Virtualization Overview

Yaozu Dong (Eddie.dong@intel.com)
Kun Tian (kevin.tian@intel.com)

Open Source Technology Center, Intel

Legal Disclaimer

Agenda

- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
  - I/O
- Commercial VMMs
- Summary

What is Virtualization

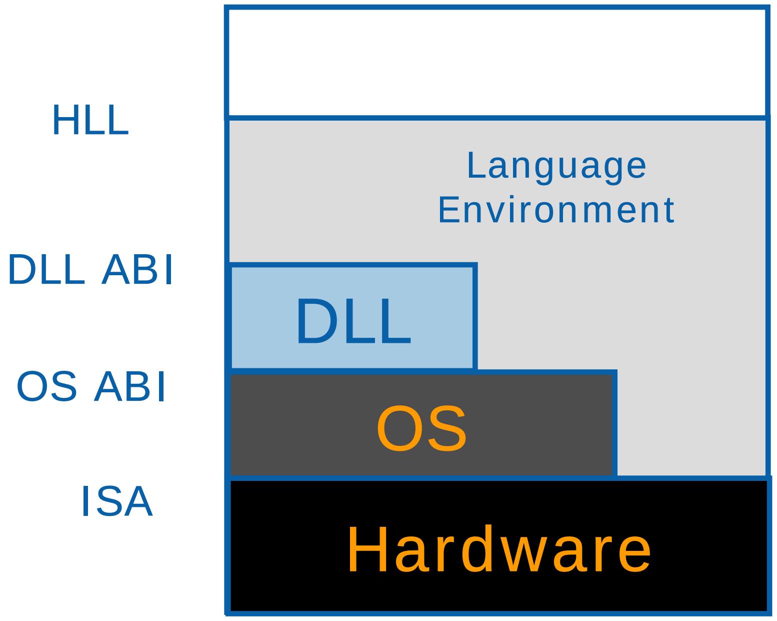
- **All computer problems can be solved with another layer of redirection**

- **Virtualization is a term that refers to the abstraction of computer resources (wikipedia)**
  - System Virtualization, or Platform Virtualization
    - Redirection in platform layer
  - Process Virtual Machines, or Application Virtual Machines
    - Redirection in application ABI layer
    - Multiprogramming
    - High Level Language (HLL)

  - (Refer to wikipedia and JE smith paper here …)

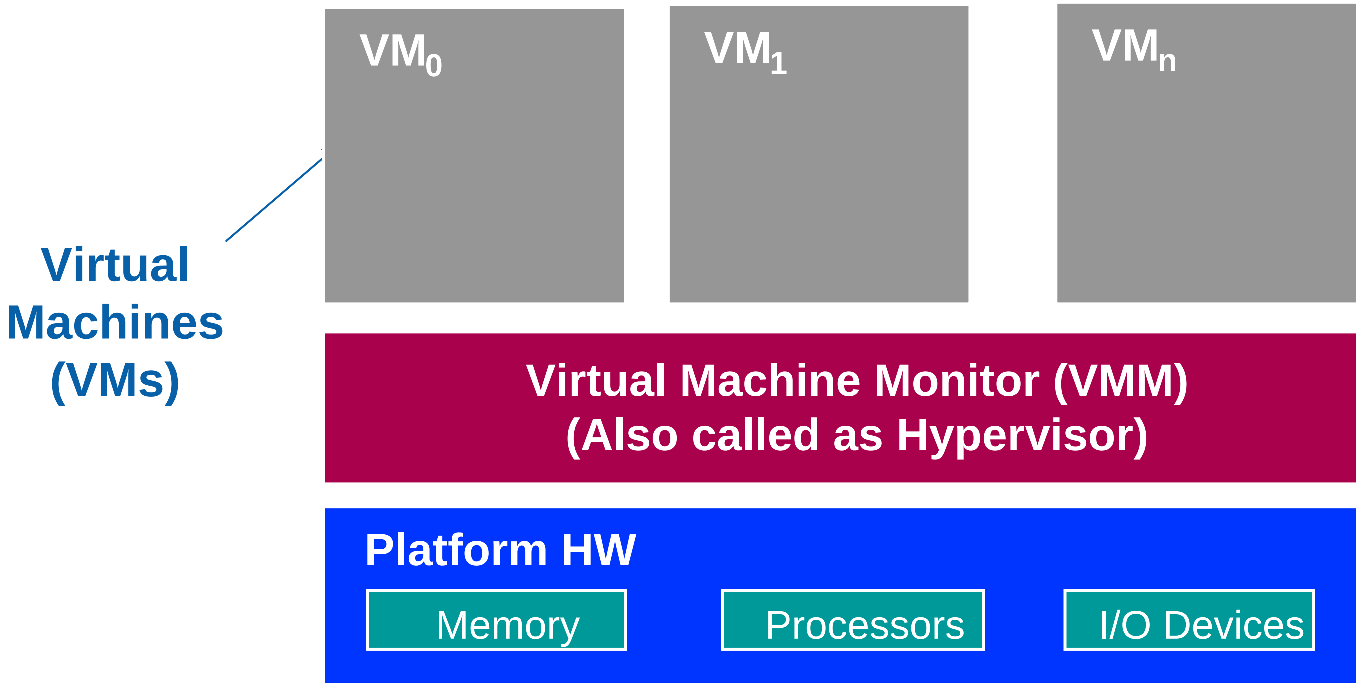
Different Layer of Virtualization

- **Instruction Set Architecture, or ISA**
  - System Virtualization
- **OS Application Binary Interface, or OS ABI**
- **Dynamic Link Library ABI, or DLL ABI**
- **High Level Language, or HLL**
  - Java, C etc.



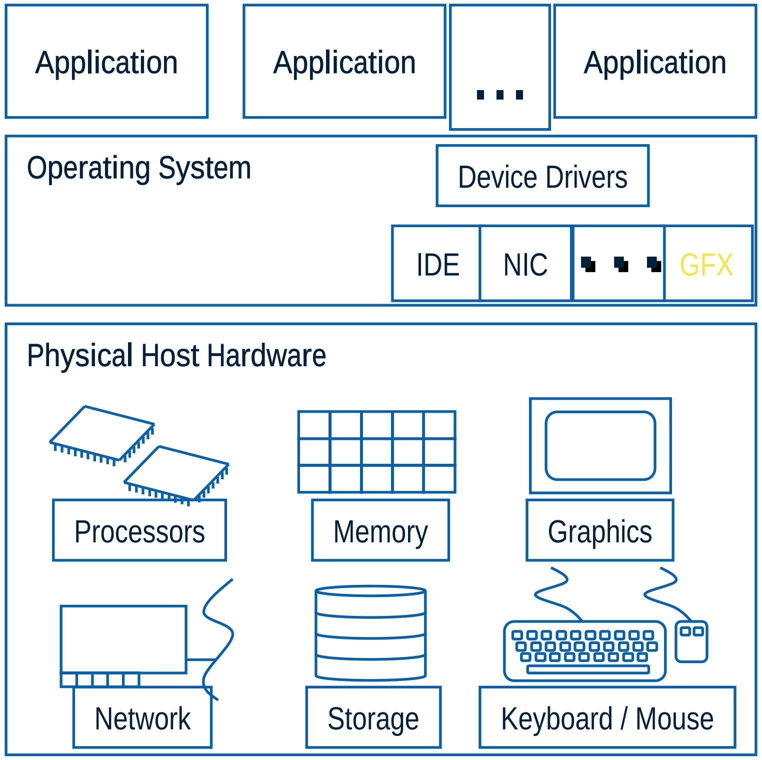
This talk focuses on system virtualization.

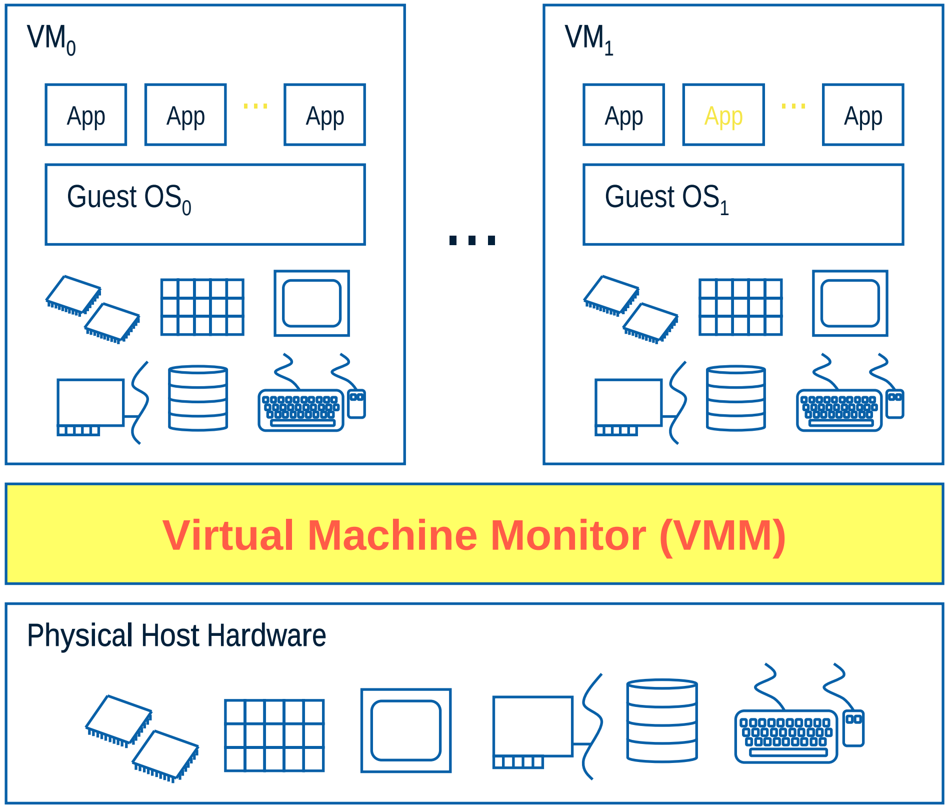
Virtual Machine Monitor



- **VMM transforms the single machine interface into the illusion of many. Each of these interfaces (virtual machines) is an efficient replica of the original computer system, complete with all of the processor instructions**
  - Robert P. Goldberg.

What does a VM look like





Without VMs: Single OS owns all hardware resources

With VMs: Multiple OSes share hardware resources

- **A virtual machine is implemented by adding software to an execution platform to give it the appearance of a different platform, or for that matter, to give the appearance of multiple platforms.**
  - J. E. Smith and Ravi Nair, “An Overview of Virtual Machine Architectures”.

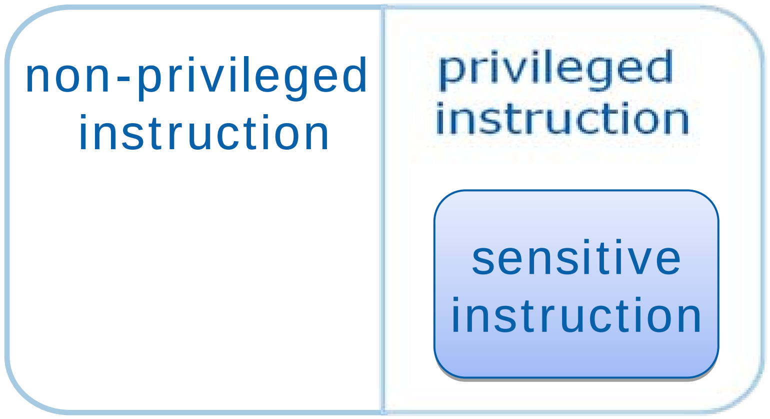
Essential characteristics of VMM

- **Equivalence**
  - Essentially identical virtual platform, except
    - Differences caused by the availability of system resources
      - E.g. amount of memory available to VM
    - Differences caused by timing dependencies.
- **Isolation, or resource control**
  - VMM is in complete control of system resources
    - VM can't access any resources not explicitly allocated to it
    - VMM may regain control of resources already allocated
- **Efficiency**
  - At worst only minor decreases in speed
  - Rules out traditional emulators and complete software interpreters (simulators)

*Source from Gerald J. Popek & Robert P. Goldberg, "Formal Requirements for Virtualizable Third Generation Architectures"

Virtualizable ISA

- **The set of sensitive instructions for that computer is a subset of the set of privileged instructions**
  - Privileged instructions
    - Those that trap if the processor is in user mode and do not trap if it is in system mode.
  - Sensitive instructions
    - Control sensitive instructions
      - Those that attempt to change the configuration of resources in the system.
    - 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).



*Source from Gerald J. Popek & Robert P. Goldberg, “Formal Requirements for Virtualizable Third Generation Architectures”

System Virtualization Approaches

- **Operating system-level virtualization**
  - An operating system allows for multiple isolated user-space instances
  - Often called containers
- **Full virtualization**
  - Hardware-assisted virtualization
  - Binary Translation etc.
- **Partial virtualization**
  - Most but not all of the hardware features are simulated in VM (such as IBM M44/44X)
- **Paravirtualization**
  - Modify guest OS to cooperatively work with hypervisor to close virtualization hole or for performance.

Agenda

- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
  - I/O
- Commercial VMMs
- Summary

VMM Requirements

- **A VMM must be able to:**
  - Protect itself from guest software
  - Isolate guest software stacks (OS + Apps) from one another
  - Present a (virtual) platform interface to guest software

- **To achieve this, VMM must be able to control:**
  - CPU(s)
  - Memory
  - I/O devices

Virtualization holes

- **Conventional Intel® 64, or IA-32, is not virtualizable architecture**
  - Sensitive register instructions: read or change sensitive registers and/or memory locations such as a clock register or interrupt registers
    - SGDT, SIDT, SLDT
    - SMSW
    - PUSHF, POPF
  - Protection system instructions: reference the storage protection system, memory or address relocation system
    - LAR, LSL, VERR, VERW
    - POP
    - PUSH
    - CALL, JMP, INT n, RET
    - STR
    - MOVE

*Source from proceeding of 2000 USENIX ATC

Controlling the CPU Resource

- **With an Intel® 64 CPU, a VMM must be able to retain control over:**
  - Access to privileged state (CRn, DRn, MSRs)
  - Exceptions (#PF, #MC, etc.)
  - Interrupts and interrupt masking
  - Address translation (via page tables)
  - CPU access to I/O (via I/O ports or MMIO)
- **Intel® 64 allows control of these resources**
  - But at some cost …

CPU Control via “Ring Deprivileging”

- **Definition of ring deprivileging:**
  - Guest OS kernel runs in a less privileged ring than usual (i.e., above ring 0)
    - For example, Guest OS kernel is in ring 1 and application is in ring 3.
  - VMM runs in the most privileged ring 0

- **Goal of ring deprivileging: prevent guest OS from**
  - Accessing privileged instructions / state
  - Modifying VMM code and data

Problems with Ring Deprivileging

- **Ring deprivileging encounters various issues in current Intel® 64 architecture:**
  - Ring compression
  - Ring aliasing
  - CPU-state context switching
  - Non-faulting reads of privileged state
  - Excessive faulting
  - Interrupt-virtualization issues

Addressing Intel® 64 "Virtualization Holes"

- **Paravirtualization**
  - Modify guest OS to cooperatively work with hypervisor
    - Fix virtualization hole with source modification to guest OS
    - Linux and Windows 8
- **Binary translation (or patching)**
  - Modify guest OS binaries "on-the-fly" with translation cache
    - Complicated
    - New complexities. E.g., consider self-modifying code, interrupt etc.
  - Certain forms of excessive trapping remain
- **Hardware-assisted virtualization**
  - Closing virtualization holes in hardware
  - Simplify VMM software
  - Optimizing for performance

Intel® Virtualization Technology

- **A hardware-assisted virtualization technology, named as Intel® Virtualization Technology, or Intel® VT**
  - For Intel® Itanium®, VT-I
  - For Intel® 64, VT-x
  - For directed I/O, VT-d
  - For connectivity, VT-c
- **VT-x: A new form of Intel® 64 CPU operation**
  - Provides mechanisms for VMM software control of Intel® 64 CPUs
  - Includes “hooks” necessary for software control of memory and I/O resources
- **VT-d: An extension of chipset technology for directed I/O**
  - DMA remapping
  - Interrupt remapping etc.
- **VT-c: A collection of I/O virtualization technologies**
  - Lower CPU utilization
  - Reduced system latency
  - Improved throughput

Agenda

- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
  - I/O
- Commercial VMMs
- Summary

Virtual Platform

- **Simulates a hardware platform and all its software platforms.**
  - Hardware platform
    - CPU
    - Memory
    - I/O
  - Software platforms/components layer
    - BIOS
    - OS
    - Run time library
    - Applications

Agenda

- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
  - I/O
- Commercial VMMs
- Summary

Binary Translation (BT)

- **Translate guest OS binary on the fly to close virtualization hole**
  - OS binary is still visible to guest, but the executed guest code is actually in translation cache.
  - Need to steal guest address space to hold the translated cache.
    - Depend on guest OS
  - Be able to run unmodified guest OS
- **Dynamic patching is a special case of BT**
  - Normally the patched code is in original place
    - Same length, or NOP padded
  - Mangle guest visible code
    - Problem to memory guard such as that in Vista 64
- **Static patching is explored in architectures such as Itanium**
  - Modify guest binary offline
  - Replace in original place as well

Paravirtualization (PV)

- **Modify guest OS source to cooperatively work with hypervisor for performance, simplicity etc.**
  - Hypercall to request for hypervisor service
    - Close virtualization hole
    - Share the global resource
  - Event channel to receive asynchronous notification from hypervisor
  - Share memory for massive information communication
- **Widely used in device driver by commercial VMMs**
  - VMware
  - Xen/KVM
  - Hyper-V

Hardware-assisted virtualization

- **Hardware extension to close virtualization hole, e.g. Intel® Virtualization Technology (Intel® VT)**
  - Guest OS runs in de-privileged mode
  - Guest access to privilege resources triggers exit from VM to VMM
- **Be able to run unmodified guest OS**

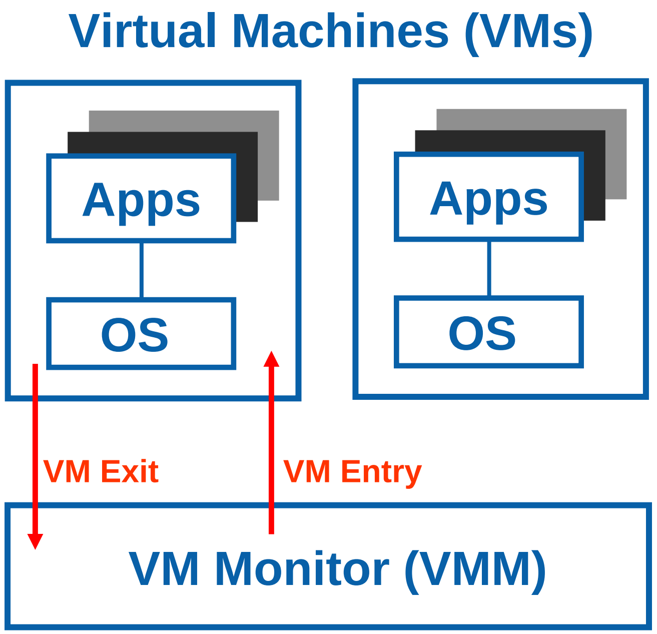
Intel® VT for Intel® 64 (VT-x)

- **Virtual Machine Extensions--VMX**
  - VMX root operation
    - Full privilege levels (ring 0-3)
    - Fully privileged, intended for VMM
    - Entered on exits from guest software
  - VMX non-root operation
    - Full privilege levels (ring 0-3)
    - Not fully privileged, intended for guest
    - Explicitly entered by VMM (new instructions)

In general, a VMM will run in VMX root operation and guest software will run in VMX non-root operation.

VMX Transitions: Overview

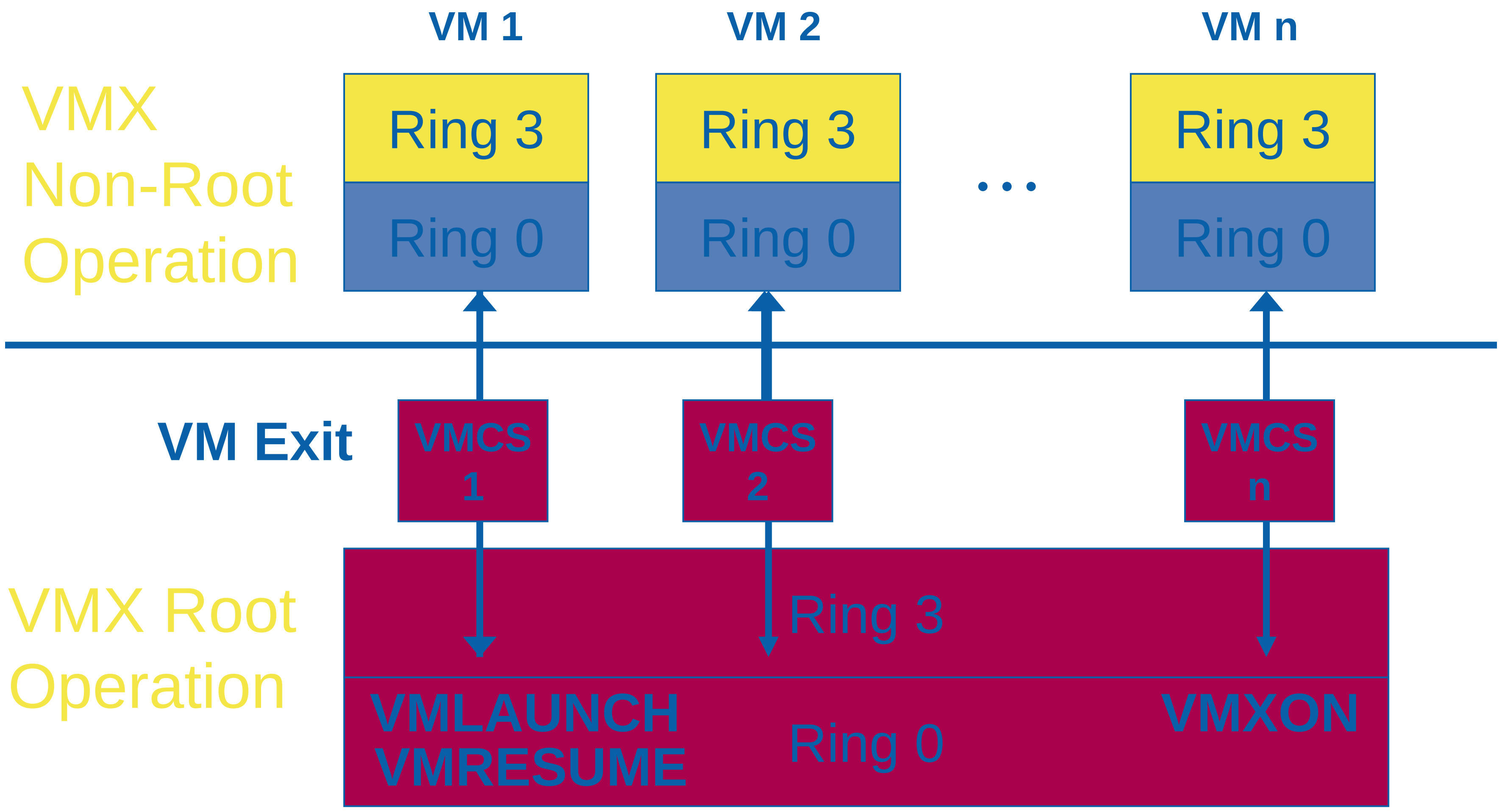
- **VM entry**
  - Transition from VMM to guest
  - Enters VMX non-root operation
  - VMLAUNCH used for initial entry
  - VMRESUME used subsequently
- **VM exit**
  - Guest-to-VMM transition
  - Enters VMX root operation
  - Caused by external events, exceptions, some instructions



Virtual Machine Control Structure (VMCS)

- **Controls VMX non-root operation, VMX transitions**
  - Each virtual CPU should have its own VMCS
  - Only one VMCS active at a time on a physical CPU
- **Each VMCS stored in a memory region**
  - Physical address set by new VMPTRLD instruction
  - Need not reside in linear-address space (unlike IDT)
  - VMPTRLD also used to switch VMCS's

VT-x Operation



Enlighten OS for virtualization

- **A trend of OS development to optimize for virtualization**
  - Os is designed as virtualization aware
    - Windows 7 is hyper-V enlightened
    - Linux pv_ops provides hook APIs for VMMs to optimize
- **To run PV OS in hardware virtual machine (HVM) container**
  - Take both advantage of paravirtualization and hardware-assist virtualization
  - An example is 64 bits XenLinux
    - 64 bits XenLinux kernel runs in ring3 due to lack of segmentation protection
    - Page table switch, TLB flush and hypervisor involvement is a must for system call in 64 bits XenLinux

Agenda

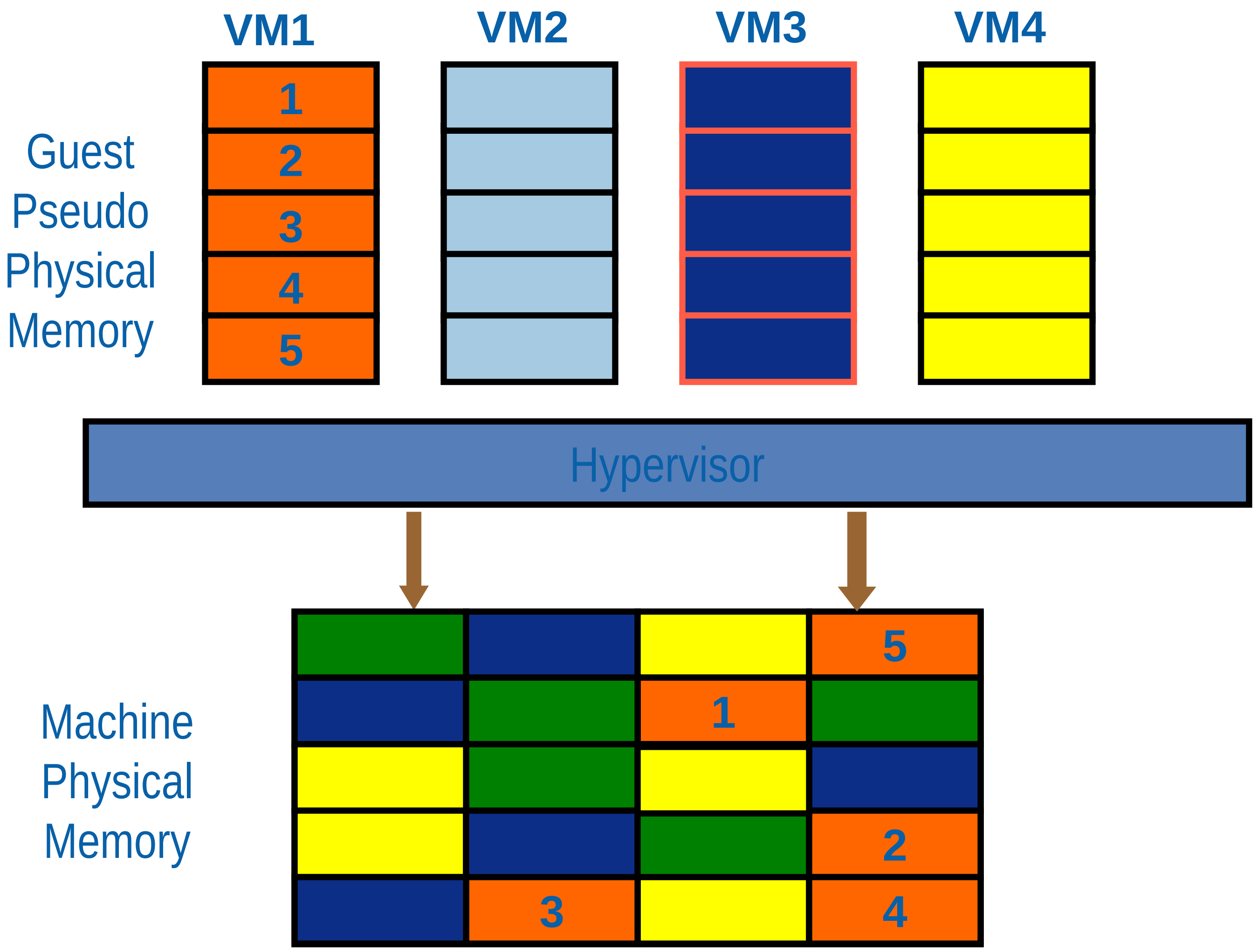
- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
    - Physical page frame virtualization
    - MMU virtualization
  - I/O
- Commercial VMMs
- Summary

Memory virtualization challenges

- **OS expect to see physical memory starting from 0**
  - BIOS/Legacy OS are designed to boot from address low 1M
- **OS expect to see contiguous memory in address space**
  - OS kernel requires minimal contiguous low memory
    - OS, such as Linux, requires minimal contiguous low memory
  - OS components may require certain level of contiguous memory
    - For example, DMA pages, identical mapped kernel data structure…
  - Efficient page frame management
    - Less management overhead
  - Efficient TLB
    - Super page TLB

VMM has to remap guest physical address to host (or machine) physical address

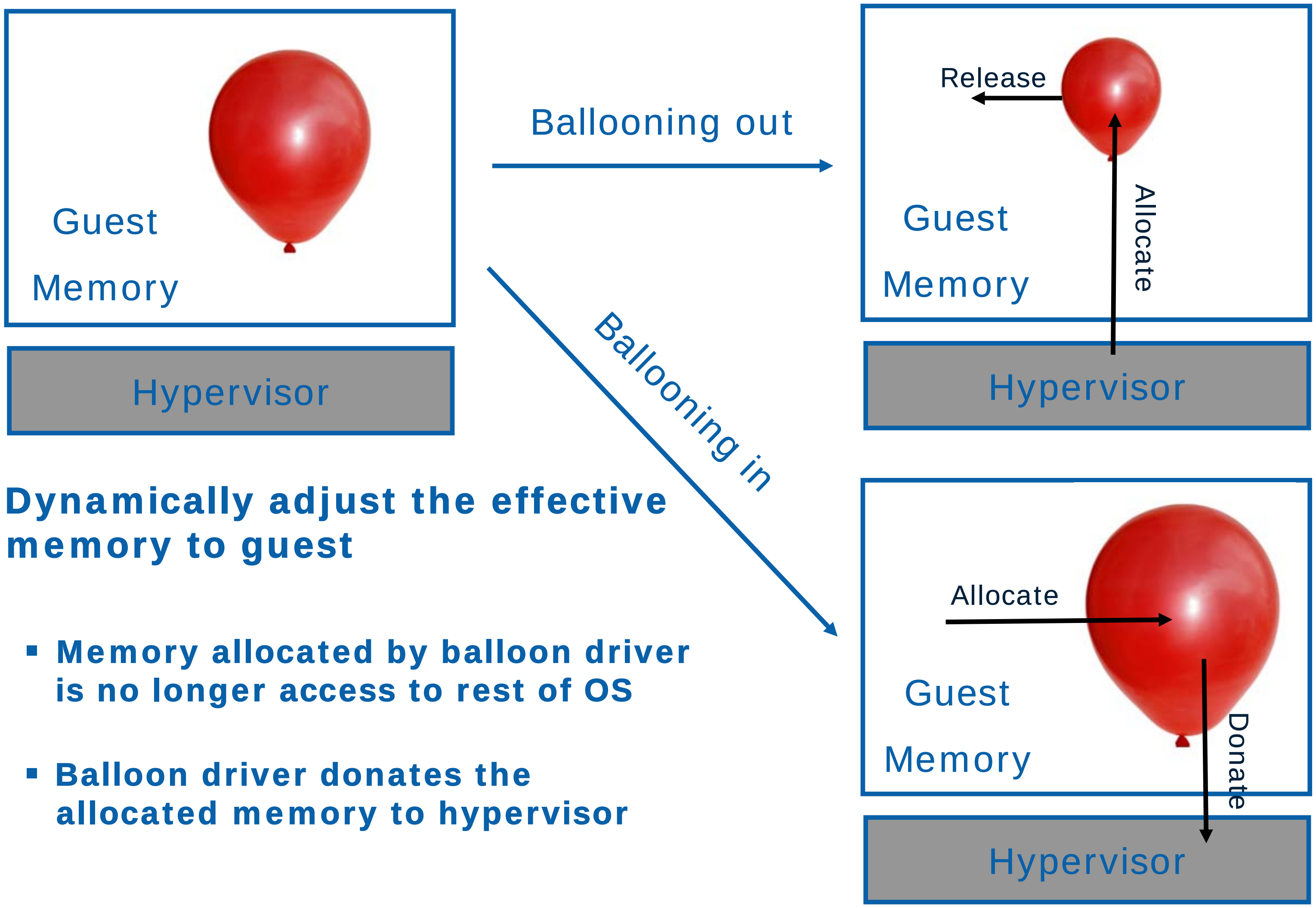
Physical page frame redirection



Page frame allocation

- **Partitioning**
  - Simple and high performance
  - $ to buy more memory
- **Contents based sharing**
  - Pages with same contents can share the physical frame
    - Such as zero page, guest code pages etc.
- **Ballooning**
  - A VMM-aware balloon driver running in guest OS to dynamically allocate memory from OS and release them to VMM, and vice versa
- **Host swapping**
  - The physical frames for guest pages may be swapped out

Ballooning



- **Dynamically adjust the effective memory to guest**
  - **Memory allocated by balloon driver is no longer access to rest of OS**
  - **Balloon driver donates the allocated memory to hypervisor**

Agenda

- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
    - Physical page frame virtualization
    - MMU virtualization
  - I/O
- Commercial VMMs
- Summary

MMU Virtualization

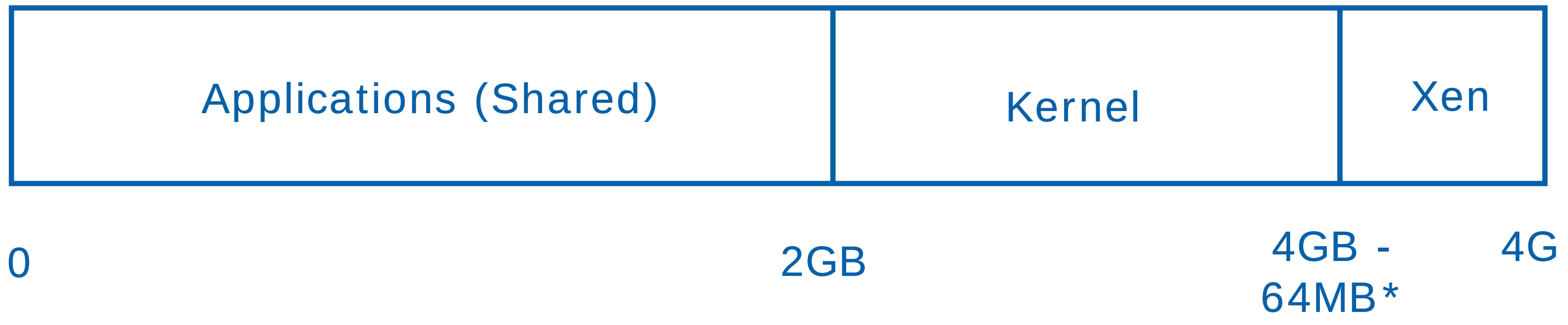
- **Challenges**
  - Guest TLB and guest page table is no longer valid to hardware
    - Guest physical address is remapped

- **MMU Virtualization**
  - Direct page table
  - Virtual TLB
  - Shadow page table
  - Extended page table

Memory remap (gpa → hpa) requires MMU virtualization

Direct page table

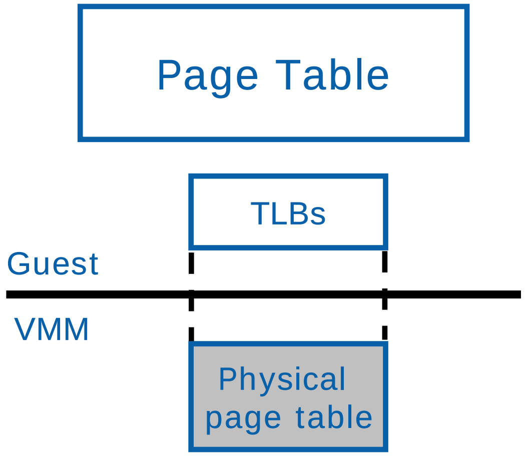
- **Guest and hypervisor share same page table**
  - Partition address space with segmentation protection
    - Isolate guest from access to hypervisor
  - Avoid page table switch between hypervisor and guest
  - Improve TLB efficiency
  - But, hypervisor has to check guest modification of page table
    - Avoid guest map for hypervisor pages
    - Avoid ruin of hypervisor translations
- **Example of XenLinux32 address space**



*The size of Xen address space varies version to version

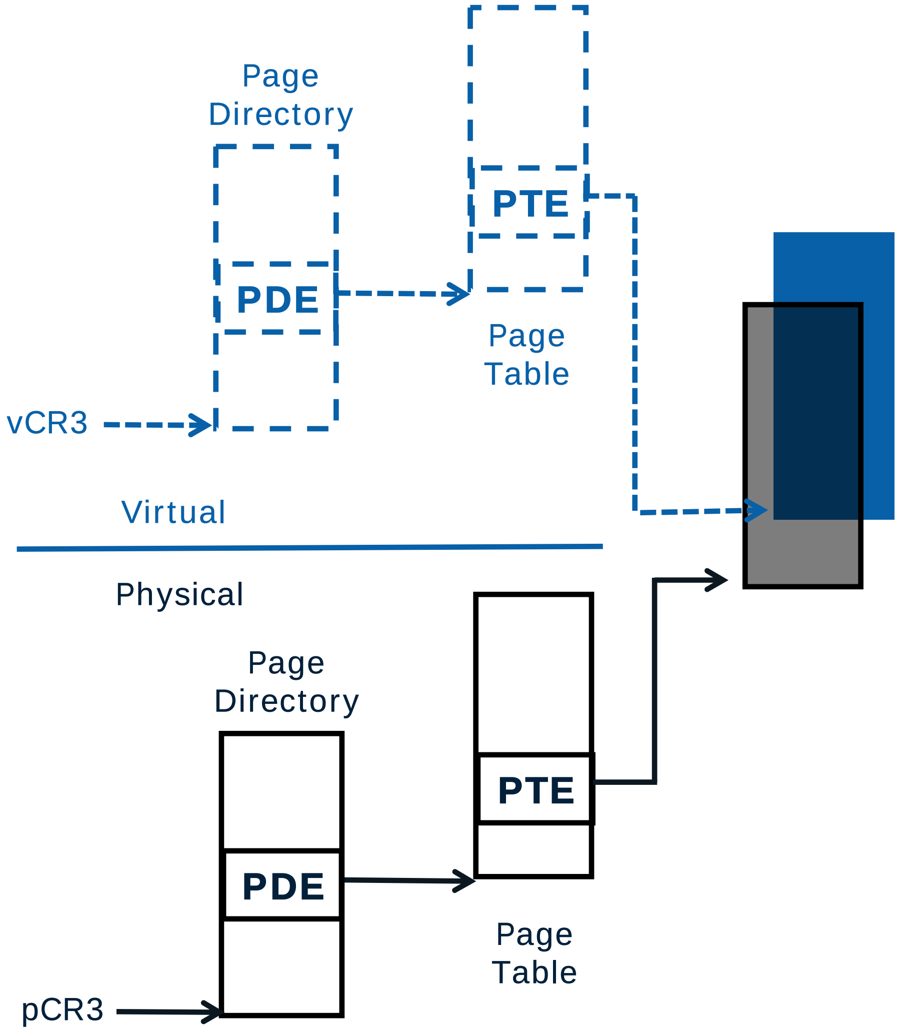
Virtual TLB

- **Hypervisor emulates a complete set of guest TLBs**
  - TLB operation in Intel® 64 is limited to:
    - INVPLG, move to CR3 (CR0/CR4), page walk on TLB miss
  - A physical page table to represent virtual TLBs
    - Insert by walking guest page table on host page fault
    - Remove when INVPLG, move to CR3 (CR0/CR4)
  - Simple, but not optimal
- **How big should virtual TLB be?**

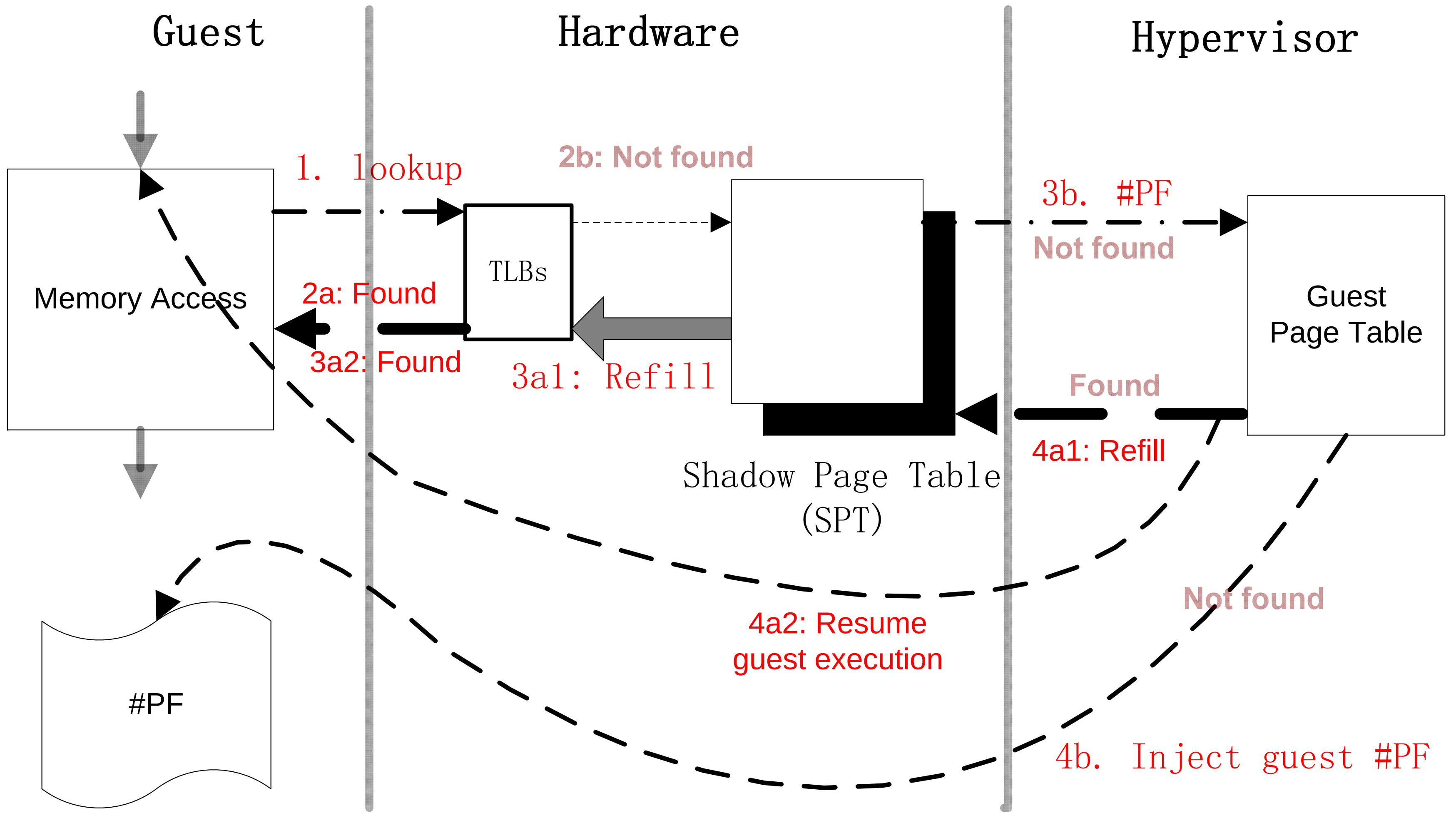


Shadow page table

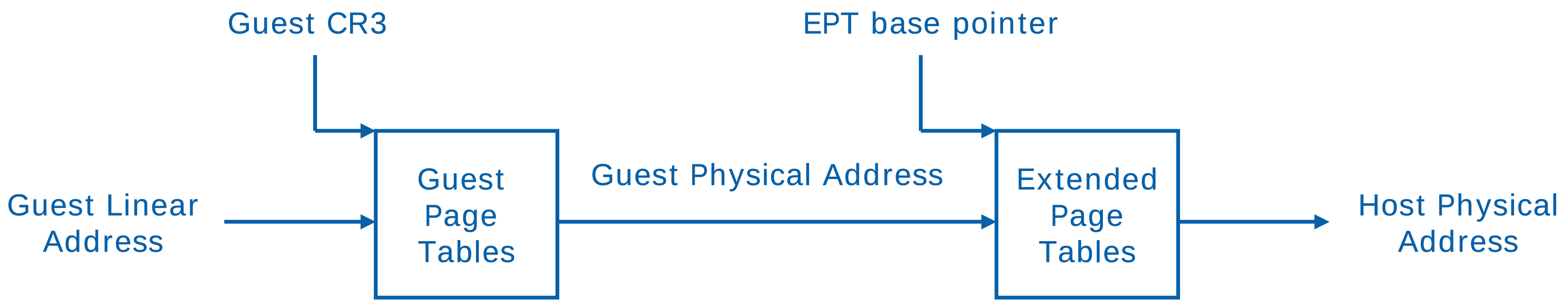
- **Guest page table remains unmodified to guest**
  - Translate from gva -> gpa
- **Hypervisor create a new page table for physical**
  - Use hpa in PDE/PTE
  - Translate from gva -> hpa
  - Invisible to guest
- **A special case of virtual TLB when the guest TLB size is unlimited**



Guest memory access under vTLB/shadow



Extended Page Table (EPT)

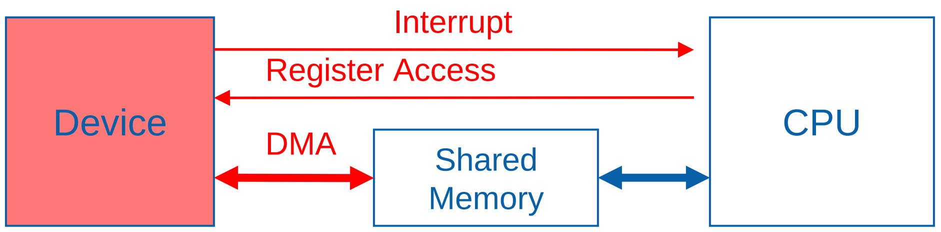


- **Guest can have full control over its page tables and events**
  - CR3, INVLPG, page fault
- **VMM controls Extended Page Tables**
  - Complicated shadow page table is eliminated
  - Improved scalability for SMP guest

Agenda

- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
  - I/O
- Commercial VMMs
- Summary

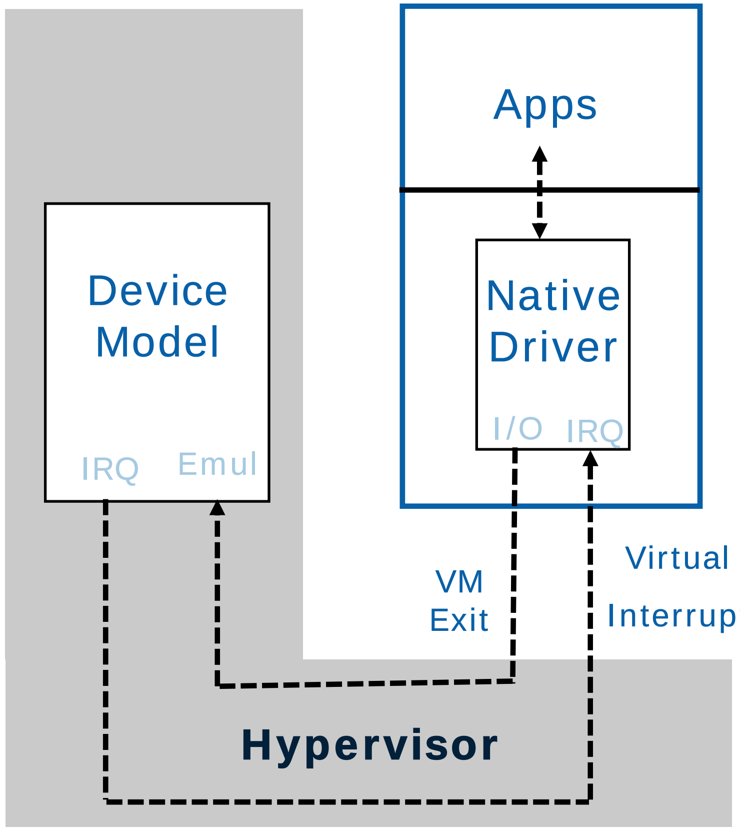
Device interface



- **Interaction between device and driver:**
  - Driver programs device through register access
  - Device notifies driver through interrupt
  - Device could DMA for massive data movement
- **I/O Virtualization requires the hypervisor to present guest a complete device interface**
  - Presenting an existing interface
    - Software Emulation
    - Direct assignment
  - Presenting a brand new interface
    - Paravirtualization

Software Emulation

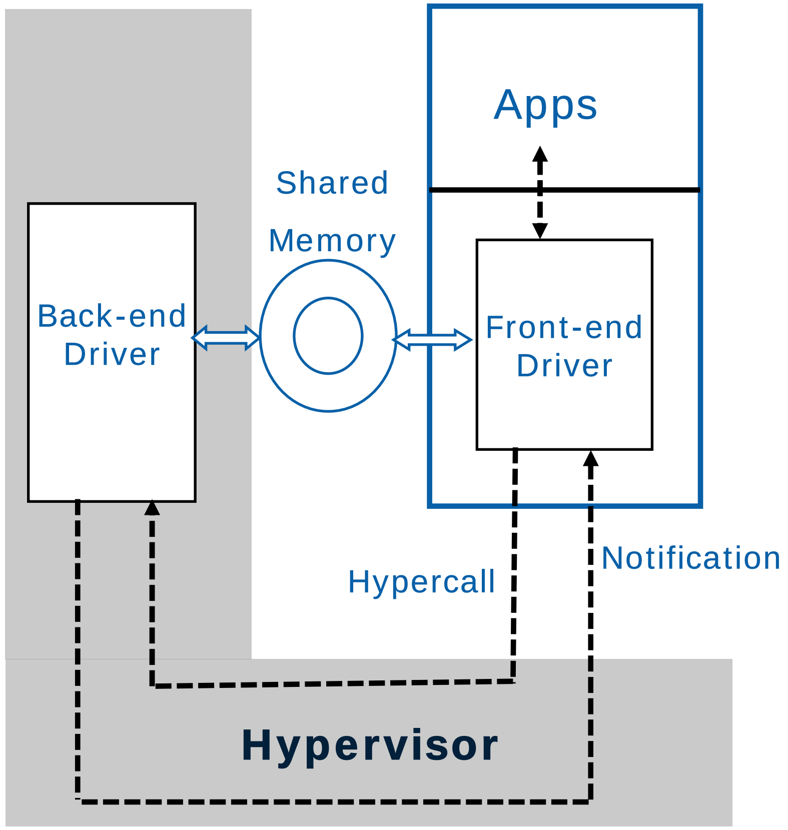
- **Guest runs native device driver, e.g. NE2000**
  - I/O access is trap-and-emulated by device model in hypervisor
    - Translation for MMIO is zapped
  - Virtual Interrupt is signaled by device model per semantics
- **Excessive trap and emulation**



Excessive hypervisor intervention for performance data movement

Paravirtualization

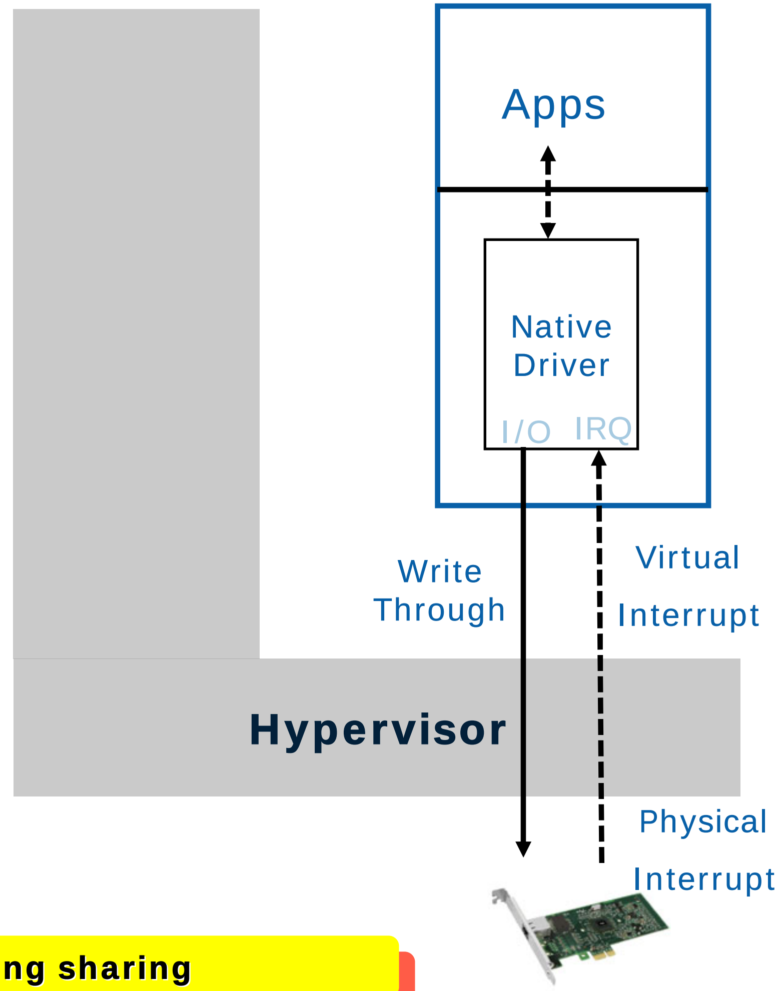
- **A new front-end driver (FE driver) is run in guest**
  - Optimized request through hypercall
- **Hypervisor runs a back-end driver (BE driver) to service request from FE driver**
  - Notify guest for processing
- **Shared memory is used for massive data communication**
  - To reduce guest/hypervisor boundary crossing
  - E.g. Xen VNIF, KVM Virtio-Net



Runtime Hypervisor intervention is largely reduced

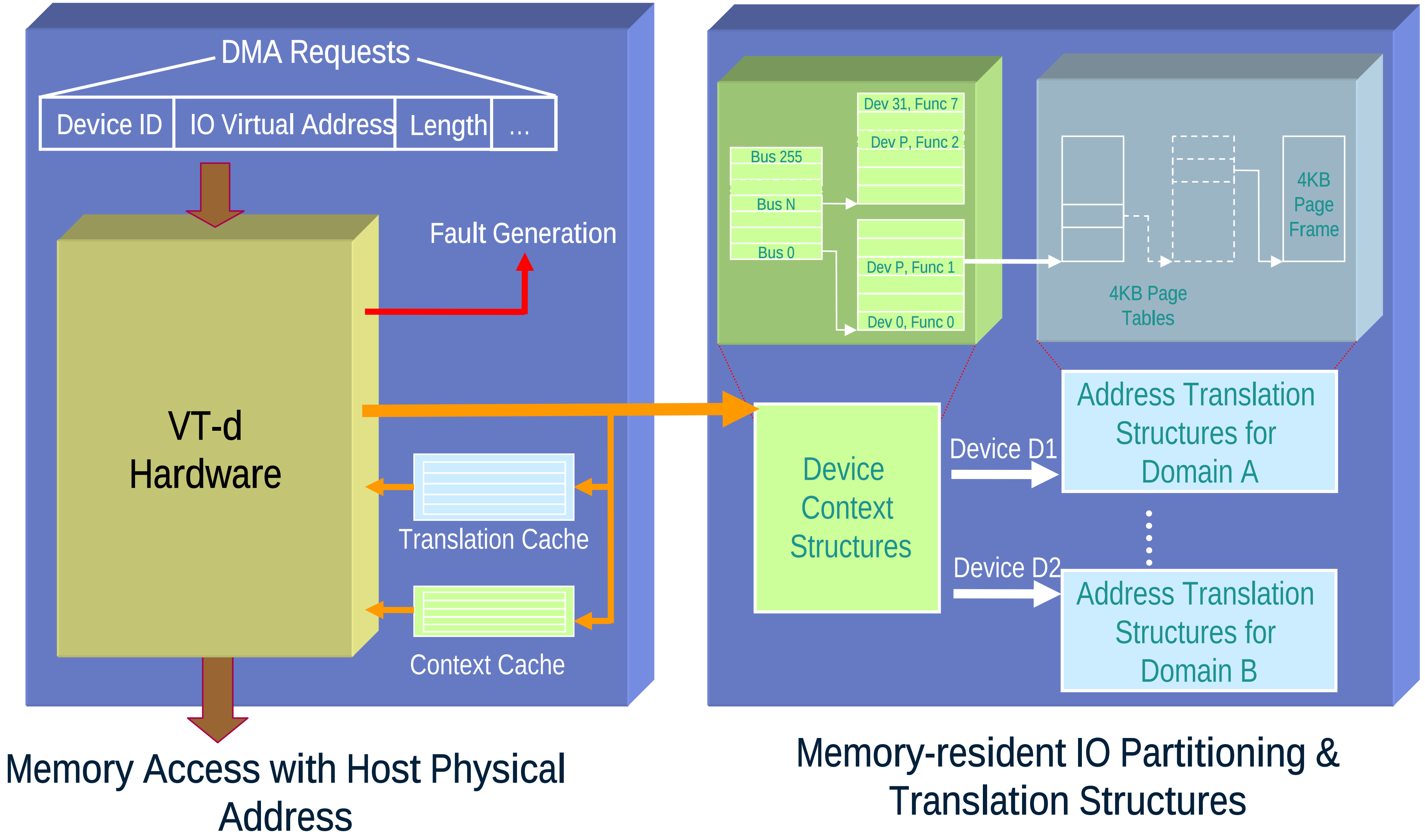
Direct assignment

- **Guest runs native driver**
- **I/O is written through**
  - Translation for MMIO is presented
- **Interrupt**
  - Physical interrupt is captured by hypervisor (pIRQ)
  - Virtual interrupt is signaled for guest (vIRQ)
  - Remapping from pIRQ → vIRQ in hypervisor
- **How about device DMA ?**
  - DMA address programmed by device driver (write-through I/O) is no longer real physical address

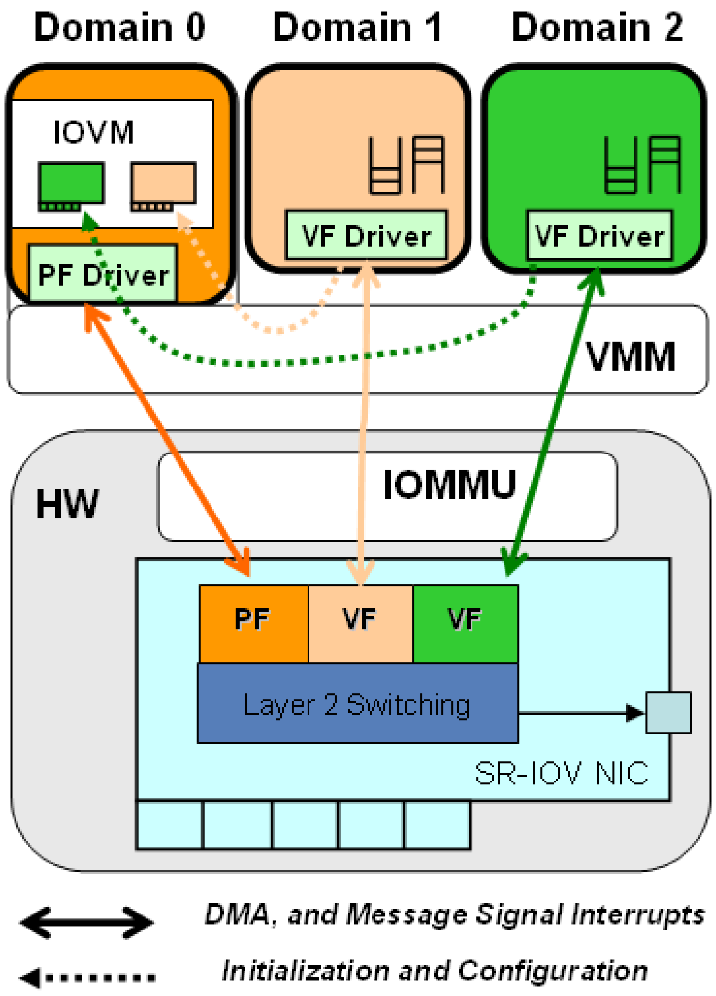


Maximizing performance, but sacrificing sharing

Intel® VT for directed I/O (VT-d)

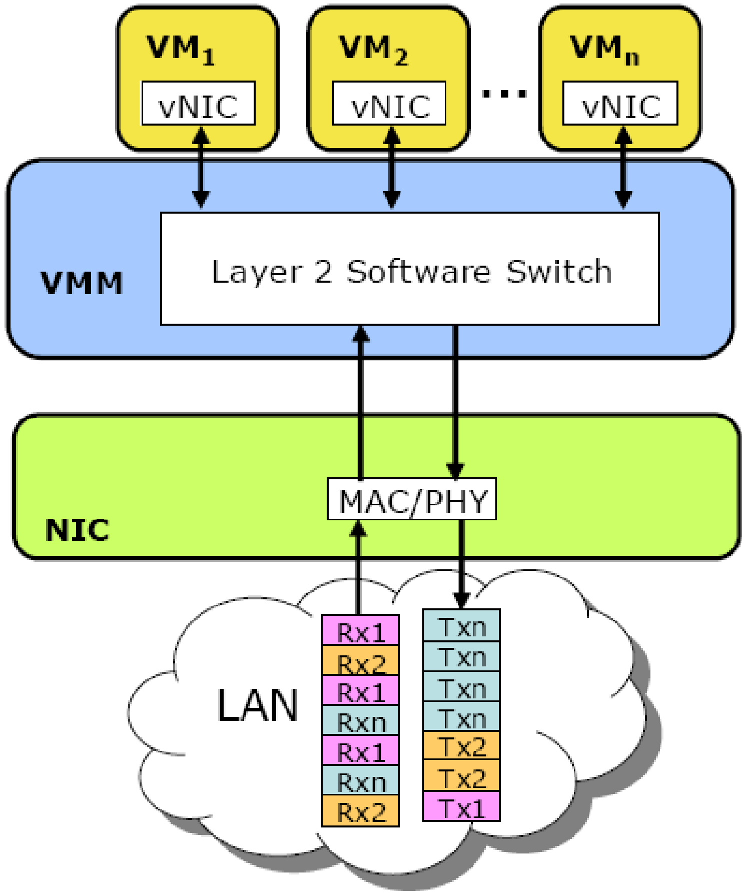


Single Root I/O Virtualization (SR-IOV)

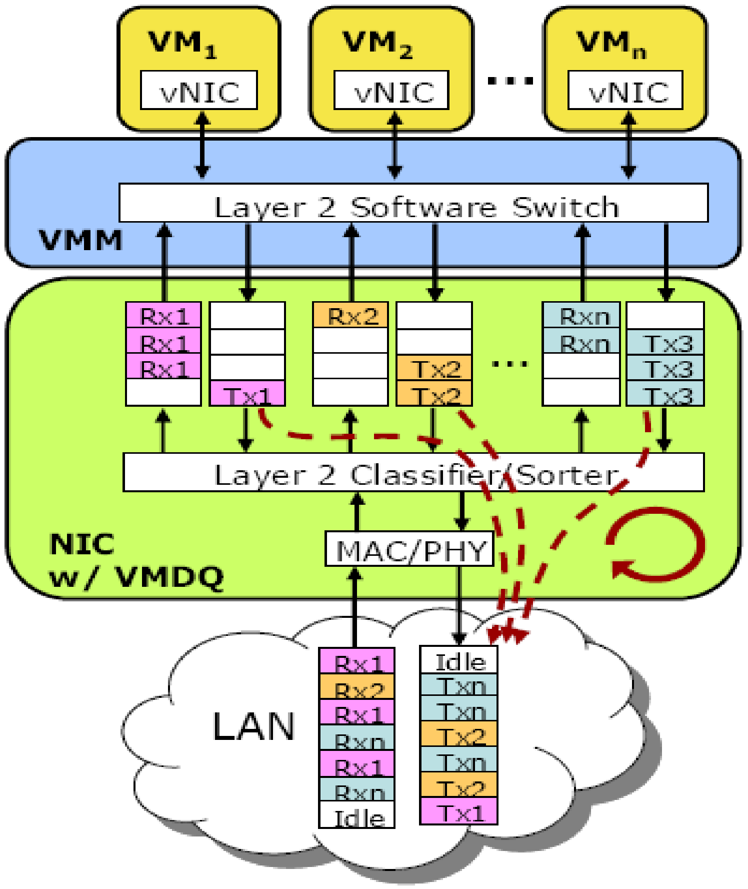


- **Part of Intel® VT for connectivity, i.e. VT-c**
  - Close to native performance
  - Limited CPU overhead
  - Flexible sharing
  - Security control through PF

Virtual Machine Device Queue (VMDq)



Software Network virtualization



Network virtualization with VMDq

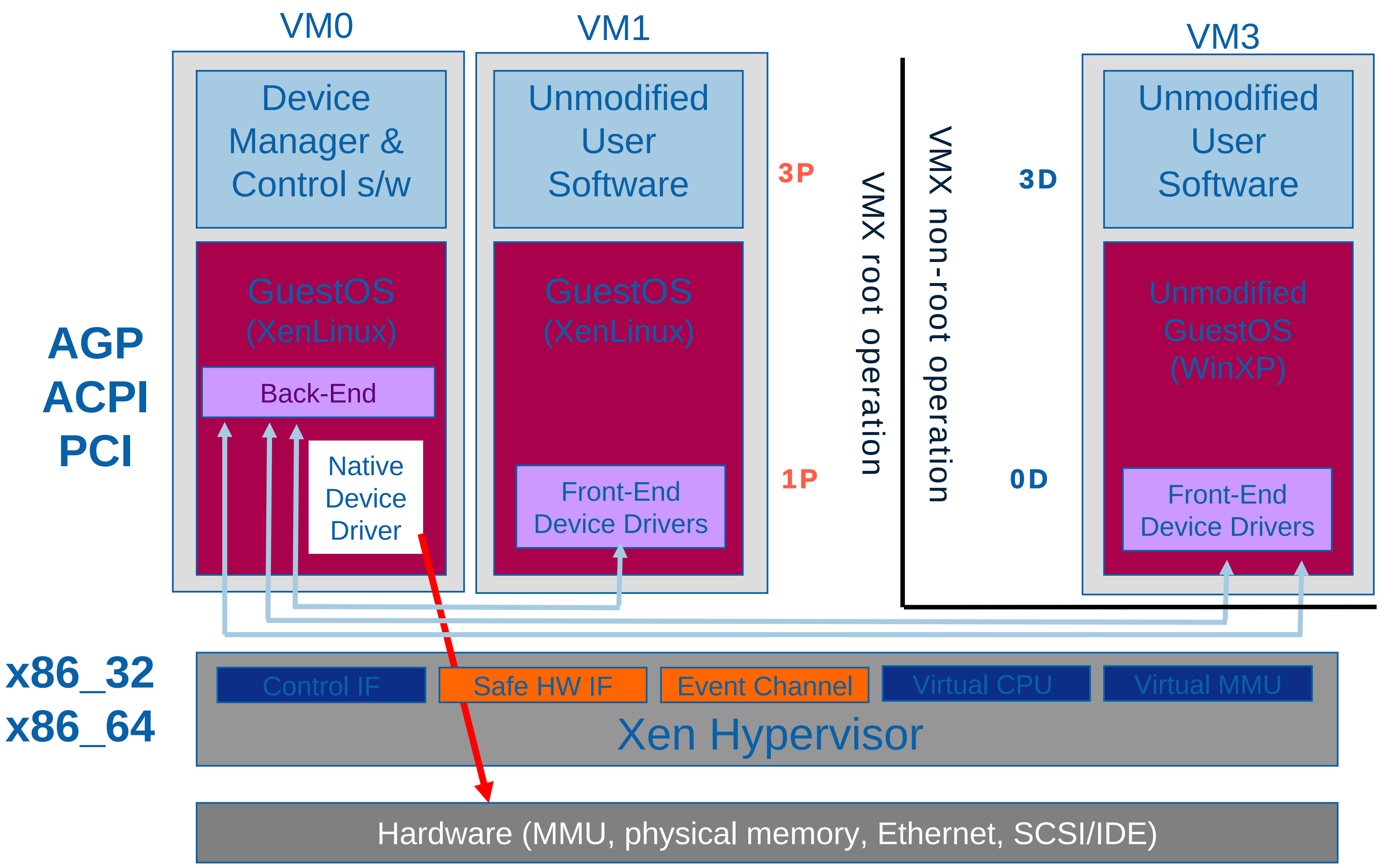
Agenda

- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
  - I/O
- Commercial VMMs
- Summary

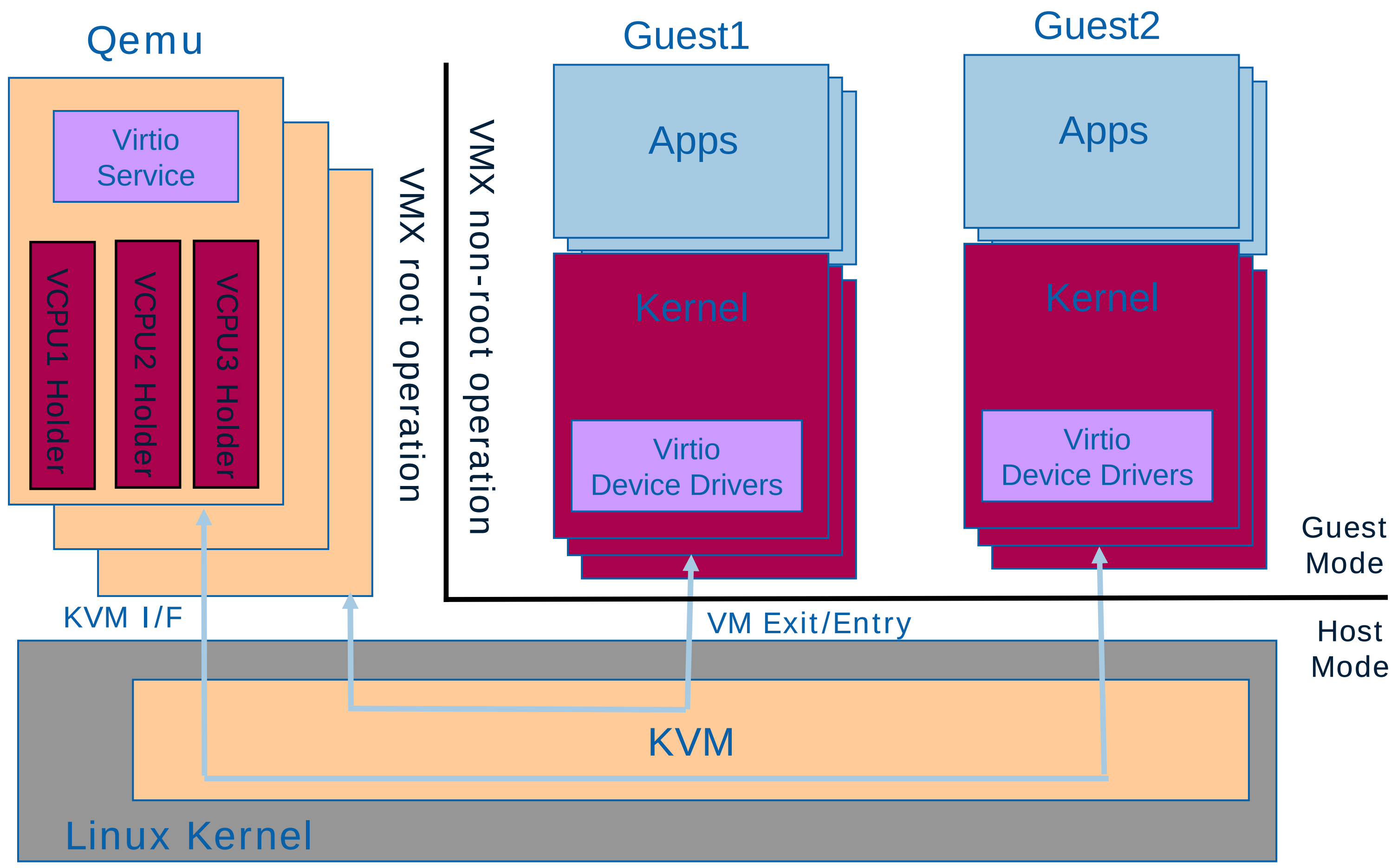
Commercial VMMs

- Xen
- KVM
- Hyper-V
- VMware ESX
- VMware workstation
- Parallels

Xen Architecture

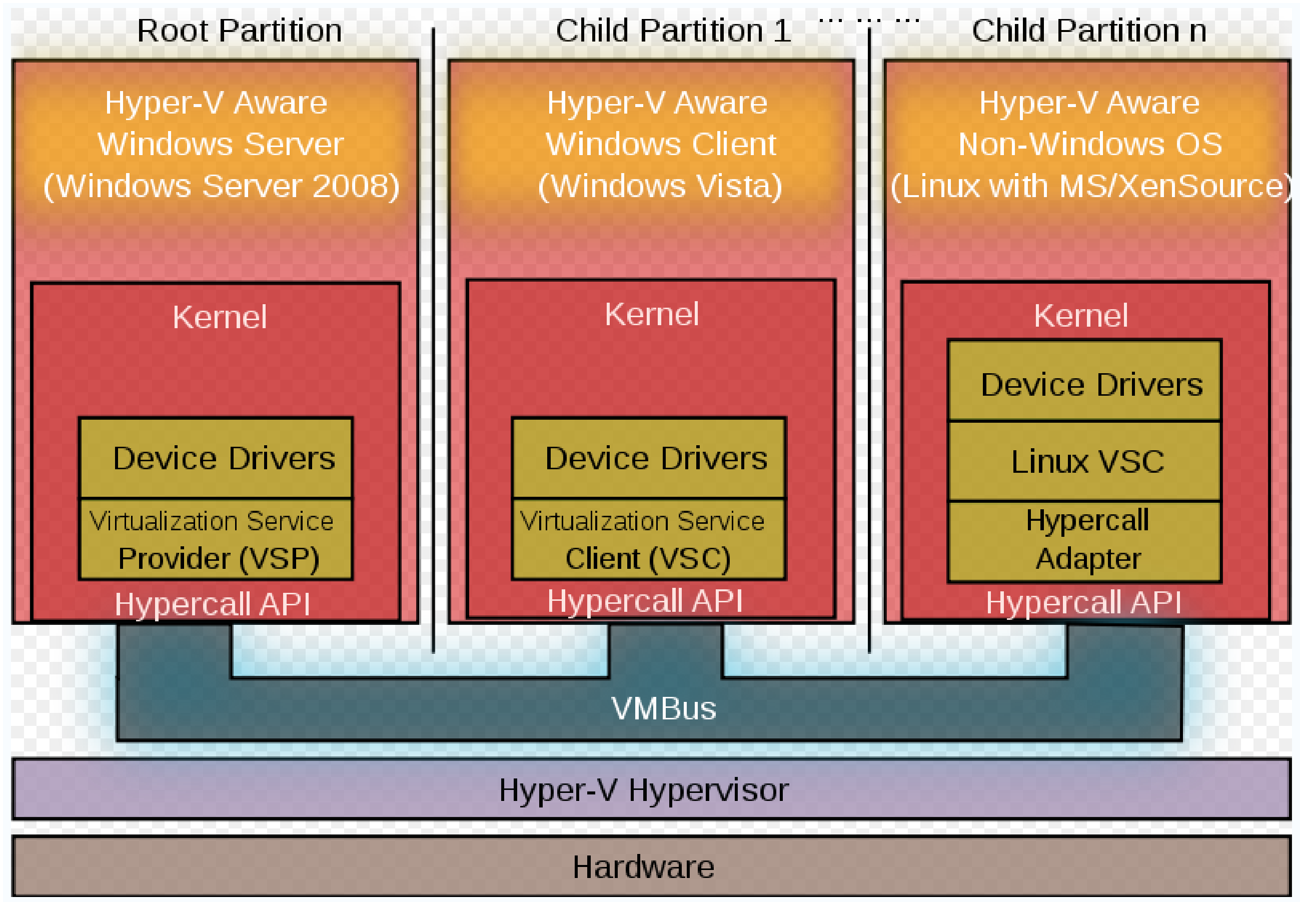


KVM architecture



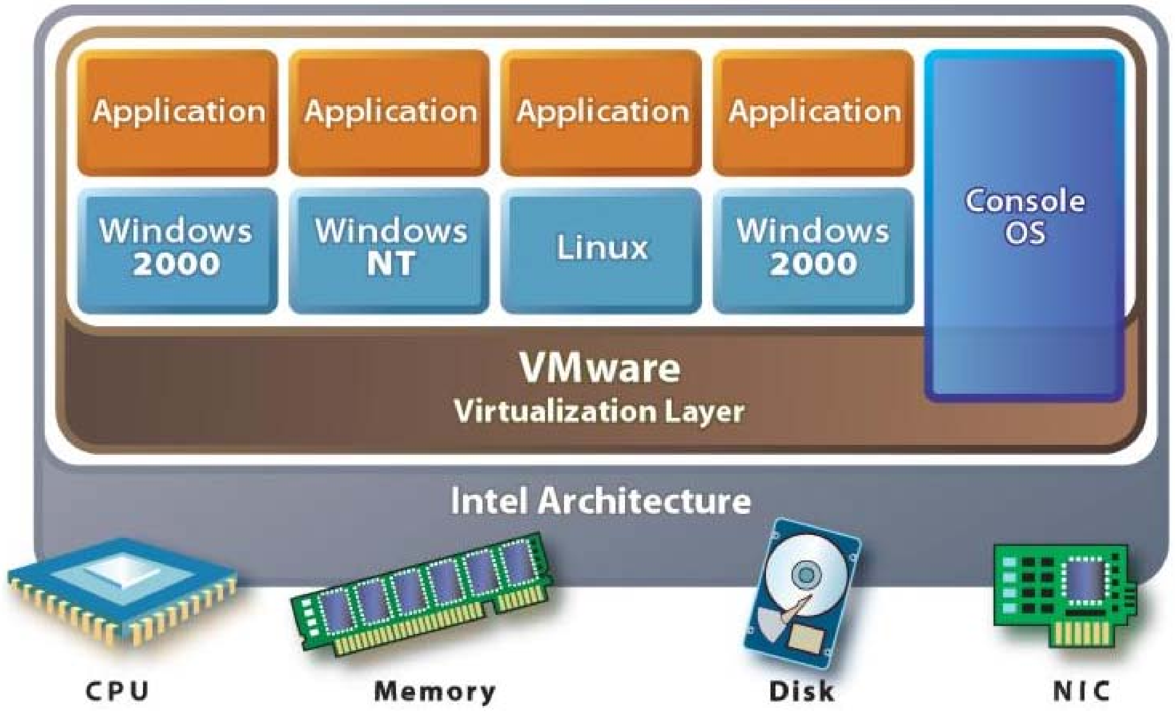
KVM runs in HVM container, PVM features may be used

Hyper-V architecture


Root Partition
Child Partition 1
… … …
Child Partition n
Hyper-V Aware Windows Server (Windows Server 2008)
Hyper-V Aware Windows Client (Windows Vista)
Hyper-V Aware Non-Windows OS (Linux with MS/XenSource)
Kernel
Device Drivers
Virtualization Service Provider (VSP)
Virtualization Service Client (VSC)
Linux VSC
Hypercall Adapter
Hypercall API
VMBus
Hyper-V Hypervisor
Hardware

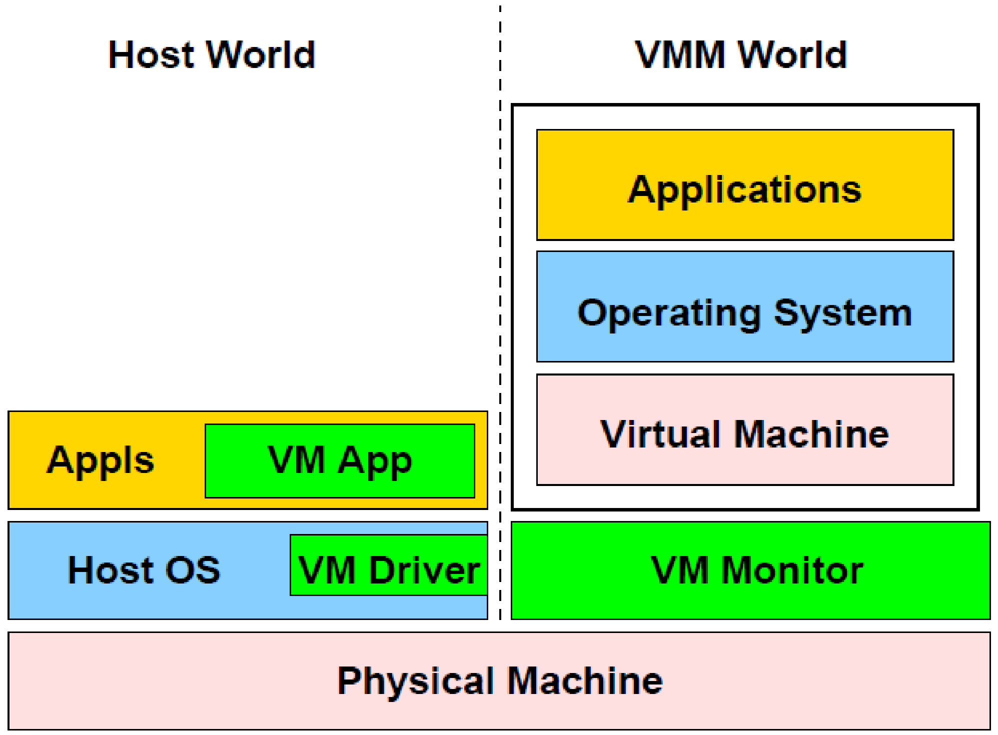
*Source from wikipedia under **GNU Free Documentation License**

VMware ESX Server architecture



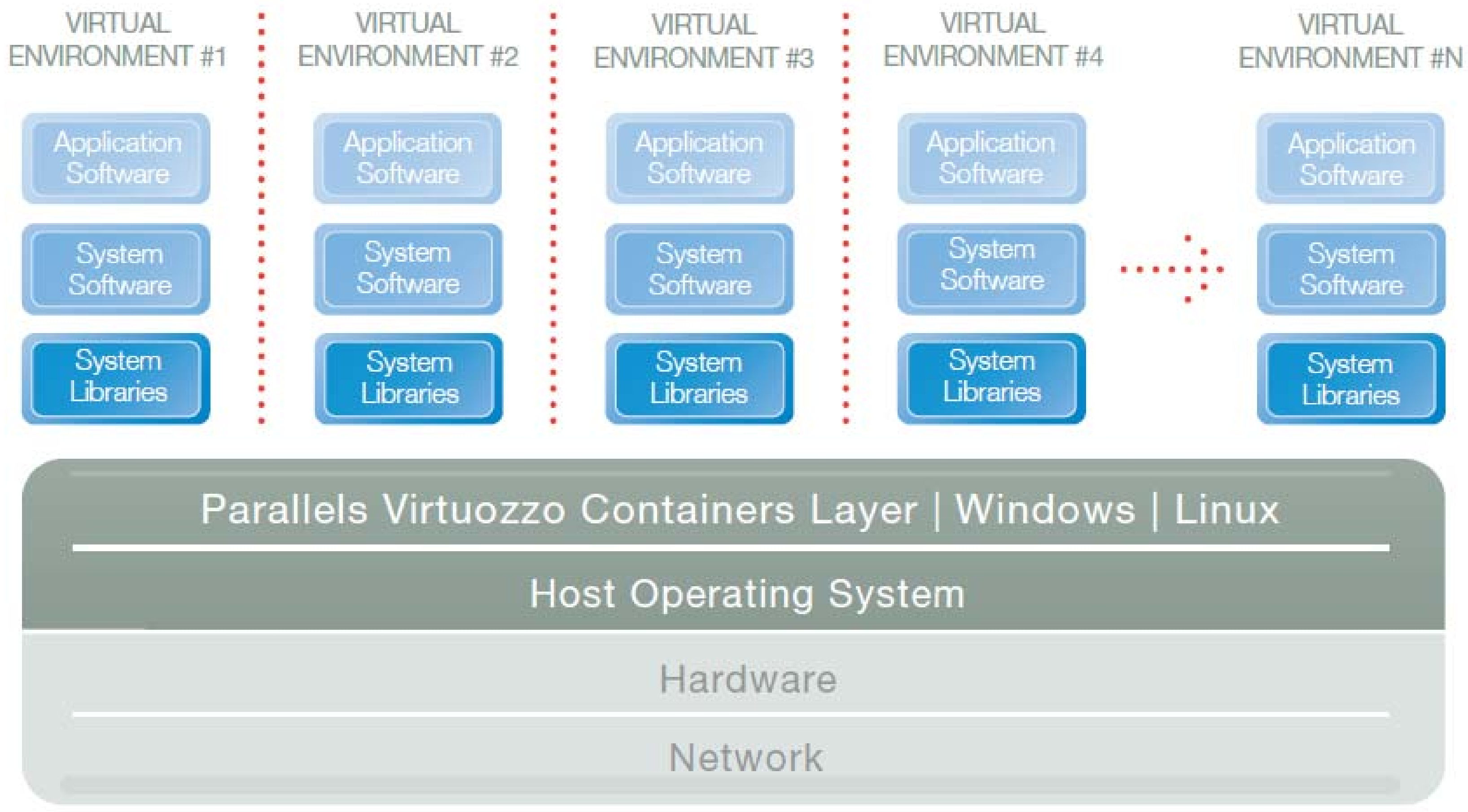
*Source from Vmware

VMware Workstation Architecture



*Source from proceeding of 2001 USENIX ATC

Parallels - Container



Agenda

- Introduction
- Traditional Virtualization Challenges
- Virtualization Technologies
  - CPU
  - Memory
  - I/O
- Commercial VMMs
- Summary

Intel® Virtualization Technology Evolution

VT capabilities build upon one another to address full-system virtualization issues in sync with VMM software evolution

Vector 3: IO Focus

Assists for IO-device sharing (e.g., Intel® Virtualization Technology for connectivity, etc.)

Vector 2: Chipset Focus

Hardware assists for translated phys-mem access:
- Support for IO-device assignment to VMs
- DMA Remapping Mechanism

Vector 1: Processor Focus

VT-x

VT-i

Close basic processor "virtualization holes" in Intel® 64 and IPF processors

Continuous evolution of processor virtualization assists, some micro-architectural, others architectural (e.g., extended page tables, EPT)

VMM Software Evolution

Software-only VMMs
- Binary translation
- Paravirtualization

Simpler and more **secure** VMM through use of hardware support

Better IO/CPU **perf** and **functionality** via hardware-mediated access to memory

Richer IO-device **functionality** and IO resource **sharing**

No HW Support

VMM software evolution over time with hardware support

Links for materials

- Intel® 64 Software Development Manual
  - http://www.intel.com/products/processor/manuals/
- Extending Xen* with Intel® Virtualization Technology
  - http://www.intel.com/technology/itj/2006/v10i3/3-xen/1-abstract.htm
- ClientVisor: leverage COTS OS functionalities for power management in virtualized desktop environment
  - http://portal.acm.org/citation.cfm?id=1508293.1508312
- Towards high-quality I/O virtualization
  - http://portal.acm.org/citation.cfm?id=1534547
- High performance network virtualization with SR-IOV
  - Will be in HPCA-16, Bangalore, India.
- Intel Developer Forum
  - http://www.intel.com/idf/

Thank You!