# Predicting the QoS of an Electronic Commerce Server: Those Mean Percentiles

Diwakar Krishnamurthy and Jerome Rolia
*Systems and Computer Engineering, Carleton University, Ottawa, Canada, K1S 5B6*
*{diwa,jar}@sce.carleton.ca*

## Abstract

This paper presents a case study on Quality of Service (QoS) measures and Service Level Agreements (SLA) for an electronic commerce server. Electronic commerce systems typically rely on a combination of an HTTP server and a database server that may be integrated with other enterprise information resources. Some interactions with these systems cause requests for static HTML pages. Others cause significant amounts of database processing. Response time percentiles are well-accepted measures of QoS for such requests. In this paper we measure the behavior of an electronic commerce server under several controlled loads and study response time measures for several workload abstractions. Response time measures are captured for individual URLs, groups of functionally related URLs, and for sequences of URLs. We consider the utility of these workload abstractions for providing SLA. We also show that empirical evidence of server behavior in conjunction with analytic modeling techniques may be useful to predict the 90-percentile of response times for sequence based workload classes. The model predictions could be used to support real-time call admission algorithms.

## Introduction

Electronic commerce systems provide transactional access to up to date information about products and services. These systems can support on-line shopping for consumer markets and automate interactions between corporate entities. As businesses begin to rely on these servers, performance management increases in importance. The purpose of this paper is to consider the advantages and disadvantages of several workload abstractions for assessing the QoS offered by an electronic commerce server and to provide techniques for estimating 90-percentiles of response time for such servers. The results can be used to support call admission algorithms.

Electronic commerce systems [1][2][3] typically rely on web based interfaces to access and present content to users distributed across the Internet. Client requests are translated into database queries with results rendered back to the client in the form of HTML. The resulting HTML can include other hyperlinks that enable subsequent interactions with the system. Security protocols are used in tandem with HTTP to provide privacy.

SLA are typically based on *workload classes*. They represent guarantees regarding the quality of a supported service in the system. In transaction processing systems, the quality of service is often measured by the 90 percentile of response times and SLA are defined using this metric. For example, a system could have a SLA with its clients that states that the 90 percentile of response time for a particular workload class is 2 seconds with a throughput of up to 100 requests per minute. This implies that as long as the throughput constraint is satisfied, the server must process requests from the workload class in less that 2 seconds 90% of the time.

Choosing workload classes is not always a straightforward task. Too many or too few classes can lead to very tight or loose performance requirements. Furthermore, requests for service from an electronic commerce system typically require the use of the Internet or an Intranet. Network latencies and variability can dominate a client's view of server response time but are beyond the control of the system's administrator. These environmental features should be taken into account when considering QoS requirements for Internet servers.

In this paper we conduct experimental runs on a web based shopping application to collect response time measurements. We present a brief review of benchmarking and measurement techniques for Internet server systems from the literature. Arlitt and Williamson [4] create a workload model based on the network level of abstraction for requests submitted by World Wide Web (WWW) clients. They achieve this by instrumenting the Mosaic browser and observing the client traffic traces. This provides a simulation model of the client side workload for WWW servers. For electronic commerce systems, we must focus on the server system itself. It is the only aspect of the system within our administrative domain of control.

Several efforts are underway to provide industry standard benchmarks for Web Servers. Webperf [5] from Standard Performance Evaluation Corporation (SPEC) and Webstone [6] are benchmarking tools for web servers. They measure the throughput in HTTP operations per second and client response times per HTTP operation. The workload characterization parameters include request rate distribution, content size distribution and request type distribution. However, electronic commerce servers maintain state information about customers using the servers. More sophisticated workload generation techniques are needed to cause valid user behavior.

Analytic models provide fast feedback on system performance required for capacity planning and online admission control. Dilley et al [7] developed a predictive model for a large scale Web server. Three workload classes were used to characterize the system's behavior namely *html, cgi, and image*. Each class used the system and Internet resources differently and had very different response time measures. The analytic model was used to predict response times for increasing numbers of customers accessing the system with varying network bandwidths. To support online admission control predictive models must provide fast estimates for the 90-percentile of response times of the workload classes in addition to the mean estimates. This presents a challenge since it requires very detailed models with parameters that are difficult to measure.

In this paper we employ a custom measurement infrastructure [8] to submit representative loads on a web based shopping server. Response time measures are collected for various URL based workload abstractions. The results are analyzed to propose a technique that combines information from empirical measurement data and analytic models to predict 90-percentile of response times. An analytic model is developed and validated for the web based shopping server under test and the technique developed previously is used to estimate 90-percentile of response time from the mean response time estimates of the model.

In web based shopping servers, customers follow typical sequences of URLs as they move towards the completion of transactions. The sequences make use of three significantly different kinds of URL links. They are for:

1. *Static HTML pages* – These are files with fixed and known content. In the case of an online shopping mall, these could be store and mall home pages and images.
2. *Fixed database queries* – The content returned by these requests are generated by static SQL queries. For an electronic shopping mall, links that browse and view products in the database would constitute this request type.
3. *Transaction requests* – These requests rely on user inputs and may cause a sequence of SQL queries on many tables in the database. The database queries are constructed from field values in the HTML forms and content is generated dynamically from the query results.

In general, Transaction requests cause the highest CPU demands at the server followed by the fixed database queries and the static pages. They perform application specific processing, validate inputs, and retrieve data. In a shopping mall scenario these functions include inventory checking, cost calculation and other shopping related functions. From the shopping mall perspective, the static HTML page requests and the Fixed database queries can be grouped together as *browse* requests since customers use these links to navigate the online catalog of products.

In the following sections we discuss qualitative and quantitative issues involved in selecting workload classes for these systems and providing SLA for customers based on these workload classes. QoS measures for classes based on individual URLs, groups of functionally related URLs and sequences of URLs are discussed. The choice of classes along with a specified 90-percentile for response time determines how many customers can be admitted to the site concurrently and affects the throughput and utilization of the server.

Section 2 describes the experimental set up used to collect measures of server QoS and resource demand measures. Choices for classes of work are studied in Section 3. Section 4 considers the suitability of analytic models based on MVA for predicting QoS in terms of the 90-percentile of response time. Section 5 summarizes our results and offers conclusions.

## Section 2 Experimental Setup and Design

The hardware setup consists of a web based shopping server running on a dedicated Windows NT node. It provides an online shopping mall for potential Internet shoppers. The components of the server include a web server, a pool of shopping servers to handle shopping related tasks and the database. All these components reside on the same node. A number of Windows NT client workstations are employed to generate synthetic loads on the system. A dedicated 10Mbit Ethernet network connects the client and server nodes. The experimental test bed consists of a number of additional software components including a Workload generator applet, a performance monitor, and an experiment manager.

The w*orkload generator* is a Java applet, which generates a specified mix of browse and transaction URL requests to the server. Each client node runs a browser process that executes the applet. Client response time measurements for each URL hit are collected by the applet. All response time measures are stored and used for subsequent analysis.

The primary inputs required by the workload generator include a Java object called *URLGraph* and a list of transaction URLs that stress the important shopping functions of the server. The URLGraph is a hierarchical, graph-like representation of the static HTML and Fixed database query URLs in the system. Static HTML and Fixed database query requests are submitted to the server by traversing the URLGraph. The number of nodes to be visited in the URLGraph before retrieving a transaction URL depends on the specified ratio of the mix of transaction and browse URLs.

For this study a list of eight transaction URLs are specified to the workload generator. They correspond to searching for a product, adding a product to the shopping cart, deleting a product from the shopping cart, adding another product to the shopping cart, assigning products to a shipment address, preparing an order, finalizing the order and processing the order. We introduce random user behavior between these constraining steps by traversing a random number of nodes (chosen to maintain the ratio of transaction to browse requests) in the URLGraph, as mentioned previously. Branching probabilities of the URLGraph edges decide the path taken and for this study all branches from a node have equal probability of getting chosen. Invoking the transaction URLs in the sequence specified by the list causes an order of a product from the shopping mall. The ratio of transaction URL requests to browse URL requests is specified as 1:3.5. In future studies the workload will be chosen to better match real site behavior and/or the Transaction Processing Council TPC/W [12] benchmark for electronic commerce servers once it is released.

The experimental design varies the number of clients using the system from 1 to 5 and the mean times between successive requests. The time between client URL hits is defined as the client *think* time. These values are exponentially distributed and for most runs have the value of 2.5 seconds. One run was also performed with a think time of 15 seconds to assess the sensitivity of our results to this parameter. The workload applet measured the response times for individual URLs as well as for sequences of URLs that we have specified. The sequences of URLs consist of consist of meaningful groups of static HTML pages, fixed database queries and Transaction URL requests. All response time measures were gathered during each run to ensure a fair quantitative comparison of the different abstractions for workload classes. Since the network was lightly loaded and since clients were on dedicated machines we use Client response times to estimate server response times.

Performance monitor and experiment manager components complete the experimental setup. The *performance monitor* runs on the server node and collects CPU and disk utilization from the performance counters provided by NT. It logs the data as a comma-delimited file for further analysis. Client synchronization and the aggregation of performance data are the responsibilities of an *experiment manager* that also runs on the server node. Java and RMI technologies are used to implement the components.

## Section 3 QoS Measures for URLs, Functional URL Groups, and URL Sequences

We now consider three abstractions for workload classes. Individual URLs, groups of functionally related URLs and sequences of related URLs.

There are many distinct URLs that support an electronic commerce system. Moreover, they change as the products supported by the server change. It would be very difficult to specify a sensible requirement for every one of them. At the other extreme if one requirement were given for all URLs it would either be too weak and permit high variability for all but the most demanding URLs or too tight and not permit the effective utilization of the server. This is because the resource demands and response times of different URLs vary considerably.

From detailed results we also found that the size of the content returned by individual URLs was not highly correlated with URL response time. This is because most content is dynamically generated and has significant database processing costs over and above data transmission overheads. These results suggest that a simple QoS functions that relate size with demands and/or response times for all URLs will not be effective for managing the QoS of e-commerce servers.

Service classes can also be defined by grouping functionally related URLs in the system. For example we can consider static html page links, fixed database queries and Transaction URL requests to be workload classes. The advantage of this approach is that it aligns qualitatively with user perception of QoS. Users learn which types of actions take longest and adjust their expectations accordingly. Reasonable but different requirements can be specified for the different classes. Assigning and monitoring QoS agreements based on groups of URLs is much easier than for individual URLs. However meting requirements may still be a challenge due to the variability of response times within the groups. Subgroups could be further identified to overcome this problem.

Another approach to model the services offered is to consider classes of service based on sequences of URLs. This helps to build up models that describe the relationship between workload components and may help reduce the number of requirements to be managed. The sequences are chosen based on common sequences of tasks performed by a user. In the context of a shopping mall examples would be sequences of URL visits to choose an item or to complete an order. A combination of static HTML page request, fixed database queries and Transaction URL requests are needed to complete such tasks. The sequences need to be chosen so that there is little variability in the number of URL visits for each sequence.

The performance requirement on a sequence of URLs is based on the end-to-end response time provided by the server (not including user think times) for the sequence. Response time constraints could be chosen for sequences individually. In this paper we consider a more crude approach. We assume the end-to-end response time is constrained by the product of the mean number of URLs in the sequence and the per URL 90-percentile constraint. Clearly the longer the sequence the weaker the constraint.

This approach offers the advantage of defining meaningful and commonly carried out tasks in the system as workload classes. However the QoS might not be well aligned with end user expectations since sequences typically take a longer time to complete and the user may suffer annoying variability in response times for individual URLs within the same functional groups. In this way, sequences provide the weakest constraint on QoS.

For this study, the Transaction URL requests described in the previous section define the user's behavior (ordering a product from the mall). In between each pair of Transaction URL requests there are many static pages and fixed database queries as well. We define three sequence based workload classes – Big, Medium, Small. *Big* encompasses the entire sequence of URLs to complete the order based on the list of Transaction URLs specified. Two *Medium* sequences each include four subsequent Transaction URLs in the ordering process. Four *Small* sequences each include two subsequent Transaction URLs. We note that each Big sequence contains the four Small and two Medium sequences. We defined to the three classes to compare their usefulness from the perspective of QoS monitoring and modeling.

The workload generator discussed in Section 2 is instrumented to collect the response times for all the above mentioned abstractions of services. The data was collected for the experimental design specified in that section. The CPU and disk utilization measurements and the mean response time measurements that are reported have a 95% confidence interval within 10 percent of the reported mean values. Reported values for 90-percentiles are for results from all experiments.

To study the impact of QoS constraints on server utilization for the three different abstractions for workload classes, we consider the following QoS constraints:

- 90-percentile of response time for individual URLs is 2 sec
- 90-percentile of response time for individual URLs is 5 sec, and
- 90-percentile of response time for individual URLs is 10 sec.

We note again that for sequences, the 90-percentile of response time is the product of the 90-percentile value above and the number of URLs in the sequence. Figures 1 to 3 illustrate the results of the measurement study.

Figure 1 shows the number of concurrent customers, with respect to a specific mean think time, the system can admit before violating its QoS requirement. Figures 2 and 3 show the CPU and Disk utilization for these same scenarios. In the figures the workload class for aggregated static HTML pages are not included. These requests have very low response times and did not limit the number of customers that could access the system.

From Figure 1, we see that Transaction and Small sequence classes simply cannot support 2 second 90-percentiles for response times even for one admitted customer. Their service demands are too large. Medium sequences can admit one customer and still satisfy the QoS requirement. Big can support two. This is because the sequences are a combination of static HTML, Fixed, and Transaction URL requests and the aggregate end-to-end requirement is less constraining. We also see that for this example, the Small sequence is no more constraining than the Medium sized sequence. Longer sequences cut down on the number of requirements to monitor; shorter sequences offer more requirements and modeling flexibility. As the size of the sequence approaches one, the workload classes would be individual URLs or functionally related groups of URLs.

As the permitted 90-percentile for response time increases more customers can be admitted to the system without violating the weaker QoS constraints. With a 5 second requirement the *Big* sequence admits 3 customers, while *Medium* and *Small* only permit 2. A 10 second requirement removes the differences between the sequence based workload classes. A 10 second QoS requirement on CGI programs still only permits two customers to be admitted into the system.

It can be argued that defining QoS with respect to sequence avoids the problem of QoS, but in some ways so does increasing response time percentile requirements for individual groups of URLs. A 90-percentile of 2 seconds has much less variability than a 90-percentile of 10 or 15 seconds and it is the variability that would annoy a user.

When constraints are too tight CPU and Disk utilization remain unacceptably low, as evident from Figures 2 and 3. Given that client perceptions of response times from such services are likely to be dominated by highly variable Internet transmission delays care must be taken not to over-engineer the system by specifying constraints that are too tight. In general we suggest choosing sequences that are as short as possible (possibly functional groups) and yet permit acceptable server utilization.

## Section 4 Predicting the 90-percentile of Response Times

Measurements provide important information about past and current system behavior. Predictive models are needed to provide fast feedback for capacity planning and on-line admission control. In this section we present a technique that combines Mean Value Analysis based predictive models with empirical measures to demonstrate useful estimates for 90-percentiles of response times for systems that have admission control.

We choose to use Layered Queuing Models (LQM) and the Method of Layers (MOL) [9] for modeling the system. LQMs are QNMs [10] extended to include contention for software resources such as pools of server processes as well as contention for devices including CPUs and Disks. Process can offer many services and can request service from other processes in a layered manner. The MOL divides the layered model into a sequence of QNMs and solves them iteratively to provide performance measures for the system as a whole. Process interactions can be synchronous or asynchronous and processes can be multi-threaded.

Model input parameters include each service's hardware demands (CPU and Disk demands) and the mean number of visits it makes to other services. The choice of services is determined by the choice of workload classes

Mean Value Analysis (MVA) residence time expressions have been developed that take into account many kinds of software interactions [11] and have been integrated within the MOL. For the LQM developed within this paper, expressions are used that capture the performance impact of an HTTPd server pool with a bounded size [11]. Also the performance impact of work caused by a customer request but that does not take place until after the customer leaves the server is captured [11]. These are referred to as *multi-server* and *post-rendezvous* service behaviors, respectively.

We now describe the construction, calibration and validation of the model. Our choice of processes and services on the server node was problematic and largely determined by the monitoring support offered by Windows NT. Figure 5a presents a LQM for the Electronic Commerce Server. Windows NT provides CPU utilization at node (system) level and at the per-process and thread levels and Disk utilization at the node (system) level. The coarseness of reported data has a big impact on the level of detail that can be included in the model. For example it is not possible to use NT monitoring facilities to directly measure the CPU and Disk demands associated with any specific URL or subsets URLs. The only workload class presented in Section 3 that is directly supported by the monitor is the Big sequence. It is supported because we can attribute all of the node's CPU and Disk utilization to the Big sequence class. As a consequence we choose to place all processing costs within a finite sized pool of servers associated with the HTTP daemon. Clients share the pool of servers. More detailed models are the subject of future work. Table 1 gives the parameters for LQM and the results of the performance evaluation.

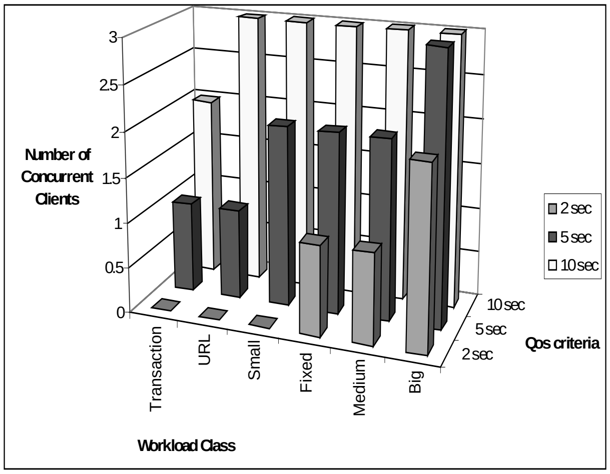


**Figure 2: Number of concurrent clients allowed in the system for various QoS constraints across the workload classes.**

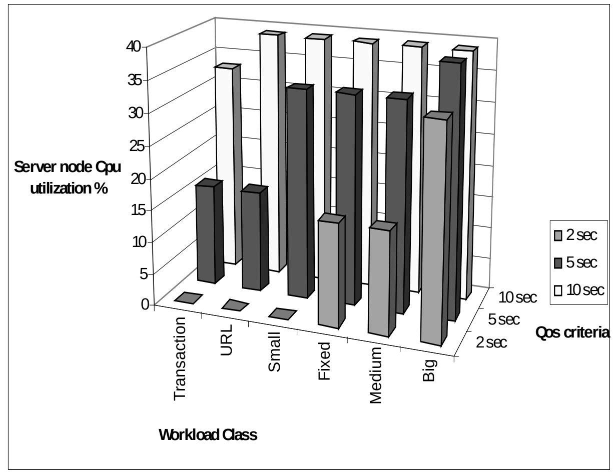


**Figure 3: Server node CPU utilization for various QoS constraints across the workload classes**

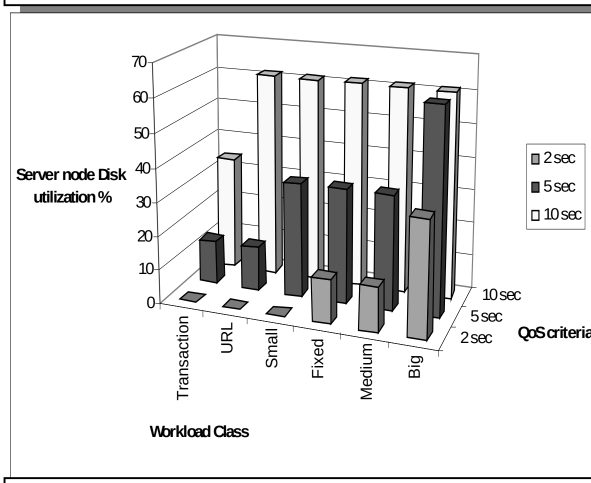


**Figure 4: Server node Disk utilization for Various QoS constraints across the workload classes**

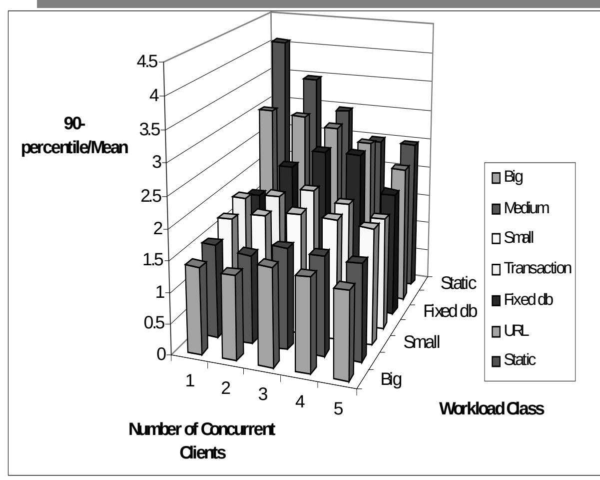


**Figure 5: Ratio of 90-Percentile to Mean for different concurrent client populations across the workload classes**

The device utilization $U_{CPU}$ and $U_{Disk}$ were gathered via NT's monitoring service, the mean client response times $R_{meas}$ were measured within the client applets. Big sequence completions, $X$ was measured by the clients. Mean think times $Z$ were computed for each experiment as the per throughput equations for a closed system.

The measured system's server pool had 10 processes. In the model, the number of processes was set as twice the number of customers up to a maximum of 10. This ensured that a server would be available for each client's post-rendezvous phase and a subsequent rendezvous phase.

There are several interesting aspects to these demand values. The mean CPU demand *per* Big sequence increases up to 139% depending on the workload conditions. The Disk demand increases up to 390% (not a typo) of its initial value. The reported utilizations have 95 percent confidence intervals within only a few percent of their mean values; these large changes are not due to measurement error. The reason for the changes is not due to measurement error. The reason for the variation is that the server was memory constrained. If more memory were added to this system, its resource demands would drop for high load conditions.

A second interesting point is that for the 1 client case, the sum of the demands is 50.57, yet the mean response time is 42.26. This indicates that a significant portion of the user's CPU and/or Disk demands take place after the client returns from the server. To reflect this in the LQM, the client's demand on the server is split into two phases, as shown in Figure 5b, with the client blocked until the end of the first phase. A residence time expression supported by the MOL reflects the performance impact of this post-rendezvous feature in the performance estimates. Since we have no other measurements to guide the partitioning of these demands, we calibrated the

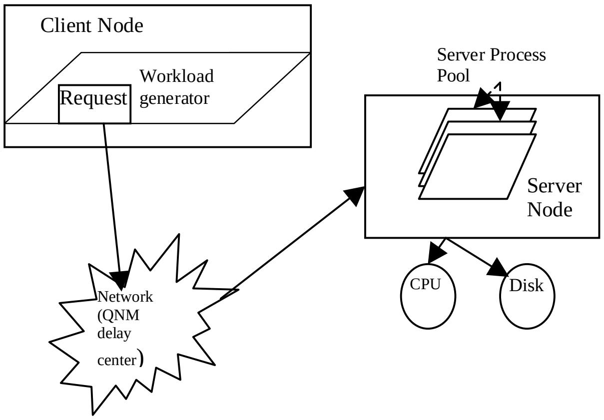


**Figure 5a: Layered Queuing Model for the Electronic Commerce Server**

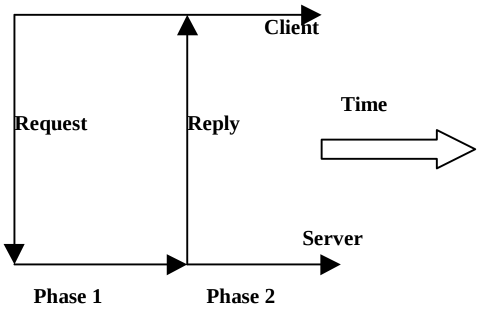


**Figure 5 b: Client-Server Phase relationships**

| Clients | $U_{cpu}$ | $U_{cpu}$ % | $D_{cpu}$sec | $U_{disk}$ | $U_{disk}$ % | $D_{disk}$sec | $X_{meas}$ | $X_{meas}$ | $R_{meas}$sec | $R_{meas}$ | Think sec | $R_p$sec | Rel Error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | (NT) | 95% CI+-% | D=U/X/T | (NT) | 95% CI+-% | D=U/X/T | (Big) | 95% CI+-% | Mean | 95% CI+-% | Z=NT/X -R | mean | |
| 1 | 16.07 | 2.48 | 27.813 | 13.16 | 8.47 | 22.777 | 41.6 | 2.86 | 42.26 | 4.37 | 131 | 42.27 | -0.0004 |
| 2 | 32.95 | 2.22 | 31.113 | 33.81 | 3.29 | 31.926 | 76.25 | 4.67 | 68.83 | 6.01 | 120 | 65.97 | -0.045 |
| 3 | 39.46 | 2.19 | 33.036 | 60.84 | 4.11 | 50.936 | 86 | 2.91 | 119.62 | 5.15 | 131 | 120.6 | -0.0081 |
| 4 | 37.54 | 2.06 | 36.232 | 88.58 | 1.38 | 85.493 | 74.6 | 6.14 | 249.09 | 4.59 | 138 | 264.3 | -0.0612 |
| 5 | 28.86 | 4.1 | 43.29 | 98.39 | 0.27 | 147.59 | 48 | 5.92 | 534.58 | 4.11 | 220 | 569.2 | -0.0647 |
| 5 | 24.48 | 3.25 | 38.744 | 56.28 | 0.94 | 89.08 | 45.49 | 3.98 | 215.39 | 4.56 | 576 | 198.5 | 0.0784 |

T= Time of each experiment run=7200 seconds

**Table 1 Model Parameters and validation**

model by finding a single partition that provided an accurate response time estimate for the 1 client case.

The partition we chose placed 56.5% and 100% of the CPU and Disk demand in the first phase, respectively. Given these values, the MOL estimates for mean Big sequence response time were accurate to within 5 or 10% of the measured values, for ***all*** the test cases. To verify the effectiveness of the analytic technique, we increased the client think time for the 5 client case to 576. This caused a significant reduction in client response time, which was well predicted by the model. Given the appropriate CPU and Disk demands, the analytic model behaves well. Each measurement required 15 hours to obtain statistical confidence in the reported measures. The analysis takes a fraction of a second per test case.

As shown above, predicting the mean response times for applications running on commercial operating system platforms is a challenge. The task requires load dependent models for CPU and Disk demand as well as queuing models. Statistical models can be used to help predict demands under various load conditions, but require empirical data from existing systems. Estimates for higher moments and percentiles of response times are likely to be even more sensitive to input parameter errors than estimates for mean response times.

We propose a pragmatic approach towards estimating the 90-percentile of response times. The approach combines the results of MVA with empirical evidence of the relationship between 90-percentiles and mean response times for a specific system. Figure 4 illustrates the ratio of 90-percentile to mean response time for the results presented in Table 1. We see that as the number of clients increases the ratio peaks then diminishes. This is expected since the system supports a finite number of server pool processes acting as a form of call admission. Customers within the system incur bounded delays when competing for resources thereby limiting their 90-percentiles. Systems that support admission control are likely to have such bounds and limit the ratio of 90-percentile and mean.

From the empirical evidence of Figure 4, the ratio of 90-percentile of response times to mean response time at the server was different for the three workload abstractions. URL, static HTML, and Fixed have relatively low response times and high ratios -- the worst being static HTML with a range of 1.4 to 4.0. The ratios are high because low response times can be significantly inflated by contention with other system users. The Transaction URL, Small, Medium, and Big classes require more processing and have fairly tight ranges. These ranges are adequate for estimating the 90 percentile of response time for the classes. The Big sequence class has a range of 1.4 to 1.6 including all the cases presented in Table 1.

The ratio tends to be lowest when servers in the pool have low or high utilization. For the Big sequences the ratio is consistent even though mean client response time increases by a factor of 12.6! This demonstrates the value and effectiveness of this approach. If a system were able to support many more customers, for example 30 customers, then the ratio of mean to 90-percentile may be larger than is shown in this example. The ratio is also likely to be sensitive to the mix of work taking place on the system. For these reasons empirical evidence is needed for each system to be studied. However characterizing this relationship is not an unreasonable requirement for systems supporting call admission policies.

## Summary and Conclusions

This paper considers three workload characterization abstractions for an electronic commerce server. Response time measures were captured for individual URLs in the system, groups of functionally related URLs, and sequences of URLs that correspond to user tasks. For each of these classes the mean and 90-percentiles of the response time are measured for several customer populations. The Big sequence class was used to build and validate an LQM for the system. The mean response times from the analysis were good predictors of Big sequence 90-percentile response times. The suitability of this approach for other server configurations and database sizes is work in progress.

The choice of workload abstraction and its corresponding response time percentile requirement used in SLA determines the number of users that can be admitted into the system. If SLA are based on individual URLs or groups of functionally related URLs then the response time requirements must be high relative to the mean to ensure that CPU and Disk resources can be sufficiently utilized. Defining SLA in terms of workload classes based on sequences of URLs cuts down on the variability of the corresponding end-to-end response time (Erlang principle). It offers a weaker form of QoS constraint. The sequences have the advantage that the mean and 90-percentile are better correlated so that it is possible to predict the 90-percentile of response time with more confidence and to use the results to support call admission algorithms. Functionally related groups of URLs tend to have similar responsiveness. We believe that they provide the most natural link to user expectations of QoS. However, Internet delays can be expected to distort user perceptions of the server's provided QoS, so it is not assured that this advantage can be exploited in practice.

Future work includes repeating the tests on a system with no memory constraints and the development of more detailed analytic models for the system. The interactions between the various components of the server and their individual resource demand and visit patterns should be characterized and reflected in the models. Models for predicting the CPU and Disk demands of the server need to be developed. The relationship between the mean and 90-percentiles of these system should be explored further and studied for configurations that admit more customers.

## Acknowledgements

This work has received financial and equipment support from the Telecommunications Research Institute of Ontario, IBM Canada LTD, Hewlett Packard Labs of Palo Alto, California, and the Natural Sciences and Engineering Research Council of Canada.

## References

[1] *Microsoft Site Server* - http://www.microsoft.com/siteserver
[2] *Oracle Internet Commerce Server* – http://www.oracle.com/
[3] *IBM Net.Commerce* – http://www.internet.ibm.com/commercepoint/net.commerce
[4] *A Synthetic Workload Model for Internet Mosaic Traffic*, Martin Arlitt, Carey Williamson, University of Saskatchewan. In ACM SIGMETRICS, Philadelphia, May 1996.
[5] Webperf - http://playground.sun.com/pub/prasadw/webperf/
[6] Webstone: Performance Benchmarking – http://www.sgi.com/products/webforce/
[7] *Measurement Tools and Modeling Techniques for Evaluating Web Server Performance*, J. Dilley, R. Friedrich, T. Jin, and J. Rolia, Lecture Notes in Computer Science, 1245, R. Marie, B. Plateau, M. Calzarossa, G. Rubino (Eds.), pages 155-168.
[8] *Performance Characterization of Web Based Shopping Systems*, D. Krishnamurthy, MEng. Thesis, Submitted to the Department of Systems and Computer Engineering, Carleton University, Ottawa, September 1998.
[9] *The Method of Layers,* J. Rolia and K. Sevcik. In IEEE Transactions on Software Engineering, Vol. 21, No. 8, pp. 689 – 700, August 1995.
[10] *Quantitative System Performance: Computer System Analysis Using Queuing Network Models.* E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik, Prentice-Hall, 1984.
[11] *Predicting the Performance of Software Systems*, J. Rolia. In CSRI Technical Report 260, University of Toronto, January 1992.
[12] *TPC Electronic Commerce Web Benchmark*- http://www.tpc.org/miscellaneous/TPC_W.folder/TPCW12.15.97/INDEX.HTM.