Supporting Parallel Soft Real-Time Applications in Virtualized Environment

Like Zhou, Song Wu, Huahua Sun, Hai Jin, Xuanhua Shi
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
wusong@hust.edu.cn

ABSTRACT

The prevalence of multicore processors and virtualization technology enables parallel soft real-time applications to run in virtualized environment. However, current hypervisors do not provide adequate support for them because of soft real-time constraints and synchronization problems, which result in frequent deadline misses and serious performance degradation. In this paper, we propose a novel parallel soft real-time scheduling algorithm which addresses them well, and implement a parallel soft real-time scheduler, named Poris, based on Xen. Our evaluation shows that Poris shortens the execution time of PARSEC benchmark by up to 44.12% compared to Credit scheduler.

Categories and Subject Descriptors
D.4.1 [OPERATING SYSTEMS]: Process Management—Scheduling

Keywords
Virtualization; Soft Real-Time; Parallel; Scheduling

1. INTRODUCTION

With the prevalence of multicore processors in computer systems, many soft real-time applications use parallel programming models to utilize hardware resources better and possibly shorten response time. We call this kind of applications as parallel soft real-time ones, abbreviated as PSRT applications. Examples include cloud-based video streaming, real-time transcoding, computer vision, and gaming, etc. Recent advances in virtualization technology make more and more applications run in virtual machines (VMs). However, when running in virtualized environment, PSRT applications do not behave well and only obtain inadequate performance [3, 4].

On one hand, PSRT applications must respond to requests in a timely fashion due to their soft real-time constraints. A PSRT application can have responses only when the virtual CPU (VCPUs) of its hosting VM is scheduled. However, current CPU schedulers in hypervisors do not consider the soft real-time constraints of PSRT applications, which cause frequent deadline misses. On the other hand, PSRT applications have synchronization requirements due to their parallel feature. In virtualized environment, PSRT applications always run in the VMs with multiple VCPUs. However, the asynchronous CPU scheduling used by hypervisors causes synchronization problems, such as lock-holder preemption [5], which results in a waste of CPU time doing useless work, and then affects the performance of PSRT applications seriously.

In summary, PSRT applications have to face soft real-time constraints and synchronization problems. In this paper, we propose a novel parallel soft real-time scheduling algorithm, which considers them simultaneously.

2. OUR APPROACH

Our approach is based on Xen [1] because it is open-source and widely used. Since the parallel soft real-time scheduling algorithm needs to address soft real-time constraints and synchronization problems simultaneously, the scheduling problem can be divided into two sub-problems: 1) how to schedule VCPUs to ensure soft real-time constraints? 2) how to schedule all the VCPUs of a VM to solve synchronization problems? We present different solutions to address these sub-problems, and the parallel soft real-time scheduling algorithm consists of both solutions.

2.1 How to Handle Soft Real-time Constraints

Soft real-time applications can be divided into event-driven ones and time-driven ones according to their characteristics. Event-driven soft real-time applications are executed when external events arrive. Time-driven soft real-time applications are executed periodically. Some soft real-time applications may have both characteristics.

In order to satisfy the performance of event-driven soft real-time applications, we introduce real-time priority, which is higher than boost in Xen's Credit scheduler, to achieve timely scheduling for RT-VCPUs. (For simplicity, we call the VM hosting soft real-time applications as RT-VM, and the VCPU of RT-VM as RT-VCPU.) If a RT-VCPU with under priority in the run queue receives an external event, the priority of the RT-VCPU is promoted to real-time. The RT-VCPU preempt current running VCPU if the priority of RT-VCPU is higher than that of the current running VCPU, and its priority degrades to under when the RT-VCPU is descheduled.

Dynamic time-slice mechanism is introduced to guarantee the performance of time-driven soft real-time applications. If there is no RT-VM in the system, long time slice (i.e. 30ms) is used to schedule VCPUs. Otherwise, the time slice
Section 2.1 and 2.2 respectively. We first define some variables as follows:

- \( S \): the length of time slice that the scheduler used.
- \( LTS \): long time slice (i.e., 30ms in Credit scheduler).
- \( N_R \): the total number of RT-VCPUs in the system.
- \( N_V \): the number of VCPUs per PCPU, which is the ratio of total number of VCPUs and total number of PCPUs.
- \( L \): the expected latency of soft real-time applications.

\( S \) can be calculated by equation (1):

\[
S = \begin{cases} 
  \frac{LTS}{N_R} & N_R > 0 \text{ and } N_V = 1 \\
  \frac{L}{N_V} & N_R > 0 \text{ and } N_V > 1 
\end{cases}
\]

### 2.2 How to Handle Synchronization Problems

We use parallel scheduling to address synchronization problems when PSRT applications run in VMs. It schedules all the VCPUs of a RT-VM at the same time. (As to the non-real-time VMs, the scheduler uses default scheduling strategy, i.e., asynchronous scheduling, to schedule their VCPUs.) Such scheduling strategy can eliminate the synchronization problems of PSRT applications while minimizing the impact on non-real-time applications. The design of parallel scheduling is as follows. First, the scheduler distributes all the VCPUs of a RT-VM across different PCPUs. Second, when a VCPU is scheduled, if it is the first scheduled VCPU of a RT-VM, the priorities of other VCPUs, belonging to the RT-VM, are promoted to real-time. The scheduler reinserts the VCPUs into the proper position of corresponding PCPUs’ run queue (mostly, it is the head of the run queue), and soft interrupts are sent to corresponding PCPUs to trigger rescheduling.

### 3. PERFORMANCE EVALUATION

The physical machine has two 2.4GHz Intel Xeon CPUs (two quad-core CPUs), 24GB memory, 1TB SCSI disk and 1Gbps Ethernet card. We use Xen-4.0.1 as the hypervisor and CentOS 5.5 distribution with the Linux-2.6.31.8 kernel as the OS. All the configurations of VMs are as follows: 8VCPUs, 1GB memory and 10GB virtual disk.

We implement a parallel soft real-time scheduler, named \textit{Portis}, based on Xen 4.0.1, and use PARSEC benchmark suite to evaluate the effectiveness of \textit{Portis}. PARSEC benchmark suite contains 13 multithreaded programs from many different areas. All of them are parallel programs, and some of them have soft real-time constraints, such as \textit{vips}, \textit{fluidanimate}, \textit{facesim}, and \textit{streamcluster}. For comparison, we also implement soft real-time scheduler (RS) and parallel scheduler (PS) under Xen-4.0.1 based on the descriptions in Section 2.1 and 2.2 respectively.

In this test, we run PARSEC benchmark in a VM, and set the VM as RT-VM. We specify the thread parameter of PARSEC benchmark as eight threads, and choose native data set as the input set. We use three other VMs run CPU-intensive workloads to compete with the RT-VM. The expected latency is set to 15ms. Figure 1 shows the results where the bars are normalized execution time.

![Figure 1: Normalized execution time of PARSEC benchmark under different schedulers](image)

As can be seen from the test results, for each individual benchmark, the performance of \textit{Portis} is the best among these schedulers. The performance of \textit{Portis} is up to 44.12% better than Credit, 41.28% better than RS, and 28.02% better than PS.

### 4. FUTURE WORK

Currently, our approach needs administrators to set the type of a VM manually, and only supports multithreaded soft real-time applications running in a VM. In the future, we will research the adaptive parallel real-time scheduling in which the type of a VM can be identified automatically according to the runtime characteristics of applications, and extend current scheduling algorithm to support multiple VMs running the same PSRT applications.

### 5. ACKNOWLEDGEMENTS

The research is supported by National Science Foundation of China under grant No.61073024 and 61232008. It is also supported by National 863 Hi-Tech Research and Development Program under grant No.2011CDA086S, and MOE-Intel Special Research Fund of Information Technology under grant MOE-INTEL-2012-01.

### 6. REFERENCES


