International Journal of Security and Its Applications
Vol. 6 No. 2, April, 2012

Efficient Virtual Machine Scheduling Exploiting VCPU Characteristics

Byung Ki Kim*, Jin Kim, Young Woong Ko

Department of Computer Engineering, Hallym University, Chuncheon, Korea
{bkkim,jinkim,yuko}@hallym.ac.kr

Abstract

Xen supports CPU-intensive domains fairly well, however, it has drawback for supporting I/O domains because I/O domain requires low latency data processing. In this paper, we propose a VCPU scheduling framework exploiting VCPU characteristics for supporting accurate resource measuring scheme. Our approach categorizes the VCPU characteristics into three model, CPU-intensive, disk-intensive and network-intensive, respectively. We designed and implemented a virtual machine monitoring tool for predicting the behavior of each domain.

Keywords: Xen, Load balancing, VCPU Shaping, Xentrace, Virtulization

1 Introduction

Virtualization allows multiple commodity operating systems to share a single physical machine for supporting multiple guest operating systems. Server virtualization is widely used and also embedded system[1] and ECU[2] field adapt virtualization technology. To enhance the performance of virtual machine, efficient resource allocation on a virtual machine is one of the key issues. Unfortunately, the complexity of virtualization system causes additional resource management challenges. To meet resource requirements for each domain, virtual machine monitor need to observe the exact behavior of each domain and allocates proper amount of resources in a timely manner. In this paper, we propose a resource monitoring tool that analyzes scheduling information from Xentrace information and determines scheduling parameters. Our goal is to improve the overall resource utilization of virtual machine by considering VCPU characteristics. To accomplish this goal, we designed and implemented monitoring tools and VCPU shaping mechanism for predicting the resource usage of each domain. Virtual machine monitor must support both CPU intensive and I/O intensive domains. Although Xen supports CPU-intensive domains fairly well, it has drawback for supporting I/O domains because I/O domain requires low latency, high bandwidth and must provide isolated execution. Xen scheduler needs to distinguish which VCPU is a I/O-intensive domain because I/O domains require careful resource allocation scheme. In this work, we can shape the characteristics of each VCPU as CPU-intensive,

* This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0005442)

disk-intensive and network-intensive by considering scheduling events from the monitoring tool. The rest of paper is organized as follows. In section 2, we present the profiling mechanism for Xen virtual machine and VCPU characterizing mechanism. Section 3 presents the profiling result and analyzes the scheduling events. We present the related work on section 4 and finally conclude our work in section 5.

2 System architecture

To predict the characteristics of each domain, we have to guess if a VCPU is I/O-intensive or CPU-intensive considering the behavior of each domain. There are some evidences to figure out I/O domain. Generally, I/O-intensive domains are quickly blocked by I/O requests from tasks on a guest operating system. When a domain has a I/O operations, Xen scheduler blocks this domain and gives control to domain0 for processing I/O requests from guest domains. Therefore, each domain consumes short period of CPU time less than average 1ms. By considering these features, we can easily guess if a domain is I/O-intensive or CPU-intensive. In this paper, we devised a VCPU monitoring tool which predicts if a domain is I/O-intensive or not by tracing system information. To predict the domain characteristics is very important in a virtualization environment because VMM scheduler can support the requirement of each domain. For example, if a domain is I/O-intensive then the task on the domain has to finish I/O requests within a very short time with low latency. If a domain is CPU-intensive then this domain is not related to low latency tasks. Furthermore, hypervisor will allocate system resources to guest domains accurately.

2.1 System information monitoring scheme

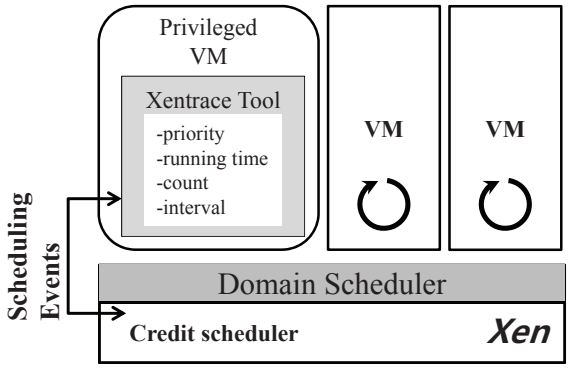


Figure 1. Scheduling monitoring architecture using xentrace

In this work, we modified the Xentrace tool and made system monitoring tool for predicting a domain. Figure 1 shows the system architecture of the proposed system. We implemented the scheduling monitor that observes scheduling events and analyzes the characteristics of each domain using Xentrace[3] mechanism. Xentrace collects scheduling events and analyzes it using trace buffer information. Xentrace is an event logging tool that captures trace buffer data generated from Xen hypervisor and domains. With this scheduling events, we can evalueate the characterics of each domain. Following code shows the binary format of Xentrace.

CPU(uint) TSC(u64) EVENT(u32) D1 D2 D3 D4 D5 (all u32)

CPU(uint) depicts processor number in a multiprocessor machine, *TSC* means time stamp counter that stores the time when the event is captured. *EVENT* represents the ID of the event. In Xentrace, there are lots of events defined for notifying system status. *D1* to *D5* are the event parameters used for event specific purpose. The Xentrace events is composed of binary format and translated into ASCII format using Xentrace_format utility. In this work, we modified Xen hypervisor to generate scheduling event, so we used following code to capture scheduling information.

TRACE_4D(TRC_SCHED_LOG, prev->domain->domain_id, prev->pri, now - prev->runstate.state_entry_time, NOW())

TRC_SCHED_LOG means event ID, *domain_id* is a ID of current domain that is scheduled in virtual machine, *prev->pri* is a priority of domain, *now - prev->runstate.state_entry_time* is a execution time and *NOW()* is current time when the event is captured. We measured the number of scheduling count , which means how many times the VCPU scheduled and how long it consumed CPU time for each priority. Also we collected the scheduling intervals to figure out domain characteristics. For example, CPU-intensive domains generally consumes its credits in UNDER priority state, I/O-intensive domains frequently switched to BOOST priority and consumes CPU slice intensively, however the amount of CPU usage is not significant.

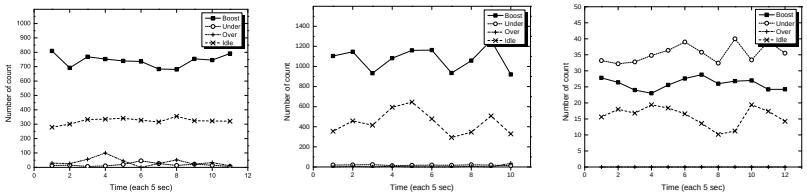
2.2 VCPU prediction scheme

In this section, we describe the mechanism of tracing domains on Xen virtual machine and figure out the characteristics of VCPU. In Xen virtual machine, the Credit scheduler is a default scheduler featuring automatic load balancing of virtual CPUs across physical CPUs. Each VM is assigned scheduling information using weight and cap value. If the cap is 0 then the VM receives extra CPU. Otherwise, it limits the amount of CPU that a domain receives from VM. Credit uses 30 millisecond time slices for CPU allocation. Every 30 ms, the priorities of all runnable VMs are recalculated. Credit supports global load balancing on multiprocessor platform. In Credit scheduling algorithm, a domain may have three states for represent VCPU states when they are scheduled. UNDER state means the domain has enough credits and OVER means a domain consumed all of its credit. In this case, the domain consumes CPU slice under work conserving mode. There is an additional scheduling state called BOOST. BOOST state is higher priority than UNDER and usually used when an idle domain receives a virtual interrupt. If we can count the number of each state and the processing time of each state, we can roughly predict the characteristics of each domain. In this work, we made preliminary experiments for predicting each domain and find several simple rules for guessing domain characteristics. First, we assume that a domain is CPU-intensive if the number of UNDER scheduling count is dominent and bigger than the number of BOOST. And the running time of UNDER is longer than OVER and BOOST. For predicting a disk-intensive domain, we assume that a domain is disk-intensive if the number of BOOST scheduling count is dominant and the number of OVER and UNDER is very small. Furthermore, the disk-intensive domain should have long idle time. The network-intensive domain is very similar with disk-intensive one, therefore, we assumes network-intensive and disk-intensive is the same.

3 Experiment results

The purpose of experiment is to check if the monitoring tool can accurately predict the behavior of each domain. We configured four domains and pinned on a PCPU not to move to other PCPU by load balancing mechanism. Three domains have CPU intensive job which consumes 25% of CPU share by periodically calculating MD5 hash function. We made additional three types of domains(CPU-intensive, disk-intensive and network-intensive) and inserted these domains one by one on test platform.

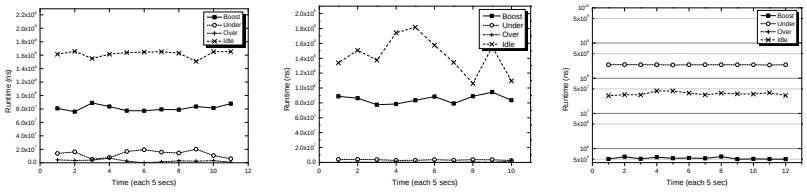
- CPU-intensive domain : calculate 1.2MB MD5 hash every 30ms (25% of CPU share)
- Disk-intensive domain : read 4GB file with 8k blocks with dd command on a Linux system
- Network-intensive domain : measure the web-server performance using httperf with 240000 connections (400 connections for a second) for 60 seconds



(a) httperf (b) dd (c) CPU

Figure 2. Scheduling count for each domain

Figure 2 shows the scheduling count for each state. As we can guess with our monitoring system, httperf domain shows lots of BOOST state count compared with CPU-intensive one. The dd domain is very similar with httperf one. CPU-intensive domain shows that the number of BOOST is very small and the number of OVER count is very big compared with BOOST.



(a) httperf (b) dd (c) CPU

Figure 3. Running time for each domain

Figure 3 shows the running time for each state. In httperf domain, the running time of BOOST state is very long compare with OVER and UNDER state, furthermore, IDLE time is dominant in network-intensive domain. The dd domain is very similar with httperf one. CPU-intensive domain shows that the running time of BOOST is very short and almost all of the running time is comsumed by UNDER state.

4 Related works

There are lots of works in virtualization system, however their works are limited on I/O performance enhancement or scheduling policy. Performance analysis and resource allocation have been well conducted on the Xen VMM[4]. Many researches are roughly focused on improving I/O performance, network response, CPU allocation, resource monitoring and real time guarantee. Three scheduling mechanisms are well analyzed in terms of I/O performance and CPU utilization according to different parameters in various workloads [5]. Analyzing I/O bound tasks is very complicate because isolated driver domain (IDD) processes I/O processing on behalf of VMs[6]. To improve I/O processing like disk I/O and communication via NICs, SEDF-DC introduces a performance monitoring and profiling tool that reports VPU usage of VMs and VM scheduler with feedback [7]. Govindan introduced communication-aware VM scheduling mechanism for high throughput network intensive workloads. Their scheduling mechanism is picking a domain that is likely to experience the most overall reduction in scheduling delay. The domain that has received the most number of packets has the highest priority [8]. Vsched proposed by Lin and Dinda[9] is a user-level scheduling tool using a periodic real-time scheduling model. Vsched is implemented for a type-II virtual machine monitor that does not run directly on the hardware but rather on the top of a host operating system. Therefore, the domains are executed as a process inside the host operating system. Vsched provides an EDF (Earliest Deadline First) scheduler using the SCHED FIFO scheduling class in Linux. Their approach is quite straightforward to describe real-time workloads because a domain is regarded as a process. However, to support real-time workloads accurately, the host operating system should support real-time characteristics such as a fine-grained preemption mechanism, prevention of priority inversion, and fast interrupt handling, among others. Our work is very unique compared with previous results because we can schedule lots of domain by predicting the behavior of each domain. We believe that our work is very simple and easily ported to various of virtualization system.

5 Conclusion

It is very important to characterize the behavior of virtual machines to improve virtualization system performance. Virtual machine monitor must support both CPU intensive and I/O intensive domains, Xen supports CPU-intensive domains fairly well, but I/O-intensive one has difficulties for supporting predictable resource allocation. In this paper, we propose a virtual machine characterizing scheme using Xentrace which analyzes scheduling information from Xentrace information and determines scheduling parameters. Our goal is to improve resource management efficiently considering VCPU characteristics. Experiment result shows that our system can predict the behavior of each domain very well.

References

[1] KD. Kim, YC. Kim and JH. Kim, A study on reliability evaluation for embedded software. The Journal of IWIT. 9, 3 pp. 209–215 (2009)

[2] JY. Ahn, YS. Kim, SH. Kim and KI. Hur, A study on voice recognition pattern matching level for vehicle ecu control. The Journal of IWIT. 10, 1 pp. 75–80 (2010)

[3] D. Gupta, R. Gardner and L. Cherkasova, Xenmon: Qos monitoring and performance profiling tool. Hewlett-Packard Labs, Tech. Rep. HPL-2005-187. (2005)

[4] 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. 37, 5 pp. 164–177 (2003)

[5] L. Cherkasova, D. Gupta and A. Vahdat, Comparison of the three cpu schedulers in xen. Performance Evaluation Review. 35, 2 pp. 42 (2007)

[6] H. Kim, H. Lim, J. Jeong, H. Jo and J. Lee, Task-aware virtual machine scheduling for i/o performance. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, ACM (2009)

[7] D. Gupta, L. Cherkasova, R. Gardner and A. Vahdat, Enforcing performance isolation across virtual machines in xen. In: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware, Springer-Verlag New York, Inc. (2006)

[8] S. Govindan, A. Nath, A. Das, B. Urgaonkar and A. Sivasubramaniam, Xen and co.: communication-aware cpu scheduling for consolidated xen-based hosting platforms. In: Proceedings of the 3rd international conference on Virtual execution environments, ACM (2007)

[9] B. Lin and P. Dinda, Vsched: Mixing batch and interactive virtual machines using periodic real-time scheduling. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, IEEE Computer Society (2005)