INFRASTRUCTURE AS A SERVICE CLOUD ARCHITECTURES

CIT 3400

LECTURE 2

DR. AMOS CHEGE, PH.D.

LECTURER COMPUTER SCIENCE

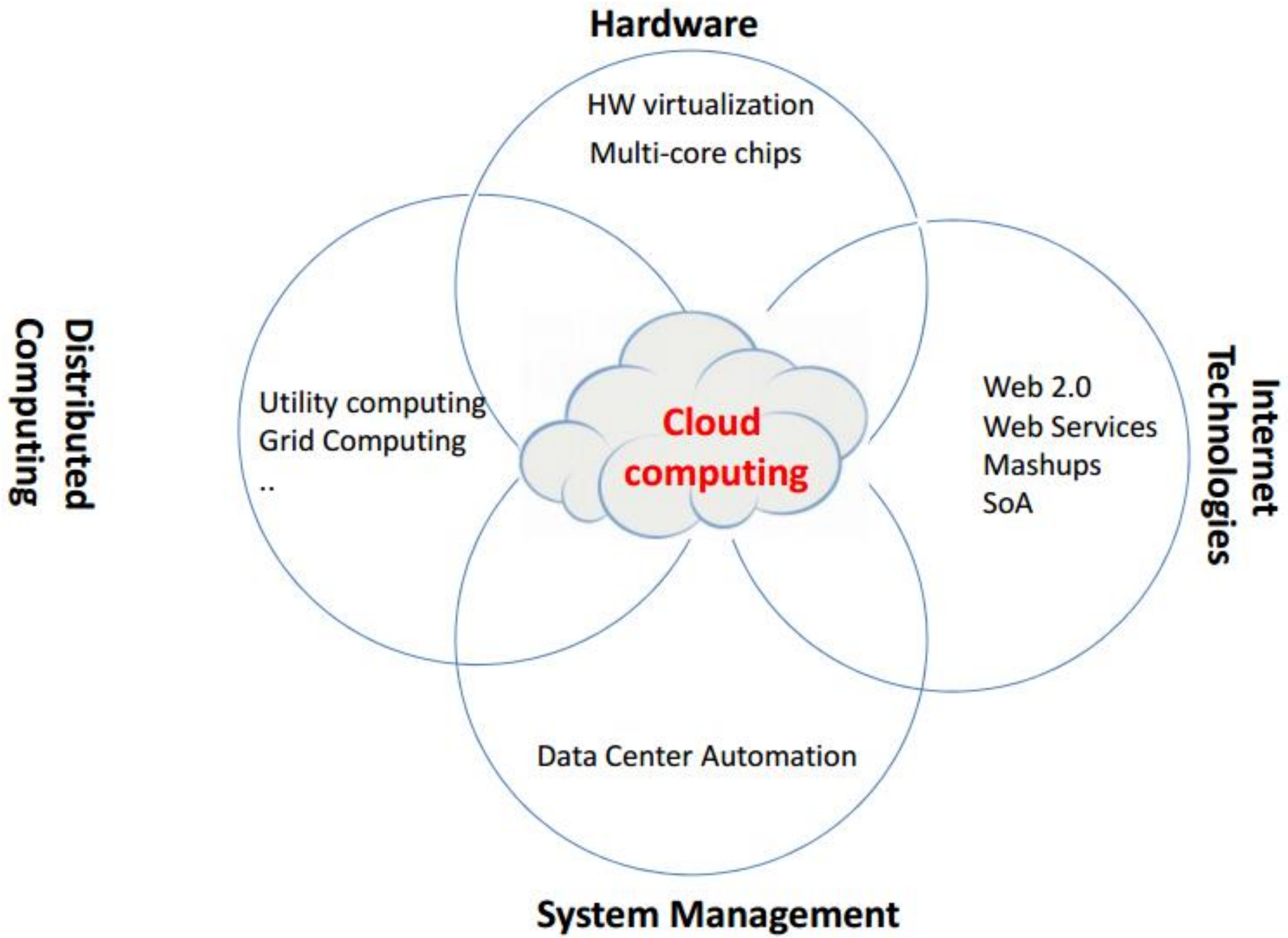
IAAS CLOUD ARCHITECTURES

- Introduction to IaaS
- Hardware virtualization
  - CPU
  - Memory
  - I/O
  - Network
- Software virtualization
  - Hypervisors
    - KVM
    - Xen
    - VirtualBox
  - Full Virtualization
  - Para Virtualization
  - Host OS Virtualization
  - Container-based Virtualization
- IaaS Ecosystems
  - Open Source
    - Eucalyptus
    - Openstack
    - Cloudstack
    - OpenNebula
    - Nimbus
  - Public Clouds
    - Amazon AWS
    - Google App/Compute Engines
    - Microsoft Azure
- Other Cloud Issues
  - Live Migration
  - Scalability
  - Availability
  - Management
  - Performance
  - Security

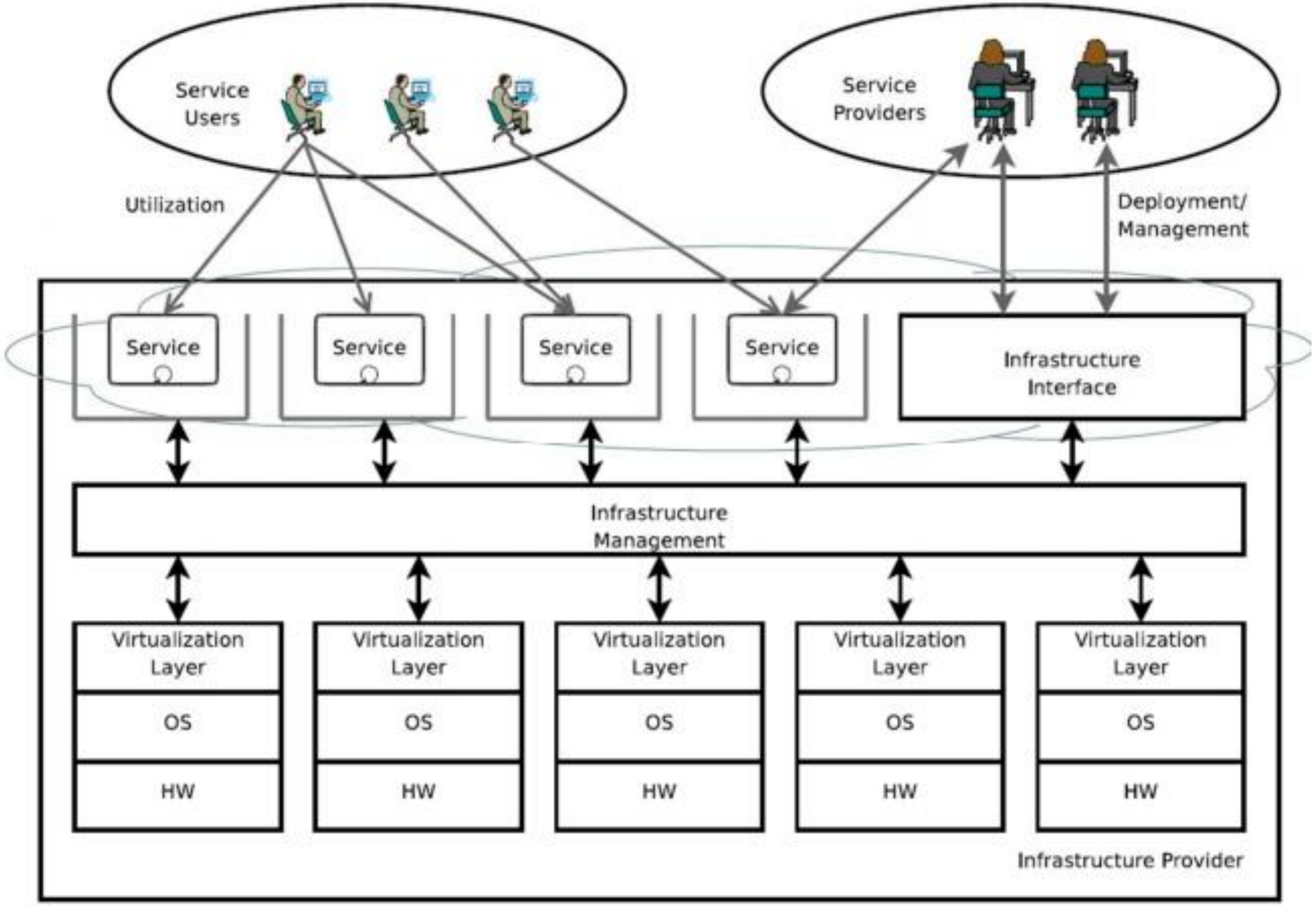
IAAS CLOUD ARCHITECTURES

- **Local Cloud?**
  - **What resources would we need to do this?**
    - Compute Servers
    - Persistent Storage Servers
    - VM Image Server(s)
    - Cloud Administrative Server(s)
    - Network Infrastructure
      - Copper

FACTORS ENABLING CLOUD COMPUTING



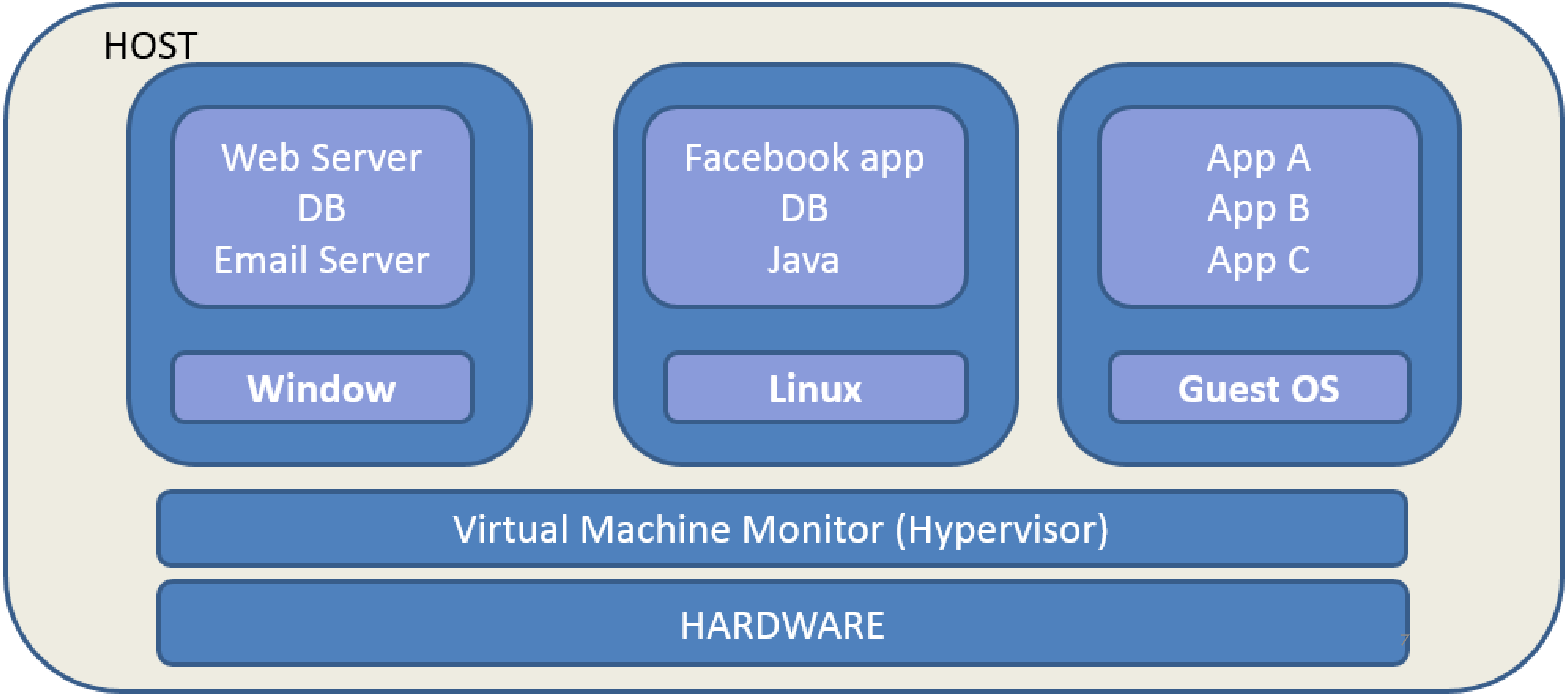
CLOUD ACTORS



IAAS MAIN IDEAS - VIRTUALIZATION

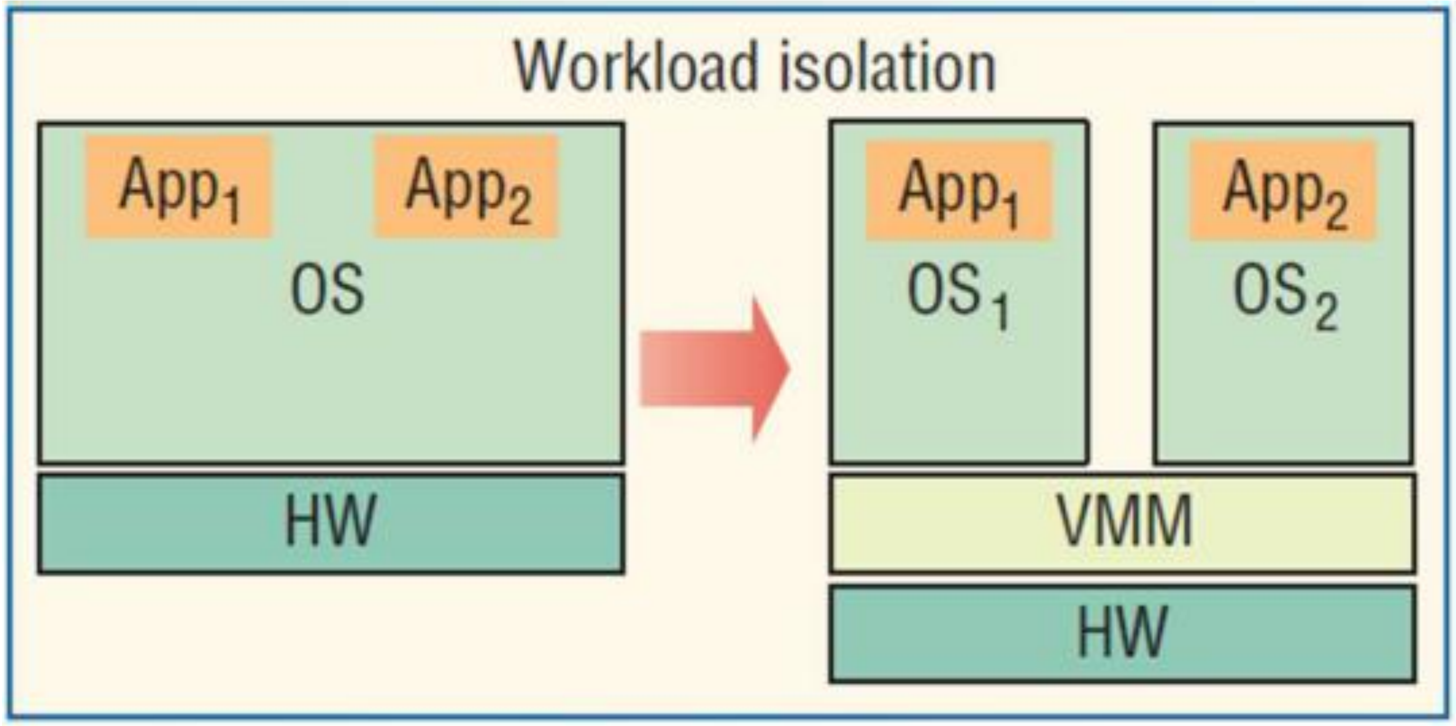
- **Memory:**
  - Virtual Memory and memory management
- **Multitasking:**
  - Several processes concurrently running on the same hardware
  - Hardware is shared thanks to special OS processes and CPU extensions
- **Virtual Machine:**
  - Concept began with the use of IBM mainframes
  - Abandoned with the advent of PC's, but now used again for cloud computing
  - Even different Instruction Set Architectures (ISAs) and/or OS kernels
  - Achieved using a Virtual Machine Monitor (VMM) or Hypervisor

IAAS Main Ideas – Virtualization



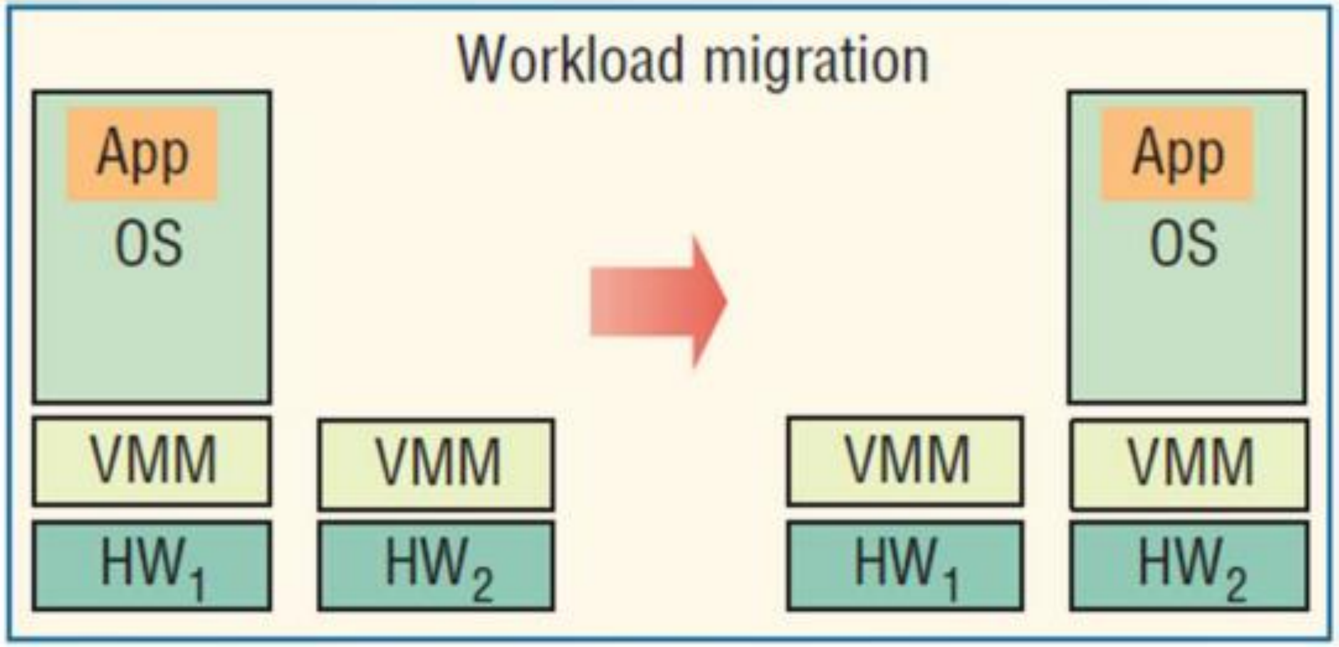
IAAS MAIN IDEAS - ISOLATION

- Through virtualization, workloads are isolated since all program instructions are fully confined inside a virtual machine (VM).
- Better reliability and performance is also achieved because software failures inside one VM do not affect others.

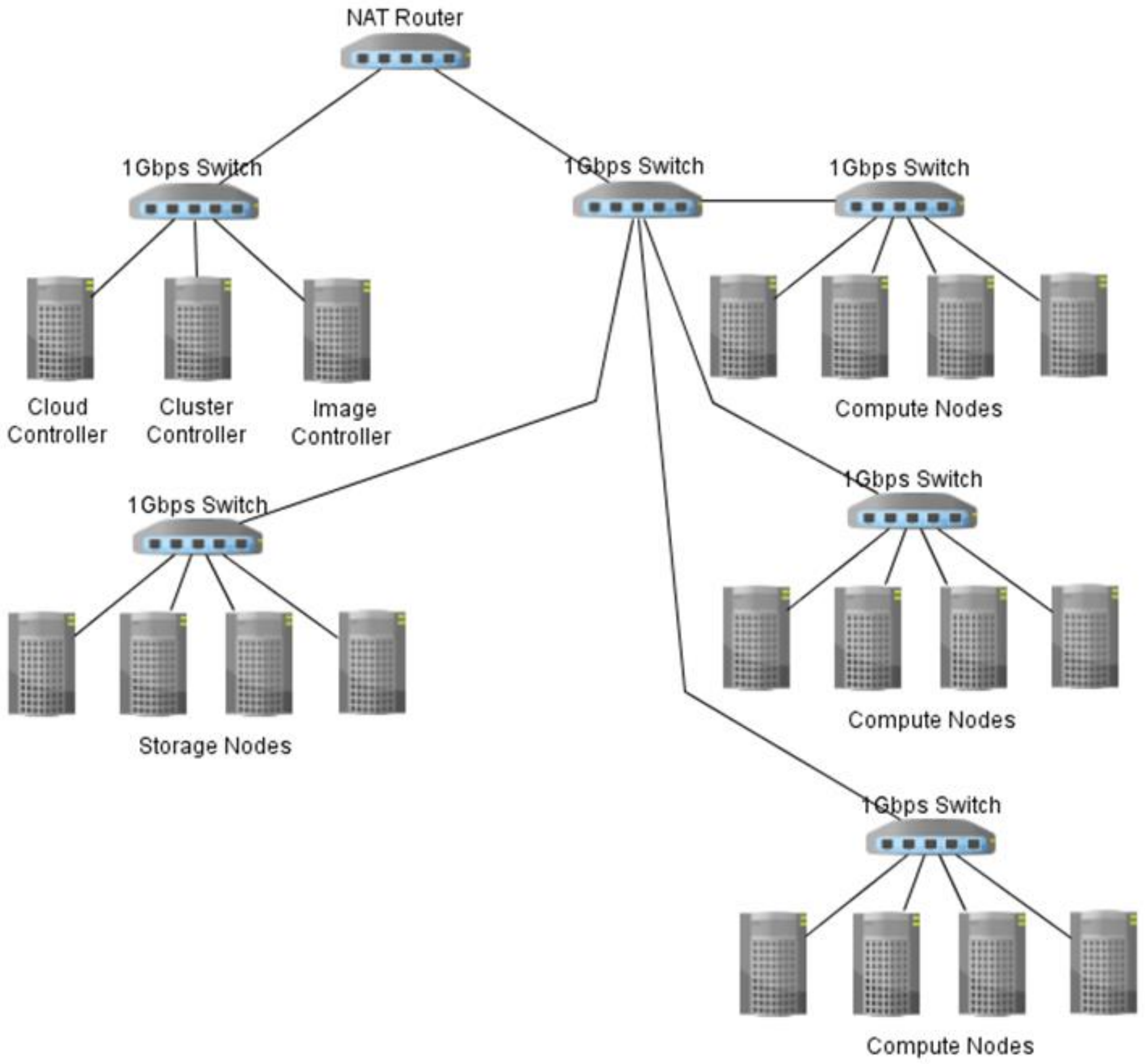


IAAS MAIN IDEAS – APPLICATION MOBILITY

- **Workload migration (application mobility) targets at facilitating hardware maintenance, load balancing, fault tolerance and disaster recovery.**
- **It is done by encapsulating a guest OS state within a VM and allowing it to be suspended, migrated to a different platform, and resumed immediately or preserved to be restored at a later date.**
  - A VM's state includes a full disk or partition image, configuration files, and an image of its RAM.



IAAS CLOUD ARCHITECTURES

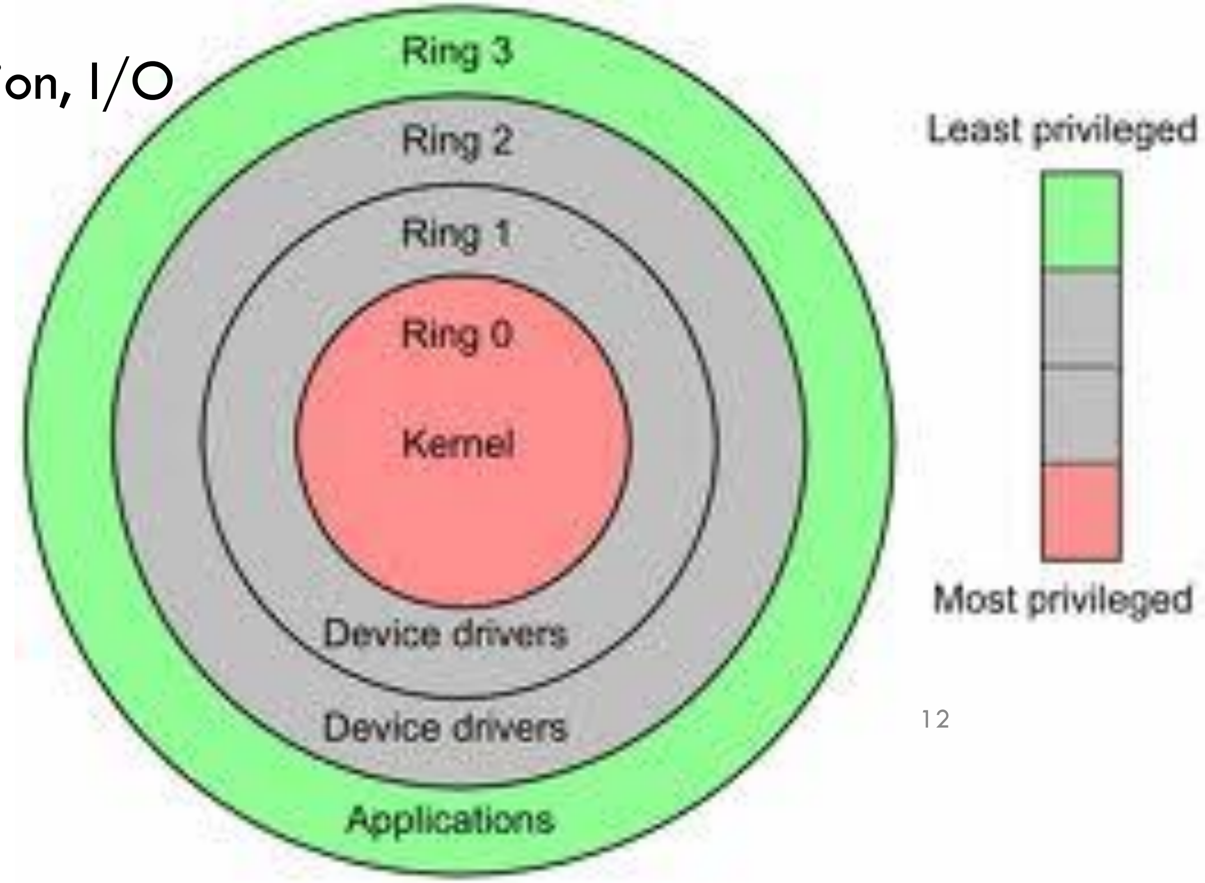


IAAS CLOUD ARCHITECTURES

- Introduction to IaaS
- Hardware virtualization
  - CPU
  - Memory
  - I/O
  - Network
- Software virtualization
  - Hypervisors
    - KVM
    - Xen
    - VirtualBox
  - Full Virtualization
  - Para Virtualization
  - Host OS Virtualization
  - Container-based Virtualization
- IaaS Ecosystems
  - Open Source
    - Eucalyptus
    - Openstack
    - Cloudstack
    - OpenNebula
    - Nimbus
  - Public Clouds
    - Amazon AWS
    - Google App/Compute Engines
    - Microsoft Azure
- Other Cloud Issues
  - Live Migration
  - Scalability
  - Availability
  - Management
  - Performance
  - Security

CPU ARCHITECTURE - MODES

- **Modern CPU status is usually classified as several modes.**
- **In general, we conceptually divide them into two modes :**
  - **Kernel mode (Ring 0)**
    - CPU may perform any operation allowed by its architecture, including any instruction execution, I/O operation, area of memory access, and so on.
  - **User mode (Ring 1 - 3)**
    - Ring 1: Reserved for device drivers
    - Ring 2: Privileged code such as user programs with I/O access permissions
    - Ring 3: Unprivileged code (Most user programs)



CPU ARCHITECTURE

- **By the classification of CPU modes, we divide instructions into following types:**
  - **Privileged instructions**
    - Those instructions that trap (interrupt) if the machine is in user mode and do not trap if the machine is in kernel mode.
  - **Sensitive instructions**
    - Those instructions that interact with hardware, which include control-sensitive and behavior-sensitive instructions.

CPU ARCHITECTURE - TRAPS

- **CPU trap:**
  - **When CPU is running in user mode, some internal or external events, which need to be handled in kernel mode, take place.**
    - System calls
    - Hardware interrupts
    - Exceptions
  - **Looks like an internal CPU interrupt since a trap handler acts like an interrupt handler**
    - Registers and stack pointers are saved
    - Context switch occurs in the CPU to handle the trap

CPU ARCHITECTURE - TRAPS

- **Trap types:**
  - **System Call**
    - Invoked by application in user mode.
    - For example, application ask OS for system I/O.
  - **Hardware Interrupts**
    - Invoked by some hardware events in any mode.
    - For example, hardware clock timer trigger event.
  - **Exception**
    - Invoked when unexpected error or system malfunction occur.
    - For example, execute privilege instructions in user mode.

CPU ARCHITECTURE – EMULATION VS. VIRTUALIZATION

- **Emulation:**
  - **Hardware components and interactions are replaced with software components.**
    - **Advantages:**
      - Entire hardware architectures can be created by software interfaces
      - It is possible to run isolated software on an emulated server, even software designed for different hardware architectures.
    - **Disadvantages:**
      - Software running in emulated environments have no direct access to hardware devices.
      - Using this technique creates an emulation tax (the processing cycles needed to emulate the hardware).

CPU ARCHITECTURE – EMULATION VS. VIRTUALIZATION

- **Virtualization:**
  - **VMM or hypervisors have direct access to hardware devices**
    - **Advantages:**
      - The virtual machine hosted by the hypervisor can access hardware allowing performance increases.
    - **Disadvantages:**
      - VMM schedulers limit what can be run in virtual machines due to complexities such as context switching and VMM resource management.

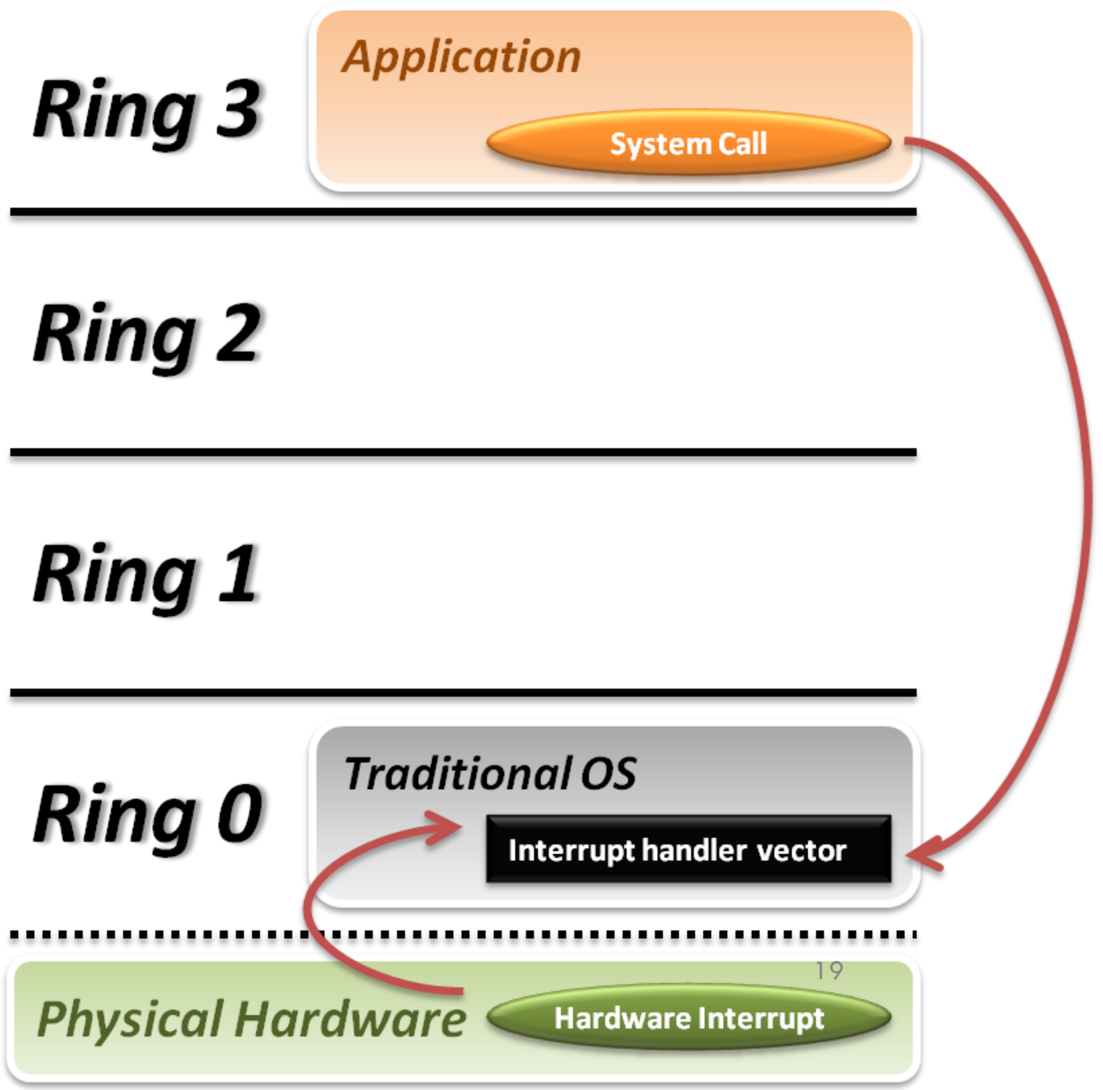
TRAP AND EMULATE MODEL

- **If we want CPU virtualization to be efficient, how should we implement the VMM ?**
  - We should make guest binaries run on CPU as fast as possible.
  - Theoretically speaking, if we can run all guest binaries natively, there will NO overhead at all.
  - But we cannot let guest OS handle everything, VMM should be able to control all hardware resources.

- **Solution :**
  - Run VMM in kernel mode.
    - Then VMM will be able to intercept all trapping events.

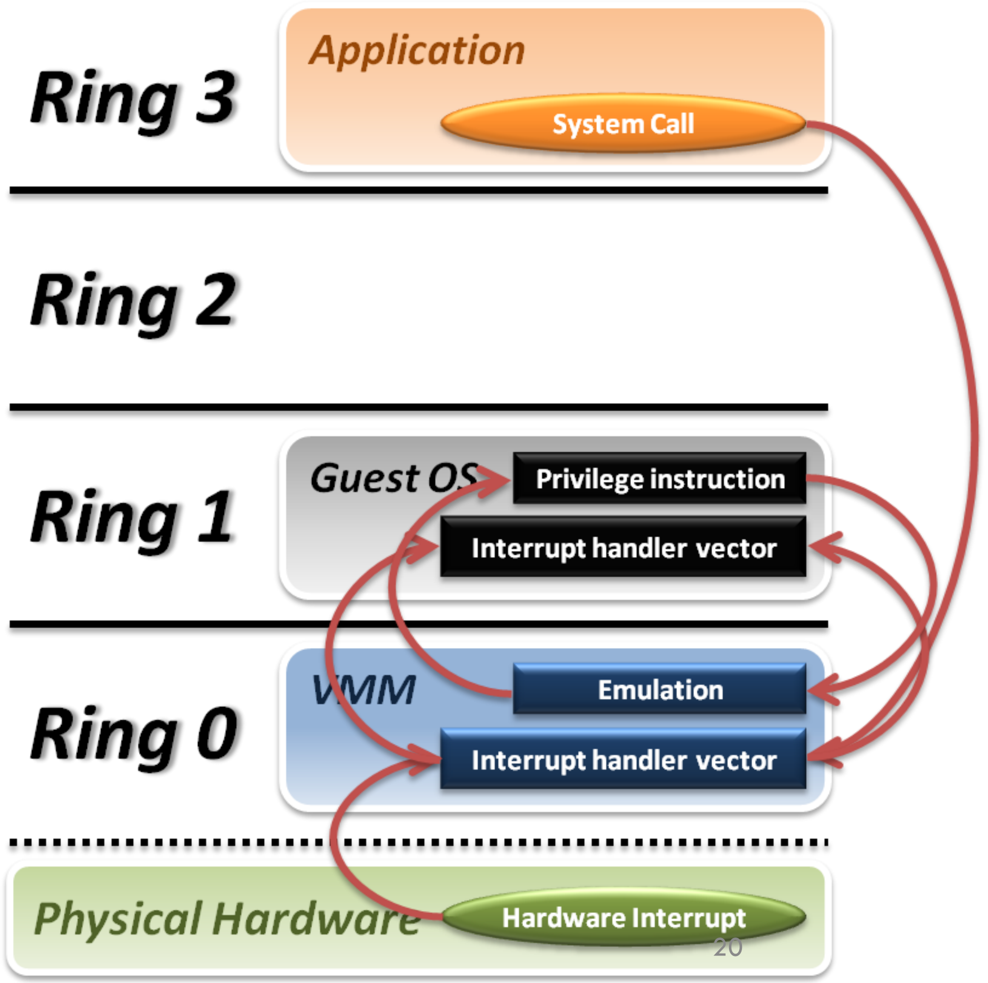
TRAP AND EMULATE MODEL

- **Traditional OS:**
  - **System call:**
    - CPU will trap to interrupt handler vector in OS.
    - CPU will switch to kernel mode (Ring 0) and execute OS instructions.
  - **Hardware event:**
    - Hardware will interrupt CPU execution, and jump to interrupt handler in OS.



TRAP AND EMULATE MODEL

- **VMM and Guest OS:**
  - **System Call**
    - CPU will trap to interrupt handler vector of VMM.
    - VMM jump back into guest OS.
  - **Hardware Interrupt**
    - Hardware make CPU trap to interrupt handler of VMM.
    - VMM jump to corresponding interrupt handler of guest OS.
  - **Privilege Instruction**
    - Running privilege instructions in guest OS will be trapped to VMM for instruction processing.
    - After processing the instruction, the VMM jumps back to guest OS.

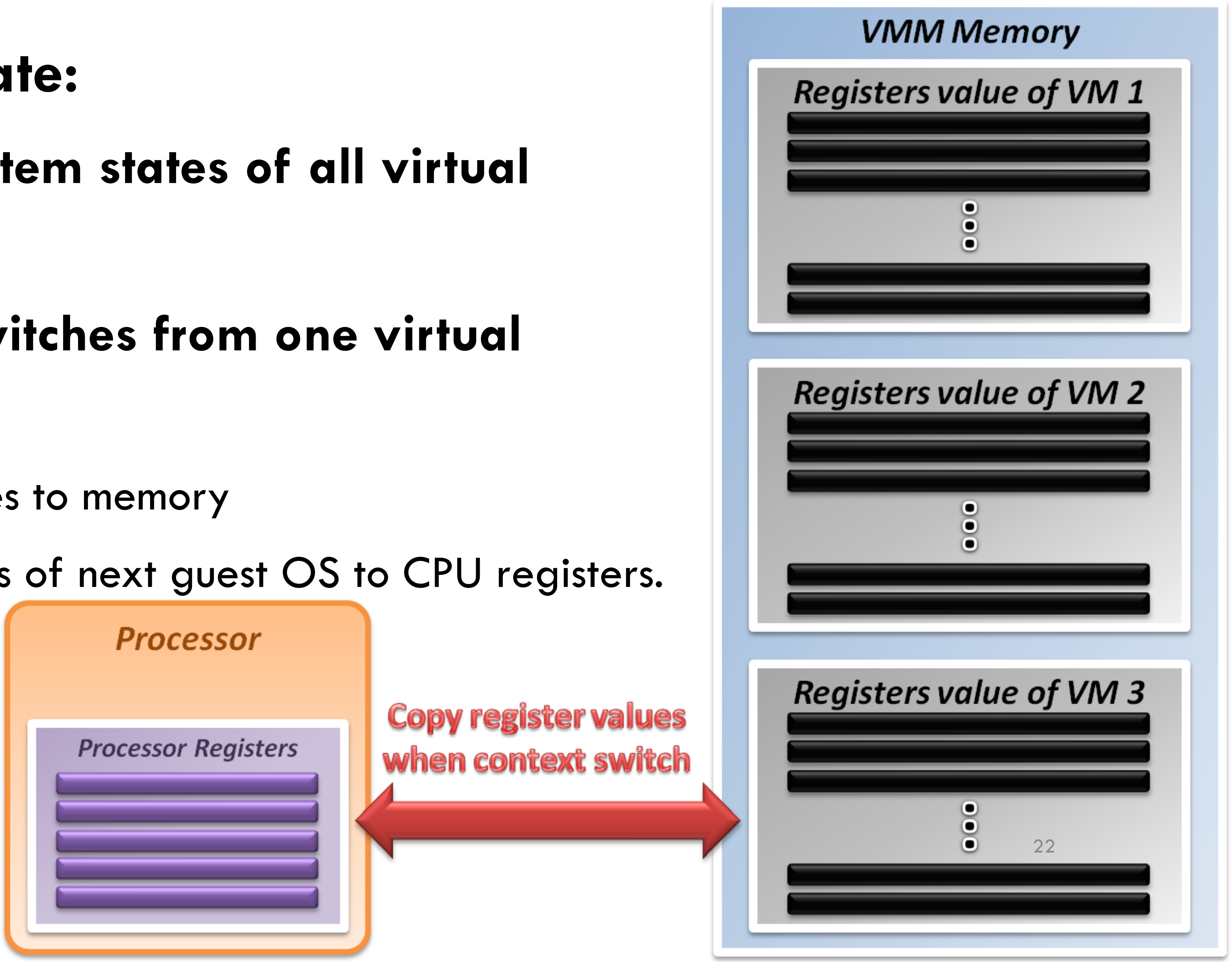


CONTEXT SWITCH

- **Steps for VMM to switch active virtual machines:**
  1. Timer Interrupt in running VM.
  2. Context switch to VMM.
  3. VMM saves state of running VM.
  4. VMM determines next VM to execute.
  5. VMM sets timer interrupt.
  6. VMM restores state of next VM.
  7. VMM sets PC to timer interrupt handler of next VM.
  8. Next VM active.

SYSTEM STATE MANAGEMENT

- **Virtualizing system state:**
  - **VMM will hold the system states of all virtual machines in memory.**
  - **When VMM context switches from one virtual machine to another:**
    - Write the register values to memory
    - Copy the register values of next guest OS to CPU registers.

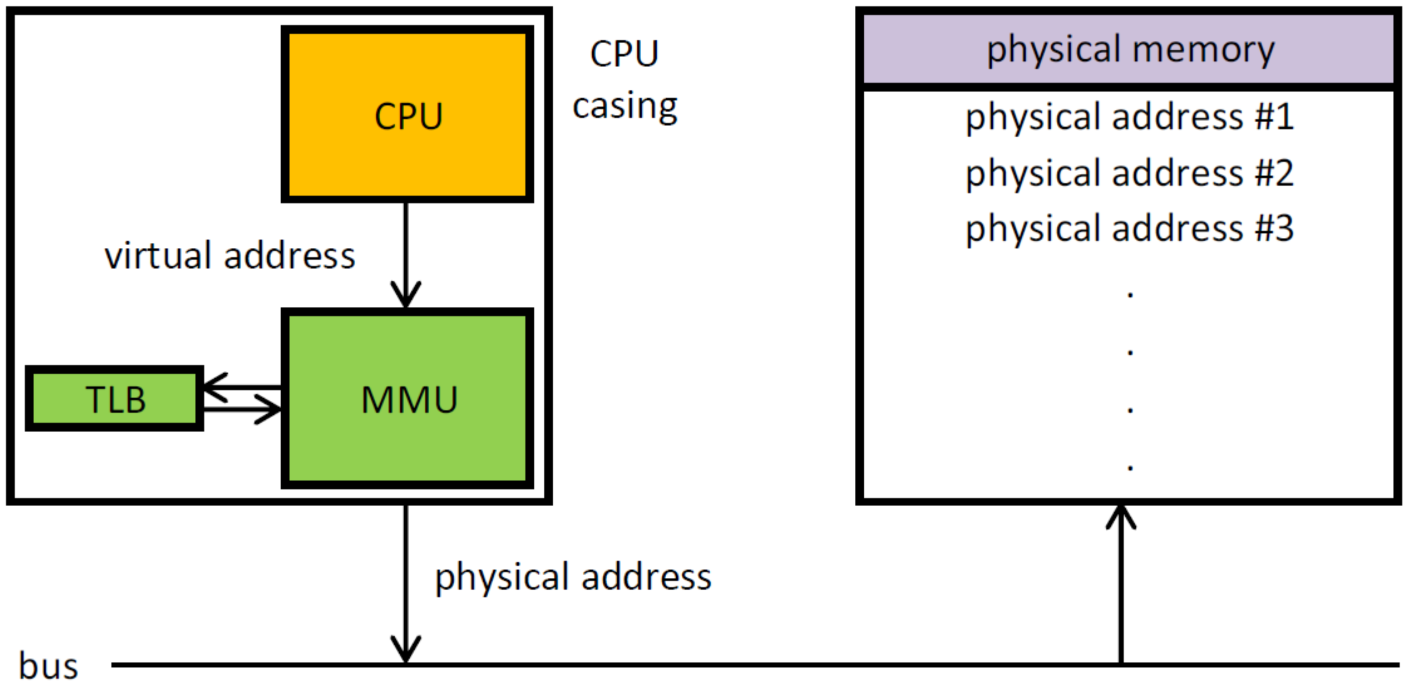


MEMORY VIRTUALIZATION

- **Memory management in OS**
  - **Traditionally, OS fully controls all physical memory space and provide a continuous addressing space to each process.**
  - **In server virtualization, VMM should make all virtual machines share the physical memory space without knowing the fact.**
- **Goals of memory virtualization :**
  - **Address Translation**
    - Control memory mapping techniques that accesses translation tables in main memory.
  - **Memory Protection**
    - Define specific memory access permissions for VMs.
  - **Access Attribute**
    - Define attribute and type of memory to be accessed.

MEMORY ARCHITECTURE

- **Memory Management Unit (MMU)**
  - A computer hardware component responsible for handling accesses to memory requested by the CPU.
  - Its functions include translation of virtual addresses to physical addresses, memory protection, cache control, etc.
  - Most modern CPUs include the MMU.



CPU: Central Processing Unit
MMU: Memory Management Unit
TLB: Translation lookaside buffer

MEMORY ARCHITECTURE - ADDRESS TRANSLATION

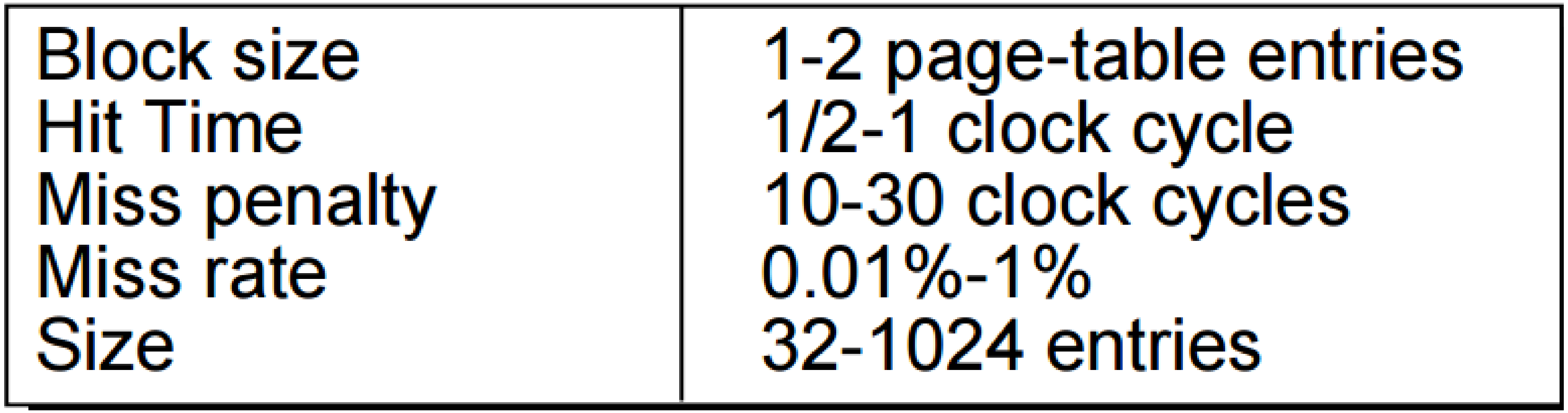
- **How to locate the physical address?**
  - **Search impractical (too many pages)**
- **A page table is a data structure which contains the mapping of virtual pages to physical addresses**
  - Usually located in fast CPU caches, special registers, or main memory
- **Each process running in the system has its own page table**
  - **How to access this data quickly?**

MEMORY ARCHITECTURE - TRANSLATION LOOKASIDE BUFFER

- **TLB is a small cache (on the CPU) that stores recent translations of virtual memory to physical addresses for faster retrieval.**
- **When a virtual memory address is referenced by a program, the search starts in the CPU.**
    - First, L1 and L2 caches are checked.
        - If the required data is not in these very fast caches, the system has to look up the data's physical address.
        - At this point, TLB is checked for a quick reference to the location in physical memory.

MEMORY ARCHITECTURE – TRANSLATION LOOKASIDE BUFFER

- **The TLB is a small cache of the most recent virtual-physical mappings.**



Block size	1-2 page-table entries
Hit Time	1/2-1 clock cycle
Miss penalty	10-30 clock cycles
Miss rate	0.01%-1%
Size	32-1024 entries

MEMORY ARCHITECTURE – VIRTUALIZATION TECHNIQUES

- **The performance drop of memory access in Guest OSs can be unbearable. VMM needs optimization techniques for data access.**

- **Accessing memory page tables:**
  - Translation Lookup Buffer not directly accessible by VM, only accessible through VMM.
  - Page tables in each VM must be accurately mapped to the memory systems of the host.
    - Overhead of memory subsystems and address translations can lead to dramatic performance decrease.

MEMORY ARCHITECTURE – VIRTUALIZATION TECHNIQUES

- **VMM Shadow Page Table:**
  - **Maps virtual pages used by VMs to actual pages regulated by VMM.**
  - **VMM maps page table addresses to addresses assigned by host system.**
    - Increases performance by decreasing translation and lookup time.
- **Deduplication:**
  - **Virtual pages containing the same content is shared among VMs**
- **Memory Balloon:**
  - **Module loaded into VMs as a pseudo device driver that communicates with the VMM.**
    - Inflates when memory usage percentage is high.
    - Deflates when memory usage percentage is low.

MEMORY ARCHITECTURE - DIRECT MEMORY ACCESS

- **DMA allows certain hardware subsystems within the computer to access system memory for reading and/or writing independently of the central processing unit.**
  - **DMA increases performance for some data fetch/decode operations in the VMs.**
- **Two types of DMA:**
  - **Synchronous DMA**
    - The DMA operation is caused by software.
    - For example, sound card driver may trigger DMA operation to play music.
  - **Asynchronous DMA**
    - The DMA operation is caused by devices (hardware).
    - For example, network card use DMA operation to load data into memory and interrupt CPU for further manipulation.

I/O VIRTUALIZATION

- **Goal :**
  - **Share or create I/O devices for virtual machines.**
- **Two types of I/O subsystem architecture:**
  - **Port Mapped I/O**
    - Uses CPU instructions specifically for performing I/O.
  - **Memory Mapped I/O (MMIO)**
    - Allows reading/writing to I/O devices in the same way as reading/writing to system memory.

PORT MAPPED I/O

- **I/O devices are mapped into a separate address space**
  - **I/O devices have a separate address space from general memory**
    - Accomplished by an extra I/O pin on the CPU's physical interface
    - Or bus dedicated to I/O.
  - **Generally found on Intel microprocessors**
- **Pros & Cons**
  - **Pros**
    - Less logic is needed to decode a discrete address.
    - Benefits for CPUs with limited addressing capability.
  - **Cons**
    - More instructions are required to accomplish the same task.
    - I/O addressing space size is not flexible.

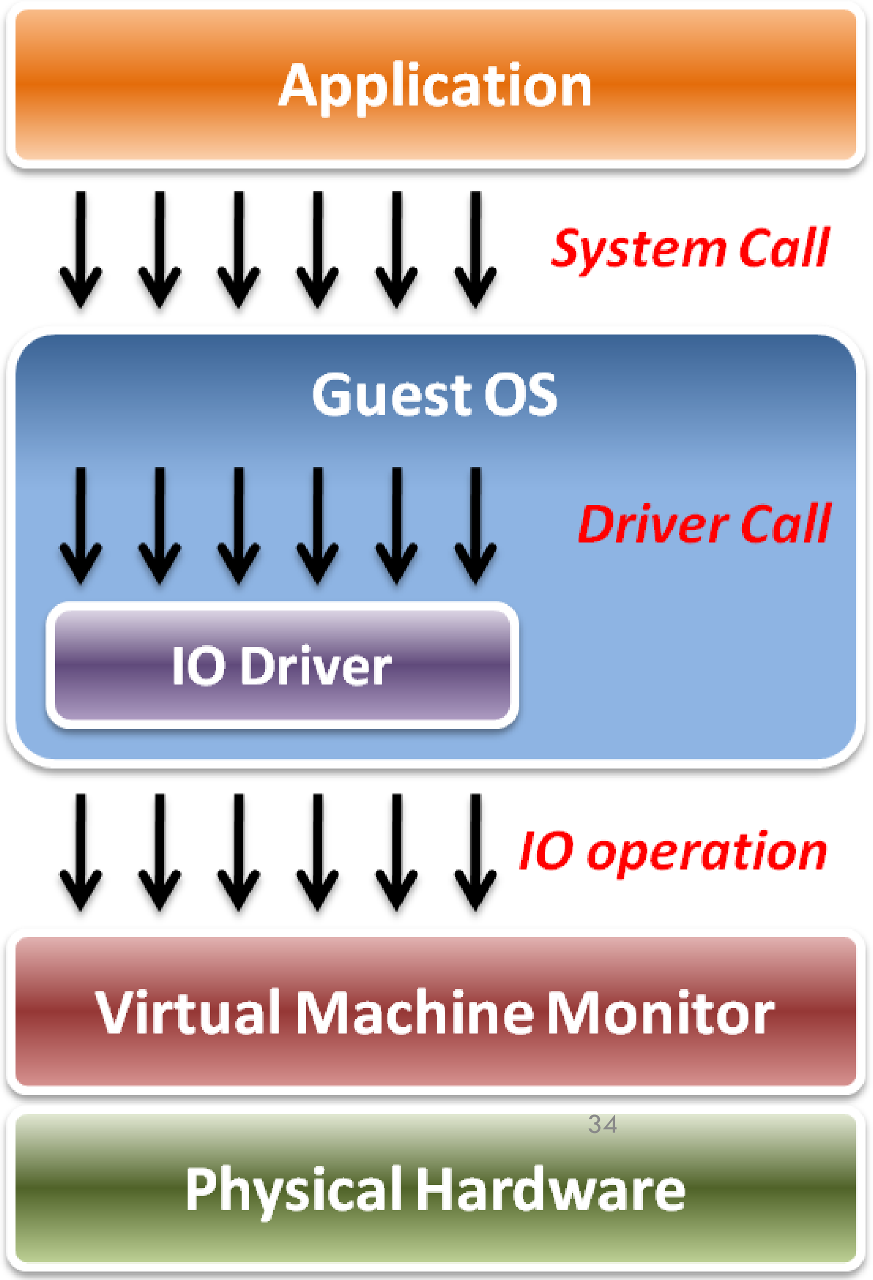
MEMORY MAPPED I/O

- **I/O devices are mapped into the system memory map along with RAM and ROM.**
  - **To access a hardware device, simply read or write to those 'special' addresses using the normal memory access instructions.**

- **Pros & Cons**
  - **Pros**
    - Instructions which can access memory can be used to operate an I/O device.
    - Control I/O devices with fewer instructions.
  - **Cons**
    - Physical memory addressing space must be shared with IO devices.
    - Generally not accessible by software applications as I/O communication is restricted to device drivers.

I/O VIRTUALIZATION

- **Implementation Layers:**
  - **System call**
    - The interface between applications and guest OS.
  - **Driver call**
    - The interface between guest OS and I/O device drivers.
  - **I/O operation request**
    - The interface between I/O device driver of guest OS and virtualized hardware (in VMM).



NETWORK VIRTUALIZATION

- The process of combining hardware and software network resources and network functionality into a single, software-based administrative entity

- Provides network-like functionality to the software containers on a single system.

NETWORK VIRTUALIZATION

- **Desirable properties of network virtualization:**
  - **Scalability**
    - Easy to extend resources in need such as bandwidth and IP/MAC addresses
    - Administrator can dynamically create or delete virtual network connections
  - **Resilience**
    - Recover from failure, failovers
    - Virtual network can automatically redirect packets by redundant links
  - **Security**
    - Increased path isolation and user segmentation
    - Virtual network should work with firewall software
  - **Availability**
    - Access network (internal and external) resources anytime

NETWORK VIRTUALIZATION

- **Network virtualization in different layers:**
  - **Layer 1 (Physical Layer)**
    - Hypervisor usually do not need to emulate the physical layer.
  - **Layer 2 (Link Layer)**
    - Implement virtual L2 network devices, such as switch, in hypervisor.
    - Example: Linux bridge.
  - **Layer 3 (Network Layer)**
    - Implement virtual L3 network devices, such as router, in hypervisor.
  - **Layer 4 or higher (Application and Transport Layers)**
    - Layer 4 or higher is usually implemented in guest OS.
    - Applications should determine how to communicate.

NETWORK VIRTUALIZATION

- **Network architecture:**
  - **Bridge (Virtual Switch)**
    - Make virtual machines on one node share physical NICs.
  - **DHCP**
    - Map virtual MAC addresses of VMs to private IPs in the LAN.
  - **NAT**
    - Forward the packages to public network (WAN).
  - **IP/MAC mapping table**
    - IP addresses are assigned by DHCP service.
    - MAC addresses are assigned by hypervisor.
    - This mapping table is maintained by the cloud middleware

TOPICS FOR THE PROJECT PROPOSAL PRESENTATION

- **Provide an overview of your project. (10 – 12 minutes, everyone must participate)**
  - Use photos, sketches, and other visual aids to quickly orient others to your project
- **Provide a thorough description of your project design.**
  - Problem description, related systems, background research
- **Provide high level details on how your project group will attempt to solve the problem.**
  - How will the group meet?
  - Smaller focus teams?
  - Organizational approaches
  - Meeting times
- **Sample presentation outline:**
  - Introduction
  - Project objectives
  - Background studies
  - Brief project design/approach
  - Expected end results