Oracle9iR2 on Linux: Performance, Reliability and Manageability Enhancements on Red Hat Linux Advanced Server 2.1

An Oracle Technical White Paper
June 2002

Table of Contents

Oracle9iR2 on Linux: Performance, Reliability and Manageability Enhancements on Red Hat Linux Advanced Server 2.1

EXECUTIVE OVERVIEW

Linux is on the threshold of large-scale deployment for IT applications for small, medium and large size companies around the world. As customers migrate business critical applications to Linux, there are increasing demands for perfomance, reliability and manageability on the operating system. Oracle - the largest provider of enterprise software having years of experience deploying on multiple operating system platforms - is uniquely positioned to influence the evolution of Linux from a workstation environment to a high-end server platform.

Oracle has established a strong relationship with Red Hat, working closely with them to drive key enhancements into the Linux kernel, with the goal of delivering an operating system that supports enterprise-class functionality and mission-critical applications. The result is a new distribution from Red Hat called “Advanced Server” which is tuned for Oracle’s market leading database and application server. This paper describes the most important new features available to customers deploying Oracle9iR2 on Red Hat Linux Advanced Server 2.1.

ORACLE ON LINUX ARCHITECTURE

Linux provides programming interfaces and functionality that is similar to most UNIX platforms. Hence the Oracle9iR2 architecture on Linux is very similar to Oracle’s architecture on UNIX-based operating systems including Solaris, HP-UX, and Tru64. This guarantees that Oracle products on Linux inherit the performance and stability developed over many years on traditional UNIX platforms.

Process-based Model

Oracle9iR2 on Linux is a process-based architecture similar to UNIX platforms, compared to the thread-based architecture on Windows. On Linux, Oracle uses processes to implement background tasks such as database writers, log writer, process monitors, etc., as well as foreground tasks such as handling incoming client connections. On Windows, all these tasks are implemented as threads within a single process. The thread-based model has limitations on 32-bit Intel Pentium-based systems because all the threads share the same 3GB address space for SGA, PGA, and other memory. For example, when large sort areas are in use, available memory can be an issue since all foreground threads get their memory from the same 3GB pool. In a process model, each foreground process gets its own address space and therefore has more memory available for use. It should be noted that this is not a limitation on 64-bit systems since each process has at least 8TB of user address space.

Filesystem Support

Oracle9iR2 is certified with the standard Linux filesystem "ext2" on Red Hat Linux Advanced Server 2.1. In addition, Oracle is also certified to run on "raw" (unformatted) disk partitions. Raw files have the advantage of improved I/O performance because there is no filesystem overhead. However, raw files are harder to manage, hence they are only used for high-end installations or when using RAC which requires raw files.

64-bit File I/O

Linux supports 64-bit file I/O even on 32-bit platforms like Intel Pentium-based servers. Oracle9iR2 supports 64-bit file offsets internally, hence there are no 2GB or 4GB limitations on data, log and control files. The limits on number of files per database (64K), number of blocks per file (4 million), and maximum block size (16KB) are common to other Oracle platforms. Based on these limits, the maximum size for a database file is 64GB, and the maximum database size with 16KB blocks is 4 petabytes.

PERFORMANCE ENHANCEMENTS

Oracle has conducted extensive performance-related testing in order to identify improvements that could be made to the Linux kernel for database and application server performance. This section describes features available for Oracle9iR2 on Red Hat Linux Advanced Server 2.1 that result in a significant improvement in performance compared to previous releases of Oracle on Linux. Although this paper discusses enhancements in the context of Intel's 32-bit (IA-32) platform, most of the information is applicable to the upcoming production release of Oracle9iR2 and Red Hat Linux Advanced Server on Intel's 64-bit (IA-64) platform based on the Itanium 2 processor.

I/O Subsystem

The I/O throughput of the system under Oracle database workloads is dramatically higher with Advanced Server. The most important enhancements in the I/O subsystem are asynchronous i/o, elimination of multiple copies to memory buffers while writing to disk, reducing contention for kernel locks, and numerous IO driver enhancements.

Asynchronous I/O

One of the most important enhancements is asynchronous I/O (or non-blocking I/O) in the kernel. Before the introduction of asynchronous I/O in Advanced Server, processes submitted disk I/O requests sequentially. Each I/O request would cause the calling process to sleep until the request was completed. Asynchronous I/O allows a process to submit an I/O request without waiting for it to complete. The implementation also allows Oracle processes to issue multiple I/O requests to disk with a single system call, rather than a large number of single I/O requests. This improves performance in two ways. First, because a process can queue multiple requests for the kernel to handle, the kernel can optimize disk activity by reordering requests or combining individual requests that are adjacent on disk into fewer, larger requests. Secondly, because the system does not put the process to sleep while the hardware processes the request, the process is able to perform other tasks until the I/O is complete.

This paper outlines steps to enable various features for Oracle9iR2 on Red Hat Linux Advanced Server 2.1, with the goal of promoting increased understanding of the technical concepts. Readers are urged to refer to relevant product documentation (Release Notes, Administrator's Reference, Installation Guide, etc.), Oracle Support, and Redhat Support for the most current and accurate information.

By default, Oracle9iR2 is shipped with asynchronous I/O support disabled. This is necessary to accommodate other Linux distributions that do not support this feature. To enable asynchronous I/O for Oracle9iR2 on Red Hat Linux Advanced Server 2.1, customers will need to follow these steps as outlined in the product documentation:

1) cd to $ORACLE_HOME/rdbms/lib
    a) make -f ins_rdbms.mk async_on
    b) make -f ins_rdbms.mk ioracle
2) If asynchronous I/O needs to be disabled for some reason, cd to $ORACLE_HOME/rdbms/lib
    a) make -f ins_rdbms.mk async_off
    b) make -f ins_rdbms.mk ioracle
3) Parameter settings in init.ora for raw devices:
    a) set 'disk_asynch_io=true' (default value is true)
4) Parameter settings in init.ora for filesystem files:
    a) Make sure that all Oracle datafiles reside on filesystems that support asynchronous I/O. (For example, ext2)
    b) set 'disk_asynch_io=true' (default value is true)
    c) set 'filesystemio_options=asynch'

Eliminate copy to bounce buffer

On IA-32 systems, the first gigabyte (GB) of physical memory is known as “low memory”. Physical memory above 1GB is refered to as “high memory”. Previous versions of the Linux kernel required data buffers passed to storage devices to be located in the low memory region of physical RAM, even though applications could use both high and low memory. I/O requests from data buffers located in low memory resulted in a direct memory access operation (no copy). However, when the application passed a data buffer in high memory to the kernel as part of an I/O request, the kernel would be forced to allocate a temporary data buffer in low memory, and copy the data to/from the application's buffer in high memory. This additional copying of data, referred to as "bounce buffering", can significantly degrade performance of I/O intensive database applications. This is because the large number of bounce buffers that are allocated consume a lot of memory, and because the bounce buffer copying places additional load on the system memory bus. Red Hat Linux Advanced Server 2.1 greatly reduces, and in many cases, completely eliminates the need to perform bounce buffering.

Reduce contention for io_request_lock

The Linux kernel uses spin locks to maintain the integrity of kernel data structures that are accessed concurrently by multiple processes. Earlier versions of the kernel used a single spin lock for the entire block device subsystem. This forced all processes performing I/O to compete for this single resource even when performing I/O to unrelated devices, creating an artificial bottleneck that resulted in reduced overall system I/O throughput. Red Hat Linux Advanced Server 2.1 has implemented a new, fine-grained locking scheme for the block device subsystem, which includes a separate lock for each individual block device. The result is a much higher I/O throughput for SMP systems with multiple I/O controllers under heavy database load.

I/O driver optimizations

There were numerous enhancements in the driver code for I/O cards used in server hardware from partners including Dell, HP/Compaq, EMC, etc. Some of these enhancements eliminated bottlenecks identified in our performance labs, while other enhancements enabled these drivers to exploit the bounce buffer and io_request_lock changes mentioned in previous paragraphs.

Virtual Memory Subsystem

For more details on Oracle's interaction with the VM subsystem on Linux, please refer to the Oracle technical whitepaper titled "Linux Virtual Memory in Red Hat Linux Advanced Server 2.1 and Oracle's Memory Usage Characteristics".

The Virtual Memory (VM) subsystem is critical for database performance. The Oracle database architecture depends heavily on the Shared Global Area (SGA), which is an area of memory shared by all background and foreground processes. The size of the SGA, which holds the database block buffer cache, is a major tuning parameter that impacts Oracle performance. A larger SGA allows more data to be cached in memory, leading to significantly improved performance for many database workloads. The SGA size, as well as other parameters including OS page size, are determined by the VM subsystem.

While the VM subsystem enhancements described in this section can provide significant gains in performance and stability for various workloads, it is important to understand the tradeoff between benefits and limitations. Different workloads have different database cache usage characteristics, and there is no "formula" to determine the best set of enhancements and tuning parameters in any given situation. It may be necessary to gather statistics on the actual workload in order to determine the optimal configuration for a system.

Larger SGA for systems with up to 4GB of RAM

Red Hat Linux Advanced Server 2.1 is shipped with two kernels. The default "smp" kernel supports up to 4GB RAM and uses two-level page tables. The "enterprise" kernel supports up to 64GB RAM and requires three-level page tables to access the larger RAM. On systems with 4GB or less of RAM, the "smp" kernel is recommended due to the lower memory and CPU overhead of using two-level pagetables.

On systems with upto 4GB of RAM, most Linux distributions allow Oracle to use about 1.7GB of address space for its SGA. Red Hat Linux Advanced Server 2.1 is the first Linux distribution to provide an adjustable parameter in the /proc filesystem to allow more usable address space in processes. To increase this size, Oracle needs to be relinked with a lower SGA base and Linux needs to have the mapped base lowered for processes running Oracle. The following two steps are required to achieve this:

First, the SGA base address that Oracle uses must be lowered by relinking Oracle. Currently, Oracle ships with this base address set at 0x50000000 so that it is compatible with the defaults set by most distributions of Linux. Lowering this address allows Oracle to use more of the address space in the process, but it is important to note that the modified and relinked Oracle binary will not work unless a corresponding modification is also made to Advanced Server 2.1. Follow these steps to lower the SGA base address within Oracle:

- Shutdown all instances of Oracle
- cd $ORACLE_HOME/lib
- cp -a libserver9.a libserver9.a.org (to make a backup copy)
- cd $ORACLE_HOME/bin
- cp -a oracle oracle.org (to make a backup copy)
- cd $ORACLE_HOME/rdbms/lib
- genksms -s 0x15000000 >ksms.s (lower SGA base to 0x15000000)
- make -f ins_rdbms.mk ksms.o (compile in new SGA base address)

- make -f ins_rdbms.mk ioracle (relink)

Next, the Linux kernel's mapped base needs to be lowered below Oracle's new SGA base. Advanced Server 2.1 has a parameter in /proc that lowers the kernel's mapped base for each process. This parameter is not a system-wide parameter. It is a per-process parameter, but it is inherited by the child processes. This parameter can only be modified by root. The following steps document how to lower the mapped base for one bash terminal session. Once this session has been modified with the lower mapped base, this session (window) will need to be used for all Oracle commands so that Oracle processes use the inherited (lower) mapped base:

- Shutdown the instance of Oracle.
- Open a terminal session (Oracle session).
- Open a second terminal session and su to root (root session).
- Find out the process id for the Oracle session. For example, do "echo $$" in the Oracle session.
- Now lower the mapped base for the Oracle session to 0x10000000. From the root session, echo 268435456 >/proc/<pid>/mapped_base, where <pid> is the process id determined in the earlier step.
- Increase the value of shmmax so that Oracle will allocate the SGA in one segment. From the root session, echo 3000000000 >/proc/sys/kernel/shmmax
- From the Oracle terminal session, startup the Oracle instance. The SGA now begins at a lower address, so more of the address space can be used by Oracle.

Now you can increase the init.ora values of db_cache_size or db_block_buffers to increase the size of the database buffer cache. Detailed instructions, limitations, and scripts to automate some of these steps, can be found at the Oracle Support website.

Very Large Memory (VLM) on systems with up to 64GB of RAM

As mentioned in the sidebar above, the "enterprise" kernel used to access more than 4GB RAM requires three-level pagetables, which results in higher memory and CPU utilization. However, for many workloads, the access to additional RAM can provide significantly higher scalability and performance on systems with more than 4GB RAM. It is important to take into account both the benefits and limitations of using this option.

The Advanced Server 2.1 kernel allows Oracle to allocate and use more than 4GB of memory for the database buffer cache on a 32-bit Intel platform. This feature is also called VLM (Very Large Memory) in Oracle documents. With Oracle9iR2 on Red Hat Linux Advanced Server 2.1, the SGA can be increased to a theoretical limit of about 62GB depending on the available RAM memory on the system. Current hardware limitations and practical considerations further limit the actual size of SGA that can be configured, but it is still several times larger than the size of the SGA without VLM. Running in VLM mode requires some setup changes to Linux and imposes some limitations on what features and init.ora parameters can be used.

Linux OS modification: The administrator as 'root' must mount an in-memory filesystem on /dev/shm equal to or larger than the amount of memory that will be used for the database buffer cache. If an in-memory filesystem is already mounted, it can be used as long as it is large enough. If there is no such filesystem mounted, a mount command similar to the one below will work:

```
mount -t shm shmfs -o nr_blocks=2097152 /dev/shm
```

This will create a shmfs filesystem on /dev/shm of size 8GB. Each nr_block is 4096 bytes. When Oracle is started with the extended buffer cache feature enabled, a file is created in /dev/shm that corresponds to Oracle's buffer cache.

init.ora changes: The extended buffer cache feature is enabled by setting 'use_indirect_data_buffers=true' in the init.ora file. This will allow a larger buffer cache to be specified. However, this feature is not compatible with the newer dynamic cache parameters. So the following parameters cannot be used when the extended cache feature is enabled:

- db_cache_size
- db_2k_cache_size
- db_4k_cache_size
- db_8k_cache_size
- db_16k_cache_size
- db_32k_cache_size

If the extended cache feature is used, the database cache size must be specified using db_block_buffers. The extended cache feature and other limitations are described in more detail in Oracle9iR2 documentation.

If you are running in VLM mode (init.ora parameter 'use_indirect_data_buffers=true') and already have a large buffer cache, you can lower the Oracle SGA base and mapped base for Linux (as described in the previous section) to increase the indirect buffer window size. Doing this may slightly increase performance under certain conditions because a larger indirect window reduces the overhead of mapping an indirect buffer into Oracle's address space. The default indirect window size is 512MB. To increase the size, set the environment variable VLM_WINDOW_SIZE to the window size in bytes before starting up the Oracle instance. For example, export VLM_WINDOW_SIZE=1073741824 to set the indirect window size to 1GB. Any value set should be a multiple of 64KB.

Notes:

- Increasing the buffer cache size (or the indirect window size) too high can cause Oracle "attach errors" while starting up.
- If you try to use an Oracle binary that has a lower SGA base but did not lower the /proc/<pid>/mapped_base value, you will experience unpredictable results ranging from ORA-3113 errors, attach errors, etc. while starting up.
- If you don't increase the shmmax value, you could get attach errors while starting up.

Large Pages

IA-32 can support two page sizes simultaneously - 4KB and "large page". In normal mode (Linux 4GB kernel) with Page Size Extension (PSE) turned on, "large page" = 4MB and the cpu uses a 2-level page table. In Physical Address Extension (PAE) mode (Linux 64GB kernel) with PSE turned on, "large page" = 2MB and the cpu uses a 3-level page table.

The regular page frame size in the Linux kernel is 4KB. Advanced Server 2.1 allows Oracle9iR2 to use large pages (2MB or 4MB) for allocating the SGA. The use of large pages for the SGA can improve performance for several reasons:

- IA-32 processors deal with memory in 4KB pages using a 2- or 3-level page table to map virtual addresses to physical addresses. A TLB (Translation Look-aside Buffer) entry is internally used to cache this address translation so that the cpu does not have to walk the page tables on every memory access to find the physical address. There are very few TLB entries on a processor, so applications like Oracle that access large amounts of memory can have a high TLB miss rate. With large page support, the processor deals with memory in 4MB (or 2MB, depending on the mode) pages. In this case, a PTE (Page Table Entry) entry in the TLB will cover 4MB instead of 4KB (1024 times more memory). The TLB miss rate decreases significantly because touching address A, A+4KB, A+8KB, … upto A+4MB no longer cause a TLB miss.
- Page table size is greatly reduced which leads to improved memory utilization. This is because 1024 4KB PTEs are now handled by one 4MB PTE.
- Better performance is also achieved because the big pages are not swapped out which means the entire db_block_buffers are locked in physical memory. The system performance increases as a result of the kernel not having to 'think' about swapping out these pages. Since swap space is not pre-allocated for these pages, there is more swap area available and less page cache complexity.

Instructions to enable large pages:

- In the kernel boot options, add "bigpages=<size>MB" to the boot loader file (e.g. /etc/lilo.conf), where size is a value in MB appropriate for your system.
- Set the /proc/sys/kernel/shm-use-bigpages file to contain the value 2. The other possible values are 0 for no bigpages and 1 for bigpages using sysV shared memory (as opposed to shmfs).

Maximum bigpages value for a system can be obtained by following formula: Maximum value of bigpages = HighTotal / 1024 * 0.8 MB where HighTotal is the value in Kbytes obtained from /proc/meminfo. The assumption is that 20% of the memory is reserved for kernel bookkeeping. For example, a machine with 8GB RAM has a HighTotal of 7208944 KB, and therefore the maximum bigpages value is about 5631MB. If the value for bigpages is set very high, the memory available for user connections would be low.

There is a trade-off between the number of users and the value of bigpages. If we can estimate the maximun number of user connections and the amount of memory consumed by each user connection, then the exact value of bigpages can be calculated as follows: bigpages = (HighTotal - Memory required by maximum user connections in KB) / 1024 *0.8 MB

Process Scheduler

The Process Scheduler is responsible for controlling process access to the CPU. Advanced Server 2.1 uses a new scheduler that overcomes many disadvantages of the previous design. The following is a summarized list of improvements based on a description by Ingo Molnar on the Linux kernel mailing list dated Jan 03, 2002:

- O(1) fixed time scheduling algorithm.
- The new scheduler eliminates the big runqueue lock, and implements per-CPU runqueues and locks. This allows scheduling of tasks on separate CPUs in parallel without any interlocking, leading to improved SMP scalability.
- The old scheduler caused heavy migration of processes between CPUs under heavy load. The new scheduler implements CPU affinity by distributing timeslices on a per-CPU basis, without global synchronization or recalculation.
- The scheduler implements cache-affinity by tracking the cache-footprint of threads, and not scheduling them away if they are still cache-hot for a given amount of time. This improves the cache-hit rate for processes.

RELIABILITY ENHANCEMENTS

To improve reliability and stability, Oracle9iR2 on Advanced Server 2.1 was tested across a large spectrum of workloads, with the goal of improving resiliency to handle resource shortages. A robust enterprise-class operating system must handle the tremendous demand of peak usage times without failure. Furthermore, performance on enterprise class systems is expected to degrade gracefully as user loads exceed system capacity. Oracle and Red Hat collaborated to identify "break points" across the entire software stack (database and kernel), and have added a number of enhancements to improve stability under high user load. The enhancements were in the areas of IO, memory management, networking and process scheduling. Both Oracle9iR2 and the Advanced Server kernel have been hardened to support a significantly larger number of database users on in single-node as well as RAC multi-node configurations. For example, the performance of workloads such as Oracle Applications 11i will degrade gracefully as the number of concurrent users exceeds system capacity. Some of the important enhancements to improve stability and enable graceful degradation are described in this section.

High memory PTE patch

As described in a previous section, the first gigabyte (GB) of physical memory on IA-32 systems is known as “low memory”. Physical memory above 1GB is refered to as “high memory”. In older kernels, Linux could allocate Page Table Entries (PTEs) only in low memory, which has a limitation of 1GB. For applications like Oracle9iR2, which use a large amount of memory and lots of processes, the total size of the PTEs is high. As more users connect to the database, the kernel would run out of space for PTEs, and even if free memory and swap space was available the system would hang or crash.

The high memory PTE patch allows the VM to use the “high memory” pool for allocating PTEs. As an increasing number of users connect to the database and spawn additional processes, the area to store PTEs overflows to high memory, allowing the system to support 3 to 5 times the number of users than allowed by previous kernels! Morever, as the number of users is increased beyond system capacity, there is a gradual degradation in database response time, and eventually no new database user connections are accepted. The Advanced Server kernel does not crash or hang as described earlier for previous kernels.

Improved Oracle9iR2 RAC scalability

For Oracle9i and earlier releases, the RAC Cluster Manager communicated between the nodes of a cluster using the TCP/IP protocol. The number of TCP connections for each Cluster Manager process increased in proportion to the product of the user count and number of nodes. On clusters with a large number of nodes (e.g. eight nodes or more), the limit on the number of TCP/IP connections per process limited the total number of concurrent database user connections across the cluster.

In Oracle9iR2, the Cluster Manager communicates between the nodes of a cluster using the UDP protocol. Since UDP is a connection-less protocol, the limit imposed by the maximum number of TCP/IP connections per process does not apply in this case. This allows Linux RAC clusters with a large number of nodes to scale to a much higher number of concurrent users than previous releases of Oracle on Linux.

MANAGEABILITY ENHANCEMENTS

Oracle and Red Hat are working aggressively to create manageability enhancements that would promote ease-of-use and improved supportability for Oracle products on Linux. While much of this work is ongoing, this section describes some of the new enhancements available at the time of release of Oracle9iR2 on Red Hat Linux Advanced Server 2.1.

lsraid utility

One example is the "lsraid" utility created by Oracle for software RAID storage management. The following is a description of lsraid from the “man page” in Advanced Server 2.1.

“lsraid is a program for querying Linux md devices. It can describe the composite device and the block devices that belong to it. It can also provide a description of the md device suitable for including in the /etc/raidtab configuration file.

lsraid also has the ability to operate on online and offline devices. It can read an online device via the kernel interface and provide information about it. When a device is offline, lsraid can look at any of the block devices that are a part of the md device and read the persistent md superblock for information.”

Network Console and Crash Dump Facility

Advanced Server 2.1 is bundled with Red Hat’s first “crash dump” facility. The network console functionality provides the ability to log all kernel messages, including Linux crash signature messages, to a centralized server via the network. The tool provides centralized and consistent views of system and kernel logs, which enables quicker resolution of problems at customer sites. For details, please refer to the whitepaper titled “Red Hat, Inc.’s Network Console and Crash Dump Facility” on the Red Hat website at http://www.redhat.com/support/wpapers/redhat/netdump/index.html

Cluster Management

The Oracle Cluster Manager is a component of Oracle’s RAC architecture. The Cluster Manager implements cluster membership and node monitoring services. A crucial part of the Cluster Manager’s functionality is fencing off a “failed” node from the cluster, which is implemented via a hardware node reset by the

Linux watchdog device. Oracle has contributed several enhancments to the watchdog code for improved management of RAC on Linux.

For example, the watchdog timer margin is defined by the soft_margin parameter specified during watchdog module loading. However, there was no mechanism in previous kernels to obtain the value of soft_margin from the kernel at runtime. This required the user to explicitly set an additonal parameter for the Cluster Manager that mirrored the soft_margin. Oracle has contributed code in Advanced Server 2.1 that allows the Cluster Manager to get the soft_margin value from the kernel via an ioctl call at runtime.

The overall I/O fencing solution in Oracle9iR2 on Advanced Server 2.1 is superior to the I/O fencing implemented on previous versions of Oracle on Linux. For example, the default configuration will not cause a hardware reset after issuing a "shutdown abort" command to a database instance on that node.

64-BIT SUPPORT

With the introduction of 64-bit Itanium-based systems, Linux has achieved a dramatically higher level of performance and scalability on the Intel platform. Oracle has released developer versions of its database software on IA-64 since Oracle8i Release 8.1.7. The production release of Oracle9iR2 on IA-64 will coincide with the availability of systems based on Intel's Itanium 2 processor. Oracle on IA-64 will be able to scale to a higher number of users, support larger SGA allocation, and provide much higher CPU and I/O capacity than the database on 32-bit Linux.

Oracle is working closely with Red Hat, Intel, and OEM partners to take advantage of all the improved features in Itanium 2-based systems. In addition to all the enhancements included in the 32-bit version, Red Hat Linux Advanced Server 2.1 on IA-64 will include numerous enhancements that are specific to Itanium 2-based systems.

CONCLUSION

Oracle has always been committed to Linux. To further enhance the Oracle platform on Linux, Oracle has worked closely with Red Hat to define and develop enhancements that are required for high-end performance, reliability and manageability. With the new features available in Red Hat Linux Advanced Server 2.1, Oracle on Linux offers enterprise-class performance and stability while maintaining the cost advantages of deploying on Linux and commodity hardware platforms.

Oracle9iR2 on Linux: Performance, Reliability and Manageability Enhancements on Red Hat Linux Advanced Server 2.1
June 2002

Author: Jamshed N Patel
Contributing Authors: Van Okamura, Lance Larsh

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com