CSE 451: Operating Systems
Autumn 2009

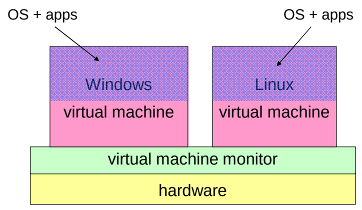
Module 24
Virtual Machine Monitors

Ed Lazowska
lazowska@cs.washington.edu
Allen Center 570

What do Virtual Machine Monitors enable?

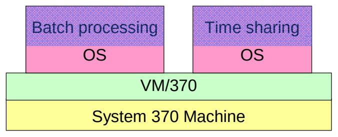
- Running multiple operating systems (and their applications) on a single physical computer, as if each were running on its own private virtual computer
- Contemporary examples
  - VMWare
  - Microsoft's VirtualPC / VirtualServer
  - Parallels (Macintosh)
  - Xen

VMM structure



VMM History

- Conceived by IBM in the late 1960's
  - CP-40, CP-67, VM/360
- Sold continuously since then
- Used first for OS development and debugging, then for time sharing (multiple single-user OS's, plus a few single-job batch OS's), eventually for server consolidation



VMMs Today

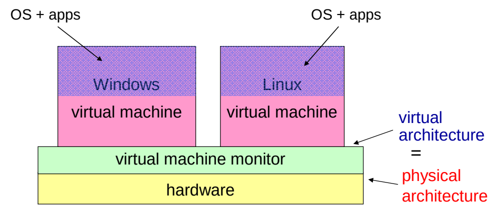
- OS development and debugging
- Software compatibility testing
- Running software from another OS
  - Or, OS version
- Virtual infrastructure for Internet services (server consolidation)
- Two architectures:
  - Type I VMM runs on the raw hardware
    - We'll focus on this approach
  - Type II VMM runs hosted on another OS

Amazon's Elastic Compute Cloud (EC2)

- Provides service developers with a set virtual machines and storage resources
- Scalability
  - New machines can be created in minutes
- Security
  - Virtual machines provide stronger isolation than OS processes
- Developer control
  - Developers choose the OS, software, libraries, etc.
- Low cost
  - Developers pay only for what they use

VMM Implementation Overview

- A VMM is just an operating system that exposes a (virtual) hardware interface



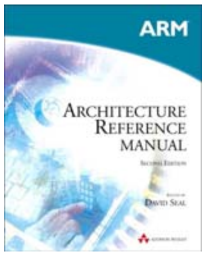
Comparing the Unix and VMM APIs

	UNIX	VMM
Storage	File system	(virtual) disk
Networking	Sockets	(virtual) Ethernet
Memory	Virtual Memory	(virtual) Physical memory
Display	/dev/console	(virtual) Keyboard, display device

Possible Implementation Strategy: Complete machine emulation

- The VMM implements the complete hardware architecture in software

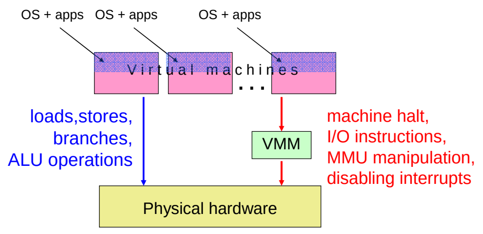
```
while(true) {
  Instruction instr = fetch();

  // emulate behavior in software
  instr.emulate();
}
```



Drawback: This is **really** slow

Alternative: VMM gets control on privileged instructions only

- Treat guest operating systems (and their apps) like an application
  - Guest OS (and its apps) run in user mode
  - Most instructions execute natively on the CPU
  - Privileged instructions are trapped and emulated


OS + apps
OS + apps
OS + apps
Virtual machines
. . .
loads,stores, branches, ALU operations
VMM
machine halt, I/O instructions, MMU manipulation, disabling interrupts
Physical hardware

Virtualizing the User/Kernel Boundary

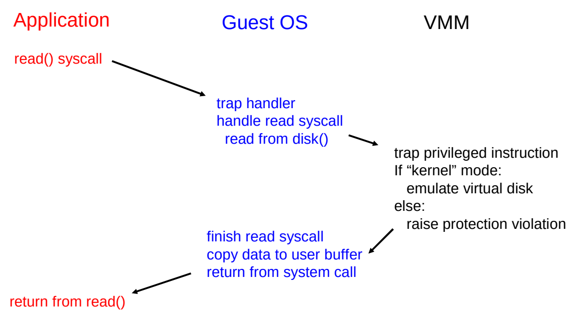
- Both the guest OS **and** applications run in (physical) user-mode
  - This is necessary so that privileged instructions trap into the VMM
- For each virtual machine, the VMM keeps a software mode bit:
  - During a system call, switch to “kernel” mode
  - On system call return, switch to “user” mode

- How does the VMM know to do this? How does it get control??

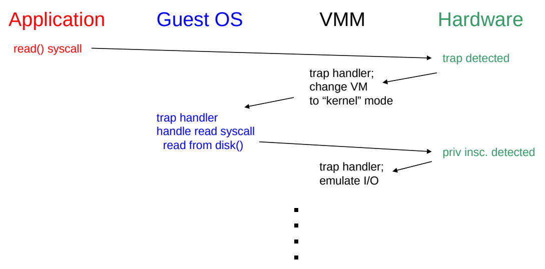
Handling Privileged Instructions

- Virtual machine issues a privileged instruction (e.g., disk read)
- VMM determines whether the virtual machine was in “user” mode or “kernel” mode
  - Note: the virtual mode is distinct from the physical mode (Yikes!)
- If “user” mode, raise a protection exception
  - reflect it to the guest OS just as the hardware would do
    - start executing at the entry point of the interrupt handler of the guest OS
- If “kernel” mode, emulate the disk read in software; then, return control to the guest OS
  - just as would happen after a “real” start I/O operation

Tracing Through a File System Read (simplified conceptual picture



But really …



Questions, to clarify …

- What if the I/O could be handled from the buffer cache?
- Does the VMM handle a VM's I/O request synchronously?
- There are a zillion different types of disks (and networks and …) … Do the device drivers for these reside in the guest OS or in the VMM?

Virtual Disk: Possible Implementations

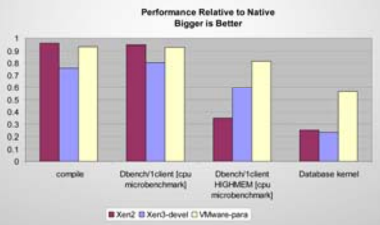
- Static disk partitions
- A file in the file system
  - Especially for type-II VMMs
- A special virtual disk file system
- A network storage abstraction
  - e.g., Amazon's S3

Caveats

- Must be sure that all instructions that modify hardware state are privileged (so that VMM can get control, modify the virtual hardware state for that guest, and not modify the physical hardware state)
  - Not the case on x86!
    - E.g., instructions that work in either user or kernel mode but have different effects depending on the mode
  - So must do binary re-writing
- Binary re-writing is also (part of) the solution for Type II VMM's

- VMM's also utilize memory protection (in addition to privileged instructions) to do their job
- Have not described how memory is virtualized by a VMM, creating "virtual physical memory" for the guest OS's
  - Involves the VMM futzing with the page tables in the guest OS's

VMware Paravirtualization Performance


Performance Relative to Native
Bigger is Better
compile
Dbench/1client [cpu microbenchmark]
Dbench/1client HIGHMEM [cpu microbenchmark]
Database kernel
Xen2
Xen3-devel
VMware-para