Introduction to Performance Management

Application developers and system administrators face similar challenges in managing the performance of a computer system. Performance management starts with application design and development and migrates to administration and tuning of the deployed production system or systems. It is necessary to keep performance in mind at all stages in the development and deployment of a system and application. There is a definite overlap in the responsibilities of the developer and administrator. Sometimes determining where one ends and the other begins is more difficult when a single person or small group develops, administers and uses the application. This chapter will look at:

- Application developer’s perspective
- System administrator’s perspective
- Total system resource perspective
- Rules of performance tuning

1.1 Application Developer’s Perspective

The tasks of the application developer include:

- Defining the application
- Determining the specifications
- Designing application components
- Developing the application codes
- Testing, tuning, and debugging
- Deploying the system and application
- Maintaining the system and application
1.1.1 Defining the Application

The first step is to determine what the application is going to be. Initially, management may need to define the priorities of the development group. Surveys of user organizations may also be carried out.

1.1.2 Determining Application Specifications

Defining what the application will accomplish is necessary before any code is written. The users and developers should agree, in advance, on the particular features and/or functionality that the application will provide. Often, performance specifications are agreed upon at this time, and these are typically expressed in terms of user response time or system throughput measures. These measures will be discussed in detail later.

1.1.3 Designing Application Components

High-level design should be performed before any code is actually written. Performance should begin to play a part at this point in the development process. Choices must be made as to what operating systems services are to be used. Trade-offs such as ease of coding versus algorithm efficiency appear to be easily made at this stage, but they can often negatively affect performance once the application is deployed.

Not considering performance at this phase of the life cycle condemns the application to a lifetime of rework, patches, and redesign. The designer should also be considering ways to design instrumentation for transaction rate and other application-specific measurements that cannot be made by general-purpose tools.

1.1.4 Developing the Application Codes

After high-level design is complete, the developer creates the actual codes in one or more computer languages. Codes are based on the application specifications, including performance specifications, developed in conjunction with users.

1.1.5 Testing, Tuning, and Debugging

All too often, performance does not become an issue until the application has been written and is in the testing phase. Then it quickly becomes apparent that the application performs poorly or will not meet the response or throughput requirements of the user.

1.1.6 Deploying the System and/or Application

If not discovered sooner, performance deficiencies normally become quite apparent once the system and/or application is deployed out to the user community. Complaints that the performance is poor may be frequent and loud once the system is fully operational.


1.1.7 System and Application Maintenance

Once the system is operational, the developers work on making changes in the application, which may include adding new features, fixing bugs, and tuning for performance. It is often too late to have a big impact on performance by making slight modifications to the application. Developers may have to consider large-scale application redesign in order to meet performance specifications. This area begins the overlap between the duties of the developer and the system administrator.

1.2 System Administrator’s Perspective

The tasks of the system administrator include:

• Making the system available to others
• Monitoring the usage of the system
• Maintaining a certain level of performance
• Planning for future processing needs

1.2.1 Making the System Available to the Users

This may include such responsibilities as loading the system and application software, configuring devices, setting up the file system, and adding user and group names.

1.2.2 Monitoring the Usage of the System

Measuring the utilization of various system resources is done for several reasons. Monitoring CPU and disk utilization can provide the basis for chargeback accounting. Identifying and documenting the various applications being used allows management to understand the trends in resource usage within the system. And of course, monitoring is the quickest way to detect problems.

1.2.3 Maintaining a Certain Level of System Performance

System administrators often see performance tuning as a response to user complaints. After monitoring and analyzing system performance, administrators must take appropriate action to ensure that performance returns to acceptable levels. System or application tuning may be necessary to improve responsiveness.

1.2.4 Planning for Future Processing Needs

This involves predicting the size of systems needed either for new applications or for growth of existing applications.

Almost any change to a system or application will have some impact on performance. Proposed changes must be weighed against the potential resulting change in performance. Performance management is the process of measuring, evaluating, and modifying the level of
performance that a system and application provide to their users, and assessing the effect of proposed changes. It is important to understand that performance management is a proactive and not a reactive function. It is not very pleasant to be told by either the users or management that the computer system is not living up to expectations, and that you, the system administrator or developer, need to do something about it.

1.3 Total System Resource Perspective

To understand performance management, it is necessary to understand the various system resources that interact to affect the overall performance of the system. Traditionally, system administrators viewed performance as the interaction of three system resources: CPU, memory, and disk. Figure 1-1 shows this view. The Central Processing Unit, or CPU, is the hardware that executes the program codes, which exist as one or more processes. Each process may run as one or more threads, depending on the operating system (OS) version or variant. The part of the OS that controls the CPU resource is the process management subsystem, which handles the scheduling of the processes that want to use the CPU.

Memory is the part of the computer that stores the instructions and data. Physical memory is typically semiconductor Random Access Memory (RAM). Virtual memory includes physical memory and also the backing store (which is the term for the overflow area used when memory is fully utilized) that is typically on disk. The Memory Management Subsystem controls this resource. Processes may be swapped and/or paged in and out as demands for memory change.

The disk resource includes all of the hard disk drives attached to the computer system. These disk drives may be of various capacities and interface types. From a performance standpoint, the disk resource involves the movement of data between the computer and the disks.
rather than the utilization of the space on the disk. The File System and I/O Management sub-
systems control the disk resource. Although the disk resource is a specific case of the general
resource known as I/O, the most important I/O resource is the disk, and most of the I/O tuning
that can be done is related to the disk resource.

When one of these resources is completely consumed and is the source of a performance
problem, then we often conclude that there is a bottleneck. For example, if memory is com-
pletely utilized and response is slow, then there is a memory bottleneck. However, reality is
always more complicated. We will see later that it is not sufficient to use 100% utilization as the
sole criterion for defining a resource bottleneck.

Figure 1-1 shows the primary resources: CPU, memory, and disk. The double-sided
arrows between them are meant to emphasize that the resources are interrelated and that reliev-
ing a given bottleneck will often change the system to a different bottleneck. Several years ago,
one of the authors was involved in a performance problem on a database server system. Analysis
showed that the CPU was saturated and was limiting the throughput of the database. After the
system was upgraded with a faster CPU, the performance was limited by the disk subsystem. It
was then necessary to better distribute the tables of the database across multiple disk drives. It is
often the case that one system resource that is bottlenecked can mask another resource problem.

1.4 Rules of Performance Tuning

Here is the first of several “Rules of Performance Tuning.”

RULE 1: When answering a question about computer system
performance, the initial answer is always: “It
depends.”

(These rules will often be referred to by just the number, or the short form. So, here we
would say: “Rule #1” or “It depends.”) The following shows the application of the rule in a typ-
ical consulting situation.

From Bob’s Consulting Log—On arriving at a customer site for a performance consul-
tation, one of the first questions we were asked was, “What will happen if I add 20
users to my system?” The answer was, of course, “It depends!” Implicit in this ques-
tion, though unspoken, was the fact that performance was actually satisfactory at the
time. But the customer did not know whether any resources were at or near the satu-
ration point.

Of course, when you say, “It depends,” you have to qualify the dependencies—what does
good performance for added users depend on? Does each extra user require additional memory,
or does the application support additional users without requiring more memory? Will more data
storage be needed for the new users? This book aims to show you how to understand what the dependencies are, to ask the right questions, to investigate, to weigh the dependencies, and then to take appropriate action.

The weighing of dependencies brings us to the second “Rule of Performance Tuning.”

**RULE 2: Performance tuning always involves a trade-off.**

When an administrator tries to improve the performance of a computer system, it may be necessary to modify operating system parameters, or change the application itself. These modifications are commonly referred to as performance tuning, that is, tuning the system kernel, or tuning the application. Tuning the kernel involves changing the values of certain tunable parameters in what is called the system file followed by relinking the kernel and rebooting the system. Application tuning involves adjusting the design of the application, employing the various compiler optimization techniques, and choosing the appropriate trade-offs among the various kernel services that will be discussed more fully in Part 4. Application tuning also may involve adjusting any tunable parameters of the application itself, if any are available.

System tuning requires an understanding of how the operating system works, and a knowledge of the dependencies that affect performance. Choosing a particular modification always involves trading off one benefit for another. For example, tuning to improve memory utilization may degrade file system performance. Understanding the needs of the application and this interaction between memory and file system permits the selection of the appropriate trade-off. Cost or risk may also be involved in the trade-off. Choosing RAID disk configurations for data integrity may be less expensive than alternative mirroring solutions that often improve performance. It may be more cost-effective to purchase a CPU upgrade rather than spend days or weeks analyzing how the application could be changed to improve performance.

In the case of a bottleneck, it is necessary to strike a balance between the demand for a resource and the availability of that resource. In order to relieve a given bottleneck, we must either increase the availability of a resource or decrease demand for it. Figure 1-2 shows this balance. A more complex view analyzes the demand for and availability of resources over time. Shifting some of the demand for a resource from a high demand time period to one of lower demand is a way of balancing resource and demand without reducing the total demand. One way to view this graphically is shown in Figure 1-3.

The traditional view of system resources is, however, incomplete. HP-UX systems are often used in a distributed environment where many computer systems are joined in a cooperative network. Other important resources, such as the network and graphics, are not included in the traditional view.
Figure 1-2  Balancing Resources and Demands

Figure 1-3  Balancing Demand Over Time
A fuller view of system resources is shown in Figure 1-4. Again, the double-sided arrows denote that all the resources are intertwined, and that relieving one bottleneck will likely result in a new one.

![Figure 1-4 Complete View of System Resources](image)

The general I/O resource, which includes direct-connected terminal I/O and magnetic tape, is not included in Figure 1-4. Network-connected terminals, such as those connected via a terminal server, are included in the network resource. However, network-connected terminals use the same *termio* subsystem as direct-connected terminals. The *termio* subsystem can consume a large amount of the CPU resource to process the characters as they are transmitted from the terminals.

The network resource includes system-to-system connections, network-connected terminals, and other smarter clients such as PCs. Network transports such as Ethernet, SNA, and X.25 are included here. Applications that consume this resource include virtual terminal, file transfer, remote file access, electronic mail, X/Windows or Motif applications, and networked inter-process communication (IPC) methods such as sockets and remote procedure calls (RPCs).

Client/Server applications are very complex. One must consider the traditional resource consumption on the server and client systems, plus the network resource consumption and latency involved in communication between the client and server programs.

Kernel resources are system tables that provide the mechanism for file access, intrasystem interprocess communication (IPC), and process creation.

The graphics resource typically involves specialized hardware and software that provides graphics capabilities. This resource will not be discussed in detail in this book. The tuning possibilities for the graphics resource are limited to upgrading to a faster CPU, upgrading to faster
graphics hardware, or tuning the specialized graphics software, for which the typical user does not have the source code (X-11, PHIGS, etc.).

It is useful to look at each of these resources separately for the purpose of analysis. The various performance monitoring tools, which will be discussed in Part 3, provide data from a particular set of metrics, and may not provide data for all the system resources shown above. It will be demonstrated that no one tool provides all the data needed to monitor overall system performance.

An important aspect of resource analysis is that it is invasive, as shown in another rule.

**RULE 3: Performance cannot be measured without impacting the system that is being measured.**

Rule #3 is a loose interpretation of the Heisenberg principle. Physicist Werner Heisenberg discovered that he could not measure both the speed and the position of an atomic particle, since the act of making the measurement perturbed the system that was being measured. This is also true of computer systems. The following is an example from Bob’s consulting experience.

*From Bob’s Consulting Log—I was called in to evaluate a database server system used for OLTP. There were several hundred users on the system, and memory was configured at 128 MB. There had been complaints of poor performance. I did preliminary measurements with `vmstat` and `glance`. Then I started using `gpm`, the graphical version of `glance`, for more detailed measurements. The system started thrashing and exhibiting a very different behavior: application processes were being starved of CPU. The system was already at the verge of a severe virtual memory problem, and I pushed it over the edge by running `gpm`, which consumes much more virtual memory than the other tools (over 5 MB).*

Making measurements consumes resources. The level of resource consumption caused by performance data collection and presentation depends upon:

- The type, frequency, and number of measurements
- Data presentation method (text and/or graphics)
- Whether the data is being stored to disk
- The source of the data (to be discussed later)
- General activity on the system (the busier the system is, the more data is generated)

Different tools consume varying amounts of memory, CPU, and disk bandwidth (if the data is being recorded in a file). It is important to realize that it is not possible to collect performance data without consuming resources and even changing the *system* that is being measured. There may not be sufficient system resources available to run the tool. The added overhead due
to using the tools may push the system over the edge. In that case, what is being measured is no longer the same system. For instance, if memory consumption is already very high, and the tool requires a large amount of memory just to run, memory thrashing may occur if it is not already occurring, and change the characteristics of what is being measured. The “Catch-22” with this situation is that the poorer the system’s performance, the more measurements must be made to provide information about what the system is doing. One must decide what level of impact is acceptable in order to acquire the necessary information.

Finally, there is no single tool for all your needs.

**RULE 4: No single performance tool will provide all the data needed to solve performance problems.**

The perfect performance tool that solves all performance problems in all situations simply does not exist. It is necessary to develop familiarity with multiple tools so that all needs can be met. In fact, it is sometimes useful to cross-check measurements between the various tools to ensure that the tool itself is reporting information correctly. In other words, if the data doesn't make sense, question the tool!