BITS Pilani

Cloud Computing

CC ZG527

Shwetha Vittal
shwetha.vittal@pilani.bits-pilani.ac.in

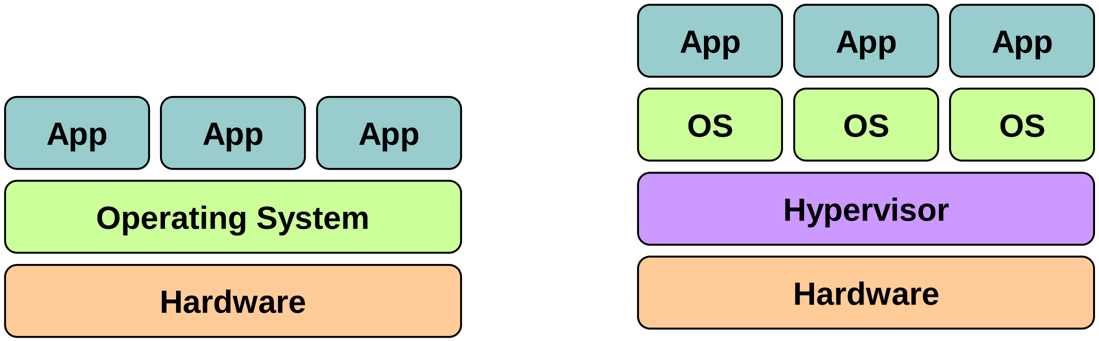
Agenda

Virtualization Techniques and Types

- ❑ Introduction to Virtualization
- ❑ Emergence of Virtualization
- ❑ Types of Virtualization
- ❑ Types of Hypervisors
- ❑ Use & demerits of Virtualization

Technology That Made Cloud Possible

Key Technology is Virtualization



Virtualization gives:

- An enabling technology for datacentre implementation
- An abstract compute, network, and storage service platforms from the underlying physical hardware

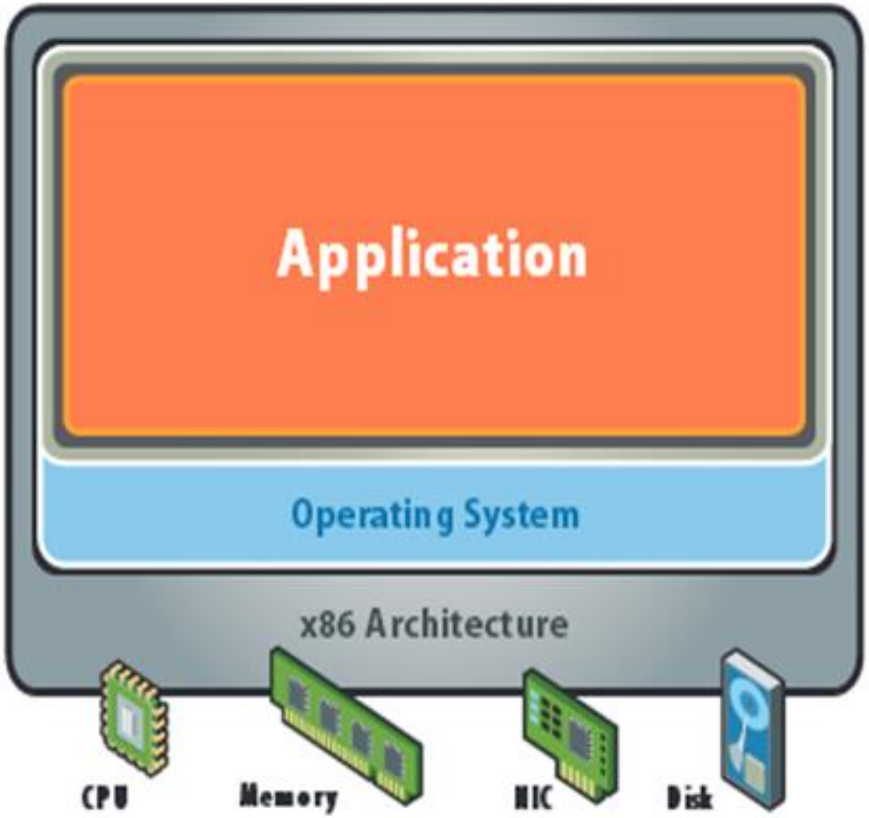
Importance of Virtualization in Cloud Computing

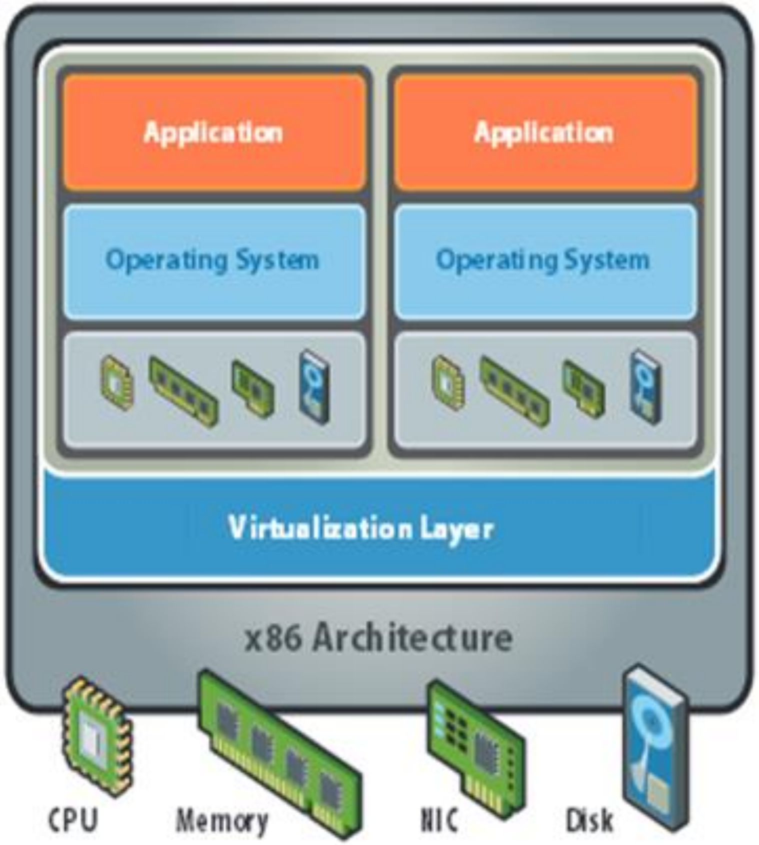
- Cloud can exist without Virtualization, although it will be difficult and inefficient.
- Cloud makes notion of “Pay for what you use”, “infinite availability- use as much you want”.
- These notions are practical only if we have
  - lot of flexibility
  - efficiency in the back-end
- This efficiency is readily available in Virtualized Environments and Machines

Virtualization Re-emergence

- Server Sprawl: Multiple redundant servers present with low utilization
- Goal was to improve server utilization with application isolation
- Server Consolidation
- Platform independence by Virtual Machine Encapsulation
- Improved scalability and availability with per application isolation
- Faster provisioning for newer applications

What is Virtualization?

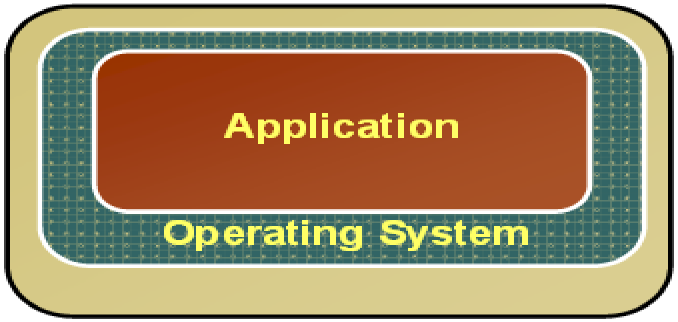


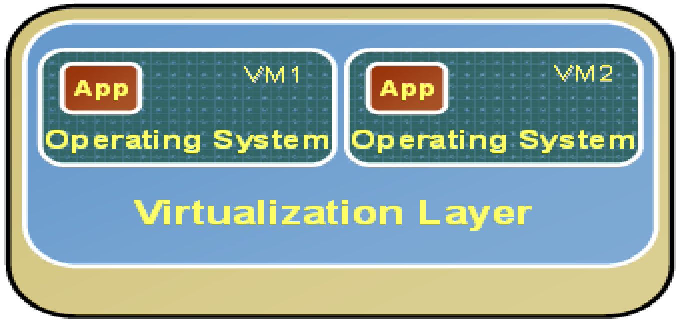


What does Virtualization do?

- Virtualization allows multiple operating system instances to run concurrently on a single computer
- Each “guest” OS is managed by a Virtual Machine Monitor /Manager (VMM), also known as a hypervisor.
- Because the virtualization layer sits between the guest and the hardware,
    - it can control the guests’ use of CPU, memory, and storage
    - Allows a guest OS to migrate from one machine to another

Changes after Virtualization





Before Virtualization

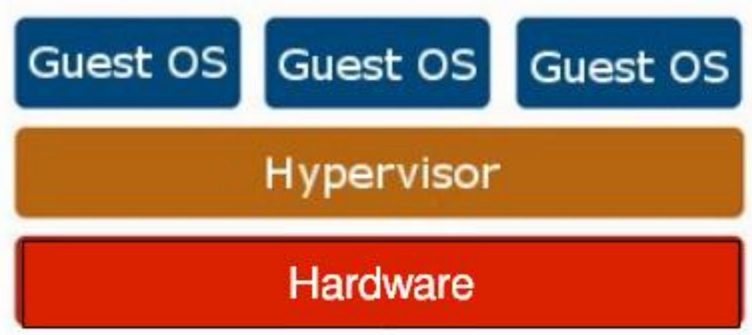
- Single OS image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources
- Inflexible and costly infrastructure

After Virtualization

- Hardware-independence of operating system and applications
- Virtual machines can be provisioned to any system
- Can manage OS and application as a single unit by encapsulating them into virtual machines

Virtualization Architecture

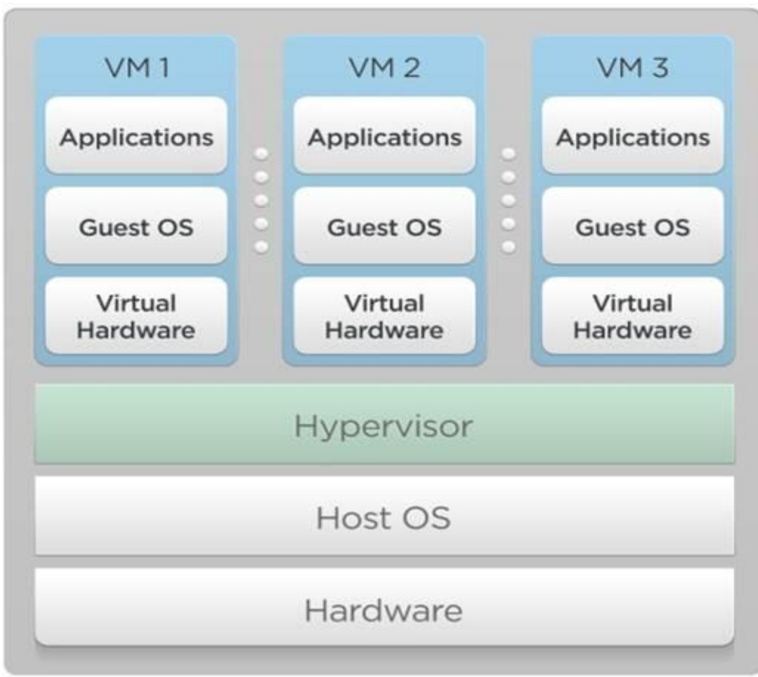
- OS assumes complete control of the underlying hardware.
- Virtualization architecture provides this illusion through a **Hypervisor/VMM.**
- Hypervisor/VMM is a software layer which:
    - Allows multiple Guest OS (Virtual Machines) to run simultaneously on a single physical host
    - Provides hardware abstraction
    - Multiplexes underlying hardware resources

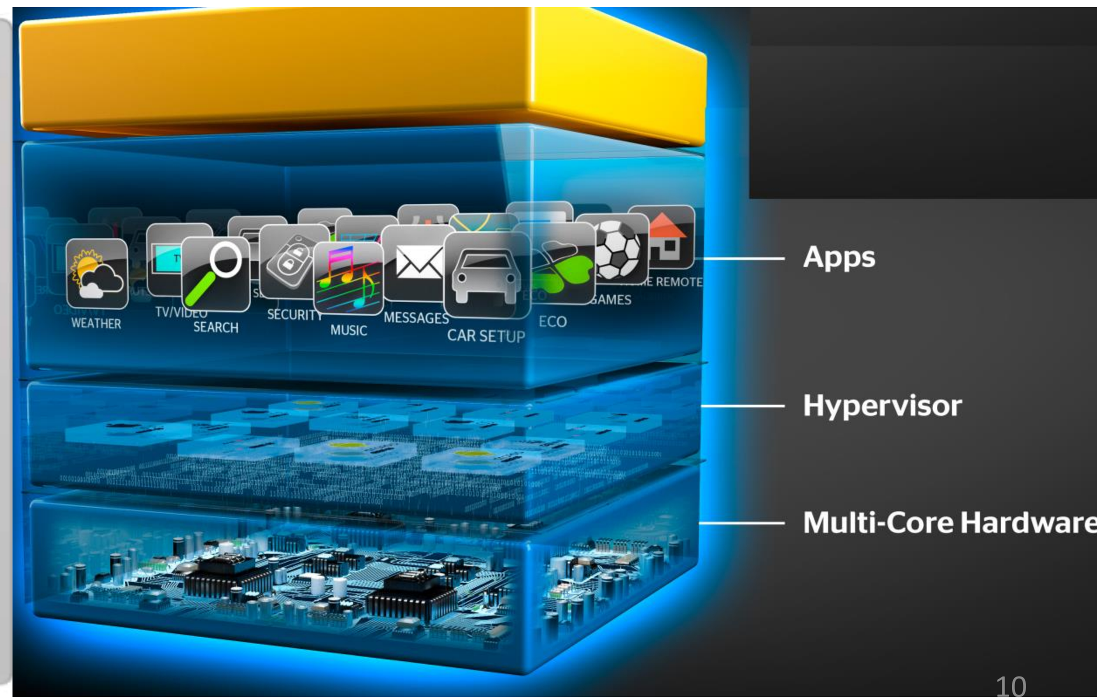


Hypervisor

A layer of software that generally provides virtual partitioning capabilities which runs directly on hardware.

Sometimes referred to as a “bare metal” approach.





Principles of Virtualization

- **Equivalence**
  - Guest OS should run (close to) unmodified

- **Safety**
  - Should (absolutely) not be able to escape its isolated environment

- **Performance**
  - With (close to) native performance

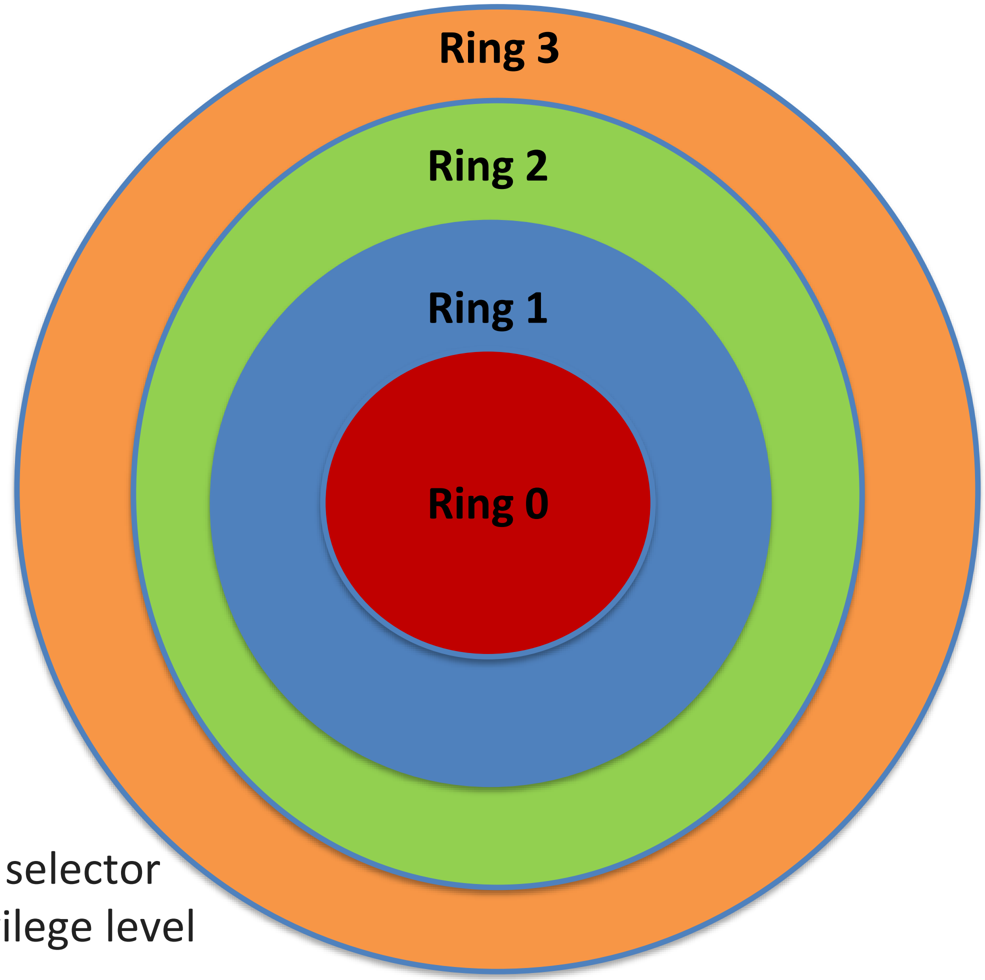
Hypervisor Design Goals

- **Isolation**
    - Security isolation
    - Fault isolation
    - Resource isolation
- **Reliability**
    - Minimal code base
    - Strictly layered design
- **Scalability**
    - Scale to large number of cores
    - Large memory systems

CPU Privileges

CPUs have multiple privilege levels

- Ring 0,1,2,3 in x86 CPUs
- Ring 0 – Highest Privilege
- Ring 3 – Lowest Privilege



Two bits in a register called the code selector (**CS**) register indicate the current privilege level or **CPL** of that program.

CPU Instructions & Registers

1. **Privileged**

   Modify the physical hardware have highest privilege level.

   Can be accessed only in ring 0.

   CR3, CR2 – Control registers, EFLAGS – has many flags like Interrupt flag, IOPL : IO Privilege level

2. **Non Privileged**
   - Do not have to be run in privileged mode.
   - E.g: A MOV (move one operand to another) instruction that does not operate on a privileged register

3. **Sensitive**: Can only be executed when CPL <= IOPL. Otherwise a GP exception will result.

4. **Privileged Without Exception**

[Masum Z Hasan, PhD - X86 Architecture Basics: Privilege Levels and Registers]

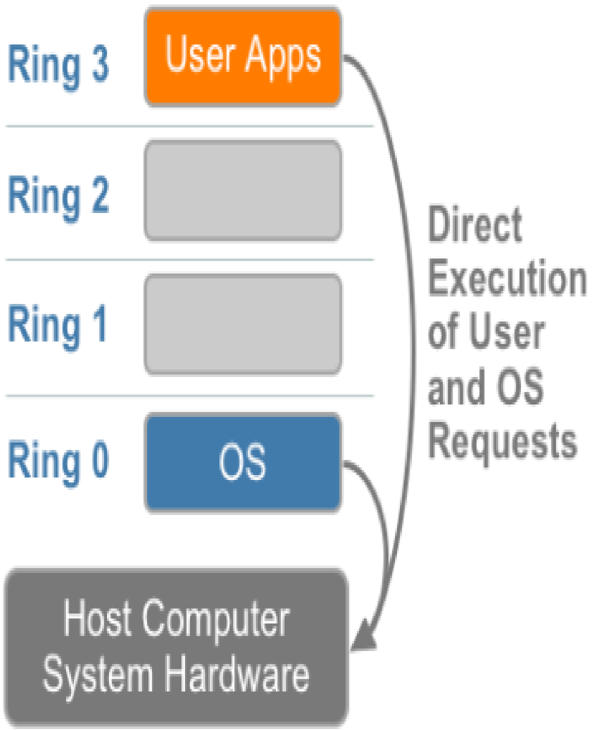
Trap Handling in OS

- Genuine System Call
- Illegal memory access : Unprivileged operation
- Interrupt from I/O
- Termed as “Trap” -> A Fault
- Trap Handler: Trap (,n)
  - CPU switches from user mode to kernel mode
  - Trap(,n) -> OS peeks into Interrupt Description Table (IDT) to handle the trap
  - N -> Trap number
  - Fetches the address of trap handler function

CPU Virtualization - 1

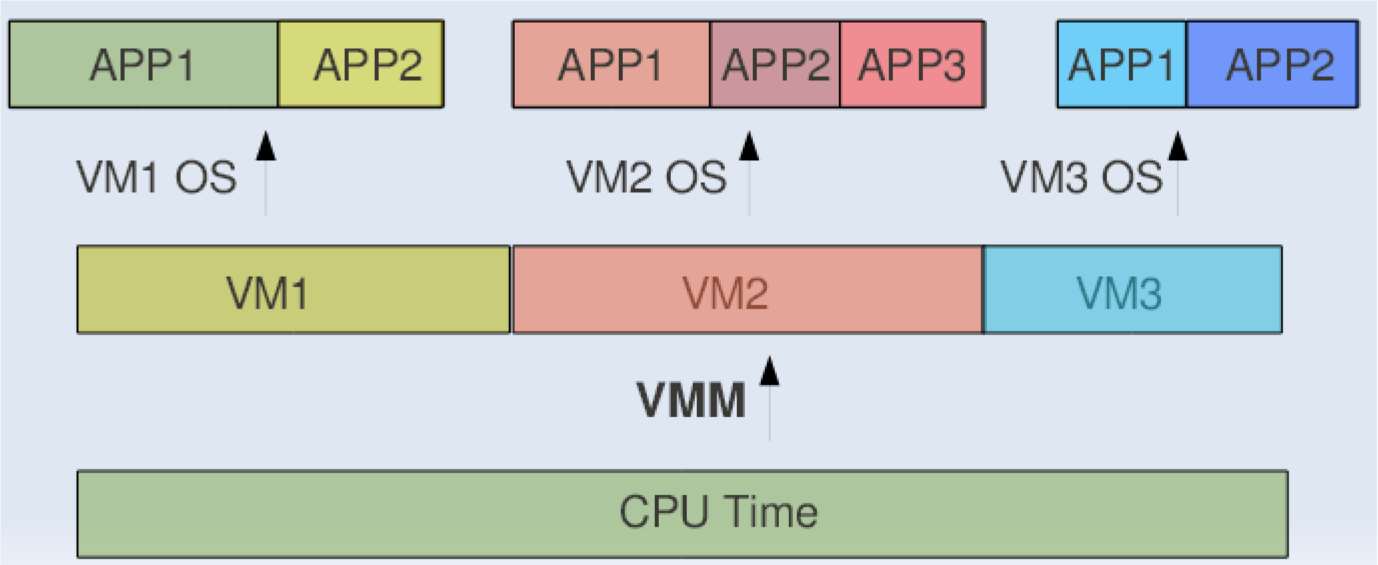
CPUs have multiple privilege levels

- Ring 0,1,2,3 in x86 CPUs
- Virtualizing the x86 architecture requires placing a virtualization layer under the operating system (which expects to be in the most privileged Ring 0)
- Normally,
  - User process in ring 3,
  - Host OS in ring 0
  - Privileged instructions only run in ring 0
    - VMM in ring 0
- Guest OS must be protected from guest apps
  - But not fully privileged like host OS/VMM
  - Run it in ring 1?



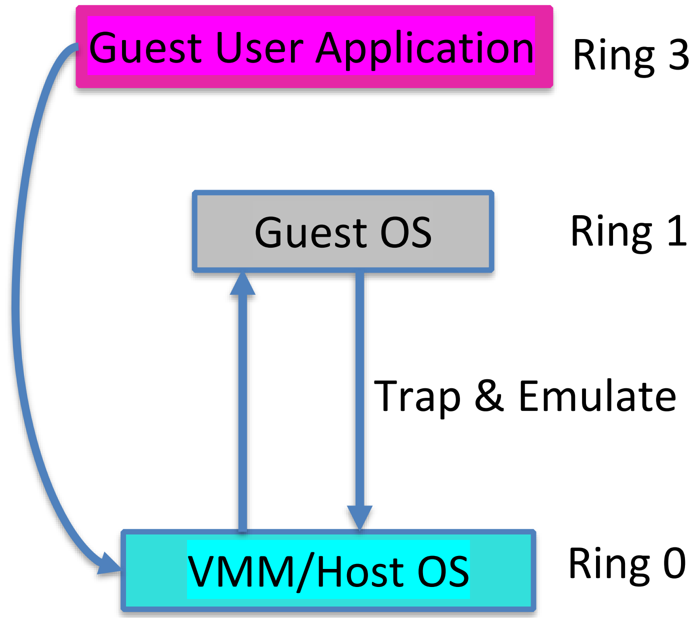
CPU Virtualization - 2

- VMM or Hypervisor provides a virtual view of CPU to VMs.
- In multi processing, CPU is allotted to the different processes in form of time slices by the OS.
- Similarly VMM or Hypervisor allots CPU to different VMs.



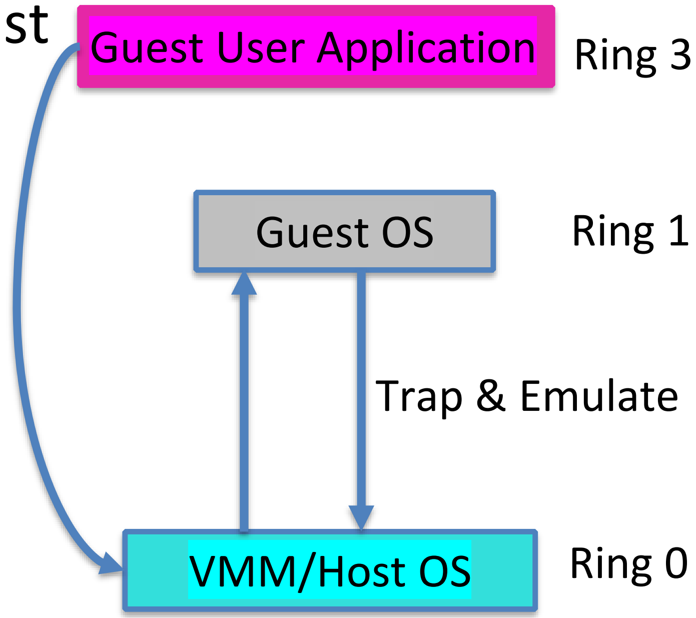
Trap & Emulate based Virtualization

- **Scope of Guest OS**
  - Runs at lower privilege level than VMM
  - Traps to VMM for privileged operation
- **Guest app handling syscall/interrupt**
  - Special trap instr (int n), traps to VMM
  - VMM doesn't know how to handle trap
  - VMM jumps to guest OS trap handler
  - Trap handled by guest OS normally
- **Returning from Trap**
  - Guest OS performs return from trap
  - VMM jumps to corresponding user process



Trap & Emulate based Virtualization

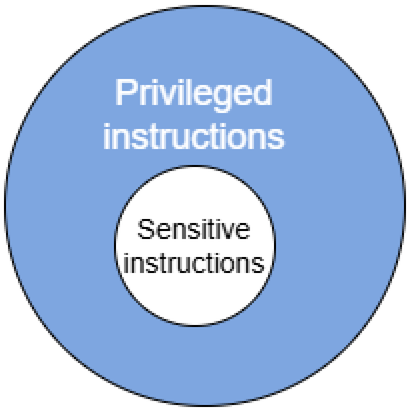
- **Handling privileged instruction**
  - Sensitive data structures like IDT must be managed by VMM, not guest OS
  - Any privileged action by guest OS traps to VMM, emulated by VMM
  - Guest OS traps to VMM
  - VMM jumps to corresponding user process
    - E.g: Set IDT, set CR3, access hardware

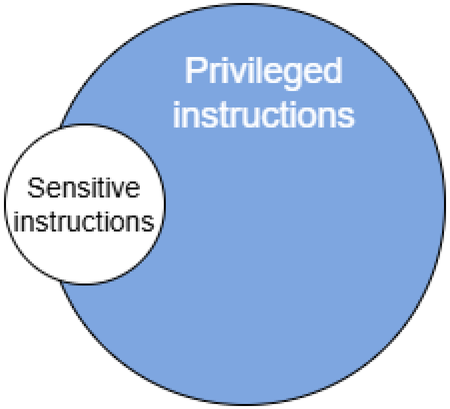


Problem with Trap & Emulate

- OSs not ready /unaware of Virtualization
  - To run at a lower privilege level
  - Machines not designed for virtualization
- Guest OS may realize it is running at lower privilege level
- Some registers in x86 reflect higher privilege level
- **Sensitive instructions**:
  - Can change hardware state, while running in both privileged and unprivileged modes
  - Will behave differently when guest OS is in ring 0 vs in less privileged ring 1
  - OS behaves incorrectly in ring1, will not trap to VMM

Popek Goldberg theorem



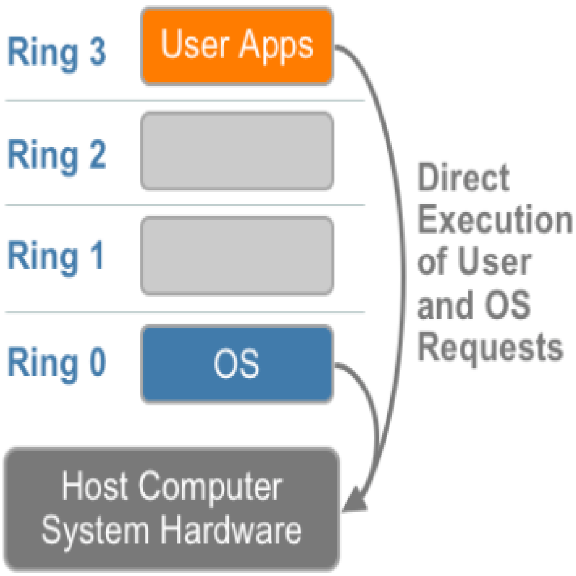


- **Sensitive instruction** may change hardware state
- **Privileged instruction** runs only in privileged mode
  - Traps to ring 0 if executed from unprivileged rings
- **In order to build a VMM efficiently via trap-and-emulate method, sensitive instructions should be a subset of privileged instructions**
- x86 does not satisfy this criteria, **so trap and emulate VMM is not possible**

Approaches to CPU Virtualization

Three techniques now exist for handling sensitive and privileged instructions to virtualize the CPU on the x86 architecture

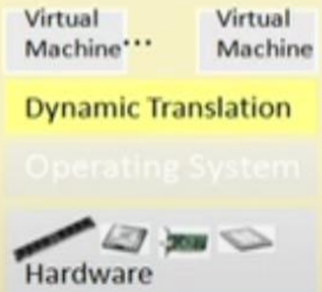
1. **Full virtualization** using binary translation
2. **Para-virtualization** or OS assisted virtualization
3. **Hardware assisted virtualization**



Evolution of CPU Virtualization

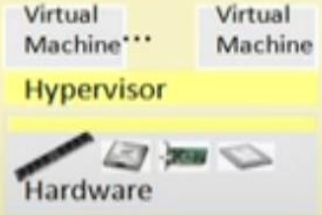
Evolution of Software Solutions

- 1st Generation: Full virtualization (Binary rewriting)
  - Software Based
  - VMware and Microsoft



- 2nd Generation: Para-virtualization
  - Cooperative virtualization
  - Modified guest
  - VMware, Xen

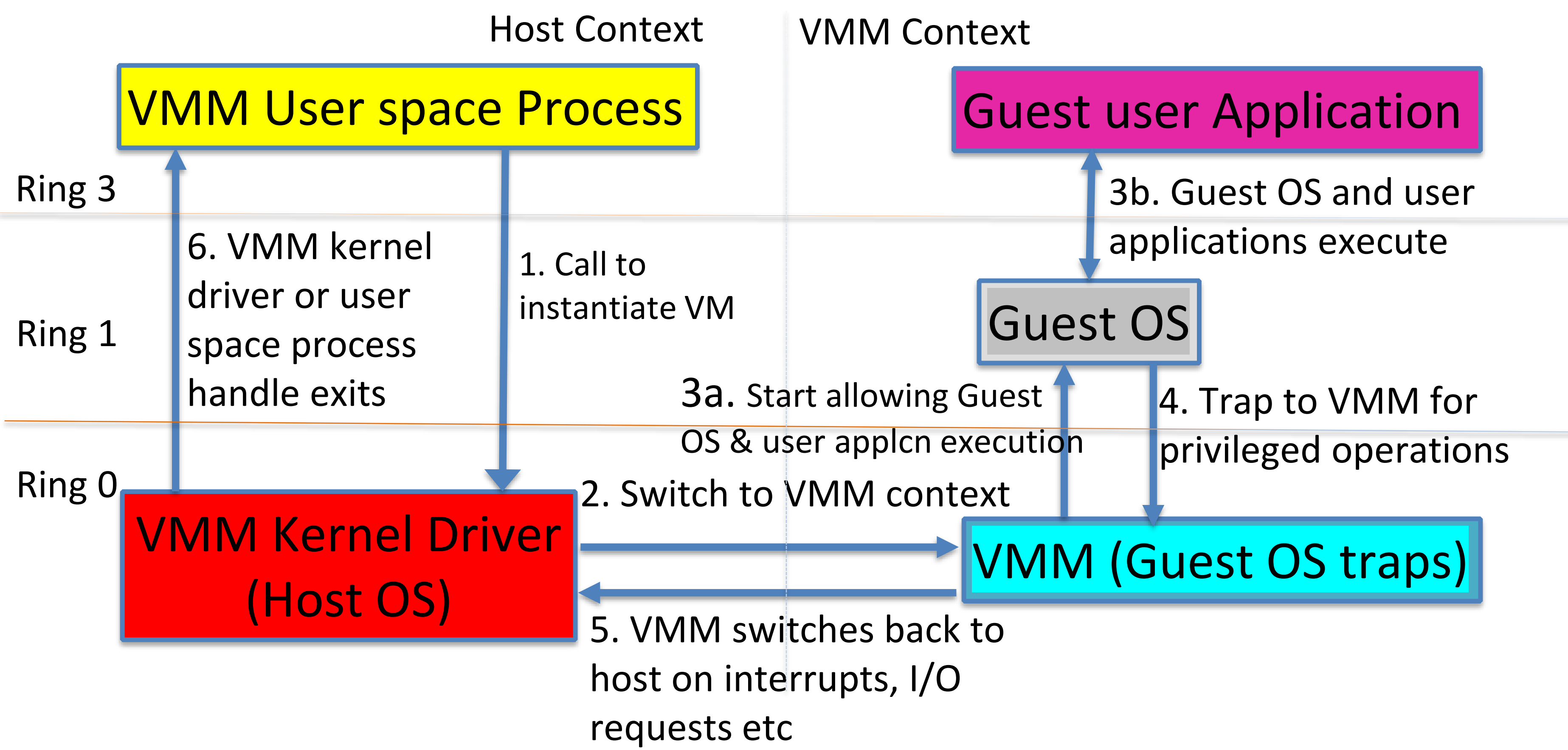
- 3rd Generation: Silicon-based (Hardware-assisted) virtualization
  - Unmodified guest
  - VMware and Xen on virtualization-aware hardware platforms



Time

Virtualization Logic

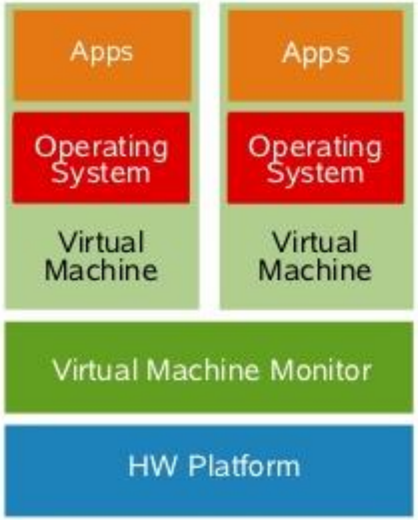
Full Virtualization

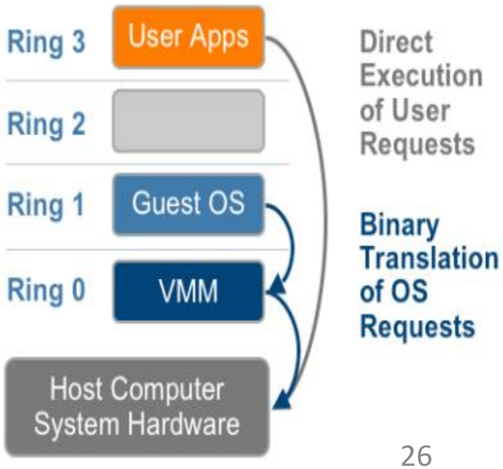


"Bringing Virtualization to the x86 Architecture with the Original VMware Workstation", Edouard Bugnion, Scott Devine, Mendel Rosenblum, Jeremy Sugerman, Edward Y. Wang.

Full Virtualization using Binary Translation

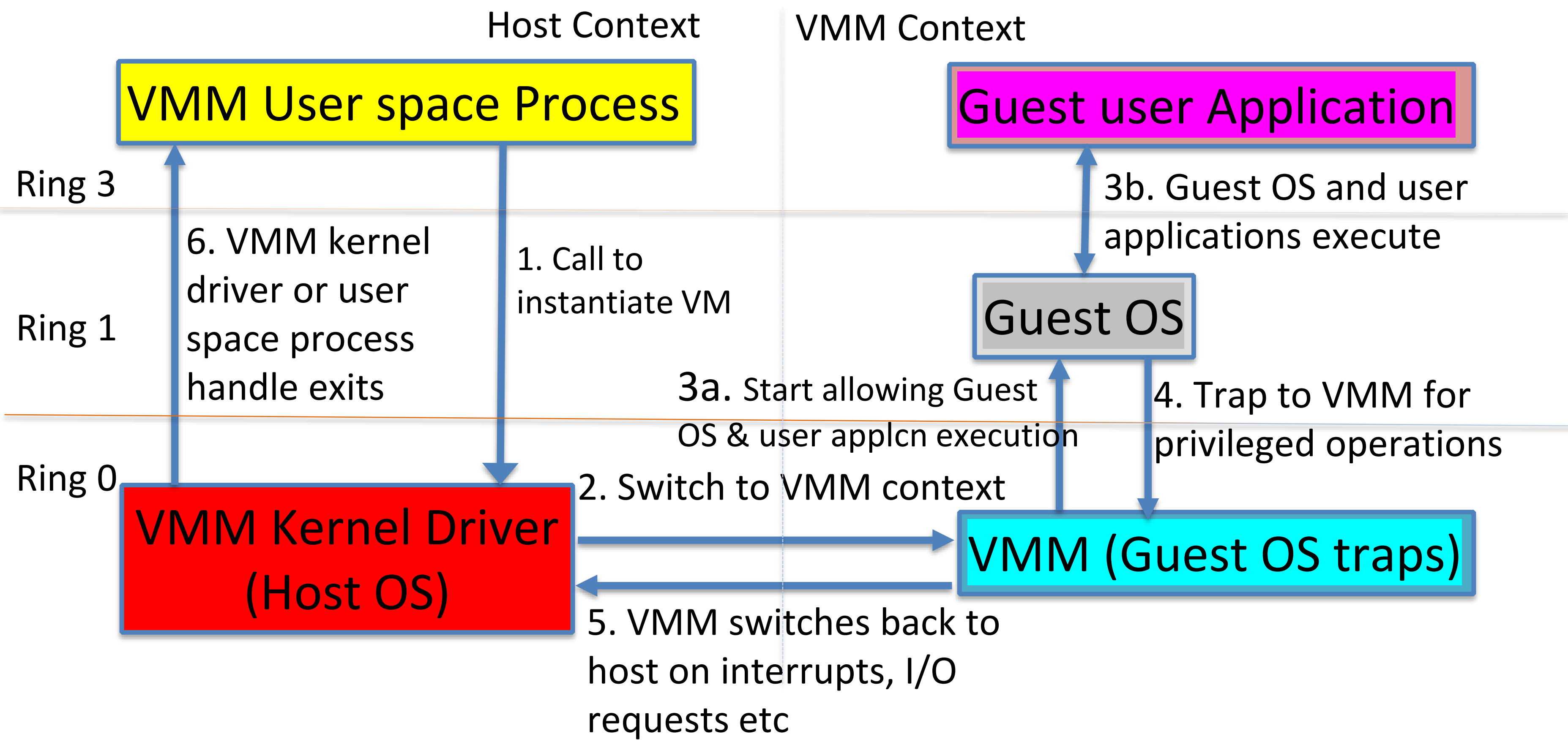
- Vmware(Full Virtualization) can virtualize any x86 operating system using a combination of
  - binary translation
  - direct execution techniques
- User level code is directly executed on the processor for high performance virtualization
- Kernel code is translated to replace non-virtualizable instructions with new sequences of instructions
- This combination of binary translation and direct execution provides Full Virtualization
- Guest OS is fully abstracted (completely decoupled) from the underlying hardware by the virtualization layer
- VMWare Workstation[1] is an example for full virtualization





"Bringing Virtualization to the x86 Architecture with the Original VMware Workstation", Edouard Bugnion, Scott Devine, Mendel Rosenblum, Jeremy Sugerman, Edward Y. Wang – 2012 ACM Transactions.

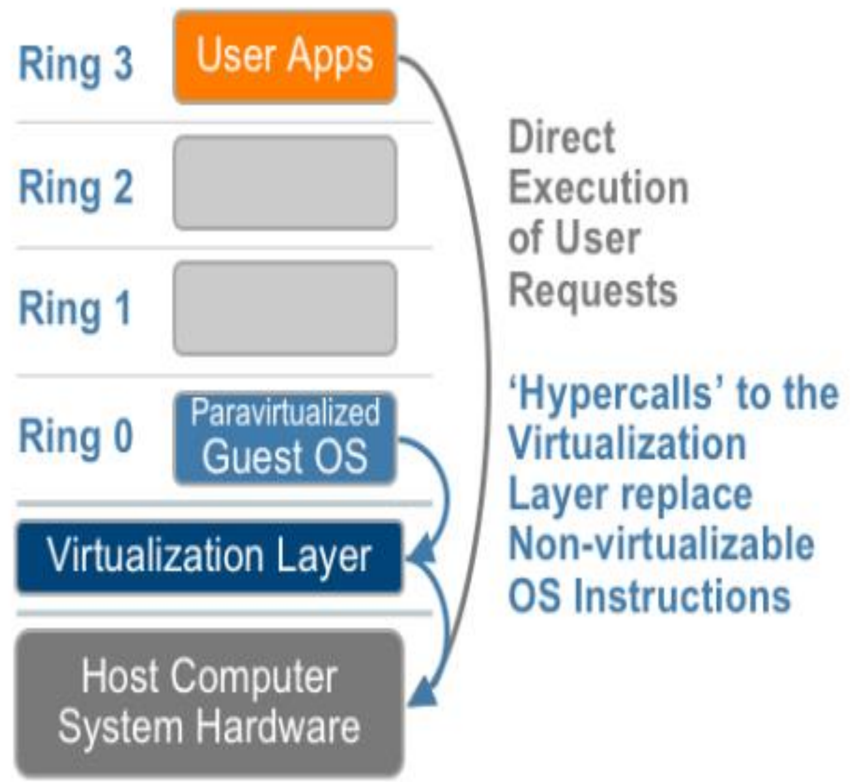
Para Virtualization



“Xen and the Art of Virtualization”, Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield

Para Virtualization

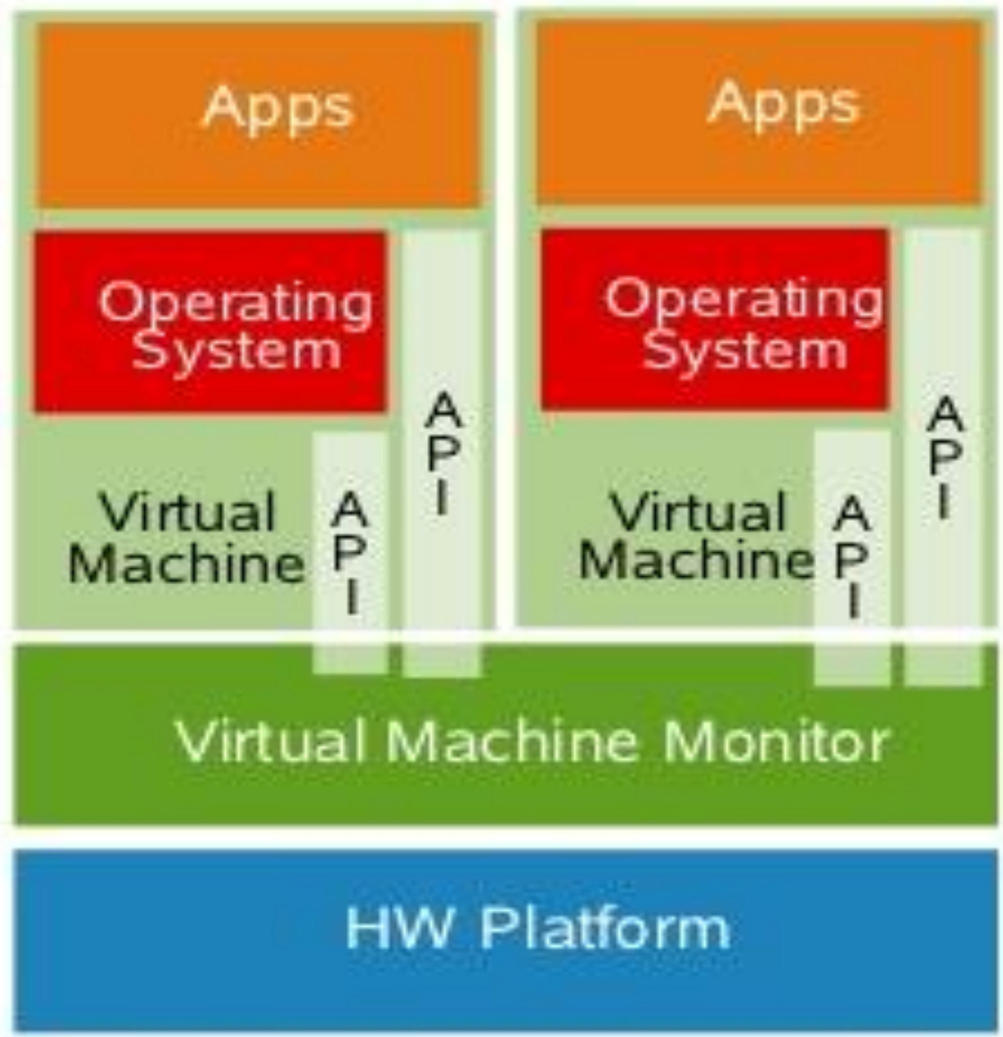
- Para virtualization involves modifying the OS kernel to replace non-virtualizable instructions with **hypercalls**.
- **Hypercalls** communicate directly with the virtualization layer
- The hypervisor also provides hypercall interfaces for other critical kernel operations such as:
  - Memory management
  - Interrupt handling
- Ex: open-source Xen – Type1 Hypervisor



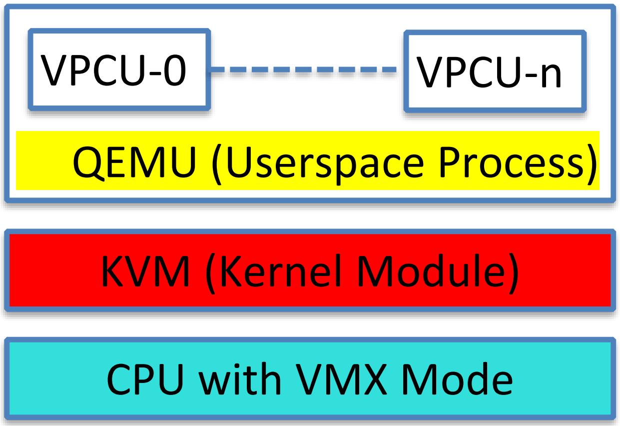
"Xen and the Art of Virtualization", Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield

Para Virtualization

- Para virtualization cannot support unmodified operating systems
- Compatibility and portability is poor
- Ex: open-source Xen



Hardware-assisted Virtualization

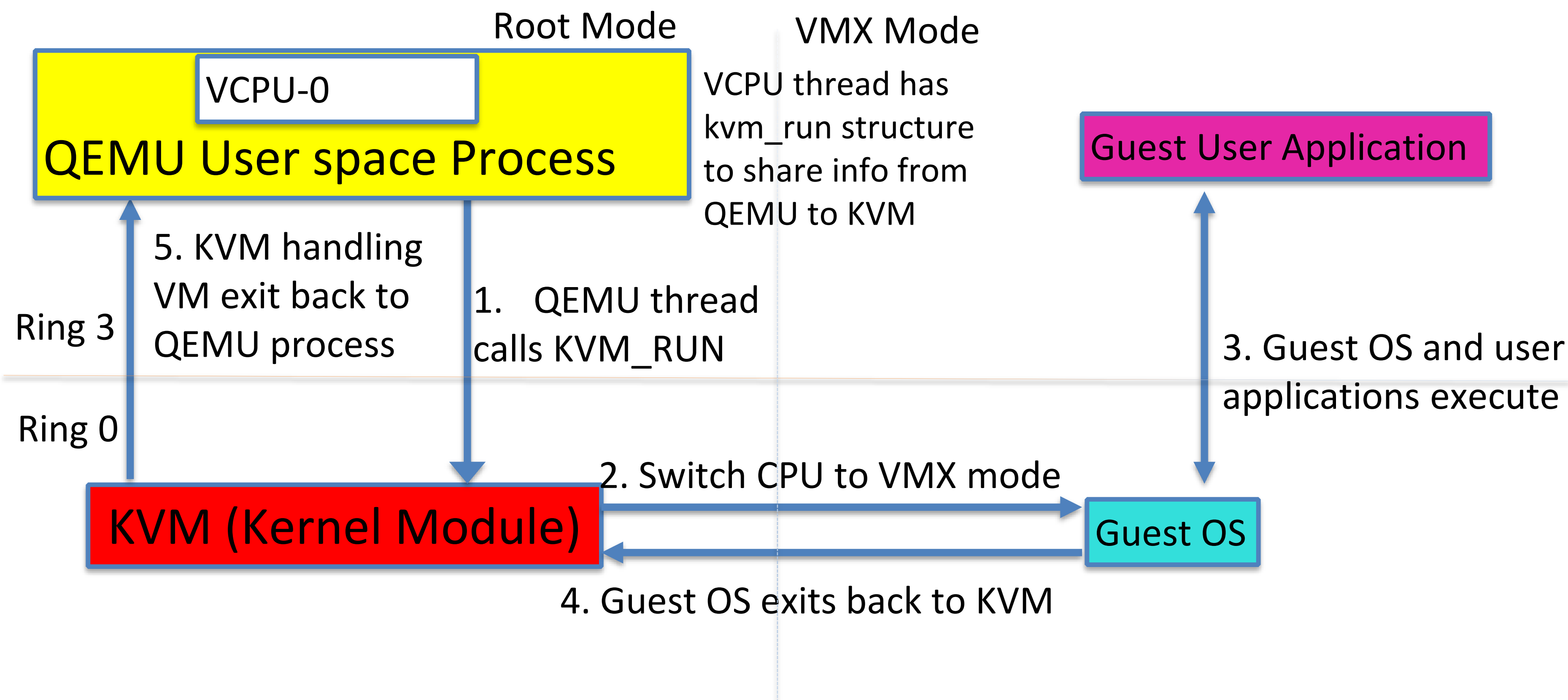


- Intel Virtualization Technology (VT-x) and AMD's AMD-V
- Target privileged instructions with a new CPU execution mode feature that allows the VMM to run in a new root mode below ring 0

Example: QEMU/KVM in Linux

- QEMU is user space process
- QEMU talks to KVM via open/ioctl syscalls
- Host OS sees QEMU as a regular multi-threaded process
- Creates one thread for each virtual CPU (VCPU) in guest
- Multiple file descriptors to /dev/kvm(one for QEMU, one for VM, one for VCPU and so on)
- Ioctl on fds to talk to KVM

Hardware-assisted Virtualization

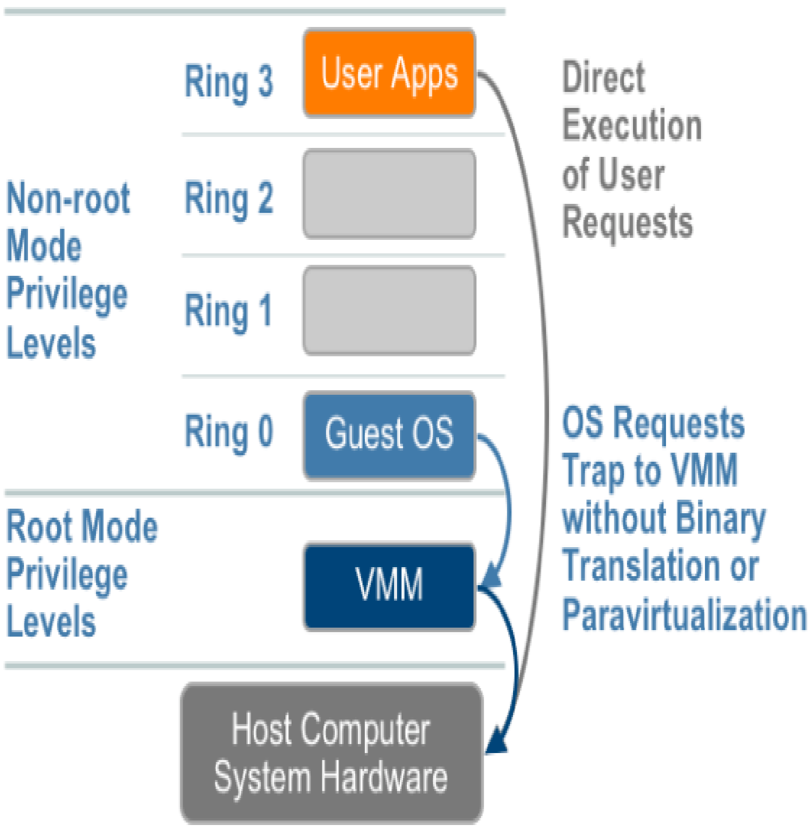


VMX Mode Execution

- Special CPU instructions
  - VMLAUNCH, VMRESUME invoked by KVM to enter VMX mode
  - VMEXIT invoked by guest OS to exit VMX mode

- CPU switches context between host OS to guest OS
- VMCS (VM control structure) is for storing CPU context.
- Page tables (address space), CPU register values etc switched
- Hardware manages the mode switch

Hardware Assisted Virtualization



- Intel Virtualization Technology (VT-x) and AMD's AMD-V which both target privileged instructions with a new CPU execution mode feature that allows the VMM to run in a new root mode below ring 0

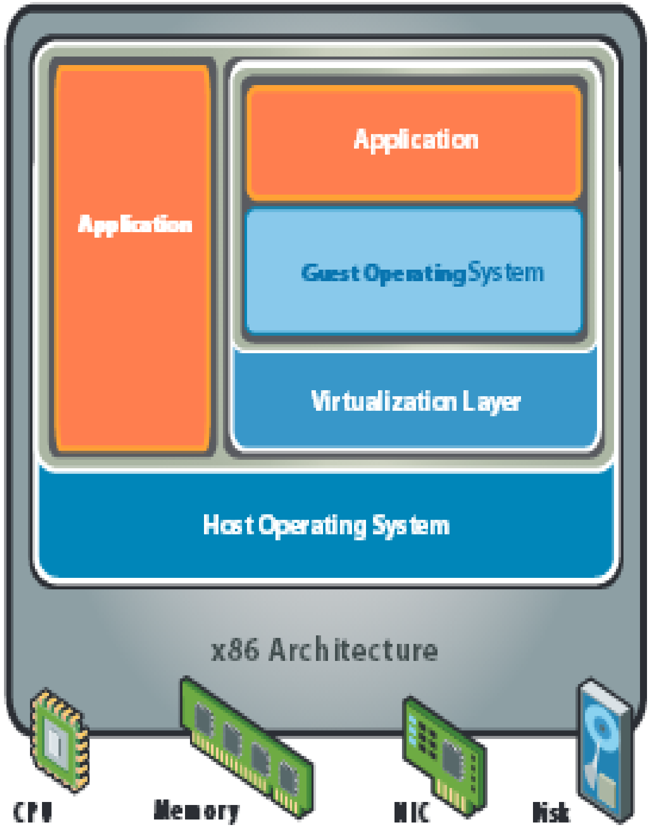
Example: QEMU/KVM in Linux

- Privileged and sensitive calls are set to automatically trap to the hypervisor, removing the need for either binary translation or para virtualization.
- Underlying hardware provides special CPU instructions to aid virtualization

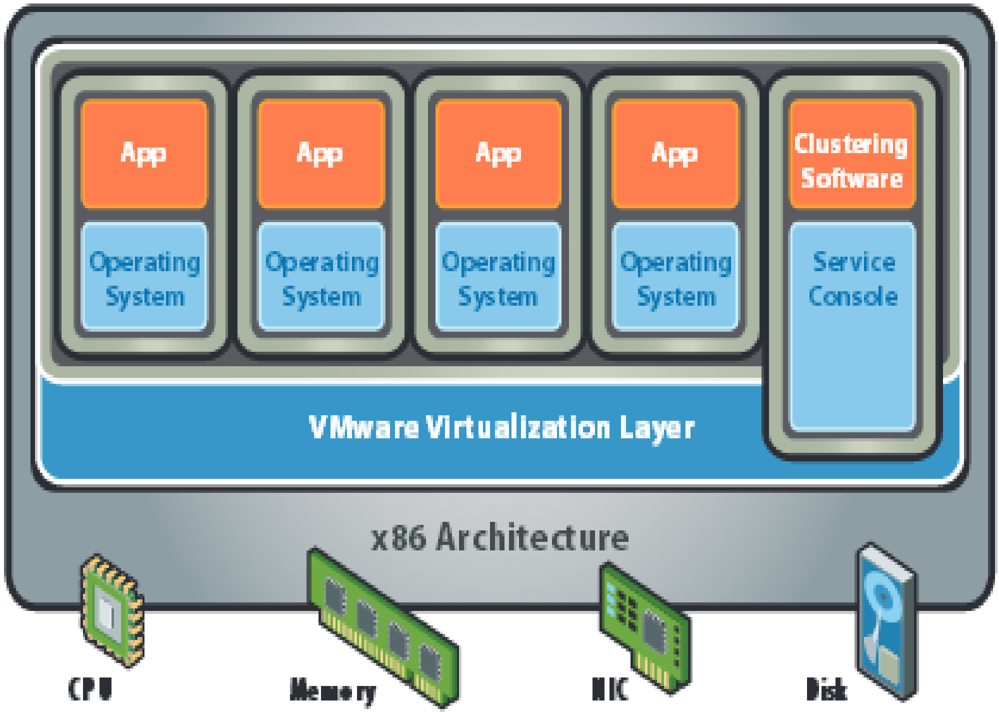
Types of Hypervisors

- For Industry-standard x86 systems, the two approaches typically used with software-based partitioning
  - Bare-metal architectures
  - Hosted
- Bare Metal / Native (Type I)
  - Hypervisor installed on a clean x86-based system.
  - Direct access to the hardware resources, a hypervisor is more efficient than hosted architectures.
  - Enables greater scalability, robustness and performance
  - E.g: VMware ESXi, Citrix XenServer, and Microsoft Hyper-V hypervisor
- Hosted (Type II)
  - A hosted approach provides partitioning services on top of a standard operating system.
  - Supports the broadest range of hardware configurations.
  - E.g: VMware Player or Parallels Desktop

x86 Hardware Virtualization



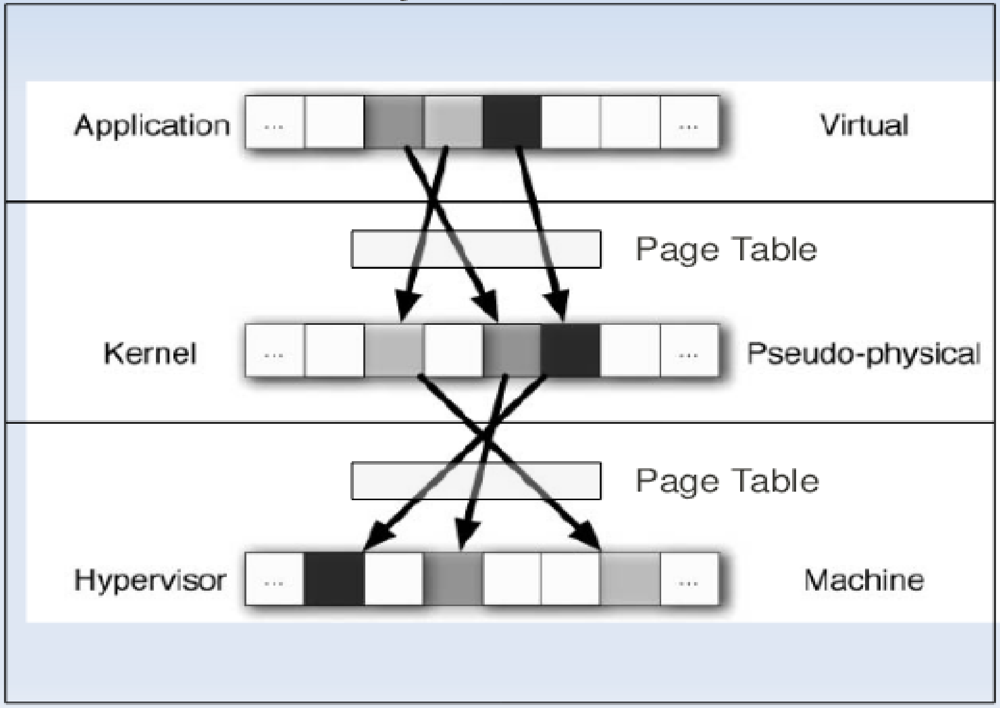
Hosted Architecture



Bare-Metal (Hypervisor) Architecture

Memory Virtualization

- In multiprogramming there is a single level of indirection maintained by Kernel.
- In case of Virtual Machines there is one more level of indirection maintained by VMM

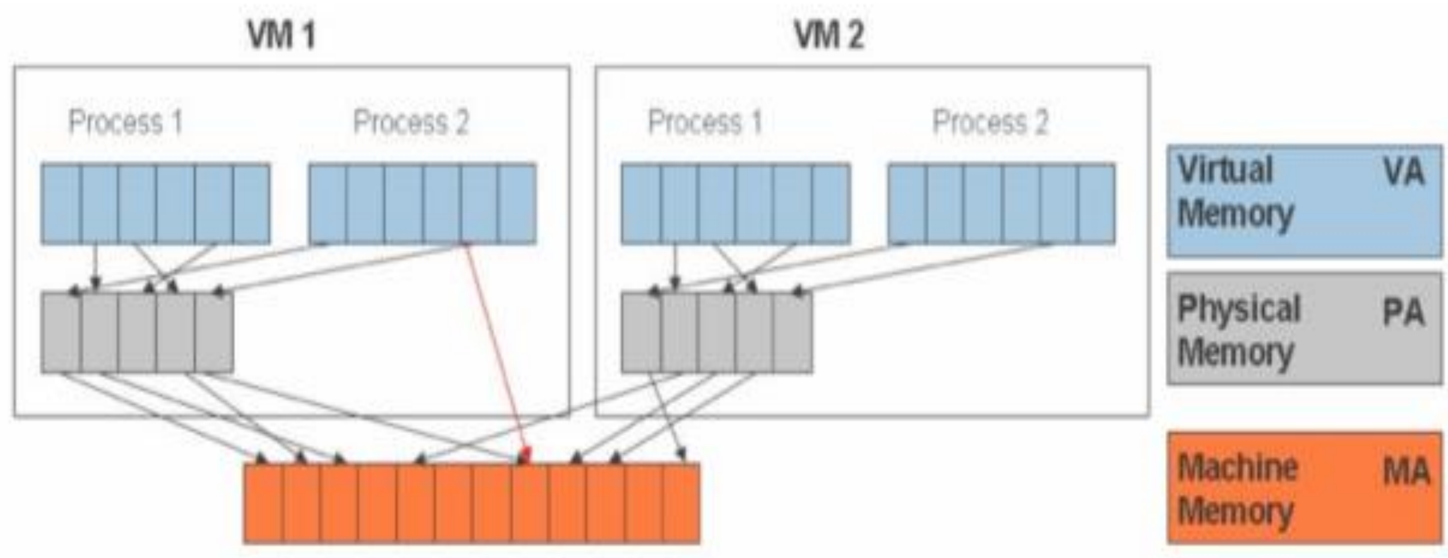


Applications use Virtual Addresses

Kernel translates Virtual Addresses to Pseudo-Physical Addresses

Hypervisor translates Pseudo-Physical Addresses to Machine addresses

Memory Virtualization



GVA: Guest Virtual Address

GPA: Guest Physical Address

HVA: Host Virtual Address

HPA: Host Physical Address

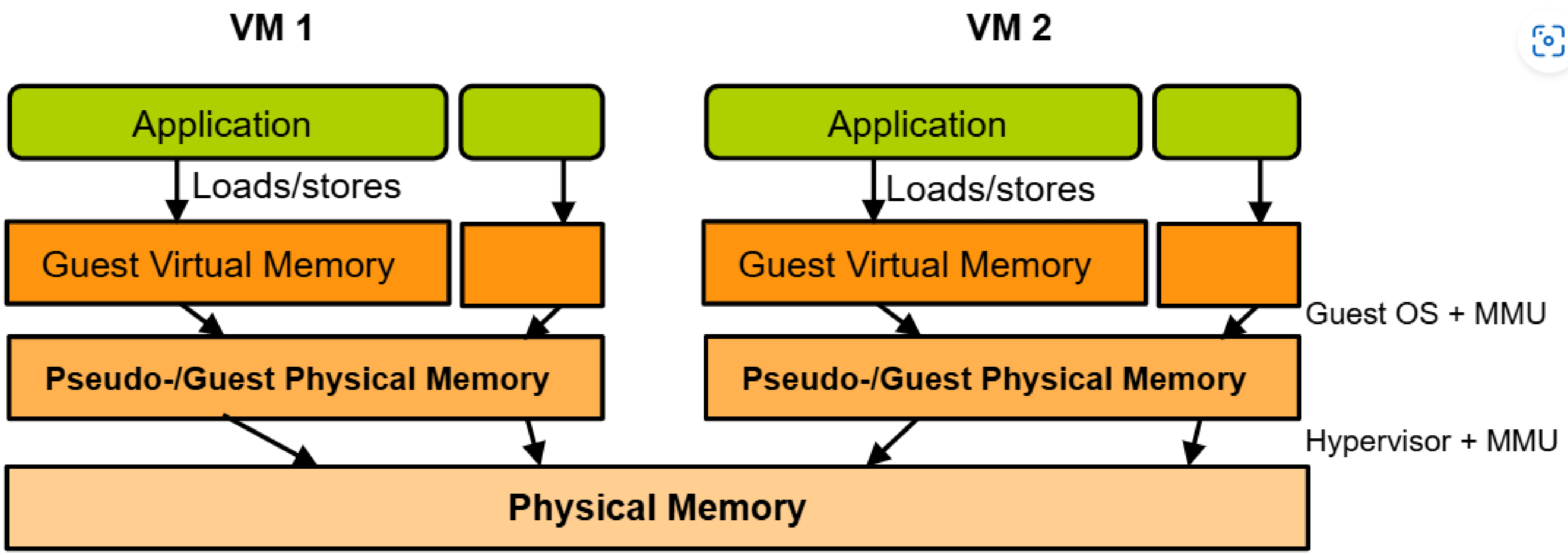
- Guest page table has GVA->GPA mapping
- Each guest OS thinks it has access to all RAM starting at address 0
- VMM / Host OS has GPA->HPA mapping
- Guest “RAM” pages are distributed across host memory

Memory Virtualization Techniques

1. **Shadow paging**
   - VMM creates a combined mapping GVA->HPA and Memory Management Unit (MMU) is given a pointer to this page table
   - VMM tracks changes to guest page table and updates shadow page table
2. **Extended page tables (EPT)**
   1. MMU hardware is aware of virtualization, takes pointers to two separate page tables.
   2. Address translation walks both page tables

- EPT is more efficient but requires hardware support

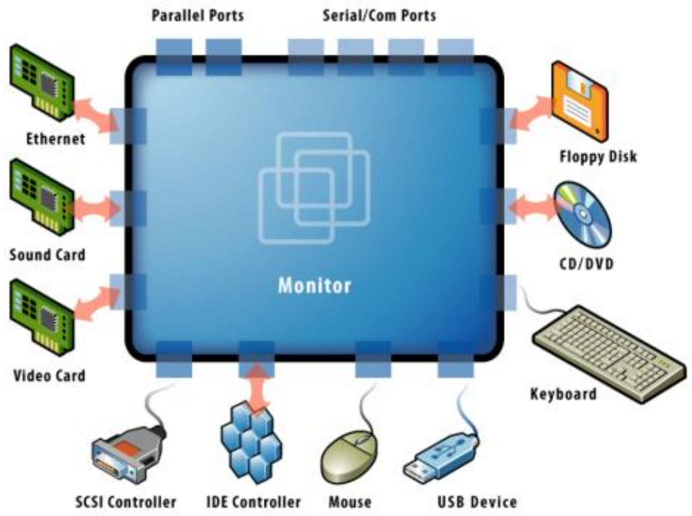
Memory Virtualization



Guest virtual memory → (guest) pseudo-physical memory → (host) physical memory

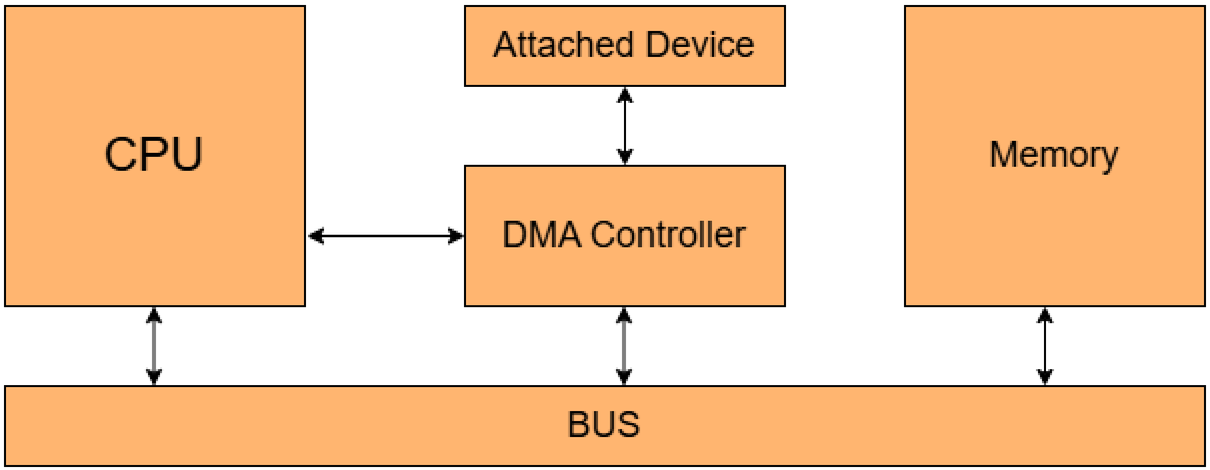
[Virtualization 101 - Introduction to Virtualization (olivierpierre.github.io)](olivierpierre.github.io)

Device and I/O Virtualization



- This involves managing the routing of I/O requests between virtual devices and the shared physical hardware
- Virtual NICs and switches create virtual networks between virtual machines without the network traffic consuming bandwidth on the physical network
- The hypervisor virtualizes the physical hardware and presents each virtual machine with a standardized set of virtual devices
- These virtual devices effectively emulate well-known hardware and translate the virtual machine requests to the system hardware

Direct Memory Access



- Direct Memory Access (DMA) - > Alternative to Polling
- Modern I/O devices perform Direct Memory Access (DMA)
- Copy data from device memory to RAM, Then raise an interrupt
- Device driver provides physical address of DMA buffers to device

Image Courtesy: [where_direct_memory_access_fits_in-f_mobile.png (560×252)]

Device and I/O Virtualization

Device Registers for exposing device's memory

- Command: What to do
- Data: Actual data to be read or to write
- Status: Status of handling the operation – E.g: Data ready

1. **Explicit I/O instructions**:

   To perform read/write to specific registers on/from a device – in/out instructions

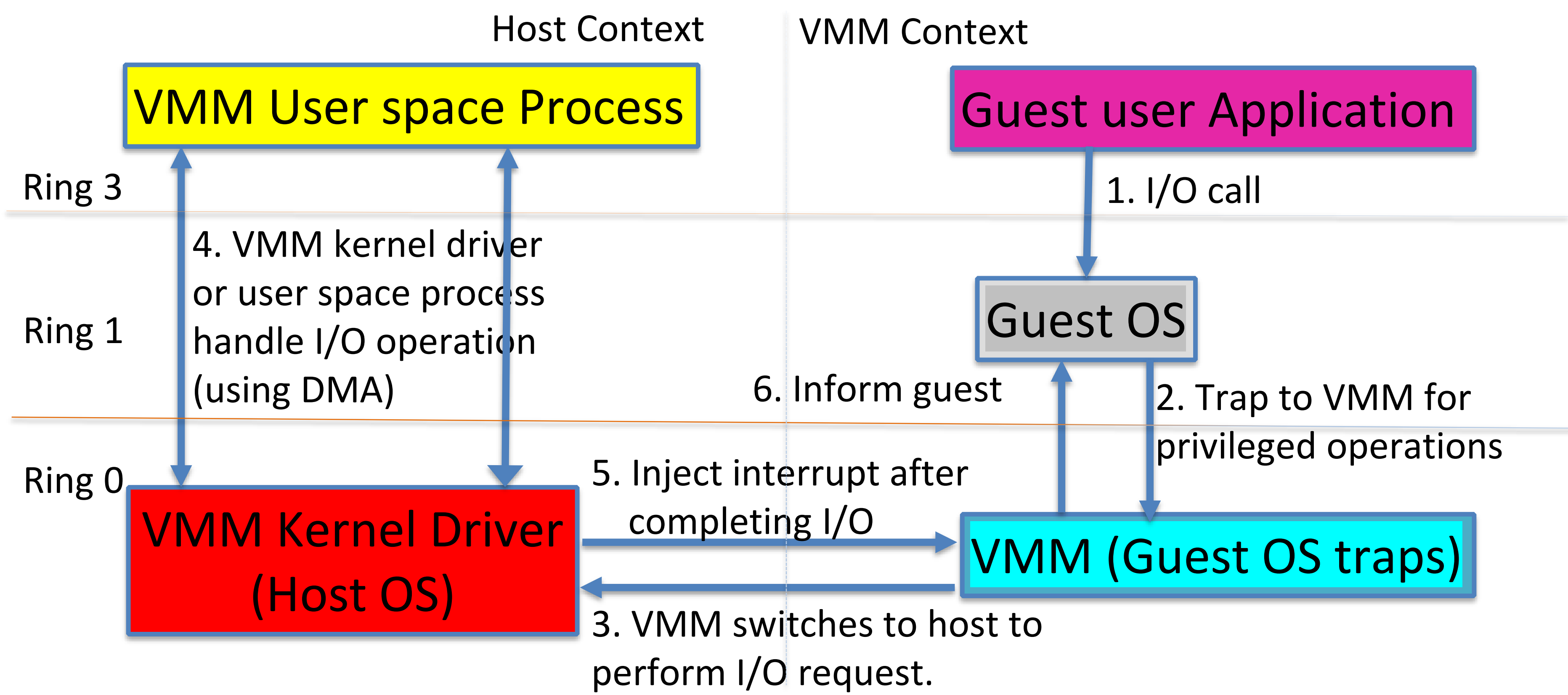
2. **Memory mapped I/O:**

   - Few memory addresses are assigned to device.
   - I/O happens by reading/writing this memory.

Device and I/O Virtualization

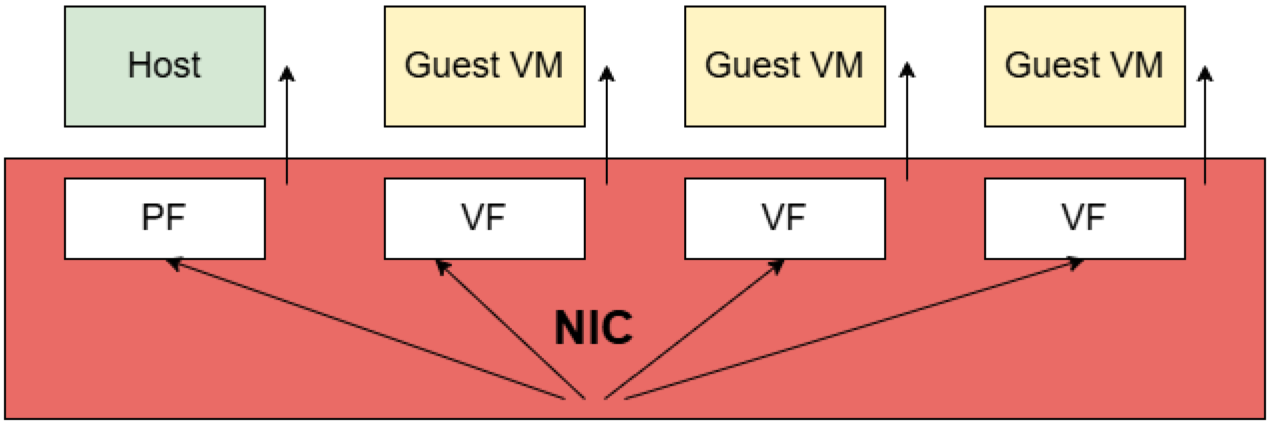
- Guest OS cannot be given full access to I/O devices
- VMM must share I/O device access across guests
- Two ways to virtualize I/O devices:
  - **Emulation:** I/O access from guest traps to VMM. VMM then performs I/O.
  - **Direct I/O or device passthrough**: A portion of device is assigned to guest

Emulated I/O Virtualization (In Full/Para Mode)



"Bringing Virtualization to the x86 Architecture with the Original VMware Workstation", Edouard Bugnion, Scott Devine, Mendel Rosenblum, Jeremy Sugerman, Edward Y. Wang.

Direct I/O Virtualization



- One Network Card
  - Many virtual functions (VFs)
  - One physical function (PF)
- Host OS manages PF
- Every guest VM is assigned a VF
  - Each VF acts like a separate NIC and bound to a guest VM
  - Switching is used to send the packets destined to the MAC address of VM.

Benefits of Virtualization (1)

- Allows most efficient use of the compute resources
- Maximize performance
  - Few apps take advantage of multiple CPUs and huge memory
- Run VMs on minimal # of servers, shutting off the others
  - Automated, live migration critical
  - Provide performance guarantees for dynamic workloads
  - Balance load to minimize number of active servers
- Run-anywhere Capabilities
  - Shared network and storage allows flexible mappings
  - Enables additional availability guarantees

Benefits of Virtualization (2)

- Instant provisioning - fast scalability
- Live Migration is possible
- Load balancing and consolidation in a Data Center is possible.
- Low downtime for maintenance
- Security and fault isolation

Issues to be aware of

- Interoperability among vendor products is still evolving.
- Failure of the virtualization device, leading to loss of the mapping table .
- Hardware Investment
- IT Training
- Software Licensing

Summary

- Virtualization is a key enabler to cloud computing
- Virtualization Types
  - CPU Virtualization
    - Problems with standard Trap & Emulate Virtualization
    - Full: Dynamic binary translation
    - Para: Rewrite guest OS source code
    - Hardware-assisted: CPU has special virtualization mode
  - Memory Virtualization
    - Shadow page tables: Combined GVA->HPA mappings tracked by VMM
    - Extended page tables: Two separate pointers for GVA to GPA and GPA to HPA mappings given to MMU
  - I/O virtualization: emulation, device passthrough
- Benefits of Virtualization

Further Reading

- T1: Related Technologies, T2: Cloud Resource Virtualization
- Bringing Virtualization to the x86 Architecture with the Original VMware Workstation", Edouard Bugnion, Scott Devine, Mendel Rosenblum, Jeremy Sugerman, Edward Y. Wang.
- J.S. Robin, and C.E. Irvine, "Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor"
- M. Rosenblum, and T. Garfinkel, "Virtual Machine Monitors: Current Technology and Future Trends"
- P. Barham, et. al., "Xen and the Art of Virtualization"
- "Xen Developer's Reference"
- Virtualization in Xen 3.0 | Linux Journal
- S. Devine, E. Bugnion, and M. Rosenblum, "Virtualization system including a virtual machine monitor for a computer with a segmented architecture", US Patent 6397242
- L8 – IIT Delhi