Proceedings of 2014 International Symposium on Embedded Technology (ISET 2014)
Seocho-gu Seoul, Korea, 22-23 May, 2014.

Real-time Schedulability Analysis for Hardware Supported Virtual Machine on ARM Cortex-A15 Processor

Tae Kyoung Kim* Se Won Kim† Chuck Yoo†

*Department of Convergence Software
Korea University, Seoul, Republic of Korea
E-mail: tkkim84@os.korea.ac.kr

†Department of Computer Science and Engineering
Korea University, Seoul, Republic of Korea
E-mail: {swkim, hxy}@os.korea.ac.kr

Abstract

The researches applying virtualization techniques to mobile environment have been studied. Previous mobile virtualization researches needed to modify its guest OSes. Newly published processor, ARM Cortex-A15, provides hardware support for a virtualization (HVM) and it is no more necessary to modify the guest OSes. Thus, we investigated real-time schedulability and bottleneck of performance for the virtualized mobile system with this processor. In our performance analysis, we found that hardware supported virtual machine was hard to guarantee the real-time support and the bottleneck of performance was guest OS, not the hypervisor.

Keywords: mobile virtualization, hardware supported virtual machine, real-time support.

1. Introduction

The technology of virtualization has been developed through decades, and recently, this techniques flow into the mobile environment. Applying virtualization techniques to mobile derives benefits as following: providing multiple OSes[1] and security through isolation[2]. The researches to accommodate virtualization into mobile have been studied. The one of the main obstacles of mobile virtualization is real-time support. In [3], the author suggested 'SH-quantization' algorithm to support real-time scheduling in Xen-ARM virtual machine. Nevertheless, previous researches have limitation that their guest OSes need to be modified called as Para-virtualization. Newly published processor, ARM Cortex-A15, provides hardware support for a virtualization. Therefore, in this paper, we show whether virtualization with this processor guarantees real-time support, and investigate the bottleneck of performance in the virtualized mobile system through the performance analysis.

2. Implementation

The hardware platform we used in our experiments was Arndale Board which applied ARM Cortex-A15 processor. The board had an Exynos 5250 CPU with 1.7GHz core frequency, dual cores, and 2GB memory size. For software platforms, we used Xen 4.4(arch/arm) as a hypervisor and linux kernel 3.9 as the Domain0 and DomainU guest OSes. The memory size of the Domain0 was 256MB as a default and 128MB for the DomainU. Scheduling policy for Xen, we followed the default policy, credit scheduling, and for Domains we followed default linux scheduler.

3. Performance analysis

To determine whether real-time schedulability was guaranteed in HVM, we did performance analysis in two different perspectives. First, we used benchmark, called 'Cyclictest', to confirm the deadline of the interrupt handling was guaranteed. The table 1 shows result of benchmark executing. The interrupts were occurred by POSIX interval timer with 10ms interval and 10000 loops. This result showed that the deadline of Xen-ARM(HVM) could not guaranteed because of its maximum latency value, about 7ms, instead the native linux system satisfied deadline within maximum value of 0.4ms. In real-time schedulability perspective, it is more significant that finishing tasks within deadline than lower average latency.

Table 1. Interrupt latency on the native linux and the virtualized system.

Target	Interrupt latency (us)		
	Min	Max	Avg
Native Linux	12	482	83
Xen-ARM(HVM) - Domain0	20	7187	167

The other performance analysis was that calculating interrupt handling time to define where the bottleneck of performance was in virtualized mobile system. It was measured section by section using a performance monitoring unit (PMU). Figure 1 illustrates the measurement boundary of interrupt handling either the native linux and Xen-ARM(HVM). In the native linux system, we counted the CPU cycles from vector table to __irq_svc(), and calculated the execution time by dividing cycle by CPU clock speed (actual survey clock speed was 1GHz). The average interrupt handling time of the native linux was 17us. On the other hand, in case of Xen-ARM(HVM), there was the hypervisor vector table that determined which domain should be delivered the interrupt. Therefore, there was one more stage than the native linux's case. The average time of interrupt handling in the hypervisor only took 0.25us, but in domain0 kernel it took 24.78us until the interrupt handling was finished.

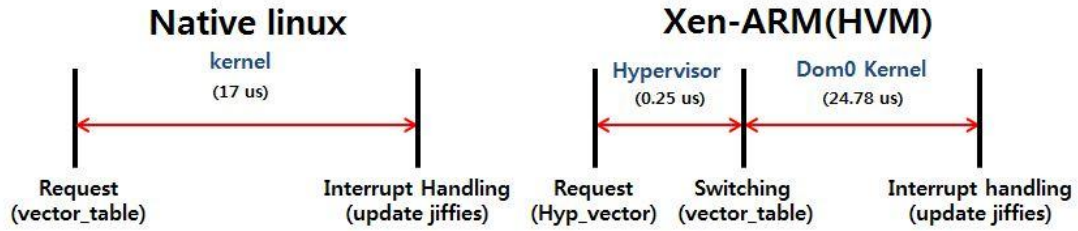

Figure 1. Measurement boundary of interrupt handling with PMU.

4. Conclusions

The performance experiments we had done in section 3 indicated two essential phenomena. First, the Xen-ARM hypervisor with hardware support is hard to guarantee a real-time schedulability because of its maximum interrupt latency. The other feature is that the bottleneck of performance during the interrupt handling comes out Domain0 not the hypervisor. In the future, we will demonstrate cause of overhead in the Domain0, and will measure performance in DomainU with a real-time operating system such as a uC/OS-II.

Acknowledgement
This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No.2010-0029180) with KREONET.

References

[1] Heiser, Gernot. "Hypervisors for consumer electronics." Consumer Communications and Networking Conference, 2009.
[2] Brakensiek, Jörg, et al. "Virtualization as an enabler for security in mobile devices." Proceedings of the 1st workshop on Isolation and integration in embedded systems. ACM, 2008.
[3] Yoo, Seehwan, and Chuck Yoo. "Real-time Scheduling for Xen-ARM Virtual Machines." IEEE Transactions on Mobile Computing, 2013