



CHAPTER 15

Configuring the
Hardware for Linux
and Oracle



uning the hardware is as important as tuning the operating system and tuning the Oracle instance. But what is hardware tuning? Hardware tuning involves proper installation and configuration of the server hardware and storage. Purchasing sufficient storage capacity and configuring it properly is also crucial to the proper performance and, in some cases, such as with Oracle RAC, the functionality of your Oracle system.

Hardware Tuning Fundamentals

Hardware tuning picks up from where hardware sizing, which you learned about in Chapter 4, leaves off. Where hardware sizing is the act of determining how much and what kind of hardware to purchase, hardware tuning is the act of properly configuring (or reconfiguring) that hardware for optimal performance. Material in this chapter overlaps a bit with that in Chapter 4, as some hardware topics are extremely important to both tuning and sizing.

As hardware components become more and more sophisticated, the amount of tuning that can be done has also increased. Both SAN storage and SCSI RAID controllers have numerous tuning parameters, and the system BIOS and even the processors now offer optional settings. In fact, at the publication of this book, both Intel and AMD have CPUs that can operate in 32-bit or 64-bit modes, depending on what operating system you install. In this chapter, you will learn how to configure and tune the hardware components for optimal performance.

In addition to tuning the individual hardware components, the “system” might consist of several or many different tiers of web, application, and database servers. With multiple layers and sophisticated interactions between the different layers, infrastructure tuning is also important, but it is beyond the scope of this book.

Choosing the Right Hardware

The computer system is made up of many individual components all working together. Each of these components has its own job to perform, and each has its own performance characteristics.

The core of the system is the central processing unit (CPU), which actually processes all the calculations and instructions that run on the computer. It is better to be bound by the performance of the CPUs than by that of other subsystems. Built-in Linux operating system CPU utilities such as `ps`, `top`, and `vmstat` make it easy to tell when the CPUs are at full capacity. For example, CPU is saturated if the `vmstat` run queue is more than four times the number of CPUs. Or you may run across a single process consuming an entire CPU. The job of the hardware tuner is to configure the system to use CPU resources efficiently.

The base computer system utilizes three main resources (and many peripheral layers): CPU, memory, and the I/O subsystem. This section describes these components and what makes them tick.

CPU

The CPU and its cache(s) are the fastest components of the system. The cache(s) are one or more pieces of very high-speed memory, usually located on the same chip as the CPU, that are used to store recently used data and instructions so that they can provide quick access if this data is used again in a short time. Most modern CPU hardware designs actually take advantage of a cache built into the CPU core. This internal cache is known as a Level 1 (or L1) cache. Typically, an L1 cache is very small, on the order of 8 to 16 kilobytes in size, and is accessible in a few nanoseconds.

When a certain piece of data is wanted, the hardware first looks in the L1 cache. If the data is there, the hardware processes that data immediately. If the data is not available in the L1 cache, the hardware looks in the L2 cache, which is external to the CPU core but resides on the same chip in order to maximize performance. The L2 cache is connected to the CPU chip(s) on the same side of the memory bus as the CPU. To get to main memory, you need to use the memory bus, thereby reducing the speed of the memory access. Although the L2 cache is typically much slower than the L1 cache, it is usually much larger. Its larger size provides a better chance of getting a “cache hit.” Typical L2 caches range from 128K to 4M in size and are accessible in hundreds of nanoseconds.

Slower yet is the speed of the main memory. It is characteristically much slower than the L2 cache, on the order of 2–10 microseconds (roughly 1000 times slower than the L1 cache). The size of system memory can vary widely: 32-bit CPUs are limited to 4GB of directly addressed memory (they can address more through a multiphase operation), while 64-bit CPUs can directly address terabytes of memory directly. A comparison of component speeds is shown in Figure 15-1.

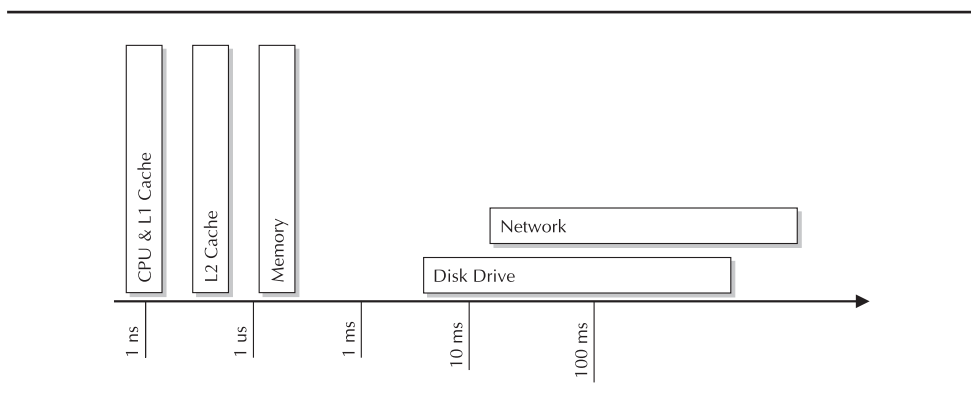


FIGURE 15-1. *Component speed comparison*

As you can see from the comparison of CPU speeds, the advantage in speed is enormous when you can retrieve data from the L1 cache rather than from disk. This is why we spend so much time trying to take advantage of the SGA in memory. This is also why hardware vendors expend such effort to improve CPU cache and memory bus speeds. However, other hardware vendor considerations also affect performance, as described in the following sections.

CPU Design The CPU works in synchronization with the system clock and executes instructions based on clock signals. The clock rate and CPU type determine how fast these instructions are executed. Virtually all modern CPUs are one of two types of processors: complex instruction set computer (CISC) or reduced instruction set computer (RISC), as described next.

CISC processors (such as those built by Intel and AMD) are by far the most common processors sold today. CISC processors are more common (in terms of volume of processors shipped) and offer a larger instruction set to the program developer than do RISC processors. Some of these instructions can be quite complicated, requiring several clock cycles to complete.

CISC processors are very complex and difficult to build. Because these CPU chips contain millions of internal components, the components are extremely close together. The physical scale causes problems because there is no room for error. Each year, technology allows more complex and faster chips to be built, but chip designers are starting to run into fundamental limits imposed by physics (at least the physics we know today).

These complex processors carry out a wide range of tasks and can sometimes perform two or more instructions at a time in parallel. CISC processors perform most tasks very efficiently, including those involved in RDBMS processing.

By contrast, RISC processors are based on the theory that if you can reduce the complexity and number of distinct instructions the CPU processes, the CPU can be simpler to build and can run faster. In addition, by putting fewer internal components inside the chip, its speed can be increased. One of the most popular RISC chips on the market today is the PA RISC chip from HP.

The system compiler determines what instructions are executed by the CPU. When the number of types of instructions was reduced, compilers were written to take advantage of this fact and to compensate for the missing instructions.

By reducing the instruction set, RISC manufacturers have been able to increase the clock speed to many times that of CISC chips. Although this faster clock speed is beneficial in some cases, it offers little improvement in others. One effect of a faster CPU is that the surrounding components such as L2 cache and memory must also run faster, at an increase in hardware cost.

Another goal of some RISC manufacturers is to design the chip so that the majority of instructions complete within one clock cycle. Some RISC chips today

can already do this. However, because some operations that can be performed by a single instruction on a CISC chip may require many instructions for a RISC chip, speed-to-speed comparisons cannot be made.

Multiprocessors Multiprocessor systems can provide significant performance with very good value. With many multiprocessor systems, you can start out with one or two processors and add additional processors as your business needs grow. However, planning is required, since not all systems are upgradeable. For example, you can purchase a four-CPU system with one CPU and upgrade it, but not past four CPUs. Multiprocessors fall into several categories; the two main types are symmetric multiprocessor (SMP) systems and massively parallel processing (MPP) systems.

Oracle typically scales very well with additional CPUs. By adding additional CPUs, you can see significant performance improvement with very little additional cost. Some factors that determine how much improvement you will get by adding more processors are the CPU cache and memory bus bandwidth. Systems that support large numbers of users performing small independent operations, such as OLTP systems, tend to scale very well with multiple CPUs. Other systems such as data warehouses might not benefit from large numbers of CPUs.

Symmetric multiprocessor (SMP) systems usually consist of a standard computer architecture with two or more CPUs that share the system memory, I/O bus, and disks. The CPUs are called symmetric because each processor is identical to any other processor in terms of function. Because the processors share system memory, each processor looks at the same data and the same operating system. In fact, the SMP architecture is sometimes called a tightly coupled architecture because the CPUs can even share the operating system.

In a typical SMP system, only one copy of the operating system runs. Each processor works independently of the others by taking the next available process that is ready to run. Because the Oracle architecture is based on many processes, each working independently, you can see great improvement by adding additional processors.

The SMP system has several advantages: it is cost effective, high performing, and easily upgradeable. The biggest advantage of SMP systems is that they are the most common and (because of economies of scale) the most cost effective. A typical SMP system supports from four to eight CPUs. A disadvantage of an SMP system is that all CPUs typically share the system bus and memory, so adding more processors increases the likelihood that the bandwidth of the bus will become saturated.

Massively parallel processor (MPP) architecture systems are based on many independent units. Each processor in an MPP system typically has its own resources such as its own local memory and I/O system. Each processor in an MPP system runs an independent copy of the operating system and its own independent copy of Oracle. An MPP system is sometimes referred to as a loosely coupled architecture.

You can think of an MPP system as large clusters of independent units that communicate through a high-speed interconnect. As with SMP systems, as you add processors, you will eventually hit the bandwidth limitations of the interconnect. However, the number of processors with which you hit this limit is typically much larger than with SMP systems.

If you can divide the application among the nodes in the cluster, MPP systems can achieve quite high scalability. Although MPP systems can achieve much higher performance than SMP systems, they generally cost more.

Regardless of whether you use a single-processor system, an SMP system, or an MPP system, the basic architecture of the CPUs is similar. In fact, you can find the same Intel processors in both SMP and MPP systems. It is the way those processors handle memory and how CPUs are clustered together that differs between SMP and MPP systems. Included in MPP systems are computers that utilize the NUMA (Non-Uniform Memory Architecture) design. NUMA systems do not share main memory like SMP systems; instead, each processor or set of processors uses its own memory. In some ways, a NUMA system is almost like a cluster of individual systems; in other ways, it is like a big multiprocessor.

CPU and CPU Cache As you learned earlier in this chapter, the system cache is very important to overall system performance. The job of the cache is to allow quick access to recently used instructions or data. In addition, the cache is used to perform read-aheads, much in the same way that `DB_FILE_MULTIBLOCK_READ_COUNT` allows Oracle to perform read-ahead operations in the database. A cache is always used to store and retrieve data faster than the next level of storage (the L1 cache is faster than the L2 cache, and the L2 cache is faster than main memory).

By caching frequently used instructions and data, you increase the likelihood of a cache hit, which can save precious clock cycles that otherwise would have been spent retrieving data from memory or disk.

A large CPU cache allows more data and executable code to be stored on the local processor than in memory, thereby reducing the number of times the CPU must access main memory. Whenever the CPU accesses memory, system performance suffers while the CPU waits for that data or code to be retrieved; if the memory bus is busy at that time, the CPU must wait for the bus to become free, lengthening the delay even more.

Random Access Memory (RAM)

The main system memory is a set of memory chips, either protected or not protected from faults, that stores data and instructions used by the system. System memory can be protected by parity or a more sophisticated, advanced ECC correction method. The system memory can theoretically range in size from 4MB on a small PC to 4GB

(or more using the Intel PAE architecture) on a large 32-bit server, and up to 16TB on a 64-bit processor; however, today even the smallest PCs come with at least 256MB of RAM.

Given our discussion earlier in this chapter on how the system hardware operates, it is obvious that any operation that has to access slower components, such as a disk or a network, will slow down processing operations. Thus, it is very important to have a sufficient amount of memory in your system. This system memory is allocated to the Oracle System Global Area (SGA), and the user memory (the Program Global Area). Tune the shared pool first, as an insufficient shared pool can hurt Oracle performance more than problems with any other SGA component. Once the Shared Pool has enough memory resources, the more database buffers you can allocate to the DBMS, the better. Be careful, though, to not starve out the PGA memory that is needed by your processes, and at all costs, avoid paging. You can never have too much memory in your system. Anything that can be cached will reduce system I/O, thus improving performance.

System memory is accessed by the CPUs through a high-speed bus that allows large amounts of data and instructions to be moved from the CPU to the L2 cache very quickly. Typically, data and instructions are read from memory in large chunks and put into the cache, anticipating that the next instruction or data in that chunk is likely to be accessed next. This process is known as prefetching.

Depending on the specific implementation of an SMP system, the memory bus may be shared by all system processors. Alternatively, each processor may have a private bus to memory.

The amount of memory available to processes is limited by the physical memory in the system, or extended if you are using a virtual memory system.

Linux Memory Tuning for Oracle Linux does not require much memory tuning for Oracle, but the little that is needed is critical. The Linux parameters that need to be set are as follows:

```
/proc/sys/kernel/shmall 2097152
```

```
/proc/sys/kernel/shmmni 4096
```

```
/proc/sys/kernel/shmmax <slightly larger than the SGA size>
```

The parameter `shmmax` is typically set to 2147483648 (2GB), by default.

If you need to create an SGA greater than 2GB, you must increase the value of `shmmax`. This parameter represents the maximum size of a single shared memory segment.

Virtual Memory System In a virtual memory system, special memory management functions allow programs to address more memory than is physically installed in the system. This memory, known as *virtual memory*, appears to programs as a large amount of memory that can be mapped to physical memory. Data that is stored in virtual memory must be paged into physical memory for the CPU to use it, and then copied out to disk to make room in physical memory for another process. The process of paging in or paging out can be quite time-consuming and uses a lot of system resources. It bears repeating that access to disk is approximately 1,000 times slower than an access to memory. Paging should be avoided at all costs, even if it means reducing the size of the Oracle SGA.

32-bit vs. 64-bit Processors

The more powerful 64-bit processors have been available for well over a decade but have not become mainstream, because each 64-bit processor architecture was proprietary and was not supported by more than one operating system. Nonetheless, 64-bit processors offer several advantages over 32-bit processors, including the ability to address more memory and to process more information in a single clock cycle. Unfortunately, these 64-bit processors have not achieved the popularity of 32-bit processors. However, the new hybrid 64-bit / 32-bit processors are gaining popularity quickly. These processors are described in a few paragraphs.

32-bit Addressing Various 32-bit architectures have been around for many years and are the most common architecture in production. Almost every PC uses a 32-bit processor from either Intel or AMD. The disadvantage of the 32-bit processor is that 32 bits can directly address a maximum of 4GB of RAM. In the past few years, there have been some workarounds by allowing more memory to be addressed via the Physical Address Extension (PAE).

PAE allows physical memory to be accessed above 4GB, but it does not increase the amount of virtual memory available to a process. Although it does allow for more

Linux Paging and Swapping

Linux does not require as much swap space as most Unix operating systems. In fact it is not uncommon to find Linux systems with 4GB or more memory with 1GB or less swap space. Since swap space is used only when you run out of physical memory, theoretically the more memory that you have, the less chance that you will use the swap space.

physical memory to be accessed, it is somewhat inefficient and does not provide all of the benefits of a 64-bit system. Oracle can access this memory using indirect data buffers. With indirect data buffers, a RAM disk is created that Oracle uses as an intermediate storage area. Although much faster than disk access, it is significantly slower than accessing memory directly.

64-bit Addressing With 64-bit systems, you can access up to 16 terabytes of RAM. This can be a great advantage for larger databases. Although it is unlikely that systems will be built with 16 terabytes of RAM in the near future, systems with hundreds of gigabytes of RAM are becoming commonplace.

The 64-bit bus is twice as large as the 32-bit bus, thus allowing twice as much data to be transferred from one subsystem to another at a time. This is very useful, especially when performing 64-bit mathematical calculations.

Hybrid X86-64 Systems Some of the newer processors such as the Opteron and Athlon-64 from AMD and the EM64T from Intel are hybrid systems, essentially 32-bit processors that are fully compatible with both x86 operating systems and 64-bit memory addressing. These systems are fast becoming very popular and are now supported by both Linux and Windows operating systems. Oracle 10g for both Linux and Windows is available for the x86-64 architecture in 64-bit mode.

These systems have the advantage of running either a 64-bit or 32-bit operating system. If you are running a 64-bit operating system, you have the option of running either a 64-bit or 32-bit version of Oracle. Of course, if you are running 64-bit, you can access more memory with higher performance.

Since these processors are currently shipping in large quantities from vendors such as Dell, HP, and IBM, you might actually be running a 64-bit-capable CPU and not even know it. Check with your hardware vendor. You might consider upgrading your system from 32-bit processing to 64-bit processing by just changing the operating system.

Tuning Hardware for Linux

In addition to choosing the right hardware, it is very important to properly configure and tune this hardware. Probably the most important areas of hardware tuning are in both the hardware and the filesystem.

I/O Tuning

I/O tuning is key to the performance of the Oracle database server, since the main task of Oracle is to serve up data, and moving and delivering data means I/Os. The I/O subsystem was covered in some detail in Chapter 4. Tuning the I/O subsystem is

not much different from sizing it. Care should be taken to configure and tune the I/O subsystem to optimize I/O performance. The `iostat` command is one of the most useful for monitoring the I/O subsystem in Linux. Running this command with the `-x` flag shows a great deal of information about the performance of the I/O subsystem, as shown here:

```
ptc6:~ # iostat -x 10 100
Linux 2.4.21-138-default (ptc6)          06/19/04

avg-cpu:  %user   %nice   %sys   %idle
           1.77    0.00    0.68   97.55

Device:            rrqm/s  wrqm/s   r/s    w/s  rsec/s  wsec/s   kB/s    kB/s  avgrq-sz
avgqu-sz  await  svctm  %util
/dev/hda      8.64    2.14   4.23   1.16  100.30   26.42    50.15   13.21   23.49
2.92 541.04 48.52  2.62
/dev/hda1     0.36    0.00   0.02   0.00    0.75    0.00    0.38    0.00   44.33
0.00 56.67 40.00  0.01
/dev/hda2     0.00    0.00   0.00   0.00    0.01    0.00    0.00    0.00    8.00
0.00 50.00 50.00  0.00
/dev/hda3     8.27    2.14   4.21   1.16  99.51   26.42    49.76   13.21   23.44
2.92 542.98 48.49  2.61
/dev/hdb      0.05    1.84   4.11   0.57   33.09   19.30    16.55    9.65   11.17
1.10 234.95 91.98  4.31
/dev/hdb1     0.00    0.00   0.00   0.00    0.00    0.00    0.00    0.00    8.00
0.00 100.00 100.00  0.00
/dev/hdb2     0.03    1.84   4.11   0.57   33.05   19.30    16.52    9.65   11.18
1.10 235.12 91.99  4.31

avg-cpu:  %user   %nice   %sys   %idle
           0.40    0.00    1.60   98.00

Device:            rrqm/s  wrqm/s   r/s    w/s  rsec/s  wsec/s   kB/s    kB/s  avgrq-sz
avgqu-sz  await  svctm  %util
/dev/hda      0.00   24.40   0.00  34.50    0.00  491.20    0.00   245.60   14.24
314.54 8297.97 53.91 18.60
/dev/hda1     0.00    0.00   0.00   0.00    0.00    0.00    0.00    0.00    0.00
0.00 0.00 0.00 0.00
/dev/hda2     0.00    0.00   0.00   0.00    0.00    0.00    0.00    0.00    0.00
0.00 0.00 0.00 0.00
/dev/hda3     0.00   24.40   0.00  34.50    0.00  491.20    0.00   245.60   14.24
314.54 8297.97 53.91 18.60
/dev/hdb      0.10   63.00  77.70 22.80   623.20  686.40   311.60   343.20   13.03
38.24 380.40 94.73 95.20
/dev/hdb1     0.00    0.00   0.00   0.00    0.00    0.00    0.00    0.00    0.00
0.00 0.00 0.00 0.00
/dev/hdb2     0.10   63.00  77.70 22.80   623.20  686.40   311.60   343.20   13.03
38.24 380.40 94.73 95.20
```

By default, the `iostat` command outputs one line of I/O information for all devices. The command `iostat -x 10 100` outputs data 100 times at ten-second intervals, and if you are interested in only one device, you can specify the device name, as shown here:

```

ptc6:~ # iostat -x /dev/hdb 10 100
Linux 2.4.21-138-default (ptc6)          06/19/04

avg-cpu:  %user   %nice   %sys    %idle
           1.68    0.00    0.78   97.54

Device:    rrqm/s  wrqm/s   r/s    w/s  rsec/s  wsec/s   kB/s    kB/s  avgrq-sz
avgqu-sz  await  svctm  %util
/dev/hdb   0.04   9.18   6.80   4.27  54.55  109.85   27.27   54.92   14.86
50.06 4221.90  80.42   8.90

avg-cpu:  %user   %nice   %sys    %idle
           3.10    0.00    3.30   93.60

Device:    rrqm/s  wrqm/s   r/s    w/s  rsec/s  wsec/s   kB/s    kB/s  avgrq-sz
avgqu-sz  await  svctm  %util
/dev/hdb   0.00 235.50   0.00 68.50   0.00 2044.80   0.00  1022.40   29.85
765.52 20539.27 51.53 35.30

```

The `iostat` command is very useful in determining if you are experiencing an I/O bottleneck. I/O problems are related to capacity and sizing, misconfiguration, or tuning.

I/O and Oracle Datafile Placement Recently, Oracle started recommending that all data, index, and redo log files be placed across one or more large hardware LUNs extending over many disk drives. This approach is referred to as SAME (stripe and mirror everything or storage administration made easy, depending on which paper you read).

Although some believe that using large hardware volumes is preferable, I still prefer to store redo log files and data files on different physical volumes. Although this is somewhat contrary to the Oracle-recommended method, there are several reasons that I recommend this approach.

- Redo log files use sequential I/O. By separating the redo log files, you can achieve better performance on those volumes with fewer drives, as discussed in Chapter 4. However, a logical drive (LUN) might not consist of a dedicated disk drive.
- In the event of a single disk failure, you would not lose both the data files and the online redo log files if they were separated onto separate sets of drives.
- Even though most I/O subsystems are protected against a single point of failure such as a single disk failure, you could lose an entire RAID set due to human error. By separating the redo log files and data files, you gain an additional level of protection.

I feel that, for both performance and data protection, the redo logs, the undo tablespace(s), and the data files should be separated.

Miscellaneous I/O Tuning There are a number of tunable parameters in the I/O subsystem and the filesystem. The filesystem can be tuned via the `tune2fs` command and can be viewed with the `dumpe2fs` command, as shown here:

```
ptc6:~ # dumpe2fs /dev/hdb2 | less
dumpe2fs 1.28 (31-Aug-2002)
Filesystem volume name:   /1
Last mounted on:         <not available>
Filesystem UUID:         aa00942b-4b36-4248-bb9f-962dce0b5bab
Filesystem magic number: 0xEF53
Filesystem revision #:   1 (dynamic)
Filesystem features:     has_journal filetype needs_recovery sparse_super
Filesystem state:        clean
Errors behavior:         Continue
Filesystem OS type:      Linux
Inode count:              7225344
Block count:              14430150
Reserved block count:    721507
Free blocks:              12709195
Free inodes:              7079490
First block:              0
Block size:               4096
Fragment size:           4096
Blocks per group:        32768
Fragments per group:    32768
Inodes per group:        16384
Inode blocks per group:  512
Last mount time:         Sat Jun 19 08:38:59 2004
Last write time:        Sat Jun 19 08:38:59 2004
Mount count:             27
Maximum mount count:    -1
Last checked:            Sun Nov  2 10:13:11 2003
Check interval:         0 (<none>)
Reserved blocks uid:     0 (user root)
Reserved blocks gid:     0 (group root)
First inode:             11
Inode size:              128
Journal UUID:            <none>
Journal inode:           8
Journal device:          0x0000
First orphan inode:      0
```

Although it is interesting to see how the filesystem is configured, there is not really much to tune after it has been created. The most important filesystem tuning is done at creation time. One of the most important parameters is the block size. Since the typical Oracle database uses an 8K block size, a smaller OS block size means that the OS must perform multiple I/Os for each Oracle request. This does

not mean that I/Os occur in OS block size units, only that it can perform an I/O operation up to that size. You should make the OS block size as close to the Oracle block size as possible, or larger. Unfortunately, ext2 and ext3 filesystems have a maximum of 4K block sizes. The Oracle Cluster Filesystem (OCFS) can create a 1M block size. I usually prefer a 128K block size in OCFS. When creating an ext filesystem, use 4K as the block size.

This does not mean OCFS writes only 128K at a time. The actual size of the write will be the size of the I/O that has been issued to the filesystem. This is merely a function of how space is allocated on the system, and how efficiently OCFS responds to small files as opposed to large files. IOs to OCFS are a minimum of 4K in size but can be much larger if necessary.

In addition to the filesystem parameters, the command `elvtune` can be used to tune the elevator sorting algorithm in Linux. Elevator sorting is used to sort I/Os such that the heads pick up requests in physical sequence, rather than bouncing back and forth. When an elevator is moving up, it will stop at floor 3 and then 4, even if 4 was pushed before 3. Running `elvtune device` will show you the current settings, as shown here:

```
ptc6:~ # elvtune /dev/hdb2

/dev/hdb2 elevator ID          2
      read_latency:           128
      write_latency:          512
      max_bomb_segments:      8192
```

These parameters specify an artificial latency that is designed to allow multiple I/Os to queue up in order to allow for elevator sorting. Although this is a good thing for a single disk drive, it is bad for a RAID controller or a SAN. This is because it essentially delays I/Os so that the device driver can sort them. But with a RAID controller or SAN, the hardware, not the device driver, does elevator sorting.

Tuning RAID Controllers Sometimes it is possible to tune your RAID controller or SAN in order to achieve better performance. What can be tuned and how depend upon the type and brand of RAID controller. Typically, you can tune the following components:

- **Cache page size** The cache page size should be at least as large as the Oracle page size. This reduces extra overhead that might be incurred by having to allocate multiple cache pages for one Oracle I/O.
- **Read and write cache sizes** Typically, the read and write caches can be specified independently. Depending on what your application does, you might want more read cache or you might want more write cache.

- **Read and write cache usage** Most RAID and SAN systems allow you to allocate read and write caches on a per-LUN basis. Some LUNs might benefit from a read cache, others from a write cache, and some from both. Random I/Os (OLTP) don't benefit from a read cache. Heavy writes (redo, archive, etc.) benefit from a write cache.
- **Read-ahead or prefetch** In some cases, the prefetch might help Oracle performance when an application requires a significant number of table scans, such as in data warehouse environments. In an OLTP environment, when I/Os are mostly random in nature, prefetching probably won't help.

There are a number of ways that you can tune the I/O subsystem's hardware, depending on what brand and type of hardware you have. How to tune it depends on the type of application, I/O access patterns, and the number of I/Os that are occurring.

OCFS Tuning

The Oracle Cluster Filesystem (OCFS) was introduced to provide a convenient, high-performance filesystem that is clusterable. By being clusterable, OCFS allows multiple systems in a RAC cluster to access the same data files, which is key to RAC. Accessing OCFS is done via a mechanism called O_DIRECT. O_DIRECT allows direct mode access to the filesystem, bypassing the Linux I/O cache.

Because of characteristics of OCFS, you should access OCFS files only with O_DIRECT. Oracle provides a set of utilities known as the coreutils. The coreutils from Oracle replaces standard Linux utilities such as cp, dd, and mv with utilities of the same name with the addition of the O_DIRECT option. By using the O_DIRECT option (-o) with a block size option (*direct*), performance can be dramatically improved. In a recent test that I performed, dd performance

RAC I/O Access

Oracle Real Application Clusters (RAC) require shared storage that can be seen by all nodes in the cluster. Various options for this shared storage are raw partitions, ASM (Automatic Storage Management), and OCFS (Oracle Cluster Filesystem). Each has its own characteristics and is a viable option. The option you choose will depend on your needs.

was increased from 1.5 MBps to nearly 50 MBps using a 256K block size, as shown in the following table.

O_DIRECT size	Time	MB/sec	GB/hr
0	815.00	1.26	4.42
64K	24.03	42.62	149.84
128K	21.28	48.13	169.20
256K	20.58	49.75	174.91
512K	43.78	23.39	82.23
1M	44.10	23.22	81.64

These tests were performed by copying a 1GB file from disk to /dev/null, as follows:

```
time dd if=filetest | dd of=/dev/null
time dd -o_direct=256K if=filetest | dd of=/dev/null
```

This test used an EMC SAN configured using RAID 10. As you can see in the table, the performance difference is dramatic. This test was performed multiple times and is reproducible.

Tuning OCFS using the O_DIRECT flag can dramatically improve backup and restore operations as well as other file operations that are necessary as part of regular DBA duties. Don't be afraid to experiment and take advantage of O_DIRECT features.

Network Tuning on Linux

The network is very important to overall system performance. This is especially true of an Oracle RAC cluster. The performance of the interconnect can be very important to the performance of the cluster because of the amount of interconnect traffic and global buffer caching. There are a few things that can be done to improve the performance of the network.

Coreutils

The Oracle coreutils for OCFS are available at www.ocfs.org.

Choosing the Right Network

The first step in configuring your system for optimal network performance is to configure the network properly. It is important to make sure that the network is performing as specified. When configuring a network to run at gigabit speeds, it is usually necessary to configure the adapter for autonegotiation (depending on the controller). If your network switches are configured incorrectly, it is possible that the adapter will not run at the proper speed.

The Linux utility `ethtool` will show you how your adapter is configured. Run the command **`ethtool adapter`** in order to view the adapter settings, as shown here:

```
ptc6:~ # ethtool eth0
Settings for eth0:
    Supported ports: [ TP ]
    Supported link modes:   10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
    Supports auto-negotiation: Yes
    Advertised link modes:  10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
    Advertised auto-negotiation: Yes
    Speed: 1000Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 0
    Transceiver: internal
    Auto-negotiation: on
    Supports Wake-on: umbg
    Wake-on: g
    Link detected: yes
```

If the adapter is not running at the correct speed and duplex setting, you must determine the source of the problem and correct it in order for the adapter to run correctly.

Tuning Linux for Network Performance

It is often advantageous to tune the UDP send and receive buffers in order to achieve higher UDP performance. This is especially important if you are running in an Oracle RAC cluster. These parameters are as follows:

Parameter	Explanation
<code>/proc/sys/net/core/rmem_default</code>	Default receive window
<code>/proc/sys/net/core/rmem_max</code>	Maximum receive window
<code>/proc/sys/net/core/wmem_default</code>	Default send window
<code>/proc/sys/net/core/wmem_max</code>	Maximum send window
<code>/proc/sys/net/ipv4/tcp_rmem</code>	Memory reserved for TCP rcv buffers
<code>/proc/sys/net/ipv4/tcp_wmem</code>	Memory reserved for TCP snd buffers

In order to set these parameters, you can echo to the preceding filenames, as shown here:

```

echo 262144 > /proc/sys/net/core/rmem_default
echo 262144 > /proc/sys/net/core/rmem_max
echo 262144 > /proc/sys/net/core/wmem_default
echo 262144 > /proc/sys/net/core/wmem_max
echo "4096 65536 4194304" > /proc/sys/net/ipv4/tcp_rmem
echo "4096 65536 4194304" > /proc/sys/net/ipv4/tcp_wmem

```

Alternatively, you can set them in the `/etc/sysctl.conf` file, as shown here:

```

net.core.rmem_default = 262144
net.core.rmem_max = 262144
net.core.wmem_default = 262144
net.core.wmem_max = 262144
net.ipv4.tcp_rmem = 4096 65536 4194304
net.ipv4.tcp_wmem = 4096 65536 4194304

```

In addition, the following `/etc/sysctl.conf` network parameters can help improve the failover performance in a RAC cluster:

```

net.ipv4.tcp_keepalive_time = 3000
net.ipv4.tcp_keepalive_intvl = 30
net.ipv4.tcp_retries2 = 3
net.ipv4.tcp_syn_retries = 2

```

Configuring the network properly can improve the performance of the system greatly. As mentioned earlier, the interconnect performance is critical to the performance of an Oracle RAC cluster.

Summary

In this chapter, you have seen many different ways that the system hardware itself can be tuned via the modification of driver parameters and firmware configurations. The configuration parameters to be changed depend on the type of hardware and the way your system and application operate. What might work well for an OLTP system might not work well for a batch or decision support system. The difference between hardware tuning and sizing is that sizing is the act of choosing the right hardware for your needs, whereas hardware tuning is the act of modifying that hardware to perform optimally in your configuration.

