Predicting the Performance of ASP.NET Web-Based Information Systems with Optimized Algorithms

Ágnes Bogárdi-Mészöly *
agi@aut.bme.hu

Tihamér Levendovszky, Hassan Charaf*
tihamer@aut.bme.hu, hassan@aut.bme.hu

Abstract: Web-based information systems 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 a queueing model handling multiple session classes with approximate MVA (Mean-Value Analysis) evaluation algorithm and with balanced job bounds calculation. We estimated the model parameters based on one measurement. We implemented the approximate 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: Web-based information systems, web performance, queueing models, performance prediction, and measurements

1 Introduction

New frameworks and programming environments were 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 ad performance questions have become of key importance. One of the most important factors is performance, because network systems face a large number of users, and 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, such as hypothesis tests are used in order to retrieve factors influencing the performance. An approach [1] applies analysis of variance, another [2] 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. Several methods have been proposed to address this goal recently. A group of them is based on queueing networks or extended versions of queueing networks [3] [4] [5]. By solving the queueing model using analytical and simulation solutions, performance metrics can be predicted. Another group uses

* Budapest University of Technology and Economics, Department of Automation and Applied Informatics, Goldmann György tér 3. IV. em., H-1111 Budapest

Petri-nets or generalized stochastic Petri-nets [6] [7], 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) [8], EMPA (Extended Markovian Process Algebra) [9], and PEPA (Performance Evaluation Process Algebra) [10].

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 a queueing model handling 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 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 approximate evaluation algorithm along with the calculation of the balanced job bounds, and Section 3.3 demonstrates our experimental configuration and experimental validation of the model. Finally, we draw conclusions.

2 Backgrounds and Related Work

Queueing theory [3] is one of the key analytical modeling techniques used for computer system performance analysis. Queueing networks and their extensions (such as queueing Petri nets [11]) are proposed to model web applications [4] [5] [11] [12].

In [12], a basic queueing model with some enhancements is presented for multi-tier web applications. An application is modeled as a network of M queues: $Q_1, ..., Q_M$ (Fig. 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 ($1 \leq m \leq M$).

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 Fig. 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 the 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.

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] [13] 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 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

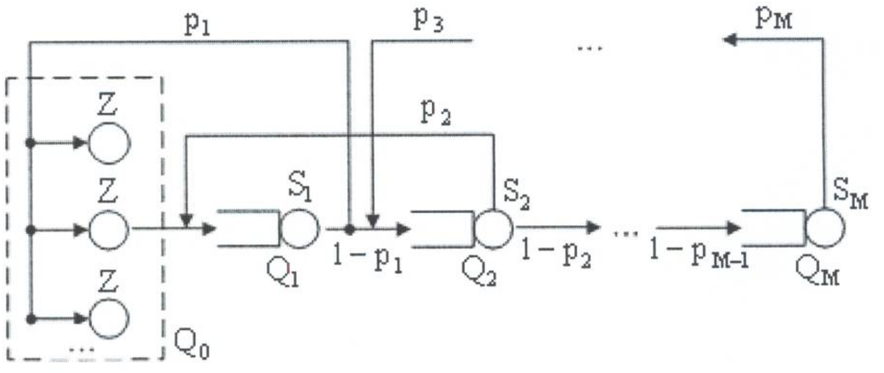

Fig. 1: Modeling a multi-tier web application using a queueing network

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. In addition, the queues are assumed either fixed-capacity service centers or infinite servers, and in both cases exponentially distributed service times are assumed.

An enhancement of the baseline model [12] 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 as previously, 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 time and space complexities of the recursive MVA algorithm are proportional to the number of feasible populations, and this number rapidly grows for relatively few classes and jobs per class. Thus, for the evaluation of a model with multiple session classes, it is worth using an approximate MVA algorithm [3] [14] or a set of two-sided bounds [3] [15] without having to iterate through all smaller feasible populations. These bounds referred to as balanced job bounds are based on the principle 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 and to the real performance.

The model validation presented in [12] was executed in a J2EE (Java 2 Enterprise Edition) environment, while in this paper the model is 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, along with the global and application queue limit [16], since in previous work [2] we have proven by statistical methods that the limits of the four thread types and the two queues, namely, the *maxWorkerThreads*, *maxIOThreads*, *minFreeThreads*, *minLocalRequestFreeThreads*, *requestQueueLimit*, *appRequestQueueLimit* 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 (Fig. 2). Compared to a typical web application, it has been slightly modified to suit the needs of the measurement process.

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

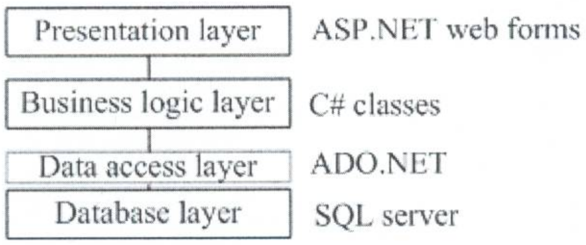

Fig. 2: The test web application architecture

3.1 Estimation of Model Parameters

The input parameters of the model are the number of tiers, the number and the maximum number of customers (simultaneous browser connections), respectively, on a per-class basis the average user think time $\bar{Z}_c$, the visit number $V_{m,c}$ and the average service time $\bar{S}_{m,c}$ for $(1 \leq m \leq M, 1 \leq c \leq C)$. 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 were measured separately.

During the measurements the number of tiers was constant (three). 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 varied up to a maximum number of customers. The maximum number of customers means that the load was characterized as follows: we started from one simultaneous browser connection then we continued with 5, 10, until 70 was reached. In order to determine $\bar{Z}_c$, we averaged the sleep times in the user scenario per class. To determine $V_{m,c}$, we summed the number of requests of each page and class belonging to the given tier and class in the user scenario. To estimate $\bar{S}_{m,c}$, we averaged the service times of each page and class belonging to the given tier and class.

3.2 Model Evaluation

The MVA algorithm can be applicable to evaluate the model (Fig. 1) of the test web application (Fig. 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 queues Q_1, Q_2, Q_3 are fixed-capacity centers, and the Q_0 queue is an infinite server.

We implemented the approximate MVA algorithm for closed queueing networks and the calculation of the balanced job bounds with the help of MATLAB. Our MATLAB scripts can be downloaded from [17]. The inputs of the script are 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. Using approximate MVA, the script computes the average response times, the throughputs and the tier utilizations per class up to a maximum number of customers in a few steps. Furthermore, it calculates the balanced job bounds of the response time in one step, along with the throughput and the tier utilizations 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.

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 [18] 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. 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 (Fig. 3).

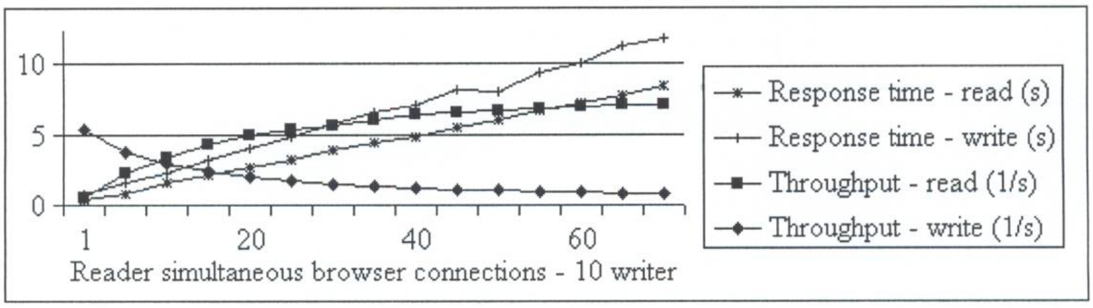

Fig. 3: The observed response times and throughputs

The results presented in Fig. 3 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 unacceptable high.

We have experimentally validated the model to demonstrate its ability to predict the response time and the throughput of ASP.NET web applications with approximate MVA algorithm. We have found that the model predicts the response time and throughput acceptably (Fig. 4). In addition, from the model, the utilization of the tiers can be predicted. The results are depicted in Fig. 5. 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.

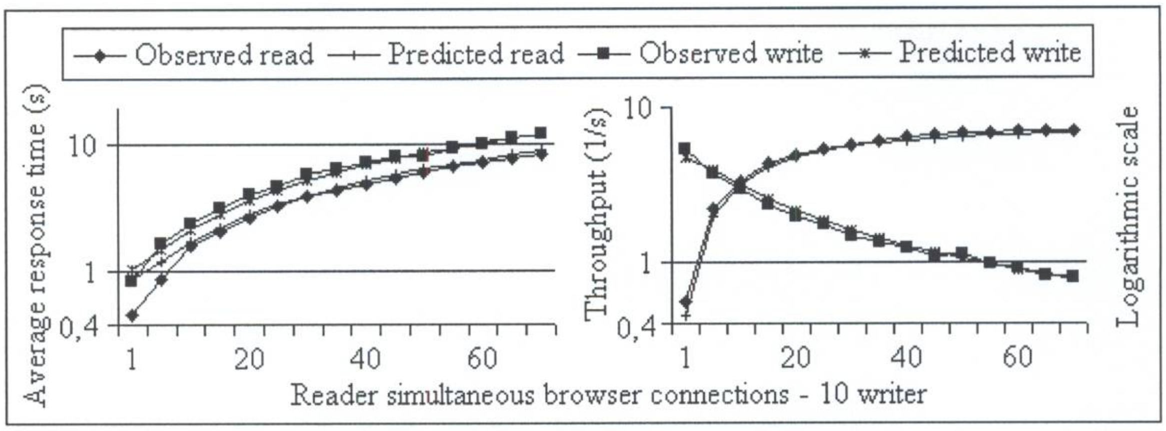

Fig. 4: The observed and predicted response times and throughputs with approximate MVA

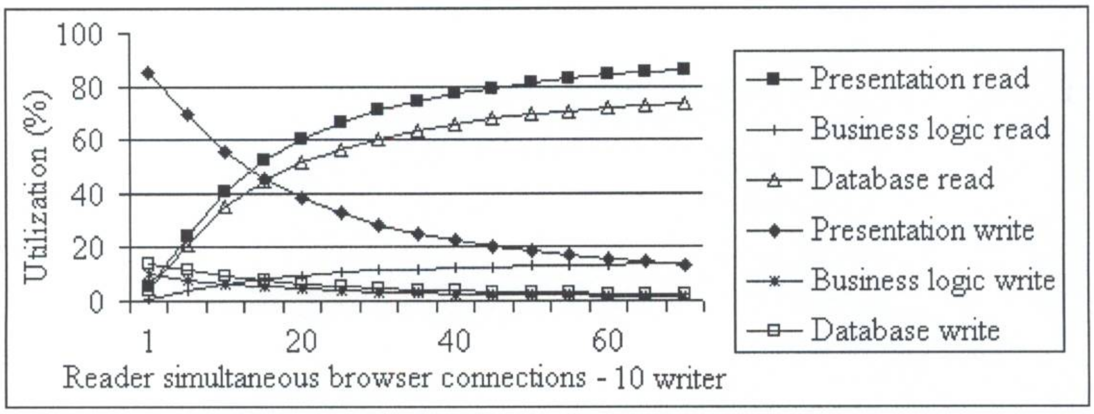

Fig. 5: The tier utilization with approximate MVA

We demonstrate that the response time, the throughput and the tier utilization of ASP.NET web applications move within tight upper and lower bounds (Fig. 6, and Fig. 7). We have found that the response time, the throughput, and the queue utilization from the approximate MVA 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.

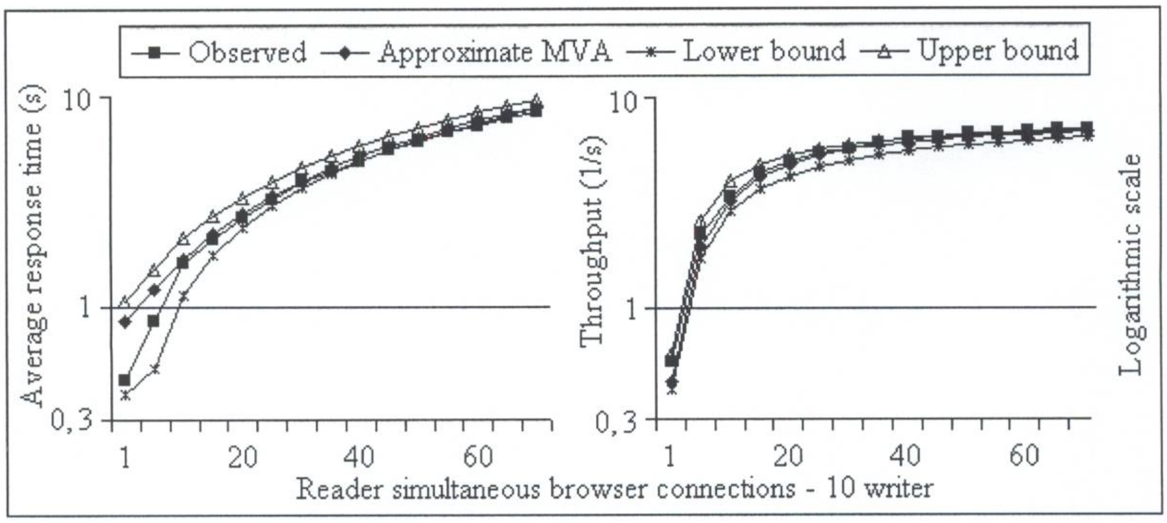

Fig. 6: The observed and predicted response times and throughputs with balanced job bounds

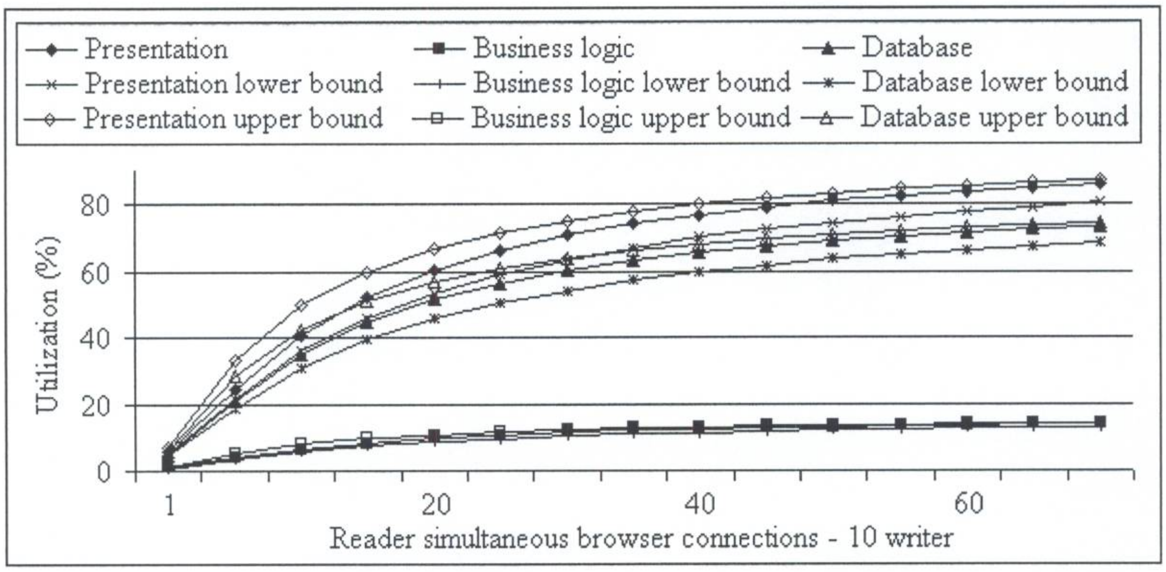

Fig. 7: The tier utilization with balanced job bounds

4 Conclusions and Future Work

We have demonstrated and validated a queueing model handling multiple session classes in ASP.NET environment, namely, the input model parameters were estimated from one measurement, the approximate MVA 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 acceptably with both approximate MVA evaluation algorithm and the 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, moreover, the global and application queue limit must be handled along with other features. These extensions of the model and the validation of the enhanced models are subjects of future work.

Bibliography

1. M. Sopitkamol, D.A. Menascé: *A Method for Evaluating the Impact of Software Configuration Parameters on E-Commerce Sites.* Proceedings of the ACM 5th International Workshop on Software and Performance, Palma, Illes Balears, Spain, 2005, pp. 53-64
2. Á. Bogárdi-Mészöly, Z. Szitás, T. Levendovszky, H. Charaf: *Investigating Factors Influencing the Response Time in ASP.NET Web Applications.* Proceedings of Lecture Notes in Computer Science, 3746, 2005, pp. 223-233
3. R. Jain: *The Art of Computer Systems Performance Analysis.* John Wiley and Sons, 1991
4. D.A. Manescé, V.A.F. Almeida: *Capacity Planning for Web Services.* Prentice Hall, 2002
5. C.U. Smith, L.G. Williams: *Building responsive and scalable web applications.* Computer Measurement Group Conference, Orlando, FL, USA, 2000, pp. 127-138
6. S. Bernardi, S. Donatelli, J. Merseguer: *From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models.* Proceedings of ACM International Workshop Software and Performance, Rome, Italy, 2002, pp. 35-45
7. P. King, R. Pooley: *Derivation of Petri Net Performance Models from UML Specifications of Communication Software.* Proceedings of 25th UK Performance Eng. Workshop, 1999
8. U. Herzog, U.Klehmet, V. Mertsiotakis, M. Siegle: *Compositional Performance Modelling with the TIPPtool.* Journal of Performance Evaluation, 39, 2000, pp. 5-35
9. M. Bernardo, R. Gorrieri: *A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time.* Journal of Theoretical Computer Science, 202, 1998, pp. 11-54
10. A S. Gilmore, J. Hillston: *The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling.* Proceedings of Seventh International Conference Modelling Techniques and Tools for Performance Evaluation, 1994, pp. 353-368
11. S. Kounev, A. Buchmann: *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
12. B. Urgaonkar, G. Pacifici, P. Shenoy, M. Speitzer, A. Tantawi: *An Analytical Model for Multi-tier Internet Services and its Applications.* Journal of ACM SIGMETRICS Performance Evaluation Review, 33(1), 2005, pp. 291-302
13. M. Reiser, S.S. Lavenberg: *Mean-Value Analysis of Closed Multichain Queuing Networks.* Journal of Association for Computing Machinery, 27, 1980, pp. 313-322
14. B. Sinclair: *Mean Value Analysis.* Computer Systems Performance Handout, 2005
15. J. Zahorjan, K.C. Sevcik, D.L. Eager, B. Galler: *Balanced Job Bound Analysis of Queueing Networks.* Journal of Communications of the ACM, 25(2), 1982, pp. 134-141
16. J.D. Meier, S. Vasireddy, A. Babbar, A. Mackman: *Improving .NET Application Performance and Scalability (Patters & Practices).* Microsoft Corporation, 2004
17. Our MATLAB scripts can be downloaded from -
`http://avalon.aut.bme.hu/~agi/research/`
18. J. Aldous, L. Finnel: *Performance Testing Microsoft .NET Web Applications.* Microsoft Press, 2003