Approach to the Effective Use of Limited Computing Resources in Educational Institutions for Providing Multimedia Services

Irina Bolodurina, Denis Parfenov
Orenburg State University
Orenburg, Russia
prmat@mail.osu.ru

Alexander Shukhman
Orenburg State University
Orenburg, Russia
ais@mail.osu.ru

***Abstract*—Existing approaches to the use of computing resources is too wasteful for educational institutions. Modern multimedia services require significant computing power, which are not always available. In this paper, we introduce an approach that allows more efficient use of limited resources by dynamically scheduling the distribution of data flows at several levels: between the physical computing nodes, virtual machines, and multimedia applications with use cloud computing.**

I. Introduction

The information flows between computing nodes in local and global networks has been steadily increasing each year. It is true not only for large data processing centers, but also for locally datacenters (DC) specializing in industry, economy, health and so on. An important area to use local DCs is education. Universities are increasingly using their own DCs to support integrated automated information systems (IAIS), providing end users with network multimedia services.

The need for more resources is one of the problems of high-loaded IAIS. The consumption of resources unlike the available volumes grows exponentially. [5]. The analysis of request flows to IAIS services shows their structure heterogeneity [1]. Modern IAIS services are based on the concept of cloud computing. However, the problem of limited resources used for cloud systems remains relevant [4].

The use of virtualization and cloud computing allows to consolidate several online services located on virtual machines (VM). It reduces the number of physical servers. But to effectively deploy applications on VM it is necessary to solve the problem of resource planning based on variable loads and service level agreement (SLA) [3]. The most flexible architecture of cloud computing is the infrastructure as a service (IaaS). This architecture allows the user to control a pool of computing resources. This approach can imply the start of operating systems and applications, and the creation of virtual machines and networks. Thus, cloud computing leads to significant cost savings due to the increased load density [2].

However, the above is not enough to consolidate computing power, to reduce the infrastructure overheads and to reach optimal performance of cloud systems. To use the cloud infrastructure effectively new methods and algorithms should be developed to control components of cloud systems. It demands determining the formal structure of a cloud system [6].

II. Model of Resource Virtualization of Cloud Systems

In our research, we have developed a model of computing resources of cloud systems. The conception of virtualization of computing resources is based on abstractions representing the tuples of relations between the interconnected elements of subsets.

The cloud system can be represented as a set of interconnected objects. They are computing nodes (*Snode)*, system storages (*Sstg)*, network attached storages (*Snas)* and scheduling servers (*Srasp*). The number of objects and the content of each set may vary depending on the cloud's size and its use.

Each compute node can run multiple instances of virtual machines represented as a set:

$$Snode_i=\{VM_{i,1}, VM_{i,2}, \ldots, VM_{i,k}\}, \quad (1)$$

where k is the number of virtual machines on a compute node i, i = 1...l (l – number of nodes).

Each virtual machine belonging to the set (1) can support several applications and services represented as a set:

$$VM_j=\{App_{j,1}, App_{j,2}, \ldots, App_{j,n}\}, \quad (2)$$

Authors thank for support the Russian Foundation for Basic Research (project 13-07-00198 A).

where n is the total number of applications and services, j=1... m (m - number of VMs).

The network attached storage includes a set of predefined VM images.

$$Snas_y=\{VMimg_{y,1}, VMimg_{y,2}, \dots VMimg_{y,p}\}, \quad (2)$$

where y = 1 ... z (z - number of network attached storages).

Each VM image contains an operating system with preinstalled software and predetermined hardware parameters.

$$VMimg_{y,z}=\{OS_1, OS_2, \dots OS_r\}, \quad (4)$$

The work of entire cloud system is performed using the planning system for certain operations defined by the scheduling servers.

$$Srasp=\{Rtask_1, Rtask_2, \dots Rtask_f\}, \quad (5)$$

The distributed storage system usually consists of failover RAID arrays $Sstg_f=\{RDsik_1, RDsik_2, \dots, RDsik_d\}$ containing the information for multimedia services

$$RDsik_d=\{Data_1, Data_2, \dots, Data_s\}, \quad (6)$$

In addition, the cloud system also contains virtual and physical switches for interconnection between all the components in a network.

Each component of a cloud system $Shcn=\{Snode, Snas, Srasp, Sstg, VM \dots\}$ has the following characteristics:

$$Shcn=(State, Mem, Disk, Diskn, Core, Lan), \quad (7)$$

where $State \in$ {"on","off"} is the state of the component;

$Mem \in N$ is the size of RAM;

$Disk \in N$ is the disk capacity for storage;

$Diskn \in N$ is the number of storage devices;

$Core \in N$ is the number of processor cores;

$Lan \in N$ is maximum bandwidth of the network adapter;

The set of virtual machines can be divided into subsets $VMnode=\{Snode, Snas, Sstg, \dots\}$ to isolate computing resources for different services from each other.

The cloud system is a dynamic object changing at time t. Its state can be formalized in an oriented graph form:

$$Shcn(t) = (Node(t), Connect(t), App(t)), \quad (8)$$

where $Node(t)=\{Node_1, Node_2, \dots, Node_\lambda\}$ are active elements included in one of the sets $Snode_i$, $Sstg_j$, $Snas_k$, $Srasp_m$;

$Connect(t)=\{Connect_1, Connect_2, \dots, Connect_v\}$ are active connections by users to the virtualized applications;

$App(t)=\{App_1, App_2, \dots App_n\}$ are active instances of applications running on virtual resources.

So we determine the structure of a cloud system and mechanisms of its component interaction. In such a system simultaneous servicing heterogeneous user requests is not trivial task.

To optimize the mechanism of access to information system resources it is necessary to analyze the main data flows transferred within the cloud system.

III. Model of Data Flows in Highload Information Systems Based on Cloud Computing

For flows analysis in our study, we used information systems of educational institutions. For analysis the most popular multimedia services have been determined. The research considered distance education systems (DES) consisting of different interactive applications.

In our research has built a level classification of applications:

- Level 1: The subsystem for monitoring the students' knowledge in real time;
- Level 2: The subsystem of the electronic library;
- Level 3: The subsystem of webcasts and webinars.

In our study, we have determined the general features of the use of the local DC's equipment.

- the load on the key resources is periodic and irregular;
- requests to multiple types of resources come at the same time;
- load distribution is not optimal, which results in loss of service at peak loads;
- up to 90% of the load is predetermined, as pre-registration is used for access to resources;
- up to 70% of the load arises due to multimedia educational resources.

Information flows at each level have their own characteristics. The intensity of servicing requested flows in the information system depends on the target application level. In a study we use the statistical analysis of the load on the most popular applications used in information systems of the university. Evaluation time for requests to various applications allow to forecast flows and ensure efficient allocation of resources. We using the goodness of fit chi-square Pearson to obtain data to test the hypothesis of distribution laws requests for incoming flow. In general, the intensity of incoming and service of a request flow for each class of applications is determined by the distribution

function, which is described by the following distribution laws:

- for level 1 - Chi-squared distribution;
- for level 2 - Weibull distribution;
- for level 3 - Pareto distribution.

Flows of data transmitted in the IAIS are usually processed in several phases. At the same time in each phase several similar elements can be used providing balancing and load sharing between the components of the information system. The number of components in each phase depends on the functionality of the information system and the number of applications included in its composition. Suppose an information system has the form:

$$IS = \{S_1, \ldots, S_r\} \quad (9)$$

where S_i - a component that performs data processing on the basis of the incoming flow of user requests, $i = 1..r$ (r – the total number of components of the information system). The number of phases f in the flow path of user requests in an information system depends on its architecture.

The purpose of each phase according to its location in the processing sequence is:

- The first phase is the distribution of data flows between the IAIS resources in the cloud;
- The second phase is the dynamic scaling of the computing resources in the cloud;
- The third phase is data processing by user applications using storage systems and databases.

The components of the third phase include nodes of storage systems and database management systems for providing access to multimedia services in the cloud.

In detail the set of components of an information system is represented in form:

$$IS=\{S^1_1,\ldots,S^1_n, S^2_1,\ldots,S^2_m,S^3_1,\ldots,S^3_k\}, \quad (10)$$

where S^j_i is the i component of the j phase;

$m \in N$, $n \in N$, $k \in N$ are the numbers of components included in the system for the respective phases f.

We also introduce the input components S^0_i which transmit data flows into an information system, and output components S^4_i receiving data flows from the cloud infrastructure. Consequently, the set describing the information system is transformed to:

$$IS=\{S^0_1,\ldots,S^0_l,S^1_1,\ldots,S^1_n, S^2_1,\ldots,S^2_m,S^3_1,\ldots,S^3_k S^4_1,\ldots,S^4_p\}, \quad (11)$$

where $p \in N$, $l \in N$ are the numbers of components in the input and output of cloud information system.

Each component S_i^j of the information system at any time can service multiple requests from different users. In the process of the user request data flows are generated upstream and downstream of the component. Their individual characteristics vary in time.

We designate all the incoming flows of component S_i^j as X_i^j, and the outcoming as Y_i^j, where i is the number of the components at the j service phase. Each request flow can be described as a set of characteristics. Suppose, there are l_i^j incoming flows and p_i^j outcoming flows for a component S_i^j.

Then for the incoming flow $\nu=1..l_i^j$, we introduce a set of characteristics:

$$X_i^{(j,\nu)}(t) = \left(x_{1,i}^{(j,\nu)}(t),\ldots,x_{k,i}^{(j,\nu)}(t)\right)^T \quad (12)$$

where

$x_{1,i}^{(j,\nu)}$ is the intensity of receiving requests in each incoming flow ν of the component S_i^j;

$x_{2,i}^{(j,\nu)}$ is the service time of the request flow v of the component S_i^j;

$x_{3,i}^{(j,\nu)}$ is the intensity of servicing requests of the request flow v of the component S_i^j;

$x_{4,i}^{(j,n)}$ is the service discipline of the flow ν of S_i^j, which determines the order of service in accordance with the prioritization algorithm in the information system;

$x_{5,i}^{(j,\nu)}$ is the service class of the flow ν of S_i^j;

$x_{6,i}^{(j,\nu)}$ is the number of requests received from the flow ν of S_i^j.

For outcoming flow $\mu=1..p_i^j$ of the component S_i^j the feature set includes:

$$Y_i^{(j,\mu)}(t) = \left(y_{1,i}^{(j,\mu)}(t),\ldots,y_{k,i}^{(j,\mu)}(t)\right)^T \quad (13)$$

The service path for each flow can be dynamically changed. The number of unique flows depends on the number of components in each phase.

A set of incoming flows at each phase j can be represented as:

$$X^j = \bigcup_{i=0}^{n_j} X_i^j \quad (14)$$

where j is the number of the service phases, n_j is the number of flows at phase j.

Consequently, all the incoming flows of the information system can be represented as:

$$X = \bigcup_{j=0}^{f} X^j \quad (15)$$

where f is the number of service phases.

For output flows the similar conditions are used :

$$Y^j = \bigcup_{i=0}^{n_j} Y_i^j \Rightarrow Y = \bigcup_{j=0}^{f} Y^j \quad (16)$$

To effectively serve user requests forming data flows in the information system, there must be an single-valued mapping of the form $R : X \to Y$.

In addition, for service of any request at each moment of time the matrix H of transitions between the phases of service is constructed depending on the class of the request and the current load of the system.

The graph of transitions between phases can be built using the function:

$$Y_e^{j-1} = R(X_i^{j,v}), \qquad Y_e^{j-1} \in Y \quad (17)$$

where e is the component of phase j-1 directing data flow v to component S_i^j of phase j, v=1.. l_i^j .

Then for any component S_i^j the set of all the input flows received from component S_i^{j-1} located in the previous phase is represented in the form:

$$X_i^{j,j-1} = R_j^{-1}\left[Y_i^{j-1} \cap R(X_i^j)\right] \quad (18)$$

where j is the phases of service.

Then effluents element S_i^j directed to the element S_i^{j+1} represented in the form:

$$Y_i^{j,j+1} = Y_i^j \cap R(X_i^{j+1}) \quad (19)$$

So $X^{j*} = \bigcup_{i=0}^{n} X_i^j$ and $Y^{j*} = \bigcup_{i=0}^{m} Y_i^j$ can describe the incoming and outcoming flows of phase j respectively.

In real systems, outcoming flows can overlap and get serviced on the same computing node that results in the formation of internal queues at each service phase.

To describe this process it is necessary to determine the connections between output flows of component S_i^j at phase j and all the components at phase j +1. Considering the above the set Y^{j*} becomes:

$$Y^{j*} = \bigcup_{S_i^j}\left[Y_i^{j,0} \bigcup \left(\bigcup_{S_i^{j+1}} Y_i^{j,j+1}\right)\right] \quad (20)$$

For a description of intersecting incoming flows within one phase two functions are introduced:

$$X^{j,j+1} = Q_x^j(Y^{j*}) \quad (21)$$

$$Y^{j,j+1} = Q_y^j(Y^{j*}) \quad (22)$$

where $Q_x^j(Y^{j*})$ characterizes input intersecting flows and $Q_y^j(Y^{j*})$ characterizes output intersecting flows for phase j +1.

Similarly, a set of input flows entering the phase of service can be defined. The flows of user requests can also intersect.

Consequently, an input data flow arriving on the component S_i^j at phase j from all the components at phase j-1 can be represented as:

$$X^{j*} = \bigcup_{S_i^j}\left[X_i^{j,0} \bigcup \left(\bigcup_{S_i^{j-1}} X_i^{j,j-1}\right)\right] \quad (23)$$

To describe the intersecting flows from the phase we introduce two functions:

$$X^{j,j-1} = P_x^j(X^{j*}) \quad (24)$$

$$Y^{j,j-1} = P_y^j(X^{j*}) \quad (25)$$

where $P_x^j(X^{j*})$ characterizes intersecting input flows, and $P_y^j(X^{j*})$ characterizes intersecting output flows from phase j -1.

Thus, the functions (21) and (25) describe the data flows between phases of service in an information system within a cloud.

To describe the whole multiphase information system we formalize the description of flows in each phase in the form $R^j : X^j \to Y^j$.

Thus data flows in an information system within a cloud can be represented as:

$$Y_i^j = R^j(X_i^j) = \begin{cases} R(X_i^j), & X_i^j \in X^j \\ P_Y^j(X^{j*}), & X^{j*} \in \bigcup_{S_i^j} \left[X_i^{j,0} \bigcup \left(\bigcup_{S_i^{j-1}} X_i^{j,j-1} \right) \right] \\ Q_X^j(Y^{j*}), & Y^{j*} \in \bigcup_{S_i^j} \left[Y_i^{j,0} \bigcup \left(\bigcup_{S_i^{j+1}} Y_i^{j,j+1} \right) \right] \end{cases} \quad (26)$$

Data flows and their characteristics may change over time and our representation thereof should also include time *t*.

The description of an information system should include both internal and external factors so the parameter of external influence *F* should be introduced.

Then data flows in a cloud system can be described in the form:

$$Y_i^j = R^j(X_i^j, t, F) \quad (27)$$

IV. Cloud system virtual resources control algorithm

The above models allow to determine the most appropriate computing nodes of the information system and the virtual machines that contain the required instances of multimedia applications. The control system should provide uninterrupted user service and effective virtual resource control in case of limited physical resources.

The main task of the control system is scheduling of computing resources at each moment of time. For highload information systems effective scheduling is important because the load on the services may vary greatly within short time intervals. In a cloud system there is a need to plan resource consumption optimally to prevent resource exhaustion for the application already running.

As distinct from other information systems the flow of user requests in the educational environment is predictable due to the subscriptions for multimedia services. The control algorithm for user access to virtual information resources consists of two interconnected processes.

One of these processes is scheduling. The scheduling algorithm collects data on the incoming requests and classifies them by the levels determined with the priorities of applications for business processes. The input data for the algorithm are the applications described according to the template that includes a virtual machine image with the given configuration of hardware and software and user session characteristics.

Based on this template and data analysis of connections the algorithm calculates the configuration to deploy the required service. In the case of identical sets of VM software the already stored images are used. To optimize the use of computing resources the algorithm generates three variants of virtual machine configurations.

The first variant provides reserve performance in the case of unexpected increase in the number of users. The scaling factor in this case is calculated dynamically.

The second variant provides a predetermined low performance of virtual machines for the given number of users. This approach is most effective for small special purpose user groups. It allows to reduce the overhead in case few working users, the number of subscribers being large.

The third variant uses user-predetermined characteristics,
including a fixed number of running instances of virtual machines regardless of the number of users. In this case the algorithm is only used to limit the computing resources. It calculates the maximum number of virtual machines that are available in the configuration selected by the user.

The second process within the algorithm is direct service of user requests and resource scaling during the work of applications. The algorithm considers the total number of requests from each source which allows to predict the load on the running applications within the cloud. Then the algorithm migrates virtual machines between computing nodes based on the collected data in accordance with a predetermined plan, thereby scaling the work of applications.

For efficient use of resources within the above processes, additional instances of virtual machines are created in the online storage of images for support the applications providing an access for the minimum amount of users.

In the case of predicted load increase on a certain service, the algorithm deploys a full image of the media resource and analyzes the incoming user requests. If the load does not exceed the number of queries in an ordinary flow, the algorithm switches the load to the appropriate image and turns off the virtual machine.

Our approach allows to consider the physical limitations of computing resources and organize the work of a cloud information system adjusting the number of instances of running applications based on the incoming flow of user requests.

V. EXPERIMENTAL PART

We have studied the work of the cloud information system with different parameters to evaluate the effectiveness of our virtual resource control algorithm. We have used the standard algorithms from the cloud system OpenStack [5] as reference for comparison in the experiment.

In the experiment, we used the flow of requests similar to the real flow within the information system of distance learning. The number of concurrent requests received by the system was about 10,000, which is equal to the maximum number of potential users of the system.

All the user requests are classified into six user groups corresponding to the types of user behavior. The requests from the first three user groups directed to the allocated application using other applications at the same time. The groups from 4 to 6 simulate the work of the application in the case of computing resource shortage because of an excess number of concurrent requests.

The intensity of using the system components (video portal, testing system, and electronic library) and the amount of the requested data were assigned for each user group. Experiment lasted for one hour which corresponds to the longest period of peak load in the real system. Experimental results are presented in the Table I.

The results of the experiments show a decrease of 12-15% of the number of service denials in accessing to multimedia services with limited resources. Within the experiment in the OpenStack cloud system we compared the consumption of virtual resources by the number of virtual servers for each of the subsystems.

Our control algorithm provides collaborative work of all running instances of applications in accordance with user requirements due to the optimal allocation of resources on each computing node. So the optimization algorithms may release 20 to 30% of the allocated resources (virtual servers) (Fig. 1).

TABLE I. SERVICE EFFICIENCY OF USER REQUESTS

Systems	testing system	electronic library	video portal	testing system	electronic library	video portal
Experiment		**1**			**3**	
Number of requests	8000	1000	1000	1000	1000	8000
Volume of information	32650	9330	10340	4750	8210	92300
Number of serviced requests (without load balancing)	5443 (4352)	622 (418)	517 (356)	592 (465)	643 (512)	4320 (3985)
The intensity of service	90,71 (72,53)	10,36 (6,96)	8,61 (5,93)	9,8 (7,75)	10,71 (8,5)	72 (66,4)
Experiment		**2**		**4**	**5**	**6**
Number of requests	1000	8000	1000	10000	10000	10000
Volume of information	4250	67200	10670	41700	87600	108000
Number of serviced requests (without load balancing)	632 (525)	5384 (4625)	560 (376)	6753 (5642)	6351 (5215)	5860 (4129)
The intensity of service	10,5 (4,2)	89,73 (77,08)	9,3 (6,26)	112,5 (94,03)	105,85 (89,91)	97,6 (68,81)

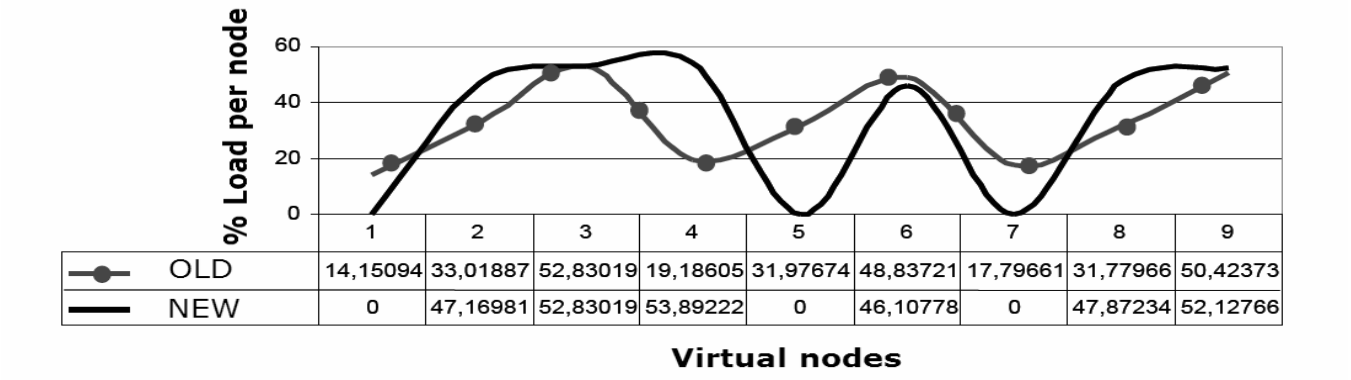

Fig. 1. Load balancing between nodes in the cloud system

VI. Conclusion

Thus, the effectiveness evaluation of the algorithm for control of virtual resources of the cloud system shows a performance boost from 12 to 15% compared to the standard. Our algorithm is very effective for high-intensity requests.

Besides the reduction of the number of allocated virtual resources allows to scale a cloud system more efficiently and provides a reserve for the case of increase in the intensity of using applications.

VII. References

[1] Qingjia Huang, Kai Shuang, Peng Xu, Jian Li, Xu Liu, Sen Su *Prediction-based Dynamic Resource Scheduling for Virtualized Cloud Systems* Journal of Networks, Vol 9, No 2 (2014), 375-383, Feb 2014. http://doi:10.4304/jnw.9.2.375-383

[2] S. J. E. C. I. C. Clark, K. Fraser and A. Warfield, "Live migration of virtual machines, " In Proc. NSDI, 2005.

[3] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on. IEEE, 2007, pp. 119–128.

[4] Q. Huang, S. Su, S. Xu, J. Li, P. Xu, and K. Shuang, "Migration-based elastic consolidation scheduling in cloud data center," in Proceedings of IEEE ICDCSW 2013.

[5] *A scalable infrastructure for CMS data analysis based on OpenStack Cloud and Gluster file system* S Toor et al 2014 J. Phys.: Conf. Ser. 513 062047

[6] A. Corradi, M. Fanelli, and L. Foschini. *VM Consolidation: a Real Case Based on OpenStack Cloud.* Future Generation Computer Systems, In Press.