Balanced Job Bounds Calculation for Approximating ASP.NET Performance Factors

Ágnes Bogárdi-Mészöly, Zoltán Szitás, Tihamér Levendovszky, Hassan Charaf

Department of Automation and Applied Informatics, Department of Electronics Technology, Budapest University of Technology and Economics
Goldmann György tér 3, H-1111 Budapest, Hungary
agi@aut.bme.hu, szitas@ett.bme.hu, tihamer@aut.bme.hu, 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 cost-effective and high-availability services in the long term, thus they have to be scaled to meet the expected load. Performance measurements can be the base for performance modeling and prediction. 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 a queueing model with MVA evaluation algorithm and with balanced job bounds calculation. We estimated the model parameters (number of customers, number of tiers, user think time, visit numbers, and service times) based on a measurement. We implemented the evaluation algorithm and the calculation of the balanced job bounds with the help of MATLAB. We tested a web application with concurrent user sessions to validate the model in ASP.NET environment.

Keywords: Systems engineering, web performance, queueing models, performance prediction, and measurements

1 Introduction

New frameworks and programming environments were released to aid the development of complex web applications and to support building services offering dynamic content. 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 in a cost-effective way.

Performance measurements can serve as a basis for performance modeling and prediction. 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 [1] [2]. 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 [3] [4] [5] [6]. 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 [7] [8], which can represent blocking and synchronization aspects much more than queueing networks. The third proposed approach uses stochastic extension of process algebras, like TIPP (Time Processes and Performability Evaluation) [9], EMPA (Extended Markovian Process Algebra) [10] and PEPA (Performance Evaluation Process Algebra) [11].

Today one of the most prominent technologies of distributed systems and web applications is Microsoft .NET [12]. The commonly used performance metrics are response time, throughput, and resource utilization. Our primary goal was to predict the response time of ASP.NET web applications based on a queueing model, 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 of web applications.

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

2 Backgrounds and Related Work

Queueing theory [3] [4] is one of the key analytical modeling techniques used for computer system performance analysis. Queueing networks and their extensions (like queueing Petri nets [13]) are proposed to model web applications [5] [6] [13] [14].

In [6] 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 the most 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_i either returns to Q_{i-1} with a certain probability p_i, or proceeds to Q_{i+1} with the probability $1 - p_i$. 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_i denotes the service time of a request at Q_i $(1 \le i \le M)$.

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.

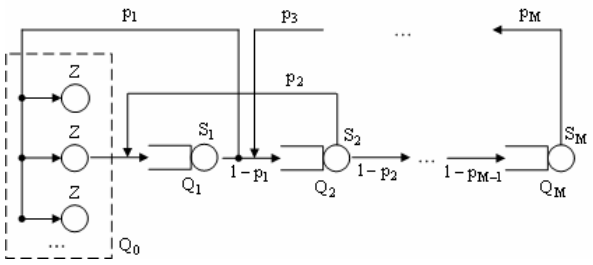

Figure 1
Modeling a multi-tier web application using a queueing network

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 [3] [15] iteratively computes the average response time of a request and the throughput. The algorithm introduces 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 [3]. The utilization of the *i*th queue is $U_i = XV_iS_i$, where X is the throughput.

The model can handle multiple visits to a tier regardless of whether they occur sequentially or in parallel, since the evaluation algorithm uses visit numbers instead of transition probabilities. The visit number V_i is the number of visits to Q_i made by a request during its processing. The visit numbers can be computed

from the transition probabilities. They provide an alternative representation of the queueing network [3].

The MVA algorithm is only applicable if the queueing network is in product form, namely the network have to satisfy the conditions of the job flow balance, one-step behavior, and device homogeneity. In addition, 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. For large values of N or if the performance for smaller values of N is not required, the algorithm can be too expensive computationally. In order to avoid recursion, several approximate analysis techniques have been developed [3] and a set of two-sided bounds has been derived [16] [3]. 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 in all queues are equal. The balanced job bounds are very tight, the upper and lower bounds are very close to each other and 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.

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

$$\le R(N) \le D + (N-1)D_{\max}\frac{(N-1)D}{(N-1)D+Z} \tag{1}$$

$$\frac{N}{Z + D + (N-1)D_{\max}\dfrac{(N-1)D}{(N-1)D+Z}}$$

$$\le X(N) \le \min\left\{\frac{1}{D_{\max}}, \frac{N}{Z + D + (N-1)D_{avg}\dfrac{D}{D+Z}}\right\} \tag{2}$$

The model validation presented in [6] was executed in a J2EE environment, while in this paper the model was 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 [17], since in previous work [18] we have proven by statistical methods (chi square test of independence [19]) that the limits of the four thread types, namely, the *maxWorkerThreads, maxIOThreads, minFreeThreads, minLocalRequestFreeThreads* 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). It has been slightly modified compared to a typical web application to suit the needs of the measurement process.

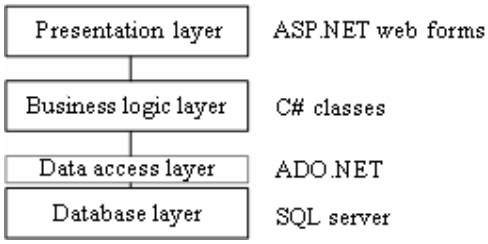

Figure 2
The test web application architecture

Thereafter, with the help of the performed measurements, we demonstrate and validate the model in ASP.NET environment, namely, we estimate the input values of the model parameters from the results of one measurement, implement the MVA algorithm and the calculation of the balanced job bounds with the help of MATLAB [20], finally, evaluate and validate the model in ASP.NET environment.

3.1 Estimation of the Model Parameters

The input parameters of the model are the maximum number of customers (simultaneous browser connections), the number of tiers, the average user think time $\overline{Z}$, the visit number V_i and the average service time $\overline{S}_i$ for Q_i $(1 \le i \le M)$.

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

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 was reached. The number of tiers was constant (three). In order to determine the average user think time we averaged the sleep times in the user scenario. To determine V_i we summed the number of requests of each page and class belonging to the given tier in the user scenario. To estimate $\overline{S}_i$ we averaged the service times of each page and class belonging to the given tier (Table 1).

Input	Constant	Maximum number of customers		52
		Number of tiers		3
	Estimated	User think time		1,296 s
		Service time	Presentation	2,79 ms
			Business Logic	1,81 ms
			Database	3,02 ms
		Visit number	Presentation	45
			Business Logic	12
			Database	12

Table 1
Model parameters

3.2 Model Evaluation by a MATLAB Implementation of the MVA Algorithm and the Balanced Job Bounds

The MVA algorithm can be applicable to evaluate the model (Figure 1, Table 1) of the test web application (Figure 2), because the model is in a product form. Since 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 Q_1, Q_2, Q_3 queues are fixed-capacity centers, and the Q_0 queue is an infinite server.

We implemented the MVA algorithm for closed queueing networks and the calculation of the balanced job bounds with the help of MATLAB. Our MATLAB script can be downloaded from [21].

Given the maximum number of customers, the number of tiers, the average service times, the visit numbers, and the average user think time (Table 1), the script recursively computes the average response times, the throughputs and the utilizations of the tiers up to a maximum number of customers, furthermore, it calculates in one step the balanced job bounds of the response time, the throughput and the utilizations of the tiers up to a maximum number of customers.

3.3 Model Validation

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

3.3.1 Experimental Configuration

The web server of our test web application was Internet Information Services (IIS) 6.0 [22] with ASP.NET 1.1 runtime environment [23], 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 the measuring of the response time was 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 runs 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 [24] 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. While the number of simultaneous browser connections varied, we measured the average response time and throughput (Figure 3).

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

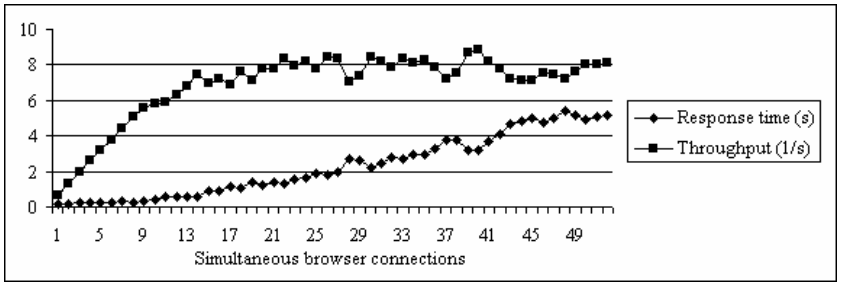

Figure 3
The observed response time and throughput

3.3.2 Performance Prediction with MVA

We experimentally validated the model to demonstrate its ability to predict the response time and the throughput of ASP.NET web applications. We have found that the model predicts the response time (Figure 4) and throughput (Figure 5) well. In order to improve the model, it must be enhanced to handle the limits of the four thread types in .NET thread pool and it may be enhanced by other features. These are subjects of future work.

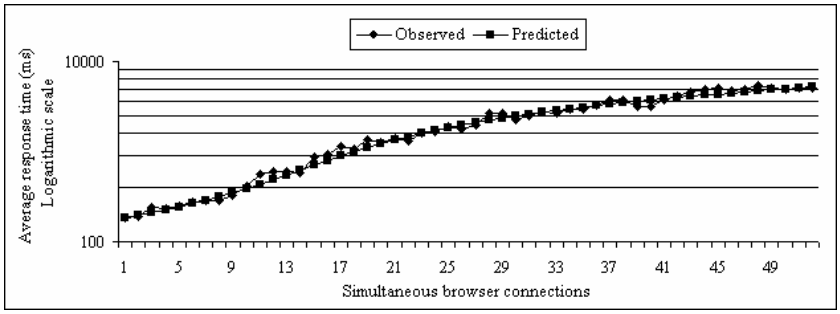

Figure 4
The observed and predicted response times

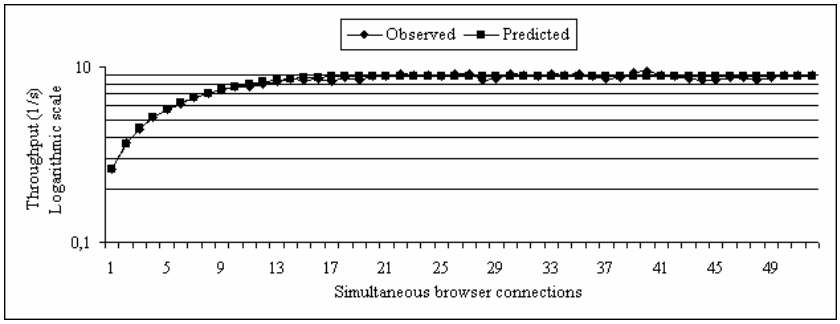

Figure 5
The observed and predicted throughputs

In addition, 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, and the utilization of the business logic queue is the last one. When the presentation tier is congested, the utilization of the database queue is about 29% and the utilization of the business logic queue is about 17%.

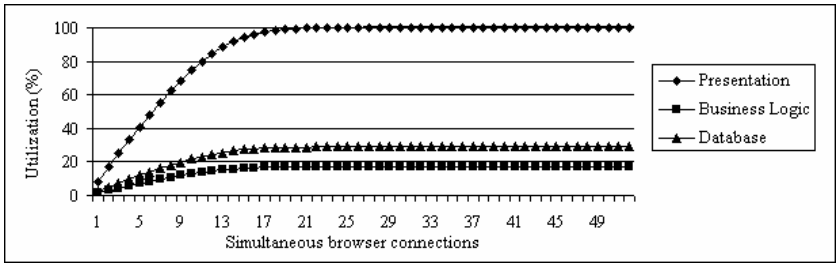

Figure 6

The utilization of the queues

3.3.3 Performance Prediction with Balanced Job Bounds

Finally, we demonstrate that the response time, the throughput and the utilization of the tiers of ASP.NET web applications move within tight upper and lower bounds (Figure 7, Figure 8, and Figure 9). We have found that the response time, the throughput, and the utilization of the queues from the MVA recursive algorithm and from the observations are fallen into the upper and lower bounds. Thus the balanced job bounds predict the response time, the throughput, and the utilization of the tiers acceptably well.

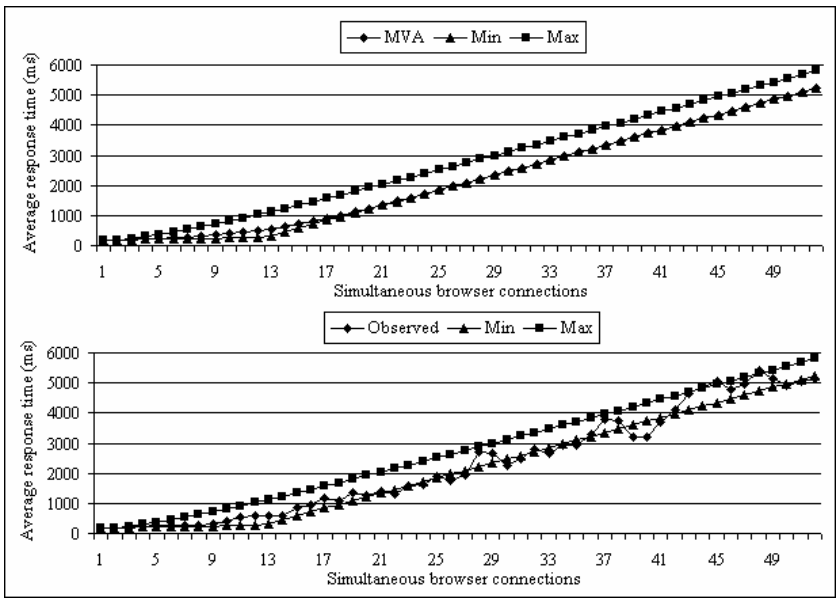

Figure 7

Balanced job bounds of the response time

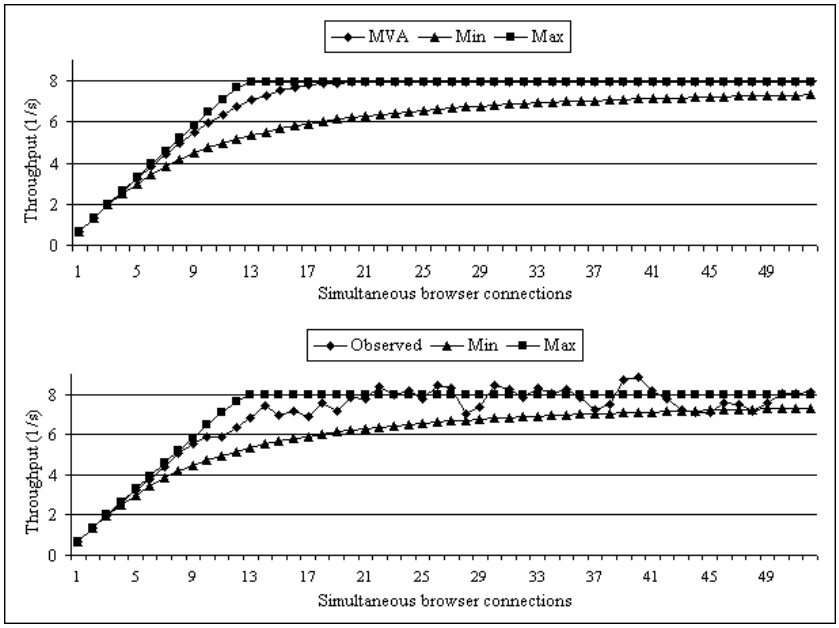

Figure 8
Balanced job bounds of the throughput

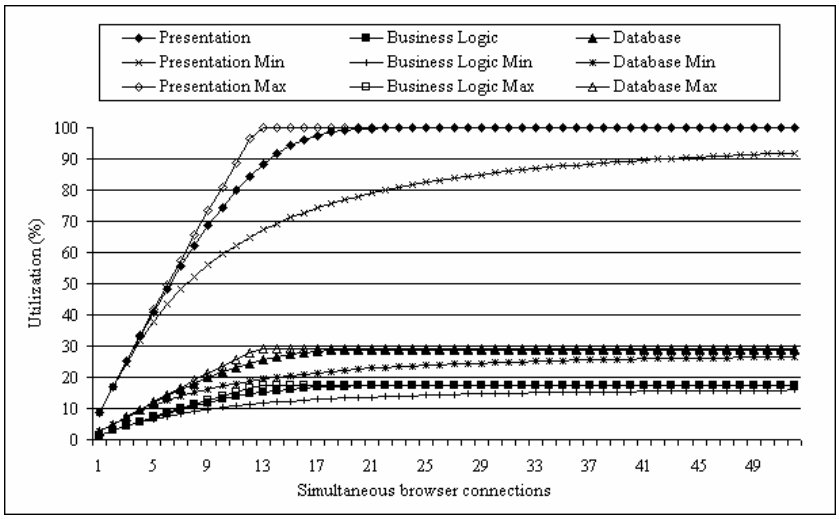

Figure 9
Balanced job bounds of the utilization of the tiers

Conclusions and Future Work

We have demonstrated and validated a queueing model in ASP.NET environment, namely, the input model parameters were estimated from a measurement, the MVA recursive evaluation algorithm and the one-step 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 model.

Our results have shown that the model predicts the response time and the throughput well with MVA evaluation algorithm and with calculation of balanced job bounds as well. Furthermore, the presentation tier is the first to become congested. The utilization of the database tier is the second, and the utilization of the business logic queue is the last one.

To improve the model, the limits of the four thread types in .NET thread pool must be handled and other features may be handled. These extensions of the model and the validation of the enhanced models are subjects of future work.

References

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

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

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

[4] A. Willig: Performance Evaluation Techniques, Lecture Notes, Potsdam, 2004, pp. 281

[5] D.A. Manescé, & V.A.F. Almeida: Capacity Planning for Web Services, Prentice Hall, 2002

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

[7] S. Bernardi, S. Donatelli, & J. Merseguer: From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models, ACM Proc. International Workshop Software and Performance, 2002, pp. 35-45

[8] P. King, & R. Pooley: Derivation of Petri Net Performance Models from UML Specifications of Communication Software, Proc. 25th UK Performance Eng. Workshop, 1999

[9] U. Herzog, U.Klehmet, V. Mertsiotakis, & M. Siegle: Compositional Performance Modelling with the TIPPtool, Performance Evaluation, 39, 2000, pp. 5-35

[10] M. Bernardo, & R. Gorrieri: A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time, Theoretical Computer Science, 202, 1998, pp. 1-54

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

[12] Microsoft .NET Homepage - http://www.microsoft.com/net/

[13] S. Kounev, & A. Buchmann: Performance Modelling of Distributed E-Business Applications using Queueing Petri Nets, IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'03), 2003

[14] C.U. Smith, & L.G. Williams: Building responsive and scalable web applications, Computer Measurement Group Conference, 2000, pp. 127-138

[15] M. Resiser, & S. S. Lavenberg: Mean-Value Analysis of Closed Multichain Queuing Networks, Journal of the Association for Computing Machinery, 27, 1980, pp. 313-322

[16] J. Zahorjan, K. C. Sevcik, D. L. Eager, & B. Galler: Balanced Job Bound Analysis of Queueing Networks, Communications of the ACM, 25(2), 1982, pp. 134-141

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

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

[19] C. H. Brase, & C.P. Brase: Understandable Statistics, D. C. Heath and Company, 1987

[20] MATLAB - http://www.mathworks.com/products/matlab/

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

[22] Internet Information Services 6.0 - http://www.microsoft.com/WindowsServer2003/iis/default.mspx

[23] ASP.NET Homepage - http://asp.net/

[24] J. Aldous, & L. Finnel: Performance Testing Microsoft .NET Web Applications, Microsoft Press, 2003