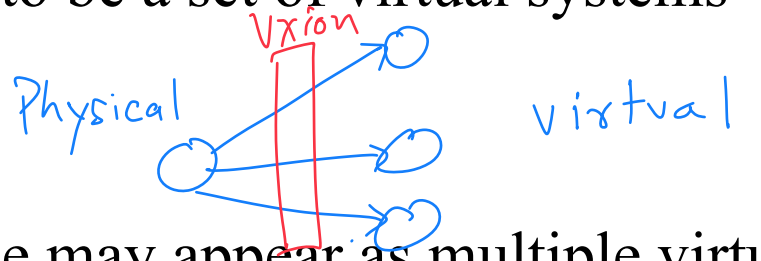
Introduction to Virtual Machines

Kartik Gopalan

From
"Virtual Machines" ,Smith and Nair, Chapter 1
Also, Chapter 7 Andrew Tanenbaum's book

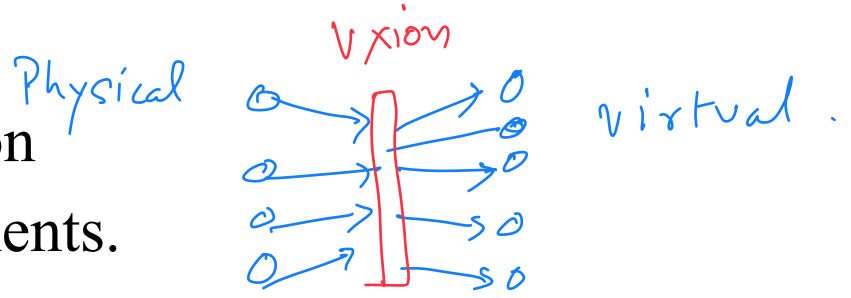
Virtualization

- Makes a real system appear to be a set of virtual systems
- One-to-many virtualization
  - E.g. one physical machine may appear as multiple virtual machines
  - one physical disk may look like multiple virtual disk
  - one physical network may look like multiple virtual networks

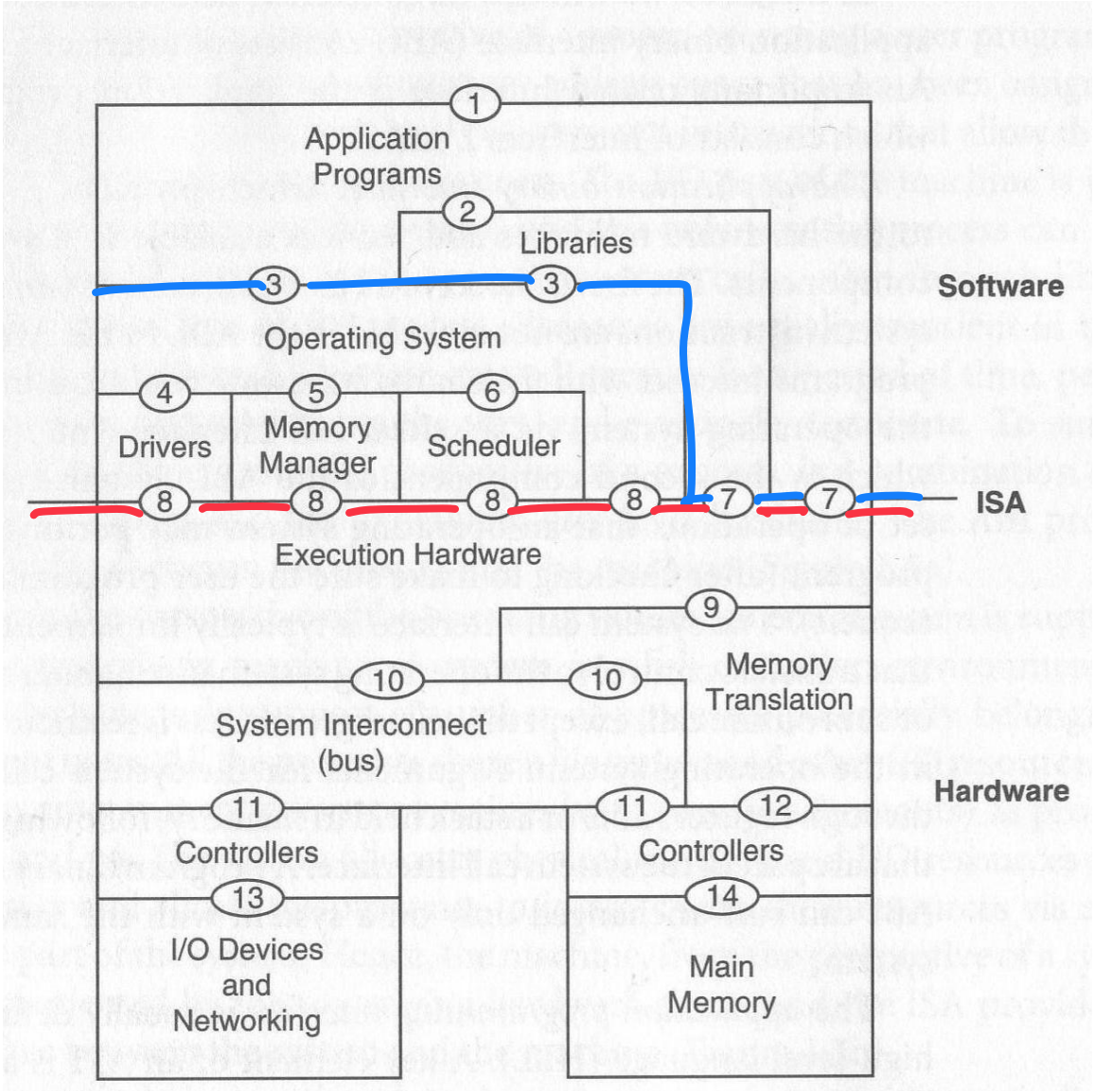


- Many-to-one virtualization
  - Many physical machines/disks/networks may appear to look like one virtual machine/disk/network etc

- Many-to-many virtualization
  - Extend the above statements.



Interfaces of a computer system

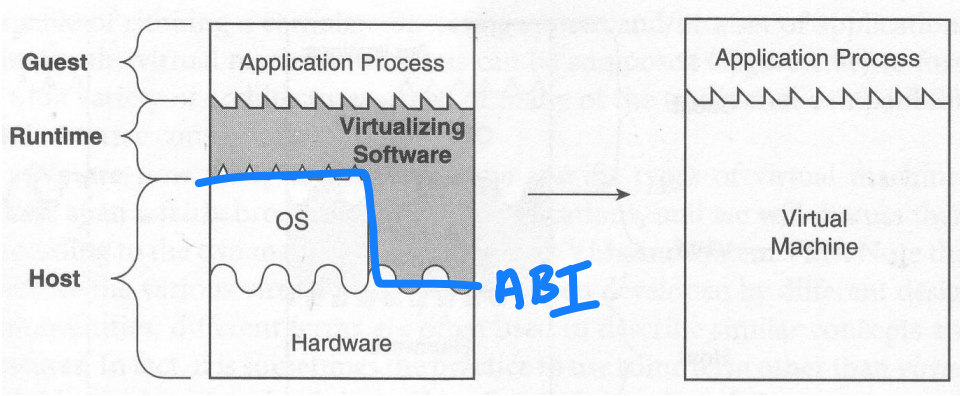


User ISA : 7
System ISA : 8
Syscalls : 3
ABI : 3, 7
API : 2,7

Two Types of VMs

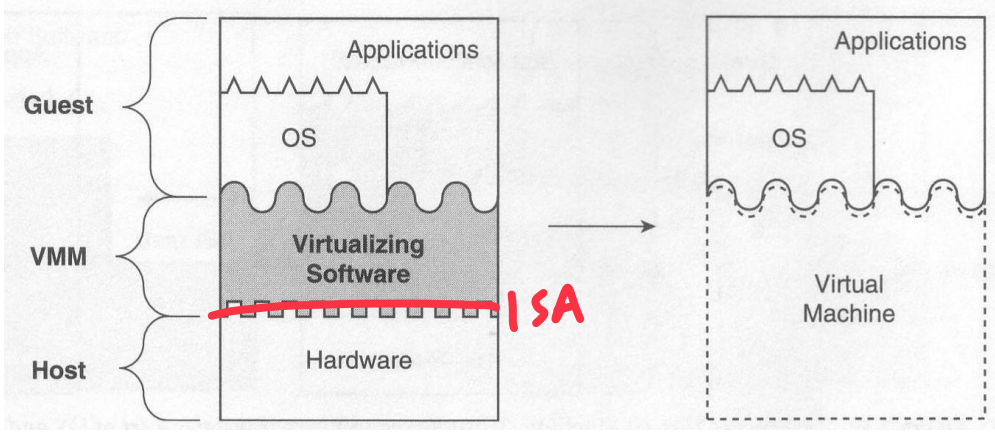
Process VM

- Virtualizes the ABI
- Virtualization software = Runtime
  - Runs in non-privileged mode (user space)
  - Performs binary translation.
- Terminates when guest process terminates.



System VM

- Virtualizes the ISA
- Virtualization software = Hypervisor
  - Runs in privileged mode
  - Traps and emulates privileged instructions
- Lasts as long as physical host is alive



System Virtual Machines

(focus of this lecture)

System Virtual Machines

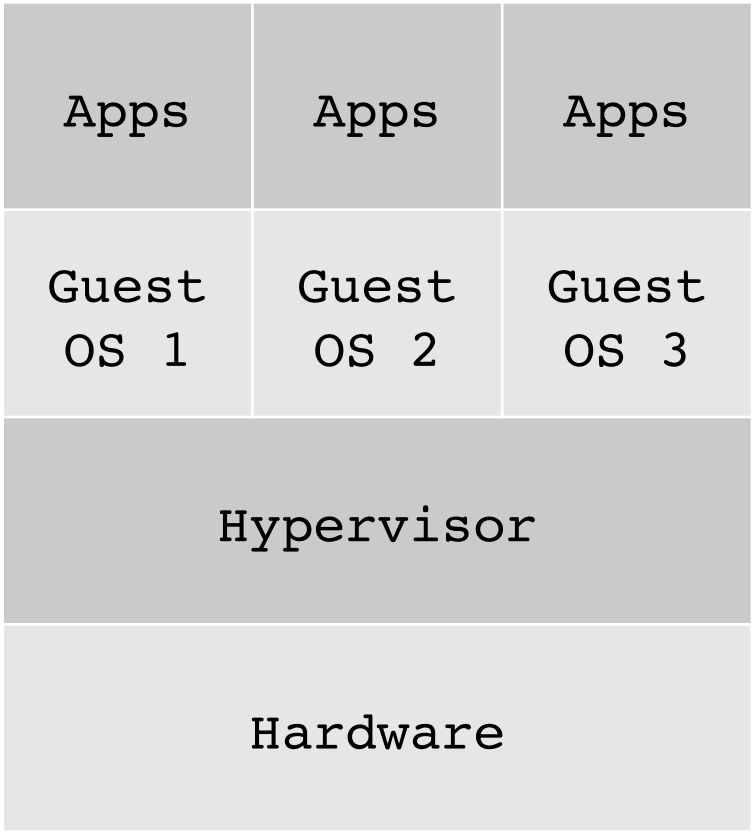
- Virtualize the whole machine ... all resources
  - CPU + memory + I/O

- Uses:
  - Multiple OSes on one machine, including legacy OSes
  - Isolation
  - Enhanced security
  - Live migration of servers
  - Virtual environment for testing and development
  - Platform emulation
  - On-the-fly optimization
  - Realizing ISAs not found in physical machines

Hypervisor

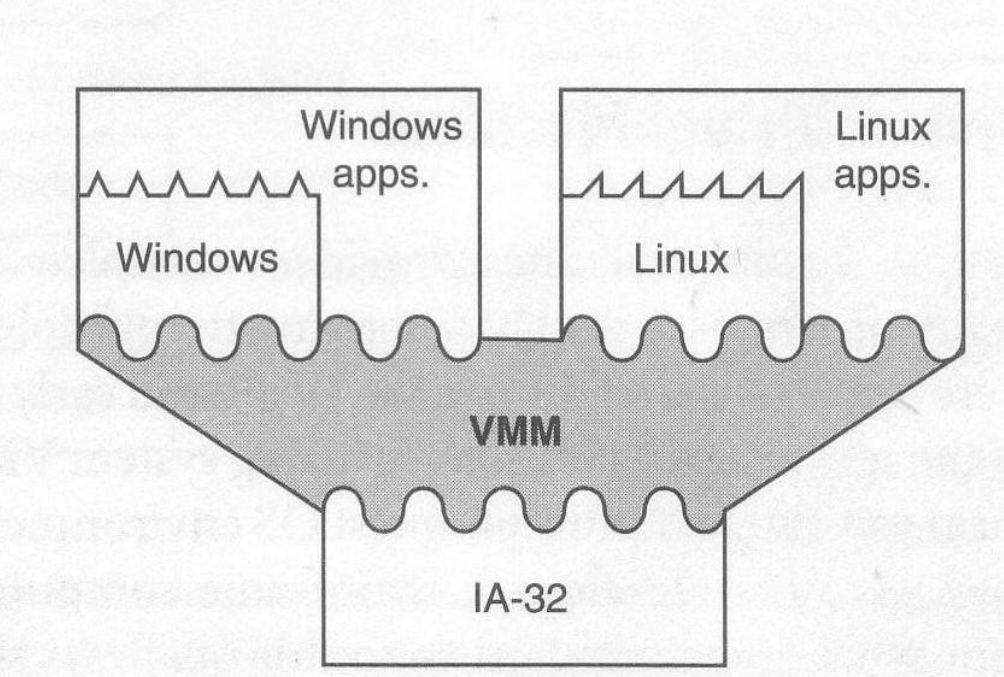
- Also called Virtual Machine Monitor (VMM)

- A hypervisor is an operating system for operating systems
  - Provides a virtual execution environment for an entire OS and its applications
  - Controls access to hardware resources
  - When guest OS executes a privileged instruction, Hypervisor intercepts the instruction, checks for correctness and emulates the instruction.



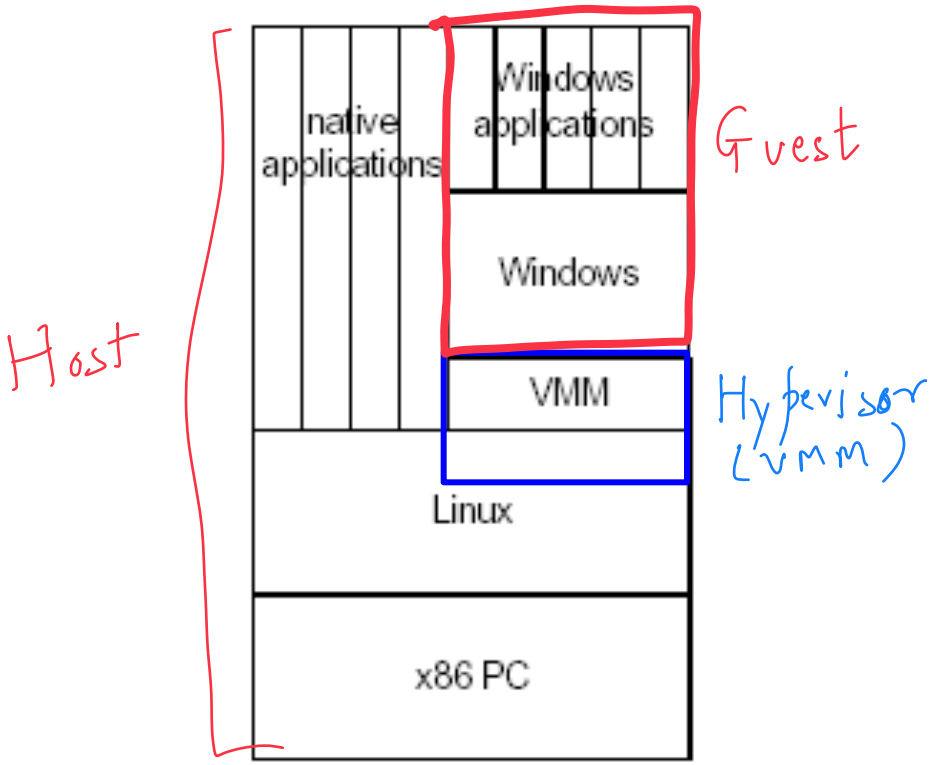
Type 1 Hypervisors (Classical System VMs)

- Hypervisor executes natively on the host ISA
- Hypervisor directly controls hardware and provides all device drivers
- Hypervisor emulates sensitive instructions executed by the Guest OS
- E.g. KVM and VMWare ESX Server

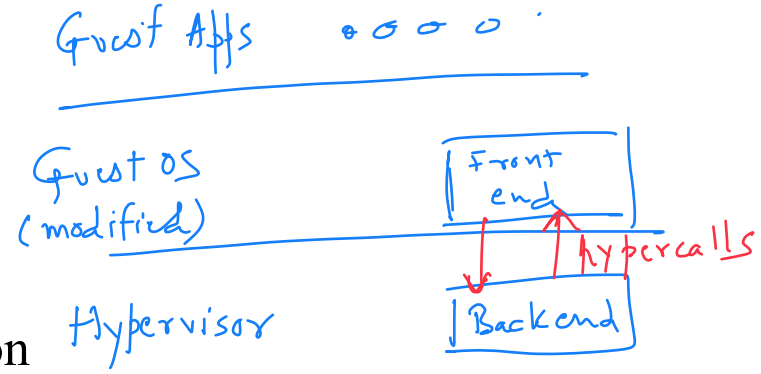


Type-2 Hypervisors (Hosted VMs)

- A host OS controls the hardware
- The Hypervisor runs partly in process space and partly in the host kernel
- Hypervisor Relies on host OS to provide drivers
- E.g. VMWare Desktop Client



Para-virtualized VMs



- Modify guest OS for better performance

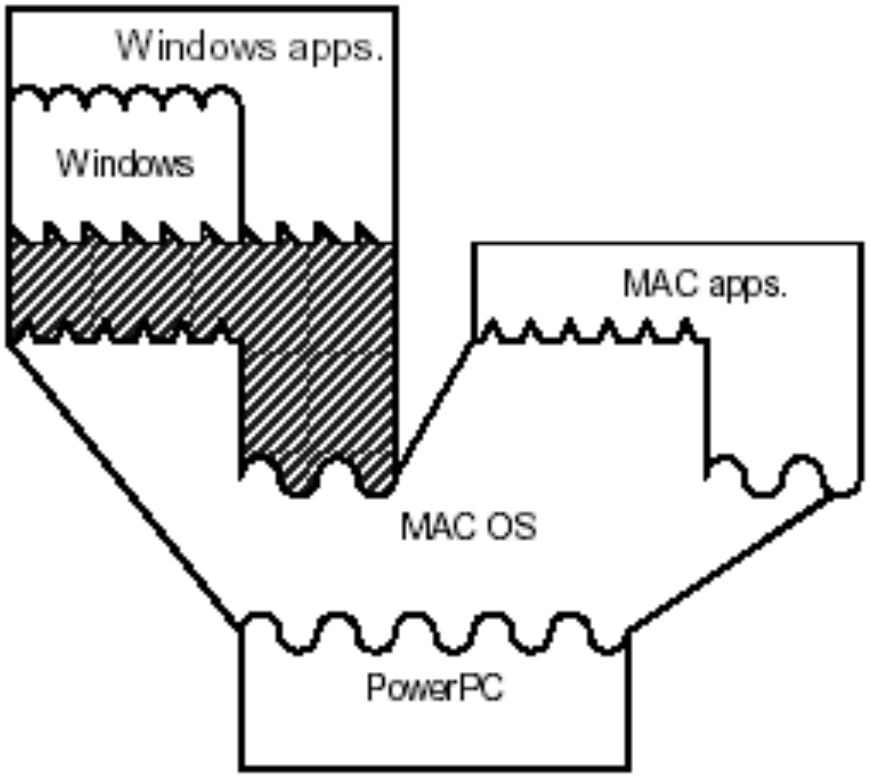
- Traditional Hypervisors provide full-virtualization
  - They expose to VMs virtual hardware that is functionally identical to the underlying physical hardware.
  - Advantage : allows unmodified guest OS to execute
  - Disadvantage: Sensitive instructions must be trapped and emulated by Hypervisor.
  - E.g. KVM and VMWare ESX provide full virtualization

- Para-virtualized VM
  - Sees a virtual hardware abstraction that is similar, but not identical to the real hardware.
  - Guest OS is modified to replace sensitive instructions with "hypercalls" to the Hypervisor.
  - Advantage: Results in lower performance overhead
  - Disadvantage: Needs modification to the guest OS.
  - E.g. Xen provides both para-virtual as well as full-virtualization

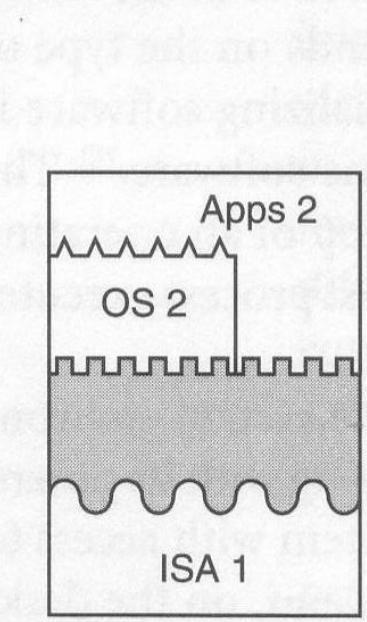
- Often traditional Hypervisors are partially para-virtualizated
  - Device drivers in guest OS may be para-virtualized whereas CPU and Memory may be fully virtualized.

Whole System VMs: Emulation

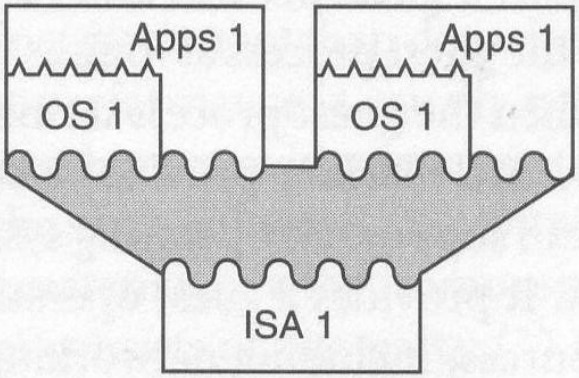
- Host and Guest ISA are different
- So emulation is required
- Hosted VM + emulation
- E.g. Virtual PC (Windows on MAC)



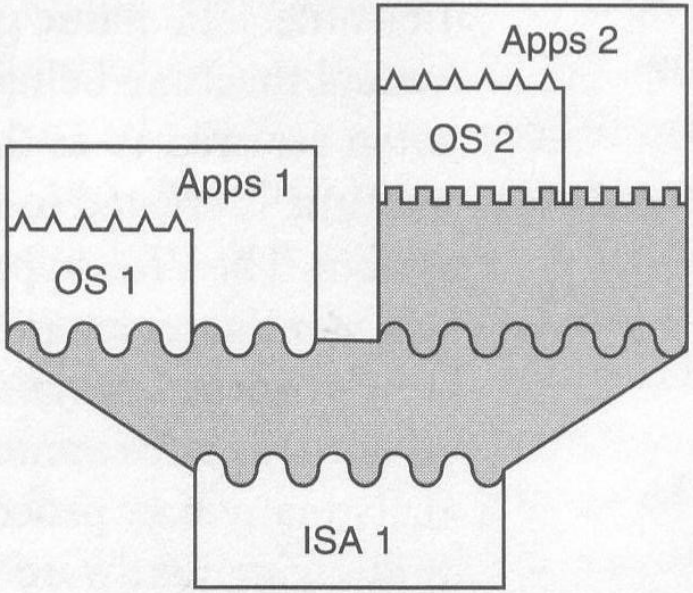
What can you do with system VMs?



Emulation & Optimization



Replication



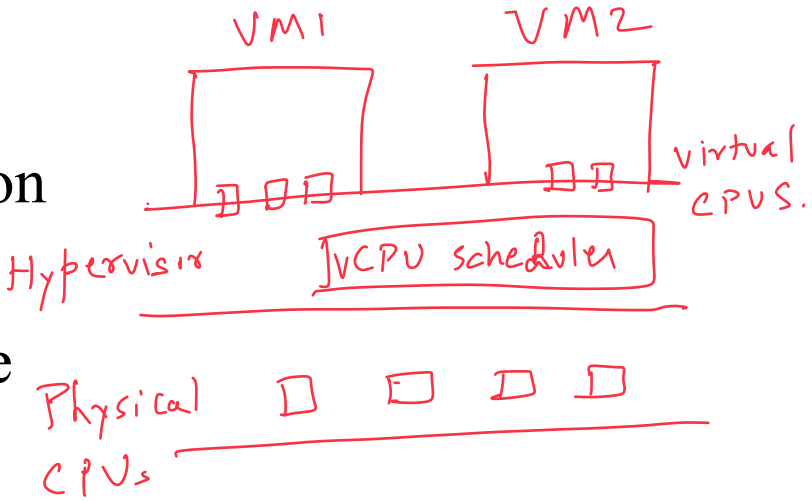
Composition

- Emulation: Mix-and-match cross-platform portability
- Optimization: Usually done with emulation for platform-specific performance improvement
- Replication: Multiple VMs on single platform
- Composition: form more complex flexible systems

Virtualizing individual resources in System VMs

CPU Virtualization for VMs

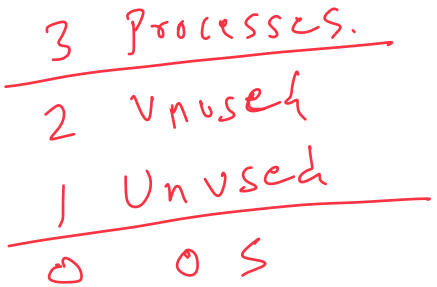
- Each VM sees a set of "virtual CPUs"



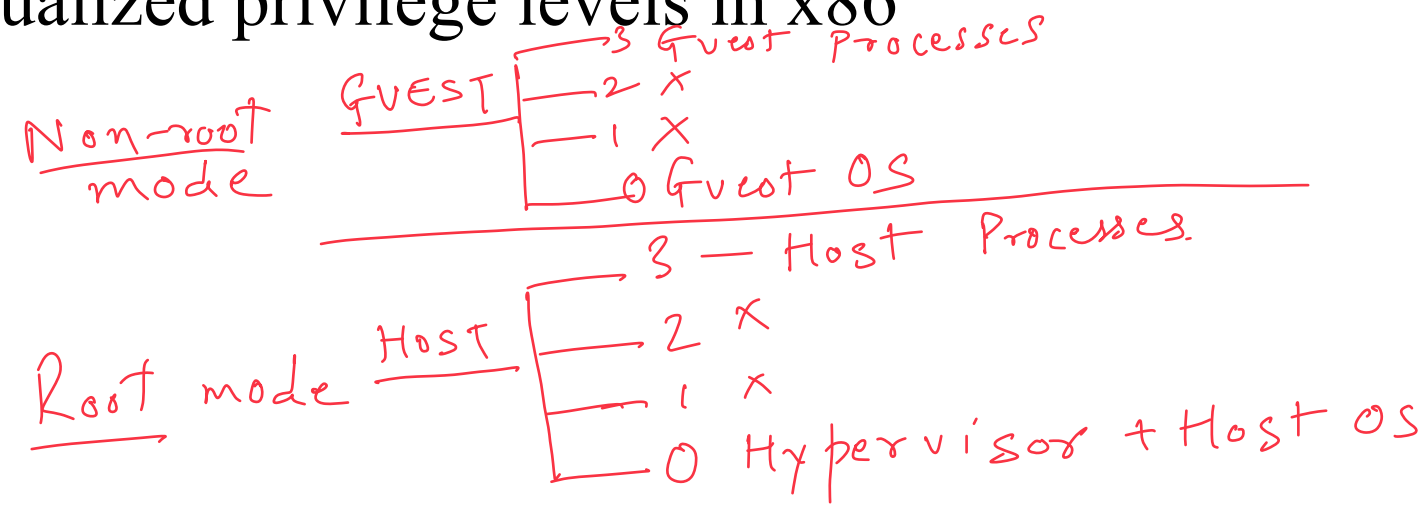
- CPU Hardware support for virtualization
  - Intel provides the VTx interface
  - AMD provides the AMD-v interface
- These special ISA interfaces allow the Hypervisors to efficiently emulate privileged instructions executed by the guest OS.
- When a vCPU in guest OS executes a privileged instruction
  - Hardware traps the instruction to the hypervisor
  - Hypervisor checks whether instruction must be emulated.
  - If so, Hypervisor reproduces the effect of the privileged operation.

CPU Privilege Levels

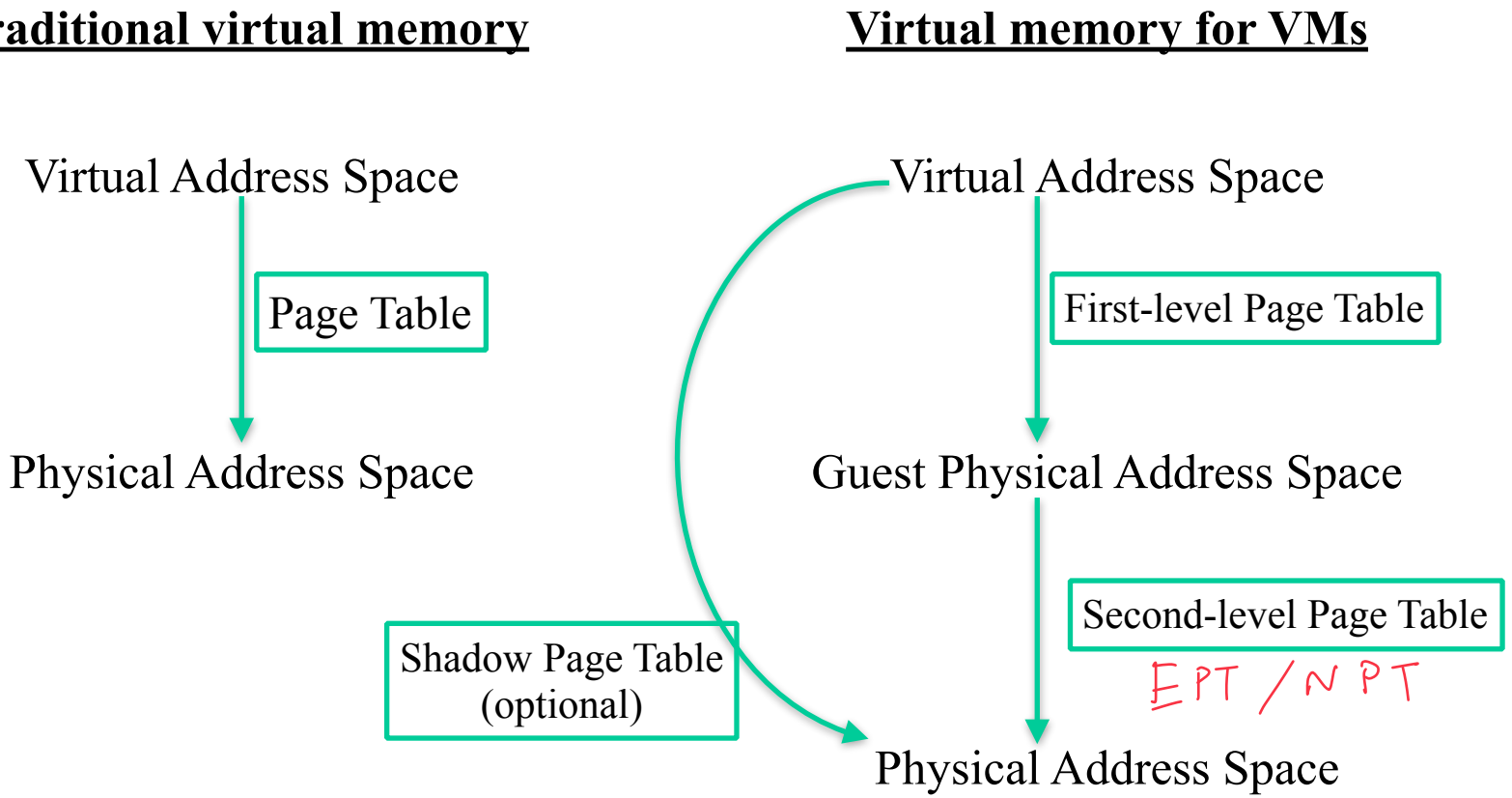
- Traditional non-virtualized privilege levels in x86



- Virtualized privilege levels in x86



Memory Virtualization for VMs



- Guest OS in each VM sees a "guest"-physical address (GPA) space instead of the physical addresses
- Often hardware supports two-level page tables
  - EPT in Intel VT-x and NPT in AMD-v
- When hardware doesn't, then Hypervisor needs to emulate two-level page tables using "shadow page tables".

I/O Virtualization for VMs

- Hypervisor provides a virtual version of each physical device
- I/O activity directed at the virtual device is trapped by Hypervisor and converted to equivalent request for the physical device.
- Options:
  - Device emulation (rare)
    - Hypervisor traps and emulates each I/O instruction from Guest in Hypervisor.
    - Very slow.
    - Difficult to emulate the effect of combinations of I/O instructions.
  - Para-virtual devices (common)
    - Special device drivers inserted in guest OS to talk to Hypervisor.
    - Most common.
  - Direct device access (increasingly common)
    - Allow the VM to directly access physical device.
    - Fastest option but not scalable.
    - Requires IOMMU and VT-d support from hardware.