COMPUTING AS A SERVICE (IAAS)

PART II

INSTRUCTION SET ARCHITECTURE (ISA)

POPEK AND GOLDBERG INTRODUCED A CLASSIFICATION OF INSTRUCTIONS OF AN ISA INTO 3 DIFFERENT GROUPS:

- **PRIVILEGED INSTRUCTIONS**
- **CONTROL SENSITIVE INSTRUCTIONS**
- **BEHAVIOR SENSITIVE INSTRUCTIONS**

https://en.wikipedia.org/wiki/Popek_and_Goldberg_virtualization_requirements

PRIVILEGED INSTRUCTIONS

- The **state management instructions plus the I/O instructions** are termed **privileged**.
  - I/O **instructions** and Halt **instructions**.
  - Turn off all Interrupts.
  - Set the Timer.
  - Context Switching.
  - Clear the Memory or Remove a process from the Memory.
  - Modify entries in Device-status table.
- Such instructions are usually directly executed only by the OS, because you do not want users to be able to access state associated with other computations.
- The OS has the ability to allow user programs (encapsulated as processes) to run the unprivileged instructions.
- But, as soon as the user program attempts to access an I/O operation or other privileged instruction, the OS traps the instruction, inspects the request, and, if the request proves to be acceptable, runs a program that executes a safe version of the operation.

CONTROL SENSITIVE INSTRUCTIONS

Those that attempt to change the configuration of resources in the system.

- PUSHF
- POPF
- SGDT (Store Global Descriptor Table Registers)
- SIDT (Store Interrupt Table Registers)

……

BEHAVIOR SENSITIVE INSTRUCTIONS

Those whose behavior or result depends on the configuration of resources (the content of the relocation register or the processor's mode)

POP

PUSH

CALL

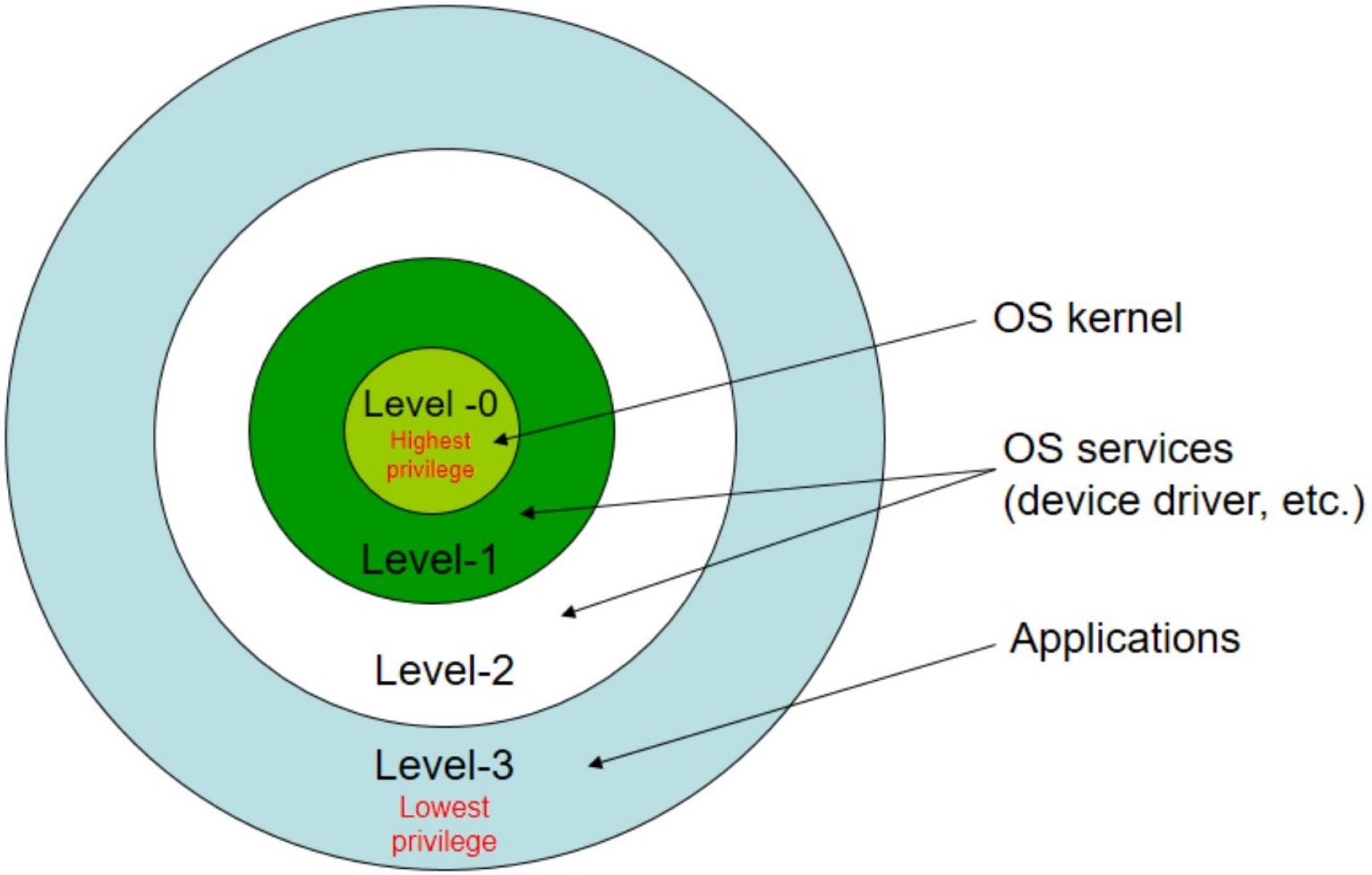
JMP

INT n

RET

MOV

Restrict on Intel IA32 Protection Rings



Full Virtualization

- A certain kind of virtual machine environment: one that provides a complete simulation of the underlying hardware.
- 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.
- Comprehensively simulate all computing elements as instruction set, main memory, interrupts, exceptions, and device access.
- Full virtualization is only possible given the right combination of hardware and software elements.

- 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.

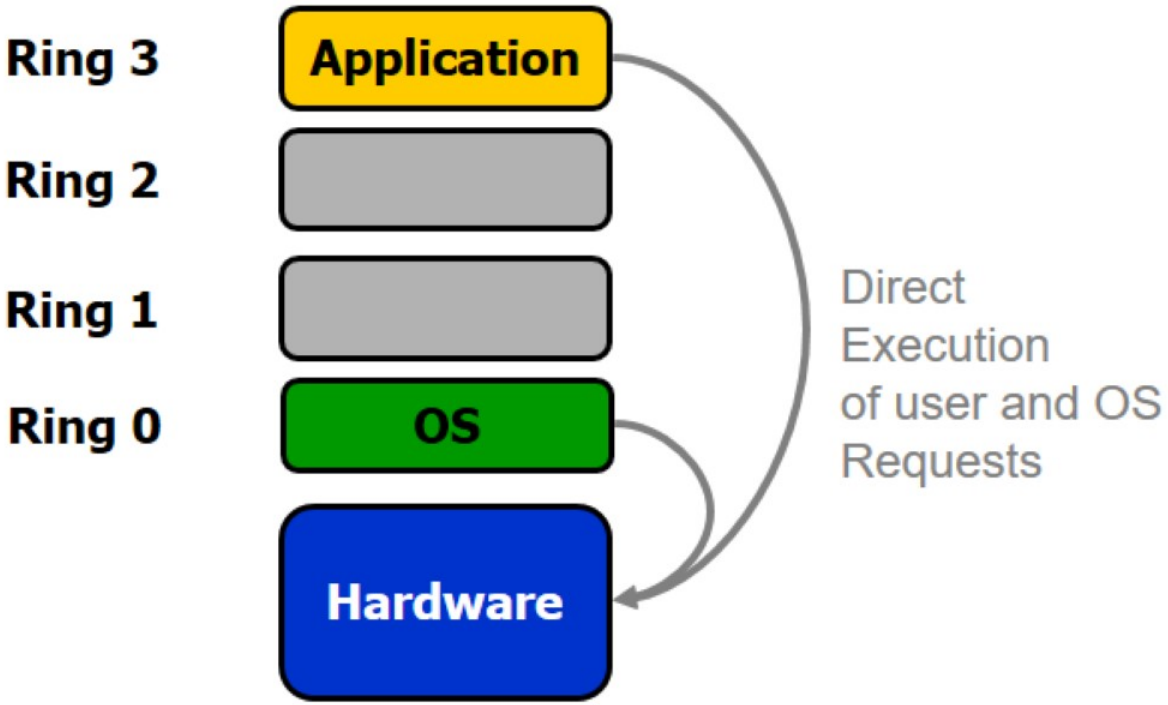
Full Virtualization -- challenge

- Security issues -- Interception
- Simulation of privileged operations -- I/O instructions
- The effects of every operation performed within a given virtual machine must be kept within that virtual machine – virtual operations cannot be allowed to alter the state of any other virtual machine, the control program, or the hardware.
- Some machine instructions can be executed directly by the hardware,
  - E.g., memory locations and arithmetic registers.
- But other instructions that would "pierce the virtual machine" cannot be allowed to execute directly; they must instead be trapped and simulated. Such instructions either access or affect state information that is outside the virtual machine.
- Some hardware is not easy to be used for full virtualization, e.g., x86

The Problem

- OS uses kernel mode / user mode to protect the OS.
  - System calls (privileged instructions) generate a trap (software interrupt) that forces a switch to kernel mode
  - These calls trigger sensitive instructions (I/O, MMU control, etc.) that must only be executed by the kernel

The challenges of x86 hardware virtualization



The Problems and the Solutions

- Originally designed for "personal use" (PC)
- Security problems caused by Interception and privileged operations becomes critical
- Solutions to Full virtualization of x86 CPU
  - Full description of operations of all x86 hardware (but they evolve)
  - Binary translation (almost established)
  - OS-assisted (or paravirtualization)
  - Hardware-assisted (future direction)

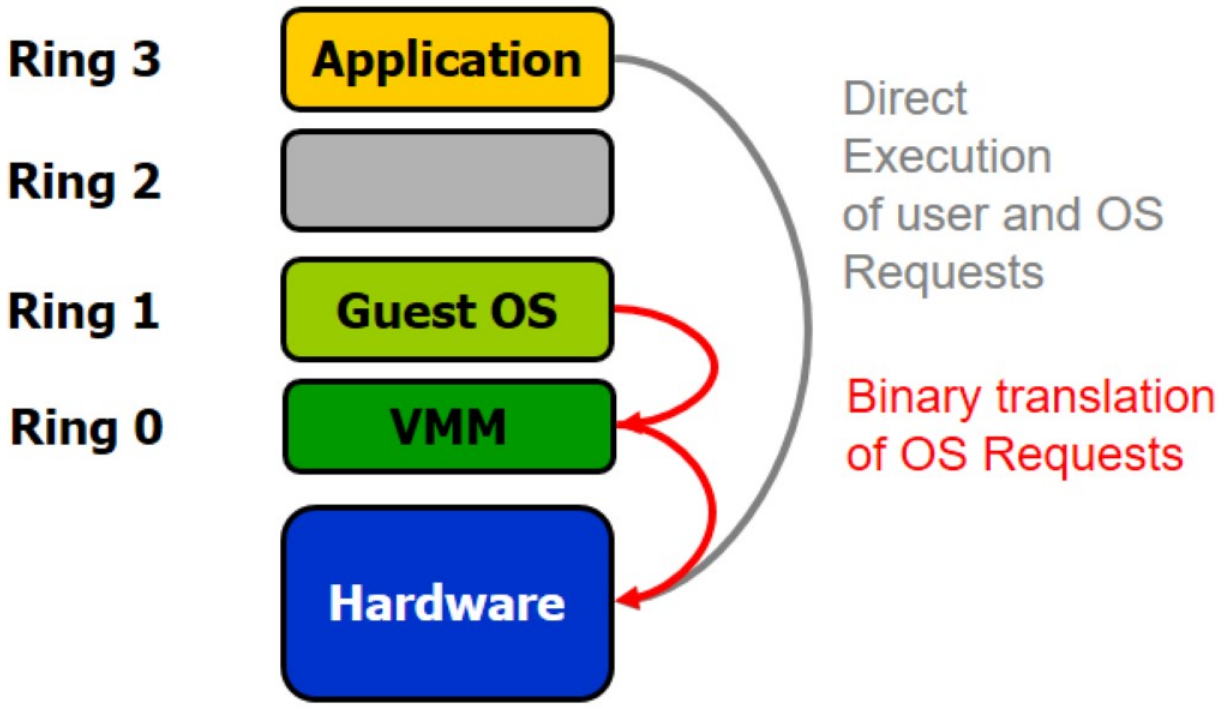
The Solution

Binary translation

- Kernel code of non-virtualizable instructions are translated to replace with new sequences of instructions that have the intended effect on the virtual hardware. Each virtual machine monitor provides each Virtual Machine with all the services of the physical system, including a virtual BIOS, virtual devices and virtualized memory management.
- This 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.
- 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.
- Examples
  - VMware
  - Microsoft Virtual Server

Binary translation

Examine the executable code of the virtual guest for "unsafe" instructions, translate them into "safe" instructions and execute the translated code



VMM: Virtual Machine Monitor

Full Virtualization

- As of full virtualization, the advantage is no need to modify OS. However, this approach of binary translation slows down the performance a lot.

So, ultimately it is the job of the VMM to check and translate everything. Guest OS is unaware and completely decoupled from the underlying hardware.

OS assisted (Paravirtualization)

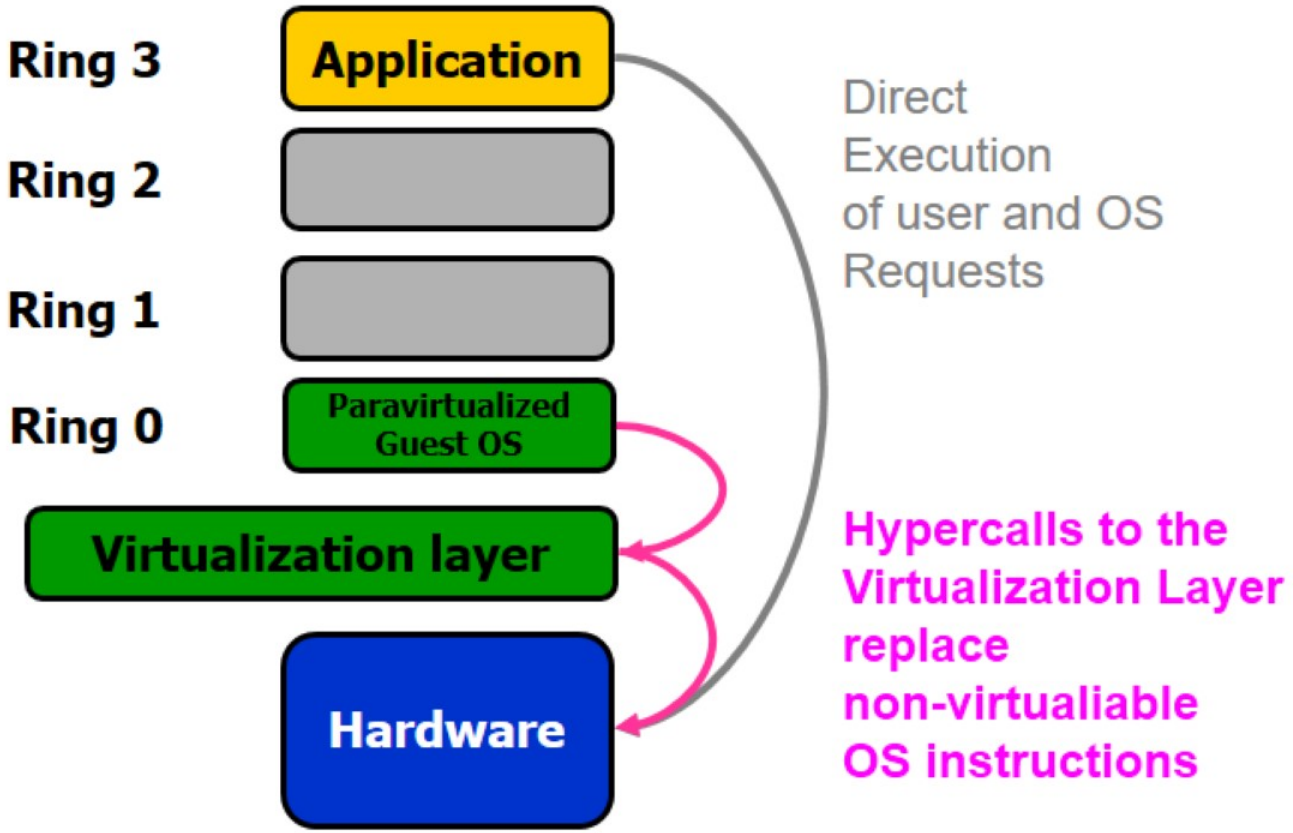
- Paravirtualization – via an modified OS kernel as guest OS
  - It is very difficult to build the more sophisticated binary translation support necessary for full virtualization.
  - Paravirtualization involves modifying the OS kernel to replace non-virtualizable instructions with hypercalls that communicate directly with the virtualization layer hypervisor.
  - The hypervisor also provides hypercall interfaces for other critical kernel operations such as memory management, interrupt handling and time keeping.
  - Paravirtualization is different from full virtualization, where the unmodified OS does not know it is virtualized and sensitive OS calls are trapped using binary translation.
  - Paravirtualization cannot support unmodified OS
- Example:
  - Xen -- modified Linux kernel and a version of Windows XP

Paravirtualization

- Modify Guest OS so that all calls to sensitive instructions are changed to hypervisor calls.
- Much easier (and more efficient) to modify source code than to emulate hardware instructions (as in binary translation).
- In effect, turns the hypervisor into a microkernel.
- Lower overheads and better performance

Rather than making VMM responsible for all the checks, now this approach makes guest OS to know about the limitations and accordingly responsible to tell VMM if any priviledged instruction is there by trapping to it

OS assisted (Paravirtualization)



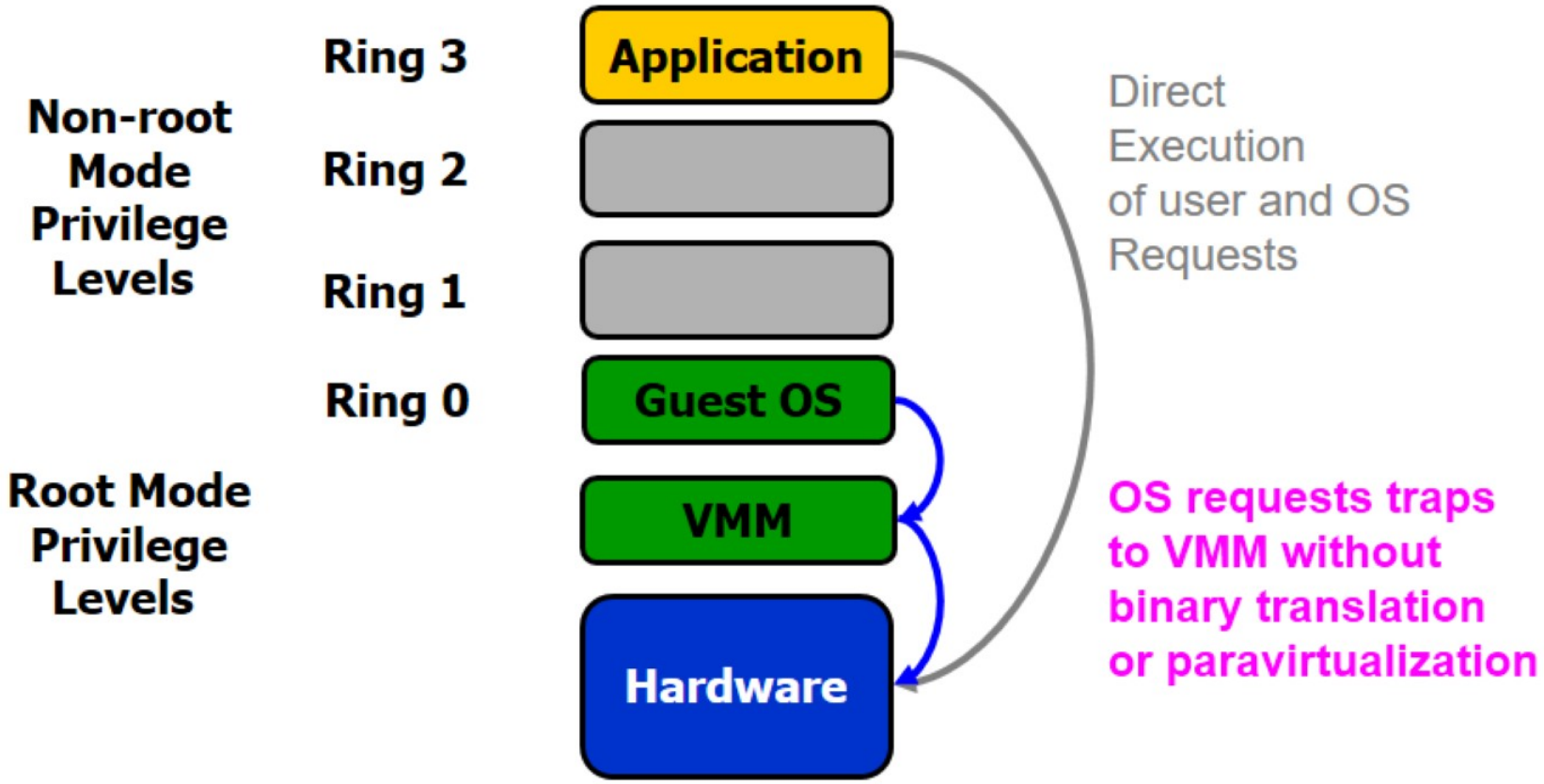
VMM: Virtual Machine Monitor

Hardware Assisted Virtualization

- 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

  target privileged instructions with a new CPU execution mode feature that al lows 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 paravirtualization.
- The guest state is stored in Virtual Machine Control Structures (VT-x) or Virtual Machine Control Blocks (AMD-V).
- High hypervisor to guest transition overhead and a rigid programming model

Hardware Assisted Virtualization



VMM: Virtual Machine Monitor

The Problem

- If our VM now runs in user space, we cannot run sensitive instructions in it, since those must trap to kernel space.
- Solved in 2005 with new CPUs
  - Intel Core 2 – VT (Virtualization Technology)
  - AMD Pacific – SVM (Secure Virtual Machine)
  - Provides new instructions that allow VM to capture traps

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 codified 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"

Virtualization

- 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
- Paravirtualization delivers performance benefits with maintenance costs
  - Xen
  - VMWare

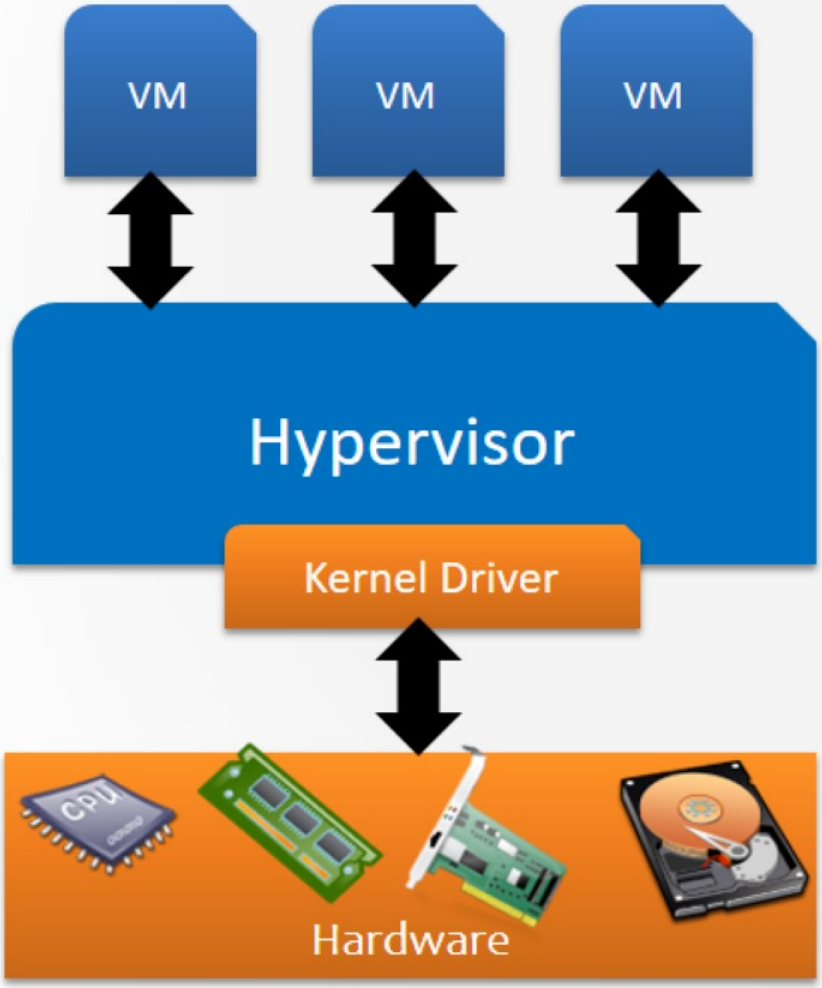
Popular hypervisors

- Xen
- KVM
- QEMU
- virtualBox
- VMWare
- Xen is the selected hypervisor of the project.

Implementation

- Type 1 Hypervisor
- Type 2 Hypervisor
- Paravirtualization

HYPERVISOR IMPLEMENTATION APPROACHES



Bare metal Approach

- Type I Hypervisor.
- Runs directly on the system hardware.
- May require hardware assisted virtualization technology support by the CPU.
- Limited set of hardware drivers provided by the hypervisor vendor.
- E.g.: Xen, VMWare ESXi

Bare Machine Hypervisor (Type I)

- Bare-metal virtualization means the hypervisor has direct access to hardware resources, which results in better performance, scalability and stability.
- A bare-metal virtualization hypervisor does not require admins to install a server operating system first.
- One disadvantage of a bare-metal virtualization hypervisor, however, is that hardware support is typically more limited, because the hypervisor usually has limited device drivers built into it.
- Bare-metal virtualization is well suited for enterprise data centers, because it usually comes with advanced features for resource management, high availability and security.
- Admins can centrally manage this kind of virtualization hypervisor, which is very critical with multiple applications running
- VMware ESX and ESXi
- Microsoft Hyper-V
- Citrix Systems XenServer

Type 1 Hypervisors

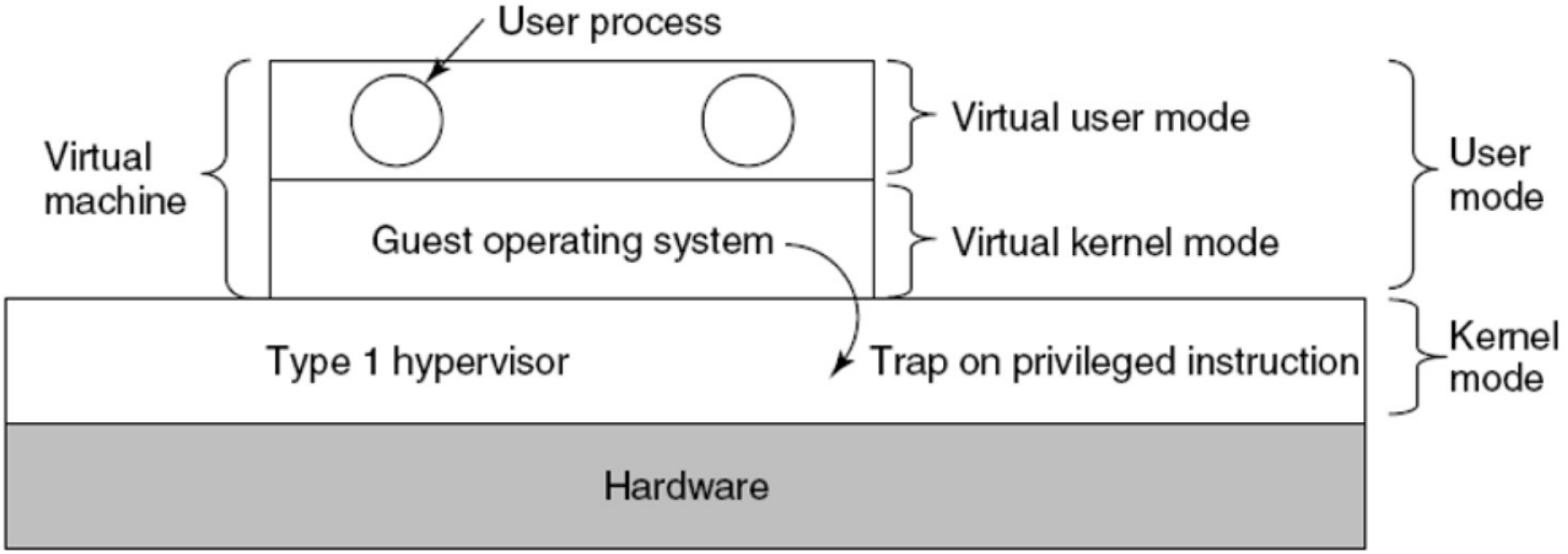
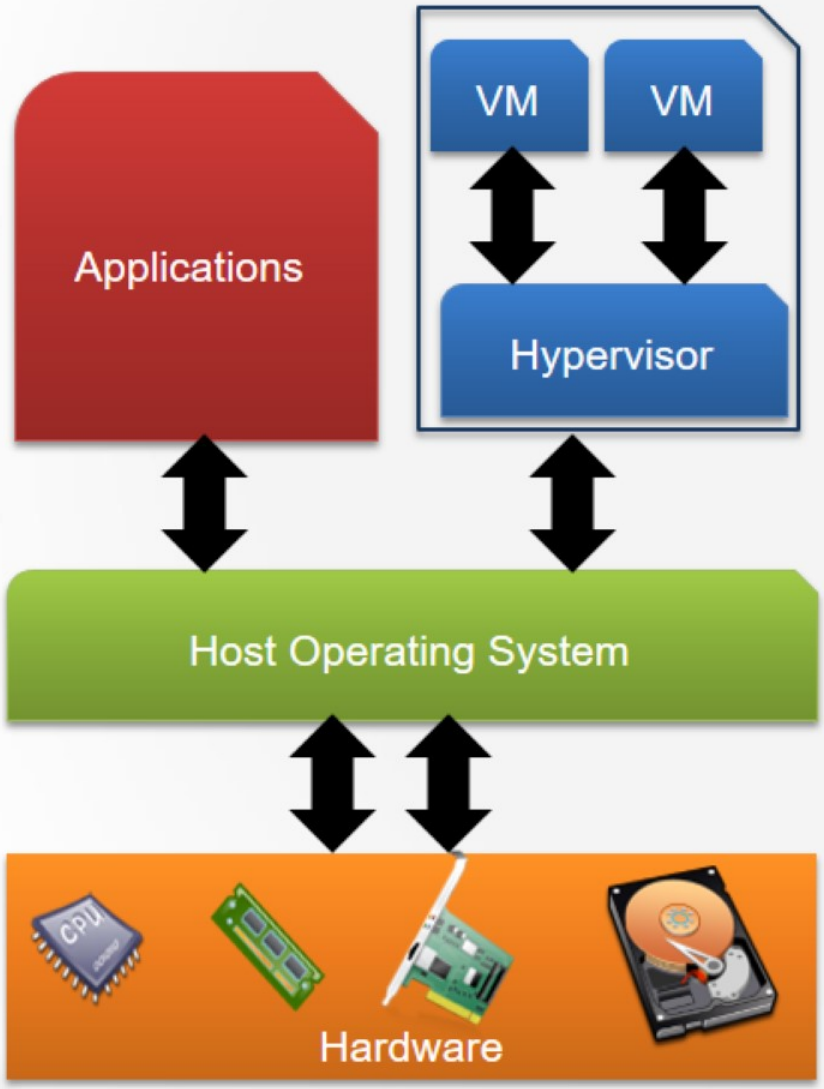


Figure 8-26. When the operating system in a virtual machine executes a kernel-only instruction, it traps to the hypervisor if virtualization technology is present.

UNF University of NORTH FLORIDA

HYPERVISOR IMPLEMENTATION APPROACHES



Hosted Approach

- Type II Hypervisor.
- Runs virtual machines on top of a host OS (windows, Unix etc.)
- Relies on host OS for physical resource management.
- Host operating system provides drivers for communicating with the server hardware.
- E.g.: VirtualBox

28

Hosted virtualization hypervisors (Type II)

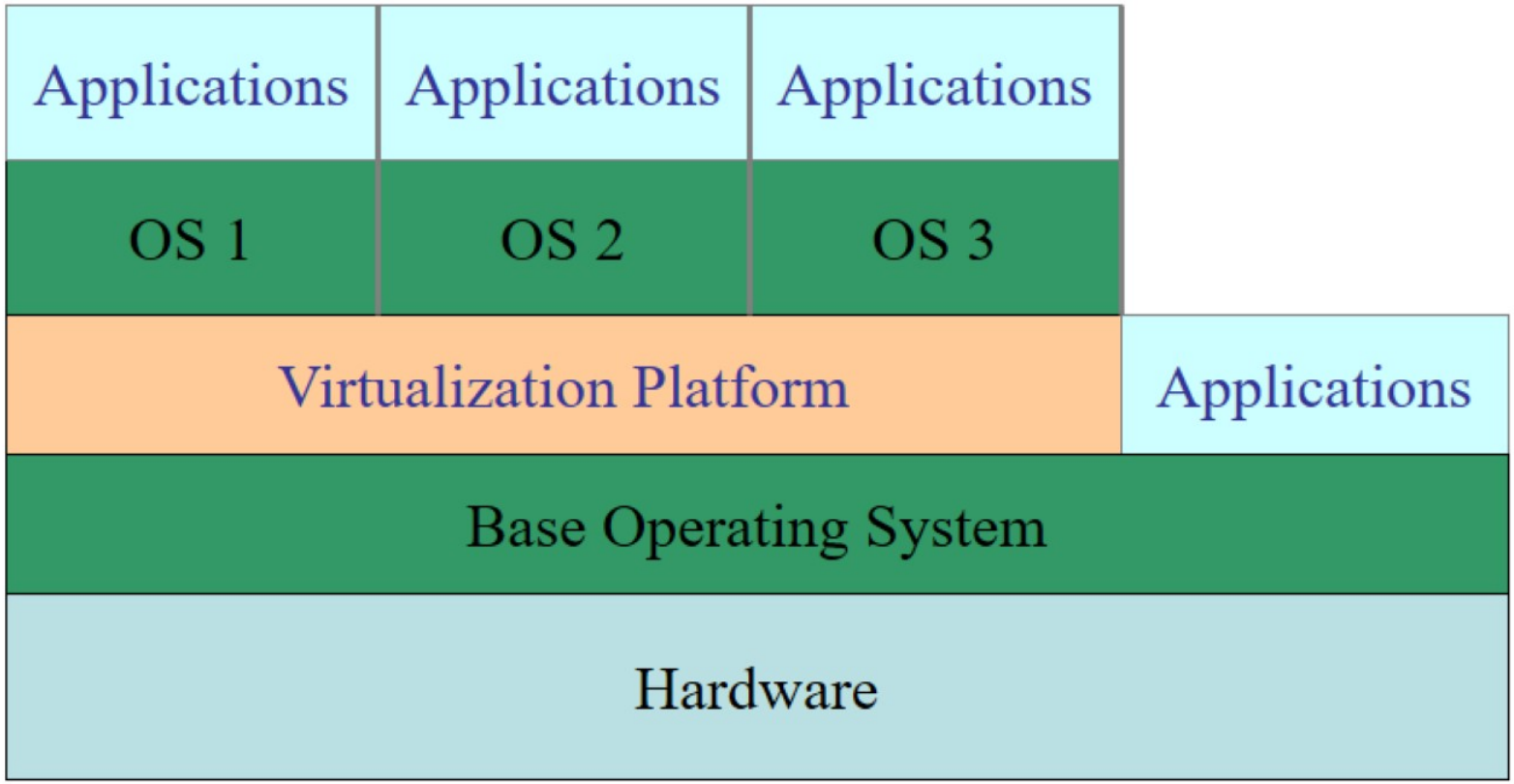
- Unlike the bare-metal virtualization hypervisor, a hosted hypervisor requires you to first install an OS.
- These hypervisors are basically like applications that install on a guest OS.
- This approach provides better hardware compatibility than bare-metal virtualization, because the OS is responsible for the hardware drivers instead of the hypervisor.
- Hosted hypervisors are common for desktops, because they allow you to run multiple OSes. These virtualization hypervisor types are also popular for developers, to maintain application compatibility on modern OSes.
- VMware Workstation, Server, Player and Fusion
- Oracle VM VirtualBox, Microsoft Virtual PC, Parallels Desktop

Type 2 Hypervisor

- Runs from within a OS.
- Supports guest OSs above it.
  - Boot from CD to load new OS
  - Read in code, looking for basic blocks
  - Then inspect basic block to find sensitive instructions. If found, replace with VM call (process called binary translation)
  - Then, cache block and execute.
  - Eventually all basic blocks will be modified and cached, and will run at near native speed.

30

Type 2 Hypervisor



31

Hosted virtualization hypervisors

- But, as with the bare-metal hypervisor, there are disadvantages.
- A hosted virtualization hypervisor does not have direct access to hardware and must go through the OS, which increases resource overhead and can degrade virtual machine (VM) performance.
- Also, because there are typically many services and applications running on the host OS, the hypervisor often steals resources from the VMs running on it.

VMMs

- **Traditional:** Also called bare metal VMM
  - A software that runs directly on the host machine
  - Examples: *VMWare, ESX, ESXi, Denali*
- **Hybrid:** The VMMs shares the hardware with the existing OS
  - Example: *VMWare workstation*
- **Hosted:** The VM runs on top of the existing OS
  - Easier to build and install
  - Could use several components of the host OS such as scheduler, pager and IO drivers rather than providing its own
  - Price of increased overhead and associated performance penalty as requests not handled directly by the VMM but by the host OS
  - Example: User-mode *Linux*

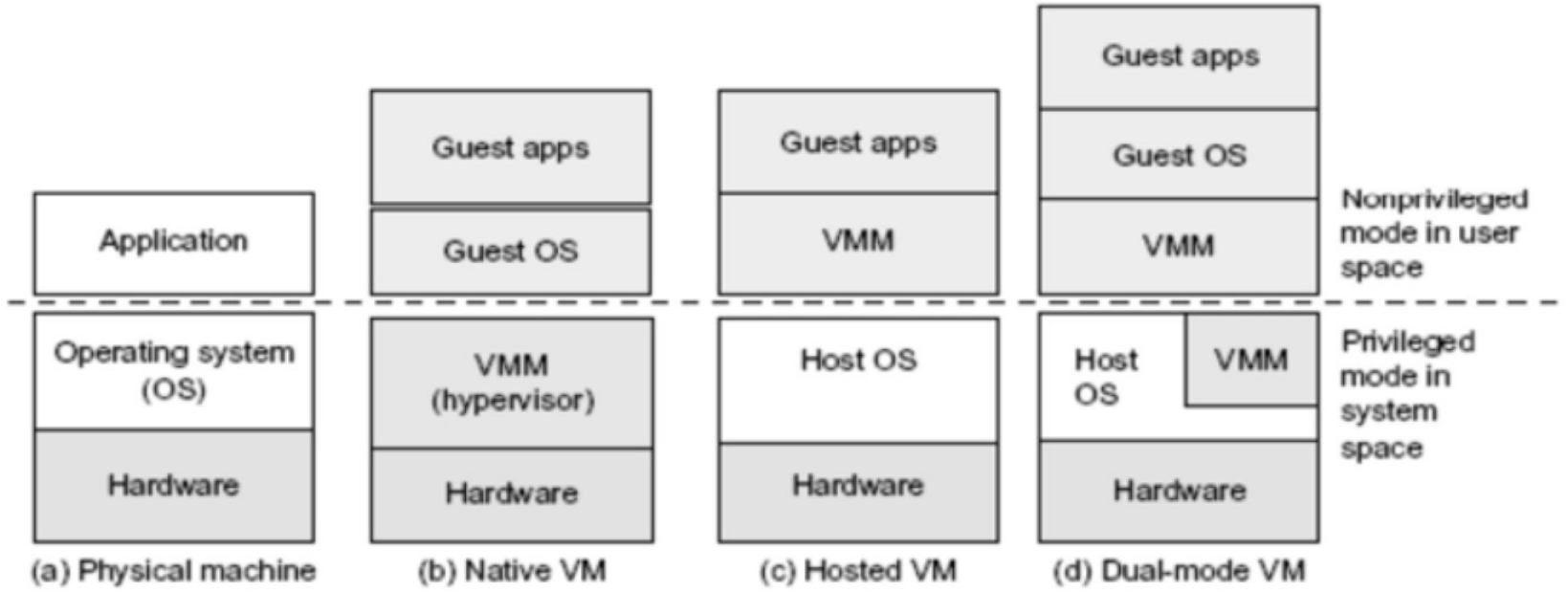


FIGURE 1.12

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

Types of Virtualization

- Virtual memory
- Desktop virtualization
- Platform virtualization
  - Full virtualization
  - Paravirtualization
  - Hardware-assisted virtualization
  - Partial virtualization
  - OS-level virtualization
  - Hosted environment (e.g. User-mode Linux)
- Storage virtualization
- Network virtualization
- Application virtualizationPortable application
  - Cross-platform virtualization
  - Emulation or simulation
  - Hosted Virtual Desktop

Category in Wiki

THANK YOU

https://images.app.goo.gl/U49g5dYRga5x7em57