Virtualization: Introduction, Characteristics of Virtualized, Environments Taxonomy of Virtualization Techniques, Execution Virtualization, Other Types of Virtualization, Virtualization and Cloud Computing, Pros and Cons of Virtualization, Technology Examples

Module-2					
Q. 03	a	Explain the characteristics of virtualized environments.	L1,L2,L3	2, 3	7
	b	Give the taxonomy of virtualization techniques.	L1,L2,L3	2, 3	7
	c	What is virtualization and what are its benefits.	L1,L2,L3	2, 3	6
OR					
Q.04	a	Explain virtualization and cloud computing and pros and cons of virtualization.	L1,L2,L3	2, 3	7
	b	Explain hypervisors and its types.	L1,L2,L3	2, 3	7
	c	Discuss machine reference model of execution virtualization.	L1,L2,L3	2, 3	6

1. What is virtualization and what are its benefits?

2. What are the characteristics of virtualized environments?

3. Discuss classification or taxonomy of virtualization at different levels.

4. Discuss the machine reference model of execution virtualization.

5. What are hardware virtualization techniques?

6. List and discuss different types of virtualization.

7. What are the benefits of virtualization in the context of cloud computing?

8. What are the disadvantages of virtualization?

9. Discuss the reference model of full virtualization.

10. Discuss the architecture of Hyper-V. Discuss its use in cloud computing

3.1 Introduction to Virtualization

Virtualization is a technology that creates an abstract, or "virtual," version of something, such as hardware or an operating system. It allows multiple systems or applications to run on a single physical machine by emulating different environments. This is key for cloud computing, especially for offering **Infrastructure-as-a-Service (IaaS)**, where virtual machines (VMs) are used to provide flexible computing power.

Why is Virtualization Becoming Popular?

1. **More Powerful Computers**:

- Modern computers, even regular desktop PCs, are much more powerful than they used to be. Often, they have more capacity than needed for daily tasks. This extra power can be used to run virtual machines without slowing down the main system.

2. **Better Use of Resources**:
   - Many computers, especially in offices, are underutilized. They may only be used during work hours and sit idle at night. Virtualization allows these systems to be used more efficiently, even when no one is physically working on them.
3. **Space Limitations**:
   - Building data centers (where companies store their servers) is expensive and requires a lot of space. Big companies like Google and Microsoft can afford huge data centers, but smaller companies can't. Virtualization helps by allowing several virtual servers to run on fewer physical machines, saving space and costs. This is known as **server consolidation**.
4. **Energy Efficiency**:
   - Data centers consume a lot of electricity, not just to run servers but also to keep them cool. By reducing the number of physical servers needed, virtualization helps lower energy use, making it more eco-friendly.
5. **Reduced Management Costs**:
   - Managing large numbers of physical servers can be expensive and time-consuming. Virtualization reduces the need for as many physical servers, cutting down on tasks like system monitoring, updates, and hardware repairs, which lowers administrative costs.

History of Virtualization

- In the mid-1990s, **Java** introduced the concept of running small programs (called applets) in a virtual environment. This was a game-changer for developers and paved the way for larger, enterprise-level applications.
- In 2002, **Microsoft** launched the **.NET Framework**, which allowed developers to use multiple programming languages in a virtualized environment, further expanding the use of virtualization in business applications.

3.2 Characteristics of Virtualized Environments

Virtualization refers to the process of creating a virtual version of something, whether it's hardware, software, storage, or a network. It allows multiple environments to run on the same physical system. There are three main components in a virtualized environment:

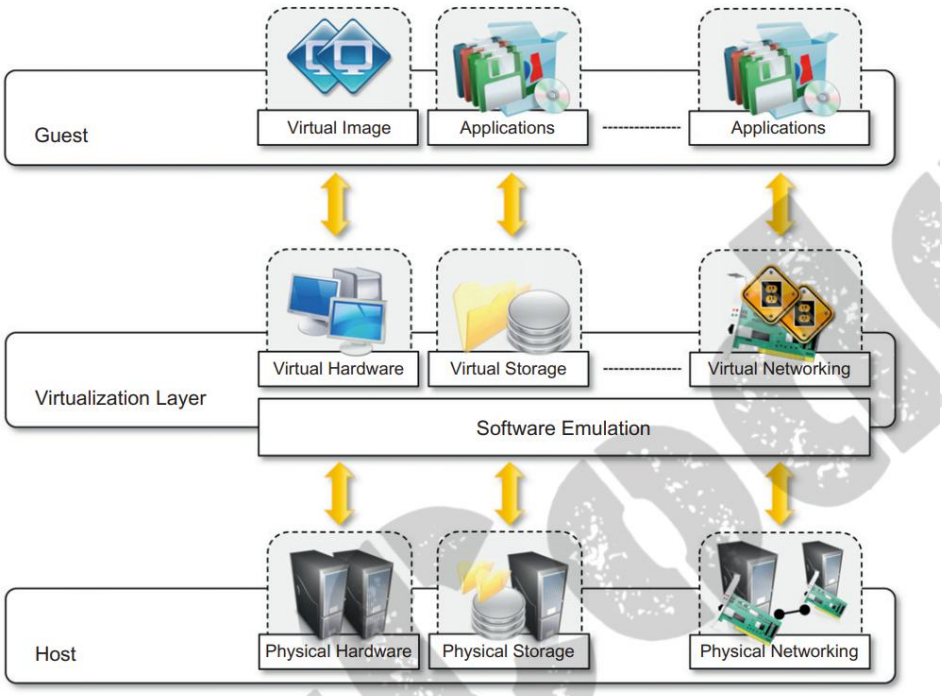


FIGURE 3.1
The virtualization reference model.

1. **Guest**: This is the virtual system (like a virtual machine) that runs applications or an operating system. The guest interacts with the virtual environment, not directly with the physical system.
2. **Host**: This is the physical system or server where the guest runs. The host provides the resources like CPU, memory, and storage for the virtual machines.
3. **Virtualization Layer**: This is the software that manages the virtual environment and creates a virtual system for the guest. It handles the communication between the guest and the host, making it appear to the guest as though it's running on its own hardware.

Types of Virtualization

- **Hardware Virtualization**: The virtual machine manager (also called a hypervisor) creates virtual hardware where the guest (an operating system and its applications) can run.
- **Virtual Storage**: Virtual storage management software lets applications interact with virtualized storage, not the real physical storage.

- **Virtual Networks**: Tools like VPN (Virtual Private Networks) make it appear that you're within a different physical network, allowing access to resources otherwise unavailable.

Key Characteristics of Virtualized Environments

1. **Increased Security**:
   - Virtualization makes it possible to run applications in a controlled and isolated environment. The guest system runs in a virtual space separate from the physical machine, which adds an extra layer of security.
   - For example, when you run programs in a virtual machine, they can't access the host's system directly, which protects the host from harmful operations.
   - Virtual machines are perfect for testing untrusted software without risking harm to the host system.
2. **Managed Execution**: Virtualization doesn't just add security but also allows better control over resources with these key features:

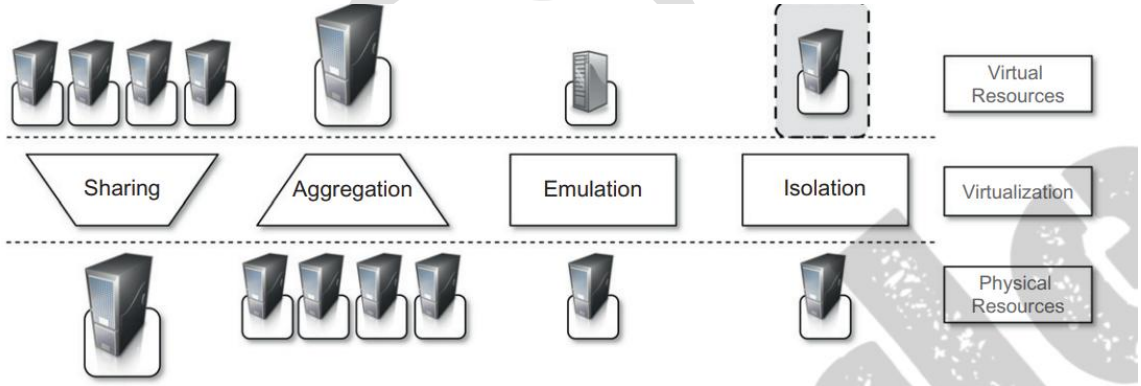


FIGURE 3.2
Functions enabled by managed execution.

   - **Sharing**: Virtualization allows multiple virtual systems to run on the same physical system, sharing the host's resources. For example, one physical server can support many virtual machines, making full use of powerful hardware. This is important for data centers where it helps reduce the number of servers in use, saving energy.
   - **Aggregation**: In some cases, virtualization combines several physical servers into one virtual system. For example, several physical machines can be grouped to appear as one virtual system to the guest. This is useful in distributed computing environments.

- **Emulation**: Emulation allows you to create a virtual environment that mimics a different system. For example, you can run a Windows operating system inside a virtual machine on a Linux computer. This is especially useful for testing software on different platforms or running older software that isn't compatible with modern systems.
- **Isolation**: Each virtual machine is isolated from the others. This means that even if one guest system crashes or is attacked, it won't affect the host or the other guest systems. This feature is essential for running multiple applications on the same hardware without worrying about interference between them.

3. **Portability**:
   - **Virtual machines are portable**, meaning they can easily be moved and run on different physical systems. For example, a virtual machine can be created on one server and later moved to another server with minimal effort. This is possible because the virtual machine is packaged as a "virtual image."
   - In programming, languages like Java use virtual machines that allow the same application to run on different operating systems without any changes.

Benefits of Virtualization

1. Cost Savings: Virtualization reduces the need for additional hardware, as multiple virtual machines (VMs) can run on a single physical server, lowering hardware and maintenance costs.
2. Efficient Resource Utilization: Resources like CPU, memory, and storage are dynamically allocated among virtual machines, ensuring optimal utilization without wastage.
3. Scalability: Virtualization allows easy provisioning of new virtual machines or scaling resources for existing ones without requiring new hardware.
4. Flexibility: Different operating systems and applications can run on the same hardware, making it easier to support diverse workloads.
5. Isolation: Virtual machines operate independently of each other, providing isolation that enhances security and reliability.
6. Simplified Management: Centralized management tools make it easier to monitor and control virtual machines, networks, and storage systems from a single interface.

7. Disaster Recovery: Virtualization supports features like snapshots and replication, enabling quick recovery of workloads in case of system failures.
8. Energy Efficiency: By consolidating multiple workloads on fewer physical servers, virtualization reduces power consumption and cooling requirements.
9. Testing and Development: Developers and testers can create isolated environments for testing software without impacting production systems.

3.3 Taxonomy of Virtualization Techniques

Virtualization is a way to create virtual (or pretend) versions of physical things, like computers or storage. There are different ways to do this, depending on what you want to virtualize. Here, we will break down the different types of virtualization and how they work.

Types of Virtualization

1. **Execution Virtualization**:
    - This means creating a virtual environment where programs or operating systems can run. Think of it as making a pretend computer that works like a real one. This is the most common and oldest form of virtualization.
    - There are two ways to do this:
        - **Process-Level Virtualization**: This type runs on top of an existing operating system (like Windows or Linux). The operating system controls everything, but it also creates a virtual space where programs can run separately.
        - **System-Level Virtualization**: This type interacts directly with the hardware (without an operating system). It creates virtual machines that can run their own operating systems.
2. **Storage Virtualization**:
    - Instead of using physical hard drives, storage virtualization makes it look like there is a single big storage space, even though it might be spread across many devices.

3. **Network Virtualization**:

   - This virtualizes networks. Instead of using physical network cables and routers, network virtualization makes it seem like computers are connected over a virtual network, even though they might not physically be in the same place.

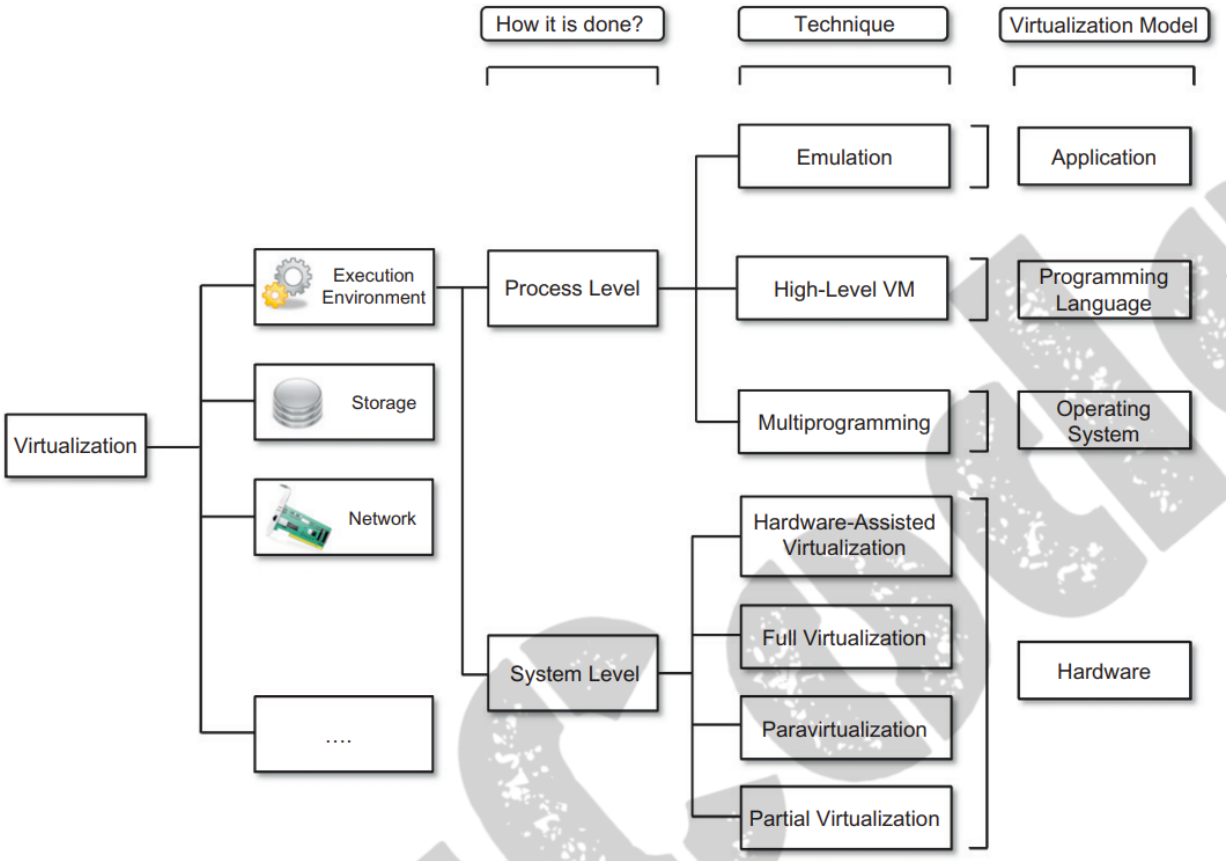


FIGURE 3.3

A taxonomy of virtualization techniques.

3.3.1 Execution Virtualization

Execution virtualization is about making pretend environments where programs, applications, or entire operating systems can run without knowing they are on virtual hardware.

Levels of Execution Virtualization

1. **Bare Hardware Virtualization**:

   - Here, the virtual environment talks directly to the physical hardware (like the CPU or memory). The virtual machine manager (also called a hypervisor) creates a pretend computer that runs like a real one.

2. **Operating System Virtualization**:
    - The virtual environment is created using the operating system's resources. For example, if you are using Linux, you can create multiple virtual spaces (like containers) to run programs.

Programming Language-Level Virtualization

Programming language-level virtualization is a type of virtualization focused on running applications in a controlled environment that supports easy deployment, platform compatibility, and efficient execution. Here's how it works:

1. **Execution Through Virtual Machines (VMs)**:
    - Instead of running directly on hardware, applications run on a **virtual machine** (VM) that executes **bytecode**—an intermediate code generated when the program is compiled.
    - This bytecode represents machine instructions for an abstract, simplified architecture and is not tied to any specific physical machine.
2. **Bytecode Interpretation and JIT Compilation**:
    - **Interpretation**: The bytecode can be interpreted, where each instruction is processed one-by-one, though this can slow down execution.
    - **Just-in-Time (JIT) Compilation**: The bytecode can also be "jitted," meaning it is compiled to machine code right before execution, which improves performance for frequently used instructions. The compiled code can be cached and reused, reducing the need for repeated compilation.
3. **Advantages**:
    - **Portability**: Applications compiled into bytecode can run on any platform with a compatible virtual machine, making it unnecessary to create separate versions for different systems.
    - **Security**: Virtual machines manage memory access and filter input/output operations, making it easier to sandbox applications and prevent unauthorized access to system resources.

- **Uniform Execution**: These virtual machines offer a consistent environment across various platforms, simplifying the development process

4. **Performance Considerations**:
    - Although interpreted or JIT-compiled languages generally run slower than natively compiled programs, improvements in JIT technology and powerful modern CPUs have helped reduce the performance gap. However, performance may still be a limitation for highly compute-intensive applications.

Application-Level Virtualization

Application-level virtualization lets applications run on systems that don't fully support them by emulating parts of the needed runtime environment, like certain libraries or OS components. This allows applications to work as if they're on a compatible system, without needing to install or modify the host.

How It Works

1. **Runtime Emulation**:
    - A lightweight layer provides the missing elements the application requires. This could include parts of a file system, libraries, or other system components.
2. **Emulation Methods**:
    - **Interpretation**: Each instruction is processed one by one, which can be slow.
    - **Binary Translation**: Instructions are converted in batches, which speeds up performance after initial setup.
3. **Key Advantages**:
    - **Compatibility**: Applications run even if the host lacks certain libraries or system components.
    - **Low Overhead**: Only necessary parts are emulated, making it faster and lighter than full system virtualization.

Examples

- **Wine**: Allows Linux to run Windows applications by mimicking Windows libraries.

- **CrossOver**: Similar to Wine, but for macOS.
- **VMware ThinApp**: Creates portable versions of applications that can run without installation on various systems.

1. **Hardware-Assisted Virtualization**:
    - Modern processors (like Intel's **Intel VT** and AMD's **AMD-V**) have special features that support virtualization. These features allow the CPU to manage sensitive instructions directly, without relying on the hypervisor to intercept them.
    - Before this hardware assistance, hypervisors had to use techniques like **binary translation** to manage privileged instructions, which slowed down performance. With hardware-assisted virtualization, the CPU helps the hypervisor directly manage these instructions, leading to better performance and efficiency.
    - **Example**: Most modern virtualization platforms, such as KVM, VMware, and Hyper-V, use hardware-assisted virtualization.
2. **Full Virtualization**:
    - In **full virtualization**, the guest operating system runs as if it were on real physical hardware without needing any modifications. The hypervisor completely emulates the underlying hardware, including the CPU, memory, and I/O devices.
    - Full virtualization provides **complete isolation** between the VMs and the host, making it highly secure. However, this full emulation can introduce performance overhead, especially when handling privileged instructions.
    - A combination of software and hardware-assisted techniques is used to optimize full virtualization.
    - **Example**: VMware Workstation, KVM, Hyper-V.
3. **Paravirtualization**:
    - In **paravirtualization**, the guest operating system is **aware** that it is running in a virtualized environment. The guest OS is modified to interact directly with the

hypervisor, allowing it to bypass the hypervisor for some tasks and perform them more efficiently on the host hardware.

- This results in better performance compared to full virtualization because there is less overhead. However, it requires changes to the guest OS, which might not always be possible.
- **Example**: Xen hypervisor.

4. **Partial Virtualization**:
   - **Partial virtualization** emulates only part of the hardware environment. Some aspects of the guest OS interact directly with the host hardware, while other parts are virtualized.
   - This technique allows applications to run on the guest OS but may not support all features of the operating system.
   - It's typically used for isolating application execution rather than fully virtualizing an entire OS.
   - **Example**: Time-sharing systems where multiple applications share the same memory and processor but don't have complete isolation from each other.
5. **Operating System-Level Virtualization**:
   - Also known as **containerization**, this type of virtualization creates isolated execution environments (called **containers**) within a single operating system.
   - The OS kernel allows multiple user-space instances to run concurrently, each in its own isolated environment.
   - Containers share the same kernel but provide isolated file systems, network interfaces, and resources for each instance. This is more lightweight compared to full hardware virtualization because it doesn't require a full OS for each virtual instance.
   - **Example**: Docker, FreeBSD Jails, Solaris Zones, OpenVZ.

OTHER TYPES OF VIRTUALIZATION

Other Types of Virtualization

In addition to **execution virtualization**, there are several other types of virtualization that provide an abstract environment for specific resources, like storage, networking, and desktop environments.

3.3.2.1 Storage Virtualization

- **Purpose**: Separates the physical storage hardware from its logical representation, allowing users to access data without needing to know where it's physically stored.
- **How It Works**: Combines multiple storage resources into a single, unified system. A common example is **Storage Area Networks (SANs)**, which use high-speed networks to provide accessible, centralized storage.

3.3.2.2 Network Virtualization

- **Purpose**: Creates a virtual version of physical networks, which can simplify and optimize network resources.
- **Types**:
    - **External Network Virtualization**: Merges multiple physical networks into a single virtual network, such as a **Virtual LAN (VLAN)**, to make hosts communicate as if they're in the same domain.
    - **Internal Network Virtualization**: Provides network functionality within a single operating system partition, allowing virtual machines to communicate via NAT or private networks without direct access to the host's network.

3.3.2.3 Desktop Virtualization

- **Purpose**: Makes a desktop environment accessible from any device, often remotely.
- **How It Works**: The desktop environment is stored on a server or data center and accessed through a network. This is common in cloud computing, where users can access their desktop from different devices.
- **Examples**: Windows Remote Services, Virtual Network Computing (VNC), and cloud solutions like Citrix XenDesktop.

3.3.2.4 Application Server Virtualization

- **Purpose**: Combines multiple application servers into one virtualized environment, allowing for load balancing and high availability.
- **How It Works**: Distributes applications across several servers but presents them as a single service to users. This improves performance and reliability.

3.3.1.1 Machine Reference Model

To understand how virtualization works, we use a **machine reference model**. This model shows the different layers in a computer system and how they interact with each other.

Here's a breakdown of the layers:

1. **Hardware Layer (Bottom Layer)**:
    - This is where the physical components of the computer, like the CPU, memory, and hard drive, are found.
    - The **Instruction Set Architecture (ISA)** sits here. ISA is like a set of rules that tells the hardware how to execute instructions, such as storing data or performing calculations. It's important for both the operating system (OS) and the software that interacts directly with the hardware.
2. **Operating System (OS) Layer**:
    - The OS manages the hardware. It decides how to share resources (like memory) between programs.
    - This layer interacts with the **Application Binary Interface (ABI)**. ABI is like a translator between the OS and the programs. It allows programs to use the OS to access hardware without needing to know the hardware details.
3. **Application Layer (Top Layer)**:
    - This is where applications (like web browsers or games) run.
    - Applications use the **Application Programming Interface (API)** to interact with the OS or libraries to perform tasks, like opening files or connecting to the internet.

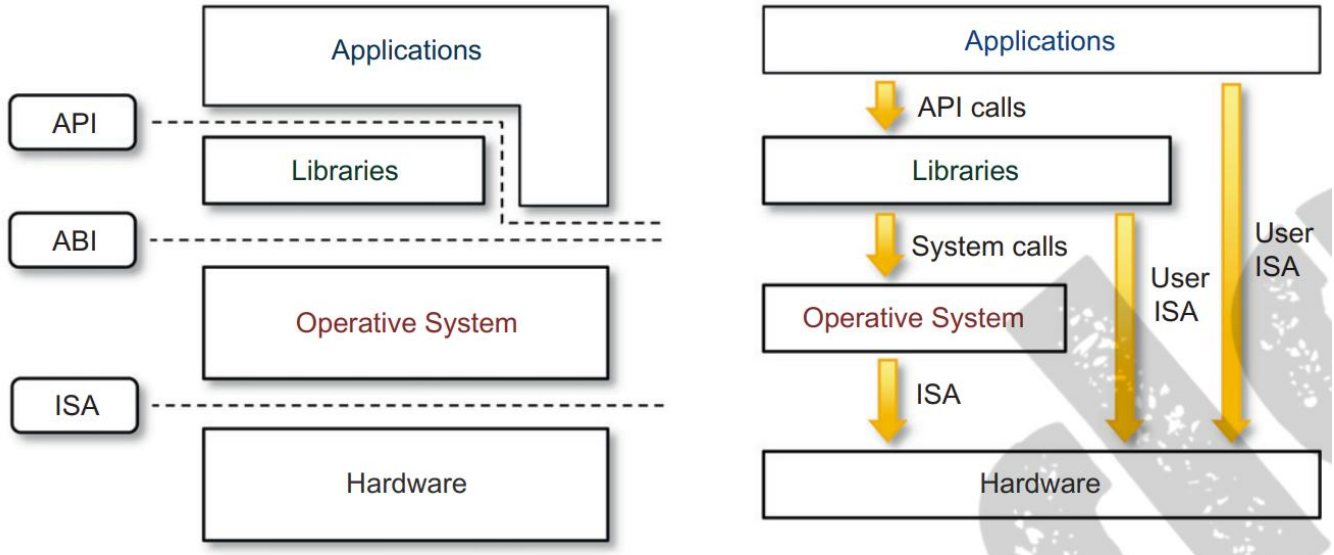


FIGURE 3.4

A machine reference model.

How Virtualization Fits In

In virtualization, one of these layers is replaced or modified. For example, a **hypervisor** (software that creates virtual machines) sits between the hardware and the OS. It creates multiple virtual computers, each with its own OS, using the same physical hardware.

By separating these layers, virtualization makes it possible to run multiple operating systems (or environments) on the same physical hardware.

Privileged and Non-Privileged Instructions

To make sure the system is secure and stable, different instructions (commands) in a computer have different privilege levels:

- **Privileged Instructions**: These are sensitive commands that control important hardware resources, like CPU registers. Only the **supervisor mode** (used by the OS or hypervisor) can run these commands.
- **Non-Privileged Instructions**: These are safer commands (like basic math operations) that can be used by regular applications.

Security Rings

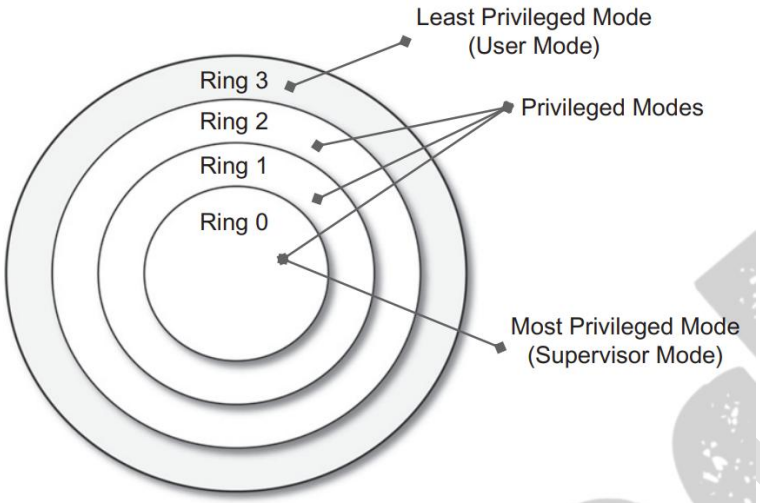


FIGURE 3.5
Security rings and privilege modes.

Computer systems have different **rings** or levels of security:

- **Ring 0**: The most powerful level (used by the OS kernel or hypervisor). It has full control of the hardware.
- **Ring 3**: The least powerful level (used by regular applications). It has limited access to the system to avoid damaging or interfering with other processes.

Virtualization uses these rings to create isolated environments. For example, the hypervisor runs in **Ring 0**, while virtual machines run in less privileged rings (Ring 1 or 3), keeping them separate and secure.

Key Point: Hypervisor's Role

A hypervisor, also known as a virtual machine manager, creates virtual machines and runs them in a controlled environment. It manages the virtual machines and ensures they don't interfere with each other or the physical machine.

For virtualization to work efficiently, all privileged instructions need to be controlled by the hypervisor. However, older systems had instructions that could be run by regular applications (user mode) without the hypervisor's control, which caused problems. Newer systems (like **Intel VT** or **AMD Pacifica**) solved this by making these sensitive instructions privileged.

3.3.1.2 Hardware-Level Virtualization

Hardware-level virtualization refers to the process of creating a virtual environment directly on top of the physical computer hardware, allowing multiple operating systems (OS) to run simultaneously as **guests** on the same hardware. This type of virtualization abstracts the physical resources, such as the CPU, memory, and storage, to create multiple isolated virtual machines (VMs). Each VM behaves as if it were running on a separate physical computer.

The **virtual machine manager (VMM)** or **hypervisor** is the key software layer that facilitates this process by managing the interaction between the guest operating systems and the physical hardware.

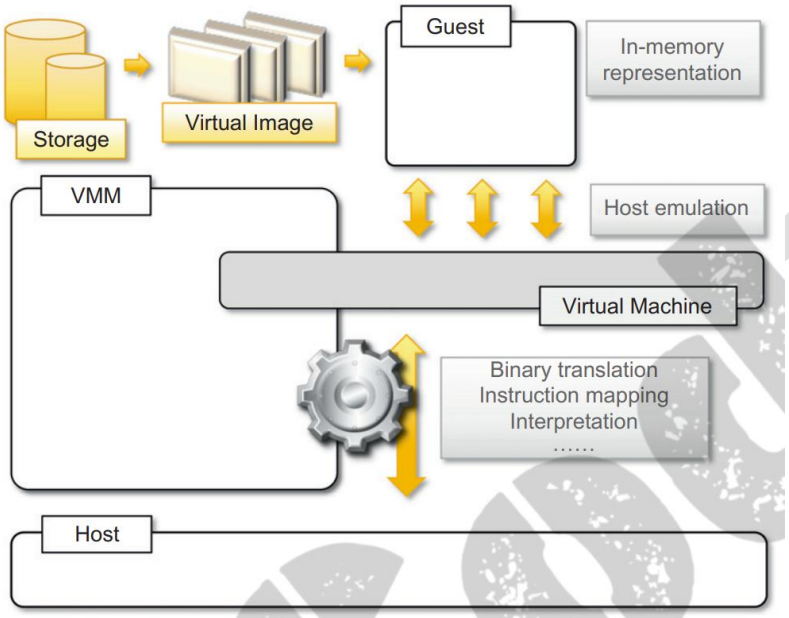


FIGURE 3.6
A hardware virtualization reference model.

Key Components of the Hypervisor

A hypervisor consists of three main components that manage the execution of the guest virtual machines:

1. **Dispatcher**:
    - This is the "gatekeeper" of the hypervisor. Every instruction from the virtual machine passes through the dispatcher, which decides whether to send the instruction to the **allocator** or the **interpreter**.

2. **Allocator**:
    - The allocator is responsible for assigning hardware resources (like memory, CPU time, etc.) to the virtual machines. When a VM tries to execute an instruction that modifies the resources allocated to it (e.g., requesting more memory), the dispatcher sends this request to the allocator.
3. **Interpreter**:
    - The interpreter handles **privileged instructions**—those that need special access to hardware resources, such as controlling I/O devices or modifying CPU registers. When a VM attempts to execute these sensitive operations, the interpreter steps in to safely manage the request, ensuring that the VM doesn't directly alter the physical system.

These three components work together to ensure the smooth functioning of VMs while isolating them from one another and the host.

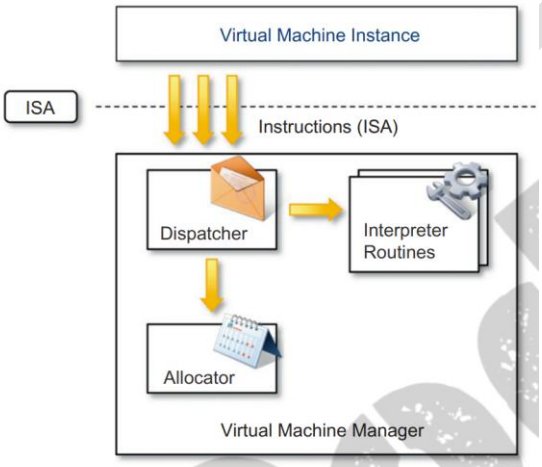


FIGURE 3.8
A hypervisor reference architecture.

Properties of a Virtual Machine Manager (Popek and Goldberg, 1974)

In 1974, Popek and Goldberg defined three key properties that a virtual machine manager (VMM) must satisfy to support effective hardware-level virtualization:

1. **Equivalence**:
    - The guest operating system should behave as though it is running directly on the physical hardware. In other words, the VM should function as if there is no virtual layer at all, without affecting its performance or behavior.
2. **Resource Control**:

- The VMM must have complete control over the resources (CPU, memory, disk, etc.). This ensures that no guest operating system can interfere with the host or other guests.

3. **Efficiency**:
   - Most instructions executed by the guest OS should not require intervention by the hypervisor. Only special instructions, such as those that modify hardware configurations, should be handled by the hypervisor to avoid performance bottlenecks.

These properties ensure that virtualization remains both secure and efficient.

Types of Hypervisors

A **hypervisor** (or virtual machine manager) is the core component of hardware-level virtualization. It creates, manages, and controls the virtual machines. There are two main types of hypervisors:

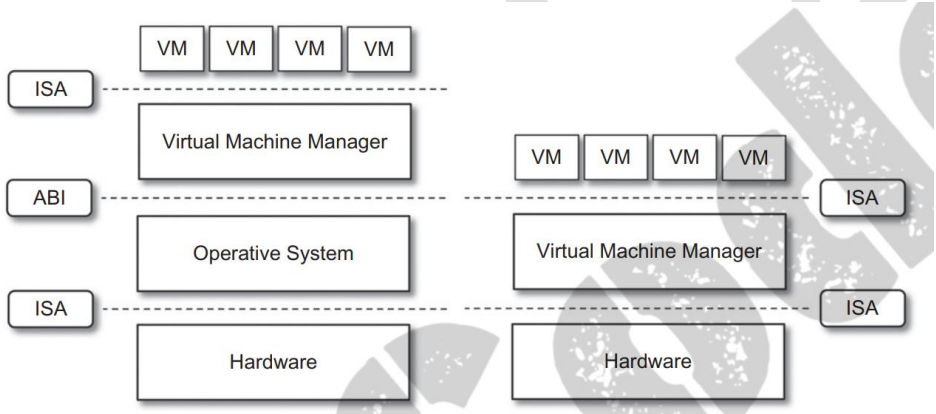


FIGURE 3.7

Hosted (left) and native (right) virtual machines. This figure provides a graphical representation of the two types of hypervisors.

1. **Type I Hypervisor (Native or Bare-Metal Hypervisor)**:
   - **Runs directly on the hardware**, replacing the operating system.
   - It interacts directly with the hardware via the **Instruction Set Architecture (ISA)** interface, emulating the hardware to manage the guest OS.
   - **Example**: VMware ESXi, Microsoft Hyper-V, Xen.
   - Type I hypervisors are commonly used in data centers and cloud environments because they offer high performance and control.

2. **Type II Hypervisor (Hosted Hypervisor)**:
    - **Runs on top of an existing operating system**, meaning it is a software application managed by the OS.
    - The hypervisor interacts with the OS through the **Application Binary Interface (ABI)** and emulates the hardware for the guest OS.
    - **Example**: VMware Workstation, Oracle VirtualBox.
    - Type II hypervisors are easier to set up and are typically used on personal computers for testing and development purposes.

3.4 Virtualization and Cloud Computing

Virtualization is crucial in **cloud computing** because it enables flexible, secure, and isolated environments for delivering on-demand IT services. Here's how it fits into cloud infrastructure:

1. **Core Uses of Virtualization in Cloud Computing**:
    - **Customizable Environments**: Virtualization allows users to tailor computing environments to specific needs while maintaining strong isolation and security between different users.
    - **Resource Management**: It enables efficient resource utilization by allocating virtual resources (like storage and processing power) based on demand, reducing waste.
2. **Key Virtualization Techniques in Cloud Computing**:
    - **Hardware Virtualization**: Essential for **Infrastructure-as-a-Service (IaaS)**, where users can rent virtualized computing infrastructure. This is managed using virtual machines, allowing secure, isolated environments on shared physical hardware.
    - **Programming Language Virtualization**: Used in **Platform-as-a-Service (PaaS)** solutions, such as Java or .NET platforms, to run applications in a controlled and sandboxed environment.
3. **Storage Virtualization**:
    - Cloud providers use storage virtualization to offer **scalable storage solutions**. They partition storage into manageable units or slices that can be dynamically resized and managed.

4. **Desktop Virtualization in Cloud**:
    - Cloud desktop virtualization allows full desktop environments to be hosted in the cloud, accessible via the internet from thin clients. This offers users flexibility to access their desktops from anywhere, mirroring the convenience of traditional desktop virtualization but with cloud-based scalability.

Virtualization enhances cloud computing by enabling efficient, flexible, and isolated virtual environments for users. This makes it possible for providers to deliver reliable and scalable cloud services across IaaS, PaaS, and even full desktop solutions.

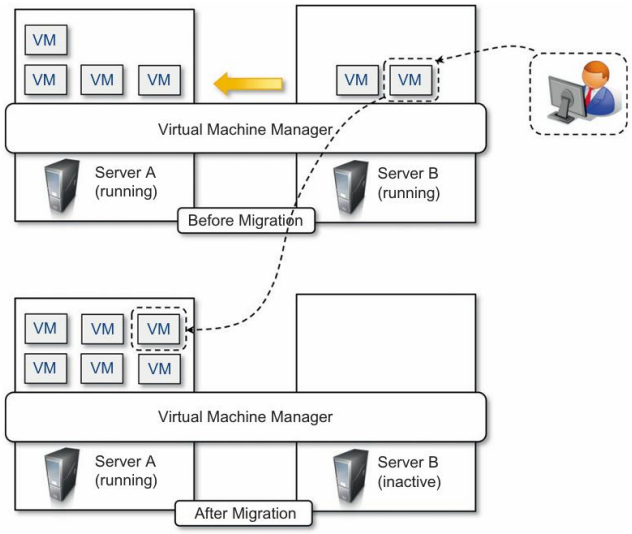


FIGURE 3.10
Live migration and server consolidation.

3.5 Pros and Cons of Virtualization

Virtualization is now widely used, particularly in cloud computing, as advancements in technology have addressed many past limitations. However, it has both benefits and potential drawbacks.

3.5.1 Advantages of Virtualization in Cloud:

1. Server Consolidation: Combines multiple virtual machines on fewer physical servers when resources are underutilized. This reduces hardware costs and energy consumption.

- **Virtual Machine Migration**:
    - **Live Migration**: Moves virtual machines across physical hosts without downtime, maintaining uninterrupted service.

2. **Managed Execution and Isolation**:
    - Virtualization allows applications to run in controlled, isolated environments. This isolation means that harmful actions or faults within one VM won't affect other VMs or the host system. For example, a VM acting as a sandbox can prevent harmful software from impacting the host, making it ideal for secure environments and **server consolidation** scenarios.
3. **Portability**:
    - Virtual machines (VMs) are typically represented as files, which makes them highly portable. For instance, you can create a VM on one physical server, then move it to another location by transferring the file. This portability also simplifies **migration** between servers, essential for load balancing or maintenance, and allows applications to "compile once, run anywhere" if the right virtual environment is available.
4. **Cost Efficiency and Resource Optimization**:
    - Virtualization can reduce hardware costs by consolidating multiple VMs on a single physical machine. This reduces the need for additional physical servers and cuts down on power and cooling requirements. For example, multiple virtual servers sharing one physical machine use only the required resources, minimizing waste.
5. **Control and Simplified Management**:
    - Because virtual environments are isolated, they provide controlled execution that limits risks to the hardware. VMs allow fine-tuned control over resources, like memory or processing power, which is crucial in managing performance for different applications.

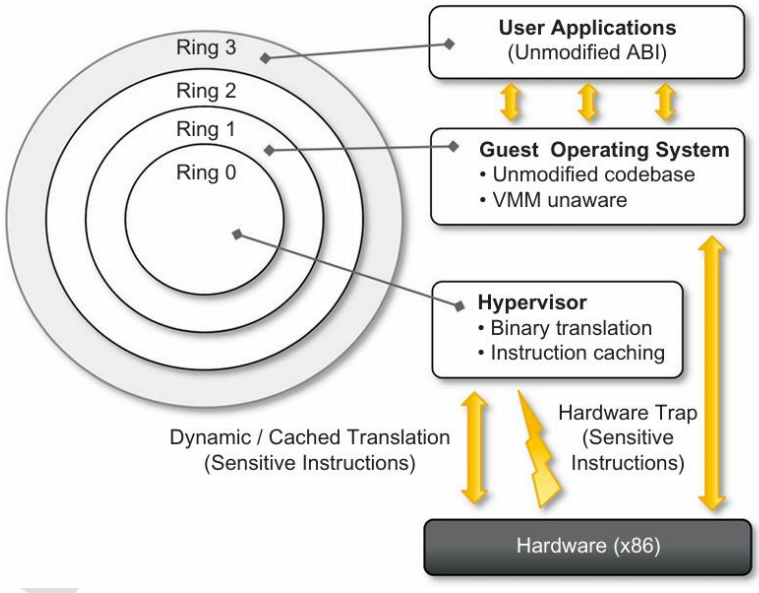
3.5.2 Disadvantages of Virtualization

1. **Performance Degradation**:
    - The virtualization layer, which sits between the VM and the host hardware, can cause **latency** or delays. For example, in hardware virtualization, the VM may need additional steps to process tasks, like:
        - **Managing virtual processors**,
        - **Handling privileged instructions** (instructions the VM can't directly execute),
        - **Paging within VM memory**.
    - Additionally, if the VM software is running on top of an OS, it competes for resources, slowing down both the virtual and host systems. Techniques like **paravirtualization** and **native code**

compilation help reduce these delays by allowing the VM to directly access some host resources.

2. **Inefficiency and Limited Access to Host Features**:
    - Some of the advanced features of the host hardware may not be fully available within a virtual environment. For instance, a VM might only use a simplified virtual graphic card with limited functionality, which can affect performance and user experience for certain applications.
3. **Security Vulnerabilities**:
    - Virtualization introduces unique security risks. For example, **phishing attacks** targeting VMs can imitate the environment, capturing sensitive data. Malware like **BluePill** and **SubVirt** can install themselves as a thin layer between the host OS and the hardware, gaining control over the OS and extracting information. Such malware takes advantage of virtualization vulnerabilities in CPUs not initially designed with security in mind, although newer processors from Intel (Intel VT) and AMD (Pacifica) now include improved security features.

Reference Model of Full Virtualization



The reference model for full virtualization explains how a virtualized environment allows guest operating systems to run unmodified while ensuring isolation and efficient utilization of physical hardware. This process involves multiple layers, each contributing to the abstraction and execution of virtual machines.

Key Layers and Components

1. **Hardware (x86 Architecture)**:
    - Represents the physical hardware layer, including CPU, memory, and I/O devices.

- It is shared among multiple virtual machines, with virtualization managed by the hypervisor.

2. **Hypervisor**:
   - Acts as an abstraction layer between the physical hardware and virtual machines.
   - Responsible for:
     - **Binary Translation**: Converts privileged or sensitive instructions from the guest OS into safe instructions executable by the hardware.
     - **Instruction Caching**: Optimizes performance by caching frequently translated instructions.
     - **Hardware Traps**: Captures sensitive instructions from the guest OS and safely emulates them.
3. **Guest Operating System**:
   - The guest OS is unaware of being virtualized ("VMM-unaware").
   - Operates as if it has full access to the underlying hardware, even though the hypervisor intercepts and manages all privileged instructions.
4. **User Applications**:
   - Applications running within the virtual machine interact with the guest OS using the Application Binary Interface (ABI).
   - These applications remain unmodified and function as they would on a physical system.

Privilege Rings and Execution

The x86 architecture supports **privilege rings**, which control access to hardware resources. These rings are layered hierarchically:

- **Ring 0**: Used by the hypervisor for full control over hardware (highest privilege).
- **Ring 1 and Ring 2**: Typically unused or reserved for specific tasks.
- **Ring 3**: Used by user-level applications (lowest privilege).

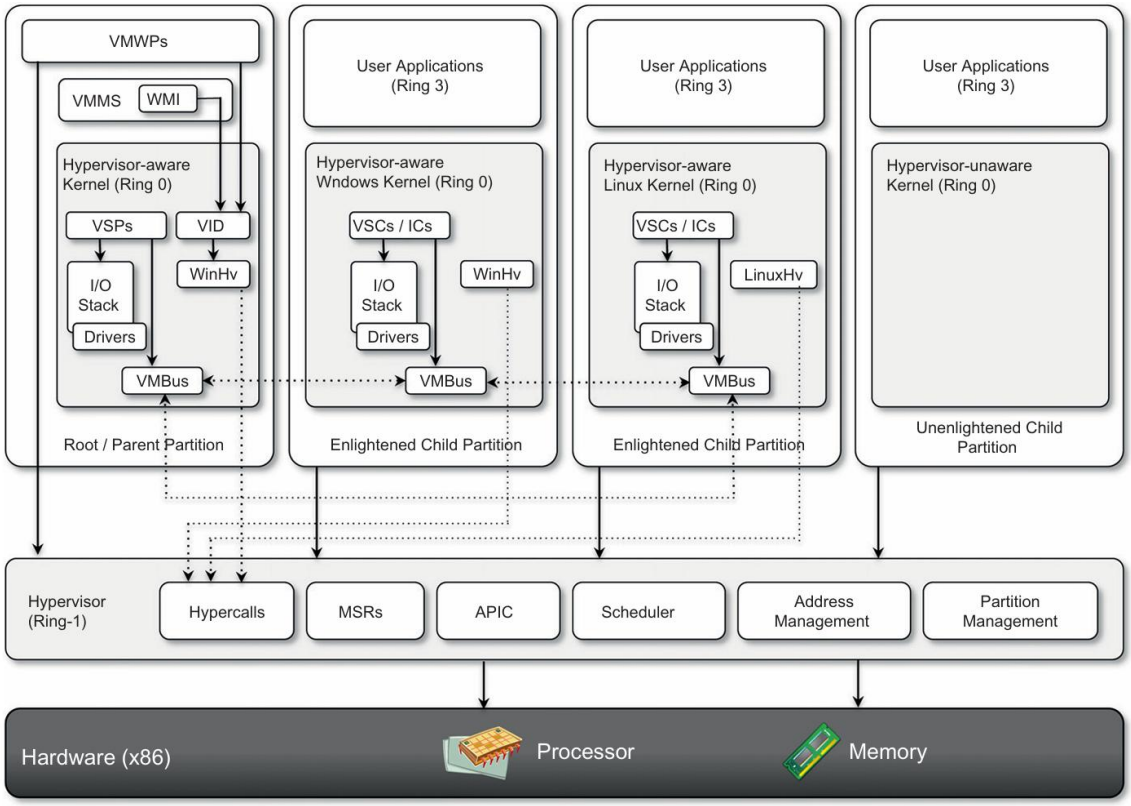


FIGURE 3.17
Microsoft Hyper-V architecture.

Architecture of Hyper-V

Hyper-V is Microsoft's virtualization platform that provides a foundation for running virtual machines (VMs) on physical hardware. Its architecture is designed to ensure efficiency, scalability, and security for virtualized environments, making it a key component in modern cloud computing.

Key Components of Hyper-V Architecture

1. **Hypervisor (Virtual Machine Monitor - VMM):**
   - The core layer of Hyper-V.
   - It runs directly on physical hardware and is responsible for managing hardware resources (CPU, memory, network, and storage) across virtual machines.
   - Operates in **Ring -1** (root level), ensuring complete control over the hardware.
2. **Parent Partition (Management OS):**
   - The first VM created on the Hyper-V host.
   - Runs a specialized version of Windows (called Windows Server Core or Windows with Hyper-V enabled).

- Manages and interacts with the hypervisor.
- Responsible for:
  - Creating and managing child partitions (guest VMs).
  - Allocating virtualized resources.
  - Providing I/O request handling for guest partitions through Virtual Service Providers (VSPs).

3. **Child Partitions (Guest VMs):**
   - The partitions where guest operating systems (Windows, Linux, etc.) run.
   - Do not have direct access to hardware but communicate with the parent partition for resource allocation.
   - Use **Virtual Service Clients (VSCs)** to handle I/O requests via the parent partition.
4. **Virtual Devices:**
   - Hyper-V virtualizes hardware components (network adapters, storage, etc.) for guest VMs.
   - These virtual devices allow guest OSs to use hardware resources efficiently.
5. **Integration Services:**
   - A set of drivers and services installed on guest operating systems.
   - Improves VM performance and enables better communication between guest VMs and the host.
   - Features include:
     - Time synchronization.
     - Host-guest file sharing.
     - Enhanced device drivers for virtualized hardware.
6. **Virtual Switch:**
   - A software-defined networking component.
   - Enables VMs to communicate with each other and the external network.
   - Provides network isolation, traffic monitoring, and resource allocation for virtual networks.

Hyper-V in Cloud Computing

Hyper-V plays a significant role in cloud computing by enabling virtualization, scalability, and efficient resource utilization. Its features align with the requirements of Infrastructure as a Service (IaaS) in cloud environments.

Uses of Hyper-V in Cloud Computing:

1. **Efficient Resource Utilization**:
    - Hyper-V enables organizations to run multiple virtual machines on a single physical server, maximizing hardware usage.
2. **Scalability**:
    - Supports dynamic scaling of resources for VMs based on workload demands, which is essential for cloud environments.
3. **Multi-Tenancy**:
    - Isolates virtual machines to ensure data and operational security, enabling cloud providers to host multiple clients on shared hardware.
4. **Disaster Recovery and Backup**:
    - Hyper-V supports live migration, replication, and snapshot capabilities, ensuring data redundancy and quick recovery.
5. **Cost-Effectiveness**:
    - Reduces the need for physical hardware, lowering infrastructure and operational costs.