COS 318: Operating Systems

Virtual Machine Monitors

Kai Li and Andy Bavier
Computer Science Department
Princeton University

http://www.cs.princeton.edu/courses/archive/fall13/cos318/

Introduction

- Have been around since 1960's on mainframes
  - used for multitasking
  - Good example – VM/370
- Have resurfaced on commodity platforms
  - Server Consolidation
  - Web Hosting centers
  - High-Performance Compute Clusters
  - Managed desktop / thin-client
  - Software development / kernel hacking

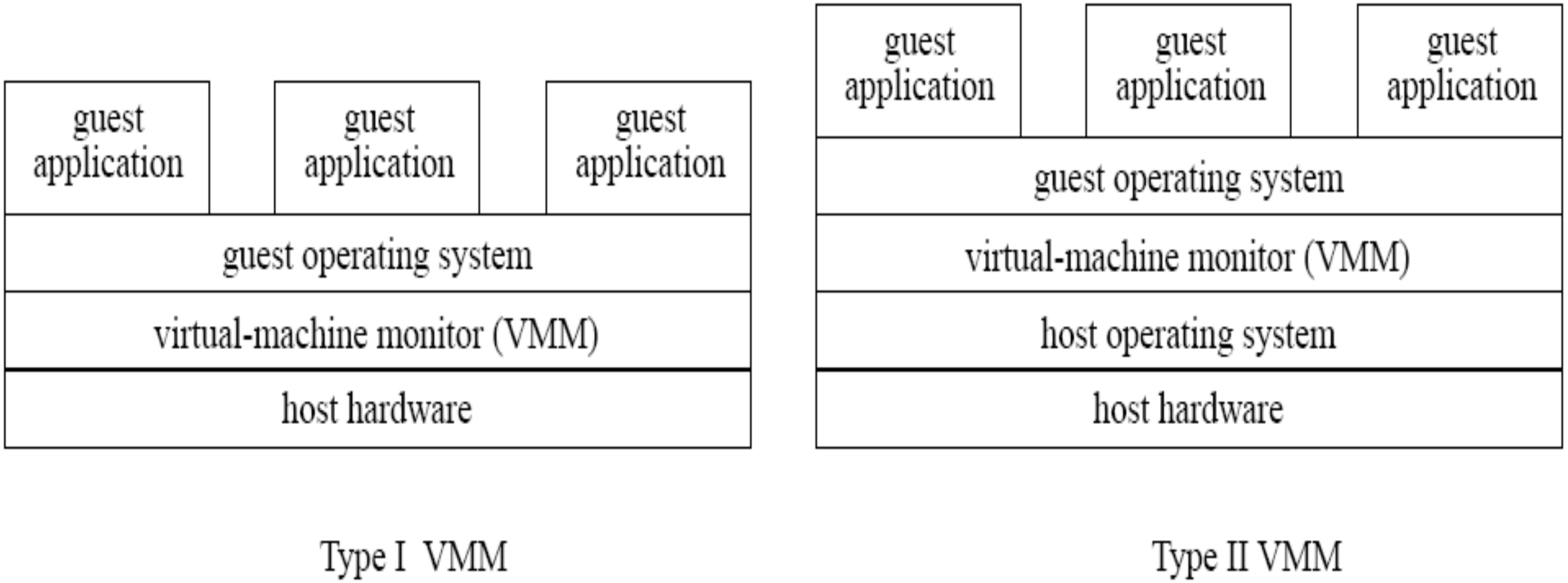
Goals

- Manageability
  - Ease maintenance, administration, provisioning, etc.
- Performance
  - Overhead of virtualization should be small
- Power saving
  - Server consolidation
- Isolation
  - Activity of one VM should not impact other active VMs
  - Data of one VM is inaccessible by another
- Scalability
  - Minimize cost per VM

Virtual Machine Monitor (VMM)

- Resides as a layer below the operating system
- Presents a hardware interface to an OS
- Multiplexes resources between several virtual machines (VMs)
- Performance Isolates VMs from each other

VMM Types


guest application
guest application
guest application
guest operating system
virtual-machine monitor (VMM)
host hardware
Type I VMM
guest application
guest application
guest application
guest operating system
virtual-machine monitor (VMM)
host operating system
host hardware
Type II VMM

Virtualization Styles

- Fully virtual*izing* VMM
  - Virtual machine looks exactly like a physical machine
  - Run guest OS unchanged
  - VMM is transparent to the OS
- Para- virtual*izing* VMM
  - Sacrifice transparency for better performance
  - VMM can provide idealized view of hardware
  - VMM can provide a “hypervisor API”
  - Guest OS is changed to cooperate with VMM

VMM Classification

	Type I	Type II
Fully-virtualized	VMware ESX	VMware Workstation
Para-virtualized	Xen	User Mode Linux

DEI SUB NUMINE VIGET

VMM Implementation

Should efficiently virtualize the hardware

- Provide illusion of multiple machines
- Retain control of the physical machine

Subsystems

- Processor Virtualization
- I/O virtualization
- Memory Virtualization

Processor Virtualization

Popek and Goldberg (1974)

- Sensitive instructions: only executed in kernel mode
- Privileged instructions: trap when run in user mode
- CPU architecture is virtualizable only if sensitive instructions are subset of privileged instructions

- When guest OS runs a sensitive instruction, must trap to VMM so it maintains control

Example: System Call

Process	**Operating System**	**VMM**
1.System call: Trap to OS		
		2. Process trapped: call OS trap handler (at reduced privilege)
	3. OS trap handler: Decode trap and execute syscall; When done: issue return-frrom-trap	
		4. OS tried to return from trap; do real return-from-trap
5. Resume execution (@PC after trap)		

x86 Processor Virtualization

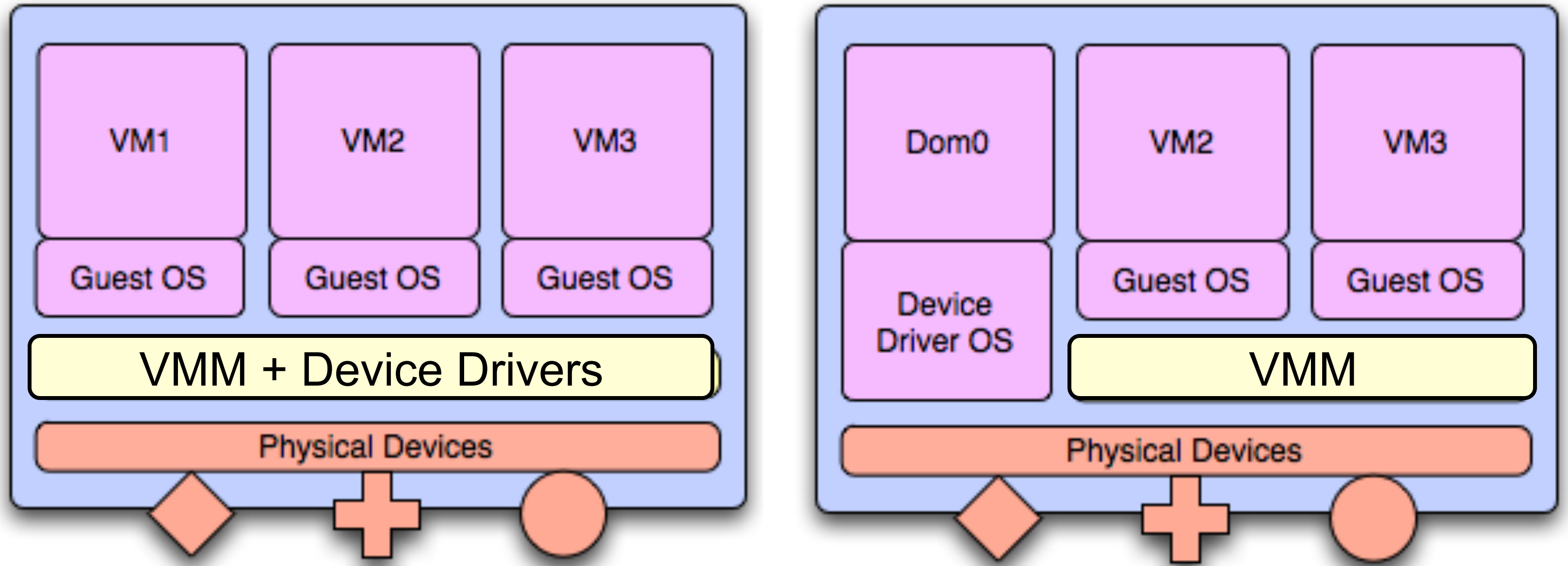
- x86 architecture is not fully virtual*izable*
  - Certain privileged instructions behave differently when run in unprivileged mode
  - Certain unprivileged instructions can access privileged state

- Techniques to address inability to virtualize x86
  - Replace non-virtualizable instructions with easily virtualized ones statically (Paravirtualization)
  - Perform Binary Translation (Full Virtualization)

I/O Virtualization

- Issue: lots of I/O devices
- Problem: Writing device drivers for all I/O device in the VMM layer is not a feasible option
- Insight: Device driver already written for popular Operating Systems
- Solution: Present *virtual* I/O devices to *guest* VMs and channel I/O requests to a trusted *host* VM running popular OS

I/O Virtualization



Memory Virtualization

- Traditional way is to have the VMM maintain a shadow of the VM's page table
- The shadow page table controls which pages of machine memory are assigned to a given VM
- When guest OS updates its page table, VMM updates the shadow

VMware ESX Server

- Type I VMM - Runs on bare hardware
- Full-virtualized – Legacy OS can run unmodified on top of ESX server
- Fully controls hardware resources and provides good performance

ESX Server – CPU Virtualization

- Most user code executes in Direct Execution mode; near native performance
- Uses *runtime* Binary Translation for x86 virtualization
  - Privileged mode code is run under control of a Binary Translator, which emulates problematic instructions
  - Fast compared to other binary translators as source and destination instruction sets are nearly identical

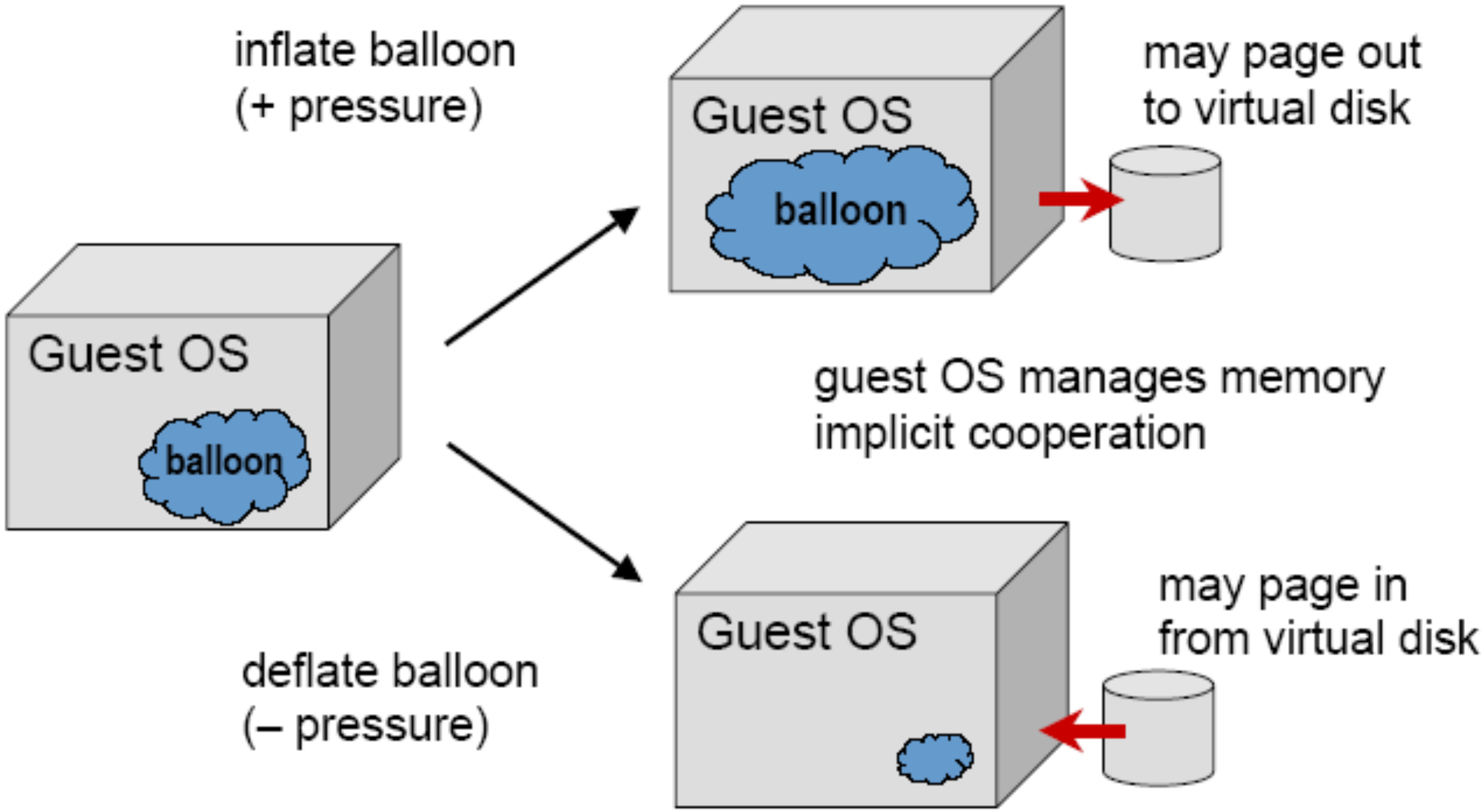
ESX Server – Memory Virtualization

- Maintains shadow page tables with virtual to machine address mappings.
- Shadow page tables are used by the physical processor
- ESX maintains the pmap data structure for each VM with "physical" to machine address mappings
- ESX can easily remap a machine page

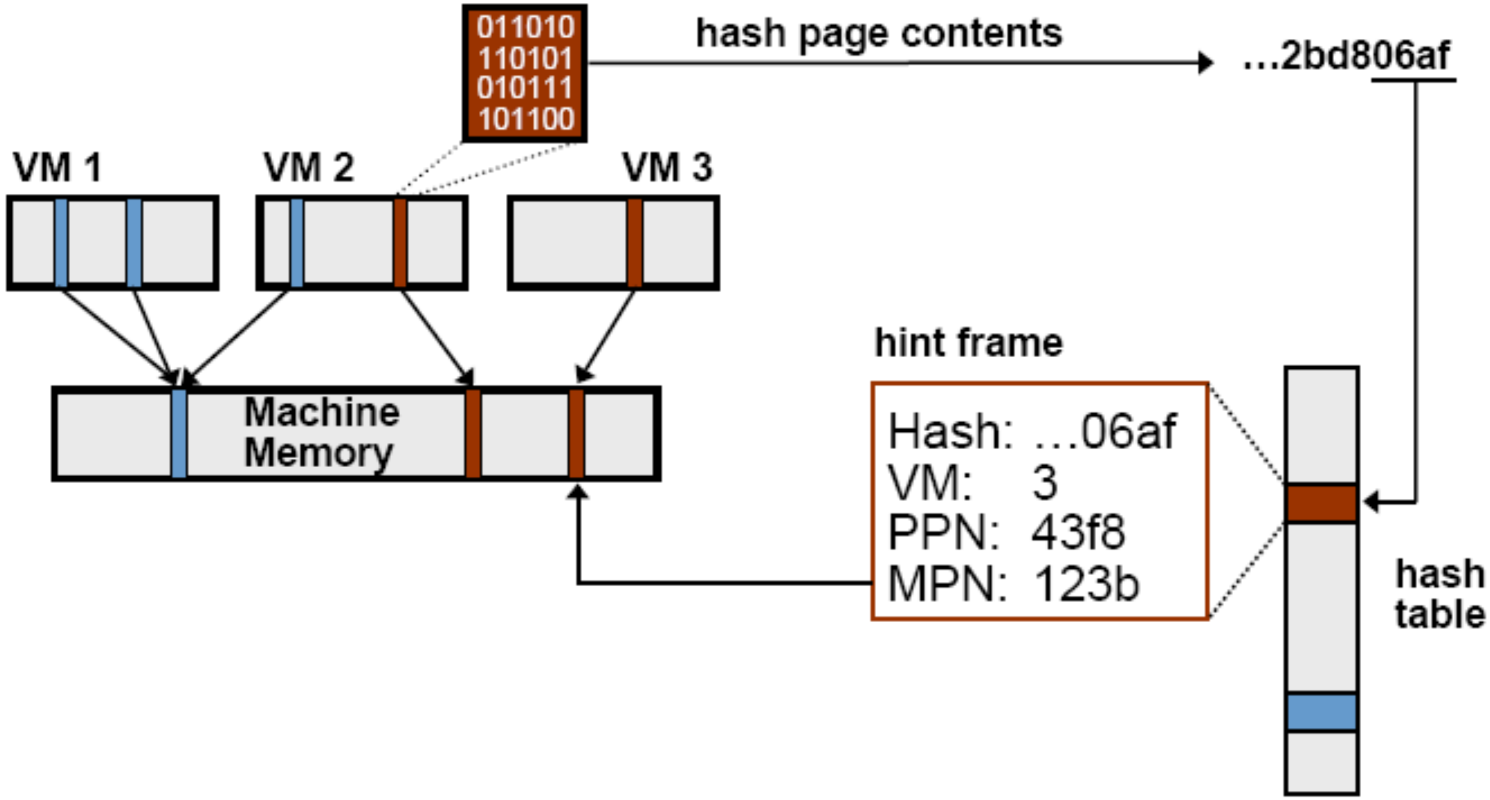
ESX Server – Memory Mgmt

- Page reclamation – Ballooning technique
  - Reclaims memory from other VMs when memory is overcommitted
- Page sharing – Content based sharing
  - Eliminates redundancy and saves memory pages when VMs use same operating system and applications

ESX Server- Ballooning



ESX Server – Page Sharing



Real World Page Sharing

		Total	Saved	
Workload	Guest Types	MB	MB	%
Corporate IT	10 Windows	2048	673	32.9
Nonprofit Org	9 Linux	1846	345	18.7
VMware	5 Linux	1658	120	7.2

Corporate IT – database, web, development servers (Oracle, Websphere, IIS, Java, etc.)

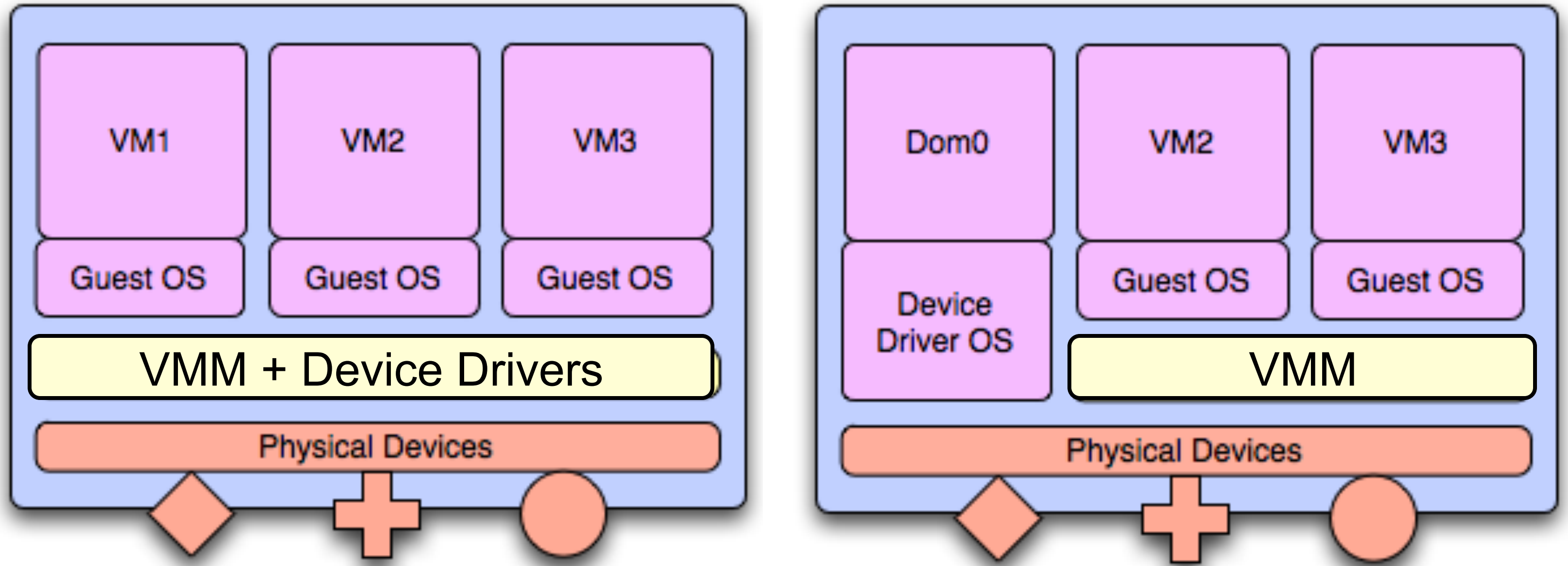
Nonprofit Org – web, mail, anti-virus, other servers (Apache, Majordomo, MailArmor, etc.)

VMware – web proxy, mail, remote access (Squid, Postfix, RAV, ssh, etc.)

ESX Server – I/O Virtualization

- Has highly optimized storage subsystem for networking and storage devices
  - Directly integrated into the VMM
  - Uses device drivers from the Linux kernel to talk directly to the device
- Low performance devices are channeled to special “host” VM, which runs a full Linux OS

I/O Virtualization



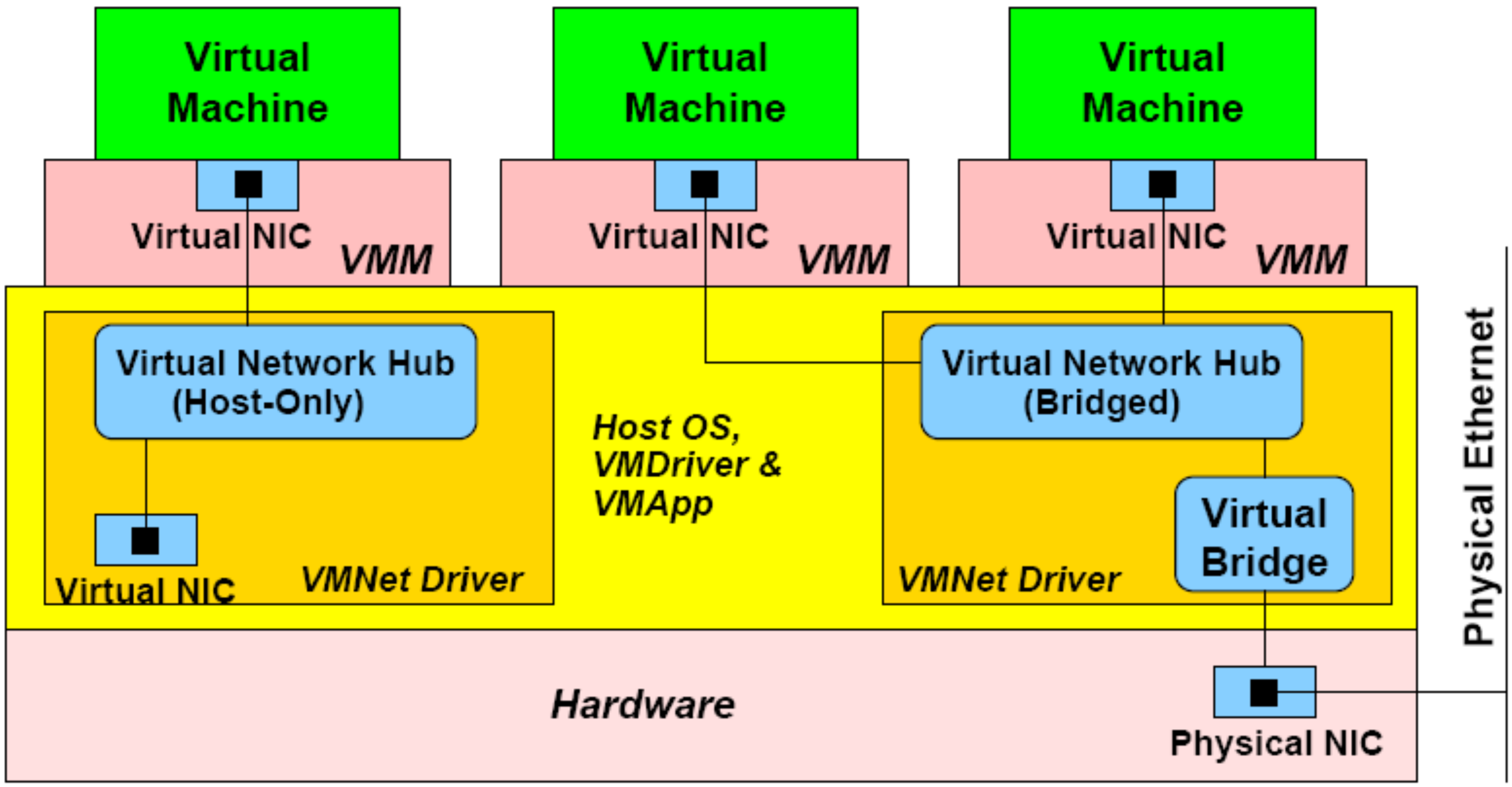
VMware Workstation

- Type II VMM - Runs on host operating system
- Full-virtualized – Legacy OS can run unmodified on top of VMware Workstation
- Appears like a process to the Host OS

Workstation - Virtualization

- CPU Virtualization and Memory Virtualization
  - Uses Similar Techniques as the VMware ESX server
- I/O Virtualization
  - Workstation relies on the Host OS for satisfying I/O requests
  - I/O incurs huge overhead as it has to switch to the Host OS on every IN/OUT instruction.
  - E.g., Virtual disk maps to a file in Host OS

Workstation – Virtualize NIC



Xen

- Type I VMM
- Para-virtualized
- Open-source
- Designed to run about 100 virtual machines on a single machine

Xen – CPU Virtualization

- Privileged instructions are para-virtualized by requiring them to be validated and executed with Xen
- Processor Rings
  - Guest applications run in Ring 3
  - Guest OS runs in Ring 1
  - Xen runs in Ring 0

Xen – Memory Virtualization(1)

- Initial memory allocation is specified and memory is statically partitioned
- A maximum allowable reservation is also specified.
- Balloon driver technique similar to ESX server used to reclaim pages

Xen – Memory Virtualization(2)

- Guest OS is responsible for allocating and managing hardware page table
- Xen involvement is limited to ensure safety and isolation
- Xen exists in the top 64 MB section at the top of every address space to avoid TLB flushes when entering and leaving the VMM

Xen – I/O Virtualization

- Xen exposes a set of clean and simple device abstractions
- I/O data is transferred to and from each domain via Xen, using shared memory, asynchronous buffer descriptor rings
- Xen supports lightweight event delivery mechanism used for sending asynchronous notifications to domains

Summary

- Classifying Virtual Machine Monitors
  - Type I vs. type II
  - Full vs. para-virtualization
- Processor virtualization
- Memory virtualization
- I/O virtualization