Payoff

Client/server technologies, although essential for most business process redesign and reengineering initiatives, introduce new complexities into systems development. Organizations that incorporate performance engineering at an early stage of the decision and design process can improve the performance and cost-effectiveness of client/server systems.

Introduction

As part of an overall strategy to lower its IT costs, an oil and gas company considered deploying its database management system within a client/server environment, thus cutting both hardware expenditures and software licensing fees. The organization was skeptical, however, of the network’s ability to support 150 concurrent users at an acceptable response rate. Long transaction times would reduce the level of customer service yet increase organizational costs for customer service representatives.

To evaluate the proposed system environment, the company devised and conducted a benchmark test to emulate its anticipated workload. This proof-of-concept benchmark demonstrated that the architecture could indeed handle the requisite load within acceptable performance parameters. The company could confidently install the new system without compromising customer service or incurring unexpected expenditures. By incorporating performance engineering at an early stage of the decision and design process, the company achieved its goals with minimal disruption and maximum economy.

What Is Performance Engineering?

Performance engineering models the computing resource consumption of alternative technical architectures and measures their ability to perform business functions. Its aim is to design acceptable levels of performance into an installation’s hardware configuration, network design, online and batch architecture, and data design.

Performance engineering’s two approaches—benchmarking and predictive modeling—allow IT organizations to systematically anticipate component interaction rather than respond to it after installation. Performance engineering is particularly important for large transaction-processing, decision support, and executive information systems. In these large-scale and complex applications, performance problems after rollout are more likely, more troublesome, and more expensive than in simpler settings.

Approaches to Performance Engineering

Benchmarking and predictive modeling are two closely related approaches to performance engineering in both the mainframe and client/server environments. Both may be done at various points in the systems development process. The later they are done, the more the results will reflect real design information as opposed to conjecture and
approximation—but the cost of change will be greater. Both approaches also require approximations of application transactions volumes, data base use by each transaction, and data base structures.

**Benchmarking.**

Benchmarking is the process of executing a set of test transactions against a test data base on an actual computer system or network. After running the benchmark, results are measured in terms of average response time or elapsed time to determine whether performance falls within acceptable parameters.

**Predictive Modeling.**

Predictive modeling—which is possible only as a result of prior benchmarking—generates estimates of system performance. The performance factors include quantitative models of application transaction volumes and resource use, data base volumes and structure, and processor capabilities. Unlike benchmarking, predictive modeling cannot accurately estimate competition for shared resources.

In the mainframe environment, the few technical options available changed slowly. It was possible to do benchmarking as a research and development activity, and then use the knowledge gained to build quantitative models that would support predictive modeling for years. The situation is very different in client/server environments.

Predictive modeling is quicker, easier, and less costly to do than benchmarking, but it requires an accurate set of performance factors for the target hardware and software. In the current environment, products and technology change so quickly that assembling all the information needed to build a predictive model for target technologies is difficult. In many cases, benchmarking is the only option if performance engineering is to be undertaken.

**Performance Engineering Versus Hardware Expenditures**

Some organizations try to accomplish performance engineering through postimplementation tuning or by buying more hardware. Usually, such attempts are unsuccessful. When they do work, these measures are expensive, with operating costs disproportionate to the service levels provided. At the same time, as hardware prices fall, companies must balance the cost of performance engineering against the amount saved in hardware costs and possible implementation delays.

Several tradeoffs thus must be evaluated when considering the merits of performance engineering versus hardware expenditures. Before making a decision to invest in hardware rather than performance engineering, companies should take into account three key factors:

- **Performance engineering for an application is a one-time investment.** Hardware and operating costs are ongoing for the life of the application.

- **Numerous performance problems cannot be solved by increased hardware.** For example, many availability requirements and service levels can only be met through performance-oriented design.

- **Scalability should be engineered into an application during the design phase.** If not, the cost of scaling an application later may be prohibitive.
More than ever before, a company's competitiveness depends on how well it builds new applications and uses advanced technology. Development costs are usually high, and if new applications do not perform to expected levels, that investment is substantially misspent. Fixing a poorly designed or badly installed application can delay its implementation by months or even years. Performance engineering can reduce the risk of unplanned costs.

Benefits and Challenges in Client/Server Environments

Although most organizations are familiar with performance engineering in mainframe environments, they are far less familiar with its implementation in client/server environments. Client/server technologies are essential for most Business Process Redesign and reengineering initiatives, and have been deployed in every functional area. Often, an organization will move toward a client/server architecture simply to make use of current technology or to take advantage of falling hardware prices.

The client/server environment is emerging so quickly that no consensus on its definition has yet been reached. What is clear, however, is that the technologies being applied differ from the ones that companies have come to know so well during the past 20 years in the mainframe environment. This fundamental change poses new and different challenges in applications that must perform at required levels with acceptable availability.

At the same time that they offer distinct advantages, client/server systems introduce several issues that were either nonexistent or easier to address in mainframe environments.

Large Numbers of Components.
The sheer number of components is greatly increased by the size and complexity of the systems being built, and by the tendency to create smaller, more modular components. As a result, there are more rocks to look under to uncover performance problems. Where performance efforts once focused on data base design and application data access, they must now be increasingly sensitive to network load and design, desktop performance issues, middleware bottlenecks, and the new challenges created by shortcomings in data base management system (DBMS) software available on the midrange platforms.

Multiple Environments and Configurations.
Client/server technology enables and encourages the creation of heterogeneous environments comprising many different hardware and software components. Because each of these components is becoming a commodity, it is almost inevitable that an architecture will have components from many different vendors. The large number of possible configurations makes it difficult to judge the practicality of an architecture by comparing one system to another. Even if comparable architectures can be found, it is likely that some of the components will be from different hardware vendors, so performance metrics will vary. Performance engineering thus must address large numbers of possible component synergies.

Object Orientation.
Both client/server systems and object orientation represent dramatic changes in applications architecture, and many companies attempt to implement them simultaneously. To date, there have been few, if any, instances of object orientation implementation for large transaction-processing applications in client/server environments. Companies must strive to understand how these applications can accommodate good performance design and how to integrate Object-Oriented Design and implementation with relational data bases.
Accelerated Application Development (X/AD) and Prototyping.
The union of X/AD, prototyping, and performance engineering is not always a happy one. X/AD and prototyping emphasize small, decentralized activities; performance engineering is a centralized activity. X/AD and prototyping techniques encourage developers to “go fast.” Performance engineering’s message is to “be careful.” X/AD and prototyping do not tolerate bottlenecks. Performance engineering must sometimes be a bottleneck. The important thing to remember is that the overall business objective is the same: to deliver business benefits through technology.

Service Levels.
Client/server technologies—and the types of business applications being implemented with these technologies—change the way organizations define service levels. IT organizations face complicated decisions about acceptable response time and batch versus online processing. Even the definition of a transaction has become less clear. As a result, IT managers must tie service levels more closely to business transactions than to technical ones. If the average phone call to customer service now takes two minutes instead of three minutes, it should be counted as a success even if the technical processes take longer to complete.

Meeting the Challenges of Client/Server Environments
The new challenges presented to performance engineering by client/server environments are not insurmountable. At one time, performance engineering was new to the mainframe environment. It is now pervasive.

Similarly, as client/server environments evolve, the importance of engineering performance into the applications development process is increasing. Despite the abundance of inexpensive processing power on the desktop and server platforms, client/server systems have several characteristics that make performance engineering critical.

First of all, the hardware and architectural choices within a given client/server profile—such as the location of application logic and Structured Query Language, tiers in the architecture, use of stored procedures and triggers, and network topology—greatly affect system performance. In addition, with the system spread over several processors, the network—especially the wide area network—can be a substantial bottleneck. Data distribution decisions can profoundly affect system performance and scalability.

Second, new products or releases that have yet to be proven in actual use, especially with substantial volumes, are frequently necessary to implement new architectures. While the resistance to using unproved products has eased by necessity, too many product choices are based on the promises of vendors rather than on track records. Even if it has been proven, the software used in nonmainframe client/server systems has shortcomings that make it difficult to implement the very large data bases that can be accommodated by a mainframe data base management system.

Despite the shift in environments from mainframe to client/server, the objective of performance engineering remains the same: building applications that meet business requirements. Regardless of the environment, performance engineering is still a requisite for optimal system design.
Implementation Phases

As in the mainframe environment, performance engineering for client/server environments is implemented through a clearly defined development life cycle (see Exhibit 1). A brief outline of the specific tasks and deliverables involved in each phase of the project follows.

Phases, Tasks, and Deliverables of Performance Engineering in Client/Server Environments
<table>
<thead>
<tr>
<th>Phases</th>
<th>Tasks</th>
<th>Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision and Strategy</td>
<td>Begin planning</td>
<td>PE components of project plan</td>
</tr>
<tr>
<td></td>
<td>Identify resources</td>
<td>PE personnel requirements</td>
</tr>
<tr>
<td></td>
<td>Collect data</td>
<td>Key estimating factors</td>
</tr>
<tr>
<td>Architecture</td>
<td>Review methodology</td>
<td>Add PE deliverables to methodology</td>
</tr>
<tr>
<td></td>
<td>Conduct PE review of architecture</td>
<td>Performance-oriented system architecture</td>
</tr>
<tr>
<td></td>
<td>Define service levels</td>
<td>Service-level definitions</td>
</tr>
<tr>
<td></td>
<td>Plan training</td>
<td>(response time, availability, processing windows, report turnaround)</td>
</tr>
<tr>
<td></td>
<td>Refine key estimating numbers</td>
<td>Training curriculum for performance engineers, programmers, data base designers, architects</td>
</tr>
<tr>
<td>Business Requirements</td>
<td>Influence data model</td>
<td>Data model with statistics for performance modeling</td>
</tr>
<tr>
<td></td>
<td>Influence technical architecture</td>
<td>Initial models for peak processing period, key batch processes, key online processes, utility</td>
</tr>
<tr>
<td></td>
<td>Construct predictive models</td>
<td>schedule, daily/weekly/monthly processing schedules, network load</td>
</tr>
<tr>
<td></td>
<td>Conduct initial benchmark</td>
<td>Proof-of-concept benchmark for technical architecture</td>
</tr>
<tr>
<td></td>
<td>Refine technical architecture</td>
<td></td>
</tr>
<tr>
<td>Detailed Specifications</td>
<td>Influence physical data base design</td>
<td>Performance-oriented physical data base design</td>
</tr>
<tr>
<td></td>
<td>Perform capacity planning</td>
<td>Capacity estimates (CPU, disk, network, desktop)</td>
</tr>
<tr>
<td></td>
<td>Execute detailed benchmark</td>
<td>Detailed performance models</td>
</tr>
<tr>
<td></td>
<td>Enhance performance models</td>
<td>Precise benchmark for application to support capacity planning effort</td>
</tr>
<tr>
<td></td>
<td>Complete performance-oriented training</td>
<td>Trained development personnel</td>
</tr>
<tr>
<td>Coding and Unit Testing</td>
<td>Conduct stress test tests</td>
<td>Established performance monitoring capabilities</td>
</tr>
<tr>
<td></td>
<td>Perform response time tests</td>
<td>Refined technical architecture</td>
</tr>
<tr>
<td></td>
<td>Conduct code reviews</td>
<td>Improved code, especially SQL or other data access</td>
</tr>
<tr>
<td></td>
<td>Execute benchmark against alternative technical architectures</td>
<td>Refined capacity plan</td>
</tr>
<tr>
<td></td>
<td>Provide additional training</td>
<td>Performance monitoring capability</td>
</tr>
<tr>
<td>Installation and Support</td>
<td>Perform tuning</td>
<td>Performance improvement, cost reduction</td>
</tr>
</tbody>
</table>

**Vision and Strategy.**
During this initial phase, data collection commences and key estimating factors are determined.

**Architecture.**
Data collection continues to ensure that the methodology being assembled contains components that will provide the information necessary for performance engineering.
Service levels are defined for transaction response time, query response expectations, data and system availability, processing windows, and key processing deadlines. Vendor software products are identified for both application processing and performance monitoring.

**Business Requirements.**
Data collection continues, and a first-level benchmark is conducted as a technical proof of concept. Predictive models are set up to provide initial estimates and to serve as the basis for expansion in subsequent steps.

**Detailed Specifications.**
The project team enhances the models established in the previous phase, completes physical data design, and conducts any additional benchmarking. Next, the tools and procedures needed to monitor performance in production are defined and procured.

**Coding and Unit Testing.**
Unit testing is performed to check response time and to isolate and correct performance issues at any level, but particularly in the data access or desktop architecture. As part of the system and integration testing phase, the system is subjected to stress testing, and code is reviewed—especially data base access code.

**Installation and Support.**
During and shortly after installation, heavy performance monitoring and tuning occur. In addition, data bases that begin with little or no data and grow and fragment with the use of the system will require careful monitoring.

**Conclusion**
Performance engineering has repeatedly demonstrated its value in mainframe environments. As many organizations migrate to client/server environments, performance engineering projects are producing similarly dramatic results in a new environment. The challenges are different, but the lesson remains the same: realizing the benefits of client/server technology is probable only when performance is engineered into large applications from the earliest stage of development.

**Author Biographies**

Scott T. Smith

Scott T. Smith is a senior partner at CSC Consulting, in Waltham MA. He leads the company's Database Center of Excellence and directs internal research and development projects to benchmark leading data base management products.

Copyright 1995 by Computer Sciences Corp. Reprinted with permission.