Designing Networked High-Load Distributed Computing Web Systems for Intensive Processing of Information Data Flows

Mykola Pasieka[a], Vasyl Sheketa[a], Iryna Halushchak[b], Svitlana Verbeshchuk[b], and Mykhailo Yasinskyi[c]

[a] *National Technical University of Oil and Gas, 15 Karpatska str., Ivano-Frankivsk, 76068, Ukraine*
[b] *Vasyl Stefanyk Precarpathian National University, 57 Shevchenko str., Ivano-Frankivsk, 76000, Ukraine*
[c] *Ukraine Academy of Printing, 19 Pid Goloskom str., Lviv, 79020, Ukraine*

Abstract

The grid model and algorithm of the high-load distributed fault-tolerant web system architecture are improved. The main difference lies in the possibility of aggregation and the sharing of a large number of heterogeneous computing resource sets to process information and data flows distributed between different geographical areas. Compared with the traditional method, the proposed model and algorithm allow the use of other network resources connected to the functional network. In the traditional method, these resources are not available within a single computing node on an independent computing platform. Further developed innovative methods for building high-load fault-tolerant distributed cluster software systems, which generally greatly increase the total amount of information flow in the node communication system. Therefore, this method is suitable for distributed fault-tolerant software systems in which the operational loss of information data flow is unacceptable. Further developed an algorithm for automatic load control on an independent computer platform for information data flow to achieve effective expansion (clustering) of the software system.

Keywords

Software system, algorithm, architecture, web system, network.

1. Introduction

Computing node load balancing ensures an even load of hardware and software systems on independent computing platforms. The software system that is balancing the computational load must automatically decide on which node to perform the computation for the new information task. Thus, the main task of balancing is the process of effectively supporting the process of moving (migrating) part or all of the calculations from the most loaded computing nodes to the less loaded ones, loading input factors i of $i = 1, ..., r$; u_j are weighing output parameters, weighing output parameters j of $j = 1, ..., s$.

When solving the problem of maximizing efficiency standards, there is an urgent problem, that is, there is a share in the distribution of two linear aggregate values [46].

In addition, the problem of maximizing the efficiency of web systems is called linear particle programming. At the same time, the possibility of converting linear particle programming into linear programming problems is very high:

$$f_0 = \frac{\sum_{i=1}^{r} v_i x_i}{\sum_{j=1}^{s} u_j y_j} \to min!, (1)$$

with $\frac{\sum_{i=1}^{r} v_i x_i}{\sum_{j=1}^{s} u_j y_j} \geq 1$ or all modules $m = 1, 2, ..., n$; $u_j \geq 0, j = 1, 2, ..., s$; $v_i \geq 0$,

Cybersecurity Providing in Information and Telecommunication Systems, January 28, 2021, Kyiv, Ukraine
EMAIL: pms.mykola@gmail.com (A.1); vasylsheketa@gmail.com (A.2); pasyekanm@gmail.com (B.3); iryna.galushchak@gmail.com (B.4); leuro@list.ru (C.5)
ORCID: 0000-0002-3058-6650 (A.1); 0000-0002-1318-4895 (A.2); 0000-0002-4824-2370 (B.3); 0000-0001-9445-6348 (B.4); 0000-0002-4824-2370 (C.5)

$i = 1, 2, ..., r$; y_j are expression j and the output parameter of the tested computing module (node); x_{im} is expression i that input factor m of that computing node with $i = 1, ..., r$ and $m = 1, ..., n$; y_{jm} expression j that input factor m of that computing node with $i = 1, ..., r$ and $m = 1, ..., n$; v_i – input factor weights i with $i = 1, ..., r$; u_j are weighing the output parameter j with $j = 1, ..., s$.

For further research, we use a complex fractional programming theory algorithm (a more traditional linear programming algorithm with less resource occupancy) to modify the non-linear problem of efficiency standard optimization. Therefore, the use of linear optimization methods can simplify the complex problems of efficiency standard optimization into linear problems [28]. In order to obtain the value of the efficiency standards of all computing modules (nodes), it is necessary to solve the problem of maximizing each module involving scientific research separately. In this case, replace the vectors x_{im} and y_{im} [24] with the input and output parameter configuration files of the studied computing module, respectively. In another task of maximizing efficiency standards, they remain the same for each computing node [18, 30, 35, 45, 50]. In this task, restrictions are imposed to ensure that qualitative efficiency values are found in the region between zero and one ($e_0 \in [0,1]$). The basic properties of the model are formed by the efficiency limit criterion. Its objective function proportionally attempts to increase the observed output parameters of the calculation module to the limit of the efficiency standard. In the mathematical decoupling of a certain function, after applying the duality theorem to it, an equivalent form or so-called hug form is formed. The dual principle of linear programming is used to prove that for each directly optimized linear model considered, the efficiency criterion has a corresponding dual optimized linear model, and when one model decision includes all expressions, the other model decision also includes all expressions. formula. Therefore, the initial optimization model will be regarded as a direct linear programming problem. Thus, using the dual method in our scientific research, we are guaranteed to obtain minimization of the weighted sum of input parameters in one normalized output parameter. As a result of these manipulations, we obtain a mathematical formulation of the linear programming model using the dual method, which is performed as follows:

$$\min g_0 = \sum_{i=1}^{r} t_i x_{i,} \quad (2)$$

with g_0 is the value of the efficiency criterion of the studied computing node; x_i is an expression i of that computing node input factor with $i = 1, ..., r$; t_i is variable weighting factors.

The linear combination determines the potential reference group, which is used to measure the efficiency standards of the computing nodes involved in the research. Therefore, according to the optimization task of minimizing the objective function (2), the proposed method selects such a reference group in which the efficiency standard of the computing node involved does not seem to be effective enough [10, 37, 40].

Therefore, the mathematical problem can be explained as for the computing node to be studied, the minimum efficiency standard for determining the input parameter G_0 is compared with the weighted probability of the comparison unit. The weighted combination of any input parameter and output parameter does not subtract the output parameter, any The weighted integral combination of the input factors is G_0 times larger than the input factors.

2. Parametric Network Model with Time Index to Calculate Load

The basic principles of the operation and function of the cloud computing network model on an independent computing platform are very different from the traditional serial and parallel models. The main feature of the cloud computing network model is the ability to aggregate and collectively use a large information set of heterogeneous computing key data streams distributed in geographic locations. This method has significant advantages. For example, when the developed software system needs information resources that are not available in a computing node, it can get it to other computing nodes connected to the cloud network. [5, 8, 9, 11, 23, 27].

Suppose that m computing resources are available in the network cloud environment, and there is a task flow t distribution system, which evenly distributes information tasks $j \in t$ among the available resources. In the framework of combining this developed software system with cloud technology, each user's information task can be divided into certain computational actions $k \in j$.

When setting the task, determine the processing time d_j to obtain the corresponding node. Each information task of user j and all corresponding actions $k \in j$ are in the cloud grid and at a certain point r_j. Therefore, there is a cloud network running in online mode with custom r_j. A large number of these tasks have unknown values beforehand. Once a custom task to be processed arrives on the cloud network, plan to search for and allocate the necessary resources to run it. Assume that the final allocation result of the calculation task S for each action $k \in j$ is to process $C_k(S)$ within a certain period. It's required. Therefore, the processing speed of the user's computing task in the cloud network j will not be lower than the processing speed in the specific period determined by the expression:

$$C_j(S) = \max_{k \in j} C_k(S), \quad (3)$$

with $C_k(S)$ is the maximum processing time of the user's task; k is action; j is mission.

Let us define p_j as the processing time of the user's task $k \in j$. So, the processing time of a user's task in a cloud network can be calculated as follows:

$$p_j = C_j(S) - \min_{k \in j}(C_k(S) - p_k), \quad (4)$$

with p_j is user task processing time; $C_k(S)$ is execution time; $C_j(S)$ is maximum execution time; p_k is decision implementation time.

The received point value allows estimation of the properties of the cloud network computing environment. To analyze the quality of cloud services provided by the network computing environment, you can use the value of the maximum delay in the user task:

$$L_{max} = \max_{j \in t}\left(C_j(S) - d_j\right), \quad (5)$$

with L_{max} is a maximum delay of the user's task; $C_j(S)$ is maximum processing time; d_j is a time of uncompleted user tasks.

Thus, to optimize distributed cloud resources it is necessary to minimize the value of L_{max}, and equal to the previous value of the parameter you can also use the value of T_j, which will determine for us the delay time of user tasks that are calculated:

$$j \in t \wedge C_j > d_j, \quad (6)$$

with t is information computing resources are available; d_j is a time of uncompleted user tasks; C_j is time to complete one task; k is action; j is mission.

Therefore, the indicator provides information about the number of outstanding calculation requests of users who have entered the cloud network for processing. The computing resource consumption of the RC_k cloud network is a specific subtask of computing, which is defined as the product of the corresponding solution time multiplied by the number of resources used in the network:

$$RC_k = p_k \times m_k, \quad (7)$$

with RC_k is total consumption of network computing resources; p_k is decision implementation time; m_k is the number of resources used.

Using the total use value of cloud computing resources used, you can calculate the total cost of available necessary information resources U:

$$U = \frac{RC(S)}{m \times (\max_{j \in t} C_j(s) - \min_{j \in t}(C_j(S) - p_j)}, \quad (8)$$

with U is the amount of use of available information resources; m is resource quantity; p_j is processing time of the j^{th} problem; $RC(S)$ – time of total consumption; $C_j(S)$ is the time of the j^{th} task.

Thus using the formed standard that characterizes the optimal use of computing resources in a cloud distributed network through a certain value. The following situations usually occur in the process of data processing of information requests in a cloud distributed network: the software system crashes when processing highly loaded user tasks [6, 7, 26, 43, 44, 47].

Based on the mathematics outlined above, we conclude that the user's task of processing an information request must be performed several times before it can be completed. Since users and administrators may have different (or even conflicting) requirements for cloud-based web systems, it is difficult to find a common metric that everyone is satisfied with. From the perspective of a distributed software system developed by users using cloud technology, the following metrics can be distinguished between average query response time (Average Response Time, ART) and average query latency (Average Wait Time, AWT):

$$ART = \frac{1}{|t|}\sum_{j\in t}(C_j(S)), (9)$$

with $C_j(S)$ is the time of the j^{th} task; p_j is the time of realization of the jth task; t is available resources.

$$AWT = \frac{1}{|t|}\sum_{j\in t}\left(C_j(S) - p_j\right), (10)$$

with $C_j(S)$ is the time of execution of the j^{th} task; t is available resources.

The *ART* parameter value represents the average response time to user information requests, that is, the speed of processing user tasks. In addition, the value of the parameter *AWT* is very important for developers of action algorithms in small information tasks and their query processing. Therefore, the best and simple way to measure the efficiency of fair use of information and hardware resources is to calculate the deviation of the expected weighted average time for query processing:

$$AWTD = \frac{1}{|t|}\sqrt{\sum_{j\in t}(C_j(S) - p_j)^2 - (\sum_{j\in t}\frac{C_j(S) - p_j}{|t|})^2}, (11)$$

with *AWTD* is *the* deviation of the average waiting time for processing the user's information request; t is available resources; $C_j(S)$ is the time of execution of the jth task; p_j is the time of realization of the j^{th} task.

The *AWTD* must be minimized to get the best results in the cloud network. In the modern cloud network web system, the ability to complete the processing of a given number of tasks is more important than the acceleration of the distributed high-productivity web system obtained by using this processing method. It should be noted that, compared with the user's information task executed in a parallel system of the traditional architecture, the user's information task executed in a cloud network environment may have a quite complex architecture. For example, the information flow of a user task has a more complex logical structure than the corresponding task package [25]. The use of cloud grid needs to change concepts such as errors in the web service development program system designed based on the grid model. Once there is a situation where the user information task cannot be successfully executed and completed, an error message will be generated.

Using a cloud mesh requires changing such notions as errors of a developed program system of web services which is designed on a mesh model, generates error messages as soon as there comes a situation when it is impossible to successfully execute and finish a user's information task. For example, if suitable resources cannot be found to perform information calculation or information calculation cannot be completed, the developed software system using cloud technology may malfunction.

Using the concept of fault tolerance in software systems using cloud technology, we can define it as the main possibility of increasing the time delay of software and hardware errors until the work of processing user requests cannot be completed. An indicator of completed user request processing in a software system that uses cloud technology "workload completion", which is formed by the ratio of completed user tasks to the total number of requests set by the cloud network scheduler:

$$WC = \frac{\sum j \in t \wedge (j\ complited)}{|t|}, (12)$$

with WC is the indicator of completed user requests processing; *t* is available resources; *j* is the task.

This indicator allows defining the main limitations of the cloud network software system, the maximization of which may be the main goal. However, from the perspective of using free information and hardware resources, it also has some key limitations, because user tasks with a small number of computing operations have a significant impact on the change of this value [16].

Task completion calculates the number of completed operations relative to the total number of operations performed. These have been implemented in the cloud software system for user task assignments.

Through this indicator, you can define the main limitations of the cloud network software system, and maximizing it can be the main goal. However, from the perspective of using free information and hardware resources, it also has some key limitations, because the tasks of users with fewer computational actions have a greater impact on the change of the value.

Task completion calculates the number of completed operations relative to the total number of operations performed. These have been implemented in the cloud software system for user task assignment:

$$TC = \frac{\sum j \in t \wedge k \in j \wedge (k\ complited)}{\sum j \in t\ |j|}, (13)$$

with *TC* is the index of completed actions for processing user tasks; *t* is available resources; *k* is the action; *j* is the task.

It is also worth considering the concept of completing unlocked operations from user tasks to handle the completion of enabled tasks, i.e., they can only be executed when all corresponding dependencies of a given sequence of operations will be executed by a software system using cloud technology:

$$ETC = \frac{\sum j \in t \wedge k \in j \wedge (k\ complited)}{\sum j \in t \wedge k \in j \wedge (k\ enabled)}, (14)$$

with *ETC* is the integral indicator of completed unlocked actions relative to user tasks; *t* is available resources; *k* is the action; *j* is the task.

Therefore, when developing a program system that is very different from traditional serial and parallel models, we have considered the main principles of cloud network model operation. The main difference between them is that they can gather and share a large number of heterogeneous information resources distributed among geographically independent computing independent platforms. This architecture method of using cloud technology to develop program systems has brought considerable advantages and financial benefits. For example, when a program web system needs a large number of information resources that cannot be accessed within a computing node, it can combine them Connect to another connected node to the cloud network.

3. Models and Algorithms for Optimization of Distributed High-Performance Software Web System Architecture

Any web system or cloud service that serves many users with a priori high load. However, high-load distributed software web systems cannot effectively apply models, methods, and algorithms. These models, methods, and algorithms are used as the basic methods for developing ordinary websites [4, 42]. If there is no proper way to use cloud technology to optimize the system development architecture of the program, then a large number of users and corresponding calculations will lead to complexity in timely maintenance [1, 17, 21, 29]. Now, all models, methods, and algorithms used to develop distributed fault-tolerant web system architecture have been developed and used through recognized information technology [19]. The method of sending information packets should use an algorithm to calculate the checksum of the control protocol transmission of the packet, which allows you to use the checksum of the header of this packet to display the checksum of the control protocol transmission of the data of the packet [2, 12, 20, 22, 36].

Consider the checksum recalculation algorithm, which will use the output packet checksum and the input packet checksum to calculate the checksum. Therefore, if the clipboard is divided into two information parts, the checksum of the entire buffer can be expressed linearly by the checksum of its parts. The typical length of the packet header protocol control transmission is usually several times shorter than the base of the entire information packet. Therefore, the calculation amount of the entire software web system resources is reduced by n times, and the redirection cost of client requests from one node to another can be significantly reduced. In the next technical stage, user requests will be passed to the computing node for processing. The architecture of computing nodes used to process user information requests should be organized in a hierarchical structure. At the bottom of this hierarchy are important computing modules used to process user requests. The failure of these modules will temporarily slow down the overall operating speed of the web system, but will not paralyze the entire system. If the lower computing module does not work, the cloud network web system allows the corresponding computing module of other nodes to be used. However, in the calculation process of this organization, it is obvious that the communication level between nodes when processing user requests should be perfect.

The organizational characteristics of the architecture of computing nodes for processing high user requests require that these modules have a clear hierarchical structure. The essence is that each computing module of a node must be isolated from each other as much as possible, and exchange information at the message level of the public system while relying on the information. Therefore, the design of the software web system should be developed according to the following principles: “The fault is a specific part of the web service work and should be controlled at the system message exchange level.” The architecture of the fault hierarchy is that any web system is composed of computing modules, and when the descendant modules refuse, the next one has a series of operations. The organizational features of the computational node architecture for processing high user requests require that these modules have a clear hierarchical structure. The essence is that each node’s computing module must be as isolated as possible from each other and exchange information at the level of the public system message, relying on the information. Therefore, the design of the web system software should be developed according to the following rules: “The fault is a specific part of the website operation and should be controlled at the system message exchange level.” The architecture of the fault hierarchy is that each network system consists of computing modules, and when the child modules refuse, the next one performs a series of operations (Fig. 1).

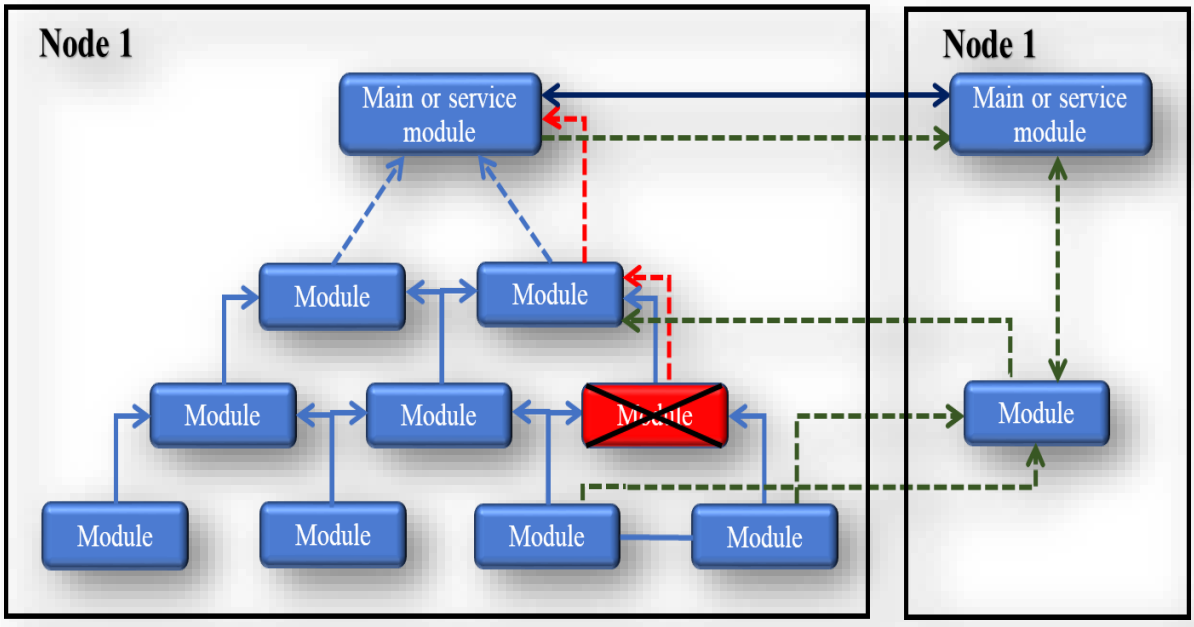

Figure 1: Model of failure hierarchy processing in software web systems

The selection of the computing module used to execute user requests is not only based on its load but also based on information about the number of failures in other modules of the cloud node. This

algorithm is necessary in order not to create a packet for processing the instantaneous data flow from a very busy node module to a less busy node module. As a result of these manipulations, a complex cloud network for exchanging computing information arrays between corresponding modules of computing nodes is formed. However, the main purpose of building a high-load distributed software web system is to create such a software system without using cloud technology, which will not fail during industrial use and will be able to work and process various types for as long as possible information. Challenges and own mistakes [41]. After the information message flow has been redirected, the computing module will not stop working.

Therefore, we can emphasize the strategies that can be implemented in each computing module [15].

In the first strategy, we will understand that the load on the computing module is large. In this case, the module will execute the entire list of user tasks that enter or remain in it for processing, perform a master restart, and generate an information message to the module higher in the hierarchy, and prepare to work [48]. It is assumed that the cloud network module of the computing node and the main load balancer constantly exchange information messages about readiness, and the module will receive a new data stream for processing in each subsequent data stream iteration of load distribution on the node. The snapshot/restore method of data replication is usually used in high-load distributed web systems. Therefore, a hybrid system model with a centrally balanced node for load balancing and cloud network communication between computing nodes allows you to simply add more nodes by applying the cluster structure. The high-level communication between computing nodes brings problems to the data cloning method because, in the event of a failure of one of the nodes, it is necessary to recalculate the overall performance of the entire web system. Obviously, in this case, we can divide the problems related to computing node failures into two condition groups. The first group of software and hardware technical issues. If an error occurs in the software of the developed web system, because the redistribution of the calculation amount occurs at the level of the calculation module, if the calculation module at the top of the hierarchy is designed correctly, the system will automatically and quickly restore its performance and not get the level connection error [51, 53, 54].

Then it will warn other computing nodes about the error and redirect the computing information data flow to other modules. Therefore, in this case, there is no duplication of specific computing nodes on an independent platform, but only the launch of a standard instance. In order to significantly reduce the risk of losing a large number of user requests, the balanced computing node must include a buffer that will act as a "retry" when a node is physically unavailable. Therefore, the use of replication methods can provide the creation of typical computing nodes and at the same time restore the performance of the web system. However, when using cloud technology to develop program systems, there is a whole class of distributed algorithms, in which certain structures of information data streams (arrays, records) depend on the total number of interactive processes at the same time. The processing is performed by different computing nodes on independent platforms. An example of the application of this algorithm is a protocol for making coordinated decisions [34]. When the topology of the distributed and highly advantageous computing web system is modified, the structure of the data stream and its processing algorithm will also change.

Therefore, there is an urgent need to modify the behavior scheme of the simulation model used, that is, to modify the algorithm and structure of the data flow describing the behavior of the corresponding object.

4. The Realization of the Data Stream Processing Method to Restore the Computing Node

In the development of a high-load distributed web system, the architect consists of multiple computing nodes, and they are an instance of a single computing system, thus forming a “cluster.” However, the modified data stream processing algorithm not only needs to be separated at the physical level but also needs to be separated in the program. Traditionally, the software of the web system is divided into logical modules according to its functional purpose [33]. By designing a software web system, architects need to design so that the computing modules in the structure have as little logic as possible because the main

task of redistributing the load is to support system balance. An equally important requirement for the software architecture of web systems is to avoid strong connections. The strong connectivity between computing modules is that one of the modules is in contact with many other modules on the independent platform.

In other words, if a failure occurs, a large number of modules will send a message to the service module to notify other nodes that they need to be replaced temporarily. In order to effectively overcome possible calculation problems, you can use the following methods to restore the functions of the developed software web system using cloud technology: The clone_args method stores the values of the parameters that have been sent to the failed calculation module [3, 14, 38, 39, 49].

Since in this case, the module did not respond in a timely manner, it is classified as a failure, so the parent method of the module must obtain appropriate information about the parameters that have been sent to the child method. However, the direct function of the clone_args method does not stop there, especially the indirect attributes they involve are:

- Calculate the current integral processing time value of the module or the faulty module method.
- Fixed time for processing failures of computing nodes in the cloud web system.
- Calculate the start-up frequency of the module, which allows you to determine the demand for this module. Therefore, when distributing the load between nodes, you need to allocate the necessary software and hardware resources more efficiently.

The *reporting method* allows the service module to send statistical data about the information flow of processing all child nodes at certain intervals. Moreover, its use helps to automatically predict the degradation dynamics of the developed software web system. Therefore, the reporting method can not only be used to notify the statistical indicators in the plan but also can be used when one or more sub-computing modules fail.

The *restart method* provides the use of cloud technology for the developed web system program by restarting the computing module in the event of a failure. The main purpose of restarting is to avoid disruptive incoming data flow.

The *callback method* allows information messages about the successful or failed restart of the compute node to be sent to the parent module. This method is passed to the calculation module as a parameter. After starting or restarting the software web system that provides the "callback method," it should be checked whether the module is running stably. After analyzing the characteristics of errors that may occur in the calculation module of the developed program web system, we will notice several main reasons: errors in the program code, and high load in the problematic data flow and nodes.

The *feedback method* uses cloud technology to provide communication between the computing nodes of the developed program web system and is independent of each other, which is why it can be surrounded by the microservices of many web systems, and these microservices must exchange information flows with them. For this, you should of course notify the service computing module, which is at the top of the corresponding hierarchy. This is why each computing module adopts a feedback method that knows how to contact it and can redistribute the data flow from modules that have failed or are heavier than working nodes. The revised calculation model of the recovery method (the failed module) is shown in Fig. 2 [32].

This modified algorithm only implements a service module for processing data streams. Compared with traditional computing modules, the number of operations provided by the service modules is much smaller, and positive results can be obtained, which directly affects their work. In this case, the service module acts as "a stable part of the software system in a computing point" on a node on an independent computing platform.

Therefore, when developing a program web system, it is important to use methods to provide the smallest loopholes in this computing module.

These calculation methods will continuously notify the results of the work for a period, or unexpectedly notify the results of the work when leaving the work state. The service module has working information about all the software web systems developed. The less time the information flow about the

work results of the developed program web system is updated, the more accurately and effectively the necessary computing nodes can be defined to transfer its load.

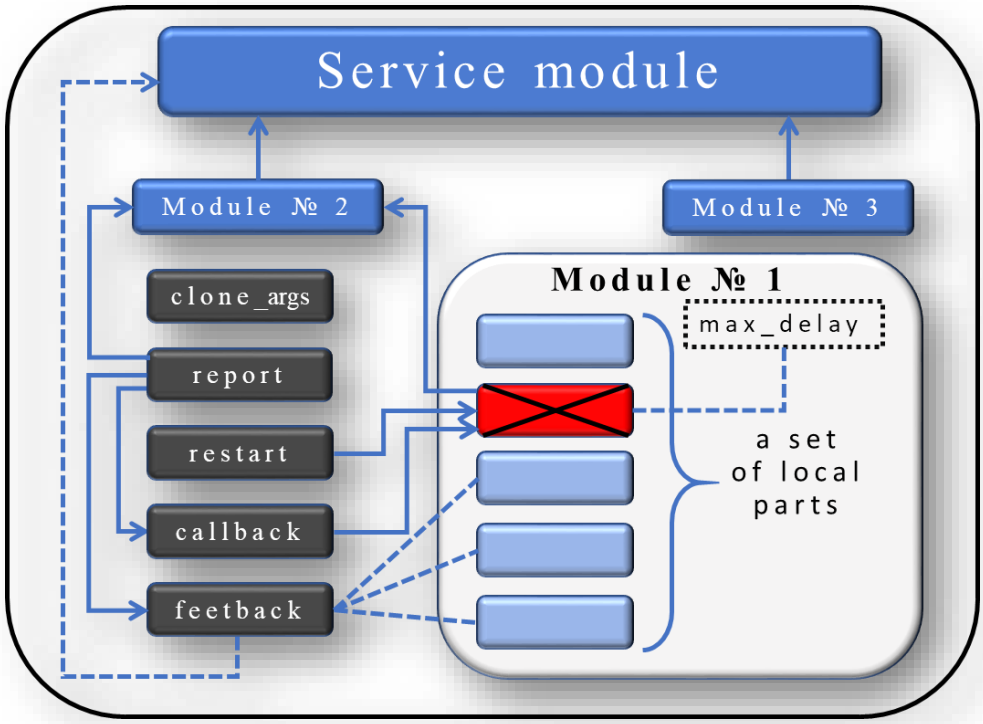

Figure 2: Load redistribution algorithm when a computing node fails in a software web system

Packing, state transfer, and data streaming. In the web system of the developed program using cloud technology, once a consensus is reached between the computing nodes that authorize and accept part of the computing work, it is necessary to prepare a package of raw data for the new node. And only after this, a connection can be established between the user module that failed to send the user request to another working module in another computing node. In addition, the service module will also send a message stating that the computing module is damaged—"temporarily unavailable to process new requests for data streams," and then perform the standard restart process. Due to the corresponding restart method, this process is implemented in each computing module and is also connected to the service module on the computing node through a feedback method. Once the software system receives the message that the computing module has failed, clone_args will copy all the input data streams sent for processing and a set of indirect data about the current state of the system. In this case, there is almost no time to make management decisions, so the service module may rather try to acquire a new computing node while copying the entire information call thread through the clone_args method and directing it to the service module.

When analyzing the sequence of the corresponding actions, it is obvious that it is necessary to form a list of processing calls, which are now waiting for other parent computing modules or their methods. However, this list is a kind of queue for processing user requests processed by the new work calculation module from the created queue. Amazon Simple Queue Service works according to the same principle-a service that can quickly receive user data flow queues. The service node combines this data and links the corresponding modules and their status messages for storage and processing. Once the compute node is available for information. In order to ensure the reliability of data streaming, it is necessary to divide the specific set of these data into appropriate fragments and add a header with a sequence number to each fragment. The fragments of the information data stream obtained in this way form a segment. In the next step of data processing, each segment enters a packet, and then reaches the recipient's computing node using the transmission information protocol. After the packet is delivered to the recipient's computing node, the correctness of the received information flow in this segment is checked by means of checksum

recalculation, and it is automatically determined that the previous information flow segment has also been successfully received Fig. 3.

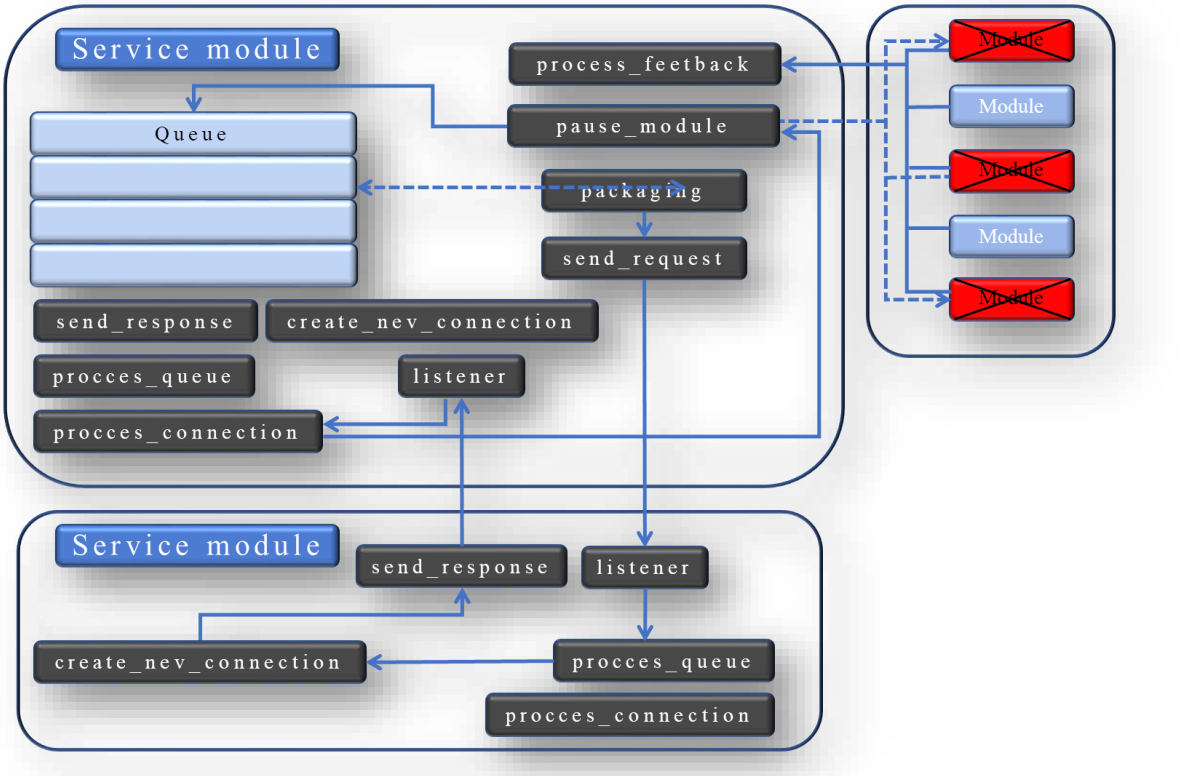

Figure 3: Packing, state transition, and data information flow model

At this stage, the recipient's computing node sends a request for information about new data packets or repeated transmissions of previous data packets to the sender's node. This operation algorithm ensures that all previous data packets in the data stream sequence have been successfully received. In the improved model developed, the computing modules are isolated from each other and do not have information about the overall state of the web system when establishing connections between them and receiving messages about their status. For the interaction of computing modules, we use the asynchronous exchange of information messages, where each corresponding module has its information message turn. Therefore, the computing module that sent the message will wait for the next notification about its delivery, otherwise, the receiver will ignore the message.

5. Conclusions

The system analysis performed considers the basic principles of distributed web system design and the technology most commonly used by the architects of the program system in the design. We also concluded that redundancy—an indispensable attribute in most web system development, redundancy is highly loadable relative to user requests. The main criterion for web system development is scalability. When the operation requires a large number of computing resources, its scalability will be used, which will greatly reduce the performance of the system and require it to increase the overall capacity. In addition, methods for estimating the reliability and fault tolerance of the developed program web system are studied because any programming system should be objectively monitored and predicted during work. Consider the basic aspects of the operation of distributed fail-safe software web systems because they have management problems and the ability to recover from failures during highly advantageous operations. By analyzing the technology used in the development of the software web system in the research, we improved the method and comprehensively analyzed the balance of the computing load based on this. This is the main task of designing the software

distributed fail-safe web system. In order to solve the problem of effective working of the developed software system, we used the algebraic method of optimizing the calculation of load balance on the node and its distributed network model, thereby creating a hybrid mechanism for information flow transmission. It can be considered to provide theoretical and practical foundations for the effective functions of the fault hierarchy mechanism. These foundations can provide effective calculation of the overall computing load of the program modules and nodes. By analyzing the high-load distributed fault-tolerant software web system, it can be determined that its basic criterion is the efficiency of processing an information data flow, which is usually calculated as the partial division of the sum of all output parameters and the sum. All input data streams. For each specific computing module or node, determine its efficiency value. The comparison of computational node efficiency is carried out using linear programming methods and using different basic models and their variants at the same time. It is recommended to determine the number of computing modules or nodes involved by constructing efficiency boundary criteria and all other conditions (its invalid criteria). The improved function of the network model is very different from the usual serial and parallel models. The main difference lies in the ability to gather and share important heterogeneous computing resource sets to develop geographically distributed information flows. In many cases, the development of software web systems brings considerable advantages and requires other resources that are not available in a computing node, and these resources can be obtained from other nodes connected to the functional grid. The research improves the algorithm for transferring computing load without using redundant resources, and proposes an improved interaction model between computing modules to handle the information flow within a single node, and proposes software developed as a whole web system. In addition, the technology to determine the cause of the failure of the computing module and the node is considered, and two basic problems related to the failure of the node are emphasized namely, software and hardware problems. Modified the computer process control model used for the development of information data flow, in which modules are isolated from each other and have no common state, but information communication can be established between them, and notifications about their status can be established. For the interaction of computing modules, asynchronous exchange of information messages is used, in which each module organizes its message wheel. The module sends a reference message, waits for confirmation of the corresponding delivery message, but does not wait for the receiver of the message to ignore it. An important aspect of this research is that the free computing resources of the developed program web system can be found within the scope of its functions.

6. References

[1] S. Aleti, Björnander, L. Grunske, I. Meedeniya, ArcheOpterix: An extendable tool for architecture optimization of AADL models, in: Proceedings of the 2009 ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software, MOMPES, 2009, pp. 73–77. doi:10.1109/MOMPES.2009.5069138.

[2] V. Andrunyk, A. Vasevych, L. Chyrun, N. Chernovol, N. Antonyu, A. Gozhyj, M. Korobchynskyi: Development of information system for aggregation and ranking of news taking into account the user needs, 2020. CEUR-WS.org, online CEUR-WS.ORG/Vol-2604/paper74.pdf.

[3] S. Babichev, V. Lytvynenko, J. Skvor, M. Korobchynskyi, M. Voronenk, Information technology of gene expression profiles processing for purpose of gene regulatory networks reconstruction, in: IEEE 2nd International Conference on Data Stream Mining and Processing, DSMP, 2018, pp. 336-–341. doi:10.1109/DSMP.2018.847845.

[4] B. Boehm, D. Rosenberg, N. Siegel, Critical Quality Factors for Rapid, Scalable, Agile Development, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2019, pp. 514–515. doi: 10.1109/QRS-C.2019.00101.

[5] C. Chen, M. Shoga, B. Boehm, Exploring the Dependency Relationships between Software Qualities, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2019, pp. 105–108. doi:10.1109/QRS-C.2019.00032.

[6] D. Ageyev, A. Mohsin, T. Radivilova, L. Kirichenko, Infocommunication Networks Design with Self-Similar Traffic, in: 2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), 2019, pp. 24–27. doi:10.1109/CADSM.2019.8779314.
[7] D. Ageyev, A. Mohsin, T. Radivilova, L. Kirichenko, Infocommunication Networks Design with Self-Similar Traffic, in: 2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), 2019, pp. 24–27. doi:10.1109/CADSM.2019.8779314.
[8] D. Ageyev, O. Bondarenko, T. Radivilova, W. Alfroukh, Classification of existing virtualization methods used in telecommunication networks, in: 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), 2018, pp. 83–86. doi:10.1109/DESSERT.2018.8409104.
[9] I. Dronyuk, I. Moiseienko, J. Gregus, Analysis of Creative Industries Activities in European Union Countries, Procedia Computer Science 160 (2019) 479–484. (2019). doi:10.1016/j.procs.2019.11.061.
[10] E. Awad, M. W. Caminada, G. Pigozzi, M. Podlaszewski, I. Rahwan, Pareto optimality and strategy-proofness in group argument evaluation, Journal of Logic and Computation 27 (2017) 2581–2609.
[11] P. Galkin, R. Umiarov, O. Grigorieva, D. Ageyev, Approaches for Safety-Critical Embedded Systems and Telecommunication Systems Design for Avionics Based on FPGA, in: 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019, pp. 391–396 (2019). doi:10.1109/PICST47496.2019.9061421.
[12] A. Hassan, Y. Jamalludin, Analysis of success factors of technology transfer process of the information and communication technology, in: 2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES), 2016, pp. 382–387. doi:10.1109/ICAEES.2016.7888074.
[13] M. Kabir, M. Rashed, Multi-level client server network and its performance analysis, LAP Lambert Academic Publishing, Saarbrücken, 2012.
[14] M. L. Abbott, M. T. Fisher, The art of scalability: scalable web architecture, processes, and organizations for the modern enterprise, 2nd ed., Addison-Wesley Professional, Boston, 2015.
[15] M. Pasyeka, V. Sheketa, N. Pasieka, S. Chupakhina, I. Dronyuk, System Analysis of Caching Requests on Network Computing Nodes, in: 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT). 2019, pp. 216–222. doi:10.1109/AIACT.2019.8847909.
[16] M. O. Medykovskyy, M. S. Pasyeka, N. M. Pasyeka, O. B. Turchyn, Scientific research of life cycle perfomance of information technology, in: 2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 2017, pp. 425–428 (2017). doi:10.1109/STC-CSIT.2017.8098821.
[17] M. Nazarkevych, A. Marchuk, L. Vysochan, Y. Voznyi, H. Nazarkevych, A. Kuza: Ateb-Gabor Filtering Simulation for Biometric Protection Systems, in: Conference on Computer Science and Information Technologies, 2020. CEUR-WS.org, online CEUR-WS.ORG/Vol-2746/paper2.pdf
[18] O. Mishchuk, R. Tkachenko, I. Izonin, Missing Data Imputation through SGTM Neural-Like Structure for Environmental Monitoring Tasks, in: Advances in Intelligent Systems and Computing, 2020, pp. 142–151. doi:10.1007/978-3-030-16621-2_13.
[19] M. A. J. Idrissi, H. Ramchoun, Youssef Ghanou and Mohamed Ettaouil, Genetic algorithm for neural network architecture optimization, in: 3 International Conference on Logistics Operations Management (GOL), 2016, pp. 1–4.
[20] H. Mykhailyshyn, N. Pasyeka, V. Sheketa, M. Pasyeka, O. Kondur, M. Varvaruk, Designing Network Computing Systems for Intensive Processing of Information Flows of Data, in: T. Radivilova, D. Ageyev, N. Kryvinska, (Eds.), Data-Centric Business and Applications: ICT Systems-Theory, Radio-Electronics, Information Technologies and Cybersecurity , volume 5, Springer International Publishing, Cham, 2021, pp. 391–422. doi:10.1007/978-3-030-43070-2_18.
[21] N. Pasieka, V. Sheketa, Y. Romanyshyn, M. Pasieka, U. Domska and A. Struk, Models, Methods and Algorithms of Web System Architecture Optimization, in: IEEE International Scientific-

Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 2019, pp. 147–153. doi:10.1109/PICST47496.2019.9061539.
[22] N. Pasyeka, H. Mykhailyshyn and M. Pasyeka, Development algorithmic model for optimization of distributed fault-tolerant web-systems, in: IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T'2018), 9-12 October, Kharkiv, 2018, pp. 663–669.
[23] M. Nazarkevych, M. Logoyda, O. Troyan, Y. Vozniy, Z. Shpak, The Ateb-Gabor Filter for Fingerprinting, in: International Conference on Computer Science and Information Technology, Springer, Cham, 2019, pp. 247–255.
[24] M. Nazarkevych, M. Logoyda, S. Dmytruk, Y. Voznyi, O. Smotr: Identification of biometric images using latent elements, 2019. CEUR-WS.org, online CEUR-WS.ORG/Vol-2488/paper8.pdf
[25] M. Nazarkevych, N. Lotoshynska, I. Klyujnyk, Y. Voznyi, S. Forostyna, I. Maslanych, Complexity Evaluation of the Ateb-Gabor Filtration Algorithm in Biometric Security Systems, in: 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), 2019, pp. 961–964.
[26] M. Nazarkevych, N. Lotoshynska, V. Brytkovskyi, S. Dmytruk, V. Dordiak, I. Pikh, Biometric identification system with ateb-gabor filtering, in: 11th International Scientific and Practical Conference on Electronics and Information Technologies (ELIT), 2019, pp. 15–18. doi:10.1109/ELIT.2019.8892282.
[27] O. Riznyk, Yu. Kynash, O. Povshuk and V. Kovalyk, Recovery schemes for distributed computing based on bib-schemes, in: First International Conference on Data Stream Mining & Processing (DSMP), 2016, pp.134–137.
[28] P. Ravi, V. K. Sundar, A. Chattopadhyay, S. Bhasin, A. Easwaran, Authentication Protocol for Secure Automotive Systems: Benchmarking Post-Quantum Cryptography, in: IEEE International Symposium on Circuits and Systems (ISCAS), Sevilla, 2020, pp. 1–5. doi: 10.1109/ISCAS45731.2020.9180847.
[29] N. Pasieka, V. Sheketa, Y. Romanyshyn, M. Pasieka, U. Domska, A. Struk, Models, methods and algorithms of web system architecture optimization, in: IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S&T, 2019, 147–152. doi:10.1109/PICST47496.2019.9061539.
[30] M. Pasyeka, V. Sheketa, N. Pasieka, S. Chupakhina, I. Dronyuk, System analysis of caching requests on network computing nodes, in: 3rd International Conference on Advanced Information and Communications Technologies, AICT, 2019, pp. 216–222. doi:10.1109/AIACT.2019.8847909.
[31] M. Pasyeka, T. Sviridova, I. Kozak, Mathematical model of adaptive knowledge testing, in: 5th International Conference on Perspective Technologies and Methods in MEMS Design, MEMSTECH, 2009, pp. 96–97.
[32] M. Pasyeka, V. Sheketa, N. Pasieka, S. Chupakhina, I. Dronyuk, System analysis of caching requests on network computing nodes, in: 3rd International Conference on Advanced Information and Communications Technologies, AICT, 2019, pp. 216–222. doi:10.1109/AIACT.2019.8847909.
[33] Q. Linling, W. Qingfeng, Research on Automatic Test of WEB System Based on Loadrunner, in: 13th International Conference on Computer Science & Education, Sri Lanka, 2018, pp. 1–4. doi:10.1109/ICCSE.2018.8468852.
[34] R. Alléaume, I. P. Degiovanni, A. Mink, T. E. Chapuran, N. Lutkenhaus, M. Peev, C. J. Chunnilall, V. Martin, M. Lucamarini, M. Ward, A. Shields, Worldwide standardization activity for quantum key distribution, in: 2014 IEEE Globecom Workshops (GC Wkshps), 2014, pp. 656–661. doi:10.1109/GLOCOMW.2014.7063507.
[35] R. Paul, J. R. Drake, H. Liang, Global Virtual Team Performance: The Effect of Coordination Effectiveness, Trust, and Team Cohesion, IEEE Transactions on Professional Communication 59 (2016) 186–202. doi:10.1109/TPC.2016.2583319.
[36] R. Privman, S. R. Hiltz, Y. Wang, In-Group (Us) versus Out-Group (Them) Dynamics and Effectiveness in Partially Distributed Teams, IEEE Transactions on Professional Communication 56 (2013) 33–49. doi:10.1109/TPC.2012.2237253.

[37] O. Riznyk, O. Povshuk, Y. Kynash, M. Nazarkevich, I. Yurchak, Synthesis of non-equidistant location of sensors in sensor network, in: 14th International Conference on Perspective Technologies and Methods in MEMS Design, MEMSTECH, 2018, pp. 204–208. doi:10.1109/MEMSTECH.2018.8365734.
[38] L. Sikora, N. Lysa, B. Fedyna, B. Durnyak, R. Martsyshyn, Y. Miyushkovych, Technologies of Development Laser Based System for Measuring the Concentration of Contaminants for Ecological Monitoring, in: 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 2018, pp. 93–96. doi:10.1109/STC-CSIT.2018.8526602.
[39] L. Sikora, R. Martsyshyn, Y. Miyushkovych, N. Lysa, B. Yakymchuk, Problems of data perception by operators of energy-active objects under stress, in: The Experience of Designing and Application of CAD Systems in Microelectronics, 2015, pp. 475–477. doi:10.1109/CADSM.2015.7230909.
[40] T. Aslam, T. Rana, M. Batool, A. Naheed, A. Andaleeb, Quality Based Software Architectural Decision Making, in: 2019 International Conference on Communication Technologies (ComTech), 2019, pp. 114–119. doi:10.1109/COMTECH.2019.8737836.
[41] R. Tkachenko, I. Izonin, N. Kryvinska, I. Dronyuk, K. Zub, An Approach towards Increasing Prediction Accuracy for the Recovery of Missing IoT Data based on the GRNN-SGTM Ensemble, Sensors 20 (2020). doi:10.3390/s20092625.
[42] R. Tkachenko, I. Izonin, P. Vitynskyi, N. Lotoshynska, O. Pavlyuk, Development of the non-iterative supervised learning predictor based on the ito decomposition and sgtm neural-like structure for managing medical insurance costs, Data 3 (2018). doi:10.3390/data3040046.
[43] U. Banerjee, A. Pathak, A. P. Chandrakasan, An Energy-Efficient Configurable Lattice Cryptography Processor for the Quantum-Secure Internet of Things, in: IEEE International Solid- State Circuits Conference - (ISSCC), San Francisco, CA, USA, 2019, pp. 46–48. doi:10.1109/ISSCC.2019.8662528.
[44] V. B. Dang, F. Farahmand, M. Andrzejczak, K. Gaj, Implementing and Benchmarking Three Lattice-Based Post-Quantum Cryptography Algorithms Using Software/Hardware Codesign, in: International Conference on Field-Programmable Technology (ICFPT), Tianjin, China, 2019, pp. 206–214. doi: 10.1109/ICFPT47387.2019.00032.
[45] V. Drăgoi, T. Richmond, D. Bucerzan and A. Legay, Survey on cryptanalysis of code-based cryptography: From theoretical to physical attacks, in: 7th International Conference on Computers Communications and Control (ICCCC), Oradea, 2018, pp. 215–223. doi:10.1109/ICCCC.2018.8390461.
[46] W. Liu, J. Yang, Y. Song, X. Yu and S. Zhao, Research on Software Quality Evaluation Method Based on Process Evaluation and Test Results, in: 6-th International Conference on Dependable Systems and Their Applications (DSA), Harbin, China, 2020, pp. 480–483. doi: 10.1109/DSA.2019.00077.
[47] Y. Romanyshyn, V. Sheketa, L. Poteriailo, V. Pikh, N. Pasieka, Y. Kalambet, Social-communication web technologies in the higher education as means of knowledge transfer, in: IEEE 2019 14th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2019, pp. 35–39.
[48] Y. Gao-Wei, H. Zhanju, A Novel Atmosphere Clouds Model Optimization Algorithm, in: International Conference on Computing, Measurement, Control and Sensor Network, 2012 pp. 217–220.
[49] Q. Linling, W. Qingfeng, Research on Automatic Test of WEB System Based on Loadrunner, in: 13th International Conference on Computer Science & Education, Sri Lanka, 2018, pp. 1–4, doi:10.1109/ICCSE.2018.8468852.
[50] M. Zharikova, V. Sherstjuk, Academic integrity support system for educational institution, in: IEEE 1st Ukraine Conference on Electrical and Computer Engineering, UKRCON, 2017, pp. 1212–1215. doi:10.1109/UKRCON.2017.8100445.
[51] M. Zharikova, V. Sherstjuk, N. Baranovskiy, The plausible wildfire model in geoinformation decision support system for wildfire response, in: Paper presented at the International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM, 2015, pp. 575–584.