Oracle Maximum
Availability Architecture

Redo Apply Best Practices

Oracle Data Guard and Active Data Guard

ORACLE WHITE PAPER | UPDATED AUGUST 2018

Table of Contents

Introduction

Oracle media recovery or redo apply is a fundamental element of Oracle Maximum Availability Architecture (Oracle MAA) environments. Media recovery occurs when one or more data files or the control files are restored from a previous backup or when using Data Guard Redo Apply for physical standby databases. The goal of media recovery and transaction recovery is to recover a database to a consistent point in time or to apply all primary database transactions using the redo change vectors 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. This technical paper is intended for Oracle Database Administrators familiar experienced with Oracle Database recovery and having a working knowledge of Data Guard and Active Data Guard.

Redo Apply Best Practices

The best practices outlined in this paper have been derived from extensive testing of media recovery by the Oracle MAA development team dedicated to documenting high availability (HA) best practices for Oracle Database. 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.

Size Online Redo Logs / Standby Redo Logs Appropriately

To a large degree, the issue that most often impacts redo apply performance is frequent log switches. Online redo logs and standby redo logs should use redo log size = 4GB or redo log size >= peak redo rate/minute x 20 minutes. To determine peak redo rates, please refer to AWR reports during peak workload periods such as batch processing, quarter or year-end processing. Table 1 provides a quick mapping of redo-rate to the minimum recommended redo log size:

TABLE 1: OPTIMAL REDO LOG SIZE

Peak redo rate according to EM or AWR reports	Recommended redo log group size
<= 5 MB/sec	4 GB
<= 25 MB/sec	16 GB
<= 50 MB/sec	32 GB
> 50 MB/sec	64 GB

Note: It is very important to determine peak rate using a small sample size as the average rate over a longer period of time will result in an artificially low measurement of peak rates. Also note that the terms 'redo apply' and 'media recovery' used in this paper are interchangeable and refer to the same process of using redo to recover Oracle Database transactions.

Proper Standby Redo Log Placement and Creation

Once the online redo logs have been appropriately sized you should create standby redo logs of the same size. It is critical for performance that standby redo log groups only contain a single member. In addition, for each redo log thread (a thread is associated with an Oracle RAC database instance), the number of Standby Redo Logs = number of Redo Log Groups + 1. Be sure to place single member standby redo log groups in the fastest available diskgroup. The objective is to have standby log file write times that are comparable to log file I/O on the primary database optimal redo apply performance.

Use Automatic Workload Repository or Standby Statspack

The Automatic Workload Repository (AWR) is supported with Active Data Guard standby databases as of Oracle Database 12c Release 2. For full details refer to the *Oracle Database Performance and Tuning Guide*. Appendix C contains configuration steps for Standby AWR.

Install and utilize standby statspack in Oracle Database versions prior to 12.2 to regularly assess redo apply performance (See Appendix A). It is a best practice to keep performance baseline statspack reports so that they can be used for comparison if redo apply performance changes. In addition, standby statspack reports can be used to identify and tune read-only queries executed on an Active Data Guard standby. If standby statspack cannot be installed, consider using in-memory ASH reports (See Appendix B).

Set DATA PROTECTION Parameters

Setting `DB_BLOCK_CHECKING=FULL` and `DB_LOST_WRITE_PROTECT=TYPICAL` is recommended for all primary and standby databases to prevent and detect physical block corruptions and lost writes. Setting `DB_BLOCK_CHECKING=MED | FULL` is recommended for additional checks to prevent logical block corruptions. MAA recommends that all of these parameters be enabled to provide the best data corruption protection.

Some customers may compromise by reducing `DB_BLOCK_CHECKING` to `MEDIUM` or `FALSE` for the primary or standby database if the performance impact is unacceptable; however, MAA strongly recommends enabling `DB_BLOCK_CHECKING` on all databases, or at least one database in a Data Guard environment, to protect the databases from logical block corruptions. Logical block corruptions are very rare but can lead to longer recovery time and more data loss when they occur.

In addition, set the `DB_LOST_WRITE_PROTECT` parameter to `TYPICAL` on the standby database to enable Oracle to detect writes that are lost in the I/O subsystem. The impact on redo apply is less than 5 percent for OLTP applications.

See My Oracle Support Note 1302539.1 - Best Practices for Corruption Detection, Prevention, and Automatic Repair for more information about all corruption detection configuration options and performance trade-offs in a Data Guard configuration.

Review Database Wait Events

With the Active Data Guard option and real-time query, you can use AWR or Statspack from the primary database to collect data from a standby database that is opened read-only and performing recovery. Any tuning or troubleshooting exercise should start with collecting Standby Statspack reports.

- In a properly tuned system, the top wait event is db file parallel write followed by checkpoint completed when recovery is the only workload running. A read-only workload in an Active Data Guard standby could change this expectation.
- Set the `DB_WRITER_PROCESSES` parameter to a value greater than 1 and up to CPU count 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 (>30% of DB Time) after implementing the above recommendation, then consider increasing the buffer cache.
- If, during normal processing, logs are switching more than once every 10 minutes (6 times/hour) consider increasing the size of the databases' online redo logs and standby redo logs. Both primary and standby logs should be changed and they should all be the same size.

Note: AWR support for Active Data Guard workloads is planned for the next release of Oracle Database 12c that follows 12.1.0.2.

Tune I/O Operations

DBWR must write out modified blocks from the buffer cache to the data files as quickly as possible. It is recommended to use native asynchronous I/O by setting `DISK_ASYNCH_IO` to `TRUE` (default). Ensure that you have sufficient I/O bandwidth and that I/O response time is reasonable for your system either by doing base I/O tests, comparing the I/O statistics with those for the primary database, or by looking at historical I/O metrics. It should be kept in mind that shared storage infrastructure is shared by many applications and the response time will vary. In the unusual case that asynchronous I/O is not available; use `DBWR_IO_SLAVES` to improve the effective data block write rate with synchronous I/O.

Review System Resources

Ensure sufficient resources are available and no major bottleneck exists on the system. See System Resources, later in this paper.

Redo Apply Performance

Redo apply performance largely depends on the type of workload being recovered as well as the system resources allocated to recovery.

Performing recovery of an OLTP workload is generally very I/O intensive as there will be a large amount of small random read and writes. The higher the number of IOPS (I/O per second) that a storage subsystem can handle, the faster will be the recovery rate can be achieved. In contrast, recovery of batch workloads is naturally more efficient due to large sequential reads and writes, resulting in much faster recovery rates than OLTP workloads running on equivalent system resources. In addition, optimizations for recovery of batch direct load operations result in greater efficiency and even higher recovery rates. The difference between OLTP and batch recovery performance profiles explains why applications with varying mixtures of both OLTP and batch type workloads can have different recovery rates at a standby database even if primary database redo generation rates are similar.

Multi-Instance Redo Apply (MIRA)

Starting with Oracle Database 12.2, Multi-Instance Redo Apply (MIRA) greatly improves scalability of redo apply for Oracle RAC databases. Instead of merging all threads of redo into a single apply process, multiple apply instances divide the threads of redo between the apply instances. For example, when two apply nodes are used to apply four threads of redo from the primary, each apply instance will apply two threads.

When the workload is well balanced between all threads of the primary, the scalability of MIRA is predictable (2x, 4x etc.). Unbalanced workloads, where one instance does more work than another, will get mixed scaling results compared to single instance redo apply.

Activation

The number of apply instances is controlled by the Data Guard Broker database property `ApplyInstances` or the following SQL*Plus command;

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE INSTANCES [ALL|integer];
```

Figures 1 and 2 illustrate the increase in recovery rates for each type of workload over recent releases of Oracle Database on general purpose computer systems and Oracle Exadata Database Machine.

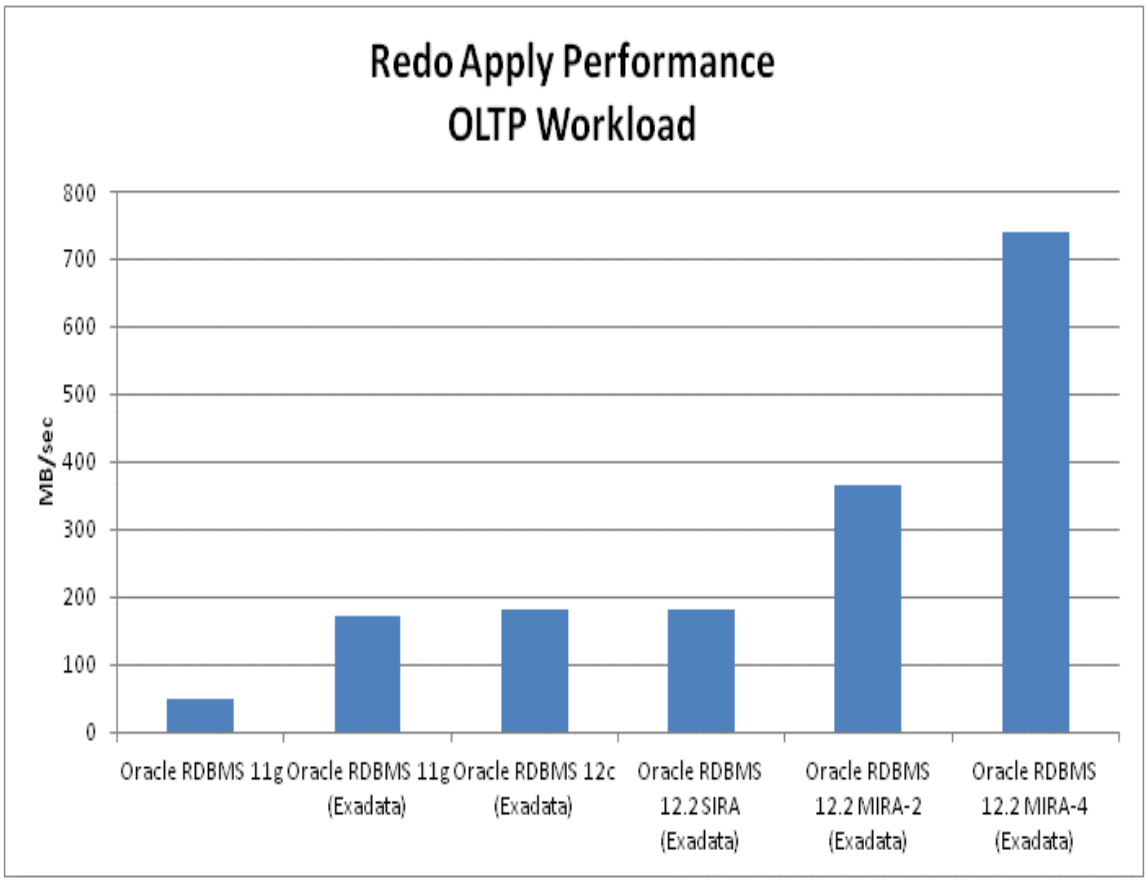

Figure 1: Redo Apply Performance for OLTP Workload

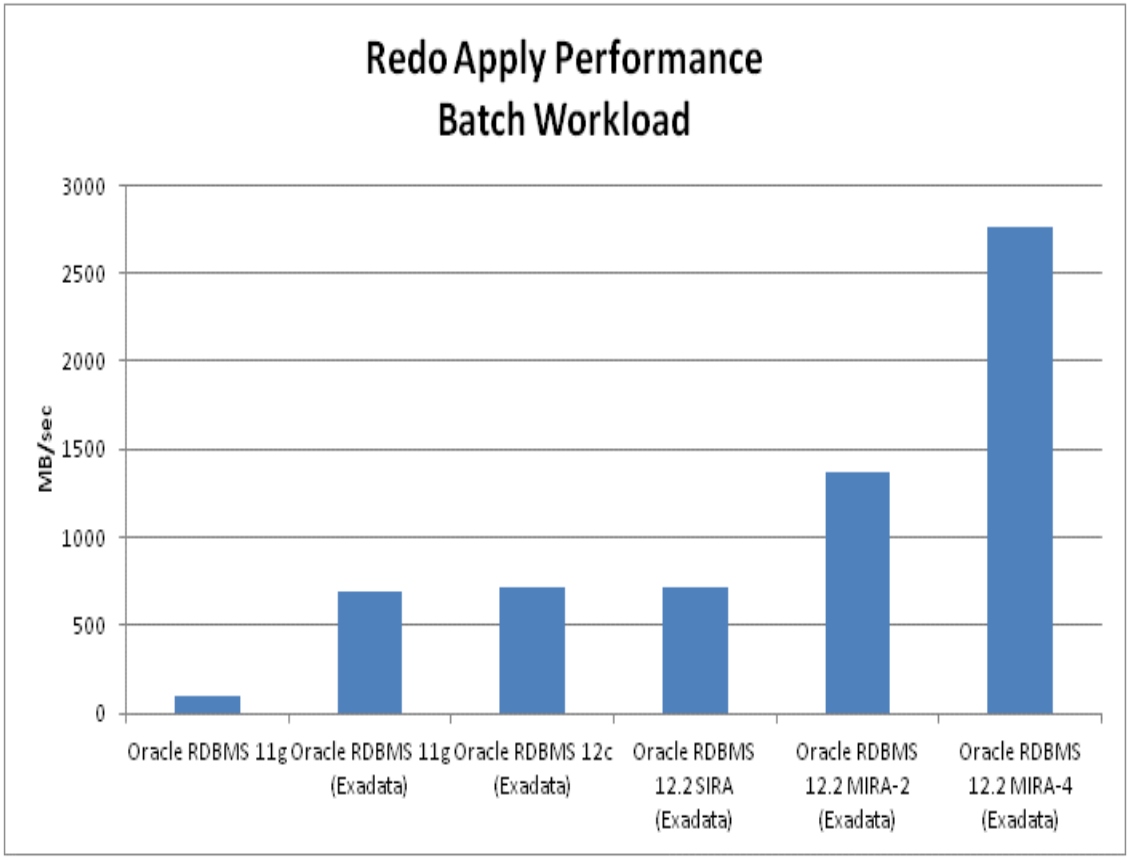

Figure 2: Redo Apply Performance for Batch Workloads

The following sections discuss how to assess the redo apply performance for a given workload and system.

As changes occur on the primary database, redo is generated and sent to the standby database. The frequency of shipping redo to the standby is determined by whether the remote destination is utilizing synchronous or asynchronous transport. If Redo Apply was started using real-time apply, redo generated by the primary database is applied to the standby database as soon as it is received (e.g. there is no wait for the database to switch logs). An Active Data Guard standby database that is open read-only while recovery is active enables users to query current data.

If your standby database is lagging relative to the primary database, you should focus on the following main areas:

- Determining the transport lag
- Determining the apply lag
- Determining the query SCN or time
- Determining how far behind the standby data is compared to the primary database

Redo Transport

Data Guard can transmit redo either synchronously or asynchronously.

- Synchronous transport. When using synchronous transport mode, transactions that commit on the primary database are not acknowledged to the application as committed until all redo generated by that transaction has been received by the standby. While this provides the highest degree of data synchronization between primary and standby databases in Active Data Guard environment, this must be weighed against the potential impact of round-trip network latency on primary database throughput.
- Asynchronous transport. When sending data asynchronously, redo is sent asynchronous with respect to a transaction commit. The primary database will not wait for standby acknowledgment before it acknowledges the commit to the application. While the standby database will not be as up to date as when using synchronous transport, asynchronous transport can offer a very high level of data protection without impacting the primary database performance even with very high network latency.

Determining transport or apply lag is done on the standby database by querying the v$dataguard_stats view using a query similar to the following:

```
select name,value,time_computed,datum_time from v$dataguard_stats where name='%lag';
```

The DATUM_TIME column is the local time on the standby database when the datum used to compute the metric was received. The lag metrics are computed based on data that's periodically received from the primary database. An unchanging value in this column across multiple queries indicates that the standby database is not receiving data from the primary database. The potential data loss in this scenario would be from the last datum time from v$dataguard_stats to the current time on the standby.

To obtain a histogram that shows the history of apply lag values since the standby instance was last started, query the V$STANDBY_EVENT_HISTOGRAM view. For example:

```
select * from v$standby_event_histogram where name like '%lag' and count >0;
```

To evaluate the apply lag over a time period, take a snapshot of V$STANDBY_EVENT_HISTOGRAM at the beginning of the time period and compare that snapshot with one taken at the end of the time period.

```
SQL> col NAME format a10
SQL> select NAME,TIME,UNIT,COUNT,LAST_TIME_UPDATED from V$STANDBY_EVENT_HISTOGRAM
where name like '%lag' and count >0 order by LAST_TIME_UPDATED;

NAME             TIME UNIT                  COUNT LAST_TIME_UPDATED
---------- ---------- ---------------- ---------- --------------------
apply lag          41 seconds                   3 04/05/2018 16:30:59
apply lag          44 seconds                   1 04/05/2018 16:31:02
apply lag          45 seconds                   2 04/05/2018 16:31:03
apply lag          46 seconds                   2 04/05/2018 16:31:04
```

You need to address any transport lag due to insufficient network bandwidth or the overhead of redo encryption or compression first.

Redo Apply Rate

To monitor and assess redo apply performance, query the V$RECOVERY_PROGRESS view. This view contains the columns described in Table 2.

TABLE 2: COLUMNS IN THE V$RECOVERY_PROGRESS VIEW

Column	Description

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.
Maximum Apply Rate	Redo Applied / Active Time is peak measured throughput or maximum rate achieved over a moving average over last 3 minutes. The rate does not include time spent waiting for redo to arrive.
Redo Applied	Represents the total amount of data in bytes that has been applied.
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.
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.
Active Time	Represents the total duration applying the redo , but not waiting for redo
Elapsed Time	Represents the total duration applying the redo , including waiting for redo
Standby Apply Lag	Represents Redo Apply has not applied for N seconds, possible standby is behind the primary.
Log Files	Represents Number of Log files applied so far.

The most useful statistic is the Active Apply rate because the Average Apply Rate includes idle time spent waiting for redo to arrive making it less indicative of apply performance.

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 workloads, 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 Active Apply Rate columns in the `V$RECOVERY_PROGRESS` view to determine if the standby database is able to maintain the pace.

Database Wait Events

Once you have verified that you are not bottlenecked on any system or network resources you are ready to assess database wait events. On the primary database this is done using AWR reports while on the standby database you will use standby statspack(Pre-12.2) or standby AWR(12.2+) reports (See Appendices A & C for complete details on Standby Statspack and Standby AWR).

Prior to assessing database wait events it is important to understand the process flow involved in recovery. In general there are three distinct phases to standby recovery; the log read phase, the apply phase, and the checkpoint phase. The following describes that process flow at a high level:

1. Redo is received on the standby by the RFS (Remote File Server) process. The RFS process writes newly received redo for each thread into the current standby redo log for that thread. The RFS write operation is tracked by the ‘rfs random I/O’ wait event.
2. Once redo has been written the recovery coordinator process (pr00) will read the redo from the standby redo logs for each thread. This read I/O is tracked by the ‘log file sequential read’ operation. The recovery coordinator then merges redo from all threads together and place the redo into memory buffers for the recovery slaves. The wait events for writing and reading into recovery memory buffers is tracked by the ‘parallel recovery read buffer free’ and ‘parallel recovery change buffer free’ wait events.

3. The recovery slaves retrieve redo or change vectors from the memory buffers and begin the process applying the changes to data blocks. First the recovery slaves determine which data blocks need to be recovered and reads those into the buffer cache if it's not already present. This read I/O by the recovery slaves is tracked by the 'recovery read' wait event.
4. When a log is switched on the primary for any thread the standby will coordinate a switch of the standby redo log for that thread at the same time. A log switch on a standby will force a full checkpoint which will result in flushing all dirty buffers from the buffer cache out to the data files on the standby. A checkpoint on the standby is currently more expensive than on a primary. Multiple DB writer processes (DBWR) will write the data file blocks down to the data files with its write time tracked by the 'db file parallel write' wait event. The total time for the checkpoint to complete is covered by the 'checkpoint complete' wait event.

In the description above the log read phase is step 2, the apply phase step 3, while the checkpoint phase consists of step 4. During the apply phase it is normal to see the recovery coordinator process (pr00) with a high utilization on a single CPU while during the checkpoint phase normally will see an increase in the write I/O's to the data files.

Table 3 provides a description as well as tuning advice for wait events involved in the recovery process:

TABLE 3: RECOVERY PROCESS WAIT EVENTS

Column	Description	Tuning Recommendations
Logfile sequential read	The parallel recovery coordinator is waiting on I/O from the online redo log or the archived redo log.	Tune or increase the I/O bandwidth for the ASM diskgroup where the archive logs or online redo logs reside.
Parallel recovery read buffer free	This event indicates that all read buffers are being used by slaves, and usually indicates that the recovery slaves lag behind the coordinator.	Increase "_log_read_buffers" to max 256
Parallel recovery change buffer free	Redo Applied / Active Time is peak measured throughput or maximum rate achieved over a moving average over last 3 minutes. The rate does not include time spent waiting for redo to arrive.	Tune or increase the I/O bandwidth for the ASM diskgroup where data files reside.
Data file init write	The parallel recovery coordinator is waiting for a buffer to be released by a recovery slave. Again, this is a sign the recovery slaves are behind the coordinator.	Tune or increase the I/O bandwidth for the ASM diskgroup where data files reside.
Parallel recovery control message reply	The parallel recovery coordinator is waiting for a file resize to finish, as would occur with file auto extend.	This is a non-tunable event.
Parallel recovery slave next change	The parallel recovery slave is waiting for a change to be shipped from the coordinator. This is in essence an idle event for the recovery slave. To determine the amount of CPU a recovery slave is using, divide the time spent in this event by the number of slaves started and subtract that value from the total elapsed time. This may be close, because there are some waits involved.	N/A. This is an idle event.
DB File Sequential Read	A parallel recovery slave (or serial recovery process) is waiting for a batch of synchronous data block reads to complete.	Tune or increase the I/O bandwidth for the ASM diskgroup where data files reside.
Checkpoint completed	Recovery is waiting for checkpointing to complete, and Redo Apply is not applying any changes currently.	Tune or increase the I/O bandwidth for the ASM diskgroup where data files reside. Also, increase the number of db_writer_processes until the checkpoint completed wait event is lower than the db file parallel write wait event. Consider also increasing the online log file size on the primary and standby to decrease the number of full checkpoints at log switch boundaries.
Recovery read	A parallel recovery slave is waiting for a batched data block I/O.	Tune or increase the I/O bandwidth for the ASM diskgroup where data files reside.
Parallel recovery change buffer free (MIRA)	The parallel recovery coordinator is waiting for a change mapping buffer to be released by one of the recovery slaves.	Increase `_change_vector_buffers` to 2 or 4
Recovery apply pending and/or recovery receive buffer free (MIRA)	Recovery apply pending: the time the logmerger process waited (in centiseconds) for apply slaves to apply all pending changes up to a certain SCN. Recovery receive buffer free: the time (in centiseconds) spent by the receiver process on instance waiting for apply slaves to apply changes from received buffers so that they can be freed for the next change.	Increase `_mira_num_local_buffers` and `_mira_num_receive_buffers` Note: these parameters use space from the shared pool equal to the sum of their values (in MB) * the number of apply instances.

System Resources

To a large degree redo apply performance depends on the system resources allocated for its use. It is critical to monitor system resources such as CPU, memory, and most importantly I/O, to identify any bottlenecks. The following sections provide methods that can be used to monitor system resources.

Monitoring CPU

The uptime, mpstat, sar, dstat, and top utilities allow you to monitor CPU usage. When a system's CPU cores are all occupied executing work for processes, other processes must wait until a CPU core or thread becomes free or the scheduler switches a CPU to run their code. If too many processes are queued too often, this can represent a bottleneck in the performance of the system.

The commands mpstat -P ALL and sar -u -P ALL display CPU usage statistics for each CPU core and averaged across all CPU cores.

The %idle value shows the percentage of time that a CPU was not running system code or process code. If the value of %idle is near 0% most of the time on all CPU cores, the system is CPU-bound for the workload that it is running. The percentage of time spent running system code (%systemor %sys) should not usually exceed 30%, especially if %idle is close to 0%.

The system load average represents the number of processes that are running on CPU cores, waiting to run, or waiting for disk I/O activity to complete averaged over a period of time. On a busy system, the load average reported by uptime or sar -q should not exceed two times the number of CPU cores. If the load average exceeds four times the number of CPU cores for long periods, the system is overloaded.

In addition to load averages (ldavg-*), the sar -q command reports the number of processes currently waiting to run (the `run-queue size`, runq-sz) and the total number of processes (plist_sz). The value of runq-sz also provides an indication of CPU saturation.

Determine the system's average load under normal loads where users and applications do not experience problems with system responsiveness, and then look for deviations from this benchmark over time. A dramatic rise in the load average can indicate a serious performance problem.

Monitoring Memory Usage

The sar -r command reports memory utilization statistics, including %memused, which is the percentage of physical memory in use.

sar -B reports memory paging statistics, including pgscank/s, which is the number of memory pages scanned by the kswapd daemon per second, and pgscand/s, which is the number of memory pages scanned directly per second.

sar -W reports swapping statistics, including pswpin/s and pswpout/s, which are the numbers of pages per second swapped in and out per second.

If %memused is near 100% and the scan rate is continuously over 200 pages per second, the system has a memory shortage.

Once a system runs out of real or physical memory and starts using swap space, its performance deteriorates dramatically. If you run out of swap space, your programs or the entire operating system are likely to crash. If free or top indicate that little swap space remains available, this is also an indication you are running low on memory.

The output from the dmesg command might include notification of any problems with physical memory that were detected at boot time.

Monitoring I/O

The iostat command monitors the loading of block I/O devices by observing the time that the devices are active relative to the average data transfer rates. You can use this information to adjust the system configuration to balance the I/O loading across disks and host adapters.

iostat -x reports extended statistics about block I/O activity at one second intervals, including %util, which is the percentage of CPU time spent handling I/O requests to a device, and avgqu-sz, which is the average queue length of I/O requests that were issued to that device. If %util approaches 100% or avgqu-sz is greater than 1, device saturation is occurring and the storage I/O Bandwidth needs to be augmented by adding disks or storage.

You can also use the sar -d command to report on block I/O activity, including values for %util and avgqu-sz.

The iotop utility can help you identify which processes are responsible for excessive disk I/O. iotop has a similar user interface to top. In its upper section, iotop displays the total disk input and output usage in bytes per second. In its lower section, iotop displays I/O information for each process, including disk input output usage in bytes per second, the percentage of time spent swapping in pages from disk or waiting on I/O, and the command name. Use the left and right arrow keys to change the sort field, and press A to toggle the I/O units between bytes per second and total number of bytes, or O to toggle between displaying all processes or only those processes that are performing I/O.

PERFORMANCE TUNING EXAMPLE

The first step to assess performance and determine steps to improve performance is to take standby statspack snaps that capture statistics on the standby database (see appendix A). For a 12.2 Standby AWR see appendix C. After generating a standby statspack report based off of those snaps the first step is to examine the load profile section.

Load Profile	Total	Per Second
DB time(s):	9.2	0.0
DB CPU(s):	0.7	0.0
Redo MB applied:	**14,497.8**	**9.7**
Logical reads:	471.0	0.3
Physical reads:	6,869,213.0	4,570.3
Physical writes:	9,722,929.0	6,469.0
User calls:	510.0	0.3
Parses:	1,192.0	0.8
Hard parses:	3.0	0.0
W/A MB processed:	35.9	0.0
Logons:	45.0	0.0
Executes:	1,410.0	0.9
Rollbacks:	0.0	0.0

From the load profile we can see the amount of redo recovered during the report period as well as the amount of redo recovered on a per second basis. In addition we can see the amount of physical reads and writes being performed. The amount of reads and writes occurring should be compared to baseline reports that showed acceptable performance. Increased reads could be coming from read only queries that might not have been seen previously, while increased writes could indicate a change in the type of workload being recovered.

Perhaps the most important section to review is the top five wait events. This details where the majority of the time is spent waiting, for example:

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                   wait   Call
Event                                            Waits    Time (s)   (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
checkpoint completed                            4,261       8,210   1927   42.1
db file parallel write                         20,982       2,658    127   13.6
lreg timer                                        501       1,502   2998    7.7
free buffer waits                             119,108       1,278     11    6.5
parallel recovery read buffer free              8,046       1,101    137    5.6
          -------------------------------------------------------------
```

In the above example we see the 42% call time is spent waiting for ‘checkpoint complete’, with the second wait event ‘db file parallel write’ being associated with the checkpoint complete wait as it relates to DBWR performing writes from the buffer cache. The ‘free buffer waits’ wait event indicates that the buffer cache is full and recovery slaves reading buffers into the buffer cache are stalled. The action to take from the wait event profile is that we should increase the number of DBWR processes to help increase the rate that buffers can be flushed from the buffer cache.

Prior to making any changes we should consult the statspack report to make note of the recovery ‘active apply rate’ as well as the time being spent in the apply phase versus the checkpoint phase. Once we adjust the number of DBWR processes we will compare the subsequent standby statspack reports to these numbers.

```
Recovery Start Time Item                 Sofar Units   Redo Timestamp
------------------- ----------------- -------------- ------- ------------------
21-Oct-15 08:03:56  Log Files                     6 Files
21-Oct-15 08:03:56  Active Apply Rate        10,809 KB/sec
21-Oct-15 08:03:56  Average Apply Rat        10,708 KB/sec
21-Oct-15 08:03:56  Maximum Apply Rat        80,592 KB/sec
21-Oct-15 08:03:56  Redo Applied             15,111 Megabyt
21-Oct-15 08:03:56  Last Applied Redo             0 SCN+Tim 20-Oct-15 12:48:25
21-Oct-15 08:03:56  Active Time               1,408 Seconds
21-Oct-15 08:03:56  Apply Time per Lo            121 Seconds
21-Oct-15 08:03:56  Checkpoint Time p             14 Seconds
21-Oct-15 08:03:56  Elapsed Time             1,445 Seconds
21-Oct-15 08:03:56  Standby Apply Lag        70,785 Seconds
          ---------------------------------------------------------------
```

It’s important to make one change at a time and obtain new standby statspack reports to assess any improvement based on that change. For each change follow the same methodology of assessing the physical reads and writes, the top wait events, and the time spent in the different recovery phases.

CONCLUSION

With Oracle Database 11g Release 2 and 12c, 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.

Appendix A - Configure Standby Statspack

Using the Active Data Guard option from Oracle release 11GR1 onwards, statspack from the primary database can be used to collect data from a standby database that is opened read-only. The standby statspack is installed in a separate schema on the primary database which is then propagated to the standby.

Standby Statspack Usage

1. Primary perfstat/statspack Installation. The perfstat schema and statspack related objects must exist on the primary and standby prior to installing standby statspack. If the perfstat user does not exist in the primary site then you will need to create the perfstat schema using the following:

```
$sqlplus / as sysdba
SQL>@?/rdbms/admin/spcreate.sql
```

2. Configure Oracle Net aliases for each standby instance so that instance specific statspack snaps can be obtained.

```
stby =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = scan-name)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ssb)
    )
  )

stby1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = host)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ssb)
      (INSTANCE_NAME = ssb1)
     )
  )

stby2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = host)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ssb)
      (INSTANCE_NAME = ssb2)
    )
  )
```

3. Verify that the standby is opened read only

Note: In order for standby statspack to create the schema and objects and retrieve data from the standby, the standby needs to be open with READ ONLY. For Active Data Guard customers, the physical standby database should be opened READ ONLY and in real time apply. To do this log into the standby host running managed recovery:

Check the current status:

```
SQL> select open_mode ,database_role from v$database;

OPEN_MODE            DATABASE_ROLE

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

READ ONLY WITH APPLY  PHYSICAL STANDBY
```

If you do not see READ ONLY WITH APPLY as open mode, perform the following steps to open the database in read only mode.

```
SQL> recover managed standby database cancel;
```

On each standby host execute the following:

```
SQL> alter database open read only;
```

Check the open mode again:

```
SQL> select open_mode ,database_role from v$database;
```

Then begin the managed recovery on first standby host:

```
SQL> recover managed standby database disconnect using current logfile;
```

4. The standby statspack installation script (sbcreate.sql) creates the standby statspack schema to hold the standby snapshots. The script prompts for the following information:

- A password for stdbyperf user
- Default tablespace
- Temporary tablespace
- The password for the perfstat user
- Oracle Net alias that can be used to reach the standby database. It is recommended to supply the Oracle Net alias that connects to the standby instance that normally performs managed recovery

Login to the primary database and execute the following:

```
SQL>@?/rdbms/admin/sbcreate.sql
```

5. In addition to monitoring the recovery on a particular instance, you can also monitor performance on the other instance running reports. In order to do this, you need create the standby PL/SQL package for each instance in the standby configuration. For the remaining instances in the standby configuration run the sbaddins.sql for those instances. Login to the primary database as the stdbyperf user and execute the following:

```
SQL> >@?/rdbms/admin/sbaddins.sql
```

6. Taking standby statspack snaps for standby instances is done using the PL/SQL package for each standby instance created in the above steps. The statspack_<instance_name>.snap procedure accesses the data dictionary and stats$ views on the standby database via database link connected to the original perfstat user and stores the data to stats$tables on the primary instance. For example, while the standby is opened read only, login to the primary database and create the snap:

```
SQL> connect stdbyperf/your_password
SQL> exec statspack_<db_unique_name>_<instance_name>.snap
```

7. The script sbreport.sql generates the standby statistics report. The script asks for the following information to generate the report: database ID, instance number, begin and end snapshots ids. For example, login to the primary database and execute the following:

```
SQL>@?/rdbms/admin/sbreport
```

8. The script sbpurge.sql is used to purge a set of snapshots. The script asks for database id, instance number, begin and end snapshots ids. The script purges all snapshots between the begin and end snapshot ids for the given standby instance. For example, login to the primary database and execute the following:

```
SQL>@?/rdbms/admin/sbpurge
```

9. The script sbdelins.sql deletes an instance from the configuration, and deletes the associated PL SQL package. The scripts asks for instance name. The snapshots are not automatically purged when the instance is deleted. After deleting the instance, you are not able to generate reports for that instance. For example, login to the primary database and execute the following:

```
SQL> @?/rdbms/admin/sbdelins
```

10. The script sbdrop.sql drops the stdbyperf user and tables. The script must be run when connected to SYS (or internal). For example, login to the primary database and execute the following:

```
SQL> connect / as sysdba
SQL> @?/rdbms/admin/sbdrop
```

Appendix B – Generate Active Session History Report

Real-time stats can be collected on an Active Data Guard Standby database using in-memory ASH. ASH reports provide analysis of transient performance problems that typically last for a few minutes. ASH also performs scoped or targeted performance analysis by various dimensions or their combinations, such as time, session, module, action, or SQL Identifier. For more information please refer to the ASH documentation[1].

To generate an ASH report on a physical standby instance, the standby database must be opened read-only. The ASH data on disk represents activity on the primary database and the ASH data in memory represents activity on the standby database. You must specify whether to generate the report using data sampled from the primary or standby database.

The ashrpt.sql (Single Instance) and ashrpti.sql (RAC) SQL scripts generate an HTML or text report that displays ASH information for a specified duration on a specified database and instance. This ashrpti.sql script enables you to specify a database and instance for which the ASH report will be generated.

```
SQL> @?/rdbms/admin/ashrpti.sql
```

The following is output expected from running the above script:

```
You are running ASH report on a Standby database. To generate the report
over data sampled on the Primary database, enter 'P'.
Defaults to 'S' - data sampled in the Standby database.
Enter value for stdbyflag: S
Using Primary (P) or Standby (S): S
ASH Samples in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Oldest ASH sample available:  08-May-15 12:28:33   [      33 mins in the
past]
Latest ASH sample available:  08-May-15 13:01:18   [       0 mins in the
past]
Specify the timeframe to generate the ASH report
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter begin time for report:
Enter value for begin_time: -30
Report begin time specified: -30
Report duration specified:
Using 08-May-15 12:33:46 as report begin time
Using 08-May-15 13:01:49 as report end time
Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is ashrpt_rac_0508_1301.html.  To use this
name,
press <return> to continue, otherwise enter an alternative.
Enter value for report_name: ashrpt_rac_0508_1301.html
SQL>
```

After generating an ASH report, analyze its contents to identify possible causes of transient performance problems. The ASH report is divided into sections such as Top Events, Load Profile, Top Sessions and Top SQL etc. The Activity over time section is one of the most informative sections of the ASH report. This section is particularly useful

[1] http://docs.oracle.com/database/121/RACAD/monitor.htm#RACAD987

for analyzing longer time periods because it provides in-depth details about activities and workload profiles during the analysis period.

Appendix C – Automated Workload Repository for Standby

Starting with Oracle Database 12.2, the Automated Workload Repository (AWR) can be configured to take snapshots of Active Data Guard standby databases. The benefits of Standby AWR over Statspack are many, including Oracle RAC-wide reports for Oracle RAC databases, Exadata metrics and HTML output. The Maximum Availability Architecture group recommends Standby AWR for monitoring of standby databases at Oracle Database version 12.2 and later.

Standby AWR is available for standalone databases and CDB level snapshots in multitenant databases. PDB level detail is not available as of the release of this paper.

For full details about Standby AWR and the Remote Management Framework consult the *Oracle Database Performance Tuning Guide*.

Configuration

In a Standby AWR configuration the primary database is known as the destination database all standby databases are known as the sources databases. The primary is also a source database as it can have snapshots taken of it.

Appendix D contains a script which can be modified to configure a 1:1 primary to standby Data Guard configuration.

Usage

Below are a couple of common operations to control snapshots.

Change automatic snapshot interval:

```
exec DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS( retention => <#Minutes>,
interval  => <#Minutes>, topnsql   => <#sqls>, dbid      => <dbid>);
```

Take a manual snapshot:

```
exec  DBMS_WORKLOAD_REPOSITORY.CREATE_REMOTE_SNAPSHOT(node_name=>'<node name>',
topology_name=>'<topology name>');
```

Reporting

As with all AWR reporting, a report can be generated by running the Oracle provided SQL scripts in $ORACLE_HOME/rdbms/admin. These can each be run interactively or in a script by setting the appropriate variables. The necessary variables can be gleaned from the header of each script and are listed below.

awrrpti.sql generates a report for a specific instance

```
define  inst_num     = 1;
define  num_days     = 3;
define  inst_name    = 'Instance';
```

```
define  db_name      = 'Database';
define  dbid         = 4;
define  begin_snap   = 10;
define  end_snap     = 11;
define  report_type  = 'text';
define  report_name  = /tmp/swrf_report_10_11.txt
@@?/rdbms/admin/awrrpti
```

awrgrpti.sql for a specific Oracle RAC database(all instances)

```
define  num_days     = 3;
define  db_name      = 'Database';
define  dbid         = 4;
define  begin_snap   = 10;
define  end_snap     = 11;
define  report_type  = html;
define  report_name  = /tmp/awrrac1.html
define  instance_numbers_or_ALL = '1,2,3'
@@?/rdbms/admin/awrgrpti
```

Analysis

Diagnosing issues with redo apply is best done by analyzing the single instance AWR report (awrrpti.sql) from the apply instance. Similar strategies of the performance tuning example using standby statspack can be applied with standby AWR reports. Below are some critical sections

The Load profile in the figure below provides information about the workload of the instance. The I/O profile and Redo Apply rate can be gleaned from this section.

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	2.7	0.2	0.01	0.04
DB CPU(s):	0.3	0.0	0.00	0.00
Background CPU(s):	1.8	0.1	0.01	0.00
Redo size (bytes):	0.0	0.0		
Logical read (blocks):	6,996.3	509.6		
Block changes:	0.0	0.0		
Physical read (blocks):	72.5	5.3		
Physical write (blocks):	5,116.8	372.7		
Read IO requests:	72.5	5.3		
Write IO requests:	1,599.4	116.5		
Read IO (MB):	0.6	0.0		
Write IO (MB):	40.0	2.9		
IM scan rows:	0.0	0.0		
Session Logical Read IM:	0.0	0.0		
Global Cache blocks received:	0.0	0.0		
Global Cache blocks served:	0.6	0.0		
User calls:	74.0	5.4		
Parses (SQL):	182.4	13.3		
Hard parses (SQL):	0.0	0.0		
SQL Work Area (MB):	0.2	0.0		
Logons:	0.1	0.0		
Executes (SQL):	241.3	17.6		
Rollbacks:	13.7	1.0		
Transactions:	13.7			
Redo MB applied:	19,493.1	1,420.0		

Recovery progress statistics can also provide useful information for the active apply rate at the last snapshot as well as checkpoint time per log which could identify an I/O issue, as shown below.

Recovery Progress Statistics

- Only rows for Media Recovery at end snapshot are displayed
- Ordered by Start Time desc, Item, Total desc

Recovery Start Time	Item	Sofar	Units	Redo Timestamp
02/21 11:34:44	Active Apply Rate	20,520	KB/sec	
02/21 11:34:44	Active Time	4,716	Seconds	
02/21 11:34:44	Apply Time per Log	71	Seconds	
02/21 11:34:44	Average Apply Rate	114	KB/sec	
02/21 11:34:44	Checkpoint Time per Log	0	Seconds	
02/21 11:34:44	Elapsed Time	76,027	Seconds	
02/21 11:34:44	Last Applied Redo	0	SCN+Time	02/22 08:41:51
02/21 11:34:44	Maximum Apply Rate	49,775	KB/sec	
02/21 11:34:44	Recovery ID	0	RCVID	
02/21 11:34:44	Redo Applied	8,507	Megabytes	
02/21 11:34:44	Standby Apply Lag	2	Seconds	
02/21 11:34:27	Recovery ID	0	RCVID	

The top 10 waits section, shown below, can be a bit misleading for analysis purposes because unlike the top 5 waits from standby statspack, it does not include the background waits which are often the culprits for poor redo apply performance.

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Avg Wait	% DB time	Wait Class
Data Guard server operation completion	5,950	226.9	38.13ms	66.3	Other
DB CPU		42.9		12.5	
RFS write	24,832	17.7	711.11us	5.2	System I/O
SQL*Net vector data from client	24,916	4.1	163.55us	1.2	Network
cell single block physical read	1,732	3.5	2.04ms	1.0	User I/O
RFS dispatch	54	1.9	34.97ms	.6	Other
control file parallel write	402	1.8	4.58ms	.5	System I/O
control file sequential read	890	1	1.12ms	.3	System I/O
enq: CF - contention	19	.7	34.62ms	.2	Other
log file sequential read	130	.6	4.44ms	.2	System I/O

Using the Background Wait Events, as shown below, with a focus on the **% bg time** column helps to identify the potential bottlenecks in redo apply. In the output below, based on the checkpoint time per log from the Recovery Progress statistics and the checkpoint completed time being the top background wait, it is possible that we are switching logs too frequently, and the size of the logs needs to be increased.

Background Wait Events

- ordered by wait time desc, waits desc (idle events last)
- Only events with Total Wait Time (s) >= .001 are shown
- %Timeouts: value of 0 indicates value was < .5%. Value of null is truly 0

Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait	Waits /txn	% bg time
checkpoint completed	5,875	0	295	50.30ms	3.32	44.12
log file sequential read	9,694	0	33	3.44ms	5.47	4.99
control file sequential read	11,985	0	12	.97ms	6.76	1.74
reliable message	10,525	0	8	756.96us	5.94	1.19
recovery read	4,326	0	5	1.25ms	2.44	0.81

This conclusion can be confirmed by the instance activity stats, shown below, which estimates over 300 log switches per hour with this workload. The target is 3-6 log switches per hour to limit the checkpoints, which hinders redo apply.

Instance Activity Stats - Thread Activity

- Statistics identified by '(derived)' come from sources other than SYSSTAT

Statistic	Total	per Hour
log switches (derived)	12	334.68

Appendix D – Standby Automated Workload Repository Configuration Script

The following script can be used to configure Standby Automated Workload Repository (AWR) for a basic primary with one standby Data Guard configuration. Paste these contents into a script, modify the variables, and execute it from the primary database environment. The primary and standby TNS descriptors should already be configured for redo transport. For Oracle RAC these descriptors should use the SCAN listener.

References to 'primary' and 'standby' are that of the initial configuration.

```
ptns=<primary database tns descriptor>
stns=<standby database tns descriptor>
sysumf_passwd=<desired password for sys$umf user>
primary_name=<primary database unique name>
standby_name=<standby database unique name>
stby2pri_link=<Database link name for standby to primary>
pri2stby_link=<Database link name for primary to standby>
topology_alias=<alias for the topology/configuration>


echo "1. Create node"
echo "1.1 Create target node name at $ptns"
sqlplus /nolog <<EOF
   spool crumf_pnode.log
   connect sys/welcome1@$ptns as sysdba
   alter system set "_umf_remote_enabled"=TRUE scope=BOTH;
   alter user sys\$umf account unlock identified by $sysumf_passwd;

   drop database link $stby2pri_link;
   create database link $stby2pri_link connect to sys\$umf identified by
$sysumf_passwd  using '$ptns';

   exec dbms_umf.unconfigure_node;
   exec dbms_umf.configure_node('$primary_name');
   select dbms_umf.get_node_name_local as node_name from dual;
EOF

grep ORA-  crumf_pnode.log | grep -v ORA-02024

if [ $? -eq 0 ]
then
   echo "Error: Failed to create target node at $ptns."
   echo "Please check crumf_pnode.log for more detail"
```

```
   exit 2
fi

echo "1.2 Create source node name at $stns"
sqlplus /nolog <<EOF
   spool crumf_snode.log
   connect sys/welcome1@$ptns as sysdba
   drop database link $pri2stby_link;
   create database link $pri2stby_link connect to sys\$umf identified by
$sysumf_passwd using '$stns';

   connect sys/welcome1@$stns as sysdba
   alter system set "_umf_remote_enabled"=TRUE scope=BOTH;
   exec dbms_umf.unconfigure_node;
   exec dbms_umf.configure_node('$standby_name');
   select dbms_umf.get_node_name_local as node_name from dual;
EOF

grep ORA-  crumf_snode.log | grep -v ORA-02024

if [ $? -eq 0 ]
then
   echo "Error: Failed to create source node at $stns."
   echo "Please check crumf_snode.log for more detail"
   exit 2
fi

echo "2. Create topology"

sqlplus /nolog <<EOF
   spool crumf_topology.log
   connect sys/welcome1@$ptns as sysdba
   exec dbms_umf.drop_topology('$topology_alias');
   exec dbms_umf.create_topology('$topology_alias');

   -- Query the topology XML and X$
   select * from umf\$_topology_xml;
   select * from x\$keumtoptb;

   alter system archive log current;
EOF

grep ORA-  crumf_topology.log | grep -v ORA-20507 | grep -v ORA-06512| grep -v ORA-
15767

if [ $? -eq 0 ]
then
   echo "Error: Failed to create topology at $ptns."
   echo "Please check crumf_topology.log for more detail"
   exit 2
fi


sleep 5
echo "3. Register remote node at $stns"
sqlplus /nolog <<EOF
   spool crumf_regremotenode.log
   connect sys/welcome1@$ptns as sysdba
   set echo on;
   exec dbms_umf.register_node('$topology_alias', '$standby_name', '$pri2stby_link',
'$stby2pri_link');
```

```
EOF

grep ORA-  crumf_regremotenode.log

if [ $? -eq 0 ]
then
   echo "Error: Failed to register remote node $stns at $ptns."
   echo "Please check crumf_regremotenode.log for more detail"
   exit 2
fi

sleep 10

echo "4. Register remote database at $stns"
sqlplus /nolog <<EOF
   spool crumf_regremotedb.log
   connect sys/welcome1@$ptns as sysdba
   exec dbms_workload_repository.register_remote_database('$standby_name');
EOF

grep ORA-  crumf_regremotedb.log

if [ $? -eq 0 ]
then
   echo "Error: Failed to register remote database $stns at $ptns."
   echo "Please check crumf_regremotedb.log for more detail"
   exit 2
fi

echo "5. Verify setup"
sqlplus /nolog <<EOF
   spool crumf_verify.log
   connect sys/welcome1@$ptns as sysdba

   select TOPOLOGY_NAME, NODE_NAME, NODE_ID, NODE_TYPE from  umf\$_registration;
   select * from DBA_UMF_REGISTRATION;

   -- Query x$'s.
   select count(*) from x\$keumtoptb;
   select count(*) from x\$keumregtb;
   select count(*) from x\$keumlinktb;
   select sysdate from dual@$pri2stby_link;
   select dbid from v\$DATABASE@$stby2pri_link;
   select * from sys.umf\$_topology@$stby2pri_link;
   select sysdate from dual@$stby2pri_link;
   select dbid from v\$DATABASE@$stby2pri_link;
   select * from sys.umf\$_topology@$stby2pri_link;

   -- Execute UMF/PLSQL API to query.
declare
  topology_name VARCHAR2(128);
  my_node_id    NUMBER;
  link_name     VARCHAR2(128);
  tid           NUMBER;
begin
  -- Query local node registration info. This will also sync.
  dbms_umf.query_node_info(NULL, '$standby_name', my_node_id);
  dbms_output.put_line('my_node_id='||my_node_id);
  -- Get the target id.
  select target_id into tid from umf\$_topology_xml where topology_name =
'$topology_alias';
  dbms_output.put_line('tid='||tid);
```

```
  -- Query the link info.
  dbms_umf.query_link_info('$topology_alias',my_node_id, tid , link_name);
  dbms_output.put_line('link_name='||link_name);
end;
/

select target_id  from umf\$_topology_xml where topology_name = '$topology_alias';
EOF
```

Appendix E - Additional Tuning Opportunities

For those trying to squeeze every bit of performance out of redo apply, below are some additional tuning opportunities that are not preferred, but can help achieve higher apply rates.

Set DB_CACHE_SIZE to a Value Greater than on the Primary Database

Having a large database cache size can improve media recovery performance by reducing the amount of physical data block reads. Because media recovery does not require `DB_KEEP_CACHE_SIZE` and `DB_RECYCLE_CACHE_SIZE`, or require a large `SHARED_POOL_SIZE`, the memory can be reallocated to the `DB_CACHE_SIZE`. Set `DB_CACHE_SIZE` to a value greater than that for the primary database. Set `DB_KEEP_CACHE_SIZE and DB_RECYCLE_CACHE_SIZE` to 0.

Note: Before converting the standby database into a primary database, reset these parameters to the primary database settings.

Set DB_BLOCK_CHECKING=FALSE (not recommended)

It is rare that redo apply performance is unable to keep pace with an application's redo generation rate. If, however, the apply rate is unable to keep pace with redo generation, you can temporarily disable block checking at the standby database by the setting `DB_BLOCK_CHECKING=FALSE`. Changing the setting to `FALSE` can provide as much as a 2x increase in the apply rate. While not ideal, it is acceptable to forego `DB_BLOCK_CHECKING` on a standby database given other validation performed by Data Guard and assuming that block checking remains enabled on the production database. Block checking has much less impact on production database workloads than it has on media recovery at a standby database.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services

Redo Apply Best Practices
Updated August 2018
Author: Michael Smith and Andy Steinorth

Oracle is committed to developing practices and products that help protect the environment

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com