ISSN 2707-1898 (print)
ISSN 2786-9172 (online)

Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2025.02.138
Article received 16.08.2025 р.
Article accepted 30.10.2025 р.
UDC 004.9

Correspondence author
@ O.Y. Pitsun
o.pitsun@wunu.edu.ua

O. Y. Pitsun
West Ukrainian National University, Ternopil, Ukraine

ARCHITECTURE FOR HIGH-LOAD WEB RESOURCES OPTIMIZATION

The large amount of data used on web resources contributes to their slowdown, which negatively affects the loading time and the overall impression of the work. Caching servers, which temporarily store frequently requested data closer to the user, can significantly reduce the response time of servers, reduce the load on the primary computing resources, and increase the stability of web applications. Their implementation becomes especially relevant in the case of highly loaded web services. Modern web pages often take a long time to load due to a combination of technical factors, such as the volume of data and server load. In this work, the architecture of the server part of the web resource for processing big data with high load elements has been developed to speed up the work of web resources. A load balancer and caching servers have been chosen to speed up the work. One of the key factors determining the speed of modern web resources is the effective use of caching mechanisms. Caching servers allow you to store intermediate calculation results, static files, and pre-generated pages, which significantly reduces the system's response time to repeated requests. The lack of a cache means that each request is processed "from scratch", even if the data hasn't changed. This increases the risk of overload, especially during peak traffic or DDoS-like spikes in activity. Powerful computing nodes waste resources regenerating the same responses, which increases infrastructure costs. Effective caching often requires separate servers or cloud services like Redis, Varnish, CDN. This increases hardware costs or the need to rent additional resources from cloud providers. The balancer prevents overloading of individual servers by distributing requests between them. This ensures stable operation even during peak hours of user activity and reduces the risk of failures. In the event of a server failure, the balancer can automatically redirect traffic to other working nodes. This reduces the likelihood of complete system downtime and increases service availability. The proposed architecture is adapted to the development of web resources with elements of Unet networks, which are characterized by the presence of a large amount of static content. Comparative analysis demonstrates that web page loading time decreased by an average of 3 times using caching servers.

Keywords: web server, Unet, caching server, Varnish, Redis.

Introduction

Increasingly, when developing web applications, developers and users pay attention to the speed and performance of sites, as the number of mobile devices has increased dramatically, and the emphasis should be on adaptability and page loading speed on such devices. Modern systems use servers not only to ensure the operation of websites, but also to ensure the operation of mobile applications and API nodes. This significantly increases the load on the system and causes time delays in the server's response to the client. Another factor that affects the need to improve site performance is the impact of response time on SEO optimization and site conversion. Optimization of website performance in such conditions becomes a key factor of competitiveness, as it directly affects the user experience and business results. Currently, website performance analysis is being actively implemented based on metrics that relate to both the server and client sides.

The object of the study is high-load web servers for processing a large flow of data and requests.

The subject of the study is approaches to optimizing the load on servers in the form of caching servers, which allows reducing the load on the server part.

The purpose of the study is to analyze modern means of optimizing the operation of websites at different levels, which will allow identifying key aspects in the development of web applications based on machine learning. Based on the analysis, a proposed architecture for the server part of the web resource for processing big data with elements of neural networks aims to speed up web resource operations.

To achieve the stated purpose, the following main research tasks were identified:

- Conduct a comparative analysis of tools for optimizing the front-end part of websites.
- Conduct an analysis of tools for optimizing the back-end part of websites, which will allow us to identify the main elements that can improve the performance of websites.
- Develop the architecture of a software module consisting of a load balancer and caching servers, which allows us to speed up the work of highly loaded websites.

Analysis of Recent Research and Publications. The dynamic and interactive behavior of users of modern web applications, justified by the development of social networks, e-commerce, and artificial intelligence, significantly complicates the modeling of web loads. Experimental results show that taking into account such a situation can increase processor usage by up to 30 % and significantly worsen the average response time of the system [1]. In the study [2], the behavior of web servers under conditions of a large number of users was analyzed with a focus on modeling the marginal values of the response time. Based on the conclusions obtained, the implementation of technical and organizational measures (load balancing, monitoring, configuration settings) to increase the reliability and stability of web servers was recommended. In the work [3], an evaluation of the open edge platform EdgeX for use in IoT services was conducted. The results confirm the prospects of EdgeX for developing solutions in the field of edge computing. In [4], a new approach to analyzing web server log files by transforming them into horizontal visibility graphs for further application of complex network analysis methods was proposed. In article [5], an empirical study of Apache web server protection against DoS attacks was conducted by evaluating two well-known connected modules. Experiments involving both flooding and slow attacks showed that there is no universal solution, although the existing modules can be helpful in practice.

In [6], a variable feedback strategy for load balancing in a web mapping services platform (WMSP) is proposed, which takes into account the temporal patterns of user access intensity. Experimental results confirmed that the approach provides fast response, high throughput, and stable operation.

In [7], proposed a new method for load balancing in cloud systems, which manages the dynamic distribution of work flows among servers. The approach modifies the existing balancing technique. In [8], an approach is presented in which the algorithm parameter settings are achieved to provide more accurate detection of high load, and the proposed optimization reduces the number of page faults and improves performance by 7.36 %. In [9], an automatic strategy for optimizing the performance of front-end frameworks based on the Bayesian optimization algorithm is proposed. In [10], the authors investigate the problem of load balancing in microservice architectures, which affects the sestems efficiency and reliability . Based on Spring Cloud and Alibaba components, a dynamic load balancing algorithm is proposed that considers Redis for caching. To improve the load balancing efficiency in conditions of dynamic changes, the authors in [11] propose an algorithm based on long-term forecasting using Informer, which predicts the future use of CPU, memory, network, and I/O and adjusts the weights of servers in advance. In [12], a spatial version of the Power of Two method is proposed, which assigns a task to the least loaded of the two geographically closest servers. In [13, 14], approaches to designing a microservice architecture are considered, in particular for systems based on artificial intelligence elements, which involve processing large data sets.

Analyzing the above data on the current state of research, we can conclude that the topic is relevant and essential for web resources with high load. The authors draw attention to the importance of optimizing the server component when processing large amounts of data and server requests.

Materials and Methods. This study uses an analytical approach to analyze existing techniques and tools for optimizing the server side of web applications, which allows us to highlight relevant tools and their advantages and disadvantages. Approaches to developing the architecture of the server-side of web applications based on caching servers are used, which allows us to optimize the load.

Research results and their discussion

Optimization of the client-side of web resources

A list of tools and resources that allow you to improve performance at the frontend level is shown in Fig. 1.

Table 1. Comparative analysis of means

Optimization method	Tools	metrics
Responsive Design	Bootstrap, Tailwind CSS, CSS Grid/Flexbox	LCP, CLS, Bounce Rate
Minimization CSS/JS	UglifyJS, Terser, CSSNano	TTFB, LCP, FCP
image formats (WebP, AVIF)	Squoosh, ImageMagick, TinyPNG	LCP, Page Size
Lazy Loading	Intersection Observer API	LCP, Speed Index

Therefore Client-side optimization techniques aim to reduce page load time and improve user experience by adapting content, minimizing resources, and efficiently loading images and scripts.

Optimization of the server-side of web resources

Server-level optimization methods involve the use of various approaches to improve the performance and stability of websites. Table 2 provides a comparative analysis of server-level optimization tools for web resources.

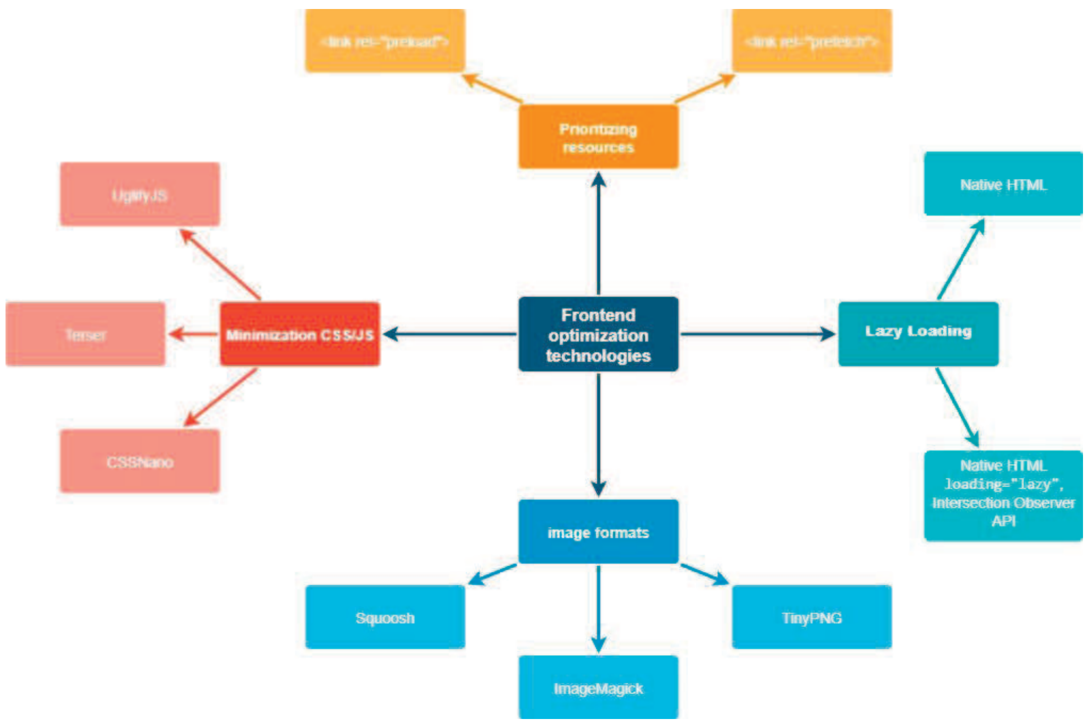


Fig. 1. Tools for improving performance at the frontend level

Table 2. Comparative analysis of server-side optimization tools for web resources

Optimization method	Tools	metrics
Load Balancing	Nginx, HAProxy, AWS ELB, Kubernetes Ingress	Response time, Throughput, Server utilization, Downtime
Server-side caching	Varnish Redis Memcached	TTFB, LCP, Server load, Requests / sec
Data compression	Nginx Apache modules	Page size, Time to First Byte, Bandwidth usage
Event-driven servers	Node.js Nginx, LiteSpeed	Concurrency, Response time, Throughput
Database replication and sharding	MySQL Replicaion, MongoDB Sharding	DB query latency, Throughput, Availability

Load balancing is the distribution of requests between multiple servers, which allows for even resource utilization and high system availability.

Server caching involves storing query processing results or static content for reuse. This reduces server response time and optimizes overall computing resource utilization. Fig. 2 shows a classification of servers.

Caching servers can be classified by type of caching and purpose. A Reverse Proxy stores ready-made HTTP responses and serves client requests without contacting backend servers. An example of such technology is Varnish.

In-memory caching stores data in memory for quick access, reducing latency when working with databases. Examples include Redis and Memcached.

Web server-level caching, such as Nginx FastCGI cache, stores the processed results of application requests and allows re-serving requests without re-generating content.

Proposed architecture.

Figure 3 shows an example of the proposed architecture for a web resource processing system based on caching services. The proposed system contains elements of both a web resource and machine learning elements, in particular, Unet segmentation.

The algorithm of the proposed module and architecture is as follows:

1. System users load web pages written using Twitter Bootstrap technology to ensure adaptability and Vue.js for asynchrony.
2. After the site is loaded, a load balancer is activated, which distributes tasks between several environments combined by orchestration and the use of Docker.
3. At the next stage, traffic is redirected to the Varnish server, which acts not only as a proxy server, but also as a server caching static content.
4. At the next stage, the Apache web server is loaded.
5. The PHP programming language and the Laravel framework are used as the backend part. Data is stored in the MySQL database, and the Redis cache server is used.
6. A feature of the proposed architecture is the use of a block for implementing semantic segmentation using Unet networks using TensorFlow and Keras technologies.
7. An additional web interface has been developed for engineers to work with, which works directly with the web server and the backend part based on PHP, Laravel. This is done so that engineers can implement automatic segmentation without the risk of caching individual page blocks.

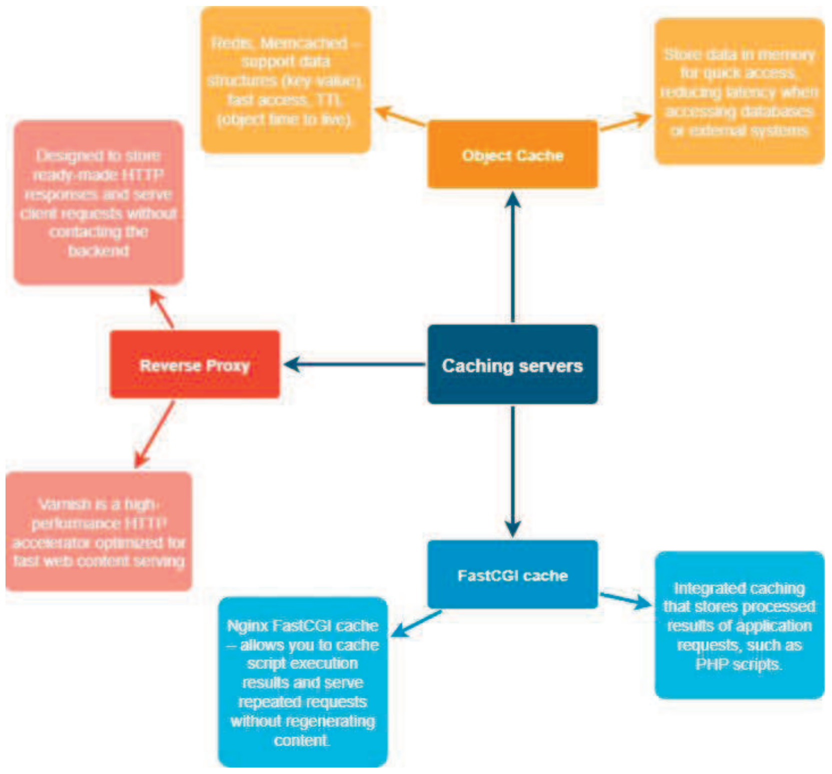

Redis, Memcached – support data structures (key-value), fast access, TTL (object time to live).
Store data in memory for quick access, reducing latency when accessing databases or external systems
Object Cache
Designed to store ready-made HTTP responses and serve client requests without contacting the backend
Reverse Proxy
Caching servers
Varnish is a high-performance HTTP accelerator optimized for fast web content serving
FastCGI cache
Nginx FastCGI cache – allows you to cache script execution results and serve repeated requests without regenerating content.
Integrated caching that stores processed results of application requests, such as PHP scripts.

Fig. 2. Classification of caching servers

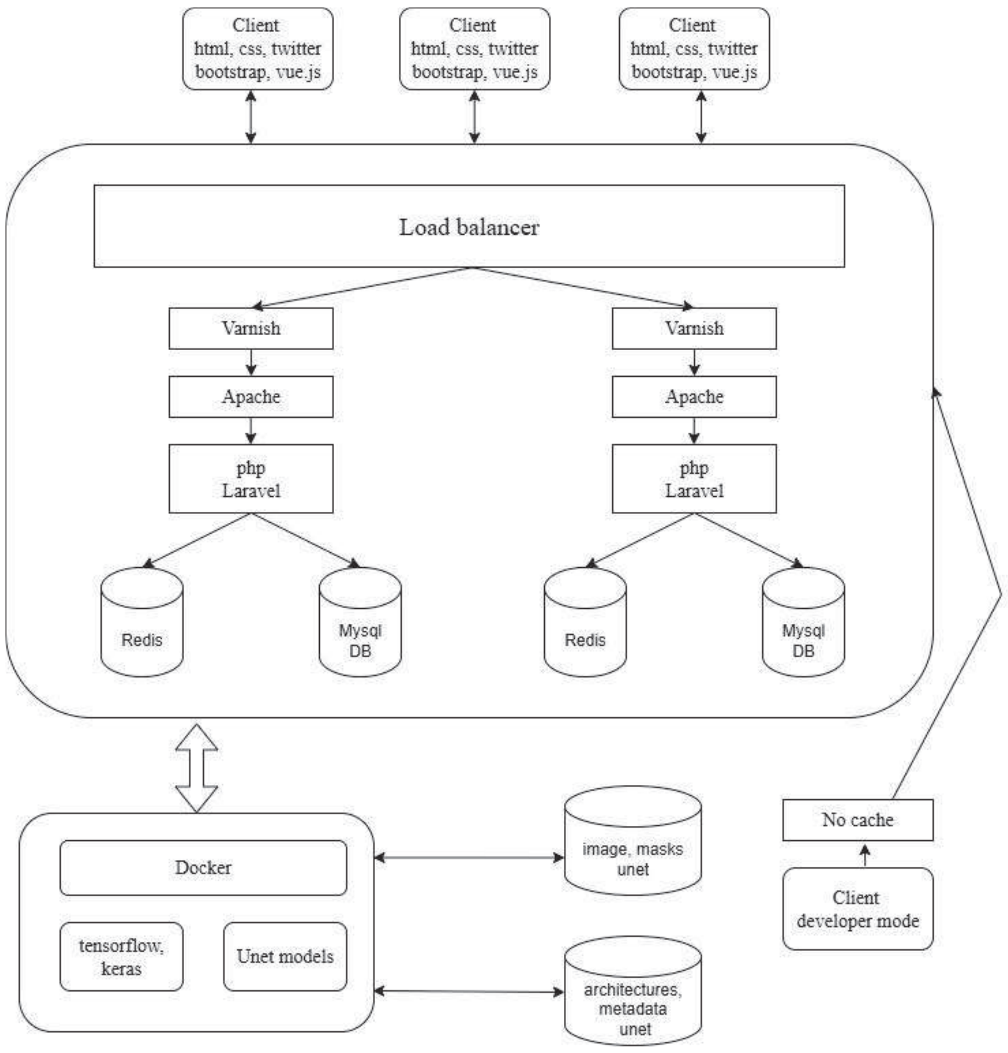


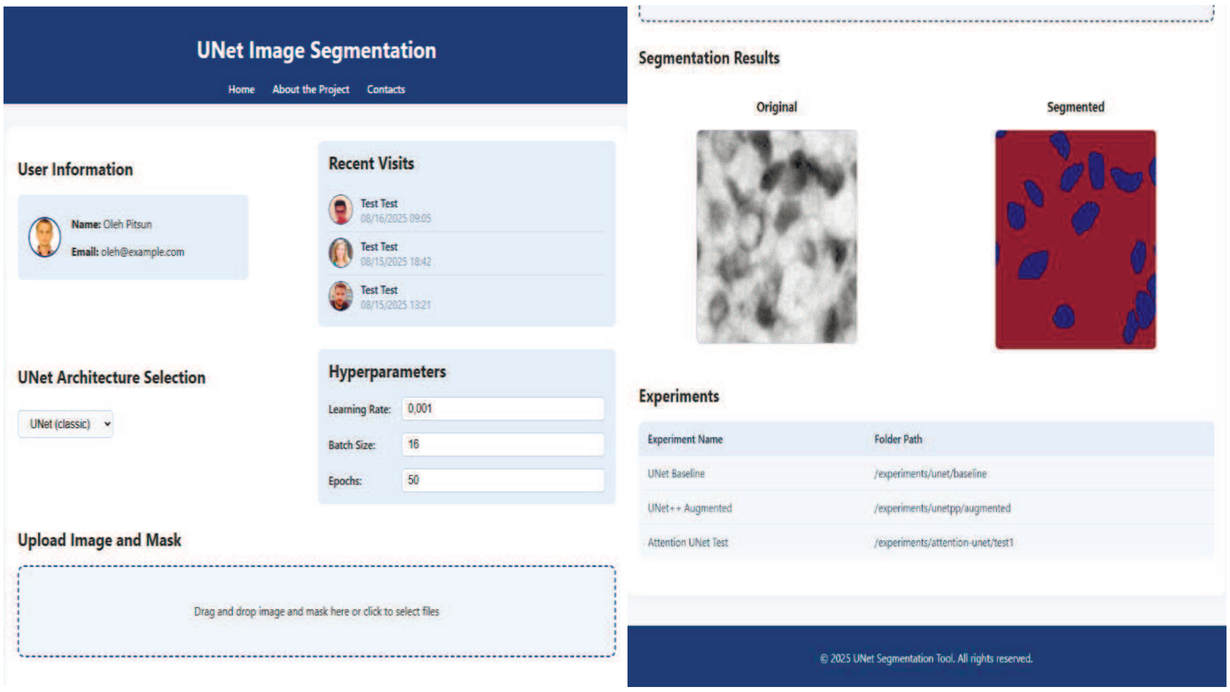
Fig. 3. Proposed architecture

The web interface of the developed module is shown in Fig. 4.

The webpage consists of the following blocks:

1. User information block, with photo and contact information.
2. Information about the latest site visitors.
3. Ability to choose Unet architecture.
4. Ability to choose hyperparameters.
5. New image loading block.
6. Image output block (original and segmented)
7. Table for viewing previous experiments.

The results of the experiments are shown in Fig. 5.



UNet Image Segmentation

Home About the Project Contacts

User Information

Name: Oleh Pitsun

Email: oleh@example.com

Recent Visits

Test Test 08/16/2025 09:05

Test Test 08/15/2025 18:42

Test Test 08/15/2025 13:21

UNet Architecture Selection

UNet (classic)

Hyperparameters

Learning Rate: 0,001

Batch Size: 16

Epochs: 50

Upload Image and Mask

Drag and drop image and mask here or click to select files

Segmentation Results

Original

Segmented

Experiments

Experiment Name	Folder Path
UNet Baseline	/experiments/unet/baseline
UNet++ Augmented	/experiments/unetpp/augmented
Attention UNet Test	/experiments/attention-unet/test1

© 2025 UNet Segmentation Tool. All rights reserved.

Fig. 4. Web interface of the developed module

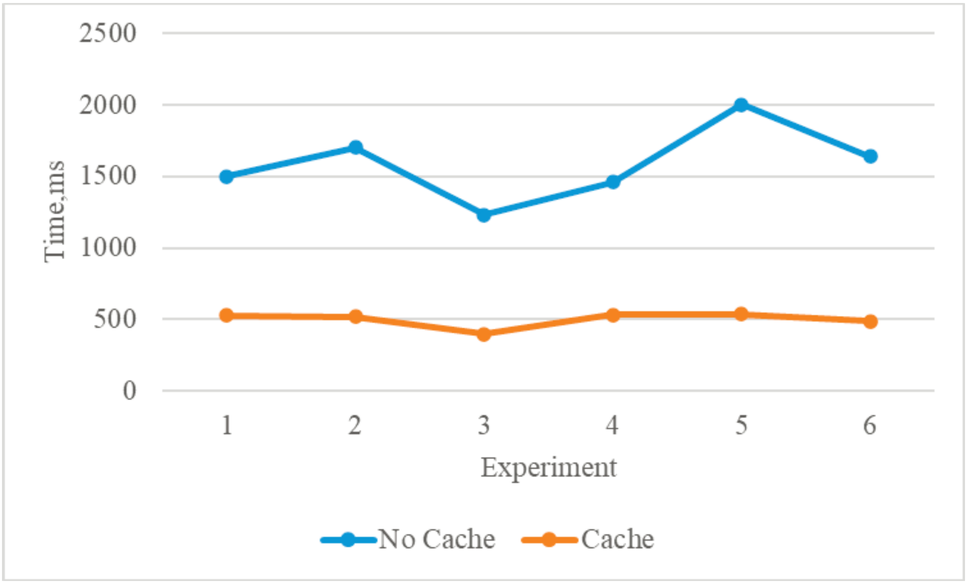


Fig. 5. Results of page loading experiments with and without caching

Based on the result obtained, we can conclude that the loading time of pages with cache is 3 times better.

Discussion of the Research Results. The authors of [8] investigate the problem of high server memory load, which leads to frequent page faults and reduced performance. The paper proposes a performance detection and optimization method for collecting detailed metrics from Linux and the Isolation Forest algorithm for anomaly detection.

In [9], an automatic performance optimization strategy for front-end frameworks based on the Bayesian optimization algorithm is proposed. After optimization, the key indicators are significantly reduced: LCP – up to 2052 ms, TTI – up to 2923 ms, CPU load – up to 65.9 %, memory consumption – up to 324.4 MB. The research data are relevant, but they do not sufficiently offer approaches to server optimization, the use of caching servers, and proxy servers.

In [10], a dynamic load balancing algorithm is proposed that takes into account the state of computing nodes and uses Redis for caching. Thus, the researchers confirm the relevance of using caching servers for load optimization.

In [11], a load balancing algorithm for microservice systems is proposed, which uses Informer for long-term forecasting of CPU, memory, network resources, and I/O usage, enabling the adjustment of server weights in advance.

In [12], the problem of static load distribution in distributed systems is considered, taking into account the implementation cost. sPOT is proposed, a spatial version of the Power of Two method, which assigns a task to the least loaded of the two geographically closest servers.

An analysis of previous studies demonstrates the relevance of developing approaches to speeding up the work of web resources, which are designed to process modules with artificial intelligence elements. Unlike prior studies, this work uses an approach that employs a load balancer and caching servers.

Scientific novelty of the obtained research results – the architecture of the server part of a web resource for processing big data with elements of neural networks to speed up the work of web resources has been developed.

Practical significance of the research results – lies in the development of a software module for processing Unet networks with caching elements to speed up the operation of web resources.

Conclusions

This paper proposes an architecture of the server part of a web resource for processing big data with elements of a load balancer and caching servers.

1. Based on an analytical approach, an analysis of means of optimizing the front-end part of web resources was carried out.

2. Based on an analytical approach, an analysis of modern techniques for optimizing web resources was carried out, which allowed us to identify the main technologies and types of caching servers.

3. An architecture of the server part of a web resource with elements of caching servers was developed, which allowed us to speed up the loading of web pages by an average of 3 times.

References

1. Peña-Ortiz, R., Gil, J. A., Sahuquillo, J., & Pont, A. (2013). Analyzing web server performance under dynamic user workloads. *Computer communications*, 36(4), 386–395. https://doi.org/10.1016/j.comcom.2012.11.005
2. Rafiu, H. A., Adesina, O. S., & Adekeye, K. S. (2024). Modelling the response rate of the Apache web server using extreme value theory. *Scientific African*, 23, https://doi.org/10.1016/j.sciaf. 2024.e02086
3. Dhulfiqar, A., Abdala, M. A., Pataki, N., & Tejfel, M. (2024). Deploying a web service application on the EdgeX open edge server: An evaluation of its viability for IoT services. *Procedia Computer Science*, 235, 852–862. https://doi.org/10.1016/ j.procs.2024.04.081
4. Sulaimany, S., & Mafakheri, A. (2023). Visibility graph analysis of web server log files. *Physica A: Statistical Mechanics and its Applications*, 611, https://doi.org/10.1016/ j.physa.2023.128448
5. Catillo, M., Pecchia, A., & Villano, U. (2022). No more DoS? An empirical study on defense techniques for web server Denial of Service mitigation. *Journal of Network and Computer Applications*, 202, https://doi.org/10.1016/j.jnca.2022.103363
6. Li, R., Dong, G., Jiang, J., Wu, H., Yang, N., & Chen, W. (2019). Self-adaptive load-balancing strategy based on a time series pattern for concurrent user access on the Web map service. *Computers & Geosciences*, 131, 60–69. https://doi.org/10.1016/ j.cageo.2019.06.015
7. Kumar, V. D., Praveenchandar, J., Arif, M., Brezulianu, A., Geman, O., & Ikram, A. (2023). Efficient Cloud Resource Scheduling with an Optimized Throttled Load Balancing Approach. *Computers, Materials & Continua*, 77(2). https://doi.org/10.32604/cmc.2023.034764
8. Xu, D., Gao, Y., & Chen, L. (2024). Research on Server Memory High Load Performance Optimization Method Based on eBPF and Isolation Forest Algorithm. In Proceedings of the 2024 7th International Conference on Artificial Intelligence and Pattern Recognition, 792–797. https://doi.org/10.1145/3703935.3704086
9. Xu, H. (2024). In-memory database load balancing optimization for massive information processing of the Internet of Things. ACM Transactions on Asian and Low-Resource Language Information Processing. https://doi.org/10.1145/3670996
10. Chen, J., Fan, R., Shao, C., Hu, Z., Zhu, S., Li, X., ... & Zhang, J. (2024). Optimisation Strategies for Load Balancing Algorithms Based on Spring Cloud Alibaba. In Proceedings of the 2024 3rd Asia Conference on Algorithms, *Computing and Machine Learning*, 207–211. https://doi.org/10.1145/ 3654823.3654861
11. Wu, Y., & Chen, S. (2024). Optimization of load balancing algorithm based on Informer long time series prediction. In Proceedings of the 2024 8th International Conference on Electronic Information Technology and Computer Engineering, 985–991. https://doi.org/10.1145/3711129.3711297
12. Піцун, О., Пришляк, К., Каліновський, Р., & Поворозник, В. (2023). Мікросервісна архітектура системи опрацювання імуногістохімічних зображенЬ. Herald of Khmelnytskyi National University. *Technical sciences*, 321(3), 166–174. https://www.doi.org/10.31891/2307-5732-2023-321-3-166-174

О. Й. Піцун

Західноукраїнський національний університет, м. Тернопіль, Україна

АРХІТЕКТУРА ДЛЯ ОПТИМІЗАЦІЇ ВЕБРЕСУРСІВ ВИСОКИХ НАВАНТАЖЕНЬ

Великий об'єм даних, використовуваних на вебресурсах, сприяє уповільненню їх роботи, що негативно впливає на тривалість завантаження та загальне враження від роботи. Одним із найефективніших підходів до оптимізації продуктивності є застосування кешувальних серверів, які дають можливість тимчасово зберігати часто запитувані дані ближче до кінцевого користувача. Це дає змогу істотно скоротити час відповіді сервера, знизити навантаження на центральну інфраструктуру та забезпечити стабільність і безперервність роботи вебресурсу. Крім того, використання систем балансування навантаження дозволяє рівномірно розподіляти запити між кількома серверами, що додатково підвищує відмовостійкість та масштабованість системи. Кешувальні сервери, що тимчасово зберігають часто запитувані дані ближче до користувача, дають змогу істотно зменшити час відгуку серверів, знизити навантаження на основні обчислювальні ресурси та підвищити стабільність роботи вебдодатків. Їхнє впровадження стає особливо актуальним у випадку високонавантажених вебсервісів. Сучасні вебсторінки часто довго завантажуються через комплекс технічних факторів, таких як об'єм даних, завантаженість серверів. Одним із ключових факторів, що визначають швидкодію сучасних вебресурсів, є ефективне використання механізмів кешування. Кешувальні сервери дають змогу зберігати проміжні результати обчислень, статичні файли та попередньо сформовані сторінки, що істотно зменшує тривалість відповіді системи на повторні запити. У статті розроблено архітектуру серверної частини вебресурсу опрацювання великих даних з елементами високих навантажень для пришвидшення роботи вебресурсів. Як засоби пришвидшення роботи вибрано балансувальник навантажень та кешувальні сервери. Запропонована архітектура адаптована до розроблення вебресурсів з елементами Unet мереж, які характеризуються наявністю великої кількості статичного контенту. Порівняльний аналіз показав, що час завантаження вебсторінки зменшився в середньому утричі з використанням кешувальних серверів.

Ключові слова: вебсервер, Unet, кешуючий сервер, varnish, redis.

Інформація про авторів:

Піцун Олег Йосипович, канд. техн. наук, доцент, кафедра комп'ютерної інженерії.

Email: o.pitsun@wunu.edu.ua; https://orcid.org/0000-0000-0000-0000

Цитування за ДСТУ: Піцун О. Й. Архітектура для оптимізації вебресурсів високих навантажень. *Український журнал інформаційних технологій*. 2025, т. 7, № 2. С. 138–144.

Citation APA: Pitsun, O. Y. (2025). Architecture for high-load web resources optimization. *Ukrainian Journal of Information Technology*, 7(2), 138–144. https://doi.org/10.23939/ujit2025.02.138