MODULE 2

What is Virtualization?

Virtualization refers to the representation of physical computing resources in simulated form having made through the software. This special layer of software (installed over active physical machines) is referred as layer of virtualization. This layer transforms the physical computing resources into virtual form which users use to satisfy their computing needs.

In simple sense, the virtualization is the logical separation of physical resources from direct access of users to fulfill their service needs. Virtualization provides a level of *logical abstraction* that liberates user-installed software (starting from operating system and other systems as well as application software) from being tied to a specific set of hardware. Rather, the users install everything over the logical operating environment (rather than physical ones) having created through virtualization.

FIG 7.1: Users' interaction with computer in traditional and virtualized computing environment

Virtualizing Physical Computing Resources

Apart from basic computing devices like processor, primary memory, other resources like storage, network devices (like switch, router etc.), the communication links and peripheral devices (like keyboard, mouse, printer etc.) can also be virtualized. But, it should be noted that in case of core computing resources a virtualized component can only be operational when a physical resource empowers it from the back end. For example, a virtual processor can only work when there is a physical processor linked with it.

Figure 7.2 represents a virtualized computing environment comprising of processor, memory and storage disk. The layers of virtualization transforms these physical computing devices into virtual form and presents them before user. The important thing here to be noted is that the simulated devices produced through virtualization may or may not resemble the actual physical components (in quality, architecture or in quantity). For instance, in the given figure below, users get access to three processors while there is one physical processor in reality. Or, 32-bit processor can be produced (in virtual form) from 64-bit actual physical processor.

The software for virtualization consists of a set of control programs. It offers all of the physical computing resources in custom made simulated (virtual) form which users can utilize to build virtual computing setup or *virtual computers* or *virtual machines* (VM). Users can install operating system over virtual computer just like they do it over physical computer. Operating system installed over virtual computing environment is known as *guest operating system*. When virtualization technique is in place, the guest OS executes as if it were running directly on the physical machine.

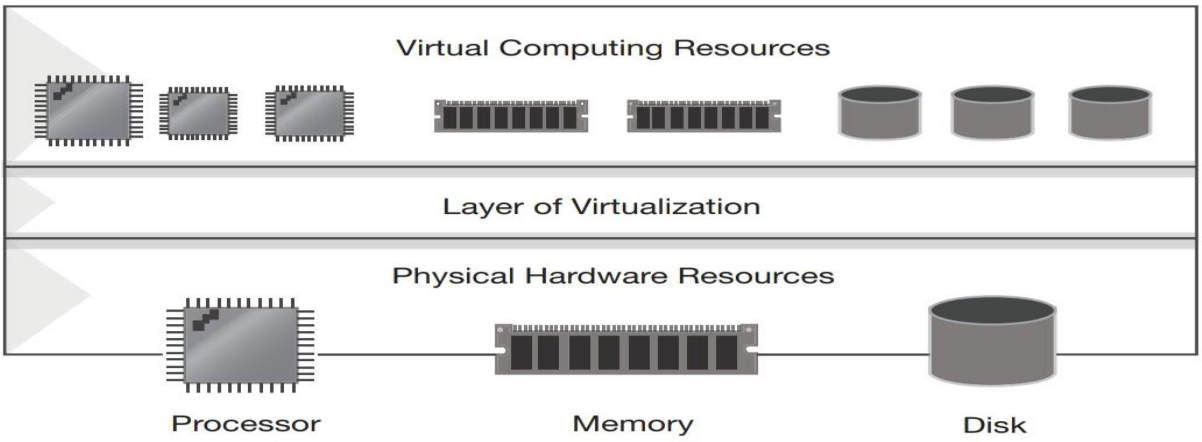


FIG 7.2: Transformation of physical resources into virtual resources through virtualization

Machine Virtualization or Server Virtualization

- *Machine virtualization* (also called *server virtualization*) is the concept of creating virtual machine (or *virtual computer*) on actual physical machine.
- The parent system on which the virtual machines run is called the *host system*, and the virtual machines are themselves referred as *guest systems*.
- In conventional computing system, there has always been a *one-to-one relationship* between physical computer and operating system. At a time, a single OS can run over them. Hardware virtualization eliminates this limitation of having a one-to-one relationship between physical hardware and operating system.
- It facilitates the running of multiple computer systems having their own operating systems on single physical machine. As shown in Figure 7.3, the OS of the guest systems running over same physical machine need not to be similar.
- All these virtual machines running over a single host system, remain independent of each other. Operating systems are installed into those virtual machines. These guest systems can access the hardware of the host system and can run applications within its own operating environment.

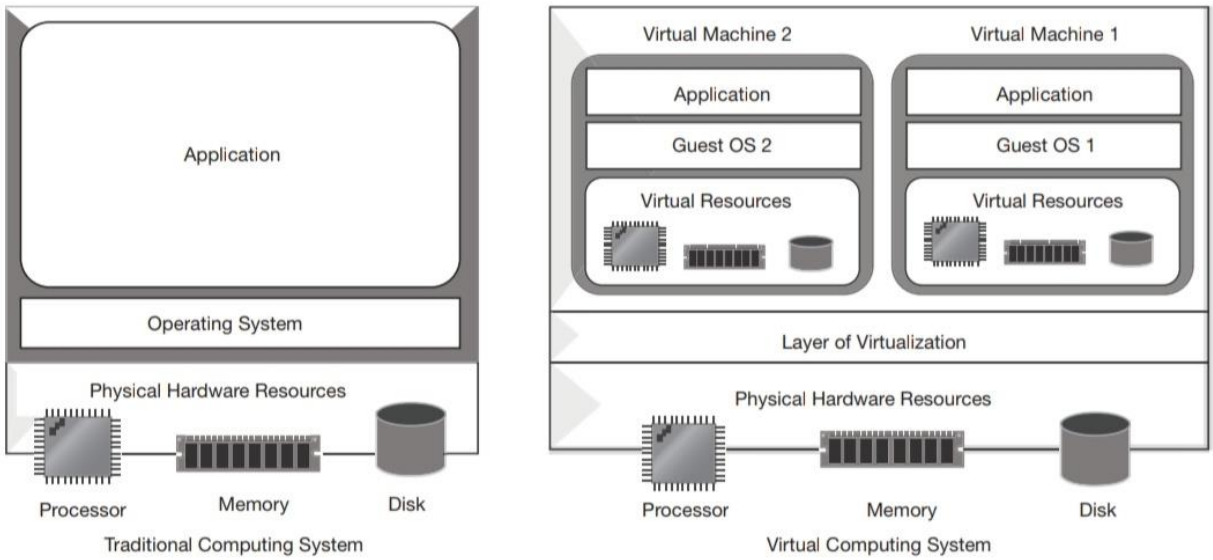


FIG 7.3: Conventional computing system versus Virtualized computing system

Comparison between non-virtualized and virtualized machine environments

Non-Virtualized Environment	Virtualized Environment
At a moment, one single OS can run on a physical machine.	Multiple OS can run simultaneously on one physical machine.
Application and hardware system remain tightly coupled.	Virtual Machines isolates applications from the underlying hardware.
Resource utilization rate is low in most of the times.	Resource utilization improves as multiple VMs share same set of physical resources.
These increase cost of business due to low resource utilization.	They are cost-effective if planned properly
They have the inflexible approach	They provide lot of flexibility to system designers

Hypervisor or Virtual Machine Monitor

- The *hypervisor* or *virtual machine monitor* (VMM) presents a virtual operating platform before the guest systems.
- It also monitors and manages the execution of guest systems and the virtual machines.
- All of the virtual machines run as self-sufficient computers isolated from others, even though they are served by the same set of physical resources.
- Alternately, it can be said that a hypervisor or VMM facilitates and monitors the execution of virtual machines and allows the sharing of the underlying physical resources among them.

The virtual machines are created over the virtualization layers. This virtualization layer is actually a set of control programs that creates the environment for the virtual machines to run on. This layer provides the access to the system resources to the virtual machines. It also controls and monitors the execution of the virtual machines over it. This software layer is referred as the *Hypervisor* or *Virtual Machine Monitor (VMM)*.

The hypervisor abstracts the underlying software and/or hardware environments and represents virtual system resources to its users. This layer also facilitates the existence of multiple VMs those are not bound to share same (underlying) OS kernel. Due to this reason, it becomes possible to run different operating systems in those virtual machines as created over a hypervisor.

Types of Virtual Machines

A Virtual Machine is a software implementation of a computer machine. It creates an isolated duplication of a real computer and allows us to perform operations as we perform operations on a real computer.

Based on use and degree of correspondence to a real machine a virtual machine is divided into two

1. System Virtual Machine
2. Process Virtual Machine

System Virtual Machine

A System virtual machine is also called hardware virtual machine. It is the software emulation of a computer system. It mimics the entire computer. In computing, an emulator is hardware or software that enables one computing system (called the host) to behave like another computing system. It is an environment that allows multiple instances of the OS (VM) to run on a host system, sharing the physical resources.

System Virtual Machine provides a platform for the execution of a complete operating system. It will create a number of different isolated identical execution environments in a single computer by partitioning computer memory to install and execute the different operating systems at the same time. It allows us to install applications in each OS, run the application in this OS as if we work in real work on a real computer. For example, we can install Windows XP/7/8 or Linux Ubuntu/Kali in Windows 10 OS with the help of VM.

Examples of System VM are

1. VMware
2. VirtualBox
3. Windows Virtual PC
4. Parallels
5. Citrix Xen

Advantages of System VM are:

1. Multiple OS environments can run in parallel on the same piece of hardware in strong isolation from each other.
2. It can provide an Instruction Set Architecture that is slightly different from the real machine.

Drawbacks of System VM are:

1. Since the VM indirectly accesses the same hardware so the efficiency is compromised.
2. Multiple VMs running in parallel on the same physical machine may result in varied performance depending on the system. Implementing proper isolation techniques may address this drawback

Process Virtual Machine

A process virtual machine allows a single process to run as an application on a host machine, providing a platform-independent programming environment by masking the information of the underlying hardware or OS. An example of process VM is the Java Virtual Machine, which enables any OS to run java applications as if they were native to the system.

A process virtual machine is also called Language Virtual Machine or Application Virtual Machine or Managed Runtime Environment. Process Virtual Machine is a software simulation of a computer system. It provides a run time environment to execute a single program and supports a single process. The purpose of Process Virtual Machine is to provide a platform independent programming environment that abstracts the details of the underlying hardware or OS and allows a program to execute in the same way on any platform.

Examples of Process Virtual Machine are:

1. JVM (Java Virtual Machine) is used for the Java language
2. PVM (Parrot Virtual Machine) is used for the Perl language

Emulation, Interpretation and Binary translation

Emulation in computing is done by making one system imitating another. This means a system having some architecture is made enable to support instruction set of some other machine architecture. For example, let a piece of software has been made for architecture 'A' and is not supported by architecture 'B'. Through emulation, it is possible to imitate the working of system 'A' (i.e. architecture 'A') on system 'B' (i.e. architecture 'B') and then the piece of software to run on system B. Emulators can be software or hardware both.

Emulation software converts binary data written for execution on one machine to an equivalent binary form suitable to execute on another machine. This is done by translating the binary instructions. There are two ways for implementation of emulations like interpretation and binary translation.

In *binary translation* (also known as *recompilation*), a total conversion of the binary data (made for the emulated platform) is done. The conversion recompiles the whole instruction into another binary form

suitable to run on the actual or targeted platform. There are two types of binary translation like *static recompilation* and *dynamic recompilation*.
In *interpretation*, each instruction is interpreted by the emulator every time it is being encountered. This method is easier to implement but slower than binary translation process.

Types of Machine Virtualization & Types of Hypervisors

There are two different techniques of server or machine virtualization they are hosted approach and the bare metal approach. The techniques differ depending on the type of hypervisor used.

Bare Metal Approach & Type 1 Hypervisor

- In this approach of machine virtualization, the hypervisor is directly installed over the physical machine.
- Since, the hypervisor is the first layer over hardware resources, hence, the technique is referred as bare metal approach.
- Here, the VMM or the hypervisor communicates directly with system hardware.
- In this approach, the hypervisor acts as *low-level virtual machine monitor* and also called as *Type 1 hypervisor* or *Native Hypervisor*.
- VMware's ESX and ESXi Servers, Microsoft's Hyper-V, solution Xen are some of the examples of *bare-metal hypervisors*.

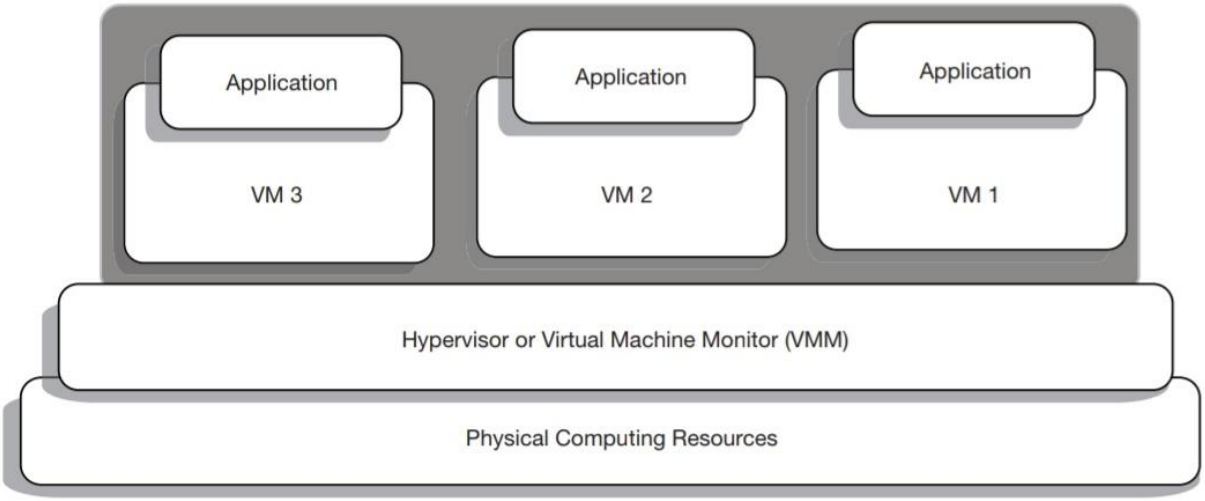


FIG 7.5: A model for the bare metal approach of machine virtualization

Benefits: Since the bare metal hypervisor can directly access the hardware resources in most of the cases it provides better performance in comparison to the hosted hypervisor. For bigger application like enterprise data centers, bare-metal virtualization is more suitable because usually it provides advanced features for resource and security management. Administrators get more control over the host environment.

Drawbacks: As any hypervisor usually have limited set of device drivers built into it, so the bare metal hypervisors have limited hardware support and cannot run on a wide variety of hardware platform.

Hosted Approach & Type 2 Hypervisor

- In this approach, an operating system is first installed on the physical machine to activate it.
- This OS installed over the host machine is referred as *host operating system*.
- The hypervisor is then installed over this host OS. This type of hypervisor is referred to as *Type 2 hypervisor* or *Hosted hypervisor*.

- Figure 7.4 represents the hosted machine virtualization technique. So, here the host OS works as the first layer of software over the physical resources.
- Hypervisor is the second layer of software and guest operating systems run as the third layer of software.
- Products like VMWare Workstation and Microsoft Virtual PC are the most common examples of *type 2 hypervisors*.

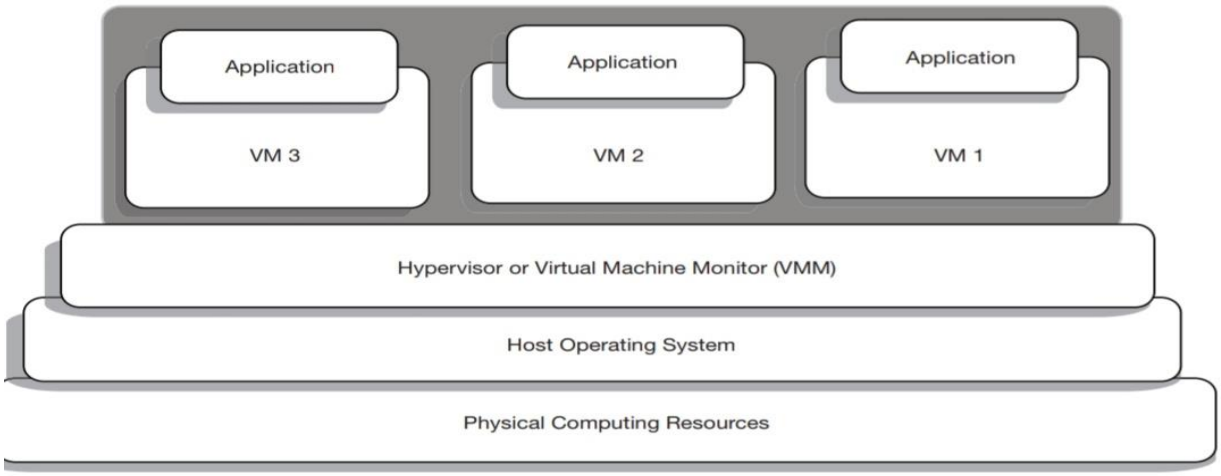


FIG 7.4: A model of hosted machine virtualization approach

Benefits: In this approach, the host OS supplies the hardware drivers for the underlying physical resources. This eases the installation and configuration of the hypervisor. It makes the type-2 hypervisors compatible for a wide variety of hardware platform.

Drawbacks: A hosted hypervisor does not have direct access to the hardware resources and hence, all of the requests from virtual machines must go through the host OS. This may degrade the performance of the virtual machines. Another drawback of the hosted virtualization is the lack of support for real-time operating systems. Since the underlying host OS controls the scheduling of jobs it becomes unrealistic to run a real-time OS inside a VM using hosted virtualization.

Hypervisor- Based Virtualization Approaches

Hypervisor-based virtualization techniques can be divided into three categories as full virtualization, para-virtualization and hardware-assisted virtualization.

Full Virtualization

- In full virtualization (also called as *native virtualization*), the hypervisor fully simulates or emulates the underlying hardware.
- Virtual machines run over these virtual set of hardware.
- The guest operating systems assume that they are running on actual physical resources and thus remain unaware that they have been virtualized. This enables the unmodified versions of available operating systems (like Windows, Linux and else) to run as guest OS over hypervisor.
- In this model, it is the responsibility of the hypervisor to handle all OS-to-hardware (i.e. guest OS to physical hardware) requests during running of guest machines. The guest OS remains completely isolated from physical resource layers by the hypervisor. This provides flexibility as almost all of the available operating systems can work as guest OS.
- VMware's virtualization product *VMWare ESXi Server* and *Microsoft Virtual Server* are few examples of full virtualization solution.
- In full virtualization technique, the guest operating systems can directly run over hypervisor.

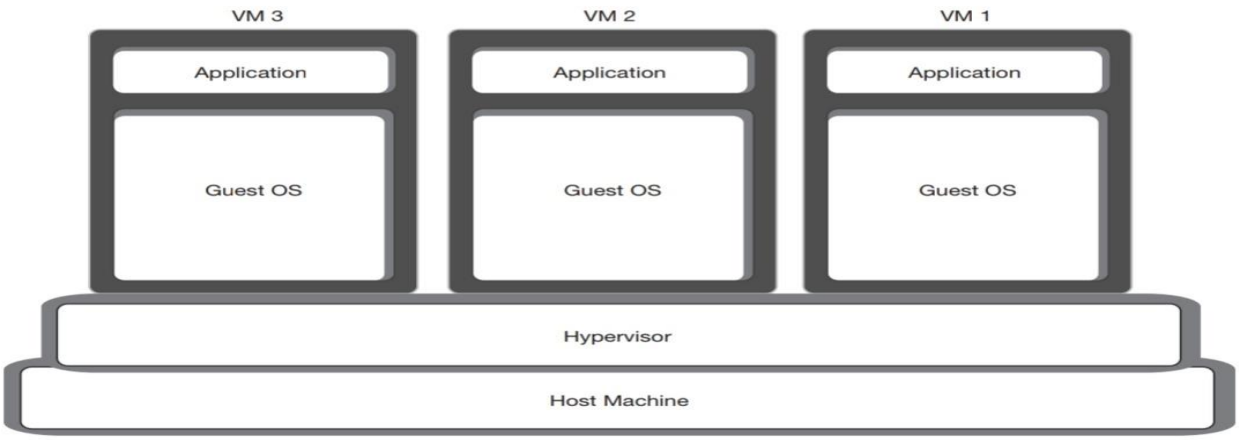


FIG 7.6: A model of full virtualization

Para-Virtualization or OS-Assisted Virtualization

- In *para-virtualization*, a portion of the virtualization management task is transferred (from the hypervisor) towards the guest operating systems. Normal versions of available operating systems are not capable of doing this. They need special modification for this capability inclusion.
- This modification is called *porting*. Each guest OS is explicitly ported for the para-application program interface (API).
- Thus in para-virtualization, each guest OS needs to have prior knowledge that it will run over the virtualized platform. Moreover, it also has to know on which particular hypervisor they will have to run.
- Depending on the hypervisor, the guest OS is modified as required to participate in the virtualization management task.
- The unmodified versions of available operating systems (like Windows, Linux) cannot be used in para-virtualization. Since it involves modifications of the OS the para-virtualization is sometimes referred to as *OS-Assisted Virtualization* too.
- This technique relieves the hypervisor from handling the entire virtualization tasks to some extent. Best known example of paravirtualization hypervisor is the open-source *Xen project* which uses a customized Linux kernel.
- Para-virtualization requires hypervisor-specific modifications of guest operating systems.

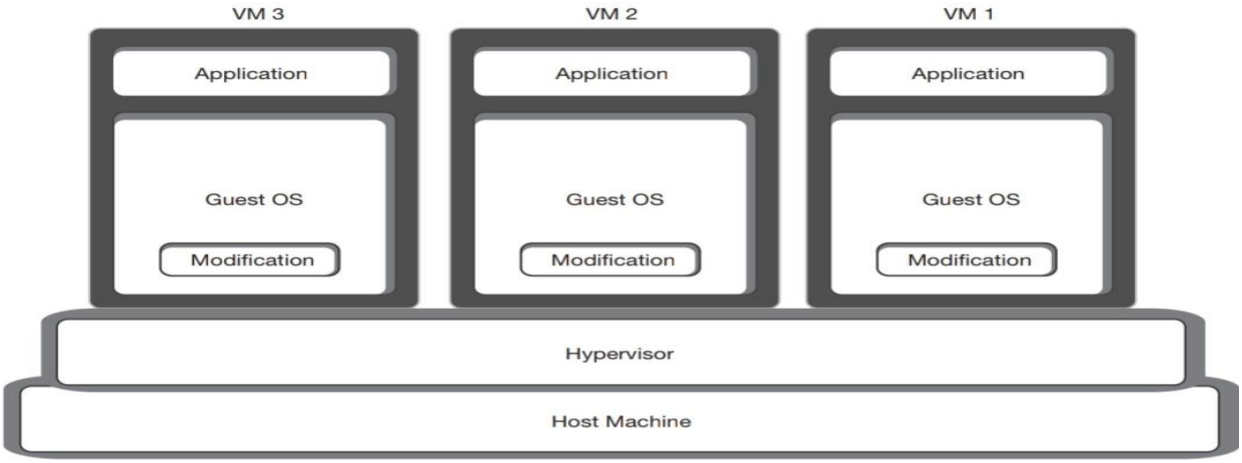


FIG 7.7: A model of para-virtualization

Hardware-Assisted Virtualization

- Also known as native virtualization
- In this technique, **underlying hardware provides special CPU instructions to aid virtualization**.
- This technique is also highly portable as the hypervisor can run an unmodified guest OS.
- This technique makes hypervisor implementation less complex and more maintainable.
- Intel's Intel-VT and AMD's AMD-V processors provide CPU virtualization instructions that software vendors use to implement hardware-assisted virtualization.
- This kind of virtualization is only possible when specific combinations of hardware components are used.
- Hardware-assisted virtualization requires explicit features in the host machine's CPU.

Comparison among Full Virtualization, Para-Virtualization and Hardware-Assisted Virtualization

Full Virtualization	**Para-Virtualization or OS-Assisted Virtualization**	**Hardware-Assisted Virtualization**
Guest OS has no role in virtualization.	Guest OS plays role in virtualization.	Guest OS has no role in virtualization.
Guest OS remains unaware about the virtualization.	Guest OS has to be aware about the virtualization.	Guest OS remains unaware about the virtualization
Normal version of available OS can be used as guest OS.	Modified version of available OS is required.	Normal version of available OS can be used as guest OS.
It provides good options for guest OS.	It provides lesser options for guest OS.	It provides good options for guest OS.
Guest OS is not hypervisorspecific.	Guest OS is tailored to be hypervisor-specific.	Guest OS is not hypervisorspecific.
Here it requires no special feature in the host CPU.	Here it requires no special feature in the host CPU.	Here it requires explicit features in the host CPU.
Hardware does not play role in virtualization.	Hardware does not play role in virtualization.	Hardware plays role in virtualization.
Hypervisor takes care of all of the virtualization tasks.	Guest OS along with hypervisor take care of the virtualization tasks.	Specialized hardware device along with hypervisor take care of virtualization tasks.
Virtualization performance is little slow.	Virtualization performance is better.	Virtualization performance is better.

Operating system level virtualization

- also called as *system level virtualization*
- Here, the host–guest paradigm does not work. The virtual machine terminology is also not used here.

- No hypervisor is used and the virtual servers are enabled by the kernel of the operating system of physical machine.
- Here, the kernel of the operating system installed over physical system is shared among all of the virtual servers running over it. Since all of the virtual servers share a single kernel, it is evident that all of them will have same OS as the parent system.
- The goal of this approach is to create multiple logically-distinct user-space instances (virtual servers) over a single instance of an OS kernel. This approach is also known as *Operating System Virtualization* or *Shared Kernel Approach*.
- Virtualization solutions such as FreeBSD's jail, Linux VServer, OpenVZ are few examples of OS-level virtualization. All of them can run logically-distinct user-spaces on top of a single kernel.

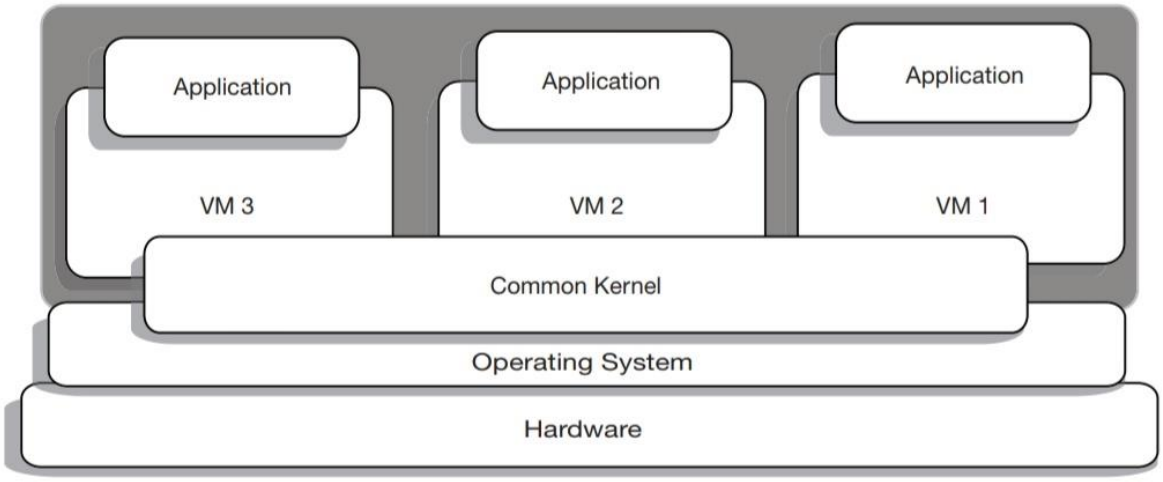


FIG 7.8. A model of operating system-level virtualization approach

Network Virtualization

Network virtualization is the process of combining network resources and network functionality into a single, software-based administrative entity called as a *virtual network*. There are two common forms of network virtualization

- *Virtual device-based virtual network*: Here, virtualized devices form the network. All virtual networking devices (including virtual computers, virtual switches, virtual routers etc.) communicate using actual (non-virtual) network protocols such as Ethernet as well as virtualization protocols such as the VLAN. This is actual network virtualization where the network is formed with all virtual components.
- *Protocol based virtual network*: Rather than virtualizing devices, it creates virtual area network. Virtual LAN (VLAN) and virtual private network (VPN) are examples of such virtualizations. These are logical local area networks (logical LANs) where the underlying physical LAN's structure is something else. Here, several physical LANs which are actually part of public network (such as the Internet) can function as a single logical LAN. This enables network devices (such as computers and switches) to send and receive data across shared or public networks as if they are part of a private network. The devices can communicate using LAN protocols which make faster and secure network communication.

Storage Virtualization

- In traditional computing system, the storages have always been directly linked with the physical servers. With virtualization of storage, this concept has been changed. Now virtualized storage systems are linked with servers and actual (physical) storage systems remain hidden.
- Like other computing resources, virtualization of storage also happens through layer of software which creates *logical abstraction* of the pooling of physical storage devices having linked together by network. Data stored in logical (virtualized) storage devices ultimately get stored in some physical storage disks. The advent of Storage Area Networks (SAN) has made the pooling (and hence the virtualization as well) of physical storage systems easier.
- There are many commercial virtualized cloud storage systems available in the market. Google Cloud Storage, Microsoft's Azure Storage, Simple Storage System (S3) and Elastic Block Store (EBS) of Amazon are few to name among them.

Desktop Virtualization

Desktop virtualization abstracts the desktop environment available on a personal computer in order to provide access to it using a client/server approach. Desktop virtualization provides the same out- come of hardware virtualization but serves a different purpose. Similarly to hardware virtualization, desktop virtualization makes accessible a different system as though it were natively installed on the host, but this system is remotely stored on a different host and accessed through a network con- nection. Moreover, desktop virtualization addresses the problem of making the same desktop envi- ronment accessible from everywhere. Although the term *desktop virtualization* strictly refers to the ability to remotely access a desktop environment, generally the desktop environment is stored in a remote server or a data center that provides a high-availability infrastructure and ensures the acces- sibility and persistence of the data. Infrastructures for desktop virtua- lization based on cloud computing solutions include Sun Virtual Desktop Infrastructure (VDI), Parallels Virtual Desktop Infrastructure (VDI), Citrix XenDesktop, and others.

Advantages of Virtualization

- **Better Utilization of Existing Resources:** In Traditional computing most of the processing power simply remain unutilized for most of the computer systems. Running multiple virtual machines on one physical server makes better utilization of the resources and this is known as *server consolidation*.
- **Reduction in Hardware Cost:** As virtualization makes better use of physical resources by running multiple virtual machines on single set of physical resources, automatically cost of computing comes down.
- **Reduction in Computing Infrastructure Costs:** Reduced physical computing resource requirements in turn reduces many other associated assets, like physical floor space, power requirement, cooling system and human resource to administrate the systems.
- **Improved Fault Tolerance or Zero Downtime Maintenance:** In cases of any hardware failure, the virtual system can be migrated to another physical setup. This helps to build fault tolerant system by creating scope for *zero downtime maintenance*.
- **Simplified Capacity Expansion:** Capacities of virtual resources are easier to increase than expanding and then synchronizing physical computing resources. This also becomes possible due to the decoupling of physical resources from virtual systems.

- **Simplified System Installation:** Installation of a new system has become easier, cost-effective and hassle-free in virtual environment. A new system can be installed almost within no time by cloning a virtual machine instance. Fresh installation is much easier too than physical machine installation.
- **Security:** Virtualization adds a layer of abstraction over physical hardware. Virtual machines cannot directly access physical resources any more. This can restrict the amount of destruction that might occur when some malicious software attempts to damage the system or corrupt data. For example, if an entire virtual hard disk gets damaged or corrupted, the actual physical disk remains unaffected.
- **Simplified System-Level Development:** System software (like device driver) development and testing require frequent re-booting of the system. This is easier and faster in virtual environment since VM rebooting does not require physical machine to restart and can be performed with some clicks of mouse.

Downsides of Virtualization

- **Single Point of Failure Problem:** The major benefit of virtualization is resource sharing. Multiple virtual machines can run over one physical machine. But, this has a downside. It increases the probability of failure of a number of virtual servers in cases of failure of single physical machine. Although this situation can be handled easily by keeping backup resources and porting those virtual servers on the backup set of physical resources. Porting is not a difficult task as virtualization decouples virtual systems from physical resources.
- **Lower Performance Issue:** There is a concern whether virtual environments have the capacity to accomplish the full performance of the actual physical system. It has been seen that virtual servers can achieve up to 85 percent to 90 percent of the performance of the actual physical server as VMs cannot get direct access to the hardware.
- **Difficulty in Root Cause Analysis:** With virtualization, a new layer of complexity is added which can cause new problems. The main difficulty is that if something does not work as it is supposed to it may require considerable extra efforts to find the cause of the problem.

Ques) Why is hypervisor also called virtual machine monitor?

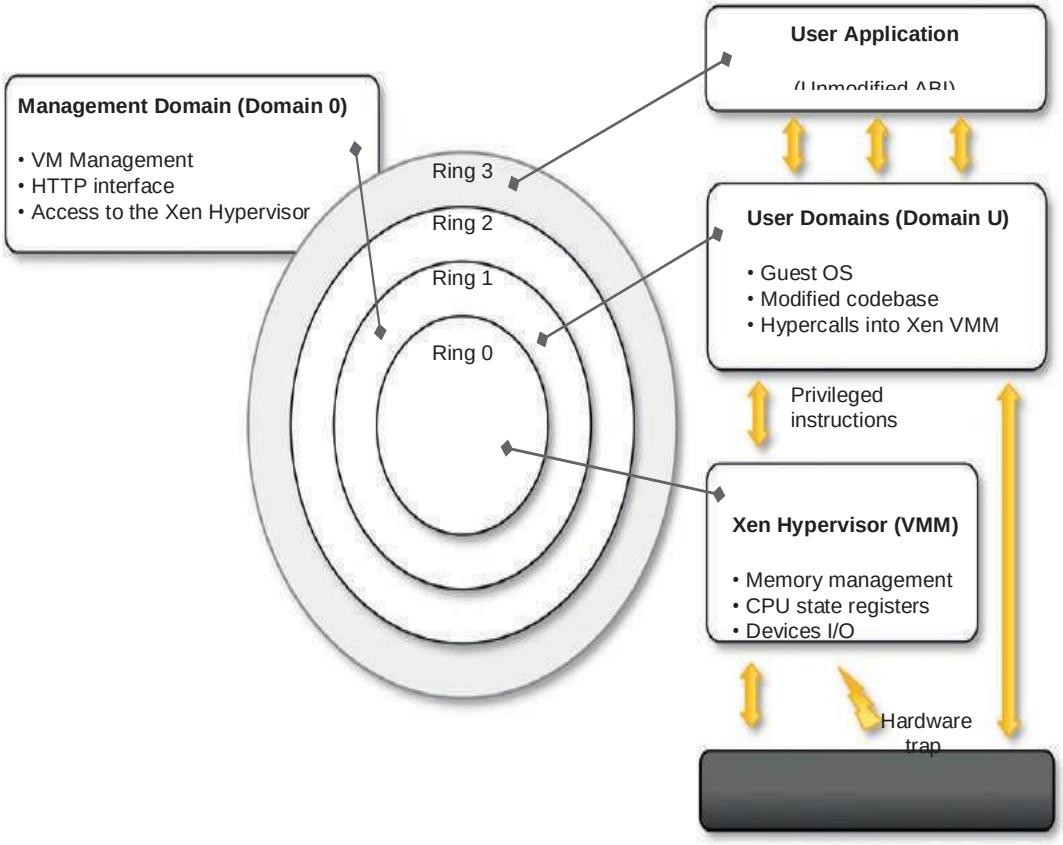
Hypervisor produces all of the virtual computing resources. Virtual machines are created over the layer of hypervisor using these virtual resources. It is the responsibility of the hypervisor to manage activities, monitor functionalities and provide support to the virtual machines. This is why hypervisor is also called as virtual machine monitor or VMM.

Case Study

Xen: paravirtualization

- Xen is an open-source initiative implementing a virtualization platform based on paravirtualization.
- Initially developed by a group of researchers at the University of Cambridge in the United Kingdom
- Xen-based technology is used for either desktop virtualization or server virtualization, and recently it has also been used to provide cloud computing solutions by means of Xen Cloud Platform (XCP).
- Xen is the most popular implementation of *paravirtualization*, which, in contrast with full virtualization, allows high-performance execution of guest operating systems.

- Figure describes the architecture of Xen and its mapping onto a classic x86 privilege



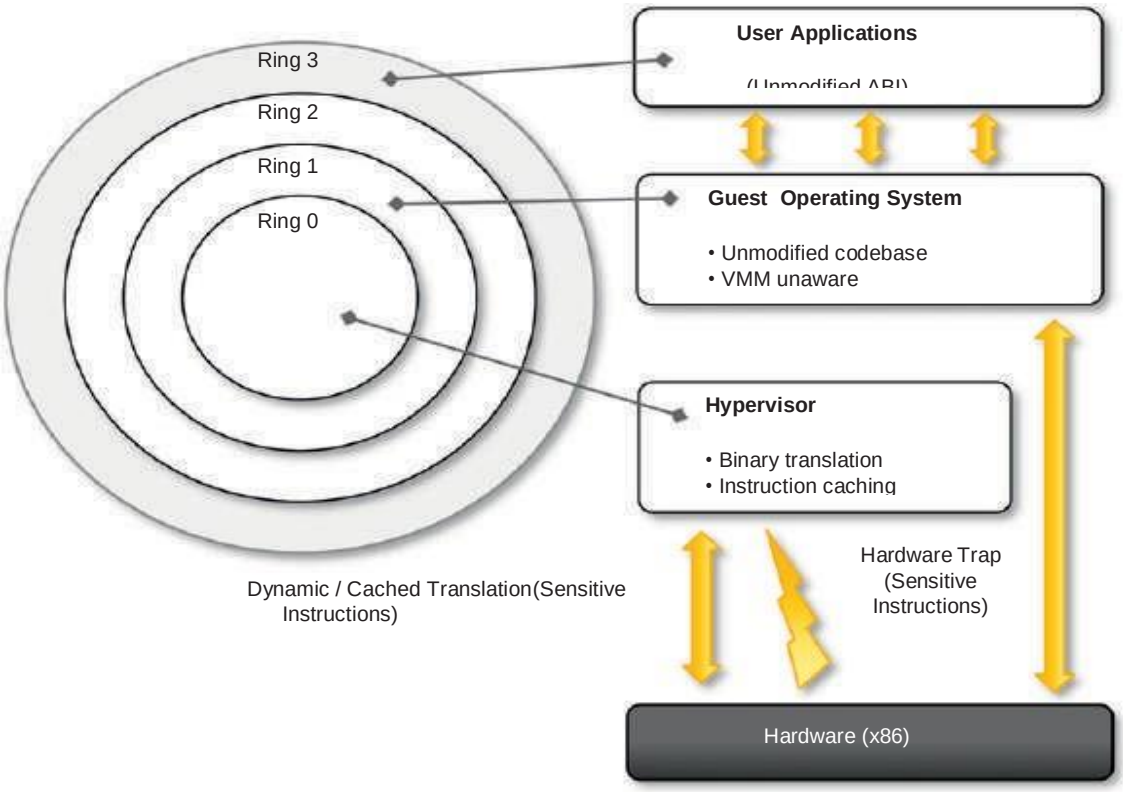
model. A Xen-based system is managed by the *Xen hypervisor*, which runs in the highest privileged mode and controls the access of guest operating system to the underlying hardware.

- Guest operating systems are executed within *domains*, which represent virtual machine instances.
- Many of the x86 implementations support four different security levels, called *rings*, where Ring 0 represent the level with the highest privileges and Ring 3 the level with the lowest ones.
- Almost all the most popular operating systems, except OS/2, utilize only two levels: Ring 0 for the kernel code, and Ring 3 for user application and non-privileged OS code. This provides the opportunity for Xen to implement virtualization by executing the hypervisor in Ring 0, Domain 0, and all the other domains running guest operating systems
- Because of the structure of the x86 instruction set, some instructions allow code executing in Ring 3 to jump into Ring 0 (kernel mode). Such operation is performed at the hardware level and therefore within a virtualized environment will result in a *trap* or *silent fault*, thus preventing the normal operations of the guest operating system.
- To avoid this situation, operating systems need to be changed in their implementation
- Paravirtualization needs the operating system codebase to be modified, and hence not all operating systems can be used as guests in a Xen-based environment. More precisely, this condition holds in a scenario where it is not possible to leverage hardware-assisted virtualization.
- Therefore, Xen exhibits some limitations in the case of legacy hardware and legacy operating systems.
- Open-source operating systems such as Linux can be easily modified, since their code is publicly

available and Xen provides full support for their virtualization, whereas components of the Windows family are generally not supported by Xen unless hardware-assisted virtualization is available.

VMware: Full Virtualization

- VMware's technology is based on the concept of full virtualization, where the underlying hardware is replicated and made available to the guest operating system, which runs unaware of such abstraction layers and does not need to be modified.
- VMware implements full virtualization either in the desktop environment, by means of Type II hypervisors, or in the server environment, by means of Type I hypervisors. In both cases, full virtualization is made possible by means of direct execution (for nonsensitive instructions) and binary translation (for sensitive instructions), thus allowing the virtualization of architecture such as x86.



Full virtualization and binary translation

- VMware is well known for the capability to virtualize x86 architectures, which runs unmodified on top of their hypervisors
- x86 architecture design does not satisfy the first theorem of virtualization .
- This causes a different behavior when such instructions are not executed in Ring 0, which is the normal case in a virtualization scenario where the guest OS is run in Ring 1.
- Generally, a trap is generated and the way it is managed differentiates the solutions in which virtualization is implemented for x86 hard- ware.
- This approach has both advantages and disadvantages. The major advantage is that guests can run unmodified in a virtualized environment, which is a crucial feature for operating systems for which source code is not available. This is the case, for example, of operating systems in the Windows family.

- Binary translation is a more portable solution for full virtualization.
- On the other hand, translating instructions at runtime introduces an additional overhead that is not present in other approaches.
- Even though such disad- vantage exists, binary translation is applied to only a subset of the instruction set, whereas the others are managed through direct execution on the underlying hardware. This somehow reduces the impact on performance of binary translation.
- VMware achieves full virtualization by providing virtual representation of memory and I/O devices.
- Finally, VMware also provides full virtualization of I/O devices such as network controllers and other peripherals such as keyboard, mouse, disks, and universal serial bus (USB) controllers.

****