Oracle Maximum
Availability Architecture

An Oracle White Paper
October 2013

Best Practices for Database Consolidation On Exadata Database Machine

Executive Overview

Consolidation can minimize idle resources and lower costs when you host multiple schemas, applications or databases on a target system. Consolidation is a core enabler for deploying Oracle Database on public and private clouds.

This paper provides the Exadata Database Machine (Exadata) consolidation best practices to setup and manage systems and applications for maximum stability and availability.

Introduction

Within IT departments, consolidation is one of the major strategies that organizations are pursuing to achieve greater efficiencies in their operations. Consolidation allows organizations to increase the utilization of IT resources so that idle cycles can be minimized. This in turn lowers costs because fewer resources are required to achieve the same outcome. For example, applications that experience peak load at different times of the day can share the same hardware, rather than using dedicated hardware which will be idle during the non-peak periods.

Database consolidation can be achieved in many different ways depending on the systems and circumstances involved. Running multiple application schemas in one single database, hosting multiple databases on a single platform, or a hybrid of the two configurations are all valid forms of database consolidation.

Exadata is optimized for Oracle Data Warehouse and OLTP database workloads, and its balanced database server and storage grid infrastructure makes it an ideal platform for database consolidation. Oracle Database, storage and network grid architectures combined with Oracle resource management features provide a simpler and more flexible approach to database consolidation than other virtualization strategies (for example. hardware or operating system virtualization). Exadata currently relies on the simplicity of Oracle resource management for efficient database consolidation. Exadata does not support virtual machines or Solaris Zones today.

This white paper describes the recommended approach for consolidating on Exadata that includes the initial planning, setup, and management phases required to maximize stability and availability when supporting multiple applications and databases that use shared resources.

This white paper complements other Exadata and Maximum Availability Architecture (MAA) best practices, with references to the appropriate paper provided where relevant. This paper does not address database consolidation on SPARC Supercluster, Oracle Exalogic, virtual machines, or Oracle 12c Multitenant architecture. Separate Oracle 12c Exadata MAA consolidation papers are targeted for late 2013 and early 2014.

Planning for Exadata Consolidation

1. Define High Availability (HA), planned maintenance, and business requirements.

   When consolidating on Exadata, a few core principles should be applied. First, the availability and planned maintenance objectives of the target databases should be similar. If this is not the case, then inevitably one of the applications will be compromised in some way. For example, if mission critical applications share the same environment with development or test systems, then the frequent changes made in development and test systems will impact the availability and stability of the mission critical applications. Oracle follows the approach of consolidating target databases of like service levels due to the efficiency and relative simplicity of managing a common infrastructure (for example, Orade Grid Infrastructure (consisting of Orade Clusterware and Oracle ASM) and Exadata Storage Cells) and operating system environment. This strength can become a liability if the targeted applications do not have similar service level requirements.

   Businesses should first define these key HA requirements: recovery time objective (RTO or application's target recovery time), recovery point objective (RPO or application's maximum data loss tolerance), disaster recovery (DR) recovery time, and the allotted planned maintenance windows.

   Other considerations, such as performance objectives with estimated peak, average and idle workload periods, system requirements, and security and organization boundaries must also be considered to ensure compatibility between the various application candidates.

2. Categorize databases into groups.

   Group databases planned for consolidation based upon the HA requirements determined in step 1. For example, databases could be categorized into the following groups:

   - Gold: Mission critical revenue generating, customer-facing databases which requires zero or near zero RTO and RPO requirements
   - Silver: Business critical databases with slightly higher RTO and RPO requirements
   - Bronze: Other non-critical production databases or development and test databases

   Each group can be further sub-divided, if necessary, so that all applications have non-conflicting HA requirements and planned maintenance windows.

3. Create Exadata Hardware Pools.

The term ***Hardware Pool*** describes the machine or group of machines used as the target consolidation platform. An enterprise might create multiple Hardware Pools to make each consolidation target platform more manageable. The recommended minimum Hardware Pool size is Exadata Half Rack and the maximum recommended Hardware Pool size is two Exadata Database Machines Full Racks (plus additional Exadata storage expansion racks if required). Hardware Pools that fall within this range are the most common Exadata configurations for consolidation and provide sufficient capacity to efficiently achieve objectives for database consolidation.

There is also the option of deploying a smaller Hardware Pool consisting of an Exadata X3-2 quarter rack or eighth rack for entry-level consolidation. This option is acceptable for critical applications if a standby database on separate Exadata is deployed. A slight HA disadvantage of an Oracle Exadata Database Machine X3-2 quarter or eighth rack is that there are insufficient Exadata cells for the voting disks to reside in any high redundancy disk group which can be worked around by expanding with 2 more Exadata cells. Voting disks require 5 failure groups or 5 Exadata cells; this is one of the main reasons why an Exadata half rack is the recommended minimum size. There is a risk of Oracle ASM and Oracle Clusterware failing when the ASM disk group (where the voting disks reside) is dismounted due to storage failures. In this situation, all the databases fail but they can be restarted manually by following My Oracle Support (MOS) note 1339373.1 (Refer to section on *impact due to loss of DBFS_DG*).

It is also possible to deploy a Hardware Pool consisting of eight or more Oracle Exadata Database Machine Full Racks. This is not recommended because the scheduling and management complexity in such a consolidated environment would increase to the point where it offsets the benefits of consolidation.

It is also possible to have multiple Hardware Pools coexist within one Oracle Exadata Database Machine by partitioning nodes and Exadata storage cells (aka Exadata cells). This configuration is also discouraged because a partitioning approach can result in less efficient resource utilization and has the following disadvantages:

- Limits access to the entire server and storage grid bandwidth
- Increases complexity
- Lacks complete fault and maintenance isolation with common components such as the InfiniBand fabric, Cisco switches and the physical rack itself

4. Map groups of databases and applications to specific Hardware Pools.

Continuing the above example, each group of databases and applications would be deployed on their own Hardware Pool: GOLD, SILVER, and BRONZE. You should avoid consolidating databases from different categories within the same Hardware Pool. If you have a mixed purpose hardware pool containing multiple groups such as GOLD, SILVER and BRONZE to save costs, you have to manage the Hardware Pool according to the highest database category. An example of this mixed purpose hardware pool is when consolidating GOLD Active Data Guard standby databases with some BRONZE test databases. In mixed purpose hardware pools there are still a

lot shared components such as InfiniBand network and software, Exadata storage cells and Orade Grid Infrastructure. Furthermore, mixed purpose hardware pools will contain databases with conflicting HA requirements and planned maintenance windows; making operational management more challenging.

If a group requires more capacity than provided by a single Hardware Pool, you can expand by upgrading the memory or adding more storage or inter-racking more Oracle Exadata Database Machines. If you have reached the maximum Hardware Pool size, then divide the target databases in that group into two separate groups and deploy each group on its own Hardware Pool. Any single database that requires more capacity than offered by two Oracle Exadata Database Machine full racks plus additional Exadata storage expansion racks (the maximum recommended Hardware Pool size) should not be considered a viable candidate to benefit from consolidation; such databases are best deployed on a dedicated duster comprised of as many Exadata systems as needed.

The following table provides an example of how different High Availability requirements may require different architectures using Exadata Hardware Pools.

TABLE 1. HIGH AVAILABILITY REQUIREMENTS AND THEIR RECOMMENDED ARCHITECTURES

HIGH AVAILABLILITY REQUIREMENTS	RECOMMENDED ARCHITECTURE
GOLD Mission Critical applications with the following requirements: • RTO < 1 minute • RPO=0 • Planned Maintenance 4 hours/quarter (weekend)	Three critical Hardware Pools consisting of • Primary Critical using RAC • LocalDataGuardCritical using Active Data Guard • RemoteDataGuardCritical using Data Guard • Test • RTO/RPO based on Data Guard failover plus application failover.
SILVER Critical applications with the following requirements: • 1 minute > RTO < 2 hours • 2 seconds > RPO < 2 hours • Planned Maintenance 8 hours/quarter (weekend)	Two critical Hardware Pools consisting of • Primary Critical using RAC One or RAC • RemoteDataGuardCritical using Active Data Guard • Test • RTO/RPO based on Data Guard failover plus application failover. Application failover can consume most of the time.
BRONZE Standard applications with the following requirements: • 12 hours > RTO < 24 hours • 12 hours > RPO < 24 hours	One standard Hardware Pool consisting of • STDPOOL1 using Single instance or RAC One • RTO/RPO based on restoring and recovering from backups. Backup frequency and restore

• Planned Maintenance 48 hours/quarter	rates impact RTO and RPO.
Development applications with the following requirements: • High availability for development usage • 24 hours > RTO < 72 hours • Planned Maintenance 48 hours/month	One non-production Hardware Pool consisting of • DEVPOOL1 using single instance or RAC One • RTO/RPO based on restoring and recovering from production backups. Backup frequency and restore rates impact RTO and RPO.
Test applications with the following requirements: • Test system to validate system changes and patches, to evaluate new functionality, performance and HA • This is a recommendation for all production applications	Recommended to be identical to production environment OR Minimum Exadata Eighth Rack • TESTPOOL1

5. Evaluate sizing requirements before migrating to an Exadata Hardware Pool.

   If you are migrating databases from an existing system, then extrapolate the current CPU, I/O and memory utilization rates, and obtain future growth projections from the business. You can then leverage these calculations and evaluate how many databases and applications can reasonably fit in a Hardware Pool. For more information on sizing, please contact Oracle consulting and Orade Advanced Customer Support Services (ACS).

   Other considerations include:

   - Reserve system capacity to accommodate various high availability (HA) and rolling upgrade activities such as Oracle Real Application Clusters (Oracle RAC) and Exadata cell rolling upgrade activities.
   - Reserve system capacity for critical Hardware Pools to maximize stability. For example, it is a common best practice for critical Hardware Pools to be configured with 25% of the resources unallocated to accommodate spikes.
   - Remember that less data storage space is available when using Oracle Automatic Storage Management (ASM) high redundancy disk groups for critical Hardware Pools.
   - Evaluate whether business or workload cycles between applications and databases allow for further consolidation. For example, application A may have a peak workload during times when application B is relatively idle.

6. Gather accurate performance requirements for each application and database.

   Gather accurate performance expectations for throughput and response time for each application. Use Orade Enterprise Manager (EM) Grid Control or EM Cloud Control to monitor key application metrics, including a history of the application, database, and system performance statistics. This data will be required to debug any future performance issues.

Setting Up and Managing Key Resources for Stability

After the initial planning and sizing exercises, you will transition to setting up the Exadata Hardware Pool. This section provides recommendations from the initial deployment to specific configuration settings as you consolidate your databases on Exadata.

Recommended Storage Grid (Disk Group) Configuration

The recommended storage configuration is **one shared Exadata storage grid** for each Hardware Pool. This storage grid contains all Exadata cells and Exadata disks, and is configured with either ASM high or normal redundancy (ASM redundancy is discussed further below).

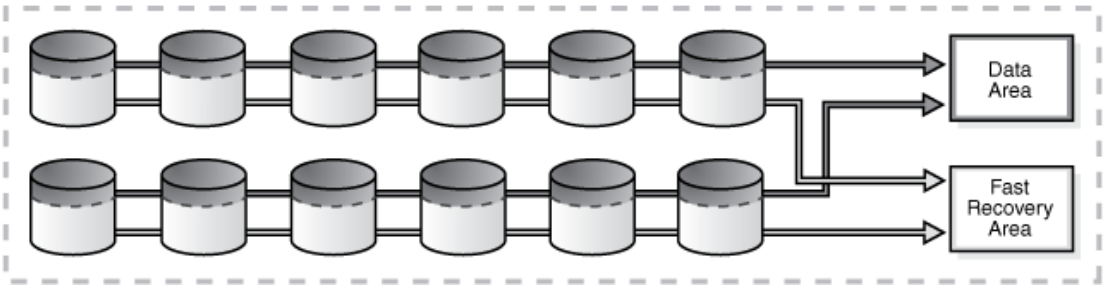

Figure 1. Shared Exadata Storage Grid

The major benefits are:

- Simpler and easier to manage
- Most used and most validated configuration
- Balanced configuration where applications have full access to the I/O bandwidth and storage
- Tolerance for failures and rolling upgrade

Managing one shared Exadata storage grid is simpler with lower administrative costs. Space and bandwidth utilization are also more efficient with shared storage. If more storage space and I/O bandwidth are required than available in an Exadata full rack, you can add an Exadata storage expansion rack or you can migrate an application to another Hardware Pool.

If this is a critical Hardware Pool, it is strongly recommended to use ASM high redundancy for the DATA and RECO disk groups for best tolerance from storage failures especially during Exadata storage cell rolling upgrade and other maintenance activities. For more information on DATA and RECO disk group configurations, MAA storage grid configuration and about Exadata quarter rack and eighth rack restrictions refer to "About Orade ASM for Maximum Availability" in the *Oracle Exadata Storage Server Software User's Guide*. If you are space constrained, you may consider using ASM normal redundancy for both DATA and RECO disk groups if you have also deployed a standby Hardware Pool (standby pool) using Orade Data Guard. The standby pool enables the most comprehensive data corruption protection and fast failover in the event of a database, duster or storage failure, mitigating the HA and data protection risk of not utilizing high redundancy on the primary Hardware Pool.

If you are space constrained and you are not able to deploy a standby pool, a second option is to use high redundancy for the DATA disk group, and normal redundancy for the RECO disk group. Note that this has the potential of compromising availability and simplicity if there is a storage failure. Refer to My Oracle Support (MOS) note 1339373.1 for more information on how to recover should you lose either DATA or RECO ASM disk groups.

You should use the disk group configuration resulting from the OneCommand configuration process (described in *Exadata Database Machine Owner's Guide*). By default, the first high redundancy disk group stores the online logs, controlfiles, spfiles, clusterware, and voting devices. The DATA disk group stores database files and the RECO disk group contains the recovery related files for the Fast Recovery Area (FRA). If application isolation is required, separate DATA and RECO disk groups can be created. Refer to Security and Management Roles section or Oracle Exadata Database Machine Consolidation: Segregating Databases and Roles MAA paper.

Furthermore, when creating the ASM diskgroups, ensure that the ASM disk group COMPATIBLE.RDBMS attribute is set to the minimum Oracle Database software version on the Hardware Pool.

Recommended Database Grid (Cluster) Configuration

The recommended setup for Oracle Grid Infrastructure (which includes Oracle Clusterware and Oracle ASM) is to use **one cluster per Hardware Pool**. Oracle Database services that are managed by Oracle Clusterware should be used to further load balance and route applications to specific Oracle database instances in the cluster. The key benefits are:

- Simpler and easier to manage
- Balanced configuration where applications have full access to the database CPU and memory bandwidth if required or desired
- Tolerance for failures
- Oracle RAC and Grid Infrastructure rolling upgrade capabilities

As a specific application resource grows or shrinks, the service can be routed and load balanced to available database instances and nodes easily and transparently.

The Oracle Grid Infrastructure software version must be equal to or higher that the highest version of the Oracle RAC software you plan to use in the cluster. The Grid Infrastructure software can be upgraded anytime but planning is required to reduce the patching frequency of Grid Infrastructure when individual databases upgrade at different schedules. All Grid Infrastructure upgrades should be rolling.

For critical Hardware Pools, allocate a Data Guard Hardware Pool (standby pool) to protect from cluster failures, database failures, data corruptions, and disasters. You can also switch over to the standby pool for site, system or database maintenance.

> **NOTE:** The maximum number of database instances per cluster is 512 for Oracle 11*g* Release 1 and higher. An upper limit of 128 database instances per X2-2 or X3-2 database node and 256

database instances per X2-8 or X3-8 database node is recommended. The actual number of database instances per database node or cluster depends on application workload and their corresponding system resource consumption.

Recommended Parameter Settings

The following parameter settings are particularly important for each Hardware Pool. They help allocate and limit system resources efficiently. Periodic monitoring is required to adjust and tune the settings as workloads change or databases are added or dropped.

Operating System Parameters

- **If PageTables in /proc/meminfo is set to more than 2% of the physical memory size, then set the operating system parameter HugePages to the sum of all shared memory segments. (LINUX ONLY)**

  HugePages is a mechanism that allows the Linux kernel to utilize the multiple page size capabilities. Linux uses pages as the basic unit of memory, where physical memory is partitioned and accessed using the basic page unit. The default page size is 4096 bytes. Hugepages allows large amounts of memory to be utilized with a reduced overhead. Linux uses Transaction Lookaside Buffers (TLB) in the CPU architecture. These buffers contain mappings of virtual memory to actual physical memory addresses. So a system with a large page size provides higher density of TLB entries, thus reducing pagetable space. Furthermore, each process uses a smaller private pagetable and those memory savings can grow significantly as the process count grows.

  HugePages is generally required if PageTables in /proc/meminfo is more than 2% of physical memory. When set, HugePages should equal the sum of the shared memory segments used by all the database instances. When all the database instances are running, the amount of shared memory being used can be calculated by analyzing the output from the *ipcs -m* command. MOS note 401749.1 provides a script which can be used to determine the amount of shared memory in use. MOS note 361468.1 describes how to set HugePages.

  Starting in 11.2.0.2, setting the database initialization parameter USE_LARGE_PAGES=ONLY on all instances prevents any instance from starting unless sufficient HugePages are available.

  > **NOTE:** The value to use for HugePages must be recalculated if the database parameter SGA_TARGET changes, or whenever the number of database instances changes. Hugepages can only be used for SGA, so do not over-allocate. Also, the database parameters MEMORY_MAX_TARGET and MEMORY_TARGET are not compatible when HugePages are enabled.
  >
  > **NOTE:** On Solaris, HugePages are automatically configured and used via intimate shared memory (ISM).

- **Set the number of shared memory segments (kernel.shmmni) greater than the number of databases.**

  The number of shared memory identifiers should be set greater than the number of database instancess running on the node. Check the setting for kernel.shmmni using the sysctl command (for example, /sbin/sysctl -a | grep shmmni). If necessary, make adjustments by setting kernel.shmmni in /etc/sysctl.conf on Linux systems. The SHMMNI default setting (4096) should accommodate all cases.

- **Set the maximum shared memory segment size (kernel.shmmax) to 85% of physical memory size, which is the default**

  The maximum shared memory segment size should be set to 85% of the database server physical memory size. Check the setting for kernel.shmmax using the sysctl command. If necessary, make adjustments by setting kernel.SHMMAX in /etc/sysctl.conf on Linux systems.

- **Set the maximum total number of system semaphores (SEMMNS) greater than the sum of all database processes.**

  The maximum number of semaphores in the system must be greater than the sum of the processes, as specified by the database PROCESSES parameter, for all of the database instances running on the system. Check the setting for kernel.sem using the sysctl command. The setting for kernel.sem contains a list of four numeric values. The setting for the total number of semaphores on the system, also known as SEMMNS, is the second value in the list. If necessary, make adjustments by setting kernel.sem in /etc/sysctl.conf. The SEMMNS default setting (60000) should accommodate all cases.

- **Set the maximum number of semaphores in a semaphore set (SEMMSL) greater than the largest number of processes in any single database.**

  The maximum number of semaphores in a semaphore set should be set greater than the largest number of processes for any single database instance running on database server. Check the setting for kernel.sem using the sysctl command. The setting for the maximum number of semaphores in a semaphore set, also known as SEMMSL, is the first value in the list. If necessary, make adjustments by setting kernel.sem in /etc/sysctl.conf on Linux systems.

Database Memory Settings

For Exadata, the system comes delivered with the following physical memory:

- Exadata X3-2 has 256 GB of memory.
- Exadata X3-8 has 2 TB per database server.
- Exadata X2-2 based on the Sun Fire X4170 Oracle Database Servers (also known as V2) has 72 GB per database server
- Exadata X2-2 has 96 gigabytes (GB) of memory in the default configuration, with an option to expand to 144 GB of memory (with the Exadata memory expansion kit).
- Exadata X2-8 has 1 terabyte (TB) (with the X4800) or 2 TB (with the X4800M2) per database server.

Excessive memory paging or "page outs" should always be avoided because it leads to poor application performance and node instability. Monitor system memory periodically for page out activity.

NOTE:

- Use the Linux *free* command to display information about free and used memory on the system.
- Use the *vmstat* command to monitor for zero or very low page-outs.
- Use the operating system *ps top* (Linux),or *prstat* (Solaris) commands to view more process-level information, induding private and shared memory utilization.

  Database queries on V$PROCESS can alert you when PGA_USED_MEM exceeds an application threshold. In some cases, you may have to kill the process to prevent node instability affecting all databases and applications on that database node. For example, it may be reasonable to kill any process consuming more than 1 GB of PGA memory for OLTP and 10 GB for Data Warehouse reporting. The exact threshold depends on your known application behaviour and requirements but should be less than PGA_AGGREGATE_TARGET.

  Here is an example query and operational flow:

  ```
  SELECT s.inst_id, s.sid, s.serial#, p.spid, s.username,  s.program, p.pga_used_mem
     FROM   gv$session s JOIN gv$process p ON p.addr = s.paddr AND p.inst_id = s.inst_id
     WHERE  s.type != 'BACKGROUND'  and s.program not like '%(P%'   and

     p.pga_used_mem >  <APPLICATION_MEMORY_THRESHOLD>
     order by s.inst_id, s.sid, s.serial#;
  ```

  Execute the above query every 15 minutes, substituting the value of the application memory threshold for <APPLICATION_MEMORY_THRESHOLD>.

  Investigate any suspicious process that is using an unfair share of memory. If the culprit process exceeds the memory thresholds and is not critical, it may be important to terminate the process to maintain node stability and performance for other clients. For example, you could use the SQL statement `ALTER SYSTEM KILL SESSION '<S.SID>,<S.SERIAL#>,@<S.INST_ID>'` to terminate the session, or use the operating system command KILL -9 <SPID> as a last resort.

For initial deployment and migration of applications into the Hardware Pool, use the following formula to ensure you have allowed enough memory for SGA, PGA and individual server processes. If an application requires more SGA or PGA memory allocations to meet their performance requirements and there's insufficient physical memory, then leverage another Hardware Pool. For critical Hardware Pools, we recommend an even more conservative approach by not exceeding 75% physical memory per database node. You should avoid memory swapping for all cases.

OLTP applications:

SUM of databases (SGA_TARGET + PGA_AGGREGATE_TARGET) + 4 MB * (Maximum PROCESSES) < Physical Memory per Database Node

DW/BI applications:

SUM of databases (SGA_TARGET + 3 * PGA_AGGREGATE_TARGET) < Physical Memory per Database Node

NOTE:

Memory per process varies, but 4 MB allocation has been observed in our application MAA studies on Exadata. Re-adjust this value in your calculations after monitoring actual process memory utilization.

Continue to monitor memory statistics from Automatic Workload Repository (AWR) reports over time to get the most accurate calculation. For example, the following tables are taken from an AWR report depicting PGA growth.

TABLE 2. PGA MEMORY

COMPONENT	BEGIN SNAP SIZE (MB)	CURRENT SIZE (MB)	MIN SIZE (MB)	MAX SIZE (MB)	OPER COUNT	LAST OP TYP/MOD
PGA Target	16,384.00	**16,384.00**	**16,384.00**	**29,384.00**	0	STA/

TABLE 3. MEMORY STATISTICS

	BEGIN	END
Host Mem (MB)	96,531.5	96,531.5
SGA Use (MB)	24,576.0	24,576.0
PGA Use (MB)	577.7	578.2
% Host Mem used for SGA+PGA	26.06	26.06

PGA_AGGREGATE_TARGET is not a hard limit as observed in the above example. In some Data Warehouse applications, cumulative PGA memory usage three times PGA_AGGREGATE_TARGET has been observed. For OLTP applications, the potential overrun is generally much lower unless the application is using a lot of PL/SQL. Query V$PGASTAT for the instance-level statistics on the PGA memory usage including: current target setting, total in use, total allocated, and maximum allocated. If PGA growth is being observed, then you must determine if this is expected and if sufficient memory on the database server is available for this growth. If not, you might have to increase PGA_AGGREGATE_TARGET and reduce SGA_TARGET settings to accommodate for the larger PGA memory allocation or essentially move the application to another database server or Hardware Pool.

Starting in Oracle 12c, the PGA_AGGREGATE_LIMIT initialization parameter enables you to specify a hard limit on PGA memory usage. In Oracle 11g, you can use add undocumented event to limit PGA usage. For example, you can set the following the spfile

- event="10261 trace name context forever, level <MEM in KB>" or
- ALTER [SYSTEM | SESSION] SET EVENTS '10261 TRACE NAME CONTEXT FOREVER, LEVEL <MEM IN KB>'

When the limit is exceeded, the following error is thrown

  - For 11.2.0.4 and 12.1.0.1+, ORA-10260
  - For 11.1 - 11.2.0.3, ORA-600 [723]

Database CPU Settings and Instance Caging

Instance caging is an important tool for managing and limiting CPU utilization per database. Instance caging is used to prevent runaway processes and heavy application loads from generating a very high system load, resulting in an unstable system and poor performance for other databases.

To enable instance caging, do the following for each instance on the server:

1. Enable the Oracle Database Resource Manager by assigning a resource plan to RESOURCE_MANAGER_PLAN initialization parameter. The resource plan must have CPU directives to enable instance caging. See Oracle Database Administration's Guide 11g Release 2 "Enabling Oracle Database Resource Manager and Switching Plans" for instructions. If you are not planning on managing workloads within a database, you can simply set RESOURCE_MANAGER_PLAN to "DEFAULT_PLAN".

   ```
   SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'DEFAULT_PLAN'
   SID='*' SCOPE='BOTH';
   ```

2. Set the CPU_COUNT initialization parameter to the maximum number of CPUs the instance should use at any time. By default, CPU_COUNT is set to the total number of CPUs on the server. For hyper-threaded CPUs, CPU_COUNT includes CPU threads. CPU_COUNT is a dynamic parameter, so its value can be altered at any time but it is best set at instance startup because the CPU_COUNT parameter influences other Oracle parameters and internal structures (for example, PARALLEL_MAX_SERVERS, buffer cache, and latch structure allocations).

   The minimum value for CPU_COUNT is 2. Note this does not mean the database instance will fully use 2 CPUs, just that it will have access to 2 CPU threads. In situations where instances in operation require fewer than 2 full CPUs, it is acceptable to use the over-subscribed approach described below.

   Setting CPU_COUNT to 1 causes database instances to be susceptible to hangs or poor performance for the following reasons:

   - Queries that do not respond to CTL-C

- One thread is not a lot of processing power
- Process death followed by slow process deanup

The peak potential load for the server is equal to the sum of the CPU_COUNT parameters across all the database instances on the server. By comparing the peak potential load to the number of CPUs per server, you can assess the potential for resource contention between the database instances. Exadata has the following number of CPUs per server. Note that these counts pertain to CPU threads, not cores or sockets.

- Exadata X3-2 has 16 cores or 32 CPUs per database server
- Exadata X3-8 has 80 cores or 160 CPUs per database server
- Exadata X2-2 has 12 cores or 24 CPUs per database server.
- Exadata V2 has 8 cores or 16 CPUs per database server.
- .
- Exadata X2-8 has 64 cores or 128 CPUs for X4800 database server. Exadata X2-8 has 80 cores or 160 CPUs for X4800M2. database server.

The most important step for configuring instance caging is to determine the value of CPU_COUNT for each database instance and the sum of CPU_COUNT values across the server. There are two approaches:

- **Partitioned approach for mission-critical GOLD and SILVER Hardware Pools.** If the sum of the CPU_COUNT values of all the databases instances on the target database server does not exceed the number of CPUs on the server, then the server is partitioned. In this case, there should be no contention for CPU resources between database instances. However, if one database instance does not use its allocated CPU resources, then these CPU resources cannot be utilized by the other database instances.

  The advantage of the partitioned approach is that there is no CPU contention, but CPUs may be underutilized. The partitioned approach is therefore recommended for mission-critical databases in critical Hardware Pools.

  The specific sizing recommendation is as follows: by limiting the sum of CPU_COUNT values to less than 100% of the total number of CPUs on the server, each instance's CPU allocation should be derived by your sizing analysis and taking account maximum historical CPU usage.

  sum(CPU_COUNT) <= 100% * Total CPUs

- **Over-subscribed approach for non-critical BRONZE Hardware Pools.** If the sum of the CPU_COUNT values of all the database instances on the target database server exceeds the number of CPUs on the server, then the server is over-subscribed. In this case, if all databases are heavily loaded at the same time, then there will be some contention for CPU resources and degraded performance.

The advantage of the over-subscribed approach is better resource utilization, but with potential CPU contention. The over-subscribed approach is therefore recommended for test or non-critical database Hardware Pools whose peak periods do not coincide.

It is recommended to limit the sum of CPU_COUNT values to three times the number of CPUs.

sum (CPU_COUNT) <= up to 3 * Total CPUs

For example, using the formula above, the maximum value of the sum of CPU_COUNT values for all database instances on the target database server (an X3-2 with 32 CPUs) is 3 * 32 = 96. So, you can have 4 instances, each with CPU_COUNT set to 24, or 8 instances, each with CPU_COUNT set to 12, or all instances with a different value for CPU_COUNT but with the sum less than or equal to 96. With instance caging, the CPU_COUNT value guarantees that no single database instance can leverage more than their designated number of CPUs.

Once instance caging has been configured, you can use the scripts in MOS note 1362445.1 to monitor the actual CPU usage of each instance. With this information, you can tune instance caging by adjusting the CPU_COUNT value for each instance as needed. For very active Oracle RAC systems, you should allocate at least 2 CPUs to each database instance so the background processes (for example, SMON, PMON, LMS, and so on) continue to function efficiently. Also refer to MOS note 1340172.1 for the recommended patches for instance caging.

Process Settings

The following Oracle ASM and Oracle Database process settings and operational practices help you to avoid common process configuration pitfalls.

- **For the Oracle ASM instance, set PROCESSES= 50 * MIN (# database instances on db node+ 1, 11) + 10 * MAX (# database instances on db node – 10, 0)**

  The Oracle ASM initialization parameter PROCESSES needs to be adjusted when there is more than one instance per database node or when you add or remove instances. By using the above guideline, you can avoid Oracle ASM hitting errors attempting ASM allocation and deallocation requests. Using the above formula, five databases instances require 300 Oracle ASM processes, while 20 database instances require 650 Oracle ASM processes.

- **Limit PARALLEL_MAX_SERVERS**
  - X2-2 and X3-2: sum(PARALLEL_MAX_SERVERS) for all instances <= 240
  - X2-8 and X3-8: sum(PARALLEL_MAX_SERVERS) for all instances <= 1280

  PARALLEL_MAX_SERVERS specifies the maximum number of parallel execution processes and parallel recovery processes for an instance. As demand increases, an Oracle database increases the number of parallel execution processes up to PARALLEL_MAX_SERVERS. Ensure that each application can meet its performance requirements with the lower value. If PARALLEL_MAX_SERVERS is set too high, then memory resource shortages may occur during peak periods, which can degrade performance and destabilize the database node.

- **Limit Redo Apply Parallelism**

For Oracle Data Guard, the default redo apply parallelism is implicitly set to the number of CPUs which consumes system resources unnecessarily with negligible gain. Instead, explicitly use recovery parallelism of only 16 (or less) with the following command:

SQL> RECOVER MANAGED STANDBY DATABASE … PARALLEL 16.

- **Limit the number of processes and connections to the database servers**

  Having a lower process count or appropriate number of processes has many advantages such as avoiding or reducing memory, CPU and database latch contention, *log file sync* waits, and overall application failover times. The reduced process and connection count can also lead to better performance and throughput. Refer to the Real World Performance educational video on the correlation between processes and connections with performance at: http://www.youtube.com/watch?v=xNDnVOCdvQ0

  The Oracle performance experts have advised a conservative **active** process count[1] of 5 times CPU cores, or optionally, for CPU intensive applications such as Siebel use an active process count of 1 or 2 times CPU cores. By using one or more of the following techniques, you can lower the overall process count and improve performance:

  1. **Application connection pooling** with min and max settings

     Using application connection pooling to avoid the expensive process of opening and closing connections saves valuable system resources. By using min and max settings within the connection pool, you can achieve quick response time, efficient memory and CPU utilization and avoid logon storms. The overall number of connections can drop significantly without sacrificing any of your performance goals.

     A connection pool is a cache of database connection objects. The objects represent physical database connections that can be used by an application to connect to a database. At runtime, the application requests a connection from the connection pool. If the connection pool contains a connection that can satisfy the request, it returns the connection to the application. If no connections are found, a new connection is created and returned to the application. The application uses the connection to perform some work on the database and then returns the object back to the connection pool. The connection is then available for the next connection request.

     Connection pools promote the reuse of connection objects and reduce the number of times that connection objects are created. Connection pools significantly improve performance for

[1] Active process count can be obtained by querying V$SESSION where STATUS="ACTIVE" or monitoring V$ACTIVE_SESSION_HISTORY. Enterprise Manager's performance page provides all these details plus helpful insights to any potential contention.

database-intensive applications because creating connection objects is costly both in terms of time and resources. Tasks such as network communication, reading connection strings, authentication, transaction enlistment, and memory allocation all contribute to the amount of time and resources it takes to create a connection object. In addition, because the connections are already created, the application waits less time to get the connection.

Connection pools often provide properties that are used to optimize the performance of a connection pool. The properties control behaviors such as the minimum and maximum number of connections allowed in the connection pool or the amount of time a connection can remain idle before it is returned to the connection pool. The best configured connection pools balance quick response times with the memory consumed maintaining connections in the connection pool. It is often necessary to try different settings until the best balance is achieved for a specific application.

Furthermore, the configuration of application connection pooling will vary for each application and may not be available for some applications. Please refer to your application documentation for details.

2. Database Shared Servers

In an Oracle Database shared server architecture, a dispatcher directs multiple incoming network session requests to a pool of shared server processes, eliminating the need for a dedicated server process for each connection. An idle shared server process from the pool picks up a request from a common queue.

If the application tier cannot change, using the shared server architecture for many short requests and transactions dramatically reduces your overall process count. Check your application documentation for any restrictions or specific guidance about using the shared server architecture of Oracle Database.

When first evaluating database shared servers, review the following documentation and adopt these practices:

- "Configuring Oracle Database for Shared Server" in Oracle Database Administration Guide
- "Performance considerations for shared servers and dispatchers" in the Oracle Database Performance Tuning Guide
- Start with 20-30 shared servers per 500 sessions and then tune (slide 44 of Oracle Net Services best practices)
- Start with 1 dispatcher per 50-100 sessions and then tune
- For long running queries, reporting jobs, Data Guard transport, and administrative connections, continue to use dedicated servers. You can always have a mixture of shared servers and dedicated servers, which is useful when transitioning into using database shared servers.

- In the sqlnet.ora file, set USE_ENHANCED_POLL=on (slide 46 of Oracle Net Services best practices)

3. Database resident connection pooling (DRCP) – new in Oracle Database 11*g*

   DRCP uses a pooled server, which is the equivalent of a dedicated server process (not a shared server process) and a database session combined. The pooled server model avoids the overhead of dedicating a server process for every connection that requires database access for a short period of time. Clients obtaining connections from the DRCP connect to an Oracle background process known as the connection broker. The connection broker implements the DRCP pool functionality and multiplexes pooled servers among inbound connections from client processes.

   Although more efficient than shared servers, DRCP requires minor application code changes to get and release sessions. Refer to the "Database Resident Connection Pooling (DRCP)" white paper on OTN at

   http://www.oracle.com/us/products/database/oracledrcp11g-1-133381.pdf.

   Check your application documentation for any restrictions or specific guidance about using DRCP.

- **Exadata software 11.2.3.1 or higher can support up to 60,000 simultaneous connections originating from one or more database servers in a hardware pool. This implies that no more than 60,000 processes can simultaneously remain connected to a cell and perform I/O operations. The limit was 32,000 connections in release Exadata 11.2.2.4. Prior to Exadata 11.2.2.4, the limit was 20,000 connections. Upper limit target is dependent on memory or CPU consumption but approximately 7,500 processes per node for X2-2 or X3-2 and 30,000 processes per node for X2-8 or X3-8.**

  To calculate the number of processes per Hardware Pool, query each database including the Oracle ASM instance, the Database File System (DBFS) instance, and the application databases with the following:

  ```
  SQL> SELECT COUNT(1) FROM GV$PROCESS;
  ```

  Higher process counts can lead to memory paging, which results in node instability, CPU contention, database latch contention, and longer application failover or brownout times. Testing is very important before increasing the process count.

- **Oracle listeners can be configured to throttle incoming connections to avoid logon storms after a database node or instance failure.**

  The connection rate limiter feature in the Oracle Net Listener enables a database administrator (DBA) to limit the number of new connections handled by the listener. When this feature is enabled, Oracle Net Listener imposes a user-specified maximum limit on the number of new connections handled by the listener every second.

  Depending on the configuration, the rate can be applied to a collection of endpoints, or to a specific endpoint.

  This feature is controlled through the following two listener.ora configuration parameters:

  - CONNECTION_RATE_<*listener_name*>=*number_of_connections_per_second* sets the global rate that is enforced across all listening endpoints that are rate-limited. When this parameter is specified, it overrides any endpoint-level numeric rate values that might be specified.
  - RATE_LIMIT indicates that a particular listening endpoint is rate limited. The parameter is specified in the ADDRESS section of the listener endpoint configuration. When the RATE_LIMIT parameter is set to a value greater than 0, the rate limit is enforced at that endpoint level.

  Example: Throttle new connections to prevent logon storms.

```
APP_LSNR=
   (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521)(RATE_LIMIT=10))
    (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1522)(RATE_LIMIT=20))
    (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1523))
    )
```

  In the preceding example, the connection rates are enforced at the endpoint level. A maximum of 10 connections are processed through port 1521 every second. The connections through port 1522 are limited to 20 every second. Connections through port 1523 are not limited. When the rate is exceeded, a TNS-01158: Internal connection limit reached is logged.

  Refer to Net Services Reference guide.

Other Settings

Exadata configuration and parameter best practices have been documented in MOS notes 1274318.1 and 1347995.1.

Start Monitoring and Checking the System

Enterprise Manager, statistics gathering using Automatic Workload Repository (AWR), or Active Data Guard Statspack for Active Data Guard environments are required for monitoring and managing your database and system resources. Refer to the following documents for more information:

- Exachk in MOS note 1070954.1

- Exadata Monitoring Best Practices with Enterprise Manager 12*c* in Oracle Enterprise Manager 12c: Oracle Exadata Discovery Cookbook and MOS note 1110675.1
- Exadata Automatic Service Request at http://www.oracle.com/asr
- Active Data Guard statspack in MOS note 454848.1
- "Cluster Health Monitor" in Oracle Clusterware Administration and Deployment Guide

Resource Management

Oracle Database Resource Manager (the Resource Manager) is an infrastructure that provides granular control of system resources between workloads and databases. You can use Resource Manager to manage CPU, disk I/O, and parallel execution. Resource Manager helps you in two distinct scenarios:

- For intra-database consolidation, managing resource utilization and contention between applications.
- For inter-database consolidation, managing resource utilization and contention between database instances.

Using Resource Manager for Intra-Database (Schema) Consolidation

You can use Resource Manager in an intra-database (schema-level) consolidation to control how applications share CPU, I/O, and parallel servers within a single databases. Resources are allocated to user sessions according to a **resource plan** specified by the database administrator. The plan specifies how the resources are to be distributed among **resource consumer groups**, which are user sessions grouped by resource requirements. For schema consolidation, you would typically create a consumer group for each application. A **resource plan directive** associates a resource consumer group with a resource plan and specifies how CPU, I/O, and parallel server resources are to be allocated to the consumer group.

CPU resource management has the additional benefit that critical background processes (for example, LGWR, PMON, and LMS) are not starved. This can result in improved performance for OLTP workloads and can reduce the risk of instance eviction for Oracle RAC databases.

To manage resource usage for each application, you must configure and enable a resource plan. For more information on how to do this, see "Managing Resources with Oracle Database Resource Manager" in the Oracle Database Administrator's Guide, the MAA white paper "Using Oracle Database Resource Manager," and MOS note 1339769.1 for example setup scripts.

To manage the disk I/Os for each application, you must enable **I/O Resource Management (IORM).** The resource plan used to manage CPU is also used to manage disk I/Os. IORM manages the Exadata cell I/O resources on a per-cell basis. Whenever the I/O requests threaten to overload a cell's disks, IORM schedules I/O requests according to the configured resource plan. IORM schedules I/Os by immediately issuing some I/O requests and queuing others. IORM selects I/Os to issue based on the allocations in the resource plan; databases and consumer groups with higher allocations are scheduled more frequently than those with lower allocations. When the cell is operating below capacity, IORM does not queue I/O requests.

When IORM is enabled, it automatically manages background I/Os. Critical background I/Os such as log file syncs and control file reads and writes are prioritized. Non-critical background I/Os are deprioritized.

Using Resource Manager for Database Consolidation

Resource Manager can help database consolidation in two ways. First, Resource Manager can help control CPU usage and manage CPU contention through instance caging. Second, Resource Manager can control disk I/O usage and contention through IORM's inter-database resource plans.

Inter-database IORM plans enable you to manage multiple databases sharing Exadata cells. Inter-database IORM plans are configured using the Cell Control Command-Line Interface (CellCLI) utility. With inter-database IORM plans, you can specify the following for each database:

- Disk I/O resource allocation: Databases with higher resource allocations are able to issue disk I/Os more rapidly. Resource allocation for workloads within a database is specified through the database resource plan. If no database resource plan is enabled, then all user I/O requests from the database are treated equally. Background I/Os, however, are still prioritized automatically.
- Disk utilization limit: In addition to specifying resource allocations, you can also specify a maximum disk utilization limit for each database. For example, if production databases OLTP and OLTP2 are sharing the Exadata storage, then you can set a maximum utilization limit for both databases. By setting a limit on the database's disk utilization, you can obtain more predictable, consistent performance, which is often important for hosted environments.

  If a maximum utilization limit is specified, then excess capacity cannot be used by the database. It is possible that the disks are running below full capacity when maximum utilization limits are specified.
- Flash cache usage: IORM supports flash cache management starting with Exadata software version 11.2.2.3.0 or higher. The ALTER IORMPLAN flash cache attribute can be set to "off" to prevent a database from using the flash cache. This allows flash cache to be reserved for mission-critical databases. You should only disable flash cache usage if you are sure it is affecting the flash cache hit rate of critical databases. Disabling flash cache has the negative side effect of increasing disk I/O load.

  > **NOTE:** Starting with Exadata Storage Server Software 11.2.3.1 and higher, I/O Resource Management (IORM) supports share-based plans which can support up to 1024 databases, and up to 1024 directives for interdatabase plans. The share-based plans allocate resources based on shares instead of percentages. A share is a relative distribution of the I/O resources. In addition, the new `default` directive specifies the default value for all databases that are not explicitly named in the database plan. Prior releases only supported 31 databases.

Category resource management is an advanced feature. It allows you to allocate resources primarily by the category of the work being done. For example, suppose all databases have three categories of workloads: OLTP, reports, and maintenance. To allocate the I/O resources based on these workload categories, you would use category resource management. See "About Category Resource Management" in the Oracle Exadata Storage Server Software User's Guide.

This section highlights the key consolidation practices for Exadata. For complete Exadata resource management **prerequisites, best practices, and patches**, refer to the Master Note for Oracle Database Resource Manager (MOS note 1339769.1).

Guidelines for Managing CPU and I/O Resources within a Hardware Pool

- Enable instance caging. Set CPU_COUNT for each instance according to your sizing analysis.
- For critical Hardware Pools, use the partitioning approach: sum(CPU_COUNT) <= 100% of the number of CPUs.
- For non-critical and test Hardware Pools, use the over-subscribe approach: sum(CPU_COUNT) <= (up to 3) * CPUs, depending on acceptable response time or throughput requirements.
- Configure and enable inter-database IORM plans.
- To configure Resource Manager for workloads within a database, refer to the "Database Resource Management and Task Scheduling" section in the Oracle Database Administrator's Guide.

Monitoring and Tuning

The key recommendations are:

- Check to see if each application's performance metrics are met. These metrics are ultimately the most telling.
- For instance caging, monitor the CPU utilization of each instance's using the following query. Also refer to MOS note 1338988.1.

```
SQL> select to_char(begin_time, 'HH24:MI') time,
max(running_sessions_limit) cpu_count, sum(avg_running_sessions)
avg_running_sessions, sum(avg_waiting_sessions)
avg_waiting_sessions  from v$rsrcmgrmetric_history group by
begin_time order by begin_time;
```

  If **avg_running_sessions** is consistently less than the value of CPU_COUNT, then you can reduce the value of CPU_COUNT without impacting performance and increase the value of CPU_COUNT for another instance that needs the extra resources.

  If **avg_waiting_sessions** is consistently greater than the value of CPU_COUNT and response time and throughput are not acceptable, you can improve performance by increasing CPU_COUNT (if you have the available CPU resources). Otherwise, consider migrating this application or database to a different Hardware Pool.

- When using the over-subscribe instance caging approach, monitor the system. If the system CPU run queue is greater than 9, consider lowering the sum of CPU_COUNT across all instances.
- Monitor I/O metrics, using the script in MOS note 1337265.1.

This script provides per-minute metrics for disk and flash usage by database and consumer group. It also provides per-minute metrics for disk I/O latencies. If IORM is enabled, it also provides per-minute metrics for I/O throttling.

> **NOTE:** The maximum I/Os per second (IOPS) per disk is 300 IOPS for high performance disks and 150 IOPS for high capacity disks. Maximum throughput for Exadata cell is about 1.5-2.0 GB/sec for high performance disks and 1 GB/sec for high capacity disks. Typical symptoms of I/O throughput issues are high I/O utilization (%util from iostat), high disk queue size (avgqu-sz from iostat), poor response time and I/O latency from iostat and throughput.

Tuning for Low-Latency Workloads

For OLTP workloads, low latency is paramount. Check for the following:

1. High buffer cache hit ratio (> 98%). Tune buffer cache accordingly.
2. High flash cache hit ratio (> 90%). Keep hot tables in flash cache if needed.
3. Low disk utilization (< 60%). At high disk utilization rates, you will see good throughput, but poor latency. Therefore, if the disk utilization is high, use the I/O metrics to determine which databases and applications are generating the load.
4. Low latency for database wait event *log file sync* (for example, < 10 ms), wait event *db file sequential read* (for example, < 15 ms) and wait event *cell single block physical read* (for example, < 15 ms).
5. Disk utilization per database. If the disk utilization for one database increases dramatically, it can affect the other databases. Monitor using the I/O metrics or the top I/O transactions in AWR and ASH reports.

You can use IORM to improve OLTP latencies as follows:

1. Increase resource allocations to databases and consumer groups with OLTP workloads.
2. If the latency is still not low enough, set IORM's objective to "low latency".
3. If the latency is still not low enough, your performance objectives may be too aggressive to permit OLTP and DSS workloads to share storage. There is an intrinsic trade-off between throughput and latency. For extremely low disk latencies, the disk utilization may need to be so low that it doesn't make sense to share the storage between OLTP and DSS workloads. In this case, consider creating separate storage or Hardware Pools for OLTP and DSS.

Scenario 1: Consolidation with OLTP DBs

This scenario illustrates how to distribute I/O resources among three critical gold-levelOLTP databases and how to configure instance caging on an Exadata Database Machine X3-2. In general, we recommend using simple, single-level resource plans. In addition to specifying allocations, the resource plan below uses an I/O limit directive of 50% for COMMERCE and SALES database and 30% for the PAYROLL database so that performance does not vary widely, depending on the load from other databases. To allow each database to use 100% of the disk resources when the other databases are idle, remove the limit directive.

TABLE 4. OLTP CONSOLIDATION

DATABASE NAME	DESCRIPTION	LEVEL	ALLOCATION	LIMIT	CPU_COUNT
OLTP-A	Gold: COMMERCE	1	35	50	8
OLTP-B	Gold: SALES	1	35	50	8
OLTP-C	Gold: PAYROLL	1	20	30	6
Other	(Any other gold databases)	1	10	30	4

```
ALTER IORMPLAN  -
  dbplan=((name=oltp-a, level=1, allocation=35, limit=50), -
    (name=oltp-b, level=1, allocation=35, limit=50), -
    (name=oltp-c, level=1, allocation=20, limit=30), -
    (name=other,   level=1, allocation=10, limit=30))
```

To enable Instance Caging, set the cpu_count of each instance and then enable CPU Resource Manager. For example, you can enable instance caging for OLTP_A instance by

```
SQL> ALTER SYSTEM SET CPU_COUNT = 8 SCOPE=SPFILE;
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'DEFAULT_PLAN'
SID='*' SCOPE='BOTH';
```

Scenario 2: Consolidation with Mixed Workloads

This scenario illustrates prioritizing OLTP workloads over other OLTP and DW workloads to keep the I/O latency low.

TABLE 5. MIXED CONSOLIDATION WITH OLTP PRIORITY

DATABASE NAME	DESCRIPTION	LEVEL	ALLOCATION
OLTP-A	Gold- database	1	50
OLTP-B	Gold database	1	20
OLTP-X	Gold database	1	10
DW-Y	Gold data warehouse	1	10
Other	(Any other databases)	1	10

Since the OLTP databases have larger resource allocations than the data warehouse databases, IORM will automatically optimize for low disk latency. To explicitly control this objective, use ALTER IORMPLAN to set the objective to "low latency". IORM can reduce the disk latency to around 15 ms in a mixed workload environment.

Scenario 3: Consolidation with Data Warehouses

This scenario illustrates how to distribute I/O resources among three data warehouse databases. In general, we recommend using simple, single-level resource plans. In addition to specifying allocations, the resource plan below uses an I/O limit directive of 40% for DW-X and DW-Y databases and smaller allocation for the rest so that performance does not vary widely, depending on the load from other databases. Limits are useful in a hosting environment that requires a "pay for performance" model.

TABLE 6. DATAWAREHOUSE CONSOLIDATION

DATABASE NAME	DESCRIPTION	LEVEL	ALLOCATION	LIMIT
DW-X	Silver- data warehouse	1	40	30
DW-Y	Silver- data warehouse	1	40	30
DW-Z	Silver data warehouse	1	10	20
Other	(Any other Silver level database)	1	10	20

Scenario 4: Consolidation mixed categories

This scenario illustrates how to distribute I/O resources among three mission critical gold-level Active Data Guard standby databases co-locating with some bronze-level test databases. In general, we recommend using simple, single-level resource plans. In addition to specifying allocations, the resource plan below uses an I/O limit directive of 30% for standby databases and 20% for the other databases so that performance does not vary widely, depending on the load from other databases. If there's a Data Guard role transition and one of the standby databases become primary database, the limits will need to be increased for the new primary database and test databases will actually be shutdown or limits reduced even further. Due to these additional operational considerations including software updates, consolidation with mixed categories is not recommended approach and should be used as an exception.

TABLE 6. DATAWAREHOUSE CONSOLIDATION

DATABASE NAME	DESCRIPTION	LEVEL	ALLOCATION	LIMIT
ADG1	Gold: Commerce Active DG1	1	20	30

ADG2	Gold: Sales Active DG2	1	20	30
ADG3	Gold: Payroll Active DG3	1	20	30
TEST1	Bronze Test	1	10	20
TEST2	Bronze Dev	1	20	20
TEST3	Bronze Q/A	1	10	20

Resource Management Best Practices

For the latest resource management best practices including scripts and recommended software versions specific for effective resource management, refer to MOS note 1339769.1

Maintenance and Management Considerations

Oracle Software and File System Space

Maintain only a few active Oracle Home versions, because a proliferation of different database versions causes additional maintenance complexity. Also remember that the Oracle Grid Infrastructure software version must be the same as, or later than, the highest Oracle Database software version. Use a shared Oracle Home whenever possible and maintain a fixed number of terminal releases.

To ensure that the root partition space is maximized to accommodate all the Oracle Homes, Oracle Grid Infrastructure home, and the various log directories:

- Ensure you have reclaimed disk space after selecting your preferred operating system. Refer to the section "Reclaiming Disk Space After Selecting the Operating System" in the Oracle Exadata Database Machine Owner's Guide.
- Evaluate if you can add more space to your non-root partition (/u01). Refer to Oracle Exadata Database Machine Owner's *Guide's Resize LVM Partitions* section.
- Refer to *How to Expand Exadata Compute Node File Systems* (MOS note 1357457.1) for step by step instructions
- Manage audit file directory growth with cron (MOS note 1298957.1).
- Alternatively, you can copy and purge log directories and files to DBFS or external NFS directories.

Security and Management Roles

When multiple databases are consolidated within a Hardware Pool, it may be necessary to isolate certain database components or functions so the administration and privileges are clearly segregated. Refer to the Oracle Exadata Database Machine Consolidation: Segregating Databases and Roles paper

for a detailed description of how to use operating system and database-scoped security to prevent unauthorized access for administrators or end users.

NOTE: The maximum number of ASM Disk Groups is 63 per storage grid in Orade 11*g*.

The figure below depicts how access can be restricted by leveraging some of these security practices which is different from the default recommended single DATA and RECO disk group configuration.

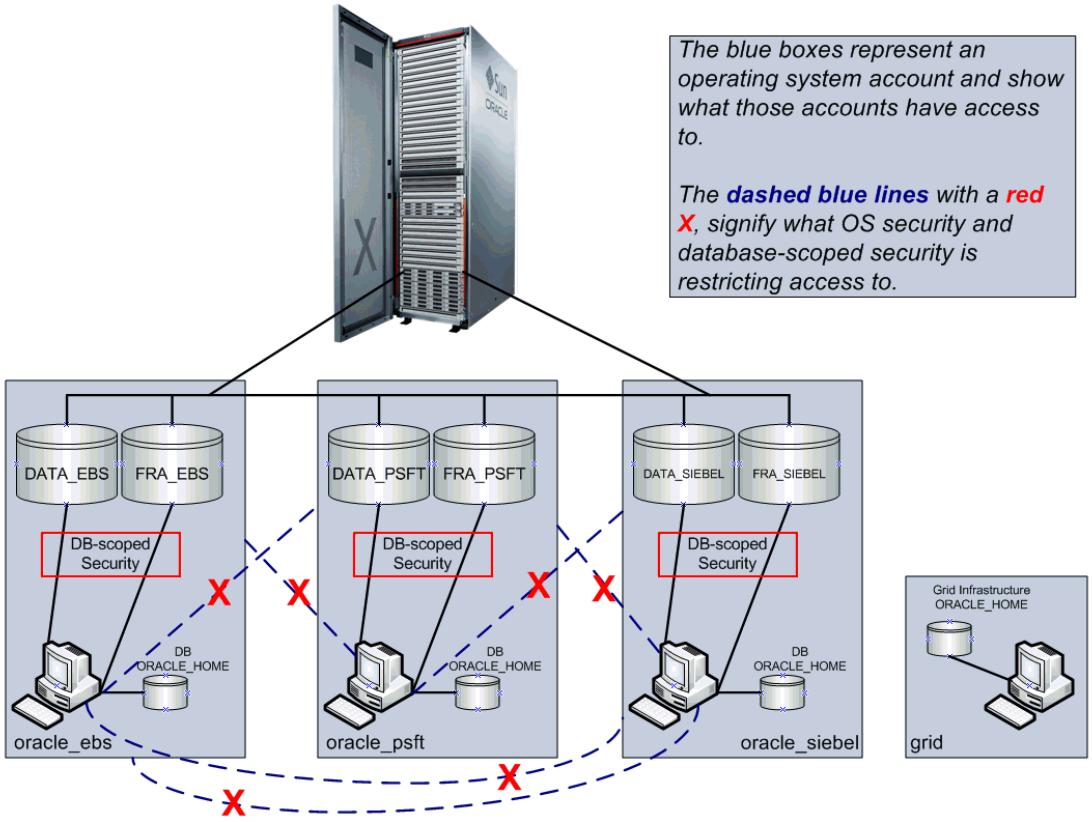

Figure 2. Security practices for access restriction

Exadata MAA for Consolidation

Orade Maximum Availability Architecture (MAA) is especially relevant to consolidated environments where the impact of planned and unplanned downtime is much greater than when applications ran on disparate systems. Refer to the MAA best practices for Oracle Exadata white paper for general best practices for Exadata. The following sections highlight the key MAA recommendations in the context of a consolidated environment.

A consolidated MAA environment for business critical applications is comprised of:

- Production Hardware Pool(s)
- Standby pool(s). Data Guard is used to maintain a standby pool that is an exact synchronized replica of the production Hardware Pool. Depending upon the criticality of the production Hardware Pool,

it may have both a local standby pool using synchronous zero data loss protection, and a remote standby pool using asynchronous transport to provide geographic protection from wide-spread outages. Standby pools are ideally configured with similar capacity as the production Hardware Pool to provide equivalent service levels if a failover is required. A number of strategies can be used to make productive use of standby pools while in standby role. See the MAA paper: Disaster Recovery for Oracle Exadata Database Machine for more details. A standby pool can also be used to validate patches and software changes with standby-first patching (refer to MOS note 1265700.1).

- Development/Test Hardware Pools which are also very critical because all changes should be validated in your development and test systems first to maintain maximum stability and availability of your production and standby hardware pools.

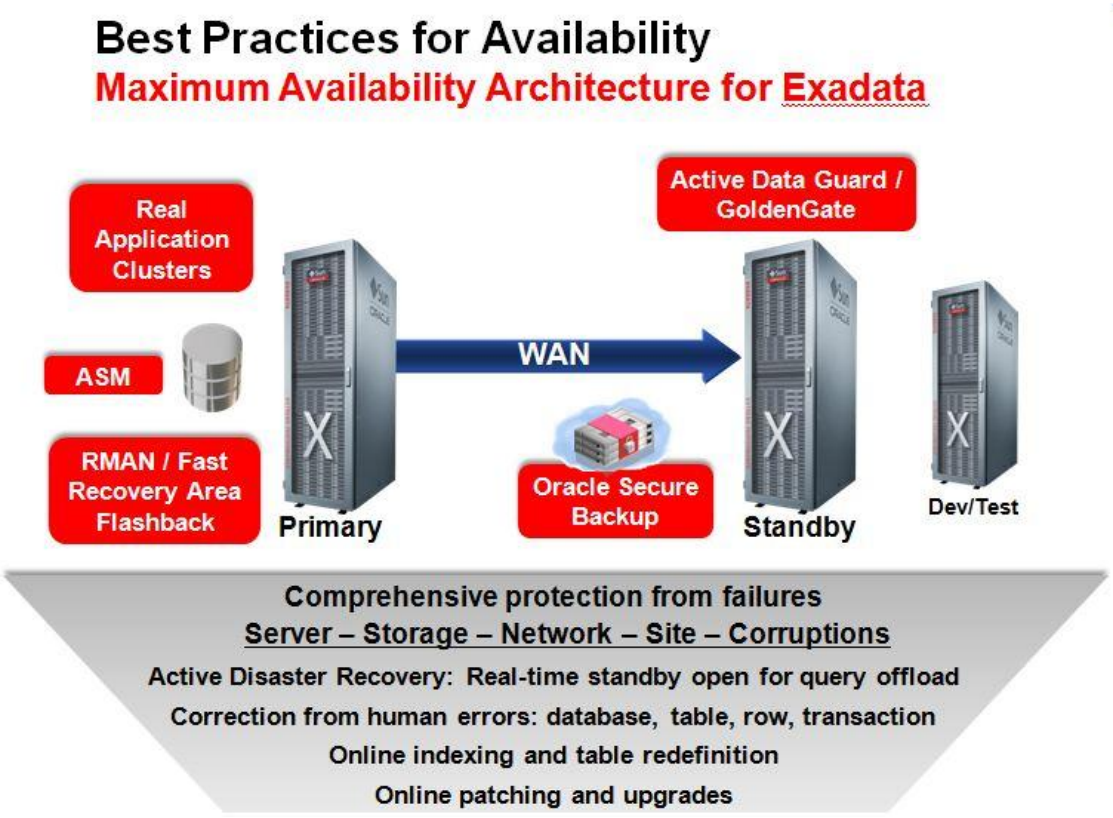

Figure 3. Maximum Availability Architecture for Exadata: Protecting Consolidated Database Environments

The Maximum Availability Architecture (Figure 3) provides the following HA capabilities for a consolidated environment:

- Oracle Real Application Clusters (RAC)—to tolerate node and instance failures.
- Oracle ASM and Oracle Exadata—to tolerate various storage failures.

- Active Data Guard—to automatically repair physical block corruptions, protect a standby pool from data corruptions or outages that may impact the primary Hardware Pool, and enable fast failover to maintain high availability if an individual database or an entire Hardware Pool fails for any reason.
- Best Practices for Corruption Detection, Prevention, and Automatic Repair - in a Data Guard Configuration (MOS 1302539.1)
- Oracle Online Patching, Oracle Grid Infrastructure and Oracle Exadata cell rolling patching, Oracle RAC rolling maintenance, Oracle Data Guard Standby-First patching, Oracle Data Guard database "transient logical" rolling upgrades, and Oracle GoldenGate heterogeneous migrations and zero downtime maintenance —to minimize or eliminate downtime for planned maintenance.
- Flashback technologies for fast point-in-time recovery to recover from logical corruptions.

HA Best Practices for the above capabilities are documented in the technical white paper, MAA Best Practices for Oracle Exadata Database Machine.

The following Exadata MAA operational practices are also important for achieving the HA goals for consolidated database environment:

1. Document HA Service Level Agreements.
2. Validate repair strategies and rolling upgrade solutions for different outages.
3. Follow the testing and patching practices described in MOS note 1262380.1, including standby-first patching (MOS note 1265700.1).
4. Execute the latest version of the exachk utility (MOS note 1070954.1) before and after planned maintenance activities and at least once every month. Address software gaps and recommendations, and address hardware, software, and MAA alerts and recommendations.
5. Execute Data Guard role transitions periodically, such as every 6 months.
6. Configure Exadata monitoring, Oracle Configuration Manager (OCM) and Automatic Service Request best practices (MOS notes 1110675.1 and 1319476.1).
7. Refer to the latest MAA best practices and papers at www.oracle.com/goto/maa. Also refer to the generic Exadata MAA practices in the MAA Best Practices for Oracle Exadata Database Machine technical white paper.

The following sections highlight additional Exadata MAA considerations for database consolidation.

Patching and Upgrading

When initially configuring the software on your Exadata Hardware Pool execute exachk and refer to software guidance. Except for rolling upgrade scenarios, Exadata cells should all have the same version. All database nodes should use the same Oracle Grid Infrastructure software version and at the most a few shared Oracle Database homes.

When updating Oracle Database software versions it is a good practice to follow out-of-place patching and upgrade procedures. This is particularly applicable in a database consolidation environment

because it allows the targeted database to be migrated to the new version without impacting the rest of the databases and also enables easy fallback to the previous version in case of unexpected performance issues. Leverage tools such as OPlan and OPatch for database servers and patchmgr for Exadata cells.

Backup and Recovery

Your backup, restore, and recovery strategy does not change with consolidation. Start with a good understanding of how to leverage your backups for different recovery scenarios and your expected recovery time objective (RTO), recovery point objective (RPO) and backup windows. Your Hardware Pool should contain applications and databases with similar HA requirements. For a critical Hardware Pool using the MAA Hardware Pool architecture, backups might be used for bare metal restore of the database node, for double disasters, and for some corruption or logical failure cases. Next, understand what is achievable with disk-based backups on Exadata (internal or with Exadata storage expansion rack) or to tape. Refer to the Backup and Recovery Performance and Best Practices for Exadata Database Machine - Oracle Database 11.2.0.2 and later , Backup and Recovery Performance and Best Practices using Sun ZFS Storage Appliance with Oracle Exadata Database Machine and Oracle Exadata Database Machine - Backup & Recovery Sizing: Tape Backups technical papers for potential backup and restore rates and configuration practices.

The next step is to understand that backing up or restoring databases is all about available system resources, especially the following:

- Exadata Disk throughput for backups and restore.
- Network bandwidth: InfiniBand for Exadata based backups or the external network for disk-based backups external to Exadata or to tape
- Third Party I/O throughput: Tape throughput for tape-based backups or storage throughput for non-Exadata storage targets

You can then choose to backup or restore one or more databases at the same time. The bottleneck will be from one of the above culprits.

Example 1: Exadata Database Machine Full Rack Backups to Disk

In this example, there are five databases on an Exadata Database Machine X2-2 full rack using High Performance SAS disks. The sum of the used database space for all databases is about 50 TB.

According to our MAA studies, we can expect backup rates of up to 20+ TB/hour when the RMAN operation leverages all database nodes and allocates 2 to 8 RMAN channels per database node. You may choose to throttle that backup rate to ensure that application performance is not impacted during the backup operations. Assuming the backup rate is 10 TB/hour using only 1 or 2 RMAN channels per node, we can back up all of the databases in five hours. We can choose to back up one database at a time or multiple databases simultaneously. The key is to distribute the work across all of the nodes so the database nodes are evenly impacted. For example, we can execute two RMAN operations for two different databases where a single backup service leverages database nodes 1-4 and a second backup service leverages database nodes 5-8. The combined backup rate will still be 10 TB/hour. Using our recommended incremental backup approach, the backup time can be less than one hour.

Example 2: Exadata Database Machine Full Rack Backups to Tape

In this example, there are 10 databases on Exadata Database Machine X2-8 using high performance SAS disks. This database uses a tape-based backup solution so the DATA area has more usable space. The sum of the used database space for all databases is about 100 TB. There are two media servers and 16 tape drives (for example, 200 MB/sec). The number of tape drives and their throughputs are the bottleneck. With this tape-based infrastructure, the backup/restore throughput is close to 11 TB/hour. Distribute the backup service across two database nodes allocating 8 RMAN channels per node. Full database backups of the entire 100 TBs can complete in less than 10 hours, and daily cumulative incremental backups can complete in less than two hours, depending on the actual size of the backup set.

Furthermore, you can use resource management to prioritize different backups and application workloads.

Data Guard

Refer to Oracle Data Guard: Disaster Recovery Best Practices for Exadata Database Machine paper for Data Guard on Exadata practices. The key consolidation considerations for your standby pool are:

- Ensure that the primary and standby pools are **not** on the same InfiniBand fabric.
- If zero data loss is required, you may need a local standby pool, and a performance evaluation is required to assess the performance impact with Data Guard sync transport.
- Sufficient network bandwidth is required to handle the sum of the redo traffic between all primary and standby databases. If the primary and standby pools are over a WAN with insufficient network bandwidth, consider using redo compression. Redo compression can consume 5-10% CPU overhead on both the primary and standby pools.
- Conduct role transition testing, including application and Data Guard switchover and failover operations, for a single database, multiple databases, and-in the worse case-the entire Hardware Pool to ensure that in all circumstances you can meet your RTO and RPO requirements.
- Verify that sufficient system resources are available on the standby pool to support all target production workload in the case of complete production Hardware Pool failure.
- Due to the complexity of managing so many databases within the primary and standby pools, Data Guard broker and Enterprise Manager are highly recommended. If automatic failover is required to reduce RTO and eliminate the time to detect and react to some failures, consider Data Guard Fast-Start failover and MAA client failover best practices (Client Failover Best Practices for Data Guard 11*g* Release 2).

Recovery with Schema Consolidation

For schema-level consolidation (running multiple application schemas in one database) the main consideration is whether or not the application schemas are suitable to co-exist within the same database. If schema-level consolidation is feasible, then the result is a single database that should be

configured and managed in-line with all the standard recommendations for Exadata MAA. In addition, the following recommendations and considerations apply:

- Each application schema should be contained in a separate tablespace or set of tablespaces. This enables easy and efficient space management. It also facilitates the use of tablespace point-in-time recovery (TSPITR) or flashback table operations to perform recovery of a specific application schema without impacting the other applications.
- Each application schema should use a separate database service.
- Tune your backup and recovery practices so that individual application schemas or tablespaces can be maintained with minimal disruption to other applications. This includes understanding the overheads and requirements of different approaches, and ensuring that the required system resources are available to support the desired objectives. The different repair options include TSPITR, flashback table or flashback transaction, exportand import, or Data Pump operations.
- If TSPITR is considered, then also consider whether image copies can be used to enable faster recovery. In this case, RMAN does not need to restore the datafiles from a backup.
- Refer to "MAA Schema Recovery Options in an Exadata Consolidate Environment practices" (MOS note 1386048.1)

Summary

Exadata is the optimal database consolidation platform, engineered from the ground up to provide extreme performance and scalable capacity for both OLTP and Data Warehouse databases. Oracle Database, storage and network grid architectures combined with Oracle resource management provide a simpler and more flexible approach to database consolidation than other virtualization strategies (for example, hardware or operating system virtualization). Follow the best practices documented in this paper and in associated references to maximize stability and availability when supporting multiple applications and databases that use shared resources in a database consolidated environment.

Oracle White Paper Title
October 2013
Author: Lawrence To, Sue K. Lee,

Contributing Authors: Exadata MAA Team, Juan Loaiza, Joe Meeks, Frank Kobylanski, Rene Kundersma, Mahesh Subramaniam, Doug Utzig, Mike Nowak, Andrew Babb, Hector Pujol, Virginia Beecher, DW and OLTP Performance teams

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

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

Hardware and Software, Engineered to Work Together