UNIT III

Virtualization: Approaches to Virtualization, Hypervisors, Virtualization to Cloud Computing, **Programming Models for Cloud Computing:** MapReduce, Cloud Haskell, Software Development in Cloud, Different Perspectives on SaaS Development, **New Programming Models Proposed for Cloud:** Orleans, BOOM and Bloom, Grid Batch, Simple API for Grid Applications.

Chapter Overview: Virtualization in Cloud Computing

Virtualization is a fundamental enabling technology for delivering various cloud computing services. It enhances **scalability**, **flexibility**, and **resource utilization** of the underlying infrastructure, while simplifying administrative tasks for IT personnel. Through **resource sharing**, virtualization also contributes to **green IT initiatives** by reducing energy consumption and optimizing hardware usage.

This chapter introduces the concept of virtualization and explains its benefits in the context of cloud computing. It discusses the different types of resources that can be virtualized, including **processor, memory, storage, and network resources**. Further, the chapter presents various approaches to virtualization, such as:

- **Full Virtualization**
- **Hardware-Assisted Virtualization**
- **Paravirtualization**

Additionally, the role of **hypervisors** (types, functions, and security concerns) is examined in detail. Toward the end, the chapter highlights the key differences between **cloud computing and virtualization**, and explains how virtualization serves as the backbone for cloud computing to deliver on-demand services effectively.

Virtualization -Introduction

1. Need for Virtualization

Modern computing environments require **large-scale infrastructure** to handle complex workloads. Traditionally, organizations purchased **new physical hardware** whenever extra resources were needed. However, this approach created problems:

- **High CapEx (Capital Expenditure):** Big upfront investments for servers, storage, and networking equipment.
- **High OpEx (Operational Expenditure):** Ongoing costs for power, cooling, maintenance, and management.
- **Low Resource Utilization:** A physical server might only use **10–20% of its capacity**, leaving most of the hardware idle.
- **Poor ROI (Return on Investment):** Huge spending but little benefit from underutilized systems.

As a result, organizations needed a solution that would **increase utilization, reduce cost, and improve flexibility**. This led to the widespread adoption of **virtualization**.

2. What is Virtualization?

Virtualization is a **technology that enables a single physical infrastructure to act like multiple independent logical systems**.

- It allows multiple **operating systems (OSs)** and applications to run on a **single physical machine**.
- A **software layer called Hypervisor (Virtual Machine Monitor)** manages the sharing of hardware resources among these virtual environments.
- Each virtual environment (called a **Virtual Machine – VM**) behaves like a real physical computer, with its own OS, applications, and storage.

👉 **Definition:** Virtualization is the process of creating a virtual version of computing resources (like servers, storage, memory, or networks) to enable resource sharing and better utilization.

3. Benefits of Virtualization

Virtualization provides several advantages:

1. **Efficient Resource Utilization** – Hardware resources are shared and used effectively.
2. **Cost Savings** – Reduces CapEx and OpEx by consolidating multiple servers into fewer physical machines.
3. **Increased ROI** – Better use of purchased infrastructure improves return on investment.
4. **Flexibility & Scalability** – Virtual machines can be created, modified, or deleted easily.
5. **Ease of Administration** – IT staff can manage multiple virtual systems from a central console.
6. **Green IT Support** – Fewer physical servers = reduced energy consumption, less space, lower carbon footprint.
7. **Improved Disaster Recovery** – Virtual machines can be backed up, cloned, or moved across systems quickly.

4. Types/Forms of Virtualization

Virtualization is not limited to hardware. Different forms include:

1. **Processor Virtualization** – Running multiple virtual CPUs on one physical CPU.
2. **Memory Virtualization** – Combining multiple memory resources into one pool, or allocating memory to VMs as needed.
3. **Storage Virtualization** – Abstracting physical storage (disks) into logical volumes for flexible usage.
4. **Network Virtualization** – Creating virtual networks (VLANs, VPNs) independent of physical hardware.
5. **Operating System Virtualization** – Running multiple isolated OS environments on the same hardware.

6. **Data Virtualization** – Allowing applications to access and process data without worrying about its physical location.
7. **Application Virtualization** – Running applications in virtualized containers independent of the underlying OS.

5. Before vs After Virtualization

Aspect	**Before Virtualization**	**After Virtualization**
Hardware Usage	One server → One OS + applications	One server → Multiple OSs + applications
Resource Utilization	Low (10–20%)	High (70–90%)
Costs (CapEx/OpEx)	Very High (new hardware for every need)	Lower (shared resources, fewer servers)
Scalability	Slow – need to buy and install new hardware	Fast – create new virtual machines easily
Experimentation/Testing	Requires separate hardware for each test	Multiple test environments on the same machine
Green IT	More power, space, cooling required	Less power and space, energy-efficient

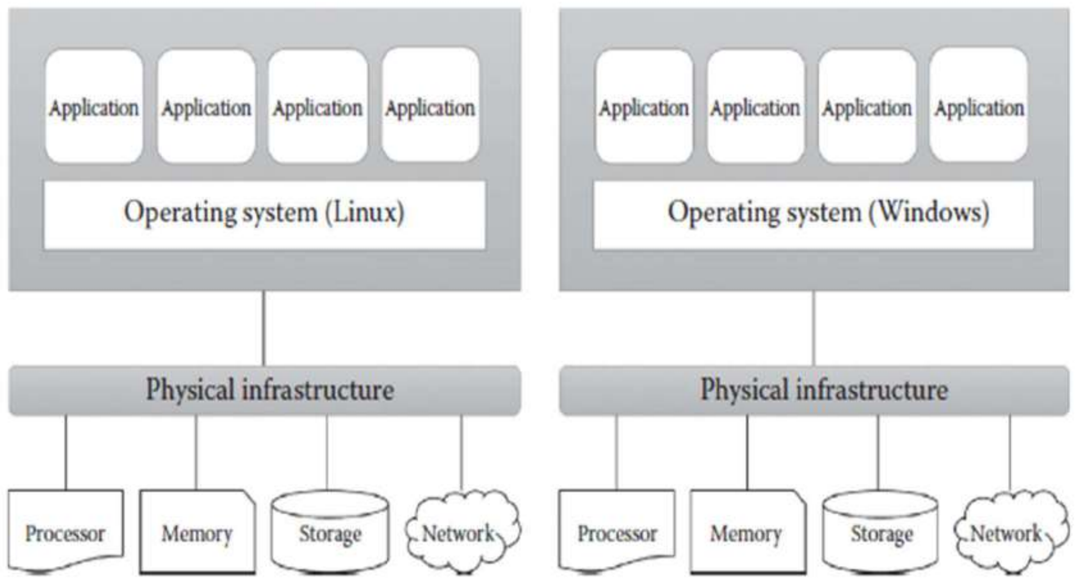

FIGURE 7.1
Before virtualization.

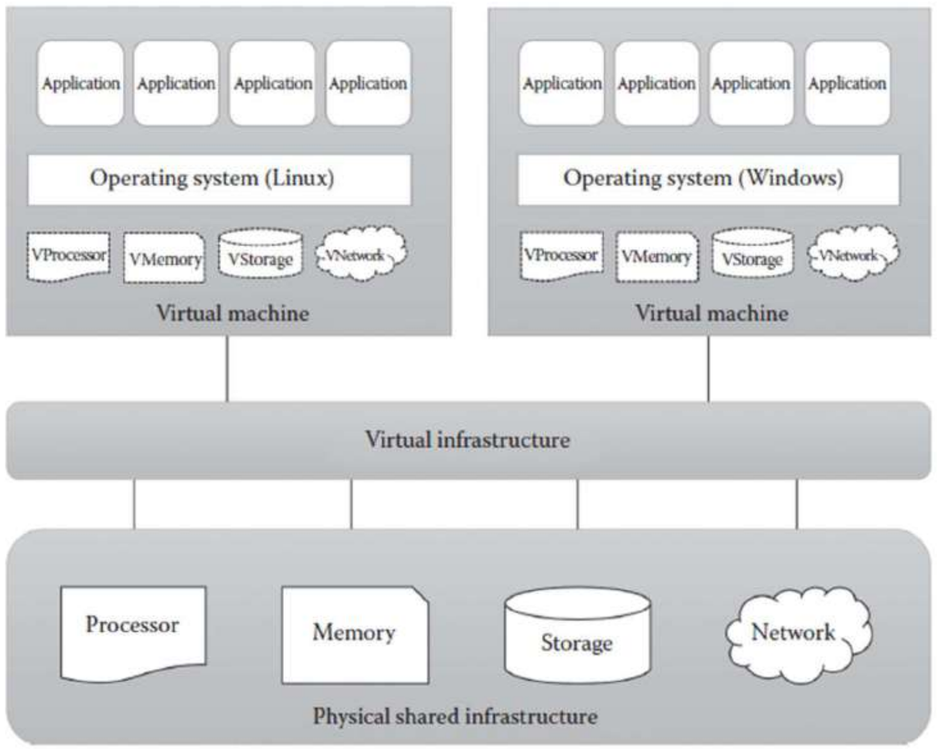

FIGURE 7.2
After virtualization.

6. Virtualization and Cloud Computing

- **Virtualization ≠ Cloud Computing**
  - Virtualization is a **technology** (creates virtual resources).
  - Cloud computing is a **service model** (delivers IT resources on demand using virtualization).
- **Relationship:**
  - Virtualization is the **foundation of cloud computing**.
  - Without virtualization, cloud computing cannot provide **scalability, resource pooling, and multi-tenancy**.
  - Example: In a public cloud (like AWS, Azure, GCP), virtualization enables multiple users to share the same physical data center infrastructure securely.

2 Virtualization Opportunities

Virtualization is the process of **abstracting physical resources into virtual resources** that can be allocated to **Virtual Machines (VMs)**.

The major resources that can be virtualized include **processor, memory, storage, network, data, and applications**.

Below are the different virtualization opportunities:

1 Processor Virtualization

- Allows VMs to share virtual processors that are created from the **physical processors** of the system.
- The **virtualization layer (hypervisor)** abstracts physical CPUs into **virtual CPUs (vCPUs)**.

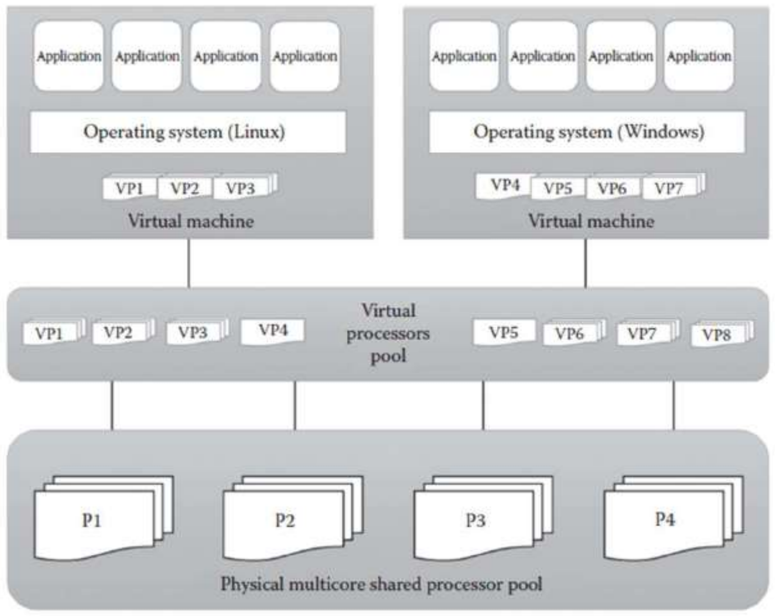

FIGURE 7.3
Processor virtualization.

- **How it works:**
  - A hypervisor manages the scheduling of physical processor time across multiple VMs.
  - Each VM "thinks" it has its own processor.
- **Where used:**
  - Data centers and cloud environments.
  - Can be implemented from a **single server** or even from **distributed servers** (resource pooling).

2 Memory Virtualization

- Provides VMs with **virtual main memory** mapped from the underlying **physical memory**.
- Similar to the concept of **virtual memory** in OS.

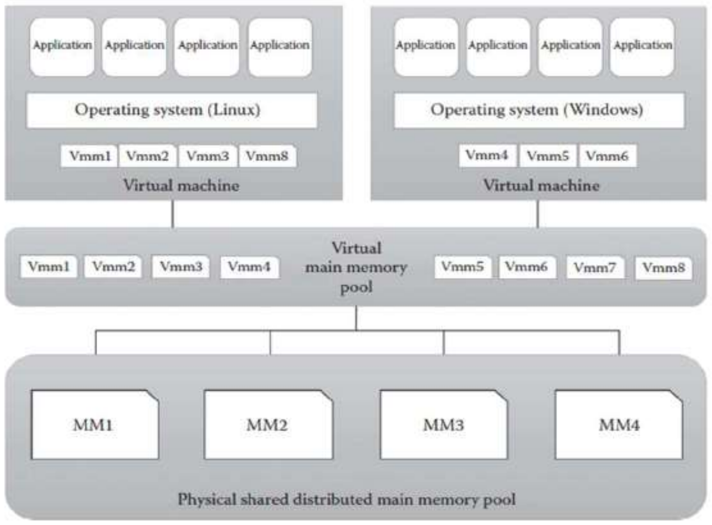

FIGURE 7.4
Main memory virtualization.

- **How it works:**
  - Maps **virtual page numbers → physical page numbers**.
  - Supported by modern x86 processors.
- **Advanced usage:**
  - Hypervisors can consolidate unused memory from different servers into a **virtual memory pool**.
  - This pool is dynamically allocated to VMs when required.
- **Benefit:**
  - Increases flexibility and ensures no memory remains idle.

3 Storage Virtualization

- Combines multiple **physical storage devices** into a pool of **logical storage (virtual disks)**.
- Each VM sees only the logical storage, not the underlying hardware.

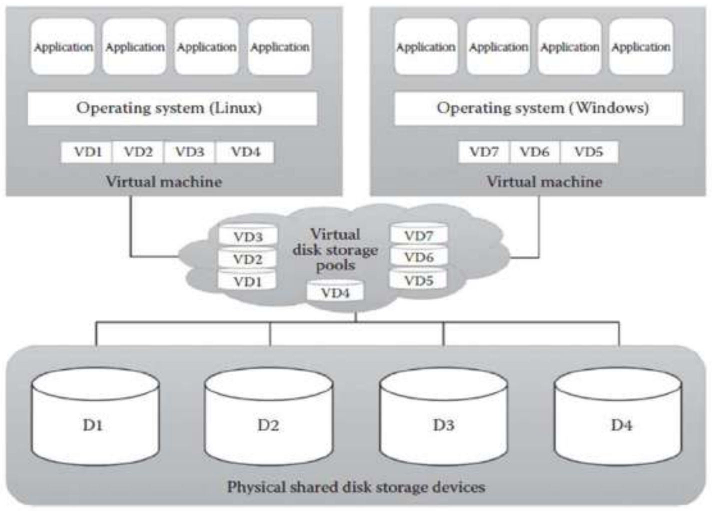

FIGURE 7.5
Storage virtualization.

- **Uses:**
  - Efficient utilization of physical storage.
  - Data **backup and replication**.
  - Ensures **high availability** of data.
- **Technologies used:**
  - Hypervisors
  - **Storage Area Networks (SANs)**
  - **Network-Attached Storage (NAS)**

4 Network Virtualization

- Creates a **virtual network** by abstracting physical networking components (routers, switches, NICs).
- Managed by virtualization software.

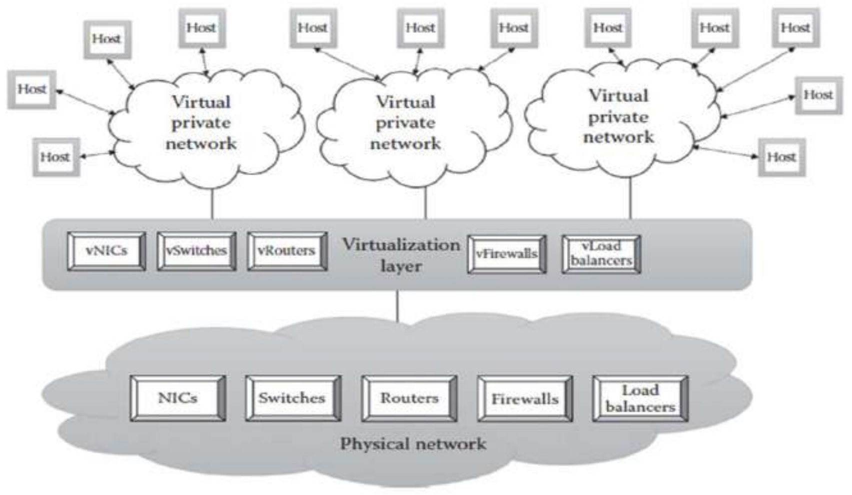

FIGURE 7.6
Network virtualization.

- **Features:**
  - Virtual networks act as a **software-based entity** that includes both network hardware and software.
  - Enables **communication between VMs** on the same physical network.
- **Types of VM Network Access:**

1. **Bridged Network** – VM directly connected to physical network.
2. **NAT (Network Address Translation)** – VM shares the host's IP address.
3. **Host-Only Network** – VM communicates only with the host system.

5 Data Virtualization

- Provides access to data **without concern for type or physical location**.
- Aggregates heterogeneous data from different sources into a **single logical data view**.

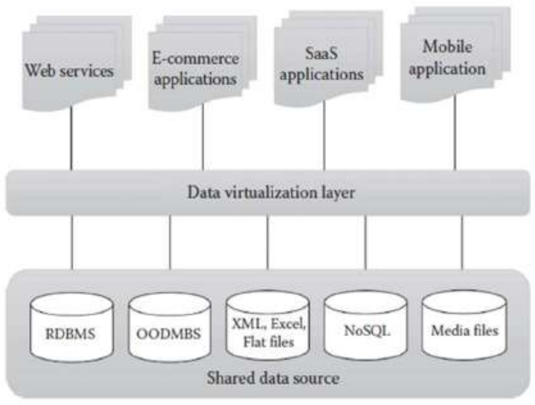

FIGURE 7.7
Data virtualization.

- **Features:**
  - Hides complexity of data type and location from applications.
  - Ensures **single point of access** to distributed data.
- **Uses:**
  - **Data integration**
  - **Business Intelligence (BI)**
  - **Cloud computing** (data available to SaaS, mobile apps, portals, etc.)

6 Application Virtualization

  - Allows users to run applications **without installing them locally**.
  - Applications are **developed, hosted, and virtualized on a central server**.

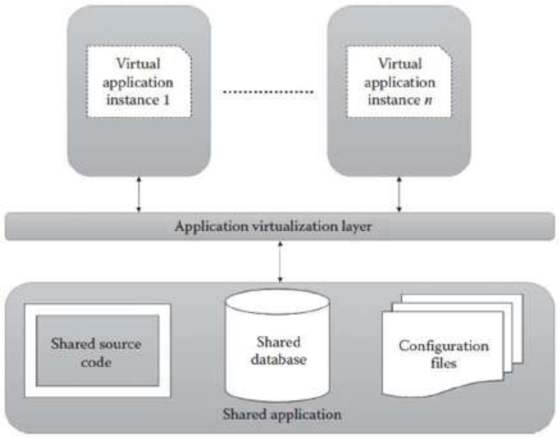
Virtual application instance 1
Virtual application instance n
Application virtualization layer
Shared source code
Shared database
Configuration files
Shared application

FIGURE 7.8
Application virtualization.

- **How it works:**
  - Users are provided with an **isolated virtual copy** of the application.
  - Simplifies deployment (no need for client-side installations).
- **Relation to Cloud:**
  - Key enabling technology for **Software as a Service (SaaS)**.
- **Examples:**
  - Microsoft Office 365, Google Workspace, Virtual Desktop applications.

Approaches to Virtualization

Before understanding the approaches, we need to know about **protection rings in Operating Systems (OSs).**

- **Protection Rings in OSs**
  - Protection rings define **privilege levels** to separate trusted OS code from untrusted applications.
  - **Ring 0** → Most privileged (Kernel of OS, full hardware access).
  - **Ring 1 & Ring 2** → Device drivers.
  - **Ring 3** → Least privileged (user applications).
  - Purpose: Prevent malicious or faulty applications (at Ring 3) from directly accessing physical resources.

👉 In virtualization, depending on the approach, the **hypervisor (VMM)** and **guest OS** are placed at different rings.

1 Full Virtualization

- **Definition:** Guest OS is **completely abstracted** from hardware.
  - OS is **unaware** it is running in a virtualized environment.
- **How it works:**
  - Hypervisor (VMM) runs at **Ring 0** with highest privilege.
  - Guest OS runs at a less privileged level (Ring 1).
  - Guest OS communicates with hardware **through the hypervisor only**.
  - Uses:
    - **Binary translation** → Translates non-virtualizable instructions into safe instructions.
    - **Direct execution** → User apps at Ring 3 can execute directly on hardware for efficiency.

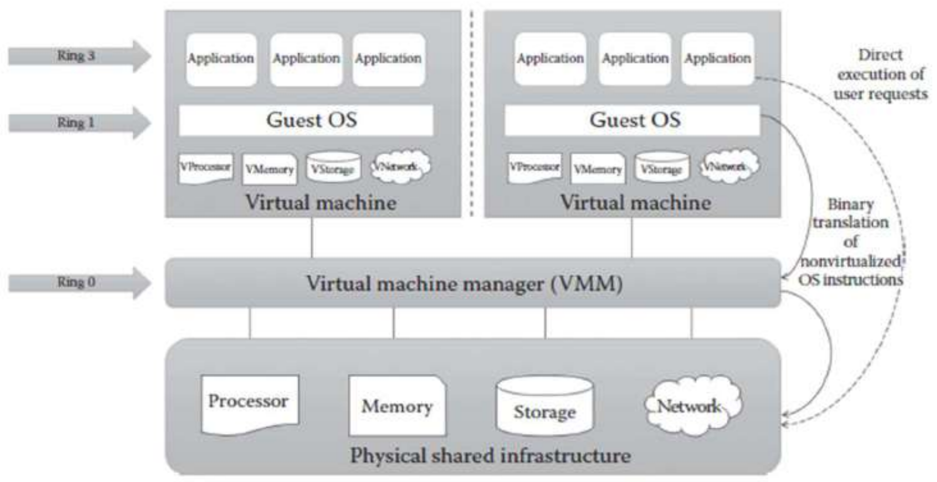

FIGURE 7.10
Full virtualization.

✅ **Pros:**

- Strong isolation and security between VMs.
- Multiple OSs can run simultaneously.
- Guest OS can be migrated easily to native hardware.
- Easy to use (no modification of guest OS).

❌ **Cons:**

- Binary translation adds overhead, reducing performance.
- Requires proper hardware-software combinations.

2 Paravirtualization

- **Definition:** Also called **partial virtualization** or **OS-assisted virtualization**.
    - Guest OS **knows** it is running in a virtual environment.
- **How it works:**
    - Guest OS is **modified** to replace non-virtualizable instructions with **hypercalls**.
    - Hypercalls = special calls (like system calls) → direct communication between guest OS and hypervisor.
    - Guest OS and hypervisor both run at privileged level (**Ring 0**).
    - User applications run at **Ring 3**.

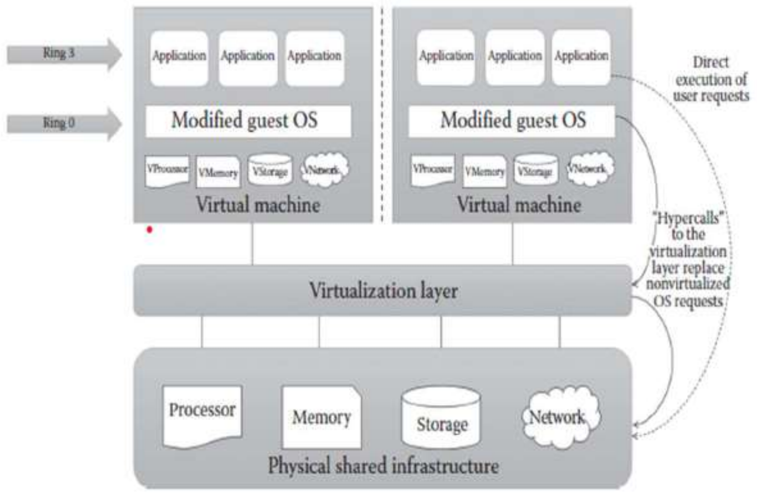

FIGURE 7.11
Paravirtualization.

✅ **Pros:**

- No binary translation → Higher efficiency and performance.
- Easier to implement compared to full virtualization.

❌ **Cons:**

- Requires **guest OS modification** → not always possible.
- Modified OS cannot run on physical hardware directly.
- Lack of backward compatibility → harder migration across different hosts.

3 Hardware-Assisted Virtualization

- **Definition:** Hardware vendors (Intel, AMD) provide built-in support for virtualization in CPUs.
    - Eliminates overhead of **binary translation** (Full Virtualization) and **guest OS modification** (Paravirtualization).
- **Examples of Technologies:**
    - Intel → **VT-x** (Virtualization Technology)
    - AMD → **AMD-V** (AMD Virtualization)
- **How it works:**
    - Hardware provides special instructions for virtualization.
    - VMM runs with **root privilege** (even below Ring 0).
    - Guest OS runs at Ring 0 (non-root privilege).
    - User applications run at Ring 3.
    - OS requests trap directly into the hypervisor (no translation needed).

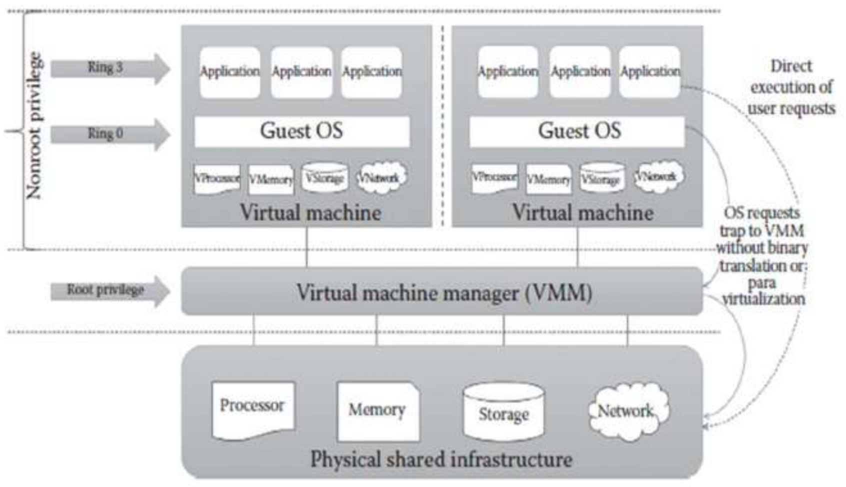

FIGURE 7.12
Hardware-assisted virtualization.

✅ **Pros:**

- Removes inefficiencies of other approaches.
- Guest OS does not require modification.
- Higher performance and better compatibility.

❌ **Cons:**

- Dependent on specific hardware (Intel VT-x, AMD-V).
- Slight hardware overhead compared to native execution.

📊 Comparison of Virtualization Approaches

Feature	**Full Virtualization**	**Paravirtualization**	**Hardware-Assisted Virtualization**
Guest OS Awareness	Not aware	Aware (modified)	Not aware
Guest OS Modification	Not required	Required	Not required
Techniques Used	Binary translation + direct execution	Hypercalls	Hardware support (VT-x, AMD-V)
Performance	Moderate (overhead due to translation)	High (no translation)	High (direct trap to hypervisor)
Compatibility	High	Low (needs modified OS)	High
Security	Strong	Strong	Strong

4 Hypervisors

◆ Introduction

- In modern IT, **Virtual Machines (VMs)** are widely used instead of physical machines.
- VMs improve **resource utilization**, simplify management tasks, and support **green IT** by reducing hardware and energy needs.
- The **hypervisor** (also called **Virtual Machine Monitor – VMM**) is the key software tool that enables virtualization by creating and managing virtual environments.
- Hypervisors provide **virtual infrastructure** such as:
  - Virtual CPUs (vCPUs)
  - Virtual Memory
  - Virtual NICs (vNICs)
  - Virtual Storage
  - Virtual I/O devices
- Examples: **VMware, Xen, Hyper-V, KVM, OpenVZ**

1 Types of Hypervisors

1. Type 1 Hypervisor (Bare Metal / Native Hypervisor)

- Runs **directly on physical hardware** without a host OS.
- Provides VMs with direct access to hardware resources → reduces overhead.

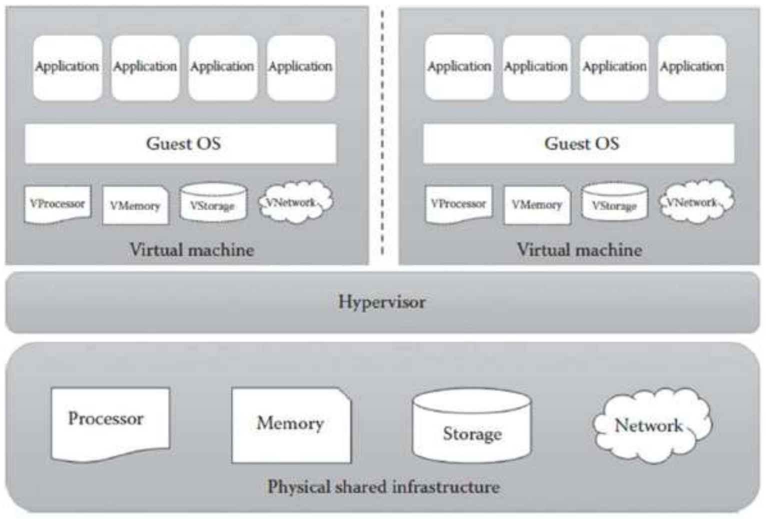

FIGURE 7.13
Type 1 or bare metal hypervisor.

- **Advantages:**
  - Better **performance and efficiency**.
  - More **secure** (harder for attackers to exploit).
- **Best for:**
  - Enterprise servers, heavy workloads, data centers.
- **Examples:**
  - Microsoft Hyper-V
  - Citrix XenServer
  - VMware ESXi
  - Oracle VM Server for SPARC

2. Type 2 Hypervisor (Hosted Hypervisor)

- Runs **on top of a host OS** like any other software application.
- VMs access hardware **through the host OS**.

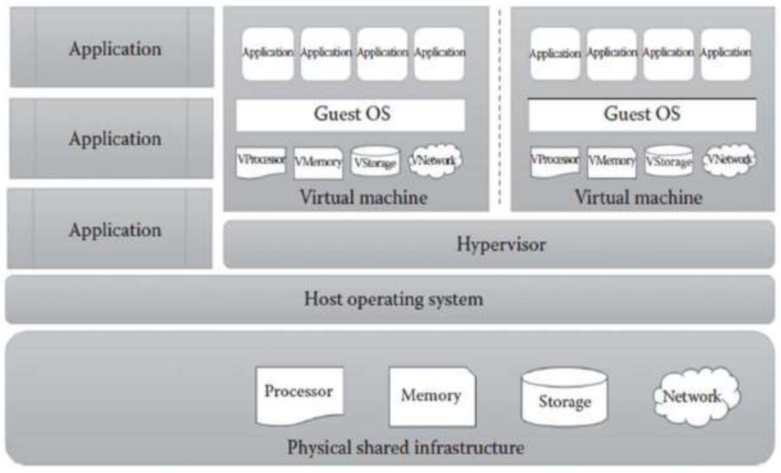

FIGURE 7.14
Type 2 or hosted hypervisor.

- **Disadvantages:**
    - Extra overhead → lower efficiency.
    - If the **host OS crashes**, all VMs also fail.
- **Best for:**
    - **Client systems** where performance is less critical (testing, development).
- **Examples:**
    - VMware Workstation
    - Oracle VirtualBox

---

2 Security Issues in Hypervisors

Since hypervisors manage multiple VMs and have **direct access to hardware**, they are a **prime target for attackers**.

Main attack vectors:

1. **Attack through Host OS (Type 2 hypervisors):**
    - Exploit vulnerabilities in host OS.
    - If host OS is compromised → attacker gains control of hypervisor.

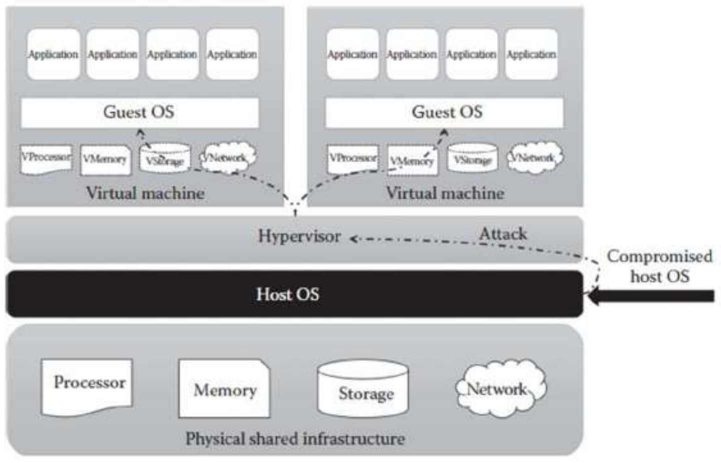

FIGURE 7.15
Attack through the host OS.

    - **Possible attacks:**
        - **Denial of Service (DoS):** Deny virtual resources to new VMs.
        - **Data theft:** Steal confidential VM data.

2. **Attack through Guest OS (Type 1 & Type 2):**
    - Malicious guest OS or VM sends harmful requests to hypervisor.
    - **Possible attacks:**
        - Unauthorized access to other VMs.
        - Abuse hardware resources (resource exhaustion).

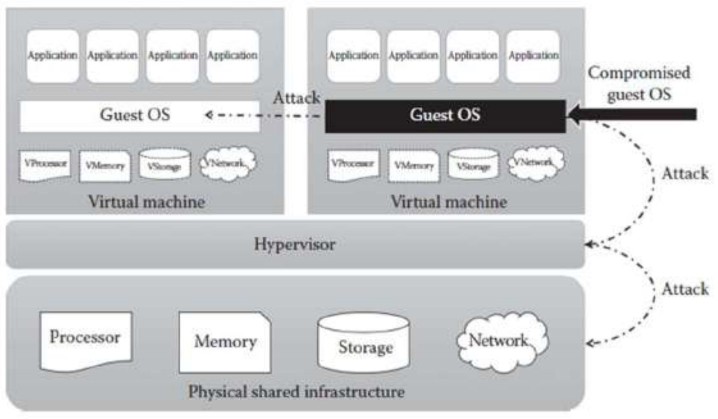

FIGURE 7.16
Attack through the guest OS.

Recommendations for Securing Hypervisors

To protect hypervisors from attacks:

- **Regular updates:** Keep hypervisor and host OS patched.
- **Disable unused hardware resources** to reduce attack surface.
- **Least privilege principle:** Restrict hypervisor and guest OS privileges.
- **Monitoring tools:** Deploy intrusion detection/prevention in hypervisor.
- **Strong guest isolation:** Prevent compromised VM from affecting others.
- **Mandatory access control (MAC):** Enforce strict policies on access rights.

From Virtualization to Cloud Computing

Many people mistakenly think **virtualization** and **cloud computing** are the same.
➡️ **Reality:** They are **different** technologies, but **virtualization is an enabler of cloud computing**.

We can compare them on several parameters:

1. Type of Service

- **Virtualization:**
  - Primarily provides **infrastructure-level services** (servers, storage, networking).
  - Focused on resource utilization within data centers.
- **Cloud Computing:**
  - Provides **Infrastructure (IaaS), Platform (PaaS), and Software (SaaS)** services.
  - Covers complete IT service delivery.

2. Service Delivery

- **Virtualization:**
  - Not designed for **on-demand** service delivery.
  - Services are provisioned and managed manually.

- **Cloud Computing:**
  - Services are **on-demand** and accessed anytime by end users.
  - Example: AWS, Azure, GCP provide resources instantly when requested.

3. Service Provisioning

- **Virtualization:**
  - Requires **manual intervention** by IT administrators to allocate resources.
- **Cloud Computing:**
  - Supports **automated, self-service provisioning** for end users.
  - Example: A developer can launch a VM or database instance from a portal without admin help.

4. Service Orchestration

- **Virtualization:**
  - Cannot orchestrate or compose multiple services automatically.
- **Cloud Computing:**
  - Supports **service orchestration and composition**.
  - Providers even offer automated orchestration tools (e.g., Kubernetes, AWS CloudFormation).

5. Elasticity

- **Virtualization:**
  - Limited flexibility.
  - Starting/stopping VMs is **manual** and scaling is **difficult**.
- **Cloud Computing:**
  - Offers **elasticity and auto-scaling**.
  - Resources can be **added/removed dynamically** as per workload.

6. Target Audience

- **Virtualization:**
  - Mainly for **service providers or IT owners** (to improve hardware utilization and reduce costs).
- **Cloud Computing:**
  - Targets both **service providers** (higher ROI) and **end users** (pay-per-use, no infrastructure ownership).

Cloud computing delivers IT resources as a service using **virtualization** as the enabling technology. The three major service models are:

1. **Infrastructure as a Service (IaaS)**
2. **Platform as a Service (PaaS)**
3. **Software as a Service (SaaS)**

1 Infrastructure as a Service (IaaS)

- IaaS provides **virtualized infrastructure resources** (compute, storage, network) to customers on a **pay-per-use** basis.

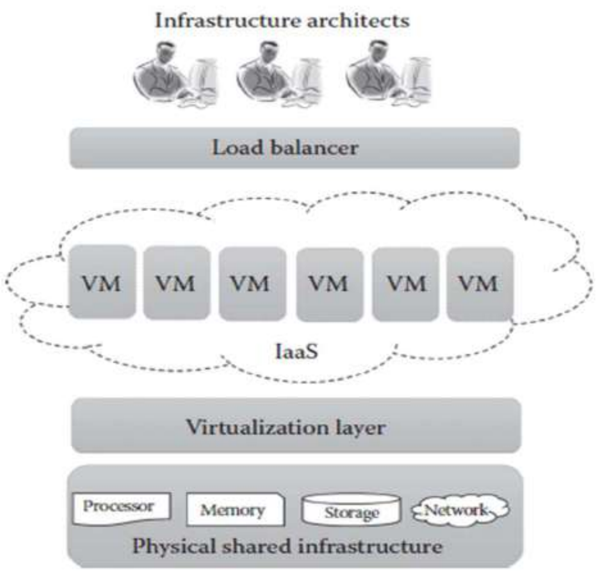

FIGURE 7.17
IaaS.

- **How it works:**
  - Uses **processor, memory, storage, and network virtualization**.
  - Hypervisors (mainly **Type 1**) abstract the physical resources into virtual resources.
  - Delivered in the form of **Virtual Machines (VMs)**.
- **Features:**
  - Customers can create, configure, and manage VMs.
  - Provides **virtual CPUs, memory, storage, and networks**.
  - Load balancers distribute traffic across multiple servers for scalability.
- **Examples of IaaS providers:**
  - Amazon Web Services (EC2)
  - Microsoft Azure IaaS
  - OpenStack
  - CloudStack
  - Eucalyptus

✅ **Use case:** Hosting servers, running enterprise applications, test and development environments.

2 Platform as a Service (PaaS)

- PaaS provides a **virtual development platform** that enables users to **develop, test, and deploy applications** without worrying about the underlying infrastructure.

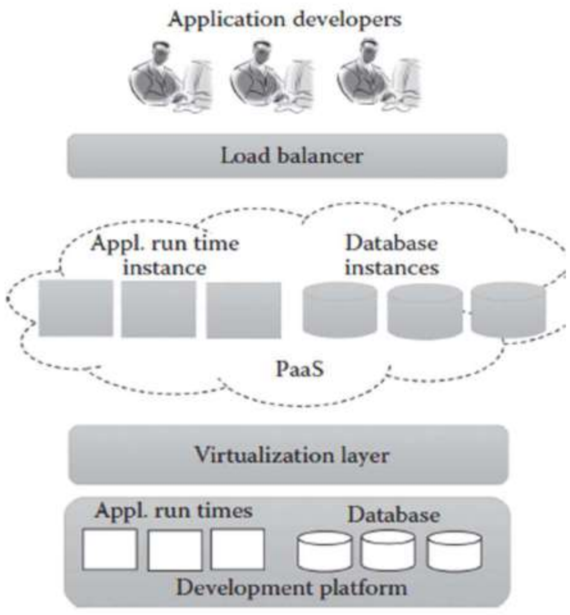

FIGURE 7.18
PaaS.

- **How it works:**
    - Service providers offer **programming languages, runtime environments, databases, middleware, and libraries** virtually.
    - Developers can build applications using online platforms accessed via **Web UIs, REST APIs, or WebCLI**.
    - Supports both **online and offline development** (integration with IDEs like Eclipse).
- **Features:**
    - Developers don't need to install software tools locally.
    - Application deployment depends on **cloud deployment model**:
        - **Public cloud** → off-premise hosting.
        - **Private cloud** → on-premise hosting.
    - Uses **OS-level and software-level virtualization**.
    - Supports **scalability** via load balancing.
- **Examples of PaaS providers:**
    - Google App Engine
    - Microsoft Azure App Services
    - RedHat OpenShift
    - Force.com (Salesforce platform)

✅ **Use case:** Application development and deployment (web apps, APIs, business apps).

3 Software as a Service (SaaS)

- SaaS provides **software applications** over the internet, hosted in the provider's data center.

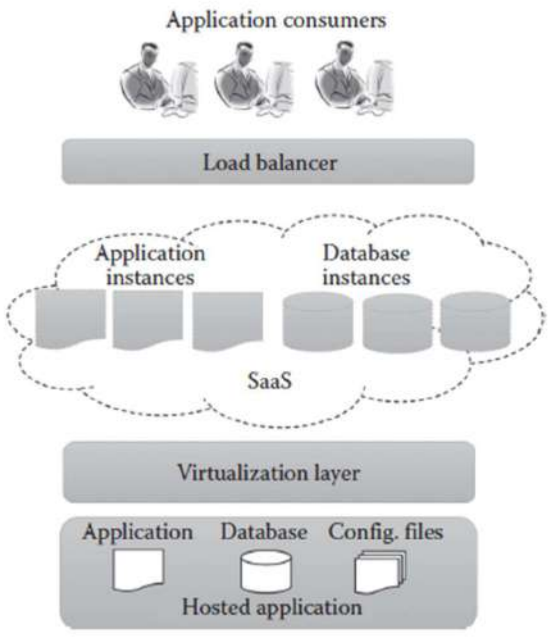

FIGURE 7.19
SaaS.

- **How it works:**
  - Applications are accessed through a **web browser**; no need to install locally.
  - Uses **application-level virtualization** to deliver apps.
  - Supports **multitenancy** → multiple users share the same software instance securely.
- **Features:**
  - Subscription-based pricing (not licensed software).
  - Highly **scalable** using software load balancers that replicate application/database servers.
  - Supports multiple application and database instances for high availability.
- **Examples of SaaS applications:**
  - Google Docs, Google Drive
  - Microsoft Office 365
  - Salesforce CRM

✅ **Use case:** Email, office suites, collaboration tools, customer relationship management (CRM).

⚙ Beyond IaaS, PaaS, SaaS

Other cloud services enabled by **virtualization**:

- **Network as a Service (NaaS)** → using network virtualization.
- **Storage as a Service (STaaS)** → using storage virtualization.
- **Database as a Service (DBaaS)** → using database virtualization.

📊 **Comparison of Cloud Service Models**

Feature	**IaaS**	**PaaS**	**SaaS**
What it provides	Virtual infrastructure (VMs, storage, network)	Virtual development & deployment platform	Ready-to-use applications
User controls	OS, middleware, runtime, apps	Apps only (platform managed)	None (just use the app)
Virtualization used	Processor, memory, storage, network	OS-level, language runtime, database	Application-level
Target users	System admins, IT managers	Developers	End users
Examples	AWS EC2, Azure IaaS, OpenStack	Google App Engine, Azure PaaS, OpenShift	Google Docs, MS Office 365, Salesforce