CONTROL PROCESSES

This study's object is the process of managing (balancing) the load on the computing nodes in a server cluster of information systems. The task addressed is to enable optimal (even) distribution of server resources within a cluster system.

A simulation model of the process that distributes the load on computing nodes of a server cluster has been built, based on an improved model of the optimal use of server resources in a cluster based on Nash equilibrium and an improved method of adaptive load balancing in cluster systems according to Nash equilibrium.

The simulation model has been constructed on the basis of fuzzy logic theory to determine the feasibility of decomposing tasks into subtasks, as well as game theory, in particular Nash equilibrium, to determine the optimal distribution of tasks/subtasks across cluster system servers for their parallel processing.

The model built makes it possible to improve the efficiency (uniformity) of server resource distribution by an average of 13% compared to the classical load management method based on Nash equilibrium, by 61% with the Round Robin method, and by 63% with the Least Connection method throughout the entire process of cluster operation. This, in turn, allows for a more than 2-fold increase in the number of processed tasks from client requests compared to the above load balancing methods.

In addition, the impact of the load balancing process on the processing time of tasks/subtasks by the cluster system servers was estimated. Based on the results of simulation modeling, it can be concluded that the application of the devised model does not exceed the total allowable processing time (no more than 315 ms) of tasks/subtasks from client requests compared to existing load balancing methods

Keywords: simulation model, server cluster, optimization, load management, resilience

UDC 004.451:519.8
DOI: 10.15587/1729-4061.2025.340603

CONSTRUCTION OF A SIMULATION MODEL FOR MANAGING LOAD ON COMPUTING NODES IN A SERVER CLUSTER BASED ON THE THEORY OF FUZZY LOGIC AND NASH EQUILIBRIUM

Yevhenii Neroznak
Corresponding author
Doctor of Philosophy in Information Systems and Technologies, Senior Lecturer*
E-mail: eugene.neroznak@viti.edu.ua
Olexandr Trotsko
PhD, Associate Professor, Head of Department*
Vitalii Fesokha
Doctor of Philosophy in Information Systems and Technologies, Associate Professor
Department of Scientific and Organizational**
Dmytro Balan
Lecturer*
Robert Bieliakov
PhD, Associate Professor, Deputy Head of Department*
*Department of Automated Control Systems**
**Kruty Heroes Military Institute of Telecommunications and Information Technology
Knyaziv Ostrozkyh str., 45/1, Kyiv, Ukraine, 01011

Received 21.08.2025
Received in revised form 08.09.2025
Accepted 16.09.2025
Published 31.10.2025

How to Cite: *Neroznak, Y., Trotsko, O., Fesokha, V., Balan, D., Bieliakov, R. (2025). Construction of a simulation model for managing load on computing nodes in a server cluster based on the theory of fuzzy logic and Nash equilibrium. Eastern-European Journal of Enterprise Technologies, 5 (3 (137)), 6–17. https://doi.org/10.15587/1729-4061.2025.340603*

1. Introduction

Within the framework of implementing the provisions of the Strategic Defense Bulletin of Ukraine [1] and the Strategy for the Development of the Defense Industrial Complex of Ukraine [2], the leadership of the Armed Forces of Ukraine is working to design an effective system of operational control, communications, intelligence, and surveillance (C4ISR), which would meet NATO standards, and enable its integration with the Unified Defense Resource Management System. In militarily developed countries of the world, C4ISR systems evolve according to the principles of the network-centric concept of conducting combat operations. Such principles provide for the creation of a single information space for the exchange of information about the situation on the battlefield to ensure full awareness of relevant officials in the process of making decisions on the management of troops (forces) and weapons. At the same time, when implementing such systems into the existing structure of the troop and weapons management system, it is necessary to take into account the peculiarities of the functioning of military systems, internal processes, and tasks defined for individual subsystems.

It is obvious that the information circulating in such military information systems is of critical importance for the effective management of troops and weapons, making management decisions on the performance of assigned tasks, etc., at the same time, the processes associated with its collection, processing, storage, and analysis have a direct and significant impact on the effective control over computing resources within their technical infrastructure.

In the context of the rapid growth of data processing volumes, the increase in the complexity of both client requests and military information systems themselves, as well as enabling their uninterrupted operation, the task of managing the load on the technical component becomes critically important. The optimal distribution of tasks between the computing resources of the server cluster in military information systems has a decisive impact on their productivity, reliability, stability, survivability, and readiness to function under various influences, both external and internal.

Scientific achievements in this area open up ways to improve the efficiency of the functioning of information systems and computing processes. Research and development of new load management methods is becoming a key factor in the proper functioning and evolution of such systems. This makes scientific research on improving the efficiency of optimal use of server resources in military cluster systems relevant.

2. Literature review and problem statement

In [3], a multiple regression-based search algorithm (MRBS) is proposed for optimal server and route selection in a data center network. The server selection is performed in the case of high loads, such as sudden spikes in the intensity of incoming requests, unpredictable traffic patterns. When selecting a suitable server based on the above parameters, regression analysis is used, which has a high dependence on the quality of training, low verification, and robustness, as well as insufficient guarantee of finding the global optimum. At the same time, the proposed algorithm is aimed at reducing the delay and processing time and response to the request. Insufficient attention is paid to the optimal use of server resources.

In [4], a mathematical model of a load balancing system is proposed, in which the load balancer is described using a queuing system. The indicators of the central processor, network card, and RAM utilization are taken into account. However, the proposed model does not take into account the probabilistic distribution of tasks across cluster servers for more optimal use of server resources but only calculates the probability of pushing packets with a lower service priority out of the queue. That is, some tasks will not be processed by servers. At the same time, the application of the model is considered without taking into account resource limitations. The possibility of decomposing tasks for their parallel processing is also not taken into account. The issue of reducing the response time to a user request is mainly considered, and the issue of optimal use of server resources is not investigated.

In [5], a multi-level system based on the theory of fuzzy logic is proposed for assessing effective load balancing in cloud data centers. The input linguistic variables are weighted indicators of the data center, the number of virtual machines, and Internet of Things tools. However, it should be noted that the choice of an effective method (algorithm) of load balancing depends not on the general weighted indicators (parameters) of the data center but on a wider set of technical parameters of the data center. However, insufficient attention is paid to taking into account the indicators of the use of server and virtual resources of the data center, only quantitative indicators. When applying the theory of fuzzy logic for this class of problems, the global optimum is not always achieved, the efficiency of load balancing may be within the local optimum. The resource limitation factor is not taken into account either.

In [6], a load balancing algorithm for homogeneous cluster systems is proposed, which is based on the maximum entropy method. It takes into account the calculation of maximum entropy probabilities without any restrictions on the execution of tasks by cluster servers. It should be noted that tasks caused by client requests may require different amounts of server resources for processing. Therefore, it is not always possible to achieve the maximum uniform load of cluster servers. In addition, in the case of achieving uniform load (maximum entropy), it will not be stable because more complex requests will be executed by one server and with a longer processing time compared to simple requests. This is explained by the fact that the possibility of decomposition of complex tasks is not taken into account. Attention is paid to reducing the time of task execution and response to the request, increasing the cluster performance, but not enough attention is paid to the uniform use of server resources. At the same time, the cited work investigates the operation of this algorithm in a homogeneous cluster while most information and communication systems can be deployed and function on the basis of heterogeneous cluster systems.

In work [7], an improved *Round Robin* load balancing algorithm (*EWRR*) is proposed for distributing incoming requests and tasks between servers of a cloud cluster. The efficiency assessment was carried out in comparison with the corresponding modifications of the Round Robin algorithm, as well as other algorithms. The attention of the researchers focuses on reducing the response time to the request, downtime, task migration, and delayed tasks by determining the required number of virtual machines to improve the above parameters. This need is caused by the uneven load of virtual machines since the load indicators of the virtual machine components are not fully taken into account by the algorithm. In addition, the factor of limited cluster resources and their optimal use during task processing is not taken into consideration.

In [8], an approach to load balancing is proposed, which is based on a modified genetic algorithm. The approach is aimed at reducing the time of execution of tasks by servers of a cloud cluster. This approach includes a processing vector, which consists of the number of instructions per second that the server can execute, the cost of executing the instruction, and the cost of delaying the processing of the user request. However, the authors do not pay enough attention to the factor of limited server resources, and the issue of optimizing the consumption of cluster resources for performing tasks caused by client requests is not considered.

In [9], an improved load balancing algorithm based on *Weighted Round Robin* using machine learning methods is proposed. This algorithm includes a machine learning model based on a decision tree to predict the most suitable server for performing tasks. It takes into account the task profile and the current state of the servers. However, this algorithm is aimed at finding the shortest task processing time and subsequently reducing the response time to the request. The authors did not pay enough attention to increasing the efficiency of using server resources with an acceptable task processing time.

However, scientific works [3–9] are united by the fact that they report the results of applying different approaches to reduce the response time to the request and improve the efficiency of the functioning of the software component of information systems. In most cases, this leads to an imbalance in the load of the cluster system servers and subsequent overloading of both individual servers and the entire cluster system as a whole. To some extent, the indicators of optimal (fair) use

of available resources are neglected. This is explained by the fact that the authors do not pay enough attention to ensuring the optimal use of server component resources in the context of reducing the load distribution dispersion index between cluster system servers. The process of processing tasks by cluster servers caused by client requests has a significant impact on the stable and efficient functioning of information systems, especially those integrated into the military industry. Therefore, the tasks associated with ensuring the stable functioning of the server infrastructure of information systems are quite important and relevant.

All this gives grounds to argue that it is advisable to conduct research on increasing the efficiency of load management on server nodes within the cluster system in the context of reducing the load distribution dispersion index between cluster system servers.

In accordance with [10, 11], the use of methods based on the theory of games – Nash equilibrium, makes it possible to effectively find the optimal solution for using cluster resources under conditions of limited resources and acceptable time for processing requests. This is explained by the growing requirements for ensuring efficiency, stability, survivability, productivity, and readiness of military information systems to function in the languages of modern information environment.

It is on the basis of the devised scientific and methodological apparatus [10, 11] that it is advisable to build a simulation model of load management on computing nodes in a server cluster as a scientific task.

3. The aim and objectives of the study

The purpose of our study is to improve the efficiency of load distribution on server nodes in the cluster system by reducing the load dispersion index. This will allow for a more even use of the cluster server resources during the operation of information systems under conditions of dynamic loads and resource constraints.

To achieve this aim, the following objectives were accomplished:

– to design a module for determining the feasibility of task decomposition by client requests initiated based on the theory of fuzzy logic;

– to design a module for determining the optimal distribution of tasks/subtasks across servers in the cluster system for their parallel processing based on the Nash equilibrium.

4. The study materials and methods

The object of our study is the process of managing (balancing) the load on the computing nodes in the server cluster of information systems.

The principal hypothesis of the study assumes that the integration of fuzzy logic methods and the concept of game theory – Nash equilibrium into the load management process would make it possible to enable adaptive and optimal distribution of computing resources of the server cluster, which could lead to a decrease in the imbalance of the server load and an increase in the stability of the cluster system.

The following assumptions and simplifications were adopted in our study. The process of handling client requests q_i by a finite set of servers S of the cluster system is considered. Each request q_i initiates a task that must be performed. Client-server interaction is represented by a non-cooperative, dynamic, symmetric game G with a non-zero sum since clients do not know about each other but compete at the level of occupying server resources for a certain time in order to process their requests [11].

Characteristics of cluster nodes: duplicate; database: distributed.

Cluster node status assessment: a generalized indicator of the free space of the current l_{Si} load level (processor, RAM, network interface).

Game cycle G: for each client request:

– start of game G: client request received by the load balancer;

– end of game G: task performed, generated by client query on the mixed profile of cluster system servers, free resources of which were in Nash equilibrium.

Load balancing priorities: ensuring maximum functional stability and survivability of the cluster system without increasing the time for processing client requests.

The time of delivery of the client query for processing to the cluster system and the return time are not taken into account.

The simulation model of the process of distributing the load to the computing nodes in the server cluster by the load management subsystem is divided into two main modules:

– determining the feasibility of decomposition of tasks initiated by client queries;

– determining the optimal distribution of tasks/subtasks across cluster system servers for their parallel processing based on Nash equilibrium.

The essence of simulation modeling in the context of this study is to reproduce the process of load distribution on the computing nodes in the server cluster based on the results obtained in the course of scientific research reported in [10, 11].

The module for determining the feasibility of task decomposition by client-initiated requests based on the fuzzy logic apparatus is proposed to be designed using the Fuzzy Logic Toolbox™ package, which provides MATLAB® functions and the Simulink® block [12, 13]. The functional diagram of the module is shown in Fig. 1, where x_1 ... are input variables, y is the output variable.

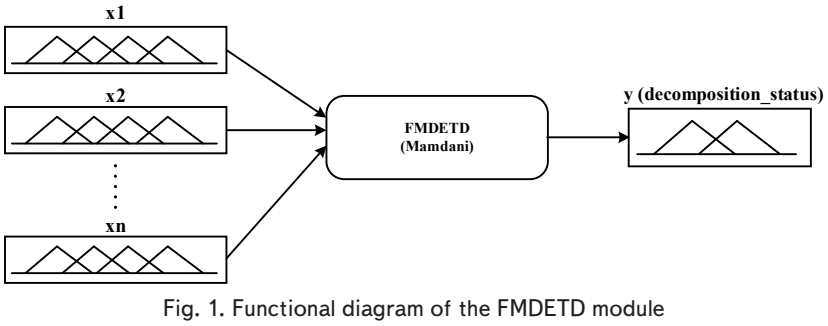

Fig. 1. Functional diagram of the FMDETD module

According to the model proposed in [10], the design of the module involves the following:

1) forming a set of statistical data on the parameters of client queries and the current level of load on the cluster server components during testing;

2) determining the class of client queries by types to be processed:

– simple – requests that do not require significant computing resources and/or requests that are impractical to decompose (video/audio streaming);

– complex – requests that require significant computing resources (for example, related to the processing of large data sets (Big Data), using information and analytical systems, complex requests to the database cluster, etc.);

3) determining the type of membership functions to describe the ranges of v alues of the studied parameters and the power of term sets for input and output linguistic variables;

4) determining input and output linguistic variables;

5) defining the algorithm for fuzzy logical inference based on the knowledge base: Mamdani;

6) defining the defuzzification method to obtain the value of the complexity weight indicator w_{qi} of the received query q_i;

7) defining the weight coefficient for all fuzzy rules;

8) preparing the test data format for the Fuzzy Logic Toolbox™ software;

9) creating fuzzy production rules for the knowledge base based on the generated set of statistical data;

10) writing a script for correct data entry from the generated set of statistical data for the purpose of their further analysis by the Fuzzy Logic Toolbox™ library [12, 13];

11) conducting experimental studies on identifying complex queries for their decomposition by the devised simulation model, the functioning of which is based on the application of the improved model [10]. Thus, the task of determining the feasibility of decomposing the resulting task into 2 subtasks is reduced to the following analytical expression (1)

$$\begin{aligned} &X=(x_1,x_2,...,x_n)\to \\ &\to y=\left(a_1^{jk_j},a_2^{jk_j},...,a_n^{jk_j}\right)\in D=\left(d_1,d_2,...,d_n\right), \\ &i=\overline{1,n},\ j=\overline{1,m}, \end{aligned} \tag{1}$$

where X is the feature space of the description of all received tasks q_i by the balancer (network protocol; size of the resource to be processed; indicators of the processor/RAM and network card utilization of the cluster node), based on the analysis of which their type is determined (simple, complex); y is the linguistic description of the decision (conclusion) $d_j \in D$ about the type of task for some fixed vector of values of the feature space $(x_1^*, x_2^*, ..., x_n^*)$ of task q_i; jk_j are the numbers of combinations of fuzzy terms a_i of feature space X of the received tasks q_i, which correspond to value d_j.

The functional dependence between the studied feature space of the received tasks and the corresponding decision made on the need for their decomposition is formalized in the form of the following system of fuzzy logical statements (rules) of the type "IF-THEN, ELSE" (2):

$$\begin{aligned} &\text{IF } (x_1=a_1^{11}) \text{ AND } (x_2=a_2^{11}) \text{ AND } \ldots \text{AND } (x_n=a_n^{11}) \\ &\text{OR } (x_1=a_1^{12}) \text{ AND } (x_2=a_2^{12}) \text{ AND } \ldots \text{AND } (x_n=a_n^{12}) \\ &\text{OR } (x_1=a_1^{1k_1}) \text{ AND } (x_2=a_2^{1k_1}) \text{ AND } \ldots \text{AND } (x_n=a_n^{1k_1}) \\ &\text{THEN } y=d_1, \text{ OTHERWISE} \\ &\text{IF } (x_1=a_1^{21}) \text{ AND } (x_2=a_2^{21}) \text{ AND } \ldots \text{AND } (x_n=a_n^{21}) \\ &\text{OR } (x_1=a_1^{22}) \text{ AND } (x_2=a_2^{22}) \text{ AND } \ldots \text{AND } (x_n=a_n^{22}) \\ &\text{OR } (x_1=a_1^{2k_2}) \text{ AND } (x_2=a_2^{2k_2}) \text{ AND } \ldots \text{AND } (x_n=a_n^{2k_2}) \\ &\text{THEN } y=d_2, \text{ OTHERWISE} \\ &\text{IF } (x_1=a_1^{m1}) \text{ AND } (x_2=a_2^{m1}) \text{ AND } \ldots \text{AND } (x_n=a_n^{m1}) \\ &\text{OR } (x_1=a_1^{m2}) \text{ AND } (x_2=a_2^{m2}) \text{ AND } \ldots \text{AND } (x_n=a_n^{m2}) \\ &\text{OR } (x_1=a_1^{mk_m}) \text{ AND } (x_2=a_2^{mk_m}) \text{ AND } \ldots \text{AND } (x_n=a_n^{mk_m}) \\ &\text{THEN } y=d_m. \end{aligned} \tag{2}$$

Thus, as a decision on the need to decompose the obtained task q_i into symmetric subtasks $sq_j^{q_i}$ based on a fixed vector of values of the analyzed features, the result with the maximum value obtained as a result of convolution of the membership functions of the terms of the fuzzy rules for describing tasks q_i is selected.

The module for determining the optimal distribution of tasks/subtasks across cluster system servers for their parallel processing based on Nash equilibrium is proposed to be developed using the Numpy and Nashpy libraries in the Python programming language [14, 15].

Within the framework of this study, to simulate the distribution of tasks/subtasks, it is proposed to develop the following components by writing program code:

– 4 servers in the form of lists (List);

– a generator of tasks/subtasks with different weights of the complexity of their processing;

– functions for distributing tasks/subtasks across cluster servers using the Round Robin (RR), Least Connection (LC), Nash Equilibria (NE), and Nash Equilibria with Task Splitting (NETS) methods;

– functions for calculating the value of dispersion indicator D_L of load distribution between cluster servers for each of the above methods;

– functions for plotting server load charts and their dispersion for each of the above methods;

– function for plotting a general chart of the dispersion indicators of load distribution between cluster servers for each of the above methods;

– function for calculating the average balancing time and the processing time of tasks/subtasks by cluster system servers for the selected load management (balancing) methods.

The procedure for finding the optimal distribution of the use of a set of cluster system servers $(s_i^* \times \cdots \times s_n^*)$ based on the Nash equilibrium (*NE*) under conditions of fuzziness of the values of the studied space of features of the received tasks and the generalized indicator of server workload is carried out based on the mathematical model proposed in [11] (3), (4):

$$\begin{aligned} &NE: p\left(sq_1^{q_i}\to S\right)= \\ &=\begin{cases} p_{sq_1^{q_i}}(s_i)=p_{sq_2^{q_i}}(s_i)*u_{11}\pm\left(\sum\limits_{s_i\in S}p_i(s_i)-p_{sq_2^{q_i}}(s_i)\right)*u_{1n}, \\ p_{sq_1^{q_i}}(s_n)=p_{sq_2^{q_i}}(s_i)*u_{n1}\pm\left(\sum\limits_{s_i\in S}p_i(s_i)-p_{sq_2^{q_i}}(s_i)\right)*u_{nn}, \end{cases} \end{aligned} \tag{3}$$

$$\begin{aligned} &NE: p\left(sq_2^{q_i}\to S\right)= \\ &=\begin{cases} p_{sq_2^{q_i}}(s_i)=p_{sq_1^{q_i}}(s_i)*u_{ii}\pm\left(\sum\limits_{s_i\in S}p_i(s_i)-p_{sq_1^{q_i}}(s_i)\right)*u_{ni}, \\ p_{sq_2^{q_i}}(s_n)=p_{sq_1^{q_i}}(s_i)*u_{in}\pm\left(\sum\limits_{s_i\in S}p_i(s_i)-p_{sq_1^{q_i}}(s_i)\right)*u_{nn}, \end{cases} \end{aligned} \tag{4}$$

where $p_{sq_j^{q_i}}(s_i)$ is the probability of choosing server s_i to perform the subtask $sq_j^{q_i}$, u_i the gain (server utilization indicator l_{Si}) that the subtask $sq_j^{q_i}$ receives as a result of the non-cooperative choice of server $s_i \in S$.

The generalized server utilization indicator l_{Si} for the game payoff matrix G is determined based on the values of unused resources, which are determined as a percentage of the total amount of server resources:

– utilization indicator U_{CPU} of the central processor (*CPU*): the average percentage of *CPU* usage over a certain period of time (5)

$$U_{CPU}=\frac{CPU_{Usage}}{CPU_{Capacity}}*100, \tag{5}$$

where CPU_{Usage} is the number of operations performed by the processor in one second; $CPU_{Capacity}$ is the maximum number of operations that the processor can perform in one second;

– utilization rate U_{RAM} of the random access memory (*RAM*): the percentage of occupied memory at the current time (6)

$$U_{RAM} = \frac{RAM_{Usage}}{RAM_{Capacity}} * 100, \quad (6)$$

where RAM_{Usage} – amount of RAM used; $RAM_{Capacity}$ – total amount of RAM;

– network card utilization (*NW*): the percentage of the server's network card bandwidth usage at the current time (reception, transmission of bytes) (7)

$$U_{NW} = \frac{\left(b_{Sent} - b_{SentPrev}\right) + \left(b_{Recv} - b_{RecvPrev}\right)}{b_{Sent} + b_{Recv}} * 100, \quad (7)$$

where b_{Sent} is the number of bytes transmitted at the current time; b_{Recv} is the number of bytes received at the current time; $b_{SentPrev}$ is the number of bytes transmitted in the previous time unit; $b_{RecvPrev}$ is the number of bytes received in the previous time unit.

On this basis, the server utilization indicator l_{Si} is based on the determined percentage of idle resource of the most loaded component (U_{CPU}, U_{RAM}, U_{NW}) (8)

$$l_{s_i} = 100 - \max\left(U_{CPU}, U_{RAM}, U_{NW}\right). \quad (8)$$

The load dispersion indicator D_L of cluster servers is calculated from (9)

$$D_L = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2, \quad (9)$$

where n is the number of servers; $X_i, i = \overline{1,n}$ – load value on each server; $\bar{X}$ – average value of loads on servers; $(X_i - \bar{X})^2$ – square deviation from the average for each load value.

It is advisable to evaluate the distribution of cluster resources by the proposed model of optimal use of server resources [10] based on a comparison of its application with the method of adaptive cluster load balancing [11] before and after improvement. At the same time, it is also advisable to compare it with the most common and at the same time simpler methods: Round Robin and Least Connections [16, 17].

5. Results of investigating the efficiency of load distribution on the computing nodes in the cluster

5. 1. Results of designing a module for determining the feasibility of task decomposition by client queries initiated based on the theory of fuzzy logic

The basis of the proposed module is a block diagram of the organization of sequential iterative interaction between components: a data input module for analysis, a fuzzification module, a knowledge base, a fuzzy logical inference module, and a defuzzification module [12, 13]. The structural diagram of the simulation model for determining the feasibility of task decomposition is shown in Fig. 2.

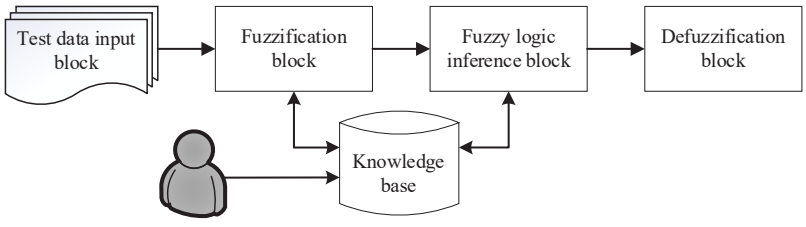

Fig. 2. Structural diagram of the fuzzy model of the task decomposition feasibility determination module

Test data input module – a set of statistical data on client query parameters (protocol, resource size) and load on server components (percentage of current CPU usage, RAM, network card) (*xlsx* files).

Fuzzification module – quantitative and qualitative values of the studied parameters are represented using term sets and linguistic variables (the degree of their belonging to the term sets of linguistic variables specified by experts is determined).

Knowledge base – a set of fuzzy production rules built by an expert.

Fuzzy inference module – a decision-making module on task decomposition based on determining the dependence between input data (client query parameters) and expert conclusions using fuzzy logic.

Defuzzification module – conversion of the obtained values of fuzzy inference into clear ones.

Thus, the input of the simulation model is fed with the studied parameters in the form of data set files with feature vectors of the generated statistical data set: the total number of feature vectors is 9999 (Fig. 3).

	Protocol	Size_resource	Load_CPU	Load_RAM	Load_NW
2	XMPP	34	35	37	65
3	SMB	73	72	87	42
4	NTP	13	10	1	76
5	FTP	478931	20	48	9
6	BackEnd	38008	43	79	12
7	FTP	3225678	27	36	3
8	SMB	153	58	86	88
9	DHCP	208	51	85	23
10	LDAP	159	44	10	68
11	FTP	26943	37	64	12

Fig. 3. A set of statistical data on client query parameters and load indicators on cluster server components

The module for determining the feasibility of decomposition of the received tasks q_i implements the following settings:

1) types of membership functions for describing the ranges of values of the studied parameters and the power of term sets for the input and output linguistic variables:

– x_1 (*Protocol*) – artificial encoding of network protocol names, membership function: triangular, term set power – 18 (Fig. 4);

– x_2 (*Size-resource*) – resource size (KB) to be processed, membership functions: Gaussian, *S* – function, *Z* – function, term set power – 5 (Fig. 5);

– x_3 (*LoadCPU*) – processor load indicator in percent, membership function: triangular, term set power – 5 (Fig. 6);

– x_4 (*LoadRAM*) – RAM load indicator in percent, membership function: triangular, term set power – 5 (Fig. 7);

– x_5 (*LoadNW*) – network card load indicator in percent, membership function: triangular, term set power – 5 (Fig. 8);

– y (*decomposition_status*) – membership functions: *S* – function, *Z* – function, term set power – 2 (Fig. 9);

2) input and output linguistic variables: 5 input linguistic variables, which correspond to the number of parameters under study and one output – an indicator of the decomposition state. Each input value (x_i) corresponds to a parameter according to the generated set of statistical data, and the membership functions configured by the expert (author) are represented by the following term sets:

– x_1 – {"*BackEnd* (web service server logic) – [0.5, 1, 1.5]", "*Telnet* – [1.5, 2, 2.5]", "*DNS* – [2.5, 3, 3.5]", "*DHCP* – [3.5, 4, 4.5]", "*SMTP* – [4.5, 5, 5.5]", "*SNMP* – [5.5, 6, 6.5]", "*FTP* – [6.5, 7, 7.5]", "*SSH* – [7.5, 8, 8.5]", "*NFS* – [8.5, 9, 9.5]", "*NTP* – [9.5, 10, 10.5]", "*SNTP* – [10.5, 11, 11.5]", "*XMPP* – [11.5, 12, 12.5]", "*LDAP* – [12.5, 13, 13.5]", "*SIP* – [13.5, 14, 14.5]", "*RTP* – [14.5, 15, 15.5]", "*IMAP* – [15.5, 16, 16.5]", "*POP3* – [16.5, 17, 17.5]", "*SMB* – [17.5, 18, 18.5]"};

– x_2 – {"Very low – [0, 2097152]", "Low – [104857, 262144]", "Average – [104857, 524300]", "High – [104857, 786432]", "Very high – [838900, 1049000]"};

– x_3 – {"Very low – [0, 0, 20]", "Low – [10, 25, 40]", "Average – [30, 45, 60]", "High – [50, 65, 80]", "Very high – [70, 100, 100]"};

– x_4 – {"Very low – [0, 0, 20]", "Low – [10, 25, 40]", "Average – [30, 45, 60]", "High – [50, 65, 80]", "Very high – [70, 100, 100]"};

– x_5 – {"Very low – [0, 0, 20]", "Low – [10, 25, 40]", "Average – [30, 45, 60]", "High – [50, 65, 80]", "Very high – [70, 100, 100]"};

– y: d_1 – simple problem: {"Does not require decomposition – 0, 10]"}, d_2 – complex problem: {"Requires decomposition – 10, 38+]"};

3) algorithm for fuzzy logical inference based on the knowledge base: Mamdani;

4) defuzzification method: Centroid (center of weight) – used to obtain the value of the complexity weight indicator w_{qi} of the received query q_i;

5) the knowledge base (set of fuzzy production rules) is built according to the system of fuzzy logical statements (2) and is shown in Fig. 10;

6) weight coefficient w for all fuzzy rules – 1;

7) total number of fuzzy production rules – 166.

The general scheme of the module for determining the feasibility of decomposition of obtained tasks q_i is shown in Fig. 11.

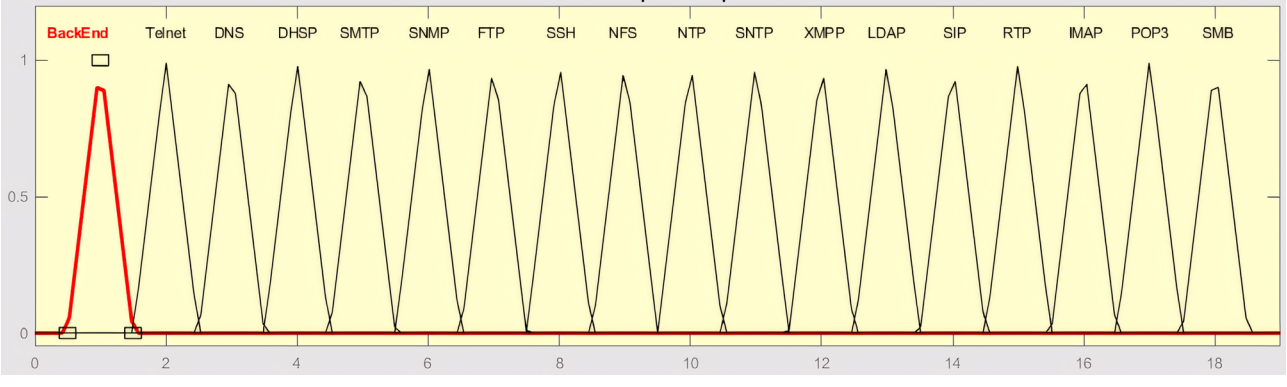

Fig. 4. Input linguistic variable *Protocol*

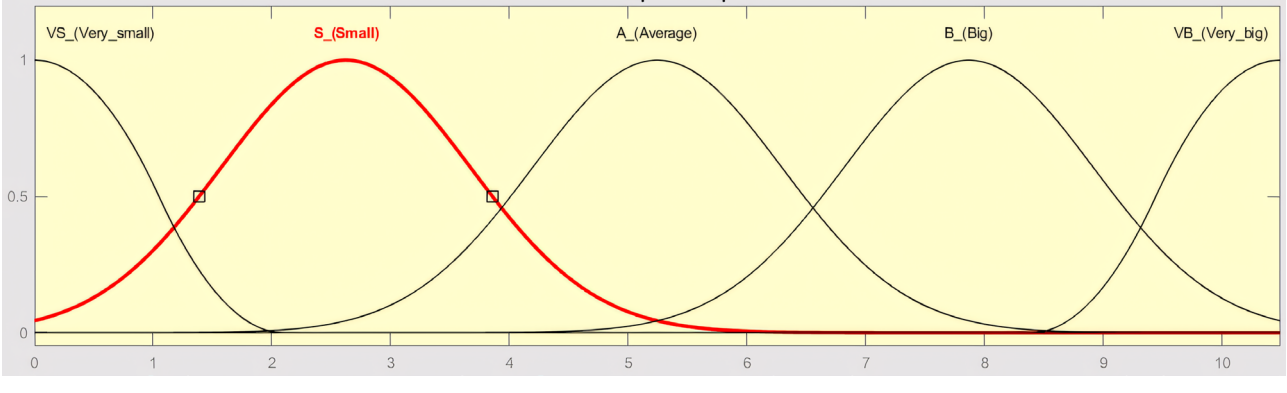

Fig. 5. Input linguistic variable *Size-resourse*

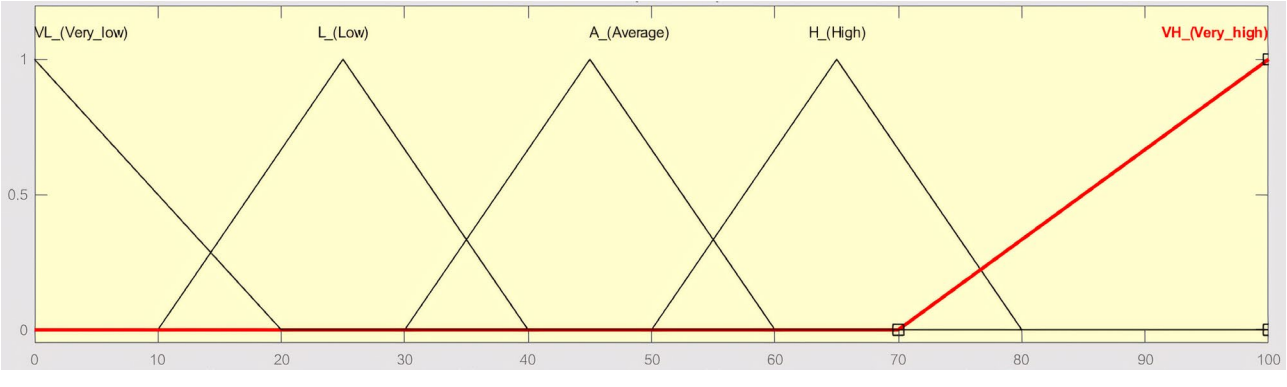

Fig. 6. Input linguistic variable *LoadCPU*

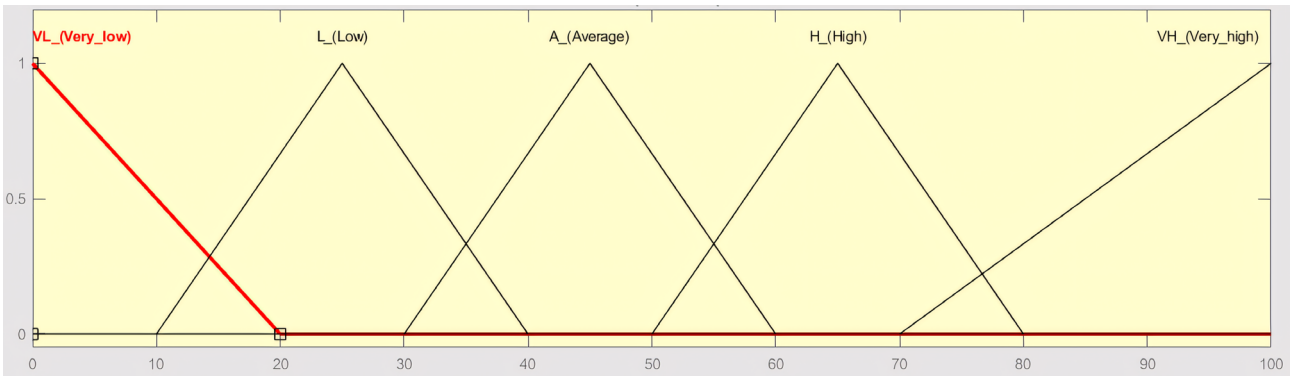

Fig. 7. Input linguistic variable *LoadRAM*

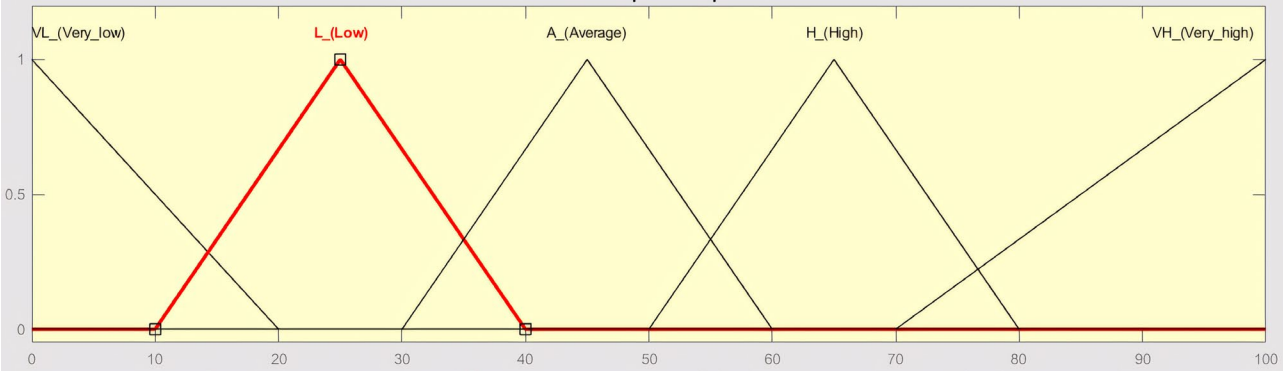

Fig. 8. Input linguistic variable *LoadNW*

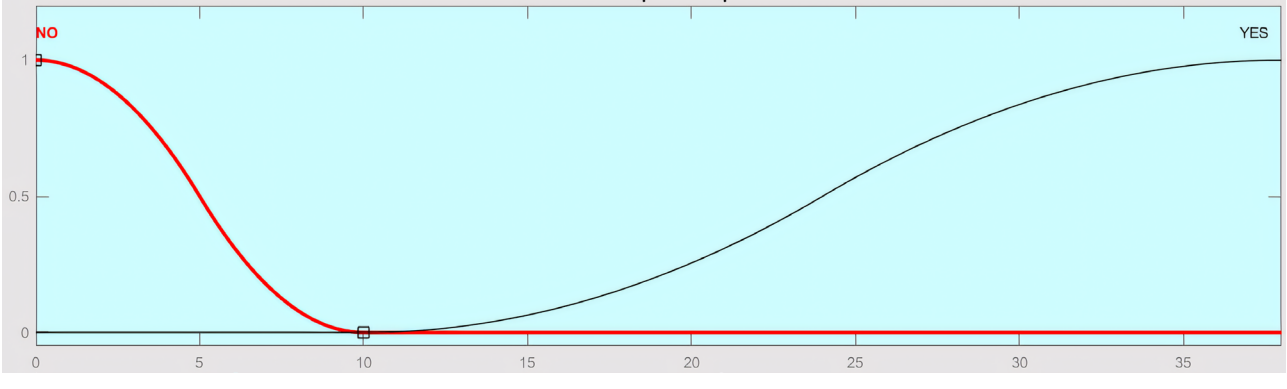

Fig. 9. Output linguistic variable *decomposition_status*

1. If (Protocol is FTP) and (Size-resourse is VS_(Very_small)) and (LoadCPU is VL_(Very_low)) then (Decomposition is NO) (1)
2. If (Protocol is FTP) and (Size-resourse is VS_(Very_small)) and (LoadCPU is L_(Low)) then (Decomposition is NO) (1)
3. If (Protocol is FTP) and (Size-resourse is VS_(Very_small)) and (LoadCPU is A_(Average)) then (Decomposition is NO) (1)
4. If (Protocol is FTP) and (Size-resourse is VS_(Very_small)) and (LoadCPU is H_(High)) then (Decomposition is YES) (1)
5. If (Protocol is FTP) and (Size-resourse is VS_(Very_small)) and (LoadCPU is VH_(Very_high)) then (Decomposition is YES) (1)
6. If (Protocol is FTP) and (Size-resourse is S_(Small)) and (LoadCPU is VL_(Very_low)) then (Decomposition is NO) (1)
7. If (Protocol is FTP) and (Size-resourse is S_(Small)) and (LoadCPU is L_(Low)) then (Decomposition is NO) (1)
8. If (Protocol is FTP) and (Size-resourse is S_(Small)) and (LoadCPU is A_(Average)) then (Decomposition is NO) (1)
9. If (Protocol is FTP) and (Size-resourse is S_(Small)) and (LoadCPU is H_(High)) then (Decomposition is YES) (1)
10. If (Protocol is FTP) and (Size-resourse is S_(Small)) and (LoadCPU is VH_(Very_high)) then (Decomposition is YES) (1)
11. If (Protocol is FTP) and (Size-resourse is A_(Average)) and (LoadCPU is VL_(Very_low)) then (Decomposition is NO) (1)
12. If (Protocol is FTP) and (Size-resourse is A_(Average)) and (LoadCPU is L_(Low)) then (Decomposition is NO) (1)
13. If (Protocol is FTP) and (Size-resourse is A_(Average)) and (LoadCPU is A_(Average)) then (Decomposition is NO) (1)
14. If (Protocol is FTP) and (Size-resourse is A_(Average)) and (LoadCPU is H_(High)) then (Decomposition is YES) (1)
15. If (Protocol is FTP) and (Size-resourse is A_(Average)) and (LoadCPU is VH_(Very_high)) then (Decomposition is YES) (1)
16. If (Protocol is FTP) and (Size-resourse is B_(Big)) and (LoadCPU is VL_(Very_low)) then (Decomposition is NO) (1)
17. If (Protocol is FTP) and (Size-resourse is B_(Big)) and (LoadCPU is L_(Low)) then (Decomposition is NO) (1)

Fig. 10. Knowledge base (a set of fuzzy production rules)

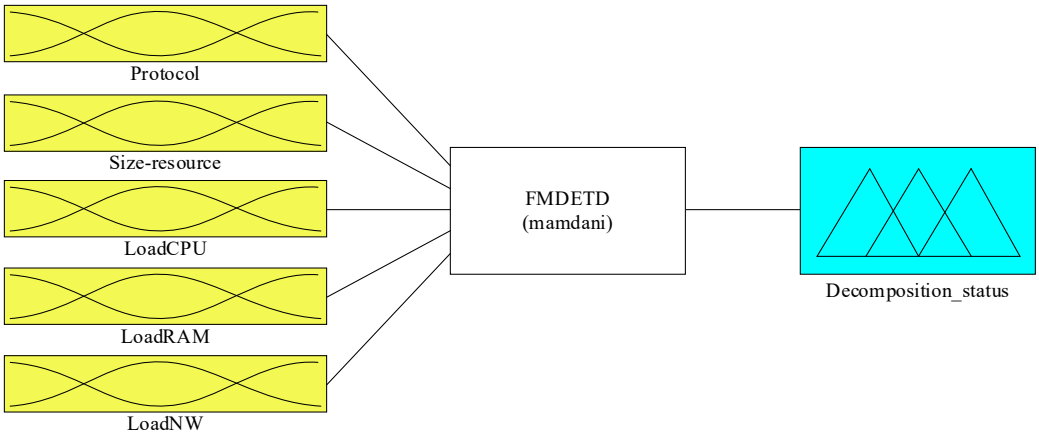

Fig. 11. General scheme of the module for determining the feasibility of decomposition of received tasks q_i

Thus, the need to decompose the received task into 2 subtasks with the appropriate division of the weight w_{q_i} of its complexity is determined. In the case of the absence of feasibility of decomposition of the received task q_i, the players of the bimatrix game G are the received task q_i and the virtual task with zero weight w_0 [10, 11].

Fig. 12 shows results of the module for determining the feasibility of decomposition of client tasks based on the analysis of the previously obtained statistical data set during cluster testing.

The results of the module for determining the feasibility of task decomposition demonstrate 100% efficiency in terms of accuracy (out of 993 tasks for which there was feasibility of decomposition, 993 were decomposed).

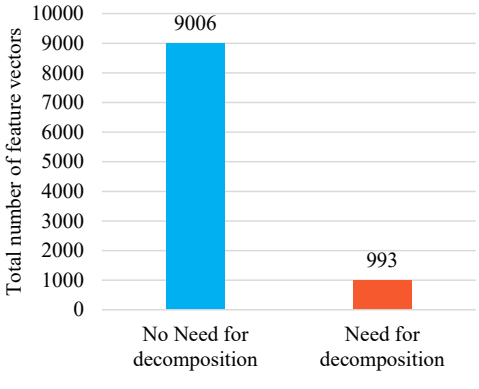

Fig. 12. Results of the task decomposition feasibility determination module

5. 2. Results of developing a module for determining the optimal distribution of tasks/subtasks across servers in a cluster system

Four cluster servers are gradually loaded with a significant number of tasks with different weights (processing complexity) (Fig. 13). The optimality criterion in this case is the indicator of uniform distribution (dispersion) of the use of cluster nodes during processing of the received tasks.

Fig. 13 shows components of the module for determining the optimal distribution of tasks/subtasks across cluster system servers for their parallel processing, which were defined and developed using the *Python* programming language, and which include the following:

– 4 servers in the form of lists (*List*), which receive from the task/subtask generator the values of different weights of the complexity of their processing;

– a task/subtask generator with different weights of the complexity of their processing (the percentage value of server resource usage for each task/subtask), the task/subtask weight is generated using the *Random* library and has a range from 0 to 10;

– functions for distributing tasks/subtasks across cluster servers using the Round Robin (RR), Least Connection (LC), Nash Equilibria (NE), and Nash Equilibria with Task Splitting (NETS) methods;

– functions for calculating the value of dispersion indicator D_L of load distribution between cluster servers for each selected load balancing method;

– functions for plotting server load charts and their dispersion in the process of distributing tasks/subtasks using each load balancing method;

– function for constructing a general plot of load distribution dispersion indicators between cluster servers for each load balancing method;

– function of calculating the average balancing time and processing time of tasks/subtasks by cluster system servers using the selected load management (balancing) methods.

Fig. 14, 15 show the results of modeling the obtained distribution of cluster resources in the process of managing tasks received from clients without their decomposition based on the Least Connection (Fig. 14) and Round Robin (Fig. 15) methods in the form of load variation curves of the above servers. On the abscissa axis – the number of tasks that servers manage to complete by the time the cluster is fully loaded (the ordinate axis is in percent).

Fig. 16, 17 show the results of modeling the resulting distribution of cluster resources during the processing of tasks received from clients before improving the model of optimal use of server resources (Fig. 16) and after, taking into account the decomposition of individual complex tasks (Fig. 17) in the form of variational load curves of the above-mentioned servers.

The general plot of load distribution dispersion indicators among cluster servers is shown in Fig. 18.

A comparative analysis of the calculations performed based on the obtained results of modeling the process of optimal task/subtask distribution across cluster servers using the proposed approach (*NE-TS*) and existing similar solutions (*RR*, *LC*, *NE*) is shown in Fig. 19.

An assessment of the impact of load balancing time on the total task processing time is shown in Fig. 20.

Fig. 20 shows the average task processing time and the time spent on the load balancing process using the selected load management (balancing) methods.

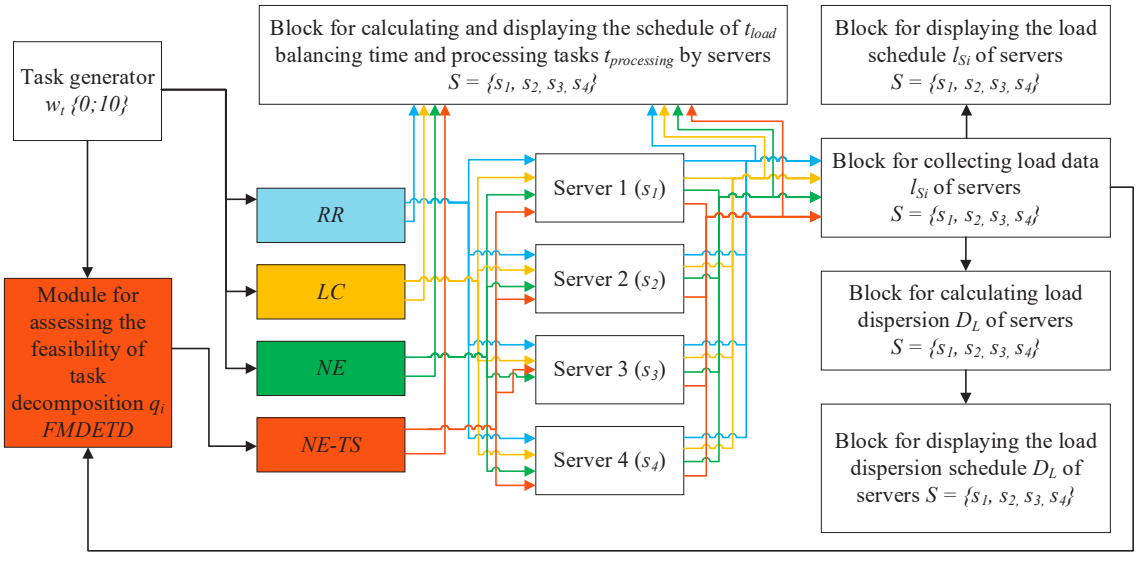

Fig. 13. Module for determining the optimal distribution of tasks/subtasks across cluster system servers for their parallel processing

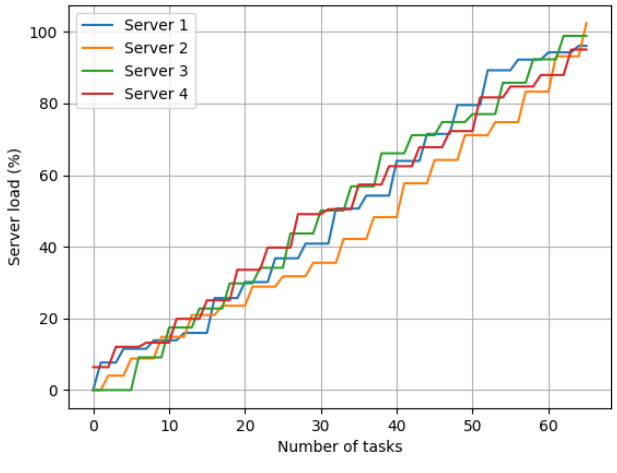

Fig. 14. Results of simulation modeling of the obtained cluster resource distribution based on the *Least Connections* method

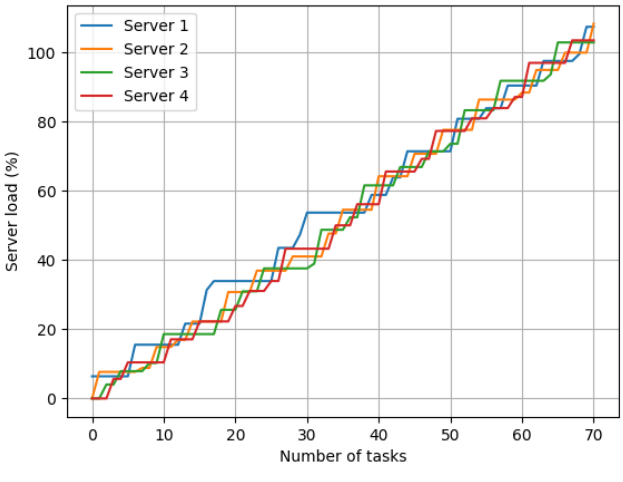

Fig. 16. Results of simulation modeling of the obtained cluster resource distribution based on the *Nash Equilibria* method

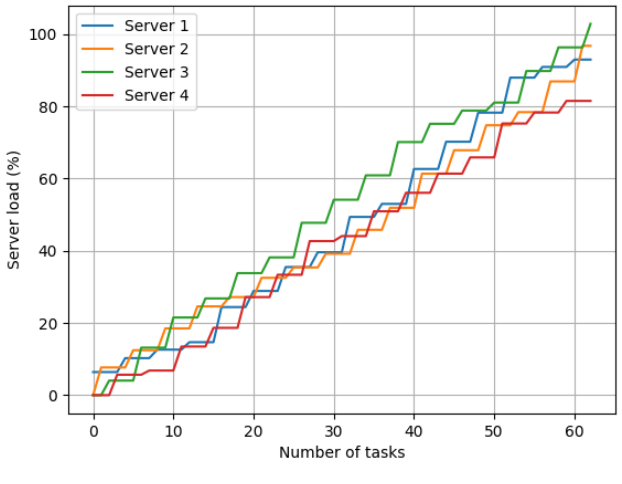

Fig. 15. Results of simulation modeling of the obtained cluster resource distribution based on the *Round Robin* method

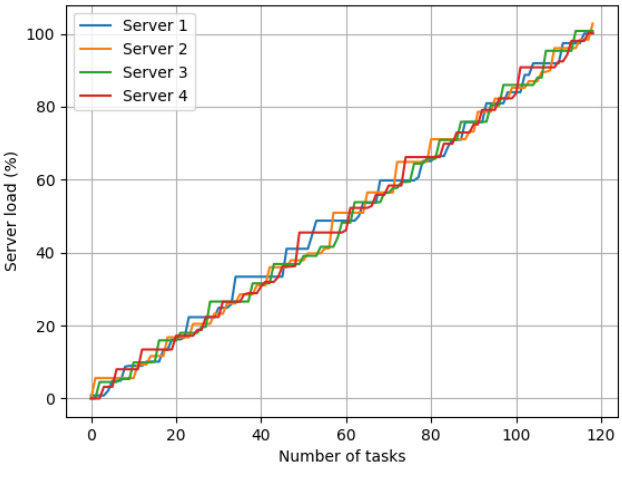

Fig. 17. Results of simulation modeling of the resulting cluster resource distribution based on the *Nash Equilibria with Task Splitting* method

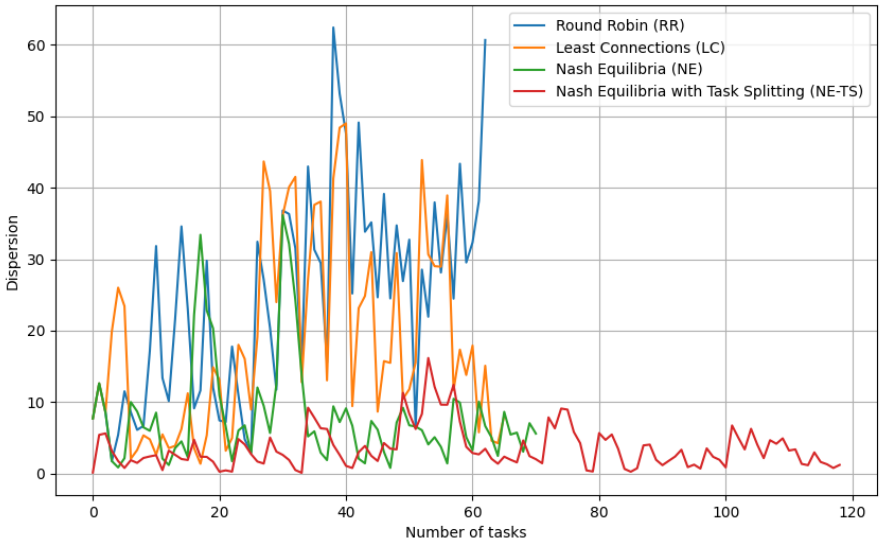

Fig. 18. Results of simulation modeling of selected methods of load management on a server cluster based on variance

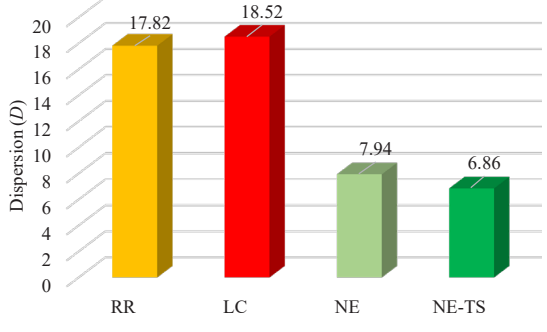

Fig. 19. Indicator of optimum load distribution on cluster servers using different dispersion-based methods

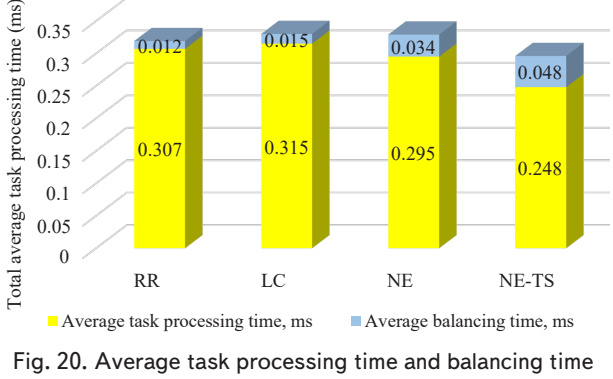

Fig. 20. Average task processing time and balancing time for selected load management (balancing) methods

6. Discussion of results based on investigating the efficiency of the distribution of server resources in the cluster system

The results of the module for determining the feasibility of task decomposition (Fig. 12) demonstrate 100% efficiency in terms of accuracy. During the operation of the fuzzy model of the developed module, it was found that out of the total number of feature vectors of the generated set of statistical data (9999), 933 needed decomposition, the processing of which, in the case without decomposition, would lead to faster overloading of both individual nodes and the entire cluster, as well as the loss of some tasks.

Fig. 18 shows that in the case of using the Least Connections method (Fig. 14), the indicator of uniform distribution of server resources of the cluster is higher than the similar indicator when using the Round Robin method (Fig. 15) since the dispersion of server load indicators at each point in time is smaller, which in turn makes it possible to increase the number of processed tasks initiated by client queries under the conditions of limited hardware resources and acceptable time for processing these requests.

Fig. 16–18 demonstrate that in the case of using the improved model (Fig. 17), the indicator of uniform distribution of cluster resources (Fig. 18) is significantly higher in contrast to the indicator of the model before improvement (Fig. 16, 18), which is significantly higher in the case of using the Round Robin method. This, in turn, leads to an increase in the number of processed tasks by two times.

Fig. 20 demonstrates that when using the model built (*NE-TS*), the time spent on load balancing (average balancing time) is slightly higher than the time spent in comparison with other methods. But at the same time, the average task processing time is much shorter in contrast to *RR*, *LC*, and *NE*. As a result, the total average task processing time when using *NE-TS* does not exceed the total time when using the *RR*, *LC*, and *NE* methods.

Our results demonstrate an increase in the efficiency of resource allocation according to the optimality criterion – uniform (stable in equilibrium) resource allocation throughout the entire process of cluster functioning. This is evidenced by the results of simulation modeling of this process, which are shown in Fig. 14–19. This, in turn, makes it possible to significantly increase the number of processed tasks, as well as increase the level of functional stability of military IS (defense and security forces of the state), especially in wartime.

At the same time, the results shown in Fig. 20 indicate that the use of this apparatus does not exceed the total average processing time of tasks in comparison with existing methods (*RR*, *LC*, *NE*).

A special feature of the proposed model of managing load on computing nodes in a server cluster based on the theory of fuzzy logic and Nash equilibrium, in contrast to existing ones [7, 9], is to identify and determine the feasibility of decomposition of individual (complex) tasks after receiving incoming requests, using fuzzy logic theories. After that, the probabilistic distribution of tasks/subtasks across cluster servers is calculated based on Nash

equilibrium. Thus, optimal (even) use of server resources of the cluster is achieved, which is also stable in equilibrium.

Our solution could be used in highly loaded information systems during the operation of which the server infrastructure can be scaled, at regular and peak loads. The built model of load management for computing nodes in the server cluster based on the theory of fuzzy logic and Nash equilibrium can adapt to changes in the operating environment.

The disadvantages include the fact that in the process of determining the feasibility of task decomposition, the balancing system may miss some task that was potentially subject to decomposition. This is explained by the fact that the value of the server load indicators may be on the border of the terms of the input linguistic variables of the fuzzy model and in the future the calculation of the weight (complexity) of the task may be incorrect. As a result, it is necessary to adjust the fuzzy rule base by adding a new fuzzy rule that did not exist for the given values of the input linguistic variables.

Promising areas for future research include the design of a mechanism for automatic correction of the knowledge base of the module for determining the feasibility of task decomposition for their further processing, development of new and/or improvement of our model for managing the load on computing nodes in a server cluster at the network and/or transport levels of the OSI model as the model built takes into account application-level information exchange protocols, etc.

7. Conclusions

1. A module for determining the feasibility of task decomposition by client-initiated requests has been designed, which is built on the basis of a fuzzy logic apparatus using the Fuzzy Logic Toolbox™ package. To construct the module, the following steps were taken:

1) a set of statistical data on the parameters of client queries and the current level of load on the cluster server components during testing was generated;

2) classes of client queries were defined by types to be processed:

– simple – requests that do not require significant computing resources and/or requests that are impractical to decompose (video/audio streaming);

– complex – requests that require significant computing resources (for example, related to the processing of large data sets (Big Data), using information and analytical systems, complex requests to a database cluster, etc.);

3) input and output linguistic variables were defined;

4) the types of membership functions, the power of term sets, and the ranges of input and output linguistic variables were defined;

5) the algorithm for fuzzy logical inference based on the knowledge base was defined;

6) the defuzzification method was defined to obtain the value of the complexity weight indicator w_{qi} of received query q_i;

7) the weight coefficient w was defined for all fuzzy rules;

8) the test data format was prepared;

9) fuzzy production rules were devised for the knowledge base;

10) a script was developed for correct data entry;

11) experimental studies on detecting complex queries for their decomposition were conducted.

The results of the fuzzy model of the module for determining the feasibility of task decomposition demonstrate 100% efficiency in terms of accuracy (for 993 out of 9999 tasks for which there was feasibility of decomposition, 993 are decomposed).

2. A module for determining the optimal distribution of tasks/subtasks across cluster system servers for their parallel processing based on Nash equilibrium has been built, which was developed in the Python programming language. To build the module, the following were selected and designed:

– 4 servers in the form of lists (List);

– a generator of tasks/subtasks with different weights of their processing complexity (percentage of server resource usage for each task/subtask);

– functions for distributing tasks/subtasks across cluster servers using the Round Robin (RR), Least Connection (LC), Nash Equilibria (NE), and Nash Equilibria with Task Splitting (NETS) methods;

– functions for calculating the value of dispersion indicator D_L of load distribution between cluster servers for each of the above methods;

– functions for plotting server load charts and their dispersion in the process of distributing tasks/subtasks using each of the above methods;

– the function of constructing a general chart of load distribution dispersion indicators between cluster servers for each of the above methods;

– the function of calculating the average balancing time and processing time of tasks/subtasks by cluster system servers using the selected load management (balancing) methods.

The results of the module for determining the optimal distribution of tasks/subtasks across cluster system servers for their parallel processing demonstrate an increase in the efficiency of resource distribution according to the optimality criterion – uniform (stable in equilibrium) distribution of resources throughout the entire process of cluster operation. At the same time, an assessment of the impact of the load balancing process on the task processing time was carried out. Based on the results of simulation modeling, it can be concluded that the application of the simulation model built does not exceed the permissible task processing time in comparison with existing load balancing methods.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Pro rishennia Rady natsionalnoi bezpeky i oborony Ukrainy vid 20 serpnia 2021 roku "Pro Stratehichnyi oboronnyi biuleten Ukrainy". Ukaz Prezydenta Ukrainy vid 17.09.2021 r. No. 473/2021. Available at: https://zakon.rada.gov.ua/laws/show/473/2021#Text
2. Pro rishennia Rady natsionalnoi bezpeky i oborony Ukrainy vid 18 chervnia 2021 roku "Pro Stratehiiu rozvytku oboronno-promyslovoho kompleksu Ukrainy"/ Ukaz Prezydenta Ukrainy vid 20.08.2021 r. No. 372/2021. Available at: https://zakon.rada.gov.ua/laws/show/372/2021#Text
3. Begam, G. S., Sangeetha, M., Shanker, N. R. (2021). Load Balancing in DCN Servers through SDN Machine Learning Algorithm. Arabian Journal for Science and Engineering, 47 (2), 1423–1434. https://doi.org/10.1007/s13369-021-05911-1
4. Klots, Y. P., Stefanovytch, K. Y., Shakhoval, Y. S., Demeshko, V. I. (2019). Dynamic traffic balance between several providers. Herald of Khmelnytskyi national university, 4 (275). 62–67. Available at: https://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/01/12-7.pdf
5. Naz, N. S., Abbas, S., Adnan, M., Abid, B., Tariq, N., Farrukh, M. (2019). Efficient Load Balancing in Cloud Computing using Multi-Layered Mamdani Fuzzy Inference Expert System. International Journal of Advanced Computer Science and Applications, 10 (3). https://doi.org/10.14569/ijacsa.2019.0100373
6. Chen, L., Wu, K., Li, Y. (2014). A Load Balancing Algorithm Based on Maximum Entropy Methods in Homogeneous Clusters. Entropy, 16 (11), 5677–5697. https://doi.org/10.3390/e16115677
7. Priya, S. S., Rajendran, Dr. T. (2025). Enhanced Weighted Round Robin: A New Paradigm in Cloud Load Balancing. Indian Journal Of Science And Technology, 18 (15), 1220–1228. https://doi.org/10.17485/ijst/v18i15.3976
8. Dash, Y., Dalei, R. K., Dhal, K. (2025). Modified Genetic Algorithms (GA) for Load balancing in Cloud Computing. Journal of Information Systems Engineering and Management, 10 (54s), 1–8. https://doi.org/10.52783/jisem.v10i54s.11028
9. Matiwure, T., Ndlovu, A. (2025). Enhancing throttled load balancing algorithm with machine learning for dynamic resource allocation in cloud computing environments. International Journal of Computer Science and Mobile Computing, 14 (6), 20–25. https://doi.org/10.47760/ijcsmc.2025.v14i06.003
10. Fesokha, V. V., Neroznak, E. I., Sova, O. Ya. (2023). An improved model of optimal use of resources of a cluster system of military assignment based on nash equilibrium. Collection of Scientific Works of the Military Institute of Kyiv National Taras Shevchenko University, 79, 159–171. https://doi.org/10.17721/2519-481x/2023/79-15
11. Fesokha, V., Neroznak, Y., Sova, O., Nesterov, O. (2023). Method of adaptive load balancing in cluster systems for military purposes based on Nash equilibrium. Zbirnyk naukovykh prats Tsentru voienno-stratehichnykh doslidzhen NUOU imeni Ivana Cherniakhovskoho, 3 (76), 101–110. https://doi.org/10.33099/2304-2745/2022-3-76/101-110
12. Sivanandam, S. N., Sumathi, S., Deepa, S. N. (2007). Introduction to Fuzzy Logic using MATLAB. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-35781-0
13. MATLAB. The MathWorks, Inc. Available at: https://www.mathworks.com/help/matlab/
14. Welcome to Nashpy's documentation! Nashpy. Available at: https://nashpy.readthedocs.io/en/stable/
15. Campesato, O. (2020). Python 3 for Machine Learning. Mercury Learning and Information, 364. Available at: https://www.amazon.com/Python-Machine-Learning-Oswald-Campesato/dp/1683924959
16. Neroznak, Ye. I., Merkotan, D. Yu., Sova, O. Ya. (2021). Metody ta alhorytmy balansuvannia navantazhennia v klasternykh systemakh na osnovi elementiv shtuchnoho intelektu. Systemy i tekhnolohiyi zviazku, informatyzatsiyi ta kiberbezpeky: aktualni pytannia i tendentsiyi rozvytku: I Mizhnarodna nauk.-tekhn. konf. Kyiv, 215–217.
17. Load Balancing Algorithms and Techniques. Available at: https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques