Data Guard Redo Apply and Media Recovery Best Practices Oracle Database 10*g* Release 2

Oracle Maximum Availability Architecture White Paper
September 2007

Maximum Availability Architecture

Oracle Best Practices for High Availability

Data Guard Redo Apply and Media Recovery Best Practices Oracle Database 10*g* Release 2

Data Guard Redo Apply and Media Recovery Best Practices Oracle Database 10*g* Release 2

INTRODUCTION

Oracle media recovery is a fundamental element of Maximum Availability Architecture (MAA) environments. Media recovery occurs when one or more datafiles or the controlfiles are restored from a previous backup or when using Data Guard Redo Apply (physical standby database). The goal of media recovery is to recover a database to a consistent point in time or to apply all primary database transactions to a physical standby database.

In most cases, the default Oracle settings result in satisfactory performance for media recovery. As applications and databases increase in size and throughput, however, media recovery operations can benefit from additional tuning to further optimize recovery time or Redo Apply throughput on a standby database. The goal of this paper is to provide best practices for monitoring media recovery performance and, if necessary, for tuning media recovery for optimal performance.

EXECUTIVE SUMMARY

Based on MAA test results and customer experiences, the following list describes the results obtained after adopting the best practices outlined in this paper:

- The redo apply rate for the Oracle Internal Mail Database (supporting 65,000 active users) increased from 8 megabytes per second (MB/sec) to 38 MB/sec.
- The redo apply rate on a physical standby database for a customer workload consisting of batch processing of online transaction processing (OLTP) transactions was observed to be over 25MB/sec.
- The recovery performance in MAA tests for OLTP workload is affected most by I/O bandwidth. For example, doubling the number of spindles increased recovery rates by over 100%.
- In contrast, media recovery performance in MAA tests using direct load operations did not display the same benefits from increasing the I/O bandwidth. For example, direct load operations sustained 38MB/sec

recovery rate using 28 spindles. Reducing the spindle count by half had little effect on the recovery rate.

- For additional protection against data corruption at the standby database, the impact of setting DB_BLOCK_CHECKSUM to FULL was approximately 1% for an OLTP primary workload and approximately 3% for a direct path primary workload.

DATA GUARD REDO APPLY AND MEDIA RECOVERY BEST PRACTICES

The best practices outlined in this paper have been derived as a result of testing media recovery on Oracle Database 10*g* Release 2 in the Oracle Maximum Availability Architecture (MAA) lab. In addition to Oracle MAA testing, this paper also utilizes results obtained by Oracle customers who have optimized media recovery in their environments.

Oracle recommends the following best practices for improving the performance of media recovery:

- Allow media recovery to control the number of processes for parallel media recovery. By default, media recovery uses the CPU_COUNT initialization parameter to determine the number of processes to use for recovery operations.
- Configure PARALLEL_EXECUTION_MESSAGE_SIZE = 65535. The message size parameter is used by all parallel query operations and pulls memory from the shared pool. Therefore, when increasing PARALLEL_EXECUTION_MESSAGE_SIZE (PEMS) consider the amount of available shared pool memory. Note that 64k is the maximum value for a non-Oracle RAC database. For an Oracle RAC database, you should continue to set PEMS to 64k but be aware that the actual value might be automatically lowered to the IPC limit for the OS.
- The overall recovery rate can be significantly impacted by enabling DB_BLOCK_CHECKING. Setting DB_BLOCK_CHECKING=FALSE during media recovery can provide as much as a two fold increase in the apply rate. The lack of block checking during recovery must be an accepted risk. Block checking is recommended for the production database.
- Tune the I/O subsystem. The most important factor in obtaining high recovery rates is tuning the I/O subsystem. When tuning I/O for media recovery:
    - Ensure that Oracle can use ASYNC I/O. Note that, by default, the Oracle database is configured for asynchronous I/O. However, you must also properly configure the operating system, the host bus adapter (HBA) driver, and storage array. Consult your operating system and storage array documentation for further information on enabling ASYNC I/O.

  - Maximize the I/O write size through all layers of the I/O stack. The layers will include one or more of the following: operating system (including async write size), device drivers, storage network transfer size, and disk array. Consult your operating system and storage array documentation for further information on enabling ASYNC I/O tuning I/O write sizes. Media recovery issues 1MB read I/O to the redo log.
  - Set the initialization parameter DISK_ASYNCH_IO=TRUE (default). If asynchronous I/O is not available, consider using the DBWR_IO_SLAVES parameter to simulate asynchronous I/O.
  - Set the DB_WRITER_PROCESSES parameter to a value greater than 1 when asynchronous I/O is available and *free buffer waits* or *checkpoint completed* wait events are the top Oracle waits. Multiple database writers are only applicable if you have sufficient CPU and I/O bandwidth.
  - If you continue to receive high "free buffer wait" as a significant database wait event in V$SYSTEM_EVENT after implementing the above recommendation, then consider increasing the buffer cache.
  - Increase the size of the primary database's online redo log and standby database's standby redo logs to reduce the number of times a full checkpoint is performed.
  - Place online redo logs and standby redo logs in ASM diskgroups with the best performance characteristics.
- Query the V$RECOVERY_PROGRESS view to monitor and assess recovery performance in real time.
- Query the V$SYSTEM_EVENT view to monitor and assess database wait events that might lead to recovery optimizations.

ASSESSING MEDIA RECOVERY PERFORMANCE

The first step in any tuning exercise is to assess the current performance and use that as a baseline. As changes are made, reassess and compare the performance level to the baseline to get a level of improvement. Monitor the performance of media recovery by querying the V$RECOVERY_PROGRESS view. This view contains the following columns:

Average Apply Rate: Redo Applied / Elapsed Time includes time spent actively applying redo and time spent waiting for redo to arrive.

Active Apply Rate: Redo Applied / Active Time is a moving average over the last 3 minutes. The rate does not include time spent waiting for redo to arrive.

Apply Time per Log: Average time spent actively applying redo in a log file.

Checkpoint Time per Log: Average time spent for a log boundary checkpoint.

Last Applied Redo: SCN and Timestamp of last redo applied. This is the time as stored in the redo stream, so it can be used to compare where the standby database is relative to the primary.

Note that the Average Apply Rate (includes the) time waiting for redo to arrive. This can falsely indicate an apply rate that is much lower than the actual apply rate. To get a true assessment, you may wish to stop media recovery, allow several archive logs to accumulate, and then start media recovery. By doing so, you remove the idle time from the average apply rate and get a much more accurate picture of the true apply rate

In a Data Guard physical standby environment, it is important to determine if the standby database can recover redo as fast as, or faster than, the primary database can generate redo. The simplest way to determine application throughput in terms of redo volume is to collect Automatic Workload Repository (AWR) reports on the primary database during normal and peak workload and determine the number of bytes per second of redo data the production database is producing. You can then compare the speed at which redo is being generated with the Average Apply Rate or Active Apply Rate to determine if the standby database is able to maintain the pace.

While using AWR reports is an effective method of comparing application throughput and recovery rates over a defined period of time, it is possible to get an instantaneous view of redo generate rate and recovery rates using V$SYSSTAT. On the primary database take two snapshots (P1 and P2) of the "redo blocks written" statistic from the V$SYSSTAT view separated by an interval of time (T). The instantaneous primary redo generation rate can be derived as:

```
(P2 - P1)/T
```

On the standby database, take two snapshots (S1 and S2) of the "redo blocks read for recovery" statistic from the V$SYSSTAT view at the same interval of time (T) as above. The instantaneous apply rate can be derived as:

```
(S2 - S1)/T
```

To determine if media recovery tuning is required, use the redo apply rate quick assessment chart below.

Table 1: Redo Apply Rate Quick Assessment

Redo Generation Rate vs Redo Apply Rate	Assessment and Recommendation
2 * Max Primary Database Redo Generation Rate < Redo Apply Rate	Excellent – No tuning required with plenty of room for future growth
Max Primary Database Redo Generation Rate < Redo Apply Rate	Good – Tuning is Optional
Average Primary Redo Generation Rate < Redo Apply Rate	OK – Could benefit from tuning in order to accommodate peaks in workload
Average Primary Redo Generation Rate > Redo Apply Rate	Bad - Needs tuning. Call Oracle Technical Support if all tuning steps have been followed and the redo apply rate is still too slow. Refer to the Tuning Media Recovery section.

The recovery rate may vary depending on the primary database's transaction activity. Typically, recovery rate is much higher when the number of distinct blocks being changed is small or during batch processing because both cases result in less I/O either because of the number of changed blocks or because of coalescing blocks in a single I/O. In most applications, a predictable pattern surfaces after monitoring for several days.

TUNING MEDIA RECOVERY

The physical standby database or a recovery instance is unlike the primary database. While the typical primary instance may be comprised of 95% or more query activity with many CPU intensive operations, the media recovery instance is very write and update intensive. The recovery instance's goal is to apply changes to data blocks and write them to the data files. In many cases, the media recovery instance requires much less overall CPU resources but equal or greater I/O or memory capacity. Instead of possibly hundreds of CPU intensive operations on the primary, only the recovery coordinator (PID of the foreground process in `V$PROCESS`) or MRP process (MRP0 PID found in `V$MANAGED_STANDBY`) is generally CPU intensive. In many cases, fewer but faster CPUs will typically enhance recovery performance. There is no simple formula to predict the standby database system utilization. Here are some general observations found during MAA testing.

- The higher the read ratio on the primary database, the greater the difference in CPU utilization between the primary and standby databases.
- Higher numbers of sorts or complex queries executed will require more CPU utilization on the primary database. Queries and sorts do not create additional redo and thus do not create additional work on the standby database.

- If the recovery set or the number of distinct blocks that are being modified during recovery is small, you may utilize less CPU resources and also get higher Redo Apply rates because the blocks are usually cached and applying redo is more optimized. For example, the SQL*Loader runs had a relatively small recovery set compared to our OLTP runs; thereby the redo apply rate was higher and CPU consumption was 30% less. .

Therefore, tuning media recovery focuses primarily on removing system resource contention and database wait constraints.

Use the following methodology if your monitoring of the media recovery indicates that tuning is required. Media recovery consists of three distinct phases. Each phase must be assessed and tuned if the recovery rate is not sufficient.

1. **Log Read Phase** involves the reading of redo from the standby redo logs or archived redo logs by the recovery coordinator or Managed Recovery Process (MRP).
2. **Redo Apply Phase** involves the reading of data blocks into the buffer cache and the application of redo, by parallel recovery slave processes. The recovery coordinator (or MRP) ships redo to the recovery slaves using the parallel query (PQ) interprocess communication framework.
3. **Checkpoint Phase** involves database writer (DBWR) flushing to disk modified (dirty) data blocks and the update of data file headers to record checkpoint completion.

It is important to understand that in a Data Guard Redo Apply configuration with an Oracle RAC standby database, media recovery will always execute on a single instance in the cluster, called the *apply instance*. In addition, when using the Data Guard Broker with an Oracle RAC standby database, a surviving instance automatically assumes the role of the apply instance should the original apply instance fail. The first step in tuning media recovery is to follow the best practices outlined in the beginning of this paper. If after implementing best practices your monitoring of media recovery indicates that performance still requires improvement, conduct additional tuning using the following methodology:

- Collect data necessary for tuning
- Assess system resources
- Assess database waits

Collect Data Necessary for Tuning

To perform accurate recovery tuning, set Timed_Statistics=TRUE on the primary and standby databases and collect the following information during the recovery session:

- Output from the operating system (OS) sar command, which was invoked with a 60-second interval from the primary and standby databases

- Output from the OS vmstat command for the recovery period for the standby database
- AWR reports from the primary database
- Output from the following query from V$SYSSTAT run on a 60-second interval on the standby database:

```
select name, value, to_char(sysdate, 'DD-MON-YYYY HH:MI:SS')
from v$sysstat where value > 0 order by name;
spool off
```

- Output from the following query from V$SYSTEM_EVENT run on a 60-second interval on the standby database:

```
set echo off
set feedback off
set termout off
set pagesize 100
set numwidth 10
column event format a40 truncate
column name format a35 truncate
column opname format a35 truncate
column value format 99999999999999
```

```
select event, total_waits, time_waited,
average_wait*10
from v$system_event where time_waited > 100 and
event not like 'rdbms ipc %' and event not like
'%timer%'and lower(event) not like '%idle%'
and lower(event) not like 'sql%net%'
and event not like 'ges%'
order by time_waited;
```

- Output from the following query on V$RECOVERY_PROGRESS run once at the completion of the media recovery operation on the standby:

```
set echo off
set feedback off
set termout off
set pagesize 100
set numwidth 15
column type format a20 truncate
column item format a25 truncate
column units format a10 truncate
column name format a30 truncate
```

```
select to_char(start_time,'dd-mon-yyyy HH:MI:SS')
start_time, type, item, units, sofar, total,
to_char(timestamp,'dd-mon-yyyy HH:MI:SS') timestamp from
v$recovery_progress;
```

Assess System Resources

Use system commands such as UNIX sar and vmstat or system monitoring tools to assess system resources.

- If there are I/O bottlenecks or excessive wait I/Os, then stripe across more spindles/devices or leverage more controllers. A stripe size between 256KB to 1MB is optimal to leverage your I/O subsystem. Verify that this is not a bus or controller bottleneck or any other I/O bottleneck. The read I/O from the standby redo log should be greater than the expected recovery rate.
- Assess I/O read rates from standby redo logs or archived redo logs

  ```
  ALTER SYSTEM DUMP LOGFILE ‘redo log name’ VALIDATE;
  ```

  A trace file is generated with the redo read rate from a recovery perspective. The overall recovery rate will always be bounded by the rate at which redo can be read; so ensure that the redo read rate surpasses your required recovery rate. The following UNIX example shows how to measure the maximum redo read rate for recovery. Oracle uses a 4 MB read buffer for redo log reads.[1]

  ```
  % /bin/time dd if=/redo_logs/t_log8.f of=/dev/null bs=4096k

  50+1 records in
  50+1 records out

  real 6.4
  user 0.0
  sys 0.1
  ```

  Estimated Read Rate (200 MB log file) = (50 * 4 MB) / 6.4s = 31.25 MB/sec

- Check for excessive swapping or memory paging.
- Check to ensure the recovery coordinator or MRP is not CPU bound during recovery.
- If excessive CPU is being used, consider removing non-database CPU resource consuming programs off the system and setting DB_BLOCK_CHECKING to lower values.

[1] If you repeat this simple test, use a different SRL or archive log since the data may be cached making the results artificially high and incorrect.

Assess Database Waits

- Database wait events from `V$SYSTEM_EVENTS` and `V$SESSION_WAITS`
    - Refer to the top 10 system wait events and tune the biggest waits first. You can determine the top system and session wait events by querying `v$session_wait` and `v$system_event` and taking the top waits with the largest “`TIME_WAITED`” value

      If recovery is applying a lot of redo efficiently, the system will be I/O bound and the I/O wait should be reasonable for your system. The following are the top recovery-related waits that you may observe. Only apply the tuning tips if the recovery events are in the top 10 waits.

Table 2: Key Wait Event Table

Wait Name	**Description**	**Tuning Tips**
Log File Sequential Reads	Coordinator (recovery session or MRP process) wait for log file read I/O.	Tune Log Read I/O
PX Deq: Par Recov Reply	Coordinator synchronous wait for Slave (wait for checkpoints)	Increase PARALLEL EXECUTION MESSAGE SIZE to 65535
PX Deq Credit: send blkd PX Deq Credit: need buffer	Coordinator streaming wait for Slave (wait for apply)	Increase PARALLEL EXECUTION MESSAGE SIZE to 65535
Free buffer waits	Foreground waiting available free buffer in the buffer cache	Increase DB CACHE SIZE and remove any KEEP or RECYCLE POOL settings.
Direct path read	Coordinator wait for file header read at log boundary checkpoint	Tune File Read I/O
Direct path write	Coordinator wait for file header write at log boundary checkpoint	Tune File Write I/O
Checkpoint completed	Wait for checkpoint completed	Tune File Write I/O Increase number of DB WRITER PROCESSES
db file parallel read	Wait for data block read	Tune File Read I/O

TUNING CASE STUDIES

The following section provides examples of the potential benefits of tuning for media recovery performance. In this example, following the tuning recommendations in this paper produced the following performance gains:

Test Run	Redo Volume Recovered	Seconds for Media Recovery to complete recovery	MB of Redo applied/second (Redo Apply Rate)	Percentage improvement on baseline
Baseline	8723 MB	1290 sec	6.7 MB/sec	n/a - baseline
Recovery Tuning	8723 MB	1134 sec	7.7 MB/sec	14%
Recovery + I/O Tuning	8723 MB	1009 sec	8.6 MB/sec	28%

Details for this tuning example are as follows:

The hardware used in this exampled consisted of the following:

Host:

- Sun V65x with 2 3.0Ghz CPU's (4 hyper threaded)
- 6 GB of main memory
- Red Hat Enterprise Linux AS release 4 (Nahant Update 3) (2.6.9-34.ELsmp)

Storage:

- EMC CX700 for Data Area (two test with 14 and 28 spindles)
- EMC CX500 for Flash Recovery Area (two test with 14 and 28 spindles)
- ASM used for both Data Area and Flash Recovery Area

Database:

- RDBMS 10.2.0.2
- 2.4 GB SGA
  - 1.7 GB buffer cache
- 4 redo log groups of 1 GB each
- 2 members per group
- 2 controlfiles
- UNDO tablespace
- Flashback Database

Application used to generate redo

- 250 Warehouse Database TPCC for OLTP

Flashback Database was enabled and a guaranteed restore point was created before starting the test. The TPCC workload was run generating 8.7GB of OLTP data. To test recovery performance, the database was flashed back to the restore point, and media recovery was used to apply the archived logs generated in the original run. A script was used that performed the following:

1. Flashback Database to a beginning restore point
2. Recover 8.7 GB of OLTP load (TPCC application)
3. Gather statistics

The first recovery run was performed prior to performing any tuning. At the end of the recovery run the apply rate was obtained using the V$RECOVERY_PROGRESS view.

```
ITEM                 UNIT                     TOTAL
=================    =========                =======
Log Files            Files                          9
Active Apply Rate    KB/sec                      6038
Average Apply Rate   KB/sec                      7006
Redo Applied         Megabytes                   8723
Active Time          Seconds                      946
Apply Time per Log   Seconds                       86
Checkpoint Time per  Seconds                       18
Elapsed Time         Seconds                     1290
```

Looking at the top wait events in the V$SYSTEM_EVENT view for the initial recovery run we see:

```
EVENT                           TOTAL_WAITS     TIME_WAIT
===========================     ===========     =========
db file sequential read              17285             38
control file parallel write            447            171
recovery read                        68648            191
checkpoint completed                   125            392
db file parallel write                7314            514
PX Deq Credit: need buffer            5320            766
PX Deq: Test for msg                 90053           1003
PX Deq Credit: send blkd             2496            2224
```

From the above we see that the top wait events are centered on messaging between the parallel query slaves and coordinator. To make the message passing more efficient we can increase the PARALLEL_EXECUTION_MESSAGE_SIZE from the default of 2k to 64k. After making the parameter change a flashback to the restore point was performed and the test was re-run. The following illustrates the new recovery rate:

```
ITEM                 UNIT                     TOTAL
=================    =========                =======
Log Files            Files                          9
Active Apply Rate    KB/sec                      7690
Average Apply Rate   KB/sec                      7916
Redo Applied         Megabytes                   8723
Active Time          Seconds                      823
Apply Time per Log   Seconds                       86
Checkpoint Time per  Seconds                       18
Elapsed Time         Seconds                     1134
```

We see an increase in over 13% increase as a result of our change. Looking at the top wait events we see the following:

```
EVENT                               TOTAL_WAITS     TIME_WAIT
===========================         ===========     =========
PX qref latch                            145151            28
PX Deq Credit: need buffer                17285            38
control file parallel write                 447           171
PX Deq Credit: send blkd                  68648           191
checkpoint completed                        125           374
db file parallel write                     7314           517
PX Deq: Par Recov Reply                    5320           666
PX Deq: Test for msg                      90053           903
recovery read                              2496          2627
```

The “recovery read” wait event is the amount of time needed to perform reads from the log files in the flash recovery area. In general, when the database is tuned optimally, the top wait events will be I/O related. From this point we can attempt to further optimize the I/O sub system. In this environment the following changes were made at the OS kernel level to assure that the I/O rate was optimal:

```
      fs.aio-max-nr = 1048576
      fs.aio-max-pinned = 1671168
```

After making the changes, the workload was run once again. The recovery rate increased as follows:

```
ITEM                 UNIT                     TOTAL
=================    =========                =======
Log Files            Files                          9
Active Apply Rate    KB/sec                      8634
Average Apply Rate   KB/sec                      8852
Redo Applied         Megabytes                   8723
Active Time          Seconds                      724
Apply Time per Log   Seconds                       86
Checkpoint Time per  Seconds                       15
Elapsed Time         Seconds                     1009
```

ENABLING BLOCK CHECKING

By default, the Oracle database always validates the data blocks that it reads from disk. Enabling data and log block checksums by setting `DB_BLOCK_CHECKSUM` to `TYPICAL` enables Oracle to detect other types of corruption caused by underlying disks, storage systems, or I/O systems. Before a data block is written to disk, a checksum is computed and stored in the block. When the block is subsequently read from disk, the checksum is computed again and compared with the stored checksum. Any difference is treated as a media error.

With DB_BLOCK_CHECKING enabled, the Oracle Database verifies that the block is self-consistent by executing block-type specific logical checks after every change. If the block is inconsistent, then it is marked corrupt, an ORA-1578 error is returned, and a trace file is created containing details of the problem. Without block checking enabled, corruption can go undetected until a subsequent change discovers the inconsistency.

In the MAA lab, testing showed the following impact of enabling block checking:

- The impact of setting DB_BLOCK_CHECKING to MEDIUM, or FULL was approximately 80% for OLTP.
- The impact of setting DB_BLOCK_CHECKING to MEDIUM, or FULL was approximately 15% for direct path operations.

CONCLUSION

With Oracle Database 10*g* Release 2, you should be able to inherently achieve fast media recovery performance. The best practices described in this white paper provide a checklist you can use to ensure that media recovery is not being constrained by any bottlenecks. Optimized media recovery reduces the time required for Data Guard switchover, failovers, or database media recovery. This equates to more uptime and higher availability in the case of an unplanned or planned outage, which helps enterprises meet the Service Level Agreements associated with recovery time objectives.

Data Guard Redo Apply and Media Recovery Best Practices:
Oracle Database 10*g* Release 2
September 2007

Author: Michael Smith
Contributing Authors: Lawrence To, Joseph Meeks, Vinay Srihari

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com