A Cluster-Based Quality Aware Web System

Krzysztof Zatwarnicki, Maciej Płatek and Anna Zatwarnicka

Abstract Providing high quality of Web services is now very important for most of the services. In this article, we present the MLF (Most Loaded First) system, adaptive and intelligent cluster-based Web system. The MLF system provides Quality of Web Service (QoWS) on a fixed level, guaranteeing the page response time within established boundaries, even if Web servers are heavy loaded. We present experiments conducted on real cluster of Web servers and show that MLF system is more reliable than other examined systems.

Keywords Quality of web services · Guaranteeing web page response time · HTTP request scheduling · Request distribution

1 Introduction

The Internet as the medium of information gained popularity at the beginning of the 21st century. The increase of amount of Internet users brought problems with service of large number of clients to many Web services. This resulted in rapid development of Web system technology.

In the past few years most of the works was focused on the Web systems which are very efficient, and able to handle numerous of incoming HTTP requests. Nowadays Internet systems are not only used for entertainment and information purpose but also in business and government administration. This causes the Web

K. Zatwarnicki (✉) · M. Płatek · A. Zatwarnicka
Department of Electroengineering, Automatic Control and Computer Science, Opole University of Technology, ul. Prószkowska 76, 45-758 Opole, Poland
e-mail: k.zatwarnicki@gmail.com

M. Płatek
e-mail: mac.platek@gmail.com

A. Zatwarnicka
e-mail: anna.zatwarnicka@gmail.com

A. Grzech et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*, Advances in Intelligent Systems and Computing 430, DOI 10.1007/978-3-319-28561-0_2

systems should not only be efficient but also reliable and able to service the user in demanded time [6].

The most common way to build highly efficient Internet system is to use cluster of servers, which give the opportunity to scale the system to the needs. The problem of constricting highly efficient cluster-based Web systems was already discussed in our previous works. We have designed a locally distributed Web cluster system [14], a globally distributed Web cluster system with a broker [4] and a globally distributed Web cluster system without a broker [13].

In our later work we proposed new construction of cluster-based MLF Web system providing fixed QoWS [16]. The system was evaluated in simulation experiments which have shown that it works better than other well-known cluster-based systems. This article presents the results of experiments conducted for the MLF system with the use of real Web servers. We analyze these results to determine if the system can provide quality of the service on demanded level.

The proposed system keeps the page response time within established boundaries in such a way that with a heavy workload, the page response times, for both small and complex pages, would not exceed the imposed time limit.

There have been many works on how to guarantee Web service quality. Most of them concern maintaining the quality of the service for individual HTTP requests [1, 2, 5, 7, 8, 10]. Very few papers were dedicated to the problem of designing Web service, which would guarantee servicing the whole WWW pages within a limited time.

QoWS often is combined with granting privileges to certain group of users [11, 12]. The MLF system proposed in this article not only provides fixed QoWS, keeping the page response time within established boundaries, but also treats all users equally.

The paper is divided into four sections. Section 2 describes the MLF system and the method of scheduling and distributing HTTP requests. Section 3 presents the conducted experiments with results, and Sect. 4 contains concluding remarks.

2 MLF System

The MLF system consists of (Fig. 1a): clients sending HTTP requests, MLF Web switch and Web servers. Web Switch and Web servers with database form a Web cluster. The Web switch task is to receive all of the requests from clients, queue them and send into chosen Web servers. The Web server services requests with accompany of database servers and send requests back to the clients.

The Web switch is the most crucial element of the system because it controls the operations of the cluster. Its main task is to deliver entire Web page with the fixed time tmax. To achieve this, the Web switch calculates the term in which each of the incoming requests have to be serviced on the Web server. The requests are queued in the switch according to these calculated terms. When each request leaves the

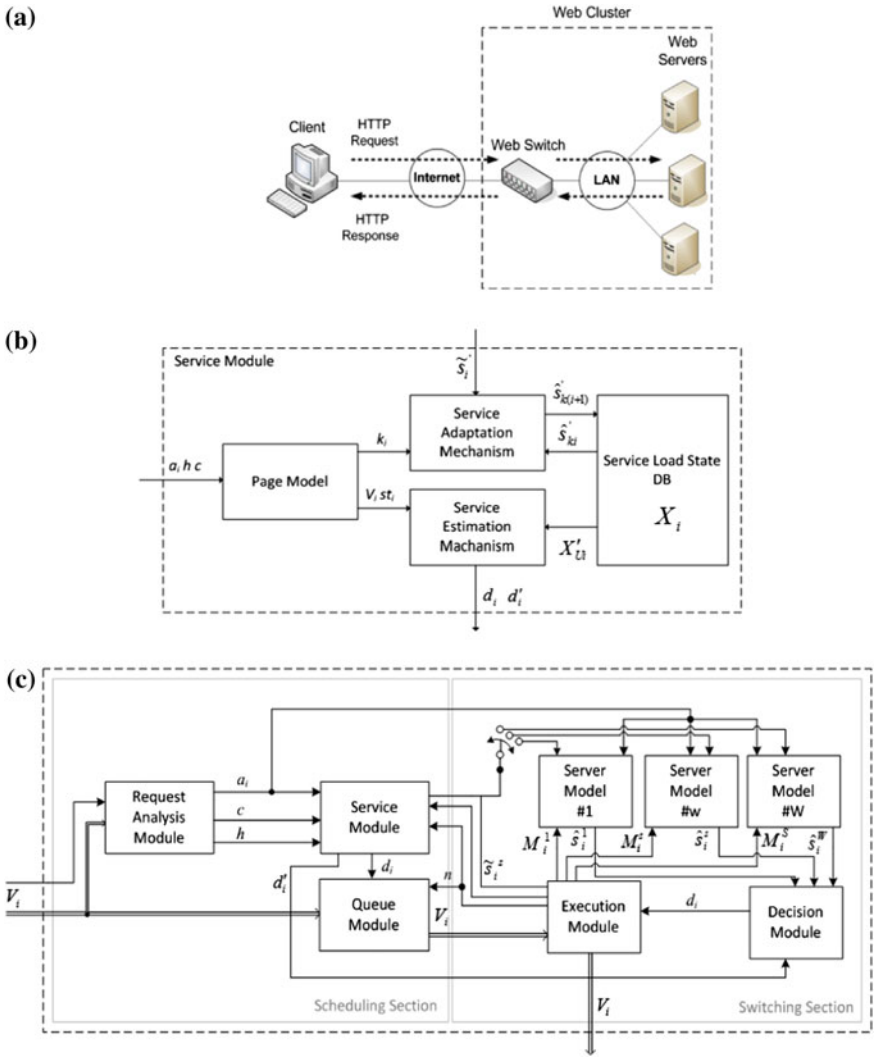

Fig. 1 MLF system: **a** overall view, **b** web switch, **c** service module

queue, the Web switch determines which Web server is able to service the request in required time.

The MLF Web switch consist of two logical sections (Fig. 1c). First, scheduling section is responsible for scheduling request. The second one is a switch section and its role is to distribute requests among Web servers.

Upon receiving incoming HTTP request ri, where i is the request index, the Web switch passes it to the Request Analysis Module (RAM). At the RAM following information are fetched: address ai of requested object, client identifier c and page

identifier h where the requested object belongs. Both identifier h and c are fetched from cookie section of the HTTP request. After receiving all information from RAM the Service Module (SM) computes deadline di which specify when service of the HTTP request has to be started, and d'_i which is the moment when the request service has to come to the end (Fig. 1b). When deadline di is successful calculated, it is used in Queue Module (QM) to queue request according to Earliest Deadline First policy (request are prioritized based on deadline di).

Request skips QM if number of currently serviced requests is smaller than n_{max}, which stand for the lowest value of requests serviced by web servers in the cluster, for which the service reaches the maximal throughput. Primary element of scheduling section is the SM, which is composed of Page Model (PM), Service Load State Database (SLSDB), Service Estimation Mechanism (SEM) and Service Adaptation Mechanism (SAM). The PM stores information about structure of Web pages serviced by the Web cluster, the HTTP clients and the classes of that objects. Basing on a_i, h and c the PM specifies time st_i and vector V_i. Time st_i is measured from the moment the client requests the first object which belongs to the page h to the moment the request u_i arrives. Vector $V_i = \left[k_i^1, \ldots, k_i^l, \ldots, k_i^L\right]$ stores information about classes of objects of page not jet downloaded by the cth client. Classes are composed based on basic information about object. Static objects (files) are partitioned into classes on the base of its size and each of dynamic object (created at the request arrival) has its own class. Object belonging to the same class have similar service time.

The SLSDB stores information about service times for different classes $Z'_i = [\hat{s}'_{1i}, \ldots, \hat{s}'_{ki}, \ldots, \hat{s}'_{Ki}]$, where $\hat{s}'_{ki}$ is the estimated time to service request from specified kth class.

SEM uses information $Z'_{Vi} = [\hat{s}'_{1i}, \ldots, \hat{s}'_{ki}, \ldots, \hat{s}'_{Ki}]$ about service times of objects pointed in the V_i to compute d_i and d'_i. The deadline d_i is calculated as follows: $d_i = \tau_i^{(1)} + \Delta d_i - \hat{s}_{1i}$, where $\tau_i^{(1)}$ is the time of arrival of the ith request, and $\Delta d_i = \hat{s}'_{k^l i}(t_{\max} - h_i) \Big/ \left(\lambda \sum_{l=1}^{L} \hat{s}'_{k^l i}\right)$ is the period of time which the request can spend being queued and serviced by the Web server. The λ is a concurrency factor, and it depends on the number of Web objects currently downloaded for given Web page. According to [15], the value of this factor for average Web pages can be set to $\lambda = 0.267$. The term d'_i is calculated in the following way $d'_i = \tau_i^{(1)} + \Delta d_i$, and it is the term the service of request has to be done by the Web server.

The SAM module is responsible for adapting the information X'_i. Service time for each class of request is updated after queued request leaves queue for service. The modification is calculated as follows: $\hat{s}'_{k\,(i+1)} = \hat{s}'_{k\,i} + \hat{\eta}(\tilde{s}_i - \hat{s}'_{k\,i})$, where $\tilde{s}_i$ is a measured value of the service time, and $\hat{\eta}$ is an adaptation factor.

When the request r_i leaves the QM it is directed to the switching section, which consist server model modules (SMM), a decision module (DM), and an execution module (EM).

SMM's are assigned one-to-one into Web servers working in the cluster. The SMM is responsible for estimation of the request service time $\hat{s}_i^w$ for the Web server the module is assigned to, where $w \in \{1, \ldots, W\}$ is the server index, and W is the number of Web servers in the cluster. The service time is computed on the base of information of the requested object class k_i and the load $M_i^w = \left[e_i^w, f_i^w\right]$ of the server the SMM is assigned to, where e_i^w is the number of requests being concurrently serviced by the wth server and f_i^w is the number of dynamic request concurrently serviced.

The DM chooses the Web server z_i to service the request. The decision is made in the following way:

$$z_i = \begin{cases} \min\left\{w\text{: } w \in \{1, \ldots, W\} \wedge \Delta d_i' \le \hat{s}_i^w\right\} \text{ if } r = r_{\max} \text{ and } \underset{w \in \{1,\ldots,W\}}{\exists} \hat{s}_i^w \ge \Delta d_i' \\ w_{\min}\text{: } \hat{s}_i^{w_{\min}} = \min\left\{\hat{s}_i^w : \ w \in \{1, \ldots, W\}\right\} \text{ in other case} \end{cases}, \tag{1}$$

where $\Delta d_i' = \tau_i^{(2)} - d_i'$, and $\tau_i^{(2)}$ is the moment the request r_i leaves the QM. Using formula (1) the DM chooses the server with the lowest index, which is able to service the request in a time not longer than $\Delta d'$. If such server is not available, it offers the server with the shortest service time. Proposed solution guarantee that the Web servers with the lowest indexes are the most loaded, but still have ability to service requests, and the Web servers with the higher indexes are unloaded and able to handle requests in a short time.

After the decision z_i is taken, the EM sends the request r_i to the chosen server. The module also collects information e_i^w and f_i^w, and measures service time $\tilde{s}_i$.

The SMM is the most complex module of the switching section. It use a neuro-fuzzy model to estimate service time $\hat{s}_i^w$ of the request for the given Web server (Fig. 2a). For each class $k_i = 1, \ldots, K$ of objects the same model is used, but with different parameters (weights). Parameter database $Z_i = [Z_{1i}, \ldots, Z_{ki}, \ldots, Z_{Ki}]$ stores parameters for different classes, where $Z_{ki} = [C_{ki}, D_{ki}, S_{ki}]$, $C_{ki} = \left[c_{1\,ki}, \ldots, c_{l\,ki}, \ldots, c_{(L-1)\,ki}\right]$ and $D_{ki} = \left[d_{1\,ki}, \ldots, d_{m\,ki}, \ldots, d_{(M-1)\,ki}\right]$ are parameters of input fuzzy set functions, and $S_{ki} = \left[s_{1\,ki}, \ldots, s_{j\,ki}, \ldots, s_{J\,ki}\right]$ are parameters of output fuzzy set functions. Input fuzzy set functions are triangular (Fig. 2b) and are denoted as $\mu_{F_{el}}(e_i)$, $\mu_{F_{fm}}(f_i)$, $l = 1, \ldots, L$, $m = 1, \ldots, M$, whereas output fuzzy sets functions $\mu_{Sj}(s)$ are singletons (Fig. 2c). The values L and M were chosen experimentally and set to 5, and $J = L \cdot M$.

The service time is calculated in the following way: $\hat{s}_i = \sum_{j=1}^{J} s_{jki} \mu_{R_j}(e_i, f_i)$, where $\mu_{R_j}(e_i, f_i) = \mu_{F_{el}}(e_i) \cdot \mu_{F_{fm}}(f_i)$. The values of parameters C_{ki}, D_{ki}, S_{ki} are modified in an adaptation process using the Back Propagation Method each time the service of the request on a given server finishes. The parameters of output fuzzy sets are modified as follow: $s_{j\,k(i+1)} = s_{j\,ki} + \eta_s \cdot (\tilde{s}_i - \hat{s}_i) \cdot \mu_{R_j}(e_i, f_i)$, whereas parameters of input fuzzy sets are computed in following way $c_{\varphi k(i+1)} = c_{\varphi ki} + \eta_c(\tilde{s}_i - \hat{s}_i)$

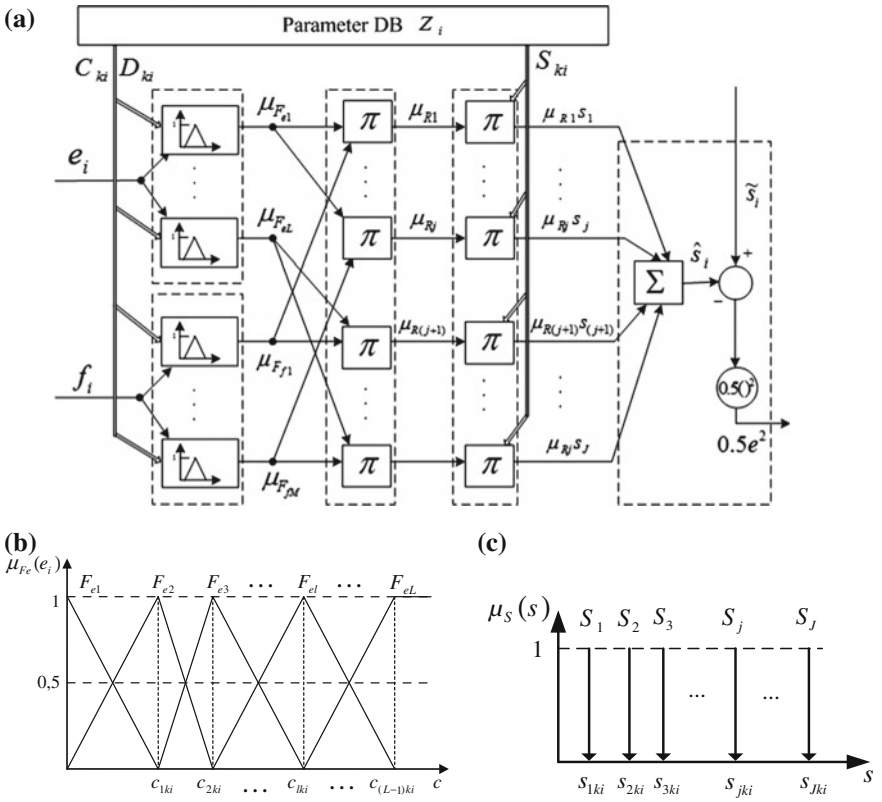

Fig. 2 Server model: **a** neuro-fuzzy model, **b** input fuzzy sets functions, **c** output fuzzy sets functions

$\sum_{m=1}^{M}\left(\mu_{F_{fm}}(f_i)\sum_{l=1}^{L}\left(s_{((m-1)\cdot L+l)ki}\partial\mu_{F_{el}}(e_i)/\partial c_{\varphi ki}\right)\right)$ and $d_{\gamma k(i+1)} = d_{\gamma ki} + \eta_d$ $(\tilde{s}_i - \hat{s}_i)\sum_{l=1}^{L}\left(\mu_{F_{el}}(e_i)\sum_{m=1}^{M}\left(s_{((l-1)\cdot M+m)ki}\partial\mu_{F_{fm}}(f_i)\Big/\partial d_{\gamma ki}\right)\right)$, where η_s, η_c,η_d are adaptation ratios, $\varphi = 1,\ldots,L-1$, $\gamma = 1,\ldots,M-1$.

3 Testbed and Results of Experiments

Our previous research [16] shows that MLF system can provide higher QoWS than other reference distribution methods. The main goal of presented in this article experiments was to evaluate the MLF system using real Web server and the Web switch. We tried to determine if MLF system can provide service with demanded quality of the service.

The experiments were conducted with use of six computers. First of them was responsible for simulating the behavior of Web clients. The next four were Web servers and the last one was a Web switch. Table 1 presents their hardware configuration.

Web servers had the lowest computational power which allowed to reach maximal number of requests they can handle without overloading other elements of the system. All computers were connected to LAN network using gigabyte Repotec RP-G3224V network switch.

The Web server hosted five different Web pages. Table 2 presents the structure of the pages.

Pages contained from 10 to 30 embedded objects with size from 1 to 100 KB and the dynamic pages used PHP as the script language to generate the content of the page. Moreover one page used SQL requests to the MySQL database which contain three related tables containing 10,000 rows each.

Software used in Web switch was written in the C++ language with the use of the following libraries: *libsoup* [9] to implement virtual Web server managing incoming HTTP requests and *boost* [3] for time measuring.

Along with the MLF method other popular distribution methods were implemented in the Web switch:

Table 1 Hardware configuration of computers used in experiments

Name	CPU	Operating system
Web servers	Intel Celeron 1.7 GHz	Ubuntu 13.04 Server
Web switch	Intel Core i7-2670 2.2 GHz	Fedora 18
Client simulator	Intel Core i5-3470 3.3 GHz	Ubuntu 13.04 Desktop

Table 2 Web pages used in the experiments

Name	Type and size of the frame object of the page	Embedded objects number and sizes	MySQL database
Static 10	Static 1 KB	10, size 1–100 KB, sum 477 KB	–
Static 30	Static 1 KB	30, size 1–100 KB, sum 1.39 MB	–
Dynamic 10	Dynamic, PHP	10, size 1–100 KB, sum 477 MB	–
Dynamic 30	Dynamic, PHP	30, size 1–100 KB, sum 1.39 MB	–
Dynamic MySQL 30	Dynamic, PHP	30, size 1–100 KB, sum 1.39 MB	Requests to database containing 3 tables, 10,000 rows each

- LARD (Locality-Aware Request Distribution): one of the best distribution methods taking into account localization of the previously requested object,
- CAP (Content Aware Policy): an algorithm uniformly distributing HTTP requests of different types,
- RR (Round-Robin): an algorithm distributing uniformly all incoming requests.

Simulation of clients was done by HTTP Request Generator (RG), which was written in the C++ language with the use of the *LibcURL* library. The way of working of the RG was similar to way of operation of modern Web browsers. Each client generated by RG was opening up to 6 TCP connections to download single Web page.

Except of generating HTTP requests, the RG was able to collect information about requests, such as: mean value of request response time and satisfaction. Satisfaction is the measure of user's satisfaction of the page response time. The user's satisfaction depends on the response time of the whole page according to function presented on Fig. 3. Value of the satisfaction is equal to 1 when the page response time is shorter than $t^s_{\max}$, and reaches value 0 when the time is longer than $t^h_{\max}$. In all of the experiments following values of $t^s_{\max} = t_{\max} = 2000$ ms and $t^h_{\max} = 2t^s_{\max}$ were adopted.

The Fig. 4a presents diagram of the mean request response time vs. the load for different Web pages. Diagrams from b to f in Fig. 4 present satisfaction vs. number of clients generating HTTP requests. As one can see the highest value of satisfaction was obtained by the MLF method for static small pages as well as pages containing more embedded objects. Results for dynamic pages are not spectacular however in every experiment the satisfaction is higher for the MLF then for other methods. Results presenting the mean request response time shows that for the MLF method not only the satisfaction is the highest but also the response time is no longer than for other methods.

The presented results confirm results obtained during simulations and it can be concluded that using MLF method can help to provide QoWS on a fixed level in the cluster-based Web systems.

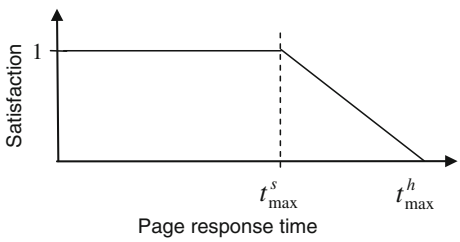

Fig. 3 Satisfaction function

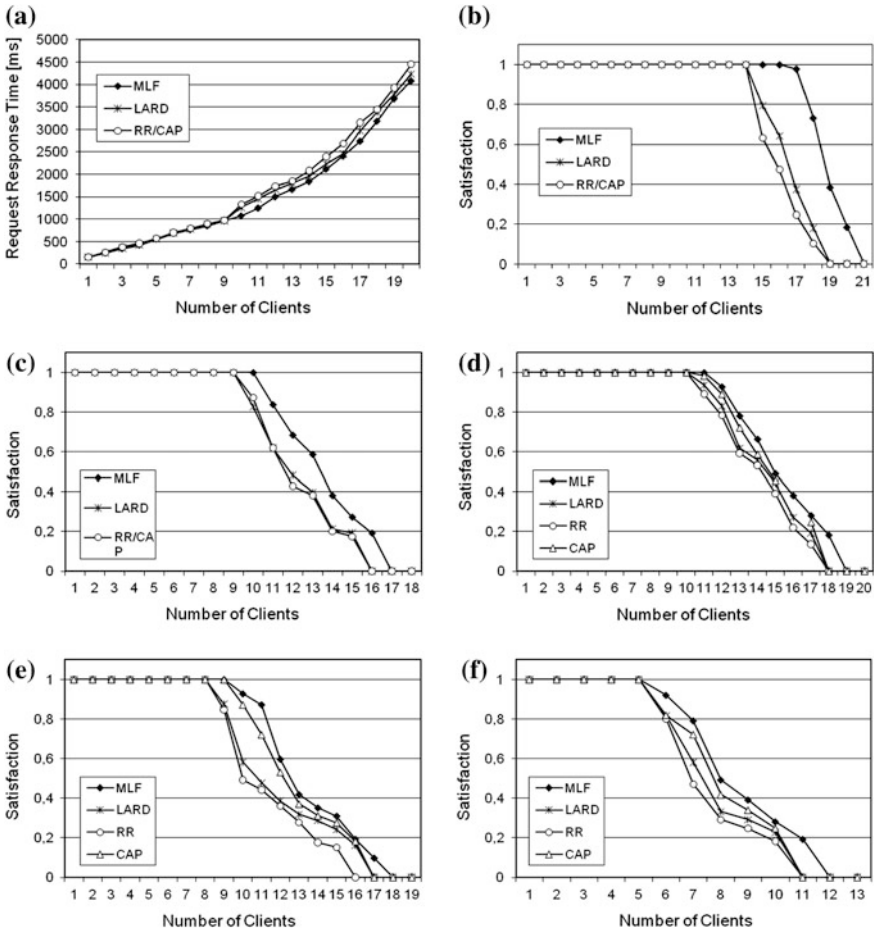

Fig. 4 The results of experiments. **a** Mean request response time for Static 10; Satisfaction versus load for the Web pages: **b** static 10, **c** static 30, **d** dynamic 10, **e** dynamic 30, **f** dynamic MySQL 30

4 Summary

In this paper we presented HTTP request scheduling and distribution cluster-based Web system, which provide service at fixed level. The proposed MLF system can deliver requested content for the users with higher value of satisfaction than other examined methods. Decision algorithms, used in the MLF method, applies adaptive algorithms using neuro-fuzzy models in its construction. Obtained results confirms experiments conducted during simulations and presented in other articles. At high load on Web servers, the MLF system can provide acceptable value of satisfaction for higher number of clients than other tested methods. When comparing the same load for each method it is noticeable that MLF system supplies HTTP requests in

lower response times, causing higher values of satisfaction in each case. The results of experiments show that MLF method can makes Web system more reliable than it will be under control of other examined methods.

References

1. Abdelzaher, T.F., Shin, K.G., Bhatti, N.: Performance guarantees for web server end-systems: a control-theoretical approach. IEEE Trans. Parallel Distrib Syst **13**(1), 80–96 (2002)
2. Blanquer, J.M., Batchelli, A., Schauser K., Wolski R.: Quorum: Flexible quality of service for internet services. In: Proceedings of Symposium on Networked Systems Design and Implementation (2005)
3. Boost C++ libraries, http://www.boost.org/. Access 10 Sept 2015
4. Borzemski, L., Zatwarnicka, A., Zatwarnicki, K.: Global Adaptive Request Distribution with broker, Vol. 4693, pp. 271–278. Lecture Notes in Artificial Intelligence. Springer, Berlin (2007)
5. Harchol-Balter, M., Schroeder, B., Bansal, N., Agrawal, M.: Size-based scheduling to improve web performance. ACM Trans. Comput. Syst. **21**(2), 207–233 (2003)
6. Hunek, W.P., Dzierwa, P.: New results in generalized minimum variance control of computer networks. J. Inf. Technol. Control **43**(3), 315–320 (2014)
7. Kamra, A., Misra, V., Nahum, E.: Yaksha: A self tubing controller for managing the performance of 3-tiered websites. In: Proceedings of International Workshop on Quality of Service, pp. 47–56 (2004)
8. Kanodia, V., Knightly, E.: Multi-class latency-bounded web services. In Poceedings of te 8th International Workshop on Quality of Service (IWQOS) (2000)
9. Libsoup library description, http://developer.gnome.org/libsoup/stable/. Access 10 Sept 2015
10. Schroeder, B., Harchol-Balter, M.: Web servers under overload: how scheduling can help. In: 18th International Teletraffic Congress, Berlin, Germany (2003)
11. Suchacka, G., Borzemski, L.: Simulation-based performancestudy of e-commerce Web server system-results for FIFO scheduling. In: Multimedia and Internet Systems: Theory and Practice, AISC, Vol. 183, pp. 249–259. Springer, Berlin (2013)
12. Wie, J., Xue, C.Z., QoS: Provisioning of client-perceived end-to-end QoS guarantees in web servers. IEEE Trans. Comput. **55**(12) (2006)
13. Zatwarnicki, K.: Neuro-Fuzzy Models in Global HTTP Request Distribution, Vol. 6421/2010, pp. 1–10. Lecture Notes in Computer Science. Springer, Berlin (2010)
14. Zatwarnicki, K.: Adaptive control of cluster-based web systems using neuro-fuzzy models. Int. J. Appl. Math. Comput. Sci. (AMCS) **22**(2), 365–377 (2012)
15. Zatwarnicki, K., Zatwarnicka, A.: Estimation of Web Page Download Time, Communication in Computer and Information Science, Vol. 291/2012, pp. 144–152. Springer, Berlin (2012)
16. Zatwarnicki, K.: Operation of Cluster-Based Web System Guaranteeing Web Page Response Time, Vol. 8083/2013, pp. 477–486. Lecture Notes in Artificial Intelligence. Springer, Berlin (2013)

http://www.springer.com/978-3-319-28559-7

Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II

Grzech, A.; Borzemski, L.; Swiatek, J.; Wilimowska, Z. (Eds.)

2016, XIV, 243 p. 82 illus., 33 illus. in color., Softcover
ISBN: 978-3-319-28559-7