Creative Oracle 12c Redo Maneuvers

Craig A. Shallahamer
OraPub, Inc.
craig@orapub.com

Connect with Craig and OraPub.

Email: craig@orapub.com

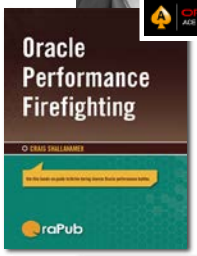
Twitter: @CShallahamer

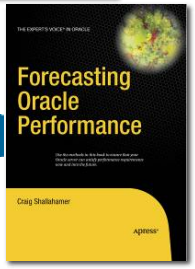
OraPub.Com: Everything starts here!

LinkedIn: Connect and network with Craig and the OraPub Group.

Relevant background info...

- Studied economics, mathematics and computer science at Cal Polytechnic State University San Luis Obispo, California, USA.
- Started working with Oracle technology in 1989 as a Forms 2.3 developer on Oracle version 5.
- Soon after started performance firefighting daily.
- Co-founded both Oracle's Core Technology and System Performance Groups.
- Left Oracle to start OraPub, Inc. in 1998.
- Authored 24+ technical papers and worked in 31 countries.
- Author two books: Oracle Performance Firefighting and Forecasting Oracle Performance.
- Teaches performance analysis around the wor
- Oracle ACE Director.
- Blogs performance research: A Wider View





We exist because DBAs want to improve but can't get the resources they need.

Fast-paced one hour seminars segmented into 8 to 10 digestible modules

Focusing exclusively on Oracle systems performance analysis

Resources

- Performance Blog
- Free Tools
- Free Presentations
- Free Papers
- Books
- Consulting
- Training

Creative Redo Maneuvers

- Redo architecture
- Using multiple log writers
- Commit write facility
- Changing log writer operating system priority

New 12c redo latches?

```
SQL> l
  1  select name from v$latchname where upper(name) like '%REDO%'
  2  minus
  3  select name from latches_11g where upper(name) like '%REDO%'
  4* order by name
SQL> /

NAME
----------------------------------------------------------------
instance redo on-disk SCN
redo transport task latch
```

Log Writer Workers: New wait events

```
SQL> SQL> l
  1  select name
  2  from   v$event_name
  3  where  wait_class != 'Idle'
  4*   and  lower(name) like 'lgwr%'
SQL> /

NAME
----------------------------------------
LGWR wait on LNS
LGWR-LNS wait on channel
LGWR wait for redo copy
LGWR any worker group
LGWR all worker groups
LGWR worker group ordering
LGWR intra group sync
LGWR intra group IO completion
...

11 rows selected.
```

The “worker” and “group” events are new in 12c.

How many redo allocation latches?

```
SQL> select count(*) from v$latch_children
     where name = 'redo allocation';

  COUNT(*)
----------
        54

SQL> show parameter processes

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
aq_tm_processes                      integer     1
db_writer_processes                  integer     1
gcs_server_processes                 integer     0
global_txn_processes                 integer     1
job_queue_processes                  integer     1000
log_archive_max_processes            integer     4
processes                            integer     300

SQL> select count(*) from v$latch_children
     where name = 'In memory undo latch';

  COUNT(*)
----------
        52
```

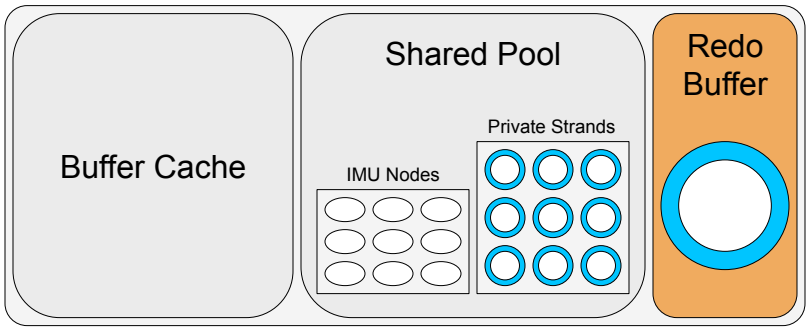
In 11gR2 with processes set to 300, there are 37 child latches, not 54...

Always 2 fewer IMU latches than redo alloc latches.

IMU and private strands reside in the shared pool.

```
SQL> l
  1  select name, pool, bytes/1024/1024 MB
  2  from v$sgastat
  3  where lower(name) like '%redo%'
  4       or lower(name) like '%kti%'
  5       or lower(name) like '%strand%'
  6* order by 1,2
SQL> /

NAME                       POOL                 MB
-------------------------- ------------ ----------
KKTIN                      shared pool  .017539978
KTI freelists              shared pool  .000083923
KTI latch structure        shared pool   .00793457
KTI latches                shared pool  .001586914
KTI pool states            shared pool  .000053406
KTI-UNDO                   shared pool  3.51263428
LGWR per strand PIC array  shared pool  .009887695
Redo Transmit Queue Descr  shared pool  .000305176
array for shared redo blo  shared pool  .000823975
arrays for shared redo bl  shared pool  .000022888
ktilm hash latches         shared pool  .015563965
ktilm hash table           shared pool     .015625
private strands            shared pool  6.75390625
redo allocation latch(es)  shared pool  .008239746

14 rows selected.
```

There are many “latch” related memory areas in the shared pool.



Multiple redo allocation latches allow multiple server processes to simultaneously copy redo into their private strand, also speeding redo into the redo buffer.

In 10gR2 “batch redo” was introduced. On **commit** (not just a change) both transaction IMU and redo (in a private strand) are copied into the redo general/ central buffer and then the LGWR flushes the general/central redo buffer to the current on-line redo log.

When does the LGWR write?

- The LGWR writes when:
  - buffer fills to min(1/3 full, 1MB)
  - 3 second timeout (pre-10gR2)
  - Commit issued. Tests always show an immediate LGWR write, unless *commit write* facility is used.
  - DBWR tells LGWR to write when it wants to write to a dbf file. DBF file change history must be recorded in an on-line redo log or how the DBF file arrived at its current state will be a mystery.

The log file sync wait event.

- Log file synchronization is about preparing for a commit and waiting for it to complete.
- For a commit to complete:
  - Both the server process and the LGWR are deeply involved.
  - They require CPU and IO resources to complete their tasks.
  - Their work must be synchronized to ensure everything is completed in the proper order.
  - When the server process is not consuming CPU but waiting on something else, probably the LGWR, it posts the `log file sync` event.
- Because significant CPU and IO resources are required to complete a commit, there can be **either** an operating system CPU or IO bottleneck.

Watching the 12c LGWR sleep.

```
$ ps -eaf | grep prod35 | grep lgwr | grep -v grep
oracle   14232     1  0 10:53 ?        00:00:00 ora_lgwr_prod35
[oracle@sixcore opload]$ strace -rp 14232
Process 14232 attached - interrupt to quit
...
     0.000040 times(NULL)               = 480558852
     0.000039 clock_gettime(CLOCK_MONOTONIC, {511651, 787854953}) = 0
     0.000040 clock_gettime(CLOCK_MONOTONIC, {511651, 787894843}) = 0
     0.000041 semtimedop(2523139, {{16, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource t
     3.000421 clock_gettime(CLOCK_MONOTONIC, {511654, 788383370}) = 0
     0.000212 gettimeofday({1382550833, 974954}, NULL) = 0
     0.000146 clock_gettime(CLOCK_MONOTONIC, {511654, 788760644}) = 0
     0.000151 getrusage(0x1 /* RUSAGE_??? */, {ru_utime={0, 5999}, ru_stime={0, 179
...
     0.000041 times(NULL)               = 480559452
     0.000039 clock_gettime(CLOCK_MONOTONIC, {511657, 793383347}) = 0
     0.000042 clock_gettime(CLOCK_MONOTONIC, {511657, 793424252}) = 0
     0.000041 semtimedop(2523139, {{16, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource t
     3.000962 clock_gettime(CLOCK_MONOTONIC, {511660, 794495395}) = 0
     0.000241 gettimeofday({1382550839, 981036}, NULL) = 0
     0.000225 clock_gettime(CLOCK_MONOTONIC, {511660, 794932663}) = 0
     0.000134 getrusage(0x1 /* RUSAGE_??? */, {ru_utime={0, 5999}, ru_stime={0, 189
...
```

o12.1

Watching all 12c LGWR processes.

It's a good thing to have multiple LGWR processes!

```
[oracle@sixcore ~]$ ps -eaf | grep lg | grep -v grep
oracle    1007     1  0 Sep03 ?        00:00:02 ora_lgwr_prod25
oracle    1011     1  0 Sep03 ?        00:00:02 ora_lg00_prod25
oracle    1015     1  0 Sep03 ?        00:00:00 ora_lg01_prod25
```

I issued the below commands. The following three pages show the OS trace of the LGWR, LG00, and LG01 process.

```
exec dbms_lock.sleep(4);
update bogus set owner = 'bogus' where owner = 'MG';
exec dbms_lock.sleep(4);
update bogus set owner = 'bogus' where owner = 'OE2';
exec dbms_lock.sleep(4);
update bogus set owner = 'bogus' where owner = 'SYSTEM';
exec dbms_lock.sleep(4);
commit;
```

What are the conditions to trigger the LGWR to write?
When does my redo get written to an on-line redo log?

o12.1 LGWR

```
[oracle@sixcore trace]$ ps -eaf | grep ora_lgwr_prod25 | grep -v grep
oracle   28197     1  0 11:14 ?        00:00:00 ora_lgwr_prod25
[oracle@sixcore trace]$ strace -rp 28197
Process 28197 attached - interrupt to quit
. . .
     0.000054 semtimedop(4292610, {{16, -1, 0}}, 1, {2, 960000000}) = 0
     2.663111 times(NULL)               = 996209263
     0.000363 times(NULL)               = 996209263
     0.000080 semctl(4292610, 18, SETVAL, 0x7fff00000001) = 0
     0.000066 getrusage(0x1 /* RUSAGE_??? */, {ru_utime={0, 77988}, ru_stime={0, 60990}, ...}) = 0
     0.000085 getrusage(0x1 /* RUSAGE_??? */, {ru_utime={0, 77988}, ru_stime={0, 60990}, ...}) = 0
     0.000045 times(NULL)               = 996209263
     0.000102 semtimedop(4292610, {{16, -1, 0}}, 1, {0, 300000000}) = 0
     0.016311 times(NULL)               = 996209265
     0.000044 semtimedop(4292610, {{16, -1, 0}}, 1, {0, 280000000}) = -1 EAGAIN (Resource temporar
     0.281252 times(NULL)               = 996209293
     0.000054 times(NULL)               = 996209293
     0.000034 times(NULL)               = 996209293
     0.000025 semtimedop(4292610, {{16, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.000946 getrusage(0x1 /* RUSAGE_??? */, {ru_utime={0, 77988}, ru_stime={0, 60990}, ...}) = 0
     0.000240 getrusage(0x1 /* RUSAGE_??? */, {ru_utime={0, 77988}, ru_stime={0, 60990}, ...}) = 0
     0.000235 times(NULL)               = 996209593
     0.000044 times(NULL)               = 996209593
     0.000039 times(NULL)               = 996209593
     0.000029 times(NULL)               = 996209593
     0.000026 semtimedop(4292610, {{16, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.000310 getrusage(0x1 /* RUSAGE_??? */, {ru_utime={0, 77988}, ru_stime={0, 60990}, ...}) = 0
     0.000316 getrusage(0x1 /* RUSAGE_??? */, {ru_utime={0, 77988}, ru_stime={0, 60990}, ...}) = 0
     0.000169 times(NULL)               = 996209893
     0.000081 times(NULL)               = 996209893
     0.000028 times(NULL)               = 996209893
     0.000032 times(NULL)               = 996209893
     0.000028 semtimedop(4292610, {{16, -1, 0}}, 1, {3, 0}^C <unfinished ...>
Process 28197 detached
```

o12.1 LG00

```
[oracle@sixcore ~]$ ps -eaf | grep ora_lg00_prod25 | grep -v grep
oracle   28201     1  0 11:14 ?        00:00:00 ora_lg00_prod25
[oracle@sixcore ~]$ strace -rp 28201
Process 28201 attached - interrupt to quit
     0.000000 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = 0
     0.659749 pwrite(256, "\1\"\0\0\r\227\0\0v\0\0\0\20\200\251\nx\0\0\0\6-\0\0Af6\0\1\0\0\23"...,
     0.012459 times(NULL)               = 996205995
     0.000114 semctl(4292610, 16, SETVAL, 0x7fff00000001) = 0
     0.000045 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.000710 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.001085 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.000948 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.000914 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.001066 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.001020 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.000950 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.001047 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.000895 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = 0
 --> 2.982625 pwrite(256, "\1\"\0\0\16\227\0\0v\0\0\0\20\200\30\203\334_\0\0\r\377\0\0Pf6\0\1\0\0\
     0.036613 times(NULL)               = 996208997
     0.000274 semctl(4292610, 16, SETVAL, 0x7fff00000001) = 0
     0.000122 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = 0
 --> 2.663741 pwrite(256, "\1\"\0\0v\230\0\0v\0\0\0\20\200\24\302\260\0\0\0\0051\0\0Sf6\0\1\0\0\1"
     0.016192 times(NULL)               = 996209265
     0.000061 semctl(4292610, 34, SETVAL, 0x1) = 0
     0.000131 semctl(4292610, 16, SETVAL, 0x7fff00000001) = 0
     0.000044 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.001337 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.001004 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unav
     3.001000 semtimedop(4292610, {{18, -1, 0}}, 1, {3, 0}^C <unfinished ...>
Process 28201 detached
```

o12.1 LG01

```
[oracle@sixcore ~]$ ps -eaf | grep ora_lg01_prod25 | grep -v grep
oracle   28203     1  0 11:14 ?        00:00:00 ora_lg01_prod25
[oracle@sixcore ~]$ strace -rp 28203
Process 28203 attached - interrupt to quit
     0.000000 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.001145 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.000990 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.000917 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.001085 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.001043 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.000946 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.001020 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.001034 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.000968 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.001012 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.000987 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.001066 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (
     3.000945 semtimedop(4292610, {{19, -1, 0}}, 1, {3, 0}^C <unfinished
Process 28203 detached
```

What just happened?

- Multiple updates were made over a 12+ second period and there was **no** redo log writing occurring.
- The “flush redo every 3 seconds rule” clearly does not mean “all redo.” It appears private strand redo does not get written. This is the same in 11g.
- When a commit occurred, a write system call was executed and redo was written to an on-line redo log. Good news!
- In the previous example, LG00 did the writing, not the LGWR parent.

Let's do this again,

but follow a timeline.

Search blog.orapub.com for "redo timeline"
http://blog.orapub.com/20150112/Do-Oracle-Database-LGWRs-Always-Sleep-For-The-Full-Three-Seconds.html

```
exec dbms_lock.sleep(12);
update bogus
  set object_name='$object1'
  where object_id=$object1;
exec dbms_lock.sleep(10);
commit;
```

sleep **10** -> update -> sleep 10 -> commit -> wait

Note: the server process strace
starts 2 seconds after this snippet begins.

```
. . .
11:17:55.381347 svpr semtimedop(229378, {{11, -1, 0}}, 1, {3, 0}) =
11:17:56.366391 lg01 semtimedop(229
11:17:56.386301 lgwr semtimedop(229
. . .
11:18:05.369421 lg01 semtimedop(229378, {{19, -1, 0}}, 1, {3, 0}) =
11:18:05.392490 lgwr semtimedop(229378, {{16, -1, 0}}, 1, {3, 0}) =
11:18:05.416490 svpr read(10, "\0\0\1k\6\0\0\0\0\0\0\2\0\0"..., 8208) = 363
11:18:05.423523 svpr read(10, "\0\0
11:18:05.424563 svpr semtimedop(229
11:18:06.498555 lg00 semtimedop(229
11:18:08.370442 lg01 semtimedop(229
11:18:08.394473 lgwr semtimedop(229
11:18:08.425440 svpr semtimedop(229
11:18:14.372355 lg01 semtimedop(229378, {{19, -1, 0}}, 1, {3, 0}) =
11:18:14.399056 lgwr semtimedop(229378, {{16, -1, 0}}, 1, {3, 0}) =
11:18:15.429473 svpr read(10, "\0\0\0017\6\0\\1\0\0\0\0\0\2\0\0"..., 8208) = 311
11:18:15.430573 svpr semctl(229378,
11:18:15.431027 lgwr semctl(229378,
11:18:15.431241 lg00 pwrite(256, "\
11:18:15.431309 lgwr semtimedop(229
11:18:15.431617 svpr semtimedop(229
11:18:15.446695 lg00 semctl(229378,
11:18:15.446986 lg00 semctl(229378, 16, SETVAL, 0x7fff00000001) = 0
11:18:15.447223 lg00 semtimedop(229378, {{18, -1, 0}}, 1, {3, 0}) =
11:18:15.447420 lgwr semtimedop(229378, {{16, -1, 0}}, 1, {1, 950000000}) =
11:18:15.447897 svpr read(10, "\0\0\0\r\6\0\0\0\0\0\3\t\30", 8208) = 13
```

svrprc sleeps for 10 seconds

svrprc updates and waits for msg

svrprc sleeps for 10 seconds

svrprc commits and waits (LFS)

LGWRs write

svrprc waits for msg

Looking at a more active 12C LGWR situation.

- Many processes are involved.
- The system is very active.
- Questions:
  - Are both LGWR children active?
  - Are both children just as active?
  - Is the LGWR parent active?

Parent LGWR

```
$ ps -eaf|grep lg
oracle   37090     1  0 12:32 ?        00:00:00 ora_lgwr_prod35
oracle   37094     1  0 12:32 ?        00:00:00 ora_lg00_prod35
oracle   37096     1  0 12:32 ?        00:00:00 ora_lg01_prod35
oracle   40107 31242  0 15:56 pts/0    00:00:00 grep lg

$ strace -rp 37090
Process 37090 attached - interrupt to quit
     0.000027 gettimeofday({1400281182, 241662}, NULL) = 0
     0.000029 times(NULL)                = 2253324483
     0.000030 clock_gettime(CLOCK_MONOTONIC, {18242003, 109327107}) = 0
     0.000028 clock_gettime(CLOCK_MONOTONIC, {18242003, 109355937}) = 0
     0.000031 semtimedop(7307267, {{16, -1, 0}}, 1, {2, 130000000}) = 0
     0.101744 clock_gettime(CLOCK_MONOTONIC, {18242003, 211131220}) = 0
     0.000031 gettimeofday({1400281182, 343555}, NULL) = 0
     0.000029 gettimeofday({1400281182, 343585}, NULL) = 0
     0.000027 times(NULL)                = 2253324493
     0.000027 times(NULL)                = 2253324493
```

o12.1

LG00

```
$ strace -rp 37094
Process 37094 attached - interrupt to quit
     0.000000 clock_gettime(CLOCK_MONOTONIC, {18241951, 746042039}) = 0
     0.000055 gettimeofday({1400281130, 878476}, NULL) = 0
     0.000047 clock_gettime(CLOCK_MONOTONIC, {18241951, 746129707}) = 0
     0.000032 semtimedop(7307267, {{18, -1, 0}}, 1, {3, 0}) = 0
     0.054282 clock_gettime(CLOCK_MONOTONIC, {18241951, 800443453}) = 0
     0.000031 gettimeofday({1400281130, 932868}, NULL) = 0
     0.000031 gettimeofday({1400281130, 932899}, NULL) = 0
     0.000033 clock_gettime(CLOCK_MONOTONIC, {18241951, 800536977}) = 0
     0.000028 clock_gettime(CLOCK_MONOTONIC, {18241951, 800563872}) = 0
     0.000027 clock_gettime(CLOCK_MONOTONIC, {18241951, 800594344}) = 0
     0.000030 clock_gettime(CLOCK_MONOTONIC, {18241951, 800620822}) = 0
     0.000026 pwrite(256, "\1\"\0\0\32536\5\1\0\0\5"..., 1024, 10715648) = 1024
     0.007834 clock_gettime(CLOCK_MONOTONIC, {18241951, 808483126}) = 0
. . .
     0.000031 gettimeofday({1400281130, 941118}, NULL) = 0
     0.000030 semctl(7307267, 16, SETVAL, 0x7fff00000001) = 0
     0.000030 clock_gettime(CLOCK_MONOTONIC, {18241951, 808783134}) = 0
     0.000027 clock_gettime(CLOCK_MONOTONIC, {18241951, 808809667}) = 0
     0.000029 semtimedop(7307267, {{18, -1, 0}}, 1, {3, 0}) = 0
     0.035167 clock_gettime(CLOCK_MONOTONIC, {18241951, 844011841}) = 0
     0.000034 gettimeofday({1400281130, 976435}, NULL) = 0
. . .
     0.000028 clock_gettime(CLOCK_MONOTONIC, {18241951, 844186855}) = 0
     0.000031 pwrite(256, "\1\"\0\0\303Q\0\0\3120\0\0"..., 1536, 10716672) = 1536
     0.005887 clock_gettime(CLOCK_MONOTONIC, {18241951, 850103698}) = 0
. . .
```

o12.1

LG01

```
$ strace -rp 37096
...
     0.000035 semtimedop(7307267, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Reso...
     3.000966 clock_gettime(CLOCK_MONOTONIC, {18241874, 839087672}) = 0
     0.000036 semtimedop(7307267, {{19, -1, 0}}, 1, {3, 0}) = -1 EAGAIN (Reso...
     3.000951 clock_gettime(CLOCK_MONOTONIC, {18241877, 840073952}) = 0
     0.000036 semtimedop(7307267, {{19, -1, 0}}, 1, {3, 0}) = 0
     0.459974 clock_gettime(CLOCK_MONOTONIC, {18241878, 300083902}) = 0
...
     0.000029 clock_gettime(CLOCK_MONOTONIC, {18241878, 300259088}) = 0
     0.000026 pwrite(256, "\1\"\0\0JM\0\0\312\77\\0\1"..., 512, 10130432) = 512
     0.039775 clock_gettime(CLOCK_MONOTONIC, {18241878, 340061420}) = 0
     0.000030 clock_gettime(CLOCK_MONOTONIC, {18241878, 340090049}) = 0
     0.000027 times(NULL)               = 2253312008
...
     0.000027 clock_gettime(CLOCK_MONOTONIC, {18241879, 29236928}) = 0
     0.000025 clock_gettime(CLOCK_MONOTONIC, {18241879, 29262238}) = 0
     0.000026 pwrite(256, "\1\"\0\0UM\0\0\315\1\0\0"..., 1536, 10136064) = 1536
     0.026714 clock_gettime(CLOCK_MONOTONIC, {18241879, 56009284}) = 0
     0.000035 clock_gettime(CLOCK_MONOTONIC, {18241879, 56038239}) = 0
...
```

o12.1

But then a little later... LGWR writing!?

```
$ ps -eaf|grep lg
oracle   37090     1  0 12:32 ?        00:00:01 ora_lgwr_prod35
oracle   37094     1  0 12:32 ?        00:00:01 ora_lg00_prod35
oracle   37096     1  0 12:32 ?        00:00:00 ora_lg01_prod35
oracle   40484 31242  0 16:23 pts/0    00:00:00 grep lg

$ strace -cp 37090
Process 37090 attached - interrupt to quit
^CProcess 37090 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 89.35    0.008998          28       325           pwrite
  9.93    0.001000           3       333        19 semtimedop
  0.40    0.000040           0      3697           clock_gettime
  0.28    0.000028           0      4408           gettimeofday
  0.05    0.000005           0      1366           times
  0.00    0.000000           0         1           read
  0.00    0.000000           0         1           open
  0.00    0.000000           0         1           close
  0.00    0.000000           0        62           getrusage
------ ----------- ----------- --------- --------- ----------------
100.00    0.010071                 10194        19 total
```

o12.1

```
$ strace -cp 37094
Process 37094 attached - interrupt to quit
^CProcess 37094 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  -nan    0.000000           0         1           gettimeofday
  -nan    0.000000           0        20        20 semtimedop
  -nan    0.000000           0        22           clock_gettime
------ ----------- ----------- --------- --------- ----------------
100.00    0.000000                    43        20 total


$ strace -cp 37096
Process 37096 attached - interrupt to quit
^CProcess 37096 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  -nan    0.000000           0         1           gettimeofday
  -nan    0.000000           0        20        20 semtimedop
  -nan    0.000000           0        22           clock_gettime
------ ----------- ----------- --------- --------- ----------------
100.00    0.000000                    43        20 total
```

o12.1

What we have observed.

- Any of the log writers can write.
- While the LGWR process can appear to be the "parent" or "master," don't count on it.
- LGWRs normally sleep for 3 seconds, but they can be woke up.
- Oracle redo does not have to be written to an online redo log every 3 seconds.
- By default, when a server process commits, control is not regained until it's redo has been written to an online redo log.

What is the commit write facility?

- Normally, when a commit occurs the server process waits for the LGWR to immediately and synchronously write the committed data to an on-line redo log.
- The commit write facility allows us to change these rules!
- We can do this at the instance level, session level, and commit statement level.
- Instance parameters changed in o11, same in 12c.
- wait options: wait or nowait
- logging options: immediate or batch

http://shallahamer-orapub.blogspot.com/2012/02/speed-sometimes-means-changing-rules.html

```
SQL> show parameter commit_wait

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
commit_wait                          string

SQL> show parameter commit_logging

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
commit_logging                       string
SQL> alter system set commit_wait=nowait;

System altered.

SQL> alter system set commit_logging=batch;

System altered.

SQL> show parameter commit_wait

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------
commit_wait                          string      NOWAIT
SQL> show parameter commit_logging

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
commit_logging                       string      BATCH
```

Control at the instance level.

O 11g, 12c

Control at statement level

```
-- This is the dml2.sql SQL
--
def startv=&1
def endv=&2
def wait=&3
def batch=&4
def opt='write &wait &batch';
begin
  loop
    update op_test
    set    object_name='kldfkjldfkjkldakldfkjldfskjldfsk;l'
    where  object_id=trunc(dbms_random.value(&startv,&endv));
    commit &opt;
  end loop;
end;
/
```

commit write nowait batch

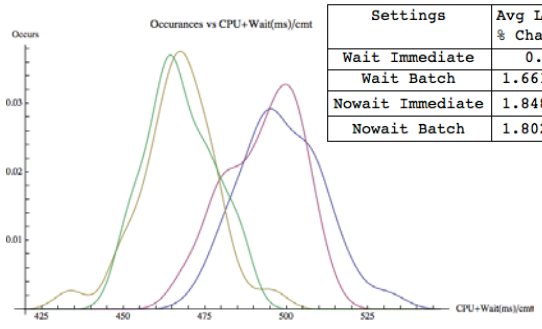
An example of how to investigate the impact of the commit write options.

```
wait=nowait
batch=batch

sqlplus tester/jester @./dml2.sql     1  5000 $wait $batch &
sqlplus tester/jester @./dml2.sql  5500 10000 $wait $batch &
sqlplus tester/jester @./dml2.sql 10500 15000 $wait $batch &
sqlplus tester/jester @./dml2.sql 15500 20000 $wait $batch &
sqlplus tester/jester @./dml2.sql 20500 25000 $wait $batch &
sqlplus tester/jester @./dml2.sql 25500 30000 $wait $batch &
```

Why would I do this?

Results w/CPU bottleneck.

Settings	Avg Work (cmt)	Avg L (cmt/ms)	Avg CPUt (ms/cmt)	Avg Wt (ms/cmt)	Avg Rt (ms/cmt)	Samples
Wait Immediate	749.	0.00831995	470.976	27.6505	498.626	32
Wait Batch	761.656	0.00845821	465.452	27.2719	492.724	32
Nowait Immediate	762.75	0.00847372	465.186	1.33771	466.523	32
Nowait Batch	762.406	0.00846991	467.24	0.196875	467.436	32



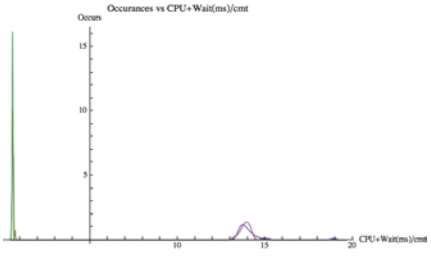
Settings	Avg L % Change	Avg L Ttest pvalue	Avg Rt % Change	Avg Rt Ttest pvalue
Wait Immediate	0.	1.	0.	1.
Wait Batch	1.66179	0.0119672	-1.18367	0.0541915
Nowait Immediate	1.84825	0.0245961	-6.43826	1.95955×10^{-15}
Nowait Batch	1.80249	0.00458214	-6.25515	5.38924×10^{-16}

Is there really an improvement in response time?

Results w/IO bottleneck

Setting	Commit/sec	CPU %	log file sync Wait Time %	log file par write Wait Time %
wait immediate	0.521	4%	84%	14%
wait batch	0.526	4%	80%	13%
nowait batch	12.639	45%	0%	14%
nowait immediate	13.004	42%	0%	14%

Settings	Avg Work (cmt)	% Change	Avg L (cmt/ms)	% Change	Avg CPUt (ms/cmt)	% Change	Avg Wt (ms/cmt)	% Change	Avg Rt (ms/cmt)	% Change
Wait Immediate	31238.5	0.	0.52056	0.	0.24456	0.	13.8391	0.	14.0836	0.
Wait Batch	31593.9	1.13787	0.526483	1.13783	0.241944	-1.06973	13.656	-1.32262	13.898	-1.31823
Nowait Immediate	758601.	2328.42	12.6386	2327.88	0.129128	-47.1998	0.45859	-96.6863	0.587718	-95.8269
Nowait Batch	780506.	2398.54	13.0041	2398.09	0.129801	-46.9245	0.441475	-96.8099	0.571276	-95.9437



Changing the LGWR OS priority.

- Oracle is already messing with priorities via the _high_priority_processes parameter.
- 12.1c default is “LMS*|VKTM”
- If there is a log writer issues and the OS is CPU constrained, consider adding “LGWR” or “LG*” to the high priority list.

Research Blog Postings:
http://shallahamer-orapub.blogspot.com/2014/10/how-to-change-priority-of-oracle.html
http://shallahamer-orapub.blogspot.com/2014/11/does-increasing-oracle-background.html

Changing the LGWR priority.
default in 11g and 12c

```
SQL> @ipx %high%priority%proc%

Instance Parameter and Value                        Description          Dflt?
--------------------------------------------------- -------------------- -----
_high_priority_processes            = LMS*|VKTM     High Priority        TRUE
                                                    Process Name Mask

$ ps -eo pid,class,pri,nice,time,args | grep prod35
20117 TS   19   0 00:00:15 ora_pmon_prod35
20121 RR   41   - 00:35:54 ora_vktm_prod35
20131 TS   19   0 00:00:04 ora_diag_prod35
20137 TS   19   0 00:00:12 ora_dbw0_prod35
20139 TS   19   0 00:00:05 ora_lgwr_prod35
20141 TS   19   0 00:00:24 ora_ckpt_prod35
20143 TS   19   0 00:00:05 ora_lg00_prod35
20145 TS   19   0 00:00:02 ora_lg01_prod35
20147 TS   19   0 00:00:04 ora_smon_prod35
20155 TS   19   0 00:00:54 ora_mmnl_prod35
```

O 11g, 12c

Changing the LGWR priority.
added LGWR to priority list

```
SQL> @ipx %high%priority%proc%

Instance Parameter and Value                        Description          Dflt?
--------------------------------------------------- -------------------- -----
_high_priority_processes            =               High Priority        FALSE
LMS*|VKTM|LGWR                                      Process Name Mask

$ ps -eo pid,class,pri,nice,time,args | grep prod35
60963 TS   19   0 00:00:00 ora_pmon_prod35
60967 RR   41   - 00:00:01 ora_vktm_prod35
60977 TS   19   0 00:00:00 ora_diag_prod35
60983 TS   19   0 00:00:00 ora_dbw0_prod35
60985 RR   41   - 00:00:00 ora_lgwr_prod35
60989 TS   19   0 00:00:00 ora_ckpt_prod35
60991 TS   19   0 00:00:00 ora_lg00_prod35
60993 TS   19   0 00:00:00 ora_lg01_prod35
60995 TS   19   0 00:00:00 ora_smon_prod35
61003 TS   19   0 00:00:00 ora_mmnl_prod35
```

O 11g, 12c

Changing the LGWR priority.
added LG* to priority list

```
SQL> @ipx %high%priority%proc%

Instance Parameter and Value                        Description          Dflt?
--------------------------------------------------- -------------------- -----
_high_priority_processes            =               High Priority        FALSE
LMS*|VKTM|LG*                                       Process Name Mask

$ ps -eo pid,class,pri,nice,time,args | grep prod35
61143 TS   19   0 00:00:00 ora_pmon_prod35
61147 RR   41   - 00:00:00 ora_vktm_prod35
61157 TS   19   0 00:00:00 ora_diag_prod35
61163 TS   19   0 00:00:00 ora_dbw0_prod35
61165 RR   41   - 00:00:00 ora_lgwr_prod35
61169 TS   19   0 00:00:00 ora_ckpt_prod35
61171 RR   41   - 00:00:00 ora_lg00_prod35
61175 RR   41   - 00:00:00 ora_lg01_prod35
61179 TS   19   0 00:00:00 ora_smon_prod35
61187 TS   19   0 00:00:00 ora_mmnl_prod35
```

O 11g, 12c

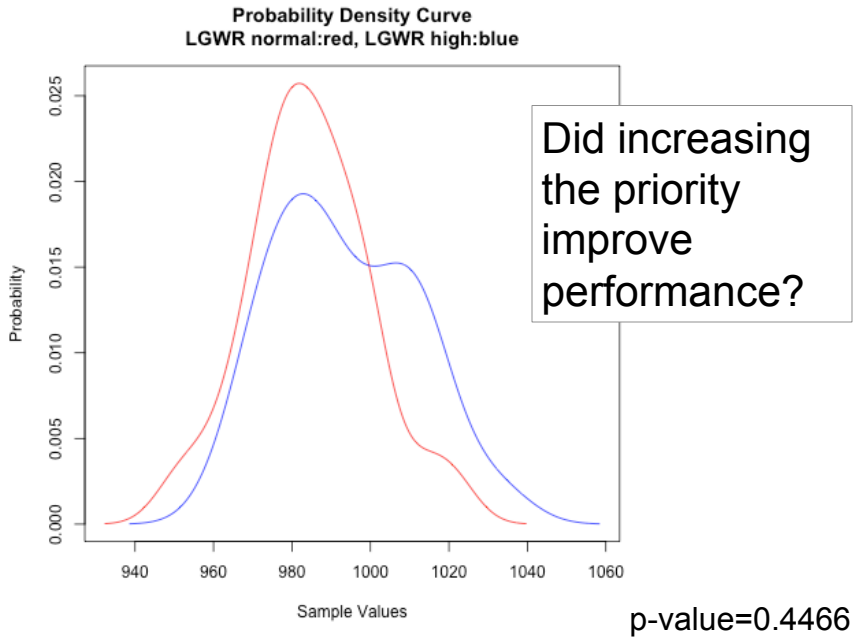
Changing the LGWR priority.
added *ORACLEPROD35 to priority list

```
SQL> @ipx %high%priority%proc%

Instance Parameter and Value                        Description          Dflt?
--------------------------------------------------- -------------------- -----
_high_priority_processes            =               High Priority        FALSE
LMS*|VKTM|*ORACLEPROD35                             Process Name Mask

$ ps -eo pid,class,pri,nice,time,args | grep prod35
31451 RR   41   - 00:00:00 ora_pmon_prod35
31455 RR   41   - 00:00:00 ora_psp0_prod35
31459 RR   41   - 00:00:00 ora_vktm_prod35
31487 RR   41   - 00:00:00 ora_dbw0_prod35
31491 RR   41   - 00:00:00 ora_lgwr_prod35
31495 RR   41   - 00:00:00 ora_ckpt_prod35
31499 RR   41   - 00:00:00 ora_lg00_prod35
31503 RR   41   - 00:00:00 ora_lg01_prod35
31507 RR   41   - 00:00:00 ora_smon_prod35
...
31669 TS   19   0 00:00:00 oracleprod35 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL
31671 TS   19   0 00:00:00 grep prod35
```

O 11g, 12c

But does increasing the LG* priority improve performance?

- I created a CPU bottleneck from DML processes.
- 99% of the total time was CPU.
- The top wait event was log file parallel write.
- With the normal process priority, an average of 984.5 commits/sec and a median of 983.0 commits/sec occurred.
- With the LG* high process priority, an average of 993.6 commits/sec and a median of 991.0 commits/sec occurred.
- The LGWR was never at the top of “top.”



Did increasing the priority improve performance?

p-value=0.4466

Creative Redo Maneuvers

- Redo architecture
- Using multiple log writers
- Commit write facility
- Changing log writer operating system priority

utah oracle users group

The most recent version of this presentation is on-line at www.orapub.com. Do an OraPub search for "serial".

Creative Oracle 12c Redo Maneuvers

Craig A. Shallahamer
OraPub, Inc.
craig@orapub.com

LinkedIn

ORACLE ACE Director