7.2. Benefits of Monitoring

There are two approaches to monitoring. You may want to ensure nothing has changed (no degradation of performance and no security breaches) or to investigate what has changed or gone wrong. Monitoring the system to ensure nothing has changed is called *proactive monitoring*, whereas monitoring to see what went wrong is called *reactive monitoring*. Sadly, most monitoring occurs in a reactive manner. Very few IT professionals have the time or resources to conduct proactive monitoring. Reactive monitoring is therefore the only form of monitoring some professionals understand.

However, if you take the time to monitor your system proactively, you can eliminate a lot of reactive work. For example, if your users complain about poor performance (the number one trigger for reactive monitoring), you have no way of knowing how much the system has degraded unless you have previous monitoring results with which to compare. Recording such results is called *forming a baseline* of your system. That is, you monitor the performance of your system under low, normal, and high loads over a period of time. If you do the sampling frequently and consistently, you can determine the typical performance of the system under various loads. Thus, when users report performance problems, you can sample the system and compare the results to your baseline. If you include enough detail in your historical data, you can normally see, at a glance, which part of the system has changed.

7.3. System Components to Monitor

You should examine four basic parts of the system when monitoring performance:

*Processor*

Check to see how much of it is utilized and what peaks are reached by utilization.

*Memory*

Check to see how much is being used and how much is still available to run programs.

*Disk*

Check to see how much disk space is available, how disk space is used, and what demand there is for it and how fast it delivers content (response time).

*Network*

Check for throughput, latency, and error rates when communicating with other systems on the network.

#### 7.3.1. Processor

Monitor the system’s CPU to ensure there are no runaway processes and that the CPU cycles are being shared equally among the running programs. One way to do this is to call up a list of the programs running and determine what percentage of the CPU each is using. Another method is to examine the load average of the system processes. Most operating systems provide several views of the performance of the CPU.

***Note:***

A process is a unit of work in a Linux or Unix system. A program may have one or more processes running at a time. Multithreaded applications, such as MySQL, generally appear on the system as multiple processes.

When a CPU is under a performance load and contention is high, the system can exhibit very slow performance and even periods of seeming inactivity. When this occurs, you must either reduce the number of processes or reduce the CPU usage of processes that seem to be consuming more CPU time. But be sure to monitor the CPUs to make sure that high CPU utilization is really the cause of the problem—slowness is even more likely to occur because of memory contention, discussed in the next section.

Some of the common solutions to CPU overloading include:

*Provision a new server to run some processes*

This is, of course, the best method, but requires money for new systems. Experienced system administrators can often find other ways to reduce CPU usage, especially when the organization is more willing to spend your time than to spend money.

*Remove unnecessary processes*

An enormous number of systems run background processes that may be useful for certain occasions but just bog down the system most of the time. However, an administrator must know the system very well to identify which processes are nonessential.

*Kill runaway processes*

These probably stem from buggy applications, and they are often the culprit when performance problems are intermittent or rare. In the event that you cannot stop a runaway process using a controlled or orderly method, you may need to terminate the process abruptly using a *force quit* dialog or the command line.

*Optimize applications*

Some applications routinely take up more CPU time or other resources than they really need. Poorly designed SQL statements are often a drag on the database system.

*Lower process priorities*

Some processes run as background jobs, such as report generators, and can be run more slowly to make room for interactive processes.

*Reschedule processes*

Maybe some of those report generators can run at night when system load is lower.

Processes that consume too much CPU time are called *CPU-bound* or *processor-bound*, meaning they do not suspend themselves for I/O and cannot be swapped out of memory.

If you find the CPU is not under contention and there are either few processes running or no processes consuming large amounts of CPU time, the problem with performance is likely to be elsewhere: waiting on disk I/O, insufficient memory, excessive page swapping, etc.

#### 7.3.2. Memory

Monitor memory to ensure your applications are not requesting so much memory that they waste system time on memory management. From the very first days of limited random access memory (RAM, or main memory), operating systems have evolved to employ a sophisticated method of using disk memory to store unused portions or pages of main memory. This technique, called *paging* or *swapping*, allows a system to run more processes than main memory can load at one time, by storing the memory for suspended processes and later retrieving the memory when the process is reactivated. While the cost of moving a page of memory from memory to disk and back again is relatively high (it is time-consuming compared to accessing main memory directly), modern operating systems can do it so quickly that the penalty isn’t normally an issue unless it reaches such a high level that the processor and disk cannot keep up with the demands.

However, the operating system may perform some swapping at a high level periodically to reclaim memory. Be sure to measure memory usage over a period of time to ensure you are not observing a normal cleanup operation.

When periods of high paging occur, it is likely that low memory availability may be the result of a runaway process consuming too much memory or too many processes requesting too much memory. This kind of high paging, called *thrashing*, can be treated the same way as a CPU under contention. Processes that consume too much memory are called *memory-bound*.

When treating memory performance problems, the natural tendency is to add more memory. While that may indeed solve the problem, it is also possible that the memory is not allocated correctly among the various subsystems.

There are several things you can do in this situation. You can allocate different amounts of memory to parts of the system—such as the kernel or filesystem—or to various applications that permit such tweaking, including MySQL. You can also change the priority of the paging subsystem so the operating system begins paging earlier.

***Warning:***

Be very careful when tweaking memory subsystems on your server. Be sure to consult your documentation or a book dedicated to improving performance for your specific operating system.

If you monitor memory and find that the system is not paging too frequently, but performance is still an issue, the problem is likely to be related to one of the other subsystems.

#### 7.3.3. Disk

Monitor disk usage to ensure there is enough free disk space available, as well as sufficient I/O bandwidth to allow processes to execute without significant delay. You can measure this using either a*per-process* or *overall transfer* rate to and from disk. The per-process rate is the amount of data a single process can read or write. The overall transfer rate is the maximum bandwidth available for reading and writing data on disk. Some systems have multiple disk controllers; in these cases, overall transfer rate may be measured per disk controller.

Performance issues can arise if one or more processes are consuming too much of the maximum disk transfer rate. This can have very detrimental effects on the rest of the system in much the same way as a process that consumes too many CPU cycles: it “starves” other processes, forcing them to wait longer for disk access.

Processes that consume too much of the disk transfer rate are called *disk-bound*, meaning they are trying to access the disk at a frequency greater than the available share of the disk transfer rate. If you can reduce the pressure placed on your I/O system by a disk-bound process, you’ll free up more bandwidth for other processes.

One way to meet the needs of a process performing a lot of I/O to disk is to increase the block size of the filesystem, thus making large transfers more efficient and reducing the overhead imposed by a disk-bound process. However, this may cause other processes to run more slowly.

***Warning:***

Be careful when tuning filesystems on servers that have only a single controller or disk. Be sure to consult your documentation or a book dedicated to improving performance for your specific operating system.

If you have the resources, one strategy for dealing with disk contention is to add another disk controller and disk array and move the data for one of the disk-bound processes to the new disk controller. Another strategy is to move a disk-bound process to another, less utilized server. Finally, in some cases it may be possible to increase the bandwidth of the disk by upgrading the disk system to a faster technology.

There are differing opinions as to where to optimize first or even which is the best choice. We believe:

* If you need to run a lot of processes, maximize the disk transfer rate or split the processes among different disk arrays or systems.
* If you need to run a few processes that access large amounts of data, maximize the per-process transfer rate by increasing the block size of the filesystem.

You may also need to strike a balance between the two solutions to meet your unique mix of processes by moving some of the processes to other systems.

#### 7.3.4. Network Subsystem

Monitor network interfaces to ensure there is enough bandwidth and that the data being sent or received is of sufficient quality.

Processes that consume too much network bandwidth, because they are attempting to read or write more data than the network configuration or hardware make possible, are called *network-bound*. These processes keep other processes from accessing sufficient network bandwidth to avoid delays.

Network bandwidth issues are normally indicated by utilization of a percentage of the maximum bandwidth of the network interface. You can solve these issues with processes by assigning the processes to specific ports on a network interface.

Network data quality issues are normally indicated by a high number of errors encountered on the network interface. Luckily, the operating system and data transfer applications usually employ*checksumming* or some other algorithm to detect errors, but retransmissions place a heavy load on the network and operating system. Solving the problem may require moving some applications to other systems on the network or installing additional network cards, which normally requires a diagnosis followed by changing the network hardware, reconfiguring the network protocols, or moving the system to a different subnet on the network.

***Note:***

You may hear the terms *I/O-bound* or *I/O-starved* when referring to processes. This normally means the process is consuming too much disk or network bandwidth.

7.4. Monitoring Solutions

For each of the four subsystems just discussed, a modern operating system offers its own specific tools that you can use to get information about the subsystem’s status. These tools are largely standalone applications that do not correlate (at least directly) with the other tools. As you will see in the next sections, the tools are powerful in their own right, but it requires a fair amount of effort to record and analyze all of the data they produce.

Fortunately, a number of third-party monitoring solutions are available for most operating and database systems. The following are a few of the more notable offerings. It is often best to contact your systems providers for recommendations on the best solution to meet your needs and maintain compatibility with your infrastructure. Most vendors offer system monitoring tools as an option.

*up.time*

[**http://www.uptimesoftware.com/**](http://www.uptimesoftware.com/)

*Cacti*

[**http://www.cacti.net/**](http://www.cacti.net/)

*KDE System Guard (KSysGuard)*

[**http://docs.kde.org/stable/en/kdebase-workspace/ksysguard/index.html**](http://docs.kde.org/stable/en/kdebase-workspace/ksysguard/index.html)

*Gnome System Monitor*

[**http://library.gnome.org/users/gnome-system-monitor/**](http://library.gnome.org/users/gnome-system-monitor/)

*Nagios*

[**http://www.nagios.org/**](http://www.nagios.org/)

*Sun Management Center*

[**http://www.sun.com/software/products/sunmanagementcenter/index.xml**](http://www.sun.com/software/products/sunmanagementcenter/index.xml)

*MySQL Enterprise Monitor*

[**http://www.mysql.com/products/enterprise/monitor.html**](http://www.mysql.com/products/enterprise/monitor.html)

***Note:***

We will discuss the MySQL Enterprise Monitor and automated monitoring and report in greater detail in [**Chapter 13**](http://proquest.safaribooksonline.com.ezproxy.torontopubliclibrary.ca/9780596807313/the_mysql_enterprise#the_mysql_enterprise).

The following sections describe the built-in monitoring tools for some of the major operating systems. We will study the Linux and Unix commands in a little more detail, as they are particularly suited to investigating the performance issues and strategies we’ve discussed. However, we will also include an examination of the monitoring tools for Mac OS X and Microsoft Windows.

### 7.5. Linux and Unix Monitoring

Database monitoring on Linux or Unix can involve tools for monitoring the CPU, memory, disk, network, and even security and users. In classic Unix fashion, all of the core tools run from the command line and most are located in the *bin* or *sbin* folders. [**Table 7-1**](javascript:moveTo('system_monitoring_tools_for_linux_and_un');) includes the list of tools we’ve found useful, with a brief description of each.

| Table 7-1. System monitoring tools for Linux and Unix | |
| --- | --- |
| **Utility** | **Description** |
| *ps* | Shows the list of processes running on the system. |
| *top* | Displays process activity sorted by CPU utilization. |
| *vmstat* | Displays information about memory, paging, block transfers, and CPU activity. |
| *uptime* | Displays how long the system has been running. It also tells you how many users are logged on and the system load average over 1, 5, and 15 minutes. |
| *free* | Displays memory usage. |
| *iostat* | Displays average disk activity and processor load. |
| *sar* | System activity report. Allows you to collect and report a wide variety of system activity. |
| *pmap* | Displays a map of how a process is using memory. |
| *mpstat* | Displays CPU usage for multiprocessor systems. |
| *netstat* | Displays information about network activity. |
| *cron* | A subsystem that allows you to schedule the execution of a process. You can schedule execution of these utilities so you can collect regular statistics over time or check statistics at specific times, such as during peak or minimal loads. |

***Note:***

Some operating systems provide additional or alternative tools. Consult your operating system documentation for additional tools for monitoring your system performance.

As you can see from [**Table 7-1**](javascript:moveTo('system_monitoring_tools_for_linux_and_un');), a rich variety of tools is available with a host of potentially useful information. The following sections discuss some of the more popular tools and explain briefly how you can use them to identify the problems described in the previous sections.

#### 7.5.1. Process Activity

Several commands provide information about processes running on your system, notably top, iostat*,*mpstat*,* and ps.

##### 7.5.1.1. The top command

The top command provides a summary of system information and a dynamic view of the processes on your system ranked by the most CPU-intensive tasks. The display typically contains information about the process, including the process ID, the user who started the process, its priority, the percentage of CPU it is using, how much time it has consumed, and, of course, the command used to start the process. However, some operating systems have slightly different reports. This is probably the most popular utility in the set because it presents a snapshot of your system every few seconds. [**Figure 7-1**](javascript:moveTo('the_top_command-010');)shows the output when running top on a Linux (Ubuntu) system under moderate load.

##### Figure 7-1. The top command

The system summary is located at the top of the listing and has some interesting data. It shows the percentages of CPU time for user (%us); system (%sy); nice (%ni), which is the time spent running users’ processes that have had their priorities changed; I/O wait (%wa); and even the percentage of time spent handling hardware and software interrupts. Also included are the amount of memory and swap space available, how much is being used, how much is free, and the size of the buffers.

Below the summary comes the list of processes, in descending order (which is from where the name of the command derives) based on how much CPU time is being used. In this example, a Bash shell is currently the task leader followed by one or several installations of MySQL.

|  |
| --- |
| Niceness You can change the priority of a process on a Linux or Unix system. You may want to do this to lower the priorities of processes that require too much CPU power, are of lower urgency, or could run for an extended period but that you do not want to cancel or reschedule. You can use the commands nice*,* ionice*,* and renice to alter the priority of a process.  Most distributions of Linux and Unix now group processes that have had their priorities changed into a group called nice. This allows you to get statistics about these modified processes without having to remember or collate the information yourself. Having commands that report the CPU time for nice processes gives you the opportunity to see how much CPU these processes are consuming with respect to the rest of the system. For example, a high value on this parameter may indicate there is at least one process with too high of a priority. |

Perhaps the best use of the top command is to allow it to run and refresh every three seconds. If you check the display at intervals over time, you will begin to see which processes are consuming the most CPU time. This can help you determine at a glance whether there is a runaway process.

***Note:***

You can change the refresh rate of the command by specifying the delay on the command. For example,top -d 3 sets the delay to three seconds.

Most Linux and Unix variants have a top command that works like we have described. Some have interesting interactive hot keys that allow you to toggle information on or off, sort the list, and even change to a colored display. You should consult the manual page for the top command specific to your operating system, since the special hot keys and interactive features differ among operating systems.

##### 7.5.1.2. The iostat command

The iostat command gives you different sets of information about your system, including statistics about CPU time, device I/O, and even partitions and network filesystems (NFS). The command is useful for monitoring processes because it gives you a picture of how the system is doing overall related to processes and the amount of time the system is waiting for I/O. [**Figure 7-2**](javascript:moveTo('the_iostat_command-011');) shows an example of running the iostat command on a system with moderate load.

##### Figure 7-2. The iostat command

***Note:***

The iostat*,* mpstat*,* and sar commands might not be installed on your system by default, but they can be installed as an option. For example, they are part of the sysstat package in Ubuntu distributions. Consult your operating system documentation for information about installation and setup.

[**Figure 7-2**](javascript:moveTo('the_iostat_command-011');) shows the percentages for CPU usage from the time the system was started. These are calculated as averages among all processors. As you can see, the system is running on a dual-core CPU, but only one row of values is given. This data includes the percentage of CPU utilization:

* Executing at the user level (running applications)
* Executing at the user level with nice priority
* Executing at the system level (kernel processes)
* Waiting on I/O
* Waiting for virtual processes
* Idle time

A report like this can give you an idea of how your system has been performing since it was started. While this means that you might not notice periods of poor performance (because they are averaged over time), it does offer a unique perspective on how the processes have been consuming available processing time or waiting on I/O. For example, if %idle is very low, you can determine that the system was kept very busy. Similarly, a high value for %iowait can indicate a problem with the disk. If %systemor %nice is much higher than %user, it can indicate an imbalance of system and prioritized processes that are keeping normal processes from running.

##### 7.5.1.3. The mpstat command

The mpstat command presents much of the same information as iostat for processor time, but splits the information out by processor. If you run this command on a mul⁠tipro⁠cessor system, you will see the percentage of data per processor as well as the totals for all processors. [**Figure 7-3**](javascript:moveTo('the_mpstat_command-012');) shows an example of the mpstat command.

##### Figure 7-3. The mpstat command

There is an option to tell the mpstat command to refresh the information based on an interval passed. This can be helpful if you want to watch how your processors are performing with respect to the processes over a period of time. For instance, you can see whether your processor affinity is unbalanced (too many processes are assigned to one specific processor).

To find out more about the mpstat command, consult the manual page for your operating system.

##### 7.5.1.4. The ps command

The ps command is one of those commands we use on a daily basis but never take the time to consider its power and utility. This command gives you a snapshot of the processes running on your system. It displays the process ID, the terminal the process is running from, the amount of time it has been running, and the command used to start the process.

What makes the ps command so versatile is the number of options available for displaying data. You can display the processes for a specific user, get related processes for a specific process by showing its process tree, and even change the format of the output. Consult your documentation for information about the options available on your operating system.

One of the ways you can use this output to diagnose problems is to look for processes that have been running for a long time or check process status (for instance, check those that are stuck in a suspicious state or sleeping). Unless they are known applications like MySQL, you might want to investigate why they have been running for so long.

[**Figure 7-4**](javascript:moveTo('the_ps_command-013');) shows an abbreviated example of the ps command run on a system under moderate load.

##### Figure 7-4. The ps command

Another use for the output is to see whether there are processes that you do not recognize or a lot of processes run by a single user. Many times this indicates a script that is spawning processes, perhaps because it has been set up improperly, and can even indicate a dangerous security practice.

Perhaps the most common way some use the ps command is to determine the process IDs for a given program. For example, if you want to know the process IDs of all of the mysqld programs, issue the command:

ps -A | grep mysqld

This will send the list of all processes to the grep command, which will in turn only show those rows with “mysqld” in them. You can use this technique to find a process ID so you can get detailed information about that process using other commands.

There are many other utilities built into operating systems to display information about processes. As always, a good reference on performance tuning for your specific operating system will be the best source for more in-depth information about monitoring processes.

#### 7.5.2. Memory Usage

Several commands provide information about memory usage on your system. The most popular ones include free and pmap.

##### 7.5.2.1. The free command

The free command shows you the amount of physical memory available. It displays the total amount of memory, the amount used, and the amount free for physical memory, and it displays the same statistics for your swap space. It also shows the memory buffers used by the kernel and the size of the cache.[**Figure 7-5**](javascript:moveTo('the_free_command-014');) shows an example of free run on a system with a moderate load.

##### Figure 7-5. The free command

***Note:***

In the output from an Ubuntu system, shown in [**Figure 7-5**](javascript:moveTo('the_free_command-014');), the shared column is obsolete.

There is a switch that puts the command into a polling mode where the statistics are updated for the number of seconds provided. For example, to poll memory every five seconds, issue free -t -s 5.

##### 7.5.2.2. The pmap command

The pmap command gives you a detailed map of the memory used for a process. To use this command, you must first find the process ID for the process you want to explore. You can get this information using the ps command, or even the top command if you are looking at a process that is consuming lots of CPU time.

You can also get the memory map of multiple processes by listing the process IDs on the command line. For example, pmap 12578 12579 will show the memory map for process IDs 12578 and 12579.

The output shows a detailed map of all of the memory addresses and the sizes of the portions of memory used by the process at the instant the report was created. It displays the command used to launch the process, including the full path and parameters, which can be very useful for determining where the process was started and what options it is using. You’d be amazed how handy that is when trying to figure out why a process is behaving abnormally. The display also shows the mode (access rights) for the memory block. This can be useful in diagnosing interprocess issues. Figures [**Figure 7-6**](javascript:moveTo('the_pmap_commandmpart_1');)and [**Figure 7-7**](javascript:moveTo('the_pmap_commandmpart_2');) show an example of a mysqld process map when running on a system with moderate load.

##### Figure 7-6. The pmap command—part 1

##### Figure 7-7. The pmap command—part 2

Notice that the listing chosen is the device output format (selected by issuing the -d parameter on startup) as well as where the memory is being mapped or used. This can be handy in diagnosing why a particular process is consuming lots of memory and which part (e.g., a library) is consuming the most.

[**Figure 7-7**](javascript:moveTo('the_pmap_commandmpart_2');) shows the final line of the pmap output, which displays some useful summary information.

The final line shows how much memory is mapped to files, the amount of private memory space, and the amount shared with other processes. This information may be a key piece of data needed to solve memory allocation and sharing issues.

There are several other commands and utilities that display information about memory usage (e.g.,dmesg, which can display messages from bootup); consult a good reference on performance tuning for your operating system.

#### 7.5.3. Disk Usage

A number of commands can reveal the disk usage statistics on your system. This section describes and demonstrates the iostat and sar commands.

##### 7.5.3.1. The iostat command

As you have already seen in [**Section 7.5.1**](javascript:moveTo('process_activity');), the iostat command shows the CPU time used and a list of all of the disks and their statistics. Specifically, iostat lists each device, its transfer speed, the number of blocks read and written per second, and the total number of blocks read and written. For easy consultation, [**Figure 7-8**](javascript:moveTo('the_iostat_command-016');) repeats [**Figure 7-2**](javascript:moveTo('the_iostat_command-011');), which is an example of the iostat command run on a system with a moderate load.

##### Figure 7-8. The iostat command

This report can be very important when diagnosing disk problems. At a glance, it can tell you whether some devices are being used more than others. If this is the case, you can move some processes to other devices to reduce demand for a single disk. The output can also tell you which disk is experiencing the most reads or writes—this can help you determine whether a particular device needs to be upgraded to a faster one. Conversely, you can learn which devices are underutilized. For example, if you see that your shiny new super-fast disk is not being accessed much, it is likely that you have not configured the high-volume processes to use the new disk. On the other hand, it could be that your program is using memory caches that I/O is seldom performed on.

##### 7.5.3.2. The sar command

The sar command is a very powerful utility that displays all sorts of information about your system. It records data over time and can be configured in a variety of ways, so it can be a little tricky to set up. Consult your operating system’s documentation to ensure you have it set up correctly. Like most of the system utilization commands we show, you can also configure sar to generate reports at regularintervals.

***Note:***

The sar command can also display CPU usage, memory, cache, and a host of other data similar to that shown by the other commands. Some administrators set up sar to run periodically and cull the data to form a benchmark for their system. A complete tutorial on sar is beyond the scope of this book. For a more detailed examination, please see *System Performance Tuning* by Gian-Paolo D. Musumeci and Mike Loukides (O’Reilly, [**http://oreilly.com/catalog/9780596002848/**](http://oreilly.com/catalog/9780596002848/)).

In this section, we will look at how to use the sar command to display information about disk usage. We do this by combining displays of the I/O transfer rates, swap space and paging statistics, and block device usage. [**Figure 7-9**](javascript:moveTo('the_sar_command_for_disk_usage');) shows an example of the sar command used to display disk usage statistics.

##### Figure 7-9. The sar command for disk usage

The report displays so much information that it seems overwhelming at first glance. Notice the first section after the header. This is the paging information that displays the performance of the paging subsystem. Below that is a report of the I/O transfer rates, followed by the swap space report and then a list of the devices with their statistics. The last portion of the report displays averages calculated for all parameters sampled.

The paging report shows the rate of pages paged in or out of memory, the number of page faults per second that did not require disk access, the number of major faults requiring disk access, and additional statistics about the performance of the paging system. This information can be helpful if you are seeing a high number of page faults (major page faults are more costly), which could indicate too many processes running. Large numbers of major page faults can cause disk usage problems. That is, if this value is very high and disk usage is high, it may not be that the poor performance of the disk is just a symptom of something going wrong in the application or operating system.

The I/O transfer report shows the number of transactions per second (tps), the read and write requests, and the totals for blocks read and written. In this example, the system is not using I/O but is under heavy CPU load. This is a sign of a healthy system. If the I/O values were very high, we would suspect one or more processes of being stuck in an I/O-bound state. For MySQL, a query generating a lot of random disk accesses or tables that reside across a fragmented disk could cause such a problem.

The swap space report shows the amount of swap space available, how much is used, the percentage used, and how much cache memory is used. This can be helpful in indicating a problem with swapping out too many processes and, like the other reports, can help you determine whether the problem lies in your disks and other devices or with memory or too many processes.

The block device (any area of the system that moves data in blocks like disk, memory, etc.) report shows the transfer rate (tps), the reads and writes per second, and average wait times. This information can be helpful in diagnosing problems with your block devices. If these values are all very high (unlike this example, which shows almost no device activity), it could mean you have reached the maximum bandwidth of your devices. However, this information should be weighed against the other reports on this page to rule out a thrashing system, a system with too many processes, or a system without enough memory (or a combination of such problems).

This composite report can be helpful in determining where your disk usage problems lie. If the paging report shows an unusually high rate of faults, it’s an indication you may have too many applications running or not enough memory. However, if these values are low or average, you need to look to the swap space; if that is normal, you can examine the device usage report for anomalies.

##### 7.5.3.3. Disk Usage Analyzer

In addition to operating system utilities, the GNOME desktop project has created a graphical application called the Disk Usage Analyzer. This tool gives you an in-depth look at how your storage devices are being used. It also gives you a graphic that depicts disk usage. The utility is available in most distributions of Linux.

[**Figure 7-10**](javascript:moveTo('disk_usage_analyzer-017');) shows a sample report from the Disk Usage Analyzer.

##### Figure 7-10. Disk Usage Analyzer

Basically, this report gives you a look at how the devices are performing alongside the paging and swap systems. Naturally, if a system is swapping a lot of processes in and out of memory, the disk usage will be unusual. This is why it is valuable to look at these items together on the same report.

Diagnosing disk problems can be challenging, and only a few commands give you the kind of detailed statistics about disk usage we’ve described. However, some operating systems provide more detailed and specific tools for examining disk usage. Don’t forget that you can also determine available space, what is mounted, which filesystems each disk has, and much more from more general commands such as ls*,* df, and fdisk. Consult your operating system documentation for a list and description of all disk-related commands, as well as for disk usage and monitoring commands.

***Note:***

The vmstat command, shown later in this chapter, can also show this data. Use the vmstat -dcommand to get a text-based representation of the data.

#### 7.5.4. Network Activity

Diagnosing network activity problems may require specialized knowledge of hardware and networking protocols. Detailed diagnostics are normally left to the networking specialists, but there are two commands you, as a MySQL administrator, can use to get an initial picture of the problem.

##### 7.5.4.1. The netstat command

The netstat command allows you to see network connections, routing tables, interface statistics, and additional networking-related information. The command provides a lot of the information that a network specialist would use to diagnose and configure complex networking problems. However, it can be helpful to see how much traffic is passing through your network interfaces and which interfaces are being accessed the most. [**Figure 7-11**](javascript:moveTo('the_netstat_command-018');) shows a sample report of all of the network interfaces and how much data has been transmitted over each one.

##### Figure 7-11. The netstat command

In systems that have multiple network interfaces, it may be helpful to determine whether any interface is being overutilized or if the wrong interfaces are active.

##### 7.5.4.2. The ifconfig command

The ifconfig command, an essential tool for any network diagnostics, displays a list of the network interfaces on your system, including the status and settings for each. [**Figure 7-12**](javascript:moveTo('the_ifconfig_command-019');) shows an example of the ifconfig command.

##### Figure 7-12. The ifconfig command

Notice how each interface, whether it is up or down, is listed, along with its configuration information. This can be very helpful in determining how an interface is configured and can tell you, for example, that instead of communicating over your super-fast Ethernet adapter, your network has failed over to a much slower interface. Many times the root of networking problems is not the traffic on the network, but rather the network interface choice or setup.

If you produce the reports shown here for your system and still need help diagnosing the problem, having this data ahead of time can help your networking specialist zero in on the problem more quickly. Once you have eliminated any processes consuming too much network bandwidth and determined where you have a viable network interface, the networking specialist can then configure the interface for optimal performance.

#### 7.5.5. General System Statistics

Along with the subsystem-specific commands we’ve discussed, and grouped statistical reporting commands, Linux and Unix offer additional commands that give you more general information about your system. These include commands such as uptime and vmstat.

##### 7.5.5.1. The uptime command

The uptime command displays how long a system has been running. It displays the current time; how long the system has been running; how many users have been using the system (logged on); and load averages for the past 1, 5, and 15 minutes. [**Figure 7-13**](javascript:moveTo('the_uptime_command-020');) shows an example of the command.

##### Figure 7-13. The uptime command

This information can be helpful if you want to see how the system has been performing on average in the recent past. The load averages given are for processes in an active state (not waiting on I/O or the CPU). Therefore, this information has limited use for determining performance issues, but can give you a general sense of the health of the system.

##### 7.5.5.2. The vmstat command

The vmstat command is a general reporting tool that gives you information about processes, memory, the paging system, block I/O, disk, and CPU activity. It is sometimes used as a first stop on a quest for locating performance issues. High values in some fields may lead you to examine those areas more closely using other commands discussed in this chapter.

[**Figure 7-14**](javascript:moveTo('the_vmstat_command-021');) shows an example of the vmstat command run on a system with low load.

The data shown here includes the number of processes, where r indicates those waiting to run and bindicates those in an uninterruptible state. The next set of columns shows the swap space totals including amount of memory swapped in (si) or out (so). The next area shows the I/O reports for blocks received (bi) or sent (bo). The next area shows the number of interrupts per second (in), number of context switches per second (cs), time spent running processes in user space (us), time spent running processes in kernel space (sy), idle time (id), and time waiting for I/O (wa). These times are all in seconds.

There are more parameters and options for the vmstat command. Check your operating system manual for more details on the options available for your operating system.

##### Figure 7-14. The vmstat command

#### 7.5.6. Automated Monitoring with cron

Perhaps the most important tool to consider is the cron facility. As described in the section [**Section 2.3.1.2**](http://proquest.safaribooksonline.com.ezproxy.torontopubliclibrary.ca/9780596807313/performing_common_tasks_with_replication#scheduling_tasks_on_unix), you can use cron to schedule a process to run at a specific time. This allows you to run commands and save the output for later analysis. It can be a very powerful strategy, allowing you to take snapshots of the system over time. You can then use the data to form averages of the system parameters, which you can use as a benchmark to compare to when the system performs poorly in the future. This is important because it allows you to see at a glance what has changed, saving you considerable time when diagnosing performance problems.

If you run your performance monitoring tools daily, and then examine the results and compare them to your benchmark, you may be able to detect problems before your users start complaining. Indeed, this is the basic premise behind the active monitoring tools we’ve described.