Low Latency Task Scheduling Mechanism on Xen Virtual Machines

Young Jun Yoo[1], Jin Kim[1], Sun Jung Kim[2], Young Woong Ko[1]

[1.] Department of Computer Engineering, Hallym University, Chuncheon , Korea
[2.] Department of Ubiquitous Computing, Hallym University, Chuncheon , Korea
{willow72, jinkim, sunkim, yuko}@hallym.ac.kr; yuko@hallym.ac.kr

Abstract: In virtual machine environments, latency-sensitive tasks are difficult to support in a timely manner, especially when many domains have boosted priorities. Our approach considers where multiple domains compete CPU resource with several I/O intensive domains. This results in increased latency for time-sensitive domains, because the credit scheduler is not aware of the urgency of tasks within different guest domains. In this paper, we present a low-latency scheduling mechanism that uses VCPU shaping. The key idea of this paper is to adapt VCPU characteristics into a scheduling policy by investigating and measuring the behavior of each domain. Our experiment results show that the proposed method effectively allocates CPU resources for low-latency tasks.

[Young Jun Yoo, Jin Kim, Sun Jung Kim, Young Woong Ko. **Low Latency Task Scheduling Mechanism on Xen Virtual Machines.** *Life Sci J* 2014;11(7):721-725] (ISSN:1097-8135). http://www.lifesciencesite.com. 105

Keywords: Xen; realtime; scheduler; Multi-Boost; VCPU

1. Introduction

In this paper, our key idea is to support a real-time framework for a Xen virtual machine. A virtual machine is a software implementation of a machine that executes programs like a physical machine and is separated into two categories: a system virtual machine and a process virtual machine. The system virtual machine provides a system platform that supports the execution of the complete operating system. In contrast, the process virtual machine is designed to run a single program, which means that it supports a single process. Traditional operating systems should be executed on a hardware platform; however, in virtualization environments, multiple operating systems can be consolidated into one hardware platform. Therefore, several operating systems running on a virtual machine must share hardware resources.

Virtualization is an emerging technology that provides scaling solutions to computer systems efficiently and reduces management costs by running multiple operating systems simultaneously on a single computer [1][2]. To take advantage of this resource-saving potential, considerable effort has been made to exploit virtualization technology in a realtime system. A variety of possible requirements must be satisfied when handling realtime applications, such as streaming servers, telephony servers, and distributed systems [3]. However, providing realtime guarantees is not easy because of the difficulties in accurately predicting the CPU requirements of each domain. Many research groups are actively working on supporting latency-sensitive workloads in a virtualization system, and many studies on virtual machines are broadly focused on improving I/O performance, network response, CPU allocation, resource monitoring, and realtime guarantee [4][5].

In this paper, we present a virtual machine scheduling scheme using a VCPU shaping mechanism. In particular, our approach considers where multiple domains compete CPU resource with several I/O intensive domains. This results in increased latency for time-sensitive domains, because the credit scheduler is not aware of the urgency of tasks within different guest domains. The proposed system can give realtime priority to a latency-sensitive domain by predicting the resource activity of the VCPU. Our mechanism is based on a VCPU shaping scheme that predicts the type of each VCPU running on each physical CPU.

The rest of this paper is organized as follows. The next section presents the related works. In section 3, we will discuss the design and implementation of the proposed system. Section 4 explains the experimental results. In section 5, we will conclude the paper and present future works.

2. Related Works

Intensive research has been carried out to support hard real-time guarantees for embedded hypervisors. Examples of these approaches have been proposed by many companies, such as OK Labs (Microkernel), Real-time Systems (RTS Real-Time Embedded Hypervisor) and LynuxWorks (small separation kernel). The characteristics of an embedded hypervisor are its small size, fast type-1 hypervisor with support for multiple VMs, and low-latency communication between system components. Typical real-time solutions associated with these approaches use a fixed hardware partition technique and divide non-real-time parts and real-time parts for over-all hardware resources, including the processor, disk, and memory. The real- time part supports servicing an application's real-time needs and the non-real-time part provides functionalities for running a general-purpose operating system (GPOS), such as Linux or Microsoft Windows. With these dual deployment strategies, the real-time part processes real-time workloads and the GPOS part is responsible for data processing, displaying, and non-real-time workloads. In this approach, supporting soft real-time applications of the type that are widely used in a GPOS is difficult because the programs should be rewritten using the API of a real-time hypervisor. Even more unfortunate problems arise when we deal with mixed workloads that are latency-sensitive, those that come in a batch, and those that are highly interactive and occur in real time.

There are other approaches for handling real-time workloads in virtualization environments by providing real-time scheduling in a hypervisor. Lee et al[4] suggest a soft real-time scheduler for the Xen hypervisor by modifying a Credit scheduler to calculate scheduling priorities dynamically. They define a laxity value that provides an estimate of when tasks need to be scheduled next. When a VCPU of a real-time domain is ready, it is inserted where its deadline is expected to be met. This approach allows low-latency tasks to be executed in a timely manner. However, it too cannot guarantee real-time workloads because it does not provide an admission control mechanism. Therefore, if the workloads increase, it cannot meet the deadline of real-time tasks.

"Vsched," proposed by Lin and Dinda[5], 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 top of a host operating system. Therefore, the domains are executed as a process inside the host operating system. Vsched provides an Earliest Deadline First (EDF) 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 fine-grained preemption mechanisms, prevention of priority inversion, and fast interrupt handling, among others.

3. System Overview

Virtualization allows multiple commodity operating systems to share a single physical machine. Efficient resource allocation on virtual machines is a key issue in enhancing virtual machine performance. Unfortunately, the complexity of virtualization systems presents additional resource management challenges. To meet the requirements for each domain on a virtual machine, the VMM needs to observe the exact behavior of each VCPU and hardware resources and allocate sufficient resources in a timely manner. Determining scheduler parameters is non-trivial because it is hard to characterize a VM's behavior. We propose a monitoring tool that analyzes scheduling information from Xentrace[6]. Our goal is to improve response time for low-latency domains considering VCPU characteristics. To achieve this goal, we designed and implemented monitoring tools and a VCPU shaping mechanism to predict the CPU usage of each domain.

The VMM must support both CPU-intensive and I/O-intensive domains. Although Xen supports CPU-intensive domains fairly well, it is not as efficient at supporting I/O domains because they require low latency, high bandwidth and it has to provide isolated execution regardless of the workload effects of the other domains.

For these reasons, the Xen scheduler needs to distinguish the I/O domains because they require a different resource allocation approach. There are several pieces of evidence that help to distinguish an I/O domain. I/O domains tend to block quickly. When a domain requests I/O operations, the Xen scheduler blocks this domain to process I/O request on Dom0. The I/O domain consumes a short period of CPU time—less than an average of 10 ms. Determining the shape of the virtual machine is important in virtualization environments. The key to determining a domain's performance is a domain scheduling mechanism. A hypervisor allocates server resources to domains according to the scheduling mechanism or user preference. In this study, we closely examine a hypervisor scheduler. We detail domain scheduling information in terms of how to schedule various workloads. Using this information can help load-balancing or controlling CPU allocation.

To show the scheduling latency over the xen credit scheduler, we measured packet arrival latency at time sensitive domain using ping. Five domains are running over the Xen : Dom0, domain 1 for latency-sensitive I/O domain, domain 2 to 4 for CPU-intensive task, that is calculating MD-5 hash for every 10 ms(30 \% CPU load). Then, external server receives the ping from latency-sensitive I/O domain (ping interval is fixed to 10ms). Our server has the 2 physical cores so the domains' VCPUs are migrating across the physical cores by the credit load balancing.

Figure 1 shows the cumulative latency distribution for 3,000 packets of arrival time on the external server from the guest domain. When CPU-intensive domains block and wake periodically, their scheduling priority is BOOST. In this worst case, the latency-intensive domains have to contend with the CPU-intensive domains. As CPU utilization by the multi-boosted domains increases, the latency for time-sensitive domains increases.

This delay is caused not only by delayed domains but also by Dom0. Figure 1 presents detailed examples of delay caused by multi-BOOST. Here, time sensitive domains are about to send a packet to the external server. Time sensitive domains are delayed by boosted domains. Also, Dom0 is delayed by other boosted domains. In this multi-BOOST situation, to improve I/O latency, Dom0 has to be compartmentalized in to other boosted domains.

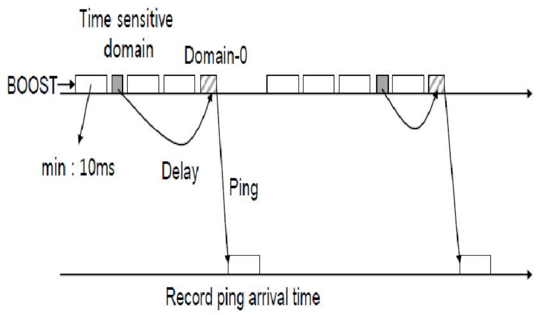


Figure 1. Multi-boost problem in Xen virtual machine

To shape the characteristics of each domain, we have to guess whether a VCPU is I/O-intensive or CPU-intensive by analyzing the behavior of each domain. Generally, I/O-intensive domains are quickly blocked by I/O requests from tasks in a guest operating system. When a domain has I/O operations, the Xen scheduler blocks this domain and gives control to domain0 to process I/O requests from guest domains; therefore, each domain consumes very little CPU time. Predicting the characteristics of each domain is important in a virtualization environment, because the VMM (Virtual Machine Monitor) scheduler can exploit VCPU information very effectively, if necessary. In this paper, we adapt the VCPU shaping information for predicting latency-sensitive workloads in a domain.

To shape each VCPU type, we measured the scheduling count, which means the number of times that the VCPU scheduled a task and how much CPU time the task consumed. Further, we collected the scheduling count for each priority (UNDER, OVER, and BOOST) during the given timespan. For example, CPU-intensive domains generally consume the allotted CPU cycles with UNDER priority tasks, I/O-intensive domains frequently switch to the BOOST priority and consume CPU cycles intensively.

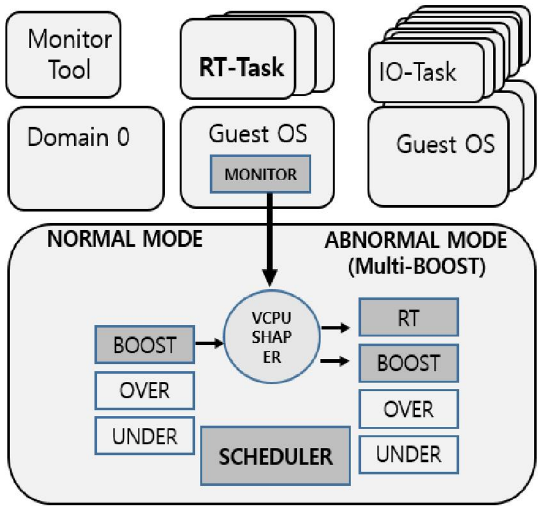


Figure 2. System architecture for supporting low latency task scheduling

Figure 2 shows the overall architecture of the proposed system, which is composed of three parts. The monitor module in a domain determines whether realtime tasks miss their deadline. If a deadline is missed, the monitor module of the VM sends an urgent message to the VMM scheduler through the hypercall interface. In this work, we introduced a new priority RT (RealTime) to denote the highest priority and created an additional hypercall interface that requests RT priority from the VMM scheduler.

In this paper, we assume that Multi-Boost is an abnormal state where lots of I/O requests and BOOST priority tasks dominate overall system resources. We assume that the Multi-Boost situation occurs when many BOOST priority domains exist and when BOOST priority domains use more CPU cycles than UNDER- and OVER priority domains. In these situations, latency-sensitive domains miss the deadlines of tasks. To overcome this situation, we divide the system state into two states, normal and abnormal state, respectively. We protect latency-sensitive workloads from general non-realtime workloads using the RT priority. When an urgent message is received, *change_rt* hypercall is generated for the scheduler to change the priority of the domain to RT in order to avoid resource competition with BOOST priority domains.

Finally, in hypervisor, scheduler check whether the system is normal state or abnormal state by using VCPU Shaper. VCPU Shaper specifies domains into two categories, CPU-intensive domain and I/O-intensive domain. If there are lots of I/O-intensive domains with heavy CPU consumption, then low-latency tasks are turn to realtime domain.

With this approach, we can achieve low latency task scheduling in virtual machine.

4. Experiment Results

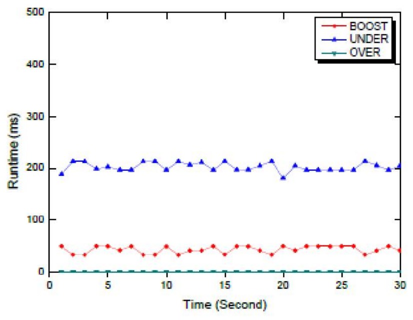
In this experiment, we made a various experiments to draw all the aspects of the proposed system capabilities. Table 1 shows experiment environment including hardware and software configurations.

Table 1. Experiment Environment

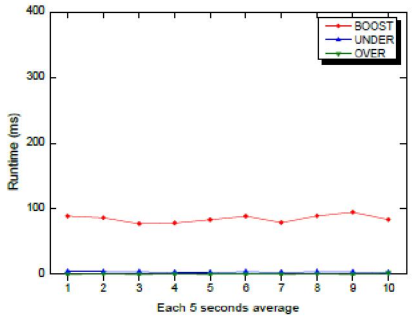
PLATFORM SPECIFICATION		
Xen hypervisor platform		
H/W	CPU	Intel Xeon E5520 2.27GHz, Quad
	Cache	L2: 256KB * 4, L3: 8MB
	Memory	DDR3 2G * 3
	Network	Gigabit ethernet
	Disk	Seagate 1TB 7200 RPM * 3
S/W	Hypervisor	Xen 3.4.2
	Dom0 OS	CentOS 5 2.6.18.8-xen0
	DomU OS	CentOS 5 26.18-164.el5xen

Our hardware platform has quad core processor and can be extended to 8 cores logically using hyper-threading. The software platform is based on CentOS Linux kernel that is widely used in Xen virtualization. We installed 12 domains on Xen hypervisor and allocated 400MByte memory for each domain. To conduct the experiment, we created 1 domain0 and 10 guest domains to generate CPU, disk I/O, and network activity. We also launched three latency-sensitive domains with periodic realtime tasks; namely, a CPU-intensive domain and a disk-intensive domain. The CPU-intensive domain periodically calls an MD5 hash function, the disk-intensive domain processes disk I/O operations.

Figure 3-(a) shows the runtime of each priority when the Multi-Boost problem occurred under CPU-intensive workloads. However, in that situation, the overall CPU runtime was dominated by UNDER priority tasks. Figure 3-(b) shows the results under disk-intensive. In Figure 3-(a), latency-sensitive workloads failed to meet their deadline, due to the excessive number of BOOST priority domains. Even though CPU consumption was fairly small, CPU switching was very frequent, and this caused realtime tasks to miss their deadlines. In Figure 3-(b), I/O workloads caused frequent CPU switching between VCPUs, and most domains remained in BOOST priority. Note that the CPU consumption of BOOST priority tasks is very high compared with the CPU-intensive case.



(a) CPU-intentive case



(b) Disk-intentive case

Figure 3. Experiment result of latency sensitive domains and non-realtime domains

5. Conclusion

In this work, we propose a practical scheduling mechanism for supporting latency-sensitive guest domains using a realtime priority approach. To support realtime guarantees for tasks running on guest domains, both the guest domain and the VMM must have realtime capabilities. To achieve this goal, we adapted the VCPU shaping mechanism that predicts the whether the behavior of a VCPU in the domain is CPU-intensive or I/O-intensive. We modified the credit scheduler and extended several modules to support latency-sensitive domains. With this approach, we can guarantee the latency for realtime workloads while satisfactorily handling non-realtime workloads.

Acknowledgements:
This research was supported by Basic Science Research Program through the NRF funded by the MEST(.2012R1A1A2044694)

Corresponding Author:
Dr. Young Woong Ko
Department of Computer Engineering
Hallym University
Chuncheon, Gangwondo 200702, South Korea
E-mail: yuko@hallym.ac.kr

References

1. Barham, P. Dragovic, B. Fraser, K. Hand, S. Harris, T. Ho, A. Neugebauer, R. Pratt, I. and Warfield, A., Xen and the art of virtualization,ACM SIGOPS Operating Systems Review, 2003:37(5):164-77.
2. Praveen G, V., Analysis of Performance in the Virtual Machines Environment, J. Advanced Science and Technology, 2011;32:53-64.
3. Gernot Heiser., The role of virtualization in embedded systems. In IIES '08: Proceedings of the 1st workshop on Isolation and integration in embedded systems, 2008;11–16
4. Lee, M. and Krishnakumar, AS and Krishnan, P. and Singh, N. and Yajnik, S., Supporting soft realtime tasks in the xen hypervisor, ACM Sigplan Notices, 2010;45(7):97-108.
5. Lin, B. and Dinda, P.A., Vsched: Mixing batch and interactive virtual machines using periodic realtime scheduling, Proceedings of the 2005 ACM/IEEE conference on Supercomputing, 2005:8-2
6. A. Menon, J. R. Santos, Y. Turner, G. Janakiraman, and W. Zwaenepoel., Diagnosing Performance Overheads in the Xen Virtual Machine Environment. In 1st ACM/USENIX VEE, 2005.

5/26/2014