


Virtualization and Types of Virtualization

Y. V. Lokeswari AP/ CSE
SSN College Of Engineering



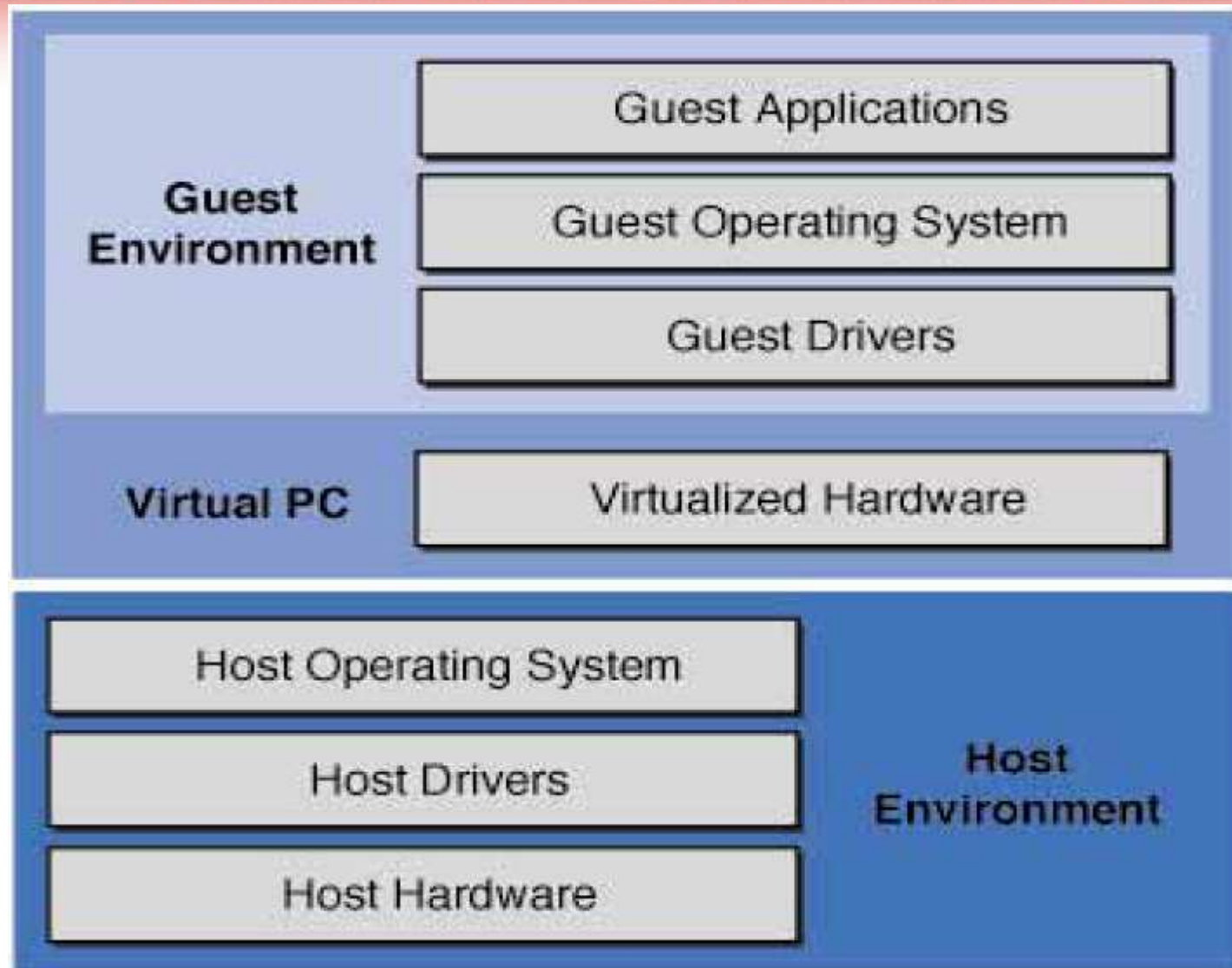
Outline

- Virtualization
 - Virtualization Structure Tools and Mechanisms
 - Virtualization of CPU, Memory and IO
 - Importance of Virtualization in Cloud
 - Disadvantages of Virtualization
- 

Virtualization

- **Virtualization** -- the abstraction of computer resources.
- Virtualization hides the physical characteristics of computing resources from their users, applications, or end users.
- This includes making a **single physical resource** (such as a server, an operating system, an application, or storage device) appear to function as **multiple virtual resources**; it can also include making **multiple physical resources** (such as storage devices or servers) appear as a **single virtual resource**.
- In computing, a process of creating a **illusion** of something like computer hardware, operating system (OS), storage device, or computer network resources is Virtualization.

Virtualization Layers



Privilege Rings

CPUs provide a range of protection levels also known as rings in which code can execute.

Ring 0 has the highest level privilege.

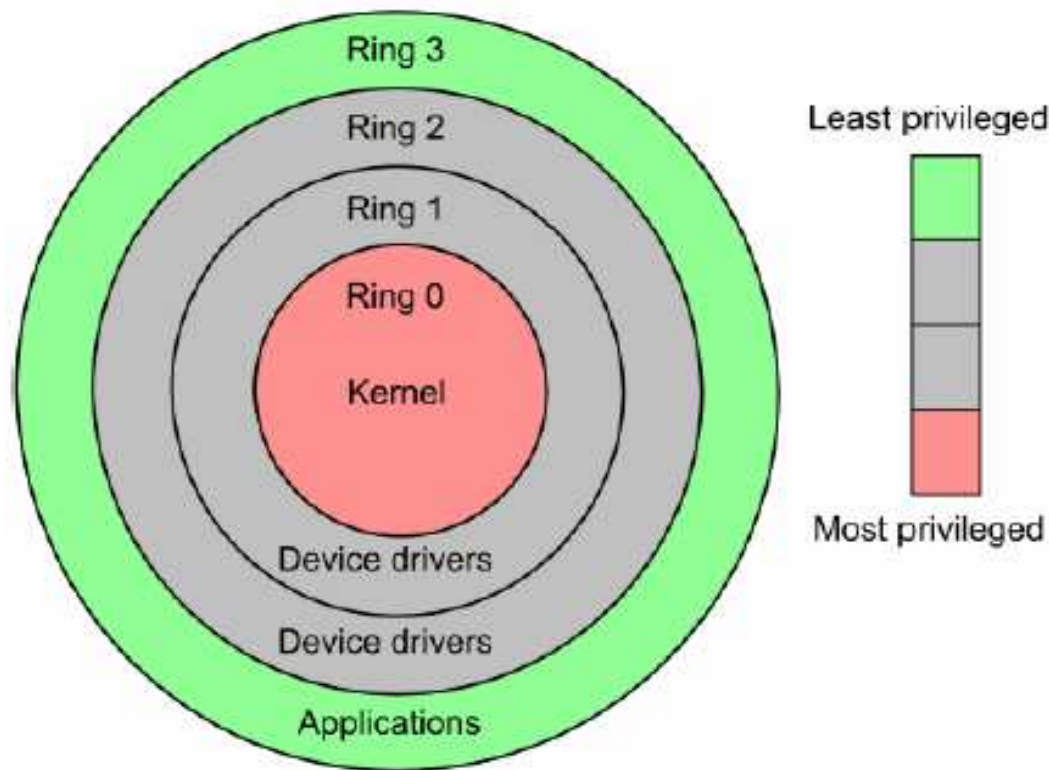


Figure: Privilege Rings [9]

Virtual Machines and Virtualization Middleware

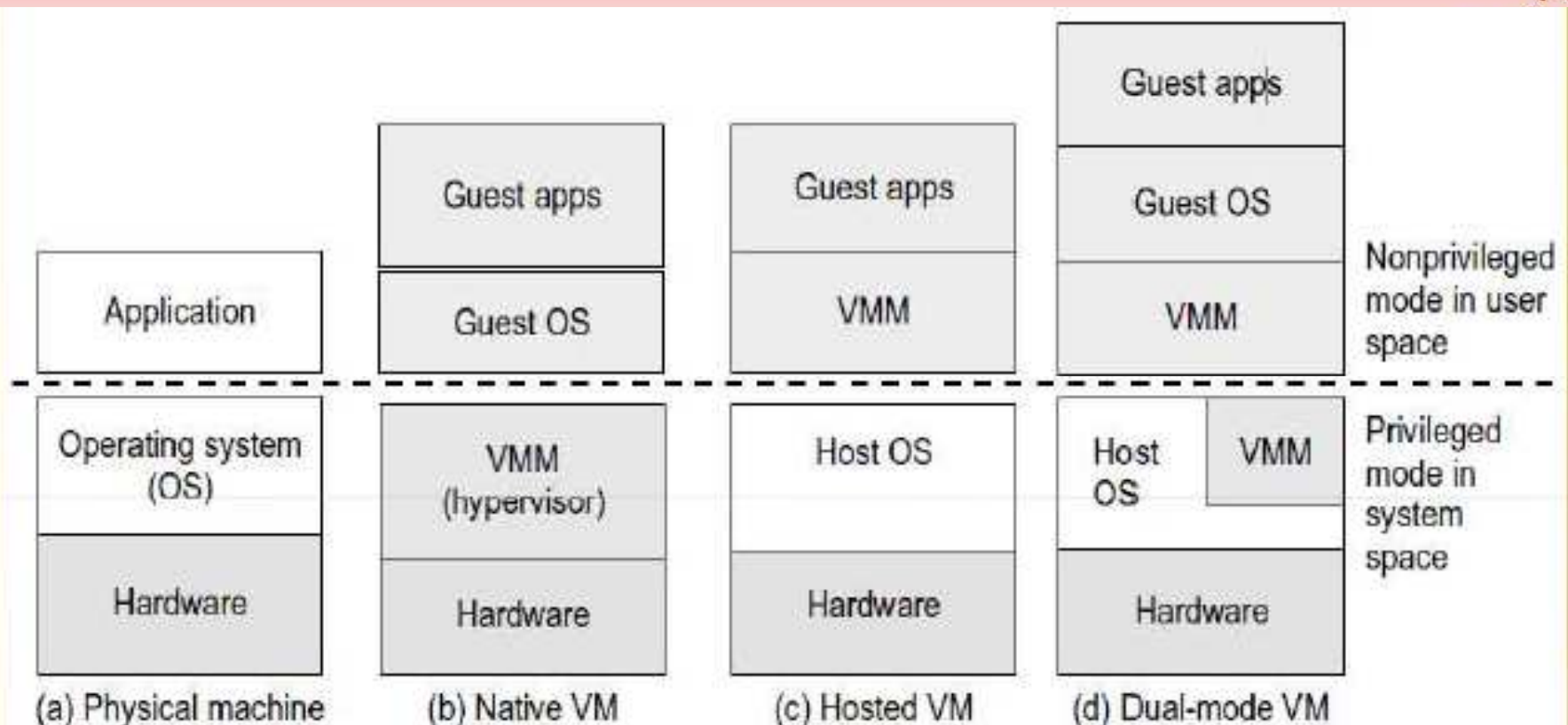


FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

Types of Virtualization

- We mainly focus on Platform virtualization which is mostly related to cloud-computing [1]
 - Full virtualization with Binary translation
 - Para-Virtualization with Compiler support
 - Hardware-assisted virtualization
 - OS-level virtualization

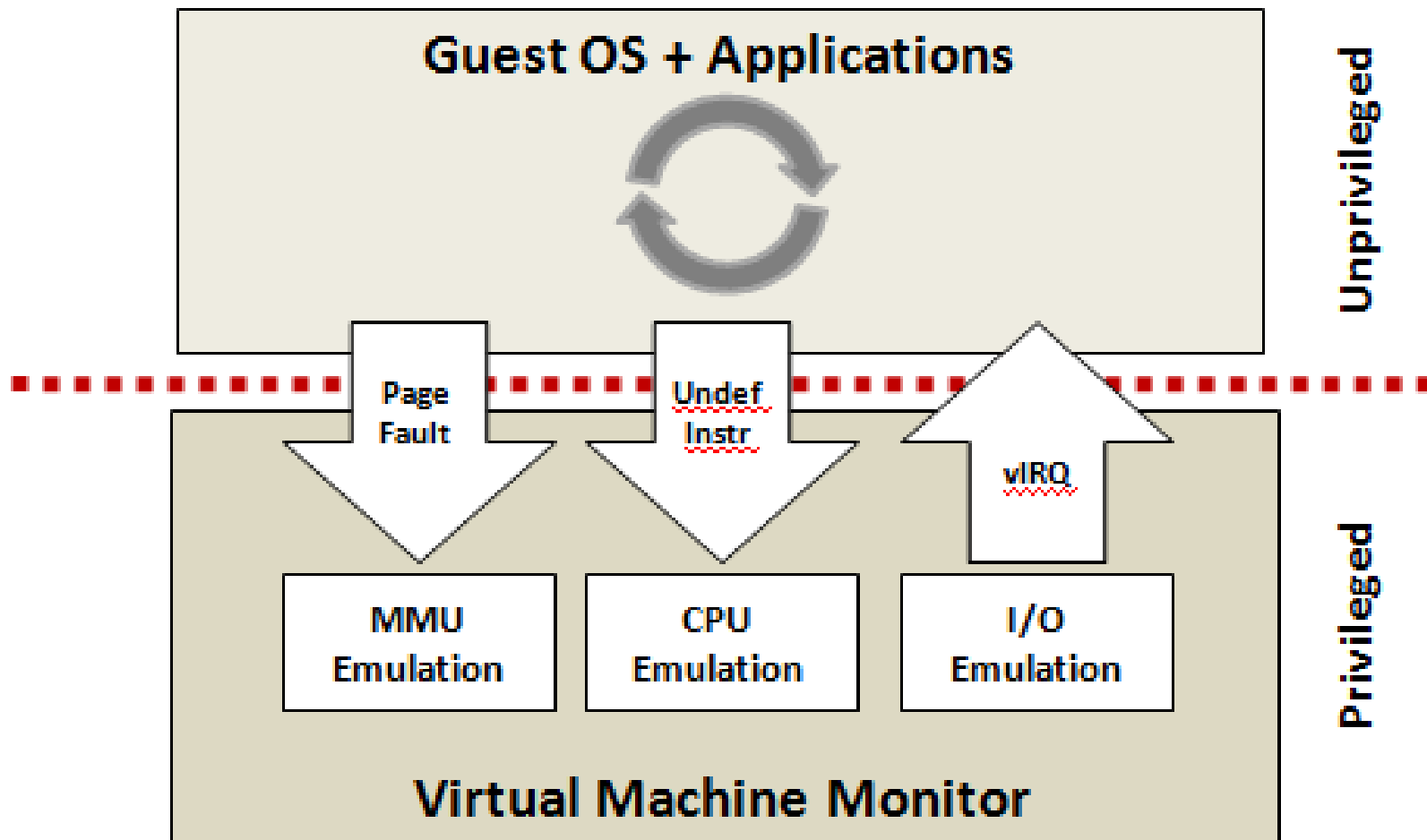
Virtualization Structure Tools and Mechanisms

- Full Virtualization with Binary Translation
- Para-Virtualization with Compiler Support

Binary translation

- Binary translation is one specific approach to implementing full virtualization
- It involves examining the executable code of the virtual guest for “unsafe” instructions, translating these into “safe” equivalents, and then executing the translated code. [3]
- An unsafe instruction is one that for example tries to access or modify the memory of another guest.

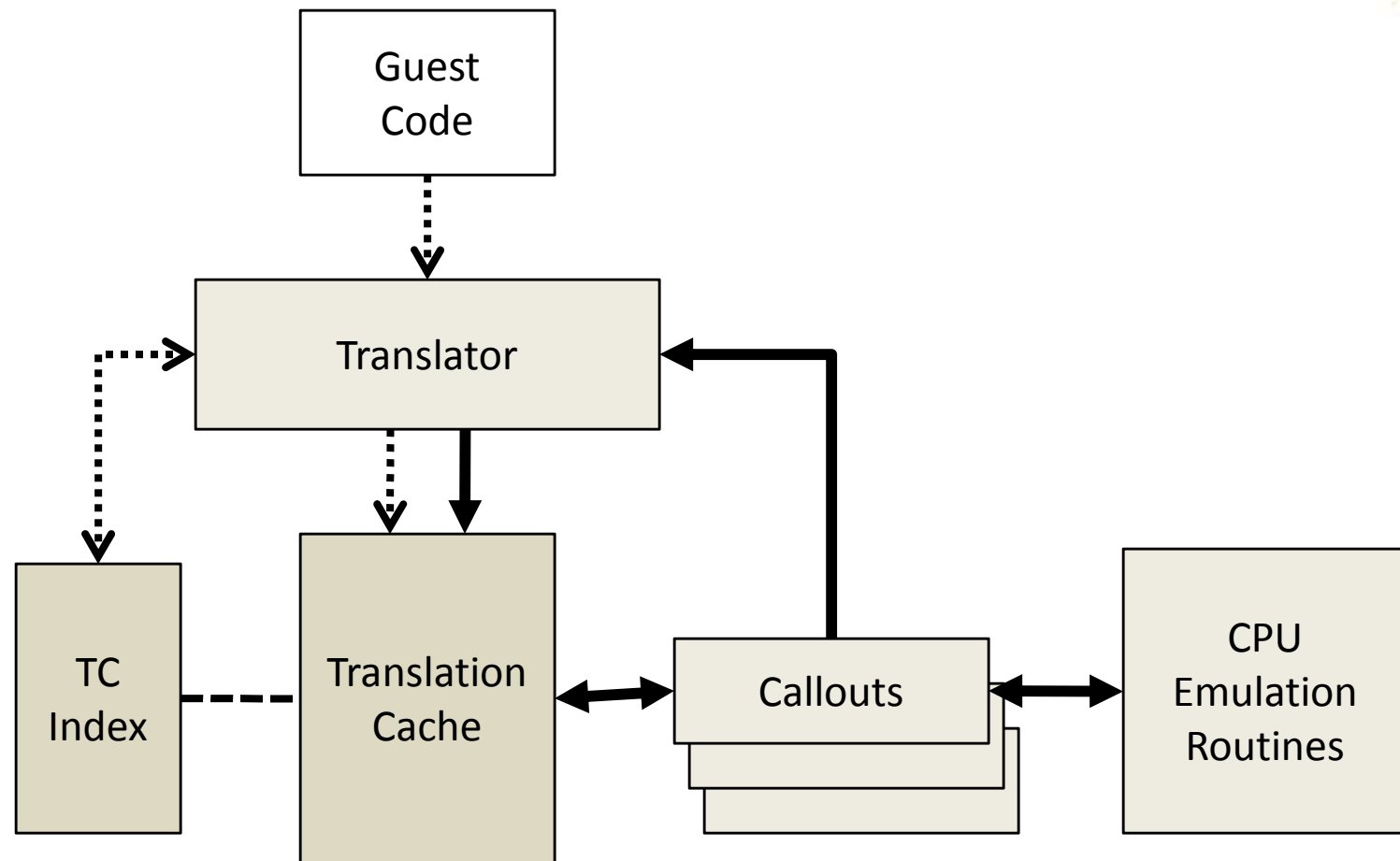
Trap and Emulate



Binary translation

- **Trap & Emulate** : Executable code from the guest is allowed to execute directly on the host CPU by the hypervisor
- A **Trap** is an exceptional condition that transfers control back to the hypervisor.
- Once the hypervisor has **received** a **trap**, it will **inspect** the offending **instruction**, **Emulate** it in a **safe** way, and **continue execution** after the instruction
- **Disadvantage**: It is **time-consuming** and **degrades** the performance.

Binary Translator



Direct Execution

- This is a mode that can be combined with binary translation.
- With **direct execution**, most code is **executed directly** on the CPU, and only the code that **needs** to be **translated** is **actually translated**

Virtualization Techniques

- Full virtualization using Binary Translation.
- OS Assisted Virtualization or Paravirtualization.
- Hardware Assisted Virtualization.

Full Virtualization / Host-based Virtualization

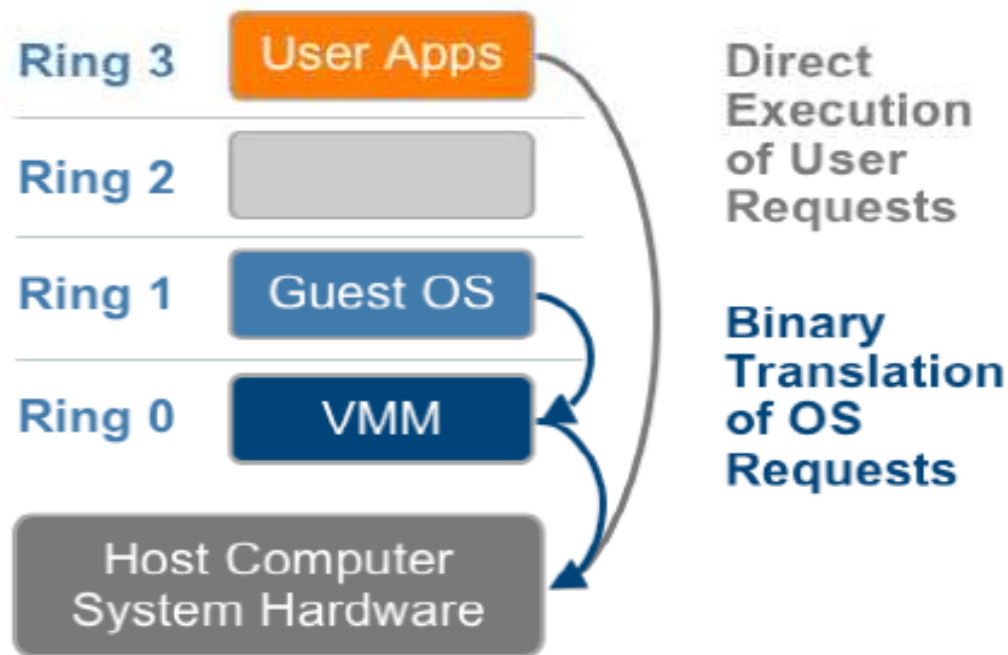
- **Full Virtualization** provides complete simulation of the underlying hardware.
- Simulate all computing elements such as instruction set, main memory, interrupts, exceptions, and device access.
- The result is a system in which all software (including all OS's) capable of execution on the raw hardware can be run in the virtual machine
- The combination of binary translation and direct execution provides Full Virtualization as the guest OS is fully abstracted (completely decoupled) from the underlying hardware by the virtualization layer.
- The guest OS is not aware it is being virtualized and requires no modification.

Full Virtualization

- The hypervisor translates all **operating system instructions** on the fly and **caches** the **results** for future use, while **user level instructions** run **unmodified** at native speed.
- **Full Virtualization** is done at **run time**.
- Examples
 - VMware
 - Microsoft Virtual Server
- Full virtualization has proven highly successful
 - **Sharing a computer system among multiple users**
 - **Isolating users from each other (and from the control program) and**
 - **Emulating new hardware to achieve improved reliability, security and productivity.**

Full Virtualization with Binary translation

Almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified.



VMM: Virtual Machine Monitor

Full Virtualization

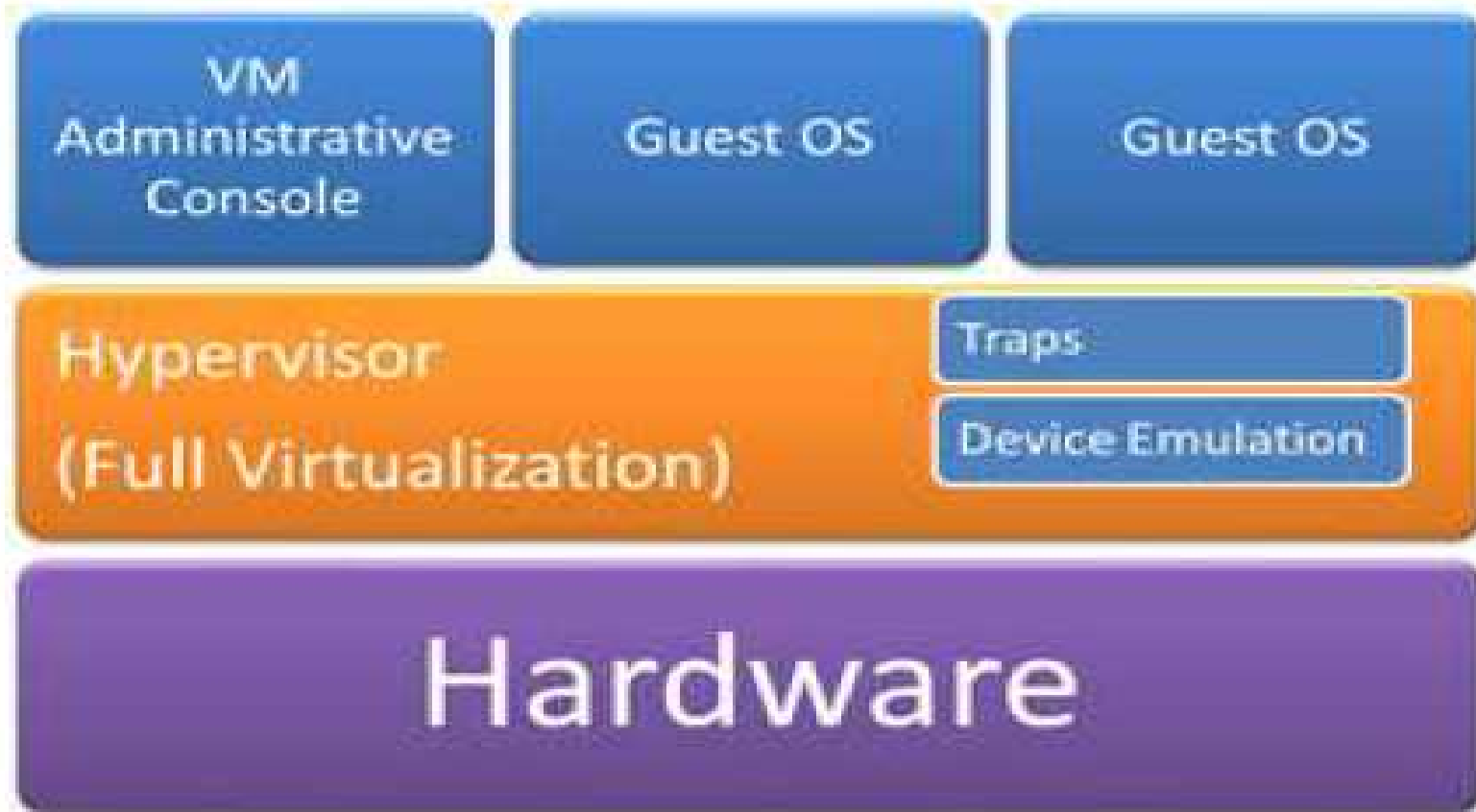


Figure: Full Virtualization

Full Virtualization

- Advantages:
 - It is flexible
 - Guest OS is decoupled from Hardware and unaware that it is virtualized.
- Disadvantages:
 - It may be slow.
 - Binary Translation incurs time which affects performance.
 - Full Virtualization of I/O intensive applications is a big challenge.

Full Virtualization

- Full Virtualization can be done in two forms
 - 1. Bare- Metal Virtual Machine (Native VM)
 - 2. Hosted Virtual Machine (**Host-Based Virtualization**)

Types of Hypervisor – Bare-Metal VM

A Type-1 hypervisor interacts directly with hardware that is being virtualized.

- It is completely independent from the operating system.
- Boots before the operating system (OS).

They are often referred to as a "native" or "bare metal" or "embedded" hypervisors in vendor literature.

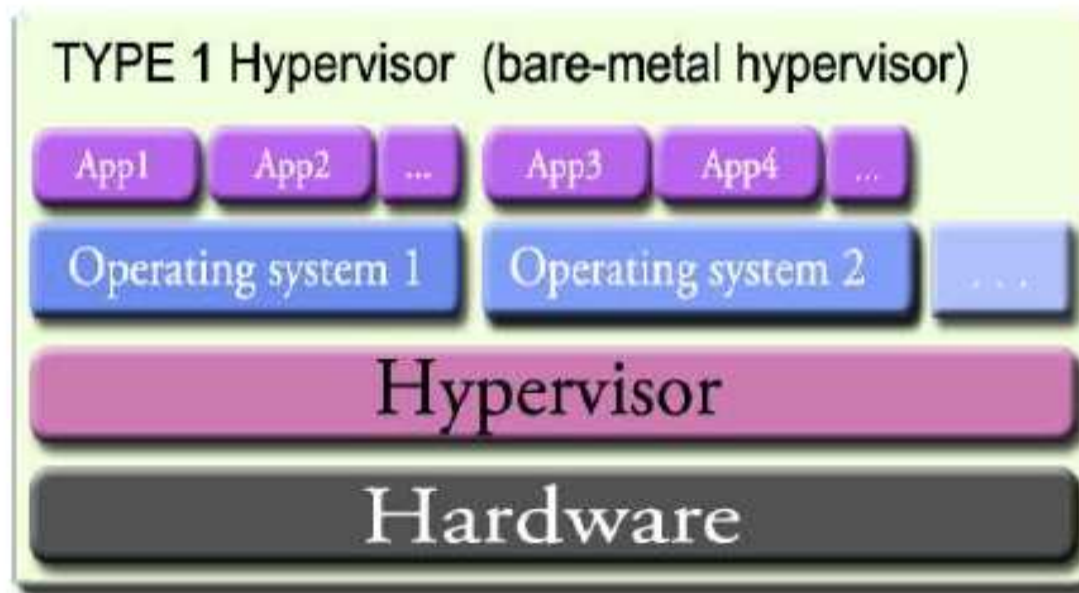


Figure: Type1 hypervisor

Types of Hypervisor- Hosted VM

Host-Based Virtualization

A Type-2 hypervisor sits on top of an operating system.

- Relies heavily on the operating system.
- It cannot boot until the operating system is already up and running.
- If operating system crashes, all end-users are affected.
- Since Type-2 hypervisors depend on an OS, they are not in full control of the end user's machine.

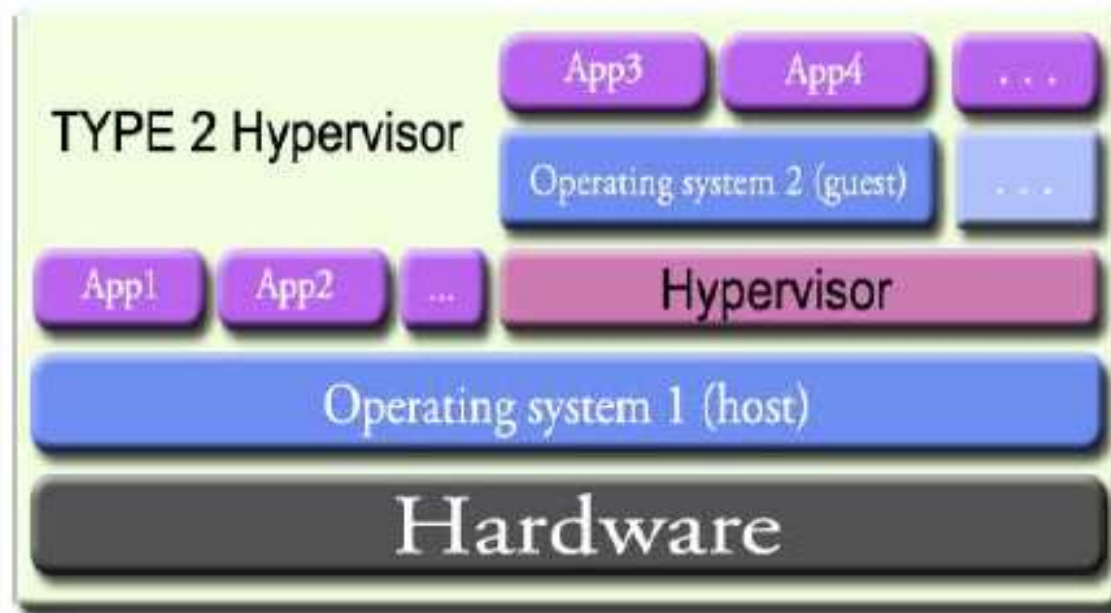


Figure: Type2 hypervisor

Types of Hypervisor- Hosted VM

Host-based Virtualization

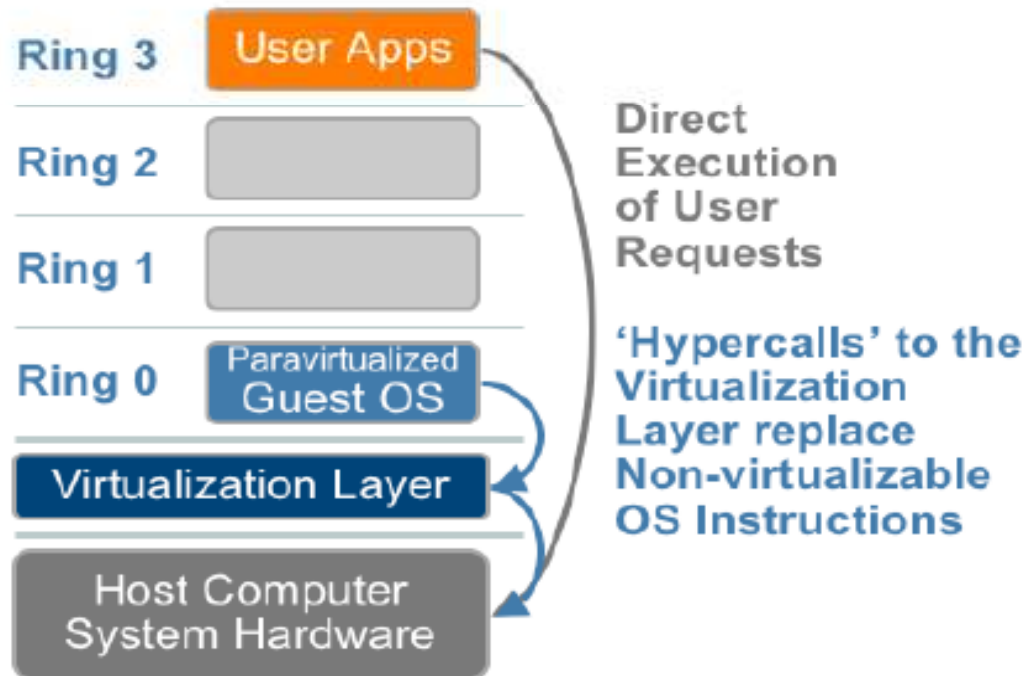
- **Advantages:**
 - User installs VM architecture **without modifying** Host OS.
 - More Flexible
 - Host Based Virtualization appeals to many **host machine configuration**.
- **Disadvantages:**
 - It may be **slow**.
 - Accessing hardware involves four layer mapping which **degrades performance**.
 - Requires **Binary Translation** when ISA of guest is different from ISA of host.

OS assisted (Para-virtualization)

- **Para-virtualization** – via an **modified OS kernel** as **guest OS**
 - To overcome the problem of degraded performance, **Guest OS kernel is modified**.
 - Para-virtualization is assisted with intelligent **compiler** to replace **non-virtualizable instructions** with **hypercalls** that communicate directly with the **virtualization layer or hypervisor**.
 - Para-virtualization is different from **full virtualization**, where the **unmodified OS** does **not know** it is **virtualized** and **sensitive OS** calls are **trapped** using binary translation.
 - Para-Virtualization is done at **Compile time**.
- Example:
 - Xen -- modified Linux kernel and a version of Windows XP
 - KVM (Kernel-based Virtual Machine) and VMware ESX

OS assisted (Para-virtualization)

A hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment.



VMM: Virtual Machine Monitor

OS assisted (Para-virtualization)

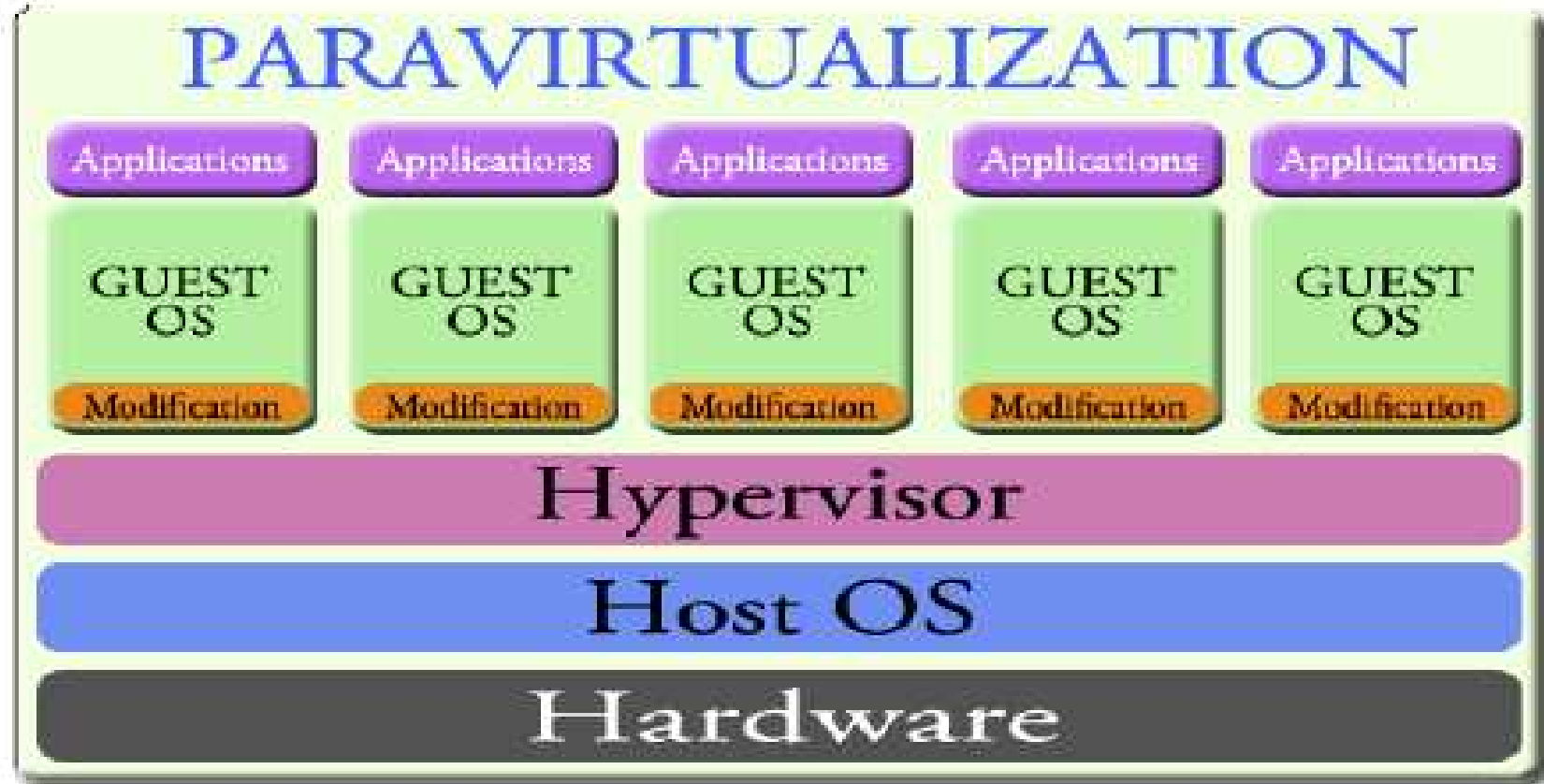
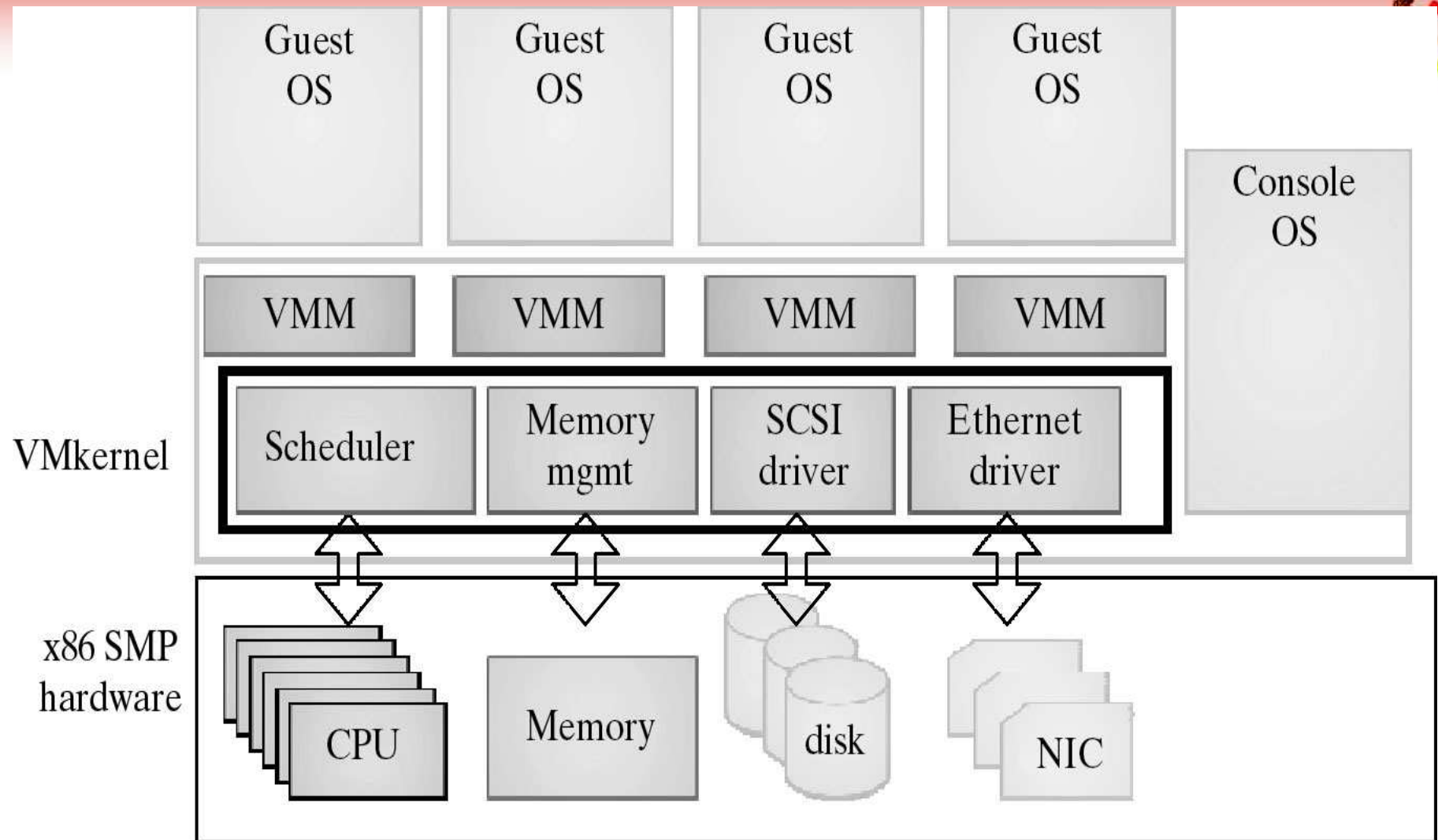


Figure: OS Assisted or Paravirtualization

OS assisted (Para-virtualization)

- Advantages
 - It eliminates Binary Translation
 - It is relatively easy and more practical.
- Disadvantages
 - Para-virtualization cannot support unmodified OS
 - Compatibility and portability is doubtful.
 - Maintaining Para-Virtualization is high as it required deep OS kernel modification.

VMware ESX Server : Para-virtualization)

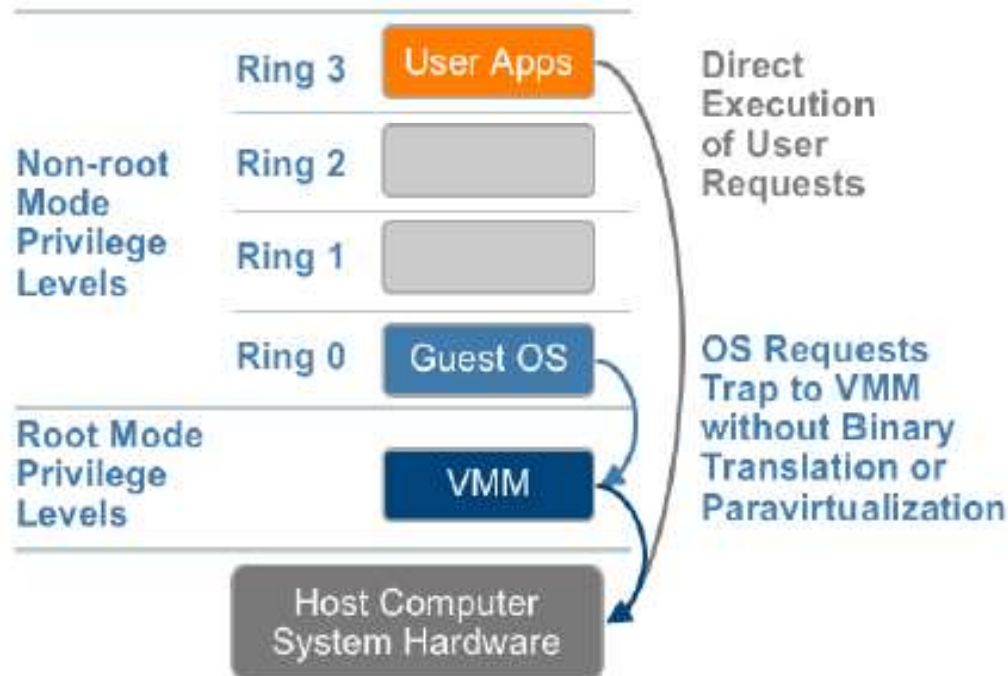


Hardware Assisted Virtualization

- Processor run multiple processes. All instructions from process will access hardware directly.
- Processors have two modes. **Supervisor** and **User** modes.
- Instructions running in **supervisor** mode are **privileged** instructions. Others are **unprivileged** instructions.
- Processor such as **x86** employ a **special running mode called root mode**.
- **VMM** and **Guest OS** run in **different modes**.
- **All sensitive instructions** of guest OS and its applications are **trapped** in **VMM**.

Hardware Assisted Virtualization

It is a way of improving the efficiency of hardware virtualization. It involves employing specially designed CPUs and hardware components that help improve the performance of a guest environment.



VMM: Virtual Machine Monitor

Hardware Assisted Virtualization

- This is also known as **Accelerated Virtualization**, hardware virtual machine (Xen), native virtualization (Virtual iron).
- **Hardware switch supported** by **CPU**, e.g.
 - Intel Virtualization Technology (VT-x)
 - AMD's AMD-V
- Intel and AMD add an additional mode to x86 processor.
- OS run in Ring 0 and Hypervisor run below Ring 0.
- Target privileged instructions with a **new CPU execution mode** feature allows the **VMM to run** in a **new root mode** below Ring 0.
- Privileged and sensitive calls are set to **automatically trap** to the **hypervisor**, removing the **need** for either **Binary Translation** or **OS modification (Para-Virtualization)**

Hardware Assisted Virtualization

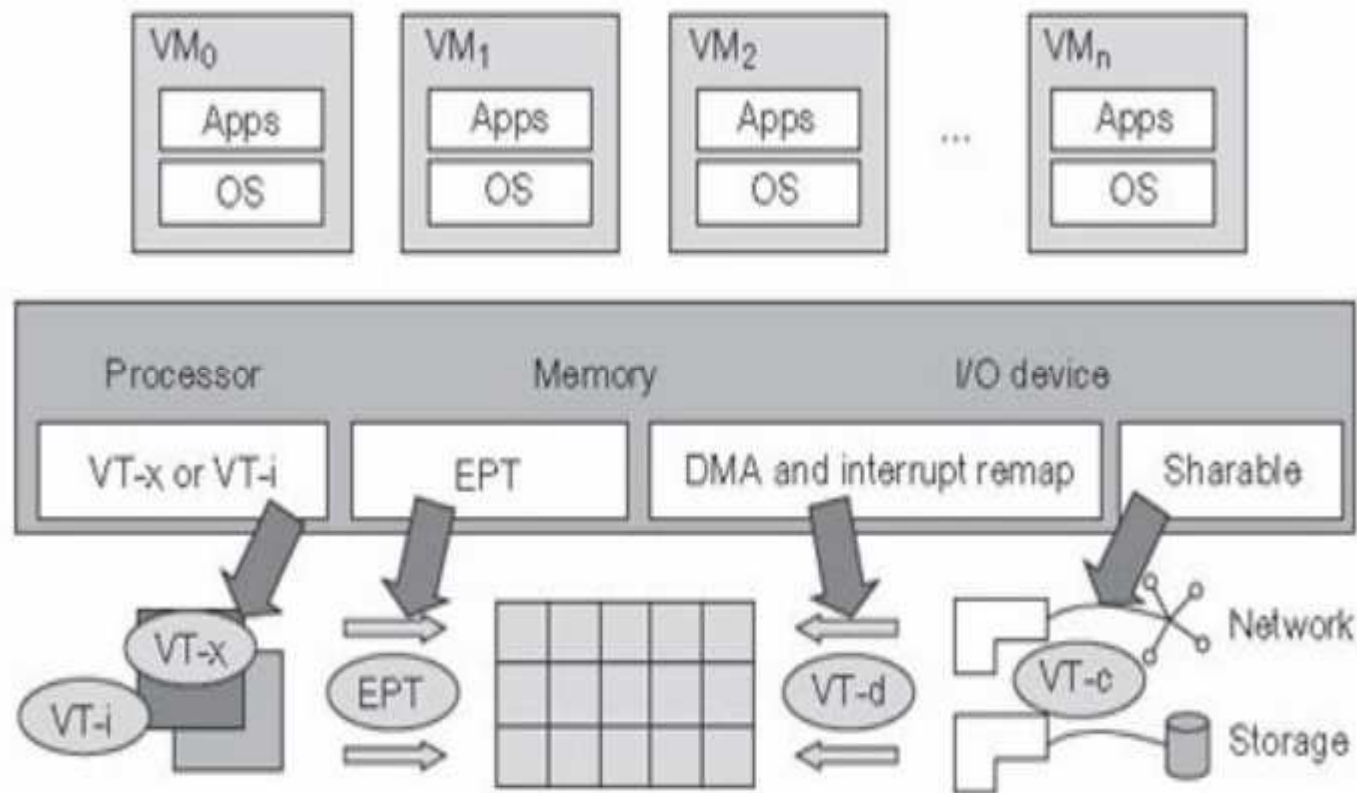


FIGURE 3.10

Intel hardware support for virtualization of processor, memory, and I/O devices.

(Modified from [68], Courtesy of Lizhong Chen, USC)

Hardware Assisted Virtualization

- KVM (*Kernel-based Virtual Machine*) is a Linux kernel virtualization infrastructure.
- KVM can support hardware-assisted virtualization and para-virtualization by using the Intel VT-x or AMD-v and VirtIO framework, respectively.
- The VirtIO framework includes a para-virtual Ethernet card, a disk I/O controller, a balloon device for adjusting guest memory usage.
- VGA graphics interface using VMware drivers.

Hardware Assisted Virtualization

- Advantages
 - Removes difficulty of Binary Translation of Full Virtualization and lets the OS to run VMs without modification.
- Disadvantages
 - Transition from Hypervisor to Guest OS incurs high overhead switches between processor modes.
- Virtualization such as VMware uses hybrid approach
 - Few tasks are offloaded to hardware and others are done with software.
 - Para-Virtualization and hardware-assisted virtualization can be combined together to improve performance

Virtualization of CPU, Memory & IO

- CPU Virtualization
- Memory Virtualization
- IO Virtualization

CPU Virtualization

- A **VM** is a **duplicate** of an **existing computer** system
- Majority of the VM instructions are executed on the host processor in native mode.
- **Unprivileged instructions** of VMs run **directly** on the **host machine** for higher efficiency.
- Other **critical instructions** should be **handled carefully** for correctness and stability.
- Critical Instructions are divided as *privileged instructions*, *control-sensitive instructions*, and *behavior-sensitive instructions*.

CPU Virtualization

Critical 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 instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

CPU Virtualization

- VM's privileged and unprivileged instructions run in the CPU's user mode while the VMM runs in supervisor mode.
- When the privileged instructions including control- and behavior-sensitive instructions of a VM are executed, they are trapped in the VMM.
- Not all CPU architectures are virtualizable. RISC CPU architectures can be naturally virtualized because all control and behavior-sensitive instructions are considered privileged instructions.
- On the contrary, x86 CPU architectures are not primarily designed to support virtualization.
- This is because about IO sensitive instructions, such as *SGDT* and *SMSW*, are *not privileged instructions*. They modify the *global memory segments*.
- When these instructions execute in virtualization, they cannot be trapped in the VMM.
- Binary Translation need to be implemented. (Full Virtualization)

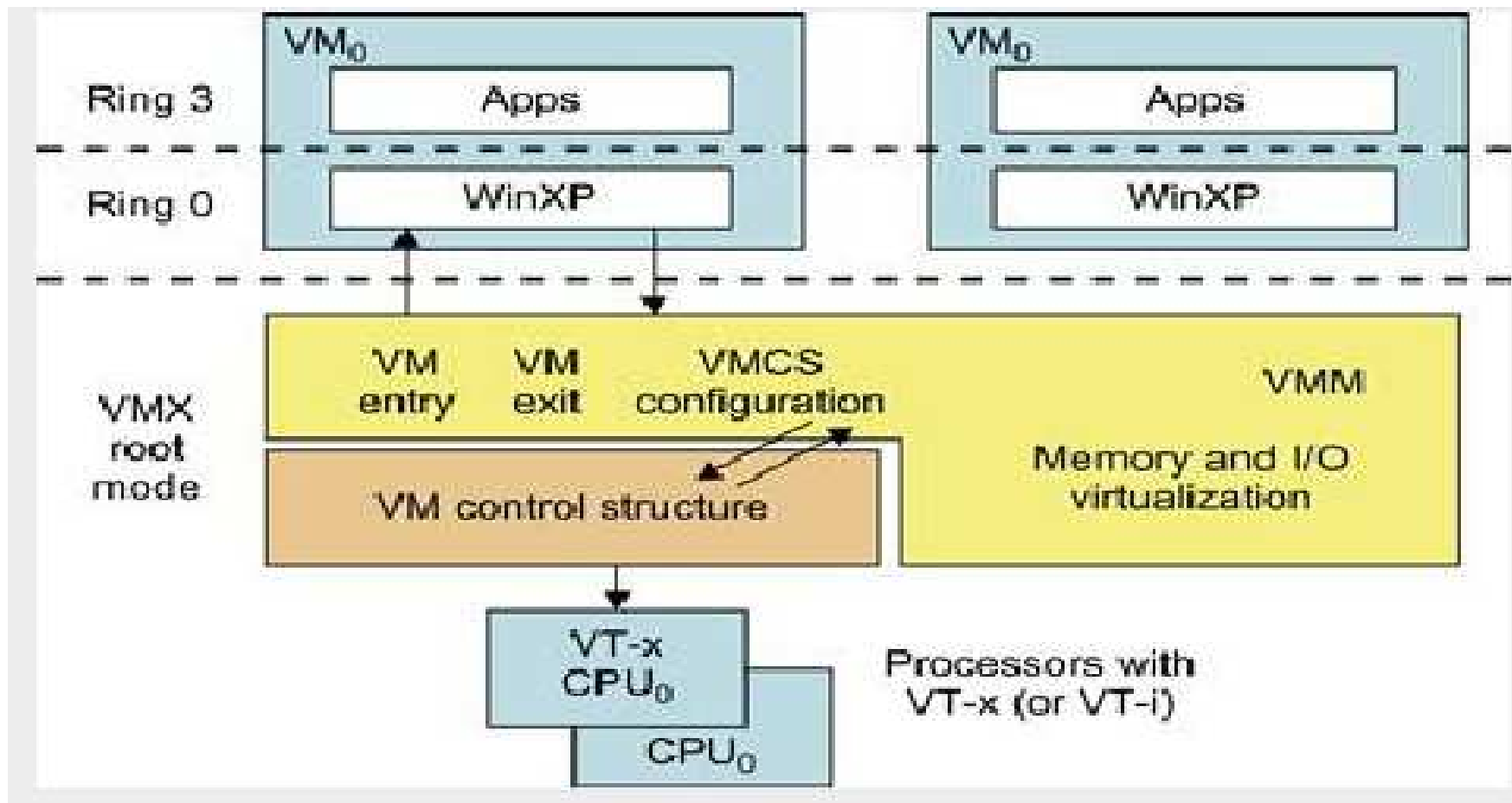
CPU Virtualization

Solution: Hardware-Assisted CPU Virtualization

- Intel and AMD add an additional mode called **privilege mode** level (some people call it Ring-1) to **x86 processors**.
- Therefore, **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**.

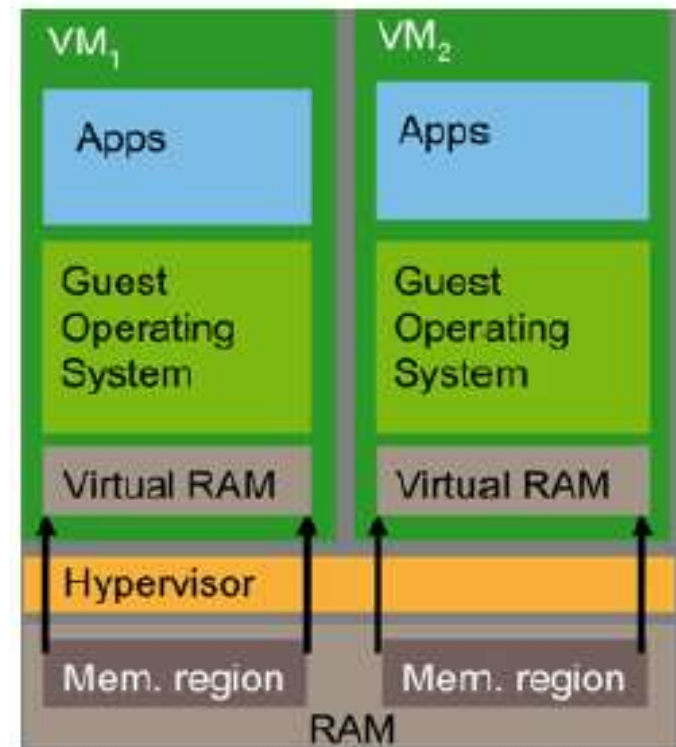
CPU Virtualization

Solution: Hardware-Assisted CPU Virtualization



Memory Virtualization

- To run **multiple virtual machines** on a **single system**, another level of **memory virtualization** is required.
- This involves sharing the **physical** system **memory** and dynamically **allocating** it to **virtual machines**.
- Not only virtual memory
- Hardware support
 - e.g., x86 MMU and TLB



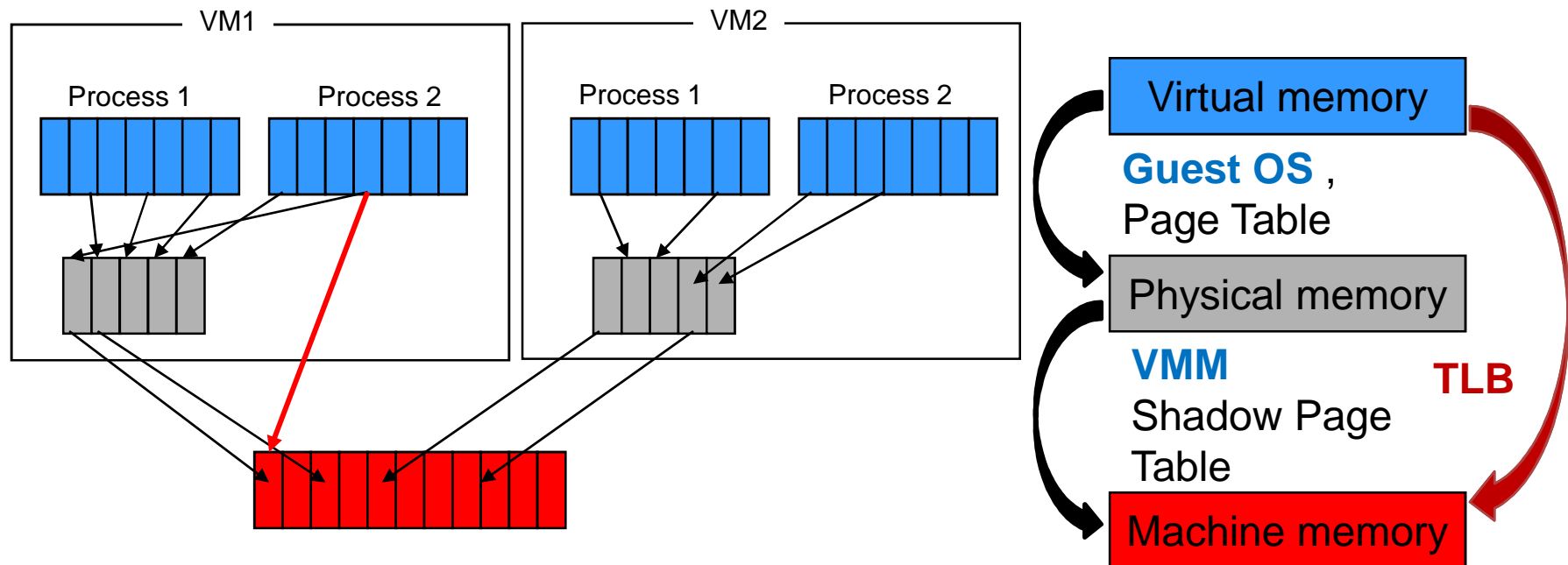
Memory Virtualization

Memory Virtualization

- The **guest OS** continues to control the mapping of **virtual addresses** to the **guest memory physical addresses** using **Page Table**
- The guest OS cannot have direct access to the actual machine memory.
- The **VMM** is responsible for mapping **guest physical memory** to the **actual machine memory**, and it uses **shadow page tables** to accelerate the mappings.
- **VMM** uses **TLB** hardware to map the **virtual memory directly** to the **machine memory** to avoid the two levels of translation on every access. (as indicated by red line in next slide fig)

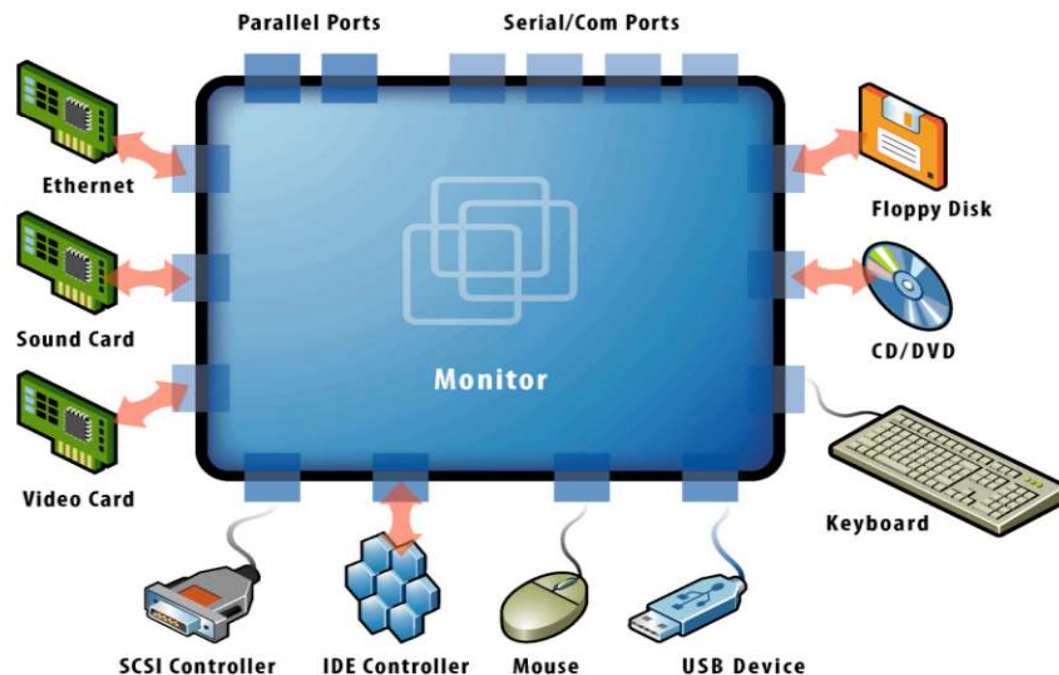
Memory Virtualization

- When the **guest OS** changes the **virtual memory** to **physical memory** mapping, the **VMM** updates the **shadow page tables** to enable a direct lookup.
- The **shadow page table** controls **which pages** of **machine memory** are **assigned** to a **given VM**



Device and I/O Virtualization

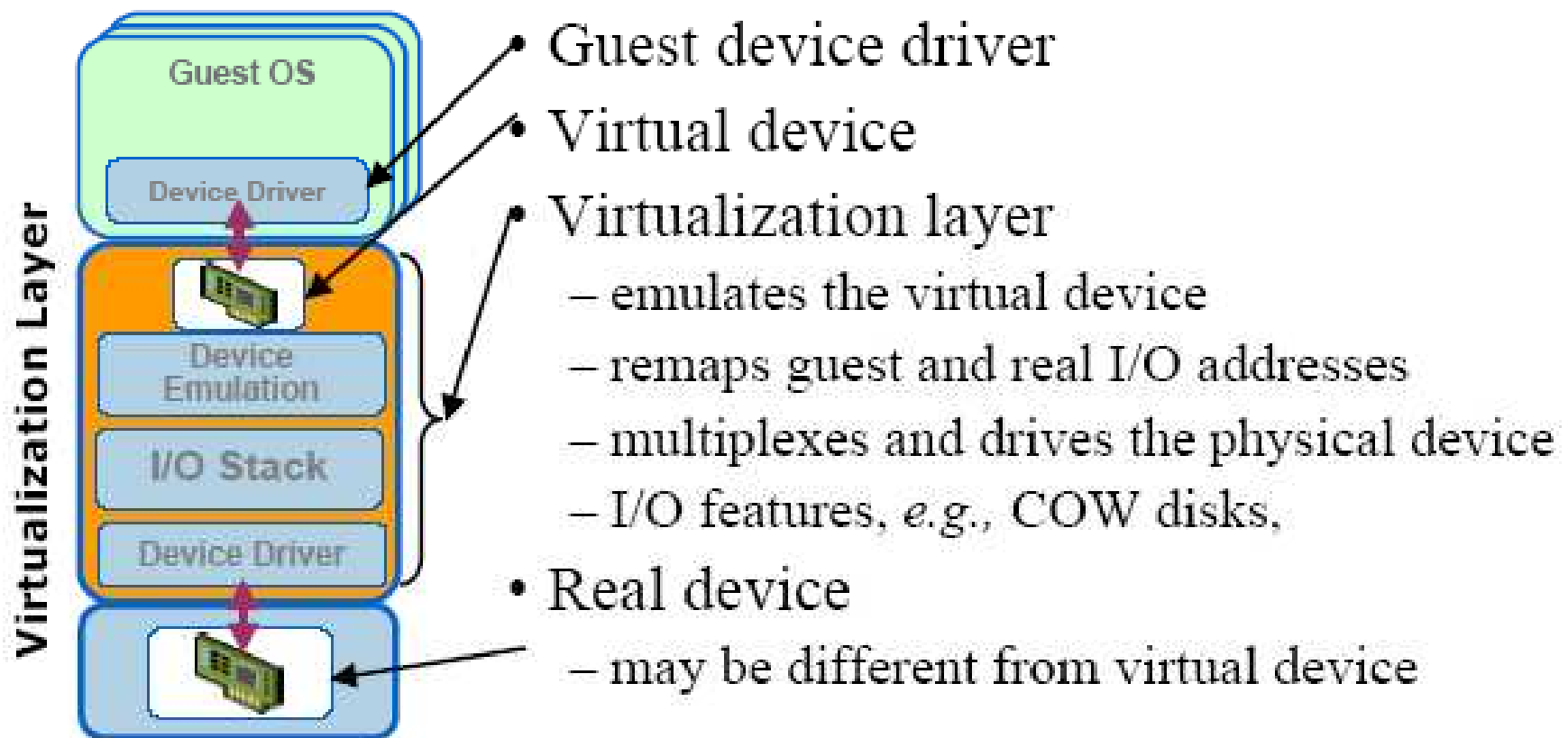
- VMM supports all device/IO drivers
- Physically/virtually existed



Source: VMware white paper, "Understanding Full Virtualization, Paravirtualization, and Hardware Assist"

Device and I/O Virtualization

Current virtual I/O devices



Device and I/O Virtualization

Para-Virtualized I/O:

Frontend driver manages I/O from Guest OS. Backend driver manages real I/O devices

Full Device emulation:

Replication of all devices is done in a software.
S/w is located in VMM

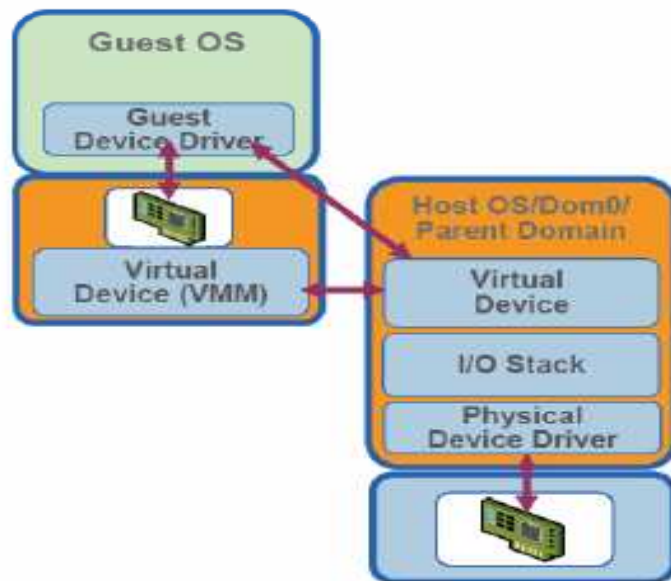
Direct I/O:

VM access device directly. It is challenging task

I/O Virtualization Implementations

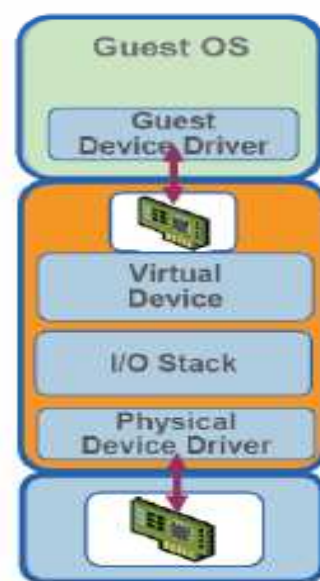
Virtualized I/O

Hosted or Split



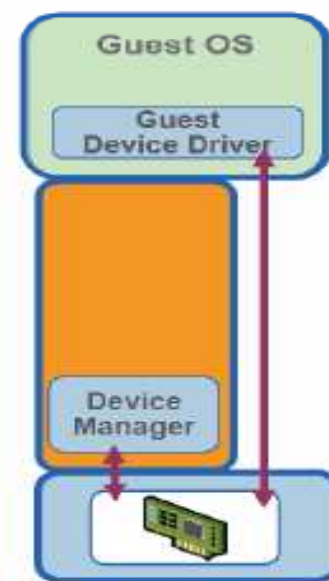
VMware Workstation, VMware Server,
VMware ESX Server
Microsoft Virtual Server, Xen

Hypervisor Direct



VMware ESX Server
(storage and network)

Passthrough I/O



A Future Option

Conclusions on CPU, Memory and I/O Virtualization

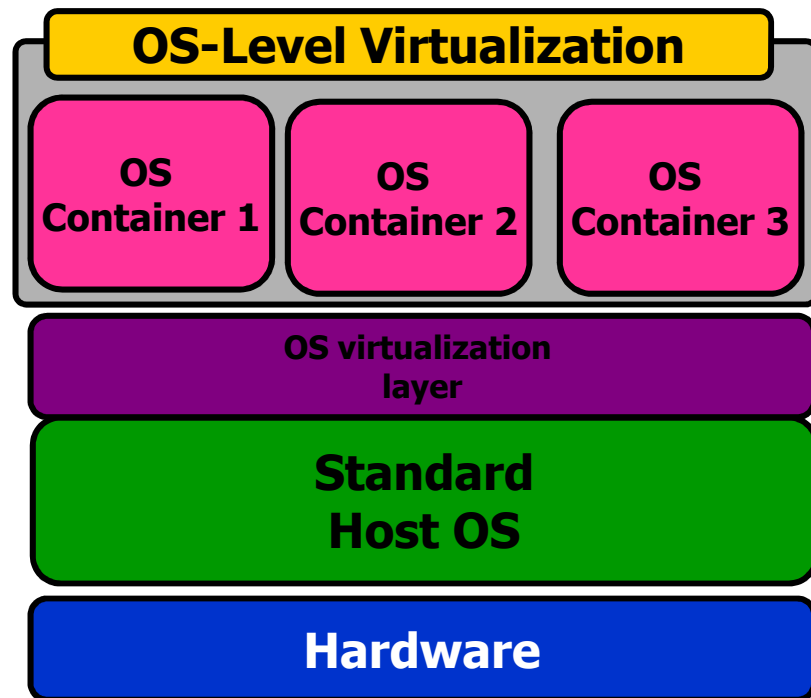
- CPU virtualization demands **hardware-assisted traps** of **sensitive instructions** by the VMM
- Memory virtualization demands special **hardware** support (shadow page tables by VMWare or extended page table by Intel) to help translate **virtual address** into **physical address** and **machine memory** in two stages.
- I/O virtualization is the most **difficult** one to **realize** due to the **complexity** of I/O service routines and the **emulation** needed between the **guest OS** and **host OS**

Conclusions on Full , Para and Hardware – assisted Virtualization

- Full virtualization at the hardware level has the disadvantages of slow performance and low density
- Need for Para-Virtualization is to modify the guest OS.
- To reduce the performance overhead of hardware level virtualization, even hardware modification is needed
- OS-level virtualization provides a feasible solution for these hardware-level virtualization issues

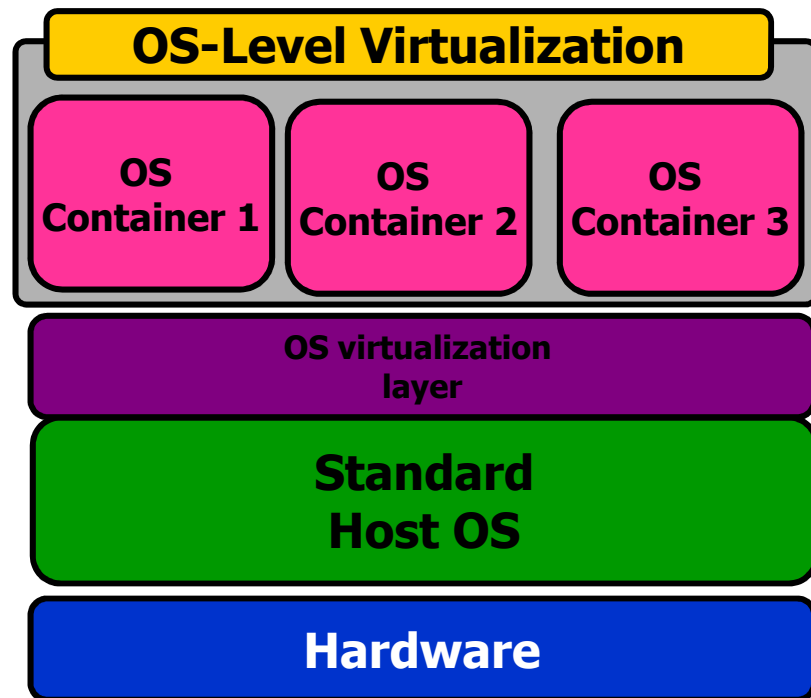
OS-Level Virtualization

- OS-level virtualization
- Operating system level virtualization **inserts a virtualization layer** inside an **operating system** to partition a machine's physical resources.
- It enables **multiple isolated VMs** within a **single operating system** kernel.
- This kind of VM is often called a *Virtual execution Environment (VE)*, *Virtual Private System (VPS)*, or simply *container*



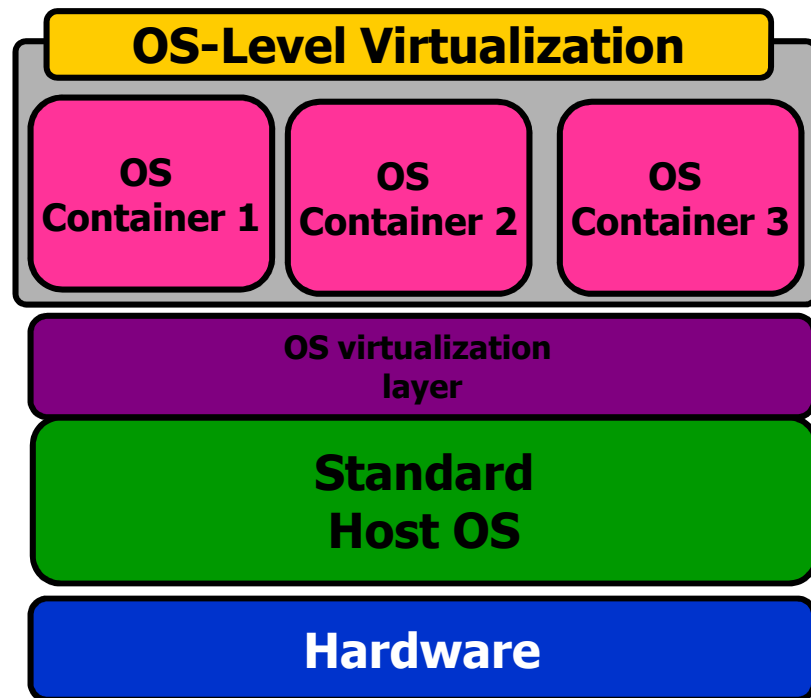
OS-Level Virtualization

- OS-level virtualization
- An OS-level virtualization approach **doesn't use a hypervisor** at all.
- Instead, the **virtualization** capability is **part** of the **host OS**, which performs all the functions of a fully virtualized hypervisor.



OS-Level Virtualization

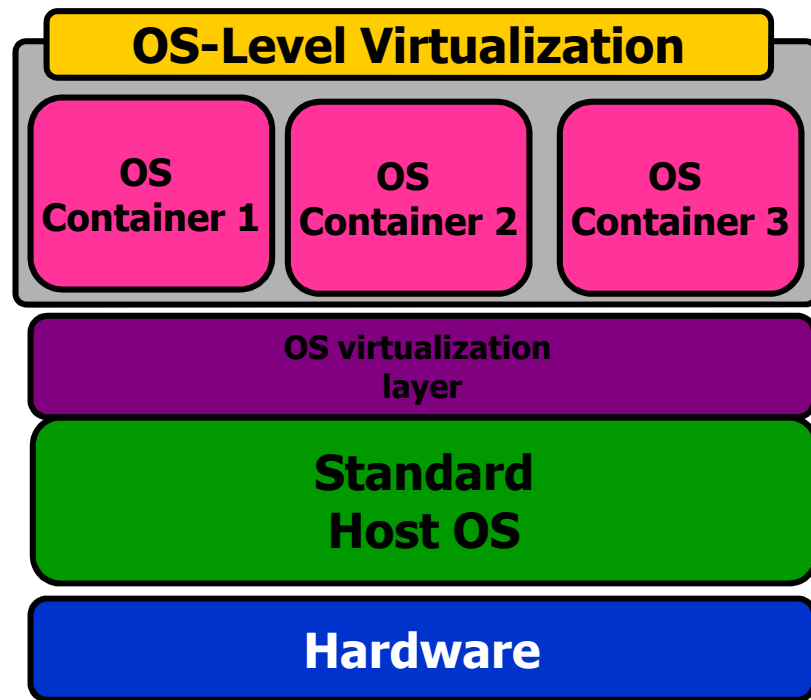
- OS-level virtualization
- From the user's point of view, **VEs** look like **real servers**.
- This means a VE has its own set of processes, file system, user accounts, network interfaces with IP addresses, routing tables, firewall rules, and other personal settings.
- Although **VEs** can be customized for **different people**, they **share** the **same operating system kernel**.
- Therefore, OS-level virtualization is also called **single-OS image** virtualization..



OS-Level Virtualization

OS-level virtualization

- The biggest limitation of this approach is that **all** the **guest servers** must run the **same OS**.
- Each **virtual server** remains **independent** from all the **others**, but you can't mix and match operating systems among them.
- Because all the guest operating systems must be the same, this is called a **homogeneous** environment.



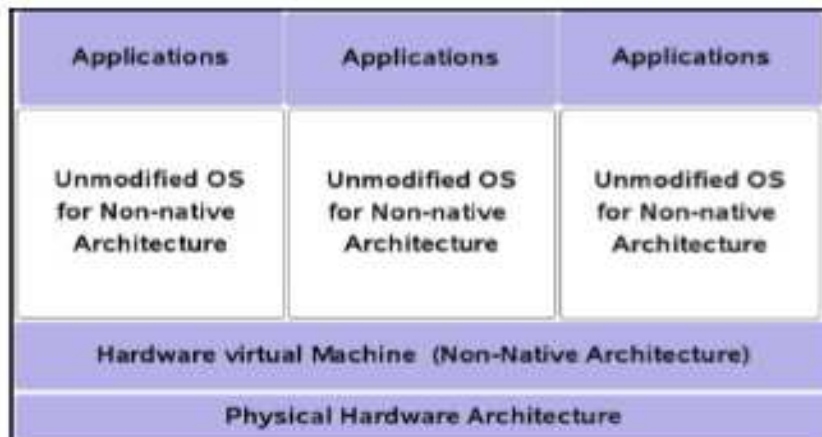
Confusion...

- **OS-Level Virtualization.** A type of server virtualization technology which works at the **OS layer**. The physical **server** and single instance of the **operating system** is **virtualized** into **multiple** isolated partitions, where each partition replicates a real server. The **OS kernel** will **run** a **single operating system** and provide that operating system functionality to each of the partitions.
- **Operating system virtualization** refers to the use of **software** to allow system **hardware** to run **multiple instances** of **different operating systems** concurrently, allowing you to **run different applications** requiring **different operating systems** on **one computer** system. The operating systems do not interfere with each other or the various applications.

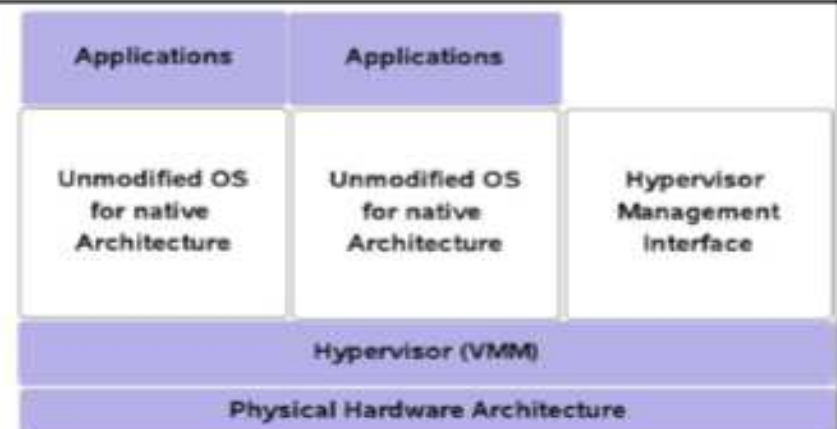
Summary of Virtualization Types

- Binary translation is the most established technology for full virtualization
- Hardware assist is the future of virtualization, but it still has a long way to go
- Para-Virtualization delivers performance benefits with maintenance costs
 - Xen
 - VMWare

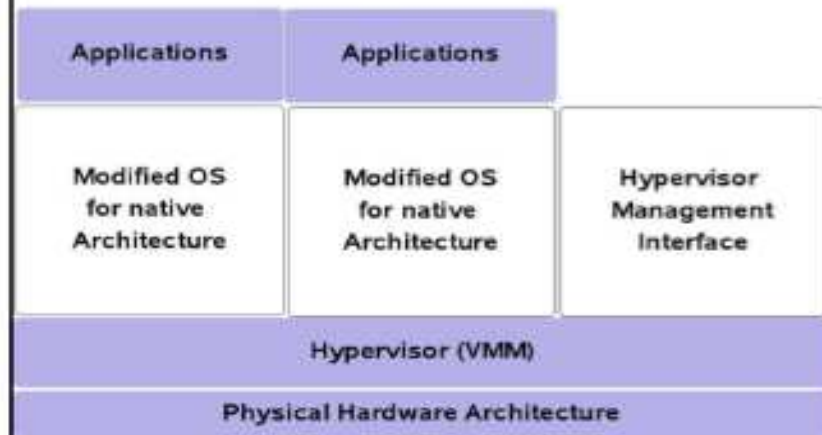
Types of Virtualization



Emulation



Full Virtualization

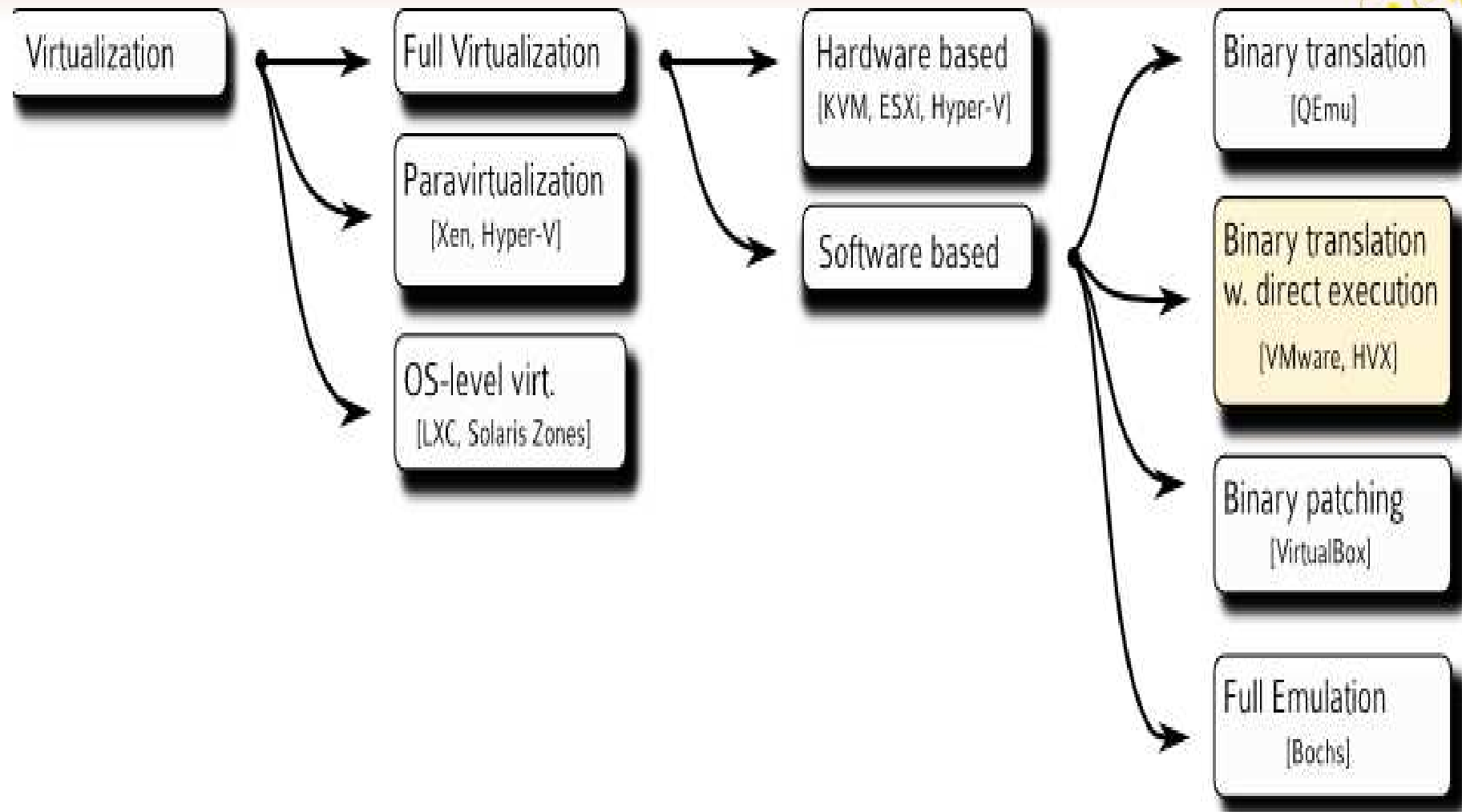


Para Virtualization



OS level Virtualization

Virtualization Taxonomy



Techniques for X86 virtualization

	Full Virtualization with Binary Translation	Hardware Assisted Virtualization	OS Assisted Virtualization / Paravirtualization
Technique	Binary Translation and Direct Execution	Exit to Root Mode on Privileged Instructions	Hypercalls
Guest Modification / Compatibility	Unmodified Guest OS Excellent compatibility	Unmodified Guest OS Excellent compatibility	Guest OS modified to issue Hypercalls . So it can't run on Native Hardware or other Hypervisors Poor compatibility; Not available on Windows OSes
Performance	Good	Fair Current performance lags Binary Translation virtualization on various workloads but will improve over time	Better in certain cases
Used By	VMware, Microsoft, Parallels	VMware, Microsoft, Parallels, Xen	VMware, Xen
Guest OS Hypervisor Independent?	yes	yes	XenLinux runs only on Xen Hypervisor VMI-Linux is Hypervisor agnostic

Source: VMware white paper, "Understanding Full Virtualization, Paravirtualization, and Hardware Assist"

Importance of Virtualization in Cloud

- Cloud can exist without Virtualization, although it will be difficult and inefficient.
- Cloud makes notion of "Pay for what you use" and "infinite availability- use as much as you want".
- These notions are practical only if we have
 - lot of flexibility.
 - efficiency in the back-end.
- This efficiency is readily available in Virtualized Environments and Machines.
- Virtualization is not necessary to create a cloud environment, but it enables rapid scaling of resources in a way that non-virtualized environments find hard to achieve.

Disadvantages of Virtualization

- Specific hardware requirements
- Upfront cost
- Some programs don't take Virtualization
- VMs need lot of power
- You still need those old desktops
- You will need fast network
- Computing is virtual, peripherals are real.
- Some hardware architectures or features are impossible to virtualize such as:
 - Certain registers or state not exposed
 - Clocks, time, and real-time behavior
- Virtualization may not work well for :
 - Resource-intensive applications
 - VMs may have RAM/CPU limitations
 - Performance testing
 - Hardware compatibility testing

Multi-Core Virtualization:

VCPU vs. traditional CPU

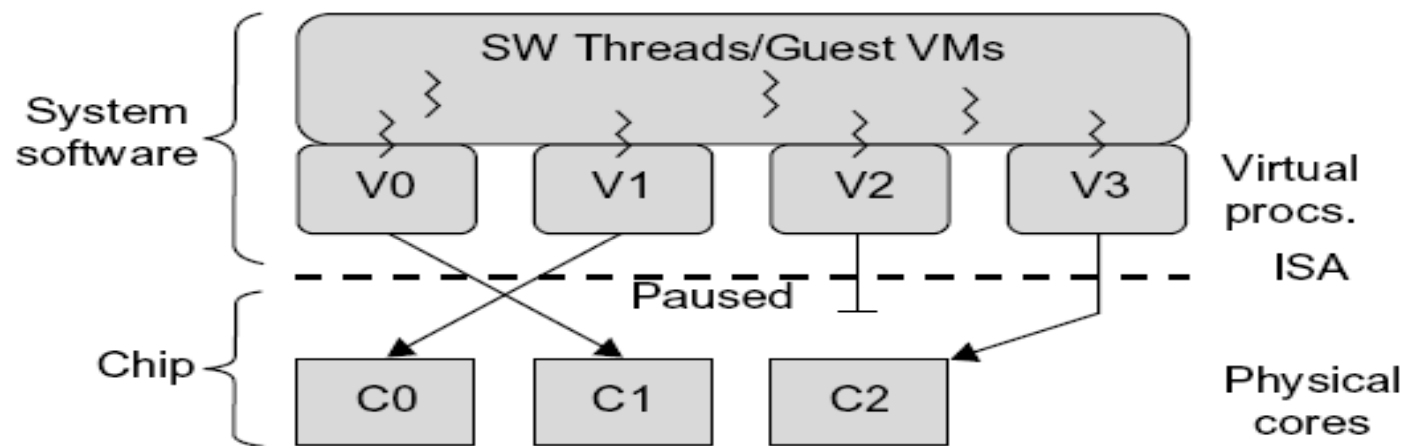


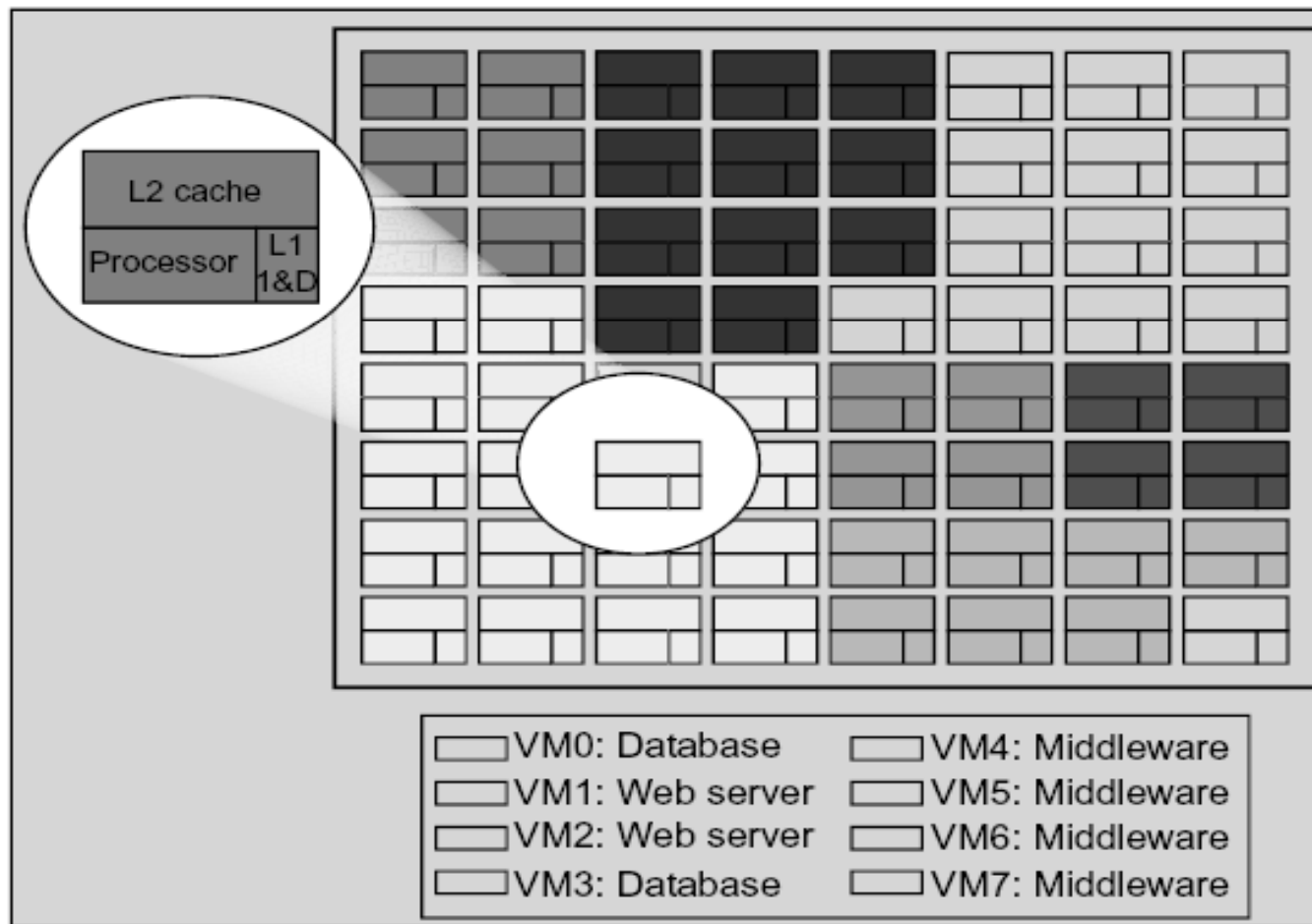
Figure 3.16 Four VCPUs are exposed to the software, only three cores are actually present. VCPUs V0, V1, and V3 have been transparently migrated, while VCPU V2 has been transparently suspended. (Courtesy of Wells, et al., “Dynamic Heterogeneity and the Need for Multicore Virtualization”, *ACM SIGOPS Operating Systems Review*, ACM Press, 2009 [68])

Virtual Cores vs. Physical Processor Cores

Physical cores	Virtual cores
The actual physical cores present in the processor.	There can be more virtual cores visible to a single OS than there are physical cores.
More burden on the software to write applications which can run directly on the cores.	Design of software becomes easier as the hardware assists the software in dynamic resource utilization.
Hardware provides no assistance to the software and is hence simpler.	Hardware provides assistance to the software and is hence more complex.
Poor resource management.	Better resource management.
The lowest level of system software has to be modified.	The lowest level of system software need not be modified.

CMP Server consolidation by Space-sharing of VMs

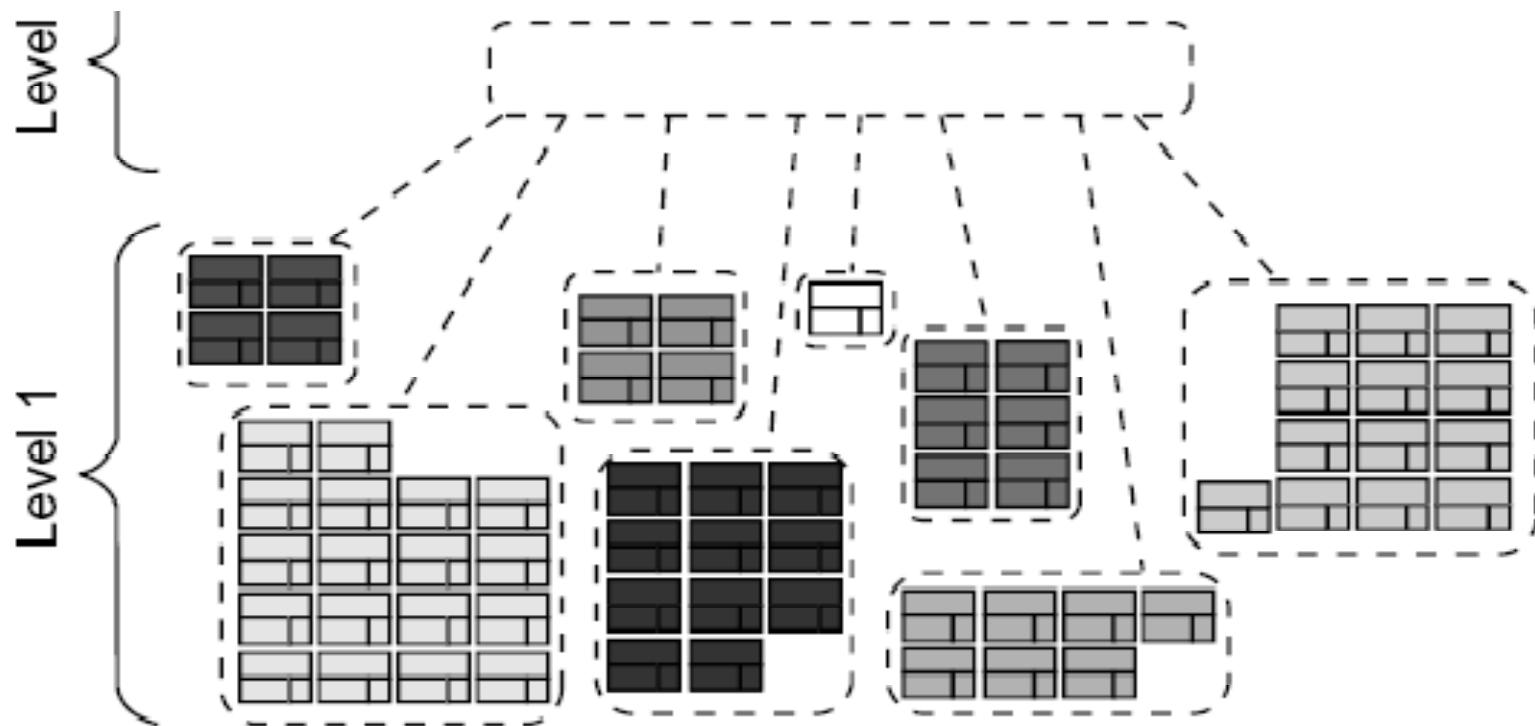
(Courtesy of Marty and Hill, 2007)



(a) Mapping of VMs into adjacent cores

Virtual Clusters in Many Cores

Space Sharing of VMs -- Virtual Hierarchy



(b) Multiple virtual clusters assigned to various workloads

(Courtesy of Marty and Hill, 2007)

Major VMM and Hypervisor Providers

VMM Provider	Host CPU	Guest CPU	Host OS	Guest OS	VM Architecture
VMware Work-station	X86, x86-64	X86, x86-64	Windows, Linux	Windows, Linux, Solaris, FreeBSD, Netware, OS/2, SCO, BeOS, Darwin	Full Virtualization
VMware ESX Server	X86, x86-64	X86, x86-64	No host OS	The same as VMware workstation	Para-Virtualization
XEN	X86, x86-64, IA-64	X86, x86-64, IA-64	NetBSD, Linux, Solaris	FreeBSD, NetBSD, Linux, Solaris, windows XP and 2003 Server	Hypervisor
KVM	X86, x86-64, IA64, S390, PowerPC	X86, x86-64, IA64, S390, PowerPC	Linux	Linux, Windows, FreeBSD, Solaris	Para-Virtualization

Summary

- Virtualization
- Virtualization Structure Tools and Mechanisms
- Virtualization of CPU Memory and IO
- Importance of Virtualization in Cloud
- Disadvantages of Virtualization

References

1. “Understanding Full, Para and Hardware Assisted Virtualization,”
[www.vmware.com/files/pdf/VMware paravirtualization.pdf](http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf).
2. <http://www.computerworld.com/article/2551154/virtualization/emulation-or-virtualization-.html>
3. <http://www.ravellosystems.com/blog/nested-virtualization-with-binary-translation/>
4. “OVA-open-virtualization-alliance,”
<https://openvirtualizationalliance.org/>.
5. <http://www.xen.org/>
6. “Open Virtualization Format for Virtual Machines”, available at
<http://www.vmware.com/appliances/getting-started/learn/ovf.html>
7. M. Marty, M. Hill, Virtual hierarchies to support server consolidation, in: Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA), 2007.

Primitive Operations in Virtual Machines

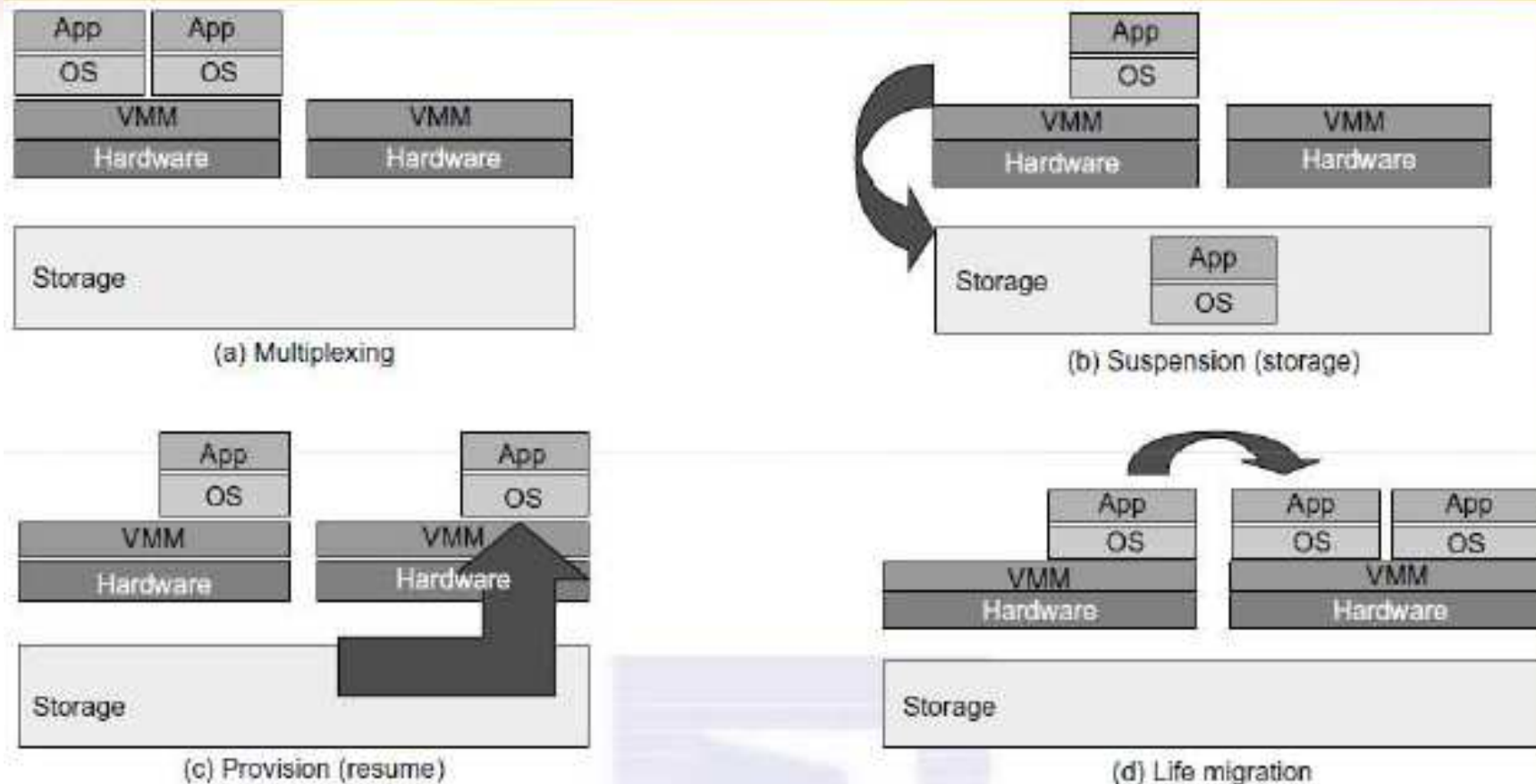


FIGURE 1.13

VM multiplexing, suspension, provision, and migration in a distributed computing environment.

Implementation Levels of Virtualization

- Levels of Virtualization Implementation
- VMM Design Requirements and Providers
- Virtualization Support at the OS Level
 - OS-Level Virtualization
- Middleware Support for Virtualization
 - CUDA & vCUDA

Virtualization Structures/Tools and Mechanisms

- Hypervisor and Xen Architecture
- Binary Translation with Full Virtualization
- Para-Virtualization with Compiler Support

Virtualization of CPU, Memory, and I/O Devices

- Hardware Support for Virtualization
- CPU Virtualization
- Memory Virtualization
- I/O Virtualization
- Virtualization in Multi-Core Processors



Thank You