Module 1

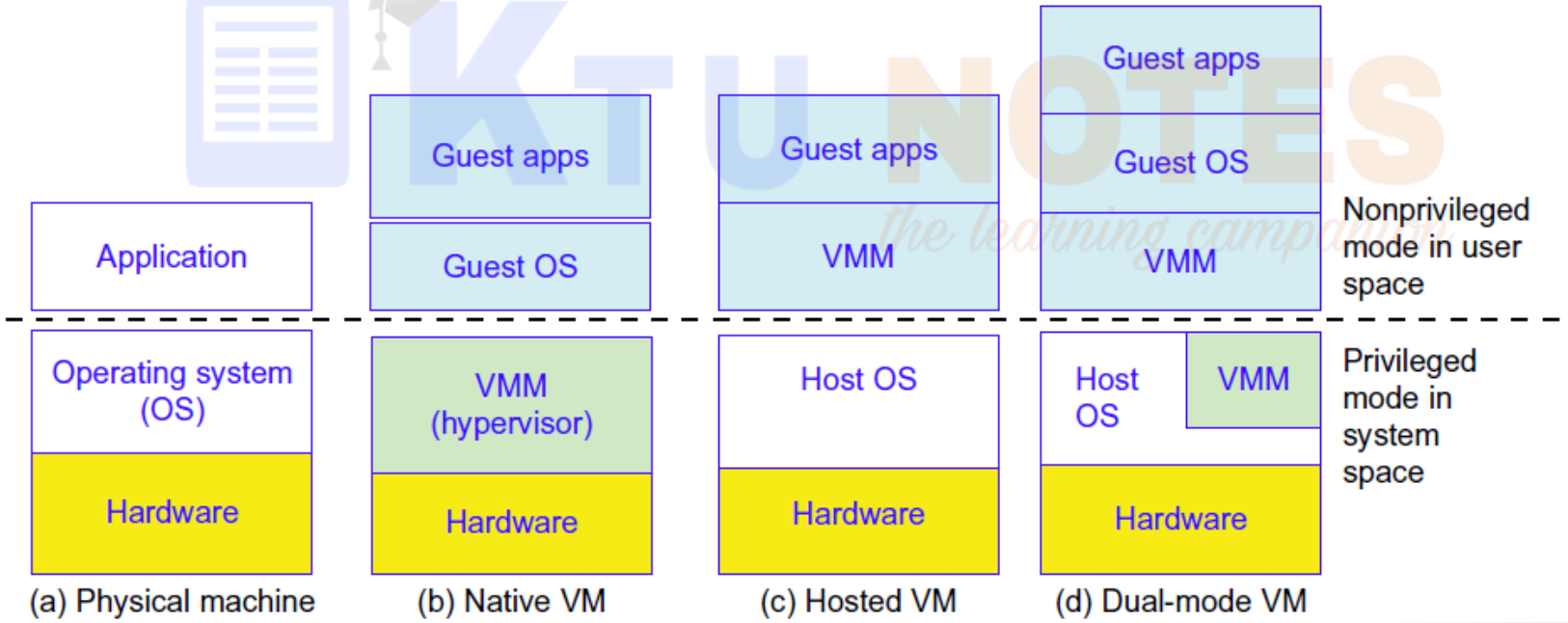
INTRODUCTION TO VIRTUALIZATION

Virtual Machines and Virtualization Middleware

- **Conventional computer:**
  - Single OS image
  - Rigid architecture that tightly couples application software to a specific hardware platform
- Software compatible with one machine may not be compatible with another platform with a different instruction set under a fixed OS.
- **Virtual machines (VMs)** offer novel solutions for:
  - Underutilized resources
  - Application inflexibility
  - Software manageability and
  - Security concerns in existing physical machines.

Virtual Machines

- **Comparison of the 3 VM architectures with traditional physical machine**



- **Physical Machine:**
  - Equipped with the physical hardware
  - Example: x-86 architecture desktop running its installed Windows OS

- **Virtual Machine:**
  - Can be provisioned for any hardware system.
  - Built with virtual resources managed by a guest OS to run a specific application.
  - Deploys a middleware layer called a virtual machine monitor (VMM) between the VMs and the host platform.

- **Native VM:**
  - Installed with the use of a VMM called a hypervisor in privileged mode.
  - Guest OS could be a Linux system and the hypervisor is the XEN system.
  - Also called bare-metal VM, because the hypervisor handles the bare hardware (CPU, memory, and I/O) directly.

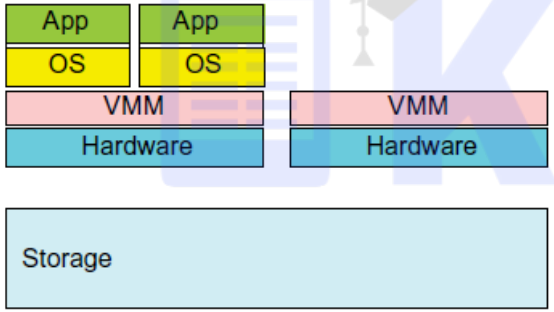
- **Hosted VM:**
  - VMM runs in non privileged mode.
  - The host OS need not be modified.

- **Dual mode VM:**
  - Part of the VMM runs at the user level and another part runs at the supervisor level.
  - The host OS may have to be modified to some extent.

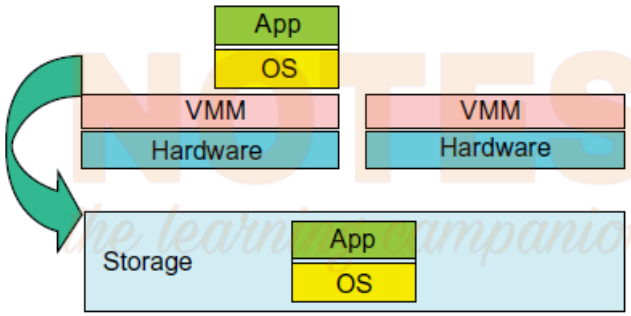
- **Advantages:**
  - Multiple VMs can be ported to a given hardware system to support the virtualization process.
  - Offers hardware independence of the OS and applications.
  - The user application running on its dedicated OS could be bundled together as a virtual appliance that can be ported to any hardware platform.
  - The VM could run on an OS different from that of the host computer.

- **VM Primitive Operations:**
  - VMM provides the VM abstraction to the guest OS.
  - With full virtualization, the VMM exports a VM abstraction identical to the physical machine.

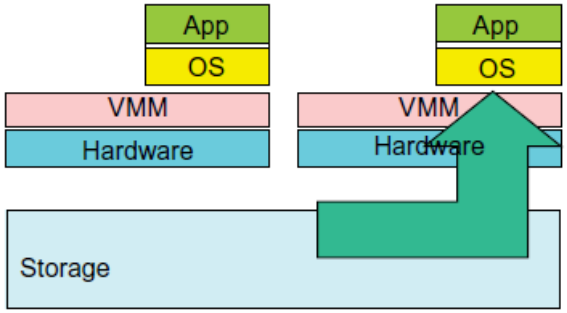
- **Low level VM Operations:**



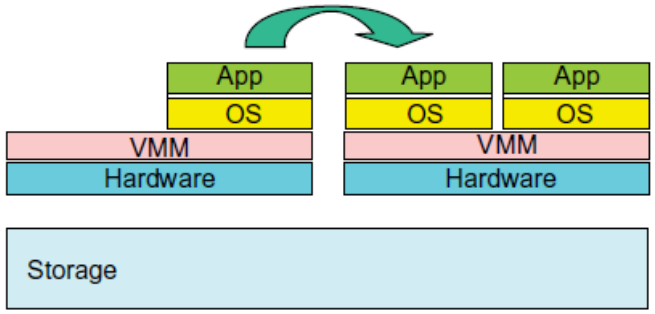
(a) Multiplexing



(b) Suspension (storage)



(c) Provision (resume)



(d) Life migration

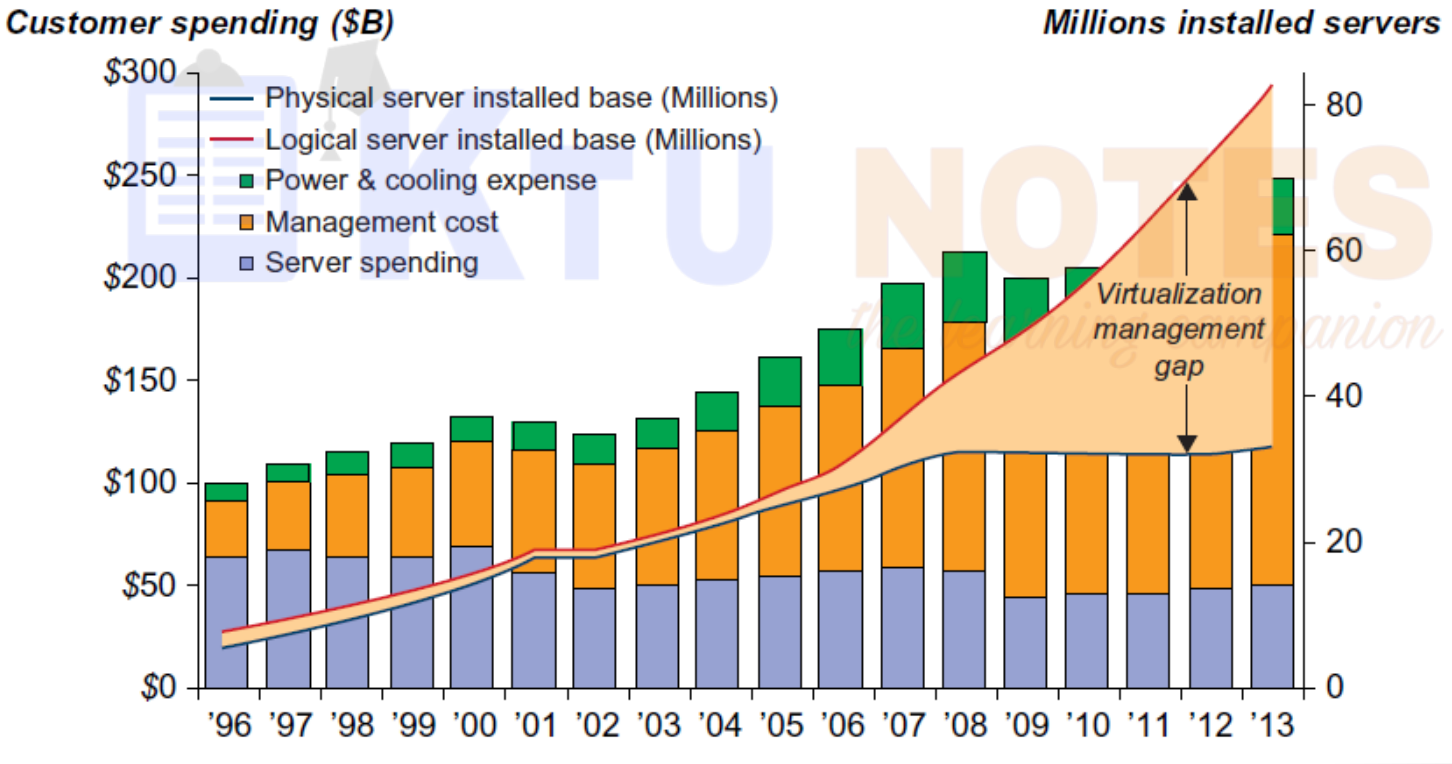
- These operations enable a VM to be provisioned to any available hardware platform.
- Enable flexibility in porting distributed application executions.
- Enhance the utilization of server resources.
- Multiple server functions can be consolidated on the same hardware platform to achieve higher system efficiency.
- This will eliminate server sprawl via deployment of systems as VMs, which move transparency to the shared hardware.
- With this approach, VMware claimed that server utilization could be increased from its current 5–15 percent to 60–80 percent.

Data Center Virtualization for Cloud Computing

- Cloud architecture is built with commodity hardware and network devices.
- Almost all cloud platforms choose the popular x86 processors.
- Low-cost terabyte disks and Gigabit Ethernet are used to build data centers.
- Data center design emphasizes the performance/price ratio over speed performance alone.
- Storage and energy efficiency are more important than shear speed performance.

- **Data Center Growth and Cost Breakdown:**

- Cost to build and maintain data center servers has increased over the years.



- 30 percent → purchasing IT equipment (such as servers and disks)
- 33 percent → chiller
- 18 percent → uninterruptible power supply (UPS)
- 9 percent → computer room air conditioning (CRAC)
- 7 percent → power distribution, lighting, and transformer costs.
- Total 60 percent of the cost to run a data center → management and maintenance.
- The server purchase cost → not much increase with time.
- The cost of electricity and cooling → increased from 5 percent to 14 percent in 15 years.

- **Low-Cost Design Philosophy:**

- High-end switches or routers :- Not cost effective for building data centers.
- High-bandwidth networks :- not fit for the economics of cloud computing.
- Commodity switches and networks are more suitable in data centers because of the fixed budget.
- And commodity x86 servers is more desired over expensive mainframes.
- The software layer handles network traffic balancing, fault tolerance, and expandability.
- Ethernet is the fundamental network technology for most cloud computing data centers.

- **Convergence of Technologies:**

- Cloud computing is enabled by the convergence of technologies in four areas:

  - Hardware virtualization and multi-core chips - enable the existence of dynamic configurations in the cloud

  - utility and grid computing - lay the necessary foundation for computing clouds

  - SOA, Web 2.0, and WS mashups - pushing the cloud another step forward

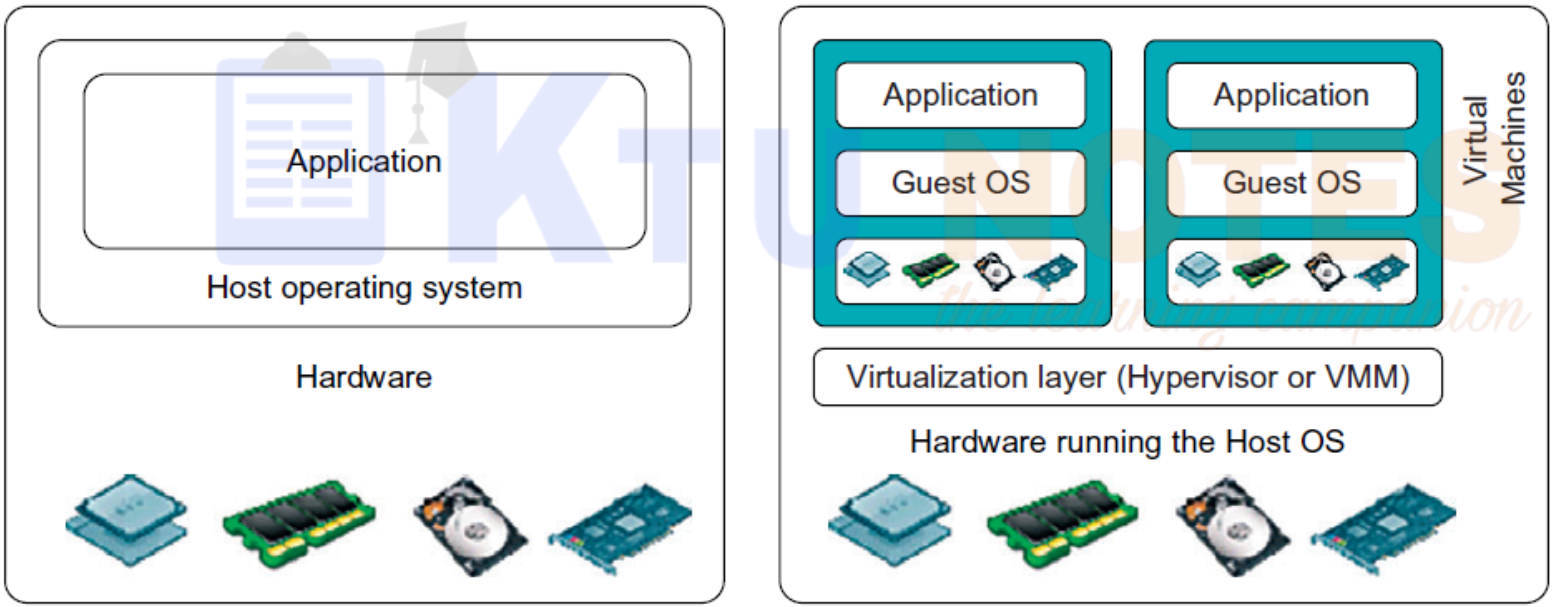
  - autonomic computing and data center automation - contribute to the rise of cloud computing

Implementation levels of Virtualization

- **Virtualization:**
  - Computer architecture technology by which multiple VMs can be multiplexed in the same hardware machine.
  - Improves computer performance in terms of resource utilization and application flexibility.
  - Hardware or software resources can be virtualized in various functional layers.
  - Eg. – virtual memory
  - Applied to enhance the use of compute engines, networks, and storage.
  - 2009 Gartner Report → top strategic technology poised to change the computer industry.

- **Levels of Virtualization Implementation:**

- Architecture of a computer system before and after virtualization:



(a) Traditional computer

(b) After virtualization

- **After virtualization:**

- Different user applications using different operating systems (guest OS) can run on the same hardware, independent of the host OS.

- An additional software called a virtualization layer is added to achieve this.

- Virtualization layer →hypervisor or virtual machine monitor (VMM).

- Main function → virtualize the physical hardware of a host machine into virtual resources to be used exclusively by the VMs.

- Creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system.

- **Common virtualization layers include:**

- instruction set architecture (ISA) level
- hardware level
- operating system level
- library support level
- application level

- **Instruction Set Architecture Level**

- Virtualization is performed by emulating a given ISA by the ISA of the host machine.

- Example - MIPS binary code can run on an x86-based host machine.

- Legacy binary code written for various processors can be run on any given new hardware host machine.

- Basic emulation method → code interpretation (one by one)

- One source instruction may be translated to tens or hundreds of native target instructions.

- Dynamic binary translation can be used for better performance (basic blocks of dynamic source instructions to target instructions).

- **Hardware Abstraction Level**

- Performed right on top of the bare hardware, generates a virtual hardware environment for a VM.

- Also manages the underlying hardware through virtualization.

- A computer's resources →processors, memory, and I/O devices are virtualized.

- Upgrade the hardware utilization rate by having multiple users concurrently.

- First implemented in the IBM VM/370 (1960s).

- Xen hypervisor → virtualize x86-based machines to run Linux or other guest OS applications.

- **Operating System Level**

- Abstraction layer between traditional OS and user applications.
- OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers.
- The containers behave like real servers.
- Commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.
- Also in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.

- **Library Support Level**

- Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS.

- Since most systems provide well-documented APIs, such an interface becomes another candidate for virtualization.

- Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks.

- The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts.

- Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration

- **User Application Level**

- Virtualizes an application as a VM. Also known as process-level virtualization.

- Most popular approach is to deploy high level language (HLL) VMs.

- The virtualization layer sits as an application program on top of the OS, and exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition.

- Any program written in the HLL and compiled for this VM will be able to run on it.

- The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM.

- **Merits of Virtualization at Various Levels**

Level of Implementation	Higher Performance	Application Flexibility	Implementation Complexity	Application Isolation
ISA	X	XXXXX	XXX	XXX
Hardware-level virtualization	XXXXX	XXX	XXXXX	XXXX
OS-level virtualization	XXXXX	XX	XXX	XX
Runtime library support	XXX	XX	XX	XX
User application level	XX	XX	XXXXX	XXXXX

- **VMM Design Requirements and Providers**

- Three requirements:

  - VMM should provide an environment for programs which is essentially identical to the original machine.

  - Programs running in this environment should show, at worst, only minor decreases in speed.

  - A VMM should be in complete control of the system resources.

- Complete control of the resources by a VMM includes the following aspects:

  - The VMM is responsible for allocating hardware resources for programs

  - It is not possible for a program to access any resource not explicitly allocated to it

  - It is possible under certain circumstances for a VMM to regain control of resources already allocated.

- **Virtualization Support at the OS Level:**

  - Cloud computing → emerging computing mode with the help of VM technology

  - Challenges faced by cloud computing: -

    - Ability to use a variable number of physical machines and VM instances depending on the needs of a problem. That is the usage should be based on the requirement.

    - The time required for instantiating new VMs – slow operation because each VM creates its own image from scratch.

    - VM images needs to be stored and there is a lot of repeated content among them.

    - Full virtualization at the hardware level also has the disadvantages of slow performance and low density.

    - Sometimes hardware modification is needed to reduce the performance overhead of hardware-level virtualization.

- **Importance of OS Level Virtualization:**

  - OS-level virtualization provides a feasible solution for these hardware-level virtualization issues.
  - OS virtualization inserts a virtualization layer inside an operating system to partition a machine's physical resources.
  - Allows multiple isolated VMs within a single operating system kernel.
  - Often called a virtual execution environment (VE), Virtual Private System (VPS), or simply container.
  - VEs →real servers from the user point of view.
  - VEs → share the same OS kernel and hence called as single-OS image virtualization

- **Advantages:**
- OS level VMs have minimal startup/shutdown costs, low resource requirements, and high scalability.

- It is possible for an OS level VM and its host environment to synchronize state changes when necessary.

- Achieved via two mechanisms of OS-level virtualization:

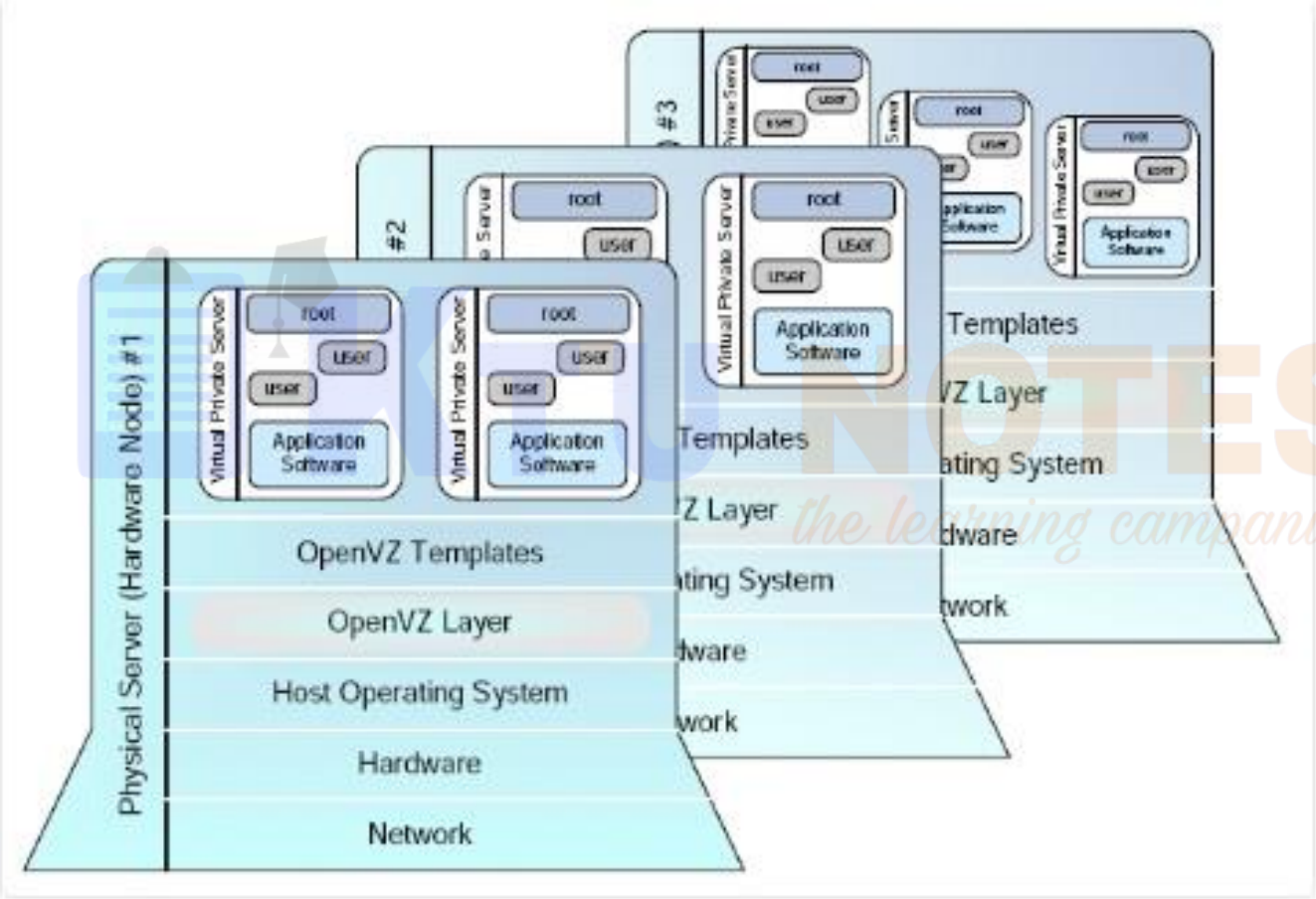
- (1) All OS-level VMs on the same physical machine share a single operating system kernel.

- (2) The virtualization layer can be designed in a way that allows processes in VMs to access as many resources of the host machine as possible, but never to modify them.

- **Disadvantages:**
- All the VMs at operating system level on a single container must have the same kind of guest operating system.

- Different OS-level VMs may have different operating system distributions, they must pertain to the same operating system family.

- **OS virtualization from the point of view of a machine stack:**



- The virtualization layer is inserted inside the OS to partition the hardware resources for multiple VMs to run their applications in multiple virtual environments.
- To implement OS-level virtualization, isolated execution environments (VMs) should be created based on a single OS kernel.
- The chroot command in a UNIX system can create several virtual root directories within a host OS.
- These virtual root directories are the root directories of all VMs created.
- 2 ways to implement virtual root directories: duplicating common resources to each VM partition; or sharing most resources with the host environment and only creating private resource copies on the VM on demand.
- The first way incurs significant resource costs and overhead on a physical machine.
- This issue neutralizes the benefits of OS-level virtualization, compared with hardware-assisted virtualization.
- Hence, OS-level virtualization is often a second choice.

- **Virtualization Support for Linux and Windows NT Platforms**

Virtualization Support	Functionality and Application Platforms
Linux vServer for Linux platforms	Extends Linux kernels to implement a security mechanism to help build VMs by setting resource limits and file attributes and changing the root environment for VM isolation
OpenVZ for Linux platforms	Supports virtualization by creating virtual private servers (VPSes); the VPS has its own files, users, process tree, and virtual devices, which can be isolated from other VPSes, and checkpointing and live migration are supported
FVM(Feather-Weight Virtual Machines) for virtualizing the Windows NT platforms	Uses system call interfaces to create VMs at the NT kernel space; multiple VMs are supported by virtualized namespace and copy-on-write

Virtualization Structures/Tools and Mechanisms

- Before virtualization → OS manages the hardware.
- After virtualization → a virtualization layer is inserted between the hardware and the OS.
- The virtualization layer converts portions of the real hardware into virtual hardware.
- Thus different operating systems such as Linux and Windows can run on the same physical machine, simultaneously.
- Depending on the position of the virtualization layer, there are several classes of VM architectures
  - the hypervisor architecture
  - paravirtualization
  - host-based virtualization.

- **Hypervisor**

- Hardware virtualization technique allowing multiple OS called guests to run on a host machine.
- Also called the Virtual Machine Monitor (VMM).
- Supports hardware-level virtualization on bare metal devices like CPU, memory, disk and network interfaces
- Sits directly between the physical hardware and its OS.
- Provides hypercalls for the guest OSes and applications.

- Assumes a micro-kernel architecture like the Microsoft Hyper-V.
  - Includes only the basic and unchanging functions (such as physical memory management and processor scheduling).
  - The device drivers and other changeable components are outside the hypervisor
- Or it can assume a monolithic hypervisor architecture like the VMware ESX for server virtualization.
  - implements all the above functions, including those of the device drivers.
- So the size of the hypervisor code of a micro-kernel hypervisor is smaller than that of a monolithic hypervisor.
- A hypervisor must be able to convert physical devices into virtual resources dedicated for the deployed VM to use.

- **Xen Architecture**

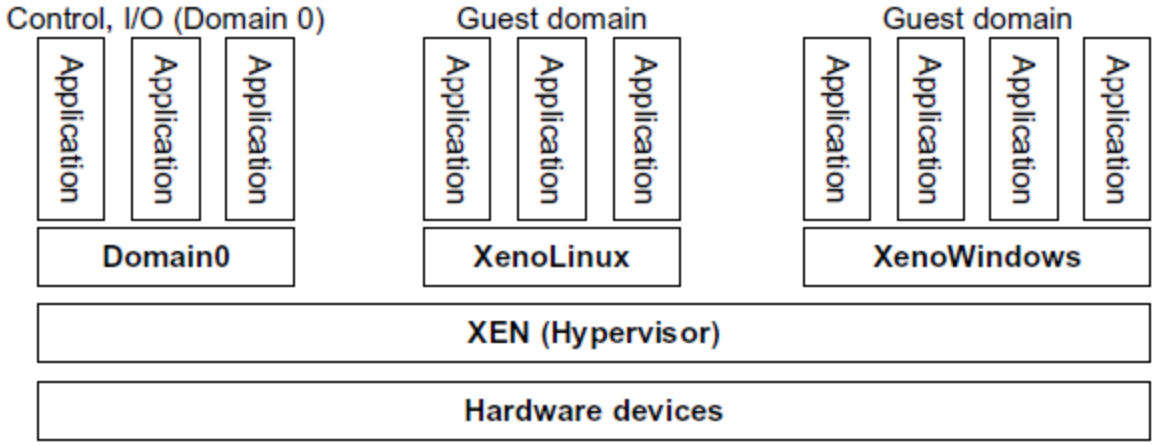
- Microkernel hypervisor →does not include any device drivers natively.

- Provides a mechanism by which a guest OS can have direct access to the physical devices.

- So the size of the Xen hypervisor is kept rather small.

- Xen provides a virtual environment located between the hardware and the OS.

- Core components of a Xen system are:
  - hypervisor
  - kernel
  - applications



- Not all guest OSes are created equal → one controls the others → Domain 0, others are called Domain U.
- Domain 0 is a privileged guest OS of Xen → first loaded when Xen boots without any file system drivers being available.
- Domain 0 is designed to access hardware directly and manage devices
- Therefore, one of the responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains (the Domain U domains)
- If Domain 0 is compromised, the hacker can control the entire system. So, in the VM system, security policies are needed to improve the security of Domain 0

- **Binary Translation with Full Virtualization**

- Depending on implementation technologies, hardware virtualization can be classified into two categories:
  - full virtualization
  - host-based virtualization.

- Full virtualization → no need to modify the host OS.

- Relies on binary translation to trap and to virtualize the execution of certain sensitive, non-virtualizable instructions.

- Host-based system → both a host OS and a guest OS are used.

- A virtualization software layer is built between the host OS and guest OS.

- **Full Virtualization**

- With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software.

- Binary translation can incur a large performance overhead.

- Noncritical instructions do not control hardware or threaten the security of the system, but critical instructions do.

- Therefore, running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security.

- **Binary Translation of Guest OS Requests Using a VMM**

- VMware: puts the VMM at Ring 0 and the guest OS at Ring 1.

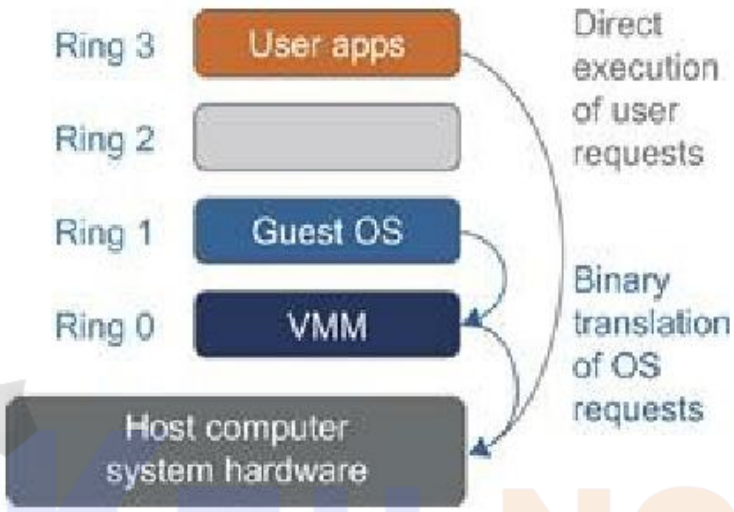
- The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions.

- Once identified →trapped into the VMM → emulates the behavior of these instructions.

- The method used in this emulation is called binary translation.

- Full virtualization combines binary translation and direct execution.

- The guest OS is completely decoupled from the underlying hardware and so is unaware that it is being virtualized.



- The performance of full virtualization may not be ideal, because it involves binary translation which is rather time-consuming.
- In particular, the full virtualization of I/O-intensive applications is a really a big challenge.
- Binary translation employs a code cache to store translated hot instructions to improve performance, but it increases the cost of memory usage.

- **Host Based Virtualization**

- Alternative VM architecture → install a virtualization layer on top of the host OS.
- This host OS is still responsible for managing the hardware.
- The guest OSes are installed and run on top of the virtualization layer.
- Dedicated applications may run on the VMs.
- Certainly, some other applications can also run with the host OS directly

- Advantages:
  - the user can install this VM architecture without modifying the host OS.
  - OS installs drivers and other low-level services → simplify the VM design and ease its deployment.
  - host-based approach appeals to many host machine configurations.

- Compared to the hypervisor/VMM architecture, the performance of the host-based architecture may also be low.

- When an application requests hardware access, it involves four layers of mapping which downgrades performance significantly.

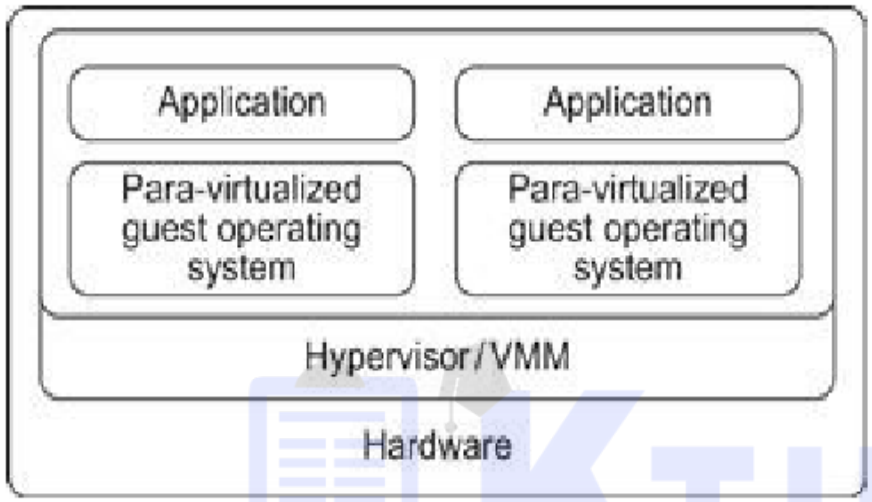
- When the ISA of a guest OS is different from the ISA of the underlying hardware, binary translation must be adopted.

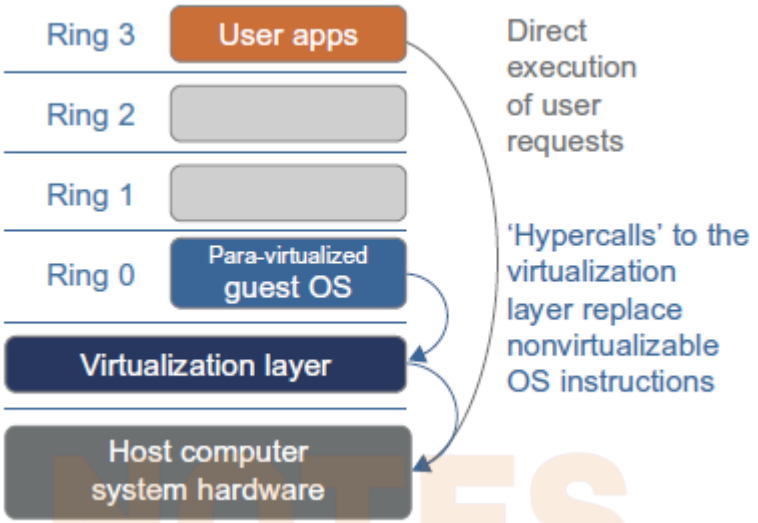
- Although the host-based architecture has flexibility, the performance is too low to be useful in practice.

- **Para-Virtualization with Compiler Support**

- Para-virtualization needs to modify the guest operating systems.
- A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications.
- Performance degradation is a critical issue of a virtualized system.
- Para-virtualization attempts to reduce the virtualization overhead - performance is improved by modifying only the guest OS kernel.
- The guest operating systems are para-virtualized.
- And are assisted by an intelligent compiler to replace the nonvirtualizable OS instructions by hypercalls.

- **Para-Virtualized VM Architecture**





- The traditional x86 processor offers four instruction execution rings: Rings 0, 1, 2, and 3.

- The lower the ring number, the higher the privilege of instruction being executed.

- The OS is responsible for managing the hardware and the privileged instructions to execute at Ring 0, while user-level applications run at Ring 3.
- When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS.
- According to the x86 ring definition, the virtualization layer should also be installed at Ring 0.
- Different instructions at Ring 0 may cause some problems.
- Para-virtualization replaces nonvirtualizable instructions with hypercalls that communicate directly with the hypervisor or VMM.

- **Advantages:**
  - Reduces overhead
  - Compared with full virtualization, para-virtualization is relatively easy and more practical.
  - Full virtualization - low performance in binary translation. It is not easy to speed up binary translation.
  - Hence many virtualization products employ the para-virtualization architecture
  - Eg - Xen, KVM, and VMware ESX
- **Disadvantages:**
  - Compatibility and portability - it must support the unmodified OS as well.
  - The cost of maintaining para-virtualized OSes is high, because they may require deep OS kernel modifications.
  - The performance advantage of para-virtualization varies greatly due to workload variations.

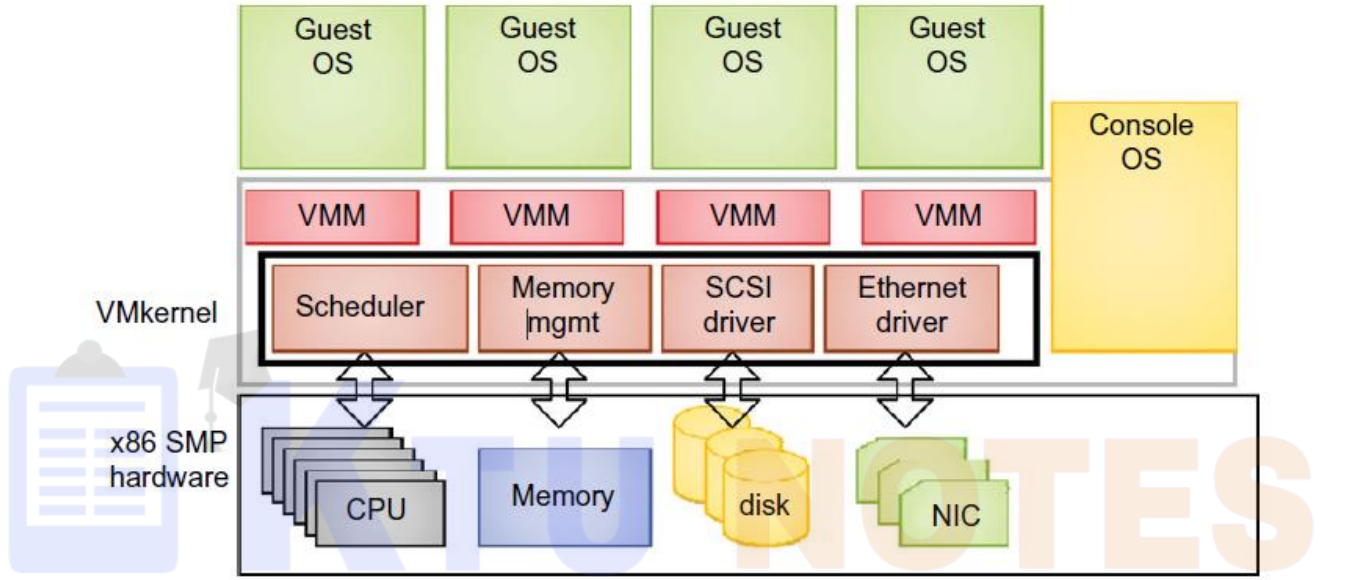
- **VMware ESX Server for Para-Virtualization**

  - ESX is a VMM or a hypervisor for bare-metal x86 symmetric multiprocessing (SMP) servers.

  - It accesses hardware resources such as I/O directly and has complete resource management control.

  - An ESX-enabled server consists of four components:
    - a virtualization layer, a resource manager, hardware interface components, and a service console.

  - To improve performance, the ESX server employs a para-virtualization architecture in which the VM kernel interacts directly with the hardware without involving the host OS.



The VMware ESX server architecture using para-virtualization.

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

- To support virtualization, processors such as the x86 employ a special running mode and instructions, known as hardware-assisted virtualization.
- So the VMM and guest OS run in different modes and all sensitive instructions of the guest OS and its applications are trapped in the VMM.
- **Hardware Support for Virtualization:**
- All processors have at least two modes, **user mode and supervisor mode**, to ensure controlled access of critical hardware.
- Instructions running in supervisor mode are called **privileged instructions**. Other instructions are **unprivileged instructions**.
- Eg :- Vmware Workstation is a VM software suite for x86 and x86-64 computers

- **CPU Virtualization**

- Unprivileged instructions of VMs run directly on the host machine for higher efficiency.

- Other critical instructions should be handled carefully for correctness and stability.

- The critical instructions are divided into three categories: **privileged instructions, control sensitive instructions, and behavior-sensitive instructions.**

- Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode.

- Control-sensitive instructions attempt to change the configuration of resources used.

- Behavior-sensitive have different behaviors depending on the configuration of resources

- A CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode.

- When the privileged instructions of a VM are executed, they are trapped in the VMM.

- The VMM acts as a unified mediator for hardware access from different VMs to guarantee the correctness and stability of the whole system.

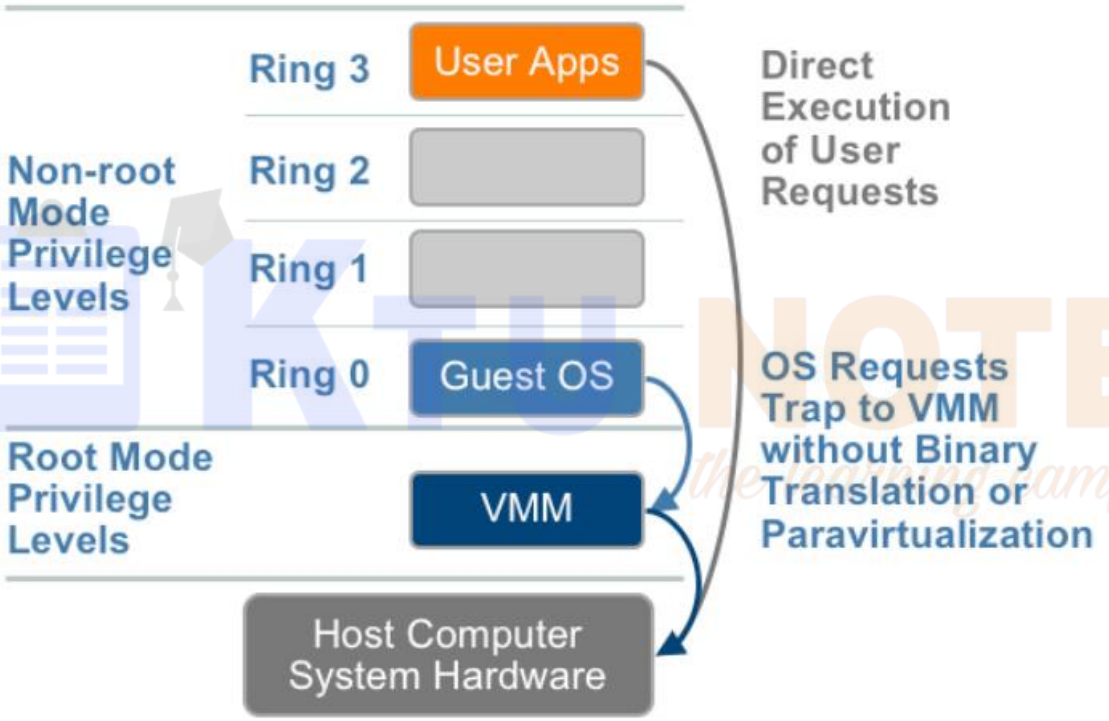
- Not all CPU architectures are virtualizable. RISC CPU architectures can be naturally virtualized because all control- and behavior-sensitive instructions are privileged instructions.

- x86 CPU architectures are not primarily designed to support virtualization.

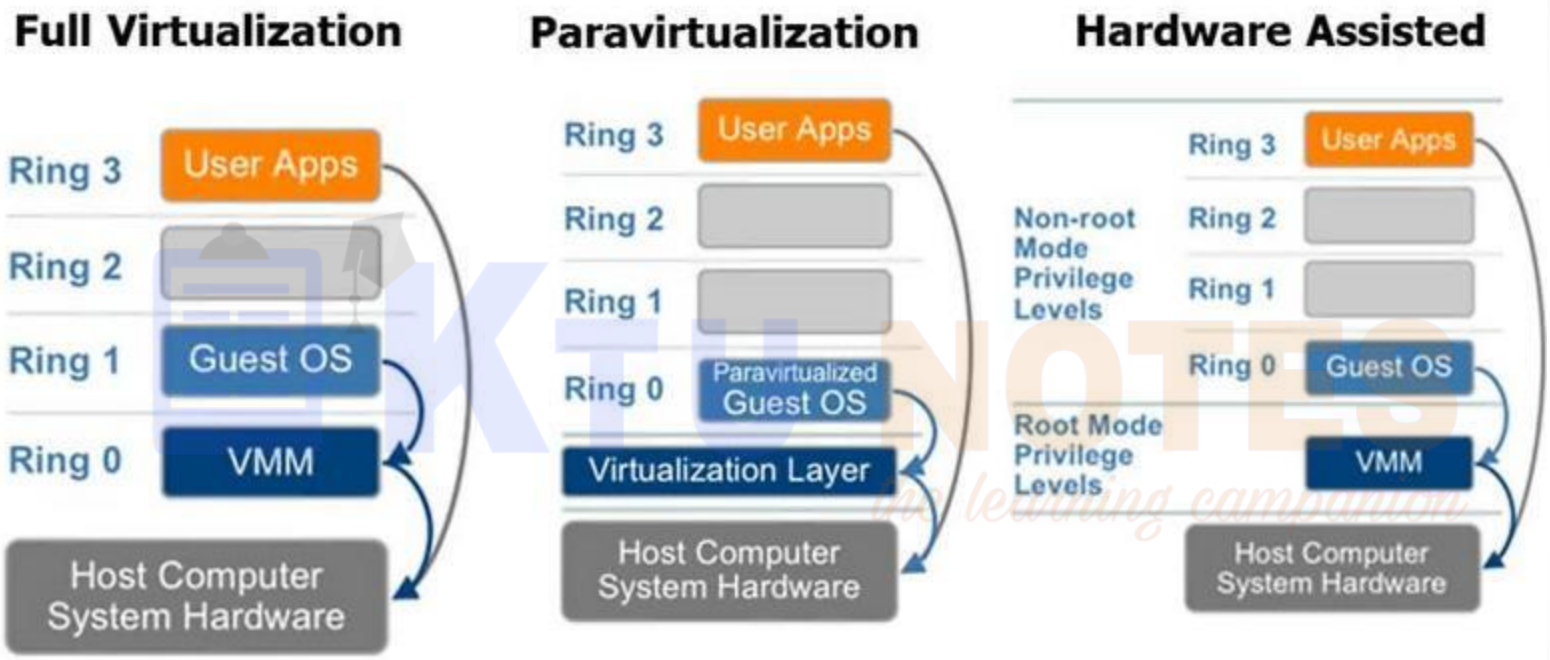
- This is because about 10 sensitive instructions, are not privileged instructions. When these instructions execute in virtualization, they cannot be trapped in the VMM.

- **Hardware-Assisted CPU Virtualization.**
- Intel and AMD add an additional mode called privilege mode level (Ring-1) to x86 processors.
- Guest operating systems can still run at Ring 0 and the hypervisor can run at Ring -1.
- All the privileged and sensitive instructions are trapped in the hypervisor automatically.
- This technique removes the difficulty of implementing binary translation of full virtualization.
- It also lets the operating system run in VMs without modification.
- Ie overcomes the complication of both full and para-virtualization

- **Hardware-Assisted CPU Virtualization.**



Architectural Comparison



- **Memory Virtualization**

- Similar to the virtual memory support provided by modern OSs.

- Traditional execution environment:
  - the OS maintains mappings of virtual memory to machine memory using page tables

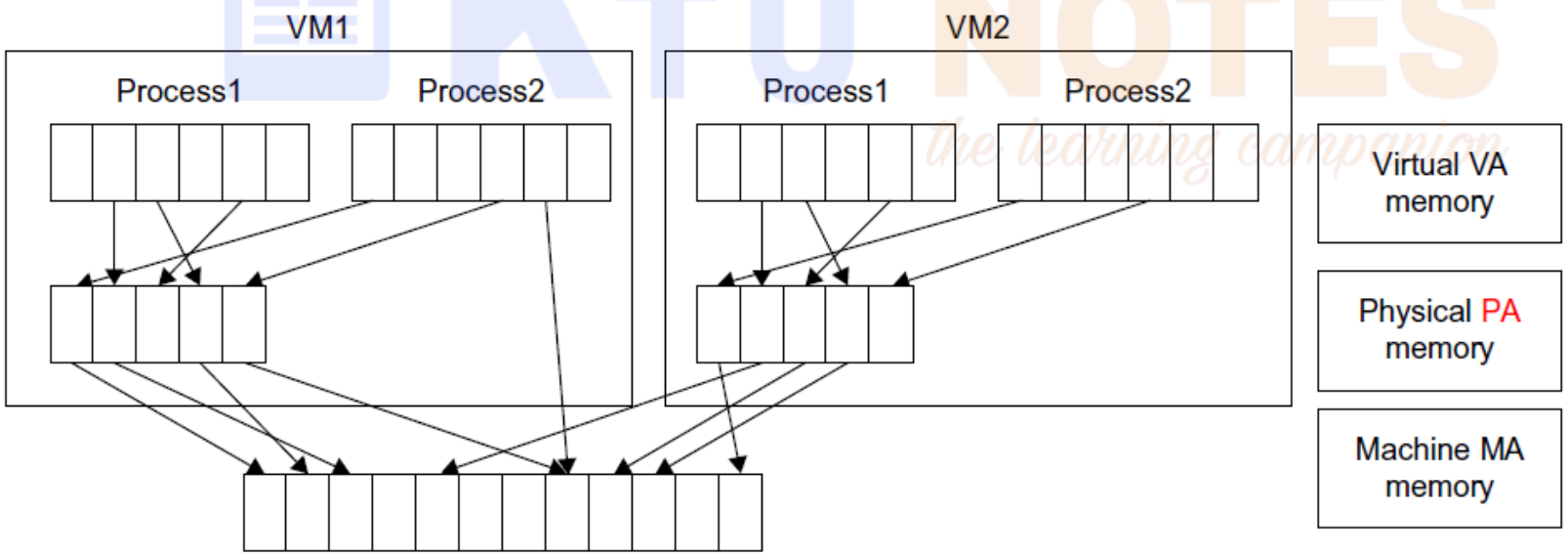
  - One-stage mapping from virtual memory to machine memory

  - All modern x86 CPUs include a MMU and a TLB to optimize virtual memory performance.

- Virtual execution environment:
  - Virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs.

  - Two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory.

- Mapping of virtual addresses to the physical memory addresses of VMs → Guest OS.
- Guest OS cannot directly access the actual machine memory.
- Guest physical memory to the actual machine memory → VMM



- Each page table of the guest OSes has a separate page table in the VMM corresponding to it, called the shadow page table.
- The MMU → handles virtual-to-physical translations as defined by the OS.
- Physical memory addresses are translated to machine addresses using another set of page tables defined by the hypervisor.
- Modern OSes maintain a set of page tables for every process, the shadow page tables will get flooded. → The performance overhead and cost of memory will be very high.
- **VMware** uses shadow page tables to perform **virtual-memory-to-machine-memory address translation**
- Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access.
- When the guest OS changes the virtual memory to a physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup.

- **I/O Virtualization**

- Involves managing the routing of I/O requests between virtual devices and the shared physical hardware.

- Three ways to implement I/O virtualization:
  - **full device emulation**
  - **para-virtualization**
  - **direct I/O**

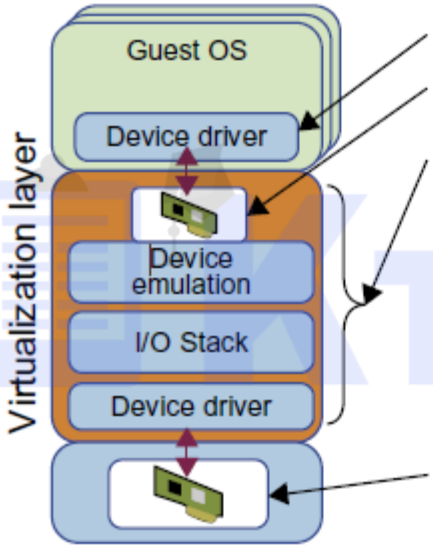
- **Full Device Emulation:**

- All the functions of a device or bus infrastructure are replicated in software.

- This software is located in the VMM and acts as a virtual device.

- The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices

- **Full device emulation:**



- Guest device driver
- Virtual device
- Virtualization layer
  - emulates the virtual device
  - remaps guest and real I/O addresses
  - multiplexes and drives the physical device
  - I/O features. e.g., COW disks
- Real device
  - may be different from virtual device

- **Para Virtualization method of I/O Virtualization:**

- Typically used in Xen.

- Also known as the split driver model consisting of a frontend driver and a backend driver.

- The frontend driver is running in Domain U and the backend driver is running in Domain 0. Interact with each other via a block of shared memory.

- The frontend driver manages the I/O requests of the guest OSes and the backend driver is responsible for managing the real I/O devices and multiplexing the I/O data of different VMs.

- Although para-I/O-virtualization achieves better device performance than full device emulation, it comes with a higher CPU overhead.

- **Direct I/O virtualization:**

- Lets the VM access devices directly. Can achieve close-to-native performance without high CPU costs.

- Lot of challenges for commodity hardware devices.

- For example, when a physical device is reclaimed, it may have been set to an arbitrary state (e.g., DMA to some arbitrary memory locations) that can function incorrectly or even crash the whole system.