Red Hat Reference Architecture Series

Oracle 10g Server on Red Hat® Enterprise Linux® 5

Deployment Recommendations

Version 1.2

November 2008

redhat.

Oracle 10g Server on Red Hat® Enterprise Linux® 5
Deployment Recommendations

1801 Varsity Drive
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Revision History

Version	*Date*	*Summary of Changes*	*Author*
1.0	22-SEP-2008	Original Report	A. Brezhnev
1.1	2-OCT-2008	Incorporated review input	A. Brezhnev
1.2	26-NOV-2008	Edited and formatted for Reference Architecture release	Solutions Architecture group

Table of Contents

Introduction

This document provides recommendations for running Oracle 10g R2 standalone and RAC databases on Red Hat Enterprise Linux (RHEL).

The recommendations presented in this document are targeted to the following hardware platforms:

- 64-bit physical system (x86_64) with SAN attached storage for Oracle
- VMware virtual machine (x86_64) with virtualized disks for Oracle

We will present generic tuning recommendations for systems with 1, 4, 8, 16, 32 and 64GB of RAM. These generic recommendations may require adjustments depending on the Oracle database characteristics and workload.

We assume that physical systems have at least two NICs and two HBAs to provide redundant network and SAN connections.

We present an example of Linux native multipath configuration for EMC CLARiiON and SYMMETRIX storage arrays.

If the database servers need to be installed on multiple target hosts, it is recommended to package the Oracle software via RPM using *RPM Packaging Guide for Oracle 10g on Red Hat Enterprise Linux 5*. Then the Oracle installation can be integrated with software management tools like *Red Hat Network Satellite* or *yum* and can be unattended.

64-bit Architecture

This is the architecture that should be used whenever possible. The 32-bit platform is also supported but for the large databases, the x86_64 architecture is strongly recommended. For supported system configurations and limits for Red Hat Enterprise Linux releases, see http://www.redhat.com/rhel/details/limits/.

If you can go with a x86-64 platform ensure that all drivers you need are supported on x86-64 (e.g. proprietary multipath drivers such as EMC PowerPath, etc.) Furthermore, ensure that all the required applications are supported on x86-64.

Packages Required for Oracle 10gR2 (10.2) Installation

Install following Red Hat Enterprise Linux 5 RPMs to satisfy dependencies of Oracle software:

```
# yum install binutils compat-db compat-gcc-34 compat-gcc-34-c++ \
              compat-libstdc++-33 elfutils-libelf-devel \
              gcc gcc-c++ gdb gdbm glibc glibc-common glibc-devel \
              ksh libXp libXtst libaio libaio-devel libgcc libgnome \
              libstdc++ libstdc++-devel make setarch sysstat unixODBC unixODBC-devel \
              util-linux xorg-x11-xinit compat-libstdc++-296
```

On 64-bit platform, the following 32-bit packages must also be installed:

```
# yum install glibc-devel.i386 libaio.i386 glibc.i686 libgcc.i386 \
              compat-libstdc++-33.i386 openssl.i686 libXp.i386 libXtst.i386
```

It is recommended to use kickstart for automated installation. Using kickstart allows the user to create a properly sized swap area and file system layout according to the Oracle best practices.

In a typical situation when Oracle database server needs to be installed on a system with a general purpose, corporate Linux build it is recommended to create an RPM with dependencies for the packages listed above and post-install script for system tuning. A sample spec-file of such an RPM is can be seen in Appendix A.

Kernel Upgrades

Make sure to install the latest kernel where all proprietary drivers, if applicable, are certified/supported.

Note that proprietary drivers are often installed under `/lib/modules/<kernel-version>/kernel/drivers/addon`.

Therefore, when you upgrade the kernel you must ensure that all proprietary modules can be found in the correct directory so that the kernel can load them.

Kernel Boot Parameters

General

The Linux kernel accepts boot parameters when the kernel is starting. Very often it is used to provide information to the kernel about hardware parameters where the kernel would have issues/problems or to supersede default values.

For a list of kernel parameters, see `/usr/share/doc/kernel-doc-*/Documentation/kernel-parameters.txt`. This file does not exist if the `kernel-doc` RPM is not installed.

I/O Scheduler

Starting with the 2.6 kernel (Red Hat Enterprise Linux 4 and later) the I/O scheduler can be modified at boot time to control the manner in which the kernel commits reads and writes to disk. Red Hat Enterprise Linux 5 allows users to change I/O schedulers dynamically for each block device (e.g. echo sched_name > /sys/block/<*sdx*>/queue/scheduler).

Red Hat Enterprise Linux 5 kernel supports four I/O schedulers:

- **cfq** (Completely Fair Queuing)
- **deadline**
- **noop**
- **anticipatory**

The CFQ scheduler waill attempt to distribute I/O bandwidth fairly over all processes. It is a default algorithm in Red Hat Enterprise Linux 5 which is suitable for a wide variety of applications and provides a good compromise between throughput and latency. The CFQ scheduler is recommended for most workloads and systems with shared storage.

The deadline scheduler reorders I/O to optimized disk heads movement and caps maximum latency per request to prevent resource starvation for I/O intensive processes. The deadline scheduler is recommended for single instance databases with dedicated storage.

The noop scheduler maintains a simple FIFO queue of I/O requests. It performs basic sorting and merging using minimal CPU resources. The scheduler is recommended for CPU-limited environments and systems with smart shared storage.

The anticipatory scheduler is not intended for servers and as such not recommended.

In virtualized environments, it is often detrimental to schedule I/O at both the host and guest layers. If multiple guests access storage on a file system or block devices managed by the host operating system, the host may be able to schedule I/O more efficiently because it alone is aware of requests from all guests and knows the physical layout of storage, which may not map linearly to the guests' virtual storage. Red Hat Enterprise Linux 4 and Red Hat Enterprise Linux 5 guests can use the noop I/O scheduler to allow the host to optimize I/O requests.

Guests using storage accessed either by iSCSI or by physical device pass-through should *not* use the noop scheduler since these methods do not allow the host to optimize I/O requests to the underlying physical device.

To configure a non-default scheduler, the *elevator* parameter must be passed to the kernelbeing used. For example, to switch to the deadline scheduler edit the `/etc/grub.conf` file and add the following parameter to the kernel line:

```
title Red Hat Enterprise Linux Server (2.6.18-92.1.13.el5)
        root (hd0,0)
        kernel /vmlinuz-2.6.18-92.1.13.el5 ro root=/dev/sda2 elevator=deadline
        initrd /initrd-2.6.18-92.1.13.el5.img
```

This entry tells the kernel to use the deadline scheduler as a new system-wide default after reboot.

In Red Hat Enterprise Linux 5, the scheduler can be set at run time and for each block device. For example, set

the noop scheduler for device /dev/sdd and verify the result:

```
# echo noop > /sys/block/sdd/queue/scheduler
# cat /sys/block/sdd/queue/scheduler
[noop] anticipatory deadline cfq
```

RHEL5 Running as a VMware Guest

By default, the timer interrupt (tick) rate in Red Hat Enterprise Linux 4 and 5 is 1000 Hz. In some situations, this rate is too high, with frequent interrupts potentially causing a performance impact or unexpected behavior. This impact would be most notable when running Red Hat Enterprise Linux as a virtual guest. For example, with the 1000 Hz tick rate, time drift might be observed in a virtual environment.

To compensate for the fact that different workloads and different environments work better with different tick rates, a new kernel parameter was added to Red Hat Enterprise Linux 5.1 and 4.7. This kernel parameter, "divider", is a boot-time setting that allows the user to divide the tick rate by the desired amount. For example, setting the divider to 10 would cause the default tick rate to be 1000/10, or 100 Hz.

The divider parameter can be set by appending "divider=N" to the kernel boot line in `/boot/grub/grub.conf`.

The recommended kernel line settings for 64-bit Red Hat Enterprise Linux 5 running as a VMware guest are:

```
divider=10 notsc iommu=soft elevator=noop
```

and for a 32-bit Red Hat Enterprise Linux 5 Vmware guest:

```
divider=10 clocksource=acpi_pm iommu=soft elevator=noop
```

Memory Usage and Page Cache

Checking Memory Usage

To determine the size and usage of memory, you can enter the following command:

```
$ grep MemTotal /proc/meminfo
```

Alternatively, you can use the `free(1)` command to check the memory:

```
$ free
             total       used       free     shared    buffers     cached
Mem:       4040360    4012200      28160          0     176628    3571348
-/+ buffers/cache:     264224    3776136
Swap:      4200956      12184    4188772
$
```

In this example the total amount of available memory is 4040360 KB. 264224 KB are used by processes and

3776136 KB are free for other applications. Don't get confused by the first line which shows that 28160KB are free! Note that most of the memory used is for buffers and cache since Linux always tries to use RAM to the fullest extent to speed up disk operations. Using available memory for buffers (file system metadata) and cache (pages containing contents of files or block devices) helps the system to run faster because disk information is already in memory, thereby reducing I/O. If space is needed by programs or applications such as Oracle, then Linux will free the necessary buffers and cache to yield memory for the applications. Typically, if the system has been up for an extended period, a small number can be observed under the "free" column on the first line.

Tuning Page Cache

Page Cache is a disk cache which holds data of files and executable programs, e.g. pages with actual contents of files or block devices. Page Cache (disk cache) is used to reduce the number of disk reads.

The Page Cache in Red Hat Enterprise Linux 5 is dynamically adjusted. You can adjust the minimum free pages using

```
# echo 1024 > /proc/sys/vm/min_free_kbytes
```

Again to make the change permanent, add the following line to the file `/etc/sysctl.conf:`

```
# echo vm.min_free_kbytes=1024 >> /etc/sysctl.conf
```

The easiest way to tune when to start reclaiming Page Cache pages is to adjust the *swappiness* percentage. The default value of `/proc/sys/vm/swappiness` is 60 which means that applications that are not actively occupying CPU cycles can be swapped out. Higher values will provide more I/O cache and lower values will wait longer to swap out idle applications.

In general, swapping out unused pages allows thesystem to be more efficient but it is not always good for database servers. An Oracle database server typically uses a big cache of its own, in which case swapping pages may be detrimental. The optimal solution for Oracle database servers is to configure Linux Huge Pages and allow Oracle processes to use them for internal caching. Huge pages are not used as a system Page Cache so Linux and Oracle caching algorithms do not interfere.

Linux usually flushes the page cache using *pdflush* kernel threads. At any given time, 2 to 8 pdflush threads are running on the system. You can monitor active threads by monitoring `/proc/sys/vm/nr_pdflush_threads`. Whenever all existing pdflush threads are busy for at least one second, an additional pdflush thread is spawned. The new ones try to write back data to device queues that are not congested, the goal being each active device has its own thread flushing data to that device. Each time a second has passed with*out* any pdflush activity, one of the threads is removed.

Exactly what each pdflush thread does is controlled by a series of parameters in `/proc/sys/vm`:

dirty_writeback_centisecs (default 500): In hundredths of second, this is how often pdflush wakes up to write data to disk. The default wakes up pdflush every five seconds.

It may be beneficial to decrease this parameter on systems with lots of memory and active writing processes. The smaller value will make pdflush threads more aggressive in cleaning dirty pages. The kernel implements some rules to prevent write congestion. It limits the number of pages that can be flushed at once and may skip one second between pdflush activations if the threads are too busy. It does not make sense to set this parameter

too low (less than 100).

Firstly, pdflush will work on is writing pages that have been dirty for longer than it deems acceptable. This is controlled by:

dirty_expire_centisecs (default 3000): in hundredths of second, how long data can be in the page cash before it is considered expired and must be written at the next opportunity. Note that this default is very long: a full 30 seconds. That means that under normal circumstances, unless you write enough to trigger the other pflush method, Linux willnot actually commit anything you write until 30 seconds later. This may be acceptable for general desktop and computational applications but for write-heavy workloads the value of this parameter should be lowered although not to extremely low levels. Because of the way the dirty page writing mechanism works, attempting to lower this value to less than a few seconds is unlikely to work well. Constantly trying to write dirty pages out will just trigger the I/O congestion code more frequently.

Secondly, pdflush will work on is writing pages if memory is low. This is controlled by:

dirty_background_ratio (default 10): Maximum percentage of active memory that can be filled with dirty pages before pdflush begins to write them. In terms of the meminfo output, the active memory is

```
MemFree + Cached – Mapped
```

This is the primary tunable to adjust downward on systems with the large amount of memory and heavy writing applications. The usual issue with these applications on Linux is buffering too much data in the attempt to improve efficiency. This is particularly troublesome for operations that require synchronizing the file system using system calls like *fsync*. If there is a lot of data in the buffer cache when this call is made, the system can freeze for quite some time to process the sync.

Another common issue is that because so much data must be written and cached before any physical writes start, the I/O appears more in bursts than would seem optimal. Long periods are observed where no physical writes occur as the large page cache is filled, followed by writes at the highest speed the device can achieve once one of the pdflush triggers has been tripped. If the goal is to reduce the amount of data Linux keeps cached in memory so that it writes it more consistently to the disk rather than in a batch, decreasing dirty_background_ratio is most effective.

There is another parameter that affects page cache management:

dirty_ratio (default 40): Maximum percentage of total memory that can be filled with dirty pages before user processes are forced to write dirty buffers themselves during their time slice instead of being allowed to do more writes.

Note that all processes are blocked for writes when this happens, not just the one that filled the write buffers. This can cause what is perceived as an unfair behavior where a single process can ”hog” all I/O on the system. Applications that can cope with their writes being blocked altogether might benefit from substantially decreasing this value.

Summary of Recommended Virtual Memory Management Settings

There is no “one size fit all” best set of the kernel VM tuning parameters for database servers because each

database and each combination of database instances running on the same system are unique. The following settings proved useful for many Oracle database users and improved the stability and performance of Oracle servers on Red Hat Enterprise Linux 5. They should be considered as a starting point for the follow-up performance tuning.

Ensure the following settings are present in `/etc/sysctl.conf`:

```
vm.swappiness=0
vm.dirty_background_ratio=3
vm.dirty_ratio=15
vm.dirty_expire_centisecs=500
vm.dirty_writeback_centisecs=100
```

Swap Space

General

There are cases where utilizing the swap partition is beneficial. For example, long running processes often access only a subset of the page frames they obtained. This means that the swap partition can safely be used even if memory is available because system memory could be better served for disk cache to improve overall system performance. In fact, in the 2.6 kernel (Red Hat Enterprise Linux 4 and later) you can define a threshold where processes should be swapped out in favor of I/O caching. This can be tuned using the `/proc/sys/vm/swappiness` kernel parameter.

Depending on the system profile ,you may observe swap usage slowly increases with system uptime. To display swap usage, use the `free(1)` command or check the `/proc/meminfo` file. When the system uses swap space it will sometimes not decrease afterward. This saves I/O if memory is needed and pages do not have to be swapped out again when the pages are already in the swap space. However, as swap usage approaches 80% - 100% (your threshold may be lower if you use a large swap space), a closer examination of system performance is recommended, see also Checking Swap Space Size and Usage. Depending on the size of the swap space, you may want to check swap activity using `vmstat` or `sar` if swap allocation is lower than 80%. However, these numbers depend on the size of the swap space. The actual numbers of swapped pages per time frame from `vmstat` or `sar` are the important numbers. Constant swapping should be avoided .

Note, never add a permanent swap file to the system due to the performance impact of the file system layer.

Swap Size Recommendations

According to Oracle Database Installation Guide 10g Release 2 at least 1024MB of RAM is required for 10g R2. Oracle provides generic recommendations regarding the size of swap in MetaLink Note 169706.1. These recommendations may lead to creation of very large swap space on systems with large amount of memory. The very large swap may cause sever system performance degradation and can be resolved by reducing swap space. Red Hat does not recommend allocating greater than 4GB for swap on Red Hat Enterprise Linux 5.

The following table summarizes the swap size recommendations for Oracle 10g R2 on Red Hat Enterprise Linux 5:

RAM	Swap SPace
1GB - 2GB	1.5 * total RAM
2GB - 4GB	1 * total RAM
> 4GB	4GB

Checking Swap Space Size and Usage

The size and current usage of swap space can be obtained using any of the following commands:

```
grep SwapTotal /proc/meminfo
cat /proc/swaps
free
```

Swap usage may slowly increase as shown above but should stop increasing at some point. If swap usage continues to grow steadily or is already large, then one of the following options may need to be considered:

- Add RAM or reduce the size of the SGA (unless using huge pages)
- Increase the size of the swap space

Constant swapping is harmful to system performance . Check current swap activity using the following commands:

```
$ vmstat 3 100
procs                      memory      swap          io     system         cpu
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 1  0      0 972488   7148  20848    0    0   856     6  138    53  0  0 99  0
 0  1      0 962204   9388  20848    0    0   747     0 4389  8859 23 24 11 41
 0  1      0 959500  10728  20848    0    0   440   313 1496  2345  4  7  0 89
 0  1      0 956912  12216  20848    0    0   496     0 2294  4224 10 13  0 77
 1  1      0 951600  15228  20848    0    0   997   264 2241  3945  6 13  0 81
 0  1      0 947860  17188  20848    0    0   647   280 2386  3985  9  9  1 80
 0  1      0 944932  19304  20848    0    0   705     0 1501  2580  4  9  0 87
```

The fields `si` and `so` show the amount of memory paged in from disk and paged out to disk, respectively.

To check the history of swap activity, use the `sar` command. For example, to check swap activity from Sep 19:

```
# ls -al /var/log/sa | grep "Sep 19"
-rw-r--r-- 1 root root 4769520 Sep 19 23:59 /var/log/sa/sa19

# sar -W -f /var/log/sa/sa19
Linux 2.6.18-92.1.10.el5 (rac01prd)       09/19/2008

12:00:01 AM  pswpin/s pswpout/s
12:01:01 AM      0.00      0.00
12:02:01 AM      0.00      0.00
12:03:01 AM      0.00      0.00
```

The fields `pswpin` and `pswpout` show the total number of pages brought in and out per second, respectively.

If the server shows sporadic swap activity or swap activity for a short period time at certain intervals, either add more swap space or more RAM. If swap usage is already very large (do not confuse with constant swapping), then more RAM is recommended.

Large Memory Optimization (HugePages)

Physical memory is partitioned into pages which are the basic unit of memory management. When a Linux process accesses a virtual address, the CPU must translate it into a physical address. Therefore, for each Linux process the kernel maintains a page table which is used by the CPU to translate virtual addresses into physical addresses. But before the CPU can do the translation it has to perform several physical memory reads to retrieve page table information. To speed up this translation process for future references to the same virtual address, the CPU saves information for recently accessed virtual addresses in its Translation Lookaside Buffers (TLB) which is a small but very fast cache in the CPU. The use of this cache makes virtual memory access very fast. Since TLB misses are expensive, TLB hits can be improved by mapping large contiguous physical memory regions by a small number of pagesso fewer TLB entries are required to cover larger virtual address ranges. The default page size for x86 CPUs is 4KB. Modern CPUs support larger pages: 2MB, 4MB and even 1GB. In Linux these pages are called “huge pages”. They can significantly reduce the page table size and memory management overhead. The memory savings and performance gains may be very significant for Oracle database servers with a large Shared Global Area (SGA), an area of memory shared by Oracle processes. The size of the SGA has a significant impact to Oracle's performance since it contains database buffer cache and more.

For example, a database server configured with the default 4KB memory pages and a 32GB SGA is using 12MB of kernel memory per Oracle process for page tables (32GB/4KB * 12 bit per Page Table Entry (PTE) = 12MB). If this server has 1000 sessions, they consume approximately 12GB of RAM just for page tables. It is important to note that *every* connected session (`ps -ef | grep oracle | wc -l`, minus about 8 core processes) results in a full pagemap of the SGA. Needless to say, the TLB cache in this case is completely overwhelmed and the database performance suffers.

The same server configured with 2MB huge pages is using only 24KB per process (32GB/2MB * 12 bit per PTE = 24KB) or approximately 24MB for 1000 sessions. Now one PTE entry in TLB covers 2MB instead of 4KB. The TLB miss rate is significantly decreased because touching address A, A+4KB, A+8KB, ... up to A+2MB no longer cause expensive TLB misses.

Better performance is also achieved because huge pages are “pinned” in RAM, meaning they cannot be swapped out and the kernel does not spent cycles having to “think” about swapping them out.

Huge Pages in RHEL5

The Huge Pages feature in Red Hat Enterprise Linux 5 allows the dynamic allocation of large memory pages without a reboot. However, if memory gets too fragmented in Red Hat Enterprise Linux 5, allocation of physically contiguous memory pages can fail and a reboot may become necessary.

The advantages of Huge Pages are:

- Increased performance through increased TLB hits
- Pages are locked in memory rather than swapped which guarantees that shared memory, such as SGA, remains in RAM

- Contiguous pages are preallocated and cannot be used for anything else but for System V shared memory (e.g. SGA)
- Less kernel bookkeeping work for that part of virtual memory due to larger page sizes

Usage of Huge Pages in Oracle 10g

Huge Pages in Red Hat Enterprise Linux 5 need to be requested explicitly by the application by using the `SHM_HUGETLB` flag when invoking the `shmget()` system call. This ensures that shared memory segments are allocated out of the Huge Pages pool. This is done automatically in Oracle 10g.

Sizing Huge Pages

With the Huge Pages feature you specify how many physically contiguous large memory pages should be allocated and pinned in RAM for shared memory like Oracle SGA. For example, if you have three Oracle instances running on a single system with 2 GB SGA each, then at least 6 GB of large pages should be allocated. This will ensure that all three SGAs use large pages and remain in main physical memory. Furthermore, if you use ASM on the same system, then it is prudent to add at least 200MB. There may be other non-Oracle processes that allocate shared memory segments as well.

It is, however, not recommended to allocate too many Huge Pages. These preallocated pages can only be used for shared memory. This means that unused Huge Pages will not be available for other use than for shared memory allocations even if the system runs out of memory and starts swapping. Note that Huge Pages are not used for the ramfs shared memory file system.

Checking Shared Memory Before Starting Oracle Databases

It is very important to always check the shared memory segments before starting an instance. If an abandoned shared memory segment from e.g. an instance crash is not removed, it will remain allocated in the Huge Pages pool. This could mean that new allocated shared memory segments for the new instance SGA will not fit into the Huge Pages pool. For more information on removing shared memory, see Removing Shared Memory.

Configuring Huge Pages in RHEL5

In Red Hat Enterprise Linux 5, the size of the Huge Pages pool is specified by the desired number of Huge Pages. To calculate the number of Huge Pages you first need to know the Huge Page size. To obtain the size of Huge Pages, execute the following command:

```
$ grep Hugepagesize /proc/meminfo
Hugepagesize:     2048 kB
$
```

The output shows that the size of a Huge Page on this system is 2MB, meaning if a 1GB Huge Pages pool should be allocated, then 512 Huge Pages need to be allocated. The number of Huge Pages can be configured and activated by setting `nr_hugepages` in the proc file system. For example, to allocate 512 Huge Pages, execute:

```
# echo 512 > /proc/sys/vm/nr_hugepages
```

Alternatively, you can use `sysctl(8)` to change it:

```
# sysctl -w vm.nr_hugepages=512
```

To make the change permanent, append the following line to the `/etc/sysctl.conf` file as follows. This file is used during the boot process. The Huge Pages pool is usually guaranteed if requested at boot time:

```
# echo "vm.nr_hugepages=512" >> /etc/sysctl.conf
```

If you allocate a large number of Huge Pages, the execution of the above commands can take a while. To verify whether the kernel was able to allocate the requested number of Huge Pages, run:

```
$ grep HugePages_Total /proc/meminfo
HugePages_Total:   512
$
```

The output shows that 512 Huge Pages have been allocated. Since the size of Huge Pages is 2048 KB, a Huge Page pool of 1GB has been allocated and pinned in physical memory.

If `HugePages_Total` is lower than what was requested with `nr_hugepages`, then the system either does not have enough memory or there are not enough physically contiguous free pages. In the latter case, the system needs to be rebooted which should result in a better chance of getting the memory.

To get the number of free Huge Pages on the system, execute:

```
$ grep HugePages_Free /proc/meminfo
HugePages_Free:   512
```

Free system memory will automatically be decreased by the size of the Huge Pages pool allocation regardless whether the pool is being used by an application such as Oracle DB or not.

In order to allow Oracle database server to use huge pages as its shared memory, you need to do two configuration changes.

First, put group id of the Oracle instance (usually “dba”) into `/proc/sys/vm/hugetlb_shm_group`. You can change this parameter dynamically:

```
# echo `id -g oracle` > /proc/sys/vm/hugetlb_shm_group
```

or assign the corresponding value to the `vm.hugetlb_shm_group` parameter in `/etc/sysctl.conf`.

The second change is related to the ulimit parameter `memlock` for the `oracle` user. It should be increased in `/etc/security/limits.conf` if "max locked memory" is not unlimited or too small, see `ulimit -a` or `ulimit -l`. For example:

```
  oracle           soft    memlock         1048576
  oracle           hard    memlock         1048576
```

The `memlock` parameter specifies how much memory the `oracle` user can lock into its address space. Note that Huge Pages are locked in physical memory. The `memlock` setting is specified in KB and must match the memory size of the number of Huge Pages that Oracle should be able to allocate. So if the Oracle database should be able to use 512 Huge Pages, then `memlock` must be set to at least `512 * Hugepagesize`, which

on this system would be 1048576 KB (512*1024*2). If `memlock` is too small, then no single page will be allocated when the Oracle database starts. For more information on setting shell limits, see Setting Shell Limits for the Oracle User section of this document.

Login as the `oracle` user again and verify the new `memlock` setting by executing `ulimit -l` before starting the database.

After an Oracle database startup, you can verify the usage of Huge Pages by verifying that the number of free Huge Pages has decreased:

```
$ grep HugePages_Free /proc/meminfo
HugePages_Free:     0
```

To free the Huge Pages pool, execute:

```
# echo 0 > /proc/sys/vm/nr_hugepages
```

This command can take a while to finish.

Setting Shared Memory

Shared memory allows processes to access common structures and data by placing them in shared memory segments. It is the fastest form of Inter-process Communication (IPC) available since no kernel involvement occurs when data is passed between the processes. In fact, data does not need to be copied between the processes.

Oracle uses shared memory segments for the SGA.

To see all shared memory settings, execute:

```
$ ipcs -lm

------ Shared Memory Limits --------
max number of segments = 4096
max seg size (kbytes) = 67108864
max total shared memory (kbytes) = 17179869184
min seg size (bytes) = 1
```

Setting SHMMAX Parameter

This parameter defines the maximum size in bytes of a single shared memory segment that a Linux process can allocate in its virtual address space. On a 32-bit platform with 4+4GB split, such as Red Hat Enterprise Linux 4 with hugemem i686 kernel, the maximum size of shared memory segment used by Oracle 10g SGA is limited to approximately 3.42 GB since virtual address space is also needed for other things such as shared libraries. Since Red Hat Enterprise Linux 5 does not provide a 32-bit kernel with 4+4GB split,it is not recommended on the x86 platform for Oracle database server use on systems with greater than 4GB of RAM. Use either Red Hat Enterprise Linux 5 for x86_64 if the hardware can operate in 64-bit mode or Red Hat Enterprise Linux 4 for x86 with hugemem kernel on legacy 32-bit hardware.

Note that if you set SHMMAX to 4294967296 bytes (4*1024*1024*1024=4GB) on a 32-bit system, then SHMMAX will essentially be set to 0 bytes since it wraps around the 32-bit 4GB value. This means that SHMMAX should not exceed 4294967295 on a 32-bit system. In fact for Red Hat Enterprise Linux 5 x86, it should be approximately 3700000000. On x86-64 platforms, SHMMAX can be much larger than 4GB since the virtual address space is not limited by 32 bits. Oracle recommends to set SHMMAX value to the half of the size of RAM but on 32-bit platform the value should not exceed 4294967295 (4GB – 1 byte).

Since the SGA is comprised of shared memory, SHMMAX can potentially limit the size of the SGA. For that reason, SHMMAX should be slightly larger than the size of the SGA. If SHMMAX is too small, you can get error messages similar to

```
ORA-27123: unable to attach to shared memory segment
```

It is **highly recommended** that the shared memory fits within the Huge Pages pool, see Large Memory Optimization (Huge Pages) in this document.

To determine the size of system RAM:

```
# grep MemTotal /proc/meminfo
MemTotal:   16777216 kB
```

To check the maximum size of a shared memory segment:

```
# cat /proc/sys/kernel/shmmax
2147483648
```

The recommended value of SHMMAX on 64-bit platform in this case is 16777216 * 1024 / 2 = 8589934592. The value of SHMMAX can be changed in the proc file system without reboot:

```
# echo 8589934592 > /proc/sys/kernel/shmmax
```

Alternatively, you can use `sysctl(8)` to change it:

```
# sysctl -w kernel.shmmax=8589934592
```

To make a change permanent, add the following line to the file `/etc/sysctl.conf` (your setting may vary). This file is used during the boot process.

```
# echo "kernel.shmmax=8589934592" >> /etc/sysctl.conf
```

Setting SHMMNI Parameter

This parameter sets the system wide maximum number of shared memory segments. Oracle recommends SHMMNI to be at least 4096 for Oracle 10g. Since this recommendation is a minimum setting, it is best to set it to 4096 or greater.

To determine the system wide maximum number of shared memory segments, run:

```
# cat /proc/sys/kernel/shmmni
4096
```

The default shared memory limit for SHMMNI can be changed in the proc file system without reboot:

```
# echo 4096 > /proc/sys/kernel/shmmni
```

Alternatively, you can use `sysctl(8)` to change it:

```
# sysctl -w kernel.shmmni=4096
```

To make a change permanent, add the following line to the file `/etc/sysctl.conf`. This file is used during the boot process.

```
# echo "kernel.shmmni=4096" >> /etc/sysctl.conf
```

Setting SHMALL Parameter

This parameter sets the total amount of shared memory, in pages, that the system can use at one time. The default value of SHMALL can be too small and may cause Oracle database error upon startup (see Metalink Note:301830.1). Set SHMALL to the total amount of physical RAM divided by page size.

The page size can be determined using the following command:

```
$ getconf PAGE_SIZE
4096
```

For 64-bit system with 16GB of RAM and PAGE_SIZE = 4096 the recommended value of SHMALL is 16 * 1024 * 1024 * 1024 / 4096 = 4194304.

To determine the system wide maximum number of shared memory pages, run:

```
# cat /proc/sys/kernel/shmall
2097152
```

The shared memory limit for SHMALL can be changed in the proc file system without reboot:

```
# echo 4194304 > /proc/sys/kernel/shmall
```

Alternatively, you can use `sysctl(8)` to change it:

```
# sysctl -w kernel.shmall=4194304
```

To make the change permanent, set the value of SHMALL in the `/etc/sysctl.conf` file:

```
  kernel.shmall=2097152
```

then run the following command:

```
# sysctl -p
```

Removing Shared Memory

Sometimes after an instance crash you may have to remove Oracle's shared memory segment(s) manually.

To see all shared memory segments that are allocated on the system, execute:

```
$ ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x8f6e2129 98305      oracle    600        77694523   0
0x2f629238 65536      oracle    640        2736783360 35
0x00000000 32768      oracle    640        2736783360 0          dest
```

In this example you can see that three shared memory segments have been allocated. The output also shows that shmid 32768 is an abandoned shared memory segment from a past ungraceful Oracle shutdown. Status "`dest`" means that this memory segment is marked to be destroyed. To find out more about this shared memory segment, run:

```
$ ipcs -m -i 32768
Shared memory Segment shmid=32768
uid=500 gid=501 cuid=500 cgid=501
mode=0640 access_perms=0640
bytes=2736783360 lpid=3688 cpid=3652 nattch=0
att_time=Fri Sep 19 09:17:02 2008
det_time=Fri Sep 19 09:17:02 2008
change_time=Fri Sep 19 11:21:06 2008
```

To remove the shared memory segment, you could copy/paste *shmid* and execute:

```
$ ipcrm shm 32768
```

Summary of Recommended Huge Pages & Shared Memory Settings

For 64-bit systems with 64GB of RAM:

```
kernel.shmmax=34359738368
kernel.shmmni=4096
kernel.shmall=16777216
vm.nr_hugepages=16384
```

For 64-bit systems with 32GB of RAM:

```
kernel.shmmax=17179869184
kernel.shmmni=4096
kernel.shmall=8388608
vm.nr_hugepages=8192
```

For 64-bit systems with 16GB of RAM:

```
kernel.shmmax=8589934592
kernel.shmmni=4096
kernel.shmall=4194304
vm.nr_hugepages=4096
```

For 64-bit systems with 8GB of RAM:

```
kernel.shmmax=4294967295
kernel.shmmni=4096
kernel.shmall=2097152
vm.nr_hugepages=2048
```

For systems with 4GB of RAM:

```
kernel.shmmax=2147483648
kernel.shmmni=4096
kernel.shmall=1048576
vm.nr_hugepages=1024
```

For systems with 1GB of RAM:

```
kernel.shmmax=536870912
kernel.shmmni=4096
kernel.shmall=262144
```

Setting Semaphores

Semaphores can be described as counters which are used to provide synchronization between processes or between threads within a process for shared resources like shared memories. System V semaphores support semaphore sets where each one is a counting semaphore. So when an application requests semaphores, the kernel releases them in sets. The number of semaphores per set can be defined through the kernel parameter SEMMSL.

To see all semaphore settings, run:

```
# ipcs -ls

------ Semaphore Limits --------
max number of arrays = 128
max semaphores per array = 250
max semaphores system wide = 32000
max ops per semop call = 32
semaphore max value = 32767
```

You can also use the following command:

```
# cat /proc/sys/kernel/sem
250     32000   32      128
```

The SEMMSL Parameter

This parameter defines the maximum number of semaphores per semaphore set.

The recommended minimum value of SEMMSL is 250.

NOTE:
If a database gets thousands of concurrent connections where the ora.init parameter `PROCESSES` is very large, then SEMMSL should be larger as well. Referencing Metalink Notes:187405.1 and 184821.1, the SEMMSL setting should be 10 plus the largest PROCESSES parameter of any Oracle database on the system. Even though these notes reference9i databases, this SEMMSL rule also applies to 10g databases. An example for setting semaphores for higher `PROCESSES` settings can be found in the Example for Semaphore Settings section in this document.

The SEMMNI Parameter

This parameter defines the maximum number of semaphore sets for the entire Linux system. Oracle validates Oracle 10.2 and 11.1 configurations with SEMMNI = 142.

The SEMMNS Parameter

This parameter defines the total number of semaphores (not semaphore sets) for the entire Linux system. A semaphore set can have more than one semaphore, and as the `semget(2)` man page explains, values greater than SEMMSL * SEMMNI makes it irrelevant. The maximum number of semaphores that can be allocated on a Linux system will be the lesser of: SEMMNS or (SEMMSL * SEMMNI).

Oracle recommends SEMMNS to be at least 32000 and uses this value in its validated configurations. This value corresponds to SEMMSL * SEMMNI (250*128=32000) semaphores. With SEMMNI increased to 142 the value of SEMMNS can be also increased to 35500 or more if the system is using more processes and SEMMSL value is more than 250.

The minimum recommended value of SEMMNS is 32000.

The SEMOPM Parameter

This parameter defines the maximum number of semaphore operations that can be performed per `semop(2)` system call (semaphore call). The `semop(2)` function provides the ability to do operations for multiple semaphores with one `semop(2)` system call. Since a semaphore set can have the maximum number of SEMMSL semaphores per semaphore set, it is often recommended to set SEMOPM equal to SEMMSL.

Oracle validates Oracle DB 10.2 and 11.1 configuations with SEMOPM = 100.

Setting Semaphore Parameters

To determine the values of the four described semaphore parameters, run:

```
# cat /proc/sys/kernel/sem
250     32000   32      128
```

These values represent (in order) SEMMSL, SEMMNS, SEMOPM, and SEMMNI.

Alternatively, you can run:

```
# ipcs -ls
```

All four described semaphore parameters can be changed in the proc file system without reboot:

```
# echo 250 32000 100 142 > /proc/sys/kernel/sem
```

The values shown above are used by Oracle for validation of Oracle DB 10.2 and 11.1 configurations.

Alternatively, you can use `sysctl(8)` to change it:

```
# sysctl -w kernel.sem="250 32000 100 142"
```

To make the change permanent, add or change the following line in the file `/etc/sysctl.conf`. This file is used during the boot process.

```
# echo "kernel.sem=250 32000 100 142" >> /etc/sysctl.conf
```

Example for Semaphore Settings

On systems where the ora.init parameter `PROCESSES` is very large, the semaphore settings need to be adjusted

accordingly.

As shown in the The SEMMSL Parameter section above, the SEMMSL setting should be 10 plus the largest PROCESSES parameter of any Oracle database on the system. So if you have one database instance running on a system where PROCESSES is set to 5000, then SEMMSL should be set to 5010.

As shown in the The SEMMNS Parameter section above, the maximum number of semaphores that can be allocated on a Linux system will be the lesser of: SEMMNS or (SEMMSL * SEMMNI). Since SEMMNI should be set to 142, we need to increase SEMMNS to 711420 (5010*142).

As shown in the The SEMOPM Parameter section above, a semaphore set can have the maximum number of SEMMSL semaphores per semaphore set and it is recommended to set SEMOPM equal to SEMMSL. Since SEMMSL is set to 5010 the SEMOPM parameter should be set to 5010 as well.

Hence, if the ora.init parameter PROCESSES is set to 5000, then the semaphore settings should be as follows:

```
# sysctl -w kernel.sem="5010 711420 5010 142"
```

Setting File Handles

The maximum number of file handles specifies the maximum number of open files on a Linux system. Oracle recommends that the file handles for the entire system is set to at least 65536.

To determine the maximum number of file handles for the entire system, run:

```
# cat /proc/sys/fs/file-max
49254
```

To determine the current usage of file handles, run:

```
# cat /proc/sys/fs/file-nr
5056    0       49254
```

The file-nr file displays three parameters:

- Total allocated file handles
- Currently number of unused file handles (2.6 kernel)
- Maximum file handles that can be allocated (see also /proc/sys/fs/file-max)

The kernel dynamically allocates file handles whenever a file handle is requested by an application but the kernel does not free these file handles when they are released by the application. The kernel recycles these file handles instead. This means that over time the total number of allocated file handles will increase even though the number of currently used file handles may be low.

Note that using ASMLib driver decreases the total number of open files required for Oracle processes because they do not have to open files or block devices directly.

The maximum number of file handles can be changed in the proc file system without reboot:

```
# echo 65536 > /proc/sys/fs/file-max
```

Alternatively, you can use `sysctl(8)` to change it:

```
# sysctl -w fs.file-max=65536
```

To make the change permanent, add or change the following line in the file `/etc/sysctl.conf`. This file is used during the boot process. Oracle uses the following value in its validated configurations:

```
# echo "fs.file-max=327679" >> /etc/sysctl.conf
```

Adjusting Network Settings

Changing Network Kernel Settings

Oracle RAC now uses UDP as the default protocol on Linux for interprocess communication, such as cache fusion buffer transfers between the instances. Starting with Oracle 10g, the network settings should be adjusted for standalone databases as well.

Linux kernel 2.6 auto-tunes the default and maximum values of send and receive socket buffers. Oracle recommends increasing them to satisfy installer checks. The kernel auto-tuning takes precedence of the specified default values but the specified static maximum takes precedence of the auto-tuned maximum.

Oracle recommends the default and maximum send buffer size (`SO_SNDBUF` socket option) and the default receive buffer size (`SO_RCVBUF` socket option) to be set to 256 KB. The recommended maximum receiver buffer size is 2MB for Oracle 10.2 and 4MB for Oracle 11.1. The receive buffers are used by TCP and UDP to hold the received data for the application until its read. This buffer cannot overflow because the sending party is not allowed to send data beyond the buffer size window. This means that datagrams will be discarded if they do not fit in the receive buffer. This could cause the sender to overwhelm the receiver.

The default and maximum window size can be changed in the proc file system without reboot:

```
Default setting in bytes of the socket receive buffer

# sysctl -w net.core.rmem_default=262144

Default setting in bytes of the socket send buffer

# sysctl -w net.core.wmem_default=262144

Maximum socket receive buffer size which may be set by using the SO_RCVBUF socket option

# sysctl -w net.core.rmem_max=4194304

Maximum socket send buffer size which may be set by using the SO_SNDBUF socket option


# sysctl -w net.core.wmem_max=262144
```

To make the change permanent, add the following lines to the `/etc/sysctl.conf` file, which is used during the boot process:

```
net.core.rmem_default=262144
net.core.wmem_default=262144
net.core.rmem_max=4194304
net.core.wmem_max=262144
```

Minimum, default and maximum values of TCP socket send and receive buffers should correspond to the settings above:

```
net.ipv4.tcp_rmem=4096 262144 4194304
net.ipv4.tcp_wmem=4096 262144 262144
```

To improve failover performance in a RAC cluster, consider changing the following IP kernel parameters as well:

```
net.ipv4.tcp_keepalive_time=30
net.ipv4.tcp_keepalive_intvl=60
net.ipv4.tcp_keepalive_probes=9
net.ipv4.tcp_retries2=3
net.ipv4.tcp_syn_retries=2
```

Changing these settings may be highly dependent on your system, network, and other applications. For suggestions, see Metalink Note:249213.1.

On Red Hat Enterprise Linux systems, the default range of IP port numbers that are allowed for TCP and UDP traffic is too low for Oracle database servers. Oracle recommends the following port range:

```
# sysctl -w net.ipv4.ip_local_port_range="1024 65000"
```

To make the change permanent, add the following line to the `/etc/sysctl.conf` file, which is used during the boot process:

```
net.ipv4.ip_local_port_range=1024 65000
```

The first number is the first local port allowed for TCP and UDP traffic and the second number is the last port number.

Flow Control on NICs and Network Switches

Verify that NICs and network switches are using flow control. If you have heavy traffic, then the RAC interconnects may lose blocks, see Metalink Bug:5058952.

To verify flow control settings on the NIC use `ethtool`:

```
# ethtool -k eth0
```

Make sure you have flow control enabled in both direction: for transmit (Tx) and receive (Rx). Some switches have flow control enabled by default for Rx only. Ask you network administrator to fix switch settings, if necessary.

Some drivers may require driver parameters in the `/etc/modprobe.conf` file. For example, the driver e1000 may be used with flow control parameter specified as

```
option e1000 FlowControl=1,1
```

It is recommended to configure network switches and NICs for autonegotiation and let them agree on the right combination of flow control settings.

Network Interface Bonding

Conceptually the NIC bonding is a method for aggregating multiple network interfaces into a single logical bonded interface. The behavior of the bonded interfaces depends upon the mode; generally speaking, modes provide either hot standby or load balancing services.

The most popular is the active-backup mode. This mode provides fault tolerance. Only one slave interface in the bond is active. A different slave becomes active if, and only if, the active slave fails. The bond's MAC address is externally visible on only one port (network adapter) to avoid confusing the switch.

If active-backup mode is used on Oracle 10g RAC interconnect and the interconnect is implemented with the separate switches, it may be desirable to configure primary interfaces on all cluster nodes to be connected to the same switch. In this case, the interconnect traffic is localized on the same switch and avoids interswitch link.

The adaptive load balancing mode (balance-alb) provides transmit and receive load balancing and does not require any special switch support.

There are a couple of issues to consider. The first is that by default, when bringing up a server with bond devices in the configuration, the NIC drivers will not have been loaded yet when the bonding driver initializes. Therefore it is entirely possible that the bonding driver's attempts to start the interfaces belonging to that bond will result in those NIC's receiving different device names from the kernel. So the best way to start bonding is to force the system to load the NIC drivers first, in the same order your `/etc/modprobe.conf` shows. For example, if your eth0 and eth1 devices use bnx2, and eth2 and eth3 use e1000 drivers, then you want to modprobe bnx2 first, then e1000. After that you can load the bonding driver and it should find the NICs it is looking for where it expects to find it (with the correct device name). The order in which NIC drivers are loaded determines the *ethX* device names the kernel will assign .

The second issue is the syntax necessary to load the bonding driver multiple times. In order to use different bonding options on different bond devices, you have to load the bonding driver multiple times. Specifying "max_bonds=X" (where X is > 1) will not provide multiple bond devices capable of having different options.

The following two lines are 'install' lines from the `/etc/modprobe.conf`. Note that there should not be any 'alias bond0 bonding' lines for any bond interface as this will confuse the bonding driver and/or system initscripts. These two lines are the only bonding related lines that should be present in `/etc/modprobe.conf`:

```
install bond0  /sbin/modprobe bnx2; /sbin/modprobe e1000; \
  /sbin/modprobe -i bonding -o bond0 mode=balance-alb \
  arp_ip_target=192.168.0.19 arp_interval=200

install bond1 /sbin/modprobe bnx2; /sbin/modprobe e1000; \
  /sbin/modprobe -i bonding -o bond1 mode=1 \
```

```
miimon=100 downdelay=300 updelay=300 primary=eth1
```

Note that *bond0* uses balance-alb mode and ARP-based monitoring, whereas *bond1* calls for mode 1 (failover) using mii link status monitoring. These are provided as working syntax examples. You can run both with miimon, if preferred, or other differences in your optionsbut both issues mentioned above are covered. We force the kernel to load the NIC modules first to address the device "renaming"/"reordering" issues, and the syntax showed above will allow multiple instances of the bonding driver to be loaded into the kernel so that different options can be used on each bond device. One will be named *bond0*, the other will be named *bond1*.

If you have several network adapters with multiple interfaces, you can eliminate a single point of failure by bonding interfaces from different network adapters. For example, the above case configures *bond0* using eth0 and eth2 while *bond1* uses eth1 and eth3.

Note that any time you need to change the bonding configuration on the system, you will need to stop the network ('service network stop'), then remove all instances of the bonding driver ('rmmod bonding', 'rmmod bond0', etc.). Then restart the network ('service network start') and the new configuration changes should take effect.

Use the output of 'cat /proc/net/bonding/bond0' (or bond1, etc.) to confirm that the desired bonding configuration options are configured.

Changing Network Adapter Settings

To check the speed and settings of network adapters, use the `ethtool` command which works now for most NICs. For example, to check the adapter settings of eth0, run:

```
# ethtool eth0
```

To force a speed change to 1000 full duplex, run:

```
# ethtool -s eth0 speed 1000 duplex full autoneg off
```

To make a speed change permanent for `eth0`, set or add the `ETHTOOL_OPT` environment variable in `/etc/sysconfig/network-scripts/ifcfg-eth0`:

```
ETHTOOL_OPTS="speed 1000 duplex full autoneg off"
```

This environment variable is sourced in by the network service scripts each time the network is started. However, some network interface parameters can not be specified as ETHTOOL_OPTS options. For example, ethtool options -A and -K are not processed through ETHTOOL_OPTS (see the `ethtool` manpage for details). These options can be used to set tx/rx flow control and enable/disable the TCP Segmentation Offload (TSO) feature on the network card. TSO, also known as "large send", enables the protocol stack to offload portions of outbound TCP processing to a network interface card, reducing CPU utilization and improving performance.

The corresponding ethtool command can be placed into an "install" statement in `/etc/modprobe.conf`. Make sure tx/rx flow control on the network interface corresponds to the settings on the switch.

For example, to disable TSO on e1000 interfaces:

```
install bond0  /sbin/modprobe bnx2; /sbin/modprobe e1000; \
```

```
    /sbin/ethtool -K eth2 tso off; \
    /sbin/modprobe -i bonding -o bond0 \
    mode=active-backup miimon=100 downdelay=300 updelay=300 primary=eth0

  install bond1  /sbin/modprobe bnx2; /sbin/modprobe e1000; \
  /sbin/ethtool -K eth3 tso off; \
    /sbin/modprobe -i bonding -o bond1 \
    mode=balance-alb miimon=100 downdelay=300 updelay=300
```

This example also shows the *bond1* interface configured for load balancing. It may improve network throughput for Oracle 10g RAC installations with dedicated switches for cluster interconnect.

Configure the PortFast feature on Cisco Ethernet switches for ports connected to the host. This should decrease time required for link negotiation, especially for Gigabit links.

If all hosts on your LAN and network switches support jumbo frames ,you can enable this feature on network interfaces to improve average network throughput and decrease CPU utilization. The real advantage can be achieved only if your application is sending large amount of data through TCP or big UDP messages.

To configure jumbo frames, put the parameter `MTU=9000` into the `/etc/sysconfig/network-scripts/ifcfg-ethX` file.

Setting Shell Limits for the Oracle User

Most shells like bash provide control over various resources like the maximum allowable number of open file descriptors or the maximum number of processes available to a user.

To see all shell limits, run:

```
# ulimit -a
```

For more information on `ulimit` for the bash shell, see `man bash` and search for 'ulimit'.

Limiting Maximum Number of Open File Descriptors for the Oracle User

After `/proc/sys/fs/file-max` has been changed (see the Setting File Handles section of this document) there is still a per user limit of maximum open file descriptors:

```
$ su - oracle
$ ulimit -n
1024
$
```

To change this limit, edit the `/etc/security/limits.conf` file as root and make the following changes or add the following lines, respectively:

```
  oracle           soft    nofile          131072
```

```
oracle           hard    nofile          131072
```

These are the recommended values used in Oracle validated configurations. The "soft limit" in the first line defines the number of file handles or open files that the Oracle user will have after login. You can set the "soft" and "hard" limits higher if necessary.

NOTE:
It is not recommend to set the "hard" limit for `nofile` for the `oracle` user equal to `/proc/sys/fs/file-max`. If you do that and the user uses all the file handles, then the entire system will run out of file handles. This could mean that you will not be able to initiate new logins any more since the system will not be able to open any PAM modules that are required for the login process.

To apply these limits you also need to ensure that `pam_limits` is configured in the `/etc/pam.d/system-auth` file, or in `/etc/pam.d/sshd` for ssh, `/etc/pam.d/su` for su, or `/etc/pam.d/login` for local logins and telnet if you do not want to enable it for all login methods. Here are examples of the two session entries in the `/etc/pam.d/system-auth` file:

```
session     required      pam_limits.so
session     required      pam_unix.so
```

Limiting Maximum Number of Processes for the Oracle User

To see the current limit of the maximum number of processes for the `oracle` user, run:

```
$ su - oracle
$ ulimit -u
```

Note the ulimit options are different for other shells.

To change the "soft" and "hard" limits for the maximum number of processes for the `oracle` user, add the following lines to the `/etc/security/limits.conf` file:

```
oracle           soft    nproc           131072
oracle           hard    nproc           131072
```

These are the recommended values for Oracle validated configurations.

Increasing Maximum Size of Per-Processes Locked Memory

To use `hugepages` you must increase the default maximum size of the per-process locked memory. To increase the per-process max locked memory limit, add the following lines to the `/etc/security/limits.conf` file:

```
oracle           soft    memlock         3145728
oracle           hard    memlock         3145728
```

The size is in KB and should correspond to the size of *hugepages* allocated to Oracle. In the example above, the memlock value allows to use 1536 huge pages (3145728 KB / (1024 KB * 2) = 1536 huge pages).

Enabling Asynchronous I/O and Direct I/O Support

Asynchronous I/O permits Oracle to continue processing after issuing I/Os requests, which leads to higher I/O performance. Red Hat Enterprise Linux also allows Oracle to issue multiple simultaneous I/O requests with a single system call. This reduces context switch overhead and allows the kernel to optimize disk activity.

Oracle 10g Release 2 is shipped with asynchronous I/O support enabled by default. It does not need to be re-linked as previous versions of Oracle database servers but you may have to apply a patch, see below.

Enabling Asynchronous I/O in Oracle 10g

To enable async I/O in Oracle, the `disk_asynch_io` parameter needs to be set to true in the init.ora file:

```
disk_asynch_io=true
```

Note this parameter is set to true by default in Oracle 10g:

```
SQL> show parameter disk_asynch_io;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------
disk_asynch_io                       boolean     TRUE
SQL>
```

If you use file systems instead of raw devices or ASM for data files, then you need to ensure that the data files reside on file systems that support asynchronous I/O (e.g., ext3, gfs, etc.). To do async I/O on file systems, the `filesystemio_options` parameter needs to be set to "asynch" in addition to `disk_asynch_io=true`:

```
filesystemio_options=asynch
```

This parameter is platform-specific. By default, this parameter is set to none for Linux and thus needs to be changed:

```
SQL> show parameter filesystemio_options;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------
filesystemio_options                 string      none
SQL>
```

The `filesystemio_options` variable can have the following values: `asynch:` Enables asynchronous I/O on file system files
`directio:` Enables direct I/O on file system files
`setall:` Enables both asynchronous and direct I/O on file system files
`none:` Disables both asynchronous and direct I/O on file system files

If you also wish to enable Direct I/O Support, set `filesystemio_options` to "setall".

For Red Hat Enterprise Linux 5, it is strongly recommended to use "setall" for ext2, ext3, GFS, NFS and

OCFS2 file systems.

Checking Asynchronous I/O Usage

To verify whether `$ORACLE_HOME/bin/oracle` was linked with async I/O, you can use the Linux commands `ldd` and `nm`.

In the following example, `$ORACLE_HOME/bin/oracle` was relinked with async I/O:

```
$ ldd $ORACLE_HOME/bin/oracle | grep libaio
        libaio.so.1 => /usr/lib/libaio.so.1 (0x0093d000)
$ nm $ORACLE_HOME/bin/oracle | grep io_getevent
         w io_getevents@@LIBAIO_0.1
```

In the following example, `$ORACLE_HOME/bin/oracle` has NOT been relinked with async I/O:

```
$ ldd $ORACLE_HOME/bin/oracle | grep libaio
$ nm $ORACLE_HOME/bin/oracle | grep io_getevent
         w io_getevents
```

If `$ORACLE_HOME/bin/oracle` is relinked with async I/O, it does not necessarily mean that Oracle is really using it. You also need to ensure that Oracle is configured to use async I/O calls, see the Enabling Asynchronous I/O Support section of this document.

To verify whether Oracle is making async I/O calls, you can examine the `/proc/slabinfo` file assuming there are no other applications performing async I/O calls on the system. This file shows kernel slab cache information in real time.

Once Oracle makes async I/O calls, the output will look like this:

```
$ egrep "kioctx|kiocb" /proc/slabinfo
kioctx 64    110    384   10    1 : tunables   54   27    8 : slabdata     11     11
0
kiocb   3     15    256   15    1 : tunables  120   60    8 : slabdata      1      1
0
```

The numbers in red (number of active objects) show whether Oracle makes async I/O calls. The first column displays the cache names `kioctx` and `kiocb`. The second column shows the number of active objects currently in use.

To see kernel slab cache information in real time, you can also use the `slabtop` command:

```
$ slabtop
 Active / Total Objects (% used)    : 293568 / 567030 (51.8%)
 Active / Total Slabs (% used)      : 36283 / 36283 (100.0%)
 Active / Total Caches (% used)     : 88 / 125 (70.4%)
 Active / Total Size (% used)       : 81285.56K / 132176.36K (61.5%)
 Minimum / Average / Maximum Object : 0.01K / 0.23K / 128.00K

  OBJS ACTIVE  USE OBJ SIZE  SLABS OBJ/SLAB CACHE SIZE NAME
178684  78396  43%    0.12K   5764       31     23056K size-128
127632  36292  28%    0.16K   5318       24     21272K dentry_cache
102815  74009  71%    0.69K  20563        5     82252K ext3_inode_cache
```

```
 71775  32434  45%    0.05K    957       75      3828K buffer_head
 19460  15050  77%    0.27K   1390       14      5560K radix_tree_node
 13090  13015  99%    0.03K    110      119       440K avtab_node
 12495  11956  95%    0.03K    105      119       420K size-32
...
```

Slab caches are a special memory pool in the kernel for adding and removing objects (e.g., data structures or data buffers) of the same size. It is a cache for commonly used objects where the kernel does not have to re-allocate and initialize the object each time it is being reused, and subsequently free the object each time it is being destroyed. The slab allocator scheme basically prevents memory fragmentation and it prevents the kernel from spending too much time allocating, initializing, and freeing the same objects.

The kernel parameter `/proc/sys/fs/aio-max-nr` limits the number of concurrent asynchronous I/O requests. The default value 65536 (64K) may be too low for Oracle database servers. You can monitor the current number of the concurrent I/O requests represented in `/proc/sys/fs/aio-nr` and increase the above limit if necessary. For example,

```
# cat /proc/sys/fs/aio-nr
65536
# echo 3145728 > /proc/sys/fs/aio-max-nr
```

The recommended value for Oracle validated configurations is 3145728.

Storage Configuration for Oracle 10g RAC and Standalone Server

Red Hat Enterprise Linux provides variety of options for storage management. Red Hat Enterprise Linux 5 supports dynamic device configuration and re-configuration, native multipathing, logical volume manager, GFS cluster file system (can also be used in the standalone mode), ext3, NFS file systems, and iSCSI initiator. n addition, Oracle provides OCFS2 file system and Automatic Storage Management (ASM) for Oracle database files.

Oracle database storage management provides the following choices for placing database files:

1. Traditional file systems – standalone and cluster-aware

2. Standard UNIX raw devices

3. ASM
   - on raw devices
   - on block devices with ASMLib
   - on block devices without ASMLib

Raw devices are deprecated in Red Hat Enterprise Linux 5. The implementation of raw devices in Linux is just a wrapper that opens underlying block devices with O_DIRECT flag for direct I/O. The raw devices do not give any performance benefits to Oracle database server. They are required only for Oracle 10g R2 RAC with Clusterware 10.2.0. We will not consider raw device configuration for any other purposes.

Oracle 10g R2 is certified on Red Hat Enterprise Linux 5 with ext3, GFS, OCFS2, and ASM storage options in

the standalone and RAC configurations. The NFS file system is supported with certified NFS servers.

The main difficulties with storage configuration are related to the dynamic device detection processes in the Linux kernel. Devices may be recognized in the different order across reboots and as a result, their kernel names are not persistent. Usually, this is not a problem for devices with file systems or devices included into LVM volume groups because tools used to format file systems or create volumes can also create unique labels that can be used as persistent references to the devices. Oracle Clusterware and ASM require additional configuration steps to ensure device name persistence and set the right combination of device ownership and access permissions.

We will consider the two most common storage configurations:

- Oracle 10g RAC or standalone server with single path storage devices and ASM. This is typical for virtual machines or non-production environments.

- Oracle 10g RAC or standalone server with Linux native multipathing and ASM.

The difference between RAC and standalone configurations is in the Oracle Clusterware, a RAC foundation component requiring special storage configuration for its files, the Oracle Cluster Registry (OCR), its optional mirror(s), and the Cluster Synchronization Services (CSS) voting disk(s). In the production setup, Oracle RAC should be configured with at least two OCR disks (primary and mirror) and three voting disks. Each OCR and voting disk should be at least 300MB in size. The minimal setup requires one OCR and one voting disk. The Clusterware files can be created on the partitioned devices.

It is recommended to place the Server Parameter File (SPFILE) for the Oracle ASM instance on a separate partition. It may coexist on the same LUN with one of OCR and voting disk partitions. The recommended size is 300MB.

Oracle 10g server with ASM uses at least two storage areas:

- The first is for database datafiles, online redo log files, control files, SPFILES for database instances, and temp files for temporary tablespaces.

- The second is for the Oracle Flash Recovery. It should be twice the size of the database data area.

It is recommended to separate the above storage areas onto their own LUNs on separate disk groups. The separation can enable better I/O performance by ensuring these files do not share the same physical disks.

Configuring Storage Access with Device Mapper Multipath

Red Hat Enterprise Linux supports large variety of storage devices with native Linux multipathing. As an example, we will show the configuration for EMC CLARiiON and SYMMETRIX storage arrays.

The following steps detail the procedure for configuring multipath on a Red Hat Enterprise Linux 5 host.

1. If the *device-mapper-multipath* package is not installed as part of the default operating system installation install the package:

   # yum install device-mapper-multipath

2. Load the dm_multipath kernel module if it is not already loaded.

   # modprobe dm_multipath

3. Replace the default /etc/multipath.conf with the following multipath.conf file.

```
defaults {
    path_grouping_policy failover
    user_friendly_names yes
}
#multipaths {
#    multipath {
#        wwid                    3600508b4000156d700012000000b0000
#        alias                   asm1
#    }
#}
devices {
    ##
    ## Device attributes for EMC CLARiiON
    device {
        vendor "DGC"
        product "*"
        product_blacklist “(LUNZ|LUN_Z)”
        path_grouping_policy group_by_prio
        getuid_callout "/sbin/scsi_id -g -u -s /block/%n"
        prio_callout "/sbin/mpath_prio_emc /dev/%n"
        path_checker emc_clariion
        path_selector "round-robin 0"
        features "1 queue_if_no_path"
        no_path_retry 300
        hardware_handler "1 emc"
        failback immediate
    }
    ## Device attributes for EMC Symmetrix
    device {
        vendor “EMC”
        product “SYMMETRIX”
        path_grouping_policy multibus
        getuid_callout “/sbin/scsi_id -g -u -ppre-spc3-83 -s /block/%n”
        path_checker readsector0
        features “1 queue_if_no_path”
        no_path_retry 60
        hardware_handler “0”
        rr_weight uniform
        rr_min_io 10
}
```

4. Optionally you can assign aliases in *multipaths* section. The value of WWID can be obtained with *scsi_id* utility. For example:

   ```
   # scsi_id -g -u -s /block/sda
   ```

   In Oracle RAC, the clusterware device names must be persistent across reboots. The aliases can help to create human friendly persistent named for the OCR (primary and mirror) disks, CSS voting disks and also for ASM disks (if using ASM without ASMLib).

5. Perform a multipath dry run and evaluate the setup by running the following command.

   ```
   # multipath -v2 -d
   ```

6. If the listing is appropriate, commit the configuration as follows:

   a. Start multipath daemon

   ```
   # service multipathd start
   ```

   b. Execute the multipath command:

   ```
   # multipath -v2
   ```

7. To get a listing of the current setup do:

   ```
   # multipath -ll
   ```

8. Integrate the startup of the appropriate daemons in the boot sequence as follows:

   ```
   # chkconfig --add multipathd
   # chkconfig multipathd on
   ```

The device mapper creates kernel block devices (/dev/dm-*) and block devices with persistent names in the `/dev/mapper` directory. The Red Hat Enterprise Linux 5 `udev` facility reacts to the event of creating a new `/dev/dm-*` device by executing rules specified in `/etc/udev/rules.d/40-multipath.rules`. In particular, it creates symlinks under `/dev/mpath` directory pointing to the `/dev/dm-*` name.

Optimization of Disk Partitions on RAID

Storage arrays usually stripe data across several physical disk drives in large chunks, typically 64KB in size. Aligning partitions to these stripe sizes can improve performance, particularly with database servers using direct I/O.

Red Hat Enterprise Linux 5 contains two utilities that can be used to create disk partitions: `fdisk` and `parted`. To ensure maximum compatibility with other operating systems, `fdisk` creates partitions that are aligned to cylinder boundaries using the legacy cylinder/head/sector (C/H/S) geometry. Usually this geometry is not aligned to the stripes in a RAID array. If a RAID volume is only going to be accessed by operating systems and utilities that fully support Logical Block Addressing (LBA), then C/H/S alignment is not necessary and stripe alignment can be created with the `parted` tool.

Because it supports larger volumes, `parted` is the recommended tool to create partitions on Red Hat Enterprise Linux 5.

The disk management utilities for x86/x86_64 based systems generally write 63 sectors of metadata (msdos label or partition table) directly at the beginning of the LUN. The addressable space begins immediately after these initial sectors causing a misalignment with the RAID strips and, as a result, excessive cross disk accesses leading to performance degradation.

To avoid this issue, the partition should be aligned so that its data space starts on the same sector where RAID stripe begins. For example, if the CLARiiON array has 64KB striping, the first writable sector of the partition should be sector 128 (128 * 512 bytes = 64KB). Below is an example of how to create a partition on a CLARiiON storage array using `parted` utility. It is assumed that the LUN is available through the device-mapper-multipath as device `/dev/mapper/asm1`.

```
# /sbin/parted /dev/mapper/asm1 mklabel msdos unit s mkpart primary “128 -1”
Information: Don't forget to update /etc/fstab, if necessary.
```

Check the results:

```
# /sbin/parted /dev/mapper/asm1 unit s print

Model: Linux device-mapper (dm)
Disk /dev/mapper/asm1: 251658239s
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start  End         Size        Type     File system  Flags
 1      128s   251658224s  251658097s  primary

Information: Don't forget to update /etc/fstab, if necessary.
```

Configuring Raw Devices for Oracle 10g RAC

Oracle RAC 10g R2 can be installed with Clusterware 10.2.0 or 11.1. Oracle Universal Installer for Oracle 10g R2 RAC can't verify visibility of Clusterware devices from all cluster nodes unless they are raw devices. As a result, the installation of Clusterware 10.2.0 requires raw devices. Oracle 10g R2 database server and ASM do not require raw devices but can use them. Starting from Oracle 11g Release 1, Clusterware files can be placed on either block or raw devices located on shared disk partitions. Using Cluserware 11.1 with Oracle RAC 10g R2 is simpler because it does not require raw devices. Nevertheless, below we are describing how to configure raw devices with single path and multipath LUNs.

Let us assume that the LUNs have been created on a supported shared storage, appropriately sized, partitioned, and visible from all cluster nodes. In multipath setup, the same LUN is visible several times and the operating system creates a separate SCSI device (/dev/sd*) for each path.

1. Verify visibility of the Clusterware devices and partitions:

```
# cat /proc/partitions
```

Partitions on the Clusterware devices may be not visible if they have been created on another cluster node or on iSCSI target. In this case, re-read the partitions with `/bin/partprobe <device>`.

2. Obtain unique SCSI disk identifiers of the Clusterware devices:

```
# for i in `awk '/sd/ {print $4}' /proc/partitions`; \
  do echo "$i: `scsi_id -g -u -s /block/$i`"; done
```

These identifiers are required to create persistent device names.

3A. Creating persistent raw devices for single path LUNs

Red Hat Enterprise Linux 4 had a *rawdevices* service and the `/etc/sysconfig/rawdevices` file for binding raw devices to block devices. Red Hat Enterprise Linux 5 has deprecated raw devices. However, it is possible to use the `udev` facility to configure raw devices and set their permissions.

On Red Hat Enterprise Linux 5 system, the *util-linux* package provides the default udev raw mapping file `/etc/udev/rules.d/60-raw.rules`.

Create a custom udev raw mapping rule file, `/etc/udev/rules.d/61-oracleraw.rules`. For example, for unpartitioned block device with WWID=360a98018486f6959684a45333527173 add the following rule:

```
# Raw bind to unpartitioned Oracle Clusterware devices
ACTION=="add", KERNEL=="sd*[!0-9]", PROGRAM=="/sbin/scsi_id -g -u -s %p",
RESULT=="360a98018486f6959684a453333527173", RUN+="/bin/raw /dev/raw/raw1 %N"
```

This will create the raw device `/dev/raw/raw1` that will be persistently bound to the LUN with the specified WWID.

Test the udev configuration with the `udevtest` utility:

```
# udevtest /block/sda | grep raw
```

If the block device is partitioned the rule should look like:

```
# Raw bind to partitioned Oracle Clusterware devices
ACTION=="add", KERNEL=="sd*[0-9]", PROGRAM=="/sbin/scsi_id -g -u -s %p",
RESULT=="360a98018486f6959684a453333527173", RUN+="/bin/raw /dev/raw/raw%n %N"
```

This will create raw devices `/dev/raw/raw1, /dev/raw/raw2`, etc. for each partition on the LUN and they will be persistently bound to the LUN with the specified WWID.

To test the configuration run, for example,

```
# udevtest /block/sda/sda1 | grep raw
```

Finally, to actually create the raw devices, use the `start_udev` command:

```
# start_udev
Starting udev:                             [OK]
```

Check the raw device binding:

```
# raw -qa
```

3B. Creating persistent raw devices for multipath LUNs

It is assumed that the *device-mapper-multipath* package has been configured as described above. For the example below, we made following assumptions:

1. OCR primary and mirror files and CSS voting disks are not partitioned.

2. The *multipaths* section of the `/etc/multipath.conf` file contains aliases for OCR primary and mirror disks (ocr1 and ocr2) as well as CSS voting disks (vote[1-3]).

3. Devices for ASM have only one partition.

4. If ASM is configured to use block devices directly without ASMLib driver, then the ASM devices should have aliases (asm*) in the `/etc/multipath.conf` file.

Device mapper

a) creates `/dev/dm-*` block devices for each multipath group
b) createsblock devices with WWID or user friendly names (or corresponding aliases if they were specified) under the `/dev/mapper` directory
c) `udev` creates symlinks under the /`dev/mpath` directory pointing to `/dev/dm-*` devices

The original Red Hat Enterprise Linux 5 udev configuration for multipath devices is in `/etc/udev/rules.d/40-multipath.rules`. It should be used as a basis for any customization required by Oracle 10g RAC and ASM.

Copy the `40-multipath.rules` file to `39-oracle-multipath.rules` and make amendments. The original `40-multipath.rules` file looks like:

```
# multipath wants the devmaps presented as meaninglful device names
# so name them after their devmap name
SUBSYSTEM!="block", GOTO="end_mpath"
KERNEL!="dm-[0-9]*", ACTION=="add", \
  PROGRAM=="/bin/bash -c '/sbin/lsmod | /bin/grep ^dm_multipath'", \
  RUN+="/sbin/multipath -v0 %M:%m"
KERNEL!="dm-[0-9]*", GOTO="end_mpath"
PROGRAM!="/sbin/mpath_wait %M %m", GOTO="end_mpath"
ACTION=="add", \
  RUN+="/sbin/dmsetup ls --target multipath --exec '/sbin/kpartx -a -p p' -j %M -m %m"
PROGRAM=="/sbin/dmsetup ls --target multipath --exec /bin/basename -j %M -m %m",\
  RESULT=="?*", NAME="%k", SYMLINK="mpath/%c", OPTIONS="last_rule"
PROGRAM!="/bin/bash -c '/sbin/dmsetup info -c --noheadings -j %M -m %m | \
  /bin/grep -q .*:.*:.*:.*:.*:.*:.*:part[0-9]*-mpath-'", GOTO="end_mpath"
PROGRAM=="/sbin/dmsetup ls --target linear --exec /bin/basename -j %M -m %m", \
  NAME="%k", RESULT=="?*", SYMLINK="mpath/%c", OPTIONS="last_rule"
LABEL="end_mpath"
```

The amendments in `39-oracle-multipath.rules` file are highlighted:

```
# multipath wants the devmaps presented as meaninglful device names
# so name them after their devmap name
SUBSYSTEM!="block", GOTO="end_mpath"
KERNEL!="dm-[0-9]*", ACTION=="add", \
  PROGRAM=="/bin/bash -c '/sbin/lsmod | /bin/grep ^dm_multipath'", \
  RUN+="/sbin/multipath -v0 %M:%m"
KERNEL!="dm-[0-9]*", GOTO="end_mpath"
PROGRAM!="/sbin/mpath_wait %M %m", GOTO="end_mpath"
ACTION=="add", \
  RUN+="/sbin/dmsetup ls --target multipath --exec '/sbin/kpartx -a -p p' -j %M -m %m"
PROGRAM=="/sbin/dmsetup ls --target multipath --exec /bin/basename -j %M -m %m",\
  RESULT=="?*", NAME="%k", SYMLINK="mpath/%c", OPTIONS="last_rule" , GOTO="oracle_rules"
PROGRAM!="/bin/bash -c '/sbin/dmsetup info -c --noheadings -j %M -m %m | \
  /bin/grep -q .*:.*:.*:.*:.*:.*:.*:part[0-9]*-mpath-'", GOTO="end_mpath"
PROGRAM=="/sbin/dmsetup ls --target linear --exec /bin/basename -j %M -m %m", \
 NAME="%k", RESULT=="?*", SYMLINK="mpath/%c", OPTIONS="last_rule" , GOTO="oracle_rules"
LABEL="oracle_rules"
RESULT=="ocr1", GROUP="oinstall", MODE="640", \
  RUN+="/bin/raw /dev/raw/raw1 /dev/mapper/%c"
RESULT=="ocr2", GROUP="oinstall", MODE="640", \
  RUN+="/bin/raw /dev/raw/raw2 /dev/mapper/%c"
RESULT=="vote1", OWNER="oracle", GROUP="oinstall", MODE="660", \
  RUN+="/bin/raw /dev/raw/raw3 /dev/mapper/%c"
RESULT=="vote2", OWNER="oracle", GROUP="oinstall", MODE="660", \
  RUN+="/bin/raw /dev/raw/raw4 /dev/mapper/%c"
RESULT=="vote3", OWNER="oracle", GROUP="oinstall", MODE="660", \
  RUN+="/bin/raw /dev/raw/raw5 /dev/mapper/%c"
# Set owner, group and mode parameters on ASM devices. Not required with ASMLIB
#RESULT=="asm*p1", OWNER="oracle", GROUP="dba", MODE="660", \
#  RUN+="/bin/chown oracle:dba /dev/mapper/%c; /bin/chmod 660 /dev/mapper/%c"
OPTIONS="last_rule"
LABEL="end_mpath"
```

4. Setting raw device permissions

Create a custom udev file to set user, group and permissions of raw devices. For example, add the following lines to the custom `/etc/udev/rules.d/61-oracleraw.rules` file:

```
# Settings for OCR files
KERNEL=="raw[1-2]", OWNER="root", GROUP="oinstall", MODE="640"
# Settings for CSS voting disks
KERNEL=="raw[3-5]", OWNER="oracle", GROUP="oinstall", MODE="660"
```

To check the permissions run, for example,

```
# udevtest /class/raw/raw1 | grep mode
```

Issues with Oracle 10gR2 Clusterware Installation on RHEL5

There are several known issues related to mutipath devices on Red Hat Enterprise Linux 5. The Oracle 10gR2 Clusterware initializes OCR devices as part of running the *root.sh* script. Before running this script, be aware of the following known issues that will cause the Clusterware installation to fail:

Oracle Bug 4679769: Failed to format OCR disk using CLSFMT.

Oracle Note 414163.1: 10gR2 RAC Install issues on Oracle EL5 or Red Hat Enterprise Linux 5 or SLES10 (VIPCA Failures).

Before running *root.sh*, download and apply the patch for Bug 4679769. If the *root.sh* script execution caused the failure, erase the content of the OCR disks:

```
# dd if=/dev/zero of=/dev/raw/raw1 bs=1M count=25
# dd if=/dev/zero of=/dev/raw/raw2 bs=1M count=25
```

Before re-running *root.sh*, review Metalink Note 414163.1 to address known VIPCA issues.

Upon completion of the Oracle 10g R2 Clusterware installation, verify that it is working with the raw devices bound to multipath devices:

```
[oracle]$ ocrcheck
Status of Oracle Cluster Registry is as follows :
Version : 2
Total space (kbytes) : 262144
Used space (kbytes) : 1164
Available space (kbytes) : 260980
ID : 1749049955
Device/File Name : /dev/raw/raw1
Device/File integrity check succeeded
Device/File Name : /dev/raw/raw2
Device/File integrity check succeeded

Cluster registry integrity check succeeded

# crsctl query css votedisk
0.     0    /dev/raw/raw4
1.     0    /dev/raw/raw5
2.     0    /dev/raw/raw6

located 3 votedisk(s).

# crsctl check crs
CSS appears healthy
CRS appears healthy
EVM appears healthy
```

Configuring Storage for ASM

Using ASM without ASMLib driver

ASM simplifies storage management for Oracle RAC and standalone systems with a large number of disks. ASM is a user space process that can use Linux block devices directly. In Red Hat Enterprise Linux 5, it is recommended to use ASM with partitioned block devices. ASM processes should have read-write access to the ASM partitions. Usually the ASM partitions should belong to user “oracle” and group “dba” and have permissions 660.

You can use the `udev` utility to configure ASM device owner, group and permissions. A sample `/etc/udev/rules.d/39-oracle-multipath.rules` file is shown below.

```
# multipath wants the devmaps presented as meaningful device names
# so name them after their devmap name
SUBSYSTEM!="block", GOTO="end_mpath"
KERNEL!="dm-[0-9]*", ACTION=="add", PROGRAM=="/bin/bash -c '/sbin/lsmod | \
    /bin/grep ^dm_multipath'", RUN+="/sbin/multipath -v0 %M:%m"
KERNEL!="dm-[0-9]*", GOTO="end_mpath"
PROGRAM!="/sbin/mpath_wait %M %m", GOTO="end_mpath"
ACTION=="add", RUN+="/sbin/dmsetup ls --target multipath \
    --exec '/sbin/kpartx -a -p p' -j %M -m %m"
PROGRAM=="/sbin/dmsetup ls --target multipath --exec /bin/basename -j %M -m %m", \
     RESULT=="?*", NAME="%k", SYMLINK="mpath/%c", GOTO="oracle_rules"
PROGRAM!="/bin/bash -c '/sbin/dmsetup info -c --noheadings -j %M -m %m | \
    /bin/grep -q .*:.*:.*:.*:.*:.*:.*:part[0-9]*-mpath-'", GOTO="end_mpath"
PROGRAM=="/sbin/dmsetup ls --target linear --exec /bin/basename -j %M -m %m", \
    NAME="%k", RESULT=="?*", SYMLINK="mpath/%c", GOTO="oracle_rules"
LABEL="oracle_rules"
RESULT=="asm[0-9]*p1", OWNER="oracle", GROUP="dba", MODE="660", \
    RUN+="/bin/chown oracle:dba /dev/mapper/%c; /bin/chmod 660 /dev/mapper/%c"
OPTIONS="last_rule"
LABEL="end_mpath"
```

On a system with a single path SAN-attached storage you should be able to use udev rules identifying LUNs by WWID:

```
# Configure ASM devices
ACTION=="add", KERNEL=="sd*[!0-9]", PROGRAM=="/sbin/scsi_id -g -u -s %p", \
    RESULT=="360a98018486f6959684a453333527173", OWNER=”oracle”, GROUP=”dba”, \
    MODE=”660”, NAME=”asm1”
```

Using ASM with ASMLib driver

ASMLib driver is a kernel module providing the ASMLib API to the user space ASM instance for device access. The ASM instance can use this API instead of the interface provided by the Linux block or raw devices. ASMLib requires partitioned ASM devices. The label created on the partitioned device allows ASMLib daemon to scan and auto-detect ASM devices. It simplifies device management because auto-detection works around the issue with non-persistent device names. In addition, there is no need to configure permissions on the ASM devices because the ASM instance accesses them via the ASMLib kernel module.

ASMLib requires the installation of the following additional packages:

oracleasm-<KERNEL-VERSION-and RELEASE>
oracleasmlib
oracleasm-support

As you can see, the oracleasm package is specific for the Linux kernel version and release and it needs to be recompiled or downloaded from the Oracle repository for each kernel update. This is a trade off for the easier way to configure ASM devices. System administrators will need to have the corresponding ASMlib driver ready before applying any kernel upgrades.

On a system with multipathing, the same LUN can be visible as a regular SCSI block device (/dev/sd*) and as a multipath device (/dev/dm-*). ASMLib should be configured to use only multipath devices. This can be accomplished by setting the ASMLib scanning order in the `/etc/sysconfig/oracleasm` configuration file:

```
  ORACLEASM_SCANORDER="dm-*"
```

Configure ASMLib and create ASM volumes:

```
# service oracleasm configure
Configuring the Oracle ASM library driver.
...

# service oracleasm createdisk VOL0 /dev/mapper/asm0p1
# service oracleasm createdisk VOL1 /dev/mapper/asm1p1
...
```

You can list available ASM devices by running:

```
# service oracleasm listdisks
```

Re-scan ASM devices, if necessary:

```
# service oracleasm scandisks
```

Appendix A. Sample oracle10g-platform.spec file

```
Name: oracle10g-platform
Version: 1.0
Release: 1%{?dist}
Summary: Prepares system to be a platform for Oracle Database 10g server.
Group: System Environment/Oracle
License: GPLv2+
Source0: 39-oracle-multipath.rules
BuildRoot: %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)
Requires: binutils compat-db compat-gcc-34 compat-gcc-34-c++
Requires: compat-libstdc++-33 elfutils-libelf-devel
Requires: gcc gcc-c++ gdb gdbm glibc glibc-common glibc-devel
Requires: ksh libXp libXtst libaio libaio-devel libgcc libgnome
Requires: libstdc++ libstdc++-devel make setarch sysstat
Requires: unixODBC unixODBC-devel util-linux xorg-x11-xinit compat-libstdc++-296
%ifarch x86_64
Requires: /usr/lib/libc.so
Requires: /usr/lib64/libc.so
Requires: libaio.so.1 libc.so.6 libgcc_s.so.1 libstdc++.so.5 libssl.so.6
Requires: libXp.so.6 libXtst.so.6 libstdc++-libc6.2-2.so.3 libgdbm.so.2
%endif

%description
This package performs system reconfiguration to prepare system for Oracle 10g
installation. It has dependencies from all rpms required for Oracle.

%prep
#%setup

%build

%install
rm -rf $RPM_BUILD_ROOT

%{__mkdir_p} ${RPM_BUILD_ROOT}%{_sysconfdir}/udev/rules.d
install -m 644 %{SOURCE0} ${RPM_BUILD_ROOT}%{_sysconfdir}/udev/rules.d

%{__mkdir_p} ${RPM_BUILD_ROOT}/usr/lib
ln -s /usr/lib/libgdbm.so.2 ${RPM_BUILD_ROOT}/usr/lib/libdb.so.2
ln -s /usr/lib/libgdbm.so.2.0.0 ${RPM_BUILD_ROOT}/usr/lib/libdb.so.2.0.0

%ifarch x86_64
%{__mkdir_p} ${RPM_BUILD_ROOT}/usr/lib64
ln -s /usr/lib64/libgdbm.so.2 ${RPM_BUILD_ROOT}/usr/lib64/libdb.so.2
ln -s /usr/lib64/libgdbm.so.2.0.0 ${RPM_BUILD_ROOT}/usr/lib64/libdb.so.2.0.0
%endif

%clean
rm -rf $RPM_BUILD_ROOT

%files
%defattr(-,root,root)
%{_sysconfdir}/udev/rules.d/*
/usr/lib/libdb.so.2
/usr/lib/libdb.so.2.0.0
```

```
%ifarch x86_64
/usr/lib64/libdb.so.2
/usr/lib64/libdb.so.2.0.0
%endif

%post

# Checking Oracle user real uid and gid
oracle_uid=$(id -ru oracle 2> /dev/null)
oracle_gid=$(id -rg oracle 2> /dev/null)

# Checking the available memory
memtotal=$(awk '/MemTotal/ { print $2 }' /proc/meminfo)

# Checking the memory page size
pagesize=$(getconf PAGE_SIZE)

# Checking the huge page size
hugepagesize=$(awk '/Hugepagesize/ { print $2 }' /proc/meminfo)

# Approximation of the total system memory
if [ $memtotal -gt 66060288 ]; then     # > 63GB
    memtotal=67108864
elif [ $memtotal -gt 32505856 ]; then   # > 31GB
    memtotal=33554432
elif [ $memtotal -gt 15728640 ]; then   # > 15GB
    memtotal=16777216
elif [ $memtotal -gt 7340032 ]; then    # > 7GB
    memtotal=8388608
elif [ $memtotal -gt 3145728 ]; then    # > 3GB
    memtotal=4194304
elif [ $memtotal -gt 2096852 ]; then    # about 2GB
    memtotal=2097152
fi

# Shared memory parameters
shmmax=$(($memtotal * 1024 / 2))
shmmni=4096
shmall=$(($memtotal * 1024 / $pagesize))

# Semaphore parameters used in the Oracle validated configurations
semmsl=250
semmni=142
semmns=32000
semopm=100

# Allocate huge pages if the system has 4GB+
if [ $memtotal -gt 3145728 ]; then
    nr_hugepages=$(($memtotal / 2 / $hugepagesize))
else
    nr_hugepages=0
fi

# Set memlock value corresponding to the size of memory allocated for huge pages
if [ $nr_hugepages -gt 0 ]; then
    memlock=$(($nr_hugepages * $hugepagesize))
else
```

```
    memlock=32
fi


cat >> /etc/sysctl.conf << __EOF__
## Added by oracle10g-platform
fs.aio-max-nr = 3145728

kernel.shmmax = ${shmmax}
kernel.shmmni = ${shmmni}
kernel.shmall = ${shmall}
kernel.sem = ${semmsl} ${semmns} ${semopm} ${semmni}

net.core.rmem_default = 262144
net.core.wmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_max = 262144

net.ipv4.tcp_rmem = 4096 262144 4194304
net.ipv4.tcp_wmem = 4096 262144 262144

net.ipv4.ip_local_port_range = 1024 65000

vm.swappiness = 0
vm.dirty_background_ratio = 3
vm.dirty_ratio = 15
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100
vm.hugetlb_shm_group = ${oracle_gid:=0}
vm.nr_hugepages = ${hugepagesize:=2048}
## End of additions from oracle10g-platform
__EOF__


cat > /etc/security/limits.conf << __EOF__
# /etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#<domain>        <type>  <item>  <value>
#
#Where:
#<domain> can be:
#        - an user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax,
#                 for maxlogin limit
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
```

```
#        - nofile - max number of open files
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit
#        - maxlogins - max number of logins for this user
#        - maxsyslogins - max number of logins on the system
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#        - sigpending - max number of pending signals
#        - msgqueue - max memory used by POSIX message queues (bytes)
#        - nice - max nice priority allowed to raise to
#        - rtprio - max realtime priority
#
#<domain>      <type>  <item>         <value>
#

#*               soft    core            0
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#@student        -       maxlogins       4

## Added by oracle10g-platform
oracle          soft    nofile          131072
oracle          hard    nofile          131072

oracle          soft    nproc           131072
oracle          hard    nproc           131072

oracle          soft    memlock         4194304
oracle          hard    memlock         4194304
## End of additions from oracle10g-platform

*               soft    nofile          8196
*               hard    nofile          32768
# End of file
__EOF__

/sbin/sysctl -p > /dev/null

%changelog
* Thu Oct  9 2008 Aleksandr Brezhnev <brezhnev@redhat.com>
- Initial build
```

Appendix B. Sample 39-oracle-multipath.conf

```
# multipath wants the devmaps presented as meaninglful device names
# so name them after their devmap name
SUBSYSTEM!="block", GOTO="end_mpath"
KERNEL!="dm-[0-9]*", ACTION=="add", PROGRAM=="/bin/bash -c '/sbin/lsmod | /bin/grep
^dm_multipath'", RUN+="/sbin/multipath -v0 %M:%m"
KERNEL!="dm-[0-9]*", GOTO="end_mpath"
PROGRAM!="/sbin/mpath_wait %M %m", GOTO="end_mpath"
ACTION=="add", RUN+="/sbin/dmsetup ls --target multipath --exec '/sbin/kpartx -a -p p' -j
%M -m %m"
PROGRAM=="/sbin/dmsetup ls --target multipath --exec /bin/basename -j %M -m %m",
RESULT=="?*", NAME="%k", SYMLINK="mpath/%c", GOTO="oracle_rules"
PROGRAM!="/bin/bash -c '/sbin/dmsetup info -c --noheadings -j %M -m %m | /bin/grep -q
.*:.*:.*:.*:.*:.*:.*:part[0-9]*-mpath-'", GOTO="end_mpath"
PROGRAM=="/sbin/dmsetup ls --target linear --exec /bin/basename -j %M -m %m", NAME="%k",
RESULT=="?*", SYMLINK="mpath/%c", GOTO="oracle_rules"
LABEL="oracle_rules"
RESULT=="asm[0-9]*p1", OWNER="12961", GROUP="oracle", MODE="dba", RUN+="/bin/chown
oracle:dba /dev/mapper/%c; /bin/chmod 660 /dev/mapper/%c"
OPTIONS="last_rule"
LABEL="end_mpath"
```

Appendix C. Sample oracle.spec file

```
# Custom Oracle base directory
%define oracle_base             /u01/app/oracle/product

# Custom Oracle home name prefix
%define oracle_name_prefix      RHT

# Oracle version and CPU release
%define oracle_version          10.2.0.4
%define oracle_cpu              2008.10

%define __os_install_post       %{nil}

Name: oracle
Version: %{oracle_version}.%{oracle_cpu}
Release: 3%{?dist}
Summary: Oracle Database Server %{oracle_version} with CPU %{oracle_cpu}
Group: Applications/Database
License: Commercial
Source0: %{name}-%{oracle_version}-%{oracle_cpu}.tar.gz
NoSource: 0
AutoReqProv: 0
BuildRoot: %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)
Requires: oracle10g-platform

%description
Oracle Database Server %{oracle_version} with CPU %{oracle_cpu}.

%prep
#%setup
```

```
%build

%install
rm -rf %{buildroot}

%{__mkdir_p} %{buildroot}%{oracle_base}
cd %{buildroot}%{oracle_base}
%{__tar} zxvf %{SOURCE0}

oracle_home_version=$(echo -n %{oracle_version} | sed 's/\.//g')
cat > %{buildroot}%{oracle_base}/%{oracle_version}-%{oracle_cpu}/reg_inventory <<EOF
#!/bin/bash
#
ORACLE_HOME=%{oracle_base}/%{oracle_version}-%{oracle_cpu}
ORACLE_HOME_NAME=%{oracle_name_prefix}${oracle_home_version}

cd \$ORACLE_HOME/clone/bin
perl clone.pl ORACLE_HOME="\$ORACLE_HOME" ORACLE_HOME_NAME="\$ORACLE_HOME_NAME"
EOF

chmod 554 %{buildroot}%{oracle_base}/%{oracle_version}-%{oracle_cpu}/reg_inventory

%clean
rm -rf %{buildroot}

%files
%defattr(-,oracle,dba)
%verify(not size md5 mtime) %{oracle_base}/%{oracle_version}-%{oracle_cpu}

%post
su -c %{oracle_base}/%{oracle_version}-%{oracle_cpu}/reg_inventory oracle
%{oracle_base}/%{oracle_version}-%{oracle_cpu}/root.sh

%changelog
* Fri Oct 31 2008 Aleksandr Brezhnev <brezhnev@redhat.com> - 10.2.0.4-3
- Oracle 10.2.0.4 with October 2008 CPU

* Mon Oct 27 2008 Aleksandr Brezhnev <brezhnev@redhat.com> - 10.2.0.3-2
- Installation to a separate Oracle home for different CPU

* Thu Oct  9 2008 Aleksandr Brezhnev <brezhnev@redhat.com> - 10.2.0.3-1
- Initial build for 10.2.0.3 with CPU July 2008
```

References

Tuning and Optimizing Red Hat Enterprise Linux for Oracle 9i and 10g Databases
http://www.redhat.com/docs/manuals/enterprise/RHELTuningandOptimizationforOracleV11.pdf

Oracle Database 10g Release 2 (10.2) Documentation
http://www.oracle.com/pls/db102/homepage

Understanding the Linux Kernel
http://www.bookpool.com/sm/0596005652

The Linux Page Cache and pdflush: Theory of Operation and Tuning for Write-Heavy Loads
http://www.westnet.com/~gsmith/content/linux-pdflush.htm

Online Storage Reconfiguration Guide
http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.2/html/Online_Storage_Reconfiguration_Guide/

Optimizing Linux I/O http://www.oracle.com/technology/deploy/availability/pdf/S939_SusairajLee.ppt.pdf

Oracle MetaLink Notes:
169706.1: Installation and Configuration Requirements Quick Reference
249213.1: Performance problems with Failover when TCP Network goes down
294430.1: CSS Timeout Computation in Oracle Clusterware
301830.1: Upon startup of Linux database get ORA-27102
309815.1: Configuring Oracle ASMLib on Multipath Disks
341782.1: Linux Quick Reference
357472.1: Configuring device-mapperfor CRS/ASM
376183.1: Defining “default RPMs” installation of the RHEL OS
419646.1: Requirements For Installing Oracle 10gR2 on RHEL/OEL 5 (x86)
421308.1: Requirements For Installing Oracle 10gR2 on RHEL/OEL 5 (x86_64)
465001.1: Configuring raw devices (singlepath) forOracle 10g Release 2 (10.2.0) on RHEL5 / OEL5
564580.1: Configuring raw devices (multipath) forOracle 10g Release 2 (10.2.0) on RHEL5 / OEL5
602952.1: How to Setup ASM & ASMLIB On Native Linux multipath Mapper disks?
605828.1: Configuring non-raw devices Multipath Clusterware for Oracle 11g (11.1.0) on RHEL5 / OEL5
728346.1: Linux OS Installation with Reduced Set of Packages for Running Oracle Database Server