Proceedings of the Asia-Pacific Advanced Network ***2011*** *v. 32, p. 36-44.*

Scheme of Resource Optimization using VM Migration for Federated Cloud

Yanjue Xu [1,*], Yuji Sekiya [1]

Department of Electrical Engineering and Information Systems, The University of Tokyo/ 2-11-16, Yayoi, Bunkyo-ku, Tokyo, Japan

E-Mails: yanjue@cnl.k.u-tokyo.ac.jp; sekiya@wide.ad.jp

* Tel.: +81 3-5841-3008
http://dx.doi.org/10.7125/APAN.32.5 ISSN 2227-3026

Abstract: Recently, federation of clouds drew lots of attentions, by which we can share the virtual resources among private clouds. In the same time, how to provide an easier way of managing the virtual resources while meeting the needs of the users and the different polices of resources providers has become a challenge.

In this paper we present a new scheme of VM migration using VM live migration technology in the federated cloud environment. This method will be used to detect the overloaded servers and initiate the migration to the optimized location in the cloud automatically, considering the different locations and policies, thus eliminating the hotspots and balancing the load including CPU, memory and network utilization. According to the experimental result, our technique has been proven that it can detect and remove the hotspots efficiently and balance the load.

Keywords: Virtual Machine Migration, Cloud Computing, Resource Management, Migration Strategy, Load Balancing, Federation

1. Introduction

Most infrastructure providers provide two types of clouds: public cloud and private cloud. Nowadays, a new type called federated cloud drew the attention. For example, WIDE cloud [9] is a federated cloud project among several universities in Japan. The federated cloud is the interconnection of the private clouds belonging to different organizations, which make the provision and management of the virtual resources according to the different policies of each organization a complicated task.

In the federated cloud, physical servers are belonging to different infrastructure providers and located in different networks, and all the VMs serve as individual servers running multiple applications, which are unknown to the cloud manager [3]. And like in all other cloud environments, the workloads of VMs fluctuate due to the incremental growth, time-of-day effects, and flash crowds. When a server is overloaded, Service Level Agreement (SLA) will start to degrade, which leads, for instance, that the response time of the request from the user becomes longer. Thanks to the VM live migration technology, the remapping of VMs in a dynamic manner became possible. Therefore, how to allocate the virtual resources dynamically in order to avoid hotspots and improve resource utility considering clouds federation has become a widely concerned problem of cloud computing.

Some researches on dynamic allocation of resources of grid computing have been working on allocating the processes to the CPU resources, rather than consider VM as an individual unit. Other works on VM scheduling and provisioning usually based on the migration within the same data center. The widely used cloud manager, such as OpenNebula [7] and OpenStack [8], have solved the problem of initial provisioning of the virtual resources, but don't consider the automatic dynamic allocation while the workloads and demands of VMs fluctuate in real time. Some commercial products, such as VMware DRS adopts the live VM migration technology to manage the operational cost of the cloud by moving several VMs to a single server and shutting down the idle ones, and to improve the load balance within the single data center. However, it did not take the federation of the cloud into consideration [4, 6].

In order to address these problems, this paper proposes a mechanism to efficiently utilize the idle resources and dynamically remove the hotspots, which is essential for the management of increasing federated clouds. Our approach automates tasks of monitoring the current load of servers and VMs, detecting the overloaded servers, choosing the best physical host for the busy VM, and initiating the migration. During this process, we consider the different locations and policies among organizations those provide the resources. According to the simulation result, our technique has been proven that it can detect and remove the hotspots efficiently and balance the load in the federated cloud environment.

This paper is structured as follows: Section 2 describes the proposed migration algorithm. In section 3, the results of experiments are showed and analyzed. And we will discuss about the future work in section 4.

2. Methods

The cloud manager for the federated cloud should

– monitor the current workload of the physical servers and detect the overloaded servers efficiently.
– optimize the VM replacement considering the federated environment.

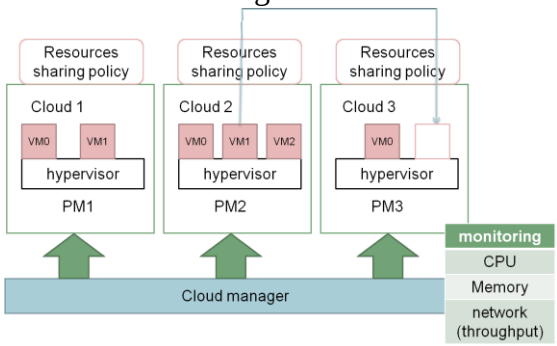


Figure 1. Cloud Manager

As shown in the Fig.1, the cloud manager is composed of monitor component and migration management component.

2.1. Parameters

We need three system inputs to define server overload and make migrating decision. Running a VM on a physical server requires certain amount of CPU, memory and network bandwidth usage [1, 5].

Given the state of each physical server or virtual server, the current load is defined as CPU usage (*CPUu*), memory usage (*MEMu*), network bandwidth usage (*NETu)* over each capacities. We set *CPUcap* as 800 for 8 core machines, 1600 for 16 core machines and 3200 for 32 core machines.

$$LoadCPU = \frac{CPUu}{CPUcap} \qquad LoadMEM = \frac{MEMu}{MEMcap} \qquad LoadNET = \frac{NETu}{NETcap} \tag{1}$$

In this research, we sum up all the current load of VMs running on the same physical server, and assign certain amount of CPU usage for each VM, as the current load of this physical server.

2.2. Hotspot Detection

The hotspot detection algorithm follows a threshold violation manner. This research applies a proactive migration since the threshold is defined in the level before the SLA start to degrade.

The CPU, memory, network bandwidth usage are defined separately, and hotspot will be flagged if the threshold is exceeded any of the dimensions.

$$LoadThreshold = < Tcpu, Tmem, Tnet > \tag{2}$$

2.3. Measuring Multi-dimensional Loads

In order to capture the multi-dimensional loads of the VM for the cloud manager to take migration decision using single parameter, we define *Volume* as a metric to measure the combined CPU, Memory and Network loads of the VMs, while considering the difference of the thresholds.

$$Volume = \frac{LoadCPU}{Tcpu} \times \frac{LoadMEM}{Tmem} \times \frac{LoadNET}{Tnet} \quad (3)$$

2.4. Different Locations

The throughputs influence the speed of data transmission between clouds. In order to optimize the destination of VM migration with both light loads and minimized migration duration, we have to take throughputs into consideration.

We define C as a metric in the migration algorithm.

$$C = \alpha \times Volume + \beta \times \frac{NETbw - NETthr}{NETbw} \quad (4)$$

In equation (6), *NETbw* stands for network bandwidth of the migration source, and *NETthr* stands for the throughput between migration source and destination. We introduce α and β to adjust the weights of the load part and location part in the formula. In the simulation, we set α =1, β = 2, to make the two parts the same weight.

2.5. Selecting Destinations

Once a server is flagged as overloaded, the migration algorithm will be triggered to search for a best migrating destination. Our destination selection deploys a strategy considering both the loads and transmission speed, according to the metric *C* in formula (6).

The cloud manager keeps the table of underloaded servers and overloaded servers sorted by *C*, it will be updated every time when the monitored parameters change during the monitoring interval, or after any VM has been migrated to another server.

In the overloaded server table, each overloaded server has a sub table of their VMs. The VM of the largest value of *Volume* will be mapped to the first in the underloaded server table.

Before initiating the real migration, the cloud manager calculates the server status after migration. Migration process will be executed by the cloud manager if CPU, memory, network bandwidth usage can all be lower than the destination's threshold after migration, the migration

duration is shorter that the time threshold and migration is allowed by the source or destination according to the resource sharing policy.

In the simulation, we overlook the heterogeneous servers when calculate the loads after migration. The CPU, memory or network bandwidth usage of the VM after migration remains the same. The loads of the source and destination servers after migration are shown in equation (7). *U_current* stands for the current usage of the physical servers, *U_vm* stands for the usage of the migrated VM, and *Cap* indicates the capacity of the physical servers.

$$Load\_after = \frac{U\_current \pm U\_vm}{Cap} \tag{5}$$

The value of the migration duration depends on the current memory usage of the VM and the throughput between migration source and destination. The migration process will not be initiated if it does not meet equation (8).

$$Migration\ duration = \frac{MEMu}{NETthr} < T \tag{6}$$

In the process of initial allocation and migration, it should always meets

$$\frac{VMfixed}{VMall} \geq T \tag{7}$$

VMall stands for the number of VMs belong to a certain location. *VMfixe* stands for the number of VMs belong to a certain location and in the same time hosted in this location. We set T to 60%, it indicates that one private cloud are willing to let other private clouds to host maximum 40% of their own VMs.

2.6. Implementation

WIDE project developed a open source cloud manager WCC [9]. It provides a web based application written in Ruby on Rails framework that supports libvirt API, to manage the virtualized servers with hypervisors KVM or Xen, in the federated cloud among universities.

In our research, we aim to add a monitoring and an auto-migrating component to the WCC, in order to optimize the resource utilization. We adopt Simple Network Management Protocol (SNMP), a feasible, UDP-based, lightweight software to monitor the current CPU, memory and network bandwidth usage of each VM and hypervisor. If any server is flagged as overloaded, the auto-migrating component will re-map the VM to the optimized server according to our proposed migrating algorithm. And the cloud manager will execute the migration decision.

3. Results

3.1. Simulation Environment

We did the simulation to evaluate the efficiency of our migration algorithm working in the complicated, real-condition liked federated cloud environment.

Our experiments present results from a simulation program written in C language. In the simulation, we defined a cloud consists 50 servers distributed in three different locations with different capacities and thresholds. Table.1 shows the distribution of the servers. In order to present the VM behavior in the federated cloud, we defined 5 different types of VMs.

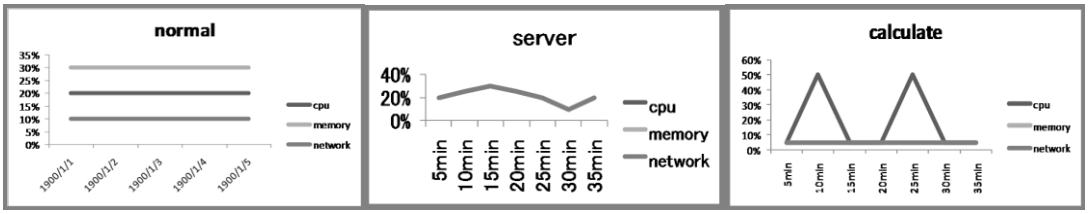
In this simulation, we set the interval of monitoring as 5 minutes.

We managed to do 100 times experiments based on the random data. The average values and 95% confidence interval will be shown in the results.

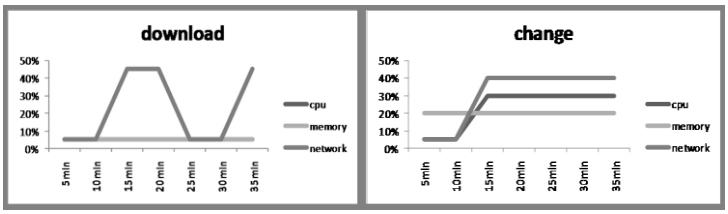
Table 1. Distribution of the Servers

	cloud 1	Cloud 2	Cloud 3
server numbers	16	16	18
CPU	8 core	16 core	32 core
memory	16G	32G	64G
network	1G	1G	1G
CPU threshold	60%	70%	70%
memory threshold	60%	80%	60%
network threshold	70%	60%	70%

3.2. Types of VMs



(a) **(b)** **(c)**



(d) **(e)**

Figure 2. **(a)** Normal node. **(b)** Server node. **(c)** Calculate node. **(d)** Download node. **(e)** Change node.

In the federated cloud, VMs are usually set up according to a certain task. The normal node represents those VMs built up by the individual users to do the routine work such as run the office software or visit the website. So the CPU, memory and network bandwidth usage usually keep in a certain amount. Server node serves as normal mail, ns or website server whose load fluctuate within the certain width when the number of visitors changes. The network bandwidth usage of download node increases during the certain period within the certain circle when users connect to the FTP server and start the download process. In the typical distributed computing system, the sub worker nodes will be assigned the task from the master nodes, accomplishing the calculation task and reporting the answers in a short time. Calculation node represents this behavior by driving the CPU using to the peak in the random short time over a certain time circle. The change node helps us to evaluate the efficiency of the load balancer when the loads of VMs change. In the simulation, the width and peak value of the load of every VMs are randomly assigned by the simulation program.

3.3. Migration Cost

In this simulation, we model the CPU and Network bandwidth usage during the migration. The amount of network traffic of VM migration equals the current memory usage of the VM (*MEMu*). And we define 5% CPU usage in the 8 core server during migration.

3.4. Increase of VMs

The number of VMs increases every 5 minute randomly by from 10 to 15, and decrease randomly by from 0 to 5 until no available server can be found to host the VM. The total number of VMs is increasing nearly linearly. The initial locations of VM are chosen from the most under-loaded servers according to their values of *volume* by the cloud manager within the same location, and the numbers of VMs in every location will be increased equally until one reach the limit. When it gets to the limit, the VMs will be initially allocated in other locations in the same manner.

3.5. Results

Fig.3 shows as VMs increases, the difference of the overloaded server number between 2 simulations.

In the first simulation, throughput between cloud 1 and cloud 2 is 100M, throughput between cloud 1 and cloud 3 is 100M, throughput between cloud 2 and cloud 3 is 500M, and the threshold for the migration duration is 80 seconds(100-100-500,80), in which case VMs from cloud 1 can

be hardly migrated to other cloud. In the second simulation, throughputs are set equally to 500M, and the threshold for migration duration is 200 seconds (500-500-500,200). This enables VMs in three locations transferred freely.

As shown in the Fig 4 and Fig 5, the load balance of the first simulation reaches the peak due to the block of the VMs being migrated to other clouds. While the overloaded server numbers are low because VMs are trapped in the heavily overloaded servers. The performance with larger throughputs is better than the one with smaller throughputs.

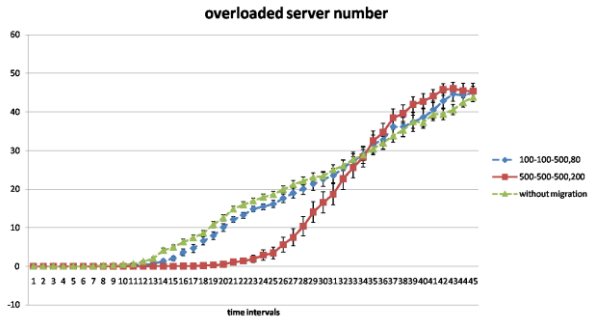


Figure 3. Overloaded server number

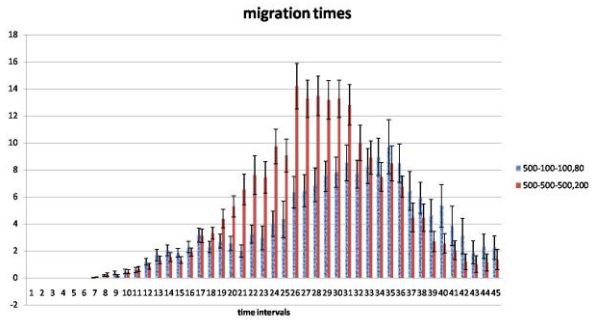


Figure 4. Times of migration

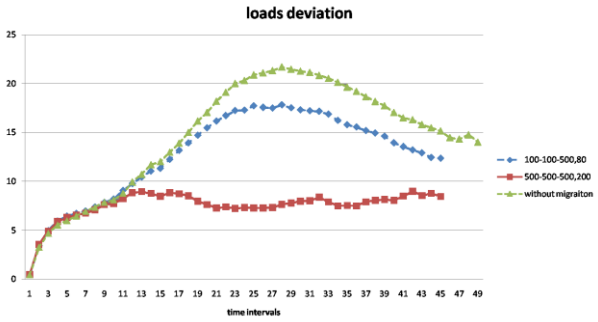


Figure 5. Standard deviation of current load

4. Conclusions

In this research, we proposed a dynamic virtual resources allocation mechanism using light-weighted monitoring by SNMP and live VM migration technology. According to the simulation results, the migrating algorithm works well in detecting and eliminating the hotspots efficiently and balancing the load. And the proposed monitoring-migrating scheme has been proven effective in the federated cloud environment by implementation in the cloud manager WCC. Moreover, since we adopt the open source and widely used technologies, our proposal can be also applied in other cloud managers as a general mechanism to efficiently utilize the virtual resources of different organizations in the federated cloud.

Our future work of this research is:

–Measure the tradeoff of the proposed migration scheme in the actual cloud environments.

References

1. Emmanuel Arzuaga; David R. Kaeli. Quantifying load imbalance on virtualized enterprise servers. In *WOSP/SIPEW '10: Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, ACM, New York, NY, USA, 2010; pp. 235–242.
2. Ferm´ın Gal´an; Americo Sampaio; Luis Rodero-Merino; Irit Loy; Victor Gil; Luis M. Vaquero. Service specification in cloud environments based on extensions to open standards. In *COMSWARE '09: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE,* ACM, New York, NY, USA, 2009; pp. 1–12.
3. Sanjay Kumar; Vanish Talwar; Vibhore Kumar; Parthasarathy Ranganathan; Karsten Schwan. Vmanage: loosely coupled platform and virtualization management in data centers. In *ICAC '09: Proceedings of the 6th international conference on Autonomic computing,* ACM, New York, NY, USA, 2009; pp. 127–136.
4. Liang Liu; Hao Wang; Xue Liu; Xing Jin; Wen Bo He; Qing Bo Wang; Ying Chen. Greencloud: a new architecture for green data center. In *ICAC-INDST '09: Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session,* ACM, New York, NY, USA, 2009; pp. 29–38.
5. Aameek Singh; Madhukar Korupolu; Dushmanta Mohapatra. Server-storage virtualization: integration and load balancing in data centers. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, IEEE Press, Piscataway, NJ, USA, 2008; pp. 1–12.
6. Akshat Verma; Puneet Ahuja; Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Springer-Verlag New York, Inc., New York, NY, USA, 2008; pp. 243–264.
7. OpenNebula, http://opennebula.org/
8. OpenStack, http://www.openstack.org/
9. WIDE cloud, https://wcc.wide.ad.jp/