ViMo-S: A Lightweight Hypervisor based on ARM Virtualization Extensions

Song-Woo Sok, Young-Woo Jung, Cheol-Hun Lee

Abstract: The ViMo-S, a type 1 hypervisor for ARMv7 and ARMv8-based ARM server systems, supports full virtualization to run existing operating systems and applications unmodified. It uses ARM hardware virtualization extensions to optimize the performance of virtual machines. Therefore, its virtual machines' system call latency is near physical machine's, while other hypervisors like Xen and KVM show relatively slower and unstable performances in benchmark tests.

Index Terms: ARM, Hypervisor, Virtualization.

I. INTRODUCTION

The ARM processor dominates the market for mobile and embedded devices thanks to low-power characteristics. With the performance of the ARM processor nearing the x86 processor and increasing the clock speed, the attempt to apply the ARM processors to the server is increasing. Therefore, high density servers are becoming popular for their low power consumption and low heat emission properties.

The low-power high-density servers have benefits for their smaller physical installation space needs and lower power consumption for computing and cooling. To increase the benefits more, server consolidation based on virtualization is very important [1].

Traditional ARM architectures have no hardware support for virtualization technologies. However, ARM virtualization extensions (VEs) have been introduced since ARMv7 architecture to support virtualization software [2]. Thanks to ARM VEs, virtualization software are able to be more lightweight and efficient.

Traditional ARM architectures support the least privileged EL0 mode for user applications and the more privileged EL1 mode for kernel. ARM VEs added new EL2 mode which has more privileged than the EL1 to support hardware virtualization. Hypervisor software can run in EL2 mode and support virtualization for existing operating systems (OSs) and applications unmodified.

Two types of hypervisor software are shown in Fig 1. The type 1 hypervisors are independent and most privileged software in the system. The hardware resources of the system and virtual machines (VMs) are controlled by the hypervisor. Generally, Type 1 hypervisors support a special VM (Domain0) to control hardware devices. And, other VMs (DomainU) can access virtualized resources with domain0 VM's support.

The Xen is a popular type 1 hypervisor software that supports x86 and ARM architectures. Xen ARM executes before VM's OS boot. Then it creates a special VM (Domain0) to control hardware devices. After domain 0 booted, user can create domainU VMs which can executes I/O helped by the domain0 VM [3].

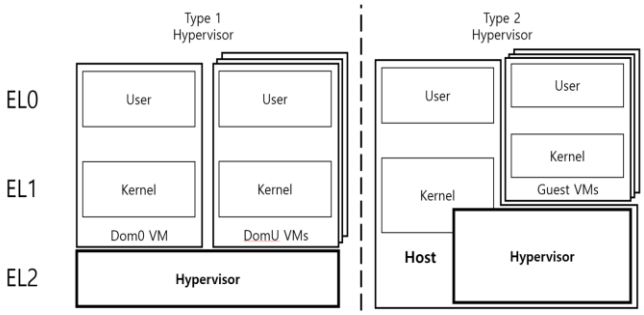

Figure 1. Software stacks of two types of virtualized systems.

The type 2 hypervisors are executed as a part of the host OS. However, this type of hypervisor is not fitted to ARM architectures because the ARM VEs' EL2 mode is more privileged than the EL1 mode which is used for OS' kernel and the EL2 mode cannot be accessed from the EL1.

The KVM is a type 2 hypervisor for x86 architectures and runs as a part of the Linux kernel. However, it was needed to change its structure to fit to the ARM VEs architectures. KVM/ARM has two separated parts, one is called highvisor which is a part of the Linux kernel, and the other is called lowvisor which runs in EL2 mode to use ARM VEs [4].

This paper presents the ViMo-S which is a type 1 hypervisor for ARM architectures that have ARM VEs. It supports full-virtualization for CPU and memory and para-virtualization for I/O with VirtIO interface.

II. VIMO-S

A. *Developing of ViMo-S*

The ViMo-S is the following project of ViMo (Virtualization for Mobile), a micro virtual machine monitor for ARM mobile systems. The ViMo was developed for ARM architectures which has no VEs, its design was complicated and inefficient. Its VMs had quite slower performances than physical machines' [5].

Revised Version Manuscript Received on October 28, 2016.

Song-Woo Sok, High-Performance Computing Dept., Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.

Young-Woo Jung, High-Performance Computing Dept., Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.

Prof. Cheol-Hun Lee, Department of Computer Science & Engineering, Chungnam National University, Daejeon, Republic of Korea.

The ViMo-s has relatively simple and efficient design thanks to support of ARM VEs. Also, it was needed to support only three types of virtual devices like console, disk, and network because its main purpose is to support server applications. While, ViMo was needed to support more device types like pen, touch screen, and sensors for mobile applications.

B. CPU Virtualization

The ARM virtualization extensions are more suitable to type 1 hypervisors than to type 2 hypervisor [1]. The ViMo-S was planned to be a type 1 hypervisor from the start. As a type 1 hypervisor, The ViMo-S runs in the EL2 mode just after a boot loader and before any OS boots; then it initializes the first VM, domain0, and boots its operating system.

All VMs can have one to four virtual CPUs (VCPUs) and the ViMo-S supports virtualized inter-core communications to support multi-core VMs. VCPUs are data structure kept in memory, and it is loaded to physical registers when it is scheduled. If another VCPU is running in the core, ViMo-S saves the current context of the core to the related VCPU then loads scheduled VCPU's context to the core. Therefore, 2 or more VCPUs can temporally share a core and be executed simultaneously.

User applications call system calls frequently when they are running. ViMo-S configures system calls to be routed to EL1, not to EL2. Because the VM's kernel directly handles all system calls and there is no intervention of the hypervisor, system call latency of the VMs on ViMo-S is near to a physical system.

C. Memory Virtualization

The ARM VEs support hardware 2-stage memory translation to support memory virtualization for VMs. The first stage's page tables are configured by operating systems of VMs, and its translated address is an intermediate physical address (IPA). If the second stage page tables are configured in EL2, then an IPA can be translated to a physical address (PA) by hardware MMU. The ViMo-S allocates physical memory and configures second stage page tables for each VMs. ViMo-S allocates memory area exclusively to each VMs. However, the ViMo-S can create shared memory space by configuring second stage page tables of VMs to have same physical memory address entries.

D. I/O Virtualization

The ViMo-S supports VirtIO interface for para-virtualized I/O operations. If domainU's operating system has VirtIO drivers, it can use a console, disk and network VirtIO virtual devices to execute I/O operations.

The processing path of virtual I/O requests are shown in Fig 2. Domain U's VirtIO requests are trapped to ViMo-S and ViMo-S delivers I/O requests to the domain0's VirtIO backend driver. Then, the VirtIO backend driver maps the requests to real hardware I/O operations and calls hardware drivers to process the requests.

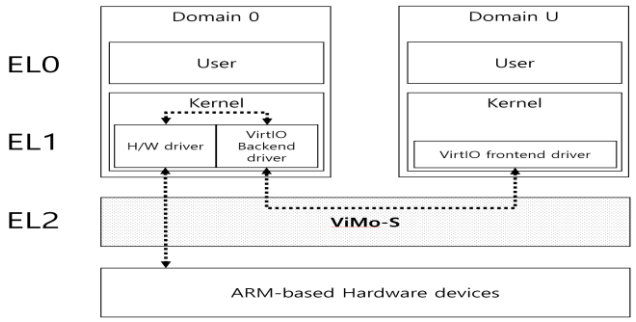

Figure 2. I/O Virtualization of ViMo-S.

E. Implementation

ViMo-S has been implemented on X-C1 Server Development Platform board developed by AppliedMicro.

The main SoC of the board is AppliedMicro's X-Gene 1, an ARMv8 compliant processor. It has eight cores and runs at 2.4 GHz clock speed. The X-C1 has 16GB of DDR3 RAM, three ports of Gigabit Ethernet and two SATA ports. A Samsung 850 EVO 500GB SSD was used as storage device.

We used Linux 3.15-rc8 kernel for domain0 and Linux 3.18.0 for domainUs. Ubuntu 14.04 ARM64 version was used for both domain0 and domainU's root file systems.

III. BENCHMARK TESTS AND RESULTS

A. Benchmark test configuration

To evaluate the performance of ViMo-S, we used lmbench 3.0 to check system call latencies of virtualized and non-virtualized environments on the X-C1 board. The tests included ViMo-S, Xen ARM, and KVM/ARM configurations. The lmbench is a benchmark suite to measure system's micro performance factors like system calls, memory accesses, network, and disk performances [6]. Table 1 shows configurations for each system.

Table 1. Configurations for benchmark tests.

	Version	Domain 0/Host Core/Memory	Domain 0/Host Kernel	Domain U/Host Core/Memory	Domain-U Kernel
ViMo-S	-	1core/2GB	Linux-3.15-rc8	1core/512MB	Linux-3.18.0
Xen ARM	4.4.0	1core/2GB	Linux-3.15-rc8	1core/512MB	Linux-3.15-rc8
KVM/ARM	Linux-3.15-rc8	1core/2GB	Linux-3.15-rc8	1core/512MB	Linux-3.18.0

B. Raw Linux and Domain0s

First, we conducted lmbench tests of non-virtualized Linux, ViMo-S and Xen's domain0 VMs.

Table 2 shows the results of lmbench system call latency tests of raw Linux and domain0 VMs. For easy comparison, it includes latency ratios based on raw Linux results also.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Table 2. lmbench test results for raw Linux and domain-0s (unit=microseconds).

	Simple syscall	Simple read	Simple write	Simple stat	Simple fstat	Simple open/close	Protection fault	Pipe latency
Raw Linux	0.1318	0.1845	0.2312	0.8316	0.1936	2.4462	0.2057	4.7348
ViMo-S	0.1316	0.1833	0.2308	0.8406	0.1951	2.1689	0.1993	4.6733
Xen ARM	0.3834	0.5356	0.6008	2.5131	0.4516	5.9182	0.5072	13.6326
ViMo-S/ Linux	99.8%	99.3%	99.8%	101%	100.8%	88.7%	96.9%	98.7%
Xen/Linux	291%	290%	260%	302%	233%	242%	247%	288%

As shown in Table 2, raw Linux and ViMo-S have very similar results, while Xen Domain0 shows relatively slower performances. Xen's system call latencies are 2.3 to 3 times slower than ViMo-S and raw Linux.

C. Domain Us and KVM/ARM guest

We did same lmbench tests on KVM/ARM's guest, ViMo-S and Xen's domain-U VMs.

Table 3 shows the results of lmbench system call latency tests of KVM guest, domain-U VMs. The raw Linux results from Table 2 are also included for comparison.

Table 3. lmbench test results for KVM/ARM guest and domain-Us (unit=microseconds).

	Simple syscall	Simple read	Simple write	Simple stat	Simple fstat	Simple open/close	Protection fault	Pipe latency
Raw Linux	0.1318	0.1845	0.2312	0.8316	0.1936	2.4462	0.2057	4.7348
ViMo-S	0.1221	0.2279	0.3104	1.0876	0.1825	3.0000	0.0992	6.8388
Xen ARM	0.3935	0.6091	0.6129	2.3186	0.4395	5.7837	0.3724	14.0291
KVM/ARM	0.1218	0.1963	0.2913	2.7275	0.1964	6.9162	0.2673	108.6665
ViMo-S/ Linux	93%	124%	134%	131%	94%	122%	48%	144%
Xen/Linux	299%	330%	265%	279%	227%	236%	181%	296%
KVM/Linux	92%	106%	126%	328%	101%	283%	130%	2295%

ViMo-S's domain–U has relatively slower performance than ViMo-S's domain–0. However,~~But~~ its latencies did not~~esn't~~ exceed 1.5 times of raw Linux's.

Xen's domain–U shows similar performance with Xen's domain-0's. It has 1.8 to 3.3 times slower than raw Linux.

KVM/ARM's guest VM shows interesting results. For some system call tests, its latencies are slightly faster than ViMo-S~~,~~ but significantly slower for simple stat, simple open/close and pipe latency tests.

IV. CONCLUSION AND FUTURE WORKS

This paper proposed ViMo-S, a new type 1 hypervisor for ARM architecture. ViMo-S supports ARM virtualization extensions to optimize VM's performance and minimize virtualization overheads. We showed that ViMo-S has minimized system call latency for VMs and provides similar performance with bare metal machines.

According to our experiment results, ViMo-S has better and more stable system call latency performances compared to Xen and KVM/ARM. We are trying to optimize I/O performance of ViMo-S and doing more benchmark tests with more complex benchmark tools and real world applications.

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) [R0101-16-237, Development of General-Purpose OS and Virtualization Technology to Reduce 30% of Energy for High-density Servers based on Low-power Processors]

REFERENCES

1. Dall, C., Li, S. W., Lim, J. T., Nieh, J., and Koloventzos, G. (2016, June) "ARM Virtualization: Performance and Architectural Implications," In Proceedings of International Symposium on Computer Architecture (ISCA 2016).
2. Varanasi, P., and Heiser, G. (2011, July). Hardware-supported virtualization on ARM. In Proceedings of the Second Asia-Pacific Workshop on Systems (p. 11). ACM.
3. Stabellini, S., and Campbell, I. (2012). Xen on arm cortex a15. Xen Summit North America, 2012.
4. Dall, C., and Nieh, J. (2014, February). KVM/ARM: the design and implementation of the Linux ARM hypervisor. In ACM SIGPLAN Notices (Vol. 49, No. 4, pp. 333-348). ACM.
5. Oh, S. C., Kim, K., Koh, K., and Ahn, C. W. (2010). ViMo (virtualization for mobile): a virtual machine monitor supporting full virtualization for ARM mobile systems. In Proceedings of Advanced Cognitive Technologies and Applications, COGNITIVE.
6. McVoy, L. W., & Staelin, C. (1996, January). lmbench: Portable Tools for Performance Analysis. In USENIX annual technical conference (pp. 279-294).

AUTHORS PROFILE

Song-Woo Sok Senior Researcher, High-performance Computing System Lab., High-performance Computing Dept., Electronics and Telecommunications Research Institute, Republic of Korea.

Young-Woo Jung Chief Researcher, High-performance Computing System Lab., High-performance Computing Dept., Electronics and Telecommunications Research Institute, Republic of Korea.

Prof. Cheol-Hun Lee PhD, Department of Computer Science & Engineering, Chungnam National University, Republic of Korea.