Hypervisor Design Considering Network Performance for Multi-core CE Devices

Cheol-Ho Hong, Miri Park, Seehwan Yoo, Chuck Yoo
Department of Computer Science and Engineering, Korea University, South Korea

***Abstract*—Recently, system virtualization has been applied to consumer electronics such as smart mobile phones. Although multi-core processors have become a viable solution for complex applications on consumer electronics, how to utilize the multi-core resources in virtualization layer is not researched sufficiently. In this paper, we present the hypervisor design and implementation for multi-core CE devices; the hypervisor targets to improve network performance by fully utilizing the multi-core processor.**

I. INTRODUCTION

Recently, system virtualization has been applied to consumer electronics such as smart mobile phones to solve security problems derived from internet malwares [1]. The system virtualization technology allows multiple virtual machines (VMs) to be consolidated simultaneously in a physical machine. In a virtualized system, even though a guest operating system (guest OS) is compromised by malwares, other guest OSs can remain uninfected by the isolation service provided by the virtualization layer; this enables availability of a whole system.

At the bottom of system software stacks, a virtual machine monitor (VMM) or hypervisor virtualizes system resources such as a processor, memory and I/O devices. The VMM provides virtualized resources to each guest operating system on it as a form of VM.

Although the complexity of applications on consumer electronics has led embedded multi-core processors to be a viable solution, research for the VMM for embedded multi-core processors has not been accomplished sufficiently. Xen on ARM, a representative VMM for consumer electronics, is targeting for the single-core platform. Other commercial VMMs support embedded multi-core processors, but their design and implementation are unknown [3].

In this paper, we present the hypervisor design and implementation for multi-core CE devices; the hypervisor aims to improve network performance by fully utilizing the multi-core resources. The multi-core hypervisor is based on the ARM11 MPCore processor [4] and is revised from the single-core MobiVMM which is our research VMM for the soft real-time application [5], [6].

II. STATE OF THE ART

Xen on ARM is migrated from the open source Xen [2] which targets server consolidation; Xen on ARM targets embedded OSs consolidation and security features between OSs. It adopts para-virtualization technique which requires operating system modification.

Xen on ARM provides ARM CPU, memory and I/O device virtualization. In CPU virtualization, Xen on ARM replaces sensitive instructions, which modify system states and are executed in the supervisor mode, of guest OSs to hypercalls. The hypercalls request the VMM to execute the instructions on behalf of the guest OSs. In memory virtualization, it provides memory protection between a VMM, guest OSs and applications by exploiting domain protection mechanism as ARM architecture has fewer privilege rings (two rings) than x86 architecture. In device virtualization, the domain0 VM which is an administrator domain (VM) runs most of native device drivers; all of the hardware interrupts are delivered to the domain0, before they are redirected to target VMs as virtual interrupts. This architecture degrades network performance, because network packets of the target VM are sent or received through a domain0 [2].

III. HYPERVISOR DESIGN

A. Multi-core processor virtualization

ARM11 MPCore processor consists of four major parts; four individual core, local timer, CPU interface and a interrupt distributor which distributes hardware interrupts from physical devices to target cores. Fig. 1 illustrates whole components of the processor.

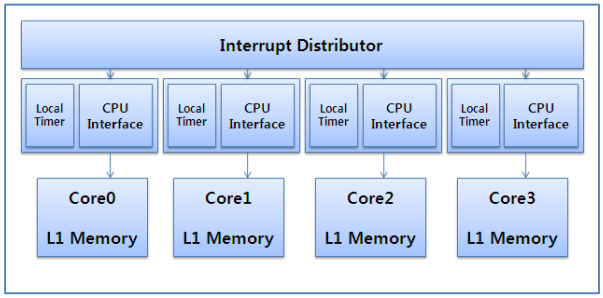

Fig. 1. Components of ARM11 MPCore processor.

For the virtualization of each individual core, the hypervisor replaces all of the sensitive instructions to hypercalls. The local timers, CPU interfaces and interrupt distributor are also virtualized by emulation of related instructions.

To fully utilize all cores when interrupt loads are heavy, we extend functionality of the virtualized interrupt distributor to scatter interrupts dynamically among cores. Originally, ARM11 MPCore interrupt distributor does not support dynamic distribution of interrupts, whereas the local APIC of x86 platforms provides both static and dynamic distribution;

978-1-4244-4316-1/10/$25.00 ©2010 IEEE

hardware interrupts are delivered to core0 in default settings of the MPCore processor. The dynamic distribution of the virtualized interrupt distributor operates as follows. When a hardware interrupt occurs at one core, the hypervisor generates inter-processor interrupt (IPI) to the target core. Then, the target core recognizes what the interrupt is by fetching the IPI number, and executes interrupt handling processes. After finishing the processes, it acknowledges the end of interrupt to the source core through the CPU interface alias of the source core as the interrupt has occurred on the source core.

B. *Memory virtualization*

The hypervisor shares the same address space with a guest OS to avoid frequent cache and TLB flush. It resides at the highest 32MB of the 4GB address space. Because the memory is an independent part irrelative to multi-core platforms, implementation of the hypervisor completely followed that of the single-core MobiVMM.

C. *I/O virtualization*

Compared to Xen on ARM of which domain0 has all of the native drivers, our hypervisor is designed to allow each guest OS to have native drivers. Each native driver is dedicated to a guest OS or shared between domains by I/O ring buffer mechanism which performs delivery of I/O requests and responses. This architecture helps to decrease response time of a soft real-time VM by avoiding indirect access to native drivers. In our approach, each interrupt of peripheral devices is delivered to the VM that has the native device driver for that interrupt.

IV. NETWORK PERFORMANCE IMPROVEMENT

When we don't use the dynamic interrupt distribution functionality in the I/O virtualization model, the hypervisor suffers from network performance degradation, because heavy workloads are concentrated on one core. Usually, consumer electronic device is composed of a soft real time VM, which executes soft real time applications such as MP3 decoders, and a general purpose VM for Internet use. In this architecture, audio interrupts are delivered to the soft real time domain; network interrupts are delivered to the general purpose domain. When both audio and network interrupts occur simultaneously and continuously, they cannot be processed in expected time as only one core becomes busy context switching between the VMs and processing the interrupts. As a result, this architecture degrades network performance. We have measured bandwidth of network, using the Iperf benchmark program in the general purpose domain while the soft real time VM is idle. The result is 15.7 Mbits/sec on average. When the soft real-time domain plays an mp3 file, the result is, however, 2.30 Mbits/sec on average.

To improve network performance when heavy workloads are on one core, we scatter the workloads by help of the virtual interrupt distributor and VMM profiler to other cores which are idle or underutilized. Fig. 2 describes our approach. When both audio and network interrupts occur on one core continuously, the virtual interrupt distributor decides whether it will deliver the network interrupt to the same core or to other underutilized cores. The decision is based on the information of the profiler in the VMM, which analyzes utilization of each core.

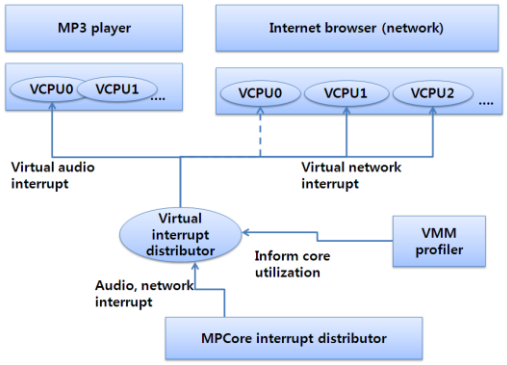

Fig. 2. Distribution of virtual interrupts. Note that a VCPUn (virtual CPU) is executed on a physical CPUn, where n is a core number.

V. EVALUATION

We have implemented the hypervisor on ARM11 MPCore platform which has four 250MHz ARM11 cores. We use two para-virtualized guest OSs of which kernels are Linux 2.6.21.

Using the Iperf network benchmark program, we have measured network bandwidth in the general purpose domain. Table I illustrates the benchmark result. Compared to the case no. 2 whose bandwidth is 2.30 Mbits/sec, our approaches (case no.3 and case no.4) improve network performance by 5.5 times.

TABLE I
Iperf benchmark result

case no.	audio	network	bandwidth (Mbits/sec)
1. max bandwidth	no audio	core0	15.7
2. heavy workloads	core0	core0	2.30
3. distribution #1	core0	core1~3	13.0
4. distribution #2	core0	core1	13.4

VI. CONCLUSION

We present the hypervisor design and implementation for multi-core CE devices. The hypervisor achieves improvement of network performance by fully utilizing the multi-core resources in interrupt processing.

REFERENCES

[1] J.-Y. Hwang, S.-B. Suh, S.-K. Heo, C.-J. Park, J.-M. Ryu, S.-Y. Park, and C.-R. Kim, "Xen on ARM: System Virtualization Using Xen Hypervisor for ARM-Based Secure Mobile Phones," in *Proc. CCNC,* Jan. 2008.

[2] 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," in *Proc. SOSP*, Oct. 2003.

[3] http://www.virtuallogix.com/

[4] http://www.arm.com/products/CPUs/ARM11MPCoreMultiprocessor.html.

[5] S. Yoo, Y. Liu, C.-H. Hong, C. Yoo, Y. Zhang, "MobiVMM: a Virtual Machine Monitor for Mobile Phones," in *Proc. MobiVirt*, June 2008.

[6] S. Yoo, M. Park, C. Yoo, "A Step to Support Real-Time in Virtual Machine," in *Proc. CCNC,* Jan. 2009.