ASSURING PERFORMANCE IN E-COMMERCE SYSTEMS

Dr. John Murphy*

Abstract

Performance Assurance is a methodology that, when applied during the design and development cycle, will greatly increase the chances of an e-Commerce project satisfying user performance requirements first time round. This paper discusses the primary risk factors in development projects, the keys to a successful risk management programme, and the tools required. It also discusses problems that can occur in e-Commerce systems and some examples of how these might manifest themselves. A definition and the reasoning behind a performance assurance methodology is given, and the performance assurance methodology that is proposed is outlined.

1. Introduction

Distributed computing has captured almost universal attention as corporations struggle to gain competitive advantage in the information age. Currently only a small fraction of distributed computing projects are completed within budget and on time. E-Commerce solutions promise some of the greatest advances in information technology. However the very aspects of the e-Commerce systems that give them power and flexibility also pose significant new challenges and risks. Their open distributed nature makes designing and maintaining them far more complex than mainframe-based systems. In addition, most distributed computer systems pioneer new territory in both functionality and performance. Performance Assurance is a methodology that, when applied during the design and development cycle, will greatly increase the chances of an e-Commerce project satisfying user performance requirements first time round.

This paper begins with some of the performance problems that can occur in e-Commerce systems, and then continues with the benefits of having a performance assurance method. A proposed performance assurance methodology is outlined in the next section, with a number of steps to follow to achieve the benefits of the methodology. Some of the details of the methodology are then described, with simulation results, and finally some conclusions and future work are presented.

2. Performance Problems in e-Commerce

There are a number of factors that increase the risk of performance problems in complex computer and software systems, such as e-Commerce systems.

- **Inadequately Estimating Demand.** Any system, no matter how well designed, will not be able to cope with a massive increase in user demand. Therefore it is imperative to determine, in advance and as accurately as possible, the workload and scalability requirements of the system.
- **Loose Performance Requirements.** Poor specification of performance requirements can easily lead to a system not achieving its performance requirements. This is because, unless performance requirements are stated clearly and sensibly, system designers tend to ignore them in the effort to get the system to work correctly.

* Performance Engineering Laboratory http://www.eeng.dcu.ie/~pel
School of Electronic Engineering, Dublin City University, Dublin 9, Ireland. murphyj@eeng.dcu.ie

- **Unbalanced Architecture.** A well-performing system will in general have a balanced architecture where no component is over- or under-utilized. For example, a car engine is sized according to the dimensions of the car and its other components, even though more powerful engines are available. Equally, if some system component is severely under-utilized, it is either not really needed, or the system design is preventing its proper use.
- **Poor Work Scheduling.** The design architecture may be capable of performing the work, but if the work is not scheduled properly or not prioritized correctly then performance problems can occur. This is particularly important at times of high load, when low-priority operations may need to be suspended.
- **No Live Monitoring.** A system cannot be left to run unmonitored and still deliver its target performance as time progresses. It is crucial that changes in system performance are planned for and monitored.
- **Single Threaded Code.** This represents a potential performance bottleneck in software systems. The specifics of the application can be used to determine which parts of the system would benefit from being multi-threaded, and therefore executed in parallel.
- **Poor Testing.** Poor testing will not find performance bottlenecks and may even give an incorrect positive impression about the capabilities of the system.
- **Believing Vendor's Benchmarks.** Although vendors do not lie when presenting benchmark results, it must be remembered that a vendor's benchmark is only indicative of system performance. The vendor's benchmark may not represent the actual usage of the system or the actual system configuration.

3. Benefits of Performance Assurance

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. Before expanding further on the proposed performance assurance methodology, some expected benefits of its use are outlined by way of motivation. Performance assurance benefits a project by avoiding costly mistakes. This involves both foreseeing structural and functional problems before they arise, and detecting existing problems in the early stages so that they can be corrected at a lower cost. Since distributed computerized systems present so many risk factors [1], performance assurance in a computer system development project offers great rewards. Organizations that have implemented performance assurance to address each of these areas have had a far greater chance of developing a successful system. A successful performance assurance methodology should begin early and continue throughout the project life cycle. For e-Commerce systems development, this is especially true. The cost of making a change to a system increases dramatically as the project progresses as is shown in Figure 1.

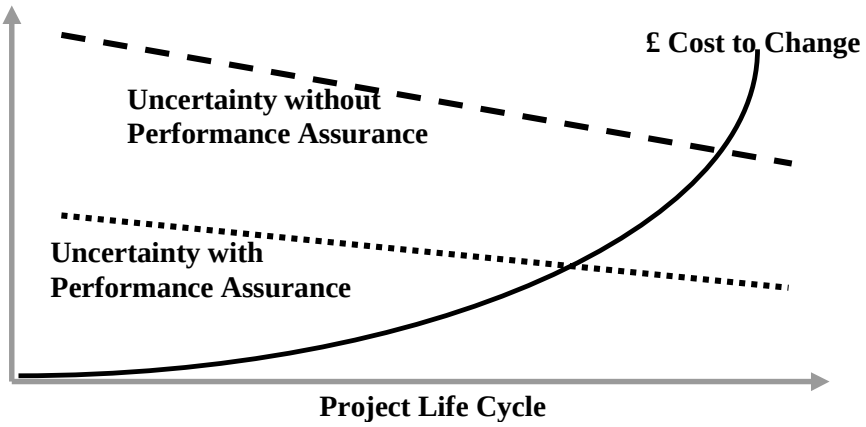

Figure 1: Project Life Cycle: Costs and Uncertainty

Making a change during the design phase is vastly simpler than making a change during development. For example, re-designing a system on paper is always much easier than re-designing one that has already been

built. If hardware and software have already been purchased, there is less latitude in what can be done to correct a problem. The uncertainty about how a system will function and perform decreases as the project progresses in time, but in the early stages of a project, uncertainty can be quite high. Without performance assurance, much of the early phase of system design is "guesswork", increasing the likelihood that costly changes will be needed later in the development life cycle. Performance assurance reduces uncertainty, allowing necessary changes to be implemented early or eliminating the need for changes altogether.

Late stage changes and fixes are costly for many reasons. The further into development, the more there is to repair if a flaw appears. Database tables, stored procedures and triggers, C and C++ routines, GUI windows and much more could all be impacted by a single change. Worse, if the system fails or needs modification during the production phase, the cost of downtime (not to mention lost business, customers, or reputation) must be factored into the cost of a fix. In short, performance assurance applied early can save a great deal of time and money in the long run and boost the overall quality of an information system. Applying performance assurance throughout the project life cycle, from planning through production, is the key to a successful distributed system. While risk and uncertainty are especially high in the early stages of a project, they never fully disappear. Every addition or modification to a system introduces new risk. During development, the addition of new features affects existing code and data structures. The sooner bugs and errors are detected, the less costly they are to fix, so new features should be tested as they are added. Likewise, during production, the addition or removal of users, data, hardware, software and networks, and the imposition of new requirements, all contribute to the need for continued risk management.

4. Performance Assurance Methodology

A successful performance assurance methodology covers the full project life cycle as indicated above. Different project stages call for different approaches. The following steps constitute the key ingredients for a successful performance assurance methodology and are shown in Figure 2. The amount of work performed at each stage will depend on the specifics of the project. The methodology that has been developed is general and must be customized to the specific needs of a given application.

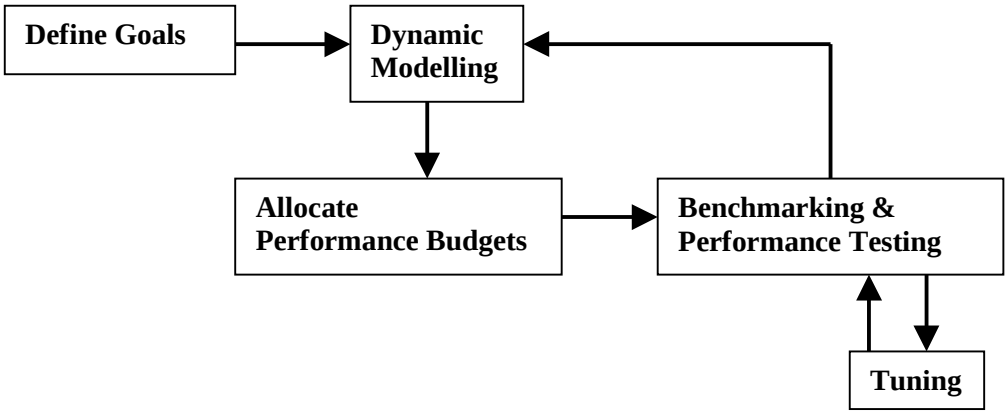

Figure 2: Performance Assurance Methodology

- **Define Goals:** Before proceeding with the project, the performance requirements and the workload the system is going to be subjected to must be known. The performance requirements must be driven by User and Business requirements, but consideration must be given to the cost of achieving those requirements. For example, on-line credit card authentication is desirable but expensive to implement, while off-line and email notification is a more cost-effective solution. The workload the system is expected to handle must also be defined.
- **Dynamic Modelling:** Use of Discrete Event Simulation can capture the component interactions, the variability in the system, the effect of priorities and scheduling, the congestion of resources, the

nonlinearities in the system, and the probabilistic nature of demands and resources. This allows for a more accurate prediction of response times than simple modeling techniques. Simple models cannot cope with congestion and queueing in the system, or priority schemes (which may be used to give performance guarantees). Scheduling can play an important part in determining system performance, but again can only be modeled dynamically.

- **Simulation and Modeling:** Simulation provides a means of evaluating a system design before the actual system is implemented. A model of a system, including descriptions of the hardware, software and networks, is constructed. The proposed workload is then simulated on the model system, and statistics are gathered. Response time, throughput, and other system performance statistics can be determined, and often bottlenecks are pinpointed. Simply scaling the model or workload can test system scalability. Since modifying a model or workload and performing multiple simulations is much easier and more economical than modifying an actual system, it is an efficient method for evaluating design alternatives in the early stages of a project. Simulation results are analyzed to determine the source of the problem, then the system design is modified. The new system is then modeled and simulated. If new problems arise, more modifications can be made until the system meets its requirements. Thus, modeling and simulation address system performance and scalability in the early project phases, reducing the risk of starting out on a wrong path. Simulation will also identify the critical components that are candidates for benchmarking.
- **Allocate Performance Budgets:** Performance Budgets are a crucial component of large projects where software components are constructed. A performance budget is allocated to individual subsystems or modules so that developers for those modules are aware of the performance requirements of their code. Without performance budgets the designers do not know if they should optimize their code or not. This can lead to un-optimized code that should be optimized, or even code that doesn't need to be optimized being optimized! Also at this stage it is important to define and implement metrics that will measure system performance and usage.
- **Benchmarking & Performance Testing:** Benchmarking is the testing of critical components of the system that have been identified as sensitive in the model. Performance testing (once the system has completed functional testing) will highlight any final performance problems.
- **Tuning:** The operational use of the system will always be different from the expected usage predicted in the requirements, and therefore the configuration options in the system must be tuned to achieve optimal performance. Tuning is performed by collecting various metrics defined during the performance budget stage and other system utilities.

5. Details of Performance Assurance in e-Commerce

Will simple models work? It is always possible to start out with simple models, but later on the move usually has to be made to more complex models. There is no need to move to a more complex model if it is possible to answer "No" to all of the following questions:

- Are there any dynamics in the system?
- Is there any priority scheme in use?
- Is parallel processing being used in the system?
- Is there any scheduling in the system?
- Is the load always small and not near capacity on any system component?
- Is the workload variation small?

As can be seen, few complex systems will be modelled effectively with simple models beyond the initial stages of the design. Dynamic modelling however provides a solution. With dynamic modelling it is possible to use discrete event simulation to capture the component interactions, the variability in the system and the effect of priorities and scheduling on the performance. It further captures the congestion of resources, the nonlinearities in the system, the probabilistic nature of demands and resources, and a more accurate prediction of response times. Looking at some simple scheduling algorithms, the workload time scales and congestion of resources shows some examples of how dynamic models can produce better results than simple models. The details outlined in this paper were simulated using SES/*workbench* [2], a discrete-event simulator that allows

hardware and software simulation. The models were created by use of its graphical user interface. SES/*workbench* then compiles the graphical code to C and creates an executable. The simulation execution platform was a cluster of Sun workstations.

Bottleneck Hopping

A bottleneck is simply the single resource of the system that limits the number of transactions that can flow through the system at a particular time. It can be seen that for different transactions there can be different bottleneck points, and further that these can interact with each other. Without dynamic modelling it is still possible to find performance bottlenecks. Typically this involves running a set of tests where the loading of the particular transactions are steadily increased, while monitoring all the systems resources, and once a resource is overloaded then this is the bottleneck point. This can only find the current bottleneck and can be a tedious procedure as the bottlenecks keep hopping around as they are found and addressed. This leads to a costly iterative process. An example of a simple architecture of an e-commerce system is shown in Figure 3 and this will be used to give an illustration of bottleneck hopping.

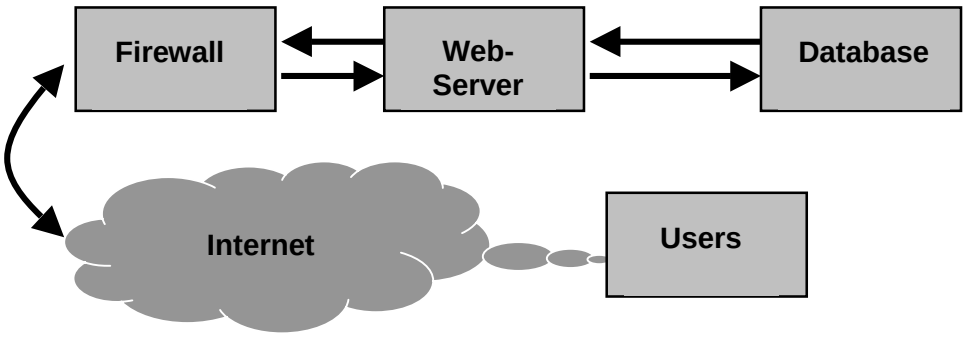

Figure 3: Example e-commerce System

In Figure 3, the users go through the Internet to access a database, however they also traverse a web-server that is protected by a firewall. Initially the firewall, the web-server and the database have two processors each. The specified workload is 60 transactions per second, however due to an initial bottleneck this input rate is not possible, and the first test loads the system with 30 transactions per second. The utilizations of all the servers are monitored and tabulated in Figure 4.

	Test 1			**Test 2**			**Test 3**			**Test 4**			**Test 5**		
	CPU	**Util**	**%**	**CPU**	**Util**	**%**	**CPU**	**Util**	**%**	**CPU**	**Util**	**%**	**CPU**	**Util**	**%**
Firewall	2	0.61	30	2	0.83	42	2	1.20	60	4	1.21	30	2	1.21	61
Web Srv.	2	**1.99**	**100**	4	2.78	70	4	**3.99**	**100**	4	**3.99**	**100**	6	4.11	69
Database	2	1.42	71	2	**1.99**	**100**	4	2.89	72	4	2.86	72	4	2.96	74
Number In		5011			6679			4931			6679			9828	
Number Out		4724			6640			4917			6640			9821	
Throughput		0.94			0.99			1.00			0.99			1.00	
Input Rate		30			45			60			60			60	

Figure 4: Bottleneck Hopping

From Figure 4, Test 1 it can be seen that the web-server is overloaded (100%) while the other servers are not (Firewall is 30% and Database is 70%). It can also be seen that the throughput for this rate is 94% so that most of the input transactions are in fact getting through the system. The usual remedial action would be to increase the number of processors in the web-server from two to four. Test 2 shows the results in this action. It was possible to increase the loading on the system from 30 transactions per second to 45 per second while still achieving 99% throughput. The utilization on the web-server has been decreased to 70% and the firewall is still only utilized at 42%, but the database is now the bottleneck (100% utilization). At this stage either the number of processors is increased in the database from two to four (Test 3), or all the servers are upgraded from two to four (Test 4). However both of these remedial actions only shift the bottleneck from the database

back to the web-server while allowing the full input rate has been attained at 60 transactions per second. When the bottleneck arrives back to the web-server this is a serious case and can not be predicted from the serial testing of the system. It is likely that the project managers will wonder what understanding of the system is in place and when will the problems be solved. As can be seen from Test 5, this is done by increasing the number of processors from four to six in the web-server. A better understanding of the system could have been gained, and the iterative testing could have been avoided, if a dynamic model was used.

Scheduling Priorities

In complex systems there any many different types of transactions present concurrently in the system, many systems could have hundreds of types. Some of these transactions are more important than other ones, and so the uses of either priority schemes or scheduling algorithms are widespread. While it is possible to take into account some of these issues with simple models in a rather crude way, it is not possible to fully capture the effects that these schemes have on the response times or potential throughput. To illustrate the benefits of the proposed methodology, we simulate a single server with four different transaction types as shown in Figure 5.

Transaction Type	No Priority		High Priority To Orders		Low Priority To Downloads	
	Utilization	Response Time	Utilization	Response Time	Utilization	Response Time
Change	12.0%	3.80	13.9%	4.88	12.6%	0.71
Download	70.3%	3.98	68.8%	4.98	71.1%	**6.21**
Joining	9.3%	3.63	9.8%	4.78	10.0%	0.63
Order	2.3%	3.58	2.3%	**0.43**	2.5%	0.60
Total	93.9%		94.8%		96.2%	

Figure 5: Priority Allocation

As can be observed from Figure 5, with no priority scheme in place the response times of all the transactions types are similar and between 3.58 and 3.98 seconds. If the "Orders" are given high priority over all the others then the response time for them drops from 3.58 seconds to 0.43 seconds, while the other transaction types increase to between 4.78 and 4.98 seconds. If on the other hand the "Downloads" are given low priority then the average response time for these increases from 3.98 to 6.21 seconds, while all the other transaction types decrease to between 0.60 and 0.71 seconds. Due to the random number generators in the simulations the total load in the three sets of results are slightly different.

Workload Time-scales

Apart from the usual daily variation in workloads, and the workload variation during the day for most non-scheduled workload, there are other issues with the time scales. If the workload is known at one time scale (for example one month) and the response times are known at another time scale (for example seconds) then it is difficult to transform the workload to that new time scale. An exponentially distributed Inter-Arrival Time (IAT) generates the workload for the example in this paper. One week's data was generated which consisted of 60,000 samples and the results are shown in Figure 6.

Duration	Peak Number	Average Inter-Arrival Time	Apparent Load
1 Day	8725	9.82	**100.00%**
1 Hour	402	8.88	**110.58%**
1 Minute	17	3.50	**280.57%**

Figure 6: Time-Scale Changes

As can be seen the average inter-arrival time over one day is given as 9.82 seconds, but if the peak hour is examined then this average IAT decreases to 8.88 seconds which is an apparent increase in the workload of

over 10%. When the peak minute is examined the average IAT decreases to just 3.5 seconds which has nearly tripled the apparent workload of the system. The conclusion of this is that when the response time-scale is different to the workload time-scale then one has to be extremely careful.

No Congestion Means No Queueing

The view is often held that if the utilization on any of the devices is not near 100% then there is no large delay or performance problem. However as an example if there is an input rate of one transaction every minute and the server take 30 seconds to process it, the server is loaded to only 50%. Some would claim that there is no queueing in this system, however in the normal case of exponential service rates and Poisson arrival rates this is not the case. In fact there will be 30 seconds queueing and 30 seconds process time to give a total response time of 60 seconds. Furthermore if it had taken 0.9 minutes to process it then the queueing delay alone would be 8.1 minutes. The result of this is that there can be substantial queueing delays even when there are no devices with high utilization present in the system.

6. Conclusions and Future Work

The performance problems associated with e-commerce systems have been presented along with the benefits of performance assurance, and a proposed methodology to achieve that assurance. Some details of the performance assurance methodology are presented along with preliminary simulation results to illustrate the points being made on bottleneck hopping, priority and scheduling algorithms and relevance of time scales and queueing in the system resources.

Some future work would be to further enhance the methodology and to assess the various commercial tools available for performance evaluation under a number of headings. For example:

- Price
- Ease of Use - How easy is it to create test scripts?
- Security - Will the tool work with the various levels of security?
- Points of Measurement
- What response time does it measure?
- Number of Virtual Users
  - How many users can be created?
  - Can users do different things?
  - Does the tool emulate different IP addressees?
  - Does it recognize errors from the web-server?
- How good are the results captured and presented?

7. References

[1] "Why Worry about Performance in E-Commerce Systems", Ed Upchurch & John Murphy, Proc. of 16th UK IEE Teletraffic Symposium, May 2000.

[2] Scientific & Engineering Software, SES/*workbench* Reference Manual, Release 3.2, 1998.