Unit - III

Virtual Machines

Virtual machines basics, Process virtual machines: Memory architecture emulation, Instruction emulation, Operating system emulation, Dynamic binary optimization, High level VN architecture, System virtual machines: Resource virtualization (Processors, Memory, Input/Output), Case Study of Intel VT-x

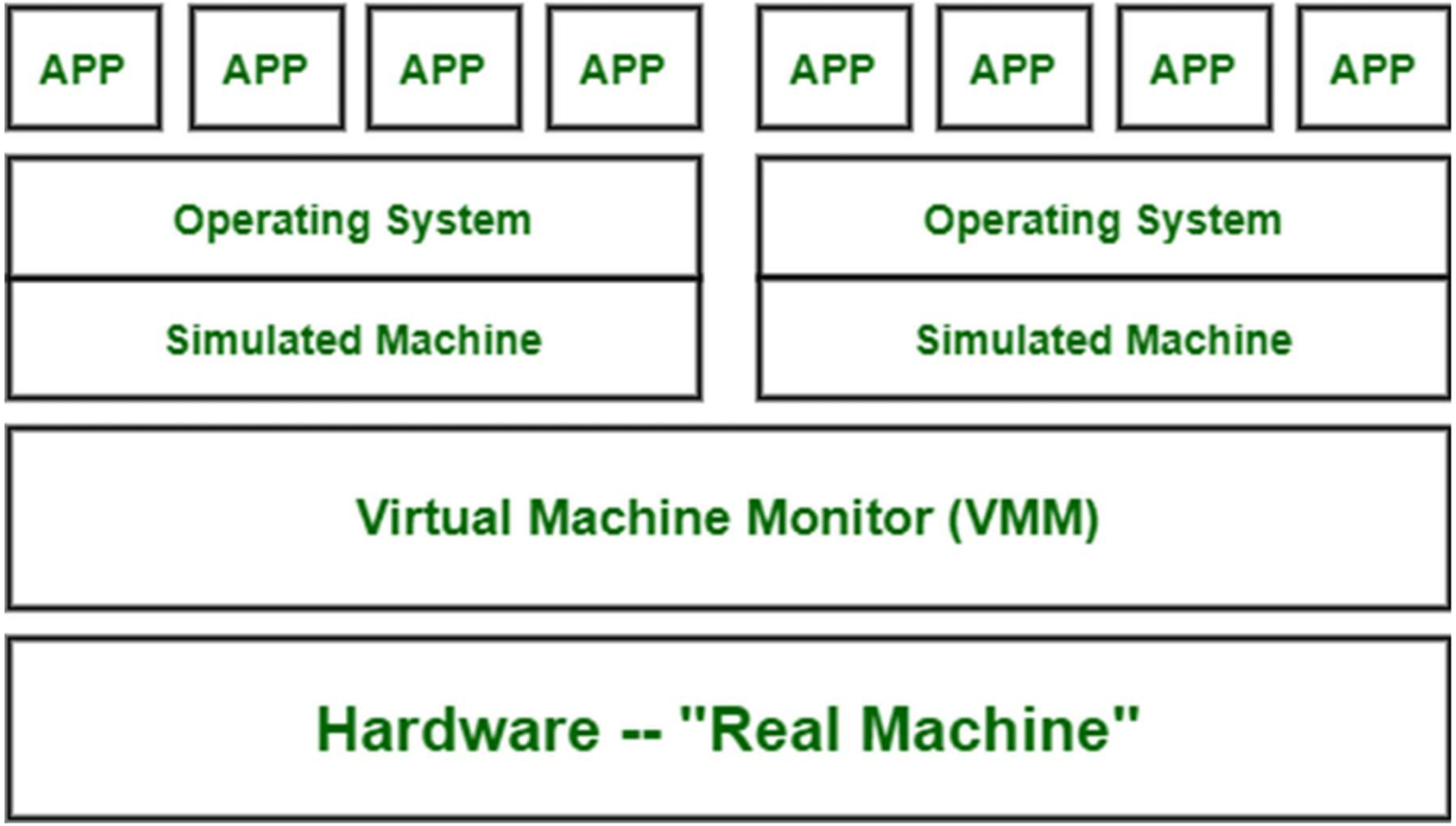
Virtual machines basics

- A virtual machine (VM) is a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system (located off- or on-premises). Software called a hypervisor separates the machine's resources from the hardware and provisions them appropriately so they can be used by the VM.
- The physical machines, equipped with a hypervisor such as Kernel-based Virtual Machine (KVM), is called the host machine, host computer, host operating system, or simply host. The many VMs that use its resources are guest machines, guest computers, guest operating systems, or simply guests. The hypervisor treats compute resources—like CPU, memory, and storage—as a pool of resources that can easily be relocated between existing guests or to new virtual machines.
- Types of Virtual Machines : Two types

System Virtual Machine

- These types of virtual machines gives us complete system platform and gives the execution of the complete virtual operating system. Just like virtual box, system virtual machine is providing an environment for an OS to be installed completely.
- We can see in below image that our hardware of Real Machine is being distributed between two simulated operating systems by Virtual machine monitor. And then some programs, processes are going on in that distributed hardware of simulated machines separately.

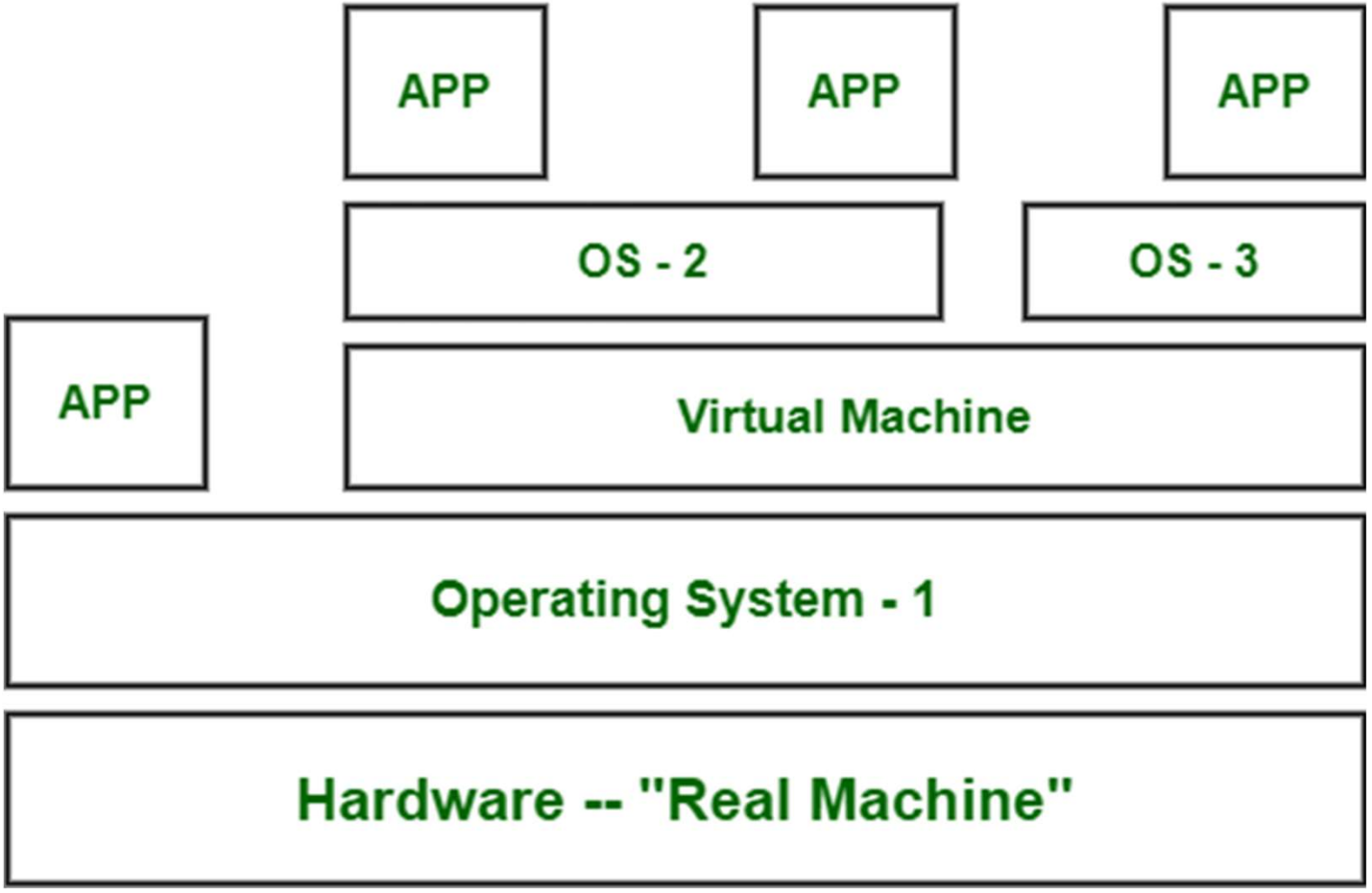
System Virtual Machine



Process virtual machines

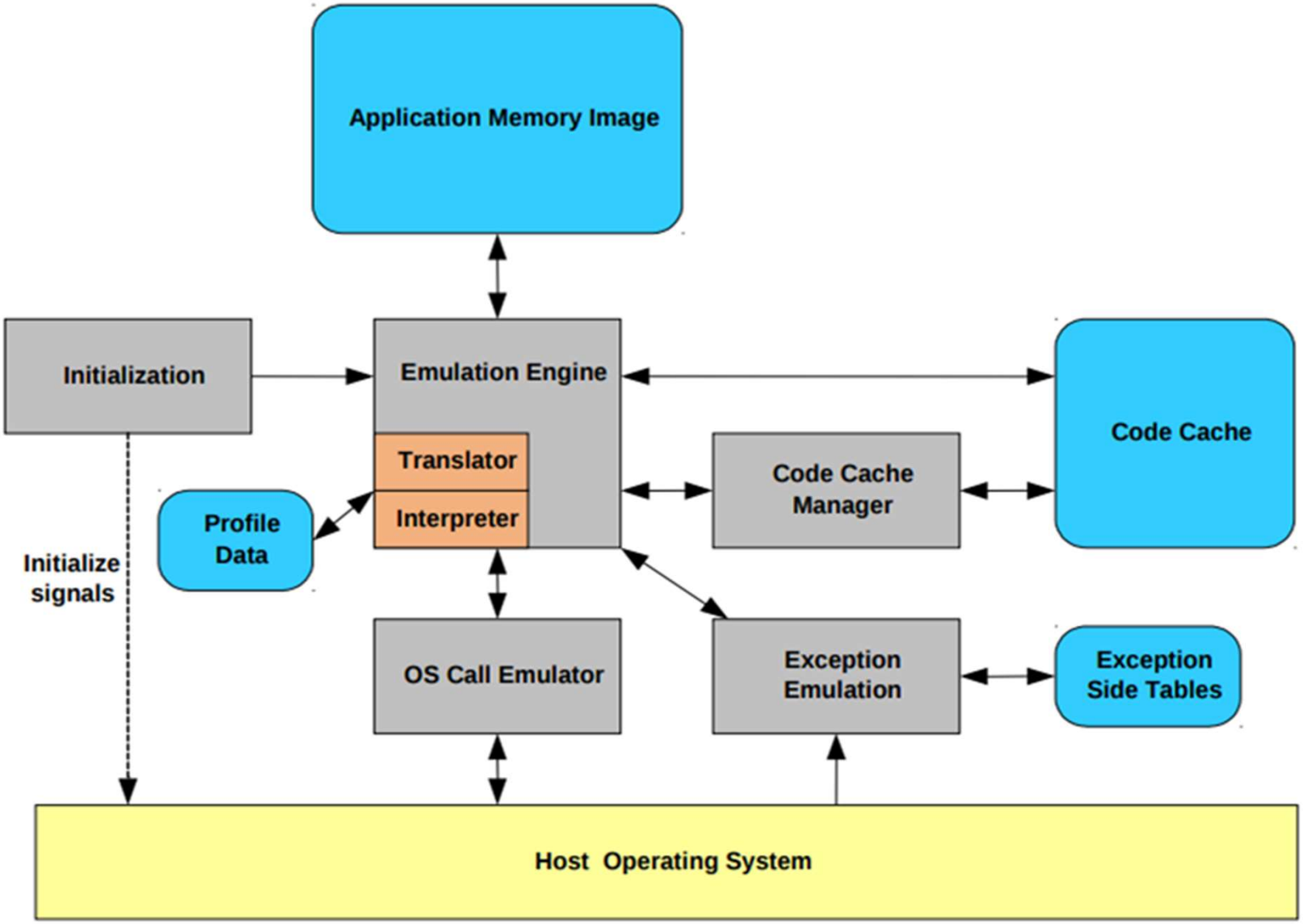
- While process virtual machines, unlike system virtual machine, does not provide us with the facility to install the virtual operating system completely.
- Rather it creates virtual environment of that OS while using some app or program and this environment will be destroyed as soon as we exit from that app.
- Like in below image, there are some apps running on main OS as well some virtual machines are created to run other apps.
- This shows that as those programs required different OS, process virtual machine provided them with that for the time being those programs are running.

Process Virtual Machine



Virtual Machine Language

- It’s type of language which can be understood by different operating systems. It is platform-independent.
- Just like to run any programming language (C, python, or java) we need specific compiler that actually converts that code into system understandable code (also known as byte code).
- The same virtual machine language works.
- If we want to use code that can be executed on different types of operating systems like (Windows, Linux, etc) then virtual machine language will be helpful.


Application Memory Image
Initialization
Emulation Engine
Translator
Interpreter
Code Cache
Code Cache Manager
Profile Data
Initialize signals
OS Call Emulator
Exception Emulation
Exception Side Tables
Host Operating System

Memory architecture emulation

- Aspects of the ABI memory architecture that need to be emulated.
  - Address space structure
    - segmented or flat
  - Access privilege types
    - combination of N, R, W, E
  - Protection / allocation granularity
    - size of the smalled block of memory that can be allocated by the OS

NONE, READ, WRITE, AND EXECUTE

Instruction emulation

Techniques for instruction emulation

– interpretation, binary translation

Start-up time (S)

– cost of translating code for emulation

– one time cost for translating code

Steady-state performance (T)

– cost of emulation

– average rate at which instructions are emulated

Operating system emulation

- A PVM emulates the function or semantics of the guest's OS calls

– not emulate individual instructions in the guest OS

- Different from instruction emulation

– given enough time, any function can be performed on the input operands to produce a result

– most ISAs perform same functions, ISA emulation is always possible

– with OS, it is possible that providing some host function is impossible, operation semantic mismatch

Different source and target OS

– semantic translation of mapping required
– may be difficult or impossible
– ad-hoc process on a case-by-case basis

Same source and target OS

– emulate the guest calling convention
– guest system call jumps to runtime, which provides wrapper code

High level VN architecture

- Virtual networking enables communication between multiple computers, virtual machines (VMs), virtual servers, or other devices across different office and data center locations. While physical networking connects computers through cabling and other hardware, virtual networking extends these capabilities by using software management to connect computers and servers over the Internet. It uses virtualized versions of traditional network tools, like switches and network adapters, allowing for more efficient routing and easier network configuration changes.
- Virtual networking enables devices across many locations to function with the same capabilities as a traditional physical network. This allows for data centers to stretch across different physical locations, and gives network administrators new and more efficient options, like the ability to easily modify the network as needs change, without having to switch out or buy more hardware; greater flexibility in provisioning the network to specific needs and applications; and the capacity to move workloads across the network infrastructure without compromising service, security, and availability.

How does virtual networking work?

- A virtual network connects virtual machines and devices, no matter their location, using software. In a physical network, layer 2 and 3 functions of the OSI model happen within physical switches and routers. Plus, physical network interface cards (NIC) and network adapters are used to connect computers and servers to the network. Virtual networking shifts these and other activities to software. A software application, called a virtual switch or vSwitch, controls and directs communication between the existing physical network and virtual parts of the network, like virtual machines. And a virtual network adapter allows computers and VMs to connect to a network, including making it possible for all the machines on a local area network (LAN) to connect to a larger network.
- In a physical network, LANs are created to connect multiple devices to shared resources, like network storage, usually through Ethernet cables or Wi-Fi. But virtual networking creates the possibility for virtual LANs (VLANs), where the grouping is configured through software. This means that computers connected to different network switches can behave as if they're all connected to the same one, and, conversely, computers that share cabling can be kept on separate networks, rather than physically connecting machines using cabling equipment and hardware.

Advantages of virtual networking

- Virtual networking delivers a variety of business benefits, from lowering capital expenditures and maintenance costs to easily segmenting networks. Specifically, a virtual network:
- Streamlines the amount of network hardware (cabling, switches, etc.) through shifting many functions to software
- Reduces the cost and complexity of managing network hardware and software through centralized control
- Offers more flexible options for network routing structure and configuration, including easier options for segmenting and subdividing the network
- Improves control over network traffic with more fine-grained options, like configuring firewalls at the virtual NIC level
- Increases IT productivity through remote and automated service activation and performance testing
- Boosts business scalability and flexibility by enabling virtual upgrades, automated configuring, and modular changes to network appliances and applications

Resource virtualization (Processors, Memory, Input/Output)

- To support virtualization, processors such as the x86 employ a special running mode and instructions, known as hardware-assisted virtualization.
- In this way, the VMM and guest OS run in different modes and all sensitive instructions of the guest OS and its applications are trapped in the VMM.
- To save processor states, mode switching is completed by hardware. For the x86 architecture, Intel and AMD have proprietary technologies for hardware-assisted virtualization.

Hardware Support for Virtualization

- Modern operating systems and processors permit multiple processes to run simultaneously.
- If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash.
- Therefore, all processors have at least two modes, user mode and supervisor mode, to ensure controlled access of critical hardware. Instructions running in supervisor mode are called privileged instructions.
- Other instructions are unprivileged instructions. In a virtualized environment, it is more difficult to make OSes and applications run correctly because there are more layers in the machine stack.

- The VMware Workstation is a VM software suite for x86 and x86-64 computers. This software suite allows users to set up multiple x86 and x86-64 virtual computers and to use one or more of these VMs simultaneously with the host operating system. The VMware Workstation assumes the host-based virtualization. Xen is a hypervisor for use in IA-32, x86-64, Itanium, and PowerPC 970 hosts. Actually, Xen modifies Linux as the lowest and most privileged layer, or a hypervisor.
- One or more guest OS can run on top of the hypervisor. KVM (Kernel-based Virtual Machine) is a Linux kernel virtualization infrastructure. KVM can support hardware-assisted virtualization and paravirtualization by using the Intel VT-x or AMD-v and VirtIO framework, respectively. The VirtIO framework includes a paravirtual Ethernet card, a disk I/O controller, a balloon device for adjusting guest memory usage, and a VGA graphics interface using VMware drivers.

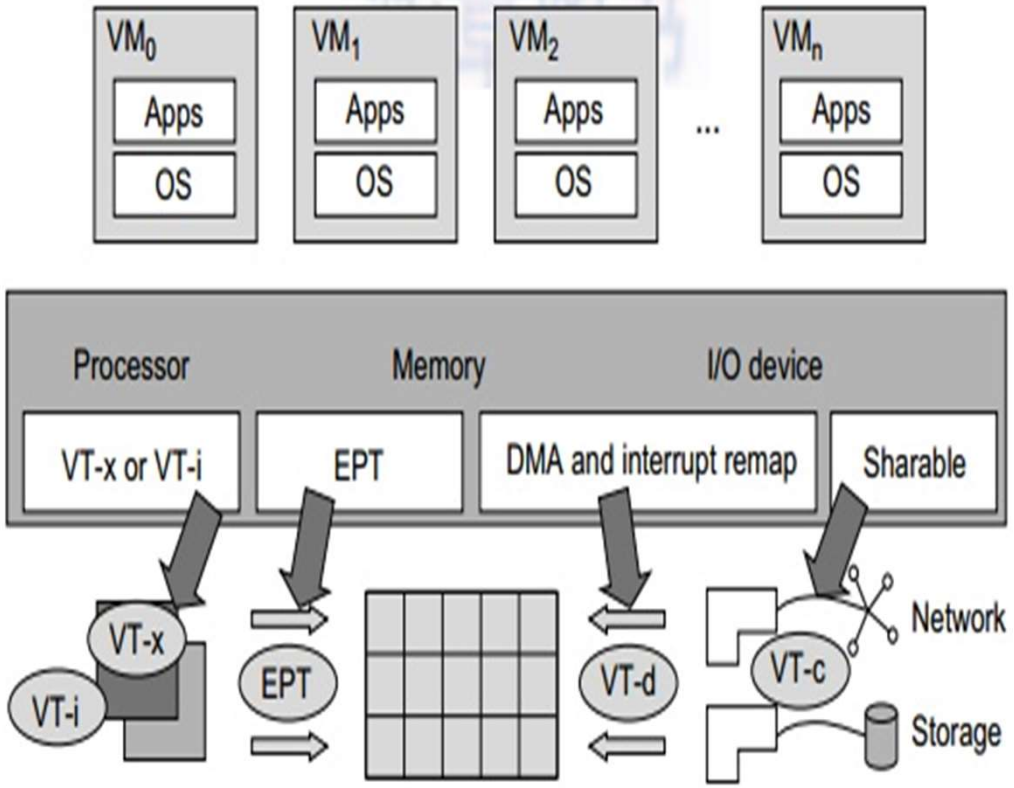


FIGURE 3.10

Intel hardware support for virtualization of processor, memory, and I/O devices.

Since software-based virtualization techniques are complicated and incur performance overhead, Intel provides a hardware-assist technique to make virtualization easy and improve performance. Figure 3.10 provides an overview of Intel's full virtualization techniques. For processor virtualization, Intel offers the VT-x or VT-i technique. VT-x adds a privileged mode (VMX Root Mode) and some instructions to processors. This enhancement traps all sensitive instructions in the VMM automatically. For memory virtualization, Intel offers the EPT, which translates the virtual address to the machine's physical addresses to improve performance. For I/O virtualization, Intel implements VT-d and VT-c to support this.

CPU Virtualization

- A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode.
- Thus, unprivileged instructions of VMs run directly on the host machine for higher efficiency.
- Other critical instructions should be handled carefully for correctness and stability.
- The critical instructions are divided into three categories: privileged instructions, control-sensitive instructions, and behavior-sensitive instructions.
- Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode.
- Control-sensitive instructions attempt to change the configuration of resources used.
- Behavior-sensitive instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

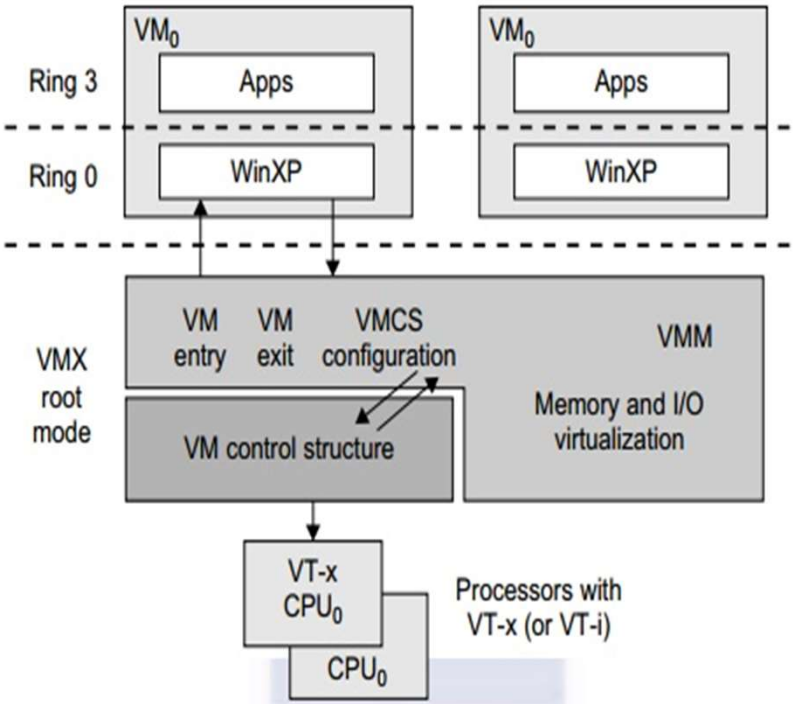


FIGURE 3.11

Intel hardware-assisted CPU virtualization.

Although x86 processors are not virtualizable primarily, great effort is taken to virtualize them. They are used widely in comparing RISC processors that the bulk of x86-based legacy systems cannot discard easily. Virtualization of x86 processors is detailed in the following sections. Intel's VT-x technology is an example of hardware-assisted virtualization, as shown in Figure 3.11. Intel calls the privilege level of x86 processors the VMX Root Mode. In order to control the start and stop of a VM and allocate a memory page to maintain the CPU state for VMs, a set of additional instructions is added. At the time of this writing, Xen, VMware, and the Microsoft Virtual PC all implement their hypervisors by using the VT-x technology.

Memory Virtualization

- Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems.
- In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory.
- All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance.
- However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs.

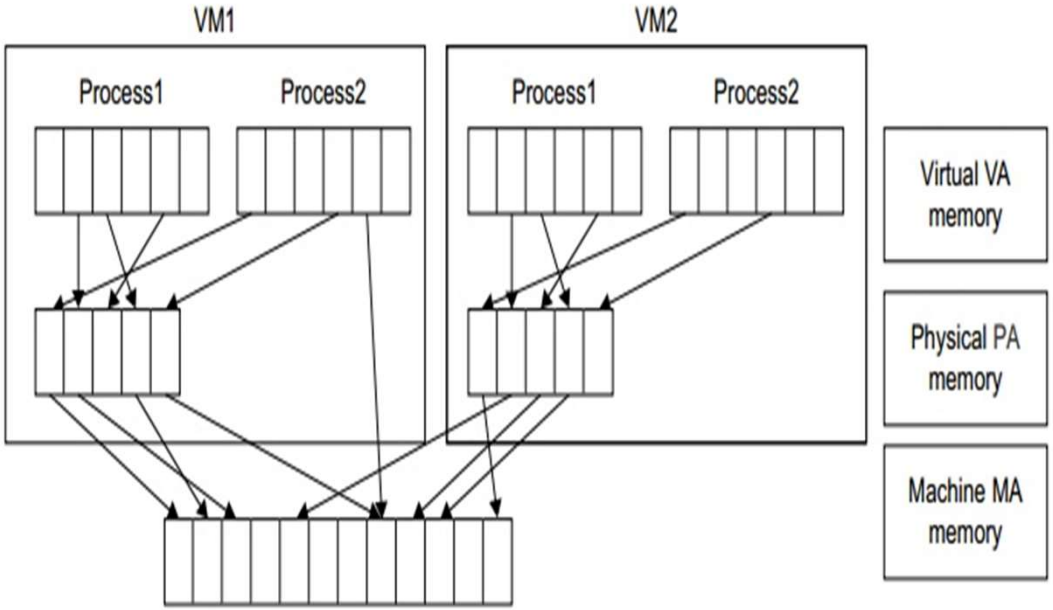


FIGURE 3.12

Two-level memory mapping procedure.

That means a two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory. Furthermore, MMU virtualization should be supported, which is transparent to the guest OS. The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs. But the guest OS cannot directly access the actual machine memory. The VMM is responsible for mapping the guest physical memory to the actual machine memory. Figure 3.12 shows the two-level memory mapping procedure.

I/O Virtualization

- I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware.
- At the time of this writing, there are three ways to implement I/O virtualization: full device emulation, para-virtualization, and direct I/O.
- Full device emulation is the first approach for I/O virtualization. Generally, this approach emulates well-known, real-world devices.

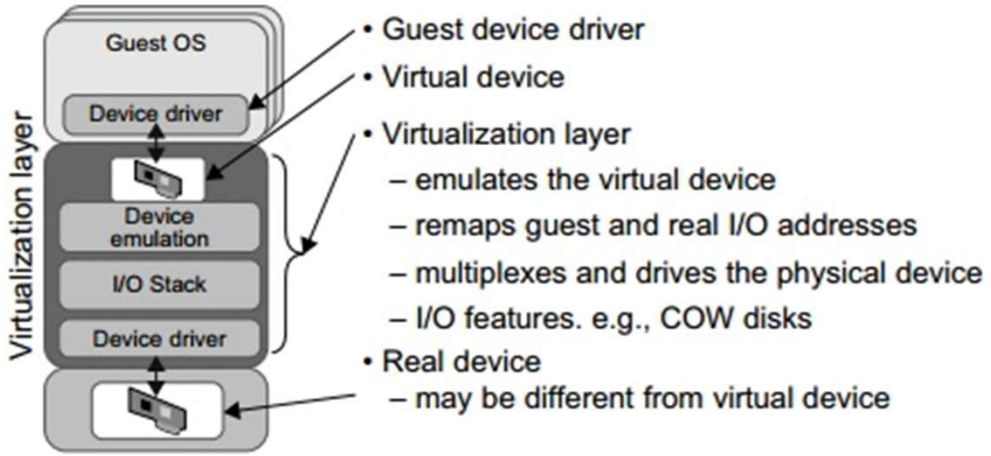


FIGURE 3.14

Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices into the virtual devices for the guest device driver to use.

All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device. The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices. The full device emulation approach is shown in Figure 3.14.

Assignment 3

- Evaluate Virtual machines basics with suitable example.
- Evaluate Process virtual machines with suitable example.
- Evaluate System virtual machines with suitable example.
- Case study of AWS services: Redshift, AppFlow, Gluedata Quality, Athena for Apachespark, Quicksight, Kinesis Data Streams, MSK, Security Lake, Open Search Serverless, Eventbridge pipes, Wickr, Simspace Weaver, Lambda snapstart, ENA Express, EC2 instance (C7gn, R7iz, Hpc7g, Inf2), Connect, EKS Clusters, Aurora, RDS application composer, Code catalyst, Gamelift, Omics, Sagemaker, Code Whisperer, Cloudwatch, VPS Lattice Inspector, EFS, IoT Twinmaker, EMR, Amplify Studio.

Submission link: https://forms.gle/f8vGUapeDMowSL7n9

Presentation III (Choose any one topic)

- Case study on Process virtual machines
- Case study on System virtual machines

Submission link: https://forms.gle/f8vGUapeDMowSL7n9

Lab 3

To simulate a cloud scenario and run a scheduling algorithm in CloudSim.

Submission link: https://forms.gle/f8vGUapeDMowSL7n9