The object of research is the process of load distribution in the edge environment of the Internet of Things.

The task to improve the efficiency of the functioning of the network of computing devices in the Internet of Things edge environment has been solved. Free resources of heterogeneous single-board computers were used to this end.

In the process of conducting research, an approach to the construction of an architecture for a virtual cluster of computers with limited resources was devised. The design took into account specific features of the edge environment on the Internet of Things. This has made it possible to propose a four-layer architecture instead of the standard seven-layer architecture of IoT sensor information processing device networks.

Stages in the virtual cluster construction in the edge environment on the Internet of Things were also defined. A three-stage procedure to form a virtual cluster was justified. This procedure made it possible to devise a method for the virtual clustering in the Internet of Things edge environment based on the proposed virtual cluster architecture.

The proposed method for building a virtual cluster in the Internet of Things edge environment was investigated. With a small network load, a virtual cluster has no advantage over a classic cluster. But with the growth of the network load, the virtual cluster prevails over the classic cluster in total performance; the advantage in total performance can exceed 10 %. It was also proven that for a heterogeneous environment, performance changes at full network load significantly depend on the number of virtual node groups. The research results on the method for building a virtual cluster in the Internet of Things edge environment can be explained by improving the balance of the network load at virtual clustering

Keywords: Internet of Things, virtual cluster, edge environment, heterogeneity, fuzzy computing, balance

UDC 004.41:519.7
DOI: 10.15587/1729-4061.2024.298431

DEVISING A METHOD FOR THE VIRTUAL CLUSTERING OF THE INTERNET OF THINGS EDGE ENVIRONMENT

Heorhii Kuchuk
Doctor of Technical Sciences, Professor*
Oleksandr Mozhaiev
Corresponding author
Doctor of Technical Sciences, Professor
Department of Cyber Security and DATA Technologies**
E-mail: mozhaev1957@gmail.com
Nina Kuchuk
Doctor of Technical Sciences, Professor*
Serhii Tiulieniev
PhD
Director
Scientific Research Center for Forensic Science of Information Technologies and Intellectual Property of the Ministry of Justice of Ukraine
L. Ukrainka blvd., 26, Kyiv, Ukraine, 01133
Mykhailo Mozhaiev
Doctor of Technical Sciences, Head of Laboratory***
Yurii Gnusov
PhD, Associate Professor
Department of Cyber Security and DATA Technologies**
Mykhailo Tsuranov
Senior Lecture
Department of Cyber Security and DATA Technologies**
Tetiana Bykova
Acting Head of Laboratory ***
Sergii Klivets
PhD
Science Center
Ivan Kozhedub Kharkiv National Air Force University
Sumska str., 77/79, Kharkiv, Ukraine, 61023
Alexander Kuleshov
PhD, Associate Professor
Science Center
Ivan Kozhedub Kharkiv National Air Force University
Sumska str., 77/79, Kharkiv, Ukraine, 61023
*Department of Computer Engineering and Programming
National Technical University "Kharkiv Polytechnic Institute"
Kyrpychova str., 2, Kharkiv, Ukraine, 61002
**Kharkiv National University of Internal Affairs
L. Landau ave., 27, Kharkiv, Ukraine, 61080
***Laboratory of Copyright and Information Technologies
Scientific Research Center for Forensic Science of Information Technologies and Intellectual Property of the Ministry of Justice of Ukraine
L. Ukrainka blvd., 26, Kyiv, Ukraine, 01133

Received date 17.11.2023
Accepted date 08.02.2024
Published date 28.02.2024

How to Cite: *Kuchuk, H., Mozhaiev, O., Kuchuk, N., Tiulieniev, S., Mozhaiev, M., Gnusov, Y., Tsuranov, M., Bykova, T., Klivets, S., Kuleshov, A. (2024). Devising a method for the virtual clustering of the internet of things edge environment. Eastern-European Journal of Enterprise Technologies, 1 (9 (127)), 60–71. doi: https://doi.org/10.15587/1729-4061.2024.298431*

1. Introduction

The Internet of Things (IoT) is one of those areas that is increasingly becoming part of everyday life and is showing rapid growth in technology. The number of devices connected to the Internet has already significantly exceeded the population of the Earth [1]. The global IoT market is constantly increasing [2]. IoT is used in many fields, such as construction [3], industry [4], monitoring the state of complex systems [5], health care [6].

IoT is based on cloud computing technology [7]. However, in recent years, difficulties have arisen as a result of the following factors [8]:

– the presence of geographical distribution of IoT components;

– an increase in network delays;

– high cost of communication channels;

– availability of mobility of end devices.

Most of these difficulties were solved by introducing an additional layer called fuzzy calculations [4]. Fuzzy calculations brought data processing closer to end devices of IoT networks but did not solve the problem of processing and transmitting operational information.

One of the solutions to this issue is the development of edge calculation technologies (Edge Calculations, EC). ECs make it possible to take part of the load on them and reduce the response time to an emerging event. In the last 5–6 years, single-board computers have been used for the organization of edge calculations. Such computers have compact dimensions, high energy efficiency, and low cost. But they demonstrate low performance, which does not allow solving complex computing tasks using the resources of one device. One of the solutions to this problem is to build a local cluster consisting of computers with limited computing resources. Individual elements of such a cluster do not contain additional computing expanders, coprocessors for computing special tasks. At the same time, computers and clusters involved in edge calculations often perform certain tasks and do not utilize the full potential of these devices. Although such components are computers with limited computing resources, they can also be used to solve other tasks. The task of using free computing resources is especially important when overloading the network of the edge IoT environment.

Most networks within the IoT edge environment have become significantly heterogeneous. This was facilitated by the rapid growth in the production of single-board computers, a decrease in their cost, and an increase in the number of different modifications. Constructing a virtual cluster (VC) is one of the ways of using the free resources of different heterogeneous devices. VC unites and organizes the devices of the existing infrastructure, which are in the field of edge calculations, primarily designed to use the resources of single-board computers. Such a cluster could also make it possible to employ resources of existing infrastructure more rationally, for example, by deploying additional data processing and storage services.

Therefore, in order to improve the efficiency of the network of computing devices within the IoT edge environment, it is necessary to utilize the free resources of single-board computers, taking into account their significant heterogeneity. Therefore, the issue of devising a method for the virtual clustering of the IoT edge environment is becoming more relevant.

2. Literature review and problem statement

The issue of resource redistribution in any computer network is directly related to the task of transferring and parallelizing the load. Such a task usually includes two stages:

– selection of load placement nodes;

– solving the load placement problem.

In centralized systems, these tasks are performed by a central control node or device. In distributed decentralized systems, the execution of these tasks is much more difficult. Let's consider some scientific works on this topic that can be applied to edge calculations.

In [9], the task of resource allocation for operational tasks of cloud and edge calculations is considered. But only homogeneous environments are considered.

In work [10], the task of adaptive allocation of resources is considered only for centralized systems. Therefore, the proposed method of resource redistribution can be applied in the cloud layer but not in the edge layer. Similarly, the two-stage method of optimal redistribution of resources proposed in [11] is not adapted to the edge cloud.

The method of resource planning in distributed systems, proposed in [12], can be used both in fuzzy and edge environments. But this method is focused only on simple topologies. Therefore, it cannot be used for significantly heterogeneous systems.

The algorithm proposed in [13] for scheduling requests of IoT devices on the edge and fuzzy layers optimizes performance and energy consumption. But as in previous papers, this algorithm does not take into account the heterogeneity of the environment. Similar problems arise when applying the algorithm given in [14].

The algorithm given in [15] is focused on transferring the computing load to reduce delays when performing tasks in distributed systems. But this algorithm is not oriented to take into account the characteristics of the computational task, in particular, the computation time in the edge environment.

The methods proposed in [16, 17] are focused on load balancing in distributed systems. But they do not take into account the most important characteristic of edge layer devices – limited resources.

In works [18, 19], a multi-level architecture of predictive task planning is proposed, which is focused on reducing delays in decentralized systems. But this approach does not take into account the costs associated with data transmission.

Algorithms for offloading nodes of the IoT functioning network are proposed in [20, 21]. But they are used only in a homogeneous environment.

The algorithms proposed in [22, 23] can be applied to the heterogeneous environment of IoT support networks. But these algorithms are focused only on structures similar to the structures of the cloud environment.

The distributed resource allocation algorithm proposed in [24] based on deep learning is focused on edge calculations. But this algorithm is effective only for a homogeneous environment, similar to the algorithm used in [25].

A strategy for distributing computations for collaborative work, based on deep learning with the pooling of resource capabilities, is proposed in [26]. But it cannot be implemented on the edge layer with limited device capabilities.

Therefore, the above scientific works do not sufficiently take into account the characteristic features of the edge environment of IoT when allocating computing resources. Therefore, the computing resources of network devices will not be fully utilized, especially in a heterogeneous environment. It is possible to eliminate a significant load imbalance of edge devices through network virtualization. At the same time, it will be necessary to form virtual groups of devices that are similar in basic characteristics. Therefore, it is reasonable to devise a method for the virtual clustering of the IoT edge environment.

3. The aim and objectives of the study

The purpose of our work is to devise a method for building a virtual cluster in the edge environment of the Internet of Things. This will make it possible to improve the performance of the network of edge devices with limited resources by increasing the balance of the load of network nodes.

To achieve the goal, the following tasks were set:

– to devise an approach to designing an architecture for a virtual cluster of computers with limited resources;

– to define stages of virtual cluster construction in the edge environment of the Internet of Things;

– to devise and research a method for building a virtual cluster of the edge environment of the Internet of Things.

4. The study materials and methods

The object of our research is the process of load distribution in the edge environment of the Internet of Things. The work considers devices that have limited computing resources and are components of nodes in a heterogeneous environment. Such devices are usually used at the edge layer, for example, they can be single-board computers like the Raspberry Pi.

The process of devising a method for constructing a virtual cluster in the edge IoT environment involved working with heterogeneous devices. When devising the method, the following conditions were used:

Condition 1. The IoT edge environment is significantly heterogeneous.

Condition 2. IoT edge environment nodes are subordinate to one master node and make up one physical cluster.

Condition 3. Virtually all IoT edge environment nodes have limited computing resources.

Condition 4. Part of the tasks involves parallelization when executing subtasks.

A number of different methods were used in the process of virtualizing the IoT edge environment.

When building a virtual cluster, each node is assigned a predefined role. Choosing the most optimal devices for each of the required roles is a typical task of multicriteria optimization on a finite set. To solve such problems, the following optimality criteria were used: Pareto optimality and Slater optimality [27].

Slater's principle was applied in the cases of clear superiority of one or more devices over the rest at once in all target characteristics. If there were no clear leaders among the devices according to all criteria, then the Pareto principle was used.

When building a virtual group of nodes, data clustering algorithms were used without a trainer. Three methods were considered: K-Means, agglomerative clustering, spectral clustering [28].

The K-Means algorithm is one of the simplest and most common clustering methods. Let's introduce the metric of the distance between two points of the considered space – $d(\bullet, \bullet)$. Then, according to the algorithm, it is necessary to minimize the total quadratic deviation of cluster points from the centers of these clusters:

$$D=\sum_{i=1}^{k}\sum_{x\in S_k}\left(d\left(x,\mu_i\right)\right)^2\to\min, \tag{1}$$

where k is the number of clusters, S_i is the set of points of the ith cluster, μ_i is the center of mass of the points of the ith cluster, $d(\bullet, \bullet)$ is the metric of the distance between two points of the considered space.

The main problem of this method is that the number of clusters must be known in advance. To solve it, a method based on the "Silhouette" quality criteria was used. Let us have a fixed partition η_k of the given space of points into k clusters. The partition consists of k disjoint sets C_i, each of which describes all points of the ith cluster ($1\le i\le k$). For each point of the space $x_j\in C_i$, we shall introduce the characteristics of compactness and separation.

The normalized value of compactness is calculated according to the following formula:

$$\mathrm{comp}\left(x_j\in C_i\right)=\frac{\sum_{x\in C_i}d\left(x_j,x\right)}{\mathrm{card}\left(C_i\right)}, \tag{2}$$

where $\mathrm{card}(C_i)$ is the number of points in the ith cluster.

The normalized value of separation is calculated according to the following formula:

$$\mathrm{sep}\left(x_j\in C_i\right)=\min_{\ell\neq i}\frac{\sum_{x\in C_\ell}d\left(x_j,x\right)}{\mathrm{card}\left(C_\ell\right)}. \tag{3}$$

Then the value of the silhouette of the $x_j\in C_i$ space point is calculated as:

$$\mathrm{Sil}\left(x_j\in C_i\right)=\frac{\mathrm{sep}\left(x_j\right)-\mathrm{comp}\left(x_j\right)}{\max\left(\mathrm{sep}\left(x_j\right);\mathrm{comp}\left(x_j\right)\right)}. \tag{4}$$

Accordingly, the silhouette value of cluster C_i is calculated as follows:

$$\mathrm{Sil}\left(C_i\right)=\frac{\sum_{j\in C_i}\mathrm{Sil}\left(x_j\right)}{\mathrm{card}\left(C_i\right)}. \tag{5}$$

Now you can determine the silhouette value for the partition η_k of the given point space:

$$\mathrm{Sil}\left(\eta_k\right)=\frac{\sum_{i=1}^{k}\mathrm{Sil}\left(C_i\right)}{k}. \tag{6}$$

The greater the value of the silhouette of the partition of the given space of points into clusters, the better the clustering is carried out. So, to find the desired number of clusters k_0 for the K-Means method, you need to find the partition with the largest silhouette:

$$k_0=\left(m\middle|\mathrm{Sil}\left(\eta_m\right)>\mathrm{Sil}\left(\eta_\ell\right)\forall\ell\neq m,1\le\ell\le k\right). \tag{7}$$

A distinctive feature of this method of agglomerative clustering compared to the previous one is the absence of requirements for determining the number of clusters. The algorithm of the method is described in the following steps:

Step 1. A cluster is built for each point.

Step 2. Sorting of pairwise distances of elements between cluster centers by growth is performed.

Step 3. The nearest clusters are merged into one. We calculate the new cluster center.

Step 4. Steps 2 and 3 are repeated cyclically until the termination condition is met.

The spectral clustering method is based on the concept of graph connectivity. Unlike previous methods that look for dense, compact, convex clusters, spectral clustering can find clusters of arbitrary shape. The first step of the algorithm of

the spectral clustering method is to determine the matrix of similar elements in relation to the "similarity" of the elements. This matrix describes a complete graph with vertices of elements and edges connecting them. The basic algorithm is as follows:

Step 1. The normalized Laplacian is calculated.

Step 2. The first k eigenvectors are calculated.

Step 3. The matrix built by the first k eigenvectors is constructed.

Step 4. The graph nodes are clustered based on this matrix.

Sometimes, if it was necessary to reduce the dimensionality of the data, the method of principal components was used in the work. To reduce the dimensionality from n to k, $k<n$, it is necessary to choose k axes, sorted by decreasing variance along the axes.

The first step is to calculate the variance and covariance of the original elements using the covariance matrix. As a further step, we decompose the covariance matrix as a product of direct and transposed matrices. From the Rayleigh relation, we determine that the maximum variation is reached along the eigenvector of the matrix, which corresponds to the maximum eigenvalue. So, we choose k eigenvectors of the matrix, which will be the main components. To obtain the data projection in the orthogonal basis of the components, it is necessary to multiply the data matrix by these components. We transpose the data matrix and the matrix of vectors of the main components. If, as a result of transformations, the number of components was less than the dimension of the original space, then part of the information is lost. But thanks to this method, a minimal amount of information is lost.

To evaluate the clustering results, three efficiency metrics were considered. The Silhouette Coefficient metric is determined for each sample and consists of two estimates. In this metric, a higher coefficient refers to a model with better defined data. The Calinski-Harabasz Index metric is known as the deviation ratio criterion. This metric represents the ratio of the sum of variance between clusters and variance within clusters for all clusters. The clustering result is better, the larger the value of the criterion. The Davies-Bouldin Index metric determines the average "similarity" between clusters. In it, similarity is a measure that compares the distance between clusters with the size of the clusters themselves. A lower measure index refers to a model with a better distribution of clusters.

When building the computing layer of virtual groups of nodes, the heterogeneity of devices and their selection according to approximately the same characteristics and performance are taken into account. In addition, it is necessary to determine the potential acceleration of the algorithm when the number of processors increases. For this purpose, Amdahl's weighted law was used [29]. For heterogeneous computing nodes, it takes into account network performance and bandwidth. Then the best configuration of the virtual node group is the one in which the performance of the cluster with the minimum increase in performance is the maximum.

The empirical filling method was used to fill the virtual cluster with computing nodes. This approach involves filling the virtual cluster with computing nodes until the condition limiting the growth of computing acceleration is met. A limitation may be an increase in the speed of execution of the test task. Thus, it is possible to configure the best performance of a virtual cluster for a given task with heterogeneous devices. The disadvantage of this method is the need to select virtual cluster nodes for a new task each time.

The meta-learning approach is based on previously known combinations of devices and the necessary parameters for building a virtual cluster [30]. At the same time, the selection of virtual computing nodes is based on the templates of tested devices in the cluster for a predetermined algorithm. When accumulating a large amount of test data, this method makes it possible to build a virtual cluster from existing devices, focusing on the most optimal templates. This will make it possible to quickly assemble effective virtual clusters for current tasks and also quickly adapt to new ones. The disadvantage of this approach is the need to conduct versatile tests on a large number of combinations of devices. In addition, when adding a new model, new tests must be conducted.

The client program SSHFS (Secure SHell FileSystem) was used to construct the directory of the remote machine as a file system.

API Rest (Representational State Transfer) technology was used when designing API services to connect images, video images, text data, and time series data.

5. Results of devising and researching the method of virtual clustering of the edge environment on the Internet of Things

5. 1. Construction of the architecture for a virtual cluster of computers with limited resources

In 2018, a general seven-layer IoT architecture was proposed [31], but depending on the task, the number of layers can vary from three to seven. Let's determine which layers are required to build a virtual cluster on the edge IoT environment.

The architecture for a virtual cluster assumes the use of computers with limited computing resources that are not uniform in their characteristics. The presence of heterogeneity requires solving the problem of selecting devices similar in characteristics and building a virtual cluster from them. Since several clusters can be formed, the task of designing their management layer follows. In addition, it is necessary to provide disk space for storing files on computers with limited computing resources. Microservices are used in the designed architecture for the virtual cluster. They assume the use for each type of data of a separate microservice, independent of the rest of the others. This condition makes it possible by replacing the module to enable the adaptability and configuration of the system for receiving different types of data. The main feature of the designed architecture is the construction of all layers of the cluster on the basis of external devices that can be connected. In addition, it is necessary to select configurations according to the specified parameters to enable the performance of the computational task.

Therefore, the distributed architecture of a virtual heterogeneous cluster (AVHC) should include four layers (Fig. 1):

– layer of physical devices (AL, Application Layer),

– cluster construction and management layer (PrL, Processing Layer),

– cluster coordination layer (NL, Network Layer),

– the layer of the virtual group of cluster devices (PeL, Perception Layer).

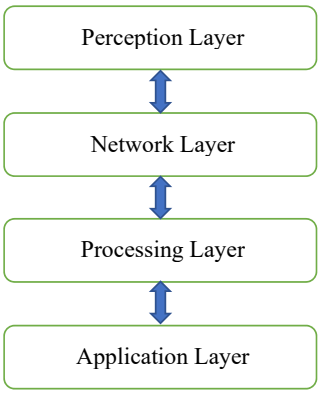

Fig. 1. Virtual heterogeneous cluster architecture

At the same time, it should be noted that the main feature of computers with limited computing resources is the lack of resources of one computer to solve a computationally complex task. Therefore, the possibility of scaling the data storage parameter is considered. Scaling makes it possible to use resources of multiple computers to provide enough of the required resource. In most cases, scaling is used to store processed files and received results on the coordination layer of the virtual cluster.

Let's consider these layers of the virtual cluster in more detail.

The bottom layer of the architecture is the physical device layer. This layer contains heterogeneous single-board computers with a preinstalled Linux operating system. Client software for connecting to the cluster server via heterogeneous communication channels (Ethernet, Wi-Fi) is also installed. Based on these devices, a virtual cluster and cluster coordination layer are built.

Direct communication with the lower layer is carried out by the cluster construction and management layer. This layer provides the following functions:

– registration, testing of performance and quality characteristics of nodes;

– selection of devices for cluster construction;

– management of a virtual cluster using the appropriate module.

This layer is on a separate device. In the edge IoT environment, this device is usually a single-board computer based on the Linux operating system.

The main components of the cluster construction and management layer and the relationships between them are shown in Fig. 2.

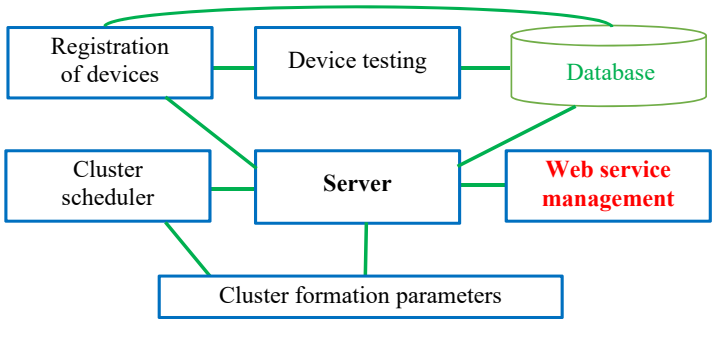

Fig. 2. Main components of the cluster construction and management layer

Let's take a closer look at each of the components of this layer.

The "Server" component is responsible for the operation and launch of all system components, their interaction with each other and with the operating system. In addition, this component is responsible for connecting/registering new devices. The server is assigned and configured manually by the user.

The "Device Registration" component makes it possible to make up a list of all active devices connected to the server. This component stores information about all the devices involved and the roles assigned to them. The list of registered devices is stored in the database. If the device has not been tested, a corresponding flag is added.

The Device Performance Testing component is designed to determine the performance of IoT edge environment nodes. Each new device undergoes performance testing on various computing tasks, such as:

– testing the speed of data transmission over the network;

– device testing in single-stream and multi-stream modes;

– testing of the RAM device for read/write speed;

– file compression testing in single-stream and multi-stream modes.

After passing these tests, a record of the test results is added to each device, in addition to its characteristics. The newly built list is stored in the database for use by other system components.

The "Cluster Planner" component is used to assign the user the following initial parameters for the construction of a virtual group of nodes:

– parameters for the coordination cluster indicating the required disk space;

– the minimum number of coordination nodes;

– the number of virtual clusters and computing nodes in them;

– a list of modules for loading virtual clusters that are being formed.

The virtual group of nodes is built taking into account the selection of the most suitable nodes in terms of average characteristics to enable the performance of the layer and cluster.

The "Cluster construction parameters" component forms the coordination layer of the cluster and virtual groups of nodes based on available devices and specified parameters. The results of the preliminary construction of the virtual cluster are transferred to the "Web service management" component. If there are free, unassigned nodes, you can distribute them in the current configuration or assemble a new one. In particular, you can assign the role of redundant nodes to quickly replace failed nodes.

The "Web Service Management" component makes it possible to get up-to-date information about devices, their characteristics, assigned role and status in multiple virtual clusters.

Let's move on to consider the structure of the next layer.

The coordination layer of the cluster is responsible for the following functions:

– management of virtual clusters, monitoring of their status;

– organization of virtual memory for storing data received for processing;

– results of calculations from virtual clusters.

This layer contains one coordination node that interacts with the main nodes of virtual clusters. In addition, this layer controls the organization of data storage. The coordination node is the connecting link between the main server and the virtual clusters. It acts as an information aggregator that collects data about the state of the virtual cluster. It transmits information about work results to the main server, where the user can observe the system's operation through the web interface.

The main components of the coordination layer of the cluster and the relationships between them are shown in Fig. 3.

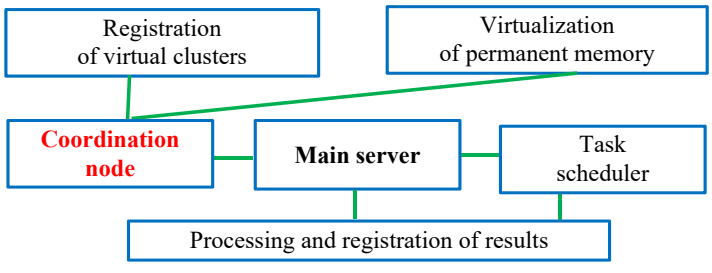

Fig. 3. The main components of the cluster coordination layer

Let's take a closer look at each of the components of this layer.

The "Main Server" component is given appropriate instructions for downloading and installing the necessary modules of the coordination node. This module enables the functionality of all components of the layer, their connection and interaction with other components of the system.

The "Registration of virtual clusters" component contains information about:

– all virtual clusters;

– assigned devices and assigned roles;

– allocated disk space for data storage.

This component makes it possible to receive information about the status of each device and its interaction with the system by logging the performed operations. Information about the nodes of the formed virtual cluster allows it to be reformed in case of node failure. In addition, a virtual cluster can be supplemented with computing nodes to increase cluster performance.

Let's consider the functions of the "Persistent memory virtualization" component. Computers with limited computing resources have various characteristics in a number of parameters, including disk space. Therefore, the task of organizing data storage on the basis of such computers is to provide the necessary space on the basis of permanent memory. Devices with the following parameters are selected to organize data storage:

– maximum free disk space;

– high speed of data transmission over the network;

– minimum performance compared to other devices registered in the system.

The device with the role of "data storage node" is connected to the coordination node using the SSHFS client program. The coordination node registers the allocated disk space of the data storage node and writes the necessary files to it. At the same time, the path to the file is stored in the database on the coordination node. If necessary, it is possible to increase the amount of permanent memory by adding the appropriate node to the coordination layer and assigning it the appropriate role.

The Task Scheduler component for a virtual node group performs the following functions:

– registers the task assigned by the user to the formed virtual cluster;

– defines a reference to input data;

– provides a link to dedicated disk space for data storage.

The component is also responsible for data pre-analysis consisting of microservices. Microservices are responsible for different types of data: image/video, text, or time series data, linked through the Rest API. Analysis of data types takes place at this layer on microservices. In case of data inconsistency or errors, the message is returned to the data source. In case of repeated errors and it is impossible to eliminate them, a message is sent to the user using the "Web Service Management" component.

The "Result Processing and Registration" component is actually a data presentation layer. It is responsible for outputting results depending on the task, as well as logging them and writing them to the allocated disk space. Information about new results is passed to the management layer to notify the user. Including information about the current state of the nodes and their load (processor, memory, disk, network).

Let's consider in detail the upper layer of the proposed architecture – the layer of the virtual group of cluster devices or the layer of the virtual cluster directly.

In its essence, this layer of the architecture corresponds to an ordinary cluster of single-board computers, which can be assembled on a local stand and consists of a main (controlling) node, subordinate computing nodes and a switch. The only difference is that this layer can include several unrelated virtual clusters, and the devices participating in cluster construction are heterogeneous and interact in heterogeneous network conditions. Each virtual cluster interacts with the coordination node through the master node. That is, in the developed architecture, the virtual cluster layer is a partial component of the system, unlike alternative solutions with the implementation of a full-fledged cluster on single-board computers.

The second difference of this layer is that a virtual cluster can be built from heterogeneous devices, similar in characteristics and performance in different tasks. The master node can be selected from more productive devices to enable maximum stability and performance of the cluster.

The master node is inextricably linked to the coordination node by means of the Server component. receiving data from it for analysis, as well as giving it its current state. If the master node disconnects from the network, the task pool assigned to it will be transferred to the next node that will take its role.

The interconnections of the components of this layer are shown in Fig. 4.

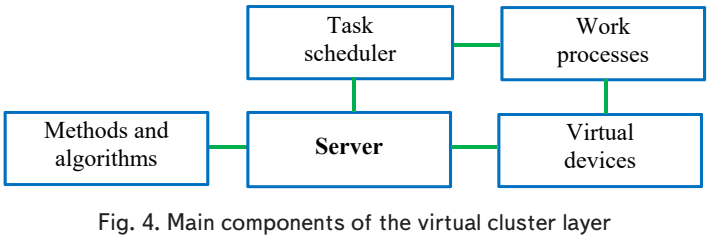

Fig. 4. Main components of the virtual cluster layer

Let's take a closer look at each of the components of this layer.

The "Methods and Algorithms" component is responsible for the applied methods and algorithms in performing a computational task. Depending on the task, the user selects the necessary module, where s/he can set his/her parameters or use pre-trained models. Next, the module is loaded on the main node of the virtual cluster. Thanks to the modular principle, you can write and load your own module, describing the methods and algorithms for solving the given task.

The Task Scheduler component distributes tasks between computing nodes of a virtual cluster. The scheduler also contains information about the connection to the data storage server. All requests from computing nodes to the server go through this component, which makes it possible to monitor free ports. Ports are issued according to the FIFO algorithm (first in, first out).

The Workflows component performs a complete cycle of task calculation. After connecting to the task scheduler, it gets a free port to connect to the server. After that, data is received for processing from the task queue and an indication of which task should be performed on this data. Next, the task is solved, after which the component is connected to the scheduler again. Now it completes the loop by getting a port to connect to the server to transfer the processed data.

So, after considering the proposed architecture in detail, we shall consider the process of building a virtual cluster.

5. 2. Stages of virtual cluster construction in the edge environment of the Internet of Things

The construction of a virtual cluster according to the proposed architecture is divided into several successive stages. The cluster construction and management layer is the first to be involved in the work of the cluster. It is responsible for:

– registration and testing of new nodes with the assignment of a node to a role corresponding to its characteristics and current cluster parameters;

– selection of parameters for virtual clusters based on available devices;

– performance of all levels of virtual clusters;

– user interaction with the formed cluster through the web interface.

After registration and testing of available nodes, the initial parameters are set for the construction of a virtual cluster, which begins with the coordination layer. This layer consists of a coordination node (CN) and data storage nodes. At the same time, the CN is responsible for the interaction between the server and virtual groups of nodes (VGN), as well as for interaction with data storage nodes.

The next step is to appoint a master node, as well as subordinate to it the computing nodes that will form VGN. It should be noted that a coordination node can serve several VGNs.

At the same time, important parameters of computers, such as network speed, processor and RAM load, and the amount of free external memory, can differ significantly. In order to construct a more balanced VGN, it is necessary to select devices with similar parameters.

The main steps of these stages are shown in Fig. 5.

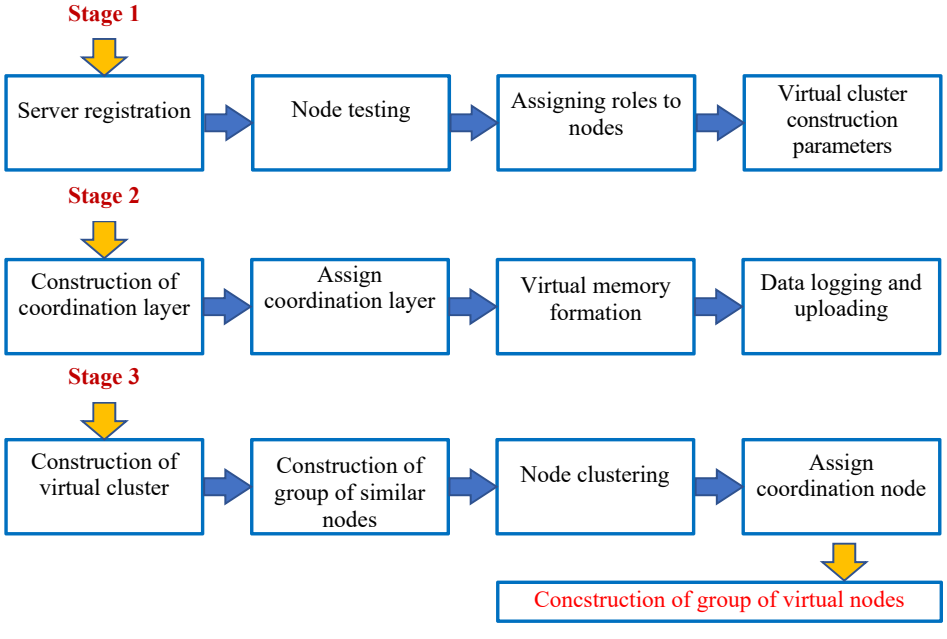

Fig. 5. Main stages in the construction of a virtual cluster

Let's take a closer look at some essential components of this process, such as registration, testing, and node role assignment.

After starting the server and downloading the necessary components, the control node starts accepting external requests for the registration of nodes with the installed client part. All connected nodes undergo registration and send basic technical characteristics. After that, they are tasked with conducting performance tests.

After receiving the list of nodes and the test results, the cluster construction begins, using the basic set parameters for VGN. The user can set the parameters in advance or based on the received data about the registered nodes through the web interface. At this stage, all involved nodes are assigned a role in the VGN, a position in each layer, and the necessary modules are transferred for loading.

After loading all necessary modules, the node sends a signal about readiness to the server. If a node has received the role of "Server", it will wait for nodes to connect until the layer is fully formed according to the received data. Nodes with a different role send a signal about readiness to the server after completing downloads and starting all modules. After that, they wait for data to connect to the desired node.

The node registration process consists of several steps and is available for nodes with client software installed. After the server starts up and appears on the network, the client connects to it. After that, the client transmits data about himself/herself in the form of IP and MAC addresses for unambiguous identification in the future. After registering and entering relevant data into the database of the "Node

Registration" component, the command is sent to the "Testing" component. Next, appropriate modules are assigned for performance testing. The test results are transferred to the "Testing" component, from where the data goes to the registrar and enters them into the database. The registered node is added to the list participating in cluster construction.

Two layers are distinguished in the designed architecture: the cluster coordination layer and the VGN layer. Each of these layers is assigned a node with a specific role.

A coordination node is assigned to the cluster coordination layer. A node with this role is responsible for distributing tasks between VGNs. In addition, it is responsible for organizing data storage virtualization and file distribution between data storage nodes. Including registration and storage of data processing results. A data storage node is a node designated to organize permanent disk space for data storage.

The virtual node group layer has a master node or Master Node (MN) directly connected to the coordination node. The master node manages the state of the cluster and distributes tasks among the computing nodes. This layer also assigns virtual computing nodes that perform tasks and return the results to the MN.

5. 3. Devising and studying effectiveness of the method for building a virtual cluster of the edge environment on the Internet of Things

The virtual cluster is built according to a three-stage approach in accordance with the scheme shown in Fig. 5. A number of algorithms and methods are added to the built architecture, focused on a certain class of tasks. They are focused on increasing the performance of the virtual cluster.

The coordination node when allocating virtual external memory uses a balancing algorithm for the distribution of tasks for data storage nodes. The purpose of the algorithm is to increase the speed of data transfer from data storage nodes to computing nodes.

Also, the virtual cluster is supplemented by the MN replacement algorithm. The master node can fail and lose connectivity to the cluster, so it is important to monitor its health to minimize VGN performance losses. If the computing node cannot access the MN, it sends the event data to the coordinating node. The coordination node checks the availability of MN. When the failure of the current Master node is confirmed, the algorithm to replace it with a node with the closest characteristics to the replaced MN is initiated. The new node is given all the necessary modules to load, as well as a queue of tasks to perform calculations. Data clustering algorithms without a trainer were used to solve the problem of node selection.

When building the VGN computing layer, the heterogeneity of devices and their selection based on approximately the same characteristics and performance are taken into account. For this, an algorithm implementing Amdahl's law was used.

The proposed method involves the use of already operational nodes in the IoT network. Therefore, these devices already perform some tasks on a permanent basis, and it is necessary to take into account their free resources. The heterogeneity of nodes in terms of their characteristics is also taken into account. In this approach, VGN filling is based on the distribution of nodes into groups based on similar characteristics and resources of available nodes. Virtual groups are filled in stages, for each group separately, and a common list of nodes is used.

When devising the method at the step of preliminary processing of input data, it was necessary to choose a clustering method based on the most important characteristics.

To this end, a virtual cluster was built for 5,000 devices, including 12 different models of single-board computers. Device test data is obtained from open sources. In the coordination layer, two roles were considered:

– coordination node;

– data storage node.

First of all, a coordination node was specified, and data storage nodes were already assigned to it.

When selecting devices for the role of computing nodes on the generated data, the qualitative characteristics of the devices were generated based on the following parameters:

– performance in single-threaded and multi-threaded tests;

– free RAM;

– speed of network interaction.

Due to the comparison of different values, normalization was used. The normalization task was reduced to the following formula:

$$X_{norm} = \frac{X_e - X_{dpl}}{\Delta X}, \quad (8)$$

where X_i is the current value; X_{dpl} – displacement value; ΔX is the value of the considered interval; it is the difference between the maximum and minimum values of the normalized parameter.

For comparison, three clustering algorithms were considered: K-means, agglomerative clustering, spectral clustering. Also, for simplification, the main components algorithm was considered. The input data space was considered according to three dimensions: free RAM, Free Disk Space, Ethernet Speed. The space of the main components was reduced to two dimensions.

After the clustering, it was evaluated based on the results of the three metrics described in the previous chapter. The results of the assessment are given in Tables 1, 2. The methods of principal components and agglomerative clustering, which showed similar results and differ in the level of error, worked best. At the same time, the spectral clustering algorithm showed the worst result for the Davies-Bouldin and Calinski&Harabas metrics.

Table 1

Results of metrics in the input space

Algorithm title	Performance Metric		
	Silhouette	Davies-Bouldin	Calinski& Harabasz
K-Means	0.404	1.093	1844.8
Agglomerative Clustering	0.413	1.085	1835.3
Spectral Clustering	0.406	1.920	1404.8

Clusters of points turned out to be sufficiently distant from each other and, accordingly, all three clustering methods built similar clusters. This explains the similar results of the metrics.

Table 2

Results of metrics in the space of principal components

Algorithm title	Performance Metric		
	Silhou-ette	Davies-Bouldin	Calinski& Harabasz
K- Means	0.618	0.556	8148.6
Agglomerative Clustering	0.604	0.574	7998.5
Spectral Clustering	0.605	0.602	7742.5

Since the clustering algorithms worked well, and the difference of the metrics is at the level of error, the choice of the clustering algorithm was made based on the speed of work. The execution time of the clustering algorithms is given in Table 3.

Table 3

Execution time of clustering algorithms

Algorithm title	Execution time (s)
K-Means	0.66
Agglomerative Clustering	1.39
Spectral Clustering	6.13

Our results demonstrate a clear advantage of the k-Means algorithm over the other two.

After all available devices have been identified and assigned the appropriate role, they can be used to build a virtual cluster. Coordination nodes and master nodes are defined from the very beginning, then the coordination layer is filled with storage nodes until the predetermined parameters are reached. Computational nodes fill a virtual group of nodes according to the previously performed clustering.

At the last stage of design, comparative testing of virtual and classic clusters was carried out.

The first virtual cluster testing was performed on a manually built virtual cluster for a small number of computers with limited computing resources.

The virtual cluster had the following structure:

– coordination node;

– the first virtual group of nodes consists of 12 nodes, including the master node;

– the second virtual group of nodes consists of 12 nodes, including the master node.

The second cluster was built in a classic design and consisted of 25 nodes, including the master node.

Performance testing was performed on the classic task of adding matrices of size: 100, 1000, 10000, 100000. Matrices were generated once randomly.

For the built virtual cluster, the coordination node acts as the first balancer of tasks and gives the available tasks to master nodes, which distribute them among free nodes.

In a classic cluster, the master node is responsible for data storage and task distribution. The test results are shown in Fig. 6.

As the test results showed (Fig. 6), the productivity of the IoT edge environment when using the proposed method in comparison with the existing approach improves with the increase in network load. At the same time, with a small network load, performance may decrease slightly.

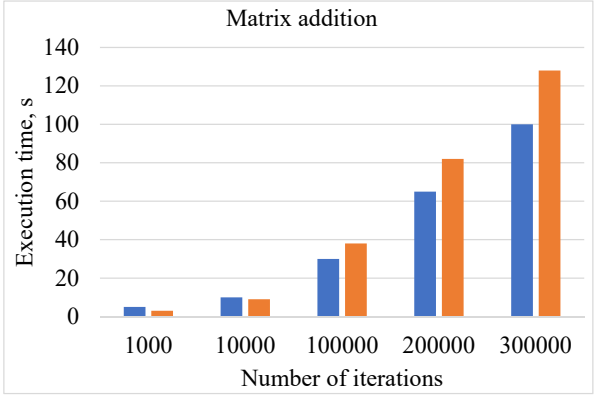

Fig. 6. Results of cluster testing in the matrix addition test: blue color – task execution time by the virtual cluster; orange color – task execution time by the classic cluster

The next test concerned the dependence of the performance of the edge environment on the number of virtual groups of nodes. This testing of the virtual cluster was carried out using the method of empirically filling the layer of virtual groups of nodes with computing nodes. All available options for 25 different devices were searched. This made it possible to lay the foundation for the "meta-learning" method. Testing was carried out in two variants:

– a simple test: the classic problem of matrix addition;

– a difficult test: image recognition using a convolutional neural network with a trained model.

In the course of testing the selection of different configurations of the cluster, an additional condition was formulated. Each master node must have at least two computing nodes. Therefore, the minimum VGN consists of three devices. For 25 devices available for testing, the maximum number of virtual clusters is 8. Testing starts with 1 virtual cluster of three devices, after passing the test, a new free device is added. A limitation was introduced: the process works until the performance increase in the test is less than 5 %. As soon as such a limitation appears, a new virtual cluster is built in addition to the existing one. But the current test will involve all available devices.

The best test results selected among identical VGN configurations are shown in Fig. 7.

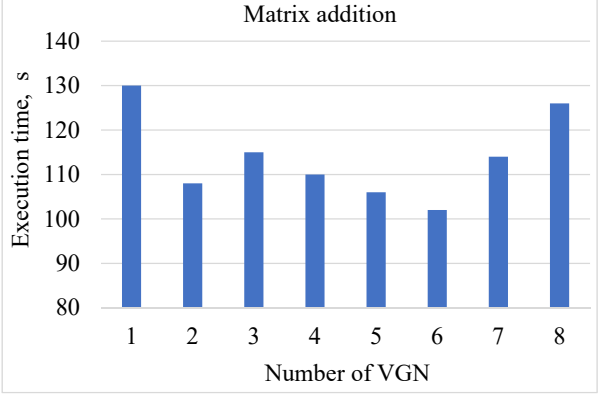

Fig. 7. Choosing the best VGN configuration

The best was a virtual cluster with a configuration consisting of 6 VGNs and 24 devices, each VGN includes 4 devices.

Next, a 1000×500 pixel, 2MB image recognition test was run. A convolutional neural network of the standard library "face_detector" with a previous model was used. The test was conducted using the same virtual cluster construction algorithm as in the previous test.

The averaged results of the dependence of the recognition time on the number of VGNs are shown in Fig. 8.

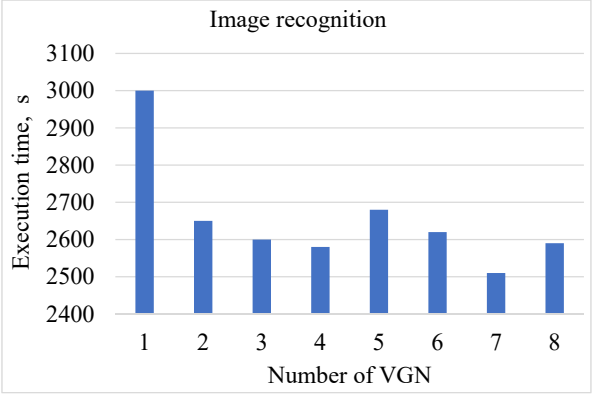

Fig. 8. Image recognition time

Note that the variant with the absence of additional VGNs practically coincides with the results obtained with classical clustering.

6. Discussion of results of devising a method for the virtual clustering of the edge environment on the Internet of Things

A detailed architecture of a virtual cluster of computers with limited resources has been designed. The development took into account the specific features of the IoT edge environment. It is proved that the distributed architecture of a virtual heterogeneous cluster should include the four layers shown in Fig. 1. The justified structure of the cluster construction and management layer, the main components of which and the relationships between them are shown in Fig. 2. There is a well-founded need to introduce an additional coordination layer of the cluster. A separate function of organizing virtual memory for storing data received for processing is specified (Fig. 3). In the designed architecture, the virtual cluster layer is a partial system component. This is a significant difference from existing solutions with the implementation of a full-fledged cluster on single-board computers (Fig. 4).

We have justified a three-stage procedure for building a virtual cluster in the edge environment of IoT (Fig. 5). Essential components of this procedure are the processes of registration, testing, and assigning the role of the node. This procedure made it possible to devise a method for the virtual clustering of the IoT edge environment based on the proposed virtual cluster architecture.

The effectiveness of the proposed method of building a virtual cluster of the IoT edge environment was investigated. Comparative testing of virtual and classic clusters showed the following results:

– at a network load of up to 50 %, a virtual cluster has no advantage over a classic cluster and can lose up to 3 % in overall performance;

– at a network load of 50 % to 80 %, the virtual cluster outperforms the classic cluster by up to 10 % in overall performance;

– at a high network load (from 80 % to full), the virtual cluster significantly outperforms the classic cluster in terms of total performance, by more than 10 %.

In addition, a study of the dependence of the productivity of the virtual cluster of the IoT edge environment on the number of virtual cluster groups was carried out. For a significantly heterogeneous environment, performance changes at full network load can vary by up to 20 % depending on the number of virtual node groups. The results of the study are given in Tables 1–3 and shown in the diagrams in Fig. 6–8.

Our results of studying the method for building a virtual cluster of the IoT edge environment can be explained by increasing the balance of the network load during virtual clustering.

Unlike the methods reported in [9, 13, 20, 21], the proposed method allows taking into account the heterogeneity of the environment. Also, in contrast to the methods given in [15, 18, 19], the resource costs of nodes of the edge environment are taken into account. Compared to the methods described in [16, 17], when using the method of building a virtual cluster of the IoT edge environment, the performance of the network increases. This becomes possible thanks to the increase in load balancing of the physical devices of the edge layer.

Therefore, our solutions allow the utilization of unused resources of IoT edge layer devices in a heterogeneous environment. A significant load imbalance of devices is reduced due to network virtualization.

When applying in practice, as well as in further theoretical studies, one should take into account the limitation of the current study, which is the heterogeneity of physical devices.

As a shortcoming of this study, one should note the lack of analysis of the situation with network congestion of devices of the IoT edge environment. To eliminate the shortcoming, it is necessary to conduct additional research into the nature of queues at nodes of the edge layer of the IoT and to identify their impact on the performance of the virtual cluster.

The following may be the topic of further research.

First, the performance of a virtual cluster depends significantly on the network load. At the same time, in cases where performance does not increase compared to a classic cluster, virtualization of the edge environment is impractical. Therefore, it is necessary to determine the network load threshold at which it makes sense to carry out network virtualization.

Second, the advantages of a virtual cluster depend on the level of network heterogeneity. Therefore, further improvement of the proposed method should be focused on determining the level of network heterogeneity and adding it to the input data of the corresponding algorithm.

7. Conclusions

1. An approach to the construction of an architecture for a virtual cluster of computers with limited resources has been devised. The development took into account the specific features of the edge environment of the Internet of Things. This made it possible to propose a four-layer architecture in-

stead of the standard seven-layer architecture of IoT sensor information processing device networks.

2. We have defined stages in the virtual cluster construction in the edge environment of the Internet of Things. A three-stage procedure for building a virtual cluster was justified. This procedure made it possible to devise a method for the virtual clustering in the Internet of Things edge environment based on the proposed virtual cluster architecture.

3. We have investigated a method for building a virtual cluster of the edge environment of the Internet of Things. The results of the study made it possible to compare the performance of virtual and classic clusters. At a network load of up to 50 %, a virtual cluster has no advantage over a classic cluster and can lose up to 3 % in overall performance. With higher network load, the virtual cluster outperforms the classic cluster in terms of overall performance. At close to full load, the overall performance advantage exceeds 10 %. It is also proven that for a heterogeneous environment, performance changes at full network load significantly depend on the number of virtual node groups. Such fluctuations can reach up to 20 %.

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 and the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available 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. Schulz, A. S. (2023). User Interactions with Internet of Things (IoT) Devices in Shared Domestic Spaces. Proceedings of the 22nd International Conference on Mobile and Ubiquitous Multimedia. https://doi.org/10.1145/3626705.3632615
2. Pardo, C., Wei, R., Ivens, B. S. (2022). Integrating the business networks and internet of things perspectives: A system of systems (SoS) approach for industrial markets. Industrial Marketing Management, 104, 258–275. https://doi.org/10.1016/j.indmarman.2022.04.012
3. Zakharchenko, A., Stepanets, O. (2023). Digital twin value in intelligent building development. Advanced Information Systems, 7 (2), 75–86. https://doi.org/10.20998/2522-9052.2023.2.11
4. Chalapathi, G. S. S., Chamola, V., Vaish, A., Buyya, R. (2021). Industrial Internet of Things (IIoT) Applications of Edge and Fog Computing: A Review and Future Directions. Advances in Information Security, 293–325. https://doi.org/10.1007/978-3-030-57328-7_12
5. Zuev, A., Karaman, D., Olshevskiy, A. (2023). Wireless sensor synchronization method for monitoring short-term events. Advanced Information Systems, 7 (4), 33–40. https://doi.org/10.20998/2522-9052.2023.4.04
6. Krishnan, S., Ilmudeen, A. (2023). Internet of Medical Things in Smart Healthcare. Apple Academic Press. https://doi.org/10.1201/9781003369035
7. Fatlawi, A., Al-Dujaili, M. J. (2023). Integrating the internet of things (IoT) and cloud computing challenges and solutions: A review. AIP Conference Proceedings. https://doi.org/10.1063/5.0181842
8. Qayyum, T., Trabelsi, Z., Waqar Malik, A., Hayawi, K. (2022). Mobility-aware hierarchical fog computing framework for Industrial Internet of Things (IIoT). Journal of Cloud Computing, 11 (1). https://doi.org/10.1186/s13677-022-00345-y
9. Lu, S., Wu, J., Wang, N., Duan, Y., Liu, H., Zhang, J., Fang, J. (2021). Resource provisioning in collaborative fog computing for multiple delay-sensitive users. Software: Practice and Experience, 53 (2), 243–262. https://doi.org/10.1002/spe.3000
10. Petrovska, I., Kuchuk, H. (2023). Adaptive resource allocation method for data processing and security in cloud environment. Advanced Information Systems, 7 (3), 67–73. https://doi.org/10.20998/2522-9052.2023.3.10
11. Kuchuk, G., Nechausov, S., Kharchenko, V. (2015). Two-stage optimization of resource allocation for hybrid cloud data store. 2015 International Conference on Information and Digital Technologies. https://doi.org/10.1109/dt.2015.7222982
12. Li, G., Liu, Y., Wu, J., Lin, D., Zhao, S. (2019). Methods of Resource Scheduling Based on Optimized Fuzzy Clustering in Fog Computing. Sensors, 19(9), 2122. https://doi.org/10.3390/s19092122
13. Jamil, B., Shojafar, M., Ahmed, I., Ullah, A., Munir, K., Ijaz, H. (2019). A job scheduling algorithm for delay and performance optimization in fog computing. Concurrency and Computation: Practice and Experience, 32 (7). https://doi.org/10.1002/cpe.5581
14. Gomathi, B., Saravana Balaji, B., Krishna Kumar, V., Abouhawwash, M., Aljahdali, S., Masud, M., Kuchuk, N. (2022). Multi-Objective Optimization of Energy Aware Virtual Machine Placement in Cloud Data Center. Intelligent Automation & Soft Computing, 33 (3), 1771–1785. https://doi.org/10.32604/iasc.2022.024052
15. Proietti Mattia, G., Beraldi, R. (2023). P2PFaaS: A framework for FaaS peer-to-peer scheduling and load balancing in Fog and Edge computing. SoftwareX, 21, 101290. https://doi.org/10.1016/j.softx.2022.101290
16. Kuchuk, N., Mozhaiev, O., Semenov, S., Haichenko, A., Kuchuk, H., Tiulieniev, S. et al. (2023). Devising a method for balancing the load on a territorially distributed foggy environment. Eastern-European Journal of Enterprise Technologies, 1 (4 (121)), 48–55. https://doi.org/10.15587/1729-4061.2023.274177

17. Kuchuk, N., Ruban, I., Zakovorotnyi, O., Kovalenko, A., Shyshatskyi, A., Sheviakov, I. (2023). Traffic Modeling for the Industrial Internet of NanoThings. 2023 IEEE 4th KhPI Week on Advanced Technology (KhPIWeek). https://doi.org/10.1109/khpiweek61412.2023.10312856
18. Sharma, S., Saini, H. (2019). A novel four-tier architecture for delay aware scheduling and load balancing in fog environment. Sustainable Computing: Informatics and Systems, 24, 100355. https://doi.org/10.1016/j.suscom.2019.100355
19. Khudov, H., Diakonov, O., Kuchuk, N., Maliuha, V., Furmanov, K., Mylashenko, I. et al. (2021). Method for determining coordinates of airborne objects by radars with additional use of ADS-B receivers. Eastern-European Journal of Enterprise Technologies, 4 (9 (112)), 54–64. https://doi.org/10.15587/1729-4061.2021.238407
20. Malik, U. M., Javed, M. A., Frnda, J., Rozhon, J., Khan, W. U. (2022). Efficient Matching-Based Parallel Task Offloading in IoT Networks. Sensors, 22 (18), 6906. https://doi.org/10.3390/s22186906
21. Liu, L., Chen, H., Xu, Z. (2022). SPMOO: A Multi-Objective Offloading Algorithm for Dependent Tasks in IoT Cloud-Edge-End Collaboration. Information, 13 (2), 75. https://doi.org/10.3390/info13020075
22. Ghenai, A., Kabouche, Y., Dahmani, W. (2018). Multi-user dynamic scheduling-based resource management for Internet of Things applications. 2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC). https://doi.org/10.1109/iintec.2018.8695308
23. Kuchuk, G. A., Akimova, Yu. A., Klimenko, L. A. (2000). Method of optimal allocation of relational tables. Engineering Simulation, 17 (5), 681–689.
24. Wei, J.-Y., Wu, J.-J. (2023). Resource Allocation Algorithm in Industrial Internet of Things Based on Edge Computing. Dongbei Daxue Xuebao / Journal of Northeastern University, 44 (8). https://doi.org/10.12068/j.issn.1005-3026.2023.08.002
25. Yaloveha, V., Podorozhniak, A., Kuchuk, H. (2022). Convolutional neural network hyperparameter optimization applied to land cover classification. Radioelectronic and computer systems, 1, 115–128. https://doi.org/10.32620/reks.2022.1.09
26. Zhang, Z. (2021). A computing allocation strategy for Internet of things' resources based on edge computing. International Journal of Distributed Sensor Networks, 17 (12), 155014772110648. https://doi.org/10.1177/15501477211064800
27. Attar, H., Khosravi, M. R., Igorovich, S. S., Georgievan, K. N., Alhihi, M. (2021). E-Health Communication System with Multiservice Data Traffic Evaluation Based on a G/G/1 Analysis Method. Current Signal Transduction Therapy, 16 (2), 115–121. https://doi.org/10.2174/1574362415666200224094706
28. Kammoun, N., Abassi, R., Guemara, S. (2019). Towards a New Clustering Algorithm based on Trust Management and Edge Computing for IoT. 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC). https://doi.org/10.1109/iwcmc.2019.8766492
29. Kovalenko, A., Kuchuk, H. (2022). Methods to Manage Data in Self-healing Systems. Studies in Systems, Decision and Control, 113–171. https://doi.org/10.1007/978-3-030-96546-4_3
30. Yang, J., Bao, L., Liu, W., Yang, R., Wu, C. Q. (2023). On a Meta Learning-Based Scheduler for Deep Learning Clusters. IEEE Transactions on Cloud Computing, 11 (4), 3631–3642. https://doi.org/10.1109/tcc.2023.3308161
31. Pisching, M. A., Pessoa, M. A. O., Junqueira, F., dos Santos Filho, D. J., Miyagi, P. E. (2018). An architecture based on RAMI 4.0 to discover equipment to process operations required by products. Computers & Industrial Engineering, 125, 574–591. https://doi.org/10.1016/j.cie.2017.12.029