Why is Flashback Slow?

by *Chris Lawson*

Flashback - a Misleading Name

Despite the appealing name, "flashback," a flashback query can run very slowly. On a large production system, a flashback query going back a few hours can easily take 10 hours. What--how can that be?

This happens because Oracle must reconstruct an object as it existed at a certain time. This is the same idea of read-consistency. This reconstruction happens one block at a time, going backwards in time, undoing each transaction.[1]

Starting the Undo

There are other issues with a flashback query that make the process run even slower. Of course, Oracle does indeed save the undo information--we can certainly find it, and a flashback query really does work. Here's the problem: The structure of undo segments is heavily biased towards quickly *saving* transaction information--not quickly *reversing* transactions.

Before Oracle can reconstruct an object, it has to *identify* what needs to be undone. One would think this is a trivial step, but not so. This can be very time-consuming--especially when the database has undergone lots of recent transactions.

Transaction Table

In each undo segment header there lies a critical structure known as the *transaction table.* It's not a "table" as we normally think of it. Maybe a "list" would have been a better name. The transaction table identifies the undo information held in that undo segment. For example, any given entry points to where to find the actual undo block.

That sounds excellent, but the entire transaction table only has information for 34 transactions. (Yes, that sounds small to me, too.) Each entry is called a transaction *slot*. As more transactions are housed in a given undo segment, transaction slots, being so few, are very often *overwritten*. The information is not lost, of course, but to find it, several extra steps will be required. On a very busy system, it could

---

[1] Thanks especially to Jonathan Lewis, who has tested this interesting feature in *Oracle Core Essential Internals for DBAs and Developers,* Apress, 2011.

take thousands of extra reads just to find where to start. (That's why I observed that Oracle seems very biased towards going forward with the undo, not actually applying it.)

Remember--all this effort is before Oracle even starts the "real work" of rebuilding the object of interest to the time desired. Of course, that final step will add even more time. The point is, the delay of determining where to start can be vastly more than the work required to actually *do* the reconstructing of the object.

Troubleshooting

Troubleshooting a flashback query delay is not so easy. On a busy system, I have seen flashback queries require *millions* of extra reads to flashback a small table with only 20,000 transactions that needed to be undone. If you query the active session history for the session of interest, it will show that it is performing sequential reads from an undo tablespace. One could easily be fooled into thinking (as I did) that there must have been a huge number of transactions on the table of interest. We know better now--the reads were actually Oracle synthesizing the undo information in the transaction table, not actually applying it to the object of interest.

Recycling Undo?

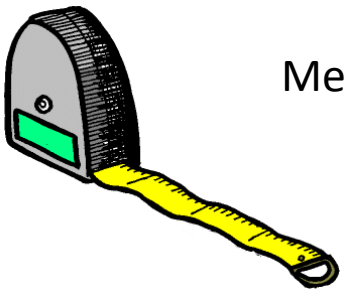
When a transaction table slot is reused, what happens to the valuable information that used to be kept in that slot? Well, there's one logical place for it to go--somewhere in undo-land. In fact, Oracle stores the old slot information right at the beginning of the new undo block that used that slot. In this way, the information is linked together. Therefore, when we perform a flashback query, we can discover what the transaction table looked like at some prior state.

Undoing the Undo?

Hey, what a minute--all this almost sounds like "undoing the undo!" You're right, and Oracle calls it, "Transaction Table Rollback." You can also get a summary in the AWR report, in the *Instance Activity* section:

Instance Activity Stats

Statistic	Total	per Second	per Trans
transaction tables consistent read rollbacks	1,869	0.10	0.00
transaction tables consistent reads - undo records applied	9,577,664	531.95	3.91



Measuring Undo of the Undo?

You can also quantify this event in real time, to get a feel for how often this is happening. On a busy system, it is likely happening all the time. Let's see how we do this on a busy RAC system. Here is one way to see this occurring in the current connected sessions. This would be helpful to know if someone is doing a flashback query that seems to be running far longer than expected.

In this script, I look for large values of transaction table undo, and list the sessions. I also ignore the background processes (that's why I exclude programs like 'oracle'):

```
Col Module Format A22
Col Sid Format 99999
Col Program Format A20
Col Inst Format 9999
Col Trundo Format 9999999

Select One.Inst_Id INST, One.Sid, Substr(Program,1,20) PROG,
Substr(Module,1,20) Mod, Value TRUNDO
From Gv$Sesstat One, V$Statname Two, Gv$Session Three
Where One.Statistic# = Two.Statistic#
And One.Inst_Id = Three.Inst_Id
And One.Sid = Three.Sid
And Name =
'transaction tables consistent reads - undo records applied'
And Program Not Like 'Oracle@%'
And Value > 90000
Order By Value;

INST    SID PROG                 MOD                  TRUNDO
---- ------ -------------------- -------------------- --------
   7   1978 xtsora@cisxx01 (TNS  xtsora@risint01 (TNS    157315
   4    408 xtsora@cisxx01 (TNS  xtsora@risint01 (TNS    178481
```

We can see above that there were two active sessions that appear to be impacted.

What can I do?

The essence of the problem is having to repeatedly reconstruct the contents of the "slots" in the transaction table. If there were fewer re-uses of the slots, then there would be less work required. Oracle support has suggested keeping more undo segments online--and therefore more slots available.

This is accomplished by setting the underscore parameter, ***_rollback_segment_count***. The idea is, to override the automatic undo process, and force more undo segments to stay online. It seems like the number of reused "slots" should go down commensurate with the extra undo segments that are kept online. So, if we keep 4x as many undo segments online, I would expect to see approximately a 4x reduction in transaction table rollbacks. That's the theory, anyway, but I haven't confirmed that yet.

How does the Story End?

As of this date, we have not yet tried the secret rollback segment parameter. We are wondering about adverse effects, and intend to test the parameter with all of our batch jobs.

We can't help wondering why the database thinks it's a good idea to take undo segments offline in the first place (and what will happen when we block that.) Perhaps the caching effect is better when there are fewer undo segments involved?

We haven't been able to get a clear answer to that question. I am anxious to see what happens.

Chris Lawson is an Oracle *Ace* and performance specialist in the San Francisco Bay Area. He is the author of *The Art & Science of Oracle Performance Tuning*, as well as *Snappy Interviews: 100 Questions to Ask Oracle DBAs*. When he's not solving performance problems, Chris is an avid hiker and geocacher, where he is known as *Bassocantor*. Chris can be reached at: Chris@OracleMagician.com