Amirkabir University of Technology
(Tehran Polytechnic)

Cloud Computing

Hardware virtualization

Seyyed Ahmad Javadi

sajavadi@aut.ac.ir

Spring 2023

Introduction

Hardware-level Virtualization

- **An abstract execution environment in terms of computer hardware** on top of which a ***guest operating system*** can be run.

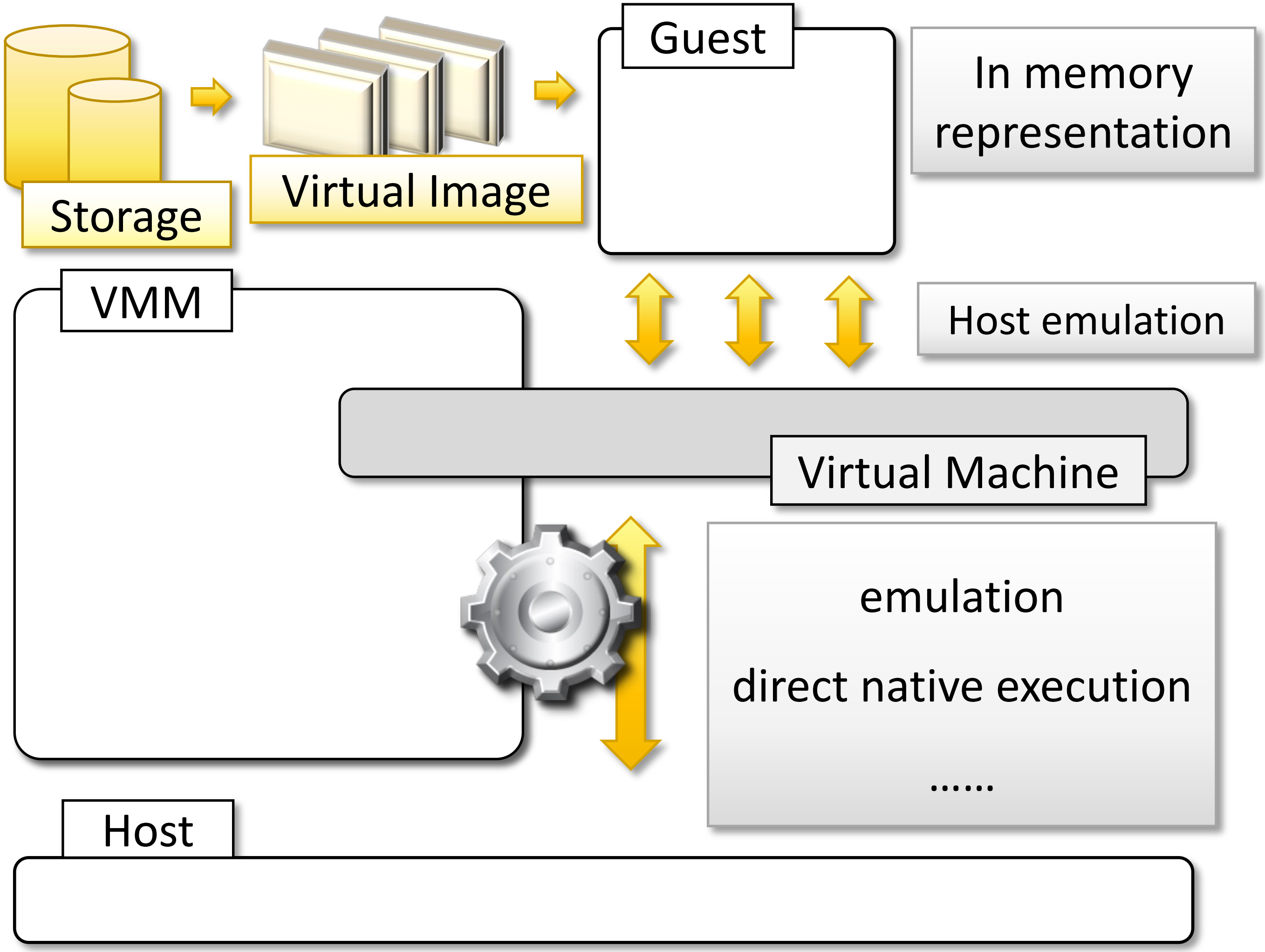
Concept	Represented by
Guest	Operating system
Host	Physical computer hardware
Virtual machine	Its emulation
Virtual machine manager	Hypervisor

What is Hypervisor?

Hypervisor is a program enabling the abstraction of the underlying physical hardware.

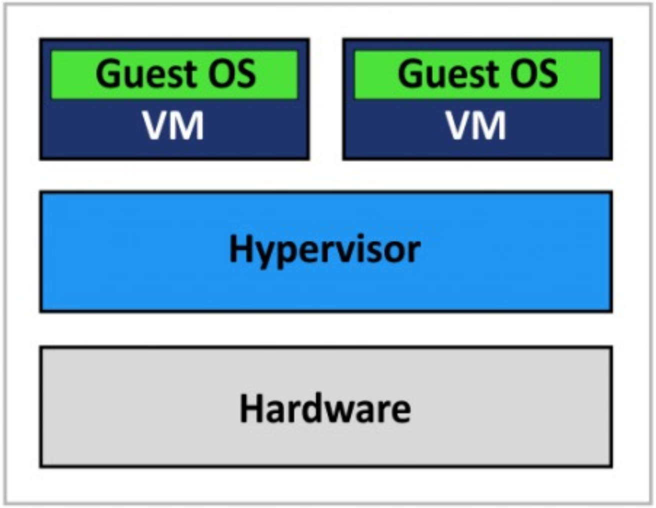
Hypervisor is also called Virtual Machine Manager (**VMM**)

Hardware-level Virtualization



Types of Hypervisor

- ***Type I hypervisors*** (native VM)
  - Run *directly* on top of the hardware.
  - *Take the place* of the operating systems
  - Interact directly with the ISA interface

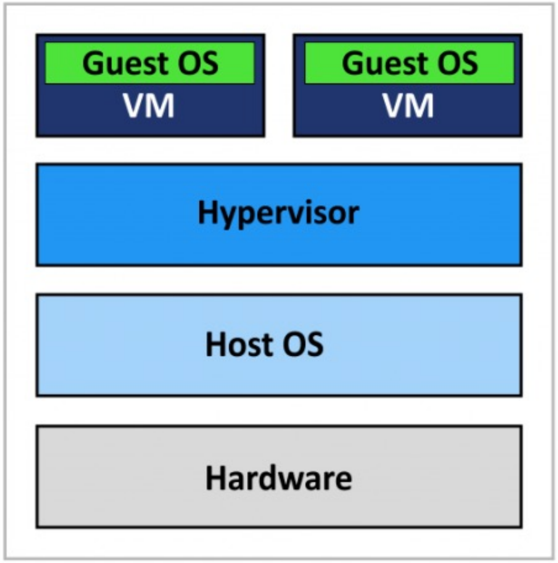


Type 1 Hypervisor
(Bare-Metal Architecture)

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

Types of Hypervisor (cont.)

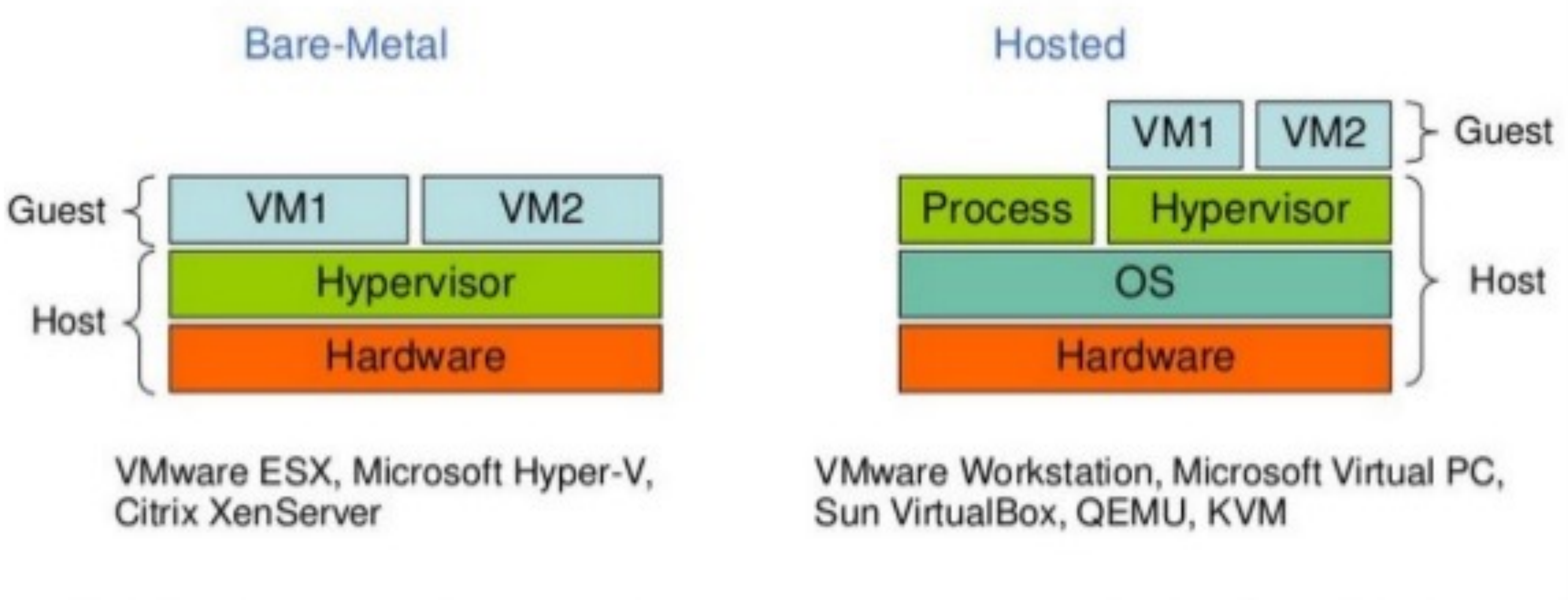
- ***Type II hypervisors*** (hosted VM)
  - Require the support of an operating system
  - Are programs **managed by the operating system**
  - Interact with operating system through the **ABI**.



Type 2 Hypervisor
(Hosted Architecture)

Source: http://: 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

Approaches of Executing Guest Instructions

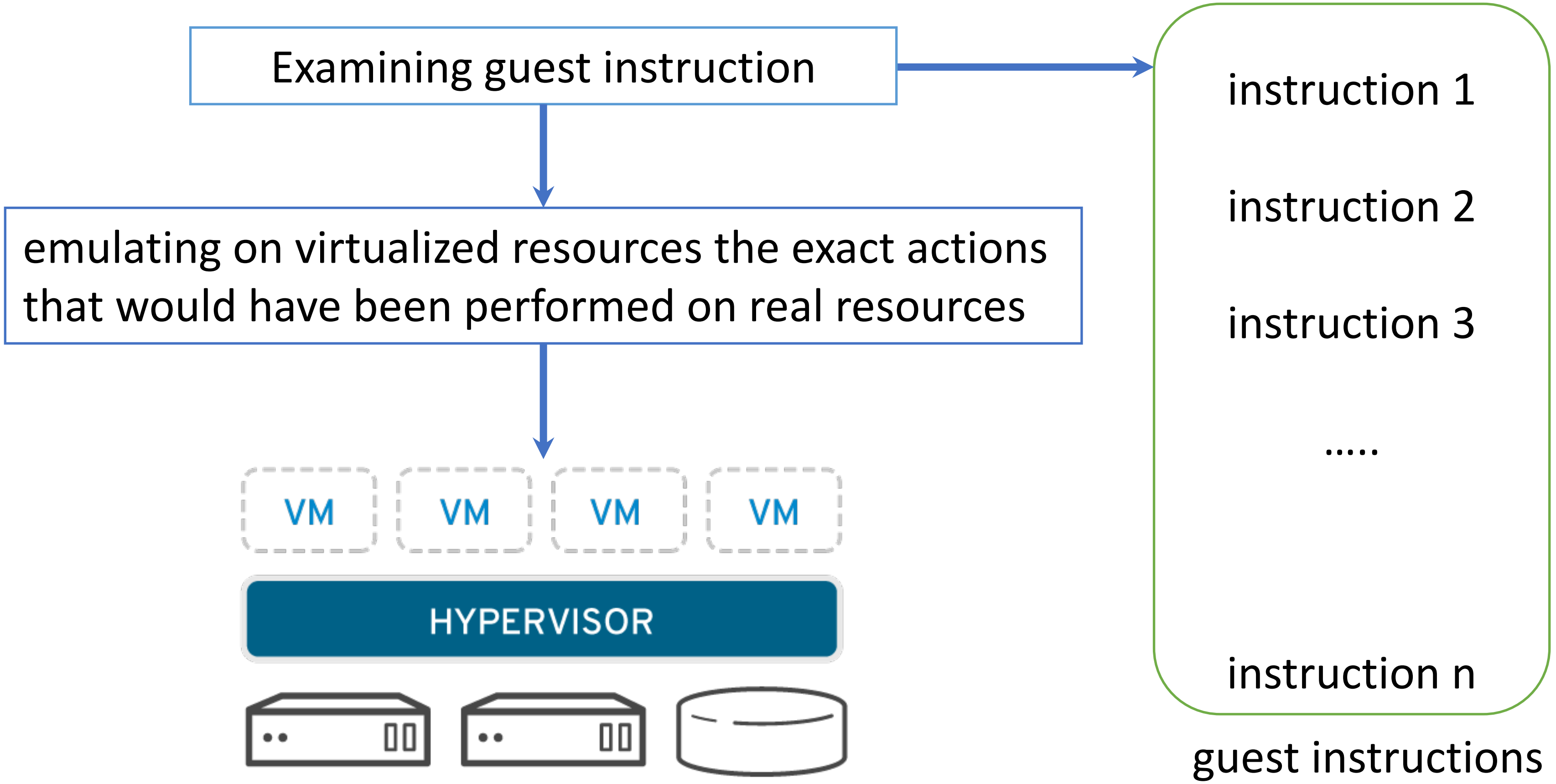
Executing Guest Instructions

- **Emulation**
- **Direct native execution**

Emulation

“the process of implementing the interface and functionality of one system or subsystem on a system or subsystem having a different interface and functionality...”

Emulation (cont.)



Examining guest instruction

emulating on virtualized resources the exact actions that would have been performed on real resources

instruction 1

instruction 2

instruction 3

.....

instruction n

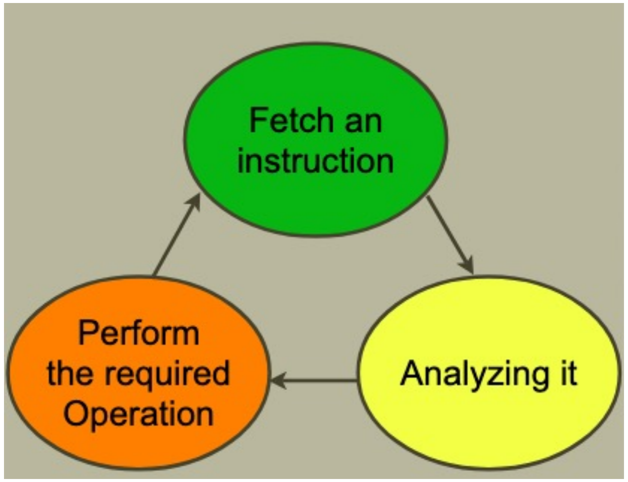
guest instructions

Only available mechanism when the **ISA of the guest** is ***different*** from the **ISA of the host**.

Emulation Approaches

- Interpretation
  - Done in software,

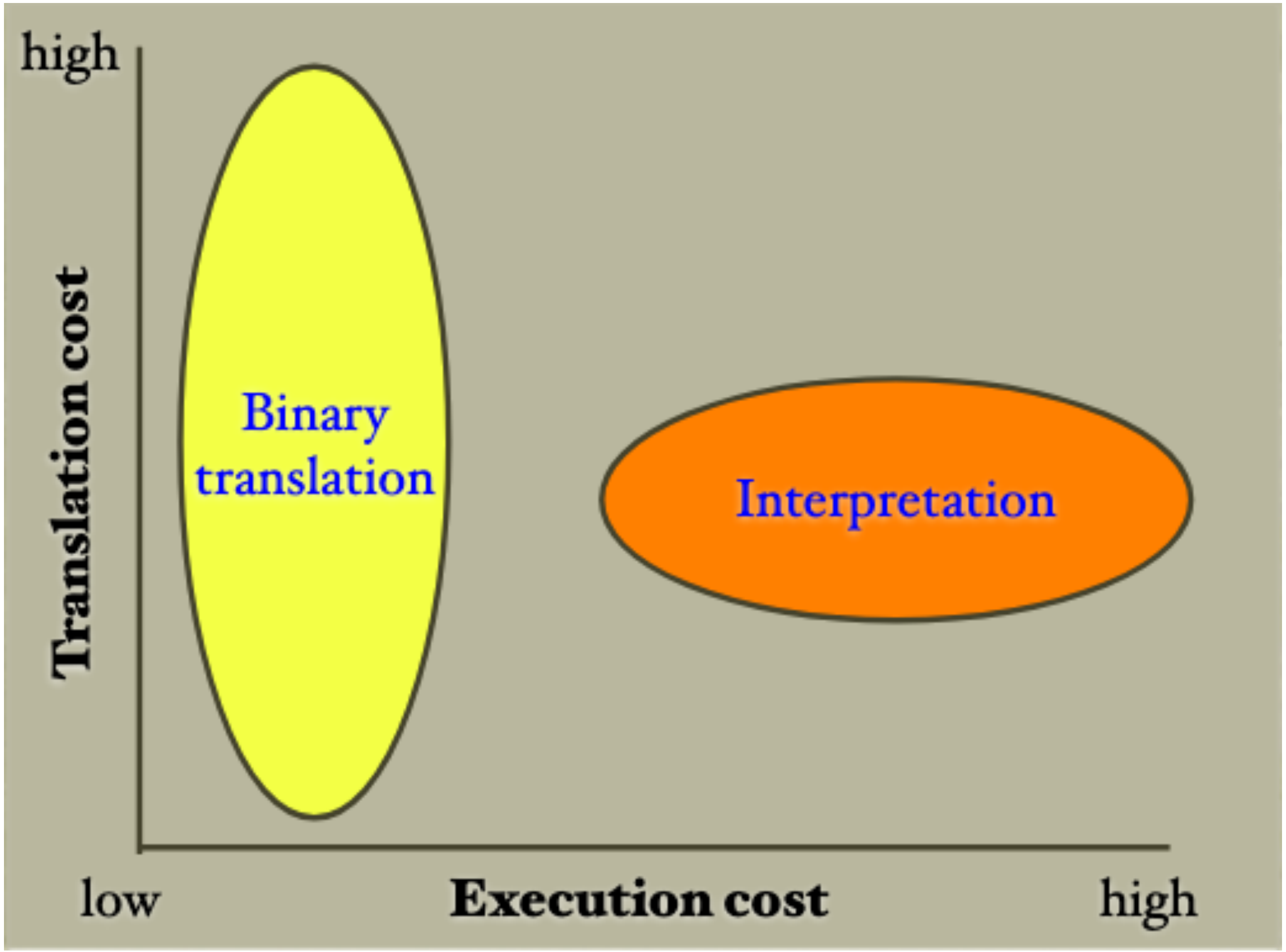
    one instruction at a time



http://cse.unl.edu/~witty/class/embedded/material/note/emulation.pdf

- Binary translation
  - Translating a block of source instructions to target instructions.
  - Saving the translated code for repeated use

Interpretation versus Binary Translation



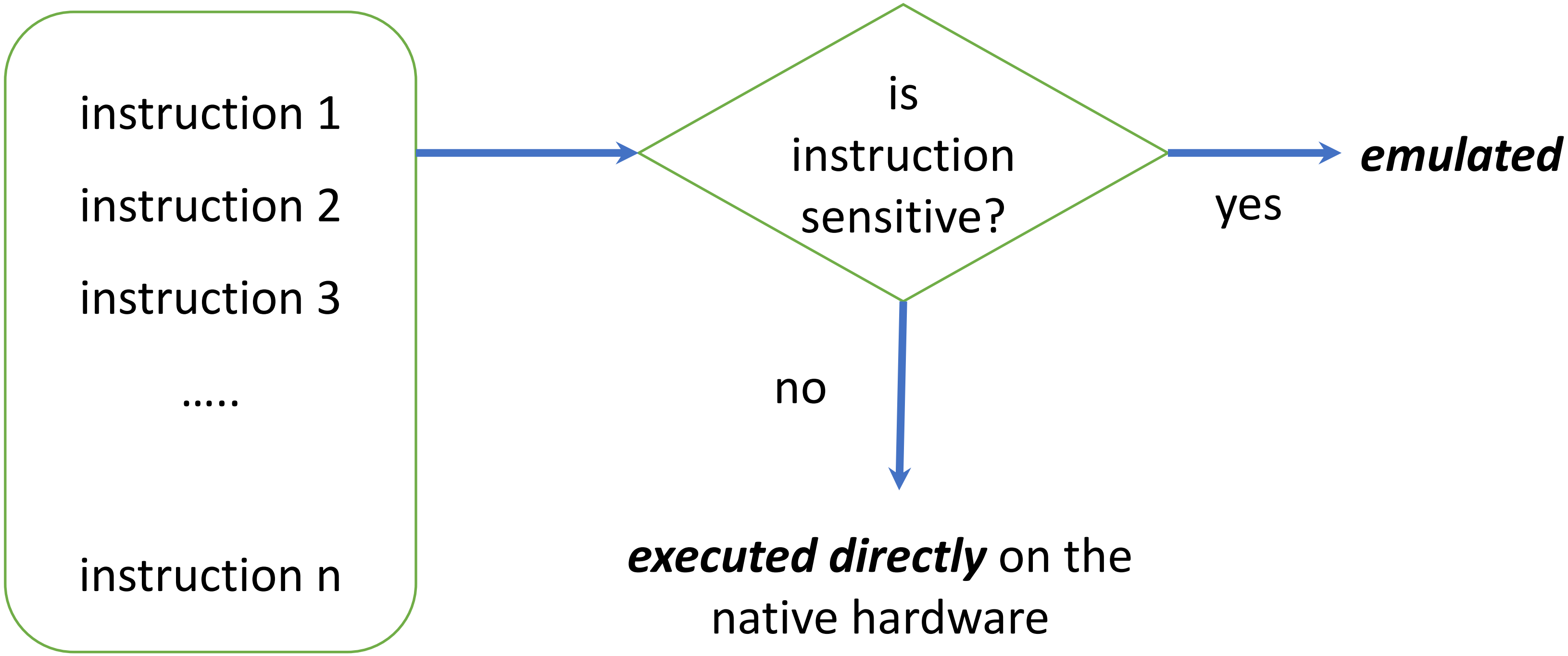
http://se.unl.edu/~witty/class/embedded/material/note/emulation.pdf

Interpretation versus Binary Translation (cont.)

	Implementation	Performance
Interpretation	simple and easy	low
Binary Translation	complex	***high initial*** translation cost, ***small execution*** cost

http://www.ittc.ku.edu/~kulkarni/teaching/EECS768/slides/chapter2.pdf

Direct Native Execution



instruction 1

instruction 2

instruction 3

.....

instruction n

is instruction sensitive?

yes

emulated

no

executed directly on the native hardware

guest instructions

Only if the ***ISA of the host is identical to the ISA of the guest***.

Hardware Virtualization Methods

Hardware Virtualization Methods

- **Full Virtualization**

- **Paravirtualization**

Full Virtualization

- Run a program **directly on top of a VM** and **without any modification**
  - The program thought 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.)

- In some architectures, **some sensitive instructions are not privileged**
  - They cannot be virtualized in the classic way.
  - Like the non-hardware-assisted x86

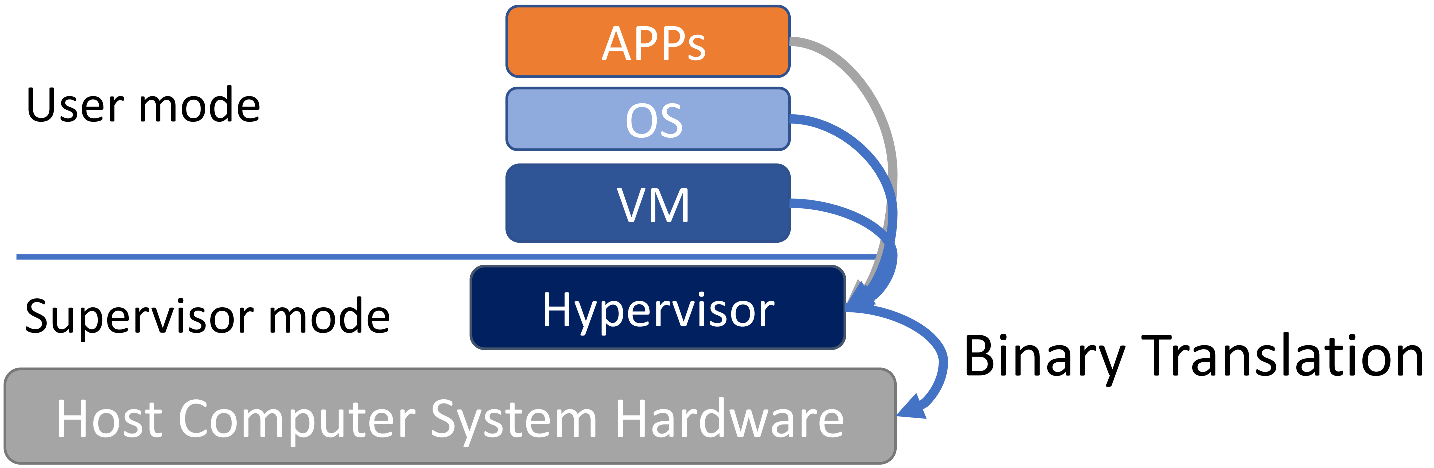
- ***Two technologies:***
  - Emulation (Binary Translation and Interpretation)
  - Hardware-assisted virtualization

First scenario:
Host ISA != Guest ISA

Main Challenge

- Guest Instruction can not directly executed on the host
- Feasible virtualization technologies
  - Interpretation
  - Binary Translation

Binary Translation



Binary Translation (cont.)

- **Static Binary Translation**
  - On a full program
- **Dynamic Binary Translation**
  - 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.
  - Be executed start to finish by a CPU
  - 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.

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

Static vs Binary Translation

	Input type	Granularity	Translation time
Static binary translation	Binary program	Full program	Before running program
Dynamic binary translation	Binary program	Basic block	At runtime

Demo

- Host: Mac M1 chip (ARM-based processor)
- Guest: Ubuntu ISO compiled for X86 processor (AMD)
- Start time of Ubuntu AMD ISO installation: 8:45
- Finish time: 10:05
- Why it takes so long?

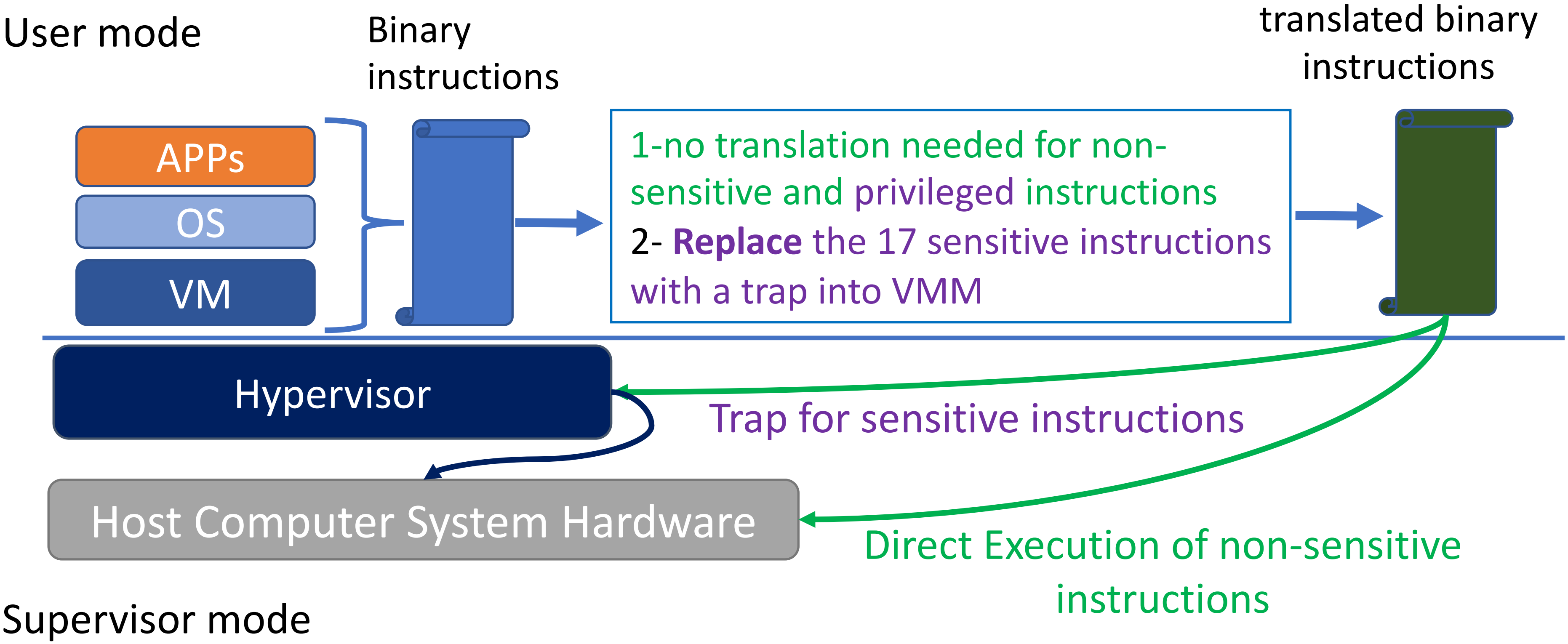
Second scenario:
Host ISA = Guest ISA =
Original ISA Or
Non-hardware assisted X86

Main Challenge

- **Some (17) sensitive** instructions are not privileged.
  - See previous lecture
- They cannot be virtualized in the classic way.
- Feasible virtualization technologies
  - Interpretation
  - Binary Translation and direct execution-based technique

Binary Translation and direct execution-based technique

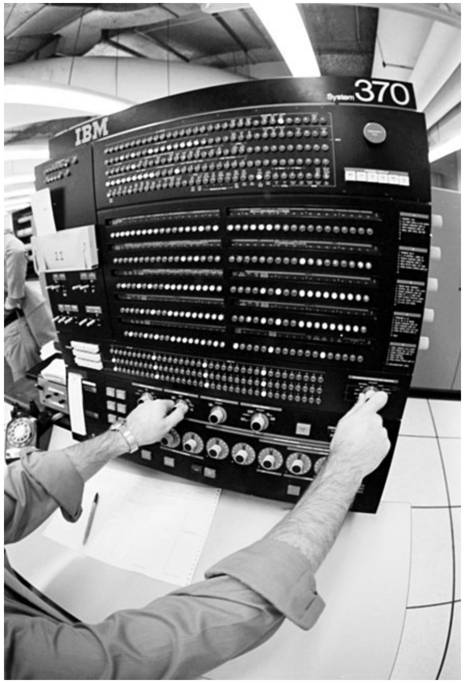
- Replaces the sensitive instructions that do not generate traps (critical instructions) with a trap into the VMM to be emulated in software.


User mode
Binary instructions
translated binary instructions
APPs
OS
VM
1-no translation needed for non-sensitive and privileged instructions
2- **Replace** the 17 sensitive instructions with a trap into VMM
Hypervisor
Trap for sensitive instructions
Host Computer System Hardware
Direct Execution of non-sensitive instructions
Supervisor mode

Third Scenario:
Host ISA = Guest ISA =
Hardware assisted X86
or
Hardware assisted ARM

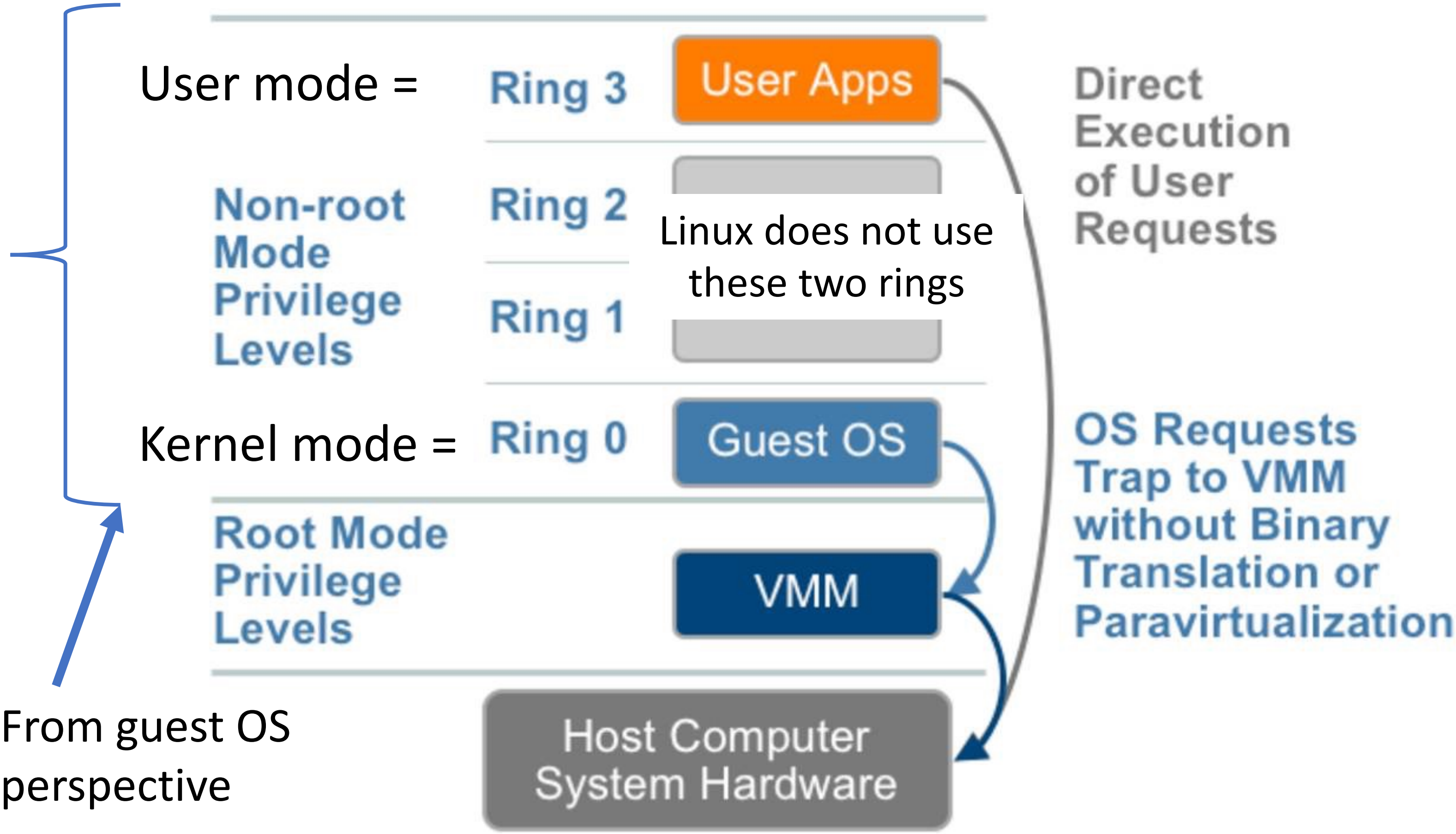
Hardware-assisted Virtualization

- ***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.



- Extensions to x86-64b architecture
  - Introduced with Intel-VT and AMD-V.

Hardware-assisted Virtualization (cont.)



User mode =

Kernel mode =

Linux does not use these two rings

From guest OS perspective

Ring 3
User Apps
Direct Execution of User Requests
Non-root Mode Privilege Levels
Ring 2
Ring 1
Ring 0
Guest OS
OS Requests Trap to VMM without Binary Translation or Paravirtualization
Root Mode Privilege Levels
VMM
Host Computer System Hardware

https://thecustomizewindows.com/2014/09/hardware-assisted-virtualization/

Intel-VT and AMD-V

- New CPU execution mode feature
- This allows the VMM to run in a new root mode below ring 0
  - **Ring 0P**: privileged root mode (VMM)
  - **Ring 0D** : de-privileged non-root mode (Guest OS)
- Sensitive calls are set ***to automatically trap*** to the hypervisor and handled by hardware
  - Removing the need for either binary translation or para-virtualization.

Intel-VT

- Main feature: inclusion of the new VMX mode of operation.

	all four IA-32 privilege levels (rings)	VMX instructions
VMX non-root operation	✓	✗
VMX root operation	✓	✓

VMX Instructions

- "VMX" stands for Virtual Machine Extensions

13 new instructions

VMPTRLD	VMPTRST	VMCLEAR	VMREAD	VMWRITE
VMCALL	VMLAUNCH	VMRESUME	VMXOFF	VMXON
INVEPT	INVVPID	VMFUNC		

- 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.

- ***Critical shared resources are kept under the control of a monitor running in VMX root operation***.

- VMM is run in VMX root mode
- Virtual machine and the guest OS are run in non-root mode.

Examples of Hardware-assisted Virtualization

- VirtualBox
- VMware
- Microsoft Hyper-V

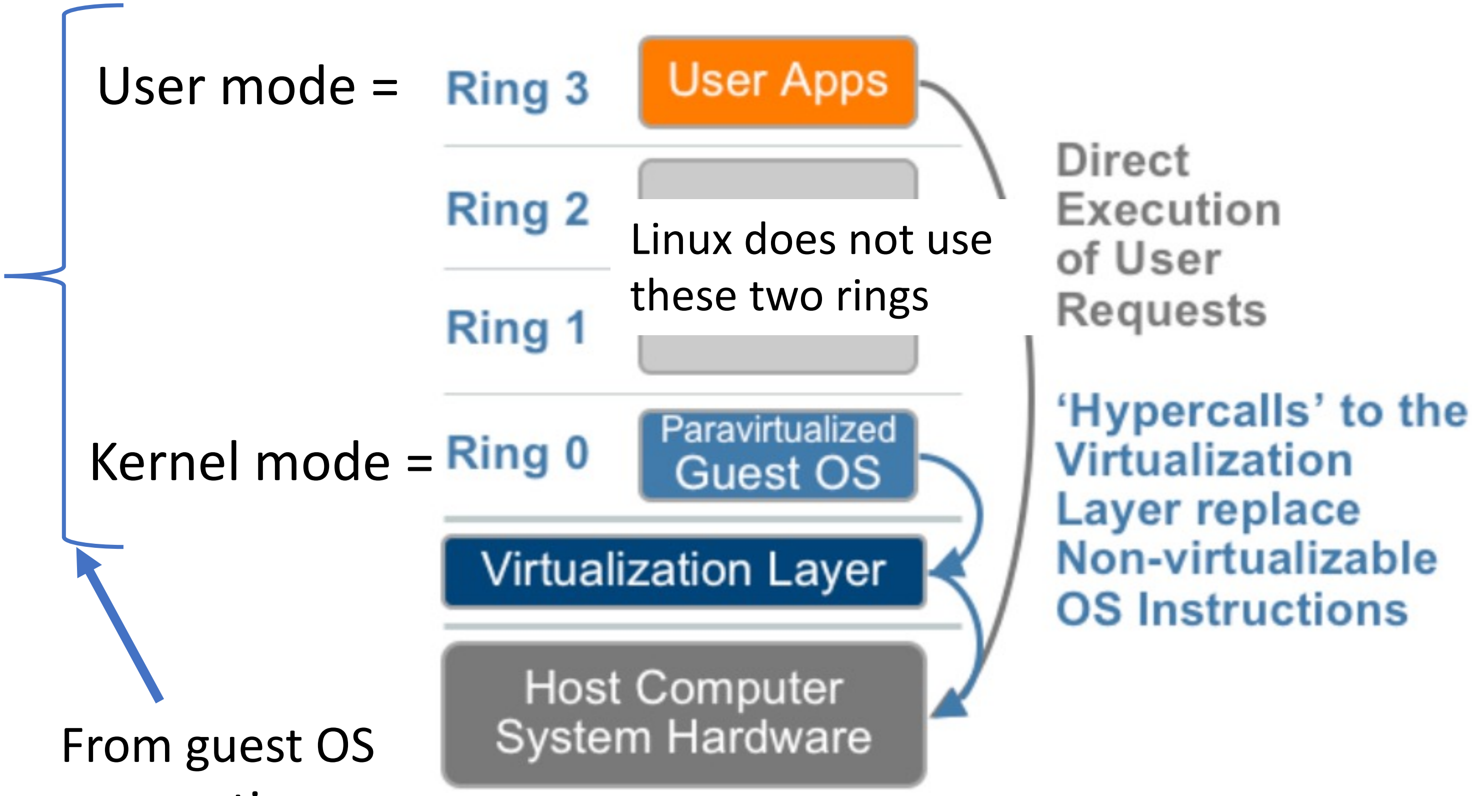
Demo

- Host: Mac M1 chip (ARM-based processor)
- Guest: Ubuntu ISO compiled for ARM processor
- Time taken to install Ubuntu ARM ISO: less than 10 mins
- Let’s do a simple benchmark

Paravirtualization

Paravirtualization

- Paravirtualization refers to communication between the guest OS and the hypervisor to improve performance and efficiency.



User mode =

Kernel mode =

From guest OS perspective

Linux does not use these two rings

Ring 3
User Apps
Ring 2
Ring 1
Ring 0
Paravirtualized Guest OS
Virtualization Layer
Host Computer System Hardware
Direct Execution of User Requests
'Hypercalls' to the Virtualization Layer replace Non-virtualizable OS Instructions

Paravirtualization (cont.)

- It is not a transparent virtualization solution
  - Allows implementing ***thin*** virtual machine managers.
  - Remapping the performance-critical operations through the virtual machine software interface.

- Expose a software interface to the virtual machine that is slightly modified from the host
  - As consequence, guests need to be modified.

Paravirtualization (cont.)

- Provide the capability to demand the execution of performance-critical operations ***directly on the host***
  - Preventing performance losses that would otherwise be experienced in managed execution.

- Allows **a simpler implementation of virtual machine managers**
  - VMM have to simply **transfer** the execution of performance-critical operations **directly to the host**.
  - These instructions were ***hard to virtualize***

Paravirtualization (Cont.)

- Xen is ***the most popular implementation*** of paravirtualization.

- The guest operating systems need to be changed

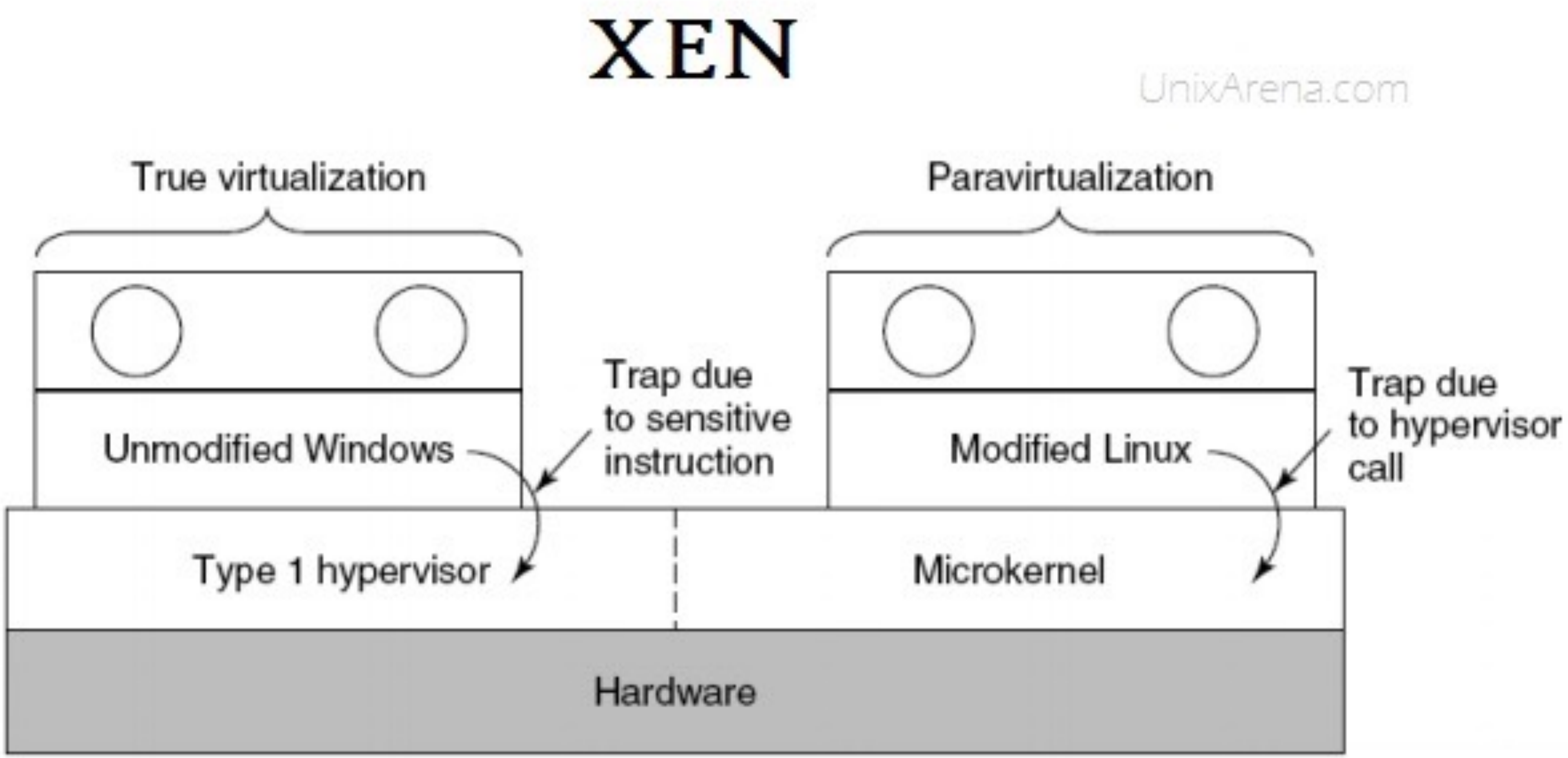
- The sensitive system calls need to be re-implemented with ***hypercalls***
  - Are specific calls exposed by the virtual machine interface of Xen.

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
  - Their code is publicly available
  - Xen provides full support for their virtualization

- Components of the Windows family ***are generally not supported*** by Xen unless hardware-assisted virtualization is available.