A new framework for baremetal OS

Hypervisor is a Device Abstraction Layer

Iori Yoneji[†‡†̆] iori@osss.cs.tsukuba.ac.jp
Yushi Omote[‡̆] omote@osss.cs.tsukuba.ac.jp
Takahiro Shinagawa[⸸̈] shina@ecc.u-tokyo.ac.jp
Kazuhiko Kato[†̆] kato@cs.tsukuba.ac.jp

†: student, ‡: presenter †̆: University of Tsukuba ⸸̈: The University of Tokyo ‡̆: Japan Society for the Promotion of Science

1. INTRODUCTION

Many devices are introduced in a short period while the society has been saturated with them. Therefore, today's general purpose operating systems have the mission to serve a huge number of device drivers and are needed to increase the number. For example, a snapshot of linux 4.1 consists of 17.6 million lines of code, and more than 60% of it is in "drivers/" directory. Also, 22.7% of code of "drivers" is for network drivers. This fact makes kernel faults to be often, because device drivers are very fragile. Bugs of device drivers are hisotrically the largest number and the most often reason of kernel faults[1], and they are still dominant reason of faults [3]. This situation is hard for newcomers, that aim to develop new efficient or convenient operating systems. Such a competitor has less device drivers, and this means the competitor is less useful in real applications, even if it introduces better operating system design.

Also, each of operating systems has its own abstraction layers to unify different devices of the same class that serve similar functionality. These layers prevent developers from porting device drivers from one to another and limit response time and throughput. Therefore, device drivers from an unique operating system cannot be contributed to another operating system; thus such abstraction layers are not ideal "abstraction".

In past study, LeVasseur et al. have shown that virtualization enables device drivers to be "reused"[2]. While this approach has an advantage that drivers can be reusable without any modification, however, the original operating system the driver is based on is required to be run on the virtual machine as the environment for drivers to be run.

Our goal is to eliminate these abstraction layers and offload device drivers from operating systems. In this poster, we propose a new device abstraction layer out of and independent of any operating system. This layer is an ultimately thin hypervisor to conceal real device and expose para-virtualization device–virtio[4] instead. Virtio, a de-facto para-virtualization device standard, provides low-latency and high-throughput virtual devices for virtualized guest operating systems. By this architecture, the hypervisor takes responsibility for all of control plane and provides high performance data plane. The reason we use hypervisor is that virtualization is necessary to handle virtio access from guest operating systems. Moreover, virtualization technique enables migration of operating system and memory-space isolation of operating system from device drivers to protect the operating system from malbehaving device drivers.

Para-virtualization of network devices in present hypervisors like KVM or Xen, guest operating systems can utilize such devices that is connected to machine-internal virtual network. So packets sent from guest operating system are routed along a long path with overheads. Our abstraction layer does not perform any operating system arbitation, does not newly add virtualized device, but just "falsifies" real device as virtualized device. It is not routing any packets, but does straight translation from virtio to real devices.

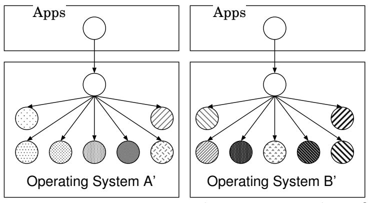

Figure 1: Dependency graph of Present OS and device drivers

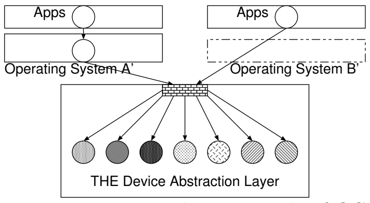

Figure 2: Dependency graph of OS and device drivers with DAL

2. GOALS

2.1 High applicationability

Because virtio-net is a de-facto standard, almost every major operating systems can run on it. Hence, our device abstraction layer can be inserted under almost every operating system without any modification.

2.2 Low overhead

Our abstraction layer is based on a hypervisor named "BitVisor"[5] that runs only one guest operating system. This hypervisor does not emulate or intercept for any kind of device accesses including APIC, ACPI, and BIOS if unnecessary. Because these common and frequently-accessed devices can controlled by guest directly, the accesses can be done without any overhead. Also, device abstraction is without routing or rewriting so it is done in very short time.

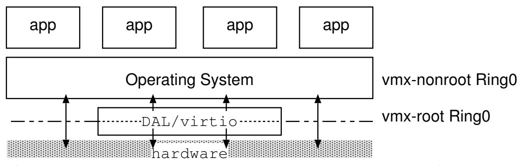

Figure 3: Architecture of combination of Device Abstraction Layer and Operating System

2.3 Isolation and Migration

Practically, real devices have internal and unmodifiable states. This is the reason that nonvirtualized operating systems can not be, or hard to be migrated. Our abstraction layer is a hypervisor, and all internal states of virtual devices are settable and gettable, so the operating system is migratable from one machine to another machine.

3. REFERENCES

[1] Chou, A., et al. An empirical study of operating systems errors. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2001), SOSP '01, ACM, pp. 73–88.

[2] LeVasseur, J., et al. Unmodified device driver reuse and improved system dependability via virtual machines. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6* (Berkeley, CA, USA, 2004), OSDI'04, USENIX Association, pp. 2–2.

[3] Palix, N., et al. Faults in linux: Ten years later. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, 2011), ASPLOS XVI, ACM, pp. 305–318.

[4] Russell, R. Virtio: Towards a de-facto standard for virtual i/o devices. *SIGOPS Oper. Syst. Rev. 42*, 5 (July 2008), 95–103.

[5] Shinagawa, T., et al. Bitvisor: A thin hypervisor for enforcing i/o device security. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (New York, NY, USA, 2009), VEE '09, ACM, pp. 121–130.