Approaches to the optimization of the placement of service-oriented cloud applications in the software-defined infrastructure of the virtual data center

I. Bolodurina[1], D. Parfenov[1], K. Haenssgen[2]

[1]Orenburg State University, 13 Pobedy ave., 460018, Orenburg, Russia
[2]Leipzig University of Applied Sciences, 132 Karl-Liebknecht-Straße, 04251, Leipzig, Germany

Abstract

Nowadays, we see a steady growth in the use of service-oriented cloud applications in modern business. However, there are some issues related to the placement of service-oriented cloud applications in the software-defined infrastructure of the virtual data center. The goal of optimization is to control the service-oriented cloud applications within data centers. The advantage of modern infrastructure virtualization is the possibility to use software-defined networks and software-defined data storages. However, the existing optimization of algorithmic solutions does not take into account the specifics of working with multiple class service-oriented cloud applications types.
The paper describes the models which describe the basic structures of service-oriented cloud applications including: a level distribution model of the software-defined infrastructure with the technology of cloud applications containerization, a generalized model of a service-oriented cloud application, a model of virtualization of service-oriented cloud applications based on containers. Besides, we developed the efficient algorithm for optimizing the technology of containerization of cloud applications and services in the virtual data center (VDC) infrastructure. We propose an efficient algorithm for placing applications in the infrastructure of a VDC The optimization of the placement of service-oriented cloud applications based on the VM template and containers with VDC disabilities infrastructure is reduced to packing in containers. Besides, we generalize the well-known heuristic and deterministic Karmakar-Karp's algorithms.
During the experimental studies, we have found that the use of our algorithm enables to decrease the response time of cloud applications and services and, therefore, to increase the productivity of user requests processing and to reduce the number of refusals.

Keywords: software-defined network; virtual data center; cloud applications and services; IT infrastructure; virtualization

1. Introduction

The technology of cloud computing is based on the virtualization of the individual components that make up the data center infrastructure. The approaches to the organization of the virtualization layer are divided according to the levels of application of this technology. Typically, the following types of virtualization are distinguished: operating system, software, memory, data storage, databases and network [1, 6]. The most active development level is the virtualization of the operating system. It allows you to create a virtual environment for running multiple instances of user space within the same operating system used to run service-oriented cloud applications within the network environment [24].

Nowadays, cloud applications are a fairly complex multi-level mechanism that interacts with various objects of the network infrastructure of the virtual data center in the course of its work [3, 5]. To deploy them, you need to use an integrated approach that can provide the performance with the least resource consumption. One of the approaches to virtualization, which allows implementing this approach, is containerization technology. A container is an object that provides the user with access to necessary libraries and contains the required set of software for launching the development environment, a ready application or service. Besides, the advantage of using virtualization based on containers for placing applications and services in the network environment of the virtual data center is an easy solution, which enables to reduce costs and increase the productivity of cloud-based service-oriented applications. The control system based on Docker the most effective technology of containerization [4]. The advantage of this technology is its reliability, the availability of open source code and a convenient API for sharing in a networked virtual data center environment. A significant drawback of containers is the inability to provide the proper level of data flows isolation, as well as the lack of support for migration between compute nodes in real time [7-9].

A key difference between virtual machine virtualization and virtualization is the use of a hypervisor that emulates the hardware of a physical computing node. Within each virtual machine, full-fledged operating system functions based on one or several cloud applications or services can be deployed. All this leads to significant overhead in terms of resource consumption in a virtual data center [2].

In this study, we propose a solution that enables to solve the problem of container migration and to optimize the overhead costs for computing resources for the virtual data center infrastructure. The developed solution is based on hybrid virtualization, which is a combination of two approaches to the deployment of cloud applications and services in the software-defined infrastructure of a virtual data center: a container and based on virtual machines.

With the resources of virtualization technology development, the number of layers that form infrastructure decisions and are used in the cloud computing technology is steadily increasing. Nowadays, we can find up to six levels of the software-defined infrastructure of the virtual data center used for the deployment of modern cloud platforms.

The paper is organized as follows. Section 2 is devoted to a level distribution model of the software-defined infrastructure with the technology of containerization of service-oriented cloud applications. Section 3 describes a generalized model of a service-oriented cloud application. Section 4 deals with the model of the virtualization of service-oriented cloud applications based on containers. Section 5 describes an optimization algorithm for the launching and deployment of applications and services in the virtual data center infrastructure using different methods of placement. Section 6 provides the experimental results of our investigation. Section 7 is devoted to the traditional approaches to route traffic based on load-balancing and the solutions of this problem proposed by world scientists. Conclusion section includes a summary of our investigation and the overview of our future work.

Let us describe a model of the virtual data center structure based on the technology of containerized service-oriented cloud applications.

2. The level distribution model of the software-defined infrastructure with the technology of containerization of service-oriented cloud applications

Let us introduce the level model of the software-defined infrastructure of the virtual data center, which supports the containerization method of applications and services placement in the cloud system. The first level is the hardware component of any data center, which includes computing nodes (Nodes), filing systems (Storages) and physical network units (NetObj). Let us introduce it as a set of solutions:

$$PhysLayer=\{Nodes, Storages, NetObj\}. \qquad (1)$$

The next level represents the software-defined layer. This layer consists of the same number of objects as the first level but the main difference is that all the infrastructure elements are dynamic, easily transformed and adjusted within the limits of the physical database network environment. The second level can be presented as the following set of connections:

$$SDLayer=\{SDNodes, SDStorages, SDNetwork\}, \qquad (2)$$

where $SDNodes$ are software-defined computing units; $SDStorages$ are software-defined storages; $SDNetwork$ is a software-defined network.

Above the layer of the software-defined infrastructure, there is a level of the specific objects virtualization. The main objects are computing nodes (VirtNodes), virtual data storages and the elements of the software-defined network used in the work of the cloud platform and consolidated in VirtNetwork multitude.

$$VirtLayer=\{VirtNodes, VirtStorages, VirtNetwork\} \qquad (3)$$

In the software-defined infrastructure, computing nodes and data storages are more often presented as virtual machines that discharge the set of given functions.

To control such multi-layer infrastructure, a separate orchestration layer is needed (the forth level). It contains a number of functions as well as computing nodes and program systems to execute them. The main functions are to orchestrate the virtualization objects (virtual machines and data storages) (ONodes, OStorages) and the software-defined network (ONetwork). Lately, experimental Network function virtualization is also added to them.

$$OrchLayer=\{ONodes, OStorages, ONetwork\}. \qquad (4)$$

The next level (service level) represents the services used in the working process (either the very cloud platform, or applications distributed there, for example, DBMS, Hadoop, Nginx and others).

$$ServiceLayer=\{Service_1, \ldots, Service_n\}. \qquad (5)$$

All the multitude of ServiceLayer cloud services that work in the virtual data center infrastructure can be divided into two disjoint subsets $ServVM \cup ServDocker = ServiceLayer$. The first set (Serv VM) involves services that use virtualization based on other machines. In the second set (ServDocker), there are services based on containers under Docker control.

The top level includes cloud applications that are exploited by users for flexible scalability providing (AppLayer).

$$AppLayer=\{App_1, \ldots, App_m\}. \qquad (6)$$

Like at the previous level, cloud applications App$_i$ can be placed in containers and form the AppVM set. Or they can form the AppDocker set using containerization. At the same time, $AppVM \cup AppDocker = AppLayer$.

Thus, the set of objects of the software-defined infrastructure can be divided into two groups by the methods of placing. Virtual objects that use a container placing method can be referred to the first group. Let us describe them in this way:

$$Docker=\{ServDocker, AppDocker\} \qquad (7)$$

In the second group, there are services and applications that use virtual machines as a placing platform:

$$VM=\{ServVM, AppVM\} \qquad (8)$$

Before we talk about the ways of placing service-oriented cloud applications in the virtual data center, we need to determine their structure, basic parameters, and key characteristics of their operation that affect the efficiency of their use. For this purpose, we have developed a generalized model of a service-oriented cloud application.

3. A generalized model of a service-oriented cloud application

The specific feature of the service-oriented cloud applications is the approach, where users have access to them and to their services; however, they do not know anything about their actual location. In most cases, users only know the address of the aggregation node and the application name. The cloud system automatically selects the optimal virtual machine for the request, on which it is to be processed.

The generalized model of the service-oriented cloud application is a multilayer structure, described in a form of graphs to characterize the connections of individual elements. The model can be represented in the form of three basic layers, detailing the connections of the specific objects of virtual cloud infrastructure: applications, related services and provided resources.

The cloud application is a weighted directed acyclic graph of data dependencies:

$$CloudAppl=(G, V), \qquad (9)$$

Its vertices G are tasks that get information from the sources and process it in accordance with the user requests; its directed edges V between corresponding vertices are a link between tasks in a schedule plan. The schedule plan is defined as a procedure which is prepared to follow the user's request (*SchemeTask*).

Each vertex $g \in G$ is characterized by the following tuple:

$$g = (Res, NAppl, Utime, SchemeTask), \qquad (10)$$

where *Res* are the resource requirements; *NAppl* is the number of application instances; *Utime* is the estimated time for of the users' request execution; *SchemeTask* is a communication scheme of data transmission between sources and computing nodes.

Each directed edge $v \in V$ connects the application with the required data source. It is characterized by the following tuple:

$$v = (u, v, Tdata, Mdata, Fdata, Vdata, Qdata), \qquad (11)$$

where *u* and *v* are linked vertices; *Tdata* is the type of transmitted data; *Mdata* is the access method to the data source; *Fdata* is the physical type of the accessed object (a file in the storage system, a local file, distributed database, data services and so on); *Vdata* is the traffic volume estimated by the accessed data (in Mb), *Qdata* is the requirements for the QoS (quality of services).

The model is original because it enables to calculate the consolidated assessment of its work with data sources for each application. It allows predicting the performance of the whole cloud system.

As mentioned earlier, a cloud service is one of the key slices in the generalized model of a cloud application. The cloud service is an autonomous data source for the application, for which it acts as a consolidated data handler. Generally, the cloud service is highly specialized and designed to perform a limited set of functions. The advantage of connecting a cloud application to the service is the isolated data processing, in contrast to direct access to the raw data, when a cloud application does not use a service. The usage of services reduces the execution time for user requests. The cloud service is described as a directed graph of data dependencies. The difference lies in the fact that from the user's viewpoint, the cloud service is a closed system.

The cloud service can be formalized as a tuple:

$$CloudServ = (AgrIP, NameServ, FormatIN, FormatOUT), \qquad (12)$$

where *AgrIP* is the address of aggregation computing node; *NameServ* is the service name; *FormatIN* is the format of input data; *FormatOUT* is the format of output data.

The aggregator of a service selects the optimal virtual machine; it is executed on this machine. In addition, all its applications are distributed between predefined virtual machines or physical servers. Their new instances are scaled dynamically depending on the number of incoming requests from cloud applications, users or other services.

To describe the placement of cloud applications and services in the data center infrastructure, we have also implemented the model of a cloud resource. A cloud resource is an object of a data center, which describes the behavior and characteristics of the individual infrastructure elements, depending on its current state and parameters. The objects of data center are disk arrays including detached storage devices, virtual machines, software-defined storages, various databases (SQL/NoSQL) and others. In addition, each cloud service or application imposes requirements on the number of computing cores, RAM and disk sizes, and the presence of special libraries on physical or virtual nodes used to launch their executing environments.

Each cloud resource can be formalized as follows:

$$CloudRes = (TRes, Param, State, Core, Rmem, Hmem, Lib), \qquad (13)$$

where *TRes* is the type of resource; *Param* is the set of parameters; *State* is the state of resource; *Core* is the number of computing cores; *Rmem* is the size of RAM; *Hmem* is the size of a disk; *Lib* are for the libraries requirements.

The distinctive feature of the model suggested implies analyzing cloud resources from the user viewpoint and from the viewpoint of a software-defined infrastructure of the virtual data center. The model is innovative, since it simultaneously describes the application data placements and the state of the virtual environment, taking into account the network topology.

We have developed the model of the software-defined storage, which details the resource model of the virtual data center. It is represented in the form of a directed multigraph; its vertices are the virtual data center elements, which are responsible for application data placement (e.g. virtual disk arrays, DBMS and so on):

$$Stg_{ki} = \left(MaxV_{ki}, P_{ki}^{stg}, Vol_{ki}(t), \overline{R}_{ki}(t), \overline{W}_{ki}(t), s_{ki}^{stg}(t)\right), \qquad (14)$$

where $MaxV_{ki} \in N$ is the maximum storage capacity in Mb; $P_{ki}^{stg} = \{p_{kij}^{stg}\}_j$ is the set of network ports; $Vol_{ki}(t) \in N \cup \{0\}$ is the available storage capacity in Mb; $\overline{R}_{ki}(t)$ and $\overline{W}_{ki}(t)$ are read and write speeds; $s_{ki}^{stg}(t) \in \{"online", "offline"\}$ is state of software-defined storage.

The data storage system for applications is like a layer cake. It uses the principles of self-organization of resources. The basis of self-organization of data storages is an adaptive model of dynamic reconfiguration when resources are changing. The model allows optimizing the organizational structure of the cloud platform based on algorithms for searching optimal control nodes and allocating control groups. Our control model assumes two control levels for nodes and resources.

When a software-defined storage is created on each virtual computing node, the software module for exchanging state data between devices is executed. This exchange is carried out within a group of nodes by a single storage method. The least loaded node in the group is selected as the control node. This approach reduces the risk of the control node degradation.

If the control node is failed, the remaining group of virtual machines has all the information about each other, which allows choosing a new control node automatically. Each control node also carries out cooperation with control nodes from other groups to maintain up-to-date information on the state of the entire system.

Thus, the system of software-defined storages is constructed as a hierarchy that includes three basic levels: the level of local access, the level of the controlled group, and the level of data exchange within the whole system. In our model, the description of cloud applications consists of task descriptions and data source descriptions specifying directions and methods of data transfer as well as required resources.

The data obtained allows us to proceed to a description of the model of virtualization service-oriented cloud applications on the basis of containers.

4. A model of the virtualization of a service-oriented cloud applications based on containers

Let us describe the formalized structure and communications of cloud applications and services to describe the model of virtualization using the method of containerization.

Every cloud application can be described as a set of components, which are the following union of sets:

$$App_i = Lib \cup Qu \cup StgData \cup Service \cup PM \quad (15)$$

where Lib is a set of operating system libraries used in the application; Qu is a set of queues formed at the requests of the users accessing the application; StgData is set of storage systems used for placing data for cloud applications; Service is a set of services that uses cloud applications in the course of their work; PM is set of methods for placing cloud applications in th software-defined infrastructure of the data center.

In this study, we consider three methods of placing cloud applications. The first method is based on using virtual machines – P_{vm}. The next method involves the use of containers placed on physical compute nodes – P_d. The last method uses a hybrid approach based on containerization inside a virtual machine – P_{dvm}.

A service-oriented application (App_i) placed in a software-defined infrastructure of data center is a set of instances running in the cloud platform $Vapp \in App_i$. Thus, at each moment of time, the cloud application may be represented as a dynamic weighted directed graph:

$$App_i(t) = (Vapp, Eapp, FlowApp(e_a, t), Iapp(vapp)_i) \quad (16)$$

where Vapp is a set of nodes that represent instances of cloud applications placed in the software-defined infrastructure of the data center; Eapp is the maximum total number of network connections forming the graph of the arc in the process of balancing the requests between instances Vapp; FlowApp(e_a,t) is the function that determines the number of transmitted data on the arc $e_a \in$ Eapp at time $t \geq 0$. If FlowApp (e_a,t) = 0 , no arc e_a at time t; Iapp(vapp)$_i$ is a set of the characteristics of an instance of the cloud applications $vapp \in Vapp$.

In turn, each cloud service in the software-defined infrastructure of the data center can be described by the following set of parameters:

$$Service_i = \{AgrIP, NameServ, Ma, PM, Format\} \quad (17)$$

AgrIP is the address of computing node for requests aggregation to cloud service; *NameServ* is a name of the cloud service placed in the infrastructure of the virtual data center; *Ma* is a set of supported methods of access to the service; *PM* is a set of methods for placing cloud service in the software-defined infrastructure of the data center; *Format* is the data format.

Cloud service is a set of instances running in the virtual data center infrastructure. Thus, it can be represented as a dynamic weighted directed graph:

$$Service_i(t) = (Vserv, Eserv, FlowServ(e_s, t), Iserv(vserv)_i) \quad (18)$$

where Vserv is a set of vertices, which are the running instances of the cloud service;

Eserv is the most complete set of arcs (network connections) allowed between multiple cloud applications vertices (application-level) and service instances $vserv \in Vserv$ and deployed in a virtual data center; FlowServ(e_s,t) is afunction that determines the number of transmitted data on the arc $e_s \in$ Eserv at time $t \geq 0$. If FlowServ (e_s,t) = 0 , no arc e_s at time t; Iserv(vserv) is a set of the characteristics of an instance of the cloud service $vserv \in Vserv$.

Every instance of a running cloud application $vapp \in Vapp$ or a cloud service $vserv \in Vserv$ has the vectors:

$$Iapp(vapp)_i = n_i(t), m_i(t), u_i(t), \Delta t_i, p_i; \quad (19)$$

$$\mathrm{Iserv}(\mathrm{vserv})_i = n_i(t), m_i(t), u_i(t), \Delta t_i, p_i \quad (20)$$

where $n_i(t)$ is the number of requests flows that are processed for one instance at the moment of time t;
$m_i(t)$ is the volume of consumed memory for one instance of application for placing on the computing node at the time t;
$u_i(t)$ is the average load cores on the computing node for one instance of application at the time t;
Δt is the average time of response to the incoming flow of requests for the instance;
$p \in PM$ is a method of placing an instance in the virtual data center infrastructure.

The developed model describes the mapping of the service-oriented cloud applications in the virtual data center infrastructure using different methods of placement.

Let us describe the flow of user requests coming to the service-oriented cloud applications by the following functional:

$$Fur=(U, AppLayer, Q) \quad (21)$$

where U is a set of users; $AppLayer = \{App_1, ..., App_m\}$ is a set of service-oriented cloud applications; Q is a set of user requests.

To ensure the efficient use of the resources of the virtual data center and the required quality of service to users, we formulate the optimization problem. The flow of user requests should be distributed efficiently between the running instances of the service-oriented cloud applications.

$$\mathrm{Fur}(t): Q \to Vapp \quad (22)$$

At the same time, the copies of applications and services should be optimally placed in the virtual data center infrastructure.

$$\mathrm{App_i}(t): Vapp \to PM \quad (23)$$

It was found that the consumption of basic resources of the virtual data center using different methods of the placement of a set of service-oriented cloud applications has a different weight. To take into account this feature, we introduce the model of optimizing the weighting factors k_1, k_2, k_3 for each type of placement. Then, the function of resource consumption will be:

$$Rvapp_i = \sum_{l=1}^{L} k_l Rapp_i \tag{24}$$

$$Rvserv_i = \sum_{l=1}^{L} k_l Rserv_i \tag{25}$$

$Rapp_i$ and $Rserv_i$ –is a basic weight of resource-intensive applications and services, respectively.

We use the expression "optimal placement" to denote the minimum number of the instances of running applications and services. This ensures minimal consumption in the virtual data center resources. Besides, this approach supports the response time within the allowable value for serving maximum number of users per unit time. This can be formalized as follows:

$$\begin{gathered} \sum_{i=1}^{N} \sum_{j=m}^{M} vapp_i^j \, Rvapp_i \to \min \\ \sum_{i=1}^{N} \sum_{j=m}^{M} vserv_i^j \, Rvserv_i \to \min , \\ \sum_{i=1}^{D} \mathrm{Fur}_i \to \max \end{gathered} \tag{26}$$

where $i = 1 \quad D$ is a number of applications received in the interval of time.

This will minimize the number of concurrent computing devices in the virtual data center infrastructure and maximize processing user requests at a given time interval ΔT.

To achieve these requirements, we should observe a number of functional limitations.

The time of response to user's request is limited and must not exceed the permissible value. The following restrictions apply to architecture of applications. The response time of the request queue must be less than the maximum response time to a request to the application. Otherwise, the request not will be serviced. Another limitation is the request time of cloud applications and service response time to a request data for application from storage or services.

$$Tu_{resp} \le Tu_{resp}^{\max} \tag{27}$$

$$Tqu_{resp} + Tapp + Tserv < Tu_{resp}^{\max} \tag{28}$$

5. The algorithm of optimizing the launch and deployment of applications and services in the virtual data center infrastructure using different placement methods

The models presented allow us to choose the most suitable methods of placing the instance of cloud applications and services in the virtual data center infrastructure based on the current load and the incoming flow of requests. The main task of the distribution of cloud applications and services is to choice the number of instances in time interval, which is formulated as making a plan. When accessing to the service-oriented cloud applications, it is especially important to prepare a plan. The load on the compute nodes may vary greatly over relatively short time intervals and depend on the method of placement of the virtual data center infrastructure. To solve the optimization problem, we developed an algorithm to monitor the virtual data center infrastructure and schedule and launch applications and services. It is based on a biased random-key genetic algorithm (BRKGA). However, in comparison with the BRKGA, the algorithm uses the heuristic analysis of request flows and their classification depending on the application placed in the virtual data center.

The enlarged algorithm has the following steps.

Step 1. Evaluate the incoming flow of requests to cloud applications. Group the requests by type of application. Rank the type of application by the number of requests.

Step 2. Count the number of running instances of each cloud application and determine the amount of used cloud services. Determine the load on the physical computing nodes. Rank the cloud applications and services on the load generated by the virtual data center infrastructure.

Step 3. Based on the data obtained in step 1 and 2, compare the data and determine the applications and services that require scaling.

Step 4. For applications and services that are not involved in the processing, to implement release of the resources . Add the minimum number of instances of using containers.

Step 5. Find the applications and services that require scaling and which creating the maximum load on the infrastructure to evaluate the method of placement.

Step 6. Distribute the most loaded applications and services using a hybrid method of placing (containers deployed in a virtual machine).

Step 7. Translate less loaded applications and services, which require scaling, into operation in the virtual machine.

Step 8. Move virtual machines to the least loaded nodes.

The approach used in the proposed algorithm of controlling service-oriented applications takes into account the way of accommodation and organizes the work of the virtual data center. It also takes into account the incoming flow of user requests while adjusting the number of running instances of applications and services.

6. Experimental part

The aim of the experimental research is to define the effectiveness the algorithm for placing service-oriented cloud applications in the virtual data center infrastructure.

To evaluate the performance of applications, we have used the flows of different intensity. In the first case, flows create minimum load capacity (to evaluate response time and delays, which make data center infrastructure) (experiment 1). In the second case, we have created workload applications placed in the data center virtual infrastructure, traditional for each application. Thus, we can to evaluate the application response time (experiment 2). In the third case (experiment 3), we have applied the developed algorithm for load balancing between the instances of applications and services. We have defined the consumption of resources by each of the running instances; therefore, we can predict the required resources for a third computing experiment.

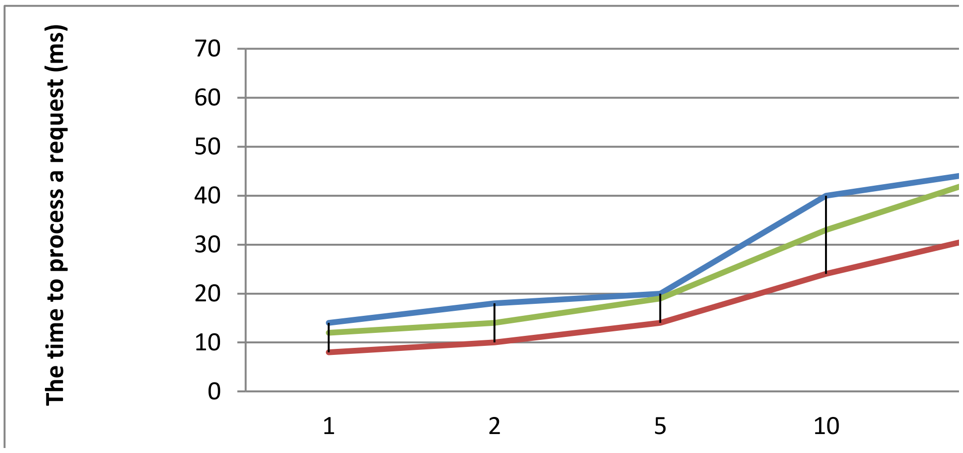

Fig. 1. The result of computing experiment.

The research has shown that the static placement of containers on the physical nodes is not effective because it does not allow redistributing the load quickly. In addition, the movement of the container to another computing unit leads to a loss of the current connections. The placement of applications based on virtual machines due to the flexibility of load balancing showed better results; however, the load on computing nodes has increased considerably due to the additional overhead associated with the use of virtual machines. In this research, the most effective placement was the use of containers inside the virtual machines. It is possible to increase the density of the placement of applications and managed services and software within the data center, Besides, we can place containers as well as data services and network applications in close proximity to each other. Thus, we reduce the time of response to users' requests by applications and increase the efficiency of the system.

7. Discussion

Traditional approaches to route traffic based on load-balancing are reactive. They use simple classical algorithms for distributed computing tasks First Fit or Best Fit. Such algorithms as [10–13] First Come First Served Scan, Most Processors First Served Scan, and Shortest Job First Scan are popular too. Their main disadvantage is poor utilization of a computer system due to a large number of windows in the task launch schedule and problem with "hanging up" when their service is postponed indefinitely due to tasks of higher priority. The solution proposed by D. Lifka from Argonne National Laboratory is usually applied as an alternative method of load distribution between nodes. It is based on the aggressive variant of Backfill algorithm [10, 11, 13] and has two conflicting goals – a more efficient use of computing resources by filling the empty windows schedule and prevention of problems with "hanging up" due to redundancy mechanism. D. Feytelson and A. Weil offered a conservative variant of Backfill algorithm [11]. Further, various modifications have been created by B. Lawson and E. Smyrni [13], D. Perkovic and P. Keleher [14], S. Srinivasan [15]. The main drawback of these algorithms is the time lag during calculation, which is not acceptable for critical services at the time of failure.

In addition to the traditional reactive fault-tolerant technology, such as replication and redundancy to ensure reliability of networked storage cloud platforms, a group of scientists from Nankai University proposed an approach based on the Markov model, which provides secure storage of data without excessive redundancy [16]. However, a significant drawback of this model is the lack of classification and analysis of the types and sources of data to be placed in their consumption. Nevertheless, the model demonstrates a proactive approach that gives certain advantages to achieve the desired resiliency of cloud storage.

Reliability and availability of applications and services play an important role in the assessment of its cloud platform performance. A major shortcoming of existing software reliability solutions in the data center infrastructure is the use of traditional data flow routing methods. In this work, we offer to use the software-defined network technology to adjust the network to the current load of the applications and services that are hosted in a cloud platform before they start using pre-computed and installation routes of transmission (in case of known oriented acyclic graph task dependencies and communication schemes). The principles of a software-defined network first emerged in research laboratories at Stanford and Berkeley, and are currently being developed by the Open Network Foundation consortium, GENI project, the European project OFELIA [17] and the Russian University Consortium for the Development of Software-Defined Network Technology with Orenburg State University as its member.

Centralized decision on the organization of data center heterogeneous infrastructure proposed in the papers has some drawbacks including reliability support, cost of obtaining a complete and current network conditions, low scalability [18, 19]. We assume that the development of a fully decentralized solution is the best option; however, in this case, there is a problem of interaction between the controllers of autonomous systems. We are going to address this issue within a framework of our research using SDX technology, which will be extended to exchange not only information about the network, but also distributed sections and condition of cloud services and applications.

The algorithms for routing data flows in a software-defined network in case of track selection published in scientific sources do not take into account the need to ensure the QoS parameters for the previously installed and routed data flows [21, 22]. We are going to do it within a framework of the developed methods of adaptive network communications routing.

The existing QoS algorithms to provide a software-defined network are also quite inefficient. The paper [20] describes an approach to dynamic routing of multimedia flows transmission that provide a guaranteed maximum delay via the LARAC algorithm (Lagrangian Relaxation Based Aggregated) [22]. However, the authors consider only the cases of single delays on each network connection and do not take into account the minimum guaranteed bandwidth. A similar approach is described in the paper [21]; the authors pose and solve the optimization problem for the transfer of multimedia traffic without losses on alternative routes, leaving the shortcuts for common data.

The researchers from Stanford have offered an algorithm for adaptive control of QoS Shortest Span First, which enables to calculate the optimal priorities for each flow mathematically, to minimize crosstalk influence of flows on delay, to manage priorities dynamically depending on the current situation, and to lay the flow of data transmission through specific port queues [23].

We are going to formulate optimization problems for laying routes with QoS constraints and load balancing within a framework of adaptive routing methods of network communications cloud services and applications developed in this research. In their solution, we may use heuristics similar to the Shortest Span First algorithm. Besides, we will account for the distributed nature of a cloud platform.

The analysis of scientific sources on the topic of the study has shown that:

a) so far, there are no effective algorithmic solutions for planning virtual machines, cloud services, application-oriented accounting topology of the computer system, and communication tasks schemes;

b) the existing solutions for managing distributed scientific computing on multi-cloud platforms plan computing tasks without subsequent adjustment of network to their communication schemes and use traditional routing methods;

c) the existing methods of data flow routing can be enhanced by taking into account the QoS requirements and distributed nature of a heterogeneous cloud platform.

This demonstrates the novelty of the solutions offered by the project.

Thus, the development of new methods and algorithms to improve the efficiency of cloud computing with the use of heterogeneous cloud platforms is a crucial task.

8. Conclusion

We propose an efficient algorithm for placing applications and services in the infrastructure of a virtual data center. The the optimization of placing service-oriented cloud applications based on the VM template and containers with disabilities infrastructure of the virtual data center is reduced to packing in containers We also generalize the well-known heuristic and deterministic Karmakar-Karp’s algorithms. We have developed an efficient algorithm to placing VM by neural network optimization. If we compare the exact algorithm with the developed algorithm, we will find that its approximate solutions do not differ much from the exact solutions.

Thus, the use of the algorithm provides a 12-15% profit compared to conventional methods. This is extremely effective in case of high intensity of requests.

Acknowledgements

The research has been supported by the Russian Foundation of Basic Research (grants 16-37-60086 mol_a_dk, 16-07-01004 a), and the President of the Russian Federation within the grant for state support of young Russian scientists (MK-1624.2017.9).

References

[1] Bein D, Bein W, Venigella S. Cloud Storage and Online Bin Packing. Proc. of the 5th Intern. Symp. on Intelligent Distributed Computing 2011; 63–68.

[2] Nagendram, S. Efficient Resource Scheduling in Data Centers using MRIS / S. Nagendram, J.V. Lakshmi, D.V. Rao // Indian J. of Computer Science and Engineering. – 2011. – Vol. 2. Issue 5. – P. 764–769.

[3] Arzuaga, E. Quantifying load imbalance on virtualized enterprise servers / E. Arzuaga, D.R. Kaeli // Proc. of the first joint WOSP/SIPEW international conference on Performance engineering. – 2010. – P. 235–242.

[4] Mishra, M. On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach / M. Mishra, A.Sahoo // IEEE International Conference Cloud Computing. – 2011. – P. 275–282.

[5] Bolodurina I, Parfenov D. Development and research of models of organization storages based on the software-defined infrastructure. Proc. 39th International Conference on Telecommunications and Signal Processing 2016; 1–6. DOI: 10.1109/TSP.2016.7760818.

[6] Singh A, Korupolu M, Mohapatra D. Server-storage virtualization: integration and load balancing in Data Centers. Proc. of the ACM/IEEE Conf. on Supercomputing 2012; 1–12.

[7] Plakunov A, Kostenko V. Data center resource mapping algorithm based on the ant colony optimization. Proc. of Science and Technology Conference (Modern Networking Technologies), 2014; 1–6. DOI: 10.1109/MoNeTeC.2014.6995596.

[8] Darabseh A, Al-Ayyoub M, Jararweh Y, Benkhelifa E, Vouk M, Rindos A. SDStorage: A Software Defined Storage Experimental Framework. Proc. of Cloud Engineering (IC2E). Tempe: IEEE Press, 2015; 341–346.

[9] Bolodurina I, Parfenov D. Approaches to the effective use of limited computing resources in multimedia applications in the educational institutions. WCSE 2015-IPCE, 2015.

[10] Garey M, Graham R. Bounds for multiprocessor scheduling with resource constraints. SIAM Journal on Computing 1975; 4(2): 187–200. DOI: 10.1137/0204015.

[11] Arndt O, Freisleben1 B, Kielmann T, Thilo F. A comparative study of online scheduling algorithms for networks of workstations. Cluster Computing 2000; 4(2): 95–112. DOI: 10.1023/A:1019024019093.

[12] Feitelson D, Weil A. Utilization and predictability in scheduling the IBM SP2 with backfilling. Parallel Processing Symposium 1998; 542–546. DOI: 10.1109/IPPS.1998.669970.

[13] Lawson B, Smirni E. Multiple-queue Backfilling Scheduling with Priorities and Reservations for Parallel Systems. Lecture Notes in Computer Science 2002; 2537: 40–47. DOI: 10.1007/3-540-36180-4_5.

[14] Perkovic D, Keleher P. Randomization, Speculation, and Adaptation in Batch Schedulers. Supercomputing ACM/IEEE Conference 2000; 7–18. DOI: 10.1109/SC.2000.10041.

[15] Srinivasan S, Kettimuthu R. Selective Reservation Strategies for Backfill Job Scheduling. Lecture Notes in Computer Science 2002; 2357: 55–71. DOI: 10.1007/3-540-36180-4_4.

[16] Jing L, Mingze L, Gang W, Xiaoguang L, Zhongwei L, Huijun T. Global reliability evaluation for cloud storage systems with proactive fault tolerance. Lecture Notes in Computer Science 2015; 9531: 189–203. DOI: 10.1007/978-3-319-27140-8_14.

[17] OFELIA: OpenFlow in Europe. URL: http://www.fp7-ofelia.eu (27.02.2017).

[18] Mambretti J, Chen J, Yeh F. Software-Defined Network Exchanges (SDXs) and Infra-structure (SDI): Emerging innovations in SDN and SDI interdomain multi-layer services and capabilities. Proc. of Science and Technology Conference (Modern Networking Technologies) 2014; 1–6. DOI: 10.1109/MoNeTeC.2014.6995590.

[19] Lin T, Kang J, Bannazadeh H. Enabling SDN Applications on Software-Defined Infrastructure. Network Operations and Management Symposium. IEEE Network Operations and Management Symposium (NOMS) 2014; P. 1–7. DOI: 10.1109/NOMS.2014.6838226.

[20] Ibanez G, Naous J, Rojas E, Rivera D, Schuymer T. A Small Data Center Network of ARP-Path Bridges made of Openflow Switches. 36th IEEE Conference on Local Computer Networks 2011; 15–23.

[21] Shimonishi H, Ochiai H, Enomoto E, Iwata A. Building Hierarchical Switch Network Using OpenFlow. International Conference on Intelligent Networking and Collaborative Systems 2009; 391–394. DOI: 10.1109/INCOS.2009.66.

[22] Egilmez H. OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end quality of service over software-defined networks. Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC) 2012; 1–6.

[23] Kim W, Sharma P, Lee J, Banerjee S, Tourrilhes J, Lee S, Yalagandula P. Automated and Scalable QoS Control for Network Convergence. Internet network management conference on Research on enterprise networking 2010; 1–1.

[24] Bolodurina I, Parfenov D. Development and research of models of organization distributed cloud computing based on the software-defined infrastructure. Procedia Computer Science 2017; 103: 569–576. DOI: 10.1016/j.procs.2017.01.064.