STUDY OF HIGH AVAILABILITY AND PERFORMACE OFF SERVER CLUSTER

Ernestas Danilevičius, Liudvikas Kaklauskas
Šiaulių valstybinė kolegija
Lithuania

Annotation

The article analyzes selected software solutions for balancing traffic in the server cluster: Traefik, HAProxy, and NGINX. For the demonstration, a system model consists of a cluster of three servers connected to the management servers with load-balancing solutions that share a public IP address. The application servers use the same database available in a multi-master configuration and the management servers are connected via the BGP protocol. User requests reach the server cluster through redundant traffic balancing subsystems. After testing the designed system, it was found that HAProxy is the best among all selected load-balancing solutions and ensures high cluster availability. While setting up the HAProxy it is recommended to choose the dynamic Least-Connections load balancing algorithm.

Key words: *a server cluster, load-balancing, Round Robin, Least connections, HAPoxy.*

Introduction

The growing need for internet speed and efficiency is driving companies to modernize their marketing strategies. Whether a person paying his bills, an employer storing documents of his employees, or a hospital accessing a patient's health records online, they all need web applications ready for use at the press of a button. Ongoing web load studies show that under heavy loads, the availability and performance of websites are greatly reduced (Backlinko, 2018). In order to reduce the load on web applications it is necessary to increase their efficiency and speed. For this, it is best to use load-balancing solutions, that distribute traffic on the internet through several servers ensuring their constant availability (MW Team, 2023).

Load balancing is a distribution of a computer network, including internet traffic, between several servers, thus ensuring efficient server utilization (Bourke, 2001). Server groups are also called server farms (Kumar et. etc. 2023), server pools (Hindy Link et., etc, 2022), or server clusters (Yao et., etc, 2022). According to an article (Purkis, 2022) by *Liquid Web* a server cluster is a group of servers that use a single *IP* address. The utilization of server clusters and the load balancing solutions allow their loads to be distributed evenly. Load balancing increases the performance, and high availability of the web application or website for all users and provides an opportunity to expand it. The servers in the cluster are different devices, for example, a cluster of servers with two servers allows avoiding system downtime, i.e. if one of the servers stops working the other server supports the entire system.

Modern web applications that serve hundreds, thousands, or even millions of requests must always receive the correct data, this is achieved by increasing the volume of servers. Here, the load balancer acts as a railroad switch, which is located in front of the servers and splits customer requests to all active, free servers (Abdul Hameed Mohammed Farook, 2022). Load balancing solutions helps more efficiently meet increasing service requests and perform the following functions:

- Effectively distributes customer requests or network load to several servers;
- Ensures the availability and reliability of servers by sending requests only to servers that are active at the current moment;
- Flexibly adds or removes servers when demand requires it.

There are two types of load balancers, hardware and software. Hardware load balancers have been found to be more expensive, less flexible, and complexly combined. Meanwhile, software load balancers use the latest technological solutions and cloud computing. A sufficiently large number of free software load-balancing solutions can be found. They are more flexible when it comes to modernizing or expanding the system used (AVI Networks, 2023, Gandhi et., etc, 2014). An assessment of the pros and cons of hardware or software load-balancing solutions found that load-balancing provided by software solutions is superior in several aspects. Using a hardware solution would make system maintenance more expensive, also it would be necessary to place all the servers in one server module. Additionally, this solution does not allow adding physically distant servers to an existing server cluster.

After analyzing research on server cluster resiliency and performance, it was found that the availability of a cluster of two servers was evaluated (Kamilla, etc.etc., 2022), and a new load balancing resiliency algorithm in heterogeneous systems was analyzed (Yu et., etc, 2022), the fault tolerance of cloud infrastructures were evaluated (Martinez et.etc., 2022). It should be noted that the availability and performance study of a cluster of more than two servers is not done.

The purpose of the article: is to select the best server cluster load balancing management solution that guarantees sufficient performance and availability of the client's system.

1. The model of research

According to an article by *CloudFlare*, a company providing cloud computing services, on load balancers. A load balancer is software or hardware solution that protects any server from overload, the load balancing algorithm is the logic that the load balancer uses to distribute network traffic between servers (an algorithm is a set of predefined rules).

A dynamic or static load balancing algorithm can be applied to server load distribution. Static load balancing algorithms are *Round Robin*; *Weighted Round Robin*; *Randomized*, *Central Manager*, and *Threshold* algorithms. *Round Robin* algorithm distributes jobs evenly to all slave server applications. *Weighted Round Robin* allows administrators to assign different weights to different servers. The dynamic algorithm uses a transfer strategy, location strategy, and information strategy for monitoring changes in the system. This load-balancing algorithm can use three different controlling forms: centralized, distributed, or semi-distributed. Transfer strategy uses:

- *Least Connections* is a solution, when checking which servers have the least number of connections open at that time and send traffic to those servers, It is assumed that all connections require approximately the same processing power;
- *Weighted Least Connections* is a solution where administrators assign different weights to each server, assuming that some servers can process more connections than others

Information strategy is applied in the *Weighted* response time solution, where the average response time of each server and the number of open connections determine where to send data flow.

Location strategy is applied in a *Resource-based* solution, when the load is distributed according to what resources each server has at that time (Beniwal, Garg, 2014, Rajguru, Apte, 2012, Sharma et. etc., 2008).

It was decided to use dynamic traffic balancing algorithms to study the performance and availability of the server cluster. Open source load balancing solutions will be used in this cluster. Three open-source load balancers were selected based on technical articles from cloud computing and web technology solutions companies *Geekflare* and *logz.io* (Kumar, 2023, Reback, 2023): *NGINX, Seesaw, HAProxy* and *Traefik.*

In the system of a server cluster, it is planned to combine the *servers hosting* the *web application*, a simple *WordPress* website in this case, using the same internal database. For the management of the database *Multi-Master* synchronization solution was selected. This solution allows the system to maintain the same data on multiple servers. It also allows all of the connected servers to make changes to the database, unlike the *Master-Slave* configuration where no changes to the database can be done if the cluster's master server breaks. As it is recommended for this type of database cluster the system will use an odd number of servers. There are two servers containing load balancing software, both servers share the same public *IPv4* address that uses local *BGP* routing protocol. To ensure the security of this system a *Bastion* or *VPN* solution can be used to allow authorized people to connect to the servers (Fig. 1).

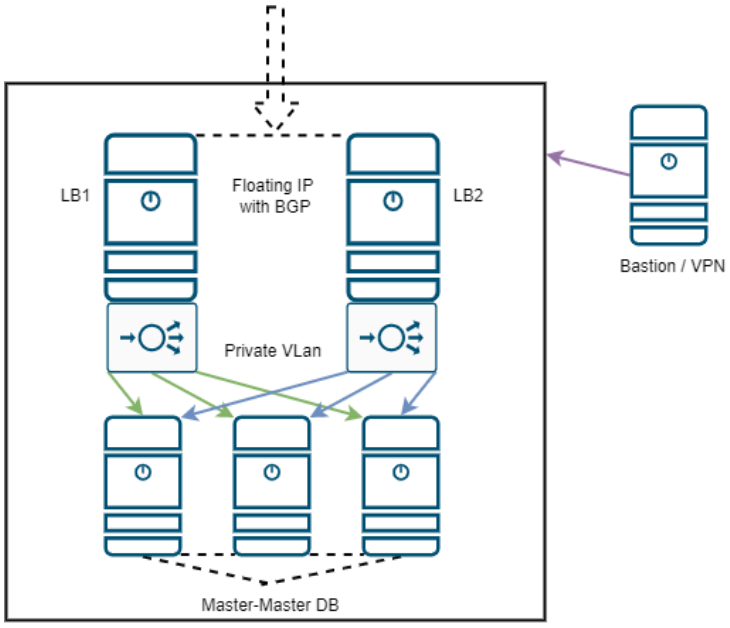

Fig. 1. Scheme of load balanced system

2. Research results discussion

For the study, a cluster containing three servers that are connected to a load-balancing subsystem is made. They each run the same *WordPress* site accessing the database connection using the multi-master solution. The operating servers in the system model are divided into two levels: controller and controlled servers. The controllers will perform the routing function, and load balancing tasks directing the traffic coming from the outside to the controlled servers. Controlled servers communicate with each other sharing the database and hosting a website.

Cloud computing solutions were used to implement the load balancing solutions: dedicated (lightweight servers that have *Intel Xeon E* processors) and virtual (VPS) servers, from the *Cherry Servers* portal (Server Deployment, 2023) selected servers with different parameters: E3-1240v3(4Core, 3.4 Ghz 16 GB RAM, NET traffic up to 1Gbps), E3-1240v5(4 Core, 3.5 GHz, 32 GB RAM, NET traffic up to 3 Gbps), E3-1240Lv5(4 Core, 2.1 Ghz, 32 GB RAM, NET traffic up to 3 Gbps), E5-1620v4(4 Core, 3.5 GHz, 32 GB RAM, NET traffic up to 3 Gbps), VPS1 (1 VCpu, 1 GB RAM, NET traffic up to 1 Gbps).

The website for the testing is accessible via a single IP address assigned to two servers containing a load-balancing solution with *BGP* enabled. This ensures the stability of the load balancing subsystem, i.e. when one server becomes unavailable, this IP address redirects traffic to another server that is available. The system communicates via *LAN IP* addresses, so it is not accessible from the outside. The administrator can only access the system through a *Bastion* server or *VPN*.

For the software implementation of load balancing solution, the following were selected: *Seesaw* (Google, 2023), *Traefik* (Traefik Labs, 2023), *HAProxy* (HAProxy, 2023), and *NGINX* (NGINX, 2023). Seesaw has been crossed out of the list due to lack of detailed documentation and errors when trying to install the system, it is not used in further testing. For *Traefik* it was selected to use a *Docker* container. Two algorithms are selected for load balancing: *Round-Robin* (static) and *Least-Connections* (dynamic). It was found that *Traefik* only supports *Round-Robin* algorithms. To test the system it was selected to use two tools capable of generating load to the *web application*: *WRK* (Apache, 2017) and *Oha* (Kornelski, 2023).

Testing was performed from a physically remote server. The speed of the server's response time to requests and the number of requests sent by the testing tools were monitored. And the traffic is split by the balancing algorithm used with the average load of the servers, which is directly related to the number of connections set on testing tools and response time to the server that is running the test script. The open-source IT infrastructure monitoring tool *Zabbix* (Zabbix 2023) was chosen for server load monitoring. A script has been created for test automation that starts testing with specified arguments and testing tools. During testing the number of requests per second and the average response time to the requests were monitored. Testing of the system took two hours per tool with a fifteen-minute break between the tests to monitor the backend server load during testing hours and during the breaks between them. All of the tested servers had the same load-balancing software installed and were in the queue for their turn for testing (Fig. 2).

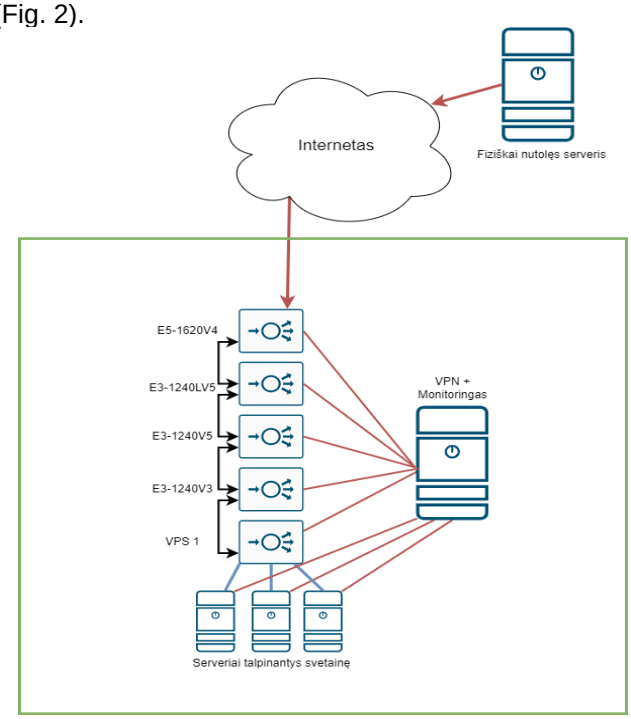

Fig. 2. Model for system testing

After testing all results were logged and entered into the tables of content from which charts were made to compare accepted requests per second and average response time to a request (Fig. 3), (Fig. 4).

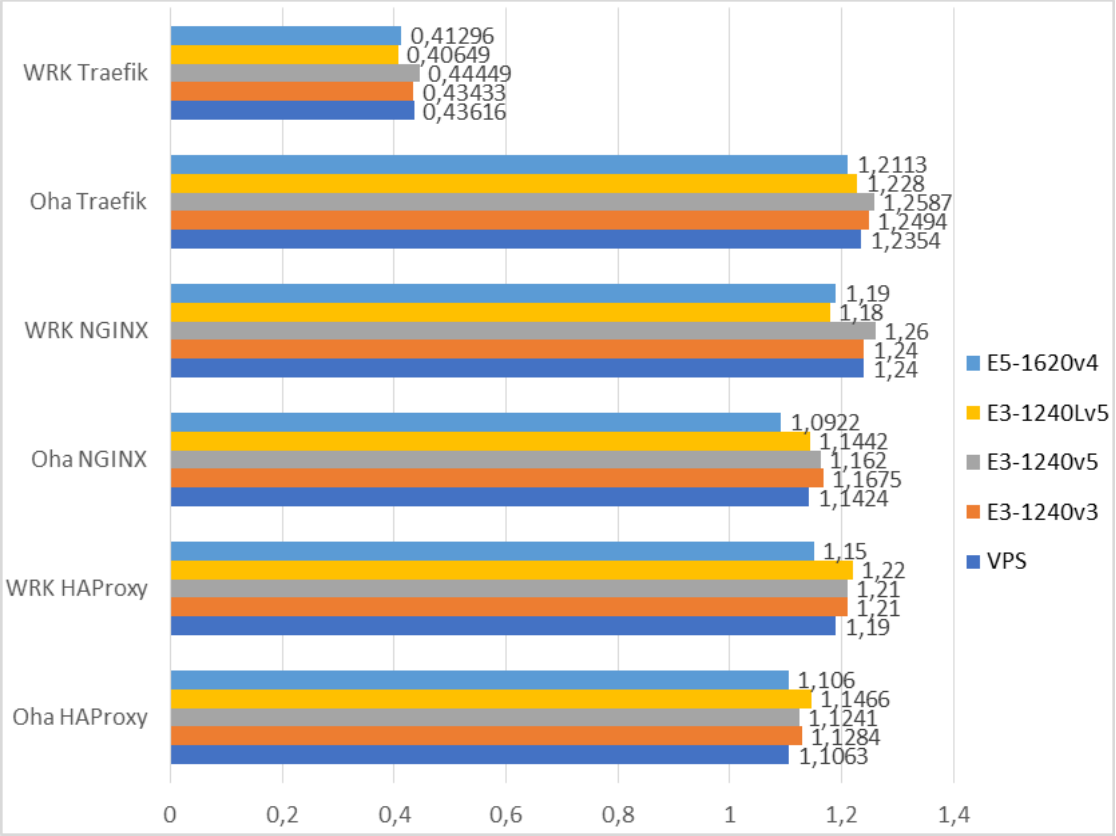

Fig. 3. Average response time

In the chart representing the average response time of the cluster (Fig.3), it is clearly noticeable, that *Traefik* had the lowest response times on all of the selected servers. But by monitoring the system during testing it was noticed that all of the load generated by testing tools ends up in only one of three servers from the cluster. In order to keep the cluster's database safe it was decided to lower the load-generating tool's *WRK* active connections three times compared to all other tests. After analyzing the response time chart it was decided that *HAProxy* has slightly faster response times on the majority of the tested servers.

Taking a look at the chart of average requests per second accepted by the cluster (Fig. 4), it was noticed that *HAProxy* and *NGINX* are pretty close by the number of requests accepted. After more in-depth analysis, it was decided that *HAProxy* has a better performance on *VPS* servers, which is a more affordable option.

Analyzing the results of the study, it was found that the load-balancing system *Traefik* does not evenly distribute incoming traffic to the server cluster, usually; one server of the system is heavily loaded while others are almost in an idle state. A dynamic Least-connections load-balancing algorithm was used for *NGINX* and *HAProxy* testing. The analysis of the research results showed that the *HAProxy* system that uses the least-connections algorithm is more efficient in distributing load evenly to identical application servers; it guarantees higher availability and performance of the tested system. The Round-Robin load balancing algorithm is worse at processing and distributing requests under heavy loads.

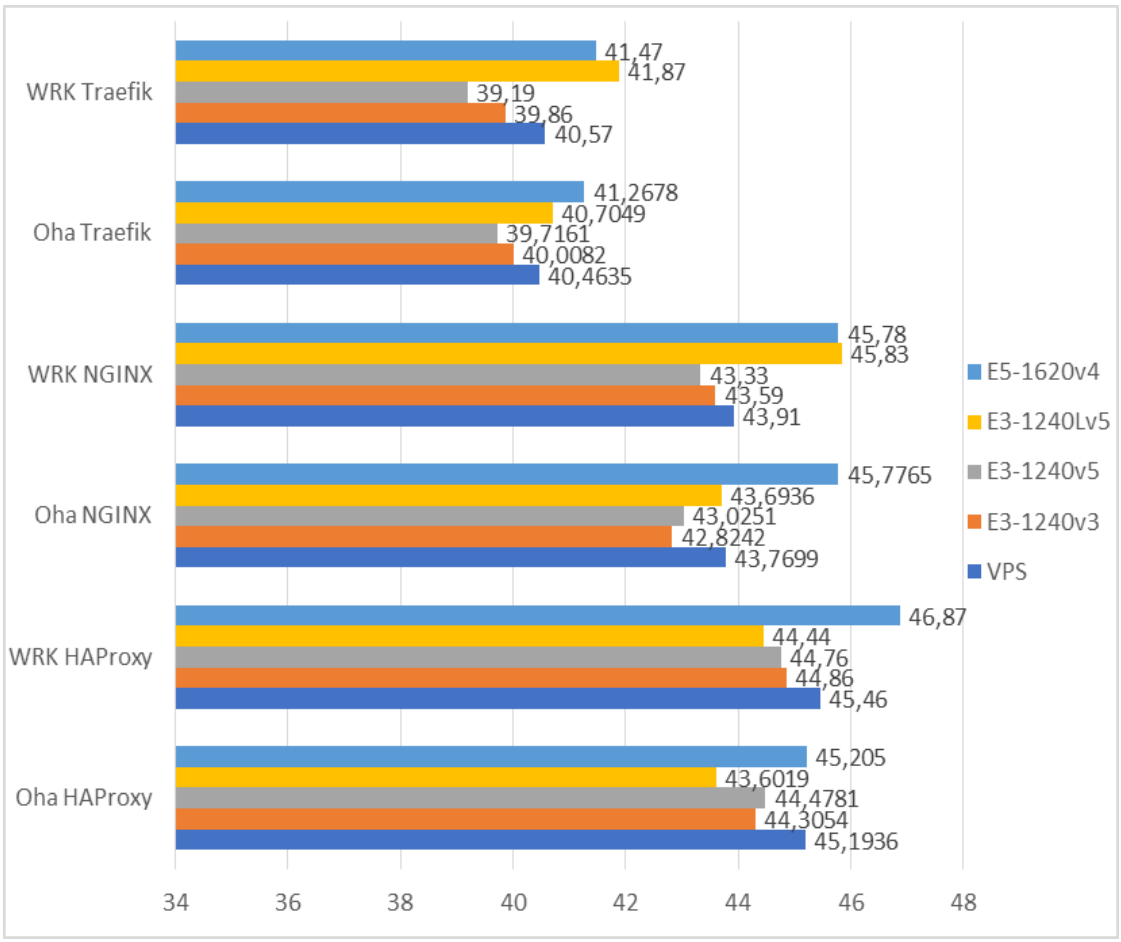

Fig. 4. Requests per second

3. Conclusions

1. Based on the analysis of load balancers, it was found that *HAProxy* is the best at distributing load to the cluster of servers and ensures their constant availability.

2. When configuring *HAProxy* with a cluster of identical servers it is recommended to choose the dynamic *Least-Connections* algorithm.

References

1. MW Team. Why your business application needs a Load Balancer. MiddleWare. [Tinkle] 2021 m. 10 25 d. [Cituota: 2023 m. 05 05 d.] https://middleware.io/blog/why-business-application-needs-load-balancer/.

2. Backlinko. Here's what we learned about page speed. [Tinkle] 2018 m. 10 08 d. [Cituota: 2023 m. 05 05 d.] https://backlinko.com/page-speed-stats.

3. Gundu, S. R., Panem, C. A., & Thimmapuram, A. (2020). Real-time cloud-based load balance algorithms and an analysis. SN Computer Science, 1, 1-9.

4. Ghomi, E. J., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: A survey. Journal of Network and Computer Applications, 88, 50-71.

5. Katyal, M., & Mishra, A. (2014). A comparative study of load balancing algorithms in cloud computing environment. arXiv preprint arXiv:1403.6918.

6. Bourke, T. (2001). Server load balancing. " O'Reilly Media, Inc.".

7. Kumar, P., Vinodh Kumar, S., & Priya, L. (2023, January). An Approach for Energy-Efficient Resource Allocation Through Early Planning of Virtual Machines to Servers in Cloud Server Farms. In Machine Learning, Image Processing, Network Security and Data Sciences: Select Proceedings of 3rd International Conference on MIND 2021 (pp. 725-737). Singapore: Springer Nature Singapore.

8. Hindy Ling, H., & Bar-Gera, H. Server Pooling Models for Separate and Bounded Queues. Hillel, Server Pooling Models for Separate and Bounded Queues.

9. Yao, J., & Yuan, Z. (2022, December). Research on Performance Optimization of Virtualized Server Cluster Based on Cloud Computing. In 2022 International Symposium on Advances in Informatics, Electronics and Education (ISAIEE) (pp. 387-391). IEEE.

10. Abdul Hameed Mohammed Farook, S. A. (2022). Enhance Microservices Placement by Using Workload Profiling Across Multiple Container Clusters (Doctoral dissertation, Dublin, National College of Ireland).

11. Purkis, M. (2022) How does a clustered server environment help businesses save money? liquidweb. [Tinkle] 2022 m. 10 17 d. [Cituota: 2023 m. 01 10 d.] https://www.liquidweb.com/blog/what-is-server-cluster/.

12. AVI Networks (2023). Load Balancing Guide. [Tinkle] [Cituota: 2023 m. 05 05 d.] https://avinetworks.com/what-is-load-balancing/.

13. Gandhi, R., Liu, H. H., Hu, Y. C., Lu, G., Padhye, J., Yuan, L., & Zhang, M. (2014). Duet: Cloud scale load balancing with hardware and software. ACM SIGCOMM Computer Communication Review, 44(4), 27-38.

14. Kamila, N. K., Frnda, J., Pani, S. K., Das, R., Islam, S. M., Bharti, P. K., & Muduli, K. (2022). Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach. Journal of King Saud University-Computer and Information Sciences, 34(10), 9991-10009.

15. Yu, H., Wang, X., Xing, C., & Xu, B. (2022). A Microservice Resilience Deployment Mechanism Based on Diversity. Security and Communication Networks, 2022.

16. Martinez, H. F., Mondragon, O. H., Rubio, H. A., & Marquez, J. (2022). Computational and Communication Infrastructure Challenges for Resilient Cloud Services. Computers, 11(8), 118.

17. Beniwal, P., & Garg, A. (2014). A comparative study of static and dynamic load balancing algorithms. International journal of advance research in computer science and management studies, 2(12), 1-7.

18. Sharma, S., Singh, S., & Sharma, M. (2008). Performance analysis of load balancing algorithms. International Journal of Civil and Environmental Engineering, 2(2), 367-370.

19. Rajguru, A. A., & Apte, S. S. (2012). A comparative performance analysis of load balancing algorithms in distributed system using qualitative parameters. International Journal of Recent Technology and Engineering, 1(3), 175-179.

20. Kumar, C. (2023) 10 Open Source Load Balancer for HA and Improved Performance. GeekFlare. [Tinkle] 2022 m. 09 06 d. [Cituota: 2023 m. 01 06 d.] https://geekflare.com/open-source-load-balancer/.

21. Redback, G. (2023) The 5 Best Open Source Load Balancers. logz.io. [Tinkle] [Cituota: 2023 m. 01 06 d.] https://logz.io/blog/best-open-source-load-balancers/.

22. Server Deployment. [Tinkle] Cherry Servers. [Cituota: 2023 m. 05 05 d.] https://portal.cherryservers.com/#/deployment.

23. Google (2023). Seesaw v2. [Tinkle] Google Seesaw. [Cituota: 2023 m. 05 05 d.] https://github.com/google/seesaw.

24. Traefic Labs (2023). Traefik & Docker. [Tinkle] Traefik & Docker. [Cituota: 2023 m. 05 05 d.] https://doc.traefik.io/traefik/routing/providers/docker/.

25. Apache (2017). Wrk – a HTTP benchmarking tool. [Tinkle] Wrk – a HTTP benchmarking tool. [Cituota: 2023 m. 05 05 d.] https://github.com/wg/wrk.

26. Kornelski (2023). Oha [Tinkle] Oha. [Cituota: 2023 m. 05 05 d.] https://lib.rs/crates/oha.

27. Zabix LLC (2023). Zabbix [Tinkle] Zabix. [Cituota: 2023 m. 05 05 d.] https://www.zabbix.com/.

28. HAProxy (2023). HAProxy [Tinkle] HAProxy. [Cituota: 2023 m. 05 05 d.] https://www.haproxy.org/

29. NGiNX (2023). Using nginx as HTTP load balancer [Tinkle] NGiNX. [Cituota: 2023 m. 05 05 d.] http://nginx.org/

Received: 29 April 2023.
Accepted: 16 May 2023.