4

Virtualization

Learning Objectives

After reading this chapter, you will be able to

- Summarize the significance of Virtualization
- Appraise the Virtualization techniques
- Realize the pros and cons of Virtualization

The term “virtualization” was coined in the 1960s to refer to a virtual machine (sometimes called “pseudo machine”), a term which itself dates from the experimental IBM M44/44X system. The creation and management of virtual machines (VM) has been called “platform virtualization”, or “server virtualization”, more recently. Platform virtualization is performed on a given hardware platform by host software (a control program), which creates a simulated computer environment, a VM for its guest software. The guest software is not limited to user applications; many hosts allow the execution of complete operating systems. The guest software executes as if it were running directly on the physical hardware, with several notable caveats.

In this chapter we discuss about the virtulization technology, different techniques of virtualization, pros and cons of virtulaization.

Preliminaries

The following are some of the terms and terminologies defined here for easier understanding in the remaining part of this chapter.

Virtualization: Virtualization is a broad term that refers to the abstraction of computer resources. Virtualization hides the physical characteristics of computing resources from their users, be thee applications, or end users.

Hypervisor: A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

Emulation: Emulation is the use of an application program or device to imitate the behavior of another program or device.

Binary translation: Binary translation is a form of binary recompilation where sequences of instructions are translated from a source instruction set to the target instruction set.

Paravirtualization: Para-virtualization is a virtualization technique that presents a software interface to the virtual machines which is similar, yet not identical to the underlying hardware-software interface.

Software virtualization: It is just like a virtualization but able to abstract the software installation procedure and create virtual software installations. Virtualized software is an application that will be “installed” into its own self-contained unit.

Hardware virtualization: Hardware virtualization enables multiple copies of the same or different operating systems to run in the computer and prevents the OS and its applications in one VM from interfering with the OS and applications in another VM.

Network and Storage Virtualization: In a network, virtualization consolidates multiple devices into a logical view so they can be managed from a single console. Virtualization also enables multiple storage devices to be accessed the same way no matter their type or location.

Containers: Containers are the products of operating system virtualization. They provide a lightweight virtual environment that groups and isolates a set of processes and resources such as memory, CPU, disk, etc., from the host and any other containers.

OS Virtualization: Under the control of one operating system, a server is split into containers where each one handles an application.

Fig. 4.1 shows the virtualization architecture. It shows the guest operating system which has the capability to invoke VM instances. The VM is a software computer that, like a physical computer, runs an operating system and applications. The hypervisor serves as a platform for running virtual machines and allows for the consolidation of computing resources. Both these layers are supported by the host operating system layer.

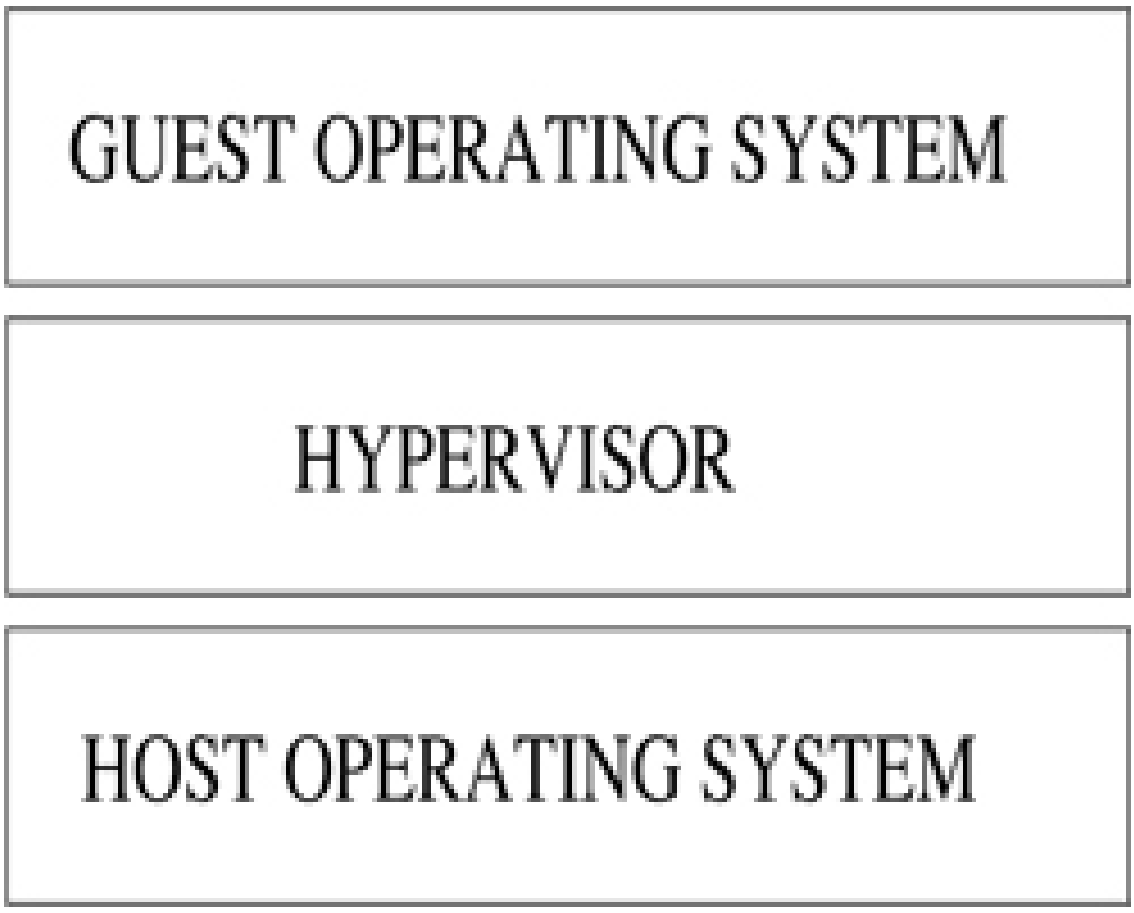

FIGURE 4.1
Virtualization architecture

4.1 Virtualization Technology

Any discussion of Cloud computing typically begins with virtualization. Virtualization is means of using computer resources to imitate other computer resources or whole computers. It separates resources and services from the underlying physical delivery environments.

Characteristics of Virtualization in Cloud Computing

Virtualization has several characteristics that make it ideal for Cloud computing, which are discussed as follows:

Partitioning: In virtualization, many applications and operating systems (OS) are supported in a single physical system by partitioning (separating) the available resources.

Isolation: Each VM is isolated from its host physical system and other virtualized machines. Because of this isolation, if one virtual-instance crashes, it does not affect the other VMs. In addition, data is not shared between one virtual container and another.

Encapsulation: A VM can be represented (and even stored) as a single file, so its identification is easy, based on the services that it provides. In essence, the encapsulated process could be a business service. This encapsulated VM can be presented to an application as a complete entity. Therefore, encapsulation can protect each application in order to stop its interference with another application.

Consolidation: Virtualization eliminates the need of a dedicated single system to one application and hence, multiple OS can run in the same server. Both old and advanced version of OS may be deployed in the same platform without purchasing additional hardware. Further, new required applications may be run simultaneously on their respective OS.

Easier development flexibility: Application developers may able to run and test their applications and programs in heterogeneous OS environments on the same virtualized machine. It facilitates the VM to host heterogeneous OS. Isolation of different applications in their respective virtual partition also helps the developers.

Migration and cloning: VM can be moved from one site to another to balance the workload. As the result of migration, users can access updated hardware as well as make recovery from hardware failure. Cloned VMs are easy to deploy in the local sites as well as remote sites.

Stability and security: In a virtualized environment, host OS hosts different types of multiple guest OS containing multiple applications. Each VM is isolated from each other and they do not interfere into the other's work, which in turn helps the security and stability aspect.

4.2 Virtualization Platforms

Platform virtualization software, specifically emulators and hypervisors, are software packages that emulate the whole physical computer machine, often providing multiple virtual machines on one physical platform. Here, we discuss the basic information about platform virtualization hypervisors, namely Xen and VMware hypervisors.

4.2.1 Xen Virtualization

It is available for the Linux kernel, and is designed to consolidate multiple OS to run on a single server, normalize hardware accessed by the OS, isolate misbehaving applications, and migrate running OS instances from one physical server to another. Recent advances in virtualization technologies, such as enabling data centers to consolidate servers, normalize hardware resources and isolate applications on the same physical server, are driving rapid adoption of server virtualization in Linux environments.

Fig. 4.2 shows Xen hypervisor architecture. The Xen hypervisor is the basic abstraction layer of software that sits directly on the hardware below any operating systems. It is responsible for CPU scheduling and memory partitioning of the various virtual machines running on the hardware device. Domain 0, a modified Linux kernel, is a unique virtual machine running on the Xen hypervisor that has special rights to access physical I/O resources as well as interact with the other virtual machines (Domain U:

Para Virtual (PV) and Hardware Virtual Machine (HVM) Guests) running on the system. All Xen virtualization environments require Domain 0 to be running before any other virtual machines can be started.

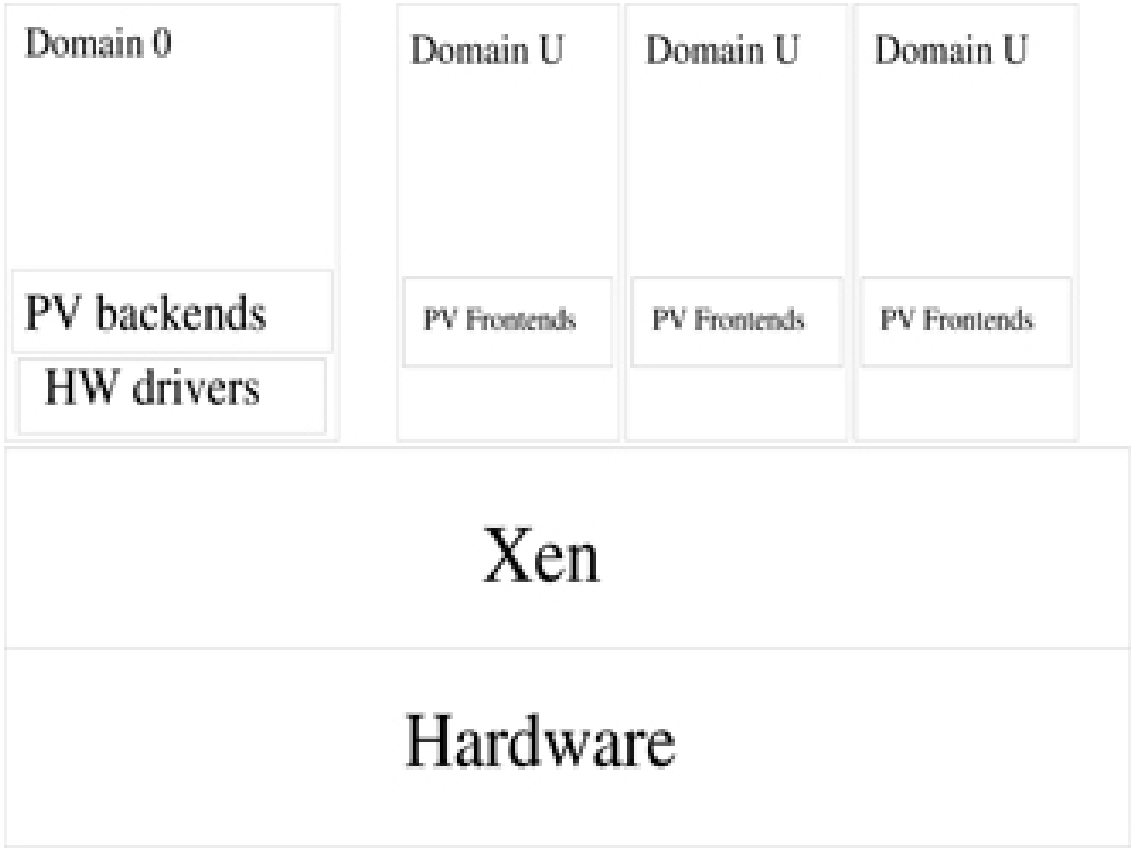

FIGURE 4.2
Xen Hypervisor architecture

All paravirtualized virtual machines running on a Xen hypervisor are referred to as Domain U PV Guests and run modified Linux operating systems, Solaris, FreeBSD and other UNIX operating systems. All fully virtu-

erating system.

Domain 0 is the initial domain started by the Xen hypervisor on boot. Domain 0 is a privileged domain that starts first and manages the Domain U unprivileged domains. The Xen hypervisor is not usable without Domain 0. This is essentially the “host” operating system (or a “service console”). As a result, Domain 0 runs the Xen management toolstack, and has special privileges, like being able to access the hardware directly.

Domain 0 has drivers for hardware, and it provides Xen virtual disks and network access for guests each referred to as a domU (unprivileged domains). For hardware that is made available to other domains, like network interfaces and disks, it will run the BackendDriver, which multiplexes and forward the hardware requests from the FrontendDriver in each Domain U. Unless DriverDomain's are being used or the hardware is passed to the domain U, the domain 0 is responsible for running all of the device drivers for the hardware.

The Xen Cloud platform addresses the needs of Cloud providers by combining the isolation and multi-tenancy capabilities of the Xen hypervisor with enhanced security, storage and network virtualization technologies to offer a rich set of virtual infrastructure Cloud services. The platform also addresses user requirements for security, availability, performance and isolation across both private and public Clouds.

4.2.2 VMware

The traditional mainframe approach runs virtual machines in a less privileged mode in order to allow the Virtual Machine Monitor (VMM) to regain control on privileged instructions, and relies on the VMM to virtualize and interface directly to the I/O devices. Also, the VMM is in complete control of the entire machine. This approach does not apply as easily to PCs for the following reasons.

Non-virtualizable processor: The Intel IA-32 processor architecture is not naturally virtualizable. Because the IA-32 processor does not meet this condition, it is not possible to virtualize the processor by simply executing all virtual machine instructions in a less privileged mode.

PC hardware diversity: There is a large diversity of devices that may be found in PCs. This is a result of the PC's "open" architecture. In a traditional implementation, the virtual machine monitor would have to manage these devices. This would require a large programming effort to provide device drivers in the VMM for all supported PC devices.

Pre-existing PC software: Unlike mainframes that are configured and managed by experienced system administrators, desktop and workstation

PC's are often pre-installed with a standard OS set up and managed by the end-user. In this environment, it is extremely important to allow a user to adopt virtual machine technology without losing the ability to continue using the existing OS and applications. It would be unacceptable to completely replace an existing OS with a virtual machine monitor.

VMware Workstation has a hosted architecture that allows it to co-exist with a pre-existing host operating system, and rely upon that operating system for device support. **Figure 4.3** illustrates the components of this hosted architecture. VMware Workstation installs like a normal application on an operating system, known as the host operating system. When run, the application portion (VM APP) uses a driver loaded into the host operating system (VM DRIVER) to establish the privileged virtual machine monitor component (VMM) that runs directly on the hardware. From then on, a given physical processor is executing either the host world or the VMM world, with the VM Driver facilitating the transfer of control between the two worlds. A world switch between the VMM and the host worlds involves saving and restoring all user and system visible state on the CPU, and is thus more heavyweight than a normal process switch.

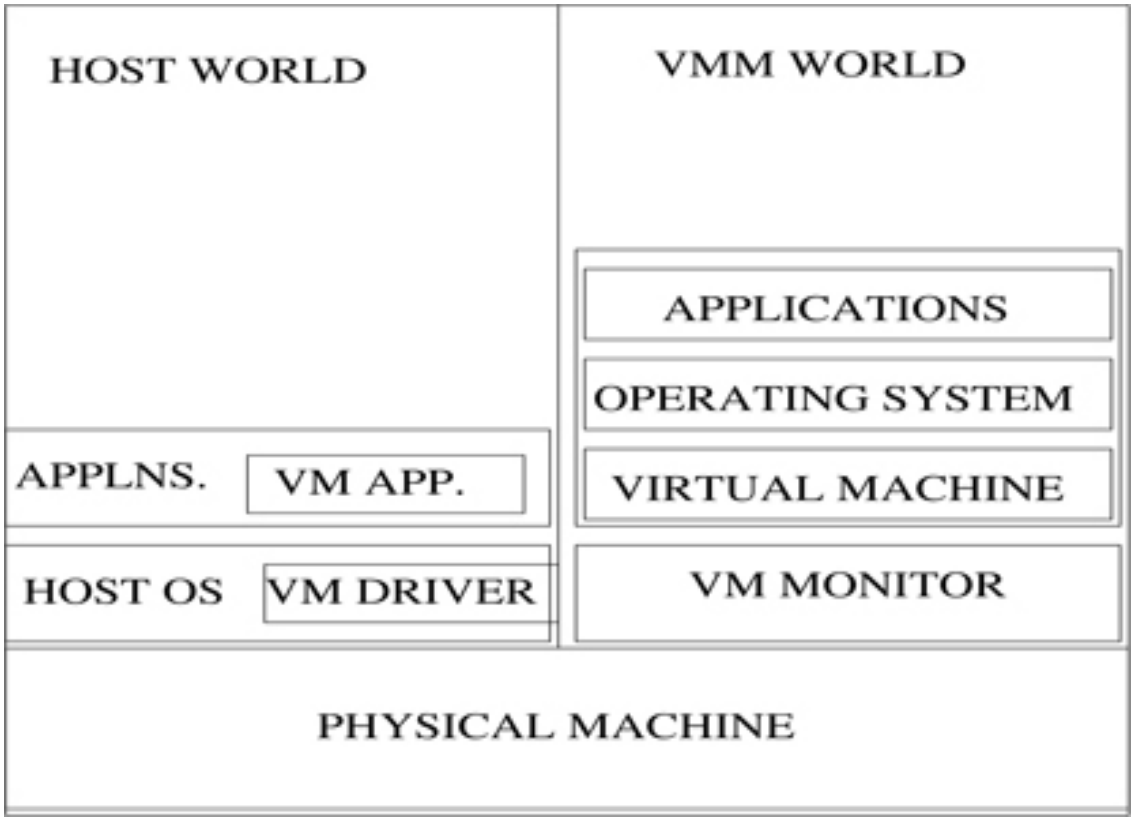

FIGURE 4.3
VMWare's hosted VM

4.3 Virtualization Techniques

When deciding on the best approach to implementing virtualization, it is important to have a clear understanding of the different virtualization solutions which are currently available. We discuss the four virtualization techniques that are commonly used today, namely hypervisor, guest operating system, shared kernel and kernel level.

4.3.1 Hypervisor Virtualization

The x86 family of CPUs provide a range of protection levels, also known as rings in which code can execute (**Figure 4.4**). Ring 0 has the highest level privilege and it is in this ring that the OS kernel normally runs. Code executing in ring 0 is said to be running in system space, kernel mode or supervisor mode. All other code such as applications running on the OS operates in less privileged rings, typically ring 3. Under hypervisor virtualization, a program known as a hypervisor (also known as a type 1 virtual machine monitor or VMM) runs directly on the hardware of the host system in ring 0. The task of this hypervisor is to handle resource and memory allocation for the VM in addition to providing interfaces for higher level administration and monitoring tools.

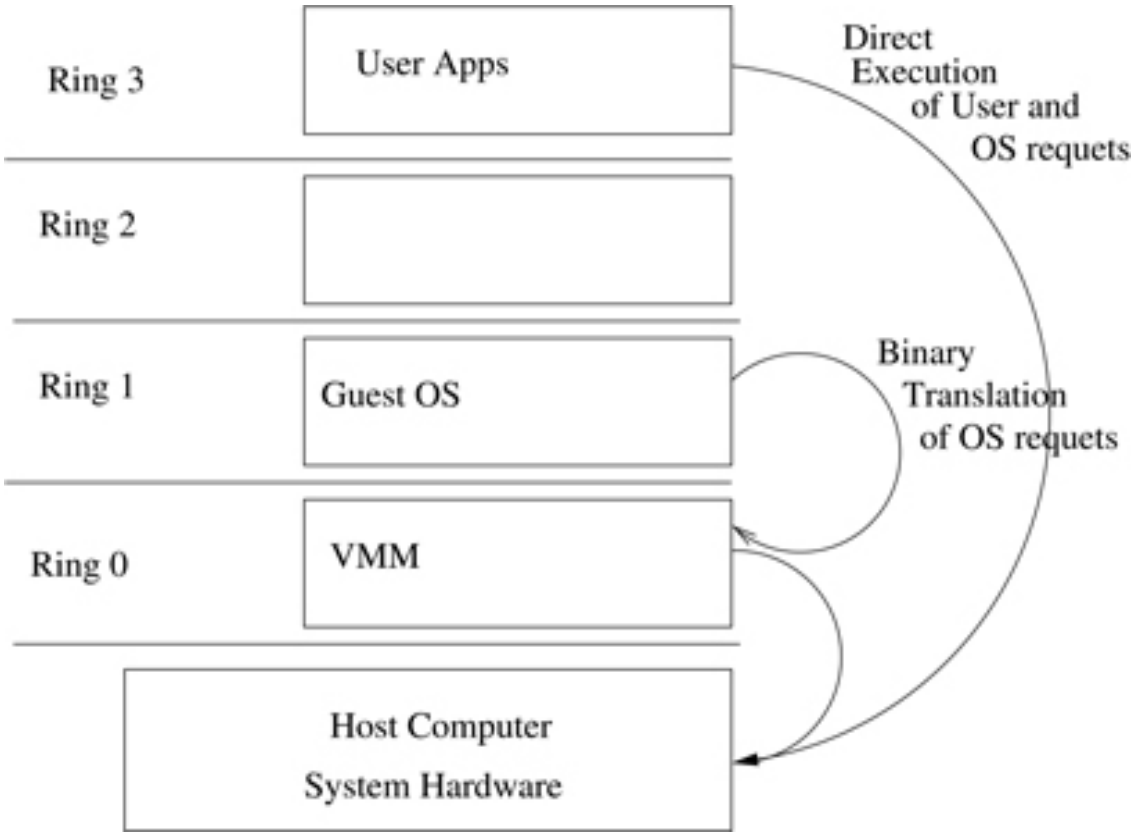

FIGURE 4.4
Binary translation approach to x86 virtualization

Clearly, with the hypervisor occupying ring 0 of the CPU, the kernels for any guest operating systems running on the system must run in less privileged CPU rings. Unfortunately, most operating system kernels are written explicitly to run in ring 0 for the simple reason that they need to perform tasks that are only available in that ring, such as the ability to execute privileged CPU instructions and directly manipulate memory. A number of different methodologies to this problem have been devised in recent years, each of which is given here.

- Binary translation
- Full virtualization
- Paravirtualization
- Hardware assisted virtualization

Binary translation is one specific approach to implement full virtualization that does not require hardware virtualization features. It involves examining the executable code of the virtual guest for unsafe instructions, translating these into safe equivalents, and then executing the translated code. Alternatives to binary translation are binary patching, and full system emulation.

Full Virtualization provides support for unmodified guest OS. The term unmodified refers to OS kernels which have not been altered to run on a hypervisor and therefore still execute privileged operations as though running in ring 0 of the CPU. The hypervisor provides CPU emulation to handle and modify privileged and protected CPU operations made by unmodified guest OS kernels. Unfortunately, this emulation process requires both time and system resources to operate, resulting in inferior performance levels when compared to those provided by para-virtualization.

Full Virtualization Using Binary Translation: This approach relies on binary translation to trap (into the VMM) and to virtualize certain sensitive and non-virtualizable instructions with new sequences of instructions that have the intended effect on the virtual hardware. Meanwhile, user level code is directly executed on the processor for high performance virtualization, as shown in **Figure 4.5**.

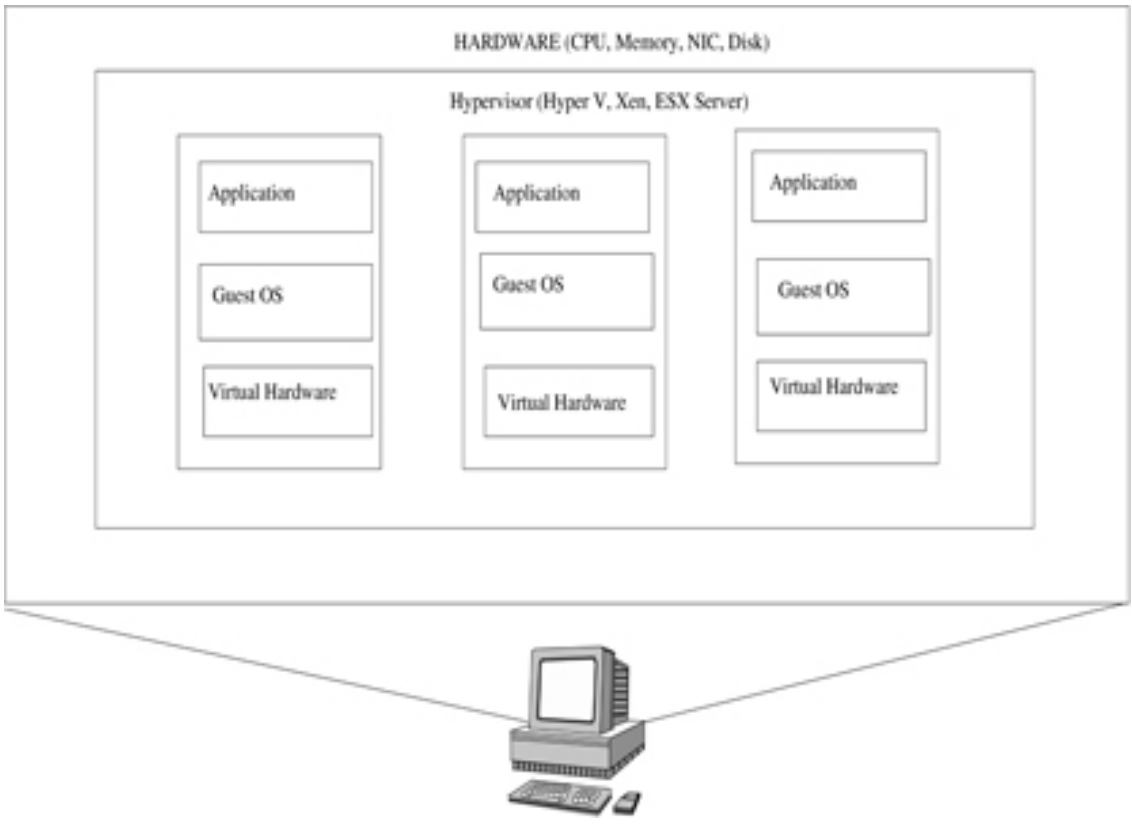

FIGURE 4.5
Logical Diagram of Full Virtualization

Paravirtualization is a technique that presents a software interface to virtual machine that is similar, but not identical to that of the underlying hardware. The intent of the modified interface is to reduce the portion of the guest's execution time spent performing operations which are sub-

stantially more difficult to run in a virtual environment compared to a non-virtualized environment. The paravirtualization provides specially defined “hooks” to allow the guest(s) and host to request and acknowledge these tasks, which would otherwise be executed in the virtual domain (where execution performance is worse). A successful paravirtualized platform may allow the virtual machine monitor (VMM) to be simpler (by relocating execution of critical tasks from the virtual domain to the host domain), and/or reduce the overall performance degradation of machine-execution inside the virtual-guest.

Paravirtualization requires the guest operating system to be explicitly ported for the para-API. A conventional OS distribution that is not paravirtualization-aware cannot be run on top of a paravirtualizing VMM. However, even in cases where the operating system cannot be modified, components may be available that enable many of the significant performance advantages of paravirtualization.

Under paravirtualization, the kernel of the guest OS is modified specifically to run on the hypervisor as shown in **Figure 4.6**. This typically involves replacing any privileged operations that only runs in ring 0 of the CPU with calls to the hypervisor (known as hypercalls). The hypervisor in turn performs the task on behalf of the guest kernel. This typically limits support to open source OS such as Linux which is freely altered and proprietary OS where the owners agree to make the necessary code modifications to target a specific hypervisor. These issues not withstanding, the ability of the guest kernel to communicate directly with the hypervisor results in greater performance levels than other virtualization approaches.

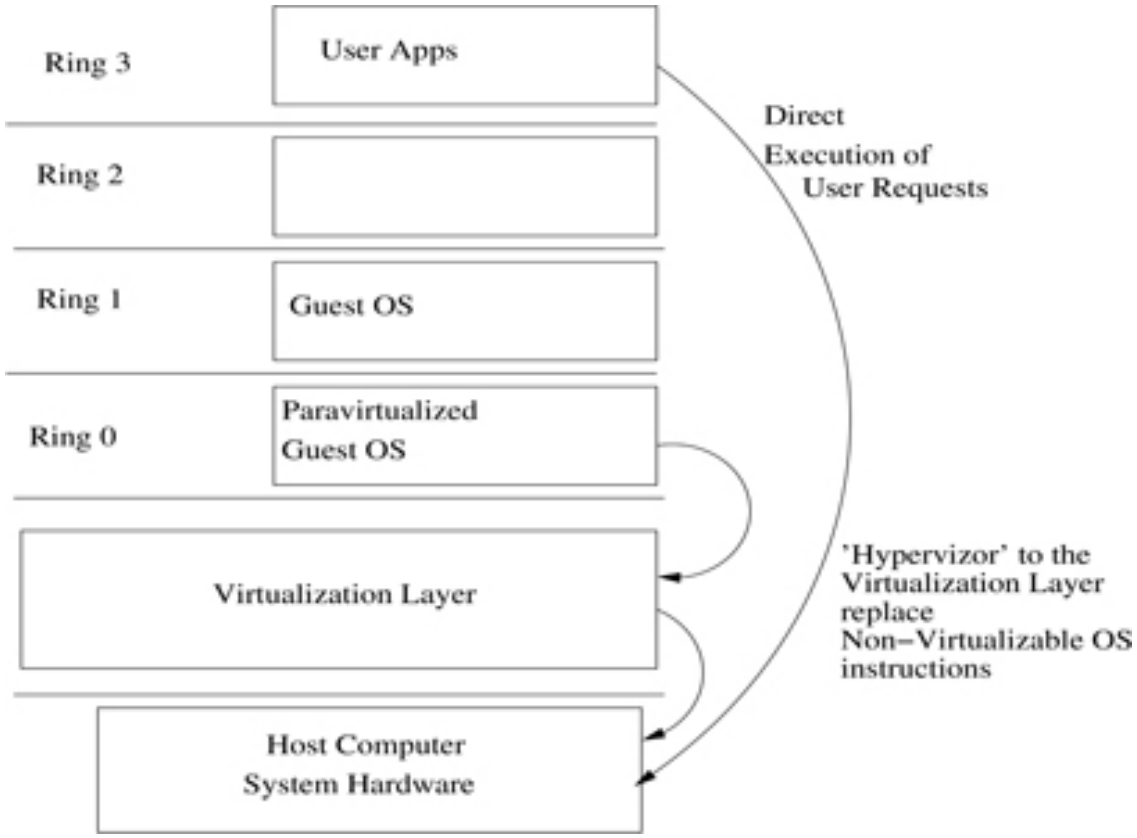

FIGURE 4.6
Paravirtualization approach to x86 virtualization

Hardware Virtualization

In computing, hardware-assisted virtualization is a plaform virtualization approach that enables efficient full virtualization using help from hardware capabilities, primarily from the host processors. Full virtualization is used to simulate a complete hardware environment, or virtual machine, in which an unmodified guest operating system (using the same instruction set as the host machine) executes in complete isolation. Hardware-assisted virtualization was added to x86 processors (Intel VT-x or AMD-V) in 2005 and 2006, respectively. Hardware-assisted virtualization is also known as accelerated virtualization; Xen calls it hardware virtual machine (HVM), Virtual Iron calls it native virtualization.

Hardware virtualization leverages virtualization features built into the latest generations of CPUs from both Intel and AMD. These technologies, known as Intel VT and AMD-V, respectively, provide extensions necessary to run unmodified guest VM without the overheads inherent in full virtualization CPU emulation. In very simplistic terms, these new processors provide an additional privilege mode above ring 0, in which the hypervi-

sor operates essentially leaving ring 0 available for unmodified guest OS. Figure 4.7 illustrates hardware assisted approach to x86 virtualization.

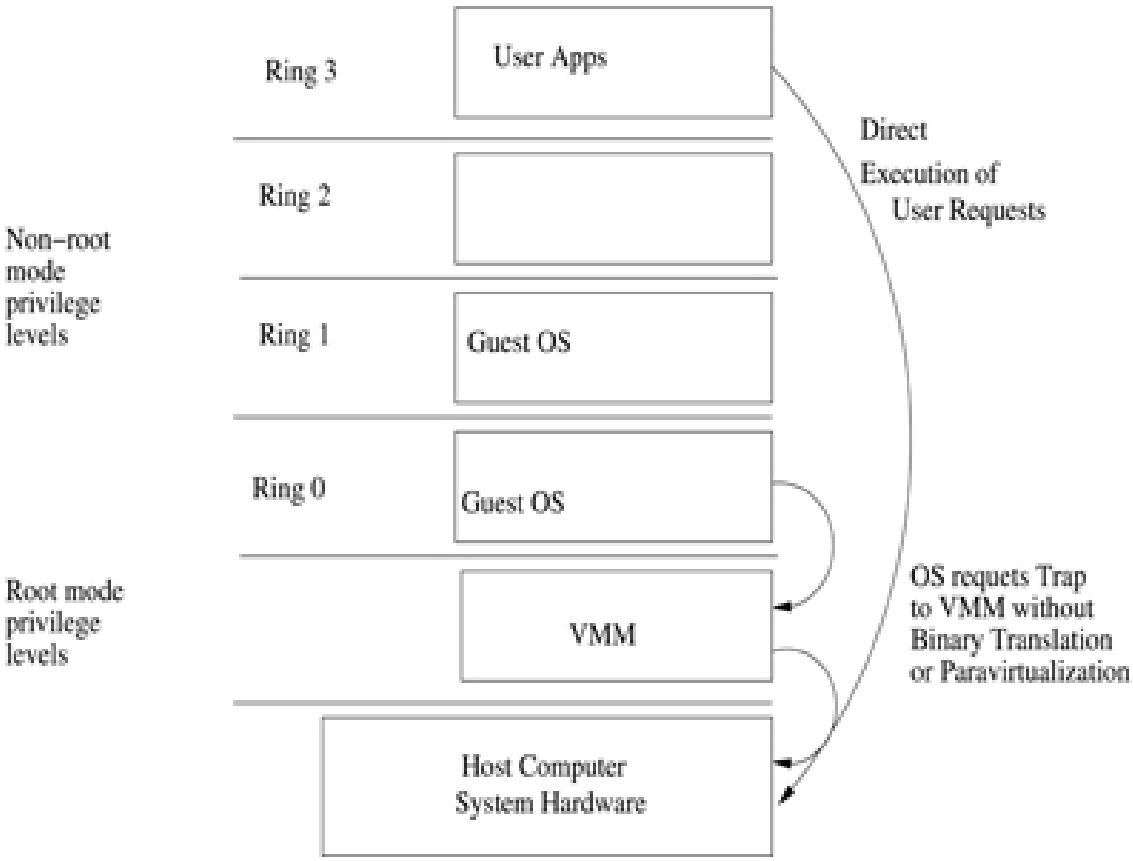

FIGURE 4.7
Hardware assisted approach to x86 virtualization

Intel's Virtualization Technology (VT-x) (e.g. Intel Xeon) and AMD's AMD-V both target privileged instructions with a new CPU execution mode feature that allows the VMM to run in a new root mode below ring 0, also referred to as Ring 0P (for privileged root mode) while the Guest OS runs in Ring 0D (for de-privileged non-root mode). Privileged and sensitive calls are set to automatically trap to the hypervisor and handled by hardware, removing the need for either binary translation or para-virtualization. VMware only takes advantage of these first generation hardware features in limited cases such as for 64-bit guest support on Intel processors.

4.3.2 Guest OS Virtualization

Guest OS virtualization is perhaps the easiest concept to understand. In this scenario, the physical host computer system runs a standard unmodified OS such as Windows, Linux, Unix or MacOS X. Running on this OS is a virtualization application which executes in much the same way as any

other application such as a word processor or spreadsheet would run on the system. It is within this virtualization application that one or more VMs are created to run the guest OS on the host computer. The virtualization application is responsible for starting, stopping and managing each VM and essentially controlling access to physical hardware resources on behalf of the individual VMs. The virtualization application also engages in a process known as binary rewriting which involves scanning the instruction stream of the executing guest system and replacing any privileged instructions with safe emulations. This has the effect of making the guest system think that, it is running directly on the system hardware, rather than in a VM inside an application.

Some examples of guest OS virtualization technologies include VMware Server and VirtualBox. **Figure 4.8** provides an illustration of guest OS-based virtualization. Here, the guest OS operate in VMs within the virtualization application which, in turn, runs on top of the host OS in the same way as any other application. Clearly, the multiple layers of abstraction between the guest OS and the underlying host hardware are not conducive to high levels of VM performance. This technique does, however, have the advantage that no changes are necessary to either host or guest OS and no special CPU hardware virtualization support is required.

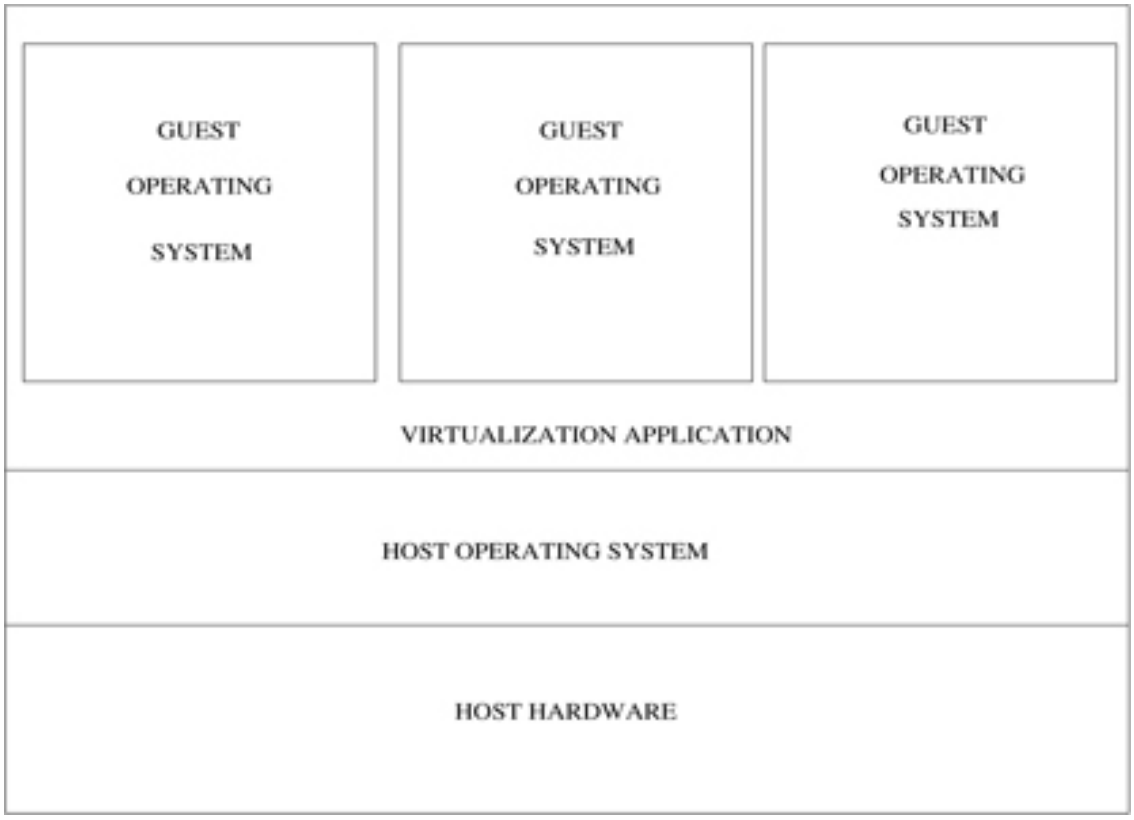

FIGURE 4.8
Guest OS virtualization

4.3.3 Shared Kernel Virtualization

The structure of shared kernel virtualization is illustrated in **Figure 4.9**. Shared kernel virtualization (also known as system level or operating system virtualization) takes advantage of the architectural design of Linux and UNIX based OS. In order to understand how shared kernel virtualization works, we need to first understand the two main components of Linux or UNIX OS. At the core of the OS is the kernel. The kernel, in simple terms, handles all the interactions between the OS and the physical hardware. The second key component is the root file system which contains all the libraries, files and utilities necessary for the OS to function. Under shared kernel virtualization, each of the virtual guest systems (virtual server in **Figure 4.9**) have their own root file system, but share the kernel of the host OS.

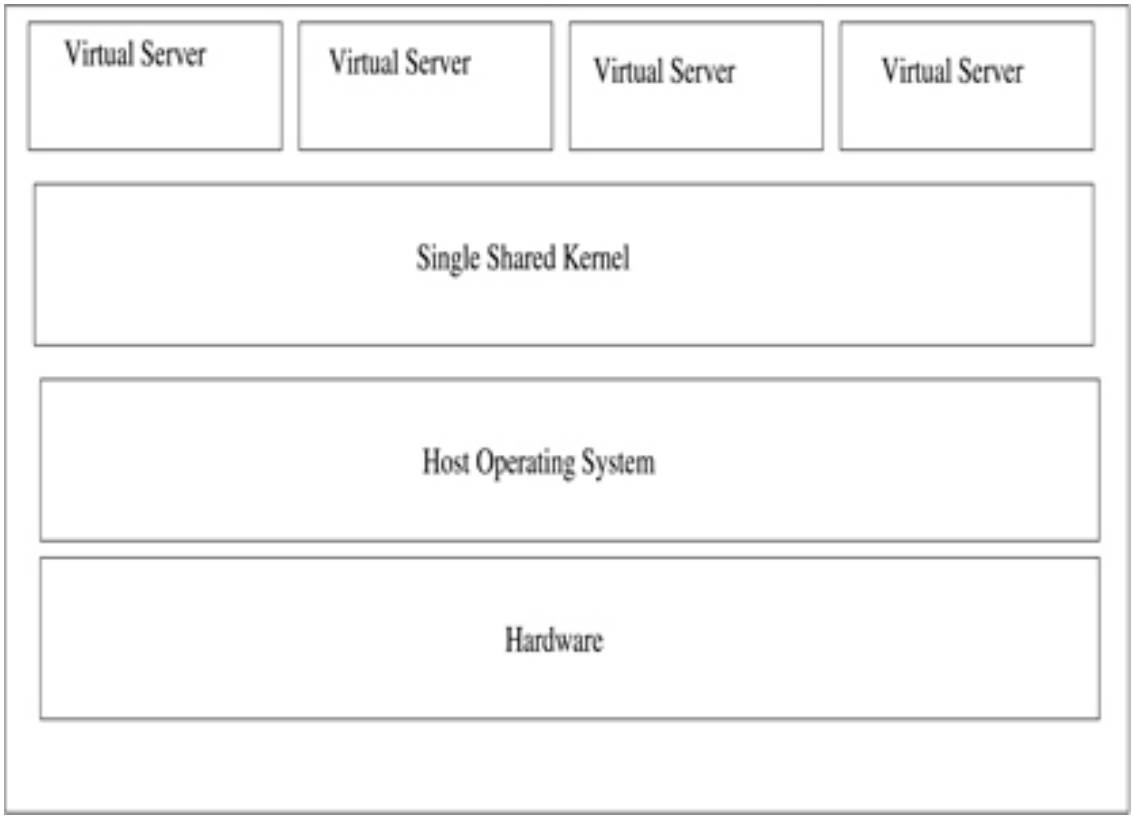

FIGURE 4.9
Shared kernel virtualization

4.3.4 Kernel Level Virtualization

Figure 4.10 provides an overview of the kernel level virtualization architecture. Under kernel level virtualization, the host OS runs on a specially modified kernel which contains extensions designed to manage and control multiple VMs each containing a guest OS. Unlike shared kernel virtualization, each guest (virtual machine in Figure 4.10) runs its own kernel, although similar restrictions apply in that the guest OS must have been compiled for the same hardware as the kernel in which they are running. Examples of kernel level virtualization technologies include User Mode Linux (UML) and Kernel-based Virtual Machine (KVM).

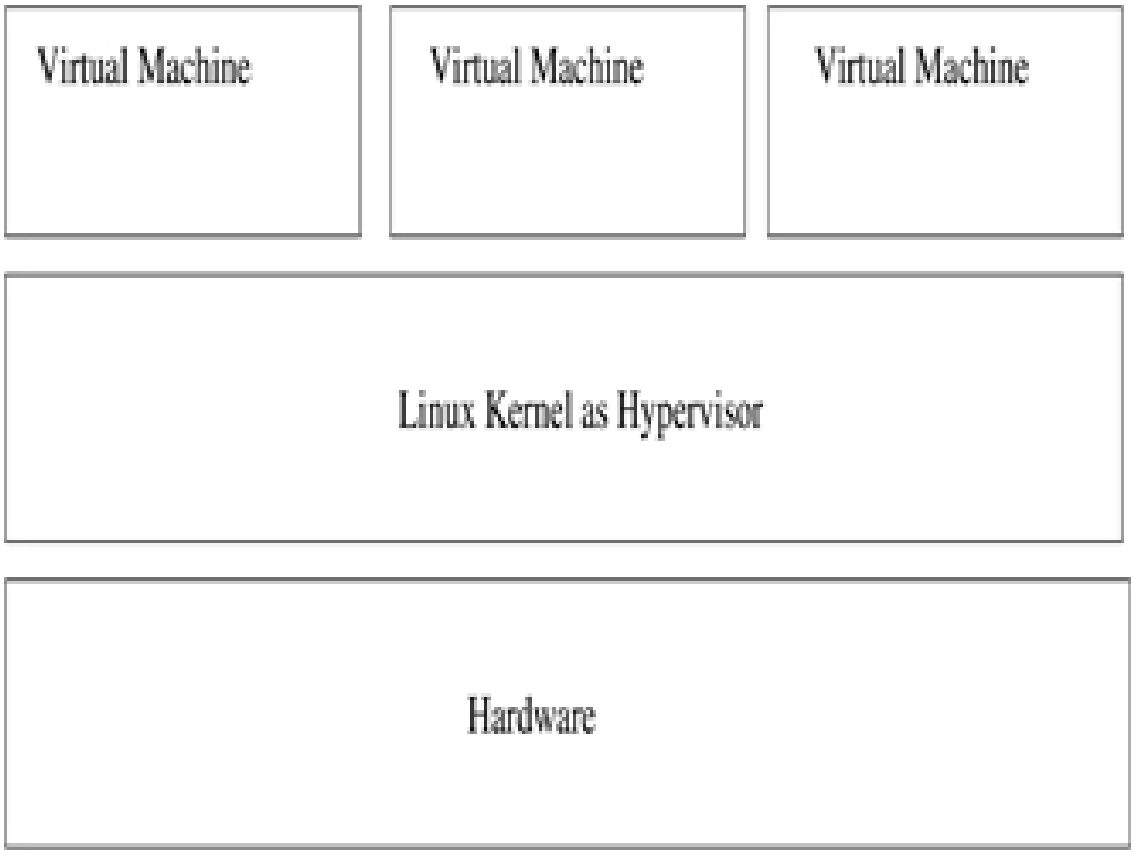

FIGURE 4.10
Kernel virtualization

4.4 Pros and Cons of Virtualization

Most common benefits of virtualization offered up by both organizations as well as vendors are seen to be the following.

It is cheaper: Because virtualization does not require actual hardware components to be used or installed, IT infrastructures find it to be a cheaper system to implement. There is no longer a need to dedicate large

areas of space and huge monetary investments to create an on-site resource. We just purchase the license or the access from a third-party provider and begin to work, just as if the hardware were installed locally.

It keeps costs predictable: Because third-party providers typically provide virtualization options, individuals and corporations can have predictable costs for their information technology needs.

It reduces the workload: Most virtualization providers automatically update their hardware and software that will be utilized. Instead of sending people to do these updates locally, they are installed by the third-party provider.

It offers a better uptime: Thanks to virtualization technologies, uptime has improved dramatically. Some providers offer an uptime that is 99.9999%. Even budget-friendly providers offer uptime at 99.99% today.

It allows for faster deployment of resources: Resource provisioning is fast and simple when virtualization is being used. There is no longer a need to set up physical machines, create local networks, or install other information technology components.

It promotes digital entrepreneurship: Before virtualization occurred on a large scale, digital entrepreneurship was virtually impossible for the average person. Thanks to the various platforms, servers, and storage devices that are available today, almost anyone can start their own side hustle or become a business owner.

It provides energy savings: For most individuals and corporations, virtualization is an energy-efficient system. Because there are not local hardware or software options being utilized, energy consumption rates can be lowered. Instead of paying for the cooling costs of a data center and the operational costs of equipment, funds can be used for other operational expenditures over time to improve virtualization's overall Return-on-Investment (ROI).

Disadvantages of Virtualization There can be some downsides to virtualization, as well. They are discussed below.

It can have a high cost of implementation: The cost for the average individual or business when virtualization is being considered will be quite low. For the providers of a virtualization environment, however, the implementation costs can be quite high.

It creates a security risk: Information is our modern currency. If you have it, you can make money. If you do not have it, you may be ignored. Because data is crucial to the success of a business, it is targeted frequently.

It creates an availability issue: The primary concern that many have with virtualization is what will happen to their work should their assets not be available. If an organization cannot connect to their data for an extended period of time, they will struggle to compete in their industry.

It creates a scalability issue: Although you can grow a business or opportunity quickly because of virtualization, you may not be able to become as large as you would like. You may also be required to be larger than you want to be when first starting out.

It requires several links in a chain that must work together cohesively: If you have local equipment, then you are in full control of what you can do. With virtualization, you lose that control because several links must work together to perform the same task.

It takes time: Although you save time during the implementation phases of virtualization, it costs users time over the long-run when compared to local systems.

It still has limitations: Not every application or server is going to work within an environment of virtualization. That means an individual or corporation may require a hybrid system to function properly.

The advantages and disadvantages of virtualization show us that it can be a useful tool for individuals, small and medium businesses, entrepreneurs, and corporations when it is used properly. Because it is so easy to use, however, some administrators begin adding new servers or storage for everything and that creates sprawl. By staying disciplined and aware

of communication issues, many of the disadvantages can be tempered, which is why this is such an effective modern system.

Summary

Nowadays, virtualization is a technology that is applied for sharing the capabilities of physical computers by splitting the resources among OSs. The concept of Virtual Machines (VMs) started back in 1964 with a IBM project called CP/CMS system. Currently, there are several virtualization techniques that can be used for supporting the execution of entire operating systems. In this chapter, we classified the virtualization techniques from the OS view. First, we discussed two techniques that executes modified guest OSs: operating system-level virtualization and para-virtualization. Second, we discussed two techniques that executes unmodified guest OSs: binary translation and hardware assisted. Finally, we highlighted on the prons and cons of virtualization.

Keywords

Virtualization	Binary translation
Paravirtualization	Software virtualization
Hardware virtualization	

Objective type questions

1. Point out the wrong statement:

(a) Abstraction enables the key benefit of Cloud computing: shared, ubiquitous access

(b) Virtualization assigns a logical name for a physical resource and then provides a pointer to that physical resource when a request is made

(c) All Cloud computing applications combine their resources into pools that can be assigned on demand to users