**UNIT- II**

# VIRTUALIZATION

- Basics of Virtualization
- Types of Virtualization
- Implementation Levels of Virtualization
- Virtualization Structures / Tools and Mechanisms
- Virtualization of CPU, Memory, I/O Devices
- Virtual Clusters and Resource management
- Virtualization for Data-center Automation

## ⇒ Basics of Virtualization

- In computing, **virtualization** refers to the act of creating a virtual **(rather than actual)** version of something, including virtual computer hardware platforms, storage devices, and computer network resources.

- Virtualization technology is one of the fundamental components of cloud computing, especially in regard to infrastructure based service**. "Virtualization allows the creation of a secure, customizable, and isolated execution environment for running applications, even if they are un-trusted, without affecting other user's applications".**

- The term virtualization is often synonymous with hardware virtualization, which plays a fundamental role in efficiently delivering Infrastructure-as-a-Service (IaaS) solutions for cloud computing.

### Why Virtualization needed?

- ✓ Increased performance and computing capacity.
- ✓ Underutilized hardware and software resources.
- ✓ Lack of space.
- ✓ Greening initiatives.
- ✓ Rise of administrative costs.

### Characteristics of virtualized environments

"Virtualization is a broad concept that refers to the creation of a virtual version of something, whether hardware, a software environment, storage, or a network". In a virtualized environment there are three major components: **guest, host, and virtualization layer**. The guest represents the system component that interacts with the virtualization layer rather than with the host, as would normally happen. The host represents the original environment where the guest is supposed to be managed. The virtualization layer is responsible for recreating the same or a different environment where the guest will operate (see Figure 3.1).

- **Increased security:** The ability to control the execution of a guest in a completely transparent manner opens new possibilities for delivering a secure, controlled execution environment.

- **Managed execution:** Virtualizationoftheexecutionenvironmentnotonlyallowsincreasedsecurity, but a wide range of features also can be implemented. In particular, sharing, aggregation, emulation, and isolation are the most relevant features (see Figure3.2).

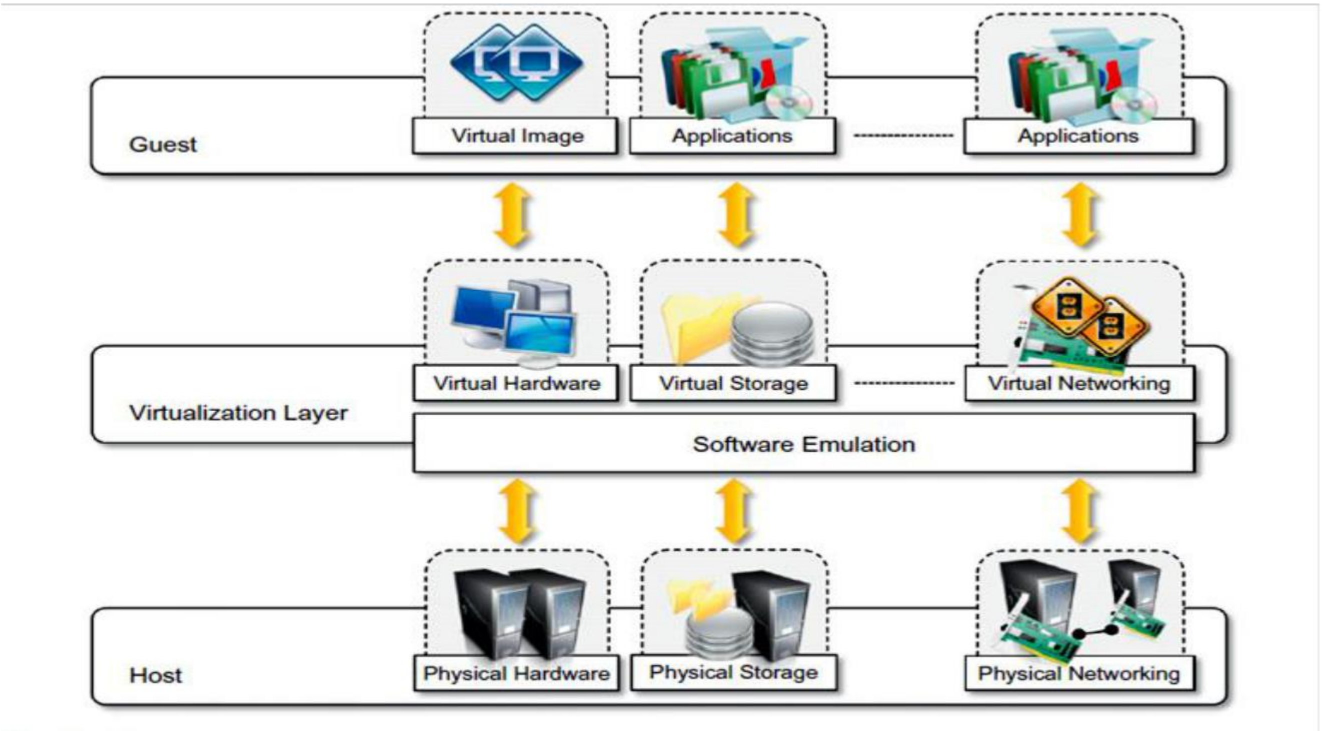


FIGURE 3.1
The virtualization reference model.

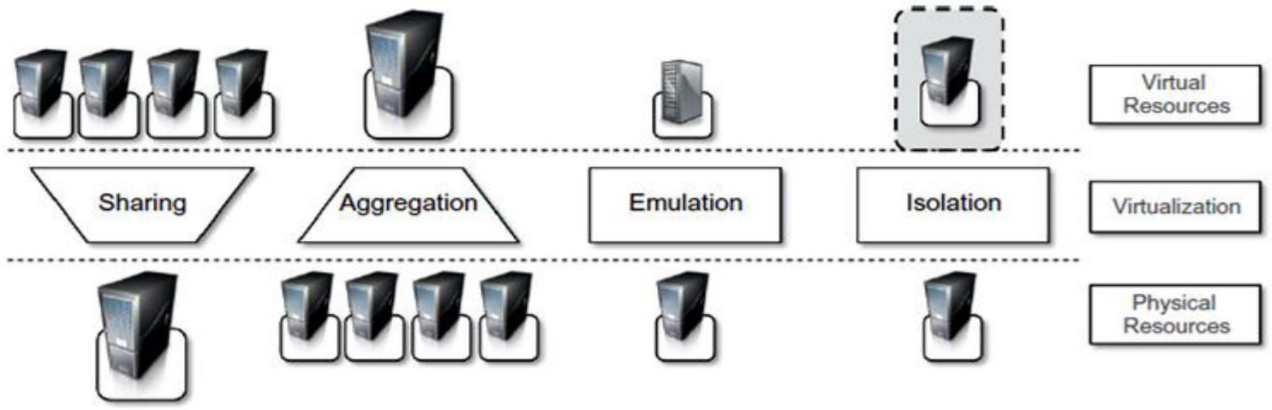


FIGURE 3.2
Functions enabled by managed execution.

- **Portability:** The concept of portability applies in different ways according to the specific type of virtualization considered. In the case of a hardware virtualization solution, the guest is packaged in to a virtual image that, in most cases, can be safely moved and executed on top of different virtual machines.

## Benefits of Virtualization

1. More flexible and efficient allocation of resources.
2. Enhance development productivity.

3. It lowers the cost of IT infrastructure.
4. Remote access and rapid scalability.
5. High availability and disaster recovery.
6. Pay per use of the IT infrastructure on demand.
7. Enables running multiple operating system.

## ⇒ Types of Virtualization

### 1) Application Virtualization:

Application virtualization helps user to have a remote access of an application from a server. The server stores all personal information and other characteristics of the application, but can still run on a local workstation through internet. Example of this would be a user who needs to run two different versions of the same software. Application virtualization is abstracting the application layer away from the operating system. This way the application can run in an encapsulated form without being depended upon on the operating system underneath. This can allow a Windows application to run on Linux and vice versa, in addition to adding a level of isolation.

### 2) Network Virtualization:

The ability to run multiple virtual networks that each has a separate control and data plan. It co-exists together on top of one physical network. It can be managed by individual parties that potentially confidential to each other. Network virtualization, provides a facility to create and provision virtual networks logical switches, routers, firewalls, load balancer, Virtual Private Network (VPN), and workload security within days or even in weeks.

Network virtualization is a method of combining the available resources in a network by splitting up the available bandwidth into channels, each of which is independent from the others and can be assigned -- or reassigned -- to a particular server or device in real time.

### 3) Desktop Virtualization:

Desktop virtualization allows the users' OS to be remotely stored on a server in the data center. It allows the user to access their desktop virtually, from any location by different machine. Users who want specific operating systems other than Windows Server will need to have a virtual desktop. Main benefits of desktop virtualization are user mobility, portability, easy management of software installation, updates and patches.

### 4) Storage Virtualization:

Storage virtualization is an array of servers that are managed by a virtual storage system. The servers aren't aware of exactly where their data is stored, and instead function more like worker bees in a hive. It makes managing storage from multiple sources to be managed and utilized as a single repository. Storage virtualization software maintains smooth operations, consistent performance and a continuous suite of advanced functions despite changes, breaks down and differences in the underlying equipment. Storage virtualization is commonly used in storage area networks.

### 5) Server virtualization:

It is the **masking of server resources** -- including the number and identity of individual physical servers, processors and operating systems from server users. The intention is to spare the user from having to understand and manage complicated details of server resources while increasing resource sharing and utilization and maintaining the capacity to expand later.

6) **Data virtualization:**

It is abstracting the traditional technical details of data and data management, such as location, performance or format, in favor of broader access and more resiliencies tied to business needs.

## ⇒ Implementation Levels of Virtualization

✓ Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.

✓ 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.

A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in Figure 3.1(a). 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 3.1(b). 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.

Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level (see Figure 3.2).

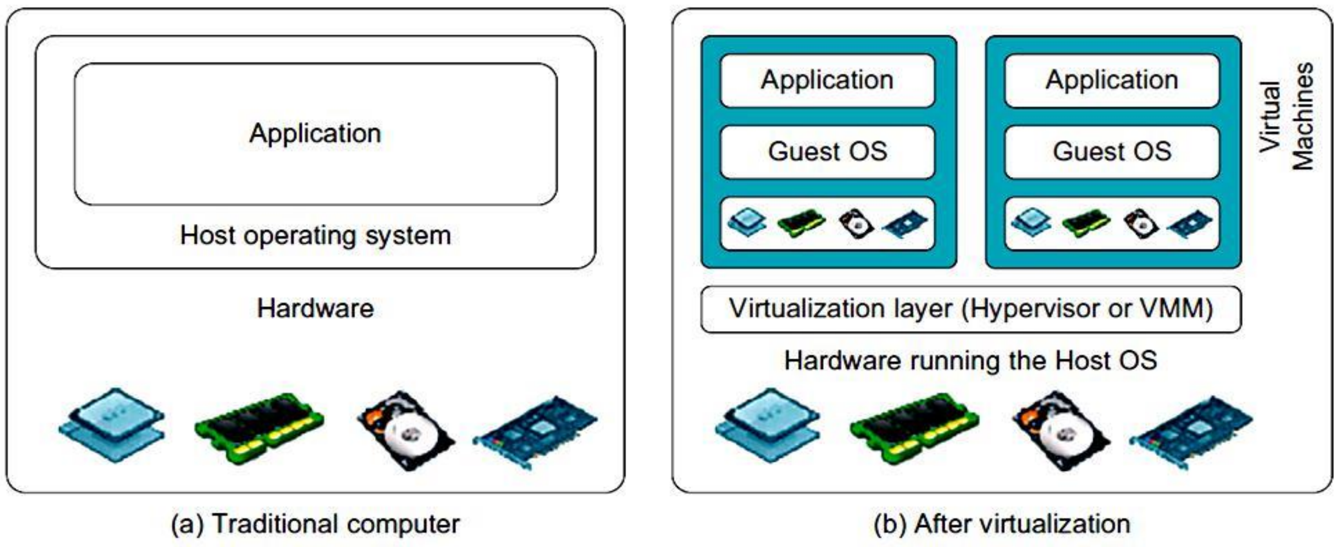
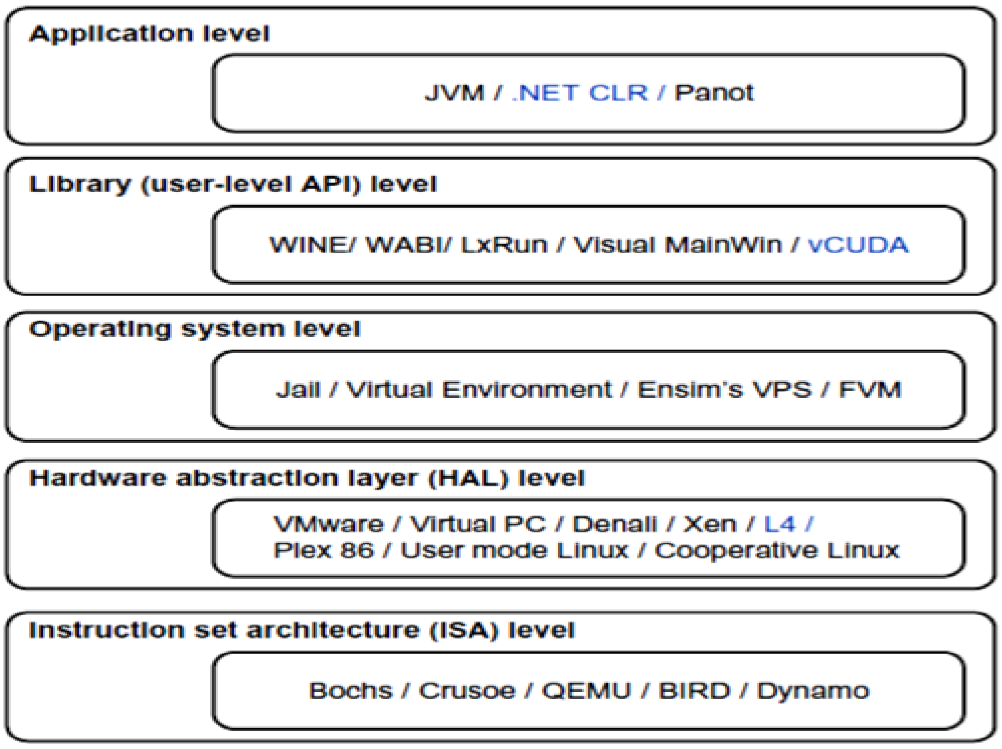


**FIGURE 3.1**

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


Application level
JVM / .NET CLR / Panot
Library (user-level API) level
WINE/ WABI/ LxRun / Visual MainWin / vCUDA
Operating system level
Jail / Virtual Environment / Ensim's VPS / FVM
Hardware abstraction layer (HAL) level
VMware / Virtual PC / Denali / Xen / L4 / Plex 86 / User mode Linux / Cooperative Linux
Instruction set architecture (ISA) level
Bochs / Crusoe / QEMU / BIRD / Dynamo

1. **Instruction Set Architecture Level**

At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. With this approach, it is possible to run a large amount of legacy binary code written for various processors on any given new hardware host machine. Instruction set emulation leads to virtual ISAs created on any hardware machine.

2. **Hardware Abstraction Level**

Hardware-level virtualization is performed right on top of the bare hardware. The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices. Hardware-level virtualization inserts a layer between real hardware and traditional operating systems. This layer is commonly called the Virtual Machine Monitor (VMM) and it manages the hardware resources of a computing system.

3. **Operating System Level**

This refers to an abstraction layer between traditional OS and user applications. OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers. The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.

4. **Library Support Level**

Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS. Since most systems provide well-documented APIs, such an interface becomes another candidate for virtualization.

5. **User-Application Level**

Virtualization at the application level virtualizes an application as a VM. On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. Other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming. The process involves wrapping the application in a layer that is isolated from the host OS and other applications. The result is an application that is much easier to distribute and remove from user workstations.

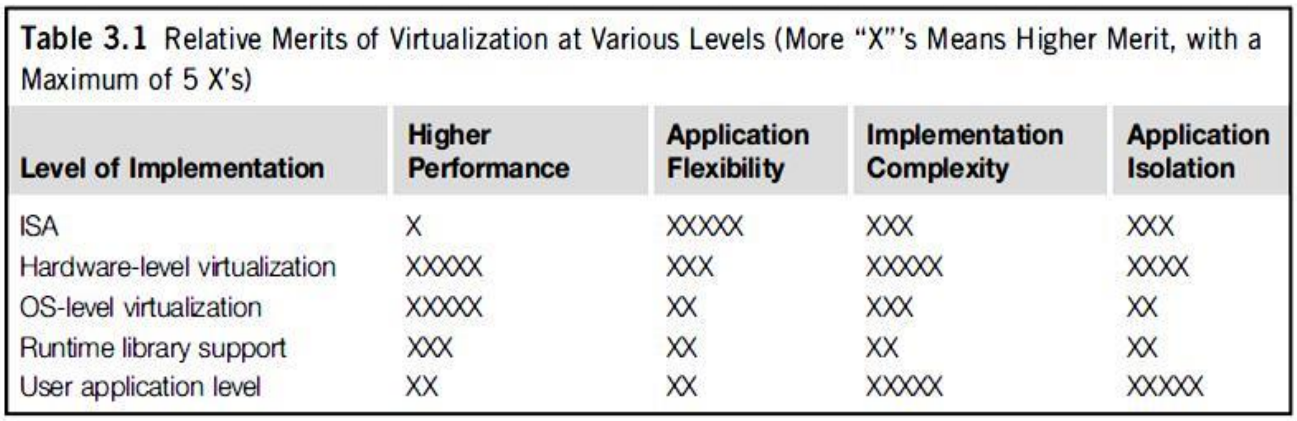


**Table 3.1** Relative Merits of Virtualization at Various Levels (More "X"'s Means Higher Merit, with a Maximum of 5 X's)

| Level of Implementation | Higher Performance | Application Flexibility | Implementation Complexity | Application Isolation |
|---|---|---|---|---|
| ISA | X | XXXXX | XXX | XXX |
| Hardware-level virtualization | XXXXX | XXX | XXXXX | XXXX |
| OS-level virtualization | XXXXX | XX | XXX | XX |
| Runtime library support | XXX | XX | XX | XX |
| User application level | XX | XX | XXXXX | XXXXX |

## ⇒ Virtualization Structures / Tools and Mechanisms

- In general, there are three typical classes of VM architecture. Figure 3.1 showed the architectures of a machine before and after virtualization. Before virtualization, the operating system manages the hardware. After virtualization, a virtualization layer is inserted between the hardware and the operating system.
- The virtualization layer is responsible for converting portions of the real hardware into virtual hardware. Therefore, different operating systems such as Linux and Windows can run on the same physical machine, simultaneously.
- Depending on the position of the virtualization layer, there are several classes of VM architectures, namely **the Hypervisor architecture, Para virtualization, and Host-based virtualization**. The hypervisor is also known as the VMM (Virtual Machine Monitor).

### Hypervisor and Xen Architecture

The hypervisor supports hardware-level virtualization on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor. 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 hypervisor is smaller than that of a monolithic hypervisor.

### The Xen Architecture

Xen is an open source hypervisor program developed by Cambridge University. Xen is a microkernel 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 provides a virtual environment located between the hardware and the OS. The core components of a Xen system are the hypervisor, kernel, and applications.

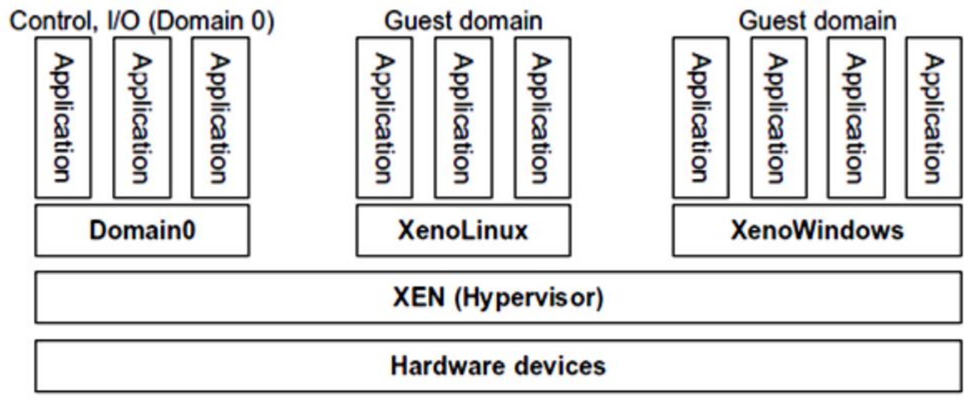


**FIGURE 3.5**

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

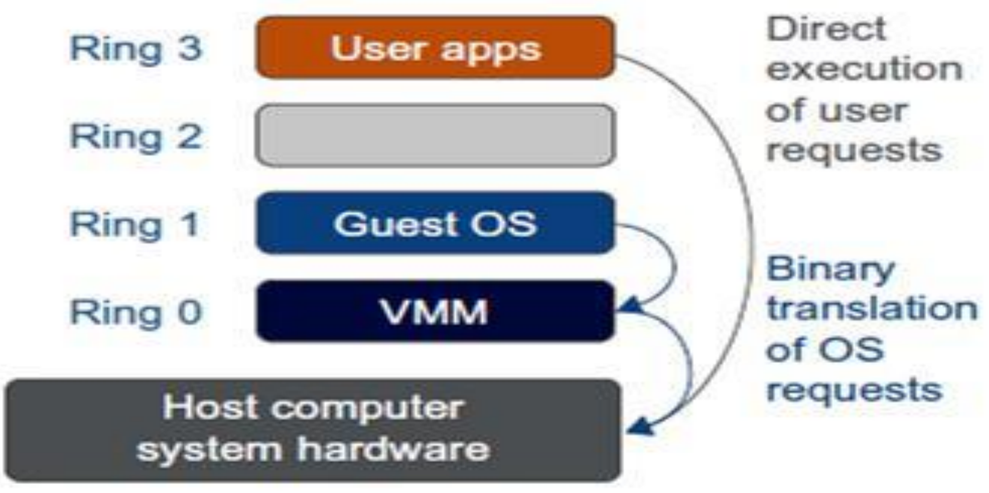
### Binary Translation with Full Virtualization

Depending on implementation technologies, hardware virtualization can be classified into two categories: 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, non virtualizable instructions. 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.

### Host-Based Virtualization

In a host-based system, both a host OS and a guest OS are used. A virtualization software layer is built between the host OS and guest OS. This host OS is still responsible for managing the hardware. The guest OSs are installed and run on top of the virtualization layer. Dedicated applications may run on the VMs.


Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

**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.

Figure 3.7 illustrates the concept of a para-virtualized VM architecture. The guest operating systems are para-virtualized. They are assisted by an intelligent compiler to replace the non virtualizable OS instructions by hyper calls as illustrated in Figure 3.8. The traditional x86 processor offers four instruction execution rings: Rings 0, 1, 2, and 3. The lower the ring number, the higher the privilege of instruction being executed. The OS is responsible for managing the hardware and the privileged instructions to execute at Ring 0, while user-level applications run at Ring 3. The best example of para-virtualization is the KVM to be described below.

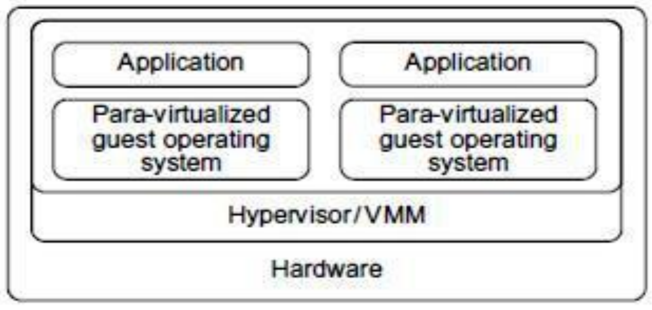

Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor/VMM
Hardware

**FIGURE 3.7**

Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization process (See Figure 3.8 for more details.)

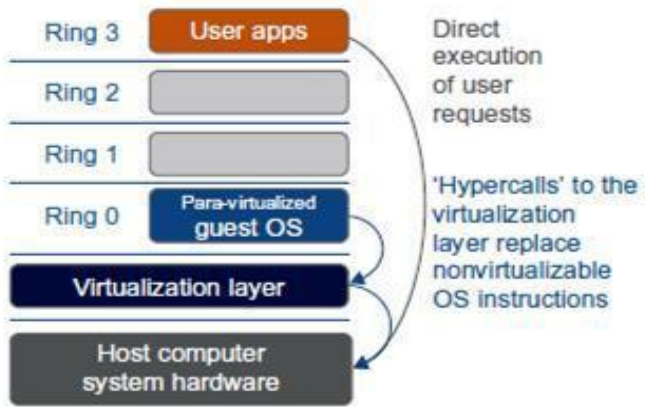

Ring 3
User apps
Ring 2
Ring 1
Ring 0
Para-virtualized guest OS
Virtualization layer
Host computer system hardware
Direct execution of user requests
'Hypercalls' to the virtualization layer replace nonvirtualizable OS instructions

**FIGURE 3.8**

The use of a para-virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizable OS instructions by hypercalls.

*(Courtesy of VMWare [71])*

**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 OSs such as Windows, Linux, Solaris, and other UNIX variants.

Unlike the full virtualization architecture which intercepts and emulates privileged and sensitive instructions at runtime, para-virtualization handles these instructions at compile time.

## ⇒ <u>Virtualization of CPU, Memory, And I/O Devices</u>

To support virtualization, processors such as the x86 employ a special running mode and instructions, known as hardware-assisted virtualization. In this way, the VMM and guest OS run in different modes and all sensitive instructions of the guest OS and its applications are trapped in the VMM.

**Hardware Support for Virtualization**

Modern operating systems and processors permit multiple processes to run simultaneously. If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash. Therefore, all processors have at least two modes, user mode and supervisor mode, to ensure controlled access of critical hardware.

Instructions running in supervisor mode are called privileged instructions. Other instructions are unprivileged instructions.
**Example:** Hardware Support for Virtualization in the Intel x86 Processor.

### CPU Virtualization

A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode. Other critical instructions should be handled carefully for correctness and stability. The critical instructions are divided into three categories: privileged instructions, control sensitive instructions, and behavior-sensitive instructions.

CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode.
**Example:** Intel Hardware-Assisted CPU Virtualization

### Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. All modern x86 CPUs include a memory management unit (MMU) and a translation look aside buffer (TLB) to optimize virtual memory performance. However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs.

That means a two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory.

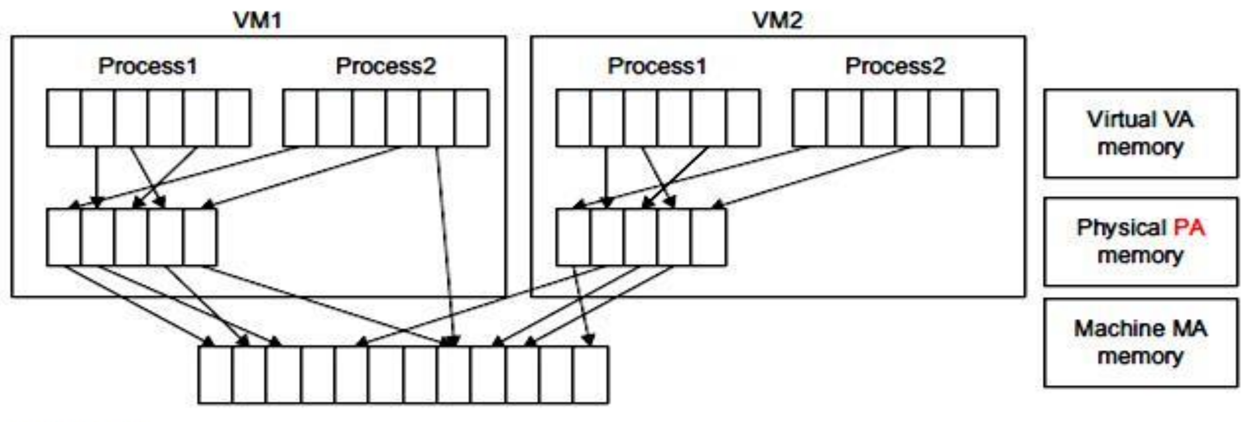


**FIGURE 3.12**
Two-level memory mapping procedure.

VMware uses shadow page tables to perform virtual-memory-to-machine-memory address translation. Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access.

### I/O Virtualization

I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware. At the time of this writing, there are three ways to implement I/O virtualization:

- **Full device emulation,**
- **Para-virtualization, and**
- **Direct I/O.**

**Full device emulation** is the first approach for I/O virtualization. The **Para-virtualization** method of I/O virtualization is typically used in Xen. It is also known as the split driver model consisting of a frontend driver and a backend driver. **Direct I/O virtualization** lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs.

### ➤ Virtualization in Multi-Core Processors

Virtualizing a multi-core processor is relatively more complicated than vitalizing a uni-core processor. Though multi core processors are claimed to have higher performance by integrating multiple processor cores in a single chip, muti-core virtualization has raised some new challenges to computer architects, compiler constructors, system designers, and application programmers.

## ⇒ <u>Virtual Clusters and Resource Management</u>

- A physical cluster is a collection of servers (physical machines) interconnected by a physical network such as a LAN.
- Here, we introduce virtual clusters and study its properties as well as explore their potential applications. In this section, we will study three critical design issues of virtual clusters: live migration of VMs, memory and file migrations, and dynamic deployment of virtual clusters.

### ➤ Physical versus Virtual Clusters

Virtual clusters are built with VMs installed at distributed servers from one or more physical clusters. The VMs in a virtual cluster are interconnected logically by a virtual network across several physical networks. Figure 3.18 illustrates the concepts of virtual clusters and physical clusters. Each virtual cluster is formed with physical machines or a VM hosted by multiple physical clusters. The virtual cluster boundaries are shown as distinct boundaries.

The provisioning of VMs to a virtual cluster is done dynamically to have the following interesting **properties:**

- The virtual cluster nodes can be either physical or virtual machines. Multiple VMs running with different OSs can be deployed on the same physical node.
- A VM runs with a guest OS, which is often different from the host OS, that manages the resources in the physical machine, where the VM is implemented.
- The purpose of using VMs is to consolidate multiple functionalities on the same server. This will greatly enhance server utilization and application flexibility.

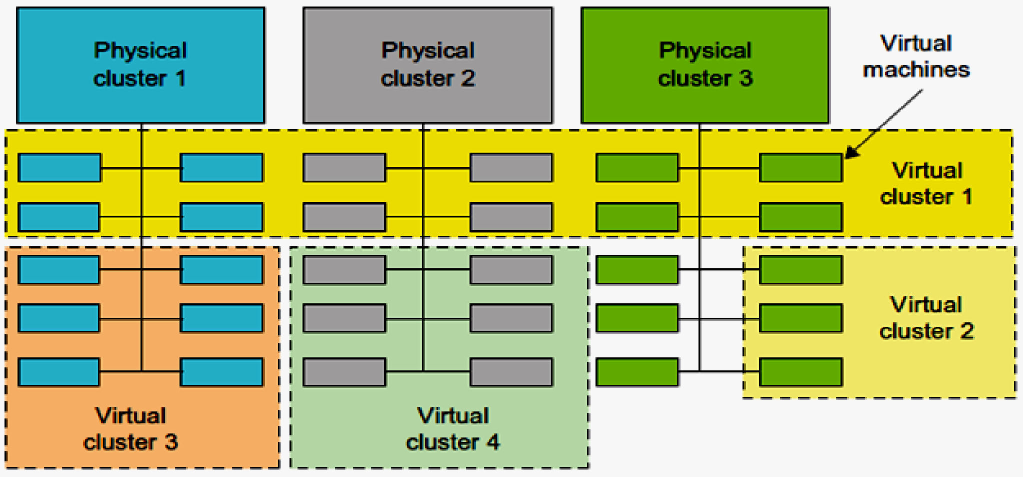


**FIGURE 3.18**

A cloud platform with four virtual clusters over three physical clusters shaded differently.

- VMs can be colonized (replicated) in multiple servers for the purpose of promoting distributed parallelism, fault tolerance, and disaster recovery.
- The size (number of nodes) of a virtual cluster can grow or shrink dynamically, similar to the way an overlay network varies in size in a peer-to-peer (P2P) network.
- The failure of any physical nodes may disable some VMs installed on the failing nodes. But the failure of VMs will not pull down the host system.

### Fast Deployment and Effective Scheduling

The system should have the capability of fast deployment. Here, deployment means two things:

- To construct and distribute software stacks (OS, libraries, applications) to a physical node inside clusters as fast as possible, and
- To quickly switch runtime environments from one user’s virtual cluster to another user’s virtual cluster.

### High-Performance Virtual Storage

Some storage architecture design can be applied to reduce duplicated blocks in a distributed file system of virtual clusters. Hash values are used to compare the contents of data blocks.

### Live VM Migration Steps and Performance Effects

Live migration means moving a VM from one physical node to another while keeping its OS environment and applications unbroken. This capability is being increasingly utilized in today’s enterprise environments to provide efficient online system maintenance, reconfiguration, load balancing, and proactive fault tolerance. It provides desirable features to satisfy requirements for computing resources in modern computing systems, including server consolidation, performance isolation, and ease of management.

Figure shows the process of live migration of a VM from host A to host B. The migration copies the VM state file from the storage area to the host machine.

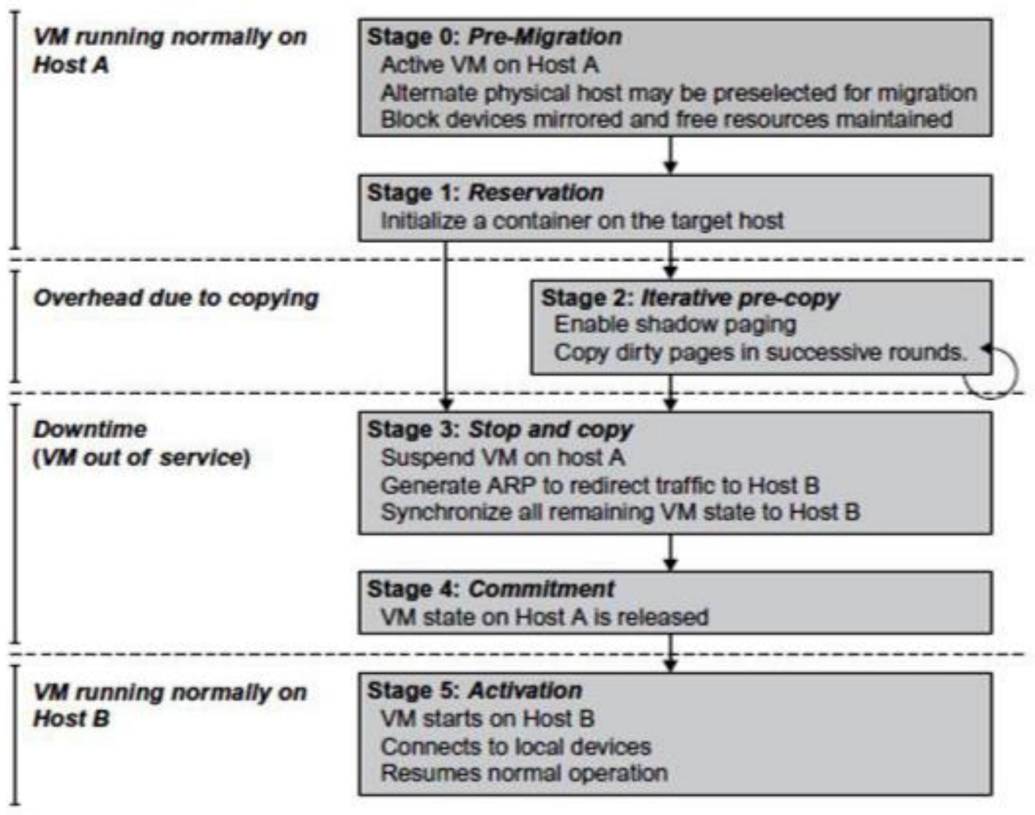

VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime
(VM out of service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
Resumes normal operation

**FIGURE 3.20**

Live migration process of a VM from one host to another.

## Memory Migration

This is one of the most important aspects of VM migration. Moving the memory instance of a VM from one physical host to another can be approached in any number of ways. Memory migration can be in a range of hundreds of megabytes to a few gigabytes in a typical system today, and it needs to be done in an efficient manner.

## File System Migration

To support VM migration, a system must provide each VM with a consistent, location-independent view of the file system that is available on all hosts. A simple way to achieve this is to provide each VM with its own virtual disk which the file system is mapped to and transport the contents of this virtual disk along with the other states of the VM.

## Network Migration

A migrating VM should maintain all open network connections without relying on forwarding mechanisms on the original host or on support from mobility or redirection mechanisms. To enable remote systems to locate and communicate with a VM, each VM must be assigned a virtual IP address known to other entities.

## Dynamic Deployment of Virtual Clusters

Table 3.5 summarizes four virtual cluster research projects. We briefly introduce them here just to identify their design objectives and reported results. The Cellular Disco at Stanford is a virtual cluster built in a shared-memory multiprocessor system. The INRIA virtual cluster was built to test parallel algorithm performance. The COD and VIOLIN clusters are studied in forthcoming examples.

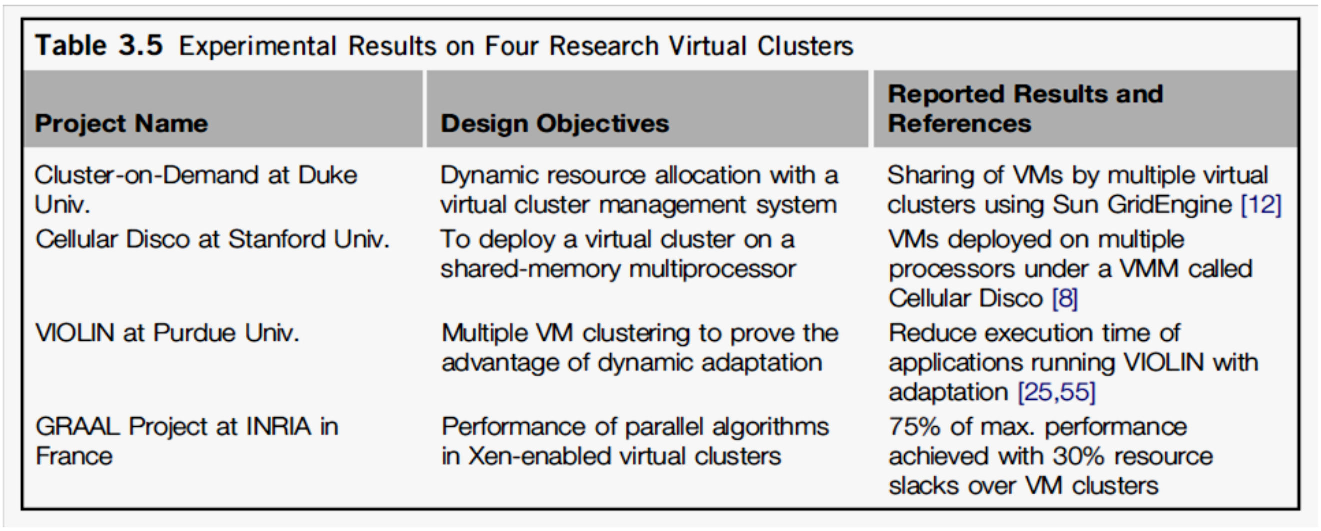


**Table 3.5** Experimental Results on Four Research Virtual Clusters

| Project Name | Design Objectives | Reported Results and References |
|---|---|---|
| Cluster-on-Demand at Duke Univ. | Dynamic resource allocation with a virtual cluster management system | Sharing of VMs by multiple virtual clusters using Sun GridEngine [12] |
| Cellular Disco at Stanford Univ. | To deploy a virtual cluster on a shared-memory multiprocessor | VMs deployed on multiple processors under a VMM called Cellular Disco [8] |
| VIOLIN at Purdue Univ. | Multiple VM clustering to prove the advantage of dynamic adaptation | Reduce execution time of applications running VIOLIN with adaptation [25,55] |
| GRAAL Project at INRIA in France | Performance of parallel algorithms in Xen-enabled virtual clusters | 75% of max. performance achieved with 30% resource slacks over VM clusters |

## ⇒ Virtualization for Data-Center Automation

- Data centers have grown rapidly in recent years, and all major IT companies are pouring their resources into building new data centers. In addition, Google, Yahoo!, Amazon, Microsoft, HP, Apple, and IBM are all in the game. All these companies have invested billions of dollars in datacenter construction and automation.
- Data-center automation means that huge volumes of hardware, software, and database resources in these data centers can be allocated dynamically to millions of Internet users simultaneously, with guaranteed QOS and cost-effectiveness.

### Server Consolidation in Data Centers

In data centers, a large number of heterogeneous workloads can run on servers at various times. These heterogeneous workloads can be roughly divided into two categories: chatty workloads and non interactive workloads. Chatty workloads may burst at some point and return to a silent state at some other point. A web video service is an example of this, whereby a lot of people use it at night and few people use it during the day. Non interactive workloads do not require people's efforts to make progress after they are submitted.

Server consolidation is an approach to improve the low utility ratio of hardware resources by reducing the number of physical\ servers. Among several server consolidation techniques such as centralized and physical consolidation, virtualization-based server consolidation is the most powerful. Data centers need to optimize their resource management.

To automate data-center operations, one must consider resource scheduling, architectural support, power management, automatic or autonomic resource management, performance of analytical models, and so on.

### Virtual Storage Management

In system virtualization, virtual storage includes the storage managed by VMMs and guest OSes. Generally, the data stored in this environment can be classified into two categories: VM images and application data. *Parallax* is a distributed storage system customized for virtualization environments, in which storage features that have traditionally been implemented directly on high-end storage arrays and switchers are relocated into a federation of storage VMs. These storage VMs share the same physical hosts as the VMs that they serve.

## Cloud OS for Virtualized Data Centers

Data centers must be virtualized to serve as cloud providers. Nimbus, Eucalyptus, and OpenNebula are all open source software available to the general public. Only vSphere4 is a proprietary OS for cloud resource virtualization and management over data centers.

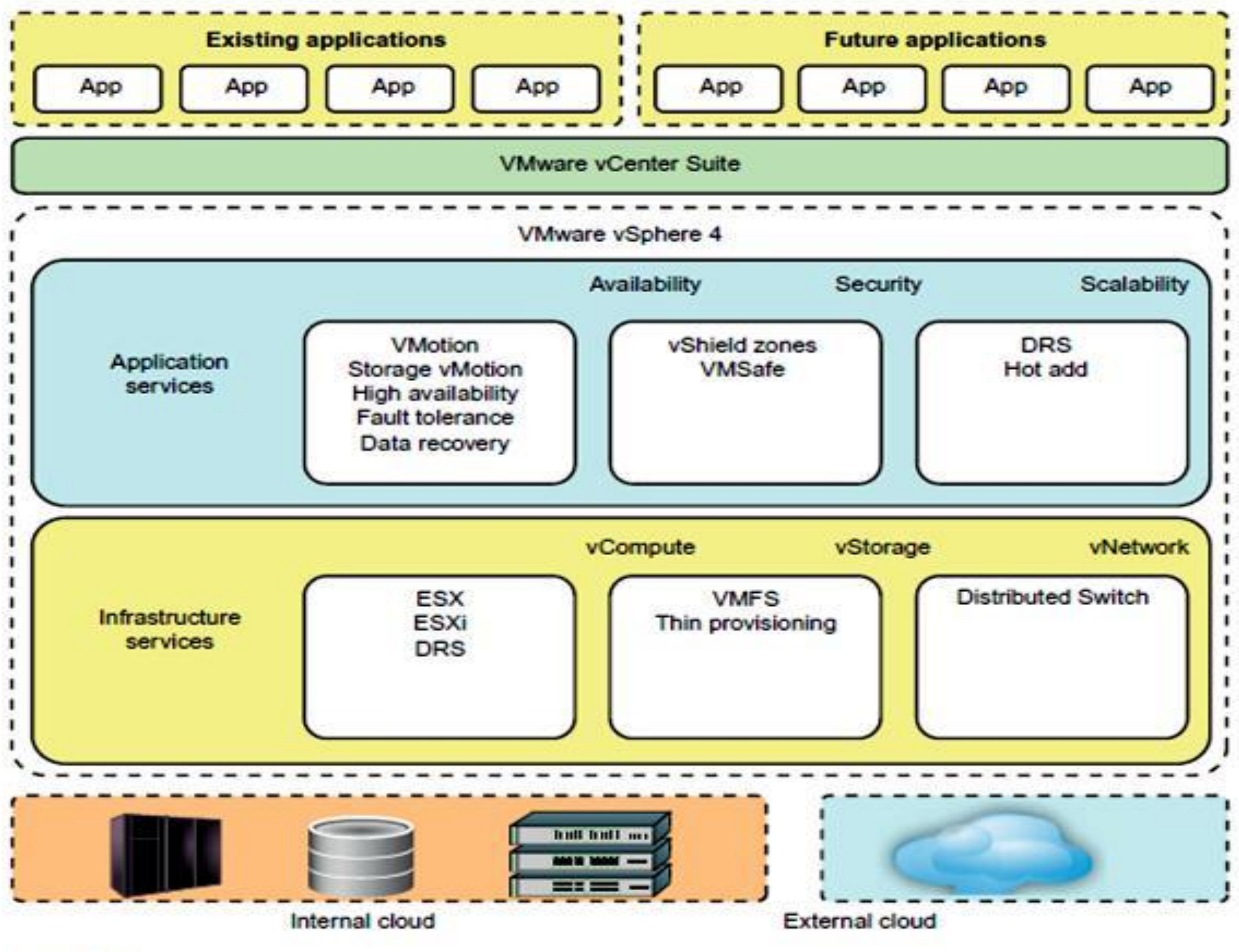


**FIGURE 3.28**

vSphere/4, a cloud operating system that manages compute, storage, and network resources over virtualized data centers.

## Trust Management in Virtualized Data Centers

A VMM can provide secure isolation and a VM accesses hardware resources through the control of the VMM, so the VMM is the base of the security of a virtual system. One VM is taken as a management VM to have some privileges such as creating, suspending, resuming, or deleting a VM. Once a hacker successfully enters the VMM or management VM, the whole system is in danger.

## VM-Based Intrusion Detection

Virtualization-based intrusion detection can isolate guest VMs on the same hardware platform. An intrusion detection system (IDS) is built on operating systems, and is based on the characteristics of intrusion actions. A typical IDS can be classified as a host-based IDS (HIDS) or a network-based IDS (NIDS), depending on the data source. A HIDS can be implemented on the monitored system. When the monitored system is attacked by hackers, the HIDS also faces the risk of being attacked. A NIDS is based on the flow of network traffic which can’t detect fake actions.