International Conference on Applied Internet and Information Technologies, 2018 DOI:10.20544/AIIT2018.P09

Performance Evaluation of Clustered Web Servers Using the Round Robin Scheme

I. Hristoski*, T. Dimovski**, Z. Kotevski**, R. Markoska** and N. Rendevski**
* "St. Kliment Ohridski" University - Bitola/Faculty of Economics, Prilep, Republic of Macedonia
** "St. Kliment Ohridski" University - Bitola/Faculty of ICTs, Bitola, Republic of Macedonia
{ilija.hristoski, tome.dimovski, zoran.kotevski, ramona.markoska, nikola.rendevski}@uklo.edu.mk

Abstract - Contemporary e-Commerce systems are constantly facing huge and bursty workloads. Continuous performance evaluation of Web server clusters is a cornerstone of capacity planning methodology, whilst their performance modeling and simulation strives to foresee the system's behavior under various scenarios. In this paper, we present a simulation model of a generic e-Commerce Web server cluster, consisting of a variable number of Web servers, using the class of Generalized Stochastic Petri Nets (GSPNs). The model also implements one of the basic static load balancing algorithms, the Round Robin algorithm, for equal distribution of incoming HTTP requests across the cluster's Web servers. We convey a steady-state and a transient analysis of the GSPN model by numerical simulations using dedicated software, TimeNET, and present the obtained results. This way, we accomplish a threefold aim, (1) to establish a modeling framework suitable for performance analysis of arbitrary e-Commerce Web server cluster; (2) to get valuable insights into the dynamics and behavior of such systems; (3) to demonstrate the modeling and evaluation power of TimeNET regarding the performance analysis of arbitrary Discrete-Event Dynamic Systems (DEDSs).

I. INTRODUCTION

As e-Commerce paradigm became mainstream worldwide during the mid-1990s, e-Commerce Websites began to cope with a heavily increased traffic. As a result, single Web servers started reaching their capacity limits to effectively handle the ever-increasing workload. Soon e-Commerce Websites needed inclusion of additional Web servers to successfully handle the incoming Internet traffic generated by e-Customers, which easily outreached tens of millions of HTTP requests per day. This phenomenon has led to the emergence of Web clusters, a locally distributed Web system that is "… an architecture consisting of multiple Web servers and mechanisms to route incoming requests among several server nodes in a user-transparent way" [1]. As a result, a multi-tiered Web architecture for e-Commerce has emerged (Fig. 1). Cluster computing appears in several varieties, each offering different advantages, like high availability, load balancing, high-performance computing, and grid computing [2]. In such architectures, it is of an utmost importance to split the workload as much as equally across the Web servers (i.e. to balance the workload), in order to preserve the general availability and high responsiveness of the whole system.

Load balancing refers to "the process of distributing the load among various nodes of a distributed system to improve both job response time and resource utilization while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or lightly loaded" [3]. The load distributing policy aims to maximize the throughput of a distributed system as a whole.

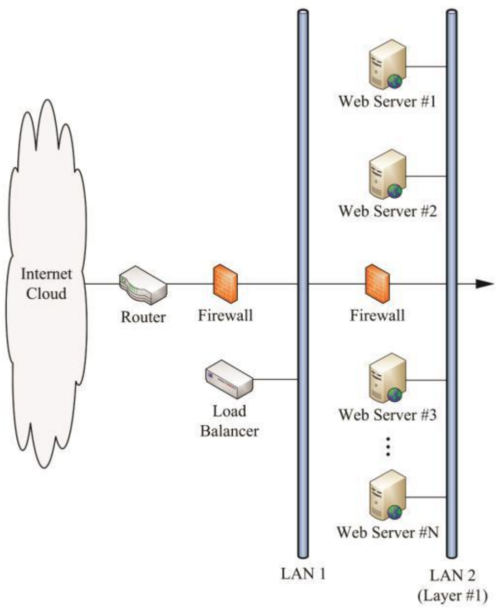

Figure 1. Schematic representation of the first tier of a generic multi-tiered e-Commerce Website architecture, portraying the load balancer and a cluster comprised of a number of Web servers

Efficient load balancing is based on the utilization of appropriate algorithms, known as load-scheduling or load-balancing methods/schemes. They are used by load balancers to determine the next Web server to send an HTTP request to. These can be roughly categorized into three major categories: static, dynamic, and hybrid [4].

Whilst in dynamic load balancing the workload is distributed across the Web servers in a non-deterministic way, based on the feedback information from Web servers during runtime, in static load balancing the workload is distributed across Web servers in accordance with a pre-determined scheme that does not depend on any feedback information from the Web servers. Static load balancing is independent regarding the current system state; it does not depend on the time instances when decisions are made. It should be notified that, in general, dynamic and hybrid algorithms deliver better results than static algorithms [4].

Because of its intrinsic simplicity, and the ability to be represented by stochastic Petri nets, in this paper we use the Round Robin (RR) static load balancing scheme, which is being widely used both in practice and theory. The RR scheme is easy to be implemented; it is a starvation-free algorithm since it presents a circular queueing process. Theoretically, it assumes that all Web servers are identical machines, exhibiting identical performances. In addition, it does not require any inter-process communication. However, the RR scheme cannot provide good results in a general case, and/or when jobs are unequal (i.e. it provides the best performance only for special purpose applications). Besides, it does not support the prioritization of more important tasks. In this particular case, we do not use time-slicing RR scheme on queued jobs (HTTP requests) in the load balancer, but rather we assume that complete jobs (e.g. equally sized HTTP requests) are being distributed towards Web servers in the cluster.

The paper is organized as follows. In Section 2 some of the most recent research closely related to the subject has been elaborated. Section 3 briefly introduces the class of Generalized Stochastic Petri Nets (GSPNs) that has been used as a modeling formalism in building the proposed performance evaluation model. The modeling framework has been described in Section 4. Section 5 presents the simulation results in two subsections. Section 6 concludes.

II. Related Research

During the recent three decades, an abundance of research has been done vis-à-vis the performance evaluation of Web server clusters, the efficiency of load balancing algorithms, or both. Here we present some of the most recent and relevant research.

Reference [5] focuses on the simulation of four static load balancing algorithms that have been carried out in order to compare their performances. Performance evaluation and a proposed dynamic architecture of Web server farms have been studied in [6]. Reference [7] deals with the evaluation of two performance measures (i.e. the mean response time and the utilization of Web servers) using a software simulator that implements the algorithms of several load balancing schemes, including the RR scheme. Performance evaluation of distributed Web server architectures under e-Commerce workloads has been treated in [8]. The RR scheme has been subject to analysis in [9] and [10]. A comparison of load balancing algorithms for clustered Web servers has been carried out in [11].

III. Generalized Stochastic Petri Nets (GSPNs)

Initially been proposed by Marsan, Balbo, and Conte in 1984, the class of Generalized Stochastic Petri Nets (GSPNs) is now recognized as a widely-known tool for performance analysis of distributed systems, which utilizes the graphical notation introduced by ordinary Petri Nets (PNs) [12-13]. In GSPNs some transitions are timed, whilst others are immediate. Random, exponentially distributed firing delays are associated with timed transitions, whereas the firing of immediate transitions takes place in zero time, with priority over timed transitions. In addition, the selection among several possibly conflicting enabled immediate transitions is made by utilizing their corresponding firing probabilities. In general, immediate transitions are used for modeling instantaneous actions or logical actions (typically choices), whilst timed transitions with an exponentially distributed delays are used for modeling the duration of activities (events) within the GSPN model.

The analysis of a GSPN model can be two-fold: (1) qualitative: performed by studying the structural characteristics of the underlying Petri Net; (2) quantitative: performed by computing the steady-state (stationary) and/or the transient (time-dependent) probability distributions of the associated stochastic model (process), equivalent to a GSPN model. GSPNs are isomorphic to semi-Markov processes, i.e. their quantitative analysis can be performed on a reduced Embedded CTMC (Embedded Markov Chain, EMC), defined solely on a set of tangible states, or by reducing the GSPN to an equivalent Stochastic Petri Net (SPN) [13]. The stationary distribution of the underlying stochastic process is usually a basis for obtaining a plethora of performance metrics, like calculating the probabilities of specific state conditions, resource utilization, expected throughputs, expected number of clients (active resources), expected waiting times, etc. On the other hand, transient analysis is a basis for investigating the system behavior over time, i.e. it describes the evolution of the observed system up to a given time and thus it can be used for obtaining specific performance metrics such as probabilities of reaching particular states and probabilities of satisfying assigned deadlines [13].

IV. The Modeling Framework

For demonstration purposes, we come up with a series of GSPN models of clusters comprised of a different number of Web servers that implement the RR load balancing algorithm (Fig. 2). The modeling has been carried out using TimeNET®, a dedicated tool for modeling and simulation of several classes of Stochastic Petri Nets, including GSPNs [14-16].

All developed GSPN models are based on the existence of a single load balancer (a place named *P_LB*) and a cluster of an arbitrary number of Web servers ($N = 2 \dots 5$), that belong to the first tier of a typical e-Commerce Website architecture. The workload posed to the load balancer has an intensity of λ [HTTP requests/s], represented by a flow of tokens coming from the place named *P_HTTPs* at random time instances that follow the Poisson distribution, after the exponential transition *T_HTTPs* fires at a rate of $\lambda = 1/arrival\_time$. The variable *arrival_time* [s] denotes the mean inter-arrival time of HTTP requests. All incoming HTTP requests are presented by individual tokens, one per a request, residing in the place *P_LB* (i.e. the load balancer).

The GSPN substructure, comprised of both the places *P_choose_WS*(*i*) and immediate transitions

T_send_to_WS(*i*) (*i* = 2 … 5), along with the place *P_LB*, implements the load balancer that distributes the tokens from the place *P_LB* (i.e. the HTTP requests) across the clustered Web servers according to the RR scheme.

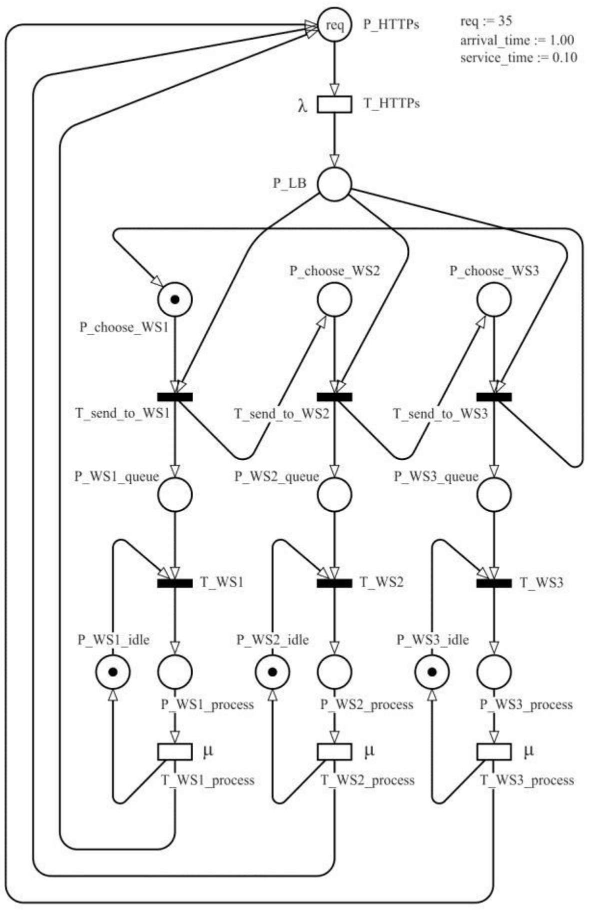

Figure 2. GSPN model of a Web cluster comprised of N = 3 Web servers and a load balancer that distributes the incoming HTTP requests according to the Round Robin algorithm

As long as there is a token in the place *P_LB* and a token in the place *P_choose_WS*(*i*), the immediate transition *T_send_to_WS*(*i*) becomes enabled and fires, such that a single token is being removed from the places *P_LB* and *P_choose_WS*(*i*), and a single token is being put to the place *P_WS*(*i*)*_queue* (*i* = 2 … 5). This means that an HTTP request has been sent from the load balancer to the corresponding Web server. At the same time, the firing of the transition *T_send_to_WS*(*i*) also puts a token in the place *P_choose_WS*(*i*+1), to point out the next Web server an HTTP request should be sent to. The firing of the last transition *T_send_to_WS*(*N*) puts a token back to the place *P_choose_WS1*, such that a circle becomes closed.

During the execution, places *P_WS*(*i*)*_queue* (*i* = 2 … 5) contain an arbitrary number of tokens, resembling HTTP requests waiting in a buffer queue to be processed by particular Web servers, *WS*(*i*). Whenever the place *P_WS*(*i*)*_idle* contains a token (i.e. the corresponding Web server is idle), the immediate transition *T_WS*(*i*) becomes enabled and fires, taking away a single token from the place *P_WS*(*i*)*_queue* and putting it into the place *P_WS*(*i*)*_process*. In such a way, the exponential transition *T_WS*(*i*)*_process* becomes enabled; it fires after a mean time delay *service_time* [s], such that the service rate μ = 1/*service_time* [HTTP requests/s].

For simplicity reasons, it is assumed that each of the Web servers processes HTTP requests at a constant service rate of μ [HTTP requests/s], i.e. all of them represent identical machines exhibiting an identical individual performance, which is often referred to as a horizontal scaling. In practice, however, it is more likely that Web servers are going to be different machines that would process the HTTP requests at rather different service rates.

Concurrent firings of the exponential transitions *T_WS*(*i*)*_process* put a single token back in places *P_WS*(*i*)*_idle* (to denote that the Web server is ready to process the next HTTP request in the queue), and also put a token back to the place *P_HTTPs*, in order to preserve the initial number of tokens in the model and make its state-space a finite one. A finite state-space is necessary for the simulation model to be computationally tractable.

The whole modeled system is equivalent to an M/M/*N*/*req* queuing system, i.e. a multi-server, finite-capacity system with maximum *N* servers and *req* customers, where the HTTP requests arrive according to a Poisson process with a rate of λ (i.e. the inter-arrival times are independent, exponentially distributed random variables with parameter λ), whilst the service times are also assumed to be independent and exponentially distributed with parameter μ [17].

V. Simulation Results

Given that the service rate μ has a fixed value of 10 [HTTP requests/s], we have altered the arrival rate λ, so it took its values from the interval [1 … 35] [HTTP requests/s], with a step of 2. Thus, the resulting ratio ρ = λ/μ ranges from 0.1 to 3.5, with a step of 0.2. It should be notified that such values have been deliberately chosen to investigate the system behavior when ρ ≈ 1 and ρ > 1.

All simulations have been carried out with a finite initial number of tokens in the place *P_HTTPs*, *req* = 35 [HTTP requests]. Further increasing of the value of variable *req*, and/or the value *N* representing the number of Web servers in the cluster has led to computationally intractable simulation models, because of the state-state explosion (e.g. for *N* = 4 and *req* = 35, the GSPN model has 326,340 tangible markings, whilst for *N* = 5 and *req* = 35, the GSPN model has 3,247,860 tangible markings!).

The specification of the corresponding reward measures (performance metrics) for the Web server WS1 is presented in Table 1. The corresponding reward measures have been defined for all other Web servers included in the GSPN models, as well. However, since the distribution of tokens (i.e. HTTP requests) across the places *P_WS*(*i*)*_queue*, *i* = 2 … 5 (i.e. Web servers) has been done using the RR scheme, and since all tokens are mutually equal, the obtained results (i.e. the values of the reward measures) for all Web servers are identical, i.e. the RR scheme assures that all modeled Web servers in the cluster are equally loaded.

TABLE I. DEFINED REWARD MEASURES

Measure	TimeNET v4.4 Definition
utilization_WS1	(#P_WS1_proc > 0)
queue_length_WS1	(#P_WS1_queue)
waiting_time_WS1	((#P_WS1_queue) - (#P_WS1_queue > 0)) * arrival_time
Prob_0_requests	(#P_WS1_queue == 0)
Prob_*x*_requests ($x = 1 \dots 5$)	(#P_WS1_queue == *x*)
Prob_*x*_and_more_ requests ($x = 0 \dots 5$)	(#P_WS1_queue >= *x*)
Prob_*x*_and_less_ requests ($x = 1 \dots 5$)	(#P_WS1_queue <= *x*)

A. Steady-state analysis

The steady-state (stationary) analysis has been done using the Stationary Analysis module, which computes the steady-state solution of the model with continuous time, by solving the corresponding Embedded Markov Chain (EMC) [15].

The functional dependency of the Web server utilization vis-à-vis the arrival rate λ, for a various number of Web servers in the cluster, is depicted in Fig. 3.

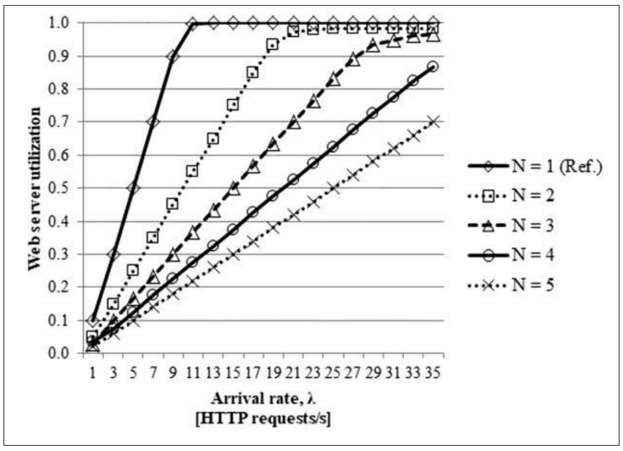

Figure 3. Web server utilization as a function of the HTTP requests' arrival rate, λ, for different number of Web servers, given that the service rate μ equals 10 [s^{-1}]

Simulations show that, for all values of the arrival rate λ (i.e. $0 < \lambda \leq 9$) that precede Web server saturation (i.e. high levels of utilization, greater than 0.90), adding a second identical Web server reduces the utilization by half (50%), adding a third Web server reduces their utilization by 67%, adding a fourth Web server reduces their utilization by 75% and adding a fifth Web server reduces their utilization by 80%, relative to the utilization levels gained in the reference case (i.e. when a single Web server is used).

Fig. 4 depicts the average queue length at Web servers vis-à-vis the arrival rate λ, for a various number of Web servers in the cluster. Simulation results show that for all values of the arrival rate λ (i.e. $\lambda \geq 29$) that follow Web server saturation (i.e. high levels of average queue length, generally greater than 33.5), adding a second identical Web server reduces the average queue length at Web servers by more than a half (approximately by 53%).

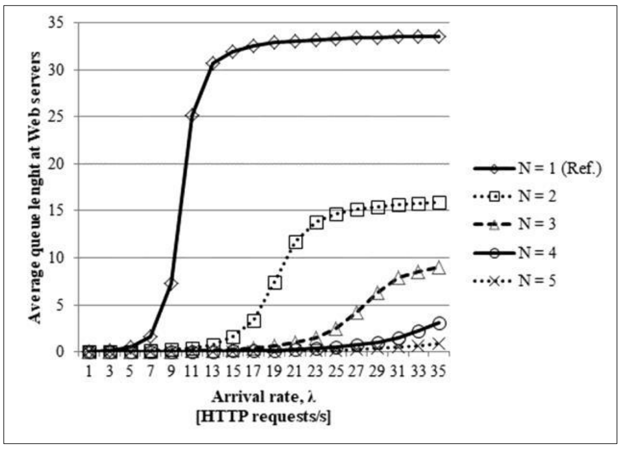

Figure 4. Average queue length at Web servers as a function of the HTTP requests' arrival rate, λ, for different number of Web servers, given that the service rate μ equals 10 [s^{-1}]

Further significant reductions of average queue length at Web servers are evident by adding a third (approximately by 75%), fourth, and a fifth Web server to the cluster, relative to the average queue length gained in the reference case (i.e. when a single Web server is used).

The average waiting time of HTTP requests in Web server queues vis-à-vis the arrival rate λ, for a various number of Web servers in the cluster, is depicted in Fig. 5.

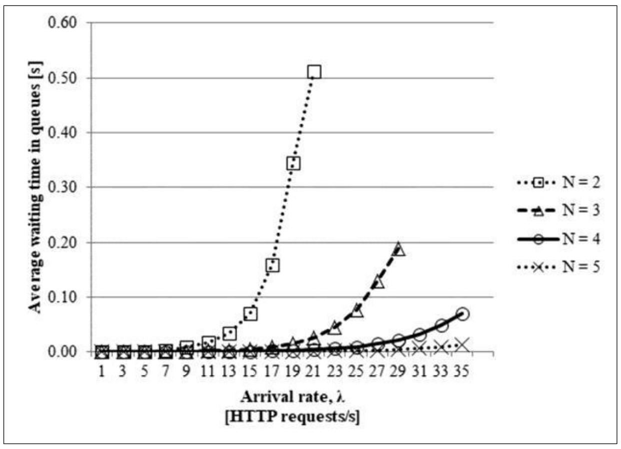

Figure 5. Average waiting time at Web server queues as a function of the HTTP requests' arrival rate, λ, for different number of Web servers, given that the service rate μ equals 10 [s^{-1}]

According to the simulations, adding a third, fourth and fifth Web server in the cluster reduces the average waiting time of HTTP requests in Web server queues at least by 91%, 83%, and 76%, respectively, and relative to each other.

A 3D view of the probability that Web servers are idle (i.e. there are no HTTP requests waiting in Web server queues) is shown in Fig. 6. It is obvious that as the ratio $\rho = \lambda/\mu$ rises, the probability $P(r = 0)$ of having zero HTTP requests in Web servers' queues drops down. However, such decrease slows down as the number of Web servers in the cluster, N, increases. In other words, the probability P(r = 0) that the Web servers would be idle rises as their number in the cluster increases.

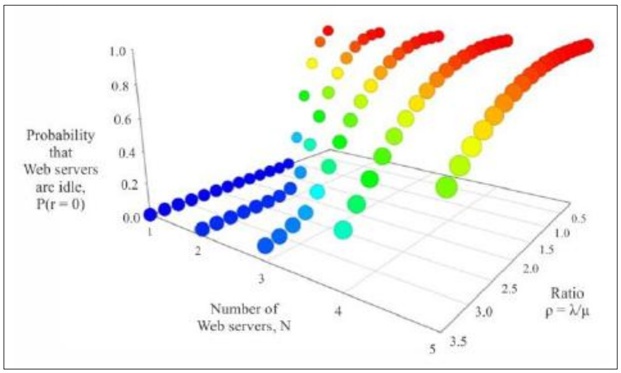

Figure 6. Probability P(r = 0) that Web servers are idle, as a function of the ratio λ/μ, and different number of Web servers, given that the service rate μ equals 10 [s^{-1}]

The probability that exactly R HTTP requests are present in Web server queues, P(r = R), is portrayed in Fig. 7.

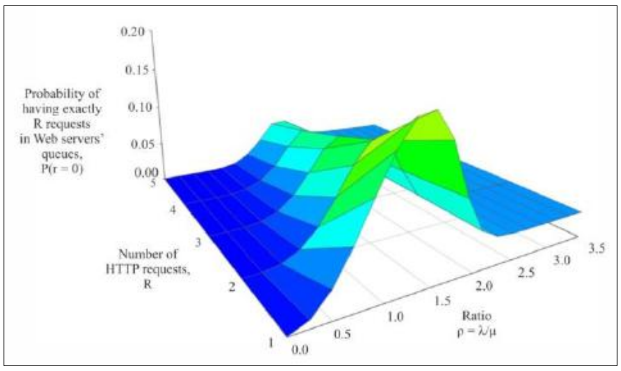

Figure 7. Probability P(r = R) that there are exactly R HTTP requests waiting in Web server queues, as a function of the ratio λ/μ, given that the service rate μ equals 10 [s^{-1}], for N = 2 Web servers in the cluster

Such probability gets its maximum for R = 1 HTTP request and continually decreases as the value of R grows. For N = 2 Web servers in the cluster, the probability approximates P(r = 1) = 0.16396418 for λ/μ = 1.5 (i.e. for λ = 15 [HTTP requests/s] and μ = 10 [HTTP requests/s]).

The probability P($r \geq R$) that at least R HTTP requests are waiting in Web server queues is depicted in Fig. 8. Obviously, as the ratio ρ = λ/μ increases, such probability rises from values near to 0 to values near to 1. However, the increase becomes more severe as the reference number of HTTP requests in the queues, R, rises, such that P($r \geq$ 1) = 0.95882778, whilst P($r \geq$ 5) = 0.84179076, for λ = 35.

Finally, we also assess the probability P($r \leq R$) that at most R HTTP requests are waiting in Web server queues, graphically depicted in Fig. 9. In this case, the probability decreases as the ratio ρ = λ/μ increases and/or the reference number of HTTP requests in the queues, R, decreases, such that P($r \leq$ 1) = 0.06956791, whilst P($r \leq$ 5) = 0.18810534, for λ = 35 [HTTP requests/s].

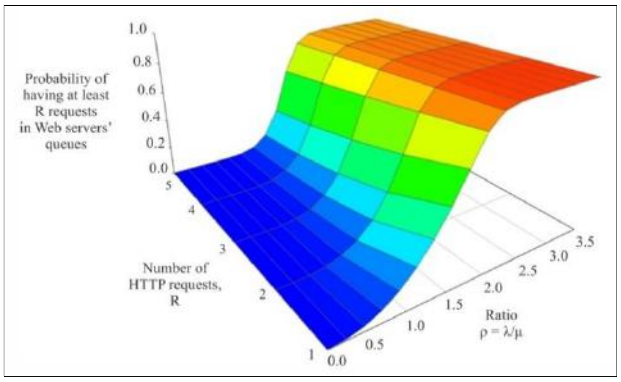

Figure 8. Probability P($r \geq R$) that there are at least R HTTP requests waiting in Web server queues, as a function of the ratio λ/μ, given that the service rate μ equals 10 [s^{-1}], for N = 2 Web servers in the cluster

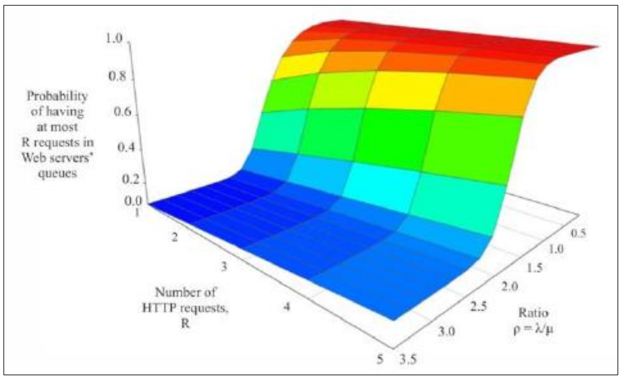

Figure 9. Probability P($r \leq R$) that there are at most R HTTP requests waiting in Web server queues, as a function of the ratio λ/μ, given that the service rate μ equals 10 [s^{-1}], for N = 2 Web servers in the cluster

B. *Transient analysis*

Transient (time-dependent) analysis has been done using the Transient Simulation module. It estimates the system's behavior until a given time point, but it is restricted solely to the assessment of basic reward measures (e.g. excluding the assessment of the waiting time in Web server queues) [15]. The transient analysis has been carried out in the case when the Web server cluster consists of N = 3 Web servers, given that the maximum number of HTTP requests in the system is req = 35, the arrival rate λ = 17 [HTTP requests/s], and μ = 10 [HTTP requests/s].

The evolution of the Web server utilization in the cluster is presented in Fig. 10.

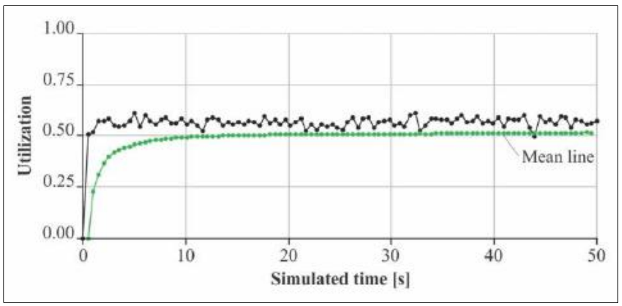

Figure 10. Transient behavior of the Web server utilization

The transient analysis of Web server queue length is depicted in Fig. 11.

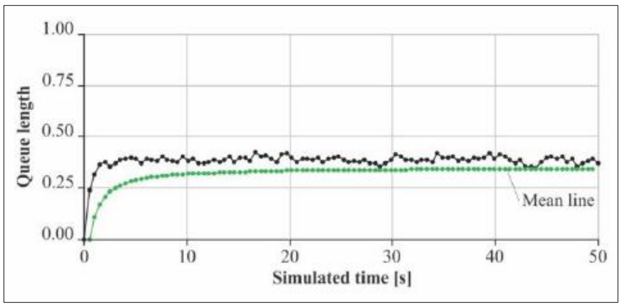

Figure 11. Transient behavior of the Web server queue length

Finally, the transient behavior of the probability P(r = 0) that there are no HTTP requests waiting in Web server queues is shown in Fig. 12.

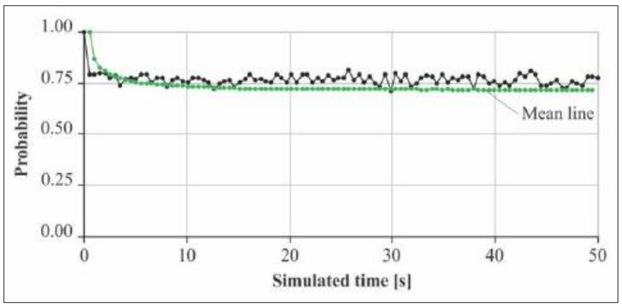

Figure 12. Transient behavior of the probability P(r = 0) that there are no HTTP requests waiting in Web server queues

VI. Conclusion

Since the Internet traffic increases dramatically on a daily basis, especially the one being generated by e-Customers, e-Commerce Websites are facing the challenges of high availability and high responsiveness of their Web servers more than ever before. The results of this study have shown undoubtedly that by the appliance of the horizontal scaling approach, i.e. by increasing the number of Web servers in the first hierarchical tier, and by organizing them logically into a cluster, significant gains in terms of performance can be achieved.

Such GSPN-based approach has few limitations, including (1) the inability to model HTTP requests with variable size (i.e. all tokens are mutually equal, meaning that the HTTP requests they present have the same/similar size); and (2) the inability to model other, more efficient, dynamic schemes of load balancing (i.e. the Round Robin algorithm is suitable/efficient for load balancing in a case when the incoming HTTP requests do not differ significantly from each other vis-à-vis the service demands they pose to Web servers, like in this particular case). In addition, TimeNET® as a software tool also exhibits a number of limitations, like (1) the inability to convey simulations for bigger number of Web servers in the cluster and/or bigger number of HTTP requests in the GSPN-based model (i.e. for larger GSPN models), due to a state-space explosion and a lack of main memory to handle the computation; (2) the limitation to deal with finite state-spaces only; (3) the inability to compute other relevant performance metrics, like the average time tokens need to proceed from one place to another, distant one.

Future research includes utilization of the class of Colored Stochastic Petri Nets (CSPNs) to take into account variable-sized HTTP requests, conducting performance analyses, and carrying out mutual comparisons of the results with those obtained by this study.

References

[1] D. A. Menascé and V. A. F. Almeida, “4.6 Server Architectures,” in Capacity Planning for Web Services: Metrics, Models, and Methods, Upper Saddle River: Prentice Hall PTR, 2002, pp. 149–163.

[2] J. Soininen, “Website Performance Evaluation and Estimation in an E-business Environment,” Tampere University of Technology Publication, vol. 1083, Tampere University of Technology, Tampere, Finland, 2012, pp. 25–31.

[3] R. Prajapati, D. Rathod, and S. Khanna, “Comparison of Static and Dynamic Load Balancing in Grid Computing,” Int. J. Tech. Res. Engin., vol. 2, issue 7, pp. 1337–1340, March 2015.

[4] S. Hamadah, “A Survey: A Comprehensive Study of Static, Dynamic and Hybrid Load Balancing Algorithms,” Int. J. Comp. Sci. Inf. Tech. Sec., vol. 7, no. 2, pp. 27–32, March-April 2017.

[5] H. Rahmawan and Y. S. Gondokaryono, “The simulation of static load balancing algorithms,” 2009 International Conference on Electrical Engineering and Informatics, Selangor, Malaysia, August 2009, pp. 640–645.

[6] H. Liu and S. Wee, “Web Server Farm in the Cloud: Performance Evaluation and Dynamic Architecture,” IEEE International Conference on Cloud Computing (CloudCom 2009), Beijing, China, pp. 369–380, December 2009.

[7] P. Kanungo, “Scheduling Algorithms in Web Servers Clusters,” Int. J. Comp. Sci. Mob. Comp., vol. 2, issue 10, pp. 78–85, October 2013.

[8] X. He and Q. X. Yang, “Performance Evaluation of Distributed Web Server Architectures under E-Commerce Workloads,” International Conference on Internet Computing (IC 2000), Las Vegas, NV, USA, pp. 285–292, June 2000.

[9] Z. Xu and X. Wang, “A modified round-robin load-balancing algorithm for cluster-based web servers,” 33rd Chinese Control Conference, Nanjing, China, pp. 3580–3584, July 2014.

[10] M. E. Mustafa, “Load Balancing Algorithms Round-Robin (RR), Least-Connection and Least Loaded Efficiency,” Computer Science & Telecommunications, vol. 51, issue 1, pp. 25–29, October 2017.

[11] A. Mahmood and I. Rashid, “Comparison of load balancing algorithms for clustered web servers,” 5th International Conference on Information Technology & Multimedia (ICIMU 2011), Kuala Lumpur, Malaysia, pp. 1–6, November 2011.

[12] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, “Modelling with Generalized Stochastic Petri Nets,” Wiley, 1995.

[13] G. Balbo, “Introduction to Generalized Stochastic Petri Nets,” in Formal Methods for Performance Evaluation, SFM 2007, Lecture Notes in Computer Science, vol. 4486, M. Bernardo and J. Hillston, Eds., Berlin, Heidelberg: Springer, 2007, pp. 83–131.

[14] C. Hellfritsch, “TimeNet – Examples of Extended Deterministic and Stochastic Petri Nets,” Technische Universität Ilmenau Publication, Ilmenau, Germany, February 2009.

[15] A. Zimmermann and M. Knoke, “TimeNET 4.0 User Manual,” Faculty of EE&CS Technical Report 2007-13, Technische Universität Berlin, August 2007.

[16] A. Zimmermann, “Modelling and Performance Evaluation with TimeNET 4.4,” 14th International Conference on Quantitative Evaluation of Systems (QEST 2017), Berlin, September 2017.

[17] J. Sztrik, Basic Queueing Theory: Foundations of System Performance Modeling, GlobeEdit, 2016, pp. 55–57, May 2016.