A high-load architecture for image processing based on microservices

Oleh Pitsun[*,†] and Myroslav Shymchuk[†]

West Ukrainian National University, 11 Lvivska st., Ternopil, 46001, Ukraine

Abstract

The use of artificial intelligence tools for image analysis and research is a relevant and important task, in particular in medicine. The disadvantage of existing systems is their "heaviness", since the use of artificial intelligence requires significant hardware and software resources. The availability of a large number of libraries and frameworks for data processing and machine learning requires a large amount of computation. In this case, the use of cloud technologies facilitates the adaptive use of many technologies to solve various tasks. The development of high-performance architectures is a pressing task, especially in the field of image processing. The use of monolithic architecture is an insufficient approach and requires new solutions, in particular microservice architecture, which allows improving the performance and reliability of the system. In this paper, a microservice architecture is proposed for image processing with elements of high system loads. Comparative analysis demonstrates the presence of a larger amount of functionality for processing highly loaded systems.

Keywords

CNN, high-load architecture, images, microservises

1. Introduction

In today's rapidly growing data volume and the popularity of intelligent systems, the development of architectures for high-load software solutions is becoming particularly relevant. This is especially critical for applications working with machine learning and image processing, which are characterized by large amounts of calculations, a significant number of simultaneous users, high frequency of requests and the need for scalability. This situation requires the implementation of reliable, modular and elastic architectures that can support both online real-time query processing and offline analytics of large data sets. In particular, microservice architecture and distributed processing approaches are increasingly becoming the basis for building productive and supported ML solutions in the field of computer vision, telemedicine, autonomous systems and more. Traditional monolithic approaches, despite their initial effectiveness, have proven to be poorly suited for scaling and support in high-load environments. This is especially relevant in the context of systems that process large amounts of input data in real time, in particular in the fields of artificial intelligence, machine learning, image processing, telemedicine, autonomous systems.

The emergence of the concepts of Big Data, distributed computing, cloud computing environments and microservice architecture has become a response to the need for flexible, fault-tolerant and scalable solutions. Such architectures allow you to optimize resources, ensure horizontal scaling of individual components, reduce the time for deployment and updating services, as well as improve the overall reliability and performance of systems.

Microservice architecture has a number of advantages over monolithic, in particular in conditions of high loads, the need for scaling and frequent system updates. Microservices allow you to scale individual components independently of each other, which allows for optimal use of resources. Updating individual microservices does not require a complete restart of the system. This reduces risks and speeds up the release cycle. A monolith requires a complete recompilation

[*] *CIAW-2025: Computational Intelligence Application Workshop, September 26-27, 2025, Lviv, Ukraine*

[1,*] Corresponding author.

[†] These authors contributed equally.

o.pitsun@wunu.edu.ua (O. Pitsun); Miroslavshimchuk@gmail.com (M. Shymchuk);

0000-0003-0280-8786 (O. Pitsun); 0009-0009-4270-8131 (M. Shymchuk);

and redeployment of the entire system for each change. Microservices allow for independent CI/CD pipelines for each service, which improves automation and control.
The relevance of researching high-load architectures is due not only to the technical complexity of modern applications, but also to the need to maintain their scalability, elasticity, and high availability in the face of increasing computational and service loads.

2. Problem statements

The following tasks are set in this work:

1. to make a comparative analysis of architectures for the implementation of high-load software tools;
2. to consider the main components of the high-load architecture for the further design of an improved architecture with high-growth of artificial intelligence elements;
3. to implement the proposed high-load architecture for working with images based on artificial intelligence elements.

3. Literature review

The paper proposes an architectural solution for reducing the load in high-performance networks by dynamically distributing requests based on the current system state and time synchronization using the Marzullo algorithm. The implemented mechanisms reduce the computation time polynomially and provide effective resource management in a distributed environment [5]. The article discusses key challenges and effective solutions for designing high-load software systems, in particular through scaling microservice architecture, distributed data storage, caching and load balancing. The author proposes a generalized architectural model that provides high availability, low latency and fault tolerance through automation, asynchronous processing, monitoring and testing, adapted to different technology stacks [6].

The article analyzes ways to optimize microservice architecture under high load conditions by combining modern technologies such as containerization (Docker), orchestration (Kubernetes), architectural patterns (CQRS, Event Sourcing), caching (Redis) and fault tolerance mechanisms (Circuit Breaker, Bulkhead). The conducted testing confirmed the effectiveness of the approach: with 5000 simultaneous users, the average response time was 450–500 ms with a minimum number of errors (<0.5%), which indicates the significant potential of the proposed model for creating scalable and reliable distributed systems[7]. The article presents an automatic load balancing architecture (ALBRL) for software-defined networks (SDN), which is based on an advanced deep reinforcement learning algorithm (DDPG). The proposed solution effectively solves the problems of slow convergence and uneven traffic distribution, improving network throughput and accelerating learning due to the optimized experience sampling mechanism based on the SumTree structure [8].

In the article [9], the feasibility of using microservice architecture for developing high-load, fault-tolerant software solutions is substantiated using the example of the MedicinePlanner medication reminder system. An approach to implementing client-server interaction using .NET, gRPC, Angular and containerization is presented, which provides scalability, component isolation, automatic deployment to Azure and increased fault tolerance, unlike monolithic architecture. In the paper [10], an approach to implementing Operational Intelligence using mathematical modeling and machine learning for monitoring, load forecasting and increasing the reliability of high-load IT systems is proposed. A transaction analysis model in a Big Data environment is presented, which ensures error detection before user interaction, as well as an algorithm for implementing a monitoring system for projects focused on processing large amounts of data.

The proposed engineering framework combines Model-Driven Development (MDD) techniques with scalability principles, contributing to the creation of flexible, modular and reusable educational platforms. Particular attention is paid to the use of adaptive algorithms, cloud

technologies and the implementation of a learning content management system (LCMS), which allows dynamically adjusting educational routes to individual student needs. The prototype of the system, tested under variable load conditions, demonstrated improved performance, scalability and user engagement compared to traditional e-learning solutions [11].

The paper [12] discusses the challenges associated with processing heterogeneous and explosively growing medical data in the era of 5G and the development of the Internet of Medical Things (IoMT). The authors reviewed the integration of cloud and edge computing with artificial intelligence technologies to ensure timely processing of medical big data, increase the efficiency of medical resource use and guarantee the security and confidentiality of personal data. Particular attention is paid to the potential of edge–cloud architectures in the context of providing high-quality medical services, and promising areas of further research in the field of secure, scalable and intelligent IoMT infrastructure are outlined. In [13], an approach to developing a highly loaded and fault-tolerant medication reminder software – the MedicinePlanner system – is presented. The main architectural basis is a microservice approach that provides scalability, error isolation and ease of maintenance. The server part is implemented using .NET and gRPC, the client part is implemented using Angular with a modular structure. In [14], the impact of serverless computing and microservice architecture on the transformation of cloud platforms is considered, which allows creating scalable, cost-effective and flexible solutions with an emphasis on business logic instead of infrastructure management. Integrating event-driven patterns with serverless models improves the resilience, scalability, and operational efficiency of systems by providing independent scaling, resource management automation, and high availability in enterprise environments. In [15], we propose an adaptive AI-based skin cancer classification framework that combines microservice architecture, Kubernetes orchestration, and event-driven architecture (EDA) using Google Cloud Pub/Sub. The framework provides real-time processing with high accuracy (97%), throughput (1000 queries/s), and reliability (99.99% availability), allowing for efficient integration of AI models into clinical workflows for early diagnosis and monitoring of skin cancer.

4. Hight-load architecture components

High-performance architecture components can be classified according to several key criteria that reflect functional purpose, level of responsibility, type of load, method of scaling, and interaction between them.

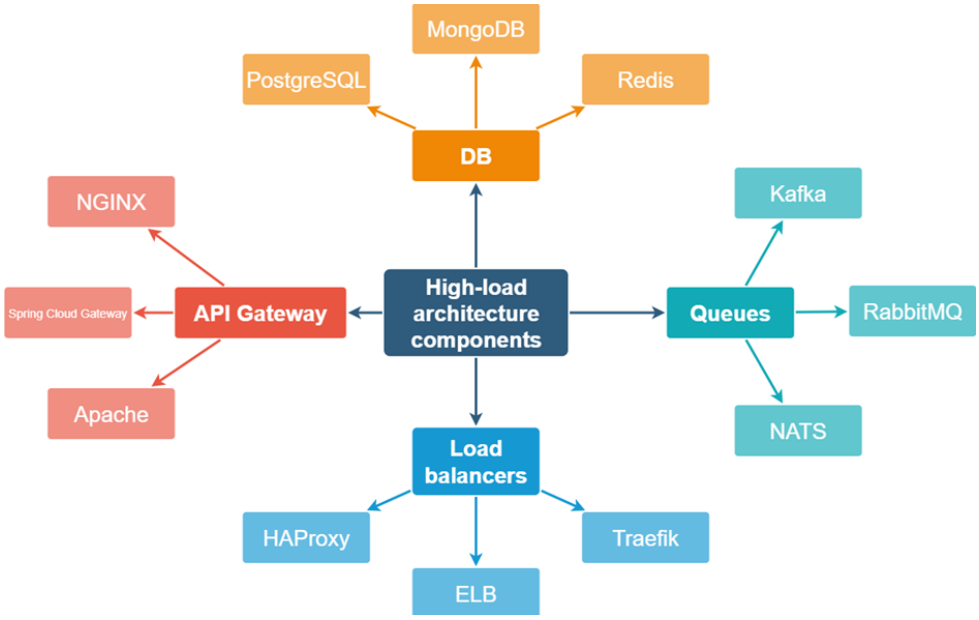

Figure 1: High-load architecture components.

API gateways act as a central entry point for external and internal clients. They are responsible for routing requests, load balancing, authentication, authorization, response aggregation, and format transformation. In high-load systems, an API gateway often acts as the first layer of protection and optimization, allowing centralized management of the flow of requests.

Databases and other information storage are critical for high-load systems. They provide storage of structured (SQL, relational) or unstructured (NoSQL, object) data.

Message queuing systems allow for asynchronous communication between services, which significantly reduces component interdependencies and increases fault tolerance. Such components provide data buffering, delivery delay, delivery guarantees, and event prioritization.

Load balancers evenly distribute incoming requests between microservice or server instances. In high-availability systems, these components provide automatic traffic redirection in the event of a node failure, and also support health-check mechanisms. Both software (HAProxy, Traefik) and hardware solutions are used.

Caching mechanisms significantly reduce the load on the main data sources and reduce response time. Caches can be implemented as in-memory storage (for example, Redis or Memcached) and used to store query results, user sessions or configurations. In complex architectures, hierarchical or distributed caching with TTL policies and eviction can be used.

Monitoring and logging components are responsible for monitoring the system status, query tracing, performance analysis and anomaly detection. Such systems use metrics (Prometheus), centralized event logs (ELK-stack, Loki), dashboards (Grafana) and tracing systems (Jaeger, Zipkin).

Service lifecycle management in distributed architectures is implemented using containerization (Docker) and orchestrators (Kubernetes). These tools provide automatic deployment, scaling, updates, disaster recovery, and load balancing between instances. In high-load systems, support for rolling updates, blue-green deployment, and CI/CD pipelines is also important, which allows for rapid implementation of changes without compromising system stability.

A comparative analysis of high-load architecture components by request processing method is given in Table 1.

Table 1
Comparative analysis of high-load architecture components by request processing method

Component	Technologies	Role
API gateways	Kong, NGINX, Spring Cloud Gateway	Load reduction, access control
Computing cores	Python, TensorFlow, FastAPI, gRPC	Independent scaling, resource optimization
Data storage	PostgreSQL, MongoDB, Redis, S3	Parallel access, replication, sharding
Load balancers	HAProxy, Traefik, AWS ELB	Increased fault tolerance, uniform load
Cache	Redis, Memcached	Minimizing delays, optimizing traffic
Monitoring and logging	Prometheus, ELK, Grafana, Jaeger	Performance monitoring

5. Architectural approaches for high-load systems

To implement high-performance architectures in machine learning (ML) and image analysis software, a number of architectural approaches are used to ensure scalability, performance, fault tolerance, and flexibility.
The distribution of architectures for developing systems based on high-performance is shown in Figure 2.

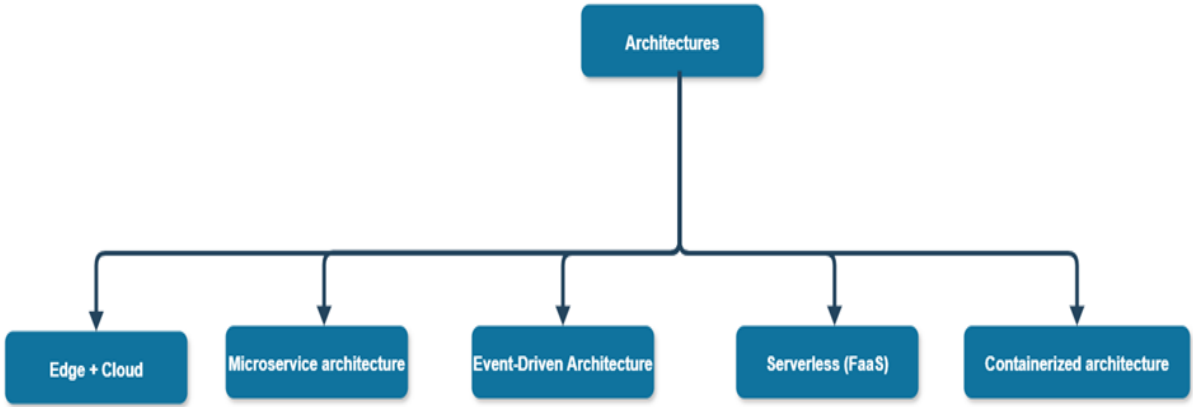

Figure 2: Classification of architectures for developing systems based on high loads.

The comparative analysis is presented in Table 2.

Table 2
Comparative analysis of architectures

Approach	Features	Advantages
Microservice architecture	Decomposition of the application into independent modules	Scalability, flexibility, fault tolerance
Event-Driven Architecture	Event processing via message brokers (Kafka, RabbitMQ)	Asynchrony, efficient interaction between services, scalability
Serverless (FaaS)	Executing code as functions on demand	Automatic scaling, low cost for rare tasks
Containerized architecture	Using Docker + Kubernetes to manage ML components	Flexible deployment, environment standardization, CI/CD support
Edge + Cloud	Computations are performed partly on the device, partly in the cloud	Reduced latency, reduced traffic, security
Streaming Architecture	Real-time data processing (Flink, Spark Streaming)	High performance for streaming data, low latency
MLops	ML model lifecycle management	Automate model deployment, updates, and monitoring

Each of these approaches is distinguished by a certain set of characteristics, in particular, the level of distribution, scalability flexibility, monitoring complexity and DevOps integration. For example,

the microservice architecture provides independent scaling of components, which is especially useful for tasks with a high load on individual stages of the ML pipeline. In turn, the streaming architecture is more effective for real-time systems, in particular for video analytics or surgical monitoring. Of particular note is the hybrid (Edge + Cloud) model, which allows you to transfer part of the calculations to peripheral devices, which reduces latency and reduces the requirements for the data transmission channel. At the same time, architectures with an emphasis on MLOps provide automation of the entire model life cycle - from training to monitoring in a productive environment, which is critical for maintaining the quality of models in conditions of dynamic data.

6. Framework's examples

Large-Scale Intelligent Microservices This is a framework using Apache Spark to orchestrate intelligent microservices: image classification, text analysis, anomalies, search. The system runs on hundreds of machines, combining the capabilities of Spark distribution with a low-latency containerized API. It is designed to integrate ML services directly with databases, with minimal overhead [1]. Keralty covers infrastructure from dozens of data sources and implements Big-Data architecture and machine learning with batch and real-time modes. Components based on big-data storage (, scalable warehouse, real and batch layers, plus an inference layer and services via microservices with open APIs[2]. Smartly.io is a system for processing hundreds of images per second via HTTP requests (for advertising banners). Key approaches: caching of originals and results, distribution of tasks to worker services without a single node of failure, automatic scaling of GPU/CPU components according to load [3]. Simpliroute is characterized by the presence of an ETL pipeline on Google BigQuery + Dataproc, Kubeflow, MLFlow, Spark + XGBoost. Data is orchestrated via Pub/Sub, BigQuery, and large-scale pipelines with batch processing and retraining models [4]. All these systems demonstrate classic architectural patterns of high-load ML systems: distribution, asynchrony, containerization, ML service orientation, ETL integration, stream/batch, and infrastructure automation.

7. Proposed architecture

The proposed system is developed based on microservice architecture, characterized by mobility, scalability and wide functionality with the possibility of using cloud services. The algorithm of work is as follows:

1. loading and deploying docker containers;
2. the frontend part is implemented using standard tools and vue.js technology to implement reactive interaction with the server and increase functionality for users;
3. API Gateway is implemented as a separate service and is designed to provide routing;
4. the backend component is implemented as a laravel framework and in combination with the mysql database. This functionality is designed to process information about research and joint interaction over records;
5. the service for managing tools for training neural, convolutional, Unet, recurrent, graph is used for training and implementing artificial intelligence tools. The separation of this block from the monolith will make the system "lighter" and open to expansion;
6. the image processing service requires separate software, in particular the OpenCV library, therefore it is implemented as a separate microservice. The task of the block is to process images, in particular, improve quality, calculate quantitative characteristics;

Visually, the architecture is shown in Figure 3.

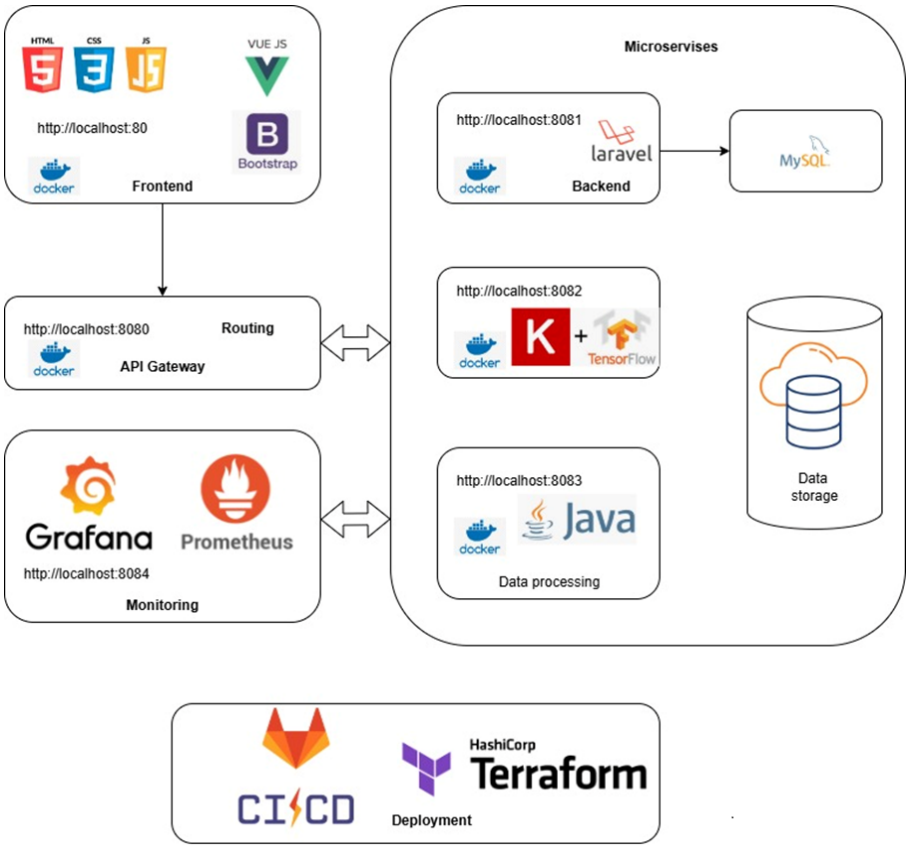

Figure 3: Proposed system architecture.

8. Results

A comparative analysis of the developed architecture with analogues is given in Table 3. Software tools that perform similar functions in biomedical image processing were selected as analogues.

Table 3
Comparative analysis of the developed architecture with analogues

Function	HIAMS	ImageJ	epredia	Leicabio systems	Proposed system
Image preprocessing	+	+	+	+	+
Image segmentation	+	+	+	+	+
Microservice architecture	-	-	-	-	-
Load balancing	-	-	-	-	+
CI/CD	-	-	-	-	+
Grafana/prometheus	-	-	-	-	+

A prototype web page for segmentation using unet is shown in the figure 4.

Existing systems HIAMS, ImageJ, Epredia, Leicabiosystems provide basic image processing and segmentation functions, but do not support modern approaches to scalability and monitoring. The proposed system, in contrast, is supplemented with load balancing capabilities, CI/CD, and integration with Grafana/Prometheus, which makes it more flexible, scalable, and suitable for industrial use.

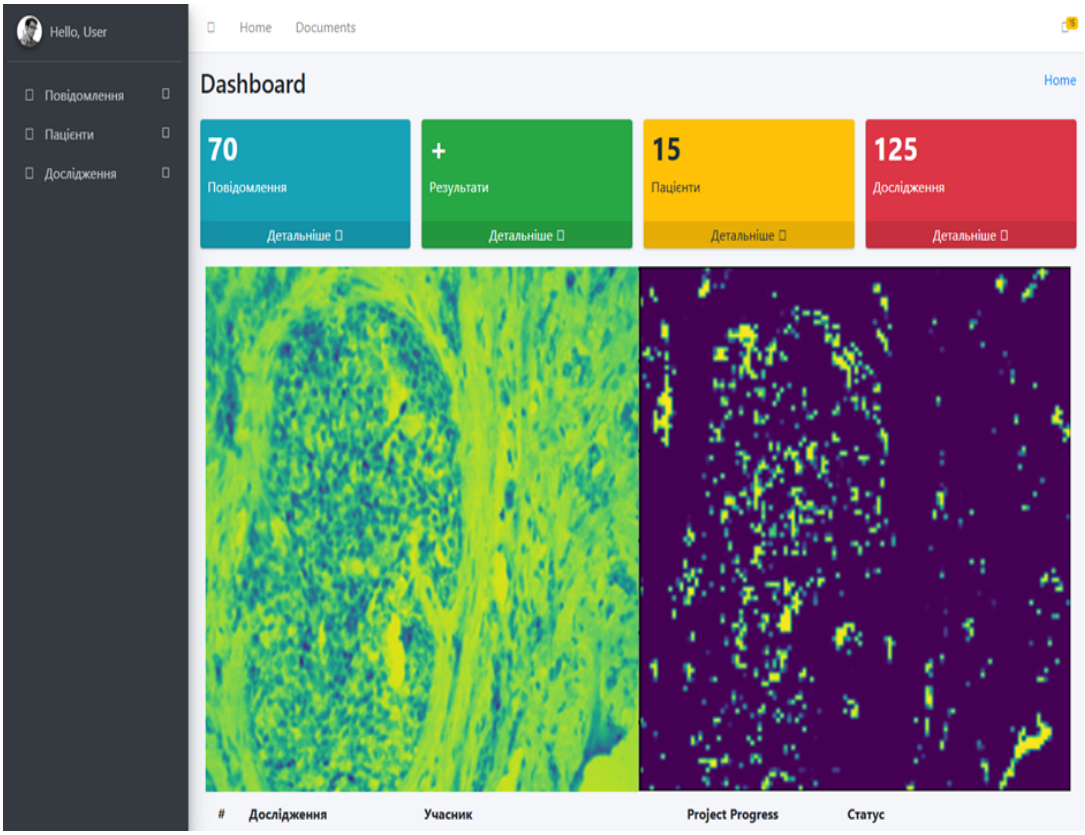

Figure 4: Prototype of a web page for segmentation using u net.

The proposed interface consists of the following elements:

1. side menu;
2. main dashboard;
3. reference information about the conducted studies, including data on the number of patients;
4. messages, study results, etc.;
5. original histological, cytological or immunohistochemical image;
6. result of segmentation using Unet tools;
7. table with a list of studied files.

The main purpose of MLOps is to ensure the complete and continuous lifecycle of machine learning models — from development and training to deployment, monitoring, and updates in a production environment.

MLOps [16,17] allows you to turn experimental ML developments into stable and efficient services.

Analyzing the data in Table 1, we can conclude that the proposed approach to developing architecture based on microservice architecture allows us to implement a complex system based on machine learning with maximum consideration of high system load.

Figure 5 shows a comparative analysis of the deployment time of different types of software for processing deep learning on a cloud DigitalOcean service.

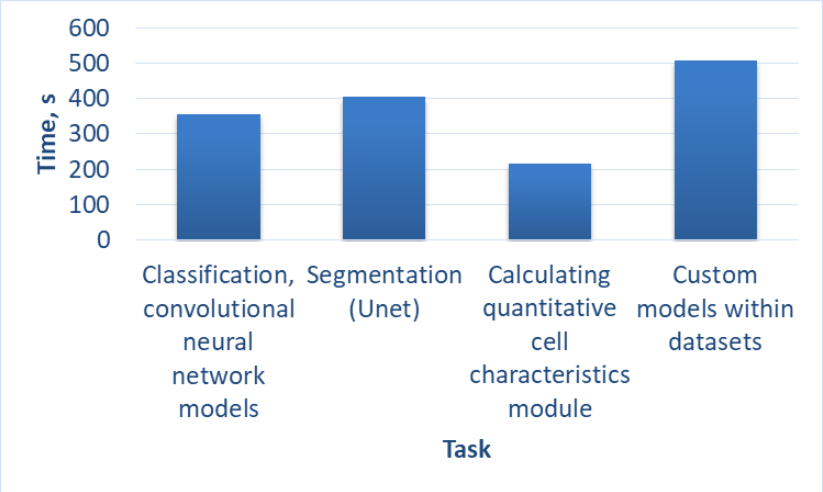

Figure 5: Comparative analysis of the deployment time of different types of software.

The analysis shows that the software deployment time using continuous code delivery for software with deep learning elements is higher than for traditional web applications. The longest integration time was taken by the process of deploying software for custom models with many datasets, which confirms the uniqueness of developing tools for implementing deep learning in the cloud.

Conclusions

In this work, the following was implemented:

1. based on an analytical approach, a comparative analysis of architectures for the implementation of high-load software tools was conducted, which allowed us to identify modern approaches to the implementation of high-load systems;
2. the main components of the high-load architecture were considered for the further design of an improved architecture using artificial intelligence elements;
3. a high-load architecture for working with images was implemented programmatically based on artificial intelligence elements and using a microservice approach to improve the quality and stability of the system.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. Hamilton, N. Gonsalves, C. Lee, A. Raman, B. Walsh, S. Prasad, D. Banda, L. Zhang, L. Zhang, and W. Freeman. "Large-scale intelligent microservices." In 2020 IEEE International Conference on Big Data (Big Data), pp. 298-309. IEEE, 2020. https://arxiv.org/pdf/2009.08044

[2] F. López-Martínez, E. Núñez-Valdez, V. García-Díaz, and Z. Bursac. 2020. "A Case Study for a Big Data and Machine Learning Platform to Improve Medical Decision Support in Population Health Management" Algorithms 13, no. 4: 102. https://doi.org/10.3390/a13040102

[3] R. Auvinen "Identifying relevant metrics as performance indicators in a B2B SaaS company: Case Smartly. io." (2017). https://aaltodoc.aalto.fi/items/bb5a38bf-051c-4e69-bf68-69d87ea73a2c

[4] Atoum, I. (2023). Measurement of key performance indicators of user experience based on software requirements. Science of Computer Programming, 226, 102929.https://doi.org/10.1016/j.scico.2023.102929

[5] Kryvenchuk, Y., Mykalov, P., Novytskyi, Y., Zakharchuk, M., Malynovskyy, Y., Řepka, M. (2020). Analysis of the Architecture of Distributed Systems for the Reduction of Loading High-Load Networks. In: Shakhovska, N., Medykovskyy, M.O. (eds) Advances in Intelligent Systems and Computing IV. CSIT 2019. Advances in Intelligent Systems and Computing, vol 1080. Springer, Cham. https://doi.org/10.1007/978-3-030-33695-0_50

[6] Volodymyr Kozub. (2025). Problems and Solutions in Building Highly Loaded Software. The American Journal of Engineering and Technology, 7(03), 230–236. https://doi.org/10.37547/tajet/Volume07Issue03-20

[7] Artem Iurchenko. (2025). Optimization of Microservices Architecture Performance in High-Load Systems. The American Journal of Engineering and Technology, 7(05), 123–132. https://doi.org/10.37547/tajet/Volume07Issue05-10

[8] Chen, Junyan, Yong Wang, Jiangtao Ou, Chengyuan Fan, Xiaoye Lu, Cenhuishan Liao, Xuefeng Huang, and Hongmei Zhang. "ALBRL: Automatic Load-Balancing Architecture Based on Reinforcement Learning in Software-Defined Networking." Wireless Communications and Mobile Computing 2022, no. 1 (2022): doi: 3866143.

[9] V. Kornuta, E. Sobotnyk, I. -M. Katamai and Y. Katamai, "Using Microservice Architecture for High-Load Information Systems on the Example of MedicinePlanner Service," 2022 12th International Conference on Advanced Computer Information Technologies (ACIT), Ruzomberok, Slovakia, 2022, pp. 437-442, doi: 10.1109/ACIT54803.2022.9913185.

[10] S. Fedushko, T. Ustyianovych, and M. Gregus. 2020. "Real-Time High-Load Infrastructure Transaction Status Output Prediction Using Operational Intelligence and Big Data Technologies" Electronics 9, no. 4: 668. https://doi.org/10.3390/electronics9040668

[11] Sutarman, A., Williams, J., Wilson, D., & Ismail, F. B. (2024). A Model-Driven Approach to Developing Scalable Educational Software for Adaptive Learning Environments. International Transactions on Education Technology (ITEE), 3(1), 9–16. https://doi.org/10.33050/itee.v3i1.663

[12] L. Sun, X. Jiang, H. Ren and Y. Guo, "Edge-Cloud Computing and Artificial Intelligence in Internet of Medical Things: Architecture, Technology and Application," in IEEE Access, vol. 8, pp. 101079-101092, 2020, doi: 10.1109/ACCESS.2020.2997831.

[13] V. Kornuta, E. Sobotnyk, I. -M. Katamai and Y. Katamai, "Using Microservice Architecture for High-Load Information Systems on the Example of MedicinePlanner Service," 2022 12th International Conference on Advanced Computer Information Technologies (ACIT), Ruzomberok, Slovakia, 2022, pp. 437-442, doi: 10.1109/ACIT54803.2022.9913185.

[14] D. Kalita. Embracing serverless microservices: A decoupled, scalable, and event-driven evolution in cloud architecture. World Journal of Advanced Research and Reviews, 2025, 27(01), 902-915. doi: https://doi.org/10.30574/wjarr.2025.27.1.2470.

[15] M. Elmeligy, A. Ezzat, M. Mobtasem, S. Moustafa and M. S. Darweesh, "An Efficient Hybrid Infrastructure for AI Healthcare System: Skin Cancer Classification," 2024 12th International Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC), Alexandria, Egypt, 2024, pp. 150-155, doi: 10.1109/JAC-ECC64419.2024.11061241.

[16] O. Berezsky, O. Pitsun, G. Melnyk, B. Derysh, P. Liashchynsky. Application Of MLOps Practices For Biomedical Image Classification - Ceur Workshop Proceedings, 2022, 3302, pp. 69–77 https://ceur-ws.org/Vol-3302/short3.pdf

[17] O. Berezsky, O. Pitsun, G. Melnyk, H. Poperechna, Microservices-based architecture for biomedical image processing software - Ceur Workshop Proceedings, 2024, 3892, pp. 73–80 https://ceur-ws.org/Vol-3892/short1.pdf