Data Guard Redo Transport & Network Best Practices
Oracle Database 10*g* Release 2

Oracle Maximum Availability Architecture White Paper
February 2007

Maximum
Availability
Architecture

Oracle Best Practices for High Availability

Data Guard Redo Transport and Network Best Practices Oracle Database 10*g* Release 2

Data Guard Redo Transport and Network Best Practices Oracle Database 10g Release 2

INTRODUCTION

There are two areas to consider when tuning the performance of an Oracle Data Guard configuration [1]. The first is Redo Transport Services that transmit redo data generated by a production database to a local or remote standby database. The second is Log Apply Services that applies redo data to a standby database. This Maximum Availability Architecture (MAA) [2] paper focuses on best practices to optimize the performance of Redo Transport Services. Additional MAA papers available on the Oracle Technology Network (OTN) provide best practices for Log Apply Services for a physical standby database using Redo Apply [3], and for a logical standby database using SQL Apply [4].

This paper also provides results from Oracle's MAA test lab showing performance improvements that can be achieved with a properly tuned configuration. For readers familiar with previous Data Guard releases, the paper also describes redo transport enhancements delivered with Oracle Data Guard 10g Release 2 (10.2.0.2).

Note: This paper assumes that the reader is familiar with Oracle Data Guard Redo Transport Services and has read Oracle Data Guard Concepts and Administration [5].

EXECUTIVE SUMMARY

Data Guard 10g Release 2 significantly improves Recovery Point and Recovery Time Objectives across a wide range of network environments and application workload. Tests conducted by the Oracle Maximum Availability Architecture (MAA) development team discussed further in this paper demonstrate the following benefits:

Minimal Impact on Production Database Performance

The following are realistic scenarios for configurations with sufficient network bandwidth based upon MAA test results:

- A Data Guard environment between New York and Montreal (up to 330 miles apart with 10ms RTT latency) using synchronous transport mode can provide zero data loss with minimal performance impact (less than 5%) for production databases generating redo data at rates up to 4MB/sec.
- A Data Guard environment between Boston and London (up to 3,300 miles apart with 100ms RTT) using asynchronous transport mode can have less than 5% impact on production databases generating redo data at rates up to 20MB/sec.

Highest Level of Data Protection

Minimal impact on database performance means users can use Data Guard to significantly enhance their level of data protection.

- Synchronous, zero data loss protection can be utilized for a wide range of application workloads and for network topologies having distances between production and standby sites ranging up to hundreds of miles.
- Asynchronous transport mode can be used for Data Guard environments that span greater distances and having higher network latencies. For example, at redo rates of 2MB/sec (typical of many high-throughput applications) asynchronous transport mode when used between Hong Kong and Singapore can achieve less than 3 seconds of total data loss exposure (up to 1,600 miles apart with 50ms RTT).
- Enhancements to ARCH redo transport have accelerated the speed at which production and standby databases can be resynchronized following an extended network or standby system outage. For example, the time required to ship a 1GB log file using ARCH has been reduced up to 55%. By rapidly resynchronizing production and standby databases, critical data is quickly returned to a protected state.

Faster Switchover & Failover

- Data Guard 10g Release 2 enhancements to asynchronous mode reduce the time required to ship and apply data to a standby database. Because the standby database is more closely synchronized with the production database, role transitions that promote a standby database to the production role can be completed in seconds [6].

The remainder of this paper describes in detail Data Guard Redo Transport and Network Best Practices that will help you achieve the highest level of data protection for your network and system environment.

REDO TRANSPORT BEST PRACTICES

Data Guard provides three methods of Redo Transport: `ARCH`, `LGWR ASYNC`, and `LGWR SYNC` [5]. The following sections describe considerations for determining network requirements and configuration best practices that apply to each method of redo transport.

How Much Bandwidth is Enough?

The goal for all Data Guard configurations is to ship redo data to the remote disaster recovery site fast enough to meet recovery time and recovery point objectives. No amount of tuning can achieve this goal if there is insufficient bandwidth available to handle the required volume. To answer to the question of how much bandwidth is needed, start by projecting the redo volume that will be generated by your production database. Ideally, there is either an existing production application or an application running in a test environment where this volume can be measured.

The simplest way to determine application throughput in terms of redo volume is to collect AWR reports during normal and peak workload and determine the number of bytes per second of redo data the production database is producing. For example, if the application is producing 3MB/sec of redo data during peak periods, the network link between the primary and standby databases should be able to transmit a minimum of 3MB/sec of network bandwidth. Network bandwidth is described in terms of Megabits/second, (Mbps). Doing this conversion shows that 3MB/sec = 25.2Mbps of network throughput (1048576 bytes in 1MB, 8 bits in 1 byte and 1 million bits in a megabit, equals 25.2Mbps). This volume is beyond the capacity of a T1/DS1 link (bandwidth of 1.544 Mbps), and into the range of a T3/DS-3 link (bandwidth of 44.7Mbps)

You must also consider various characteristics of your network and the underlying TCP/IP protocol that will influence the actual throughput that can be achieved. These include the overhead caused by network acknowledgements, network latency, and other factors. Their impact will be unique to your workload and network, and will reduce the actual network throughput that you will be able to achieve.

Note: Ordinarily, one expects to see a direct correlation between latency and distance, a general rule of thumb being 1ms RTT per 33 miles (53 KM). This ratio is a rule of thumb and will vary depending on network configuration. Distance (propagation delay) does affect the latency, which in turn affects the network RTT. However, other factors also impact the network RTT; such as number of repeaters, network traffic, poor systems performance and network congestion. Therefore, rather than give a ratio of network latency to distance, MAA test results presented in this paper include the network RTT. The network RTT can easily be measured by using the 'traceroute' utility and the network RTT metric can be a basis of comparison with other environments.

One further consideration in determining bandwidth requirements is the Data Guard protection mode and transport service utilized. For example, Data Guard `LGWR SYNC` synchronous transport will impact production database performance if there is insufficient bandwidth available to handle redo data generation rates (this is typical of any synchronous transport method). The Data Guard `LGWR ASYNC` transport, in contrast, uses online redo logs to buffer peaks in redo generation that may temporarily exceed the maximum network throughput that can be achieved without impacting production database performance. The Data Guard `ARCH` transport produces a similar result by using local archive logs as a buffer (although with higher potential data loss exposure than `LGWR ASYNC`).

Oracle recommends that users conduct an initial assessment of redo volume and network bandwidth as described in this paper. Use the data obtained as a starting place from which to test workload in an environment that is representative of the expected production environment (system and network resources, network latency, and Data Guard protection mode). Insure that your tests implement the tuning recommendations and best practices detailed in this paper

Once the bandwidth required between the primary and standby databases has been determined, implement the following best practices for each type of transport:

ARCH Redo Transport

- Consider increasing the number of ARC*n* processes. The default number of ARC*n* processes created when the database is 2. The LOG_ARCHIVE_MAX_PROCESSES initialization parameter controls the maximum number of ARC*n* processes. This parameter can be set as high as 30 in Oracle Database 10g Release 2. The maximum value in previous releases was 10.

  The larger number of ARC*n* processes make it possible to quickly resolve archive log gaps that can occur during extended network or standby database outages. A large number will also provide enough ARC*n* processes to support remote archiving parallelism, which can be enabled using the MAX_CONNECTIONS attribute discussed in the next section.

  Note: Setting LOG_ARCHIVE_MAX_PROCESSES to a high value may increase contention with other application that use the same network resources. Hence, you should consider the impact on other applications when determining the optimal value for LOG_ARCHIVE_MAX_PROCESSES. Determining the optimal value can only be achieved through testing large archive log gap scenarios in your environment.

- Set MAX_CONNECTIONS attribute to 2 or higher (on the LOG_ARCHIVE_DEST_*n* initialization parameter) for all destinations. Doing so enables remote parallel archiving which can significantly reduce the overall time needed to transfer an archive log. The maximum value for MAX_CONNECTIONS is 5.

LGWR ASYNC Redo Transport

- Beginning with Oracle Database 10g Release 2, be sure to allow for sufficient I/O bandwidth for LNS read I/Os to online redo logs on the production database. The number of additional reads will vary between workloads and should be assessed through performance testing.

LGWR SYNC Redo Transport

- Reduce the value of the NET_TIMEOUT attribute to decrease the impact of network outages on production database performance. This attribute specifies the number of seconds that LGWR on the production database waits for Oracle Net Services to respond to a LGWR request.

In Oracle Database 10g Release 2 the default NET_TIMEOUT value is 180 seconds [5]. Although a minimum value of 1 second is allowed, Oracle recommends 10 seconds as a minimum to avoid disconnecting from the standby database unnecessarily.

Note: Users are often tempted to set this value lower than 10 seconds. It is very important to understand your environment in determining an optimum value before setting it so low. For example, if you set it to 2 seconds, and you have a network with frequent brownouts that can last as long as 3 seconds, this will result in frequent timeouts and impact data protection levels.

- When using LGWR SYNC commits are not returned to the foreground until the redo for that transaction has been written locally on the primary and remotely on the standby. Optionally, the parameter COMMIT NOWAIT can be used so that commits are returned to the application without waiting for redo to be written to disk. Therefore, applications or transactions that can utilize COMMIT NOWAIT will have a significant improvement in response time and database throughput over applications or transactions that utilize the default COMMIT WAIT behavior. For more information refer to Appendix C of this paper.

All Redo Transports

- Tune standby redo logs for efficient I/O. Doing so prevents the RFS process writes on the standby from slowing down sending processes (LNS and ARCH) on the production database. RFS is the Data Guard process on the standby database that receives redo and writes it to the standby redo log.
- Typical areas in which to tune the standby I/O subsystem can include:
  - Configure the system such that 1 MB I/Os are issued to the storage array as a single I/O request. For example, for information on Linux configurations refer to Best Practices for Creating a Low-Cost Storage Grid for Oracle Databases [7].
  - Ensure that standby redo logs are properly placed within the fast disk group.
  - Do not multiplex standby redo logs. Remove additional standby redo log members to prevent additional writes.

NETWORK TUNING BEST PRACTICES

This section describes network tuning applicable to all redo transport methods that has the potential to significantly enhance Data Guard redo transport performance. Several examples of performance gains measured during Oracle MAA testing are provided later in this paper. The sample commands shown in the following sections are for Linux and, as a result, may vary on different platforms. Details are discussed in the following sections, but in general, network tuning involves:

- Ensure bandwidth is sufficient for the volume of redo data to be shipped to the standby location
- Set the Oracle Net RECV_BUF_SIZE and SEND_BUF_SIZE parameters equal to 3 times the *Bandwidth Delay Product* (BDP). This will produce the largest increase in network throughput.
- Use an Oracle Net Session Data Unit (SDU) size of 32767.
- Increase the default send and receive queue sizes associated with networking devices. For Linux increase the TXQUEUELENGTH interface option via the ifconfig command on the sending side, and the NET_DEV_MAX_BACKLOG kernel parameter on the receiving side. As a proactive measure to prepare for future role transitions it is helpful to change both parameters on the production and all standby databases.
- Ensure that the Oracle Net TCP_NODELAY parameter is set to YES (which is the default value).

Oracle Net Session Data Unit (SDU) Size

When sending data across the network, Oracle Net buffers data into session data unit (SDU) sized units. When large amounts of data are being transmitted or when the message size is consistent, increasing the size of the SDU buffer can improve performance and network utilization. You can configure SDU size within an Oracle Net connect descriptor or globally within the sqlnet.ora [8].

For Data Guard broker configurations configure the DEFAULT_SDU_SIZE parameter in the sqlnet.ora file:

```
DEFAULT_SDU_SIZE=32767
```

For non Data Guard broker configurations that use a connect descriptor, you can override the current settings in the primary database sqlnet.ora file. When setting SDU in a connect descriptor you must use a static SID, as dynamic service registration will use the default SDU size defined by DEFAULT_SDU_SIZE. In a connect descriptor, you specify the SDU parameter in a description list.

```
sales.us.acme.com=
(DESCRIPTION=
    (SDU=32767)
      (ADDRESS=(PROTOCOL=tcp)
      (HOST=sales-server)
      (PORT=1521))
    (CONNECT_DATA=
     (SID=sales.us.acme.com))
)
```

On the standby database, set SDU in the SID_LIST of the listener.ora file:

```
SID_LIST_listener_name=
(SID_LIST=
 (SID_DESC=
  (SDU=32767)
  (GLOBAL_DBNAME=sales.us.acme.com)
  (SID_NAME=sales)
  (ORACLE_HOME=/usr/oracle)))
```

TCP Socket Buffer Sizes

TCP socket buffer settings will control how much network bandwidth can be used regardless of the bandwidth available in the network circuit. Socket buffer sizes need to be increased from their default values in order to improve utilization of available bandwidth. When network latency is high, larger socket buffer sizes are needed to fully utilize network bandwidth.

The optimal socket buffer size is three times the size of the *Bandwidth Delay Product* (BDP). To compute the BDP, the bandwidth of the link and the network Round Trip Time (RTT) are required. RTT is the time required for a network communication to travel from the production database to the standby and back and is measured in milliseconds (ms). The following example assumes a gigabit network link with a RTT of 25 ms:

BDP= 1,000 Mbps * 25msec (.025 sec)

1,000,000,000 * .025

25,000,000 Megabits / 8 = 3,125,000 bytes

Given this example, the optimal send and receive socket buffer sizes are calculated as follows:

socket buffer size = 3 * bandwidth * delay

= 3,125,000 * 3

= 9,375,000 bytes

The size of the socket buffers can be set at the operating system level or at the Oracle Net level.   As socket buffer size requirements can become quite large (depending on network conditions) it is recommended to set them at the Oracle Net level [8] so that normal TCP sessions, such as telnet, do not use additional memory. Please note that some operating systems have parameters that set the maximum size for all send and receive socket buffers. You must ensure that these values have been adjusted to allow Oracle Net to use a larger socket buffer size.

For Data Guard broker or Enterprise Manager configurations configure the `sqlnet.ora` file to reflect the desired send and receive buffer sizes. For example:

```
RECV_BUF_SIZE=9375000
SEND_BUF_SIZE=9375000
```

For non Data Guard broker configurations that use a connect descriptor, you either specify the buffer space parameters for a particular protocol address or description.

The following example shows the send and receive socket buffer size being set as a description attribute for a particular connect descriptor in the client-side sqlnet.ora file:

```
standby =
 (DESCRIPTION=
   (SEND_BUF_SIZE=9375000)
   (RECV_BUF_SIZE=9375000)
     (ADDRESS=(PROTOCOL=tcp)
     (HOST=hr1-server)(PORT=1521))
   (CONNECT_DATA=
     (SERVICE_NAME=standby)))
```

The socket buffer sizes must be configured at all sites within a Data Guard configuration. On a standby database this can be accomplished within either the `sqlnet.ora` or `listener.ora` file.

In the `listener.ora` file, you can either specify the buffer space parameters for a particular protocol address or for a description.

```
LISTENER=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)
    (HOST=sales-server)(PORT=1521)
    (SEND_BUF_SIZE=9375000)
    (RECV_BUF_SIZE=9375000)))
```

Network Device Queue Sizes

You can regulate the size of the queue between the kernel network subsystems and the driver for network interface card. Any queue should be sized so that losses do not occur due to local buffer overflows. Therefore, careful tuning is required to ensure that the sizes of the queues are optimal for your network connection, particularly for high bandwidth networks.

These settings are especially important for TCP, because losses on local queues cause TCP to fall into congestion control, which limits the TCP sending rates.

For Linux there are two queues to consider, the interface transmit queue and the network receive queue. The transmit queue size is configured with the network interface option `txqueuelen`. The network receive queue size is configured with the kernel parameter `netdev__max_backlog`. For example:

```
echo 20000 > /proc/sys/net/core/netdev_max_backlog
echo 1 > /proc/sys/net/ipv4/route/flush
ifconfig eth0 txqueuelen 10000
```

The default value of 100 for `txqueuelen` is inadequate for long-distance, high-throughput network links. For example, a gigabit network with a latency of 100ms would benefit from a `txqueuelen` of at least 10000.

Consult your operating system vendor for additional information on setting the queue sizes for various latencies.

The following table shows network improvements seen in the MAA test lab for the adjustments related to TCP socket buffer sizes and network device queue sizes.

Table 1: TCP/IP Tuning Effects

Testing Stage	**Test duration (seconds)**	**Amount of data transferred**	**Network throughput achieved (Megabits/sec)**	**% change**
Prior to tuning	60	77.2 MB	10.8 Mbps	N/A
After increasing network socket buffer size to 3*BDP from default of 16K	60	5.11 GB	731.0 Mbps	665% improvement over baseline prior to tuning
After above adjustment and increase of device queue lengths to 1,000 from default of 100	60	6.55 GB	937.0 Mbps	28% improvement

STANDBY I/O TUNING BEST PRACTICES

As redo is received by the standby it is written to disk. In certain configurations the disk write must occur prior to sending acknowledgment back to the primary that the redo has been received. Therefore it is important to optimize I/O on the standby. To improve I/O performance on the standby consider the following best practices:

1. Ensure that Oracle is able to utilize 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, Host Bus Adapter (HBA) driver, and storage array.
2. 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, disk array.
3. Place SRLs in an ASM diskgroup that has at least the same number of disk as the ASM diskgroup where the primary online redo logs reside.
4. Do not multiplex standby redo logs.
5. Typically RAID controllers configured in RAID5 perform writes slower than when configured with mirroring. If the process of writing to the SRL becomes the bottleneck consider changing the RAID configuration.

REDO TRANSPORT SERVICES PERFORMANCE TUNING

As described above, factors that influence the performance of a Data Guard configuration include the protection mode, the transport option that is chosen, network bandwidth and latency, and the performance of I/O to online redo logs and standby redo logs. This section shows how to assess any possible impact to the production database performance because of these factors and also how to tune the configuration settings to minimize this impact.

Assessing Production Database Performance

The first step in any performance tuning exercise is to gather a good baseline. Prior to implementing Data Guard transport services gather AWR reports that represent both normal and peak workloads. Reduce the AWR automatic snapshot interval to 10-20 minutes to capture performance peaks during stress testing or peak loads. During normal operations a 60 minute interval should be sufficient.

Key items to examine in the baseline AWR reports that will give you a good indication of production database throughput or performance are:

- Redo volume – the amount of redo bytes generated during this report
- Transactions – TPS for the report.
- Redo writes – Number of redo writes made during this report

The ‘redo volume’ divided by the ‘redo writes’ give the average redo write size in bytes. This is an important metric if you are analyzing `LGWR SYNC` or `ASYNC` performance to be discussed later.

In addition to using data obtained from AWR reports, you can also measure production database response time by examining the `V$SYSMETRIC_HISTORY` view. The information in this view will give a good indication of the response time for user queries. Consider the following metrics when examining this view:

- Response Time Per Txn - Response time for transactions
- SQL Service Response Time - Response time per user call in centiseconds
- Database Time Per Sec - DB time in centiseconds / elapsed time in secs

After obtaining a good baseline enable Data Guard and gather AWR reports during both normal and peak operations. These AWR reports can be used to compare against the baseline to derive the production database performance with Data Guard enabled. A detailed examination of this process is given later.

Wait Events Related to Data Guard

The time involved in sending redo from the production database to the standby can be broken down into two major sections; the time spent to traverse the network and the time spent writing to the disk on the standby host. The wait events that capture the total time are obtained on the production database while the time spent doing I/O can be viewed on the standby.

The following wait events deal with the process of sending redo from the production database to the standby database using either the ARCH or LGWR (SYNC or ASYNC) processes. These wait events are obtained from the production database.

Table 2: Key Data Guard Wait Events on Primary Database

Event Name	Process	Description
ARCH wait on ATTACH	ARCH	This wait event monitors the amount of time spent by all archiver processes to spawn a new RFS connection.
ARCH wait on SENDREQ	ARCH	This wait event monitors the amount of time spent by all archiver processes to write archive logs to the local disk as well as write them remotely.
ARCH wait on DETACH	ARCH	This wait event monitors the amount of time spent by all archiver processes to delete an RFS connection.
LNS wait on ATTACH	LGWR	This wait event monitors the amount of time spent by all LNS processes to spawn a new RFS connection.
LNS wait on SENDREQ	LGWR	This wait event monitors the amount of time spent by all LNS processes to write the received redo to disk as well as open and close the remote archived redo logs.
LNS wait on DETACH	LGWR	This wait event monitors the amount of time spent by all LNS processes to delete an RFS connection.
LGWR wait on LNS	LGWR	This wait event monitors the amount of time spent by the log writer (LGWR) process waiting to receive messages from LNS processes.
LNS wait on LGWR	LGWR	This wait event monitors the amount of time spent by LNS processes waiting to receive messages from the log writer (LGWR) process.
LGWR-LNS wait on channel	LGWR	This wait event monitors the amount of time spent by the log writer (LGWR) process or the LNS processes waiting to receive messages.

Important I/O related waits events that should be monitored on the primary database include:

Table 3: Key Database Wait Events Related to Data Guard

Event Name	Process	Description
Log File Sync	LGWR	Wait time from when foreground commits a transaction to when LGWR acknowledging the completion of the commit to the foreground. When a user session commits, the session's redo information needs to be flushed to the redo logfile. The user session will post the LGWR to write the log buffer to the redo log file. When the LGWR has finished writing, it will post the user session.
Log File Parallel Write	LGWR	Time it takes for LGWR I/Os to complete for a commit write. Even though redo records are written in parallel, the parallel write is not complete until the last I/O is on disk.
DB File Sequential Read	Foreground	The session waits while a sequential read from the database is performed. This event is also used for rebuilding the controlfile, dumping datafile headers, and getting the database file headers.

Viewing the standby I/O wait events associated with log transport services is done on the standby. While the I/O wait events are the same for both physical and logical standbys the method to collect them differ. If the standby is a logical standby then AWR reports can be used to view the waits events. If the standby is a physical then SQL queries are used to get the information. The following queries can be used for a physical standby:

```
SQL> SELECT EVENT, TOTAL_WAITS,
     ROUND(TIME_WAITED/100) "TIME(S)",
     AVERAGE_WAIT*10 "AVG(MS)",
     TO_CHAR(SYSDATE, 'DD-MON-YYYY HH:MI:SS') TIME
     FROM V$SYSTEM_EVENT
```

The following wait events describe time spent doing I/O on the standby:

Table 4: Key Data Guard Wait Events on Standby Database

Event Name	Process	Description
RFS Write	RFS	The elapsed time for the write to the standby redo log or archive log to occur as well as non I/O work such as redo block checksum validation
RFS random i/o	RFS	The elapsed time for the write to a standby redo log to occur.
RFS sequential i/o	RFS	The elapsed time for the write to an archive log to occur.

The time spent in the above wait events are dependent on the average write size and the I/O capacity of the storage array. For tuning suggestions reference the "Standby I/O Tuning Best Practices" section earlier in this paper.

LGWR SYNC Tuning Example

As described in Data Guard documentation [5], `LGWR SYNC` redo transport is a synchronous process that causes the database to delay acknowledgment of user commits until the following actions have been completed:

1. LGWR writes redo to the online redo log at the production database (when LGWR SYNC is not used, user commits are acknowledged once this step completes except when the parameter COMMIT NOWAIT is used, as described in Appendix C).
2. The Data Guard LNS process on the production database performs a network send to the Data Guard RFS process on the standby database. For redo write sizes larger than 1MB LNS will issue multiple network sends to the RFS process on the standby.
3. The RFS process receives the redo being sent by LNS and completes the I/O into the standby redo log.
4. RFS process sends acknowledgment back to LNS that the redo has been received and written to disk.
5. LNS posts LGWR that the all the redo has been successfully received and written to disk by the standby.

On the production database the time for step one is represented with the "log file parallel write" wait event. Steps 2 through 5 are represented with the "LNS wait on SENDREQ" wait event. You can further divide the "LNS wait on SENDREQ" wait event into network time and RFS I/O time by subtracting the "RFS write" wait event obtained on the standby. These wait events can be assessed on the standby by using the query described in the "Data Guard Specific Wait Events" section above for a physical standby or by using AWR for a logical standby.

The "RFS write" wait event covers the time needed to perform the I/O as well as RFS processing. You can assess the actual RFS I/O time by examining "RFS random i/o" wait event.

The following `LGWR SYNC` tuning example is based on an OLTP application that produces 152 KB of redo per second with a Gigabit network and 10ms latency. After obtaining baseline AWR reports a physical standby was created and the production database configured to ship redo the standby using `LGWR SYNC AFFIRM.`

Note: `AFFIRM` will instruct RFS to send an acknowledgement to the production database only after the redo has been received and written to a standby redo log at the standby database; this is the default configuration for `LGWR SYNC`.

Before performing any network and I/O tuning, the application was run and AWR reports collected. After analyzing the AWR reports, the tuning recommendations outlined in this paper were performed and AWR reports were again collected.

With `LGWR SYNC` enabled the load profiles for the different tests were:

Table 5: Load Profile for LGWR SYNC

Load Profile	Without Data Guard	With Data Guard
Redo Rate	152,057 bytes/sec	145,993 bytes/sec
Redo Writes	137	114
Redo Write Size	1,109 bytes	1,280 bytes

The following table illustrates important wait events in each of the different stages of testing. The I/O and log file sync are of interest as they are a good indicator of application performance.

Table 6: Wait Events for LGWR SYNC

Wait Event	Without Data Guard	With Data Guard
log file sync	12	17
db file sequential read	19	17
log file parallel write	1	1
LNS wait on SENDREQ	N/A	13
RFS random i/o	N/A	2

Note: Wait Events in milliseconds. The LNS wait on SENDREQ and RFS random i/o wait event times are not applicable for the baseline as Data Guard was not enabled.

The log file sync wait event is an important wait event to examine. The log file sync (LFSY) wait event is the time the foreground spends waiting for redo to be flushed to make its commit durable. The log file sync wait can be broken up into the following subsections:

1. Wakeup LGWR if idle
2. LGWR gather the redo to be written and issue the I/O
3. Time for the log write I/O to complete
4. LGWR I/O post processing
5. LGWR posting the foreground that the write has completed
6. Foreground wakeup

Steps number 2 and 3 are accumulated in the "redo write time" statistic. Step 3 is the "log file parallel write" wait event. Steps 5 and 6 can become very significant as the system load increases, because even after the foreground has been posted it may take a while for the OS to schedule it to run. When you see high 'log file sync' wait times you should break down the total wait time down into the individual components and then tune those components that make up the largest time.

With an application that has high concurrency; increasing network latency may have a negligible impact on overall database throughput, although response time for individual transactions may increase. The average LGWR write size is expected to increase as latency increases for applications with high concurrency. Once tuning was performed the impact on production database throughput was reduced to just 4%. Tuning consisted of the following items:

- On the standby system increase /proc/sys/fs/aio-max-size to 1048576 from the default of 131072.
- Removed second logfile members for the standby log file groups
- Added additional disks to the ASM diskgroup on the standby
- Set `RECV_BUF_SIZE` and `SEND_BUF_SIZE` to 3 times the bandwidth delay product
- Increased device queue sizes associated with the network interfaces from the default of 100 to 10,000

In this example it can be seen that due to the small redo write size, increasing the network send and receive buffer sizes did not have a significant impact on LNS throughput. Increasing the buffer sizes would provide improvement as the redo write size increases. With small redo write sizes the biggest gain in tuning comes from optimizing the I/O on the standby. It's also important to note that by default LGWR SYNC will issue a maximum of 1MB network sends to the standby regardless of the redo write size. This will cause multiple network round trips to occur for applications with redo write sizes larger than 1MB. This behavior was changed in 10.2.0.4 and forward to accommodate for redo write sizes larger than 1MB. If you are on a version of Oracle Database 10g prior to 10.2.0.4 and have redo write sizes larger than 1MB, see MetaLink note 404985.1 for details on how to obtain this optimization for your version.

For users who wish to greatly reduce the impact of LGWR SYNC on production database performance, evaluate if your application or parts of your application can utilize COMMIT NOWAIT or non-durable commits. With COMMIT NOWAIT IMMEDIATE, the foreground process posts LGWR to do the log I/O but does not wait for the redo to be written. With COMMIT NOWAIT BATCH, the foreground simply returns. In both cases the foreground does not have to wait for the completion of the local LGWR write or for the redo to be received by the standby prior to resuming work. With LGWR SYNC enabled, COMMIT NOWAIT BATCH compared with the default COMMIT IMMEDIATE WAIT we see the following in a sample OLTP workload on Real Application Clusters:

- Increased production redo rate by 10-35%
- Increased user calls rate by 10-35%

- Increased production txn rate by 10-33%
- Reduced user call response time by 92%
- Reduced txn response time by by 90%

This option may be ideal for customers who can tolerate non-durable commits in the case of instance or database failures. Examples of these type of applications may be shopping cart applications, but not for the purchase transaction or sampling or tracking systems for trends or maybe Data Warehouse or Data Mart applications. See Appendix C for more details.

LGWR ASYNC Tuning Example

`LGWR ASYNC` is an asynchronous transport service where user commits do not have to wait for redo to be sent remotely. To understand how to tune the efficiency of `LGWR ASYNC` consider the following:

1. LGWR writes redo to the online redo log at the production database.
2. The Data Guard LNS process on the production database reads the online redo log and performs a network send to the Data Guard RFS process on the standby database.
3. The RFS process receives the redo being sent by LNS.
4. The RFS process sends acknowledgement back to LNS that the redo has been received
5. The RFS Process writes the redo to a standby redo log.

On the production database the “LNS wait on SENDREQ” wait event represents steps 2 through 4. Step five is represented by “RFS write” obtained on the standby using the query described in the above section. The actual RFS write time is illustrated with “RFS random i/o” for writes to the standby redo logs or “RFS sequential i/o” for writes into archive log files. The “RFS write” wait event covers the time needed to perform the I/O as well as some additional RFS processing. These wait events can assessed on the standby by using the query described in the “Data Guard Specfic Wait Events” section above for a physical standby or by using AWR for a logical standby.

The following `LGWR ASYNC` tuning example is for an OLTP application that produces just over 5.5 MB/sec of redo per second. After obtaining baseline AWR reports, a physical standby was created and the production database configured to ship redo the standby using `LGWR ASYNC`. Prior to performing network or I/O tuning, the application was run and new AWR reports collected. Then the tuning recommendations outlined in this paper were performed and another round of AWR reports were collected.

With LGWR ASYNC enabled the load profiles for the three different tests were:

Table 7: Load Profile for LGWR ASYNC

Load Profile	Without Data Guard	With Data Guard
Redo Rate	5.66 MB/sec	5.49 MB/sec
Redo Writes	741	746

As the above table illustrates, the impact of enabling `LGWR ASYNC` was at about 4%. This impact can be attributed to the additional disk reads being performed by LNS on the production database's online redo logs.

Tuning consisted of the following items:

- On the standby increase /proc/sys/fs/aio-max-size to 1048576 from the default of 131072
- Removed logfile members for the standby log file groups
- Added additional disks to the ASM diskgroup on the standby
- Set RECV_BUF_SIZE and SEND_BUF_SIZE to 3 times the bandwidth delay product
- Increased device queue sizes associated with the network interfaces

While production database overhead and the impact of tuning appears negligible with LGWR ASYNC, it is still important to have efficient network transfer to the standby to avoid any delay in shipping redo. This minimizes potential data loss. The benefit of network tuning is more obvious in the following table when we examine the following wait events:

Table 8: Wait Events for LGWR SYNC

Wait Event	Without Data Guard	With Data Guard
LNS wait on SENDREQ	N/A	179 ms
RFS random i/o	N/A	21 ms

Note: Wait Events Average in milliseconds. Some wait events are not applicable for the baseline as Data Guard was not enabled.

As the above table illustrates, the time needed for LNS to perform a network send was significantly impacted by the I/O on the standby as well as the network socket buffer sizes. Note that the amount of data sent by LNS can vary depending on workload. Knowing the LNS send size enables network and I/O testing to be performed to see if the time spent in various stages seems reasonable. To determine the network send size for LNS run the following query on the production database:

```
SQL> SELECT OPEN_COUNT, CLOSE_COUNT, WRITE_COUNT, MINIMUM_WRITE,
MAXIMUM_WRITE, TOTAL_WRITE, LOGS_SKIPPED,TERMINATIONS
     FROM X$KCRRASTATS WHERE TOTAL_WRITE > 0;
```

To get the average LNS write size divide TOTAL_WRITE by WRITE_COUNT.

Optimizing ARCH Performance

Regardless of redo transport chosen, there is always one ARCH process dedicated to archiving online redo logs locally (Data Guard 10g onward). When ARCH redo transport services are configured the local archive will complete first and a different ARCH process will begin the remote archive using the following logic:

1. Read 10 megabytes from the local archive log and issue a network send to the RFS process on the standby
2. The RFS process receives the redo sent by the ARCH process and performs I/O into either the standby redo log or archive redo logs, depending upon how the user has configured Data Guard.
3. Once the I/O has completed the RFS sends an acknowledgement back to ARCH
4. ARCH reads the next 10 megabytes and then repeats the above process

If the MAX_CONNECTIONS attribute has been configured for the remote archive destination then up to five ARCH processes (depending upon configuration) can participate in sending a single archive log remotely. The actual RFS write time is illustrated with "RFS random i/o" for writes to standby redo logs or "RFS sequential i/o" for writes into archive log files.

Examining the ARCH wait on SENDREQ wait event to assess ARCHs performance can be misleading. This is due to the fact that the network activity by the ARCH ping mechanism (periodically checking on standby availability) is captured within that wait event and it can lead to misleading numbers. The most reliable method to assess ARCH performance is to enable log_archive_trace to level 128 to get additional messages printed to the alert log. Using the timestamps and the archive log file size you can assess overall throughput.

As with LGWR SYNC/ASYNC, it is important to have efficient disk I/O on the standby and primary flash recovery area as well as properly sized network socket buffers. The same tuning items performed for ASYNC should be performed for ARCH. In addition to those tuning items the remote destination should be configured with the MAX_CONNECTIONS attribute set to 5. This will enable parallel ARCH transfer and improve overall throughput for individual archive log files.

The LOG_ARCHIVE_MAX_PROCESSES initialization parameter can be also be set as high as 30, enabling up to 30 ARCH processes (1 dedicated to local archival, and 29 also able to archive either locally or remotely). The Oracle Database 10g Release 2 default is 2. It is always a good idea to increase this level beyond the default when using Data Guard. ARCH processes will be consumed by local archival, by resolving archive log gaps at the standby that result from network or standby database outages, or by normal remote archival if Redo Transport Services have been configured to utilize archiver processes. Increase the `LOG_ARCHIVE_MAX_PROCESSES` value to a minimum level needed to accommodate what you configure for `MAX_CONNECTIONS`. If you have the bandwidth to support additional redo streams, consider setting `LOG_ARCHIVE_MAX_PROCESSES` to as high a value as your network can accommodate. This makes it possible to send multiple archive logs in parallel to handle peaks in workload or to more quickly resolve log archive gaps caused by network or standby failures.

Diagnosing Network Performance Issues

First ensure that all items described in the Network Tuning section have been addressed. If problems still persist, use standard diagnostic techniques at the OS level to investigate network performance issues and bottlenecks. Using OS utilities investigate the following areas:

1. CPU activity
2. Memory utilization
3. Network errors or collisions
4. Network throughput using a tool such as iperf [9]
5. TCP packet dumps (sniffer traces)

Once OS level bottlenecks have been eliminated, examine the database alert log and the database dump destinations, bdump, udump, and cdump to make sure that no errors are occurring. Frequent errors or disconnection with the standby RFS process(es) can cause added overhead to the production database when using `LGWR SYNC` redo transport services.

APPENDIX A – REDO TRANSPORT ENHANCEMENTS

Redo transport services transmit redo data from a production database to local and remote standby database destinations. Remote destinations can include any of the following types: Data Guard standby databases (physical and logical standby databases), archived redo log repositories, Oracle Change Data Capture staging databases, and Oracle Streams downstream capture databases. Oracle Database 10*g* Release 2 includes the following improvements to Redo Transport Services:

LGWR ASYNC Enhancements

Enhanced Performance: Prior to Oracle Database 10*g* Release 2, the network server process (LNS*n*) will issue network I/O's and wait for each network I/O to complete. Each `LNS`*n* process has a user-configurable in-memory buffer that accepts outbound redo data from the `LGWR` process. If the `LNS`*n* process cannot keep up, (for example in cases of peak redo volume that may exceed available network bandwidth) then the buffer becomes full and the `LGWR` process stalls until either sufficient buffer space is made available by a successful network transmission or a timeout occurs. This temporary stall can affect production database throughput.

In Oracle Database 10*g* Release 2, the new `LGWR ASYNC` behavior eliminates the potential to stall the production database. When the `LGWR` and `ASYNC` attributes are specified on the `LOG_ARCHIVE_DEST_`*n* initialization parameter, the log writer process writes to the local online redo log file with no additional writes to the `ASYNC` buffer. After the `LGWR` has completed writing to the online log, the `LNS`*n* process reads from the online redo log and ships the redo to the standby database. With this approach, the `LGWR` process writes are completely decoupled from `LNS`*n* network writes. Asynchronous redo transmission using `LGWR` is no longer constrained by the size of the `ASYNC` buffer. When an online redo log file becomes full, a log switch occurs, and an archiver process archives the log file locally, as usual.

Enhanced data protection: In Oracle Database 10*g* Release 1, the potential data loss is bounded by the size of the `ASYNC` buffer, unless the buffer full condition is reached as described above. Once the buffer full condition is reached LNS will cease shipping the current redo stream to the standby database and the destination will be serviced by the `ARCH` process. When this occurs the potential data loss exposure is determined by the size of the current online log or by how many log files it lags behind until Data Guard can resynchronize the standby database and LNS can resume shipping.

With the new `ASYNC` model used in Oracle Database 10*g* Release 2, the potential data loss is measured by how far behind the `LNS`*n* process is from the current position of the LGWR process. If `LGWR` switches into a new online log file before the `LNS`*n* process has completed, then `LNS`*n* continues to remotely archive its current log file. If `LGWR` once again switches to a new online redo log file while the

LNS*n* process is still on the original log file, then the ARCH process remotely archives the log file that the LNS*n* process has not yet begun to archive. Once LNS*n* has completed archiving its original log file, it begins archiving the current log file for LGWR.

ARCH Enhancements

Oracle Database 10*g* Release 2 introduces the ability to have multiple ARC*n* processes sending redo for a single archived-redo log over the network, reducing the time required to ship a single archive log to a remote destination. In previous Oracle Database releases, only one ARC*n* process at a time could archive redo from a given log file. The maximum number of network connections that will be used in parallel to perform archival to a destination is controlled using the MAX_CONNECTIONS attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter. The LOG_ARCHIVE_MAX_PROCESSES initialization parameter can be set as high as 30, (the previous maximum was 10).

Additional information about Redo Transport Services, the LGWR, ARC*n*, and LNS*n* background processes, and the related initialization parameters is provided in *Oracle Data Guard Concepts and Administration*[3].

Enhancements to Network Send Size

Prior to Oracle Database 10*g* Release 2, log transport services sent redo information across the network in 1 MB segments.

In Oracle Database 10*g* Release 2, the 1 MB network-send size has been increased to 10 MB. The larger size minimizes network idle time by reducing the total time spent waiting for RFS confirmation for a given amount of redo transmitted.

APPENDIX B – REDO TRANSPORT PERFORMANCE TESTS

Oracle conducted several MAA performance tests for representative Data Guard configurations using Maximum Performance mode with LGWR ASYNC redo transport services. The tests measured the degree to which Redo Transport and network tuning best practices could minimize overhead on the production database while at the same time minimize potential data loss. Consider the following performance profiles:

- **Configuration 1:** An application that generates redo data at an average rate of just under 1 MB/sec (Megabytes per second). At this rate, a 500MB online log will switch every 8 minutes. This volume is representative of the majority of applications encountered in the field. The standby database is located across a wide area network (WAN) with an RTT of 25 ms (milliseconds).

  Impact: Less than a 5% impact on production database performance and potential data loss exposure of approximately 5 seconds of data in the event of a sudden failover.

- **Configuration 2:** An application that generates redo data at an average rate of 2 MB/sec. At this rate, a 500MB online log file will switch every 4 minutes. The standby database is located across a WAN with RTT of 25 ms.

  Impact: Less than a 5% impact on production database performance and potential data loss exposure of approximately 7 seconds of data in the event of a sudden failover.

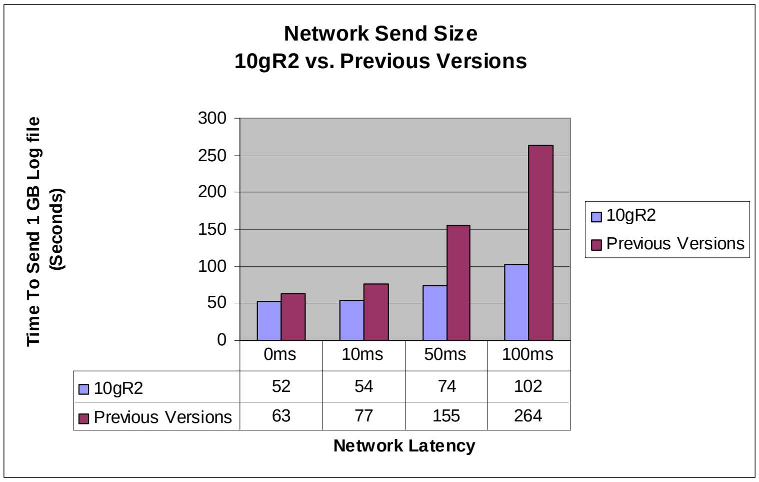
- **Configuration 3:** An application generates redo data at a very high average rate of 20 MB/sec. At this rate, a 500MB online log file will switch in 25 seconds. The standby database is located across a WAN with RTT of 6ms.

  Impact: Less than 10% impact on the production database and potential data loss exposure of 40 seconds of data in the event of a sudden failover.

Network Send Size – Better Network Utilization

Figure 1 shows how the increase in the network send size from 1 MB to 10 MB results in a drop in elapsed time when transferring a 1 GB archived redo log file over various network latencies.

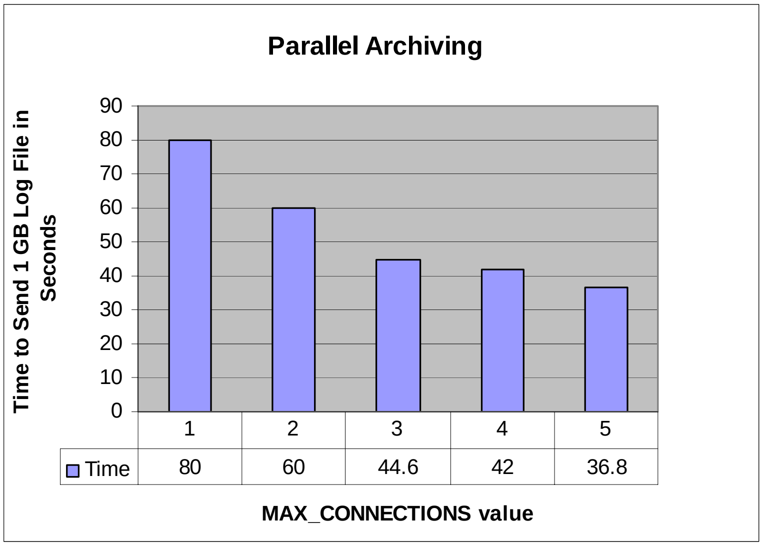
Figure 1: Network Send Size Effect s on Network Idle Time



ARCH Transport – Faster Remote Archival

Figure 2 shows how increasing the value of the `MAX_CONNECTIONS` attribute to 5 can reduce the elapsed time required to send a 1 GB log file by 55%.

Figure 2: Parallel Archiving Through Multiple ARCn Network Connections



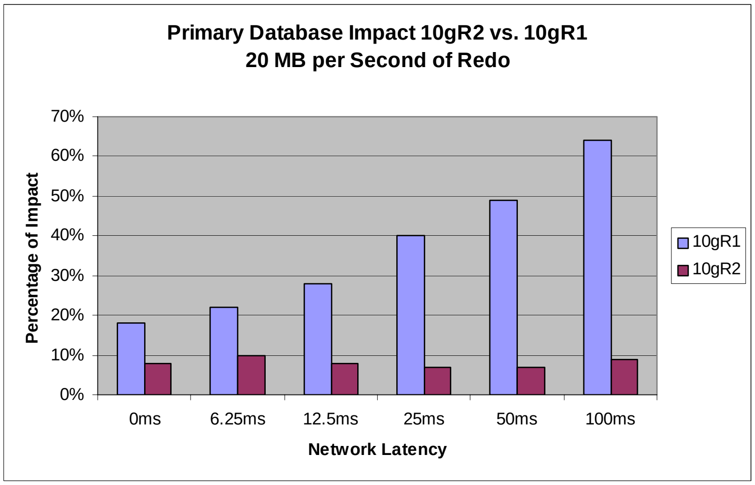
LGWR ASYNC – Less Production Overhead

In Oracle Database 10*g* Release 2, as redo rates and network latencies increase, the impact on the production database does not increase. Instead, it remains relatively constant. For example, at redo rates of less than or equal to 2 megabytes per second (MBs), the production database impact (measured by the change in redo bytes per second from the baseline) is less than 5% across different latencies ranging from 0 to 100ms RTT.

The same relationship holds true even when workload is increased significantly. Figure 3 shows how the difference between the ASYNC models used in Oracle Database 10g Release 1 and Release 2 becomes very significant when the production database redo rate is increased to approximately 20 MB/sec.

Note: Tests simulated various latencies using a 1GB network.

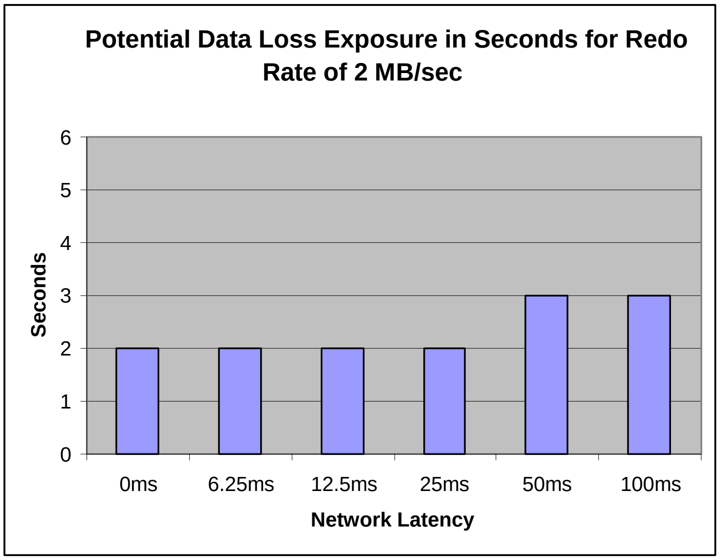
Figure 3: Performance Effect of 20 MB/sec Redo Rate On a Production Database



LGWR ASYNC – Better Data Protection

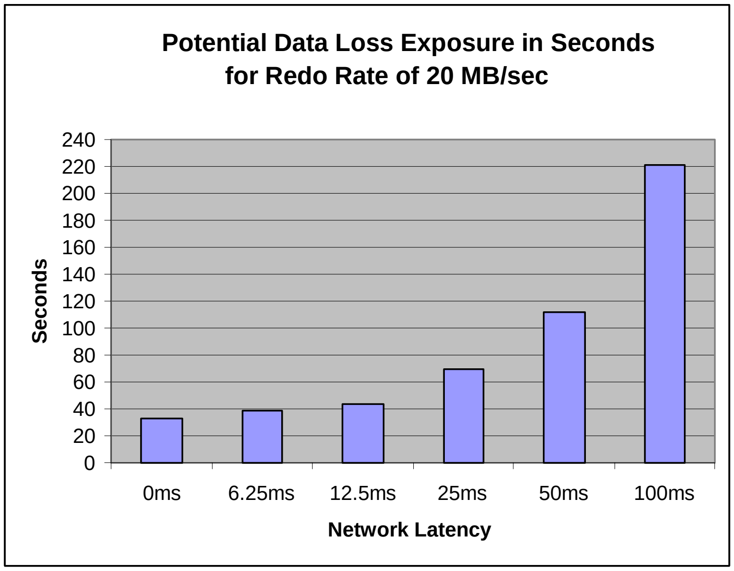
Testing within the MAA lab was performed to determine the impact of Data Guard 10g Release 2 enhancements on potential data loss exposure when using the new LGWR ASYNC model. The first test scenario simulates various network latencies with a constant workload of 2 MB/sec on a correctly tuned gigabit network and 1 GB online redo log files. It was found that redo transport could maintain pace with production database redo generation up to a network latency of 50ms, keeping potential data loss exposure to a minimum, as shown in Figure 4.

Figure 4: Potential Data Loss Exposure for 2 MB/sec Redo Rate



Results from a second test are provided in Figure 5 showing the impact on data protection of various latencies for a very high workload generating 20 MB/sec of redo data. Network tuning was optimized for the higher workload using the same gigabit network and 1 gigabyte online redo log file configuration as in the previous test. As figure 5 shows, even with very high redo rates of 20 MB/sec the potential data loss exposure remains relatively low as the network latency increases.

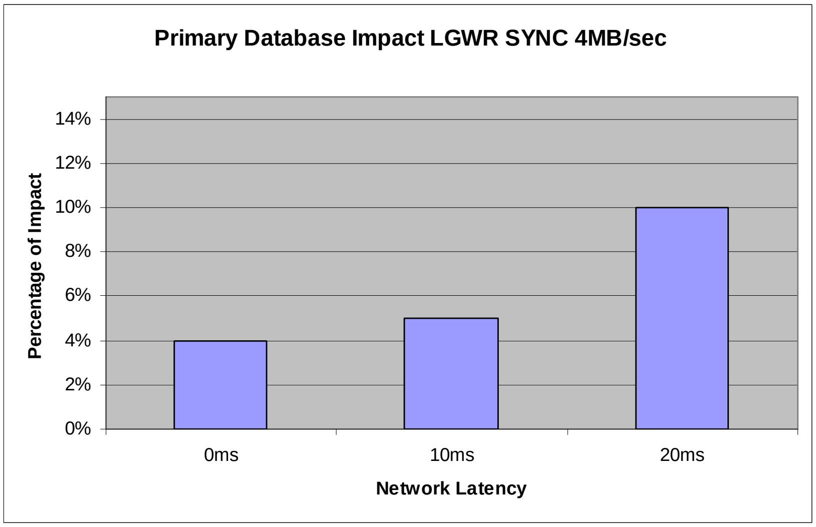
Figure 5: Potential Data Loss Exposure for 20 MB/sec Redo Rate



LGWR SYNC – Impact of Network Latency

Due to the nature of synchronous redo transport the speed and latency of the network, as well as the I/O bandwidth of the standby server will impact the production database throughput as shown in Figure 6.

Figure 6: LGWR SYNC Production Database Impact



APPENDIX C – COMMIT NOWAIT AND COMMIT WAIT COMPARISON

When a transaction updates the database, it generates a redo entry corresponding to this update. Oracle Database buffers this redo in memory until the completion of the transaction. When the transaction commits, the log writer (LGWR) process writes redo for the commit, along with the accumulated redo of all changes in the transaction, to disk. By default Oracle Database writes the redo to disk before the call returns to the client. This behavior introduces latency in the commit because the application must wait for the redo to become persistent on disk.

Suppose your database supports a number of applications that have different requirements for data loss. In this scenario, one application may require very high transaction throughput, but can tolerate the potential data loss exposure that comes with `COMMIT_NOWAIT`. The parameter COMMIT NOWAIT can be used so that commits are returned to the application without waiting for redo to be written to disk. Therefore, applications or transactions that can utilize COMMIT NOWAIT will have a significant improvement in response time and database throughput over applications or transactions that utilize the default COMMIT WAIT behavior. If you are willing to trade commit durability for lower commit latency, then you can change the default `COMMIT` options so that the application does not need to wait for Oracle Database to write data to the online redo logs. By changing the default COMMIT options on this one application; zero data loss protection for other applications utilizing the same database is unchanged. Oracle Database enables you to change the handling of commit redo depending on the needs of your application. You can change the commit behavior in the following locations:

- `COMMIT_WRITE` initialization parameter at the system or session level
- `COMMIT` statement

The options in the `COMMIT` statement override the current settings in the initialization parameter. Table 2-1 describes redo persistence options that you can set in either location.

Table 2-1 Initialization Parameter and COMMIT Options for Managing Commit Redo

Option	Specifies that . . .
`WAIT`	The commit does not return as successful until the redo corresponding to the commit is persisted in the online redo logs (default).
`NOWAIT`	The commit should return to the application without waiting for the redo to be written to the online redo logs.
`IMMEDIATE`	The log writer process should write the redo for the commit immediately (default). In other words, this option forces a disk I/O.

Option	Specifies that . . .
BATCH	Oracle Database should buffer the redo. The log writer process is permitted to write the redo to disk in its own time.

The following example shows how to set the commit behavior to BATCH and NOWAIT in the initialization parameter file:

COMMIT_WRITE = BATCH, NOWAIT

You can change the commit behavior at the system level by executing ALTER SYSTEM as in the following example:

ALTER SYSTEM SET COMMIT_WRITE = BATCH, NOWAIT

After the initialization parameter is set, a COMMIT statement with no options conforms to the options specified in the parameter. Alternatively, you can override the current initialization parameter setting by specifying options directly on the COMMIT statement as in the following example:

COMMIT WRITE BATCH NOWAIT

In either case, your application specifies that log writer does not have to write the redo for the commit immediately to the online redo logs and should not wait for confirmation that the redo has been written to disk.

The following is a summary of the COMMIT NOWAIT vs COMMIT WAIT comparison.

- Without Data Guard, commit batch-nowait compared with the default commit immediate-wait reduced user call response time by 43% (approx 4 ms to 2 ms) and transaction response time by 31% (approx 12 ms to 9ms)
- With Data Guard, commit batch-nowait compared with the default commit immediate-wait
    - increased redo rate by 10-35%
    - increased txn rate by 10-33%
    - reduced txn response time by by 90%
- Commit batch nowait results were essentially identical between non-Data Guard and Data Guard nowait runs. The key indicators are when log file sync waits are near zero. For example for a 5 minute run of the calling circle application, we observed 56,391 log file sync waits totaling 241 secs of wait time with commit WAIT set while with commit NOWAIT, we observed only 1 log file sync wait for less 1 sec of wait time.
- Customer Impact: Nowait option can provide reduced user call and transaction response time for applications in general. Nowait option provides a significant performance and throughput benefit for Data Guard

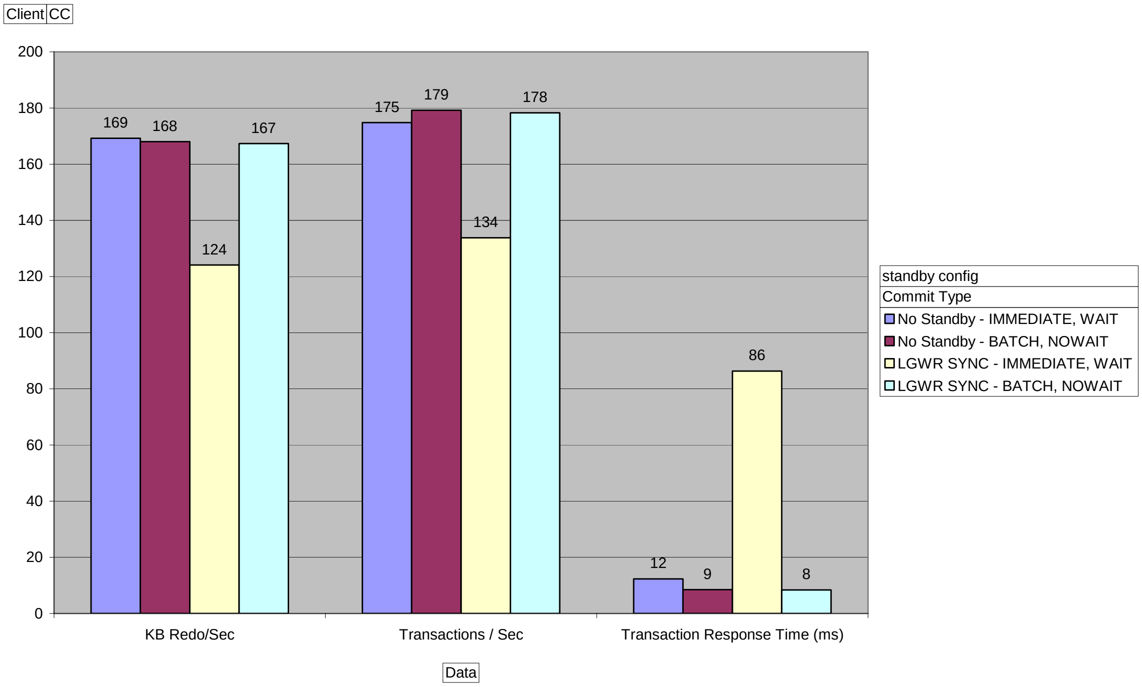
environments configured with synchronous transport. This option may be ideal for customers who can tolerate non-durable commits in the case of instance or database failures. Examples of these type of applications may be shopping cart applications but not for the purchase transaction or sampling or tracking systems for trends or maybe Data Warehouse or Data Mart applications.

Note: Refer to Oracle Database Application Developer's Guide – Fundamentals 10g Release 2 [10] for more information on using COMMIT NOWAIT to improve transaction performance.

Application Examples:

Calling Circle application description

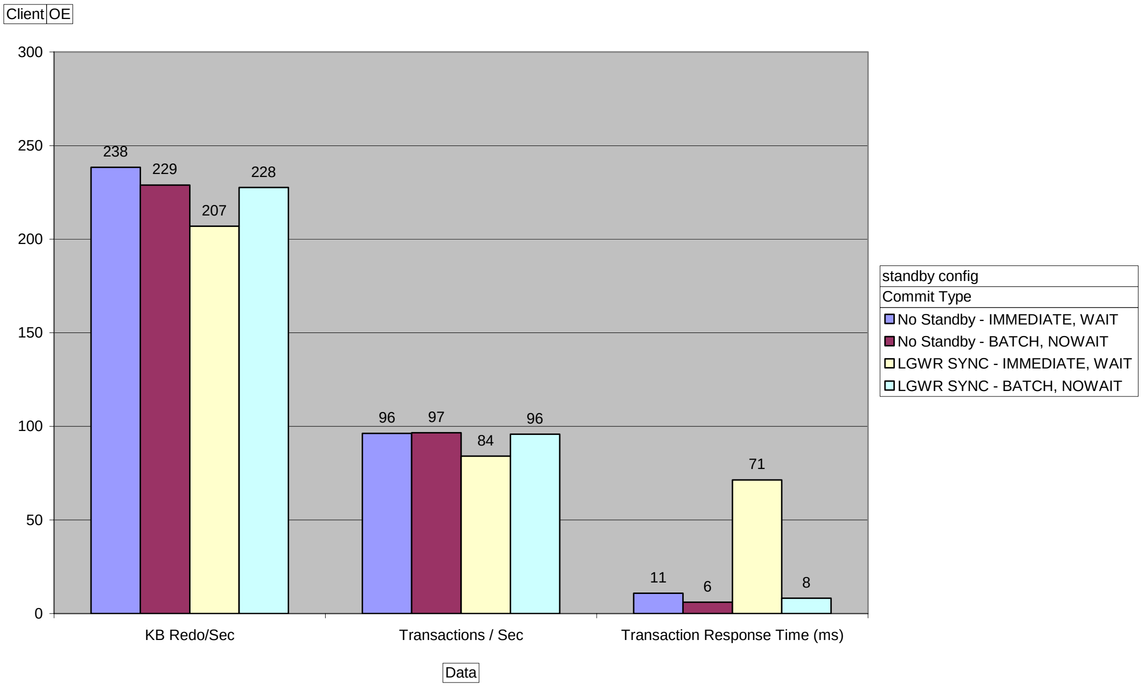
The Swingbench load generation tool is used to generate the Calling Circle workload. The Calling Circle workload represents a self-service OLTP application. The application models the customers of a telecommunications company registering, updating and enquiring on a calling circle of their most frequently called numbers in order to receive discounted call pricing. The IO size for this workload is small (generally the tablespace blocksize which is 8K in this case). It connects to the database using thin JDBC and uses JDBC connection pools, Fast Start Failover and Fast Application Notification (FAN) for failover.



Order Entry application description Order Entry –

The Swingbench load generation tool is used to generate the Order Entry workload. It consists of 5 transactions types which are typical in an order entry application: Creating a Customer, Browsing Products, Placing Orders, Processing Orders and Reviewing Orders. The IO size for this workload is small (generally the tablespace blocksize which is 4K in this case). It connects to the database using thin JDBC and uses JDBC connection pools, Fast Start Failover and Fast Application Notification (FAN) for failover.

Note: v$servicemetric.dbtimepercall was leveraged to assess call response time and v$sysmetric.metric_name='Response Time Per Txn' was used to assess transaction response time



REFERENCES

1. Oracle Data Guard
   http://www.oracle.com/technology/deploy/availability/htdocs/DataGuardOverview.html

2. Oracle Maximum Availability Architecture
   http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm

3. Redo Apply (physical standby) MAA Best Practices
   www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gRecoveryBestPractices.pdf

4. SQL Apply (logical standby) MAA Best Practices
   www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_SQLApplyBestPractices.pdf

5. *Oracle Data Guard Concepts and Administration*
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14239

6. Oracle Data Guard 10g Release 2: Switchover and Failover Best Practices
   http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_SwitchoverFailoverBestPractices.pdf

7. Best Practices for Creating a Low-Cost Storage Grid for Oracle Databases
   http://www.oracle.com/technology/deploy/availability/pdf/ora_lcs.pdf

8. *Oracle Database Net Services Reference*
   http://download-west.oracle.com/docs/cd/B19306_01/network.102/b14213/sqlnet.htm

9. iperf
   http://dast.nlanr.net/Projects/Iperf/

10. Oracle Database Application Developer's Guide – Fundamentals 10g Release 2
    http://download-west.oracle.com/docs/cd/B19306_01/appdev.102/b14251/adfns_sqlproc.htm - CIHIJGDD

Data Guard Redo Transport and Network Best Practices: Oracle Database 10*g* Release 2
February 2007
Author: Michael T. Smith
Contributing Authors: Joseph Meeks, Lawrence To, Douglas Utzig, Robert McGuirk

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