Handling Session Classes for Predicting ASP.NET Performance Metrics

Ágnes Bogárdi-Mészöly
BUTE, Department of Automation and Applied Informatics
Goldmann György tér 3. IV. em.
1111, Budapest, Hungary
agi@aut.bme.hu

Tihamér Levendovszky
BUTE, Department of Automation and Applied Informatics
Goldmann György tér 3. IV. em.
1111, Budapest, Hungary
tihamer@aut.bme.hu

Hassan Charaf
BUTE, Department of Automation and Applied Informatics
Goldmann György tér 3. IV. em.
1111, Budapest, Hungary
hassan@aut.bme.hu

ABSTRACT

Distributed systems and web applications play an important role in computer science nowadays. The most common consideration is performance, because these systems must provide services with low response time, high availability, and certain throughput level. With the help of performance models, the performance metrics can be determined at the early stages of the development process. The goal of our work is to predict the response time, the throughput and the tier utilization of web applications, based on queueing models handling one and multiple session classes, with MVA and approximate MVA (Mean-Value Analysis) evaluation algorithm, in addition to balanced job bounds calculation. We estimated the model parameters based on one measurement. We implemented the MVA and the approximate MVA evaluation algorithm for closed queueing networks along with the calculation of the balanced job bounds with the help of MATLAB. We have tested a web application with concurrent user sessions in order to validate the models in ASP.NET environment.

Keywords

Web application, web performance, queueing models, performance prediction, and measurements.

1. INTRODUCTION

New frameworks and programming environments have been released to aid the development of complex web-based information systems. These new languages, programming models and techniques are proliferated nowadays, thus, developing such applications is not the only issue anymore: operating, maintenance and performance questions have become of key importance. One of the most important factors is performance, because network systems face a large number of users, they must provide high-availability services with low response time, while they guarantee a certain level of throughput.

These performance metrics depend on many factors. Several papers have investigated various configurable parameters, how they affect the performance of a web-based information system. Statistical methods, hypothesis tests are used in order to retrieve factors influencing the performance. An approach [Sop05a] applies analysis of variance, another [Bog05a] performs independence test.

The performance-related problems emerge very often only at the end of the software project. With the help of properly designed performance models, the performance metrics of a system can be determined at the earlier stages of the development process [Smi90a] [Smi01c]. In the past few years several methods have been proposed to address this goal. A group of them is based on queueing networks or extended versions of queueing networks [Jai91a] [Man02a] [Urg05a]. By solving the queueing model using analytical and simulation solutions, performance metrics can be predicted. Another group uses Petri-nets or generalized stochastic Petri-nets [Ber02b] [Kin99a], which can represent blocking and synchronization aspects much more than queueing networks. The third proposed approach uses a stochastic extension of process algebras, like TIPP (Time Processes and Performability Evaluation) [Her00a], EMPA (Extended Markovian Process Algebra) [Ber98a] and PEPA (Performance Evaluation Process Algebra) [Gil94a].

.NET Technologies 2006
Copyright UNION Agency – Science Press,
Plzen, Czech Republic.

Today one of the most prominent technologies of web-based information systems is Microsoft .NET. Our primary goal was to predict the response time of ASP.NET web applications based on queueing models handling one and multiple session classes, because response time is the only performance metric to which the users are directly exposed. Our secondary goals were to predict the throughput and the utilization of the tiers.

The organization of this paper is as follows. Section 2 covers backgrounds and related work. Section 3 presents our demonstration and validation of the models in the ASP.NET environment, namely, Section 3.1 describes our estimation of the model parameters, Section 3.2 presents our implementation of the MVA and the approximate MVA evaluation algorithm along with the calculation of the balanced job bounds, and Section 3.3 demonstrates our experimental configuration as well as our experimental validation of the models. Finally, we draw conclusions.

2. BACKGROUNDS AND RELATED WORK

Queueing theory [Jai91a] [Kle75a] is one of the key analytical modeling techniques used for computer system performance analysis. Queueing networks and their extensions (such as queueing Petri nets [Kou03a]) are proposed to model web applications [Man02a] [Urg05a] [Urg05b] [Smi00b].

In [Urg05a] [Urg05b], a basic queueing model with some enhancements is presented for multi-tier web applications. An application is modeled as a network of M queues: $Q_1,...,Q_M$ (Figure 1). Each queue represents an application tier, and it is assumed to have a processor sharing discipline, since this discipline closely approximates the scheduling policies applied by most of the operating systems.

A request can take multiple visits to each queue during its overall execution, thus, there are transitions from each queue to its successor and its predecessor as well. Namely, a request from queue Q_m either returns to Q_{m-1} with a certain probability p_m, or proceeds to Q_{m+1} with the probability $1-p_m$. There are only two exceptions: the last queue Q_M, where all the requests return to the previous queue $(p_M = 1)$, and the first queue Q_1, where the transition to the preceding queue denotes the completion of a request. S_m denotes the service time of a request at Q_m $(1 \le m \le M)$.

Internet workloads are usually session-based. The model can handle session-based workloads as an infinite server queueing system Q_0, that feeds the network of queues and forms the closed queueing network depicted in Figure 1. Each active session is in accordance with occupying one server in Q_0. The time spent at Q_0 corresponds to the user think time *Z*. It is assumed that sessions never terminate. Because of the infinite server queueing system, the model captures the independence of the user think times and the service times of the request at the application.

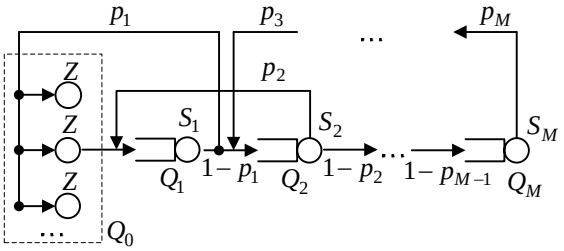

Figure 1. Modeling a multi-tier web application using a queueing network

An enhancement of the baseline model [Urg05a] can handle multiple session classes. Incoming sessions of a web application can be classified into multiple (*C*) classes. *N* is the total number of sessions, and N_c denotes the number of sessions of class *c*, thus, $N = \sum_{c=1}^{C} N_c$. A feasible population with *n* sessions means that the number of sessions within each class *c* is between *0* and N_c, and the sum of the number of sessions in all classes is *n*. In order to evaluate the model, the service times, the visit ratios, and the user think time must be measured on a per-class basis.

The model can be evaluated for a given number of concurrent sessions *N*. A session in the model corresponds to a customer in the evaluation algorithm. The MVA (Mean-Value Analysis) algorithm for closed queueing networks [Jai91a] [Rei80a] iteratively computes the average response time of a request and the throughput. The algorithm introduces the customers into the queueing network one by one, and the cycle terminates when all the customers have been entered.

In addition, the utilization of the queues can be determined from the model, using the utilization law [Jai91a] [Kle75a]. The utilization of the queue *m* is $U_m = XV_mS_m$, where *X* is the throughput and V_m is the visit number (the number of visits to Q_m made by a request during its processing).

The MVA algorithm is only applicable if the queueing network is in product form, namely, the network has to satisfy the conditions of the job flow

balance, one-step behavior, and device homogeneity. Furthermore, the queues are assumed either fixed-capacity service centers or infinite servers, and in both cases exponentially distributed service times are assumed.

MVA is a recursive algorithm. Handling one session class for large values of customers, or if the performance for smaller values is not required, MVA can be too expensive computationally. If we handle multiple session classes, the time and space complexities of MVA are proportional to the number of feasible populations, and this number rapidly grows for relatively few classes and jobs per class. Thus, it can be worth using an approximate MVA algorithm [Rai91a] [Sin05a] or a set of two-sided bounds [Rai91a] [Zah82a].

These bounds referred to as balanced job bounds are based on the issue that a balanced system has a better performance than a similar unbalanced system. A system without a bottleneck device is called a balanced system, in other words, the total service time demands are equal in all queues. The balanced job bounds are very tight, the upper and lower bounds are very close to each other as well as to the real performance. D is the sum of total service demands, $D_{avg} = D/M$ is the average service demand per queue, and $D_{\max}$ is the maximum service demand per queue.

$$\max\left\{ND_{\max} - Z, D + (N-1)D_{avg}\frac{D}{D+Z}\right\}$$
$$\le R(N) \le D + (N-1)D_{\max}\frac{(N-1)D}{(N-1)D+Z}$$

$$\frac{N}{Z + D + (N-1)D_{\max}\frac{(N-1)D}{(N-1)D+Z}}$$
$$\le X(N) \le \min\left\{\frac{1}{D_{\max}}, \frac{N}{Z + D + (N-1)D_{avg}\frac{D}{D+Z}}\right\}$$

The model validation presented in [Urg05a] was executed in a J2EE environment, while in this paper the models are demonstrated and validated in an ASP.NET environment. In order to improve the model, it must be enhanced to handle the limits of the four thread types in .NET thread pool, in addition to the global and the application queue limit [Mei04a], since in previous work [Bog05a] we have proven by statistical methods [Bra87a], that the limits of the four thread types, namely, the *maxWorkerThreads*, *maxIOThreads*, *minFreeThreads*, *minLocalRequest-FreeThreads*, along with the global and application queue limit, namely, the *requestQueueLimit*, *app-RequestQueueLimit* parameters have a considerable effect on performance, in other words, they are performance factors.

3. CONTRIBUTIONS

We have implemented a three-tier ASP.NET test web application (Figure 2). Compared to a typical web application, it has been slightly modified to suit the needs of the measurement process.

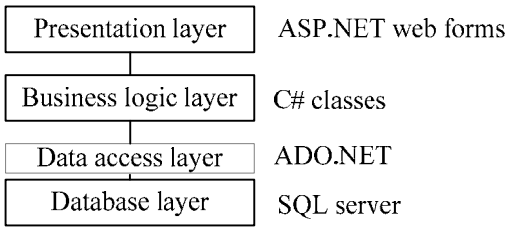

Figure 2. The test web application architecture

Thereafter, we have demonstrated and validated the models in the ASP.NET environment. Firstly, we have estimated the input values of the model parameters from one measurement. Secondly, we have implemented the MVA and the approximate MVA algorithm, along with the calculation of the balanced job bounds with the help of MATLAB, and we have evaluated the model. Finally, we have tested a typical web application with concurrent user sessions, comparing the observed and predicted values in order to validate the models in the ASP.NET environment.

We expect that the baseline model and the model handling multiple session classes can be validated in ASP.NET environment. The thread pool settings and the queue limits are common in the two environments (J2EE and .NET), but the concrete threads (four types, their partitioning in the thread pool) and queues (two types, their placement and configuration) are specific to .NET. Thus, a general model for the two environments with specific extensions is expected, which is more accurate than the baseline model or the model handling multiple session classes. The algorithms presented in [Urg05a] could not be reused directly, because they must be extended.

Estimation of the Model Parameters

The web application was designed in a way that the input values of the model parameters can be determined from the results of one measurement. Each page and class belonging to the presentation, business logic or database was measured separately.

Handling one session class, the input parameters of the model are the number of tiers, the maximum number of customers (simultaneous browser connections), the average user think time $\overline{Z}$, the visit number V_m and the average service time $\overline{S}_m$ for Q_m $(1 \le m \le M)$.

During the measurements the number of tiers was constant (three). The maximum number of customers means that the load was characterized as follows: we started form one simultaneous browser connection then we continued with two, until 52 had been reached. In order to determine the average user think time we averaged the sleep times in the user scenario. To determine V_m we summed the number of requests of each page and class belonging to the given tier in the user scenario. To estimate $\overline{S}_m$ we averaged the service times of each page and class belonging to the given tier.

Handling multiple session classes, the input model parameters are the number of tiers, the number and the maximum number of customers, respectively, on a per-class basis, the average user think time $\overline{Z}_c$, the visit number $V_{m,c}$, and the average service time $\overline{S}_{m,c}$ for Q_m $(1 \le m \le M,\ 1 \le c \le C)$.

There were two classes. The number of sessions for one class was constant 10, while the number of simultaneous browser connections for the other class was varied up to a maximum number of customers. The load was characterized as follows: we started from one simultaneous browser connection then we continued with 5, 10, until 70 had been reached. To determine $\overline{Z}_c$, the sleep times in the user scenario were averaged per class. In order to determine $V_{m,c}$, the number of requests of each page and class belonging to the given tier and class in the user scenario was summed. In order to estimate $\overline{S}_{m,c}$, the service times of each page and class belonging to the given tier and class were averaged.

Model Evaluation

The conditions described in Section 2 have been satisfied: the number of arrivals to a queue equals the number of departures from the queue, the simultaneous job moves are not observed, since the queues have processor sharing discipline, and finally, the service rate of a queue does not depend on the state of the system in any way except for the total queue length. In addition, the queues Q_1, Q_2, Q_3 are fixed-capacity centers, and the Q_0 queue is an infinite server. Therefore, the MVA algorithm can be applicable to evaluate the model (Figure 1) of the test web application (Figure 2), because the model is in a product form.

We implemented the MVA and approximate MVA algorithm for closed queueing networks, in addition the calculation of the balanced job bounds with the help of MATLAB. Our MATLAB scripts can be downloaded from [Mat06a].

When we handle one session class, the inputs of the script are the number of tiers, the maximum number of customers, the average service times, the visit numbers, and the average user think time. When we handle multiple session classes, the inputs the number of tiers, the number and the maximum number of customers, respectively, on a per-class basis the average service times, the visit numbers, and the average user think time. The scripts compute the response times, the throughputs and the tier utilizations up to a maximum number of customers. MVA provides a recursive way, approximate MVA computes these in a few steps, while balanced job bounds method completes in one step.

Model validation

Finally, our experimental configuration and experimental validation of the model in ASP.NET environment are demonstrated.

The web server of our test web application was Internet Information Services (IIS) 6.0 with ASP.NET 1.1 runtime environment, one of the most proliferated technologies among the commercial platforms. The database management system was Microsoft SQL Server 2000 with Service Pack 3. The server runs on a 2.8 GHz Intel Pentium 4 processor with Hyper-Threading technology enabled. It had 1GB of system memory; the operating system was Windows Server 2003 with Service Pack 1. The emulation of the browsing clients and measuring the response time were performed by ACT (Application Center Test), a load generator running on another PC on a Windows XP Professional computer with Service Pack 2 installed. It ran on a 3 GHz Intel Pentium 4 processor with Hyper-Threading technology enabled, and it also had 1GB system memory. The connection among the computers was provided by a 100 Mb/s network.

ACT [Ald03a] is a well-usable stress testing tool included in Visual Studio .NET Enterprise and Architect Editions. The test script can be recorded or manually created. Virtual users send a list of HTTP requests to the web server concurrently. Each test run takes 2 minutes and 10 seconds warm-up time for the load to reach a steady-state. In the user scenario, sleep times are included to simulate the realistic usage of the application.

When we handle one session class, while the number of simultaneous browser connections varied, the average response time and throughput per class were measured (Figure 3).

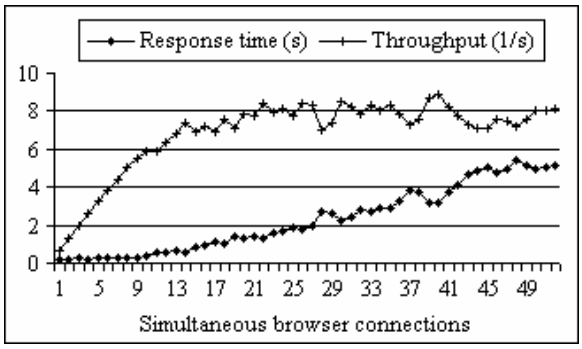

Figure 3. The observed response times and throughputs handling one session class

Handling multiple session classes, there were two classes of sessions: a database reader and a database writer. The number of simultaneous browser connections of one class was fixed at 10, while the number of simultaneous browser connections of the other class varied, and we measured the average response time and throughput per class (Figure 4).

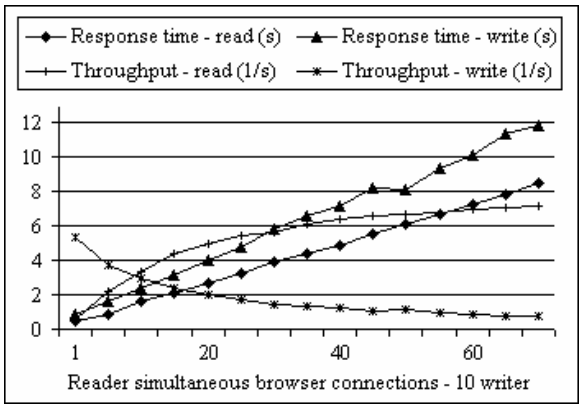

Figure 4. The observed response times and throughputs handling multiple session classes

The results presented in Figure 3 and in Figure 4 correspond to the common shape of response time and throughput performance metrics. Increasing the number of concurrent (reader) clients, the (reader) throughput (served requests per second) grows linearly, while the average (reader) response time advances barely. After the saturation the (reader) throughput remains approximately constant, and an increase in the (reader) response time can be observed. In the overloaded phase, the (reader) throughput falls, while the (reader) response time becomes unacceptably high.

Handling one session class, we experimentally validated the model to demonstrate its ability to predict the response time and the throughput of ASP.NET web applications with MVA (Figure 5), and approximate MVA algorithm. We have found that the model handling one session class predicts the response time and throughput acceptably.

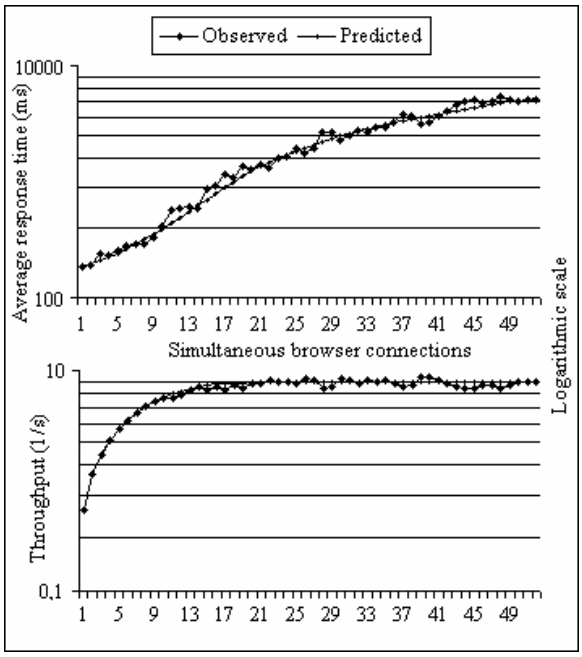

Figure 5. The observed and predicted response times and throughputs handling one session class with MVA

Moreover, from the model, the utilization of the tiers can be predicted. The results are depicted in Figure 6. The presentation tier is the first that becomes congested. The utilization of the database queue is the second (29%), and the utilization of the business logic queue is the last one (17%).

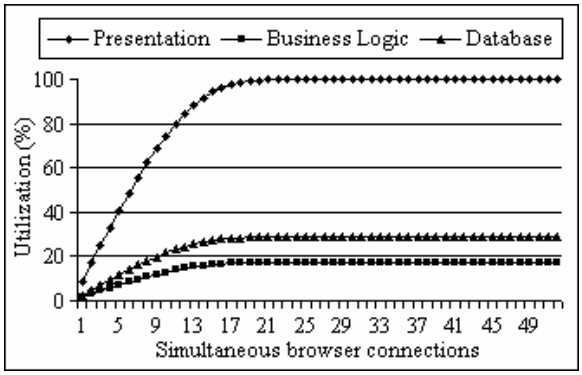

Figure 6. The tier utilization handling one session class with MVA

Thereafter, we demonstrate that the response time, the throughput and the tier utilization of ASP.NET web applications move within tight upper and lower bounds (Figure 7, Figure 8). We have found that the

response time, the throughput, and the queue utilization from the observations fell into the upper and lower bounds. Thus, the balanced job bounds handling one session class predict the response time, the throughput, and the utilization of the tiers acceptably.

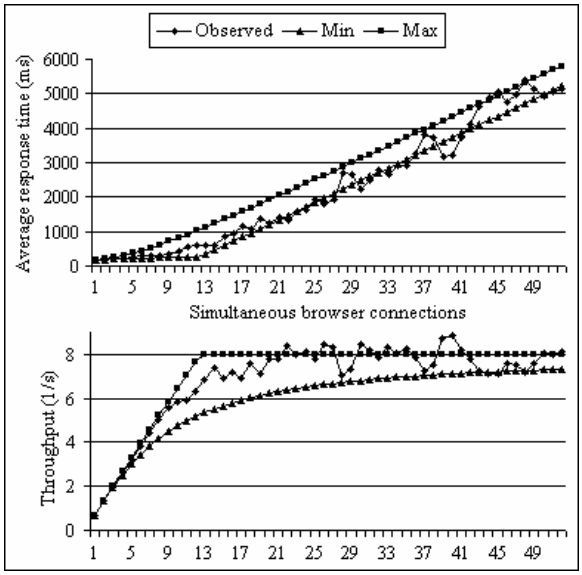

Figure 7. The observed and predicted response times and throughputs handling one session class with balanced job bounds

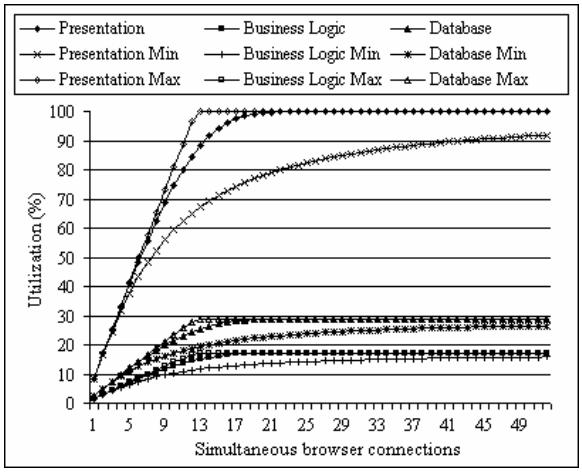

Figure 8. The tier utilization handling one session class with balanced job bounds

Finally, the model handling multiple session classes was experimentally validated. We have found that the model predicts the response time and throughput with approximate MVA acceptably (Figure 9). While the presentation tier is congested, the utilization of the database queue is about 84%, and the utilization of the business logic queue is about 16% (Figure 10). We have found that the response time, the throughput, and the utilization from the observations as well as from the approximate MVA fell into the upper and lower bounds. Hence, the balanced job bounds predict the response time, the throughput, and the utilization acceptably (Figure 11).

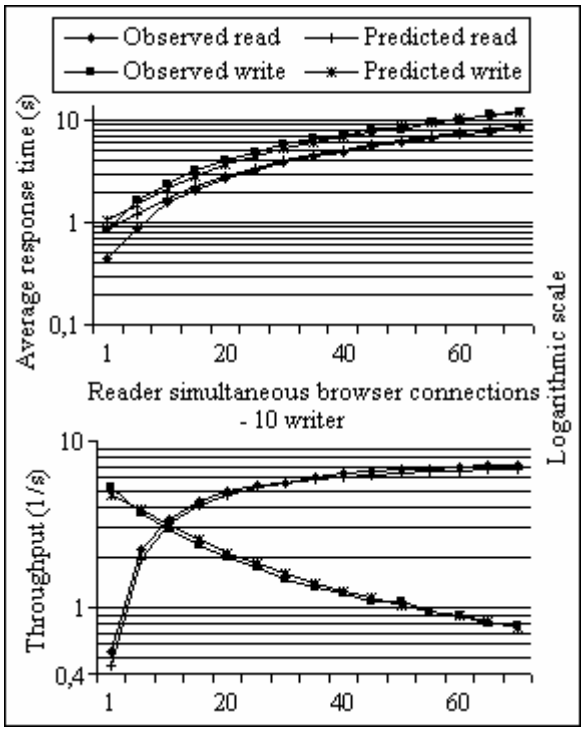

Figure 9. The observed and predicted response times and throughputs handling multiple session classes with approximate MVA

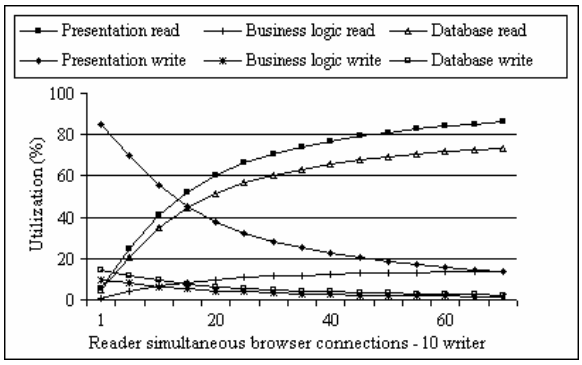

Figure 10. The tier utilization handling multiple session classes with approximate MVA

4. CONCLUSIONS AND FUTURE WORK

We have demonstrated and validated queueing models handling one and multiple session classes in ASP.NET environment, namely, the input model parameters were estimated from one measurement, the MVA and approximate MVA evaluation algorithm, in addition the calculation of the balanced job bounds were implemented with the help of MATLAB, and a measurement process was executed in order to experimentally validate the models.

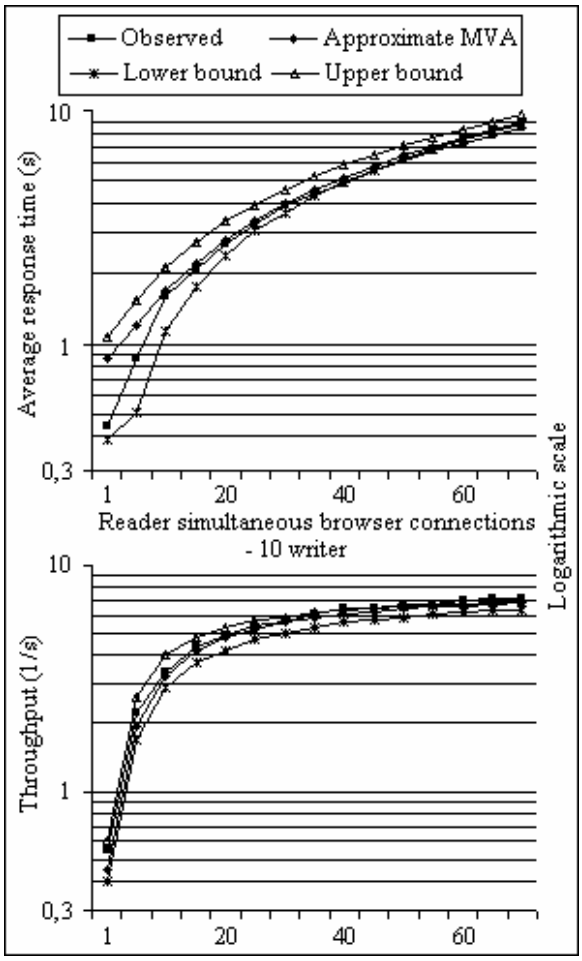

Figure 11. The observed and predicted response times and throughputs handling multiple session classes with balanced job bounds

Our results have shown that the models handling one and multiple session classes predict the response time and the throughput acceptably with MVA and approximate MVA evaluation algorithm, along with the calculation of balanced job bounds. Furthermore, the presentation tier is the first to become congested. The utilization of the database tier is the second one, and the utilization of the business logic queue is the last one.

In order to improve the model, the limits of the four thread types in .NET thread pool, the global and application queue limits must be handled along with other features. These extensions of the model and the validation of the enhanced models, as well as the validation of the models in ASP.NET 2.0 environment are subjects of future work.

5. REFERENCES

[Ald03a] Aldous, J., and Finnel, L. Performance Testing Microsoft .NET Web Applications. Microsoft Press, 2003.

[Ber98a] Bernardo, M., and Gorrieri, R. A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time. Journal of Theoretical Computer Science, Vol. 202, pp. 11-54, 1998.

[Ber02b] Bernardi, S., Donatelli, S., and Merseguer, J. From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models. In Proceedings of ACM International Workshop Software and Performance. Rome, Italy, pp. 35-45, 2002.

[Bog05a] Bogárdi-Mészöly, Á., Szitás, Z., Levendovszky, T., Charaf, H. Investigating Factors Influencing the Response Time in ASP.NET Web Applications. Proceedings of Lecture Notes in Computer Science, 3746, pp. 223-233, 2005.

[Bra87a] Brase, C.H., and Brase, C.P. Understandable Statistics. D. C. Heath and Company, 1987.

[Gil94a] Gilmore, A.S., and Hillston, J. The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling. In Proceedings of Seventh International Conference Modelling Techniques and Tools for Performance Evaluation, pp. 353-368, 1994.

[Her00a] Herzog, U., Klehmet, U., Mertsiotakis, V., and Siegle, M. Compositional Performance Modelling with the TIPPtool. Journal of Performance Evaluation, Vol. 39, pp. 5-35, 2000.

[Jai91a] Jain, R. The Art of Computer Systems Performance Analysis. John Wiley and Sons, 1991.

[Kin99a] King, P., and Pooley, R. Derivation of Petri Net Performance Models from UML Specifications of Communication Software. In Proceedings of 25th UK Performance Eng. Workshop, 1999.

[Kle75a] Kleinrock, L. Queueing Systems, Volume 1: Theory. John Wiley and Sons, 1975.

[Kou03a] Kounev, S., Buchmann, A. Performance Modelling of Distributed E-Business Applications using Queueing Petri Nets. Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software, Austin, Texas, USA, 2003.

[Man02a] Manescé, D.A., and Almeida, V.A.F. Capacity Planning for Web Services. Prentice Hall, 2002.

[Mat06a] Our MATLAB scripts can be downloaded from - http://avalon.aut.bme.hu/~agi/research/

[Mei04a] Meier, J.D., Vasireddy, S., Babbar, A., and Mackman, A. Improving .NET Application Performance and Scalability (Patters & Practices). Microsoft Corporation, 2004.

[Rei80a] Reiser, M., and Lavenberg, S.S. Mean-Value Analysis of Closed Multichain Queuing Networks. Journal of Association for Computing Machinery, Vol. 27, pp. 313-322, 1980.

[Sin05a] Sinclair, B., Mean Value Analysis. Computer Systems Performance Handout, 2005.

[Smi90a] Smith, C.U. Performance Engineering of Software Systems. Addison-Wesley, 1990.

[Smi00b] Smith, C.U., Williams, L.G. Building responsive and scalable web applications. Computer Measurement Group Conference, Orlando, FL, USA, pp. 127-138, 2000.

[Smi01c] Smith, C.U., and Williams, L. G. Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley, 2001.

[Sop05a] Sopitkamol, M., and Menascé, D.A. A Method for Evaluating the Impact of Software Configuration Parameters on E-Commerce Sites. In Proceedings of the ACM 5th International Workshop on Software and Performance, Palma, Illes Balears, Spain, pp. 53-64, 2005.

[Urg05a] Urgaonkar, B. Dynamic Resource Management in Internet Hosting Platforms. Dissertation, Massachusetts, 2005.

[Urg05b] Urgaonkar, B., Pacifici, G., Shenoy, P., Speitzer, M., and Tantawi, A. An Analytical Model for Multi-tier Internet Services and its Applications. Journal of ACM SIGMETRICS Performance Evaluation Review, Vol. 33, No. 1, pp. 291-302, 2005.

[Zah82a] Zahorjan, J., Sevcik, K.C., Eager D.L., and Galler, B. Balanced Job Bound Analysis of Queueing Networks. Journal of Communications of the ACM, Vol. 25, No. 2, pp. 134-141, 1982.