Virtualization

Prof. James L. Frankel
Harvard University

Version of 6:19 PM 2-May-2017

Hypervisor emulates architectures

- Type 1: run on bare hardware
- Type 2: built on top of a native OS's system calls

Advantages

- Can run multiple services/applications on one computer even if they require different ISA (instruction set architectures) or OSes
  - Share hardware, reduce cost
- Fault tolerance/Isolation
  - One crash does not affect other systems
- Can migrate applications/services more easily than with physical processors
- Can still support legacy OSes
- Enables developers to develop code and test against a large range of configurations/OSes/etc.
- Easier to switch than with multiboot
- Use virtualization to provide virtual computers on the cloud as needed

Challenge: how to emulate the architecture and maintain good performance

- Safety: Hypervisor has full control of virtualized resources
- Fidelity: Same behavior as on bare hardware
- Efficiency: Much of the code in the VM should run without hypervisor intervention

Long History of VMs

- IBM's VM/370 was released in 1972

Instruction Set Issues

- Sensitive instructions: behave differently in kernel mode vs. user mode
- Privileged instructions: trap if executed in user mode

Efficient Implementation of Sensitive Instructions

- To efficiently implement a VM, sensitive instructions must be a subset of privileged instructions
- An architecture should not allow a user mode program to read sensitive state data

Implementation Strategy

- Trap and emulate!

VMware’s Approach

- VMware was able to virtualize architectures before the constraints above were met!
    - Problematic sequences of code were replaced on-the-fly by safe code sequences (binary translation -- also used to allow CISC instruction sets to be run efficiently on RISC computers)

Paravirtualization

- The VM is visible to the user program
- Calls exist to interact with the VM

Comparison of Type 1 vs. Type 2 Hypervisors

- Type 1 Hypervisors support multiple guest operating systems
- Type 2 Hypervisors (AKA hosted hypervisors) run on top of a host operating system and support multiple guest operating systems

Virtualization of the x86 Architecture

- Virtualization of the x86 architecture was possible by utilizing two of the four protection rings that were unused in x86 OSes
  - This allowed memory accesses to trap to the more privileged ring

- Ring 0: most privileged, Ring 3: least privileged
- Native OS: Ring 0 was OS kernel, Ring 3 was User Processes
- Virtualized: Ring 0 was Hypervisor, Ring 1 was OS kernel, Ring 3 was User

Dynamic Inspection & Rewriting of Code

- Dynamically inspect and possibly re-write basic blocks that include sensitive instructions
  - End each basic block with a trap to the hypervisor so that the next basic block can be inspected
  - Almost all blocks do not contain sensitive instructions
  - Cache the re-written code so that it can be executed repeatedly without need to re-inspection
  - Optimize the end of inspected/translated basic blocks to branch to the next basic block without requiring a trap to the hypervisor