Journal of Theoretical and Applied Information Technology
31st December 2020. Vol.98. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RESEARCH OF APPROACHES AND METHODS FOR ORGANIZATION OF COMPUTATIONAL PROCESSES IN THE CLOUD ENVIRONMENT

[1]RAISSA USKENBAYEVA, [2]ZHULDYZ KALPEYEVA, [3]AIZHAN KASSYMOVA

[1]Professor, International Information Technology University, Almaty, Kazakhstan
[2]Assistant-professor, International Information Technology University, Almaty, Kazakhstan
3Assistant-professor, International Information Technology University, Almaty, Kazakhstan

E-mail: [1]ruskenbayeva@iitu.kz
*Corresponding Author E-mail: [2]zh.kalpeyeva@iitu.kz
E-mail: [3]u.aizhan@gmail.com

ABSTRACT

Today, in the context of a constant increase in the number of solved scientific and applied problems and a significant increase in the load on computing systems, cluster systems, grid systems and cloud systems are widely used. Providers of network, information and computing services, relying on large consolidated data centers, began to pay special attention to improving methods of organizing the computing process, which includes planning and allocating suitable resources to meet the resource needs of users, as well as the discipline of servicing tasks with computing resources.
An overview of modern approaches and methods of organizing the computational process in distributed systems is given. Based on the results of the analysis of the current situation in the studied area, the formulation of the research problem. The essence and mathematical models of the organization of the computational process are proposed.
Keywords: *Cloud Computing, Grid, Distributed Computing Systems (DCS), Iaas (Infrastructure As A Service), Task Scheduling*

1. INTRODUCTION

One of the main trends in the development of information technology at the present time is the massive introduction of cloud computing technologies. Thanks to the development of hardware computing and network technologies, it has become possible to combine heterogeneous distributed computing resources, data storages, and corresponding software into a single computing environment, access to which is provided via the Internet.

The main advantage of cloud computing is the ability to dynamically manage resources depending on their demand.

However, with its practical implementation, a number of still unsolved scientific problems arise that impede the full use of all the potential advantages of this approach.

The creation of a cloud infrastructure inevitably faces the need to work in a heterogeneous environment, to organize the access of users with their individual applications to computing resources without reducing the overall performance of the system.

To enable the practical use of a heterogeneous cloud environment, it is necessary to organize a universal system for scheduling and allocating resources for user requests.

The issues of planning and managing computing resources are among the most difficult to organize distributed dynamically scalable systems such as Cloud Computing. It is necessary to take into account the heterogeneity, dynamic composition and scale of the distributed environment. The solution to this problem requires new research and development of planning tools and efficient allocation of resources to fulfill the tasks of users, but already based on the characteristics of current technologies for the creation and operation of modern distributed computing systems.

Modern distributed computing systems are heterogeneous, multiarchitectural, dynamically scalable information processing systems, which complicates the organization of their functioning, scheduling and distribution of tasks between

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

computing resources. The issues of planning and managing computing resources are among the most complex distributed dynamically scalable systems such as Cloud Computing in the organization. It is necessary to take into account the heterogeneity, dynamic composition and scale of the distributed environment. The solution to this problem requires new research and development of planning tools and effective allocation of resources to fulfill the tasks of users, but already based on the characteristics of current technologies for the creation and operation of modern distributed systems.

Taking into account that the listed problems have not yet been finally resolved, we can assume that the topic of this paper is relevant and practically significant.

In this paper, models and methods of organizing computational processes in a distributed cloud environment will be considered. The proposed models take into account the peculiarities of the operation of the components of the "cloud" system and their main characteristics.

2. ANALYSIS OF METHODS AND ALGORITHMS FOR ORGANIZING COMPUTATIONAL PROCESSES IN DISTRIBUTED COMPUTING SYSTEMS

In a distributed environment such as cloud computing, the efficient organization of computing processes is an important task. The process of organizing computing processes in a distributed cloud environment consists of a number of tasks, the main of which are:

In a priori (before the start of computing processes):

- determination of the location of information resources, data and application packages required for the execution of user applications and tasks;
- determination of discipline and method of interaction of tasks with environmental resources;

In a posteriori (during or during a task execution session):

- determination of the discipline of selection of tasks from the input stream of tasks for servicing by the system;
- determination of the order of distribution (appointment) of tasks by performers;
- ensuring the uniformity of the load on load balancing resources;
- determination of the moment of completion of the computational process of the current session.

In some cases, the tasks or the process of assigning tasks to performers and the process of scheduling the execution of tasks by individual performers are combined and called the process of scheduling computational processes.

There are many works in the literature on dynamic resource allocation in cloud computing, addressing the issue from a variety of angles. Therefore, we can not provide a comprehensive treatment of related work, but focus mainly on aspects of determining the discipline of selecting tasks from the input stream of tasks for servicing by the system, determining the order of distribution (assignment) of tasks by executors, taking into account the uniformity of the load on computing resources.

Let us consider further the existing methods of organizing computational processes in distributed systems.

There are a number of fairly simple scheduling algorithms that allow you to select a job to assign to a freed processor from the list of ready jobs, which include list algorithms [1].

List scheduling algorithms order the tasks entering the queue by some of their parameters, for example, by the number of processors required for execution or the time of its execution. And then they look through it, starting from the head, and assign tasks to free computing resources [2].

The most famous list scheduling algorithms are FCFS (First Come First Served), SJF (Shortest Job First) (Shortest Job First), LJF (Longest Job First) (Longest Job First), RJF (Random Job First) (the random problem is served first), etc.

The simplest to implement is the *FCFS* (First Come First Served) discipline, according to which tasks are serviced "in the first place", that is, in the order of their appearance. Those tasks that were blocked in the course of work (got into any of the waiting states, for example, due to I / O operations), after entering the ready state, are put into this ready queue before those tasks that have not yet been executed. In other words, two queues are formed: one queue is formed from new tasks, and the second queue is formed from previously executed, but in the waiting state. This approach makes it possible to implement the servicing strategy “finish computations in the order of their appearance, if possible”. This service discipline does not require external

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

intervention in the course of computations, and it does not redistribute processor time [3]. The advantages of this discipline, first of all, can be attributed to the simplicity of implementation and low consumption of system resources for the formation of the task queue, and there is no need for information about the duration of the tasks. An example of the implementation of this method can be found in [4]. In practice, FCFS is often complemented by modifications to other algorithms.

However, along with ease of use, the FCFS discipline leads to the fact that with an increase in the load of the computing system, the average waiting time for service also grows, and short jobs (requiring little computer time) have to wait as much as labor-intensive jobs. The SJN and SRT disciplines allow avoiding this disadvantage.

Service discipline *SJF* (Shortest Job First) which means: the shortest job will be executed next, requires that for each job the estimate in machine time needs is known. The need to communicate to the scheduler the characteristics of tasks that would describe the needs for computing resources led to the development of appropriate language tools. In particular, JCL (Job Control Language) [5] was one of the best known. Users were forced to indicate the estimated execution time, and so that they did not abuse the opportunity to indicate a deliberately lower execution time (in order to get results earlier than others), they introduced a calculation of real needs. The task manager compared the ordered time and the execution time, and if the specified estimate in this resource was exceeded, it put this task not at the beginning, but at the end of the queue. In some operating systems, in such cases, a system of penalties was used, in which, if the ordered computer time was exceeded, the payment for computing resources was carried out at different rates.

The SJF service discipline assumes that there is only one queue of jobs ready to run. And the tasks that were temporarily blocked during their execution (for example, were waiting for the completion of I / O operations), again fall into the end of the queue ready for execution along with the new ones. This leads to the fact that jobs that take very little time to complete, have to wait for the processor along with long jobs, which is not always good.

To eliminate this shortcoming, the *SRT* discipline (Shortest Remaining Time, the next task takes the least time to complete) was proposed.

All these service disciplines can be used for batch processing modes, when the user is not forced to wait for the system's response, but simply surrenders his task and receives his calculation results in a few hours. For interactive computations, it is desirable, first of all, to provide an acceptable system response time and equality in service [3].

Quite popular is the algorithm where the processors form a virtual ring with a marker and the next ready-to-run job is assigned to the processor that has the marker, and then the marker is transferred to the next processor (Round Robin, RR, circular, carousel). The carousel scheduling discipline is best suited when all tasks have the same rights to use CPU resources. However, as we know, equality in life is much less than inequality. Some tasks always need to be solved first, while others can wait. This can be done due to the fact that we (or the task manager) assign one priority to one task, and another task to another. Tasks in the queue will be ranked according to their priorities.

At present, algorithms such as process and work migration, joint scheduling (Gang-scheduling) and the Backfilling algorithm, or filling [6], are actively used to plan computations in scalable systems.

The Backfilling algorithm requires information about the duration of the tasks. The goal of the algorithm is to fill the resulting windows as densely as possible. For this, among the available windows, the widest window is selected, that is, with the maximum number of processors, and the next jobs that fall into the queue will be assigned to the processors of this window. If the new job does not fit in any of the available windows, then it is placed at the end of the queue. Thus, the works are propagated in the opposite direction relative to the timeline. The backfilling algorithm is quite often used in queue systems that provide "fair" access for user tasks to the resources of multiprocessor systems.

Advantages of the backfill algorithm [7]:

- allows you to create schedules for heterogeneous DCS;
- avoids hanging low-priority jobs in queues, guaranteeing their launch;
- can be quite time-dense schedules;
- has acceptable performance characteristics.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Systems that implement the backfilling algorithm include, for example, the Maui scheduler [8].

The Gang-scheduling algorithm allocates the resources of the multiprocessor system among the work groups. The works are grouped into priority groups. Jobs in one group share multiple processors in the same way as in the case of the Backfill algorithm, but work interruption is allowed if a group of jobs with a high priority arrives at the multiprocessor system [9]. Gang-scheduling is included in the IBM LoadLeveler queue management system [10].

The scheduling problem belongs to the class of NP-complete problems with exponential growth of the complexity of the solution with respect to the dimension [11]. Existing exact methods for solving in the worst case, during the search, enumerate all possible options for distributing work by performer, which requires large computational costs for high-dimensional problems. Therefore, finding the optimal solution to the distribution problem in a reasonable time becomes difficult to achieve [11].

The solution of this class of problems requires large computational and, accordingly, time resources to find the optimal solution to the problem, as a result of which the gain from using the found solution does not cover the huge costs of obtaining it. In practice, to solve NP-hard problems, heuristic methods are often used that do not guarantee finding the optimal solution, but allow one to quickly obtain solutions of acceptable quality.

Heuristic optimization methods are based on the use of various reasonable, in most cases based on life, natural analogies, considerations, habits, and rules aimed at achieving a compromise between striving for the best result of actions and reducing the time-consuming direct enumeration of options for these actions. Despite the insufficient theoretical validity, these methods make it possible to obtain acceptable solutions with a relatively small investment of time and other resources [11].

The advantages of heuristic methods can also be attributed to the convenience of their implementation on a computer, even when solving high-dimensional problems.

The disadvantages of these methods lie in the complexity of assessing their actual effectiveness, i.e. the proximity of the obtained solutions to the optimal ones. In addition, for each heuristic approach, there are problems for which the application of this approach is either impossible or leads to frankly bad results. This requires a thorough experimental study of heuristic methods with the aim of identifying classes of problems in the solution of which these methods are most effective [11].

The most effective and popular heuristic methods include the so-called metaheuristics - generalized strategies for finding the optimum in the solution space. Examples include:

- algorithms for simulating annealing (Simulated Anneal, SA) [12],
- genetic and evolutionary algorithms (Genetic Algorithms, GA),
- ant algorithms (Ant colony optimization, ACO) [13].

Algorithm for simulating annealing. Annealing method (synonyms: firing method, simulated annealing method, model hardening method, simulated annealing) is an optimization technique that uses an ordered random search based on analogy with the formation of a crystalline structure in a substance with a minimum energy upon cooling [14].

In practice, the simulated annealing method shows good results, but setting it up for a specific task can be time-consuming. The effectiveness of the method significantly depends on the chosen scheme, in particular, on the initial temperature and the cooling function.

The greedy algorithm consists of making locally optimal decisions at each stage, assuming that the final solution is also optimal. The procedure starts from the vertex with the smallest degree. All neighboring vertices are marked, and then the vertices adjacent to the neighbors. The first n / p labeled vertices are assigned to one subdomain, and the procedure is applied to the rest of the graph until all vertices are labeled.

Genetic algorithm. A genetic algorithm is a technique for solving certain problems by imitating the processes observed during the evolution of natural nature. For the first time, the paradigm of the genetic algorithm was proposed in [15], but it took another ten years for it to be noticed in wide scientific and research circles. In its purest form, a genetic algorithm is a general technique for solving problems, for the implementation of which a relatively small amount of information about the subject area is required. As a consequence, this approach can be used for a fairly large number of poorly structured problems, where specialized methods do not show successful results.

Genetic algorithms are based on the use of mechanisms of natural evolution. Evolution,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

according to Darwin, is carried out as a result of the interaction of three main factors: variability, heredity, natural selection. Variability serves as the basis for the formation of new signs and features in the structure and functions of the body.

Heredity reinforces these characteristics. Natural selection eliminates organisms that are poorly adapted to the conditions of existence.

Genetic and evolutionary computing gained widespread acceptance after the publication of Holland's book [15]. The concept of constructing genetic algorithms proposed in this work turned out to be extremely effective for solving many problems that cannot be solved by traditional methods.

In its work, a genetic algorithm tries to develop a population of bitstrings or “chromosomes”, where each chromosome encodes a solution to a problem in a certain specific formulation. This evolution is realized through the use of several operators imitating the phenomena of living nature (reproduction, modification, etc.).

As noted in [16], the decisive factor for assessing the practical suitability and effectiveness of evolutionary genetic algorithms are speed (the time required to perform a user-specified number of iterations) and search stability (the ability to constantly increase quality from generation to generation). An attempt made in [98] to comparatively analyze competing heuristic algorithms from the point of view of their efficiency revealed the following advantages and disadvantages of evolutionary computations [17].

The Benefits of Evolutionary Computing:

- wide range of applications;
- the possibility of problem-oriented coding of solutions, selection of the initial population;
- suitability for searching solutions of large-scale problems in a complex space;
- no restrictions on the type of objective function;
- the ability to easily parallelize computations (for example, you can split a generation into several groups and work with each of them independently, exchanging several chromosomes from time to time);
- clarity of the scheme and basic principles of evolutionary computing;
- the ability to combine evolutionary computations with non-evolutionary algorithms.

Disadvantages of Evolutionary Computing:

- the heuristic nature does not guarantee the optimality of the solution obtained (however, it is often important to obtain one or several suboptimal alternative solutions in a given time, especially since the initial data in the problem may dynamically change, be inaccurate or incomplete);
- evolution can "wedge" on an unproductive branch (as in real life). The converse is also true: two unpromising parents, who will be excluded from evolution when the GA works, are capable of producing a highly efficient offspring. This becomes especially noticeable when solving high-dimensional problems with complex internal connections;
- relatively high computational complexity, which is overcome due to parallelization at the level of organization of evolutionary computations and their direct implementation in a computing system;
- relatively low efficiency at the final phases of evolution modeling (search operators in evolutionary algorithms are not focused on quickly reaching the local optimum);
- unresolved issues of self-adaptation.

In [18], the possibility of using genetic algorithms for computational scheduling problems was considered, and problems arising when using genetic algorithms to solve such problems were described.

There are also a large number of examples of using GA to solve the scheduling problem [19-21]. In [21], a controlled genetic algorithm (CAA) is proposed, which makes it possible to adjust the parameters of the algorithm at all stages of solving the problem.

The use of genetic algorithms for finding the optimal schedule of tasks in GRID is considered in [22]. In [.23], the use of GA for solving the problem of scheduling training sessions is considered.

The range of problems solved using genetic algorithms is expanding every year. As a consequence, the scope of GA is gradually shifting from theoretical problems, such as the Traveling Salesman Problem, to real-world applications. This shift is an important confirmation of the effectiveness of GA work in solving complex practical problems.

Over the past decade, a large number of genetic operators and functions have been developed, the combination of which gives rise to many genetic methods [24]. Practice has shown

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

that for the successful application of the genetic approach, it is necessary to modify the basic genetic algorithm and adjust its parameters to solve specific problems.

Ant algorithm. Ant algorithms are based on imitating the self-organization of social insects through the use of dynamic mechanisms by which the system achieves a global goal as a result of local low-level interaction of elements. An ant colony can be viewed as a multi-agent system in which each agent (ant) functions autonomously according to simple rules. The basis of the "social" behavior of ants is self-organization. A fundamental feature of such interaction is the use of only local information by the system elements, and any centralized control is excluded. Self-organization is the result of the interaction of the following four components [13]:

- randomness;
- multiple;
- positive feedback;
- negative feedback.

The interaction is determined through a special chemical substance, pheromone, deposited by ants along the path traveled. The higher the concentration of pheromone on the trail, the more ants will move along it. Over time, the pheromone evaporates, which allows ants to adapt their behavior to changes in the external environment [13,25].

In [25], the solution of the classical NP-hard traveling salesman problem based on ant algorithms is considered. Computer experiments show that ant algorithms find good traveling salesman routes much faster than exact combinatorial optimization methods. The efficiency of ant algorithms increases with an increase in the dimension of the optimization problem.

The ant algorithm can be used to solve various combinatorial problems: Quadratic Assignment Problem, Vehicle Routing Problem, Job-Shop Schedule Planing, Graph Coloring Problem and others. Ant algorithms find solutions to discrete optimization problems no worse than other general metaheuristic technologies and some problem-oriented methods [13].

Most of the methods considered assumed that the resources are homogeneous. In particular, in most cases it is assumed that they all have the same performance and the same cost. In this case, the task of selecting resources is greatly simplified, since resources are interchangeable. At the same time, in a real environment, resources are heterogeneous and differ from each other in architecture, performance, memory, price, bandwidth, etc. Selecting the right resource for a custom job becomes a multi-parameter task, with the individual parameters being interdependent. Accordingly, the task of organizing the computing process becomes more complicated, taking into account both the interests of users and consumers.

When using these approaches, resources are used inefficiently in cases where the computing nodes of the cloud data center differ in capacity (resource heterogeneity), user tasks are heterogeneous and have very different resource requirements;

And also the existing "system-centric" (system-centric) approaches to planning [26-27] do not take into account the requirements of consumers for the quality of service. Scheduling methods like these place jobs according to system parameters, such as resource utilization or system throughput. Schedulers focus on either minimizing response time, the sum of the wait time and the actual job execution time, or maximizing the overall resource utilization. In this situation, the goal of planning is to improve the situation of the system as a whole, but not the requirements imposed by individual consumers [27].

Traditional approaches to resource management focus on system-wide criteria such as infrastructure utilization or bandwidth. At the same time, the needs of cloud infrastructures require the development of such models that would more reflect the interests of users. In other words, it is required to create such job distribution algorithms that would provide maximum utility for an individual consumer. However, due to the fundamentally selfish behavior of the participants, it is impossible to force them to objectively take into account system-wide criteria otherwise than by organizing incentive, stimulating mechanisms. A promising approach in this direction includes the use of economic mechanisms to organize resource management. Market value considerations are an important part of modern computing resource allocation research [27].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. THE COMPLEXITY OF THE ORGANIZATION OF THE COMPUTING PROCESS IN CLOUD SYSTEMS

It should be noted that the use of known algorithms for organizing the computational process in traditional distributed environments is not possible in cloud systems due to a number of factors [28]:

- a random change in the number of users, which results in unpredictable dynamics of demand / supply / availability of resources. This feature requires dynamic resource scalability;

- a change in the number of users and the structure of demand makes it necessary to make adjustments to the calculation plans;

- multifactoriality, i.e. the presence of many different criteria, policies, preferences and restrictions on computing work leads to the need for balancing between them;

- a variety of requirements and preferences of users of a computing system requires an individual approach to service consumers;

- due to the inaccuracy of knowledge about the characteristics of physical nodes, virtual machines, tasks, the load is unbalanced during the execution of tasks. The resource system must be balanced, that is, resources must correspond to the general planning strategy in terms of quantity, quality of service, and productivity.

All these factors determine the specifics of the formulation of the problem of organizing computational processes in distributed cloud environments (DCE) with a dynamic structure.

4. STATEMENT OF THE RESEARCH PROBLEM

The scheduling system in the cloud infrastructure must be ready to accept the flow of user requests for the allocation of computing resources in the form of virtual machines to process their tasks, generate schedules and set tasks for execution in accordance with them.

The computing environment consists of a dispatcher (scheduler) D and computational nodes $P_{1,i}$, $P_{2,i}$, …, $P_{m,.i}$ The task flow Zd, sent by users is placed in the queue Q of dispatcher D.

Tasks submitted by users through the cloud interface are placed in a global queue maintained by a central cloud scheduler. This scheduler, implementing the scheduling algorithm embedded in it, makes a decision on assigning tasks to computational clusters, computational nodes, to their virtual machine instances [28].

Each physical compute node does not use an on-premises task scheduler, and its compute resources are completely dedicated to the cloud dispatcher. Thus, the planning system is based on a hierarchical principle.

Each compute node is characterized by the number of processors and cores, the size of memory, the size of free storage, and the cost of computation.

The task is described by the parameters of the waiting time and execution time, arrival time and resource requirements. The task also has a priority.

A feature of cloud tasks is the ability to migrate tasks between nodes and a data processing center (DPC). Data transfer between nodes in the clouds is based on the principle of a separate network data storage and dedicated information exchange services [28].

5. GENERAL METHOD FOR SOLVING THE RESEARCH PROBLEM

It follows from the review that the principle of organizing calculations in the DCS of the Cloud computing type is built on economic principles. The goal of organizing the computational process will be to minimize the total cost of performing user tasks, while the requirement for maximum load of physical nodes should be taken into account, or in the literature there is a definition of maximum utilization of resources. Filling servers with virtual machines as densely as possible leads to a reduction in energy and maintenance costs by shutting down idle servers, or allows you to serve more customers with the same server park.

We will assume that there are two types of resource requests from users: static and dynamic. A static request assumes the provision of services according to the "subscription price" method, in other words, when the user predetermines his resource requests. A pool of input requests is formed, as in the case of batch processing of requests, and the supplier has time to preliminary plan its resources, taking into account all time constraints, cost constraints, etc.

A little more complicated is the situation with a dynamic request for the allocation of the required amount of resources, when the system needs to be triggered "on the fly" to allocate or suspend the work of certain resources. At the same time, take into account the current state of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the system and make adjustments to the current schedule of the computing environment, in this case it is important to take into account such features as "live migration" of virtual machines from one physical host to another, dynamic load balancing of computing resources.

In this paper, a two-fold resource allocation is proposed: first, a schedule is constructed for static queries using well-proven metaheuristic algorithms, in particular a genetic algorithm. A certain set of resources is selected that corresponds to the characteristics of user tasks, among this set of resources, tasks are distributed according to the corresponding criteria, in this case, the criterion will mean the "importance" of the task. When specifying a resource request or a job passport, the user indicates the sign of "importance" or "urgency" of the job, based on this parameter, the total cost of the job or resource lease will be calculated. And also on the basis of this criterion, the selection of certain physical nodes will be made, since they are heterogeneous, i.e. heterogeneous and differ in the degree of performance, speed, amount of RAM, etc.

Processing a dynamic request involves the second stage of resource allocation, we called this stage "improving the existing schedule". It is necessary to take into account the fact that the distribution of user tasks is done, the computing environment is working, virtual machines are distributed among physical nodes. It is necessary to monitor the resources involved, identify the percentage of system utilization and, if possible, use all the resources of the physical nodes to the maximum.

For dynamic requests, we propose to maintain an additional queue of input requests, select several from this pool of requests and place them in an intermediate buffer. Having information about the current state of the computational nodes, we select the most appropriate task for the resource from the buffer and place it on it. In this case, list algorithms such as FCFS, SJF, LJF, Backfill, etc. can be used to retrieve a task from the buffer. Or, the genetic algorithm can be used again.

For the pricing policies of the provided cloud services, it is proposed to introduce a bonus pricing system when the consumer is credited with additional bonuses or conventional units to pay for consumed resources.

The method described in the work is intended to increase the rationality of using the hardware resources of the DCS (resource utilization) and to reduce the total cost of performing the flow of tasks, provided that the processing nodes are heterogeneous.

6. FORMALIZATION OF THE STUDIED SUBJECT AREA

The development of methods and models for organizing computing processes in cloud environments based on market mechanisms requires the elaboration of issues such as delineating the roles of system users, assigning a bonus calculation system for consumed computing resources according to user roles, billing services per unit of time of use, and the procedure for collecting payments for services (prepayment or subscription price for services), regulation of disputable situations in case of failure of one of the parties to fulfill its obligations, accounting for technical failures in the system in case of failure of a certain subset of computing resources or in case of refusal to suspend the services provided by the user. In practice, these aspects are pre-written in the service layer agreement (SLA).

6.1. Features of modern distributed cloud systems

Before starting the analysis of the features of distributed cloud systems, it should be noted that the following terms exist simultaneously with this term: complex computing systems, corporate computing systems, distributed computing systems, distributed information systems, distributed computing network, etc. We will consider them identical, since they all denote the same class of distributed computing systems, but have different properties.

Different names emphasize different aspects of the application or definition of the properties of distributed cloud systems. For example, if the main feature (characteristic) of the system is the ability to provide ubiquitous and convenient network access on demand to a common pool of computing resources, the system can be called a cloud system.

If we take the provision of user information (i.e. logistics with information) as the purpose of the system, the system can be called an information system. In the case of considering the main feature (characteristic) of the system, the computing processes performed by the system for the purpose of data processing, then the system can be called computational.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Thus, the system is cloud or computational depends on the look. Moreover, in modern component technologies (and architectures), the components from which the system itself is constructed merge data, computational procedures (their programs), access and interaction protocols.

6.2 Scenario of using computing resources of DCS

In the cloud computing environment there are two parties or actors: One of the parties is represented by "cloud" providers who provide cloud services and computing resources located in data centers to users on a rental basis or in another form, depending on the type of "cloud". The other side is represented by users who rent computing resources from vendors to carry out their applications and tasks.

Both parties, consumers and providers, have different goals. The goal of cloud providers is to maximize profits, maximize resource utilization, minimize resource idleness, and balance the load on physical servers.

Users pursue other goals - minimizing the cost and overhead of performing their tasks in order to meet the resource performance requirements.

The required level of service provision is pre-negotiated in a service level agreement (SLA) between the actors.

7. MATHEMATICAL MODEL OF THE ORGANIZATION OF COMPUTING PROCESS

"Job" or "operation" are the basic concepts of a distributed computing system.

A job is an elementary task to be performed in some "computing system". In the general case, a computing system can be considered as a set of interconnected "executors". It is customary to call an executor a server, a computing node capable of performing all the necessary actions to complete a task.

Certain relationships may exist between tasks, i.e. some tasks require other prior tasks to be completed. Having this kind of relationship between jobs means defining an ordering relationship on a set of jobs. Otherwise, the tasks are independent and can be executed in any order.

Accordingly, tasks are executed in parallel if they are independent, or sequentially if there are dependencies. Dependencies determine the order in which work is performed and the flow of data between tasks.

A computing system can consist of both identical and different components. In the first case, the executive system is usually called homogeneous, in the other - heterogeneous. In a homogeneous system, the efficiency of performing the same task on different performers is the same, while in a heterogeneous system it is different. For example, if a multiprocessor system consists of processors of the same model, then such a system is homogeneous. Since the processors have the same clock frequency, cache memory size and other technical characteristics, as a result of which the speed of execution of an individual program is the same on any processor of the computing system and does not depend on the processor on which it is executed. Accordingly, an example of a heterogeneous system is a computing system, which includes processors of different performance. This leads to additional difficulties in drawing up an execution plan for a complex of programs, since it is necessary to take into account the speed of their execution on various processors.

By the nature of the receipt of tasks in the computing system, static and dynamic distribution tasks are distinguished. In static tasks, the executive system simultaneously receives a certain set of tasks, for which a schedule is drawn up and executed. At the same time, during the implementation of the previously drawn up plan, new tasks do not enter the system, i.e. the schedule is formed for a certain and known number of tasks in advance. Thus, in static problems, all input parameters of the problem are assumed to be given before starting to solve it.

In dynamic tasks, tasks are performed continuously. Tasks enter the computer system at unknown moments in time, which can only be predicted in a statistical sense. The schedule is built in parts as jobs arrive in the service system. Thus, schedules for dynamic tasks are built in real time, i.e. the input parameters of the distribution problem are unknown until the start of its solution.

We have chosen a dynamic problem and methods for its solution for the study. This choice is due to the widespread occurrence of this class of problems in distributed computing systems.

7.1 Mathematical model of a user task

We define the mathematical model of the user task as follows [30-31]:

Let the computing environment consist of i heterogeneous, parallel working performers:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$N = (N_1, N_2, ...N_i) \quad (1)$$

A set of independent tasks (jobs) j are received at the input of the computing environment:

$$Zd = (Zd_1, Zd_2 ... Zd_j) \quad (2)$$

which must be distributed among the performers.

Since the construction of the hardware part of cloud infrastructures is based on server virtualization technology, m virtual machines can act as executors:

$$VM = (VM_1, VM_2, ...VM_m) \quad (3)$$

The resource requirements r_i for each task Zd_j are known.

As resource requirements r_i for tasks Zd_j, we will consider processor time or CPU time - the time spent by the processor to process the task and random access memory (RAM), assuming other types of computing system resources such as: disk memory, data storage, bandwidth ability of communication environment, etc. sufficient.

The essence of the organization of the computational process lies in the fact that each task $Zd = (Zd_1, Zd_2 ... Zd_j)$ is assigned a performer $VM = (VM_1, VM_2, ...VM_m)$. Thus, the distribution will result in any set:

$$D^{Zd} = \{Zd_1, ..., Zd_m\} \quad (4)$$

a subset of jobs $Zd_n = \{zd_k, zd_k \in Zd, k \in [1, j]\}$ that matches the following property:

$$\forall n, k \in [1, m] \rightarrow \bigcup_{j=1}^{m} Zd_n = Zd; Zd_n \bigcup_{j \neq k} Zd_k = \varnothing \quad (5)$$

Condition (1) is interpreted that all tasks must be distributed (but not all performers in the used computing system can be busy).

Also, additional restrictions can be defined for the tasks performed, described in the work:

- the moment of receipt of the task for service t_i - this parameter determines the moment of receipt of the task Zd_j in the computing environment;

- deadline for completion of service d_i - this value is the moment in time by which the task is completed and the computing resource is released;

- rate of a job or rent of a virtual machine c_i - the unit cost of a job being in a computing environment.

7.2 Mathematical model of the computing environment

The computing environment (CE) is designed to process the flows of jobs arriving at it. A computing environment built on the basis of distributed cloud computing technology is a set of computing resources in the form of physical servers and virtual machines on which tasks are performed, a control center (scheduler), as well as a telecommunications transmission medium through which communication between the control center and computing knots.

Let's describe the structure and architecture of the given aircraft through a tuple:

$$DC = (N, R, VM, St, Zd), \quad (6)$$

where, $N = (N_1, N_2, ...N_i)$ is a set of physical servers or computational nodes, which can be either of the same type or heterogeneous.

$R = (R_1, R_2, ...R_n)$ - a set of communication (telecommunications environment) between physical servers.

$VM = (VM_1, VM_2, ...VM_m)$ - many virtual machines (VMs) that need to be distributed across physical servers.

$St = (St_1, St_2, ...St_i)$ - a set of data storage systems (DSS) that store VM images, data, application packages.

$Zd = (Zd_1, Zd_2 ... Zd_j)$ - the set of tasks for which the given system is intended.

Each computational node can simultaneously run several instances of virtual machines described by a tuple:

$$N_i = \{VM_{i,1}, VM_{i,2}, ...VM_{i,m}\} \quad (7)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

where, m is the number of virtual machines on the i-th computing node, $i=1,n$ (n is the total number of computing nodes).

Each virtual machine contains a set of applications and programs that provide the execution of custom tasks:

$$VM_j = \{\Pr_{j,1}, \Pr_{j,2}, ... \Pr_{j,n}\} \quad (8)$$

where, n is the number of applications of the j-th virtual machine, $j=1,k$ (k is the total number of virtual machines).

The types of possible instances of virtual machines are specified by the set:

$$St_h = \{VMimg_{h,1}, VMimg_{h,2}, ...VMimg_{h,p}\} \quad (9)$$

where, p is the number of types of images (images) of virtual machines containing the required operating system with preinstalled software, $h=1,z$ (z is the number of data stores).

Each type of participants or classes in the cycle of the computational process of servicing requests is characterized by certain properties, attributes, characteristics, which can be formally represented as features.

Each physical server N_i is specified by a model from a tuple of characteristics:

$$\chi(N_i) = (\chi_1(N_i), \chi_2(N_i), ..., \chi_k(N_i), ..., \chi_h(N_i), ..., \chi_m(N_i)) \quad (10)$$

where,

$\chi_1(N_i)$ - an indication of the performance or speed of the processor N_i;

$\chi_2(N_i)$ - an indication of the capacity \ volume of the server's N_i RAM, the additional designation of this sign is set as $Mr(N_i)$;

$\chi_3(N_i)$ - an indication of the server N_i cache memory size;

- an indication of the server N_i hard disk volume;

$\chi_4(N_i)$ - an indication of server N_i availability;

$\chi_5(N_i)$ - an indication of server reliability;

$\chi_6(N_i)$ - an indication of server security;

A virtual machine VM_i is characterized by the following features:

$$\chi(VM_i) = (\chi_1(VM_i), \chi_2(VM_i), ..., \chi_k(VM_i), ..., \chi_h(VM_i), ..., \chi_m(VM_i)) \quad (11)$$

where,

$\chi_1(VM_i)$ - the number of VM VM_i cores;

$\chi_2(VM_i)$ - an indication of the capacity \ volume of the VM VM_i RAM;

$\chi_2(VM_i)$ - an indication of the volume of the hard disk of the VM VM_i;

$\chi_3(VM_i)$ - an indication of a pre-installed server operating system;

$\chi_4(VM_i)$ - an indication of server reliability;

$\chi_5(VM_i)$ - an indication of VM VM_i security;

$\chi_6(VM_i)$ - an indication of the cost per unit of time of VM VM_i operation;

The task Zd_i is characterized by the following features: - the sign of the capacity \ volume of the server's RAM, the additional designation of this sign is set as:

$$Zd_i = (\chi_1(Zd_i), \chi_2(Zd_i), ..., \chi_k(Zd_i), ..., \chi_h(Zd_i), ..., \chi_m(Zd_i)) \quad (12)$$

where,

$\chi_1(Zd_i)$ - an indication of the time of receipt of the task Zd_i for service;

$\chi_2(Zd_i)$ - an indication of the urgency of the task Zd_i, the additional designation of the sign of the urgency of the task Zd_i is set as: $Sh(Zd_i)$;

$\chi_3(Zd_i)$ - an indication of the permissible duration of execution;

$\chi_4(Zd_i)$ - an indication of the acceptable cost of execution;

$\chi_5(Zd_i)$ - an indication the required level of continuity;

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$\chi_6(Zd_i)$ - an indication required level of security;

$\chi_7(Zd_i)$ - an indication of the required amount of memory for execution, the additional designation of the sign of the required amount of memory for the task is denoted as: $Mr(Zd_i)$;

The total amount of memory required to complete all tasks:

$$Mr(Zd) = \sum_{i=1}^{m} Mr(Zd_i) \quad (13)$$

When solving tasks in a computing environment, a parameter that is significant for the user is its "importance" or "urgency." Such characteristic of the task as "urgency" for the user means the latest "acceptable time" for receiving the results of the task.

The variant of the organization of computing processes in a distributed cloud environment proposed in this work is based on the market-value method, which allows users to make an individual assessment of the informational importance of a task, expressing it in conventional units that determine the availability of computing resources.

The value of a characteristic or an indication of urgency of task Zd_i can be as follows:

$$\chi_1(Zd_i) = (\chi_{11}(Zd_i), \chi_{12}(Zd_i), ..., \chi_{1j}(Zd_i)), \; j = \overline{1,n}, \quad (14)$$

where,

$\chi_{11}(Zd_i)$ = 1 - an indication of the urgency of execution - not urgent;

$\chi_{12}(Zd_i)$ = 2 - an indication of the urgency of implementation Zd_i - not very urgent;

$\chi_{13}(Zd_i)$ = 3 - an indication of urgency Zd_i - a little urgent;

$\chi_{14}(Zd_i)$ = 4 - an indication of the urgency of execution Zd_i - urgent;

$\chi_{15}(Zd_i)$ = 5 - an indication of the urgency of execution Zd_i - extra urgent.

It is assumed that among all physical servers, there will always be one or more physical servers that satisfy the requirements of any arbitrary job.

Thus, we have determined the composition of the computing environment of the distributed cloud environment, described the interaction mechanisms of the main components of the cloud system. The organization of the computational process for servicing the heterogeneous requests of users of such a system is a non-trivial task. It is also worth considering the limited computing resources.

7.3 Rules for setting tasks for processing with the market-value method

Determination of the set of performers, tasks (operations) and the rules for assigning tasks to the appropriate performers is called the organization of the computing process.

To organize the computational process, it is necessary to find suitable resources, select the required number of them, and establish the order in which tasks from the input queue will be executed on these resources. If the resources requested by jobs exceed the available resources, this leads to conflicts that must be resolved by the scheduling algorithm.

Based on the fact that the provision of cloud services is regulated by market mechanisms, i.e. payment for services occurs upon use according to the established tariffs, in this work we will rely on the market-value method (MVM).

A model using the market-value method for resource allocation is understood as a model of competitive access to computing resources by user tasks based on a notional value calculus.

The actors (parties) of such market relations are resource providers and service consumers. This approach makes it possible to organize taking into account the balance of interests of consumers of computing resources and their suppliers in the same way as it happens in the context of economic relations.

Step 1. Prioritizing the “importance” of the task.

At the stage of scheduling tasks and placing them in the queue for execution, the procedure for calculating the priority of a task based on the "importance" parameter, the urgency specified by the user, is applied. The user is provided with a mechanism that allows them to raise or lower the priorities of their jobs, thereby affecting the processing time of a job in the queue.

Step 2. Establishing the "role" priority of the task.

In addition to the "importance" parameter set by the user, the task priority is influenced by the "role" priority of the user of the computing environment.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Users of the computing environment are divided into role categories:

- administrators of the computing environment manage the functioning of the computing environment;

- privileged users who get access to computing resources of the ROS on a paid basis;

- authorized users who get free access to the computing resources of the environment. This category of users includes members of the organization that owns the cloud infrastructure. For the case of a private university cloud, these can be researchers, teachers, students.

- guest access. This group includes users who have not been authorized and authenticated in the system and have only limited guest access to the computing environment.

At the stage of scheduling jobs and placing them in a queue for execution, the procedure for calculating the priority of a job based on the user's "role" priority is applied. By allocating a larger budget to a job, the user increases the priority value. At the same time, it is not true that the task with a larger budget is a higher priority in planning. If the priorities of the jobs are equal, the job with the longer waiting time in the service queue is the first to enter the execution queue.

A high-budget "urgent" task is first sent to the queue for distribution, then tasks of authorized users with a dedicated budget are selected for processing.

To ensure the distribution of all jobs with each new scheduling cycle, the priority of the job is increased by one value. The dynamic priority mechanism is flexible enough to arrange the jobs in the queue. The method eliminates the possibility of an endless waiting for some task to start for execution (the longer the task is in the queue, the higher the priority becomes).

In order to prevent the monopolization of computing resources, authorized users who receive free access to computing resources are allocated a limited budget (bonuses), within which they can rent computing resources. The introduction of such a bonus calculation system allows the user to subjectively order the priorities of his own tasks and acts as a universal metric for the mutual ranking of incoming tasks in the system.

The user's budget (bonuses) is determined for the reporting period, measured in arbitrary units and is intended to be distributed among the user's tasks to ensure the use of computing resources.

8 . TASK SCHEDULING STRATEGIES AND POLICIES

The process of organizing the computing process in the a distributed cloud environment should take into account the following components:

- Various planning strategies or policies.

- Criteria for optimality or objective function.

- Algorithm for scheduling.

A task scheduling policy is a set of rules by which the final application-level plan of that task should be built. A strategy is also nothing more than a set of planning parameters for each specific task submitted by the user. The selection of a task for execution is based on the procedure for sorting tasks in accordance with their priorities. The following can be cited as examples of the formation of typical strategies:

• Fair distribution of resources (fairshare) - guarantee each job a certain amount of time using the computing resources of the environment, trying to avoid a situation where the job of one user constantly occupies the resource, while the job of another user has not actually started to run.

• Efficiency in the use of computing resources (resource utilization) - it is necessary to occupy a resource for 100% of the working time, not allowing it to stand idle while waiting for tasks ready to be executed. In real computing systems, the processor load ranges from 40 to 90%.

- Reducing the total turnaround time - to ensure the minimum time between the start or queue of a task for downloading and its completion.

- Reduction of waiting time - reduce the time that tasks spend in the queue for loading. User tasks with time requirements for the task will be guaranteed to be completed by the required deadline. In this case, the resulting plan will have the minimum cost and will be executed in the maximum possible time (within the specified period).

- Reduced response time - to minimize the time it takes for a process in interactive systems to respond to a user request.

- Reducing the total cost of performing tasks (budget constraint) - tasks of users with a limited budget are planned within this budget with the cost as close as possible to it.

Actually, the choice of a particular strategy and a specific resource allocation depends on many factors: the load state of nodes, the policy

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of storing and moving data in the environment, the structure of tasks and user estimates of their execution time. When choosing any of the strategies, one should also take into account the possibility of varying the direction of resource load optimization in order to take into account the requirements of the owners of computing nodes.

Planning models should adequately and at the same time reflect the interests of users and owners of resources, the policy of providing and consuming resources and formalized by a set of criteria, which makes the strategy multi-criteria.

Based on the interests of both actors of the cloud environment, we will choose the following goals as a strategy for organizing the computing process:

- *Strategy 1:* minimizing the total cost of performing user tasks;
- *Strategy 2:* maximize the balanced load of physical nodes;
- *Strategy 3:* maximally dense filling of servers with virtual machines;
- Mixed strategies from strategies 1, 2, 3.

To evaluate the schedule, the criteria of optimality are used, otherwise the objective function, which ensures their unambiguous comparison. The objective function is the function of evaluating the schedule according to a given criterion. Thus, it is possible to rank schedules according to their effectiveness. The objective function can contain one or several criteria. The objective function can be based on different criteria: the average execution time of all tasks, the length of the schedule, the average waiting time for the launch of tasks, resource utilization, and others.

9. DISCUSSION

Application of the developed models and methods for organizing computing processes in a distributed cloud environment will provide the required quality of service for resource requests of users in terms of minimizing the total cost of renting computing resources and balancing the load of server resources.

The results obtained in this work are a contribution to the theory of constructing planning systems and organizing computing processes in distributed cloud environments.

The practical significance of the results obtained lies in the fact that the developed comprehensive approach to the process of organizing computational processes in modern types of DCS, such as cloud computing, makes it possible to increase the efficiency of resource use, taking into account the interests of users and resource providers.

Specific practical results form a software complex for remote access to distributed computing resources of a private university cloud, modified to practical use, which implements the methods and algorithms proposed in this work.

The following results were obtained:

1. A model of the organization of the computing process in a distributed environment by distributing virtual machines formulated for the characteristics of individual tasks from the task pool.
2. Rules for setting tasks for the scheduler processing in a distributed environment.
3. Algorithmically developed, software implemented and experimentally investigated version of the modified genetic algorithm.
4. An experimental prototype of a private cloud deployed with the direct participation of the authors, within the framework of which the approbation and implementation of scientific provisions were carried out.

10. CONCLUSION

In this paper, the following results were obtained.

The analysis of modern models of the organization of distributed computing: cluster, peer-to-peer and Grid systems, leased infrastructure of cloud environments. The features (characteristics) of modern types of distributed systems are considered.

An overview of various approaches and methods of resource allocation in distributed computing environments is given.

The drawbacks of the existing methods and algorithms for organizing the computing process in distributed systems are revealed and their inapplicability for organizing computing processes in cloud DCS is indicated.

Based on the results of the analysis of the current situation in the research area, a meaningful formulation of the research problem was formulated, which will be investigated and solved.

A general technique for solving the research problem is proposed.

The essence and mathematical models of the organization of the computational process are proposed.

The basic concepts of the organization of the computing process are determined.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Mathematical models of user tasks and computing environment are proposed.

The rules for setting tasks for processing with the market-value method (MVM) have been formulated.

ACKNOWLEDGMENT

This research has been/was/is funded by the Science Committee of the Ministry of Education and Science of the Republic of Kazakhstan (Grant No. AP05134071).

REFERENCES:

[1] E. Ilavarasan. Task scheduling algorithms for distributed heterogeneous computing systems.- Thesis of Doctor of Philosophy, 2007.

[2] V.P. Gergel',, P.N. Polezhaev.- Issledovanie algoritmov planirovanija parallel'nyh zadach dlja klasternyh vychislitel'nyh sistem s pomoshh'ju simuljatora.-Vestnik Nizhegorodskogo universiteta, 2010 №5(1), p.201-208.

[3] Gordeev A. V. Operacionnye sistemy: Uchebnik dlja vuzov. 2-e izd. —- SPb.: Piter, 2007. - 416 s.: il.

[4] Uwe Schwiegelshohn and Ramin Yahyapour. Analysis of First-Come-First-Serve Parallel Job Scheduling. In Proceedings of the 9th SIAM Symposium on Discrete Algorithms, pages 629{638, January 1998.

[5] Karl Czajkowski, Ian Foster, Nick Karonis, Carl Kesselman, Stuart Martin, Warren Smith, Steven Tuecke. A resource management architecture for metacomputing systems // Lecture Notes in Computer Science Volume 1459, 1998, pp 62-82.

[6] Toporkov V.V. Modeli raspredelennyh vychislenij.- M.: Fizmatlit, 2004.-320 p.

[7] Shapovalov T.S.. Planirovanie vypolnenija zadanij v raspredelennyh vychislitel'nyh sistemah s primeneniem geneticheskih algoritmov: dissertacija kandidata tehnicheskih nauk.- Habarovsk, 2011.- 146 s.: il. RGB OD, 61 11-5/1758.

[8] D. Jackson. The Maui Scheduler. Technical report.

[9] Kovalenko V.N., Semjachkin D.A. Ispol'zovanie Alogritma Backfill v GRID // Institut prikladnoj matematiki im. M.V.Keldysha RAN.

[10] IBM LoadLeveler for AIX 5L using and Administrating Version 3 Release 1,pp.381-394, 2001.

[11] Zhikulin A.A. Metody vysokojeffektivnoj resursnoj modifikacii algoritma Romanovskogo dlja reshenija odnorodnyh raspredelitel'nyh zadach: dis. k-ta tehn. nauk: 05.13.01 / A.A. Zhikulin; DGTU. – Rostov n/D, 2008. – 325 p.

[12] Uossermen F. Nejrokomp'juternaja tehnika. Teorija i praktika.- M.: Mir, 1992. – 240p.

[13] Shtovba S.D. Murav'inye algoritmy: teorija i primenenie// Programmirovanie. 2005. №4.

[14] Lopatin, A.S. Metod otzhiga / A.S. Lopatin // Stohasticheskaja optimizacija v informatike, 2005. - № 1. – S. 133-149.

[15] Holland J.H. Adaptation in Natural and Artificial Systems: An introductory Analysis with application to biology, control, and artificial intelligence. – USA: University of Michigan, 1975. – p. 211.

[16] Kurejchik, V.M. Jevoljucionnye vychislenija: geneticheskoe i jevoljucionnoe programmirovanie / V.M. Kurejchik, S.I. Rodzin // Novosti iskusstvennogo intellekta. – 2003. – № 5. – S. 13-19.

[17] Gorobec V.V. Matematicheskie modeli i algoritmy optimizacii razmeshhenija dannyh billingovyh OLTP-sistem: dissertacija kandidata tehnicheskih nauk.- Novocherkassk, 2014.- 191 s.

[18] Kostenko V.A. Vozmozhnosti geneticheskih algoritmov dlja reshenija zadach sinteza arhitektur i planirovanija parallel'nyh vychislenij.

[19] A.B. Gavriljuk, V.A. Alekseev. Metod optimal'nogo staticheskogo planirovanija zadach v raspredelennyh vychislitel'nyh sistemah s ispol'zovaniem geneticheskogo algoritma,

[20] R.M. Alguliev. Geneticheskij podhod k optimal'nomu naznacheniju zadanij v raspredelennoj sisteme.

[21] Telenik S.F. Upravljaemyj geneticheskij algoritm v zadachah raspredelenija virtual'nyh mashin v COD / S.F. Telenik, A.I. Rolik, P.S. Savchenko, M.E. Bodanjuk // Visnik ChDTU. — 2011. — № 2. —pp. 104—113.

[22] T.S. Shapovalov. Primenenie geneticheskih algoritmov dlja poiska optimal'nogo raspisanija zadanij v GRID .

[23] H. Glibovec, S.A. Medvid'. Geneticheskie algoritmy i ih ispol'zovanie dlja reshenija zadachi sostavlenija raspisanija.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[24] Gorbachev V.N. Optimizacija struktury gibridnogo geneticheskogo algoritma dlja reshenija zadach sinteza raspisanij i raspredelenija resursov. Avtoreferat.

[25] M. Kurejchik, A.A. Kazharov. O nekotoryh modifikacijah murav'inogo algoritma.- Izvestija JuFU. Tehnicheskie nauki. Tematicheskij vypusk «Intellektual'nye SAPR». – Taganrog: Izd-vo TTI JuFU, 2008, № 4(81). – 268 p.- pp.7-12.

[26] Konovalov, M. G. Upravlenie zadanijami v geterogennyh vychislitel'nyh sistemah / M. G. Konovalov, Ju. E. Malashenko, I. A. Nazarova // Izvestija RAN. Teorija i sistemy upravlenija: [nauch. zhurn.]. - 2011. - № 2. - S. 43-61.- ISSN 0002-3388.

[27] Konovalov M.G., Malashenko Ju.E., Nazarova I.A. Modeli i metody upravlenija zadanijami v sistemah raspredelennyh vychislitel'nyh resursov. M.: VC RAN, 2009.

[28] Prohorov A.V., Pahnina E.M. Mul'tiagentnye tehnologii upravlenija resursami v raspredelennyh vychislitel'nyh sredah // Second International Conference «Cluster Computing» CC 2013 (Ukraine, Lviv, June 3-5, 2013). p.184-190.

[29] Konnov A.L. Modelirovanie oblachnyh tehnologij v vychislitel'nyh sistemah // Informacionno- kommunikacionnye tehnologii v obrazovanii i nauke, p. 1854. 2012.

[30] R. K. Uskenbayeva, A. A. Kuandykov, Y. I. Cho and Z. B. Kalpeyeva, "Tasks scheduling and resource allocation in distributed cloud environments," 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014), Seoul, 2014, pp. 1373-1376, doi: 10.1109/ICCAS.2014.6987770.

[31] R. K. Uskenbayeva, J. B. Kalpeyeva and A. B. Kassymova, "Organization of computational processes in distributed cloud environments," 2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Hangzhou, 2015, pp. 947-952, doi: 10.1109/SICE.2015.7285564.