Operating Systems
Virtualization

David Hay
Dror Feitelson

Some slides from Bob Cotter (UMKC)

the Idea: Decouple SW from HW

Perhaps the most important advance in computer systems in the last 25 years

For organizations

- Server consolidation
- Reduce costs for HW, cooling, electricity, maintenance.

For users

- Run 2 OSs on one machine
- Use the best one for each task

The basis for cloud computing

From Lecture 1

Virtualization

The OS virtualizes the hardware

- Decouple from physical limitations
- Support the illusion that there are more resources available
  - Each app thinks it has the machine for itself
- Implementation:
  - By introducing a level of indirection
  - By juggling resources among applications
  - Need some hardware support

18

Technical Definition

Virtualization is:

- Decoupling from physical limitations
  - What you get does not exist in reality
- Using a level of indirection
  - How we hide the limited reality
- Providing exactly the same interfaces
  - Distinction from abstraction

Processes as Virtual Machines

A process is a virtual machine

as applications see it

- CPU – but only non-privileged instructions
- Virtual memory
- Also new abstractions supported by the OS
  - e.g. a file system
  - So actually an “abstract virtual machine”

“Real” Virtual Machines

Virtualize the complete hardware

As the operating system sees it

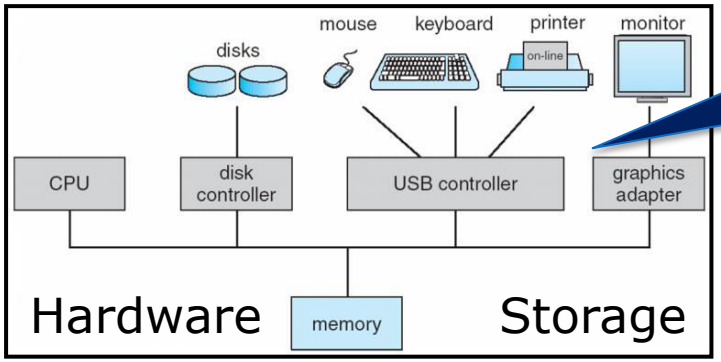
- CPU with all instructions, including privileged ones
- The physical memory and address mapping
- All the peripheral devices

You can run an OS on this virtual machine!

From Physical to Virtual Machine

Applications

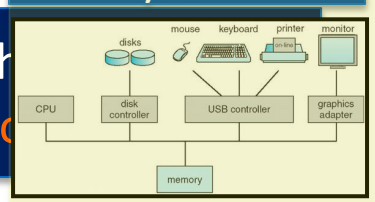
Operating System



OS controls th
and is tightly co

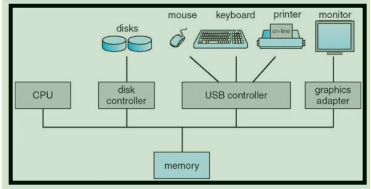
Applications

Operating System



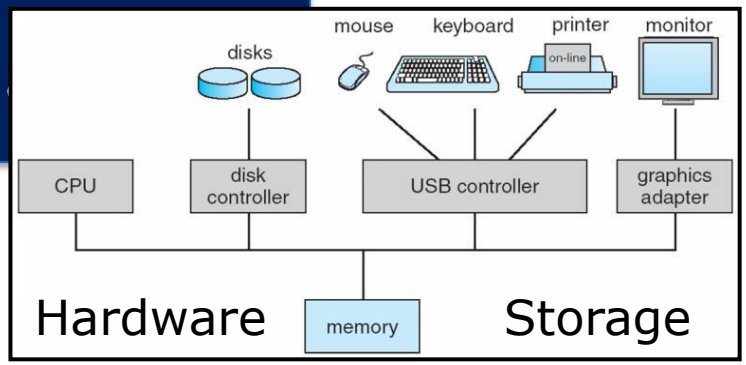
Applications

Operating System



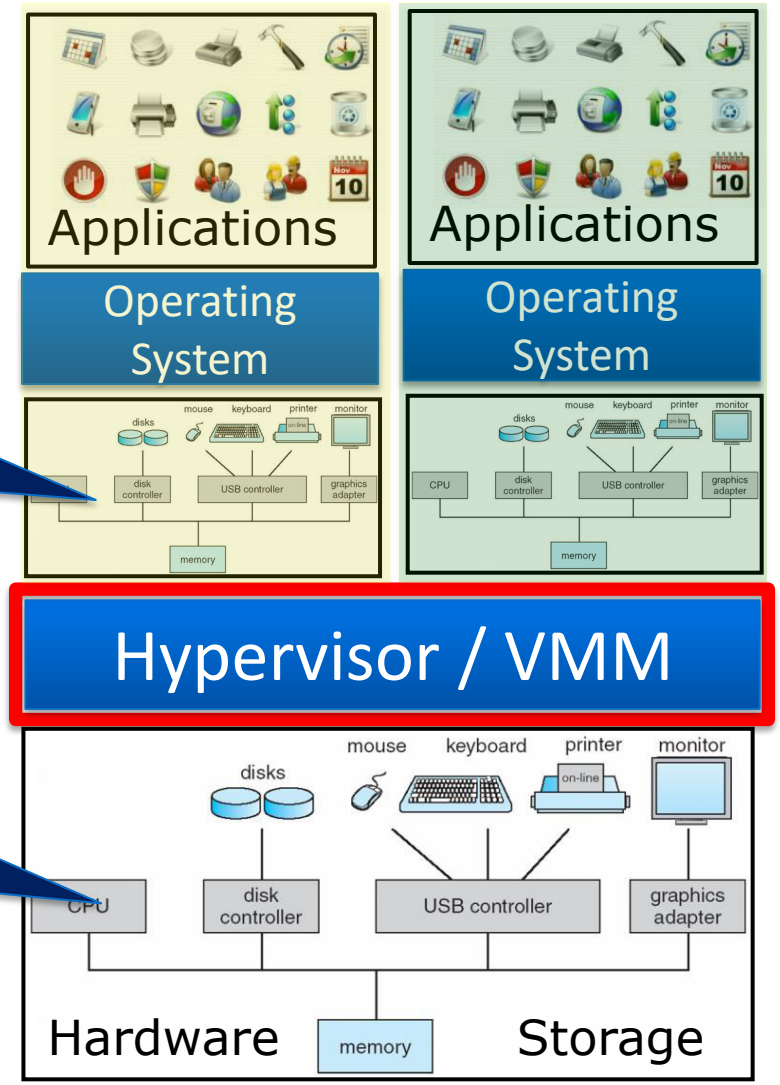
Hypervisor / VMM

Physical H



The Virtualization Layer

- "Hypervisor" or "Virtual Machine Monitor"
- Extra level of indirection, decouples hardware and OS
- Multiplex
- Strong isolation
- Managing physical resources
- Performs functions similar to an OS
- Other possibilities (later…)



Virtualization Requirements
Popek and Goldberg, 1974

- **Equivalence**: virtual machine is identical to physical machine
  - Any OS/apps combination behaves the same way
- **Safety**: virtual machines are isolated from physical machine and from each other
  - As if each is running on a distinct computer
- **Performance**: only minor decrease in execution speed relative to running on physical hardware

Virtualization with Multiplexing

Create multiple virtual devices on one physical device

- Virtual memory: support multiple address spaces on a single smaller physical memory
  - Each accessed with byte addressing
- Disk partitions: make a single disk look like multiple (smaller) disks
  - Each providing independent block storage
- Virtual machines
  - E.g. for server consolidation (more on this later)
- Virtual private Network (VPN)
  - Tunnel through the Internet in private

Virtualization with Aggregation

Create a virtual device using multiple physical devices

- RAID: array of disks with replication
  - Data is replicated internally to improve reliability
  - Interface stays conventional block storage: any file system stored on a conventional disk can be stored on RAID instead

Some History

- 1960s/70s – virtualization on mainframes
  - e.g. IBM's VM/CMS
- 1980s/90s – shift to personal machines
  - Development of unvirtualizable hardware
    - E.g. read-only access to privileged data
- 2000s – rediscovery of benefits of virtualization
  - Extensive use in cloud infrastructure
  - New hardware support

Why Bother?

- OS diversity: run Windows and Linux on the sam
- Serv ty and effi
- Secu ection syst
- Avai d ben

Cloud = Infrastructure as a Service

- Rent VMs from cloud provider, strongly isolated from the other VMs running on the same physical server
- Avoid expenses on in-house infrastructure and maintenance personnel

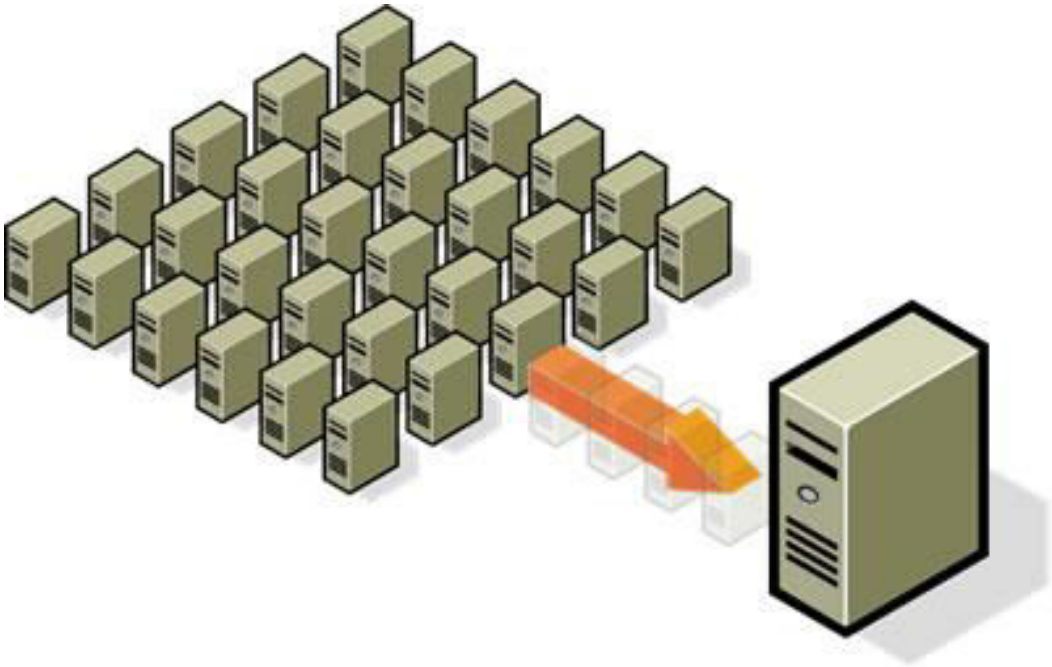
→ Foundations for cloud computing

Server Consolidation

- Servers often run a single major application or service
  - Provides strong isolation between services and instances (sometimes including performance isolation)
  - But inefficient inflexible use of hardware and energy: some are idle while others are overloaded
- Virtualization facilitates consolidation: multiple logical servers on shared physical infrastructure
  - saves on hardware & energy
  - Disaster Recovery
  - High Availability
  - Testing and Deployment

Server Consolidation/Data Centers

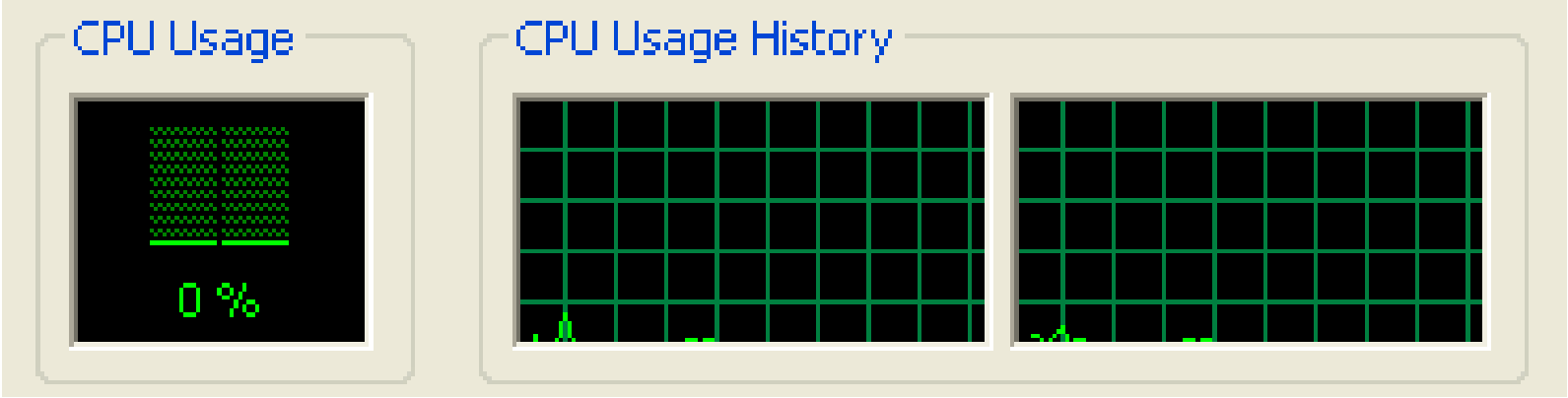
Reduce costs by consolidating services onto the fewest number of physical machines



http://www.vmware.com/img/serverconsolidation.jpg

Non-Virtualized Data Centers

- Too many servers for too little work

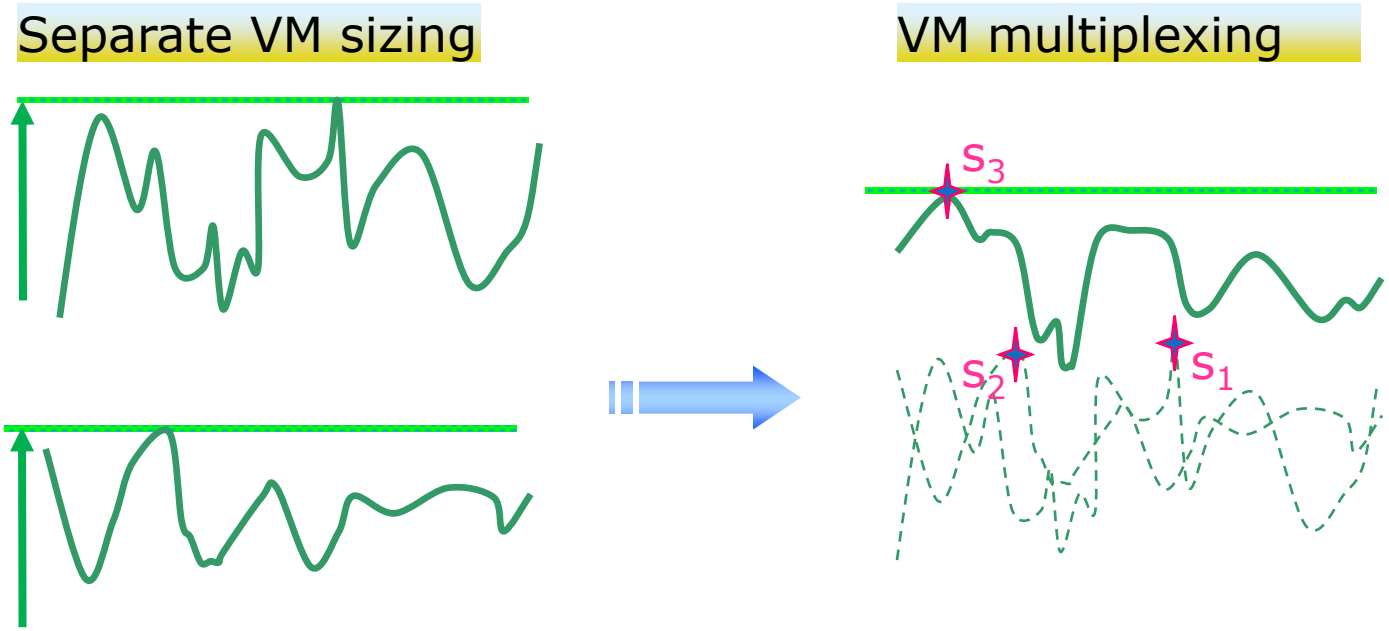


- High costs and infrastructure needs
  - Power
  - Networking
  - Floor space
  - Cooling
  - Maintenance
  - Disaster Recovery

Dynamic Data Center

- Virtualization helps us break the “one service per server” model
- Consolidate many services into a fewer number of machines when workload is low, reducing costs
- Conversely, as demand for a particular service increases, we can shift more virtual machines to run that service
- We can build a data center with fewer total resources, since resources are used as needed instead of being dedicated to single services

VM Workload Multiplexing



We expect $s_3 < s_1 + s_2$. Benefit of multiplexing !

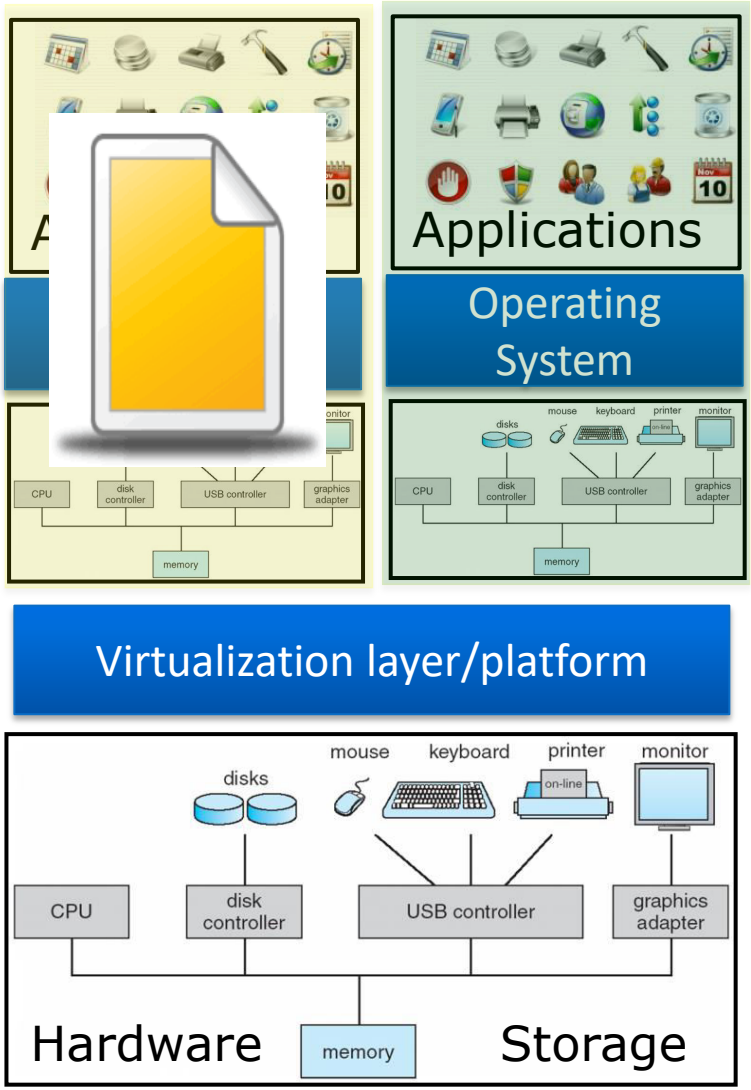
- Multiplex VMs' workload on same physical server
  - Aggregate multiple workload. Estimate total capacity need based on aggregated workload
  - Performance level of each VM is preserved

Other Uses

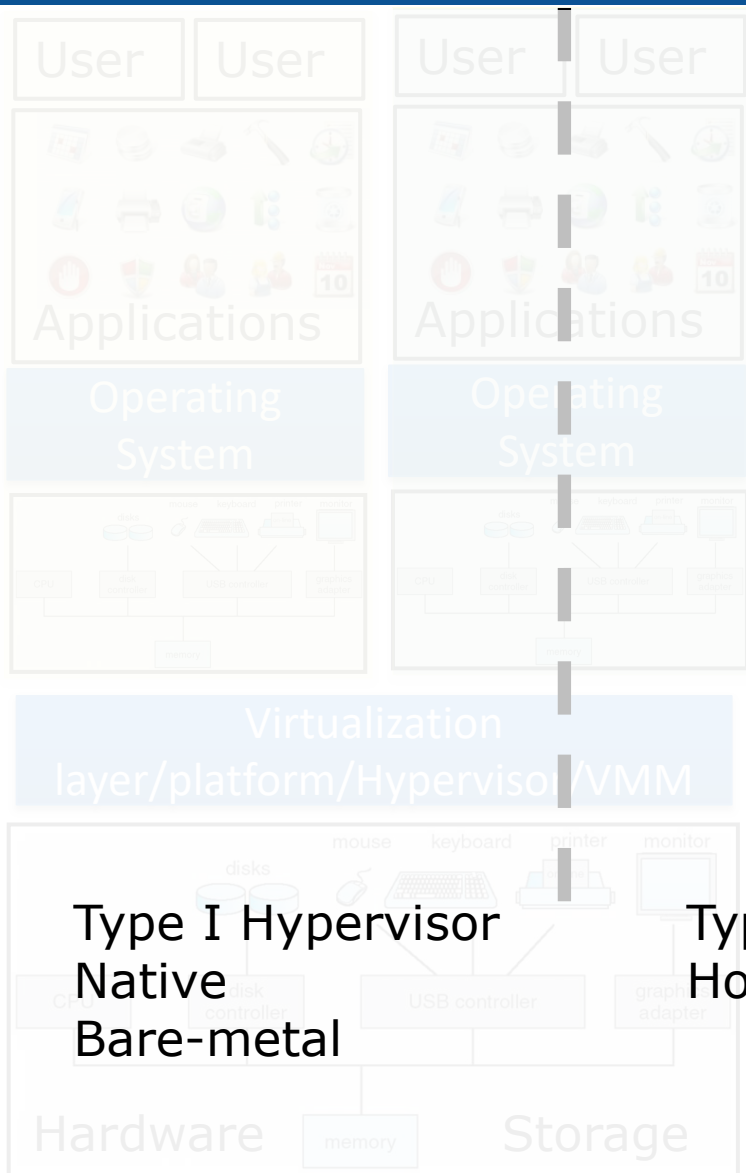
- Support for legacy applications
  - Bring up a VM emulating the previous-generation system
- Software Development
  - If software crashes, might crash the VM
  - But will not crash the whole system
  - Special case: OS development

VM Encapsulation

- Entire VM in a file
  - OS, Application, Data
  - Memory and device state
- Snapshots and Clones
  - Capture VM state on the fly and restore to point-in-time
  - Rapid System provisioning, backup, migration for better load balancing

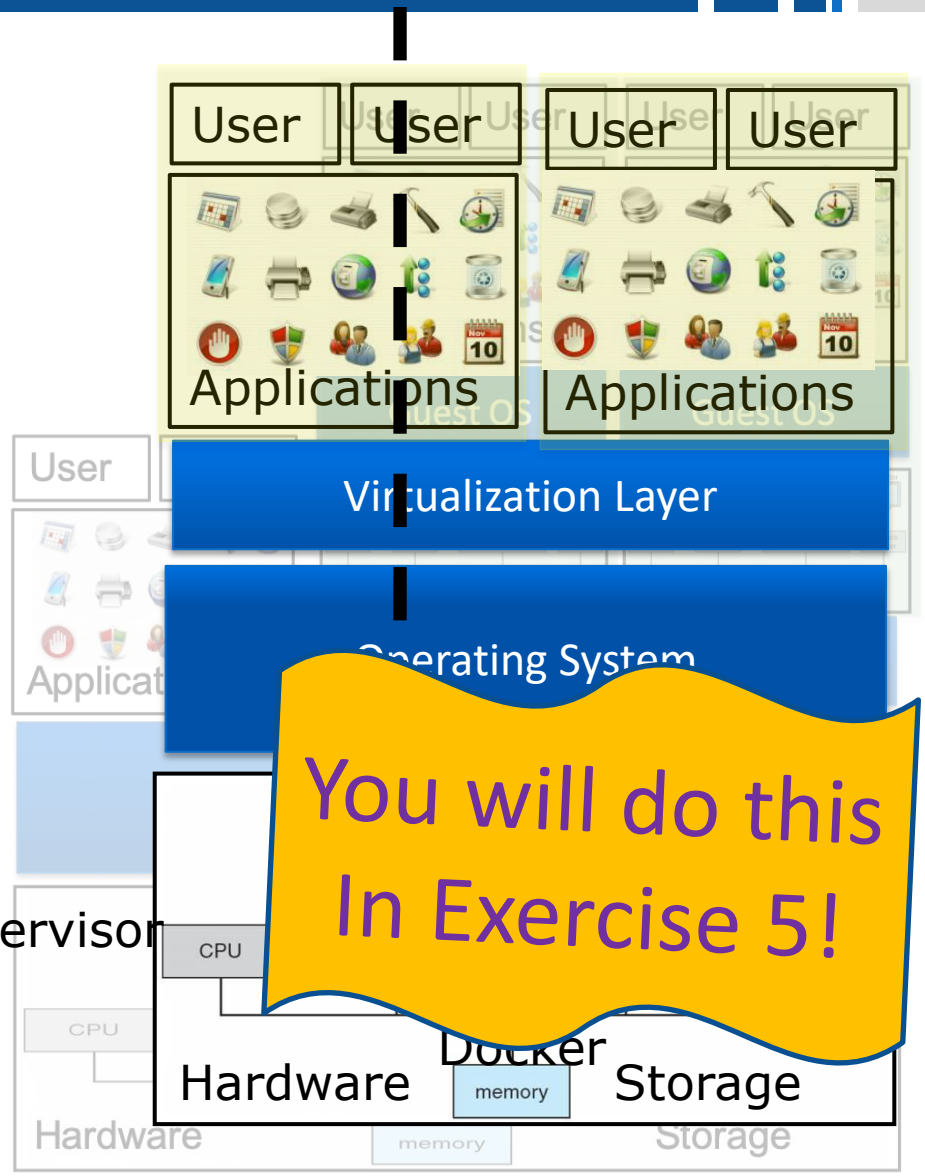


Types of Hypervisors



Type I Hypervisor
Native
Bare-metal

Type II Hypervisor
Hosted



You will do this In Exercise 5!

Virtualization Basics

- Hypervisor is like OS kernel
- VMs are like processes
- Hypervisor controls everything
  - Schedules VMs
  - Allocates memory
  - Multiplexes I/O devices
- But… the VM OSs think they run directly on the HW and control everything
- The hypervisor needs to fake it

Virtualization Mechanics (How to Fake It)

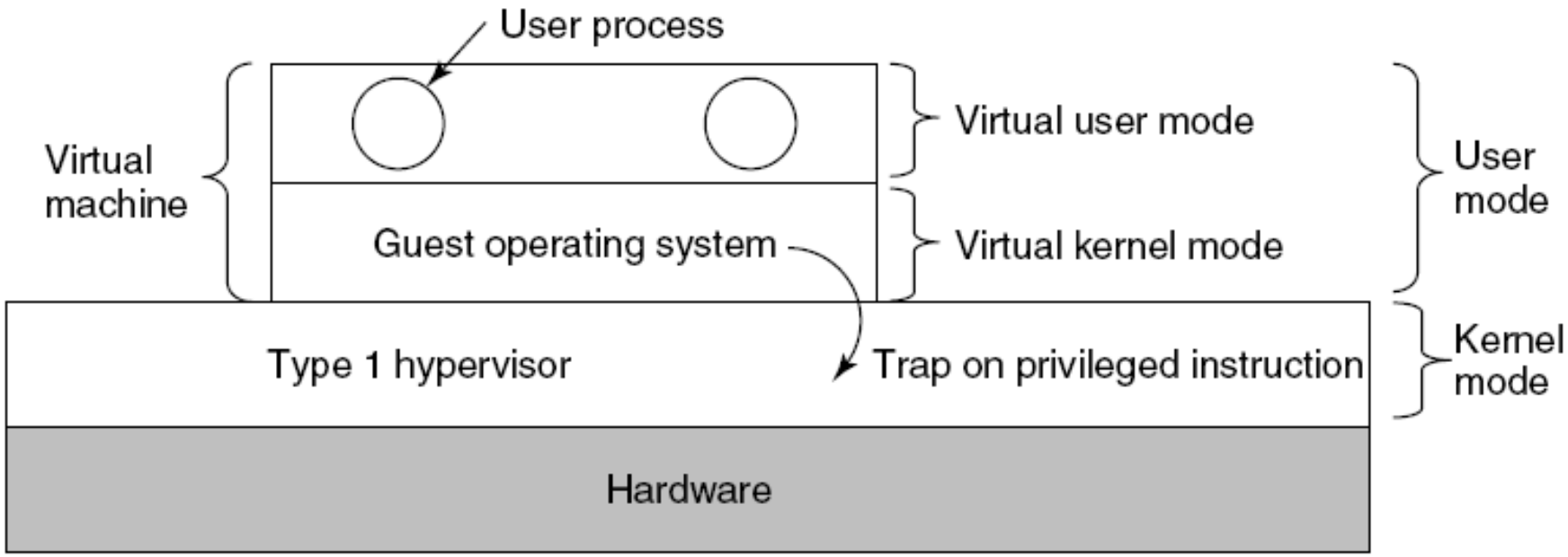
- Trap and Emulate
- Binary Translation
- Paravirtualization
- Hardware Assistance

Trap and Emulate Type-1 Hypervisor

- Runs on "bare metal"
- Virtual machines run directly on HW in user mode
  - Bo
- Trap
  - Ap ne
    hy er (in
    us
  - If t os to
    the
    hyp half
- Can work if the number of traps is not too large (and if the architecture supports T&E)

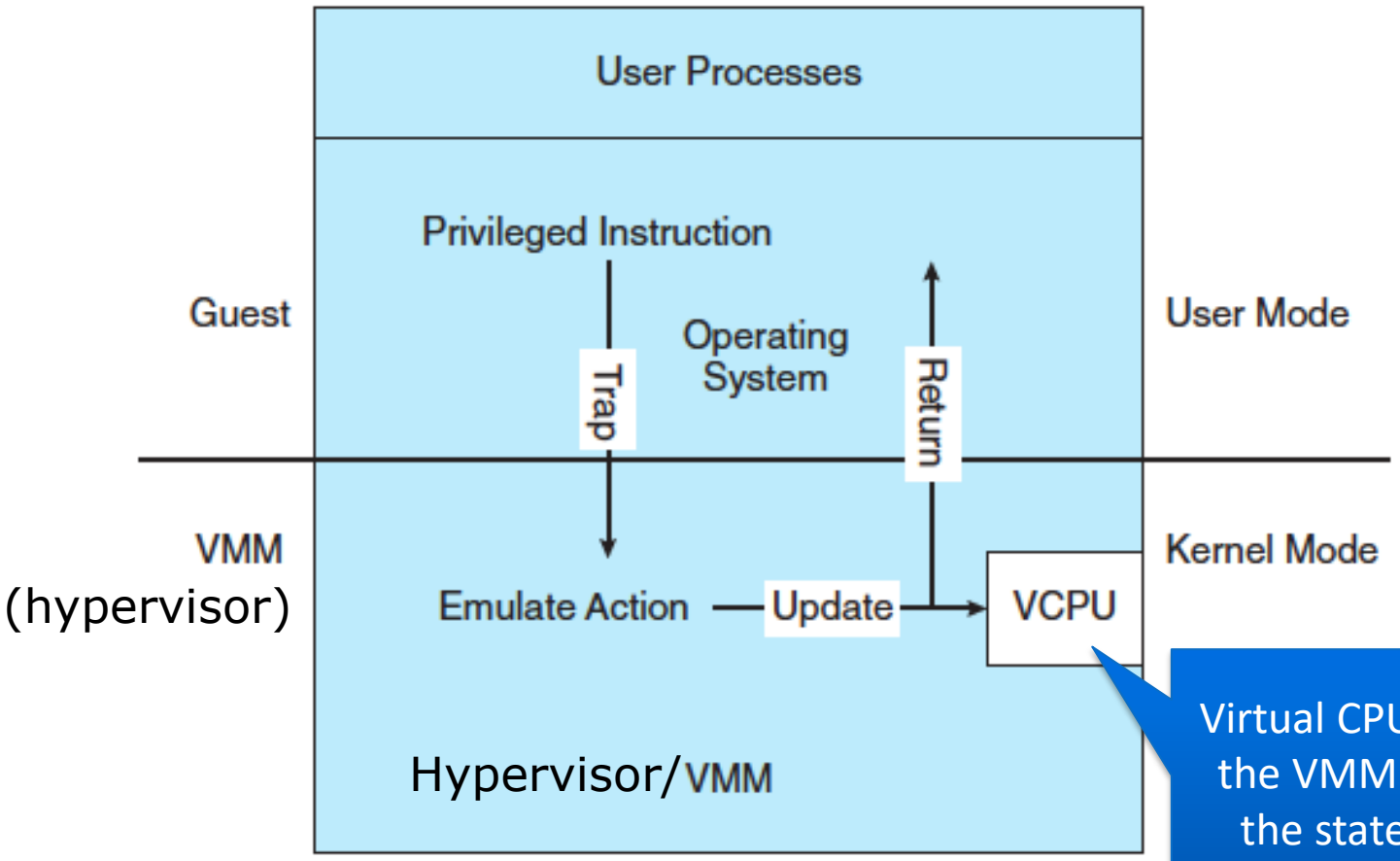
- Assumption: all sensitive instructions are privileged
  - So cause trap if attempted in user mode
- Was not so in Intel x86 architecture in the 1980s
- Is so on modern architectures

Type 1 Hypervisors



When the operating system in a virtual machine executes a kernel-only instruction, it traps to the hypervisor

Trap and Emulate



Virtual CPU (maintained by the VMM and represents the state of the CPU of each VM)

Dynamic Binary Translation

Needed if not all sensitive instructions are privileged

1. Translate all VM code blocks into safe code
   a) For normal code use identity translation
   b) For sensitive code, replace with “hypercalls” (calls to the hypervisor, like system calls)
2. Put this in the code cache
3. From now on, whenever this code is encountered, the safe version will be executed

Type-2 Hypervisor

- Type 1 duplicates much OS functionality
- Type 2 avoids duplication and uses OS services
  - OS schedules VMs
  - OS allocates memory
  - Files are used to emulate disks
  - VMs can be killed with signals
- Requires a hypervisor-related kernel module

Paravirtualization

- Modify Guest OS so that all calls to privileged instructions are changed to hypervisor calls (hypercalls)
  - Reduce number of traps, remove sensitive instructions…
  - Need to recompile the Guest OS specifically for each host OS
- Much easier (and more efficient) to modify source code than to emulate hardware instructions (as in binary translation).
- But cannot run any guest OS as is
  - Need access to source code

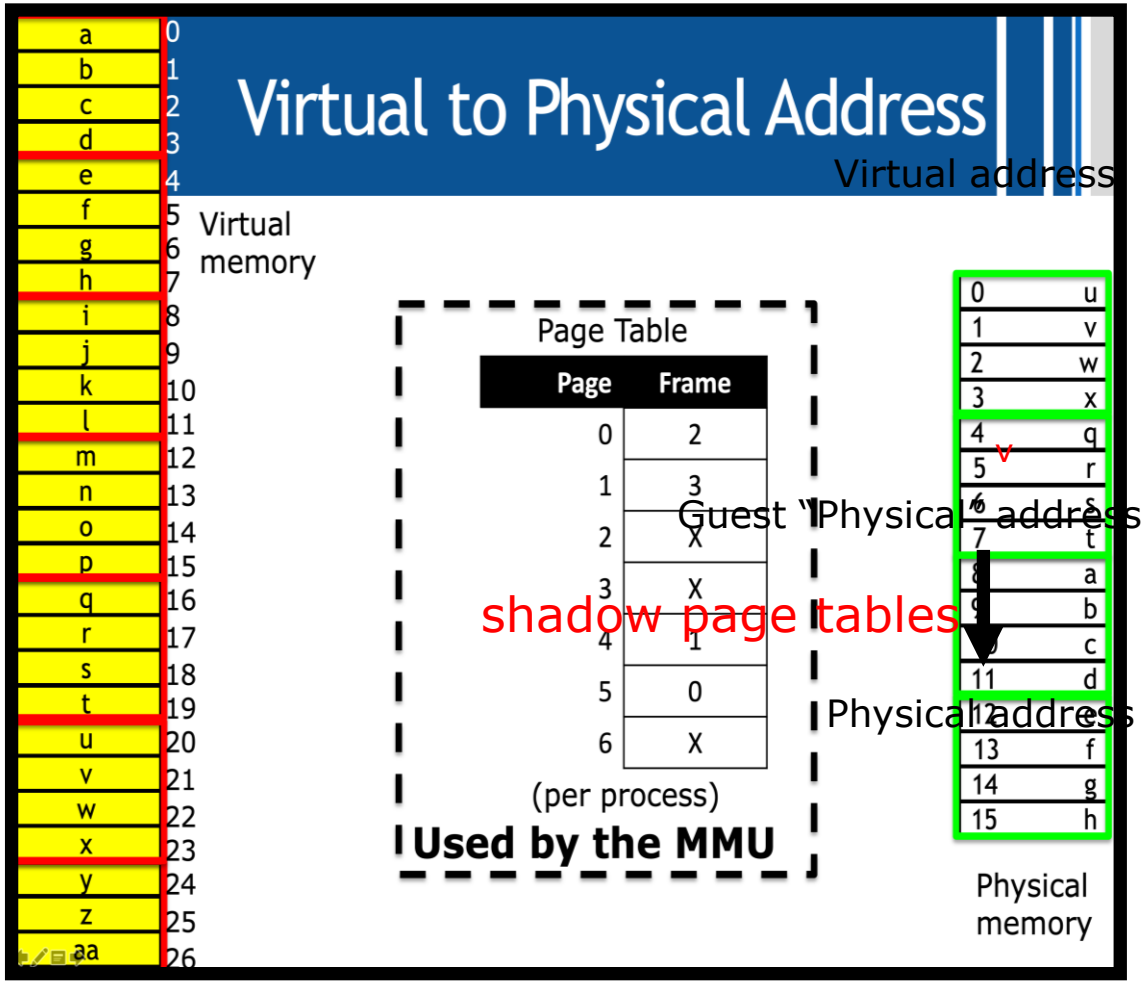
Hardware Assistance for Type 1

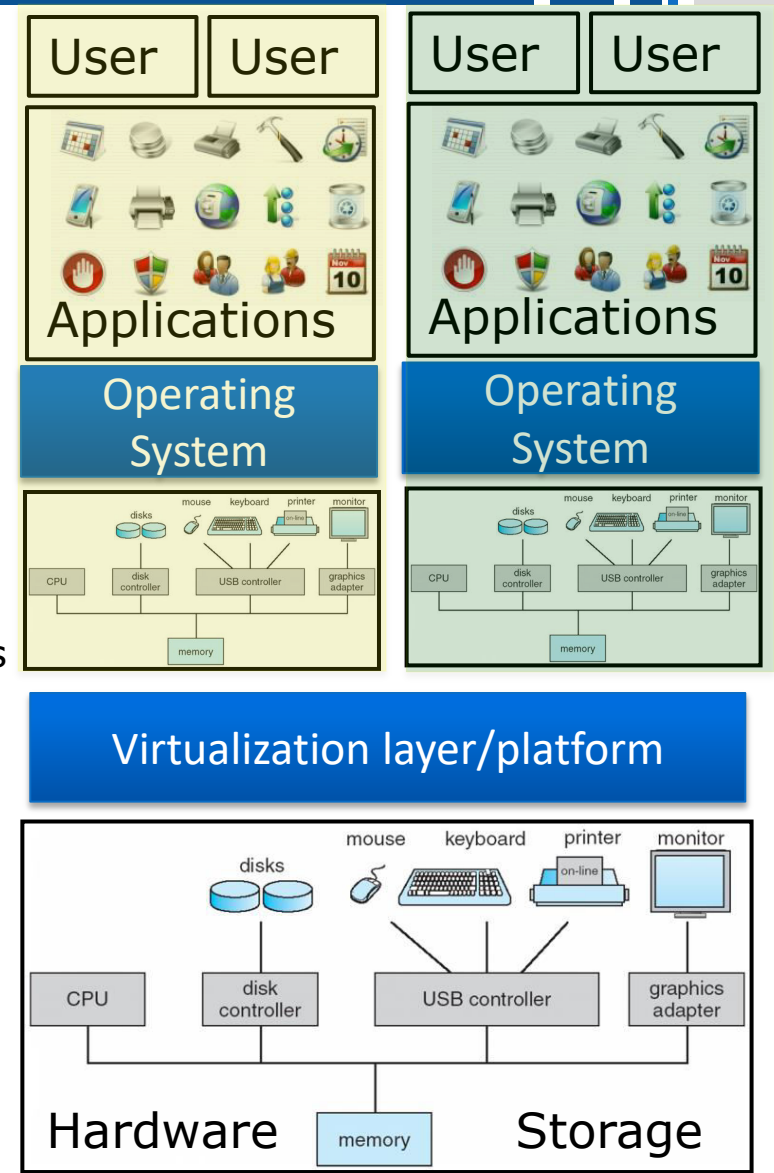
- Comply with Popek/Goldberg theorem
  - All sensitive operations must be privileged → cause trap which can be caught
- Intel VT/AMD SVM (2006)
- Support virtualization in the processor hardware:
  - New privilege mode
  - New instructions (vmrun, vmexit)
  - New data structure: VM control block

Products (partial List)

- Microsoft – Virtual PC, Hyper-V
- QEMU – Processor Emulation & VM
- Sun Microsystems – xVM, VirtualBox
- VMware – ESX Server, Workstation, Fusion, Player, Server
- Xen – Xen
- VirtualIron

Virtualizing Virtual Memory





Virtualizing Virtual Memory

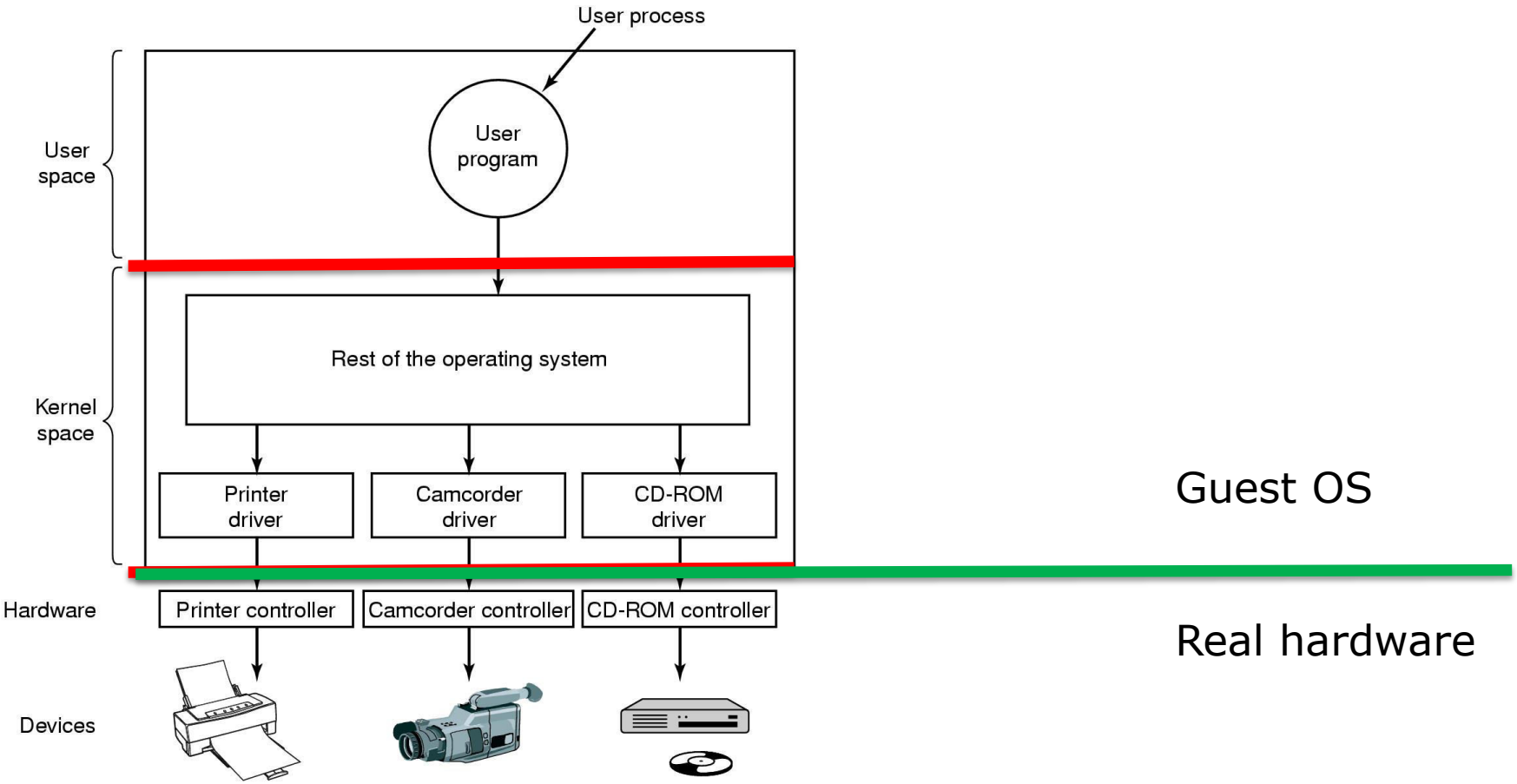
More points that need to take care of:

- What happens when page tables at VMs are being updated?
- How to interact with the TLB?

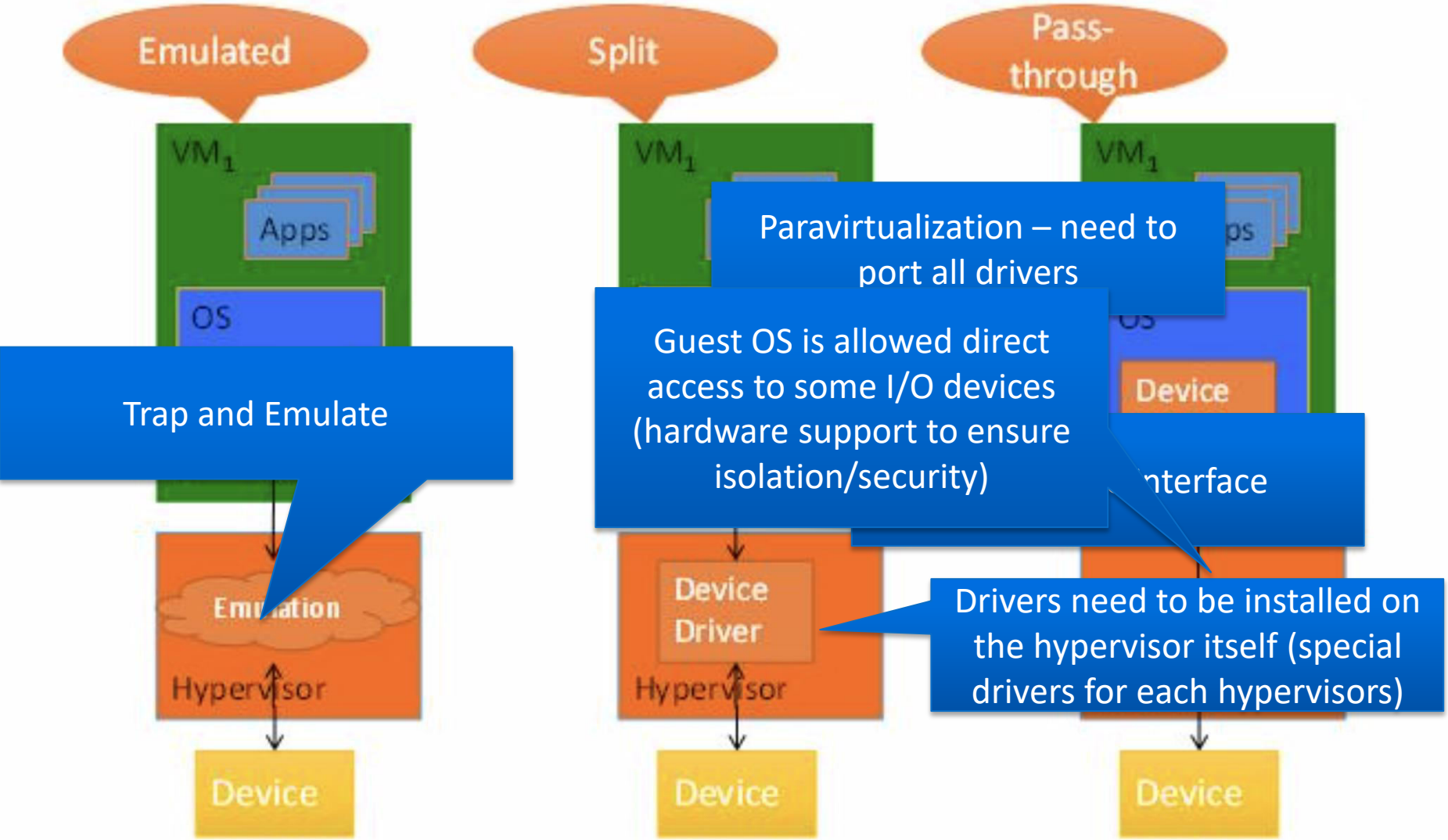
In paravirtualization, more opportunities to optimize things (VM knows it doesn't run natively)

Since 2008: Hardware-assisted page tables called *nested paging*

I/O Virtualization



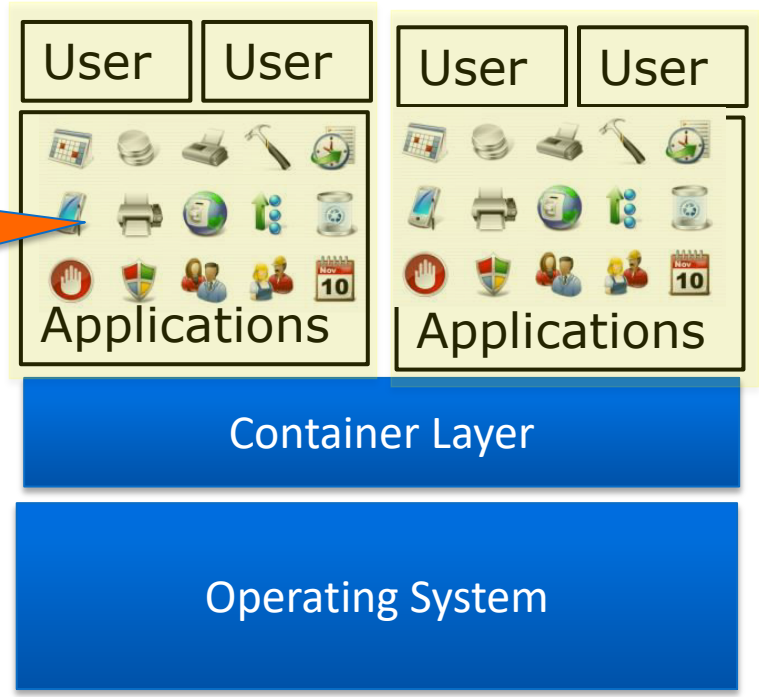
I/O Virtualization: Device Models



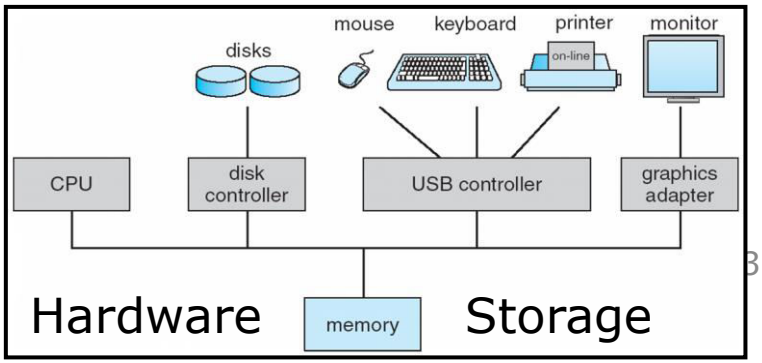
Containers

Containers are middle-ground between processes and VMs. For example, processes share the file systems and containers do not. Containers mostly have:

- Application
- Dependencies
- Libraries
- Binaries
- Configuration files



Virtualizing the OS and not the hardware. Each Container has its own file-system, memory, & OS resources as if it runs by itself



Containers and Processes

- Wait! But we have said that processes are virtual machines as the application see it.
  - Exactly like the containers
- Another view on containers: packaging (a set of) processes having their own OS resources
  - Defined by the concept of namespaces
  - Controlled by the concept of cgroups

Namespaces

- A namespace wraps a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource.
- 7 namespaces are defined: PID, User, Mount, Network, IPC, UTS, cgroups

Namespaces and cgroups

- Process has its own address space and restricted privileges
- Containers can be defined as having its own namespaces
  - Container A has its own PID namespace, implying it sees only processes in the container. The first process it the container has ID 1, etc. Container B's Process with ID 1 is a different process
- Resources are controlled by cgroups
  - How much of the resource should I give a container

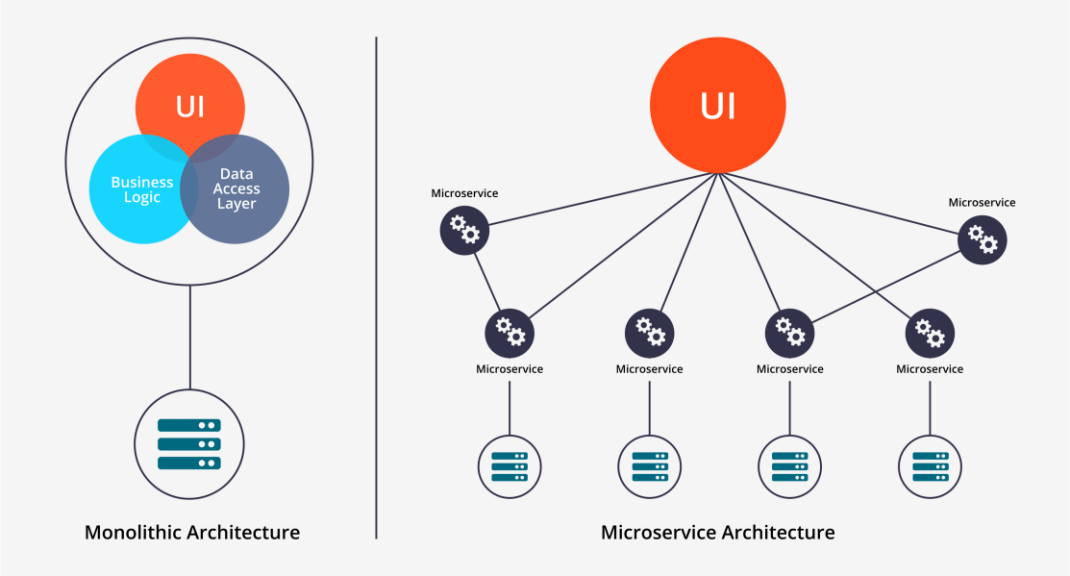
Comparison

	Process	Container	VM
Definition	A representation of a running program.	Isolated group of processes managed by a shared kernel.	A full OS that shares host hardware via a hypervisor.
Use case	Abstraction to store state about a running process.	Creates isolated environments to run many apps.	Creates isolated environments to run many apps.
Type of OS	Same OS and distro as host,	Same kernel, but different distribution.	Multiple independent operating systems.
OS isolation	Memory space and user privilieges.	Namespaces and cgroups.	Full OS isolation.
Size	Whatever user's application uses.	Images measured in MB + user's application.	Images measured in GB + user's application.
Lifecycle	Created by forking, can be long or short lived, more often short.	Runs directly on kernel with no boot process, often is short lived.	Has a boot process and is typically long lived.

Source: https://jessicagreben.medium.com/

Containers in the Real World

- Many applications in the cloud today run in a containerized manner
- Monolithic services → Microservices
- Microservices may be implemented as containers
- Orchestration syste
  deployment, scalir
  discovery, network
  - Kubernetes



Summary

- Virtualization provides a way to consolidate OS installations onto fewer hardware platforms
- 4 basic approaches
  - type 1 hypervisor
  - type 2 hypervisor
  - Paravirtualization
  - Containers
- Must also account for virtual access to shared resources (memory, I/O)