4.3. IMPLEMENTATION LEVELS OF VIRTUALIZATION

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The idea of VMs can be dated back to the 1960s .

The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility.

Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers. This virtualization technology has been revitalized as the demand for distributed and cloud computing increased sharply in recent years.

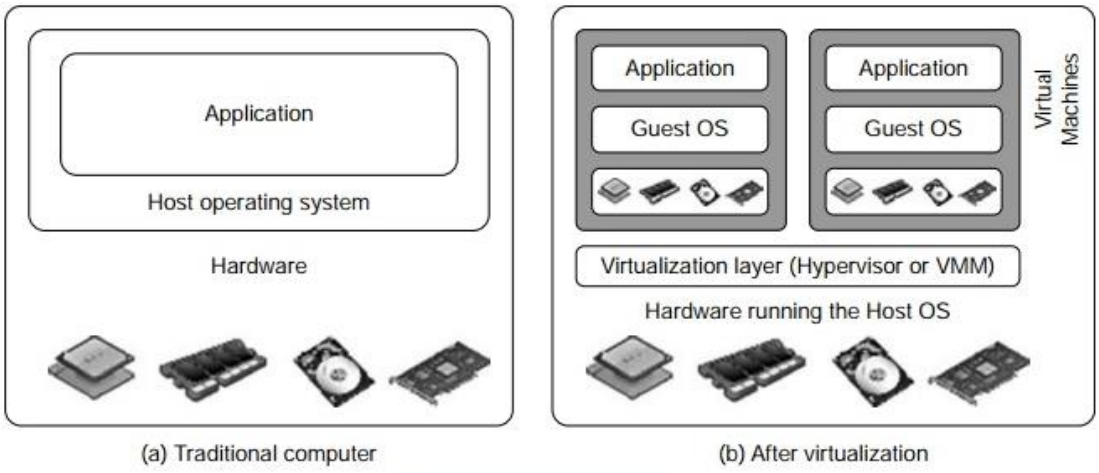
Levels of Virtualization Implementation

A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in After virtualization, different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS.

This is often done by adding additional software, called a virtualization layer as shown in Figure

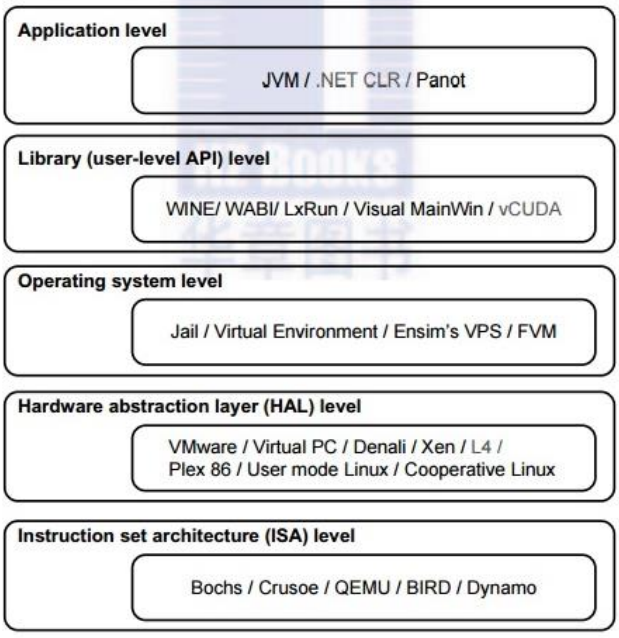
. This virtualization layer is known as hypervisor or virtual machine monitor (VMM) . The VMs are shown in the upper boxes, where applications run with their own guest OS over the virtualized CPU, memory, and I/O resources.

The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively. This can be implemented at various operational levels, as we will discuss shortly. The virtualization software creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system. Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level



GURE 3.1

The architecture of a computer system before and after virtualization, where VMM stands for virtual machine monitor.



4.5. **Architecture of Virtualization in Cloud Computing**

Virtualization in cloud computing helps create virtual versions of hardware, such as desktop computers, with a virtual ecosystem of operating systems, storage, memory, and networking services. The virtualization architecture uses the same hardware to run multiple operating systems on the same machine and optimize their performance.

What is Virtualization?

Virtualization plays an important role and function in cloud computing. It helps in reducing the space or costs associated with the investment. This technology

allows end users to run multiple desktop operating systems and applications simultaneously on the same hardware and software.

Virtualization in cloud computing simplifies the creation of virtual machines and makes it easier to run multiple machines. It also helps create a virtual ecosystem of server operating systems, multiple storage facilities, and multiple operating systems.

Cloud computing is an application or service associated with a virtual ecosystem. Such ecosystems can be public or private. Due to virtualization, the need for physical infrastructure can be reduced. The terms cloud computing and virtualization are now used interchangeably and are rapidly converging.

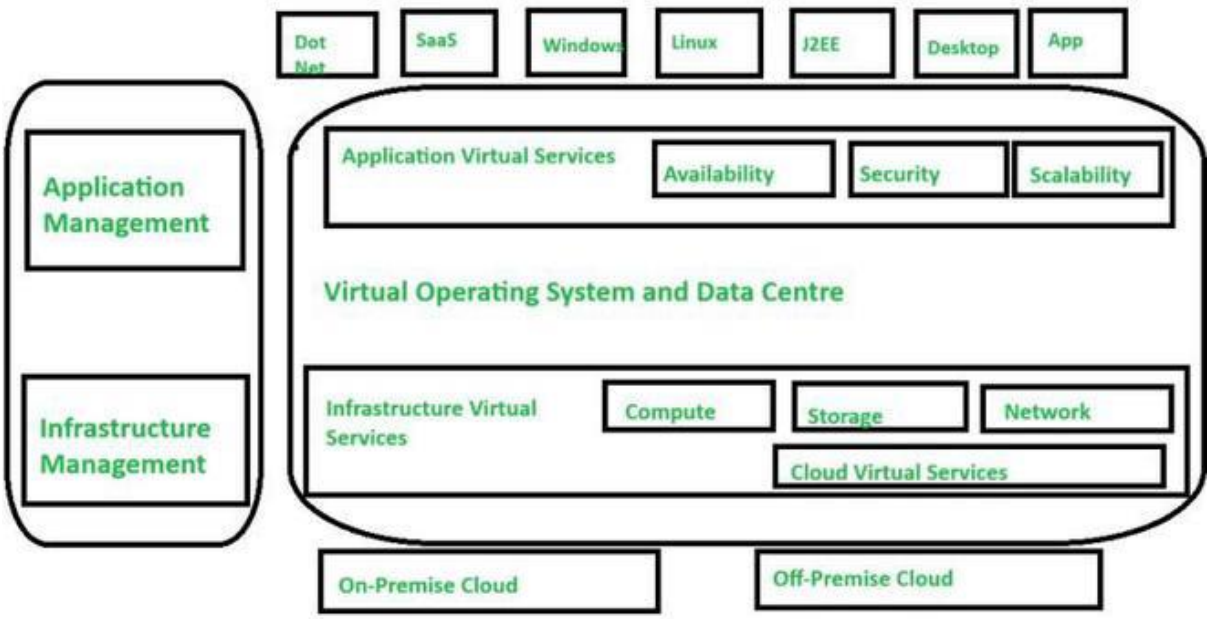
Important points of virtualization

There are some technologies in virtualization defined as follows:

- **Virtual Machine:** A virtual machine can be defined as a type of virtual machine that runs under the hypervisor of the computer.
- **Hypervisor:** It can be defined as a process running on real hardware. The virtual counterpart of the operating system is the underlying part that executes or emulates a virtual process.
- **Container:** They can be defined as a lightweight virtual machine that is a process of the same operating system (e.g. hypervisor).
- **Virtualization software:** This type of software helps in virtualizing the computer.
- **Virtual Network:** It is defined as a separate network and resides on the server. This type of network can be expanded to many servers.

What is Virtualization Architecture?

Virtualization Architecture is defined as a model that describes the concept of virtualization. The use of virtualization is important in cloud computing. In cloud computing, end users share data through an application called the cloud. However, end users can share their entire IT infrastructure with virtualization itself.



Architecture of the Virtualization

In the diagram above, virtualization includes virtual applications and virtual infrastructure services.

- The virtual application services help in application management, and the virtual infrastructure services can help in infrastructure management.
- Both services are integrated into the virtual data center or operating system. Virtual services can be used on any platform and programming environment. These services can be accessed from the local cloud or external cloud. In return, cloud users must pay a monthly or annual fee to the third party.
- This fee is paid to third parties for providing cloud services to end users, who in turn provide applications in different forms according to the needs of cloud end users.
- A hypervisor separates the operating system from the underlying hardware. It allows the host computer to run multiple virtual machines simultaneously and share the same computer resources.

Cloud computing and Virtualization Architecture

Virtualization in cloud computing helps create virtual versions of hardware such as desktop computers with a virtual ecosystem of operating systems, storage, memory, and networking services. Virtualization architecture uses the same hardware to run multiple operating systems on the same machine and optimize their performance.

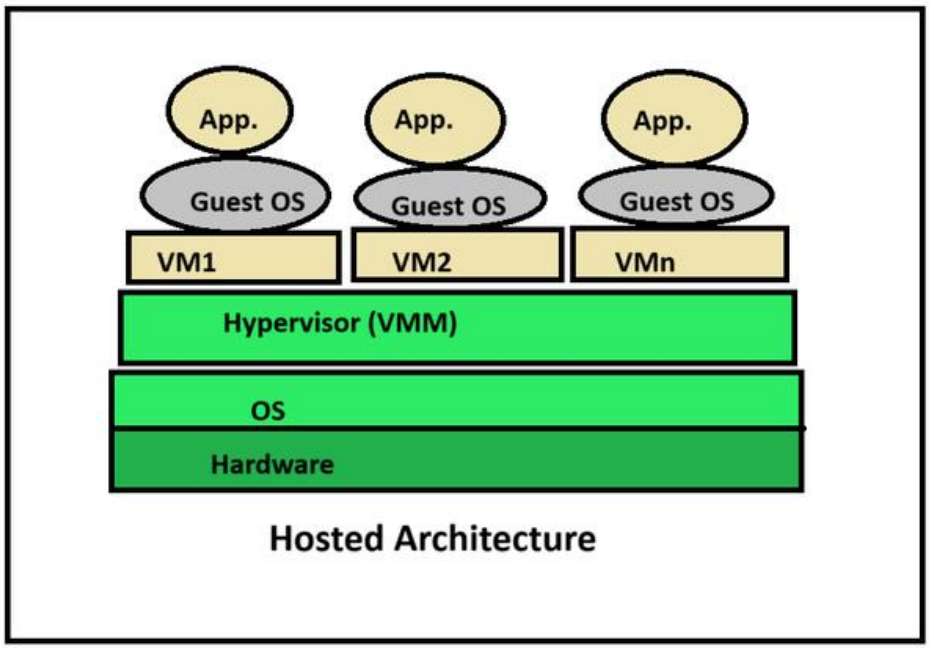
Virtualization and virtualization architecture are important concepts in cloud computing. In fact, since the definition of cloud computing also includes virtual ecosystems, these terms are often used interchangeably. Whether the ecosystem is private (i.e., cloud) or public (public cloud), virtualization reduces the need for organizations to maintain physical (on-premises) infrastructure for their computing needs. With cloud computing and virtualization architecture, applications can be shared with many active users. With a public cloud like Amazon Web Services (AWS) or Microsoft Azure, these can be shared with multiple businesses.

Types of Virtualization Architectures

There are two main types of virtualization architectures: hosted and bare metal.

Hosted Architecture

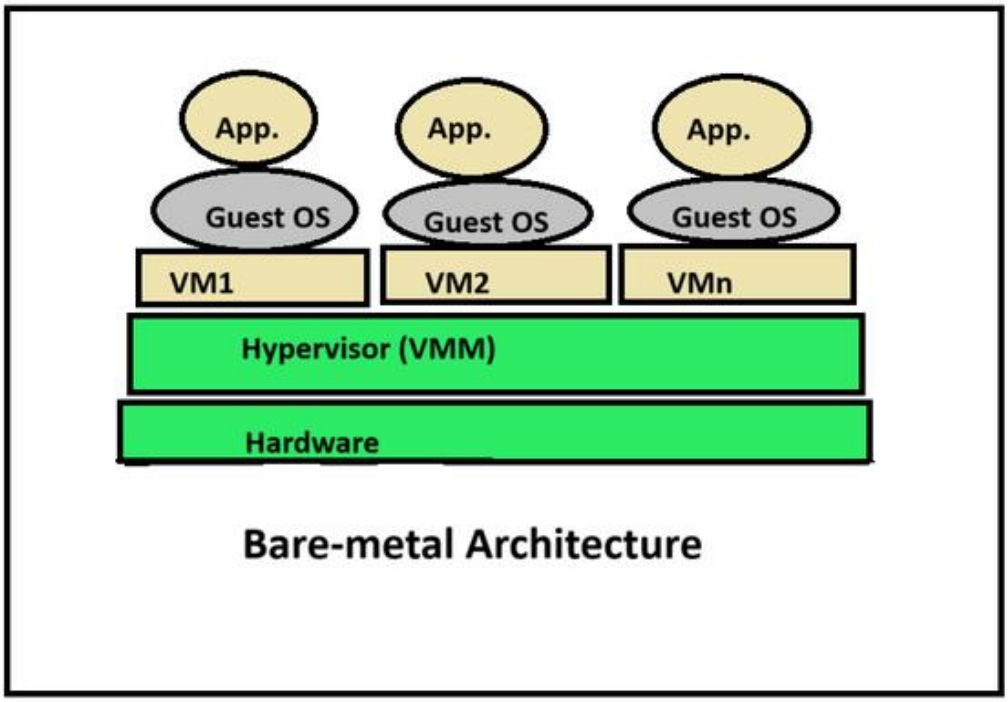
In this type of configuration, first, the host operating system is installed on the hardware, then the software is installed. The software is a hypervisor or virtual machine (VM) that requires many guest operating systems or VMs to be installed on the hardware to set up the virtualization architecture. Once the hypervisor is in place, applications can be installed and run on the virtual machine as if they were installed on the physical machine.



Hosted Architecture

Bare Metal Architecture

In this architecture, the hypervisor is installed directly on the hardware, not on top of the operating system. Hypervisors and virtual machines are configured the same way as infrastructure. Bare metal virtualization architecture is designed for applications that provide real-time access or perform some form of data processing.



More about Hypervisors in Virtualization Architecture

Virtualization is hypervisor based. A hypervisor separates the operating system and applications from the underlying computer hardware so that the host computer can run multiple virtual machines as guests and share physical resources such as network processor, memory space, and network bandwidth. A hypervisor allocates memory or storage services and distributes some of these services to each virtual machine according to the needs of the virtualization architecture.

- **Type 1 hypervisors:** Sometimes called bare metal hypervisors, run directly on top of the host system hardware. The bare metal hypervisor provides a high level of control and management. Their direct access to system hardware provides better performance, scalability and stability. Examples include: Microsoft Hyper-V, Citrix XenServer, VMware ESXi.

- **Type 2 hypervisors:** (also called management hypervisors) are installed on the host operating system rather than directly on the hardware like Type 1 hypervisors. All guest operating systems or virtual machines run on top of the hypervisor. The simplicity of the host operation is known to facilitate the installation and management of the project. But adding a layer of layering can limit functionality and introduce a security vulnerability. Example: VMware Workstation Pro, VMware Fusion, Oracle VirtualBox, Oracle Solaris Zones, Oracle VM Server for x86.

Benefits of a Virtualization Architecture

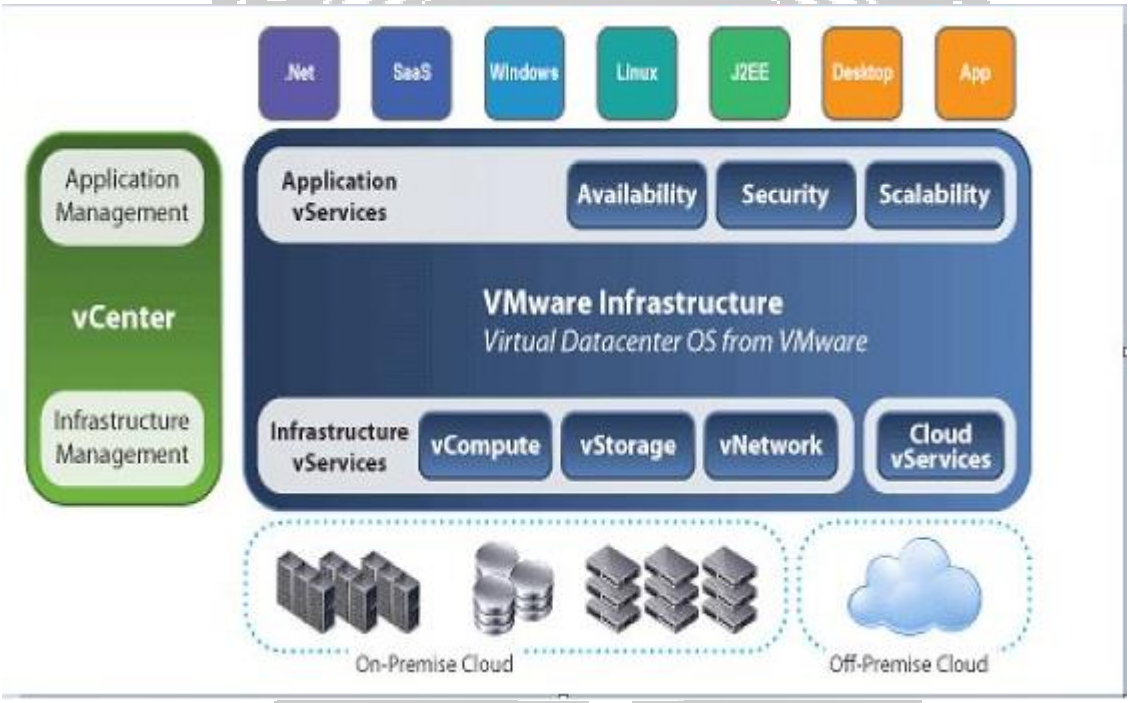
- Virtualization architecture provides cloud-native organizations with a flexible, cost-effective, and versatile way to run multiple virtual machines or systems using a single service or host. As a result, it provides efficient and economical deployment and reduces IT infrastructure costs.
- Since the host computer user can limit the number of users of virtual resources, the environment provides better control while also reducing energy and resource consumption.
- Virtualization also enables remote, anytime, anywhere access, which is essential for remote or isolated locations.
- It also reduces the need for physical equipment such as servers, resulting in more uptime and fewer interruptions, better error management, and more efficient and capable product balance. Resources can be replaced or removed as needed, providing business teams with greater scalability and agility.
- All major cloud service providers offer virtualization solutions based on a one-time payment model. This ensures that organizations only pay for resources that are actually used, not for resources that are not used or have no capacity or potential. As a result, they have more control over the weather budget and spending.
- Virtualization is also beneficial from a security perspective. You can configure tasks, devices, applications, etc. on each virtual machine. There may be more than one guest user who can run it. This extraction helps protect sensitive or business-critical data.

Virtualization and Containerization

The alternative to hypervisor-based virtualization in cloud computing is containerization. For example, operating system virtualization is an important volume-based virtualization approach. In this architecture, the operating system

is set up to act as multiple separate systems, allowing distributed applications to be deployed and run without having to turn on the entire virtual machine for each system. Instead, multiple isolated machines (called volumes) run on a single host, each accessing a single key.

Like virtual machines, container is a way to create virtual packages. Essentially, a container is a lightweight virtual machine that is part of the same operating system instance or hypervisor. But virtualization is a way to run multiple operating systems on a single physical server. Containerization, on the other hand, is a way of running multiple applications on the same machine in a virtual machine. VMs are best suited for applications that require full-scale performance. Containers are a better choice for short-term projects that need to reduce the number of servers used by multiple applications



1. Hypervisor and Xen Architecture

The hypervisor supports hardware-level virtualization (see Figure 3.1(b)) on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physi-cal hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor. The hypervisor provides hypercalls for the guest OSes and applications. Depending on the functional-ity, a hypervisor can assume a micro-kernel architecture like the Microsoft Hyper-V. Or it can assume a monolithic hypervisor architecture like the VMware ESX for server virtualization.

A micro-kernel hypervisor includes only the basic and unchanging functions (such as physical memory management and processor scheduling). The device drivers and other changeable components are outside the hypervisor. A monolithic hypervisor implements all the aforementioned functions, including those of the device drivers. Therefore, the size of the hypervisor code of a micro-kernel hyper-visor is smaller than that of a monolithic hypervisor. Essentially, a hypervisor must be able to convert physical devices into virtual resources dedicated for the deployed VM to use.

1.1 The Xen Architecture

Xen is an open source hypervisor program developed by Cambridge University. Xen is a micro-kernel hypervisor, which separates the policy from the mechanism. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0, as shown in Figure 3.5. Xen does not include any device drivers natively [7]. It just provides a mechanism by which a guest OS can have direct access to the physical devices. As a result, the size of the Xen hypervisor is kept rather small. Xen provides a virtual environment located between the hardware and the OS. A number of vendors are in the process of developing commercial Xen hypervisors, among them are Citrix XenServer [62] and Oracle VM [42].

The core components of a Xen system are the hypervisor, kernel, and applications. The organi-zation of the three components is important. Like other virtualization systems, many guest OSes can run on top of the hypervisor.

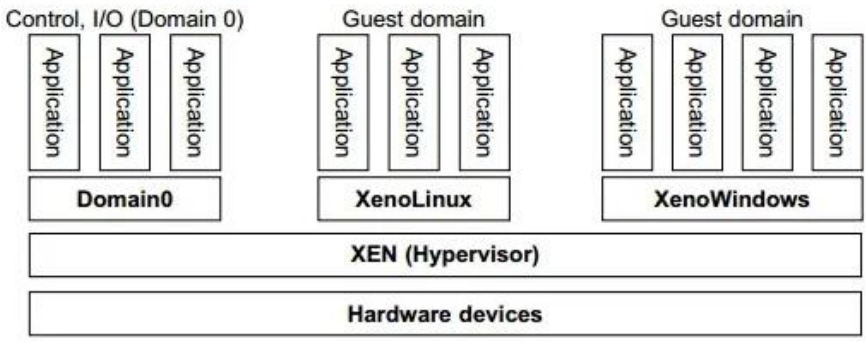


FIGURE 3.5

The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

2. Binary Translation with Full Virtualization

Depending on implementation technologies, hardware virtualization can be classified into two cate-gories: full virtualization and host-based virtualization. Full virtualization does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive,

nonvirtualizable instructions. The guest OSes and their applications consist of noncritical and critical instructions. In a host-based system, both a host OS and a guest OS are used. A virtuali-zation software layer is built between the host OS and guest OS. These two classes of VM architec-ture are introduced next.

2.1 Full Virtualization

With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software. Both the hypervisor and VMM approaches are considered full virtualization. Why are only critical instructions trapped into the VMM? This is because binary translation can incur a large performance overhead. Noncritical instructions do not control hardware or threaten the security of the system, but critical instructions do. Therefore, running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security.

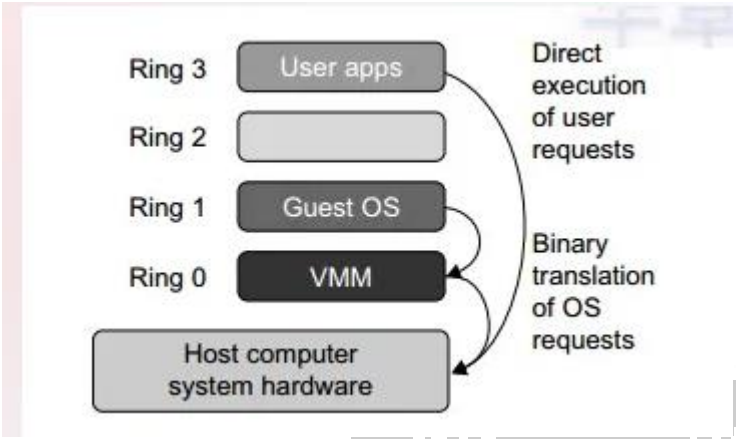
2.2 Binary Translation of Guest OS Requests Using a VMM

This approach was implemented by VMware and many other software companies. As shown in Figure 3.6, VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation. Therefore, full vir-tualization combines binary translation and direct execution. The guest OS is completely decoupled from the underlying hardware. Consequently, the guest OS is unaware that it is being virtualized.

The performance of full virtualization may not be ideal, because it involves binary translation which is rather time-consuming. In particular, the full virtualization of I/O-intensive applications is a really a big challenge. Binary translation employs a code cache to store translated hot instructions to improve performance, but it increases the cost of memory usage. At the time of this writing, the performance of full virtualization on the x86 architecture is typically 80 percent to 97 percent that of the host machine.

2.3 Host-Based Virtualization

An alternative VM architecture is to install a virtualization layer on top of the host OS. This host OS is still responsible for managing the hardware. The guest OSes are installed and run on top of the virtualization layer. Dedicated applications may run on the VMs. Certainly, some other applications



3. Para-Virtualization with Compiler Support

Para-virtualization needs to modify the guest operating systems. A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. Performance degradation is a critical issue of a virtualized system. No one wants to use a VM if it is much slower than using a physical machine. The virtualization layer can be inserted at different positions in a machine soft-ware stack. However, para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel.

Para-Virtualization Architecture

When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS. According to the x86 ring definition, the virtualization layer should also be installed at Ring 0. Different instructions at Ring 0 may cause some problems. In Figure 3.8, we show that para-virtualization replaces nonvirtualizable instructions with hypercalls that communicate directly with the hypervisor or VMM. However, when the guest OS kernel is modified for virtualization, it can no longer run on the hardware directly.

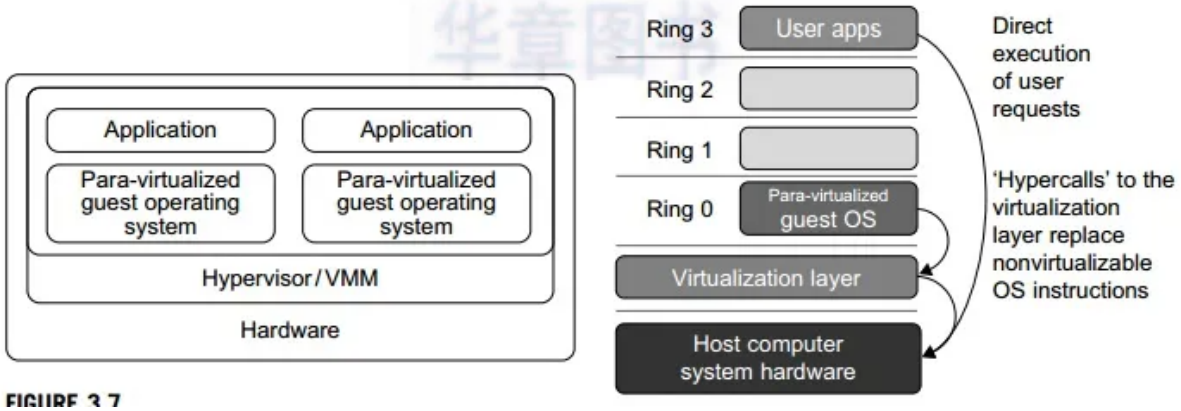


FIGURE 3.7

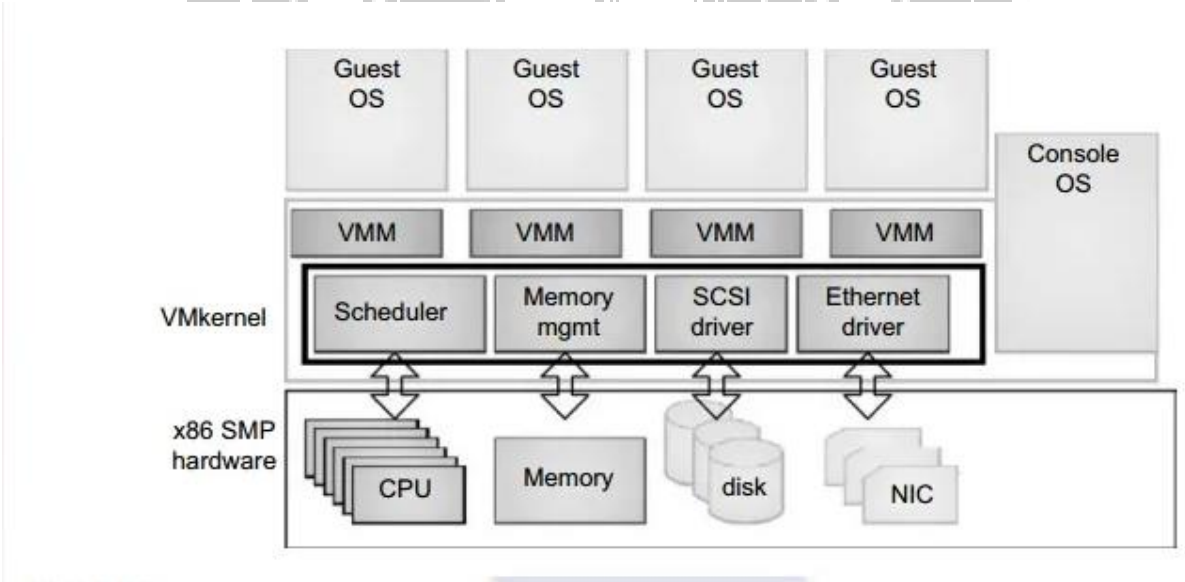
3.2 KVM (Kernel-Based VM)

This is a Linux para-virtualization system—a part of the Linux version 2.6.20 kernel. Memory management and scheduling activities are carried out by the existing Linux kernel. The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine. KVM is a hardware-assisted para-virtualization tool, which improves performance and supports unmodified guest OSes such as Windows, Linux, Solaris, and other UNIX variants.

3.3 Para-Virtualization with Compiler Support

Unlike the full virtualization architecture which intercepts and emulates privileged and sensitive instructions at runtime, para-virtualization handles these instructions at compile time. The guest OS kernel is modified to replace the privileged and sensitive instructions with hypercalls to the hypervi-sor or VMM. Xen assumes such a para-virtualization architecture.

The VMM layer virtualizes the physical hardware resources such as CPU, memory, network and disk controllers, and human interface devices. Every VM has its own set of virtual hardware resources. The resource manager allocates CPU, memory disk, and network bandwidth and maps them to the virtual hardware resource set of each VM created. Hardware interface components are the device drivers and the



VMware ESX Server File System. The service console is responsible for booting the system, initiating the execution of the VMM and resource manager, and relinquishing control to those layers. It also facilitates the process for system administrators.