UNIT III

Cloud Resource virtualization: Virtualization, layering and virtualization, virtual machine monitors, virtual machines, virtualization- full and para, performance and security isolation, hardware support for virtualization, Case Study: Xen, vBlades, Cloud Resource Management and Scheduling: Policies and Mechanisms, Applications of control theory to task scheduling, Stability of a two-level resource allocation architecture, feedback control based on dynamic thresholds, coordination, resource bundling, scheduling algorithms, fair queuing, start time fair queuing, cloud scheduling subject to deadlines, Scheduling Map Reduce applications, Resource management and dynamic application scaling.

Cloud Resource virtualization:

Virtualization:

Virtualization is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources". Virtualization is a technique, which allows to share a single physical instance of a resource or an application among multiple customers and organizations. It does by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded.

- Virtualization is a basic principle of cloud computing – that simplifies some of the resource management tasks.
- Users can run multiple operating systems such as Windows, Linux, on a single physical machine at the same time.
- Virtual machine software can run programs and operating systems, store data, connect to networks, and do other computing functions, and requires maintenance such as updates and system monitoring.

Virtualization simulates the interface to a physical object by:

1. ***Multiplexing.*** Create multiple virtual objects from one instance of a physical object. For example, a processor is multiplexed among a number of processes or threads.
2. ***Aggregation.*** Create one virtual object from multiple physical objects. For example, a number of physical disks are aggregated into a RAID disk.
3. ***Emulation.*** Construct a virtual object from a different type of physical object. For example, a physical disk emulates a random access memory.

4. ***Multiplexing and emulation.*** Examples: Virtual memory with paging multiplexes real memory and disk, and a Virtual address emulates a real address.

Types of Virtualization:

1. Hardware Virtualization.
2. Operating system Virtualization.
3. Server Virtualization.
4. Storage Virtualization.

1) Hardware Virtualization:

When the virtual machine software or virtual machine manager *(VMM) is directly installed on the hardware system* is known as hardware virtualization.

The main job of hypervisor is to control and monitoring the processor, memory and other hardware resources.

After virtualization of hardware system we can install different operating system on it and run different applications on those OS.

Usage:

Hardware virtualization is mainly done for the server platforms, because controlling virtual machines is much easier than controlling a physical server.

2) Operating System Virtualization:

When the virtual machine software or virtual machine manager *(VMM) is installed on the Host operating system* instead of directly on the hardware system is known as operating system virtualization.

Usage: Operating System Virtualization is mainly used for testing the applications on different platforms of OS.

3) Server Virtualization:

When the virtual machine software or virtual machine manager (VMM) is directly installed on the Server system is known as server virtualization.

Usage: Server virtualization is done because a single physical server can be divided into multiple servers on the demand basis and for balancing the load.

4) Storage Virtualization:

Storage virtualization is the process of grouping the physical storage from multiple network storage devices so that it looks like a single storage device.

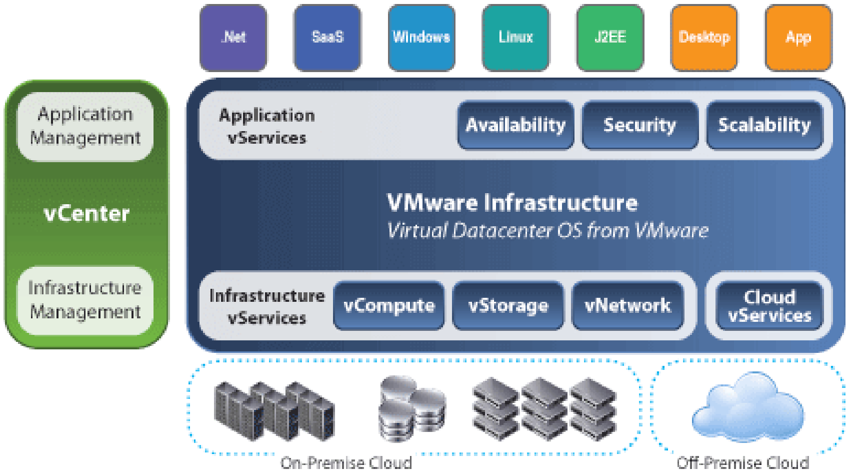
Storage virtualization is also implemented by using software applications.

Usage: Storage virtualization is mainly done for back-up and recovery purposes.

Virtualization is a critical aspect of cloud computing, equally important to the providers and consumers of cloud services, and plays an important role in:

1. System security because it allows isolation of services running on the same hardware.
2. Performance and reliability because it allows applications to migrate from one platform to another.
3. The development and management of services offered by a provider.
4. Performance isolation.

User convenience is a major advantage of a VM architecture over a traditional operating system. For example, a user of the Amazon Web Services (AWS) could submit an Amazon Machine Image (AMI) containing the applications, libraries, data, and associated configuration settings. The user could choose the operating system for the application, then start, terminate, and monitor as many instances of the AMI as needed, using the Web Service APIs and the performance monitoring and management tools provided by the AWS.



The **main usage of Virtualization Technology** is to provide the applications with the standard versions to their cloud users, suppose if the next version of that application is released, then cloud provider has to provide the latest version to their cloud users and practically it is possible because it is more expensive.

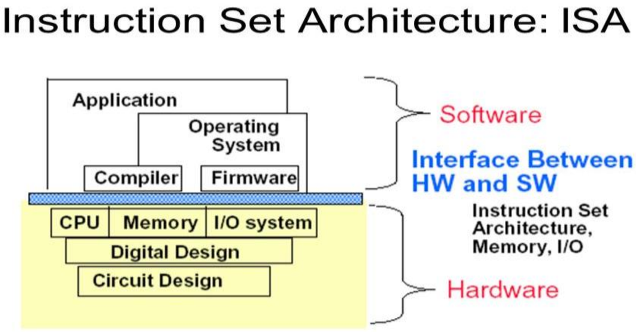
Layering and virtualization

A common approach to managing system complexity is to identify a set of layers with well-defined interfaces among them.

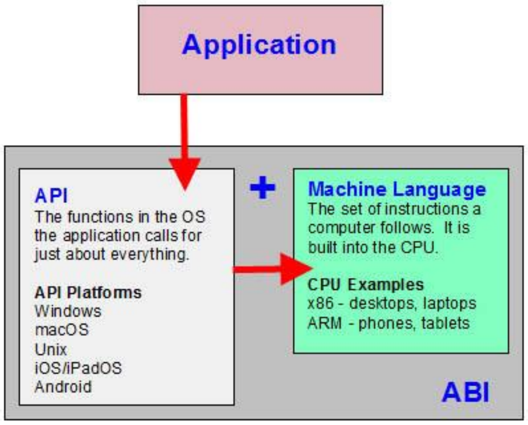
- The interfaces separate different levels of abstraction. Layering minimizes the interactions among the subsystems and simplifies the description of the subsystems.
- Each subsystem is abstracted through its interfaces with the other subsystems. Thus, we are able to design, implement, and modify the individual subsystems independently.

Interfaces:

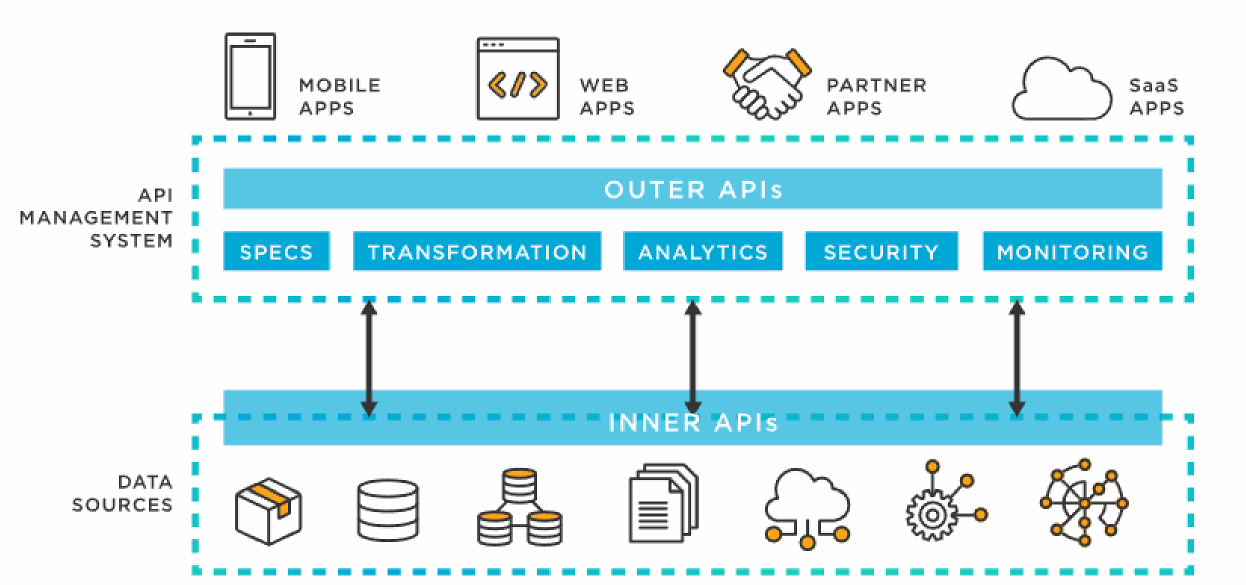
Instruction Set Architecture (ISA) – at the boundary between hardware and software. The instruction set architecture (ISA) defines a processor's set of instructions. For example, the Intel architecture is represented by the x86-32 and x86-64 instruction sets for systems supporting 32-bit addressing and 64-bit addressing, respectively. The hardware supports two execution modes, a privileged, or kernel, mode and a user mode.

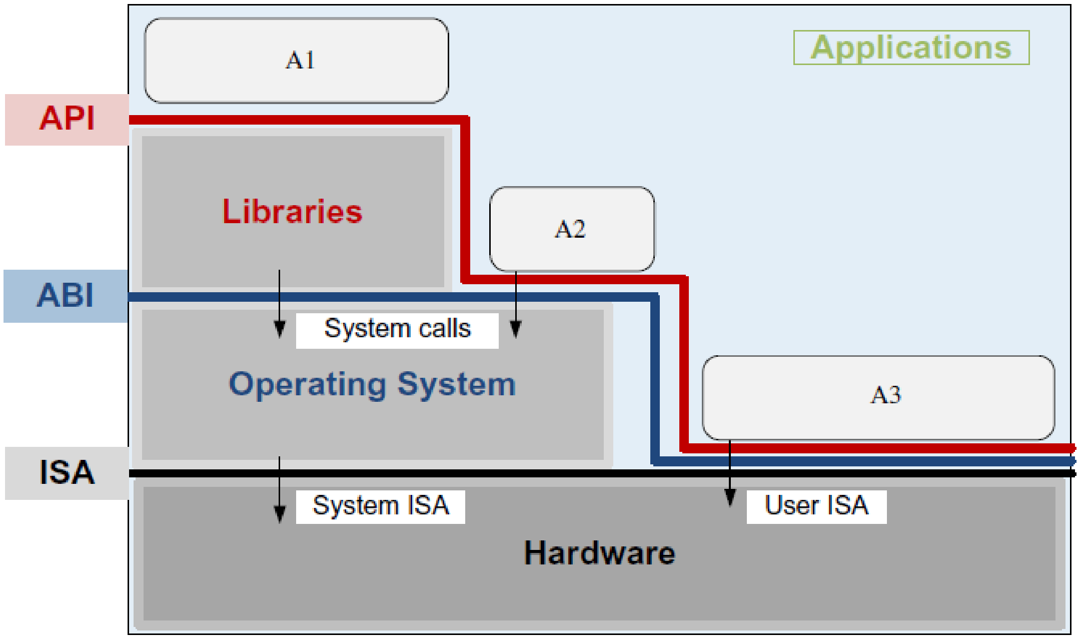


Application Binary Interface (ABI) – allows the ensemble consisting of the application and the library modules to access the hardware; the ABI does not include privileged system instructions, instead it invokes system calls.

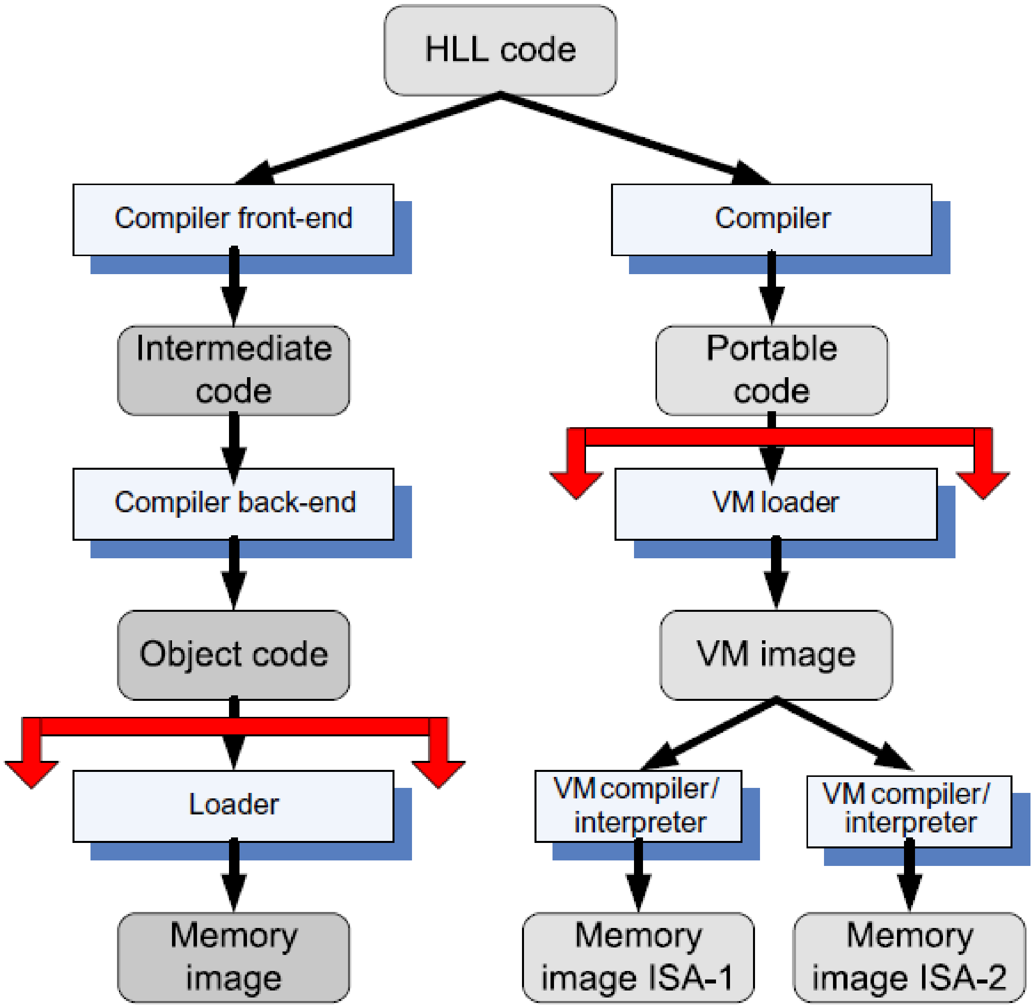

Application
API
The functions in the OS the application calls for just about everything.
API Platforms
Windows
macOS
Unix
iOS/iPadOS
Android
+
Machine Language
The set of instructions a computer follows. It is built into the CPU.
CPU Examples
x86 - desktops, laptops
ARM - phones, tablets
ABI

Application Program Interface (API) - defines the set of instructions the hardware was designed to execute and gives the application access to the ISA; it includes HLL library calls which often invoke system calls.

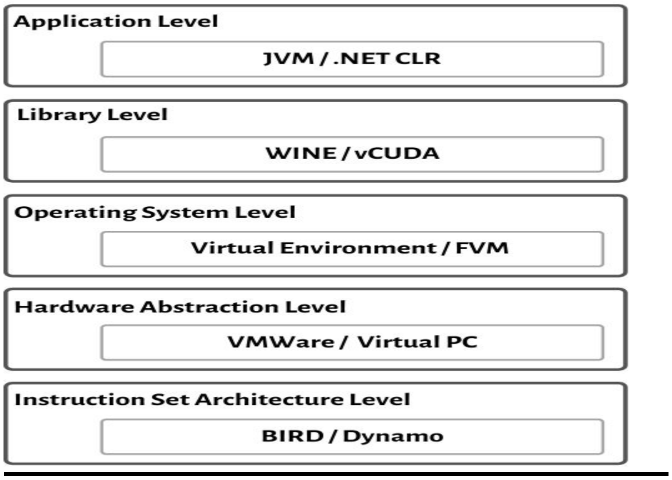




Such code cannot run on a computer with a different ISA or on computers with the same ISA but different operating systems. However, it is possible to compile an HLL program for a VM environment, where portable code is produced and distributed and then converted by binary translators to the ISA of the host system. A dynamic binary translation converts blocks of guest instructions from the portable code to the host instruction and leads to a significant performance improvement as such blocks are cached and reused.



High-level language (HLL) code can be translated for a specific architecture and operating system. HLL code can also be compiled into portable code and then the portable code translated for systems with different ISAs. The code that is shared/distributed is the object code in the first case and the portable code in the second case.



Virtual machine monitors

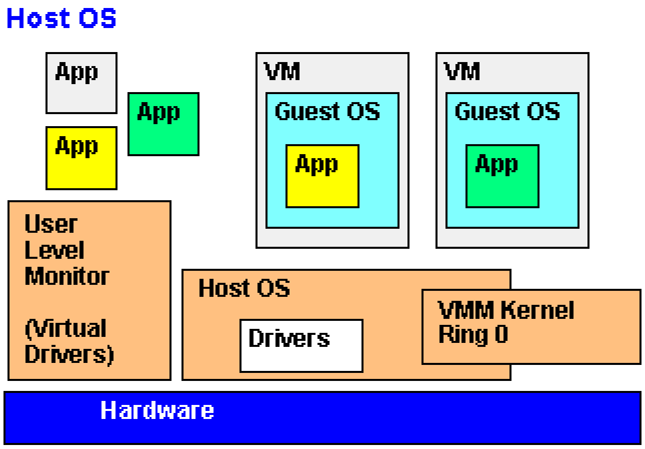
A virtual machine monitor (VMM), also called a hypervisor, is the software that securely partitions the resources of a computer system into one or more virtual machines.

- A guest operating system is an operating system that runs under the control of a VMM rather than directly on the hardware.
- The VMM runs in kernel mode, whereas a guest OS runs in user mode. Sometimes the hardware supports a third mode of execution for the guest OS.

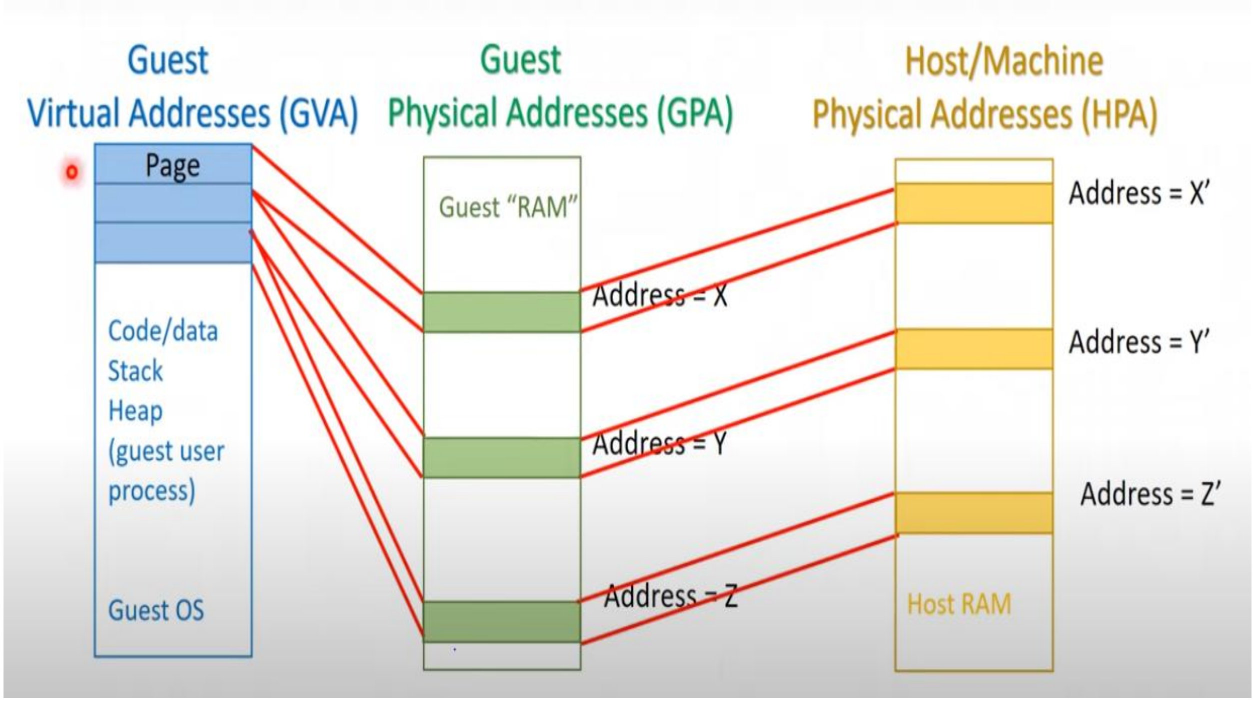
VMMs allow several operating systems to run concurrently on a single hardware platform; at the same time, VMMs enforce isolation among these systems, thus enhancing security. A VMM controls how the guest operating system uses the hardware resources. The events occurring in one VM do not affect any other VM running under the same VMM. At the same time, the VMM enables:

- Multiple services to share the same platform.
- The movement of a server from one platform to another, the so-called live migration.
- System modification while maintaining backward compatibility with the original system.

When a guest OS attempts to execute a privileged instruction, the VMM traps the operation and enforces the correctness and safety of the operation. The VMM guarantees the isolation of the individual VMs, and thus ensures security and encapsulation, a major concern in cloud computing.

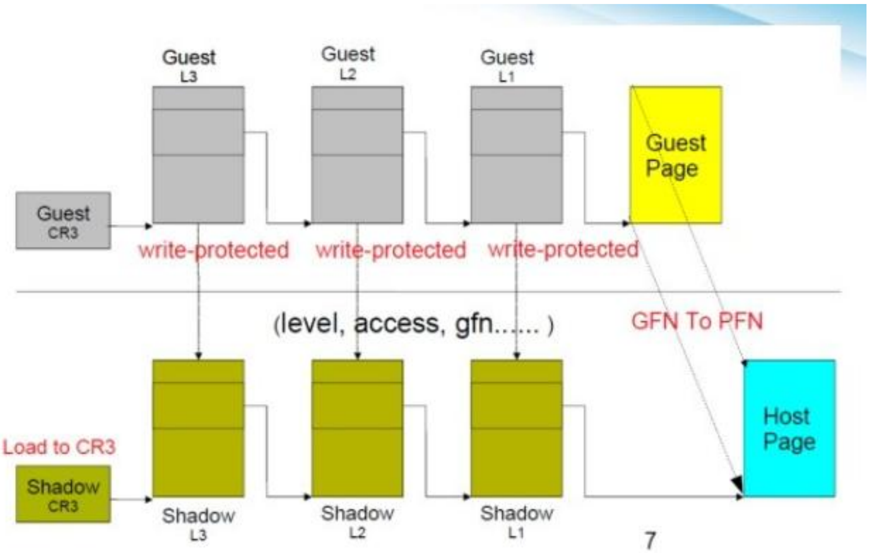


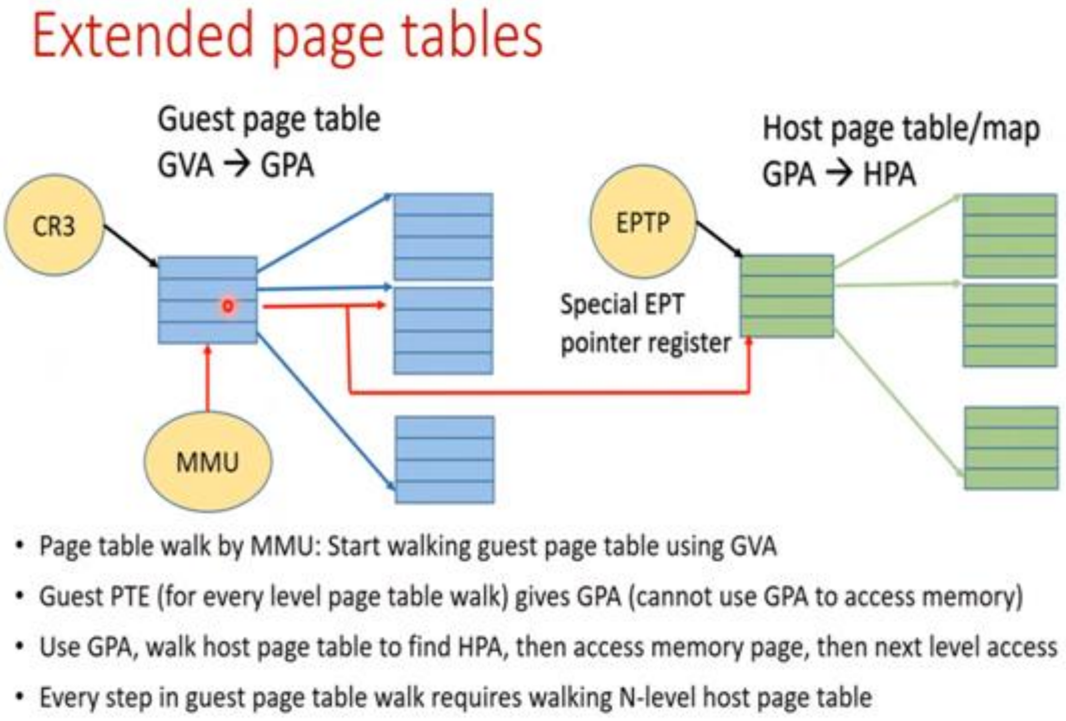
A VMM virtualizes the CPU and memory. For example, the VMM traps interrupts and dispatches them to the individual guest operating systems. If a guest OS disables interrupts, the VMM buffers such interrupts until the guest OS enables them.



The VMM maintains a shadow page table for each guest OS and replicates any modification made by the guest OS in its own shadow page table. This shadow page table points to the actual page frame and is used by the hardware component called the memory management unit (MMU) for dynamic address translation. VMM manages the backend operation of these VMs by allocating the necessary computing, memory and other input/output (I/O) resources.

The shadow page table is **a data structure that is actively maintained and re-filled by the VMM**. The shadow page table mirrors what the guest is doing in terms of its own page tables and in terms of what the VMM translates the guest physical address to the host physical address.





Extended page tables

Guest page table
GVA → GPA

Host page table/map
GPA → HPA

- Page table walk by MMU: Start walking guest page table using GVA
- Guest PTE (for every level page table walk) gives GPA (cannot use GPA to access memory)
- Use GPA, walk host page table to find HPA, then access memory page, then next level access
- Every step in guest page table walk requires walking N-level host page table