International Journal of Grid Distribution Computing
Vol.8, No.1 (2015), pp.247-258
http://dx.doi.org/10.14257/ijgdc.2015.8.1.23

A Workload-aware Resources Scheduling Method for Virtual Machine

Hongshan Qu, Xiaodong Liu, Huating Xu

School of Computer, Henan Institute of Engineering, Zhengzhou, 451191, China
E-mail: qhs@haue.edu.cn,liuxiaodongxht@qq.com,2279883989@qq.com,

Abstract

Virtualization-based cloud computing platforms allow multiple virtual machines (VMs) running on the same physical machine. Efficient allocation of limited underlying resources has been a key issue. In order to improve the CPU resources utilization, this paper presents a workload-aware CPU resources scheduling method (WARS). WARS uses the allocated credits and consumed credits to diagnose the CPU resources requirements of VMs and dynamically adjusts CPU resources according to the requirements of VMs. The adjustment of CPU resources is converted into increased or decreased weights of VMs. The implementation of WARS is confined to the VMM layer, without VM dependency. Our evaluation shows that WARS can improve the overall utilization of CPU resources.

Keywords: *Virtualization; Virtual Machine; Resource allocation.*

1. Introduction

Virtualization-based cloud computing environment allows multiple virtual machines (VMs) running on the same physical machine. 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. 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 the VM are usually varying. In order to satisfy the resources requirements of the VM, the resources of the VM must be allocated according to its peak requirement. The resources of the VM will be wasted except in peak condition. Secondly, the workloads of some virtual machine are light but they occupy physical machine resources. The idle resources cannot be used by the VMs whose workloads are heavy. Thirdly, some applications have been completed but the user may not destroy the virtual machine, so that the resources of the VM will be wasted.

For improving the CPU resources utilization, some dynamic CPU resources allocation methods have been proposed [1-3]. These methods use the average CPU utilization rate [1, 2] or the time intervals between two consecutive virtual clock cycles [3] to diagnose resources requirements of VMs. The virtual machine monitor (VMM) allocates CPU resources according to the resources requirements of VMs. In order to improve the predictive ability, the authors of [4, 5] use fuzzy modeling to learn and to predict the CPU resources requirements of the VM. However, CPU resources metrics such as CPU utilization, response time and throughput are not particularly useful in predicting workload.

This paper presents a workload-aware CPU resources scheduling method (WARS). The WARS uses the allocated credits and consumed credits to diagnose the CPU resources requirements of VMs. If the VM has not consumed its allocated credits in a schedule period, it means the VM needs less CPU resources. Or else, if the VM has

ISSN: 2005-4262 IJGDC

consumed all its credits before the schedule period, it means the VM needs more CPU resources. The ratio of the consumed credits and consumed allocated credits can diagnose the actual CPU resources requirements more accurately. Based on the CPU resources requirements of VMs, the WARS will adjust CPU resources of VMs in the next schedule period. The CPU resources adjustment of WARS is converted into increased or decreased weight of VMs.

The proposed scheme has been implemented in the xen VMM. The distinguished features of WARS compared to prior work are as follows: First, WARS uses the ratio of the consumed credits and consumed allocated credits to diagnose the CPU resources requirements. Our credits based CPU resources diagnose scheme is more useful in predicting workload. Second, our implementation is confined to the VMM layer, without VM dependency. Previous resources diagnose scheme is implemented by VM itself [1-3, 6] and they are VM dependency.

The remainder of this paper is organized as follows: Section 2 describes the background. Section 3 describes the scheduling model. Section 4 shows the experimental results of the WARS. Section 5 introduces the related work. Finally, Section 6 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 machine safely. It provides performance isolation among VMs and manages access to underlying physical resources.

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.

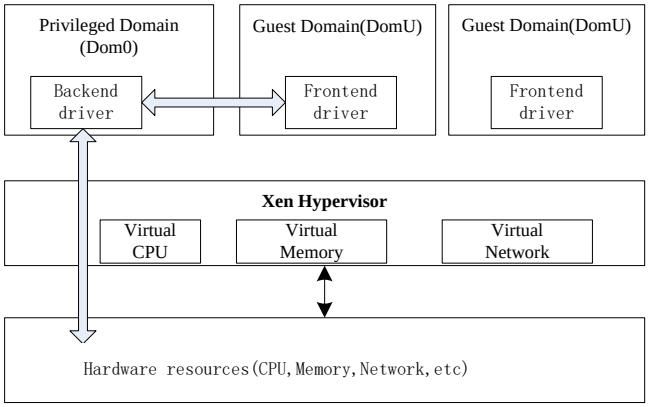


Figure 1. Xen Architecture

The credit scheduler is Xen's default scheduler at present. Its overall objective is to allocate the processor resources fairly [8]. Every physical CPU has a run queue of virtual CPUs (VCPUs). The queue is sorted by the priority of the VCPUs and the head of the queue is always selected to run. As a VCPU runs, it consumes credits. 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. The domain is assigned a weight value when it is created. Each domain is allocated a certain number of credits according to its weight every 30 milliseconds. The credit will be allocated to VCPUs of the domain fairly. Then the priority of each VCPU will be recalculated.

The advantage of the credit scheduler is to guarantee each domain shares the underlying CPU resources fairly. However, credit scheduler is a static allocation method. The CPU resources cannot be fully utilized when the load of the domain is varying and the load of domains is different.

3. The WARS Scheduling Model

In this section, we describe the WARS scheduling model and the evaluation model.

3.1. WARS Architecture

With WARS, the underlying physical CPU resources can be allocated dynamically according to the demand of VMs. This can make full use of idle CPU resources and increase the overall CPU resources utilization. Fig.2 shows the WARS architecture. In the xen hypervisor, we add a monitor module which can monitor the CPU resources consumption of every VMs and VCPUs and add resources allocate module which allocates underlying CPU resources according to the monitor. WARS monitors consumption of CPU resources of every VCPUs and VMs at every time interval. If the consumed CPU resources of VCPUs or VMs are different, the WARS will adjust the CPU resources of VCPUs and VMs according to their requirements.

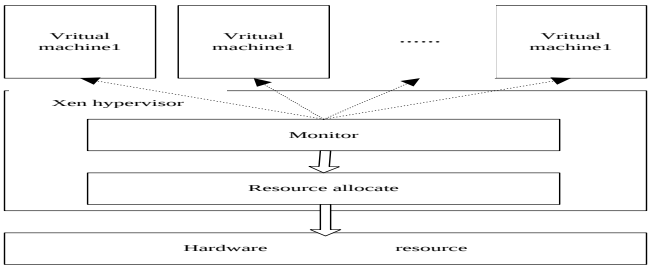


Figure 2. WARS Architecture

3.2. Resources Requirement Diagnose Model

We assume there are N VMs running on the same physical machine. VM_i is the i-th VM and VC_{ij} is the j-th VCPU of the VM_i. Let w_i be the weight of the VM_i and v_i be the number of VCPUs of the VM_i. The total weight can be calculated as follows:

$$W_{total} = \sum_{i=1}^{N} w_i \times v_i \quad (1)$$

In order to diagnose the actual resource requirements of a VM, the WARS gathers the consumed credits and allocated credits of every VCPUs and VMs every time interval. The CPU utilization rate of VC_{ij} in the i-th time interval t_i can be calculated as follows:

$$U_{ij}(t_i) = \frac{C_{ij}^{con}(t_i)}{C_{ij}^{alc}(t_i)} \quad (2)$$

where $C_{ij}^{con}(t_i)$ represents the consumed credits of VC_{ij} at time interval t_i and $C_{ij}^{alc}(t_i)$ represents the allocated credits of VC_{ij} at time interval t_i. We also define the CPU utilization rate of the VM_i using the following formula 3.

$$U_i(t_i) = \frac{\sum_{j=1}^{v_i} C_{ij}^{con}(t_i)}{\sum_{j=1}^{v_i} C_{ij}^{alc}(t_i)} \quad (3)$$

If the consumed credits of the VM_i are less than its allocated credits, the CPU resources of the VM_i are abundant. Or else, the CPU resources of the VM_i are shortage. Therefore, the $U_i(t_i)$ can be used to diagnose the CPU resources requirement of the VM_i.

In our WARS, the CPU resources adjustment among VMs is converted into increased or decreased weights of VMs. The CPU resources adjustment is based on the value of the $U_i(t_i)$. If the $U_i(t_i)$ is less than a threshold $U_{\min}$, in the next schedule period, the VMM will get redundant weight of VMs back so that the weight can be assigned to VMs which request more weight. The VMM don not get all redundant resources of VMs. The VMM must guarantee VMs can work when their resources are got back, so we define the status U_{normal} which is the CPU utilization that can guarantee VMs to work. The redundant weight of the VM_i is the corresponding weight of ratio $(U_{normal} - U_i(t_i))$ to $U_i(t_i)$. The redundant weight of the VM_i that can be calculated as follows:

$$w_i(t_{i+1}) = \frac{w_i(t_i) \times v_i \times (U_{normal} - U_i(t_i))}{U_i(t_i)} \quad (4)$$

On the contrary, if the $U_i(t_i)$ is more than a threshold $U_{\max}$, in the next scheduling period, the VM_i will request more weight to VMM. The request weight strategy is to reduce the CPU utilization $U_i(t_i)$ to U_{normal}. The requested weight of the VM_i is the corresponding weight of ratio $(U_i(t_i) - U_{normal})$ to $U_i(t_i)$. The requested weight can be calculated as follows.

$$w_i(t_{i+1}) = \frac{w_i(t_i) \times v_i \times (U_i(t_i) - U_{normal})}{U_i(t_i)} \quad (5)$$

3.3. Resource Scheduling Model

The WARS diagnoses the CPU resources requirements of VMs according to the CPU utilization defined by formula (2-3) at every time interval. Based on the CPU resources requirements of VMs, WARS will adjust the CPU resources of VMs. The adjustment of VMs' CPU resources is converted into increased or decreased weights of VMs.

Next, we explain WARS in more details.

CPU resources diagnoses. In a virtualization environment, the VM may be created or destroyed at any time. When a new VM is created, there may be multiple VMs running and the VM is not monitored immediately. Only when a scheduling period is finished, the VM begins to be monitored. WARS will record the allocated credits,the consumed credits and the running time in one scheduling period. The CPU resources requirement can be predicted through the CPU utilization defined by formula (2-3).

Request more or less CPU resources to VMM. If the CPU utilization of VM_i is more than a threshold $U_{\max}$, it will apply to borrow more weights in the next scheduling period. The borrowed weight which is requested by VM_i is (6)

$$w_i^b(t_{i+1}) = \frac{w_i \times v_i \times (U_i(t_i) - U_{Normal})}{U_i(t_i)} \tag{6}$$

The total weight ω_{total}^b which needs to borrow in the next scheduling period is (7)

$$\omega_{total}^b(t_{i+1}) = \sum_{i=1}^{n} \omega_i^b(t_{i+1}) \tag{7}$$

If the CPU utilization of VM_i is less than a threshold $U_{\min}$, it will apply to lend some weight in the next scheduling period. The lent weight can be calculated as follows.

$$w_i^l(t_{i+1}) = \frac{w_i(t_i) \times v_i \times (U_{Normal} - U_i(t_i))}{U_i(t_i)} \tag{8}$$

The total weights which can lend in the next scheduling period are (9)

$$\omega_{total}^l(t_{i+1}) = \sum_{j=1}^{n} \omega_j^l(t_{i+1}) \tag{9}$$

Weight adjustment. The adjustment of WARS is divided into three conditions according to $\omega_{total}^b(t_{i+1})$ and $w_i^l(t_{i+1})$.

If $w_i^l(t_{i+1}) > 0$ and $\omega_{total}^b(t_{i+1}) > w_i^l(t_{i+1})$, the weights which VM apply to borrow cannot be satisfied completely. WARS allocates idle weights according to the needed weight of VMs. The actual borrowed weight of VM_i can be calculated as follows:

$$\omega_i^{ab}(t_{i+1}) = \frac{w_{total}^l(t_{i+1}) \times w_i^b(t_i)}{w_{total}^b(t_{i+1})} \tag{10}$$

If $w_i^l(t_{i+1}) > 0$ and $\omega_{total}^b(t_{i+1}) <= w_i^l(t_{i+1})$, the VMs which apply to borrow weight can be satisfied and the VMs which apply to lend only need to lend a part. The actual lent weight of VM_i can be calculated as follows:

$$\omega_i^{al}(t_{i+1}) = w_i^l(t_{i+1}) - \frac{(w_i^l(t_{i+1}) - \omega_{total}^b(t_{i+1})) \times w_j^l(t_{i+1})}{w_{total}^l(t_{i+1})} \tag{11}$$

If $w_i^l(t_{i+1}) <= 0$ and $\omega_{total}^b(t_{i+1}) > 0$, there are VMs which apply to more weights but there is no idle CPU resources can be lent. WARS will reallocate CPU resources. The allocation algorithm should have the following properties: 1) the interest of every VM must be guaranteed; 2) the schedule should do its best to help those VMs which CPU resources are short of.

In order to reallocate resources satisfied the above two properties, we borrow the idea of stock shares [9, 10]. VMs which have more shares will be allocated more resources. The shares can be weight of VMs or fees paid by users. In this paper, we use the number of VCPUs to define shares of VMs. The total shares can be calculated as follows.

$$V_{total} = \sum_{i=1}^{N} v_i \tag{12}$$

The weight of VM_i which should be assigned in the next scheduling can be calculated using following formula (13-14).

$$w_i(t_{i+1}) = W_{total} \times (\alpha \frac{v_i}{V_{total}} + \beta \frac{w_i^b(t_{i+1})}{w_{total}^b(t_{i+1})}) \tag{13}$$

$$\alpha + \beta = 1 \tag{14}$$

Where α and β reflect the importance of shares and resources utilization rate. The weight is assigned according to the shares of CPU resources when $\alpha = 1$. The weight is assigned according to the utilization of VMs.

3.4 The Evaluating Model

We assume the number of physical CPU is N_{cpu} and the credits of the physical CPU can allocate at every time interval Δt is $C_{default}$, in the m time intervals t, the total CPU resource is (15)

$$C_{total} = N_{cpu} \times C_{default} \times m \times \Delta t \tag{15}$$

When the VCPU is running, it will consume the credit. If the running time of the VC_{ij} is t_{ij} in the time interval t, the consumed credit is (16)

$$C_{con}^{ij}(t_{ij}) = t_{ij} \times 10 \tag{16}$$

The total consumed credits in the time interval t can be calculated as follows:

$$C_{con}(t) = \sum_{i=1}^{n} \sum_{j=1}^{v_i} C_{con}^{ij}(t_{ij}) \tag{17}$$

The overall CPU utilization rate is (18)

$$U(t) = \frac{C_{con}(t)}{C_{total}(t)} \tag{18}$$

4. Performance Evaluation

4.1. Experiment Setup

We have implemented WARS mechanism in xen 4.1.2 hypervisor. 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 application running in the VMs is matrix multiply implemented by BSPCloud [11-13]. This section presents evaluation results for different types of application and various workloads.

The scheduling period has an important influence on the scheduling algorithm 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. On the other hand, in order to reduce the number of timer and reduce unnecessary system overheads, the scheduling period t of WARS should be the integer time of VM scheduling

period Δt , i.e. $t = m \times \Delta t$. Based on our previous research, this paper selects $9 \times \Delta t$ for the scheduling period of WARS.

4.2. Performance Evaluation

In this set of experiments, we estimate the performance of WARS. To measure the effectiveness of WARS, we compare the schedule of WARS (WARS _xen) with the xen scheduling (Orig_Xen).

Table 1. Application Types by Different VMs

VM	VCPUs	Application type	size	used threads
VM1	4	Matrix multiplication	1248×1248	1
VM2	4	Matrix multiplication	1554×1554	2
VM3	4	Matrix multiplication	2100×2100	4
VM4	4	I/O operation	ping	1

In the first set of experiments, we run four VMs concurrently and the application running in the VM is shown in Table 1. The VM3 is CPU resources shortage. The CPU resources of the VM1 and VM4 are abundant. When they are running on the same physical machine, the WARS scheduling will reclaim excess resources of VM1 and VM4, and the reclaimed resources will lend to VM3. Because the WARS scheduling only reclaim the excess resources of VMs, the performance of the application which running on the VM1 and VM4 will not influenced. The performance of VM4 will improved because it will get more resources by WARS scheduling. Fig.3 shows the experiment result. Because there are idle resources in the VM1 and VM4, the WARS scheduling will adjust the CPU resources. The idle CPU resources of the VM1 and VM4 will be allocated to the VM4. The computing time of the application run in the VM4 declines about 15%.

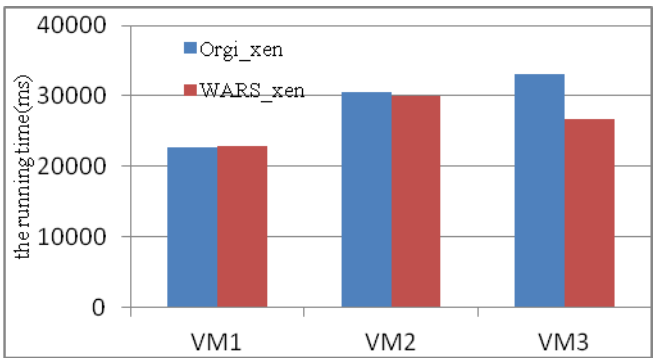


Figure 3. CPU Resources Allocation for Multiple VMs Running on the Same Physical Machine (the Idle Resources are More than Needed Resources)

We notice that the CPU overall utilization only improves 6%. Although VM1, VM2 and VM4 have many idle resources, the VM3 need less CPU resources. So the CPU overall utilization only improves 6%.

In the second set of experiment, we also run four VMs concurrently and the application running in the VM is shown in table 2. The VM2, VM3 and VM4 are all CPU resources shortage. Only VM1 has idle CPU resources. The excess CPU resources of VM1 will be reclaimed by xen VMM. These excess CPU resources will be allocated to the VM2, VM3 and VM4 according to their CPU utilization. So the performance of the application running on the VM2, VM3 and VM4 will be improved through WARS scheduling. The performance of the application running on the VM1is not influenced, because only its

excess CPU resources are reclaimed by xen VMM. Fig.4 shows the experiment result. The computing time of the application run in the VM2, VM3 and VM4 decline about 13%. Because the CPU resources requirement of the VM2, VM3 and VM4 is almost the same, they will generate the same CPU resources request and get the same CPU resources. Only the idle CPU resources of the VM1 are allocated to the VM2, VM3 and VM4, the run time of the application run in the VM1 is almost not changed.

We also notice that the CPU overall utilization improves 14.6%, this is because the idle CPU resources of VM3 is made full use of.

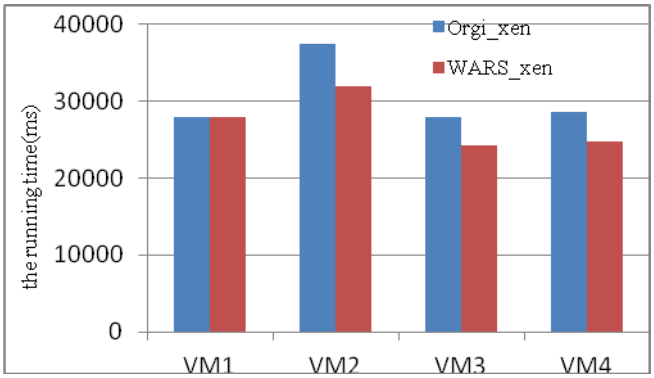


Fig.4. CPU Resources Allocation for Multiple VMs Running on the Same Physical Machine (the Idle Resources are Less than Needed Resources)

In the third set of experiment, we run four VMs concurrently and all VMs are CPU resources shortage. The application running in the VM is shown in table 3. We set parameter $\alpha = 0, \alpha = 0.2, \alpha = 0.4, \alpha = 0.6, \alpha = 0.8,$ and $\alpha = 1.0$ respectively. Fig.5 shows the experiment results. When the parameter α is big and β is small, the VMM will allocate more CPU resources to VMs which have more weight. On the contrary, the VMM will allocate more CPU resources to VMs which have more resources requirement. α and β reflect the significance of shares and resources utilization rate. When we use parameters $\alpha = 0$, the computing time of the application run in the VM1 and VM4 declines about 15%, and the computing time of the application run in the VM2 and VM3 increases 24% and 11% respectively. With the increase of α, the computing of the application run on the VM1 increases and the computing time of the application run on the VM3 and VM4 declines. With the increase of α, the weight will become more significance in resources allocation. The weight of VM1 is smallest. So its allocated resources will less with the increase of α. Similarly, the VM3 and VM4 will be allocated more resources. We also notice that the computing time of the application run on the VM2 first declines but then increases when the parameter $\alpha = 0.8.$ This is because the influence of resources utilization is more than weight when the $\alpha > 0.8.$

We notice that the CPU overall utilization is not improved. This is because all VMs have not idle resources.

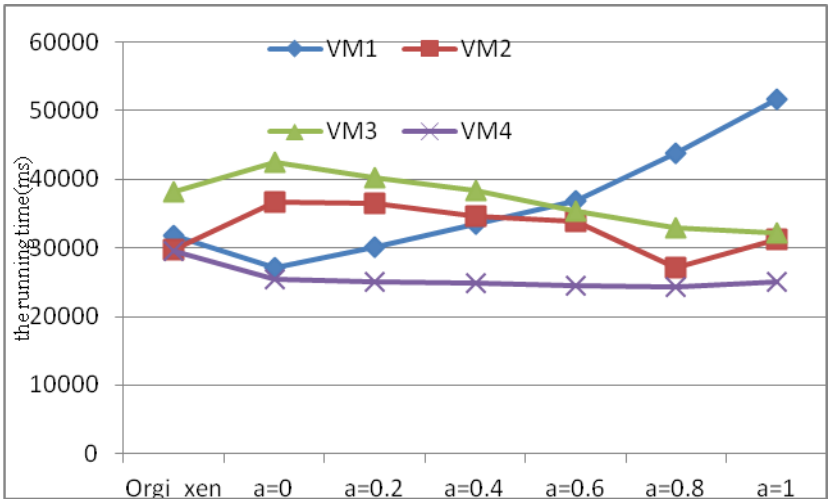


Figure 5. CPU Resources Allocation for Multiple VMs Running on the Same Physical Machine (There are no Idle Resources)

Table 2. Application Types by Different VMs

VM	VCPUs	Application type	size	Used threads
VM1	4	Matrix multiplication	1248×1248	1
VM2	2	Matrix multiplication	1554×1554	2
VM3	3	Matrix multiplication	1972×1972	3
VM4	4	Matrix multiplication	2100×2100	4

Table 3. Application Types by Different VMs

VM	VCPUs	Application type	size	Used threads
VM1	2	Matrix multiplication	1554×1554	2
VM2	3	Matrix multiplication	1554×1554	2
VM3	4	Matrix multiplication	1972×1972	3
VM4	4	Matrix multiplication	2100×2100	4

5. Related Work

The task types worked in VM may be diverse. The scheduling algorithm pays little attention on the task types of diversification would be unable to meet the requirement of some VMs, such as latency-sensitive tasks running on the VM. The authors of [14] proposed a scheduling algorithm based on the task types. To support real-time tasks in VMs, Byung Ki Kim et al. [15] proposed a scheduling algorithm that support real time tasks in VM have to finish before their deadline. Y. Hyun-jun et al. [16] considered the overhead of privileged domain Dom0 and proposed a technique to make VM responded with expected response time. The authors of [17] enhance the credit scheduler by making it full-time aware of inter-VM events and physical interrupt request events that improve the responsiveness of VMs doing mixed workloads. HwanjuKim et al. [18] presents virtual machine scheduling techniques for transparently bridging the semantic gap between the VMM and VMs. Task-aware scheduling [19] and Communication-aware scheduling [20] are both application-aware scheduling strategies.

The authors of [21] present an asymmetric virtual machine monitor (AVMM). AVMM divides underlying platforms into two asymmetric partitions: a user partition and a service partition. The user partition is assigned to most of the underlying resources and the service partition consumes only the needed resources for its tasks. Near-native performance is realized by AVMM by assigning a set of peripheral devices for exclusive use by a single user OS, as well as efficient resource management.

6. Conclusions

In this paper, we have proposed a workload-aware CPU resources scheduling method (WARS) to improve the CPU overall utilization. WARS is a periodical allocating algorithm. With each periodical round consisting of CPU resources diagnose, request more or less CPU resources to VMM and weight adjustment. WARS uses the allocated and consumed credits to diagnose CPU resources requirements of VMs, which can improve predictive ability further. Based on the predict information, WARS will adjust the CPU resources dynamically. The adjustment of CPU resources is converted into increased or decreased weight of VMs.

WARS is implemented confined to the VMM layer, without VM dependency. The experiment results show that WARS can improve CPU resources overall utilization. We also show that the performance of WARS is influenced by scheduling period. In the future, we will research on the time of scheduling period.

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

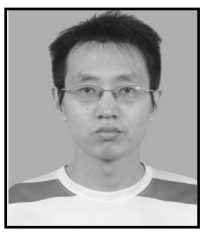
References

[1] 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.
[2] 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)**.
[3] 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)**.
[4] 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)**.
[5] 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)**.
[6] J. Hai, D. Li, W. Song and S. Xuanhua, "Dynamic Processor Resource Configuration in Virtualized Environments, Proceedings of IEEE International Conference on Services Computing, **(2011)**.
[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] C. A. Waldspurger, "Memory resource management in VMware ESX Server", Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, **(2002)**.
[10] W. Z. Zhang, H. L. Zhang, D. Zhang and T. Cheng, "Memory cooperation optimization strategies of multiple virtual machines in cloud computing environment", Journal of Computers, vol. 34, **(2011)**, pp. 2265-2277.
[11] X. D. Liu, W. Q. Tong and Y. Hou, "BSPCloud: A Programming Model for Cloud Computing", IEEE 12th International Conference on Computer and Information Technology, **(2012)**.
[12] X. D. Liu, W. Q. Tong, F. Z. Ren and L. W. Zhao, "BSPCloud: A Hybrid Distributed-memory and Shared-memory Programming Model", International Journal of Grid and Distributed Computing, vol. 6, no. 1, **(2013)**, pp. 87-97.
[13] X. D. Liu, "A Programming Model for the Cloud Platform", International Journal of Advanced Science and Technology, vol. 57, **(2013)**, pp. 75-82
[14] 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.
[15] 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.
[16] Y. H. Jun, K. Y. Rok, K. J. Young and P. Sungyong, "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.
[17] 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)**.
[18] 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.
[19] 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.
[20] H. Kim,H. Lim, J. Jeong, H.Jo and J. Lee, "Task-aware virtual machine scheduling for I/O performance", Proc. off the ACM SIGPLAN/ SIGOPS International Conference on Virtual Execution Environment, **(2009)**.
[21] Y. Z. Zhou, Y. X. Zhang, H. Liu, N. X. Xiong, A. V. Vasilakos, "A Bare-Metal and Asymmetric Partitioning Approach to Client Virtualization", IEEE Transactions on Services Computing, **(2012)**.

Authors

Hongshan Qu. He is currently an Professor of Henan Institute of Engineering. His primary research interests is network information security



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. She is currently an assistant of Henan Institute of Engineering. Her primary research interests is cloud computing