Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 23 (2011) 695 – 703

Procedia Engineering

www.elsevier.com/locate/procedia

PEEA 2011

A Threshold-based Dynamic Resource Allocation Scheme for Cloud Computing

Weiwei Lin[a,*], James Z. Wang[b], Chen Liang[c], Deyu Qi[a]

[a]*School of Computer Engineering and Science, South China University of Technology, Guangzhou, China*
[b]*School of Computing, Clemson University, Box 340974, Clemson, SC 29634-0974,*
[c]*USA Department of Computer Science, Brown University, Providence, RI, USA*

Abstract

Compared to traditional distributed computing paradigms, a major advantage of cloud computing is the ability to provide more reliable, affordable, flexible resources for the applications (or users). The need to manage the applications in cloud computing creates the challenge of on-demand resource provisioning and allocation in response to dynamically changing workloads. Currently most of these existing methods focused on the optimization of allocating physical resources to their associated virtual resources and migrating virtual machines to achieve load balance and increase resource utilization. Unfortunately, these methods require the suspension of the cloud computing applications due to the mandatory shutdown of the associated virtual machines. In this paper, we study the resource allocation at the application level, instead of studying how to map the physical resources to virtual resources for better resource utilization in cloud computing environment. We propose a threshold-based dynamic resource allocation scheme for cloud computing that dynamically allocate the virtual resources (virtual machines) among the cloud computing applications based on their load changes (instead of allocating resources needed to meet peak demands) and can use the threshold method to optimize the decision of resource reallocation. The proposed threshold-based dynamic resource allocation scheme is implemented by using CloudSim, and experimental results show the proposed scheme can improve resource utilization and reduce the user usage cost.

Keywords: cloud computing; dynamic resource allocation; virtual resource; scheme

1. Introduction

Resource allocation in traditional IT infrastructure often assigns fixed computing resources to a particular application to satisfy its peak load requirement. Such peak-load-based static resource allocation schemes often result in the underutilization of computing resources. On the other hand, average-load-

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2011.11.2568

based static resource allocation schemes assign computing resources to applications based on their average workload, sometimes, failing to satisfy their peak load requests. The emergence of cloud computing [1, 2] offers the flexibility of managing the computing resources in a more dynamic manner because it uses the virtualization technology to abstract, encapsulate, and partition computing resources. However, this new computing paradigm has also raised new challenges to efficient resource allocation.

Recently, quite a few schemes [2-8] have been proposed to address the resource allocation problem in cloud computing. Rajkumar Buyya et al. [3] proposed a market-oriented resource allocation scheme that integrated both customer-driven service management and computational risk management to sustain service level agreement (SLA) oriented resource allocation. However this scheme requires a market-maker to bring service providers and consumers together, and a market registry for publishing and discovering cloud service providers and their services. Another market-based resource allocation strategy, RAS-M, was proposed by You et al. [4]. This scheme is based on market economy theory, where the problem of resource allocation is transformed into finding the equilibrium price vector and corresponding equilibrium solution, and a GA-based price adjusted algorithm is introduced to deal with the problem. But RAS-M scheme is proposed for resource allocation at the physical level of the cloud computing, and it only manages the CPU resource. Jean-Marc Menaud et al. [5, 6] proposed an autonomic resource manager to control the virtualized environment which decouples the provisioning of resources from the dynamic placement of virtual machines. This manager aims to optimize a global utility function which integrates both the degree of SLA fulfillment and the operating costs. They resorted to a Constraint Programming approach to formulate and solve the optimization problem. In [7], Fabien Hermenier et al. proposed a new approach, Entropy, in a homogeneous cluster environment, which takes into account both the problem of allocating the virtual machines (VMs) to available nodes (physical hosts) and the problem of how to migrate the VMs to these nodes. The performance overhead is determined by the time required to choose a new configuration and the time required to migrate VMs according to the configuration. The Entropy resource manager can choose migrations that can be implemented efficiently, incurring a low performance overhead. Wei et al. [8] used game theory to handle the resource allocation in cloud computing. In their approach, a Binary Integer Programming method is proposed to solve the parallel tasks allocation problem on unrelated machines connected across the Internet. Their algorithms take both optimization and fairness into account and provide a relatively good compromise resource allocation. But these methods can only be used for seeking optimal allocation solution for the complex and dynamic problems that can be divided into multiple cooperative subtasks.

Nonetheless, most of these existing methods focused on the optimization of allocating physical resources to their associated virtual resources and migrating virtual machines to achieve load balance and increase resource utilization. Unfortunately, these methods require the suspension of the cloud computing applications due to the mandatory shutdown of the associated virtual machines. To address this problem, a different approach to resource allocation in cloud computing environment has become more attractive. This new approach uses virtual machine as the minimum resource allocation unit. When a user starts an application, a virtual machine that satisfies the minimum resource requirement for the application is allocated. When workload of the application increases, a new virtual machine is allocated for this application. Unlike the existing resource allocation schemes, which allocate more physical resources (CPU, Memory, etc.) to the exiting virtual machine, this new approach does not need to shutdown the existing virtual machine for resource reallocation. This new approach is more practical [14] because most of the mission critical applications, such as online Web services, cannot afford any downtime. Therefore, this paper focuses on studying the resource allocation at the application level, instead of studying how to map the physical resources to virtual resources for better resource utilization in cloud computing environment. The goal is to dynamically allocate the virtual resources among the cloud computing applications based on their load changes to improve resource utilization and reduce the user usage cost.

The rest of the paper is organized as follows. In Section 2, we propose the threshold-based dynamic resource allocation scheme for cloud computing and discuss its principle and implementation considerations. Then, in Section 3, we discuss how to implement the proposed threshold-based dynamic resource allocation scheme under CloudSim environment. In section 4, we conduct performance studies using simulation, and analyze the performance of the proposed resource allocation scheme under different resource usage sceneries. Finally, we give our concluding remarks and discuss future studies in section 5.

2. Threshold-based Dynamic Resource Allocation Scheme

2.1. The principle of the threshold-based dynamic resource allocation

In general, the workload of network applications (such as Web applications) fluctuates over the application lifetime. If we use a static resource allocation scheme to assign fixed resources to an application, the application may be slowed down sometimes due to insufficient resources, or excessive resources are wasted when application is not at its peak load. Therefore, it is ideal to design a dynamic resource allocation scheme that can adjust the resources allocated to an application according to its workload, thus, improving the resource utilization.

The main idea of the proposed threshold-based dynamic resource allocation scheme is to monitor and predict the resource needs of the cloud applications and adjust the virtual resources based on application's actual needs. A dynamic resource allocation scheme needs to address two issues: when to reallocate resources and how much resource to be adjusted.

2.2. When to reallocate cloud resources

Since reallocating virtual resources for cloud applications cost computing time and physical resources, over-frequent allocation of resources may reduce the efficiency of cloud applications. Therefore, when to reallocate the virtual resources becomes critical to the efficiency of the resource allocation scheme. If the time interval between two resource re-allocation events is too large, the system may not be able to respond the load changes timely. Such a slow response may either affect the performance of certain applications or waste virtual resources. On the other hand, if the time interval between two allocation events is too short, the overhead for resource allocation is too much.

In the proposed dynamic resource allocation scheme, the interval between two consecutive allocation events is set to be adaptive to the load change of a cloud application. If the load of an application changes slowly in a steady pace, a longer interval is selected. If the load of an application changes rapidly, a shorter interval is used. However, sometimes, the load of an application oscillates during the application's lifetime. If we schedule merely based on current workload, the system may have to frequently reallocate the virtual resources, resulting extra overhead. To avoid the unnecessary overhead caused by the application's workload oscillation, a threshold is used to regulate the timing of resource reallocation.

Assume the maximum workload of a virtual machine (virtual resource) is $L_{\max}$, we define its normal workload L_{norm} as: $L_{norm} = L_{\max} \times \text{rate}_{norm}$, where $\text{rate}_{\text{norm}} \in [0,1]$ is called the normal workload rate and this value is predefined by the system administrator according to application's workloads. In general, we assign physical resources to virtual machines so that applications on these virtual machines are around their normal workloads. In essential, the system reserves some virtual resources to anticipate the sudden increase of the application's load (a similar approach is used in many of the Web applications). We further define: $L_{threshold} = \text{rate}_{\text{threshold}} \times (1 - rate_{norm}) \times K \times L_{\max}$ (1),

where K is the current number of virtual machines, $\text{rate}_{\text{threshold}}$ is a threshold rate ranging between 0 to 1. A resource reallocation takes place only if the application's load changes (up or down) is over $L_{threshold}$.

2.3. Threshold-based dynamic resource allocation scheme

The proposed threshold-based dynamic resource allocation scheme consists of two procedures, Datacenter and Broker. The broker procedure runs on user's machine with the application. The datacenter procedure, which works as the manager of the cloud computing resources, runs on the datacenter's central computer. These two procedures interact with each other to dynamically manage the virtual resources for cloud applications.

In this scheme, the Datacenter procedure, which manages the physical resources such as CPU and RAM, waits for requests from brokers, and provides extra virtual resources (VMs) or revoke excessive virtual resources (VMs) based on the requests from brokers. The broker procedure determines whether an application needs more virtual resources or excessive virtual resources owned by an application need to be revoked based on application's load changes. The pseudo code of these two procedures is depicted as follows:

```
Datacenter:
while the system is running do
   wait for brokers to submit requests
   if broker requests virtual resource then
     allocate virtual resource to broker;
     start the application on new virtual resource;
   end if
   if broker releases virtual resource then
     revoke the virtual resource released by a broker
   end if
     return the operation result;
end while
```

```
Broker:
while the system is running do
   wait for the next reallocation cycle;
   initBroker();
   calThreshold();//calculate Threshold
   if currentLoad() > normLoad()+Threshold then
     calculate the amount of virtual resources needed;
        send a request to datacenter for more virtual resources;
        wait for response from datacenter;
        add new virtual resources to the local resource list;
   end if
   if currentLoad()<normLoad()-Threshold then
     calculate the amount of virtual resources to be released;
     send a request to datacenter to revoke virtual resources;
     remove the released virtual resources from the local resource list;
   end if
end while
```

Fig. 1. (a) the pseudo code of Datacenter; (b) the pseudo code of Broker

3. Performance Study

3.1. The simulation model for performance evaluation

To evaluate the efficiency of our proposed dynamic resource allocation scheme, we simulate various resource allocation sceneries under CloudSim [9] environment. We model a cloud application as a set of tasks, which have the same completion deadline, require the same amount of memory, CPU, and bandwidth, etc. The number of tasks represents the workload of the cloud application. We further model the load variation of a cloud application by varying the user task request intervals. Without losing the generality, we assume all virtual machines have the same workload capacity.

3.2. The threshold-based dynamic resource allocation scheme in CloudSim environment

In CloudSim environment, we need to implement two essential aspects of the threshold-based dynamic resource allocation scheme. First, we need to simulate the interaction between the datacenter and brokers. We also need to simulate the load variation of the cloud application by varying the user resource request

and release intervals. To achieve these two tasks, we implement two simulation classes by extending the original Datacenter class and the Datacenterbroker class in the CloudSim Toolkit. We discuss the extensions we made to the original Cloudsim classes here.

1)Extended Datacenter Class

The extended datacenter class can be presented as:

```
public class myDatacenter extends Datacenter {
    private LinkedList<myVm> vmlist; ...
    protected void myprocessVMCreate(){... }
    public VirtualMachineList askForVMs(){... }
    protected void releaseVMs(int[] temp){... }
    …
}
```

Compared to the original datacenter class in Cloudsim Toolkit, we added a member variable vmlist to hold the allocated virtual machines. We implemented a method **createVMs()** to initialize the vmlist based on the parameters for describing the virtual machines, and to allocate physical resources to these virtual machines. We also added a method **askForVMs()**, which is invoked when the Datacenter receives requests from brokers for more virtual machines. We note here that the virtual machines maintained in Datacenter's vmlist contain only the descriptions of virtual machines, such as the amount of memory, CPU, and bandwidth needed to create the virtual machines. However, like in the original Cloudsim Toolkit, Datacenter does not allocate physical resources to a virtual machine until broker has requested the virtual machine. Finally, we added a method **releaseVMs()** for the datacenter to release the virtual machine's physical resources when a broker releases a virtual machine.

2)Extended DatacenterBroker

The extended DatacenterBroker class can be depicted as:

```
public class myDatacenterBroker extends DatacenterBroker{
   protected int calVMs(){...}
   protected void bindToVMs(){...}
   void resourceRequest(){…}
}
```

A broker is responsible for determining when to reallocate resources for an application and how much resource to be allocated. Three methods are added to the original DatacenterBroker class of the CloudSim ToolKit. Method **calVMs()** is used to determine how many virtual machines is needed for an application. It uses the capability of virtual machine, the number of tasks, and the deadline of tasks to calculate the norm workload, and then the number of virtual machines needed is determined. After determining the number of needed virtual machines using **calVMs()**, a broker uses method **resourceRequest()** to request or release virtual machines. Method **bindToVMs()** is added to bind a task to a specific virtual machine.

3.3. The simulation model for performance evaluation

We compare our threshold-based dynamic resource allocation scheme with a peak-load-based static resource allocation scheme in terms of the resource utilization, total resource cost, and overhead of the resource allocation. The resource utilization is defined as the number of tasks finished by all virtual machines divided by the capacity of all virtual machines. The total resource cost (TRC) is defined as

$$TRC = (size_{\text{mem}} \times price_{mem} + size_{\text{sto}} \times price_{sto}) \times Num_{res} \quad (2)$$

where $size_{\text{mem}}$ is memory size of virtual machine, $size_{\text{sto}}$ is storage size of virtual machine, $price_{mem}$ is the unit memory price, $price_{sto}$ is the unit storage price and Num_{res} is the number of the virtual machines.

Numerous studies [10,11,12] have shown that workloads of Internet applications can be highly dynamic with variations at multiple timescales. One popular workload variation pattern is oscillation, in which application workload varies in a small range around a certain load level. Such a workload pattern can be found in applications like as online banking, ATM machines, etc. Another popular workload variation pattern is Gaussian distribution, in which user requests increase when the application gains popularity and user interests dry down when the time goes by. This workload pattern happens in applications like movie on demand. In this paper, we simulate the proposed dynamic resource allocation scheme under these two workload sceneries and study its performance under different system conditions.

3.3.1 Experiment under oscillating workloads

In this experiment, we assume that the workload of cloud application follows the pattern depicted in Figure 2. Initial workload is at 100 tasks. The number of tasks increases to 120 and drops back periodically. Occasionally, we see a workload spike and then the load drops back to the oscillating pattern after that. In one of the points of time, load dramatically increases and then goes back down and begin to oscillate again. Aiming at the oscillations in cloud application, and to avoid the extra cost created by the oscillations, we apply our threshold-based dynamic resource allocation scheme and set different thresholds for comparison.

Some parameters in this experiment are: the deadline of tasks is 150s, the capability of CPU provided by datacenter is 16000 mips, 40 available virtual machines are created, the capability of every virtual machine is 400mips. Assume that it takes 15s for a virtual machine to process a task. Thus the maximum workload ($L_{\max}$) of single virtual machine is 10 (10=150s/15s).

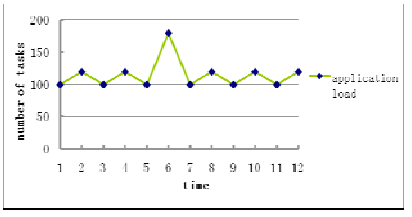

Fig.2. cloud application under oscillating workloads

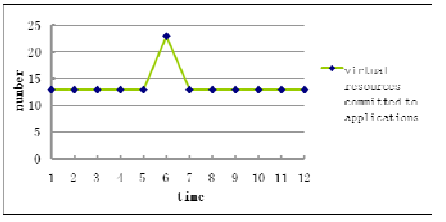

Fig.3 virtual resources committed to cloud application with threshold rate 0.95

Assume the norm workload rate is 0.8 and the threshold rate is 0.95. At the first point of time, the workload is at 100 tasks and L_{norm} =0.8× $L_{\max}$ =8. Then we can get the number of virtual resources committed to cloud application. At the time of 2, we can get $L_{threshold}$ =24.7 based on Eq. (1). The application's workload change (from 100 to 120) is under $L_{threshold}$, so the number of virtual resources committed to cloud application is still 13(does not change). At the time of 6, the application's workload change (from 100 to 180) is over $L_{threshold}$, so a resource reallocation takes place and the number of virtual resources committed to cloud application is 23 (23=⌈180/8⌉). Therefore, we get the numbers of virtual resources committed to cloud application at all points of time, which are shown in Figure 3.

Assume the norm workload rate is 0.8 and the threshold rate is 0.50. Then we calculate the numbers of virtual resources committed to cloud application based on Eq. (1), which are shown in Figure 4. We can

see that the number of virtual resources reallocated with threshold rate 0.95 is different from the case with threshold rate 0.50.

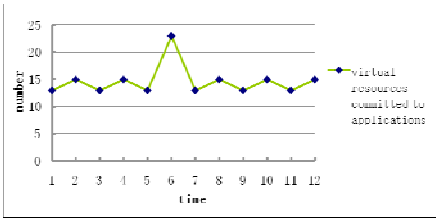

Fig.4 virtual resources committed to cloud application with threshold rate 0.50

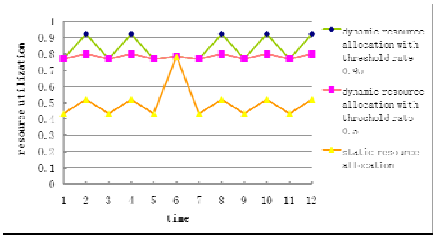

Fig.5 Resource utilization of three resource allocation algorithms

If static resource allocation is applied, virtual resources committed to cloud application are required to reach 23 (for L_{norm} =8, we can get ⌈180/8⌉=23) since the peak load is 180 tasks. Based on the definition of resource utilization, the resource utilizations of three allocations are shown in Figure 5.

Assume the price of memory is 0.05/MB and the price of storage is 0.001/MB. For every virtual machine in experiment, the size of memory is 512MB; the size of storage is 10000MB. We calculate the resource cost of these 3 allocations based on Eq. (2). The resource cost of the dynamic allocation with threshold rate 0.95 is 5909.6, the resource cost of the dynamic allocation whose threshold rate is 0.5 is 6265.6 and the resource cost of the static allocation is 9825.6, as shown in Figure 6.

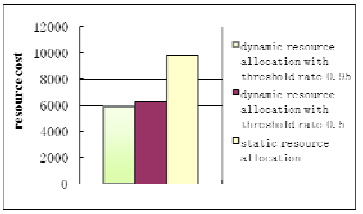

Fig.6 Resource cost of three allocations

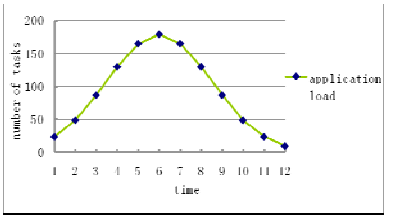

Fig.7 Cloud application when workload follows a Gaussion distribution

We can conclude from the experimental results: (1)The dynamic algorithm whose threshold rate is 0.95 is better than the one whose threshold rate is 0.50 in this situation, and both are better than the static algorithm. (2)The bigger the threshold rate is, the less resource is reserved. This means there could be more resources available for increasing load. And then the utilization is improved. The increase of resource utilization also reduces the cost associated with the request and exchange of virtual resources. However, it makes it less sensitive to the change of load. The reverse situation would happen if a smaller threshold rate were applied.

3.3.2 Experiment when workload follows a Gaussion distribution

Gaussian distribution was selected because is one of the better distributions to characterize user's reflection time[13]. We assume that the workload of cloud application follows the pattern depicted in Figure 7. Workload changes closely approximately follow Gaussian distribution.

Some parameters in this experiment are: the deadline of tasks is 150s, 40 available virtual machines are created; the capability of every virtual machine is 400 MIPS. Assume that it takes 15s for a virtual machine to process a task. Thus the capability of single virtual machine is 10. Assume the norm workload

rate is 0.8 and the threshold rate is 0.75. We can calculate the numbers of virtual resources committed to cloud application based on Eq. (1), which are shown in Figure 8.

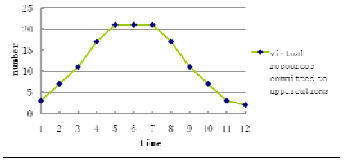

Fig.8 virtual resources committed to cloud application with threshold rate 0.75

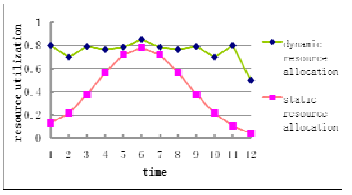

Fig.9 Resource utilization of two allocation schemes

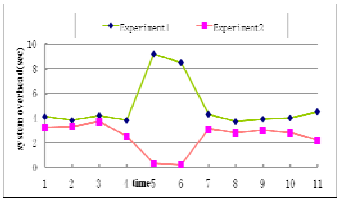

Fig.10 Overhead of resource allocation in two experiments.

If static resource allocation is applied, virtual resources committed to cloud application are required to reach 23 (for L_{norm} =8, we can get $\lceil 180/8 \rceil = 23$) since the peak workload of cloud application is 179 tasks. Based on the definition of resource utilization, the resource utilizations of two allocation schemes are shown in Figure 9.

We also calculated the resource cost of these two resource allocation schemes. The resource cost of the dynamic scheme is 5019.6 and the resource cost of the static scheme is 9825.6.

We can draw conclusions from the comparison:(1) The resource utilization of dynamic allocation is better than it of static allocation; (2) The resource cost of static allocation is obviously higher than it of dynamic allocation, that's because the static allocation scheme uses some certain amount of resource, thus a large part of resource may be wasted.

3.4. Overhead of Threshold-based dynamic resource allocation

In order to study the overhead of Threshold-based dynamic resource allocation, we define its overhead T as: $T = T_s + T_d + T_r$, where T_s is the overhead of virtual resource startup, T_d is the overhead of virtual resource downtime and T_r is the overhead of the resource reallocation. The experimental results for the overhead of Threshold-based dynamic resource allocation in the above experiments are shown in Figure 10. We can conclude from the experimental results: (1) The overheads of dynamic resource allocation in two experiments are low (about a few seconds); (2) At the time of 5 and 6, the overhead of dynamic resource allocation in Experiment 1 is almost 0 because it does not require resource reallocation, however, the overhead of dynamic resource allocation in Experiment 2 is relatively large because it need reallocate multiple resources (virtual machines); (3) As the overheads of the resource reallocation procedure are very low, the overhead of Threshold-based dynamic resource allocation focuses on the control overhead of virtual resource.

4. Conclusions

Resource allocation in traditional IT infrastructure often assigns fixed computing resources to a particular application to satisfy its peak load requirement. Such peak-load-based static resource allocation schemes often result in the underutilization of computing resources. To address this problem, we propose a new approach that uses virtual machine as the minimum resource allocation unit and the threshold-based dynamic resource allocation scheme for cloud computing that monitor and predict the resource needs of the cloud applications and adjust the virtual resources based on application's actual needs. The scheme can dynamically reconfigure the virtual resources for cloud applications according to the load

changes in cloud applications, so it can save resources and increase resource utilization. The experimental results show that the proposed dynamic resource allocation scheme can improve resource utilization and reduce the user usage cost. Moreover, the proposed scheme has a simple implementation and, unlike many other approaches, it can avoid the complexity of re-allocation of physical resources.

Acknowledgements

This work is financially supported by the Fundamental Research Funds for the Central Universities, SCUT (No. 20092M0103), Comprehensive Strategic Cooperation of Guangdong Province and Chinese Academy (No. 2009B091300069), Guangdong Natural Science Foundation (x2jsB6100260) and the National Natural Science Foundation of China (No. 61070015). James Wang's work is partially supported by NSF grant DBI-0960586.

References

[1] Michael Armbrust, Armando Fox, Rean Griffith and et al. Above the Clouds: A Berkeley View of Cloud Computing [EB/OL], http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html, February 10, 2009.

[2] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic, Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, Future Generation Computer Systems, Volume 25, Number 6, Pages: 599-616, ISSN: 0167-739X, Elsevier Science, Amsterdam, The Netherlands, June 2009.

[3] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal, Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, Keynote Paper, Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications, Sept. 25-27, 2008, Dalian, China.

[4] Xindong You, Xianghua Xu, Jian Wan, Dongjin Yu, "RAS-M: Resource Allocation Strategy Based on Market Mechanism in Cloud Computing," chinagrid, pp.256-263, 2009 Fourth ChinaGrid Annual Conference, 2009.

[5] Hien Nguyen Van, Frédéric Dang Tran, Jean-Marc Menaud, "SLA-Aware Virtual Resource Management for Cloud Infrastructures," cit, vol. 1, pp.357-362, 2009 Ninth IEEE International Conference on Computer and Information Technology, 2009

[6] Hien Nguyen Van, Frederic Dang Tran, Jean-Marc Menaud, "Autonomic virtual resource management for service hosting platforms," icse-cloud, pp.1-8, 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, 2009

[7] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller and J. Lawall. Entropy: a Consolidation Manager for Cluster. In proc. of the 2009 International Conferenceon Virtual Execution Environments (VEE'09), Mar.2009.

[8] Guiyi Wei, Athanasios V. Vasilakos, Yao Zheng and Naixue Xiong. A game-theoretic method of fair resource allocation for cloud computing services. The Journal of Supercomputing,Volume 54, Number 2, 252-269.

[9] Rodrigo N. Calheiros, Rajiv Ranjan, Cesar A. F. De Rose, and Rajkumar Buyya, "CloudSim: A Novel Framework for modeling and Simulation of Cloud Computing Infrastuctures and Services", 2009.

[10] Rahul Singh, Upendra Sharma, Emmanuel Cecchet and Prashant Shenoy. Autonomic Mix-Aware Provisioning for Non-Stationary Data Center Workloads, Proceedings of the 7th IEEE International Conference on Autonomic Computing and Communications (ICAC), Washington, DC, USA, June 7-11, 2010.

[11] J. Hellerstein, F. Zhang, and P. Shahabuddin. An Approach to Predictive Detection for Service Management. In Proceedings of the IEEE Intl. Conf. on Systems and Network Management, 1999.

[12] D. Menasce and F. Ribeiro. In search of invariants for e-business workloads. In Proceedings of the 2nd ACM conference on Electronic Commerce, pages 56–65,2000.

[13] Jakarta Apache. JMeter. http://jakarta.apache.org/jmeter/index.html,2010.12.

[14] H. Lim, S. Babu, J. Chase, and S. Parekh. Automated Control in Cloud Computing: Challenges and Opportunities,In Proc. of the First Workshop on Automated Control for Datacenters and Clouds, June 2009:13-18.