Oracle Maximum
Availability Architecture

Best Practices for Asynchronous Redo Transport

Data Guard and Active Data Guard

ORACLE WHITE PAPER | JUNE 2015

Table of Contents

Introduction

The Oracle Maximum Availability Architecture (MAA) with Oracle Active Data Guard and Data Guard provides the most comprehensive solution available to eliminate single points of failure for mission critical Oracle Databases. It prevents data loss and downtime in the simplest and most economical manner by maintaining one or more synchronized physical replicas of a production database at a remote location. If the production database becomes unavailable for any reason, client connections can quickly, and in some configurations transparently, failover to a synchronized replica to immediately restore service.

Active Data Guard extends basic Data Guard capabilities to eliminate the high cost of idle redundancy by enabling reporting applications, ad-hoc queries, data extracts, and fast incremental backups to be offloaded to read-only copies of the production database. Active Data Guard's complete focus on real-time data protection and availability and its deep integration with the Oracle Database eliminates compromises in data protection, performance, availability and complexity found in storage remote mirroring or other host-based replication solutions.

This paper provides Oracle MAA best practices for using asynchronous redo transport in a Data Guard or an Active Data Guard configuration (transport services are identical) from Oracle Database 11g onward. It is designed for database administrators who have a working knowledge of Data Guard and Active Data Guard configurations using Maximum Performance mode.

.

Data Guard Asynchronous Transport – an Overview

A Data Guard or Active Data Guard configuration includes a production database referred to as the primary database, and up to 30 directly connected replicas referred to as standby databases. Primary and standby databases connect over TCP/IP using Oracle Net Services. There are no restrictions on where the databases are physically located provided they can communicate with each other. A standby database is initially created from a backup of the primary database. Data Guard automatically synchronizes the primary database and all standby databases by transmitting primary database redo - the information used by every Oracle Database to protect transactions - and applying it to the standby database.

Data Guard offers two choices of transport services: synchronous and asynchronous. Asynchronous redo transport is the subject of this paper.

Data Guard also provides three different modes of operation that help users balance cost, availability, performance, and data protection - shown in Table 1. Each mode defines the behavior of the Data Guard configuration if a primary database loses contact with its standby. One of the three modes, Maximum Performance, uses asynchronous transport.

TABLE 1: DATA GUARD PROTECTION MODES

Mode	Risk of data loss	Transport	If no acknowledgement from the standby database, then:
Maximum Protection	Zero data loss Double failure protection	SYNC	Signal commit success to the application only after acknowledgement is received from a standby database that redo for that transaction has been hardened to disk. The production database cannot proceed until acknowledgement has been received.
Maximum Availability	Zero data loss Single failure protection	SYNC FAST SYNC FAR SYNC	Signal commit success to the application only after acknowledgement is received from a standby database or after NET_TIMEOUT threshold period expires – whichever occurs first. A network or standby database outage does not affect the availability of the production database.
Maximum Performance	Potential for minimal data loss	ASYNC	Primary never waits for standby acknowledgment to signal commit success to the application. There can be no guarantee of zero data loss.

Asynchronous Transport Architecture

Asynchronous redo transport (ASYNC) transmits redo data asynchronously with respect to transaction commitment. A transaction can commit without having to wait for an acknowledgement that the redo generated by that transaction has be successfully transmitted to a remote standby database.

With Data Guard ASYNC, the primary database LGWR will continue to acknowledge commit success even if limited bandwidth prevents the redo of previous transactions from being sent to the standby database immediately (picture a sink filling with water faster than it can drain). Data Guard ASYNC uses an LNS process to transmit redo directly from the log buffer of the primary database. If LNS is unable to keep pace and the log buffer is recycled before the redo can be transmitted to the standby, the LNS automatically transitions to reading and sending from the online redo log file (ORL) on disk. Once LNS transmission has caught up with current redo generation it automatically transitions back to reading/sending directly from the log buffer.

If ASYNC redo transport falls behind to the degree that the LNS is still in the ORL at log switch time, LNS will continue until it completes sending the contents of the original ORL. Once complete, it seamlessly transitions back

to reading/sending from the current online log file. When the LNS catches up with the LGWR, it seamlessly transitions back to reading/sending from the redo log buffer.

In the rarer case in which there are two or more log switches before the LNS has completed sending the original ORL, the LNS will still transition back to reading the contents of the current online log file. Any ORLs that were archived in between the original ORL and the current ORL are transmitted via Data Guard's gap resolution process.

Configuration Best Practices

The following MAA best practices are designed to minimize the performance impact of configuring Data Guard asynchronous redo transport to achieve near-zero data loss protection for a production database.

Set TCP Socket Buffer Size to 3 x BDP

For optimal network throughput the minimum recommended settings for TCP send and receive socket buffer sizes is a value equal to the bandwidth-delay product (BDP) of the network link between the primary and standby systems. Settings higher than the BDP may also yield incremental improvement. For example, MAA tests simulating high-latency, high-bandwidth networks continued to realize small, incremental increases in throughput as TCP send and receive socket buffer settings were increased to 3xBDP.

BDP is product of the network bandwidth and latency. Socket buffer sizes are set using the Oracle Net parameters RECV_BUF_SIZE and SEND_BUF_SIZE, so that the socket buffer size setting affects only Oracle TCP connections. The operating system may impose limits on the socket buffer size that must be adjusted so Oracle can use larger values. For example, on Linux, the parameters net.core.rmem_max and net.core.wmem_max limit the socket buffer size and must be set larger than RECV_BUF_SIZE and SEND_BUF_SIZE.

For example, if bandwidth is 622 Mbits and latency is 30 ms, then you would calculate the minimum size for the RECV_BUF_SIZE and SEND_BUF_SIZE parameters as follows:

Bandwidth Delay Product (BDP) = bandwidth x latency

BDP = 622,000,000 (bandwidth) / 8 x 0.030 (latency) = 2,332,500 bytes.

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

Socket buffer size = 3 x BDP

= 2,332,500 (BDP) x 3

= 6,997,500 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 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.

With Oracle Net you can set the send and receive socket buffer sizes globally for all connections using the following parameters in the sqlnet.ora:

```
RECV_BUF_SIZE=6997500
SEND_BUF_SIZE=6997500
```

If you only want the larger buffer sizes for the connections associated with Data Guard transport then we recommend you configure RECV_BUF_SIZE and SEND_BUF_SIZE in the Oracle Net alias used for transport as well as in the listener on the standby host. The following shows an example of the send and receive socket buffer size being set as a description attribute for a particular connect descriptor:

```
standby =
 (DESCRIPTION=
   (SEND_BUF_SIZE=6997500)
   (RECV_BUF_SIZE=6997500)
     (ADDRESS=(PROTOCOL=tcp)
     (HOST=stby_host)(PORT=1521))
   (CONNECT_DATA=
     (SERVICE_NAME=standby)))
```

The socket buffer sizes must be configured the same for all databases within a Data Guard configuration. On a standby side or the receiving side 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=stby_host)(PORT=1521)
    (SEND_BUF_SIZE=9375000)
    (RECV_BUF_SIZE=9375000)))
```

Configure Standby Redo Logs

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 extract peak redo rates, please refer to AWR reports during peak workload periods such as batch processing, quarter or year end processing. It is very important to use peak workload and not averages (averages can obscure peak redo rates and lead to provisioning too small of a redo log). Table 2 provides a quick mapping of redo-rate to the minimum recommended redo log size:

TABLE 2: RECOMMENDED 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	32GB
> 50 MB/sec	64 GB

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.

The additional standby redo log eliminates the possibility of a standby database waiting on standby redo log. For example, if a primary database has two instances (threads) and each thread has three online log groups, then you should pre-configure 8 standby redo logs on the primary database and each standby database. Furthermore, if the primary or standby databases are not symmetrical Real Application Clusters (e.g. an 8-node primary Oracle RAC cluster and a 2-node standby Oracle RAC cluster), then the primary and standby databases should have an equal number of standby redo logs (based upon the largest cluster in the configuration) and all threads should be represented. The size of the standby redo logs must always be exactly the same size as the online redo logs on the primary.

Be sure to place single member standby redo log groups in the fastest available ASM diskgroup or storage volume The objective is to have standby log file write times that are comparable to log file I/O on the primary database optimal performance.

Configure Log Buffer

Set LOG_BUFFER to a minimum of 256 MB when using Oracle Data Guard with asynchronous redo transport and you have a high redo generation rate. Doing so will allow the asynchronous redo transport to read redo from the log buffer and avoid disk I/Os to online redo logs.

Determine Available Network Bandwidth

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. However, if you use peak redo rates as obtained from AWR reports to size your network you might not truly have enough bandwidth. If you take AWR snapshot every 30 minutes it could be that during that 30 minutes period you had really high redo rates for 10 minutes and moderate to low rates for the remainder of the time. This would average down the redo rate as seen in the AWR report. To determine peak redo rate for each log use the following query:

```
select thread#,sequence#,blocks*block_size/1024/1024 MB,(next_time-first_time)*86400
sec, blocks*block_size/1024/1024)/((next_time-first_time)*86400) "MB/s" from
v$archived_log
where ((next_time-first_time)*86400<>0)
and first_time between to_date('2015/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
and dest_id=2 order by first_time;
```

You should see output similar to the following:

```
   THREAD#  SEQUENCE#         MB        SEC       MB/s
---------- ---------- ---------- ---------- ----------
         2       2291 29366.1963        831  35.338383
         1       2565 29365.6553        781 37.6000708
         2       2292 29359.3403        537  54.672887
         1       2566 29407.8296        813 36.1719921
         2       2293 29389.7012        678 43.3476418
         2       2294 29325.2217       1236 23.7259075
         1       2567 11407.3379       2658 4.29169973
         2       2295 29452.4648        477 61.7452093
         2       2296 29359.4458        954 30.7751004
         2       2297 29311.3638        586 50.0193921
         1       2568 3867.44092       5510 .701894903
```

The above query output tells us that we must accommodate for a peak redo of 61MB/sec. In general we recommend adding an additional 30% on top of the peak rate. Alternatively you can refer to the EM performance page and refer to the redo rate history.

If the network is to be shared with other applications it is important to determine how much bandwidth is available for Data Guard transport. Typically with shared network we recommend:

- During peak periods run tcp bandwidth measuring tools, such as iperf or qperf to determine how much bandwidth is available.
- Work with the network team to apply Quality of Service rules on tcp traffic to guarantee that the targeted database has sufficient network bandwidth allocated.

More information on using these utilities is provided in the sections below.

Configure Sufficient Resources for Optimal System Performance

Sufficient resources, such as CPU, log file I/O on both the primary and standby databases and for network bandwidth between primary and standby locations, are critical to the performance of an asynchronous Data Guard configuration.

Maximum Achievable Rates

Testing was performed to determine the maximum rate that the ASYNC transport could achieve on a per instance basis (e.g. a 4-node Oracle RAC database would have four Oracle instances, each with its own ASYNC process and each capable of transmitting the maximum rate). The testing results showed:

For 11.2:

- The default configuration provides for approximately 400MB/sec for each Async process. This scales by adding more instances provided you have ample network bandwidth.
- The rate can be increased to nearly 500MB/sec by setting _log_read_buffer_size=256 and _log_archive_buffers=256 on both the primary and standby. The drawback of setting these parameters is they can increase your role transition timings by 5 to 10 seconds.

For 12.1:

- Similar to 11.2 the default rate is approximately 400MB/sec. Again, this also scales by adding more instances provided you have ample network.

» Increasing the max_io_size parameter setting to 8MB pushes the maximum rate to 550MB/sec.

Detecting a Transport Lag

Given enough resources, in particular network bandwidth, async transport can maintain pace with very high workloads. In cases where resources are constrained the async transport can begin to fall behind resulting in a growing transport lag on the standby. A transport lag is the amount of data, measured in time that the standby has not received from the primary. Determining transport 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='transport lag';
```

Diagnosing a Transport Lag

The most common cause of a transport lag is switching logs too fast which will cause a quick spike in a lag. We typically see this when ASYNC is working in log 100 (for example) along with lgwr and then the log switches to 101 and then to 102 quickly while ASYNC stays behind in 100. ASYNC will skip 101 and jump straight to 102. At that point we wait for 101 to be archived locally after which we ship it remotely. During that whole period the lag will jump up. To properly size the online redo logs consider the following:

» Use a minimum of three redo log groups: this helps prevent the log writer process (LGWR) from waiting for a group to be available following a log switch.

» All online redo logs and standby redo logs are equal size.

» Use redo log size >= peak redo rate x 10 to 20 minutes

» Locate redo logs on high performance disks.

» Place log files in a high redundancy disk group, or multiplex log files across different normal redundancy disk groups, if using ASM redundancy.

» Never have multiple members for standby redo log groups.

Another common reason for a transport lag is exceeding the interface bandwidth on either the primary or standby side. Spreading the transport across several standby instances only helps with the case where a single standby network interface is getting overloaded. Take for example an eight node primary database with each instance producing 20MB/sec sending redo to a single standby instance with a 1GigE interface card. In this case it would be helpful to spread transport over several standby instances because the volume transmitted by the primary will exceed a single standby network interface's ability to keep up (160MB/sec being sent but can only receive 110MB/sec). Determine the rate at which data is being sent over interfaces using the OS command sar –n DEV.

If we are not switching logs frequently and have not exceeded the bandwidth of the primary or standby interfaces then the most likely cause of transport lag is insufficient network bandwidth between the primary and standby. Typically networks are shared with other applications and it could be that other applications using the network have temporarily consumed the bandwidth needed for the standby to maintain pace with the primary. We can use utilities such as iperf or qperf to measure available bandwidth at any given point in time. The best method for doing so is outlined below:

1. When a transport lag is occurring temporarily disable the remote archive destination that is shipping with ASYNC. If we leave the destination active then we will be measuring what network bandwidth is available in addition to ASYNC shipping instead of what is available for ASYNC to utilize. Typically networks will try and accommodate multiple streams by reducing the rate of other streams.

2. On the standby start the qperf daemon. For example: `qperf -lp 5001`

3. On the primary connect to the standby qperf daemon and send a fixed about of data for a fixed amount of time. For example: `qperf 192.168.77.97 -lp 5001 -m 10m -t 10 tcp_bw`

4. Run multiple instances of the command in step 3 to determine the available bandwidth available to ASYNC. Compare that value to the peak redo rate seen during that log switch using the query described earlier in the best practice section.

5. Re-enable the remote archive destination to resume ASYNC shipping.

Another tool to utilize in diagnosing transport lag is event 16421 (alter system set events '16421 trace name context forever, level 1';). This dynamic event should be set on all primary and standby instances and will dump statistics into the tt00 or nsa trace file once shipping for a given sequence has completed. Looking in the trace file will be krsb_end stats at the beginning and end of the file. The stats at the end of the file will provide insight into where async shipping was spending time. For example:

```
krsb_end: Begin stats dump for T-1.S-227
  max number of buffers in use             10
  Operation elapsed time (micro seconds)   343768557
  File transfer time (micro seconds)       343768545
   DESTINATION 2 - OCI REQUEST
    Total count  - OCI REQUEST             185937
    Total time   - OCI REQUEST             815196
    Average time - OCI REQUEST             4
   LOG_ARCHIVE_DEST_2 - NETWORK SEND
    Total count  - NETWORK SEND            184762
    Total time   - NETWORK SEND            5066131
    Average time - NETWORK SEND            27
    Total data buffers queued              184761
    Total data buffers completed           184761
    Total bytes written                    3878883840
    Average network send size (blocks)     41
    Average network send buffers           1.00
    Average buffer turnaround time         6658
    Throughput (MB/s)                         10.76
   DESTINATION 2 - NETWORK NO-STALL REAP
    Total count  - NETWORK NO-STALL REAP   6108
    Total time   - NETWORK NO-STALL REAP   76132
    Average time - NETWORK NO-STALL REAP   12
   DESTINATION 2 - NETWORK STALL REAP
    Total count  - NETWORK STALL REAP      1175
    Total time   - NETWORK STALL REAP      116934633
    Average time - NETWORK STALL REAP      99518
    Total network layer time               122892092
    Percentage of time in network          35.75
    -----
    Total count  - DISK READ               184761
```

```
    Total time   - DISK READ               5298978
    Average time - DISK READ               28
    Total count  - BUFFER RELEASE          184761
    Total time   - BUFFER RELEASE          256830
    Average time - BUFFER RELEASE          1
    Total disk layer time                  5555808
    Percentage of time in disk layer       1.62
    -----
    Total count  - SLEEP                   181112
    Total time   - SLEEP                   196488753
    Average time - SLEEP                   1084
    Total DG layer time                    215320645
    Percentage of time in DG layer         62.64
krsb_end: End stats dump
```

The above output was taking from a test run where no transport lag was seen. We see that we spent 35% of the time on the network, 1% of the time in the disk layer, and 63% of the time in the DG layer, mainly due to sleeps. If we were to see a lag due to network congestion we would expect the time spent in the network layer to increase and the time spent in the DG layer to decrease.

For example, when we are limited by network bandwidth you would see something similar to the following for the time spent in each layer:

```
    Percentage of time in network          63.21
    -----
    Percentage of time in disk layer       1.62
    -----
    Percentage of time in DG layer         35.17
```

Tuning

Data Guard architecture is very streamlined and efficient however like any application, there are reasonable prerequisites needed to achieve satisfactory performance:

Primary Database

- Sufficient CPU utilization for LGWR to post foregrounds efficiently
- Sufficient I/O bandwidth so local log writes maintain low I/O latency during peak rates
- Network interfaces that can handle peak redo rate volumes combined with any other network activity across the same interface
- Primary AWR, ASH and OSwatcher data that is gathered for tuning and troubleshooting

Network:

- Sufficient network bandwidth to support peak redo rates (steady state and when resolving gaps) combined with any other network activity that shares the same network. Please note that your point to point network bandwidth will be throttled by the network segment, switch, router, interface with the lowest network bandwidth. If you have 10gigE for 90% of your network route and your existing switch or network interface only supports 1 GigE, then your maximum network bandwidth is 1 GigE.
- Netstat and/or any network monitoring stats should be gathered

Note: The top network problems encountered are inconsistent network response and insufficient network bandwidth.

Standby Database

- Sufficient CPU utilization for RFS (the Data Guard process that receives redo at the standby database) to efficiently write to standby redo logs.
- Sufficient I/O bandwidth to enable local log writes to maintain low I/O latency during peak rates.
- A network interface that can receive the peak redo rate volumes combined with any other network activity across the same interface
- Standby statspack, ASH and OSwatcher data should be gathered

Note: The top problem encountered with the standby database is poor standby log write latency due to insufficient I/O bandwidth.

Monitoring System Resources

The following provides information on how to monitor system resources on primary and standby hosts.

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.

Monitoring network:

The ip -s link command displays network statistics and errors for all network devices, including the numbers of bytes transmitted (TX) and received (RX). The dropped and overrun fields provide an indicator of network interface saturation.

The ss -s command displays summary statistics for each protocol.

To monitor the current rate being transmitted via an interface use the sar –n DEV command.

Conclusion

Data Guard and Active Data Guard asynchronous redo transport provides near zero data loss protection during database, cluster, and site outages as well as natural disasters and other events that can impact a widespread geography. Data Guard's unique integration with the Oracle Database combined with MAA best practices described in this paper provides the best possible data protection with optimal performance.

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

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

Authors: Michael Smith
Contributors: Joe Meeks, Lawrence To

CONNECT WITH US


blogs.oracle.com/oracle


facebook.com/oracle


twitter.com/oracle


oracle.com

Hardware and Software, Engineered to Work Together


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