As companies deploy mission-critical applications across distributed client/server (C/S) environments, performance management complexity becomes a major obstacle. Today's distributed environments are not only more technologically diverse and geographically dispersed than in the past, they are also more dynamic. Managing performance and availability across many application, system, and network components in these complex environments is required to minimize the impact to your business. Response time delays and availability problems could cost your company its competitive edge.

One of the main goals of C/S performance management is to ensure that your distributed applications and systems have sufficient performance and availability to improve productivity and achieve a competitive advantage. This article describes 10 key aspects to successful C/S performance management. Managing C/S performance requires understanding organizational and technical aspects of performance management and applying Software Performance Engineering (SPE) principles to C/S application design and deployment. See Figure 1. The original seven SPE principles are described by Connie Smith in her book, Performance Engineering of Software Systems. The Objectives principle was added by Chris Loosley in his article "Applying Performance Principles to Client/Server Design." The Management principle is added in this article.

**THE COMPLEXITIES OF DISTRIBUTED ENVIRONMENTS**

Geographically distributed systems and applications spanning multiple organizations, time zones, and legal and political boundaries comprise today's complex computing environments. Inter-organizational communication, politics, and inter-departmental rivalry and distrust add to the complexity. Different vendors, technologies, protocols, data stores, user interfaces, and system administration tools must coexist within a distributed environment. Scalability is a major issue given that hundreds or even thousands of components need to be managed. These dynamic working environments and high rates of technological change demand skilled personnel to support multiple technologies and platforms at many sites.

**GOALS OF PERFORMANCE MANAGEMENT**

Performance management, and specifically SPE, have many goals. Typical goals include:

- ensuring adequate performance and availability of systems and applications;
- identifying the causes of unacceptable performance in existing systems;
- planning hardware capacity sufficient for current and future workloads;
- tracking workloads to manage capacity and justify upgrades;
- tuning to avoid or reduce the need for hardware and software upgrades; and
- maintaining control of system and network resources to ensure adequate performance for the most critical users and workloads.

Reaching these goals requires clearly understood business needs and priorities. Can you identify the system design components most likely to offer adequate performance and availability? A key challenge in performance management is providing a predictable service level despite resource contention bottlenecks.

Performance management is a complex issue for applications being deployed across distributed client/server environments. The 10 keys presented in this article can be used to help reach the goals of client/server management successfully.
Do you know what tools and procedures to use to identify and resolve performance bottlenecks in an automated fashion? The 10 keys that follow will help you answer these questions and successfully reach the goals of C/S performance management.

**KEYS TO MANAGING CLIENT/SERVER PERFORMANCE**

The Management principle states that managing performance is required to minimize performance problems or system outages.

<table>
<thead>
<tr>
<th>Principle</th>
<th>Statement of Principle</th>
<th>Development Phases</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Management</td>
<td>Performance must be managed</td>
<td>All phases</td>
</tr>
<tr>
<td>2. Objectives</td>
<td>Set performance goals</td>
<td>Planning, Analysis</td>
</tr>
<tr>
<td>3. Centering</td>
<td>Think globally, focus locally*</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td>4. Workload</td>
<td>Minimize the total processing load</td>
<td>Analysis, Design, Construction</td>
</tr>
<tr>
<td>5. Overhead</td>
<td>Minimize overhead costs</td>
<td>Logical and Physical Design</td>
</tr>
<tr>
<td>6. Locality</td>
<td>Optimize the distribution of work</td>
<td>Logical and Physical Design</td>
</tr>
<tr>
<td>7. Parallelism</td>
<td>Exploit parallel processing</td>
<td>Physical Design</td>
</tr>
<tr>
<td>8. Sharing</td>
<td>Use shared resources carefully</td>
<td>Physical Design</td>
</tr>
<tr>
<td>9. Instrumentation</td>
<td>Make sure you can measure it</td>
<td>Design, Construction, Deployment</td>
</tr>
</tbody>
</table>

*always work on the largest bottlenecks first


**COORDINATE THE PARADIGM SHIFT TO CLIENT/SERVER**

In the past, centralized systems were centrally managed by a few technical specialists. Today’s more complex, distributed environments share some of the same management requirements as centralized systems, but also have their own special requirements. For example, technical support needs to be distributed to manage geographically dispersed systems. Architects and tools need to provide a unified, integrated view of distributed, heterogeneous systems and applications to pinpoint and correct performance and availability problems.

Each generation of systems and technology brings with it new organizational paradigms, new design issues and techniques, increased complexity of keeping track of all the pieces in the growing distributed puzzle, and inevitably, more tradeoffs. With fewer right answers and more compromises, management of distributed environments requires flexible, adaptable, and technologically competent personnel at central and remote locations. Skilled people, beyond the "mini-expert" with expertise in a certain area or on a particular platform, are hard to find in an industry buried in an avalanche of new technology. To succeed, companies must provide sufficient education resources and build teams motivated to keep up with fast changing new technologies and methodologies.

**CHOOSE THE RIGHT OS AND PLATFORM**

Most distributed environments are made up of a wide array of client and server hardware platforms. Factors to consider when selecting C/S hardware and software include:

- price/performance ratio;
- OS software supported by the hardware;
- DBMS capabilities;
- exploitation of hardware by the OS and DBMS;
- development;
- end-user and administrative tools;
- networking hardware and software; and
- the middleware required to connect all the pieces together.

Most database servers support workgroup and departmental C/S systems. Larger distributed applications accessing mission-critical enterprise databases demand servers with key requirements such as high performance and...
scalability. Ever-increasing database sizes, transaction and query rates, and newer technologies such as data warehousing and Online Analytical Processing (OLAP) systems increase the demand for Symmetric Multi Processor (SMP) and Massively Parallel Processor (MPP) machines.

Many essential OS and DBMS features, such as portability, scalability, multitreading, preemptive scheduling, and protected and virtual memory are needed to support C/S processing. Multiprocessor support to enable scalability from a uniprocessor to multiprocessor hardware platforms is required to exploit parallel processing. Some OSes allow preemptive scheduling through assignment of relative priorities to application processes. This allows the OS scheduler to interrupt and suspend an active application task if another with a higher priority needs to use the processor.

How well the DBMS exploits specific hardware and OS features is an important performance factor in high volume environments. Does the DBMS scale across different SMP and MPP configurations? Does it fail to exploit important features, such as OS threads, asynchronous I/O, or buffering capabilities? These features are described in detail in an article by Colin White (see “References”).

In the world of downsizing and rightsizing, cost is also a major factor when choosing a hardware and OS vendor. Don’t jump at a low-end price without understanding the future consequences. In a world where increasing memory and CPU requirements are as inevitable as death and taxes, beware of vendors who design their low-end product lines and pricing so that adding memory or CPUs requires upgrading to a more expensive platform.

**MONITOR PERFORMANCE WITH AVAILABLE TOOLS**

Performance monitors have been an important management and diagnostic tool in the mainframe environment for more than 15 years. Distributed C/S applications create new demands for performance data collection, analysis, and management that is consistent across multiple OSs and platforms.

To understand why a distributed application is performing poorly, performance and availability data must be collected at a number of different levels, as depicted in Figure 3. Many two-tier systems and the increasingly popular three-tier C/S applications dictate that performance tools follow transaction flow from the desktop, through one or more servers and networks, and back again. The ability to map transaction topology flow and identify resource bottlenecks along the way is critical to pinpointing problem areas. This is a challenging goal for vendors faced with the rapid introduction of new technology to the C/S environment and the difficulty of collecting and correlating performance data across many components.

According to the SPE Instrumentation principle and the closely related Objectives principle, you must be able to measure performance in order to manage it. Measuring performance tells you if an application needs tuning, how to tune it, and when to stop tuning. You also need to monitor the most probable points of resource contention, including the server OSes, server DBMSes, and network components. Until distributed end-to-end mapping of applications is provided by vendors, your distributed applications need some level of instrumentation built into them for tracking critical activities and problems.

**ADDRESS PERFORMANCE EARLY IN THE LIFE CYCLE**

A critical factor in the success of any new distributed C/S application is how well the designers and developers focus on performance throughout the different development phases of planning, analysis, design, construction, and production deployment. The complexity of building distributed applications increases the need for quantifiable performance goals.

The Objectives principle mandates that performance objectives be established during the earliest stages of development, planning, and analysis. Determine transaction priorities, execution frequencies, and response time objectives. Predict resource utilization and response times and modify physical design choices as necessary. These steps help reduce future performance problems and the costs associated with fixing them, and improve the overall quality of your distributed applications.

During construction, monitor individual components to uncover unforeseen problem areas. During production deployment, validate prior predictions and track actual performance and usage of critical system components. Throughout each development phase, look beyond what the application is supposed to do and discover why, when, and how often different components are executed. As the Workload principle states, the goal is to reduce unnecessary processing.

**DISTRIBUTE WORK EFFICIENTLY**

How well a C/S application divides the work or processing between clients and server(s), as well as the amount of communication traffic between the components, can easily become dominant performance factors in a distributed application. Although the bulk of application code is traditionally placed on a “fat client”, fat server applications are easier to manage and deploy because they focus tuning efforts on a few servers instead of hundreds or thousands of clients. Fat server applications are designed with most of the business logic and knowledge of data organization on the server side. The client provides the user interface and input validation, interacting with the server via remote procedure calls.

Locality is an important aspect of distributed application design because it affects the amount of communication traffic between components. The Locality principle calls for the creation of actions, functions, and results that are close to physical resources, such as performing all user interface logic on the client machine with little or no involvement of LAN or WAN communication.

To further enhance performance by reducing communication traffic, carefully replicate data to distribute it closer to users; this helps reduce the number of applications that assemble their data at execution time from databases at separate locations. Predict and monitor capacity and workload demands for client, server, and network components. This allows you to quickly identify resource constraints and either increase their capacity or repartition the workload across your distributed systems.

**MINIMIZE CONTENTION**

Response time has two basic components: time spent using resources (CPU, I/O) and time spent waiting for resources to become available. This makes sharing of database and machine resources another important performance factor. As stated in the Sharing principle,
carefully evaluate how critical resources are shared by competing tasks and users.

Although lock contention during data access has the most obvious effect on application response time, other types of contention also have an impact:

- Latch contention for internal server resources and structures may exist but can be difficult to identify and tune.
- I/O contention results when capacity constraints are exceeded or when load balancing across available disks is inadequate.
- CPU contention impacts performance when one or more CPUs are overloaded and higher priorities cannot be assigned to more critical processes.
- Connection limits can create contention as client requests compete for access to the server(s); requests may wait or fail when trying to access the server during peak periods.

According to the Centering principle, identify the largest bottlenecks across your systems and applications. Focusing on your most important workloads and largest bottlenecks allows you to concentrate your limited tuning resources on the most critical areas.

In some cases, the Parallelism principle can be used to break performance bottlenecks. But not all software or workloads are designed to work in parallel. As Smith states in her book, "Execute processing in parallel (only) when the processing speedup offsets communication overhead and resource contention delays." You must determine to what extent the application logic, server OS, middleware, and database server software can exploit parallel processors without dramatically increasing the amount of process management overhead and lock and latch contention.

**DESIGN AN EFFICIENT DATABASE**

An application that performs well in development and testing frequently performs unacceptably in a high data volume production environment. Database application performance is fundamentally a function of the database structure and the size of the files being accessed and modified. In other words, how much data from how many tables must be read to satisfy user requests?

According to the Overhead principle, careful logical and physical design is needed to minimize runtime overhead costs. Major database design and access path issues include the level of data normalization used, the inappropriate use of indexes, and the type of database requests made.

To build high-performance applications in these complex environments, designers and developers must understand the key performance drivers of C/S and distributed database applications. After all, if you can't measure the performance of your applications and systems, you can't manage them.

- **Latch contention for internal server resources and structures may exist but can be difficult to identify and tune.**
- **I/O contention results when capacity constraints are exceeded or when load balancing across available disks is inadequate.**
- **CPU contention impacts performance when one or more CPUs are overloaded and higher priorities cannot be assigned to more critical processes.**
- **Connection limits can create contention as client requests compete for access to the server(s); requests may wait or fail when trying to access the server during peak periods.**

**Data normalization:** The process of restructuring a data model to its simplest form is very important in database systems. However, an inappropriate amount of data normalization during database design can result in significant runtime overhead. Over-normalization can increase the number of multiple table joins and repeated aggregation or grouping of data. Database designers should strive for "Best Normal Form" over "N Normal Form" to achieve good database performance.

- **Indexes:** The inappropriate use of indexes affects the amount of overhead used by the database server to retrieve data. Define the most appropriate indexes and ensure that they are actually used by the database server to reduce I/O and CPU and improve performance for your most critical or frequent database requests. Not all database servers use the same optimizer technology. Some require more "hand holding" by the DBA to force the best access path, the one that minimizes I/O and CPU overhead, while others have fairly advanced cost-based optimizers.

- **Sized compiled SQL:** If remote data access middleware is used to send dynamic SQL requests, query execution time may be unpredictable and dynamic bind overhead at execution becomes a factor. Use sized compiled SQL statements to make query execution time more predictable and help meet performance objectives.

- **Stored procedures:** Also well suited for creating performance-critical applications, stored procedures reduce network traffic and improve response time. They encapsulate multiple sized compiled SQL statements and their processing logic into a program residing at the database server. Some DBMSes support static SQL only through the use of stored procedures and triggers.

**TUNE CONFIGURATION PARAMETERS**

Hardware and software components support various tuning knobs in the form of externalized configuration parameters. Technical support personnel need to be familiar with these parameters, how each relates to performance and tuning, what the benefit/impact ratio is if their values are changed, and whether any can be changed dynamically. Review default parameter values for their usefulness in each environment. Tuning database server, OS, and network parameter values requires analyzing a number of tradeoffs. There are no magic numbers, especially if the real problem is a poorly designed application or database structure.

**ALLOCATE SERVER RESOURCES AND BALANCE I/O**

Disk I/O contention is a major source of bottlenecks on all types of platforms. As databases increase in size exponentially, more and more disk devices are added to support existing and new databases. Balancing I/O across available disk drives is especially difficult in these dynamic environments. Know where the critical access paths are and their expected I/O loads.

When managing physical file and disk resources, the major performance considerations include:

- **Ensuring that database files for tables and indexes are of sufficient size to maintain adequate free space and to minimize the number of extents allocated. Balance file placement across available disks to reduce I/O contention and improve response times and recoverability.**

- **Selecting the most appropriate disk devices and configurations, such as RAID disks and fast vs. slow devices. Other issues include the number of disks available to improve I/O balancing and implementation of disk striping and data mirroring.**

- **Sizing and placing log, catalog, dictionary, archive, and sort work files carefully. Do your checkpoint frequencies meet your performance and recovery requirements?**

**MINIMIZE NETWORK CONGESTION**

LANs are fast but become congested as
competing users and workloads increase. Minimizing network congestion and improving network performance may depend on how distributed applications are partitioned across available clients and servers. The type of message traffic also impacts network performance. For example, executing a stored procedure instead of many individual SQL calls reduces network traffic and improves response times.

Other networking performance management techniques include:

- reducing protocol conversions that add overhead;
- easing server congestion to help decrease network delays;
- using message priorities to prevent large data transfers from blocking smaller but critical transaction messages; and
- adding network capacity where bottlenecks are most apparent and cause the greatest impact to your business applications.

UNDERSTANDING THE KEY PERFORMANCE DRIVERS

Today's organizations must manage and tune a growing number of clients and servers as well as the application workloads that execute across them. This is frequently done without the benefit of centralized control mechanisms or even the organizational authority to fix problems as they are discovered.

To build high-performance applications in these complex environments, designers and developers must understand the key performance drivers of C/S and distributed database applications. After all, if you can't measure the performance of your applications and systems, you can't manage them. If you can't manage your applications and systems, you shouldn't deploy them.

Complex distributed environments with hundreds or thousands of processing components make identifying and recovering unavailable components and databases a real challenge. Generally, the more complicated the architecture, the more difficult it is to manage. The more nodes or layers of protocol conversions, the slower the response time.

Although C/S technology allows "mixing and matching" of parts, try to avoid this. Running the same OS on both client and server makes LANs simpler to administer, and minimizes the variances in monitoring and tuning procedures. Standardize on as few vendors, technologies, protocols, data stores, user interfaces, and system administration tools as you can to reduce complexity and training costs.

REFERENCES


Daniel D. Kitay is an advisory software engineer at Candle Corporation's Westlake, Calif., R&D office. He specializes in designing and implementing applications to manage the performance and availability of distributed systems and applications. For the past 10 years Dan has worked with IMS, VTAM, DB2 for MVS, and more recently, Oracle, Sybase, Lotus Notes and SAP. He can be reached at dan_kitay@candle.com.

©1996 Technical Enterprises, Inc. Reprinted with permission of Technical Support magazine. For subscription information, email mbership@naspa.net or call 414-768-8000, Ext. 116.