Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070551

Multiple Virtual Machines Resource Scheduling for Cloud Computing

Weizhe Zhang[*], *Hui He, Gui Chen and Jilong Sun*

School of Computer Science and Technology, Harbin Institute of Technology, 150001 Harbin, China

Received: 13 Feb. 2013, Revised: 15 Jun. 2013, Accepted: 18 Jun. 2013
Published online: 1 Sep. 2013

Abstract: Cloud computing emerges as a new computing paradigm concerned by both academia and industry. Resource management of multiple virtual machines is the core of Infrastructure as a Service. Focusing on the CPU resources, the purpose of this paper is to increase the QoS of web service by properly scheduling the CPU resource across the virtual machines. We formulate the CPU scheduling of multiple virtual machines into an integer programming problem. Then, a global regulation algorithm based on utility optimization theory is proposed. Experimental result s show that the regulation system of CPUs can significantly improve the performance of Web applications.

Keywords: Cloud Computing, Virtual Machine, Resource Management, Task Scheduling, Utility Optimization

1 Introduction

In recent years, cloud computing as an innovative computing model has attracted increasing attention by the academia and industry. Cloud computing usually builds many large-scale data centers to provide the on-demand resource-leased services for the users [1], which is different from the traditional PC-centric computing model. According to different service levels, current cloud computing service models can be divided into the following three categories [2]: (1) Infrastructure as a Service (IaaS): Outsourcing the device to support the operation, users run the operating system and application software programs according to their own will; (2) Platform as a Service (PaaS):Using the network to provide operating system and related services, the users utilize the programming languages and tools supported by the provider to prepare the services; (3) Software as a Service (SaaS): the applications are hosted by the service providers, the users make use of these services through all kinds of client devices and thin-client user interfaces.

System virtualization technology is the core of Infrastructure as a Service (IaaS). Virtualizing the computing and storage resources of one or more data centers to form a highly efficient and flexible resource pool, can help reduce the cost of infrastructures, postpone the time to expand the data center, and improve the ability to deal with the rapidly changing business needs [3]. Virtualization technology includes the virtualization of memory, CPU, IO devices. Time-sharing of memory and CPU resources in multiple virtual machines is not only becoming the bottleneck of system virtualization, but also the research focus of the resource management of the current virtual machines.

When several virtual machines are deployed in one physical host, it is necessary to bind the virtual CPUs to the physical CPUs and allocate the physical memory among the virtual machines. If the resources are allocated statically, they taken up by the virtual machines do not change during the running process. The number of virtual machines that could be executed concurrently in a physical host would be limited by the physical CPUs and the size of memory. Moreover, up-level services running on different virtual machines have different and dynamically changing requirement on CPU and memory. The static allocation would with no doubt lead to unreasonable resource allocation. This would affect the execution efficiency. Current research is in lack of a system architecture which could collaboratively manage the memory of virtual machines from a global perspective, and problems about how would the system allocate the virtual CPUs and memory for the virtual machines and the recovered memory should be allocated to which virtual machine as a priority. These are still open

[*] Corresponding author e-mail: wzzhang@hit.edu.cn

question.

In response to the problems above, in section 2, the related works are introduced. Then, we formulate the problem of CPU scheduling of multiple virtual machines by constructing a utility function of multiple virtual machines, and reduce it to an integer programming problem in section 3.We put forward the architecture of multiple virtual m achines CPU management in section 4, To solve this problem, a CPU global regulation algorithm based on the utility optimization theory comes up in section 5. The experimental results and the analysis in this paper are also presented in section 6. Finally, our work of this paper is summarized in the last section.

Table 1: Symbols of CPU information in virtual machines

U_{global}	Global utility value
U_i	Local utility value
$\triangle U_i$	Incremental utility value
RT	Response time
R_i	Allocated CPU resource
R_{total}	CPU resource in total (100%)
R_{left}	Remainder of CPU in total
R_{inc}	Incremental granularity in CPU allocating
RT_m	Expected response time
H_i	Upper bound of CPU allocating
L_i	Lower bound of CPU allocating
W_i	Every domain's weight
V_i	The i^{th} virtual machine

2 Related Works

Researchers have done a great deal of works to study the CPU resource management in multiple virtual machines. Reference [4] studies the related algorithms to realize the Xen internal Virtual CPU scheduling and compares the performance indicators and scenarios of different algorithms such as BVT, SEDF, Credit and so on. With the help of these algorithms we can realize the dynamic configuration of CPU. Reference [5] weakens the impact of the communication-intensive applications in the way of distributing relatively more CPU resources to the Domains with high communications density in virtual machines. Reference [6, 11] introduces adaptive control theory which ensures the effective use of CPU regulation in virtual machines. But they ignore the problems caused by QoS. Reference [7] also adapts the knowledge of control theory while choosing Kalman filters' design ideas to realize the controller, which is more complex. Reference [8] uses the utility function as the criterion of distribution system and proposes a set of general ideas concerning the resource management framework and the resource flow in multiple virtual machines. However the control of resources can only apply to laboratory environment and the requirements of corresponding type is strict and has no portability. Reference [9] considers the coordination of multiple virtual machines in multi-layer web applications, which is very common recently. Reference [10] combines the control theory and knowledge base of regulations and comes out with a mixed model to solve the regulation problems of CPU in multiple virtual machines. Reference [12] designs a resource management framework with a variety of applications applied to multiple virtual machines.

The work mentioned previously proposed the underlying support mechanism of the CPU and memory resources distribution. However, they fail to solve the problems such as when and how to distribute the CPU resources among the different virtual machines. There's also a lack of the collaborative management system framework and the dispatch strategies of multiple virtual machine and memory from global perspectives. This paper will study this further and bring out new effective solution.

3 Problem of CPU Regulation in Multiple Virtual Machines

The virtual CPU is called VCPU in the Xen virtual machines and every VCPU can be mapped to more than one physical CPUs. In order to improve the performance of the system, the projectors of the system bind VCPUs to one or more than one physical CPUs. But through the observation of all the physical cores, we find out that most of the cores' utilization rate is low because of the load's uncertainty. Although the Xen provides the Credit mechanism which supports the CPU to regulate dynamic, it lacks of the strategy of CPU coordinating regulation oriented to the multiple virtual machines.

Definition 1.*(CPU Scheduling Problem of Multiple Virtual Machines). The problem of Multiple Virtual Machines' CPU regulation can be concluded to following restraint optimizing problem, assuming the number of virtual machines is n.*

$$maxU_{global} = \sum W_i \times U_i(R_i)$$

$$st.\begin{cases} \sum R_i \le R_{total} \\ L_i \le R_i \le H_i (i = 1.2...n) \\ R_i = h \times R_{inc} \end{cases}$$

The goal of the restraint problem is to confirm a group of CPU mapping project $R = \{R_1, R_2 ... R_n\}$ and then get the largest U_{global}. The restraint conditions require the gross of the allocated CPU's $\sum R_i$ not to exceed the gross of the resource R_{total}. Each CPU's resource has an upper bound and a lower bound$(L_i \le R_i \le H_i)$, and CPU must be allocated discretely. Dividing with fine granularity is not only bad to the ensemble programming, but also has no real influence on improving the performance. So set the allocating granularity as R_{inc} and its range as $h \times R_{inc}$. The optimizing problem is an integer restraint optimizing problem.

4 Architecture of CPU Regulation in Multiple Virtual Machines

The architecture of CPU regulation in multiple virtual machines is shown in Figure 1. It mainly consists of four parts: the monitor, the regulator, the local effectiveness of control and the global effectiveness of control. The regulator is inside the Xen virtual machines' monitor. The monitor, local effectiveness of control and the Web server are located in DomainU's interior. The global effectiveness of control is made up of the global utility generating module and global optimizing and solving module, realized in the Domain0. Monitor: The main role of monitor is to detect a Web server QoS metrics, in the text it means the average response time of Web. In the experimental environment Apache module was developed for the Apache server to record real-time server response time. In addition, we monitor the changes of the system resources. It is mainly to do some tests for proc CPUinfo, and obtain the condition of the system CPU, etc.

In addition, it monitors the workloads of Apache through the Apache logs module.

Regulator: regulator is mainly used to actually regulate the tasks, it based on the Credit Algorithm distributes the allocation values result from the global calculation to the Xen, and regulate the time slots size allocated for each virtual machine, and hence completed the task of dynamically regulating CPU.

Local utility controller: The main role of local effectiveness of control is to accept the data collected by the monitor, using the recursively least squares method, generating the relation of resources and QoS in real time. Then we pass this relationship as a local utility function to the global effectiveness of control for generating global utility function.

Global utility controller: Global effectiveness of control mainly consists of two modules: the global optimal function generation module and the global optimal solution solving module. The global optimal function generation module is mainly to collect the local utility function generated by the local effectiveness of control and generate global utility function with SLA information. Global optimal solution solving module is mainly according to a certain method to solve the optimal allocation of resources and distribute to the regulator.

5 Utility-based Global Optimization Theory of the CPU Regulation Algorithm

Web applications are required to develop a Service Level Agreement (SLA) with the manager when submitting the task, in order to ensure the quality of service. The primary role of local effectiveness of control is to link the resources with the SLA, and gives a real value to indicate their utility; the local utility functions will be passed to the global utility function to build the global optimal allocation.

Utility function consists of two parts: the part of the income *Reward* and the part of the cost *Cost*. For each virtual machine, the utility value is the difference between them. Suppose there are n Domains, for each Domain of the corresponding benefits and costs part were $Reward_i$ and $Cost_i$, the utility value of $U_i, i \in \{1,2,...n\}$ is the difference between:

$$U_i = Reward_i - Cost_i$$

Defy the income part as:

$$Reward_i = \begin{cases} \dfrac{1}{RT - RT_M + 1}; RT \geq RT_m \\ 1 \qquad\qquad ; RT < RT_m \end{cases}$$

Defy the cost part as:

$$Cost_i = 1(RT_i + 1)$$

The relationship between effectiveness function and resources is based on the relationship of the response time and CPU resources:

$$U_i = \begin{cases} \dfrac{1}{\frac{1}{\alpha R+\beta} - RT_m + 1} - kR; RT \geq RT_m \\ 1 - kR \qquad\qquad ; RT < RT_m \end{cases}$$

Local utility function U_i, generated by the local utility device, as the basic function, will be passed to the global utility effectiveness of control by each virtual machine. Global utility function is generated in the global utility effectiveness of control. How to solve this global utility function to get the optimal solution is the key to solve the problem. There are some conventional methods to solve the multi-variable optimization problem under the constraints, for example, the approximate planning method, penalty function method, and multiplier method. But the time complexity of these methods is too large, which cannot meet the requirements of real-time. Because of the optimization problem used to solve continuous function, the solution cannot get an integer solution.

In this paper, we propose an algorithm based on heuristic to find a near-optimal solution for this integer constrained problem optimization of multiple virtual machines' CPU regulation, shown as algorithm 1. The basic idea is as follow: For all virtual machines first we allocate CPU according to their lower bound and

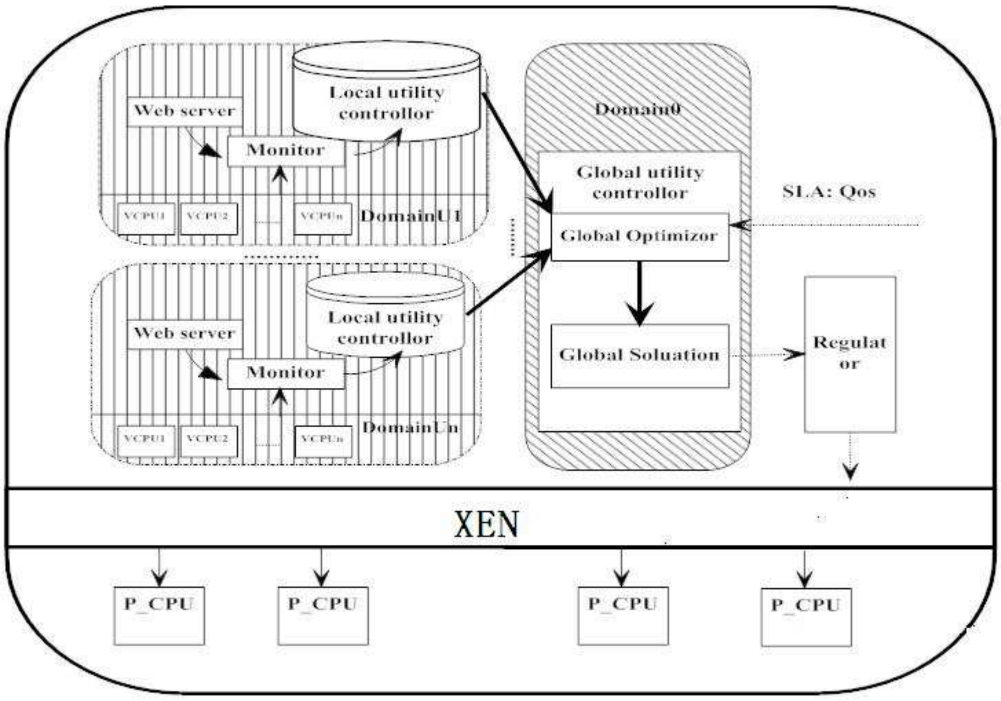


Fig. 1: CPU management framework for Multi-VMs

calculate the current utility value. Then we calculate the utility value when we allocate R_{inc} more resource. After that we calculate the incremental utility value. And then we find the virtual machine with the greatest change of utility value from all of the virtual machines and allocate R_{inc} to them. Then virtual machine's current utility value and the utility value after distribution are calculated in each allocation. We reorder them and find the group of virtual machines with the largest increase of utility, then allocate resources to them, so this part of the resources can get maximum utility. After all resources are allocated or the maximum effect of increasing ΔU is less than 0, then we stop the distribution, return the distribution program. From the algorithm, it can be seen that the solution is not necessarily to be the optimal solution, but an approximate optimal solution, the size of R_{inc} affects the degree of accuracy. It can be seen that the algorithm has linear time complexity $O(c*n)$, where n is the number of virtual machines, c depends on R_{inc}.

Algorithm 1. CPU Global Regulation Algorithm in Multiple Virtual

Input: $Cost_i$

Output: $R = \{R_1, R_2 ... R_n\}$

```
1.For any virtual machine $V_i (i \in 1,2...n)$
2.   Set lower limit $L_i$ to be the default allocating value $R_i = L_i$;
3.   Get the utility value according to the local utility function $U_i = Reward_i - Cost_i$;
4.   Set $U_{iold} = U_i$ ;
5.End for
6.While $R_{total} != \varnothing$
7.   For any virtual machine $V_i (i \in 1,2...n)$
8.      Set allocated resource as $R_i + R_{inc}$, and get the utility value according to the local utility function, $U_i = Reward_i - Cost_i$;
9.      Calculate Incremental Utility Value $\Delta U_i = W_i (U_{iold} - U_i)$;
10.  End for
11.  Find a subset $S = V_i, V_k ... V_z$,that satisfying $\forall i \in \{1,2...n\}, V_f \in S$ has $\Delta U_f \geq \Delta U_i$;
12.  If $\Delta U_f \leq 0$ then
13.     Allocating finish;
14.     Return allocating plan $R = R_1, R_2 ... R_n$;
15.  End if
16.  For any virtual machine $V_i (i \in 1,2...n)$
17.     If $R_{total} - R_{inc} \leq 0$ then
18         Allocating finish;
19         Return allocating plan $R = R_1, R_2 ... R_n$;
20.     End if
21.     Set $R_i = R_i + R_{inc}$
22.     $R_{total} = R_{total} - R_{inc}$;
23.  End for
24.End while
```

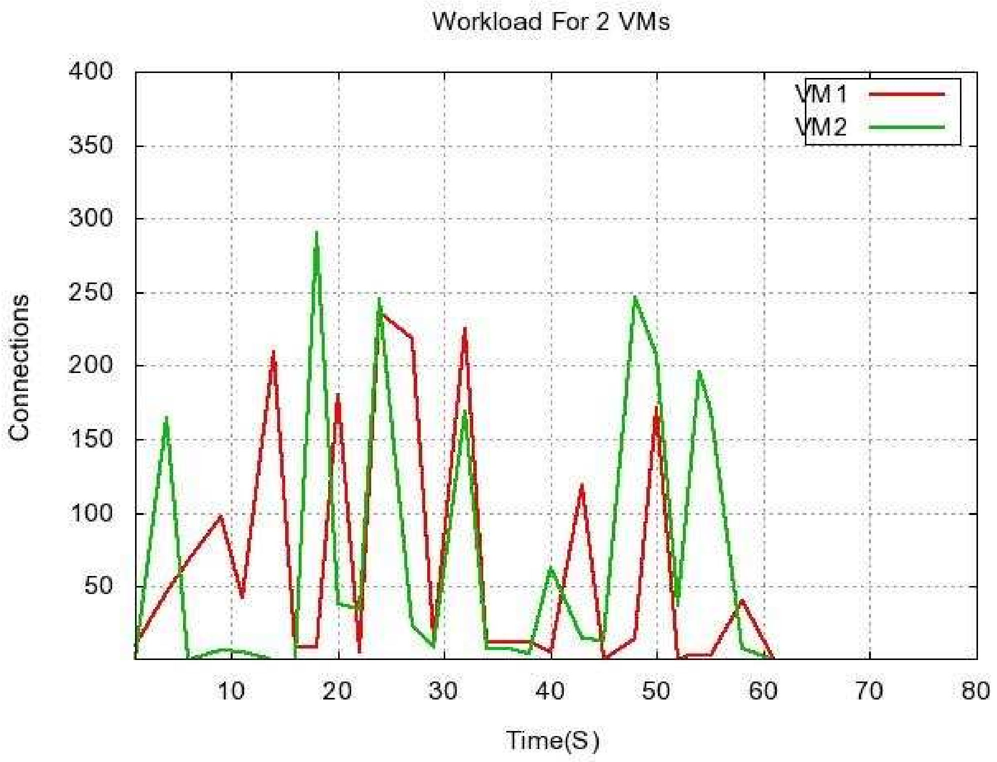


Fig. 2: The Multi-VM workloads produced by Httperf

6 Experiments

This experiment consists of two parts: a) examining the regulating strategy of multiple virtual machines' CPU; b) checking the arithmetic performance of local regulating and balance regulating respectively.

The software and hardware environment is Dawning Blade Sever, with 8-core CPUs (2 Intel Xeon E5506 2.13 GHz) and 16G 1333MHz RAM. The Virtual machine Xen is equipped with Edition 3.2 system using Web service as the application context. Open Source Apache server is adopted as the Web server in the text. Smarbits apparatus and Httperf served as the load lines, which are made by Spirent Communications for examining the performance of network devices. They can change load lines to examining the performance of the server or intermediate equipment, and the assorting software is Avalanche. Httperf, which is a text tool for open-source Web server and generates load with Httperf experimented with multiple machines. The frequency and amount of access can be regulated.

Interval of CPU regulation is 2s. In this regulating grading local effectiveness of control cannot sense the change of memory and computing cost by and large. While universal effectiveness of control has huge cost with about 3%-5% CPU occupancy rate. In this section regulating performance of main CPU is examined.

In order to examine the effectiveness of CPU regulation, the server turns on two virtual machines: VM1 and VM2. Response time in two scenes, which are static distribution (50% for each virtual machine) and dynamic regulation, is compared.

In the experiment, we set the system QoS as 10ms, and the configuration of the virtual machine is the same the experiment above. We send testing workloads (HTTP request) to two virtual machines at the same time from outside with Httperf testing collection, Fig. 2 is the workloads in the test, it shows that some of the workload peaks of the two virtual machines overlapped and staggered from time to time, which could simulate the situation of the resource competition of multiple virtual machine. Figure 3 shows CPU allocation of the two virtual machine when CPU regulation is started, system would allocate more CPU resource to the one of VM1 and VM2 which has larger workloads., while resource would sacrifice one of the two virtual machine when the workloads competition appear between them so that the situation of resource competition can be quickly solved.

Fig. 4 is the contrast of the response time accumulated probability of multiple virtual machines when regulated dynamically and allocated statically. When the CPU is allocated statically, the delay is quite large in most of the

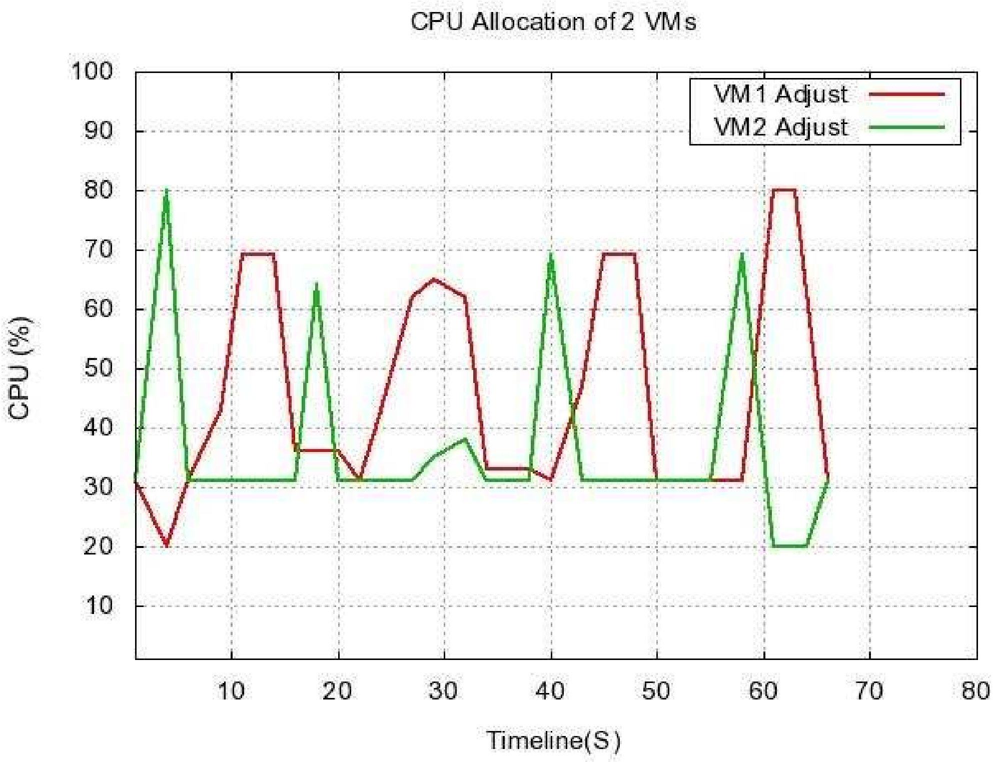


Fig. 3: The CPU allocation of Multi-VMs

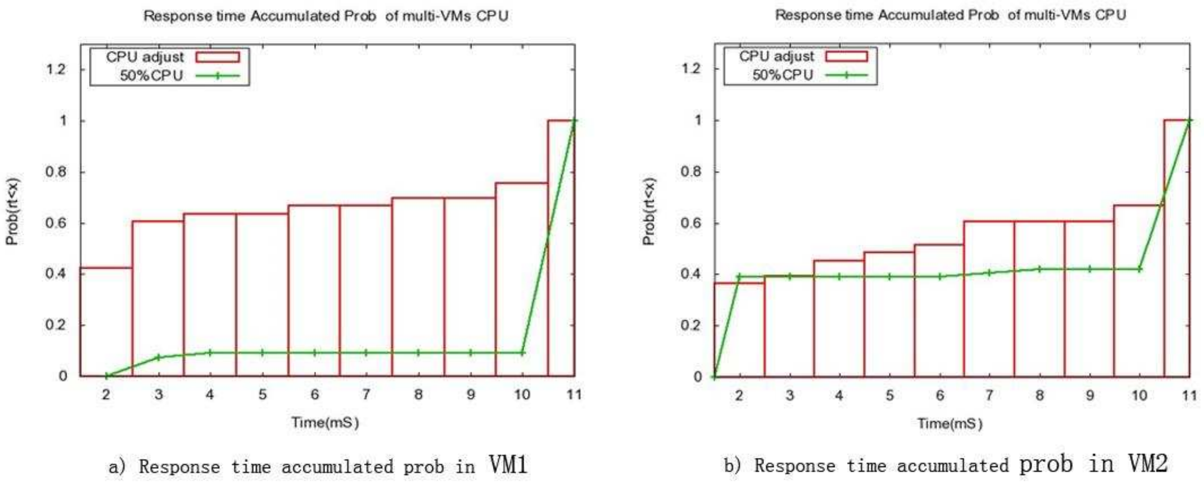


Fig. 4: The CPU allocation of Multi-VMs

time, and even worse that system would collapse due to the large workloads. The probability of response time under 10ms in VM1 is 10%, and the number of VM2 is 40%. While when CPU is regulated dynamically, the probability of response time under 10ms in the two virtual machines is about 70%.

The result of this experiment indicate that the system's CPU dynamic resource management could evidently advance the ability of disposal of WEB servers, and would use system's CPU resource more effectively.

7 Conclusions and future work

In this paper, we explore the resource dynamic management problem for multiple virtual machine in cloud computing. For the CPU performance, which is a kind of high sensitive resource to the QoS, we propose the multiple virtual machines CPU regulation system that includes both the local and global utility device. We transform the multiple virtual machine utility optimization problem into an integer constrained optimization problem. Then we propose a global CPU regulation algorithm based on the utility optimization theory. The experiment results show that the global CPU regulation can raise the Web application response time accumulation probability from 10%–40% to about 70%, which would fully guarantee the quality of user services.

Acknowledgement

This work is supported by National Natural Science Foundation of China (NSFC) under grant No. 61173145, the National Grand Basic Research Program (973 Program) of China under grant No. 2011CB302605, the National High Technology Research and Development Program of China under Grant No. 2010AA012504 and 2011AA010705.

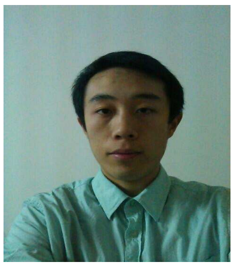
References

[1] B. Hayes. "Cloud computing". Commun. ACM., **7**, 9-11 (2008).
[2] Wikipedia. "Cloud computing". http://en.wikipedia.org/wiki/Cloud_computing
[3] R. P. Goldberg. "Survey of Virtual Machine Research," IEEE Computer, **7**, 34-45 (1974).
[4] Ludmila C, Diwaker G, Amin V. Comparison of the three CPU schedulers in Xen, Sigmetrics Performance Evaluation Review - SIGMETRICS, **2**, 42-51 (2007).
[5] Sriram G, Amitayu D, Bhuvan U et al, Xen and co.: communication-aware CPU scheduling for consolidated Xen-based hosting platforms,VEE. 126-136 (2007).
[6] Pradeep P, Kang G, Zhu X et al, Adaptive control of virtualized resources in utility computing environments, EuroSys - EUROSYS, 289-302 (2007).
[7] Evangelia K, Themistoklis C, Steven H et al, Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters, on Autonomic Computing - ICAC, 117-126.
[8] Song Y, Li Y, Wang H et al, Service-Oriented Priority-Based Resource Scheduling Scheme for Virtualized Utility Computing, High Performance Computing - HiPC, 220-231 (2008).
[9] Italo S, Jussara M. Self-Adaptive Capacity Management for Multi-Tier Virtualized Environments, Integrated Network Management, 129-138 (2007).
[10] Gueyoung J, Kaustubh R, Matti A, et al, Generating Adaptation Policies for Multi-Tier Applications in Consolidated Server Environments,Autonomic computing, 23-32 (2008)
[11] Z. Wang, X. Zhu, P. Padala et al, Capacity and performance overhead in dynamic resource allocation to virtual containers. Integrated Network Management, 149-158 (2007).
[12] Y. Koh, R. Knauerhase, M. Bowman et al. An analysis of performance interference effects in virtual environments. ISPASS, 200-209 (2007).

Weizhe Zhang received the PhD degree in computer architecture from Harbin Institute of Technology, Harbin, China, in 2006. He is currently a professor at School of Computer Science and Technology in Harbin Institute of Technology, China. His research interests include parallel and distributed system, cloud computing.

Hui He received the PhD degree in computer architecture from Harbin Institute of Technology, Harbin, China, in 2007. She is currently an associate professor at School of Computer Science and Technology in Harbin Institute of Technology, China. Her research interests include network security, network computing.

Gui Chen now studies the Bachelor degree at School of Software in Harbin Institute of Technology, China. His research interests include parallel and distributed system, cloud computing.



Jilong Sun now studies the Bachelor degree at School of Software in Harbin Institute of Technology, China. His research interests include parallel and distributed system, cloud computing.