2010 Ninth International Conference on Grid and Cloud Computing

A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing

Jing Tai Piao
School of Information Systems and Technology
University of Wollongong
Wollongong, NSW, Australia
jp928@uow.edu.au

Jun Yan
School of Information Systems and Technology
University of Wollongong
Wollongong, NSW, Australia
jyan@uow.edu.au

***Abstract*—Cloud computing represents a major step up in computing whereby shared computation resources are provided on demand. In such a scenario, applications and data thereof can be hosted by various networked virtual machines (VMs). As applications, especially data-intensive applications, often need to communicate with data frequently, the network I/O performance would affect the overall application performance significantly. Therefore, placement of virtual machines which host an application and migration of these virtual machines while the unexpected network latency or congestion occurs is critical to achieve and maintain the application performance. To address these issues, this paper proposes a virtual machine placement and migration approach to minimizing the data transfer time consumption. Our simulation studies suggest that the proposed approach is effective in optimizing the data transfer between the virtual machine and data, thus helping optimize the overall application performance.**

***Keywords*—cloud computing, virtual machine, placement, migration, network**

I. Introduction

Based on the underlying virtualization technologies, cloud computing has tremendously changed the IT services delivery paradigm as well as the hardware infrastructures thereof. Thousands of machines are composited as a pool of computation resources to host the virtual machines (VMs) that are processing the end users' requests [13]. In such an environment, the VM that executes an application is placed on a physical machine in order to use the local computation resources to execute the required tasks. At the same time, data can be stored with some geographical or logical distances (i.e. Amazon S3 etc) and these data are accessible to cloud-based applications [10]. For a data-intensive application in cloud computing, the requested data might be spread in a number of vastly distributed data centers.

As an application, especially a data-intensive application, often needs to communicate with related data frequently, the network I/O performance between the data centers that store the data and VMs that execute the applications could affect the performance of the applications significantly. Current VM placement policy mainly focuses on the effectiveness and efficiency of the computing resources utilization [2][11][12], whereas the network aspects are largely ignored. This might make a VM that executes an application be placed on physical machines that are far away from the data centers that store the related data. As a result, the overall application performance and the system overhead would eventually deteriorate due to the costly data transfer time between the application and the data storage.

Furthermore, the virtualization and processor sharing over physical machines often result in the instability of the communication within a cloud computing environment. For example, the TCP/UDP throughput between the small instances in Amazon EC2 varies between 1Gb/s and 0 frequently [4]. The unanticipated network congestion and latency places another challenge to optimizing the data transfer time between VMs and the related data.

This research addresses the above issues and proposes a policy to place the VM with consideration of the network I/O requirement. In addition, a VM migration policy is presented to deal with the situation in which the unstable network connection deteriorates the application performance and likely to jeopardize the existing agreement between the cloud service provider and the end user. In the proposed VM placement policy, our approach optimizes the data access by placing a VM on the physical machine with the smallest data transfer time to the required data. In the proposed VM migration policy, VM migration is triggered when the data transfer time crosses a certain threshold due to the unstable network. Then the next optimized physical machine is chosen according to the current network conditions, and the VM is migrated to this physical machine for better performance. Here, the threshold can be determined by a time-related Service Level Agreement (SLA) between the cloud facility provider and the cloud user. Our experiments suggest that the average task completion time is reduced because of the optimized location of the VMs. In terms of the proposed VM allocation policy, the task can access related data in a shorter time because the hosted VM is allocated on the physical machine that has better network connection status. In terms of the proposed VM migration policy, the VM would be reallocated if the network connection were deteriorated to an intolerable extent so that the tasks can still running on the alternative VM with a shorter average completion time.

The rest of this paper is organized as follows. In Section 2, the related work is discussed. Section 3 explains the proposed network-aware VM placement and migration approach. After that, Section 4 presents the experimental results, followed by the conclusions and future work in Section 5.

II. Related Work

Computation resources management and placement are critical issues in cloud computing. Over the recent years, quite a lot of efforts have been made by researchers to address the problems in this area.

In general, contemporary VM placements approaches focus on optimizing the efficiency and effectiveness of computation resources utilization, allowing VMs to share the capacity of computation resources on the physical hosts. The VM placement problem is modeled as either the Constraint Satisfaction Problem (CSP) [12] or the Constraint Multiple Knapsack Problem (CMKP) [11] to maximize the utility of computing resource. Other approaches focus on minimizing the cost of using computing resources [7]. These approaches largely ignored the network performance and its impact on the overall application performance. VMs would be allocated with a non-optimized physical distance to the relevant data. This would then cause an extra data transfer overhead and eventually lead to the degradation of application completion time.

In terms of VM migration, there have been a few proposals [3][5][8][9] to enable a VM to migrate from one physical machine to another for different purposes such as improving the power efficiency and satisfying performance requirements. Some migration approaches also considered the network performance and communication costs when determining migration policies. However, the existing research heavily relies on the statistical methods to migrate the files that most frequently communicate with each other. Thus, the optimization procedure has to be conducted after a relatively long period to obtain statistics. This drawback makes the statistic method based VM allocation or migration approaches hard to fit a runtime circumstance. The optimization approach should be implemented on the applications that interact with certain data in a relatively long term. Yet, if the applications communicate with data in a short time, the lack of statistics may cause these allocation or migration approaches inapplicable.

III. Proposed Approaches

A. Scenario

Data centers that are operated by a cloud not only provide a flexible data storage space (e.g Amazon S3 etc.) but also support the underlying virtualization infrastructure. This innovated technology facilitates the user to request data storage space and computation resources to form one or more VMs with arbitrary computational capacity. For example, Amazon S3 offers a data storage service that is priced between \$0.037 per GB to \$0.110 per GB. In the meanwhile, the Amazon EC2 provides various types of instances with different computational capacity, including standard instances, micro instances, high memory instances, and so on. Figure 1 shows the scenario in which users request the data storage space and VMs to store data and host applications to process these data, respectively. In this scenario, the applications request to access the related data across the Internet or Intranet and the link between the clouds might be logical or physical.

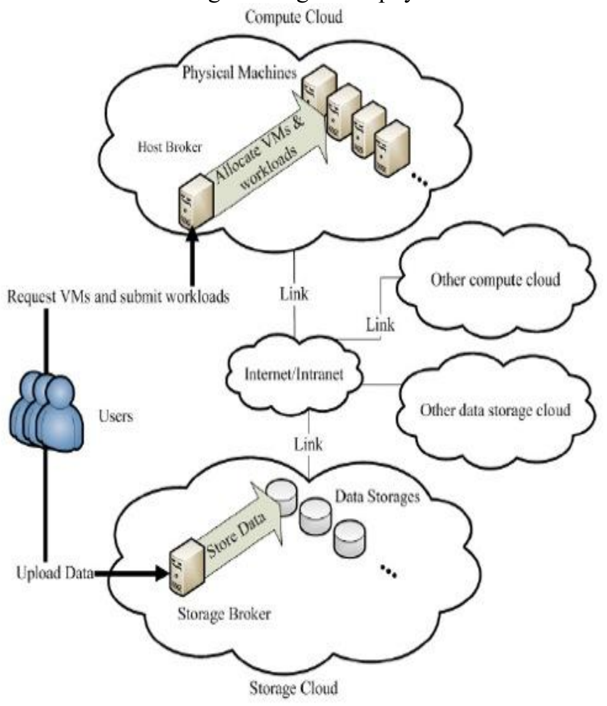

Figure 1. The Studied Scenario

Under current VM allocation policy, it can be seen that the data is stored arbitrarily and can be distributed vastly across the whole storage cloud or even over several storage clouds. The applications are allocated by the broker regardless of the data access time. This will result in the problem that the application may access its related data over a unnecessary distance. To solve this problem, the proposed VM placement and migration approach could be deployed in the host broker to allocate the VMs to the physical machine with a minimized data access time.

B. The VM placement Approach

Our approach assumes that the data which are requested by the data-intensive application is distributed within federated storage resources (e.g. Amazon S3) in advance. And the bandwidth between the storage and the host of the VM is already known.

Within the context of cloud computing, the data for a given application could be distributed in several blocks and the blocks could be stored with logical/physical distances. According to the status of data distribution, a data distribute matrix $D_{n,m}$ can be implemented as

$$D_{n,m} = \begin{vmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,m} \\ d_{2,1} & d_{2,2} & \dots & d_{2,m} \\ \dots & \dots & \dots & \dots \\ d_{n,1} & d_{n,2} & \dots & d_{n,m} \end{vmatrix}$$

Here, the column number *m* represents the total number of data which will be accessed by the applications and the row number n represents the number of data storage nodes.

For example, a data (data0) could be separated in two blocks and stored in storage0 and storage2 respectively, then the matrix D can be demonstrated as below:

$$D_{2,2} = \begin{vmatrix} data0_{Block1} \\ 0 \\ data0_{Block2} \end{vmatrix}$$

In addition, for each column of the matrix $D_{n,m}$, each data should satisfy the following relationship:

$$DataSize_m = \sum_{i=1}^{n} d_{i,m} \qquad (1)$$

The network speed between a physical machine which hosts the VM and a data storage node is represented by function *Speed*, namely

$$S = Speed(s, \Delta t) \qquad (2)$$

Here, *s* represents the size of the package and Δt is the package transfer time slot. Within a certain time interval, the network speed could be fluctuated. In addition, the data transfer speed could be different for different applications.

Within the cloud computing environment, VMs process the applications and are hosted on the physical machines. In our approach, the network status between the VMs is denoted by the matrix $S_{i,j}$, where *i* indicates the number of physical nodes that are available to store data so that matrix $D_{n,m}$ and $S_{i,j}$ satisfy:

$$j = n \qquad (3)$$

The value of each element in matrix $S_{i,j}$ is the inverse of the *Speed(s, Δt)* between the physical machine and the data storage node. Therefore, the matrix $S_{i,j}$ can be obtained as follows:

$$S_{i,j} = \begin{vmatrix} s_{1,1}^{-1} & s_{1,2}^{-1} & \dots & s_{1,j}^{-1} \\ s_{2,1}^{-1} & s_{2,2}^{-1} & \dots & s_{2,j}^{-1} \\ \dots & \dots & \dots & \dots \\ s_{i,1}^{-1} & s_{i,2}^{-1} & \dots & s_{i,j}^{-1} \end{vmatrix}$$

The data access time matrix $T_{i,m}$ represents the data access time from each physical machine to the related data. The matrix $T_{i,m}$ can be obtained by following formula:

$$T_{i,m} = S_{i,j} \times D_{n,m} \qquad (4)$$

For instance, a data access time matrix

$$T_{2,2} = \begin{vmatrix} 3 & 4 \\ 2 & 5 \end{vmatrix} \qquad (5)$$

denotes the following information:

- The array index *i* = 2 represents there are two physical machines, using notation h_1 and h_2 denote these two physical machines that host VMs.
- The array index *m* = 2 represents the related data is stored separately with two pieces, using notation d_1 and d_2 denote these two data.
- The access time from h_1 to d_1 is 3 milliseconds, whereas the access time from h_1 to d_2 is 4 milliseconds.
- The access time from h_2 to d_1 is 2 milliseconds, whereas the access time from h_2 to d_2 is 5 milliseconds.

For an arriving application that intends to access d_1, it is desirable to place the VM that will host this application on h_2 because it has the shortest data access time.

If the next arriving application intends to access d_1 as well, it is necessary to check whether the idle computation resources on the h_2 satisfy the requirement of upcoming task. Assuming that the computing resource set

$$C = \{P, M\} \qquad (6)$$

indicates the total computation capacity on a physical machine, where the notation *P* represents the processor capacity on the physical machine and *M* denotes the memory capacity on the physical machine. The set

$$C_{occupied} = \{P_{VM_i}, M_{VM_i}\} \qquad (7)$$

represents the computation capacity requirement of VM_i. Therefore, the available computing resource set can be represented as:

$$C_{Available} = \{P_{Available}, M_{Available}\} = \{P - \sum_{i=1}^{n} P_{VM_i}, M - \sum_{i=1}^{n} M_{VM_i}\} \qquad (8)$$

where *n* indicates the number of running VMs. Using

$$C_{VM_{arriving}} = \{P_{VM_{arriving}}, M_{VM_{arriving}}\} \qquad (9)$$

denotes the computation capacity requirement of the arriving VM, if the available computation resource on the physical machine satisfies

$$P - \sum_{i=1}^{n} P_{VM_i} > P_{VM_{Arriving}} \qquad (10)$$

and

$$M - \sum_{i=1}^{n} M_{VM_i} > M_{VM_{Arriving}} \qquad (11)$$

then the VM would be placed on this physical machine. For simplicity, we use $C_{Available} > C_{VMArriving}$ to indicate the relationship in (10) and (11). After the placement of the arriving VM, the new available computation resource can be obtained by the formula:

$$C_{Available} = \{P_{Available} - P_{VM_{Arriving}}, M_{Available} - M_{VM_{Arriving}}\} \qquad (12)$$

For a given application α that requests data $d_1, d_2, \ldots, d_\omega$, where $\omega \in m$, the process of proposed network-aware VM placement algorithm can be demonstrated by the following pseudo code:

```
Input: data storage matrix $D_{n,m}$
Input: network status matrix $S_{i,j}$
Input: a coming application requested data set
( $data_0, data_1, \ldots, data_\omega, \omega \in m$)
1. Calculating data transfer time matrix
$T_{i,m} = S_{i,j} \cdot D_{n,m}$
2. Traverse all of the column in the matrix $T_{i,m}$ to find the
minimum $\sum_{y=1}^{\omega} T_{x,y}$ && $C_{Available} > C_{VMArriving}$
3. return x;
4. Allocating the arriving VM on the host $h_x$, $h_x \in h_i$.
5. if (the VM is allocated succeed )
6. $C_{Available} = \{P_{Available} - P_{VM_{Arriving}}, M_{Available} - P_{VM_{Arriving}}\}$
7. end if
                    End Algorithm
```

C. The VM Migration Approach

Due to the inconstant network feature in cloud computing, the unexpected network latency might occur and result in performance degradation of data-intensive applications [4]. In some extreme cases, the data access time might be intolerable for the application. Eventually, this decline of the network status may jeopardize the SLA between the cloud service provider and service user. In this research, the SLA only specifies the execution time for an application to complete processing its related data.

The proposed migration approach is triggered when the execution time crosses the threshold that is specified in the SLA. For a given application which will interact with data (d_1, d_2, …, d_ω, $\omega \in$ m), the total data access time can be denoted as:

$$\sum_{y=1}^{\omega} T_{x,y} \qquad (13)$$

Assuming that the T_{SLA} is a time threshold which is requested by the end user, the migration mechanism could be triggered when

$$\sum_{y=1}^{\omega} T_{x,y} > T_{SLA} \qquad (14)$$

The pseudo code is shown below:

For a running application which requests data ($data_0$, $data_1$,…, $data_\omega$).

```
Input: threshold $T_{SLA}$
Input: data transfer time matrix $T_{i,m}$
Input: a coming application requested data set
($data_0, data_1, \ldots, data_\omega, \omega \in m$)
1. if ( $\sum_{y=1}^{\omega} T_{x,y} > T_{SLA}$ )
2. traverse the entire matrix $T_{i,m}$ to find the minimum
$\sum_{y=1}^{\omega} T_{x,y}$ && $C_{Available} > C_{VMArriving}$
3. return x;
4. Reallocating the VM to the host $h_x$, $h_x \in h_i$.
6. end if
7. if (the VM is allocated succeed )
8.
$C_{Available} = \{P_{Available} - P_{VM_{Arriving}}, M_{Available} - P_{VM_{Arriving}}\}$
9. end if
                    End Algorithm
```

After the migration process, the data access time should be optimized by reallocating the VMs on the host with better network status. However, it is possible that there is no such a place satisfies:

$$\sum_{y=1}^{\omega} T_{x,y} < T_{SLA} \qquad (15)$$

In order to avoid the situation that the migration is iterated forever, our approach will only migrate the VM on the available host with minimum data access time when the migration is triggered.

IV. Experiment

To evaluate the effectiveness of the proposed approach, an experiment has been designed and conducted. The proposed VM placement policy is implemented and tested using Cloudsim 2.0 which supports modeling and stimulating one or more VMs on a stimulated data center [1][2]. In the experiment, we focused on the average task completion time and compared our policy with the default VM placement policy adopted by Cloudsim 2.0. Besides, the matrix ($S_{i,j}$) is modified in the experiment to stimulate a degradation of data access status.

In the initial phase of the experiment, 3 files (i.e., data), *file1*, *file2*, and *file3*, are stored in 2 storages, storage0 and storage1. The size of the files is 6000MB, 8000MB and 4000MB, respectively. Here, *file1* is stored in *storage0*, whereas *file2* and *file3* are both stored in *storage1*. We implemented one data center, *datacenter0*, with 3 hosts (i.e., physical machines), *host0*, *host1* and *host2*. Each host has a fixed computation capacity. The connection speeds between the hosts and storages are represented by a 3× 2 matrix, and the storage of data is defined by a 2 × 3 matrix. Based on the VM allocation policy, we can calculate a 3 × 3 data transfer time matrix. In this experiment, the data transfer time is set as:

$$\begin{vmatrix} 480.0 & 160.0 & 80.0 \\ 240.0 & 240.0 & 120.0 \\ 540.0 & 480.0 & 240.0 \end{vmatrix}$$

Besides, a broker (i.e., user) submitted a request that asks for computation resources to support 3 VMs. The broker then submitted various numbers of cloudlets (i.e., tasks) to those VMs. There are two VM placement policies implemented during the experiment, namely the default VM placement policy of Cloudsim 2.0 and the proposed network aware VM placement policy. The default VM placement policy of Cloudsim, known as *VMSimpleAllocationPolicy*, allocates the VM on the least utilized host while the proposed approach allocates the VM based on network conditions. These two placement policies were compared in three experimental groups. All cloudlets in group 1, 2 and 3 requested *file1*, *file2*, and *file3* respectively, and the number of cloudlets were increased gradually (5 more cloudlets per round). The comparisons of the average task completion time in three groups are shown in Figures 2, 3, and 4, respectively. It is clear that the proposed approach resulted in shorter average task completion time in all three groups. In fact, the decline of the average task completion time is caused by the optimized location of the hosted VMs, comparing with two different VM allocation policies. The results proved that the proposed network aware VM allocation policy can lead to improved task completion time.

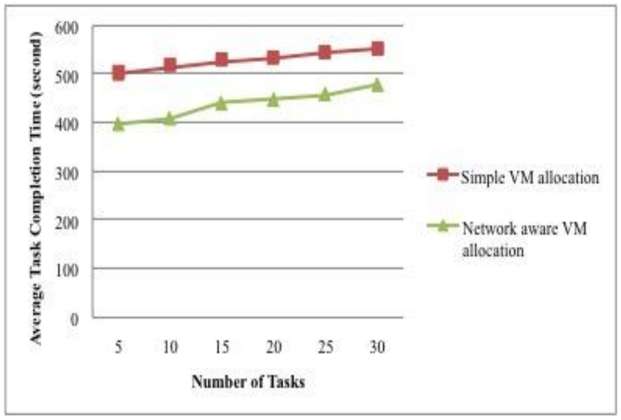

Figure 2. The Results of Cloudlets Requesting *file1*

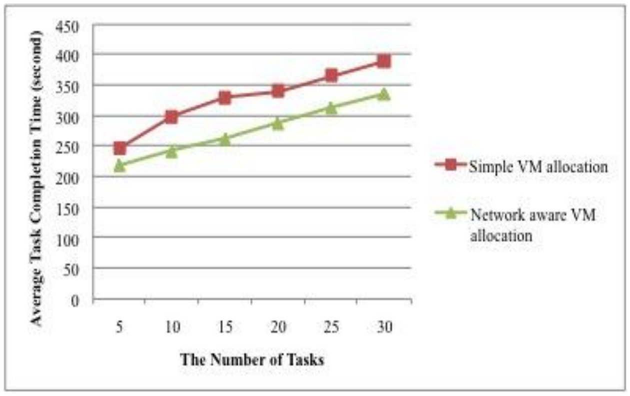

Figure 3. The Result of Cloudlets Requesting *file2*

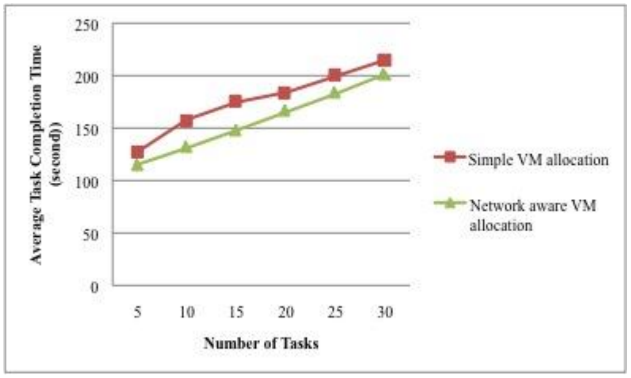

Figure 4. The Result of Cloudlets Requesting *file3*

The above three experiments demonstrated that the proposed allocation mechanism is useful to improve the execution time of tasks. Due to the different network status and different data distribution, the improvement varies from 100s to 12s.

At the final stage of our experiment, the initial network status matrix $S_{i,j}$ is modified to simulate unexpected network latency. Consequently, the change of $S_{i,j}$ should lead to a change on data access time matrix $T_{i,m}$. So, we assume that the data access time between the *host0* and *storage1* is over a certain threshold. And a variable number of tasks which request *file3* are submitted to the VMs. Before the migration policy is implemented, both of the *vm0* and *vm1* are hosted by *host0*, whereas *vm2* is hosted by *host1*. After implementing the proposed migration policy, *vm0* and *vm1* are migrated to *host1* so that the average task completion time is decreased. The comparison of the average task completion time between before and after implementing the VM migration policy is conducted and the results is shown in Figure 5.

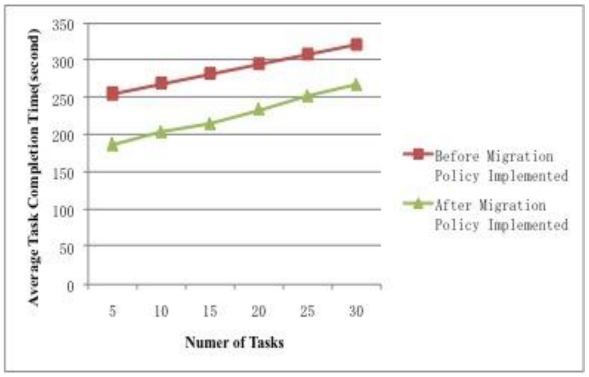

Figure 5. The Comparsion of Before and After Migration Policy Implemented

V. Conclusions And Future Work

While the data-intensive applications interact with distributed data in a cloud computing environment, the network status between the application and data could influence the performance of the application significantly. Thus, it is necessary to control the location of the VMs so that the applications hosted by the VM can obtain a shorter data access time. To address this issue, this paper presents a network aware VM placement and migration approach for data intensive applications in cloud computing environments. The proposed approach places the VMs on physical machines with consideration of the network conditions between the physical machines and the data storage. In addition, our approach has also considered the scenario in which instable network condition changed the data access behaviors and deteriorated the application performance, and dealt with this scenario by migrating the VM to other physical machines. Our simulation on Cloudsim 2.0 proved that the proposed approach could improve the task completion time.

In terms of migration approach, our approach cannot guarantee the enforcement of SLA because the optimized data access might still go over the time requirement in the SLA. Here, a negotiation between the user and service provider should be implemented in the future work. In addition, the congestion might occur due to the fact that there is no priority setting for users. Therefore, it is possible that a host is perfect for several requests simultaneously but the system would not be able to schedule all of the tasks in the cloud. This may lead to a possibility that some users' tasks always occupy a faster link on the network. To solve this issue, it is necessary to extend a payment mechanism so that the system can set priority to the users according to their payment.

Acknowledgment

This research is partly supported by a University of Wollongong URC Near Miss Grant 2010.

References

[1] R. Buyya, http://www.buyya.com/gridbus/cloudsim/, access at 20/5/2010.

[2] R. Buyya, R. Ranjan, and R. N. Calheiros, Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. High Performance Computing & Simulation, 2009. HPCS '09. International Conference on.

[3] S. K. Bose, and S. Sundarrajan, Optimizing Migration of Virtual Machines across Data-Centers. Parallel Processing Workshops, 2009. ICPPW '09. International Conference on.

[4] W. Guohui, and T. S. E. NG, The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. INFOCOM, 2010 Proceedings IEEE.

[5] T. Hirofuchi, H. Ogawa, H. Nakada, S. Itoh, and S. Sekiguchi. (2008) A Storage Access Mechanism for Wide-Area Live Migration of Virtual Machines. In Summer United Workshops on Parallel, Distributed and Cooperative Processing, pages 19–24, 2008.

[6] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi. Dynamic Placement for Clustered Web Applications. Proceedings of the 15th international conference on World Wide Web, 595 – 604, Edinburgh, Scotland, (May) 2006.

[7] J-L. Kim, and R. D. Ellis. A Framework for Integration Model of Resource-Constrained Scheduling using Genetic Algorithms. Simulation Conference, Proceedings of the Winter, 2119 – 2126, Orlando, Florida, USA, (December) 2005.

[8] Y. Luo, B. Zhang, X. Wang, Z. Wang, Y. Sun, and H. Chen, Live and incremental whole-system migration of virtual machine using block-bitmap. In IEEE International Conference on Cluster Computing, 2008.

[9] T. Maoz, A. Barak, and L. Amar, Combining Virtual Machine migration with process migration for HPC on multi-clusters and Grids. Cluster Computing, 2008 IEEE International Conference on.

[10] K. Sato, H. Sato, and S. Matsuoka. A Model-Based Algorithm for Optimizing I/O Intensive Applications in Clouds Using VM-Based Migration. Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on.

[11] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici. A Scalable Application Placement Controller for Enterprise Data Centers. Proceedings of the 16th international conference on World Wide Web, 331 – 340, Banff, Alberta, Canada, (May) 2007.

[12] H. N. Van, F. D. Tran, and J-M, Menaud. Autonomic Virtual Resource Management for Service Hosting Platforms. Software Engineering Challenges of Cloud Computing, CLOUD '09. ICSE Workshop on, Vancouver, Canada, 1-8, (May) 2009

[13] H. Wei, I. L. Yeb and B. Thuraisingham, Dynamic Service and Data Migration in the Clouds. Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International.