International Journal of Grid Distribution Computing
Vol.8, No.1 (2015), pp.133-144
http://dx.doi.org/10.14257/ijgdc.2015.8.1.13

A Formal Method of CPU Resources Scheduling in the Cloud Computing Environment

Xiaodong Liu[1], Huating Xu[2], Li He[1]
[1]School of Computer, Henan Institute of Engineering, Zhengzhou, 451191, China
[2]Library, Henan Institute of Engineering, Zhengzhou, 451191, China
E-mail:liuxiaodongxht@qq.com, 2279883989@qq.com, engineerheli@126.com

Abstract

In the virtualization based cloud computing environment, multiple computers are allowed to run as virtual machines (VM) in a single physical computer. Efficient scheduling of limited underlying resources has been a key issue. This paper presents a formal method of CPU resources scheduling (FRS). VMs are divided into three resources statuses according to resources requirements and their run information. FRS scheduling is formally scheduling CPU resources according to the resources statuses. The implementation of FRS is confined to the VMM layer, without VM dependency. The evaluation shows that idle CPU resources of VMs are be used by those VMs which need more CPU resources and the CPU resources overall utilization is improved.

Keywords: *formal method; Cloud computing; resources scheduling.*

1. Introduction

With the advantages of functional isolation, manageability and live migration, virtualization technology has been widely used in cloud computing platforms. Internet business motivate server consolidation in data centers, which has become the foundation of cloud computing. In this emerging technology, each physical server runs a software layer called Virtual Machine Monitor (VMM) that supports the execution of multiple VMs and provides safety and isolation to the overlying VMs. These VMs may run different types of applications, such as processor-intensive, I/O-intensive and latency-intensive. These different applications have different resources requirements. Therefore the use of limited underlying resources has been a key issue [1, 2].

However, the current virtualization technology uses static resources allocation mechanism. The underlying physical resources, such as CPU, cannot be fully utilized. Firstly, the workloads of VMs are usually varying. In order to satisfy the resources requirements of VMs, the resources of VMs must be allocated according to its peak requirement. Secondly, some applications, such as I/O intensive application, often consume their CPU resources more slowly. This may lead to VMs waste CPU resources [3].

For improving the CPU resources overall utilization, some dynamic CPU resources scheduling techniques have been proposed [4-6]. These techniques use the average CPU utilization rate [4, 5] or the time intervals between two consecutive virtual clock cycles [6] to diagnose resources requirements of VMs. The virtual machine monitor (VMM) allocates CPU resources according to the resources requirements of VMs. However, these techniques are not based on a formal description and a formal semantics, or provide only a partial formal semantics. In order to effectively reason about resources scheduling, the resources need to be scheduled in a formal method.

This paper presents a formal method for CPU resources scheduling (FRS). FRS scheduling first decides the CPU resources requirements of VMs. Then, VMs are divided into three statuses according to resources requirements and their run information. FRS schedule CPU resources according to VMs' status. The proposed scheme has been implemented in the xen VMM. The distinguished features of CRDA compared to prior work are as follows:

•The FRS schedule CPU resources in a formal method and it can effectively reason about CPU resources scheduling.

•Our implementation is confined to the VMM layer, without VM dependency.

We evaluate the performance of FRS using multiple VMs concurrently running on the same physical server. The experiments show that our FRS can improve CPU resources overall utilization.

The remainder of this paper is organized as follows: Section 2 describes Xen's credit scheduler. Section 3 describes the FRS scheduling model. Section 4 gives the forma method. Section 5 evaluations the performance of the FRS. Section 6 introduces related work. Finally, Section 7 summarizes our conclusions and suggests future works.

2. Background

Xen [7] is an open-source VMM that allows multiple operating systems to share the same physical server in a safe and resource managed fashion. Fig.1 describes the xen architecture. Xen hypervisor provides an abstraction layer between virtual machines and hardware resources. This layer performs functions such as scheduling CPU and allocating memory among virtual machines. There is a privileged domain (Dom0) in the xen VMM which is used to manage guest domains (DomUs). The Dom0 can access to hardware resources directly and DomUs are not allowed to access to hardware resources directly. DomUs can access hardware resources through Dom0.

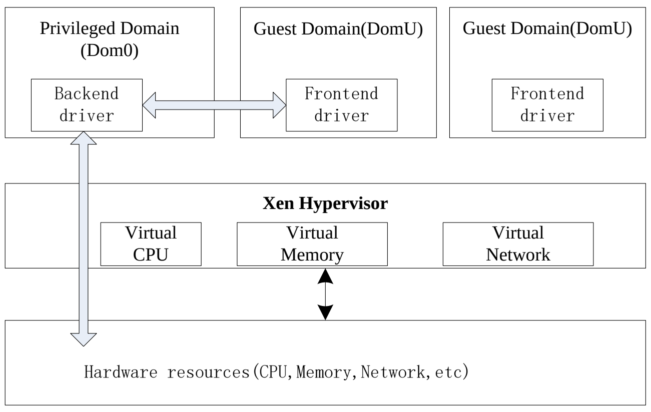


Figure 1. Xen Architecture

In the xen virtualization environment, Guest OS cannot directly schedule physical CPU. Xen establishes the virtual CPU (VCPU) structure and provides one or more VCPUs for every Guest OS. All VCPUs are time-division multiplexing physical CPU. So, xen need reasonable allocation of time slices for VCPUs and schedule them.

The credit scheduler [8] is Xen's default scheduler at present. Its overall objective is to allocate the processor resources fairly. The Guest OS is assigned a weight value when it is created. Each Guest OS is allocated a certain number of credits according to its weight every 30 milliseconds. The credits will be allocated to VCPUs of the Guest OS fairly. As a VCPU runs, it consumes credits. According to the credits of VCPU, a VCPU's priority can be one of the three values: OVER, UNDER and BOOST. If VCPUs are in the OVER state, then they have used up its fair share of CPU resources. If VCPUs are in the UNDER state, then they have CPU resources that can be consumed. The BOOST state provides a mechanism for domains to achieve low I/O response latency. All VCPUs in BOOST state are placed in front of those in UNDER state in the run queue, while those in OVER state are kept in the tail of the queue. Every physical CPU has a run queue of VCPUs. The queue is sorted by the priority of VCPUs and the head of the queue is always selected to run. The credit scheduler is a static scheduling algorithm, and it can dynamically adjust resources.

3. Problem Description

We suppose there are N VMs running on the same physical server Π and the number of CPUs of Π is C_N. All the parameters associated with the physical server and VMs are given in Table 1. The CPU resources of VMs are provided by the Π. Xen VMM allocated the CPU resources to VMs periodically.

Table 1. The Description of Notation

Notation	Meaning
Δt	Xen's scheduling period
VM_i	The i-th VM running on the Π
VC_{ij}	The j-th VCPU of the VM_i
v_i	The number of VCPUs of the VM_i
$R_i^{con}(t_i)$	The consumed CPU resources of the VM_i in the i-th scheduling period
R_i^{alc}	The allocated CPU resources of the VM_i in the i-th scheduling period
$R_i^{req}(t_{i+1})$	The CPU resources requirements of the VM_i in the (i+1)-th scheduling period
$R_i^b(t_i)$	The total idle CPU resources of VMs in the i-th scheduling period
$R_i^l(t_i)$	The total scarce CPU resources of VMs in the i-th scheduling period

Definition 3.1. We suppose the scheduling period is Δt, the total CPU resources provided by the Π in each scheduling period are $C_N \times \Delta t$.

The VM is allocated to the CPU resources according to its weight. When VMs are created, they are allocated to a weight value. We use w_i for the weight of the i-th VM.

Definition 3.2. We define the allocated CPU resources of the VM_i in each scheduling period as:

$$R_i^{alc} = \frac{w_i}{\sum_{i=1}^{N} w_i} \times C_N \times \Delta t$$

Let v_i be the number of VCPUs of the VM_i, we define the allocated CPU resources of each VCPU are v_i.

To execute an application, the VM will consume the CPU resources. We use $R_{ij}^{con}(t_i)$ for the consumed CPU resources of VC_{ij} in the i-th scheduling period and $R_i^{con}(t_i)$ for the consumed CPU resources of the VM_i in the i-th scheduling period.

Definition 3.3. Let t_{ij} is the run time of the VC_{ij} in one scheduling period. We define the $R_{ij}^{con}(t_i)$ is t_{ij} and $R_i^{con}(t_i)$ is $\sum_{j=1}^{v_i} R_{ij}^{con}(t_i)$.

When multiple VMs concurrently running different types of applications. In particular, different combinations of processor-intensive and bandwidth-intensive application are run concurrently. Because the workloads running on the VMs are different, the CPU resources requirements of VMs are also different. I/O-intensive VMs will often consume their CPU resources more slowly than CPU-intensive VMs [9]. This may lead to the I/O-intensive VMs waste CPU resources [3]. The objective of FRS scheduling is to minimize the idle resources.

The CPU resources scheduling problem can then be formulated as:

$$\text{Minimize } \left| \sum_{i=1}^{N} R_i^{con}(t_i) - C_N \times \Delta t \right| \qquad (1)$$

Subject to: (a) $R_i^{alc} \geq 0$, $\forall 1 \leq i \leq N$

(b) $0 \leq R_i^{con}(t_i) \leq v_i \times \Delta t$, $\forall 1 \leq i \leq N$

4. The Formal Method

In general, a FRS-scheduling policy consists of two phases: (1) Predicting future CPU resources requirements, in which the FRS-scheduler periodically collects the running information of VMs. The FRS-scheduler predicts the future CPU resources requirements of VMs based on the collected information. (2) Scheduling phase, in which the scheduling of CPU resources is done.

4.1. Predicting Future CPU Resources Requirements

Let the scheduling period of the FRS-scheduler is Cy. In order to reduce the number of timer and reduce unnecessary system overheads, the scheduling period Cy of FRS should be the integer time of Xen's scheduling period Δt, i.e. $Cy = m \times \Delta t$. We define the CPU resources utilization of the VM in the i-th scheduling period as follows:

Definition 4.1. We define the CPU resources utilization of the VM_i in the i-th scheduling period $U(t_i)$ is $\dfrac{\sum_{j=1}^{v_i} R_{ij}^{con}(t_i)}{Cy \times v_i}$.

After the CPU resources utilization is defined, FRS-scheduler can predict the future CPU resources requirements of VMs. the CPU resources requirements of VMs can be divided into three conditions.

(a) If $R_i^{con}(t_i) < R_i^{alc}(t_i)$ and $U(t_i)$ <1, the allocated CPU resources are not consumed completely, which means that there are idle CPU resources in the VM_i. The future CPU resources requirements can be calculated as follows.

$$R_i^{req}(t_{i+1}) = R_i^{alc}(t_i) - R_i^{con}(t_i) \qquad (1)$$

(b) If $R_i^{con}(t_i) \geq R_i^{alc}(t_i)$ and $U(t_i)$ <1, the allocated CPU resources are consumed completely and it can consume more CPU resources. We use greedy maximize consumption to calculate the future CPU resources requirements. That is to say, we will allocate the CPU resources to the VM_i as far as possible. The equation is as follows.

$$R_i^{req}(t_{i+1}) = Cy \times \Delta t \qquad (2)$$

(c) If $U(t_i)$ =1, the VM_i is running throughout the scheduling period. The future CPU resources requirements of the VM_i is calculated as the equation (2).

The predicting future CPU resources requirements algorithm is formalized in algorithm 1.

Algorithm 1. The predicting algorithm of the CPU resources

```
1 for each VMs do
2  for each VCPUs of the VM do
3    record the run time $t_{ij}$
4   calculate $R_i^{alc}(t_i)$ using definition 3.2, calculate $R_i^{con}(t_i)$ using definition 3.3, calculate
      $U(t_i)$ using equation definition 4.1
5   if $U(t_i)$ =1 then
6     calculate $R_i^{req}(t_{i+1})$ using equation (2)
7   else if $R_i^{con}(t_i) < R_i^{alc}(t_i)$ then
8     calculate $R_i^{req}(t_{i+1})$ using equation (1)
9   else if $R_i^{con}(t_i) \geq R_i^{alc}(t_i)$ then
10    calculate $R_i^{req}(t_{i+1})$ using equation (2)
11   end for
12 end for
```

4.2 Scheduling Phase

Due to the NP hardness of the FRS-scheduling problem described in Section3, it is difficult to find the optimal consumption in polynomial time. Thus, we present a scheduling algorithm which is near the optimal consumption.

Definition 4.2. We define the scheduling status s_i of the VM_i is 1, when $R_i^{req}(t_{i+1}) > R_i^{alc}(t_i)$. We define the scheduling status s_i of the VM_i is -1, when $R_i^{req}(t_{i+1}) < R_i^{alc}(t_i)$. We define the scheduling status s_i of the VM_i is 0, when $R_i^{req}(t_{i+1}) = R_i^{alc}(t_i)$.

Proposition 4.1. (1) $s_i = 1$, if and only if $R_i^{con}(t_i) \geq R_i^{alc}(t_i)$ and $U(t_i)$ <1.

(2) $s_i = -1$, if and only if $R_i^{con}(t_i) < R_i^{alc}(t_i)$ and $U(t_i)$ <1.

(3) $s_i = 0$, if and only if $R_i^{alc}(t_i) = R_i^{con}(t_i)$

Proof. When $s_i = 1$, if $U(t_i)$ =1, then $R_i^{req}(t_{i+1}) = R_i^{con}(t_i)$ according to the condition (c) in section 4.1, this is contradict definition 4.2, if $U(t_i)$ <1 and $R_i^{con}(t_i) < R_i^{alc}(t_i)$, then $R_i^{req}(t_{i+1}) = R_i^{alc}(t_i) - R_i^{con}(t_i) \leq R_i^{alc}(t_i)$, this is contradict definition 4.2, so, $R_i^{con}(t_i) \geq R_i^{alc}(t_i)$ and $U(t_i)$ <1.

If $R_i^{con}(t_i) \geq R_i^{alc}(t_i)$ and $U(t_i)$ <1, $R_i^{req}(t_{i+1}) = Cy$ according to the condition (b) in section 4.1. Because $U(t_i)$ <1, $R_i^{alc}(t_i)$ <Cy, $R_i^{req}(t_{i+1}) \geq R_i^{alc}(t_i)$, so $s_i = 1$.

The proof of (2) and (3) are similar to the (1). The proof of (2) and (3) are omitted due to space limitations.

FRS-scheduler firstly calculates the following total value, the equation are as follows.

$$R^b(t_i) = \sum_{\substack{i=1 \\ s_i=1}}^{N} (R_i^{req}(t_{i+1}) - R_i^{alc}(t_i)) \qquad (3)$$

$$R^b(t_i) = \sum_{\substack{i=1 \\ s_i=-1}}^{N} (R_i^{alc}(t_i) - R_i^{req}(t_{i+1})) \qquad (4)$$

Thus, the FRS-scheduler will schedule the CPU resources according to the equation (3) and (4).

If $R^b(t_i) \le R^l(t_i)$, VMs which need more CPU resources can be satisfied by those VMs which have idle CPU resources. FRS-Scheduler will schedule CPU resources as follows:

The allocated CPU resources of VMs can be calculated by equation (5) when $s_i = 1$ or $s_i = 0$.

$$R_i^{alc}(t_{i+1}) = R_i^{req}(t_{i+1}) \qquad (5)$$

The allocated CPU resources of VMs can be calculated by equation (6) when $s_i = -1$.

$$R_i^{alc}(t_{i+1}) = R_i^{req}(t_{i+1}) + \frac{(R^l(t_i) - R^b(t_i))}{\sum_{\substack{i=1 \\ s_{i=-1}}}^{N} R_i^{con}(t_i)} \times R_i^{con}(t_i) \qquad (6)$$

Eq. (5) indicates that VMs which need more CPU resources can be satisfied completely. Eq. (6) indicates that VMs which have idle CPU resources need not provide all their idle CPU resources.

If $R^b(t_i) > R^l(t_i)$, VMs which need more CPU resources can not be satisfied completely. The FRS-scheduler allocates CPU resources according to the requirements of VMs. The allocated CPU resources of VMs can be calculated by equation (5) when $s_i = -1$ or $s_i = 0$. The allocated CPU resources of VMs can be calculated by following formula (7) when $s_i = 1$.

$$R_i^{alc}(t_{i+1}) = R_i^{alc}(t_i) + \frac{(R^b(t_i) - R^l(t_i))}{\sum_{\substack{i=1 \\ s_{i=1}}}^{N} R_i^{req}(t_{i+1})} \times R_i^{req}(t_{i+1}) \qquad (7)$$

Eq. (7) indicates that VMs which need more CPU resources can not satisfy completely. They are allocated CPU resources according to their requirements.

The CPU resources scheduling algorithm is formalized in algorithm 2.

Algorithm 2 the CPU resources scheduling

```
1 compute $R^b(t_i)$ and $R^l(t_i)$
2 for each VMs do
3 if $R^b(t_i) \le R^l(t_i)$ then
4   if $s_i = 1$ or $s_i = 0$ then
5   compute $R_i^{alc}(t_{i+1})$ according to Eq. (5)
```

```
6   if $s_i = -1$ then
7   compute $R_i^{alc}(t_{i+1})$ according to Eq. (6)
8 else if $R^b(t_i) > R^l(t_i)$ then
9   if $s_i = -1$ or $s_i = 0$ then
10    compute $R_i^{alc}(t_{i+1})$ according to Eq. (5)
11   if $s_i = 1$ then
12    compute $R_i^{alc}(t_{i+1})$ according to Eq. (7)
13 end for
```

5. Evaluation

We have implemented FRS scheduling in xen 4.1.2 hypervisor. This section evaluates the performance of FRS scheduling.

5.1. Experiment Setup

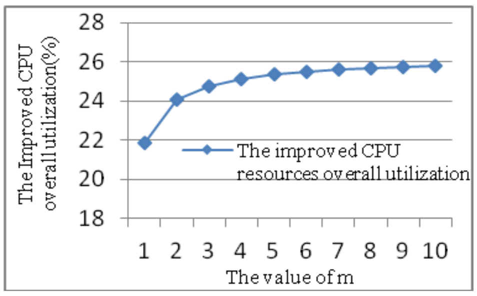
Our system is installed on the physical machine equipped with two Inter(R) Xeon(R) 4-core CPU running at 2.40GHz,32G of RAM. The operation system is running the 64-bit version of Ubuntu 12.04. The system is configured with multiple VMs, and each VM is configured with 2G memory. The number of VCPUs and used threads of VMs are listed in Table 2. In order to facilitate comparison, each threads running a matrix multiplication C=AB, and the size of A and B are both 2000×2000. The matrix multiplication is implemented by BSPCloud [10].

Table 2. The Number of VCPUs and Used Threads of VMs

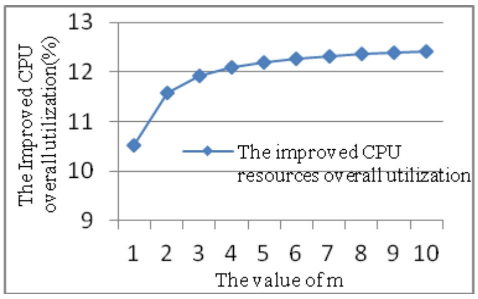
Case1			Case2			Case 3			Case4		
VM	VCPUs	Threads	VM	VCPUs	Threads	VM	VCPUs	Threads	VM	VCPUs	Threads
VM1	8	1	VM1	2	I/O	VM1	2	2	VM1	2	1
VM2	8	2	VM2	4	3	VM2	4	4	VM2	4	2
VM3	8	4	VM3	6	5	VM3	6	6	VM3	6	3
VM4	8	8	VM4	8	7	VM4	8	8	VM4	8	4

5.2. The Influence of Scheduling Period

The scheduling period has an important influence on the FRS performance. If the scheduling period is too long, the accuracy of the prediction will decline. If the scheduling period is too short, the scheduling overhead will increase. This sub-section tests the influence of scheduling period to CPU overhead and CPU overall utilization. In this set of experiments, we use the case 1 and case 2 in the Table 1. Fig.2 shows the relationship between the improved CPU overall utilization and the value of m. With the increase of m, the improved CPU resources utilization of the FRS scheduling increases rapidly. When the m increased to 9, the increase curve becomes flat. Fig.3 shows the relationship between the overhead of the FRS scheduling and the scheduling period. The CPU overhead declines with the increase of m. When the m increased to 9, the CPU overhead curve becomes flat. The more is the scheduling period, the less is the overhead. To balance the accuracy of the prediction and the overhead of the CPU, we select $9 \times \Delta t$ as the FRS scheduling period.



(a) Case1



(b) Case2

Figure 2. The Relationship between the Improved CPU Overall Utilization and the Scheduling Period

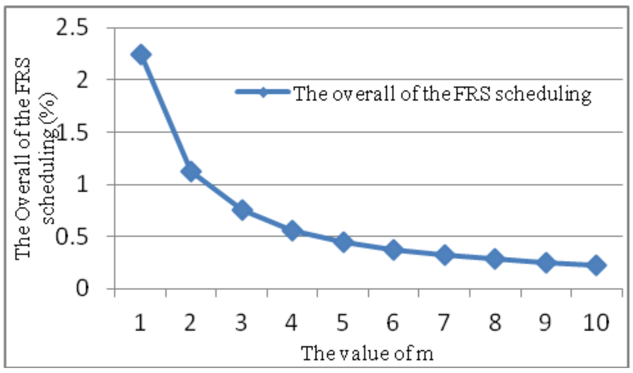


Figure 3. The Relationship between the CPU Overhead and the Scheduling Period

5.3. The Evaluation of FRS Scheduling

In this set of experiments, we evaluate the performance of FRS scheduling. To measure the effectiveness of FRS scheduling, we compare the schedule of FRS (FRS_xen) with the xen scheduling (Orig_Xen). Four VMs are concurrently running on the same physical server. We consider four scenarios of VMs.

(1) Case 1

We first consider the scenario that the idle CPU resources are more than the needed CPU resources. As is shown in the Table 2 (case 1), The VM 4 needs more CPU resources, and the CPU resources of the VM1, VM2 and VM3 are abundant. The FRS scheduling will reclaim idle CPU resources and allocate them to the VM4. Because the FRS scheduling only reclaims idle resources of VMs, the performance of VM1, VM2 and VM3 is not influenced. The performance of VM4 is improved. The run time of the application running on the VM4 declines by 60%. Fig.4 (a) shows the experiment results.

(2) Case 2

We then consider the scenario that the idle resources are less than the needed CPU resources. As is shown in the Table 2 (case 2), the application of the VM1 is perform I/O operation. The CPU resources of VM1 are abundant, and the VM2, VM3 and VM4 need more CPU resources. The FRS scheduling will reclaim idle CPU resources of the VM1 and allocate them to the VM2, VM3 and VM4. The FRS scheduling allocates idle CPU resources according to the requirements of VMs. That is to say, the VM which need more CPU resources will be allocated more. Fig.4 (b) shows the experiment results. The application running on the VM2, VM3 and VM4 declines by 13.20%, 10.11% and 8.17% respectively. This is because the threads of VM4 are more than VM2 and VM3, the increased CPU resources of each thread are least. So, the performance of VM4 increase is least. The improved CPU resources of the VM2, VM3 and VM4 increase 3.04%, 3.38% and 3.56 respectively.

(3) Case 3 and Case 4

Thirdly, we consider the scenario that all VMs need more CPU resources. In this case, FRS scheduling cannot reclaim idle CPU resources and will do not rescheduling CPU resources. We also consider the scenario that all VMs have idle CPU resources, but there is no VMs need more CPU resources. In this case, the FRS scheduling is also not rescheduling CPU resources. Fig. 4 (c) and Fig.4 (d) show the experiment results. In the Fig. 4(c) and Fig.4 (d), the performance of VMs is almost the same in the Orgi_xen scheduling and FRS scheduling.

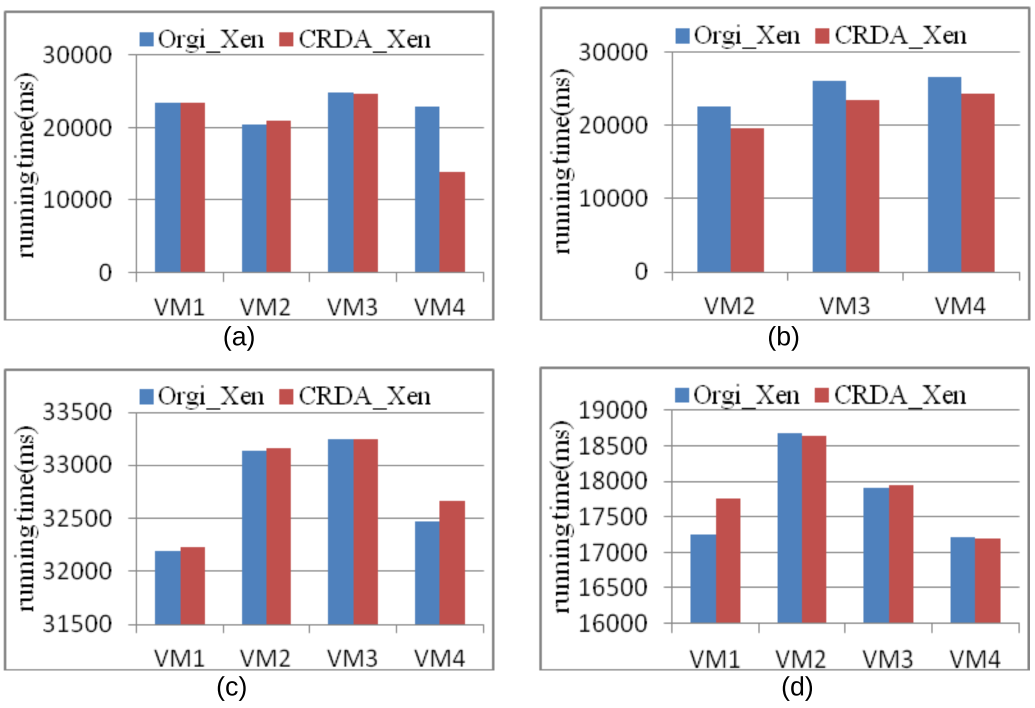


Figure 4. The Performance Comparison between Orgi_Xen Scheduling and FRS_Xen

6. Related Work

CPU resources scheduling has attracted considerable research attention. In order to improve CPU resources overall utilization, dynamic CPU resources allocation techniques have been proposed [4-6, 11, 12]. The average CPU utilization rate [4, 5] or the time intervals between two consecutive virtual clock cycles are used to predict CPU resources requirements, and the CPU resources are allocated to VMs on-demand. In order to improve the predictive ability, fuzzy modelling [11, 12] is used to learn and predict the CPU resources requirements of VMs.

There are also some work considers the task types running on the VM. The scheduling algorithms that support real time tasks in the VM have to finish before their deadline have been proposed in [13-15]. HwanjuKim et al. [16] present virtual machine scheduling techniques for transparently bridging the semantic gap between the VMM and VMs in order to improve I/O performance without compromising CPU fairness. Task-aware scheduling [17] and Communication-aware scheduling [18] are both application-aware scheduling strategies. The credit scheduler is enhanced by making it full-time aware of inter-VM events and physical interrupt request events that improve the responsiveness of VMs doing mixed workloads [19].

There has been some work on co-scheduling of VCPUs. Co-scheduling [20, 21] is a technique to prevent VCPUs of the same domain are stacked on the run queues of the same physical processor. In this technique, all the VCPUs of a domain are scheduled at the same time or none of them is scheduled. However, this technique may suffer from the drawbacks like CPU fragmentation. Pratik Shinde et al. [22] propose co-scheduling only if concurrency degree θ of domain is greater than threshold value δ. These co-scheduling also do not consider the unbalance of resources utilization.

7. Conclusions and Future Work

In this paper, we have proposed a formal method of CPU resources scheduling in cloud computing environment to improve the CPU resources overall utilization. FRS is a periodical scheduling algorithm. With each periodical round, FRS uses the allocated credits and consumed credits to diagnose the CPU resources requirements of VMs. Then, VMs are divided into three statuses according to resources requirements and their run information. FRS dynamically schedule CPU resources according to VMs' status. FRS is implemented confined to the VMM layer, without VM dependency. The experiment results show that FRS can improve CPU resources overall utilization.

Sometimes, the CPU utilization of the VM is not high, but one or more VCPUs of the VM are very high. That is to say, the CPU utilization of different VCPUs is unbalanced. For example, if a single-thread application running on the multi-VCPU VM, there will be only one VCPU is very busy. In the next, we will consider the condition that some VCPUs' utilization of the VM is higher but others in the same VM are less and research on CPU resources adjustment mechanism of VCPUs on the same VM.

Acknowledgements

This paper is a revised and expanded version of a paper entitled [A workload-aware CPU resources scheduling for the cloud computing environment] presented at [The 3rd International Conference on Electrics, Electronics, and Computer Science,October 24-26, 2014 at Hochimin, Vietnam.]

This work is supported by Doctor Foundation of Henan Institute of Engineering, Foundation of He'nan Educational Committee (13A520148), and a grant from National Science Fund for Young Scholars （No. 61301232）

References

[1] C. L. Li and L. Y. Li, "Optimal resources provisioning for cloud computing environment", Journal of Supercomputing, vol. 62, no. 2, **(2012)**, pp. 989-1022.
[2] K. Begnum, "Simplified cloud-oriented virtual machine management with MLN", Journal of Supercomputing, vol. 61, no. 2, **(2012)**, pp. 251-266.
[3] http://xenproject.org.
[4] Z. Y. Shao, H. Jin, Y. Li and J. Huang. "XenMVM: Exploring Potential Performance of Virtualized Multi-core Systems", Information-an International Interdisciplinary Journal, vol. 14, **(2011)**, pp. 2315-2326.
[5] S. B. Nigmandjanovich and C. W. Ahn, "Policy-based dynamic resource allocation for virtual machines on Xen-enabled virtualization environment", 2nd IEEE International Conference on Advanced Computer Control, **(2010)**.
[6] Y. Zhang, A. Bestavros, M. Guirguis, I. Matta and R. West, "Friendly virtual machines: leveraging a feedback-control model for application adaptation", Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments, **(2005)**.
[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, "Xen and the art of virtualization", ACM SIGOPS Operating Systems Review, vol. 37, **(2003)**, pp. 164-177.
[8] Credit scheduler. http://xen.org/files/summit_3/sched.pdf.
[9] D. Ongaro, A. L. Cox and S. Rixner, "Scheduling I/O in virtual machine monitors", Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, **(2008)**.

[10] X. D. Liu, "A Programming Model for the Cloud Platform", International Journal of Advanced Science and Technology, vol. 57, **(2013)**, pp. 75-82

[11] W. Lixi, X. Jing, Z. Ming, T. Yicheng and J. A. B. Fortes, "Fuzzy Modeling Based Resource Management for Virtualized Database Systems", Proceedings of the IEEE 19th International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems , **(2011)**.

[12] R. Jia, W. Yudi, G. Jiayu and X. C. Zhong, "DynaQoS: Model-free Self-tuning Fuzzy Control of Virtualized Resources for QoS Provisioning", IEEE Nineteenth IEEE International Workshop on Quality of Service, **(2011)**.

[13] Y. Wang, X. Wang and H. Guo, "An Optimized Scheduling Strategy Based on Task Type In Xen", Advances in Automation and Robotics, vol. 2, **(2012)**, pp. 515-522.

[14] B. K. Kim, K. W. Hur, J. H. Jang and Y. W. Ko, "Feedback scheduling for realtime task on xen virtual machine", Communication and Networking, **(2011)**, pp. 283-291.

[15] Y. H. Jun, K. Y. Rok, K. J. Young and P. S. Yong, "A CPU Allocation Method Considering the Control Domain Overhead in Xen Virtualization Environment", Journal of KIISE: Computer Systems and Theory, vol. 39, **(2012)**, pp. 1-11.

[16] H. Kim, H. Lim, J. Jeong, H. Jo, J. Lee and S. Maeng, "Transparently bridging semantic gap in cpu management for virtualized environments", Journal of Parallel and Distributed Computing, vol. 71, **(2011)**, pp. 758-773.

[17] S. Govindan, J. Choi, A. R. Nath, A. Das, B. Urgaonkar and A. Sivasubramaniam, "Xen and Co.: communication-aware CPU management in consolidated Xen-based hosting platforms", IEEE Transactions on Computers, vol. 58, **(2009)**, pp. 1111-1125.

[18] H. Kim,H. Lim, J. Jeong, H.Jo and J. Lee, "Task-aware virtual machine scheduling for I/O performance", Proc. of ACM SIGPLAN/ SIGOPS International Conference on Virtual Execution Environment, **(2009)**; Washington, DC, USA.

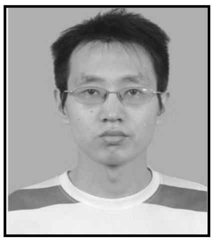
[19] D. H. Liu, J. L. Cao and J. Cao, "FEAS: A full-time event aware scheduler for improving responsiveness of virtual machines", Proceedings of the Thirty-Fifth Australasian Computer Science Conference, **(2012)**.

[20] VMware, Inc., Co-scheduling SMP VMs in VMware ESX Server, http://communities.vmware.com/docs/DOC-4960.

[21] O. S. Wong and H. S. Kim, "Is co-scheduling too expensive for SMP VMs", In: EuroSys, **(2011)**.

[22] P. Shinde, A. Tomar, K. Shah, S. Kalra and D. Kulkarni, "Concurrency Aware Dynamic Scheduler for Virtualized Environment", Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA), **(2013)**.

Authors



Xiaodong Liu. He received his Ph.D. degree from Shanghai University. He is currently an associate Professor of Henan Institute of Engineering. His primary research interests cover virtualization, cloud computing and grid computing

Xuating Xu. He is currently an assistant of Henan Institute of Engineering. His primary research interests is cloud computing

Li He. He received his Ph.D. degree from Sun Yat University. He is currently an Instructor of Henan Instituteof Engineering. His primary research interests cover swarm intelligence, things of internet.