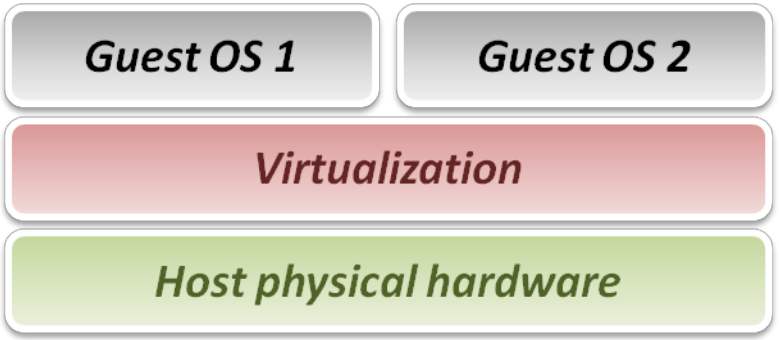
Cloud Computing

Server Virtualization

CPU VIRTUALIZATION

Virtualization Technique

- From emulation to virtualization :
  - While emulation techniques emulate guest on host, whose ISA differ from guest, in virtualization techniques, guest and host have the same ISA.
  - Some problems in emulation will not exist in virtualization :
    - No need to translate each binary instruction to host ISA.
    - No need to worry about unmatched special register mapping.
  - Some new problems didn't exist in emulation exist now :
    - Instruction privileges should be well-controlled.
- Goal of virtualization :
  - Run or simulate all instructions of gust OS.



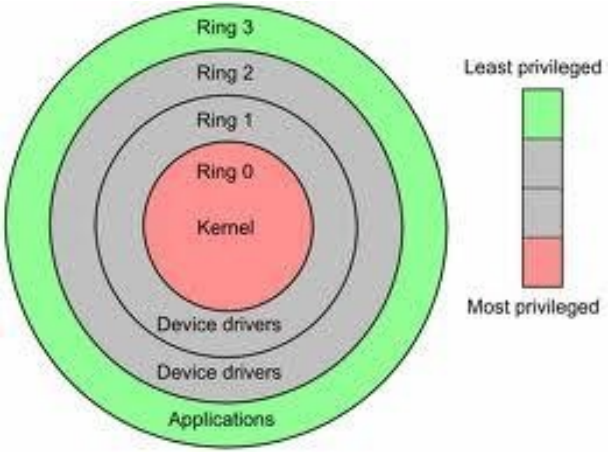
Virtualization Technique

- Virtualization requirements from Popek and Goldberg :
  - Popek and Goldberg provide a set of sufficient conditions for a computer architecture to efficiently support system virtualization.
  - Popek and Goldberg provide guidelines for the design of virtualized computer architectures.

- In Popek and Goldberg terminology, a VMM must present all three properties :
  - Equivalence (Same as real machine)
  - Resource control (Totally control)
  - Efficiency (Native execution)

CPU Architecture

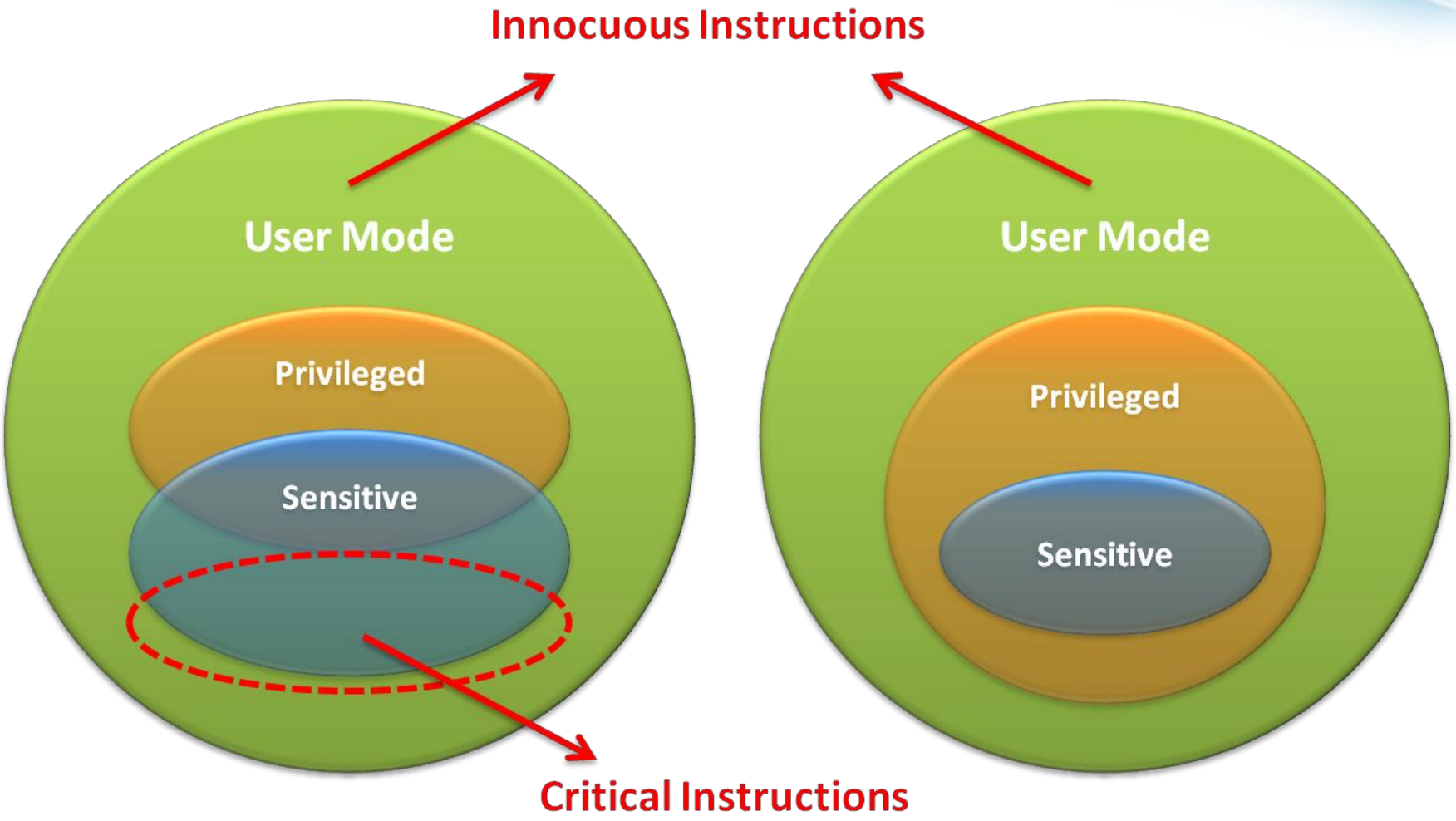
- Modern CPU status is usually classified as several modes.
- In general, we conceptually divide them into two modes :
  - Kernel mode (Ring 0)
    - CPU may perform any operation allowed by its architecture, including any instruction execution, IO operation, area of memory access, and so on.
    - Traditional OS kernel runs in Ring 1 mode.
  - User mode (Ring 1 ~ 3)
    - CPU can typically only execute a subset of those available instructions in kernel mode.
    - Traditional application runs in Ring 3 mode.



CPU Architecture

- By the classification of CPU modes, we divide instructions into following types :
  - Privileged instruction
    - Those instructions that trap if the machine is in user mode and do not trap if the machine is in kernel mode.
  - Sensitive instructions
    - Those instructions that interact with hardware, which include control-sensitive and behavior-sensitive instructions.
  - Innocuous instruction
    - All other instructions.
  - Critical instruction
    - Those sensitive but not privileged instructions.

CPU Architecture



CPU Architecture

- What is trap ?
  - When CPU is running in user mode, some internal or external events, which need to be handled in kernel mode, take place.
  - Then CPU will jump to hardware exception handler vector, and execute system operations in kernel mode.
- Trap types :
  - System Call
    - Invoked by application in user mode.
    - For example, application ask OS for system IO.
  - Hardware Interrupts
    - Invoked by some hardware events in any mode.
    - For example, hardware clock timer trigger event.
  - Exception
    - Invoked when unexpected error or system malfunction occur.
    - For example, execute privilege instructions in user mode.

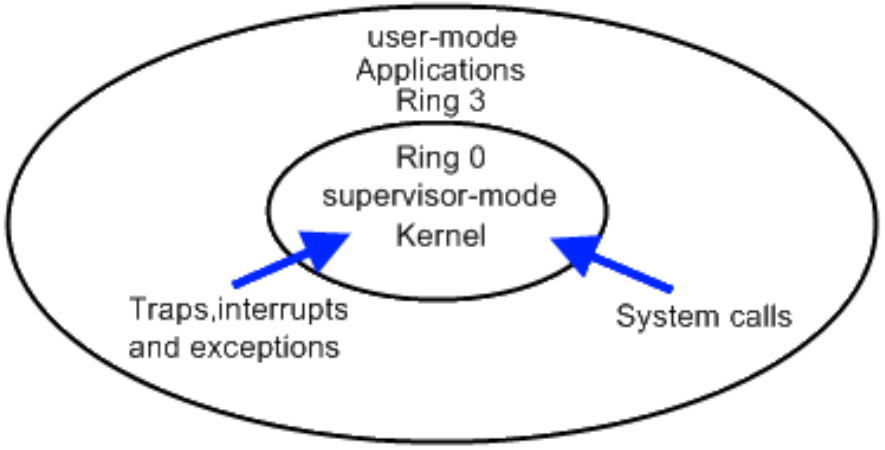
Trap and Emulate Model

- If we want CPU virtualization to be efficient, how should we implement the VMM ?
  - We should make guest binaries run on CPU as fast as possible.
  - Theoretically speaking, if we can run all guest binaries natively, there will NO overhead at all.
  - But we cannot let guest OS handle everything, VMM should be able to control all hardware resources.

- Solution :
  - Ring Compression
    - Shift traditional OS from kernel mode(Ring 0) to user mode(Ring 1), and run VMM in kernel mode.
    - Then VMM will be able to intercept all trapping event.

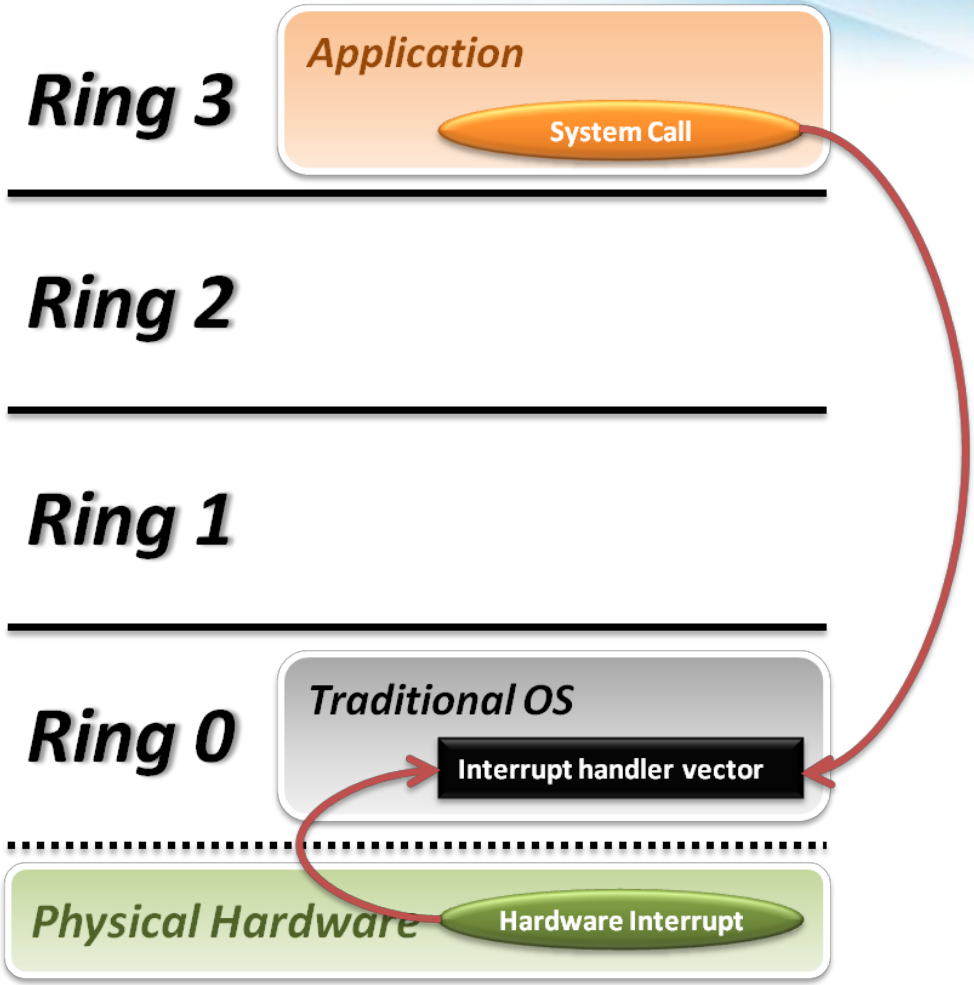
Trap and Emulate Model

- VMM virtualization paradigm *(trap and emulate)* :
  1. Let normal instructions of guest OS run directly on processor in user mode.
  2. When executing privileged instructions, hardware will make processor trap into the VMM.
  3. The VMM emulates the effect of the privileged instructions for the guest OS and return to guest.



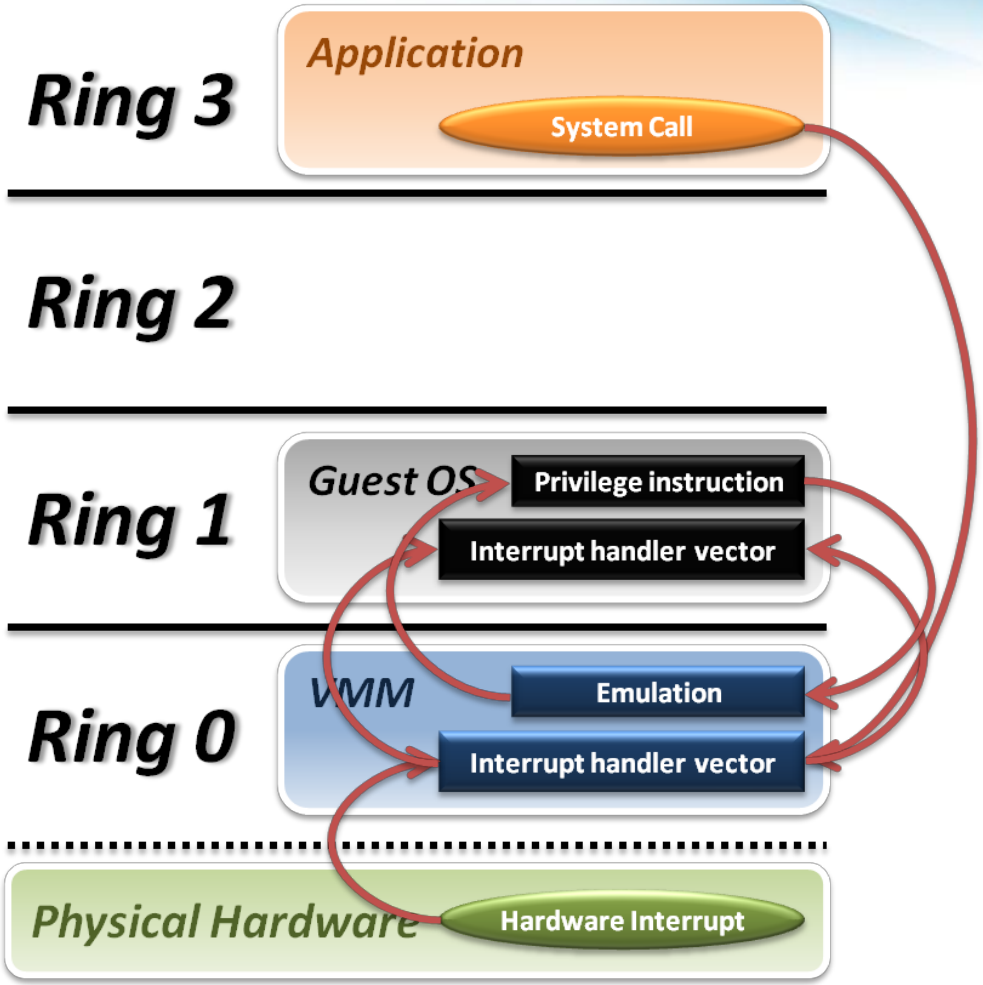
Trap and Emulate Model

- Traditional OS :
  - When application invoke a system call :
    - CPU will trap to interrupt handler vector in OS.
    - CPU will switch to kernel mode (Ring 0) and execute OS instructions.
  - When hardware event :
    - Hardware will interrupt CPU execution, and jump to interrupt handler in OS.



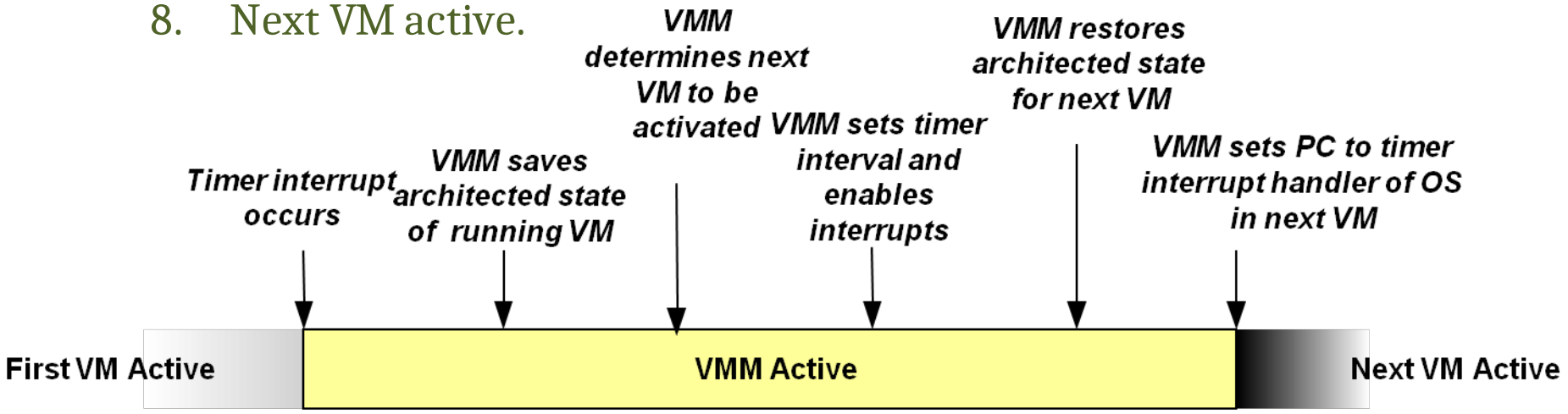
Trap and Emulate Model

- VMM and Guest OS :
  - System Call
    - CPU will trap to interrupt handler vector of VMM.
    - VMM jump back into guest OS.
  - Hardware Interrupt
    - Hardware make CPU trap to interrupt handler of VMM.
    - VMM jump to corresponding interrupt handler of guest OS.
  - Privilege Instruction
    - Running privilege instructions in guest OS will be trapped to VMM for instruction emulation.
    - After emulation, VMM jump back to guest OS.



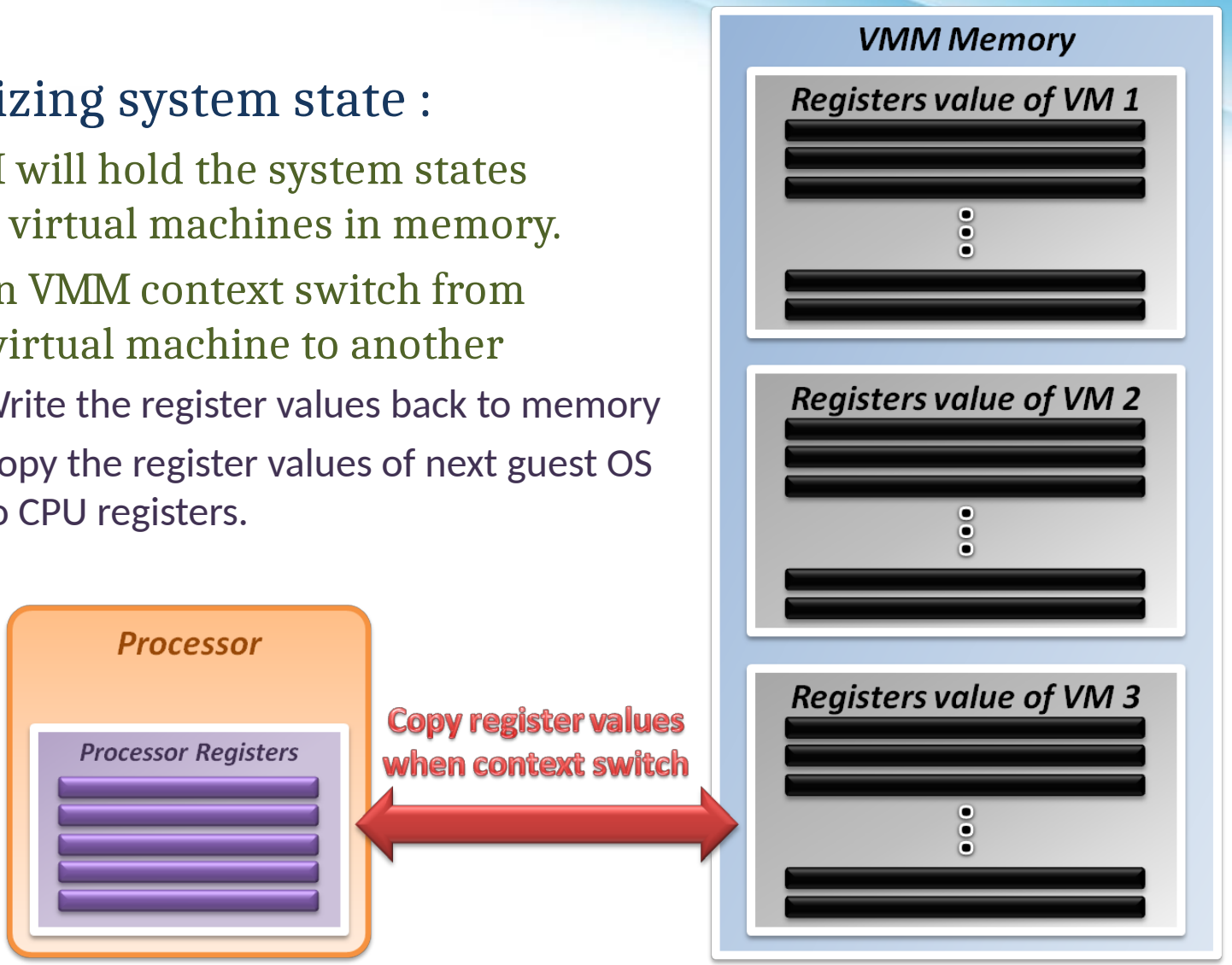
Context Switch

- Steps of VMM switch different virtual machines :
  1. Timer Interrupt in running VM.
  2. Context switch to VMM.
  3. VMM saves state of running VM.
  4. VMM determines next VM to execute.
  5. VMM sets timer interrupt.
  6. VMM restores state of next VM.
  7. VMM sets PC to timer interrupt handler of next VM.
  8. Next VM active.



System State Management

- Virtualizing system state :
  - VMM will hold the system states of all virtual machines in memory.
  - When VMM context switch from one virtual machine to another
    - Write the register values back to memory
    - Copy the register values of next guest OS to CPU registers.



Virtualization Theorem

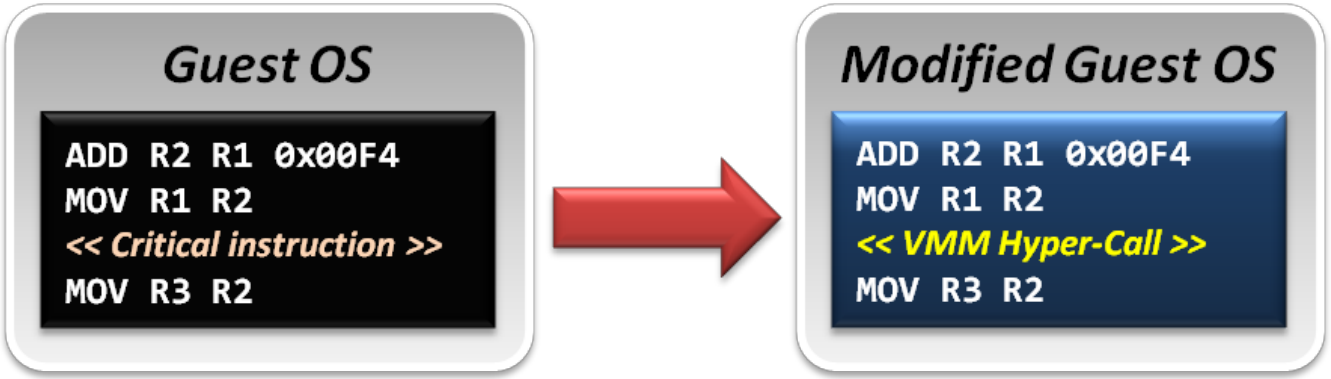
- Subset theorem :
  - For any conventional third-generation computer, a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.
- Recursive Emulation :
  - A conventional third-generation computer is recursively virtualizable if
    - It is virtualizable
    - VMM without any timing dependencies can be constructed for it.
- Under this theorem, x86 architecture cannot be virtualized directly. Other techniques are needed.

Virtualization Techniques

- How to virtualize unvirtualizable hardware :
  - Para-virtualization
    - Modify guest OS to skip the critical instructions.
    - Implement some hyper-calls to trap guest OS to VMM.
  - Binary translation
    - Use emulation technique to make hardware virtualizable.
    - Skip the critical instructions by means of these translations.
  - Hardware assistance
    - Modify or enhance ISA of hardware to provide virtualizable architecture.
    - Reduce the complexity of VMM implementation.

Para-Virtualization

- Para-Virtualization implementation :
  - In para-virtualization technique, guest OS should be modified to prevent invoking critical instructions.
  - Instead of knowing nothing about hypervisor, guest OS will be aware of the existence of VMM, and collaborate will VMM smoothly.
  - VMM will provide the hyper-call interfaces, which will be the communication channel between guest and host.



Binary Translation

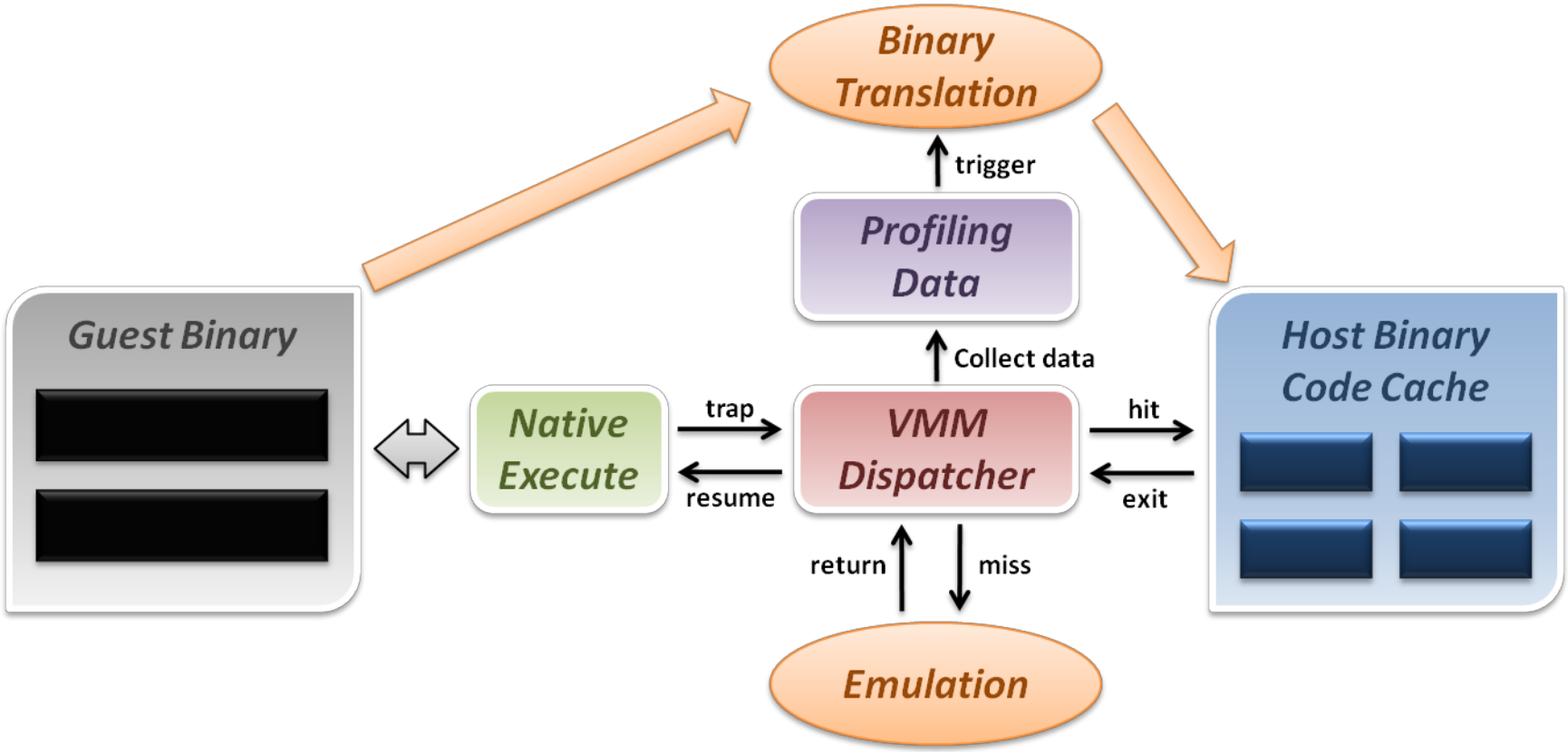
- In emulation techniques :
  - Binary translation module is used to optimize binary code blocks, and translate binaries from guest ISA to host ISA.
- In virtualization techniques :
  - Binary translation module is used to skip or modify the guest OS binary code blocks which include critical instructions.
  - Translate those critical instructions into some privilege instructions which will trap to VMM for further emulation.

Binary Translation

- Static approach vs. Dynamic approach :
  - Static binary translation
    - The entire executable file is translated into an executable of the target architecture.
    - This is very difficult to do correctly, since not all the code can be discovered by the translator.
  - Dynamic binary translation
    - Looks at a short sequence of code, typically on the order of a single basic block, translates it and caches the resulting sequence.
    - Code is only translated as it is discovered and when possible, branch instructions are made to point to already translated and saved code.

Binary Translation

- Dynamic binary translation and optimization
  - VMM can dynamically translate binary code and collect profiling data for further optimization.



Some Difficulties

- Difficulties of binary translation :
  - Self-modifying code
    - If guest OS will modify its own binary code in runtime, binary translation need to flush the responding code cache and retranslate the code block.
  - Self-reference code
    - If guest code need to reference(read) its own binary code in runtime, VMM need to make it referring back to original guest binaries location.
  - Real-time system
    - For some timing critical guest OS, emulation environment will lose precise timing, and this problem cannot be perfectly solved yet.
- Difficulty of para-virtualization :
  - Guest OS modification
    - User should at least has the source code of guest OS and modify its kernel; otherwise, para-virtualization cannot be used.

CPU VIRTUALIZATION

Hardware Solution

- Why are there so many problems and difficulties ?
  - Critical instructions do not trap in user mode.
  - Even if we make those critical instructions trap, their semantic may be also changed; which is not acceptable.
- In short, legacy processors did not design for virtualization purpose at the beginning.
  - If processor can be aware of the different behaviors between guest and host, the VMM design will be more efficient and simple.

Hardware Solution

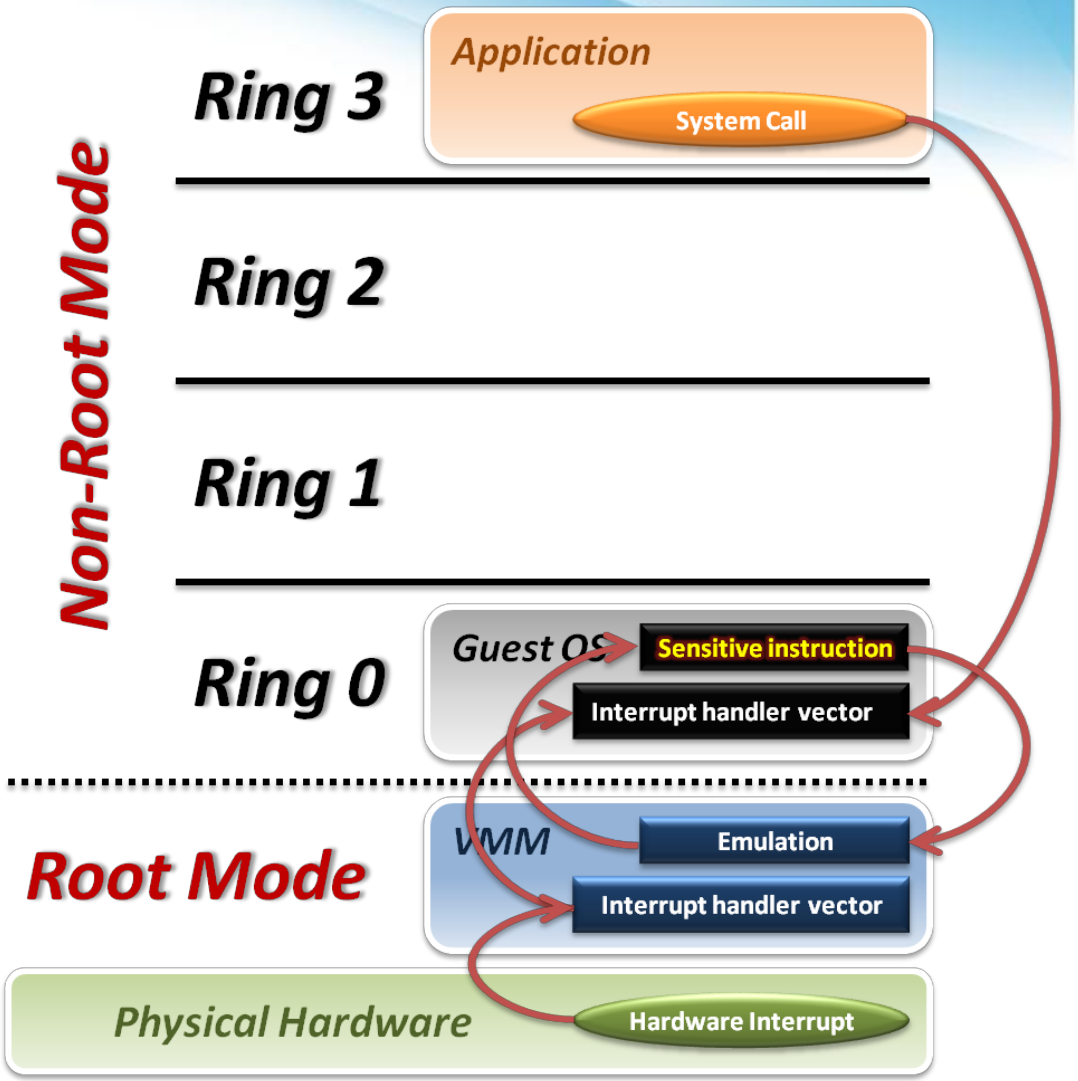
- Let's go back to trap model :
  - Some trap types do not need the VMM involvement.
    - For example, all system calls invoked by application in guest OS should be caught by gust OS only. There is no need to trap to VMM and then forward it back to guest OS, which will introduce context switch overhead.
  - Some critical instructions should not be executed by guest OS.
    - Although we make those critical instructions trap to VMM, VMM cannot identify whether this trapping action is caused by the emulation purpose or the real OS execution exception.
- Solution :
  - We need to redefine the semantic of some instructions.
  - We need to introduce new CPU control paradigm.

Intel VT-x

- In order to straighten those problems out, Intel Company introduces one more operation mode of x86 architecture.
  - VMX Root Operation (Root Mode)
    - All instruction behaviors in this mode are no different to traditional ones.
    - All legacy software can run in this mode correctly.
    - VMM should run in this mode and control all system resources.
  - VMX Non-Root Operation (Non-Root Mode)
    - All sensitive instruction behaviors in this mode are redefined.
    - The sensitive instructions will trap to Root Mode.
    - Guest OS should run in this mode and be fully virtualized through typical "*trap and emulation model*".

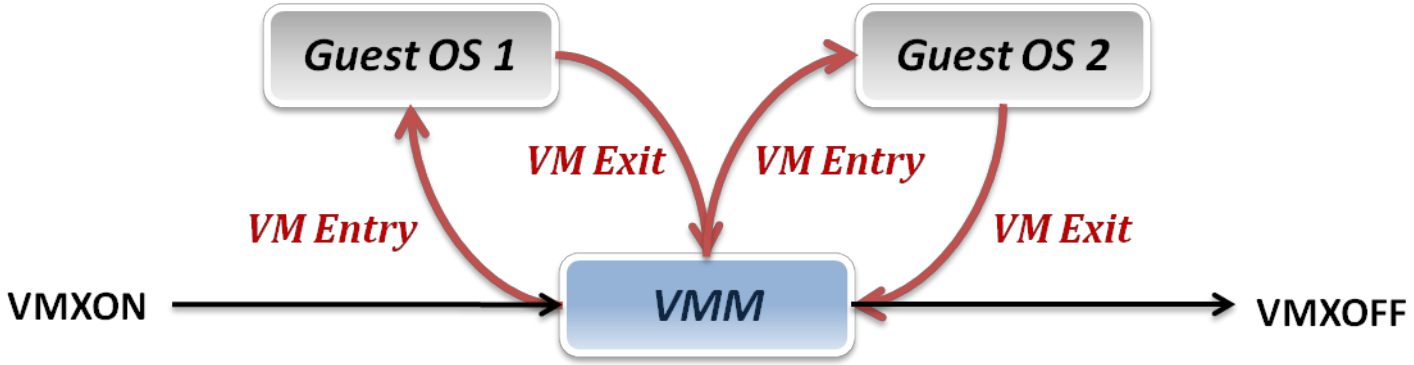
Intel VT-x

- VMM with VT-x :
  - System Call
    - CPU will directly trap to interrupt handler vector of guest OS.
  - Hardware Interrupt
    - Still, hardware events need to be handled by VMM first.
  - Sensitive Instruction
    - Instead of trap all privilege instructions, running guest OS in Non-root mode will trap sensitive instruction only.



Context Switch

- VMM switch different virtual machines with Intel VT-x :
  - **VMXON/VMXOFF**
    - These two instructions are used to turn on/off CPU Root Mode.
  - VM Entry
    - This is usually caused by the execution of **VMLAUNCH/VMRESUME** instructions, which will switch CPU mode from Root Mode to Non-Root Mode.
  - VM Exit
    - This may be caused by many reasons, such as hardware interrupts or sensitive instruction executions.
    - Switch CPU mode from Non-Root Mode to Root Mode.



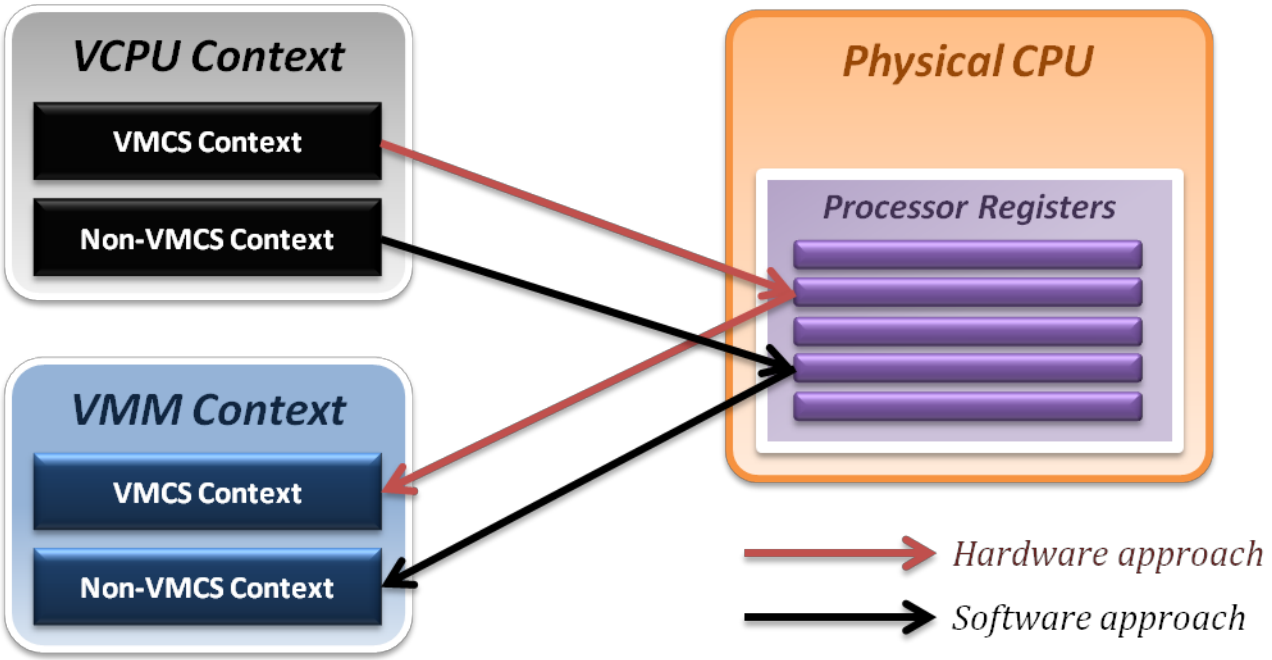
System State Management

- Intel introduces a more efficient hardware approach for register switching, **VMCS** (*Virtual Machine Control Structure*) :
  - State Area
    - Store host OS system state when VM-Entry.
    - Store guest OS system state when VM-Exit.
  - Control Area
    - Control instruction behaviors in Non-Root Mode.
    - Control VM-Entry and VM-Exit process.
  - Exit Information
    - Provide the VM-Exit reason and hardware other information.

- Whenever VM Entry or VM Exit occur, CPU will automatically read or write corresponding information into VMCS.

System State Management

- Binding virtual machine to virtual CPU
  - VCPU (Virtual CPU) contains two parts
    - VMCS maintains virtual system states, which is approached by hardware.
    - Non-VMCS maintains other non-essential system information, which is approach by software.
  - VMM needs to handle Non-VMCS part.



CPU Virtualization Summary

- Emulation technique
  - Interpretation and binary translation approaches
  - System state mapping and performance issue
    - Translation chaining, Dynamic binary optimization
- Virtualization technique
  - Modern CPU architecture
  - Trap and emulation model
  - Critical instruction issue
    - Para-virtualization, Dynamic binary translation
- Hardware assistance
  - Intel VT-x approach
    - Root Mode & Non-Root Mode

Shadow page table

Hardware assistance

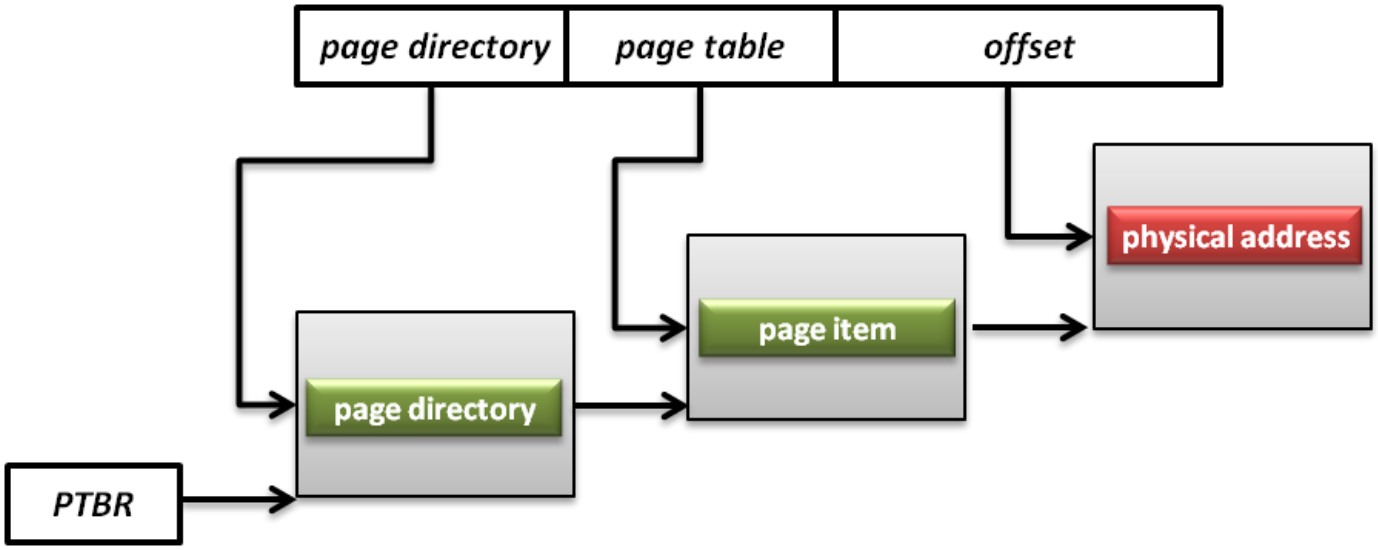
MEMORY VIRTUALIZATION

Memory Virtualization

- Memory management in OS
  - Traditionally, OS fully controls all physical memory space and provide a continuous addressing space to each process.
  - In server virtualization, VMM should make all virtual machines share the physical memory space without knowing the fact.
- Goals of memory virtualization :
  - Address Translation
    - Control table-walking hardware that accesses translation tables in main memory.
  - Memory Protection
    - Define access permission which uses the Access Control Hardware.
  - Access Attribute
    - Define attribute and type of memory region to direct how memory operation to be handled.

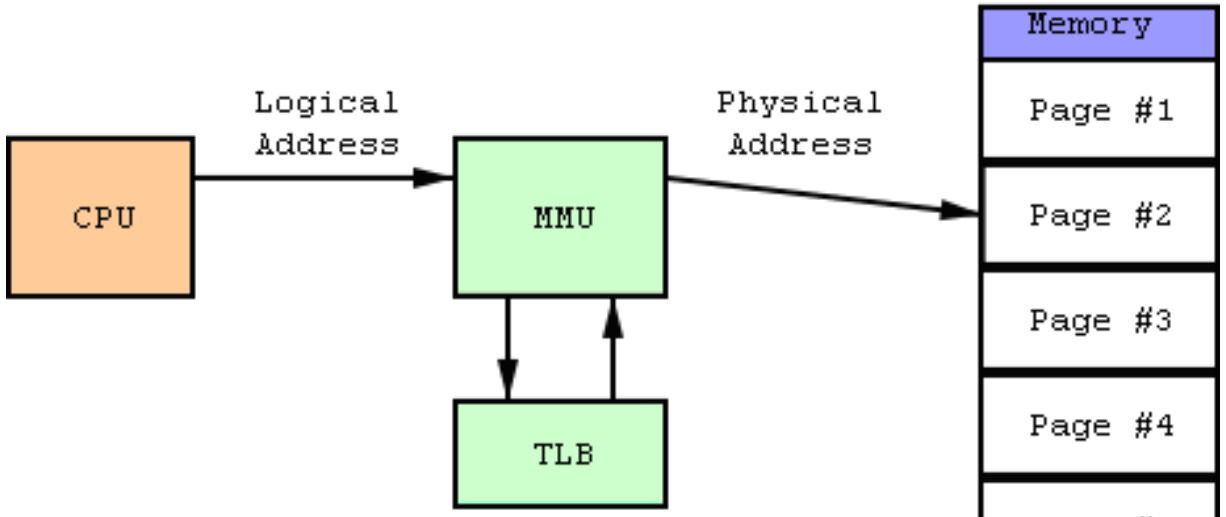
Memory Architecture

- Memory Management Unit (MMU)
  - What is MMU ?
    - A computer hardware component responsible for handling accesses to memory requested by the CPU.
    - Its functions include translation of virtual addresses to physical addresses, memory protection, cache control, bus arbitration and etc.
  - What is PTBR ?
    - Page Table Base Register (PTBR) is a register point to the base of page table for MMU.



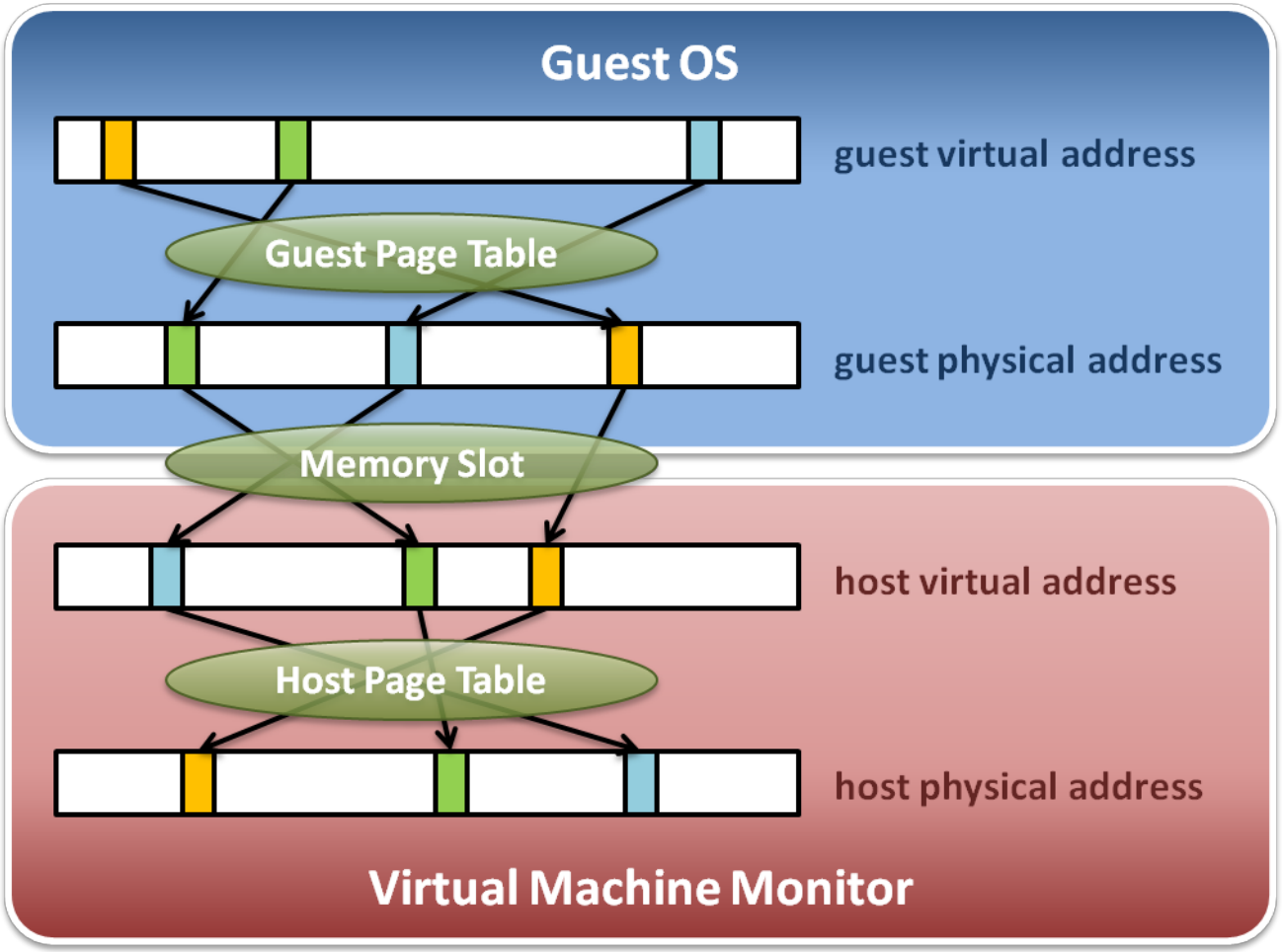
Memory Architecture

- Translation Lookaside Buffer (TLB)
  - What is TLB ?
    - A CPU cache that memory management hardware uses to improve virtual address translation speed.



Memory Virtualization

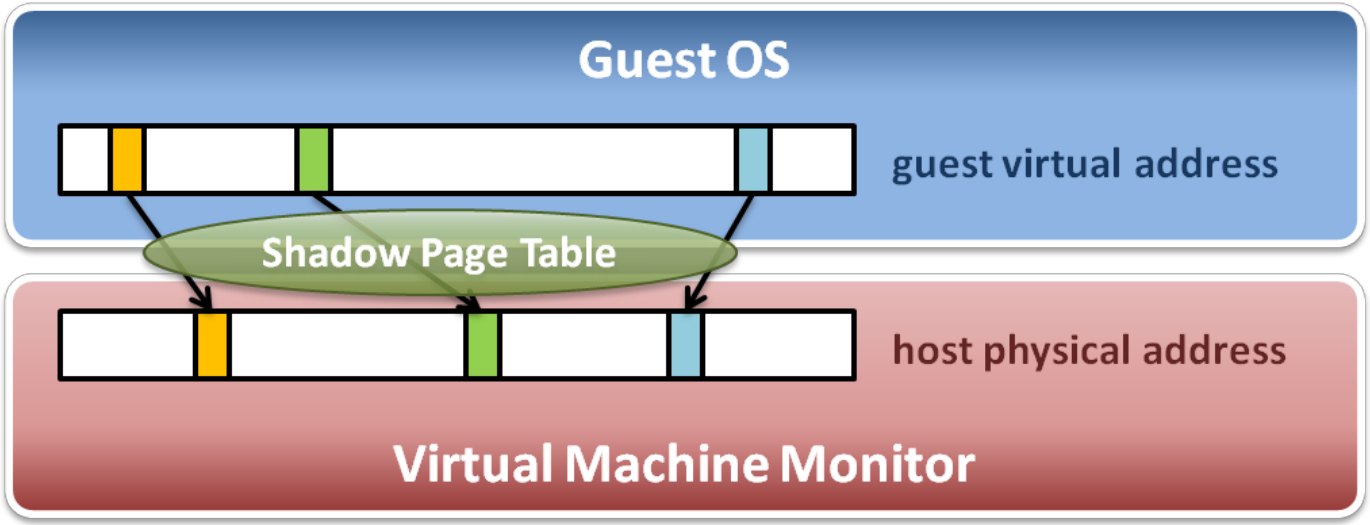
- Memory virtualization architecture



Memory Virtualization

- The performance drop of memory access is usually unbearable. VMM needs further optimization.

- VMM maintains shadow page tables :
  - Direct virtual-to-physical address mapping
  - Use hardware TLB for address translation

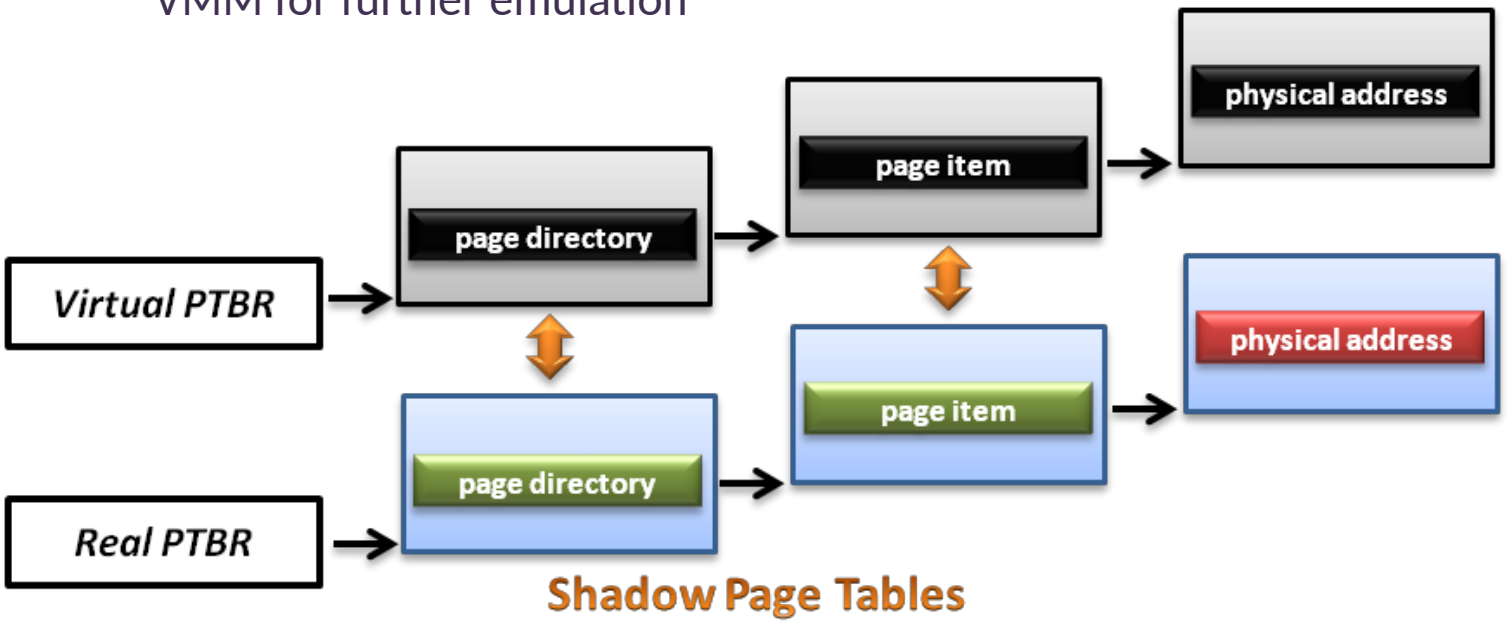


Shadow Page Table

- Map guest virtual address to host physical address
  - Shadow page table
    - Guest OS will maintain its own virtual memory page table in the guest physical memory frames.
    - For each guest physical memory frame, VMM should map it to host physical memory frame.
    - Shadow page table maintains the mapping from guest virtual address to host physical address.
  - Page table protection
    - VMM will apply write protection to all the physical frames of guest page tables, which lead the guest page table write exception and trap to VMM.

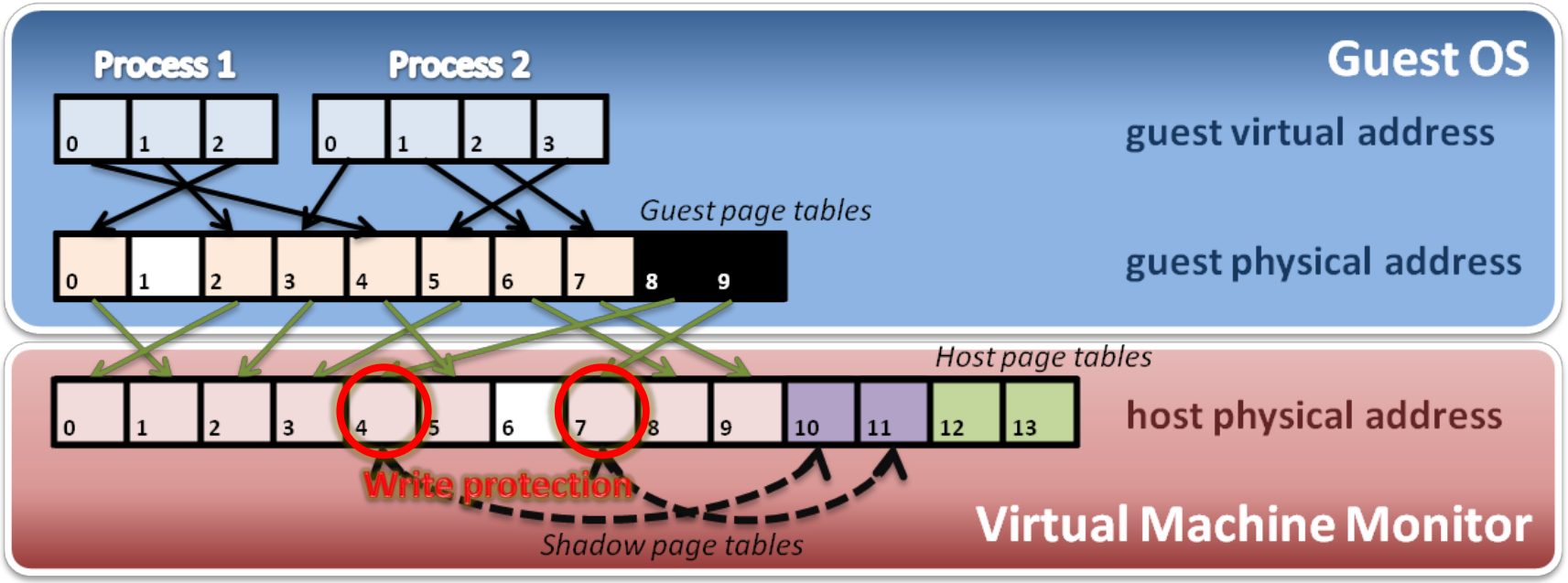
Shadow Page Table

- How does this technique work ?
  - VMM should make MMU virtualized
    - VMM manages the real PTBR and a virtual PTBR for each VM
    - When guest OS is activated, the real PTBR points to a shadow page table
    - When guest OS attempts to modify the PTBR, it will be intercepted by VMM for further emulation



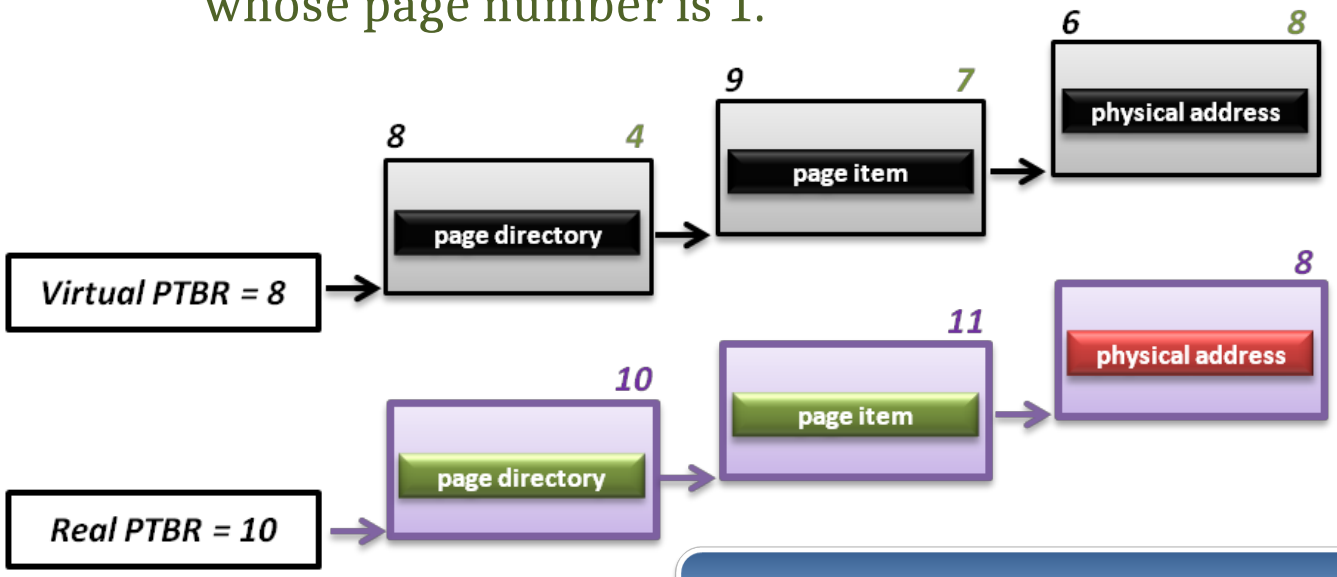
Shadow Page Table

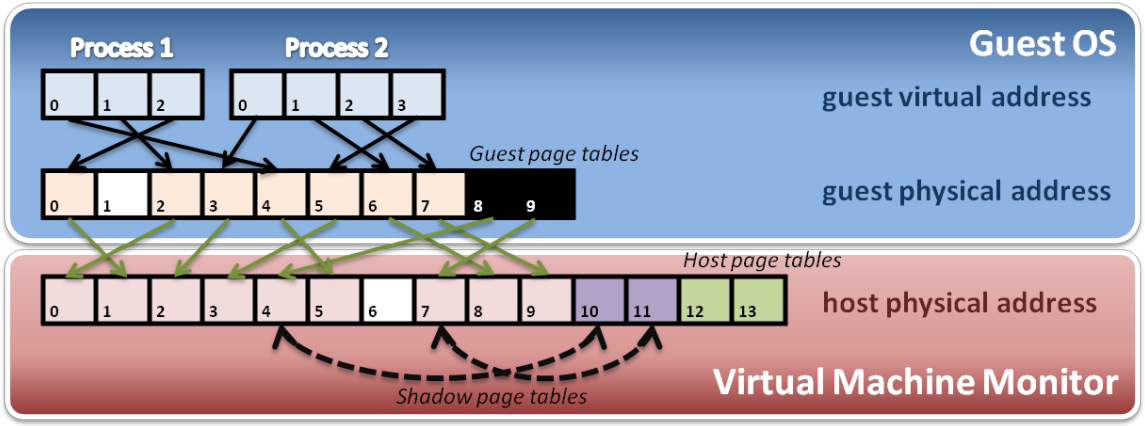
- Construct shadow page table
  - Guest OS will maintain its own page table for each process.
  - VMM maps each guest physical page to host physical page.
  - Create shadow page tables for each guest page table.
  - VMM should protect host frame which contains guest page table.



Shadow Page Table

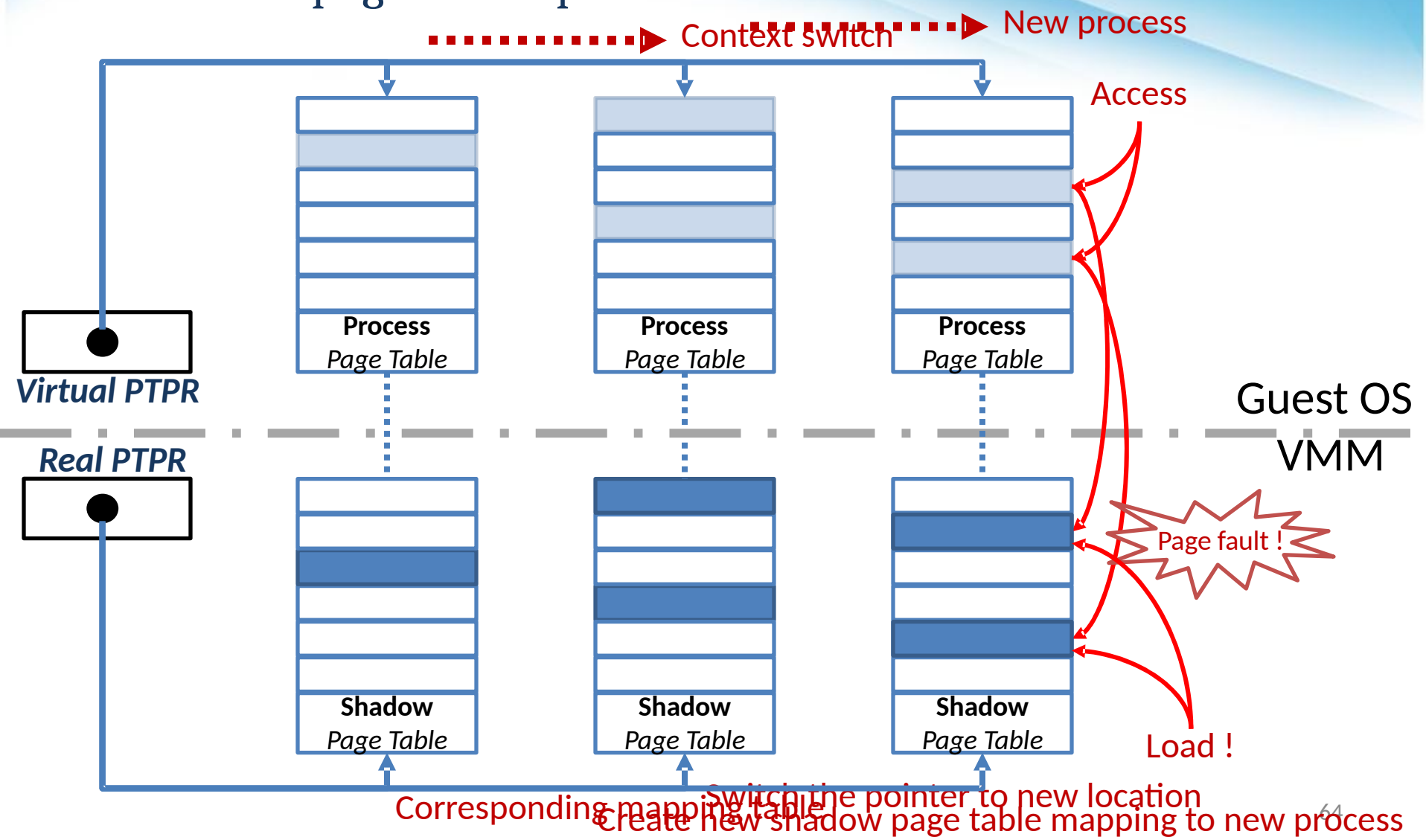
- Implement with PTBR :
  - For example, process 2 in guest OS want to access its memory whose page number is 1.





Shadow Page Table

- Shadow page table operations :



Other Issues

- Page fault and page protection issue
  - When a physical page fault occurs, VMM need to decide whether this exception should be injected to guest OS or not.
    - If the page entry in the page table of guest OS is still valid, VMM should prepare the corresponding page and not inject any exception to guest OS.
    - If the page entry in the page table of guest Os is invalid either, then VMM should directly inject the virtual page fault to guest OS.
  - When guest OS want to modify its page tables, VMM need to intercept this operation.
    - When guest OS reload PTBR, CPU will trap to VMM due to the Ring Compression nature.
    - VMM will walk the page table of guest OS and modify the related shadow page table to make MMU get host physical address.

Shadow page table

Hardware assistance

MEMORY VIRTUALIZATION

Hardware Solution

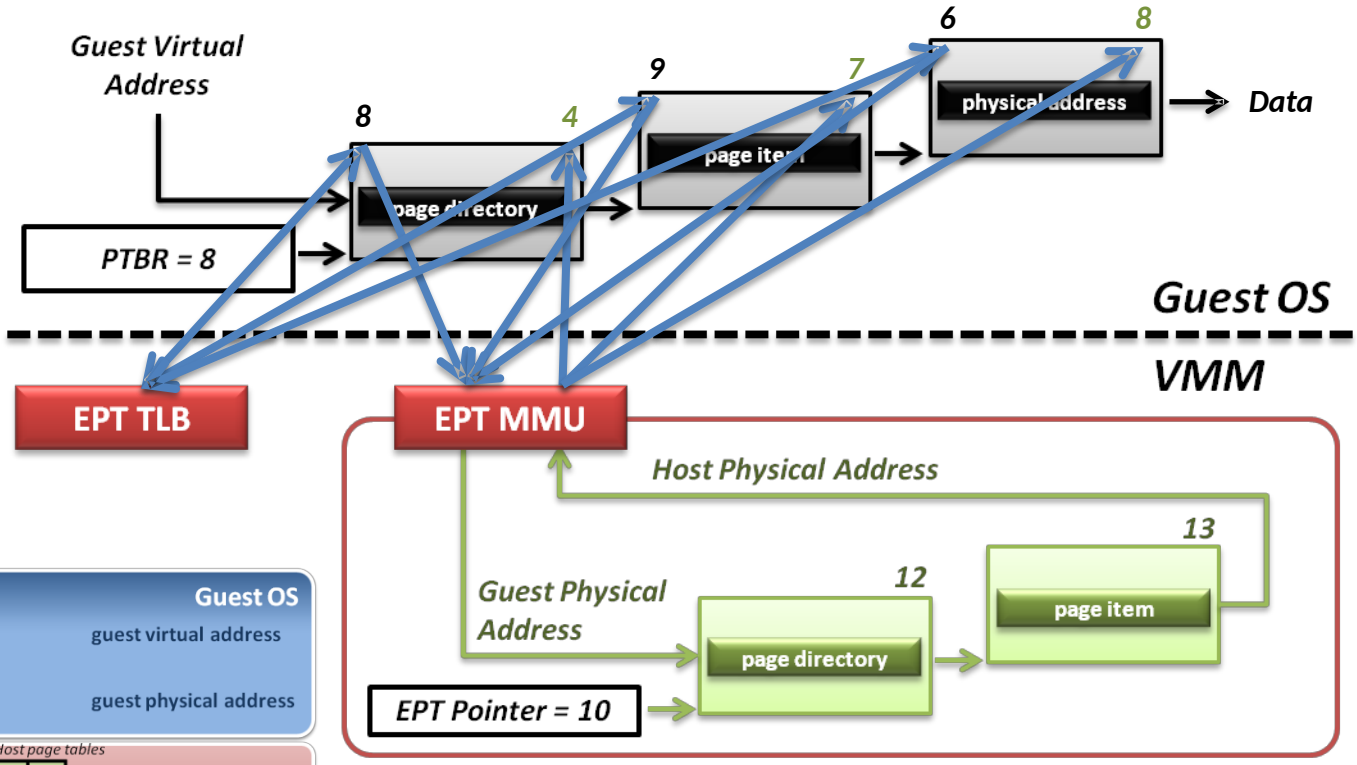
- Difficulties of shadow page table technique :
  - Shadow page table implementation is extremely complex.
  - Page fault mechanism and synchronization issues are critical.
  - Host memory space overhead is considerable.
- But why we need this technique to virtualize MMU ?
  - MMU do not first implemented for virtualization.
  - MMU is knowing nothing about two level page address translation.
- Now, let us consider hardware solution.

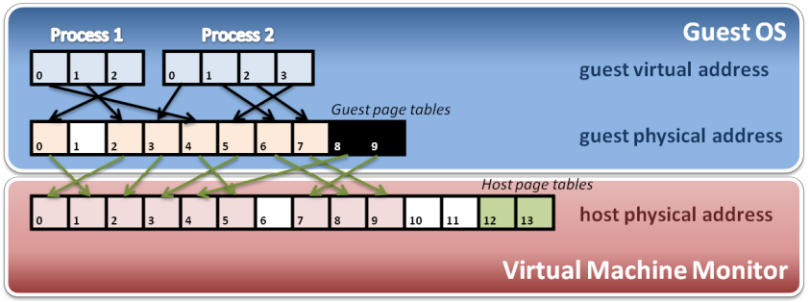
Extended Page Table

- Concept of Extended Page Table (EPT) :
  - Instead of walking along with only one page table hierarchy, EPT technique implement one more page table hierarchy.
    - One page table is maintained by guest OS, which is used to generate guest physical address.
    - The other page table is maintained by VMM, which is used to map guest physical address to host physical address.
  - For each memory access operation, EPT MMU will directly get guest physical address from guest page table, and then get host physical address by the VMM mapping table automatically.

Extended Page Table

- Memory operation :





Memory Virtualization Summary

- Software implementation
  - Memory architecture
    - MMU (memory management unit)
    - TLB (translation lookaside buffer)
  - Shadow page table
    - MMU virtualization by virtual PTBR
    - Shadow page table construction
    - Page fault and page table protection
- Hardware assistance
  - Extended page table
    - Hardware walk guest and host page table simultaneously

Overview

Device Model

Hardware Assistance

IO VIRTUALIZATION

IO Virtualization

- Goal :
  - Share or create IO devices for virtual machines.
- Two types of IO subsystem architecture :
  - Port Mapped IO
    - Port-mapped IO uses a special class of CPU instructions specifically for performing IO.
  - Memory Mapped IO (MMIO)
    - Memory Mapped IO uses the same address bus to address both memory and IO devices, and the CPU instructions used to access the memory are also used for accessing devices.
- Traditional IO techniques :
  - Direct memory Access (DMA)
  - PCI / PCI Express

Port Mapped IO

- IO devices are mapped into a separate address space
  - IO devices have a separate address space from general memory, either accomplished by an extra “IO" pin on the CPU's physical interface, or an entire bus dedicated to IO.
  - Generally found on Intel microprocessors, specifically the `IN` and `OUT` instructions which can read and write one to four bytes (`outb`, `outw`, `outl`) to an IO device.
- Pros & Cons
  - Pros
    - Less logic is needed to decode a discrete address.
    - Benefits for CPUs with limited addressing capability.
  - Cons
    - More instructions are required to accomplish the same task.
    - IO addressing space size is not flexible.

Memory Mapped IO

- IO devices are mapped into the system memory map along with RAM and ROM.
  - To access a hardware device, simply read or write to those 'special' addresses using the normal memory access instructions.

- Pros & Cons
  - Pros
    - Instructions which can access memory can be used to operate an IO device.
    - Operate on the data with fewer instructions.
  - Cons
    - Physical memory addressing space must be shared with IO devices.
    - The entire address bus must be fully decoded for every device.

Direct Memory Access

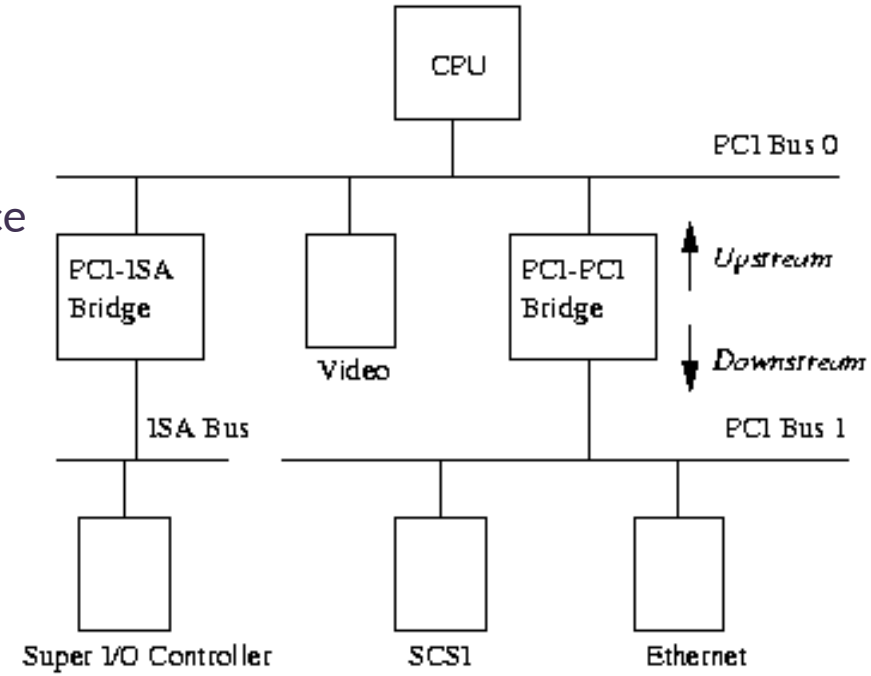
- What is DMA ?
  - Allow certain hardware subsystems within the computer to access system memory for reading and/or writing independently of the central processing unit.
- Two types of DMA :
  - Synchronous DMA
    - The DMA operation is caused by software.
    - For example, sound card driver may trigger DMA operation to play music.
  - Asynchronous DMA
    - The DMA operation is caused by devices (hardware).
    - For example, network card use DMA operation to load data into memory and interrupt CPU for further manipulation.

PCI & PCI Express

- What is PCI ?
  - PCI (Peripheral Component Interconnect) is a computer bus for attaching hardware devices.
  - Typical PCI cards used include :
    - Network cards, sound cards, modems
    - Extra ports such as USB or serial, TV tuner cards and disk controllers.
- What is PCI Express ?
  - PCIe is a computer expansion card standard designed to replace the older PCI, PCI-X, and AGP standards.
  - Its topology is based on point-to-point serial links, rather than a shared parallel bus architecture.

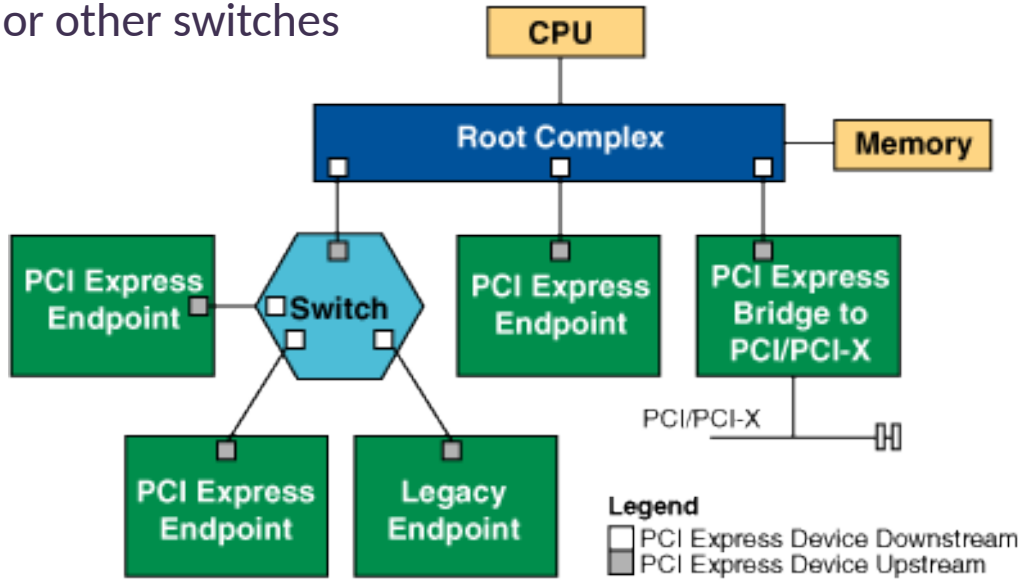
PCI & PCI Express

- PCI based system build in a tree topology
  - PCI bus
    - Parallel connect devices and bridges
  - PCI-PCI Bridge
    - Connect two PCI buses
    - Become the root of lower bus
  - PCI-ISA Bridge
    - Connect to conventional ISA device



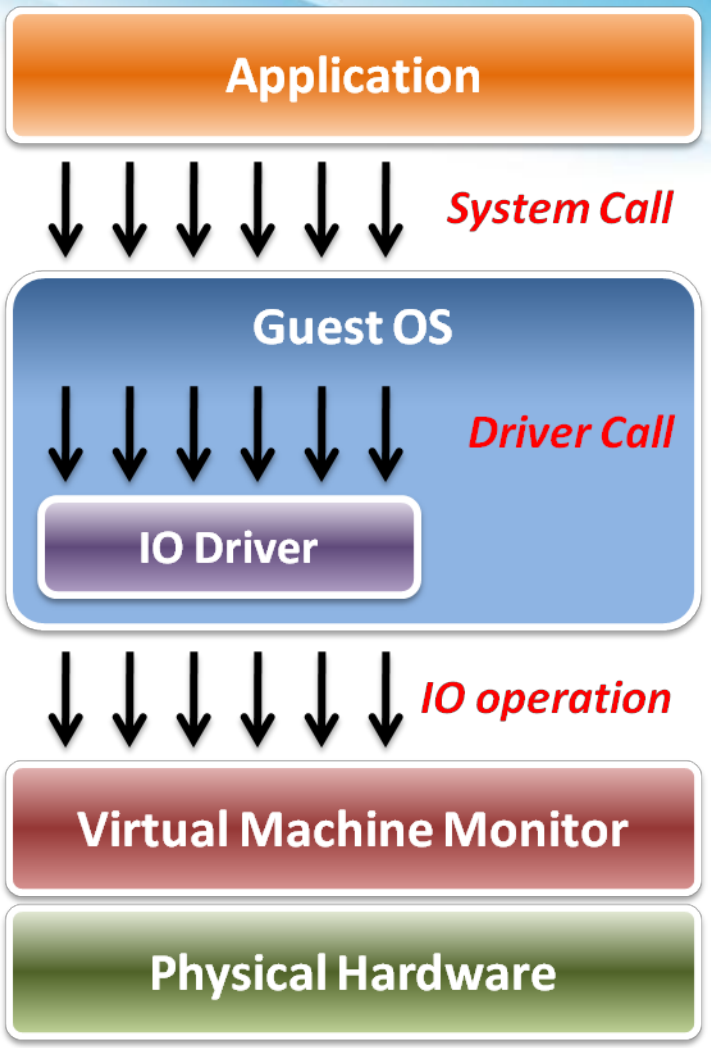
PCI & PCI Express

- PCIe based system build in a point to point architecture
  - Root Complex
    - Similar to a host bridge in a PCI system, the root complex generates transaction requests on behalf of the processor, which is interconnected through a local bus.
  - Switch
    - Connect endpoint devices or other switches
  - Endpoint Device
    - Physical PCIe devices
    - Legacy PCI devices
  - PCI Express Bridge
    - Connect to other legacy subsystems



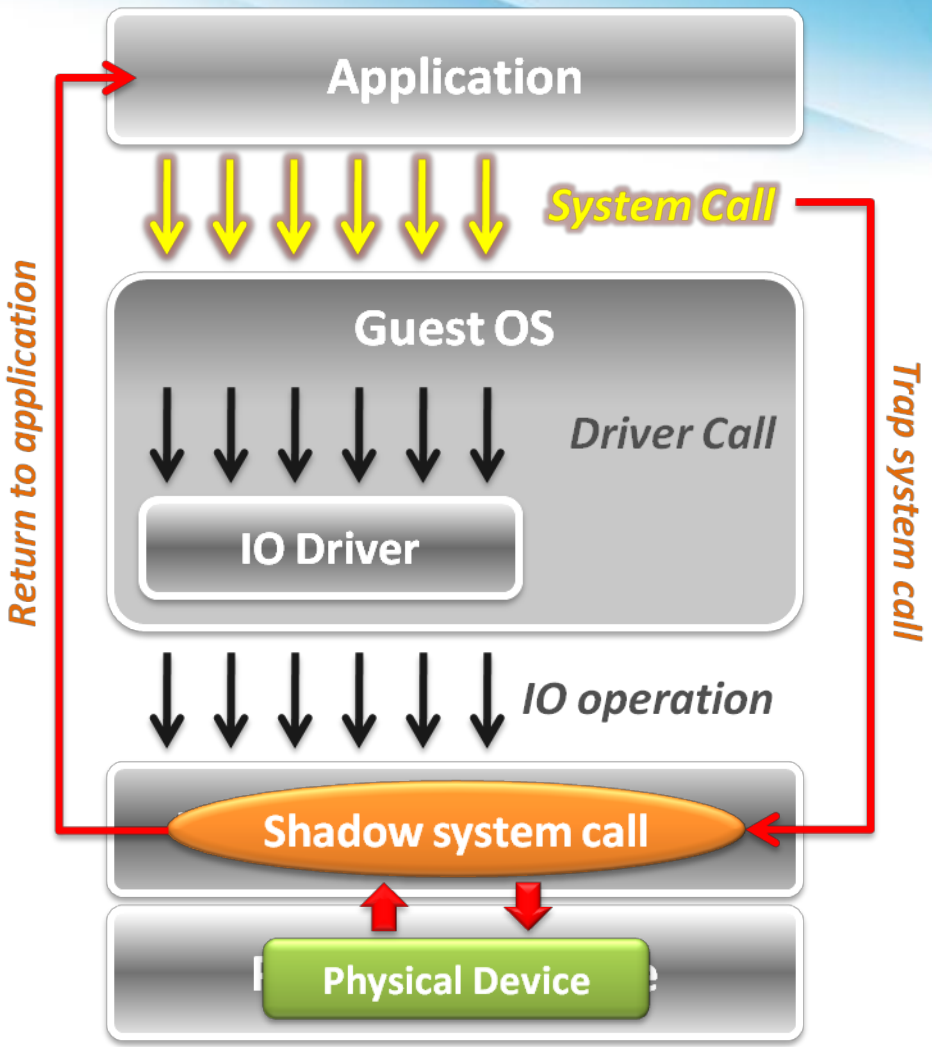
IO Virtualization

- Implementation Layers :
  - System call
    - The interface between applications and guest OS.
  - Driver call
    - The interface between guest OS and IO device drivers.
  - IO operation
    - The interface between IO device driver of guest OS and virtualized hardware (in VMM).



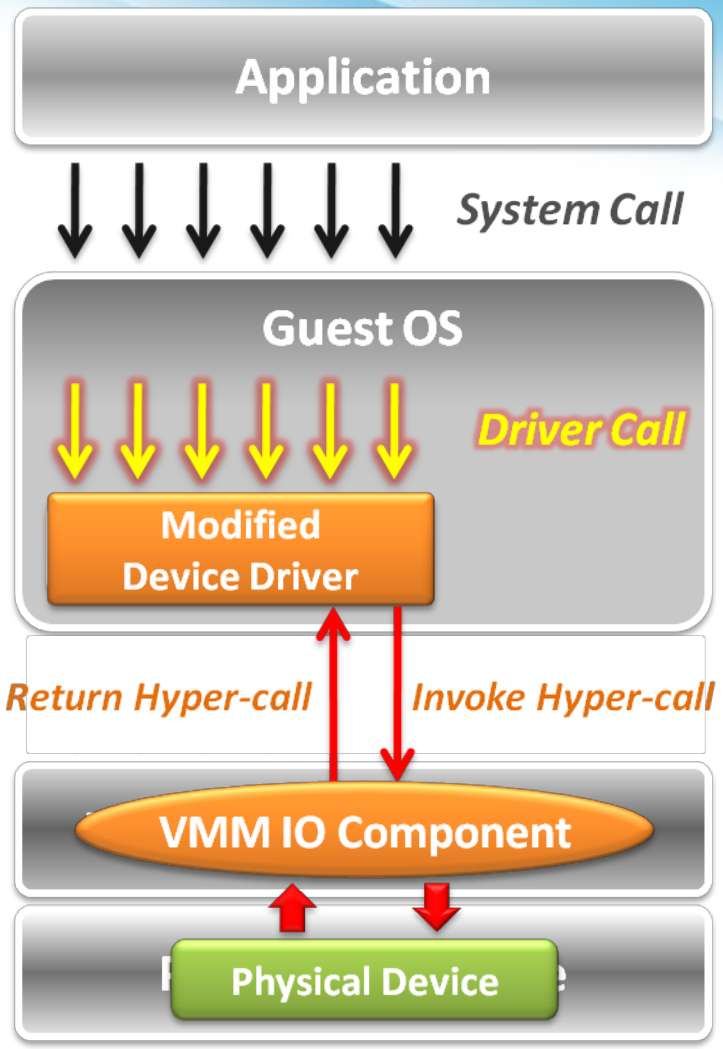
IO Virtualization

- In **system call** level :
  - When application invoke a system call, system will trap to VMM first.
  - VMM intercepts system calls, and maintains shadowed IO system call routines to simulate functionalities.
  - After simulation, VMM directly return to application in gust OS.



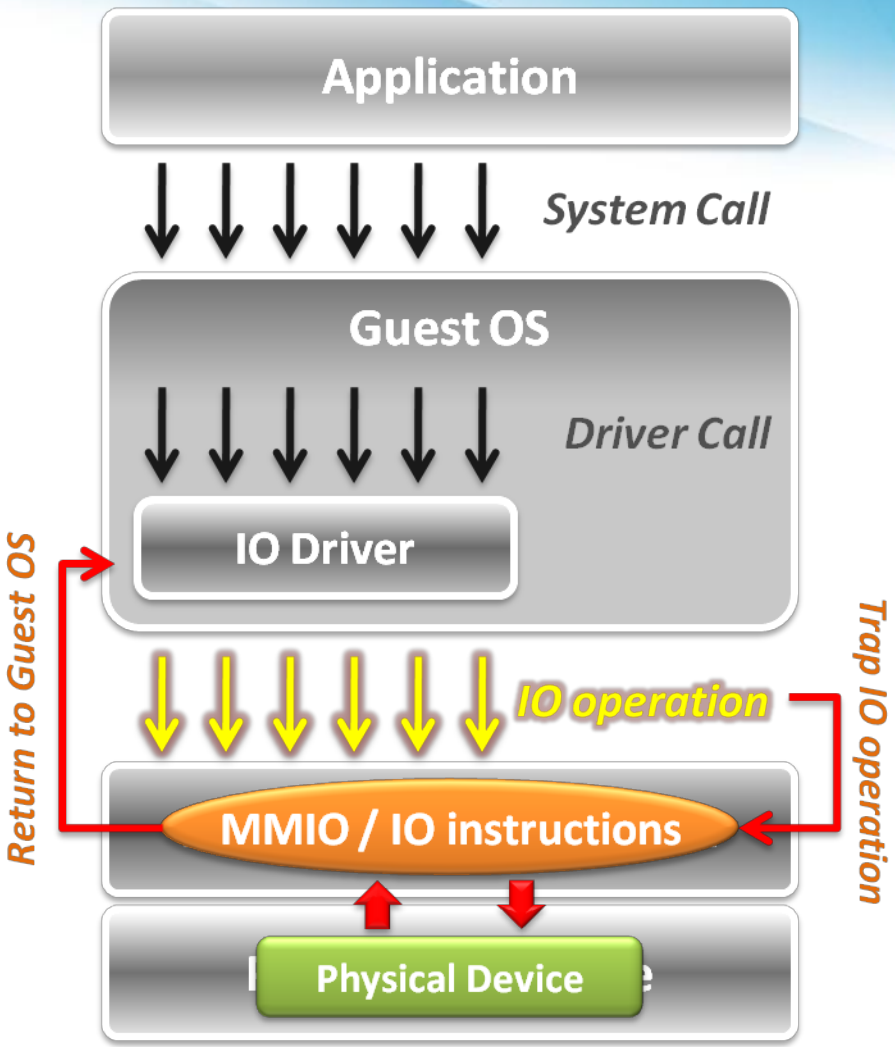
IO Virtualization

- In **device driver call** level :
  - Utilize para-virtualization technique, which means the IO device driver in guest OS should be modified.
  - The IO operation is invoked by means of hyper-call between the modified device driver and VMM IO component.



IO Virtualization

- In **IO operation** level, two approaches :
  - Memory mapped IO
    - Loads/stores to specific region of real memory are interpreted as command to devices.
    - The memory mapped IO region is protected.
  - Port mapped IO
    - Special input/output instructions with special addresses.
    - The IO instructions are privileged .
- Due to the privileged nature, these IO operations will be trapped to the VMM.



Overview

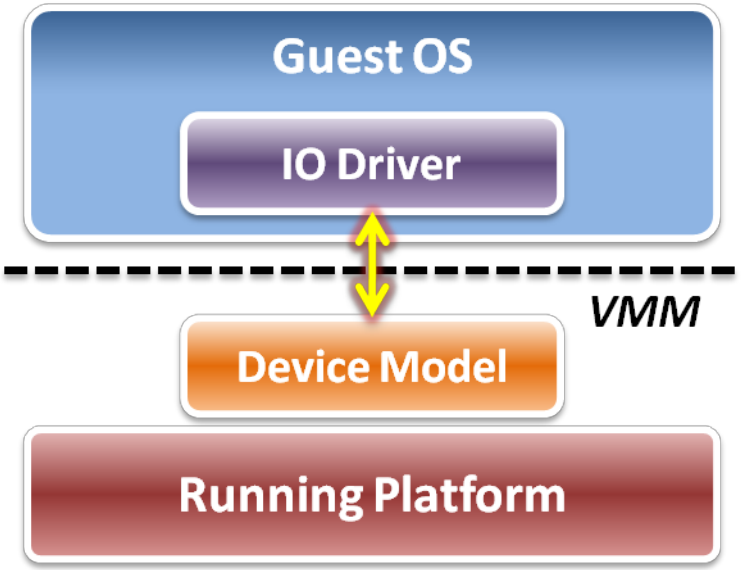
Device Model

Hardware Assistance

IO VIRTUALIZATION

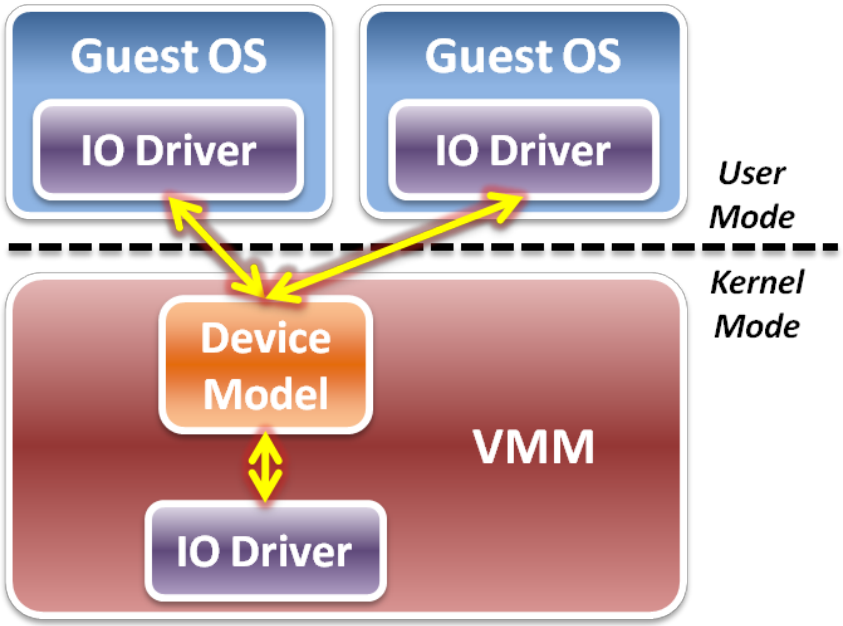
Device Model

- We focus on IO operation level implementation.
  - This is an approach of full virtualization.
- Logic relation between guest OS and VMM :
  - VMM intercepts IO operations from guest OS.
  - Pass these operations to device model on a running platform.
  - Device model need to emulate the IO operation interfaces.
    - Port mapped IO
    - Memory mapped IO
    - DMA
    - … etc.

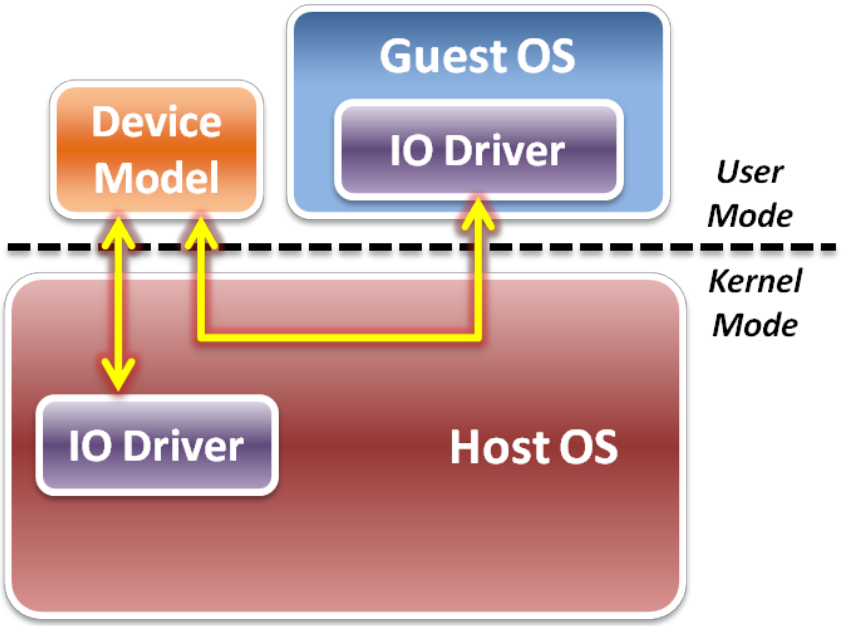


Device Model

- Two different implementations of device model :
  1. Device model is implemented as a part of VMM.
  2. Device model is running in user space as a stand alone service.



Type 1 Virtualization



Type 2 Virtualization

Device Model

- IO virtualization paradigm
  - Initialization – device discovery
    - VMM will make guest OS discover the virtualized IO devices.
    - Then guest OS will load the corresponding device driver.
  - Operation – access interception
    - When guest OS applies IO operations, VMM will intercept those accesses.
    - After virtual device operations, VMM returns the control to guest OS.
  - Virtualization – device virtualization
    - Device model should emulate the real electronic logic to satisfy all device interface definition and its effects.
    - VMM may share physical devices to all virtual machines.

Device Discovery

- Virtualize physical bus devices
  - Non-enumerable physical device
    - These devices have their own hard-coded numbers.
    - VMM should setup some status information on the virtual device ports.
    - For example, PS/2 keyboard and mouse.
  - Enumerable physical device
    - These devices defined a complete device discover method.
    - VMM have to emulate not only the device itself, but the bus behavior.
    - For example, PCI or PCI express devices.
- Virtualize non-exist devices
  - VMM must define and emulate all functions of these devices
    - VMM may define them as either non-enumerable or enumerable devices.
    - Guest OS needs to load some new drivers of these virtual devices.

Access Interception

- After virtual device discovered by guest OS, VMM has to intercept and control all the IO operations from guest OS.

- Port mapped IO operation
  - Direct device assignment
    - VMM should turn ***ON*** the physical IO bitmap.
    - All the IO instructions (**IN/OUT**) from guest OS will be directly performed onto hardware without VMM intervention.
  - Indirect device assignment
    - VMM should turn ***OFF*** the physical IO bitmap.
    - All the IO instructions from guest OS will be intercepted by VMM and forward to physical hardware.

Access Interception

- Memory mapped IO operation
  - Direct device assignment
    - VMM should use the shadow page table to map IO device addressing space of guest OS to the space of host.
    - Then all the IO operations from guest OS will not be intercepted.
  - Indirect device assignment
    - VMM should make the all entries of the IO device addressing space in the shadow page table to be invalid.
    - When guest OS access those addressing space, it will introduce the page fault which trap CPU to VMM for device emulation.
- DMA mechanism
  - Address remapping
    - Because the device driver in the guest OS have nothing to know with the host physical address, VMM need to automatic remap the DMA target when intercepting guest OS.

Device Virtualization

- IO device types :
  - Dedicated device
    - Ex : displayer, mouse, keyboard …etc.
  - Partitioned device
    - Ex : disk, tape …etc
  - Shared device
    - Ex : network card, graphic card …etc.
  - Nonexistent physical device
    - Ex : virtual device …etc.

Device Virtualization

- Dedicated device
  - Do not necessarily have to be virtualized.
  - In theory, requests of such device could bypass the VMM.
  - However, they are handled by the VMM first since OS is running in user mode.
- Partitioned device
  - Be partitioned into several smaller virtual devices as dedicated to VM.
  - VMM translates address spaces to those of the physical devices.

Device Virtualization

- Shared device
  - Should be shared among VMs.
  - Each VM has its own virtual device state.
  - VMM translates requests from a VM to physical device .
- Nonexistent physical device
  - Virtual device “attached” to a VM for which there is no corresponding physical device.
  - VMM intercepts requests from a VM, buffers it and interrupts other VMs.

Performance Issues

- When considering performance, two major problems :
  - How to make guest OS directly access IO addresses ?
    - Other than software approaches discussed above, we can make use of the hardware assistance (Intel EPT technique in memory virtualization) to map IO addresses from host to guest directly without software overhead.
  - How to make DMA directly access memory space in guest OS ?
    - For the synchronous DMA operation, guest OS will be able to assign the correct host physical memory address by EPT technique.
    - For the asynchronous DMA operation, hardware must access memory from host OS which will introduce the VMM intervention.

Overview

Device Model

Hardware Assistance

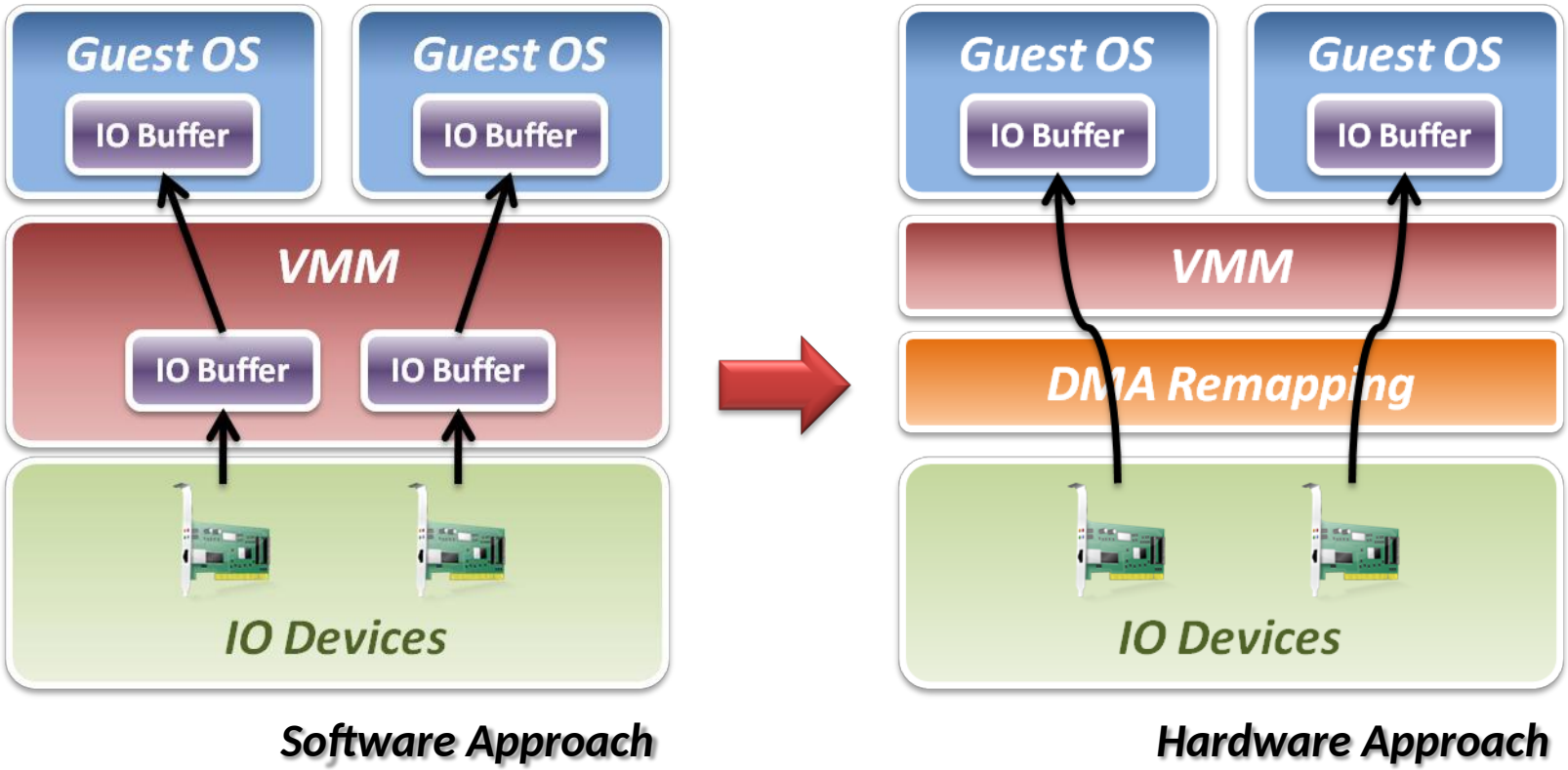
IO VIRTUALIZATION

Hardware Solution

- Difficulty :
  - Software cannot make data access directly from devices.
- Tow hardware solutions :
  - Implement DMA remapping in hardware
    - Remap DMA operations automatically by hardware.
    - For example, ***Intel VT-d*** .
  - Specify IO virtualization standards of PCI Express devices
    - Implement virtualizable device with PCI Express interface.
    - For example, ***SR-IOV*** or ***MR-IOV***.

Intel VT-d

- Add DMA remapping hardware component.



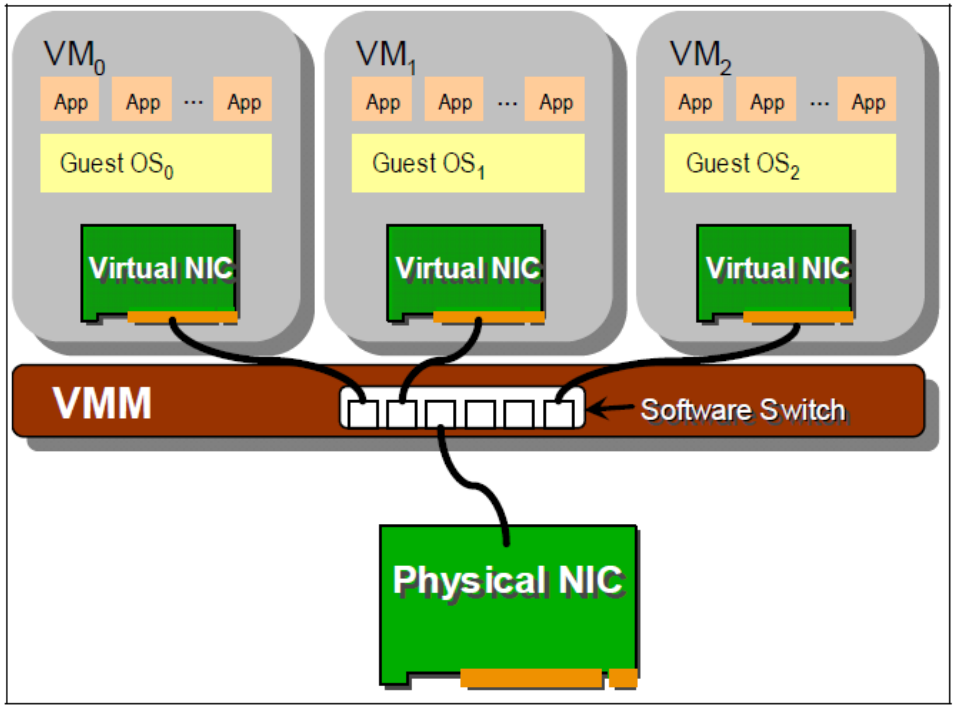
Software Approach

Hardware Approach

IO Virtualization Brief Review

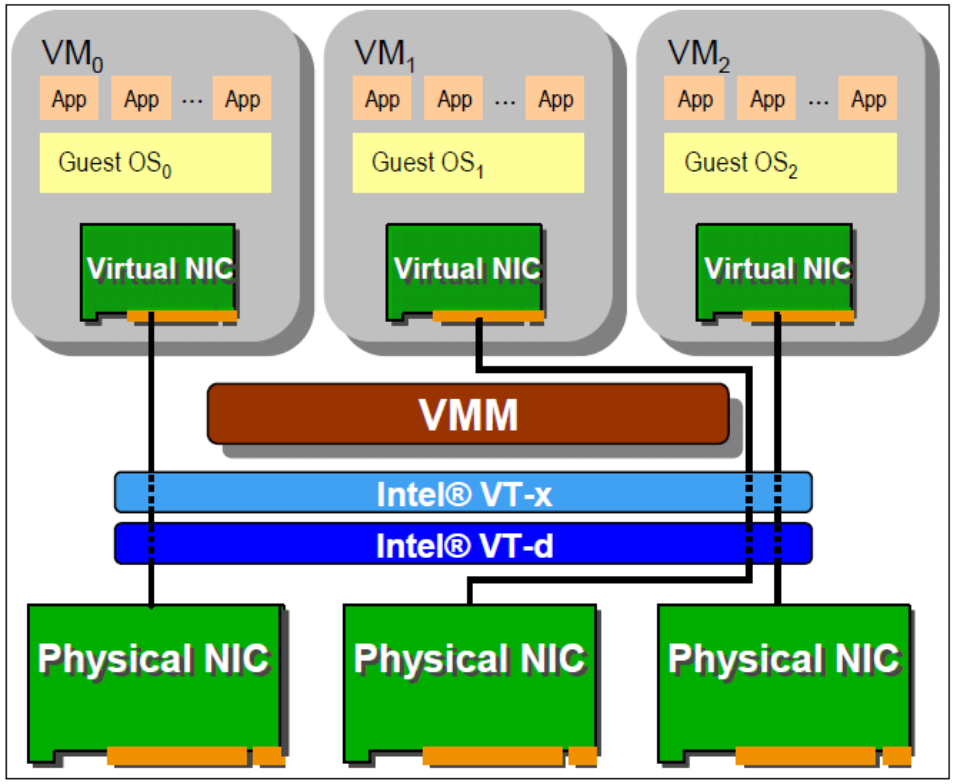
- Let's give a brief IO virtualization review.

- Software based sharing
  - Implement virtualization by VMM software stack.
  - Advantage
    - Full virtualization without special hardware support.
  - Disadvantage
    - Significant CPU overhead may be required by the VMM.



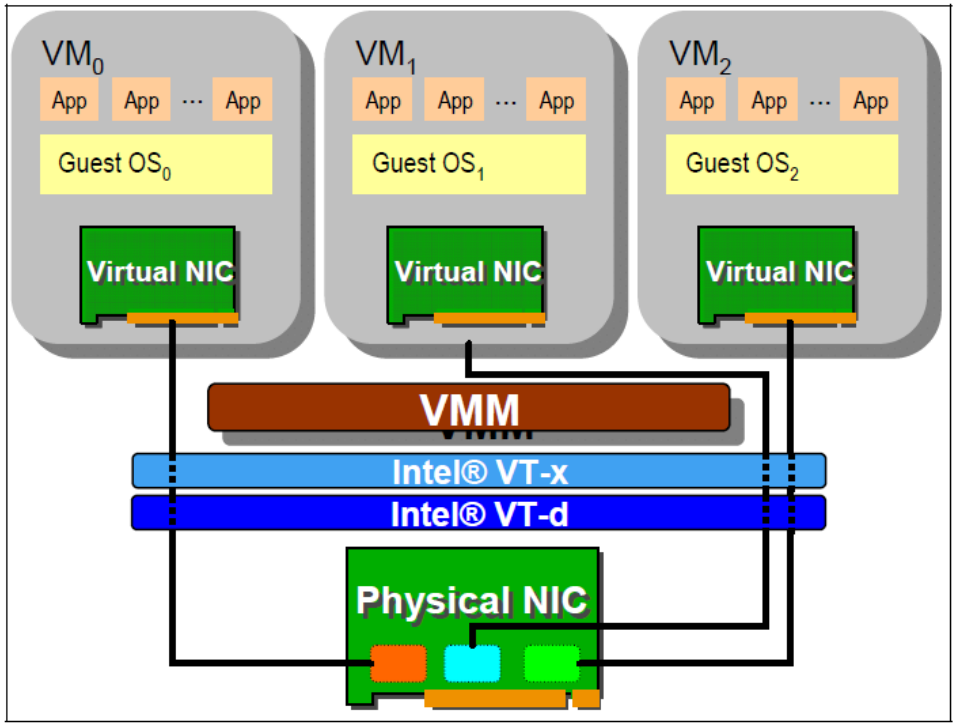
IO Virtualization Brief Review

- Hardware direct assignment
  - Implement virtualization with Intel VT-x and VT-d supports.
  - Advantage
    - Data access bypass VMM.
    - Improve IO performance.
  - Disadvantage
    - Dedicate physical device assignment limit the system scalability.



IO Virtualization Brief Review

- New industrial standard
  - Instead of implementing virtualization in CPU or memory only, industry com up with new IO virtualization standard in PCI Express devices.
  - Advantages
    - Fully collaboration with physical hardware devices.
    - Improve system scalability.
    - Improve system agility.
  - Disadvantages
    - IO devices must implement with new specification.



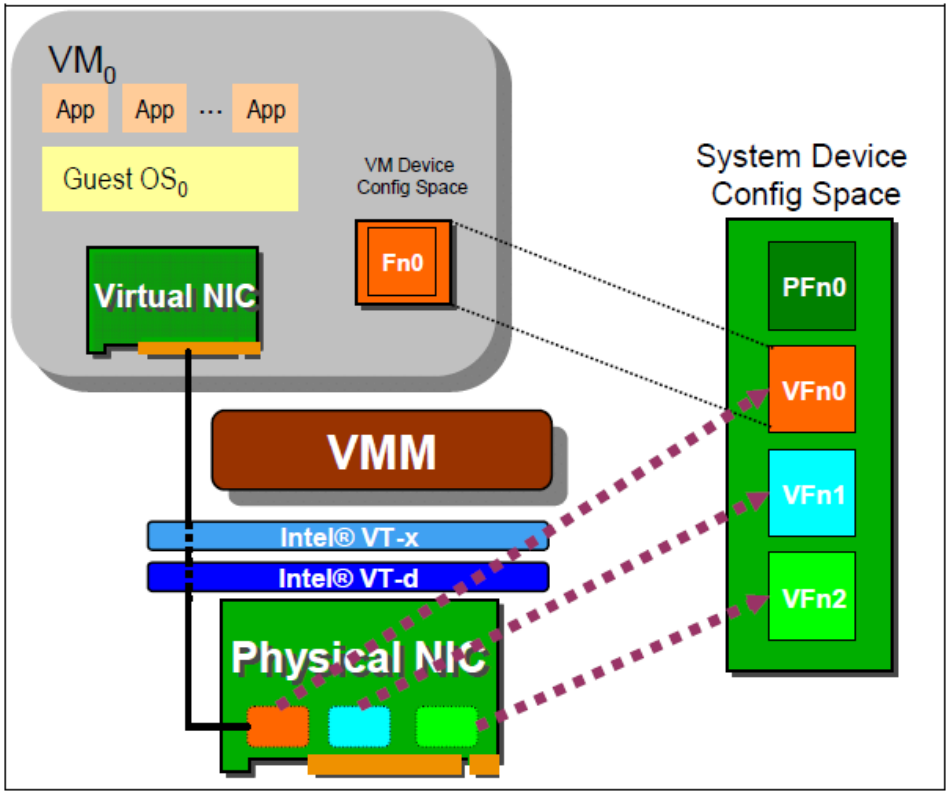
Single Root – IO Virtualization

- What is SR-IOV ?
  - The PCI-SIG Single Root I/O Virtualization and Sharing (SR-IOV) specification defines a standardized mechanism to create natively shared devices.

- Basic components :
  - Physical Functions (PFs):
    - These are full PCIe functions that include the SR-IOV Extended Capability.
    - The capability is used to configure and manage the SR-IOV functionality.
  - Virtual Functions (VFs):
    - These are “lightweight” PCIe functions that contain the resources necessary for data movement but have a carefully minimized set of configuration resources.

Single Root – IO Virtualization

- SR-IOV works with VMM :
  - VMM
    - An SR-IOV-capable device can be configured to appear in the PCI configuration space as multiple functions.
  - VM
    - The VMM assigns one or more VFs to a VM by mapping the actual configuration space the VFs to the configuration space presented to the virtual machine by the VMM.



IO Virtualization Summary

- IO subsystem architecture
  - Port Mapped IO vs. Memory Mapped IO
  - Direct Memory Access (DMA)
  - PCI / PCI Express
- IO virtualization
  - Three implementation layers
  - IO virtualization paradigm with device model
- Hardware assistance
  - DMA remapping hardware
  - Single Root – IO Virtualization specification

VMware , Xen , KVM

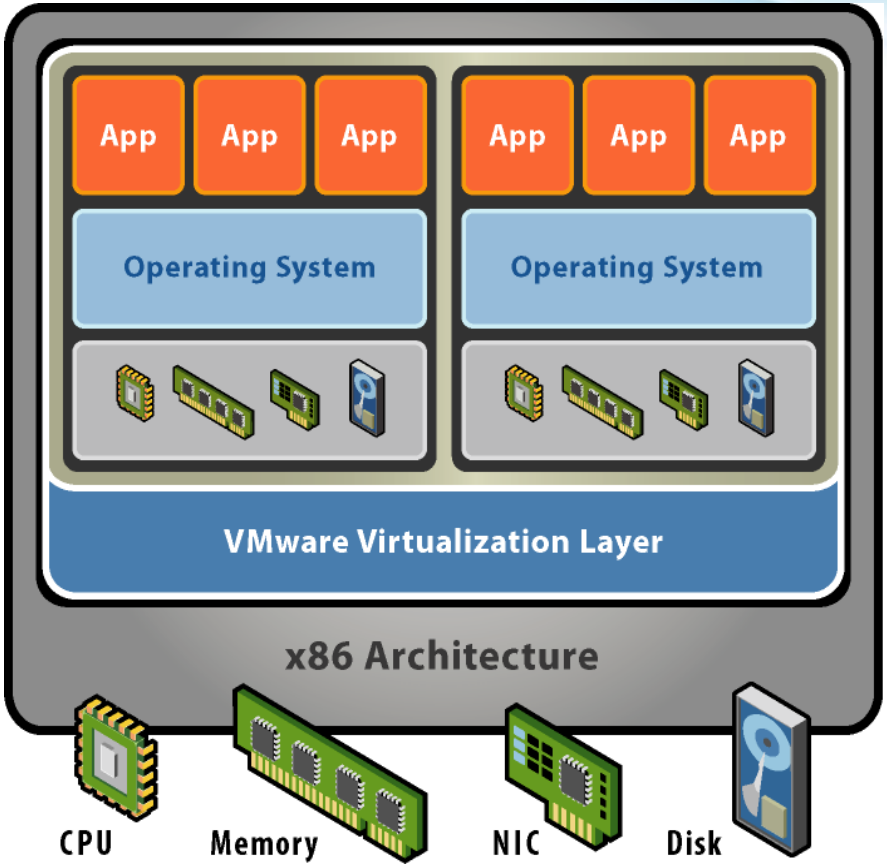
ECOSYSTEM

Venders and Projects

- Virtual machine venders :
  - VMware
    - The company was founded in 1998 and is based in Palo Alto, California. The company is majority owned by EMC Corporation.
    - Implement both type-1 and type-2 VM.
  - Xen
    - First developed in University of Cambridge Computer Laboratory.
    - As of 2010 the Xen community develops and maintains Xen as free software, licensed under the GNU General Public License (GPLv2).
    - Implement para-virtualization.
- Virtual machine project :
  - KVM (Kernel-based Virtual Machine)
    - A Linux kernel virtualization infrastructure.
    - As of 2010, KVM supports native virtualization using Intel VT-x or AMD-V.

VMware

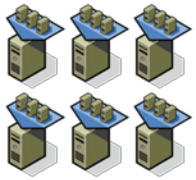
- Basic properties :
  - Separate OS and hardware – break hardware dependencies
  - OS and Application as single unit by encapsulation
  - Strong fault and security isolation
  - Standard, HW independent environments can be provisioned anywhere
  - Flexibility to chose the right OS for the right application



VMware Virtualization Stack

3 Management Automation

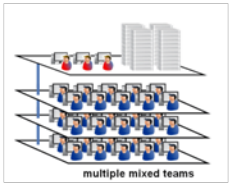
Infrastructure Optimization



Desktop Management

Business Continuity

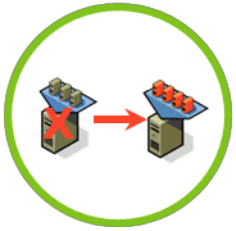
Software Lifecycle



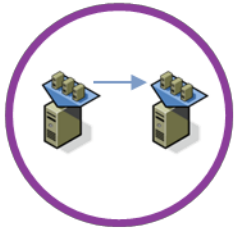
2 Distributed Virtualization

Resource Mgt

Availability

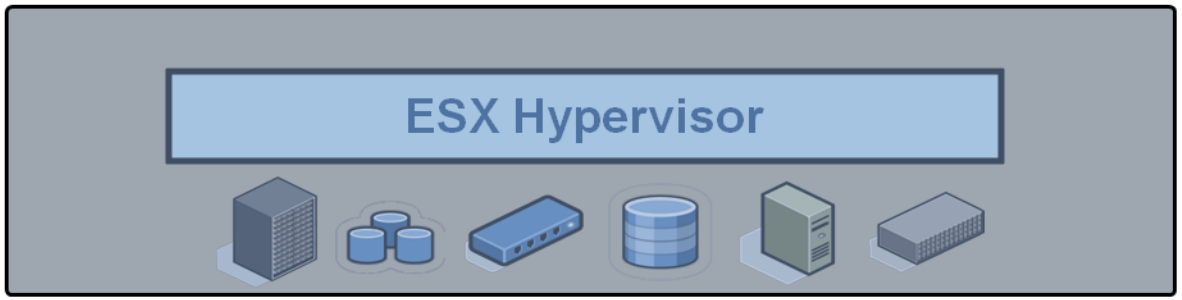


Mobility



Security

1 Virtualization Platforms



VMware Major Products

- VMware GSX Server
  - Run multiple servers on your server
  - Hosted Architecture
  - Available for Linux hosts and Windows hosts
- VMware ESX Server
  - Quality of Service
  - High-performance I/O
  - Host-less Architecture (bare-metal)

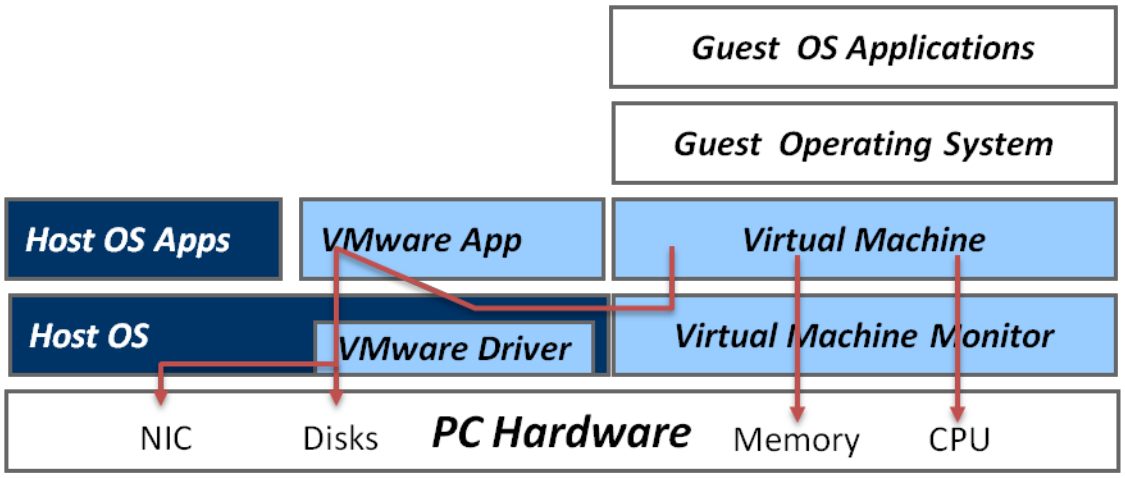
VMware GSX Server Architecture

Host Mode

VMware, acting as an application, uses the host to access other devices such as the hard disk, floppy, or network card

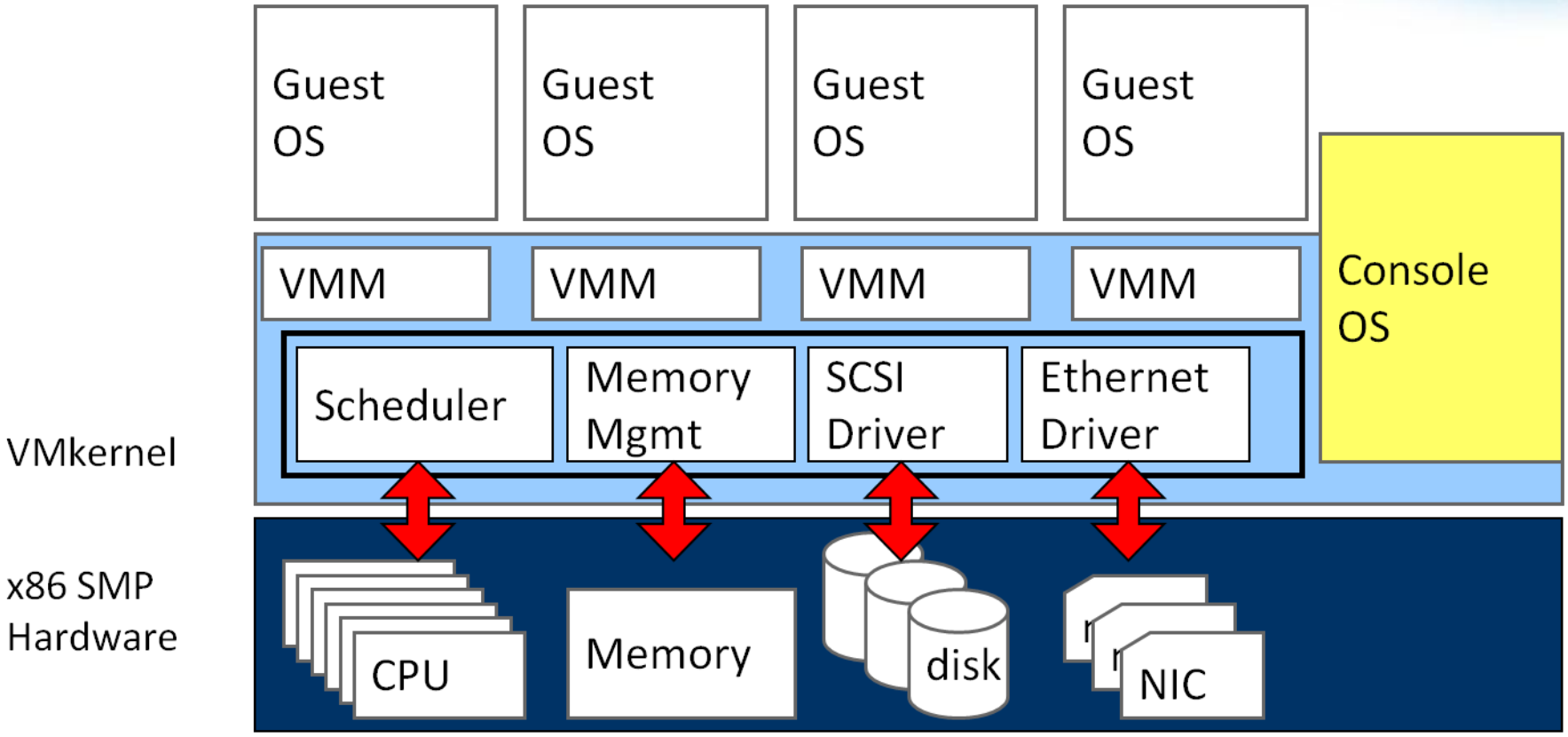
VMM Mode

The VMware Virtual machine monitor allows each guest OS to directly access the processor (direct execution)



VMware achieves both near-native execution speed and broad device support by transparently switching between Host Mode and VMM Mode.

VMware ESX Server Architecture



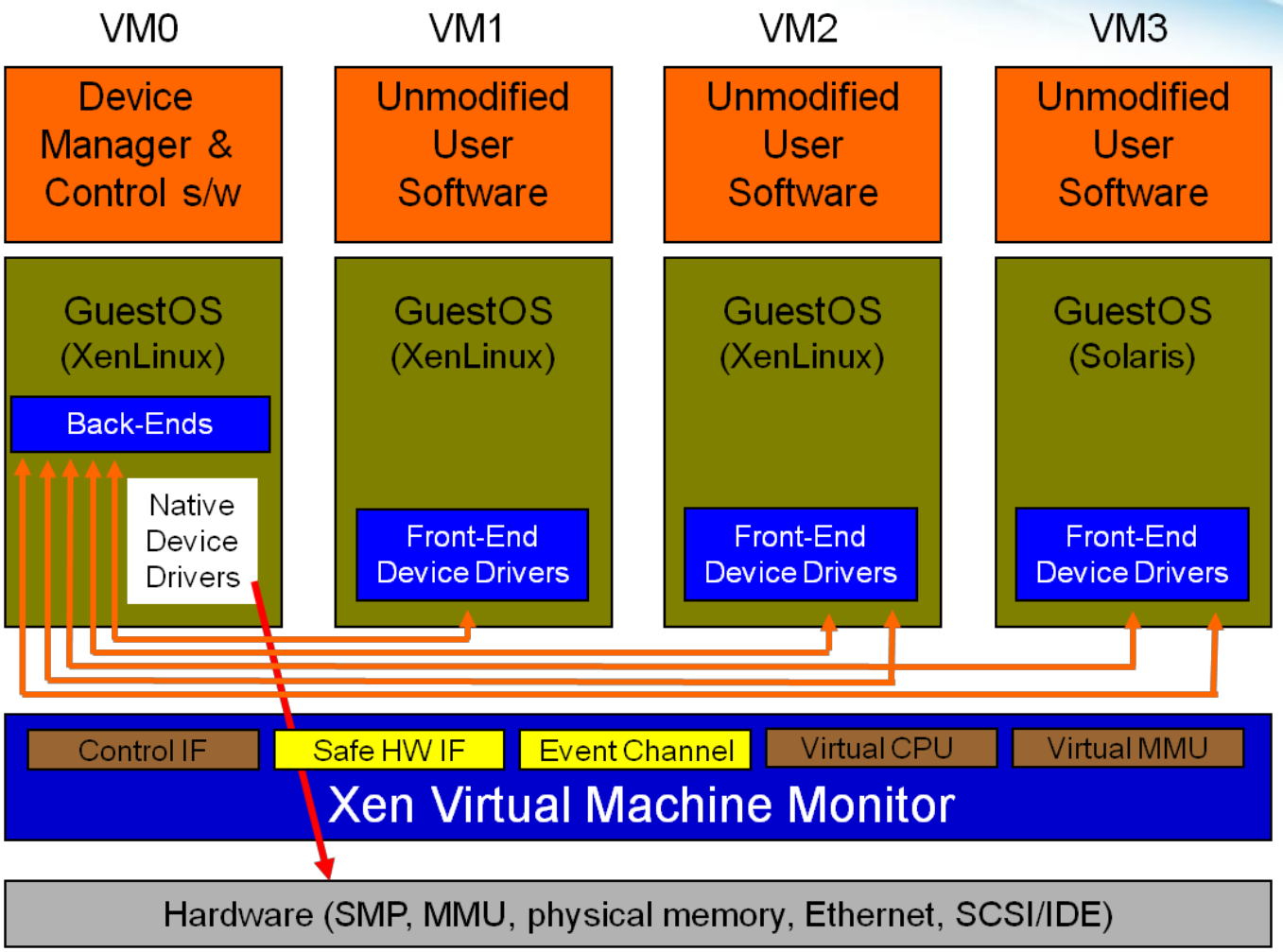
Xen

- Basic properties :
  - Para-virtualization
    - Achieve high performance even on its host architecture (x86) which has a reputation for non-cooperation with traditional virtualization techniques.
  - Hardware assisted virtualization
    - Both Intel and AMD have contributed modifications to Xen to support their respective Intel VT-x and AMD-V architecture extensions.
  - Live migration
    - The LAN iteratively copies the memory of the virtual machine to the destination without stopping its execution.
- Implement system:
  - Novell's SUSE Linux Enterprise 10
  - Red Hat's RHEL 5
  - Sun Microsystems' Solaris

Para-virtualization in Xen

- Xen extensions to x86 arch
  - Like x86, but Xen invoked for privileged instructions
  - Avoids binary rewriting
  - Minimize number of privilege transitions into Xen
  - Modifications relatively simple and self-contained
- Modify kernel to understand virtualized environment
  - Wall-clock time vs. virtual processor time
    - Desire both types of alarm timer
  - Expose real resource availability
    - Enables OS to optimize its own behaviour

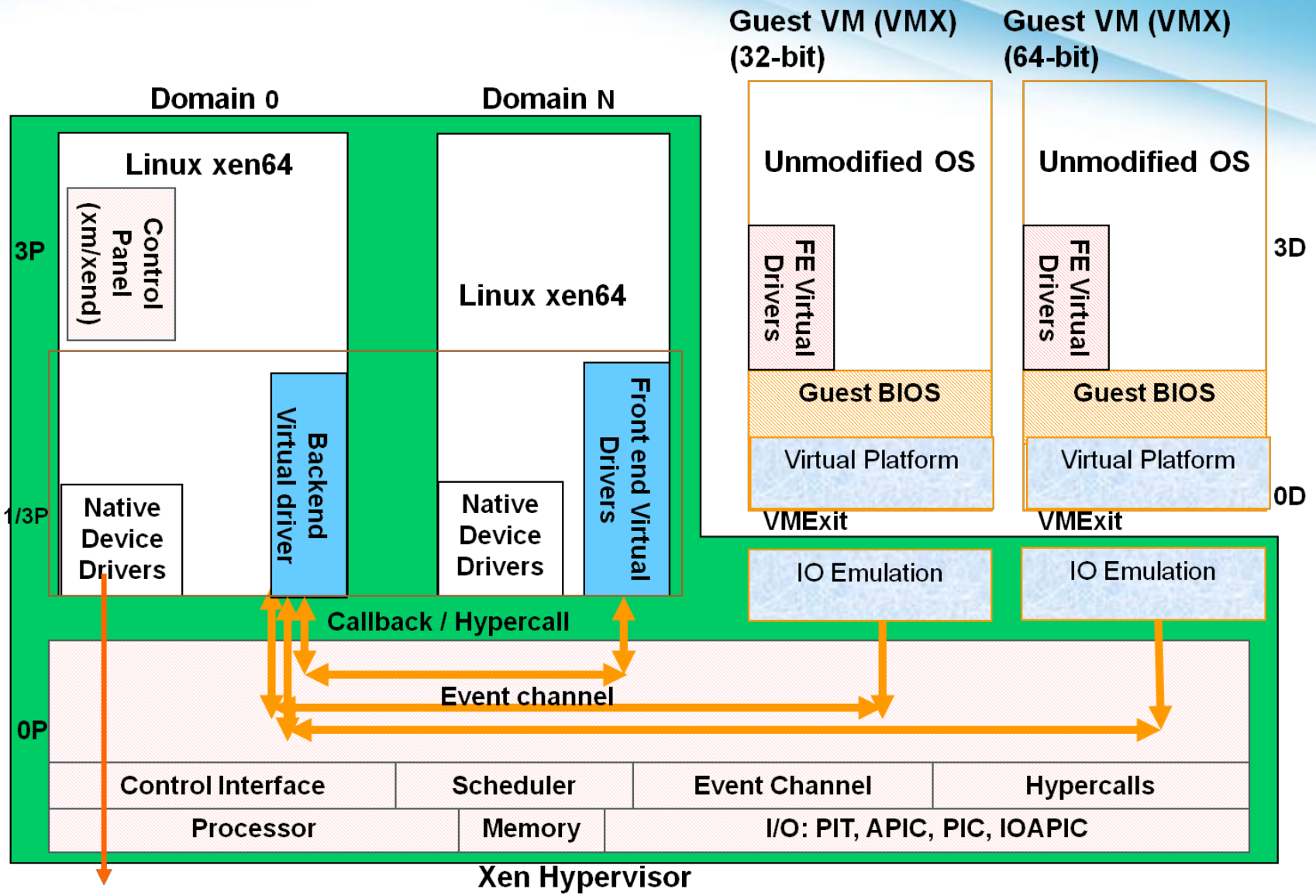
Original Xen Architecture



Hardware Assistance in Xen

- Hardware assistance :
  - CPU provides `VM Exit` for certain privileged instructions
  - Extend page tables used to virtualize memory
- Xen features :
  - Enable Guest OS to be run without modification
    - For example, legacy Linux and Windows
  - Provide simple platform emulation
    - BIOS, apic, iopaic, rtc, Net (pcnet32), IDE emulation
  - Install para-virtualized drivers after booting for high-performance IO
  - Possibility for CPU and memory para-virtualization
    - Non-invasive hypervisor hints from OS

New Xen Architecture

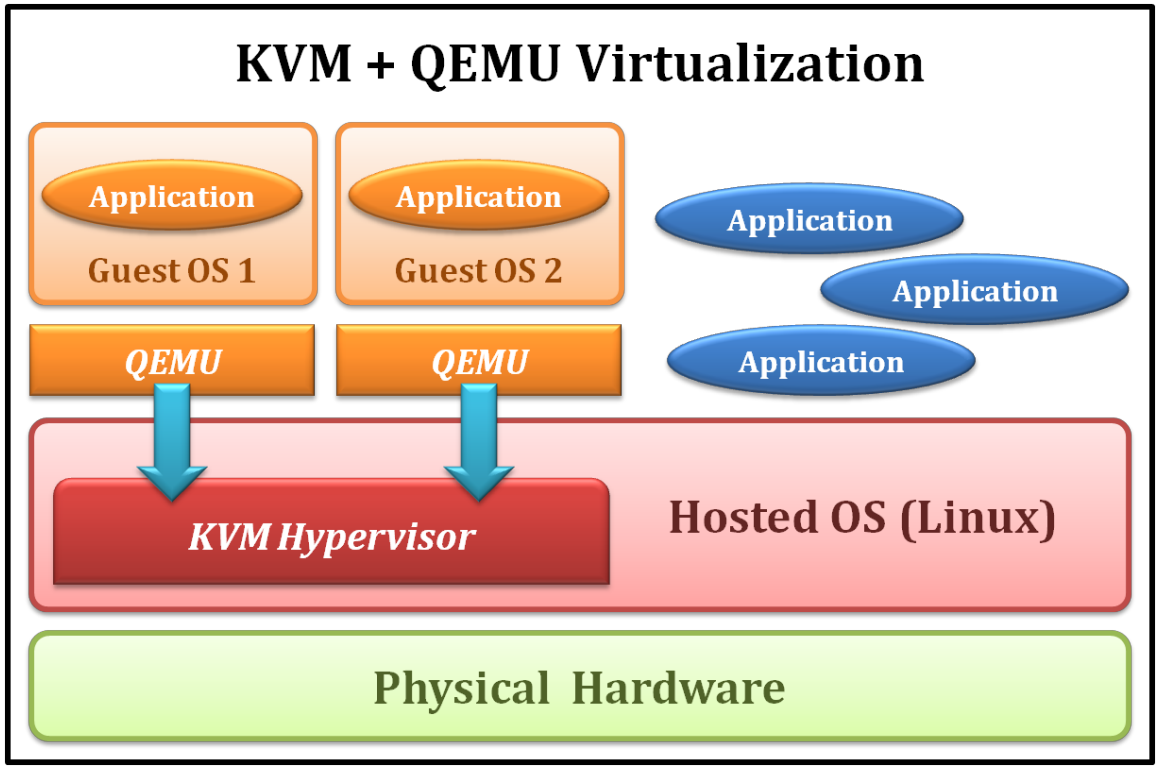


KVM

- KVM (Kernel-based Virtual Machine)
  - Linux host OS
    - The kernel component of KVM is included in mainline Linux, as of 2.6.20.
  - Full-virtualization
    - KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V).
    - Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images.
  - IO device model in KVM :
    - KVM requires a modified QEMU for IO virtualization framework.
    - Improve IO performance by **`virtio`** para-virtualization framework.

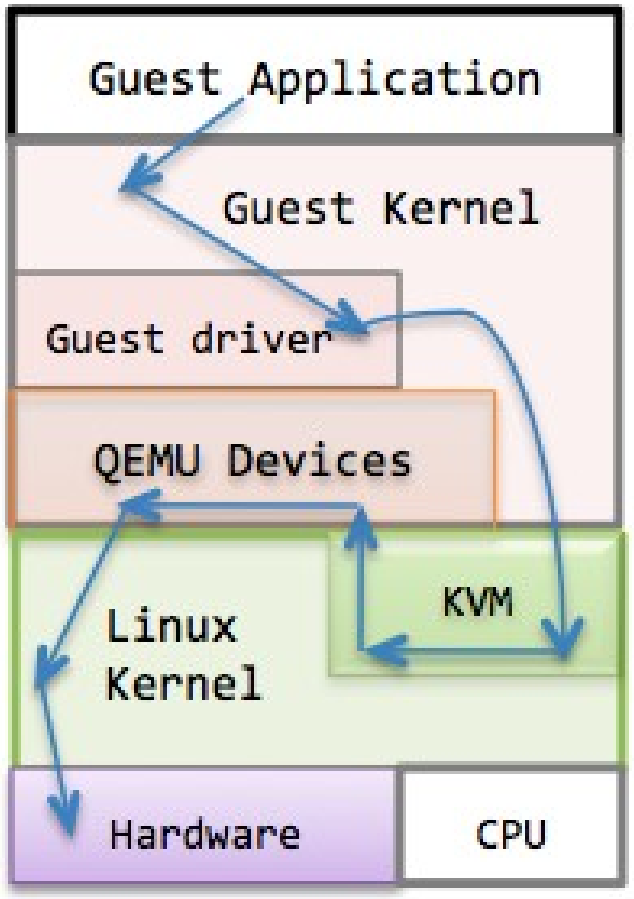
KVM Full Virtualization

- It consists of a loadable kernel module
  - `kvm.ko`
    - provides the core virtualization infrastructure
  - `kvm-intel.ko` / `kvm-amd.ko`
    - processor specific modules



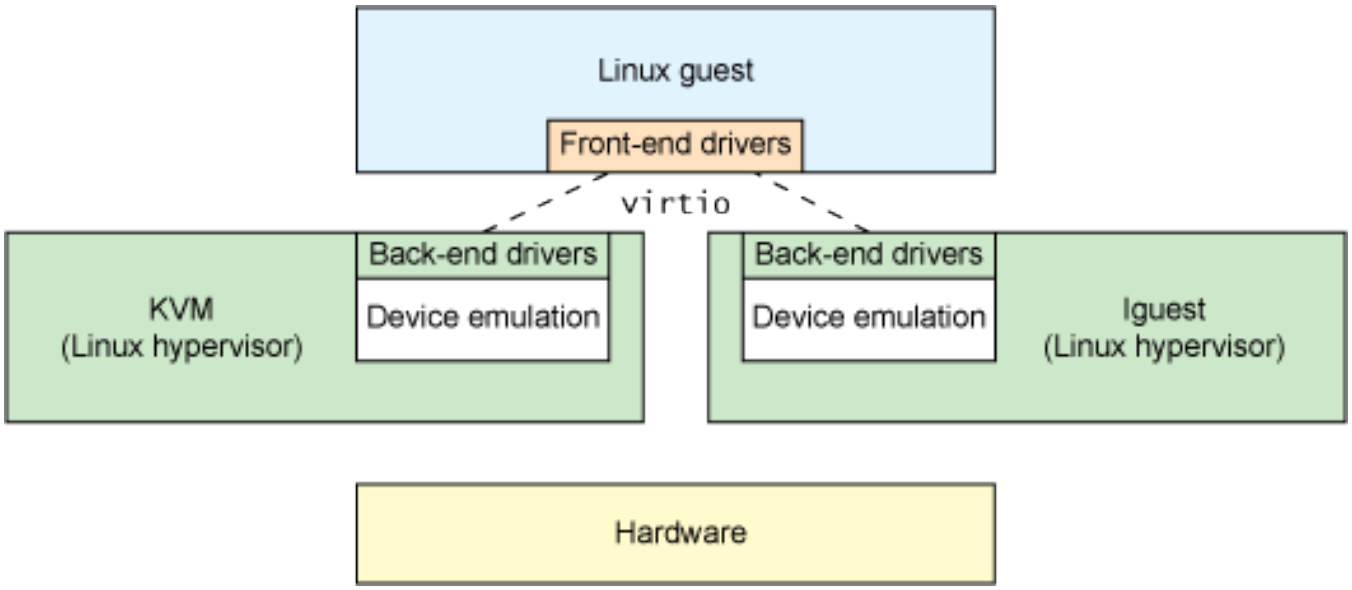
IO Device Model in KVM

- Original approach with full-virtualization
  - Guest hardware accesses are intercepted by KVM
  - QEMU emulates hardware behavior of common devices
    - RTL 8139
    - PIIX4 IDE
    - Cirrus Logic VGA



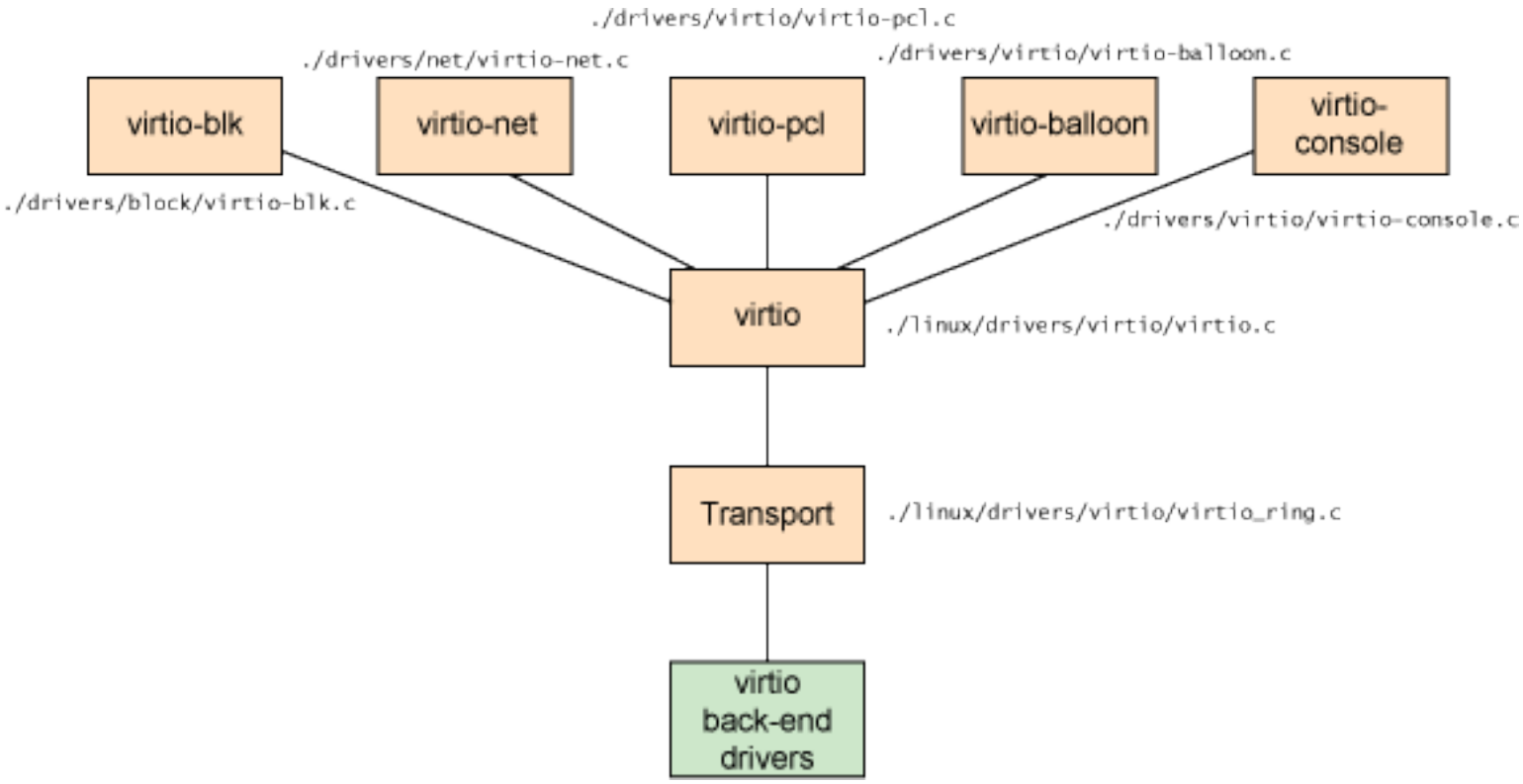
IO Device Model in KVM

- New approach with para-virtualization



IO Device Model in KVM

- **virtio** architecture



Live migration

Cloud properties

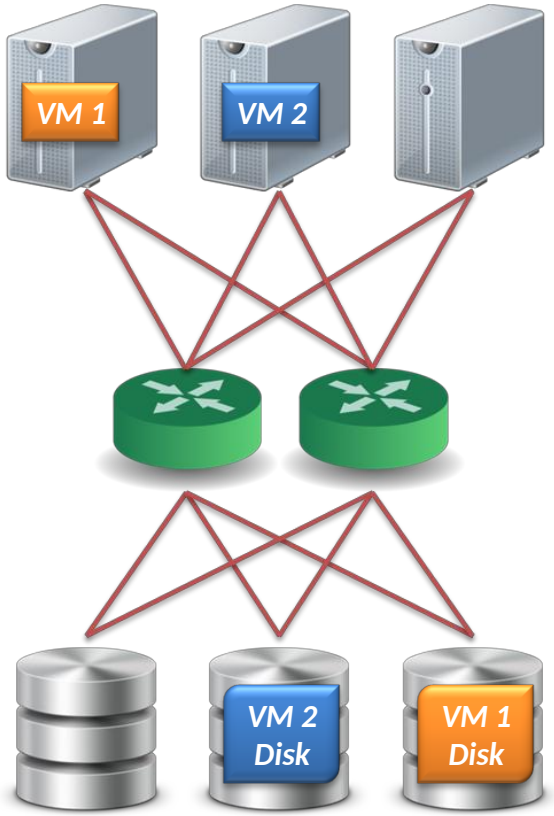
OTHER ISSUES

Other Issues

- Essential technique of cloud properties implementation
  - Live migration of virtual machines
    - Migrate a virtual machine from one physical machine to another in the run time with a small amount of performance down grade.
- Virtualization enabled cloud properties :
  - Scalability
    - Virtual machine system automatic scale up
  - Availability
    - Fault tolerant of hardware and software
  - Manageability
    - Automatic physical to virtual system transformation
  - Performance
    - Dynamically virtual machine level load balancing

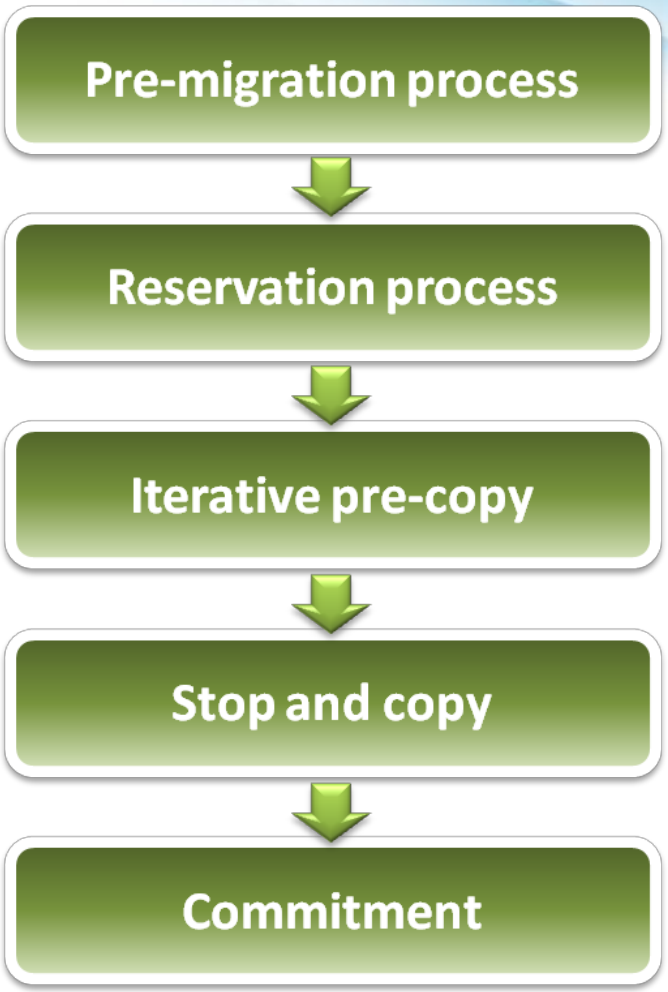
Live Migration Technique

- Pre-assumption :
  - We assume that all storage resources are separated from computing resources.
  - Storage devices of VMs are attached from network :
    - ***NAS***: NFS, CIFS
    - ***SAN***: Fibre Channel
    - ***iSCSI***, network block device
    - ***drdb*** network RAID
  - Require high quality network connection
    - Common L2 network (LAN)
    - L3 re-routing

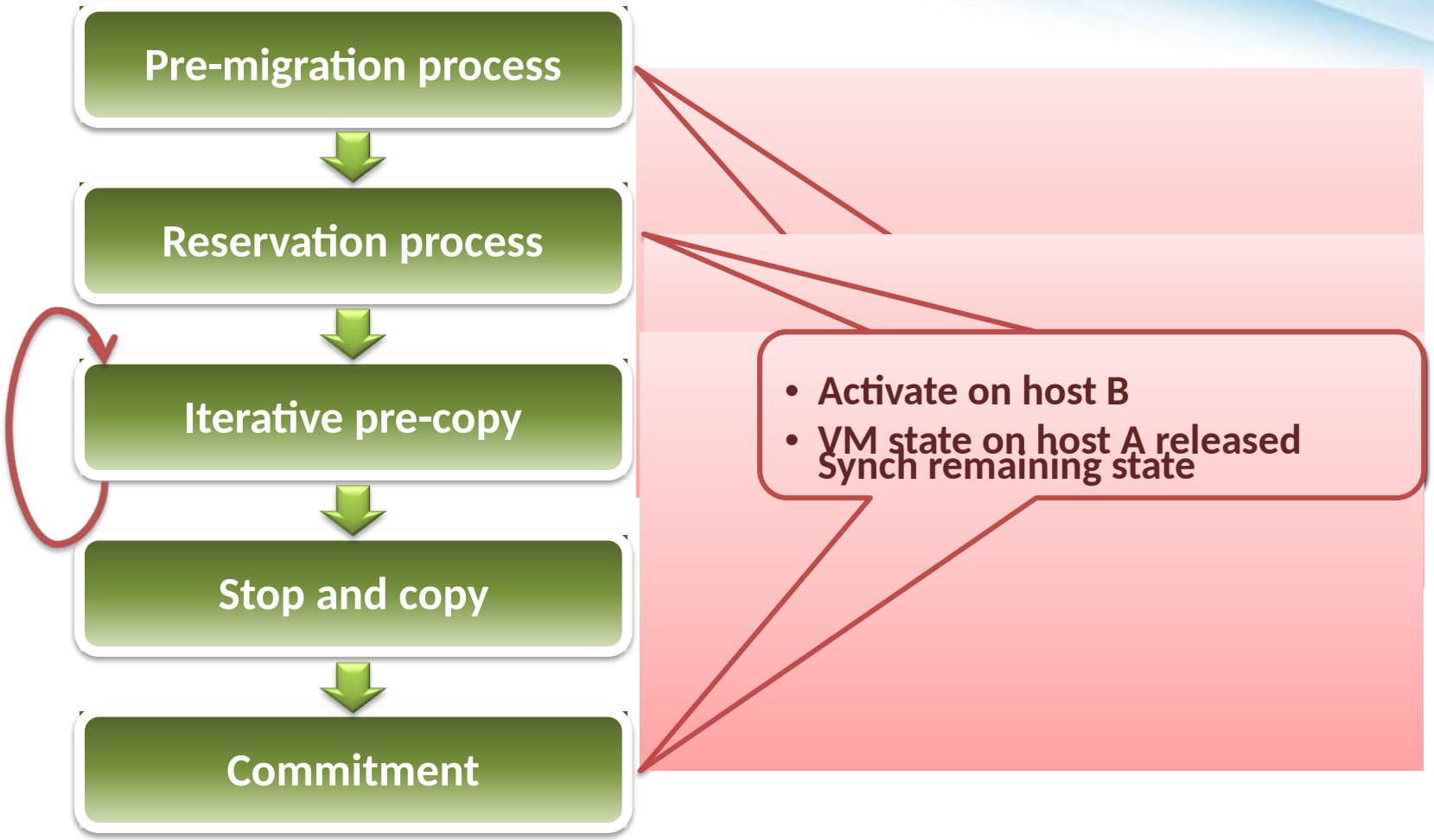


Live Migration Technique

- Challenges of live migration :
  - VMs have lots of state in memory
  - Some VMs have soft real-time requirements :
    - For examples, web servers, databases and game servers, …etc.
    - Need to minimize down-time
- Relocation strategy :
  1. Pre-migration process
  2. Reservation process
  3. Iterative pre-copy
  4. Stop and copy
  5. Commitment



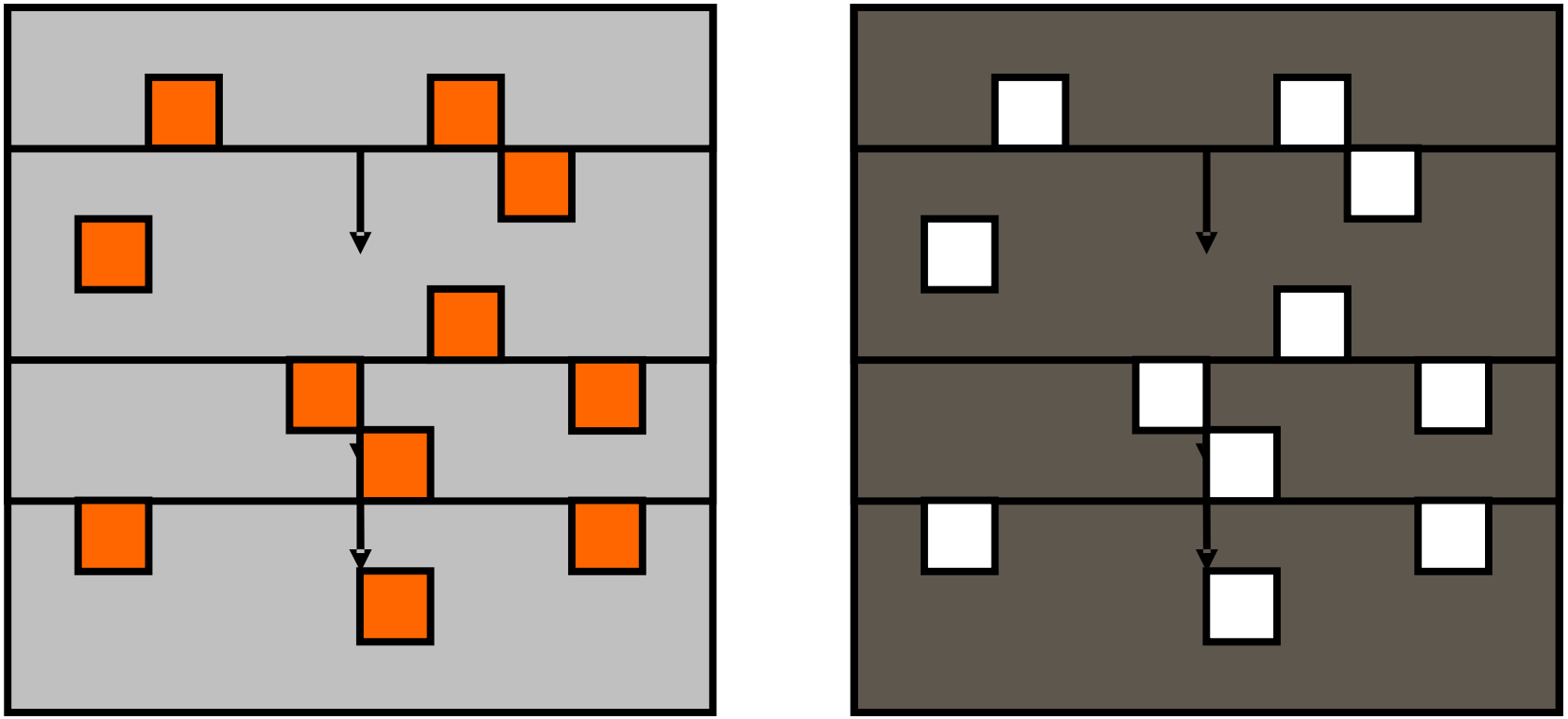
Live Migration Technique



Live Migration Technique

- Live migration process :

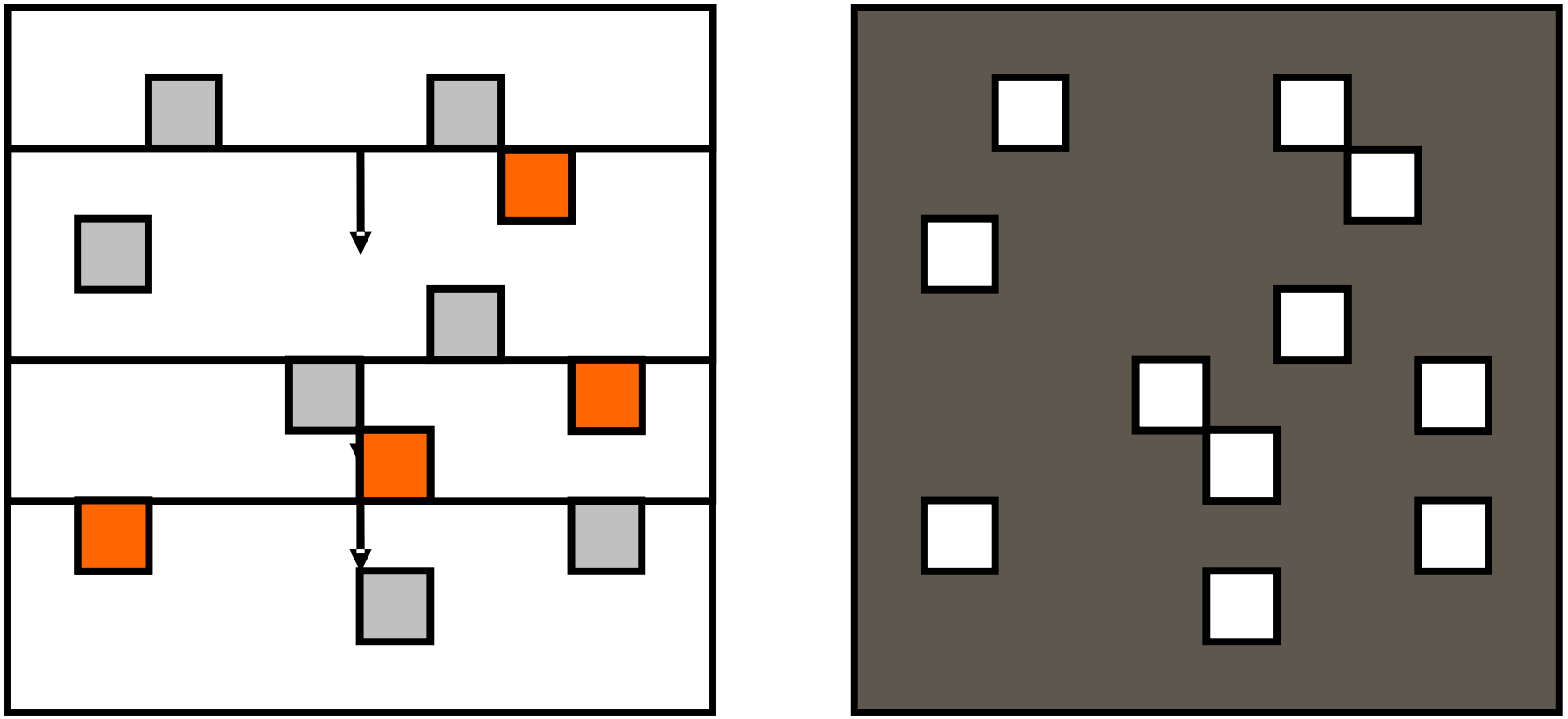
Pre-copy migration : Round 1



Live Migration Technique

- Live migration process :

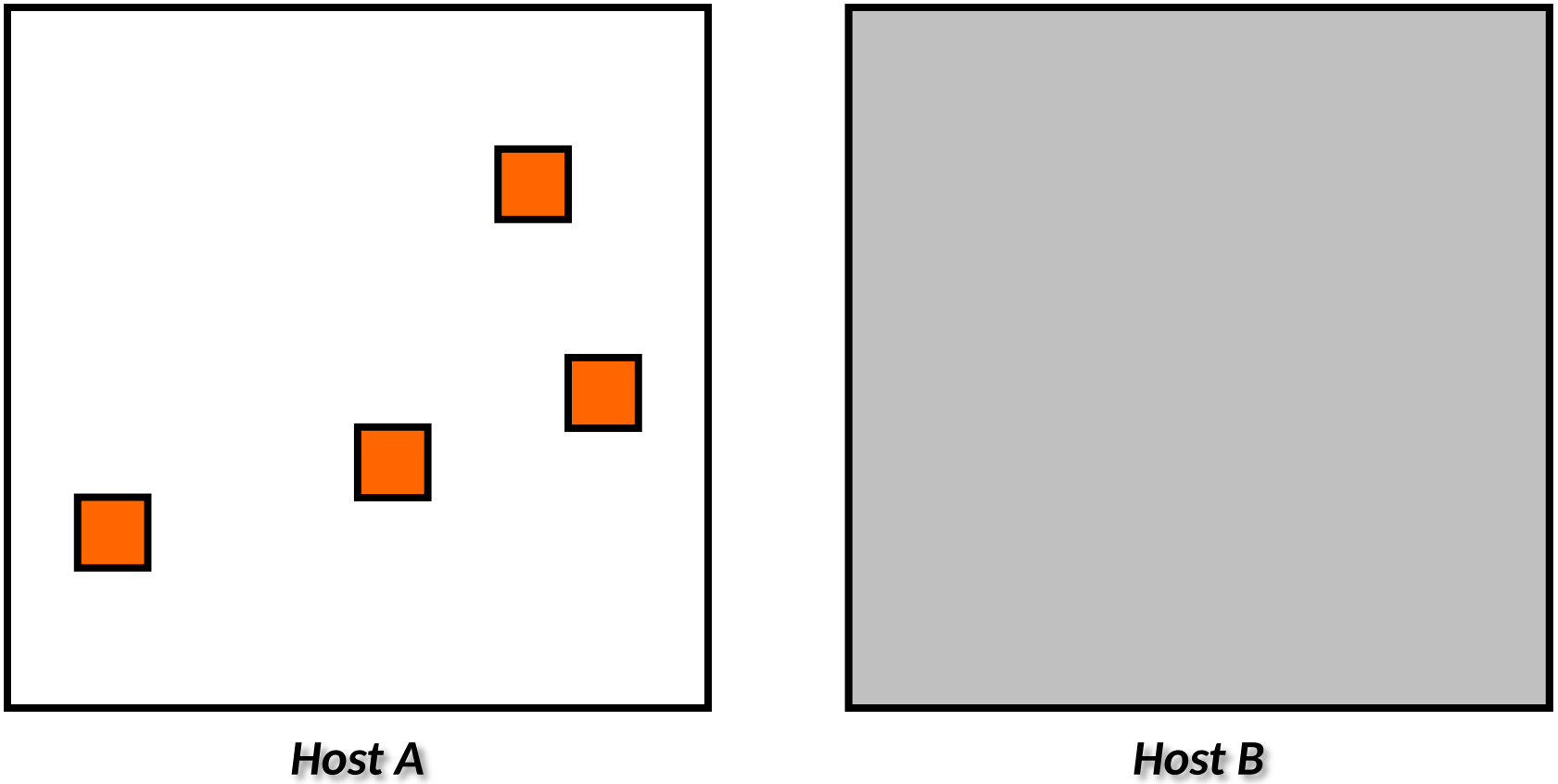
Pre-copy migration : Round 2



Live Migration Technique

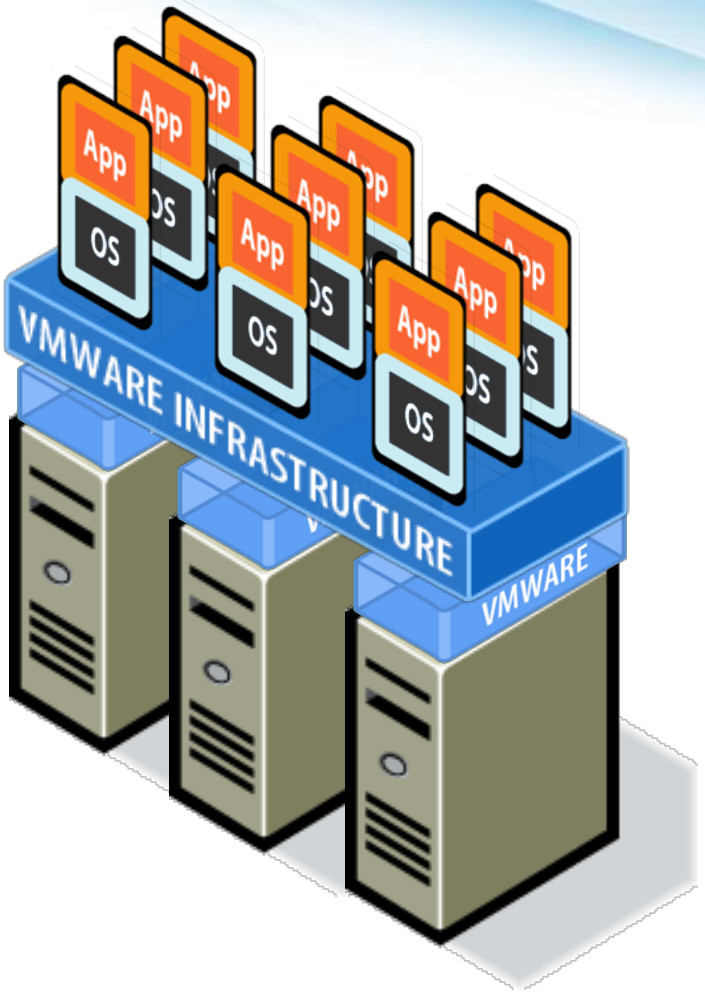
- Live migration process :

Stop and copy : Final Round



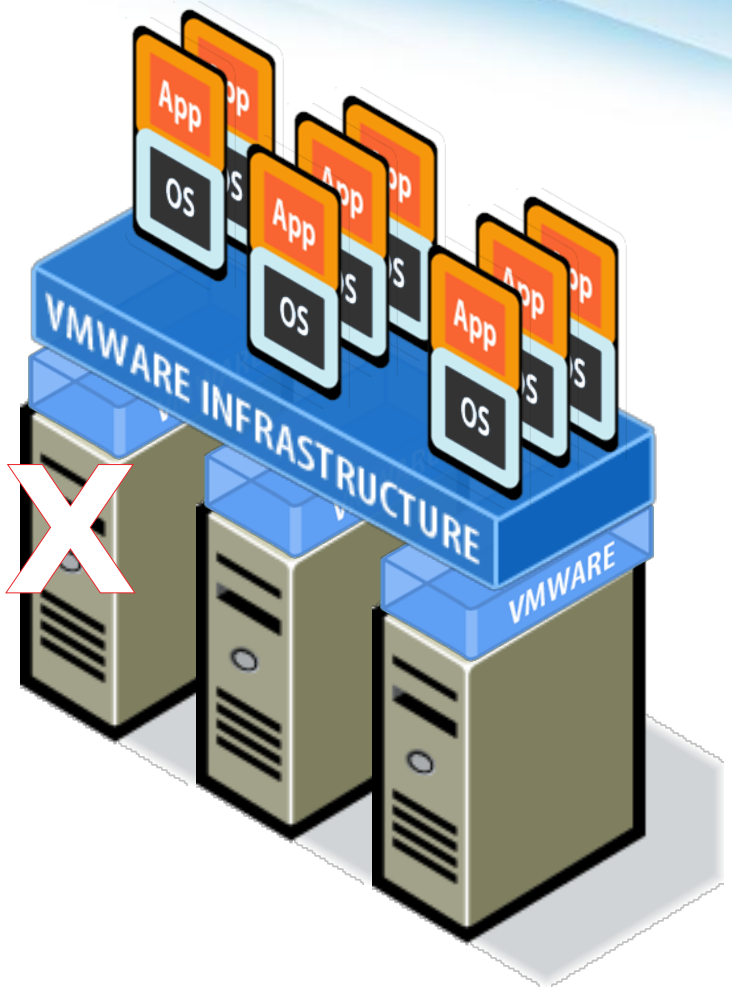
Scalability in Virtualization

- Scalability implement by VMware:
  - VMware VMotion, makes it possible to move Virtual Machines, without interrupting the applications running inside.
  - Dynamically scale up virtual machine system among physical servers.



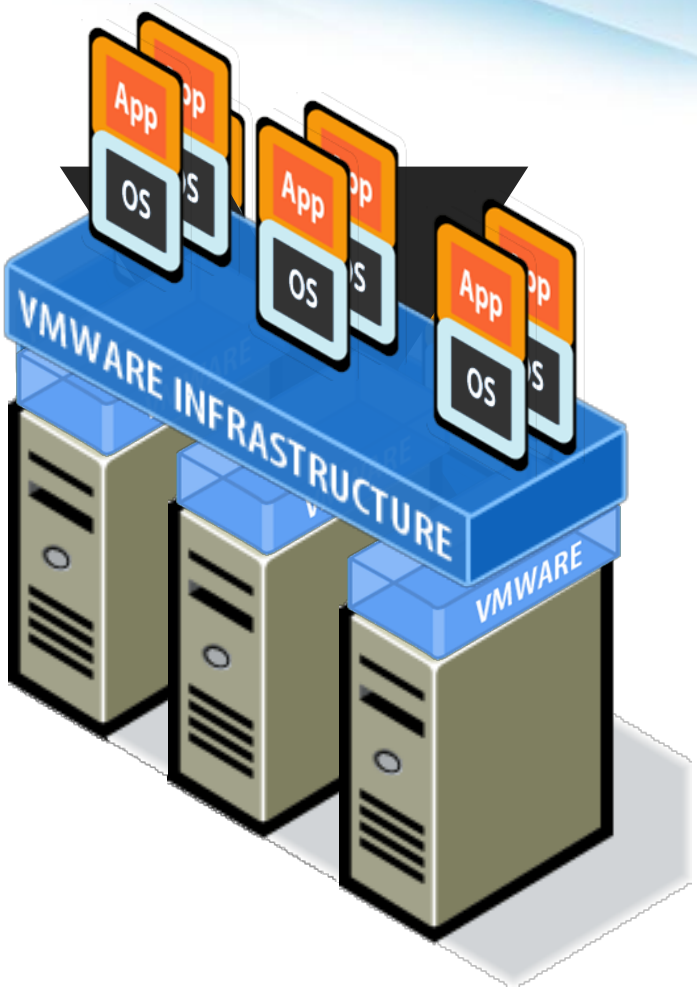
Availability in Virtualization

- Fault tolerance system :
  - VMware makes all Servers and Applications protected against component and complete system failure.
  - When system failure occurs, virtual machines will be automatic restarted on other physical servers.



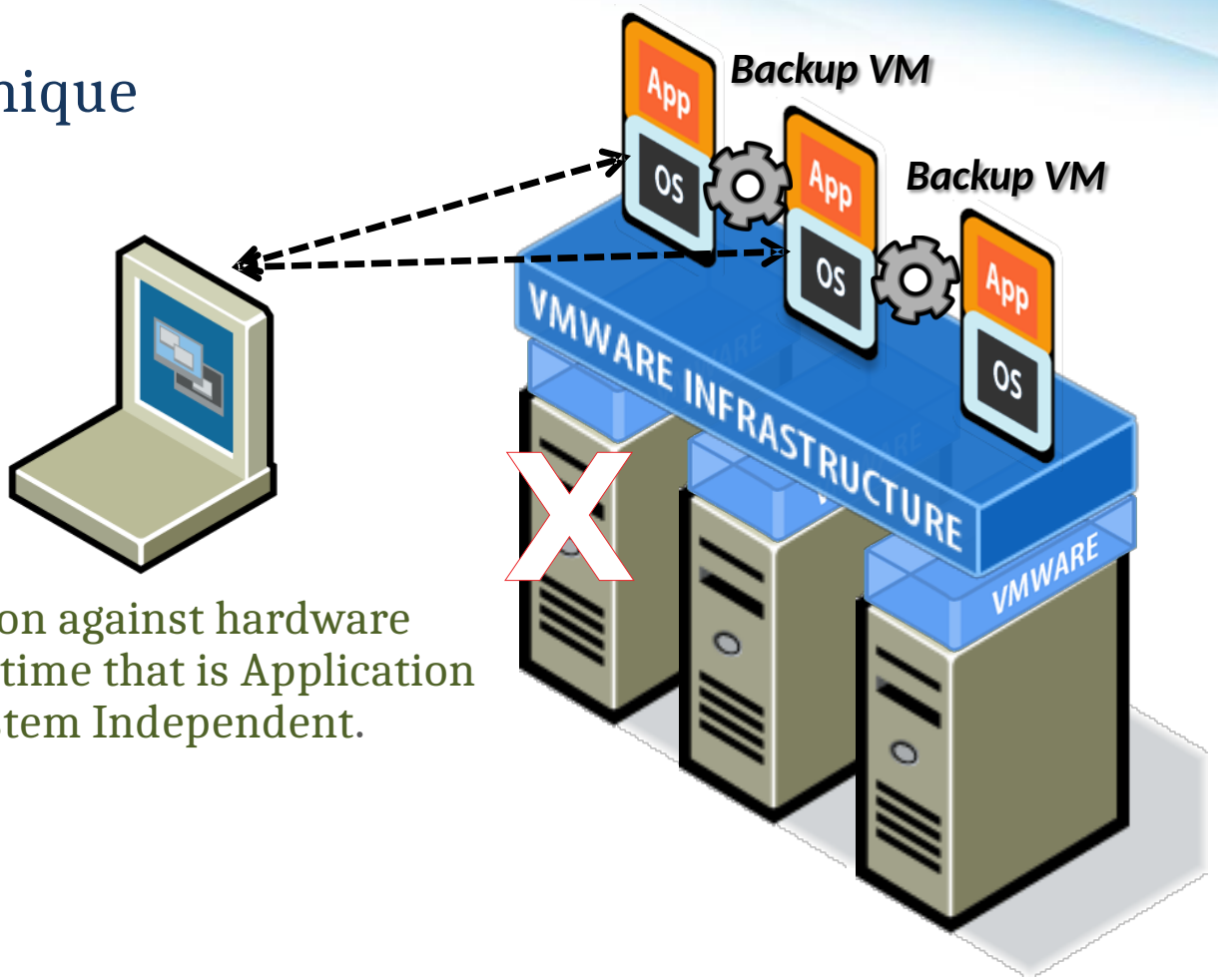
Availability in Virtualization

- Disaster recovery :
  - VMware Site Recovery Manager enables an easy transition from a production site to a Disaster Recovery site.
  - Easy Execution for real Disaster
  - Easy Testing for good night sleep



Availability in Virtualization

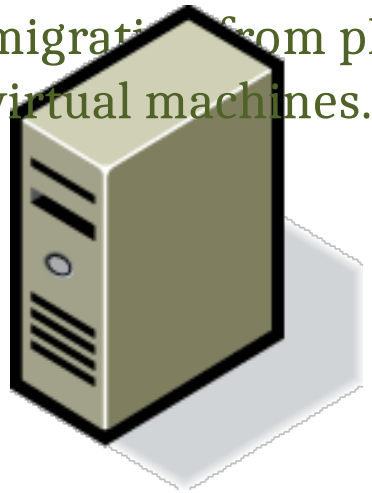
- Fail over technique

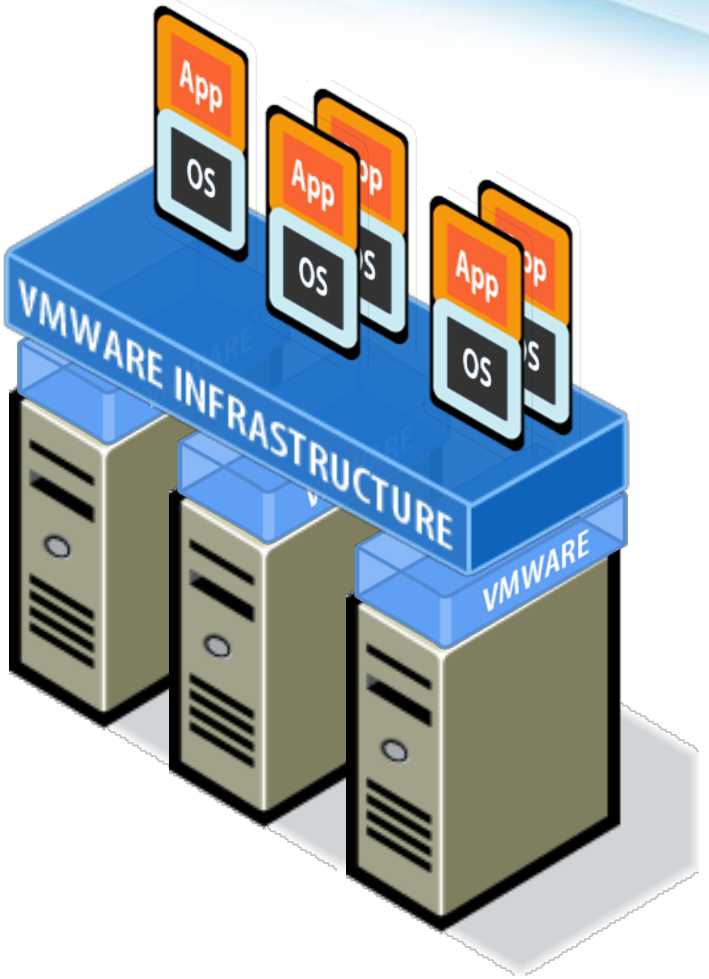


Application protection against hardware failures, with NO down time that is Application and Operating System Independent.

Manageability in Virtualization

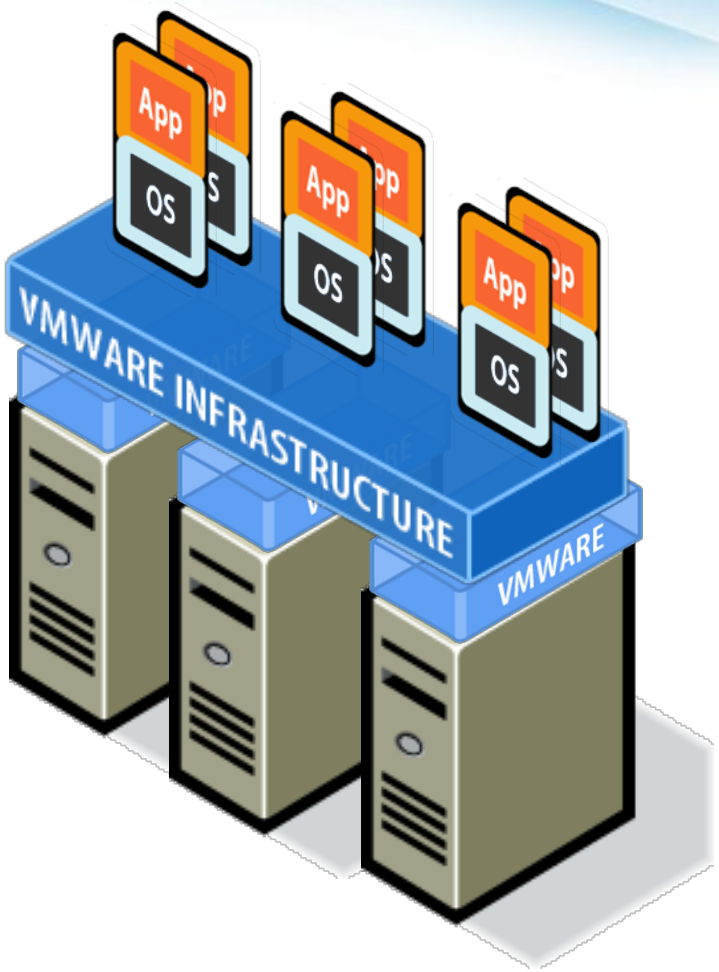
- Provide physical to virtual translation :
  - Consolidation Management with the VMware Infrastructure software will automate the migration from physical to virtual machines.





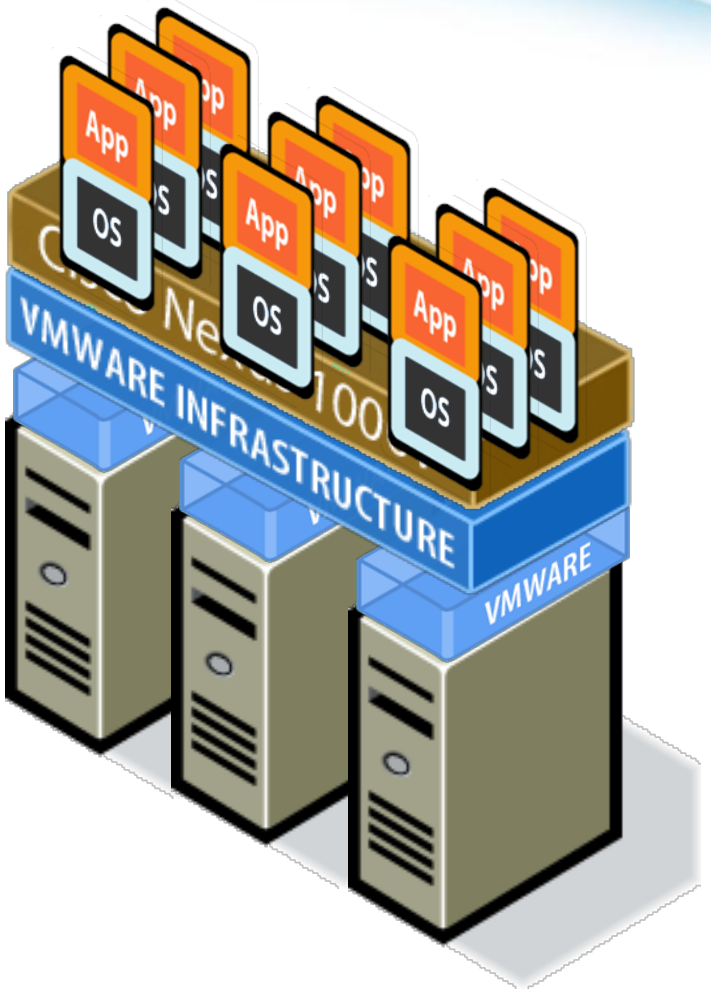
Performance in Virtualization

- Dynamic load balancing :
  - VMware Distributed Resource Scheduler automatically balances the Workloads according to set limits and guarantees.
  - Removing the need to predict resource assignment.



Performance in Virtualization

- Optimize network access :
  - VMware and Cisco are collaborating to enhance workload mobility and simpler management with virtualization-aware networks.

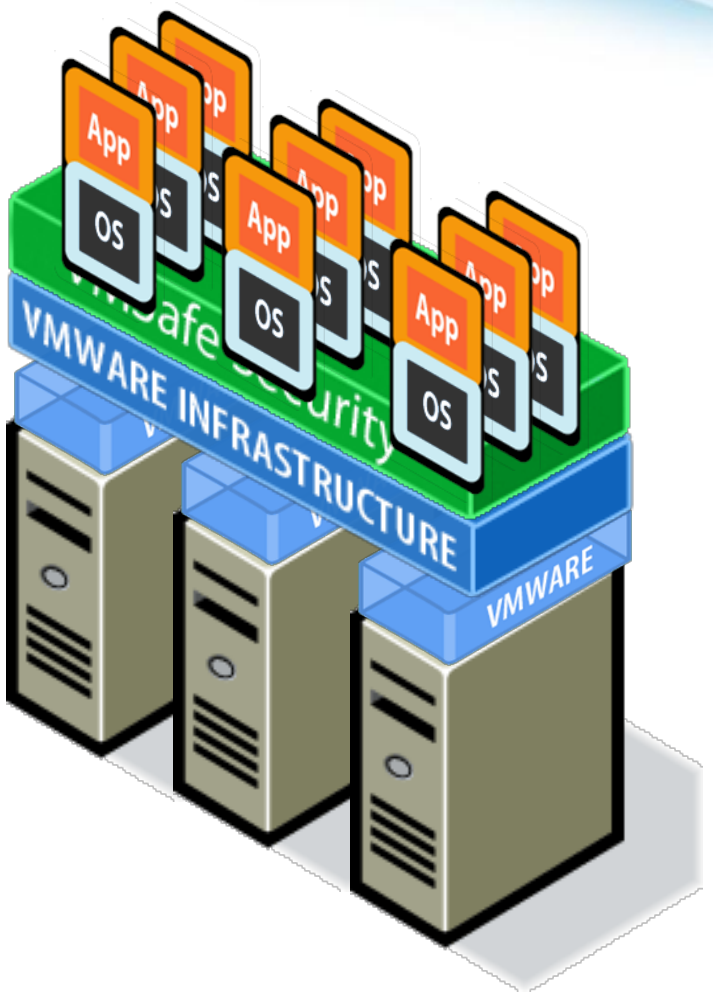


CISCO

Nexus 1000V

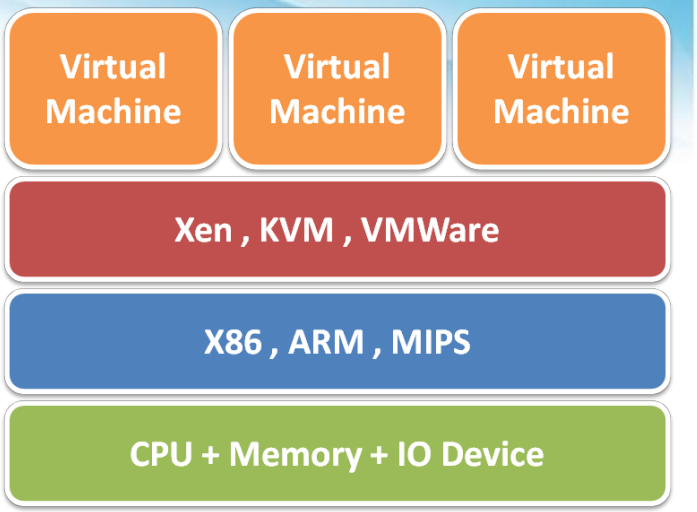
Security in Virtualization

- Enhance virtual machine security protection :
  - The Application vService VMSafe allows security vendors to add superior security solutions inside the VMware Infrastructure.



Summary

- Server virtualization technique :
  - CPU virtualization
    - Ring compression, Intel VT-x, ...etc
  - Memory virtualization
    - Shadow page table, Intel EPT, ...etc
  - IO virtualization
    - Device model, Intel VT-d, PCIe SR-IOV, ...etc



- Ecosystem :
  - VMware implements both type-1 & type-2 virtualization
  - Xen implements both para and full virtualization
  - KVM implements in Linux mainstream kernel
- Cloud properties :
  - Enabled by live migration technique
  - Scalability, Availability, Manageability and Performance

References

- Books :
  - James E. Smith & Ravi Nair, ***Virtual Machines***, Elsevier Inc., 2005
  - 英特爾開源軟件技術中心 & 復旦大學並行處理研究所，系統虛擬化 原理與實現北京 : 清華大學出版社, 2009.03
- Web resources :
  - Xen project *http://www.xen.org*
  - KVM project *http://www.linux-kvm.org/page/Main_Page*
  - IBM VirtIO survey *https://www.ibm.com/developerworks/linux/library/l-virtio*
  - PCI-SIG IO virtualization specification *http://www.pcisig.com/specifications/iov*
- Other resources :
  - Lecture slides of “Virtual Machine” course (5200) in NCTU
  - Vmware Overview Openline presentation slides *http://www.openline.nl*
  - Xen presentation *http://www.cl.cam.ac.uk/research/srg/netos/papers/2006-xen-fosdem.ppt*