Virtualization

COMS W4118
Prof. Kaustubh R. Joshi
krj@cs.columbia.edu

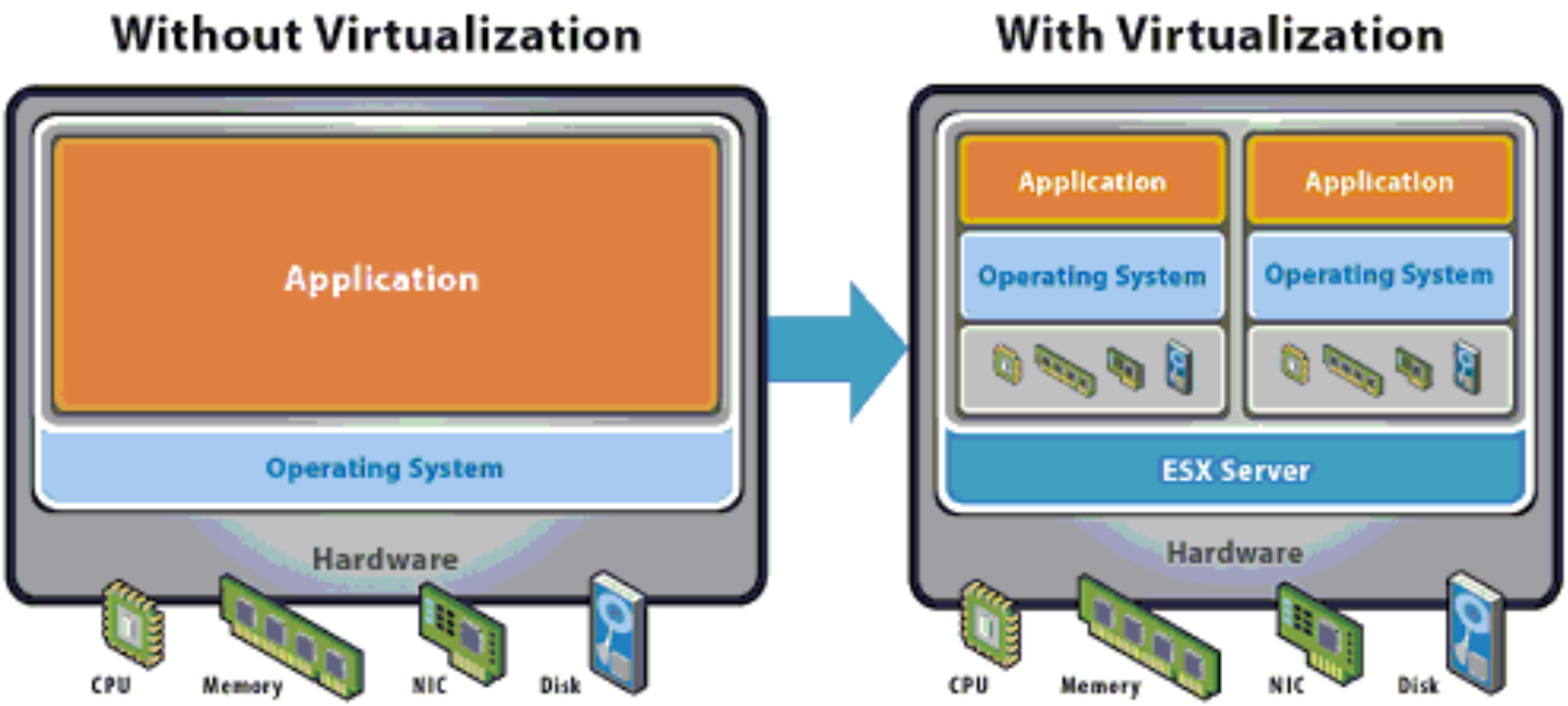
http://www.cs.columbia.edu/~krj/os

References: Operating Systems Concepts (9e), Linux Kernel Development, previous W4118s

Outline

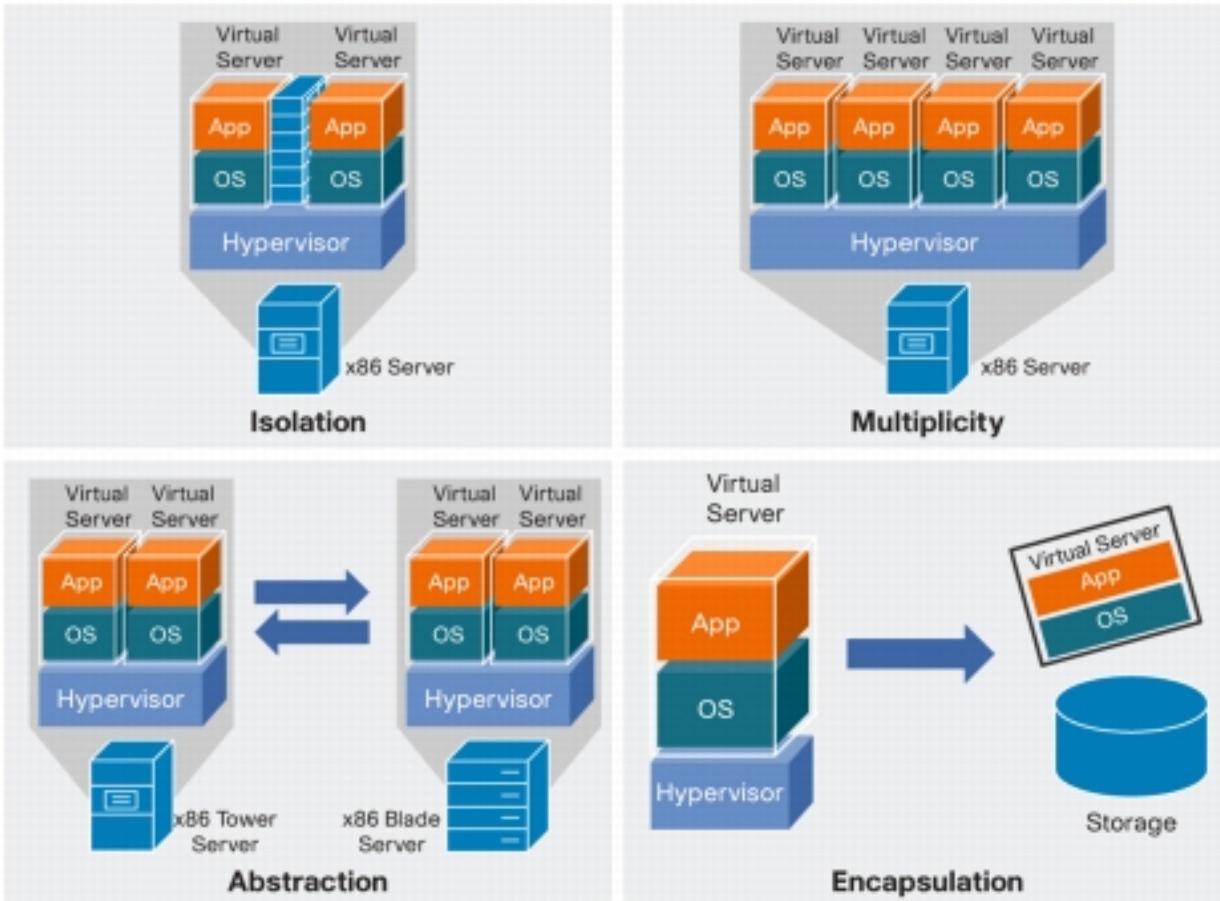
- History and uses
- How virtualization works
- Architectures
- Issues
  - Efficient sharing
  - Isolation
  - Performance

What's Server Virtualization?



Source: VMWare (www.vmware.com)

Why Virtualization?



Food for thought: recall the functions of an OS. What's different?

Source: VMWare (www.vmware.com)

The birth of virtualization

Early Beginnings: Mainframe Era

- Share expensive mainframes among multiple users
  - Support timesharing users (MIT, Bell Labs, …)
  - Do research on virtual memory systems
- 1964: IBM's CP-40 research OS
  - First support of full virtualization
  - Multiple (up to 14) CMS guest VMs supported
- 1966: IBM's CP-67
  - First commercial OS with full VM support
- 1972: IBM VM/370
  - First support for nested virtualization (VM in VM)
  - Lives today under z/VM moniker

Late 90’s: Running different Desktop OSes

- DOS/Windows on Macs
  - 1997: VirtualPC for Mac
- Windows, Solaris on Linux
  - VMWare Virtual Platform
- But, x86 architecture not easily virtualizable
  - Several tricks to make it work (later)

April 14, 1997: The Mac Emulates Windows!

by Chris Seibold Apr 14, 2011

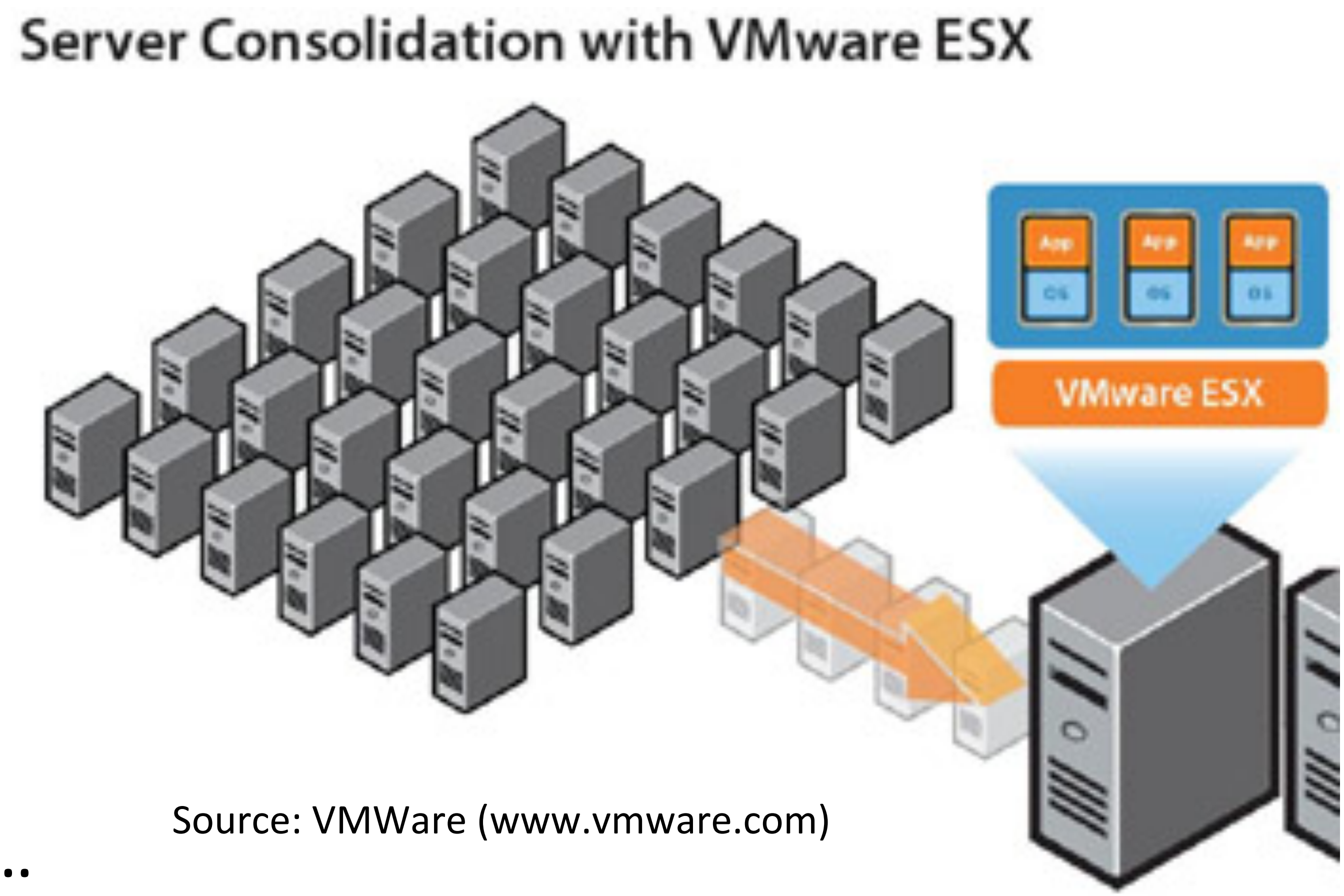
The first home run from Connectix was RAM Doubler, a useful program every Mac 680x0 owner thought they had to have. As the price of RAM predictably dropped, the allure of software that allowed more efficient use of physical RAM decreased. Connectix was smart enough not to stand still; they had something great up their sleeve.

The "something" was Virtual PC. Virtual PC allowed the Mac to run the Windows operating system in emulation. If you were forced to use say, Microsoft Access for a few minutes a day, you could opt to spend those minutes using Virtual PC on your Mac. The performance wasn't as speedy as owning an actual Windows machine but many diehard Mac fans found the tradeoff more than acceptable.

2000's: Resurgence: Server Consolidation

Enterprise Services. Reduce server sprawl.

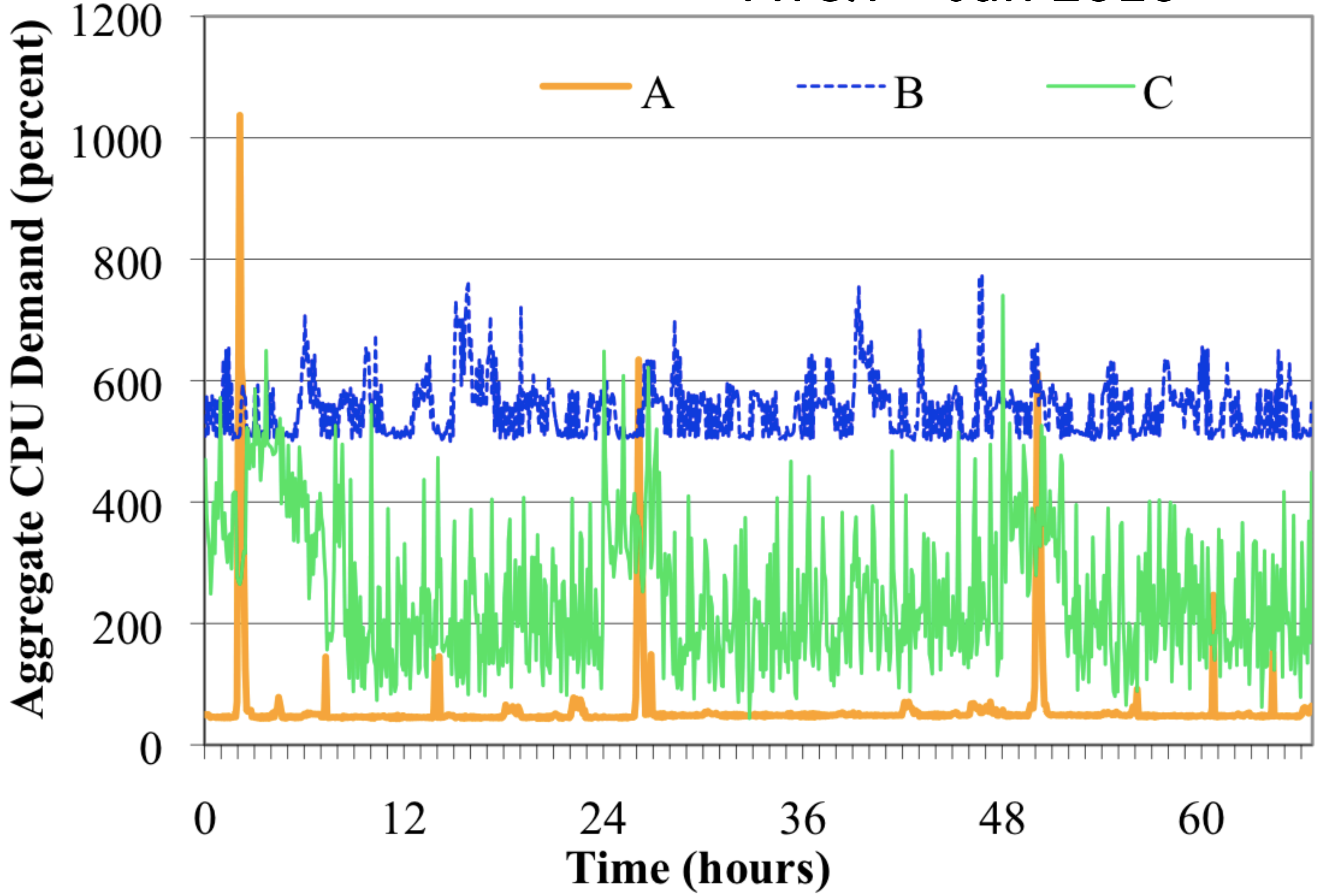
DNS
DHCP
Webserver
Databases
Email
CRM
Windows
Linux
ERP
Accounting, …



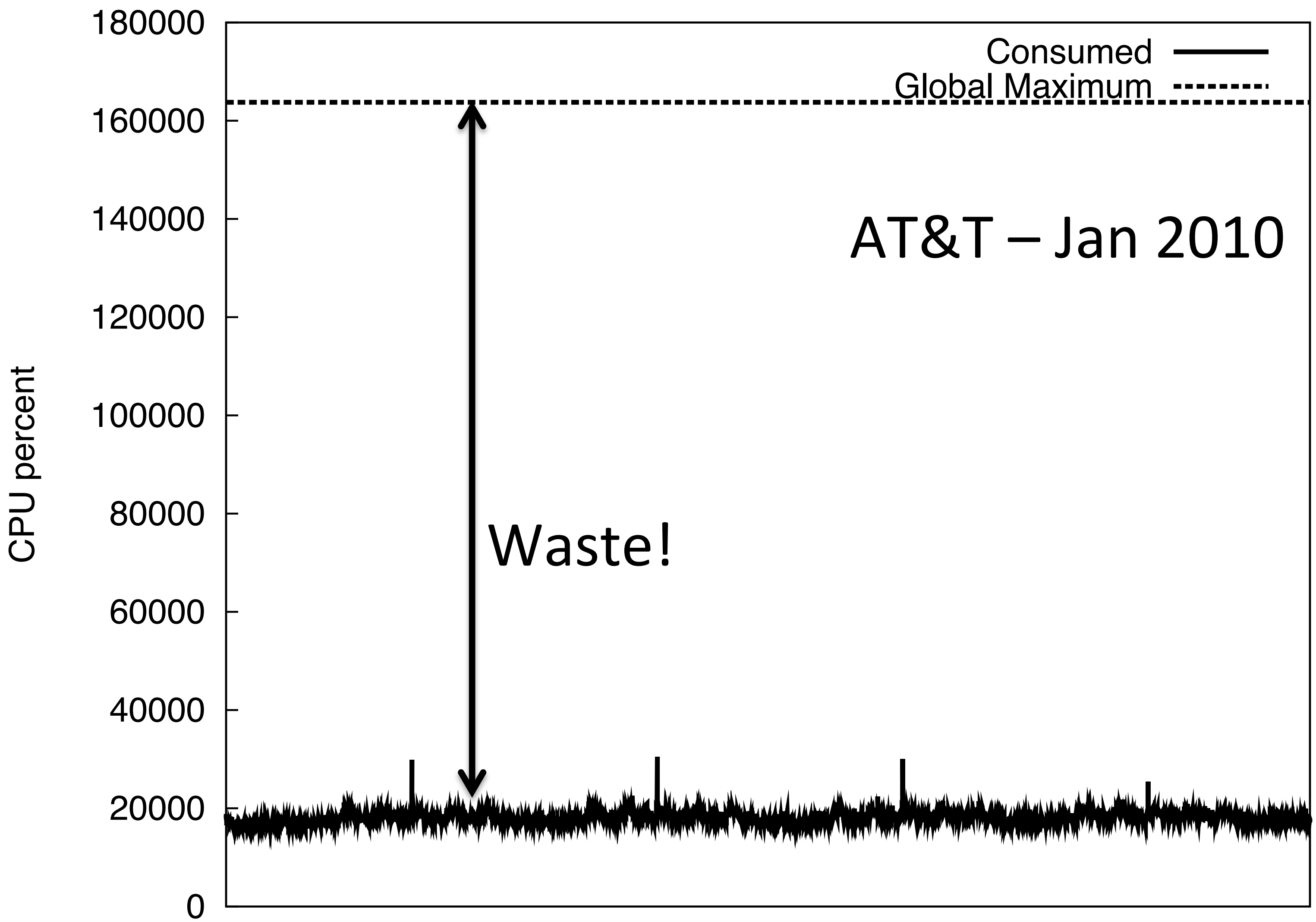
Source: VMWare (www.vmware.com)

Nature of Internet Workloads

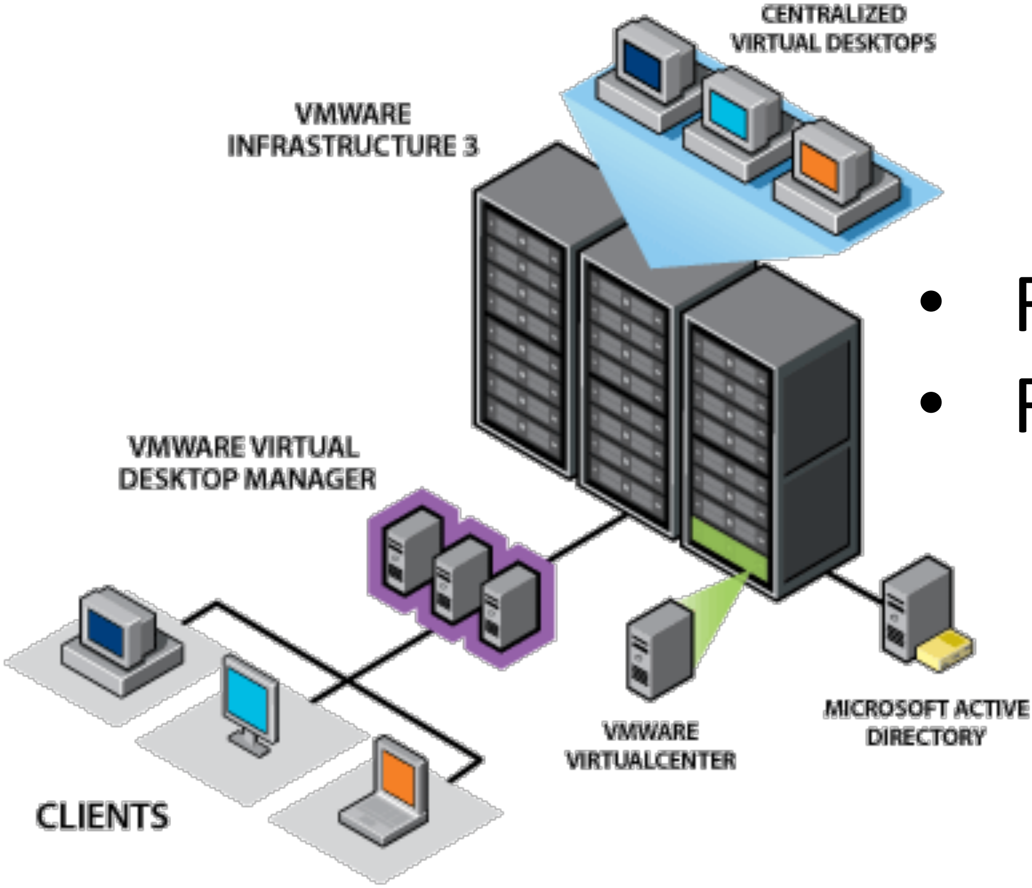
AT&T – Jan 2010



Overprovisioning and Waste



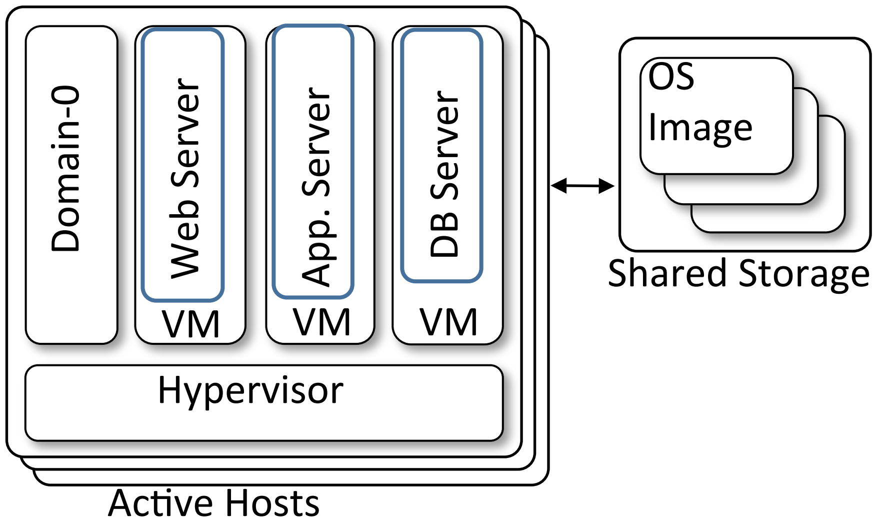
Virtual Desktop Infrastructure (VDI)



- Reduce peak power
- Reduce upgrade costs

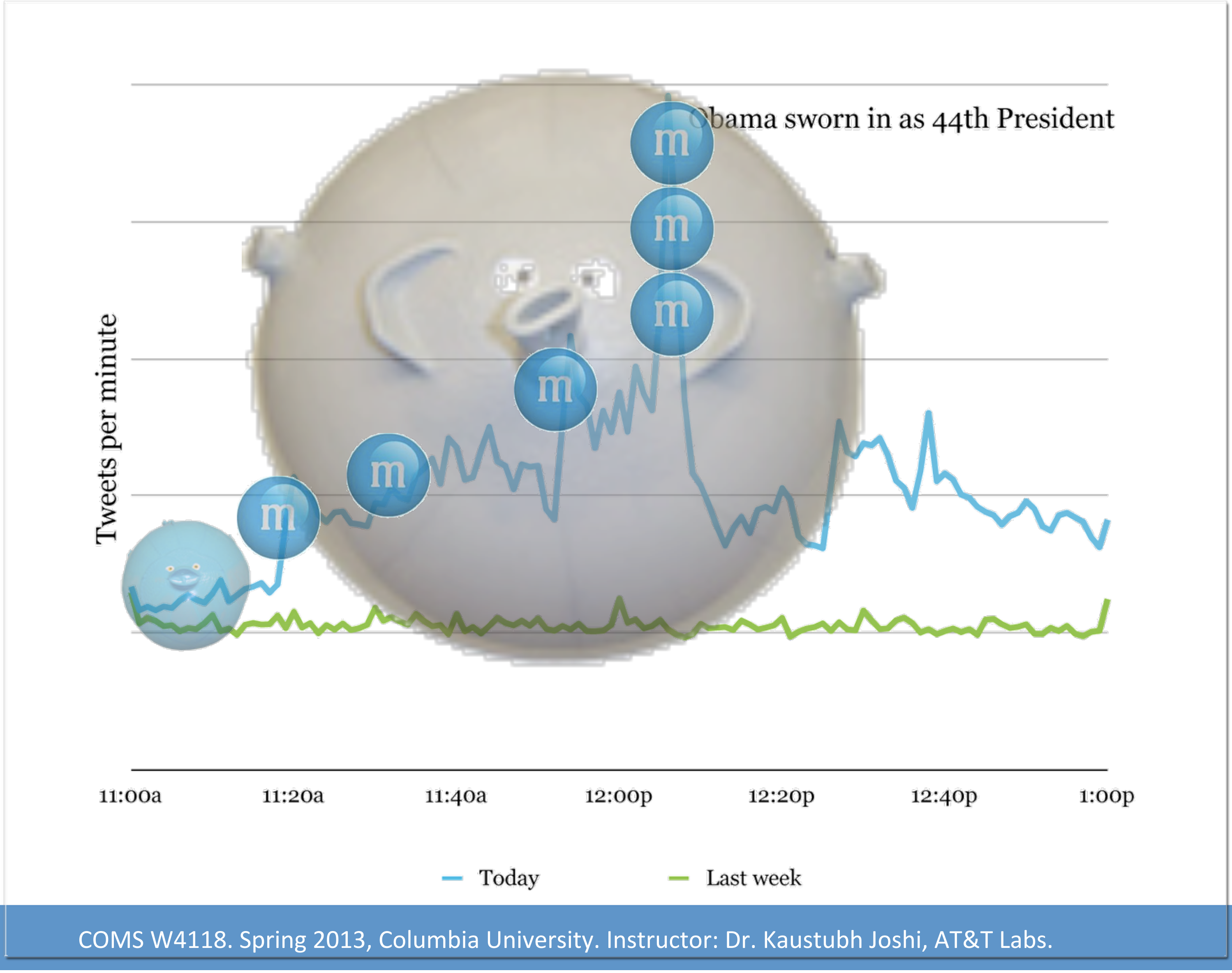
- Multiplex workloads

Infrastructure as a Service (IaaS) Clouds



- Multi-tenant shared environments
  - Spin-up VMs on others' infrastructure
  - Amazon, AT&T, Rackspace, Google, …
  - Isolation, security, flexibility

Scaling Compute based on Workload

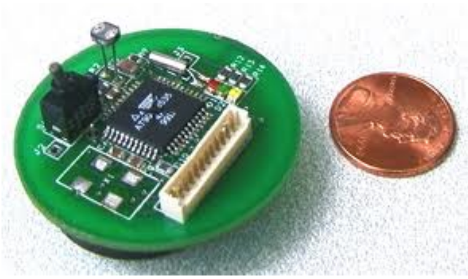


Other Virtualization Benefits

- Checkpoints/snapshots
  - Save entire machine state and restart it later
  - Create multiple copies of same machine
- Live migration
  - Move entire VM from one host to another
- High Availability
  - Continuous replication of a VM on another machine
- Live cloning
  - Create a running "copy" of a live VM
- Partial VMs
  - Create lightweight machine "shims" that allocate resources only when needed
- Security
  - Detect rootkits even if they have compromised kernel

Future uses of virtualization?

- Diversify from x86 (ARM)
- Phones
  - VMWare MVP
- Embedded Systems
  - ATMs, cars, planes, sensors
- Main driver: security and isolation



Virtualization Vendors

- Server/Cloud Virtualization
  - VMWare ESX
  - Citrix XenServer
  - Redhat KVM/QEMU
  - Microsoft HyperV
  - IBM z/VM
- Desktop Virtualization
  - VMWare Desktop
  - Parallels for Mac
  - Oracle Virtualbox
  - VMWare Mobile Virtual Platform

Outline

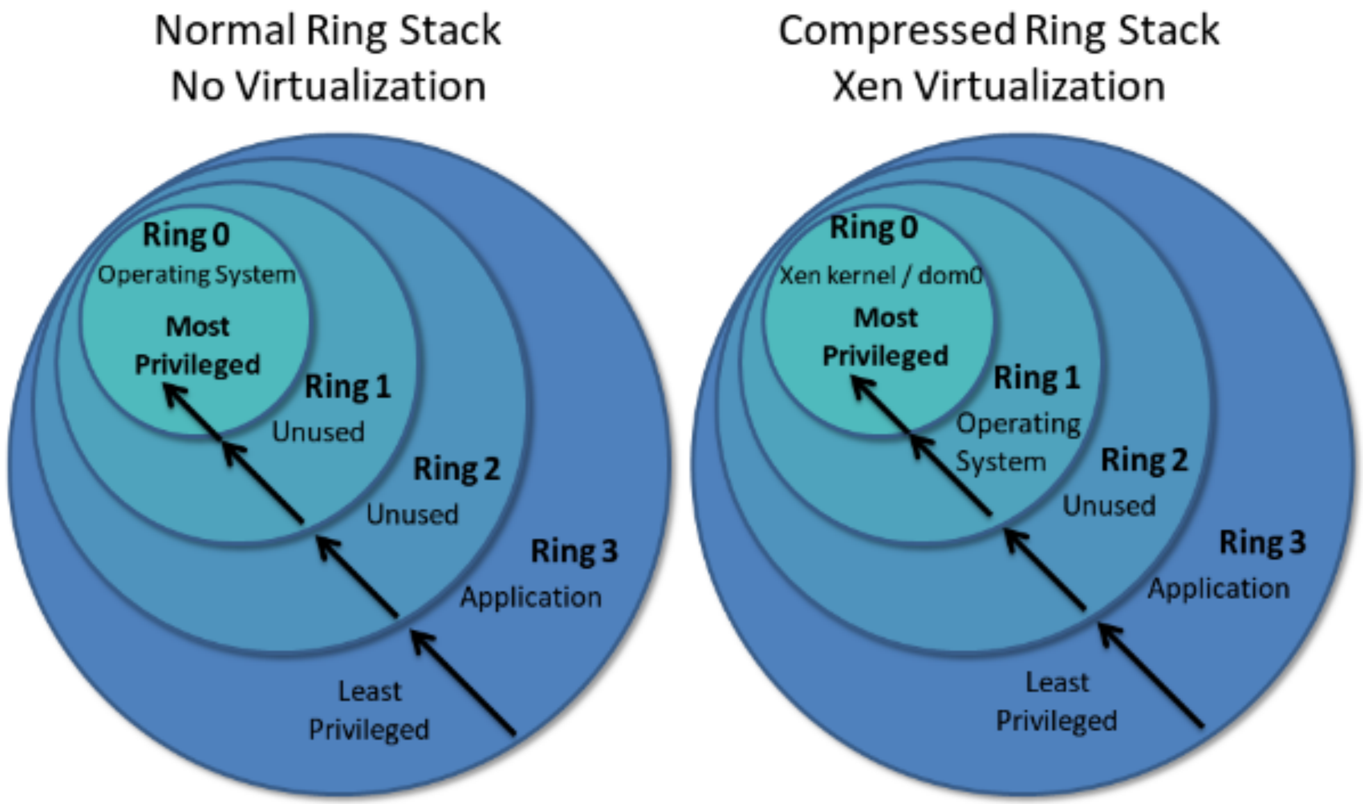
- History and uses
- How virtualization works
- Architectures
- Issues
  - Efficient sharing
  - Isolation
  - Performance

Types of Virtualization

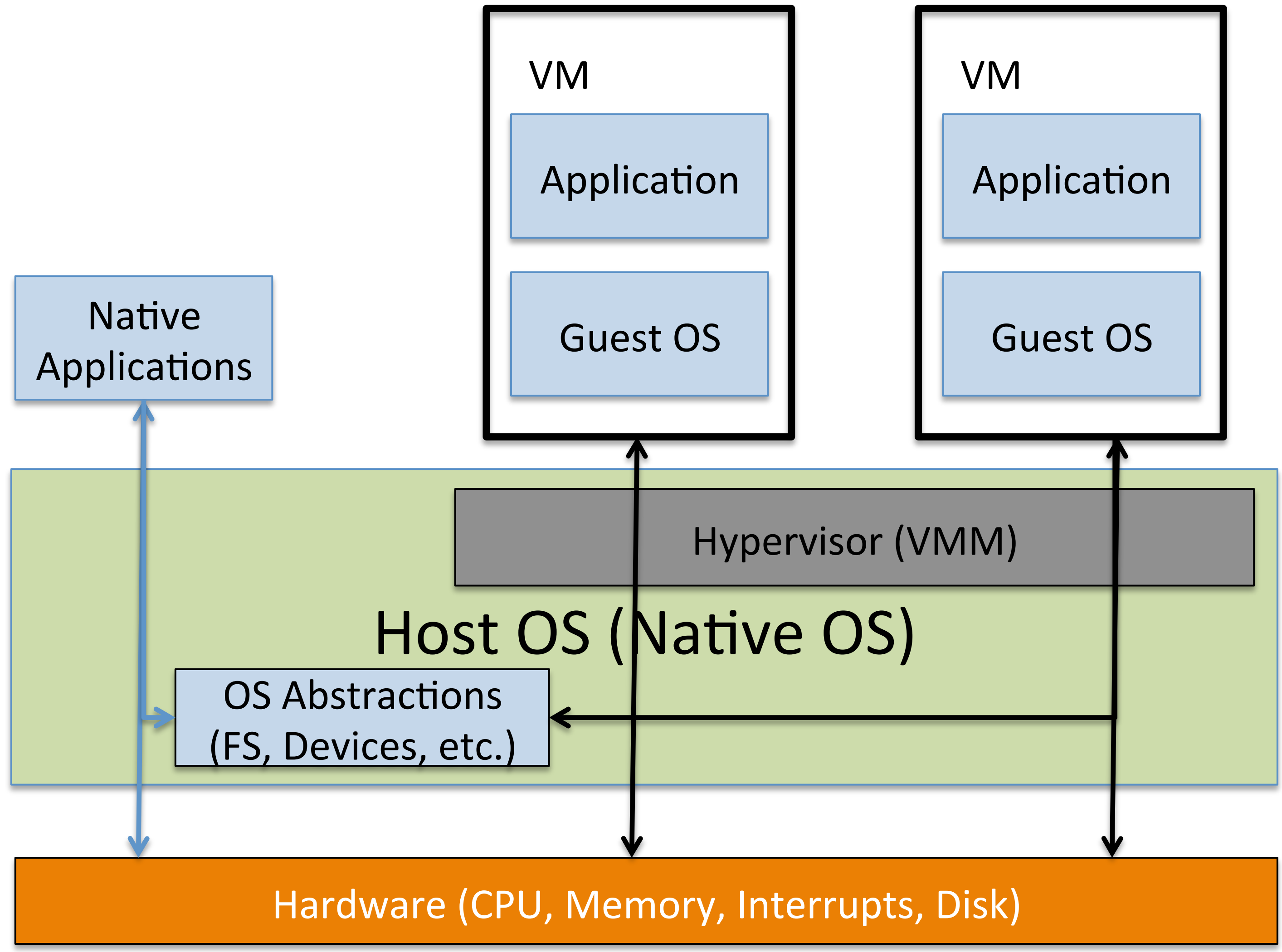
- Type 0
  - Hardware provides virtual copies (firmware VMM)
  - No OS involvement, e.g., IBM mainframe LPARs
- Type 1
  - Hypervisor more privileged than host/dom0
- Type 2
  - Host more (or equally) privileged than hypervisor
  - VM is a process in host OS
- Emulation
  - Execute hardware instruction in software (Android emulator)
  - Very flexible but very slow
- Programming Environment Virtualization
  - Compiler compiles code for special purpose environment (e.g., JVM)
- OS Containers
  - Lightweight isolation provided by OS: Linux namespaces, Solaris zones

How does virtualization work?

- Processor privilege levels



Basic Virtualization Architecture

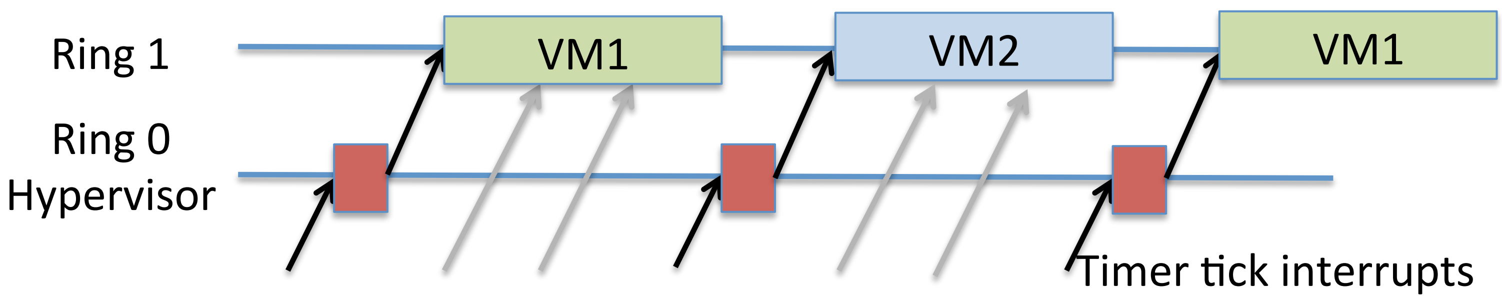


OS vs. VMM

OS	VMM
Thread/Process	VCPU
Address space	RAM (virtual)
Signals	Interrupts (virtual)
Filesystem	Virtual disk
Network sockets	Virtual network interface card
Character/Block IO device	Emulated hardware device
Synchronization	Within VM IPI (inter-processor interrupts), between VMs none

Virtualizing Interrupts

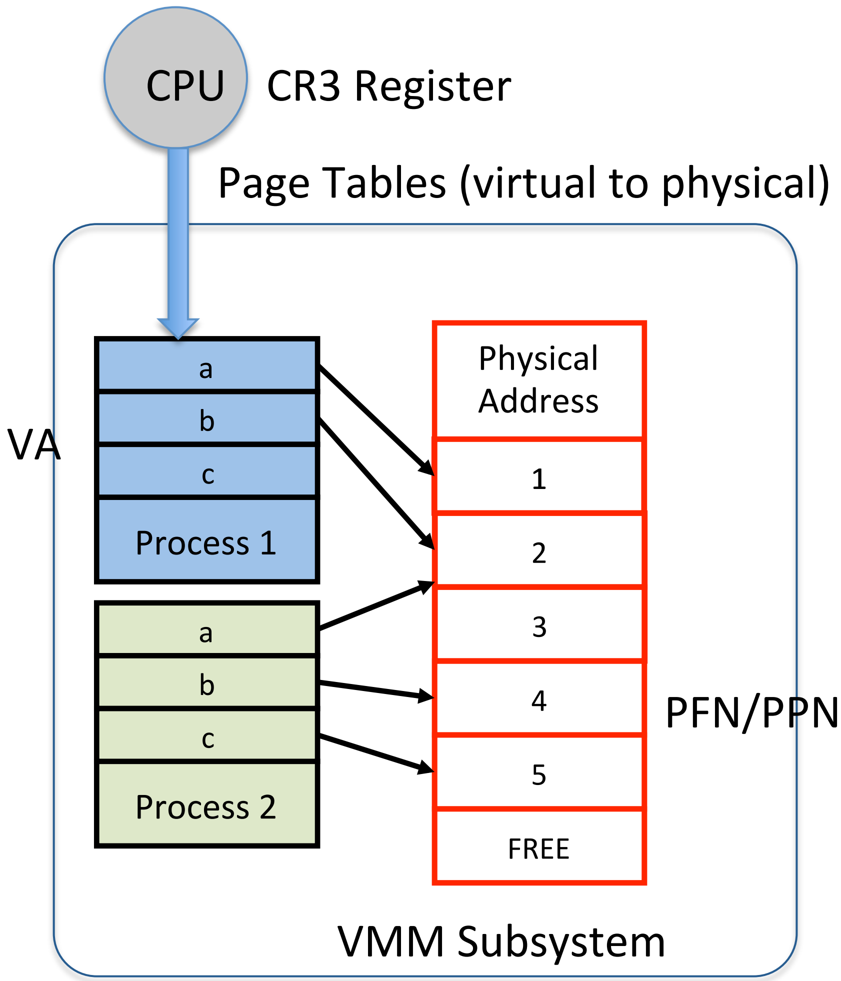
- Interrupts are the easiest
  - Delivered by processor to ring 0
  - Hypervisor sits in ring 0
  - Hypervisor can route to guest OS (call appropriate handler in guest's interrupt descriptor table)
- CPU scheduling between VMs follows
  - Timer interrupts, trapping hlt (idle) instructions

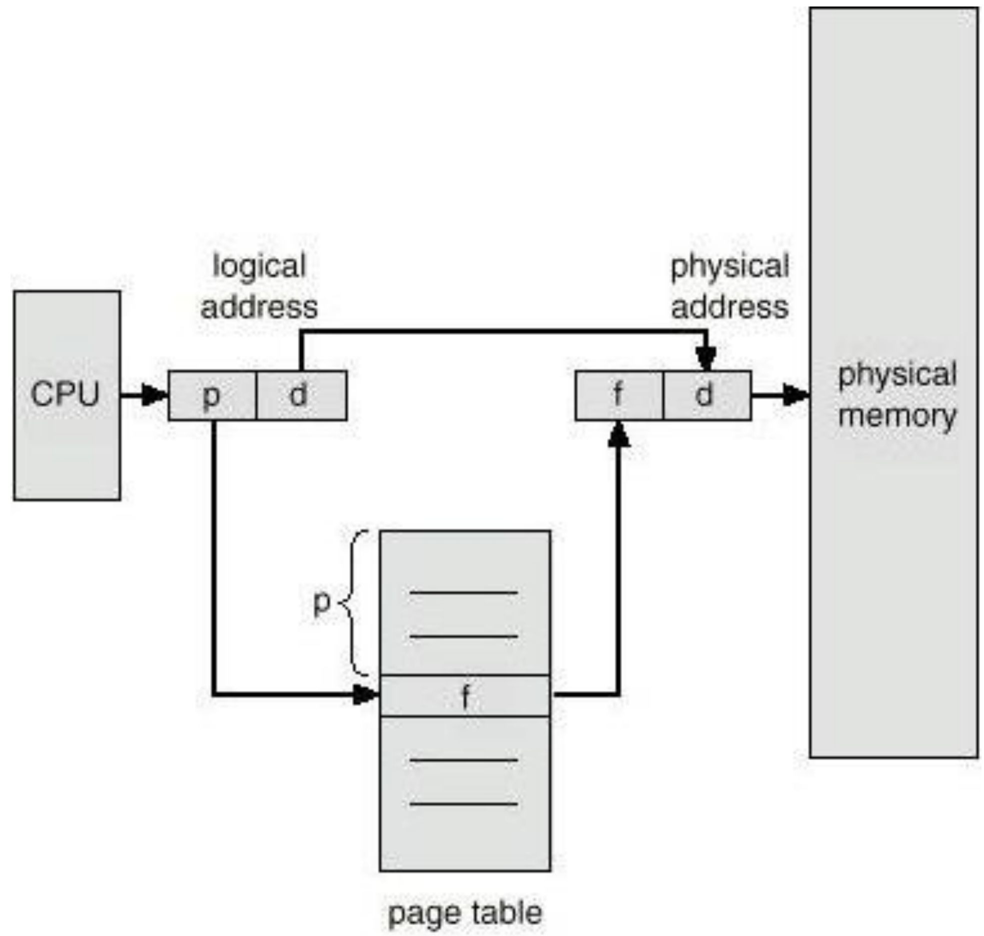


- Memory and I/O, not so easy…

OS Memory Isolation: Page Tables

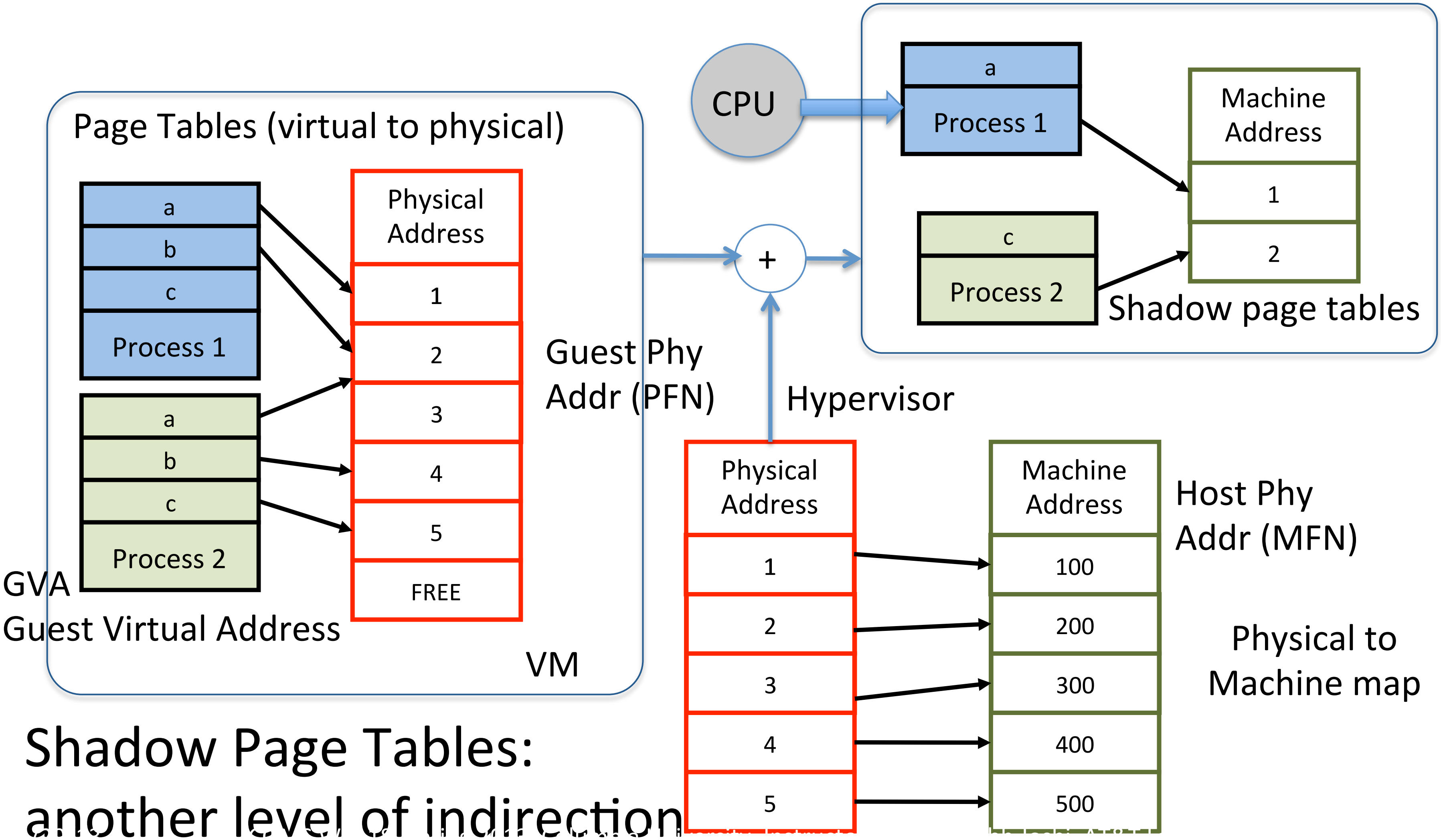
Virtual Memory





VM Memory Isolation: Shadow Page Tables

Gain control through page faults



Shadow Page Tables: another level of indirection

Anatomy of a Shadow Page Fault

- Installing a new page
  1. Guest application accesses non-existent PTE
  2. Host page fault – host looks at guest page table
  3. If guest page table has PTE
     a. Compute MFN corresponding to PTE
     b. Install real PTE in shadow page table
     c. This is a shadow page fault (guest OS doesn't see)
  4. If guest page table has no PTE
     a. Forward page fault to guest
     b. Guest OS will install new PTE in guest page table

Modifications to Guest Page Tables

- What happens when guest OS installs new PTE or changes existing PTE?
  1. Host makes Guest OS page table mem read only
  2. When guest OS changes PTE, host page faults
  3. VMM inspects the fault
  4. If guest PTE is changing
     a. Compute corresponding MFN and the real PTE
     b. Install real PTE (GVA->MFN mapping) in shadow page table
  5. Emulate the instruction
     a. Install GVA->PFN mapping in guest page table

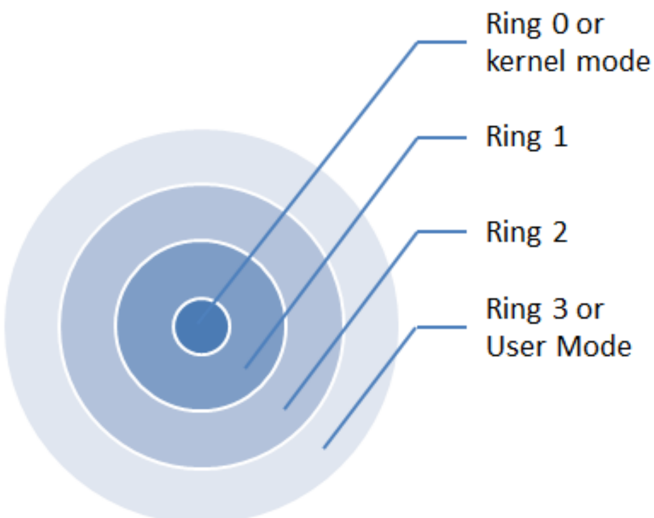
Virtualizing I/O

- When VM executes I/O instruction, VM traps because of insufficient privilege
- VMM examine I/O instruction and emulates real device
  - Maintain state of virtual device
  - Predict how the I/O instruction would impact real device, and what the return value would be
  - Change the VM's registers appropriately
- E.g., VM reads from keyboard I/O port
  - If VM is in active window, then VMM reads real keyboard, returns value in emulated keyboard register
  - If VM is running in a background window (no active control of keyboard), then VMM returns code for no key press

Virtualizing Storage

- Support dozens of guests per VMM
  - Standard disk partitioning not sufficient
  - Store as files in file system provided by host OS
  - Duplicate file -> create new guest
  - Move file to another system -> move guest
- Anatomy of VM file access
  - Guest FS converts to access to virtual disk block
  - Guest block I/O driver issues virtual disk read
  - Host FS translates to access to physical disk block
  - Host block I/O driver issues physical disk read
  - Virtual block addr and physical block addr unrelated

Principles of Virtualization

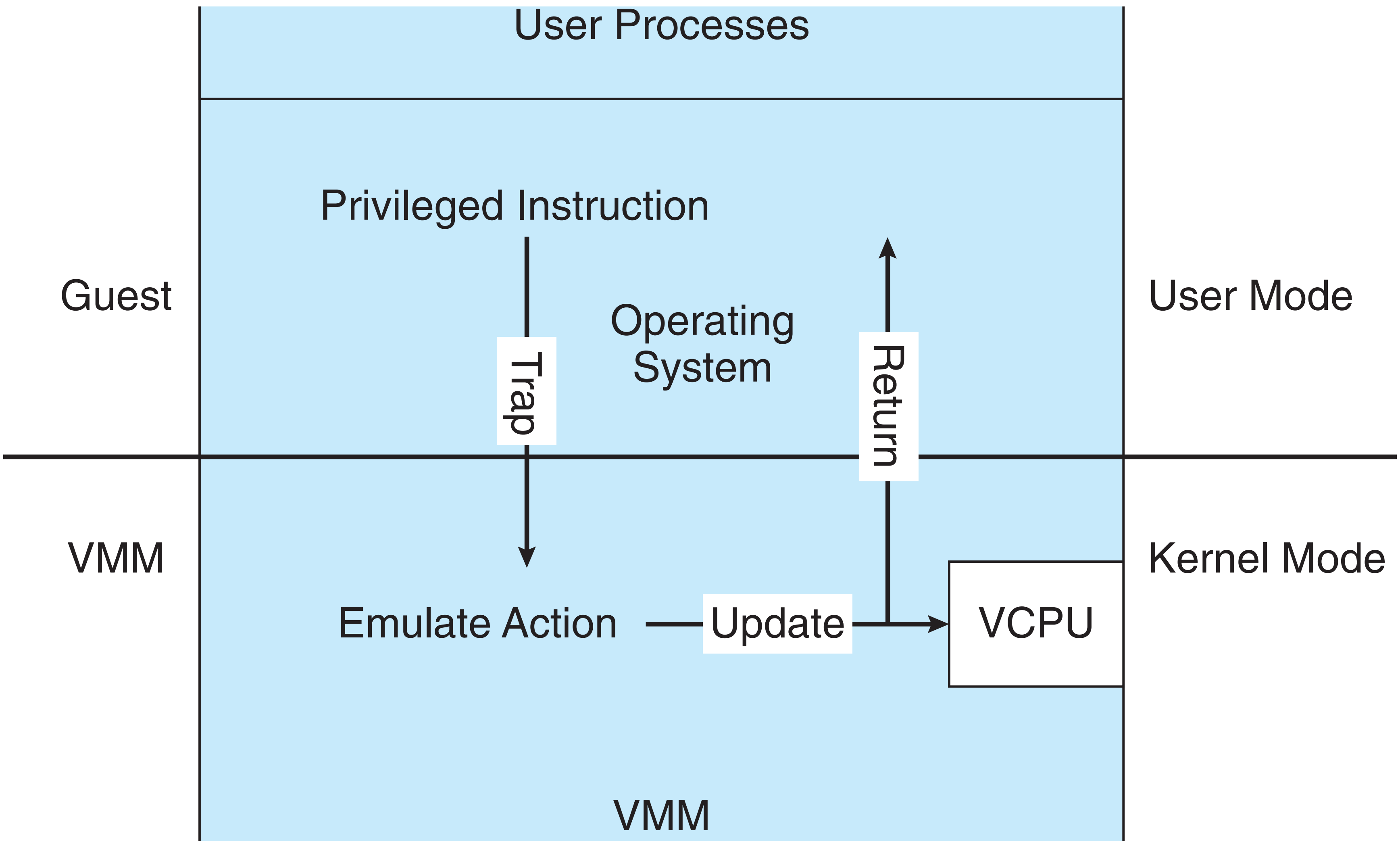


Ring 0: load/store Virtual memory protection tables, I/O instructions, privilege level get/set

Ring 1: arithmetic, normal memory load or store

- Requirements for a virtualizable architecture
  - 1974 paper, Popek and Goldberg
  - Privileged instructions (trap in user mode)
  - Sensitive instructions (behavior dependent on user/system mode or change resources)
  - Virtualizable if sensitive subset of privileged

Trap and Emulate



X86 Virtualization

- x86 is not virtualizable by Popek-Goldberg definition
- 17 sensitive but not-privileged instructions
  - Access VMM registers: SGDT, SLDT, SIDT
  - Get protection levels: PUSHF, POPF
  - Access sensitive memory (page tables): MOV, CALL
- Hardware extensions to support virtualization
  - Introduced 2006-2009
  - Intel-VTx
  - AMD-V (SVM)
  - Force traps on all privileged instructions
  - Add hardware support for MMU virtualization (EPT)

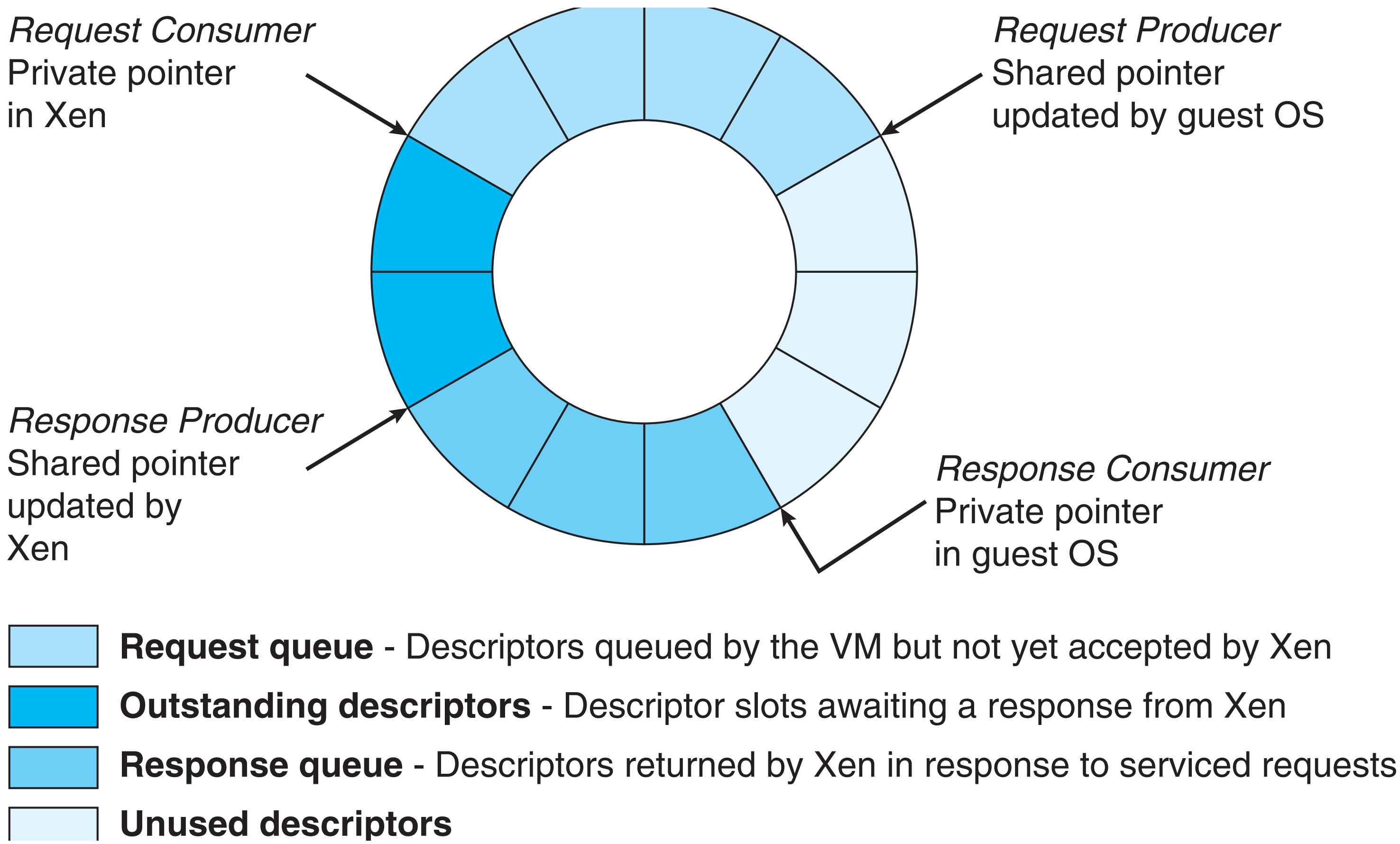
Binary Translation

- Original x86 virtualization technique
  - Popularized by VMWare (1999)
- Protect sensitive memory by unmapping
- Dynamic instruction rewriting
  - Scan machine code in memory before execution
  - Replace sensitive instructions with traps/emulated instructions
- Drawbacks
  - Complex
  - Slow
  - Caching of translated blocks to speed up execution

Paravirtualization

- Pioneered by Xen
  - 2003 ACM SOSP paper: "Xen and the art of virtualization"
- Observation: only guest OS contains sensitive instructions
  - Hypercalls: system calls between guest OS and VMM
- Modify guest OS to avoid sensitive instructions
  - Drivers use direct communication instead of register by register device emulation
  - Paging updates, hardware interrupts replaced with hypercalls
- Fastest performance

Paravirtualization Example: Xen I/O



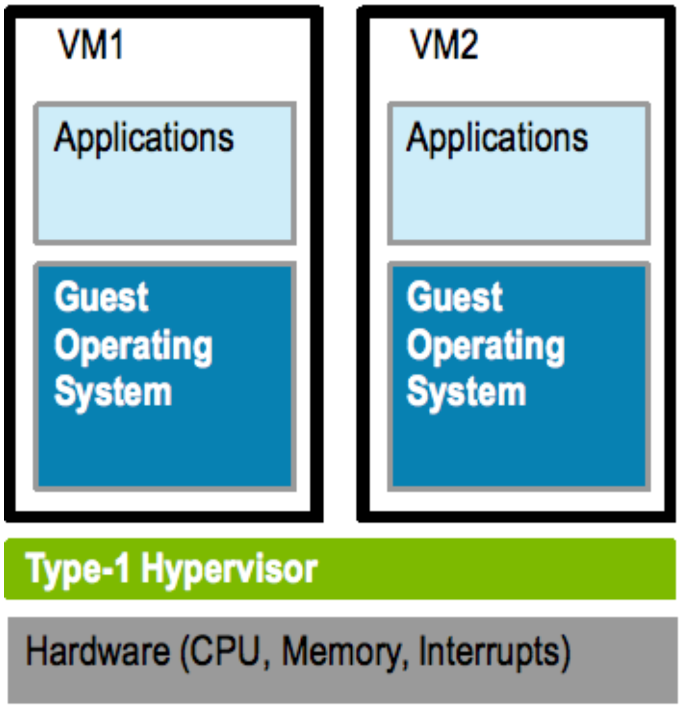
- Xen device drivers don't emulate native hardware. Directly communicate with host OS device drivers

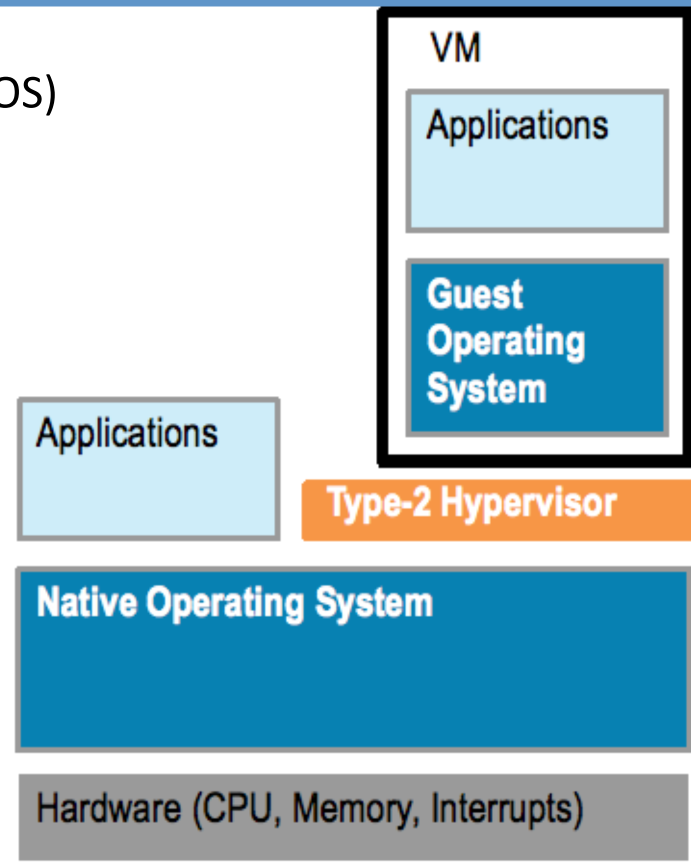
Outline

- History and uses
- How virtualization works
- Architectures
- Issues
  - Efficient Sharing
  - Isolation
  - Performance

Type 1 vs. Type 2 Hypervisors

Potentially more secure (no need to trust host OS)

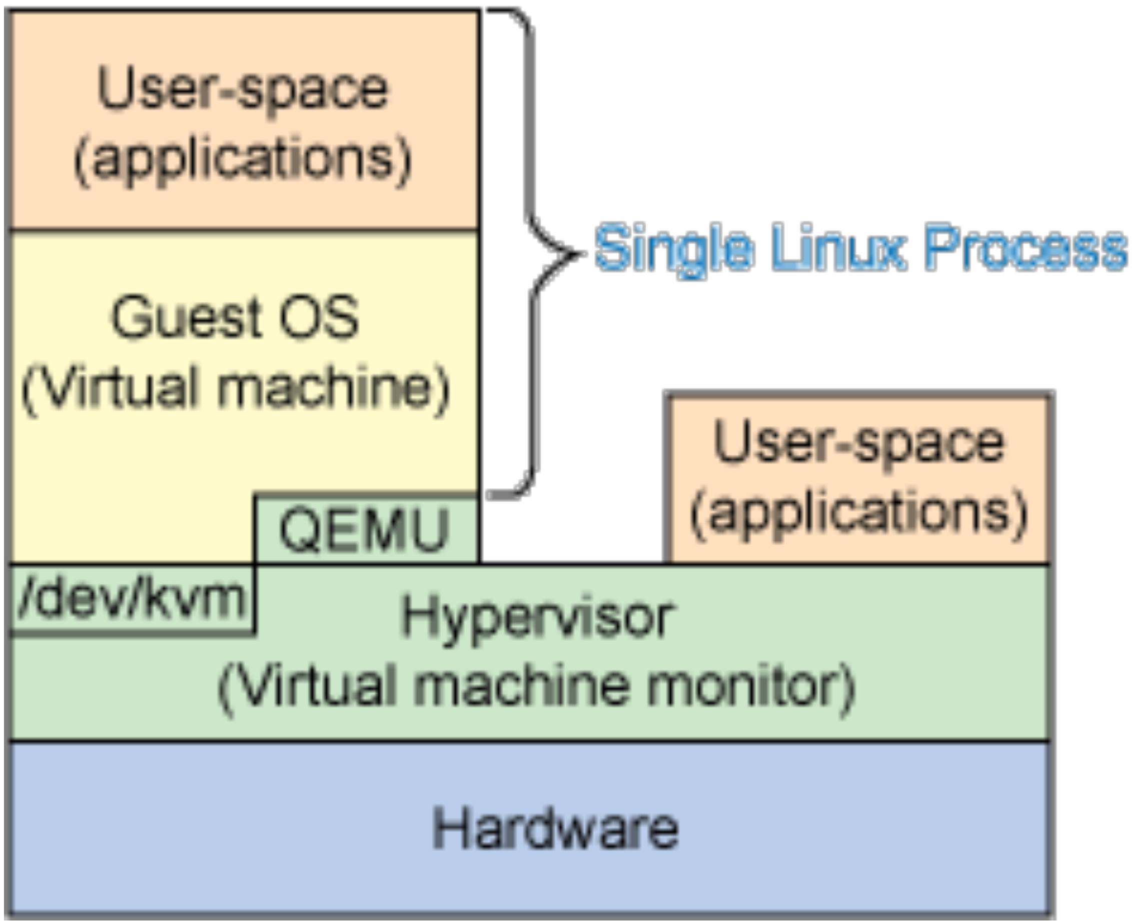


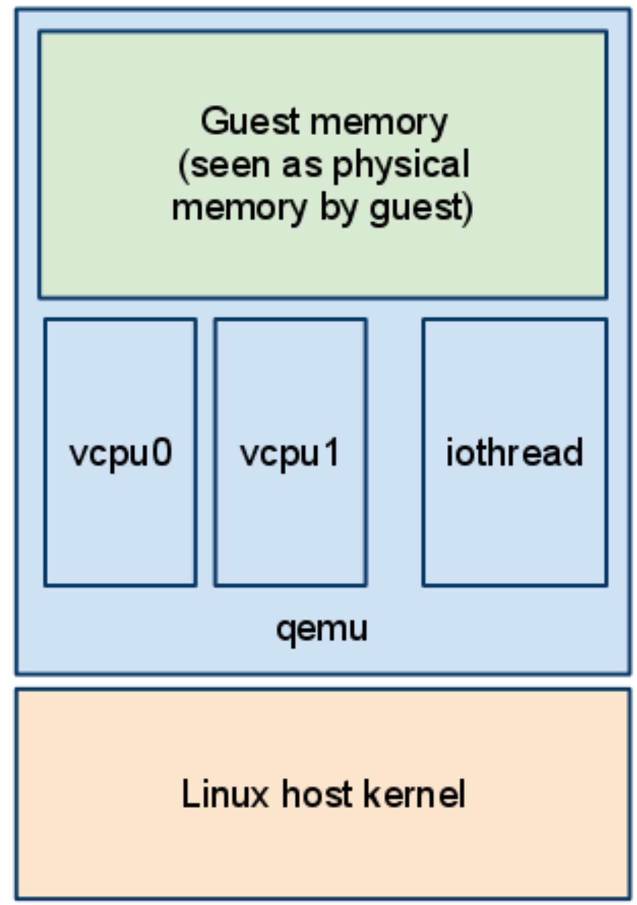


- Citrix XenServer
- VMWare ESX
- OK Labs OKL4
- Microsoft Hyper-V
- IBM z/VM

KVM (arguable)
VMWare Desktop (GSX)
Oracle VirtualBox
VMWare MVP

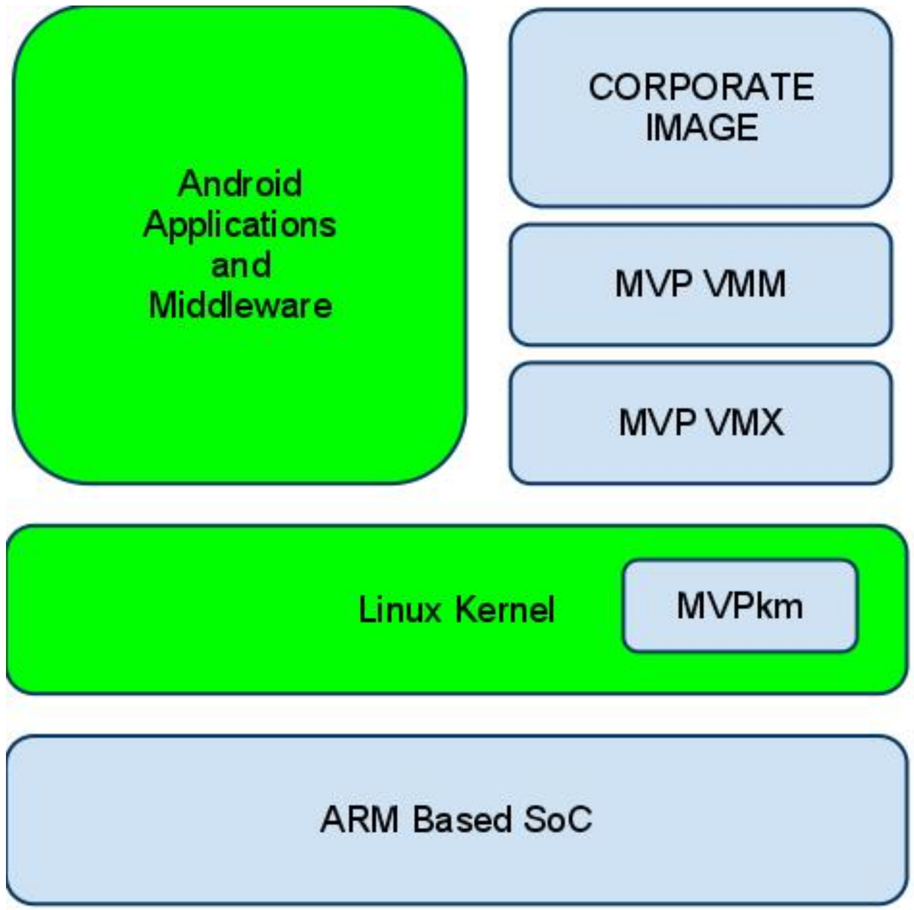
Example: KVM Architecture





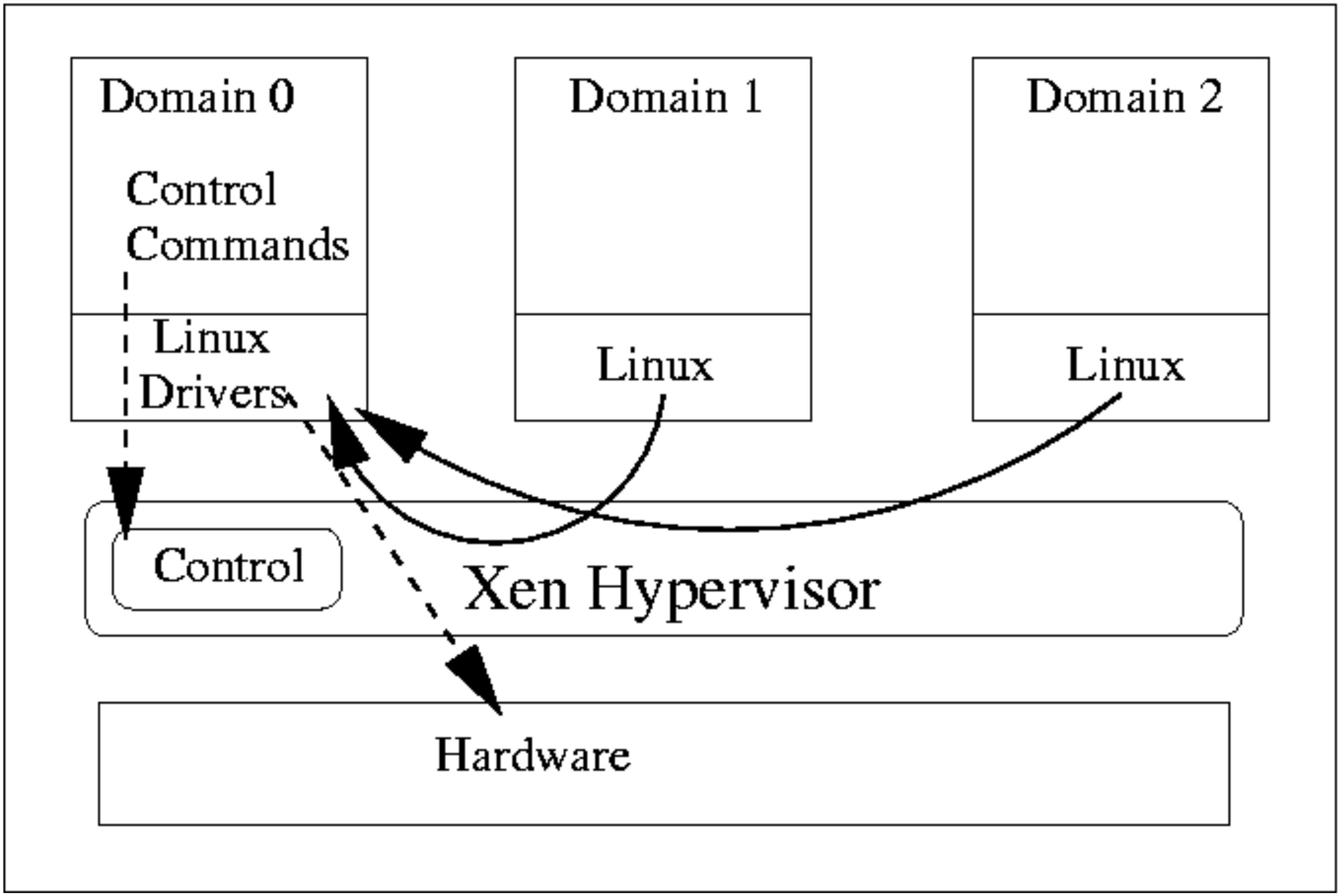
(Source: http://www.ibm.com/developerworks/linux/library/l-linux-kvm/)

Example: VMWare MVP



- Type 2 hypervisor for phones and tablets

Example: Xen Type 1 Architecture



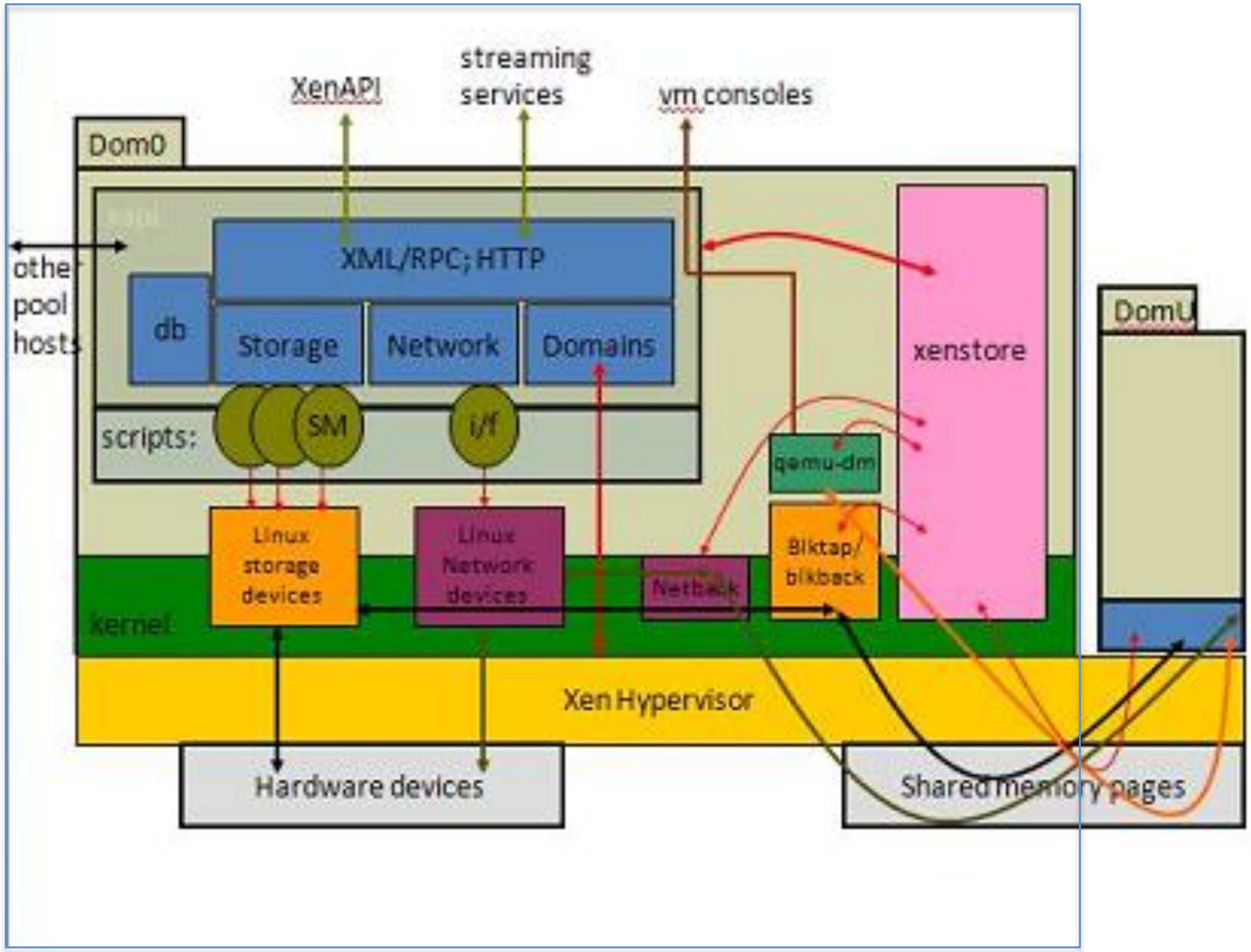
Example: Xen Trusted Computing Base

Trusted Computing Base: code that a VM must trust to be secure

Hypervisor

Domain0

- Linux Kernel
- Linux distribution
  - Network services
  - Shell
- Control stack
- VM mgmt tools
  - Boot-loader
  - Checkpointing

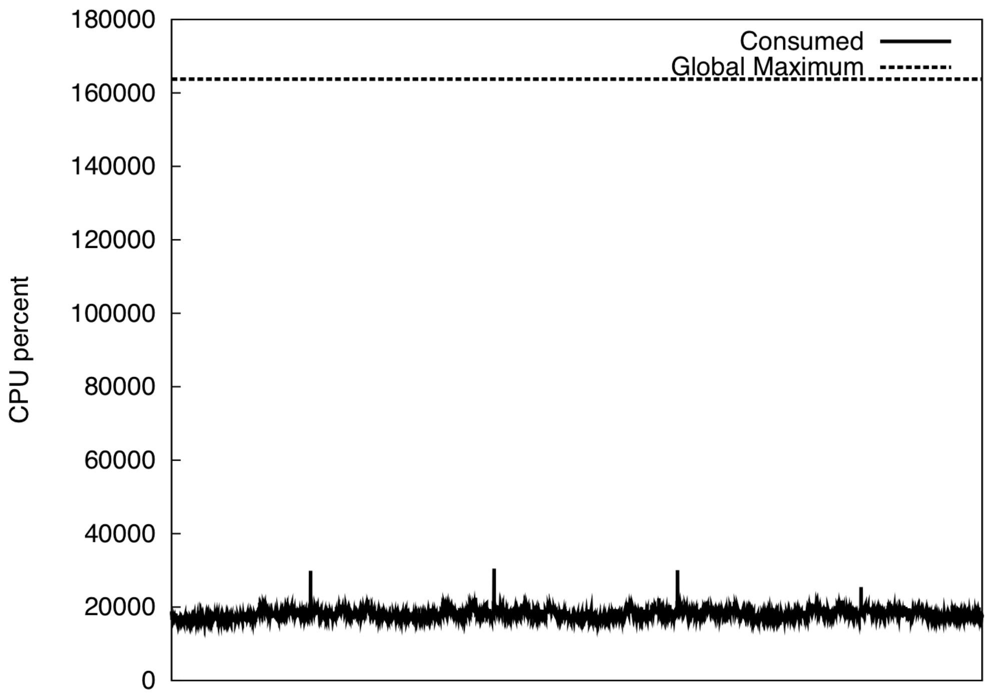


Outline

- History and uses
- How virtualization works
- Architectures
- Issues
  - Efficient Sharing
  - Isolation
  - Performance

Key VM Benefit: Resource Multiplexing

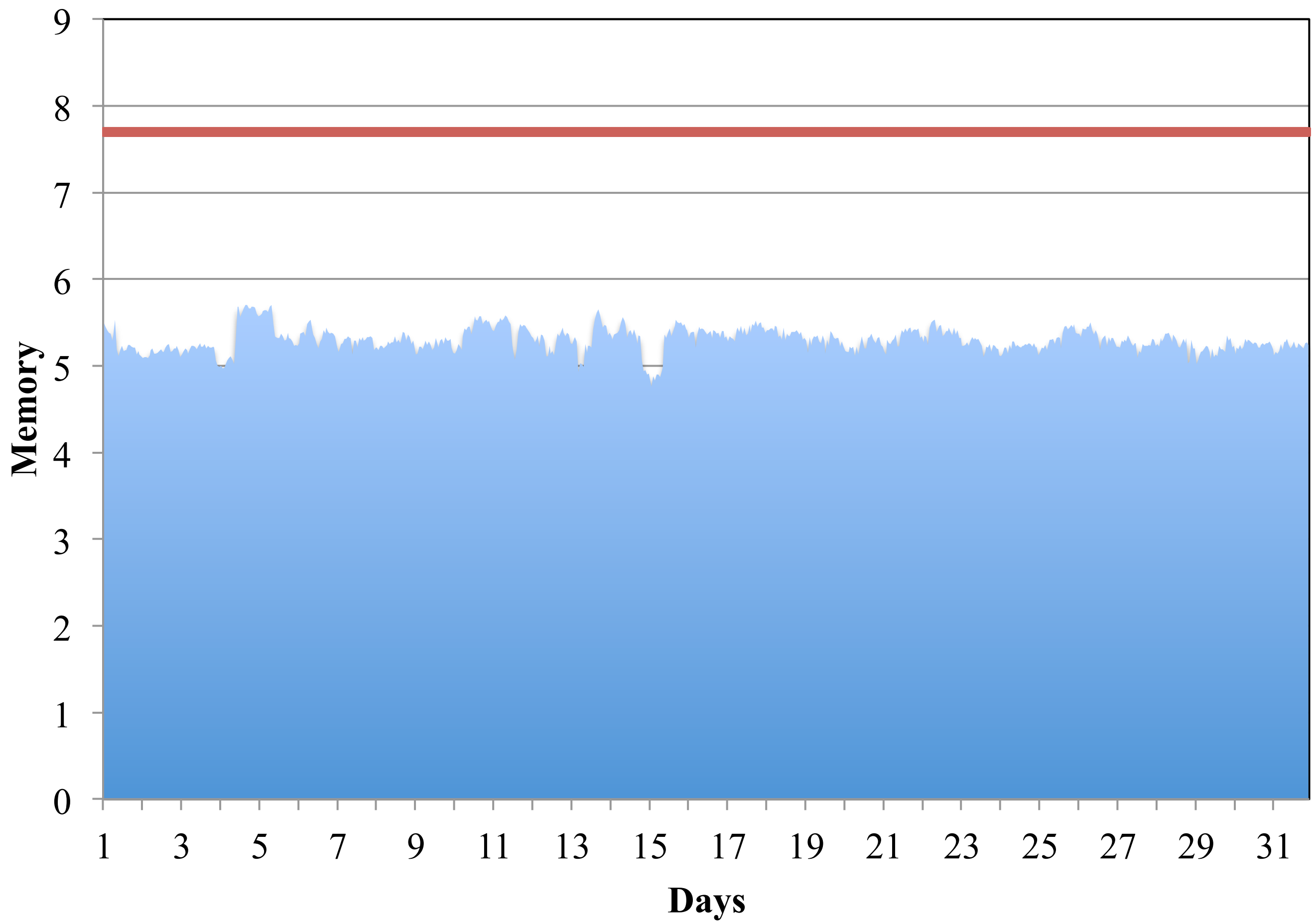
- Share hardware resources across VMs
- Overlap resource peaks in some applications with valleys in other



- CPU easy to is multiplex …

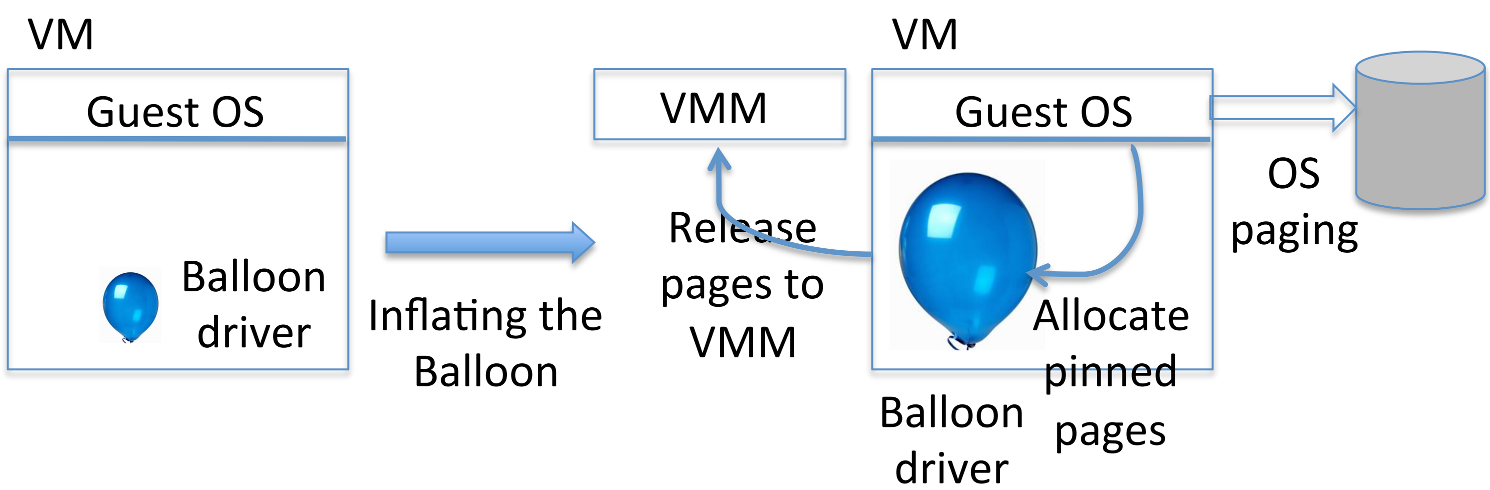
Multiplexing: The Bad News…

- Memory is not…   < 2x of peak usage



Memory Oversubscription

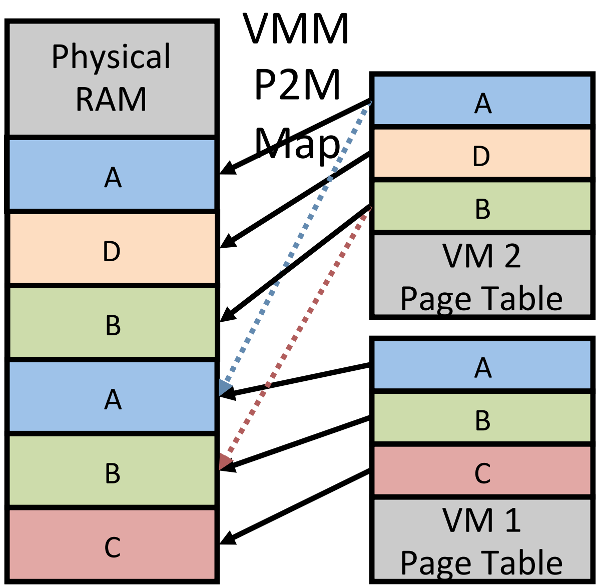
- Ballooning [Waldspurger'02]



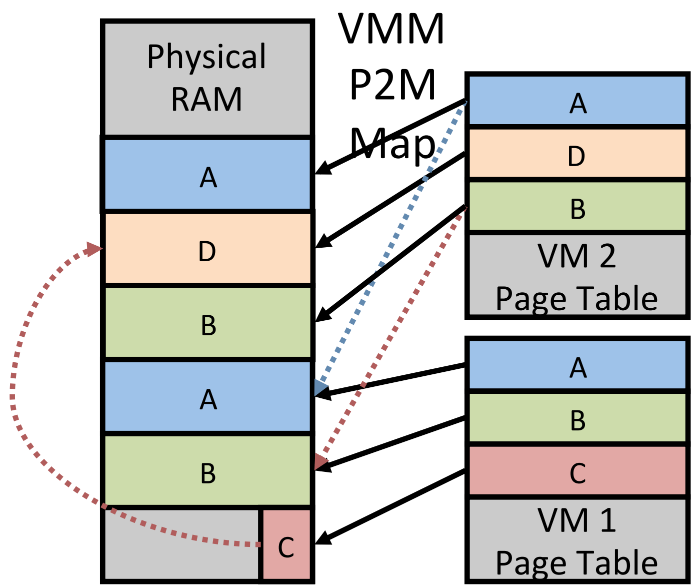
  - Respect guest OS paging/swapping policies
  - Allocates memory to free memory

Memory Consolidation (Copy-on-write)

- Trade computation for memory



Page Sharing [OSDI'02]
- VMM fingerprints pages
- Maps matching pages COW
- 33% savings

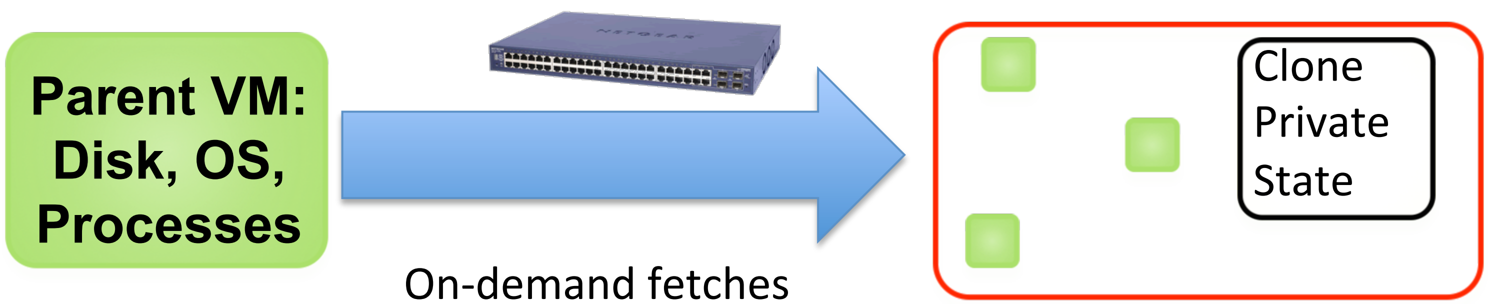


Difference Engine [OSDI'08]
- Identify similar pages
- Delta compression
- Up to 75% savings

- Memory Buddies [VEE'09]
  - Bloom filters to compare cross-machine similarity and find migration targets

Page-granular VMs

- On-demand cloning via copy-on-write (VM Fork)
  - Logical replicas
  - State copied on demand
  - Allocated on demand
- Fast VM Instantiation



VM Descriptor

Metadata, Page tables, GDT, vcpu

~1MB for 1GB VM

VM Performance

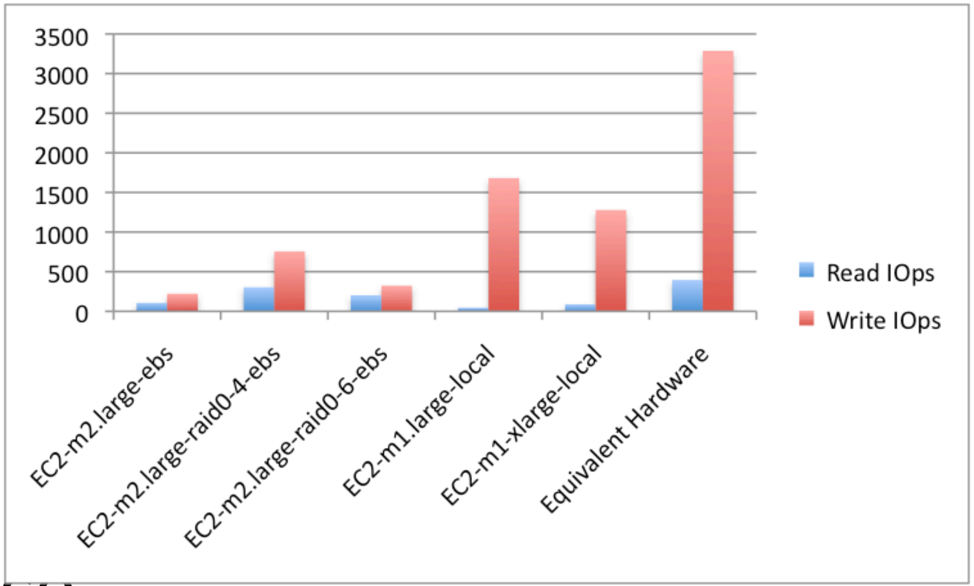
- Primary property of VMs is isolation
  - Illusion of separate machines
  - Different namespaces
- But…
  - Still share host OS/drivers
  - Still share physical hardware

Virtualization Performance Overhead

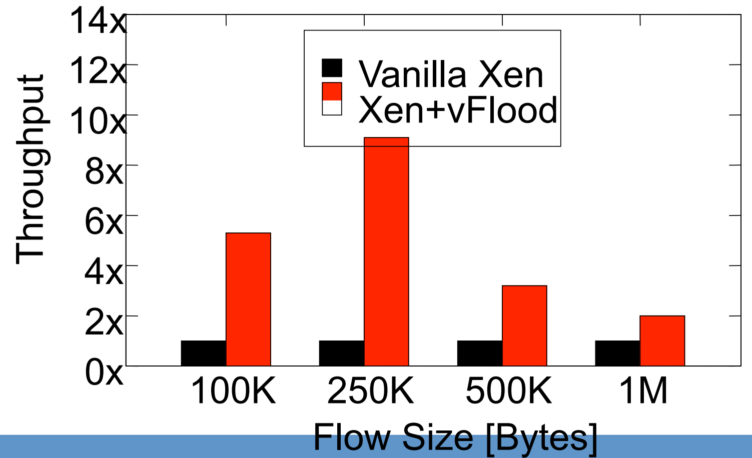
- VMM virtualization
  - Negligible (<5%). Lower with EPT
- Ring 0 (kernel) code
  - More substantial
  - Can be eliminated with paravirtualization
- I/O overhead
  - Network, disk
  - Substantial

Implications

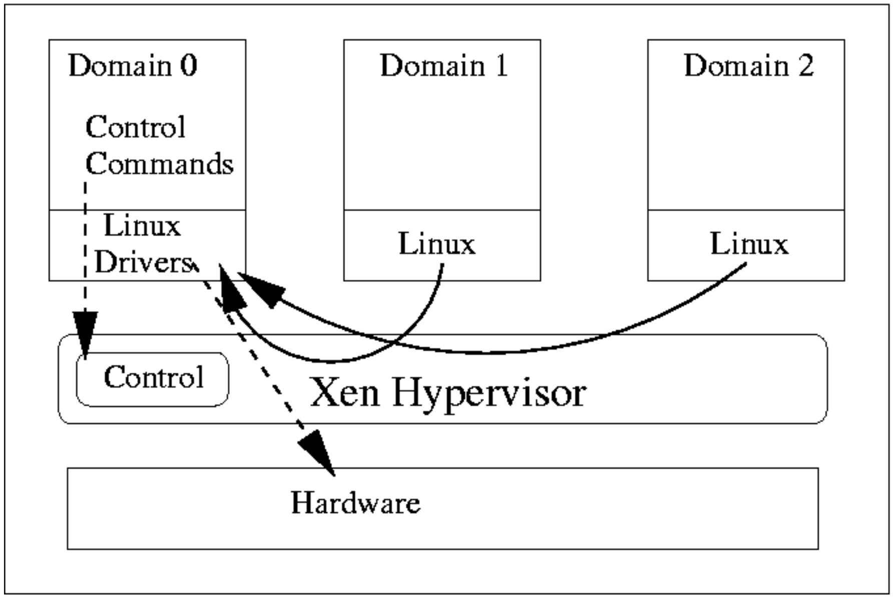
- VMs have slow I/O
- VMs can interfere
  - Cache
  - Disk
  - Network

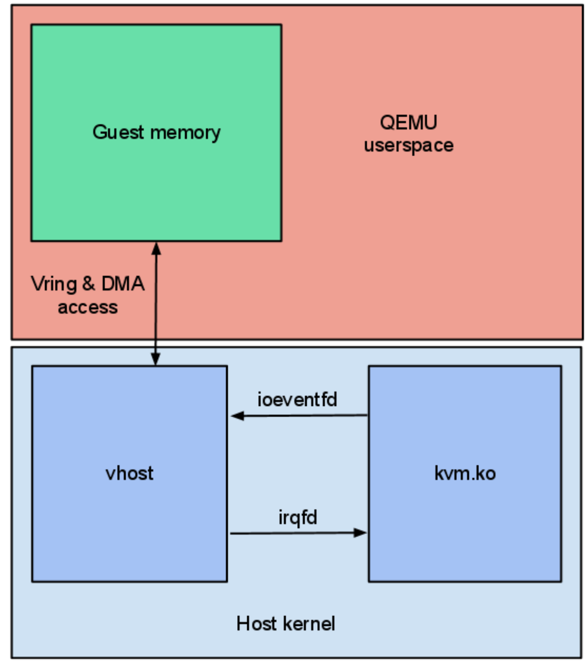


- Poor TCP performance
  - Scheduling jitter
  - Kills TCP RTT estimates



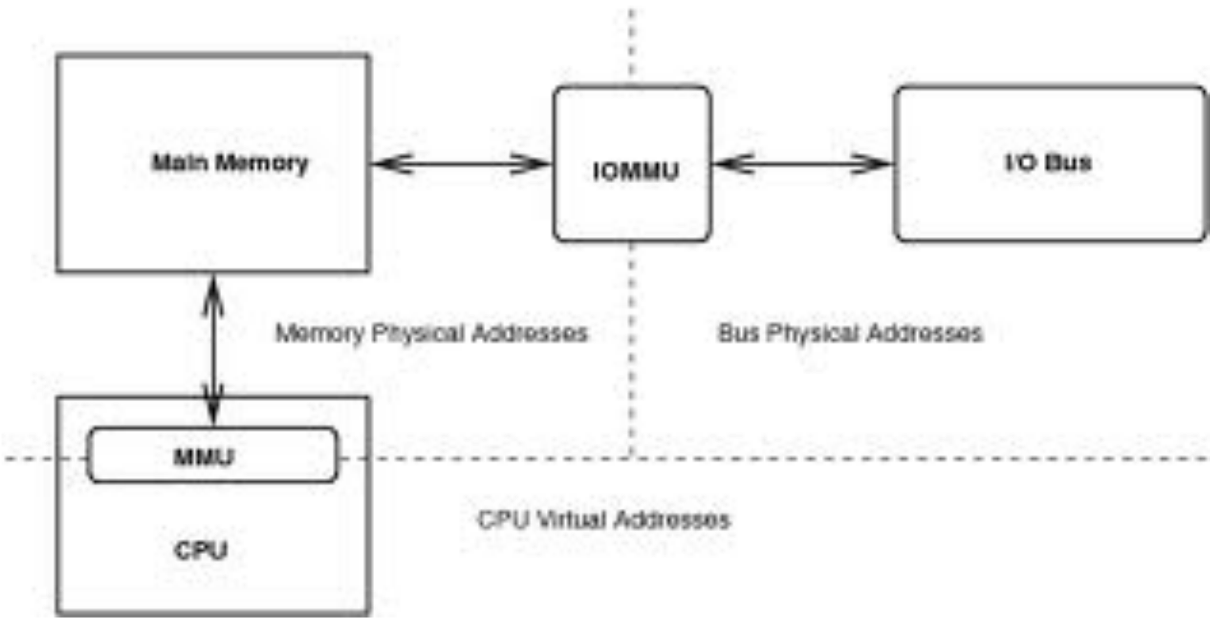
I/O Architecture





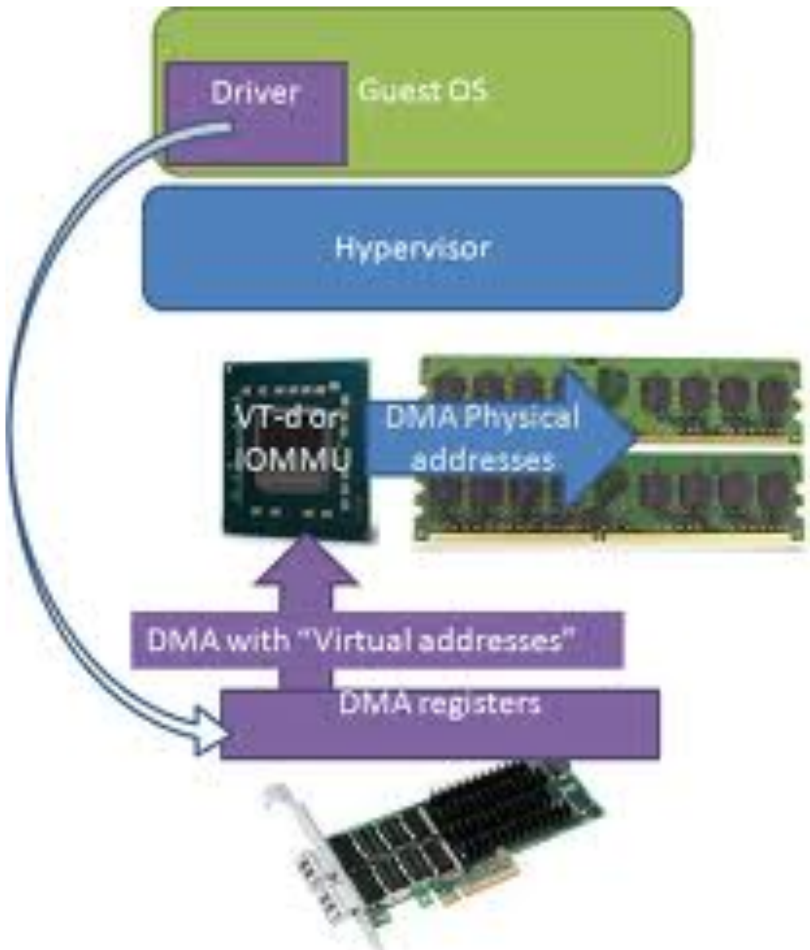
- All I/O passes through Dom-0 (or equivalent)
- Emulating device = multiple boundary crossings
- Data copy overhead
- Shared bottleneck
- Paravirtualization minimizes but not eliminates

IO-MMUs (VT-d)



- Why do we need drivers in the VMM/Dom0?
- Device memory access, DMA
- VMs can program devices to overwrite others' memory
- IOMMU provides virtual-physical mappings for devices
- Page table translation for device memory access

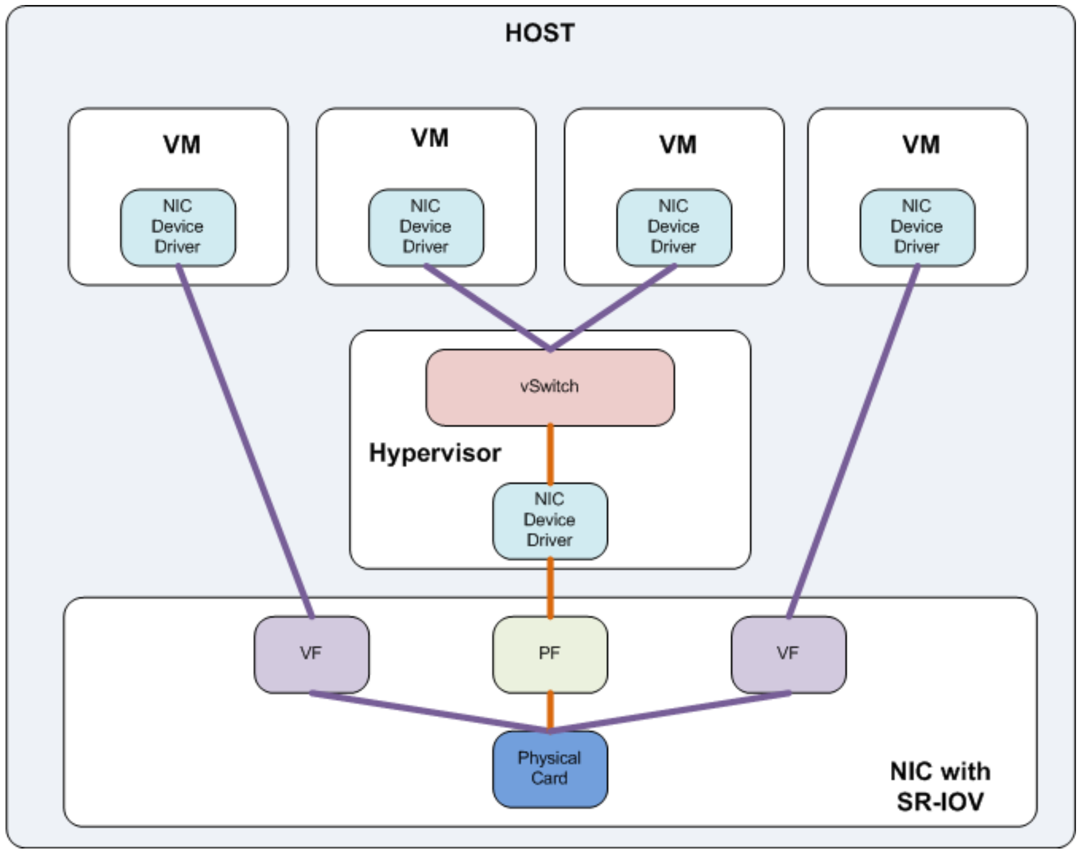
PCI Passthrough



IOMMU is a core requirement

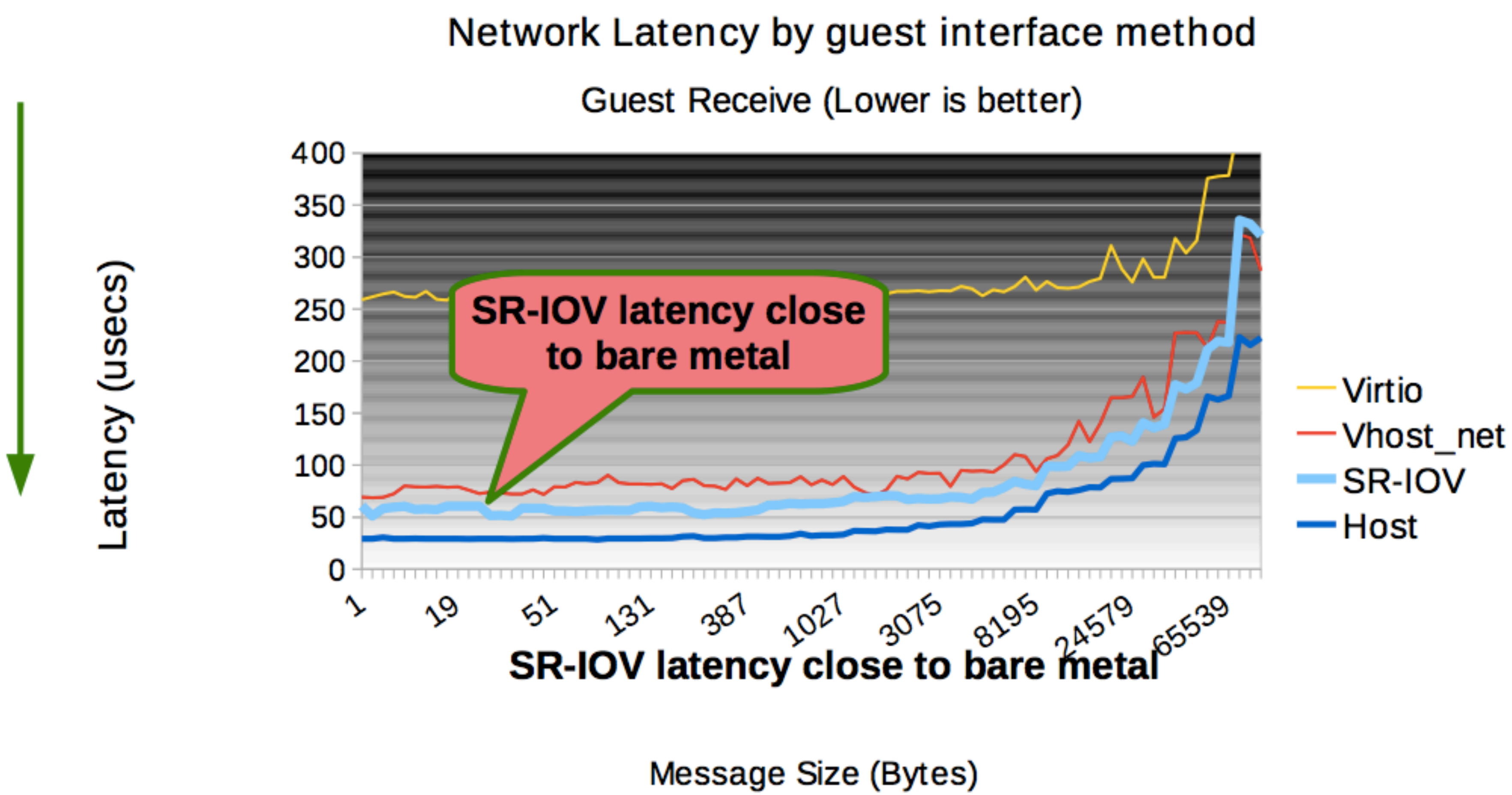
- VMs can do DMA directly
- Drivers reside in VMs
- Can assign devices to VMs
- Remove Dom-0 bottleneck
- Impact portability, migration

Single-root IO Virtualization (SR-IOV)



- Device implemented virtualization
- Hardware creates virtual copies of itself

I/O Overhead



Network Latency by guest interface method
Guest Receive (Lower is better)
Latency (usecs)
400
350
300
250
200
150
100
50
0
1
19
51
131
387
1027
3075
8195
24579
65539
Message Size (Bytes)
SR-IOV latency close to bare metal
Virtio
Vhost_net
SR-IOV
Host
SR-IOV latency close to bare metal

Finally, Orchestration Frameworks

- Manage physical infrastructure
  - Inventory of physical machines, storage, networks
  - Monitor hardware usage
- Manage collections of VMs
  - Scheduling: where to start a new VM? Enough CPU/memory?
  - Resource provisioning: How to start VM?
  - Inventory of running VMs
  - When to move VMs for balancing load?
  - Similarly, manage storage (virtual disks) and network
- Manage users
  - Who owns what VM?
  - Which VMs are allowed to see what other VMs?
  - Provide web-portals/APIs to start/stop/snapshot VMs
- An OS for VM clusters…

Orchestration Framework Examples

- VMWare vSphere
  - VMWare ESX
- OpenStack
  - Hypervisor agnostic: KVM, Xen, VMWare, HyperV
- Eucalyptus
- OpenNebula
- CloudStack