UDC 658.6
JEL Classification: L81
DOI: 10.15587/2706-5448.2022.253932
Article type «Reports on Research Projects»

Vadim Yakovenko,
Yuliia Ulianovska,
Tetiana Yakovenko

ANALYSING FEATURES OF E-COMMERCE SYSTEMS ARCHITECTURE

The object of the research is the process of designing the architecture of high-load systems. The conducted research is based on the system approach to design the architecture of e-commerce systems, characterized by high workload due to the large number of users working simultaneously with the system, a large amount of data and a significant number of complex calculations. The main hypothesis of the research is that the efficiency of such systems depends on the efficiency of each individual step to scale up the system and the consistency of these steps. The maximum efficiency can be achieved only if the resource constraints and requirements, which are determined by the key stakeholders of the projects, consider the specifics of the business system. This paper examines the methodological support of the developing high-load systems architecture. Within this research let's analyze such specific features of high-loaded systems as scalability, rigidity, and response time and demonstrate the importance of considering these features when designing the architecture of high-loaded systems. This paper analyzes approaches to developing high-load systems architecture, their advantages, and disadvantages. It is suggested to use hybrid scaling method, which is based on combining two approaches – microservices and monolithic. It is also suggested to use a microservices approach for high-loaded and requiring scaling parts and a monolithic approach for non-loaded parts of the system. The research indicates the parts of the system that are usually highly loaded in e-commerce systems and require a microservices approach to design their architecture. This paper analyzes approaches to database scaling and organization of data replication. The application of the proposed approach to design the architecture of high-load systems, including the e-commerce systems, allows designing a system that can be easily scaled when necessary. At the same time, the system can be improved and further developed.

Keywords: *high-load systems, microservice architecture, monolithic architecture, e-commerce systems, system architecture.*

Received date: 23.11.2021
Accepted date: 21.12.2021
Published date: 28.02.2022

How to cite
Yakovenko, V., Ulianovska, Y., Yakovenko, T. (2022). Analysing features of e-commerce systems architecture. Technology Audit and Production Reserves, 1 (4 (63)), 27–31. doi: http://doi.org/10.15587/2706-5448.2022.253932

1. Introduction

The pandemic, quarantine restrictions and the necessity of organizing remote work have increased the demand for the development of e-commerce systems and business process automation. These systems are often classified as high-load systems. High-load systems are applications with high workload, which occurs through:

– many users simultaneously working with the system;
– high volume of data to be processed;
– the presence of numerous complex calculations [1].

The above factors are typical for high-load systems, both separately and jointly. Such a system requires a significant number of resources to operate.

The development of high-loaded systems has certain peculiarities [1–3]:

– *The main feature of high-load business systems* is their rigidity: it is possible to modify only some parts, because the flexibility of such systems requires a significant number of resources. For example, it is impossible to make access to the data flexible. It is necessary to clearly define the database for the system to work with, considering the amount of data and the frequency of requests, in order to ensure its stable performance.
– *The response time* is another important factor. The interaction between users and the application is carried out by submitting requests, which should be responded to in a suitable time span.
– *Scalability* is a necessary feature for high-load systems, which determines their ability to increase the maximum allowable workload (the number of users working simultaneously with the system, the amount of data, etc.).

These peculiarities require critical analysis when developing the architecture of e-commerce systems. Therefore, *the aim of this research* is to develop the architecture of e-commerce systems, considering the peculiarities of high-load systems operation. *The object of this research* is the process of developing the architecture of high-load systems.

ISSN 2664-9969

2. Research methodology

The conducted research is based on the approaches described in papers [2, 4, 5]. The key hypothesis of the research is that the efficiency of such systems depends on the efficiency of each individual step to scale up the system and the consistency of these steps. The maximum efficiency can be achieved only under consideration of resource constraints and requirements, determined by the key stakeholders of the projects, and the specifics of the business system. The principles of developing the architecture of high-load systems, presented in papers [2, 5, 6], are analyzed. Based on this analysis and considering the architecture of data processing mechanisms and synchronization of modules [4, 7], database architecture [8], the necessity of using a systematic approach are determined to develop the architecture of high-load systems.

3. Research results and discussion

Development of web-based applications is optimal solution to develop e-commerce systems. A web-based application is a client-server application (the client is a browser, and the server is a web server) for which data is stored on the server, and data exchange takes place in the network. Their important advantages for e-commerce systems are the following:

– The system can be operated by a great number of users at once.

– They do not require installation on users' devices, thus they can be used whenever, and do not require additional workstations, increasing hardware power, etc. Users need only a browser and access to the Internet to work with the system.

– Developing web-based applications is cheaper.

– All the updates and changes to the web application became automatically available for all users.

A web-based application comprises the application code and database. When a user plans to use it, he or she sends a request to the server. The server processes the request, selects the necessary data from the database, generates a response and sends it to the user. According to the results of research, the maximum time to perform these operations should not exceed 6 seconds for complex requests. Optimal average time to process a request for comfortable work of the user, which indicates the normal functioning of the system, is 3 seconds. A longer response time is unacceptable for such systems due to the high probability to lose potential customers.

Another important characteristic of the system performance is a maximal number of requests processed per second (Requests Per Second (RPS)).

Therefore, the most important is scalability of the system, which give opportunity to manage its performance indicators: the time of processing requests and RPS.

Let's analyze the approaches to scaling. There are two approaches to scaling [9, 10]:

– Vertical scaling involves increasing the capacity of some components of the system to increase the overall capacity. As a rule, vertical scaling is performed by replacing some devices with more powerful ones. This is the simplest way of scaling, which does not require any changes to the program.

– Horizontal scaling involves splitting the system into structural components distributed among different computers and increasing the number of servers to perform specific functions in parallel. Consequently, the need to add nodes to the system, working as a single unit, and to modify the program for efficient use of the additional resources is evident.

The development of a high-load system requires a flexible approach to scaling and combination of both approaches. Obviously, the vertical scaling is not endless [11]. It is optimal to use it when the server is too old to bear the workload. Thus, it is quicker and cheaper to replace it with a new one, instead of changing the program code.

The horizontal approach to scaling involves splitting the application into several modules, which can be distributed among the servers, or multiplication of the highest-loaded part of the application.

For example, the most loaded part of e-commerce systems can be the catalog. In this case, the task is to share the performance of this module between different servers, i. e., to organize parallel computing of this function.

This solution also has certain restrictions, because parallel computing creates the problem of data synchronization: each server, each stream, has its own data. They are no longer synchronized. The growing workload leads to a haphazard data state: users can change the same data simultaneously, but this situation is not acceptable. The synchronizer implementation removes the parallelism, but the load on the synchronizer grows. Therefore, the optimal solution is to isolate those high-load components performed synchronously and to use a vertical scaling approach for them.

Therefore, the second approach in combination with vertical scaling is optimal to scale high-load systems. However, there is a mention that the application of this approach can differ for a particular system depending on its specific features and business requirements.

Another peculiarity to consider when scaling the system is the load on the integrating module. The more the number of separate modules is, the more the load on the integrating module grows with the increase of the load on the modules. Obviously, the goal of scaling is to get a well-performing system rather than a set of separate modules. Therefore, one should consider that the complexity of communication and load on the integrating module grow in geometrical progression with increasing the number of system modules.

In high-load systems, it is also important to duplicate critical components. All the critical components of a high-load system, which affect its functionality, must be duplicated both in software and hardware (duplicate the equipment). These duplicates are not necessarily to work simultaneously, but they must ensure availability to use them in the case when the primary duplicates are not able to handle the load.

An important component of high-load systems is the monitoring system. As a rule, problems in the system operation arise at moments of peak load, i. e., when the business earns the most. The monitoring system allowsto identify the issue causing the failure in the work, and immediately go to fixing these issues. Naturally, it is possible to identify the issue without the monitoring system. But in this case, it is necessary to involve highly qualified specialists and spend time to find the reason. Moreover, the monitoring system allowsto analyze the module's performance and prevent possible failures, and thus minimize the probability of profit loss.

The system's inflexibility allows to save money on equipment. The cost of the hardware for a high-load system is always much higher than the cost of a typical application. When developing a flexible high-load system, the number of required hardware increases significantly.

So, the first method to develop the architecture for e-commerce systems is the method of monolithic architecture: creating the program code using a set of plugins. Each plugin provides the implementation of a particular feature. For example, integration with external solutions, personalization of the payment terms, integration of payment systems, discount calculation system, etc.

An alternative approach is the microservice architecture, under which the program is composed of a set of microservices and each microservice has its own database and operates independently. The advantage is the ability to develop, maintain, scale, and improve each of the microservices separately. But an important disadvantage is its cost. For example, to improve a particular microservice, it is necessary to analyze the code, make the appropriate changes, plan updates, test the system to ensure the absence of conflicts, prepare documentation, etc.

Developing monolithic applications is much cheaper, while they have a single code, and the application works with a single database. In the case of microservices architecture it is necessary to develop a protocol for interconnection with the core, the main product page, for each microservice. In this case, when processing a user's request, the program works with several databases of different microservices, all of which provide their own response to the request, which the application consolidates into a single response for the user.

It is obvious that the optimal solution is a hybrid (combined) architecture. A monolithic part is developed for unloaded functions of e-commerce systems, and microservices are developed for functions which are high-loaded and could require scaling.

For e-commerce systems, as a rule, microservices are required to implement such functions:

– catalog;
– data buses for importing data from external software;
– API for interaction with mobile devices and other services;
– mailer and other services to communicate with clients.

The advantages of this approach are, foremost, reduction of costs for development and support of the system. The hybrid method requires less time for development, and it is cheaper. For that reason, it is the optimal solution for startup projects, companies with limited budgets, companies willing to test a business hypothesis, etc. Moreover, the application has no disadvantages in functionality and scalability.

An example of such hybrid architecture for an e-commerce system is shown in Fig. 1.

Fig. 2 shows an example of the infrastructural architecture of the e-commerce system.

The principle of development and performance of such a system is as follows. It is recommend to use Python, Framework and Django within the development process, which is optimal regarding the development terms.

Microservices architecture is used to implement such functions:

– notification service;
– search engine based on Elastic Search;
– product matching service (finding analogues for products that are not available in the catalog);
– bus of input and output data processing;
– Rest API;
– signing documents with EDI.

The PostgreS as a database of enterprise level with significant scalability is optimal for e-commerce system. Elastic is used to scale the catalog: all complex requests are sent to Elastic, and then the results of processing are shown in the catalog. Search within the system is full-text and implemented with Elastic.

Data import is arranged using the Celery task broker. Each file for import or export is sent to the task broker, which creates a list of such files, and for each of them it determines the process, performing the import or export procedure. This approach to organize data import and export enables horizontal scaling of this microservice.

E-commerce systems, as a rule, have a complex structure of data storage, complex personalization logics, huge amounts of data. Therefore, in-memory database Redis is suitable for scaling the catalog, as it stores the data in the operating system in non-relational form.

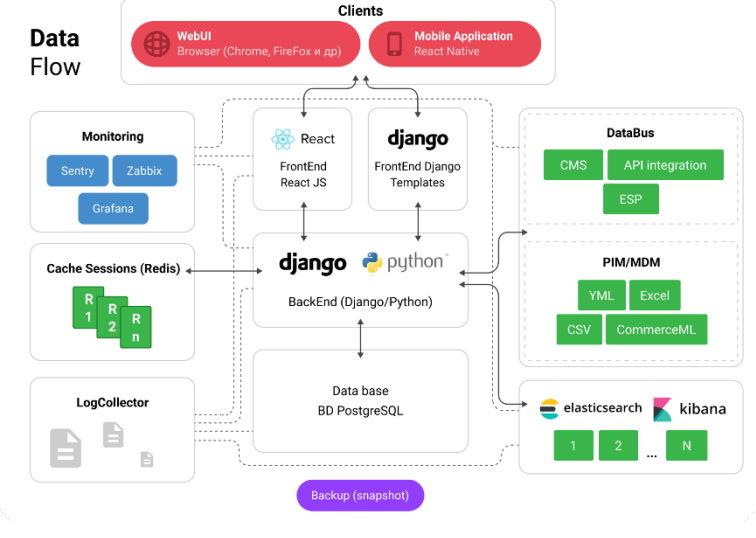

Fig. 1. An example of a software architecture for an e-commerce system

ISSN 2664-9969

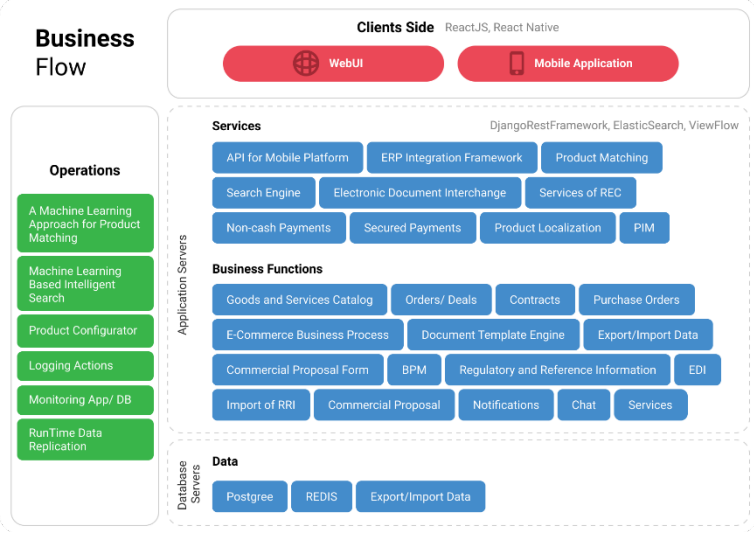

Fig. 2. An example of the infrastructure architecture of the electronic commerce system

The implementation of the log monitoring system enables identifying errors in the system. Automated monitoring of projects and errors allows responding immediately: to renew projects or, when possible, to fix errors. Similarly, monitoring is used for integration processes with external programs and services. The system Jenkins could be used to ensure continuous integration of e-commerce system with external software. The integration with the accounting system to provide displaying real-time data on the availability and stock of goods can be an example. Another example is transferring information about orders from the e-commerce system to the accounting system for document processing, updating amounts of stocks, etc.

Scaling is implemented by containerization with the technology Docker. This system allows to quickly set up a server environment for web applications and control resources. The approach has a significant advantage compared to virtualization. Virtual computers are created with software to perform some operations. As each virtual computer requires a separate operating system, which runs on the server using its resources, the efficiency of using the server's resources is low.

Containerization, however, allows to place programs and processes in a separate container and reserve a certain number of resources for it, manage its performance, accessibility, processes, transfer the container among servers, perform monitoring, and so on.

The system Docker Swarm is used for orchestration – management of containerization process. This system allows to create clusters (sets of servers) and control their performance. The software is distributed on these clusters: the master application, the database and the microservices. Each element has its own cluster. The orchestration system Docker Swarm provide monitoring and management of the servers. When a cluster is unavailable, the system replaces containers to other servers and resumes their operation, thus ensuring the system's availability and continuity of its performance.

Moreover, the orchestration system allows online replication of data. Docker Swarm routes traffic in such a way: requests to read data go to the Slave database, and requests to write data go to the Master database. At the same time, the Master database constantly transfers all the updates to the Slave databases. Due to regular replication in the cluster, there is always a stable version of the database to restore all the data.

The specific features of suggested architecture for the e-commerce solutions are relevant for the development of mobile and web applications. It is reasonable to use such principles to develop high-load systems, or systems with the potential need for scaling. A probable development of this research could be further improvement of the architecture with the aim of increasing the speed and permissible loads.

4. Conclusions

The paper analyzes the approaches to designing high-load systems, their advantages, and disadvantages. It is suggested to use hybrid method for scaling, which is based on combining two approaches – microservices and monolithic. Microservice method could be used for high-loaded and requiring scaling parts of the system, the monolithic method could be applied for non-loaded parts. It is found that usually high-loaded parts of the system are the catalog, data buses for data import, API support for interaction with mobile devices and other services, and services to communicate with users. These are the parts of e-commerce systems requiring a microservices approach to develop the architecture. The paper provides analysis of approaches to database scaling and organization of data replication. Scaling can be implemented by tools of containerization technology Docker, and the management of the containerization process could be implemented using system Docker Swarm. The routing system distributes traffic from the application and send all the reading requests to the Slave database and all the writing requests to the Master database. The information in the databases is constantly updated due to regular replication. As a result, the cluster always has a stable version of the database to restore all the data. The implementation of the suggested approach to design high-load systems architecture, including e-commerce systems, enables developing a system, which can be easily scaled up anytime. At the same time, the system can be improved and updated.

References

1. Amyrov, S. N. (2020). Features of the Development of High Load Data Systems. *International journal of open information technologies, 8,* 38–45.
2. Lackermair, G. (2011). Hybrid cloud architectures for the online commerce. *Procedia Computer Science, 3,* 550–555. doi: http://doi.org/10.1016/j.procs.2010.12.091
3. Chamkiev, A. T. (2021). Osnovnye osobennosti arkhitektury vysokonagruzhennykh sistem. *Obzor sushchestvuiushchikh reshenii. Tochnaia nauka, 98,* 19–21.
4. Voichyk, S. S., Tymoshyn, Yu. A. (2018). Arkhitektura mekhanizmiv obrobky danykh ta synkhronizatsiia moduliv u vysokonavantazhenykh systemakh Smart City. *World Science, 1(10(38)),* 22–24. doi: http://doi.org/10.31435/rsglobal_ws/31102018/6173
5. Cervantes, H., Kazman, R. (2016). *Designing Software Architectures: A Practical Approach.* Boston: Addison-Wesley.
6. Shcherbakov, I. M. (2019). Proektuvannia ta analiz servernoi arkhitektury dlia vysokonavantazhenykh Web-dodatkiv. *Infokomunikatsii – suchasnist ta maibutnie,* 435–438.
7. Yefimenko, A. A., Kovalchuk, V. N., Mishyn, H. O., Suhoniak, I. I. (2018). Model dyspetcheryzatsii potokiv danykh dlia vysokonavantazhenykh veb-system. *Problemy stvorennia, vyprobuvannia, zastosuvannia ta ekspluatatsii skladnykh informatsiinykh system, 15,* 163–172.
8. Stetsyk, O., Terenchuk, S. (2021). Comparative analysis of NoSQL databases architecture. *Management of Development of Complex Systems, 47,* 78–82. doi: http://doi.org/10.32347/2412-9933.2021.47.78-82
9. Barabanov, A., Makrushin, D. (2021). Security Audit Loggingin Microservice-Based Systems: Survey of Architecture Patterns. *Voprosy Kiberbezopasnosti, 2 (42),* 71–80. doi: http://doi.org/10.21681/2311-3456-2021-2-71-80
10. Kryvenchuk, Y., Mykalov, P., Novytskyi, Y., Zakharchuk, M., Malynovskyy, Y., Řepka, M. (2019). Analysis of the Architecture of Distributed Systems for the Reduction of Loading High-Load Networks. *Advancesin Intelligent Systems and Computing,* 759–770. doi: http://doi.org/10.1007/978-3-030-33695-0_50
11. Franke, U., Johnson, P., König, J. (2013). Anarchitecture framework for enterprise IT service availability analysis. *Software & Systems Modeling, 13(4),* 1417–1445. doi: http://doi.org/10.1007/s10270-012-0307-3

Vadim Yakovenko, *Doctor of Technical Sciences, Professor, Department of Computer Science and Software Engineering, University of Customs and Finance, Dnipro, Ukraine, ORCID: https://orcid.org/0000-0001-7762-5410*

Yuliia Ulianovska, *PhD, Associate Professor, Department of Computer Science and Software Engineering, University of Customs and Finance, Dnipro, Ukraine, ORCID: https://orcid.org/0000-0001-5945-5251*

✉***Tetiana Yakovenko,*** *PhD, Associate Professor, Department of Computer Science and Software Engineering, University of Customs and Finance, Dnipro, Ukraine, e-mail: Yakovenko.t.yu@gmail.com, ORCID: https://orcid.org/0000-0003-1900-8283*

✉*Corresponding author*