UDC 681.5:519.24:621.74
DOI: 10.15587/2706-5448.2021.237760
Article type «Reports on Research Projects»

Yevhen Artamonov,
Iurii Golovach,
Vitalii Zymovchenko

USE ANALYSIS OF MICROSERVES IN E-LEARNING SYSTEM WITH MULTI-VARIANT ACCESS TO EDUCATIONAL MATERIALS

The object of research is the electronic learning system. The subject of the research is the method of using microservices in the construction of online systems. One of the most problematic areas in the development of high-load online systems is the coordination of all microservices in a single system and the distribution of the load on hardware resources at critical indicators of system utilization. This leads to the complication of the process of development, implementation and operation of the training system, as well as high requirements for the personnel who will support the operation of the system.

In the research, during the transition from the monolithic architecture of the e-learning system to the microservice architecture, the main indicators of the server hardware and the average response time to user requests were monitored. These indicators were fundamental when setting up the system as a whole and balancing the load during its operation.

The proposed method for the implementation of the system can significantly reduce the hardware requirements and reduce the response time of the system under high load conditions (from 10,000 unique users per unit of time). Also, this method greatly simplifies the development and modification of online systems that use a large number of different user roles and differentiation of levels of access to the system.

The obtained results of the approbation of the method allow to consider it an effective tool for the development of online learning systems with multivariate access to educational materials. Unlike existing monolithic architects, the proposed method allows to manage system resources and apply new settings without rebooting, which allows to ensure the continuity of system operation. As a justification for this method, options for the implementation of online training systems and load balancing settings are proposed. The management of load balancing in the microservice architecture of the implementation of online systems is based on the analysis of the load indicators of processor cores and the use of RAM by system services.

Keywords: *monolithic architecture, microservice architecture, e-learning system, multivariate access, load balancing.*

Received date: 17.03.2021
Accepted date: 26.04.2021
Published date: 31.07.2021

How to cite
Artamonov, Y., Golovach, I., Zymovchenko, V. (2021). Use analysis of microserves in e-learning system with multi-variant access to educational materials. Technology Audit and Production Reserves, 4 (2 (60)), 45–50. doi: http://doi.org/10.15587/2706-5448.2021.237760

1. Introduction

The rapid growth in demand for e-learning systems (ELS) has given a new impetus to the development of online ELS and even the resumption of work on projects that developers have abandoned due to lack of monetization. This increase in demand was especially noticeable with the transition to distance learning in universities and schools around the world during quarantine.

Experience of work with a variety of ready-made solutions for online ELS has shown in most cases the impossibility of their use in today's reality, when the system can be accessed by hundreds of thousands of users.

The main problems in the work of ready-made solutions for online ELS include:

– long-term system responses;
– code processing errors due to lack of CPU time on the server;
– restriction of database (DB) availability due to queue overflow.

In most cases, these problems are usually solved by an exponential increase in hardware capacity. Typically, online ELS is built on a monolithic architecture, so this solution only allows to postpone problems in the system for some time, rather than solve them.

Based on the results presented in [1–3] and our own research, the transition to a microservice architecture for building online ELSs was proposed. This was an analysis of key indicators of online ELS compared with the developed solution and an analysis of possible disadvantages of this solution.

Microservice architectural style involves an approach to the development of a single application as a set of small services, each of which works in its own process [1]. These services can be deployed independently of a fully automated deployment mechanism. There is a minimum of centralized management of these services, which can be written in different programming languages and use different storage technologies.

The relevance of the research topic is due to the need to coordinate the work of all microservices in the development of high-load online ELS. In this regard, *the object of research* is the e-learning system. *The subject of research* is the method of using microservices to create of online systems. *The aim of research* is a comparative analysis of monolithic and microservice architectures for the implementation of ELS.

2. Methods of research

2.1. Monolithic architecture of online e-learning systems implementation. The consideration of online ELS should be approached as a stand-alone system within a common learning system. Therefore, it is necessary to analyze the construction of online ELS through possible options for the organization of distance learning and their specifics. This approach allows to determine for what purposes a particular option may be most acceptable and under what conditions. And also to determine the specifics of the components of each of the possible options and what impact a particular option has on the organization of the educational process, organizational forms and means of learning [4].

To build online ELS, it is necessary to determine its place in the overall education system. The analysis of work [4] allowed to schematically present the model of organization of education in higher education institutions (Fig. 1).

In the proposed monolithic solutions for the implementation of online ELS for the formation of the data warehouse used a modification of the Dexter model (Fig. 2), where the mark «...» presents modules that do not participate in the formation of educational materials. The system is implemented on a monolithic architecture (MA) using a single database on the database management system (DBMS) MySQL.

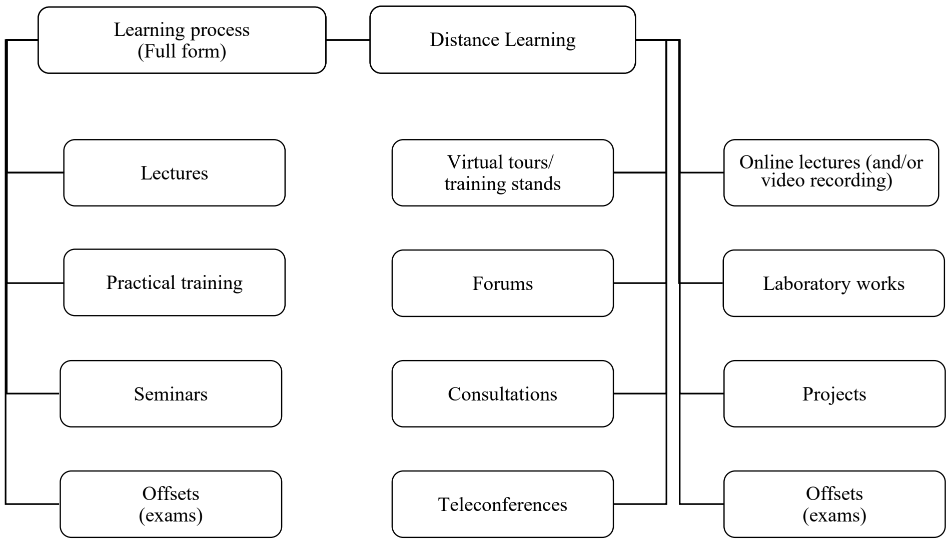


Fig. 1. Model of integration of full-time and distance learning for higher education institutions

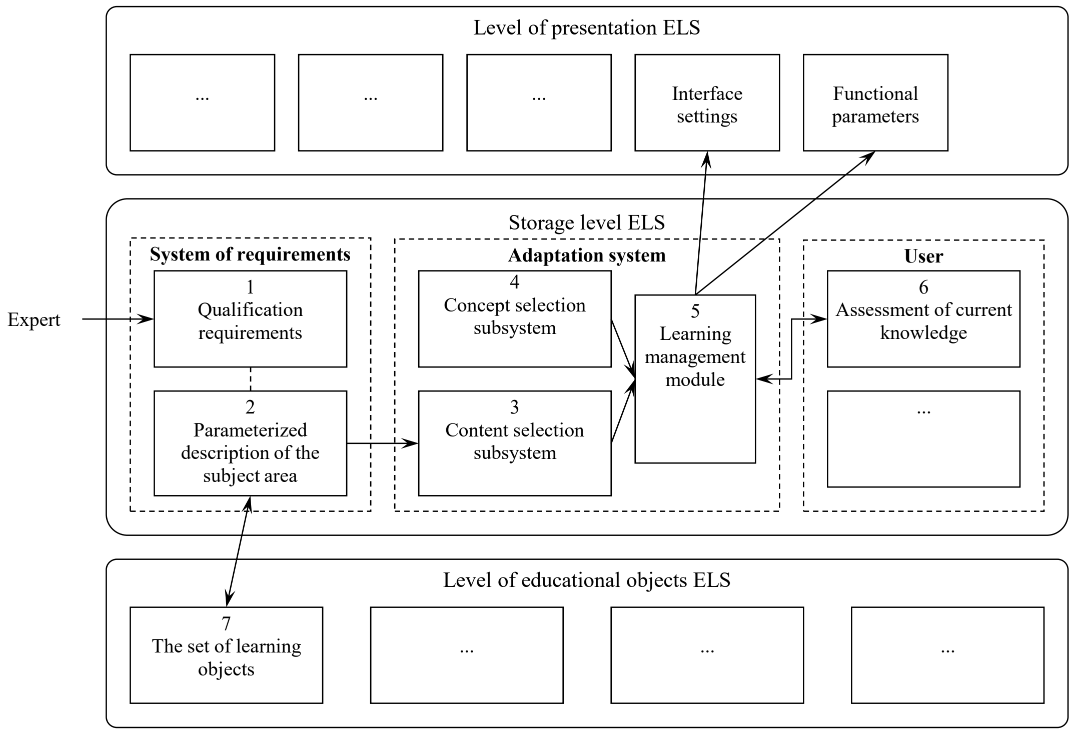

Level of presentation ELS
...
...
...
Interface settings
Functional parameters
Storage level ELS
System of requirements
1 Qualification requirements
2 Parameterized description of the subject area
Adaptation system
4 Concept selection subsystem
3 Content selection subsystem
5 Learning management module
User
6 Assessment of current knowledge
...
Expert
Level of educational objects ELS
7 The set of learning objects
...
...
...

Fig. 2. Detailing of separate subsystems of a monolithic online electronic learning system

To implement a multilevel access system and the ability to adapt the work of online ELS to the needs of users, as described in [5], and in accordance with the methods of using multi-agent systems in the construction of ELS [6, 7] was implemented:

– subsystem of selection of concepts (Fig. 2, element 3);

– content selection subsystem (Fig. 2, element 4).

As a result of the first subsystem, the parameters of the dynamic model are determined, which indicates the difference between the level of student knowledge and expectations, which are formed by the subject area (Fig. 2, element 1) and agreed with the qualification requirements (Fig. 2, element 2). The subsystem for assessing the current knowledge of students (Fig. 2, element 6) forms the main indicators of the success of the educational process.

The content selection subsystem is responsible for the formation of the list of educational objects (Fig. 2, element 7) that meet the choice of users and the requirements of the training course. The learning management module (Fig. 2, element 5) forms the interface and functional parameters of the ELS. Possible representation of educational objects is described in [8], where the approach to the formation of educational materials in the form of three scenarios (levels) of content disclosure, and [9], where the approach to the construction of interfaces, according to selected parameters.

The peculiarity of the implementation of multivariate access in a monolithic architecture involves constant monitoring of the user's access level to give it the appropriate rights, which provide a level of access to data.

2.2. Microservice architecture for the implementation of online electronic learning systems. When creating online ELS for microservice architecture (MSA), it is necessary to break the monolithic system into a set of separate services.

During the development of the system, the main properties of microservices that need to be provided for the functioning of the entire complex of online ELS were identified:

– small – one service can develop one team of no more than 6 employees, one team can develop 6–10 services, one service can be completely rewritten by one team for one Agile iteration [1];

– independent – implementation of High Cohesion and Low Coupling patterns, when each microservice works in its own process and therefore must explicitly mark its API;

– builds around business needs and uses a limited context – areas of responsibility of the microservice, formulated around a certain business need, the more compact it is, the easier it is to create a new microservice [10, 11];

– use of the Design for failure approach – one of the most critical places in microservice architecture is the need to develop code for a distributed system, the components of which interact through the network. And the network is unreliable by nature. The network may simply fail, may be slow, may suddenly stop skipping any type of message because the firewall settings have changed. Therefore, microservices must properly handle the various failure options, be able to wait, be able to return to normal operation when restoring the counterparty;

– limited centralization (many databases, many contexts, which implies the impossibility of synchronous interaction of several microservices). This problem is solved by abandoning the constant consistency of data, where possible errors are either handled by a more complex architecture, or through monitoring data;

– iterative development (gradual introduction of microservices with testing of their interaction).

The transition to the new architecture was implemented by a PHP programming language and MySQL database. It was decided to allocate the following servers for the implementation of microservices (Fig. 3):

– server of training courses (TC);

– server of virtual laboratories (VL);

– server of personal accounts (PA);

– server of knowledge testing system (KTS);

– server of user performance evaluation (UPE);

– server of analytics;

– server of data processing center (DPC);

– servers of data import for organizing access to services.

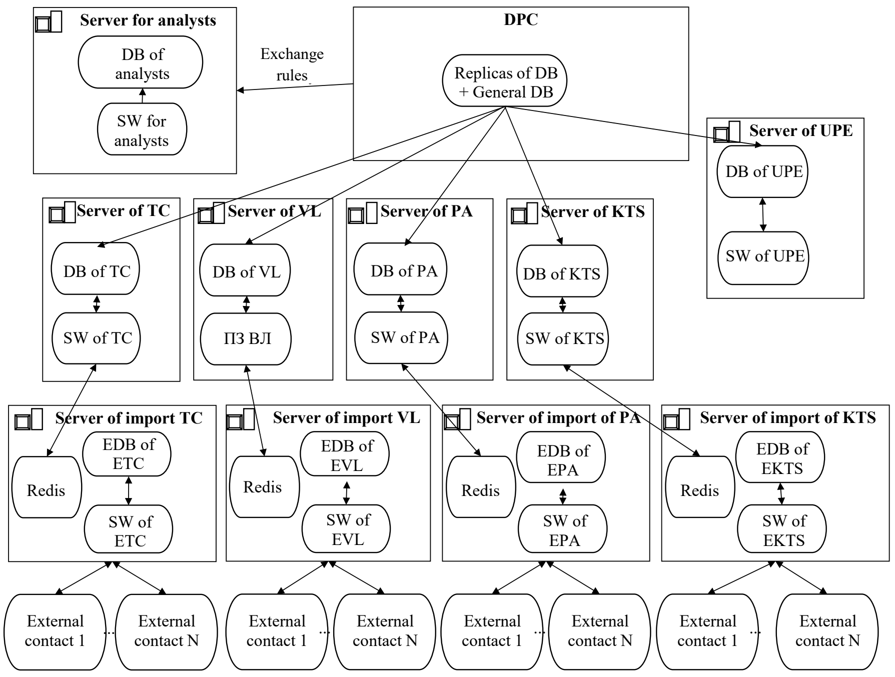


Fig. 3. Scheme of the developed online e-learning system based on microservice architecture

For external systems the mark «E» is used. For example, EPA is an external system of personal accounts, which uses EDB (external database) of personal accounts.

Multivariate access provides a variety of interfaces for different user groups (for example, administrators, teachers and students when using microservices Personal account, Training courses, Knowledge testing system will see different representations of information according to their level of access.

Each of the microservices is located on a separate virtual server, which receives the necessary resource in accordance with the general principles of load balancing on the system. The load balancing module is configured at the server core level and provides for the use of decision-making algorithms based on the state of the system [12–14] according to the following input parameters: type of protocols for network balancing.

2.3. Description of the hardware base for research. The research was conducted on two versions of dedicated servers:

1) based on Intel Xeon E3-1230 v5/6, with a frequency of 3.2 GHz, 6 cores and 6 GB of RAM;

2) based on Intel Xeon 2x E5-2690 v2, with a frequency of 3.0 GHz, the possibility of multithreading – 40 threads, 20 cores and 256 GB of RAM.

The research was conducted on the same type of training courses (100 training courses, among which were experimental courses for training UAV pilots (unmanned aerial vehicle) [15, 16]). The load was implemented using a software simulation system. The following load has been generated: 200, 1000, 10000, 50,000 users. The main characteristics for the analysis were: the average percentage of CPU cores, the average value of RAM usage, the average server response time, the average database response time, the average page load time.

Fig. 4 presents a window for analyzing the percentage of CPU cores for the case of a monolithic architecture, a server of variant 1 and a load of 180–200 simultaneous users (Fig. 5).

The use of microservice architecture has shown a number of problems that can be divided into two main groups:

– organizational:

a) maintenance in the agreed state of hundreds of microservices, which are constantly and unpredictably updated;

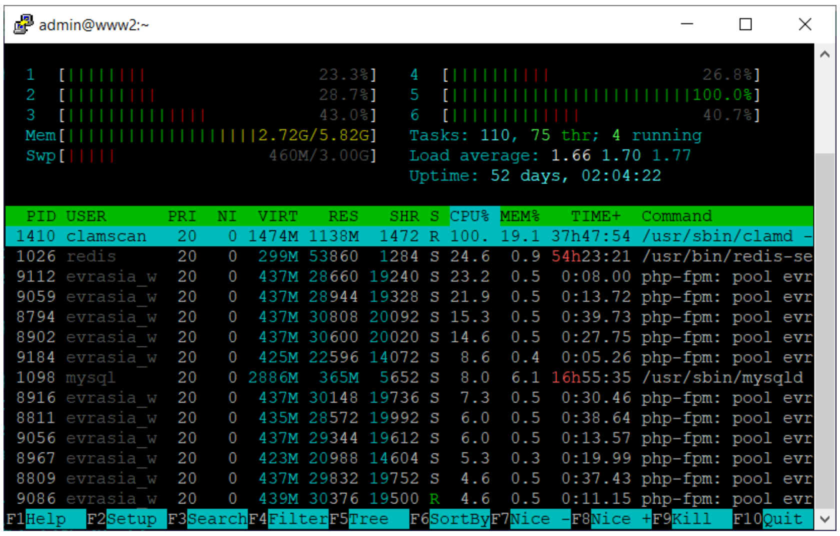
b) setting up access to environments for each engineer of each team;

c) defining a team for writing integration tests. As a solution to these problems is the practice of using Agile and DevOps systems development techniques;

– architectural: continuous operation of the system during the transition from a monolithic architecture, where everything is synchronous, coherent and unified, to a distributed microservice architecture based on many small elements, which must take into account possible data inconsistencies.

3. Research results and discussion

The results of the research, which was conducted for 2 hours on each of the servers, are presented in Table 1 (for the server with Variant 1 configuration) and in Table 2 (for the server with Variant 2 configuration) for each type of architecture.



```
admin@www2:~

  1  [||||||||                23.3%]   4  [||||||||||                26.8%]
  2  [|||||||||               28.7%]   5  [||||||||||||||||||||||||100.0%]
  3  [||||||||||||||          43.0%]   6  [|||||||||||||             40.7%]
  Mem[|||||||||||||||||||2.72G/5.82G]   Tasks: 110, 75 thr; 4 running
  Swp[|||||             460M/3.00G]   Load average: 1.66 1.70 1.77
                                      Uptime: 52 days, 02:04:22

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 1410 clamscan   20   0 1474M 1138M  1472 R 100. 19.1 37h47:54 /usr/sbin/clamd -
 1026 redis      20   0  299M 53860  1284 S 24.6  0.9 54h23:21 /usr/bin/redis-se
 9112 evrasia_w  20   0  437M 28660 19240 S 23.2  0.5  0:08.00 php-fpm: pool evr
 9059 evrasia_w  20   0  437M 28944 19328 S 21.9  0.5  0:13.72 php-fpm: pool evr
 8794 evrasia_w  20   0  437M 30808 20092 S 15.3  0.5  0:39.73 php-fpm: pool evr
 8902 evrasia_w  20   0  437M 30600 20020 S 14.6  0.5  0:27.75 php-fpm: pool evr
 9184 evrasia_w  20   0  425M 22596 14072 S  8.6  0.4  0:05.26 php-fpm: pool evr
 1098 mysql      20   0 2886M  365M  5652 S  8.0  6.1 16h55:35 /usr/sbin/mysqld
 8916 evrasia_w  20   0  437M 30148 19736 S  7.3  0.5  0:30.46 php-fpm: pool evr
 8811 evrasia_w  20   0  435M 28572 19992 S  6.0  0.5  0:38.64 php-fpm: pool evr
 9056 evrasia_w  20   0  437M 29344 19612 S  6.0  0.5  0:13.57 php-fpm: pool evr
 8967 evrasia_w  20   0  423M 20988 14604 S  5.3  0.3  0:19.99 php-fpm: pool evr
 8809 evrasia_w  20   0  437M 29832 19752 S  4.6  0.5  0:37.43 php-fpm: pool evr
 9086 evrasia_w  20   0  439M 30376 19500 R  4.6  0.5  0:11.15 php-fpm: pool evr
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Fig. 4. Screenshot of the server CPU kernel load monitoring system window

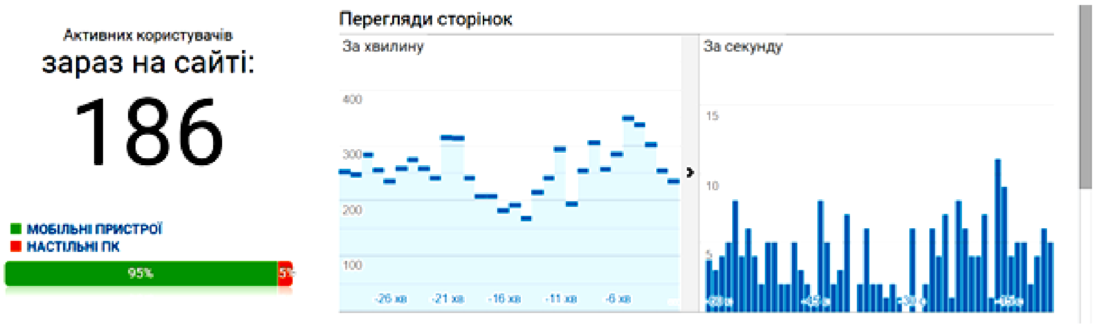


Fig. 5. Screenshot of the Google Analytics window with the number of simultaneous users per system per second

In the Table 1, 2 the following notation is used:
– *CPU* – the average percentage of CPU cores;
– *Mem* – the average value of RAM usage;
– t_a – the average response time of the server;
– t_{aDB} – the average response time of the database;
– t_d – average page load time.

It should be noted that the implementation of MSA on the server under option 1 is complicated by the small number of processor cores, because this architecture involves the deployment of 11 virtual servers. This problem was solved by distributing user flows by services, so the average response time of the pages was within acceptable limits. When estimating the average access time to the system built on the MSA, the average access time to all microservices is used without taking into account the «downtime» of a microservice per unit time.

Table 1

The average performance of the server with the configuration 1

Number of users	200				
Parameters	*CPU*, %	*Mem*, GB	t_a, sec	t_{aDB}, sec	t_d, sec
MA	15.12	2.1	0.123107	0.052311	1.245121
MSA	75.23	4.45	0.156196	0.012423	1.034537
Number of users	1000				
Parameters	*CPU*, %	*Mem*, GB	t_a, sec	t_{aDB}, sec	t_d, sec
MA	45.67	4.15	0.234235	0.254422	1.245176
MSA	78.45	4.67	0.156185	0.109924	1.034553
Number of users	10000				
Parameters	*CPU*, %	*Mem*, GB	t_a, sec	t_{aDB}, sec	t_d, sec
MA	87.89	5.07	2.124423	0.852391	5.151433*
MSA	85.67	5.11	0.915622	0.212415	2.146729
Number of users	50000				
Parameters	*CPU*, %	*Mem*, GB	t_a, sec	t_{aDB}, sec	t_d, sec
MA	99.91	5.97	5.523212*	2.151213*	14.2312*
MSA	92.14	5.34	1.556898	0.533567	4.241383

Note: * – possible error ERR_CONNECTION_TIMED_OUT

Table 2

Average performance of the server with the configuration 2

Number of users	200				
Parameters	*CPU*, %	*Mem*, GB	t_a, sec	t_{aDB}, sec	t_d, sec
MA	5.11	2.21	0.121212	0.005111	0.921255
MSA	18.53	6.87	0.096155	0.002177	0.321997
Number of users	1000				
Parameters	*CPU*, %	*Mem*, GB	t_a, sec	t_{aDB}, sec	t_d, sec
MA	25.67	14.15	0.164888	0.053234	1.092451
MSA	20.15	17.97	0.116122	0.009664	1.014553
Number of users	10000				
Parameters	*CPU*, %	*Mem*, GB	t_a, sec	t_{aDB}, sec	t_d, sec
MA	59.34	65.07	0.944493	0.151392	2.771472*
MSA	32.17	95.11	0.225677	0.011414	2.016729
Number of users	50000				
Parameters	*CPU*, %	*Mem*, GB	t_a, sec	t_{aDB}, sec	t_d, sec
MA	93.61	250.97	3.233233*	1.111244*	10.2312*
MSA	44.14	115.34	0.256898	0.093544	2.241383

Note: * – possible error ERR_CONNECTION_TIMED_OUT

The research results showed:
– for loads up to 1,000 users there is no need to switch to MSA;
– with an expected load of 10,000 users, the monolithic architecture uses the resources of a sufficiently powerful server, but still cannot ensure the smooth operation of the system. The most common error in the system was exceeding the response timeout;
– the program with MSA can handle more than 50,000 users, due to the possible use of load balancing at levels L4 and L7. This is due to the distributed processing of requests and data caching of services.

It should be noted that the research was conducted on the basis of own software solutions and optimization of microservices by speed. That is one of the possible solutions. To finally confirm the correctness of the choice of MSA should compare the work of the developed system with common software systems, which can be separate learning content management systems (for example, Moodle) or additional topics to standard content management systems (for example, LMS WordPress Theme). To compare the work of distributed ELS with implemented on the basis of MSA, it is necessary to provide them with equal conditions:
– identical in content courses;
– the same intensity of access;
– optimization of settings of the distributed ELS (on adjustment of a cache, queues and possible distribution on separate servers of a DBMS and the software engine of system).

4. Conclusions

The research confirmed the expected result from the transition to MSA online ELS, which was the feasibility of using MSA in heavily loaded systems. But in addition, positive results were obtained regarding the possible use of MA on powerful servers with multi-threaded data processing, but up to 10,000 users present on the resource at the same time. The use of MSA is especially appropriate in the case of multivariate access to online ELS resources, when there is a distribution of users by different microservices.

During the development of the MSA system, the main properties of microservices that need to be provided for the functioning of the entire complex of online ELS were identified:
– small size of microservices;
– independence;
– the need to implement the business needs using a limited context;
– the need to use the Design for failure approach;
– limited centralization;
– iterative development.

During the development of the MSA online system, a number of organizational and architectural problems were identified. Solving these problems allowed to use of the simultaneous service of more users through the use of low-power server and to lay a margin to increase the maximum number of users simultaneously present on the resource.

References

1. Di Francesco, P., Lago, P., Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software, 150*, 77–97. doi: http://doi.org/10.1016/j.jss.2019.01.001

2. Auer, F., Lenarduzzi, V., Felderer, M., Taibi, D. (2021). From monolithic systems to Microservices: An assessment framework. *Information and Software Technology*, 137. doi: http://doi.org/10.1016/j.infsof.2021.106600
3. Dragoni, N., Lanese, I., Larsen, S. T., Mazzara, M., Mustafin, R., Safina, L. (2017). Microservices: How to make your application scale. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 95–104. doi: http://doi.org/10.1007/978-3-319-74313-4_8
4. Belonozhko, M. L., Abramovskiy, A. L. (2014). Distantsionnaya model obucheniya studentov sovremennogo vuza na baze elektronnoy obrazovatelnoy sredy. *Fundamentalnye issledovaniya, 5-3,* 620–624. Available at: http://www.fundamental-research.ru/ru/article/view?id=33931
5. Chatzopoulou, D. I., Economides, A. A. (2010). Adaptive assessment of student's knowledge in programming courses. *Journal of Computer Assisted Learning, 26 (4),* 258–269. doi: http://doi.org/10.1111/j.1365-2729.2010.00363.x
6. Melesko, J., Kurilovas, E. (2016). Personalised intelligent multi-agent learning system for engineering courses. *2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*. doi: http://doi.org/10.1109/aieee.2016.7821821
7. Kristensen, T., Dyngeland, M. (2015). Design and Development of a Multi-Agent E-Learning System. *International Journal of Agent Technologies and Systems, 7 (2),* 19–74. doi: http://doi.org/10.4018/ijats.2015040102
8. Artamonov, E. B., Zholdakov, O. O. (2010). Concept of creating a software environment for automated text manipulation. *Proceedings of National Aviation University, 44 (3),* 111–115. doi: http://doi.org/10.18372/2306-1472.44.1916
9. Artamonov, Ye. B. (2017). Rozrobka pidkhodu do formuvannia adaptyvnykh navchalnykh resursiv. *Visnyk inzhenernoi akademii Ukrainy, 1,* 239–243.
10. Gnatyuk, S., Sydorenko, V., Polihenko, O., Sotnichenko, Y., Nechyporuk, O. (2020). Studies on the disasters criticality assessment in aviation information infrastructure. *CEUR Workshop Proceedings,* 282–296.
11. Gnatyuk, S. (2019). Multilevel Unified Data Model for Critical Aviation Information Systems Cybersecurity. *2019 IEEE 5th International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD),* 242–247. doi: http://doi.org/10.1109/apuavd47061.2019.8943833
12. Al-Azzeh, J., Litvinenko, A., Kucherov, D., Kashkevych, I.-F., Bagisov, Z. (2020). Methods for obtaining of management decisions during evaluating the controlled parameters by qualitative categories. *CEUR Workshop Proceedings, 2654,* 402–420.
13. Litvinenko, A. (2020). Algorithms for Solution Inference Based on Unified Logical Control Models. *Cybernetics and Systems Analysis, 56 (2),* 187–194. doi: http://doi.org/10.1007/s10559-020-00234-9
14. Nechyporuk, O., Kashkevich, I.-F., Suprun, O., Nechyporuk, V., Poburko, O., Apenko, N. (2020). Identification of Combinations of Faults in Multilevel Information Systems. *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH),* 76–81. doi: http://doi.org/10.1109/memstech49584.2020.9109465
15. Kucherov, D., Sushchenko, O., Kozub, A. (2019). Operator Training for Unmanned Aerial Vehicles Control. *2019 IEEE 5th International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD),* 31–34. doi: http://doi.org/10.1109/apuavd47061.2019.8943918
16. Kucherov, D., Sushchenko, O., Kozub, A., Petrov, A. (2019). Assessment of operator-pilot training in conflict situations. *CEUR Workshop Proceedings.*

✉***Yevhen Artamonov,*** *PhD, Department of Computerized Control System, National Aviation University, Kyiv, Ukraine, e-mail: eart@ukr.net, ORCID: https://orcid.org/0000-0002-9875-7372*

Iurii Golovach, *Company «Squad», Kyiv, Ukraine, ORCID: https://orcid.org/0000-0001-9872-8144*

Vitalii Zymovchenko, *Researcher, Ukrainian Research Institute of Special Equipment and Forensic Science of the Security Service of Ukraine, Kyiv, Ukraine, ORCID: https://orcid.org/0000-0001-9834-8547*

✉*Corresponding author*