

Virtual Machines

Concepts

- Virtualization:
 - Creation of flexible substitutes for actual resources.
 - The substitutes and their actual counterparts:
 - have same functions and external interfaces
 - differ in size, performance, cost etc.
 - Resources to virtualize
 - CPU
 - Memory
 - I/O

Concepts

- System Virtualization
 - System virtualization creates several virtual systems within a single physical one.
- VMM (or hypervisor)
 - Virtual machine monitor is the software layer providing the virtualization.
- VM
 - Virtual machine is the virtual systems running on top of VMM

Brief History

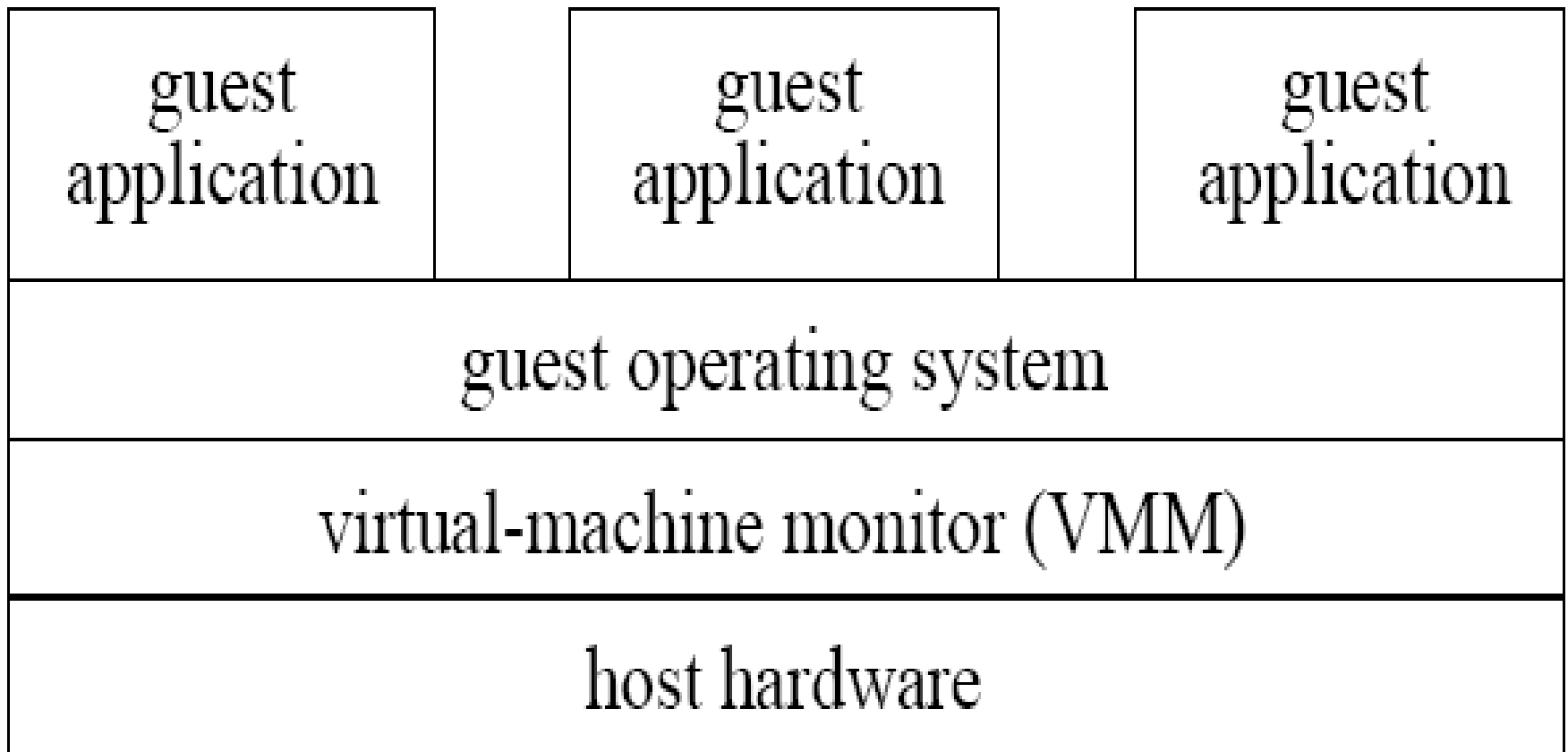
- 1960s, first introduced, for main frames
 - Motivation: hardware cost etc.
- 1970s, an active research area
- 1980s, underestimated
 - Multitask modern operating systems took its place
 - Decreasing in hardware cost
- late 1990s, resurgence: software techniques for x86 virtualization
 - Many applications: mixed-OS develop environment, security, fault tolerance etc.
- mid 2000s, hardware support from both Intel and AMD

Types of Virtualization

- Process virtualization (virtualize one process)
 - The VM supports an ABI: user instructions plus system calls
 - Dynamic translators, JVM, ...
- Lightweight or OS or Namespace virtualization (multiple logical VMs that share the same OS kernel)
 - Chroot++: Isolates VMs by partitioning all objects (not just files) into namespaces
 - Linux containers and vServer, Solaris zones, FreeBSD jails
- System virtualization (whole system: OS+apps)
 - The VM supports a complete ISA: user+system instructions
 - Classic VMs, whole system emulators (and many others we discuss in next slides)

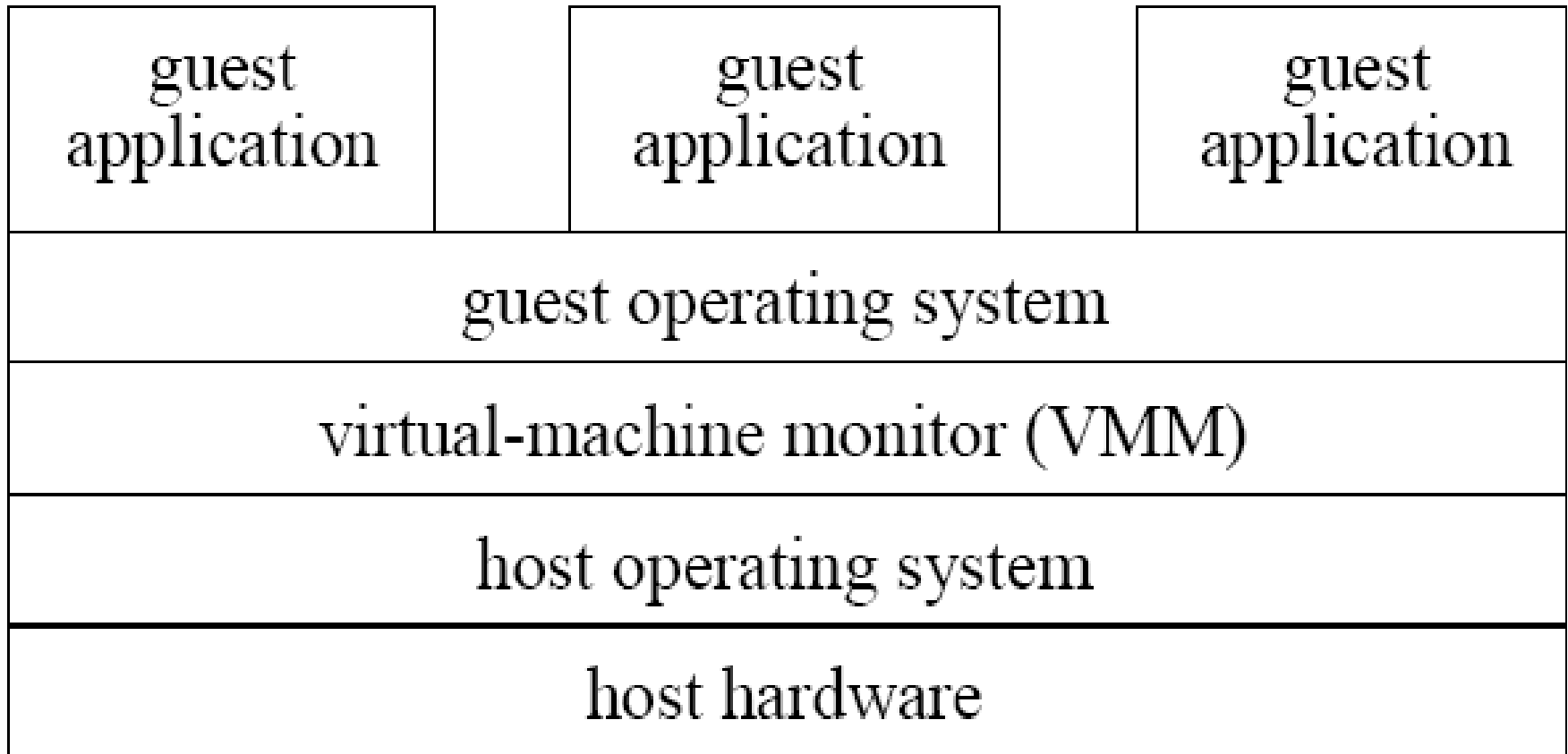
Architectures

- Type I: The VMM runs on bare hardware (“bare-metal hypervisor”)



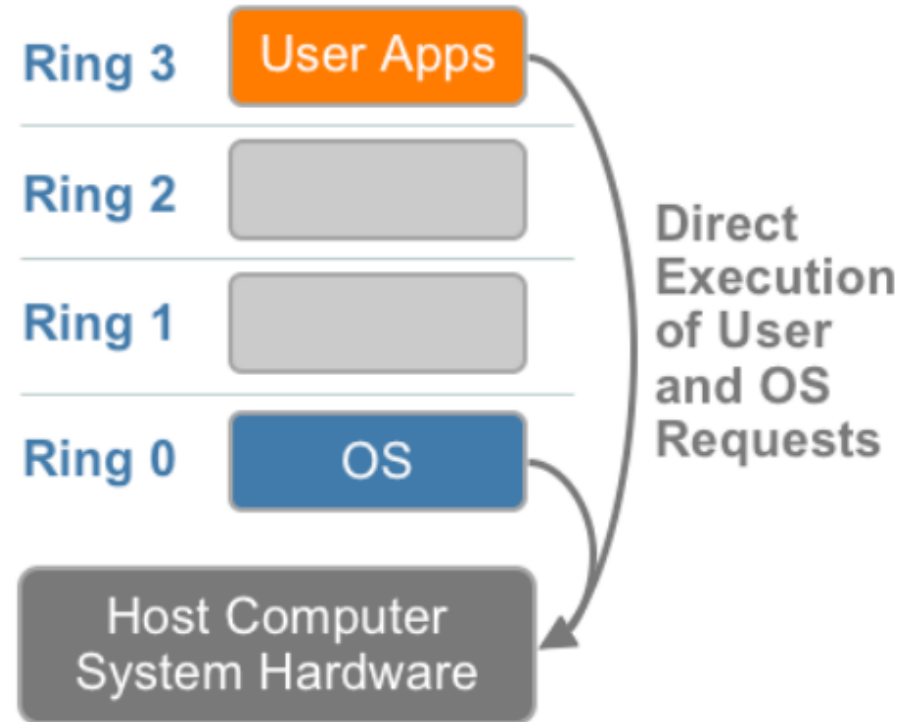
Architectures

- Type II: The VMM runs as an ordinary application inside host OS (hosted hypervisor)



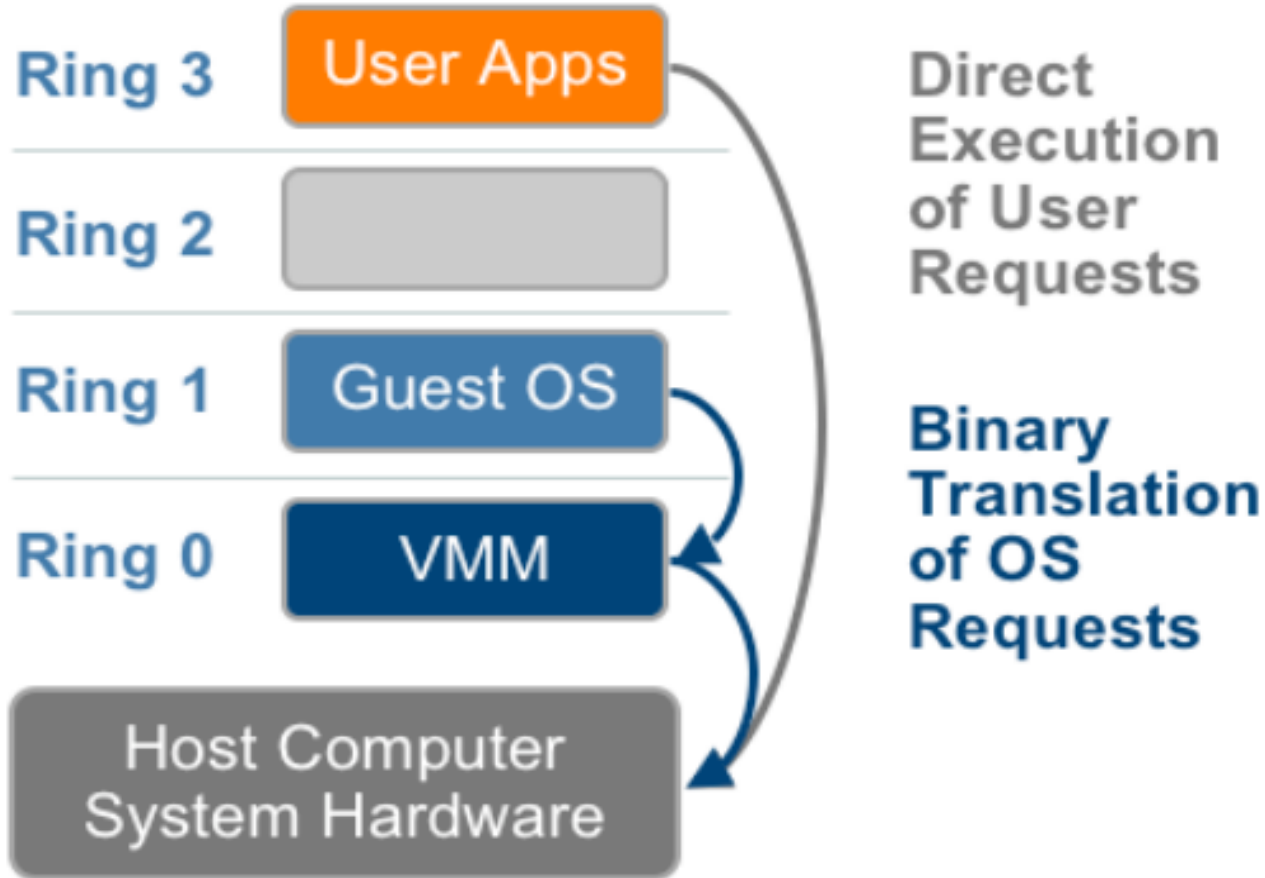
Key Issues in CPU Virtualization

- Protection levels
 - Ring 0 (most privileged)
 - Ring 3 (user mode)
- Requirement for efficient/effective virtualization
 - Privileged instructions
 - Trap if executed in user mode
 - Sensitive instructions
 - affect important “system state”
 - If privileged==sensitive, can support efficient “trap and emulate” approach
 - Virtualized execution = native execution+exception handling code that emulates privileged instructions
- For x86, not all sensitive instructions are privileged
 - Some instructions simply exhibit different behaviors in user and privileged mode



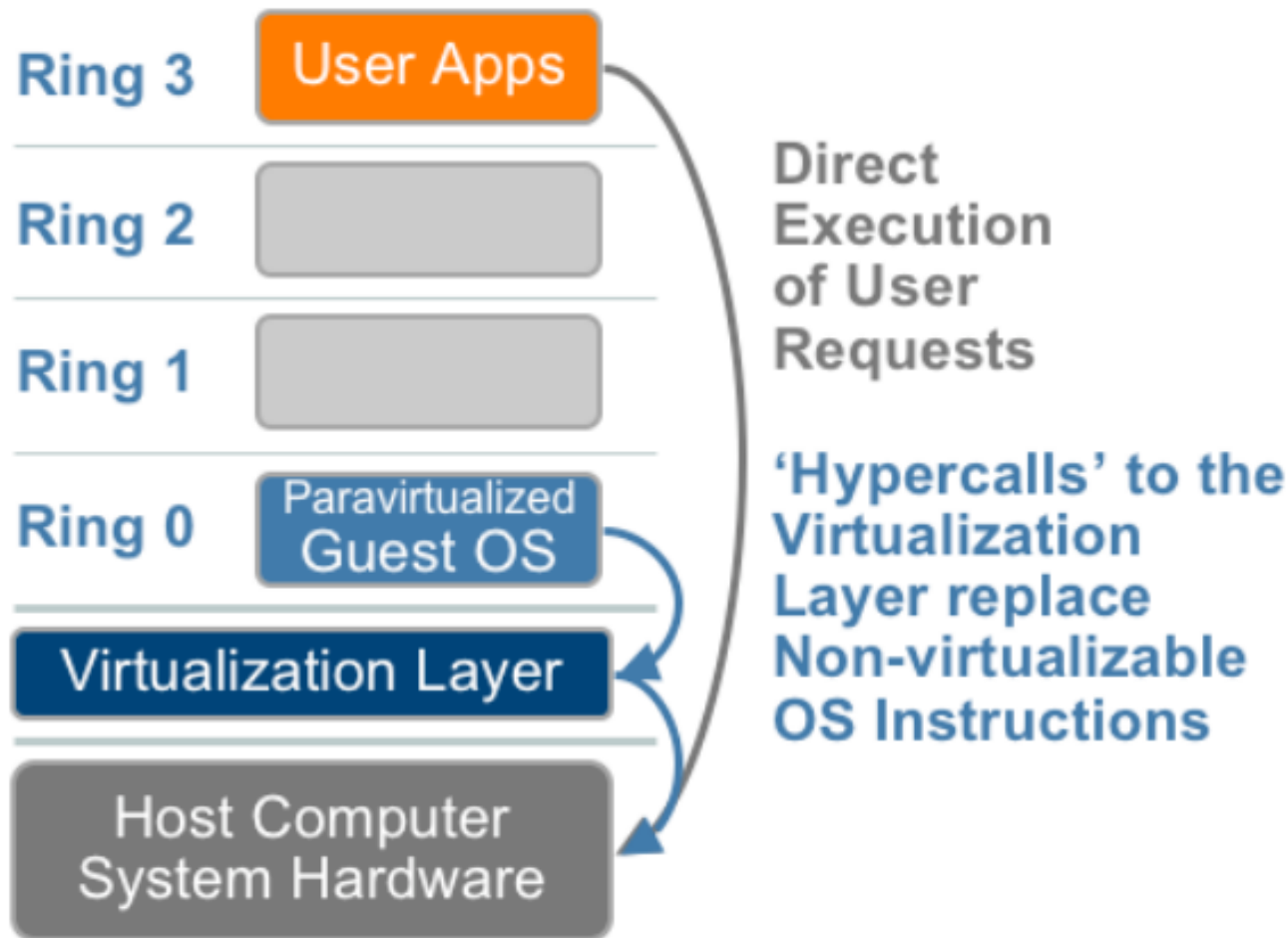
Virtualization Approaches

- Full virtualization using binary translation
 - Problem instructions translated into a sequence of instructions that achieve the intended function
 - Example: VMware, QEMU



Virtualization Approaches

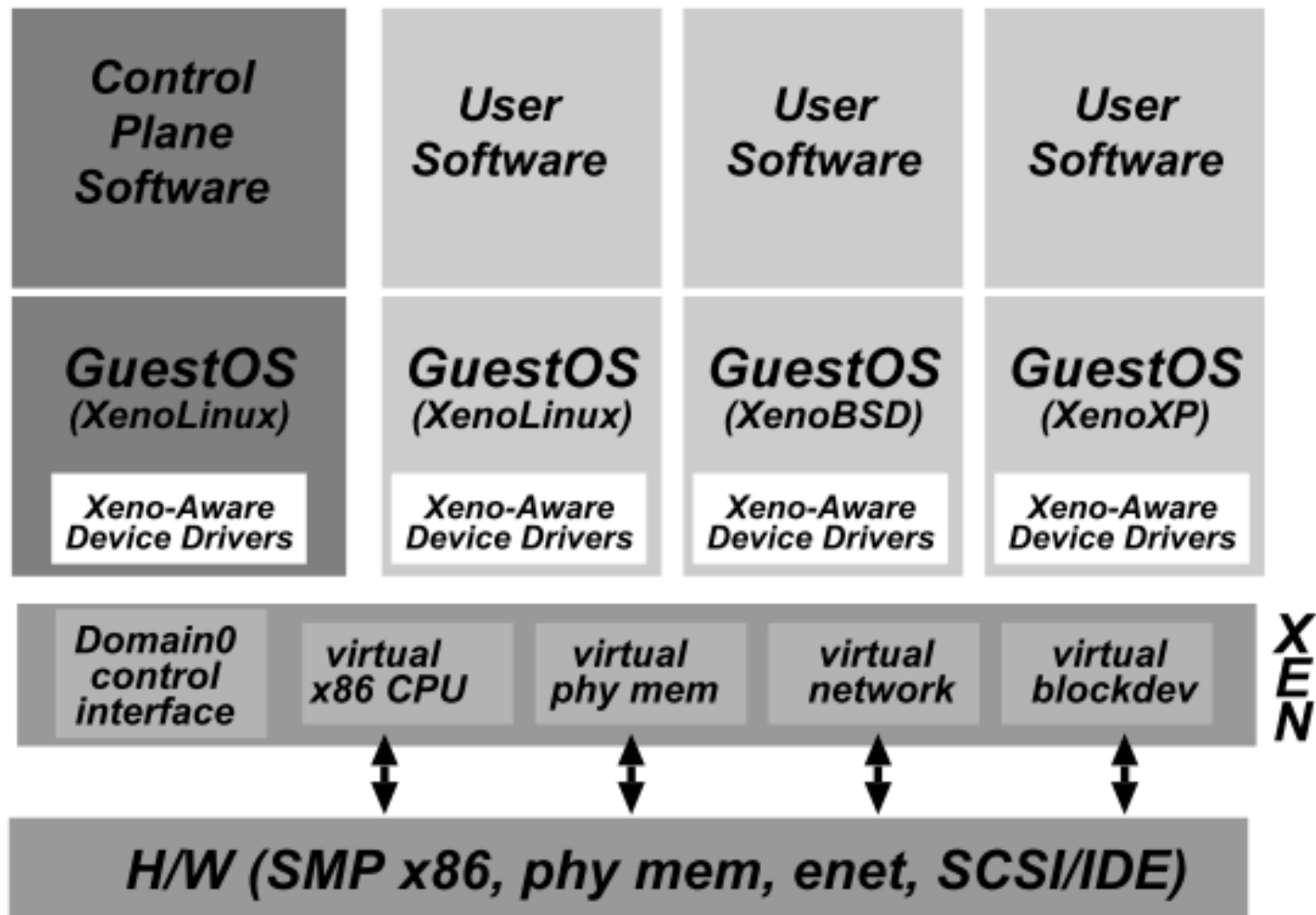
- Paravirtualization: OS modified to run on VMM
 - Example: Xen



Paravirtualization

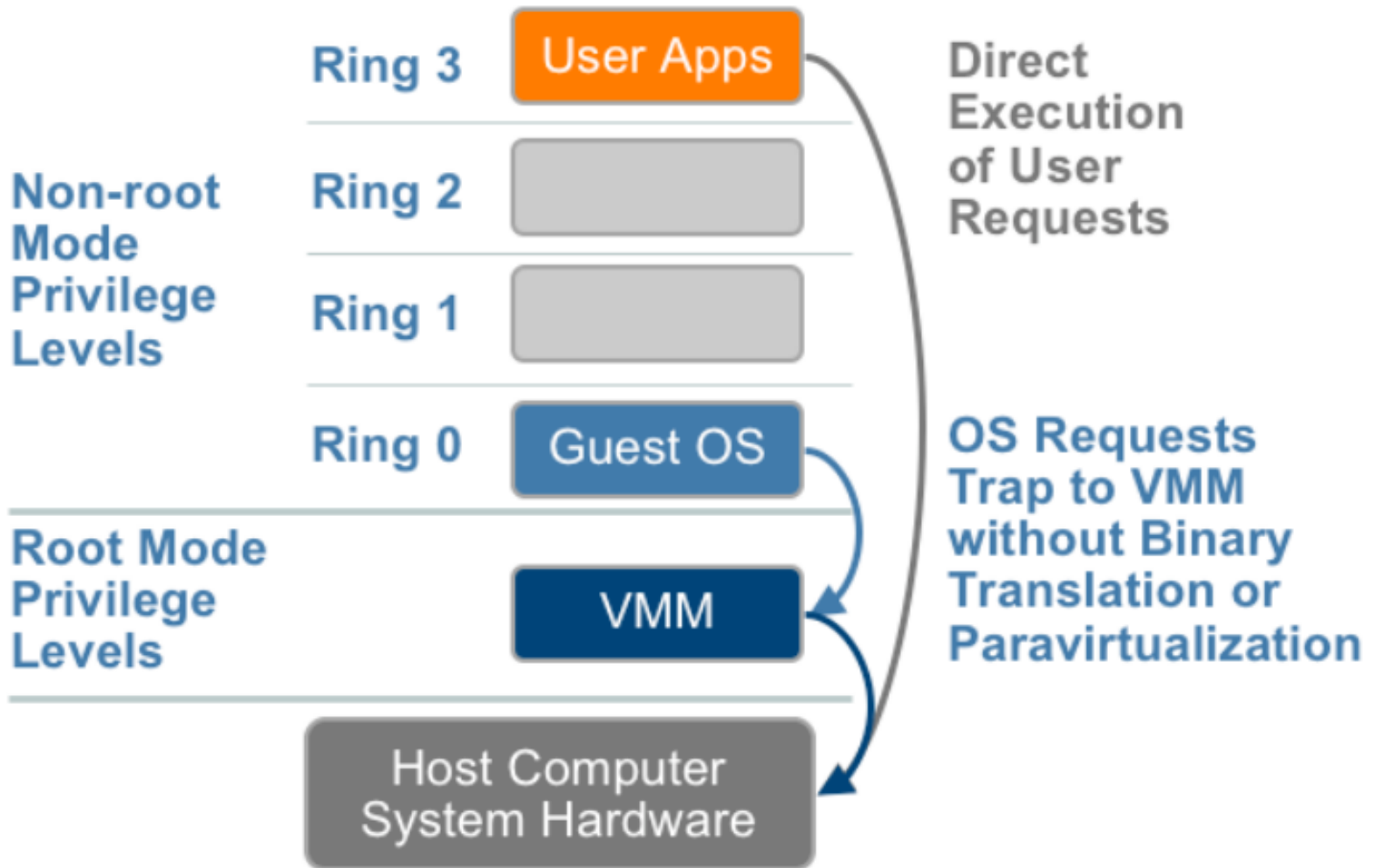
- No longer 100% interface compatible, but better performance
 - Guest OSes must be modified to use VMM's interface
 - Note that ABI is unchanged
 - Applications need not to be modified
- Guest OSes are aware of virtualization
 - privileged instructions are replaced by hypervisor calls
 - therefore, no need for binary translation

Xen and the Art of Virtualization



Virtualization Approaches

- Hardware-assisted virtualization



Hardware-assisted Virtualization

- Processor
 - AMD virtualization (AMD-V)
 - Intel virtualization (VT-x)

AMD-V: CPU virtualization

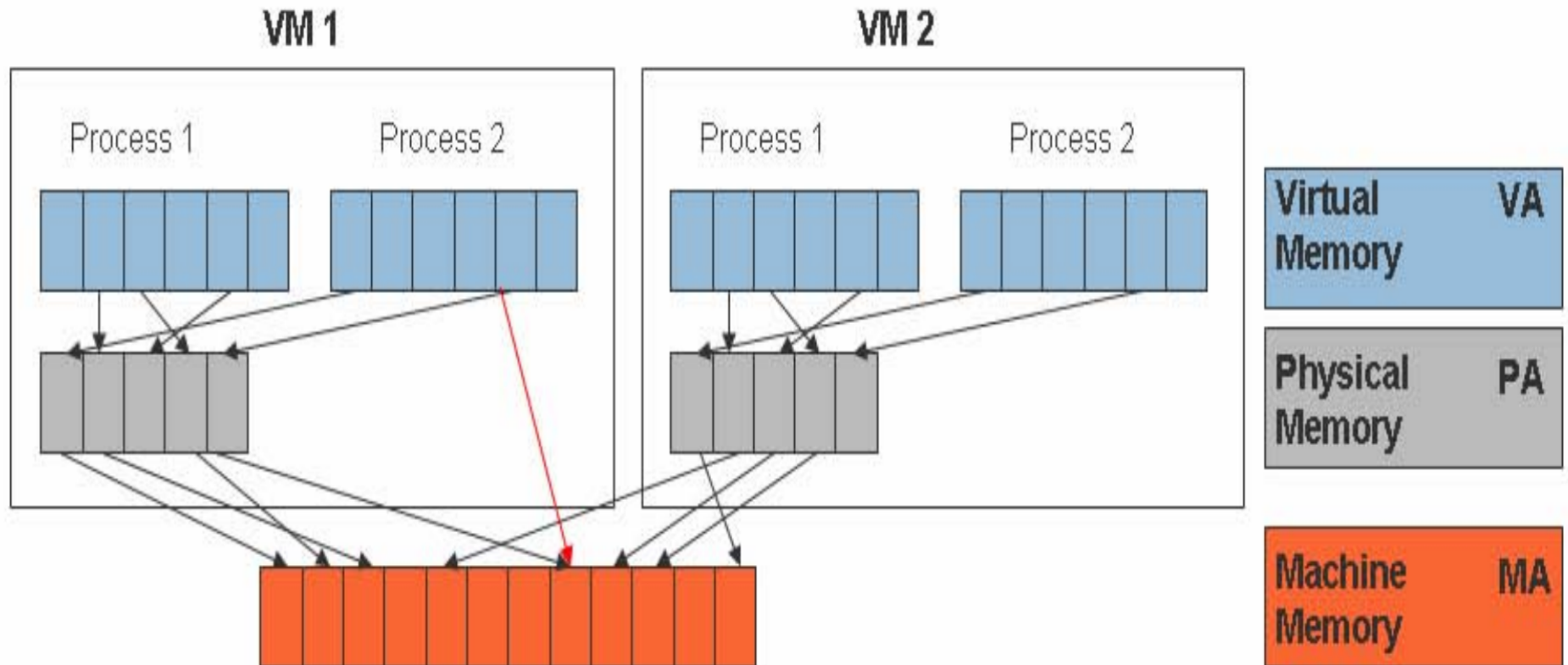
- Separates CPU execution into two modes
 - hypervisor executes in host mode
 - all VMs execute in guest mode
- Both hypervisor and VMs can execute in any of the four rings
- Hypervisor can
 - explicitly switch from host mode to guest mode
 - specify which events (e.g. interrupts) cause exist from guest mode

Memory Virtualization

- Access to MMU needs to be virtualized
 - Otherwise guest OS may directly access physical memory and/or otherwise subvert VMM
- Physical Memory is divided among multiple VMs
 - Two levels of translation
 - Guest OS: guest virtual addr → guest physical addr
 - VMM: guest physical addr → machine addr

Memory Virtualization

- Shadow page table needed to avoid 2-step translation
 - When guest attempts to update, VMM intercepts and emulate the effects on the corresponding shadow page table



AMD-V: Memory Virtualization

- CPU is aware of
 - the existence of VM
 - two-level address translation
- AMD's nested page table
 - (Intel VT-x has a similar scheme called Extended Page Table)
 - managed by VMM
 - guest physical addr -> machine addr
 - guest OS directly updates its guest page table
 - therefore, no need for a shadow page table

I/O Virtualization

- The VMM
 - intercepts a guest's I/O action
 - converts it from a virtual device action to a real device action

Security Applications

- Honeypot systems and Malware analysis
 - VM technology provides strong isolation that is necessary to run malware without undue risks
 - Strong resource isolation: CPU, memory, storage
 - Snapshot/restore features to speed up testing and recovery
- High-assurance VMs
 - On a single workstation, can run high assurance VMs that support some security functions, but may not provide general-purpose functions
 - single-purpose VM scheme facilitates stricter security policies
 - In contrast, security policies that are compatible with the range of desktop applications being used today will likely be too permissive.

Security Applications

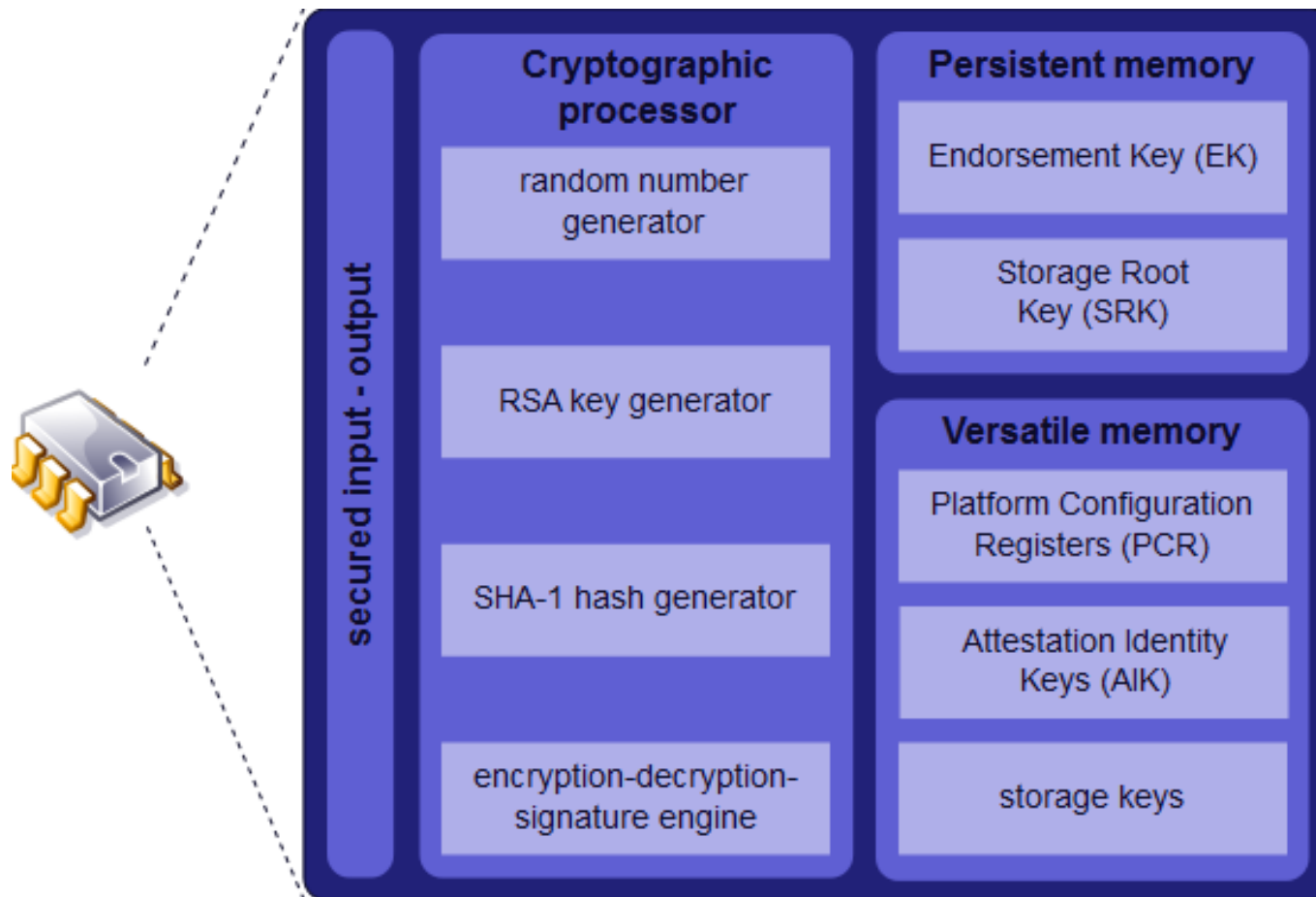
- Protection from compromised OSES
 - Modern OSES are too complex to secure
 - Malware-infested OS may subvert security software (virus and malware scanners)
 - Instead, rely on VMM
 - run malware and rootkit detection techniques in VMM
 - enforce security properties from within the VMM

Trusted Computing

- With TC, the computer will consistently behave in expected ways
 - Enforced by hardware (TPM chip) and software
- Main functionality: allow someone else to verify that only authorized code runs on a system (remote attestation)
 - verify initial booting and kernel
 - may also verify applications and various scripts
- Note: TC itself does not protect against attacks that exploit security vulnerabilities introduced by programming bugs

TPM Introduction

- The Trusted Platform Module (TPM) is a dedicated security chip



TPM functionality



- Can provide an attestation to remote parties
 - Platform Configuration Registers (PCRs) summarize the computer's software state
 - $\text{Extend}(N, V): \text{PCR}_N = \text{SHA-1}(\text{PCR}_N \parallel V)$
 - TPM provides a signature over PCR values (Quote)
 - Can provide sealed storage (Seal, Unseal)
 - Data is sealed (encrypted) providing a key and target PCR values
 - Only when TPM PCR values match target PCR values (which indicates valid system state), data can be unsealed (decrypted) using the right key