**ORACLE-PROD IS61-log-error**

Tue Feb 23 14:19:21 2021

Thread 1 cannot allocate new log, sequence 267445

Checkpoint not complete

Current log# 1 seq# 267444 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_1\_9JG8DOYB\_.LOG

Current log# 1 seq# 267444 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_1\_9JG8DP0J\_.LOG

Thread 1 advanced to log sequence **267445** (LGWR switch)

Current log# 3 seq# 267445 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP5W\_.LOG

Current log# 3 seq# 267445 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP7T\_.LOG

Tue Feb 23 14:19:24 2021

Archived Log entry 267014 added for thread 1 sequence 267444 ID 0xdddb2ce5 dest 1:

Tue Feb 23 14:19:39 2021

Thread 1 cannot allocate new log, sequence 267446

Checkpoint not complete

Current log# 3 seq# 267445 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP5W\_.LOG

Current log# 3 seq# 267445 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP7T\_.LOG

Thread 1 advanced to log sequence 267446 (LGWR switch)

Current log# 2 seq# 267446 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_2\_9JG8DP2H\_.LOG

Current log# 2 seq# 267446 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_2\_9JG8DP3Y\_.LOG

Tue Feb 23 14:19:42 2021

Archived Log entry 267015 added for thread 1 sequence 267445 ID 0xdddb2ce5 dest 1:

Tue Feb 23 14:22:54 2021

Thread 1 cannot allocate new log, sequence 267447

Checkpoint not complete

Current log# 2 seq# 267446 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_2\_9JG8DP2H\_.LOG

Current log# 2 seq# 267446 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_2\_9JG8DP3Y\_.LOG

Thread 1 advanced to log sequence 267447 (LGWR switch)

Current log# 1 seq# 267447 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_1\_9JG8DOYB\_.LOG

Current log# 1 seq# 267447 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_1\_9JG8DP0J\_.LOG

Tue Feb 23 14:22:58 2021

Archived Log entry 267016 added for thread 1 sequence 267446 ID 0xdddb2ce5 dest 1:

Tue Feb 23 14:27:10 2021

Thread 1 cannot allocate new log, sequence 267448

Checkpoint not complete

Current log# 1 seq# 267447 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_1\_9JG8DOYB\_.LOG

Current log# 1 seq# 267447 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_1\_9JG8DP0J\_.LOG

Thread 1 cannot allocate new log, sequence 267448

Private strand flush not complete

Current log# 1 seq# 267447 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_1\_9JG8DOYB\_.LOG

Current log# 1 seq# 267447 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_1\_9JG8DP0J\_.LOG

Thread 1 advanced to log sequence 267448 (LGWR switch)

Current log# 3 seq# 267448 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP5W\_.LOG

Current log# 3 seq# 267448 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP7T\_.LOG

Tue Feb 23 14:27:16 2021

Archived Log entry 267017 added for thread 1 sequence 267447 ID 0xdddb2ce5 dest 1:

Tue Feb 23 14:30:25 2021

Thread 1 cannot allocate new log, sequence 267449

Checkpoint not complete

Current log# 3 seq# 267448 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP5W\_.LOG

Current log# 3 seq# 267448 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP7T\_.LOG

Thread 1 cannot allocate new log, sequence 267449

Private strand flush not complete

Current log# 3 seq# 267448 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP5W\_.LOG

Current log# 3 seq# 267448 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_3\_9JG8DP7T\_.LOG

Thread 1 advanced to log sequence 267449 (LGWR switch)

Current log# 2 seq# 267449 mem# 0: D:\DATABASE\IS61\CTL1\IS61\ONLINELOG\O1\_MF\_2\_9JG8DP2H\_.LOG

Current log# 2 seq# 267449 mem# 1: D:\DATABASE\IS61\CTL2\IS61\ONLINELOG\O1\_MF\_2\_9JG8DP3Y\_.LOG

Tue Feb 23 14:30:31 2021

Archived Log entry 267018 added for thread 1 sequence 267448 ID 0xdddb2ce5 dest 1:

Here is a handy script that will show you the current sizes of your redo log files:

SELECT a.group#,  
 substr(b.member,1,30) name,  
 a.members,  
 a.bytes,  
 a.status  
FROM v$log a,  
 v$logfile b  
WHERE a.group# = b.group#  
;

GROUP# NAME MEMBERS BYTES STATUS

-------- ---------------------------------------- ---------- ---------- ----------------

1 D:\DATABASE\IS61\CTL1\IS61\ONL 2 52428800 ACTIVE

1 D:\DATABASE\IS61\CTL2\IS61\ONL 2 52428800 ACTIVE

2 D:\DATABASE\IS61\CTL1\IS61\ONL 2 52428800 ACTIVE

2 D:\DATABASE\IS61\CTL2\IS61\ONL 2 52428800 ACTIVE

3 D:\DATABASE\IS61\CTL1\IS61\ONL 2 52428800 CURRENT

3 D:\DATABASE\IS61\CTL2\IS61\ONL 2 52428800 CURRENT

GROUP# NAME MEMBERS BYTES STATUS

-------- ---------------------------------------- ---------- ---------- ----------------

1 D:\DATABASE\U611\CTL1\U611\ONL 2 52428800 ACTIVE

1 D:\DATABASE\U611\CTL2\U611\ONL 2 52428800 ACTIVE

2 D:\DATABASE\U611\CTL1\U611\ONL 2 52428800 ACTIVE

2 D:\DATABASE\U611\CTL2\U611\ONL 2 52428800 ACTIVE

3 D:\DATABASE\U611\CTL1\U611\ONL 2 52428800 CURRENT

3 D:\DATABASE\U611\CTL2\U611\ONL 2 52428800 CURRENT

Tom,I am receiving the following message in my alert log and can find  
no documentation explaining what this error means or how to correct it.  
Sun Feb 13 03:30:26 2000  
>>>>>**Thread 1 cannot allocate new log, sequence 41801**  
>>>>>Checkpoint not complete  
Current log# 1 seq# 41800 mem# 0: /kla/klaquest/dym/keep/redo1.log  
Current log# 1 seq# 41800 mem# 1: /kla/qdb1/dym/keep/redo1a.log  
Sun Feb 13 03:30:38 2000  
Thread 1 advanced to log sequence 41801  
Current log# 2 seq# 41801 mem# 0: /kla/klaquest/dym/keep/redo2.log  
Current log# 2 seq# 41801 mem# 1: /kla/qdb1/dym/keep/redo2a.log  
Any advice would be much appreciated.

**and we said...**

the infamous "checkpoint not complete, cannot allocate new log" message.  
**this occurrs when Oracle attempts to reuse a log file but the checkpoint that would flush the blocks that may have redo in this log file has not yet completed** -- **we must wait until that checkpoint completes before we can reuse that file** -- thats when this message is printed. during this time (when we cannot allocate a new log) processing is suspended in the database while the checkpoint is made to complete ASAP.  
  
The major way to relieve this is to have sufficient log to carry you through peak times. that way, we can complete the checkpoint while you are not busy.  
also **make sure your checkpoints happen as fast as they can** (eg: **enable ASYNC IO or configure >1 DBWR** if ansyc IO cannot be used, make sure disks are not contending with other apps and so on)  
**Another way is to make the log files smaller**, **hence increasing the frequency with which we checkpoint (log checkpoint interval and other init.ora parameters achieve the same effect btw**).  
  
**I myself prefer to use a modest sized log file (around 25meg) and will add logfiles until these messages cease...**

**Is it possible to increase this way..**log\_buffer = 2163840  
log\_checkpoint\_interval=10000  
lock\_checkpoint\_timeout=1800  
If this is a production instance and you are not in archive log mode (I'm ASSUMING by having log\_archive\_start=false you are not) your DAY WILL ONLY GET WORSE as the sudden loss of data tends to ruin peoples day and running in noarchivelog is a 100% sure way to have a sudden loss of lots of data

Nothing in your init.ora is going to affect this. **You need more log**, you have insufficient log to carry you through your load of work. **Add more logfiles**.  
Just add more logfiles -- see the **alter database command** for the syntax.  
(you know, a 16meg buffer cache and a 125m shared pool, 62m large pool -- you might have that BACKWARDS.... thats a HUGE shared pool, simply HUGE. I normally start at 50/60m or thereabouts). A 160k log buffer is a tad small as well.  
If this is a production instance and you are not in archive log mode (I'm ASSUMING by having log\_archive\_start=false you are not) your DAY WILL ONLY GET WORSE as the sudden loss of data tends to ruin peoples day and running in noarchivelog is a 100% sure way to have a sudden loss of lots of data!

wonder if you could clarify something for me regarding the "cannot allocate new log" message. From our alert log (see below), it seems that log#2 would have been available for Oracle to use. Instead, our system grinds to a halt until a checkpoint completes.

As soon as a checkpoint was completed for **[0x1a01.2.10],** then **log# 2** was able to be allocated. If I am reading the log correctly, it was not doing anything anyway so why could it not be used prior to the checkpoint completing? I'm matching up the "beginning log switch checkpoint up to RBA [xxxx]" with the "completed checkpoint up to RBA [xxxx]" to see that #1 and #3 were busy, but what was #2 doing that it could not be allocated? I think I'm repeating myself.

**here is the begin of the checkpoint on the data protected by log 2**  
  
Beginning log switch checkpoint up to RBA [**0x1a01.2.10**], SCN: 0x0000.644a1dbb  
Thread 1 advanced to log sequence 6657  
Current log# 3 seq# 6657 mem# 0: /redo02/oradata/eisp/redo3a.log  
Current log# 3 seq# 6657 mem# 1: /extra01/oradata/eisp/redo3b.log  
  
**Arch finished copying log 2 for us here**  
Tue Feb 25 05:06:35 2003  
ARC0: Beginning to archive log# 2 seq# 6656  
Tue Feb 25 05:07:00 2003  
ARC0: Completed archiving log# 2 seq# 6656  
  
Tue Feb 25 05:51:12 2003  
Beginning log switch checkpoint up to RBA [0x1a02.2.10], SCN: 0x0000.644a251e  
Thread 1 advanced to log sequence 6658  
Current log# 1 seq# 6658 mem# 0: /redo01/oradata/eisp/redo1a.log  
Current log# 1 seq# 6658 mem# 1: /redo02/oradata/eisp/redo1b.log  
  
Tue Feb 25 05:51:12 2003  
ARC0: Beginning to archive log# 3 seq# 6657  
Tue Feb 25 05:51:35 2003  
ARC0: Completed archiving log# 3 seq# 6657  
  
Tue Feb 25 06:39:05 2003  
**Thread 1 cannot allocate new log, sequence 6659**  
**Checkpoint not complete**  
Current log# 1 seq# 6658 mem# 0: /redo01/oradata/eisp/redo1a.log  
Current log# 1 seq# 6658 mem# 1: /redo02/oradata/eisp/redo1b.log  
  
**Here is where the checkpoint finished up -- the one started for number 2**  
  
Tue Feb 25 06:56:30 2003  
Completed checkpoint up to RBA [0x1a01.2.10], SCN: 0x0000.644a1dbb  
  
Tue Feb 25 06:56:30 2003  
Beginning log switch checkpoint up to RBA [0x1a03.2.10], SCN: 0x0000.644a2b29  
Thread 1 advanced to log sequence 6659  
Current log# 2 seq# 6659 mem# 0: /redo01/oradata/eisp/redo2a\_NEW.log  
Current log# 2 seq# 6659 mem# 1: /extra01/oradata/eisp/redo2b.log  
  
**that is why is could not be reused -- the blocks it protects in the buffer cache were NOT yet on disk...**  
  
Now, given how fast the archive went -- and how long it took to catch up -- I'd be looking at "why"  
 - perhaps you are not checkpointing frequently enough (too many dirty blocks)   
--or dbwr is not "tuned" -- the disks are really slow?

In your book, you mention that Log Writer is useful to improve the performance since it does sequnetial writing. Using only **DBWR** (non-sequential writing) will degrade the performance.  
But, as I understand, the "checkpoint not complete, cannot allocate new log", error \*indicates\* "DBWR cannot compete with **LGWR**".Where do I fail to understand ?  
  
Umm, it is not an "either or" thing - I don't know how you got that idea.  
**lgwr** is a point of serialization in the database -- we all wait on him.  
**dbwr** is a guy that goes on in the background.  
**the checkpoint not complete indicates that dbwr was not keeping up with lgwr and is generally caused by undersizing your log files (or number thereof)**

"the checkpoint not complete indicates that dbwr was not keeping up with lgwr and  
is generally caused by undersizing your log files (or number thereof) "  
I understand that they do totally diffeent works, but, LGWR has to wait for DBWR to complete. What I was refering the \*only\* benifit of sequential writing by the LGWR.  
Suppose there are 10 Log files. By making it 100 or even 1000 does not guarantee that the error would not occur.  
At some point, sooner or later, DBWR has to go ahead of LGWR.

not ahead, just keep up with.  
Yes, if you constantly run at full steam -- 100% -- **you must architect a system such that DBWR is able to keep up with lgwr. this is done by having sufficient devices and cpus to spread the scattered write work out**.  
what I said stands 100% though "IS GENERALLY caused by ....." Most systems run with peaks and valleys -- on these systems (the vast majority out there) the solution is in fact to size logs such that they carry you over the peak and lets the valleys catch up.

Tom,  
Ive read the concepts manual on the DBWR, LGWR, CKPT background processes about 3 times before posting this question :). I still find it a little hard to grasp as to why checkpoints take very long sometimes.  
From what I understand, 1) **DBWRs usually do a good job in flushing dirty buffers to disk**. One could ofcourse configure multiple DBWRs if they see a need. 2) **LGWRs also seem to do a good job of flushing from redo buffers to redolog files**. It is the checkpointing process (either using CKPT or LGWR) that is invariably the real bottleneck. But why??  
Checkpoints merely update file headers with updated SCNs upon a commit correct? I realize that it is proportional to number of datafiles, io/disk speed. But conceptually, I don’t understand why it should take very long sometimes to update a few datafiles (say 100), controlfiles with a few SCNs? I can understand if every data block had something that had to be updated with this information. But that isn’t the case right? Just the datafile headers and controlfiles get updated with the new SCN. The data has already been flushed to disk (the individual data blocks) by the DBWR. So why the big delay now to just update file headers? Even if the disks were really slow, why should it take all that long??  
Thanks for your time.

lgwr signals a checkpoint,

**dbwr is the main checkpoint process**, **ckpt has the relatively simple job of updating file headers**.

dbwr is invariably the bottleneck in a checkpoint.

**checkpoints flush the blocks in the buffer cache that are protected by redo**. **if you generate 50meg of redo -- you might be protecting thousands and thousands of blocks in the buffer cache**. before we can reuse that 50meg redo log file -- **we must flush to disk all of those thousands of dirty blocks**.

remember:

update t set indexed\_number\_column = 0 where primary\_key = 1;

updates a single row and generates very little redo (lets say 228 bytes for grins

ops$tkyte@ORA9IR2> update t set x = 0 where pk = 1;

Statistics

----------------------------------------------------------

0 recursive calls

1 db block gets

1 consistent gets

0 physical reads

228 redo size

784 bytes sent via SQL\*Net to client

800 bytes received via SQL\*Net from client

4 SQL\*Net roundtrips to/from client

1 sorts (memory)

0 sorts (disk)

1 rows processed

(just enough redo to say "change indexed\_number\_column to zero for rowid "x").

but -- in the buffer cache, we have just

a) changed an 8k table block that will need to be written

b) maybe caused a row migration (update a null to a non-null) so we might

have 2 8k table blocks

c) updated perhaps 2 index leaf blocks (column was 1000000000000000, is now 0,

index entry moved from the right to the left.

d) maybe caused a split up an index -- say 2 more blocks.

so that measely 228 bytes of redo might be protecting 4 8k blocks (32k of data) that dbwr must flush out at some point.

this is why we use redo -- instead of just writing to the datafiles when you commit. we can write 228 bytes sequentially to the log files -- instead of writing 32k of data out using random IO (lots of seeking around) **the checkpoint writes out all of these dirty blocks -- that is what takes so "long**"

select \* from v$sysstat where upper(name) like '%DBWR%';

select name, value from v$sysstat where name like '%redo%';

NAME VALUE

------------------------------ ----------

redo synch writes 3

redo synch time 0

redo entries 492

redo size 126044

redo buffer allocation retries 0

redo wastage 11120

redo writer latching time 0

redo writes 38

redo blocks written 277

redo write time 0

redo log space requests 0

redo log space wait time 0

redo log switch interrupts 0

redo ordering marks 0

14 rows selected.

redo size... lots of redo stats.

**ONZE ORACLE-PROD**

**Let op: Dit getal zegt niets over de grootte van de REDO-LOG, dit is optelsom van alle redo die gegenereerd is.**

redo blocks read for recovery 0

redo k-bytes read for recovery 0

redo k-bytes read for terminal recovery 0

redo entries 164744457

redo size 63.910.986.304

redo entries for lost write detection 0

redo size for lost write detection 0

redo size for direct writes 703788608

redo buffer allocation retries 2374

redo wastage 4337983476

redo writes 8912499

redo blocks written 137741661

redo write time 348941

redo blocks checksummed by FG (exclusive) 28194534

redo blocks checksummed by LGWR 0

redo log space requests 3131

redo log space wait time 3344

redo ordering marks 104694

redo subscn max counts 1196418

redo write broadcast ack time 0

redo write broadcast ack count 0

redo write broadcast lgwr post count 0

redo synch time 300454

redo synch time (usec) 3437521859

redo synch time overhead (usec) 222299299632

redo synch time overhead count (<2 msec) 2002750

redo synch time overhead count (<8 msec) 695541

redo synch time overhead count (<32 msec) 149264

redo synch time overhead count (<128 msec) 1290

redo synch time overhead count (>=128 msec) 2405

redo synch writes 2932374

redo synch long waits 550299

redo synch poll writes 340112

redo synch polls 245340

redo write info find 2851332

redo write info find fail 82

redo KB read 68870722

redo KB read (memory) 0

redo KB read for transport 0

redo KB read (memory) for transport 0