The Oracle Magician

Sept, 2004 *OracleMagician.com* *Volume III, number 3*

From the Editor

Welcome to the new issue of our online magazine, The *Oracle Magician*! This occasional newsletter focuses on various "tricks of the trade" in the Oracle world--from DBAs, architects, developers, designers, and report writers.

Thank you to the many notes and comments from readers of my book, *The Art & Science of Oracle Performance Tuning*. Your notes are most appreciated!

In this issue we present some cautions about using Oracle's parallel query feature. I was surprised to learn of some odd consequences of using this feature.

Also in this issue is a brief summary about redo activity with materialized views. I think you will find the results surprising.

As always, we accept ideas or articles from reads that have interesting performance ideas. .

Please send all ideas to Editor@OracleMagician.com

Chris Lawson

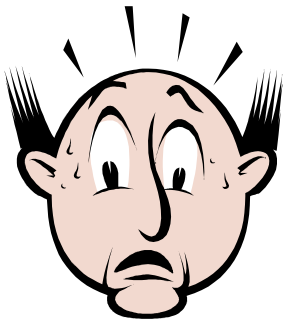
Editor

Surprising Side Effects of Oracle Parallelism

By Chris Lawson

The Issue

While working on a performance issue on a large data warehouse, I came across an interesting issue related to storage of database tables. We discovered that using Oracle parallelism could have very surprising (and undesirable) ramifications. This article reveals our findings.



Typical Table Create

To illustrate the issue, let's first create a typical table, without any parallelism. Note that our tablespace has a default size of 1m for initial and next extents.

```
Create Table Chris_Tempx
Nologging
As Select *
From Source_Table
Where Rownum < 1000
```

This table is created very quickly. Let's check the number of extents using a simple query to a data dictionary view. We expect to find very few extents.

Continued on page 2

Table of Contents

From the Editor ..... Page 1
Surprising Side Effects of Oracle Parallelism ..... Page 1
Logging with Materialized Views ..... Page 4
Updates and Notes ..... Page 4

Continued from page 1

Oracle did NOT add 8 more extents! The Insert command seems to be safe!

Select Extents, Bytes
From Dba_Segments
Where Segment_Name =
'CHRIS_TEMPX'

```
  EXTENTS     BYTES
--------- ----------
        1    1048576
```

The above results are as expected. We have a small table in just one extent.

Now let's repeat the above test, but this time, try it with parallel 8

Create Table Chris_Tempx
Nologging Parallel 8
As Select * From Source_Table
Where Rownum < 1000

```
   EXTENTS      BYTES
---------- ----------
         8    8388608
```

Now, our small table, with exactly the same number of rows as before, consumes eight times as much storage.

Of course, with such a small value for initial and next extents, this isn't really much of a problem. In fact, we only discovered this because the extents for our tablespace were sized at 256 MB. We noticed that full table scans were taking far too long for a certain small table.

Inserts

What about inserts? Do they also generate extra extents when you use parallel processing?

To test this, we first rebuilt our sample table without specifying parallel. The brought our table back to a single extent. Next we inserted rows while specifying parallel 8:

Insert into Chris_Tempx
(Select /*+Parallel (R 8) */
*** From Source_Table R**
Where rownum < 1000)

Oracle did NOT add 8 more extents! The Insert command seems to be safe!

Just to be sure, we repeated the *Insert*, but this time putting the *Parallel* hint after the *Insert* keyword:

Insert /*+Parallel (C 8) */
Into Chris_Tempx C
(Select * from Source_Table
Where rownum < 1000)

The above case also worked as expected. There were no extra extents.

We didn't give up yet; we tried one more thing. We changed the default degree for the table to be degree 8:

Alter table Chris_Tempx
Parallel 8;

Then, we repeated the two insert

Continued on page 3

Continued from page 2

tests above. As before, Oracle did *not* add 8 more extents in either case.

Merging

Since many of our jobs involved use of the *Merge* command, we needed to determine if this command would cause extra extents to be allocated each time we used it.

Caution with Merge

When using the Merge command, remember that the matches cannot produce more than one row; otherwise, you will get an error message,

ORA-30926:
Unable to get a stable set of rows in the source tables

Getting a Good Baseline

To get a good baseline (and make sure the merge would not normally create 8 extents) we first used the non-parallel method. (At this point, the default degree for the table was set back to degree 1.)

```
Merge into Chris_Tempx C
Using Chris_Tempy R
On (C.User_Key = - R.User_Key)
When Matched then Update Set
C.JOB_NO= R.JOB_NO
When Not Matched Then
Insert (USER_KEY, JOB_NO)
Values (R.User_Key, R.JOB_NO)
```

Nothing will match above (due to the minus sign), so the above code will cause about 1000 inserts.

As expected, the table remains at a single extent after the code above completes. Even if you set the table default to parallel degree 8, and repeat the merge, the table remains at one extent.

Repeating the Merge

Let's repeat our merge, but this time let's set everything parallel:

```
Merge
/*+PARALLEL (C 8) Parallel (R 8) */
Into chris_tempx C
Using Chris_Tempy R
On (C.User_Key = - R.User_Key)
When Matched then Update Set
C.JOB_NO= R.JOB_NO
When Not Matched Then
Insert (USER_KEY, JOB_NO) Values
(R.User_Key, R.JOB_NO)
```

As before, running the above command did not cause extra extents to be allocated.

There's another way to set parallelism, and that's the way that got us into trouble.

Session Level Parallelism

It is possible to change parallelism for your entire session. You can specify it for DML or DDL activity, or for queries. Here is the syntax:

```
ALTER SESSION
FORCE PARALLEL
DML|DDL|QUERY;
```

For our test, we ran all 3 options. That is, we altered our session to specify parallel processing for all of the above cases.

We repeated our last *Merge* command with parallel everything. Would this cause extra extents to be allocated?

The answer was yes--with parallelism set at the session level, Oracle does allocate an extra 8 extents! We confirmed that an extra 8 extents are added *every single time* the merge is run! Similarly, we discovered that *Inserts* also add 8 extents if the session has been altered to force parallel 8.

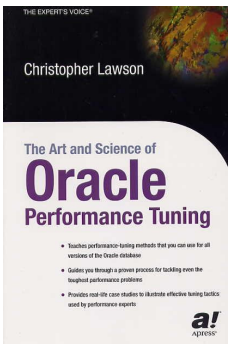
Summary

Our experiments showed that forcing parallelism at the session level causes extra extents to be used for each slave, whether the *Insert* or *Merge* command is used.

The extra extents may be a problem only when extent sizes are way too high. In addition, if you don't alter your session to force parallelism, this may never be a problem for you.

Updates and Notes

If you find these tuning issues interesting, I discuss performance tuning in greater detail in my book, *The Art and Science of Oracle Performance Tuning*. It is available at most large bookstores, or online at Amazon.com.



Logging Materialized Views

Here's something to keep in mind when using materialized views. Surprisingly, when you create a materialized view with the Nologging clause, logging doesn't really stop! Instead, you need to explicitly change the MV using the *Alter Table* command.

Here's how you can do this. First, create your materialized view *Build Deferred*. Second, Use the *Alter Table* command to turn logging off. Finally, command the refresh.

Here's a simple example:

```
Create Materialized View MV1
Build Deferred
As Select * From Source_Tables;

Alter Table MV1 Nologging;

Exec Dbms_Mview.Refresh( 'MV1', 'C' );
```

You can try it yourself to confirm that you are not logging changes. Here's an easy way to find the redo for your session:

```
Select A.Name, B.Value
From V$Statname A, V$Mystat B
Where A.Statistic# = B.Statistic#
And A.Name = 'redo size'
```

Thanks to Tom Kyte for his script on quantifying redo.

The Oracle Magician

Editor: Chris Lawson
OracleMagician.com