Virtual Machines

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

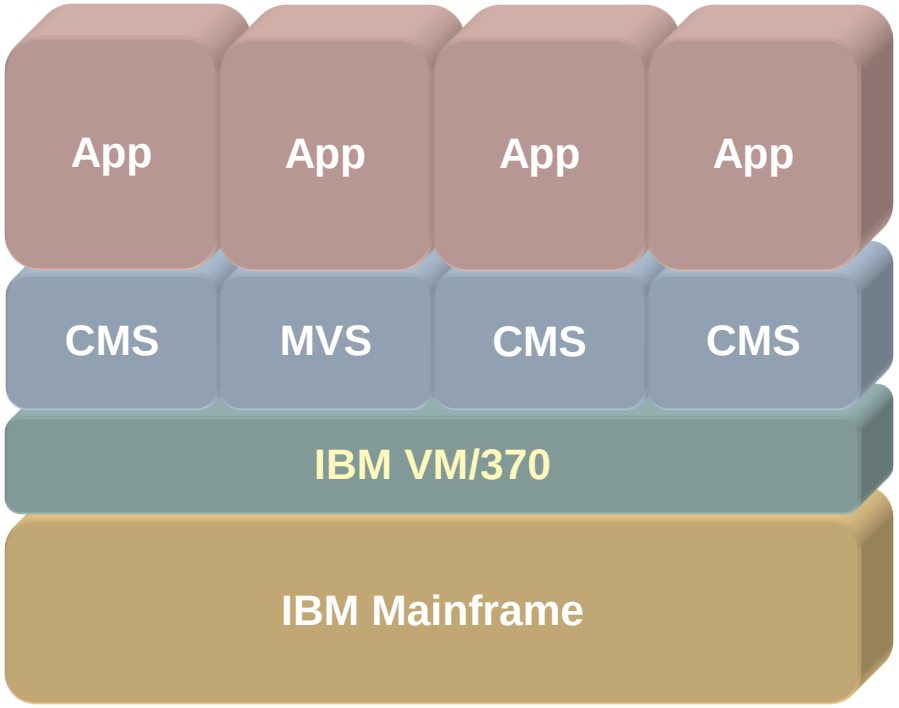
Seoul National University

Spring 2019

Virtual Machine

- A fully protected and isolated copy of the underlying physical machine's hardware (definition by IBM)

- Virtual machine monitor (VMM)
  - A thin software layer that sits between hardware and the operating system – virtualizing and managing all hardware resources
  - "Hypervisor"



History: Old Idea from 1960s

- IBM VM/370 – A VMM for IBM mainframe
  - Multiple OS environments on expensive hardware
  - Desirable when few machine around
- Popular research idea in 1960s and 1970s
  - Entire conferences on virtual machine monitor
  - Hardware/VMM/OS designed together
  - Robert Goldberg, Architectural Principles for Virtual Computer Systems, Ph.D. Thesis, Harvard University, 1972.
- Interest died out in the 1980s and 1990s
  - Hardware got cheap
  - OS got more powerful (e.g., multi-user)

A Return to Virtual Machines

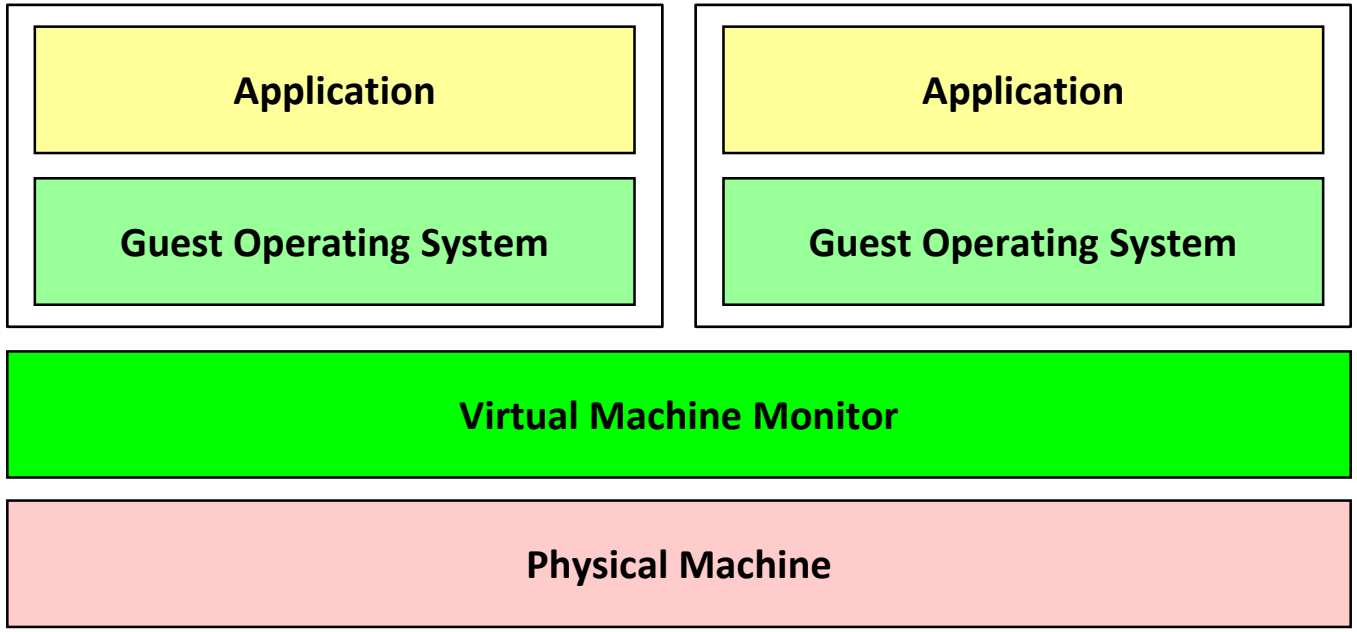
- Disco: Stanford research project (SOSP '97)
  - Run commodity OSes on scalable multiprocessors
  - Focus on high-end: NUMA, MIPS, IRIX
- Commercial virtual machines for x86 architecture
  - VMware Workstation (now EMC/Dell) (1999 -)
  - Connectix VirtualPC (now Microsoft)
- Research virtual machines for x86 architecture
  - Xen (SOSP '03), plex86
- OS-level virtualization
  - FreeBSD Jails, Linux Docker

Why Virtual Machines?

- Create the illusion of multiple VMs
  - Ability to handle multiple, different, complete OSes
  - Useful in the development and testing of new software
- Strong isolation between VM instances
  - Enable to run multiple, untrusted applications safely
- Software compatibility
  - OS can be upgraded without losing the ability to run older "legacy" OS and its apps
- Logical partitioning and server consolidation
  - Improve utilization and ease server management with less cost
  - Workload can be migrated to a different physical machine
- Convenient environment for debugging OSes

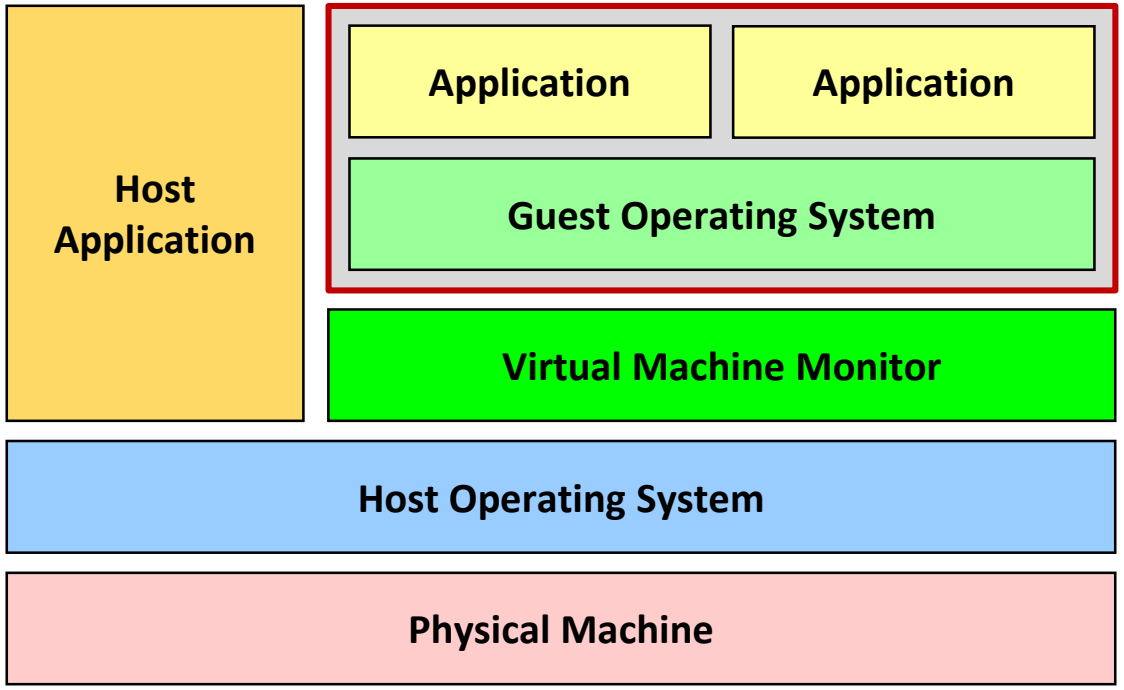
Type I VMM: Bare-Metal Hypervisors

- VMM is implemented directly on the physical hardware
- VMM performs the scheduling and allocation of the system's resources
- IBM VM/370, Disco, VMware ESX Server, Xen, Hyper-V



Type II VMM: Hosted Hypervisors

- VMMs are built completely on top of a host OS
- A guest OS runs as a process on the host
- VMware Workstation/Player, Virtual Box, Parallels Desktop for Mac, KVM?



Related Technologies

- Complete machine simulators
  - Bochs (x86), Qemu (x86), SimOS (MIPS R4000/R10000), SimICS (x86, Alpha, ARM, IA-64, MIPS, PowerPC, Sparc v9)
  - Portable: Runs instructions purely in software
  - Slow (e.g., 100x slow down for Bochs)
  - Portability vs. performance
- ABI/API emulators
  - WINE (Windows Emulator or Wine Is Not an Emulator)
    - Port of Windows API to X-windows/Unix.
  - Focuses on getting system call for a particular operating system's interface.
- High-level language VMs: Java VM, etc.

Disco: Running Commodity Operating Systems on Scalable Multiprocessors

(E. Bugnion et al., SOSP, 1997)

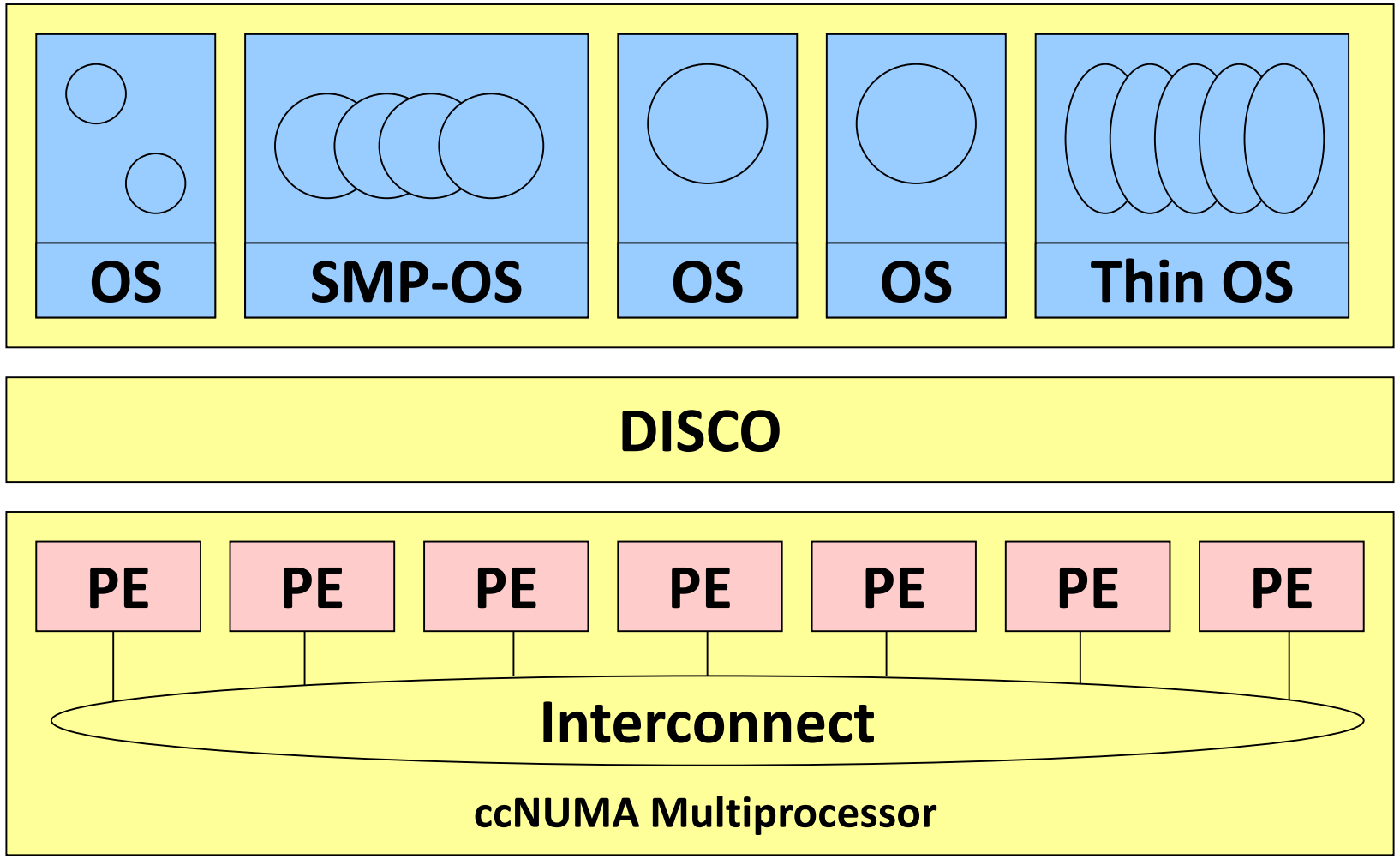
Background

- ccNUMA: Cache-coherent non-uniform memory architecture
  - Multiprocessor with high-performance interconnect
- Non-uniform memory
  - Global address space
  - But memory distributed amongst processing elements
- Cache-coherence
  - Issue: How to ensure that memory in processor caches is consistent?
  - Solutions: Bus snooping, directory
- Targeted system: FLASH, Stanford's own ccNUMA
- "Commodity OS": SGI IRIX

The Challenge

- Commodity OSes not well-suited for ccNUMA
  - Do not scale
    - Lock contention, memory architecture
  - Do not isolate/contain faults
    - More processors → more failures
- Customized operating systems
  - Take time to build, lag hardware
  - Cost a lot of money

The Solution: DISCO



How to Virtualize?

- Virtualize physical resources
  - CPU: instructions → Trap all privileged instructions
  - Memory: address spaces → Map "physical pages" managed by the guest OS to machine pages, handle translation, etc.
  - Devices → Any I/O communication needs to be trapped and passed through/handled appropriately
- Dispatch events
  - e.g., forward page fault trap to guest OS
- Manage resources
  - e.g., divide real memory in some way between the physical memory of each guest OS

Virtualizing CPUs

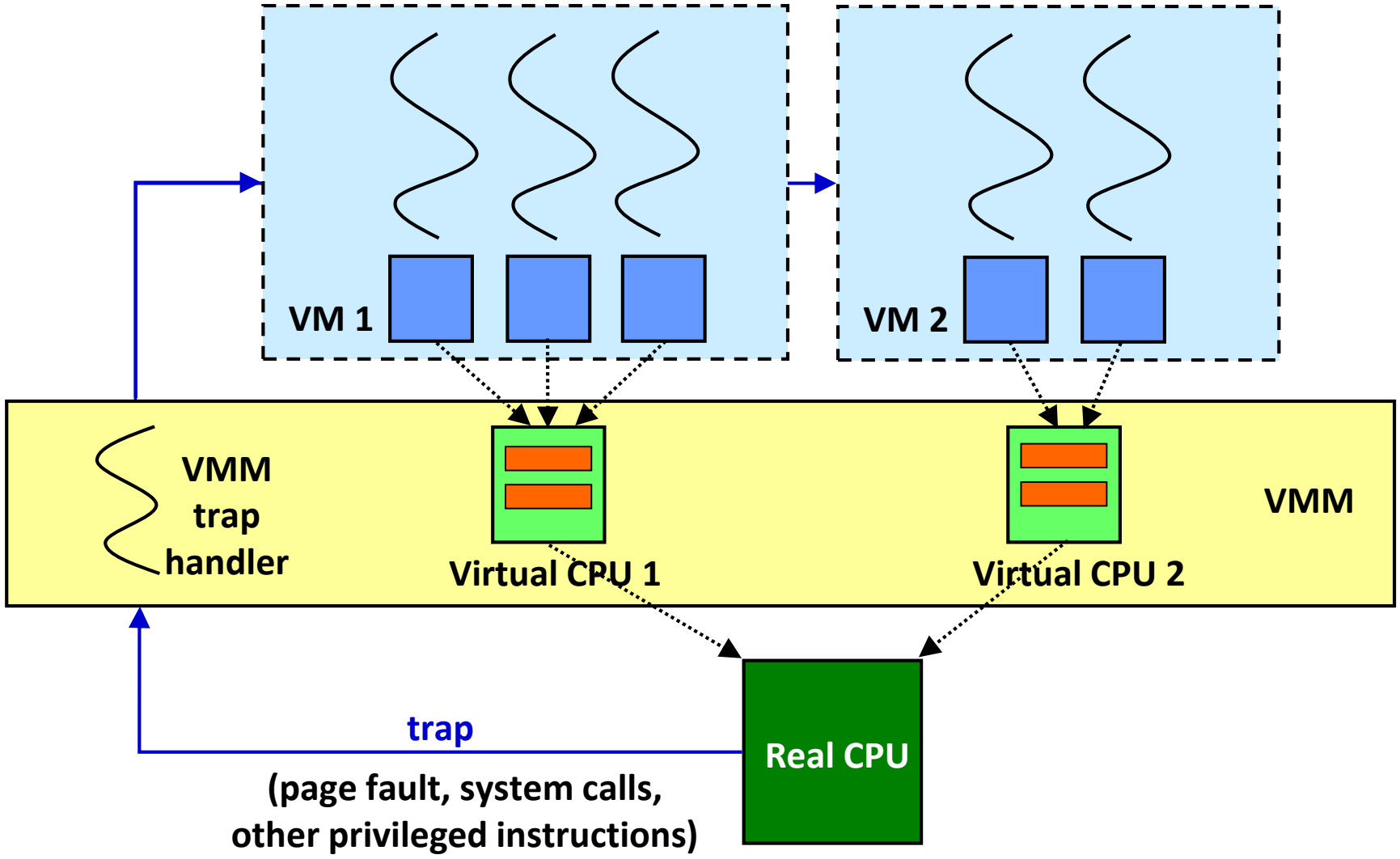
- MIPS R10000 in FLASH: the (unfortunate) choice for Disco
- MIPS R10000 has three operating modes
  - Kernel mode: Disco
  - Supervisor mode: Guest OS
  - User mode: applications
- MIPS R10000 does not support the complete virtualization
  - A processor running in supervisor mode cannot access the KSEG0 segment efficiently, that bypasses the TLB
  - IRIS 5.3 places the kernel code and data in the KSEG0 segment
  - Requires modifications to the IRIX kernel to relocate the kernel to the mapped supervisor segment

Virtualizing CPUs: Virtual CPU

- For each virtual CPU, Disco keeps a data structure for
  - The saved registers
  - The privileged registers
  - TLB contents of the virtual CPU
  - Other state of a virtual CPU

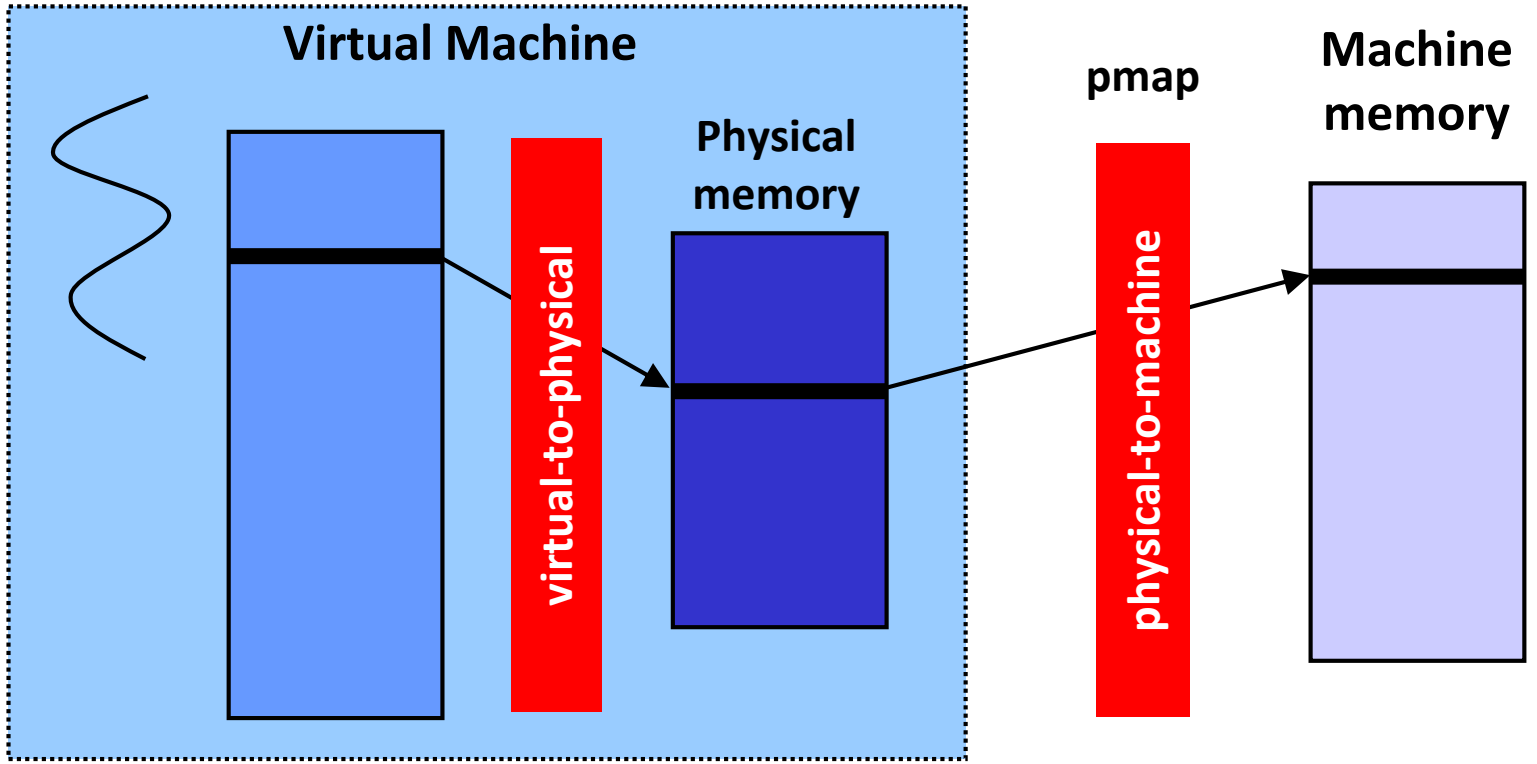
- Virtualizing CPUs
  - To schedule a virtual CPU, Disco sets the real machines' registers to those of the virtual CPU and jumps to the current PC of the virtual CPU
  - Disco emulates the operations that cannot be issued in the supervisor mode
  - Disco simply time-shares the virtual processors

Virtualizing CPUs: Example



Virtualizing Memory: Machine Address

- Address used by the (physical) memory system of the FLASH machine
- Adds a level of address translation: physical-to-machine



Virtualizing Memory

- When a guest OS attempts to insert a virtual-to-physical mapping into the TLB:
  - Disco translates the physical address into the corresponding address, and inserts this corrected TLB entry
- Pmap data structure accelerates the computation of the corrected TLB entry
  - For each physical page of a virtual machine, pmap entry contains a precomputed TLB entry that references the physical page location in real memory
  - Disco only merges that entry with the protection bits
  - Backmaps in pmap points to the virtual address; used to invalidate mappings from the TLB

Virtualizing Memory: TLB Handling

- TLB in Disco
  - MIPS TLB is software-managed and supports ASID (Address Space Identifier)
  - TLB is flushed on virtual CPU switches
  - TLB miss handling is expensive
    - Emulation of the trap architecture
    - Emulation of privileged instructions in the OS's TLB miss handler
    - Remapping of physical addresses

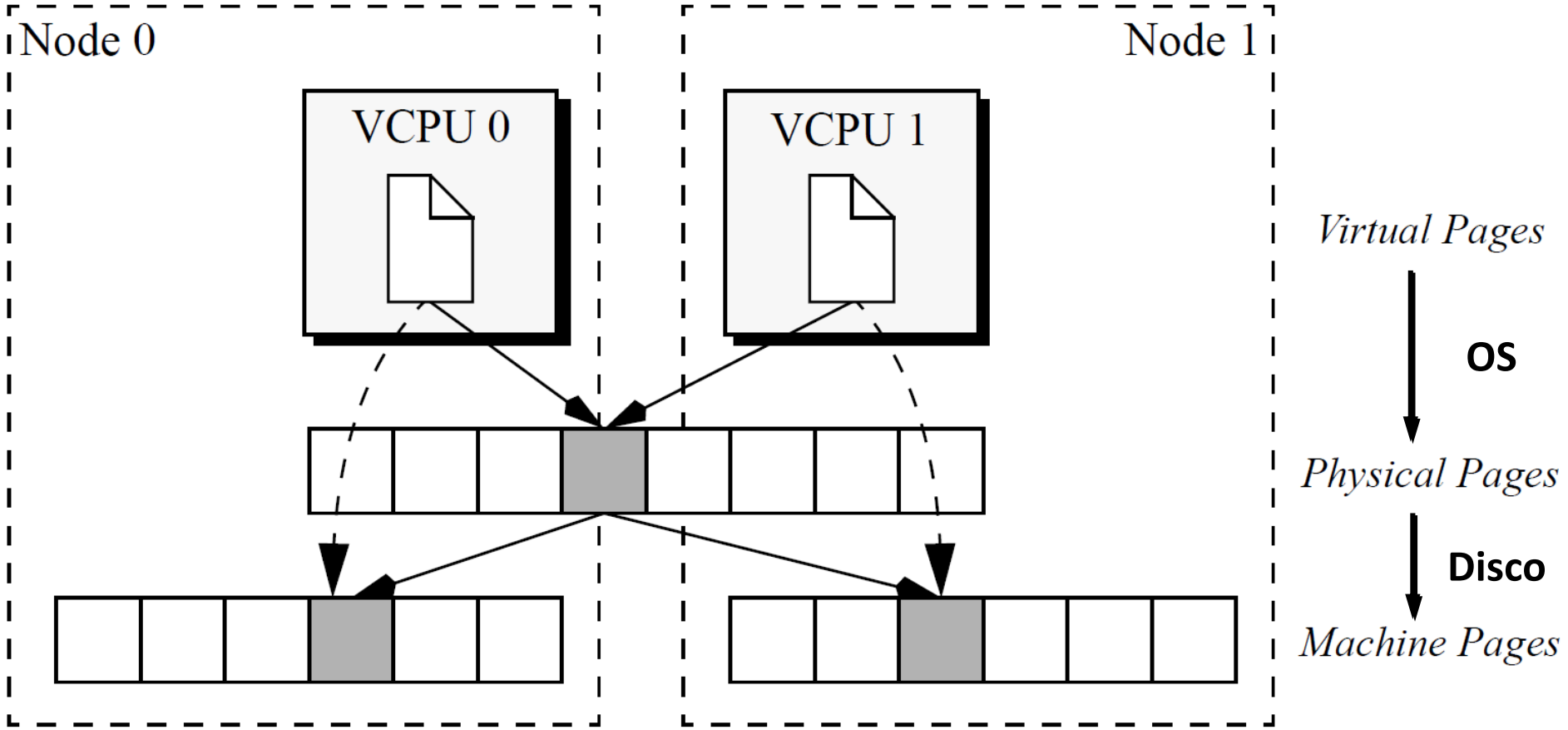
- L2TLB (second-level software TLB)
  - L2TLB caches recent virtual-to-machine translations
  - On a TLB miss, Disco consults L2TLB first
  - If there is no match, Disco forwards the TLB miss exception to the OS

Virtualizing Memory: NUMA Handling

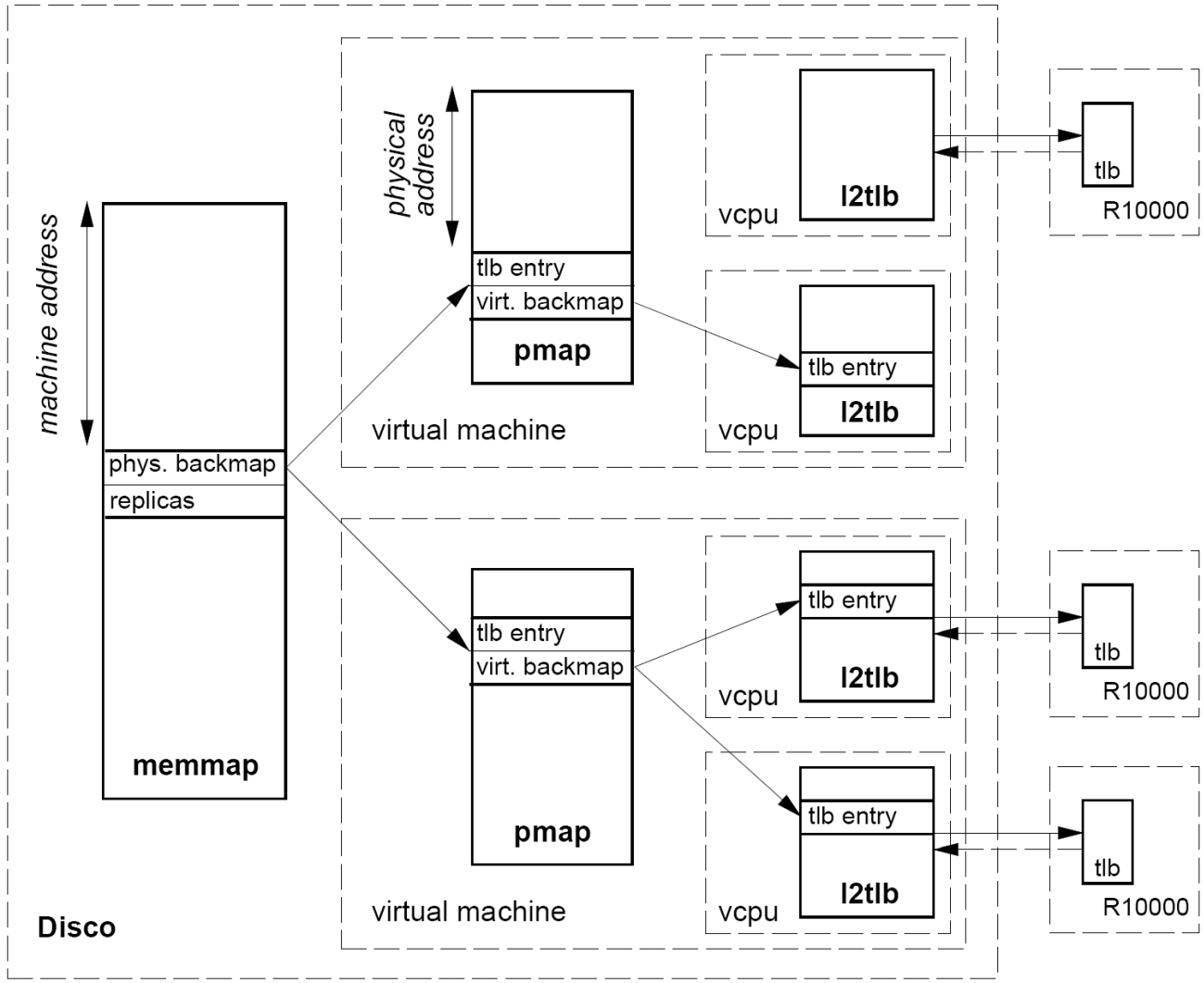
- Dynamic page migration
  - Pages that are heavily accessed by only one node are migrated to that node
- Page replication
  - Pages that are primarily read-shared are replicated to the nodes most heavily accessing them
- FLASH detects a hot page by counting cache misses to each page from every physical processor
- Memmap data structure
  - A list of the virtual machines using the machine page and the virtual addresses used to access them
  - Used for TLB shootdown during page migration and replication

Virtualizing Memory: Page Replication

- Transparent page replication



Virtualizing Memory: Data Structures



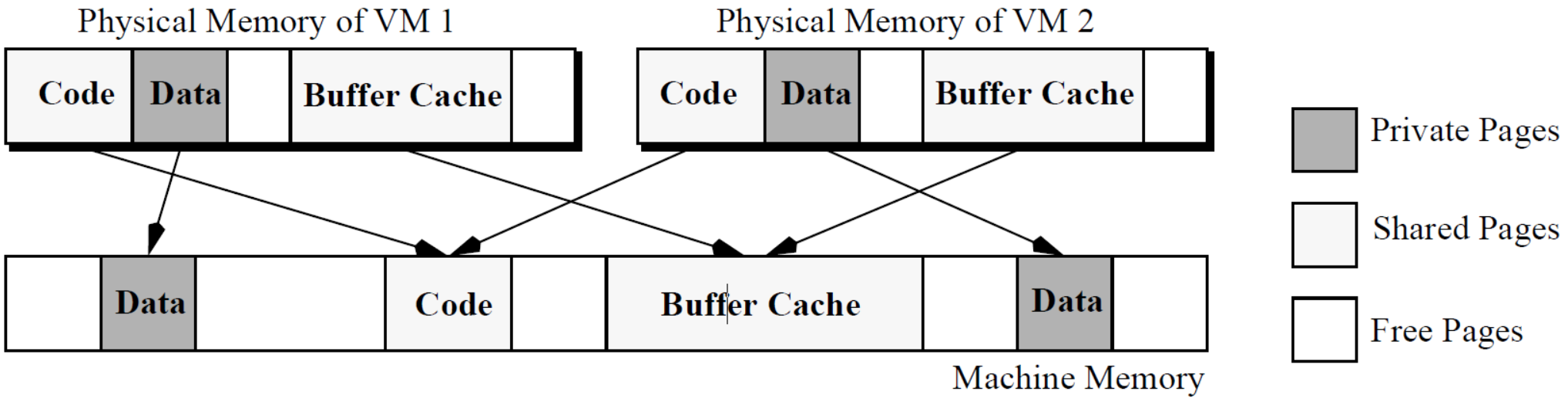
Virtualizing I/O Devices

- Intercept the programmed I/O from the guest OS and emulate the functionality of the hardware device
  - Complex, specific to each device, and require many traps

- Disco: Add special device drivers into the guest OS
  - Use a monitor call to pass all command arguments in a single trap
  - DMA target addresses (physical addresses) should be translated into machine addresses
  - Disco supports UART, SCSI disks, and ethernet drivers

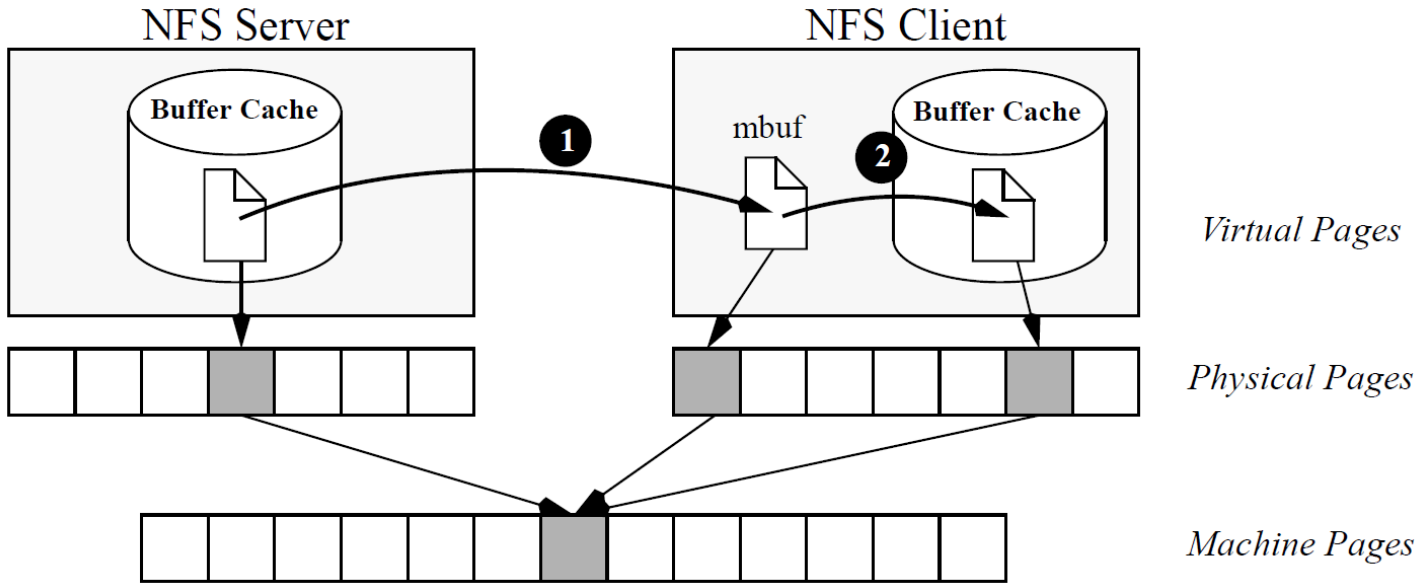
Virtualizing I/O Devices: Disks

- Share disk blocks by mapping the page into the VM's physical memory
- For code and other read-only data (e.g., root disks)
- Sharing read-write data
  - Mount separate disk partition
  - Use NFS



Virtualizing I/O Devices: NICs

- Use copy-on-write mappings to reduce copying and to allow for memory sharing
- Send buffer is remapped to receive buffer
- Receive buffer is remapped again to the buffer cache



Discussion

- Running commodity OS on VM
- Disco still requires kernel modification
  - Some for inherent CPU restrictions
  - Some for optimizations

- Disco and after …
  - Cellular Disco (SOSP '99): For SMPs
  - VMware founded in 1998: For Windows/x86

Intel Virtualization Technology

CPU Requirements

- The CPU should be designed with the virtualization in mind
- Strictly virtualizable processor architectures
  - All instructions that can inspect or modify privileged machine state will trap when executed from any but the most privileged mode
  - No software running inside the VM can determine the presence of the VMM
  - Examples: IBM S/390, DEC Alpha, IBM PowerPC
- Non-strictly virtualizable processor architectures
  - Some instructions have different semantics at various levels
  - Executing certain instructions with insufficient privilege fails silently rather than causing a trap
  - Examples: IA-32, IA-64

Virtualizable Architecture

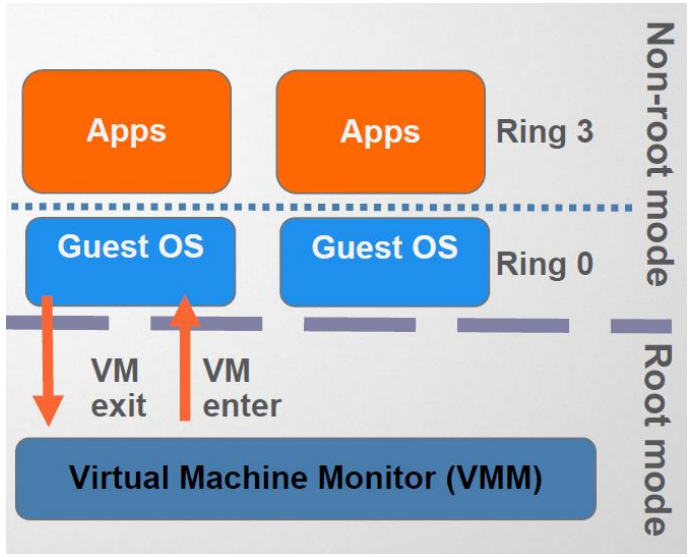
- Normal instructions
  - Can be freely executed at user mode
- *Privileged* instructions
  - Cause a trap when executed without appropriate privilege
- *Sensitive* instructions
  - Must be emulated (virtualized) for fidelity and safety for VMM
- Virtualizable architecture [Popek & Goldberg, CACM, 1974]
  - Sensitive instructions $\subseteq$ Privileged instructions
- x86 was not virtualizable before 2005
  - Cannot emulate sensitive instructions that are not privileged (e.g., SGDT, SLDT, SIDT, …)

What about x86?

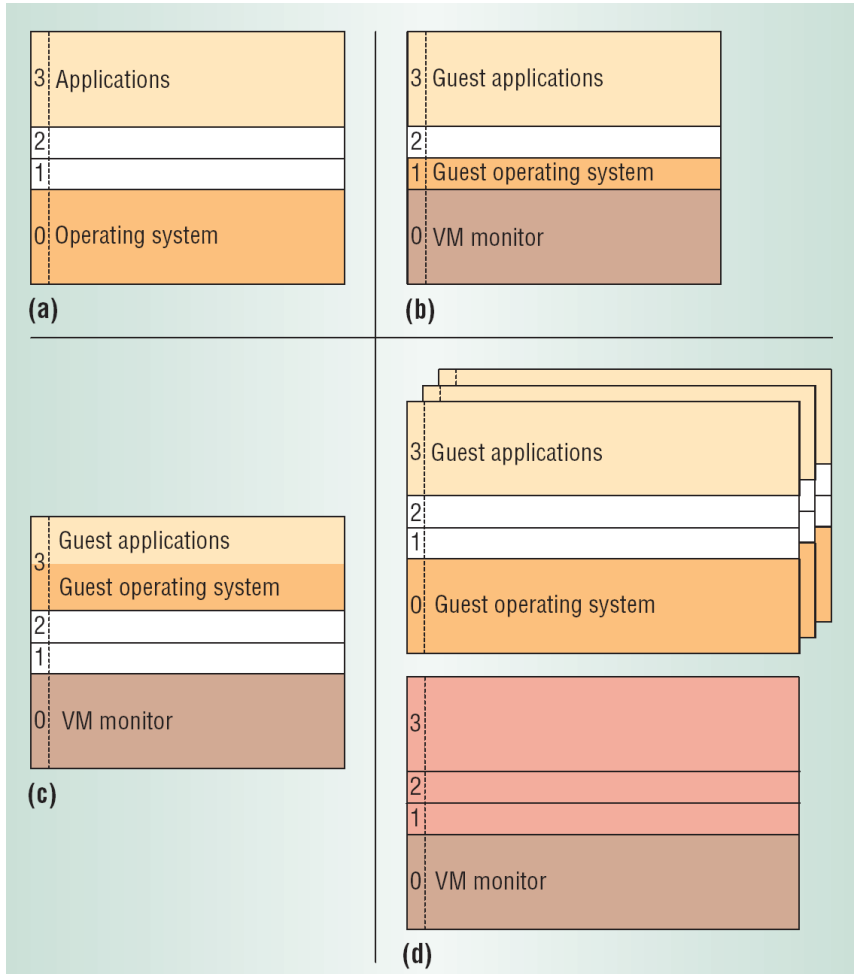
- VMware ESX server
  - Dynamically rewrites portions of the hosted machine code to insert traps whenever VMM intervention might be required
  - Runs unmodified operating systems
- Xen: paravirtualization
  - Presents a VM abstraction that is similar but not identical to the underlying hardware
  - Requires modifications to the guest OS (but not to guest apps)
- Plex86
  - Executes the modified Linux kernel at user privilege
  - Privileged instructions will naturally generate exceptions and can be monitored

Intel Virtualization Technology (VT-x)

- New VMX root mode & instructions
  - (a) Nonvirtualized system
  - (b) The 0/1/3 model
  - (c) The 0/3/3 model
  - (d) Virtualized system using VT-x



Source: J. Lo, "VMware and CPU Virtualization Technology", VMworld, 2005.



Source: R. Uhlig et al., "Intel Virtualization Technology," IEEE Computer, 2015.

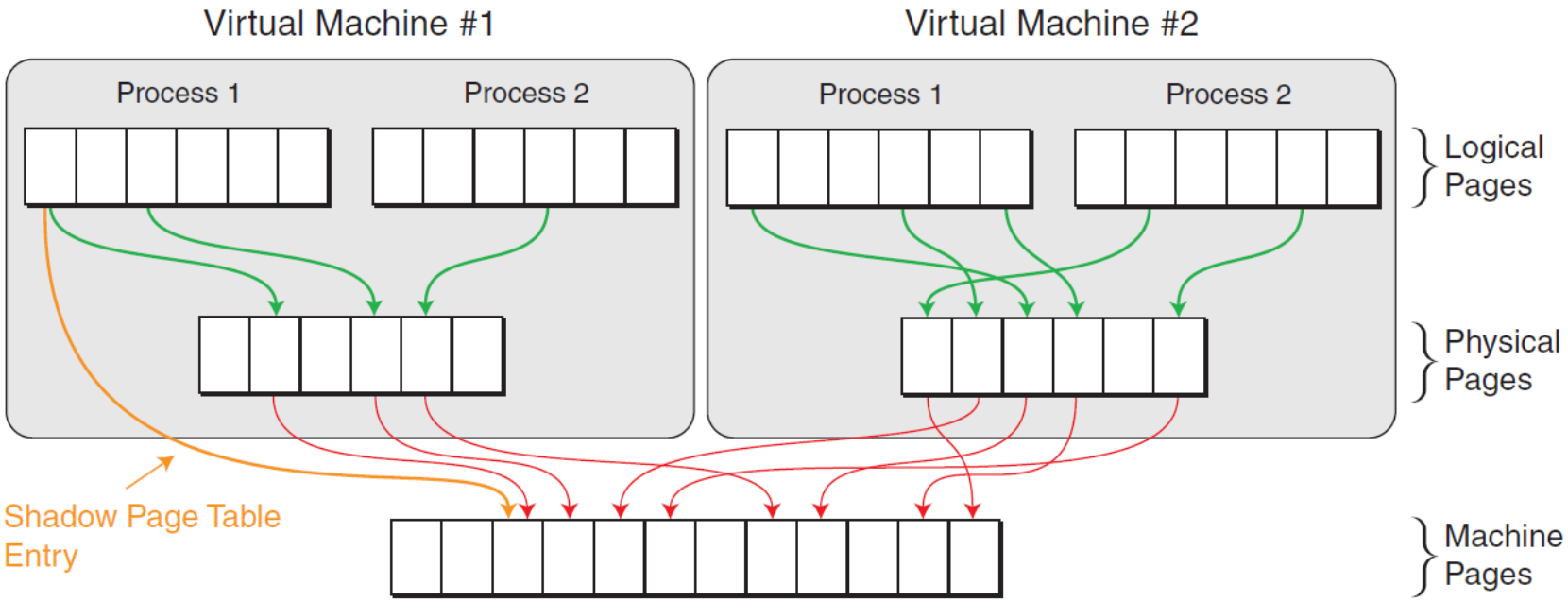
Memory Virtualization

- Virtualizing virtual memory:
  Virtual address → Physical address → Machine address

- Secure memory isolation among VMs
  - A VM is NOT permitted to access another VM's memory region
  - A VM is NOT permitted to manipulate "physical-to-machine" mapping
  - All mappings to machine memory MUST be verified by VMM

Shadow Page Tables

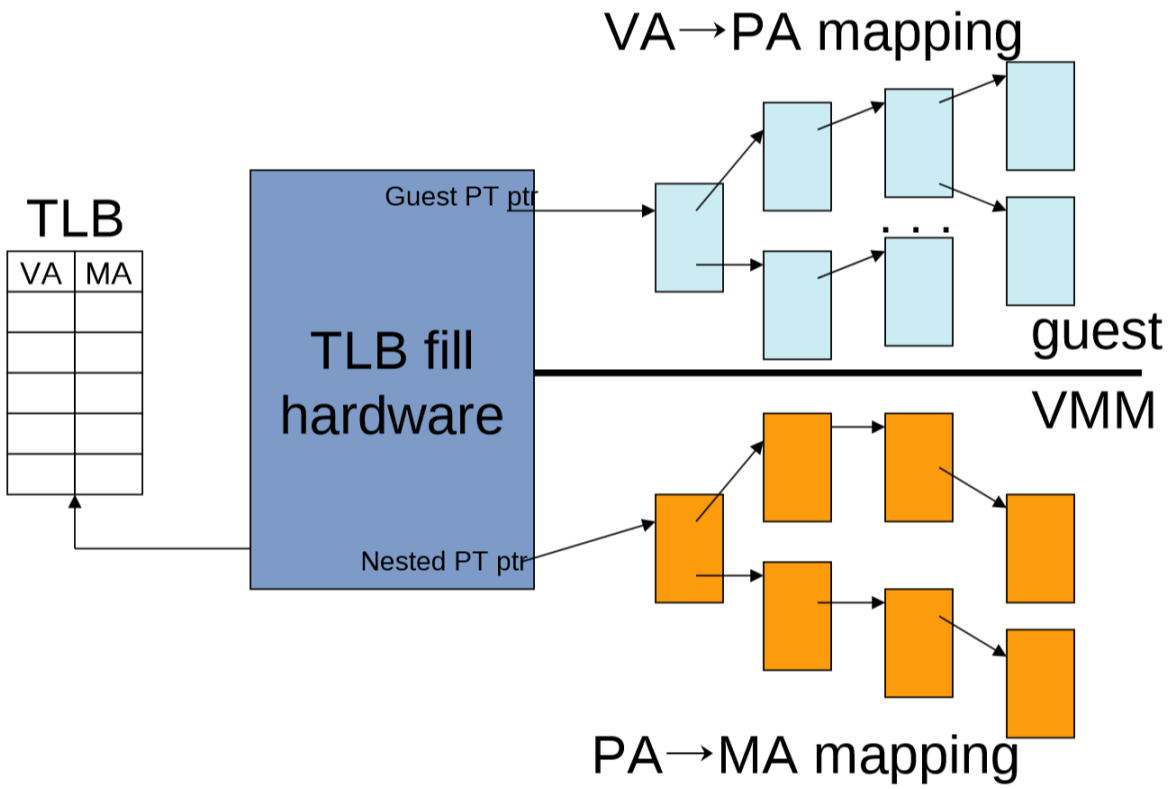
- VMM maintains PPN→MPN mappings in its internal data structures
- Shadow page table has LPN→MPN mappings (visible to hardware)
- Shadow page tables should be synchronized to guest page tables



Source: Performance Evaluation of Intel EPT Hardware Assist, VMware, 2009.

Intel Extended Page Tables (EPT)

- VMM maintains PPN→MPN mappings in "nested page tables"
- For every PPN accessed during guest page table walk, the hardware also walks nested page tables to determine the corresponding MPN



I/O Virtualization

- Trap-and-emulate
  - I/O operations are all privileged and trapped: programmed I/O (PIO), memory-mapped I/O (MMIO), direct memory access (DMA)
- Para-virtualization
  - Rewrite (front-end) device drivers for guest VM
  - Back-end drivers either in the VMM, in the host OS, or in the separate domain (Xen)
- Intel Virtualization Technology for Directed I/O (VT-d)
  - Unmodified guest OS can obtain direct access to the I/O device without intervention of the VMM
  - For PCIe devices with SR-IOV (Single-Root I/O Virtualization) support