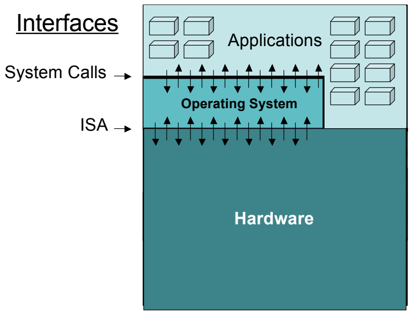
CS 537
Lecture 19
Virtual Machines

Michael Swift

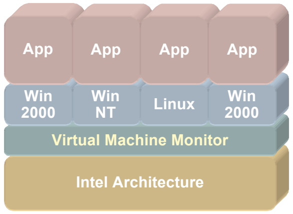
Background Information: Execution Stack



- •Lots of applications
- •One OS

- •Why?
  - •Talks to single instance of HW

Virtual Machines



A thin software layer that sits between Intel hardware and the operating system—virtualizing and managing all hardware resources

Old idea from the 1960s

- IBM VM/370 – A VMM for IBM mainframe
  - Multiple OS environments on expensive hardware
  - Desirable when few machine around
- Popular research idea in 1960s and 1970s
  - Entire conferences on virtual machine monitor
  - Hardware/VMM/OS designed together
- Interest died out in the 1980s and 1990s.
  - Hardware got cheap
  - Operating systems got more more powerful (e.g multi-user)

A return to Virtual Machines

- Disco: Stanford research project (1996-):
  - Run commodity OSes on scalable multiprocessors
  - Focus on high-end: NUMA, MIPS, IRIX
- Hardware has changed:
  - Cheap, diverse, graphical user interface
  - Designed without virtualization in mind
- System Software has changed:
  - Extremely complex
  - Advanced networking protocols
  - But even today :
    - Not always multi-user
    - With limitations, incompatibilities, …

Virtual Machine Monitors

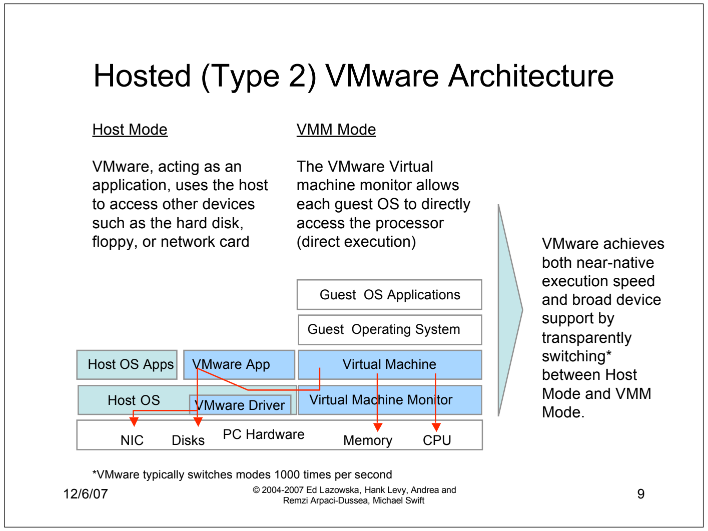
- A **virtual machine monitor** virtualizes the hardware to provide a **virtual machine** in which:
  - All commands/instructions that reference privileged processor state refer to a software copy
  - All commands/instructions that refer to specific physical resources (e.g., memory pages) refer to virtual resources selected by the VMM
  - All commands/instructions that refer to specific physical devices refer to software that implements/emulates that device interface
  - All interrupts from physical devices are handled by VMM
  - VMM must be at higher privilege level than guest VM, which generally runs in user mode
    - ⇒ Execution of privileged instructions handled by VMM
- A VMM implements the hardware interface in software

Virtual Machine Monitors (VMMs)

- Virtual machine monitor (VMM) or hypervisor is software that supports VMs
- VMM determines how to map virtual resources to physical ones
- Physical resource may be time-shared, partitioned, or emulated in software
- VMM much smaller than a traditional OS;
  - Isolation portion of a VMM is ≈ 10,000 lines of code

Virtual Machine Types

- Pure/Para-virtualized
  - Pure virtualized systems present the interface of real, existing HW and can run unmodified operating systems
  - Para virtualized systems present a new, simpler interface but require OS modifications
- Type 1 / Type 2
  - Type 1 VMMs (called Hypervisors) sit just above the HW and virtualize the complete hardware
    - Example: Xen, VMware ESX server
  - Type 2 VMMs run within an OS, and rely on OS services to manage HW
    - Example: QEMU, VMware Worksation



Hosted (Type 2) VMware Architecture

Host Mode

VMware, acting as an application, uses the host to access other devices such as the hard disk, floppy, or network card

VMM Mode

The VMware Virtual machine monitor allows each guest OS to directly access the processor (direct execution)

Guest OS Applications

Guest Operating System

Host OS Apps

VMware App

Virtual Machine

Host OS

VMware Driver

Virtual Machine Monitor

NIC

Disks

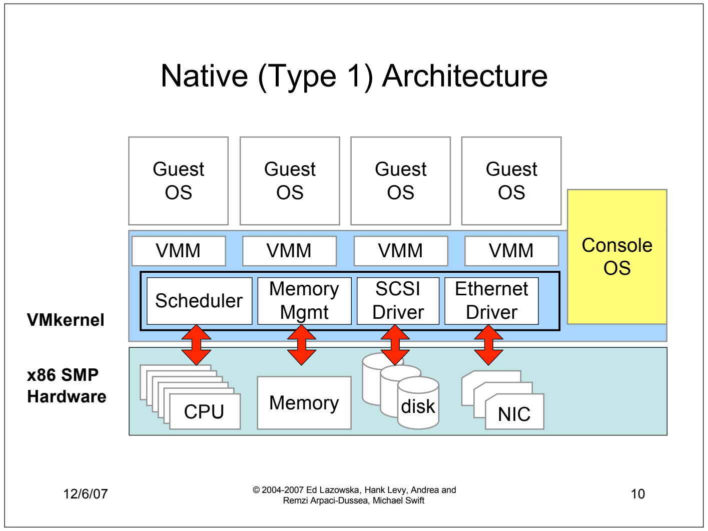
PC Hardware

Memory

CPU

VMware achieves both near-native execution speed and broad device support by transparently switching* between Host Mode and VMM Mode.

*VMware typically switches modes 1000 times per second



Native (Type 1) Architecture

Guest OS

Guest OS

Guest OS

Guest OS

VMM

VMM

VMM

VMM

Console OS

VMkernel

Scheduler

Memory Mgmt

SCSI Driver

Ethernet Driver

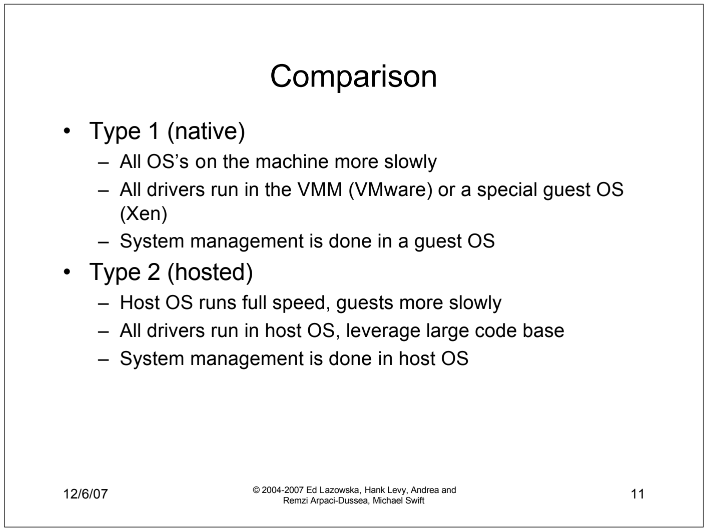
x86 SMP Hardware

CPU

Memory

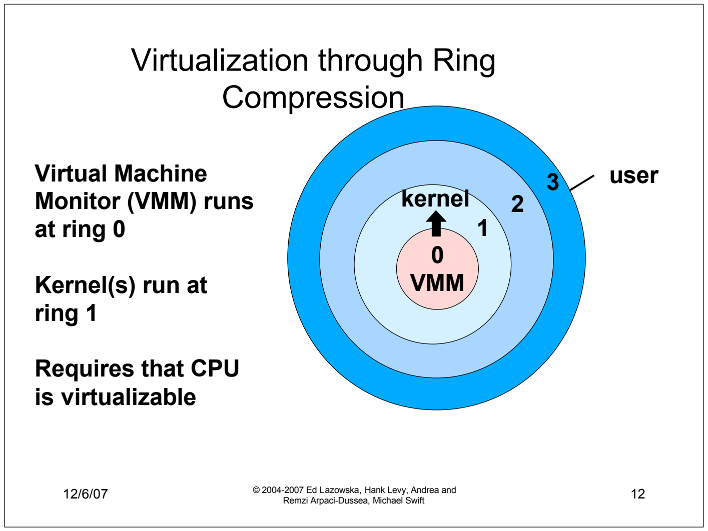
disk

NIC



Comparison

- Type 1 (native)
  - All OS's on the machine more slowly
  - All drivers run in the VMM (VMware) or a special guest OS (Xen)
  - System management is done in a guest OS
- Type 2 (hosted)
  - Host OS runs full speed, guests more slowly
  - All drivers run in host OS, leverage large code base
  - System management is done in host OS



Virtualization through Ring Compression

Virtual Machine Monitor (VMM) runs at ring 0

Kernel(s) run at ring 1

Requires that CPU is virtualizable

user

3

2

kernel

1

0

VMM

Virtualization Technology

- Basic approach: execute privileged software at unprivileged level
  - Privileged instructions will trap: I/O, memmgmt
  - Emulate behavior of privileged instructions in software in VMM
- VMM has complete control over the HW
  - Presents another layer of virtual memory under the OS with a separate page table
  - Presents a different set of devices to the OS
- What happens to instructions that return different results in priv. mode and normal mode?

Classification of processor architectures

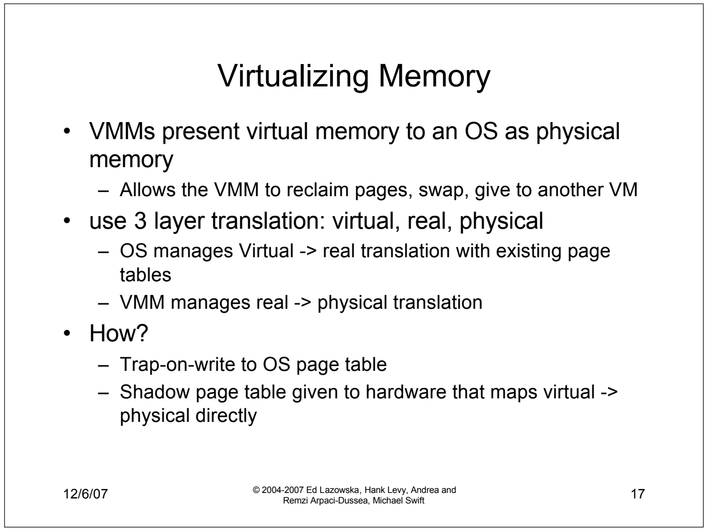
- Strictly virtualizable processor architectures
  - Can build a VMM based on trap emulation exclusively
    - No software running inside the VM cannot determine the presence of the VMM (short of timing attacks)
  - Examples: IBM S/390, DEC Compaq Intel Alpha, PowerPC
- (Non-strictly) virtualizable processor architectures
  - Trap emulation alone is not sufficient and/or not complete
    - E.g. instructions have different semantics at various levels (sufficient)
    - E.g Some software sequences can determine the presence of the VMM (complete)
  - Examples: IA-32, IA-64
- Non virtualizable processor architectures
  - Basic component missing (e.g. MMU, …)

ISA Impact on Virtual Machines

- Consider x86 PUSHF/POPF instructions
  - Push flags register on stack or pop it back
  - Flags contains condition codes (good to be able to save/restore) but also interrupt enable flag (IF)
- Pushing flags isn’t privileged
  - Thus, guest OS can read IF and discover it’s not the way it was set
    - VMM isn’t invisible any more
- Popping flags in user mode ignores IF
  - VMM now doesn’t know what guest wants IF to be
  - Should trap to VMM
- Possible solution: modify code, replacing pushf/popf with special interrupting instructions
  - But now guest can read own code and detect VMM

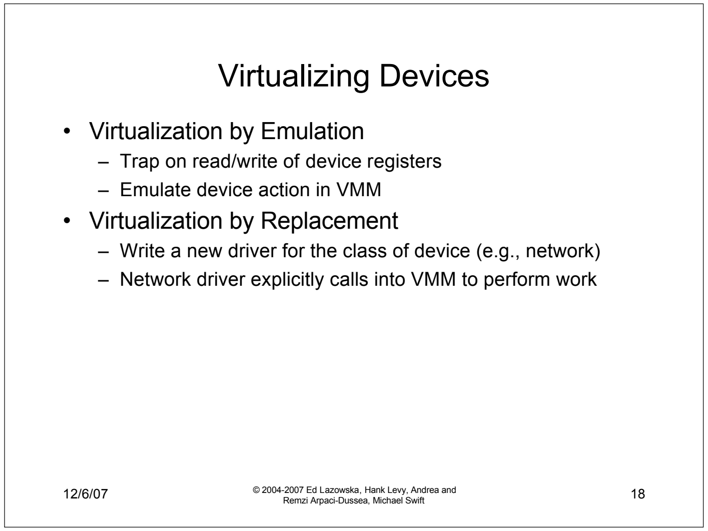
Virtualizing x86

- Binary translation
  - Convert kernel code into a new binary that calls into VMM for all privileged instructions / instructions that do something different between kernel/user mode (VMware)
- Emulation
  - Emulate all instructions in kernel mode (VirtualPC)
- ParaVirtualization
  - Change kernel code to avoid all privileged instructions
  - Issue explicit **HyperCalls** into VMM to provide these services
- New hardware
  - Intel VT, AMD Pacifica adds new ring (-1) that traps correctly



Virtualizing Memory

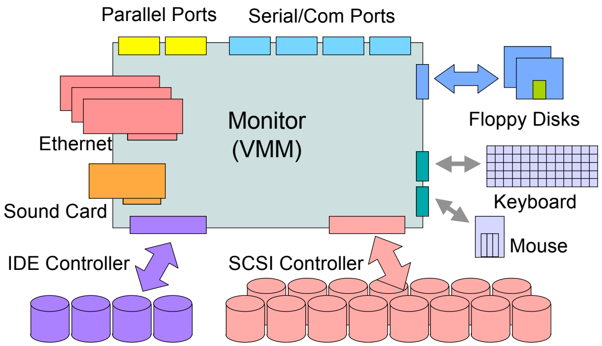
- VMMs present virtual memory to an OS as physical memory
  - Allows the VMM to reclaim pages, swap, give to another VM
- use 3 layer translation: virtual, real, physical
  - OS manages Virtual -> real translation with existing page tables
  - VMM manages real -> physical translation
- How?
  - Trap-on-write to OS page table
  - Shadow page table given to hardware that maps virtual -> physical directly



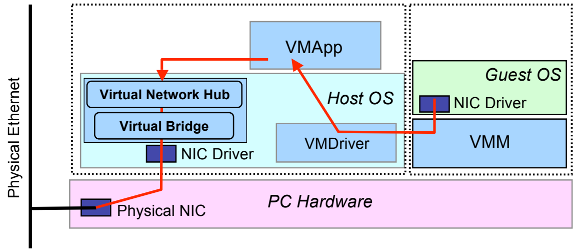
Virtualizing Devices

- Virtualization by Emulation
  - Trap on read/write of device registers
  - Emulate device action in VMM
- Virtualization by Replacement
  - Write a new driver for the class of device (e.g., network)
  - Network driver explicitly calls into VMM to perform work

Virtual Hardware

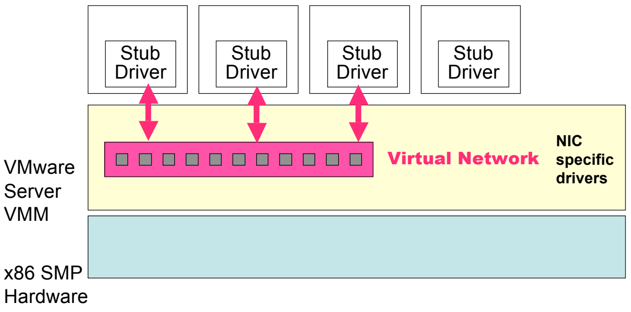


Virtualizing a Network Interface



Intra-system networking

- Executes at memory speed



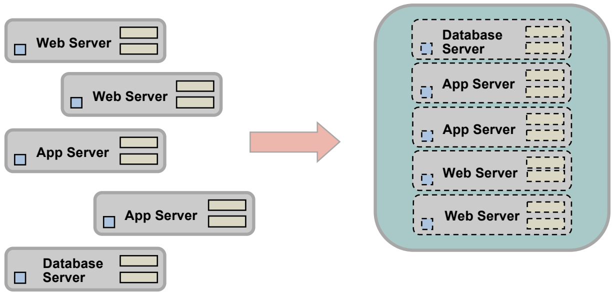
Virtualizing Disks

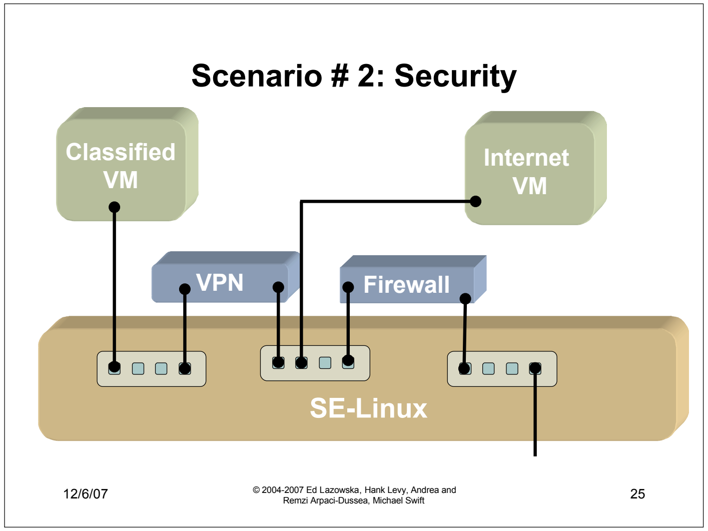
- Sharing
  - Networking shared a single device through time multiplexing
  - Disks share through space multiplexing
  - Some device might not be shared, but just assigned to a single VMM, which can run the driver itself
    - USB flash drive
- VMM makes a file in the FS act like a disk to the VMM
  - Can grow incrementally as disk is used
  - Can be copied between systems
- Done by implementing a SCSI or IDE device that talks to the FS

Uses of Virtual machines

- Suspend/resume
  - All OS state controlled by VMM, so it can be saved to disk
- Server consolidation
  - Take 10 servers, run all 10 in their own OS on one machine
- Testing
  - Run all test platforms on one machines
- Security
  - Run insecure apps in one VM
  - Run secure apps in another VM with strong firewall around it
- Migration
  - Move a VM running on one machine

Scenario # 1: Server Consolidation





Scenario # 2: Security

12/6/07