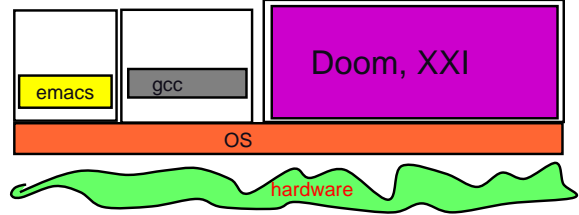


CS 140: Operating Systems

Lecture 27: Virtual Machine Monitors

Mendel Rosenblum

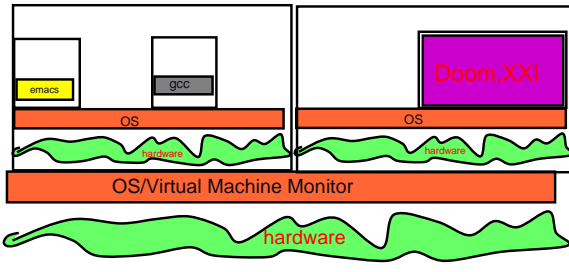
Review: What is an OS?



- ◆ Software between applications and reality:
 - Abstracts hardware and makes portable.
 - Makes finite into (near)infinite.
 - Provides protection.

What If?

- ◆ Process abstraction looked just like hardware!



How is a process different from HW?

- | Process | Hardware |
|---|---|
| ◆ CPU – Non-Privileged registers and instructions. | ◆ CPU – All registers and instructions. |
| ◆ Memory – Virtual memory. | ◆ Memory – Both virtual and physical memory, memory management, TLB/page tables, etc. |
| ◆ Exceptions – signals, errors. | ◆ Exceptions – Trap architecture, interrupts, etc. |
| ◆ I/O - File System, Directory, Files, raw devices. | ◆ I/O – I/O devices accessed using programmed I/O, DMA, interrupts. |

One Way: Complete Machine Simulation

- ◆ Build a simulation of all the hardware.
 - CPU** – A loop that fetch an instruction, decode it, simulate its effect on the machine, state.
 - Memory** – Physical memory is just an array, simulate the MMU on all memory accesses.
 - I/O** – Simulate I/O devices, programmed I/O, DMA, interrupts.
- ◆ Problem: Too slow!
 - 100x slowdown makes it not too useful.
 - CPU/Memory – 100x CPU/MMU simulation.
 - I/O Device – <2x slowdown.
- ◆ Need to emulate CPU/MMU fast enough.

Making a process look like hardware - CPU

- ◆ Observations: Most instructions are the same regardless of processor privileged level.
 - Example: `inc %eax`
- ◆ Why not just give CPU to execute the instructions?
 - Safety** – How we going to get it back? Or stop it from stepping on us? How about CLI/HALT?
 - Answer: Use protection mechanism.
- ◆ Run virtual machine directly on CPU at non-privileged level.
 - Most instruction just work.
 - Privileged instructions trap into monitor and run simulator on instruction.
 - Makes some assumptions about architecture.

CPU Trap architecture virtualization

- ◆ What happens when an interrupt or trap occurs.
Like all OSes: we trap into the monitor.
- ◆ What if the interrupt or trap should go to the VM?
Example: Page fault, illegal instruction, system call, interrupt.
- ◆ Run the simulator again.
X86 example: Lookup trap vector in VM's IDT.
Push cs, eip, eflags, on stack.
Switch to privileged mode.

CPU Virtualization Requirements

- ◆ Need protection levels to run VMs and monitors
- ◆ All unsafe/privileged operations should trap
Example: disable interrupt, access I/O dev, ...
x86 problem: POPF (different semantics in different rings)
- ◆ Privilege level should not be visible to software
Software in VM should be able to query and find its in a VM
x86 problem: MOV ax, cs
- ◆ Trap should be transparent to software in VM
Software in VM should be able to tell if instruction trapped.
x86 problem: traps can destroy machine state.
- ◆ Lost art
Re-found - Intel's VT

Virtualization Requirements - Virtualizing Memory

- Basic MMU functionality:
OS manages physical memory (0...MAX_MEM).
OS sets up page tables mapping VA->PA.
CPU accesses VA to should go to PA. Paging off: PA=VA.
Used for every instruction fetch, load, or store.
- Need to implement a *virtual* physical memory
Logically need additional level of indirection
VM's VA -> VM's PA -> machine address
- Trick: Use hardware MMU to simulate virtual MMU.
Can be folded into page tables: VA->machine address

MMU Virtualization

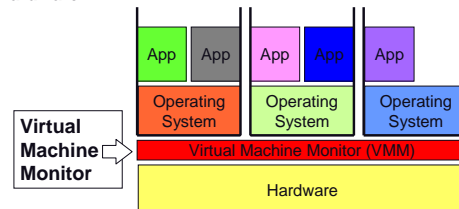
- ◆ Trick: Monitor keeps shadow of VM's page table
Contains mapping to physical memory allocated for that VM.
Access causes Page Fault:
Lookup in VM's page table mapping from VPN to PPN.
Determine where PPN is in machine memory (MPN).
◆ Monitor can demand page the virtual machine
Insert mapping from VPN->MPN into shadow page table.
- ◆ Uses hardware protection
Monitor never maps itself into VM's page table
Monitor never maps memory allocated to other VMs in VM's page table
- ◆ AMD's Nested Page Tables

I/O device virtualization

- ◆ Type of communication:
Special instruction – IN/OUT.
Memory mapped I/O (PIO).
Interrupts.
DMA.
- ◆ Virtualization
Make IN/OUT and PIO trap into monitor.
Run simulation of I/O device.
- ◆ Simulation:
Interrupt – Tell CPU simulator to generate interrupt.
DMA – Copy data to/from physical memory of virtual machine.

Virtual Machine Monitor

- ◆ Thin layer of software that virtualizes the hardware
Exports a virtual machine abstraction that looks like the hardware.



Old idea from the 1960s

- ◆ IBM VM/370 – A VMM for IBM mainframe
 - Multiplex multiple OS environments on expensive hardware.**
 - Desirable when few machine around.**
- ◆ Interest died out in the 1980s and 1990s.
 - Hardware got cheap.**
 - Compare Windows NT verses N DOS machines**
- ◆ Interesting again today
 - Difference problems today – software management**
 - VMM attributes still relevant**

Virtual Machine Monitor attributes

- ◆ Software compatibility
 - Runs pretty much all software**
 - Trick: Make virtual hardware match real hardware.**
- ◆ Low overheads/High performance
 - Near “raw” machine performance.**
 - Direct execution of CPU/MMU.**
- ◆ Complete isolation
 - Total data isolation between virtual machines.**
 - Use hardware protection.**
- ◆ Encapsulation
 - Virtual machines are not tied to physical machines.**
 - Checkpoint/Migration.**

Different thought about OSes

- ◆ Installing software on hardware is broken
 - Tight coupling of OS and applications to hardware creates management problems.**
- ◆ Want to subdivide OS:
 - Hardware drivers.**
 - Hardware management.**
 - System support software.**
- ◆ Turn OSes into normal software that can be managed

Backward compatibility with VMMs

- ◆ Backward compatibility is bane of new OSes.
 - Huge effort require to innovate but not break.**
- ◆ Recent security consideration make it impossible
 - Choice: Close security hole and break apps or be insecure**
- ◆ Example: Not all WinNT applications run on WinXP.
 - In spite of a huge effort to make WinXP compatible.**
 - Given the number of applications that run on WinNT, practically any change will break something.**
 - If (OS == WinNT),....**
- ◆ Solution: Use a VMM to run both WinNT and WinXP
 - Obvious for OS migration as well: Windows -> Linux**

Isolation: Access to Classified Networks

- ◆ Traditional tension: Security vs. Usability
 - Secure systems tend not to be that usable.**
 - Flexible systems are not that secure.**
- ◆ Additional information assurance requirement:
 - Data cannot flow between networks of different classification.**
- ◆ Solution: Run two VMs:
 - Classified VM**
 - Internet VM**
- ◆ Use isolation property to isolate two VMs
 - VMM has control of the information flow between machines**
 - Declassifier mechanism**

Logical partitioning of server machines

- ◆ Run multiple servers on same box
 - Ability to give away less than one machine.**
 - Modern CPUs more power than most services need.**
 - 0.10U rack space machine - Better power, cooling, floor space, etc.**
 - Server consolidation trend: N machine -> 1 real machine.**
- ◆ Isolation of environments
 - Printer server doesn't take down Exchange server.**
 - Compromise of one VM can't get at data of others.**
- ◆ Resource management
 - Provide service-level agreements.**
- ◆ Heterogeneous environments
 - Linux, FreeBSD, Windows, etc.**

Example: Using VMM to enhance security

- ◆ Problem Area: Intrusion Detection Systems (IDS).
- ◆ Trade-offs
 - Host-based IDS (HIDS):**
 - + Good visibility to catch intruder.
 - Weak isolation from intruder disabling/masking IDS.
 - Network-based IDS (NIDS):**
 - + Good isolation from attack from intruder.
 - Weak visibility can allow intruder to slip by unnoticed.
- ◆ Would like visibility of HIDS with isolation of NIDS.
Idea: Do it in the virtual machine monitor.

VMM-based Intrusion Detection System

- ◆ Strong isolation
 - VMM isolate software in VM from VMM.
 - Comprise OS in VM can't disable IDS in VMM.
- ◆ Introspection – Peer inside at software running in VM
 - VMM can see: Physical memory, registers, I/O device state, etc.
 - Signature scan of memory
 - Look through physical memory for patterns or signs of break-in
- ◆ Interposition – Modify VM abstraction to enhance security
 - Memory Access Enforcer
 - Interpose on page protection.
 - NIC Access Enforcer
 - Interpose on virtual network device.

Virtual Appliances

- ◆ Virtualization decouples software from hardware
 - OS no longer an extension of hardware
- ◆ OS is bundled with application
 - Choose based on the needs of the application
- ◆ Virtual Appliance

The Operating System

Traditional View



OS jobs:

1. Drive and manage hardware
2. Export better abstraction

OS is viewed as an extension of hardware

Privileged position – Only one OS

Modern OS Evolution



Goal – Support as many applications as possible

Problems – Too complex

- Security
- Reliability
- Manageability
- Performance
- Innovation

Virtual Appliance Operating System



Don't need complex hardware management

Don't need broad application support

Application-specific operating system

Look at hardware appliance operating systems for examples

Summary

- ◆ Virtualization is having a large impact on operating systems
 - More opportunities
 - Less importance