CS6006 - CLOUD COMPUTING

Module 3 - VIRTUALIZATION

Presented By
Dr. S. Muthurajkumar,
Assistant Professor,
Dept. of CT, MIT Campus,
Anna University, Chennai

VIRTUALIZATION

- Implementation Levels of Virtualization
- Virtualization Structures and Mechanisms
- Full Virtualization
- Para-Virtualization and Hardware-assisted Virtualization
- Virtualization of CPU, Memory and I/O Devices
- Understanding Hypervisors

VIRTUALIZATION

- Virtualization is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources".
- The reincarnation of virtual machines (VMs) presents a great opportunity for parallel, cluster, grid, cloud, and distributed computing.
- Virtualization technology benefits the computer and IT industries by enabling users to share expensive hardware resources by multiplexing VMs on the same set of hardware hosts.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

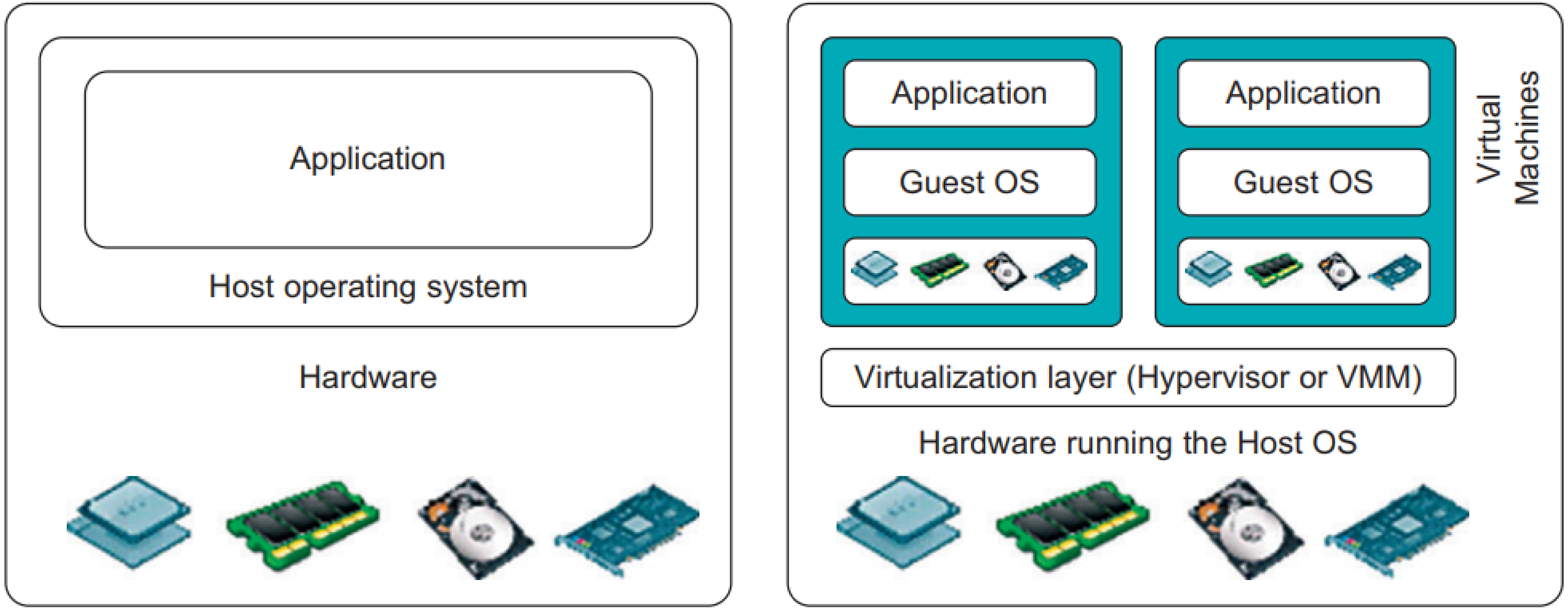
- Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.
- The idea of VMs can be dated back to the 1960s.
- The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility.
- Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers.
- This virtualization technology has been revitalized as the demand for distributed and cloud computing increased sharply in recent years.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- Levels of Virtualization Implementation
  - Instruction Set Architecture Level
  - Hardware Abstraction Level
  - Library Support Level
  - User-Application Level
  - Relative Merits of Different Approaches
- VMM Design Requirements and Providers
- Virtualization Support at the OS Level
- Middleware Support for Virtualization

IMPLEMENTATION LEVELS OF VIRTUALIZATION

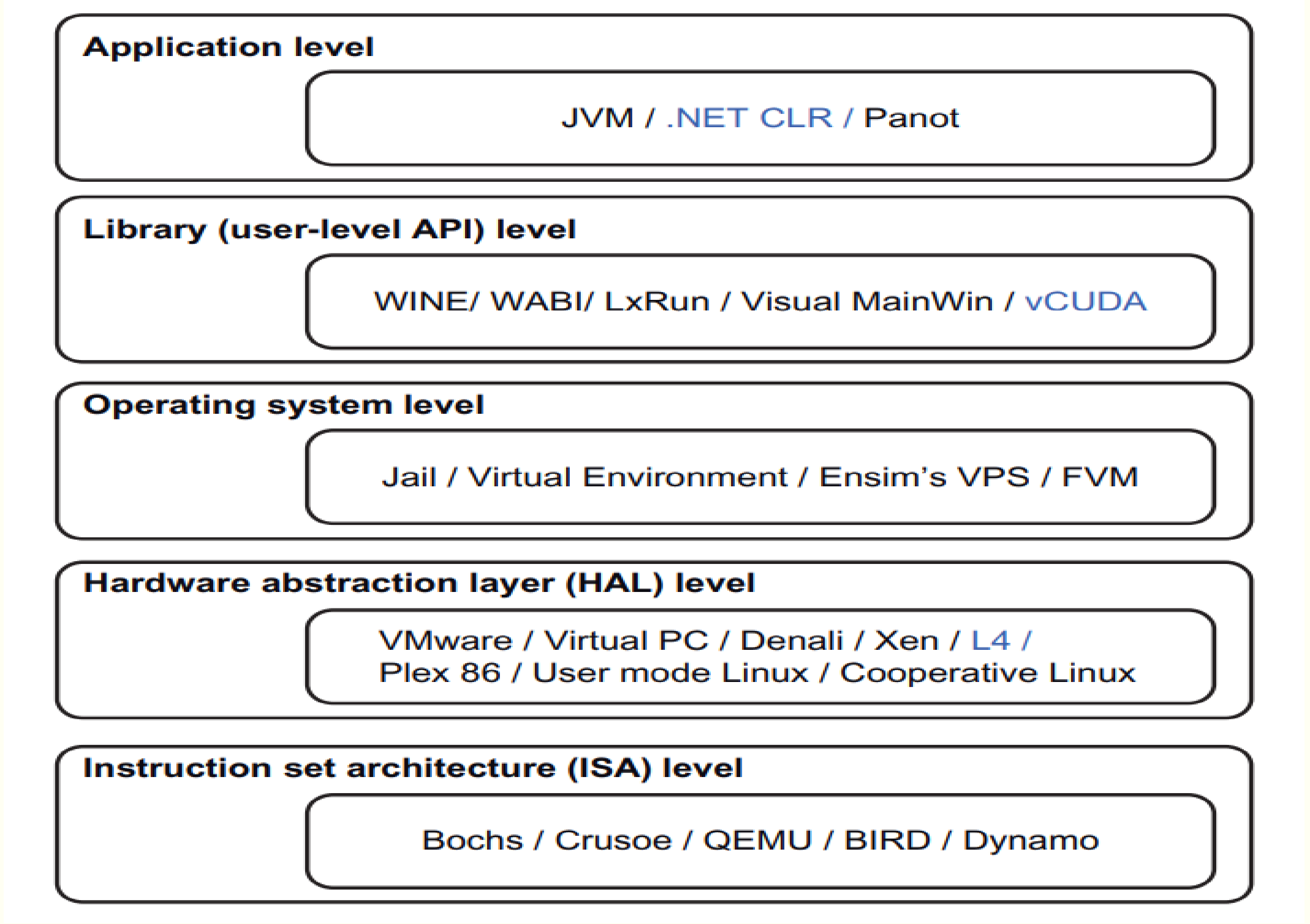
- The architecture of a computer system before and after virtualization, where VMM stands for virtual machine monitor



(a) Traditional computer (b) After virtualization

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- Virtualization ranging from hardware to applications in five abstraction levels


Application level
JVM / .NET CLR / Panot
Library (user-level API) level
WINE/ WABI/ LxRun / Visual MainWin / vCUDA
Operating system level
Jail / Virtual Environment / Ensim's VPS / FVM
Hardware abstraction layer (HAL) level
VMware / Virtual PC / Denali / Xen / L4 /
Plex 86 / User mode Linux / Cooperative Linux
Instruction set architecture (ISA) level
Bochs / Crusoe / QEMU / BIRD / Dynamo

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Instruction Set Architecture Level**
- Virtualization is performed by emulating a given ISA by the ISA of the host machine.
- Instruction set emulation leads to virtual ISAs created on any hardware machine.
- The basic emulation method is through code interpretation.
- An interpreter program interprets the source instructions to target instructions one by one.
- One source instruction may require tens or hundreds of native target instructions to perform its function.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Instruction Set Architecture Level**
- Obviously, this process is relatively slow. For better performance, dynamic binary translation is desired.
- This approach translates basic blocks of dynamic source instructions to target instructions.
- The basic blocks can also be extended to program traces or super blocks to increase translation efficiency.
- Instruction set emulation requires binary translation and optimization.
- A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Hardware Abstraction Level**
- Hardware-level virtualization is performed right on top of the bare hardware.
- On the one hand, this approach generates a virtual hardware environment for a VM.
- On the other hand, the process manages the underlying hardware through virtualization.
- The idea is to virtualize a computer’s resources, such as its processors, memory, and I/O devices.
- The intention is to upgrade the hardware utilization rate by multiple users concurrently.
- The idea was implemented in the IBM VM/370 in the 1960s.
- More recently, the Xen hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS applications.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Operating System Level**
- This refers to an 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.
- OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.
- It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **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.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **User-Application Level**
- Virtualization at the application level virtualizes an application as a VM.
- On a traditional OS, an application often runs as a process.
- Therefore, application-level virtualization is also known as process-level virtualization.
- The most popular approach is to deploy high level language (HLL) VMs.
- In this scenario, the virtualization layer sits as an application program on top of the operating system, and the layer 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.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Relative Merits of Different Approaches**
- The column headings correspond to four technical merits.
- “Higher Performance” and “Application Flexibility” are self-explanatory. “Implementation Complexity” implies the cost to implement that particular virtualization level.
- “Application Isolation” refers to the effort required to isolate resources committed to different VMs.
- Each row corresponds to a particular level of virtualization.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **VMM Design Requirements and Providers**
- As mentioned earlier, hardware-level virtualization inserts a layer between real hardware and traditional operating systems.
- This layer is commonly called the Virtual Machine Monitor (VMM) and it manages the hardware resources of a computing system.
- Each time programs access the hardware the VMM captures the process.
- In this sense, the VMM acts as a traditional OS.
- One hardware component, such as the CPU, can be virtualized as several virtual copies.
- Therefore, several traditional operating systems which are the same or different can sit on the same set of hardware simultaneously.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Virtualization Support at the OS Level**
- With the help of VM technology, a new computing mode known as cloud computing is emerging.
- Cloud computing is transforming the computing landscape by shifting the hardware and staffing costs of managing a computational center to third parties, just like banks.

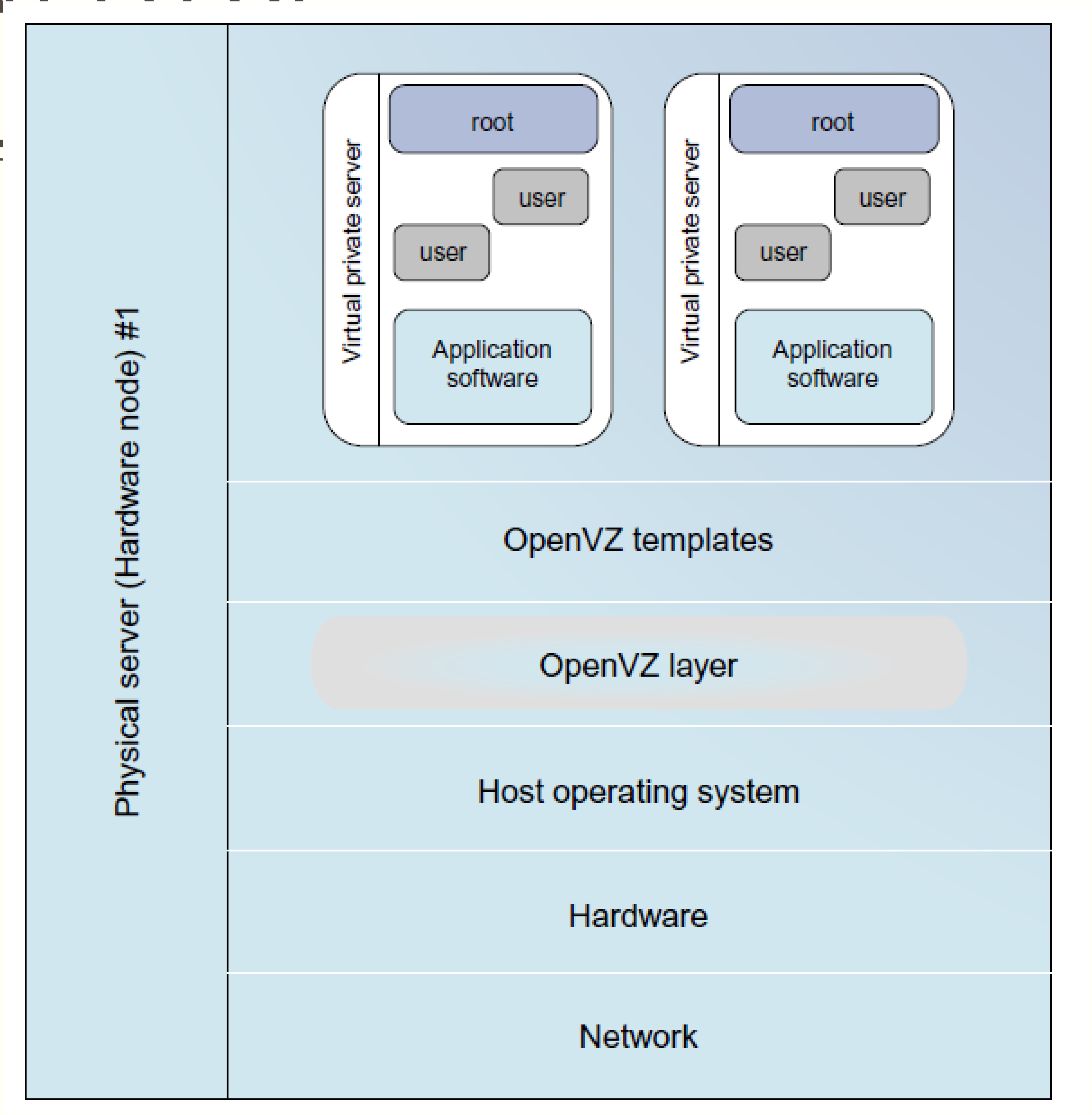
IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Virtualization Support at the OS Level**
- **Why OS-Level Virtualization?**
- As mentioned earlier, it is slow to initialize a hardware-level VM because each VM creates its own image from scratch.
- Operating system 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.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Virtualization Support at the OS Level**
- **Advantages of OS Extensions**

The OpenVZ virtualization layer inside the host OS, which provides some OS images to create VMs quickly.



IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Virtualization Support at the OS Level**
- **Disadvantages of OS Extensions**
- The main disadvantage of OS extensions is that all the VMs at operating system level on a single container must have the same kind of guest operating system.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

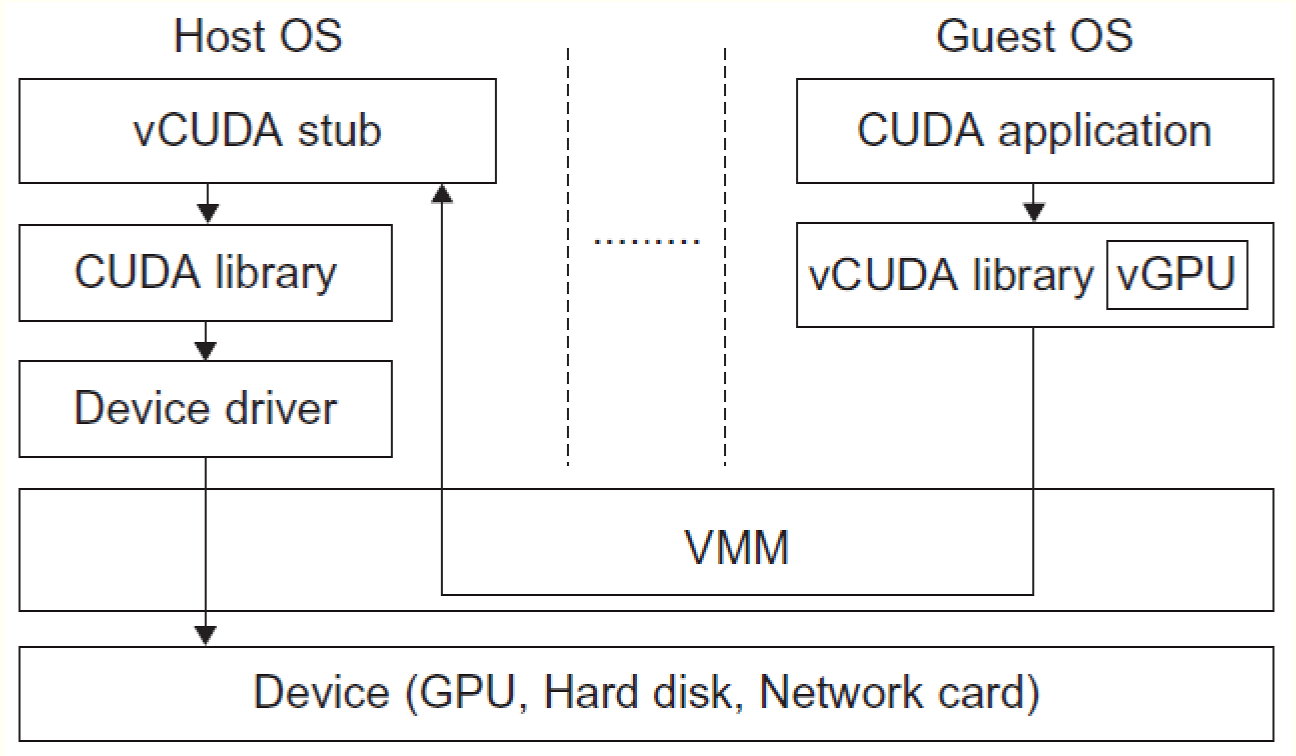
- **Virtualization Support at the OS Level**
- **Virtualization on Linux or Windows Platforms**
- By far, most reported OS-level virtualization systems are Linux-based.
- Virtualization support on the Windows-based platform is still in the research stage.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- **Middleware Support for Virtualization**
- Library-level virtualization is also known as user-level Application Binary Interface (ABI) or API emulation.
- This type of virtualization can create execution environments for running alien programs on a platform rather than creating a VM to run the entire operating system.
- API call interception and remapping are the key functions performed.
- This section provides an overview of several library-level virtualization systems: namely the Windows Application Binary Interface (WABI), lxrun, WINE, Visual MainWin, and vCUDA.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

- Basic concept of the vCUDA architecture



VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- Depending on the position of the virtualization layer, there are several classes of VM architectures, namely the hypervisor architecture, para-virtualization, and host-based virtualization.
- The hypervisor is also known as the VMM (Virtual Machine Monitor). They both perform the same virtualization operations.
- Hypervisor and Xen Architecture
  - The Xen Architecture
- Binary Translation with Full Virtualization
  - Full Virtualization
  - Binary Translation of Guest OS Requests Using a VMM
  - Host-Based Virtualization
- Para-Virtualization with Compiler Support

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Hypervisor and Xen Architecture**
- The hypervisor supports hardware-level virtualization on bare metal devices like CPU, memory, disk and network interfaces.
- The hypervisor software sits directly between the physical hardware and its OS.
- This virtualization layer is referred to as either the VMM or the hypervisor.
- The hypervisor provides hypercalls for the guest OSes and applications.
- Depending on the functionality, a hypervisor can assume a micro-kernel architecture like the Microsoft Hyper-V. Or it can assume a monolithic hypervisor architecture like the VMware ESX for server virtualization.

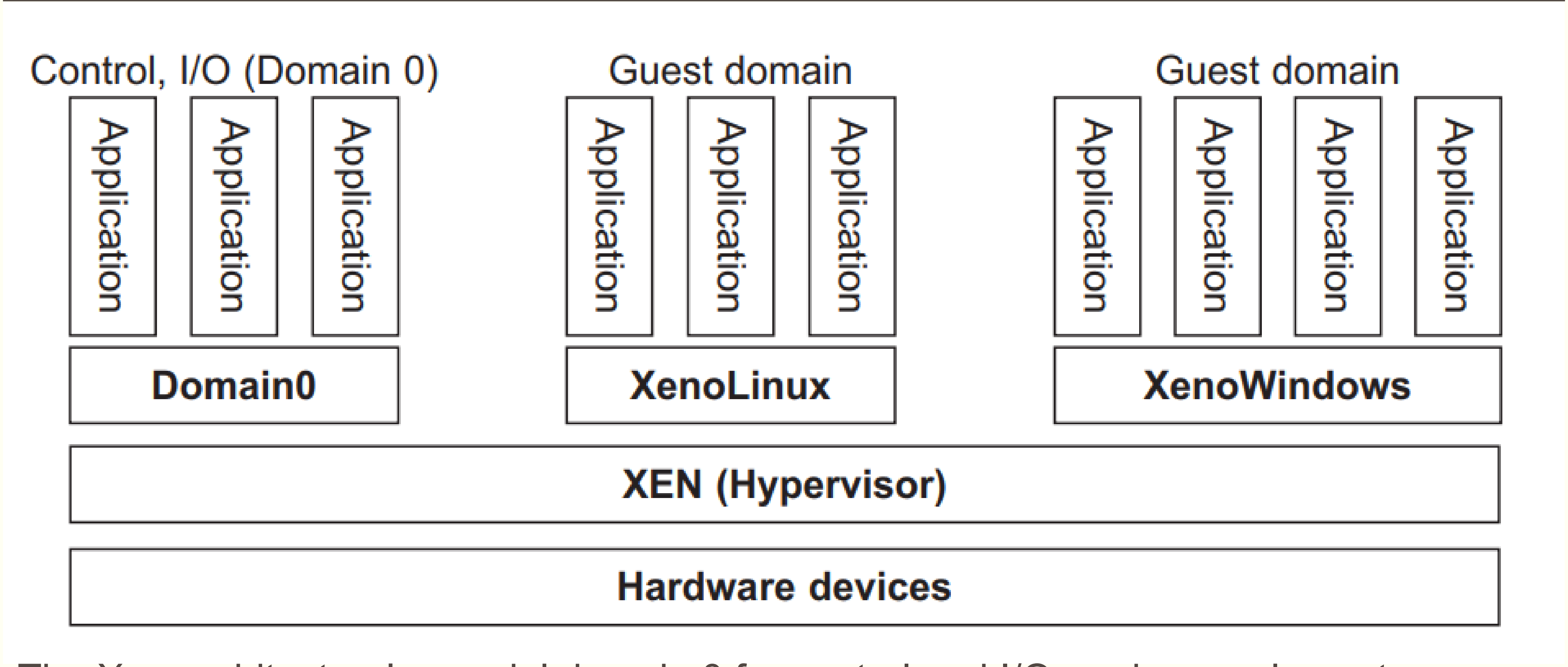
VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Hypervisor and Xen Architecture**
  - **The Xen Architecture**
- Xen is an open source hypervisor program developed by Cambridge University.
- Xen is a microkernel hypervisor, which separates the policy from the mechanism.
- The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0.
- Xen does not include any device drivers natively. It just provides a mechanism by which a guest OS can have direct access to the physical devices. As a result, the size of the Xen hypervisor is kept rather small.
- Xen provides a virtual environment located between the hardware and the OS. A number of vendors are in the process of developing commercial Xen hypervisors, among them are Citrix XenServer and Oracle VM.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Hypervisor and Xen Architecture**
  - **The Xen Architecture**
- The core components of a Xen system are the hypervisor, kernel, and applications.
- The organization of the three components is important. Like other virtualization systems, many guest Oses can run on top of the hypervisor.
- However, not all guest OSes are created equal, and one in particular controls the others.
- The guest OS, which has control ability, is called Domain 0, and the others are called Domain U. Domain 0 is a privileged guest OS of Xen. It is 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).

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS



The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Binary Translation with Full Virtualization**
- Depending on implementation technologies, hardware virtualization can be classified into two categories: full virtualization and host-based virtualization.
- Full virtualization does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, nonvirtualizable instructions.
- The guest OSes and their applications consist of noncritical and critical instructions. In a 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. These two classes of VM architecture are introduced next.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **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. Both the hypervisor and VMM approaches are considered full virtualization.
- Binary Translation of Guest OS Requests Using a VMM
- This approach was implemented by VMware and many other software companies. 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. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions.

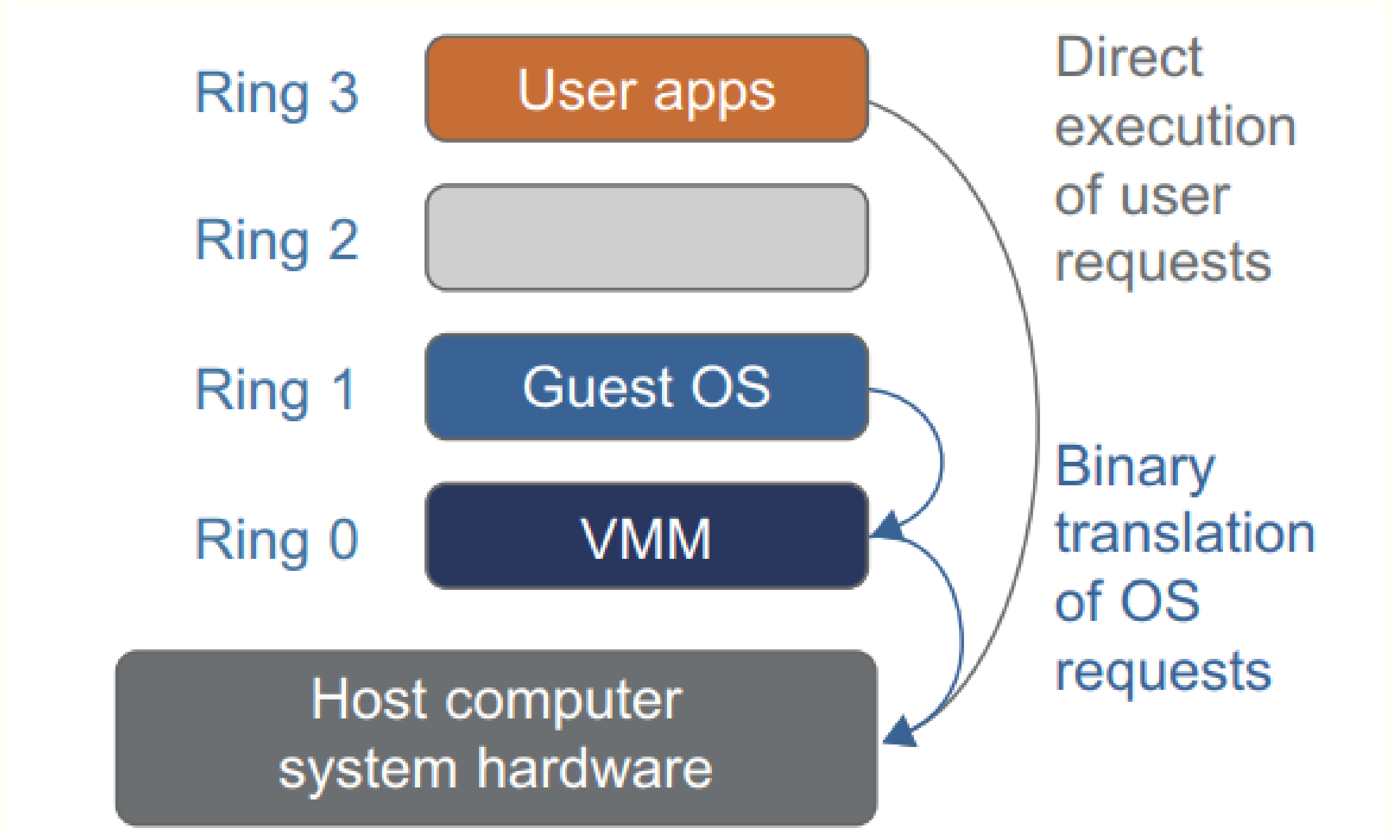
VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Full Virtualization**
- The method used in this emulation is called binary translation. Therefore, full virtualization combines binary translation and direct execution.
- The guest OS is completely decoupled from the underlying hardware. Consequently, the guest OS is unaware that it is being virtualized.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Host-Based Virtualization**
- An alternative VM architecture is to 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.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

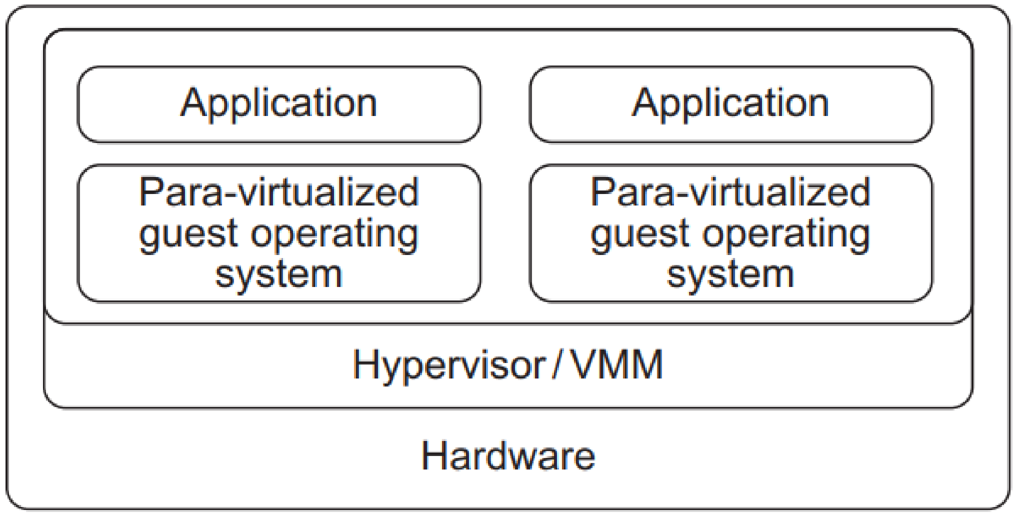

Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

Indirect execution of complex instructions via binary translation of guest OS requests using the VMM plus direct execution of simple instructions on the same host.

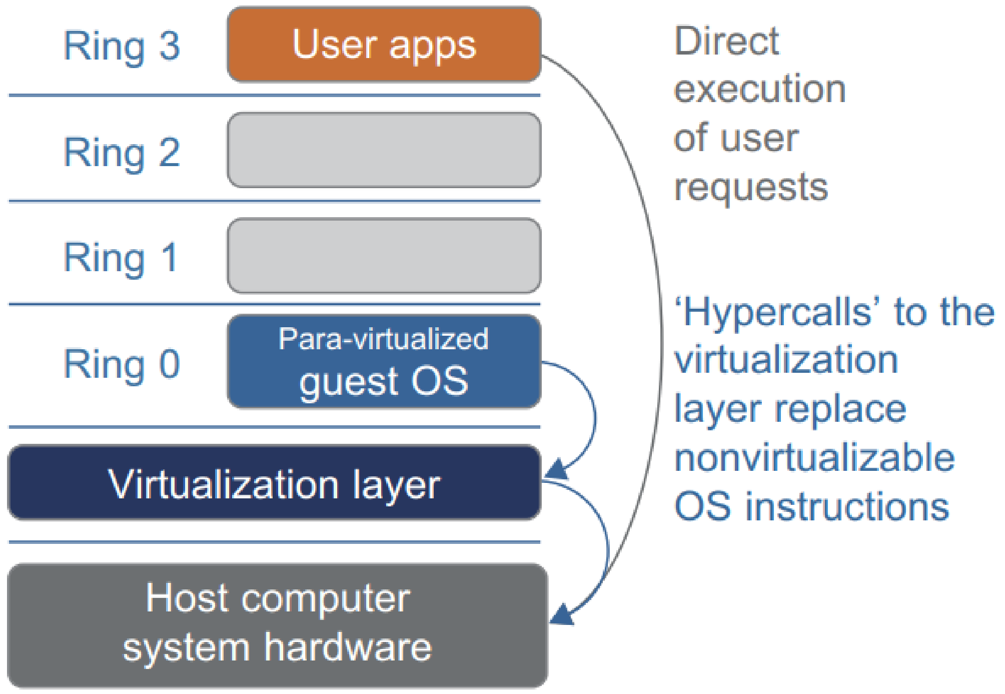
VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **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. No one wants to use a VM if it is much slower than using a physical machine.
- The virtualization layer can be inserted at different positions in a machine software stack.
- However, para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS


Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor / VMM
Hardware

Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization process


Ring 3
User apps
Ring 2
Ring 1
Ring 0
Para-virtualized guest OS
Virtualization layer
Host computer system hardware
Direct execution of user requests
'Hypercalls' to the virtualization layer replace nonvirtualizable OS instructions

The use of a para-virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizable OS instructions by hypercalls.

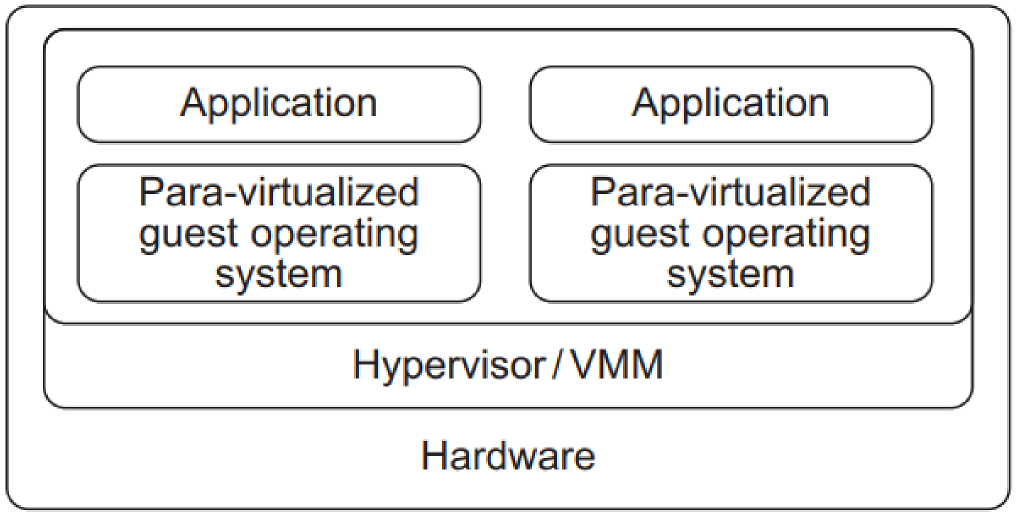
VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Para-Virtualization with Compiler Support**
- Para-Virtualization Architecture
- KVM (Kernel-Based VM)
- Para-Virtualization with Compiler Support

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

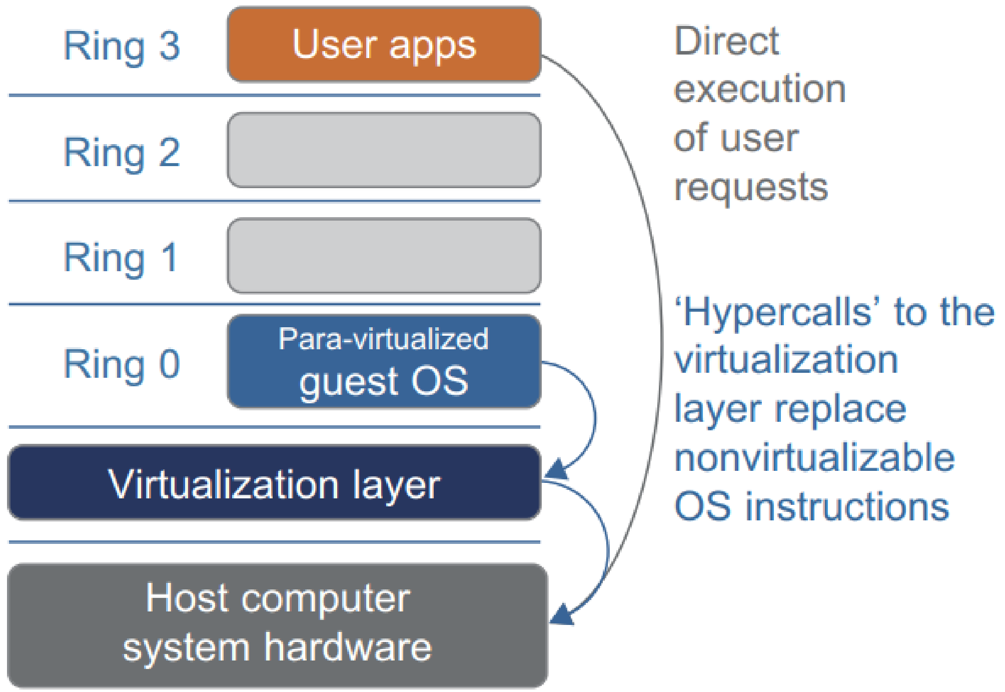
- **Para-Virtualization Architecture**
- When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS.
- According to the x86 ring definition, the virtualization layer should also be installed at Ring 0. Different instructions at Ring 0 may cause some problems.
- Para-virtualization replaces nonvirtualizable instructions with hypercalls that communicate directly with the hypervisor or VMM.
- However, when the guest OS kernel is modified for virtualization, it can no longer run on the hardware directly.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS



Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor / VMM
Hardware

Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization process



Ring 3
User apps
Ring 2
Ring 1
Ring 0
Para-virtualized guest OS
Virtualization layer
Host computer system hardware
Direct execution of user requests
'Hypercalls' to the virtualization layer replace nonvirtualizable OS instructions

The use of a para-virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizable OS instructions by hypercalls.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **KVM (Kernel-Based VM)**
- This is a Linux para-virtualization system a part of the Linux version 2.6.20 kernel.
- Memory management and scheduling activities are carried out by the existing Linux kernel.
- The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine.
- KVM is a hardware-assisted para-virtualization tool, which improves performance and supports unmodified guest OSes such as Windows, Linux, Solaris, and other UNIX variants.

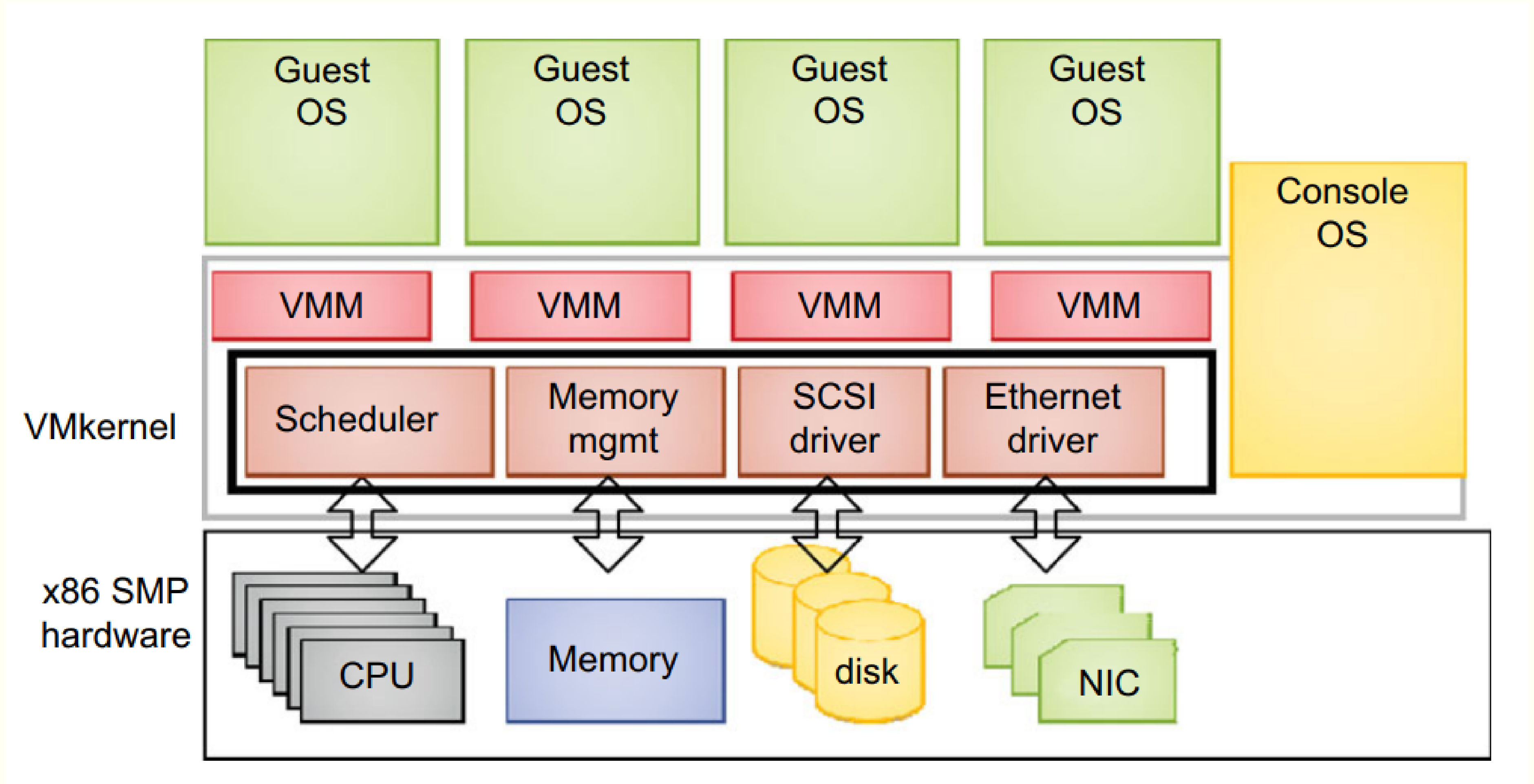
VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Para-Virtualization with Compiler Support**
- Unlike the full virtualization architecture which intercepts and emulates privileged and sensitive instructions at runtime, para-virtualization handles these instructions at compile time.
- The guest OS kernel is modified to replace the privileged and sensitive instructions with hypercalls to the hypervisor or VMM.
- Xen assumes such a para-virtualization architecture.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS

- **Para-Virtualization with Compiler Support**
- The guest OS running in a guest domain may run at Ring 1 instead of at Ring 0.
- This implies that the guest OS may not be able to execute some privileged and sensitive instructions.
- The privileged instructions are implemented by hypercalls to the hypervisor.
- After replacing the instructions with hypercalls, the modified guest OS emulates the behavior of the original guest OS.
- On an UNIX system, a system call involves an interrupt or service routine.
- The hypercalls apply a dedicated service routine in Xen.

VIRTUALIZATION STRUCTURES/TOOLS AND MECHANISMS



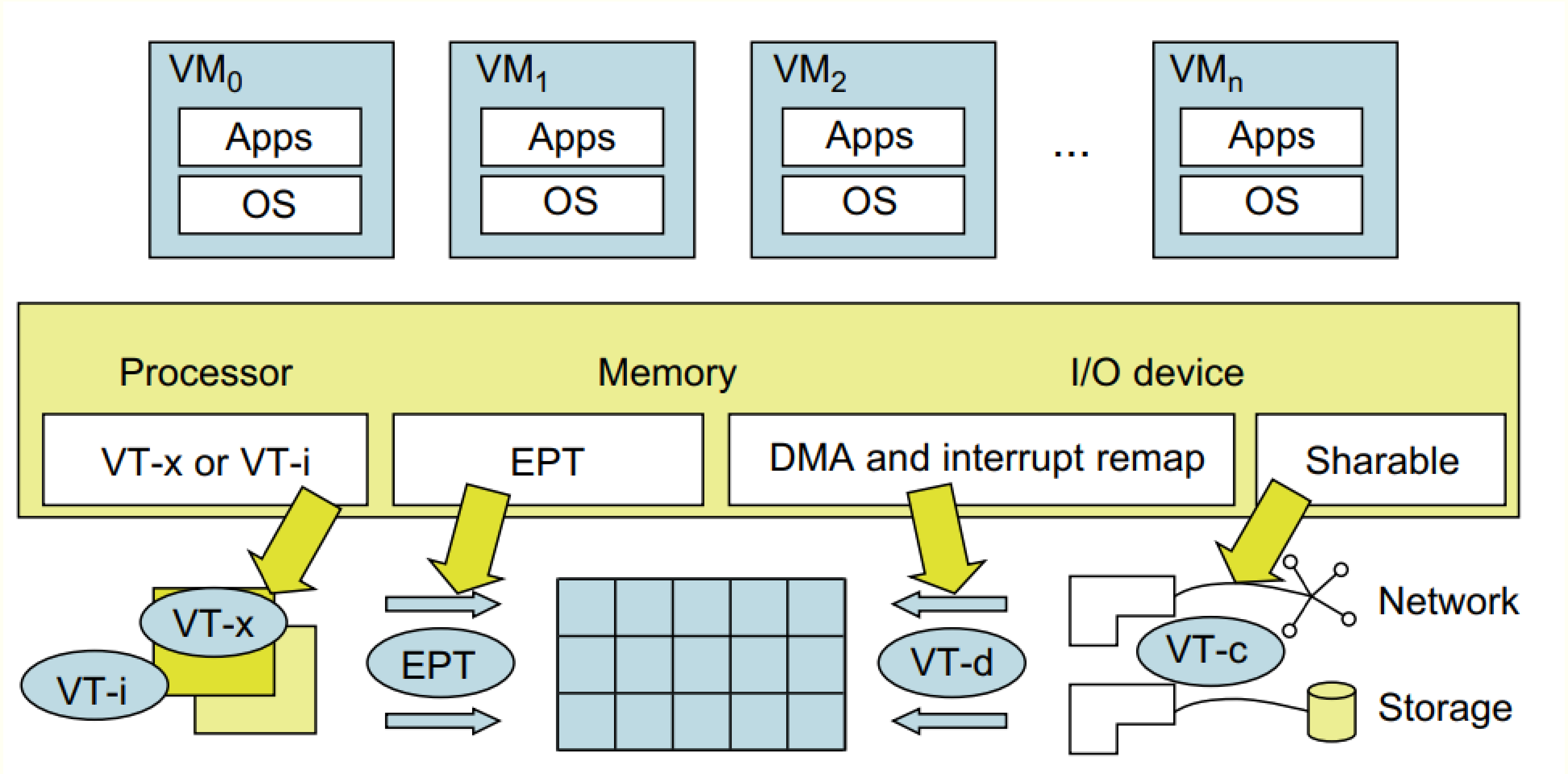
VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

- To support virtualization, processors such as the x86 employ a special running mode and instructions, known as hardware-assisted virtualization.
- 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.
- To save processor states, mode switching is completed by hardware.
- For the x86 architecture, Intel and AMD have proprietary technologies for hardware-assisted virtualization.
- Hardware Support for Virtualization
- CPU Virtualization
  - Hardware-Assisted CPU Virtualization
- Memory Virtualization
- I/O Virtualization
- Virtualization in Multi-Core Processors
  - Physical versus Virtual Processor Cores
  - Virtual Hierarchy

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

- **Hardware Support for Virtualization**
- Modern operating systems and processors permit multiple processes to run simultaneously.
- If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash.
- Therefore, 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.
- In a virtualized environment, it is more difficult to make OSes and applications run correctly because there are more layers in the machine stack.

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES



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

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

- **CPU Virtualization**
- A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode.
- Thus, 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.

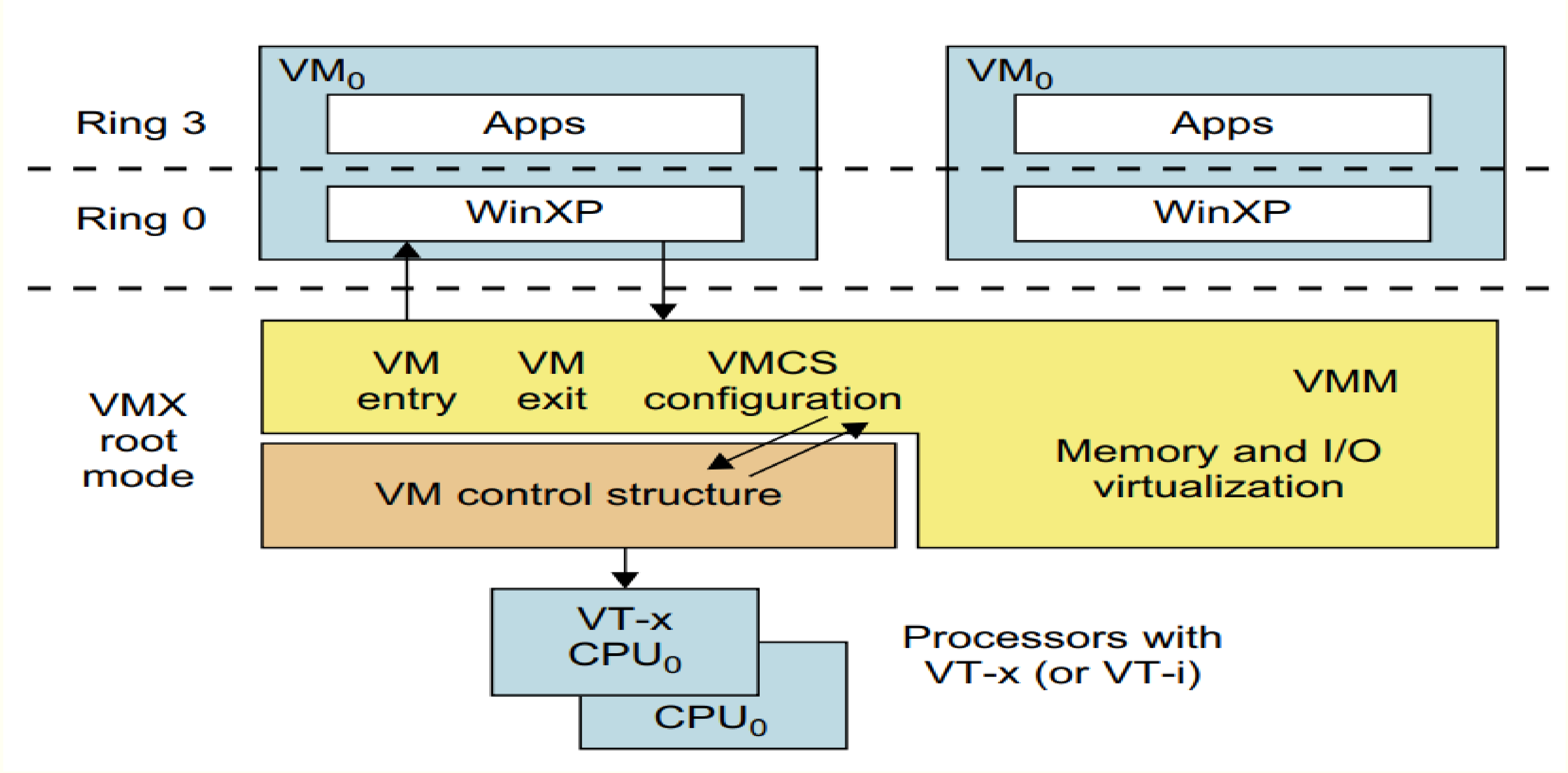
VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

- **CPU Virtualization**
- 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.

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

- **CPU Virtualization**
- **Hardware-Assisted CPU Virtualization**
- This technique attempts to simplify virtualization because full or paravirtualization is complicated.
- 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.

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

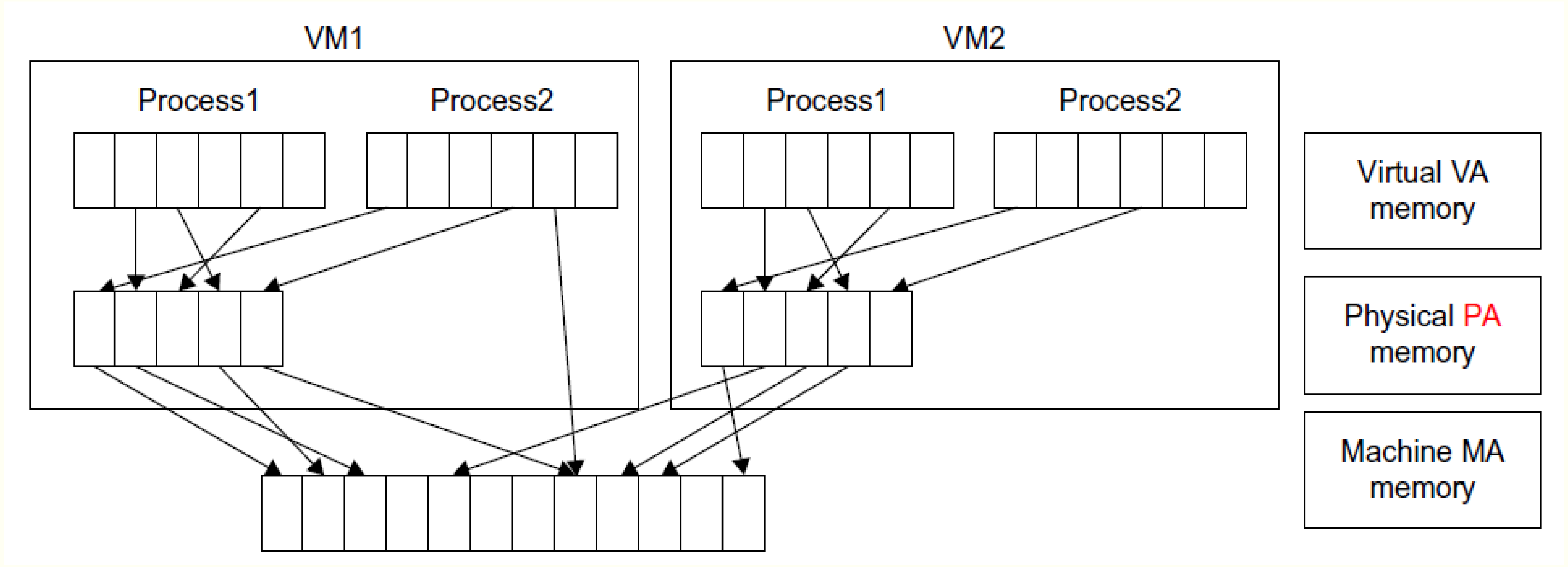


Intel hardware-assisted CPU virtualization

MEMORY VIRTUALIZATION

- Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems.
- In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory.
- All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance.
- However, in a 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.

MEMORY VIRTUALIZATION

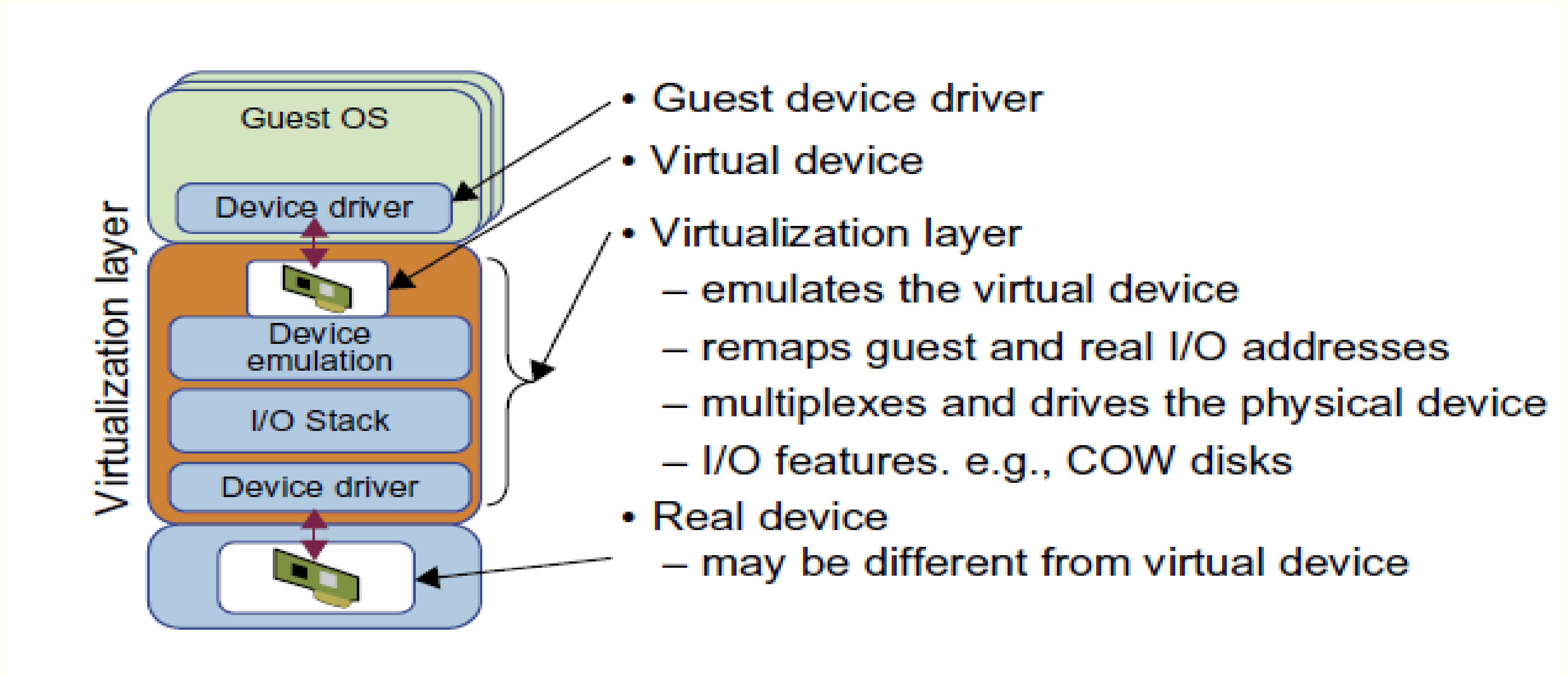


Two-level memory mapping procedure

I/O VIRTUALIZATION

- I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware.
- At the time of this writing, there are three ways to implement I/O virtualization: full device emulation, para-virtualization, and direct I/O. Full device emulation is the first approach for I/O virtualization.
- Generally, this approach emulates well-known, real-world devices.

I/O VIRTUALIZATION



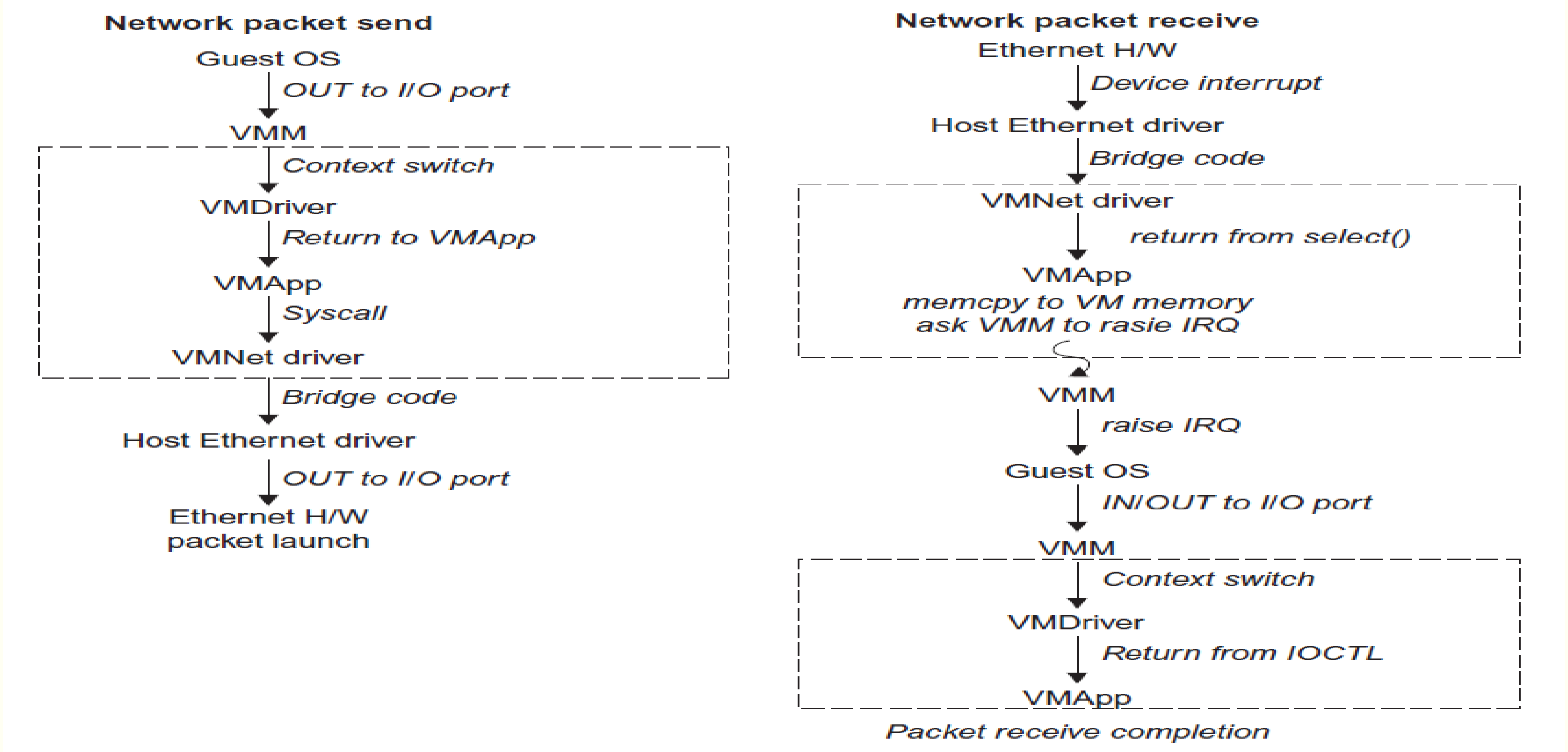
- Guest device driver
- Virtual device
- Virtualization layer
  - emulates the virtual device
  - remaps guest and real I/O addresses
  - multiplexes and drives the physical device
  - I/O features. e.g., COW disks
- Real device
  - may be different from virtual device

Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices into the virtual devices for the guest device driver to use

I/O VIRTUALIZATION

- All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, 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.

I/O VIRTUALIZATION



Functional blocks involved in sending and receiving network packets

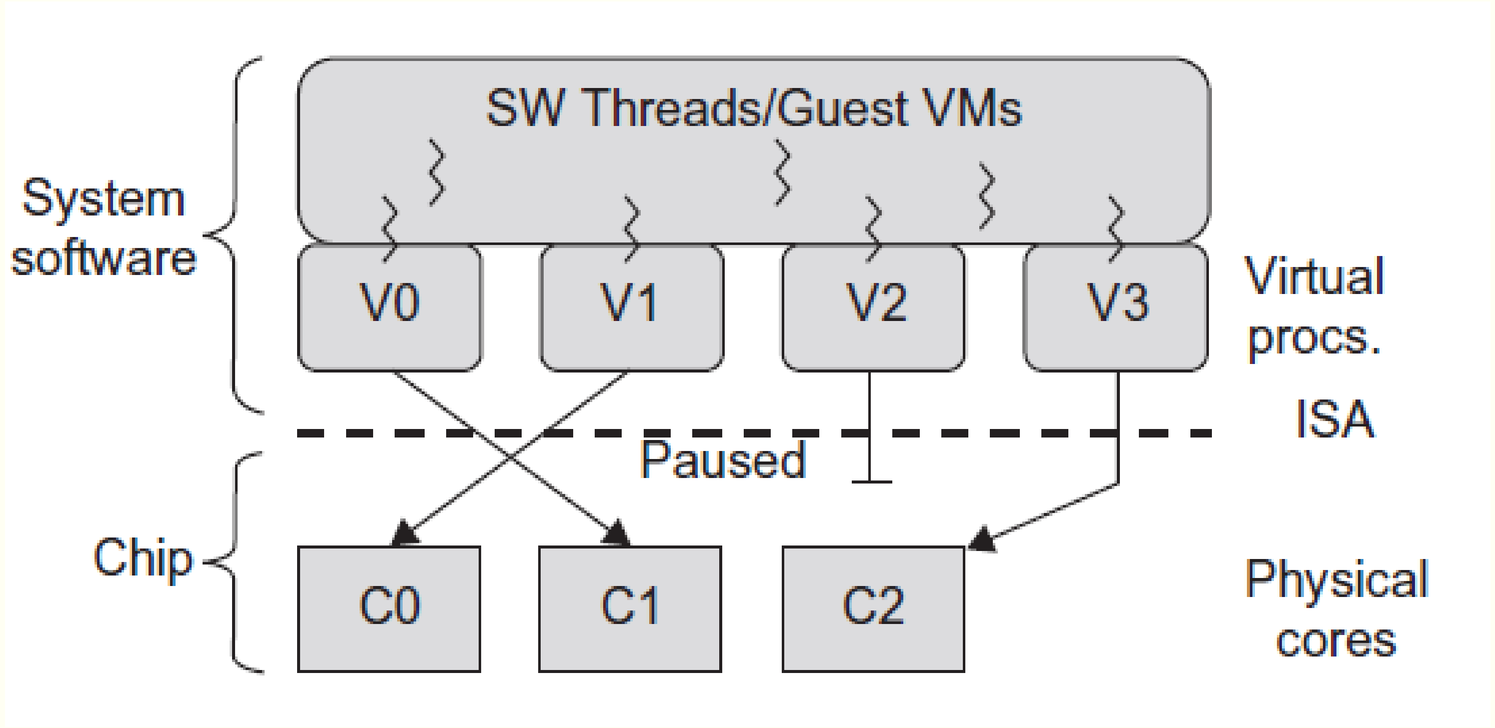
VIRTUALIZATION IN MULTI-CORE PROCESSORS

- Virtualizing a multi-core processor is relatively more complicated than virtualizing a uni-core processor.
- Though multicore processors are claimed to have higher performance by integrating multiple processor cores in a single chip, muti-core virtualization has raised some new challenges to computer architects, compiler constructors, system designers, and application programmers.
- There are mainly two difficulties: Application programs must be parallelized to use all cores fully, and software must explicitly assign tasks to the cores, which is a very complex problem.
- Physical versus Virtual Processor Cores
- Virtual Hierarchy

VIRTUALIZATION IN MULTI-CORE PROCESSORS

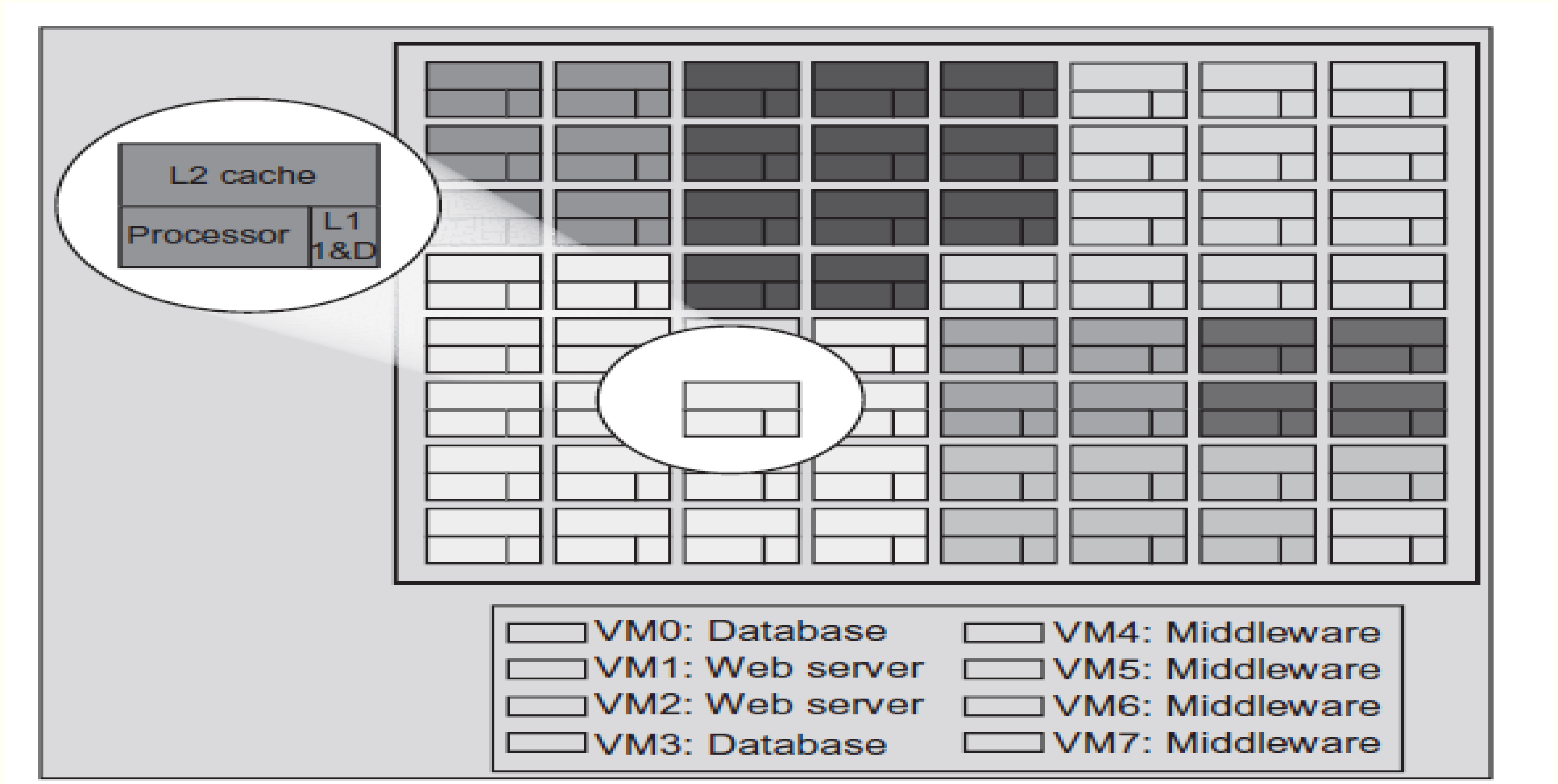
- **Physical versus Virtual Processor Cores**
- This technique alleviates the burden and inefficiency of managing hardware resources by software.
- It is located under the ISA and remains unmodified by the operating system or VMM (hypervisor).
- **Virtual Hierarchy**
- The emerging many-core chip multiprocessors (CMPs) provides a new computing landscape.
- Instead of supporting time-sharing jobs on one or a few cores, we can use the abundant cores in a space-sharing, where single-threaded or multithreaded jobs are simultaneously assigned to separate groups of cores for long time intervals.

VIRTUALIZATION IN MULTI-CORE PROCESSORS



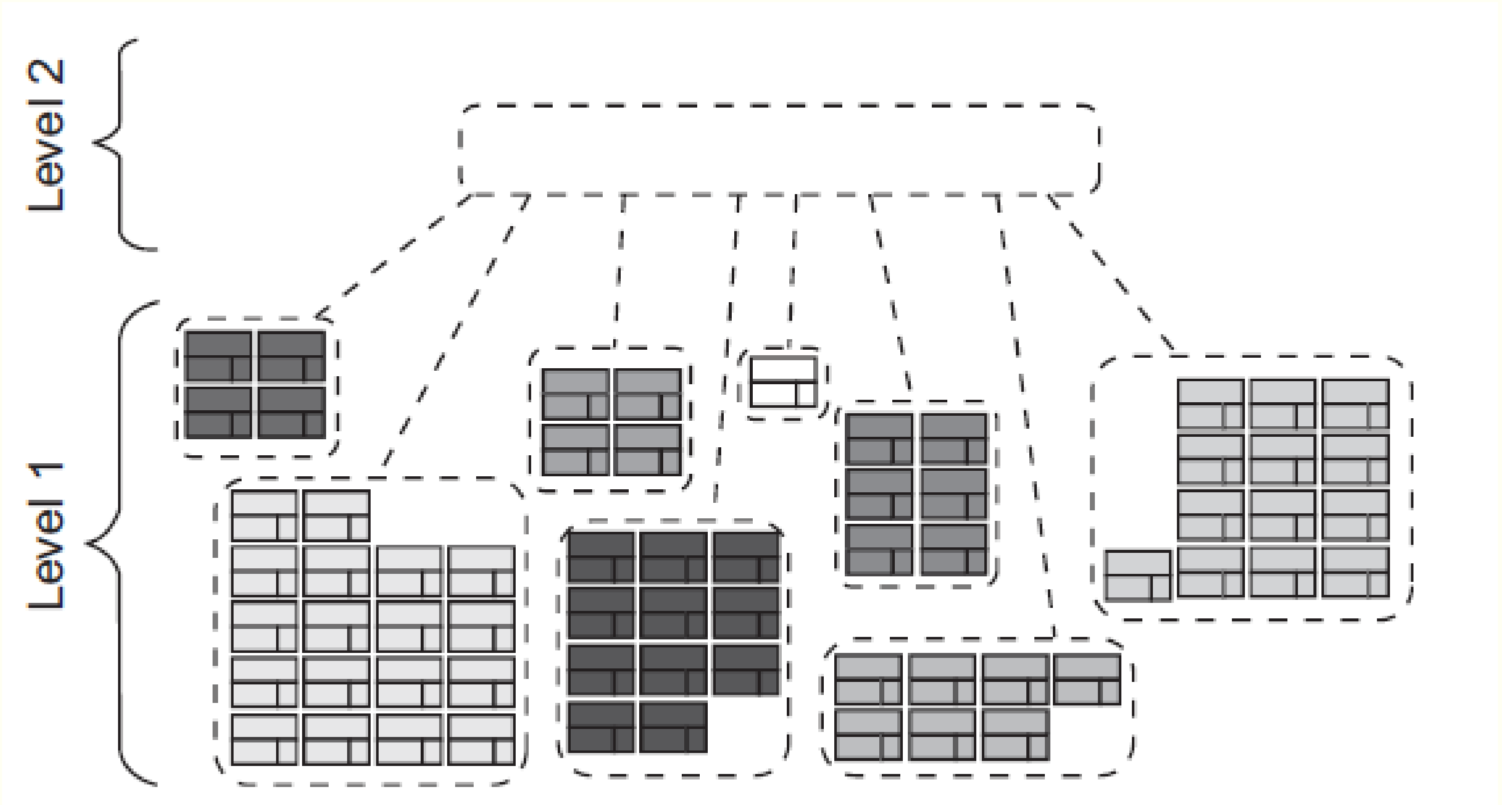
Multicore virtualization method that exposes four VCPUs to the software, when only three cores are actually present.

VIRTUALIZATION IN MULTI-CORE PROCESSORS



(a) Mapping of VMs into adjacent cores

VIRTUALIZATION IN MULTI-CORE PROCESSORS



(b) Multiple virtual clusters assigned to various workloads

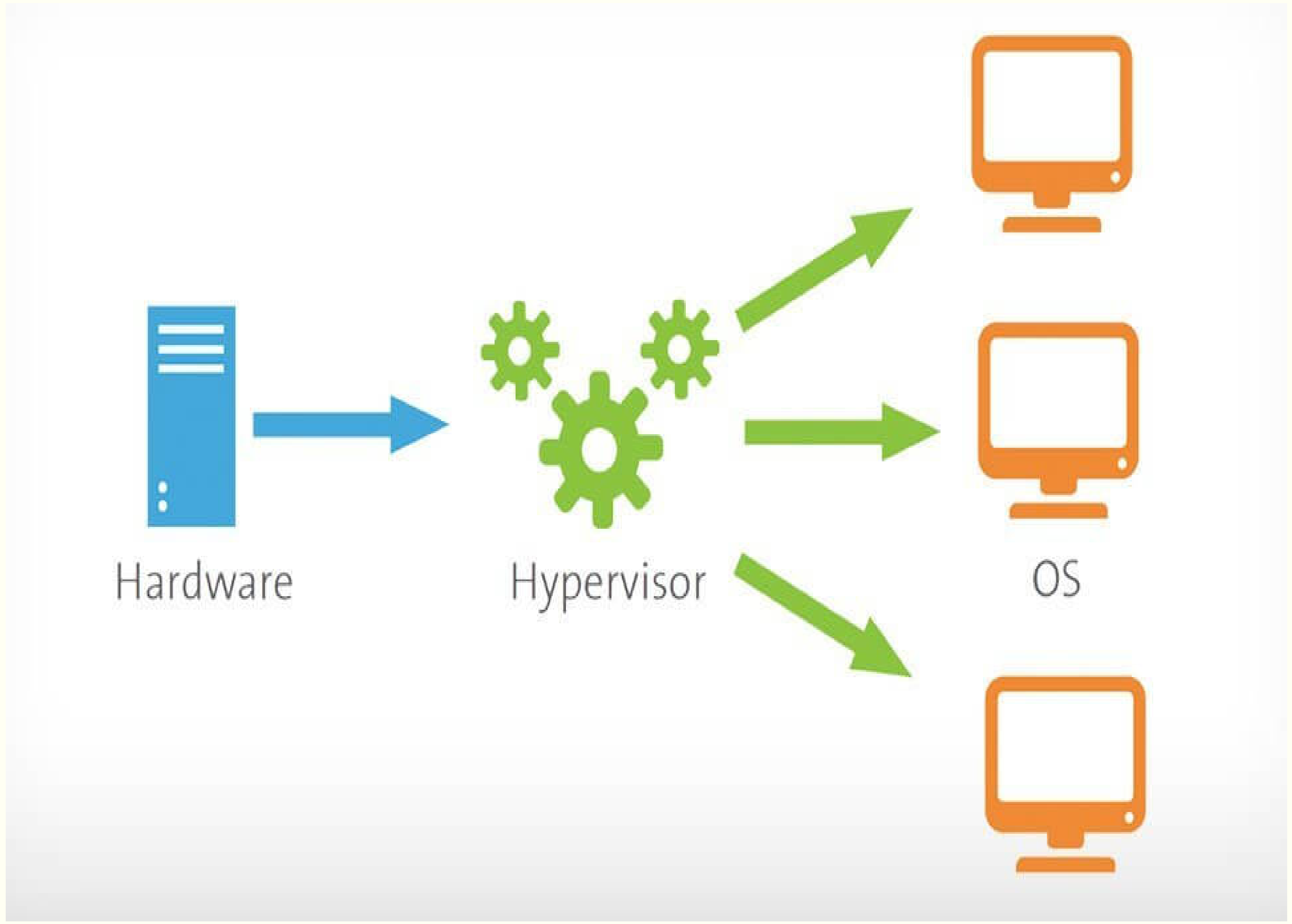
Chip Multi-processors (CMP) server consolidation by space-sharing of VMs into many cores forming multiple virtual clusters to execute various workloads.

UNDERSTANDING HYPERVISORS

- A hypervisor, also known as a virtual machine monitor or VMM.
- The hypervisor is a piece of software that allows us to build and run virtual machines which are abbreviated as VMs.
- A hypervisor allows a single host computer to support multiple virtual machines (VMs) by sharing resources including memory and processing.
- The program which provides partitioning, isolation, or abstraction is called a virtualization hypervisor.
- The hypervisor is a hardware virtualization technique that allows multiple guest operating systems (OS) to run on a single host system at the same time.
- A hypervisor is sometimes also called a virtual machine manager(VMM).

UNDERSTANDING HYPERVISORS

- **Types of Hypervisor**
- TYPE-1 Hypervisor
- TYPE-2 Hypervisor



UNDERSTANDING HYPERVISORS

- **TYPE-1 Hypervisor**
- The hypervisor runs directly on the underlying host system.
- It is also known as a “Native Hypervisor” or “Bare metal hypervisor”.
- It does not require any base server operating system.
- It has direct access to hardware resources.
- Examples of Type 1 hypervisors include VMware ESXi, Citrix XenServer, and Microsoft Hyper-V hypervisor.

UNDERSTANDING HYPERVISORS

- **TYPE-2 Hypervisor**
- A Host operating system runs on the underlying host system.
- It is also known as ‘Hosted Hypervisor”.
- Such kind of hypervisors doesn’t run directly over the underlying hardware rather they run as an application in a Host system(physical machine).
- Basically, the software is installed on an operating system.
- Hypervisor asks the operating system to make hardware calls.
- An example of a Type 2 hypervisor includes VMware Player or Parallels Desktop. Hosted hypervisors are often found on endpoints like PCs.
- The type-2 hypervisor is very useful for engineers, and security analysts (for checking malware, or malicious source code and newly developed applications).

UNDERSTANDING HYPERVISORS

- **Benefits of hypervisors**
- Flexibility
- Scalability
- Usability
- Availability
- Reliability
- Efficiency
- Reliable support

REFERENCES

1. Kai Hwang, Geoffrey C Fox and Jack G Dongarra, "Distributed and Cloud Computing, From Parallel Processing to the Internet of Things", Morgan Kaufmann Publishers, 2012.
2. Barrie Sosinky,"Cloud Computing Bible”, Wiley Publishing Inc,2011
3. Buyya R., Broberg J. and Goscinski A., "Cloud Computing: Principles and Paradigm”, First Edition, John Wiley & Sons, 2011.
4. Rajkumar Buyya, Christian Vecchiola, S. ThamaraiSelvi,"Mastering the Cloud Computing”, Morgan Kaufmann,2013
5. John W. Rittinghouse and James F. Ransome, "Cloud Computing: Implementation "Management, and Security", CRC Press, 2016.
6. David Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes”, IEEE Cloud Computing, Volume: 1 , Issue: 3 , 2014.
7. VMware (white paper),"Understanding Full Virtualization, Paravirtualization, and Hardware Assist ":www.vmware.com/files/pdf/VMware_paravirtualization.pdf.

Thank You…