WHY WORRY ABOUT PERFORMANCE IN E-COMMERCE SOLUTIONS?

Dr. Ed Upchurch & Dr. John Murphy

Abstract

This paper will discuss the evolution of computer systems, and will show that while the system performance is getting increasingly critical, it is also getting harder to master performance assurance and to master the system evolution. This paper proposes that the real challenge posed currently for the performance engineer is the explosion of e-Commerce systems where the users are increasingly impatient and the design methodologies are in their infancy. The reason that the systems are getting more complex is that the system dynamics are critical to performance, but they are not addressed by traditional design methodologies. One solution to this is to use a performance assurance process that guarantees that the system dynamics are captured. Overall the odds are not good for building complex systems, but performance assurance can reduce risk by a factor of ten or more.

1. Introduction

Computer systems have been evolving from simple mainly static systems to more dynamical systems and hence more complex systems. They have changed from relatively simple, single-threaded computers, initially used for scientific and engineering applications, through to large time-sharing mainframes for business applications, multi-threaded personal computers, and simple servers often connected into local area networks (LAN’s). Larger distributed systems were built from LAN-based systems by connecting them into client-server systems, often connected over wide area networks (WAN’s). The increasing availability of the Internet has enabled another step in this evolution: so-called e-Commerce systems that allow millions of users to connect to other users and servers. The order of magnitude increases in system size and the number of users make e-Commerce systems the most complex computer systems to date.

Throughout this evolution system performance has become increasingly critical but harder to master. Issues such as load balancing, guaranteed response times, and flow control, have become more difficult as system dynamics have increased and workload variations have also increased. The performance of mainframe computer systems has become fairly well understood and a number of performance measurement and analysis tools are available. The performance of client-server systems is not as well understood. However measurement and analysis tools are maturing so that performance of these systems is moving from an “art” to a “science”. The current challenge for performance engineers is the explosion of e-Commerce systems. Performance analysis of e-Commerce systems is in its infancy, yet the user community has shown itself to be impatient where response time is concerned.

This paper firstly deals with the evolution of complex computer systems, which were mainly distributed client-server systems and are currently mainly e-Commerce systems. The pitfalls that can occur in these systems and the reasons why these happen are discussed in the next section, where both the technical and political pitfalls are described. The following section deals with the problems that can occur when performance is ignored, or when the pitfalls mentioned previously are not neutralized. The performance assurance process is described in the next section, and finally some conclusions are drawn.

ecommerce-performance limited. Tel: +44-1684-566434 Fax: +353-1-6335651
The first author is now a consulting partner at Hyperformix, Austin, Texas, USA
Upchurch@hyperformix.com    John.Murphy@ecommerce-performance.com
2. Complex Computer Systems

Distributed Client-Server Systems

Client-server systems can be considered to be representative of complex systems in terms of performance analysis for a number of reasons. They are designed to support a large number of concurrent users, possibly thousands, and to handle a large number of concurrent transactions, possibly millions. The workloads in these client-server systems are dynamic, with large variations in loads over short time scales like hours or minutes. Distributed client-server systems generally have one or more of the following attributes:

- Distributed Networked Systems using a mixture protocols, for example ATM, IP, Leased lines (T1/E1), FDDI, Novell and Ethernet;
- Multiple Shared Parallel Servers with multiprocessors and shared memory;
- Commercial Operating Systems for example UNIX, OS2, Windows and NT;
- Multiple Large Parallel Databases with terabytes of RAID (Redundant Array of Inexpensive Disks);
- Commercial Software Products (COTS) mainly for transaction management. These are essentially “Black Boxes” from a performance viewpoint, for example Oracle, Tuxedo, Workhorse and InConcert.

Performance engineering of these systems requires an understanding of the system’s dynamics, as that is critical to system performance. This traditionally has not been addressed by design methodologies. Consequently the design of client-server systems over the last ten years does not have a good success record as is shown in Figure 1, where only one sixth of projects are completed within budget and on time.

![Figure 1: Project Completion Rates](image)

Performance is a major cause of these failures. Therefore the odds are not good for complex system builders unless they apply non-traditional design methods that can capture the dynamics of the systems for performance assurance. E-Commerce systems are even more complex than distributed client-server systems, so the odds are even worse. Performance assurance methods are designed to reduce the risks by at least an order of magnitude.

E-Commerce Systems

E-Commerce systems have all the elements of complex distributed client-server systems plus potentially many orders of magnitude more users, typically millions. The Internet was not designed for performance, reliability, security, or to handle multimedia data. There are many additions to the zoo of commercial off-the-shelf (COTS) products for e-Commerce applications. These include turnkey e-Commerce products that are often immature and of unknown performance. Standard e-Commerce benchmarks are immature and not generally widely understood. E-Commerce solutions are often delivered by the addition of a layer to legacy complex client-server systems. As the legacy system's performance may not be well known, there is not much that can be understood about the performance of the e-Commerce system. Performance is traditionally the last aspect considered in such systems.

E-Commerce system performance is proving to be critical. Millions of customers directly interact with the systems and there are a variety of systems that they can choose from. Most studies have found that the customers are very unforgiving. In a highly competitive environment, market share can be quickly won and lost. The time frame for solving performance problems has been greatly reduced, from traditionally months or years to weeks or less.
3. Performance Assurance Pitfalls

Technical Pitfalls

There are a number of technical pitfalls associated with any programme of performance engineering and assurance of e-Commerce systems. Inadequately estimating demand, workload or scalability requirements are especially important for e-Commerce systems where demand may increase dramatically. Lack of, or loose, performance requirements that are often derived ad hoc without reference to the customer needs means that system designers will ignore performance. An unbalanced architecture may lead to some resources being starved while others are bottlenecks. Work may not flow smoothly and fairly due to poor scheduling and flow control, and thus users will receive unpredictable performance. This is especially important for e-Commerce systems that have a large number of users and diversity of concurrent transactions. Some additional performance pitfalls include the following:

• No Live Monitoring: Dynamic systems must deliver near-real-time performance feedback so that changes in system performance requirements can be reacted to intelligently and quickly;
• Single Threaded Code: Multi-user systems require concurrent use of scarce resources. Multi-threaded COTS packages and operating systems may contain single-threaded code in areas critical of the application, therefore where multi-threading is needed must be determined;
• Poor Testing: Spotty testing may miss performance bottlenecks and give a false sense of system capability;
• Believing Vendor Benchmarks: Vendors use benchmarks to sell their products and so standard benchmarks may not apply to the application under consideration.

Political Pitfalls

Avoiding political pitfalls has been found to be as critical as avoiding technical pitfalls to successfully produce systems that meet performance requirements. Some of the political pitfalls that have been observed include:

• Wishful Thinking: Developers think that they know how to design systems and write software. They often feel lucky, have nice computerized development schedules, and feel that they can win the performance design lottery;
• Old School Ideas: Selecting an architecture previously used or that looks state-of-the-art from the literature, sketching the functions, picking some great-sounding COTS, quickly writing some interface code, slapping it on the big iron, meeting the project schedule and declaring success;
• Why Not Do Performance Later On: Just tune the system later, optimize the code, and if the hardware is too slow add more iron - its faster and cheaper every day;
• Perceived High Up-Front Costs: More time and planning required for a scientific performance assurance process than clueless speculation and rule-of-thumb activity;
• Process Not Understood by Management: System functionality and writing code is easy to put on a chart, whereas more depth of understanding is required to forecast the benefit of performance assurance, and often high level management are not skilled computer professionals or are out-of-date;
• Lack of Upper Management Support: Simulation & modeling should be done as it's nice insurance and sounds high-tech and state-of-the-art. However when it requires support give it low priority as it does not appear to be on the critical path;
• Lack of Distribution of Information: Modelers need to ask for system information early in the project, sometime before it has really been decided upon finally. For example, what the workload is; what are the main system functions; and what are the major system components. Of course many of these details are not yet known;
• Lack of Software Designer Support: Performance experts often perceived as being underfoot, taking time away from developing the system. Many designers don’t know about performance and want to just get the code debugged on schedule.

Performance assurance effectiveness will be severely reduced unless both types of pitfall are neutralized.
4. Performance Blunders

This section shows some examples of where the technical or political pitfalls were not fully contained and the resulting systems had serious problems.

1. **Military System:** When the question was asked, how many messages per second should the system be capable of handling on the average and in a peak battle situation, no one knew. After insisting on this point a guess was made, and performance assurance found that the current system design could only do one hundredth of this at best.

2. **Large Government System:** The pilot system response was minutes rather than seconds. Found that the proprietary database coded specifically for this system produced twice as many disk inputs and outputs than any disk system at that time could deliver. The final result was that the system had to thrown out and re-done later, by another company, after a cost of over $800 Million.

3. **Deep Space Communications System:** After spending a significant amount of time and money on a new multi-processor system, it produced no speedup in communications or increase in throughput. Performance assurance found by careful measurement that the majority of the work was going to one of the processors and so the problem was fixed by load balancing correctly.

4. **Military System:** Poor scalability was apparent and the project thought that a faster CPU with a larger disk would be the answer to scalability. The hardware was bought and installed, however the performance was worse than before. Brought in the performance analyst at this stage and found by measurement and modeling that the disk, not the CPU, was the system bottleneck. The disk that was purchased was larger but slower than the previous one. Simulation showed what the system needed to be balanced and the problems were overcome without redesign.

5. **Distributed UNIX Operating System (OS):** Vendor commissioned performance analysis to determine the OS scalability to a massively parallel machine. The software developers were of the opinion that it was impossible to model the code. Performance assurance process produced a graphical based model that included system animation. From the animation of a page fault, the software developer responsible for the memory management portion was able to produce a greatly simplified piece of code within hours, which had enhanced performance.

6. **Large Government System:** A COTS transaction management system was selected and implemented without benchmarking for either the application or for the hardware configuration that was to be used. From performance assurance the resulting performance was found to be one thousand times too slow to meet performance requirements and so costly rewrites were needed.

7. **Large Government System:** Selected and implemented a COTS workflow management software without benchmarking. Later on performance assurance method found the code to be single threaded at one point so that it did not scale to meet requirements.

8. **Large Commercial Distributed System:** The system was designed and implemented without performance assurance and afterwards asked for ad-hoc performance assessment of the system. From the performance analysis found that the LAN would be a bottleneck and only half the proposed throughput could be achieved.

9. **Large Commercial Distributed System:** This project was the fourth attempt to design the system, each previous attempt was by a different large vendor. Performance assessment of the design revealed that the COTS database would be a bottleneck, not the client workstations as previously suspected.
5. Performance Assurance Process

Performance assurance may be defined as a methodology integrated with the system life cycle that ensures the system will meet all its stated performance requirements with a risk reduction factor of at least tenfold. It further recognizes that one cannot separate functionality, performance, availability and cost in system design. All four are equally important to complete a project successfully, as is shown in Figure 2. Performance assurance must also

- Be Applied from the time the system is just a gleam in the eyes of the users to the time the system must be retired for newer technology;
- Encompass all of the following: performance requirements analysis; proposal evaluation; cost analysis; performance engineering; availability analysis; performance analysis; simulation & modeling; performance testing; benchmarking & measurement; monitoring; capacity planning;
- Coordinate all the following functions into a comprehensive approach that produces a result greater than its parts.

![Figure 2: Essential Project Components to Successfully Complete the Circle](image)

**Key Steps of Performance Assurance**

The key steps of performance assurance are presented in Figure 3, and these represent a proven performance assurance methodology.

![Figure 3: Key Steps of Performance Assurance](image)

**Detailed Steps of Performance Assurance**

The identification of system performance requirements, functional workflow and workload are fairly atomic and well-understood processes. The other steps of the performance assurance methodology, which are the
supporting entities rather than the identification entities, are more complex and less well understood and will be
discussed in more detail.

To support the system architecture that has been selected, a set of steps have to be taken in the first instance;
this includes the building of a high-level dynamic system model and characterization and building of a
workload model. Two further models have to be built to support the system architecture, a major functional
layer model and a high-level candidate architecture layer model. After building these models, which in their
own right support the system architecture, the support changes to analysis where the outcome will be
identifying a shortlist of candidate architectures, the driving workloads, the performance metrics, and the
potential bottlenecks and sensitive components. The analysis is to perform a high-level performance analysis,
which checks if the system is complete and if the transactions flow through the system. A workload sensitivity
analysis involves the testing of the model with changing workloads, pinpointing if any of the workloads are
drivers of the model, or otherwise if some workloads are not important for the performance of the system.
Bottleneck analysis is stressing the model to find if there are single points for bottlenecks, and to find if these
can be overcome where the next bottleneck occurs. Tied in with this bottleneck analysis is the scalability
analysis where the architecture quantity is increased to see if improved performance can be obtained. Another
major analysis is the component sensitivity analysis where the components of the architecture are
parameterized for performance analysis, and these parameters are varied to see if the performance of the model
changes with changes in the components. If for example components are found to be sensitive then these must
be measured by benchmarking to accurately put in place in the model the parameters that describe the
component. Many of these systems have to be 24 hour and 7 day available within a certain tolerance. To find if
this is possible, an availability analysis is conducted. In this analysis components are failed to see the impact on
the performance of the system. From this analysis the availability of the components are put into the model and
the system is analyzed over a longer time period to find the overall availability. This is a complex operation as
compartment failure may have an effect on the system performance but may not bring the system down, just
degraded it. The last of the analysis tests is the “what-if” cases and stress scenario. Here certain possibilities for
future ideas can be tested in the model and the results analyzed.

When the system architecture has been decided upon then the performance assurance process moves to support
the architecture design and implementation. This takes the form of a number of steps:

◆ Expand detail of high level model in performance sensitive areas;
◆ Iterate on high level analysis steps previously listed;
◆ Support development trade-off analyses;
◆ Evaluate development performance models;
◆ Define system test scenarios;
◆ Define model validation measurements;
◆ Provide on going system performance projections;
◆ Provide model capable of evaluating system scalability against requirements in absence of feasible test
  scenarios.

After these functions have been carried out the performance assurance process moves to another phase of the
system life cycle where performance monitoring is carried out. From the results of this it might be necessary to
perform scalability analysis or capacity planning. Furthermore as the components of the system are being
improved, technology refreshment may be needed.

5. Conclusions

This paper has shown that e-Commerce systems are more complex than previous client-server systems, and
furthermore that the performance requirements placed on these e-Commerce systems are becoming increasingly
critical to a successful project completion. The most common causes for a lack of performance assurance
process were outlined by showing the pitfalls that can befall a project. When these pitfalls are not contained
then the resulting system that is designed can result in a disaster as was exemplified by previous blunders. The
performance assurance process developed in response to these problems was then outlined.