Analysis and Evaluation of the Scheduling Algorithms in Virtual Environment

Xindong You, Xianghua Xu, Jian Wan, Congfeng Jiang
Grid and Service Computing Lab
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310037, China
youxindong@hdu.edu.cn

Abstract

Virtual machine technologies currently receive great interest both in industry and research communities. And it is one of the most important technologies for the coming Cloud Computing. We surveyed the CPU scheduling algorithms in Xen and VMWare systems, and found that both of them use a distinctive VCPUs running queue for each physical CPU, which is referred to Partition Queue Model (PQM) in this paper. As a contrast, a Sharing Queue Model (SQM) of CPU scheduling algorithm is proposed. The simulation experiments results show that the Sharing Queue Model of CPU scheduling achieves better performance than the Partition Queue Model and the deduction from Queue Theory also confirms the results. Moreover, with the CPU utilization increased, the advantage of Sharing Queue Model over Partition Queue Mode is more evident in response time.

Keywords: *Virtual machine, Scheduling algorithms, queue theory*

1. Introduction

Virtual machine (VM) technologies were first introduced in the 1960s [1], but are experiencing resurgence in recent years and becoming more and more attractive to both the industry and research communities [2]. The basis for machine virtualization is the hypervisors or the Virtual Machine Monitors (VMMS) that support the creation and execution of multiple guest OS (VMs or domains) on the same platform and enforce the isolation properties necessary to make the underlying shared platform resource appear exclusive to each domain. The representative VMMs that induce negligible performance overhead are Xen[3] and VMware ESX [4]. Using VMMs to share the underlying physical resources can achieve the high resource utilization. On the other hand, with the emergence of the multi-core or multiprocessor architecture of computer systems, the PCs become more and more powerful, which promotes the tempo of the virtual machine technologies inversely. However, the evolution of the underlying hardware brings new challenges to the CPU scheduling algorithms of the VMMs. How to allocate the physical CPU (PCPU) to the Virtual CPUs (VCPU) of the VMs is one of the most important questions in the virtual technologies domain.

In this paper, we are focusing on designing, modeling and evaluation of CPU scheduling algorithms in the VMM, especially when there is more than one core or processor. The queuing theory is utilized to model and evaluate the existing and the proposed CPU scheduling algorithms, and the comparative performance results are presented.

The remainder of this paper is organized as follows: Section 2 describes and analyzes the CPU scheduling strategies of Xen and VMWare ESX. Section 3 proposes a Sharing Queue Model to replace the Partition Queue Model in the traditional virtual systems. Section 4 utilizes the Queue Theory to model the Sharing Queue Mechanism and the Partition Queue Mechanism respectively. Section 5 describes the simulation experiments and presents performance results. Finally, Section 6 concludes the paper with future work.

2. CPU scheduling mechanism in Xen and VMware ESX

Today, VMware and Xen both provide high-performance virtualization software used by a variety of customers for server consolidation. This virtualization software is usually run on symmetric multiprocessor systems. Currently, both VMware EXS server and Xen support multi-processor VMs, which allows the VMs to configure multi-VCPUs (Virtual CPU) to simulate the multi-threads run concurrently on the PCPU (Physical CPU). And CPU scheduling in Xen and VMware generally means how to allocate VCPUs of all the VMs to the PCPU. The detailed of the scheduling mechanism of VMWare EXS and Xen will be described and analyzed in the following paragraphs.

The primary reason that ESX server provides higher performance than hosted virtualization products such as VMware server is that the VMM has direct control of system resources rather than having to request them through to a host operating system [5]. In ESX server, whose architecture as shown in Fig.1, the VMM directly controls scheduling of VCPUs on PCPUs rather than having VCPUs execute as threads in a host OS that are scheduled based on the host OS CPU scheduling algorithms. Vmare ESX server uses co-scheduling for VCPUs, which means scheduling all the VCPUs in a VM onto the PCPUs simultaneously. The co-scheduling algorithms in the ESX can maintain correctness and high performance [6], but can cause inefficiency in the scheduling because it can cause PCPUs to be idle while waiting to schedule VCPUs.

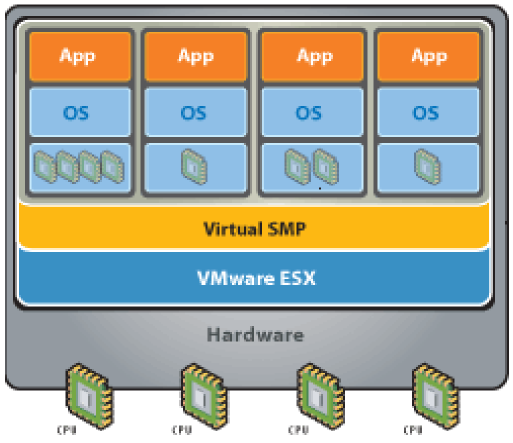


Fig.1 Architecture of the Vmware ESX

Xen offers two schedulers schemes currently, SEDF (Simple Earliest Deadline First) and the Credit-based schedulers [7]. In the current version, Credit is the default scheduler in Xen (since version 3.0) while SEDF is gradually phased out. Under Credit scheduler, each physical CPU (PCPU) manages a run queue of runnable VCPUs. The run queue of the PCPU is in ordered by VCPU's priorities which can be in one of the two sates: OVER and UNDER, where OVER represents the fair share of the VCPU's has exceeded, while the UNDER state is opposite. The CPU schedule VCPUs based on the FCFS strategy in the two sates.

Although the Credit-based scheduling algorithms in the Xen can assure the fairness to share the CPU resources among the different VMs, it is inefficient when dealing with the latency-sensitive or the real time applications. In order to address this problem, Govindan *et al* proposed a communication-aware CPU scheduling algorithm, which schedule the latency-sensitive network application preferentially [8]. Diego et al [9] modified the traditional Credit scheduling algorithm, in which another state queue BOOST is added to be scheduled before the UNDER and the OVER running queue. Moreover, in the different state queue, the VCPU is scheduled based on its value of the Credit. Meanwhile a tickle mechanism is implemented to satisfy the real time network applications. Additional, there are some researchers survey the CPU scheduling of the VMM in order to improve the performance of the network application [10][11].

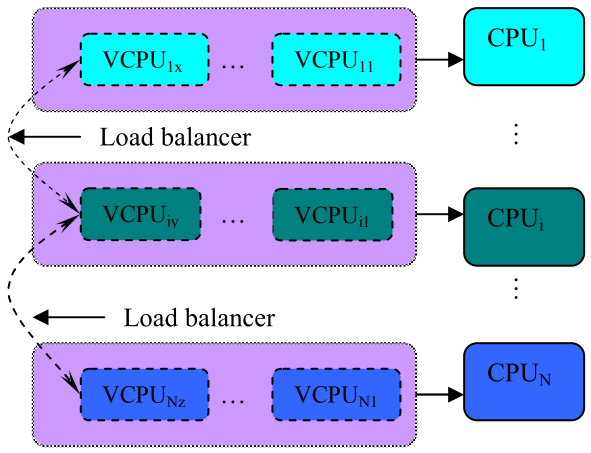
Above all of the related CPU scheduling algorithms can achieve good performance under a certain situations, and the physical CPU (PCPU) has a distinctive partition running queue when scheduling the VCPUs. Due to the disadvantages of the partition running queue of a PCPU, a Sharing Queue Model (SQM) is proposed to improve the performance of the CPU scheduling algorithm, which is described in the following sections.

3. Proposal for the CPU Scheduling in the Virtual Environment

As described above, in the scheduling algorithms of the representative VMM, all PCPUs have a partition queue for the runnable VCPU, when there is more than one Physical CPU (PCPUs). Although the Credit scheduling algorithm in the Xen has a load balancer to balance the number of the VCPUs among the PCPUs, the mechanism that all of the PCPUs sharing only one VCPU running queue will achieve better performance than the mechanism with partition one even when the loads among the distinctive running queue are balanced [12].

In order to compare the performance behave of the two different queuing mechanisms detailedly, the queue model of the Credit scheduling algorithm in Xen is chosen as contrast to our proposal Sharing Queue Model (SQM).

The queue models of the Credit scheduling algorithm in Xen and the proposed scheduling algorithm are shown as the Figure 2 and Figure 3 respectively. Assume there are the number of Physical CPUs is N, and the number of the VCPUs of all the VMs is M.



VCPU$_{1x}$ … VCPU$_{11}$
CPU$_1$
Load balancer
VCPU$_{iy}$ … VCPU$_{i1}$
CPU$_i$
Load balancer
VCPU$_{Nz}$ … VCPU$_{N1}$
CPU$_N$

Fig.2 Partition Queue Model of the Credit scheduling algorithm

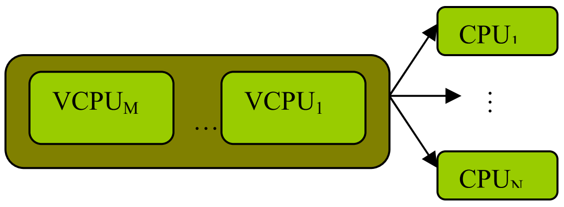


Fig.3 Sharing Queue Model of the proposed scheduling algorithm

Although the modification on the Credit scheduling algorithm of the Xen is slight, the performance improved is evident, which will be testified in the following sections.

4. Modeling the Scheduling Algorithms

It is well known that the partition queue model is in best situation when the number of the VCPUs in any queue is all the same. In this section, Queue Theory is utilized to analyze performance behave of Partition Queue Model (PQM) and Sharing Queue Model (SQM), when the PQM is in the best situation. Before deduce the performance parameters of the models, the following assumptions are made:

1) All of VCPUs arrival process conforms to the Possion distribution, which is usually true for the VMM systems where the system is open with the property of that the arrival of the current VCPU is independent to the VCPUs having arrived. And we assume the arrival rate is λ.

2) The discipline that PCPU serves the VCPUs conforms to the exponential distribution, and the service rate is μ.

3) The VCPUs in Partition Queue Model (PQM) and Sharing Queue Model are all in FCFS order (First Come First Serve), which is true for the Credit scheduling algorithm in Xen.

4) In order to make the two types of the queue be comparable, the sum of the arrival rate of all of the partition running queues is equal to arrival rate of the proposed sharing one. We can deduce the VCPUs arrival rate of one of the partition queues is λ/N when the number of CPU is N and the arrival rate of Sharing Queue Model is λ.

Based on the above assumption, the partition and the sharing queue model will be discussed and analyzed in detail in the following subsections.

4.1 The Partition Queue Model

The Credit scheduling algorithm in Xen can be modeled as N partition queue when the number of PCPU is N. The model can be shown as Fig. 4.

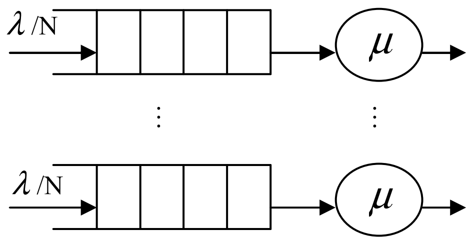


Fig.4 Partition Queue Model (PQM)

The response time of the scheduling system when using PQM is determined by the response time of any queue, because the input parameters are all the same. According to the queue theory [12], the performance parameter of response time can be calculated by formula (1).

$$T_{q1} = \frac{1/\mu}{1-\rho}, \tag{1}$$

where $\rho = \dfrac{\lambda/N}{\mu} = \lambda/N\mu$, which is the CPU utilization in the context of this paper.

And the waiting time of VCPUs is determined by the formula (2)

$$T_{w1} = \frac{\lambda/N}{1-\rho} - \frac{1}{\mu} = \frac{\rho}{\mu(1-\rho)} \tag{2}$$

There are other performance-related parameters can be deduced according to [13].

4.2 The Sharing Queue Model

The sharing queue of the proposed scheduling is modeled as the Figure 5 shown.

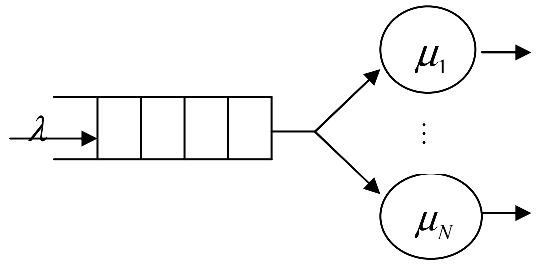


Fig.5 Sharing Queue Model (SQM)

By the same means, the performance parameter of response time is determined by the formula (3)

$$T_{q2} = \frac{q_2}{\lambda}, \quad (3)$$

where q_2 is calculated by the following formula:

$$q_2 = N\rho + \frac{(N\rho)^N}{N!} \frac{\eta_0}{(1-\rho)^2} \quad (4)$$

And η_0 is determined by the formula (5):

$$\eta_0 = \left[\sum_{k=0}^{N-1} - \rho \frac{(N\rho)^k}{k!} + \frac{(N\rho)^N}{N!} \frac{1}{1-\rho} \right]^{-1} \quad (5)$$

The utilization of the CPU ρ is the same as the partition queue model:

$$\rho = \frac{\lambda/N}{\mu} = \lambda/N\mu$$

The other parameter that represent probability of the VCPU must wait in the queue is calculated by the formula (6):

$$P[waiting] = \sum_{k=N}^{\infty} \eta_k = \frac{(N\rho)^N}{N!} \frac{\eta_0}{1-\rho} \quad (6)$$

When coming to the performance parameter of the response time, given the same value of the parameters ρ, μ and N, we can deduce that the $T_{q1} \geq T_{q2}$ always. That is to say when the number of the VCPUs, the arrival rate of the VCPUs, the number of the physical CPUs and the service rate of the PCPUs are all the same, the sharing queue model is more efficient in response time.

In the following section, the simulation experiments will be run to confirm the above conclusion.

5. Simulation Experiments and Results

In this section, we simulate the scheduling algorithms when using different queue models through Java program. The input parameters of the models are λ, u and N.

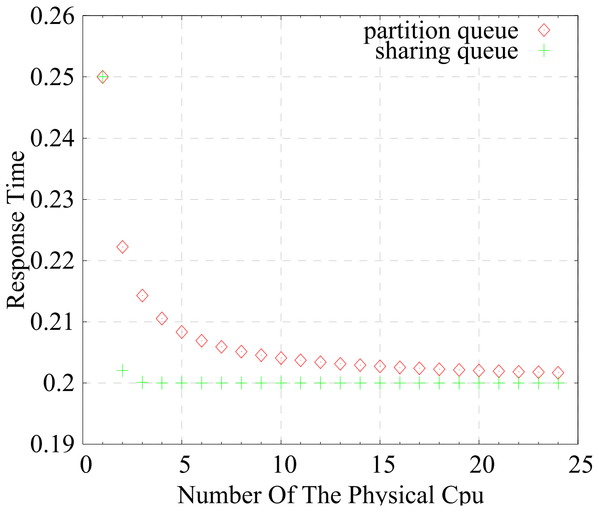


Fig.6 Response time of the two type queue as the function of the number of PCPUs when $\lambda/\mu = 0.2$

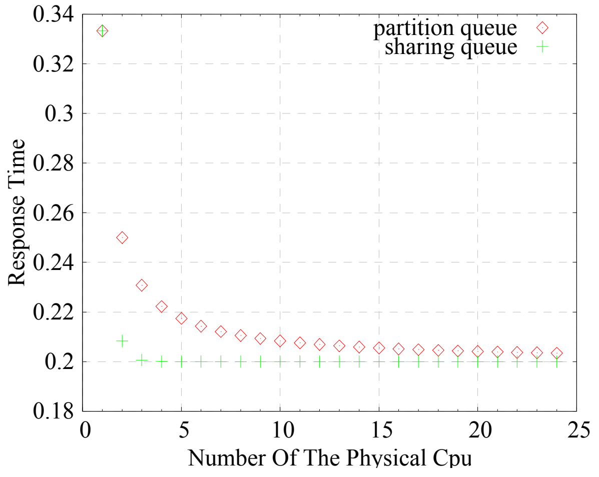


Fig.7 Response time of the two type queue as the function of the number of PCPUs when $\lambda/\mu = 0.4$

The environment to run our simulation experiment is: a Dell OptiPlex 755 with a 2.33 GHz Interl Core, 2CPU E6550, 3GB of RAM, 160GB hard disk, and one 1000M Ethernet cards.

We run the simulation experiments with a certain increment of the CPU utilization. And the results are given by the following Figure 6~11.

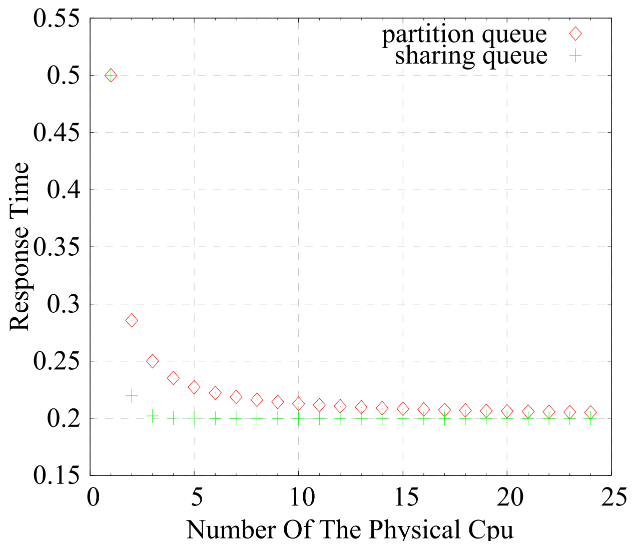


Fig.8 Response time of the two type queue as the function of the number of PCPUs when $\lambda/\mu = 0.6$

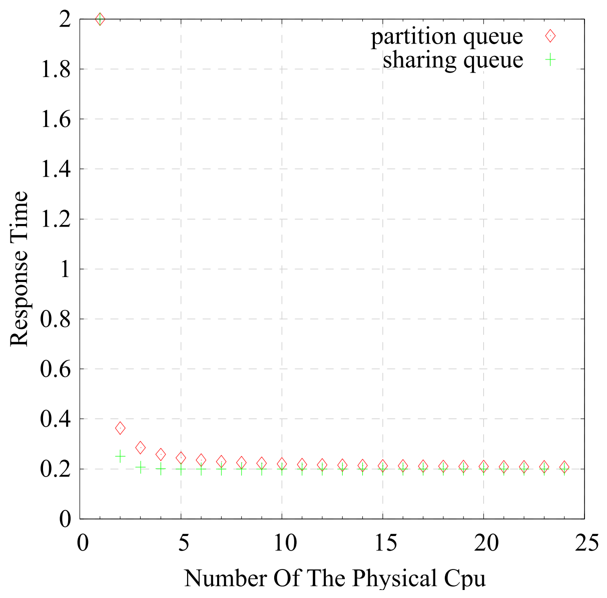


Fig.10 Response time of the two type queue as the function of the number of PCPUs when $\lambda/\mu = 0.9$

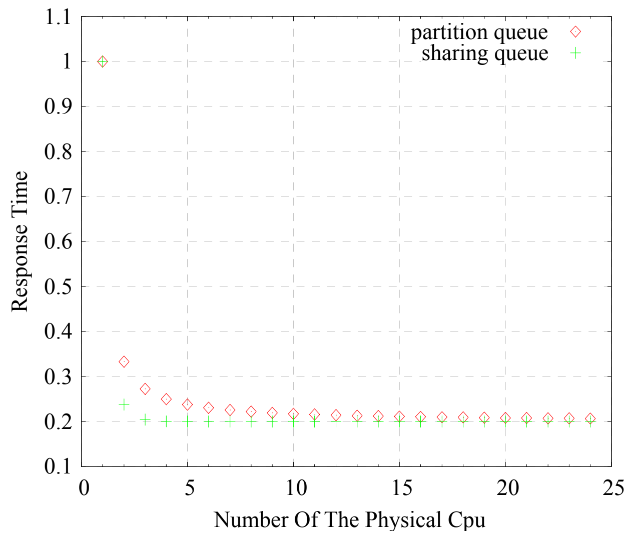


Fig.9 Response time of the two type queue as the function of the number of PCPUs when $\lambda/\mu = 0.8$

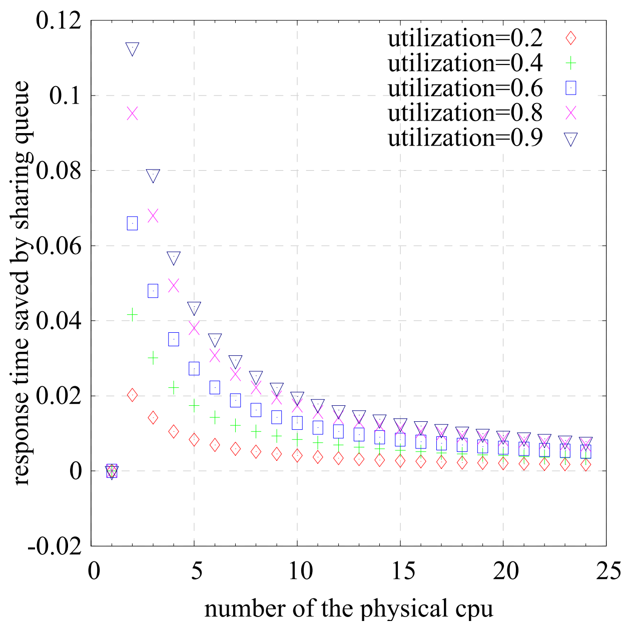


Fig.11 Response time saved by the sharing queue as the function of the number of physical CPUs, when the CPU utilization is 0.2, 0.4, 0.6, 0.8, 0.9 respectively.

From the simulation results shown in Figures 6~11, we can see that the sharing queue model achieve better performance in terms of response time than the partition one. Moreover, with the utilization increased, the response time saved by the sharing queue model is more evident (Figure 11). We can conclude that with the development of the multi-core or multi-processor architecture and the increased utilization when using VMMs to consolidate servers, the sharing queue model implemented in the CPU scheduling algorithms will receive more benefits.

6. Conclusions and Future Work

In this paper, we survey the CPU scheduling algorithms in the Xen and VMWare ESX when there are more than one processor underlying. Both the Xen and VMWare ESX implemented the scheduling algorithm with Partition Queue Model, which means that each physical CPU has its own VCPUs running queue. We employ the queue theory to model the CPU scheduling algorithm. The results deduced by the formulas and the results produced by the simulation programs imply that all the physical CPUs sharing only one running queue (Sharing Queue Model SQM) can achieve better performance than that a physical CPU with a partition queue (PQM) in response time.

In the future, we will actually implement the sharing queue model to the CPU scheduling algorithms in the open source based Xen project. Afterward, we will run the different workloads on the different VMs to evaluate the traditional CPU scheduling algorithms and our proposed one.

7. ACKNOWLEDMENT

This work is supported by State Key Development Program of Basic Research of China (Grant No. 2007CB310900) and National Science Foundation of China (Grant No. 60873023). Startup Foundation of Hangzhou Dianzi University (KYS055608033, KYS055608058)

8. References

[1] R.P.Goldberg. Survey of Virtual Machine Research.Computer, pages 34-45, June 1974.
[2] M. Rosenblum and T. Garfinkel. Virtual Machine Monitors: Current Technology and Future Trends. IEEE Computer, pages 39-47, May 2005.
[3] B.Dragovic et al. Xen and the Art of Virtualization. In Proceeding of the SOSP, 2003.
[4] The VMWare ESX Server. http://www.vmware.com/products/exs.
[5] VMWare. Virtualization Overview. http://www.vmware.com/pdf/virtualization.pdf
[6] VMWare. Performance Tuning Best Practices for ESX server 3. http://www.vmware.com/pdf/vi_performance_tuning.pdf, 2007.
[7] Credit scheduler. http://wiki.xensouce.com/xenwiki/CreditScheduler
[8] Sriram Govindan, Arjun R. Nath, Amitayu Das, Bhuvan Urgaonkar, Anand Sivasubramaniam: Xen and co.: communication-aware CPU scheduling for consolidated xen-based hosting platforms. VEE 2007: 126-136
[9] D. Ongaro Alan L. Cox and S. Rixner Scheduling I/O in Virtual Machine Environments, VEE 2008, March 5, 2008.
[10] A. Menon, A.L.Cox, and W. Zwaenpod. Optimizing network Virtualization in Xen. In Proceedings of the USENIX Annual Technical Conference, June 2006.
[11] J. Sugerman, G. Venkitachalam, and B.Lim. Virtualizing I/O devices on Vmware Workstation's Hosted Virtual Machine Monitor. In Proceedings of the USENIX Annual Technical Conference, June, 2007.
[12] C.Lin, Performance evaluation of the computer network and the computer system. Published by Tinghua University.2001, 53-71