Research about Virtualization of ARM-Based Mobile Smart Devices*

Lei Xu, Wenzhi Chen, and Zonghui Wang

College of Computer Science and Technology
Zhejiang University, Hangzhou, P.R. China
{leixu,chenwz,zjuzhwang}@zju.edu.cn

Abstract. In common sense, virtualization technology is adopted to offer several isolated execution environments and makes better use of computational resources which has been an important enabler for cloud computing. However, in embedded systems, the significance of virtualization does not come into the picture. The extensive utilization of mobile smart devices has led to a series of issues such as security, power consumption and performance limitation. Mobile virtualization can offer an effective approach in addressing these challenges. In this paper, we discuss how mobile virtualization addresses these challenges and then present a detail analysis of mainstream mobile virtualization solutions: ***Para-virtualization***, ***Hardware-Assisted Full virtualization*** and ***Microkernel Hypervisor***. At last, we carry out a series of performance comparison between these solutions and make some suggestions for further research.

Keywords: Smart Devices, Mobile Virtualization, ARM, Android.

1 Introduction

The ability of traditional virtualization brings immense benefits in terms of reliability, efficiency and scalability. It enables the datacenters to flexibly provision resource which makes the *computing-as-a-service* vision of cloud computing possible [1]. A substantial amount of works have been carried out on traditional virtualization most of whose architecture is X86.

Nevertheless, ARM-Based mobile smart devices are becoming more and more ubiquitous and the preferred platform for users' daily computing needs are shifting from traditional desktop to mobile smart devices [2]. Undoubtedly, as mobile computing advances, it brings several tough challenges, as described follows:

- **Security Threats.** Mobile device, as a kind of intimate personal portable equipment, contains lots of user's sensitive data, such as SMS, contacts and photos. People can't pay much more attention on its security issues, especially in a poor secure condition nowadays.

* This research is funded by National Science and Technology Major Project of the Ministry of Science and Technology of China under grant NO. 2013ZX03003010-002.

James J. (Jong Hyuk) Park et al. (eds.), *Multimedia and Ubiquitous Engineering*,
Lecture Notes in Electrical Engineering 308,
DOI: 10.1007/978-3-642-54900-7_37,

- **Performance Excess.** Multi-Core SOC is increasingly adopted by hardware vendor along with 2G RAM or more. It seems that these vendors are participated in a hardware competition which led to a serious performance wasting. How to make better use of these multi-core hardware resources is a new challenge.
- **Power Consumption.** Power is always the bottleneck of mobile devices. Especially, modern device architecture is becoming more and more complicated to support various modem protocol stacks (GSM, WCDMA, and LTE) simultaneously and many complex applications. People want to find a way to simplify the hardware architecture.
- **Shorter time-to-market.** For devices manufacturer, they wish to quick release their newest products to meet dynamic market requirements. They want to find a way to reduce dependencies among hardware and software components so as to reuse legacy software or legacy operating system on a new design chip/board and reduce development and integration time and effort.

To address these challenges, the role of virtualization within the mobile device is being discussed among academia and industry [3]. Actually, mobile virtualization can deal well with these challenges. But this technology seems has not yet aroused people's enough attention until now. So our team carried out this research.

In this paper, we comprehensively analyze mobile virtualization technology: Section 2 describes the definition and the benefits of mobile virtualization. Section 3 discusses the mainstream solutions in detail. Section 4 carries out performance comparison among those solutions. A summary is described in Section 5.

2 Mobile Virtualization Overview

Mobile device, a modern embedded system, is increasingly taking on characteristics of general-purpose systems. Their functionality is growing, and so is the complexity of their software [4]. This creates a demand that run more and more high-level applications originally developed for the PC world, such as virtual machines. Mobile virtualization is a variant of system virtualization that enables multiple isolated OSs run simultaneously on a single mobile device.

However, mobile device is a personal communication device rather than a totally computing devices. This means that we can't deal with it like PC which is obviously not adequate to mobile virtualization [5]. The requirements of mobile virtualization includes: (1) A small code size and lightweight hypervisor; (2) A strict system-wide security policy; (3) Strong interaction to enhance user experience; (4) Minimal impact on system resources and real-time performance, and so on.

2.1 Mobile Virtualization Benefits

2.1.1 Enhanced Security

The security issues of mobile smart devices are heavily exposed [6]. Viruses, Trojan horses and malwares from all kinds of external attackers have caused people's attention. However, deploying a security environment (such as encryption, digital

signature, safety audit, access control, digital certification, *etc.*) on mobile device is very hard for common users. So people need an innovative solution which can offer a secure and credible execution environment when use some critical applications (mobile banking), or access to sensitive data (SMS, contacts).

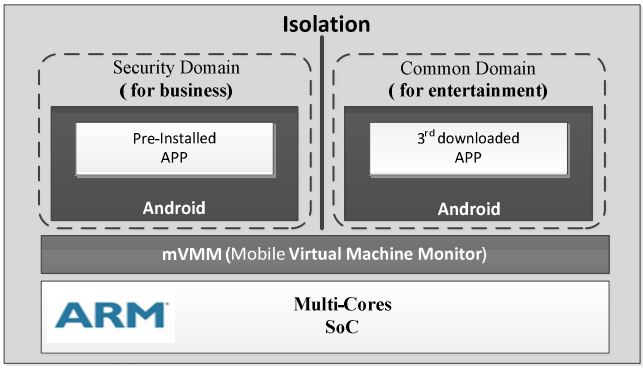

Fig. 1. Multi-OS isolation for enhanced security

Mobile virtualization is such a kind of solution! As shown in Fig.1, a security domain contains pre-installed application for the basic functions of a mobile, such as SMS, dialer and mailer. The 3rd-party downloaded applications can never affect the base domain, which are only allowed to execute in a common domain. The isolation offered by the mobile virtualization makes this possible. We can use security domain for private telephony, business office, mobile banking and so on. Also, we can create several common domains for daily use (browsing, gaming, movie, music, *etc.*).

2.1.2 Simplify Hardware Structure

Current device architecture is very complicated and inevitably brings power consumption problem, as shown in Fig.2. Each core has the different purpose: ARM-A runs general-purpose operating system, which is up to interact with users; ARM-C runs a real time operating system, mainly to complete high-level protocol stack processing of different communication formats; DSP-C has strict real-time requirements, mainly to process underlying protocol stack by interrupt trigger; while DSP-M is always used to decode audio and video.

Mobile virtualization breaks the tightly one-to-one relationship between operating systems and processors. How to enable devices support multiple new protocol stacks (GSM/WCDMA/HSPA/LTE) and avoid compatibility problems between different protocol stacks is a challenge. Instead of using multiple dedicated real-time processors, mobile virtualization offers a new architecture as shown in Fig.3. In this architecture, VMM supports multiple general purpose operating systems (GPOS) and real time operating systems (RTOS) to run concurrently on one ARM processor. And some underlying protocol processes can be scheduled to the unique DSP by the VMM. This architecture can authentically simplify hardware structure.

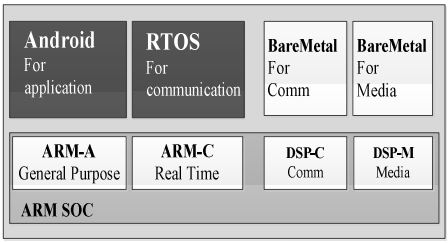

Fig. 2. Typical mobile device architecture

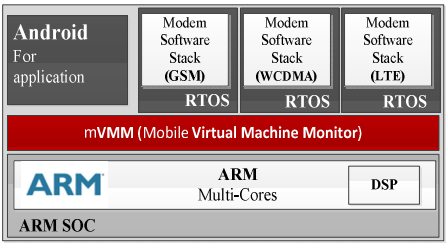

Fig. 3. Simplified hardware architecture

2.1.3 Reuse Legacy Software

Mobile virtualization empowers mobile device manufacturers and semi-conductor vendors to speed time to market and reduce costs by reusing legacy software assets while taking advantage of new designed chip. Maintaining a competitive edge is vital for mobile device manufacturers, who must integrate huge amounts of complex software on multiple chipsets and hardware platforms. Native or proprietary device drivers, protocol stacks and system modules can be integrated with ease, and legacy applications can run unmodified in the new environment because of mobile virtualization. This ensures minimum development cost and faster time to market for new products.

3 Mainstream Mobile Virtualization Solutions

3.1 Solution 1: Para-virtualization of ARM

Para-virtualization is a very mature technology used by Xen, a famous hypervisor. It refers to a technique where the guest operating system is modified and privileged instructions are replaced with calls to the hypervisor named *hypercalls*. The hypervisor layer provides a *hypercall* interface with services such as memory management, device usage and interrupts management to the guest. This ensures that all privileged mode activities are moved from the guest operating system to the hypervisor. Since para-virtualization requires changes to the guest operating system code to avoid calls to privileged instructions, it obviates the need for trap & emulate and binary translation. Of course, this benefit comes with the additional cost of maintaining a modified guest operating system.

This solution offers several advantages:

- **Relatively High Performance:** Para-virtualization provides specially defined hooks to allow the guest(s) and host to request and acknowledge tasks, which would otherwise be executed in the virtual domain, so it can reduce the portion of the guest's execution time spent performing operations.

There are some drawbacks:

- **Poor User Experience:** Solutions based on para-virtualization is not fit for mobile device. It has a complex configuration which is not easy for common user and it need to modify the guest OS code which means it can't support the latest OS and commercial closed-source operating systems.

3.2 Solution 2: Hardware-Assisted Full Virtualization of ARM

ARM announced their Cortex-A15 processor with architectural virtualization support in 2010. This virtualization extension provides a new processor mode (HYP mode) and a number of features to improve performance [7]. HYP mode is entered from other modes via a new instruction (hvc), and optionally on a configurable set of exceptions from user or kernel mode. It has banked registers, as well as additional hyp-only registers for system configuration and information on the event which caused entry of hyp mode. There is a hyp-only virtual machine identifier (VMID) register. TLB entries are tagged with the VMID, which supports coexistence of mappings from multiple guests and thus eliminates the need to flush the TLB on a world switch.

This solution offers several advantages:

- **Hardware Support:** This solution is the exclusive way that makes use of ARM hardware feature. This means it can reduce code size and increase reliability. Predictably, this solution will be the major way used in ARM-based machines, even will be applied on the ARM-based server.

There are some drawbacks:

- **Not Mature Enough:** As mentioned above, ARM virtualization extension is a new technology. Meanwhile, KVM is not originally designed based on ARM architecture. So this solution is now not mature and not stable, there is a long way to modify KVM to be adaptable to ARM hardware extensions.
- **Poor I/O Virtualization Ability:** As we know, KVM leverages QEMU (an open-source hosted hardware emulator) to offer the ability to virtualize diverse I/O devices, which is a so heavyweight software that not suitable for mobile devices. So this solution has the poor ability to virtualize I/O device without the support of QEMU.

3.3 Solution 3: Microkernel Hypervisor

With virtual machine based on microkernel architecture, we can convert hardware resources to various real-time system services, and deliver to client operating systems which run on virtual machine by mode of virtual devices. In this way, it can support real-time and non-real-time applications to run simultaneously, and provide a universal and transparent interactive interface between non-real-time applications and real-time system functions. The microkernel approach leads to a system structure that differs significantly from that of classical operating system.

This solution offers several advantages:

- **Efficient Resource Sharing:** Microvisor provides mechanisms for efficient sharing of resources. Arbitrary memory regions can be shared by setting up mappings between address spaces (providing high-bandwidth communication channels).
- **Flexible Scheduling:** Microvisor allows the guest operating system to select the appropriate global scheduling priority which means it can run at a high priority when executing real-time threads, and a lower priority when executing background tasks.

There are some drawbacks:

- **Device Emulation:** Microvisor has to provide device support and emulation, an onerous requirement for mobile devices which provide increasingly diverse hardware devices. For example, we are not aware of any OKL4 implementations that run Android on any phones other than the dated HTC G1.

4 Performance Comparison

We have carried out a series of experiments to evaluate the performance of these different solutions described in Section 3. To make a comparison, we choose out their mutual features to test. We built three open source platforms representing correspondingly those solutions: ***CodeZero*** (Microvisor), ***KVM-ARM*** (Hardware-Assisted Full virtualization) and ***EmbeddedXEN*** (Para-virtualization). Meanwhile, we use Urbetter S5PV210 development board with Exynos 4412 CPU and 2GB memory as our experiment platform. In addition, our host OS is Archlinux with 3.5.4 kernel and our guest OS is Android 4.2.2. At last, we choose LMbench3 to be our benchmark.

4.1 Evaluation Results

4.1.1 Context Switching

We measure context switching time between guest OS and host OS. A context switch is the switching of the CPU from one process or thread to another. When the VMM receives a hardware interrupt, it generally suspends the progression of the current process and starts servicing the interrupt. This is an important feature for mobile software which means a good user experience. As shown in Table 1, hardware-assisted solution has the fastest switch speed.

Table 1. Context Switching Time

	Full Virtualization	Para Virtualization	Microvisor
Average Time(μs)	18.3	30.1	23.7

4.1.2 Micro Benchmarks

We used the LMbench benchmark suite to evaluate the performance of *fork()+exec()*, *fork()+exit()*, pipe and syscall. As shown in Table 2, the performance of executing a simple syscall is the most severely impacted because its execution path is very simple. The other benchmark programs involve fair amounts of work that is executed in the guest OS, thus the performance degradation is a little severe. Among them, Para-virtualization solution got a relatively good result.

Table 2. Preliminary Performance

	Full Virtualization	Para Virtualization	Microkernel
fork + exit (μs)	4,328.53	4012.38	5,117.75
fork + exec (μs)	6,211.51	5,984.14	7,463.90
pipe (μs)	173.30	201.64	1,190.35
syscall (μs)	17.21	13.74	19.93

4.1.3 Macro Benchmarks

In order to see the virtualization's performance impact on common operations in mobile phones, we compared UI loading time, codec performance and image file saving time in table 3. For UI loading test, we used Qtopia installed at NOR flash memory. We prepare 100 files whose size are distributed from 10KB to 5MB to test image file saving and we measure the time taken to save all those image files from a NFS server to NAND flash disk. For codec tests, WMV stream encoder/decoder is used.

Table 3. UI Performance Evaluation

	Full Virtualization	Para Virtualization	Microkernel
UI loading (s)	12.32	13.45	5,117.75
Image saving (s)	45.17	54.23	7,463.90
Encoding rate (fps)	5.67	4.76	1,190.35
Decoding rate (fps)	20.41	23.14	19.93

4.1.4 Scalability Analysis

To analyze the scalability performance impact of the number of concurrent VMs (*n*), we tested four cases: n =1, 2, 3 … 20 for iterated 10 times shown in Fig.4. Root filesystems are mounted as read-only, then we run a daemon process simultaneously on all running VMs to calculate CPU utilization. Average throughputs values of domains are aggregate throughputs don't degrade much as n increases.

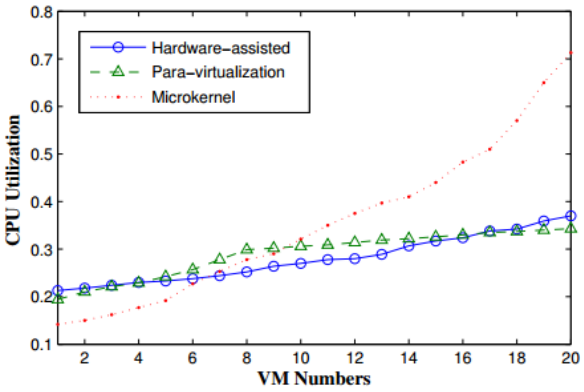

Fig. 4. Scalability performance evaluation

5 Conclusions

Virtualization of mobile devices is becoming a hot research point. Many IT companies such as VMware, OK Labs, Samsung and Red Bend have changed their attentions on this field. In this paper, we described what mobile virtualization is and the benefits it brings. We introduced these solutions in detail and talked about their advantages and limitations. At last, we built an experiment platform and carried out a series of performance evaluation among three open source projects. In the future, we plan to explore several lightweight mobile virtualization solutions, like container technology and multi-execution environments based on detached filesystems.

References

1. Xu, L., Chen, W., Wang, Z., Yang, S.: Smart-DRS: A strategy of dynamic resource scheduling in cloud data center. In: 2012 IEEE International Conference on Cluster Computing Workshops (CLUSTERWORKSHOPS), pp. 120–127. IEEE (2012)
2. Andrus, J., Dall, C., Hof, A.V., Laadan, O., Nieh, J.: Cells: a virtual mobile smartphone architecture. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, pp. 173–187. ACM (2011)
3. Yoo, S., Liu, Y., Hong, C.H., Yoo, C., Zhang, Y.: Mobivmm: a virtual machinemonitor for mobile phones. In: Proceedings of the First Workshop on Virtualizationin Mobile Computing, pp. 1–5. ACM (2008)
4. Heiser, G.: The role of virtualization in embedded systems. In: Proceedings of the1st Workshop on Isolation and Integration in Embedded Systems, pp. 11–16. ACM (2008)
5. Chen, X.: Smartphone virtualization: Status and challenges. In: 2011 International Conference on Electronics, Communications and Control (ICECC), pp. 2834–2839. IEEE (2011)
6. Kizza, J.M.: Mobile systems and their intractable social, ethical and security issues. In: Ethical and Social Issues in the Information Age, pp. 281–297. Springer (2013)
7. Dall, C., Nieh, J.: Kvm for arm. In: Proceedings of the Ottawa Linux Symposium, Ottawa, Canada (2010)