Amirkabir University of Technology
(Tehran Polytechnic)

Cloud Computing

Virtualization
(Introduction, Hardware Virtualization)

Seyyed Ahmad Javadi

sajavadi@aut.ac.ir

Spring 2020

Based on slide deck of Dr. Abrishami (Ferdowsi Uni. of Mashhad)

Main reference: Mastering Cloud Computing By Rajkumar Buyya.

Part 1

Introduction

- Virtualization is a large umbrella of technologies and concepts.
  - Provide ***an abstract environment*** to run applications.
    - Virtual hardware
    - Virtual operating system

- The term virtualization is often **synonymous** with hardware virtualization.
  - Plays a fundamental role in efficiently delivering ***Infrastructure-as-a-Service (IaaS)*** solutions for cloud computing.

Introduction (Cont.)

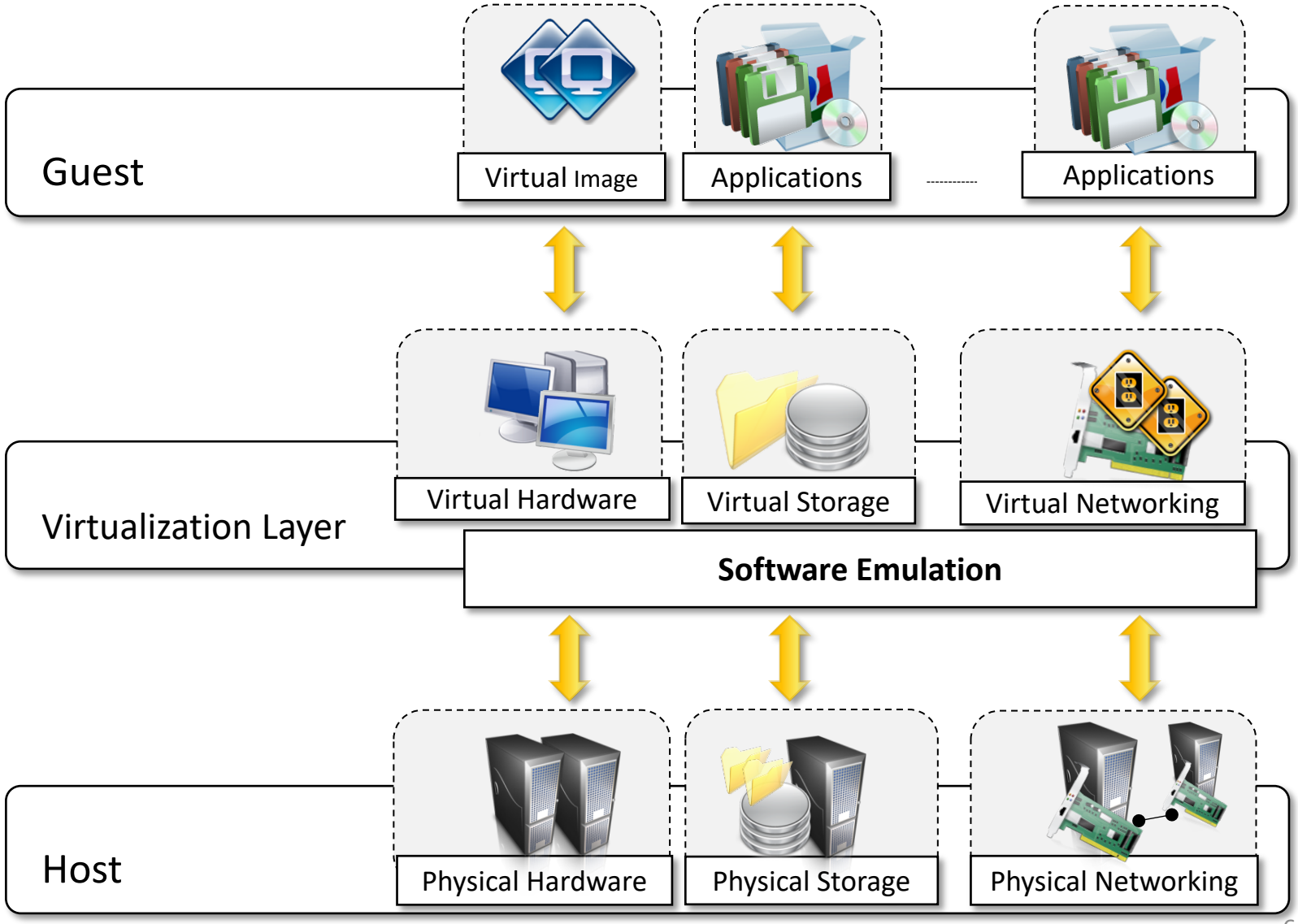
- Virtualization technologies have a long trail in the history of computer science.
- In many flavors by providing Virtual Environments (VE) at the levels below:
  - Operating system level
  - Programming language level
  - Application level

- Virtualization technologies provide a VE for not only **executing applications** but also for **storage**, **memory**, and **networking**.

Major components of a virtualized environment

- **Guest**
  - The system component that interacts with the virtualization layer rather than with the host, as would normally happen.
- **Host**
  - The original environment where the guest is supposed to be managed.
- **Virtualization layer**
  - Recreate the same or a different environment where the guest will operate.

Introduction (Cont.)



Advantages

- Increased Security
- Managed Execution
  - Sharing
  - Aggregation
  - Emulation
  - Isolation
  - Performance tuning
  - Virtual machine migration
- Portability

Advantages

Increased Security

- The ability to control the execution of a guest in a ***completely transparent manner*** opens new possibilities for delivering a secure, controlled execution environment.
- All the operations of the guest are generally ***performed against the VM***, which then translates and applies them to the host.
- This level of ***indirection*** allows the VM manager ***to control and filter the activity of the guest***, thus preventing **some harmful** operations from being performed.
- Resources exposed by the host can then be ***hidden or simply protected from the guest.***

Advantages

Managed Execution

- **Sharing**
  - Virtualization allows the creation of a separate computing environments within the same host.
  - In this way it is possible to **fully exploit the capabilities** of a powerful host, **which would otherwise be underutilized**.
- **Aggregation**
  - A group of separate hosts can be tied together and represented to guests as a single virtual host.

Advantages

Managed Execution (Cont.)

- **Emulation**
  - A completely different environment with respect to the host can be emulated, thus allowing the execution of guest programs requiring specific characteristics that are not present in the physical host.

- **Isolation**:
  - Virtualization allows providing guests—whether they are operating systems, applications, or other entities—with a **completely separate environment**, in which **they are executed**.

Advantages

Managed Execution (Cont.)

- **Performance tuning**
  - It becomes easier to control the performance of the guest by finely tuning the properties of the resources exposed through the virtual environment.
  - This capability provides a means to effectively implement a quality-of-service (QoS) infrastructure that more easily fulfills the service-level agreement (SLA) established for the guest.

Advantages

Managed Execution (Cont.)

- ***Virtual machine migration***
  - Managed execution allows easy capturing of the state of the guest program, persisting it, and resuming its execution.
  - This allows VM managers to stop the execution of a guest operating system, move its virtual image into another VM, and resume its execution in **a completely transparent manner**.
  - **This is an important feature in virtualized data centers for optimizing their efficiency in serving application demands**.

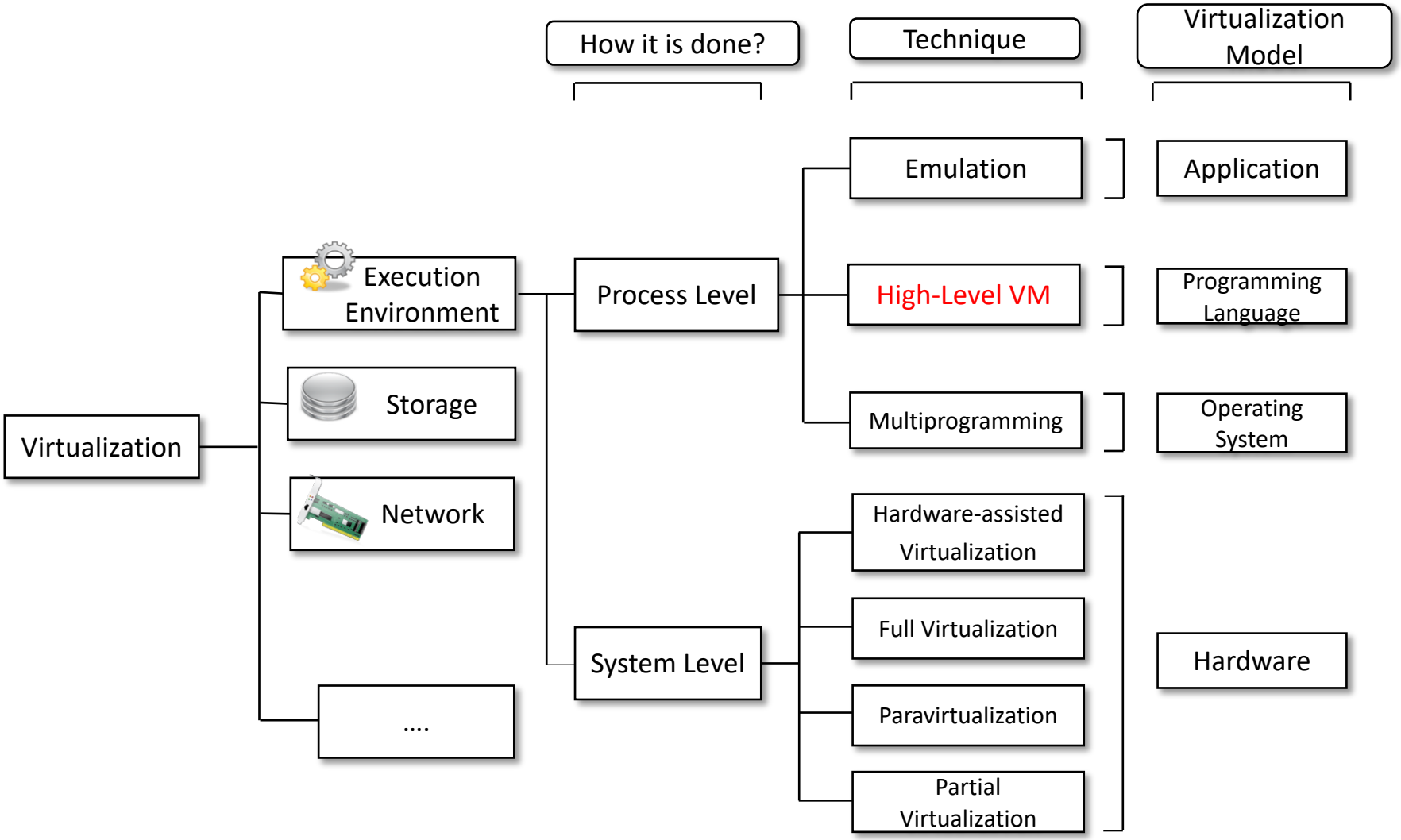
Advantages

Portability

- In the case of a hardware virtualization solution, the guest is packaged into a ***virtual image*** that, in most cases, can be safely moved and executed on top of different ***virtual machines.***

- In the case of programming-level virtualization, as implemented by the JVM or the .NET runtime, the binary code representing application components (jars or assemblies) can be run ***without any recompilation on any implementation of the corresponding virtual machine.***

Taxonomy of Virtualization Techniques



Taxonomy of Virtualization Techniques

- Execution Virtualization
  - Hardware Level
  - Operating System Level
  - Programming Language Level
- Network Virtualization
- Storage Virtualization
- Desktop Virtualization
- …

Part 2

Execution Virtualization

- Includes all techniques that aim ***to emulate an execution environment that is separate from the one hosting the virtualization layer***.

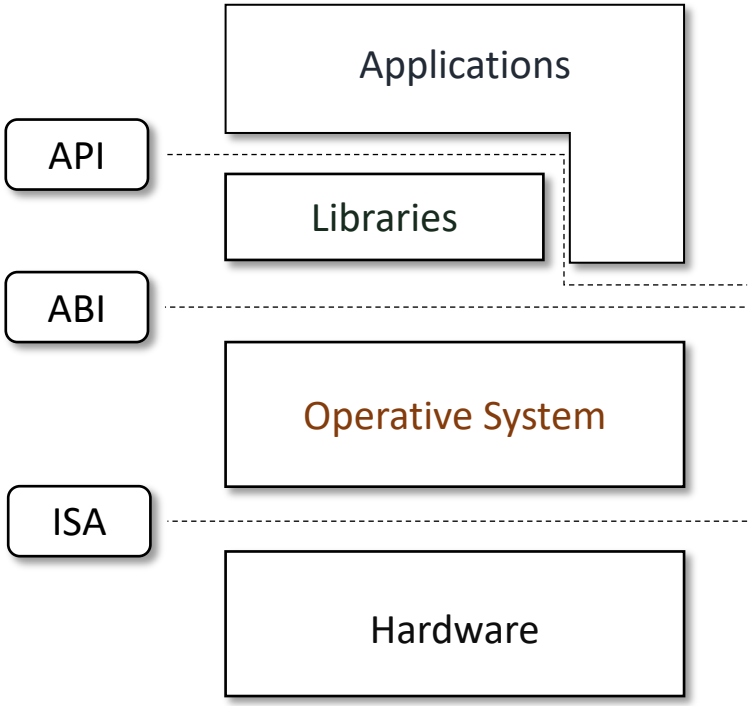
- All these techniques concentrate their interest on providing support for the execution of programs, whether these are the operating system, a binary specification of a program compiled against an abstract machine model, or an application.

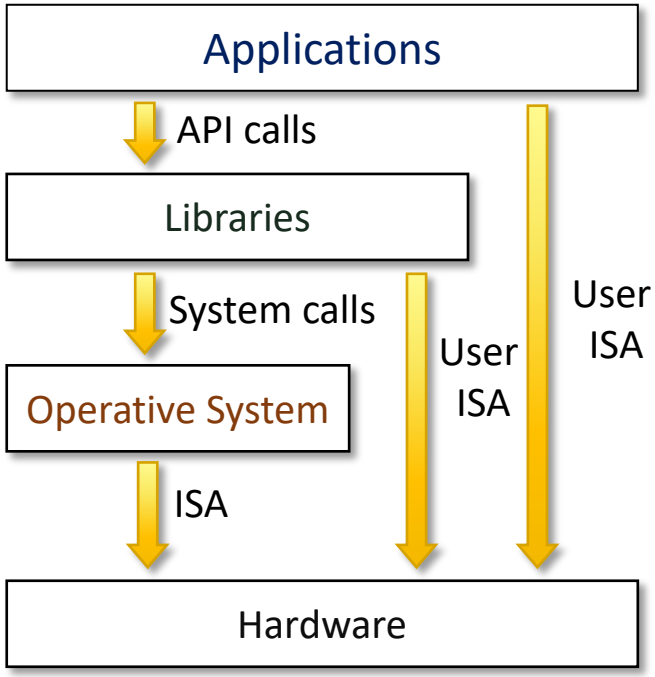
Machine Reference Model

- Virtualizing an execution environment at different levels of the computing stack requires a reference model that defines ***the interfaces between the levels of abstractions, which hide implementation details.***

- From this perspective, virtualization techniques ***actually replace one of the layers and intercept the calls that are directed toward it***.

Machine Reference Model





ISA: Instruction Set Architecture

ABI: Application Binary Interface

API: Application Programming Interface

Machine Reference Model (Cont.)

- Hardware is expressed in terms of the Instruction Set Architecture (ISA)
  - ISA for processor, registers, memory and the interrupt management.

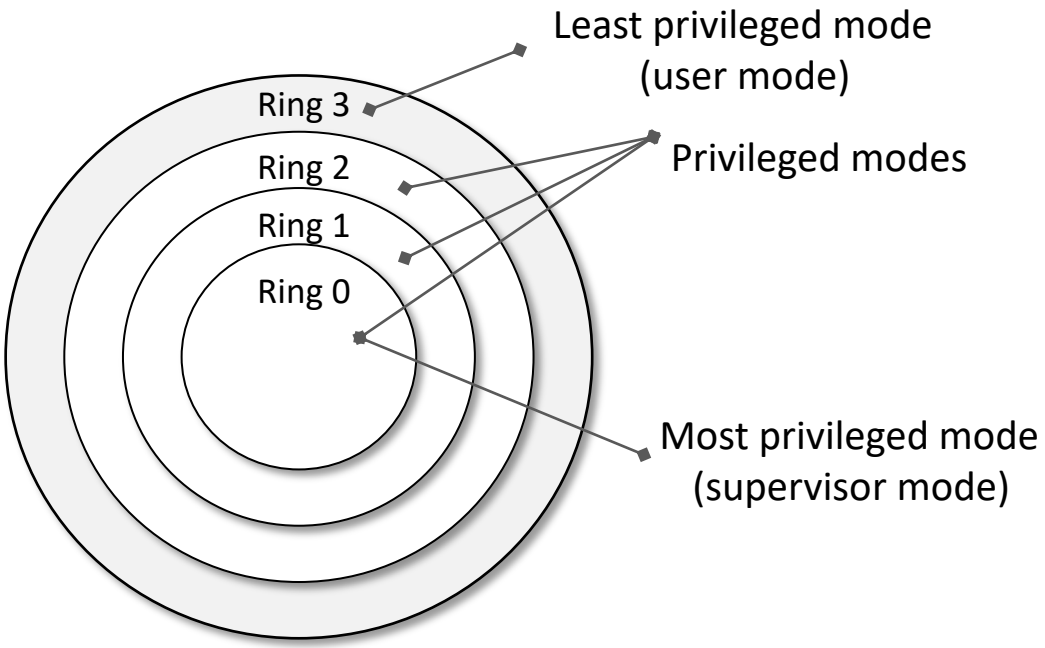
- Application Binary Interface (ABI) separates the OS layer from the application and libraries which are managed by the OS.
  - System Calls defined – Allows portabilities of applications and libraries across OS.

Machine Reference Model (Cont.)

- **Instruction Set**
  - **Non-privileged**
    - Can be used without interfering with other tasks because they do not access shared resources.
    - All the floating, fixed-point, and arithmetic instructions.
  - **Privileged**
    - Executed under specific restrictions and are mostly used for sensitive operations.
      - Behavior-sensitive instructions that operate on the I/O.
      - Control-sensitive instructions alter the state of the CPU registers.

Machine Reference Model (Cont.)

- Some types of architecture feature more than one class of privileged instructions.
- A possible implementation features a hierarchy of privileges in the form of ring-based security: Ring 0, Ring 1, Ring 2, and Ring 3.



Machine Reference Model (Cont.)

- All the current systems support at least two different execution modes:
  - Supervisor mode (master mode or kernel mode) is generally used by the operating system (or the hypervisor) to perform sensitive operations on hardware-level resources.
  - User mode: there are restrictions to control the machine-level resources.
    - If code running in user mode invokes the privileged instructions, hardware interrupts occur and trap the potentially harmful execution of the instruction.

Machine Reference Model (Cont.)

- Conceptually, the hypervisor runs above the supervisor mode, and from here the prefix hyper- is used.

- In reality, hypervisors are run in supervisor mode, and the division between privileged and non-privileged instructions has posed challenges in designing virtual machine managers.

Machine Reference Model (Cont.)

- It is expected that all the sensitive instructions will be executed in privileged mode.
- Unfortunately, this is not true for the original ISA, which allows 17 sensitive instructions to be called in user mode.
- This prevents multiple operating systems managed by a single hypervisor to be isolated from each other, since they are able ***to access the privileged state of the processor and change it.***
- More recent implementations of ISA (Intel VT and AMD Pacifica) have solved this problem by redesigning such instructions as privileged ones.

Hardware-level Virtualization

- A virtualization technique providing an abstract execution environment in terms of computer hardware on top of which a ***guest operating system*** can be run.

Concept	Represented by
Guest	
Host	
Virtual machine	
Virtual machine manager (virtualization layer)	

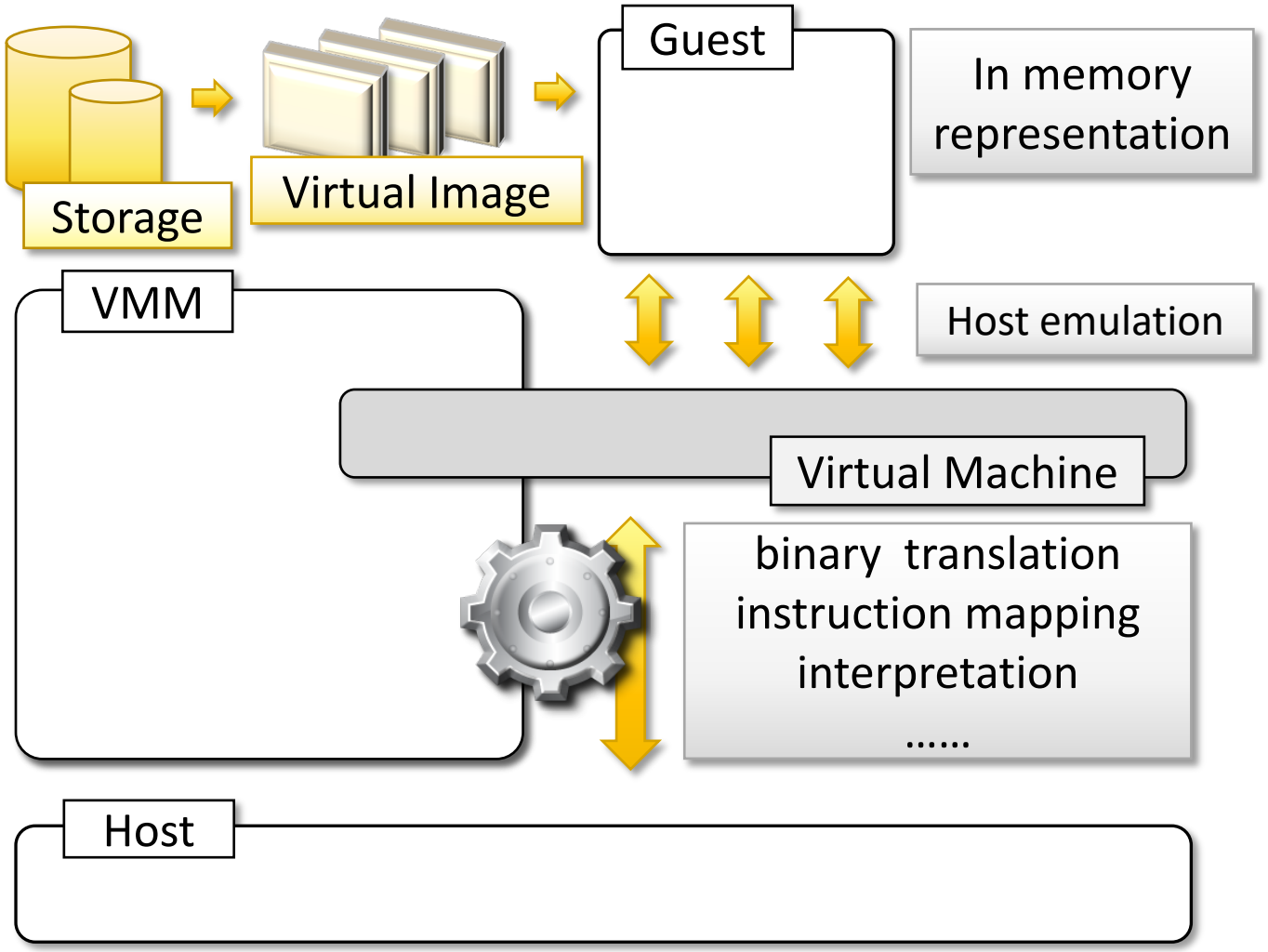
Hardware-level Virtualization

- A virtualization technique providing an abstract execution environment in terms of computer hardware on top of which a ***guest operating system*** can be run.

Concept	Represented by
Guest	Operating system
Host	Physical computer hardware
Virtual machine	Its emulation
Virtual machine manager	Hypervisor

Hypervisor is a program (or a combination of software and hardware) enabling the abstraction of the underlying physical hardware.

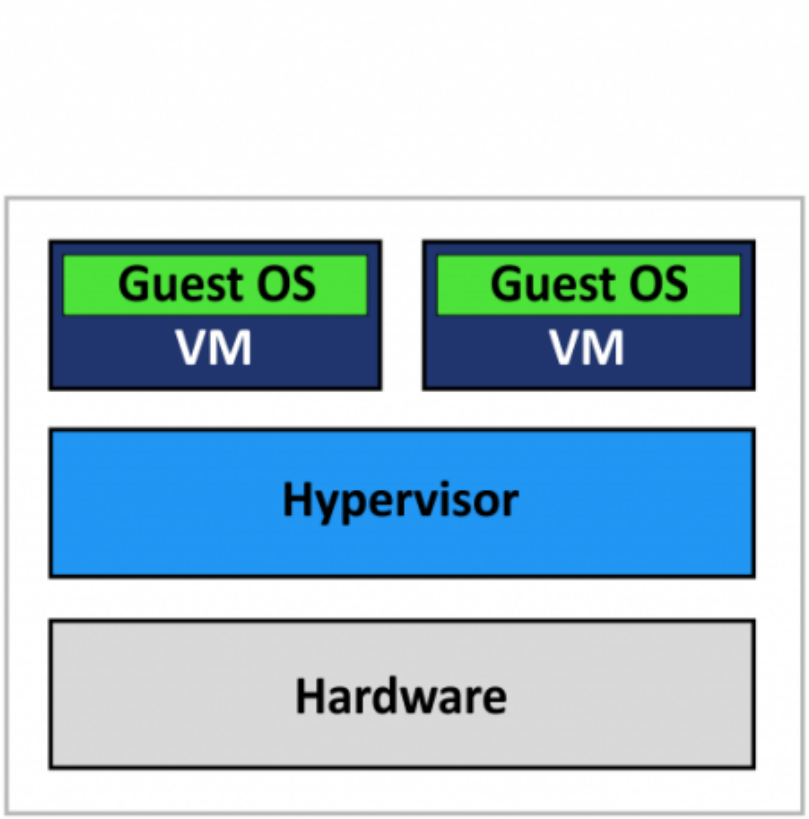
Hardware-level Virtualization



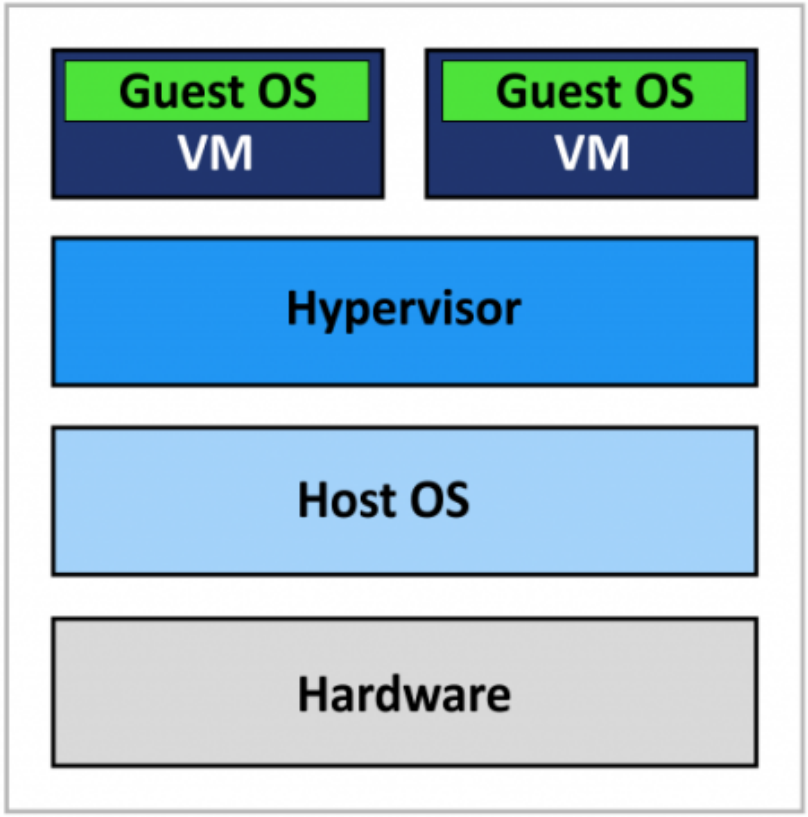
Hypervisors

- A fundamental element of hardware virtualization ***is the hypervisor, or virtual machine manager (VMM).***
- There are ***two major types of hypervisor***
  - ***Type I hypervisors*** (native VM) run directly on top of the hardware.
    - Therefore, they take the place of the operating systems and interact directly with the ISA interface exposed by the underlying hardware.
  - ***Type II hypervisors*** (hosted VM) require the support of an operating system to provide virtualization services.
    - This means that they are programs **managed by the operating system**, which interact with it through the ABI.

Type of Hypervisors



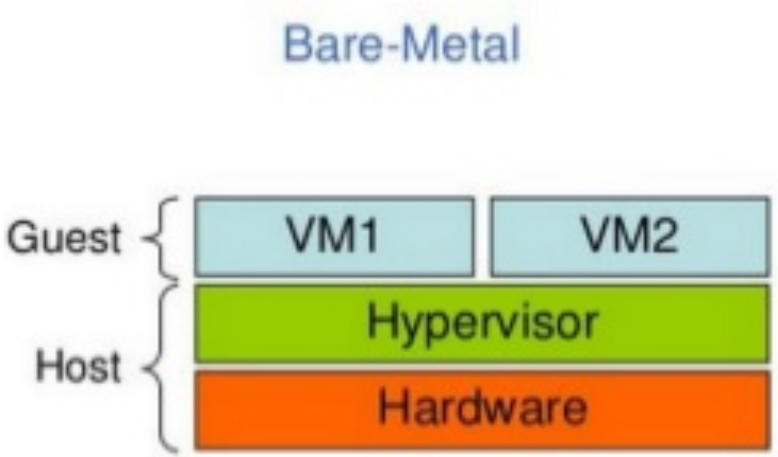
Type 1 Hypervisor
(Bare-Metal Architecture)



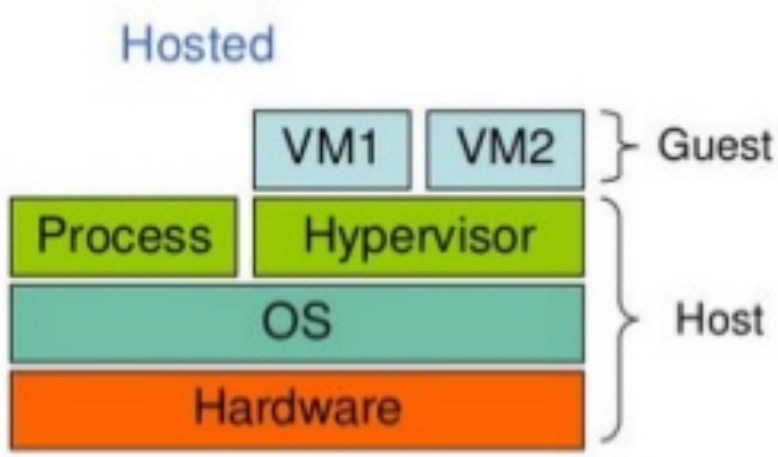
Type 2 Hypervisor
(Hosted Architecture)

Source: https://www.nakivo.com/blog/hyper-v-virtualbox-one-choose-infrastructure/

Type of Hypervisors (Cont.)



VMware ESX, Microsoft Hyper-V, Citrix XenServer



VMware Workstation, Microsoft Virtual PC, Sun VirtualBox, QEMU, KVM

Source: https://www.slideshare.net/PraveenHanchinal/virtualizationthe-cloud-enabler-by-inspiregroups/18-Types_of_hypervisors_VMM

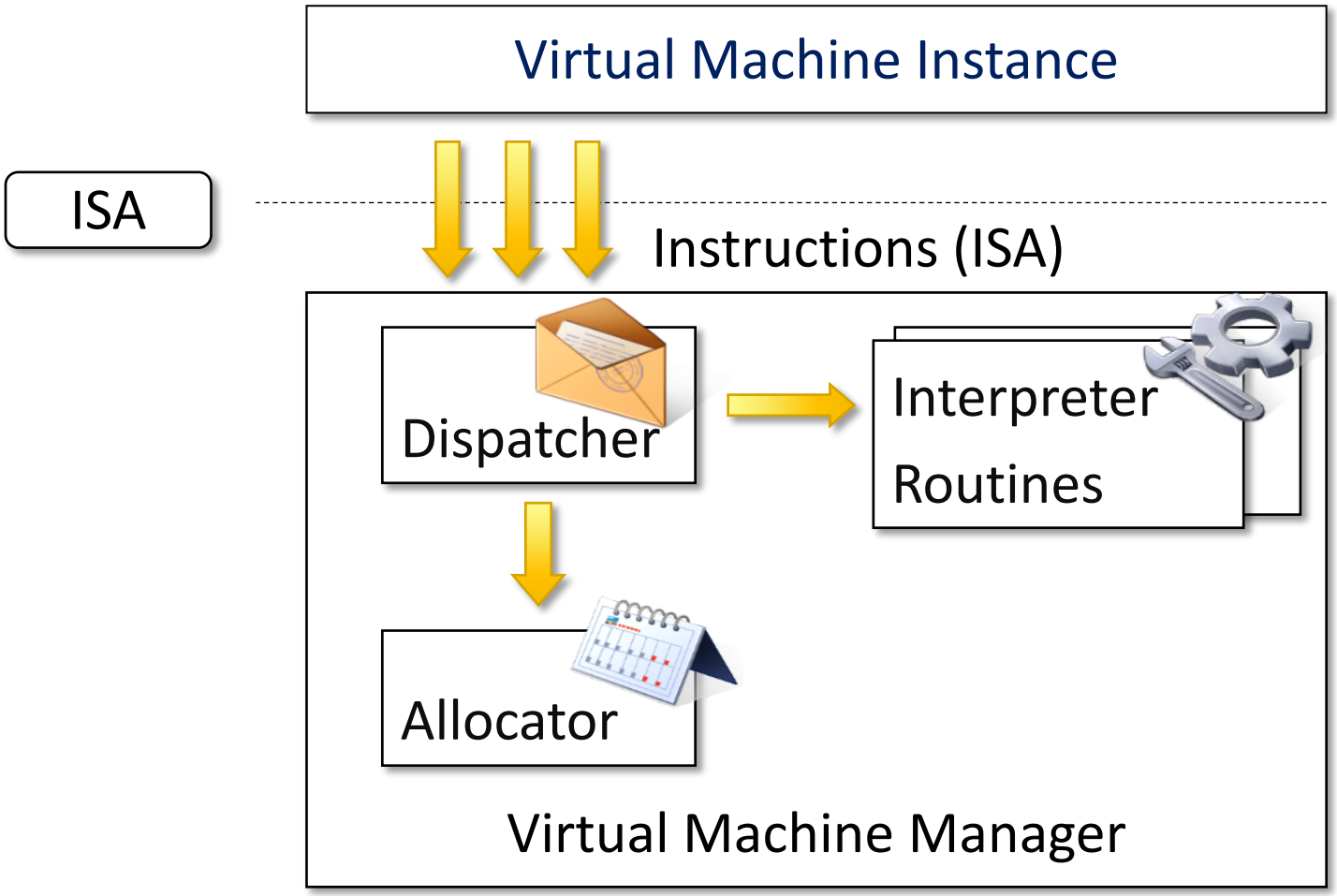
Executing Guest Instructions

- **Emulation**: examining each guest instruction in turn, and emulating on virtualized resources the exact actions that would have been performed on real resources
  - Can be done through either interpretation or binary translation.
  - It is the only available mechanism when the ISA of the guest is different from the ISA of the host.
- **Direct native execution (next slide)**

Executing Guest Instructions (Cont.)

- Emulation (previous slide)
- **Direct native execution**: a significant fraction of the instructions ***execute directly*** on the native hardware and the remaining instructions (sensitive instructions) are ***emulated***.
  - This method is possible only if the ISA of the host is identical to the ISA of the guest.

Hypervisors Modules



Hypervisors Modules

- **The dispatcher**: constitutes the entry point of the monitor and reroutes the instructions issued by the virtual machine instance to one of the two other modules.

- ***The allocator*** is responsible for deciding the system resources to be provided to the VM.
  - Whenever a virtual machine tries to execute an instruction that results in ***changing the machine resources associated with that VM***, the allocator is invoked by the dispatcher.

- ***The interpreter*** module consists of interpreter routines
  - These are executed whenever a virtual machine executes a privileged instruction: a trap is triggered and the corresponding routine is executed.

Enabling Virtualization

- The criteria that need to be met by a VM manager to efficiently support virtualization were established by Goldberg and Popek in 1974
  - ***Equivalence***: a guest running under the control of a VMM should exhibit the same behavior as when it is executed directly on the physical host.
  - ***Resource Control***: the VMM should be in complete control of virtualized resources.
  - ***Efficiency***: a statistically dominant fraction of the machine instructions should be executed without intervention from VMM.

Formal Requirements for Virtualizable Third Generation Architectures

Gerald J. Popek
University of California, Los Angeles
and
Robert P. Goldberg
Honeywell Information Systems and
Harvard University

Virtual machine systems have been implemented on a limited number of third generation computer systems, e.g. CP-67 on the IBM 360/67. From previous empirical studies, it is known that certain third generation computer systems, e.g. the DEC PDP-10, cannot support a virtual machine system. In this paper, model of a third-generation-like computer system is developed. Formal techniques are used to derive precise sufficient conditions to test whether such an architecture can support virtual machines.

Key Words and Phrases: operating system, third generation architecture, sensitive instruction, formal requirements, abstract model, proof, virtual machine, virtual memory, hypervisor, virtual machine monitor
CR Categories: 4.32, 4.35, 5.21, 5.22

This is a revised version of a paper presented at the Fourth ACM Symposium on Operating System Principles, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, October 15–17, 1973.

This research was supported in part by the U.S. Atomic Energy Commission, Contract No. AT(11-1) Gen 10, Project 14 and in part by the Electronic Systems Division, U.S. Air Force, Hanscom Field, Bedford, Massachusetts under Contract Number F19628-70-0217.

Authors' addresses: Gerald J. Popek, Computer Science Department, University of California, Los Angeles CA 90024; Robert P. Goldberg, Honeywell Information Systems, Waltham, MA 02159.

Communications of the ACM
July 1974
Volume 17
Number 7

Enabling Virtualization (Cont.)

- Popek and Goldberg provided a classification of the instruction set and proposed three theorems that define the properties that hardware instructions need to satisfy in order to efficiently support virtualization.
  - Theorem 1: For any conventional third-generation computer, a VMM may be constructed if ***the set of sensitive instructions*** for that computer is a subset of the ***set of privileged instructions.***

Enabling Virtualization (Cont.)

- The first theorem establishes that all the instructions that change the configuration of the system resources should ***generate a trap in user mode and be executed under the control of the VMM.***

- This allows hypervisors to efficiently control only those instructions that would reveal the presence of an abstraction layer while executing all the rest of the instructions ***without considerable performance loss.***

Enabling Virtualization (Cont.)

- The theorem always guarantees the resource control property when the hypervisor is in the most privileged mode.

- The non-privileged instructions must be executed without the intervention of the hypervisor.

- The equivalence property also holds good since the output of the code is the same in both cases because the code is not changed.

Part 3

Hardware Virtualization Methods

- **Full Virtualization**
  - Binary Translation
  - Hardware-assisted virtualization
- **Paravirtualization**
- **Partial virtualization**

Full Virtualization

- Refers to the ability to run a program, most likely an ***operating system***, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware.
- The principal advantage of full virtualization
  - Complete isolation → enhanced security
  - Ease of emulation of different architectures
  - Coexistence of different systems on the same platform.

Full Virtualization (Cont.)

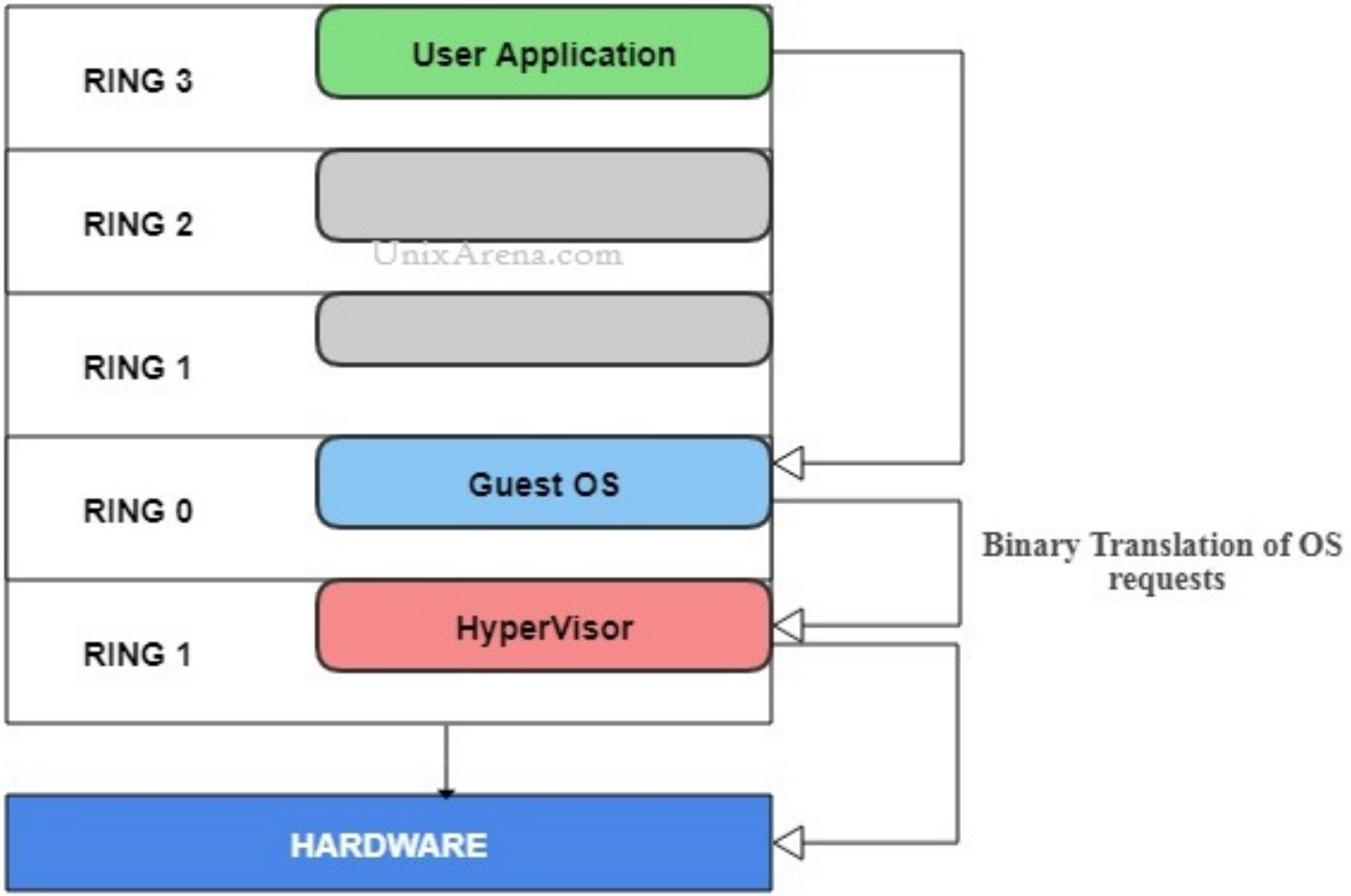
- Unfortunately, Some architectures, like the non-hardware-assisted x86, do not meet the first condition ***(Theorem 1***), so they cannot be virtualized in the classic way.

- ***Two technologies:***
  - Binary Translation
  - Hardware-assisted virtualization

Binary Translation

- Replaces the sensitive instructions that do not generate traps (critical instructions) with a trap into the VMM to be emulated in software.

Binary Translation (Cont.)



Source:https://www.unixarena.com/2017/12/para-virtualization-full-virtualization-hardware-assisted-virtualization.html/

Binary Translation (Cont.)

- Static Binary Translation
  - On a full program
- Dynamic Binary Translation
  - Translating instructions at runtime introduces an additional overhead.

Dynamic Binary Translation

- It is usually performed in small units called "basic blocks".

- A basic block is a set of instructions that ends with a branch instruction but does not have any branch instructions inside.
  - Such a block will always be executed start to finish by a CPU, and is therefore an ideal unit for translation.

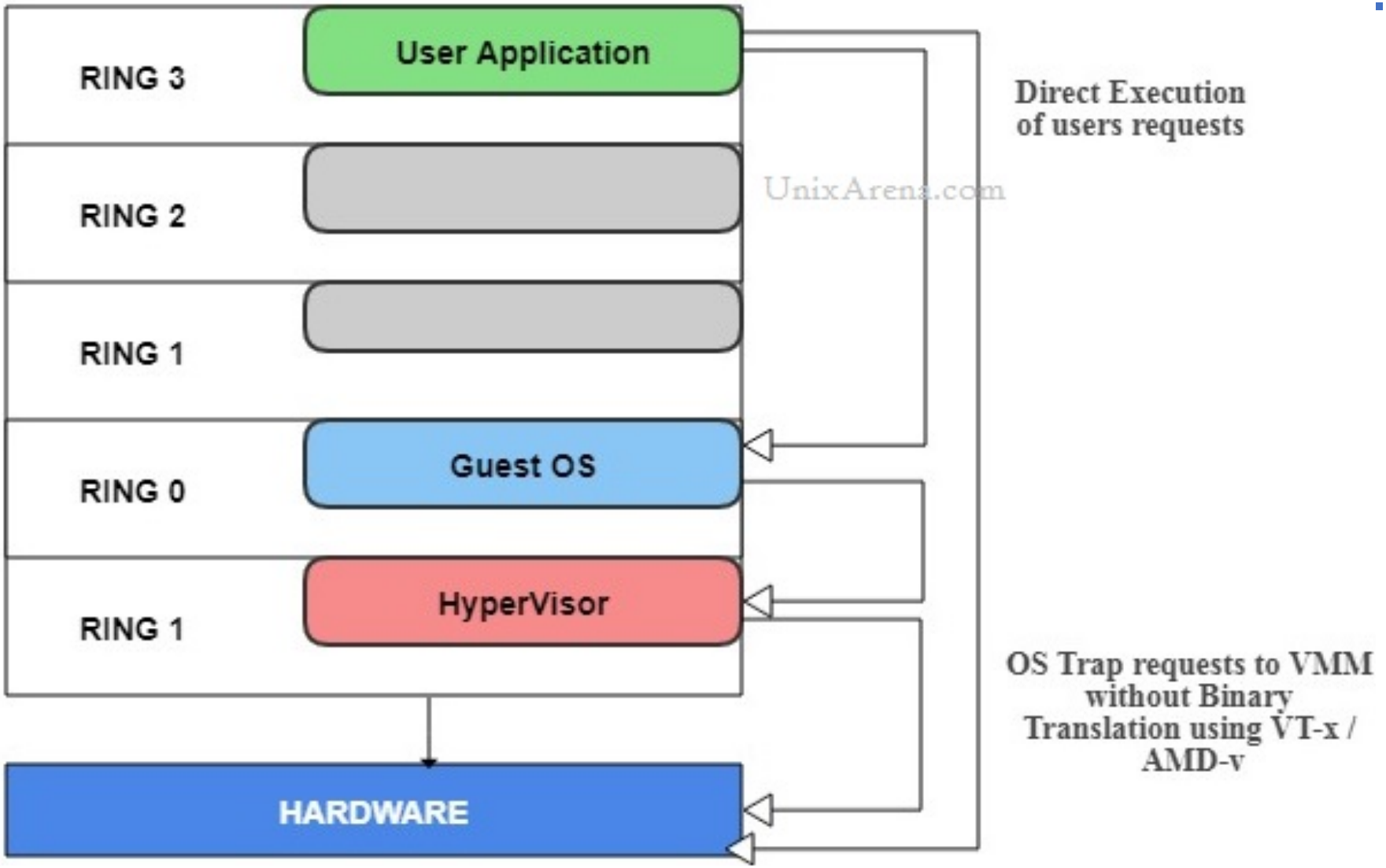
- The translations of the basic blocks are cached → overhead of translating only happens the first time a block is executed.

Source: https://blogs.oracle.com/ravello/nested-virtualization-with-binary-translation

Hardware-assisted Virtualization

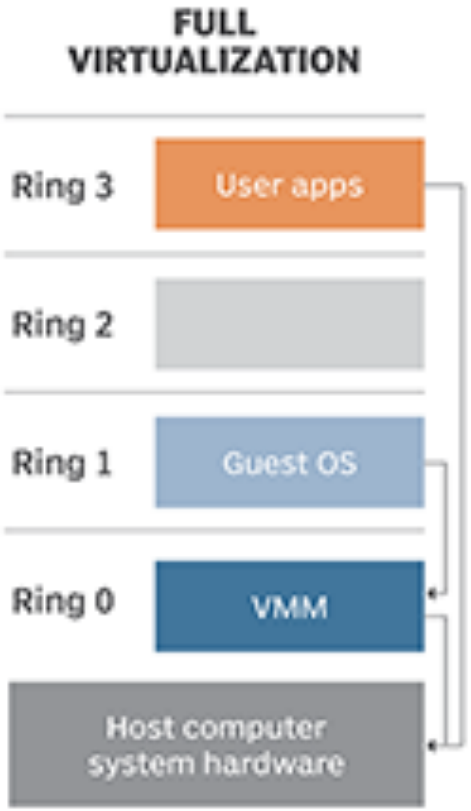
- Hardware provides architectural support for building a VMM able to run a guest operating system in complete isolation.
- This technique was originally introduced in the IBM System/370.
- At present, examples of hardware-assisted virtualization are the extensions to the x86-64 bit architecture introduced with Intel-VT and AMD-V.

Hardware-assisted Virtualization (Cont.)



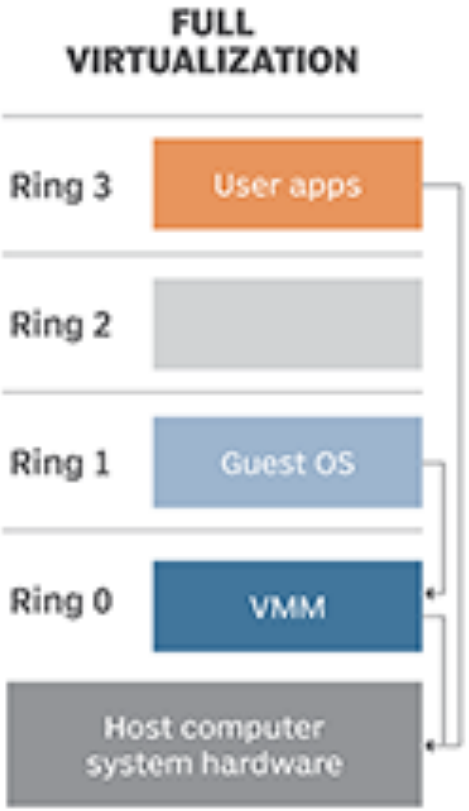
Source:https://www.unixarena.com/2017/12/para-virtualization-full-virtualization-hardware-assisted-virtualization.html/

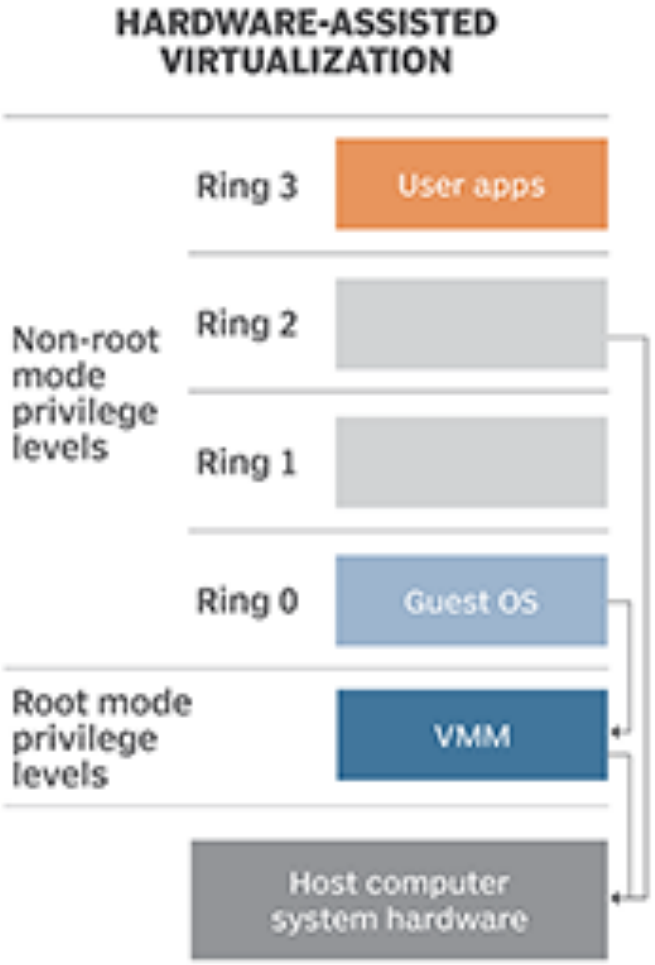
System virtualization implementations



https://searchservervirtualization.techtarget.com/definition/hardware-assisted-virtualization

System virtualization implementations





https://searchservervirtualization.techtarget.com/definition/hardware-assisted-virtualization

Hardware-assisted Virtualization

- Intel-VT and AMD-V target privileged instructions with a new CPU execution mode feature that allows the VMM to run in a new root mode below ring 0, also referred to as Ring 0P (for privileged root mode) while the Guest OS runs in Ring 0D (for de-privileged non-root mode).

- Privileged and sensitive calls are set to automatically trap to the hypervisor and handled by hardware, removing the need for either binary translation or para-virtualization.

- Full Virtualization Examples: VirtualBox, VMware, and Microsoft Hyper-V.

Hardware-assisted Virtualization

- The main feature of Intel VT is the inclusion of the new VMX mode of operation.
- When in VMX mode, the processor can be in either VMX root operation or VMX non-root operation.
- In both cases, all four IA-32 privilege levels (rings) are available for use by software.
- The behavior of the processor in VMX root operation is largely similar to its function in a normal processor not incorporating the VT technology, the main difference being the inclusion of a set of new instructions called the VMX instructions.

VMX Instructions

- "VMX" stands for Virtual Machine Extensions
- Adds 13 new instructions: VMPTRLD, VMPTRST, VMCLEAR, VMREAD, VMWRITE, VMCALL, VMLAUNCH, VMRESUME, VMXOFF, VMXON, INVEPT, INVVPID, and VMFUNC.
- These instructions permit entering and exiting a virtual execution mode where the guest OS perceives itself as running with full privilege (ring 0), but the host OS remains protected.

Hardware-assisted Virtualization

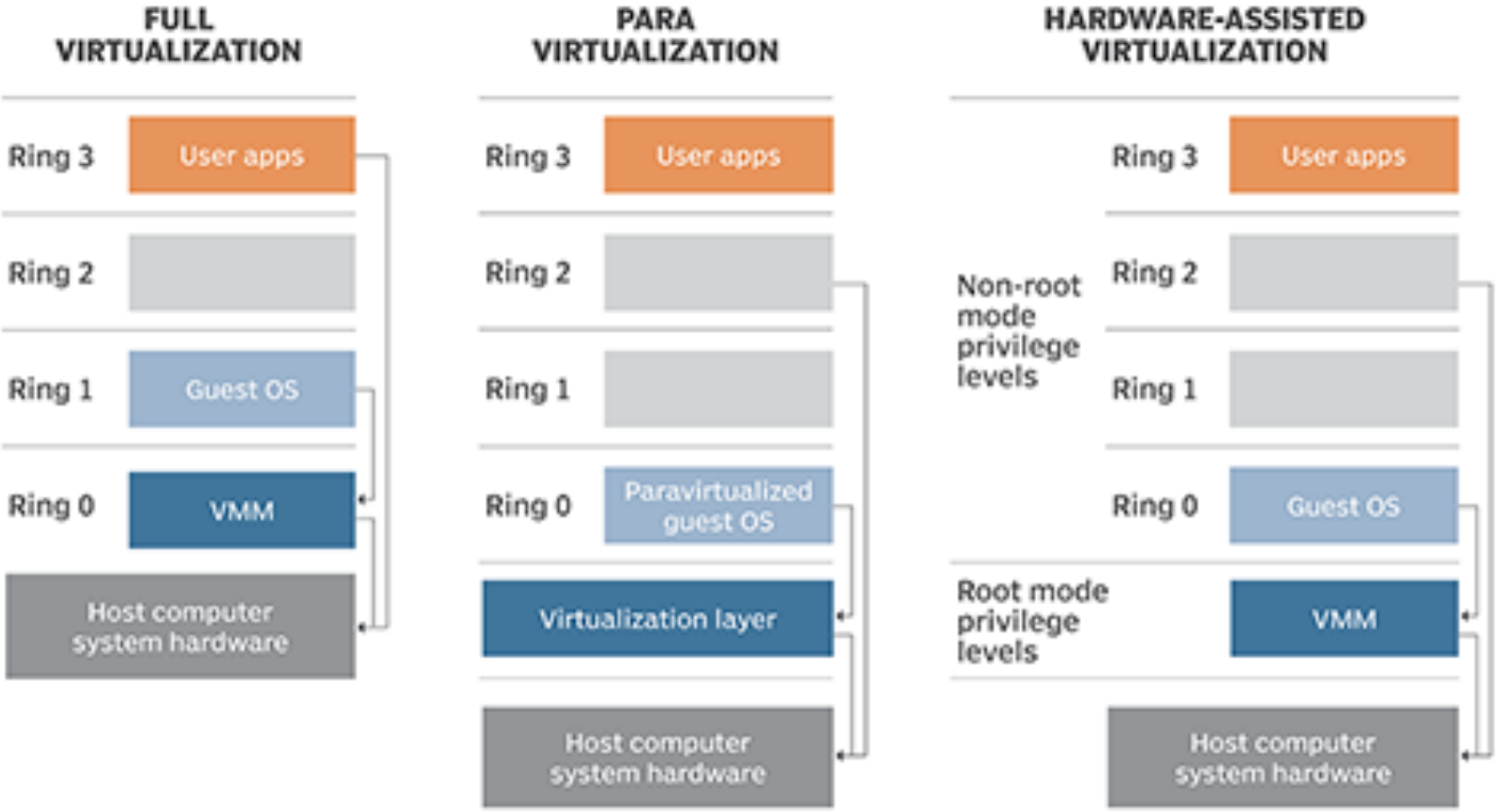
- The behavior of the processor in non-root operation is limited in some respects from its behavior on a normal processor.
- The limitations are such that critical shared resources are kept under the control of a monitor running in VMX root operation.
- VMM is run in VMX root mode, while the virtual machine and the guest OS are run in non-root mode.
- There are also some other hardware instructions to assist virtualization such as maintenance of state information.

Part 4

Paravirtualization

- This is a not-transparent virtualization solution that allows implementing thin virtual machine managers.
- Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified.
- The aim of paravirtualization is to provide the capability to demand the execution of performance-critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution.

System virtualization implementations



https://searchservervirtualization.techtarget.com/definition/hardware-assisted-virtualization

Paravirtualization (Cont.)

- This allows a simpler implementation of virtual machine managers
  - VMM have to simply transfer the execution of these operations, which were hard to virtualize, directly to the host.

Paravirtualization (Cont.)

- To take advantage of such an opportunity, guest operating systems need to be modified and explicitly ported by remapping the performance-critical operations through the ***virtual machine software interface***.

Paravirtualization (Cont.)

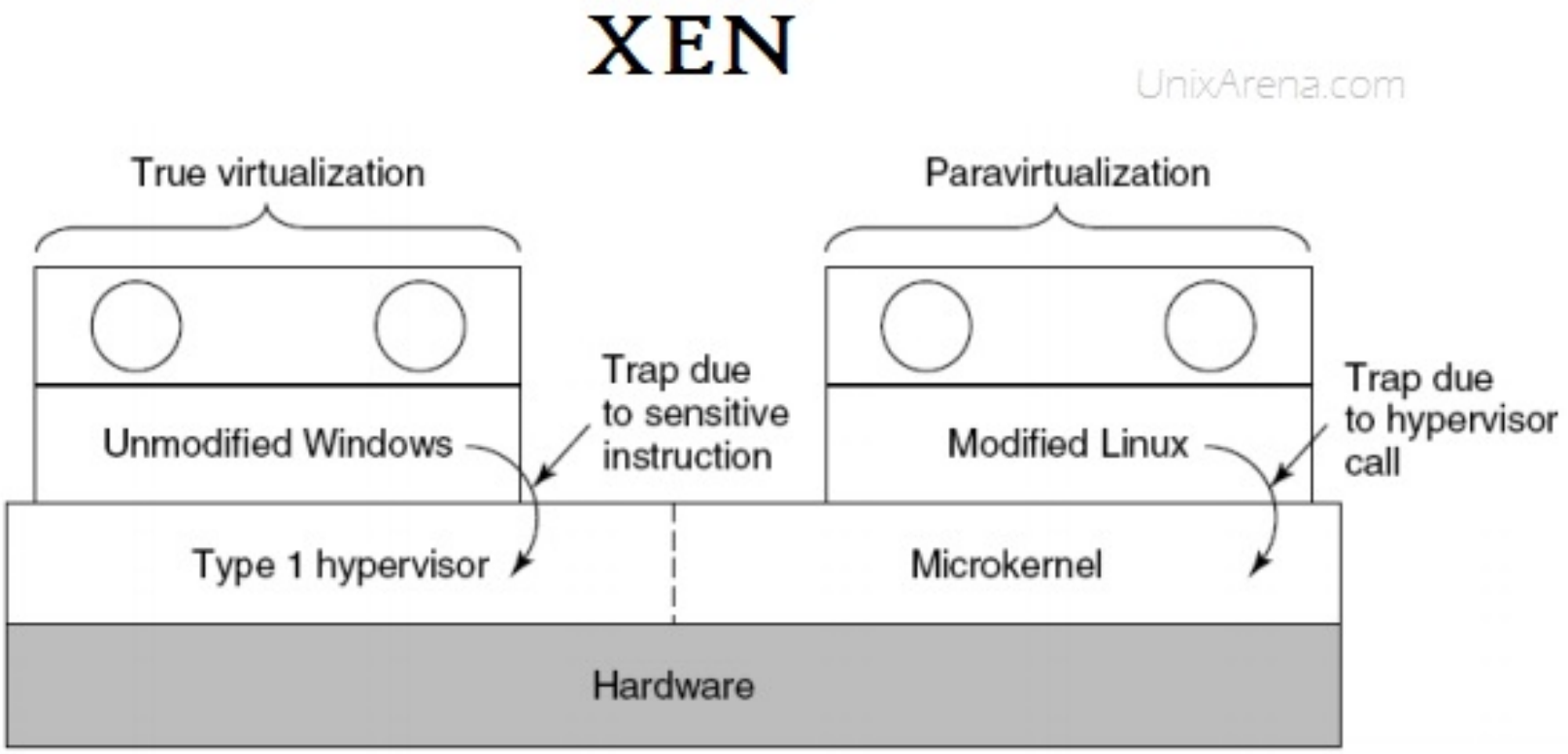
- Xen is the most popular implementation of paravirtualization.

- The guest operating systems need to be changed in their implementation, and the sensitive system calls need to be re-implemented with ***hypercalls***, which are specific calls exposed by the virtual machine interface of Xen.

Paravirtualization (Cont.)

- With the use of ***hypercalls***, the Xen hypervisor is able to catch the execution of all the sensitive instructions, manage them, and return the control to the guest operating system by means of a supplied handler.

Xen Hypervisor



Xen supports both Full virtualization and Para-virtualization

Source:https://www.unixarena.com/2017/12/para-virtualization-full-virtualization-hardware-assisted-virtualization.html/

Paravirtualization (Cont.)

- Open-source operating systems such as Linux can be easily modified, since their code is publicly available and Xen provides full support for their virtualization, whereas components of the Windows family are generally not supported by Xen unless hardware-assisted virtualization is available.