Novel Models and Algorithms for Queue Limit Modeling

Ágnes Bogárdi-Mészöly, András Rövid, Tihamér Levendovszky

Budapest University of Technology and Economics, Hungary, agi@aut.bme.hu
Budapest Tech Polytechnical Institution, Hungary, rovid.andras@nik.bmf.hu
Vanderbilt University, USA, tihamer@isis.vanderbilt.edu

Abstract: There is a demand for researching the ways how performance models of web-based software systems can become more efficient. This paper deals with the establishment and investigation of the evaluation and prediction techniques applying the identified queue limit performance factors. Novel models and algorithms are provided to model the queue limit. The proposed models and algorithms can be applied to performance prediction in ASP.NET environment. The validity of the novel models and algorithms as well as the correctness of the performance prediction is proven with performance measurements.

Keywords: Web-based system, Queueing model, Mean-Value Analysis algoritm, Queue limit, Performance prediction

1 Introduction

The performance of web-based software systems is one of the most important and complicated consideration, because they face a large number of users, and they must provide high-availability services with low response time, while they guarantee a certain throughput level. Performance metrics are influenced by many factors. Several papers have investigated various configurable parameters, and the way in which they affect the performance of web-based software systems [1][2].

The performance metrics can be predicted at early stages of the development process with the help of a properly designed performance model and an appropriate evaluation algorithm. In the past few years several methods have been proposed to address this issue. Several of them is based on queueing networks or extended versions of queueing networks [3]. Another group is using Petri-nets or generalized stochastic Petri-nets [4]. As the third kind of the approaches, the stochastic extension of process algebras, such as TIPP (Time Processes and Performability Evaluation) [5], EMPA (Extended Markovian Process Algebra) [6], and PEPA (Performance Evaluation Process Algebra) [7] can be mentioned.

Web-based software systems access some resources while executing the requests of the clients. Typically several requests arrive at the same time, thus, competitive situation is established for the resources. For modeling such situations queueing model-based approaches are widely used. Queueing networks have also been proposed to model web-based software systems [8] [9] [10].

The paper is organized as follows. Section 2 introduces the background and related work. Section 3.1 provides novel models and algorithms to model the queue limit. Section 3.2 shows that the proposed models and algorithms can be applied to performance prediction of web-based software systems in ASP.NET environment. Section 3.3 verifies the correctness of the performance prediction.

2 Backgroung and Related Work

This section is devoted to review the background and research efforts related this work, namely, the concept of thread pools and queued requests as well as the queueing network models for multi-tier software systems, which have led to the extensive research of performance models of web-based software systems.

2.1 Thread Pools and Queued Requests

In case of using a thread pool, when a request arrives, the application adds it to an incoming queue [11]. A group of threads retrieves requests from this queue and processes them. As each thread is freed, another request is executed from the queue.

The architecture of ASP.NET environment can be seen in Fig. 1. If a client is requesting a service from the server, the request goes through several subsystems before it is served.

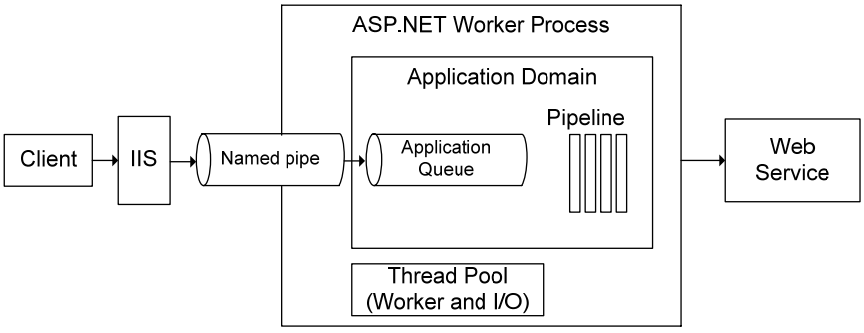

Figure 1
The architecture of ASP.NET environment

From the IIS, the accepted HTTP connections are placed into a named pipe. This is a global queue between IIS and ASP.NET, where the requests are posted from native code to the managed thread pool. The global queue is managed by the process that runs ASP.NET, its limit is set by the *requestQueueLimit* property. When the Requests Current counter – which includes the requests that are queued, being executed, or waiting to be written to the client – reaches this limit, the requests are rejected [12]. Thus, the Requests Current must be greater than the sum of *maxWorkerThreads* plus *maxIOThreads*. From the named pipe, the requests are placed into an application queue, also known as a virtual directory queue. Each virtual directory has a queue that is used to maintain the availability of worker and I/O threads. The number of requests in these queues increases if the number of available worker and I/O threads falls below the limit specified by *minFreeThreads* property. The application queue limit is configured by the *appRequestQueueLimit* property. When the limit is exceeded, the requests are rejected.

The .NET Framework offers a highly optimized thread pool. This pool is associated with the physical process where the application is running, there is only one pool per process. The *maxWorkerThreads* attribute means the maximum number of worker threads, the *maxIOThreads* parameter is the maximum number of I/O threads in the .NET thread pool. These attributes are automatically multiplied by the number of available CPUs. The *minFreeThreads* attribute limits the number of concurrent requests, because all incoming requests will be queued if the number of available threads in the thread pool falls below the value for this setting. The *minLocalRequestFreeThreads* parameter is similar to *minFreeThreads*, but it is related to the requests from the local host.

In our previous work [2][13], six influencing parameters of the performance have been found: *maxWorkerThreads*, *maxIOThreads*, *minFreeThreads*, *minLocal-RequestFreeThreads*, *requestQueueLimit*, *appRequestQueueLimit*, which is proven by a statistical method, namely, the chi square test of independence. The identified performance factors must be modeled to improve performance models. The behavior of the thread pool have been modeled in our previous work [14]. This paper introduces the modeling of the queue size limit.

2.2 Queueing Models for Multi-tier Software Systems

This section discusses the base queueing network model and the Mean-Value Analysis evaluation algoritm for multi-tier software systems used in this paper.

A product form network should satisfy the conditions of job flow balance, one-step behavior, and device homogeneity [15]. The job flow balance assumption holds only in some observation periods, namely, it is a good approximation for long observation intervals since the ratio of unfinished jobs to completed jobs is small.

Definition 2.1. The ***base queueing model*** is defined for multi-tier information systems [9] [16], which are modeled as a network of M queues $Q_1,...,Q_M$ illustrated in Fig. 2. Each queue represents an application tier. S_m denotes the service time of a request at Q_m ($1 \leq m \leq M$). 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. 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. 2. 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.

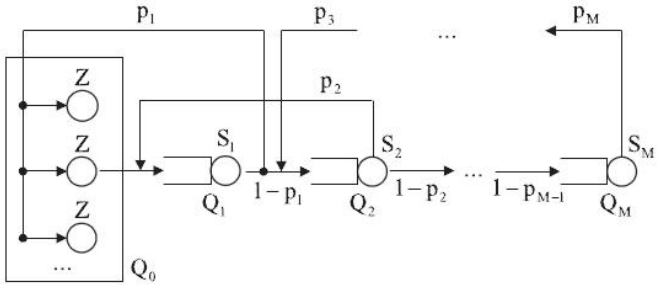

Figure 2
Modeling a multi-tier information system using a queueing network

Remark 2.2. There are transitions from each queue only to its successor and its predecessor, because in multi-tier architecture only neighboring tiers communicate each other directly. The transition of Q_1 to the preceding queue denotes the completion of a request, since requests of clients are started from and completed in the presentation tier (in three-tier architecture the presentation tier is represented by Q_1).

Remark 2.3. In this paper, closed queueing networks are applied, since closed queueing models predict the performance for a fixed number of sessions serviced by the application.

The Mean-Value Analysis (MVA) algorithm for closed queueing networks [3] [17] is applicable only if the network is in product form. In addition, the queues are assumed to be either fixed-capacity service centers or infinite servers, and in both cases, exponentially distributed service times are assumed.

Remark 2.4. Since the base queueing model satisfies the conditions above, the MVA algorithm can evaluate the base queueing model.

Definition 2.5. The ***Mean-Value Analysis*** for the base queueing model is defined by Algorithm 1, and the associated notations are in Table 1.

Algorithm 1 Pseudo code of the MVA algorithm

1: **for all** $m = 1$ to M **do**
2: $L_m = 0$
3: **for all** $n = 1$ to N **do**
4: **for all** $m = 1$ to M **do**
5: $R_m = V_m \cdot S_m \cdot (1 + L_m)$
6: $R = \sum_{m=1}^{M} R_m$
7: $\tau = n/(Z + R)$
8: **for all** $m = 1$ to M **do**
9: $L_m = \tau \cdot R_m$

Table 1
Notations of Mean-Value Analysis evaluation algorithm

Notation	Meaning	I/O	Comment
L_m	queue length of Q_m $(1 \leq m \leq M)$	O	
M	number of tiers	I	
N	number of customers	I	number of sessions in the base queueing model
R	response time	O	
R_m	response time for Q_m $(1 \leq m \leq M)$	O	
S_m	service time for Q_m $(1 \leq m \leq M)$	I	
τ	throughput	O	
V_m	visit number for Q_m $(1 \leq m \leq M)$	I	derived from transition probabilities p_m of the base queueing model
Z	user think time	I	

The model can be evaluated for a given number of concurrent sessions, note that a session in the model corresponds to a customer in the evaluation algorithm. The algorithm uses visit numbers instead of transition probabilities. The visit number is the number of times when a tier is invoked during the lifetime of a request, and visit numbers can be easily derived from transition probabilities.

Remark 2.6. The visit number concept of the MVA algorithm is more general than the transition probabilities from a queue to its successor and its predecessor in case of the base queueing model. Thus, the model could be more general, but in practice only neighboring tiers communicate each other directly.

3 Modeling the Queue Limit

The queue size must be limited to prevent requests from consuming all the memory for the server, for an application queue. By taking the queue limit (Section 2) into consideration, the base queueing model (Definition 2.1) and the MVA evaluation algorithm (Definition 2.5) can be effectively enhanced. The enhancement of the baseline model handles such concurrency limits, when each tier has an individual concurrency limit [9]. This approach manages the case in which all tiers have a common concurrency limit.

This section discusses modeling the queue limit. Firstly, this chapter proposes novel models and algorithms to model the global and the application queue limit performance factors. Then, it validates the proposed models and algorithms, and verifies the correctness of performance prediction with the novel models and algorithms.

3.1 Novel Models and Algorithms

Definition 3.1. The ***extended queueing model with global queue limit*** (QM-GQL) is defined by Fig. 3, where the Q_{drop} is an infinite server queueing system, the Z_{drop} is the time spent at Q_{drop}, the *GQL* is the global queue limit, which corresponds to the *requestQueueLimit* parameter in ASP.NET environment. If the *GQL* is less than the queued requests sum, the next requests proceed to Q_{drop}. Requests from Q_{drop} proceed back to Q_0, namely, these requests are reissued.

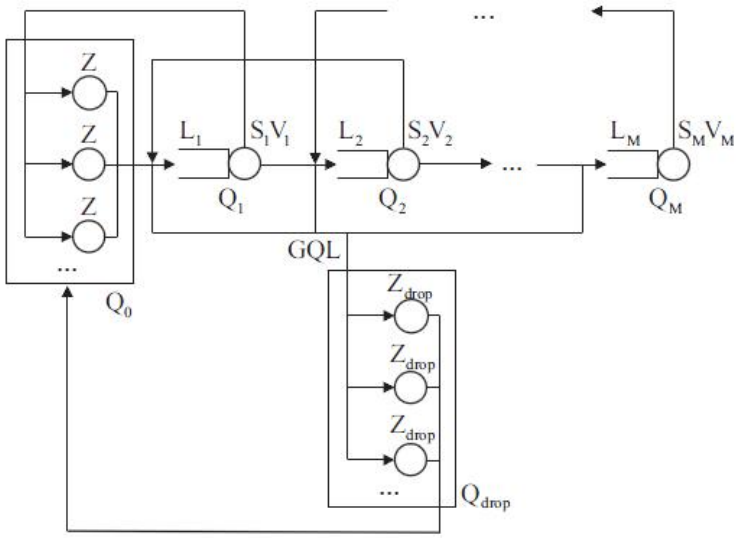

Figure 3
Extended queueing model with global queue limit

Definition 3.2. The ***extended MVA with global queue limit*** (MVA-GQL) is defined by Algorithm 2, where the Z_{drop} is the time spent at Q_{drop}, the *GQL* is the global queue limit, which corresponds to the *requestQueueLimit* parameter in ASP.NET environment.

Algorithm 2 Pseudo code of the MVA-GQL

1: **for all** $m = 1$ to M **do**
2: $L_m = 0$
3: $nql = 1$
4: **for all** $n = 1$ to N **do**
5: **for all** $m = 1$ to M **do**
6: $R_m = V_m \cdot S_m \cdot (1 + L_m)$
7: $R = \sum_{m=1}^{M} R_m$
8: $\tau = nql/(Z + Z_{drop} + R)$
9: **for all** $m = 1$ to M **do**
10: $L_m = \tau \cdot R_m$
11: **if** $\sum_{m=1}^{M} L_m > GQL$ **then**
12: **for all** $m = 1$ to M **do**
13: $L_m = oldL_m$
14: **else**
15: $nql = nql + 1$
16: **for all** $m = 1$ to M **do**
17: $oldL_m = L_m$

Considering the concept of the global queue, if the current requests – queued plus executing requests (by ASP.NET) – exceed the global queue limit (GQL), the next incoming requests will be rejected. In these cases, the queue length does not have to be updated (see Steps 10 and 13 of Algorithm 2). The queued requests sum of the model and algorithm contains not only the queued requests by ASP.NET but the working threads of ASP.NET, as well.

Proposition 3.3. The novel MVA-GQL (Definition 3.2) can be applied as an approximation method to the proposed QM-GQL (Definition 3.1).

Proof. The QM-GQL model does not satisfy the condition of job flow balance (see in Section 2). Thus, the MVA-GQL evaluation algorithm can be applied as an approximation method to the QM-GQL model. In Step 8 of Algorithm 2, when it computes the throughput, the Z_{drop} of the model is taken into consideration similarly to *Z*. In Steps 11 and 13 of the algorithm, if the *GQL* is less than the queued requests sum, the next requests proceed to Q_{drop} in the model, the queue length does not have to be updated in the algorithm.

Definition 3.4. The ***extended queueing model with application queue limit*** (QM-AQL) is defined by Fig. 3 (*AQL+WT* instead of *GQL*), where the Q_{drop} is an infinite server queueing system, the Z_{drop} is the time spent at Q_{drop}, the *AQL* is the application queue limit, which corresponds to the *appRequestQueueLimit* parameter in ASP.NET environment, in addition, the *WT* is the maximum number

of working threads, which equals *maxWorkerThreads+maxIOThreads-minFreeThreads* in ASP.NET environment. If the *AQL+WT* is less than the queued requests sum, the next requests proceed to Q_{drop}. Requests from Q_{drop} proceed back to Q_0, namely, these requests are reissued.

Definition 3.5. The extended MVA with application queue limit (MVA-AQL) is defined by Algorithm 2 (*AQL+WT* instead of *GQL)*, where the Z_{drop} is the time spent at Q_{drop}, the *AQL* is the application queue limit, which corresponds to the *appRequestQueueLimit* parameter in ASP.NET environment, in addition, the *WT* is the maximum number of working threads, which equals *maxWorkerThreads+maxIOThreads-minFreeThreads* in ASP.NET environment.

Considering the concept of the application queue, if the number of queued requests (by ASP.NET) exceeds the application queue limit (*AQL*), the next incoming requests will be rejected. In these cases, the queue length has not to be updated (see Steps 10 and 13 of Algorithm 2). Since the queued requests sum of the model and the algorithm contains not only the queued requests by ASP.NET but the working threads of ASP.NET, as well. Thus, *WT* has to be subtracted from the number of queued requests of the model and algorithm to obtain the queued requests by ASP.NET.

Proposition 3.6. The novel MVA-AQL (Definition 3.5) can be applied as an approximation method to the proposed QM-AQL (Definition 3.4).

Proof. See the proof of Proposition 3.3.

3.2 Performance Prediction and Validation

In this section, it is shown that the proposed models and algorithms can be used for performance prediction of web-based software systems. Firstly, the proposed models and algorithms have been implemented with the help of MATLAB. Secondly, the input values have been measured or estimated. For model parameter estimation and model evaluation, only one measurement or estimation in case of one customer is required in ASP.NET environment. Then, the proposed models have been evaluated by the novel algorithms to predict the performance metrics.

Proposition 3.7. The novel QM-GQL and MVA-GQL as well as QM-AQL and MVA-AQL (Definitions 3.1 and 3.2, 3.4 and 3.5) can be applied to performance prediction of web-based software systems in ASP.NET environment, these proposed models and algorithms predict the performance metrics more accurate than the original base queueing model and MVA (Definitions 2.1 and 2.5).

Proof. The novel models and algorithms have been validated and the correctness of the performance prediction with the proposed models and algorithms has been verified with performance measurements in ASP.NET environment. These validation and verification have been performed by comparing the observed values

provided by performance measurements in ASP.NET environment, the predicted values with the original model and algorithm, and the predicted values with the proposed models and algorithms. The results depicted in Fig. 4 have shown that the outputs of the proposed models and algorithms approximate the measured values much better than the outputs of the original model and algorithm considering the shape of the curve and the values, as well. Thus, the proposed models and algorithms predict the response time and throughput performance metrics much more accurate than the original model and algorithm in ASP.NET environment.

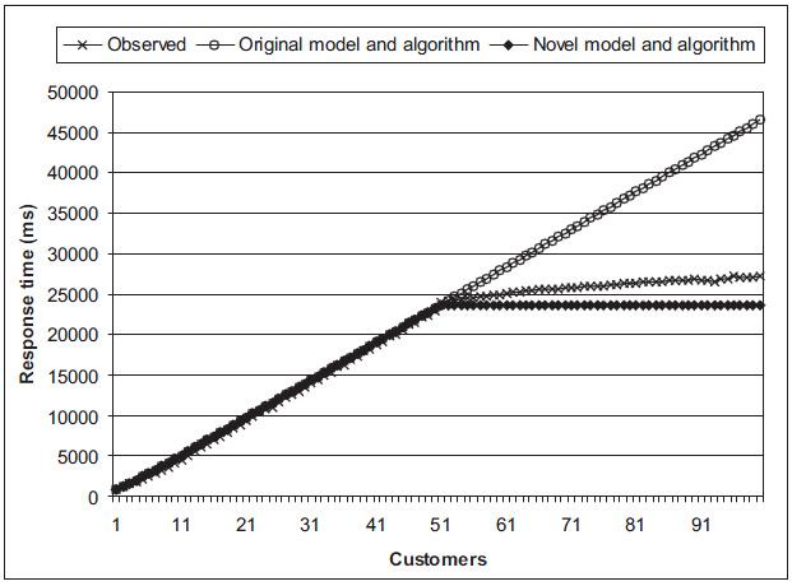

Figure 4
Observed response time, predicted with original MVA, and predicted with novel MVA-GQL

3.3 Error Analysis

Finally, the error has been analyzed to verify the correctness of the performance prediction with the proposed models and algorithms. Two methods have been applied: the average absolute error function and the error histogram.

Definition 3.8. The ***average absolute error function*** is defined by Equation 1, where the index *alg* corresponds to the values provided by the algorithm, the index *obs* is related to the observed values, *R* is the response time, and *N* is the number of measurements.

$$error_{alg-obs} = (\sum_{i=1}^{N} \left| R_{alg_i} - R_{obs_i} \right|) / N \quad (1)$$

Proposition 3.9. The average absolute error of the response time performance metric provided by the novel QM-GQL and MVA-GQL as well as QM-AQL and MVA-AQL (Definitions 3.1 and 3.2, 3.4 and 3.5) is less than the average absolute error of the response time computed by the original base queueing model and MVA (Definitions 2.1 and 2.5).

Proof. The results of the average absolute error function are presented in Table 2. The results have been shown that the error of the novel models and algorithms is substantially less than the error of original model and algorithm in ASP.NET environment.

Table 2
The results of the average absolute error function

	Global queue	Application queue
Original	4774.7	2139.3
Novel	1284.3	888.4417

The error histograms of the original and the proposed models and algorithms are depicted in Fig. 5.

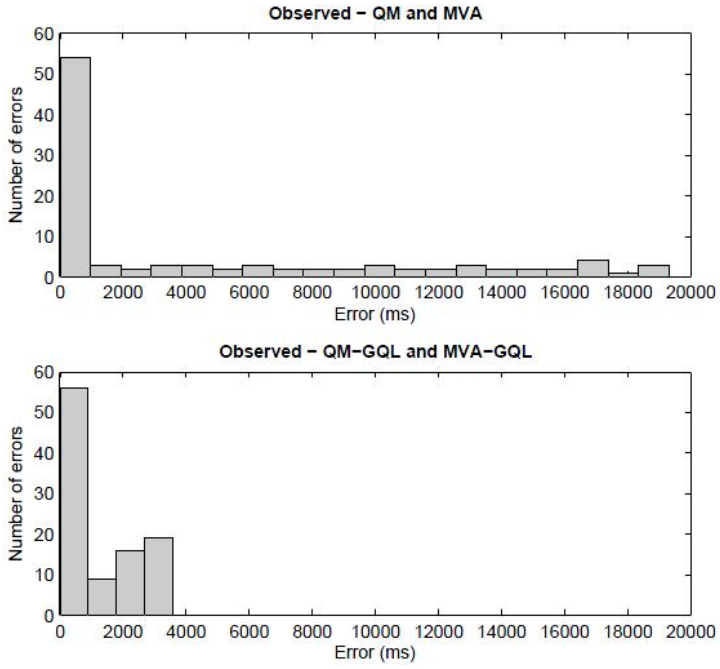

Figure 5
Error histogram in case of global queue limit

The values of the error (x-axis) with the proposed models and algorithms are substantially less than with the original model and algorithm. The number of errors in the first bin (y-axis) of the original and the novel models and algorithms are approximately the same, because the response time and the throughput computed by the proposed models and algorithms are the same as the performance metrics computed by the original model and algorithm, until all the requests have successfully been served. With the novel models and algorithms, the number of errors in the first bins (y-axis) is greater, since all of the errors are only in these first bins. In case of the original model and algorithm, the errors increase at much greater rate.

Conclusions

The queue limit parameters are dominant factors considering the response time and throughput performance metrics. These identified performance factors must be modeled to improve performance models. In this paper, novel models and algorithms have been provided to model the global and application queue limit. It has been proven that the novel algorithms can be applied as an approximation method to the proposed models. It has been shown that the proposed models and algorithms can be applied to performance prediction of web-based software systems in ASP.NET environment, the novel models and algorithms predict the performance metrics more accurate than the original model and algorithm. The proposed models and algorithms have been validated and the correctness of performance prediction with novel models and algorithms has been verified with performance measurements in ASP.NET environment. The results have shown that the proposed models and algorithms predict the performance metrics much more accurate than the original model and algorithm.

With the novel models and algorithms of this paper, the global and application queue limit performance factors can be modeled, in addition, performance prediction can be performed much more accurate. The extended MVA with thread pool [14] and the extended MVA with queue limit are independent and additive, they can be applied together or separately, as well.

Acknowledgement

This research was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

References

[1] M. Sopitkamol, D. A. Menascé: A Method for Evaluating the Impact of Software Configuration Parameters on E-commerce Sites, ACM 5th International Workshop on Software and Performance, 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, Lecture Notes in Computer Science, Vol. 3746, 2005, pp. 223-233

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

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

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

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

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

[8] D. A. Menascé, V. Almeida: Capacity Planning for Web Services: Metrics, Models, and Methods, Prentice Hall PTR, 2001

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

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

[11] D. Carmona: Programming the Thread Pool in the .NET Framework, .NET Development (General) Technical Articles, 2002

[12] T. Marquardt: ASP.NET Performance Monitoring, and When to Alert Administrators, ASP.NET Technical Articles, 2003

[13] Á. Bogárdi-Mészöly, T. Levendovszky, H. Charaf: Performance Factors in ASP.NET Web Applications with Limited Queue Models, 10th IEEE Int. Conference on Intelligent Engineering Systems, 2006, pp. 253-257

[14] Á. Bogárdi-Mészöly, T. Hashimoto, T. Levendovszky, H. Charaf: Thread Pool-Based Improvement of the Mean-Value Analysis Algorithm, Lecture Notes in Electrical Engineering, Vol. 28(2), 2009, pp. 347-359

[15] P. Denning, J. Buzen: The Operational Analysis of Queueing Network Models. Computing Surveys, Vol. 10, 1978, 225-261

[16] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, A. Tantawi: An Analytical Model for Multi-tier Internet Services and its Applications, ACM SIGMETRICS Performance Evalutaion Review, Vol. 33(1), 2005, pp. 291-302

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