Oracle Diagnostics

Hemant K Chitale

Hemant K Chitale

- whoami ?
- Oracle 5 to Oracle 10gR2 ： DOS, Xenix,8 flavours of Unix, Linux, Windows
- Financial Services, Govt/Not-for-Profit, ERP, Custom
- Production Support, Consulting, Development
- A DBA, not a Developer
- *My Oracle Blog* *http://hemantoracledba.blogspot.com*

Latches and Enqueues

- Latches are points of concurrency ; Enqueues are points of serialisation.
- Three sessions may be waiting on a Latch – any of the three may get the Latch before the other two.
- If sessions are waiting on an Enqueue, access is serialised – the first session to start waiting gets the Enqueue first.

Latches

- Imagine a Wishing Well with place for only 1 person to stand in front of it
- Four persons approach the Wishing Well almost “simultaneously”
- Mr A gets to the well first and stands before it making his wish before dropping a coin
- The other three persons (B, C and D) loiter around. B wanders off to look at garden, C sits down on a bench and D stands close to A. REMEMBER : There is no convention that people must queue at a Wishing Well.
- A takes a long time making his wish and D get’s bored (his process has hit the “spin count” limit for the number of No-Op spins it can do on the CPU) and wanders away. He relinquishes his place.
- When A finishes and walks away, C rushes forward and get’s to the well first.
- In the meantime, E and F have also walked in on the scene.
- After C, D gets to the well.
- Before B can get back, E is next in making a wish at the well

Latches

- There you have it !
- There is no “sequencing” or “ordering”.
- A process may spin on the CPU waiting for a Latch but if it doesn’t get the latch, it relinquishes the CPU and goes to “sleep”.
- It may not be the first process to get the Latch when it is released – some other process happens to be lucky enough to get the Latch.
- Latches protect access to Memory Resources.
- A datablock for a table is loaded into the SGA as a Buffer. It may contain 20 different rows.
- If 5 users need access to 5 different rows in that Buffer, they may get access to modify the Buffer in any order. Oracle really doesn’t care which of the 5 users modify the Buffer first --- as long as they are not modifying the same rows.
- Latches on Buffers are in the Headers that are on the Cache Buffers Chain lists – in-memory linked lists.

Latches Diagnosis - 1

- Identify Latches with V$LATCHNAME, V$LATCH, V$LATCH_PARENT, V$LATCH_CHILDREN
- Latch Holders are in V$LATCHHOLDER
- Latches may be “willing-to-wait” and “no-wait” latches

Latches Diagnosis - 2

- Statistics are presented “gets”, “misses” and “sleeps” – 1,2,3… where the session sleeps once or twice or thrice or four or more times, because it was unable to “get” the latch each time it awoke from it’s sleep
- Most commonly known latches :
  - library cache ; shared pool
  - cache buffers chains

Identifying Latches (listing from 11.2) :

```
SQL> select count(*) from v$latchname;
  COUNT(*)
----------
       535
SQL> select  count(*) from v$latch;
  COUNT(*)
----------
       535
SQL> select count(*) from v$latch_parent;
  COUNT(*)
----------
       535
SQL> select count(*) from v$latch_children;
  COUNT(*)
----------
      2773
SQL>
SQL> desc v$latchholder
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PID                                                NUMBER
 SID                                                NUMBER
 LADDR                                              RAW(4)
 NAME                                               VARCHAR2(64)
 GETS                                               NUMBER
SQL>
```

Library Cache Latch usage :
This latch protects SQL statements, object definitions etc. Oracle internally determines the number of latches available (as a prime number).
Adding new SQL statements and Objects require the Latch. One latch would be protecting multiple SQLs.
Note : If you have DDLs modifying object definitions, there would be waits on the Library Cache Latches protecting those objects.

Library Cache Pin Latch usage :
When a statement is re-executed (it has to be pinned to ensure that it is not modified !

Shared Pool Latch :
This latch protects the allocation of memory in the Shared Pool. Multiple child latches are created.
Frequent Hard Parsing of SQLs would cause frequent access to the Shared Pool Latch.

Cache Buffers Chains Latch :
Protects Memory Buffers for Database Blocks. Multiple (Child) Latches are present, each Latch protecting multiple buffers (blocks).
Blocks are loaded into memory based on a hash of the Database Block Address.
A Linked List of the Buffer Headers is maintained so that a Block can be found quickly in the Buffer Cache.
Look for CBC Latches with very high SLEEPs – indicating very frequent retries.

Don't run this query – it will take a long time on a busy instance / large database.

-- query from How To Identify a Hot Block Within The Database Buffer Cache. [ID 163424.1]

```
select /*+ RULE */
       e.owner ||'.'|| e.segment_name  segment_name,
       e.extent_id  extent#, x.dbablk - e.block_id + 1  block#,
       x.tch, l.child#
     from
       sys.v$latch_children  l,
       sys.x$bh  x,
       sys.dba_extents  e
     where
       x.hladdr  = '&ADDR' and
       e.file_id = x.file# and
       x.hladdr = l.addr and
       x.dbablk between e.block_id and e.block_id + e.blocks -1
     order by x.tch desc ;
```

Typically Hot Blocks can be Index Root / Branch Blocks when an Index is frequently used in a Nested Loop. So, look at the Sessions and SQLs and Execution Plans.

Cloned Buffers :
Buffers are “cloned” when different sessions require different versions for Read Consistency.

Assume User “A” started a query at time t0
Assume User “C” modified Buffer 123 (representing a specific table block) at time t5 with an UPDATE statement
If User “A”s session comes to the the same table block, the DBA (DataBlockAddress) requires it read Buffer 123. It finds, from the Buffer Header SCN, that the Block has been modified. The ITL entry identifies the Undo Segment and slot. From the Undo information, the session now as to recreate the “pre-change” image of the block. So, it “clones” the Buffer as another Buffer in memory and applies the Undo information to it – because it actually has to modify the block, which it cannot [and should not] do against the “dirty” Buffer 123 last updated by User “C”.

The same buffer can have multiple clones. Also, a session might have to keep “rolling back” a block through multiple updates to get to it’s desired state (SCN).

So : A Buffer Cache of 800MB doesn’t necessarily mean that you have 800MB of data, some of it could be multiple copies of the same database block, as of different points in time.

Cache Buffers LRU Chain Latch :
The LRU Chain is a list of buffers. Oracle maintains multiple lists.
A process that needs to load a block into memory “walks” the chain to identify a buffer that can be “used” (e.g. an empty or clean buffer). If it cannot find a buffer, it marks a list of dirty buffers for DBWR to flush to disk. When the buffers are flushed to disk, they are marked “clean” and can be reused. This latch is required whenever changes are to be made.

Waits on these would mean that DBWR isn’t fast enough.

Learnings :

1. Frequent Hard Parses strain the Shared Pool and Library Cache Latches.

2. Hot Blocks cause waits on particular Cache Buffer Chains Latches. The Hot Blocks need to be identified based the SQLs of the Sessions waiting on CBC Latches. Fixes could be SQL tuning, rebuilding table/index, reverse key indexes(avoid them !!).

3. Move to faster I/O, use Async I/O , add DB_WRITER_PROCESSES only as a last resort.

4. Latch Issues point to Concurrency issues.

Enqueues

- Imagine a Bank with one Teller
- All Customers who walk in have to form an orderly Queue at the Teller counter
- When the Teller is processing Mr A's withdrawal request, B, C and D form a queue behind him.
- The Teller make take 1minute to handle Mr A's request at the end of which B can step forward.
- The Teller make take 2.5minutes to handle Mr B's request. C and D have to wait patiently in the queue.

- If 5 users are attempting to update the same row in the Buffer, they *have* to access it in chronological (SCN) order.
- The second user cannot update the row until and unless the first user has issued a COMMIT or ROLLBACK.
- Enqueues on rows are in the ITL entries within the DataBlock (now a Buffer in memory).
- If 5 users are updating different rows in the Block (Buffer), each has a separate ITL entry in the Block, the 5 row locks are "concurrent".

Enqueues

- The most famous Enqueues are Row Locks (“TX”) and DML Locks (“TM”)
- DML Locks prevent changes to the structure of the object being locked.
- Thus a Transaction has Row Locks on Rows being updated and DML Locks on the Table to prevent any ALTER commands from modifying the structure (e.g. add/drop columns) of the Table
- There are 64 different types of Enqueues (11.2) [62 in 10.2] See Appendix D of the Reference Guide

The Controlfile (CF) Enqueue is taken when LGWR or ARCH is updating the Controlfile or when CKPT is updating checkpoint information. NOLOGGING operations also take CF enqueues !
I've seen database instances crash when the CF enqueue is held for too long by one background process. RMAN uses a snapshot controlfile to avoid CF enqeuues.

The Undo Segment (US) Enqueue is taken when adding undo segments, taking them online or offline. When a "storm" of activity occurs, you may find US enqueue waits.

The Space Transaction (ST) Enqueue is for allocation / deallocation of extents.

The Object Chekpoint (KO) Enqueue is for Oracle to checkpoint an object/segment – e.g. for a TRUNCATE or DROP or before Parallel Query

The Sequence Number (SQ) Enqueue is for incrementing Sequences. Setting appropriate CACHE sizes is important.

The Job Queue (JQ) Enqueue is for Jobs.

Cross Instance (CI) Enqueue doesn't appear only in RAC ! You will see requests and waits in non-RAC as well.

Enqueue Waits in an *idle* instance :

```
SQL> desc v$enqueue_stat
 Name                           Null?    Type
 ------------------------------ -------- --------------------
 INST_ID                                 NUMBER
 EQ_TYPE                                 VARCHAR2(2)
 TOTAL_REQ#                              NUMBER
 TOTAL_WAIT#                             NUMBER
 SUCC_REQ#                               NUMBER
 FAILED_REQ#                             NUMBER
 CUM_WAIT_TIME                           NUMBER

SQL>  select eq_type, total_req#, total_wait#
SQL>  from v$enqueue_stat where total_wait# > 0 order by 1;

EQ TOTAL_REQ# TOTAL_WAIT#
-- ---------- -----------
CF       2895           1  -- controlfile
JS      64043          15  -- not documented
KO         18           1  -- multiple object checkpoint
PR        358           3  -- process startup
PV         53           3  -- not documented
TH        361           1  -- not documented

6 rows selected.

SQL>
```

After running :

```
SQL> create table abc as select * from dba_objects
  2  union all select * from dba_objects
  3  union all select * from dba_objects;

Table created.

EQ TOTAL_REQ# TOTAL_WAIT#
-- ---------- ----------
KO         27          2
```

On running :

```
SQL> select /*+ PARALLEL (a 4) */ count(*) from abc a;

  COUNT(*)
----------
    229848

SQL>

EQ TOTAL_REQ# TOTAL_WAIT#
-- ---------- ----------
KO         36          3
```

Learnings :

1. There are many different points of serialisation other than Row Locks.

2. Watch out for critical enqueues – CF, TS, SQ, KO.