DOI: 10.14529/cmse170206

DYNAMIC ROUTING ALGORITHMS AND METHODS FOR CONTROLLING TRAFFIC FLOWS OF CLOUD APPLICATIONS AND SERVICES*

© 2017 г. I.P. Bolodurina, D.I. Parfenov
Orenburg State University
(460018 Orenburg, Pobedy avenue, 13),
E-mail: prmat@mail.osu.ru, fdot_it@mail.osu.ru

Received: 01.05.2017

Nowadays, we see a steady growth in the use of cloud computing in modern business. This enables to reduce the cost of IT infrastructure owning and operation; however, there are some issues related to the management of data processing centers. One of these issues is the effective use of companies' computing and network resources. The goal of optimization is to manage the traffic in cloud applications and services within data centers. Taking into account the multitier architecture of modern data centers, we need to pay a special attention to this task. 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 specific features of the network traffic formation with multiple application types.

The task of optimizing traffic distribution for cloud applications and services can be solved by using software-defined infrastructure of virtual data centers. We have developed a simulation model for the traffic in software-defined networks segments of data centers involved in processing user requests to cloud application and services within a network environment. Our model enables to implement the traffic management algorithm of cloud applications and optimize the access to storage systems through the effective use of data transmission channels.

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, increase the productivity of user requests processing and reduce the number of refusals.

Keywords: software-defined network, data centers, cloud computing, IT infrastructure.

FOR CITATION

Bolodurina I.P., Parfenov D.I. Dynamic Routing Algorithms and Methods for Controlling traffic Flows of Cloud Applications and Services. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no. 2. pp. 84–98. DOI: 10.14529/cmse170206.

Introduction

Nowadays, we see a steady growth in the use of cloud computing in modern business. This enables to reduce the cost of IT infrastructure owning and operation due to more efficient use of companies' computing and network resources. At present, the solutions for virtual infrastructure are dynamically developing. Thus, the container technology has been lately used for placing cloud applications and services within virtual data centers. Container technologies are mostly based on Docker. Besides, modern data centers rather use virtual infrastructures instead of physical infrastructures especially based on software-defined components: networks, data storages [2, 3], etc. This changes the mechanisms launch and placement management of applications and services.

* The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel Computational Technologies (PCT) 2017".

The most significant changes are found in the SaaS and PaaS service scheduling [4]. Conventional methods of scheduling and distributing resources using the maximum responses to user requests in the High Performance Computing (HPC) systems do not prove to be efficient [1, 5]. This is because the response time in the HPC is less significant than in cloud systems.

Thus, it is important to develop effective scheduling and resource distributing methods for cloud systems that optimize the response time in user requests. To assess the methods, we have developed a simulation model for the traffic in software-defined network segments of data centers involved in the processing of user requests to cloud application and service within a network environment.

The paper is organized as follows. Section 1 is devoted to describing structural model of virtual data centers. Section 2 is devoted to describing simulation model software-defined infrastructure of a virtual data center has several heterogeneous applications and services. Section 3 provides describes an optimization algorithm of adaptive routing and balancing of the application and services flows in a heterogeneous cloud platform, which is located in the data processing center. Section 4 provides experimental results of our investigation. Section 5 is devoted to describing traditional approaches to route traffic based on load-balancing and solution proposed by world scientists for solving this problem. Conclusion section includes summary of our investigation as well as future work overview.

1. A structural model of virtual data centers

To understand the operation principles of a cloud application, we need to define its place in the infrastructure of a virtual data center. A virtual data center is a dynamic object, changing in time *t*, its state can be formalized as a directed graph of the following form:

$$VirtDC = \big(Node(t), Connect(t)\big), \tag{1}$$

where vertices $Node(t) = \{Node_1, \cdots, Node_n\}$ — are active elements of the virtual data center infrastructure (computing nodes, storage systems and others); directed edges $Connect(t) = \{Connect_1, \cdots, Connect_v\}$ — are active user connections to the cloud applications.

The specific feature of 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.

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

The generalized model of a 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:

$$CloudAppl = (G, V), \tag{2}$$

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:

$$g = (Res, NAppl, Utime, SchemeTask), \tag{3}$$

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:

$$v = (u, v, Tdata, Mdata, Fdata, Vdata, Qdata), \tag{4}$$

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:

$$CloudServ = (ArgIP, NameServ, FormatIN, FormatOUT), \tag{5}$$

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:

$$CloudRes = (TRes, Param, State, Core, Rmem, Hmem, Lib), \tag{6}$$

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):

$$Stg_{ki} = \left(MaxV_{ki}, P_{ki}^{stg}, Vol_{ki}(t), \bar{R}_{ki}(t), \bar{W}_{ki}(t), S_{ki}^{stg}(t)\right), \tag{7}$$

where $MaxV_{ki} \in N$ is the maximum storage capacity in Mb; $P_{ki}^{stg} = \left\{P_{ki\ j}^{stg}\right\}$ is the set of network ports; $Vol_{ki}(t) \in N \cup \{0\}$ is the available storage capacity in Mb; $\bar{R}_{ki}(t)$ and $\bar{W}_{ki}(t)$ are read and write speeds; $S_{ki}^{stg}(t) \in \{\text{“}online\text{”}, \text{“}offline\text{”}\}$ 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 model has been used for the development of our data assignment algorithm for software-defined storages, and for our scheduling algorithm for cloud services and applications within the cloud platform.

2. Research methods

Generally, the software-defined infrastructure of a virtual data center has several heterogeneous applications and services. We can assume that the network of a virtual data center encompasses at least three types of application traffic: web-applications, case-applications available on DaaS or SaaS models, and video services. To generate user requests in the simulation model, we apply weight coefficients k_1, k_2, k_3 for each traffic type. Each coefficient allows us to classify requests into types and affects the following set of parameters: running time, routes, priority in the process queue, request intensity, and the distribution law for each type of traffic.

Presented as a multi-channel queuing system, the simulation model of the software-defined infrastructure of the virtual data center includes a user request source (I), a queue (Qs) and a scheduler (S) who manages application hosting and its launch (App). Besides, it contains computing cluster (Srv) and systems of data center storage (Stg). The queuing system is represented in Fig. 1.

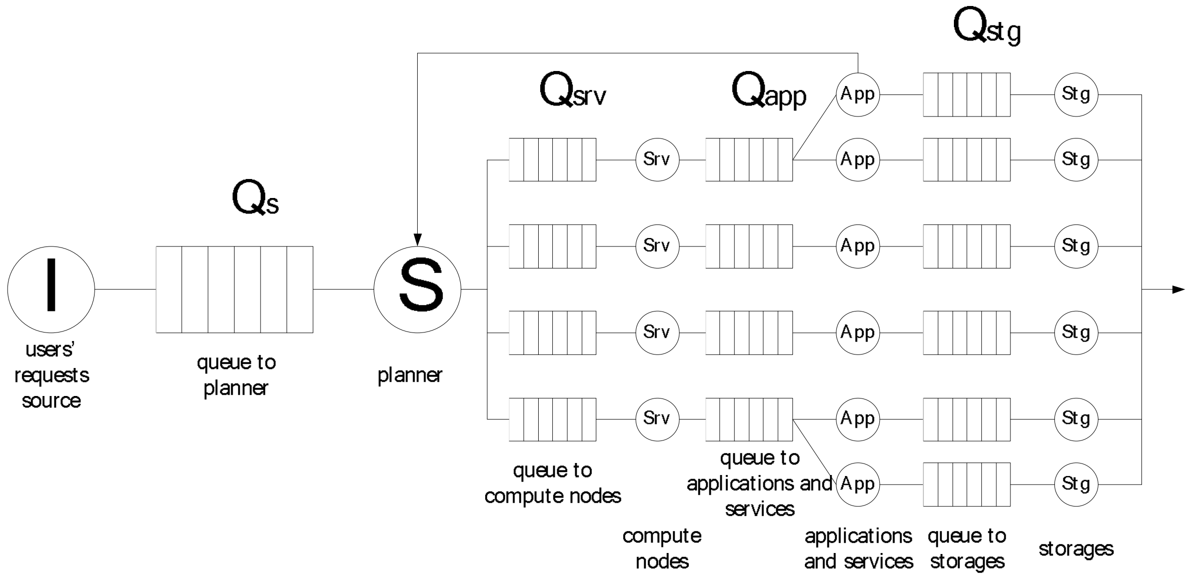

Fig. 1. A queuing system scheme of the software-defined infrastructure of the virtual data center

A queuing system model is stochastic. For its operation, it is necessary to make a user request flow to cloud applications and services with account for the distribution laws and request intensities for each type of cloud applications and services.

To optimize application distribution in the cloud environment of a virtual data center, it is necessary to determine the traffic distribution laws for each application type and distribute the traffic into access objects (virtual servers, containers, and storage systems). For this purpose, it is necessary to set a certain route and make the control law for it within the time interval $T = [t_1, t_2]$.

The dynamic of traffic in cloud applications and services of the software-defined infrastructure of a virtual data processing center can be described by the following discrete system:

$$x_{i,j}(t+\Delta t) = x_{i,j}(t) - \sum_{k=1}^{K}\sum_{l=1}^{N} s_{i,j}(t)u_{i,l}^{j,k}(t) + \sum_{m=1}^{N} s_{m,i}(t)u_{m,l}^{j}(t) + y_{i,j}(t) \qquad (8)$$

where N is the number of virtual nodes within the network; K is the number of application types within the network; $s_{i,j}(t)$ is the capacity of the channels between i-th computing node and j-th storage system $(i \neq j)$; $y_{i,j}(t) = \lambda_{i,j}(t)\Delta t$ is the traffic volume (the number of user requests) at the moment t on the virtual node *i*-th, intended for transferring to the storage system *j*-th; $\lambda_{i,j}(t)$ is the intensity of incoming load, which is defined as the total intensity of the user request flow connecting to the virtual node *i*-th and using the storage system *j*-th; $u_{i,l}^{j,k}(t)$ is a part of the channel transmission capacity in a certain segment of the software-defined network (i,l) at the moment t for the user request flow to the application of type *k*, working with the data storage system *j*-th.

To exclude the possibility of overloading the objects of the virtual data center due to the limited queue buffers on compute nodes, as well as to use data transmission channel capacity efficiently, a number of restrictions are introduced for network parameters of cloud applications in software-defined networks.

The restrictions related to channel capacity limits can be written as follows:

$$0 \le u_{i,l}^{j}(t) \le u_{i,l}^{j(\max)} \le 1; \sum_{l=1}^{N} u_{i,l}^{j}(t) \le \varepsilon_{i,l}^{j,k} \le 1 \tag{9}$$

where $u_{i,l}^{j(\max)}$ is the limit of channel capacity available for the computing node *i*-th in the software-defined network segment *l*-th for traffic transfer to the storage *j*-th; $\varepsilon_{i,l}^{j,k}$ is the part of the channel capacity for the compute node *i*-th in the software-defined network segment *l*-th for the transmission of user requests to the application of type k to the storage system *j*-th.

The use of a software-defined network within a virtual data center allows controlling the queues, which introduces extra restrictions:

$$0 \le x_{i,j}(t) \le x_{i,j}^{(\max)}; \sum_{l=1}^{N} x_{i,j}(t) \le x_{i}^{(\max)} \tag{10}$$

where $x_{i,j}^{(\max)}$ is the maximum queue length allowed on the *i*-th compute node for processing the incoming traffic to storage system *j*-th; $x_{i}^{(\max)}$ is a maximum volume of the buffer allowed on the compute node *i*-th.

Let us consider the system performance as a criterion of optimality, gained through a fixed period $T = [t_1, t_2]$, which is formalized within the model as an objective function in the following form:

$$\sum_{t=0}^{t-1} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} s_{i,j}(t) u_{i,l}^{j,k}(t) \to \max \tag{11}$$

To solve the optimization task, we use an iterative method that allows us to explore the dynamics of the system at the interval $T = [t_1, t_2]$ and control channel capacity for a certain type of applications in a software-defined network.

3. Suggested algorithm

Based on the constructed models, we have developed an optimization algorithm of adaptive routing and balancing of the application and services flows in a heterogeneous cloud platform, which is located in the data processing center. The algorithm aims to ensure the efficient management of the application and services' flows under dynamic changes in the load on the communication channels used to deploy data center software-defined networks by reducing the design complexity for optimal route schemes.

Generalized algorithm is as follows:

```
1. Divide all channels in network in two subsets, $ST$ and $SR$.
2. Generate optimal routes for data flow of the particular class of applications.
3. Determine the points of entry in two subset, $ST$ and $SR$.
4. Search for all the alternative routes at minimum cost.
5. Calculate alternative routes for dynamic load change in the communication channel.
6. Form the full list of alternative paths in network.
7. Define whether there were load changes.
If "Yes" then GO TO Step 8, else GO TO Step 7.
8. Define whether you need to change the route for the current data flow.
If "Yes" then GO TO Step 9, else GO TO Step 13.
9. Calculate metrics connection.
10. Define list of other network objects placed higher in the hierarchy, which performance has decreased.
If network objects found then GO TO Step 11, else GO TO Step 12.
11. Define the new minimum length route for every network object, whose performance has been decreased.
12. Design the new optimal route tree.
13. Transfer the current data flows to new routes. Reshape the list of alternative routes.
GO TO Step 7.
```

Fig. 2. Generalized plan of optimization algorithm of adaptive routing and balancing of the application and services flows in a heterogeneous cloud platform

Let us describe the algorithm in more detail. At the first step algorithm splits the software-defined network multiple channels into a subset of channels, which are included in the tree of routes passing through the ST set segments, a subset of alternative channels passing through the SR set segments.

At the second step algorithm generates optimal routes in the software-defined network for data flow of the particular class of applications in a software-defined network cloud platform based on the communication channels' weight for each of the subsets.

At the step 3 algorithm determines the point of entry into the tree of optimal routes and in the variety of alternative channels for each of the software-defined network channels. At the next step (Step 4) algorithm searches for all the alternative routes at minimum cost, taking into account the channel's weight in the previously constructed tree of optimal routes of the software-defined network for each network object, which is a leaf of the tree. Bind these lists to a network object incident for the reporting channel, located in the hierarchy below.

At the step 5 if a network object is not a leaf of the tree, then calculate alternative routes in the software-defined network for this object, taking into account the weight of the channel in the previously constructed tree of the optimum routes and select the best value of the alternative route. This procedure is performed to generate a list of alternative routes in case of a dynamic load change in the communication channel.

At the step 6 algorithm builds the full list of alternative paths in a software-defined network of the data center for each communication hub. At the step 7 algorithm defines whether there were load changes for some connections by analyzing the received protocol information. If they were, move to Step 8, otherwise – to Step 7. At the step 8 algorithm defines whether you need to change the route for the current data flow using the list of alternative routes. If so, move to Step 9, otherwise – to Step 13.

At the step 9 algorithm for a network object with decreased potential and changed metrics connection in the alternative routes list, defines the minimum length route and put the connection, which led to the network object potential decrease into the optimal route tree and the changed path from the optimal paths tree into a set of alternative SDN routes. When required, you should review the calculations of channel's function weight.

After the transfer to the alternative software-defined network route, at the step 10 algorithm should define if the potential of other data center network objects placed higher in the hierarchy has decreased. If so, move to Step 11, otherwise – to Step 12.

At the step 11 algorithm should define the new minimum length route for every network object, whose potential has been decreased. If the new minimal length route for every object of the network includes a route from the alternative route list, you need to put this route into the software-defined network optimal routes tree and put the route from the optimal routes tree into the set of alternative routes.

At the step 12 Algorithm designs the new optimal route tree in a software-defined network of data center. At the last step algorithm transfers the current data flows of applications and services to affordable data center routes, reconsider the tree entry points; and as for the variety of alternative routes, reshape the list of alternative routes for each network entity that has changed. Then go to step 7.

Application of the proposed algorithm for optimizing the adaptive routing of data flow balancing has allowed us to reduce the complexity of calculating the optimum route to the value *O (k N),* where *k* is the number of completed transitions to alternative routes, and *N* is the number of objects in the software-defined network of data center. Thus, the algorithm is designed to speed up the search and selection of optimal routes for application and service data flows arranged in a heterogeneous cloud platform under dynamic load changes on the communication channels.

4. Experimental results

To assess the efficiency of the developed algorithm for optimizing the adaptive routing of applications and services and balancing data flow in a heterogeneous data center cloud platform, we have conducted a pilot study. We have chosen the Openstack cloud system as a basic platform. For comparison, we have applied algorithms used in the OpenFlow version 1.4 for route control of the software-defined network of the data center in the experiment. We have created a data center prototype with basic nodes and software modules for the developed algorithms that redistribute data and applications flows for the pilot study. To verify the developed algorithm of optimal routing and traffic balancing under conditions of dynamic changes of channels in the software-defined network of the data center, we have deployed several experimental networks consisting of 25, 50, 100, 200, 300, and 400 objects. All generated requests have been reproduced consequently at two pilot sites: with the traditional routing technology (platform 1, NW) and with the technology of software-defined networks (platform 2, SDN). This restriction was caused by the need to compare the results with traditional network infrastructure incapable of dynamic reconfiguration. Two tests were carried out on platform 2. In the first case, we have used the model OpenFlow version 1.4 routing algorithms, in the second case (NEW SDN), we have applied the developed routing optimization algorithm. Experiment time was one hour, which corresponds to the most prolonged period of peak demand recorded in a real traffic network of heterogeneous cloud platform. We have chosen the response time of applications and services that work in a cloud platform as a basic metrics to assess the efficiency of the proposed solutions. The results of the experiment are provided in Fig. 2.

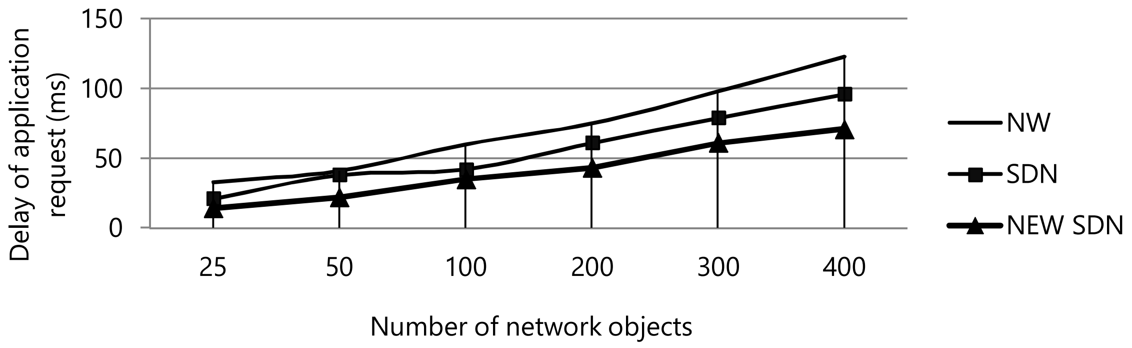

Fig. 3. A schedule of dependence of the response time of applications and services in a heterogeneous cloud platform from the quantities of network objects in the data center

As shown in our research, the algorithm for optimizing the adaptive routing of data flow balancing, based on collected information about alternative routes, has enabled to reduce the response time for applications and services of a heterogeneous cloud platform with a dynamically changing load on channels by 40% compared to traditional networks, and by 25% compared to the model algorithms of the Protocol version 1.4 of OpenFlow. Thus, the algorithm is efficient in designing optimal routes and traffic balancing in SDN data center in case of dynamic changes of load on communication channels.

5. 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 [5–8] 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 [5, 6, 8] 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 [6]. Further, various modifications have been created by B. Lawson and E.

Smyrni [7], D. Perkovic and P. Keleher [9], S. Srinivasan [8]. 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 (Ying Lee, Lee Mingze Ganges Schang, Hiaoguang Liu, Zhongschei Lee, and Huiyun Tang) proposed an approach based on the Markov model, which provides secure storage of data without excessive redundancy [10]. 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 [11], and are currently being developed by the Open Network Foundation consortium, GENI project, the European project OFELIA [13] 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 [15, 16]. We assume that the development of a fully decentralized solution is the best option [17]; 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 [18, 19]. 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 [17] describes an approach to dynamic routing of multimedia flows transmission that provide a guaranteed maximum delay via the LARAC algorithm (Lagrangian Relaxation Based Aggregated) [19]. 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 [18]; 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 [20].

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.

Conclusion

This research has enabled to solve the issue of optimizing the distribution of cloud applications and services data flows in the virtual data center. The developed models have enabled to implement the adaptive algorithm of data flow routing for the effective use of data transmission channels located in the data center. The results show that the algorithm for optimizing the adaptive routing of data flow balancing, based on collected information about alternative routes, has enabled to reduce the response time for applications and services of a heterogeneous cloud platform with a dynamically changing load on channels by 40% compared to traditional networks, and by 25% compared to the model algorithms of the Protocol version 1.4 of OpenFlow. It became possible due to the identification of traffic routes at the initial stage by using the data of the application placement in the network of the virtual data center.

The experimental studies have found that the application of the developed algorithm allows improving the efficiency of user requests processing and reducing the number of failures.

We are going to assess our approach with a larger number of flows of applications, since it will allow us to assess the accuracy of the proposed solution. For this task we prepare software package which can simulate of the routing algorithms enables to identify applications and services in the virtual data center and create the optimal routes for data transmission.

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. Bolodurina I., Parfenov D. Development and research of models of organization storages based on the software-defined infrastructure. *39th International Conference on Telecommunications and Signal Processing*, 2016, pp. 1-6. DOI: 10.1109/TSP.2016.7760818
2. Parfenov D., Bolodurina I., Shukhman A. Approach to the effective controlling cloud computing resources in data centers for providing multimedia services. *11th International Siberian Conference on Control and Communications*, 2015, pp. 1-6. DOI: 10.1109/SIBCON.2015.7147170
3. Singh D., Singh J., Chhabra A. High availability of clouds: failover strategies for cloud computing using integrated checkpointing algorithms. *International Conference on Communication Systems and Network Technologies*, 2012, pp. 698-703. DOI: 10.1109/CSNT.2012.155
4. Aida K., Kasahara H., Narita S. Job Scheduling Scheme for Pure Space Sharing among Rigid Jobs. *Lecture Notes in Computer Science*, 1998, Vol. 1459. pp. 98-121. DOI: 10.1007/bfb0053983
5. Garey M., Graham R. (1975) Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 1975, Vol. 4, Issue 2, pp. 187-200. DOI: 10.1137/0204015
6. Arndt O., Freisleben1 B., Kielmann T., Thilo F. A comparative study of online scheduling algorithms for networks of workstations. *Cluster Computing*, 2000, Vol. 4, Issue 2, pp. 95-112. DOI: 10.1023/A:1019024019093
7. Feitelson D., Weil A. Utilization and predictability in scheduling the IBM SP2 with backfilling. *Parallel Processing Symposium*, 1998, pp. 542-546. DOI: 10.1109/IPPS.1998.669970
8. Lawson B., Smirni E. Multiple-queue Backfilling Scheduling with Priorities and Reservations for Parallel Systems. *Lecture Notes in Computer Science*, 2002, Vol. 2537, pp. 40-47. DOI: 10.1007/3-540-36180-4_5
9. Perkovic D., Keleher P. Randomization, Speculation, and Adaptation in Batch Schedulers. *Supercomputing ACM/IEEE Conference*, 2000, pp. 7-18. DOI: 10.1109/SC.2000.10041
10. Srinivasan S., Kettimuthu R. Selective Reservation Strategies for Backfill Job Scheduling. *Lecture Notes in Computer Science*, 2002, Vol. 2357, pp. 55-71. DOI: 10.1007/3-540-36180-4_4
11. 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, Vol. 9531, pp. 189-203. DOI: 10.1007/978-3-319-27140-8_14
12. Rahme J., Xu H. Reliability-based software rejuvenation scheduling for cloud-based systems. *27th International Conference on Software Engineering and Knowledge Engineering*, 2015, pp. 1-6. DOI: 10.18293/seke2015-233
13. OFELIA: OpenFlow in Europe. Available at: http://www.fp7-ofelia.eu (accessed: 27.05.2016)
14. Kang J., Bannazadeh H., Rahimi H. Software-defined infrastructure and the future central office. *IEEE International Conference on Communications Workshops*, 2013, pp. 225-229. DOI: 10.1109/ICCW.2013.6649233

15. Mambretti J., Chen J., Yeh F. Software-Defined Network Exchanges (SDXs) and Infrastructure (SDI): Emerging innovations in SDN and SDI interdomain multi-layer services and capabilities. *First International Science and Technology Conference (Modern Networking Technologies)*, 2014, pp. 1-6. DOI: 10.1109/MoNeTeC.2014.6995590
16. Lin T., Kang J., Bannazadeh H. Enabling SDN Applications on Software-Defined Infrastructure. *Network Operations and Management Symposium*, 2014, pp. 1-7. DOI: 10.1109/NOMS.2014.6838226
17. 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, pp. 15-23.
18. Shimonishi H., Ochiai H., Enomoto E., Iwata A. Building Hierarchical Switch Network Using OpenFlow. *International Conference on Intelligent Networking and Collaborative Systems*, 2009, pp. 391-394. DOI: 10.1109/INCOS.2009.66
19. 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, pp. 1-6.
20. 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, pp. 1-1.

УДК 004.75, 004.042 **DOI: 10.14529/cmse170206**

АЛГОРИТМЫ И МЕТОДЫ ДИНАМИЧЕСКОЙ МАРШРУТИЗАЦИИ ДЛЯ УПРАВЛЕНИЯ ПОТОКАМИ ТРАФИКА ОБЛАЧНЫХ ПРИЛОЖЕНИЙ И СЕРВИСОВ

© 2017 И.П. Болодурина[1], Д.И. Парфёнов[1]

[1]*Оренбургский государственный университет*

(460018 Оренбург, пр. Победы, д. 13)

E-mail: prmat@mail.osu.ru, fdot_it@mail.osu.ru

Поступила в редакцию: 01.05.2017

В настоящее время доля использования технологии облачных вычислений в современных бизнес процессах компаний неуклонно растет. Несмотря на то, что это позволяет снижать стоимость владения и эксплуатации ИТ инфраструктуры, существует ряд проблем, связанных с управлением центрами обработки данных. Одной из таких проблем является эффективность использования имеющихся в распоряжении компаний вычислительных и сетевых ресурсов. Одним из направлений оптимизации является процесс управления трафиком облачных приложений и сервисов в центрах обработки данных (ЦОД). Учитывая многозвенную архитектуру современных ЦОД, такая задача весьма нетривиальна. Преимуществом современной инфраструктуры виртуализации является возможность использования программно-конфигурируемых сетей и программно-управляемых хранилищ данных. Однако существующие алгоритмические решения при оптимизации не учитывают ряд особенностей формирования трафика в сети с несколькими классами приложений.

В рамках проведенного исследования решена задача оптимизации распределения трафика облачных приложений и сервисов для программно-управляемой инфраструктуры виртуального ЦОД. Предложена имитационная модель, позволяющая описать трафик в программно-конфигурируемых сегментах сети ЦОД, участвующих в обработки запросов пользователей к приложениям и сервисам расположенных сетевой среде, включающей в себя гетерогенную облачную платформу и программно-конфигурируемые хранилища данных.

Разработанная модель позволила реализовать алгоритм управления трафиком облачных приложений и оптимизировать доступ системе хранения, за счет эффективного использование канала для передачи данных.

В ходе экспериментальных исследований установлено что, применение разработанного алгоритма позволяет сократить время отклика облачных приложений и сервисов, и как следствие повысить производительность обработки запросов пользователей и снизить количество отказов.

Ключевые слова: программно-конфигурируемая сеть, центры обработки данных, облачные вычисления, ИТ-инфраструктура.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Bolodurina I.P., Parfenov D.I. Dynamic Routing Algorithms and Methods for Controlling traffic Flows of Cloud Applications and Services // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 2. С. 84–98. DOI: 10.14529/cmse170206.

Литература

1. Bolodurina I., Parfenov D. Development and research of models of organization storages based on the software-defined infrastructure // 39th International Conference on Telecommunications and Signal Processing, 2016, P. 1-6. DOI: 10.1109/TSP.2016.7760818
2. Parfenov D., Bolodurina I., Shukhman A. Approach to the effective controlling cloud computing resources in data centers for providing multimedia services // 11th International Siberian Conference on Control and Communications, 2015, P. 1-6. DOI: 10.1109/SIBCON.2015.7147170
3. Singh D., Singh J., Chhabra A. High availability of clouds: failover strategies for cloud computing using integrated checkpointing algorithms // International Conference on Communication Systems and Network Technologies, 2012, P. 698-703. DOI: 10.1109/CSNT.2012.155
4. Aida K., Kasahara H., Narita S. Job Scheduling Scheme for Pure Space Sharing among Rigid Jobs // Lecture Notes in Computer Science, 1998, Vol. 1459. P. 98-121. DOI: 10.1007/bfb0053983
5. Garey M., Graham R. (1975) Bounds for multiprocessor scheduling with resource constraints // SIAM Journal on Computing, 1975, Vol. 4, Issue 2, P. 187-200. DOI: 10.1137/0204015
6. Arndt O., Freisleben1 B., Kielmann T., Thilo F. A comparative study of online scheduling algorithms for networks of workstations // Cluster Computing, 2000, Vol. 4, Issue 2, P. 95-112. DOI: 10.1023/A:1019024019093
7. Feitelson D., Weil A. Utilization and predictability in scheduling the IBM SP2 with backfilling // Parallel Processing Symposium, 1998, P. 542-546. DOI: 10.1109/IPPS.1998.669970
8. Lawson B., Smirni E. Multiple-queue Backfilling Scheduling with Priorities and Reservations for Parallel Systems // Lecture Notes in Computer Science, 2002, Vol. 2537, P. 40-47. DOI: 10.1007/3-540-36180-4_5
9. Perkovic D., Keleher P. Randomization, Speculation, and Adaptation in Batch Schedulers // Supercomputing ACM/IEEE Conference, 2000, P. 7-18. DOI: 10.1109/SC.2000.10041
10. Srinivasan S., Kettimuthu R. Selective Reservation Strategies for Backfill Job Scheduling // Lecture Notes in Computer Science, 2002, Vol. 2357, P. 55-71. DOI: 10.1007/3-540-36180-4_4

11. 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, Vol. 9531, P. 189-203. DOI: 10.1007/978-3-319-27140-8_14
12. Rahme J., Xu H. Reliability-based software rejuvenation scheduling for cloud-based systems // 27th International Conference on Software Engineering and Knowledge Engineering, 2015, P. 1-6. DOI: 10.18293/seke2015-233
13. OFELIA: OpenFlow in Europe. URL: http://www.fp7-ofelia.eu (дата обращения: 27.05.2016)
14. Kang J., Bannazadeh H., Rahimi H. Software-defined infrastructure and the future central office // IEEE International Conference on Communications Workshops, 2013, P. 225-229. DOI: 10.1109/ICCW.2013.6649233
15. Mambretti J., Chen J., Yeh F. Software-Defined Network Exchanges (SDXs) and Infrastructure (SDI): Emerging innovations in SDN and SDI interdomain multi-layer services and capabilities // First International Science and Technology Conference (Modern Networking Technologies), 2014, P. 1-6. DOI: 10.1109/MoNeTeC.2014.6995590
16. Lin T., Kang J., Bannazadeh H. Enabling SDN Applications on Software-Defined Infrastructure // Network Operations and Management Symposium, 2014, P. 1-7. DOI: 10.1109/NOMS.2014.6838226
17. 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, P. 15-23.
18. Shimonishi H., Ochiai H., Enomoto E., Iwata A. Building Hierarchical Switch Network Using OpenFlow // International Conference on Intelligent Networking and Collaborative Systems, 2009, P. 391-394. DOI: 10.1109/INCOS.2009.66
19. 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, P. 1-6.
20. 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, P. 1-1.

Болодурина Ирина Павловна, д.т.н., профессор, зав. каф., кафедра прикладной математики, Оренбургский государственный университет (Оренбург, Российская Федерация)

Парфёнов Денис Игоревич, к.т.н., доцент, кафедра прикладной математики, Оренбургский государственный университет (Оренбург, Российская Федерация)