performance, it would have been helpful to run the trace at level 8 and including the wait summary so that we could see where the wait time went - the number, type, and average length of the waits could be very informative.

If you feel like running the test again, please remember the significance of the db_file_multiblock_read_count.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 4, 2008 4:18 PM in response to: sp009

Reply

```
> i can see that after switching to block size
> 16 and read count 32 ( i got the opportunity to test
> first using our production data during server
> upgrade) , me and my DB users can see noticeable
> performance gain in our DW application.
```

So you've moved your production database to a new server - and to many people that would probably suggest:
New CPUs - does that mean faster with a larger cache
New bus - does that mean faster
New memory - does that mean faster, and more
New HBA to link you to the disk storage
New network cards to link you to the end-users

You've copied all the data from one database to another
which may have dealt with some cleanout overheads,
and may have eliminated empty space and packed the data better
and will have moved the data to a different part of the disk array
(maybe it's even a new disk array to go with the new server)

You've changed the block size

Perhaps you also changed the multiblock read count - as your test suggests.

When the users say the system is running faster - how can you be so confident that the improvement is due to the change from 8K to 16K ?

.
Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 4, 2008 4:24 PM in response to:

Reply

```
> >> Unfortunately the variation in the time for the
> hash join line is affected by the difference in
> estimates of the sizes of the inputs:
>
> OK . . . .
>
> So you say that this is yet another reason why
> artificially contrived test cases are invalid when we
> are proving performance issues, right?
>
> For once, we agree!
>
```

> When we are talking database-wide performance
> "proof", respesentative benches are the ONLY way to
> predict the performance benefits of different
> blocksizes , IMHO. When you scale the "small"
> improvements with different blocksizes to systems
> with thousands of concurrent transactions, it can
> make a big difference, for my clients anyway. . .

I've simply described why there is a difference between the costs calculated for
the same execution plan on two different systems, yet you seem to think that this
is confirmation of one of your pet theories about run-time activity.

Let me demonstrate, through an analogy, what this tells us about your understanding of cost-based optimsation:

Me: "Your road map was printed in 2001 so it doesn't show the M6 toll road".
You: "Good, so you agree that we have to drive from London to Birmingham to see how long it will take to get there."

Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

---

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 4, 2008 4:47 PM in response to: Charles Hooper

Reply

Charles

>>It looks like the documentation does suggest that db_block_size * db_file_multiblock_read_count
>> should be considered.

Good point. It should be considered when you setup the stripe depth of your I/O based on OLTP or
DSS environment

Here topic is "Can we increase performance in DW applications by increasing db_block_size" ?

Of course yes. How?

Maximize db_block_size (2k - 16k) and in tern maximize the I/O request, db_block_size * db_file_multiblock_read_count
(Maximum db_file_multiblock_read_count depends on OS)

My intention is to prove, in low-concurrency DSS environment, increasing the db_block_size benefits
(Of course db_file_multiblock_read_count should be a candidate too) to make less number of I/O
data request and hence increasing the performance.

Regards,
sp009

Alvaro Buitrago

Posts: 17
From: Cali
Registered: 3/25/08

Re: Larger vs. Small data block
Posted: Jun 4, 2008 5:04 PM in response to: user619401

Reply

Sp009

You don't have probe anything
The correct test would be:
1. block size = 8k and multiread = 8, versus block size = 8, multiread = 32
2. block size = 16k and multiread = 8 versus block size = 16k multiread = 32
3. block size = 8k and multiread = 64 versus block size = 16k multiread = 32

So you can compare the impact of block size versus th impact of multiread

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 4, 2008 7:53 PM in response to: Alvaro Buitrago

Reply

xxx's alter ego is not interested in a correct test. He is interested in defending a hopelessly flawed position.

Even a first-year IT students knows that to determine the impact of a change to a system ... you change one, and only one,
parameter at a time.

Re: Larger vs. Small data block
Posted: Jun 4, 2008 8:14 PM in response to: damorgan

Reply

---

jgarry

Posts: 128
From: Just outside of
beautiful Vista, California
Registered: 7/20/98

Re: Larger vs. Small data block
Posted: Jun 4, 2008 8:20 PM in response to: sp009

Reply

> TNS for 32-bit Windows: Version 10.2.0.4.0 - Production

I'm wondering if there is something in Windows itself that is optimized for 8K.

---

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 4, 2008 8:25 PM in response to: Jonathan Lewis

Reply

Mr. Jonathan,

>>Much better; however, given the interest in performance, it would have been helpful to run the trace at level 8

I would definitely get a try as per your request and will let you know the tkprof result soon as possible.

Regards,
sp009

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 4, 2008 10:07 PM in response to: damorgan

Reply

> xxx's alter ego is not interested in a correct test.
> He is interested in defending a hopelessly flawed
> position.
>
> Even a first-year IT students knows that to determine
> the impact of a change to a system ... you change
> one, and only one, parameter at a time.

Mr. Damorgan,

I would like to quote your words again,

>>That the query is faster is not being questions. What is at issue is that you are
>>drawing an unsupported inference.

>>The point I think Jonathan is making is that your test case does not prove what you
>>are claiming it does. 16x32 <> 8x8. You have no evidence that the relevant factor was
>>the block size and not the change in multi-block reads or any one of a number of
>>other possible factors.

>>The lab test should look like this:
>>Test 1: Run test using 8K blocks.
>>Test 2: Run the exact same test changing NOTHING other than the block size.

If you can't compete with my test case or if you or your students failed to create
a contrary test case, then i would encourage you to stop promoting troll and accept the fact.

Thank you,
sp009

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 4, 2008 10:10 PM in response to: Jonathan Lewis

Reply

Jonathan,

Thanks for the response. I did not identify the clues that system statistics had not been collected, but I suspected that absence of system statistics might have been the case for at least one of the test runs.

The last two quotes that I provided from the Oracle documentation (at roughly the time of your response) seemed to conflict with one another. The second to the last quote essentially reinforces/restates your comments about the effects of sp009's setting of db_file_multiblock_read_count affecting the estimated cost of a query. The final quote from the documentation states something a bit different: "the optimizer uses mbrc=8 for costing", assuming I read correctly, if sreadtim and cpuspeed statistics are not both collected.

I have a bit more reading to do before I fully understand the logic. Thanks again to you and Richard Foote for helping to clear up the misunderstandings.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 4, 2008 10:57 PM in response to: sp009

Reply

You don't have any facts xxx as has been pointed out by all the real people in this thread.

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 1:42 AM in response to:

Reply

Instead of just citing (over and over) that TPC-C uses multiple blocksizes, perhaps you could explain and educate where (what objects) and why (what performance advantage it gives) the specific sizes are used.

Also, if you have any experiments (test cases) that support your position and that quantify the possible gains, it would strengthen your argument.

--
Regards,
Greg Rahn
http://structureddata.org

Greg Rahn

Posts: 61
From: Redwood Shores, California

Re: Larger vs. Small data block
Posted: Jun 5, 2008 2:31 AM in response to: sp009

Reply

> Here topic is "Can we increase performance in DW applications by increasing db_block_size" ?
>

Registered: 10/3/07

> Of course yes. How?

The answer is **both** Yes and No. Perhaps it may be done by increasing db_block_size, but it can also be done without, which raises the question: Does blocksize matter for table scans? I'll get there in a bit.

> Maximize db_block_size (2k - 16k) and in tern
> maximize the I/O request, db_block_size * db_file_multiblock_read_count
> (Maximum db_file_multiblock_read_count depends on OS)

As you have correctly stated, the I/O request size equation is:
db_block_size * db_file_multiblock_read_count = I/O size (max)

> My intention is to prove, in low-concurrency DSS environment, increasing the db_block_size benefits
> (Of course db_file_multiblock_read_count should be a candidate too) to make less number of I/O
> data request and hence increasing the performance.

Given the above equation for I/O size there are ***two*** variables that influence the I/O size and one of them *is not* block size.

This brings up the question: Why change block size when you can get the benefit of the maximum read size (1MB) and not have to worry about potentially changing index access plans to FTS plans because of costing issues?

Think of it like this: If I grab $100 from a bucket of coins given these rules:
- with each grab, exactly $1 is retrieved
- the same denomination of coin is always retrieved for a given "run"
- the time to complete the task is only related to the number of grabs, not the number of coins obtained

Regardless of the denomination of the coins grabbed, I need to grab 100 times. I could grab 4 quarters, or 10 dimes or 20 nickels or 100 pennies and each grab "performs" the same.

To demonstrate my claim, I will create an experiment (test case). I am also going to add to my claim that no matter what the blocksize, I can get the same read performance.

The experiment:
- 4 identical tables, with block sizes of 2k, 4k, 8k and 16k
- db_file_multiblock_read_count will be unset, letting Oracle choose the best size
- cold cache so forcing physical reads
- ASM storage, so no file system cache
- query will be: select * from table

The question: Does blocksize have any impact on elapsed time for a FTS query with 100% physical I/Os?

For the data in my table I'm going to use the WEB_RETURNS (SF=100GB) table from the TPC-DS. The flat file is 1053529104 bytes (as reported from ls).

```
create tablespace tpcds_8k  datafile '+GROUP1' size 1500m;
create tablespace tpcds_2k  datafile '+GROUP1' size 1500m blocksize 2k;
create tablespace tpcds_4k  datafile '+GROUP1' size 1500m blocksize 4k;
create tablespace tpcds_16k datafile '+GROUP1' size 1500m blocksize 16k;

create table web_returns_8k  tablespace tpcds_8k  as select * from web_returns_et;
create table web_returns_2k  tablespace tpcds_2k  as select * from web_returns_et;
create table web_returns_4k  tablespace tpcds_4k  as select * from web_returns_et;
create table web_returns_16k tablespace tpcds_16k as select * from web_returns_et;

select segment_name, sum(bytes)/1024/1024 mb from user_segments group by segment_name order by 2;

SEGMENT_NAME                 MB
-------------------- ----------
WEB_RETURNS_16K             880
WEB_RETURNS_8K                            896
WEB_RETURNS_4K                            920
WEB_RETURNS_2K                            976

SQL> desc WEB_RETURNS_16K
 Name                                                              Null?    Type
 ----------------------------------------- -------- ----------------------------
 WR_RETURNED_DATE_SK                                                NUMBER(38)
 WR_RETURNED_TIME_SK                                                NUMBER(38)
 WR_ITEM_SK                                                         NUMBER(38)
 WR_REFUNDED_CUSTOMER_SK                               NUMBER(38)
 WR_REFUNDED_CDEMO_SK                                               NUMBER(38)
 WR_REFUNDED_HDEMO_SK                                               NUMBER(38)
 WR_REFUNDED_ADDR_SK                                                NUMBER(38)
 WR_RETURNING_CUSTOMER_SK                              NUMBER(38)
 WR_RETURNING_CDEMO_SK                                              NUMBER(38)
 WR_RETURNING_HDEMO_SK                                              NUMBER(38)
 WR_RETURNING_ADDR_SK                                               NUMBER(38)
 WR_WEB_PAGE_SK                                                     NUMBER(38)
 WR_REASON_SK                                                                    NUMBER(38)
 WR_ORDER_NUMBER                                                    NUMBER(38)
 WR_RETURN_QUANTITY                                                 NUMBER(38)
 WR_RETURN_AMT                                                                   NUMBER(7,2)
 WR_RETURN_TAX                                                                   NUMBER(7,2)
 WR_RETURN_AMT_INC_TAX                                              NUMBER(7,2)
 WR_FEE                                                             NUMBER(7,2)
 WR_RETURN_SHIP_COST                                                NUMBER(7,2)
 WR_REFUNDED_CASH                                                   NUMBER(7,2)
 WR_REVERSED_CHARGE                                                 NUMBER(7,2)
 WR_ACCOUNT_CREDIT                                                  NUMBER(7,2)
 WR_NET_LOSS                                                        NUMBER(7,2)
```

I'm using a pro*c program to fetch the rows with an array size of 100. This way I don't have to worry about spool space, or overhead of SQL*Plus formatting.

Output from a run is such:

```
BEGIN_TIMESTAMP   QUERY_FILE                       ELAPSED_SECONDS ROW_COUNT
----------------- -------------------------------- --------------- ----------
20080604 22:22:19 2.sql                                 125.696083    7197670
20080604 22:24:25 4.sql                                 125.439680    7197670
20080604 22:26:30 8.sql                                 125.502804    7197670
20080604 22:28:36 16.sql                                125.251398    7197670
```

As you can see, no matter what the block size, the execution is the same (discounting fractions of a second).

```
The TKPROF output:

TKPROF: Release 11.1.0.6.0 - Production on Wed Jun 4 22:35:07 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Trace file: v11_ora_12162.trc
Sort options: default

********************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
********************************************************************************

/* 2.sql */
select * from web_returns_2k

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch    71978     25.39      26.42     493333     560355          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71980     25.39      26.42     493333     560355          0     7197670

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 50

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS_2K (cr=560355 pr=493333 pw=493333 time=88067 us cost=96149 size=770150690
card=7197670)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71980        0.00          0.16
  SQL*Net message from client                 71980        0.00         93.20
  db file sequential read                         3        0.00          0.01
  direct path read                             1097        0.04          0.13
  SQL*Net more data to client                 71976        0.00          1.88
********************************************************************************

/* 4.sql */
select * from web_returns_4k

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        2      0.00       0.00          0          0          0           0
Execute      2      0.00       0.03          0          0          0           0
Fetch    71978     24.98      25.92     232603     302309          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71982     24.98      25.96     232603     302309          0     7197670

Misses in library cache during parse: 0
Parsing user id: 50

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS_4K (cr=302309 pr=232603 pw=232603 time=84876 us cost=51644 size=770150690
card=7197670)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71981        0.00          0.15
  SQL*Net message from client                 71981        0.00         93.19
  db file sequential read                         2        0.00          0.01
  direct path read                             1034        0.02          0.19
  SQL*Net more data to client                 71976        0.00          1.85
  rdbms ipc reply                                 1        0.03          0.03
********************************************************************************

/* 8.sql */
select * from web_returns_8k

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        2      0.00       0.00          0          0          0           0
Execute      2      0.00       0.01          0          0          0           0
Fetch    71978     24.61      25.71     113157     183974          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71982     24.61      25.73     113157     183974          0     7197670

Misses in library cache during parse: 0
Parsing user id: 50

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS_8K (cr=183974 pr=113157 pw=113157 time=85549 us cost=31263 size=770150690
card=7197670)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71981        0.00          0.15
  SQL*Net message from client                 71981        0.00         93.32
  db file sequential read                         1        0.01          0.01
  direct path read                              999        0.01          0.17
  SQL*Net more data to client                 71976        0.00          1.83
  rdbms ipc reply                                 1        0.01          0.01
********************************************************************************

/* 16.sql */
select * from web_returns_16k

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch    71978     24.74      25.59      55822     127217          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71980     24.74      25.59      55822     127217          0     7197670

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 50

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS_16K (cr=127217 pr=55822 pw=55822 time=82996 us cost=21480 size=770150690 card=7197670)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71980        0.00          0.15
  SQL*Net message from client                 71980        0.00         93.39
  db file sequential read                         1        0.00          0.00
  direct path read                              981        0.01          0.16
  SQL*Net more data to client                 71976        0.00          1.84
********************************************************************************
```

Why is this? Because Oracle is optimizing the multi block read count automatically.

```
select FILE_ID,TABLESPACE_NAME from dba_data_files where TABLESPACE_NAME like 'TPC%'

   FILE_ID TABLESPACE_NAME
---------- ------------------------------
        16 TPCDS_8K
        17 TPCDS_2K
        18 TPCDS_4K
        19 TPCDS_16K

2k: WAIT #2: nam='direct path read' ela= 37 file number=17 first dba=33280 block cnt=512 obj#=55839 tim=1212643347820647
4k: WAIT #2: nam='direct path read' ela= 33 file number=18 first dba=16640 block cnt=256 obj#=55840 tim=1212643474070675
8k: WAIT #1: nam='direct path read' ela= 30 file number=16 first dba=8320  block cnt=128 obj#=55838 tim=1212643599631927
16k:WAIT #2: nam='direct path read' ela= 39 file number=19 first dba=55040 block cnt=64  obj#=55841 tim=1212643838893785
```

The raw trace file show us that reads are optimized to 1MB. For example, with a 2k block, 512 blocks are read at a time.

So what does this experiment show us?
In cases where MBRC kicks in, it actually is **NOT** the blocksize that really matters, but the **read size of the I/O.** More importantly, the Oracle database can decide the optimal MBRC no matter what the blocksize, demonstrating there is no advantage to a larger (or even smaller) blocksize in this case.

--
Regards,
Greg Rahn
http://structureddata.org

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 3:45 AM in response to: Charles Hooper

Reply

Charles,

In response to your most recent posting I have gone back to review the extracts you picked out from the Oracle manuals - noting that they were all from the 11g manuals.

It's an interesting collection that demonstrates two things:
a) There is a need to consider db_file_multiblock_read_count in conjunction with db_block_size.
b) The manuals start with some errors built in, and then get out of date

>
> From:
> http://download.oracle.com/docs/cd/B28359_01/server.11
> 1/b28313/usingpe.htm#sthref1646
> "The recommended value for this parameter is eight
> for 8 KB block size, or four for 16 KB block size.
> The default is 8. This parameter determines how many
> database blocks are read with a single operating
> system READ call. The upper limit for this parameter
> is platform-dependent. If you set
> DB_FILE_MULTIBLOCK_READ_COUNT to an excessively high
> value, your operating system will lower the value to
> the highest allowable level when you start your
> database. In this case, each platform uses the
> highest value possible. Maximum values generally
> range from 64 KB to 1 MB."
>

This should have been changed dramatically after 9i - the recommendation from 10g is to leave the parameter unset and let Oracle work things out for itself. Technically it's not the operating system that lowers the value - Oracle negotiates with the O/S to discover the largest O/S read size and Oracle lowers the value.

> From:
> http://download.oracle.com/docs/cd/B28359_01/server.11
> 1/b32009/appa_aix.htm#BEHIIECG

> "Set this parameter so that its value when multiplied
> by the value of the DB_BLOCK_SIZE parameter produces
> a number larger than the Logical Volume Manager
> stripe size. Such a setting causes more disks to be
> used."
>

As above - it's a comment that should have been wiped from the 10g manuals.

> From:
> http://download-uk.oracle.com/docs/cd/B28359_01/server
> .111/b28320/initparams053.htm
> "As of Oracle Database 10g release 2, the default
> value of this parameter is a value that corresponds
> to the maximum I/O size that can be performed
> efficiently. This value is platform-dependent and is
> 1MB for most platforms.Because the parameter is
> expressed in blocks, it will be set to a value that
> is equal to the maximum I/O size that can be
> performed efficiently divided by the standard block
> size. Note that if the number of sessions is
> extremely large the multiblock read count value is
> decreased to avoid the buffer cache getting flooded
> with too many table scan buffers."
> "The maximum value is the operating system's maximum
> I/O size expressed as Oracle blocks ((max I/O
> size)/DB_BLOCK_SIZE). If you set this parameter to a
> value greater than the maximum, Oracle uses the
> maximum."
>

Succinct, covers all the important points in a well ordered manner.
You could argue that it should tell you what happens when you use a non-standard block size, but the explanation of how the value is derived gives you a good idea of how to make an intelligent guess - which makes it a good example of how to avoid adding excess details that might distract novices while ensuring that more experience readers still get good information. It doesn't say anything about what impact this setting might have on costing - but presumably that's not considered relevant at this point of the manual.

> From:
> http://download.oracle.com/docs/cd/B28359_01/server.11
> 1/b28274/optimops.htm#BABDECGJ
> "DB_FILE_MULTIBLOCK_READ_COUNT: This parameter
> specifies the number of blocks that are read in a
> single I/O during a full table scan or index fast
> full scan. The optimizer uses the value of
> DB_FILE_MULTIBLOCK_READ_COUNT to cost full table
> scans and index fast full scans. Larger values result
> in a cheaper cost for full table scans and can result
> in the optimizer choosing a full table scan over an
> index scan. If this parameter is not set explicitly
> (or is set is 0), the optimizer will use a default
> value of 8 when costing full table scans and index
> fast full scans."
>

There are various examples of poor wording and ambiguity in the explanations in this section, but most significantly, it went out of date at 9i and should have underfone a massive rewrite then. The last line is particularly bad - I'd have to go back and check earlier versions, but the last time I checked 10.2 the run-time engine used a value of 1 if you set the parameter to zero (this may have been a change that arrived with CPU costing) so if anyone reads and follows this advice in 10.2 (and a couple of people on this forum have, already) then they can run into problems with insane execution plans.

> From:
> http://download.oracle.com/docs/cd/B28359_01/server.11
> 1/b28274/stats.htm#sthref1191
> "In release 10.2, the optimizer uses the value of
> mbrc when performing full table scans (FTS). The
> value of db_file_multiblock_read_count is set to the
> maximum allowed by the operating system by default.
> However, the optimizer uses mbrc=8 for costing. The
> "real" mbrc is actually somewhere in between since
> serial multiblock read requests are processed by the
> buffer cache and split in two or more requests if
> some blocks are already pinned in the buffer cache,
> or when the segment size is smaller than the read
> size. The mbrc value gathered as part of workload
> statistics is thus useful for FTS estimation.
> During the gathering process of workload statistics,
> it is possible that mbrc and mreadtim will not be
> gathered if no table scans are performed during
> serial workloads, as is often the case with OLTP
> systems. On the other hand, FTS occur frequently on
> DSS systems but may run parallel and bypass the
> buffer cache. In such cases, sreadtim will still be
> gathered since index lookup are performed using the
> buffer cache. If Oracle cannot gather or validate
> gathered mbrc or mreadtim, but has gathered sreadtim
> and cpuspeed, then only sreadtim and cpuspeed will be
> used for costing. FTS cost will be computed using
> analytical algorithm implemented in previous
> releases. Another alternative to computing mbrc and
> mreadtim is to force FTS in serial mode to allow the
> optimizer to gather the data."
>

The opening statement is wrong - Oracle uses the value of the ***MBRC*** statistic when calculating the ***cost*** of performing the full tablescan (or index fast full scan). The whole thing is an example of writing that will not help the novice reader understand how things work - and I'm not sure that the note is correct in its description of how the optimizer responds to incomplete system stats.

It's an interesting point that from 10.2 onwards the ***MBRC*** is supposed to default to 8 if you haven't set the ***db_file_mulitblock_read_count***. (Technically, it's the ***_db_file_optimizer_read_count*** that defaults to 8 and then the ***MBRC*** copies the parameter).

You might wonder if this is a setting that is actually dependent on the ***block_size***. The last time I checked on a system with 16K blocks, though, it wasn't different - the value really does seem to be fixed at 8. This means that if you've allowed the ***db_file_mulitblock_read_count*** and system statistics to default, the optimizer will favour tablescans and index fast full scans in a system with a larger block size. (I mentioned in my book how moving an object to a tablespace with a different block size can cause a change in execution plan - this is another aspect of the same sort of thing).

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 6:08 AM in response to: Jonathan Lewis

Reply

I found sharing knowledge in this thread really great.

Thank you all !!!

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 5, 2008 6:58 AM in response to: Jonathan Lewis

Reply

> Charles,
>
> In response to your most recent posting I have gone
> back to review the extracts you picked out from the
> Oracle manuals - noting that they were all from the
> 11g manuals.
>
> It's an interesting collection that demonstrates two
> things:
> a) There is a need to consider
> db_file_multiblock_read_count in conjunction with
> db_block_size.
> b) The manuals start with some errors built in, and
> then get out of date
(snip)
> The opening statement is wrong - Oracle uses the
> value of the ***MBRC*** statistic when
> calculating the ***cost*** of performing the
> full tablescan (or index fast full scan). The whole
> thing is an example of writing that will not help the
> novice reader understand how things work - and I'm
> not sure that the note is correct in its description
> of how the optimizer responds to incomplete system
> stats.
>
> It's an interesting point that from 10.2 onwards the
> ***MBRC*** is supposed to default to 8 if you
> haven't set the
> ***db_file_mulitblock_read_count***.
> (Technically, it's the
> ***_db_file_optimizer_read_count*** that
> defaults to 8 and then the ***MBRC*** copies
> the parameter).
>
> You might wonder if this is a setting that is
> actually dependent on the ***block_size***.
> The last time I checked on a system with 16K blocks,
> though, it wasn't different - the value really does
> seem to be fixed at 8. This means that if you've
> allowed the
> ***db_file_mulitblock_read_count*** and
> system statistics to default, the optimizer will
> favour tablescans and index fast full scans in a
> system with a larger block size. (I mentioned in my
> book how moving an object to a tablespace with a
> different block size can cause a change in execution
> plan - this is another aspect of the same sort of
> thing).
>
> Regards
> Jonathan Lewis
> http://jonathanlewis.wordpress.com
> http://www.jlcomp.demon.co.uk

Very clearly explained, thank you.

This is one of the few times that I was more confused about Oracle's behavior *after* reading Oracle's documentation.

I will re-read the section of your book that you described. One of the interesting items found in Oracle 11g's 10046 trace file is that it now includes the calculated cost in the STAT lines (shown in the TKPROF output in the Row Source Operation lines). Greg Rahn's test case in this thread is a bit interesting, where the TKPROF output is showing the calculated cost for the different block sizes.

Size	Phy Rds	Cost	Phy Rds/Cost	Delta from Smaller Block Size
16	55822	21480	0.384794525	0.108514666
8	113157	31263	0.276279859	0.054253488
4	232603	51644	0.222026371	0.027129618
2	493333	96149	0.194896753	

The above shows that Oracle's calculated cost decreases with the larger block sizes in the test case, and there is a mathematical pattern to the cost shown in the last column of the above table.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Richard Foote

Re: Larger vs. Small data block
Posted: Jun 5, 2008 7:24 AM in response to: Greg Rahn

Reply

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Hi Greg

Excellent post, well done !!

It beautifully demonstrates how Oracle actually works, without the need for a complex, production environment to hide behind.

Let's just hope those that really really really need to read it actually do so, let's hope they can actually understand it and let's hope the penny finally, at long long last actually drops.

One lives in hope ...

Cheers

Richard Foote
http://richardfoote.wordpress.com/

---

oradba

Posts: 5,591
From: Germany
Registered: 9/15/00

Re: Larger vs. Small data block
Posted: Jun 5, 2008 7:25 AM in response to: Faust — Reply

... although it's more a discussion among philosophers ... ;-)

Werner

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 7:32 AM in response to: oradba — Reply

> ... although it's more a discussion among philosophers ... ;-)

Yeah, also because of that I like it so much...

It's so much to learn here in this thread, not only about Oracle technology -> also about people who working or teaching Oracle everyday :-)

Billy Verreynne

Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 5, 2008 8:14 AM in response to: Greg Rahn — Reply

Echoing Richard here... thanks Greg. Really an easy to read, consume and understand posting that illustrates the point very well.

Re: Larger vs. Small data block
Posted: Jun 5, 2008 8:33 AM in response to: Greg Rahn — Reply

---

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 5, 2008 9:37 AM in response to: — Reply

It's a shame when Greg previously demonstrated a similar example to you, you didn't get it:

http://forums.oracle.com/forums/thread.jspa?messageID=2176190&#2176190

If you've "observed increased contention with high DML on large blocksizes" why then do you still insist that the first thing an experience DBA should do is rebuild all indexes in the largest block size" ? Ummmm ...

How about the week after when you're not so swamped you produce a similar demo that shows and just as clearly explains why multi sized blocks are so beneficial and why indexes should be rebuilt in the largest block size ...

Now you know how easy it is to demonstrate a point without the need for a large production system :)

Cheers

Richard Foote
http://richardfoote.wordpress.com/

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 5, 2008 9:46 AM in response to: Greg Rahn — Reply

Thanks Greg. Your results look much like mine. I have never seen consistent, repeatable, differences except in highly contrived tests.

---

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 5, 2008 9:47 AM in response to: Greg Rahn — Reply

Greg,

Nice work. I would have expected some thing similar from the Lab Experts. Let me quote my words from my original posting

>>The example i give again is identical database in same server (created using same script).
>>All parameters same except for db_block_size. I did clean restart of both database and server
>>(no excuse for data cache or network traffic or bang on server) and executed the following sql
>>set in the server.

Now let me quote your words from the posting

>>To demonstrate my claim, I will create an experiment (test case). I am also going to add to
>>my claim that no matter what the blocksize, I can get the same read performance

```
>>create tablespace tpcds_8k  datafile '+GROUP1' size 1500m;
>>create tablespace tpcds_2k  datafile '+GROUP1' size 1500m blocksize 2k;
>>create tablespace tpcds_4k  datafile '+GROUP1' size 1500m blocksize 4k;
>>create tablespace tpcds_16k datafile '+GROUP1' size 1500m blocksize 16k;
```

In your case you have a single database (with 8k block size?) and you demonstrated the query
performance against 4 tablespace with different blocks.Do you actually think you have done correct test?.
I would like to remind the basic question, **Comparing query Performance in Two identical Database with different block size** . Not against a single database with multiple block size tablespaces

Since you have taken the effort to demonstrate a test case, i would like to encourage you to show
us execution result in separate database (Identical) with block size 8k and 16k.

Regards,
sp009

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 9:58 AM in response to: Richard Foote

Reply

>
> It's a shame when Greg previously demonstrated a
> similar example to you, you didn't get it:
>
> http://forums.oracle.com/forums/thread.jspa?messageID=
> 2176190?
>
> If you've "observed increased contention with high
> DML on large blocksizes" why then do you still
> insist that the first thing an experience DBA should
> do is rebuild all indexes in the largest block size"
> ? Ummmm ...
>
> How about the week after when you're not so swamped
> you produce a similar demo that shows and just as
> clearly explains why multi sized blocks are so
> beneficial and why indexes should be rebuilt in the
> largest block size ...
>
> Now you know how easy it is to demonstrate a point
> without the need for a large production system :)
>
> Cheers
>
> Richard Foote
> http://richardfoote.wordpress.com/

Hi Richard,

with this post you just "pour oil on fire"...

Nothing positive and constructive, I would say.

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 5, 2008 10:05 AM in response to: Faust

Reply

Hi Faust

On the contrary.

If can only **explain** why he still insists on a course of action that contradicts with his own observations and quotes from other sources and if he can actually **demonstrate** why such advice is valid and beneficial, then it would be a very positive and constructive outcome.

Cheers

Richard Foote
http://richardfoote.wordpress.com/

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 5, 2008 10:19 AM in response to: Greg Rahn

Reply

Greg,

Nice work. I would have expected some thing similar from the Lab Experts. Let me
quote my words from my original posting

>>The example i give again is identical database in same server (created using same script).
>>All parameters same except for db_block_size. I did clean restart of both database and server
>>(no excuse for data cache or network traffic or bang on server) and executed the following sql
>>set in the server.

Now let me quote your words from the posting

>>To demonstrate my claim, I will create an experiment (test case). I am also going to add to
>>my claim that no matter what the blocksize, I can get the same read performance

```
>>create tablespace tpcds_8k  datafile '+GROUP1' size 1500m;
>>create tablespace tpcds_2k  datafile '+GROUP1' size 1500m blocksize 2k;
>>create tablespace tpcds_4k  datafile '+GROUP1' size 1500m blocksize 4k;
>>create tablespace tpcds_16k datafile '+GROUP1' size 1500m blocksize 16k;
```

In your case you have a single database (with 8k block size?) and you demonstrated the query performance against 4 tablespace with different blocks.Do you actually think you have done correct test?.
I would like to remind the basic question, **Comparing query Performance in Two identical Database with different block size . Not against a single database with multiple block size tablespaces**

Since you have taken the effort to demonstrate a test case, i would like to encourage you to show us **execution result in separate database (Identical, Server too) with block size 8k and 16k.**
If you want, consider db_file_mutiblock_read_count too, so that db_block_size * db_file_mutiblock_read_count will be same on both database

Regards,
sp009

---

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 5, 2008 10:40 AM in response to: Richard Foote

Reply

Mr.Richard, Mr.Damorgan,

I have only one advice for you. Let me quote an analogy. that may be simple

You spend $5.98 for a meal and cashier says, don't have 2 cents change for $6. You will say "that's nothing for me, forget it".
Consider same routine for an year and see how much you neglected like "that's nothing for me, forget it"
Now consider 5000 people show the same attitude for a year and see how much accumulated like "that’s nothing for me forget it".

I think now you will get the point. It doesn't matter how much expert you are in a subject or not. You should have an open mind to
listen from all. That makes big difference

sp009

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 10:48 AM in response to: Richard Foote

Reply

> If can only **explain** why he ...
> ...
> can actually **demonstrate** why ...

And you believe that with words like:

"...you didn't get it..."
or
"... Now you know how easy it is to..."

you will push or anybody else to explain and/or demonstrate?

That's not good and positive pedagogy from my point of view.

BTW, also with children I will never go in that direction.

Cheers!

---

mpowel01

Posts: 2,840
Registered: 12/8/98

Re: Larger vs. Small data block
Posted: Jun 5, 2008 10:50 AM in response to: sp009

Reply

sp009, Greg test was designed to demonstrate the effect of changing the block size. Separate or same database really does not matter. When you use separate databases it is very difficult to verify, show, or prove that there are no differences in database parameter, hardware, disk, etc.... Greg's test was excellent for the intended prupose.

IMHO -- Mark D Powell --

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 10:55 AM in response to: mpowel01

Reply

> sp009, Greg test was designed to demonstrate the
> effect of changing the block size. Separate or same
> database really does not matter. When you use
> separate databases it is very difficult to verify,
> show, or prove that there are no differences in
> database parameter, hardware, disk, etc.... Greg's
> test was excellent for the intended prupose.
>
> IMHO -- Mark D Powell --

Right one for you sp009!

And please don't post your words as code - it can be misleading for the newbies...
;-)

Cheers!

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 5, 2008 11:00 AM in response to: Faust

Reply

Hi Faust

One lives in hope Faust, one lives in hope ...

Cheers

Richard Foote
http://richardfoote.wordpress.com/

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 5, 2008 11:01 AM in response to: mpowel01

Reply

> sp009, Greg test was designed to demonstrate the
> effect of changing the block size. Separate or same
> database really does not matter. When you use
> separate databases it is very difficult to verify,
> show, or prove that there are no differences in
> database parameter, hardware, disk, etc.... Greg's
> test was excellent for the intended prupose.
>
> IMHO -- Mark D Powell --

That makes a big difference, b'cos Server process the I/O request is
same in all 4 queries and the disk stripe depth/width is same, even though
the tablespace of different block size in 4 sql process.

Server process calculates the I/O request size based on db_block_size and
db_file_multiblock_read_count, not based on tablespace block size.

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 11:45 AM in response to: sp009

Reply

Could you elaborate on these points so I can further understand the details of your claims?

> That makes a big difference, b'cos Server process the I/O request is same in all 4 queries

How is the I/O request the same? In what cases would it be different? How would it be a "big difference"?

> and the disk stripe depth/width is same, even though the tablespace of different block size in 4 sql process.

In order to limit variables, I will use the same ASM disk group so it has the exact same characteristics for both databases.

> Server process calculates the I/O request size based on db_block_size and
> db_file_multiblock_read_count, not based on tablespace block size.

Exactly how do you believe the I/O request size is calculated?

If you believe that it is based on the database db_block_size and not the given tablespace db_block_size, I think I demonstrated this to **not** be the case with my first experiment:

```
2k  block * 512 MBRC = 1MB
4k  block * 256 MBRC = 1MB
8k  block * 128 MBRC = 1MB
16k block *  64 MBRC = 1MB
```

But I'll gladly run another experiment with a 8k and 16k *database* (not just tablespace) in my Oracle Laboratory and I'll post the results.

--
Regards,
Greg Rahn
http://structureddata.org

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 5, 2008 12:16 PM in response to: Greg Rahn

Reply

Greg,

> But I'll gladly run another experiment with a 8k and
> 16k *database* (not just tablespace) in my
> Oracle Laboratory and I'll post the results.

I would encourage you create two identical brand new database on same server or
identical server (same configuration) one with 8k*8k (standard) and other
with 16k*4k so that db_block_size * db_file_multiblock_read_count
will have same value in both database (as requested by Jonathan Louis)

Please test the sqls with couple of million rows, since few hundreds of rows will
not make any difference. Please remember, the debate here is performance difference
in low-concurrency DW database applications with high volume of I/O request

Also please trace the sqls to catch the wait events (again as requested by Jonathan Louis)

```
Alter Session Set Events '10046 trace name context forever, level 8';

Sqls............
Sqls............

Alter Session Set Events '10046 trace name context off';
```

I will do the same test very soon, whenever time permits

Thank you for taking the effort. After all i am not here to prove "I am Right" but to find the truth.

Regards,
sp009

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 4:38 PM in response to: sp009

Reply

I have built a db with a 16k block size and re-run the experiment.

```
SQL> show parameter db_block_size

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_block_size                        integer     16384

BEGIN_TIMESTAMP   QUERY_FILE                       ELAPSED_SECONDS ROW_COUNT
----------------- -------------------------------- --------------- ----------
20080605 11:32:32 q.sql                                 124.086276    7197670

********************************************************************************

/* q.sql */
select * from web_returns

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch    71978     23.85      24.84      55822     127217          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71980     23.85      24.84      55822     127217          0     7197670

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 28

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS (cr=127217 pr=55822 pw=55822 time=82535 us cost=21400 size=770150690 card=7197670)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71980        0.00          0.09
  SQL*Net message from client                 71980        0.00         93.20
  db file sequential read                         1        0.00          0.00
  direct path read                              981        0.01          0.22
  SQL*Net more data to client                 71976        0.00          1.79



********************************************************************************
WAIT #2: nam='direct path read' ela= 35 file number=4 first dba=53824 block cnt=64 obj#=11899 tim=1212690614763620
WAIT #2: nam='direct path read' ela= 27 file number=4 first dba=53888 block cnt=64 obj#=11899 tim=1212690614904103
WAIT #2: nam='direct path read' ela= 26 file number=4 first dba=53952 block cnt=64 obj#=11899 tim=1212690615043605
WAIT #2: nam='direct path read' ela= 38 file number=4 first dba=54016 block cnt=64 obj#=11899 tim=1212690615183407
WAIT #2: nam='direct path read' ela= 25 file number=4 first dba=54080 block cnt=64 obj#=11899 tim=1212690615324141
WAIT #2: nam='direct path read' ela= 32 file number=4 first dba=54144 block cnt=64 obj#=11899 tim=1212690615464674
WAIT #2: nam='direct path read' ela= 36 file number=4 first dba=54208 block cnt=64 obj#=11899 tim=1212690615605495
```

As you can see, the number of physical reads (55822) are exactly the same in a 16k tablespace whether the db_block_size is 8k or 16k. And again, the read I/O size is 1MB (16k block * 64 MBRC). The elapsed times are also close enough to be the same (125s vs. 124s)

Hopefully this demonstrates that either way, the results are the same.

I could care less about who is right and who is wrong. After all, when one is wrong and understands why, one learns something. This is what is important. I hope these experiments help you and others understand that is it the size of the I/O that matters, not the block size.

--
Regards,
Greg Rahn
http://structureddata.org

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 5, 2008 5:21 PM in response to:

Reply

>
> And let's not forgent the benefits of space. For
> random OLTP of 80 bytes rows (where the likelthood or
> re-using the data block is small), a 2k blocksize
> wastes less buffer cache. . . . .
>

That was one of the points made in the Metalink note about different block sizes that was clearly not thought through properly.

If the likelihood of reusing the data block is small then the number of disk I/Os made against that object will be the same whether the block is a 2K block or an 8K block. So changing the block size doesn't change the I/O load and response time, what you have to do is protect the main cache, which you can do by using the RECYCLE cache for the object.

You could argue that there is a time-saving in reading a 2K block instead of an 8K block - after all, it takes a smaller fraction of a rotation to collect 2K. However there are various mechanical reasons on modern hardware why ***small*** variations in read size are largely irrelevant - for example, I believe EMC's have a cache granularity of 32K, which means a read is not complete until 32K of data has been copied from the disk to the EMC cache.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 5, 2008 5:49 PM in response to: Jonathan Lewis

Reply

> sp009,
>
> Much better; however, given the interest in
> performance, it would have been helpful to run the
> trace at level 8 and including the wait summary so
> that we could see where the wait time went - the
> number, type, and average length of the waits could
> be very informative.
>
> If you feel like running the test again, please
> remember the significance of the
> db_file_multiblock_read_count.
>

Jonathan,

OK, i created 2 brand new identical database in same server with db_block_count
8k and 16k. All other parameters are same for both database. Let oracle decide the MBRC.
I have only 2 custom tables Employee and Department with 5m records. No index nothing.
Completed all Oracle recommended check list after creating the new databases.

```
SQL> connect / as sysdba
Connected.
SQL> Select *
  2    From v$version
  3  /

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Prod
PL/SQL Release 10.2.0.4.0 - Production
CORE         10.2.0.4.0   Production
TNS for 32-bit Windows: Version 10.2.0.4.0 - Production
NLSRTL Version 10.2.0.4.0 - Production

SQL> Select Name
  2    From v$database
  3  /

NAME
---------
DWDB

SQL> Select Name, Value
  2    From v$parameter
  3   Where Name = 'db_block_size'
  4  /

NAME
--------------------------------------------------------------------------------
VALUE
--------------------------------------------------------------------------------
db_block_size
16384


SQL> Select Count(1)
  2    From employee emp, department dept
  3   Where emp.dept_code = dept.dept_code
  4  /

  COUNT(1)
----------
   5000000

SQL> Alter Session Set Events '10046 trace name context forever, level 8'
  2  /

Session altered.

SQL> Alter Session Set Sql_trace=True
  2  /

Session altered.

SQL> Select Count(1)
  2    From employee emp, department dept
  3   Where emp.dept_code = dept.dept_code
  4  /

  COUNT(1)
----------
   5000000

SQL> Alter Session Set Sql_trace=False
  2  /

Session altered.

SQL> Alter Session Set Events '10046 trace name context off'
  2  /
```

```
Session altered.

SQL> spool off;

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SQL> connect / as sysdba
Connected.
SQL> Select *
  2    From v$version
  3  /

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Prod
PL/SQL Release 10.2.0.4.0 - Production
CORE         10.2.0.4.0   Production
TNS for 32-bit Windows: Version 10.2.0.4.0 - Production
NLSRTL Version 10.2.0.4.0 - Production

SQL> Select Name
  2    From v$database
  3  /

NAME
---------
TPDB

SQL> Select Name, Value
  2    From v$parameter
  3   Where Name = 'db_block_size'
  4  /

NAME
--------------------------------------------------------------------------------
VALUE
--------------------------------------------------------------------------------
db_block_size
8192


SQL> Select Count(1)
  2    From employee emp, department dept
  3   Where emp.dept_code = dept.dept_code
  4  /

  COUNT(1)
----------
   5000000

SQL> Alter Session Set Events '10046 trace name context forever, level 8'
  2  /

Session altered.

SQL> Alter Session Set Sql_trace=True
  2  /

Session altered.

SQL> Select Count(1)
  2    From employee emp, department dept
  3   Where emp.dept_code = dept.dept_code
  4  /

  COUNT(1)
----------
   5000000

SQL> Alter Session Set Sql_trace=False
  2  /

Session altered.

SQL> Alter Session Set Events '10046 trace name context off'
  2  /

Session altered.

SQL> spool off;

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

TKPROF RESULT


DATABASE: DWDB

TKPROF: Release 10.2.0.4.0 - Production on Thu Jun 5 16:30:22 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Trace file: dwdb_ora_2328.trc
Sort options: default

********************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
********************************************************************************

Alter Session Set Sql_trace=True


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
```

```
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0
Parsing user id: SYS

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************

Select Count(1)
  From employee emp, department dept
 Where emp.dept_code = dept.dept_code

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     12.82      24.39      18435      13900          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     12.82      24.39      18435      13900          0           1

Misses in library cache during parse: 0
Optimizer mode: FIRST_ROWS
Parsing user id: SYS

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=13900 pr=18435 pw=7560 time=24395502 us)
5000000   HASH JOIN  (cr=13900 pr=18435 pw=7560 time=21546079 us)
5000000    TABLE ACCESS FULL EMPLOYEE (cr=6095 pr=3133 pw=0 time=67 us)
5000000    TABLE ACCESS FULL DEPARTMENT (cr=7805 pr=7735 pw=0 time=243 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  direct path write temp                       1080        0.00          0.00
  db file scattered read                       1367        0.01          0.25
  direct path read temp                        1081        0.00          0.01
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************

Alter Session Set Sql_trace=False


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0
Parsing user id: SYS

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  SQL*Net message from client                     1        1.29          1.29
********************************************************************************

Alter Session Set Events '10046 trace name context off'


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0
Parsing user id: SYS



********************************************************************************

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        4      0.00       0.00          0          0          0           0
Execute      4      0.00       0.00          0          0          0           0
Fetch        2     12.82      24.39      18435      13900          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10     12.82      24.39      18435      13900          0           1

Misses in library cache during parse: 0

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       5        0.00          0.00
  SQL*Net message from client                     5        1.29          1.30
  direct path write temp                       1080        0.00          0.00
  db file scattered read                       1367        0.01          0.25
  direct path read temp                        1081        0.00          0.01
```

```
OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        0      0.00       0.00          0          0          0           0
Execute      0      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        0      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0

    4  user  SQL statements in session.
    0  internal SQL statements in session.
    4  SQL statements in session.
********************************************************************************
Trace file: dwdb_ora_2328.trc
Trace file compatibility: 10.01.00
Sort options: default

       1  session in tracefile.
       4  user  SQL statements in trace file.
       0  internal SQL statements in trace file.
       4  SQL statements in trace file.
       4  unique SQL statements in trace file.
    3596  lines in trace file.
      25  elapsed seconds in trace file.


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

DATABASE: TPDB

TKPROF: Release 10.2.0.4.0 - Production on Thu Jun 5 16:31:09 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Trace file: tpdb_ora_272.trc
Sort options: default

********************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
********************************************************************************

Alter Session Set Sql_trace=True


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0
Parsing user id: SYS

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************

Select Count(1)
  From employee emp, department dept
 Where emp.dept_code = dept.dept_code

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     13.20      27.61      34226      27954          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     13.20      27.61      34226      27954          0           1

Misses in library cache during parse: 0
Optimizer mode: FIRST_ROWS
Parsing user id: SYS

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=27954 pr=34226 pw=15285 time=27619188 us)
5000000   HASH JOIN  (cr=27954 pr=34226 pw=15285 time=34005775 us)
5000000    TABLE ACCESS FULL EMPLOYEE (cr=12254 pr=3327 pw=0 time=70 us)
5000000    TABLE ACCESS FULL DEPARTMENT (cr=15700 pr=15599 pw=0 time=260 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  direct path write temp                       1019        0.00          0.00
  db file scattered read                       2366        0.02          0.21
  db file sequential read                         1        0.00          0.00
  direct path read temp                        1020        0.00          0.01
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************

Alter Session Set Sql_trace=False
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0
Parsing user id: SYS

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  SQL*Net message from client                     1        1.35          1.35
********************************************************************************

Alter Session Set Events '10046 trace name context off'


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0
Parsing user id: SYS



********************************************************************************

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        4      0.00       0.00          0          0          0           0
Execute      4      0.00       0.00          0          0          0           0
Fetch        2     13.20      27.61      34226      27954          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10     13.20      27.61      34226      27954          0           1

Misses in library cache during parse: 0

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       5        0.00          0.00
  SQL*Net message from client                     5        1.35          1.36
  direct path write temp                       1019        0.00          0.00
  db file scattered read                       2366        0.02          0.21
  db file sequential read                         1        0.00          0.00
  direct path read temp                        1020        0.00          0.01


OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        0      0.00       0.00          0          0          0           0
Execute      0      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        0      0.00       0.00          0          0          0           0

Misses in library cache during parse: 0

    4  user  SQL statements in session.
    0  internal SQL statements in session.
    4  SQL statements in session.
********************************************************************************
Trace file: tpdb_ora_272.trc
Trace file compatibility: 10.01.00
Sort options: default

       1  session in tracefile.
       4  user  SQL statements in trace file.
       0  internal SQL statements in trace file.
       4  SQL statements in trace file.
       4  unique SQL statements in trace file.
    4474  lines in trace file.
      29  elapsed seconds in trace file.
```

I would love to see an expert explanation from you for the above cpu/cost difference in both database.

Regards,
sp009

> Regards
> Jonathan Lewis
> http://jonathanlewis.wordpress.com
> http://www.jlcomp.demon.co.uk

jgarry

Posts: 128

Re: Larger vs. Small data block
Posted: Jun 5, 2008 6:29 PM in response to: sp009

Reply

From: Just outside of beautiful Vista, California
Registered: 7/20/98

What is your db_cache_size?
Could you display the contents of v$bh after each test?

```
>call     count       cpu    elapsed       disk      query    current        rows
>------- ------  -------- ---------- ---------- ---------- ----------  ----------

>Fetch        2     12.82      24.39      18435      13900          0           1

>Fetch        2     13.20      27.61      34226      27954          0           1
```

I'm wondering if what we are seeing is the result of extra cpu overhead of Oracle having to manage twice as many blocks for the same data in the 16K v. 8K tests. Could the disk requests be misleading because some are satisfied without real disk reads? Isn't the query reasonable to be double because it has to look at twice as many blocks?

```
>Event waited on                            Times   Max. Wait  Total Waited
>----------------------------------------   Waited  ----------  ------------
>db file scattered read                       1367        0.01          0.25

>db file scattered read                       2366        0.02          0.21
```

So we wait twice as many times, but the total is less...
on a busy system, the more often you wait, the more cpu you use in the trade-off, the more likely everyone else is going to make you wait even more. Right?

It depends...

Hemant K Chitale

Posts: 1,259
Registered: 11/6/98

Re: Larger vs. Small data block
Posted: Jun 5, 2008 9:12 PM in response to: mpowel01

Reply

>> When you use separate databases it is very difficult to verify, show, or prove that there are no differences in database parameter, hardware, disk, etc....

If both the databases are on the same server (and only one database is up
at any time) what would be the difference between these tests and seperate
tablespaces with different block sizes ?

A single database with multiple block size tablespaces does have seperate
datafiles (therefore on seperate locations "on disk") and seperate caches.
BUT they share the same system and undo tablespace (both of which
have only one, default, block size). They also share the same TEMPORARY
tablespace, when running in the same schema (hmm... I wonder if anyone
tries changing the block size for the TEMPORARY Tablespace).

On the other hand, with two databases with *different* db_block_sizes,
even system, undo and temp have different block sizes. In my opinion,
sp009 is conducting a valid "TEST FOR BLOCK SIZE".

The two tests (multiple block sizes in one tablespace V different
default db_block_size) ARE Different. But the second Test is a valid test
for the hypothesis that changing the Block Size might/can make a difference.

Hemant K Chitale

Posts: 1,259
Registered: 11/6/98

Re: Larger vs. Small data block
Posted: Jun 5, 2008 9:31 PM in response to: sp009

Reply

The EMPLOYEE Table seems to be smaller than the DEPARTMENT table ??
What are their sizes ? Is it because DEPARTMENT has a much larger
AVG_ROW_LENGTH ?

In the 16K database DWDB :

```
5000000    TABLE ACCESS FULL EMPLOYEE (cr=6095 pr=3133 pw=0 time=67 us)5000000    TABLE ACCESS FULL DEPARTMENT (cr=7805
pr=7735 pw=0 time=243 us)
```

In the 8K database TPDB :

```
5000000    TABLE ACCESS FULL EMPLOYEE (cr=12254 pr=3327 pw=0 time=70 us)5000000    TABLE ACCESS FULL DEPARTMENT (cr=15700
pr=15599 pw=0 time=260 us)
```

As expected, the 16K blocks are half as many as the 8K blocks. But the
number of EMPLOYEE blocks not in the db_cache are much lower in TPDB
(8KB). It would seem that most of the EMPLOYEE blocks were present in
the db_cache in TPDB but not as many (proportionally !) in DWDB.

(Also, it seems as if DEPARTMENT is larger than EMPLOYEE -- possibly
larger AVG_ROW_LENGTH).

HOWEVER, the Gain seems to be in the HASH JOIN :

In DWDB :

```
5000000   HASH JOIN  (cr=13900 pr=18435 pw=7560 time=21546079 us
```

In TPDB :

```
5000000   HASH JOIN  (cr=27954 pr=34226 pw=15285 time=34005775 us)
```

Even if we account for the possibility that the Timings for the FullTableScans
might be part of the total time for the Hash Join and therefore deduct them,
the Hash Join was much faster in DWDB.
Either the memory allocated for the Hash Join in DWDB was larger
(depending on PGA_AGGREGATE_TARGET / WORKAREA_SIZE_POLICY ,
SORT_AREA_SIZE, HASH_AREA_SIZE) and/or the Hash Join overflows
to/from disk performed better in DWDB.

Presumably the TEMPORARY Tablespace also had the same db_block_size
(I haven't heard of anyone changing the TEMPORARY Tablespace block size
or know if it is possible).

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 6, 2008 4:08 AM in response to: sp009

Reply

I've cut and pasted the central parts of your trace files:

16K Block size:
```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     12.82      24.39      18435      13900          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     12.82      24.39      18435      13900          0           1

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=13900 pr=18435 pw=7560 time=24395502 us)
5000000   HASH JOIN  (cr=13900 pr=18435 pw=7560 time=21546079 us)
5000000    TABLE ACCESS FULL EMPLOYEE (cr=6095 pr=3133 pw=0 time=67 us)
5000000    TABLE ACCESS FULL DEPARTMENT (cr=7805 pr=7735 pw=0 time=243 us)

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  direct path write temp                       1080        0.00          0.00
  db file scattered read                       1367        0.01          0.25
  direct path read temp                        1081        0.00          0.01
  SQL*Net message from client                     2        0.00          0.00
```

8K Block size
```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     13.20      27.61      34226      27954          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     13.20      27.61      34226      27954          0           1

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=27954 pr=34226 pw=15285 time=27619188 us)
5000000   HASH JOIN  (cr=27954 pr=34226 pw=15285 time=34005775 us)
5000000    TABLE ACCESS FULL EMPLOYEE (cr=12254 pr=3327 pw=0 time=70 us)
5000000    TABLE ACCESS FULL DEPARTMENT (cr=15700 pr=15599 pw=0 time=260 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  direct path write temp                       1019        0.00          0.00
  db file scattered read                       2366        0.02          0.21
  db file sequential read                         1        0.00          0.00
  direct path read temp                        1020        0.00          0.01
  SQL*Net message from client                     2        0.00          0.00
```

Points to notice:
The "lost time" in these reports far outweighs the differences that you are worried about. The 8K block results show a 14 second difference between CPU time and elapsed time, but the wait summary accounts for less than one second of that time. It's not sensible to worry about 0.38 CPU seconds difference, and 3.22 seconds elapsed time difference when the measurement error is 14 seconds, as the error may be hiding the fact that Oracle was not actually doing the same thing in both cases.

The employee table is the same size in both tests - but the number of blocks read to scan the table is also (nearly) the same in both cases: so you must have had roughly 50% of the blocks in cache before starting the 8K test - you have to ask yourself how this could have affected results. You might want to try issuing "alter system flush buffer_cache" before each test run.

The pre-caching of the employee table may have limited the size and affected the relative efficiency of the reads in the case of the 8K block rest (hence the increased of the number of read requests) - on the other hand if you sum the *"pr="* figures for the tablescans and divide by the value of "db file scattered read" the answer comes very close to 8 blocks per read in both cases. So I think it's more likely that you've left a hard coded limit on the db_file_multiblock_read_count. In an earlier post you said something about wanting to test the effect of changing the block size in a data warehouse that was running with low concurrency and large queries - you haven't configured this database in an appropriate fashion for such a data warehouse if you've set the db_file_multiblock_read_count to 8.

The **number** of direct path writes is nearly the same on the two tests - although the number of blocks written is roughly doubled. This shows that the mechanics of the hash join behaved in very similar ways on both systems. We can infer that the number of partitions used for the hash table, and the chunk (slot) size were the same in both cases. (Note, by the way, that this means the unit I/O for the direct path write was kept constant - doubling the block count as the block size halved).

A thought about the lost time - it looks as if your tables may have been cached in a local file-system cache (unless there's a big problem with timing on your platform with 10.2.0.4). It's possible that the lost time is spent somehow at the operating system level due to odd effects of Oracle prefetching and pseudo-asynchronous I/O. (This isn't a single CPU / single core machine, is it ?)

A final point to consider - your employee rows have an average row length of about 10 bytes, and your department rows have an average row length of about 25. This isn't particularly representative of a data warehouse - so you have to ask yourself if the test will exaggerate the difference in performance that would normally appear, or would it tend to hide the difference ? By comparison, Greg Rahn's example used 7M rows of about 140 bytes totalling something 1GB of data, rather than 5M rows in 50M. Whatever results you finally get with this data set, they may simply represent an extreme special case.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 6, 2008 9:26 AM in response to: Jonathan Lewis

Reply

Jonathan,

Thanks for having a look in to that. I didn't convince my self with your answer. Never mind.

>>Whatever results you finally get with this data set, they may simply represent an extreme special case.

I have done this test with two identical database in same server with different block size.
I wish i can show the tkprof of some of the long run queries in my production and test database
(identical server, windows 2003/64 with 16k and 8k block size and data nearly same).
But the policy doesn't allow me to do that.

I would encourage every one to test the case your self and see the result. Here i am
talking about only DW applications with large volume of I/O requests. Thanks to every
one for their contributions

Regards,
sp009

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 6, 2008 11:42 AM in response to: sp009 Reply

> Thanks for having a look in to that. I didn't
> convince my self with your answer. Never mind.

Fair enough - but at least we've had a discussion which has highliighted the importance of constructuing experiments to test a hypothesis, and given other people the chance to see how careful you have to be to design the test properly/

> I wish i can show the tkprof of some of the long run
> queries in my production and test database
> (identical server, windows 2003/64 with 16k and 8k
> block size and data nearly same).
> But the policy doesn't allow me to do that.

I've never been convinced that this makes it impossible to share performance data without compromising business intelligence.
After all, if you want to examine the I/O pattern for a query you can cut one statement out of a tkprof file, delete the SQL, and change the names of the tables and indexes in the rowsource output in a consistent fashion.

You might be able so show an example of that sort of thing to your governance officer and get clearance to show it on the forum.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 6, 2008 2:36 PM in response to: Jonathan Lewis Reply

Jonathan,

I will definitely try to get tkprof of one my long running query in DW application (process 30m rows) and the same in the test server.

Thanks,
sp009

Boochi

Posts: 87
From: USA
Registered: 11/23/07

Re: Larger vs. Small data block
Posted: Jun 6, 2008 2:52 PM in response to: user619401 Reply

Hi OP,

What did you understand from these many replies?.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 6, 2008 4:29 PM in response to: Boochi Reply

I would say, for db_block_size "It Depends". Like may other intelligent software, in many areas, Oracle too doesn't play by rule. That's my experience and understanding. I am sure like me, there will be many customers thinking the same. Debates on these areas may go till 20g and beyond and can never stop.

sp009

Re: Larger vs. Small data block
Posted: Jun 6, 2008 5:09 PM in response to: Jonathan Lewis Reply

Re: Larger vs. Small data block
Posted: Jun 6, 2008 5:12 PM in response to: sp009 Reply

Madrid

Posts: 7,145
From: Mexico City
Registered: 3/8/99

Re: Larger vs. Small data block
Posted: Jun 6, 2008 6:09 PM in response to: Madrid Reply

> Daniel,
>
> I don't agree nor disagree, I am just looking for the
> truth. You said you have some lab tests, If you
> don't mind I would like to take a look at your
> research results. Have you published them in
> internet? Are they available?
>

> Regards.

Daniel Morgan,

Let me insist on this point, if you have your research results, do you mind publishing them?

Regards

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 6, 2008 9:56 PM in response to:

Reply

Same old examples as have already been addressed many times, such as here:

http://richardfoote.wordpress.com/2008/03/20/store-indexes-in-a-larger-block-tablespace-the-multiblock-read-myth-part-ii-the-fly/#comment-605

But just to hightlight a couple of them again:

The OTN link you mentioned, is that Is that the same Santosh Kumar thread where he only asks the question “Is it true” based a question on a AskTom thread where an anonymous Russian makes unsubstantiated claims on the benefits of bigger index block sizes (dismissed by Tom), where you, **yourself** admit “**Yeah, I redacted that one**” when I highlighted to you Santosh himself never actually made the claim himself !! :

Is that the same M.J Schwenger who in the same forum thread you got his quote from asks whether or not using multiple blocksizes is actually a good idea or not !!

Is that the same Balkrishan Mittal who in the very same forum discussion as M. J. Schwenger warns him not to use a larger block size as it caused him **negative results** with 100% CPU consumption and was forced within days to put the indexes back in a smaller block size !!

Is that the very same David Aldridge you banned from your forum because he actually disagreed with you that the 6% improvement had anything to do with different block sizes: http://oraclesponge.blogspot.com/2005_04_01_archive.html

Finally, why are anonymous Russians, a simple demo such as David's (who disagrees with you), etc. "credible Oracle shop" but not other demos which disagree with your conclusions ?

http://richardfoote.wordpress.com/2008/03/31/larger-block-index-tablespace-and-small-index-scans-performance-improvement-let-down/

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 6, 2008 10:24 PM in response to:

Reply

(Snip)
> You keep dissing the tests here, but I don't see you
> enlightening us with a valid test.
>
> Why is that?
>
> As my personal debunker, I expect a test showing the
> performance differences with different blocksizes
> (measuring throughput and response time).

I just completed a test case here with 2 identical new database instances, one with a default block size of 8KB, and the second with a default block size of 16KB. The test case, including all 50,000,000+ rows of data in the test tables, should allow the results to be very easily reproduced on any Oracle platform without risk of exposing company secrets. The results were surprising, at least not exactly what I was expecting. I did not test just a simple 2 table join.

First, the database initialization parameters used to create each database instance:

```
#INITIALIZATION PARAMETERS 8KB BLOCK
background_core_dump=partial
cluster_database=FALSE
compatible=10.2.0.2.0
control_files=("C:\oracle\OraData\TEST8\ctlTEST801.ctl", "C:\oracle\flash_recovery_area\TEST8\ctlTEST802.ctl")
control_file_record_keep_time=7
cursor_sharing=EXACT
cursor_space_for_time=true
db_block_size=8192
db_cache_advice=on
db_block_checking=false
db_block_checksum=typical
db_domain=world
db_files=200
db_flashback_retention_target=1440
db_name=TEST8
db_recovery_file_dest_size=14000M
db_recovery_file_dest=C:\oracle\flash_recovery_area
db_unique_name=TEST8
db_writer_processes=1
global_names=false
instance_name=TEST8
java_pool_size=1M
job_queue_processes=10
log_archive_format=arc%s_%r.%t
log_buffer=1048576
log_checkpoint_interval=65536
log_checkpoint_timeout=3600
log_checkpoints_to_alert=false
max_dump_file_size=202400
nls_language=american
nls_territory=america
O7_DICTIONARY_ACCESSIBILITY=TRUE
open_cursors=1000
open_links=4
optimizer_dynamic_sampling=2
optimizer_features_enable=10.2.0.2
optimizer_index_caching=0
```

```
optimizer_index_cost_adj=100
optimizer_mode=ALL_ROWS
pga_aggregate_target=300M
plsql_code_type=INTERPRETED
processes=210
query_rewrite_enabled=FALSE
query_rewrite_integrity=TRUSTED
recyclebin=ON
remote_login_passwordfile=EXCLUSIVE
service_names=TEST8
sessions=236
session_cached_cursors=200
sga_max_size=1100M
sga_target=900M
star_transformation_enabled=FALSE
statistics_level=typical
timed_statistics=true
transactions=259
transactions_per_rollback_segment=5
undo_management=AUTO
undo_retention=1800
undo_tablespace=ROLLBACK_DATA
workarea_size_policy=auto
background_dump_dest=C:\oracle\product\10.2.0\admin\TEST8\bdump
core_dump_dest=C:\oracle\product\10.2.0\admin\TEST8\cdump
user_dump_dest=C:\oracle\product\10.2.0\admin\TEST8\udump
utl_file_dir=C:\oracle\product\10.2.0\admin\TEST8\udump


#INITIALIZATION PARAMETER MODIFICATIONS FOR 16KB BLOCK
control_files=("C:\oracle\OraData\test16\ctltest1601.ctl", "C:\oracle\flash_recovery_area\test16\ctltest1602.ctl")
db_block_size=16384
background_dump_dest=C:\oracle\product\10.2.0\admin\test16\bdump
core_dump_dest=C:\oracle\product\10.2.0\admin\test16\cdump
user_dump_dest=C:\oracle\product\10.2.0\admin\test16\udump
utl_file_dir=C:\oracle\product\10.2.0\admin\test16\udump


#CREATE DATABASE COMMAND FOR 8KB BLOCK SIZE:
CREATE DATABASE "TEST8"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 20
MAXLOGMEMBERS 3
MAXDATAFILES 100
DATAFILE 'c:\oracle\oradata\TEST8\SystemTEST801.dbf' SIZE 700M AUTOEXTEND ON NEXT 20M MAXSIZE UNLIMITED EXTENT MANAGEMENT
LOCAL
SYSAUX DATAFILE 'c:\oracle\oradata\TEST8\SysauxTEST801.dbf' SIZE 300M AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
SMALLFILE DEFAULT TEMPORARY TABLESPACE TEMPORARY_DATA1 TEMPFILE 'c:\oracle\oradata\TEST8\TmpTEST801.dbf' SIZE 1024M AUTOEXTEND
ON NEXT 40M MAXSIZE 5000M  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
SMALLFILE UNDO TABLESPACE "ROLLBACK_DATA" DATAFILE 'c:\oracle\oradata\TEST8\undotbsTEST801.dbf' SIZE 800M AUTOEXTEND ON NEXT
20M MAXSIZE UNLIMITED
CHARACTER SET WE8MSWIN1252
NATIONAL CHARACTER SET AL16UTF16
LOGFILE GROUP 1 ('c:\oracle\oradata\TEST8\RedoTEST801.log') SIZE 512M,
GROUP 2 ('c:\oracle\oradata\TEST8\RedoTEST802.log') SIZE 512M,
GROUP 3 ('c:\oracle\oradata\TEST8\RedoTEST803.log') SIZE 512M,
GROUP 4 ('c:\oracle\oradata\TEST8\RedoTEST804.log') SIZE 512M,
GROUP 5 ('c:\oracle\oradata\TEST8\RedoTEST805.log') SIZE 512M,
GROUP 6 ('c:\oracle\oradata\TEST8\RedoTEST806.log') SIZE 512M
USER SYS IDENTIFIED BY "&&sysPassword" USER SYSTEM IDENTIFIED BY "&&systemPassword";
CREATE SMALLFILE TABLESPACE "USER_DATA" LOGGING DATAFILE 'C:\oracle\oradata\TEST8\usrTEST801.dbf' SIZE 2000M AUTOEXTEND ON
NEXT 100M MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;


#CREATE DATABASE COMMAND FOR 16KB BLOCK SIZE:
CREATE DATABASE "test16"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 20
MAXLOGMEMBERS 3
MAXDATAFILES 100
DATAFILE 'c:\oracle\oradata\test16\Systemtest1601.dbf' SIZE 700M AUTOEXTEND ON NEXT 20M MAXSIZE UNLIMITED EXTENT MANAGEMENT
LOCAL
SYSAUX DATAFILE 'c:\oracle\oradata\test16\Sysauxtest1601.dbf' SIZE 300M AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
SMALLFILE DEFAULT TEMPORARY TABLESPACE TEMPORARY_DATA1 TEMPFILE 'c:\oracle\oradata\test16\Tmptest1601.dbf' SIZE 1024M
AUTOEXTEND ON NEXT 40M MAXSIZE 5000M EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
SMALLFILE UNDO TABLESPACE "ROLLBACK_DATA" DATAFILE 'c:\oracle\oradata\test16\undotbstest1601.dbf' SIZE 800M AUTOEXTEND ON NEXT
20M MAXSIZE UNLIMITED
CHARACTER SET WE8MSWIN1252
NATIONAL CHARACTER SET AL16UTF16
LOGFILE GROUP 1 ('c:\oracle\oradata\test16\Redotest1601.log') SIZE 512M,
GROUP 2 ('c:\oracle\oradata\test16\Redotest1602.log') SIZE 512M,
GROUP 3 ('c:\oracle\oradata\test16\Redotest1603.log') SIZE 512M,
GROUP 4 ('c:\oracle\oradata\test16\Redotest1604.log') SIZE 512M,
GROUP 5 ('c:\oracle\oradata\test16\Redotest1605.log') SIZE 512M,
GROUP 6 ('c:\oracle\oradata\test16\Redotest1606.log') SIZE 512M
USER SYS IDENTIFIED BY "&&sysPassword" USER SYSTEM IDENTIFIED BY "&&systemPassword";
CREATE SMALLFILE TABLESPACE "USER_DATA" LOGGING DATAFILE 'C:\oracle\oradata\test16\usrtest1601.dbf' SIZE 2000M  AUTOEXTEND ON
NEXT 100M MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;


The tests:
#######################
#TEST RUN 1 AFTER A RESTART, ONLY 16KB DTAABASE INSTANCE STARTED
ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;

spool c:\test16.txt
set autotrace on
set timing on

SELECT
  COUNT(*)
FROM
  ALL_OBJECTS;

CREATE TABLE T1 AS
SELECT
 A.*,
 RN
FROM
  (SELECT
    *
  FROM
    ALL_OBJECTS A
  WHERE
    ROWNUM<=10000) A,
  (SELECT
    ROWNUM RN
  FROM
    DUAL
  CONNECT BY
    LEVEL<=5000);

COMMIT;

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;

CREATE INDEX T1_IND1 ON T1(OWNER,OBJECT_NAME,SUBOBJECT_NAME,RN);

CREATE TABLE T2 AS
SELECT
  *
FROM
  T1
WHERE
  1=2;

CREATE INDEX T2_IND1 ON T2(OWNER,OBJECT_NAME,SUBOBJECT_NAME,RN);

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;

INSERT INTO T2
SELECT
  *
FROM
  T1
WHERE
  RN<=100;

COMMIT;

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;

ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 8';

SELECT
  *
FROM
  T1
WHERE
  STATUS='NONE';

SELECT
  COUNT(*)
FROM
  T2;

SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1;

ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF';

SPOOL OFF

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;
######################

#TEST RUN 2 IN SECOND SESSION WITH 10046 TRACE LEVEL 8, 10053 TRACE LEVEL 1, SESSION LEVEL STATISTICS_LEVEL=ALL, DBMS_XPLAN
ALL STATS LAST, ONLY 16KB DTAABASE INSTANCE STARTED
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1;
######################

#TEST RUN 3 AFTER A RESTART, ONLY 16KB DTAABASE INSTANCE STARTED
spool c:\test16-2.txt
set autotrace on
set timing on

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T1',CASCADE=>TRUE);

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T1',CASCADE=>TRUE);

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;

ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 8';

SELECT DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
WHERE
  STATUS='NONE';
```

```
ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF';

SELECT
  TABLE_NAME,
  NUM_ROWS,
  BLOCKS,
  AVG_ROW_LEN
FROM
  USER_TABLES
WHERE
  TABLE_NAME IN ('T1','T2');

SELECT
  INDEX_NAME,
  BLEVEL,
  LEAF_BLOCKS,
  DISTINCT_KEYS,
  AVG_LEAF_BLOCKS_PER_KEY,
  AVG_DATA_BLOCKS_PER_KEY,
  CLUSTERING_FACTOR
FROM
  USER_INDEXES
WHERE
  TABLE_NAME IN ('T1','T2');

SPOOL OFF
#######################


#######################
#TEST RUN 4 AFTER A RESTART, ONLY 8KB DTAABASE INSTANCE STARTED
#SAME AS TEST RUN 1, EXCEPT SPOOL TO c:\test8.txt
#######################

#TEST RUN 5 IN SECOND SESSION WITH 10046 TRACE LEVEL 8, 10053 TRACE LEVEL 1, SESSION LEVEL STATISTICS_LEVEL=ALL, DBMS_XPLAN
ALL STATS LAST, ONLY 8KB DTAABASE INSTANCE STARTED
#SAME AS TEST RUN 2
#######################

#TEST RUN 6 AFTER A RESTART, ONLY 8KB DTAABASE INSTANCE STARTED
#SAME AS TEST RUN 3
#######################
```

The initial results will be posted next, and analysis of the 10046 trace files will follow later.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 6, 2008 10:44 PM in response to: Charles Hooper

Reply

During this test, system statistics were not collected, and the database instances were not archiving redo logs. Tested on Oracle 10.2.0.2 on a low end 32 bit Windows box with 3.8GHz P4, 2GB of RAM, and 2 hard drives in RAID 0.

On the 16KB block size database, Oracle automatically set the DB_FILE_MULTIBLOCK_READ_COUNT=64

On the 8KB block size database, Oracle automatically set the DB_FILE_MULTIBLOCK_READ_COUNT=128

The above surprised me a bit.

```
################ RESULTS ################
#TEST RUN 1 16KB
  COUNT(*)
----------
     11073

Elapsed: 00:00:00.68

Execution Plan...

Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
      19328  consistent gets
        190  physical reads
          0  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


Table created.

Elapsed: 00:01:48.15

Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:04.50

System altered.

Elapsed: 00:00:00.03

Index created.
```

```
Elapsed: 00:10:30.96

Table created.

Elapsed: 00:00:01.50

Index created.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:01.62

System altered.

Elapsed: 00:00:00.03

1000000 rows created.

Elapsed: 00:02:08.28

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |      |   751K|   101M|   122K  (2)| 00:28:37 |
|*  1 |  TABLE ACCESS FULL| T1   |   751K|   101M|   122K  (2)| 00:28:37 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RN"<=100)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       6531  recursive calls
    2490348  db block gets
     352150  consistent gets
     321601  physical reads
  444972176  redo size
        681  bytes sent via SQL*Net to client
        583  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
    1000000  rows processed


Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:10.60

System altered.

Elapsed: 00:00:00.00

Session altered.

Elapsed: 00:00:00.06

no rows selected

Elapsed: 00:01:12.87

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |  3544 |   487K|   122K  (2)| 00:28:33 |
|*  1 |  TABLE ACCESS FULL| T1   |  3544 |   487K|   122K  (2)| 00:28:33 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("STATUS"='NONE')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     321695  consistent gets
     321569  physical reads
          0  redo size
       1047  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


  COUNT(*)
----------
   1000000

Elapsed: 00:00:02.37

Execution Plan
----------------------------------------------------------
Plan hash value: 1385691034

-------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |     1 |  1230   (1)| 00:00:18 |
|   1 |  SORT AGGREGATE       |         |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| T2_IND1 |   968K|  1230   (1)| 00:00:18 |
-------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         32  recursive calls
          3  db block gets
       6812  consistent gets
       4294  physical reads
     242044  redo size
        411  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


OWNER                          OBJECT_NAME
------------------------------ ------------------------------
SUBOBJECT_NAME
------------------------------                                                    ...

9454 rows selected.

Elapsed: 00:01:28.62

Execution Plan
----------------------------------------------------------
Plan hash value: 1118578911

------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |    54M|  2666M|   574K  (1)| 02:14:00 |
|   1 |  SORT UNIQUE NOSORT|         |    54M|  2666M|   574K  (1)| 02:14:00 |
|   2 |   INDEX FULL SCAN  | T1_IND1 |    54M|  2666M|   136K  (1)| 00:31:51 |
------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     135802  consistent gets
     135073  physical reads
          0  redo size
     299135  bytes sent via SQL*Net to client
       7311  bytes received via SQL*Net from client
        632  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9454  rows processed


Session altered.

Elapsed: 00:00:00.00


#TEST RUN 2 16KB
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1;

-------------------------------------------------------------------------------------------------
| Id  | Operation          | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------
|   1 |  SORT UNIQUE NOSORT|         |      1 |     54M|   9454 |00:02:19.11 |     135K|    135K|
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     54M|     50M|00:01:40.05 |     135K|    135K|
-------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


#TEST RUN 3 16KB
```

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:30.67

PL/SQL procedure successfully completed.

Elapsed: 00:02:30.07

System altered.

Elapsed: 00:00:00.04

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.01

no rows selected

Elapsed: 00:01:15.48

Execution Plan
----------------------------------------------------------
Plan hash value: 2134347679

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    32 |   122K  (2)| 00:28:32 |
|   1 |  HASH UNIQUE       |      |     1 |    32 |   122K  (2)| 00:28:32 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |    32 |   122K  (2)| 00:28:32 |
---------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("STATUS"='NONE')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
     321597  consistent gets
     321569  physical reads
          0  redo size
        399  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


Session altered.

Elapsed: 00:00:00.00

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
T1                               50050157     322128          88
T2


INDEX_NAME                         BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------------------------ ---------- ----------- ------------- ----------------------- ----------------------- ----------
-------
T1_IND1                                 2      138623      48307975                       1                       1
49273616
T2_IND1



#TEST RUN 4 8KB
  COUNT(*)
----------
     11073

Elapsed: 00:00:00.62

Execution Plan...

Statistics
----------------------------------------------------------
        641  recursive calls
          0  db block gets
      19570  consistent gets
        380  physical reads
        116  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         25  sorts (memory)
          0  sorts (disk)
          1  rows processed


Table created.

Elapsed: 00:01:41.48

Commit complete.

Elapsed: 00:00:00.00

System altered.
```

```
Elapsed: 00:00:02.31

System altered.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:08:28.31

Table created.

Elapsed: 00:00:01.01

Index created.

Elapsed: 00:00:00.01

System altered.

Elapsed: 00:00:00.81

System altered.

Elapsed: 00:00:00.01

1000000 rows created.

Elapsed: 00:01:53.59

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |      |   776K|   104M|   178K  (2)| 00:35:46 |
|*  1 |  TABLE ACCESS FULL| T1   |   776K|   104M|   178K  (2)| 00:35:46 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RN"<=100)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       7290  recursive calls
    2854734  db block gets
     712468  consistent gets
     651602  physical reads
  469393664  redo size
        681  bytes sent via SQL*Net to client
        583  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
    1000000  rows processed


Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:17.45

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.03

no rows selected

Elapsed: 00:01:01.21

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |  7180 |   988K|   178K  (1)| 00:35:43 |
|*  1 |  TABLE ACCESS FULL| T1   |  7180 |   988K|   178K  (1)| 00:35:43 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("STATUS"='NONE')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
```

```
          5  recursive calls
          0  db block gets
     651592  consistent gets
     651470  physical reads
          0  redo size
       1047  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


  COUNT(*)
----------
   1000000

Elapsed: 00:00:02.57

Execution Plan
----------------------------------------------------------
Plan hash value: 1385691034

-------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |     1 |  1863   (1)| 00:00:23 |
|   1 |  SORT AGGREGATE       |         |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| T2_IND1 |   796K|  1863   (1)| 00:00:23 |
-------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         32  recursive calls
          3  db block gets
      14148  consistent gets
       7745  physical reads
     505960  redo size
        411  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


OWNER                          OBJECT_NAME
------------------------------ ------------------------------
SUBOBJECT_NAME
------------------------------ ...

9454 rows selected.

Elapsed: 00:01:43.59

Execution Plan
----------------------------------------------------------
Plan hash value: 1118578911

------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |    50M|  2459M|   921K  (1)| 03:04:19 |
|   1 |  SORT UNIQUE NOSORT|         |    50M|  2459M|   921K  (1)| 03:04:19 |
|   2 |   INDEX FULL SCAN  | T1_IND1 |    50M|  2459M|   276K  (1)| 00:55:24 |
------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     274741  consistent gets
     274363  physical reads
          0  redo size
     299090  bytes sent via SQL*Net to client
       7311  bytes received via SQL*Net from client
        632  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9454  rows processed


Session altered.

Elapsed: 00:00:00.00


#TEST RUN 5 8KB
-------------------------------------------------------------------------------------------------
| Id  | Operation          | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------
|   1 |  SORT UNIQUE NOSORT|         |      1 |     50M|   9454 |00:02:38.02 |     274K|    274K|
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     50M|     50M|00:01:40.08 |     274K|    274K|
-------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement

#TEST RUN 6 8KB
```

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:12.53

PL/SQL procedure successfully completed.

Elapsed: 00:02:01.07

System altered.

Elapsed: 00:00:00.06

System altered.

Elapsed: 00:00:00.03

Session altered.

Elapsed: 00:00:00.04

no rows selected

Elapsed: 00:01:00.17

Execution Plan
----------------------------------------------------------
Plan hash value: 2134347679

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    33 |   178K  (1)| 00:35:43 |
|   1 |  HASH UNIQUE       |      |     1 |    33 |   178K  (1)| 00:35:43 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |    33 |   178K  (1)| 00:35:43 |
---------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("STATUS"='NONE')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
     651498  consistent gets
     651470  physical reads
          0  redo size
        399  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


Session altered.

Elapsed: 00:00:00.00

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
T1                               50017435     652594          88
T2

INDEX_NAME                         BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------------------------ ---------- ----------- ------------- ----------------------- ----------------------- ----------
-------
T1_IND1                                 3      288099      50108357                       1                       1
51187710
T2_IND1


Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.
```

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 7, 2008 12:02 AM in response to: Charles Hooper

Reply

```
TKPROF output with direct comparision between the 16KB and 8KB block size runs:

Test 1 16KB:
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     30.57      85.72     135072     135703          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     30.57      85.74     135073     135705          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
```

```
   9454  SORT UNIQUE NOSORT (cr=135703 pr=135072 pw=0 time=85245437 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=135703 pr=135072 pw=0 time=100008470 us)(object id 11767)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     632        0.00          0.00
  db file sequential read                    135072        0.04         56.86
  SQL*Net message from client                   632        0.01          2.79
********************************************************************************

Test 4 8KB:
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     34.12     100.63     274233     274646          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     34.12     100.65     274234     274648          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=274646 pr=274233 pw=0 time=111328538 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274646 pr=274233 pw=0 time=100020266 us)(object id 11767)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     632        0.00          0.00
  db file scattered read                       6952        0.02          6.44
  db file sequential read                    225942        0.03         63.97
  SQL*Net message from client                   632        0.02          2.78
********************************************************************************


Test 1 16KB:
********************************************************************************
SELECT
  *
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          1          1          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     10.56      71.82     320429     321597          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     10.56      71.84     320430     321598          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS FULL T1 (cr=321597 pr=320429 pw=0 time=71828655 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                         1        0.01          0.01
  SQL*Net message to client                       1        0.00          0.00
  db file scattered read                       5085        0.05         62.14
  SQL*Net message from client                     1        0.00          0.00

10046 Trace file:
PARSE #14:c=109375,e=1035690,p=1140,cr=98,cu=0,mis=1,r=0,dep=0,og=1,tim=2106644614
EXEC #14:c=0,e=28,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=2106644794
WAIT #14: nam='SQL*Net message to client' ela= 3 driver id=1413697536 #bytes=1 p3=0 obj#=11766 tim=2106644834
WAIT #14: nam='db file scattered read' ela= 22393 file#=4 block#=8 blocks=5 obj#=11766 tim=2106667326
WAIT #14: nam='db file scattered read' ela= 1186 file#=4 block#=13 blocks=4 obj#=11766 tim=2106668693
WAIT #14: nam='db file scattered read' ela= 2310 file#=4 block#=17 blocks=4 obj#=11766 tim=2106671140
WAIT #14: nam='db file scattered read' ela= 6560 file#=4 block#=22 blocks=3 obj#=11766 tim=2106677823
WAIT #14: nam='db file scattered read' ela= 594 file#=4 block#=25 blocks=4 obj#=11766 tim=2106678522
WAIT #14: nam='db file scattered read' ela= 11402 file#=4 block#=29 blocks=4 obj#=11766 tim=2106690064
WAIT #14: nam='db file scattered read' ela= 599 file#=4 block#=33 blocks=4 obj#=11766 tim=2106690801
WAIT #14: nam='db file scattered read' ela= 17327 file#=4 block#=38 blocks=3 obj#=11766 tim=2106708379
WAIT #14: nam='db file scattered read' ela= 585 file#=4 block#=41 blocks=4 obj#=11766 tim=2106709105
WAIT #14: nam='db file scattered read' ela= 640 file#=4 block#=45 blocks=4 obj#=11766 tim=2106709873
WAIT #14: nam='db file scattered read' ela= 585 file#=4 block#=49 blocks=4 obj#=11766 tim=2106710597
WAIT #14: nam='db file scattered read' ela= 604 file#=4 block#=54 blocks=3 obj#=11766 tim=2106711317
WAIT #14: nam='db file scattered read' ela= 613 file#=4 block#=57 blocks=4 obj#=11766 tim=2106712028
WAIT #14: nam='db file scattered read' ela= 665 file#=4 block#=61 blocks=4 obj#=11766 tim=2106712816
WAIT #14: nam='db file scattered read' ela= 574 file#=4 block#=65 blocks=4 obj#=11766 tim=2106713517
WAIT #14: nam='db file scattered read' ela= 26634 file#=4 block#=70 blocks=63 obj#=11766 tim=2106740385
WAIT #14: nam='db file scattered read' ela= 20449 file#=4 block#=134 blocks=63 obj#=11766 tim=2106762748
WAIT #14: nam='db file scattered read' ela= 26011 file#=4 block#=198 blocks=63 obj#=11766 tim=2106790618
WAIT #14: nam='db file scattered read' ela= 28744 file#=4 block#=262 blocks=63 obj#=11766 tim=2106821296
WAIT #14: nam='db file scattered read' ela= 26001 file#=4 block#=326 blocks=63 obj#=11766 tim=2106849172
WAIT #14: nam='db file scattered read' ela= 30236 file#=4 block#=390 blocks=63 obj#=11766 tim=2106881297
...
WAIT #14: nam='db file scattered read' ela= 13668 file#=4 block#=321737 blocks=64 obj#=11766 tim=2178408686
WAIT #14: nam='db file scattered read' ela= 10157 file#=4 block#=321801 blocks=64 obj#=11766 tim=2178420732
WAIT #14: nam='db file scattered read' ela= 10221 file#=4 block#=321865 blocks=64 obj#=11766 tim=2178432836
WAIT #14: nam='db file scattered read' ela= 11175 file#=4 block#=321929 blocks=64 obj#=11766 tim=2178445891
WAIT #14: nam='db file scattered read' ela= 10204 file#=4 block#=321993 blocks=64 obj#=11766 tim=2178457994
WAIT #14: nam='db file scattered read' ela= 10203 file#=4 block#=322057 blocks=64 obj#=11766 tim=2178470070
WAIT #14: nam='db file scattered read' ela= 1341 file#=4 block#=322121 blocks=12 obj#=11766 tim=2178473204
FETCH #14:c=10562500,e=71828658,p=320429,cr=321597,cu=0,mis=0,r=0,dep=0,og=1,tim=2178473533
WAIT #14: nam='SQL*Net message from client' ela= 634 driver id=1413697536 #bytes=1 p3=0 obj#=11766 tim=2178474254
STAT #14 id=1 cnt=0 pid=0 pos=1 obj=11766 op='TABLE ACCESS FULL T1 (cr=321597 pr=320429 pw=0 time=71828655 us)'
********************************************************************************

Test 4 8KB:
********************************************************************************
SELECT
  *
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          1          1          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     12.28      60.24     648725     651498          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     12.28      60.26     648726     651499          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS FULL T1 (cr=651498 pr=648725 pw=0 time=60248818 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                         2        0.01          0.01
  SQL*Net message to client                       1        0.00          0.00
  db file scattered read                       5140        0.05         48.58
  SQL*Net message from client                     1        0.01          0.01

10046 Trace file:
PARSE #13:c=62500,e=960065,p=2745,cr=94,cu=0,mis=1,r=0,dep=0,og=1,tim=999346046
EXEC #13:c=0,e=28,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=999346223
WAIT #13: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11766 tim=999346263
WAIT #13: nam='db file scattered read' ela= 14292 file#=4 block#=12 blocks=5 obj#=11766 tim=999360658
WAIT #13: nam='db file scattered read' ela= 910 file#=4 block#=17 blocks=8 obj#=11766 tim=999361715
WAIT #13: nam='db file scattered read' ela= 18546 file#=4 block#=26 blocks=7 obj#=11766 tim=999380403
WAIT #13: nam='db file scattered read' ela= 935 file#=4 block#=33 blocks=8 obj#=11766 tim=999381468
WAIT #13: nam='db file scattered read' ela= 554 file#=4 block#=42 blocks=7 obj#=11766 tim=999382162
WAIT #13: nam='db file scattered read' ela= 623 file#=4 block#=49 blocks=8 obj#=11766 tim=999382913
WAIT #13: nam='db file scattered read' ela= 644 file#=4 block#=58 blocks=7 obj#=11766 tim=999383699
WAIT #13: nam='db file scattered read' ela= 680 file#=4 block#=65 blocks=8 obj#=11766 tim=999384505
WAIT #13: nam='db file scattered read' ela= 553 file#=4 block#=74 blocks=7 obj#=11766 tim=999385198
WAIT #13: nam='db file scattered read' ela= 626 file#=4 block#=81 blocks=8 obj#=11766 tim=999385950
WAIT #13: nam='db file scattered read' ela= 569 file#=4 block#=90 blocks=7 obj#=11766 tim=999386662
WAIT #13: nam='db file scattered read' ela= 677 file#=4 block#=97 blocks=8 obj#=11766 tim=999387466
WAIT #13: nam='db file scattered read' ela= 587 file#=4 block#=106 blocks=7 obj#=11766 tim=999388196
WAIT #13: nam='db file scattered read' ela= 634 file#=4 block#=113 blocks=8 obj#=11766 tim=999388956
WAIT #13: nam='db file scattered read' ela= 651 file#=4 block#=122 blocks=7 obj#=11766 tim=999389744
WAIT #13: nam='db file scattered read' ela= 696 file#=4 block#=129 blocks=8 obj#=11766 tim=999390576
WAIT #13: nam='db file scattered read' ela= 13029 file#=4 block#=139 blocks=126 obj#=11766 tim=999403957
WAIT #13: nam='db file scattered read' ela= 27025 file#=4 block#=267 blocks=126 obj#=11766 tim=999433238
...
WAIT #13: nam='db file scattered read' ela= 9012 file#=4 block#=652177 blocks=128 obj#=11766 tim=1059567202
WAIT #13: nam='db file scattered read' ela= 8046 file#=4 block#=652305 blocks=128 obj#=11766 tim=1059577523
WAIT #13: nam='db file scattered read' ela= 10406 file#=4 block#=652433 blocks=128 obj#=11766 tim=1059590304
WAIT #13: nam='db file scattered read' ela= 2113 file#=4 block#=652561 blocks=42 obj#=11766 tim=1059594505
FETCH #13:c=12281250,e=60248822,p=648725,cr=651498,cu=0,mis=0,r=0,dep=0,og=1,tim=1059595125
WAIT #13: nam='SQL*Net message from client' ela= 11442 driver id=1413697536 #bytes=1 p3=0 obj#=11766 tim=1059606663
*** SESSION ID:(215.5) 2008-06-06 20:30:27.109
STAT #13 id=1 cnt=0 pid=0 pos=1 obj=11766 op='TABLE ACCESS FULL T1 (cr=651498 pr=648725 pw=0 time=60248818 us)'
********************************************************************************


Test 1 16KB:
********************************************************************************
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       16      0.00       0.09          5         10          0           0
Execute     17      0.00       0.11         14        136          8           8
Fetch      642     41.40     159.09     458826     463952          2        9498
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      675     41.40     159.30     458845     464098         10        9506

Misses in library cache during parse: 9
Misses in library cache during execute: 3

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     668        0.00          0.00
  SQL*Net message from client                   668        0.01          2.81
  db file sequential read                    135098        0.04         57.06
  db file scattered read                       5152        0.05         63.23
  db file parallel read                           1        0.10          0.10
********************************************************************************


Test 4 8KB:
********************************************************************************
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
```

```
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       16      0.00       0.06          5         10          0           0
Execute     17      0.01       0.11         18        142         10           8
Fetch      642     46.75     162.55     929930     940075          2        9498
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      675     46.76     162.73     929953     940227         12        9506

Misses in library cache during parse: 9
Misses in library cache during execute: 3

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     668        0.00          0.00
  SQL*Net message from client                   668        0.02          2.81
  db file sequential read                    225979        0.03         64.20
  db file scattered read                      12216        0.05         56.04
  db file parallel read                           1        0.31          0.31
********************************************************************************

Test 1 16KB:
********************************************************************************
SELECT
  COUNT(*)
FROM
  T2

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.03          2          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.25       1.53       3325       6652          2           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.25       1.56       3327       6654          2           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=6652 pr=3325 pw=0 time=1535095 us)
1000000   INDEX FAST FULL SCAN T2_IND1 (cr=6652 pr=3325 pw=0 time=6170385 us)(object id 11769)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         4        0.01          0.05
  db file parallel read                           1        0.10          0.10
  db file scattered read                         67        0.04          1.09
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************

Test 4 8KB:
********************************************************************************
SELECT
  COUNT(*)
FROM
  T2

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          2          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.34       1.66       6972      13931          2           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.34       1.68       6974      13933          2           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=13931 pr=6972 pw=0 time=1669507 us)
1000000   INDEX FAST FULL SCAN T2_IND1 (cr=13931 pr=6972 pw=0 time=2363377 us)(object id 11769)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         8        0.01          0.04
  db file parallel read                           1        0.31          0.31
  db file scattered read                        124        0.03          1.00
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************


Test 2 16KB:
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.06       0.15          0          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       95     78.84     139.14     135069     135166          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       97     78.90     139.29     135069     135168          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
```

```
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=135166 pr=135069 pw=0 time=139105318 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=135166 pr=135069 pw=0 time=100048754 us)(object id 11767)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                      95        0.00          0.00
  db file sequential read                    135069        0.03         61.86
  SQL*Net more data to client                    84        0.00          0.00
  SQL*Net message from client                    95        0.11          0.16
********************************************************************************

Test 5 8KB:
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.06       0.19          2          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       95     84.10     158.06     274016     274110          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       97     84.17     158.25     274018     274112          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=274110 pr=274016 pw=0 time=158024102 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274110 pr=274016 pw=0 time=100078077 us)(object id 11767)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                      95        0.00          0.00
  db file sequential read                    274016        0.03         77.68
  SQL*Net more data to client                    84        0.00          0.00
  SQL*Net message from client                    95        0.68          0.73
********************************************************************************


Test 3 16KB:
********************************************************************************
SELECT DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      9.75      75.28     321569     321597          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      9.75      75.30     321569     321597          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  HASH UNIQUE (cr=321597 pr=321569 pw=0 time=75282593 us)
      0   TABLE ACCESS FULL T1 (cr=321597 pr=321569 pw=0 time=75282461 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  db file sequential read                         1        0.01          0.01
  db file scattered read                       5048        0.06         65.94
  SQL*Net message from client                     1        0.03          0.03
********************************************************************************

Test 6 8KB:
********************************************************************************
SELECT DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     11.29      59.91     651470     651498          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     11.29      59.94     651470     651498          0           0
```

```
Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  HASH UNIQUE (cr=651498 pr=651470 pw=0 time=59918740 us)
      0   TABLE ACCESS FULL T1 (cr=651498 pr=651470 pw=0 time=59918627 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  db file sequential read                         1        0.01          0.01
  db file scattered read                       5114        0.05         48.51
  SQL*Net message from client                     1        0.02          0.02
********************************************************************************
```

I also have more extensive analysis files generated from the 10046 trace files.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Aman....

Posts: 3,145
From: India
Registered: 5/21/01

Re: Larger vs. Small data block
Posted: Jun 7, 2008 12:05 AM in response to: Charles Hooper

Reply

By far ,one of the best threads !
Excellent!
Best regards
Aman....

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 7, 2008 12:38 AM in response to:

Reply

>...but I don't see you enlightening us with a valid test.
> I expect a test showing the performance differences with different blocksizes (measuring throughput and response time).

And there is not a test from you either, but perhaps there will be next week when you are less busy. =)
May I suggest not to hold others to a higher standard than you hold yourself to.

Isolated and controlled experiments are *very* meaningful if constructed correctly, generally as meaningful as a real-world workload, because they are usually modeled after one. Often times it is about taking a complex problem and simplifying it so that it can be understood, and then confirming that the observations made in isolation are also pertinent in the original situation.

--
Regards,

Greg Rahn
http://structureddata.org

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 7, 2008 12:59 AM in response to:

Reply

> >> I didn't convince my self with your answer.
>
> Me neither . . . .
>

And this post did not convince me either as it contains nothing but hearsay. There is not a single "sighting" that contains enough technical detail for anyone to determine its validity, including yourself.

I might suggest that you follow the scientific method in obtaining your empirical results to support your hypothesis.
http://www.sciencebuddies.org/mentoring/project_scientific_method.shtml
Once you have conducted your experiment, post your work, and we can discuss the results.

--
Regards,

Greg Rahn
http://structureddata.org

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 7, 2008 2:24 AM in response to: sp009

Reply

I think I have run an similiar experiment, taking a join into consideration. I used the same WEB_RETURNS table that I used in my other experiments and used the following query:

```
select count(*)
from WEB_RETURNS a, WEB_RETURNS b
where a.WR_ORDER_NUMBER = b.WR_ORDER_NUMBER
```

I have run the experiment on both a 8k block table in a 8k block database and 16k block table in a 16k block database and there appears to be no difference in elapsed times (24.59 for the 8k and 24.65 for the 16k). In each case the buffer cache is cold. Storage is ASM. Version 11.1.0.6 on 32-bit Linux.

8k experiment

```
select count(*)
from WEB_RETURNS_8K a, WEB_RETURNS_8K b
where a.WR_ORDER_NUMBER = b.WR_ORDER_NUMBER

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
```

```
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     21.24      24.59     244014     226324          0           1
-------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     21.25      24.59     244014     226324          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 50

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=226324 pr=244014 pw=244014 time=0 us)
15516562   HASH JOIN  (cr=226324 pr=244014 pw=244014 time=198610 us cost=74796 size=145886544 card=12157212)
7197670    TABLE ACCESS FULL WEB_RETURNS_8K (cr=113162 pr=113157 pw=113157 time=73018 us cost=31134 size=43186020
card=7197670)
7197670    TABLE ACCESS FULL WEB_RETURNS_8K (cr=113162 pr=113156 pw=113156 time=71056 us cost=31134 size=43186020
card=7197670)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       4        0.00          0.00
  SQL*Net message from client                     4        0.00          0.00
  db file sequential read                         1        0.01          0.01
  direct path read                             1998        0.04          2.55
  direct path write temp                        571        0.01          0.79
  direct path read temp                         571        0.00          0.11
```

16k experiment

```
select count(*)
from WEB_RETURNS_16K a, WEB_RETURNS_16K b
where a.WR_ORDER_NUMBER = b.WR_ORDER_NUMBER

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     21.29      24.65     120793     111654          0           1
-------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     21.29      24.65     120793     111654          0           1

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 28

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=111654 pr=120793 pw=120793 time=0 us)
15516562   HASH JOIN  (cr=111654 pr=120793 pw=120793 time=205948 us cost=53562 size=145886544 card=12157212)
7197670    TABLE ACCESS FULL WEB_RETURNS_16K (cr=55827 pr=55822 pw=55822 time=56183 us cost=21362 size=43186020 card=7197670)
7197670    TABLE ACCESS FULL WEB_RETURNS_16K (cr=55827 pr=55821 pw=55821 time=56739 us cost=21362 size=43186020 card=7197670)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       4        0.00          0.00
  SQL*Net message from client                     4        0.00          0.00
  db file sequential read                         1        0.01          0.01
  direct path read                             1962        0.02          2.65
  direct path write temp                        610        0.00          0.69
  direct path read temp                         610        0.00          0.11
```

--
Regards,

Greg Rahn
http://structureddata.org

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 7, 2008 4:23 AM in response to: Greg Rahn

Reply

Greg,

You do realise that this is the wrong result, so clearly you'll have to do it again !!

This time make sure you wipe the first database from the system before creating the second so that they occupy the same space on disc. Your 16K database was probably created second, which put it nearer the middle of your disc drives - which would have made the I/Os slower, thus increasing the CPU time spent in I/O waits.

The previous two paragraphs were intended to be ironic, by the way; but on a more serious note I'd also like to point out an important detail relating to the general DW vs. OLTP argument about block sizing. You're using 11g, and Oracle has gone to serial direct I/O in your case, bypassing the buffer cache - and for DW activity that may very well be the optimum strategy.

On the other hand, for some of the tests that people do (the simple *count(*)*, for example) it is the work involved in hitting the cache-related latches that is the most significant contributor to the CPU load.

If you disable serial reads, I think your test might just nudge the CPU balance in the direction of the 16K block size.

In passing - your tablescans are showing ***pw*** = ***pr*** every time. This looks like a bug.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Re: Larger vs. Small data block
Posted: Jun 7, 2008 8:16 AM in response to: Greg Rahn

Reply

Re: Larger vs. Small data block
Posted: Jun 7, 2008 8:39 AM in response to: Greg Rahn

Reply

Re: Larger vs. Small data block
Posted: Jun 7, 2008 8:51 AM in response to: Jonathan Lewis

Reply

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 7, 2008 9:25 AM in response to:

Reply

> Precisely! If you check my cited hyperlinks, they
> are direct reports from real, practicing Oracle
> DBA's.
>

Let's go through each of these links shall we ...

Tod Boss, who made the quote on an oracle-l list 4 years ago. Not much detail to go on really, certainly no measures of dispersion here. Still, a quote's a quote.

M. J. Schwenger: If you read the thread carefully, begins by asking "My question is: Am I going to get better performance if I move the indexes to the 32K blocksize as I'm expecting? " as he has doubt about it all and the answer from those helping (including among others **David Aldridge** and **Balkrishan Mittal**, both coming up) is to focus tuning efforts elsewhere ...

Balkrishan Mittal: Recommends **not** moving indexes to a bigger block size because when he tried it "My servers CPU usage went to 100% (all the time). After bearing it for two days i again restored indx tablespace to 4k block size".

David Aldridge: Who disagreed with you that the 6% difference had anything to do with different block sizes and was subsequently banned from your forum as a result - http://oraclesponge.blogspot.com/2005_04_01_archive.html

Chris Foot: Links points to an OCP Instructors Guide ???

Santosh Kumar: Didn't note anything himself but got the quote from an anonymous Russian on an AskTom thread, which even you dismissed "Yeah, I redacted that one" on this thread
http://forums.oracle.com/forums/thread.jspa?threadID=566662&tstart=15&start=12.

Steve Taylor: Who in the same forum discussion that got David Aldridge banned where his quote originated:

http://dba.ipbhost.com/index.php?showtopic=1239&st=75

says "Now from reading the thread - some of this could have been a result of external factors such as the cache segregation... We didn't do a great deal of granular tests just run the typical product cycle against the system, as there were just too many queries that could be generated. But I'm thinking now would be a perfect time to revisit... Sorry if this sounds a bit vague.. and I think I've learnt a couple of valuable lessons here..."

OK, I guess it's time to make my own judgement ...

Cheers :)

Richard Foote
http://richardfoote.wordpress.com/

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 7, 2008 9:35 AM in response to:

Reply

Note the median of x or even the most frequent value of x are also possible examples of central tendency.

Note also that variance and standard deviation are also measures of dispersion.

Had to look them up of course ;)

Cheers

Richard Foote

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 7, 2008 9:55 AM in response to: Charles Hooper

Reply

Hi Charles

Nice work :)

This might sound like a somewhat silly suggestion but it would be interesting (to me anyways) if you repeated the tests on the two different databases but with them both having the same block sizes.

What would be the differences if the block sizes were identical because the databases would still differ by having different files on different parts of the file system.

Just a thought.

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Charles Hooper

Posts: 228
From: USA

Re: Larger vs. Small data block
Posted: Jun 7, 2008 10:35 AM in response to: Richard Foote

Reply

> Hi Charles

Registered: 1/27/08

```
>
> Nice work :)
>
> This might sound like a somewhat silly suggestion but
> it would be interesting (to me anyways) if you
> repeated the tests on the two different databases but
> with them both having the same block sizes.
>
> What would be the differences if the block sizes were
> identical because the databases would still differ by
> having different files on different parts of the file
> system.
>
> Just a thought.
>
> Cheers
>
> Richard Foote
> http://richardfoote.wordpress.com/
```

Richard,

Give me another 7 hours or so to repeat the test, and I will rebuild the 16KB database as a 8KB database to repeat the test. For the test runs, I created the 8KB database first, rebooted, and then created the 16KB database. There were a couple interesting results - it appears in the 10046 trace file that Oracle started the full tablescan reading just a couple blocks at a time (64KB) and then increased to a much larger number of blocks read at the same time (1024KB). The variation in the read times in the raw trace files possibly show the effects of native command queuing supported by the SATA drives in RAID 0 and the effects of the 8MB buffer built into the drives.

If you compare side-by-side (in a spreadsheet) the elapsed times for test run 1 with those of test run 4 (also compare the elapsed times in test run 3 with test run6), you will see interesting results, like the following:

It required:
* Less time to build T1 in the 8KB block size: 1:41.48 vs. 00:01:48.15
* Less time to build the index on T1 in the 8KB block size: 8:28.31 vs. 10:30.96
* Less time to insert into T2 with an existing index in the 8KB block size: 1:53.59 vs. 2:08.28
* Recursive calls appears to be less time consuming in the 8KB block size: the table access full
 (STATUS='NONE') required 1:01.21 vs. 1:12.87 (the trace file seems to imply the opposite, but I excluded the
 recursive calls from the report I posted).
* Less time for statistics gathering on T1 in the 8KB block size: (2:12.53, 2:01.07) vs. (2:30.67, 2:30.07)
* ...

I forgot to mention, the test was run on a 3.5 year old Dell XPS Gen 4 system with the BIOS set to show a blue
colored neon light tube on the front of the system. I will change the color to red to see if it makes a difference - I am a
little surprised that someone did
not ask me the current color of the neon light tube, as
we know blue favors 8KB block sizes, and red favors 16KB block sizes ;-) On second thought, maybe I should set
it to yellow so that a scientific method is followed for the procedure.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Corrected word-wrapping problem
Message was edited by:
Charles Hooper

Re: Larger vs. Small data block
Posted: Jun 7, 2008 10:53 AM in response to: Charles Hooper

Reply

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 7, 2008 12:38 PM in response to: Charles Hooper

Reply

```
>
> On the 16KB block size database, Oracle automatically
> set the DB_FILE_MULTIBLOCK_READ_COUNT=64
>
> On the 8KB block size database, Oracle automatically
> set the DB_FILE_MULTIBLOCK_READ_COUNT=128
>
> The above surprised me a bit.
>
```

Charles,

Thanks for taking the time to do something like this.

The variation in db_file_multiblock_read_count is to be expected in your 10.2.0.2. Oracle tries to go for the largest possible read, with a limit imposed by (a) the operating system - which is often 1MByte and (b) db_cache_size/sessions. Since you have an sga_target of 900Mb and processes = 210, Oracle must have decided that 1Mb was viable.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 7, 2008 12:46 PM in response to: Charles Hooper

Reply

```
> There were a couple interesting
> results - it appears in the 10046 trace file that
> Oracle started the full tablescan reading just a
> couple blocks at a time (64KB) and then increased to
> a much larger number of blocks read at the same time
> (1024KB).
```

I'll take a guess on that - do you have system managed extent allocation ?
In clean tablespaces the first 16 would be 64KB each, the next 63 would be 1MB each, and then I can't remember the next size up, or how many there would be.

The other variation in the sizes of the first few reads would relate to the effects of ASSM - the first few blocks of the first extent are bitmap blocks, and then you get the odd extra bitmap block at the start of some of the later extents.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Re: Larger vs. Small data block
Posted: Jun 7, 2008 2:42 PM in response to: Madrid
Reply

Charles Hooper
Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 7, 2008 5:11 PM in response to: Jonathan Lewis
Reply

> >There were a couple
> interesting
> > results - it appears in the 10046 trace file that
> > Oracle started the full tablescan reading just a
> > couple blocks at a time (64KB) and then increased
> to
> > a much larger number of blocks read at the same
> time
> > (1024KB).
>
> I'll take a guess on that - do you have system
> managed extent allocation ?
> In clean tablespaces the first 16 would be 64KB each,
> the next 63 would be 1MB each, and then I can't
> remember the next size up, or how many there would
> be.
> The other variation in the sizes of the first few
> reads would relate to the effects of ASSM - the
> first few blocks of the first extent are bitmap
> blocks, and then you get the odd extra bitmap block
> at the start of some of the later extents.
>
> Regards
> Jonathan Lewis
> http://jonathanlewis.wordpress.com
> http://www.jlcomp.demon.co.uk

Yes, the scripts that I posted set up the data file "USER_DATA" with the following:
EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO

I though that the database instance was recognizing that the operating system was efficiently reading 64KB, so it decided to ramp up the read size to 1MB. Your explanation is very likely the cause of the read size changing. I knew that rule some time ago, but it caught me a bit surprised as I was thinking that the maximum read size on Windows was closer to 64KB, 128KB or 256KB.

I just completed 2 more test runs and will be posting the results shortly. The original 8KB database was created first, and the original 16KB database was built second. The original 16KB tablespace data file was specified at 2GB, and grew to 7,577,616KB. The original 8KB tablespace data file was specified at 2GB, and grew to 7,680,008KB. I left the original 8KB database in place, but did not start it.

For tests 7, 8, 9 (8KB block size):
I used ORADIM to delete the TEST16 instance, and then I deleted all control, data, and trace files related to that database, then I restarted the computer. Changing just the init.ora file to specify an 8KB block size, I re-ran the scripts to create the TEST16 database instance with a 8KB block size. I then followed exactly the same procedure as before for running the same test scripts and captured the output. Tests 7, 8, 9 relate to tests 4, 5, 6, respectively.

For tests 10, 11, 12 (16KB block size):
I used ORADIM to delete the TEST16 instance, and then I deleted all control, data, and trace files related to that database, then I restarted the computer. Changing just the init.ora file to specify an 16KB block size, I re-ran the scripts to create the TEST16 database instance with a 16KB block size. I then followed exactly the same procedure as before for running the same test scripts and captured the output. Tests 10, 11, 12 relate to tests 1, 2, 3, respectively.

The new test results will follow.

To minimize the number of changes made to the computer setup, I left the neon tube color at blue. :-)

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Greg Rahn
Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 7, 2008 5:15 PM in response to:
Reply

> Oracle performance is observed by recording response time and/or throughput.
>
> Why doesn't your test thingy measure performance?
>

My experiments include the elapsed time. Is that not a performance metric based on your first sentence?

>
> Also, I would expect any "test" by an Oracle Corporation employee to use your own scientific analysis tool, ODM, and to use statistically valid scientific methods.
>

Would you please demonstrate exactly how Oracle Data Mining would be used in this case? Personally I don't see how it is applicable here.

Based on your comments, it seems that you have a misunderstanding of how Scientific Method applies in this case. Using Scientific Method *does not* mean we need variance and standard deviation. We can apply Scientific Method to every day problems. For instance: *"What do you do when your telephone doesn't work? Is the problem in the hand set, the cabling inside your house, the hookup outside, or in the workings of the phone company? The process you might go through to solve this problem could involve scientific thinking, and the results might contradict your initial expectations."*[1]

And what I would expect from "one of the world's leading Oracle experts"[2] is something more that cheering and jeering from the sideline. How about you lead by example? Create your experiment and show your results and let others be as critical about you as you are about them. If you do not want to participate in the experiments, then I think it is reasonable that you refrain from the criticism.

>
> Oracle is a large set of computer programs, written by humans. IT'S NOT A SCIENCE!
>

I don't think anyone is defining Oracle software as a science. None the less the scientific method **can** be applied to it: *"Like any good scientist, you may question the range of situations (**outside of science**) in which the scientific method may be applied. From what has been stated above, we determine that the scientific method works best in situations where one can isolate the phenomenon of interest, by eliminating or accounting for extraneous factors, and where one can repeatedly test the system under study after making limited, controlled changes in it."*[2]

The experiments that have been conducted in this thread are about understanding cause and effect in specific situations. It also is about controlled environments and understanding the effects if a given variable is modified. The participating parties are interested in further understanding under what situations block size matters and **why** it matters.

--
Regards,

Greg Rahn
http://structureddata.org

[1] http://teacher.pas.rochester.edu/phy_labs/appendixe/appendixe.html
[2] http://www.dba-oracle.com/resume_don.htm

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 7, 2008 5:20 PM in response to: Richard Foote

Reply

> This might sound like a somewhat silly suggestion but
> it would be interesting (to me anyways) if you
> repeated the tests on the two different databases but
> with them both having the same block sizes.
>
> What would be the differences if the block sizes were
> identical because the databases would still differ by
> having different files on different parts of the file
> system.
>
> Just a thought.
>
> Cheers
>
> Richard Foote
> http://richardfoote.wordpress.com/

Richard,

That was a very good suggestion. I first removed the database instance and related files for the 16KB block size database, built an 8KB block size database in its place, tested, removed the new 8KB database instance, built a new 16KB block size database in its place, and tested again. The testing followed exactly the same procedure as before. The initial results are below. Test runs 7, 8, and 9 are for the new 8KB database, test runs 10, 11, and 12 are for the 16KB database:

```
################ RESULTS ################
#TEST RUN 7 8KB:
  COUNT(*)
----------
     11073

Elapsed: 00:00:00.65

Execution Plan...

Statistics
----------------------------------------------------------
        641  recursive calls
          0  db block gets
      19569  consistent gets
        378  physical reads
         72  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         25  sorts (memory)
          0  sorts (disk)
          1  rows processed


Table created.

Elapsed: 00:01:53.48

Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:02.51

System altered.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:08:56.64

Table created.

Elapsed: 00:00:01.01

Index created.

Elapsed: 00:00:00.01
```

```
System altered.

Elapsed: 00:00:00.86

System altered.

Elapsed: 00:00:00.01

1000000 rows created.

Elapsed: 00:02:08.21

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |      |   776K|   104M|   178K  (2)| 00:35:47 |
|*  1 |  TABLE ACCESS FULL| T1   |   776K|   104M|   178K  (2)| 00:35:47 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RN"<=100)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       8295  recursive calls
    2855691  db block gets
     713243  consistent gets
     651602  physical reads
  470340500  redo size
        681  bytes sent via SQL*Net to client
        583  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
    1000000  rows processed


Commit complete.

Elapsed: 00:00:00.01

System altered.

Elapsed: 00:00:18.18

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.04

no rows selected

Elapsed: 00:01:12.59

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |  7180 |   988K|   178K  (2)| 00:35:44 |
|*  1 |  TABLE ACCESS FULL| T1   |  7180 |   988K|   178K  (2)| 00:35:44 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("STATUS"='NONE')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     651592  consistent gets
     651470  physical reads
          0  redo size
       1047  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


  COUNT(*)
----------
   1000000

Elapsed: 00:00:02.45
```

```
Execution Plan
----------------------------------------------------------
Plan hash value: 1385691034

-------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |     1 |  1863   (1)| 00:00:23 |
|   1 |  SORT AGGREGATE       |         |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| T2_IND1 |   798K|  1863   (1)| 00:00:23 |
-------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         32  recursive calls
          3  db block gets
      14159  consistent gets
       7746  physical reads
     506724  redo size
        411  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


OWNER                          OBJECT_NAME
------------------------------ ------------------------------
SUBOBJECT_NAME
------------------------------

9454 rows selected.

Elapsed: 00:01:42.18

Execution Plan
----------------------------------------------------------
Plan hash value: 1118578911

------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |    50M|  2458M|   921K  (1)| 03:04:19 |
|   1 |  SORT UNIQUE NOSORT|         |    50M|  2458M|   921K  (1)| 03:04:19 |
|   2 |   INDEX FULL SCAN  | T1_IND1 |    50M|  2458M|   276K  (1)| 00:55:24 |
------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     274740  consistent gets
     274369  physical reads
          0  redo size
     299162  bytes sent via SQL*Net to client
       7311  bytes received via SQL*Net from client
        632  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9454  rows processed


Session altered.

Elapsed: 00:00:00.00
```

#**TEST RUN 7 8KB:**

```
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1;

-------------------------------------------------------------------------------------------------
| Id  | Operation          | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------
|   1 |  SORT UNIQUE NOSORT|         |      1 |     50M|   9454 |00:02:37.67 |     274K|    274K|
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     50M|     50M|00:01:40.04 |     274K|    274K|
-------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement
```

#**TEST RUN 9 8KB:**

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:36.67

PL/SQL procedure successfully completed.

Elapsed: 00:02:23.29

System altered.

Elapsed: 00:00:00.06
```

```
System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.03

no rows selected

Elapsed: 00:01:11.59

Execution Plan
----------------------------------------------------------
Plan hash value: 2134347679

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    32 |   178K  (2)| 00:35:43 |
|   1 |  HASH UNIQUE       |      |     1 |    32 |   178K  (2)| 00:35:43 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |    32 |   178K  (2)| 00:35:43 |
---------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("STATUS"='NONE')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
     651498  consistent gets
     651470  physical reads
          0  redo size
        399  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


Session altered.

Elapsed: 00:00:00.00

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
T1                               50072042     652594          88
T2


INDEX_NAME                         BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------------------------ ---------- ----------- ------------- ----------------------- ----------------------- ----------
-------
T1_IND1                                 3      267918      45713274                       1                       1
47110621
T2_IND1
```

#**TEST RUN 10 16KB:**

```
  COUNT(*)
----------
     11073

Elapsed: 00:00:00.62

Execution Plan...

Statistics
----------------------------------------------------------
        641  recursive calls
          0  db block gets
      19499  consistent gets
        209  physical reads
          0  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         25  sorts (memory)
          0  sorts (disk)
          1  rows processed


Table created.

Elapsed: 00:01:51.54

Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:02.21

System altered.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:08:40.39
```

```
Table created.

Elapsed: 00:00:01.09

Index created.

Elapsed: 00:00:00.01

System altered.

Elapsed: 00:00:00.71

System altered.

Elapsed: 00:00:00.01

1000000 rows created.

Elapsed: 00:01:42.42

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |      |   751K|   101M|   122K  (2)| 00:28:38 |
|*  1 |  TABLE ACCESS FULL| T1   |   751K|   101M|   122K  (2)| 00:28:38 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RN"<=100)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       7253  recursive calls
    2491314  db block gets
     352577  consistent gets
     321650  physical reads
  445453548  redo size
        681  bytes sent via SQL*Net to client
        583  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          4  sorts (memory)
          0  sorts (disk)
    1000000  rows processed


Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:14.45

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.03

no rows selected

Elapsed: 00:01:08.78

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |  3544 |   487K|   122K  (2)| 00:28:34 |
|*  1 |  TABLE ACCESS FULL| T1   |  3544 |   487K|   122K  (2)| 00:28:34 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("STATUS"='NONE')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     321695  consistent gets
     321569  physical reads
          0  redo size
       1047  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
  COUNT(*)
----------
   1000000

Elapsed: 00:00:02.62

Execution Plan
----------------------------------------------------------
Plan hash value: 1385691034

--------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |     1 |  1230   (1)| 00:00:18 |
|   1 |  SORT AGGREGATE       |         |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| T2_IND1 |   974K|  1230   (1)| 00:00:18 |
--------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         32  recursive calls
          3  db block gets
       6812  consistent gets
       4298  physical reads
     242000  redo size
        411  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


OWNER                          OBJECT_NAME
------------------------------ ------------------------------
SUBOBJECT_NAME
------------------------------

9454 rows selected.

Elapsed: 00:01:19.85

Execution Plan
----------------------------------------------------------
Plan hash value: 1118578911

-------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |    54M|  2666M|   574K  (1)| 02:14:01 |
|   1 |  SORT UNIQUE NOSORT|         |    54M|  2666M|   574K  (1)| 02:14:01 |
|   2 |   INDEX FULL SCAN  | T1_IND1 |    54M|  2666M|   136K  (1)| 00:31:51 |
-------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     135802  consistent gets
     135129  physical reads
          0  redo size
     299135  bytes sent via SQL*Net to client
       7311  bytes received via SQL*Net from client
        632  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9454  rows processed


Session altered.

Elapsed: 00:00:00.00
```

#**TEST RUN 11 16KB:**

```
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1;

------------------------------------------------------------------------------------------------
| Id  | Operation          | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
------------------------------------------------------------------------------------------------
|   1 |  SORT UNIQUE NOSORT|         |      1 |     54M|   9454 |00:02:10.37 |     135K|    135K|
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     54M|     50M|00:01:40.04 |     135K|    135K|
------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement
```

#**TEST RUN 12 16KB:**

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:30.61
```

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:29.34

System altered.

Elapsed: 00:00:00.03

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.03

no rows selected

Elapsed: 00:01:11.26

Execution Plan
----------------------------------------------------------
Plan hash value: 2134347679

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    33 |   122K  (2)| 00:28:32 |
|   1 |  HASH UNIQUE       |      |     1 |    33 |   122K  (2)| 00:28:32 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |    33 |   122K  (2)| 00:28:32 |
---------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("STATUS"='NONE')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
     321597  consistent gets
     321569  physical reads
          0  redo size
        399  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


Session altered.

Elapsed: 00:00:00.00

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
T1                               50275095     322128          88
T2


INDEX_NAME                         BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------------------------ ---------- ----------- ------------- ----------------------- ----------------------- ----------
-------
T1_IND1                                 2      138977      48810943                       1                       1
49496736
T2_IND1


TKPROF output will follow.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.
```

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 7, 2008 5:24 PM in response to: Charles Hooper

Reply

TKPROF output with direct comparison between the 8KB and 16KB block size runs:

Test 7 8KB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     33.71      99.25     274239     274645          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     33.71      99.27     274240     274647          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
```

```
   9454  SORT UNIQUE NOSORT (cr=274645 pr=274239 pw=0 time=100635543 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274645 pr=274239 pw=0 time=100036443 us)(object id 11757)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     632        0.00          0.00
  db file scattered read                       6922        0.02          5.10
  db file sequential read                    226153        0.02         63.43
  SQL*Net message from client                   632        0.01          2.77
********************************************************************************
```

Test 10 16KB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.00          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     29.46      76.99     135128     135703          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     29.48      77.00     135129     135705          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=135703 pr=135128 pw=0 time=76572511 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=135703 pr=135128 pw=0 time=50022973 us)(object id 11767)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     632        0.00          0.00
  db file sequential read                    113857        0.06         44.23
  db file scattered read                       7115        0.04          5.58
  SQL*Net message from client                   632        0.01          2.76
********************************************************************************
```

Test 7 8KB:

```
********************************************************************************
SELECT
  *
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          1          1          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     11.92      71.41     648732     651498          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     11.92      71.43     648733     651499          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS FULL T1 (cr=651498 pr=648732 pw=0 time=71414670 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                         2        0.01          0.01
  SQL*Net message to client                       1        0.00          0.00
  db file scattered read                       5140        0.05         59.73
  SQL*Net message from client                     1        0.01          0.01

10046 Trace file:
PARSE #8:c=46875,e=1167603,p=2738,cr=94,cu=0,mis=1,r=0,dep=0,og=1,tim=945576493
EXEC #8:c=0,e=27,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=945576674
WAIT #8: nam='SQL*Net message to client' ela= 3 driver id=1413697536 #bytes=1 p3=0 obj#=11756 tim=945576715
WAIT #8: nam='db file scattered read' ela= 10258 file#=4 block#=12 blocks=5 obj#=11756 tim=945587074
WAIT #8: nam='db file scattered read' ela= 15539 file#=4 block#=17 blocks=8 obj#=11756 tim=945602737
WAIT #8: nam='db file scattered read' ela= 1230 file#=4 block#=26 blocks=7 obj#=11756 tim=945604106
WAIT #8: nam='db file scattered read' ela= 556 file#=4 block#=33 blocks=8 obj#=11756 tim=945604792
WAIT #8: nam='db file scattered read' ela= 576 file#=4 block#=42 blocks=7 obj#=11756 tim=945605510
WAIT #8: nam='db file scattered read' ela= 551 file#=4 block#=49 blocks=8 obj#=11756 tim=945606191
WAIT #8: nam='db file scattered read' ela= 662 file#=4 block#=58 blocks=7 obj#=11756 tim=945606988
WAIT #8: nam='db file scattered read' ela= 556 file#=4 block#=65 blocks=8 obj#=11756 tim=945607672
WAIT #8: nam='db file scattered read' ela= 576 file#=4 block#=74 blocks=7 obj#=11756 tim=945608394
WAIT #8: nam='db file scattered read' ela= 741 file#=4 block#=81 blocks=8 obj#=11756 tim=945609263
WAIT #8: nam='db file scattered read' ela= 1259 file#=4 block#=90 blocks=7 obj#=11756 tim=945610664
WAIT #8: nam='db file scattered read' ela= 560 file#=4 block#=97 blocks=8 obj#=11756 tim=945611351
WAIT #8: nam='db file scattered read' ela= 538 file#=4 block#=106 blocks=7 obj#=11756 tim=945612034
WAIT #8: nam='db file scattered read' ela= 553 file#=4 block#=113 blocks=8 obj#=11756 tim=945612716
WAIT #8: nam='db file scattered read' ela= 667 file#=4 block#=122 blocks=7 obj#=11756 tim=945613521
WAIT #8: nam='db file scattered read' ela= 541 file#=4 block#=129 blocks=8 obj#=11756 tim=945614197
WAIT #8: nam='db file scattered read' ela= 11162 file#=4 block#=139 blocks=126 obj#=11756 tim=945625708
WAIT #8: nam='db file scattered read' ela= 11637 file#=4 block#=267 blocks=126 obj#=11756 tim=945639635
WAIT #8: nam='db file scattered read' ela= 9859 file#=4 block#=395 blocks=126 obj#=11756 tim=945651792
WAIT #8: nam='db file scattered read' ela= 10744 file#=4 block#=523 blocks=126 obj#=11756 tim=945664793
WAIT #8: nam='db file scattered read' ela= 9828 file#=4 block#=651 blocks=126 obj#=11756 tim=945676896
WAIT #8: nam='db file scattered read' ela= 12255 file#=4 block#=779 blocks=126 obj#=11756 tim=945691398
WAIT #8: nam='db file scattered read' ela= 10829 file#=4 block#=907 blocks=126 obj#=11756 tim=945704482
WAIT #8: nam='db file scattered read' ela= 9849 file#=4 block#=1035 blocks=126 obj#=11756 tim=945716588
...
WAIT #8: nam='db file scattered read' ela= 9841 file#=4 block#=651793 blocks=128 obj#=11756 tim=1016921816
WAIT #8: nam='db file scattered read' ela= 9825 file#=4 block#=651921 blocks=128 obj#=11756 tim=1016933916
WAIT #8: nam='db file scattered read' ela= 10742 file#=4 block#=652049 blocks=128 obj#=11756 tim=1016946981
WAIT #8: nam='db file scattered read' ela= 12264 file#=4 block#=652177 blocks=128 obj#=11756 tim=1016961540
WAIT #8: nam='db file scattered read' ela= 9726 file#=4 block#=652305 blocks=128 obj#=11756 tim=1016973607
WAIT #8: nam='db file scattered read' ela= 10801 file#=4 block#=652433 blocks=128 obj#=11756 tim=1016986700
WAIT #8: nam='db file scattered read' ela= 1990 file#=4 block#=652561 blocks=42 obj#=11756 tim=1016990801
FETCH #8:c=11921875,e=71414674,p=648732,cr=651498,cu=0,mis=0,r=0,dep=0,og=1,tim=1016991428
WAIT #8: nam='SQL*Net message from client' ela= 15789 driver id=1413697536 #bytes=1 p3=0 obj#=11756 tim=1017007310
STAT #8 id=1 cnt=0 pid=0 pos=1 obj=11756 op='TABLE ACCESS FULL T1 (cr=651498 pr=648732 pw=0 time=71414670 us)'
********************************************************************************
```

Test 10 16KB:

```
********************************************************************************
SELECT
  *
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          1          1          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      9.76      67.69     320423     321597          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      9.76      67.71     320424     321598          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS FULL T1 (cr=321597 pr=320423 pw=0 time=67692842 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                         1        0.01          0.01
  SQL*Net message to client                       1        0.00          0.00
  db file scattered read                       5085        0.05         58.16
  SQL*Net message from client                     1        0.02          0.02

10046 Trace File:
PARSE #14:c=93750,e=1064918,p=1146,cr=98,cu=0,mis=1,r=0,dep=0,og=1,tim=952554189
EXEC #14:c=0,e=28,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=952554367
WAIT #14: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11766 tim=952554408
WAIT #14: nam='db file scattered read' ela= 12323 file#=4 block#=8 blocks=5 obj#=11766 tim=952566858
WAIT #14: nam='db file scattered read' ela= 1193 file#=4 block#=13 blocks=4 obj#=11766 tim=952568220
WAIT #14: nam='db file scattered read' ela= 575 file#=4 block#=17 blocks=4 obj#=11766 tim=952568922
WAIT #14: nam='db file scattered read' ela= 641 file#=4 block#=22 blocks=3 obj#=11766 tim=952569678
WAIT #14: nam='db file scattered read' ela= 568 file#=4 block#=25 blocks=4 obj#=11766 tim=952570344
WAIT #14: nam='db file scattered read' ela= 828 file#=4 block#=29 blocks=4 obj#=11766 tim=952571296
WAIT #14: nam='db file scattered read' ela= 574 file#=4 block#=33 blocks=4 obj#=11766 tim=952571995
WAIT #14: nam='db file scattered read' ela= 596 file#=4 block#=38 blocks=3 obj#=11766 tim=952572716
WAIT #14: nam='db file scattered read' ela= 676 file#=4 block#=41 blocks=4 obj#=11766 tim=952573492
WAIT #14: nam='db file scattered read' ela= 1159 file#=4 block#=45 blocks=4 obj#=11766 tim=952574778
WAIT #14: nam='db file scattered read' ela= 572 file#=4 block#=49 blocks=4 obj#=11766 tim=952575486
WAIT #14: nam='db file scattered read' ela= 515 file#=4 block#=54 blocks=3 obj#=11766 tim=952576124
WAIT #14: nam='db file scattered read' ela= 567 file#=4 block#=57 blocks=4 obj#=11766 tim=952576793
WAIT #14: nam='db file scattered read' ela= 750 file#=4 block#=61 blocks=4 obj#=11766 tim=952577667
WAIT #14: nam='db file scattered read' ela= 577 file#=4 block#=65 blocks=4 obj#=11766 tim=952578373
WAIT #14: nam='db file scattered read' ela= 19168 file#=4 block#=70 blocks=63 obj#=11766 tim=952597773
WAIT #14: nam='db file scattered read' ela= 28313 file#=4 block#=134 blocks=63 obj#=11766 tim=952627962
WAIT #14: nam='db file scattered read' ela= 35142 file#=4 block#=198 blocks=63 obj#=11766 tim=952665097
WAIT #14: nam='db file scattered read' ela= 35259 file#=4 block#=262 blocks=63 obj#=11766 tim=952702226
WAIT #14: nam='db file scattered read' ela= 36198 file#=4 block#=326 blocks=63 obj#=11766 tim=952740355
WAIT #14: nam='db file scattered read' ela= 35145 file#=4 block#=390 blocks=63 obj#=11766 tim=952777475
WAIT #14: nam='db file scattered read' ela= 37517 file#=4 block#=454 blocks=63 obj#=11766 tim=952816880
WAIT #14: nam='db file scattered read' ela= 42379 file#=4 block#=518 blocks=63 obj#=11766 tim=952861118
...
WAIT #14: nam='db file scattered read' ela= 10201 file#=4 block#=321673 blocks=64 obj#=11766 tim=1020166933
WAIT #14: nam='db file scattered read' ela= 13637 file#=4 block#=321737 blocks=64 obj#=11766 tim=1020182441
WAIT #14: nam='db file scattered read' ela= 10208 file#=4 block#=321801 blocks=64 obj#=11766 tim=1020194513
WAIT #14: nam='db file scattered read' ela= 10237 file#=4 block#=321865 blocks=64 obj#=11766 tim=1020206619
WAIT #14: nam='db file scattered read' ela= 11186 file#=4 block#=321929 blocks=64 obj#=11766 tim=1020219672
WAIT #14: nam='db file scattered read' ela= 10174 file#=4 block#=321993 blocks=64 obj#=11766 tim=1020231775
WAIT #14: nam='db file scattered read' ela= 10169 file#=4 block#=322057 blocks=64 obj#=11766 tim=1020243848
WAIT #14: nam='db file scattered read' ela= 1357 file#=4 block#=322121 blocks=12 obj#=11766 tim=1020246986
FETCH #14:c=9765625,e=67692846,p=320423,cr=321597,cu=0,mis=0,r=0,dep=0,og=1,tim=1020247320
WAIT #14: nam='SQL*Net message from client' ela= 27653 driver id=1413697536 #bytes=1 p3=0 obj#=11766 tim=1020275060
STAT #14 id=1 cnt=0 pid=0 pos=1 obj=11766 op='TABLE ACCESS FULL T1 (cr=321597 pr=320423 pw=0 time=67692842 us)'
********************************************************************************
```

Test 7 8KB:

```
********************************************************************************
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       16      0.00       0.07          5         10          0           0
Execute     17      0.01       0.10         17        142          8           8
Fetch      642     45.93     172.36     929937     940085          2        9498
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      675     45.95     172.55     929959     940237         10        9506

Misses in library cache during parse: 9
Misses in library cache during execute: 3

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     668        0.00          0.00
  SQL*Net message from client                   668        0.01          2.81
  db file sequential read                     226183        0.02         63.63
  db file scattered read                       12186        0.05         65.93
  db file parallel read                            1        0.28          0.28
********************************************************************************
```

Test 10 16KB:

```
********************************************************************************
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       16      0.03       0.06          5         10          0           0
Execute     17      0.00       0.10         17        136          8           8
Fetch      642     39.46     146.40     458876     463952          2        9498
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      675     39.50     146.56     458898     464098         10        9506

Misses in library cache during parse: 9
Misses in library cache during execute: 3

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     668        0.00          0.00
  SQL*Net message from client                   668        0.02          2.80
  db file sequential read                    113888        0.06         44.43
  db file scattered read                      12267        0.05         64.88
  db file parallel read                           1        0.26          0.26
********************************************************************************
```

Test 7 8KB:

```
********************************************************************************
SELECT
  COUNT(*)
FROM
  T2

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          2          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.26       1.69       6966      13942          2           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.26       1.71       6968      13944          2           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=13942 pr=6966 pw=0 time=1690194 us)
1000000   INDEX FAST FULL SCAN T2_IND1 (cr=13942 pr=6966 pw=0 time=334249 us)(object id 11759)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         2        0.01          0.02
  db file parallel read                           1        0.28          0.28
  db file scattered read                        124        0.02          1.09
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************
```

Test 10 16KB:

```
********************************************************************************
SELECT
  COUNT(*)
FROM
  T2

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          2          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.23       1.70       3325       6652          2           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.23       1.73       3327       6654          2           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=6652 pr=3325 pw=0 time=1705485 us)
1000000   INDEX FAST FULL SCAN T2_IND1 (cr=6652 pr=3325 pw=0 time=3326572 us)(object id 11769)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         4        0.02          0.05
  db file parallel read                           1        0.26          0.26
  db file scattered read                         67        0.04          1.13
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************
```

Test 8 8KB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
```

```
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.06       0.16          0          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       95     83.29     157.70     274019     274113          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       97     83.35     157.87     274019     274115          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=274113 pr=274019 pw=0 time=157670269 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274113 pr=274019 pw=0 time=100044637 us)(object id 11757)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                      95        0.00          0.00
  db file sequential read                    274019        0.03         77.50
  SQL*Net more data to client                    85        0.00          0.00
  SQL*Net message from client                    95        0.70          0.75
********************************************************************************
```

Test 11 16KB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.06       0.15          0          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       95     75.93     130.40     135072     135166          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       97     76.00     130.55     135072     135168          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=135166 pr=135072 pw=0 time=130371766 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=135166 pr=135072 pw=0 time=100040110 us)(object id 11767)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                      95        0.00          0.00
  db file sequential read                    135072        0.03         54.73
  SQL*Net more data to client                    84        0.00          0.00
  SQL*Net message from client                    95        0.69          0.73
********************************************************************************
```

Test 9 8KB:

```
********************************************************************************
SELECT DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     11.75      71.40     651470     651498          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     11.75      71.42     651470     651498          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  HASH UNIQUE (cr=651498 pr=651470 pw=0 time=71409334 us)
      0   TABLE ACCESS FULL T1 (cr=651498 pr=651470 pw=0 time=71409264 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  db file sequential read                         1        0.01          0.01
  db file scattered read                       5114        0.05         59.96
  SQL*Net message from client                     1        0.01          0.01
********************************************************************************
```

Test 12 16KB:

```
********************************************************************************
SELECT DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     10.10      71.07     321569     321597          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     10.10      71.09     321569     321597          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  HASH UNIQUE (cr=321597 pr=321569 pw=0 time=71077823 us)
      0   TABLE ACCESS FULL T1 (cr=321597 pr=321569 pw=0 time=71077749 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  db file sequential read                         1        0.02          0.02
  db file scattered read                       5048        0.05         61.69
  SQL*Net message from client                     1        0.03          0.03
********************************************************************************
```

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 7, 2008 8:51 PM in response to: Charles Hooper

Reply

Charles,

Excellent test. I wish i could have done similar tests. What i understood is, higher the data volume request, Oracle always favor in higher data block. I will leave for Experts to comment on your valuable test results.

Regards,
sp009

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 8, 2008 8:15 AM in response to: sp009

Reply

>
> Excellent test. I wish i could have done similar
> tests. What i understood is, higher the data volume
> request, Oracle always favor in higher data block.
>

sp009,

I haven't had time to compare all the different results yet, but I can't help noticing that when Greg Rahn showed the the block size made virtually no difference (bar a small benefit in favour of smaller blocks) you didnt' leap to a sweeping conclusion that smaller was always better. Nor did you leap to such a conclusion when Charles Hooper's first set of results suggested that a larger block size was actually a liability.

But now that there are some results that agree with your favoured point of view you rush to affirm that a larger block size is always better.

Wrong approach - you're supposed to design a theory to match the facts, not select the facts to match the theory.

In comparison, when Richard saw that Charles Hooper's results suggested that the 'big is better' hypothesis was completely wrong he didn't claim that the results supported his argument, he used his knowledge of how Oracle works to suggest that there was a flaw in the test methodology that needed to be addressed.

When Greg Rahn produced a set of results that supported the theory that the block size makes virtually no difference - and actually got better results from the smaller block size - I didn't claim this as proof of a point that I've often made, I pointed out (using my knowledge of how Oracle works) a feature of 11g that had an impact on the test that could introduce a bias to the results that wouldn't necessarily appear in general - although it might be a benefit in data warehouses.

As far as Charles' latest results are concerned, there are three anomalies that I would be interested in:

a) From the query summary tables in the tkprof output, disk = query, which means the data is in an unusually clean state, and effects of read-consistency have been factored out. Is this a reasonable test from which you could safely draw your general conclusion.

b) Glancing at a couple of the tkprof outputs, the error in internal accounting (CPU + recorded waits != elapsed) is often as large as the difference in timing between the tests of different block sizes.

c) The index full scan test shows a dramatic benefit for the 16K block. You can associate some of the benefit to having halved the number of latch acquisitions needed (remember the point I made about Greg's example); but the benefit is so unexpectedly large that I would want to examine two other features of this example before I used it as a basis for a decision to rebuild a database (or even tablespace) with a larger block size.
i) Why wasn't the optimizer using the "db file parallel read" in a case where it was so obviously appropriate - it's possible that the nature of the test, particularly the database restart, has stopped Oracle from using a mechanical optimisation that would normally be available.
ii) Why is there such a difference in the total time spend on single block reads when the index is in a perfect shape to benefit from O/S readahead/prefetching - the index leaf blocks should be ordered "on disc" nearly perfectly, so Oracle's choice of block size shouldn't (or wouldn't, on other platforms) have affected the number of real disk seeks that took place. In fact, the average read times are so fast that something of that sort seems to have happened - but my next step would be to analyze the trace files in detail to check for any anomaly.

It's tedious and boring - but if you want to treat unexpectedly bad results in the same way as unexpectedly good results ... assume there's something important you've overlooked.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Re: Larger vs. Small data block
Posted: Jun 8, 2008 8:48 AM in response to: Greg Rahn

Reply

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 8, 2008 8:58 AM in response to: sp009

Reply

> Charles,
>
> Excellent test. I wish i could have done similar
> tests. What i understood is, higher the data volume
> request, Oracle always favor in higher data block. I
> will leave for Experts to comment on your valuable
> test results.
>
> Regards,
> sp009

sp009,

Thanks for the remark.

However, even though I attempted to control the experiment as best as possible, there were a couple configuration issues, limitations, and testing depth problems in the experiment. The set of tests at this point is not as conclusive as I would like to see. For example, compare the elapsed times for the first set of tests with the 8KB block size database with the last set of tests with the 16KB block size database - notice how the elapsed times are similar. Or, compare the elapsed times for the first set of tests with the 16KB block size with the last set of tests with the 16KB block size database - notice how they are different, even though I attempted to create them in the same area of the disks.

The problems that I have with the tests that I conducted are as follows (this is a short list):
* The initial size of the data file for the USER_DATA tablespace was not initially created large enough (2GB, should have been 8GB), and had to expand 100MB at a time. This might mimic actual real-world database behavior, or it might have artificially affected performance measurements.
* Redo writing speed due to disk contention may have been a problem.
* I was not creative enough when creating the SQL statements to produce different execution plans - the test did include full table scans, full index scans, and fast full index scans. All plans involved a single table. No index unique scans or range scans were included.
* The tests did not examine update performance for indexes or tables, other than the brief test of inserting 1,000,000 rows into T2 that had an existing index (if you want a real test, insert all rows from T1 into T2).
* System statistics were not gathered, nor were statistics on the SYS schema, nor were statistics gathered on fixed object statistics. Statistics were not gathered on the table (just one) and its index until the end of the experiment. This may have lead to excessive problems with dynamic sampling, or inappropriate execution plans.
* The second run of the statistics gathering shows that the statistics were gathered faster the during the second statistics gathering execution in the 8KB block size database, but that was not the case in the 16KB block size database. This may imply that had I not flushed the buffer cache several times through the test, or restarted the computer, the elapsed time for the 8KB test runs may have been more favorable.
* The trace files showed a great variation in the ela= values for db file scattered read, allowing one of the 1MB reads to complete twice as fast as one of the 64KB reads: (8KB block size)

```
 block#=17 blocks=8 ela=910
 block#=26 blocks=7 ela=18546
 block#=33 blocks=8 ela=935
 block#=42 blocks=7 ela=554
 ...
 block#=652177 blocks=128 ela=9012
```

The experiment did not show quite what I expected, but it did show a couple interesting things. As far as I can tell, the test is currently inconclusive, but that may just mean that it either needs to be repeated on several different Oracle servers, or that the flaws in the test need to be fixed and then tested several times.

The DBMS_XPLAN with actual run statistics seems to show that an INDEX FULL SCAN (single block reads) requires roughly the same amount of time regardless of block size:

```
| Id  | Operation          | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
#TEST RUN 7 8KB:
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     50M|     50M|00:01:40.04 |     274K|    274K|

#TEST RUN 11 16KB:
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     54M|     50M|00:01:40.04 |     135K|    135K|
```

However, the DISTINCT requirement seemed to tip the balance toward the larger block size:

```
#TEST RUN 7 8KB:
|   1 |  SORT UNIQUE NOSORT|         |      1 |     50M|   9454 |00:02:37.67 |     274K|    274K|

#TEST RUN 11 16KB:
|   1 |  SORT UNIQUE NOSORT|         |      1 |     54M|   9454 |00:02:10.37 |     135K|    135K|
```

I am interested in seeing the analysis and results posted by others in the group. This has been a very interesting thread.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 8, 2008 10:50 AM in response to:

Reply

>
> No, sorry. The goal of this thread (correct me if
> I'm wrong) is to challenge the conventional wisdom
> about the general observations of performance
> differences with different blocksizes, and the only
> way to validate the empirical observations
> scientifically is with a stochastic study, finding

> the correlations, variable and beta.
>

You're wrong.

The goal of this thread has moved around a bit but, to a large degree, has involved intelligent adults developing tests and sharing observations. The goal has not been to *"challenge the conventional wisdom about the general observations of performance differences with different blocksizes"*, because the conventional wisdom is that changes in block size may occasionally help, may occasionally cause problems, and typically are an irrelevant waste of effort.

>
> >> I don't think anyone is defining Oracle software as a science.
>
> You and Jonathan do, that was my point:
>
> http://www.oaktable.net/
> **"The OakTable Network: A network for the Oracle scientist"**
>
> You are listed as a member of this "Network for the Oracle scientist", right?
>
> Please enlighten us. . . .
>

If the term "Oracle Scentist" is such a problem, there's a definition near the end of the following article:
http://www.jlcomp.demon.co.uk/scientist.html

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 8, 2008 2:53 PM in response to: Charles Hooper

Reply

In the interest in determining what is happening when the 8KB block size database during test 7 reported that the elapsed time was 00:01:42.18, while the 16KB block size database during test 10 reported that the elapsed time was 00:01:19.85, I will take a closer look at the 10046 trace file captured at level 8. We start to see the significance of directly examining the 10046 trace file. For this section of the TKPROF output:

Test 7 8KB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     33.71      99.25     274239     274645          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     33.71      99.27     274240     274647          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=274645 pr=274239 pw=0 time=100635543 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274645 pr=274239 pw=0 time=100036443 us)(object id 11757)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     632        0.00          0.00
  db file scattered read                       6922        0.02          5.10
  db file sequential read                    226153        0.02         63.43
  SQL*Net message from client                   632        0.01          2.77
********************************************************************************
```

Test 10 16KB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.00          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     29.46      76.99     135128     135703          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     29.48      77.00     135129     135705          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=135703 pr=135128 pw=0 time=76572511 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=135703 pr=135128 pw=0 time=50022973 us)(object id 11767)
```

We see in the trace file an odd pattern that might be caused by the ASSM segment management for the tablespace:
EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO

The 8KB database shows a repeating pattern of reading 7 blocks, followed by the next block being read all by itself. The 16KB database shows a repeating pattern of reading 3 blocks, followed by the next block being read by by itself. For every 15 rows fetched, the 8KB block read pattern repeats approximately one extra cycle. Unlike the portion of the trace file for the full table scan where Oracle switched to a 1024KB read, Oracle never switches to using more than a 56KB read during the index full scan.

The beginning portion of this trace file follows, with spaces added in the 16KB database's trace file when an extra read was required in the 8KB database's trace file.

8KB

```
PARSING IN CURSOR #6 len=83 dep=0 uid=30 oct=3 lid=30 tim=1020768742 hv=3216823004 ad='50dca234'
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
END OF STMT
PARSE #6:c=15625,e=69901,p=130,cr=95,cu=0,mis=1,r=0,dep=0,og=1,tim=1020768738
EXEC #6:c=0,e=35,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1020768926
WAIT #6: nam='SQL*Net message to client' ela= 3 driver id=1413697536 #bytes=1 p3=0 obj#=11757 tim=1020768963
WAIT #6: nam='db file scattered read' ela= 15359 file#=4 block#=655372 blocks=5 obj#=11757 tim=1020784405
WAIT #6: nam='db file sequential read' ela= 8850 file#=4 block#=698610 blocks=1 obj#=11757 tim=1020793304
WAIT #6: nam='db file sequential read' ela= 12070 file#=4 block#=655598 blocks=1 obj#=11757 tim=1020805410
WAIT #6: nam='db file sequential read' ela= 215 file#=4 block#=655377 blocks=1 obj#=11757 tim=1020806016
WAIT #6: nam='db file scattered read' ela= 4686 file#=4 block#=655378 blocks=7 obj#=11757 tim=1020810829
WAIT #6: nam='db file sequential read' ela= 237 file#=4 block#=655386 blocks=1 obj#=11757 tim=1020811669
WAIT #6: nam='db file scattered read' ela= 427 file#=4 block#=655387 blocks=6 obj#=11757 tim=1020812223
WAIT #6: nam='db file sequential read' ela= 233 file#=4 block#=655393 blocks=1 obj#=11757 tim=1020812993
WAIT #6: nam='db file scattered read' ela= 586 file#=4 block#=655394 blocks=7 obj#=11757 tim=1020813710
WAIT #6: nam='db file sequential read' ela= 238 file#=4 block#=655402 blocks=1 obj#=11757 tim=1020814548
WAIT #6: nam='db file scattered read' ela= 397 file#=4 block#=655403 blocks=6 obj#=11757 tim=1020815073
FETCH #6:c=0,e=46165,p=37,cr=32,cu=0,mis=0,r=1,dep=0,og=1,tim=1020815171
WAIT #6: nam='SQL*Net message from client' ela= 12044 driver id=1413697536 #bytes=1 p3=0 obj#=11757 tim=1020827278
WAIT #6: nam='db file sequential read' ela= 264 file#=4 block#=655409 blocks=1 obj#=11757 tim=1020828060
WAIT #6: nam='db file scattered read' ela= 721 file#=4 block#=655410 blocks=7 obj#=11757 tim=1020828913
WAIT #6: nam='db file sequential read' ela= 260 file#=4 block#=655418 blocks=1 obj#=11757 tim=1020829761
WAIT #6: nam='db file scattered read' ela= 437 file#=4 block#=655419 blocks=6 obj#=11757 tim=1020830323
WAIT #6: nam='db file sequential read' ela= 227 file#=4 block#=655425 blocks=1 obj#=11757 tim=1020831055
WAIT #6: nam='db file scattered read' ela= 592 file#=4 block#=655426 blocks=7 obj#=11757 tim=1020831771
WAIT #6: nam='db file sequential read' ela= 215 file#=4 block#=655434 blocks=1 obj#=11757 tim=1020832573
WAIT #6: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11757 tim=1020832670
WAIT #6: nam='db file scattered read' ela= 16648 file#=4 block#=655435 blocks=6 obj#=11757 tim=1020849377
WAIT #6: nam='db file sequential read' ela= 204 file#=4 block#=655441 blocks=1 obj#=11757 tim=1020850073
WAIT #6: nam='db file scattered read' ela= 704 file#=4 block#=655442 blocks=7 obj#=11757 tim=1020850900
WAIT #6: nam='db file sequential read' ela= 205 file#=4 block#=655450 blocks=1 obj#=11757 tim=1020851666
WAIT #6: nam='db file scattered read' ela= 430 file#=4 block#=655451 blocks=6 obj#=11757 tim=1020852220
WAIT #6: nam='db file sequential read' ela= 224 file#=4 block#=655457 blocks=1 obj#=11757 tim=1020852951
WAIT #6: nam='db file scattered read' ela= 600 file#=4 block#=655458 blocks=7 obj#=11757 tim=1020853672
WAIT #6: nam='db file sequential read' ela= 248 file#=4 block#=655466 blocks=1 obj#=11757 tim=1020854506
WAIT #6: nam='db file scattered read' ela= 438 file#=4 block#=655467 blocks=6 obj#=11757 tim=1020855070
WAIT #6: nam='db file sequential read' ela= 206 file#=4 block#=655473 blocks=1 obj#=11757 tim=1020855764
WAIT #6: nam='db file scattered read' ela= 596 file#=4 block#=655474 blocks=7 obj#=11757 tim=1020856479
WAIT #6: nam='db file sequential read' ela= 365 file#=4 block#=655482 blocks=1 obj#=11757 tim=1020857403
WAIT #6: nam='db file scattered read' ela= 465 file#=4 block#=655483 blocks=6 obj#=11757 tim=1020857988
WAIT #6: nam='db file sequential read' ela= 244 file#=4 block#=655489 blocks=1 obj#=11757 tim=1020858717
WAIT #6: nam='db file scattered read' ela= 633 file#=4 block#=655490 blocks=7 obj#=11757 tim=1020859472
WAIT #6: nam='db file sequential read' ela= 240 file#=4 block#=655499 blocks=1 obj#=11757 tim=1020860273
WAIT #6: nam='db file scattered read' ela= 439 file#=4 block#=655500 blocks=5 obj#=11757 tim=1020860835
WAIT #6: nam='db file sequential read' ela= 206 file#=4 block#=655505 blocks=1 obj#=11757 tim=1020861495
WAIT #6: nam='db file scattered read' ela= 591 file#=4 block#=655506 blocks=7 obj#=11757 tim=1020862210
WAIT #6: nam='db file sequential read' ela= 210 file#=4 block#=655513 blocks=1 obj#=11757 tim=1020863010
WAIT #6: nam='db file scattered read' ela= 472 file#=4 block#=655514 blocks=7 obj#=11757 tim=1020863608
WAIT #6: nam='db file sequential read' ela= 270 file#=4 block#=655521 blocks=1 obj#=11757 tim=1020864485
WAIT #6: nam='db file scattered read' ela= 634 file#=4 block#=655522 blocks=7 obj#=11757 tim=1020865245
WAIT #6: nam='db file sequential read' ela= 265 file#=4 block#=655529 blocks=1 obj#=11757 tim=1020866117
WAIT #6: nam='db file scattered read' ela= 440 file#=4 block#=655530 blocks=7 obj#=11757 tim=1020866685
WAIT #6: nam='db file sequential read' ela= 218 file#=4 block#=655537 blocks=1 obj#=11757 tim=1020867524
WAIT #6: nam='db file scattered read' ela= 588 file#=4 block#=655538 blocks=7 obj#=11757 tim=1020868242
WAIT #6: nam='db file sequential read' ela= 248 file#=4 block#=655545 blocks=1 obj#=11757 tim=1020869198
WAIT #6: nam='db file scattered read' ela= 9739 file#=4 block#=655546 blocks=7 obj#=11757 tim=1020879079
WAIT #6: nam='db file sequential read' ela= 213 file#=4 block#=655553 blocks=1 obj#=11757 tim=1020879919
WAIT #6: nam='db file scattered read' ela= 700 file#=4 block#=655554 blocks=7 obj#=11757 tim=1020880749
WAIT #6: nam='db file sequential read' ela= 217 file#=4 block#=655561 blocks=1 obj#=11757 tim=1020881588
WAIT #6: nam='db file scattered read' ela= 451 file#=4 block#=655562 blocks=7 obj#=11757 tim=1020882168
WAIT #6: nam='db file sequential read' ela= 220 file#=4 block#=655569 blocks=1 obj#=11757 tim=1020883006
WAIT #6: nam='db file scattered read' ela= 592 file#=4 block#=655570 blocks=7 obj#=11757 tim=1020883727
WAIT #6: nam='db file sequential read' ela= 3700 file#=4 block#=655577 blocks=1 obj#=11757 tim=1020888050
WAIT #6: nam='db file scattered read' ela= 463 file#=4 block#=655578 blocks=7 obj#=11757 tim=1020888642
WAIT #6: nam='db file sequential read' ela= 230 file#=4 block#=655585 blocks=1 obj#=11757 tim=1020889518
WAIT #6: nam='db file scattered read' ela= 591 file#=4 block#=655586 blocks=7 obj#=11757 tim=1020890234
WAIT #6: nam='db file scattered read' ela= 411 file#=4 block#=655593 blocks=5 obj#=11757 tim=1020891210
WAIT #6: nam='db file scattered read' ela= 269 file#=4 block#=655599 blocks=2 obj#=11757 tim=1020891894
WAIT #6: nam='db file sequential read' ela= 190 file#=4 block#=655601 blocks=1 obj#=11757 tim=1020892266
WAIT #6: nam='db file scattered read' ela= 601 file#=4 block#=655602 blocks=7 obj#=11757 tim=1020892987
WAIT #6: nam='db file sequential read' ela= 234 file#=4 block#=655609 blocks=1 obj#=11757 tim=1020893787
WAIT #6: nam='db file scattered read' ela= 468 file#=4 block#=655610 blocks=7 obj#=11757 tim=1020894377
WAIT #6: nam='db file sequential read' ela= 220 file#=4 block#=655617 blocks=1 obj#=11757 tim=1020895171
WAIT #6: nam='db file scattered read' ela= 556 file#=4 block#=655618 blocks=7 obj#=11757 tim=1020895848
WAIT #6: nam='db file sequential read' ela= 219 file#=4 block#=655627 blocks=1 obj#=11757 tim=1020896650
WAIT #6: nam='db file scattered read' ela= 448 file#=4 block#=655628 blocks=5 obj#=11757 tim=1020897224
WAIT #6: nam='db file sequential read' ela= 237 file#=4 block#=655633 blocks=1 obj#=11757 tim=1020897884
WAIT #6: nam='db file scattered read' ela= 589 file#=4 block#=655634 blocks=7 obj#=11757 tim=1020898599
WAIT #6: nam='db file sequential read' ela= 215 file#=4 block#=655641 blocks=1 obj#=11757 tim=1020899402
WAIT #6: nam='db file scattered read' ela= 425 file#=4 block#=655642 blocks=7 obj#=11757 tim=1020899950
WAIT #6: nam='db file sequential read' ela= 195 file#=4 block#=655649 blocks=1 obj#=11757 tim=1020900789
WAIT #6: nam='db file scattered read' ela= 565 file#=4 block#=655650 blocks=7 obj#=11757 tim=1020901503
WAIT #6: nam='db file sequential read' ela= 182 file#=4 block#=655657 blocks=1 obj#=11757 tim=1020902305
WAIT #6: nam='db file scattered read' ela= 445 file#=4 block#=655658 blocks=7 obj#=11757 tim=1020902889
WAIT #6: nam='db file sequential read' ela= 213 file#=4 block#=655665 blocks=1 obj#=11757 tim=1020903727
WAIT #6: nam='db file scattered read' ela= 587 file#=4 block#=655666 blocks=7 obj#=11757 tim=1020904445
WAIT #6: nam='db file sequential read' ela= 222 file#=4 block#=655673 blocks=1 obj#=11757 tim=1020905284
WAIT #6: nam='db file scattered read' ela= 528 file#=4 block#=655674 blocks=7 obj#=11757 tim=1020905942
WAIT #6: nam='db file sequential read' ela= 256 file#=4 block#=655681 blocks=1 obj#=11757 tim=1020906780
WAIT #6: nam='db file scattered read' ela= 631 file#=4 block#=655682 blocks=7 obj#=11757 tim=1020907538
WAIT #6: nam='db file sequential read' ela= 251 file#=4 block#=655689 blocks=1 obj#=11757 tim=1020908374
WAIT #6: nam='db file scattered read' ela= 499 file#=4 block#=655690 blocks=7 obj#=11757 tim=1020908996
WAIT #6: nam='db file sequential read' ela= 219 file#=4 block#=655697 blocks=1 obj#=11757 tim=1020909797
WAIT #6: nam='db file scattered read' ela= 596 file#=4 block#=655698 blocks=7 obj#=11757 tim=1020910516
WAIT #6: nam='db file sequential read' ela= 214 file#=4 block#=655705 blocks=1 obj#=11757 tim=1020911317
WAIT #6: nam='db file scattered read' ela= 498 file#=4 block#=655706 blocks=7 obj#=11757 tim=1020911939
WAIT #6: nam='db file sequential read' ela= 271 file#=4 block#=655713 blocks=1 obj#=11757 tim=1020912777
WAIT #6: nam='db file scattered read' ela= 621 file#=4 block#=655714 blocks=7 obj#=11757 tim=1020913535
WAIT #6: nam='db file sequential read' ela= 259 file#=4 block#=655721 blocks=1 obj#=11757 tim=1020914371
WAIT #6: nam='db file scattered read' ela= 495 file#=4 block#=655722 blocks=7 obj#=11757 tim=1020914991
WAIT #6: nam='db file sequential read' ela= 241 file#=4 block#=655729 blocks=1 obj#=11757 tim=1020915792
WAIT #6: nam='db file scattered read' ela= 637 file#=4 block#=655730 blocks=7 obj#=11757 tim=1020916554
WAIT #6: nam='db file sequential read' ela= 246 file#=4 block#=655737 blocks=1 obj#=11757 tim=1020917385
WAIT #6: nam='db file scattered read' ela= 5246 file#=4 block#=655738 blocks=7 obj#=11757 tim=1020922756
WAIT #6: nam='db file sequential read' ela= 219 file#=4 block#=655745 blocks=1 obj#=11757 tim=1020923562
WAIT #6: nam='db file scattered read' ela= 698 file#=4 block#=655746 blocks=7 obj#=11757 tim=1020924388
WAIT #6: nam='db file sequential read' ela= 240 file#=4 block#=655755 blocks=1 obj#=11757 tim=1020925225
WAIT #6: nam='db file scattered read' ela= 483 file#=4 block#=655756 blocks=5 obj#=11757 tim=1020925833
WAIT #6: nam='db file sequential read' ela= 248 file#=4 block#=655761 blocks=1 obj#=11757 tim=1020926531
WAIT #6: nam='db file scattered read' ela= 633 file#=4 block#=655762 blocks=7 obj#=11757 tim=1020927290
WAIT #6: nam='db file sequential read' ela= 243 file#=4 block#=655769 blocks=1 obj#=11757 tim=1020928125
WAIT #6: nam='db file scattered read' ela= 3937 file#=4 block#=655770 blocks=7 obj#=11757 tim=1020932186
WAIT #6: nam='db file sequential read' ela= 235 file#=4 block#=655777 blocks=1 obj#=11757 tim=1020933005
WAIT #6: nam='db file scattered read' ela= 613 file#=4 block#=655778 blocks=7 obj#=11757 tim=1020933760
WAIT #6: nam='db file sequential read' ela= 206 file#=4 block#=655785 blocks=1 obj#=11757 tim=1020934561
WAIT #6: nam='db file scattered read' ela= 684 file#=4 block#=655786 blocks=7 obj#=11757 tim=1020935370
WAIT #6: nam='db file sequential read' ela= 206 file#=4 block#=655793 blocks=1 obj#=11757 tim=1020936173
WAIT #6: nam='db file scattered read' ela= 582 file#=4 block#=655794 blocks=7 obj#=11757 tim=1020936889
WAIT #6: nam='db file sequential read' ela= 233 file#=4 block#=655801 blocks=1 obj#=11757 tim=1020937766
WAIT #6: nam='db file scattered read' ela= 533 file#=4 block#=655802 blocks=7 obj#=11757 tim=1020938429
WAIT #6: nam='db file sequential read' ela= 187 file#=4 block#=655809 blocks=1 obj#=11757 tim=1020939265
WAIT #6: nam='db file scattered read' ela= 585 file#=4 block#=655810 blocks=7 obj#=11757 tim=1020939984
WAIT #6: nam='db file sequential read' ela= 227 file#=4 block#=655817 blocks=1 obj#=11757 tim=1020940860
WAIT #6: nam='db file scattered read' ela= 446 file#=4 block#=655818 blocks=7 obj#=11757 tim=1020941442
WAIT #6: nam='db file sequential read' ela= 229 file#=4 block#=655825 blocks=1 obj#=11757 tim=1020942205
WAIT #6: nam='db file scattered read' ela= 591 file#=4 block#=655826 blocks=7 obj#=11757 tim=1020942921
WAIT #6: nam='db file sequential read' ela= 214 file#=4 block#=655833 blocks=1 obj#=11757 tim=1020943723
WAIT #6: nam='db file scattered read' ela= 428 file#=4 block#=655834 blocks=7 obj#=11757 tim=1020944272
WAIT #6: nam='db file sequential read' ela= 209 file#=4 block#=655841 blocks=1 obj#=11757 tim=1020945070
WAIT #6: nam='db file scattered read' ela= 577 file#=4 block#=655842 blocks=7 obj#=11757 tim=1020945788
FETCH #6:c=31250,e=118933,p=428,cr=432,cu=0,mis=0,r=15,dep=0,og=1,tim=1020946262
WAIT #6: nam='SQL*Net message from client' ela= 3974 driver id=1413697536 #bytes=1 p3=0 obj#=11757 tim=1020950291
WAIT #6: nam='db file sequential read' ela= 246 file#=4 block#=655849 blocks=1 obj#=11757 tim=1020950768
WAIT #6: nam='db file scattered read' ela= 466 file#=4 block#=655850 blocks=7 obj#=11757 tim=1020951382
WAIT #6: nam='db file sequential read' ela= 220 file#=4 block#=655857 blocks=1 obj#=11757 tim=1020952297
WAIT #6: nam='db file scattered read' ela= 611 file#=4 block#=655858 blocks=7 obj#=11757 tim=1020953051
WAIT #6: nam='db file sequential read' ela= 220 file#=4 block#=655865 blocks=1 obj#=11757 tim=1020953966
WAIT #6: nam='db file scattered read' ela= 490 file#=4 block#=655866 blocks=7 obj#=11757 tim=1020954596
WAIT #6: nam='db file sequential read' ela= 247 file#=4 block#=655873 blocks=1 obj#=11757 tim=1020955543
WAIT #6: nam='db file scattered read' ela= 646 file#=4 block#=655874 blocks=7 obj#=11757 tim=1020956331
WAIT #6: nam='db file sequential read' ela= 263 file#=4 block#=655883 blocks=1 obj#=11757 tim=1020957288
WAIT #6: nam='db file scattered read' ela= 435 file#=4 block#=655884 blocks=5 obj#=11757 tim=1020957859
WAIT #6: nam='db file sequential read' ela= 234 file#=4 block#=655889 blocks=1 obj#=11757 tim=1020958594
WAIT #6: nam='db file scattered read' ela= 623 file#=4 block#=655890 blocks=7 obj#=11757 tim=1020959354
WAIT #6: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11757 tim=1020959927
WAIT #6: nam='db file sequential read' ela= 234 file#=4 block#=655897 blocks=1 obj#=11757 tim=1020960302
WAIT #6: nam='db file scattered read' ela= 466 file#=4 block#=655898 blocks=7 obj#=11757 tim=1020960892
WAIT #6: nam='db file sequential read' ela= 265 file#=4 block#=655905 blocks=1 obj#=11757 tim=1020961728
WAIT #6: nam='db file scattered read' ela= 618 file#=4 block#=655906 blocks=7 obj#=11757 tim=1020962484
WAIT #6: nam='db file sequential read' ela= 258 file#=4 block#=655913 blocks=1 obj#=11757 tim=1020963361
WAIT #6: nam='db file scattered read' ela= 489 file#=4 block#=655914 blocks=7 obj#=11757 tim=1020963981
WAIT #6: nam='db file sequential read' ela= 268 file#=4 block#=655921 blocks=1 obj#=11757 tim=1020964817
WAIT #6: nam='db file scattered read' ela= 639 file#=4 block#=655922 blocks=7 obj#=11757 tim=1020965576
WAIT #6: nam='db file sequential read' ela= 260 file#=4 block#=655929 blocks=1 obj#=11757 tim=1020966411
WAIT #6: nam='db file scattered read' ela= 8249 file#=4 block#=655930 blocks=7 obj#=11757 tim=1020974776
WAIT #6: nam='db file sequential read' ela= 212 file#=4 block#=655937 blocks=1 obj#=11757 tim=1020975543
WAIT #6: nam='db file scattered read' ela= 712 file#=4 block#=655938 blocks=7 obj#=11757 tim=1020976374
WAIT #6: nam='db file sequential read' ela= 218 file#=4 block#=655945 blocks=1 obj#=11757 tim=1020977138
WAIT #6: nam='db file scattered read' ela= 496 file#=4 block#=655946 blocks=7 obj#=11757 tim=1020977753
WAIT #6: nam='db file sequential read' ela= 252 file#=4 block#=655953 blocks=1 obj#=11757 tim=1020978594
WAIT #6: nam='db file scattered read' ela= 635 file#=4 block#=655954 blocks=7 obj#=11757 tim=1020979348
WAIT #6: nam='db file sequential read' ela= 253 file#=4 block#=655961 blocks=1 obj#=11757 tim=1020980189
WAIT #6: nam='db file scattered read' ela= 461 file#=4 block#=655962 blocks=7 obj#=11757 tim=1020980778
WAIT #6: nam='db file sequential read' ela= 204 file#=4 block#=655969 blocks=1 obj#=11757 tim=1020981576
WAIT #6: nam='db file scattered read' ela= 578 file#=4 block#=655970 blocks=7 obj#=11757 tim=1020982295
WAIT #6: nam='db file sequential read' ela= 209 file#=4 block#=655977 blocks=1 obj#=11757 tim=1020983094
WAIT #6: nam='db file scattered read' ela= 1750 file#=4 block#=655978 blocks=7 obj#=11757 tim=1020984967
WAIT #6: nam='db file sequential read' ela= 216 file#=4 block#=655985 blocks=1 obj#=11757 tim=1020985765
WAIT #6: nam='db file scattered read' ela= 709 file#=4 block#=655986 blocks=7 obj#=11757 tim=1020986599
WAIT #6: nam='db file sequential read' ela= 319 file#=4 block#=655993 blocks=1 obj#=11757 tim=1020987511
WAIT #6: nam='db file scattered read' ela= 463 file#=4 block#=655994 blocks=7 obj#=11757 tim=1020988100
WAIT #6: nam='db file sequential read' ela= 223 file#=4 block#=656001 blocks=1 obj#=11757 tim=1020988906
WAIT #6: nam='db file scattered read' ela= 589 file#=4 block#=656002 blocks=7 obj#=11757 tim=1020989620
WAIT #6: nam='db file sequential read' ela= 469 file#=4 block#=656011 blocks=1 obj#=11757 tim=1020990676
WAIT #6: nam='db file scattered read' ela= 415 file#=4 block#=656012 blocks=5 obj#=11757 tim=1020991216
WAIT #6: nam='db file sequential read' ela= 200 file#=4 block#=656017 blocks=1 obj#=11757 tim=1020991837
WAIT #6: nam='db file scattered read' ela= 588 file#=4 block#=656018 blocks=7 obj#=11757 tim=1020992554
WAIT #6: nam='db file sequential read' ela= 231 file#=4 block#=656025 blocks=1 obj#=11757 tim=1020993354
WAIT #6: nam='db file scattered read' ela= 434 file#=4 block#=656026 blocks=7 obj#=11757 tim=1020993908
WAIT #6: nam='db file sequential read' ela= 191 file#=4 block#=656033 blocks=1 obj#=11757 tim=1020994667
WAIT #6: nam='db file scattered read' ela= 582 file#=4 block#=656034 blocks=7 obj#=11757 tim=1020995386
WAIT #6: nam='db file sequential read' ela= 186 file#=4 block#=656041 blocks=1 obj#=11757 tim=1020996074
WAIT #6: nam='db file scattered read' ela= 609 file#=4 block#=656042 blocks=7 obj#=11757 tim=1020996803
WAIT #6: nam='db file sequential read' ela= 220 file#=4 block#=656049 blocks=1 obj#=11757 tim=1020997605
WAIT #6: nam='db file scattered read' ela= 590 file#=4 block#=656050 blocks=7 obj#=11757 tim=1020998320
WAIT #6: nam='db file sequential read' ela= 214 file#=4 block#=656057 blocks=1 obj#=11757 tim=1020999123
WAIT #6: nam='db file scattered read' ela= 499 file#=4 block#=656058 blocks=7 obj#=11757 tim=1020999748
WAIT #6: nam='db file sequential read' ela= 248 file#=4 block#=656065 blocks=1 obj#=11757 tim=1021000585
WAIT #6: nam='db file scattered read' ela= 639 file#=4 block#=656066 blocks=7 obj#=11757 tim=1021001347
WAIT #6: nam='db file sequential read' ela= 237 file#=4 block#=656073 blocks=1 obj#=11757 tim=1021002179
WAIT #6: nam='db file scattered read' ela= 460 file#=4 block#=656074 blocks=7 obj#=11757 tim=1021002765
WAIT #6: nam='db file sequential read' ela= 221 file#=4 block#=656081 blocks=1 obj#=11757 tim=1021003601
WAIT #6: nam='db file scattered read' ela= 593 file#=4 block#=656082 blocks=7 obj#=11757 tim=1021004319
WAIT #6: nam='db file sequential read' ela= 213 file#=4 block#=656089 blocks=1 obj#=11757 tim=1021005119
WAIT #6: nam='db file scattered read' ela= 504 file#=4 block#=656090 blocks=7 obj#=11757 tim=1021005747
WAIT #6: nam='db file sequential read' ela= 281 file#=4 block#=656097 blocks=1 obj#=11757 tim=1021006621
WAIT #6: nam='db file scattered read' ela= 650 file#=4 block#=656098 blocks=7 obj#=11757 tim=1021007414
WAIT #6: nam='db file sequential read' ela= 243 file#=4 block#=656105 blocks=1 obj#=11757 tim=1021008251
WAIT #6: nam='db file scattered read' ela= 451 file#=4 block#=656106 blocks=7 obj#=11757 tim=1021008832
WAIT #6: nam='db file sequential read' ela= 212 file#=4 block#=656113 blocks=1 obj#=11757 tim=1021009602
WAIT #6: nam='db file scattered read' ela= 600 file#=4 block#=656114 blocks=7 obj#=11757 tim=1021010618
WAIT #6: nam='db file sequential read' ela= 214 file#=4 block#=656121 blocks=1 obj#=11757 tim=1021011421
WAIT #6: nam='db file scattered read' ela= 6915 file#=4 block#=656122 blocks=7 obj#=11757 tim=1021018458
WAIT #6: nam='db file sequential read' ela= 203 file#=4 block#=656129 blocks=1 obj#=11757 tim=1021019223
WAIT #6: nam='db file scattered read' ela= 704 file#=4 block#=656130 blocks=7 obj#=11757 tim=1021020050
WAIT #6: nam='db file sequential read' ela= 236 file#=4 block#=656139 blocks=1 obj#=11757 tim=1021020851
WAIT #6: nam='db file scattered read' ela= 454 file#=4 block#=656140 blocks=5 obj#=11757 tim=1021021419
WAIT #6: nam='db file sequential read' ela= 214 file#=4 block#=656145 blocks=1 obj#=11757 tim=1021022010
WAIT #6: nam='db file scattered read' ela= 597 file#=4 block#=656146 blocks=7 obj#=11757 tim=1021022722
WAIT #6: nam='db file sequential read' ela= 494 file#=4 block#=656153 blocks=1 obj#=11757 tim=1021023744
WAIT #6: nam='db file scattered read' ela= 444 file#=4 block#=656154 blocks=7 obj#=11757 tim=1021024302
WAIT #6: nam='db file sequential read' ela= 369 file#=4 block#=656161 blocks=1 obj#=11757 tim=1021025185
WAIT #6: nam='db file scattered read' ela= 583 file#=4 block#=656162 blocks=7 obj#=11757 tim=1021025976
WAIT #6: nam='db file sequential read' ela= 229 file#=4 block#=656169 blocks=1 obj#=11757 tim=1021026734
WAIT #6: nam='db file scattered read' ela= 463 file#=4 block#=656170 blocks=7 obj#=11757 tim=1021027310
WAIT #6: nam='db file sequential read' ela= 249 file#=4 block#=656177 blocks=1 obj#=11757 tim=1021028182
WAIT #6: nam='db file scattered read' ela= 583 file#=4 block#=656178 blocks=7 obj#=11757 tim=1021028905
WAIT #6: nam='db file sequential read' ela= 257 file#=4 block#=656185 blocks=1 obj#=11757 tim=1021029812
WAIT #6: nam='db file scattered read' ela= 430 file#=4 block#=656186 blocks=7 obj#=11757 tim=1021030375
WAIT #6: nam='db file sequential read' ela= 227 file#=4 block#=656193 blocks=1 obj#=11757 tim=1021031250
WAIT #6: nam='db file scattered read' ela= 583 file#=4 block#=656194 blocks=7 obj#=11757 tim=1021031966
WAIT #6: nam='db file sequential read' ela= 236 file#=4 block#=656201 blocks=1 obj#=11757 tim=1021032843
WAIT #6: nam='db file scattered read' ela= 483 file#=4 block#=656202 blocks=7 obj#=11757 tim=1021033457
WAIT #6: nam='db file sequential read' ela= 250 file#=4 block#=656209 blocks=1 obj#=11757 tim=1021034327
WAIT #6: nam='db file scattered read' ela= 595 file#=4 block#=656210 blocks=7 obj#=11757 tim=1021035052
WAIT #6: nam='db file sequential read' ela= 233 file#=4 block#=656217 blocks=1 obj#=11757 tim=1021035884
WAIT #6: nam='db file scattered read' ela= 437 file#=4 block#=656218 blocks=7 obj#=11757 tim=1021036444
WAIT #6: nam='db file sequential read' ela= 195 file#=4 block#=656225 blocks=1 obj#=11757 tim=1021037243
WAIT #6: nam='db file scattered read' ela= 572 file#=4 block#=656226 blocks=7 obj#=11757 tim=1021037963
WAIT #6: nam='db file sequential read' ela= 189 file#=4 block#=656233 blocks=1 obj#=11757 tim=1021038760
WAIT #6: nam='db file scattered read' ela= 450 file#=4 block#=656234 blocks=7 obj#=11757 tim=1021039339
WAIT #6: nam='db file sequential read' ela= 200 file#=4 block#=656241 blocks=1 obj#=11757 tim=1021040188
WAIT #6: nam='db file scattered read' ela= 579 file#=4 block#=656242 blocks=7 obj#=11757 tim=1021040896
WAIT #6: nam='db file sequential read' ela= 202 file#=4 block#=656249 blocks=1 obj#=11757 tim=1021041701
WAIT #6: nam='db file scattered read' ela= 569 file#=4 block#=656250 blocks=7 obj#=11757 tim=1021042400
WAIT #6: nam='db file sequential read' ela= 271 file#=4 block#=656257 blocks=1 obj#=11757 tim=1021043201
WAIT #6: nam='db file scattered read' ela= 631 file#=4 block#=656258 blocks=7 obj#=11757 tim=1021043955
WAIT #6: nam='db file sequential read' ela= 269 file#=4 block#=656267 blocks=1 obj#=11757 tim=1021044794
WAIT #6: nam='db file scattered read' ela= 447 file#=4 block#=656268 blocks=5 obj#=11757 tim=1021045362
WAIT #6: nam='db file sequential read' ela= 230 file#=4 block#=656273 blocks=1 obj#=11757 tim=1021046026
WAIT #6: nam='db file scattered read' ela= 589 file#=4 block#=656274 blocks=7 obj#=11757 tim=1021046739
WAIT #6: nam='db file sequential read' ela= 196 file#=4 block#=656281 blocks=1 obj#=11757 tim=1021047507
WAIT #6: nam='db file scattered read' ela= 497 file#=4 block#=656282 blocks=7 obj#=11757 tim=1021048128
WAIT #6: nam='db file sequential read' ela= 242 file#=4 block#=656289 blocks=1 obj#=11757 tim=1021048968
WAIT #6: nam='db file scattered read' ela= 648 file#=4 block#=656290 blocks=7 obj#=11757 tim=1021049761
WAIT #6: nam='db file sequential read' ela= 263 file#=4 block#=656297 blocks=1 obj#=11757 tim=1021050638
WAIT #6: nam='db file scattered read' ela= 448 file#=4 block#=656298 blocks=7 obj#=11757 tim=1021051216
WAIT #6: nam='db file sequential read' ela= 241 file#=4 block#=656305 blocks=1 obj#=11757 tim=1021052061
WAIT #6: nam='db file scattered read' ela= 582 file#=4 block#=656306 blocks=7 obj#=11757 tim=1021052774
WAIT #6: nam='db file sequential read' ela= 231 file#=4 block#=656313 blocks=1 obj#=11757 tim=1021053606
WAIT #6: nam='db file scattered read' ela= 16742 file#=4 block#=656314 blocks=7 obj#=11757 tim=1021070476
WAIT #6: nam='db file sequential read' ela= 229 file#=4 block#=656321 blocks=1 obj#=11757 tim=1021071284
WAIT #6: nam='db file scattered read' ela= 722 file#=4 block#=656322 blocks=7 obj#=11757 tim=1021072132
WAIT #6: nam='db file sequential read' ela= 218 file#=4 block#=656329 blocks=1 obj#=11757 tim=1021072956
WAIT #6: nam='db file scattered read' ela= 410 file#=4 block#=656330 blocks=7 obj#=11757 tim=1021073491
WAIT #6: nam='db file sequential read' ela= 190 file#=4 block#=656337 blocks=1 obj#=11757 tim=1021074253
WAIT #6: nam='db file scattered read' ela= 594 file#=4 block#=656338 blocks=7 obj#=11757 tim=1021074972
FETCH #6:c=31250,e=125135,p=488,cr=488,cu=0,mis=0,r=15,dep=0,og=1,tim=1021075477
WAIT #6: nam='SQL*Net message from client' ela= 3955 driver id=1413697536 #bytes=1 p3=0 obj#=11757 tim=1021079486
WAIT #6: nam='db file sequential read' ela= 262 file#=4 block#=656345 blocks=1 obj#=11757 tim=1021079913
WAIT #6: nam='db file scattered read' ela= 447 file#=4 block#=656346 blocks=7 obj#=11757 tim=1021080498
WAIT #6: nam='db file sequential read' ela= 225 file#=4 block#=656353 blocks=1 obj#=11757 tim=1021081378
WAIT #6: nam='db file scattered read' ela= 588 file#=4 block#=656354 blocks=7 obj#=11757 tim=1021082130
WAIT #6: nam='db file sequential read' ela= 231 file#=4 block#=656361 blocks=1 obj#=11757 tim=1021083048
WAIT #6: nam='db file scattered read' ela= 441 file#=4 block#=656362 blocks=7 obj#=11757 tim=1021083626
WAIT #6: nam='db file sequential read' ela= 209 file#=4 block#=656369 blocks=1 obj#=11757 tim=1021084498
WAIT #6: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11757 tim=1021084555
WAIT #6: nam='db file scattered read' ela= 585 file#=4 block#=656370 blocks=7 obj#=11757 tim=1021085253
WAIT #6: nam='db file sequential read' ela= 227 file#=4 block#=656377 blocks=1 obj#=11757 tim=1021086091
WAIT #6: nam='db file scattered read' ela= 579 file#=4 block#=656378 blocks=7 obj#=11757 tim=1021086795
WAIT #6: nam='db file sequential read' ela= 331 file#=4 block#=656385 blocks=1 obj#=11757 tim=1021087750
WAIT #6: nam='db file scattered read' ela= 682 file#=4 block#=656386 blocks=7 obj#=11757 tim=1021088775
WAIT #6: nam='db file sequential read' ela= 254 file#=4 block#=656395 blocks=1 obj#=11757 tim=1021089689
WAIT #6: nam='db file scattered read' ela= 459 file#=4 block#=656396 blocks=5 obj#=11757 tim=1021090289
WAIT #6: nam='db file sequential read' ela= 210 file#=4 block#=656401 blocks=1 obj#=11757 tim=1021090987
WAIT #6: nam='db file scattered read' ela= 604 file#=4 block#=656402 blocks=7 obj#=11757 tim=1021091740
WAIT #6: nam='db file sequential read' ela= 203 file#=4 block#=656409 blocks=1 obj#=11757 tim=1021092618
WAIT #6: nam='db file scattered read' ela= 453 file#=4 block#=656410 blocks=7 obj#=11757 tim=1021093212
WAIT #6: nam='db file sequential read' ela= 215 file#=4 block#=656417 blocks=1 obj#=11757 tim=1021094125
WAIT #6: nam='db file scattered read' ela= 597 file#=4 block#=656418 blocks=7 obj#=11757 tim=1021094880
WAIT #6: nam='db file sequential read' ela= 209 file#=4 block#=656425 blocks=1 obj#=11757 tim=1021095757
WAIT #6: nam='db file scattered read' ela= 445 file#=4 block#=656426 blocks=7 obj#=11757 tim=1021096334
WAIT #6: nam='db file sequential read' ela= 222 file#=4 block#=656433 blocks=1 obj#=11757 tim=1021097172
WAIT #6: nam='db file scattered read' ela= 587 file#=4 block#=656434 blocks=7 obj#=11757 tim=1021097888
WAIT #6: nam='db file sequential read' ela= 218 file#=4 block#=656441 blocks=1 obj#=11757 tim=1021098729
WAIT #6: nam='db file scattered read' ela= 461 file#=4 block#=656442 blocks=7 obj#=11757 tim=1021099319
WAIT #6: nam='db file sequential read' ela= 225 file#=4 block#=656449 blocks=1 obj#=11757 tim=1021100159
WAIT #6: nam='db file scattered read' ela= 622 file#=4 block#=656450 blocks=7 obj#=11757 tim=1021100915
WAIT #6: nam='db file sequential read' ela= 242 file#=4 block#=656457 blocks=1 obj#=11757 tim=1021101828
WAIT #6: nam='db file scattered read' ela= 445 file#=4 block#=656458 blocks=7 obj#=11757 tim=1021102407
WAIT #6: nam='db file sequential read' ela= 237 file#=4 block#=656465 blocks=1 obj#=11757 tim=1021103291
WAIT #6: nam='db file scattered read' ela= 617 file#=4 block#=656466 blocks=7 obj#=11757 tim=1021104045
WAIT #6: nam='db file sequential read' ela= 252 file#=4 block#=656473 blocks=1 obj#=11757 tim=1021104951
WAIT #6: nam='db file scattered read' ela= 501 file#=4 block#=656474 blocks=7 obj#=11757 tim=1021105580
WAIT #6: nam='db file sequential read' ela= 244 file#=4 block#=656481 blocks=1 obj#=11757 tim=1021106381
WAIT #6: nam='db file scattered read' ela= 642 file#=4 block#=656482 blocks=7 obj#=11757 tim=1021107175
WAIT #6: nam='db file sequential read' ela= 256 file#=4 block#=656489 blocks=1 obj#=11757 tim=1021108089
WAIT #6: nam='db file scattered read' ela= 481 file#=4 block#=656490 blocks=7 obj#=11757 tim=1021108704
WAIT #6: nam='db file sequential read' ela= 249 file#=4 block#=656497 blocks=1 obj#=11757 tim=1021109619
WAIT #6: nam='db file scattered read' ela= 621 file#=4 block#=656498 blocks=7 obj#=11757 tim=1021110376
WAIT #6: nam='db file sequential read' ela= 246 file#=4 block#=656505 blocks=1 obj#=11757 tim=1021111289
WAIT #6: nam='db file scattered read' ela= 470 file#=4 block#=656506 blocks=7 obj#=11757 tim=1021111880
WAIT #6: nam='db file sequential read' ela= 268 file#=4 block#=656513 blocks=1 obj#=11757 tim=1021112719
WAIT #6: nam='db file scattered read' ela= 2679 file#=4 block#=656514 blocks=7 obj#=11757 tim=1021115524
WAIT #6: nam='db file sequential read' ela= 230 file#=4 block#=656523 blocks=1 obj#=11757 tim=1021116324
WAIT #6: nam='db file scattered read' ela= 484 file#=4 block#=656524 blocks=5 obj#=11757 tim=1021116929
WAIT #6: nam='db file sequential read' ela= 254 file#=4 block#=656529 blocks=1 obj#=11757 tim=1021117587
WAIT #6: nam='db file scattered read' ela= 642 file#=4 block#=656530 blocks=7 obj#=11757 tim=1021118344
WAIT #6: nam='db file sequential read' ela= 275 file#=4 block#=656537 blocks=1 obj#=11757 tim=1021119298
WAIT #6: nam='db file scattered read' ela= 458 file#=4 block#=656538 blocks=7 obj#=11757 tim=1021119890
WAIT #6: nam='db file sequential read' ela= 483 file#=4 block#=656545 blocks=1 obj#=11757 tim=1021120917
WAIT #6: nam='db file scattered read' ela= 584 file#=4 block#=656546 blocks=7 obj#=11757 tim=1021121635
WAIT #6: nam='db file sequential read' ela= 206 file#=4 block#=656553 blocks=1 obj#=11757 tim=1021122363
WAIT #6: nam='db file scattered read' ela= 1859 file#=4 block#=656554 blocks=7 obj#=11757 tim=1021124339
WAIT #6: nam='db file sequential read' ela= 203 file#=4 block#=656561 blocks=1 obj#=11757 tim=1021125065
WAIT #6: nam='db file scattered read' ela= 716 file#=4 block#=656562 blocks=7 obj#=11757 tim=1021125897
WAIT #6: nam='db file sequential read' ela= 439 file#=4 block#=656569 blocks=1 obj#=11757 tim=1021127000
WAIT #6: nam='db file scattered read' ela= 435 file#=4 block#=656570 blocks=7 obj#=11757 tim=1021127555
WAIT #6: nam='db file sequential read' ela= 226 file#=4 block#=656577 blocks=1 obj#=11757 tim=1021128355
WAIT #6: nam='db file scattered read' ela= 596 file#=4 block#=656578 blocks=7 obj#=11757 tim=1021129072
WAIT #6: nam='db file sequential read' ela= 313 file#=4 block#=656585 blocks=1 obj#=11757 tim=1021129949
WAIT #6: nam='db file scattered read' ela= 454 file#=4 block#=656586 blocks=7 obj#=11757 tim=1021130523
WAIT #6: nam='db file sequential read' ela= 201 file#=4 block#=656593 blocks=1 obj#=11757 tim=1021131288
WAIT #6: nam='db file scattered read' ela= 597 file#=4 block#=656594 blocks=7 obj#=11757 tim=1021132010
WAIT #6: nam='db file sequential read' ela= 221 file#=4 block#=656601 blocks=1 obj#=11757 tim=1021132805
WAIT #6: nam='db file scattered read' ela= 434 file#=4 block#=656602 blocks=7 obj#=11757 tim=1021133360
WAIT #6: nam='db file sequential read' ela= 208 file#=4 block#=656609 blocks=1 obj#=11757 tim=1021134083
WAIT #6: nam='db file scattered read' ela= 602 file#=4 block#=656610 blocks=7 obj#=11757 tim=1021134818
WAIT #6: nam='db file sequential read' ela= 202 file#=4 block#=656617 blocks=1 obj#=11757 tim=1021135686
WAIT #6: nam='db file scattered read' ela= 709 file#=4 block#=656618 blocks=7 obj#=11757 tim=1021136517
WAIT #6: nam='db file sequential read' ela= 216 file#=4 block#=656625 blocks=1 obj#=11757 tim=1021137246
WAIT #6: nam='db file scattered read' ela= 600 file#=4 block#=656626 blocks=7 obj#=11757 tim=1021137961
WAIT #6: nam='db file sequential read' ela= 216 file#=4 block#=656633 blocks=1 obj#=11757 tim=1021138688
WAIT #6: nam='db file scattered read' ela= 515 file#=4 block#=656634 blocks=7 obj#=11757 tim=1021139317
WAIT #6: nam='db file sequential read' ela= 237 file#=4 block#=656641 blocks=1 obj#=11757 tim=1021140192
WAIT #6: nam='db file scattered read' ela= 619 file#=4 block#=656642 blocks=7 obj#=11757 tim=1021140948
WAIT #6: nam='db file sequential read' ela= 259 file#=4 block#=656651 blocks=1 obj#=11757 tim=1021141862
WAIT #6: nam='db file scattered read' ela= 435 file#=4 block#=656652 blocks=5 obj#=11757 tim=1021142428
WAIT #6: nam='db file sequential read' ela= 219 file#=4 block#=656657 blocks=1 obj#=11757 tim=1021143127
WAIT #6: nam='db file scattered read' ela= 581 file#=4 block#=656658 blocks=7 obj#=11757 tim=1021143844
WAIT #6: nam='db file sequential read' ela= 225 file#=4 block#=656665 blocks=1 obj#=11757 tim=1021144721
WAIT #6: nam='db file scattered read' ela= 465 file#=4 block#=656666 blocks=7 obj#=11757 tim=1021145313
WAIT #6: nam='db file sequential read' ela= 277 file#=4 block#=656673 blocks=1 obj#=11757 tim=1021146192
WAIT #6: nam='db file scattered read' ela= 647 file#=4 block#=656674 blocks=7 obj#=11757 tim=1021146981
WAIT #6: nam='db file sequential read' ela= 265 file#=4 block#=656681 blocks=1 obj#=11757 tim=1021147857
WAIT #6: nam='db file scattered read' ela= 453 file#=4 block#=656682 blocks=7 obj#=11757 tim=1021148433
WAIT #6: nam='db file sequential read' ela= 210 file#=4 block#=656689 blocks=1 obj#=11757 tim=1021149157
WAIT #6: nam='db file scattered read' ela= 622 file#=4 block#=656690 blocks=7 obj#=11757 tim=1021149912
WAIT #6: nam='db file sequential read' ela= 241 file#=4 block#=656697 blocks=1 obj#=11757 tim=1021150789
WAIT #6: nam='db file scattered read' ela= 500 file#=4 block#=656698 blocks=7 obj#=11757 tim=1021151422
WAIT #6: nam='db file sequential read' ela= 252 file#=4 block#=656705 blocks=1 obj#=11757 tim=1021152298
WAIT #6: nam='db file scattered read' ela= 6769 file#=4 block#=656706 blocks=7 obj#=11757 tim=1021159200
WAIT #6: nam='db file sequential read' ela= 207 file#=4 block#=656713 blocks=1 obj#=11757 tim=1021160041
WAIT #6: nam='db file scattered read' ela= 481 file#=4 block#=656714 blocks=7 obj#=11757 tim=1021160654
WAIT #6: nam='db file sequential read' ela= 263 file#=4 block#=656721 blocks=1 obj#=11757 tim=1021161604
WAIT #6: nam='db file scattered read' ela= 625 file#=4 block#=656722 blocks=7 obj#=11757 tim=1021162400
WAIT #6: nam='db file sequential read' ela= 267 file#=4 block#=656729 blocks=1 obj#=11757 tim=1021163501
WAIT #6: nam='db file scattered read' ela= 437 file#=4 block#=656730 blocks=7 obj#=11757 tim=1021164096
WAIT #6: nam='db file sequential read' ela= 207 file#=4 block#=656737 blocks=1 obj#=11757 tim=1021165127
WAIT #6: nam='db file scattered read' ela= 613 file#=4 block#=656738 blocks=7 obj#=11757 tim=1021165920
WAIT #6: nam='db file sequential read' ela= 203 file#=4 block#=656745 blocks=1 obj#=11757 tim=1021166872
WAIT #6: nam='db file scattered read' ela= 502 file#=4 block#=656746 blocks=7 obj#=11757 tim=1021167522
WAIT #6: nam='db file sequential read' ela= 241 file#=4 block#=656753 blocks=1 obj#=11757 tim=1021168510
WAIT #6: nam='db file scattered read' ela= 640 file#=4 block#=656754 blocks=7 obj#=11757 tim=1021169305
WAIT #6: nam='db file sequential read' ela= 263 file#=4 block#=656761 blocks=1 obj#=11757 tim=1021170331
WAIT #6: nam='db file scattered read' ela= 453 file#=4 block#=656762 blocks=7 obj#=11757 tim=1021170926
WAIT #6: nam='db file sequential read' ela= 243 file#=4 block#=656769 blocks=1 obj#=11757 tim=1021171859
WAIT #6: nam='db file scattered read' ela= 604 file#=4 block#=656770 blocks=7 obj#=11757 tim=1021172600
WAIT #6: nam='db file sequential read' ela= 204 file#=4 block#=656779 blocks=1 obj#=11757 tim=1021173510
WAIT #6: nam='db file scattered read' ela= 6459 file#=4 block#=656780 blocks=5 obj#=11757 tim=1021180109
FETCH #6:c=31250,e=101015,p=432,cr=431,cu=0,mis=0,r=15,dep=0,og=1,tim=1021180550
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
```
16KB
```
PARSING IN CURSOR #13 len=83 dep=0 uid=30 oct=3 lid=30 tim=1024186075 hv=3216823004 ad='510b945c'
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
END OF STMT
PARSE #13:c=15625,e=18270,p=1,cr=99,cu=0,mis=1,r=0,dep=0,og=1,tim=1024186071
EXEC #13:c=0,e=34,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1024186257
WAIT #13: nam='SQL*Net message to client' ela= 3 driver id=1413697536 #bytes=1 p3=0 obj#=11767 tim=1024186294
WAIT #13: nam='db file sequential read' ela= 319 file#=4 block#=323592 blocks=1 obj#=11767 tim=1024186677
WAIT #13: nam='db file sequential read' ela= 22252 file#=4 block#=324032 blocks=1 obj#=11767 tim=1024208974
WAIT #13: nam='db file sequential read' ela= 260 file#=4 block#=323593 blocks=1 obj#=11767 tim=1024209271
WAIT #13: nam='db file scattered read' ela= 543 file#=4 block#=323594 blocks=3 obj#=11767 tim=1024210056
WAIT #13: nam='db file sequential read' ela= 303 file#=4 block#=323597 blocks=1 obj#=11767 tim=1024210896
WAIT #13: nam='db file scattered read' ela= 3503 file#=4 block#=323598 blocks=3 obj#=11767 tim=1024214604
WAIT #13: nam='db file sequential read' ela= 278 file#=4 block#=323601 blocks=1 obj#=11767 tim=1024215411
WAIT #13: nam='db file scattered read' ela= 616 file#=4 block#=323602 blocks=3 obj#=11767 tim=1024216240
WAIT #13: nam='db file sequential read' ela= 274 file#=4 block#=323606 blocks=1 obj#=11767 tim=1024217049
WAIT #13: nam='db file scattered read' ela= 413 file#=4 block#=323607 blocks=2 obj#=11767 tim=1024217669

FETCH #13:c=0,e=31566,p=17,cr=17,cu=0,mis=0,r=1,dep=0,og=1,tim=1024217894
WAIT #13: nam='SQL*Net message from client' ela= 17848 driver id=1413697536 #bytes=1 p3=0 obj#=11767 tim=1024235802
WAIT #13: nam='db file sequential read' ela= 327 file#=4 block#=323609 blocks=1 obj#=11767 tim=1024236357
WAIT #13: nam='db file scattered read' ela= 603 file#=4 block#=323610 blocks=3 obj#=11767 tim=1024237180
WAIT #13: nam='db file sequential read' ela= 305 file#=4 block#=323613 blocks=1 obj#=11767 tim=1024238018
WAIT #13: nam='db file scattered read' ela= 446 file#=4 block#=323614 blocks=3 obj#=11767 tim=1024238671
WAIT #13: nam='db file sequential read' ela= 274 file#=4 block#=323617 blocks=1 obj#=11767 tim=1024239468
WAIT #13: nam='db file scattered read' ela= 524 file#=4 block#=323618 blocks=3 obj#=11767 tim=1024240195
WAIT #13: nam='db file sequential read' ela= 244 file#=4 block#=323622 blocks=1 obj#=11767 tim=1024240955
WAIT #13: nam='db file scattered read' ela= 496 file#=4 block#=323623 blocks=2 obj#=11767 tim=1024241664
WAIT #13: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11767 tim=1024241790
WAIT #13: nam='db file sequential read' ela= 291 file#=4 block#=323625 blocks=1 obj#=11767 tim=1024242345
WAIT #13: nam='db file scattered read' ela= 771 file#=4 block#=323626 blocks=3 obj#=11767 tim=1024243319
WAIT #13: nam='db file sequential read' ela= 308 file#=4 block#=323629 blocks=1 obj#=11767 tim=1024244127
WAIT #13: nam='db file scattered read' ela= 456 file#=4 block#=323630 blocks=3 obj#=11767 tim=1024244778
WAIT #13: nam='db file sequential read' ela= 275 file#=4 block#=323633 blocks=1 obj#=11767 tim=1024245578
WAIT #13: nam='db file scattered read' ela= 552 file#=4 block#=323634 blocks=3 obj#=11767 tim=1024246324
WAIT #13: nam='db file sequential read' ela= 258 file#=4 block#=323638 blocks=1 obj#=11767 tim=1024247103
WAIT #13: nam='db file scattered read' ela= 532 file#=4 block#=323639 blocks=2 obj#=11767 tim=1024247830
WAIT #13: nam='db file sequential read' ela= 347 file#=4 block#=323641 blocks=1 obj#=11767 tim=1024248531
WAIT #13: nam='db file scattered read' ela= 654 file#=4 block#=323642 blocks=3 obj#=11767 tim=1024249388
WAIT #13: nam='db file sequential read' ela= 283 file#=4 block#=323645 blocks=1 obj#=11767 tim=1024250190
WAIT #13: nam='db file scattered read' ela= 430 file#=4 block#=323646 blocks=3 obj#=11767 tim=1024250812
WAIT #13: nam='db file sequential read' ela= 302 file#=4 block#=323649 blocks=1 obj#=11767 tim=1024251619
WAIT #13: nam='db file scattered read' ela= 541 file#=4 block#=323650 blocks=3 obj#=11767 tim=1024252358
WAIT #13: nam='db file sequential read' ela= 270 file#=4 block#=323654 blocks=1 obj#=11767 tim=1024253136
WAIT #13: nam='db file scattered read' ela= 420 file#=4 block#=323655 blocks=2 obj#=11767 tim=1024253750
WAIT #13: nam='db file sequential read' ela= 295 file#=4 block#=323657 blocks=1 obj#=11767 tim=1024254406
WAIT #13: nam='db file scattered read' ela= 549 file#=4 block#=323658 blocks=3 obj#=11767 tim=1024255163
WAIT #13: nam='db file sequential read' ela= 302 file#=4 block#=323661 blocks=1 obj#=11767 tim=1024256003
WAIT #13: nam='db file scattered read' ela= 449 file#=4 block#=323662 blocks=3 obj#=11767 tim=1024256659
WAIT #13: nam='db file sequential read' ela= 295 file#=4 block#=323665 blocks=1 obj#=11767 tim=1024257499
WAIT #13: nam='db file scattered read' ela= 549 file#=4 block#=323666 blocks=3 obj#=11767 tim=1024258254
WAIT #13: nam='db file sequential read' ela= 309 file#=4 block#=323669 blocks=1 obj#=11767 tim=1024259086
WAIT #13: nam='db file scattered read' ela= 485 file#=4 block#=323670 blocks=3 obj#=11767 tim=1024259785
WAIT #13: nam='db file sequential read' ela= 302 file#=4 block#=323673 blocks=1 obj#=11767 tim=1024260633
WAIT #13: nam='db file scattered read' ela= 571 file#=4 block#=323674 blocks=3 obj#=11767 tim=1024261423
WAIT #13: nam='db file sequential read' ela= 12632 file#=4 block#=323677 blocks=1 obj#=11767 tim=1024274604
WAIT #13: nam='db file scattered read' ela= 466 file#=4 block#=323678 blocks=3 obj#=11767 tim=1024275289
WAIT #13: nam='db file sequential read' ela= 308 file#=4 block#=323681 blocks=1 obj#=11767 tim=1024276162
WAIT #13: nam='db file scattered read' ela= 574 file#=4 block#=323682 blocks=3 obj#=11767 tim=1024276950
WAIT #13: nam='db file sequential read' ela= 320 file#=4 block#=323685 blocks=1 obj#=11767 tim=1024277831
WAIT #13: nam='db file scattered read' ela= 448 file#=4 block#=323686 blocks=3 obj#=11767 tim=1024278499
WAIT #13: nam='db file sequential read' ela= 294 file#=4 block#=323689 blocks=1 obj#=11767 tim=1024279338
WAIT #13: nam='db file scattered read' ela= 537 file#=4 block#=323690 blocks=3 obj#=11767 tim=1024280101
WAIT #13: nam='db file sequential read' ela= 253 file#=4 block#=323693 blocks=1 obj#=11767 tim=1024280906
WAIT #13: nam='db file scattered read' ela= 461 file#=4 block#=323694 blocks=3 obj#=11767 tim=1024281586
WAIT #13: nam='db file sequential read' ela= 331 file#=4 block#=323697 blocks=1 obj#=11767 tim=1024282462
WAIT #13: nam='db file scattered read' ela= 585 file#=4 block#=323698 blocks=3 obj#=11767 tim=1024283249
WAIT #13: nam='db file sequential read' ela= 344 file#=4 block#=323701 blocks=1 obj#=11767 tim=1024284094
WAIT #13: nam='db file scattered read' ela= 428 file#=4 block#=323702 blocks=3 obj#=11767 tim=1024284724
WAIT #13: nam='db file sequential read' ela= 278 file#=4 block#=323705 blocks=1 obj#=11767 tim=1024285490
WAIT #13: nam='db file scattered read' ela= 520 file#=4 block#=323706 blocks=3 obj#=11767 tim=1024286211
WAIT #13: nam='db file sequential read' ela= 255 file#=4 block#=323709 blocks=1 obj#=11767 tim=1024286975
WAIT #13: nam='db file scattered read' ela= 492 file#=4 block#=323710 blocks=3 obj#=11767 tim=1024287661
WAIT #13: nam='db file sequential read' ela= 346 file#=4 block#=323713 blocks=1 obj#=11767 tim=1024288534
WAIT #13: nam='db file scattered read' ela= 589 file#=4 block#=323714 blocks=3 obj#=11767 tim=1024289329
WAIT #13: nam='db file sequential read' ela= 7561 file#=4 block#=323718 blocks=1 obj#=11767 tim=1024297420
WAIT #13: nam='db file scattered read' ela= 440 file#=4 block#=323719 blocks=2 obj#=11767 tim=1024298063
WAIT #13: nam='db file sequential read' ela= 310 file#=4 block#=323721 blocks=1 obj#=11767 tim=1024298732
WAIT #13: nam='db file scattered read' ela= 560 file#=4 block#=323722 blocks=3 obj#=11767 tim=1024299515
WAIT #13: nam='db file sequential read' ela= 289 file#=4 block#=323725 blocks=1 obj#=11767 tim=1024300320
WAIT #13: nam='db file scattered read' ela= 417 file#=4 block#=323726 blocks=3 obj#=11767 tim=1024300940
WAIT #13: nam='db file sequential read' ela= 272 file#=4 block#=323729 blocks=1 obj#=11767 tim=1024301753
WAIT #13: nam='db file scattered read' ela= 530 file#=4 block#=323730 blocks=3 obj#=11767 tim=1024302503
WAIT #13: nam='db file sequential read' ela= 279 file#=4 block#=323733 blocks=1 obj#=11767 tim=1024303341
WAIT #13: nam='db file scattered read' ela= 447 file#=4 block#=323734 blocks=3 obj#=11767 tim=1024303999
WAIT #13: nam='db file sequential read' ela= 306 file#=4 block#=323737 blocks=1 obj#=11767 tim=1024304874
WAIT #13: nam='db file scattered read' ela= 558 file#=4 block#=323738 blocks=3 obj#=11767 tim=1024305663
WAIT #13: nam='db file sequential read' ela= 281 file#=4 block#=323741 blocks=1 obj#=11767 tim=1024306511
WAIT #13: nam='db file scattered read' ela= 447 file#=4 block#=323742 blocks=3 obj#=11767 tim=1024307165
WAIT #13: nam='db file sequential read' ela= 253 file#=4 block#=323745 blocks=1 obj#=11767 tim=1024307938
WAIT #13: nam='db file scattered read' ela= 514 file#=4 block#=323746 blocks=3 obj#=11767 tim=1024308652
WAIT #13: nam='db file sequential read' ela= 273 file#=4 block#=323749 blocks=1 obj#=11767 tim=1024309455
WAIT #13: nam='db file scattered read' ela= 455 file#=4 block#=323750 blocks=3 obj#=11767 tim=1024310110
WAIT #13: nam='db file sequential read' ela= 288 file#=4 block#=323753 blocks=1 obj#=11767 tim=1024310922
WAIT #13: nam='db file scattered read' ela= 569 file#=4 block#=323754 blocks=3 obj#=11767 tim=1024311702
WAIT #13: nam='db file sequential read' ela= 302 file#=4 block#=323757 blocks=1 obj#=11767 tim=1024312517
WAIT #13: nam='db file scattered read' ela= 452 file#=4 block#=323758 blocks=3 obj#=11767 tim=1024313163
WAIT #13: nam='db file sequential read' ela= 278 file#=4 block#=323761 blocks=1 obj#=11767 tim=1024313973
WAIT #13: nam='db file scattered read' ela= 616 file#=4 block#=323762 blocks=3 obj#=11767 tim=1024314784
WAIT #13: nam='db file sequential read' ela= 273 file#=4 block#=323765 blocks=1 obj#=11767 tim=1024315592
WAIT #13: nam='db file scattered read' ela= 8154 file#=4 block#=323766 blocks=3 obj#=11767 tim=1024323953
WAIT #13: nam='db file sequential read' ela= 791 file#=4 block#=323769 blocks=1 obj#=11767 tim=1024325261
WAIT #13: nam='db file scattered read' ela= 847 file#=4 block#=323770 blocks=3 obj#=11767 tim=1024326317
WAIT #13: nam='db file sequential read' ela= 259 file#=4 block#=323773 blocks=1 obj#=11767 tim=1024327126
WAIT #13: nam='db file scattered read' ela= 434 file#=4 block#=323774 blocks=3 obj#=11767 tim=1024327768
WAIT #13: nam='db file sequential read' ela= 285 file#=4 block#=323777 blocks=1 obj#=11767 tim=1024328571
WAIT #13: nam='db file scattered read' ela= 543 file#=4 block#=323778 blocks=3 obj#=11767 tim=1024329318
WAIT #13: nam='db file sequential read' ela= 264 file#=4 block#=323782 blocks=1 obj#=11767 tim=1024330137
WAIT #13: nam='db file scattered read' ela= 414 file#=4 block#=323783 blocks=2 obj#=11767 tim=1024330776
WAIT #13: nam='db file sequential read' ela= 269 file#=4 block#=323785 blocks=1 obj#=11767 tim=1024331412
WAIT #13: nam='db file scattered read' ela= 538 file#=4 block#=323786 blocks=3 obj#=11767 tim=1024332159
WAIT #13: nam='db file sequential read' ela= 278 file#=4 block#=323789 blocks=1 obj#=11767 tim=1024332956
WAIT #13: nam='db file scattered read' ela= 458 file#=4 block#=323790 blocks=3 obj#=11767 tim=1024333613
WAIT #13: nam='db file sequential read' ela= 309 file#=4 block#=323793 blocks=1 obj#=11767 tim=1024334447
WAIT #13: nam='db file scattered read' ela= 565 file#=4 block#=323794 blocks=3 obj#=11767 tim=1024335211
WAIT #13: nam='db file sequential read' ela= 303 file#=4 block#=323797 blocks=1 obj#=11767 tim=1024336040
WAIT #13: nam='db file scattered read' ela= 435 file#=4 block#=323798 blocks=3 obj#=11767 tim=1024336673
WAIT #13: nam='db file sequential read' ela= 310 file#=4 block#=323801 blocks=1 obj#=11767 tim=1024337557
WAIT #13: nam='db file scattered read' ela= 564 file#=4 block#=323802 blocks=3 obj#=11767 tim=1024338343
WAIT #13: nam='db file sequential read' ela= 305 file#=4 block#=323805 blocks=1 obj#=11767 tim=1024339216
WAIT #13: nam='db file scattered read' ela= 474 file#=4 block#=323806 blocks=3 obj#=11767 tim=1024339916
WAIT #13: nam='db file sequential read' ela= 337 file#=4 block#=323809 blocks=1 obj#=11767 tim=1024340822
WAIT #13: nam='db file scattered read' ela= 547 file#=4 block#=323810 blocks=3 obj#=11767 tim=1024341590
WAIT #13: nam='db file sequential read' ela= 340 file#=4 block#=323813 blocks=1 obj#=11767 tim=1024342492
WAIT #13: nam='db file scattered read' ela= 468 file#=4 block#=323814 blocks=3 obj#=11767 tim=1024343164
WAIT #13: nam='db file sequential read' ela= 7563 file#=4 block#=323817 blocks=1 obj#=11767 tim=1024351272
WAIT #13: nam='db file scattered read' ela= 823 file#=4 block#=323818 blocks=3 obj#=11767 tim=1024352346
WAIT #13: nam='db file sequential read' ela= 268 file#=4 block#=323821 blocks=1 obj#=11767 tim=1024353156
WAIT #13: nam='db file scattered read' ela= 499 file#=4 block#=323822 blocks=3 obj#=11767 tim=1024353875
WAIT #13: nam='db file sequential read' ela= 321 file#=4 block#=323825 blocks=1 obj#=11767 tim=1024354750

FETCH #13:c=31250,e=119085,p=212,cr=213,cu=0,mis=0,r=15,dep=0,og=1,tim=1024354937
WAIT #13: nam='SQL*Net message from client' ela= 3925 driver id=1413697536 #bytes=1 p3=0 obj#=11767 tim=1024358914
WAIT #13: nam='db file scattered read' ela= 612 file#=4 block#=323826 blocks=3 obj#=11767 tim=1024359642
WAIT #13: nam='db file sequential read' ela= 321 file#=4 block#=323829 blocks=1 obj#=11767 tim=1024360596
WAIT #13: nam='db file scattered read' ela= 425 file#=4 block#=323830 blocks=3 obj#=11767 tim=1024361263
WAIT #13: nam='db file sequential read' ela= 263 file#=4 block#=323833 blocks=1 obj#=11767 tim=1024362148
WAIT #13: nam='db file scattered read' ela= 507 file#=4 block#=323834 blocks=3 obj#=11767 tim=1024362894
WAIT #13: nam='db file sequential read' ela= 263 file#=4 block#=323837 blocks=1 obj#=11767 tim=1024363816
WAIT #13: nam='db file scattered read' ela= 5647 file#=4 block#=323838 blocks=3 obj#=11767 tim=1024369696
WAIT #13: nam='db file sequential read' ela= 267 file#=4 block#=323841 blocks=1 obj#=11767 tim=1024370586
WAIT #13: nam='db file scattered read' ela= 627 file#=4 block#=323842 blocks=3 obj#=11767 tim=1024371447
WAIT #13: nam='db file sequential read' ela= 254 file#=4 block#=323846 blocks=1 obj#=11767 tim=1024372328
WAIT #13: nam='db file scattered read' ela= 438 file#=4 block#=323847 blocks=2 obj#=11767 tim=1024373007
WAIT #13: nam='db file sequential read' ela= 280 file#=4 block#=323849 blocks=1 obj#=11767 tim=1024373711
WAIT #13: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11767 tim=1024373854
WAIT #13: nam='db file scattered read' ela= 538 file#=4 block#=323850 blocks=3 obj#=11767 tim=1024374503
WAIT #13: nam='db file sequential read' ela= 301 file#=4 block#=323853 blocks=1 obj#=11767 tim=1024375308
WAIT #13: nam='db file scattered read' ela= 419 file#=4 block#=323854 blocks=3 obj#=11767 tim=1024375922
WAIT #13: nam='db file sequential read' ela= 281 file#=4 block#=323857 blocks=1 obj#=11767 tim=1024376721
WAIT #13: nam='db file scattered read' ela= 548 file#=4 block#=323858 blocks=3 obj#=11767 tim=1024377471
WAIT #13: nam='db file sequential read' ela= 257 file#=4 block#=323861 blocks=1 obj#=11767 tim=1024378251
WAIT #13: nam='db file scattered read' ela= 462 file#=4 block#=323862 blocks=3 obj#=11767 tim=1024378907
WAIT #13: nam='db file sequential read' ela= 309 file#=4 block#=323865 blocks=1 obj#=11767 tim=1024379743
WAIT #13: nam='db file scattered read' ela= 568 file#=4 block#=323866 blocks=3 obj#=11767 tim=1024380501
WAIT #13: nam='db file sequential read' ela= 296 file#=4 block#=323869 blocks=1 obj#=11767 tim=1024381312
WAIT #13: nam='db file scattered read' ela= 447 file#=4 block#=323870 blocks=3 obj#=11767 tim=1024381956
WAIT #13: nam='db file sequential read' ela= 256 file#=4 block#=323873 blocks=1 obj#=11767 tim=1024382690
WAIT #13: nam='db file scattered read' ela= 540 file#=4 block#=323874 blocks=3 obj#=11767 tim=1024383429
WAIT #13: nam='db file sequential read' ela= 249 file#=4 block#=323877 blocks=1 obj#=11767 tim=1024384168
WAIT #13: nam='db file scattered read' ela= 3728 file#=4 block#=323878 blocks=3 obj#=11767 tim=1024388085
WAIT #13: nam='db file sequential read' ela= 302 file#=4 block#=323881 blocks=1 obj#=11767 tim=1024388898
WAIT #13: nam='db file scattered read' ela= 618 file#=4 block#=323882 blocks=3 obj#=11767 tim=1024389725
WAIT #13: nam='db file sequential read' ela= 280 file#=4 block#=323885 blocks=1 obj#=11767 tim=1024390529
WAIT #13: nam='db file scattered read' ela= 492 file#=4 block#=323886 blocks=3 obj#=11767 tim=1024391218
WAIT #13: nam='db file sequential read' ela= 306 file#=4 block#=323889 blocks=1 obj#=11767 tim=1024392050
WAIT #13: nam='db file scattered read' ela= 560 file#=4 block#=323890 blocks=3 obj#=11767 tim=1024392807
WAIT #13: nam='db file sequential read' ela= 312 file#=4 block#=323893 blocks=1 obj#=11767 tim=1024393643
WAIT #13: nam='db file scattered read' ela= 684 file#=4 block#=323894 blocks=3 obj#=11767 tim=1024394542
WAIT #13: nam='db file sequential read' ela= 256 file#=4 block#=323897 blocks=1 obj#=11767 tim=1024395317
WAIT #13: nam='db file scattered read' ela= 534 file#=4 block#=323898 blocks=3 obj#=11767 tim=1024396061
WAIT #13: nam='db file sequential read' ela= 279 file#=4 block#=323901 blocks=1 obj#=11767 tim=1024396866
WAIT #13: nam='db file scattered read' ela= 477 file#=4 block#=323902 blocks=3 obj#=11767 tim=1024397550
WAIT #13: nam='db file sequential read' ela= 326 file#=4 block#=323905 blocks=1 obj#=11767 tim=1024398387
WAIT #13: nam='db file scattered read' ela= 579 file#=4 block#=323906 blocks=3 obj#=11767 tim=1024399172
WAIT #13: nam='db file sequential read' ela= 294 file#=4 block#=323910 blocks=1 obj#=11767 tim=1024399989
WAIT #13: nam='db file scattered read' ela= 413 file#=4 block#=323911 blocks=2 obj#=11767 tim=1024400600
WAIT #13: nam='db file sequential read' ela= 281 file#=4 block#=323913 blocks=1 obj#=11767 tim=1024401228
WAIT #13: nam='db file scattered read' ela= 517 file#=4 block#=323914 blocks=3 obj#=11767 tim=1024401949
WAIT #13: nam='db file sequential read' ela= 261 file#=4 block#=323917 blocks=1 obj#=11767 tim=1024402711
WAIT #13: nam='db file scattered read' ela= 489 file#=4 block#=323918 blocks=3 obj#=11767 tim=1024403394
WAIT #13: nam='db file sequential read' ela= 325 file#=4 block#=323921 blocks=1 obj#=11767 tim=1024404232
WAIT #13: nam='db file scattered read' ela= 594 file#=4 block#=323922 blocks=3 obj#=11767 tim=1024405021
WAIT #13: nam='db file sequential read' ela= 319 file#=4 block#=323925 blocks=1 obj#=11767 tim=1024405863
WAIT #13: nam='db file scattered read' ela= 455 file#=4 block#=323926 blocks=3 obj#=11767 tim=1024406531
WAIT #13: nam='db file sequential read' ela= 300 file#=4 block#=323929 blocks=1 obj#=11767 tim=1024407368
WAIT #13: nam='db file scattered read' ela= 553 file#=4 block#=323930 blocks=3 obj#=11767 tim=1024408126
WAIT #13: nam='db file sequential read' ela= 302 file#=4 block#=323933 blocks=1 obj#=11767 tim=1024408965
WAIT #13: nam='db file scattered read' ela= 4196 file#=4 block#=323934 blocks=3 obj#=11767 tim=1024413368
WAIT #13: nam='db file sequential read' ela= 277 file#=4 block#=323937 blocks=1 obj#=11767 tim=1024414174
WAIT #13: nam='db file scattered read' ela= 629 file#=4 block#=323938 blocks=3 obj#=11767 tim=1024415006
WAIT #13: nam='db file sequential read' ela= 269 file#=4 block#=323941 blocks=1 obj#=11767 tim=1024415817
WAIT #13: nam='db file scattered read' ela= 439 file#=4 block#=323942 blocks=3 obj#=11767 tim=1024416473
WAIT #13: nam='db file sequential read' ela= 309 file#=4 block#=323945 blocks=1 obj#=11767 tim=1024417318
WAIT #13: nam='db file scattered read' ela= 573 file#=4 block#=323946 blocks=3 obj#=11767 tim=1024418105
WAIT #13: nam='db file sequential read' ela= 311 file#=4 block#=323949 blocks=1 obj#=11767 tim=1024418941
WAIT #13: nam='db file scattered read' ela= 402 file#=4 block#=323950 blocks=3 obj#=11767 tim=1024419560
WAIT #13: nam='db file sequential read' ela= 263 file#=4 block#=323953 blocks=1 obj#=11767 tim=1024420363
WAIT #13: nam='db file scattered read' ela= 511 file#=4 block#=323954 blocks=3 obj#=11767 tim=1024421076
WAIT #13: nam='db file sequential read' ela= 274 file#=4 block#=323957 blocks=1 obj#=11767 tim=1024421851
WAIT #13: nam='db file scattered read' ela= 5925 file#=4 block#=323958 blocks=3 obj#=11767 tim=1024427971
WAIT #13: nam='db file sequential read' ela= 776 file#=4 block#=323961 blocks=1 obj#=11767 tim=1024429278
WAIT #13: nam='db file scattered read' ela= 888 file#=4 block#=323962 blocks=3 obj#=11767 tim=1024430362
WAIT #13: nam='db file sequential read' ela= 280 file#=4 block#=323965 blocks=1 obj#=11767 tim=1024431168
WAIT #13: nam='db file scattered read' ela= 453 file#=4 block#=323966 blocks=3 obj#=11767 tim=1024431820
WAIT #13: nam='db file sequential read' ela= 275 file#=4 block#=323969 blocks=1 obj#=11767 tim=1024432592
WAIT #13: nam='db file scattered read' ela= 563 file#=4 block#=323970 blocks=3 obj#=11767 tim=1024433336
WAIT #13: nam='db file sequential read' ela= 277 file#=4 block#=323974 blocks=1 obj#=11767 tim=1024434067
WAIT #13: nam='db file scattered read' ela= 397 file#=4 block#=323975 blocks=2 obj#=11767 tim=1024434647
WAIT #13: nam='db file sequential read' ela= 278 file#=4 block#=323977 blocks=1 obj#=11767 tim=1024435234
WAIT #13: nam='db file scattered read' ela= 528 file#=4 block#=323978 blocks=3 obj#=11767 tim=1024435945
WAIT #13: nam='db file sequential read' ela= 277 file#=4 block#=323981 blocks=1 obj#=11767 tim=1024436671
WAIT #13: nam='db file scattered read' ela= 465 file#=4 block#=323982 blocks=3 obj#=11767 tim=1024437319
WAIT #13: nam='db file sequential read' ela= 288 file#=4 block#=323985 blocks=1 obj#=11767 tim=1024438055
WAIT #13: nam='db file scattered read' ela= 567 file#=4 block#=323986 blocks=3 obj#=11767 tim=1024438827
WAIT #13: nam='db file sequential read' ela= 310 file#=4 block#=323989 blocks=1 obj#=11767 tim=1024439750
WAIT #13: nam='db file scattered read' ela= 425 file#=4 block#=323990 blocks=3 obj#=11767 tim=1024440422
WAIT #13: nam='db file sequential read' ela= 252 file#=4 block#=323993 blocks=1 obj#=11767 tim=1024441259
WAIT #13: nam='db file scattered read' ela= 526 file#=4 block#=323994 blocks=3 obj#=11767 tim=1024442008
WAIT #13: nam='db file sequential read' ela= 250 file#=4 block#=323997 blocks=1 obj#=11767 tim=1024442852
WAIT #13: nam='db file scattered read' ela= 461 file#=4 block#=323998 blocks=3 obj#=11767 tim=1024443542
WAIT #13: nam='db file sequential read' ela= 335 file#=4 block#=324001 blocks=1 obj#=11767 tim=1024444417
WAIT #13: nam='db file scattered read' ela= 571 file#=4 block#=324002 blocks=3 obj#=11767 tim=1024445204
WAIT #13: nam='db file sequential read' ela= 337 file#=4 block#=324005 blocks=1 obj#=11767 tim=1024446085
WAIT #13: nam='db file scattered read' ela= 465 file#=4 block#=324006 blocks=3 obj#=11767 tim=1024446756
WAIT #13: nam='db file sequential read' ela= 296 file#=4 block#=324009 blocks=1 obj#=11767 tim=1024447596
WAIT #13: nam='db file scattered read' ela= 542 file#=4 block#=324010 blocks=3 obj#=11767 tim=1024448348
WAIT #13: nam='db file sequential read' ela= 325 file#=4 block#=324013 blocks=1 obj#=11767 tim=1024449233
WAIT #13: nam='db file scattered read' ela= 477 file#=4 block#=324014 blocks=3 obj#=11767 tim=1024449917
WAIT #13: nam='db file sequential read' ela= 342 file#=4 block#=324017 blocks=1 obj#=11767 tim=1024450797
WAIT #13: nam='db file scattered read' ela= 573 file#=4 block#=324018 blocks=3 obj#=11767 tim=1024451578
WAIT #13: nam='db file sequential read' ela= 337 file#=4 block#=324021 blocks=1 obj#=11767 tim=1024452461
WAIT #13: nam='db file scattered read' ela= 5088 file#=4 block#=324022 blocks=3 obj#=11767 tim=1024457763
WAIT #13: nam='db file sequential read' ela= 256 file#=4 block#=324025 blocks=1 obj#=11767 tim=1024458570
WAIT #13: nam='db file scattered read' ela= 642 file#=4 block#=324026 blocks=3 obj#=11767 tim=1024459427
WAIT #13: nam='db file scattered read' ela= 393 file#=4 block#=324029 blocks=3 obj#=11767 tim=1024460340
WAIT #13: nam='db file sequential read' ela= 7291 file#=4 block#=324033 blocks=1 obj#=11767 tim=1024468137
WAIT #13: nam='db file scattered read' ela= 852 file#=4 block#=324034 blocks=3 obj#=11767 tim=1024469197
WAIT #13: nam='db file sequential read' ela= 247 file#=4 block#=324038 blocks=1 obj#=11767 tim=1024469964
WAIT #13: nam='db file scattered read' ela= 408 file#=4 block#=324039 blocks=2 obj#=11767 tim=1024470579
WAIT #13: nam='db file sequential read' ela= 278 file#=4 block#=324041 blocks=1 obj#=11767 tim=1024471207
WAIT #13: nam='db file scattered read' ela= 535 file#=4 block#=324042 blocks=3 obj#=11767 tim=1024471951
WAIT #13: nam='db file sequential read' ela= 265 file#=4 block#=324045 blocks=1 obj#=11767 tim=1024472755
WAIT #13: nam='db file scattered read' ela= 400 file#=4 block#=324046 blocks=3 obj#=11767 tim=1024473371
WAIT #13: nam='db file sequential read' ela= 264 file#=4 block#=324049 blocks=1 obj#=11767 tim=1024474179
WAIT #13: nam='db file scattered read' ela= 538 file#=4 block#=324050 blocks=3 obj#=11767 tim=1024474922
WAIT #13: nam='db file sequential read' ela= 264 file#=4 block#=324053 blocks=1 obj#=11767 tim=1024475735
WAIT #13: nam='db file scattered read' ela= 486 file#=4 block#=324054 blocks=3 obj#=11767 tim=1024476429
WAIT #13: nam='db file sequential read' ela= 293 file#=4 block#=324057 blocks=1 obj#=11767 tim=1024477263
WAIT #13: nam='db file scattered read' ela= 545 file#=4 block#=324058 blocks=3 obj#=11767 tim=1024478014
WAIT #13: nam='db file sequential read' ela= 295 file#=4 block#=324061 blocks=1 obj#=11767 tim=1024478821
WAIT #13: nam='db file scattered read' ela= 428 file#=4 block#=324062 blocks=3 obj#=11767 tim=1024479444
WAIT #13: nam='db file sequential read' ela= 278 file#=4 block#=324065 blocks=1 obj#=11767 tim=1024480241
WAIT #13: nam='db file scattered read' ela= 550 file#=4 block#=324066 blocks=3 obj#=11767 tim=1024480987
WAIT #13: nam='db file sequential read' ela= 476 file#=4 block#=324069 blocks=1 obj#=11767 tim=1024481960
WAIT #13: nam='db file scattered read' ela= 445 file#=4 block#=324070 blocks=3 obj#=11767 tim=1024482613


FETCH #13:c=31250,e=123807,p=242,cr=241,cu=0,mis=0,r=15,dep=0,og=1,tim=1024482772
WAIT #13: nam='SQL*Net message from client' ela= 3935 driver id=1413697536 #bytes=1 p3=0 obj#=11767 tim=1024486760
WAIT #13: nam='db file sequential read' ela= 309 file#=4 block#=324073 blocks=1 obj#=11767 tim=1024487564
WAIT #13: nam='db file scattered read' ela= 569 file#=4 block#=324074 blocks=3 obj#=11767 tim=1024488374
WAIT #13: nam='db file sequential read' ela= 306 file#=4 block#=324077 blocks=1 obj#=11767 tim=1024489265
WAIT #13: nam='db file scattered read' ela= 490 file#=4 block#=324078 blocks=3 obj#=11767 tim=1024489986
WAIT #13: nam='db file sequential read' ela= 309 file#=4 block#=324081 blocks=1 obj#=11767 tim=1024490874
WAIT #13: nam='db file scattered read' ela= 553 file#=4 block#=324082 blocks=3 obj#=11767 tim=1024491658
WAIT #13: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11767 tim=1024491913
WAIT #13: nam='db file sequential read' ela= 312 file#=4 block#=324085 blocks=1 obj#=11767 tim=1024492572
WAIT #13: nam='db file scattered read' ela= 450 file#=4 block#=324086 blocks=3 obj#=11767 tim=1024493240
WAIT #13: nam='db file sequential read' ela= 300 file#=4 block#=324089 blocks=1 obj#=11767 tim=1024494083
WAIT #13: nam='db file scattered read' ela= 567 file#=4 block#=324090 blocks=3 obj#=11767 tim=1024494861
WAIT #13: nam='db file sequential read' ela= 305 file#=4 block#=324093 blocks=1 obj#=11767 tim=1024495716
WAIT #13: nam='db file scattered read' ela= 461 file#=4 block#=324094 blocks=3 obj#=11767 tim=1024496401
WAIT #13: nam='db file sequential read' ela= 317 file#=4 block#=324097 blocks=1 obj#=11767 tim=1024497277
WAIT #13: nam='db file scattered read' ela= 595 file#=4 block#=324098 blocks=3 obj#=11767 tim=1024498102
WAIT #13: nam='db file sequential read' ela= 282 file#=4 block#=324102 blocks=1 obj#=11767 tim=1024498991
WAIT #13: nam='db file scattered read' ela= 6797 file#=4 block#=324103 blocks=2 obj#=11767 tim=1024506013
WAIT #13: nam='db file sequential read' ela= 460 file#=4 block#=324105 blocks=1 obj#=11767 tim=1024506877
WAIT #13: nam='db file scattered read' ela= 1285 file#=4 block#=324106 blocks=3 obj#=11767 tim=1024508384
WAIT #13: nam='db file sequential read' ela= 267 file#=4 block#=324109 blocks=1 obj#=11767 tim=1024509266
WAIT #13: nam='db file scattered read' ela= 482 file#=4 block#=324110 blocks=3 obj#=11767 tim=1024509988
WAIT #13: nam='db file sequential read' ela= 328 file#=4 block#=324113 blocks=1 obj#=11767 tim=1024510861
WAIT #13: nam='db file scattered read' ela= 575 file#=4 block#=324114 blocks=3 obj#=11767 tim=1024511647
WAIT #13: nam='db file sequential read' ela= 321 file#=4 block#=324117 blocks=1 obj#=11767 tim=1024512531
WAIT #13: nam='db file scattered read' ela= 457 file#=4 block#=324118 blocks=3 obj#=11767 tim=1024513199
WAIT #13: nam='db file sequential read' ela= 279 file#=4 block#=324121 blocks=1 obj#=11767 tim=1024514043
WAIT #13: nam='db file scattered read' ela= 540 file#=4 block#=324122 blocks=3 obj#=11767 tim=1024514796
WAIT #13: nam='db file sequential read' ela= 272 file#=4 block#=324125 blocks=1 obj#=11767 tim=1024515641
WAIT #13: nam='db file scattered read' ela= 496 file#=4 block#=324126 blocks=3 obj#=11767 tim=1024516362
WAIT #13: nam='db file sequential read' ela= 343 file#=4 block#=324129 blocks=1 obj#=11767 tim=1024517273
WAIT #13: nam='db file scattered read' ela= 593 file#=4 block#=324130 blocks=3 obj#=11767 tim=1024518116
WAIT #13: nam='db file sequential read' ela= 2842 file#=4 block#=324133 blocks=1 obj#=11767 tim=1024521533
WAIT #13: nam='db file scattered read' ela= 436 file#=4 block#=324134 blocks=3 obj#=11767 tim=1024522199
WAIT #13: nam='db file sequential read' ela= 282 file#=4 block#=324137 blocks=1 obj#=11767 tim=1024523048
WAIT #13: nam='db file scattered read' ela= 557 file#=4 block#=324138 blocks=3 obj#=11767 tim=1024523824
WAIT #13: nam='db file sequential read' ela= 304 file#=4 block#=324141 blocks=1 obj#=11767 tim=1024524709
WAIT #13: nam='db file scattered read' ela= 426 file#=4 block#=324142 blocks=3 obj#=11767 tim=1024525363
WAIT #13: nam='db file sequential read' ela= 261 file#=4 block#=324145 blocks=1 obj#=11767 tim=1024526204
WAIT #13: nam='db file scattered read' ela= 550 file#=4 block#=324146 blocks=3 obj#=11767 tim=1024526978
WAIT #13: nam='db file sequential read' ela= 254 file#=4 block#=324149 blocks=1 obj#=11767 tim=1024527798
WAIT #13: nam='db file scattered read' ela= 467 file#=4 block#=324150 blocks=3 obj#=11767 tim=1024528459
WAIT #13: nam='db file sequential read' ela= 292 file#=4 block#=324153 blocks=1 obj#=11767 tim=1024529235
WAIT #13: nam='db file scattered read' ela= 560 file#=4 block#=324154 blocks=3 obj#=11767 tim=1024529980
WAIT #13: nam='db file sequential read' ela= 6640 file#=4 block#=324157 blocks=1 obj#=11767 tim=1024537121
WAIT #13: nam='db file scattered read' ela= 458 file#=4 block#=324158 blocks=3 obj#=11767 tim=1024537803
WAIT #13: nam='db file sequential read' ela= 260 file#=4 block#=324161 blocks=1 obj#=11767 tim=1024538537
WAIT #13: nam='db file scattered read' ela= 525 file#=4 block#=324162 blocks=3 obj#=11767 tim=1024539250
WAIT #13: nam='db file sequential read' ela= 271 file#=4 block#=324166 blocks=1 obj#=11767 tim=1024540012
WAIT #13: nam='db file scattered read' ela= 399 file#=4 block#=324167 blocks=2 obj#=11767 tim=1024540601
WAIT #13: nam='db file sequential read' ela= 294 file#=4 block#=324169 blocks=1 obj#=11767 tim=1024541212
WAIT #13: nam='db file scattered read' ela= 538 file#=4 block#=324170 blocks=3 obj#=11767 tim=1024541935
WAIT #13: nam='db file sequential read' ela= 258 file#=4 block#=324173 blocks=1 obj#=11767 tim=1024542666
WAIT #13: nam='db file scattered read' ela= 542 file#=4 block#=324174 blocks=3 obj#=11767 tim=1024543389
WAIT #13: nam='db file sequential read' ela= 250 file#=4 block#=324177 blocks=1 obj#=11767 tim=1024544113
WAIT #13: nam='db file scattered read' ela= 514 file#=4 block#=324178 blocks=3 obj#=11767 tim=1024544817
WAIT #13: nam='db file sequential read' ela= 260 file#=4 block#=324181 blocks=1 obj#=11767 tim=1024545549
WAIT #13: nam='db file scattered read' ela= 434 file#=4 block#=324182 blocks=3 obj#=11767 tim=1024546184
WAIT #13: nam='db file sequential read' ela= 269 file#=4 block#=324185 blocks=1 obj#=11767 tim=1024546954
WAIT #13: nam='db file scattered read' ela= 540 file#=4 block#=324186 blocks=3 obj#=11767 tim=1024547695
WAIT #13: nam='db file sequential read' ela= 280 file#=4 block#=324189 blocks=1 obj#=11767 tim=1024548473
WAIT #13: nam='db file scattered read' ela= 472 file#=4 block#=324190 blocks=3 obj#=11767 tim=1024549147
WAIT #13: nam='db file sequential read' ela= 299 file#=4 block#=324193 blocks=1 obj#=11767 tim=1024549955
WAIT #13: nam='db file scattered read' ela= 555 file#=4 block#=324194 blocks=3 obj#=11767 tim=1024550705
WAIT #13: nam='db file sequential read' ela= 307 file#=4 block#=324197 blocks=1 obj#=11767 tim=1024551501
WAIT #13: nam='db file scattered read' ela= 447 file#=4 block#=324198 blocks=3 obj#=11767 tim=1024552138
WAIT #13: nam='db file sequential read' ela= 277 file#=4 block#=324201 blocks=1 obj#=11767 tim=1024552870
WAIT #13: nam='db file scattered read' ela= 526 file#=4 block#=324202 blocks=3 obj#=11767 tim=1024553575
WAIT #13: nam='db file sequential read' ela= 248 file#=4 block#=324205 blocks=1 obj#=11767 tim=1024554269
WAIT #13: nam='db file scattered read' ela= 462 file#=4 block#=324206 blocks=3 obj#=11767 tim=1024554916
WAIT #13: nam='db file sequential read' ela= 340 file#=4 block#=324209 blocks=1 obj#=11767 tim=1024555720
WAIT #13: nam='db file scattered read' ela= 568 file#=4 block#=324210 blocks=3 obj#=11767 tim=1024556467
WAIT #13: nam='db file sequential read' ela= 336 file#=4 block#=324213 blocks=1 obj#=11767 tim=1024557268
WAIT #13: nam='db file scattered read' ela= 457 file#=4 block#=324214 blocks=3 obj#=11767 tim=1024557907
WAIT #13: nam='db file sequential read' ela= 284 file#=4 block#=324217 blocks=1 obj#=11767 tim=1024558792
WAIT #13: nam='db file scattered read' ela= 537 file#=4 block#=324218 blocks=3 obj#=11767 tim=1024559572
WAIT #13: nam='db file sequential read' ela= 296 file#=4 block#=324221 blocks=1 obj#=11767 tim=1024560451
WAIT #13: nam='db file scattered read' ela= 484 file#=4 block#=324222 blocks=3 obj#=11767 tim=1024561181
WAIT #13: nam='db file sequential read' ela= 321 file#=4 block#=324225 blocks=1 obj#=11767 tim=1024562089
WAIT #13: nam='db file scattered read' ela= 570 file#=4 block#=324226 blocks=3 obj#=11767 tim=1024562877
WAIT #13: nam='db file sequential read' ela= 307 file#=4 block#=324230 blocks=1 obj#=11767 tim=1024563727
WAIT #13: nam='db file scattered read' ela= 448 file#=4 block#=324231 blocks=2 obj#=11767 tim=1024564386
WAIT #13: nam='db file sequential read' ela= 312 file#=4 block#=324233 blocks=1 obj#=11767 tim=1024565083
WAIT #13: nam='db file scattered read' ela= 547 file#=4 block#=324234 blocks=3 obj#=11767 tim=1024565833
WAIT #13: nam='db file sequential read' ela= 313 file#=4 block#=324237 blocks=1 obj#=11767 tim=1024566672
WAIT #13: nam='db file scattered read' ela= 487 file#=4 block#=324238 blocks=3 obj#=11767 tim=1024567362
WAIT #13: nam='db file sequential read' ela= 321 file#=4 block#=324241 blocks=1 obj#=11767 tim=1024568213
WAIT #13: nam='db file scattered read' ela= 592 file#=4 block#=324242 blocks=3 obj#=11767 tim=1024569021
WAIT #13: nam='db file sequential read' ela= 328 file#=4 block#=324245 blocks=1 obj#=11767 tim=1024569911
WAIT #13: nam='db file scattered read' ela= 7974 file#=4 block#=324246 blocks=3 obj#=11767 tim=1024578096
WAIT #13: nam='db file sequential read' ela= 361 file#=4 block#=324249 blocks=1 obj#=11767 tim=1024579019
WAIT #13: nam='db file scattered read' ela= 1221 file#=4 block#=324250 blocks=3 obj#=11767 tim=1024580454
WAIT #13: nam='db file sequential read' ela= 266 file#=4 block#=324253 blocks=1 obj#=11767 tim=1024581295
WAIT #13: nam='db file scattered read' ela= 5233 file#=4 block#=324254 blocks=3 obj#=11767 tim=1024586753
WAIT #13: nam='db file sequential read' ela= 273 file#=4 block#=324257 blocks=1 obj#=11767 tim=1024587716
WAIT #13: nam='db file scattered read' ela= 540 file#=4 block#=324258 blocks=3 obj#=11767 tim=1024588544
WAIT #13: nam='db file sequential read' ela= 279 file#=4 block#=324261 blocks=1 obj#=11767 tim=1024589574
WAIT #13: nam='db file scattered read' ela= 459 file#=4 block#=324262 blocks=3 obj#=11767 tim=1024590313
WAIT #13: nam='db file sequential read' ela= 319 file#=4 block#=324265 blocks=1 obj#=11767 tim=1024591343
WAIT #13: nam='db file scattered read' ela= 535 file#=4 block#=324266 blocks=3 obj#=11767 tim=1024592130
WAIT #13: nam='db file sequential read' ela= 305 file#=4 block#=324269 blocks=1 obj#=11767 tim=1024593083
WAIT #13: nam='db file scattered read' ela= 449 file#=4 block#=324270 blocks=3 obj#=11767 tim=1024593774
WAIT #13: nam='db file sequential read' ela= 269 file#=4 block#=324273 blocks=1 obj#=11767 tim=1024594688
WAIT #13: nam='db file scattered read' ela= 517 file#=4 block#=324274 blocks=3 obj#=11767 tim=1024595453
WAIT #13: nam='db file sequential read' ela= 269 file#=4 block#=324277 blocks=1 obj#=11767 tim=1024596369
WAIT #13: nam='db file scattered read' ela= 455 file#=4 block#=324278 blocks=3 obj#=11767 tim=1024597065
WAIT #13: nam='db file sequential read' ela= 305 file#=4 block#=324281 blocks=1 obj#=11767 tim=1024597983
WAIT #13: nam='db file scattered read' ela= 547 file#=4 block#=324282 blocks=3 obj#=11767 tim=1024598772
WAIT #13: nam='db file sequential read' ela= 311 file#=4 block#=324285 blocks=1 obj#=11767 tim=1024599695



FETCH #13:c=46875,e=113065,p=210,cr=213,cu=0,mis=0,r=15,dep=0,og=1,tim=1024599878
```

Looking further into the trace file, at roughly 12.564243
seconds after the multi-block followed by single block read cycle started, the 8KB database switched to single block reads. At roughly 14.484744 seconds after the multi-block followed by single block read cycle started, the 16KB database switched to single block reads.

The average of 10 single block reads in the 8KB database is 0.0002403 seconds, with an occasional odd read of roughly 0.0004 or 0.0176 seconds. The average of 10 single block reads in the 16KB database is 0.000326 seconds, with an occasional odd read of roughly 0.0111 or 0.0347 seconds.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Added point at which trace file switches to single block reads.
Message was edited by:
Charles Hooper

Re: Larger vs. Small data block
Posted: Jun 8, 2008 3:37 PM in response to: Jonathan Lewis
Reply

Re: Larger vs. Small data block
Posted: Jun 8, 2008 4:52 PM in response to: Jonathan Lewis
Reply

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 8, 2008 5:03 PM in response to:
Reply

> >> you're supposed to design a theory to match the
> facts, not select the facts to match the theory.
>
> I think it's the other way around, Jonathan, the
> scientific method requires that you start with a
> hypothesis.
>

That's just so funny I had to preserve it for posterity. I'm sure a lot of readers on this forum have noticed how selective you are in what you quote from Metalink and other sources - now we know why you can't stop doing it.

You're supposed to start with observations (facts), then construct a theory, then make predictions based on the theory, then test the theory to see if the predictions are correct.

Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 8, 2008 5:04 PM in response to:
Reply

> Have you read the Oracle Corporation benchmark on different blocksizes?
>
> http://oss.oracle.com/~mason/blocksizes/

This benchmark is for a **filesystem**, not an **Oracle database**. Perhaps you could explain its relevancy.

http://btrfs.wiki.kernel.org/index.php/Main_Page

Also, why do you feel this benchmark is acceptable to cite?
- They certainly do not use ODM and do a random sample ,etc.
- It is not from a production system
- It does not seem the person posted their credentials

In fact, this filesystem is not even production ready:
"Btrfs is under heavy development, and is not suitable for any uses other than benchmarking and review."

So what exactly is it that you see in the real world, and can you offer an explanation of why you see what you do? (And please include some technical content like metrics etc.)

Just another comment about the experiments that have been executed on this thread: No one (that I recall) has made any sweeping statements or broad generalizations. It has merely been: in this case we observe <this> and can explain it by <whatever> and <these> metrics support the observation. Now that may or may not be the case in other situations, but at least one has learned how to analyzed the data and can do further experiments to explain other observations. Have we tested the complete universe of possibilities? Of course not, nor is it feasible. But it does not make what has been learned and observed any less relevant. My goal in participating in this forum is to educate, inform and mentor by example. Often things not binary, hence the frequent response "it depends".

As I have mentioned before, there are times when block size can make a difference, but frequently it does not. The purpose here is to understand when it matters and when it doesn't, if it matters why does it, and how to quantify it. I think a similar, related topic is partitioning. There are times when partitioning can benefit, and there are times when it does not.

--
Regards,

Greg Rahn
http://structureddata.org

Re: Larger vs. Small data block
Posted: Jun 8, 2008 5:19 PM in response to: Jonathan Lewis
Reply

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 8, 2008 5:20 PM in response to: Charles Hooper
Reply

Charles,

In the most recent post you've labelled the tests 7 and 11 - but I think from a couple of posts back they were 8 and 11. (In either case we're talking about the 'select distinct' that does an index full scan with a "sort unique nosort").

There is an oddity with the results, though. The tkprof summaries show no "db file scattered reads", but the trace outputs do show scattered reads - is this from repeating the test ?

I made a mistake in my earlier comment, by the way. An ***index full scan*** is the ideal operation for Oracle to do index prefetching, which usually means using the "db file parallel read" - a non-contiguous multi-block read. In this case though, where the index is newly created, leaf blocks that are logically adjacent will also be physically adjacent in the data extents, so the "db file parallel read" won't be used and Oracle should use the "db file scattered read" mechanism to collect adjacent leaf blocks. This explains the appearance (though not distribution) of the scattered reads in your trace file.

The parameter _db_file_noncontig_mblock_read_count is supposed to limit the number of blocks in a single "db file parallel read", and there are a couple of related parameters (_ncmb_readahead_enabled, _ncmb_readahead_tracing) that are supposed to enable it and allow tracing. The default for the limit is 11 blocks - which could allow a very large index scan to operate more efficiently in a tablespace with a larger block size - but I have no idea what might happen when a paralllel read 'collapses' to a scattered read - maybe the 11 limit still applies, rather than the db_file_multiblock_read_count limit.

The timings are quite revealing - I think it's safe to assume that a reported time for a read that falls in the region of 350 microseconds isn't a disk read, but a memory fetch from a cache somewhere. So the "slow, quick quick quick ..." pattern may be giving us a clue about an asynchronous readahead mechanism.

If you see odd patterns of this scattered read effect switching itself on and off, that's because CKPT controls the feature, and decides (every three seconds, I think) whether or not Oracle's "index prefetch" mechanism should be used.

There's always more to think of when the results show large deviations from expected behaviour - the possible interference of pre-fetch and caching makes me wonder how much CPU time was consumed outside Oracle when you were doing the different tests.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Re: Larger vs. Small data block
Posted: Jun 8, 2008 5:41 PM in response to: Greg Rahn

Reply

Niall Litchfield

Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 8, 2008 5:41 PM in response to:

Reply

> >> the conventional wisdom is that changes in block
> size may occasionally help, may occasionally cause
> problems, and typically are an irrelevant waste of
> effort.
>
> No, that's not how the vast majority of working
> Oracle professional experience different blocksizes.
> Not even close.. .

Well, the vast majority of databases I've seen have used default blocksizes and so see no difference, you have a collection of 5 or 6 quotes some of which are given by people who explicitly disagree with you and Jonathan has a conventional wisdom which corresponds pretty much to the consensus of posters on internet forums. None of those are really hard evidence though are they.
>
> Have you read the Oracle Corporation benchmark on
> different blocksizes?
>
> http://oss.oracle.com/~mason/blocksizes/

I have, I'm curious - I assume you do know that it's a test of a non-production ready filesystem for Linux don't you and has in fact nothing whatsoever to do with the Oracle database? From the project home "Btrfs is under heavy development, and is not suitable for any uses other than benchmarking and review. The Btrfs disk format is not yet finalized".

> >> developing tests and sharing observations.
>
> That's fun and interesting, but it's not science, and
> it definitively not the scientific method . . . .

Though curiously it does rather seem to be a method frequently used by scientists... Wikipedia has it about right

1) use your experience and knowledge
2) form a conjecture
3) come up with some predictions
4) perform tests of the predictions.

That's pretty much what is going on here.

> I'll keep saying it:
>
>
> - A single negative test case DOES NOT prove that a
> general concept is wrong. It's a shame that you have
> conned people into believing this nonsense.

We're in good company here though - look up Michelson-Morley..

> - Contrived tests DO NOT represent the real world.
> If you want valid observations, use one of your
> client systems.

So does the double-slit experiment not represent the real world? After all light doesn't often pass though double slits to form interference patterns in the real world.

> - Artificial tests can easily be manipulated by
> adjusting any one of hundreds of interveining
> factors. Hence, they are COMPLETELY INVALID as the
> basis for any general conclusions.

Hence they are completely valid as evidence to be disclosed reported and discussed. Client systems tend not to be like that, though I'm sure your clients who evidently agree for you to publish results based on their systems on the internet are an enlightened bunch.

> You guys are just chasing your tails. By your own

> admission, you have degress in the Arts, not Science,
> and your idea of valid testing is very different from
> what I see in the real world . . . . .

Must have missed that admission, can you remind us what your scientific qualifications are since they seem important to you.

> These individual tests don't matter, what matters is
> the conclusions that are drawn from them!

A truly enlightening quote

Niall Litchfield
http://www.orawin.info/

Message was edited to avoid bad bolding by:
Niall Litchfield

Re: Larger vs. Small data block
Posted: Jun 8, 2008 5:51 PM in response to: Niall Litchfield

Reply

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 8, 2008 6:01 PM in response to:

Reply

[nobr]> >> Wikipedia has it about right
>
> Good caveat! It changes from minute to minute . . .
>
> It's mostly anonymous junk, IMHO:
>

And yet only a few weeks ago you seemed to think it was good enough to quote for your own purposes.

http://forums.oracle.com/forums/thread.jspa?messageID=2521515?

You always want to have it both ways, don't you.

Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk[/nobr]

Niall Litchfield

Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 8, 2008 6:11 PM in response to:

Reply

> Hi Greg,
>
> >> Also, why do you feel this benchmark is acceptable
> to cite?
>
> Because the external I/O subsystem is a HUGE factor
> in the choice of blocksize:
>
> http://www.dba-oracle.com/t_physical_io_disk_metrics.h
> tm
>
> Stripe size, the speed of the I/O channels, and most
> of all, the PHYSICAL blocking all impact the choice
> of the "best" blocksize.

And that explains why a benchmark of file transfers on a filesystem that you can't run Oracle on is relevant how?

>
> >> No one (that I recall) has made any sweeping
> statements or broad generalizations.
>
> Huh?
>
> What do you call this?
>
> *“the conventional wisdom is that changes in block*
> *size may occasionally help, may occasionally cause*
> *problems, and typically are an irrelevant waste of*
> *effort.”*
>
> That’s just not true. You can say it as often as you
> want, but the choice of blocksize can have a PROFOUND
> impact on performance.

what do you mean by "can"? Do you mean "will", "will usually" or "will sometimes"? Jonathan's statement does not, at least in British English, preclude your statement from being true sometimes.

> Let’s be clear, the only problem that I have with
> this exercise (besides the validity issue), is the
> pretense that it is “scientific”.
>
> It’s not scientifically valid. Drop this "Oracle
> Scientist" stuff, and admit it.

How about logical and rational, as contrasted with illogical and irrational, will that do?

> >> My goal in participating in this forum is to
> educate
>
> If you really mean that, the real way to educate us
> is to tell us about all of the things that you see in
> the field.. . Tell war stories, tales from the
> trenches, that's how people learn.

People learn in a variety of ways and from a variety of different things. Stories from experienced individuals are one way, repeatable demonstrations are another, making your own mistakes is a third, study of literature is a fourth and so on. Greg and Charles are contributing educational material as surely as I telling tales of "databases I have crashed" has done in the past.

> >> Often things not binary, hence the frequent
> response "it depends".
>
> I agree!

I do, though the true test is the ability to state, rationally and in a method that stands up to scrutiny, upon **what** it depends.

> All we have in the world of Oracle performance is the
> human intuition that comes from years of hand-on
> experience with real-world databases.

Have you considered the quality of Bordeaux wines? A rather long time ago it was discovered that perhaps actually there were definable factors at work in what made a great bordeaux and that in fact, just maybe, understanding what affected the wine and how was a better bet than trusting the judgement of the human intuition of the self-appointed experts. (see http://query.nytimes.com/gst/fullpage.html?res=9C0CE7DD1731F937A35750C0A966958260) for example. Perhaps, just maybe, the performance of engineered systems such as Oracle might also be amenable to similar analysis.

Niall Litchfield
http://www.orawin.info/

By the way check out the quality rating of the 89 and 90 Vintages that were being predicted at the time of the NYT article.
http://www.wineontheweb.com/vintage/112_years/112_years.html

>
> There are NO ABSOLUTES, NO PROOFS, like you say "it
> depends".

Re: Larger vs. Small data block
Posted: Jun 8, 2008 6:23 PM in response to: Niall Litchfield

Reply

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 8, 2008 7:49 PM in response to: Jonathan Lewis

Reply

Based on what I am seeing, there is not any statistical difference between using 8k or 16k blocks in either a FTS or a Hash Join.

The following were performed on 10.2.0.3, 32-bit Linux.

Full Table Scan

8k db & table

```
********************************************************************************

select * from WEB_RETURNS_8K

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch    71978     37.55      67.72     102778     173743          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71980     37.55      67.72     102778     173743          0     7197670

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS_8K (cr=173743 pr=102778 pw=0 time=28832076 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71980        0.00          0.09
  SQL*Net message from client                 71980        0.00         67.19
  db file sequential read                         1        0.00          0.00
  db file scattered read                        807        0.06          7.01
  SQL*Net more data to client                359883        0.00         26.08

********************************************************************************
```

16k db & table

```
********************************************************************************

select * from WEB_RETURNS_16K

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch    71978     36.34      66.90      50726     122225          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71980     36.34      66.90      50726     122225          0     7197670

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS_16K (cr=122225 pr=50726 pw=0 time=21634648 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71980        0.00          0.09
  SQL*Net message from client                 71980        0.00         67.37
  db file sequential read                         1        0.00          0.00
  db file scattered read                        797        0.03          6.88
  SQL*Net more data to client                359883        0.00         26.46

********************************************************************************
```

Hash Join

8k db & table

```
********************************************************************************

select count(*)
from WEB_RETURNS_8K a, WEB_RETURNS_8KB b
where a.WR_ORDER_NUMBER = b.WR_ORDER_NUMBER

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     29.77      41.93     224869     205580          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     29.77      41.93     224869     205580          0           1

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=205580 pr=224869 pw=19313 time=41938967 us)
15516562   HASH JOIN  (cr=205580 pr=224869 pw=19313 time=48568651 us)
7197670    TABLE ACCESS FULL WEB_RETURNS_8K (cr=102790 pr=102778 pw=0 time=21639417 us)
7197670    TABLE ACCESS FULL WEB_RETURNS_8KB (cr=102790 pr=102778 pw=0 time=21606062 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         2        0.00          0.00
  db file scattered read                       1614        0.03         12.77
  direct path write temp                        623        0.00          0.50
  direct path read temp                         623        0.02          0.22
  SQL*Net message from client                     2      128.39        128.39

********************************************************************************
```

16k db & table

```
********************************************************************************

select count(*)
from WEB_RETURNS_16K a, WEB_RETURNS_16KB b
where a.WR_ORDER_NUMBER = b.WR_ORDER_NUMBER

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     28.52      41.35     110602     101474          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     28.52      41.35     110602     101474          0           1

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=101474 pr=110602 pw=9150 time=41353262 us)
15516562   HASH JOIN  (cr=101474 pr=110602 pw=9150 time=48030575 us)
7197670    TABLE ACCESS FULL WEB_RETURNS_16K (cr=50737 pr=50726 pw=0 time=14443360 us)
7197670    TABLE ACCESS FULL WEB_RETURNS_16KB (cr=50737 pr=50726 pw=0 time=21624217 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         2        0.00          0.00
  db file scattered read                       1594        0.05         13.51
  direct path write temp                        610        0.00          0.49
  direct path read temp                         610        0.00          0.16
  SQL*Net message from client                     2        6.94          6.94

********************************************************************************
```

I think Charles Hooper mentioned he was seeing the few reads are smaller than the MBRC and Jonathan Lewis mentioned that in an ASSM tablespace the extent starts with 64k and then it increases. That is correct, MBRC wont cross extents. In my test case I used an initial extent size of 100m and you can see that the MBRC immediately kicks in reading 128 8k blocks (1MB) at a time. The 1 block read is the segment header.

```
PARSING IN CURSOR #2 len=46 dep=0 uid=25 oct=3 lid=25 tim=1184537616560639 hv=1224141136 ad='79f6dd60'
select * from WEB_RETURNS_8k
END OF STMT
PARSE #2:c=0,e=58,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1184537616560634
EXEC #2:c=0,e=70,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1184537616561573
WAIT #2: nam='db file sequential read' ela= 2589 file#=4 block#=26 blocks=1 obj#=9793 tim=1184537616564289
WAIT #2: nam='db file scattered read' ela= 23961 file#=4 block#=27 blocks=128 obj#=9793 tim=1184537616590692
WAIT #2: nam='db file scattered read' ela= 9452 file#=4 block#=155 blocks=128 obj#=9793 tim=1184537616870244
WAIT #2: nam='db file scattered read' ela= 7807 file#=4 block#=283 blocks=128 obj#=9793 tim=1184537617053121
WAIT #2: nam='db file scattered read' ela= 7819 file#=4 block#=411 blocks=128 obj#=9793 tim=1184537617214832
WAIT #2: nam='db file scattered read' ela= 7809 file#=4 block#=539 blocks=128 obj#=9793 tim=1184537617377531
WAIT #2: nam='db file scattered read' ela= 7869 file#=4 block#=667 blocks=128 obj#=9793 tim=1184537617539113
WAIT #2: nam='db file scattered read' ela= 7847 file#=4 block#=795 blocks=128 obj#=9793 tim=1184537617700272
```

--
Regards,

Greg Rahn
http://structureddata.org

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 8, 2008 8:09 PM in response to:

Reply

> >> you're supposed to design a theory to match the
> facts, not select the facts to match the theory.
>
> I think it's the other way around, Jonathan, the
> scientific method requires that you start with a
> hypothesis.
>

Thank-you. At last, it all finally makes sense, you select facts to match your theories. Got it.

It finally explains why after I and many others show you facts that actually contradict one of your theories, they simply get ignored. You only ever seem to take note of those facts which match your theories.

Seriously, thank-you, all is now crystal clear.

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 8, 2008 8:28 PM in response to: Richard Foote

Reply

I imaging this thread is very similar to a conversation centuries ago between Galileo and the authorities.

Time to bookmark the thread for future reference - both for the reasonably careful data set produced by Charles (and discussion thereof), and your observation.

Message was edited by: Hans Forbrich

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 8, 2008 9:26 PM in response to: Hans Forbrich

Reply

For anyone interested in the academic description of the scientific method:

http://teacher.pas.rochester.edu/phy_labs/AppendixE/AppendixE.html

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 8, 2008 9:37 PM in response to:

Reply

> - **David Aldridge** notes a test where is noted a
> 6% reduction with larger index blocksizes, a
> significant difference, especially to larger shops :
>
> "there are multiple stages in deciding whether the
> larger block size is beneficial to a system ...
>
> Working out what low level operations benefit from it
> (multi-block reads, single block reads)
>
> Identifying what higher-level access methods make use
> of these operations
>
> Applying this to the type of object (table/index) and
> system type (reporting/OLTP)"
>
> ---------------------------------------

For the record, that quote came from a forum topic in which I profoundly disagreed with your multiple blocksize theory, and I still do.

http://dba.ipbhost.com/index.php?showtopic=1239&st=15

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 8, 2008 10:34 PM in response to: damorgan

Reply

> For anyone interested in the academic description of
> the scientific method:
>
> http://teacher.pas.rochester.edu/phy_labs/AppendixE/Ap
> pendixE.html

Mr. Damorgan,

I don't think you are qualified to make any comments on this tread. Let me quote your initial challenge again,

>>That the query is faster is not being questions. What is at issue is that you are
>>drawing an unsupported inference.

>>The point I think Jonathan is making is that your test case does not prove what you
>>are claiming it does. 16x32 <> 8x8. You have no evidence that the relevant factor was
>>the block size and not the change in multi-block reads or any one of a number of >>other possible factors.

>>The lab test should look like this:
>>Test 1: Run test using 8K blocks.
>>Test 2: Run the exact same test changing NOTHING other than the block size.

Being an Ace Director and hide your face in a hole and utter nonsense is not the quality of a professional, at least as i expected from you. As like many other members in this thread indicated, why don't you publish Lab results, if you have any, instead of trailing troll with your pals.
I was also in teaching industry, for more than 10 years before my current jobs in corporate worlds. I am sorry to say harsh words, but i think you deserve that.

sp009

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 8, 2008 11:14 PM in response to: sp009

Reply

DAM's point seems to me to be a fair one. The size of the multiblock read is independent of the block size and the effects of changing them ought to be tested independently of each other. It's inescapable that a procedural error was made in using different multiblock read sizes in the two test cases, and that any results would be questionable.

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 9, 2008 12:53 AM in response to: Greg Rahn

Reply

In the previous experiments I did, all of the I/O was physical: no blocks existed in the buffer cache prior to execution. I thought it would useful to consider the case if all of the data is in the buffer cache (no physical reads), so I ran another set of experiments. In each scenario (FTS and Hash Join) the elapsed times are statistically equivalent (close enough to call equal).

Oracle 10.2.0.3
Linux 32-bit
ASM Storage

8k db and table: FTS From Buffer Cache

```
********************************************************************************

select * from WEB_RETURNS_8K

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch    71978     33.47      56.00          0     173743          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71980     33.47      56.00          0     173743          0     7197670

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS_8K (cr=173743 pr=0 pw=0 time=14399933 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71981        0.00          0.09
  SQL*Net message from client                 71981        0.00         66.89
  SQL*Net more data to client                359883        0.00         25.26

********************************************************************************
```

16k db and table: FTS From Buffer Cache

```
********************************************************************************

select * from WEB_RETURNS_16K

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch    71978     33.84      56.95          0     122225          0     7197670
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total    71980     33.84      56.95          0     122225          0     7197670

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
7197670  TABLE ACCESS FULL WEB_RETURNS_16K (cr=122225 pr=0 pw=0 time=14400007 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                   71981        0.00          0.09
  SQL*Net message from client                 71981        0.00         67.36
  SQL*Net more data to client                359883        0.00         26.09

********************************************************************************
```

8k db and table: Hash Join From Buffer Cache

```
********************************************************************************

select count(*)
from WEB_RETURNS_8K a, WEB_RETURNS_8KB b
where a.WR_ORDER_NUMBER = b.WR_ORDER_NUMBER

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     23.84      23.96      10447     205580          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     23.84      23.96      10447     205580          0           1

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=205580 pr=10447 pw=10447 time=23961252 us)
15516562   HASH JOIN  (cr=205580 pr=10447 pw=10447 time=41738443 us)
7197670    TABLE ACCESS FULL WEB_RETURNS_8K (cr=102790 pr=0 pw=0 time=7197880 us)
7197670    TABLE ACCESS FULL WEB_RETURNS_8KB (cr=102790 pr=0 pw=0 time=7197860 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       5        0.00          0.00
  SQL*Net message from client                     5        0.00          0.00
  direct path write temp                        337        0.03          0.47
  direct path read temp                         337        0.01          0.07

********************************************************************************
```

16k db and table: Hash Join From Buffer Cache

```
********************************************************************************

select count(*)
from WEB_RETURNS_16K a, WEB_RETURNS_16KB b
where a.WR_ORDER_NUMBER = b.WR_ORDER_NUMBER

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2     23.42      23.69       5055     101474          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4     23.42      23.69       5055     101474          0           1

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=101474 pr=5055 pw=5055 time=23699400 us)
15516562   HASH JOIN  (cr=101474 pr=5055 pw=5055 time=41726195 us)
7197670    TABLE ACCESS FULL WEB_RETURNS_16K (cr=50737 pr=0 pw=0 time=7197866 us)
7197670    TABLE ACCESS FULL WEB_RETURNS_16KB (cr=50737 pr=0 pw=0 time=7197843 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       5        0.00          0.00
  SQL*Net message from client                     5        0.00          0.00
  direct path write temp                        337        0.00          0.43
  direct path read temp                         337        0.02          0.33

********************************************************************************
```

--
Regards,

Greg Rahn
http://structureddata.org

Niall Litchfield

Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 9, 2008 1:52 AM in response to: David Aldridge

Reply

> DAM's point seems to me to be a fair one. The size of
> the multiblock read is independent of the block size
> and the effects of changing them ought to be tested
> independently of each other. It's inescapable that a
> procedural error was made in using different
> multiblock read sizes in the two test cases, and that
> any results would be questionable.

Hi David,

I think I disagree - though only in the sense that we are now discussing a publicly available and repeatable test and refining our theories :).

Specifically I disagree that setting MBRC is 'independent' of the setting of the block size of the database. They certainly are independent variables you can set them both separately, but I'd argue that they were related variables (both together go towards determining how much data is attempted to be read in a single read). Historically of course setting MBRC also had a big impact on the costing of access paths, so you end up with more and messier factors to consider. It's also why Charles (I think it was) found that MBRC was changed by default on different blocksize databases in more recent versions.

If I were to do tests as per Charles and Greg (excellent work by both by the way - butr I would say that wouldn't I) I think the tests on the different blocksizes should be accompanied by another 2 axis of variability - setting MBRC so as to make the data transfer attempted in a single read the same (and matching the hardware) or not and having system statistics set or not. (then we'd get drowned in results in a forum thread - maybe yet another whitepaper should be written)

Niall Litchfield
http://www.orawin.info/

Niall Litchfield

Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 9, 2008 1:59 AM in response to: damorgan

Reply

> For anyone interested in the academic description of
> the scientific method:
>
> http://teacher.pas.rochester.edu/phy_labs/AppendixE/Ap
> pendixE.html

As well as the description of the process I also like the quote at the top describing the reason for the process

The scientific method is the process by which scientists, collectively and over time, endeavor to construct an accurate (that is, **reliable, consistent and non-arbitrary**) representation of the world.

It reminded me of Feynman in Cargo Cult Science

But this long history of learning how not to fool ourselves—of having utter scientific integrity—is, I'm sorry to say, something that we haven't specifically included in any particular course that I know of. We just hope you've caught on by osmosis.
The first principle is that you must not fool yourself—and you are the easiest person to fool. So you have to be very careful about that. After you've not fooled yourself, it's easy not to fool other scientists. You just have to be honest in a conventional way after that.

I hope it's obvious to my readers which side of the argument (tests, discussion and refinement vs appeal to industry recognised experts with various business interests) I think is better characterised by the Feynman integrity described above.

Niall

---

cd

Posts: 4,585
From: Vienna, Austria
Registered: 9/8/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 6:00 AM in response to: Niall Litchfield

Reply

Thing is, DKB introduced "science" by accusing others of using "unscientific tricks" in this thread. Now people like me would now assume that he'd show us some scientific approaches to counter that development, but all I'll see is some useless references to threads, hidden production databases and a CVs that doesn't even show the slightest reference to a scientific career.

For me, threads like this, with test cases that could be used to test drive my own configuration, if I ever have to, are invaluable, and I can live with the noise generated by one self-proclaimed leading Oracle Expert/DBA.

Thanks to all the others that took their time to show and explain those results, you guys rock.

C.

Niall Litchfield

Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 9, 2008 6:03 AM in response to:

Reply

> Hi Niall,
>
> >> A rather long time ago it was discovered that
> perhaps actually there were definable factors at work
> in what made a great bordeaux and that in fact, just
> maybe, understanding what affected the wine and how
> was a better bet than trusting the judgement of the
> human intuition of the self-appointed experts
>
> EXCELLENT example! This sure sounds familiar:
>
> *"Robert M. Parker Jr., generally regarded as the*
> *most influential wine critic in America, calls*
> *Professor Ashenfelter's research ''ludicrous and*
> *absurd.''*

It does indeed sound familiar doesn't it - the argument from authority and reputation vs the argument from analysis. Possibly it's somewhat unfortunate for your case then that Ashenfeltzer's predictions were more reliable than Robert Parker's.

Niall Litchfield
http://www.orawin.info/

Re: Larger vs. Small data block
Posted: Jun 9, 2008 7:53 AM in response to: David Aldridge

Reply

Re: Larger vs. Small data block
Posted: Jun 9, 2008 8:02 AM in response to: Niall Litchfield

Reply

Re: Larger vs. Small data block
Posted: Jun 9, 2008 8:10 AM in response to: Niall Litchfield

Reply

Re: Larger vs. Small data block
Posted: Jun 9, 2008 8:16 AM in response to: cd
Reply

David Aldridge
Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 8:47 AM in response to: Niall Litchfield
Reply

> I think I disagree - though only in the sense that we
> are now discussing a publicly available and
> repeatable test and refining our theories :).

I should have been more clear -- I mean the multiblock read size in terms of KB/read, rather than the init paramemter. Particularly as in 10g the advice apears to be to not set it.

I believe that multiblock read size should always be set to the maximum available. It's regrettable that it has historically been set in terms of a block count instead of bytes, especially in the presence of multiple block size in a single database where you have to be careful to set the MBRC (if you set it at all) in the context of the database default block size.

Anyhoo, I don't think we're disagreeing.

cd
Posts: 4,585
From: Vienna, Austria
Registered: 9/8/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:04 AM in response to:
Reply

> Yes, but note that I'm the only "expert" in this
> thread who is forthright enough to publish my
> credientials!

I'd rather see you publish some test cases that could be verified - but that's too much to ask, isn't it?

> Me, I don't pretend to be an "Oracle Scientist", and
> like I said before, I think that some "self annointed
> experts" are perpetuating myths by appearing to have
> a background in science, when in reality, they are
> completely unqualified to make that claim . . .

I consider myself a software developer with some technical background. I **need** verifyable test cases and concepts in order to accomplish my work. Maybe you are one of the leading Oracle DBAs on this planet, but as long as you refuse to contribute such things and simply ask others to accept your "solutions" in good faith, I'll stick to those people you keep on attacking. So please, talk to the hand, because this developer won't listen to you anymore.

C.

Billy Verreynne
Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:07 AM in response to:
Reply

>Out of all of the "experts" in this thread (Morgan, Lewis, Rahn, &c), how many have known credientials?
> None, but me. . . . .
> Why is that?

Because so-called "credentials" that at best are extremely time consuming and difficult to verify, mean absolutely *nothing* here and on most forums on the Internet.

What *does* matter.. and what you blatantly do not get (or refuse to get?) is that the CONTENT of a posting is what serve as the credentials of the posting.

In other words, references to official documentation, test case that can be read, understood and duplicated on the platform of your choice.. stuff like that determines whether the vast majority of forum members accept that posting as credible or not.

And not on your claims of how much of an expert you are because of where you've studied, the degrees (relevant or not) you may have, the type of suit and shoes you wear when consulting, and your claims of how many evil performance dragons you have slain using magical silver bullets, casted in your very own superhero Oracle lair for super-experts.

David Aldridge
Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:12 AM in response to:
Reply

> Hi David,
>
> >> I profoundly disagreed with your multiple
> blocksize theory
>
> It's not MY theory,I first learned it fro Oracle
> University in the early 1990's, it's been around
> quite some time. BTW, it was presented as fact by
> OU, not theory.
>

Really? I thought transportable tablespaces were an Oracle 8i feature, introduced in 1999. Or am I wrong?

> You know, OU has details in the official courseware,
> telling students how to choose the "best" blocksize
> for their database. It would be intersting to see
> what it says.

That's a different matter from using multiple block sizes in the same database.

>
> David, you yourself have noted differences in
> performance. Are you arguing that these are not big
> enough differences, or that the differences don't
> exist?
>
I've not noted very big differences, if any. I generally use large block sizes to reduce space wastage when dealing with very long average row lengths (more of a data mart thing than a data warehouse thing), and since you never know whether someone is going to come along with a requirement in the future that will lead to a very long average row length then I'll start off with

a high block size initially.

> Just curious, do you run your warehouses on an 8k
> blocksize?
>
16kb, usually -- I got bitten by a bug on 32kb blocks a few years ago and that makes me wary of going there again. There's almost no difference between 16kb and 32kb sizes in space saving anyway. I have used multiple block sizes for the purpose of transporting tablespaces from OLTP systems.

> ***
> ***********************************
>
> David, please note that the differences in
> performance with different blocksizes is presented on
> MetaLink, not as theory, but as fact:
>
> Metalink Note:46757.1 titled "Notes on Choosing an
> Optimal DB BLOCK SIZE"
>
> - Large blocks gives more data transfer per I/O
> call.
>

Only if you're transfering single blocks. The overwhelming majority of data warehouse reads are multiblock direct path due to parallel query, in my experience. Block size is not relevant to performance there, really.

> Larger blocksizes provides less fragmentation (row
> chaining and row migration) of large objects (LOB,
> BLOB, CLOB)
>
I've never worked with LOB's in a data warehouse. Can't imagine a case for them. I've worked with BLOBs on OLTP systems but they should have been VARCHAR2s as it happens. They weren't big enough to justify a BLOB.

> - Indexes like big blocks because index height can be
> lower and more space exists within the index branch
> nodes.
>

They don't enjoy the contention on simultaneous modification though.

> Moving indexes to a larger blocksize saves disk
> space. Oracle says "you will conserve about 4% of
> data storage (4GB on every 100GB) for every large
> index in your database by moving from a 2KB database
> block size to an 8KB database block size."
>

8kb is pretty standard stuff nowadays.

> Metalink goes on to say that multiple blocksizes may
> benefit shops that have "mixed" block size
> requirements:
>
> - What can you do if you have mixed requirements of
> the above block sizes?
>
> - Oracle9i "Multiple Block Sizes" new feature comes
> into the rescue here, it allows the same database to
> have multiple block sizes at the same time .

It's 9i that allows you to have multiple block sizes then? That was 2001, not the early 1990's

>
> ***
> **
>
> In the IOUG 2005 conference proceeding titled "OMBDB:
> An Innovative Paradigm for Data Warehousing
> Architectures", Anthony D. Noriega notes evidence
> that his databases benefited greatly from employing
> multiple blocksizes and notes that multiple
> blocksizes are commonly used in large databases with
> limited RAM resources, in applications such as
> marketing, advertisement, finance, pharmaceutical,
> document management, manufacturing, inventory
> control, and entertainment industry:
>
> http://noriegaaoracleexpert.blogspot.com/2007/08/advan
> ces-in-multiple-block-size-caches.html
>
> *"The paper and presentation will discuss how to*
> *best utilize multiple block size databases in*
> *conjunction with table partitioning and related*
> *techniques, . . .*
>
> *Utilizing Oracle multiblock databases in data*
> *warehousing based systems will prove in the long-term*
> *to be a reliable methodology to approach the*
> *diversity of information and related business*
> *intelligence applications processes when integrating*
> *existing systems, consolidating older systems with*
> *existing or newly created ones, to avoid redundancy*
> *and lower costs of operations, among other factors.*
>
> *The input received from those already using*
> *multiblock databases in highly satisfactory in areas*
> *such as marketing, advertisement, finance,*
> *pharmaceutical, document management, manufacturing,*
> *inventory control, and entertainment industry."*

I read the blog entry. No rationale is presented there at all, and there's not a single measurement presented. Just unnamed sources. I think that if someone is going to present an idea as an "innovative paradigm" then one ought at least to have something more to back it up with. If there is more then I'll gladly read it and provide my comments.

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:14 AM in response to: cd

Reply

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:20 AM in response to: David Aldridge

Reply

mpowel01

Posts: 2,840
Registered: 12/8/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:22 AM in response to:

Reply

>> I don't pretend to be an "Oracle Scientist" <<

If I remember right the scientific method was taught in my elementary school science classes. It went something like the following:

1 - Come up with a theory (hypothesis)
2 - Device an experiment to test the hypothesis
3 - Run the test
4 - analyze the results

Additional tests may be necessary based on the test results and the findings from the analysis

Some of the requirements related to the test were

The test results had to be repeatable

You had to identify as many of the variables in the test as possible and freeze them so you could properly identify the effect due of changes to a single variable

The tests were designed not to prove a certain point but rather to produce results that would result only if the hypothesis were true. To design a test to produce specific results was rigging the test.

Every high school graduate should recognize the validity of this approach.

Observation on the other hand is not that reliable. It is subject to the bias of the observer and what is observed can easily be attributed to the wrong factor.

There is also a proper way to challenge and respond to a challenge of the test design and analysis.

Mark D Powell MS (Org and Mngt), CPIM, CIRM, OCP for 8.0, 8i, 9i, & 10g

cd

Posts: 4,585
From: Vienna, Austria
Registered: 9/8/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:24 AM in response to:

Reply

> Really? You are not capable of testing concepts in
> your own?

That's what those mentioned test cases are for. It's called an efficient approach.

> There you go again. Myopia and zealotry, towards a
> single approach.

In your case, this is Vodoo-IT.

> Me, I listen closely when someone with unimpeachable
> credientials speaks from experience . . .

Sure, next time I need someone with a BA in Psychology ...

> You should keep an open mind, CD, you miss out on a
> lot. . . .

Don't worry, I will. It's just you who I'm going to ignore in the future.

C.

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:29 AM in response to: mpowel01

Reply

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:31 AM in response to: cd

Reply

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:32 AM in response to:

Reply

> Hi David,
>
> >> That's a different matter from using multiple
> block sizes in the same database.
>
> Oh, you just got here, sorry.
>
> This is a discussion of the benefits of different
> blocksizes.
>
> We agreed to defer discussion of multiple blocksizes
> until we hit the 20th page!!!
>
Huh, I guess that neither of us got that memo then. ;)

>
> David, do you agree that, all else being equal, small

> rows in a large blocksize can perform worse than
> large rows in a small blocksize under heavy DML
> load?
>
Of course they can -- and the reverse is equally true. It depends on the nature of the DML load. You have to consider contention, the physical ordering if any of the data, the method of modification of the data, etc.. That's why there's no "one-size-fits-all" solution.

> **
> **********************
>
> Oh, David, you forgot to answer my question!
>
> Does your production warehouse use an 8k blocksize?
> *Be Sirius now!*

It uses 16kb. If it used 8kb we'd be marginally less efficient in storage space utilization. It wuldn't worry me too much if it was though -- a few percent here and there is nothing.

cd

Posts: 4,585
From: Vienna, Austria
Registered: 9/8/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 9:39 AM in response to:

Reply

> Why is that?

Quite simple: I'm not one of the world's leading Oracle DBAs ...

C.

mpowel01

Posts: 2,840
Registered: 12/8/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 10:47 AM in response to:

Reply

First of all a repeatable results do not have to match exactly. There are such a thing as random factors. If you perform 10,000 IO's and measure every one to the nearest 10 thousand of a second the timings of the IO's are probably going to vary some. But if you run the exact same test that performs thousands of IO's on the same system multiple times the final results should fall within a narrow distribution. Statistics can be used to properly categorize results.

-- Mark D Powell --

Niall Litchfield

Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 9, 2008 11:19 AM in response to: mpowel01

Reply

> >> I don't pretend to be an "Oracle Scientist" <<
>
> If I remember right the scientific method was taught
> in my elementary school science classes. It went
> something like the following:
>
> 1 - Come up with a theory (hypothesis)
> 2 - Device an experiment to test the hypothesis
> 3 - Run the test
> 4 - analyze the results

Hi Mark,

The experiment had to make predictions and you missed out step 5 (I think I did as well to be fair) which is of course modify the theory to fit the observed facts, rinse and repeat. I'm glad that the scientific method is taught in elementary school in the U.S though, I was beginning to wonder what with all the insistence on recognised qualifications and all.

Niall

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 9, 2008 11:50 AM in response to: Jonathan Lewis

Reply

> > Thanks for having a look in to that. I didn't
> > convince my self with your answer. Never mind.
>
> Fair enough - but at least we've had a discussion
> which has highliighted the importance of
> constructuing experiments to test a hypothesis, and
> given other people the chance to see how careful you
> have to be to design the test properly/
>
> > I wish i can show the tkprof of some of the long
> run
> > queries in my production and test database
> > (identical server, windows 2003/64 with 16k and 8k
> > block size and data nearly same).
> > But the policy doesn't allow me to do that.
>
> I've never been convinced that this makes it
> impossible to share performance data without
> compromising business intelligence. After all, if
> you want to examine the I/O pattern for a query you
> can cut one statement out of a tkprof file, delete
> the SQL, and change the names of the tables and
> indexes in the rowsource output in a consistent
> fashion.
>
> You might be able so show an example of that sort of
> thing to your governance officer and get clearance to
> show it on the forum.
>
> Regards
> Jonathan Lewis
> http://jonathanlewis.wordpress.com
> http://www.jlcomp.demon.co.uk

Just finished the analysis of tkprof of a job scheduled on week-end, which process 30m rows, in production (16k db_block_size) and test (8k db_block_size) databases installed in identical Server Win 2003/64b ASM RAID. Before the job run, i refreshed the data in test so that both database will have same volume. Guess what, there is 18% difference in response time and the cpu utilization between the production and test database. My supervisor discussed the End-result with our consultant DBA (From a world famous Consultancy Group (Oracle???), and is labeled as performance Guru!). End result? i am expecting a pay raise pretty soon and our consultant DBA owes me a lunch at red lobster. I don't see any point in cut & paste the tkprof result in the forum. Lab experts may have hundreds of excuses for this performance difference. Also our consultant DBA promised to publish some article in Oracle Magazine regarding the benefits of higher block size in Warehouse application very soon.

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 9, 2008 11:54 AM in response to: Jonathan Lewis

Reply

Jonathan,

Thanks for the response. I executed the set of scripts 3 more times:
* 8KB block size in locally managed 1MB uniform extent size
* 8KB block size in locally managed 1MB uniform extent size, hyper-threading disabled (it was enabled in all other tests)
* 16KB block size in locally managed 1MB uniform extent size

I haven't examined the results too closely yet, but what appears to be happening:
* Execution times for the 8KB block size locally managed 1MB uniform extent size increased over the times for 8KB block size in ASSM
* Full index scan for the 8KB block size locally managed 1MB uniform extent size seems to have only used single block reads, where it started with cycles of single block read followed by a 7 block read in the ASSM run.
* With hyperthreading disabled, the execution times increased.

> In the most recent post you've labelled the tests 7
> and 11 - but I think from a couple of posts back they
> were 8 and 11. (In either case we're talking about
> the 'select distinct' that does an index full scan
> with a "sort unique nosort").

The test numbering is a bit confusing:
Test 1: 16KB block size, setting up the tables and initial performance tests.
Test 2: 16KB block size, DBMS_XPLAN with statistics level set to ALL at the session level with 10046 and 10053 traces.
Test 3: 16KB block size, DBMS_STATS, simple select.
Test 4: 8KB block size, setting up the tables and initial performance tests.
Test 5: 8KB block size, DBMS_XPLAN with statistics level set to ALL at the session level with 10046 and 10053 traces.
Test 6: 8KB block size, DBMS_STATS, simple select.
Test 7: 8KB block size files in same location as original 16KB database, setting up the tables and initial performance tests.
Test 8: 8KB block size files in same location as original 16KB database, DBMS_XPLAN with statistics level set to ALL at the session level with 10046 and 10053 traces.
Test 9: 8KB block size files in same location as original 16KB database, DBMS_STATS, simple select.
Test 10: 16KB block size files in same location as original 16KB database, setting up the tables and initial performance tests.
Test 11: 16KB block size files in same location as original 16KB database, DBMS_XPLAN with statistics level set to ALL at the session level with 10046 and 10053 traces.
Test 12: 16KB block size files in same location as original 16KB database, DBMS_STATS, simple select.

> There is an oddity with the results, though. The
> tkprof summaries show no "db file scattered reads",
> but the trace outputs do show scattered reads - is
> this from repeating the test ?

None of the tests were repeated, except that tests 2, 5, 8, and 11 were performed without bringing down the database, and those tests repeated a SQL statement from the previous test number.

> The parameter _db_file_noncontig_mblock_read_count is
> supposed to limit the number of blocks in a single
> "db file parallel read", and there are a couple of
> related parameters (_ncmb_readahead_enabled,
> _ncmb_readahead_tracing) that are supposed to enable
> it and allow tracing. The default for the limit is
> 11 blocks - which could allow a very large index scan
> to operate more efficiently in a tablespace with a
> larger block size - but I have no idea what might
> happen when a paralllel read 'collapses' to a
> scattered read - maybe the 11 limit still applies,
> rather than the db_file_multiblock_read_count limit.

Incidentally, I used a script from your website to capture all hidden database parameters at the end of tests 3 and 6. If you are interested, I will report what is found in those captured parameters.

> The timings are quite revealing - I think it's safe
> to assume that a reported time for a read that falls
> in the region of 350 microseconds isn't a disk read,
> but a memory fetch from a cache somewhere. So the
> "slow, quick quick quick ..." pattern may be giving
> us a clue about an asynchronous readahead mechanism.

Each of the two drives in the RAID 0 array has, I believe, an 8MB built-in cache. The drives also support command queuing, meaning that the drives should be able to batch together some read requests for adjacent areas of the disk. I don't know if either of these are affecting the read times.

> If you see odd patterns of this scattered read effect
> switching itself on and off, that's because CKPT
> controls the feature, and decides (every three
> seconds, I think) whether or not Oracle's "index
> prefetch" mechanism should be used.
>
> There's always more to think of when the results show
> large deviations from expected behaviour - the
> possible interference of pre-fetch and caching makes
> me wonder how much CPU time was consumed outside
> Oracle when you were doing the different tests.

That is a good question that I can't answer - I tried to minimize the outside influences of other programs consuming CPU time. The server and client were both on the same computer. I did notice long elapsed parse times in the last 8KB set of tests that I posted, when compared to the last 16KB set of tests that I posted.

Charles Hooper
IT Manager/Oracle DBA

K&M Machine-Fabricating, Inc.

Re: Larger vs. Small data block
Posted: Jun 9, 2008 1:43 PM in response to: sp009 Reply

Billy Verreynne
Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 9, 2008 3:22 PM in response to: Reply

> We need the voice of real-world experience here . .

Ah yes.. because in the real world bytes are royal blue. And as we all know, "contrived" test cases use test data and those bytes are a measly yellow.

And I/O on royal blue bytes are very different from I/O on measly yellow bytes. Which means that any test cases that clearly show Oracle's behaviour, are not applicable as I/O on royal blue bytes are different... because says so.

<in the background sp009 is holding up a " 3:16" sign>

<insert picture here of me grabbing my coat>

David Aldridge
Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 3:39 PM in response to: sp009 Reply

> Just finished the analysis of tkprof of a job
> scheduled on week-end, which process 30m rows,
> in production (16k db_block_size) and test (8k
> db_block_size) databases installed in identical
> Server Win 2003/64b ASM RAID. Before the job run, i
> refreshed the data in test so that both
> database will have same volume. Guess what, there is
> 18% difference in response time and
> the cpu utilization between the production and test
> database.

So just to be clear, the production server with higher block size showed an 18% lower cpu load and was 18% faster than the test system that had the lower block size? That's interesting -- with the two percentages being the same it implies to me that CPU is the predominant load on the servers, and that io wait is relatively very low, is that the case? Or were there off-setting differences in other wait events (eg higher read time and lower write time)?

Also, what sort of load is this .. a batch job? Or regular OLTP operations?

Message was edited by: DA. Typo, changed "tow" to "two"
David Aldridge

Jonathan Lewis
Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 9, 2008 4:01 PM in response to: sp009 Reply

> Just finished the analysis of tkprof of a job
> scheduled on week-end, which process 30m rows,
> in production (16k db_block_size) and test (8k
> db_block_size) databases installed in identical
> Server Win 2003/64b ASM RAID. Before the job run, i
> refreshed the data in test so that both
> database will have same volume. Guess what, there is
> 18% difference in response time and
> the cpu utilization between the production and test
> database. My supervisor discussed the
> End-result with our consultant DBA (From a world
> famous Consultancy Group (Oracle???),
> and is labeled as performance Guru!). End result? i
> am expecting a pay raise pretty soon and
> our consultant DBA owes me a lunch at red lobster. I
> don't see any point in cut & paste the tkprof
> result in the forum. Lab experts may have hundreds of
> excuses for this performance difference.

So, if I understand you correctly - you've analysed the tkprof results for a major job, but can't be bothered to make any comments about anything you saw that could have been the cause of an 18% performance improvement.

Your boss is going to give you a pay rise because you exported a data warehouse from a database on a new server into a database on an older server and said that a batch job ran 18% slower ? 18% shouldn't be too difficult all you have to do is lose the odd index and you could make it MUCH slower.

Go on, just one little tkprof extract from each database that shows a meaningful performance improvement without a change in execution plan. Surely it won't lose you your red lobster lunch, even if someone why there was a difference.

> Also our consultant DBA promised to publish some
> article in Oracle Magazine regarding the
> benefits of higher block size in Warehouse
> application very soon.

Please post a note on this thread when it happens - I'll be interested to see what he says.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Charles Hooper
Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 9, 2008 4:14 PM in response to: Charles Hooper Reply

> I executed the set of scripts 3 more times:
> * 8KB block size in locally managed 1MB uniform extent size
> * 8KB block size in locally managed 1MB uniform extent size, hyper-threading disabled (it was enabled in all other tests)

> * 16KB block size in locally managed 1MB uniform extent size

* Tests 13, 14, 15 - 8KB block size in locally managed 1MB uniform extent size
* Tests 16, 17, 18 - 8KB block size in locally managed 1MB uniform extent size, no HT
* Tests 19, 20, 21 - 16KB block size in locally managed 1MB uniform extent size

I am considering running a new script against 8KB and 16KB databases that repeatedly updates table rows and related indexes to determine if the database block size makes a difference in this test setup.

8KB UNIFORM 1MB
#TEST RUN 13 8KB UNIFORM 1MB

```
  COUNT(*)
----------
11073

Elapsed: 00:00:00.70

Execution Plan...

Statistics
----------------------------------------------------------
        641  recursive calls
          0  db block gets
      19569  consistent gets
        377  physical reads
         72  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         25  sorts (memory)
          0  sorts (disk)
          1  rows processed


Table created.

Elapsed: 00:02:02.09

Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:05.25

System altered.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:09:04.06

Table created.

Elapsed: 00:00:00.71

Index created.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:01.70

System altered.

Elapsed: 00:00:00.01

1000000 rows created.

Elapsed: 00:02:00.79

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |      |  7179 |   988K|   178K  (1)| 00:35:43 |
|*  1 |  TABLE ACCESS FULL| T1   |  7179 |   988K|   178K  (1)| 00:35:43 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RN"<=100)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       8382  recursive calls
    2855795  db block gets
     713983  consistent gets
     651640  physical reads
  470276500  redo size
        682  bytes sent via SQL*Net to client
        583  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
    1000000  rows processed
```

```
Commit complete.

Elapsed: 00:00:00.01

System altered.

Elapsed: 00:00:15.53

System altered.

Elapsed: 00:00:00.03

Session altered.

Elapsed: 00:00:00.01

no rows selected

Elapsed: 00:01:09.56

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |  7179 |   988K|   178K  (1)| 00:35:40 |
|*  1 |  TABLE ACCESS FULL| T1   |  7179 |   988K|   178K  (1)| 00:35:40 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("STATUS"='NONE')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          6  recursive calls
          0  db block gets
     652567  consistent gets
     651480  physical reads
          0  redo size
       1047  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


  COUNT(*)
----------
1000000

Elapsed: 00:00:02.50

Execution Plan
----------------------------------------------------------
Plan hash value: 1385691034

-------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |     1 |  1864   (1)| 00:00:23 |
|   1 |  SORT AGGREGATE       |         |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| T2_IND1 |   857K|  1864   (1)| 00:00:23 |
-------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         32  recursive calls
          3  db block gets
      14179  consistent gets
       8036  physical reads
     507292  redo size
        411  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


OWNER                          OBJECT_NAME
------------------------------ ------------------------------
SUBOBJECT_NAME
------------------------------

9454 rows selected.

Elapsed: 00:01:46.12

Execution Plan
----------------------------------------------------------
Plan hash value: 1118578911

------------------------------------------------------------------------------
```

```
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |    41M|  2030M|   808K  (1)| 02:41:48 |
|   1 |  SORT UNIQUE NOSORT|         |    41M|  2030M|   808K  (1)| 02:41:48 |
|   2 |   INDEX FULL SCAN  | T1_IND1 |    41M|  2030M|   276K  (1)| 00:55:23 |
-----------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          6  recursive calls
          0  db block gets
     275219  consistent gets
     274154  physical reads
          0  redo size
     299156  bytes sent via SQL*Net to client
       7311  bytes received via SQL*Net from client
        632  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9454  rows processed


Session altered.

Elapsed: 00:00:00.00
```

#TEST RUN 14 8KB UNIFORM 1MB

```
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1;

-------------------------------------------------------------------------------------------------
| Id  | Operation          | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------
|   1 |  SORT UNIQUE NOSORT|         |      1 |     41M|   9454 |00:02:36.66 |     274K|    274K|
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     41M|     50M|00:01:40.05 |     274K|    274K|
-------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement
```

#TEST RUN 15 8KB UNIFORM 1MB

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:30.14

PL/SQL procedure successfully completed.

Elapsed: 00:02:11.53

System altered.

Elapsed: 00:00:00.06

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.03

no rows selected

Elapsed: 00:01:11.37

Execution Plan
----------------------------------------------------------
Plan hash value: 2134347679

--------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    32 |   178K  (1)| 00:35:42 |
|   1 |  HASH UNIQUE       |      |     1 |    32 |   178K  (1)| 00:35:42 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |    32 |   178K  (1)| 00:35:42 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("STATUS"='NONE')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
     651991  consistent gets
     651480  physical reads
          0  redo size
        399  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

```
Session altered.

Elapsed: 00:00:00.00

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
T1                               49640731     652598          88
T2


INDEX_NAME                         BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------------------------ ---------- ----------- ------------- ----------------------- ----------------------- ----------
-------
T1_IND1                                 3      273198      46842892                       1                       1
48002785
T2_IND1
```

8KB UNIFORM 1MB NO HYPER-THREADING
#TEST RUN 16 8KB UNIFORM 1MB NO HT

```
  COUNT(*)
----------
11073

Elapsed: 00:00:00.68

Execution Plan...

Statistics
----------------------------------------------------------
       1022  recursive calls
          0  db block gets
      19639  consistent gets
        382  physical reads
        116  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         38  sorts (memory)
          0  sorts (disk)
          1  rows processed


Table created.

Elapsed: 00:02:00.46

Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:04.85

System altered.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:09:12.43

Table created.

Elapsed: 00:00:00.67

Index created.

Elapsed: 00:00:00.01

System altered.

Elapsed: 00:00:01.73

System altered.

Elapsed: 00:00:00.01

1000000 rows created.

Elapsed: 00:02:04.07

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |      |  7179 |   988K|   178K  (1)| 00:35:37 |
|*  1 |  TABLE ACCESS FULL| T1   |  7179 |   988K|   178K  (1)| 00:35:37 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RN"<=100)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       8426  recursive calls
    2856404  db block gets
```

```
     713868  consistent gets
     651640  physical reads
  470073780  redo size
        682  bytes sent via SQL*Net to client
        583  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
    1000000  rows processed


Commit complete.

Elapsed: 00:00:00.01

System altered.

Elapsed: 00:00:16.01

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.03

no rows selected

Elapsed: 00:01:15.50

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |  7179 |   988K|   177K  (1)| 00:35:35 |
|*  1 |  TABLE ACCESS FULL| T1   |  7179 |   988K|   177K  (1)| 00:35:35 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("STATUS"='NONE')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     652567  consistent gets
     651480  physical reads
          0  redo size
       1047  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


  COUNT(*)
----------
1000000

Elapsed: 00:00:02.40

Execution Plan
----------------------------------------------------------
Plan hash value: 1385691034

-------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |     1 |  1863   (1)| 00:00:23 |
|   1 |  SORT AGGREGATE       |         |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| T2_IND1 |   858K|  1863   (1)| 00:00:23 |
-------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         32  recursive calls
          3  db block gets
      14163  consistent gets
       7907  physical reads
     506172  redo size
        411  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


OWNER                          OBJECT_NAME
------------------------------ ------------------------------
SUBOBJECT_NAME
------------------------------
```

```
9454 rows selected.

Elapsed: 00:01:42.03

Execution Plan
----------------------------------------------------------
Plan hash value: 1118578911

------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |    41M|  2026M|   806K  (1)| 02:41:22 |
|   1 |  SORT UNIQUE NOSORT|         |    41M|  2026M|   806K  (1)| 02:41:22 |
|   2 |   INDEX FULL SCAN  | T1_IND1 |    41M|  2026M|   276K  (1)| 00:55:21 |
------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     275255  consistent gets
     274185  physical reads
          0  redo size
     299135  bytes sent via SQL*Net to client
       7311  bytes received via SQL*Net from client
        632  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9454  rows processed


Session altered.

Elapsed: 00:00:00.00
```

#TEST RUN 17 8KB UNIFORM 1MB NO HT

```
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1;

-------------------------------------------------------------------------------------------------
| Id  | Operation          | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------
|   1 |  SORT UNIQUE NOSORT|         |      1 |     41M|   9454 |00:02:34.71 |     274K|    274K|
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     41M|     50M|00:01:40.05 |     274K|    274K|
-------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement
```

#TEST RUN 18 8KB UNIFORM 1MB NO HT

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:07.73

PL/SQL procedure successfully completed.

Elapsed: 00:02:10.93

System altered.

Elapsed: 00:00:00.06

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.01

no rows selected

Elapsed: 00:01:08.59

Execution Plan
----------------------------------------------------------
Plan hash value: 2134347679

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    31 |   178K  (1)| 00:35:37 |
|   1 |  HASH UNIQUE       |      |     1 |    31 |   178K  (1)| 00:35:37 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |    31 |   178K  (1)| 00:35:37 |
---------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("STATUS"='NONE')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
     651991  consistent gets
     651480  physical reads
```

```
          0  redo size
        399  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


Session altered.

Elapsed: 00:00:00.01

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
T1                               50086655     652598          88
T2


INDEX_NAME                           BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------------------------ ---------- ----------- ------------- ----------------------- ----------------------- ----------
-------
T1_IND1                                   3      273232      47204490                       1                       1
48319593
T2_IND1
```

16KB UNIFORM 1MB
#TEST RUN 19 16KB BLOCK SIZE UNIFORM 1MB

```
  COUNT(*)
----------
11073

Elapsed: 00:00:00.67

Execution Plan...

Statistics
----------------------------------------------------------
        641  recursive calls
          0  db block gets
      19499  consistent gets
        209  physical reads
          0  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         25  sorts (memory)
          0  sorts (disk)
          1  rows processed


Table created.

Elapsed: 00:01:53.65

Commit complete.

Elapsed: 00:00:00.00

System altered.

Elapsed: 00:00:03.00

System altered.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:08:41.06

Table created.

Elapsed: 00:00:00.85

Index created.

Elapsed: 00:00:00.01

System altered.

Elapsed: 00:00:01.17

System altered.

Elapsed: 00:00:00.01

1000000 rows created.

Elapsed: 00:01:40.81

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |      |   751K|   101M|   122K  (2)| 00:28:38 |
|*  1 |  TABLE ACCESS FULL| T1   |   751K|   101M|   122K  (2)| 00:28:38 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RN"<=100)
```

```
Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       8029  recursive calls
    2492121  db block gets
     353899  consistent gets
     321655  physical reads
  446333292  redo size
        681  bytes sent via SQL*Net to client
        583  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
    1000000  rows processed


Commit complete.

Elapsed: 00:00:00.01

System altered.

Elapsed: 00:00:14.76

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.03

no rows selected

Elapsed: 00:01:08.53

Execution Plan
----------------------------------------------------------
Plan hash value: 3617692013

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |  3544 |   487K|   122K  (2)| 00:28:34 |
|*  1 |  TABLE ACCESS FULL| T1   |  3544 |   487K|   122K  (2)| 00:28:34 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("STATUS"='NONE')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     322659  consistent gets
     321574  physical reads
          0  redo size
       1047  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


  COUNT(*)
----------
1000000

Elapsed: 00:00:02.57

Execution Plan
----------------------------------------------------------
Plan hash value: 1385691034

-------------------------------------------------------------------------
| Id  | Operation             | Name    | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |     1 |  1232   (1)| 00:00:18 |
|   1 |  SORT AGGREGATE       |         |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| T2_IND1 |   909K|  1232   (1)| 00:00:18 |
-------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         32  recursive calls
          3  db block gets
       6815  consistent gets
       4034  physical reads
     242216  redo size
        411  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


OWNER                          OBJECT_NAME
------------------------------ ------------------------------
SUBOBJECT_NAME
------------------------------

9454 rows selected.

Elapsed: 00:01:18.92

Execution Plan
----------------------------------------------------------
Plan hash value: 1118578911

------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |    54M|  2666M|   574K  (1)| 02:14:01 |
|   1 |  SORT UNIQUE NOSORT|         |    54M|  2666M|   574K  (1)| 02:14:01 |
|   2 |   INDEX FULL SCAN  | T1_IND1 |    54M|  2666M|   136K  (1)| 00:31:51 |
------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
     136284  consistent gets
     135107  physical reads
          0  redo size
     299135  bytes sent via SQL*Net to client
       7311  bytes received via SQL*Net from client
        632  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9454  rows processed


Session altered.

Elapsed: 00:00:00.00
```

#TEST RUN 20 16KB BLOCK SIZE UNIFORM 1MB

```
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1;

-------------------------------------------------------------------------------------------------
| Id  | Operation          | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------
|   1 |  SORT UNIQUE NOSORT|         |      1 |     54M|   9454 |00:02:10.55 |     135K|    135K|
|   2 |   INDEX FULL SCAN  | T1_IND1 |      1 |     54M|     50M|00:01:40.04 |     135K|    135K|
-------------------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement
```

#TEST RUN 21 16KB BLOCK SIZE UNIFORM 1MB

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:10.01

PL/SQL procedure successfully completed.

Elapsed: 00:02:21.18

System altered.

Elapsed: 00:00:00.07

System altered.

Elapsed: 00:00:00.00

Session altered.

Elapsed: 00:00:00.03

no rows selected

Elapsed: 00:01:07.40

Execution Plan
----------------------------------------------------------
Plan hash value: 2134347679

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    33 |   122K  (2)| 00:28:32 |
|   1 |  HASH UNIQUE       |      |     1 |    33 |   122K  (2)| 00:28:32 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |    33 |   122K  (2)| 00:28:32 |
---------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
```

```
   2 - filter("STATUS"='NONE')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
     322079  consistent gets
     321574  physical reads
          0  redo size
        399  bytes sent via SQL*Net to client
        370  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


Session altered.

Elapsed: 00:00:00.00

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
T1                               50113013     322129          88
T2


INDEX_NAME                         BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------------------------ ---------- ----------- ------------- ----------------------- ----------------------- ----------
-------
T1_IND1                                 2      139800      48861273                       1                       1
49619952
T2_IND1


The TKPROF output will follow.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.
```

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 9, 2008 4:17 PM in response to: Reply

>Can you get him to pop-in here and tell us about it?

I think, it’s up to that person and is aware of this thread. TOE prevents me to publish any other details, but i am expecting the promised article from him.In fact he was kind enough to share some of his “Experience” with “Experts” from Oracle itself.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 9, 2008 4:26 PM in response to: David Aldridge Reply

> Also, what sort of load is this .. a batch job? Or
> regular OLTP operations?
>
> Message was edited by: DA. Typo, changed "tow" to
> "two"
> David Aldridge

[XML Row Data from OLTP] --> Batch Job --> [DW] --> Batch Job --> Reporting System

We compared the performance based on overall job completion intervals in various stages and the cpu+elapsed in tkprof for each query executed. Also we have scheduled ADDM to monitor the performance and compared the results for both the database during the batch job

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 9, 2008 4:38 PM in response to: Jonathan Lewis Reply

> Go on, just one little tkprof extract from each
> database that shows a meaningful performance
> improvement without a change in execution plan.
> Surely it won't lose you your red lobster lunch, even
> if someone why there was a difference.

Jonathan,

I don't see any point in publishing the results any more. After all i am not here to prove “I am DAM right and you are Wrong”, but to share my experience with the performance improvement in my DW application. As i said earlier, you may have hundreds of other excuses. What i see is the response time, cpu utilization and the network traffic.

sp009

Niall Litchfield

Re: Larger vs. Small data block
Posted: Jun 9, 2008 5:47 PM in response to: Reply

Posts: 301
From: Hampshire UK
Registered: 7/4/99

> Hi Niall,
>
> >> Possibly it's somewhat unfortunate for your case
> then that Ashenfeltzer's predictions were more
> reliable than Robert Parker's.
>
> Excellent, you are paying attention!
>
> Obviously wine tasting is a subjective thing (I'm
> just a country redneck, not an oenophile!), and the
> supertatives used wine snobs strike me as ridiculous!
> I like the Borat approach to wine tasting, myself:
>
> http://www.youtube.com/watch?v=oKcWtvEzdR8
>
> On the other hand, Oracle tuning has an objective
> measure of success, namely faster throughput and
> response time.

tasting certainly is, but price predicition is rather objective. It is in the objective arena that the guru lost.

> My point was that the decision rules of Oracle
> performance tuning are too complex for automation,
> else it would have been done years ago . . . .

I don't know, was it tried and found wanting years ago, tried and found difficult and abandoned, or just not tried? Too complex sounds like an admission of failure.

Niall

dbms.jedi

Posts: 1
Registered: 5/25/08

Re: Larger vs. Small data block
Posted: Jun 9, 2008 6:45 PM in response to: sp009

Reply

> Just finished the analysis of tkprof of a job scheduled on week-end, which process 30m rows,
> in production (16k db_block_size) and test (8k db_block_size) databases installed in identical
> Server Win 2003/64b ASM RAID. Before the job run, i refreshed the data in test so that both
> database will have same volume. Guess what, there is 18% difference in response time and
> the cpu utilization between the production and test database. My supervisor discussed the
> End-result with our consultant DBA (From a world famous Consultancy Group (Oracle???),
> and is labeled as performance Guru!). End result? i am expecting a pay raise pretty soon and
> our consultant DBA owes me a lunch at red lobster. I don't see any point in cut & paste the tkprof
> result in the forum. Lab experts may have hundreds of excuses for this performance difference.
> Also our consultant DBA promised to publish some article in Oracle Magazine regarding the
> benefits of higher block size in Warehouse application very soon.
>

There are a few things that are unclear to me.
1) If your production database has a 16k db_block_size, then what was the purpose of cloning it to an 8k block test db? Just to test that a db with 8k block is slower? (and getting a free lunch?)
2) What does 18% represent? CPU consumption? Elapsed time? Or was there 18% reduction in both? What did you use to capture the metrics to come up with the 18% CPU (sar, vmstat, Oracle tool)?
3) Could you describe what operations take place in this job? CTAS, inserts, updates, selects? If a mix, a rough breakdown.
4) Do you use Parallel Query or Compression?
5) It would be useful, and another case of real-world data, if you could share some technical details about this observation. Don't let the critics get to you. Let the numbers to the talking.
6) So you have this performance Guru, who is publishing an article in Oracle Magazine about the benefits of a larger block size in a data warehouse, who bet you that a 8k db_block_size would be faster than 16k (hence you won the bet). Am I missing something or was that a bad bet for him to take, given he would have some insight that 16k would be better, no?

Cheers.

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 9, 2008 6:47 PM in response to: Charles Hooper

Reply

TKPROF output for the last 3 sets of tests follows:

Test 13 8KB UNIFORM 1MB:
```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     35.40     103.21     274152     274642          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     35.40     103.24     274153     274644          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=274642 pr=274152 pw=0 time=105558079 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274642 pr=274152 pw=0 time=100021899 us)(object id 11757)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     632        0.00          0.00
  db file scattered read                        897        0.03          2.90
  db file sequential read                    246668        0.03         68.94
  SQL*Net message from client                   632        0.01          2.76
********************************************************************************
```

Test 16 8KB UNIFORM 1MB NO HYPER-THREADING:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     33.75      99.50     274183     274678          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     33.75      99.52     274184     274680          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=274678 pr=274183 pw=0 time=96086174 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274678 pr=274183 pw=0 time=100021870 us)(object id 11757)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     632        0.00          0.00
  db file scattered read                        877        0.04          2.62
  db file sequential read                    247316        0.03         66.02
  SQL*Net message from client                   632        0.01          2.38
********************************************************************************
```

Test 19 16KB UNIFORM 1MB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          1          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      632     29.09      76.03     135106     135703          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      634     29.09      76.05     135107     135705          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=135703 pr=135106 pw=0 time=79117626 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=135703 pr=135106 pw=0 time=100030548 us)(object id 11767)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     632        0.00          0.00
  db file scattered read                        902        0.02          2.68
  db file sequential read                    121747        0.04         46.01
  SQL*Net message from client                   632        0.01          2.76
********************************************************************************
```

Test 13 8KB UNIFORM 1MB:

```
********************************************************************************
SELECT
  *
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.02          1          1          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     11.90      68.78     651354     651991          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     11.92      68.80     651355     651992          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS FULL T1 (cr=651991 pr=651354 pw=0 time=68787056 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                         1        0.01          0.01
  SQL*Net message to client                       1        0.00          0.00
  db file scattered read                       5149        0.05         57.11
  SQL*Net message from client                     1        0.01          0.01

10046 Trace File:
```

```
PARSE #8:c=62500,e=756691,p=126,cr=576,cu=0,mis=1,r=0,dep=0,og=1,tim=1013390366
EXEC #8:c=0,e=30,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1013390547
WAIT #8: nam='SQL*Net message to client' ela= 3 driver id=1413697536 #bytes=1 p3=0 obj#=11756 tim=1013390588
WAIT #8: nam='db file scattered read' ela= 22563 file#=4 block#=13 blocks=124 obj#=11756 tim=1013413446
WAIT #8: nam='db file scattered read' ela= 10851 file#=4 block#=139 blocks=126 obj#=11756 tim=1013426530
WAIT #8: nam='db file scattered read' ela= 17966 file#=4 block#=267 blocks=126 obj#=11756 tim=1013446717
WAIT #8: nam='db file scattered read' ela= 9833 file#=4 block#=395 blocks=126 obj#=11756 tim=1013458815
WAIT #8: nam='db file scattered read' ela= 9822 file#=4 block#=523 blocks=126 obj#=11756 tim=1013470889
WAIT #8: nam='db file scattered read' ela= 10823 file#=4 block#=651 blocks=126 obj#=11756 tim=1013483979
WAIT #8: nam='db file scattered read' ela= 9809 file#=4 block#=779 blocks=126 obj#=11756 tim=1013496047
WAIT #8: nam='db file scattered read' ela= 9864 file#=4 block#=907 blocks=126 obj#=11756 tim=1013508149
WAIT #8: nam='db file scattered read' ela= 10431 file#=4 block#=1035 blocks=126 obj#=11756 tim=1013521209
WAIT #8: nam='db file scattered read' ela= 12268 file#=4 block#=1163 blocks=126 obj#=11756 tim=1013535706
WAIT #8: nam='db file scattered read' ela= 9776 file#=4 block#=1291 blocks=126 obj#=11756 tim=1013547806
WAIT #8: nam='db file scattered read' ela= 10788 file#=4 block#=1419 blocks=126 obj#=11756 tim=1013560865
WAIT #8: nam='db file scattered read' ela= 9850 file#=4 block#=1547 blocks=126 obj#=11756 tim=1013572967
WAIT #8: nam='db file scattered read' ela= 9841 file#=4 block#=1675 blocks=126 obj#=11756 tim=1013585035
WAIT #8: nam='db file scattered read' ela= 10831 file#=4 block#=1803 blocks=126 obj#=11756 tim=1013598125
WAIT #8: nam='db file scattered read' ela= 9838 file#=4 block#=1931 blocks=126 obj#=11756 tim=1013610197
WAIT #8: nam='db file scattered read' ela= 9846 file#=4 block#=2059 blocks=126 obj#=11756 tim=1013622299
WAIT #8: nam='db file scattered read' ela= 10833 file#=4 block#=2187 blocks=126 obj#=11756 tim=1013635383
WAIT #8: nam='db file scattered read' ela= 9777 file#=4 block#=2315 blocks=126 obj#=11756 tim=1013647455
WAIT #8: nam='db file scattered read' ela= 9846 file#=4 block#=2443 blocks=126 obj#=11756 tim=1013659558
WAIT #8: nam='db file scattered read' ela= 10803 file#=4 block#=2571 blocks=126 obj#=11756 tim=1013672614
...
WAIT #8: nam='db file scattered read' ela= 9792 file#=4 block#=651785 blocks=128 obj#=11756 tim=1082107350
WAIT #8: nam='db file scattered read' ela= 9850 file#=4 block#=651913 blocks=128 obj#=11756 tim=1082119450
WAIT #8: nam='db file scattered read' ela= 10765 file#=4 block#=652041 blocks=128 obj#=11756 tim=1082132535
WAIT #8: nam='db file scattered read' ela= 12196 file#=4 block#=652170 blocks=127 obj#=11756 tim=1082147070
WAIT #8: nam='db file scattered read' ela= 9846 file#=4 block#=652297 blocks=128 obj#=11756 tim=1082159171
WAIT #8: nam='db file scattered read' ela= 10775 file#=4 block#=652425 blocks=128 obj#=11756 tim=1082172227
WAIT #8: nam='db file scattered read' ela= 2512 file#=4 block#=652553 blocks=54 obj#=11756 tim=1082176885
FETCH #8:c=11906250,e=68787060,p=651354,cr=651991,cu=0,mis=0,r=0,dep=0,og=1,tim=1082177688
WAIT #8: nam='SQL*Net message from client' ela= 16292 driver id=1413697536 #bytes=1 p3=0 obj#=11756 tim=1082194088
STAT #8 id=1 cnt=0 pid=0 pos=1 obj=11756 op='TABLE ACCESS FULL T1 (cr=651991 pr=651354 pw=0 time=68787056 us)'
********************************************************************************
```

Test 16 8KB UNIFORM 1MB NO HYPER-THREADING:

```
********************************************************************************
SELECT
  *
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          1          1          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     12.37      74.71     651354     651991          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     12.37      74.73     651355     651992          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS FULL T1 (cr=651991 pr=651354 pw=0 time=74716184 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                         1        0.01          0.01
  SQL*Net message to client                       1        0.00          0.00
  db file scattered read                       5149        0.05         63.02
  SQL*Net message from client                     1        0.00          0.00

10046 Trace File:
PARSE #8:c=78125,e=777584,p=126,cr=576,cu=0,mis=1,r=0,dep=0,og=1,tim=1025708611
EXEC #8:c=0,e=29,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1025708796
WAIT #8: nam='SQL*Net message to client' ela= 3 driver id=1413697536 #bytes=1 p3=0 obj#=11756 tim=1025708837
WAIT #8: nam='db file scattered read' ela= 23747 file#=4 block#=13 blocks=124 obj#=11756 tim=1025732878
WAIT #8: nam='db file scattered read' ela= 29340 file#=4 block#=139 blocks=126 obj#=11756 tim=1025764447
WAIT #8: nam='db file scattered read' ela= 24745 file#=4 block#=267 blocks=126 obj#=11756 tim=1025791426
WAIT #8: nam='db file scattered read' ela= 28987 file#=4 block#=395 blocks=126 obj#=11756 tim=1025823029
WAIT #8: nam='db file scattered read' ela= 24659 file#=4 block#=523 blocks=126 obj#=11756 tim=1025849982
WAIT #8: nam='db file scattered read' ela= 29358 file#=4 block#=651 blocks=126 obj#=11756 tim=1025881582
WAIT #8: nam='db file scattered read' ela= 26131 file#=4 block#=779 blocks=126 obj#=11756 tim=1025909975
WAIT #8: nam='db file scattered read' ela= 26882 file#=4 block#=907 blocks=126 obj#=11756 tim=1025939152
WAIT #8: nam='db file scattered read' ela= 27170 file#=4 block#=1035 blocks=126 obj#=11756 tim=1025968553
WAIT #8: nam='db file scattered read' ela= 20914 file#=4 block#=1163 blocks=126 obj#=11756 tim=1025991754
WAIT #8: nam='db file scattered read' ela= 24740 file#=4 block#=1291 blocks=126 obj#=11756 tim=1026018761
WAIT #8: nam='db file scattered read' ela= 12552 file#=4 block#=1419 blocks=126 obj#=11756 tim=1026033608
WAIT #8: nam='db file scattered read' ela= 32144 file#=4 block#=1547 blocks=126 obj#=11756 tim=1026067977
WAIT #8: nam='db file scattered read' ela= 12595 file#=4 block#=1675 blocks=126 obj#=11756 tim=1026082825
WAIT #8: nam='db file scattered read' ela= 49819 file#=4 block#=1803 blocks=126 obj#=11756 tim=1026134878
WAIT #8: nam='db file scattered read' ela= 12483 file#=4 block#=1931 blocks=126 obj#=11756 tim=1026149727
WAIT #8: nam='db file scattered read' ela= 16472 file#=4 block#=2059 blocks=126 obj#=11756 tim=1026168445
WAIT #8: nam='db file scattered read' ela= 12557 file#=4 block#=2187 blocks=126 obj#=11756 tim=1026183262
WAIT #8: nam='db file scattered read' ela= 17805 file#=4 block#=2315 blocks=126 obj#=11756 tim=1026203303
WAIT #8: nam='db file scattered read' ela= 26915 file#=4 block#=2443 blocks=126 obj#=11756 tim=1026232483
WAIT #8: nam='db file scattered read' ela= 10767 file#=4 block#=2571 blocks=126 obj#=11756 tim=1026245538
...
WAIT #8: nam='db file scattered read' ela= 9812 file#=4 block#=651913 blocks=128 obj#=11756 tim=1100366689
WAIT #8: nam='db file scattered read' ela= 10792 file#=4 block#=652041 blocks=128 obj#=11756 tim=1100379776
WAIT #8: nam='db file scattered read' ela= 12267 file#=4 block#=652170 blocks=127 obj#=11756 tim=1100394310
WAIT #8: nam='db file scattered read' ela= 9792 file#=4 block#=652297 blocks=128 obj#=11756 tim=1100406412
WAIT #8: nam='db file scattered read' ela= 10790 file#=4 block#=652425 blocks=128 obj#=11756 tim=1100419469
WAIT #8: nam='db file scattered read' ela= 2647 file#=4 block#=652553 blocks=54 obj#=11756 tim=1100424242
FETCH #8:c=12375000,e=74716188,p=651354,cr=651991,cu=0,mis=0,r=0,dep=0,og=1,tim=1100425065
WAIT #8: nam='SQL*Net message from client' ela= 611 driver id=1413697536 #bytes=1 p3=0 obj#=11756 tim=1100425772
*** SESSION ID:(215.3) 2008-06-08 19:19:17.562
STAT #8 id=1 cnt=0 pid=0 pos=1 obj=11756 op='TABLE ACCESS FULL T1 (cr=651991 pr=651354 pw=0 time=74716184 us)'
********************************************************************************
```

Test 19 16KB UNIFORM 1MB:

```
********************************************************************************
SELECT
  *
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          1          1          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      9.68      67.68     321440     322079          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      9.68      67.70     321441     322080          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS FULL T1 (cr=322079 pr=321440 pw=0 time=67682309 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                         2        0.01          0.01
  SQL*Net message to client                       1        0.00          0.00
  db file scattered read                       5098        0.05         58.14
  SQL*Net message from client                     1        0.02          0.02

10046 Trace File:
PARSE #13:c=125000,e=807591,p=134,cr=580,cu=0,mis=1,r=0,dep=0,og=1,tim=994728652
EXEC #13:c=0,e=27,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=994728829
WAIT #13: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=11766 tim=994728869
WAIT #13: nam='db file scattered read' ela= 21658 file#=4 block#=8 blocks=61 obj#=11766 tim=994750723
WAIT #13: nam='db file scattered read' ela= 10290 file#=4 block#=70 blocks=63 obj#=11766 tim=994762791
WAIT #13: nam='db file scattered read' ela= 34258 file#=4 block#=134 blocks=63 obj#=11766 tim=994798898
WAIT #13: nam='db file scattered read' ela= 10220 file#=4 block#=198 blocks=63 obj#=11766 tim=994811001
WAIT #13: nam='db file scattered read' ela= 20595 file#=4 block#=262 blocks=63 obj#=11766 tim=994833446
WAIT #13: nam='db file scattered read' ela= 11211 file#=4 block#=326 blocks=63 obj#=11766 tim=994846531
WAIT #13: nam='db file scattered read' ela= 10120 file#=4 block#=390 blocks=63 obj#=11766 tim=994858602
WAIT #13: nam='db file scattered read' ela= 10254 file#=4 block#=454 blocks=63 obj#=11766 tim=994870706
WAIT #13: nam='db file scattered read' ela= 11203 file#=4 block#=518 blocks=63 obj#=11766 tim=994883763
WAIT #13: nam='db file scattered read' ela= 12652 file#=4 block#=582 blocks=63 obj#=11766 tim=994898261
WAIT #13: nam='db file scattered read' ela= 10240 file#=4 block#=646 blocks=63 obj#=11766 tim=994910361
WAIT #13: nam='db file scattered read' ela= 11216 file#=4 block#=710 blocks=63 obj#=11766 tim=994923420
WAIT #13: nam='db file scattered read' ela= 10218 file#=4 block#=774 blocks=63 obj#=11766 tim=994935524
WAIT #13: nam='db file scattered read' ela= 10141 file#=4 block#=838 blocks=63 obj#=11766 tim=994947593
WAIT #13: nam='db file scattered read' ela= 11218 file#=4 block#=902 blocks=63 obj#=11766 tim=994960683
WAIT #13: nam='db file scattered read' ela= 10191 file#=4 block#=966 blocks=63 obj#=11766 tim=994972754
WAIT #13: nam='db file scattered read' ela= 10250 file#=4 block#=1030 blocks=63 obj#=11766 tim=994984855
WAIT #13: nam='db file scattered read' ela= 11238 file#=4 block#=1094 blocks=63 obj#=11766 tim=994997942
WAIT #13: nam='db file scattered read' ela= 10231 file#=4 block#=1158 blocks=63 obj#=11766 tim=995010021
WAIT #13: nam='db file scattered read' ela= 10230 file#=4 block#=1222 blocks=63 obj#=11766 tim=995022118
WAIT #13: nam='db file scattered read' ela= 11236 file#=4 block#=1286 blocks=63 obj#=11766 tim=995035218
...
WAIT #13: nam='db file scattered read' ela= 10944 file#=4 block#=321733 blocks=64 obj#=11766 tim=1062345199
WAIT #13: nam='db file scattered read' ela= 10198 file#=4 block#=321797 blocks=64 obj#=11766 tim=1062357304
WAIT #13: nam='db file scattered read' ela= 10165 file#=4 block#=321861 blocks=64 obj#=11766 tim=1062369370
WAIT #13: nam='db file scattered read' ela= 11204 file#=4 block#=321925 blocks=64 obj#=11766 tim=1062382459
WAIT #13: nam='db file scattered read' ela= 10189 file#=4 block#=321989 blocks=64 obj#=11766 tim=1062394527
WAIT #13: nam='db file scattered read' ela= 10217 file#=4 block#=322053 blocks=64 obj#=11766 tim=1062406633
WAIT #13: nam='db file scattered read' ela= 2336 file#=4 block#=322117 blocks=17 obj#=11766 tim=1062410764
FETCH #13:c=9687500,e=67682313,p=321440,cr=322079,cu=0,mis=0,r=0,dep=0,og=1,tim=1062411223
WAIT #13: nam='SQL*Net message from client' ela= 28025 driver id=1413697536 #bytes=1 p3=0 obj#=11766 tim=1062439340
STAT #13 id=1 cnt=0 pid=0 pos=1 obj=11766 op='TABLE ACCESS FULL T1 (cr=322079 pr=321440 pw=0 time=67682309 us)'
********************************************************************************
```

Test 13 8KB UNIFORM 1MB:

```
********************************************************************************
SELECT
  COUNT(*)
FROM
  T2

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          2          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.28       1.64       6979      13950          2           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.29       1.66       6981      13952          2           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=13950 pr=6979 pw=0 time=1647916 us)
1000000   INDEX FAST FULL SCAN T2_IND1 (cr=13950 pr=6979 pw=0 time=321071 us)(object id 11759)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         7        0.01          0.04
  db file parallel read                           1        0.28          0.28
  db file scattered read                        110        0.03          1.02
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************
```

Test 16 8KB UNIFORM 1MB NO HYPER-THREADING:

```
********************************************************************************
SELECT
  COUNT(*)
FROM
  T2

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          2          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.26       1.57       6974      13934          2           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.26       1.59       6976      13936          2           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=13934 pr=6974 pw=0 time=1572482 us)
1000000   INDEX FAST FULL SCAN T2_IND1 (cr=13934 pr=6974 pw=0 time=2245925 us)(object id 11759)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         2        0.01          0.01
  db file parallel read                           1        0.20          0.20
  db file scattered read                        110        0.03          1.06
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************
```

Test 19 16KB UNIFORM 1MB:

```
********************************************************************************
SELECT
  COUNT(*)
FROM
  T2

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          2          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.21       1.72       3332       6655          2           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.21       1.74       3334       6657          2           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=6655 pr=3332 pw=0 time=1723813 us)
1000000   INDEX FAST FULL SCAN T2_IND1 (cr=6655 pr=3332 pw=0 time=211293 us)(object id 11769)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         7        0.02          0.06
  db file parallel read                           1        0.16          0.16
  db file scattered read                         53        0.03          1.23
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************
```

Test 13 8KB UNIFORM 1MB:

```
********************************************************************************
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       16      0.03       0.08          5         10          0           0
Execute     17      0.01       0.09         18        142          8           8
Fetch      642     47.59     173.65     932485     940583          2        9498
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      675     47.64     173.83     932508     940735         10        9506

Misses in library cache during parse: 9
Misses in library cache during execute: 3

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     668        0.00          0.00
  SQL*Net message from client                   668        0.01          2.79
  db file sequential read                    246703        0.03         69.16
  db file scattered read                       6156        0.05         61.04
  db file parallel read                           1        0.28          0.28
********************************************************************************
```

Test 16 8KB UNIFORM 1MB NO HYPER-THREADING:

```
********************************************************************************
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       16      0.00       0.08          5         10          0           0
Execute     17      0.03       0.09         19        142          8           8
Fetch      642     46.39     175.80     932511     940603          2        9498
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      675     46.42     175.97     932535     940755         10        9506
```

```
Misses in library cache during parse: 9
Misses in library cache during execute: 3

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     668        0.00          0.00
  SQL*Net message from client                   668        0.01          2.40
  db file sequential read                    247344        0.03         66.19
  db file scattered read                       6137        0.05         66.71
  db file parallel read                           1        0.20          0.20
********************************************************************************
```

Test 19 16KB UNIFORM 1MB:

```
********************************************************************************
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       16      0.00       0.08          5         10          0           0
Execute     17      0.00       0.07         15        136          8           8
Fetch      642     39.00     145.44     459878     464437          2        9498
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      675     39.00     145.60     459898     464583         10        9506

Misses in library cache during parse: 9
Misses in library cache during execute: 3

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                     668        0.00          0.00
  SQL*Net message from client                   668        0.02          2.80
  db file sequential read                    121778        0.04         46.24
  db file scattered read                       6053        0.05         62.06
  db file parallel read                           1        0.16          0.16
********************************************************************************
```

Test 14 8KB UNIFORM 1MB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.16          0          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       95     83.32     156.69     274014     274108          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       97     83.34     156.86     274014     274110          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=274108 pr=274014 pw=0 time=156655409 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274108 pr=274014 pw=0 time=100047277 us)(object id 11757)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                      95        0.00          0.00
  db file sequential read                    274014        0.02         76.88
  SQL*Net more data to client                    85        0.00          0.00
  SQL*Net message from client                    95        0.68          0.73
********************************************************************************
```

Test 17 8KB UNIFORM 1MB NO HYPER-THREADING:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.06       0.16          0          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       95     84.09     154.75     274048     274142          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       97     84.15     154.91     274048     274144          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=274142 pr=274048 pw=0 time=154707761 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=274142 pr=274048 pw=0 time=100051703 us)(object id 11757)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                      95        0.00          0.00
  db file sequential read                    274048        0.03         74.96
  SQL*Net more data to client                    84        0.00          0.01
  SQL*Net message from client                    95        0.68          0.73
********************************************************************************
```

Test 20 16KB UNIFORM 1MB:

```
********************************************************************************
SELECT /*+ INDEX(T1) */ DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.07       0.14          0          2          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       95     77.56     130.58     135072     135166          0        9454
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       97     77.64     130.73     135072     135168          0        9454

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
   9454  SORT UNIQUE NOSORT (cr=135166 pr=135072 pw=0 time=130551689 us)
50000000   INDEX FULL SCAN T1_IND1 (cr=135166 pr=135072 pw=0 time=100037933 us)(object id 11767)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                      95        0.00          0.00
  db file sequential read                    135072        0.03         54.33
  SQL*Net more data to client                    84        0.00          0.00
  SQL*Net message from client                    95        0.11          0.15
********************************************************************************
```

Test 15 8KB UNIFORM 1MB:

```
********************************************************************************
SELECT DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     12.14      71.07     651480     651991          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     12.14      71.09     651480     651991          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  HASH UNIQUE (cr=651991 pr=651480 pw=0 time=71073190 us)
      0   TABLE ACCESS FULL T1 (cr=651991 pr=651480 pw=0 time=71073083 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  db file sequential read                        11        0.01          0.08
  db file scattered read                       5099        0.05         59.47
  SQL*Net message from client                     1        0.01          0.01
********************************************************************************
```

Test 18 8KB UNIFORM 1MB NO HYPER-THREADING:

```
********************************************************************************
SELECT DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.02          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1     11.68      68.24     651480     651991          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3     11.68      68.26     651480     651991          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  HASH UNIQUE (cr=651991 pr=651480 pw=0 time=68245800 us)
      0   TABLE ACCESS FULL T1 (cr=651991 pr=651480 pw=0 time=68245706 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  db file sequential read                        11        0.01          0.07
  db file scattered read                       5099        0.05         56.71
  SQL*Net message from client                     1        0.02          0.02
********************************************************************************
```

Test 21 16KB UNIFORM 1MB:

```
********************************************************************************
SELECT DISTINCT
  OWNER,
  OBJECT_NAME,
  SUBOBJECT_NAME
FROM
  T1
WHERE
  STATUS='NONE'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      9.23      67.17     321574     322079          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      9.23      67.18     321574     322079          0           0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 30

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  HASH UNIQUE (cr=322079 pr=321574 pw=0 time=67171002 us)
      0   TABLE ACCESS FULL T1 (cr=322079 pr=321574 pw=0 time=67170929 us)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       1        0.00          0.00
  db file sequential read                         6        0.01          0.03
  db file scattered read                       5034        0.04         57.78
  SQL*Net message from client                     1        0.04          0.04
********************************************************************************
```

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 9, 2008 6:57 PM in response to: sp009 Reply

>

> [XML Row Data from OLTP] --> Batch Job --> [DW] -->
> Batch Job --> Reporting System
>
> We compared the performance based on overall job
> completion intervals in various stages
> and the cpu+elapsed in tkprof for each query
> executed. Also we have scheduled ADDM
> to monitor the performance and compared the results
> for both the database during the batch job
>

So was this an across the board benefit in CPU reduction, or one that you saw more on particular queries than others? What sort of operations apeared to benefit most?

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 9, 2008 8:28 PM in response to: Charles Hooper Reply

Excellent work Greg and Charles.

I will be on the road for the next couple of days but if you can package your setup and test scripts I have an 11gR1 RAC cluster in the lab and I would like to run them with the addition of cache fusion. Thanks.

Nick Naughty

Posts: 296
Registered: 5/3/07

Re: Larger vs. Small data block
Posted: Jun 10, 2008 1:45 AM in response to: user619401 Reply

nice

Howardjr

Posts: 11
Registered: 6/7/07

Re: Larger vs. Small data block
Posted: Jun 10, 2008 2:54 AM in response to: Reply

I first learned it fro Oracle University in the early 1990's,

Er, no you didn't. Oracle University didn't even exist until the late 1990s. Before they aggrandised that title to themselves, they were merely "Oracle Education". I remember the change coming in to Australia in, I think, 2000, possibly 2001... and being mightily puzzled, since it's illegal to call yourself a university in Australia unless you've been granted a charter to do so (which OU certainly hadn't at that time and probably still hasn't). But I digress...

Multiple block sizes weren't introduced to Oracle's RDBMS until version 9i, and that wasn't taught by OU until 2001. So again, the "early 1990s" timeframe is just plain wrong.

BTW, it was presented as fact by OU, not theory

Er, no it wasn't. At least, not in the sense you wish to imply. It was mentioned in the context of transportable tablespaces only. In Performance Tuning, there was a reference to the difficulty of coming up with one 'correct' blocksize when confronted with competing OLTP/OLAP-DW demands (the usual stuff about big blocks are good for full table scans, small blocks good for minimising contention). Nothing on that set of pages, however, ever suggested you should try to square the circle by combining multiple block sizes in one database.

It would be intersting to see what it says.

I thought you just said you knew what the OU material said?! Perhaps just a momentary loss of concentration on your part, then?

OU has details in the official courseware, telling students how to choose the "best" blocksize for their database

You have hit the nail on the head. 'How to choose THE best blocksize'. That would be "blocksize" singular, not "blocksizes" plural. No OU documentation published from 8.0 to 10.2 days ever recommended the use of multiple block sizes in the one database.

David, please note that the differences in performance with different blocksizes is presented on MetaLink, not as theory, but as fact:

I don't think anyone would claim that there was NOT a difference in performance, would they? What people are arguing with you about is a completely different proposition: that it makes sense to use multiple block sizes in the one database. You claim, 'it's OK, because TPC benchmarks do it'. Most others would, I think, claim that what TPC choose to do is irrelevant for your much-vaunted "real world computing experience".

marcinp1

Posts: 3
Registered: 3/1/01

Re: Larger vs. Small data block
Posted: Jun 10, 2008 3:23 AM in response to: Charles Hooper

Reply

Hi,

I really don't understand why all examples are using index full scan ?
What about index range scan ? I made some test and in my test
if you have different block in data and index tablespace response time
is a little bit worse or equal but never was better.

You can see my test results on this webpage http://oracleprof.blogspot.com/

regards,
Marcin Przepiorowski

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 10, 2008 6:21 AM in response to: marcinp1

Reply

> I really don't understand why all examples are using
> index full scan ?
> What about index range scan ? I made some test and in
> my test
> if you have different block in data and index
> tablespace response time
> is a little bit worse or equal but never was better.
>
> You can see my test results on this webpage
> http://oracleprof.blogspot.com/
>
> regards,
> Marcin Przepiorowski

What I attempted to do is to create as many possible access paths as possible with a limited and reproducible data set, while keeping as little of the previously read index and table blocks in memory to force physical reads (as if the data set were too large to fit into and remain in the buffer cache).

It takes less time to fetch a random 8KB block from main system memory (RAM) than it does to fetch a random 16KB block from main memory (RAM) - there is a certain CPU clock cycle latency with each main memory access in addition to the number of memory clock cycles required to push the data bytes back to the CPU. I would suspect that index range scans or unique scans might tip the balance toward the 8KB block size, especially if only a small number of rows are needed from each index block. The same might be true also if two tables are joined together using indexed access paths.

Random index access, as well as random index update performance might be worth exploring.

There were a couple other problems/limitations that I had with the test setup and the scripts that I constructed, and those problems were noticed after the first full set of test runs (tests 1, 2, and 3). However, I kept the test scripts unchanged through the seven full sets of test runs to limit the number of changed variables between each set of test run to just one changed variable. The majority of the problems/limitations that I found are listed here:
http://forums.oracle.com/forums/click.jspa?searchID=10228172&messageID=2575446

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Re: Larger vs. Small data block
Posted: Jun 10, 2008 6:41 AM in response to: Niall Litchfield

Reply

SeanMacGC

Posts: 7
Registered: 10/30/06

Re: Larger vs. Small data block
Posted: Jun 10, 2008 6:53 AM in response to:

Reply

>I have the same experience when I "have a feeling" about the cause of a problem. I can't put my finger on it, but I'm **often** correct . . . .

How scary is that!

What happens when you're not correct, what happens when your "*well-quantified decision rules*" are rendered worse than useless by the latest release or patch of the Oracle DBMS, what happens when your "*human intuition*" excels with Oracle 10g but bombs with Oracle 11g?

What are the scientific steps that you undertake to shine a light on the reason of your failed intuition? For, as you acknowledge herein, if you can't quite put your finger on it, you're fated to repeat it, *ad infinitum*...for no great profit at best, and disaster at worst.

Re: Larger vs. Small data block
Posted: Jun 10, 2008 7:11 AM in response to: Howardjr

Reply

Re: Larger vs. Small data block
Posted: Jun 10, 2008 7:21 AM in response to: SeanMacGC

Reply

Re: Larger vs. Small data block
Posted: Jun 10, 2008 7:29 AM in response to: sp009

Reply

SeanMacGC

Posts: 7
Registered: 10/30/06

Re: Larger vs. Small data block
Posted: Jun 10, 2008 8:14 AM in response to:

Reply

>That's absolutely untrue, published by a semi-anonymous self-proclaimed expert who goes to great >pains to hide his work experience and credentials. It's like your "fellow" Australian, Howar5d J. Rogers >noted, when he called Jonathan Lewis an "idiot":

>http://dizwell.com/2008/06/07/go-on-try-it/

Check that again, you'll find Howard was calling *himself* an idiot. Tut, tut.

!

Re: Larger vs. Small data block
Posted: Jun 10, 2008 8:19 AM in response to: SeanMacGC

Reply

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 10, 2008 10:10 AM in response to: Charles Hooper

Reply

Charles,

Since you have done so much extensive testing, do you think higher block size
benefits for certain applications? or do you ever consider creating database with
higher block size or an OLAP or DSS environment?

Regards,
sp009

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 10, 2008 10:26 AM in response to:

Reply

> >> Check that again, you'll find Howard was calling himself an idiot.
>
> Well, the link "you'll get some idiot" pointed to a Lewis web page . . . .
>

Of course, many people would be inclined to follow the link and read the article rather than using their intuition to guess what might be at the other end - especially if they were planning to use it in a discussion.

Typical approach really.

Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." (Stephen Hawking)

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 10, 2008 11:26 AM in response to:

Reply

> Today, we know the top CIO's and CEO's of large
> corporations can earn hundreds of millions of
> dollars a year, largely for their human intuition.

The more interesting fact is that even when they fail miserably they still get the big money.
http://blogs.usatoday.com/oped/2007/01/our_view_on_ceo.html

Apparently it's not possible to judge a person's competence based on their success in business.

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 10, 2008 11:44 AM in response to: David Aldridge

Reply

<off topic>
> Apparently it's not possible to judge a person's
> competence based on their success in business.

Not restricted to people.

Popular does not necessarily imply Good. Popular however often implies successful. I can think of examples in industry. For example, we are all familiar with a very successful software company as well as a very successful fast food company,

It never fails to amaze me how advertising can create popularity and create the appearance of 'good'.
</off topic>

Re: Larger vs. Small data block
Posted: Jun 10, 2008 3:20 PM in response to: David Aldridge

Reply

Re: Larger vs. Small data block
Posted: Jun 10, 2008 3:45 PM in response to: Jonathan Lewis

Reply

Niall Litchfield

Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 10, 2008 4:36 PM in response to:

Reply

And so we descend to an ad-hominem attack. Again.

> **1 -Envy** - Start with the attitude that you
> "deserve" more because you think that you are smarter
> than others.

I'm quite surprised that an author of articles about being careful what you say on the net should choose to ascribe negative motivations to others, whose state of mind he cannot by definition know.

>
> **2 - Rigid mindset** - Adopt the mantra "question
> authority" and the narrow-minded approach that
> anything that cannot be proven with rules is
> nonsense. Kinda like a hippie.

the approach is that anything is untestable and unproven, is well - unproven and not reliable.

> **3 - Hide your true credentials and experience**
> Even though you apparently never studied science and
> know almost nothing about scientific research,
> self-appoint yourself as an "Oracle Scientist".

that dates back to the OakTable description

" But, they do have one special trait in common. They strive to adopt a scientific approach to their work - so they don't make claims about Oracle's performance unless they can construct a reproducible test case; they don't believe any claims about Oracle's performance unless the claim is backed by a well-argued proof."

we're quite happy to stand by that, doesn't say we're physics PhDs, just what counts as evidence, what doesn't and what our basic approach is.

>
> **4 - Deceive your readers with nonsense.**
> Declare that science says that you can "prove" any
> concept wrong, by showing any negative test case.

Karl Popper says that. You'll find most scientists (of the white coats and labs variety) would agree that reproducible test cases that consistently negate predictions or assertions do disprove the theory that made those predictions.

> **5 - Debunk!** - Attack anyone who dares not to
> "respect your authority". Just like Cartman:
>
> http://www.poster.net/south-park/south-park-you-will-r
> espect-my-authority-3700212.jpg
>
>
> **5 - Confuse people** - Use this slight-of-hand
> trick to have extremely useful MetaLink notes removed
> and "prove" that almost every Oracle tuning concept
> is all wrong! Shame on you.

er Jonathan removed them, or Oracle?

> I see that your most recent attempt to "debunk" me
> blew-up in your face again. At least we all know it
> now, that you know very little about the scientific
> method . . . .
>
> http://jonathanlewis.wordpress.com/2008/06/08/scientif
> ic-method/
>
> Can you admit it now?

er no, that page doesn't show anything of the sort - other people do follow links by the way.

> Seriously Jonathan, keep an open mind! Everyone here
> can teach us something . . . Just cause something
> can't be proven does not mean that it's not true.

nor does it mean it's false - it does mean that promoting it as fact is rather irresponsible.

Niall

Re: Larger vs. Small data block
Posted: Jun 10, 2008 4:51 PM in response to: Niall Litchfield
Reply

sybrandb
Posts: 4,042
From: Amsterdam, Netherlands
Registered: 8/4/98

Re: Larger vs. Small data block
Posted: Jun 10, 2008 5:04 PM in response to:
Reply

I don't think you ever reported anything except for gross generalizations.
You and your experts always make general claims, which are always defeated by testcases demonstrating the contrary.
Could you consider, please, what this means for your credibility?
If you would only post 1 (ONE) testcase supporting your claims, wouldn't that make a whole lot of difference?
Wouldn't that also be a more professional contribution to this debate compared to your ongoing attacks on Jonathan Lewis?
And finally, if you state 'The doc's suck', why don't you file documentation bugs?
Everyone has the right to do so!

--
Sybrand Bakker
Senior Oracle DBA

Tubby
Posts: 917
From: Vancouver
Registered: 10/1/01

Re: Larger vs. Small data block
Posted: Jun 10, 2008 6:06 PM in response to:
Reply

> Unfortunately, this "prove it" junk has become beyond ridiculous.

I couldn't agree more.

On a completely unrelated topic, i just finished traveling back in time, where i stopped Jonathan Lewis from assassinating your great great grandfather, thereby securing your existence in this reality.

If you'd like, i can give you my pal pay account where you can properly thank me.

Howardjr
Posts: 11
Registered: 6/7/07

Re: Larger vs. Small data block
Posted: Jun 10, 2008 6:30 PM in response to:
Reply

No, I'm not Anon. The name is there in black and white.

Who cares what it was called?

Good point. I mean, let's not worry too much about mere facts and details, eh? Let's just paint with a broad brush and get the facts and details wrong, shall we?

Oracle did not have transportable tablespaces in the early 1990's

Of course they didn't. Which is how come I know your claim that you read about multiple block sizes in Oracle University documentation "in the early 1990s" is a crock of miniature horse manure.

For systems with hybrid I/O, do multiple blocksizes make sense? You Bet!

Er, no, actually, I don't bet. I realise you do, and you also don't care about facts and details, but if you have a hybrid system, you have a compromise on your hands and using multiple block sizes is not the answer. Personally, instead of betting, I'd use one block size that maximised my I/O throughput and then use PCTFREE where necessary to decrease the effective block size for those tables where the large physical block size was causing contention problems.

I've deployed multiple blocksizes in mainframes for decades, many years before Oracle became popular, and it's a well-proven technique.

The fact you did this years before Oracle became popular makes this assertion completely and utterly irrelevant for a discussion on an Oracle forum about Oracle databases in 2008.

I got my pilot's license for single engine aircraft in 1992. It makes me highly unqualified to take the controls of a Boeing 787 today, I think.

Since you seem to find find the Oracle docs credible, note this:...

The piece of Oracle documentation you link to and go on to quote is all about using separate RECYCLE and KEEP caches to keep buffer access separated and distinct. It has absolutely nothing to say about the use of multiple block sizes in the one database. The fact that it is silent on the subject is significant, I think: it's a dumb thing to do and you don't have any evidence to the contrary.

Except, of course, your tired old line about TPC benchmarks. At this point, I merely repeat what I replied to you several months ago: the TPC benchmarks also run the database in noarchivelog mode and with redo generation switched off. Do you recommend those practices to your clients, too? No?? Why ever not??? Surely, if it's good enough for TPC, it's good enough for your clients??? No???? Why then, perhaps you recognise after all that TPC setups are carefully calibrated to get the best scores possible in an artifical benchmarking contest. They do NOT represent best practice for proper production databases that need supporting and long-term management. IMHO, you're preaching the same bulldust about multiple blocksizes as you have on every other technical Oracle topic for the past 8 years.

Howardjr

Posts: 11
Registered: 6/7/07

Re: Larger vs. Small data block
Posted: Jun 10, 2008 6:34 PM   in response to:
Reply

It's like your "fellow" Australian, Howar5d J. Rogers noted, when he called Jonathan Lewis an "idiot":

Yeah, just another minor fact and detail you got 100% wrong, I see.

Great researching skills, there. Not.

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 10, 2008 6:49 PM   in response to:
Reply

>
> I'm swamped this week, but I can ask a couple of experts to pop-in and report on what they have witnessed on my systems (if empirical observation counts).
>

Just checking to see if this will happen. In addition to observation and interpretation, It would be advantageous to see some AWR or Statspack of before and after. This way there are metrics to coincide with the observations. If you need to clean them WRT an NDA, please do. As long as there is no identifiable information, I've found it's never a problem. I've found where there is a will, there is a way. We ask customers to use data every year for OOW.

>
> These are ALL 100% REPRODUCEABLE benchmarks using non-standard blocksizes, yet the "Oracle Scientists" don't bother to validate them, even though they were officially sponsored by Oracle Corporation. . .
>

I do not believe anyone is arguing they are not reproducible, etc . TPC-C does use multiple block sizes as well as multiple cache pools for a reason: performance. In fact, they were specifically invented for TPC-C:
http://www.google.com/patents?id=3vELAAAAEBAJ

But understand, when a benchmark has been tuned as well as TPC-C, people look to invent new ways to squeak out performance. In speaking with one of the inventors, he mentioned to me that it might yield maybe a 5% gain, but in the next sentence, he told me that he wouldn't expect even that much from a real-world workload. This is because TPC-C only has 9 tables and 5 transactions and is 100% understood, predictable and run in a controlled environment. TPC-C also runs its db server at near 100% CPU utilization. Oh, and they also slow down the transactions, just so they make the time limits (I believe IBM was the first to come up with that one), to allow for more throughput. Would you recommend that to a customer as well? My point here is that while competitive benchmarks use niche features, and legal "tricks", the practicality of it is probably far less for the rest of the world. To put it another way: one bad execution plan would wipe that 5% and probably another 15% along the way. So when it comes to chasing block sizes or chasing good plans or good design, I would recommend the focus be on the latter because of the amount of impact. Chase the big fish. But then again, to each their own.

My biggest problem is that some seem to be positioning block size as a secret weapon to gain performance, of which it is certainly not. Not at least on its own, meaning if more than a few percent performance is gained, other variables have also likely changed and the gain is not 100% attributable to block size. Some consultants seem to like to play the "I know something you don't know and I'm not telling/showing" game and brag how many "evil performance dragons [one has] slain using magical silver bullets"[1]. I have no time for those types. And I have cleaned up after enough of them. They "fix" today's symptoms only to have the problem come back 3x worse in months. Now those who take their experience and share, explain and demonstrate and are interested in having others learn, I salute them. They advance the knowledge of others while building a reputation and many of those who benefit may never give them a dime in consultant fees (maybe some in books or seminars, etc). There is certainly a reason that Tom Kyte presentations require two 1000+ capacity rooms for the same session at OOW.

>
> Right, but you don't accept ANY real-world reports, right?
> Me and my experts will continue to report what we see, and if you want to condemn us as idiots or liars because we won't "prove it", well, that's your right . . . .
>

There isn't anyone that has taking as far as calling people liars. Let's spare the drama and stick to the technical facts. It's perfectly reasonable to ask for proof. Would you believe something on faith alone? Are you not the one asking for credentials? 'Nuff said.

So I propose that the whinging stop and some technical evidence be placed on the table. Until then, I think we are pretty much done. Although I'm sure you will need the last word and I will let you have it…

--
Regards,

Greg Rahn
http://structureddata.org

[1] Billy Verreynne http://forums.oracle.com/forums/click.jspa?searchID=10238423&messageID=2563461

Billy Verreynne

Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 11, 2008 1:12 AM   in response to: Greg Rahn
Reply

> There isn't anyone that has taking as far as calling people liars. Let's spare the drama and
> stick to the technical facts. It's perfectly reasonable to ask for proof. Would you believe
> something on faith alone?

Faith-based Oracle database healing.... Hmmm... sounds very familiar. But you would need a TV evangelist type for that.. er.. right.. How could I have missed that?

;-)

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 6:03 AM   in response to:
Reply

--

Message was edited by:
Faust

Re: Larger vs. Small data block
Posted: Jun 11, 2008 6:31 AM in response to: sybrandb
Reply

Re: Larger vs. Small data block
Posted: Jun 11, 2008 6:37 AM in response to: Faust
Reply

Re: Larger vs. Small data block
Posted: Jun 11, 2008 6:47 AM in response to: Greg Rahn
Reply

Niall Litchfield

Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 11, 2008 6:56 AM in response to:
Reply

> >> what counts as evidence, what doesn't and what our
> basic approach is.
>
> Right, but you don't accept ANY real-world reports,
> right? Whether it's because the corporation has no
> interest in proving anything to you, or because it's
> none of your business, you close yourself to the
> entire universe of production systems!

I don't believe so no, production systems are difficult sources for reproducible test cases though - since no-one else will have my production system. But a well designed test does not become invalid because it is run on a reproducible environment, unless you happen to believe that Oracle magically behaves differently in a test environment.

> ***
> ***************************************
>
> >> and what our basic approach is.
>
> "Our"? You are one of those "Woodies"? Sorry, I did
> not know that I was talking to an Oracle Scientist,
> sorry.

The chap from xxxxxxxxxxx xxxxxxxxxxxx who was checking out my profile on LinkedIn might have noticed, and my names been on the list at the OakTable website for a while now.

> Me, I'm thinking about becoming a MS-Word scientist .
> . . . .

oh a Woody? http://wopr.com

> Unfortunately, this "prove it" junk has become beyond
> ridiculous. In case you "Woodies" don't know, in 99%
> of all shops, you can be fired for disclosing "ANY"
> data from a production database.

Someone really ought to tell Oracle that, what with the RDA/SCM and all the rest of it!

> Me and my experts will continue to report what we
> see, and if you want to condenm us as idiots or liars
> because we won't "prove it", well, that's your right

Not at all, if you want to engage with the contradictory test cases, explaining what's wrong or inappropriate in them and even better suggesting a better experiment - maybe even supplying your own then the entire community would benefit. That's surely more positive than xxx xxxxxxxx says this,Tom Kyte says that and Uncle Joe Cobbly doesn't believe either of them.

Niall

seenshoo

Posts: 285
From: Maryland, USA
Registered: 3/12/01

Re: Larger vs. Small data block
Posted: Jun 11, 2008 7:00 AM in response to:
Reply

> Why would I want to? Like Oracle Press, I have a
> vested interest in keeping the the quality of the
> Oracle docs excatly where it is!.

So you admit that some oracle documentation is bad. So when someone like Jonathan lewis try to correct it, isn't that good? Doesn't that say Jonathan is trying to help oracle community with his knowledge and you are just not? Well, actually you do seem to imply you are interested in cashing in from oracle document errors. As a DBA, your above line concerns me. Where and whom would you think next time DBA like me would run to? You or Jonathan?

It was also very bad that you had to start name calling on JL few post ahead. Seriously, not very nice. Also please refrain from bringing 'oracle scientist' theory and argument in every other post and thread. In my opinion and experience, Computers is one field where you show the result and you are God. It doesn't matter if you are 8 year old or 80 year old. Degrees and Ivy leagues and expensive suits doesn't matter as long as you solve the problem at hand. You know at heart that Jonathan is genius. Admit it and why drag this topic further. More you hurl accusations, more you lower yourself.

This post/thread has been been educational with so many test cases and inputs from so many professionals. Hope this does not get deleted due to some post going personal.

Regards,

Seenshoo.

Faust

Re: Larger vs. Small data block
Posted: Jun 11, 2008 7:23 AM in response to:
Reply

Posts: 797
From: Middle Europe
Registered: 1/1/07

> Can you provide me with the e-mail?

Well, I can try to find that email - that was in January or February - I must check it...

> The STATSPACK analyzer has been used by tens of
> thousands of people, and I've never heard this
> complaint, not once.

As I posted, I didn't come to that level, after first response I was not interested in further experience of the service.

> You have made a very serious accusation here, and you
> had better be prepared to back it up.

Well, just posted what I experienced then.
Everybody, if interested, can register and see what will get.

Cheers!

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 11, 2008 7:53 AM in response to: Faust
Reply

Faust,

I signed up for StatspackAnalyzer.com to check out your claim. I tried it three times, once with just my information and no "send me extra information" checkboxes checked, once with one of them checked, and once with both checked. Each time I received an email with no attachments at all.

The first test was sent to my mailserver which I own, and it passed through an up-to-date Spamassassin/Clam filter to Outlook 2007 on my computer running Avast! Anti-Virus with fully up-to-date definitions.

Next it was sent twice to my enterprise IMAP account through GMail, which has outstanding virus/spam protection. GMail also delivered it to my inbox and reported no issues.

If you have proof that your email from StatspackAnalyzer.com contained viruses I'd love to see it, but I can't find any evidence that backs up your claim.

Re: Larger vs. Small data block
Posted: Jun 11, 2008 8:24 AM in response to: Niall Litchfield
Reply

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 8:27 AM in response to: Steve Karam
Reply

> If you have proof that your email from
> StatspackAnalyzer.com contained viruses I'd love to
> see it, but I can't find any evidence that backs up
> your claim.

Very nice that is so :-)

This evening when I'm again in my office, I will try (as I already posted) to find that email.

And at least, I didn't want to claim anybody, but to share my experience - if somebody see my post as claim that means that person see this Forum not as community, but as marketplace.

Any customer around??
;-)

Faust

Re: Larger vs. Small data block
Posted: Jun 11, 2008 8:37 AM in response to: Faust
Reply

Re: Larger vs. Small data block
Posted: Jun 11, 2008 8:43 AM in response to: Faust
Reply

SeanMacGC

Posts: 7
Registered: 10/30/06

Re: Larger vs. Small data block
Posted: Jun 11, 2008 8:48 AM in response to:
Reply

>> unless you happen to believe that Oracle *magically* behaves differently in a test environment.

>Yes, I believe that, absolutely! Try it on any of the millions of other possible combinations, and the performance results WILL be different.

Really? You believe that performance results differ **by magic**?

How utterly reassuring!

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 11, 2008 8:49 AM in response to:
Reply

> >> But a well designed test does not become invalid
> because it is run on a reproducible environment
>
> Yes, but it's valid ONLY for a single user
> environment, only on that specific server, disks,
> release, patch level, MBRC, and so on, ad infinitum.

>

If that's true, and I think it is only to a limited extent, then it emphasises to me the need for test cases that can be executed on multiple releases, multiple MBRC, multiple servers etc.. Without that, you would have no way of knowing whether advice was valid for your circumstances.

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 8:56 AM in response to:

Reply

--

Message was edited by:
Faust

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:00 AM in response to: Greg Rahn

Reply

Interesting thread. Especially now that SPG is effectively being accused of not understanding real world performance.

Musings - none of which require a response:

1) Oracle IS also a real world company, They have their own business systems that provide financial info, payroll for +50K employees, mail services for those same employees. If any organization has an extreme real world load, it's Oracle;

2) I wonder which professional consulting companies or individual consultants have such a strong reputation that Oracle has invited them in to look at the performance for those internal Oracle apps. (Probably can't tell due to some NDA.)

3) I wonder whether RWPG is asked to look at, or work on, the performance of those same apps.

4) I wonder whether Oracle uses multiple block sizes in those apps for any reason other than transportable tablespace.

5) With that kind of load available, and with the 11G Real App Testing feature and the data masking capability, I wonder whether Oracle did/will test against a sanitized real world data set based on that real world load.

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:01 AM in response to:

Reply

> You published, as a fact, that Texas Memory Systems
> is engaging in unethical behavior.

The way I read it, he published that he received a problematic email from that source, not that the source sent him one.

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:24 AM in response to: Greg Rahn

Reply

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:30 AM in response to: David Aldridge

Reply

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:36 AM in response to: Faust

Reply

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:50 AM in response to:

Reply

With respect to your statement:
"What about the fact that many RAC shops use a 2k blocksize to improve throughput performance?"

The issue with RAC relates to cache fusion with the memory interconnect.

This solution you make reference to is limiting the number of blocks being passed between instances.

One way to do that is to limit the number of records per block which can be accomplished by a variety of techniques of which, a smaller block size, is only one of them. Any technique that minimizes block sharing will have a positive affect.

That said ... a better solution is to fix the application's design or make better use of a restricting node access using services.

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:59 AM in response to: damorgan

Reply

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 11, 2008 10:14 AM in response to: damorgan

Reply

damorgan said:
> That said ... a better solution is to fix the
> application's design or make better use of a
> restricting node access using services.

Just a couple notes to avoid generalities:

While I would agree that minimizing block touches via application tuning is a great idea, most of the time the allowed concurrency and the amount of data retrieved are business rule constants and therefore cannot be changed. So while we can definitely tune queries to minimize block touches, it still doesn't help us when large result sets are required (e.g. DSS), or when many people across many nodes need to work with data on the same block (e.g. OLTP).

I too like the idea of restricting node access with services, but I would add that you have to be careful not to put all your eggs into one basket. If one node is responsible for all of your data loads, for instance, that node is now an increased risk. If it crashes, the GRD will be frozen until a surviving instance recovers all of the crashed node's recovery data, which could take longer if it has been performing the bulk of the DML work across the cluster.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 11, 2008 10:28 AM in response to:

Reply

>> But a well designed test does not become invalid because it is run on a reproducible environment

Yes, but it's valid ONLY for a single user environment, only on that specific server, disks, release, patch level, MBRC, and so on, ad infinitum.

Well Said!. Some how i was trying to express the same in my previous posts
in the same thread. I wonder how many DBA's listed here are "Scientific DBA"
or with some real world experience with corporate data. Well organized test case
can always be reproducible in Lab. This is true not only with Oracle, but with
any modern scientific experiments. Scientific Experts where able to simulate
some of the most complex experiments like nuclear fusion or earth rotation
or future hurricane prediction etc.. in their Lab. But does that mean, you blindly
depend on those and apply in the read time scenario? No, at least i don't think so.

What i wonder is, how many of those DBA listed here ever had taken a chance
to change their DSS application data block size and compare the Performance,
instead of testing with single user environment in their Lab and sleep on the result.
I think most of them stick with **"A != B since A is not equal to B"**

sp009

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 10:28 AM in response to:

Reply

>
> Greg, I've been consulting for decades, and I've
> NEVER has a client agree to the expense of
> re-designing their application. Not one!
>
> Is your experience different?
>

Mine is.

Company X (who cannot be named for reasons of NDA) has a project which had been running for 9 months when a new manager got worried about what was going on and called me in.

I got on site 9:00 am, and explained to IT director at lunchtime why it wasn't going to work and how he had to redesign the system. He thought about it for 5 minutes, then gave me a week to build a proof of concept. The week after that they started the re-design.

Reading your resume, by the way, it looks like you were what the British would call a contract DBA until 1999, and turned consultant in 2000. Maybe that's just a terminology thing - but I'd call that just over 8 years, not decades.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 11, 2008 10:36 AM in response to:

Reply

> >> it emphasises to me the need for test cases that
> can be executed on multiple releases, multiple MBRC,
> multiple servers etc..
>
> And for real-world workloads! If these "test cases"
> could be run in a multi-user mode and become
> respresentative of a real world system, then yes, a
> statistically valid sample size would indeed provide
> some valid evidence.

A real-world workload eventually, but you have to establish the basic behaviour first. Obviously you're going to monitor the impact of a change to the MBRC (for example) on your production workload, but you need those test cases prior to that to see the effect on query optimisation. When you do monitor for a change on the real world workload you look at AWR reports to see what went better and what went worse.

>
> **
> **
>
> >> Without that, you would have no way of knowing
> whether advice was valid for your circumstances.
>
> Excellent comment. It's only by generalizing the
> tests that value can be gained.

What do you mean by generalising the test?

Re: Larger vs. Small data block
Posted: Jun 11, 2008 10:55 AM in response to: David Aldridge
Reply

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:05 AM in response to: sp009
Reply

chris_c

Posts: 160
Registered: 10/17/06

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:07 AM in response to:
Reply

>>If I wanted to, I'm sure that I could concoct contrived test cases with offbeat parms to disprove practically every tip on MetaLink. . . .

Someone may have beaten you to it, both notes 77574.1 and 122008.1 appear to have been removed from metalink.

-- new note on index rebuilds (still a draft) 182699.1

Message was edited by:
chris_c

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:14 AM in response to:
Reply

(Snip)
> BTW, I agree with Greg that MBRC is also a factor,
> but for surprizing reasons.
>
> What I found on a database just this week, is that
> ditching the 10.2 MBRC=0 (automatic MBRC tuning) and
> using manual optimization, my client saw a 22%
> throughput improvement.
>
> But even stranger, this is a well-indexed OLTP app
> that does not do many scattered reads!
>
> The conventional wisdon suggests the multi-block read
> size is only for full-scan operations, but I found
> that optimizing MBRC is also important for optimizing
> inserts on reverse key indexes, and possible index
> range scans . . .
(Snip)

You stated:
*"What I found on a database just this week, is that ditching the 10.2 MBRC=0 (automatic MBRC tuning) and using manual optimization, my client saw a 22%
throughput improvement."*

Are you stating that your client disabled automatic tuning of the multi_block_read_count by setting the parameter to 0, and you did not tell the client that doing so actually sets the parameter's value to 1? Or, is this the correct way to disable automatic tuning of the multi_block_read_count?

Demonstration:

```
SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_file_multiblock_read_count        integer     128

SQL> ALTER SYSTEM SET DB_FILE_MULTIBLOCK_READ_COUNT=0 SCOPE=SPFILE;

System altered.

(Bounce Database Instance)

SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_file_multiblock_read_count        integer     1

SQL> ALTER SYSTEM RESET DB_FILE_MULTIBLOCK_READ_COUNT SCOPE=SPFILE SID='*';

System altered.

(Bounce Database Instance)

SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_file_multiblock_read_count        integer     128
```

Your client only saw a 22% thoughput performance by allowing more than one block to be read at a time? Maybe I just misunderstood what you stated?

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT was mistakenly clipped from the SQL*Plus output just before the line showed that the value was set to 1.
Message was edited by:
Charles Hooper

user641491

Posts: 1
Registered: 6/11/08

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:14 AM in response to: Faust

Reply

Faust, (and others who have signed up for StatspackAnalyzer.com):

The initial email sent with the URL to login to the StatspackAnalyzer.com tool has included a graphical tracking bit. This bit tells us one thing... that the email has been opened. It is our best way to verify that there is not something wrong with our email system and also to do a rough check to see if people are actually opening the emails we send out with the login. Most email systems will warn you anytime there are graphics/tracking bits in an email and give the user the option of opening the graphics. For example, I am aware that companies that send emails almost always use tracking bits. Sometimes this tracking is passive and sometimes the companies use it in ways that are disconcerting to me (as in they might email me and say I saw that you opened our email). If I want them to track my "email open", I will click ok to receive the graphics. If I do not want them to track my "email open", I will not open the graphics.

In any case, from our experience in the last year we can safely conclude that most people are opening these emails. Since we do not do anything else substantial with this tracking bit, we are working to remove this graphical tracking bit so that it will not cause concern to future StatspackAnalyzer.com users.

Finally, I encourage you to post concerns and rules improvement ideas on our StatspackAnalyzer.com forums. These forums list every rule including variables considered and recommendations made. We are hoping that this results in a lively dialog and that we can continually improve the tool.

And yes, as people frequently mention, the tool does recommend using solid state disks. I encourage you to ask any Oracle customers who contact us to discuss the fit of SSD with their application and you will see that we work closely with these customers to determine if the application actually needs our product or not. In support of this, we have a fleet of evaluation units that we send out for people to do free tests of our equipment. I can tell you that sometimes we help and sometimes we don't. Anyone in the Oracle community can tell you there are no silver bullets. Having said that, it is a nice bullet to have in your arsenal should you encounter a real I/O bottleneck.

We hope you will work with us to continue to improve this free tool.

Woody Hutsell
EVP
Texas Memory Systems
woody.h@ramsan.com

stevencallan

Posts: 1,409
Registered: 5/17/02

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:30 AM in response to: user641491

Reply

[diversion on]
Many email senders embed a 1x1 gif/image in an email, or more accurately, a link to a server/content provider such as akamai. When you open the email, a request is sent to download the image. That request is then ricocheted to the sender. Another "feature" is to embed a spinner which pings back and forth between you and the content provider. This is used to determine (roughly) how long the email was open. Another technique is to embed a spinner behind a link. You may not have clicked the link, but we know that your cursor was placed over it, so it is a rough indication that you may have been reading what was in that region of the email (think of product/ad placement). Which links you click and how many times you click them are also captured via ricochet.
[diversion off]

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:34 AM in response to: Hans Forbrich

Reply

2) I wonder which professional consulting companies or individual consultants have such a strong reputation that Oracle has invited them in to look at the performance for those internal Oracle apps. (Probably can't tell due to some NDA.)

I don't believe any external people have observed an internal Oracle system.

3) I wonder whether RWPG is asked to look at, or work on, the performance of those same apps.

Yes. The RWPG is part of development, not consulting, (we are not for hire or bill) and we often times are involved in looking at performance of internal databases, as well as the developers responsible for the problematic code area.

4) I wonder whether Oracle uses multiple block sizes in those apps for any reason other than transportable tablespace.

I believe that our internal applications use an 8k block.

5) With that kind of load available, and with the 11G Real App Testing feature and the data masking capability, I wonder whether Oracle did/will test against a sanitized real world data set based on that real world load.

One of the things that the RWPG tries to do is to add as many meaningful external workloads to our test suite. Generally these have come as a result of proof of concepts. I think there is a desire to try and gain some external workloads via the 11g RAT.

--
Regards,

Greg Rahn
http://structureddata.org

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:41 AM in response to:

Reply

> The advice below was quite good, and it's sad to see
> MetaLink remove tips like this, especially when the
> person complaining claims to know the truth but does
> not replace them with anything better.

I think you'd better decide whether you want high quality Metalink notes and documentation or not -- I'm getting a mixed message here ;)

It's up to Oracle support whether they want to rewrite the advice or remove it, of course. Obviously they agree with the criticism if they do.

gintsp

Posts: 1,639
From: Latvia, Riga
Registered: 9/30/99

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:43 AM in response to: stevencallan

Reply

That's why I always read my emails as plain text so that red fonts size +3 and sh|t you mentioned doesn't work for me even in outlook. At least I hope so...
And these embedded pictures become links and convert such emails to unreadable junk so I can easily press the del button

because of spam without any scruples.

Gints Plivna
http://www.gplivna.eu

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 11:49 AM in response to: Jonathan Lewis

Reply

> > Greg, I've been consulting for decades, and I've NEVER has a client agree to the expense of re-designing their application. Not one!
> >
> > Is your experience different?
> >
>
> Mine is.

Mine is as well. Tweaking and fiddling with parameters and blocks may offer **percentage** gains. Design modifications generally offer **magnitude** gains. Generally we are not talking complete redesign, but redesign of the problematic area.

--
Regards,

Greg Rahn
http://structureddata.org

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 11, 2008 12:03 PM in response to: Greg Rahn

Reply

Thanks Greg.

> **3) I wonder whether RWPG is asked to look at, or**
> **work on, the performance of those same apps.**
>
> Yes. The RWPG is part of development, not
> consulting, (we are not for hire or bill) and we
> often times are involved in looking at performance of
> internal databases, as well as the developers
> responsible for the problematic code area.
>
> **5) With that kind of load available, and with the**
> **11G Real App Testing feature and the data masking**
> **capability, I wonder whether Oracle did/will test**
> **against a sanitized real world data set based on that**
> **real world load.**
>
> One of the things that the RWPG tries to do is to add
> as many meaningful external workloads to our test
> suite. Generally these have come as a result of
> proof of concepts. I think there is a desire to try
> and gain some external workloads via the 11g RAT.
>

Which implies that you are actually using real world systems, and creating test cases that model specific aspects of the real world so that you can determine individual influences. And probably verifying that the models and influences thereon are actually valid in the real world.

Seems contradictory to some of the comments and implications alluded to by the representative of at least one popular consulting company.

(Also seems like the purpose has not changed much since the SPG days.)

> **4) I wonder whether Oracle uses multiple block**
> **sizes in those apps for any reason other than**
> **transportable tablespace.**
>
> I believe that our internal applications use an 8k block.

Any chance of getting that verified?

We all know that Larry, Charles and Jeff have limited patience with systems [performance]. If they are moderately satisfied with the real-world performance of Oracle's internal real-world systems, and if those systems use 8K blocks (or ... if one database uses only one block size), then I'd think that makes a significant statement in terms of this thread.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 12:15 PM in response to: user641491

Reply

Woody,

> The initial email sent with the URL to login to the
> StatspackAnalyzer.com tool has included a graphical
> tracking bit. This bit tells us one thing... that
> the email has been opened. It is our best way to
> verify that there is not something wrong with our
> email system and also to do a rough check to see if
> people are actually opening the emails we send out
> with the login.

Thank you for this posting. Apart from re-assuring your potential users, it's also captured the theme of thread in a microcosm.

a) Faust was correct in his observation that the email carried a trojan - but his degree of information (or interest) did not extend far enough to discover that the trojan was a harmless graphical tracking bit.

b) Steve Karam was correct in his observation that when he did his testing there were no trojans, because he didn't see a trojan. However, he may have failed to detect the "trojan" because he saw it, knew what it really was, and discounted it; or he may simply not have noticed.

c) Both of them were wrong, and careful testing would have shown this. Both could have claimed (and did) that their observations were valid because they were based on "empirical observations" of a "real-world system".

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 11, 2008 12:29 PM in response to: Jonathan Lewis

Reply

> b) Steve Karam was correct in his observation that
> when he did his testing there were no trojans,
> because he didn't see a trojan. However, he may have
> failed to detect the "trojan" because he saw it, knew
> what it really was, and discounted it; or he may
> simply not have noticed.

> c) Both of them were wrong, and careful testing would
> have shown this. Both could have claimed (and did)
> that their observations were valid because they were
> based on "empirical observations" of a "real-world
> system".

Except I never claimed that there were no trojans. Here are the statements I made.

Faust - TRUE
I signed up for StatspackAnalyzer.com to check out your claim - TRUE
I tried it three times - TRUE
Each time I received an email with no attachments at all. - TRUE
The first test was sent to my mailserver which I own, and it passed through an up-to-date Spamassassin/Clam filter to Outlook 2007 on my computer running Avast! Anti-Virus with fully up-to-date definitions. - TRUE
Next it was sent twice to my enterprise IMAP account through GMail, - TRUE
which has outstanding virus/spam protection. - TRUE
GMail also delivered it to my inbox and reported no issues. - TRUE
I'd love to see it, - TRUE
but I can't find any evidence that backs up your claim. - TRUE

If one says they cannot find evidence, and then evidence is found, they are not wrong. They are still absolutely correct that they could not find evidence. I am glad someone with the facts, which I did not have, was able to come in and give a definitive answer. Had I pretended my tests were fact and said "there are no trojans" you're right, I would have been wrong.

Whereas you assert that "Both of them were wrong." Which is false.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 11, 2008 12:37 PM in response to: Hans Forbrich

Reply

> We all know that Larry, Charles and Jeff have limited
> patience with systems [performance]. If they are
> moderately satisfied with the real-world performance
> of Oracle's internal real-world systems, and if those
> systems use 8K blocks (or ... if one database uses
> only one block size), then I'd think that makes a
> significant statement in terms of this thread.

Hans,

I think you too ignore the actual debate on this thread. Let me remind you,
"Does higher db_block_size perform better in DW applications?".
I think you, with many, are too much eager to claim victory, rather then
presenting your test case to back the claim.

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 11, 2008 12:44 PM in response to: sp009

Reply

> > We all know that Larry, Charles and Jeff have limited
> > patience with systems [performance]. If they are
> > moderately satisfied with the real-world performance
> > of Oracle's internal real-world systems, and if those
> > systems use 8K blocks (or ... if one database uses
> > only one block size), then I'd think that makes a
> > significant statement in terms of this thread.
>
> Hans,
>
> I think you too ignore the actual debate on this
> thread. Let me remind you,
> "Does higher db_block_size perform better in DW
> applications?".
> I think you, with many, are too much eager to claim
> victory, rather then
> presenting your test case to back the claim.

I think you are too quick to ignore the fact that Oracle
internal systems include Data Warehouses and that Oracle's
decision as for block size being used on their systems is
also a real world test case that provides input to your
question.

Personally I don't need to claim victory. Indeed, I did
not even imply that Oracle's experience would provide
a conclusion. I did state that it would provide a
significant statement - meaning that it provides real
world input to your question.

I think you are very quick in trying to accept things that
support your idea and very quick to dismiss things that
seem to contradict what you wish to prove. At least that
is the feeling that comes from your last comment to me.

Message was edited by: Hans Forbrich

As I am on record as writing, and saying to my students
and my customers: with Oracle, the only conclusive
benchmark or answer is one that you have tested in your
environment. And even that might change with any patch or
change in environment.

The only thing I can bring to the table is experience that
might shortcut the time to complete that benchmark. And
I constantly get new experience.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 12:46 PM in response to: Steve Karam

Reply

> > b) Steve Karam was correct in his observation that
> > when he did his testing there were no trojans,
> > because he didn't see a trojan. However, he may
>> have failed to detect the "trojan" because he saw it,
>> knew what it really was, and discounted it; or he may
> > simply not have noticed.
>
> > c) Both of them were wrong, and careful testing would
> > have shown this. Both could have claimed (and did)
> > that their observations were valid because they were
> > based on "empirical observations" of a "real-world
> > system".
>

>
> If one says they cannot find evidence, and then
> evidence is found, they are not wrong.

Steve,

My apologies, I had not intended to hurt your feelings - let me shorten and rephrase my comments:

a) Faust observed evidence of a trojan

b) Steve observed no evidence of a trojan

c) Both were arguably wrong - Faust because the trojan could be labelled harmless, and Steve because there was a trojan, but it had not been flagged as such by any of the mechanisms that he had used.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 12:49 PM in response to:

Reply

> >> It would be advantageous to see some AWR or Statspack of before and after.
>
> I agree.

> How about some "real world" case studies on this issue.
> Any hope for "real world" reports, from real shops?

I have not recommended nor implemented a block size change so I have no data to share. This is why I've asked you to publish yours. If I had some to share, I would.

> Like I said, it's Oracle's job to educate us, not the other way around!

So, if I understand correctly, you want me to educate & share (because I happen to work for Oracle), but you do not want to reciprocate? Why are you hiding behind that argument? Are you looking for excuses?

> Why all of these artificial tests from a "real world" group?

Perhaps you have a misunderstanding of the context “real-world” here. Its meaning is related understanding how the stack (software & hardware) works as it is used on a day-to-day basis by customers. The RWPG has a broader focus on performance than say the TPC groups who have been working on the same benchmarks for say 10 years or so. Both groups contribute to database performance, but in different ways. TPC test things at extreme, but the amount of code path and features involved by the test is much less than a customer would use. RWPG brings back experience from customer benchmarks and performance challenges.

I would not call them “artificial”; they are *focused* tests. Let’s take the topic here: block size and alleged performance gains. If someone reports an observed gain by changing the block size, my immediate question is “Why?”. Is it related solely to the block size or did changing the block size in tern cause other changes that are not being recognized (like a change in execution plans)? So I take my knowledge of the software and ask: “At what levels (areas in the code) could block size (alone) potentially alter performance?” So I make a list (and ask around development to verify), and construct some focused test cases to provide metrics that will provide data. The specific areas that I ran my experiments were:
- One table FTS (100% physical reads & 100% buffer cache reads)
- FTS with 2 table Hash Join (100% physical reads & 100% buffer cache reads)
- Single Table Index FFS (100% physical reads & 100% buffer cache reads)
- I’ve even demonstrated that there is no performance advantage for a FTS with 1MB MBRC reads with any block size between 2k and 16k

This is by no means a comprehensive list but it is a start. It focuses on two key areas: physical I/O and buffer cache I/O, probably two of the most influential areas when it comes to block size and query response time (assuming all others constant). And I have yet to observe any evidence to support that block size matters.

Now, I have also offered my interpretation of the data. For example: when MBRC is left unset, a FTS will read 1MB or data, regardless of the block size. I offered my grab-from-the-coin-bucket explanation of why.

Since neither you or sp009 or have provided your supporting data, or any test cases or experiments, I don’t have much option than to try and create it.

sp009: I spoke with the editor-in-chief of Oracle Magazine and he has not seen a proposal for the paper the “Guru” you are working with mentioned. Be sure to have him submit it.

--
Regards,

Greg Rahn
http://structureddata.org

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 11, 2008 1:05 PM in response to: Jonathan Lewis

Reply

> My apologies, I had not intended to hurt your
> feelings

It's okay, you couldn't hurt my feelings. I just wanted to make sure we get all the facts straight before we toss out the word 'wrong'.

Which could arguably be a microcosm of this thread, as well.

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 1:45 PM in response to: Hans Forbrich

> Which implies that you are actually using real world systems, and creating test cases that model specific
> aspects of the real world so that you can determine individual influences. And probably verifying that
> the models and influences thereon are actually valid in the real world.
>
> Seems contradictory to some of the comments and implications alluded to by the representative of at
> least one popular consulting company.

You are correct. We do interact with real-world systems (both customer and Oracle internal) and when we observe a phenomenon at a cu
site, we do analysis to understand why, and are almost always successful in reproducing it in-house. This allows bug fixes or change
effective to the specific problem. Of course, this is verified by applying the fix/change to the internal test case as well as to th
site.

Do we bring 100% of a customer’s data and workload in? Almost always never. In almost every case an issue can be modeled and simplif
is understood. Understanding is generally just a matter of gathering enough data points.

> > **4) I wonder whether Oracle uses multiple block sizes in those apps for any reason other than transportable tablespace.**
> >
> > I believe that our internal applications use an 8k block.
>
> Any chance of getting that verified?

I just confirmed with someone who works frequently with those systems and they are not aware of any use of any other block size than

--
Regards,

Greg Rahn
http://structureddata.org

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 11, 2008 1:57 PM in response to: Charles Hooper

> (Snip)
> > BTW, I agree with Greg that MBRC is also a factor,
> > but for surprizing reasons.
> >
> > What I found on a database just this week, is that
> > ditching the 10.2 MBRC=0 (automatic MBRC tuning)
> and
> > using manual optimization, my client saw a 22%
> > throughput improvement.
> >
> > But even stranger, this is a well-indexed OLTP app
> > that does not do many scattered reads!
> >
> > The conventional wisdon suggests the multi-block
> read
> > size is only for full-scan operations, but I found
> > that optimizing MBRC is also important for
> optimizing
> > inserts on reverse key indexes, and possible index
> > range scans . . .
> (Snip)
>
> You stated:
> *"What I found on a database just this week, is*
> *that ditching the 10.2 MBRC=0 (automatic MBRC tuning)*
> *and using manual optimization, my client saw a 22%*
> *throughput improvement."*
>
> Are you stating that your client disabled automatic
> tuning of the multi_block_read_count by setting the
> parameter to 0, and you did not tell the client that
> doing so actually sets the parameter's value to 1?
> Or, is this the correct way to disable automatic
> tuning of the multi_block_read_count?
>
> Demonstration:
>

```
> SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT
>
> AME                                  TYPE
>        VALUE
> ---------------------------- ----------- -----
> db_file_multiblock_read_count        integer     128
>
> QL> ALTER SYSTEM SET DB_FILE_MULTIBLOCK_READ_COUNT=0
> SCOPE=SPFILE;
>
> ystem altered.
```

```
>
> Bounce Database Instance)
>
> QL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT
>
> AME                                    TYPE
>        VALUE
> ----------------------------- ----------- -----
> db_file_multiblock_read_count        integer     1
>
> QL> ALTER SYSTEM RESET DB_FILE_MULTIBLOCK_READ_COUNT
> SCOPE=SPFILE SID='*';
>
> ystem altered.
>
> Bounce Database Instance)
>
> QL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT
>
> AME                                    TYPE
>        VALUE
> ----------------------------- ----------- -----
> db_file_multiblock_read_count        integer     128
>

>
> Your client only saw a 22% thoughput performance by
> allowing more than one block to be read at a time?
>  Maybe I just misunderstood what you stated?
>
> Charles Hooper
> IT Manager/Oracle DBA
> K&M Machine-Fabricating, Inc.
>
> SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT was
> mistakenly clipped from the SQL*Plus output just
> before the line showed that the value was set to 1.
> Message was edited by:
>         Charles Hooper
```

Not my call, but i would like you to have a look at
https://metalink.oracle.com/metalink/plsql/f?p=200:27:1190037021398714647::::p27_id,p27_show_header,p27_show_help:714075.993,1,1

jgarry

Posts: 128
From: Just outside of beautiful Vista, California
Registered: 7/20/98

Re: Larger vs. Small data block
Posted: Jun 11, 2008 1:59 PM in response to:

Reply

> I've been consulting for decades, and I've NEVER has a client agree to the expense
>of re-designing their application. Not one!

>Is your experience different?

Yes. It depends on what particular problem they've bought you in to fix. If you have some penny-wise pound-foolish manager who has been sold on quick-fixes, you aren't likely to see it. If you have a client with a more strategic vision, they are much more open to actually making a reasonable decision. If your business model is based on the former, of course you'd have your experience. That doesn't make your clients smart, it merely shows the value of marketing. It doesn't mean anything bad about you either, unless you start advocating that is how it generally **should** be. If you do, you are having serious confusion between tactical and strategic decisions.

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 11, 2008 2:37 PM in response to: sp009

Reply

> Not my call, but i would like you to have a look at
> https://metalink.oracle.com/metalink/plsql/f?p=200:27:
> 1190037021398714647::::p27_id,p27_show_header,p27_show
> _help:714075.993,1,1

sp009,

Thanks for the link. I did quickly look at the Metalink article. Here is a quick test with SQL*Plus output, since I try to verify in order to understand what I read:

```
SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_file_multiblock_read_count        integer     128

SQL> CREATE TABLE T1 AS
  2  SELECT
  3    ROWNUM RN,
  4    TRUNC(SYSDATE)+ROWNUM-3000 DT,
  5    A.*
  6  FROM
  7    ALL_OBJECTS A
  8  WHERE
  9    ROWNUM<=10000;

Table created.

SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T1');

PL/SQL procedure successfully completed.

SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;

System altered.

SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 8';

Session altered.

SQL> SELECT
```

```
  2    COUNT(*)
  3  FROM
  4    T1;

  COUNT(*)
----------
     10000

SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF';


From the trace file:
PARSING IN CURSOR #4 len=23 dep=0 uid=429 oct=3 lid=429 tim=989014176 hv=2807425063 ad='50d1b03c'
SELECT
  COUNT(*)
FROM
  T1
END OF STMT
PARSE #4:c=0,e=1569,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,tim=989014166
EXEC #4:c=0,e=159,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=989015768
WAIT #4: nam='SQL*Net message to client' ela= 7 driver id=1413697536 #bytes=1 p3=0 obj#=386 tim=989016181
WAIT #4: nam='db file sequential read' ela= 408 file#=4 block#=1101955 blocks=1 obj#=42089 tim=989017024
WAIT #4: nam='db file scattered read' ela= 703 file#=4 block#=1101956 blocks=5 obj#=42089 tim=989018284
WAIT #4: nam='db file scattered read' ela= 966 file#=4 block#=1102217 blocks=8 obj#=42089 tim=989019972
WAIT #4: nam='db file scattered read' ela= 858 file#=4 block#=1102226 blocks=7 obj#=42089 tim=989021687
WAIT #4: nam='db file scattered read' ela= 914 file#=4 block#=1102233 blocks=8 obj#=42089 tim=989023418
WAIT #4: nam='db file scattered read' ela= 851 file#=4 block#=1102242 blocks=7 obj#=42089 tim=989025099
WAIT #4: nam='db file scattered read' ela= 958 file#=4 block#=1102249 blocks=8 obj#=42089 tim=989026863
WAIT #4: nam='db file scattered read' ela= 847 file#=4 block#=1102258 blocks=7 obj#=42089 tim=989028521
WAIT #4: nam='db file scattered read' ela= 945 file#=4 block#=1102265 blocks=8 obj#=42089 tim=989030214
WAIT #4: nam='db file scattered read' ela= 867 file#=4 block#=1102274 blocks=7 obj#=42089 tim=989032052
WAIT #4: nam='db file scattered read' ela= 931 file#=4 block#=1102281 blocks=8 obj#=42089 tim=989033709
WAIT #4: nam='db file scattered read' ela= 834 file#=4 block#=1102290 blocks=7 obj#=42089 tim=989034996
WAIT #4: nam='db file scattered read' ela= 873 file#=4 block#=1102297 blocks=8 obj#=42089 tim=989036258
WAIT #4: nam='db file scattered read' ela= 863 file#=4 block#=1102306 blocks=7 obj#=42089 tim=989037554
WAIT #4: nam='db file scattered read' ela= 874 file#=4 block#=1102313 blocks=8 obj#=42089 tim=989038817
WAIT #4: nam='db file scattered read' ela= 816 file#=4 block#=1102322 blocks=7 obj#=42089 tim=989040051
WAIT #4: nam='db file scattered read' ela= 895 file#=4 block#=1104633 blocks=8 obj#=42089 tim=989041344
WAIT #4: nam='db file scattered read' ela= 15381 file#=4 block#=1533579 blocks=24 obj#=42089 tim=989057218
FETCH #4:c=15625,e=41844,p=143,cr=146,cu=0,mis=0,r=1,dep=0,og=1,tim=989058387
WAIT #4: nam='SQL*Net message from client' ela= 662 driver id=1413697536 #bytes=1 p3=0 obj#=42089 tim=989059234
FETCH #4:c=0,e=3,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,tim=989059324
WAIT #4: nam='SQL*Net message to client' ela= 3 driver id=1413697536 #bytes=1 p3=0 obj#=42089 tim=989059380
WAIT #4: nam='SQL*Net message from client' ela= 834 driver id=1413697536 #bytes=1 p3=0 obj#=42089 tim=989060260
STAT #4 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT AGGREGATE (cr=146 pr=143 pw=0 time=41842 us)'
STAT #4 id=2 cnt=10000 pid=1 pos=1 obj=42089 op='TABLE ACCESS FULL T1 (cr=146 pr=143 pw=0 time=22164 us)'

-----

SQL> ALTER SYSTEM SET DB_FILE_MULTIBLOCK_READ_COUNT=0 SCOPE=SPFILE;

System altered.

(Bounce Database Instance)

SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_file_multiblock_read_count        integer     1

SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 8';

Session altered.

SQL> SELECT
  2    COUNT(*)
  3  FROM
  4    T1;

  COUNT(*)
----------
     10000

SQL> ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF';


From the trace file:
PARSING IN CURSOR #1 len=23 dep=0 uid=429 oct=3 lid=429 tim=1341599911 hv=2807425063 ad='50ddccb4'
SELECT
  COUNT(*)
FROM
  T1
END OF STMT
PARSE #1:c=218750,e=284159,p=38,cr=699,cu=0,mis=1,r=0,dep=0,og=1,tim=1341599899
EXEC #1:c=0,e=116,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1341600174
WAIT #1: nam='SQL*Net message to client' ela= 6 driver id=1413697536 #bytes=1 p3=0 obj#=10192 tim=1341600243
WAIT #1: nam='db file sequential read' ela= 368 file#=4 block#=1101955 blocks=1 obj#=42089 tim=1341601137
WAIT #1: nam='db file sequential read' ela= 338 file#=4 block#=1101956 blocks=1 obj#=42089 tim=1341601663
WAIT #1: nam='db file sequential read' ela= 316 file#=4 block#=1101957 blocks=1 obj#=42089 tim=1341602088
WAIT #1: nam='db file sequential read' ela= 318 file#=4 block#=1101958 blocks=1 obj#=42089 tim=1341602509
WAIT #1: nam='db file sequential read' ela= 321 file#=4 block#=1101959 blocks=1 obj#=42089 tim=1341602912
WAIT #1: nam='db file sequential read' ela= 375 file#=4 block#=1101960 blocks=1 obj#=42089 tim=1341603395
WAIT #1: nam='db file sequential read' ela= 422 file#=4 block#=1102217 blocks=1 obj#=42089 tim=1341603911
WAIT #1: nam='db file sequential read' ela= 363 file#=4 block#=1102218 blocks=1 obj#=42089 tim=1341604365
WAIT #1: nam='db file sequential read' ela= 366 file#=4 block#=1102219 blocks=1 obj#=42089 tim=1341604818
WAIT #1: nam='db file sequential read' ela= 358 file#=4 block#=1102220 blocks=1 obj#=42089 tim=1341605254
WAIT #1: nam='db file sequential read' ela= 356 file#=4 block#=1102221 blocks=1 obj#=42089 tim=1341605702
WAIT #1: nam='db file sequential read' ela= 364 file#=4 block#=1102222 blocks=1 obj#=42089 tim=1341606168
WAIT #1: nam='db file sequential read' ela= 352 file#=4 block#=1102223 blocks=1 obj#=42089 tim=1341606611
WAIT #1: nam='db file sequential read' ela= 387 file#=4 block#=1102224 blocks=1 obj#=42089 tim=1341607117
WAIT #1: nam='db file sequential read' ela= 341 file#=4 block#=1102226 blocks=1 obj#=42089 tim=1341607551
WAIT #1: nam='db file sequential read' ela= 493 file#=4 block#=1102227 blocks=1 obj#=42089 tim=1341608128
WAIT #1: nam='db file sequential read' ela= 331 file#=4 block#=1102228 blocks=1 obj#=42089 tim=1341608539
WAIT #1: nam='db file sequential read' ela= 332 file#=4 block#=1102229 blocks=1 obj#=42089 tim=1341608953
WAIT #1: nam='db file sequential read' ela= 328 file#=4 block#=1102230 blocks=1 obj#=42089 tim=1341609360
WAIT #1: nam='db file sequential read' ela= 343 file#=4 block#=1102231 blocks=1 obj#=42089 tim=1341609783
WAIT #1: nam='db file sequential read' ela= 367 file#=4 block#=1102232 blocks=1 obj#=42089 tim=1341610241
WAIT #1: nam='db file sequential read' ela= 341 file#=4 block#=1102233 blocks=1 obj#=42089 tim=1341610700
WAIT #1: nam='db file sequential read' ela= 338 file#=4 block#=1102234 blocks=1 obj#=42089 tim=1341611122
WAIT #1: nam='db file sequential read' ela= 337 file#=4 block#=1102235 blocks=1 obj#=42089 tim=1341611543
WAIT #1: nam='db file sequential read' ela= 331 file#=4 block#=1102236 blocks=1 obj#=42089 tim=1341611956
WAIT #1: nam='db file sequential read' ela= 342 file#=4 block#=1102237 blocks=1 obj#=42089 tim=1341612378
WAIT #1: nam='db file sequential read' ela= 3177 file#=4 block#=1102238 blocks=1 obj#=42089 tim=1341615637
WAIT #1: nam='db file sequential read' ela= 2676 file#=4 block#=1102239 blocks=1 obj#=42089 tim=1341618391
WAIT #1: nam='db file sequential read' ela= 1815 file#=4 block#=1102240 blocks=1 obj#=42089 tim=1341620299
WAIT #1: nam='db file sequential read' ela= 6302 file#=4 block#=1102242 blocks=1 obj#=42089 tim=1341626686
WAIT #1: nam='db file sequential read' ela= 5773 file#=4 block#=1102243 blocks=1 obj#=42089 tim=1341632553
WAIT #1: nam='db file sequential read' ela= 828 file#=4 block#=1102244 blocks=1 obj#=42089 tim=1341633462
WAIT #1: nam='db file sequential read' ela= 997 file#=4 block#=1102245 blocks=1 obj#=42089 tim=1341634541
WAIT #1: nam='db file sequential read' ela= 1298 file#=4 block#=1102246 blocks=1 obj#=42089 tim=1341635922
WAIT #1: nam='db file sequential read' ela= 1118 file#=4 block#=1102247 blocks=1 obj#=42089 tim=1341637120
WAIT #1: nam='db file sequential read' ela= 504 file#=4 block#=1102248 blocks=1 obj#=42089 tim=1341637720
WAIT #1: nam='db file sequential read' ela= 488 file#=4 block#=1102249 blocks=1 obj#=42089 tim=1341638292
WAIT #1: nam='db file sequential read' ela= 489 file#=4 block#=1102250 blocks=1 obj#=42089 tim=1341638865
WAIT #1: nam='db file sequential read' ela= 490 file#=4 block#=1102251 blocks=1 obj#=42089 tim=1341639434
WAIT #1: nam='db file sequential read' ela= 489 file#=4 block#=1102252 blocks=1 obj#=42089 tim=1341640005
WAIT #1: nam='db file sequential read' ela= 1417 file#=4 block#=1102253 blocks=1 obj#=42089 tim=1341641509
WAIT #1: nam='db file sequential read' ela= 488 file#=4 block#=1102254 blocks=1 obj#=42089 tim=1341642079
WAIT #1: nam='db file sequential read' ela= 7312 file#=4 block#=1102255 blocks=1 obj#=42089 tim=1341649495
WAIT #1: nam='db file sequential read' ela= 1186 file#=4 block#=1102256 blocks=1 obj#=42089 tim=1341650787
WAIT #1: nam='db file sequential read' ela= 445 file#=4 block#=1102258 blocks=1 obj#=42089 tim=1341651364
WAIT #1: nam='db file sequential read' ela= 479 file#=4 block#=1102259 blocks=1 obj#=42089 tim=1341651933
WAIT #1: nam='db file sequential read' ela= 492 file#=4 block#=1102260 blocks=1 obj#=42089 tim=1341652505
WAIT #1: nam='db file sequential read' ela= 490 file#=4 block#=1102261 blocks=1 obj#=42089 tim=1341653076
WAIT #1: nam='db file sequential read' ela= 1265 file#=4 block#=1102262 blocks=1 obj#=42089 tim=1341654419
WAIT #1: nam='db file sequential read' ela= 499 file#=4 block#=1102263 blocks=1 obj#=42089 tim=1341655003
WAIT #1: nam='db file sequential read' ela= 512 file#=4 block#=1102264 blocks=1 obj#=42089 tim=1341655617
WAIT #1: nam='db file sequential read' ela= 490 file#=4 block#=1102265 blocks=1 obj#=42089 tim=1341656196
WAIT #1: nam='db file sequential read' ela= 494 file#=4 block#=1102266 blocks=1 obj#=42089 tim=1341656769
WAIT #1: nam='db file sequential read' ela= 489 file#=4 block#=1102267 blocks=1 obj#=42089 tim=1341657339
WAIT #1: nam='db file sequential read' ela= 487 file#=4 block#=1102268 blocks=1 obj#=42089 tim=1341657909
WAIT #1: nam='db file sequential read' ela= 490 file#=4 block#=1102269 blocks=1 obj#=42089 tim=1341658479
WAIT #1: nam='db file sequential read' ela= 485 file#=4 block#=1102270 blocks=1 obj#=42089 tim=1341659048
WAIT #1: nam='db file sequential read' ela= 490 file#=4 block#=1102271 blocks=1 obj#=42089 tim=1341659615
WAIT #1: nam='db file sequential read' ela= 1473 file#=4 block#=1102272 blocks=1 obj#=42089 tim=1341661181
WAIT #1: nam='db file sequential read' ela= 510 file#=4 block#=1102274 blocks=1 obj#=42089 tim=1341661772
WAIT #1: nam='db file sequential read' ela= 503 file#=4 block#=1102275 blocks=1 obj#=42089 tim=1341662358
WAIT #1: nam='db file sequential read' ela= 501 file#=4 block#=1102276 blocks=1 obj#=42089 tim=1341662944
WAIT #1: nam='db file sequential read' ela= 495 file#=4 block#=1102277 blocks=1 obj#=42089 tim=1341663530
WAIT #1: nam='db file sequential read' ela= 501 file#=4 block#=1102278 blocks=1 obj#=42089 tim=1341664115
WAIT #1: nam='db file sequential read' ela= 502 file#=4 block#=1102279 blocks=1 obj#=42089 tim=1341664696
WAIT #1: nam='db file sequential read' ela= 520 file#=4 block#=1102280 blocks=1 obj#=42089 tim=1341665311
WAIT #1: nam='db file sequential read' ela= 1473 file#=4 block#=1102281 blocks=1 obj#=42089 tim=1341666867
WAIT #1: nam='db file sequential read' ela= 503 file#=4 block#=1102282 blocks=1 obj#=42089 tim=1341667454
WAIT #1: nam='db file sequential read' ela= 503 file#=4 block#=1102283 blocks=1 obj#=42089 tim=1341668037
WAIT #1: nam='db file sequential read' ela= 502 file#=4 block#=1102284 blocks=1 obj#=42089 tim=1341668621
WAIT #1: nam='db file sequential read' ela= 504 file#=4 block#=1102285 blocks=1 obj#=42089 tim=1341669204
WAIT #1: nam='db file sequential read' ela= 498 file#=4 block#=1102286 blocks=1 obj#=42089 tim=1341669784
WAIT #1: nam='db file sequential read' ela= 501 file#=4 block#=1102287 blocks=1 obj#=42089 tim=1341670365
WAIT #1: nam='db file sequential read' ela= 526 file#=4 block#=1102288 blocks=1 obj#=42089 tim=1341670979
WAIT #1: nam='db file sequential read' ela= 1495 file#=4 block#=1102290 blocks=1 obj#=42089 tim=1341672570
WAIT #1: nam='db file sequential read' ela= 496 file#=4 block#=1102291 blocks=1 obj#=42089 tim=1341673146
WAIT #1: nam='db file sequential read' ela= 492 file#=4 block#=1102292 blocks=1 obj#=42089 tim=1341673721
WAIT #1: nam='db file sequential read' ela= 498 file#=4 block#=1102293 blocks=1 obj#=42089 tim=1341674297
WAIT #1: nam='db file sequential read' ela= 493 file#=4 block#=1102294 blocks=1 obj#=42089 tim=1341674871
WAIT #1: nam='db file sequential read' ela= 497 file#=4 block#=1102295 blocks=1 obj#=42089 tim=1341675446
WAIT #1: nam='db file sequential read' ela= 495 file#=4 block#=1102296 blocks=1 obj#=42089 tim=1341676052
WAIT #1: nam='db file sequential read' ela= 489 file#=4 block#=1102297 blocks=1 obj#=42089 tim=1341676630
WAIT #1: nam='db file sequential read' ela= 1406 file#=4 block#=1102298 blocks=1 obj#=42089 tim=1341678131
WAIT #1: nam='db file sequential read' ela= 479 file#=4 block#=1102299 blocks=1 obj#=42089 tim=1341678694
WAIT #1: nam='db file sequential read' ela= 486 file#=4 block#=1102300 blocks=1 obj#=42089 tim=1341679259
WAIT #1: nam='db file sequential read' ela= 481 file#=4 block#=1102301 blocks=1 obj#=42089 tim=1341679824
WAIT #1: nam='db file sequential read' ela= 486 file#=4 block#=1102302 blocks=1 obj#=42089 tim=1341680389
WAIT #1: nam='db file sequential read' ela= 482 file#=4 block#=1102303 blocks=1 obj#=42089 tim=1341680953
WAIT #1: nam='db file sequential read' ela= 508 file#=4 block#=1102304 blocks=1 obj#=42089 tim=1341681548
WAIT #1: nam='db file sequential read' ela= 479 file#=4 block#=1102306 blocks=1 obj#=42089 tim=1341682112
WAIT #1: nam='db file sequential read' ela= 1371 file#=4 block#=1102307 blocks=1 obj#=42089 tim=1341683566
WAIT #1: nam='db file sequential read' ela= 479 file#=4 block#=1102308 blocks=1 obj#=42089 tim=1341684128
WAIT #1: nam='db file sequential read' ela= 478 file#=4 block#=1102309 blocks=1 obj#=42089 tim=1341684688
WAIT #1: nam='db file sequential read' ela= 479 file#=4 block#=1102310 blocks=1 obj#=42089 tim=1341685248
WAIT #1: nam='db file sequential read' ela= 470 file#=4 block#=1102311 blocks=1 obj#=42089 tim=1341685810
WAIT #1: nam='db file sequential read' ela= 503 file#=4 block#=1102312 blocks=1 obj#=42089 tim=1341686406
WAIT #1: nam='db file sequential read' ela= 5084 file#=4 block#=1102313 blocks=1 obj#=42089 tim=1341691577
WAIT #1: nam='db file sequential read' ela= 350 file#=4 block#=1102314 blocks=1 obj#=42089 tim=1341692008
WAIT #1: nam='db file sequential read' ela= 9164 file#=4 block#=1102315 blocks=1 obj#=42089 tim=1341701254
WAIT #1: nam='db file sequential read' ela= 489 file#=4 block#=1102316 blocks=1 obj#=42089 tim=1341701830
WAIT #1: nam='db file sequential read' ela= 494 file#=4 block#=1102317 blocks=1 obj#=42089 tim=1341702406
WAIT #1: nam='db file sequential read' ela= 505 file#=4 block#=1102318 blocks=1 obj#=42089 tim=1341702995
WAIT #1: nam='db file sequential read' ela= 1182 file#=4 block#=1102319 blocks=1 obj#=42089 tim=1341704257
WAIT #1: nam='db file sequential read' ela= 1825 file#=4 block#=1102320 blocks=1 obj#=42089 tim=1341706172
WAIT #1: nam='db file sequential read' ela= 493 file#=4 block#=1102322 blocks=1 obj#=42089 tim=1341706748
WAIT #1: nam='db file sequential read' ela= 791 file#=4 block#=1102323 blocks=1 obj#=42089 tim=1341707622
WAIT #1: nam='db file sequential read' ela= 1093 file#=4 block#=1102324 blocks=1 obj#=42089 tim=1341708793
WAIT #1: nam='db file sequential read' ela= 470 file#=4 block#=1102325 blocks=1 obj#=42089 tim=1341709345
WAIT #1: nam='db file sequential read' ela= 468 file#=4 block#=1102326 blocks=1 obj#=42089 tim=1341709892
WAIT #1: nam='db file sequential read' ela= 464 file#=4 block#=1102327 blocks=1 obj#=42089 tim=1341710439
WAIT #1: nam='db file sequential read' ela= 3975 file#=4 block#=1102328 blocks=1 obj#=42089 tim=1341714504
WAIT #1: nam='db file sequential read' ela= 1808 file#=4 block#=1104633 blocks=1 obj#=42089 tim=1341716399
WAIT #1: nam='db file sequential read' ela= 6841 file#=4 block#=1104634 blocks=1 obj#=42089 tim=1341723318
WAIT #1: nam='db file sequential read' ela= 1242 file#=4 block#=1104635 blocks=1 obj#=42089 tim=1341724642
WAIT #1: nam='db file sequential read' ela= 499 file#=4 block#=1104636 blocks=1 obj#=42089 tim=1341725222
WAIT #1: nam='db file sequential read' ela= 506 file#=4 block#=1104637 blocks=1 obj#=42089 tim=1341725806
WAIT #1: nam='db file sequential read' ela= 496 file#=4 block#=1104638 blocks=1 obj#=42089 tim=1341726387
WAIT #1: nam='db file sequential read' ela= 499 file#=4 block#=1104639 blocks=1 obj#=42089 tim=1341726966
WAIT #1: nam='db file sequential read' ela= 13127 file#=4 block#=1104640 blocks=1 obj#=42089 tim=1341740204
WAIT #1: nam='db file sequential read' ela= 4825 file#=4 block#=1533579 blocks=1 obj#=42089 tim=1341745154
WAIT #1: nam='db file sequential read' ela= 4883 file#=4 block#=1533580 blocks=1 obj#=42089 tim=1341750130
WAIT #1: nam='db file sequential read' ela= 1220 file#=4 block#=1533581 blocks=1 obj#=42089 tim=1341751433
WAIT #1: nam='db file sequential read' ela= 4578 file#=4 block#=1533582 blocks=1 obj#=42089 tim=1341756105
WAIT #1: nam='db file sequential read' ela= 5921 file#=4 block#=1533583 blocks=1 obj#=42089 tim=1341762105
WAIT #1: nam='db file sequential read' ela= 732 file#=4 block#=1533584 blocks=1 obj#=42089 tim=1341762963
WAIT #1: nam='db file sequential read' ela= 794 file#=4 block#=1533585 blocks=1 obj#=42089 tim=1341763840
WAIT #1: nam='db file sequential read' ela= 771 file#=4 block#=1533586 blocks=1 obj#=42089 tim=1341764694
WAIT #1: nam='db file sequential read' ela= 781 file#=4 block#=1533587 blocks=1 obj#=42089 tim=1341765556
WAIT #1: nam='db file sequential read' ela= 12824 file#=4 block#=1533588 blocks=1 obj#=42089 tim=1341778460
WAIT #1: nam='db file sequential read' ela= 815 file#=4 block#=1533589 blocks=1 obj#=42089 tim=1341779359
WAIT #1: nam='db file sequential read' ela= 4413 file#=4 block#=1533590 blocks=1 obj#=42089 tim=1341783858
WAIT #1: nam='db file sequential read' ela= 6167 file#=4 block#=1533591 blocks=1 obj#=42089 tim=1341790110
WAIT #1: nam='db file sequential read' ela= 478 file#=4 block#=1533592 blocks=1 obj#=42089 tim=1341790669
WAIT #1: nam='db file sequential read' ela= 476 file#=4 block#=1533593 blocks=1 obj#=42089 tim=1341791229
WAIT #1: nam='db file sequential read' ela= 477 file#=4 block#=1533594 blocks=1 obj#=42089 tim=1341791785
WAIT #1: nam='db file sequential read' ela= 6571 file#=4 block#=1533595 blocks=1 obj#=42089 tim=1341798437
WAIT #1: nam='db file sequential read' ela= 1241 file#=4 block#=1533596 blocks=1 obj#=42089 tim=1341799756
WAIT #1: nam='db file sequential read' ela= 485 file#=4 block#=1533597 blocks=1 obj#=42089 tim=1341800324
WAIT #1: nam='db file sequential read' ela= 489 file#=4 block#=1533598 blocks=1 obj#=42089 tim=1341800891
WAIT #1: nam='db file sequential read' ela= 484 file#=4 block#=1533599 blocks=1 obj#=42089 tim=1341801458
WAIT #1: nam='db file sequential read' ela= 488 file#=4 block#=1533600 blocks=1 obj#=42089 tim=1341802025
WAIT #1: nam='db file sequential read' ela= 4159 file#=4 block#=1533601 blocks=1 obj#=42089 tim=1341806266
WAIT #1: nam='db file scattered read' ela= 13817 file#=4 block#=1533602 blocks=7 obj#=42089 tim=1341820190
FETCH #1:c=15625,e=220015,p=149,cr=146,cu=0,mis=0,r=1,dep=0,og=1,tim=1341820319
WAIT #1: nam='SQL*Net message from client' ela= 604 driver id=1413697536 #bytes=1 p3=0 obj#=42089 tim=1341821085
FETCH #1:c=0,e=2,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,tim=1341821164
WAIT #1: nam='SQL*Net message to client' ela= 3 driver id=1413697536 #bytes=1 p3=0 obj#=42089 tim=1341821215
WAIT #1: nam='SQL*Net message from client' ela= 507 driver id=1413697536 #bytes=1 p3=0 obj#=42089 tim=1341821764
STAT #1 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT AGGREGATE (cr=146 pr=149 pw=0 time=220017 us)'
STAT #1 id=2 cnt=10000 pid=1 pos=1 obj=42089 op='TABLE ACCESS FULL T1 (cr=146 pr=149 pw=0 time=201385 us)'
```

As you can see, Oracle dropped from reading 7 or 8 blocks at a time (the reason is explained in this thread) to reading 1 block at a time. I have not yet run the two files through TKPROF, but I would guess that it is faster to read 8 blocks at a time than it is to read 8 blocks, one block at a time.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 11, 2008 3:00 PM in response to: Charles Hooper

> As you can see, Oracle dropped from reading 7 or 8
> blocks at a time (the reason is explained in this
> thread) to reading 1 block at a time. I have not yet
> run the two files through TKPROF, but I would guess
> that it is faster to read 8 blocks at a time than it
> is to read 8 blocks, one block at a time.

Charles,

>As you can see, Oracle dropped from reading 7 or 8 blocks at a time (the reason is explained in this thread)
>to reading 1 block at a time. I have not yet run the two files through TKPROF, but I would guess that it is faster
>to read 8 blocks at a time than it is to read 8 blocks, one block at a time

I think, that's what Mr.Burlson mean by stating

> What I found on a database just this week, is that
> ditching the 10.2 MBRC=0 (automatic MBRC tuning) and
> using manual optimization, my client saw a 22%
> throughput improvement.

Coincidently i run similar test like yours in my test system a day back. But i got a different number
in MBRC other than 1. Honestly, i don't think any point in posting any proof result any more.

Regards,
sp009

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 3:01 PM in response to: sp009

xxx xxxxxxxx wrote:
What I found on a database just this week, is that ditching the 10.2 MBRC=0 (automatic MBRC tuning) and using manual optimization, m
a 22%
throughput improvement.

sp009 wrote:
Not my call, but i would like you to have a look at
https://metalink.oracle.com/metalink/plsql/f?p=200:27:11900370213987146647::::p27_id,p27_show_header,p27_show_help:71475.993,1,1

Charles Hooper wrote:
As you can see, Oracle dropped from reading 7 or 8 blocks at a time (the reason is explained in this thread) to reading 1 block at a

There is a bug on this: bug 5768025
Setting DB_FILE_MULTIBLOCK_READ_COUNT=0 incorrectly results in DB_FILE_MULTIBLOCK_READ_COUNT=1 and does not enable self-tuning MBRC.

Workaround: do not set DB_FILE_MULTIBLOCK_READ_COUNT as an init.ora parameter

--
Regards,

Greg Rahn
http://structureddata.org

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 11, 2008 3:14 PM in response to: sp009

Reply

sp009

>
> Not my call, but i would like you to have a look at
> https://metalink.oracle.com/metalink/plsql/f?p=200:27:
> 11900370213987146647::::p27_id,p27_show_header,p27_show
> _help:714075.993,1,1

Any particular reason why you think that that page is worth reading ?
The thing that surprised me most was that the initial posting said that a range of 3.324 seconds to 5.357 seconds was indicative of "almost no effect on performance".

Switching db_file_mulitblock_read_count to zero in 10.2 wll make the optimizer use a value of 1, it doesn't enable the automatic selection. I mentioned this a few weeks ago on the following thread:
http://forums.oracle.com/forums/message.jspa?messageID=2499205#2499205

The first reply (Edward Maynard) is misleading - if the system statistics are set then (apart from edge cases which are discussed on my blog http://jonathanlewis.wordpress.com/2007/05/20/system-stats-strategy/) the optimizer uses the MBRC value for costing purposes, and the run-time engine still uses the db_file_multiblock_read_count for execution purposes.

The second reply (Oracle, Helmut Pfau) is misleading - it applies to 8i, and to later versions if system statistics have not been collected.

The third reply (from the OP) shows that he has noted that setting the db_file_multiblock_read_count to zero is equivalent to setting it to 1.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 11, 2008 6:41 PM in response to: Greg Rahn

Reply

> > > **4) I wonder whether Oracle uses multiple block sizes in those**
> > > apps for any reason other than transportable tablespace.
> > >
> > > I believe that our internal applications use an 8k block.
> >
> > Any chance of getting that verified?
>
> I just confirmed with someone who works frequently
> with those systems and they are not aware of any use
> of any other block size than 8k.
>

Thanks for that, Greg.

So we now have a data point in the discussion about multiple block sizes, as well as non-default block sizes, for various real-world loads. That data point does not make use of test data, simulations, harnesses, or any other hypothetical methods.

We know that Oracle is a 50,000+ employee, $10b+ company who run Oracle E-Business, Peoplesoft and Siebel applications and related warehouses and BI. That sounds like a real-world load to me.

The only meaningful performance tuning criteria is user satisfaction. Everything else (including traces and statspack) is simply metrics to measure, support or argue against that criteria. We can also surmise that the CEO, CFO and Presidents of a company like Oracle will not tolerate performance issues in retrieving their data for BI purposes.

We also know that the Oracle internal applications use the default block size, and we/they are not aware of any non-default or multiple block size settings, in their real world systems. They have not changed that for performance purposes.

This does not imply that such a change will never be useful. Nor does it imply that such a change will never yield benefits. (Sorry for the double-negatives.)

It does imply that such a change is likely far down the list of possible changes to review in real-world loads as represented by Oracle's own internal systems.

(Which leads me to wonder why there have been over 250 sometimes very passionate replies around this topic. Is this yet another 'making a mountain out of molehill' story? <g>)

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 11, 2008 6:58 PM in response to: Steve Karam

Reply

One of the issues with RAC, unfortunately, is that while Oracle calls a 2 node cluster a special case we all know that a large percentage of cluster builders ignore that advice and build them anyway. More nodes provides many values among which is the ability to use services to partition different workloads.

That said there are many ways to reduce the number of rows stored in a block. Changing the block size to 2K is only one of them. The claim that somehow reducing interconnect traffic supports making 2K blocks is like claiming that infections support the use of penicillin. Penicillin is one possible solution to an infection: Not the only solution. Similarly all RAC clusters will not benefit from a 2K block any more than you can kill a gram negative cell with penicillin.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 11, 2008 7:14 PM in response to: Jonathan Lewis

Reply

Same experience here. Just signed a contract today with a client that starts with a decision to solve the problem with a redesign.

Part of that redesign will likely include adding a TimesTen infrastructure.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 11, 2008 7:22 PM in response to: chris_c

Reply

My favorite paragraph from the metalink note is:

"Rebuilding such indexes can actually be detrimental to overall performance for a number of reasons. Firstly, it requires a significant amount of resources and can conflict with the general running of the database. But perhaps more importantly, it can actually be self-defeating in what rebuilds are supposed to achieve. That's because after an index rebuild, the index is more tightly packed with less overall free space (else why rebuild).
This means however that index splits are more likely to now occur which directly impacts performance due to the additional I/O and CPU this entails. And after the block split, we now have two blocks each with 50% free space. After a period of time, the index potentially has "issues" due to insufficient used space and the vicious rebuild cycle continues. The better course of action is to do nothing and let the index evolve to it's natural "equilibrium"."

This same advice was given at OpenWorld last year by Richard Foote as part of the Unconference. And has been given by Richard and other Oakies at a number of Oracle conferences I have attended.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 11, 2008 7:28 PM in response to: Hans Forbrich
Reply

I too have been told that all internal Oracle systems use an 8K block.

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 11, 2008 7:37 PM in response to: sp009
Reply

> Coincidently i run similar test like yours in my test
> system a day back. But i got a different number
> in MBRC other than 1. Honestly, i don't think any
> point in posting any proof result any more.

But here is the critical issue when it comes to proof -- Charles shared a script and methodology for determining the behaviour of Oracle when you modify a particular parameter. Maybe that behaviour does change with release, but now people have something to take to their own system to determine the effect for themnselves.

To me that is the whole essence of providing scripts and proofs -- that they allow everyone to run their own tests on their own systems and intelligently interpret the results. That is infinitely more worthwhile to the Oracle community than generalities.

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 11, 2008 7:50 PM in response to: David Aldridge
Reply

Your point was poignantly made for me with an example I created for teaching my students at the university.
Here's the statement:

```
SELECT DISTINCT srvr_id
FROM servers
WHERE srvr_id NOT IN (
  SELECT srvr_id
  FROM servers
  MINUS
  SELECT srvr_id
  FROM serv_inst);
```

Anyone wishing to create a "general rule" about this query had best run an Explain Plan in 8.1.7.4, 9.2.0.4, and 10.2.0.1 before they do so.

The query setup and test queries for the class are here:
http://www.psoug.org/reference/explain_plan.html

---

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:14 PM in response to: damorgan
Reply

Mr. Damorgan,

I would like to have a look at page#3 by JL

>>Neither database took any time to run the query - what you're looking
>>at is execution plan which is the predicted cost and time to run.

No more Comments...

sp009

David Aldridge

Posts: 1,022
From: XM Satellite Radio, Washington DC
Registered: 10/5/98

Re: Larger vs. Small data block
Posted: Jun 11, 2008 9:50 PM in response to: sp009
Reply

> Mr. Damorgan,
>
> I would like to have a look at page#3 by JL
>
> >>Neither database took any time to run the query -
> what you're looking
> >>at is execution plan which is the predicted cost
> and time to run.
>
> No more Comments...
>
> sp009

JL's point was that cost reductions do not demonstrate performance improvements. If they did then the optimizer's choice of the lowest cost execution plan would be perfect, and we know that's not the case.

DAM's point was that with changes in Oracle version the same query will be optimized to different execution plans.

The former does not invalidate the latter, if that was your point.

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 12, 2008 5:56 AM in response to: Jonathan Lewis
Reply

> Woody,
>
> > The initial email sent with the URL to login to
> the
> > StatspackAnalyzer.com tool has included a
> graphical
> > tracking bit. This bit tells us one thing... that
> > the email has been opened. It is our best way to
> > verify that there is not something wrong with our
> > email system and also to do a rough check to see
> if
> > people are actually opening the emails we send out
> > with the login.

>
> Thank you for this posting. Apart from re-assuring
> your potential users, it's also captured the theme of
> thread in a microcosm.
>
> a) Faust was correct in his observation that the
> email carried a trojan - but his degree of
> information (or interest) did not extend far enough
> to discover that the trojan was a harmless graphical
> tracking bit.
>
> b) Steve Karam was correct in his observation that
> when he did his testing there were no trojans,
> because he didn't see a trojan. However, he may have
> failed to detect the "trojan" because he saw it, knew
> what it really was, and discounted it; or he may
> simply not have noticed.
>
> c) Both of them were wrong, and careful testing would
> have shown this. Both could have claimed (and did)
> that their observations were valid because they were
> based on "empirical observations" of a "real-world
> system".

Definitely agree with you Jonathan.

Thanks to all who posted material usefull for clarifying this!

Regards,
Faust

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 12, 2008 7:34 AM in response to: Charles Hooper

Reply

> (Snip)
> > BTW, I agree with Greg that MBRC is also a factor,
> > but for surprizing reasons.
> >
> > What I found on a database just this week, is that
> > ditching the 10.2 MBRC=0 (automatic MBRC tuning)
> and
> > using manual optimization, my client saw a 22%
> > throughput improvement.
> >
> > But even stranger, this is a well-indexed OLTP app
> > that does not do many scattered reads!
> >
> > The conventional wisdon suggests the multi-block
> read
> > size is only for full-scan operations, but I found
> > that optimizing MBRC is also important for
> optimizing
> > inserts on reverse key indexes, and possible index
> > range scans . . .
> (Snip)
>
>
> You stated:
> *"What I found on a database just this week, is*
> *that ditching the 10.2 MBRC=0 (automatic MBRC tuning)*
> *and using manual optimization, my client saw a 22%*
> *throughput improvement."*

I see that Mr. xxxxxxxx has removed the above comment from his post to which I had replied.

Just to make certain that my tests of MBRC using artificial data were not invalid, I performed a test on a production database server against live data. The server is running Oracle 10.2.0.2 on Windows 2003 x64 using RAID 10 with the read cache disabled, and the database has an 8KB block size. Since this is the same Oracle version as my test case, I did not trying to set MBRC=0 to force automatic MBRC tuning, as it was found that this caused single block reads, rather than multi-block reads.

The first test sets DB_FILE_MULTIBLOCK_READ_COUNT to 32, which yields a maximum of a 256KB multi-block scattered read. I had previously seen articles recommending 64KB or 128KB as the maximum size of multi-block scattered reads on Windows (DB_FILE_MULTIBLOCK_READ_COUNT of 8 or 16 with 8KB block size), including the following older documents:
http://www.pafumi.net/Oracle_on_NT.htm
http://download-east.oracle.com/docs/html/A76956_01/create.htm

The second test removed DB_FILE_MULTIBLOCK_READ_COUNT from the spfile, thus allowing Oracle to automatically set DB_FILE_MULTIBLOCK_READ_COUNT to 128 (1MB multi-block read size). The results were a bit surprising - a single 1MB scattered read required less time to complete than a single 256KB scattered read on exactly the same data (index fast full scan was used, so it was actually scanning the index and not the table).

From the 10046 trace, with DB_FILE_MULTIBLOCK_READ_COUNT set to 32 (note: buffer cache flushed before execution):

```
PARSE #3:c=0,e=15392,p=1,cr=20,cu=0,mis=1,r=0,dep=0,og=1,tim=2385093630
EXEC #3:c=0,e=49,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=2385093737
WAIT #3: nam='SQL*Net message to client' ela= 5 driver id=1413697536 #bytes=1 p3=0 obj#=43161 tim=2385093779
WAIT #3: nam='db file sequential read' ela= 10394 file#=4 block#=979859 blocks=1 obj#=43151 tim=2385104277
WAIT #3: nam='db file scattered read' ela= 14291 file#=4 block#=979860 blocks=5 obj#=43151 tim=2385118750
WAIT #3: nam='db file scattered read' ela= 13416 file#=4 block#=979865 blocks=8 obj#=43151 tim=2385132451
WAIT #3: nam='db file scattered read' ela= 5457 file#=4 block#=979874 blocks=7 obj#=43151 tim=2385138359
WAIT #3: nam='db file scattered read' ela= 14608 file#=4 block#=979881 blocks=8 obj#=43151 tim=2385153372
WAIT #3: nam='db file scattered read' ela= 516 file#=4 block#=979890 blocks=7 obj#=43151 tim=2385154326
WAIT #3: nam='db file scattered read' ela= 545 file#=4 block#=979897 blocks=8 obj#=43151 tim=2385155264
WAIT #3: nam='db file scattered read' ela= 512 file#=4 block#=979906 blocks=7 obj#=43151 tim=2385156220
WAIT #3: nam='db file scattered read' ela= 551 file#=4 block#=979913 blocks=8 obj#=43151 tim=2385157169
WAIT #3: nam='db file scattered read' ela= 516 file#=4 block#=979922 blocks=7 obj#=43151 tim=2385158129
WAIT #3: nam='db file scattered read' ela= 554 file#=4 block#=979929 blocks=8 obj#=43151 tim=2385159079
WAIT #3: nam='db file scattered read' ela= 514 file#=4 block#=979938 blocks=7 obj#=43151 tim=2385160052
WAIT #3: nam='db file scattered read' ela= 558 file#=4 block#=979945 blocks=8 obj#=43151 tim=2385161008
WAIT #3: nam='db file scattered read' ela= 516 file#=4 block#=979954 blocks=7 obj#=43151 tim=2385161962
WAIT #3: nam='db file scattered read' ela= 548 file#=4 block#=979961 blocks=8 obj#=43151 tim=2385162910
WAIT #3: nam='db file scattered read' ela= 508 file#=4 block#=979970 blocks=7 obj#=43151 tim=2385163850
WAIT #3: nam='db file scattered read' ela= 4359 file#=4 block#=980617 blocks=8 obj#=43151 tim=2385168604
WAIT #3: nam='db file scattered read' ela= 5218 file#=4 block#=980747 blocks=32 obj#=43151 tim=2385174327
WAIT #3: nam='db file scattered read' ela= 2076 file#=4 block#=980779 blocks=32 obj#=43151 tim=2385178342
WAIT #3: nam='db file scattered read' ela= 2037 file#=4 block#=980811 blocks=32 obj#=43151 tim=2385182061
WAIT #3: nam='db file scattered read' ela= 1949 file#=4 block#=980843 blocks=30 obj#=43151 tim=2385185699
WAIT #3: nam='db file scattered read' ela= 2023 file#=4 block#=980875 blocks=32 obj#=43151 tim=2385189314
WAIT #3: nam='db file scattered read' ela= 2046 file#=4 block#=980907 blocks=32 obj#=43151 tim=2385193194
WAIT #3: nam='db file scattered read' ela= 2023 file#=4 block#=980939 blocks=32 obj#=43151 tim=2385196921
WAIT #3: nam='db file scattered read' ela= 1945 file#=4 block#=980971 blocks=30 obj#=43151 tim=2385200546
WAIT #3: nam='db file scattered read' ela= 2037 file#=4 block#=981003 blocks=32 obj#=43151 tim=2385204171
WAIT #3: nam='db file scattered read' ela= 2599 file#=4 block#=981035 blocks=32 obj#=43151 tim=2385208540
WAIT #3: nam='db file scattered read' ela= 4730 file#=4 block#=981067 blocks=32 obj#=43151 tim=2385214966
WAIT #3: nam='db file scattered read' ela= 1920 file#=4 block#=981099 blocks=30 obj#=43151 tim=2385218565
WAIT #3: nam='db file scattered read' ela= 2039 file#=4 block#=981131 blocks=32 obj#=43151 tim=2385222184
WAIT #3: nam='db file scattered read' ela= 2039 file#=4 block#=981163 blocks=32 obj#=43151 tim=2385225912
WAIT #3: nam='db file scattered read' ela= 6530 file#=4 block#=981195 blocks=32 obj#=43151 tim=2385234108
WAIT #3: nam='db file scattered read' ela= 1941 file#=4 block#=981227 blocks=30 obj#=43151 tim=2385237734
WAIT #3: nam='db file scattered read' ela= 2026 file#=4 block#=981259 blocks=32 obj#=43151 tim=2385241340
WAIT #3: nam='db file scattered read' ela= 2024 file#=4 block#=981291 blocks=32 obj#=43151 tim=2385245040
WAIT #3: nam='db file scattered read' ela= 2029 file#=4 block#=981323 blocks=32 obj#=43151 tim=2385248754
WAIT #3: nam='db file scattered read' ela= 1950 file#=4 block#=981355 blocks=30 obj#=43151 tim=2385252400
WAIT #3: nam='db file scattered read' ela= 2030 file#=4 block#=981387 blocks=32 obj#=43151 tim=2385256024
...
WAIT #3: nam='db file scattered read' ela= 2032 file#=4 block#=985227 blocks=32 obj#=43151 tim=2385788598
WAIT #3: nam='db file scattered read' ela= 2011 file#=4 block#=985259 blocks=32 obj#=43151 tim=2385792264
WAIT #3: nam='db file scattered read' ela= 2019 file#=4 block#=985291 blocks=32 obj#=43151 tim=2385795955
WAIT #3: nam='db file scattered read' ela= 2803 file#=4 block#=985323 blocks=30 obj#=43151 tim=2385800415
FETCH #3:c=312500,e=707606,p=4655,cr=4668,cu=0,mis=0,r=1,dep=0,og=1,tim=2385801426
WAIT #3: nam='SQL*Net message from client' ela= 329 driver id=1413697536 #bytes=1 p3=0 obj#=43151 tim=2385801887
FETCH #3:c=0,e=2,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,tim=2385801946
WAIT #3: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=43151 tim=2385801978
WAIT #3: nam='SQL*Net message from client' ela= 588 driver id=1413697536 #bytes=1 p3=0 obj#=43151 tim=2385802600
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT AGGREGATE (cr=4668 pr=4655 pw=0 time=707601 us)'
STAT #3 id=2 cnt=2557544 pid=1 pos=1 obj=43151 op='INDEX FAST FULL SCAN X_INV_7 (cr=4668 pr=4655 pw=0 time=5140073 us)'
```

From the 10046 trace, with DB_FILE_MULTIBLOCK_READ_COUNT not manually specified (note: buffer cache flushed before execution):

```
PARSE #3:c=31250,e=62958,p=8,cr=273,cu=0,mis=1,r=0,dep=0,og=1,tim=2580626091
EXEC #3:c=0,e=51,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=2580626198
WAIT #3: nam='SQL*Net message to client' ela= 4 driver id=1413697536 #bytes=1 p3=0 obj#=43161 tim=2580626241
WAIT #3: nam='db file sequential read' ela= 144 file#=4 block#=979859 blocks=1 obj#=43151 tim=2580626488
WAIT #3: nam='db file scattered read' ela= 133 file#=4 block#=979860 blocks=5 obj#=43151 tim=2580626768
WAIT #3: nam='db file scattered read' ela= 158 file#=4 block#=979865 blocks=8 obj#=43151 tim=2580627252
WAIT #3: nam='db file scattered read' ela= 149 file#=4 block#=979874 blocks=7 obj#=43151 tim=2580627911
WAIT #3: nam='db file scattered read' ela= 149 file#=4 block#=979881 blocks=8 obj#=43151 tim=2580628519
WAIT #3: nam='db file scattered read' ela= 143 file#=4 block#=979890 blocks=7 obj#=43151 tim=2580629156
WAIT #3: nam='db file scattered read' ela= 156 file#=4 block#=979897 blocks=8 obj#=43151 tim=2580629761
WAIT #3: nam='db file scattered read' ela= 146 file#=4 block#=979906 blocks=7 obj#=43151 tim=2580630409
WAIT #3: nam='db file scattered read' ela= 150 file#=4 block#=979913 blocks=8 obj#=43151 tim=2580631008
WAIT #3: nam='db file scattered read' ela= 146 file#=4 block#=979922 blocks=7 obj#=43151 tim=2580631659
WAIT #3: nam='db file scattered read' ela= 164 file#=4 block#=979929 blocks=8 obj#=43151 tim=2580632273
WAIT #3: nam='db file scattered read' ela= 147 file#=4 block#=979938 blocks=7 obj#=43151 tim=2580632937
WAIT #3: nam='db file scattered read' ela= 146 file#=4 block#=979945 blocks=8 obj#=43151 tim=2580633536
WAIT #3: nam='db file scattered read' ela= 143 file#=4 block#=979954 blocks=7 obj#=43151 tim=2580634180
WAIT #3: nam='db file scattered read' ela= 155 file#=4 block#=979961 blocks=8 obj#=43151 tim=2580634795
WAIT #3: nam='db file scattered read' ela= 143 file#=4 block#=979970 blocks=7 obj#=43151 tim=2580635434
WAIT #3: nam='db file scattered read' ela= 149 file#=4 block#=980617 blocks=8 obj#=43151 tim=2580636036
WAIT #3: nam='db file scattered read' ela= 1686 file#=4 block#=980747 blocks=126 obj#=43151 tim=2580638453
WAIT #3: nam='db file scattered read' ela= 1680 file#=4 block#=980875 blocks=126 obj#=43151 tim=2580648483
WAIT #3: nam='db file scattered read' ela= 1676 file#=4 block#=981003 blocks=126 obj#=43151 tim=2580658531
WAIT #3: nam='db file scattered read' ela= 1717 file#=4 block#=981131 blocks=126 obj#=43151 tim=2580668593
WAIT #3: nam='db file scattered read' ela= 1713 file#=4 block#=981259 blocks=126 obj#=43151 tim=2580678713
WAIT #3: nam='db file scattered read' ela= 1679 file#=4 block#=981387 blocks=126 obj#=43151 tim=2580688829
WAIT #3: nam='db file scattered read' ela= 1685 file#=4 block#=981515 blocks=126 obj#=43151 tim=2580698915
WAIT #3: nam='db file scattered read' ela= 1691 file#=4 block#=981643 blocks=126 obj#=43151 tim=2580709020
WAIT #3: nam='db file scattered read' ela= 1784 file#=4 block#=981771 blocks=126 obj#=43151 tim=2580719176
WAIT #3: nam='db file scattered read' ela= 1701 file#=4 block#=981899 blocks=126 obj#=43151 tim=2580729291
WAIT #3: nam='db file scattered read' ela= 1679 file#=4 block#=982027 blocks=126 obj#=43151 tim=2580739559
WAIT #3: nam='db file scattered read' ela= 1674 file#=4 block#=982155 blocks=126 obj#=43151 tim=2580749868
WAIT #3: nam='db file scattered read' ela= 1680 file#=4 block#=982283 blocks=126 obj#=43151 tim=2580759802
WAIT #3: nam='db file scattered read' ela= 1688 file#=4 block#=982411 blocks=126 obj#=43151 tim=2580769935
WAIT #3: nam='db file scattered read' ela= 1676 file#=4 block#=982539 blocks=126 obj#=43151 tim=2580780224
WAIT #3: nam='db file scattered read' ela= 1675 file#=4 block#=982667 blocks=126 obj#=43151 tim=2580790321
WAIT #3: nam='db file scattered read' ela= 1690 file#=4 block#=982795 blocks=126 obj#=43151 tim=2580800425
WAIT #3: nam='db file scattered read' ela= 1684 file#=4 block#=982923 blocks=126 obj#=43151 tim=2580810470
WAIT #3: nam='db file scattered read' ela= 1676 file#=4 block#=983051 blocks=126 obj#=43151 tim=2580820535
WAIT #3: nam='db file scattered read' ela= 1717 file#=4 block#=983179 blocks=126 obj#=43151 tim=2580830798
WAIT #3: nam='db file scattered read' ela= 1702 file#=4 block#=983307 blocks=126 obj#=43151 tim=2580840861
WAIT #3: nam='db file scattered read' ela= 1690 file#=4 block#=983435 blocks=126 obj#=43151 tim=2580850868
WAIT #3: nam='db file scattered read' ela= 1684 file#=4 block#=983563 blocks=126 obj#=43151 tim=2580860938
WAIT #3: nam='db file scattered read' ela= 1683 file#=4 block#=983691 blocks=126 obj#=43151 tim=2580870983
WAIT #3: nam='db file scattered read' ela= 1724 file#=4 block#=983819 blocks=126 obj#=43151 tim=2580881069
WAIT #3: nam='db file scattered read' ela= 1712 file#=4 block#=983947 blocks=126 obj#=43151 tim=2580891165
WAIT #3: nam='db file scattered read' ela= 1690 file#=4 block#=984075 blocks=126 obj#=43151 tim=2580901191
WAIT #3: nam='db file scattered read' ela= 1675 file#=4 block#=984203 blocks=126 obj#=43151 tim=2580911220
WAIT #3: nam='db file scattered read' ela= 1678 file#=4 block#=984331 blocks=126 obj#=43151 tim=2580921506
WAIT #3: nam='db file scattered read' ela= 1681 file#=4 block#=984459 blocks=126 obj#=43151 tim=2580931480
WAIT #3: nam='db file scattered read' ela= 1800 file#=4 block#=984587 blocks=126 obj#=43151 tim=2580942451
WAIT #3: nam='db file scattered read' ela= 1686 file#=4 block#=984715 blocks=126 obj#=43151 tim=2580953411
WAIT #3: nam='db file scattered read' ela= 1681 file#=4 block#=984843 blocks=126 obj#=43151 tim=2580963403
WAIT #3: nam='db file scattered read' ela= 1675 file#=4 block#=984971 blocks=126 obj#=43151 tim=2580973475
WAIT #3: nam='db file scattered read' ela= 1679 file#=4 block#=985099 blocks=126 obj#=43151 tim=2580983501
WAIT #3: nam='db file scattered read' ela= 1680 file#=4 block#=985227 blocks=126 obj#=43151 tim=2580993541
FETCH #3:c=328125,e=374950,p=4655,cr=4668,cu=0,mis=0,r=1,dep=0,og=1,tim=2581001232
WAIT #3: nam='SQL*Net message from client' ela= 321 driver id=1413697536 #bytes=1 p3=0 obj#=43151 tim=2581001710
FETCH #3:c=0,e=2,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,tim=2581001768
WAIT #3: nam='SQL*Net message to client' ela= 2 driver id=1413697536 #bytes=1 p3=0 obj#=43151 tim=2581001798
WAIT #3: nam='SQL*Net message from client' ela= 604 driver id=1413697536 #bytes=1 p3=0 obj#=43151 tim=2581002423
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT AGGREGATE (cr=4668 pr=4655 pw=0 time=374945 us)'
STAT #3 id=2 cnt=2557544 pid=1 pos=1 obj=43151 op='INDEX FAST FULL SCAN X_INV_7 (cr=4668 pr=4655 pw=0 time=2558111 us)'
```

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

SeanMacGC

Posts: 7
Registered: 10/30/06

Re: Larger vs. Small data block
Posted: Jun 12, 2008 8:00 AM in response to: Charles Hooper

Reply

I see that Mr. xxxxxxxx has removed the above comment from his post to which I had replied.

Indeed Charles, as his rather conspicuous absolute silence on this thread since you made the original comment bears testimony to too. All very confidence inducing.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 12, 2008 10:57 AM in response to: Charles Hooper

Reply

Again excellent work Charles. Now if we could just get you out here to the coast.

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 12, 2008 11:14 AM in response to: damorgan

Reply

> Again excellent work Charles. Now if we could just
> get you out here to the coast.

Unfortunately, the trace file that I posted makes it hard to see the performance difference that I was trying to highlight. By increasing the multi-block read by a factor of 4, the disk read performance improved by a factor of 7 to 9. I suspect that the difference between reading in one read call 1 block compared to reading 128 blocks, there would be an even greater difference.

The TKPFOF output for three select statements - the first matches the trace files that I posted:

DB_FILE_MULTIBLOCK_READ_COUNT=32
```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.31       0.70       4655       4668          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.31       0.71       4655       4668          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 39

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=4668 pr=4655 pw=0 time=707601 us)
2557544   INDEX FAST FULL SCAN X_INV_7 (cr=4668 pr=4655 pw=0 time=5140073 us)(object id 43151)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         1        0.01          0.01
  db file scattered read                        160        0.01          0.45
  SQL*Net message from client                     2        0.00          0.00

******
```
DB_FILE_MULTIBLOCK_READ_COUNT=Unset
```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.03       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.32       0.37       4655       4668          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.35       0.38       4655       4668          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 39

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=4668 pr=4655 pw=0 time=374945 us)
2557544   INDEX FAST FULL SCAN X_INV_7 (cr=4668 pr=4655 pw=0 time=2558111 us)(object id 43151)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         1        0.00          0.00
  db file scattered read                         52        0.00          0.06
  SQL*Net message from client                     2        0.00          0.00

****************************************************************************************************
```
DB_FILE_MULTIBLOCK_READ_COUNT=32
```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.14       0.37       2507       2516          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.14       0.39       2507       2516          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 39

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=2516 pr=2507 pw=0 time=379713 us)
1379582   INDEX FAST FULL SCAN X_LT_6 (cr=2516 pr=2507 pw=0 time=10877 us)(object id 43161)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         1        0.00          0.00
  db file scattered read                         92        0.00          0.24
  SQL*Net message from client                     2        0.00          0.00

******
```

DB_FILE_MULTIBLOCK_READ_COUNT=Unset

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.03       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.18       0.19       2507       2516          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.21       0.21       2507       2516          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 39

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=2516 pr=2507 pw=0 time=199744 us)
1379582   INDEX FAST FULL SCAN X_LT_6 (cr=2516 pr=2507 pw=0 time=528 us)(object id 43161)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         1        0.00          0.00
  db file scattered read                         35        0.00          0.03
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************
```

DB_FILE_MULTIBLOCK_READ_COUNT=32

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.02          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.15       0.31       2011       2020          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.17       0.33       2011       2020          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 39

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=2020 pr=2011 pw=0 time=313081 us)
1106750   INDEX FAST FULL SCAN X_R_2 (cr=2020 pr=2011 pw=0 time=8673 us)(object id 43287)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         1        0.00          0.00
  db file scattered read                         77        0.00          0.18
  SQL*Net message from client                     2        0.00          0.00
```

DB_FILE_MULTIBLOCK_READ_COUNT=Unset

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.02          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.12       0.16       2011       2020          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.14       0.18       2011       2020          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 39

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=2020 pr=2011 pw=0 time=163378 us)
1106750   INDEX FAST FULL SCAN X_R_2 (cr=2020 pr=2011 pw=0 time=760 us)(object id 43287)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file sequential read                         1        0.00          0.00
  db file scattered read                         32        0.00          0.02
  SQL*Net message from client                     2        0.00          0.00
```

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 12, 2008 1:48 PM in response to: Greg Rahn

Reply

> **xxx xxxxxxxx wrote:**
> *What I found on a database just this week, is that*
> *ditching the 10.2 MBRC=0 (automatic MBRC tuning) and*
> *using manual optimization, my client saw a 22%*
> *throughput improvement.*
>
> **sp009 wrote:**
> *Not my call, but i would like you to have a look*
> *at*
> *https://metalink.oracle.com/metalink/plsql/f?p=200:27:*
> *1190037021398714647::::p27_id,p27_show_header,p27_show*
> *_help:71475.993,1,1*
>
> **Charles Hooper wrote:**
> *As you can see, Oracle dropped from reading 7 or 8*
> *blocks at a time (the reason is explained in this*
> *thread) to reading 1 block at a time.*

```
>
> There is a bug on this: bug 5768025
> Setting DB_FILE_MULTIBLOCK_READ_COUNT=0 incorrectly
> results in DB_FILE_MULTIBLOCK_READ_COUNT=1 and does
> not enable self-tuning MBRC.
>
> Workaround: do not set DB_FILE_MULTIBLOCK_READ_COUNT
> as an init.ora parameter
>
> --
> Regards,
>
> Greg Rahn
> http://structureddata.org
```

Greg,

I see that you updated the meta link thread. Would you mind updating the same stating that "Its been taken care in 10.2.0.4 without any workaround"

Thanks

sp009
Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 12, 2008 2:36 PM in response to: Charles Hooper
Reply

Charles,

In 10.2.0.4, If you set db_file_multiblock_read_count=0 (Dynamic MBRC),
then Oracle will tend to set maximum value based on OS limit.

If i set db_file_multiblock_read_count as 1 manually , i can see high sequential
read in the tkprof. Also didn't see much performance difference between
db_file_multiblock_read_count=0 and db_file_multiblock_read_count=128.
In fact db_file_multiblock_read_count=128 setting actually reduced the
number of scattered read and the cost nearly same.

Regards,

damorgan
Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 12, 2008 5:39 PM in response to: sp009
Reply

If this parameter is not set explicitly (or is set is 0), the optimizer will use a default value of 8 when costing full table scans and index fast full scans.

Source:
http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/optimops.htm#BABDECGJ

David_Aldridge
Posts: 97
Registered: 4/22/08

Re: Larger vs. Small data block
Posted: Jun 12, 2008 6:11 PM in response to: damorgan
Reply

... except when it uses mbrc from system statistics though, from what JL was saying.

Niall Litchfield
Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 12, 2008 6:50 PM in response to: David_Aldridge
Reply

> ... except when it uses mbrc from system statistics
> though, from what JL was saying.

exactly correct. Obviously I could illustrate that with scripts, but then the cbo might magically know that there's only one real person on the system and behave differently. So in the spirit of xxx, I'm just going to claim it with no evidence.

Niall

damorgan
Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 12, 2008 7:51 PM in response to: David_Aldridge
Reply

Can you get the docs correction to Francisco?

David_Aldridge
Posts: 97
Registered: 4/22/08

Re: Larger vs. Small data block
Posted: Jun 12, 2008 8:17 PM in response to: damorgan
Reply

Who is this Francisco of whom you speak? Pretend I've not been paying attention ...

damorgan
Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 13, 2008 1:40 AM in response to: David_Aldridge
Reply

Francisco Abedrabbo. I am assuming you are inside Oracle.

If not let me know and I will send it to him. Thanks.

Niall Litchfield
Posts: 301
From: Hampshire UK
Registered: 7/4/99

Re: Larger vs. Small data block
Posted: Jun 13, 2008 4:20 AM in response to: damorgan
Reply

Actually I'm not entirely convinced by the wording of the relevant bit of the docs at all. If you look at the section on workload stats (when mbrc may be gathered by Oracle) at http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/stats.htm#CIHIEIIA then you find this bit of prose on multi block read count. .

"In release 10.2, the optimizer uses the value of mbrc when performing full table scans (FTS). The value of db_file_multiblock_read_count is set to the maximum allowed by the operating system by default. However, the optimizer uses mbrc=8 for costing. The "real" mbrc is actually somewhere in between since serial multiblock read requests are processed by the buffer cache and split in two or more requests if some blocks are already pinned in the buffer cache, or when the segment size is smaller than the read size. The mbrc value gathered as part of workload statistics is thus useful for FTS estimation.

During the gathering process of workload statistics, it is possible that mbrc and mreadtim will not be gathered if no table scans are performed during serial workloads, as is often the case with OLTP systems. On the other hand, FTS occur frequently on DSS systems but may run parallel and bypass the buffer cache. In such cases, sreadtim will still be gathered since index lookup are performed using the buffer cache. If Oracle cannot gather or validate gathered mbrc or mreadtim, but has gathered sreadtim and cpuspeed, then only sreadtim and cpuspeed will be used for costing. FTS cost will be computed using analytical algorithm implemented in previous releases. Another alternative to computing mbrc and mreadtim is to force FTS in serial mode to allow the optimizer to gather the data."

I can't help but feel that that could be clearer. :)

Niall

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 13, 2008 7:20 AM in response to: sp009

Reply

> Charles,
>
> In 10.2.0.4, If you set
> db_file_multiblock_read_count=0 (Dynamic MBRC),
> then Oracle will tend to set maximum value based on
> OS limit.
>
> If i set db_file_multiblock_read_count as 1 manually
> , i can see high sequential
> read in the tkprof. Also didn't see much performance
> difference between
> db_file_multiblock_read_count=0 and
> db_file_multiblock_read_count=128.
> In fact db_file_multiblock_read_count=128 setting
> actually reduced the
> number of scattered read and the cost nearly same.
>
> Regards,

I think that I have to take issue with the broad statement that db_file_multiblock_read_count=0 is the equivalent of "Dynamic MBRC". I checked the documentation and did not find a statement indicating that to enable automatic calculation of db_file_multiblock_read_count, db_file_multiblock_read_count should be set to 0 (if that is the definition of "Dynamic MBRC"). There are a couple references in the documentation for Oracle 10.2 that seem to imply a special behavior when that parameter is set to 0.

If you state that in 10.2.0.4, setting db_file_multiblock_read_count=0 results in automatic calculation of db_file_multiblock_read_count, then that is a change from what I demonstrated with 10.2.0.2. I will have to take a look at this, thanks for pointing it out.

The statement in your most recent post (quoted above) seemed to be stating the second of the above cases. Mr. xxxxxxxx's statement seemed to be non-version specific. Considering how short of time 10.2.0.4 has been available compared with 10.2.0.1, 10.2.0.2, or 10.2.0.3, it would seem that a version qualification of the statement, and also a definition of "Dynamic MBRC" would have been helpful.

I agree with the statement that db_file_multiblock_read_count=1 results in single block reads, as were found in the portion of the trace file that I posted. The difference was that I did not set db_file_multiblock_read_count=1, that was a result of setting db_file_multiblock_read_count=0 to test the effects of that value for the parameter (reported as a bug by Greg Rahn, and reported several times (in several web sites) as a potential problem by Jonathan Lewis).

Just as I did not set db_file_multiblock_read_count=1, I did not set db_file_multiblock_read_count=128 - that was automatically set by Oracle on my server when db_file_multiblock_read_count was removed from the spfile and the database was bounced. You may want to examine **why** you did not see much performance difference between db_file_multiblock_read_count=0 and db_file_multiblock_read_count=128, and why (and how much) "db_file_multiblock_read_count=128 setting actually reduced the number of scattered read and the cost nearly same."

Some starting points for observation of the above:
* I believe that you stated that your database is using a 16KB block size, and 16KB * 128 is what value?
* The maximum read size on most platforms is what value, and how does it compare with the above calculation?
* The extent size of the objects (when you were testing) are what size: 32KB, 64KB, 512KB, 10MB, 100MB? Jonathan Lewis explained the significance of the extent size in this thread, Greg Rahn confirmed the significance, and I was able to confirm it through examination of trace files.
* How many of the database blocks were already in the buffer cache when you were testing?
* How many blocks did the raw trace file show that Oracle was reading at one time when db_file_multiblock_read_count was set to 0 and 128?
* Did you look at a 10053 trace file to determine why the **cost** was nearly the same when db_file_multiblock_read_count was set to 0 and 128?
* When db_file_multiblock_read_count was set to 0, what did Oracle automatically set that parameter's value to?

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 13, 2008 9:44 AM in response to: Charles Hooper

Reply

> Some starting points for observation of the above:
> * I believe that you stated that your database is
> using a 16KB block size, and 16KB * 128 is what
> value?
> * The maximum read size on most platforms is what
> value, and how does it compare with the above
> calculation?
> * The extent size of the objects (when you were
> testing) are what size: 32KB, 64KB, 512KB, 10MB,
> 100MB? Jonathan Lewis explained the significance of
> the extent size in this thread, Greg Rahn confirmed
> the significance, and I was able to confirm it
> through examination of trace files.
> * How many of the database blocks were already in the
> buffer cache when you were testing?

> * How many blocks did the raw trace file show that
> Oracle was reading at one time when
> db_file_multiblock_read_count was set to 0 and 128?
> * Did you look at a 10053 trace file to determine why
> the **cost** was nearly the same when
> db_file_multiblock_read_count was set to 0 and 128?
> * When db_file_multiblock_read_count was set to 0,
> what did Oracle automatically set that parameter's
> value to?
>
> Charles Hooper
> IT Manager/Oracle DBA
> K&M Machine-Fabricating, Inc.

Give me some time, i will post the test case soon

David_Aldridge

Posts: 97
Registered: 4/22/08

Re: Larger vs. Small data block
Posted: Jun 13, 2008 10:16 AM in response to: damorgan

Reply

No, I'm not an Oracle person. I'm still waiting for the personal invite from Larry before considering the move ...

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 13, 2008 1:35 PM in response to: Charles Hooper

Reply

> I think that I have to take issue with the broad
> statement that db_file_multiblock_read_count=0 is the
> equivalent of "Dynamic MBRC". I checked the
> documentation and did not find a statement indicating
> that to enable automatic calculation of
> db_file_multiblock_read_count,
> db_file_multiblock_read_count should be set to 0 (if
> that is the definition of "Dynamic MBRC"). There are
> a couple references in the documentation for Oracle
> 10.2 that seem to imply a special behavior when that
> parameter is set to 0.

See
http://download.oracle.com/docs/cd/B19306_01/server.102/b14211/whatsnew.htm#PFGRF000
http://download.oracle.com/docs/cd/B19306_01/server.102/b14211/optimops.htm#BABDECGJ

Documentation regarding Dynamic MBRC is wrong. If any one set "Dynamic MBRC"
prior to 10.2.0.4.0, means bad performance. Mr. xxxxxxxx is correct, assuming
the version was prior to patch#4, when he said:

> "What I found on a database just this week, is
> that ditching the 10.2 MBRC=0 (automatic MBRC tuning)
> and using manual optimization, my client saw a 22%
> throughput improvement."

Below is the output for Dynamic MBRC for my 2 database in 10.2.0.4.0

```
SQL>
SQL> Connect / as Sysdba
Connected.
SQL>
SQL> Select * From v$version
  2  /

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Prod
PL/SQL Release 10.2.0.4.0 - Production
CORE         10.2.0.4.0    Production
TNS for 32-bit Windows: Version 10.2.0.4.0 - Production
NLSRTL Version 10.2.0.4.0 - Production

SQL> Show Parameter db_file_multiblock_read_count

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_file_multiblock_read_count        integer     8
SQL>
SQL> Show Parameter db_block_size

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_block_size                        integer     8192
SQL>
SQL> Alter Session  Set db_file_multiblock_read_count=0
  2  /

Session altered.

SQL> Show Parameter db_file_multiblock_read_count

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_file_multiblock_read_count        integer     128
SQL>
SQL> Disconnect
Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>


*****************************************************************************************

SQL>
SQL> Connect / as Sysdba
Connected.
SQL>
SQL> Select * From v$version
```

```
  2  /

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Prod
PL/SQL Release 10.2.0.4.0 - Production
CORE         10.2.0.4.0   Production
TNS for 32-bit Windows: Version 10.2.0.4.0 - Production
NLSRTL Version 10.2.0.4.0 - Production

SQL> Show Parameter db_file_multiblock_read_count

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_file_multiblock_read_count        integer     8
SQL>
SQL> Show Parameter db_block_size

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_block_size                        integer     16384
SQL>
SQL> Alter Session  Set db_file_multiblock_read_count=0
  2  /

Session altered.

SQL> Show Parameter db_file_multiblock_read_count

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_file_multiblock_read_count        integer     63
SQL>
SQL> Disconnect
Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>
```

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 13, 2008 1:38 PM in response to: sp009 Reply

> Give me some time, i will post the test case soon

sp009,

A test case showing what is happening would be great - if for no reason other than to satisfy curiosity about how things work, and how things may have changed from one version to another.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 13, 2008 2:14 PM in response to: Charles Hooper Reply

> > Give me some time, i will post the test case soon
>
> sp009,
>
> A test case showing what is happening would be great
> - if for no reason other than to satisfy curiosity
> about how things work, and how things may have
> changed from one version to another.
>
> Charles Hooper
> IT Manager/Oracle DBA
> K&M Machine-Fabricating, Inc.

Charles,

I have 8k and 16k block size database. What test case are you looking for?

db_file_multiblock_read_count = 8k (default) against dynamic in both my database?

sp009

Edited for terminology

Probably I shouldn't say Dynamic MBRC but Self-Tuning MBRC

Message was edited by:
sp009

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 13, 2008 2:24 PM in response to: sp009 Reply

[nobr]>
> See
> http://download.oracle.com/docs/cd/B19306_01/server.10
> 2/b14211/whatsnew.htm#PFGRF000

The DB_FILE_MULTIBLOCK_READ_COUNT initialization parameter is now automatically tuned to use a default value when this parameter is ***not set explicitly***

> http://download.oracle.com/docs/cd/B19306_01/server.10
> 2/b14211/optimops.htm#BABDECGJ
>

*If this parameter is not set explicitly (**or is set is 0**), the optimizer will use a default value of 8 when costing full table scans and index fast full scans.*

(My emphasis)

> Documentation regarding Dynamic MBRC is wrong. If any
> one set "Dynamic MBRC" prior to 10.2.0.4.0, means bad
> performance.

The documentation clearly contains a contradiction - which means that anyone reading the manual would want to check what really happens in the two different sets of circumstances. Enabling dynanamic tuning of the multiblock read count does not cause bad performance prior to 10.2.0.4; setting the db_file_multiblock_read_count to zero quite probably does.

> Mr. xxxxxxxx is correct, assuming
> the version was prior to patch#4, when he said:
> > "What I found on a database just this week, is
> > that ditching the 10.2 MBRC=0 (automatic MBRC tuning)
> > and using manual optimization, my client saw a 22%
> > throughput improvement."
>

Mr. xxxxxxxx is demonstrating the difference between what he calls the "empirical DBA" and the "scientific DBA".

It doesn't take much effort or thought from then "scientifit DBA" to notice that when you set the db_file_multiblock_read_count to zero in earlier versions Oracle it magically sets itself to 1. (Here's a note I wrote in May 2007 which happens to pick up the related details)

On the other hand, the "empirical DBA" would be more inclined to hack in a couple of different manual settings, see a couple of queries do faster tablescans, and say: "automatic tuning of the db_file_multiblock_read_count doesn't work".

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk[/nobr]

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 13, 2008 2:38 PM in response to: sp009 Reply

> > A test case showing what is happening would be
> great
> > - if for no reason other than to satisfy curiosity
> > about how things work, and how things may have
> > changed from one version to another.
> Charles,
>
> I have 8k and 16k block size database. What test case
> are you looking for?
>
> db_file_multiblock_read_count = 8k (default) against
> dynamic in both my database?
>
> sp009

sp009,

I believe that I was in the process of proof-reading my post when you submitted your SQL*Plus output showing that what happens when DB_FILE_MULTIBLOCK_READ_COUNT is set to 0 at the session level. I tried that same set of commands on Oracle 10.2.0.2:

```
SQL> ALTER SESSION SET DB_FILE_MULTIBLOCK_READ_COUNT=0
  2  /

Session altered.

Elapsed: 00:00:01.01
SQL> SHOW PARAMETER DB_FILE_MULTIBLOCK_READ_COUNT

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_file_multiblock_read_count        integer     1
```

On 10.2.0.2, changing the parameter to 0 causes it to change to 1. It is good to see that the effect of this parameter change has been improved.

Looking at one of the links that you provided:
http://download.oracle.com/docs/cd/B19306_01/server.102/b14211/whatsnew.htm#PFGRF000
"The DB_FILE_MULTIBLOCK_READ_COUNT initialization parameter is now automatically tuned to use a default value ***when this parameter is not set explicitly****."* This quote does not suggest setting the value to 0 to unset the value of DB_FILE_MULTIBLOCK_READ_COUNT, although that appears to now be the behavior on Oracle 10.2.0.4.

Looking at the second link that you provided:
http://download.oracle.com/docs/cd/B19306_01/server.102/b14211/optimops.htm#BABDECGJ
"The optimizer uses the value of DB_FILE_MULTIBLOCK_READ_COUNT to cost full table scans and index fast full scans. Larger values result in a cheaper cost for full table scans and can result in the optimizer choosing a full table scan over an index scan. If this parameter is not set explicitly (or is set is 0), the optimizer will use a default value of 8 when costing full table scans and index fast full scans." I believe that quote, and a comment on that quote is already present in this thread regarding the accuracy of this particular paragraph.

Thanks for the information that you provided. I previously suggested the following for a test case to see why performance may have changed when the value of DB_FILE_MULTIBLOCK_READ_COUNT was automatically set (your answer to the last bullet point helped):
Some starting points for observation of the above:
** I believe that you stated that your database is using a 16KB block size, and 16KB * 128 is what value?*
** The maximum read size on most platforms is what value, and how does it compare with the above calculation?*
** The extent size of the objects (when you were testing) are what size: 32KB, 64KB, 512KB, 10MB, 100MB? Jonathan Lewis explained the significance of the extent size in this thread, Greg Rahn confirmed the significance, and I was able to confirm it through examination of trace files.*
** How many of the database blocks were already in the buffer cache when you were testing?*
** How many blocks did the raw trace file show that Oracle was reading at one time when db_file_multiblock_read_count was set to 0 and 128?*
** Did you look at a 10053 trace file to determine why the cost was nearly the same when db_file_multiblock_read_count was set to 0 and 128?*
** When db_file_multiblock_read_count was set to 0, what did Oracle automatically set that parameter's value to?*

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 13, 2008 5:18 PM in response to: user619401

Reply

For all of you who have been following and replying on this thread, I would ask that you look at a blog article I made today regarding a recent situation. The link is:

http://www.oraclealchemist.com/oracle/hey-guys-does-size-matter/

As I mention in the article, I have not finished analyzing all of the collected data from this situation, but I would still appreciate any commentary, questions, etc.

---

benprusinski

Posts: 207
From: San Diego, CA
Registered: 2/1/00

Re: Larger vs. Small data block
Posted: Jun 13, 2008 6:12 PM in response to: user619401

Reply

Hi all,

My experiences have been that using different block sizes can make a difference.

For a past customer a large financial company, we improved database performance by increasing block size from 8k blocksize to 16k blocksize.
Performance for nightly data loads went down from 22 hours to 6 hours when we increased the database block size.

Full table scans benefit from larger block size based on what I seen in a data warehouse environment. Even the Oracle Database 10g Performance Tuning Guide mentions this in Chapter 8, Pages 8-1 through 8-10 that large block sizes are recommended for data warehouse environments and smaller block sizes usually are best for OLTP database environments with Oracle.

In fact when I took the Oracle 9i Database Performance Tuning course years ago at Oracle University the course materials and instructor recommend that block sizes affect performance!

There are always rare exceptions just like a broken clock can be right twice a day.
However I prefer to stick to guidelines and find the solutions that work for majority of customers that I deal with rather than the 1 out of a million exceptions.

Regards,
Ben Prusinski

My Blog on Database Technology
http://oracle-magician.blogspot.com/

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 13, 2008 11:55 PM in response to: Steve Karam

Reply

One thought that immediately comes to mind is that your export/import changed the data in ways previously discussed by Jonathan Lewis.

A better test would be to take a single export and then import it into separate but equal databases with no difference other than the block size.

---

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 14, 2008 12:03 AM in response to: benprusinski

Reply

This is what I don't like about unscientific, anecdotal, information.

Steve Karam reports:
"By going from a 16k blocksize to a 4k blocksize with all other things being equal, we experienced roughly a twenty times improvement."

and you report:
"For a past customer a large financial company, we improved database performance by increasing block size from 8k blocksize to 16k blocksize."

So one of you gets improved performance using smaller blocks the other by using larger blocks. From this a DBA trying to make a decision on what to do with their system should draw what conclusion? Throw a coin in the air and call heads or tails?

The lesson I draw is that under specific conditions with specific workloads it is possible to achieve differences, unpredictable differences, by arbitrarily changing the block size.
Thus the only thing for a DBA to do, given time and bandwidth, is to build their application using multiple blocksizes and test each and every one. I don't know anyone in a corporate environment with that luxury.

And while stating this it should also be noted that while both your test and Steve's relate to a single query ... not to the entire workload on a production system. The only lesson learned here is that there are no silver bullets.

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 14, 2008 1:01 AM in response to: benprusinski

Reply

Hi Ben,

Want to thank you and Steve for providing your stories. Additional anecdotal data points can be useful, just like the Oracle one above.

Conclusion so far - people still need to benchmark in their own environment. But I like the fact that we now have 3 referenceable stories - two that say a change in block size has a noticeable effect (at least for specific key operations), and one for OLTP that leaves block size at the default size.

> Full table scans benefit from larger block size based on
> what I seen in a data warehouse environment. Even the
> Oracle Database 10g Performance Tuning Guide mentions
> this in Chapter 8, Pages 8-1 through 8-10 that large
> block sizes are recommended for data warehouse
> environments and smaller block sizes usually are best
> for OLTP database environments with Oracle.

Your comment around the Performance tuning guide is interesting. I've looked several times and I seem to keep missing miss the 'large block sizes are recommended ...' and 'smaller block sizes usually are best ...' comments. What I did find was:

"8.2.6 Choosing Data Block Size

A block size of 8K is optimal for most systems. However, OLTP systems *occasionally* use smaller block sizes and DSS systems *occasionally* use larger block sizes. "

Perhaps you could help me find the 'recommended' and 'usually best' qualifiers.

As for the Oracle9i class material, on Page 15-32 we read:

Block Size

Data warehouse applications typically perform many table scans; therefore consider a higher value for the block size parameter.

but that has been removed in the 10g course, except for a brief statement in the summary about Block SIze in chapter 15. In that it definitely states the normal OLTP environment is default block size, although ROLAP might benefit from increasing it. (MOLAP is a completely different beast, being BLOB based.)

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 14, 2008 1:20 AM in response to: damorgan

Reply

> So one of you gets improved performance using smaller
> blocks the other by using larger blocks. From this a
> DBA trying to make a decision on what to do with
> their system should draw what conclusion? Throw a
> coin in the air and call heads or tails?

Which is why I say that there is no conclusion as of yet. I would never say "small blocks are better, und das ist alles." But acknowledging that yes, block size CAN produce a (possibly sizeable) difference, we can make sure to gather information based upon block size as a variable in the future.

Is it not worthwhile to share our experiences, all of our experiences, in the hopes that we may find a common thread? If it is necessary to add a disclaimer saying "this proves nothing definitively" then so be it. But just noticing the contradictions between my test and Ben's test is a start. You call it unpredictable, I call it "Steve Results != Ben Results", which is the start of a formula. Now we drill down and find out why. Are they the same Oracle version? What parameters are different? How is his I/O subsystem configured? And so on.

> The lesson I draw is that under specific conditions
> with specific workloads it is possible to achieve
> differences, unpredictable differences, by
> arbitrarily changing the block size.

At the beginning of this thread you said "If you implement any block size other than 8K your benefits, if any, will be marginal and your risks greater." Well, I just saw a situation where all application queries ran the same if not better, and DML performance increased between 20 and 270 times. I'm not saying it's perfect or conclusive, but isn't it worth considering and investigating?

> Thus the only thing for a DBA to do, given time and
> bandwidth, is to build their application using
> multiple blocksizes and test each and every one. I
> don't know anyone in a corporate environment with
> that luxury.

Yes, you're right, that would not be possible in nearly any case. But as I mentioned before, if we discuss experiences (including those that go against conventional wisdom), we can hopefully start to notice trends. If, after a few tests, we disregard all future findings because it was found irrelevant at some point, we may miss out on something worthwhile.

> And while stating this it should also be noted that
> while both your test and Steve's relate to a single
> query ... not to the entire workload on a production
> system. The only lesson learned here is that there
> are no silver bullets.

Okay, so there are no silver bullets that apply 100% of the time to 100% of systems, I get that. Everything is relative, right? But let's say something works for you 20% of the time; it is still worth investigating the boundaries for a successful run rather than disregarding it entirely. Isn't it?

Maybe I'm too much of an idealist. Maybe that's why I chose the "Oracle Alchemist" nom de plume. I'm very aware of the need to prove, the need to find root cause, and the need to find the 'proper' solution. But nothing says a hunch can't help you get there. To each their own.

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 14, 2008 1:22 AM in response to: Hans Forbrich

Reply

> But I like the fact that we
> now have 3 referenceable stories - two that say a
> change in block size has a noticeable effect (at
> least for specific key operations), and one for OLTP
> that leaves block size at the default size.

Exactly!

Hemant K Chitale

Posts: 1,259
Registered: 11/6/98

Re: Larger vs. Small data block
Posted: Jun 14, 2008 3:26 AM in response to: Hans Forbrich

Reply

>>"8.2.6 Choosing Data Block Size
>>
>>A block size of 8K is optimal for most systems. However, OLTP systems >>occasionally use smaller block sizes and DSS systems occasionally use larger block >>sizes. "

Once upon a time, the default block size was 2K. Right upto 8i, Oracle would
create a database with a 2K block size. We had to manually set db_block_size=8192
before running an SQL script to CREATE DATABASE.

So, now, most people think that an 8K block size is optimal. What happened in the
intervening years ? Technology changed and 8K reads and multiblock reads
of 1MB were possible. CPU speeds improved and latch time and updating rows
in a block became faster (the main issue with larger block sizes was contention
amongst sessions for rows in the same block). Those improvements made 8K
block sizes sensible. Oracle and some DataWarehouse DBAs have seen
environments where 8K performed better in the 2K days. Surely, there are

environments where 16K performs better in the 8K days ?

Billy Verreynne

Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 14, 2008 8:03 AM in response to: benprusinski
Reply

> Performance for nightly data loads went down from 22 hours to 6
> hours when we increased the database block size.

Care to back that up with something tangible?

If not, then your claim is no different than the following.. that faith in, and prayer to, the Flying Spaghetti Monster, guarantees an increase in Oracle performance.

```
SQL> set timing on
SQL> select count(*) from all_objects;

  COUNT(*)
----------
     10332

Elapsed: 00:00:04.61
SQL> -- praying hard to the Spaghedeity
SQL> select count(*) from all_objects;

  COUNT(*)
----------
     10332

Elapsed: 00:00:00.29
SQL> -- it worked!! as usual - praying to the Spaghedeity increased performce again!!
SQL> -- praised be His Noodly Goodness!! Amen.
SQL>
```

No offence.. but I'm kind of sick and tired of the xxxxxxxx style of faith-based Oracle performance claims.

You claim that "something" improves Oracle performance? THEN BACK IT UP WITH ACTUAL EVIDENCE AND PROOF!

Or else just shut up.

PS. To His Noodly Goodness. I know You said *"I'd really rather you didn't challenge the bigoted, misogynistic, hateful ideas of others on an empty stomach. Eat, then go after the bitches"*. I have not only eaten. I've had 2 (or has it been 3 already?) mugs of coffee and a very serious Mug & Bean Cuppachino. My lead pipe has been waxed shiny. I'm ready to go after them.

Billy Verreynne

Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 14, 2008 8:24 AM in response to: Billy Verreynne
Reply

Now I know someone with warm fuzzies to all mankind and especially faith-based Oracle performance tuning that deems empirical observation and anecdotal evidence to suffice for the unwashed (non official Oracle DBA) masses, will take me on.

So to preempt that and save that poor sod from whacking his (or her or it's) keyboard in churning out a "brilliant" response to me...

All I want is *something* to back up your claim. Like I/O was reduced by 80% because of ABC. Slow random single block reads were replaced by fast multi-block sequential reads. I/O thruput was improved because of XYZ.

Throw me a damn bone please if you do not want me and my lead pipe to go after yours.

Thanks! :-)

oradba

Posts: 5,591
From: Germany
Registered: 9/15/00

Re: Larger vs. Small data block
Posted: Jun 14, 2008 8:25 AM in response to: Billy Verreynne
Reply

IMO salary increase is much more important than blocksize increase *LOL*

(I think this thread is ready for Guinness Worldrecords very soon!
http://www.guinnessworldrecords.com/default.aspx)

Werner

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 14, 2008 9:24 AM in response to: Steve Karam
Reply

Your change was to one query at one point in time. How did the change affect the entire system for all workloads? How did the system perform three months later? Block sizes are not something you can arbitrarily change like, for example, an adjustment to cursor sharing.

Also, if performance changed it changed for a reason. Which metric(s) changed, or how did the plan change, such that performance improved. Please be specific.

My testing, under rigorous conditions, has shown marginal differences except in contrived conditions. We've not seen your query and its trace so we have no reason to believe it is not a special case or that it is not. We don't know if the next point-release patch caused a change that might have not happened with a standard 8K block. What we have, not to in any way denigrate your work or the anecdotal stories of others, is roughly equivalent to "I put ice cubes under my armpits and lived to be 100." Ok but was it the ice cubes that did it? I remain wholly unconvinced that in the vast majority of situations with the vast majority of applications is makes a measurable and sustained difference.

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 14, 2008 9:30 AM in response to: Hemant K Chitale
Reply

> >>"8.2.6 Choosing Data Block Size
> >>
> >>A block size of 8K is optimal for most systems. However, OLTP systems
> >>occasionally use smaller block sizes and DSS systems occasionally use larger
> >> block sizes. "

>
...
> So, now, most people think that an 8K block size is optimal.

Not sure where your get 'most people' from.

Just so we are clear, I copied the 'optimal' information directly from the Performance Tuning manual. (The quote marks indicate I am quoting ... not putting words in anyone's mouth.<g>)

> What happened in the intervening years ? Technology changed and 8K reads
> and multiblock reads of 1MB were possible. CPU speeds improved and latch
> time and updating rows in a block became faster (the main issue with larger
> block sizes was contention amongst sessions for rows in the same block). Those
> improvements made 8K block sizes sensible. Oracle and some DataWarehouse
> DBAs have seen environments where 8K performed better in the 2K
> days. Surely, there are environments where 16K performs better in the 8K days ?

Yes, the Performance Tuning manual, in the section I references (and provided a link to) supports exactly this.

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 14, 2008 9:46 AM in response to: Steve Karam — Reply

> For all of you who have been following and replying
> on this thread, I would ask that you look at a blog
> article I made today regarding a recent situation.
> The link is:
>
> http://www.oraclealchemist.com/oracle/hey-guys-does-si
> ze-matter/
>
> As I mention in the article, I have not finished
> analyzing all of the collected data from this
> situation, but I would still appreciate any
> commentary, questions, etc.

Steve,

Your blog entry is an interesting write up - thanks for sharing. I read through it several times, asking myself if there is any possible other explanation than the change in the block size - or is there anything that supports the performance change due to the change in the block size. I did not come up with much after about 2 hours looking at what was written.

Below are my comments, questions, and efforts at shooting in the dark. Someone with more experience might be able to construct a better list:

* Different plans on SYS owned objects, is it possible that statistics on SYS owned objects were collected in one of the databases, but not the other? (Bug No 3919772 for 9.2.0.5 might contain a useful explanation)
* Did both databases have locally managed or dictionary managed tablespaces?
* Is it possible that the temp tablespace in the 16KB block database was created as a permanent tablespace?
* The 16KB and 4KB database instances existed at the same time on the server - so they did not using the same areas of the disks (it can make a difference).
* Were the trace files manually examined, or sent through TKPROF? What wait events did you see in the trace files?
* Were there any indexes on the two column table?
* Is there a trigger or foreign key on the column being updated?
* How does the redo generation compare between the two databases - is it possible that the 16KB block size database was writing the entire 16KB block to the redo logs, while the 4KB database only wrote the before and after images of changes to the log files (for example, a hot backup using ALTER TABLESPACE x BEGIN BACKUP was started)?
* Reference Bug No 4260477 (reported in 9.2.0.5, fixed in 10.2), indicates that there are problems with inserting/deleting (and possibly updating) a large number of rows in a single block within a single transaction with 32KB block size. It might be interesting to see if it also applies to a table with 2 columns in a 16KB block size tablespace.
* It might be interesting to examine memory accesses. Due to memory latencies and the time difference to transfer data through the bus to the CPU, a 4KB random memory read will complete faster than a 16KB random memory read. If nearly every row being updated required in a different 16KB block(s) to be read from system memory, that might lead to some of the performance difference. More of these random blocks will fit into the lower latency L1, L2, and L3 caches on the CPUs (it might be intesting to see if the 8 CPUs caused problems).
* It appears that DB_BLOCK_CHECKING checks the entire block during an insert or update:
http://download.oracle.com/docs/cd/B10500_01/server.920/a96536/ch135.htm#1015830
"Oracle checks a block by going through the data on the block, making sure it is self-consistent. Block checking can often prevent memory and data corruption. Block checking typically causes 1% to 10% overhead, depending on workload. The more updates or inserts in a workload, the more expensive it is to turn on block checking. You should set DB_BLOCK_CHECKING to true if the performance overhead is acceptable."
* With a consistent 128KB extent size (segment size?), what was the setting for DB_FILE_MULTIBLOCK_READ_COUNT? "Comparing the initialization parameters between production and development showed the exact same parameters, except that the upcoming production box was using a 16k block size and development was using a 4k block size." "Explain plans were checked, trace files examined, and not much popped up except that the production machine was attempting larger I/Os during the update and was consequently taking much longer." There might be something here, is it possible to list the initialization parameters?
* You found a couple OS settings that were causing problems - could it be that there were other OS settings that were not quite right?
* Why did you select to change from a 16KB block size to a 4KB block size, and not something else such as 32KB or 8KB?
* The export and import process may have had a big effect. Did you pre-size the new data files large enough to contain the expected data, or create them small and allow them to grow as needed?
* Is it possible that the large number of CPUs and limited number of disks contributed to the problem?

It is interesting to ask if the change in the block size was the only change.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Hemant K Chitale

Posts: 1,259
Registered: 11/6/98

Re: Larger vs. Small data block
Posted: Jun 14, 2008 10:08 AM in response to: Hans Forbrich — Reply

"most people" would mean the Documentation and the majority of "experts" (those
with more than a few years of experience *and* on different platforms and for
different applications) in their opinions expressed on forums.oracle.com or email
discussion lists or other internet sites.

(and, yes, I acknowledge that you quoted from the Documentation).

My point is just as technological changes made 8K better sense than 2K, in some
applications (ie usages of oracle) a different block size may well make sense.
Probably, multiple block sizes within the same database are better for specific
implementations (other than the normally bandied "transportable tablespaces").

Who knows, 5 years from now, 16K might become the consensus.

So, we may well be betttter of qualifying "universal truths".

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 14, 2008 10:46 AM in response to: damorgan

Reply

> Also, if performance change it changed for a reason.
> Which metric(s) changed, or how did the plan change,
> such that performance improved. Please be specific.

I agree, and that's why I said that I am not done going through the results that I have. Here are some differences I've noticed thus far:

On the **16k blocksize instance**, this occurred 53 times:

```
PARSING IN CURSOR #2 len=36 dep=1 uid=0 oct=3 lid=0 tim=1184884649414850 hv=1254950678 ad='cdf837a8'
select file# from file$ where ts#=:1
END OF STMT
PARSE #2:c=0,e=89,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1184884649414840
EXEC #2:c=1000,e=63,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1184884649415042
FETCH #2:c=0,e=42,p=0,cr=3,cu=0,mis=0,r=1,dep=1,og=4,tim=1184884649415104
FETCH #2:c=0,e=5,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,tim=1184884649415127
FETCH #2:c=0,e=5,p=0,cr=1,cu=0,mis=0,r=0,dep=1,og=4,tim=1184884649415150
STAT #2 id=1 cnt=2 pid=0 pos=1 obj=17 op='TABLE ACCESS FULL FILE$ '
```

The final UPDATE is seen here:

```
EXEC #1:c=1822034009,e=1779788042,p=768,cr=1541885,cu=446195350,mis=0,r=829484,dep=0,og=4,tim=1184886221334077
STAT #1 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE '
STAT #1 id=2 cnt=829484 pid=1 pos=1 obj=30263 op='TABLE ACCESS FULL ***** '
XCTEND rlbk=0, rd_only=0
```

On the **4k blocksize instance**, FILE$ access was done using the FILE_I2 index, and it occurred 518 times:

```
PARSING IN CURSOR #2 len=36 dep=1 uid=0 oct=3 lid=0 tim=1184883784927327 hv=1254950678 ad='d0ee3818'
select file# from file$ where ts#=:1
END OF STMT
PARSE #2:c=0,e=25,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1184883784927324
EXEC #2:c=0,e=26,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1184883784927407
FETCH #2:c=0,e=14,p=0,cr=2,cu=0,mis=0,r=1,dep=1,og=4,tim=1184883784927437
FETCH #2:c=0,e=4,p=0,cr=1,cu=0,mis=0,r=0,dep=1,og=4,tim=1184883784927457
STAT #2 id=1 cnt=1 pid=0 pos=1 obj=17 op='TABLE ACCESS BY INDEX ROWID FILE$ '
STAT #2 id=2 cnt=1 pid=1 pos=1 obj=42 op='INDEX RANGE SCAN I_FILE2 '
```

The final update on the 4k environment looks somewhat the same, but much faster:

```
EXEC #1:c=8924643,e=10332483,p=0,cr=12681,cu=2219343,mis=0,r=829484,dep=0,og=4,tim=1184883857599857
STAT #1 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE '
STAT #1 id=2 cnt=829484 pid=1 pos=1 obj=27448 op='TABLE ACCESS FULL ***** '
XCTEND rlbk=0, rd_only=0
```

I have yet to fully explore all I/O sizing information beyond what the customer has disclosed. As noted in the documentation for 9i, since this is a 9i system (http://download.oracle.com/docs/cd/B10501_01/server.920/a96533/iodesign.htm#33483), this could be an issue of I/O size.

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 14, 2008 11:06 AM in response to: Charles Hooper

Reply

Charles, thank you, and great questions all. I'm going to answer your questions without quoting the original to save on space. Hopefully I get the numbering right. ;)

* I had gathered against both
* Local
* No
* True
* Working on that
* No
* No
* Not possible, v$backup was consulted
* Good call, I'll check that bug out
* This would definitely be interesting, but do you really think it would result in that much of a difference?
* Yes, the client has decided to keep it on. This is when you ask yourself, do you feel lucky? Well, do ya?
* The DBFMRC was tested at 8 and 16 on both environments, but I'm not giving up on that parameter yet either.
* It could be! That's why I'm not ready to say anything conclusively yet.
* Their development system performed much better, and the only real difference was a 4k blocksize. It was worth seeing if that WAS the difference. If we had gone to a 4k blocksize on the poorly performing system, and everything was the same as dev (but slower), we would be able to disregard blocksize as a factor and focus on other things.
* Created them small and allowed them to grow as needed. I am planning on doing a new test with freshly imported tables on both instances if possible. See my note below, this was for the exp/imp test, but there were others.
* Perhaps. This is my current line of thought when looking into this.

Just one other note...before testing a new instance I tried creating a 4k blocksize tablespace in the 16k instance. I did a CTAS to the new (4k) tablespace and a CTAS to another table in a normal tablespace. The results were consistent, the 16k blockszie tablespace took roughly 40 minutes during that test, the 4k blocksize tablespace took roughly 2.5 minutes. That's why I don't think it was an exp/imp issue at this point.

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 14, 2008 11:12 AM in response to: Billy Verreynne

Reply

Wow, if I had a dime every time a crazed spaghetti worshiper threatened to come after me with a lead pipe on a technical forum I'd have...

1 dime.

benprusinski

Posts: 207
From: San Diego, CA
Registered: 2/1/00

Re: Larger vs. Small data block
Posted: Jun 14, 2008 12:16 PM in response to: Billy Verreynne

Reply

>>>> Care to back that up with something tangible?

Due to NDA and confidential nature of the data for the past client, I cannot disclose the actual data and test results and it was a few years ago. Tell ya what, I am going to create some test cases just for you Billy Boy to make you happy when I get a free moment.

But it will not be right this second and making rude comments to others on this forum is pretty disrespectful so I am not in a rush to drop everything and do the testing right this second.

Cheers,
Ben

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 14, 2008 12:33 PM in response to: Steve Karam

Reply

```
> EXEC #1:c=1822034009,e=1779788042,p=768,cr=1541885,cu=446195350,mis=0,r=829484,dep=0,og=4,tim=1184886221334077
```

Unless you've uncovered an exotic bug, I don't think this has anything to do with I/O. You appear to have done virtually no I/O, and (allowing for granularity errors) you have CPU time = elapsed time.

The anomaly is the huge number of current gets (cu=446 million). Your cu count should only be slightly larger than the number of row entries update (where row entries also has to allow for index updates - were there any indexes on the table, and were any of them updated at the same time: the statistics on the 4K test suggest there may have been one that was subject to updates).

I would look at the state of the index (if there is one), and think about effects of delayed block cleanout. (There is an anomaly with excessive delayed block cleanout on large tablescans that could be responsible for some of your overhead - it would be accompanied by excessive redo generation).

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 14, 2008 1:09 PM in response to: Jonathan Lewis

Reply

> Unless you've uncovered an exotic bug, I don't think
> this has anything to do with I/O. You appear to have
> done virtually no I/O, and (allowing for granularity
> errors) you have CPU time = elapsed time.

That's consistent with what I've seen. There was virtually 0 I/O contention/usage at the time of the run.

> The anomaly is the huge number of current gets
> (cu=446 million). Your cu count should only be
> slightly larger than the number of row entries update
> (where row entries also has to allow for index
> updates - were there any indexes on the table, and
> were any of them updated at the same time: the
> statistics on the 4K test suggest there may have been
> one that was subject to updates).

There we no indexes on either test table.

> I would look at the state of the index (if there is
> one), and think about effects of delayed block
> cleanout. (There is an anomaly with excessive
> delayed block cleanout on large tablescans that could
> be responsible for some of your overhead - it would
> be accompanied by excessive redo generation).

As a matter of fact, on the 16k blocksize there were a fair amount of log file switch completion waits appearing here and there.

The observations from you and Charles have prompted me to take another look at the objects in question to make sure there are no inconsistencies that I might have missed on the first go-round. If I notice anything out of the ordinary I'll post back (at least so far as I can under the contract).

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 14, 2008 1:13 PM in response to: Hemant K Chitale

Reply

> Who knows, 5 years from now, 16K might become the consensus.

The block size, and everything else we set, is based on stated and unstated (and in some cases, unknown) assumptions.

>
> So, we may well be bettter of qualifying "universal truths".

In 5 years, we'll probably have Oracle 12e. The technology and the size of disk/memory/CPU cache/whatever will have changed sufficiently that the assumptions will no longer be valid.

So the next set of myths are underway.

oradba

Posts: 5,591
From: Germany
Registered: 9/15/00

Re: Larger vs. Small data block
Posted: Jun 14, 2008 2:50 PM in response to: Hans Forbrich

Reply

Honni soit qui mal y pense

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 14, 2008 3:27 PM in response to: Steve Karam

Reply

Steve,

** It might be interesting to examine memory accesses. Due to memory latencies and the time difference to transfer data through the bus to the CPU, a 4KB random memory read will complete faster than a 16KB random memory read. If nearly every row being* *updated required in a different 16KB block(s) to be read from system memory, that might lead to some of the performance difference. More of these random blocks will fit into the lower latency L1, L2, and L3 caches on the CPUs (it might be intesting to see if the 8 CPUs caused problems).*

> * This would definitely be interesting, but do you
> really think it would result in that much of a
> difference?

I had a little free time one day and starting running some calculations. From one of my Usenet posts, just running numbers (assumes 0 memory access latency):
*Interesting fun with mathematics, which may not be entirely relevant.
On a computer with a computer marketed as having a 1333MHz bus speed, using 333MHz quad pumped dual channel memory chips, each memory clock cycle retrieves up to 32 bytes in 0.000000003003003 seconds (maximum transfer speed of 10,162.35 MB per second), and the CPU core will wait for this duration on every memory access. A standard 8KB block requires a minimum of 256 memory clock cycles to be read, resulting in a minimum delay of 0.000000768768768 seconds to read an 8KB block from system memory. If you require the computer to perform 180,000 8KB reads (assuming the data is not cached in the CPU registers, L1, L2, or L3 caches), it will take a minimum of 0.138 seconds (consistent reads of 8KB blocks might take 5-10 times longer). What seems like a simple problem becomes a bit complicated when you dig into the details.*

Exploring latency of L1, L2, and L3 caches on a soon to be released Intel CPU (Nehalem):

http://www.anandtech.com/cpuchipsets/intel/showdoc.aspx?i=3326&p=5

CPU / CPU-Z Latency	L1 Cache	L2 Cache	L3 Cache
Nehalem (2.66GHz)	4 cycles	11 cycles	39 cycles
Core 2 Quad Q9450 - Penryn - (2.66GHz)	3 cycles	15 cycles	N/A

Exploring the latency of system memory access:
http://www.extremetech.com/article2/0,2845,2218447,00.asp
Memory read speeds
http://www.extremetech.com/print_article2/0,1217,a%253D133743,00.asp
Does RAM Latency Matter

The effects of memory access latency might be visible in the trace file extracts that you posted of the dep=1 recursive calls:

```
16KB
 tim= D
 0.00000  PARSING IN CURSOR #2 len=36 dep=1 uid=0 oct=3 lid=0 tim=1184884649414850 hv=1254950678 ad='cdf837a8'
          select file# from file$ where ts#=:1
          END OF STMT
-0.00001  PARSE #2:c=0,e=89,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1184884649414840
 0.00019  EXEC #2:c=1000,e=63,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1184884649415042
 0.00025  FETCH #2:c=0,e=42,p=0,cr=3,cu=0,mis=0,r=1,dep=1,og=4,tim=1184884649415104
 0.00027  FETCH #2:c=0,e=5,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,tim=1184884649415127
 0.00030  FETCH #2:c=0,e=5,p=0,cr=1,cu=0,mis=0,r=0,dep=1,og=4,tim=1184884649415150
          STAT #2 id=1 cnt=2 pid=0 pos=1 obj=17 op='TABLE ACCESS FULL FILE$
-------------------------------------------------------
4KB
 tim= D
 0.00000  PARSING IN CURSOR #2 len=36 dep=1 uid=0 oct=3 lid=0 tim=1184883784927327 hv=1254950678 ad='d0ee3818'
          select file# from file$ where ts#=:1
          END OF STMT
 0.00000  PARSE #2:c=0,e=25,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1184883784927324
 0.00008  EXEC #2:c=0,e=26,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1184883784927407
 0.00011  FETCH #2:c=0,e=14,p=0,cr=2,cu=0,mis=0,r=1,dep=1,og=4,tim=1184883784927437
 0.00013  FETCH #2:c=0,e=4,p=0,cr=1,cu=0,mis=0,r=0,dep=1,og=4,tim=1184883784927457
          STAT #2 id=1 cnt=1 pid=0 pos=1 obj=17 op='TABLE ACCESS BY INDEX ROWID FILE$ '
          STAT #2 id=2 cnt=1 pid=1 pos=1 obj=42 op='INDEX RANGE SCAN I_FILE2 '
```

Note that the final tim= D value for the 16KB trace is roughly 3 times the value of the final tim= D value for the 4KB trace. I can't say whether or not this is due to memory access latency, but it is interesting to see that the tim= D value on the EXEC line for the 16KB trace is twice that for the 4KB trace.

What might be interesting is the line containing "c=1000,e=63". Considering that there are 8 CPUs, and the c= value is about 16 times greater than the e= value - I thought in such a situation, the maximum value of any c= value is the e= value multiplied by the number of CPUs. I could be wrong. It might have been helpful to have captured the 10046 trace file at level 8 or 12 to determine what wait events may have contributed

Note that memory latency is not the only problem. As Jonathan pointed out, the consistent reads plus the current reads is 447,737,235 in the 16KB database, but only 2,232,024 in the 4KB database. At the maximum memory speed (no latency) per my Usenet post, it would take 688 seconds (11.4 minutes) to read that number of 16KB blocks, compared with 0.85 seconds to read that number of 4KB blocks. It might be helpful to determine what caused all of the CR and CU memory reads.

Are you able to post any of the initialization parameters, such as db_writer_processes? Kevin Closson posted a series of articles some time ago that describe how the value of that parameter might cause problems for the L1, L2, and L3 caches on CPUs - here are a couple of the articles:
http://kevinclosson.wordpress.com/2007/08/10/learn-how-to-obliterate-processor-caches-configure-lots-and-lots-of-dbwr-processes/
http://kevinclosson.wordpress.com/2007/08/17/over-configuring-dbwr-processes-part-ii/

> Just one other note...before testing a new instance I
> tried creating a 4k blocksize tablespace in the 16k
> instance. I did a CTAS to the new (4k) tablespace
> and a CTAS to another table in a normal tablespace.
> The results were consistent, the 16k blockszie
> tablespace took roughly 40 minutes during that test,
> the 4k blocksize tablespace took roughly 2.5
> minutes. That's why I don't think it was an exp/imp
> issue at this point.

Would doing the above (CTAS) compact the data into potentially fewer blocks (more rows per block)? Assuming that the second column in the table contained a very small value (or was NULL), might there have been a greater chance of row migration in the 16KB tablespace during the update as the rows expanded in size?

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Steve Karam

Re: Larger vs. Small data block
Posted: Jun 14, 2008 7:57 PM in response to: Charles Hooper

Reply

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

> I had a little free time one day and starting running
> some calculations.

Your calculations sound very interesting, I'll have to check out your findings soon.

> What might be interesting is the line containing
> "c=1000,e=63". Considering that there are 8 CPUs,
> and the c= value is about 16 times greater than the
> e= value - I thought in such a situation, the maximum
> value of any c= value is the e= value multiplied by
> the number of CPUs. I could be wrong. It might have
> been helpful to have captured the 10046 trace file at
> level 8 or 12 to determine what wait events may have
> contributed

I have those, but unfortunately I'm not at liberty to share them. As I mentioned to Jonathan however, I did notice a good deal of log buffer switch completion wait on the 16k trials.

> Note that memory latency is not the only problem. As
> Jonathan pointed out, the consistent reads plus the
> current reads is 447,737,235 in the 16KB database,
> but only 2,232,024 in the 4KB database. At the
> maximum memory speed (no latency) per my Usenet post,
> it would take 688 seconds (11.4 minutes) to read
> that number of 16KB blocks, compared with 0.85
> seconds to read that number of 4KB blocks. It might
> be helpful to determine what caused all of the CR and
> CU memory reads.

That's the tack I've been taking, I think you're right, it will probably produce the most meaningful results.

> Are you able to post any of the initialization
> parameters, such as db_writer_processes?

I can tell you that parameter is unset. I'm sorry, but I can't disclose the initialization parameters in full or much more detail than that. I was given some leeway by the client, but not much!

> Would doing the above (CTAS) compact the data into
> potentially fewer blocks (more rows per block)?
> Assuming that the second column in the table
> contained a very small value (or was NULL), might
> there have been a greater chance of row migration in
> the 16KB tablespace during the update as the rows
> expanded in size?

Definitely a possible what if. It might be worth an extra test or two. However, these results were consistent not only for that one update, but all DML testing we performed. That was against both existing objects and newly created objects. Their development environment, which was the same except for 1) 32-bit and 2) the 4k blocksize, was consistent as well without any CTAS or exp/imp necessary.

Out of curiosity, does any of this diminish the fact that for this client on this server on this Oracle version on this word size on this architecture on their app, going from 16k to 4k produced a sizeable diffrerence on DML and the same or better performance on queries? I consider it my duty to determine the actual reason why the change made a difference for my client (and I enjoy doing so as well), but at the same time a 270x can't be written off due to conventional wisdom. I know my client doesn't think so.

I'll check back on Monday, it's time for me to enjoy my Father's Day. To any fathers on the thread, I hope you enjoy yours as well!

(No, you cannot claim to be the father of your database)

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 15, 2008 5:52 AM in response to: Steve Karam

Reply

>
> Out of curiosity, does any of this diminish the fact
> that for this client on this server on this Oracle
> version on this word size on this architecture on
> their app, going from 16k to 4k produced a sizeable
> diffrerence on DML and the same or better performance
> on queries? I consider it my duty to determine the
> actual reason why the change made a difference for my
> client (and I enjoy doing so as well), but at the
> same time a 270x can't be written off due to
> conventional wisdom. I know my client doesn't think
> so.
>

At present, based on the evidence you have supplied, it's NOT a **fact** that "going from 16K to 4K" produced a sizeable difference in DML.

At best we have a fact that producing a clean copy of the data somewhere else resulted in better performance on that update.

In fact the evidence suggests that the change in block size was probably irrelevant given the enormous change in the number of current block gets and redo log generation. It is possible that you've highlighted a defect in the way ASSM handles free space; and it is possible that this is a problem that becomes more visible with your update, especially when combined with the 16K block size, and combined with an error in the initial table definition - and maybe it's all down to an error in the initial table definition.

Based on the evidence to date, I would not advise the client to move his system to a 4KB block size - after all, what's the next step going to be if and when (in three months time, say) the performance on the 4KB block size is as bad as it currently is on the 16KB block size ?

What's the average length of the columns involved ?
Does the update change the length of the column; in particular does it take the column from null to non-null ?
How many rows are there in the table in total, and is the 830,000 a fairly constant number, or a fairly constant percentage of the total ?
Are the updated rows scattered throughout the table, or are they mostly at the end of the table.
Do rows get deleted in bulk after a while ?

These are all questions that the system designer should have thought about - and then maybe the problem wouldn't exist because (for example) a suitable value for pctfree would have been chosen from the outset.

If you're allowed to give the answers to these question (and tell use the size of pctfree) then that would be helpful.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 15, 2008 6:11 AM in response to: Jonathan Lewis
Reply

Your Hawking quote is both priceless and appropriate. It succinctly summarizes the most important feature of this thread, in a single sentence, better than all of the reasoned arguments so far made.

What we have been witnessing is the illusion of knowledge. The application of the Aristolean method rather than the scientific method.

What are seeing played out here in this OTN forum thread is, at its essence, the same debate played out in the arguments made against Galileo and Newton.
http://wiki.elearning.ubc.ca/ScientificChange?show_comments=1

benprusinski

Posts: 207
From: San Diego, CA
Registered: 2/1/00

Re: Larger vs. Small data block
Posted: Jun 15, 2008 11:34 AM in response to: damorgan
Reply

This has been very interesting discussion on block size and performance. As for Daniel Morgan's quote about knowledge, I would respectfully agree to disagree.

Steve has provide plenty of data to verify performance improvement for block size changes.
And it does seem to follow the scientific method.

While there could be other factors, the fact is this: performance tuning is not a static matter. It is an ongoing exercise that will and should be conducted on both a short term and long term basis to look at all aspects of performance and what impact each tuning change affects the database performance for the entire database environment.

Like I mentioned in a previous post to this thread, unfortunately, I was not at liberty to disclose confidential information for the actual database parameters and test results for the financial services client where I improved performance by changing the block size for the database. That is why I have taken it on myself to eventually (when I get free time!) create some new test cases. Once I have these, I will gladly post the results on this forum and we can have further discussion.

This has definitely been an interesting thread and I appreciate all the active participation.

Regards,
Ben Prusinski
http://oracle-magician.blogspot.com/

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 15, 2008 11:39 AM in response to: Jonathan Lewis
Reply

> At best we have a fact that producing a clean copy of
> the data somewhere else resulted in better
> performance on that update.

Except that I produced clean copies both in 4k and 16k areas, and the DML performance results were consistent across all tables, not just the one.

> Based on the evidence to date, I would not advise the
> client to move his system to a 4KB block size

Thanks for your input. I had already told the client that rebuilding their entire environment was not advisable until we had conclusively identified the issue, as we would not want to band-aid over a deeper concern. While some on this thread are trying to paint me as some sort of reckless cowboy, I do not take my clients' multi-million dollar investments lightly. We shall see what the client decides based upon their deadlines and the results thus far.

> These are all questions that the system designer
> should have thought about - and then maybe the
> problem wouldn't exist because (for example) a
> suitable value for pctfree would have been chosen
> from the outset.

Personally the first thing I suggested was not doing such a costly update, and instead suggested using a CTAS since it was an update of many rows with no indexes and no where clause. CTAS itself worked very quickly, though updates against the new table performed poorly like the original. However, I was informed that this table was not the only one suffering, but all tables with high levels of DML.

> If you're allowed to give the answers to these
> question (and tell use the size of pctfree) then that
> would be helpful.

Time and client consent permitting, I would like to perform a new set of trials using a clean slate; meaning, new tables that I create manually with proper settings, manually loaded (not CTAS or exp/imp), and tested for all DML activity. If I can make that happen, I will blog about the results.

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 15, 2008 11:43 AM in response to: damorgan
Reply

damorgan:

> What have been witnessing is the illusion
> of knowledge.

Do not be so quick to discard the observations of others as the 'illusion of knowledge.' Doing so is insulting, close-minded, and irresponsible. If you have any scientific data to contribute to the observation that I have made, please feel free to do so. Simply making accusations from the sidelines does not prove or disprove anything.

Personally I love Stephen Hawking, but even he has conceded that currently unexplainable or partially explained observations have their merit (e.g. strings theory, spooky action at a distance).

Why not broaden our horizons?

If we value the pursuit of knowledge, we must be free to follow wherever that search may lead us. The free mind is not a barking dog, to be tethered on a ten-foot chain. - Adlai E. Stevenson Jr.
The dumbest people I know are those who know it all. - Malcolm Forbes
A little knowledge that acts is worth infinitely more than much knowledge that is idle. - Kahlil Gibran
Knowledge must come through action; you can have no test which is not fanciful, save by trial. - Sophocles
Whoever undertakes to set himself up as a judge of Truth and Knowledge is shipwrecked by the laughter of the gods. - Albert Einstein
Imagination is more important than knowledge. For while knowledge defines all we currently know and understand, imagination points to all we might yet discover and create. - Albert Einstein
Knowledge has to be improved, challenged, and increased constantly, or it vanishes. - Peter Drucker
It is beyond a doubt that all our knowledge begins with experience. - Imannuel Kant
That knowledge is not happiness, and science But an exchange of ignorance for that Which is another kind of ignorance. - Lord Byron

Hans Forbrich (...
Posts: 663
From: Alberta, Canada
Registered: 11/17/06

Re: Larger vs. Small data block
Posted: Jun 15, 2008 12:38 PM in response to: oradba
Reply

> Honni soit qui mal y pense

Könnte viele Wege missverstanden werden <g>

Billy Verreynne
Posts: 6,628
Registered: 5/27/99

Re: Larger vs. Small data block
Posted: Jun 15, 2008 1:17 PM in response to: benprusinski
Reply

> Due to NDA and confidential nature of the data for the past client, I cannot disclose the actual data and test
> results and it was a few years ago. Tell ya what, I am going to create some test cases just for you Billy Boy to
>make you happy when I get a free moment.

Not asking anything that may "compromise" a NDA. Simply *what* was observed technically that verified the increase in performance was due to using a larger block size.

Also, seeing that is is a couple of years old, how sure you are that whatever was done and observed that lead to the conclusion that block size made such a large difference is still relevant in 10r2 and 11g?

> But it will not be right this second and making rude comments to others on this forum is pretty
> disrespectful so I am not in a rush to drop everything and do the testing right this second.

Come on Ben.. that posting was done with tongue firmly in cheek. The Flying Spaghetti Monster should have been a clue. And if I was "disrespectful" to anything, it was to an unsubstantiated claim that block size made such a large performance difference.

As I would have been if you claimed that it made no difference. As I would have done if an ace or a noob posted it. (it's never about the poster to me, it is about the posting - unlike some who believe you should post your qualifications, CV and blood line in order to be taken as a valid source of information)

What matters to me (and I believe others here) are what can we technically do, similar to what you did, in order to obtain similar type of performance improvements within a similar environment. Is that an unfair expectation?

If you (or others) cannot back up your claims, then how can anyone realistically expect that such a claim can be considered when making technical decisions?

And that is exactly what these forums are about. The sharing of technical information that is accurate and comprehensive for fellow Oracle "professionals" to use.

damorgan
Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 15, 2008 2:09 PM in response to: benprusinski
Reply

Providing numbers is not science: Aristotle used math too.

What is important is in how the experiment was formulated and the conclusions drawn.

Note, for example, the commentary supplied by Jonathan Lewis in which he posted the Hawking's quotation. Note that there are many possible alternative explanations for a reported experimental result.

What is required to make something science is to create a controlled experiment in which the change to a single parameter can be observed. If you change two or more factors then you lose the ability to identify a clear cause and effect.

Just to make my point clear lets create an experiment. Lets tune a database by doing the following things:
1. Double the value of session cached cursors
2. Half the block size
3. Export the data and reimport it
4. Slowly twirl three times while reciting over-quoted lines from Shakespeare

If performance improves what was the cause?
Is it repeatable?
Is there some fact put into evidence that other DBAs can use as a rule of thumb?

damorgan
Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 15, 2008 2:17 PM in response to: Steve Karam
Reply

Lets take this out of the realm of Oracle databases and analyze the same information in the context of medical science and an experiment in which I have first hand knowledge.

A pharmaceutical company in the 1960s was testing oral contraceptives at Stanford University (a school whose female students are definitely above average in intelligence). One group was given dosage "A" and another group dosage "B". In both cases the amount of drug was the same but one group received 21 active pills and 7 placebos while the other group received 28 active pills. One group reported more pregnancies than the other.

Was the correct interpretation that one dosage was more effective than the second?

My point was not to insult anyone. But rather to point out that numbers, even numbers presented in chart form, are subject to multiple interpretations unless the experimental conditions are carefully controlled.

If you think the answer to the above is "yes" ... I will provide the rest of the story that proves otherwise.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 15, 2008 2:43 PM in response to: benprusinski

Reply

[nobr]>
> Steve has provide plenty of data to verify
> performance improvement for block size changes.

No he hasn't.

All he provided as evidence was something that suggested his performance improvement had nothing to do with the block size and everything to do with the state of the data before the test started.

He also **said** that he also did a test based on copying the data into another tablespace with 16KB blocks - but we have no evidence that he then tested this copy. Possibly he copied the data, and then ran the test against the original.

Remember that Steve said on his blog: *"Explain plans were checked, trace files examined, and not much popped up except that the production machine was attempting larger I/Os during the update and was consequently taking much longer."* then he printed in this thread a line from a trace file without commenting on the 446M current gets for a process that updates 830,000 rows - beyond the fact that it was slower than the update using the 4KB block. Also, after I suggested a cause and corroborating symptom, he confirmed that *"As a matter of fact, on the 16k blocksize there were a fair amount of log file switch completion waits appearing here and there."*

Given the fact that his (attempted) test of the copy in the 16KB block size didn't produce a variation in the run time, and that there is no good reason for a clean copy to perform the way he says it did, and that he supplied no evidence for the test, Occam's razor suggests that he just pointed his code at the wrong table.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.[/nobr]

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 15, 2008 2:57 PM in response to: benprusinski

Reply

>
> For a past customer a large financial company, we
> improved database performance by increasing block
> size from 8k blocksize to 16k blocksize.
> Performance for nightly data loads went down from 22
> hours to 6 hours when we increased the database block
> size.
>

I don't like to disagree with Hans Forbrich that this is a valid data point - but it's obvious it isn't.

Twenty-two hours for a nightly run leave only 2 hours for the daytime processing, which means most of your daytime processing would have been running concurrently with the overnight.

The likely consequences of this would be massive contention, huge overheads due to read-consistency (slowing down the day and nightly work), and the potential for index and table space wastage on a massive scale.

Although NDA does not allow you to give any details, it would be useful to see a few comments on how you justified to the client that the only cost-effective option was a complete rebuild of the entire database. What other options were indicated but discounted during your analysis ?

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 15, 2008 3:24 PM in response to: Jonathan Lewis

Reply

> I don't like to disagree with Hans Forbrich that this
> is a valid data point - but it's obvious it isn't.

Since Steve (and Ben) leave me with the impression they are competent, I think it's a valid data point.

Where it applies, what assumptions are valid or invalid, where it can be applied to any specific or generalized environment other that the poster's ... those questions remain unanswered. Same with the fact that Oracle uses 8K blocks.

So to me it's as valid as some of the other hearsay and experience points made by some of our other esteemed colleagues. I'm not quite sure which chart or where on a chart to put the point.

But it's a data point. <g>

What I am saying is that anecdotes of other people's experience should not simply be rejected, but taken for what they are - anecdotes of other people's experience. It's incorrect for me to tell them their experience is wrong. Just as it's incorrect for them to tell me that their experience will provide identical or similar results in my environment.

Until it is backed up with reproducible methods, it does stay as anecdotal evidence.

However, if their experience or anecdote opens my mind to trying something that I had not thought about when I am stuck, it is both valid and valuable.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 15, 2008 4:35 PM in response to: Hans Forbrich

Reply

> What I am saying is that anecdotes of other people's
> experience should not simply be rejected, but taken
> for what they are - anecdotes of other people's
> experience. It's incorrect for me to tell them their

> experience is wrong.

I don't think anyone is questioning the basic phenomena they observed - "time to completion was shorter" - but it's certainly correct to question their interpretation - "it's the change in block size" - if they supply no supporting argument (cp. Ben) or supply information that suggests their interpretation is wrong (cp. Steve).

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 15, 2008 4:40 PM in response to: Jonathan Lewis

Reply

I think we agree.

I am saying that I can not put it on a specific chart because it's an anecdote. You are asking for rationale to put the anecdote specifically on the Block_Size chart.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 15, 2008 5:42 PM in response to: Hans Forbrich

Reply

You wrote:
"What I am saying is that anecdotes of other people's experience should not simply be rejected, but taken for what they are - anecdotes"

And like Jonathan I don't question that they saw what they saw. Nor do I question their competence.

But our species has developed, over the millennium, a phenomenal ability to rationalize and to draw conclusions where little conclusive evidence exists. In primitive times this ability had tremendous survival value thus we are all here. But applying these same rules to medicine, or a card game, or a database is fraught with dangers.

I see that no one has yet asked me to explain why that double-blind pharmaceutical testing was invalid. No one willing to bite? The answer is surprisingly similar to something we have seen in this thread. <g>

benprusinski

Posts: 207
From: San Diego, CA
Registered: 2/1/00

Re: Larger vs. Small data block
Posted: Jun 15, 2008 6:50 PM in response to: Jonathan Lewis

Reply

Hello Jonathan,

You have some valid points. Yes, there would be possible considerations for overhead with read consistency and possible table and index wastage.

I was called in for this client after the previous Oracle DBA quit on the spot. He was inexperienced and built the database with an 8k default size for a 4TB data warehouse and reporting financial database on Oracle. Fortunately, it was not a production data warehouse but rather a copy of production.

For same odd reason, the regular production data warehouse used 16k block size and had no issues with the ETL nightly jobs which ran between 3-6 hours each night. The other database (Copy of Prod) was using 8k blocks and running very slow. I checked all the performance setups when I was called in to help them at the last minute and ran Statspack reports and checked all database and server OS parameters. Client agreed to let me rebuild the database with 16k block size and we saw the performance improvement.

I provided all the options to the client in addition to block size change including changing the application design. However, due to project deadlines and the need to have things quickly improved for performance to get the copy of PROD database back in sync with current production, I gave them the quickest option at the time which was to increase the block size to 16k from the 8k default value.

Regards,
Ben Prusinski
http://oracle-magician.blogspot.com/

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 15, 2008 7:51 PM in response to: benprusinski

Reply

Ben Prusinski wrote:
He was inexperienced and built the database with an 8k default size for a 4TB data warehouse and reporting financial database on Oracle.

Could you clarify this statement? Are you suggesting that because of his inexperience, he incorrectly chose 8k blocksize for a 4TB data warehouse, possibly suggesting that a 4TB warehouse should have a block size larger than 8k based on size alone? Or are you suggesting that because of his lack of experience, he overlooked the fact the production database used a 16k block and mistakely built the copy with an 8k block making them different?

--
Regards,

Greg Rahn
http://structureddata.org

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 15, 2008 8:01 PM in response to: benprusinski

Reply

You wrote:
"He was inexperienced and built the database with an 8k default size for a 4TB data warehouse and reporting financial database on Oracle."

I don't follow the logic here could you please explain this. I can point you to some 200+TB databases using 8K blocks that are among the most efficient on the planet.

Perhaps I am misunderstanding your intent but I don't see where A follows B. Thanks.

benprusinski

Posts: 207
From: San Diego, CA
Registered: 2/1/00

Re: Larger vs. Small data block
Posted: Jun 15, 2008 8:47 PM in response to: Greg Rahn

Reply

Actually, the previous DBA did overlook the fact that the production database was originally built with a 16k block size. I believe that when he built the database copy of production with 8k block size that it was a mistake.

benprusinski

Posts: 207
From: San Diego, CA
Registered: 2/1/00

Re: Larger vs. Small data block
Posted: Jun 15, 2008 8:49 PM in response to: damorgan

Reply

I don't follow the logic here could you please explain this. I can point you to some 200+TB databases using 8K blocks that are among the most efficient on the planet.

Very well, but in the real case that I worked on, when I changed the block size to 16k from 8k, I saw the performance improvement.

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 15, 2008 9:09 PM in response to: Jonathan Lewis

Reply

Jonathan,

> He also **said** that he also did a test
> based on copying the data into another tablespace
> with 16KB blocks - but we have no evidence that he
> then tested this copy. Possibly he copied the data,
> and then ran the test against the original.

Before you were talking anomalies, now I'm either a liar or inept?

> without commenting on the 446M current gets for a
> process that updates 830,000 rows

Omission is not ignorance.

> Given the fact that his (attempted) test of the copy
> in the 16KB block size didn't produce a variation in
> the run time,

Not a sizable variation, no.

> and that there is no good reason for a
> clean copy to perform the way he says it did, and
> that he supplied no evidence for the test,

Omission is not ignorance.

> Occam's razor suggests that he just pointed
> his code at the wrong table.

Am I translating this right? You're basically saying that we should take the simple assumption that I did it wrong in favor of the possibility something else was amiss? I was actually quite liking some of the possibilities you brought up in your interpretations up to this point. This just seems like giving up.

damorgan,

No, I would not take that assumption on your medication question. In order to keep this thread somewhat civilized I won't elaborate, but there are still many unanswered questions and factors.

And I fully understand that concept. I produced an observation from a test. Never was it said, "Jonathan Lewis is wrong, it's not an ASSM issue, or a delayed block cleanout issue, it's because I changed blocksize." Never was it said that the observation was "proof" of anything at all. Jonathan, you disputed the use of the word "fact" in a recent post; I concede that this was poor wording for that one statement. What was fact was that query times changed; I was not trying to imply that the blocksize change was the *only* factor involved.

It was an observation, not a proof, meant to be picked apart just as some in the thread have been doing. In fact, I even concurred that it would be good to run another test, time and client consent willing, that would be a 'clean slate' test with everything run from scratch and documented. This statement was glossed over in favor of saying my tests were wrong.

damorgan, I've been a DBA for many years, whether as a permanent DBA or a consultant, as well as an instructor for OU. Argue the technical specifications of a test all you like, but do not label my knowledge illusory. Leave your bias and assumptions at the door, sir.

Hans, you nailed it when you said "Until it is backed up with reproducible methods, it does stay as anecdotal evidence. However, if their experience or anecdote opens my mind to trying something that I had not thought about when I am stuck, it is both valid and valuable." Thank you for not making any assumptions.

I don't think I'll be visiting this thread for a while if at all. I've already wasted too much time on it, the page count is far too long, and anything I add will be disputed regardless of my intent. Thank you for the interpretations of the test, no thank you on the assumptions about my character or knowledge. I am still open to interpretations, theories, or anything else, just leave a comment on my blog or email me. Goodnight, good luck, and Godspeed!

Message was edited by:
Steve Karam

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 15, 2008 9:53 PM in response to: damorgan

Reply

> A pharmaceutical company in the 1960s was testing
> oral contraceptives at Stanford University (a school
> whose female students are definitely above average in
> intelligence). One group was given dosage "A" and
> another group dosage "B". In both cases the amount of
> drug was the same but one group received 21 active
> pills and 7 placebos while the other group received
> 28 active pills. One group reported more pregnancies
> than the other.
>
> Was the correct interpretation that one dosage was
> more effective than the second?
>
> My point was not to insult anyone. But rather to

> point out that numbers, even numbers presented in
> chart form, are subject to multiple interpretations
> unless the experimental conditions are carefully
> controlled.
>
> If you think the answer to the above is "yes" ... I
> will provide the rest of the story that proves
> otherwise.

> I see that no one has yet asked me to explain why
> that double-blind pharmaceutical testing was invalid.
> No one willing to bite? The answer is surprisingly
> similar to something we have seen in this thread.

.... but you haven't listed the hormone level in each group. As an Oracle instructor, you have the lack of knowledge in applied science. So far i have noted your zero contribution and high level tendency to promote troll in this thread.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 15, 2008 9:57 PM in response to: damorgan
Reply

damorgan

Also, please, i would like to see the name of at least one company with 200+TB data in their single instance of Oracle

Hemant K Chitale

Posts: 1,259
Registered: 11/6/98

Re: Larger vs. Small data block
Posted: Jun 15, 2008 10:16 PM in response to: sp009
Reply

[not replying to sp009 specifically, just wanted to add my observation, so
this post isn't a response to a specific person but some misconceptions]

There really should not be a "definitive" [but could, probably, be a "tenuous"]
relationship between the database size and the block size used for that database.

There are a number of determinants of block size :
1. Concurrent DML . Very high rates of concurrent DML on adjacent rows/blocks
can encounter waits on latches (besides the obvious ITL) waits with larger block
sizes
2. Block Clones. Too many cloned blocks means that a signficant portion of
the db_cache holds redundant data -- which situation becomes "badder" (if not
"worse") with larger block sizes
3. DWH Query environments might do better with larger block sizes -- but
we seem to have disagreements on this
4. Block sizes can impact Redo Generation if using scripted Hot Backups
5. Larger block sizes might (might !) mean better, more, compact indexes
(inspite of all those experts who disagree)
6. CPU, Bus Transfer Speeds, I/O Hardware and Transfer speeds might
manifest differently with high rates of concurrent single and multiple block reads
if block sizes are different

What I have been trying to say is that
THERE IS NO CORRECT BLOCK SIZE. Once upon a time 2K seemed to be
correct. Currently 8K seems to be correct. But that is not necessarily a
universal truth. Under certain conditions, 8K is not optimal.
And let's just all leave it at that. We all agree to disagree about the 'findings'
or 'interpretation' of test results

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 15, 2008 10:18 PM in response to: sp009
Reply

> Also, please, i would like to see the name of at
> least one company with 200+TB data in their single
> instance of Oracle

Interesting question, so I checked Google. Winter Corp has been publishing the largest VLDB stats for several years.

In 2005, Max Planck Institute for Meteorology has a 222,835 GB Oracle database according to
http://www.wintercorp.com/VLDB/2005_TopTen_Survey/TopTenWinners_2005.asp

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 15, 2008 10:20 PM in response to: Hemant K Chitale
Reply

> What I have been trying to say is that
> THERE IS NO CORRECT BLOCK SIZE. Once upon a time 2K seemed to be
> correct. Currently 8K seems to be correct. But that is not necessarily a
> universal truth. Under certain conditions, 8K is not optimal.

Yes!

>
> And let's just all leave it at that. We all agree to disagree about the 'findings'
> or 'interpretation' of test results

YES!!!!!

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 15, 2008 10:47 PM in response to: sp009
Reply

Interesting but irrelevant to the issue. The reality as clearly demonstrated in today's marketplace is that is does not matter. Mirroring, in a sense, my feeling about much of what is being posted about block size. Can it affect performance ... yes. Is it relevant to much of the anecdotal evidence put forward here? Not necessarily.

The question is still on the table. <g>

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 15, 2008 10:49 PM in response to: sp009

Reply

Think very large internet retailer. Think Seattle. Have a great day.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 15, 2008 10:54 PM in response to: Hans Forbrich

Reply

Which, by definition, means that the observations can not be relied upon as a guide in making decisions with respect to any other system.

So given that there are some database properties that can be easily configured, and reconfigured, on-the-fly. And that others, such as block size, are essentially set and forget. The most flexible solution, unless you've the luxury of rebuilding a database from scratch, is to go with the 8K block and then use all of the other tools of the trade to tune it over the years.

Hans Forbrich

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

Re: Larger vs. Small data block
Posted: Jun 15, 2008 11:10 PM in response to: damorgan

Reply

> Which, by definition, means that the observations can
> not be relied upon as a guide in making decisions
> with respect to any other system.

Yup.

Not a guide, but an alternative to consider when doing benchmarks.

And, as I said in a much earlier post, an alternative that is not very high on the list.

But an alternative, never the less.

Message was edited by: Hans Forbrich

Amusing and worthy of reading as this thread has been in places, reality does call. I've got a Spatial seminar to review modernize ...

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 16, 2008 3:13 AM in response to: Steve Karam

Reply

>
> > He also **said** that he also did a test
> > based on copying the data into another tablespace
> > with 16KB blocks - but we have no evidence that he
> > then tested this copy. Possibly he copied the
> data, and then ran the test against the original.
>
> Before you were talking anomalies, now I'm either a liar or inept?
>

One silly mistake doesn't make you incompetent. It's particularly easy to overlook an error when it gives you the answer you're expecting to see.

Injured innocence is not an intelligent response.

> > without commenting on the 446M current gets for a
> > process that updates 830,000 rows
>
> Omission is not ignorance.
>

That does rather depend on what you include and what you omit. Omitting 446M cu gets is a rather important omission when it accounts for 100% of the time difference that you think is due to a difference in block size. And what you seem to make most of accounts for virtually no time at all.

>
> > Given the fact that his (attempted) test of the copy
> > in the 16KB block size didn't produce a variation in
> > the run time,
>
> Not a sizable variation, no.
>
> > and that there is no good reason for a
> > clean copy to perform the way he says it did, and
> > that he supplied no evidence for the test,
>
> Omission is not ignorance.
>

See above. But in this case, your comment is irrelevant. A better comment might have been "Absence of evidence is not evidence of absence".

>
> > Occam's razor suggests that he just pointed
> > his code at the wrong table.
>
> Am I translating this right? You're basically saying
> that we should take the simple assumption that I did
> it wrong in favor of the possibility something else
> was amiss?

Correct. That's what Occam's razor is about.
In the absence of evidence, the simpler solution is the more sensible choice.

Have you never seen the stories of DBAs who've run a test script against the production database by accident? Silly mistakes happen.

> I was actually quite liking some of the
> possibilities you brought up in your interpretations
> up to this point. This just seems like giving up.
>

Apart from the possible impact of index updates (which we discount because you say there are no indexes), the only possibility I brought up was the impact of delayed block cleanout. That's partly why I can be so confident that the simplest explanation of your 16K test is (in the absence of any evidence to the contrary) that you made a simple mistake.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

Mohan Nair
Posts: 612
Registered: 7/14/00

Re: Larger vs. Small data block
Posted: Jun 16, 2008 4:57 AM in response to: user619401 Reply

See this link
"How to choose the correct block size"
http://www.myoracleguide.com/s/MultipleBlocksizes.htm#cbsz

Mohan

Maran Viswarayar
Posts: 4,196
From: Cecil,Singapore
Registered: 9/23/05

Re: Larger vs. Small data block
Posted: Jun 16, 2008 5:02 AM in response to: Mohan Nair Reply

Hi Nair,

I think you need to justify your points here as this thread is more on justifying your claims rather than just providing silver bullets

---

Charles Hooper
Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 16, 2008 6:54 AM in response to: Charles Hooper Reply

```
> > I really don't understand why all examples are
> using
> > index full scan ?
> > What about index range scan ? I made some test and
> in
> > my test
> > if you have different block in data and index
> > tablespace response time
> > is a little bit worse or equal but never was
> better.
> >
> > regards,
> > Marcin Przepiorowski
>
> What I attempted to do is to create as many possible
> access paths as possible with a limited and
> reproducible data set, while keeping as little of the
> previously read index and table blocks in memory to
> force physical reads (as if the data set were too
> large to fit into and remain in the buffer cache).
```

I finished putting together a more comprehensive test script that addresses many of the issues that I had with my original test script. I performed a test of the script last night to look for typos in the script, but only had a couple minutes to review the output. Foreign keys and indexes will have a significant impact on performance, but it is too early to tell if block size makes much of a difference when the foreign keys are checked during an insert or update. For comparison, data is first generated into a temp table and then copied into a table with the foreign key constraints and indexes to help isolate the cause of the execution time. The test closely resembles a component of a purchase ordering system, with data inserted in mostly non-sequential order. Also included i a test on a narrow (2 column) table with 900,000 rows.

New test script (warning: certain portions of the script generate 2+ GB of redo, run time for each block size is expected to be 5+ hours).

```
ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;

spool c:\testnew16.txt
set pagesize 100000
set autotrace on
set timing on

SELECT
  COUNT(*)
FROM
  ALL_OBJECTS;

SELECT 'CREATING LOCATIONS' FROM DUAL;

ALTER SESSION SET TRACEFILE_IDENTIFIER = 'CREATE_TABLES';
ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 8';

CREATE TABLE LOCATIONS(
  LOCATION_ID VARCHAR2(15) NOT NULL ENABLE,
  WAREHOUSE_ID VARCHAR2(15) NOT NULL ENABLE,
  DESCRIPTION VARCHAR2(80),
  LOCATION_TYPE CHAR(1) NOT NULL ENABLE,
  CONSTRAINT "CHK_LOCATIONS" CHECK (
    (LOCATION_TYPE = 'T' Or LOCATION_TYPE = 'R' Or LOCATION_TYPE = 'F')) ENABLE,
  PRIMARY KEY (WAREHOUSE_ID, LOCATION_ID));

CREATE INDEX IND_LOCATIONS_1 ON LOCATIONS (LOCATION_ID);

SELECT 'CREATING UMS' FROM DUAL;

CREATE TABLE UMS (
  UNIT_OF_MEASURE VARCHAR2(15) NOT NULL ENABLE,
  DESCRIPTION VARCHAR2(40),
  UOM_SCALE NUMBER NOT NULL ENABLE,
  CONSTRAINT "CHK_UOM_SCALE" CHECK (
    (UOM_SCALE >= 0 And UOM_SCALE <= 4)) ENABLE,
  PRIMARY KEY ("UNIT_OF_MEASURE"));

SELECT 'CREATING VENDORS' FROM DUAL;

CREATE TABLE VENDORS (
  VENDOR_ID VARCHAR2(15) NOT NULL ENABLE,
  VENDOR_NAME VARCHAR2(50),
  ADDR_1 VARCHAR2(50),
  ADDR_2 VARCHAR2(50),
  ADDR_3 VARCHAR2(50),
  CITY VARCHAR2(30),
  STATE VARCHAR2(10),
  ZIPCODE VARCHAR2(10),
  COUNTRY VARCHAR2(50),
  CONTACT_FIRST_NAME VARCHAR2(30),
  CONTACT_LAST_NAME VARCHAR2(30),
  CONTACT_INITIAL VARCHAR2(2),
  CONTACT_POSITION VARCHAR2(20),
  CONTACT_HONORIFIC VARCHAR2(4),
  CONTACT_SALUTATION VARCHAR2(60),
  CONTACT_PHONE VARCHAR2(20),
  CONTACT_FAX VARCHAR2(20),
  REMIT_TO_NAME VARCHAR2(50),
  REMIT_TO_ADDR_1 VARCHAR2(50),
  REMIT_TO_ADDR_2 VARCHAR2(50),
  REMIT_TO_ADDR_3 VARCHAR2(50),
  REMIT_TO_CITY VARCHAR2(30),
  REMIT_TO_STATE VARCHAR2(10),
  REMIT_TO_ZIPCODE VARCHAR2(10),
  REMIT_TO_COUNTRY VARCHAR2(50),
  FREE_ON_BOARD VARCHAR2(25),
  SHIP_VIA VARCHAR2(40),
  BUYER VARCHAR2(15),
  REPORT_1099_MISC CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  TERMS_NET_TYPE CHAR(1) DEFAULT ' ' NOT NULL ENABLE,
  TERMS_NET_DAYS NUMBER,
  TERMS_NET_DATE DATE,
  TERMS_DISC_TYPE CHAR(1) DEFAULT ' ' NOT NULL ENABLE,
  TERMS_DISC_DAYS NUMBER,
  TERMS_DISC_DATE DATE,
  TERMS_DISC_PERCENT NUMBER(5,3),
  TERMS_DESCRIPTION VARCHAR2(50),
  USER_1 VARCHAR2(80),
  USER_2 VARCHAR2(80),
  USER_3 VARCHAR2(80),
  USER_4 VARCHAR2(80),
  USER_5 VARCHAR2(80),
  USER_6 VARCHAR2(80),
  USER_7 VARCHAR2(80),
  USER_8 VARCHAR2(80),
  USER_9 VARCHAR2(80),
  USER_10 VARCHAR2(80),
  CONSTRAINT "CHK_VENDORS" CHECK (
    (REPORT_1099_MISC = 'Y' Or REPORT_1099_MISC = 'N')
    AND (TERMS_NET_TYPE = 'A'
      Or TERMS_NET_TYPE = 'M'
      Or TERMS_NET_TYPE = 'D'
      Or TERMS_NET_TYPE = 'N'
      Or TERMS_NET_TYPE = 'E')
    AND (TERMS_DISC_TYPE = 'A'
      Or TERMS_DISC_TYPE = 'M'
      Or TERMS_DISC_TYPE = 'D'
      Or TERMS_DISC_TYPE = 'N'
      Or TERMS_DISC_TYPE = 'E')) ENABLE,
  PRIMARY KEY (VENDOR_ID));

CREATE TABLE VENDORS_TEMP AS
SELECT
  *
FROM
  VENDORS;

SELECT 'CREATING PARTS' FROM DUAL;

CREATE TABLE PARTS (
  PART_ID VARCHAR2(30) NOT NULL ENABLE,
  DESCRIPTION VARCHAR2(40),
  STOCK_UM VARCHAR2(15) NOT NULL ENABLE,
  PLANNING_LEADTIME NUMBER DEFAULT 0 NOT NULL ENABLE,
  ORDER_POLICY CHAR(1) DEFAULT 'M' NOT NULL ENABLE,
  ORDER_POINT NUMBER(14,4),
  SAFETY_STOCK_QTY NUMBER(14,4),
  FIXED_ORDER_QTY NUMBER(14,4),
  DAYS_OF_SUPPLY NUMBER,
  MINIMUM_ORDER_QTY NUMBER(14,4),
  MAXIMUM_ORDER_QTY NUMBER(14,4),
  ENGINEERING_MSTR VARCHAR2(3),
  PRODUCT_CODE VARCHAR2(15),
  COMMODITY_CODE VARCHAR2(15),
  MFG_NAME VARCHAR2(30),
  MFG_PART_ID VARCHAR2(30),
  FABRICATED CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  PURCHASED CHAR(1) DEFAULT 'Y' NOT NULL ENABLE,
  STOCKED CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  DETAIL_ONLY CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  DEMAND_HISTORY CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  TOOL_OR_FIXTURE CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  INSPECTION_REQD CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  WEIGHT NUMBER(14,4),
  WEIGHT_UM VARCHAR2(15),
  DRAWING_ID VARCHAR2(15),
  DRAWING_REV_NO VARCHAR2(8),
  PREF_VENDOR_ID VARCHAR2(15),
  PRIMARY_WHS_ID VARCHAR2(15),
  PRIMARY_LOC_ID VARCHAR2(15),
  BACKFLUSH_WHS_ID VARCHAR2(15),
  BACKFLUSH_LOC_ID VARCHAR2(15),
  INSPECT_WHS_ID VARCHAR2(15),
  INSPECT_LOC_ID VARCHAR2(15),
  MRP_REQUIRED CHAR(1) DEFAULT 'N',
  MRP_EXCEPTIONS CHAR(1) DEFAULT 'N',
  PRIVATE_UM_CONV CHAR(1) DEFAULT 'N',
  AUTO_BACKFLUSH CHAR(1) DEFAULT 'Y',
  PLANNER_USER_ID VARCHAR2(20),
  BUYER_USER_ID VARCHAR2(20),
  ABC_CODE CHAR(1),
  ANNUAL_USAGE_QTY NUMBER(15,4),
  INVENTORY_LOCKED CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  UNIT_MATERIAL_COST NUMBER(20,6) DEFAULT 0 NOT NULL ENABLE,
  UNIT_LABOR_COST NUMBER(20,6) DEFAULT 0 NOT NULL ENABLE,
  UNIT_BURDEN_COST NUMBER(20,6) DEFAULT 0 NOT NULL ENABLE,
  UNIT_SERVICE_COST NUMBER(20,6) DEFAULT 0 NOT NULL ENABLE,
  BURDEN_PERCENT NUMBER(5,2) DEFAULT 0 NOT NULL ENABLE,
  BURDEN_PER_UNIT NUMBER(20,6) DEFAULT 0 NOT NULL ENABLE,
  PURC_BUR_PERCENT NUMBER(6,3) DEFAULT 0 NOT NULL ENABLE,
  PURC_BUR_PER_UNIT NUMBER(20,6) DEFAULT 0 NOT NULL ENABLE,
  FIXED_COST NUMBER(15,2) DEFAULT 0 NOT NULL ENABLE,
  UNIT_PRICE NUMBER(20,6),
  NEW_MATERIAL_COST NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  NEW_LABOR_COST NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  NEW_BURDEN_COST NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  NEW_SERVICE_COST NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  NEW_BURDEN_PERCENT NUMBER(5,2) DEFAULT 0 NOT NULL ENABLE,
  NEW_BURDEN_PERUNIT NUMBER(20,6) DEFAULT 0 NOT NULL ENABLE,
  NEW_FIXED_COST NUMBER(15,2) DEFAULT 0 NOT NULL ENABLE,
  MAT_GL_ACCT_ID VARCHAR2(30),
  LAB_GL_ACCT_ID VARCHAR2(30),
  BUR_GL_ACCT_ID VARCHAR2(30),
  SER_GL_ACCT_ID VARCHAR2(30),
  QTY_ON_HAND NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  QTY_AVAILABLE_ISS NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  QTY_AVAILABLE_MRP NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  QTY_ON_ORDER NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  QTY_IN_DEMAND NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  USER_1 VARCHAR2(80),
  USER_2 VARCHAR2(80),
  USER_3 VARCHAR2(80),
  USER_4 VARCHAR2(80),
  USER_5 VARCHAR2(80),
  USER_6 VARCHAR2(80),
  USER_7 VARCHAR2(80),
  USER_8 VARCHAR2(80),
  USER_9 VARCHAR2(80),
  USER_10 VARCHAR2(80),
  LT_PLUS_DAYS NUMBER,
  LT_MINUS_DAYS NUMBER,
  STATUS CHAR(1),
  USE_SUPPLY_BEF_LT CHAR(1),
  QTY_COMMITTED NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  PRT_CREATE_USER_ID VARCHAR2(30) DEFAULT USER,
  PRT_CREATE_DATE DATE DEFAULT SYSDATE,
  CONSTRAINT "CHK_PART1" CHECK (
    (PLANNING_LEADTIME >= 0)
    AND (ORDER_POLICY = 'N'
      Or ORDER_POLICY = 'M'
      Or ORDER_POLICY = 'F'
      Or ORDER_POLICY = 'E'
      Or ORDER_POLICY = 'D'
      Or ORDER_POLICY = 'P')
    AND (ORDER_POINT >= 0)
    AND (SAFETY_STOCK_QTY >= 0)
    AND (FIXED_ORDER_QTY >= 0)
    AND (DAYS_OF_SUPPLY >= 0)
    AND (MINIMUM_ORDER_QTY >= 0)
    AND (MAXIMUM_ORDER_QTY >= 0)
    AND (FABRICATED = 'Y' Or FABRICATED = 'N')
    AND (PURCHASED = 'Y' Or PURCHASED = 'N')
    AND (STOCKED = 'Y' Or STOCKED = 'N')
    AND (DETAIL_ONLY = 'Y' Or DETAIL_ONLY = 'N')
    AND (DEMAND_HISTORY = 'Y' Or DEMAND_HISTORY = 'N')
    AND (TOOL_OR_FIXTURE = 'Y' Or TOOL_OR_FIXTURE = 'N')
    AND (MRP_REQUIRED = 'Y' Or MRP_REQUIRED = 'N')
    AND (MRP_EXCEPTIONS = 'Y' Or MRP_EXCEPTIONS = 'N')
    AND (PRIVATE_UM_CONV = 'Y' Or PRIVATE_UM_CONV = 'N')
    AND (INVENTORY_LOCKED = 'Y' Or INVENTORY_LOCKED = 'N')
    AND (INSPECTION_REQD = 'Y' Or INSPECTION_REQD = 'N')) ENABLE,
  PRIMARY KEY (PART_ID),
  CONSTRAINT "FKEY_INSP" FOREIGN KEY (INSPECT_WHS_ID, INSPECT_LOC_ID)
    REFERENCES LOCATIONS (WAREHOUSE_ID, LOCATION_ID) ENABLE,
  CONSTRAINT "FKEY_PREF_VENDOR" FOREIGN KEY (PREF_VENDOR_ID)
    REFERENCES VENDORS (VENDOR_ID) ENABLE,
  CONSTRAINT "FKEY_UM" FOREIGN KEY (WEIGHT_UM)
    REFERENCES UMS (UNIT_OF_MEASURE) ENABLE,
  CONSTRAINT "FKEY_STOCK_UM" FOREIGN KEY (STOCK_UM)
    REFERENCES UMS (UNIT_OF_MEASURE) ENABLE);

CREATE INDEX IND_PARTS_1 ON PARTS (MRP_EXCEPTIONS);
CREATE INDEX IND_PARTS_2 ON PARTS (MFG_NAME, MFG_PART_ID);
CREATE INDEX IND_PARTS_3 ON PARTS (WEIGHT_UM);
CREATE INDEX IND_PARTS_4 ON PARTS (MRP_REQUIRED);
CREATE INDEX IND_PARTS_5 ON PARTS (PREF_VENDOR_ID);
CREATE INDEX IND_PARTS_6 ON PARTS (STOCK_UM);
CREATE INDEX IND_PARTS_7 ON PARTS (ORDER_POINT);

CREATE TABLE PARTS_TEMP AS
SELECT
  *
FROM
  PARTS;

SELECT 'CREATING PO_HEADER' FROM DUAL;

CREATE TABLE PO_HEADER (
  PURC_ORDER_ID VARCHAR2(15) NOT NULL ENABLE,
  VENDOR_ID VARCHAR2(15) NOT NULL ENABLE,
  CONTACT_FIRST_NAME VARCHAR2(30),
  CONTACT_LAST_NAME VARCHAR2(30),
  CONTACT_INITIAL VARCHAR2(2),
  CONTACT_POSITION VARCHAR2(20),
  CONTACT_HONORIFIC VARCHAR2(4),
  CONTACT_SALUTATION VARCHAR2(60),
  CONTACT_PHONE VARCHAR2(20),
  CONTACT_FAX VARCHAR2(20),
  PURC_ORD_ADDR_NO NUMBER,
  SHIPTO_ADDR_NO NUMBER,
  ORDER_DATE DATE DEFAULT SYSDATE NOT NULL ENABLE,
  DESIRED_RECV_DATE DATE DEFAULT TRUNC(SYSDATE),
  BUYER VARCHAR2(15),
  FREE_ON_BOARD VARCHAR2(25),
  SHIP_VIA VARCHAR2(40),
  SALES_TAX_GROUP_ID VARCHAR2(15),
  PO_STATUS CHAR(1) DEFAULT ' ' NOT NULL ENABLE,
  BACK_ORDER CHAR(1) DEFAULT ' ' NOT NULL ENABLE,
  SELL_RATE NUMBER(15,8) NOT NULL ENABLE,
  BUY_RATE NUMBER(15,8) NOT NULL ENABLE,
  ENTITY_ID VARCHAR2(5) NOT NULL ENABLE,
  POSTING_CANDIDATE CHAR(1) DEFAULT 'Y' NOT NULL ENABLE,
  LAST_RECEIVED_DATE DATE,
  TOTAL_AMT_ORDERED NUMBER(15,2) DEFAULT 0 NOT NULL ENABLE,
  TOTAL_AMT_RECVD NUMBER(15,2) DEFAULT 0 NOT NULL ENABLE,
  MARKED_FOR_PURGE CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  EXCH_RATE_FIXED CHAR(1) DEFAULT 'N' NOT NULL ENABLE,
  PROMISE_DATE DATE,
  PRINTED_DATE DATE,
  TERMS_DISC_TYPE CHAR(1),
  EDI_BLANKET_FLAG CHAR(1),
  EDI_BLANKET_PO_NO VARCHAR2(30),
  CONTRACT_ID VARCHAR2(30),
  SHIPTO_ID VARCHAR2(20),
  TERMS_NET_TYPE CHAR(1) DEFAULT ' ' NOT NULL ENABLE,
  TERMS_NET_DAYS NUMBER,
  TERMS_NET_DATE DATE,
  TERMS_DISC_DAYS NUMBER,
  TERMS_DISC_DATE DATE,
  TERMS_DISC_PERCENT NUMBER(5,3),
  TERMS_DESCRIPTION VARCHAR2(50),
  CURRENCY_ID VARCHAR2(15),
  WAREHOUSE_ID VARCHAR2(15),
  CREATE_DATE DATE DEFAULT SYSDATE NOT NULL ENABLE,
  CONTACT_MOBILE VARCHAR2(20),
  CONTACT_EMAIL VARCHAR2(50),
  USER_1 VARCHAR2(80),
  USER_2 VARCHAR2(80),
  USER_3 VARCHAR2(80),
  USER_4 VARCHAR2(80),
  USER_5 VARCHAR2(80),
  USER_6 VARCHAR2(80),
  USER_7 VARCHAR2(80),
  USER_8 VARCHAR2(80),
  USER_9 VARCHAR2(80),
  USER_10 VARCHAR2(80),
  UDF_LAYOUT_ID VARCHAR2(15),
  PO_CREATE_USER_ID VARCHAR2(30) DEFAULT USER,
  CONSTRAINT "CHK_PO" CHECK (
    (PO_STATUS = 'F' Or PO_STATUS = 'R' Or PO_STATUS = 'C' Or PO_STATUS = 'X')
    AND (BACK_ORDER = 'Y' Or BACK_ORDER = 'N')
    AND (POSTING_CANDIDATE = 'Y' Or POSTING_CANDIDATE = 'N')
    AND (MARKED_FOR_PURGE = 'Y' Or MARKED_FOR_PURGE = 'N')
    AND (TERMS_DISC_TYPE = 'A' Or TERMS_DISC_TYPE = 'M' Or TERMS_DISC_TYPE = 'D' Or TERMS_DISC_TYPE = 'N' Or TERMS_DISC_TYPE = 'E')
    AND (TERMS_NET_TYPE = 'A' Or TERMS_NET_TYPE = 'M' Or TERMS_NET_TYPE = 'D' Or TERMS_NET_TYPE = 'N' Or TERMS_NET_TYPE = 'E'))
ENABLE,
  PRIMARY KEY (PURC_ORDER_ID));

CREATE INDEX IND_PO_HEADER_1 ON PO_HEADER (VENDOR_ID, PURC_ORD_ADDR_NO);
CREATE INDEX IND_PO_HEADER_2 ON PO_HEADER (VENDOR_ID);
CREATE INDEX IND_PO_HEADER_3 ON PO_HEADER (SHIPTO_ADDR_NO);
CREATE INDEX IND_PO_HEADER_4 ON PO_HEADER (POSTING_CANDIDATE);

CREATE TABLE PO_HEADER_TEMP AS
SELECT
  *
FROM
  PO_HEADER;

SELECT 'CREATING PO_LINE' FROM DUAL;

CREATE TABLE PO_LINE (
  PURC_ORDER_ID VARCHAR2(15) NOT NULL ENABLE,
  LINE_NO NUMBER NOT NULL ENABLE,
  PART_ID VARCHAR2(30),
  VENDOR_PART_ID VARCHAR2(30),
  SERVICE_ID VARCHAR2(15),
  USER_ORDER_QTY NUMBER(14,4) NOT NULL ENABLE,
  ORDER_QTY NUMBER(14,4) NOT NULL ENABLE,
  PURCHASE_UM VARCHAR2(15),
  UNIT_PRICE NUMBER(20,6) NOT NULL ENABLE,
  TRADE_DISC_PERCENT NUMBER(6,3) DEFAULT 0 NOT NULL ENABLE,
  FIXED_CHARGE NUMBER(15,2),
  EST_FREIGHT NUMBER(15,2) DEFAULT 0 NOT NULL ENABLE,
  GL_EXPENSE_ACCT_ID VARCHAR2(30),
  SALES_TAX_GROUP_ID VARCHAR2(15),
  PRODUCT_CODE VARCHAR2(15),
  COMMODITY_CODE VARCHAR2(15),
  DESIRED_RECV_DATE DATE,
  LINE_STATUS CHAR(1) DEFAULT ' ' NOT NULL ENABLE,
  LAST_RECEIVED_DATE DATE,
  TOTAL_ACT_FREIGHT NUMBER(15,2) DEFAULT 0 NOT NULL ENABLE,
  TOTAL_USR_RECD_QTY NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  TOTAL_RECEIVED_QTY NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  TOTAL_AMT_RECVD NUMBER(15,2) DEFAULT 0 NOT NULL ENABLE,
  TOTAL_AMT_ORDERED NUMBER(15,2) DEFAULT 0 NOT NULL ENABLE,
  MFG_NAME VARCHAR2(30),
  MFG_PART_ID VARCHAR2(30),
  PROMISE_DATE DATE,
  PIECE_COUNT NUMBER(14,4),
  LENGTH NUMBER(14,4),
  WIDTH NUMBER(14,4),
  HEIGHT NUMBER(14,4),
  DIMENSIONS_UM VARCHAR2(15),
  VAT_CODE VARCHAR2(15),
  TOTAL_DISPATCH_QTY NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  TOTAL_USR_DISP_QTY NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  MINIMUM_CHARGE NUMBER(15,2),
  LAST_DISPATCH_DATE DATE,
  EDI_BLANKET_QTY NUMBER(14,4),
  EDI_BLANKET_USRQTY NUMBER(14,4),
  EDI_ACCUM_QTY_REL NUMBER(14,4),
  EDI_ACCUM_USR_REL NUMBER(14,4),
  EDI_ACCUM_QTY_REC NUMBER(14,4),
  EDI_ACCUM_USR_REC NUMBER(14,4),
  EDI_LAST_REC_DATE DATE,
  EDI_RELEASE_NO VARCHAR2(3),
  EDI_RELEASE_DATE DATE,
  EDI_QTY_RELEASED NUMBER(14,4),
  EDI_USR_QTY_REL NUMBER(14,4),
  EDI_REQ_REL_DATE DATE,
  SHIPTO_ID VARCHAR2(20),
  WAREHOUSE_ID VARCHAR2(15),
  WIP_VAS_REQUIRED CHAR(1),
  ALLOCATED_QTY NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  FULFILLED_QTY NUMBER(14,4) DEFAULT 0 NOT NULL ENABLE,
  HTS_CODE VARCHAR2(20),
  ORIG_COUNTRY_ID VARCHAR2(15),
  USER_1 VARCHAR2(80),
  USER_2 VARCHAR2(80),
  USER_3 VARCHAR2(80),
  USER_4 VARCHAR2(80),
  USER_5 VARCHAR2(80),
  USER_6 VARCHAR2(80),
  USER_7 VARCHAR2(80),
  USER_8 VARCHAR2(80),
  USER_9 VARCHAR2(80),
  USER_10 VARCHAR2(80),
  UDF_LAYOUT_ID VARCHAR2(15),
  POL_CREATE_USER_ID VARCHAR2(30) DEFAULT USER,
  POL_CREATE_DATE DATE DEFAULT SYSDATE,
  CONSTRAINT "CHK_PO_LINE" CHECK ((LINE_STATUS = 'A' Or LINE_STATUS = 'C')) ENABLE,
  PRIMARY KEY (PURC_ORDER_ID, LINE_NO),
  CONSTRAINT "FKEY_PO_HEADER" FOREIGN KEY (PURC_ORDER_ID)
    REFERENCES PO_HEADER (PURC_ORDER_ID) ON DELETE CASCADE ENABLE,
  CONSTRAINT "FKEY_PART_ID" FOREIGN KEY (PART_ID)
    REFERENCES PARTS (PART_ID) ENABLE,
  CONSTRAINT "FKEY_PURC_UM" FOREIGN KEY (PURCHASE_UM)
   REFERENCES UMS (UNIT_OF_MEASURE) ENABLE);

CREATE INDEX IND_PO_LINE_1 ON PO_LINE (WAREHOUSE_ID);
CREATE INDEX IND_PO_LINE_2 ON PO_LINE (SERVICE_ID);
CREATE INDEX IND_PO_LINE_3 ON PO_LINE (PART_ID);
CREATE INDEX IND_PO_LINE_4 ON PO_LINE (VENDOR_PART_ID);

CREATE TABLE PO_LINE_TEMP AS
SELECT
  *
FROM
  PO_LINE;

CREATE TABLE NARROW (
  C1 NUMBER,
  C2 NUMBER);

SELECT 'INSERTING INTO LOCATIONS' FROM DUAL;

ALTER SESSION SET TRACEFILE_IDENTIFIER = 'INSERT_LOCATIONS_UMS';

INSERT INTO
  LOCATIONS
SELECT /*+ ORDERED */
  LOC.LOCATION_ID,
  WH.WAREHOUSE_ID,
  RPAD(WH.WAREHOUSE_ID||'-'||LOC.LOCATION_ID,60),
  DECODE(MOD(ROWNUM,5),0,'T',1,'R','F')
FROM
  (SELECT
    TRIM(TO_CHAR(ABS(ROUND(COS(ROWNUM*3.1415/180*1.2)*1000000,0))))||'LOC' LOCATION_ID,
    ROWNUM RN
  FROM
    DUAL
  CONNECT BY
    LEVEL<=200) LOC,
  (SELECT
    TRIM(TO_CHAR(ABS(ROUND(SIN(ROWNUM*3.1415/180*10.1)*1000000,0))))||'WH' WAREHOUSE_ID,
    ROWNUM RN
  FROM
    DUAL
  CONNECT BY
    LEVEL<=20) WH
WHERE
  (MOD(WH.RN,10)*20+1) <= LOC.RN;

COMMIT;

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'LOCATIONS',CASCADE=>TRUE);

INSERT INTO
  UMS
SELECT
  DECODE(ROWNUM,1,'EA',2,'PC',3,'FT',4,'METER',5,'KG',6,'CASE',7,'LBS',8,'DOZEN'),
  NULL,
  4
FROM
  DUAL
CONNECT BY
  LEVEL<=8;

COMMIT;

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'UMS',CASCADE=>TRUE);
```

```
SELECT 'INSERTING INTO VENDORS' FROM DUAL;
ALTER SESSION SET TRACEFILE_IDENTIFIER = 'INSERT_VENDORS';

INSERT INTO
  VENDORS_TEMP
SELECT
  TRIM(TO_CHAR(ABS(ROUND(COS(ROWNUM*3.14159265/180*51.491976)*10000000,0))))||'VEN' VENDOR_ID,
  TRIM(TO_CHAR(ABS(ROUND(COS(ROWNUM*3.1415/180*.49)*1000000,0))))||'VENDOR NAME' VENDOR_NAME,
  RPAD('ADDR_1',40) ADDR_1,
  RPAD('ADDR_2',35) ADDR_2,
  NULL ADDR_3,
  RPAD('CITY',20) CITY,
  'CA' STATE,
  LPAD(TO_CHAR(ROWNUM),6) ZIPCODE,
  'NONE' COUNTRY,
  NULL CONTACT_FIRST_NAME,
  NULL CONTACT_LAST_NAME,
  NULL CONTACT_INITIAL,
  NULL CONTACT_POSITION,
  NULL CONTACT_HONORIFIC,
  NULL CONTACT_SALUTATION,
  NULL CONTACT_PHONE,
  NULL CONTACT_FAX,
  TRIM(TO_CHAR(ABS(ROUND(COS(ROWNUM*3.1415/180*4.491976)*1000000,0))))||'VENDOR NAME' REMIT_TO_NAME,
  RPAD('ADDR_1',40) REMIT_TO_ADDR_1,
  RPAD('ADDR_2',35) REMIT_TO_ADDR_2,
  NULL REMIT_TO_ADDR_3,
  RPAD('CITY',20) REMIT_TO_CITY,
  'CA' REMIT_TO_STATE,
  LPAD(TO_CHAR(ROWNUM),6) REMIT_TO_ZIPCODE,
  'NONE' REMIT_TO_COUNTRY,
  'NONE' FREE_ON_BOARD,
  'SPECIAL DEL' SHIP_VIA,
  'UNKNOWN' BUYER,
  'N' REPORT_1099_MISC,
  DECODE(MOD(ROWNUM,6),0,'A',1,'M',2,'D',3,'N','E') TERMS_NET_TYPE,
  ROWNUM TERMS_NET_DAYS,
  NULL TERMS_NET_DATE,
  DECODE(MOD(ROWNUM,6),0,'A',1,'M',2,'D',3,'N','E') TERMS_DISC_TYPE,
  MOD(ROWNUM,100)+10 TERMS_DISC_DAYS,
  NULL TERMS_DISC_DATE,
  3.5 TERMS_DISC_PERCENT,
  'STANDARD' TERMS_DESCRIPTION,
  'X' USER_1,
  TO_CHAR(TRUNC(SYSDATE,'YYYY'),'MON DD, YYYY') USER_2,
  NULL USER_3,
  NULL USER_4,
  NULL USER_5,
  NULL USER_6,
  NULL USER_7,
  NULL USER_8,
  NULL USER_9,
  NULL USER_10
FROM
  DUAL
CONNECT BY
  LEVEL<=50000;

SELECT 'ELIMINATING DUP V' FROM DUAL;

DELETE FROM
  VENDORS_TEMP
WHERE
  (VENDOR_ID,TERMS_NET_DAYS) IN
    (SELECT
      V.VENDOR_ID,
      V.TERMS_NET_DAYS
    FROM
      VENDORS_TEMP V,
      (SELECT
        VENDOR_ID,
        MIN(TERMS_NET_DAYS) TERMS_NET_DAYS
      FROM
        VENDORS_TEMP
      GROUP BY
        VENDOR_ID
      HAVING
        COUNT(*)>1) M
    WHERE
      V.VENDOR_ID=M.VENDOR_ID
      AND V.TERMS_NET_DAYS>M.TERMS_NET_DAYS);

INSERT INTO
  VENDORS
SELECT
  *
FROM
  VENDORS_TEMP;

COMMIT;

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'VENDORS',CASCADE=>TRUE);

SELECT 'INSERTING INTO PARTS' FROM DUAL;
ALTER SESSION SET TRACEFILE_IDENTIFIER = 'INSERT_PARTS';

INSERT INTO
  PARTS_TEMP
SELECT
  TRIM(TO_CHAR(ABS(ROUND(SIN(ROWNUM*3.14159265/180*10.191976)*10000000,0))))||'PART' PART_ID,
  TRIM(TO_CHAR(ABS(ROUND(SIN(ROWNUM*3.14159265/180*10.191976)*10000000,0))))||'DESCRIPTION' DESCRIPTION,
  DECODE(MOD(ROWNUM,20),2,'PC',3,'FT',4,'METER',5,'KG',6,'CASE',7,'LBS',8,'DOZEN','EA') STOCK_UM,
  1 PLANNING_LEADTIME,
  'M' ORDER_POLICY,
  ROWNUM ORDER_POINT,
  1 SAFETY_STOCK_QTY,
  1 FIXED_ORDER_QTY,
  1 DAYS_OF_SUPPLY,
  1 MINIMUM_ORDER_QTY,
  9999 MAXIMUM_ORDER_QTY,
  '0' ENGINEERING_MSTR,
  DECODE(MOD(ROWNUM,20),1,'SHOP',2,'OFFICE',3,'JANITOR',4,'INVENTORY',5,'INVENTORY','FG') PRODUCT_CODE,
  DECODE(MOD(ROWNUM,7),1,'SHOP',2,'OFFICE',3,'JANITOR',4,'INVENTORY',5,'INVENTORY','FG') COMMODITY_CODE,
  'UNKNOWN' MFG_NAME,
  'UNKNOWN' MFG_PART_ID,
  DECODE(MOD(ROWNUM,3),1,'Y','N') FABRICATED,
  DECODE(MOD(ROWNUM,3),1,'N','Y') PURCHASED,
  'N' STOCKED,
  'N' DETAIL_ONLY,
  'N' DEMAND_HISTORY,
  'N' TOOL_OR_FIXTURE,
  'N' INSPECTION_REQD,
  0 WEIGHT,
  DECODE(MOD(ROWNUM,20),2,'PC',3,'FT',4,'METER',5,'KG',6,'CASE',7,'LBS',8,'DOZEN','EA') WEIGHT_UM,
  NULL DRAWING_ID,
  NULL DRAWING_REV_NO,
  NULL PREF_VENDOR_ID,
  NULL PRIMARY_WHS_ID,
  NULL PRIMARY_LOC_ID,
  NULL BACKFLUSH_WHS_ID,
  NULL BACKFLUSH_LOC_ID,
  NULL INSPECT_WHS_ID,
  NULL INSPECT_LOC_ID,
  'Y' MRP_REQUIRED,
  'N' MRP_EXCEPTIONS,
  'N' PRIVATE_UM_CONV,
  'Y' AUTO_BACKFLUSH,
  NULL PLANNER_USER_ID,
  NULL BUYER_USER_ID,
  DECODE(MOD(ROWNUM,7),1,'A',2,'B',3,'B','C') ABC_CODE,
  ROWNUM-100000 ANNUAL_USAGE_QTY,
  'N' INVENTORY_LOCKED,
  0 UNIT_MATERIAL_COST,
  0 UNIT_LABOR_COST,
  0 UNIT_BURDEN_COST,
  0 UNIT_SERVICE_COST,
  0 BURDEN_PERCENT,
  0 BURDEN_PER_UNIT,
  0 PURC_BUR_PERCENT,
  0 PURC_BUR_PER_UNIT,
  0 FIXED_COST,
  0 UNIT_PRICE,
  0 NEW_MATERIAL_COST,
  0 NEW_LABOR_COST,
  0 NEW_BURDEN_COST,
  0 NEW_SERVICE_COST,
  0 NEW_BURDEN_PERCENT,
  0 NEW_BURDEN_PERUNIT,
  0 NEW_FIXED_COST,
  '1111111' MAT_GL_ACCT_ID,
  '2222222' LAB_GL_ACCT_ID,
  '3333333' BUR_GL_ACCT_ID,
  '4444444' SER_GL_ACCT_ID,
  ABS(ROUND(SIN(ROWNUM*3.14159265/180*2)*100000,3)) QTY_ON_HAND,
  ABS(ROUND(SIN(ROWNUM*3.14159265/180*2)*100000,3)) QTY_AVAILABLE_ISS,
  ABS(ROUND(SIN(ROWNUM*3.14159265/180*2)*100000,3)) QTY_AVAILABLE_MRP,
  0 QTY_ON_ORDER,
  0 QTY_IN_DEMAND,
  RPAD('USER_1',30) USER_1,
  RPAD('USER_2',30) USER_2,
  RPAD('USER_3',30) USER_3,
  NULL USER_4,
  NULL USER_5,
  NULL USER_6,
  NULL USER_7,
  NULL USER_8,
  NULL USER_9,
  NULL USER_10,
  0 LT_PLUS_DAYS,
  0 LT_MINUS_DAYS,
  'A' STATUS,
  'Y' USE_SUPPLY_BEF_LT,
  0 QTY_COMMITTED,
  'TESTING' PRT_CREATE_USER_ID,
  SYSDATE PRT_CREATE_DATE
FROM
  DUAL
CONNECT BY
  LEVEL<=100000;

SELECT 'REMOVING DUPLICATE PARTS' FROM DUAL;

DELETE FROM
  PARTS_TEMP
WHERE
  (PART_ID,ORDER_POINT) IN
    (SELECT
      V.PART_ID,
      V.ORDER_POINT
    FROM
      PARTS_TEMP V,
      (SELECT
        PART_ID,
        MIN(ORDER_POINT) ORDER_POINT
      FROM
        PARTS_TEMP
      GROUP BY
        PART_ID
      HAVING
        COUNT(*)>1) M
    WHERE
      V.PART_ID=M.PART_ID
      AND V.ORDER_POINT>M.ORDER_POINT);

INSERT INTO
  PARTS
SELECT
  *
FROM
```

```
  PARTS_TEMP;

UPDATE
  PARTS P
SET
  (PRIMARY_WHS_ID,PRIMARY_LOC_ID)=(
    SELECT
      WAREHOUSE_ID,
      LOCATION_ID
    FROM
      (SELECT
        WAREHOUSE_ID,
        LOCATION_ID,
        ROWNUM RN
      FROM
        LOCATIONS)
    WHERE
      MOD(P.ORDER_POINT,2000)=RN);

UPDATE
  PARTS
SET
  PREF_VENDOR_ID=TRIM(TO_CHAR(ABS(ROUND(COS((MOD(ROWNUM,9000)*2+1)*3.14159265/180*51.491976)*10000000,0))))||'VEN'
WHERE
  PURCHASED='Y';

COMMIT;

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'PARTS',CASCADE=>TRUE);

SELECT 'INSERTING INTO PO_HEADER' FROM DUAL;
ALTER SESSION SET TRACEFILE_IDENTIFIER = 'INSERT_PO_HEADER';

INSERT INTO
  PO_HEADER_TEMP
SELECT
  'PO'||TO_CHAR(ROWNUM) PURC_ORDER_ID,
  TRIM(TO_CHAR(ABS(ROUND(COS((MOD(ROWNUM,9000)*2+1)*3.14159265/180*51.491976)*10000000,0))))||'VEN' VENDOR_ID,
  NULL CONTACT_FIRST_NAME,
  NULL CONTACT_LAST_NAME,
  NULL CONTACT_INITIAL,
  NULL CONTACT_POSITION,
  NULL CONTACT_HONORIFIC,
  NULL CONTACT_SALUTATION,
  NULL CONTACT_PHONE,
  NULL CONTACT_FAX,
  1 PURC_ORD_ADDR_NO,
  1 SHIPTO_ADDR_NO,
  TRUNC(SYSDATE-(COS(ROWNUM*3.14159265/180)*1000)) ORDER_DATE,
  TRUNC(SYSDATE-(COS(ROWNUM*3.14159265/180)*1000))+10 DESIRED_RECV_DATE,
  'MY_BUYER' BUYER,
  NULL FREE_ON_BOARD,
  'BEST WAY' SHIP_VIA,
  'REGULAR' SALES_TAX_GROUP_ID,
  DECODE(MOD(ROWNUM,6),1,'F',2,'R',3,'X','C') PO_STATUS,
  'N' BACK_ORDER,
  1 SELL_RATE,
  1 BUY_RATE,
  '1' ENTITY_ID,
  DECODE(MOD(ROWNUM,3),1,'Y','N') POSTING_CANDIDATE,
  NULL LAST_RECEIVED_DATE,
  0 TOTAL_AMT_ORDERED,
  0 TOTAL_AMT_RECVD,
  'N' MARKED_FOR_PURGE,
  'Y' EXCH_RATE_FIXED,
  TRUNC(SYSDATE-(COS(ROWNUM*3.14159265/180)*1000))+10 PROMISE_DATE,
  SYSDATE PRINTED_DATE,
  DECODE(MOD(ROWNUM,6),0,'A',1,'M',2,'D',3,'N','E') TERMS_DISC_TYPE,
  NULL EDI_BLANKET_FLAG,
  NULL EDI_BLANKET_PO_NO,
  1 CONTRACT_ID,
  1 SHIPTO_ID,
  DECODE(MOD(ROWNUM,6),0,'A',1,'M',2,'D',3,'N','E') TERMS_NET_TYPE,
  1 TERMS_NET_DAYS,
  NULL TERMS_NET_DATE,
  1 TERMS_DISC_DAYS,
  NULL TERMS_DISC_DATE,
  3 TERMS_DISC_PERCENT,
  'ON TIME' TERMS_DESCRIPTION,
  'USD' CURRENCY_ID,
  NULL WAREHOUSE_ID,
  SYSDATE CREATE_DATE,
  NULL CONTACT_MOBILE,
  NULL CONTACT_EMAIL,
  NULL USER_1,
  NULL USER_2,
  NULL USER_3,
  NULL USER_4,
  NULL USER_5,
  NULL USER_6,
  NULL USER_7,
  NULL USER_8,
  NULL USER_9,
  NULL USER_10,
  'DEFAULT' UDF_LAYOUT_ID,
  'TESTING' PO_CREATE_USER_ID
FROM
  DUAL
CONNECT BY
  LEVEL<=500000;

INSERT INTO
  PO_HEADER
SELECT
  *
FROM
  PO_HEADER_TEMP;

COMMIT;
```

```
EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'PO_HEADER',CASCADE=>TRUE);

SELECT 'INSERTING INTO PO_LINES' FROM DUAL;
ALTER SESSION SET TRACEFILE_IDENTIFIER = 'INSERT_PO_LINES';

INSERT INTO PO_LINE_TEMP (
  PURC_ORDER_ID,
  LINE_NO,
  PART_ID,
  VENDOR_PART_ID,
  SERVICE_ID,
  USER_ORDER_QTY,
  ORDER_QTY,
  PURCHASE_UM,
  UNIT_PRICE,
  FIXED_CHARGE,
  GL_EXPENSE_ACCT_ID,
  SALES_TAX_GROUP_ID,
  PRODUCT_CODE,
  COMMODITY_CODE,
  DESIRED_RECV_DATE,
  TRADE_DISC_PERCENT,
  EST_FREIGHT,
  LINE_STATUS,
  TOTAL_ACT_FREIGHT,
  TOTAL_USR_RECD_QTY,
  TOTAL_RECEIVED_QTY,
  TOTAL_AMT_RECVD,
  TOTAL_AMT_ORDERED,
  TOTAL_DISPATCH_QTY,
  TOTAL_USR_DISP_QTY,
  ALLOCATED_QTY,
  FULFILLED_QTY)
SELECT /*+ ORDERED */
  PURC_ORDER_ID,
  ORDER_POINT-START_LINE+1,
  PART_ID,
  PART_ID,
  NULL,
  10,
  10,
  DECODE(MOD(ROWNUM,20),2,'PC',3,'FT',4,'METER',5,'KG',6,'CASE',7,'LBS',8,'DOZEN','EA'),
  1099.99,
  0,
  NULL,
  NULL,
  PRODUCT_CODE,
  COMMODITY_CODE,
  TRUNC(SYSDATE-1000+ROWNUM/1000),
  0,
  0,
  'A',
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0
FROM
  (SELECT
    'PO'||TO_CHAR(ROWNUM) PURC_ORDER_ID,
    ROWNUM RN,
    ABS(SIN(ROWNUM*3.14159265/180))*90000 START_LINE,
    MOD(ROWNUM,50)+1 LINES
  FROM
    DUAL
  CONNECT BY
    LEVEL<=500000) POL,
  PARTS P
WHERE
  P.ORDER_POINT BETWEEN START_LINE AND (START_LINE+LINES-1);

INSERT INTO
  PO_LINE
SELECT
  *
FROM
  PO_LINE_TEMP;

COMMIT;

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'PO_LINE',CASCADE=>TRUE);

SELECT 'UPDATE-ROLLBACK TEST' FROM DUAL;
ALTER SESSION SET TRACEFILE_IDENTIFIER = 'ROLLBACK_TEST';

UPDATE
  PO_LINE
SET
  PART_ID='8729425PART'
WHERE
  PART_ID BETWEEN '3000000PART' AND '6576035PART';

ROLLBACK;

SELECT 'INSERT-NARROW-TABLE' FROM DUAL;
ALTER SESSION SET TRACEFILE_IDENTIFIER = 'NARROW_TABLE';

INSERT INTO
  NARROW
SELECT
  ROWNUM,
  NULL
FROM
  DUAL
CONNECT BY
  LEVEL<=900000;
```

```
COMMIT;

SELECT
  SUBSTR(SN.NAME,1,25) STAT_NAME,
  MS.VALUE
FROM
  V$STATNAME SN,
  V$MYSTAT MS
WHERE
  SN.NAME IN ('table fetch by rowid','table scan rows gotten','table fetch continued row','table scan blocks gotten','consistent
gets')
  AND SN.STATISTIC#=MS.STATISTIC#
ORDER BY
  SN.NAME;

UPDATE
  NARROW
SET
  C1=ROUND(SIN(C1*3.14159265/180),2),
  C2=C1;

SELECT
  SUBSTR(SN.NAME,1,25) STAT_NAME,
  MS.VALUE
FROM
  V$STATNAME SN,
  V$MYSTAT MS
WHERE
  SN.NAME IN ('table fetch by rowid','table scan rows gotten','table fetch continued row','table scan blocks gotten','consistent
gets')
  AND SN.STATISTIC#=MS.STATISTIC#
ORDER BY
  SN.NAME;

UPDATE
  NARROW
SET
  C1=ROUND(SIN(C2*3.14159265/180),10);

UPDATE
  NARROW
SET
  C2=C1;

SELECT
  SUBSTR(SN.NAME,1,25) STAT_NAME,
  MS.VALUE
FROM
  V$STATNAME SN,
  V$MYSTAT MS
WHERE
  SN.NAME IN ('table fetch by rowid','table scan rows gotten','table fetch continued row','table scan blocks gotten','consistent
gets')
  AND SN.STATISTIC#=MS.STATISTIC#
ORDER BY
  SN.NAME;

SELECT
  *
FROM
  NARROW;

SELECT
  SUBSTR(SN.NAME,1,25) STAT_NAME,
  MS.VALUE
FROM
  V$STATNAME SN,
  V$MYSTAT MS
WHERE
  SN.NAME IN ('table fetch by rowid','table scan rows gotten','table fetch continued row','table scan blocks gotten','consistent
gets')
  AND SN.STATISTIC#=MS.STATISTIC#
ORDER BY
  SN.NAME;

DELETE FROM
  NARROW
WHERE
  C1<0;

COMMIT;

SELECT 'TABLE AND INDEX STATS' FROM DUAL;
ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF'

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'NARROW',CASCADE=>TRUE);

SELECT
  TABLE_NAME,
  NUM_ROWS,
  BLOCKS,
  AVG_ROW_LEN
FROM
  USER_TABLES
WHERE
  TABLE_NAME IN ('PO_HEADER','PO_LINE','PARTS','VENDORS','LOCATIONS','UMS','NARROW')
ORDER BY
  TABLE_NAME;

SELECT
  SUBSTR(TABLE_NAME,1,10) TABLE_NAME,
  SUBSTR(INDEX_NAME,1,15) INDEX_NAME,
  BLEVEL,
  LEAF_BLOCKS,
  DISTINCT_KEYS,
  AVG_LEAF_BLOCKS_PER_KEY,
  AVG_DATA_BLOCKS_PER_KEY,
  CLUSTERING_FACTOR
FROM
```

```
  USER_INDEXES
WHERE
  TABLE_NAME IN ('PO_HEADER','PO_LINE','PARTS','VENDORS','LOCATIONS','UMS','NARROW')
ORDER BY
  TABLE_NAME,
  INDEX_NAME;

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;

ALTER SESSION SET TRACEFILE_IDENTIFIER = 'SELECT_TEST';
ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 8';

SELECT
  PO.VENDOR_ID,
  P.PRODUCT_CODE,
  P.STOCK_UM,
  SUM(POL.ORDER_QTY) ORDER_QTY
FROM
  PO_HEADER PO,
  PO_LINE POL,
  PARTS P
WHERE
  PO.ORDER_DATE BETWEEN TRUNC(SYSDATE-90) AND TRUNC(SYSDATE)
  AND PO.PURC_ORDER_ID=POL.PURC_ORDER_ID
  AND POL.PART_ID=P.ID
GROUP BY
  PO.VENDOR_ID,
  P.PRODUCT_CODE,
  P.STOCK_UM;

SELECT
  POL.PART_ID,
  P.DESCRIPTION,
  MAX(DESIRED_RECV_DATE) LAST_RECEIVE_DATE
FROM
  PO_LINE POL,
  PARTS P
WHERE
  P.PRODUCT_CODE='FG'
  AND P.ABC_CODE='C'
  AND P.PART_ID=POL.PART_ID
GROUP BY
  POL.PART_ID,
  P.DESCIPTION;

SELECT
  COUNT(*) LOCATIONS
FROM
  LOCATIONS;

SELECT
  PRODUCT_CODE,
  COUNT(*) PARTS_LARGE_WH
FROM
  (SELECT
    WAREHOUSE_ID
  FROM
    LOCATIONS
  GROUP BY
    WAREHOUSE_ID
  HAVING
    COUNT(*)>160) W,
  PARTS P
WHERE
  W.WAREHOUSE_ID=P.PRIMARY_WHS_ID
GROUP BY
  PRODUCT_CODE
ORDER BY
  PRODUCT_CODE;

SELECT
  COUNT(*)
FROM
  PARTS
WHERE
  QTY_ON_HAND>1000;

SELECT
  COUNT(*)
FROM
  VENDORS
WHERE
  ZIPCODE>' 44444';

SELECT
  COUNT(*)
FROM
  PO_LINE POL,
  PARTS P
WHERE
  POL.PURC_ORDER_ID BETWEEN '10000' AND '20000'
  AND POL.PART_ID=P.PART_ID;

SELECT
  PART_ID,
  ABC_CODE,
  PRODUCT_CODE,
  MAX(QTY_ON_HAND) OVER (PARTITION BY PRODUCT_CODE,ABC_CODE) MAX_QTY_PRD_ABC,
  MIN(QTY_ON_HAND) OVER (PARTITION BY PRODUCT_CODE,ABC_CODE) MIN_QTY_PRD_ABC,
  DENSE_RANK() OVER (PARTITION BY PRODUCT_CODE,ABC_CODE ORDER BY QTY_ON_HAND) DR_QTY_PRD_ABC,
  DENSE_RANK() OVER (PARTITION BY PREF_VENDOR_ID ORDER BY ORDER_POINT) DR_OP_VEND
FROM
  PARTS
ORDER BY
  PART_ID;

SELECT
  V.VENDOR_ID,
  V.VENDOR_NAME
```

```
FROM
  VENDORS V,
  (SELECT DISTINCT
    PO.VENDOR_ID
  FROM
    PO_HEADER PO,
    PO_LINE POL,
    PARTS P
  WHERE
    PO.PURC_ORDER_ID=POL.PURC_ORDER_ID
    AND POL.PART_ID=P.PART_ID
    AND P.PRODUCT_CODE='FG') PV
WHERE
  V.VENDOR_ID=PV.VENDOR_ID(+)
  AND PV.VENDOR_ID IS NULL
ORDER BY
  V.VENDOR_ID;

SELECT
  PART_ID,
  DESCRIPTION,
  QTY_ON_HAND,
  RANK() OVER (PARTITION BY PRODUCT_CODE ORDER BY QTY_ON_HAND DESC NULLS LAST) RANK_PC_QTY,
  AVG(QTY_ON_HAND) OVER (PARTITION BY PRODUCT_CODE ORDER BY QTY_ON_HAND) AVG_PC_QTY,
  MIN(QTY_ON_HAND) OVER (PARTITION BY PRODUCT_CODE ORDER BY QTY_ON_HAND) MIN_PC_QTY,
  MAX(QTY_ON_HAND) OVER (PARTITION BY PRODUCT_CODE ORDER BY QTY_ON_HAND) MAX_PC_QTY,
  COUNT(UNIT_MATERIAL_COST) OVER (PARTITION BY PRODUCT_CODE ORDER BY UNIT_MATERIAL_COST) COUNT_PC,
  RANK() OVER (PARTITION BY COMMODITY_CODE ORDER BY QTY_ON_HAND DESC NULLS LAST) RANK_CC_QTY,
  AVG(QTY_ON_HAND) OVER (PARTITION BY COMMODITY_CODE ORDER BY QTY_ON_HAND) AVG_CC_QTY,
  MIN(QTY_ON_HAND) OVER (PARTITION BY COMMODITY_CODE ORDER BY QTY_ON_HAND) MIN_CC_QTY,
  MAX(QTY_ON_HAND) OVER (PARTITION BY COMMODITY_CODE ORDER BY QTY_ON_HAND) MAX_CC_QTY,
  COUNT(QTY_ON_HAND) OVER (PARTITION BY COMMODITY_CODE ORDER BY QTY_ON_HAND) COUNT_CC,
  RANK() OVER (PARTITION BY NVL(PREF_VENDOR_ID,'IN_HOUSE_FAB') ORDER BY QTY_ON_HAND DESC NULLS LAST) RANK_VENDOR_QTY,
  AVG(QTY_ON_HAND) OVER (PARTITION BY NVL(PREF_VENDOR_ID,'IN_HOUSE_FAB') ORDER BY QTY_ON_HAND) AVG_VENDOR_QTY,
  MIN(QTY_ON_HAND) OVER (PARTITION BY NVL(PREF_VENDOR_ID,'IN_HOUSE_FAB') ORDER BY QTY_ON_HAND) MIN_VENDOR_QTY,
  MAX(QTY_ON_HAND) OVER (PARTITION BY NVL(PREF_VENDOR_ID,'IN_HOUSE_FAB') ORDER BY QTY_ON_HAND) MAX_VENDOR_QTY,
  COUNT(QTY_ON_HAND) OVER (PARTITION BY PREF_VENDOR_ID ORDER BY QTY_ON_HAND) COUNT_VENDOR
FROM
  PARTS
ORDER BY
  PART_ID;

SELECT
  PRODUCT_CODE,
  RANK(1) WITHIN GROUP (ORDER BY QTY_ON_HAND DESC NULLS LAST) UNIT_PRICE,
  RANK(2) WITHIN GROUP (ORDER BY QTY_ON_HAND DESC NULLS LAST) UNIT_PRICE,
  RANK(3) WITHIN GROUP (ORDER BY QTY_ON_HAND DESC NULLS LAST) UNIT_PRICE,
  RANK(4) WITHIN GROUP (ORDER BY QTY_ON_HAND DESC NULLS LAST) UNIT_PRICE,
  RANK(5) WITHIN GROUP (ORDER BY QTY_ON_HAND DESC NULLS LAST) UNIT_PRICE
FROM
  PARTS
GROUP BY
  PRODUCT_CODE
ORDER BY
  PRODUCT_CODE;

SELECT
  PO.PART_ID,
  P.DESCRIPTION,
  PO.VENDOR_ID,
  PO.CREATE_DATE,
  PO.UNIT_PRICE,
  PO.LAST_VENDOR_ID,
  PO.LAST_CREATE_DATE,
  PO.LAST_UNIT_PRICE,
  P.PRODUCT_CODE,
  P.COMMODITY_CODE
FROM
  (SELECT
    POL.PART_ID,
    PO.VENDOR_ID,
    TRUNC(NVL(POL.POL_CREATE_DATE,PO.CREATE_DATE)) CREATE_DATE,
    POL.UNIT_PRICE,
    LEAD(PO.VENDOR_ID,1,NULL) OVER (PARTITION BY PART_ID ORDER BY NVL(POL.POL_CREATE_DATE,PO.CREATE_DATE) DESC) LAST_VENDOR_ID,
    TRUNC(LEAD(NVL(POL.POL_CREATE_DATE,PO.CREATE_DATE),1,NULL) OVER (PARTITION BY PART_ID ORDER BY
NVL(POL.POL_CREATE_DATE,PO.CREATE_DATE) DESC))  LAST_CREATE_DATE,
    LEAD(POL.UNIT_PRICE,1,NULL) OVER (PARTITION BY PART_ID ORDER BY NVL(POL.POL_CREATE_DATE,PO.CREATE_DATE) DESC) LAST_UNIT_PRICE
  FROM
    PO_HEADER PO,
    PO_LINE POL
  WHERE
    PO.ID=POL.PURC_ORDER_ID
    AND PO.CREATE_DATE>TRUNC(SYSDATE-720)
  ORDER BY
    POL.PART_ID,
    NVL(POL.POL_CREATE_DATE,PO.CREATE_DATE) DESC) PO,
  PARTS P
WHERE
  PO.PART_ID=P.PART_ID
  AND PO.CREATE_DATE>TRUNC(SYSDATE-90)
  AND (PO.VENDOR_ID<>NVL(PO.LAST_VENDOR_ID,'-')
    OR PO.CREATE_DATE>(NVL(PO.LAST_CREATE_DATE,SYSDATE-1024)+180)
    OR PO.UNIT_PRICE<>NVL(PO.LAST_UNIT_PRICE,-1));

SELECT 'FINISHED' FROM DUAL;
ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF'

SPOOL OFF
```

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Maran Viswarayar

Posts: 4,196

Re: Larger vs. Small data block

Posted: Jun 16, 2008 10:25 AM in response to: Charles Hooper

Reply

From: Cecil,Singapore
Registered: 9/23/05

Charles

I am following the entire thread..Got amazed with skills and Patience...

Finally i will participate in the forums...using your scripts

Excellent Work !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 16, 2008 10:47 AM in response to: Maran Viswarayar

Reply

> I am following the entire thread..Got amazed with
> skills and Patience...
>
> Finally i will participate in the forums...using your
> scripts
>
> Excellent Work !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Maran,

Thanks. It required 8+ hours to build the script to generate the non-sequential data, and I have not had a chance to formally test it yet to compare performance.

During the intial test run (into a 16KB database left over from previous testing), the insert into PO_LINE_TEMP required a bit less then 4 minutes to complete, while the copy from PO_LINE_TEMP to PO_LINE required about 75 minutes. The initial creation of the NARROW table completed quickly, but updates on that table were painfully slow due to PCTFREE not being specified for the table. As I mentioned, I have only had a brief chance to look the output of the initial test run due to time contraints.

Let me know the results if you perform the test. You may want to pre-size the USER_DATA tablespace to 8GB (or larger) if you use the setup that I posted in an earlier reply to this thread.

damorgan, if you are interested in trying the script on your RAC setup, let me know and I will forward the script to you.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Maran Viswarayar

Posts: 4,196
From: Cecil,Singapore
Registered: 9/23/05

Re: Larger vs. Small data block
Posted: Jun 16, 2008 10:56 AM in response to: Charles Hooper

Reply

Thanks

I will try

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 16, 2008 11:29 AM in response to: Steve Karam

Reply

>
> Am I translating this right? You're basically saying
> that we should take the simple assumption that I did
> it wrong in favor of the possibility something else
> was amiss? I was actually quite liking some of the
> possibilities you brought up in your interpretations
> up to this point. This just seems like giving up.
>

Good news, I think I can emulate your problem - and give you the solution. (That's assuming my guesses about your setup are correct).

I'm just running a test to completion - and I'll let you know the results.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 16, 2008 12:07 PM in response to: Charles Hooper

Reply

> > using your scripts
> >
> > Excellent Work !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Hi Charles,

is it possible to get your script per email or to download it from specific web-location as already formated file?

Thanks!

David_Aldridge

Posts: 97
Registered: 4/22/08

Re: Larger vs. Small data block
Posted: Jun 16, 2008 12:16 PM in response to: user619401

Reply

The longer this thread goes on, the more I feel like just throwing my hands in the air and saying "8kb it is!" ... OLTP, warehouse, whatever.

Can I get an "amen" on that?

Hans Forbrich

Re: Larger vs. Small data block
Posted: Jun 16, 2008 12:25 PM in response to: David_Aldridge

Reply

Posts: 7,483
From: AB, Canada
Registered: 3/13/99

> The longer this thread goes on, the more I feel like
> just throwing my hands in the air and saying "8kb it
> is!" ... OLTP, warehouse, whatever.
>
> Can I get an "amen" on that?

I have tried to 'amen' that several times.

Unless there is a compelling reason and other alternatives have been exhausted, using the default is often good enough. It's a happy compromise that will work well in most cases. There are, of course, exceptions.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 16, 2008 3:58 PM in response to: Charles Hooper
Reply

Definitely interested.

I'm sitting in Denver International Airport awaiting my delayed flight to New Orleans that will hopefully get me in for my presentations at ODTUG's Kaleidoscope tomorrow.

Running a RAC class Wed/Thu/Fri so I will have available at least four 2 node clusters when class is over.

Thanks.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 16, 2008 4:03 PM in response to: David_Aldridge
Reply

You can from me.
I've no doubt you can from Brynn too.

Seems to me it is about time for Greg, Graham, and a few others inside to belly up to the keyboard, write a definitive statement on the subject, and post it to OTN and metalink.

This "controversy" leads to wasted time, wasted effort, and in the end makes Oracle look bad because it seems to have no official opinion on the matter.

While you're at it please also cut down the body of multiple block sizes in a single database and a few other oft repeated myths.

Thank you.

$@m!
@$m@

Posts: 537
From: AUH
Registered: 7/3/07

Re: Larger vs. Small data block
Posted: Jun 16, 2008 4:15 PM in response to: Charles Hooper
Reply

It took me lot of time to go though the complete script. I have some issues in understanding it, I will post my doubts.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 16, 2008 4:48 PM in response to: Jonathan Lewis
Reply

>
> Good news, I think I can emulate your problem - and
> give you the solution. (That's assuming my guesses
> about your setup are correct).
>

Headline results for update:

```
16KB Block size:          1 hour 36 minutes 45.06 seconds
 8KB Block size:                  1 minute   1.08 seconds
 4KB Block size:                  1 minute  28.00 seconds
```

The tablespaces are locally managed with a uniform extent size of 128KB and using ASSM which, I think, is in accordance with the description given by Steve Karam.

The SQL for creating, populating, and updating the table is given below.

You will note that in my test case the rows are very short, and the updated column starts out null. A typical row starts at 9 bytes (11 if you count the row index entry), and grows to 15 (17) bytes. This means that the default **pctfree** of 10 is much too small, and a large number of rows will migrate leaving a 9 (11) byte forwarding address. This means that the table needs to be defined with a **pctfree** of around 35 if it is avoid problems with rows migrating. (In my second test run I used 50 to avoid having to be too exact.)

A combination of short rows, mass row extension, poor choice of pctfree, and large blocks seems to cause ASSM some problems identifying a block that will be able to accept a migrated row - and it uses a lot of resources searching for a suitable block.

There was a bug of this nature in early releases of ASSM, but I thought it had been fixed. Possibly the fix had an arithmetical component that was based on an 8KB block size and was not tested in extreme cases against larger block sizes.

```
execute dbms_random.seed(0);

drop table t1;

create table t1 (
             n1           number,
             n2           number
)
-- pctfree 50
tablespace test_4k_assm
;

insert into t1
with generator as (
             select       --+ materialize
                          rownum       id
             from         all_objects
             where        rownum <= 3000
)
```

```
select
             trunc(dbms_random.value(10000000,100000000))       n1,
             to_number(null)                                                       n2
--           trunc(dbms_random.value(10000000,100000000))       n2
from
             generator    v1,
             generator    v2
where
             rownum <= 830000
;

commit;

alter session set events '10046 trace name context forever, level 8';

update t1 set n2 = n1;
```

As part of my test code, I also took snapshots of ***v$mystat***, ***v$session_event***, and ***x$kcbsw/x$kcbwh*** (see: http://www.jlcomp.demon.co.uk/buffer_usage.html for further details on the last one). Here are some of the key statistics:

```
16KB Block size
---------------
                                                    pctfree 10    pctfree 50
Time                                                1:36:45.06         21.07
Wait time                                     2.94       12.65
db block get                           845,084,110      848,345
redo entries                             2,161,504      830,503
redo size                              491,906,180  186,504,584

Critical buffer get calls
ktspfwh10: ktspscan_bmb                144,587,672            0
ktspbwh1: ktspfsrch                    696,965,277            0


 8KB Block size
---------------
                                                    pctfree 10    pctfree 50
Time                                                   1:01.08         20.01
Wait time                                    19.16       11.69
db block get                             5,526,488      848,321
redo entries                             2,172,130      830,399
redo size                              492,560,476  186,542,972

Critical buffer get calls
ktspfwh10: ktspscan_bmb                  1,320,444            0
ktspbwh1: ktspfsrch                        664,235            0


 4KB Block size
---------------
                                                    pctfree 10    pctfree 50
Time                                                   1:28.00         21.04
Wait time                                    39.47       13.01
db block get                             5,547,182      851,170
redo entries                             2,183,488      830,458
redo size                              493,455,356  186,632,124

Critical buffer get calls
ktspfwh10: ktspscan_bmb                  1,321,618            0
ktspbwh1: ktspfsrch                        668,945            0
```

Most of the wait time recorded in my tests was due to ***log buffer space*** waits.

You will note that my model is obviously not an exact match for the details Steve Karam gave - compared to his figures, the increase I saw in current gets is too large and the increase in redo log may not be large enough to be particularly significant. This suggests that a smaller percentage of rows in his data were subject to migration, and that some of the excess work may have been related to delayed block cleanout.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." (Stephen Hawking)

David_Aldridge

Posts: 97
Registered: 4/22/08

Re: Larger vs. Small data block
Posted: Jun 16, 2008 8:09 PM in response to: Jonathan Lewis — Reply

Jonathan -- very interesting indeed.

I was thinking about how to turn this situation around to find a list of signs that might indicate that such a problem is being experienced. Do you think it would then be fair to say that a notable percentage of migrated rows + ASSM + block size greater than 8kb ought at least be enough to raise suspicion?

Steve Karam

Posts: 126
From: Virginia Beach, VA
Registered: 9/14/05

Re: Larger vs. Small data block
Posted: Jun 16, 2008 8:31 PM in response to: David_Aldridge — Reply

Okay, so it's harder to leave a thread alone than I thought.

Jonathan, your observations are very interesting. While pctfree could be said to be the centerpiece of your test, it still seems to point to a possible ASSM/large-block deficiency (due to the drastically skewed results between 4, 8, and 16k). What version of Oracle did you run for the test (sorry if you already gave it, I may have missed it)?

David, that may be enough to warrant some extra investigation; however, I'd probably add at least a check on the avg. row length and PCTFREE as well based upon Jonathan's test case. I'm just thinking of a table with migrated/chained rows in a 32k blocksize ASSM tablespace, but the table also happens to have a heavily used LOB.

Howardjr

Posts: 11
Registered: 6/7/07

Re: Larger vs. Small data block
Posted: Jun 17, 2008 12:10 AM in response to: Jonathan Lewis

Reply

[a] poor choice of pctfree, and large blocks seems to cause ASSM some problems identifying a block that will be able to accept a migrated row

See. I told you ASSM was evil!

:-)

Good to see you nailing this one down a bit.

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 17, 2008 1:10 AM in response to: David_Aldridge

Reply

> The longer this thread goes on, the more I feel like just throwing my hands in the air and saying "8kb itis!" ... OLTP, warehouse, whatever.
>
> Can I get an "amen" on that?

Amen brother!

This is precisely my position on this topic. I've mentioned it before somewhere I believe...I call 8k the Goldilocks of block sizes: Not to big, not to small, just right! There is a reason that 8192 is the default for db_block_size. Stick with the defaults unless you have a proven and understood reason to deviate (the key word being *understood*!).

If you are noticing more than a few percent difference by changing block sizes, there is likely something you are not noticing!

--
Regards,

Greg Rahn
http://structureddata.org

SeanMacGC

Posts: 7
Registered: 10/30/06

Re: Larger vs. Small data block
Posted: Jun 17, 2008 5:03 AM in response to: Jonathan Lewis

Reply

Light finally penetrates the heat!

Very interesting Jonathan.

So, all distilled to: *it depends*!

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 17, 2008 5:53 AM in response to: SeanMacGC

Reply

I've been away for the past week or so.

Have I missed much ?

Cheers ;)

Richard Foote
http://richardfoote.wordpress.com/

Terrible

Posts: 334
From: York, UK
Registered: 6/25/04

Re: Larger vs. Small data block
Posted: Jun 17, 2008 6:10 AM in response to: Richard Foote

Reply

Same old, same old really......

Xxx made a statement in response to the OP.

The 'usual suspects' jumped all over it and asked for some evidence.

Xxx didn't produce any, from what I remember it was because he has a degree from an 'Ivy league University' and the others wouldn't share their credentials.

Some really interesting test cases and technical discussion followed.....with the usual level of baiting and finger pointing of course.

I don't think the OP actually got a definitive answer although I'm willing to bet he left scratching his sore head and decided to stick with 8k blocks....

Did I miss anything.....?

---

orafad

Posts: 4,976
From: Sweden
Registered: 2/4/99

Re: Larger vs. Small data block
Posted: Jun 17, 2008 6:11 AM in response to: Richard Foote

Reply

> I've been away for the past week or so.

How was Stockholm? :)

Jonathan Lewis

Re: Larger vs. Small data block
Posted: Jun 17, 2008 7:22 AM in response to: Steve Karam

Reply

Posts: 786
From: UK
Registered: 1/23/07

> **Jonathan**, your observations are very
> interesting. While pctfree could be said to be the
> centerpiece of your test, it still seems to point to
> a possible ASSM/large-block deficiency (due to the
> drastically skewed results between 4, 8, and 16k).
> What version of Oracle did you run for the test
> (sorry if you already gave it, I may have missed
> it)?

The fact that the pctfree highlighted the bug doesn't make the pctfree the guilty party; I think there's no question that the bug is in the ASSM code, and perhaps it can only become visible in 16KB (and larger) blocks. It's possible that people haven't seen the bug before simply because the problem doesn't appear often and then becomes self-correcting over time.

The test case I've produced just manages to hit the combination of circumstances that turns what is normally a minor error into a total disaster by picking a pctfree that forces a lot of row migration. I certainly wouldn't want to suggest that the pctfree was the cause

I created the test on 9.2.0.8 - because I think that's the version you said your client was on. The test case shows the same behaviour on 10.2.0.3 and 11.1.0.6

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 17, 2008 7:37 AM in response to: David_Aldridge

Reply

> Jonathan -- very interesting indeed.
>
> I was thinking about how to turn this situation
> around to find a list of signs that might indicate
> that such a problem is being experienced. Do you
> think it would then be fair to say that a notable
> percentage of migrated rows + ASSM + block size
> greater than 8kb ought at least be enough to raise
> suspicion?

Tricky, because dbms_stats() doesn't collect information about chained or migated rows (I haven't checked that for 11g though), and so far we only see the problem appearing with migrated rows. It might apply to chained rows, it might apply when a delete/insert takes place near the boundary between "full" and "not full".

It doesn't even need to be a notable percentage of migrated rows - what if every row you migrate causes oracle to leave a block that's been migrated from as a 'bust be checked block'. You could be in a position where 1,000 migrated rows turns into 1,000 blocks always being checked for every single row insert. This is speculation of course - until we know the nature of the bug we can't work out a complete strategy for identification.

Your suggestion could give us a reason for testing a table - but might miss some tables: but that's better than nothing. Critically, the only reason for testing is if you think a process is doing too many current gets for the volumn of data inserted (which typically ought to be be in the ballpark of 2 + 3 per index).

So if you have any suspect tables, according to your suggested rule, a simple 'insert row into table' might confirm your suspicion. You might have to do this from several different sessions though, as the initial block selected depends on your process id - and you may get lucky/unlucky on the first attempt.

I believe there's a procedure to do an official fix on bitmap blocks which have gone out of synch with the data - possibly in package dbms_space_admin. Perhaps this would be a valid reason for using that package.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 17, 2008 7:40 AM in response to: Terrible

Reply

>
> I don't think the OP actually got a definitive answer
> although I'm willing to bet he left scratching his
> sore head and decided to stick with 8k blocks....
>

Perhaps the best answer to the question should be: "*If you need to ask what size your blocks should be, the answer is 8KB*".

(Who was it who said: "If you need to ask how much it costs to run a motor yacht, you can't afford to own a motor yacht." ?)

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

Terrible

Posts: 334
From: York, UK
Registered: 6/25/04

Re: Larger vs. Small data block
Posted: Jun 17, 2008 8:54 AM in response to: Jonathan Lewis

Reply

I think that was probably the big man Larry himself:

I remember reading a quick story about him in Computer Weekly a while ago, from what I remember he'd tried to buy a new yacht on his credit card but the transaction went above his available limit(!).

Now that sort of thing is imaginable for your everyday person however my jaw dropped when the article stated his credit card limit was $500 million!

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 17, 2008 9:51 AM in response to: Terrible

Reply

> Did I miss anything.....?

You seemed to sum it all quite well except perhaps for the fact tuning by "intuition" is now an approved method. I'm going to try it out tomorrow when I get to work; just sit on the floor with my legs crossed, eyes closed and just "feel" with my senses what any potential performance problems might be. I honestly believe with the right candles, background music, prevailing wind direction and the right harmonies in my humming, any evil spirits within the Oracle databases will make themselves visible and I'll be able to just fix things as appropriate.

It's a real shame Oracle doesn't have a 42K block size by default ...

Cheers ;)

Richard Foote
http://richardfoote.wordpress.com/

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 17, 2008 9:53 AM in response to: orafad

Reply

> How was Stockholm? :)

Great !!

http://richardfoote.wordpress.com/2008/06/17/ot-stockholm-and-utrecht/

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Reega

Posts: 301
From: USA
Registered: 12/21/99

Re: Larger vs. Small data block
Posted: Jun 17, 2008 10:00 AM in response to: Richard Foote

Reply

Richard,
I am excited to attend your class in Seattle. See you here in US soon :)
I will get chance to see Jonathan and Kyte again ....
Why would't you present in hotsos seminar ? or Did I miss it ?

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 17, 2008 10:19 AM in response to: Reega

Reply

Hi Reega

Just note the PSOUG website still has the wrong list of topics. They're as specified here:

http://richardfoote.wordpress.com/oracle-index-internals-seminar/

Unfortunately, I only have so much time within the year I can devote to training, maybe next year I'll get the time to present at Hotsos.

Looking forward to meeting you soon :)

Cheers

Richard Foote
http://richardfoote.wordpress.com/

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 17, 2008 12:00 PM in response to: Jonathan Lewis

Reply

You wrote:
"I think there's no question that the bug is in the ASSM code, and perhaps it can only become visible in 16KB (and larger) blocks."

Which brings us full circle to the statement Brynn made to me and that I have repeated several times in this thread. Oracle only tests 8K blocks. So I have no doubt there are many issues to be discovered by those that follow holistic rather than scientific advise with respect to block sizes. If a DBA is not going to use an 8K block size they'd better have something far more credible to go on than an opinion unsupported by rigorous testing.

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 17, 2008 12:03 PM in response to: Richard Foote

Reply

I understand tuning to David Bowie yields fewer waits. <g>

damorgan

Posts: 4,146
From: Seattle, Washington
Registered: 10/20/03

Re: Larger vs. Small data block
Posted: Jun 17, 2008 12:05 PM in response to: Richard Foote

Reply

I'm going to have to beat the webmaster with a curly brace. I will make the change personally when I get back to Seattle Wednesday ... my Wednesday. <g>

Hans Forbrich

Re: Larger vs. Small data block
Posted: Jun 17, 2008 2:50 PM in response to: damorgan

Reply

(...

Posts: 663
From: Alberta, Canada
Registered: 11/17/06

> I'm going to have to beat the webmaster with a curly
> brace. I will make the change personally when I get
> back to Seattle Wednesday ... my Wednesday. <g>

(ditto, my session <g>)

Greg Rahn

Posts: 61
From: Redwood Shores, California
Registered: 10/3/07

Re: Larger vs. Small data block
Posted: Jun 17, 2008 3:56 PM in response to: damorgan

Reply

> Seems to me it is about time for Greg, Graham, and a few others inside to belly up to the keyboard, write
> a definitive statement on the subject, and post it to OTN and metalink.
>
> This "controversy" leads to wasted time, wasted effort, and in the end makes Oracle look bad because
> it seems to have no official opinion on the matter.

> While you're at it please also cut down the body of multiple block sizes in a single database and a few other oft repeated myths.

It seems that some documentation that exists is either outdated, incomplete, or perhaps unintentionally misleading. Let me know if you find such documentation.

I do know that the RWPG has worked on parts of the Performance Tuning Guide and I just was reading through it and do see a statement that I clearly understand and support:
"The use of multiple block sizes in a single database instance is not encouraged because of manageability issues."[1]
Now, I don't see those infamous official documentation quoter types mentioning that one. Funny like that, huh?

--
Regards,

Greg Rahn
http://structureddata.org

1: http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/iodesign.htm#i19636

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 17, 2008 5:53 PM in response to: benprusinski

Reply

[nobr]Ben,

>
> I was called in for this client after the previous
> Oracle DBA quit on the spot. He was inexperienced and
> built the database with an 8k default size for a 4TB
> data warehouse and reporting financial database on
> Oracle. Fortunately, it was not a production data
> warehouse but rather a copy of production.
>
> For same odd reason, the regular production data
> warehouse used 16k block size and had no issues with
> the ETL nightly jobs which ran between 3-6 hours each
> night.
>

You've described the DBA as inexperienced ; and he's recreated a 4TB database using an extract and reload mechanism (or the block size couldn't have changed from 16KB to 8KB).

How much time were you given to find out what else he might have done that could have caused the performance to drop ? Missing indexes, disk hot spots, constraints enabled when they should have been kept disabled, missing statistics.

There are so many things that could have been done differently - how confident are you that nothing but the blocksize changed ?

Taking a different perspective - are you so sure that it was just the block size that made the difference that you're happy for xxx xxxxxxxxto atrtibute to you the claim that*"Oracle consultant Ben Prusinski notes that batch jobs can see a 3x performance improvement when moved to a larger blocksize"*

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.[/nobr]

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 17, 2008 5:59 PM in response to: Jonathan Lewis

Reply

I see from the comments on Steve Karam's blog that xxx xxxxxxxx is having some difficulty in following the technical bits of the discussion:

http://www.oraclealchemist.com/oracle/hey-guys-does-size-matter/

That's worth remembering the next time you see him insisting that he's seen *"plenty of cases where a change in block size has made a dramatic performance - especially when you have small rows in large blocks"*

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.

Howardjr

Posts: 11

Re: Larger vs. Small data block
Posted: Jun 17, 2008 8:33 PM in response to: Jonathan Lewis

Reply

Registered: 6/7/07

Interesting, in some ways:

Xxx's assertion that 'a change in block size has made a dramatic performance [difference]' is clearly **true**: change from using 16K blocks and you ***will*** stop hitting a massively-performance-sapping ASSM bug.

Xxx's intuition-driven approach to Oracle tuning, of course, means that he had no idea such a bug existed. Indeed, his refusal to believe test cases can be used to demonstrate anything prevents him from uncovering the existence of such bugs. But still, be charitable: his advice to "change block size" might actually have worked (if the change had been *from* 16K, of course, and not *to* it!)

Mark A. Williams

Posts: 1,131
Registered: 4/21/98

Re: Larger vs. Small data block
Posted: Jun 17, 2008 8:50 PM in response to: Jonathan Lewis — Reply

This is sort of a cross-posting of a comment I placed on Steve's blog in response to a comment from David Aldridge...

Here's my comment:

I'm wondering if bug 6918210 might be a good one to watch - it has the confirmed flag set to "Y" and a Dev priority of "2". It is 32KB blocksize, but involves ASSM and row migration.

While the version of the db in the bug is 10.2.0.3, it seems somewhat related to me...

- Mark

David_Aldridge

Posts: 97
Registered: 4/22/08

Re: Larger vs. Small data block
Posted: Jun 17, 2008 9:11 PM in response to: Howardjr — Reply

It seems like the investigative approach that Xxx advocates, and he should feel free to correct me if I'm wrong here because I'm just interpreting from his previous comments, is that production systems should be rebuilt on an *exact* duplicate with only the block size modified, and a real world workload should be replayed on it (outside of 11g, I'm not sure how this would work mind you).

So given that you have an 8kb block size in production, it should be rebuilt on 16kb,32kb, 4kb, and 2kb, and each one compared. That comparison can only be valid after a period of activity to allow indexes to "relax" from their freshly rebuilt state when some operations (reads) are going to be greatly favoured over others (modifications), yet the data should stay broadly the same. When it comes to multiple block sizes in a single database there are other dilemmas -- if you want some tables and indexes on a 16kb block size and some on an 8kb block size, which size should be the default used for the system tablespace? You surely have to try both. And then you have to try different segments on different sizes, because artificial tests mean nothing ...

Is that is?

Howardjr

Posts: 11
Registered: 6/7/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 1:23 AM in response to: David_Aldridge — Reply

I just thought I'd mention it in passing, but I have come across another example of where non-default block sizes appear to be a big no-no (and in passing, it would seem to resolve a mystery about Windows v. Linux performance that was asked here recently -I must remember to annotate that other thread, too).

Short story: Intemedia on Windows with 16K blocks manages to retrieve 700 rows per second. On the same server, with the same instance configuration parameters, but with the table and its index (plus the DR$ tables) all built into 8K tablespace, it manages to retrieve 127,000 rows per second.

Slightly longer details here: http://tinyurl.com/4kawfr

Repeatable on three production servers, so I'm not just making it up! Just another indication that there are lots more 'surprises' lurking for those that stray from the 8K route, I think!

Updated in light of Jonathan's comments on that blog: In case it's not clear from the short-form comment above, the table and its index were freshly built for both the 16K and 8K tests, so the usual objection to such anecdotes that 'the rebuild might be the factor, not the block size' doesn't apply. Both table and index were as freshly-rebuilt in the 16K case as they were in the 8K one.

My point, however, is not that the reduction in block size is significant. It's simply the fact that (I think) an obvious bug associated with the use of large block sizes is avoided by the change to the default block size; just as Jonathan's investigation show an ASSM-related bug is avoided by sticking to default block sizes.

I'm not, in short, arguing that 'small blocks are better'. Merely that non-default block sizes appear to have quite a number of problems associated with their use which makes the sweeping recommendations from some in these parts to deploy them with gusto because TPC benchmarks do so very silly advice to even think of following.

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 18, 2008 9:28 AM in response to: David_Aldridge — Reply

With such a long thread such as this, it's often a useful exercise to summarise some of the lessons learnt. IMHO, some of the key points to come out of this are:

1) Cause and effect is a trap that one can easily fall into. You make a change, you see an effect, you conclude that the change resulted in the effect. However, unless you fully understand **what** it is you change and **why** the change may have made the effect and **how** such a change made the effect, you potentially fall into the trap Billy Verreynee described so nicely with the mad scientist who thought by pulling the wings off a fly, the fly goes deaf as it no longer flies away when he claps his hands.

There are a number of people who think the database "can't fly" for potentially entirely the wrong reasons and this thread has classic examples.

2) Be very very careful of folk who continually make claims but lack the ability to back those claims up with either a repeatable example showing **how and why** those claims are true or lack to the ability to describe adequately how and why those claims are true. Because, without one or both of these things, such claims run the real likelihood of being just another mad scientist not understanding what impact "pulling the wings" off Oracle may have had and who have arrived at entirely the wrong conclusion.

Again this thread has classic examples of such baseless claims and their possible dangers as it promotes an approach that may have resulted in **indirectly** fixing a problem but may have been more easily addressed by simply applying the direct fix. Or it may promote a behavior of applying the indirect fix which may not have the direct implications the next time it's applied and so fails dismally.

3) You can't fix a problem effectively unless you **understand the problem** and you **understand both the direct and indirect implications** of the applied solutions. Tuning by intuition, tuning by guesswork, tuning by **thinking** the database might be deaf without **knowing** the database is deaf will lead you down the wrong path again and again and again ...

It's all here in this thread ...

4) If it's too good to be true, it's almost certainly is too good to be true. If (say)someone claims moving to a larger blocksize results in 10x faster performance, the key question that needs to be asked and clearly understood is exactly **why**.

5) Although Oracle databases can be viewed as being a rather dry subject matter, some threads can still provide hours of amusement and hilarious reading ...

Cheers ;)

Richard Foote
http://richardfoote.wordpress.com/

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 10:02 AM in response to: Charles Hooper

Reply

Hi to all!

I decided to test all 'facts' posted in this thread and I see all that 'truth' by my own...

Charles Hooper was so kind to send me his scripts per email.

Now, I'm begging also all others, if they have usefull test scripts and enviroment suggestions, to send me it on my email address.

Thanks!

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 18, 2008 10:36 AM in response to: Faust

Reply

> Hi to all!
>
> I decided to test all 'facts' posted in this thread
> and I see all that 'truth' by my own...
>
> Charles Hooper was so kind to send me his scripts per
> email.
>
> Now, I'm begging also all others, if they have
> usefull test scripts and enviroment suggestions, to
> send me it on my email address.
>
> Thanks!

There were a couple typos in the script that I provided - a couple of the SELECT statements near the bottom of the script specified columns that do not exist (ex: PO.ID instead of PO.PURC_ORDER_ID). The 16KB test run required just short of 15 hours to complete, and it appears that the 8KB test run will require roughly the same amount of time (once it finishes) with ASSM tablespaces using auto-extent management. I will try to post my results within 12 hours.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Mark A. Williams

Posts: 1,131
Registered: 4/21/98

Re: Larger vs. Small data block
Posted: Jun 18, 2008 10:52 AM in response to: Richard Foote

Reply

Hi Richard,

Just as long as no one is "jiving us that we were voodoo" :)

Cheers,

Mark

Aman....

Posts: 3,145
From: India
Registered: 5/21/01

Re: Larger vs. Small data block
Posted: Jun 18, 2008 11:07 AM in response to: Charles Hooper

Reply

Charles,
I guess this is requested already but still,is there any where over the web where you can host the scripts of yours so that all can use it? In that way you wont need to send it by email also.
Regards
Aman....

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 18, 2008 11:23 AM in response to: Aman....

Reply

Aman, It appears that KWrite on Linux (and probably vi or any other text editor on that platform) is able to properly paste a copy of the scripts that I posted, while preserving the formatting. Notepad on Windows does a terrible job in preserving the formatting, completely losing line breaks. Wordpad on Windows does better, but loses the initial spaces on the lines. Microsoft Word and Microsoft Excel are both able to preserve the spaces and line breaks when the web page contents are copied and then pasted into those programs. A final option is to view the HTML code, and change the sequence to a CRLF combination (ASCII 13 and ASCII 10), and then also fix the < and > symbols. I do not have a suitable hosting site for the scripts. Charles Hooper IT Manager/Oracle DBA K&M Machine-Fabricating, Inc.

Aman....

Posts: 3,145
From: India
Registered: 5/21/01

Re: Larger vs. Small data block
Posted: Jun 18, 2008 11:27 AM in response to: Charles Hooper

Reply

Charles,
If you can send it me in mail,I shall try to put it over the web with your permission.

Regards
Aman....

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 11:30 AM in response to: Charles Hooper

Reply

As I already replay to you per email - here at the moment happening European Soccer Championship and that's the reason why setting proper environment in the evening hours will take a little bit... ;-)

But, for sure, if not earlier, during next weekend I will run tests regarding your scripts...

And because of systematic (and optimized) setting environments, it will be useful for me already now to know/define all test cases -> because of that my previous post.

> I do not have a suitable hosting site for the
> scripts.

If you like, I can put your scripts on my web server.

Cheers!

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 18, 2008 11:34 AM in response to: Charles Hooper

Reply

If you are using legal version of Toad, then there is a better formating option (Select all code and Shift+Ctrl+F). Oracle Sql Developer also have formating option (Select all code, right click and Format).

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 18, 2008 11:36 AM in response to: damorgan

Reply

> You can from me.
> I've no doubt you can from Brynn too.
>
> Seems to me it is about time for Greg, Graham, and a
> few others inside to belly up to the keyboard, write
> a definitive statement on the subject, and post it to
> OTN and metalink.
>
> This "controversy" leads to wasted time, wasted
> effort, and in the end makes Oracle look bad because
> it seems to have no official opinion on the matter.
>
> While you're at it please also cut down the body of
> multiple block sizes in a single database and a few
> other oft repeated myths.
>
> Thank you.

I bet Oracle will never publish to mandate db_block_size as 8k across different applications. I know, there are so many companies running their Warehouse applications with higher block size with superior performance over 8k block size. If your request to Oracle is regarding DSS applications, then you may have to wait for long time.

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 11:47 AM in response to: sp009

Reply

> If you are using legal version of Toad, then there is
> a better formating option (Select all code and
> Shift+Ctrl+F). Oracle Sql Developer also have
> formating option (Select all code, right click and
> Format).

Better try by yourself and see what will happen -> in fact nothing happen...

I didn't try with Toad but I suppose it will have same behavior as SQL Navigator.

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 18, 2008 11:53 AM in response to: damorgan

> Which brings us full circle to the statement Brynn
> made to me and that I have repeated several times in
> this thread. Oracle only tests 8K blocks. So I have
> no doubt there are many issues to be discovered by
> those that follow holistic rather than scientific
> advise with respect to block sizes. If a DBA is not
> going to use an 8K block size they'd better have
> something far more credible to go on than an opinion
> unsupported by rigorous testing.

Why do you think Oracle only tests 8K blocks? Is there any official document in Metalink says, we don't test or support 16k?

If Oracle doesn't test in 16k, then why do they publish the bug list related to db_block_size in below Metalink document?

https://metalink.oracle.com/metalink/plsql/f?p=130:14:2769096811376656232::::p14_database_id,p14_docid,p14_show_header,p14_show_help

sp009

Posts: 63
Registered: 12/3/02

Re: Larger vs. Small data block
Posted: Jun 18, 2008 12:11 PM in response to: Faust

> Better try by yourself and see what will happen -> in
> fact nothing happen...

I thought you are smart enough to identify what Toad says "I don't recognize"

OK. Comment the following lines and try again. Once formated, remove those comments

```
ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT FOREVER, LEVEL 8';
ALTER SESSION SET EVENTS '10046 TRACE NAME CONTEXT OFF'+
```

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 18, 2008 12:18 PM in response to: Faust

Reply

> > I do not have a suitable hosting site for the
> > scripts.
>
> If you like, I can put your scripts on my web
> server.

Faust,

Please feel free to put the scripts on your web server. You might add the following comments, which address typos found in the script:
* set pagesize 100000 - should have been set pagesize 50000

* AND POL.PART_ID=P.ID - should have been AND POL.PART_ID=P.PART_ID

* PO.ID=POL.PURC_ORDER_ID - should have been PO.PURC_ORDER_ID=POL.PURC_ORDER_ID

The USER_DATA tablespace data file was created with an initial size of 8GB. Under ideal conditions, the undo tablespace should have also been specified at 8GB to avoid unnecessary extension of the data file for that tablespace.

The typos in the SQL statements allow another, unexpected test - how quickly is Oracle able to reject an invalid SQL statement due to a change in the system default block size.

It is my hope that this thread will serve as a final destination for anyone wondering if a non-default block size is right for their database. There have been many great comments, summarizations, and test cases in this thread.

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

user599375

Posts: 365
Registered: 10/9/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 12:27 PM in response to: Richard Foote

Reply

> 1) Cause and effect is a trap that one can easily
> fall into. You make a change, you see an effect, you
> conclude that the change resulted in the effect.
> However, unless you fully understand **what** it
> is you change and **why** the change may have made
> the effect and **how** such a change made the
> effect, you potentially fall into the trap Billy
> Verreynee described so nicely with the mad scientist
> who thought by pulling the wings off a fly, the fly
> goes deaf as it no longer flies away when he claps
> his hands.
> There are a number of people who think the database
> "can't fly" for potentially entirely the wrong
> reasons and this thread has classic examples.

The conclusion as to 'why' may be inaccurate, but the observation is still correct - if you compare two databases of different block sizes, and one of them is faster, the fact remains that one of them is faster, regardless of your conclusion.

Oracle is complex enough that while one might be able to explain a phenomena from a single test case designed to test a particular feature, it is far more difficult to predict what the outcomes would be in a multi-user, multi-processing environment where a large range of factors, including bugs and all, come into play. It may be the blocksize, it may be something else, but as long as the benefits are tangible and repeatable, and the tests have not unearthed any other undesirable side-effects, I would be happy to take the benefits without having an exact clinical understanding of all the factors at play. If I could pinpoint it, I would of course. But I would not discard the repeatable experimental results just because I couldn't.

It may not be the wings, it may not be the ears, but it could be the loud noise from the clap which paralyzed the fly's nervous system. If my intention is to stop the fly from flying, and every time I clapped my hands and pulled off the wings, the fly stops flying, I have achieved a desired outcome, ie, the database runs faster.

user599375

Posts: 365
Registered: 10/9/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 12:49 PM in response to: sp009

Reply

> I bet Oracle will never publish to mandate
> db_block_size as 8k across
> different applications. I know, there are so many
> companies running their
> Warehouse applications with higher block size with
> superior performance
> over 8k block size. If your request to Oracle is
> regarding DSS applications,
> then you may have to wait for long time.

I agree. Expecting a 'definitive statement' statement from Oracle is rather unrealistic. Because there isn't one. While 8K may be appropriate for many, it does not by any means apply to all.

Hans Forbrich (...

Posts: 663
From: Alberta, Canada
Registered: 11/17/06

Re: Larger vs. Small data block
Posted: Jun 18, 2008 1:01 PM in response to: user599375

Reply

> > I bet Oracle will never publish to mandate
> > db_block_size as 8k across
> > different applications. I know, there are so many
> > companies running their
> > Warehouse applications with higher block size with
> > superior performance
> > over 8k block size. If your request to Oracle is
> > regarding DSS applications,
> > then you may have to wait for long time.
>
> I agree. Expecting a 'definitive statement' statement
> from Oracle is rather unrealistic.

I don't know about that. I'd say this one which states

"A block size of 8K is optimal for most systems. "

is a pretty definitive and official statement coming right from Oracle.

> Because there isn't one. While 8K may be appropriate for many, it
> does not by any means apply to all.

Very, very true. Oracle always has used the 'it depends' clause, as shown in the statement following the previous quote:

"However, OLTP systems *occasionally* use smaller block sizes and DSS systems *occasionally* use larger block sizes."

Faust

Posts: 797
From: Middle Europe
Registered: 1/1/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 1:45 PM in response to: Charles Hooper

Reply

> Please feel free to put the scripts on your web
> server.

For all who wants to try Charles OLTP test scripts and don't want to test own smartness on formatting tolls...
;-)

You can download scripts from here:

http://www.krisan.eu/oracle/scripts/hooper/oltp_test.zip

Cheers!

benprusinski

Posts: 207
From: San Diego, CA
Registered: 2/1/00

Re: Larger vs. Small data block
Posted: Jun 18, 2008 2:01 PM in response to: Jonathan Lewis

Reply

Hello Jonathan,

In your reply

"How much time were you given to find out what else he might have done that could have >caused the performance to drop ? Missing indexes, disk hot spots, constraints enabled >when they should have been kept disabled, missing statistics."

AND

"There are so many things that could have been done differently - how confident are you that nothing but the blocksize changed ?"

I checked all performance factors before making the recommendation to change the block size from 8k to 16k. Yes, I checked for missing statistics and indexes, disk I/O contention issues, etc. I gave the client the recommendations and had a short time period of several days to resolve the issue. Thus, I was confident of my decision at the time.

I do have a new question, however, for you. In your Oracle Cost Based Optimizer book you mention the issue of block size and database performance. I don't have the exact quote in front of me but will find it tonight when I get home and find it. You mention that block size can affect performance. Care to elaborate further on that?

And you said:

"Taking a different perspective - are you so sure that it was just the block size that made the difference that you're happy for xxx xxxxxxxxto atrtibute to you the claim that"Oracle consultant Ben Prusinski notes that batch jobs can see a 3x performance improvement when moved to a larger blocksize"

Yes, I am sure of this because it was the solution for the client that I worked at the time.
Now, there are exceptions and new releases of Oracle will affect how performance behaves as bugs and changes to the database engine do affect matters. Hence we have good discussion of the ASSM bug which I did not know about until it was mentioned here so thats very intereresting.

Also as I mentioned earlier in the Oracle documentation most notably the Oracle 10g Performance Tuning Guide, block size is mentioned as one issue that affects overall database performance.

Now the real crux is how valuable are test cases versus real world cases of live production systems? I see two different branches of thought on this. One group of Oracle professionals believes that test cases are worthless and that only real cases from live customer systems holds any value for proving a technical point with Oracle. The second camp such as what we have with Jonathan Lewis holds merit on test cases to find new issues with the Oracle database ie) new bugs with the CBO and so forth.

Me- I value both testing and actual results on real customer systems. After all, as a practicing DBA and consultant, I would never want to test a solution right away on a production system without FIRST testing it out on a non-critical system. So to me, both groups of thought can hold value.

Regards,
Ben Prusinski

http://oracle-magician.blogspot.com/

user599375

Posts: 365
Registered: 10/9/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 2:16 PM in response to: Hans Forbrich (...

Reply

I was quoting a bit out of context there, and the unrealistic part was referring to an expectation of a '8K for ALL systems' statement from Oracle. Which would have stopped this thread in its tracks. Then again, maybe not.

> "However, OLTP systems occasionally use smaller block sizes and DSS systems
> occasionally use larger block sizes."

Which would make it pretty 'undefinitive'. Exactly what would qualify as an OLTP, and what as a DSS system.

On which occasion should an OLTP system use a smaller block size? and DSS a larger block size?

The only way to find out is to test with the application you are going to run in production. For existing systems, I wouldn't bother changing the blocksize unless there is a problem for which other remedies don't seem to gain traction, and for new systems that are critically enough, I would certainly include the blocksize variable as one of the tests.

However, I detect that some quarters are too quick to dismiss everything that is not 8K, and that is the bit I don't quite agree with.

user599375

Posts: 365
Registered: 10/9/07

Re: Larger vs. Small data block
Posted: Jun 18, 2008 2:35 PM in response to: benprusinski

Reply

> Me- I value both testing and actual results on real
> customer systems. After all, as a practicing DBA and
> consultant, I would never want to test a solution
> right away on a production system without FIRST
> testing it out on a non-critical system. So to me,
> both groups of thought can hold value.

I agree - both have their purposes. A testcase allows you to isolate and focus on the features you want to test, in controlled environments so to speak. You also need to test against production systems (hopefully on copies of) because you want to test the thing as a whole, and not just parts of it.

jgarry

Posts: 128
From: Just outside of beautiful Vista, California
Registered: 7/20/98

Re: Larger vs. Small data block
Posted: Jun 18, 2008 2:47 PM in response to: Richard Foote

Reply

continuing lessons learnt:

6. Newer database features have more bugs or misfeatures. Sometimes the issues can be more obscure, too.

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 18, 2008 3:56 PM in response to: user599375

Reply

Hi User599375

The problem with being inaccurate with the "why" means you may potentially go down the wrong path again and again trying to resolve an Oracle issue ...

Taking the fly with no wings going deaf as an example, you might try to get the poor thing to fly by going to all the trouble of inventing a mini-hearing aid, a minute little device that you can attach to the fly, improving it's hearing capacity by 10000%.

However, you clap your hands and the fly still sits there, slowly rocking from side to side ...

If you move all your indexes into a bigger block size and performance now improves, you're suggesting who cares why it now improves, the fact performance is better is the important thing.

Wrong.

Performance may only have improved say because you're moved the indexes into a tablespace that's on much faster disks. It's got nothing directly to do with the block size, the why is entirely because of the faster disks.

Missing this point, when you next go to the considerable trouble and expense to move all indexes into a bigger block size because hey, it worked before right, you're stunned and your boss is non-too pleased that performance is now no better, maybe even worse, a lot worse.

This time you're using slower disks or using a slower portion of a disk, or disks with more contention, etc etc, and you don't get the indirect benefits you got before.

Thinking the why was moving indexes into a bigger block size, or simply not caring why it worked last time, means you've just gone down the wrong path this time ...

Yes, Oracle is potentially complex, yes, I work in multi-user, multi processor environments. That's why determining what really works and really doesn't and determining the real "why" is so vitally important.

It's what differentiates a good DBA from not such a good DBA.

It's what differentiates a good fly scientist from a bad fly scientist.

It's what differentiates a good Doctor from a bad Doctor, a Dr who knows "why" that medicine will fix that illness, rather than just giving some medicine because it appeared to have worked before when he last tried it ...

Food for thought perhaps.

Anyway, 1/2 time is over, back to Euro 2008. Go Spain !!

Cheers

Richard Foote
http://richardfoote.wordpress.com/

Jonathan Lewis

Re: Larger vs. Small data block
Posted: Jun 18, 2008 4:45 PM in response to: benprusinski

Reply

Posts: 786
From: UK
Registered: 1/23/07

[nobr]Ben,

> I do have a new question, however, for you. In your
> Oracle Cost Based Optimizer book you mention the
> issue of block size and database performance. I don't
> have the exact quote in front of me but will find it
> tonight when I get home and find it. You mention that
> block size can affect performance. Care to elaborate
> further on that?
>

This was the first one I hit when I flipped the book open:

Tuning by changing block sizes: Be very cautious with the option for using different block sizes for different objects - the feature was introduced to support transportable tablespaces, not as a tuning mechanism.

You may be able to find a few special cases where you can get a positive benefit by changing an object from one block size to another; but in general you may find that a few side effects due to the optimizer changing its arithmetic may outweigh the perceived benefits of your chosen block size.

A couple of times I've advised a client to use a 16KB block size because that should reduce the random I/O requests for a popular query from an average of two reads to just one. But every time I've done that it's a follow-on from advising them to use an IOT to reduce the I/O count from a couple of hundred per query to two.

>
> "Taking a different perspective - are you so sure
> that it was just the block size that made the
> difference that you're happy for xxx xxxxxxxxto
> atrtibute to you the claim that"Oracle consultant Ben
> Prusinski notes that batch jobs can see a 3x
> performance improvement when moved to a larger
> blocksize"
>
> Yes, I am sure of this because it was the solution
> for the client that I worked at the time.

You should only be sure that recreating the entire database was the most cost-effective thing to do for the customer - and I'd be perfectly happy to go along with that strategy, i.e: "If we can't find what the problem is within X hours, we might as well recreate the database because we know the original behaves".

My point, however, was more aimed at the thought that you had described a specific case - and it had been turned into a sweeping statement that "batch jobs can go 3x as fast if you use a larger block size". I get quite irritated when my comments are distorted that badly.

>
> Now the real crux is how valuable are test cases
> versus real world cases of live production systems?
> I see two different branches of thought on this.

The distinction between "test cases" and "real world cases" is artifical.

When Steve copied the data into a table on a 4KB block and ran the update, was that still a real world case or did it become a test case ?

When I took 30 minutes to model the scenario that Steve had described, was that a test case or a real world case ? And when I'd shown that the model behaved exactly as I had expected (i.e. no statistically significant change in performance) I asked Steve for more details so that I could refine the model. And when I guessed that he'd done a "null to not null" update, I solved the problem. In what way was my work not "real world" ?

>
> One group of Oracle professionals believes that test
> cases are worthless and that only real cases from
> live customer systems holds any value for proving a
> technical point with Oracle.

> The second camp such as
> what we have with Jonathan Lewis holds merit on test
> cases to find new issues with the Oracle database ie)
> new bugs with the CBO and so forth.
>

Don't be fooled by the xxxxxxxx propaganda - test cases are about the real world. Most of my test cases are models of real world client problems. Some of my test cases are then simple refinements of real world models, used to prove a point or demonstrate a mechanism.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge." Stephen Hawking.[/nobr]

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 18, 2008 8:22 PM in response to: Charles Hooper Reply

> I finished putting together a more comprehensive test
> script that addresses many of the issues that I had
> with my original test script. I performed a test of
> the script last night to look for typos in the
> script, but only had a couple minutes to review the
> output. Foreign keys and indexes will have a
> significant impact on performance, but it is too
> early to tell if block size makes much of a
> difference when the foreign keys are checked during
> an insert or update.

For the first test run, a database using a 16KB default block size was created, specifying the USER_DATA tablespace size at 8GB using ASSM auto. All initialization parameters were identical to those previously posted in this thread. Once the 16KB test completed, all files related to the 16KB database were removed, the computer was restarted, and then an 8KB default block size database was created using the same create scripts.

A brief summary of interesting results:

Test run time:

```
16KB   14.10 Hours
 8KB   13.62 Hours

Interesting sub-results:
INSERTING INTO PO_HEADER
500000 rows created.
16KB Elapsed: 00:00:36.14
 8KB Elapsed: 00:00:50.31

Execution Plan
----------------------------------------------------------------------------------
| Id  | Operation         | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |                |   509K|   473M|  2515   (2)| 00:00:36 |
|   1 |  TABLE ACCESS FULL| PO_HEADER_TEMP |   509K|   473M|  2515   (2)| 00:00:36 |
----------------------------------------------------------------------------------

------------
INSERTING INTO PO_LINES
12205347 rows created.
16KB Elapsed: 00:03:13.82
 8KB Elapsed: 00:03:31.40

Execution Plan
-----------------------------------------------------------------------------------------------
| Id  | Operation                        | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                 |             |   249 | 19422 |    25   (0)| 00:00:01 |
|   1 |  COUNT                           |             |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID    | PARTS       |   249 |  7221 |    23   (0)| 00:00:01 |
|   3 |    NESTED LOOPS                  |             |   249 | 19422 |    25   (0)| 00:00:01 |
|   4 |     VIEW                         |             |     1 |    49 |     2   (0)| 00:00:01 |
|   5 |      COUNT                       |             |       |       |            |          |
|*  6 |       CONNECT BY WITHOUT FILTERING|            |       |       |            |          |
|   7 |        FAST DUAL                 |             |     1 |       |     2   (0)| 00:00:01 |
|*  8 |     INDEX RANGE SCAN             | IND_PARTS_7 |   449 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------


12205347 rows created.
16KB Elapsed: 01:08:11.78
 8KB Elapsed: 01:06:01.57

Execution Plan
----------------------------------------------------------
Plan hash value: 1069489789

--------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |              |    13M|    13G| 47676   (3)| 00:11:08 |
|   1 |  TABLE ACCESS FULL| PO_LINE_TEMP |    13M|    13G| 47676   (3)| 00:11:08 |
--------------------------------------------------------------------------------

------------
UPDATE-ROLLBACK TEST
3539069 rows updated.
16KB Elapsed: 05:45:30.21
 8KB Elapsed: 05:45:07.17


Rollback complete.
16KB Elapsed: 05:32:18.11
 8KB Elapsed: 05:21:42.73

------------
INSERT-NARROW-TABLE
900000 rows created.
16KB Elapsed: 00:00:07.12
 8KB Elapsed: 00:00:06.53


900000 rows updated.
16KB Elapsed: 00:25:16.75
 8KB Elapsed: 00:24:54.43


900000 rows updated.
16KB Elapsed: 00:44:08.42
 8KB Elapsed: 00:41:22.64


900000 rows updated.
16KB Elapsed: 00:11:53.21
 8KB Elapsed: 00:00:23.78


Select of narrow table
16KB Elapsed: 00:01:45.35
 8KB Elapsed: 00:01:30.06


450000 rows deleted.
16KB Elapsed: 00:00:09.04
 8KB Elapsed: 00:00:12.29

------------
Analytical functions in the test seem to favor smaller block sizes

PART_ID                        A PRODUCT_CODE    MAX_QTY_PRD_ABC MIN_QTY_PRD_ABC DR_QTY_PRD_ABC DR_OP_VEND
------------------------------ - --------------- --------------- --------------- -------------- ----------
10000000PART                   B FG                       100000            .001          13829       1546
1000022PART                    A FG                       100000            .002           1122          7
1000209PART                    A FG                       100000            .002           1016          4
1000259PART                    C FG                       100000               0           3788      31056
...
9999998PART                    B FG                       100000            .001           2205          1
9999999PART                    B SHOP                  99026.807        3489.554            475          1

99694 rows selected.
```

```
16KB Elapsed: 00:01:24.86
 8KB Elapsed: 00:00:30.64


PART_ID                        DESCRIPTION
------------------------------ ----------------------------------------
QTY_ON_HAND RANK_PC_QTY AVG_PC_QTY MIN_PC_QTY MAX_PC_QTY   COUNT_PC RANK_CC_QTY
AVG_CC_QTY MIN_CC_QTY MAX_CC_QTY   COUNT_CC RANK_VENDOR_QTY AVG_VENDOR_QTY
MIN_VENDOR_QTY MAX_VENDOR_QTY COUNT_VENDOR
----------- ----------- ---------- ---------- ---------- ---------- -----------
10000000PART                   10000000DESCRIPTION
  99939.083        1597 62825.9166          0  99939.083      74768         309
62855.4356       .002  99939.083      13940            1043      62493.765
      3489.551      99939.083        32190

1000022PART                    1000022DESCRIPTION
  17364.487       66930 7573.22913          0  17364.487      74768       12791
7921.08607       .002  17364.487       1452               8      17364.487
     17364.487      17364.487            1
...

9999999PART                    9999999DESCRIPTION
  61566.149        3319 23998.0777   3489.551  61566.149       4983        8228
31959.2693       .001  61566.149       6012               5     32125.3248
      3490.111      61566.149            4


99694 rows selected.
16KB Elapsed: 00:03:13.93
 8KB Elapsed: 00:01:36.84


Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.
```

Hans Forbrich (...

Posts: 663
From: Alberta, Canada
Registered: 11/17/06

Re: Larger vs. Small data block
Posted: Jun 18, 2008 8:27 PM in response to: user599375 Reply

> I was quoting a bit out of context there, and the
> unrealistic part was referring to an expectation of a
> '8K for ALL systems' statement from Oracle.

If Oracle wanted to pin it at 8K for ALL systems, they would not have given us an option.

They have made two statements:

1) In most cases, 8K is an appropriate compromise;
2) In some case, which need to be evaluated, tested, benchmarked for a specific situation (an occasion) that 8K recommendation is not appropriate.

You, and Oracle concur on both those points.

> The only way to find out is to test with the application you are going to
> run in production. For existing systems, I wouldn't bother changing the
> blocksize unless there is a problem for which other remedies don't seem
> to gain traction, and for new systems that are critically enough, I would
> certainly include the blocksize variable *as one of the tests.*

I started with Oracle products in 1984. The one constant in that time has been official Oracle responses, which are invariably: "you need to verify [insert definitive statement here] in your own environment"

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 18, 2008 9:33 PM in response to: Charles Hooper Reply

> For the first test run, a database using a 16KB
> default block size was created, specifying the
> USER_DATA tablespace size at 8GB using ASSM auto.
> All initialization parameters were identical to
> those previously posted in this thread. Once the
> 16KB test completed, all files related to the 16KB
> database were removed, the computer was restarted,
> and then an 8KB default block size database was
> created using the same create scripts.

Output from the 16KB test run... one more typo identified in the output:

```
16KB ASSM Auto
SP2-0267: pagesize option 100000 out of range (0 through 50000)

  COUNT(*)
----------
11073


Session altered.

Elapsed: 00:00:00.00

Session altered.

Elapsed: 00:00:00.03

Table created.

Elapsed: 00:00:00.82

Index created.

Elapsed: 00:00:00.01

'CREATINGUMS
------------
```

```
CREATING UMS

Table created.

Elapsed: 00:00:00.06

'CREATINGVENDORS
----------------
CREATING VENDORS

Table created.

Elapsed: 00:00:00.09

Table created.

Elapsed: 00:00:00.15

'CREATINGPARTS
--------------
CREATING PARTS

Table created.

Elapsed: 00:00:00.23

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.03

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.01

Table created.

Elapsed: 00:00:00.26

'CREATINGPO_HEADER
------------------
CREATING PO_HEADER

Table created.

Elapsed: 00:00:00.12

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Table created.

Elapsed: 00:00:00.12

'CREATINGPO_LINE
----------------
CREATING PO_LINE

Table created.

Elapsed: 00:00:00.14

Index created.

Elapsed: 00:00:00.03

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.01
```

```
Table created.

Elapsed: 00:00:00.11

Table created.

Elapsed: 00:00:00.01

'INSERTINGINTOLOCATIONS'
------------------------
INSERTING INTO LOCATIONS

Session altered.

Elapsed: 00:00:00.00

2200 rows created.

Elapsed: 00:00:00.31

Execution Plan
----------------------------------------------------------
Plan hash value: 2528327348

-----------------------------------------------------------------------------------------
| Id  | Operation                         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                  |      |     1 |    72 |     4   (0)| 00:00:01 |
|   1 |  COUNT                            |      |       |       |            |          |
|   2 |   NESTED LOOPS                    |      |     1 |    72 |     4   (0)| 00:00:01 |
|   3 |    VIEW                           |      |     1 |    36 |     2   (0)| 00:00:01 |
|   4 |     COUNT                         |      |       |       |            |          |
|*  5 |      CONNECT BY WITHOUT FILTERING|      |       |       |            |          |
|   6 |       FAST DUAL                   |      |     1 |       |     2   (0)| 00:00:01 |
|*  7 |    VIEW                           |      |     1 |    36 |     2   (0)| 00:00:01 |
|   8 |     COUNT                         |      |       |       |            |          |
|*  9 |      CONNECT BY WITHOUT FILTERING|      |       |       |            |          |
|  10 |       FAST DUAL                   |      |     1 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - filter(LEVEL<=200)
   7 - filter("LOC"."RN">=MOD("WH"."RN",10)*20+1)
   9 - filter(LEVEL<=20)


Statistics
----------------------------------------------------------
        322  recursive calls
       1755  db block gets
        163  consistent gets
          1  physical reads
     861820  redo size
        679  bytes sent via SQL*Net to client
       1075  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
        203  sorts (memory)
          0  sorts (disk)
       2200  rows processed


Commit complete.

Elapsed: 00:00:00.01

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.95

8 rows created.

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=8)


Statistics
----------------------------------------------------------
         53  recursive calls
         23  db block gets
          8  consistent gets
          0  physical reads
          0  redo size
        679  bytes sent via SQL*Net to client
        685  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
          8  rows processed


Commit complete.
```

```
Elapsed: 00:00:00.00

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.03

'INSERTINGINTOVENDORS'
----------------------
INSERTING INTO VENDORS

Session altered.

Elapsed: 00:00:00.00

50000 rows created.

Elapsed: 00:00:04.48

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=50000)


Statistics
----------------------------------------------------------
       2365  recursive calls
      11944  db block gets
       2400  consistent gets
          0  physical reads
   18942216  redo size
        680  bytes sent via SQL*Net to client
       2073  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
      50000  rows processed


'ELIMINATINGDUPV'
-----------------
ELIMINATING DUP V

214 rows deleted.

Elapsed: 00:00:00.21

Execution Plan
----------------------------------------------------------
Plan hash value: 2737996044

-------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name         | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | DELETE STATEMENT          |              |   277 | 12188 |       |  1557   (1)| 00:00:22 |
|   1 |  DELETE                   | VENDORS_TEMP |       |       |       |            |          |
|*  2 |   HASH JOIN RIGHT SEMI    |              |   277 | 12188 |       |  1557   (1)| 00:00:22 |
|   3 |    VIEW                   | VW_NSO_1     |  2269 | 49918 |       |  1081   (1)| 00:00:16 |
|*  4 |     HASH JOIN             |              |  2269 | 99836 |  1520K|  1081   (1)| 00:00:16 |
|   5 |      VIEW                 |              | 45379 |   974K|       |   478   (2)| 00:00:07 |
|*  6 |       FILTER              |              |       |       |       |            |          |
|   7 |        SORT GROUP BY      |              | 45379 |   974K|       |   478   (2)| 00:00:07 |
|   8 |         TABLE ACCESS FULL| VENDORS_TEMP | 45379 |   974K|       |   475   (1)| 00:00:07 |
|   9 |      TABLE ACCESS FULL    | VENDORS_TEMP | 45379 |   974K|       |   475   (1)| 00:00:07 |
|  10 |    TABLE ACCESS FULL      | VENDORS_TEMP | 45379 |   974K|       |   475   (1)| 00:00:07 |
-------------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("VENDOR_ID"="$nso_col_1" AND "TERMS_NET_DAYS"="$nso_col_2")
   4 - access("V"."VENDOR_ID"="M"."VENDOR_ID")
       filter("V"."TERMS_NET_DAYS">"M"."TERMS_NET_DAYS")
   6 - filter(COUNT(*)>1)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         64  recursive calls
        229  db block gets
       4560  consistent gets
          0  physical reads
     141680  redo size
        680  bytes sent via SQL*Net to client
        945  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          9  sorts (memory)
          0  sorts (disk)
        214  rows processed
```

```
49786 rows created.

Elapsed: 00:00:01.23

Execution Plan
----------------------------------------------------------
Plan hash value: 448063788

----------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |              | 45379 |    43M|   476   (1)| 00:00:07 |
|   1 |  TABLE ACCESS FULL| VENDORS_TEMP | 45379 |    43M|   476   (1)| 00:00:07 |
----------------------------------------------------------------------------------


Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       2077  recursive calls
     109783  db block gets
       4478  consistent gets
          0  physical reads
   33003408  redo size
        680  bytes sent via SQL*Net to client
        584  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
      49786  rows processed


Commit complete.

Elapsed: 00:00:00.34

PL/SQL procedure successfully completed.

Elapsed: 00:00:01.14

'INSERTINGINTOPARTS'
--------------------
INSERTING INTO PARTS

Session altered.

Elapsed: 00:00:00.04

100000 rows created.

Elapsed: 00:00:14.03

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=100000)


Statistics
----------------------------------------------------------
       4005  recursive calls
      23681  db block gets
       4617  consistent gets
          0  physical reads
   38076588  redo size
        680  bytes sent via SQL*Net to client
       3187  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
     100000  rows processed


'REMOVINGDUPLICATEPARTS'
------------------------
REMOVING DUPLICATE PARTS

306 rows deleted.

Elapsed: 00:00:00.43

Execution Plan
----------------------------------------------------------
Plan hash value: 201048256

-----------------------------------------------------------------------------------------------
| Id  | Operation                 | Name       | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | DELETE STATEMENT          |            |   504 | 30240 |       |  3137   (1)| 00:00:44 |
|   1 |  DELETE                   | PARTS_TEMP |       |       |       |            |          |
|*  2 |   HASH JOIN RIGHT SEMI    |            |   504 | 30240 |       |  3137   (1)| 00:00:44 |
|   3 |    VIEW                   | VW_NSO_1   |  4136 |   121K|       |  2188   (1)| 00:00:31 |
|*  4 |     HASH JOIN             |            |  4136 |   242K|  3408K|  2188   (1)| 00:00:31 |
|   5 |      TABLE ACCESS FULL   | PARTS_TEMP | 82716 |  2423K|       |   948   (1)| 00:00:14 |
|   6 |      VIEW                |            | 82716 |  2423K|       |   953   (1)| 00:00:14 |
|*  7 |       FILTER             |            |       |       |       |            |          |
|   8 |        SORT GROUP BY     |            | 82716 |  2423K|       |   953   (1)| 00:00:14 |
|   9 |         TABLE ACCESS FULL| PARTS_TEMP | 82716 |  2423K|       |   948   (1)| 00:00:14 |
|  10 |    TABLE ACCESS FULL     | PARTS_TEMP | 82716 |  2423K|       |   948   (1)| 00:00:14 |
-----------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("PART_ID"="$nso_col_1" AND "ORDER_POINT"="$nso_col_2")
   4 - access("V"."PART_ID"="M"."PART_ID")
       filter("V"."ORDER_POINT">"M"."ORDER_POINT")
   7 - filter(COUNT(*)>1)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         64  recursive calls
        333  db block gets
       8340  consistent gets
          0  physical reads
     259216  redo size
        680  bytes sent via SQL*Net to client
        909  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          9  sorts (memory)
          0  sorts (disk)
        306  rows processed


99694 rows created.

Elapsed: 00:00:06.68

Execution Plan
----------------------------------------------------------
Plan hash value: 3663493195

--------------------------------------------------------------------------------
| Id  | Operation         | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |            | 82716 |    96M|   956   (2)| 00:00:14 |
|   1 |  TABLE ACCESS FULL| PARTS_TEMP | 82716 |    96M|   956   (2)| 00:00:14 |
--------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       7388  recursive calls
     430352  db block gets
      15397  consistent gets
          2  physical reads
  139580300  redo size
        680  bytes sent via SQL*Net to client
        580  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
      99694  rows processed


99694 rows updated.

Elapsed: 00:03:01.90

Execution Plan
----------------------------------------------------------
Plan hash value: 424025735

---------------------------------------------------------------------------------------
| Id  | Operation               | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT        |             | 87825 |  2658K|   951   (1)| 00:00:14 |
|   1 |  UPDATE                 | PARTS       |       |       |            |          |
|   2 |   TABLE ACCESS FULL     | PARTS       | 87825 |  2658K|   951   (1)| 00:00:14 |
|*  3 |   VIEW                  |             |  2200 | 68200 |     4   (0)| 00:00:01 |
|   4 |    COUNT                |             |       |       |            |          |
|   5 |     INDEX FAST FULL SCAN| SYS_C004155 |  2200 | 41800 |     4   (0)| 00:00:01 |
---------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("RN"=MOD(:B1,2000))

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         99  recursive calls
     243046  db block gets
    1430226  consistent gets
          0  physical reads
   30262784  redo size
        681  bytes sent via SQL*Net to client
        798  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
      99694  rows processed


66462 rows updated.

Elapsed: 00:00:04.73

Execution Plan
----------------------------------------------------------
Plan hash value: 2752843369

-----------------------------------------------------------------------------
| Id  | Operation           | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | UPDATE STATEMENT    |       | 58484 |   685K|   952   (1)| 00:00:14 |
|   1 |  UPDATE             | PARTS |       |       |            |          |
|   2 |   COUNT             |       |       |       |            |          |
|*  3 |    TABLE ACCESS FULL| PARTS | 58484 |   685K|   952   (1)| 00:00:14 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("PURCHASED"='Y')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
        806  recursive calls
     334790  db block gets
       3523  consistent gets
          0  physical reads
   36363728  redo size
        682  bytes sent via SQL*Net to client
        687  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
      66462  rows processed


Commit complete.

Elapsed: 00:00:00.00

PL/SQL procedure successfully completed.

Elapsed: 00:00:05.70

'INSERTINGINTOPO_HEADER'
------------------------
INSERTING INTO PO_HEADER

Session altered.

Elapsed: 00:00:00.03

500000 rows created.

Elapsed: 00:00:57.25

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=500000)


Statistics
----------------------------------------------------------
       4732  recursive calls
      60178  db block gets
      12340  consistent gets
          0  physical reads
  101922912  redo size
        682  bytes sent via SQL*Net to client
       2301  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
     500000  rows processed


500000 rows created.

Elapsed: 00:00:36.14

Execution Plan
----------------------------------------------------------
Plan hash value: 2716451106
```

```
----------------------------------------------------------------------------------
| Id  | Operation         | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |                 |   509K|   473M|  2515   (2)| 00:00:36 |
|   1 |  TABLE ACCESS FULL| PO_HEADER_TEMP |   509K|   473M|  2515   (2)| 00:00:36 |
----------------------------------------------------------------------------------


Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
      10634  recursive calls
    3126101  db block gets
      43528  consistent gets
         52  physical reads
  595290444  redo size
        682  bytes sent via SQL*Net to client
        588  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
     500000  rows processed


Commit complete.

Elapsed: 00:00:00.01

PL/SQL procedure successfully completed.

Elapsed: 00:00:06.28

'INSERTINGINTOPO_LINES'
-----------------------
INSERTING INTO PO_LINES

Session altered.

Elapsed: 00:00:00.01

12205347 rows created.

Elapsed: 00:03:13.82

Execution Plan
----------------------------------------------------------
Plan hash value: 3988977532

-----------------------------------------------------------------------------------------------
| Id  | Operation                         | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                  |             |   249 | 19422 |    25   (0)| 00:00:01 |
|   1 |  COUNT                            |             |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID     | PARTS       |   249 |  7221 |    23   (0)| 00:00:01 |
|   3 |    NESTED LOOPS                   |             |   249 | 19422 |    25   (0)| 00:00:01 |
|   4 |     VIEW                          |             |     1 |    49 |     2   (0)| 00:00:01 |
|   5 |      COUNT                        |             |       |       |            |          |
|*  6 |       CONNECT BY WITHOUT FILTERING|             |       |       |            |          |
|   7 |        FAST DUAL                  |             |     1 |       |     2   (0)| 00:00:01 |
|*  8 |     INDEX RANGE SCAN              | IND_PARTS_7 |   449 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   6 - filter(LEVEL<=500000)
   8 - access("P"."ORDER_POINT">="START_LINE" AND
              "P"."ORDER_POINT"<="START_LINE"+"LINES"-1)


Statistics
----------------------------------------------------------
      10581  recursive calls
    1073189  db block gets
    1769166  consistent gets
        108  physical reads
 1923408908  redo size
        683  bytes sent via SQL*Net to client
       1686  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
   12205347  rows processed


12205347 rows created.

Elapsed: 01:08:11.78

Execution Plan
----------------------------------------------------------
Plan hash value: 1069489789

--------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |              |    13M|    13G| 47676   (3)| 00:11:08 |
|   1 |  TABLE ACCESS FULL| PO_LINE_TEMP |    13M|    13G| 47676   (3)| 00:11:08 |
--------------------------------------------------------------------------------


Note
-----
   - dynamic sampling used for this statement
```

```
Statistics
----------------------------------------------------------
     106566  recursive calls
  124734674  db block gets
     909474  consistent gets
     166177  physical reads
SP2-0642: SQL*Plus internal error state 1075, context 1:4:4294967295
Unsafe to proceed
        683  bytes sent via SQL*Net to client
        584  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
   12205347  rows processed


Commit complete.

Elapsed: 00:00:00.00

PL/SQL procedure successfully completed.

Elapsed: 00:02:54.90

'UPDATE-ROLLBACKTEST
--------------------
UPDATE-ROLLBACK TEST

Session altered.

Elapsed: 00:00:00.04

3539069 rows updated.

Elapsed: 05:45:30.21

Execution Plan
----------------------------------------------------------
Plan hash value: 2613867723

-----------------------------------------------------------------------------------
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT  |               |  4581K|    52M| 12516   (1)| 00:02:56 |
|   1 |  UPDATE           | PO_LINE       |       |       |            |          |
|*  2 |   INDEX RANGE SCAN| IND_PO_LINE_3 |  4581K|    52M| 12516   (1)| 00:02:56 |
-----------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("PART_ID">='3000000PART' AND "PART_ID"<='6576035PART')


Statistics
----------------------------------------------------------
       2092  recursive calls
  115125604  db block gets
   92059751  consistent gets
    2705378  physical reads
 1504368784  redo size
        686  bytes sent via SQL*Net to client
        632  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          1  sorts (disk)
    3539069  rows processed


Rollback complete.

Elapsed: 05:32:18.11

'INSERT-NARROW-TABL
-------------------
INSERT-NARROW-TABLE

Session altered.

Elapsed: 00:00:00.01

900000 rows created.

Elapsed: 00:00:07.12

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=900000)


Statistics
----------------------------------------------------------
       1416  recursive calls
```

```
      10454  db block gets
       1321  consistent gets
         21  physical reads
   13762316  redo size
        687  bytes sent via SQL*Net to client
        615  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          9  sorts (memory)
          0  sorts (disk)
     900000  rows processed


Commit complete.

Elapsed: 00:00:02.64

STAT_NAME                      VALUE
------------------------- ----------
consistent gets             96712536
db block gets              256118893
table fetch by rowid        12211025
table fetch continued row          3
table scan blocks gotten      376281
table scan rows gotten      14838624

900000 rows updated.

Elapsed: 00:25:16.75

Execution Plan
----------------------------------------------------------
Plan hash value: 2650735695

-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | UPDATE STATEMENT   |        |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  UPDATE            | NARROW |       |       |            |          |
|   2 |   TABLE ACCESS FULL| NARROW |     1 |    26 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
       1798  recursive calls
  400038732  db block gets
    1308238  consistent gets
          1  physical reads
  627119556  redo size
        688  bytes sent via SQL*Net to client
        597  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     900000  rows processed


STAT_NAME                      VALUE
------------------------- ----------
consistent gets             98020916
db block gets              656157670
table fetch by rowid        12211029
table fetch continued row          3
table scan blocks gotten      378418
table scan rows gotten      16898180

900000 rows updated.

Elapsed: 00:44:08.42

Execution Plan
----------------------------------------------------------
Plan hash value: 2650735695

-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | UPDATE STATEMENT   |        |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  UPDATE            | NARROW |       |       |            |          |
|   2 |   TABLE ACCESS FULL| NARROW |     1 |    26 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
        850  recursive calls
  735436883  db block gets
    1097045  consistent gets
          0  physical reads
  312503112  redo size
        688  bytes sent via SQL*Net to client
        589  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     900000  rows processed


900000 rows updated.

Elapsed: 00:11:53.21

Execution Plan
----------------------------------------------------------
Plan hash value: 2650735695

-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | UPDATE STATEMENT   |        |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  UPDATE            | NARROW |       |       |            |          |
|   2 |   TABLE ACCESS FULL| NARROW |     1 |    26 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
        274  recursive calls
  195989191  db block gets
     259615  consistent gets
          0  physical reads
  257410288  redo size
        688  bytes sent via SQL*Net to client
        559  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     900000  rows processed


STAT_NAME                      VALUE
------------------------- ----------
consistent gets             99377682
db block gets             1587583804
table fetch by rowid        12211033
table fetch continued row          3
table scan blocks gotten      390950
table scan rows gotten      23099883

        C1         C2
---------- ----------
.615661413 .615661413
.694658313 .694658313
.809016947 .809016947
.857167259 .857167259
.933580398 .933580398
.981627168 .981627168
.994521887 .994521887
         1          1
...

900000 rows selected.

Elapsed: 00:01:45.35

Execution Plan
----------------------------------------------------------
Plan hash value: 3043013035

----------------------------------------------------------------------------
| Id  | Operation         | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |        |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| NARROW |     1 |    26 |     2   (0)| 00:00:01 |
----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          2  recursive calls
          1  db block gets
      63602  consistent gets
          0  physical reads
        176  redo size
   22139480  bytes sent via SQL*Net to client
     660370  bytes received via SQL*Net from client
      60001  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
     900000  rows processed


STAT_NAME                      VALUE
------------------------- ----------
consistent gets             99441356
db block gets             1587583845
table fetch by rowid        12211037
table fetch continued row          3
table scan blocks gotten      454544
table scan rows gotten      48775982

450000 rows deleted.

Elapsed: 00:00:09.04

Execution Plan
----------------------------------------------------------
Plan hash value: 3059185100

-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | DELETE STATEMENT   |        |     1 |    13 |     2   (0)| 00:00:01 |
|   1 |  DELETE            | NARROW |       |       |            |          |
|*  2 |   TABLE ACCESS FULL| NARROW |     1 |    13 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("C1"<0)


Statistics
----------------------------------------------------------
         91  recursive calls
     847973  db block gets
       3855  consistent gets
          0  physical reads
  201202816  redo size
```

```
        690  bytes sent via SQL*Net to client
        565  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     450000  rows processed


Commit complete.

Elapsed: 00:00:00.01

'TABLEANDINDEXSTATS'
---------------------
TABLE AND INDEX STATS

PL/SQL procedure successfully completed.

Elapsed: 00:00:01.45

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
LOCATIONS                            2200         16          81
NARROW                             447112       3838          13
PARTS                               99694       2515         362
PO_HEADER                          506757       6577         162
PO_LINE                          12173239     123536         119
UMS                                     8          5           7
VENDORS                             49786       1255         341

TABLE_NAME INDEX_NAME          BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
---------- --------------- ---------- ----------- ------------- ----------------------- ----------------------- --------------
---
LOCATIONS  IND_LOCATIONS_1          1           5           200                       1                       1
204
LOCATIONS  SYS_C004155              1           6          2200                       1                       1
1802
PARTS      IND_PARTS_1              1         137             1                     137                    2493
2493
PARTS      IND_PARTS_2              1         277             1                     277                    2493
2493
PARTS      IND_PARTS_3              1         151             8                      18                    2492
19939
PARTS      IND_PARTS_4              1         137             1                     137                    2493
2493
PARTS      IND_PARTS_5              1         128          8983                       1                       7
66462
PARTS      IND_PARTS_6              1         151             8                      18                    2492
19939
PARTS      IND_PARTS_7              1         159         99694                       1                       1
4810
PARTS      SYS_C004205              1         248         99694                       1                       1
99678
PO_HEADER  IND_PO_HEADER_1          2        1024          8983                       1                      55
500000
PO_HEADER  IND_PO_HEADER_2          2        1024          8983                       1                      55
500000
PO_HEADER  IND_PO_HEADER_3          1         724             1                     724                    6562
6562
PO_HEADER  IND_PO_HEADER_4          1         624             2                     312                    6562
13124
PO_HEADER  SYS_C004260              1         931        500000                       1                       1
101066
PO_LINE    IND_PO_LINE_1            0           0             0                       0                       0
0
PO_LINE    IND_PO_LINE_2            0           0             0                       0                       0
0
PO_LINE    IND_PO_LINE_3            2       33070          3545                       9                    3496
12394446
PO_LINE    IND_PO_LINE_4            2       31466          3545                       8                    3306
11721849
PO_LINE    SYS_C004294              2       49721      11723640                       1                       1
342405
UMS        SYS_C004159              0           1             8                       1                       1
1
VENDORS    SYS_C004165              1          97         49786                       1                       1
49775

System altered.

Elapsed: 00:00:04.90

System altered.

Elapsed: 00:00:00.12

Session altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.00
  AND POL.PART_ID=P.ID
                  *
ERROR at line 13:
ORA-00904: "P"."ID": invalid identifier


Elapsed: 00:00:00.17
  P.DESCIPTION
  *
ERROR at line 14:
ORA-00904: "P"."DESCIPTION": invalid identifier


Elapsed: 00:00:00.01

 LOCATIONS
```

```
----------
2200

Elapsed: 00:00:00.04

Execution Plan
----------------------------------------------------------
Plan hash value: 3384977531

---------------------------------------------------------------------------------
| Id  | Operation             | Name            | Rows  | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                 |     1 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE       |                 |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| IND_LOCATIONS_1 |  2200 |     4   (0)| 00:00:01 |
---------------------------------------------------------------------------------



Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         13  consistent gets
         10  physical reads
          0  redo size
        412  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


PRODUCT_CODE    PARTS_LARGE_WH
--------------- --------------
FG                       23129
INVENTORY                 3091
JANITOR                   1544
OFFICE                    1548
SHOP                      1545

Elapsed: 00:00:01.01

Execution Plan
----------------------------------------------------------
Plan hash value: 3005476749

-----------------------------------------------------------------------------------------
| Id  | Operation                 | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |             |     5 |   115 |   960   (2)| 00:00:14 |
|   1 |  SORT GROUP BY            |             |     5 |   115 |   960   (2)| 00:00:14 |
|*  2 |   HASH JOIN               |             |  5534 |   124K|   959   (2)| 00:00:14 |
|   3 |    VIEW                   |             |     1 |     9 |     5  (20)| 00:00:01 |
|*  4 |     FILTER                |             |       |       |            |          |
|   5 |      HASH GROUP BY        |             |     1 |     9 |     5  (20)| 00:00:01 |
|   6 |       INDEX FAST FULL SCAN| SYS_C004155 |  2200 | 19800 |     4   (0)| 00:00:01 |
|   7 |    TABLE ACCESS FULL      | PARTS       | 99694 |  1363K|   953   (1)| 00:00:14 |
-----------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("W"."WAREHOUSE_ID"="P"."PRIMARY_WHS_ID")
   4 - filter(COUNT(*)>160)


Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
       2538  consistent gets
       2527  physical reads
          0  redo size
        581  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
          5  rows processed


  COUNT(*)
----------
98586

Elapsed: 00:00:00.04

Execution Plan
----------------------------------------------------------
Plan hash value: 3298521242

----------------------------------------------------------------------------
| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |     1 |     7 |   956   (2)| 00:00:14 |
|   1 |  SORT AGGREGATE    |       |     1 |     7 |            |          |
|*  2 |   TABLE ACCESS FULL| PARTS | 98697 |   674K|   956   (2)| 00:00:14 |
----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("QTY_ON_HAND">1000)


Statistics
----------------------------------------------------------
```

```
          8  recursive calls
          0  db block gets
       2525  consistent gets
          0  physical reads
          0  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


  COUNT(*)
----------
5528

Elapsed: 00:00:00.43

Execution Plan
----------------------------------------------------------
Plan hash value: 3333389930

------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |     1 |     7 |   474   (1)| 00:00:07 |
|   1 |  SORT AGGREGATE    |         |     1 |     7 |            |          |
|*  2 |   TABLE ACCESS FULL| VENDORS |    49 |   343 |   474   (1)| 00:00:07 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("ZIPCODE">' 44444')


Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
       1263  consistent gets
       1256  physical reads
          0  redo size
        412  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


  COUNT(*)
----------
0

Elapsed: 00:00:00.07

Execution Plan
----------------------------------------------------------
Plan hash value: 3410092070

--------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |    21 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |             |     1 |    21 |            |          |
|*  2 |   TABLE ACCESS BY INDEX ROWID| PO_LINE     |    27 |   567 |     4   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | SYS_C004294 |    27 |       |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("POL"."PART_ID" IS NOT NULL)
   3 - access("POL"."PURC_ORDER_ID">='10000' AND "POL"."PURC_ORDER_ID"<='20000')


Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
          6  consistent gets
          5  physical reads
         80  redo size
        410  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


PART_ID                        A PRODUCT_CODE    MAX_QTY_PRD_ABC MIN_QTY_PRD_ABC DR_QTY_PRD_ABC DR_OP_VEND
------------------------------ - --------------- --------------- --------------- -------------- ----------
10000000PART                   B FG                       100000            .001          13829       1546
1000022PART                    A FG                       100000            .002           1122          7
1000209PART                    A FG                       100000            .002           1016          4
1000259PART                    C FG                       100000               0           3788      31056
...
9999998PART                    B FG                       100000            .001           2205          1
9999999PART                    B SHOP                  99026.807        3489.554            475          1

99694 rows selected.

Elapsed: 00:01:24.86

Execution Plan
----------------------------------------------------------
```

```
Plan hash value: 2057956106

--------------------------------------------------------------------------------------
| Id  | Operation            | Name  | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |       | 99694 |  3796K|       |  2851   (2)| 00:00:40 |
|   1 |  SORT ORDER BY       |       | 99694 |  3796K|    10M|  2851   (2)| 00:00:40 |
|   2 |   WINDOW SORT        |       | 99694 |  3796K|    10M|  2851   (2)| 00:00:40 |
|   3 |    WINDOW SORT       |       | 99694 |  3796K|    10M|  2851   (2)| 00:00:40 |
|   4 |     TABLE ACCESS FULL| PARTS | 99694 |  3796K|       |   956   (2)| 00:00:14 |
--------------------------------------------------------------------------------------



Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       2523  consistent gets
          0  physical reads
          0  redo size
    4109253  bytes sent via SQL*Net to client
      73487  bytes received via SQL*Net from client
       6648  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
      99694  rows processed


VENDOR_ID       VENDOR_NAME
--------------- --------------------------------------------------
1000020VEN      382030VENDOR NAME
1000186VEN      773432VENDOR NAME
1001324VEN      864606VENDOR NAME
1001380VEN      580185VENDOR NAME
...
9999995VEN      802822VENDOR NAME
9999997VEN      716062VENDOR NAME

41120 rows selected.

Elapsed: 00:00:56.68

Execution Plan
----------------------------------------------------------
Plan hash value: 1378243240

---------------------------------------------------------------------------------------------
| Id  | Operation                | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |           | 40634 |  1587K|       |   120K  (2)| 00:28:04 |
|   1 |  MERGE JOIN ANTI         |           | 40634 |  1587K|       |   120K  (2)| 00:28:04 |
|   2 |   SORT JOIN              |           | 49786 |  1409K|  3920K|   727   (1)| 00:00:11 |
|   3 |    TABLE ACCESS FULL     | VENDORS   | 49786 |  1409K|       |   475   (1)| 00:00:07 |
|*  4 |   SORT UNIQUE            |           |  9152 |    98K|       |   119K  (2)| 00:27:54 |
|   5 |    VIEW                  |           |  9152 |    98K|       |   119K  (2)| 00:27:54 |
|   6 |     HASH UNIQUE          |           |  9152 |   518K|   793M|   119K  (2)| 00:27:54 |
|*  7 |      HASH JOIN           |           |    12M|   673M|       | 67484   (2)| 00:15:45 |
|*  8 |       TABLE ACCESS FULL  | PARTS     | 19939 |   331K|       |   950   (1)| 00:00:14 |
|*  9 |       HASH JOIN          |           |    12M|   475M|    15M| 66456   (1)| 00:15:31 |
|  10 |        TABLE ACCESS FULL| PO_HEADER |   506K|  9897K|       |  2500   (2)| 00:00:35 |
|  11 |        TABLE ACCESS FULL| PO_LINE   |    12M|   243M|       | 46778   (1)| 00:10:55 |
---------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("V"."VENDOR_ID"="PV"."VENDOR_ID")
       filter("V"."VENDOR_ID"="PV"."VENDOR_ID")
   7 - access("POL"."PART_ID"="P"."PART_ID")
   8 - filter("P"."PRODUCT_CODE"='FG')
   9 - access("PO"."PURC_ORDER_ID"="POL"."PURC_ORDER_ID")


Statistics
----------------------------------------------------------
         29  recursive calls
          0  db block gets
     135804  consistent gets
     129866  physical reads
     152136  redo size
    1584681  bytes sent via SQL*Net to client
      30532  bytes received via SQL*Net from client
       2743  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
      41120  rows processed


PART_ID                        DESCRIPTION
------------------------------ ----------------------------------------
QTY_ON_HAND RANK_PC_QTY AVG_PC_QTY MIN_PC_QTY MAX_PC_QTY   COUNT_PC RANK_CC_QTY
AVG_CC_QTY MIN_CC_QTY MAX_CC_QTY   COUNT_CC RANK_VENDOR_QTY AVG_VENDOR_QTY
MIN_VENDOR_QTY MAX_VENDOR_QTY COUNT_VENDOR
----------- ----------- ---------- ---------- ---------- ---------- -----------
10000000PART                   10000000DESCRIPTION
  99939.083        1597 62825.9166          0  99939.083      74768         309
62855.4356       .002  99939.083      13940            1043      62493.765
      3489.551      99939.083        32190

1000022PART                    1000022DESCRIPTION
  17364.487       66930 7573.22913          0  17364.487      74768       12791
7921.08607       .002  17364.487       1452               8      17364.487
     17364.487      17364.487            1
...

9999999PART                    9999999DESCRIPTION
  61566.149        3319 23998.0777   3489.551  61566.149       4983        8228
31959.2693       .001  61566.149       6012               5     32125.3248
```

```
      3490.111      61566.149            4


99694 rows selected.

Elapsed: 00:03:13.93

Execution Plan
----------------------------------------------------------
Plan hash value: 3734429483

---------------------------------------------------------------------------------------------
| Id  | Operation                  | Name  | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |       | 99694 |  5841K|       |  9084   (1)| 00:02:08 |
|   1 |  SORT ORDER BY             |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|   2 |   WINDOW SORT              |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|   3 |    WINDOW SORT             |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|   4 |     WINDOW SORT            |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|   5 |      WINDOW SORT           |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|   6 |       WINDOW SORT          |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|   7 |        WINDOW SORT         |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|   8 |         WINDOW SORT        |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|   9 |          WINDOW SORT       |       | 99694 |  5841K|    15M|  9084   (1)| 00:02:08 |
|  10 |           TABLE ACCESS FULL| PARTS | 99694 |  5841K|       |   956   (2)| 00:00:14 |
---------------------------------------------------------------------------------------------



Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       2523  consistent gets
          0  physical reads
          0  redo size
   16380898  bytes sent via SQL*Net to client
      73487  bytes received via SQL*Net from client
       6648  SQL*Net roundtrips to/from client
          9  sorts (memory)
          0  sorts (disk)
      99694  rows processed


PRODUCT_CODE    UNIT_PRICE UNIT_PRICE UNIT_PRICE UNIT_PRICE UNIT_PRICE
--------------- ---------- ---------- ---------- ---------- ----------
FG                   73661      73661      73661      73661      73661
INVENTORY             9971       9971       9971       9971       9971
JANITOR               4984       4984       4984       4984       4984
OFFICE                4991       4991       4991       4991       4991
SHOP                  4984       4984       4984       4984       4984

Elapsed: 00:00:00.14

Execution Plan
----------------------------------------------------------
Plan hash value: 815198312

----------------------------------------------------------------------------
| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |     5 |    60 |   961   (2)| 00:00:14 |
|   1 |  SORT GROUP BY     |       |     5 |    60 |   961   (2)| 00:00:14 |
|   2 |   TABLE ACCESS FULL| PARTS | 99694 |  1168K|   956   (2)| 00:00:14 |
----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       2523  consistent gets
          0  physical reads
          0  redo size
        901  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
          5  rows processed

    PO.ID=POL.PURC_ORDER_ID
    *
ERROR at line 25:
ORA-00904: "PO"."ID": invalid identifier


Elapsed: 00:00:00.00

'FINISHE
--------
FINISHED
```

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

Charles Hooper

Posts: 228
From: USA
Registered: 1/27/08

Re: Larger vs. Small data block
Posted: Jun 18, 2008 9:37 PM in response to: Charles Hooper

Reply

> For the first test run, a database using a 16KB
> default block size was created, specifying the
> USER_DATA tablespace size at 8GB using ASSM auto.
> All initialization parameters were identical to
> those previously posted in this thread. Once the
> 16KB test completed, all files related to the 16KB

```
> database were removed, the computer was restarted,
> and then an 8KB default block size database was
> created using the same create scripts.

8KB test run output:

8KB ASSM Auto
SP2-0267: pagesize option 100000 out of range (0 through 50000)

  COUNT(*)
----------
11073


Session altered.

Elapsed: 00:00:00.03

Session altered.

Elapsed: 00:00:00.03

Table created.

Elapsed: 00:00:00.96

Index created.

Elapsed: 00:00:00.03

'CREATINGUMS
------------
CREATING UMS

Table created.

Elapsed: 00:00:00.04

'CREATINGVENDORS
----------------
CREATING VENDORS

Table created.

Elapsed: 00:00:00.07

Table created.

Elapsed: 00:00:00.17

'CREATINGPARTS
--------------
CREATING PARTS

Table created.

Elapsed: 00:00:00.15

Index created.

Elapsed: 00:00:00.04

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.01

Table created.

Elapsed: 00:00:00.14

'CREATINGPO_HEADER
------------------
CREATING PO_HEADER

Table created.

Elapsed: 00:00:00.06

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.01
```

```
Index created.

Elapsed: 00:00:00.00

Table created.

Elapsed: 00:00:00.09

'CREATINGPO_LINE
----------------
CREATING PO_LINE

Table created.

Elapsed: 00:00:00.09

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Index created.

Elapsed: 00:00:00.01

Index created.

Elapsed: 00:00:00.00

Table created.

Elapsed: 00:00:00.09

Table created.

Elapsed: 00:00:00.01

'INSERTINGINTOLOCATIONS'
------------------------
INSERTING INTO LOCATIONS

Session altered.

Elapsed: 00:00:00.03

2200 rows created.

Elapsed: 00:00:00.36

Execution Plan
----------------------------------------------------------
Plan hash value: 2528327348

-----------------------------------------------------------------------------------------
| Id  | Operation                         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                  |      |     1 |    72 |     4   (0)| 00:00:01 |
|   1 |  COUNT                            |      |       |       |            |          |
|   2 |   NESTED LOOPS                    |      |     1 |    72 |     4   (0)| 00:00:01 |
|   3 |    VIEW                           |      |     1 |    36 |     2   (0)| 00:00:01 |
|   4 |     COUNT                         |      |       |       |            |          |
|*  5 |      CONNECT BY WITHOUT FILTERING|      |       |       |            |          |
|   6 |       FAST DUAL                   |      |     1 |       |     2   (0)| 00:00:01 |
|*  7 |    VIEW                           |      |     1 |    36 |     2   (0)| 00:00:01 |
|   8 |     COUNT                         |      |       |       |            |          |
|*  9 |      CONNECT BY WITHOUT FILTERING|      |       |       |            |          |
|  10 |       FAST DUAL                   |      |     1 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - filter(LEVEL<=200)
   7 - filter("LOC"."RN">=MOD("WH"."RN",10)*20+1)
   9 - filter(LEVEL<=20)


Statistics
----------------------------------------------------------
        322  recursive calls
       3401  db block gets
        237  consistent gets
          1  physical reads
    1048152  redo size
        680  bytes sent via SQL*Net to client
       1075  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
        203  sorts (memory)
          0  sorts (disk)
       2200  rows processed


Commit complete.

Elapsed: 00:00:00.00

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.90

8 rows created.

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519
```

```
------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=8)


Statistics
----------------------------------------------------------
         53  recursive calls
         23  db block gets
          8  consistent gets
          0  physical reads
          0  redo size
        680  bytes sent via SQL*Net to client
        685  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
          8  rows processed


Commit complete.

Elapsed: 00:00:00.01

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.01

'INSERTINGINTOVENDORS'
----------------------
INSERTING INTO VENDORS

Session altered.

Elapsed: 00:00:00.00

50000 rows created.

Elapsed: 00:00:04.54

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=50000)


Statistics
----------------------------------------------------------
       2408  recursive calls
      23094  db block gets
       3680  consistent gets
          0  physical reads
   19768216  redo size
        680  bytes sent via SQL*Net to client
       2073  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
      50000  rows processed


'ELIMINATINGDUPV'
-----------------
ELIMINATING DUP V

214 rows deleted.

Elapsed: 00:00:00.25

Execution Plan
----------------------------------------------------------
Plan hash value: 2737996044

-------------------------------------------------------------------------------------------------
| Id  | Operation                | Name         | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | DELETE STATEMENT         |              |   611 | 26884 |       |  2227   (1)| 00:00:27 |
|   1 |  DELETE                  | VENDORS_TEMP |       |       |       |            |          |
|*  2 |   HASH JOIN RIGHT SEMI   |              |   611 | 26884 |       |  2227   (1)| 00:00:27 |
|   3 |    VIEW                  | VW_NSO_1     |  2506 | 55132 |       |  1540   (1)| 00:00:19 |
|*  4 |     HASH JOIN            |              |  2506 |   107K|  1672K|  1540   (1)| 00:00:19 |
|   5 |      VIEW                |              | 50120 |  1076K|       |   690   (2)| 00:00:09 |
|*  6 |       FILTER             |              |       |       |       |            |          |
|   7 |        SORT GROUP BY     |              | 50120 |  1076K|       |   690   (2)| 00:00:09 |
|   8 |         TABLE ACCESS FULL| VENDORS_TEMP | 50120 |  1076K|       |   686   (1)| 00:00:09 |
|   9 |      TABLE ACCESS FULL   | VENDORS_TEMP | 50120 |  1076K|       |   686   (1)| 00:00:09 |
|  10 |    TABLE ACCESS FULL     | VENDORS_TEMP | 50120 |  1076K|       |   686   (1)| 00:00:09 |
-------------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("VENDOR_ID"="$nso_col_1" AND "TERMS_NET_DAYS"="$nso_col_2")
   4 - access("V"."VENDOR_ID"="M"."VENDOR_ID")
       filter("V"."TERMS_NET_DAYS">"M"."TERMS_NET_DAYS")
   6 - filter(COUNT(*)>1)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         64  recursive calls
        243  db block gets
       8434  consistent gets
          0  physical reads
     142140  redo size
        680  bytes sent via SQL*Net to client
        945  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          9  sorts (memory)
          0  sorts (disk)
        214  rows processed


49786 rows created.

Elapsed: 00:00:01.28

Execution Plan
----------------------------------------------------------
Plan hash value: 448063788

----------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |              | 50120 |    48M|   687   (1)| 00:00:09 |
|   1 |  TABLE ACCESS FULL| VENDORS_TEMP | 50120 |    48M|   687   (1)| 00:00:09 |
----------------------------------------------------------------------------------


Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       2163  recursive calls
     125204  db block gets
       8073  consistent gets
          0  physical reads
   34300492  redo size
        680  bytes sent via SQL*Net to client
        584  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
      49786  rows processed


Commit complete.

Elapsed: 00:00:00.00

PL/SQL procedure successfully completed.

Elapsed: 00:00:01.71

'INSERTINGINTOPARTS'
--------------------
INSERTING INTO PARTS

Session altered.

Elapsed: 00:00:00.03

100000 rows created.

Elapsed: 00:00:14.62

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=100000)


Statistics
----------------------------------------------------------
       4048  recursive calls
      45669  db block gets
       7183  consistent gets
          0  physical reads
   39714000  redo size
        679  bytes sent via SQL*Net to client
       3187  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
     100000  rows processed


'REMOVINGDUPLICATEPARTS'
------------------------
REMOVING DUPLICATE PARTS

306 rows deleted.

Elapsed: 00:00:00.51

Execution Plan
----------------------------------------------------------
Plan hash value: 1732788817

-----------------------------------------------------------------------------------------------
| Id  | Operation                 | Name       | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | DELETE STATEMENT          |            |  1125 | 67500 |       |  4483   (1)| 00:00:54 |
|   1 |  DELETE                   | PARTS_TEMP |       |       |       |            |          |
|*  2 |   HASH JOIN RIGHT SEMI    |            |  1125 | 67500 |       |  4483   (1)| 00:00:54 |
|   3 |    VIEW                   | VW_NSO_1   |  4613 |   135K|       |  3113   (1)| 00:00:38 |
|*  4 |     HASH JOIN             |            |  4613 |   270K|  3784K|  3113   (1)| 00:00:38 |
|   5 |      VIEW                 |            | 92253 |  2702K|       |  1375   (1)| 00:00:17 |
|*  6 |       FILTER              |            |       |       |       |            |          |
|   7 |        SORT GROUP BY      |            | 92253 |  2702K|       |  1375   (1)| 00:00:17 |
|   8 |         TABLE ACCESS FULL| PARTS_TEMP | 92253 |  2702K|       |  1368   (1)| 00:00:17 |
|   9 |      TABLE ACCESS FULL    | PARTS_TEMP | 92253 |  2702K|       |  1368   (1)| 00:00:17 |
|  10 |    TABLE ACCESS FULL      | PARTS_TEMP | 92253 |  2702K|       |  1368   (1)| 00:00:17 |
-----------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("PART_ID"="$nso_col_1" AND "ORDER_POINT"="$nso_col_2")
   4 - access("V"."PART_ID"="M"."PART_ID")
       filter("V"."ORDER_POINT">"M"."ORDER_POINT")
   6 - filter(COUNT(*)>1)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
         64  recursive calls
        363  db block gets
      16002  consistent gets
          0  physical reads
     260076  redo size
        680  bytes sent via SQL*Net to client
        909  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          9  sorts (memory)
          0  sorts (disk)
        306  rows processed


99694 rows created.

Elapsed: 00:00:11.39

Execution Plan
----------------------------------------------------------
Plan hash value: 3663493195

--------------------------------------------------------------------------------
| Id  | Operation         | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |            | 92253 |   107M|  1379   (2)| 00:00:17 |
|   1 |  TABLE ACCESS FULL| PARTS_TEMP | 92253 |   107M|  1379   (2)| 00:00:17 |
--------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
       7680  recursive calls
     518800  db block gets
      28462  consistent gets
          2  physical reads
  148738772  redo size
        680  bytes sent via SQL*Net to client
        580  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
      99694  rows processed


99694 rows updated.

Elapsed: 00:02:58.09

Execution Plan
----------------------------------------------------------
```

```
Plan hash value: 424025735

---------------------------------------------------------------------------------------
| Id  | Operation               | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT        |             | 83068 |  2514K|  1371   (1)| 00:00:17 |
|   1 |  UPDATE                 | PARTS       |       |       |            |          |
|   2 |   TABLE ACCESS FULL     | PARTS       | 83068 |  2514K|  1371   (1)| 00:00:17 |
|*  3 |   VIEW                  |             |  2200 | 68200 |     5   (0)| 00:00:01 |
|   4 |    COUNT                |             |       |       |            |          |
|   5 |     INDEX FAST FULL SCAN| SYS_C004155 |  2200 | 41800 |     5   (0)| 00:00:01 |
---------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("RN"=MOD(:B1,2000))

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
        117  recursive calls
     236588  db block gets
    1830048  consistent gets
          0  physical reads
   29202520  redo size
        679  bytes sent via SQL*Net to client
        798  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
      99694  rows processed


66462 rows updated.

Elapsed: 00:00:04.70

Execution Plan
----------------------------------------------------------
Plan hash value: 2752843369

-----------------------------------------------------------------------------
| Id  | Operation           | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | UPDATE STATEMENT    |       | 55512 |   650K|  1373   (1)| 00:00:17 |
|   1 |  UPDATE             | PARTS |       |       |            |          |
|   2 |   COUNT             |       |       |       |            |          |
|*  3 |    TABLE ACCESS FULL| PARTS | 55512 |   650K|  1373   (1)| 00:00:17 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("PURCHASED"='Y')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
        849  recursive calls
     338348  db block gets
       6665  consistent gets
          0  physical reads
   36639124  redo size
        682  bytes sent via SQL*Net to client
        687  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
      66462  rows processed


Commit complete.

Elapsed: 00:00:00.01

PL/SQL procedure successfully completed.

Elapsed: 00:00:06.06

'INSERTINGINTOPO_HEADER'
------------------------
INSERTING INTO PO_HEADER

Session altered.

Elapsed: 00:00:00.01

500000 rows created.

Elapsed: 00:00:58.20

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=500000)


Statistics
----------------------------------------------------------
       4818  recursive calls
     117983  db block gets
      19640  consistent gets
          0  physical reads
  106248468  redo size
        683  bytes sent via SQL*Net to client
       2301  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
     500000  rows processed


500000 rows created.

Elapsed: 00:00:50.31

Execution Plan
----------------------------------------------------------
Plan hash value: 2716451106

-----------------------------------------------------------------------------------
| Id  | Operation         | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |                |   501K|   465M|  3690   (2)| 00:00:45 |
|   1 |  TABLE ACCESS FULL| PO_HEADER_TEMP |   501K|   465M|  3690   (2)| 00:00:45 |
-----------------------------------------------------------------------------------


Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
      10927  recursive calls
    3939186  db block gets
      82773  consistent gets
          0  physical reads
  610376888  redo size
        683  bytes sent via SQL*Net to client
        588  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
     500000  rows processed


Commit complete.

Elapsed: 00:00:00.01

PL/SQL procedure successfully completed.

Elapsed: 00:00:07.31

'INSERTINGINTOPO_LINES'
-----------------------
INSERTING INTO PO_LINES

Session altered.

Elapsed: 00:00:00.00

12205347 rows created.

Elapsed: 00:03:31.40

Execution Plan
----------------------------------------------------------
Plan hash value: 3988977532

------------------------------------------------------------------------------------------------
| Id  | Operation                        | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                 |             |   249 | 19422 |    47   (0)| 00:00:01 |
|   1 |  COUNT                           |             |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID    | PARTS       |   249 |  7221 |    45   (0)| 00:00:01 |
|   3 |    NESTED LOOPS                  |             |   249 | 19422 |    47   (0)| 00:00:01 |
|   4 |     VIEW                         |             |     1 |    49 |     2   (0)| 00:00:01 |
|   5 |      COUNT                       |             |       |       |            |          |
|*  6 |       CONNECT BY WITHOUT FILTERING|             |       |       |            |          |
|   7 |        FAST DUAL                 |             |     1 |       |     2   (0)| 00:00:01 |
|*  8 |     INDEX RANGE SCAN             | IND_PARTS_7 |   449 |       |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   6 - filter(LEVEL<=500000)
   8 - access("P"."ORDER_POINT">="START_LINE" AND
              "P"."ORDER_POINT"<="START_LINE"+"LINES"-1)


Statistics
----------------------------------------------------------
```

```
      10948  recursive calls
    2160840  db block gets
    2563123  consistent gets
        223  physical reads
 2005089824  redo size
        683  bytes sent via SQL*Net to client
       1686  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
   12205347  rows processed


12205347 rows created.

Elapsed: 01:06:01.57

Execution Plan
----------------------------------------------------------
Plan hash value: 1069489789

----------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------
|   0 | INSERT STATEMENT  |              |    12M|    12G| 69005   (3)| 00:13:49 |
|   1 |  TABLE ACCESS FULL| PO_LINE_TEMP |    12M|    12G| 69005   (3)| 00:13:49 |
----------------------------------------------------------------------------------


Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
      89663  recursive calls
  141754417  db block gets
    1778244  consistent gets
     283312  physical reads
SP2-0642: SQL*Plus internal error state 1075, context 1:4:4294967295
Unsafe to proceed
        683  bytes sent via SQL*Net to client
        584  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
   12205347  rows processed


Commit complete.

Elapsed: 00:00:00.01

PL/SQL procedure successfully completed.

Elapsed: 00:03:23.98

'UPDATE-ROLLBACKTEST
--------------------
UPDATE-ROLLBACK TEST

Session altered.

Elapsed: 00:00:00.00

3539069 rows updated.

Elapsed: 05:45:07.17

Execution Plan
----------------------------------------------------------
Plan hash value: 2613867723

-----------------------------------------------------------------------------------
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT  |               |  4595K|    52M| 25429   (1)| 00:05:06 |
|   1 |  UPDATE           | PO_LINE       |       |       |            |          |
|*  2 |   INDEX RANGE SCAN| IND_PO_LINE_3 |  4595K|    52M| 25429   (1)| 00:05:06 |
-----------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("PART_ID">='3000000PART' AND "PART_ID"<='6576035PART')


Statistics
----------------------------------------------------------
       2454  recursive calls
  467030361  db block gets
  383084403  consistent gets
    2847244  physical reads
 1528989796  redo size
        687  bytes sent via SQL*Net to client
        632  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          1  sorts (disk)
    3539069  rows processed


Rollback complete.

Elapsed: 05:21:42.73

'INSERT-NARROW-TABL
-------------------
```

```
INSERT-NARROW-TABLE

Session altered.

Elapsed: 00:00:00.00

900000 rows created.

Elapsed: 00:00:06.53

Execution Plan
----------------------------------------------------------
Plan hash value: 1731520519

------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | INSERT STATEMENT              |      |     1 |     2   (0)| 00:00:01 |
|   1 |  COUNT                        |      |       |            |          |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(LEVEL<=900000)


Statistics
----------------------------------------------------------
       1226  recursive calls
      16656  db block gets
       1956  consistent gets
         17  physical reads
   14130936  redo size
        689  bytes sent via SQL*Net to client
        615  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          5  sorts (memory)
          0  sorts (disk)
     900000  rows processed


Commit complete.

Elapsed: 00:00:04.93

STAT_NAME                      VALUE
------------------------- ----------
consistent gets            390140533
db block gets              627207988
table fetch by rowid        12211909
table fetch continued row         78
table scan blocks gotten      653182
table scan rows gotten      14697509

900000 rows updated.

Elapsed: 00:24:54.43

Execution Plan
----------------------------------------------------------
Plan hash value: 2650735695

-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | UPDATE STATEMENT   |        |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  UPDATE            | NARROW |       |       |            |          |
|   2 |   TABLE ACCESS FULL| NARROW |     1 |    26 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
       1587  recursive calls
  337707586  db block gets
    1350729  consistent gets
          1  physical reads
  525073528  redo size
        689  bytes sent via SQL*Net to client
        597  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     900000  rows processed


STAT_NAME                      VALUE
------------------------- ----------
consistent gets            391491412
db block gets              964915617
table fetch by rowid        12211913
table fetch continued row         78
table scan blocks gotten      656119
table scan rows gotten      16501306

900000 rows updated.

Elapsed: 00:41:22.64

Execution Plan
----------------------------------------------------------
Plan hash value: 2650735695

-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | UPDATE STATEMENT   |        |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  UPDATE            | NARROW |       |       |            |          |
|   2 |   TABLE ACCESS FULL| NARROW |     1 |    26 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
        739  recursive calls
  583033051  db block gets
    1593474  consistent gets
          0  physical reads
  409352180  redo size
        688  bytes sent via SQL*Net to client
        589  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     900000  rows processed


900000 rows updated.

Elapsed: 00:00:23.78

Execution Plan
----------------------------------------------------------
Plan hash value: 2650735695

-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | UPDATE STATEMENT   |        |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  UPDATE            | NARROW |       |       |            |          |
|   2 |   TABLE ACCESS FULL| NARROW |     1 |    26 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
        316  recursive calls
    5490094  db block gets
     191730  consistent gets
          0  physical reads
  269343620  redo size
        689  bytes sent via SQL*Net to client
        559  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     900000  rows processed


STAT_NAME                      VALUE
------------------------- ----------
consistent gets            393276734
db block gets             1553438826
table fetch by rowid        12211917
table fetch continued row         78
table scan blocks gotten      678262
table scan rows gotten      23114805

        C1         C2
---------- ----------
-0.08715570 -0.0871557
-0.19080896 -0.19080896
-0.24192186 -0.24192186
-0.34202011 -0.34202011
-0.43837111 -0.43837111
...
 .97814398  .97814398
.999847391 .999847391


900000 rows selected.

Elapsed: 00:01:30.06

Execution Plan
----------------------------------------------------------
Plan hash value: 3043013035

----------------------------------------------------------------------------
| Id  | Operation         | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |        |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| NARROW |     1 |    26 |     2   (0)| 00:00:01 |
----------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          2  recursive calls
          1  db block gets
      66188  consistent gets
          0  physical reads
        176  redo size
   22139480  bytes sent via SQL*Net to client
     660370  bytes received via SQL*Net from client
      60001  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
     900000  rows processed


STAT_NAME                      VALUE
------------------------- ----------
consistent gets            393343002
db block gets             1553438867
table fetch by rowid        12211921
table fetch continued row         78
table scan blocks gotten      744445
table scan rows gotten      38441953

450000 rows deleted.

Elapsed: 00:00:12.29

Execution Plan
----------------------------------------------------------
Plan hash value: 3059185100

-----------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | DELETE STATEMENT   |        |     1 |    13 |     2   (0)| 00:00:01 |
|   1 |  DELETE            | NARROW |       |       |            |          |
|*  2 |   TABLE ACCESS FULL| NARROW |     1 |    13 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("C1"<0)


Statistics
----------------------------------------------------------
        100  recursive calls
     863894  db block gets
       6544  consistent gets
          0  physical reads
  201784852  redo size
        691  bytes sent via SQL*Net to client
        565  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     450000  rows processed


Commit complete.

Elapsed: 00:00:00.00

'TABLEANDINDEXSTATS'
---------------------
TABLE AND INDEX STATS

PL/SQL procedure successfully completed.

Elapsed: 00:00:01.36

TABLE_NAME                       NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ ---------- ---------- -----------
LOCATIONS                            2200         28          81
NARROW                             449533       6480          13
PARTS                               99694       5032         362
PO_HEADER                          494003      13409         162
PO_LINE                          12211036     249506         119
UMS                                     8          5           7
VENDORS                             49786       2512         341

TABLE_NAME INDEX_NAME          BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
---------- --------------- ---------- ----------- ------------- ----------------------- ----------------------- --------------
---
LOCATIONS  IND_LOCATIONS_1          1          10           200                       1                       1
223
LOCATIONS  SYS_C004155              1          12          2200                       1                       1
1907
PARTS      IND_PARTS_1              1         317             1                     317                    4985
4985
PARTS      IND_PARTS_2              2         556             1                     556                    4985
4985
PARTS      IND_PARTS_3              1         305             8                      38                    4971
39774
PARTS      IND_PARTS_4              1         317             1                     317                    4985
4985
PARTS      IND_PARTS_5              1         256          8983                       1                       7
66462
PARTS      IND_PARTS_6              1         305             8                      38                    4971
39774
PARTS      IND_PARTS_7              1         318         99694                       1                       1
9416
PARTS      SYS_C004205              1         485         99694                       1                       1
99683
PO_HEADER  IND_PO_HEADER_1          2        2048          8983                       1                      55
500000
PO_HEADER  IND_PO_HEADER_2          2        2048          8983                       1                      55
500000
PO_HEADER  IND_PO_HEADER_3          2        1386             1                    1386                   13158
13158
PO_HEADER  IND_PO_HEADER_4          2        1196             2                     598                   13158
26316
PO_HEADER  SYS_C004260              2        1850        500000                       1                       1
106840
PO_LINE    IND_PO_LINE_1            0           0             0                       0                       0
0
PO_LINE    IND_PO_LINE_2            0           0             0                       0                       0
0
PO_LINE    IND_PO_LINE_3            2       63711          3602                      17                    3199
11525977
PO_LINE    IND_PO_LINE_4            2       69482          3602                      19                    3497
12599568
PO_LINE    SYS_C004294              2      102972      12418918                       1                       1
666825
UMS        SYS_C004159              0           1             8                       1                       1
1
VENDORS    SYS_C004165              1         199         49786                       1                       1
49776
```

```
System altered.

Elapsed: 00:00:07.10

System altered.

Elapsed: 00:00:00.01

Session altered.

Elapsed: 00:00:00.00

Session altered.

Elapsed: 00:00:00.00
  AND POL.PART_ID=P.ID
                  *
ERROR at line 13:
ORA-00904: "P"."ID": invalid identifier


Elapsed: 00:00:00.12
  P.DESCIPTION
  *
ERROR at line 14:
ORA-00904: "P"."DESCIPTION": invalid identifier


Elapsed: 00:00:00.01

 LOCATIONS
----------
2200

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 3384977531

---------------------------------------------------------------------------------
| Id  | Operation             | Name            | Rows  | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                 |     1 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE       |                 |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| IND_LOCATIONS_1 |  2200 |     4   (0)| 00:00:01 |
---------------------------------------------------------------------------------



Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         17  consistent gets
         14  physical reads
          0  redo size
        412  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


PRODUCT_CODE    PARTS_LARGE_WH
--------------- --------------
FG                       25474
INVENTORY                 3389
JANITOR                   1697
OFFICE                    1694
SHOP                      1696

Elapsed: 00:00:01.01

Execution Plan
----------------------------------------------------------
Plan hash value: 3005476749

-----------------------------------------------------------------------------------------
| Id  | Operation                 | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |              |     5 |   115 |  1383   (2)| 00:00:17 |
|   1 |  SORT GROUP BY            |              |     5 |   115 |  1383   (2)| 00:00:17 |
|*  2 |   HASH JOIN               |              |  5534 |   124K|  1381   (1)| 00:00:17 |
|   3 |    VIEW                   |              |     1 |     9 |     6  (17)| 00:00:01 |
|*  4 |     FILTER                |              |       |       |            |          |
|   5 |      HASH GROUP BY        |              |     1 |     9 |     6  (17)| 00:00:01 |
|   6 |       INDEX FAST FULL SCAN| SYS_C004155  |  2200 | 19800 |     5   (0)| 00:00:01 |
|   7 |    TABLE ACCESS FULL      | PARTS        | 99694 |  1363K|  1374   (1)| 00:00:17 |
-----------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("W"."WAREHOUSE_ID"="P"."PRIMARY_WHS_ID")
   4 - filter(COUNT(*)>160)


Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
       5059  consistent gets
       5048  physical reads
          0  redo size
        581  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
```

```
          2  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
          5  rows processed


  COUNT(*)
----------
98586

Elapsed: 00:00:00.04

Execution Plan
----------------------------------------------------------
Plan hash value: 3298521242

----------------------------------------------------------------------------
| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |     1 |     7 |  1378   (2)| 00:00:17 |
|   1 |  SORT AGGREGATE    |       |     1 |     7 |            |          |
|*  2 |   TABLE ACCESS FULL| PARTS | 98697 |   674K|  1378   (2)| 00:00:17 |
----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("QTY_ON_HAND">1000)


Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
       5042  consistent gets
          0  physical reads
          0  redo size
        413  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


  COUNT(*)
----------
5528

Elapsed: 00:00:00.34

Execution Plan
----------------------------------------------------------
Plan hash value: 3333389930

------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |     1 |     7 |   685   (1)| 00:00:09 |
|   1 |  SORT AGGREGATE    |         |     1 |     7 |            |          |
|*  2 |   TABLE ACCESS FULL| VENDORS |    49 |   343 |   685   (1)| 00:00:09 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("ZIPCODE">' 44444')


Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
       2520  consistent gets
       2514  physical reads
          0  redo size
        412  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


  COUNT(*)
----------
0

Elapsed: 00:00:00.06

Execution Plan
----------------------------------------------------------
Plan hash value: 3410092070

--------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |    21 |     5   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |             |     1 |    21 |            |          |
|*  2 |   TABLE ACCESS BY INDEX ROWID| PO_LINE     |    27 |   567 |     5   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | SYS_C004294 |    27 |       |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("POL"."PART_ID" IS NOT NULL)
   3 - access("POL"."PURC_ORDER_ID">='10000' AND "POL"."PURC_ORDER_ID"<='20000')
```

```
Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
          6  consistent gets
          5  physical reads
         80  redo size
        410  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed


PART_ID                        A PRODUCT_CODE    MAX_QTY_PRD_ABC MIN_QTY_PRD_ABC DR_QTY_PRD_ABC DR_OP_VEND
------------------------------ - --------------- --------------- --------------- -------------- ----------
10000000PART                   B FG                       100000            .001          13829       1546
1000022PART                    A FG                       100000            .002           1122          7
1000209PART                    A FG                       100000            .002           1016          4
1000259PART                    C FG                       100000               0           3788      31056
...
9999998PART                    B FG                       100000            .001           2205          1
9999999PART                    B SHOP                  99026.807        3489.554            475          1

99694 rows selected.

Elapsed: 00:00:30.64

Execution Plan
----------------------------------------------------------
Plan hash value: 2057956106

-------------------------------------------------------------------------------------
| Id  | Operation            | Name  | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |       | 99694 |  3796K|       |  4398   (1)| 00:00:53 |
|   1 |  SORT ORDER BY       |       | 99694 |  3796K|    10M|  4398   (1)| 00:00:53 |
|   2 |   WINDOW SORT        |       | 99694 |  3796K|    10M|  4398   (1)| 00:00:53 |
|   3 |    WINDOW SORT       |       | 99694 |  3796K|    10M|  4398   (1)| 00:00:53 |
|   4 |     TABLE ACCESS FULL| PARTS | 99694 |  3796K|       |  1377   (1)| 00:00:17 |
-------------------------------------------------------------------------------------



Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       5040  consistent gets
          0  physical reads
          0  redo size
    4109388  bytes sent via SQL*Net to client
      73487  bytes received via SQL*Net from client
       6648  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
      99694  rows processed


VENDOR_ID       VENDOR_NAME
--------------- --------------------------------------------------
1000020VEN      382030VENDOR NAME
1000186VEN      773432VENDOR NAME
1001324VEN      864606VENDOR NAME
1001380VEN      580185VENDOR NAME
...
9999995VEN      802822VENDOR NAME
9999997VEN      716062VENDOR NAME

41120 rows selected.

Elapsed: 00:00:54.95

Execution Plan
----------------------------------------------------------
Plan hash value: 1378243240

--------------------------------------------------------------------------------------------
| Id  | Operation               | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |           | 40976 |  1600K|       |   155K  (2)| 00:31:06 |
|   1 |  MERGE JOIN ANTI        |           | 40976 |  1600K|       |   155K  (2)| 00:31:06 |
|   2 |   SORT JOIN             |           | 49786 |  1409K|  3928K|  1087   (1)| 00:00:14 |
|   3 |    TABLE ACCESS FULL    | VENDORS   | 49786 |  1409K|       |   686   (1)| 00:00:09 |
|*  4 |   SORT UNIQUE           |           |  8579 | 94369 |       |   154K  (2)| 00:30:53 |
|   5 |    VIEW                 |           |  8579 | 94369 |       |   154K  (2)| 00:30:53 |
|   6 |     HASH UNIQUE         |           |  8579 |   485K|   795M|   154K  (2)| 00:30:53 |
|*  7 |      HASH JOIN          |           |    12M|   675M|       | 93284   (1)| 00:18:40 |
|*  8 |       TABLE ACCESS FULL | PARTS     | 19939 |   331K|       |  1371   (1)| 00:00:17 |
|*  9 |       HASH JOIN         |           |    12M|   477M|    15M| 91821   (1)| 00:18:22 |
|  10 |        TABLE ACCESS FULL| PO_HEADER |   494K|  9648K|       |  3672   (2)| 00:00:45 |
|  11 |        TABLE ACCESS FULL| PO_LINE   |    12M|   244M|       | 68156   (1)| 00:13:38 |
--------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("V"."VENDOR_ID"="PV"."VENDOR_ID")
       filter("V"."VENDOR_ID"="PV"."VENDOR_ID")
   7 - access("POL"."PART_ID"="P"."PART_ID")
   8 - filter("P"."PRODUCT_CODE"='FG')
   9 - access("PO"."PURC_ORDER_ID"="POL"."PURC_ORDER_ID")


Statistics
----------------------------------------------------------
         29  recursive calls
```

```
          0  db block gets
     277040  consistent gets
     262432  physical reads
     504764  redo size
    1584681  bytes sent via SQL*Net to client
      30532  bytes received via SQL*Net from client
       2743  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
      41120  rows processed


PART_ID                        DESCRIPTION
------------------------------ ----------------------------------------
QTY_ON_HAND RANK_PC_QTY AVG_PC_QTY MIN_PC_QTY MAX_PC_QTY   COUNT_PC RANK_CC_QTY
AVG_CC_QTY MIN_CC_QTY MAX_CC_QTY   COUNT_CC RANK_VENDOR_QTY AVG_VENDOR_QTY
MIN_VENDOR_QTY MAX_VENDOR_QTY COUNT_VENDOR
----------- ----------- ---------- ---------- ---------- ---------- -----------
10000000PART                   10000000DESCRIPTION
  99939.083        1597 62825.9166          0  99939.083      74768         309
62855.4356       .002  99939.083      13940            1043      62493.765
      3489.551      99939.083        32190

1000022PART                    1000022DESCRIPTION
  17364.487       66930 7573.22913          0  17364.487      74768       12791
7921.08607       .002  17364.487       1452               8      17364.487
     17364.487      17364.487            1
...

9999999PART                    9999999DESCRIPTION
  61566.149        3319 23998.0777   3489.551  61566.149       4983        8228
31959.2693       .001  61566.149       6012               5     32125.3248
      3490.111      61566.149            4


99694 rows selected.

Elapsed: 00:01:36.84

Execution Plan
----------------------------------------------------------
Plan hash value: 3734429483

--------------------------------------------------------------------------------------------
| Id  | Operation                  | Name  | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |       | 99694 |  5841K|       | 14340   (1)| 00:02:53 |
|   1 |  SORT ORDER BY             |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|   2 |   WINDOW SORT              |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|   3 |    WINDOW SORT             |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|   4 |     WINDOW SORT            |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|   5 |      WINDOW SORT           |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|   6 |       WINDOW SORT          |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|   7 |        WINDOW SORT         |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|   8 |         WINDOW SORT        |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|   9 |          WINDOW SORT       |       | 99694 |  5841K|    15M| 14340   (1)| 00:02:53 |
|  10 |           TABLE ACCESS FULL| PARTS | 99694 |  5841K|       |  1377   (1)| 00:00:17 |
--------------------------------------------------------------------------------------------



Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       5040  consistent gets
          0  physical reads
          0  redo size
   16377604  bytes sent via SQL*Net to client
      73487  bytes received via SQL*Net from client
       6648  SQL*Net roundtrips to/from client
          9  sorts (memory)
          0  sorts (disk)
      99694  rows processed


PRODUCT_CODE    UNIT_PRICE UNIT_PRICE UNIT_PRICE UNIT_PRICE UNIT_PRICE
--------------- ---------- ---------- ---------- ---------- ----------
FG                   73661      73661      73661      73661      73661
INVENTORY             9971       9971       9971       9971       9971
JANITOR               4984       4984       4984       4984       4984
OFFICE                4991       4991       4991       4991       4991
SHOP                  4984       4984       4984       4984       4984

Elapsed: 00:00:00.15

Execution Plan
----------------------------------------------------------
Plan hash value: 815198312

----------------------------------------------------------------------------
| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |     5 |    60 |  1384   (2)| 00:00:17 |
|   1 |  SORT GROUP BY     |       |     5 |    60 |  1384   (2)| 00:00:17 |
|   2 |   TABLE ACCESS FULL| PARTS | 99694 |  1168K|  1377   (1)| 00:00:17 |
----------------------------------------------------------------------------



Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       5040  consistent gets
          0  physical reads
          0  redo size
        901  bytes sent via SQL*Net to client
        381  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          1  sorts (memory)
```

```
         0  sorts (disk)
         5  rows processed

    PO.ID=POL.PURC_ORDER_ID
    *
ERROR at line 25:
ORA-00904: "PO"."ID": invalid identifier


Elapsed: 00:00:00.00

'FINISHE
--------
FINISHED
```

Charles Hooper
IT Manager/Oracle DBA
K&M Machine-Fabricating, Inc.

user599375

Posts: 365
Registered: 10/9/07

Re: Larger vs. Small data block
Posted: Jun 19, 2008 12:59 AM in response to: Richard Foote

Reply

> The problem with being inaccurate with the "why"
> means you may potentially go down the wrong path
> again and again trying to resolve an Oracle issue

Not necessarily. If it achieves the results you want, and the results are repeatable, you, by definition, are on the right path. The conclusion may still be wrong, but the desired result is not.

Achieving perfect accuracy is great, and certainly should be strived for. But, how realistic is that? Why do you think this thread has gone to this size if accuracy were easily obtainable, and the method of achieving that accuracy consistently repeatable for all?

The fact is, even when you think you know the answer, its precise role in a busy multi-user, multi-processing environment is going to be less cut-and-dry. And you could have wasted a lot of time in attempting to arrive at a perfect answer when a simple experiment (with its less than perfect conclusions) would have pointed you in the right direction early on. Don't get me wrong - I still value accuracy, but I don't think achieving 100% is the best value for money.

> Taking the fly with no wings going deaf as an
> example, you might try to get the poor thing to fly
> by going to all the trouble of inventing a
> mini-hearing aid, a minute little device that you can
> attach to the fly, improving it's hearing capacity by
> 10000%.
> However, you clap your hands and the fly still sits
> there, slowly rocking from side to side ...

No trouble at all, because I don't want the fly to umm, fly. Desired results achieved. If I wanted the fly to fly, I would have taken a different approach like not pulling the wings off in the first place.

>
> If you move all your indexes into a bigger block size
> and performance now improves, you're suggesting who
> cares why it now improves, the fact performance is
> better is the important thing.
> Wrong.

Never said that. Hope the other parts of my reply makes this clear.

> Performance may only have improved say because you're
> moved the indexes into a tablespace that's on much
> faster disks. It's got nothing directly to do with
> the block size, the why is entirely because of the
> faster disks.

Irrelevant argument. That would apply if the person doesn't know the difference between slower and faster disks, and ignored it completely as a variable, in which case the point of this whole thread is moot.

> Thinking the why was moving indexes into a bigger
> block size, or simply not caring why it worked last
> time, means you've just gone down the wrong path this
> time ...

That's over-simplification. You deal with the variables you know and can control, but also accept that there are some variables you don't know, but the effects of which you can deduce from your repeatable experiments. Maybe some don't know the difference between faster and slower disks, but I'm sure the majority do.

> Yes, Oracle is potentially complex, yes, I work in
> multi-user, multi processor environments. That's why
> determining what really works and really doesn't and
> determining the real "why" is so vitally important.

Commendable aim. I prefer the 80-20 rule.You can expend 80% of the effort in determining the 'why', but recognise that the remaining 20% may not be cost-effective for the employer.

SeanMacGC

Posts: 7
Registered: 10/30/06

Re: Larger vs. Small data block
Posted: Jun 19, 2008 6:34 AM in response to: user599375

Reply

>Commendable aim. I prefer the 80-20 rule.You can expend 80% of the effort in determining the 'why', but recognise that the remaining 20% may not be cost-effective for the employer.

Surely with the improving performance analytics of Oracle with each release that 20% will steadily diminish as a practicable threshold of effort?

benprusinski

Posts: 207
From: San Diego, CA
Registered: 2/1/00

Re: Larger vs. Small data block
Posted: Jun 19, 2008 1:40 PM in response to: Jonathan Lewis

Reply

Jonathan we actually agree on something!

You should only be sure that recreating the entire database was the most cost-effective thing to do for the customer - and I'd *be perfectly happy to go along with that strategy, i.e: "If we can't find what the problem is within X hours, we might as well recreate the database because we know the original behaves".*

But aside from our differences in opinion and so forth, we can at least agree that

1) Testing and evidence is important
2) Bugs do exist in Oracle code and always will
3) Oracle Documentation is never perfect
4) There is possible bug in ASSM

I think that the next time I find something wrong in the documentation or Metalink, I will follow your recommendation and file a documentation bug with Oracle to get it fixed or addressed. I must admit that aside from the heated debate this has been an interesting thread!

Regards,
Ben Prusinski
http://oracle-magician.blogspot.com/

David_Aldridge

Posts: 97
Registered: 4/22/08

Re: Larger vs. Small data block
Posted: Jun 19, 2008 1:47 PM in response to: user619401

Reply

(inspired by comments here: http://www.oraclealchemist.com/oracle/hey-guys-does-size-matter)

The issue of the potential bug appears to be as perfect an illustration as one could wish for of the importance of understanding the root cause for a problem. Who would want to move their application to a new database with a new block size, or take on the increased complexity of a multi-blocksize configuration, when they can potentially address the same problem by modifying PCTFREE on a couple of tables and maybe performing a "move" to avoid future migration problems?

Jonathan Lewis

Posts: 786
From: UK
Registered: 1/23/07

Re: Larger vs. Small data block
Posted: Jun 19, 2008 2:21 PM in response to: benprusinski

Reply

```
>
> But aside from our differences in opinion and so
> forth, we can at least agree that
>
> 1) Testing and evidence is important
> 2) Bugs do exist in Oracle code and always will
> 3) Oracle Documentation is never perfect
> 4) There is possible bug in ASSM
>
```

Agreed on all four. And I'd say that any differences we've expressed are more a matter of degree and timing rather than principle.

Regards
Jonathan Lewis
http://jonathanlewis.wordpress.com
http://www.jlcomp.demon.co.uk

Richard Foote

Posts: 279
From: Canberra Australia
Registered: 12/13/99

Re: Larger vs. Small data block
Posted: Jun 19, 2008 4:27 PM in response to: David_Aldridge

Reply

```
> The issue of the potential bug appears to be as
> perfect an illustration as one could wish for of the
> importance of understanding the root cause for a
> problem. Who would want to move their application to
> a new database with a new block size, or take on the
> increased complexity of a multi-blocksize
> configuration, when they can potentially address the
> same problem by modifying PCTFREE on a couple of
> tables and maybe performing a "move" to avoid future
> migration problems?
```

Hi David

Precisely !!

Can you imagine implementing the use of a different sized block tablespace/database when perhaps say changing the db_file_multiblock_read_count would have achieved the same results.

However, if one has been advocating the use of multi sized blocks for years, if one may have perhaps implemented such so-called "solutions" and charging for such at client sites, if one has perhaps written and is selling such advice in books, perhaps one is placed in a position of choosing just which facts meets ones theories and disregard the rest.

Cheers

Richard Foote
http://richardfoote.wordpress.com/