MODEL AND ALGORITHM FOR INCREASING THE EFFICIENCY OF REMOTE SERVICE SYSTEMS SERVER

Vakhid Zakirov and Eldor Abdullaev

Department of Radioelectronic devices and systems, Tashkent state transport university, Tashkent, Uzbekistan

ABSTRACT

This article examines the aspects that influence productivity in distant service systems during times of request congestion. At the same time, the issue of enhancing system server efficiency without the use of additional hardware or software has been investigated. It has been found that service system efficiency declines during traffic congestion because requests are processed in multiple stages, each with a different service order. Hence, to enhance system efficiency during traffic congestion, a proposal is made to prioritize users whose requests have been successfully served at least once. A physical model of the service process was constructed to assess the efficacy of the proposed approach.The research conducted using this model contributed to an average increase in the efficiency of service systems by 6%.

KEYWORDS

server performance indicator, priority service, request traffic, service algorithm, mass service model, service time

1. INTRODUCTION

Our dependence on web applications is steadily deepening, permeating various facets of our lives. These platforms serve as conduits for a myriad of services across different social strata, fueling a surge in their usage. Consequently, this heightened utilization brings forth a host of systemic challenges [7], [12], [29]. Specifically, as user numbers and requests increase, the system grapples with escalating request traffic, resulting in diminished server performance metrics. Hence, optimizing server efficiency amidst traffic congestion emerges as a paramount concern [2], [5], [13], [30].

The significance of the situation has resulted in the carrying out of numerous studies in this area. Such studies were conducted in the following directions: implementation of various hardware and code optimizations in the service of requests [4], [20], [21], [27], server service times for request reduction [18], load balancing [22], [25], [26], service time prediction [28], web server performance evaluation [23], [24].

The focus of research into various optimizations in the service of requests is to improve the user interface, reduce server load, and increase system interaction through the use of FrontEnd, Backend, and database technologies. These optimization methods have been built in order to reduce latency and increase system efficiency.

The research aimed to reduce server request service times by utilizing various online optimization methods, such as Newton's Method, Fuzzy Control, Heuristic Approaches, Feedback Mechanisms, and Qualitative Comparisons [4], [20], [21], [27]. These optimizations were applied to the hardware, software backend, and frontend components.Additionally, the research examined the impact of technical device configurations, including browser caching, DNS caching, HTTP keep-alive timeout, and gzip compression, on their effectiveness and the dynamic changes in request volume.

The research also explored load balancing techniques to improve system efficiency. The goal was to equally distribute the incoming requests and their associated load among the service servers [22], [25], [26].

The research also involved predicting service times and evaluating web server performance. Web monitoring tools were used to analyze how web server performance depends on factors such as the distance between the server and the user, as well as the size of the web page [23], [24], [28].

Unlike earlier researches, this study focuses on developing algorithms and models to modify the order of service requests when the system becomes congested. The purpose is to handle circumstances where traffic jams develop in the service system, which were not addressed by the previous activities stated.

The research work consists of five sections, conclusions, and a reference list. The analysis of the reasons for the problem is presented in the second section, the methods used in the research in the third section, the experimental setup section is located in the fourth section, and the results and discussions of the research in the fifth section.

2. PROBLEM STATEMENT

The problem of request congestion in the system usually occurs during times of high intensity of requests entering the system. Also, the origin of traffic directly depends on the service algorithm of the system. Currently, requests are served using a web server in the order shown in Figure 1.

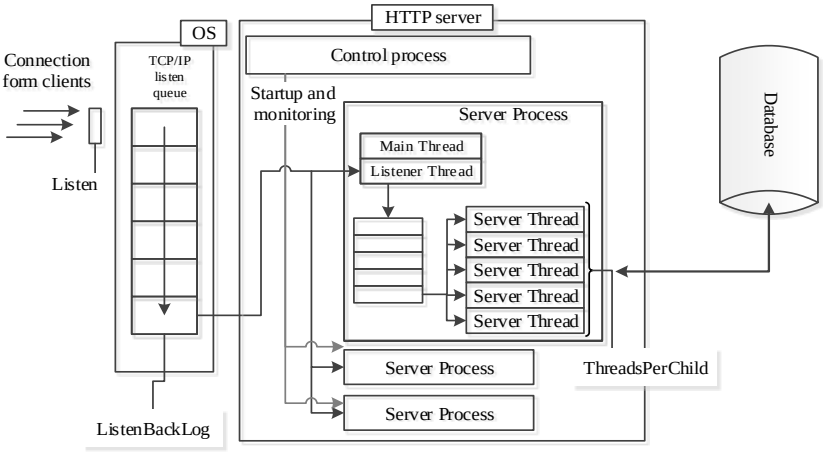

Figure 1. The way the web server serves requests

Users typically request multiple times from the system in order to receive the one crucial piece of information. In this instance, the system's response time to the user's first and subsequent requests will differ. In this instance, the user's initial request is authenticated by the system.Authentication

necessitates a sequence of checks between the user and the system, including seeking up the identification information in the system database, verifying its validity, and then allocating a service device to the user, which takes more time to service requests. This technique may also be associated with receiving data from external devices. This procedure is not repeated for further user requests for this information, unless the request interval exceeds the system's user-assigned session time. As a result, the time taken to service these two types of requests is different. As a result, during the identification process, the system provides no helpful service to the user and consumes system resources inefficiently. This creates various delays in allowing new users into the system and serving other requests into the system when there is traffic, resulting in a drop in system efficiency.

Requests that have successfully passed the identification process will continue to be served if the service devices are empty.

If all service devices are busy, they will be queued. At the same time, since the number of system service devices and request queues has a certain threshold value, it causes them to be quickly filled with requests during times of high traffic.When the service devices are empty, the system can serve the requests in the waiting areas in different order. One of these FIFO (First in First out), LIFO (Last in First out) or SIRO (Service in random order) methods can be used [16]. In our view, requests in waiting areas are served on a FIFO basis. In the process, the system does not care whether the user has passed the identification or not and accepts the request for service.This causes the user's requests to remain in the queue more often or to be excluded from the service. Because of this, the system's efficiency declines.

As a consequence, developing appropriate approaches, models, and algorithms for solving these types of challenges is one of today's most critical issues.. At the same time, troubleshooting without using additional hardware or software tools does not increase the system server load and does not slow down the service process. Therefore, it is important to research service algorithms, models, and optimization processes to achieve positive results.

3. METHODS

Taking into account the difference in service time for the initial and subsequent requests mentioned above gives some degree of results when solving problems of service requests during high-traffic times.For this, it is necessary to divide the requests into two types. The first type includes requests that need to go through the identification process. The latter includes requests that have previously successfully passed identification to obtain a single piece of information from this process. It allows to divide the requests into two types, to serve them in different order during times of high traffic. This reduces the time it takes to service a request and thus increases the number of fully serviced requests.

We can use the following model of mass service to divide incoming requests into two and serve them in different order [14], [15], [16], [17](Figure 2). This model is a service model based on the priority of requests during traffic congestion.

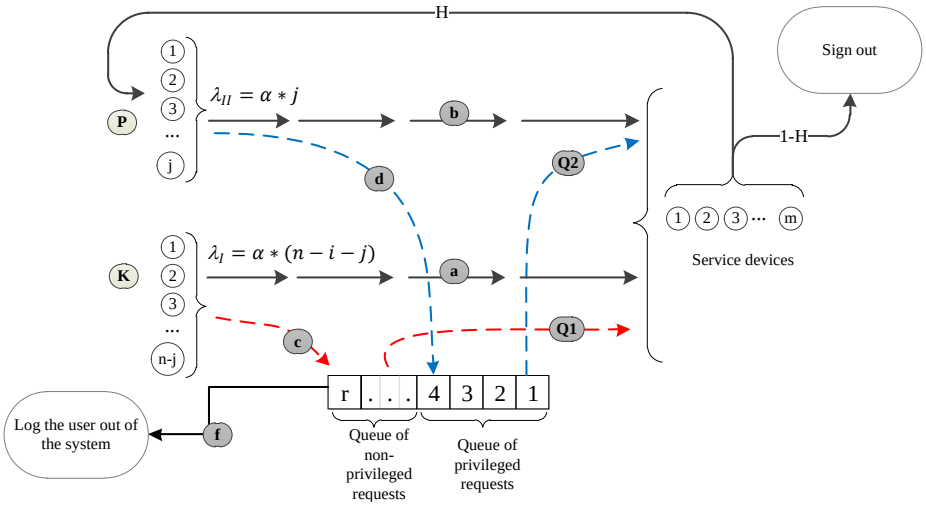

Figure 2.A service model based on the priority of requests during traffic congestion

In this process, two groups of K and P - unprivileged and privileged requests are coming to the system. The first K group of users form the initial request stream. They are equal to $\lambda_I = \alpha \cdot (n - i - j)$ the total rate, and the α incoming requests from the n source, each of which has a rate, create flows.The second P group is formed by the flow of requests successfully served at least once by the system $\lambda_{II} = \alpha * j$.

A common request stream from these two sources K and P is served in standby mode $\lambda_\Sigma = \lambda_I + \lambda_{II}$ by a system of service devices m (e.g., a server).The total number of request queues is r.

In this case, service of requests is carried out in two stages - preliminary and main. At the initial stage, group K requests go through the identification process. In the process of identification, an HTTP request, information exchange with the DNS server, etc. are carried out. At the second stage, the necessary information exchange is carried out and the requests are fully serviced. If the user takes the necessary information for itself and wants to complete his work, then he will log out with a probability of 1-H.

A request received through the first K group goes through the request identification process when the service devices are free. In the second step, the service device (server) prepares the desired result for the request and sends it to the user. If the process is completed successfully, the source is assigned a certain symbol - the "priority" symbol, and the source is included in the group P with probability H. Such sources form the second group of sources. When traffic is not observed in the system, all K and P group requests are served in the same order. In times of traffic congestion, requests with a priority mark are served with a certain privilege.This privilege consists of the following. If there is an empty service device at the moment of time when the request falls, then such a request is served immediately without an identification process, and the service is carried out directly through channel b in Figure 2.If there is no free service device available at the moment of the request, privileged requests are transferred to the request queue through the d channel and are served in the FIFO method [2]. Requests in non-privileged queues are serviced after privileged requests have been serviced.

If all the service devices are busy at the moment of the request from the first K group source, the request is transferred to the standby devices through channel c. If the queues are busy, the request

will be rejected.If all service devices are busy when a request is received from the second source P, the request is transferred to the queue devices. If all the queues are busy, the request from the first K source in the queue is rejected through channel f, and the second source request is placed in the queue.

Also, this process is terminated when the service devices of the system are free, and the service of requests from the source K is continued. It should also be noted that there is a number of requests that can be received simultaneously from groups K and P, and these values are equal to n in total.Here, n is equal to the sum of service devices - m and the number of queues - r. In this case, when there are no users in the system at all, n values belong to K groups. However, as soon as there are users who successfully used the system at least once, n values are given to P group. The number of users in group P is j.And when it is equal to the value of j, incoming requests in the system are accepted only through group P, and all users are considered users with priority. The algorithmic view of this model is as follows (Figure 3).

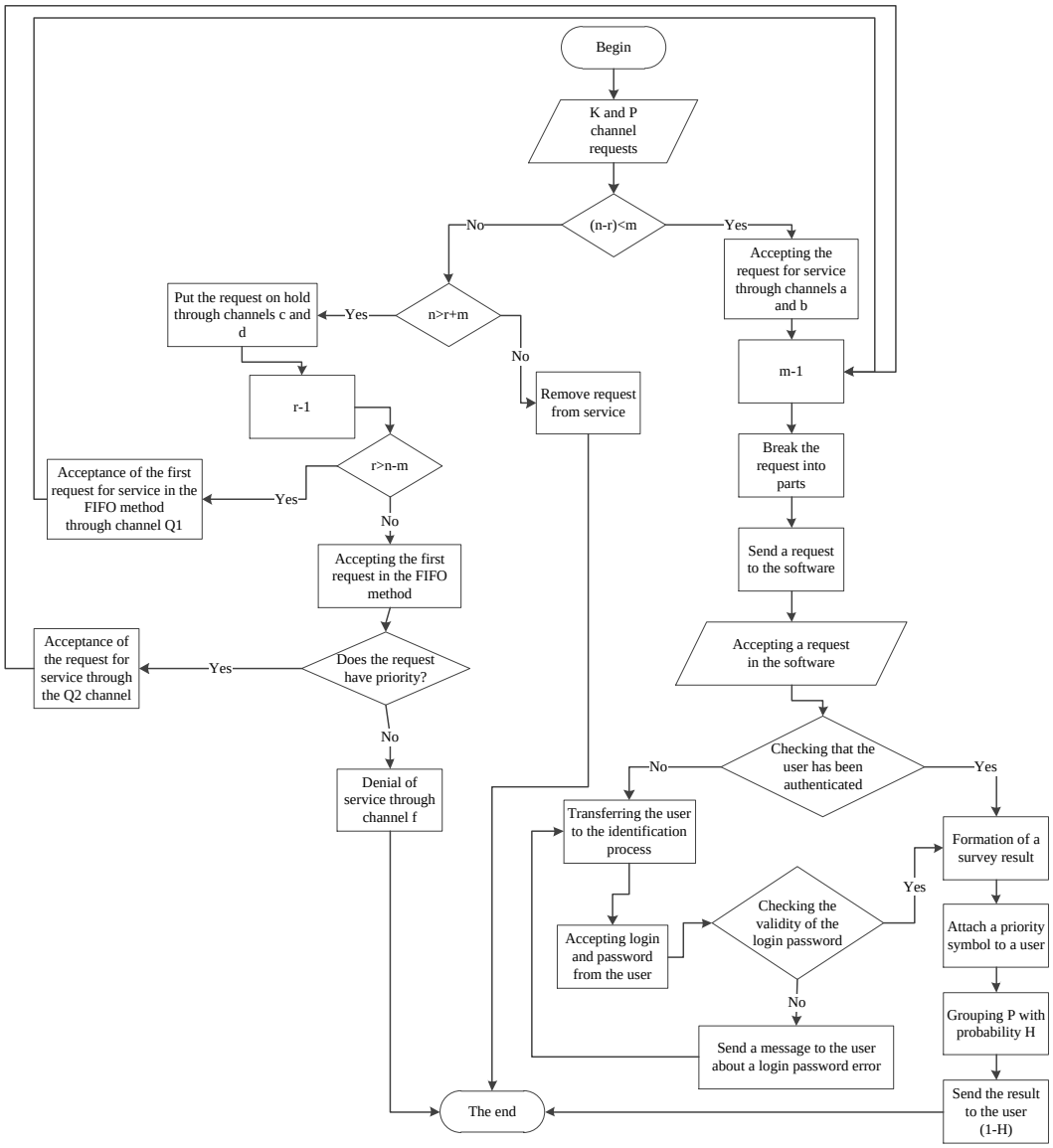

Figure 3. Algorithm of priority service during traffic congestion

4. EXPERIMENTAL SETUP

To solve the models of this type and to determine the quality indicators of the system (waiting probability, waiting time, etc.), it is possible to use two methods - strictly mathematical and modeling methods. The mathematical method is based on constructing and solving the steady state equation of the system using the Markov birth and death process for Markov chains or one type of process [16]. But this process is a complex and separate research work, which is being carried out in our next studies. In this research work, solutions and results of this problem based on physical modeling are given.

Based on the model shown in Figure 2 above, management of incoming requests to the server and priority service during traffic jams were implemented using the system's web server.In this case, the number of service threads of the web server was evaluated by evaluating the server's ability to provide service at a certain time (in the study, the server's capabilities in 1 second). The research was conducted on a server device with technical indicators Montage Jintide® C5220R 2.20 GHz, RAM 32.0 GB, HDD 500 GB. Also, openresty-1.25.3.1 version of nginx was used as the system web server.

5. RESULT AND DISCUSSION

The service capabilities of the server in one second were determined based on its technical indicators according to the methods carried out in the study [3]. In doing so, server process time (SPT), queuing time (QPT) and server connection time (RTT) were determined under different system conditions (Table 1). The Apache Jmeter 5.6.3 program was used to generate different volumes of users and requests in the system for a certain period of time. In this case, it was noted that the user's internet speed was 23 Mbit/s, and the server's internet speed was 170 Mbit/s.

Table 1. Server Capability Analysis Table

Test number	**Number of requests in the system (from different users)**	**Server process time (SPT), ms**	**Queuing time (QPT), ms**	**Server connection time (RTT), ms**	**Server availability, (in one second)**
1	Regular work times	90	1,49	43	177
2	10	260	1,8	78	77
3	50	465	4,47	102	49
4	100	684	1,48	194	30
5	200	1140	1,5	204	21
6	400	4200	2,43	256	7
7	500	6600	1,86	312	5
8	700	8720	3,35	409	4
9	800	9750	2,05	564	3
10	1000	10035	2,48	632	3

Calculations based on these obtained values estimated the server's service capacity per second at 177. However, when the number of users utilising the system hit 1000, this value decreased to 3. Because, in this process, the large number of service lines created by the web server for each user caused the server's response time to increase. This in itself had an impact on the efficiency of the server.

As defined in the research work, the web server's service method for users is adapted to the model presented in Figure 2, and research on the organization of the service process and the reduction of the server's loss of requests during times of high traffic of requests were carried out as follows.

First, the user service algorithm of the nginx web server was analyzed and changes were made to its operation algorithm.These changes include constantly monitoring whether a service thread is in use or not, and once the server reaches a concurrent no-rejects service value, users are served based on whether or not they have previously used the system (Figures 4 and 5, changes and additions are shown in yellow in Figure 5).

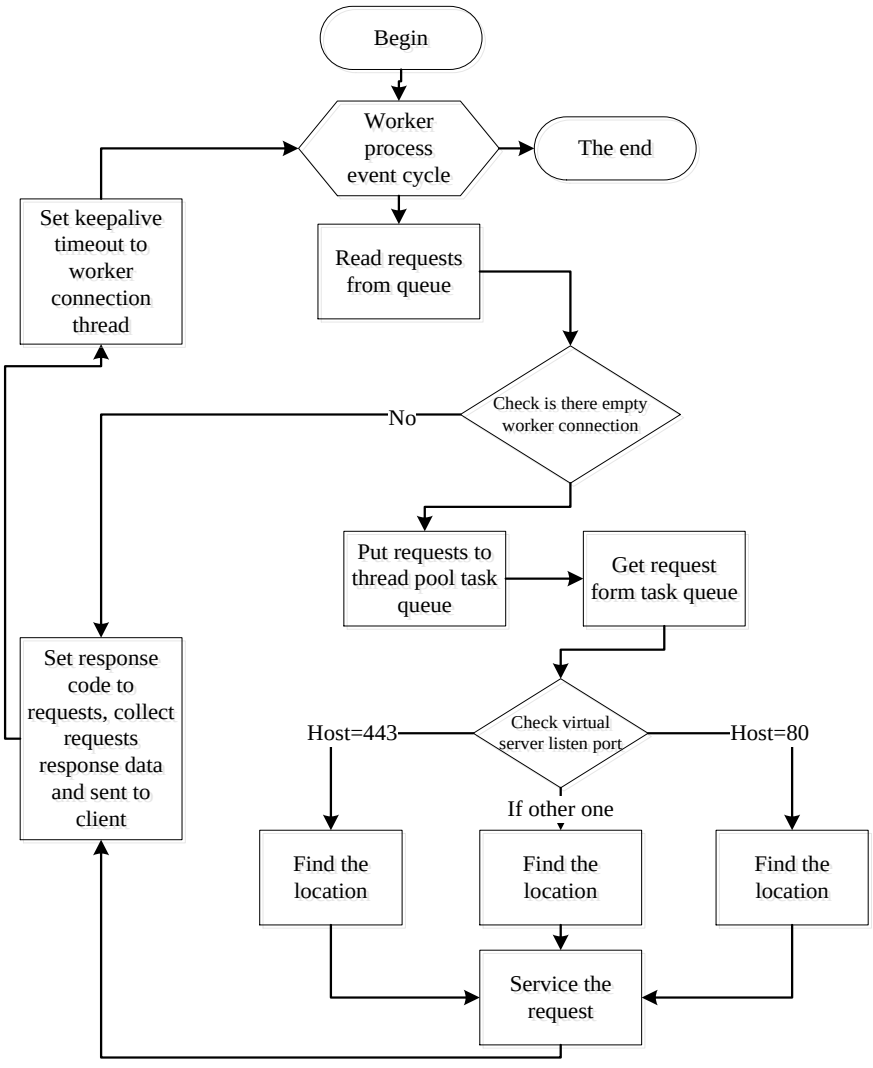

Figure 4. The default service algorithm of the web server for requests

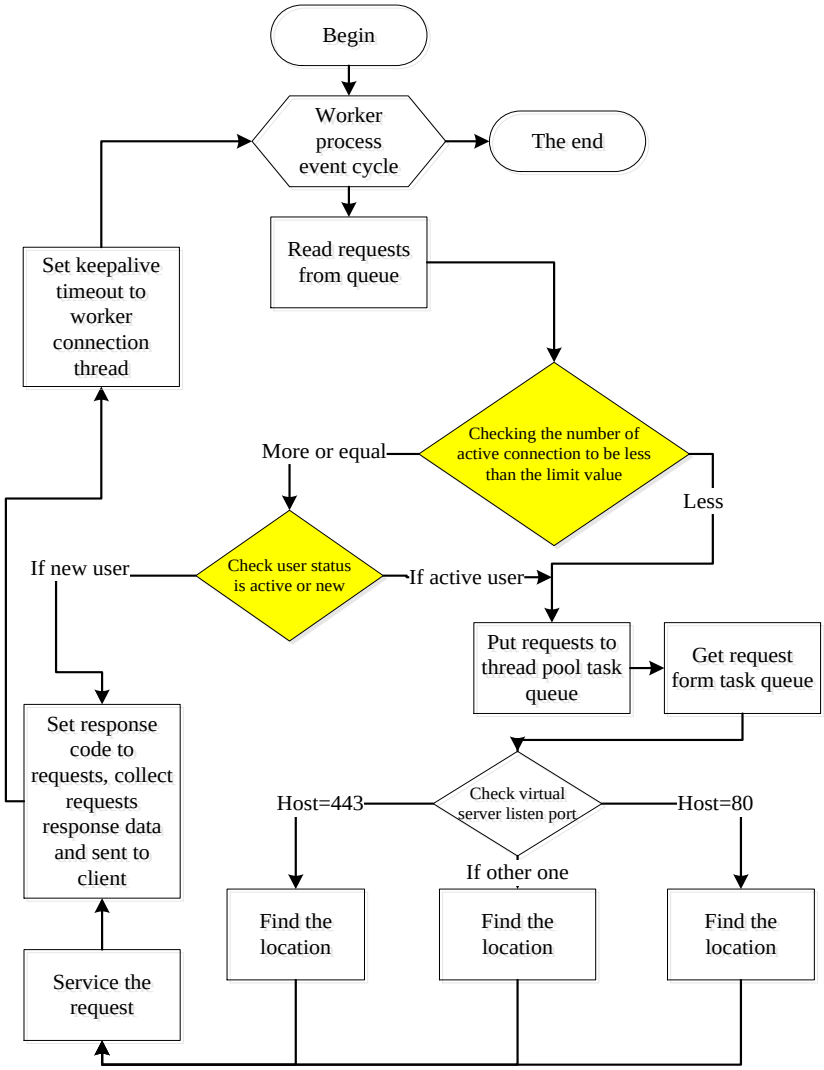

Figure 5. Algorithm of the web server for serving requests with priority

The part of the algorithm changes to control the number of threads in constant use served to adapt the service threads created in the system to the ability of the server to provide services at the same time. Because, according to the standard settings, the web server creates a separate service thread for a certain time for any new user admitted to the system.This increased the loading process of the system and the time it takes to alternately refer to threads in use. As a result, interruptions in the system's response to temporary requests occurred and led to a sharp decrease in the server's ability to provide concurrent services.

Therefore, system service threads are assigned to the level of server availability, to ensure the continuous performance of the server during times of high request traffic and to prevent service times from exceeding dramatically.

Service requests through the algorithm based on the above model, during high traffic times of requests, serving requests of users with priority status using the system and temporarily limiting service requests of new users who have entered the system at that time made it possible. This, in turn, raised the number of requests handled by an average of 5-7% while maintaining request service time during peak times nearly unchanged as compared to typical algorithm-based web

servers, resulting in significantly lower server downtime (Figures 6 and 7). Also, all users' requests have the same priority when serving requests with the default and optimized settings that are presented in [1], [4], [**Error! Reference source not found.**], [7], [9], [10], [11]. Therefore, operating based on the standard algorithm, the probability of rejection was almost the same. For this reason, most of the requests of users who used the system before were not served.

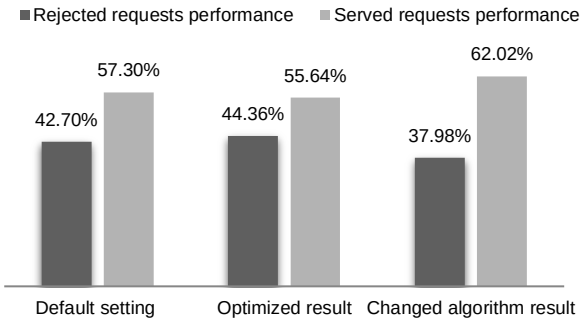

Figure 6. Request service performance during high traffic times

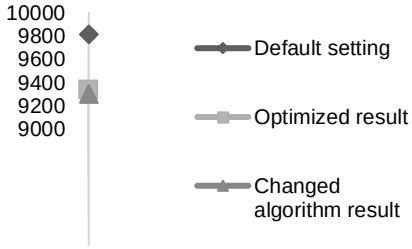

Figure 7. Request service time (ms)

Based on the above model and the modified algorithm, almost 83% of user requests with service priority were served, and this was 18% better than a web server with standard settings, and 22% better than those with optimized settings. As a result, the loss rate of new user requests increased by 9% compared to the indicators of standard and optimized settings, but compared to the rejection rate of requests in total, it was less than 5-7% (Figure8). There was also a significant change in the latency of user requests to connect to the server, and its results are shown in Figure 9.

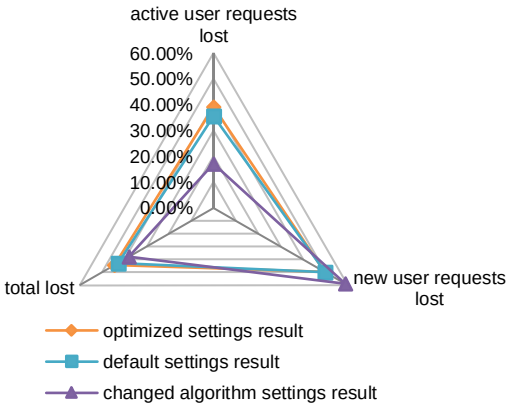

Figure 8. Rejection rates of requests for user statuses

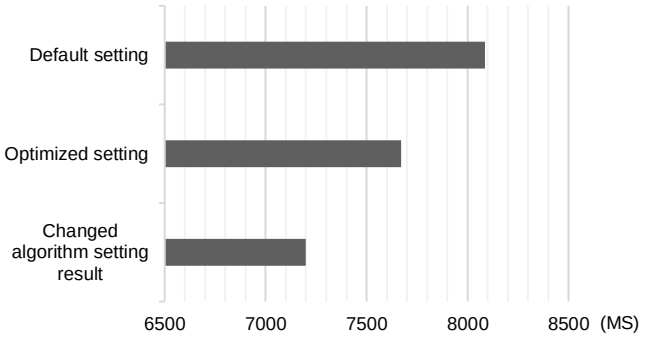

Figure 9. Average server connection latency

6. CONCLUSION

In remote service systems, users are offered multi-level assistance for their requests. Initially, users undergo an identification process. However, during periods of congestion, this process can lead to decreased system performance. During the identification process, the system effectively ties up service devices for a certain duration, thereby hindering the processing of requests at subsequent stages. This inefficiency can lead to delays and decreased overall system performance. According to the model proposed in this study, serving requests during high traffic times contributed to alleviating request congestion within the system. Consequently, server connection delays decreased by 6.2% and 11% respectively, in comparison to systems utilizing standard settings and optimized settings based on conventional work algorithms. Furthermore, overall server efficiency experienced a noteworthy enhancement, averaging between 5% to 7%. Concurrently, the rate of successfully servicing requests from users with priority access during peak hours surged to an impressive 83%, and it was more to 20% than achieved by the default algorithm.

REFERENCES

[1] Nedelcu, C. (2015). Nginx HTTP Server. Packt Publishing Ltd.
[2] Zakirov, V., and Abdullaev, E. (2024). Enhancing the efficiency of the remote service process. E3S Web of Conferences, 501, 02006. EDP Sciences.
[3] Abdullaev, E., Zakirov, V., and Shukurov, F. (2023). Assessment of the distance learning server's operation strategies and service capacity in advance. E3S Web of Conferences, 420, 06016. EDP Sciences.
[4] Manchanda, P. (2013). Analysis of optimization techniques to improve user response time of web applications and their implementation for MOODLE. In Advances in Information Technology: 6th International Conference, IAIT 2013, Bangkok, Thailand, December 12-13, 2013. Proceedings 6 (pp. 150-161). Springer International Publishing.
[5] Mirhosseini, A., Sriraman, A., and Wenisch, T. F. (2019). Enhancing server efficiency in the face of killer microseconds. In 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA) (pp. 185-198). IEEE.
[6] Chi, X., Liu, B., Niu, Q., and Wu, Q. (2012). Web load balance and cache optimization design based nginx under high-concurrency environment. In 2012 Third International Conference on Digital Manufacturing an Automation (pp. 1029-1032). IEEE.
[7] Nurmukhamedov, T., and Gulyamov, Z. (2024). Simulation modeling of stock management systems. In Artificial Intelligence, Blockchain, Computing and Security Volume 2 (pp. 706-711). CRC Press.
[8] Bartenev, V. (2015). Thread Pools in NGINX Boost Performance 9x. Retrieved from https://www.nginx.com/blog/thread-pools-boost-performance-9x/

[9] Wang, J., and Kai, Z. (2021). Performance analysis and optimization of nginx-based web server. Journal of Physics: Conference Series, 1955(1), 012033. IOP Publishing.
[10] Ma, C., and Chi, Y. (2022). Evaluation test and improvement of load balancing algorithms of nginx. IEEE Access, 10, 14311-14324.
[11] Kavon, A. (2020). How To Optimize Nginx Configuration. Retrieved from https://www.digitalocean.com/community/tutorials/how-to-optimize-nginx-configuration
[12] Zakirov, V. (2024). Model of Assessing the Quality Indicators of the Service Process in Transport. IETE Technical Review, 41(1), 66-72.
[13] Martinez, J., et al. (2020). Exact response time analysis of fixed priority systems based on sporadic servers. Journal of Systems Architecture, 110, 101836.
[14] Lozhkovsky, A. G. (2012). Theory of queuing in telecommunications: textbook. Odessa: ONAT named after A.S. Popova.
[15] Kornyshev, Yu. N. and Fan, G. L. (1985). Information distribution theory. Moscow: Radio and communication.
[16] M. Zukerman. (2013). Introduction to Queueing Theory and Stochastic Teletraffic Models.- City University of Hong Kong. 304p.
[17] Гнеденко Б.В., Коваленко И.Н. (1987). Введение в теорию массового обслуживания. –М.: Наука. Гл. ред. Физ.-мат. Лит. – 336с.
[18] Liu, X., Sha, L., Diao, Y., Froehlich, S., Hellerstein, J. L., & Parekh, S. (2003). Online Response Time Optimization of Apache Web Server. Quality of Service — IWQoS 2003, 461–478. https://doi.org/10.1007/3-540-44884-5_25
[19] Jugo, I., Kermek, D., Meštrović, A. (2014). Analysis and Evaluation of Web Application Performance Enhancement Techniques. Web Engineering, 40–56. https://doi.org/10.1007/978-3-319-08245-5_3
[20] Kirichek, G., Skrupsky, S., Tiahunova, M., &Timenko, A. (2020). Implementation of web system optimization method. Computer Modeling and Intelligent Systems, 2608, 199–210. https://doi.org/10.32782/cmis/2608-16
[21] NishaIlame. (2024). Enhancing Web Application Performance through Database Optimization: A Comprehensive Study. Journal of Artificial Intelligence, V. 1. – Issue 1. – pp. 1-13.
[22] Rathore, N. (2018). Performance of hybrid load balancing algorithm in distributed web server system. Wireless Personal Communications, 101, 1233-1246.
[23] Dhingra, A., &Sachdeva, M. (2020). Analyzing the Performance of Web-services during Traffic Anomalies. International Journal of Advanced Computer Science and Applications, 11(6).
[24] Mission, R. (2021). Web Service Performance Analysis using the GTMetrix Web Monitoring Tool. Journal of Innovative Technology Convergence, 3(1).
[25] Zatwarnicki, K. (2021). Providing predictable quality of service in a cloud-based web system. Applied Sciences, 11(7), 2896.
[26] Ibrahim, I. M., Ameen, S. Y., Yasin, H. M., Omar, N., Kak, S. F., Rashid, Z. N., Salih, A.A., Nareen O. M. Salim., Ahmed, D. M. (2021). Web server performance improvement using dynamic load balancing techniques: A review. system, 19, 21.
[27] Pavic, F., Brkic, L. (2021). Methods of Improving and Optimizing React Web-applications. 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO). https://doi.org/10.23919/mipro52101.2021.9596762
[28] Ahmed, W., Wu, Y., &Zheng, W. (2013). Response time based optimal web service selection. IEEE Transactions on Parallel and distributed systems, 26(2), 551-561.
[29] Mathew, A. (2019). Cybersecurity infrastructure and security automation. AdvComput: Int J (ACIJ), 10(6).
[30] Ruke, M. M., Gore, S., Chavan, S., &Dakhare, B. (2017). Maintaining data integrity for shared data in cloud. Advanced Computing: An International Journal (ACIJ), 8(1/2).

AUTHOR

VakhidZakirovwas born on September 11, 1960, in the Tashkent city of the Republic of Uzbekistan. He is Candidate of technical Sciences, in 1988 he defended his thesis for the degree of candidate of technical Sciences on the topic: "Research and development of a method for optimizing call quality indicators on telephone networks", specialty 05.12.14 - " Networks, communication centers and information distribution systems».
Associate docent, on the recommendation of the Tashkent electrotechnical Institute of communications, she was approved by the Higher attestation Commission of the Republic of Uzbekistan with the academic title of associate Professor, specialty code 05.04.01 - " Telecommunications and computer systems, networks and devices of telecommunications. Distribution of information", 06.06.1993.

EldorAbdullaevwas born on February 23, 1993, in the Tashkent region of theRepublic of Uzbekistan. He is PhD researcher of technical Sciences with Tashkent State Transport University. He defended his thesis for the master's degree in 2020.He is the author of more than thirty scientific works. Right now he is working on his PhD dissertation in the field of improving and estimating technical aspects of remote service.