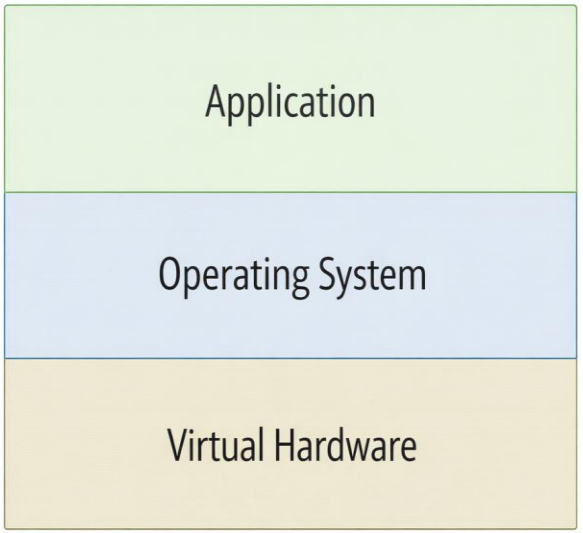
UNIT-I

Understanding Virtual Machines

Understanding Virtual Machines

- VMs represent the core abstraction that enables virtualization.
- They allow a complete computing environment to be implemented entirely in software while sharing physical hardware.



Introduction

- VMs are the fundamental components of virtualization.
- They serve as containers (not to confused with OS level container) be for traditional OS and applications.
- These OS and applications run on top of a hypervisor installed on a physical server.

Why Study Virtual Machines?

- VMs form the core abstraction that enables modern data centers.
- They allow consolidation, flexibility, isolation, and portability.
- Understanding VMs is essential for understanding cloud computing and modern infrastructure.

Chapter Scope and Focus

- This chapter focuses on how virtual machines differ from physical machines.
- It explains how VMs interact with the physical hardware underneath.
- It also introduces foundational VM management concepts.

Characteristics Virtual Machine

- A virtual machine (VM) has many of the same characteristics as a physical server.
- It supports an operating system and applications.
- Resources are allocated to the VM just as they are to a physical machine.

VMs Compared to Physical Servers

- A physical server typically runs a single operating system.
- A VM runs alongside many other VMs on the same physical server.
- Each VM can run a different operating system and different applications.

Multiplicity of Virtual Machines

- Multiple VMs can coexist on a single physical host.
- These VMs are isolated from one another.
- Isolation ensures stability, security, and fault containment.

VM as a Set of Files

- Unlike physical servers, a VM is not a tangible object.
- A VM exists as a collection of files stored on disk.
- These files fully describe the virtual server.

Implications of File-Based Existence

- Because VMs are files, they can be copied, moved, and backed up easily.
- This property enables advanced management features.
- It fundamentally changes how servers are provisioned and maintained.

Virtual Machine Configuration Files

- The configuration file defines the virtual hardware of the VM.
- It specifies CPU count, memory allocation, storage controllers, and networking.
- This file acts as a blueprint for the VM.
- Ex. VMware VM configuration (.vmx)

```
.encoding = "UTF-8"
config.version = "8"
virtualHW.version = "19"

displayName = "Ubuntu-VM"
guestOS = "ubuntu-64"

memsize = "4096"
numvcpus = "2"

firmware = "efi"

# Disk
scsi0.present = "TRUE"
scsi0.virtualDev = "lsilogic"
scsi0:0.present = "TRUE"
scsi0:0.fileName = "Ubuntu-VM.vmdk"

# Network
ethernet0.present = "TRUE"
ethernet0.connectionType = "nat"
ethernet0.virtualDev = "vmxnet3"

# CD/DVD
ide1:0.present = "TRUE"
ide1:0.deviceType = "cdrom-image"
ide1:0.fileName = "ubuntu-22.04.iso"

# Graphics
svga.present = "TRUE"

# USB
usb.present = "TRUE"

# Power settings
powerType.powerOff = "soft"
powerType.reset = "soft"
powerType.suspend = "soft"
```

Analogy: VM as an Empty Server Chassis

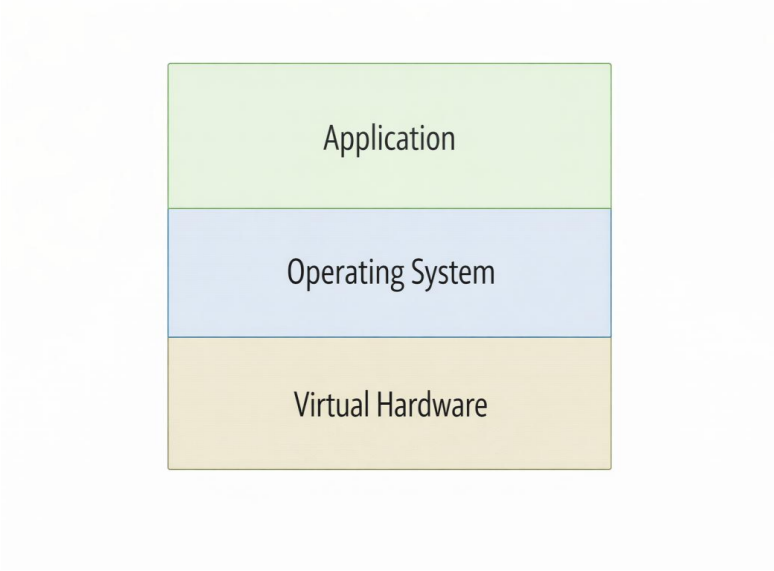
- A VM can be thought of as an empty server chassis.
- The configuration file lists which components are installed.
- Software gives the VM its function and purpose.

Virtual Disk Files

- Virtual disk files store the operating system and application data.
- They appear as standard disks inside the guest OS.
- Physically, they are files on shared storage.

Figure 3.1 – Virtual Machine Structure

- Figure illustrates a simplified VM.
- It shows the relationship between virtual hardware and the OS.
- This figure helps visualize the VM as a complete computer.



Two Perspectives of a Virtual Machine

- A VM can be viewed from two perspectives.
- The external view focuses on host resources.
- The internal view focuses on the guest operating system.

External Perspective of a VM

- From outside, administrators see VM configuration and files.
- Resources originate from the host server.
- The hypervisor manages allocation and scheduling.

Internal Perspective of a VM

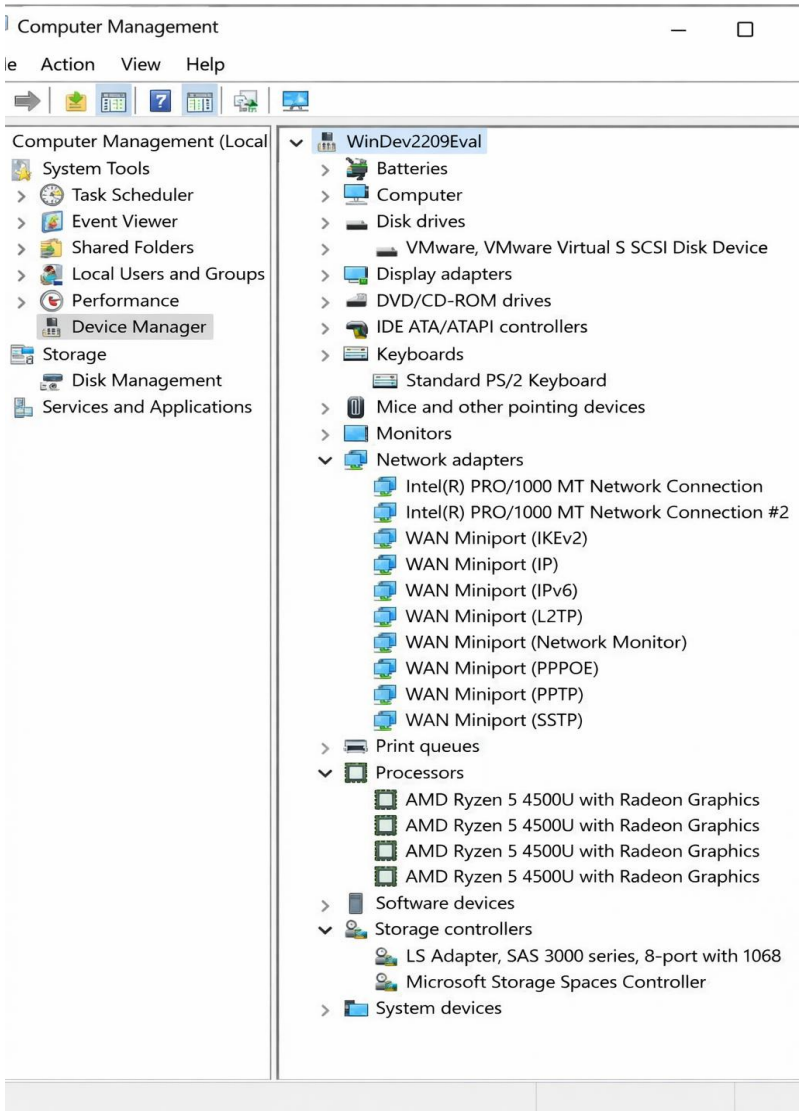
- From inside, the VM appears identical to a physical machine.
- The OS sees CPUs, memory, disks, and network interfaces.
- Applications run without modification.

Transparency of Virtualization

- The goal of virtualization is transparency.
- Guest operating systems are unaware of virtualization.
- This enables compatibility with existing software.

Figure 3.2 – Device Manager View

- Figure shows the Windows Device Manager inside a VM.
- Some devices differ from physical counterparts.
- These devices are standardized and portable.



Virtual vs Physical Device Drivers

- Virtual devices use specialized drivers.
- These drivers are optimized for virtual environments.
- They improve performance and efficiency.

CPU Resources in Virtual Machines

- Each VM is configured with one or more virtual CPUs.
- A virtual CPU represents scheduled access to physical CPUs.
- The VM does not own a physical processor.

Hypervisor CPU Scheduling

- The hypervisor schedules CPU cycles across all VMs.
- CPU time is shared dynamically.
- This enables efficient utilization.

No Dedicated CPUs

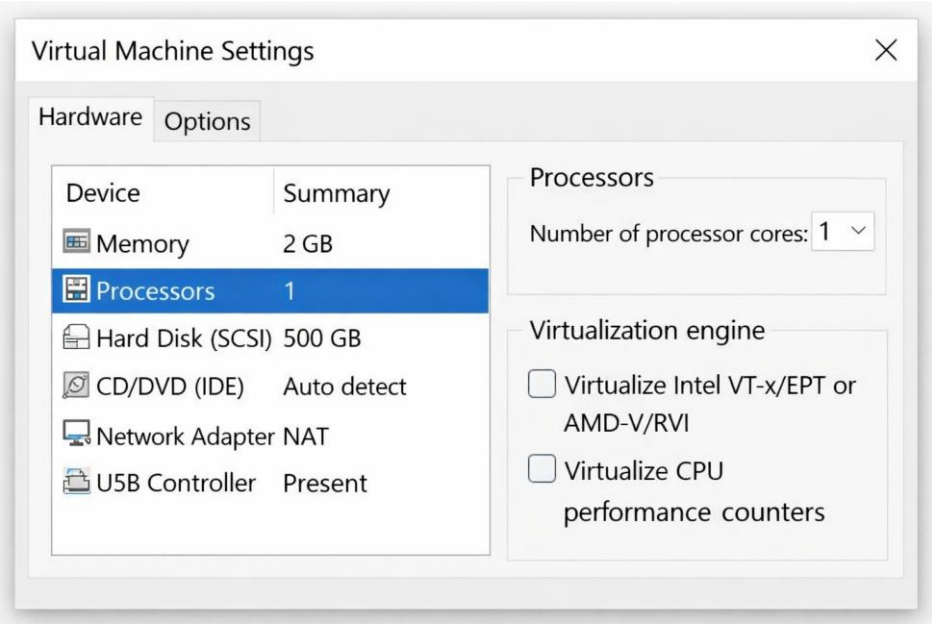
- Physical CPUs are not dedicated to individual VMs.
- Dedicated CPUs would undermine consolidation.
- Virtualization relies on time-sharing.

Multi-Core and Multi-Socket Systems

- Modern servers have multiple CPU sockets.
- Each socket contains multiple cores.
- VMs view each core as a virtual CPU.

Figure 3.3 – CPU Scheduling Model

- Figure illustrates CPU scheduling for a VM.
- Requests are intercepted by the hypervisor.
- Results are returned transparently.



Memory in Virtual Machines

- Memory is critical for VM performance.
- Each VM is allocated a fixed amount of RAM.
- The VM cannot exceed its allocation.

Memory Overcommitment

- Hypervisors support advanced memory techniques.
- Physical memory is shared across VMs.
- Efficient use reduces wasted resources.

Balancing Memory Allocation

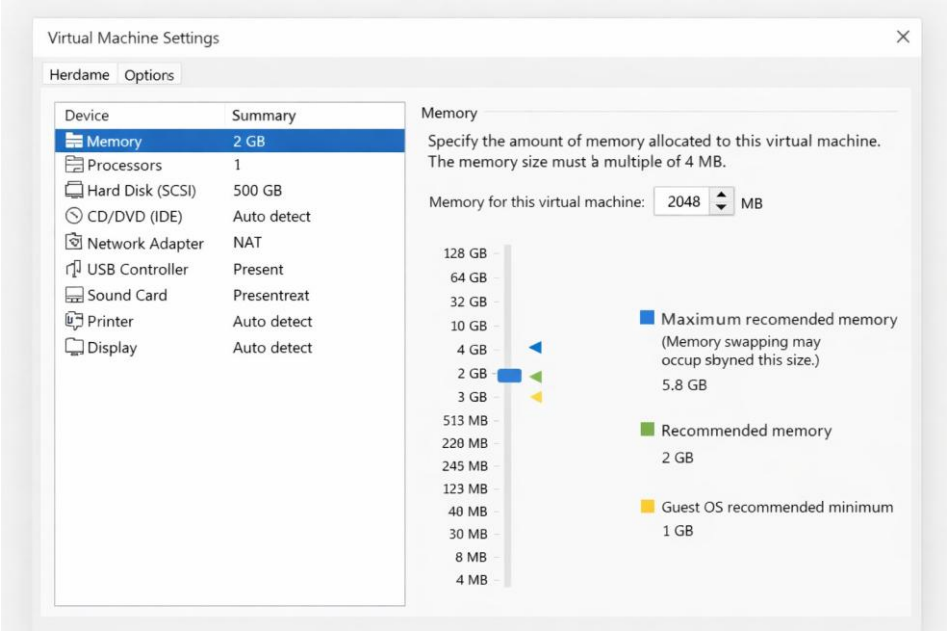
- Too little memory causes performance issues.
- Too much memory wastes shared resources.
- Careful planning is required.

Dynamic Memory Reconfiguration

- Some hypervisors allow memory changes without reboot.
- This improves availability.
- Administrators can respond to demand quickly.

Figure 3.4 – Memory Allocation

- Figure illustrates memory assigned to a VM.
- The hypervisor enforces allocation limits.
- Physical memory complexity is hidden.



Virtual Networking Overview

- Virtual networking enables communication.
- Each VM has one or more virtual NICs.
- These NICs connect to virtual networks.

Virtual NICs

- Virtual NICs appear as standard network adapters.
- They are software-defined.
- They provide consistent hardware interfaces.

Virtual Switches

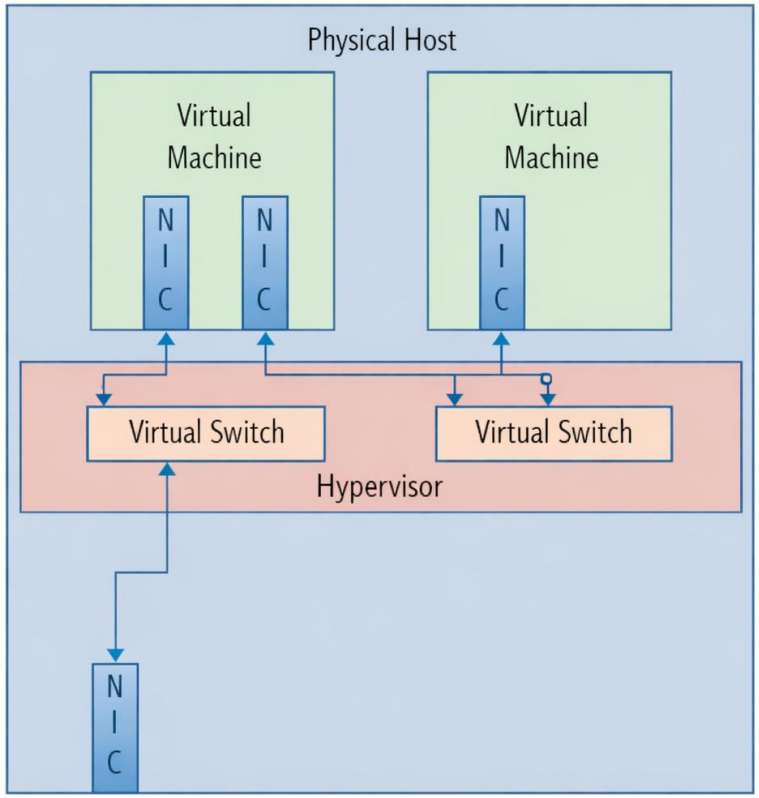
- Virtual switches are created by the hypervisor.
- They connect VMs to each other.
- They may also connect to physical networks.
- They:
  - Connect **VM** ↔ **VM**
  - Connect **VM** ↔ **host**
  - Connect **VM** ↔ **physical network** (via uplink)
  - VMs plug into a virtual switch using **virtual NICs**.

Physical NIC Integration

- Physical NICs act as uplinks (The connection that links a virtual switch to the physical network.)
- They connect virtual networks to external systems.
- This separation improves flexibility.

Figure 3.5 – Virtual Networking Model

- Figure illustrates VM networking.
- Virtual NICs connect to virtual switches.
- Physical NICs provide external access.

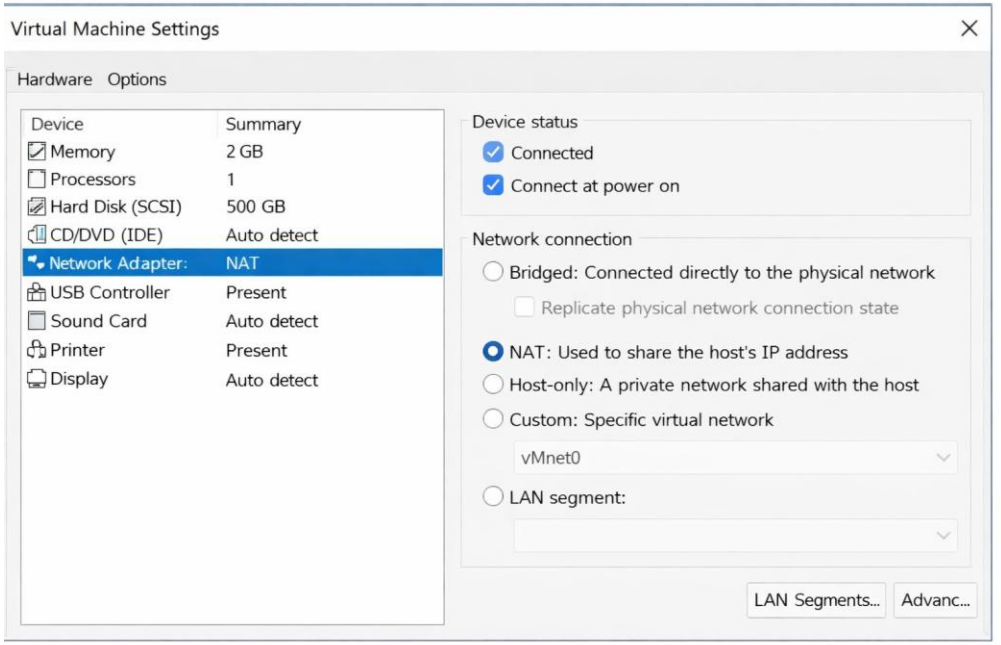


Security Through Virtual Networking

- VMs can communicate without leaving the host.
- Internal traffic is isolated.
- This reduces attack surfaces.

Figure 3.6 – Networking Options

- Figure shows different networking modes.
- Isolation and connectivity can be tuned.
- Design impacts security and performance.



Virtual Storage Fundamentals

- Virtual machines require persistent storage.
- They see logical disk drives.
- Storage is abstracted by the hypervisor.

Logical vs Physical Storage

- Logical drives are presented to the VM.
- Physical storage may be local or shared.
- The VM is unaware of the physical layout.

Virtual Storage Adapters

- VMs communicate via virtual storage adapters.
- Adapters are standardized.
- Virtual machines do **not access physical disks directly**. Instead, they use **virtual storage controllers/adapters** (e.g., virtual SCSI, SATA, NVMe) provided by the hypervisor.
- The VM thinks it is talking to a disk controller, but it's actually talking to a **software-defined adapter**.
- Examples of standardized virtual adapters:
  - LSI Logic / VirtIO (storage)
  - IDE / SATA / SCSI (virtual)

Hypervisor Storage Translation

- The hypervisor translates VM requests.
- Data blocks are mapped to physical storage.
- Storage protocols are hidden.

Figure 3.7 – VM Storage View

- Figure shows drives as seen by the VM.
- They resemble traditional disks.
- Underlying storage is pooled.

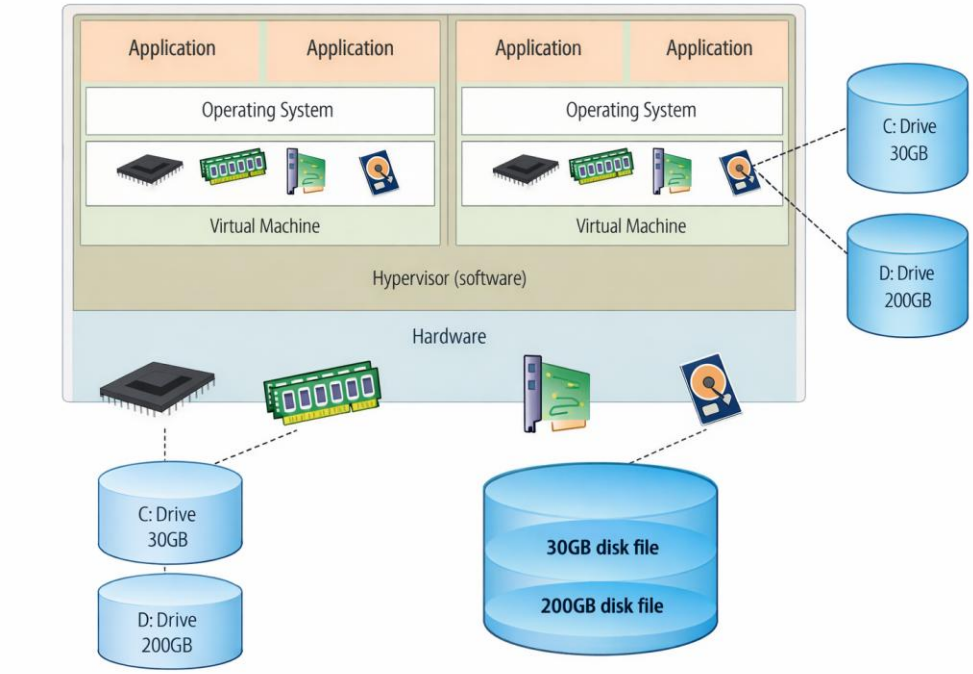
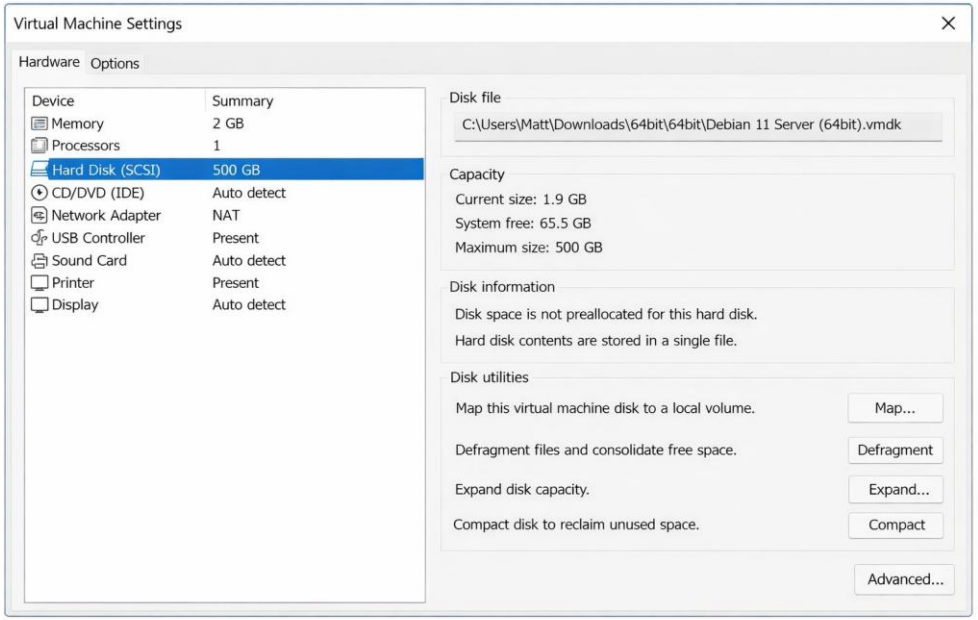


Figure 3.8 – Storage Abstraction

- Figure illustrates storage data flow.
- SAN, iSCSI, and NFS are abstracted.
- VMs remain storage-agnostic.



Storage Abstraction

Technology	Abstraction Level	How It Works	Typical Use Cases
SAN (Storage Area Network)	**Block-level**	Dedicated high-speed network (often Fibre Channel or iSCSI) that makes remote disks appear as local drives.	Databases, high-performance applications, enterprise storage consolidation.
iSCSI (Internet Small Computer System Interface)	**Block-level over IP**	Encapsulates SCSI commands into TCP/IP packets, allowing block storage access over standard Ethernet.	Cost-effective SAN alternative, virtualization, remote storage access.
NFS (Network File System)	**File-level**	Clients access files directly from a server using NFS protocol; operates at the application layer.	File sharing, home directories, collaborative environments, general-purpose storage.

How Virtual Machines Work

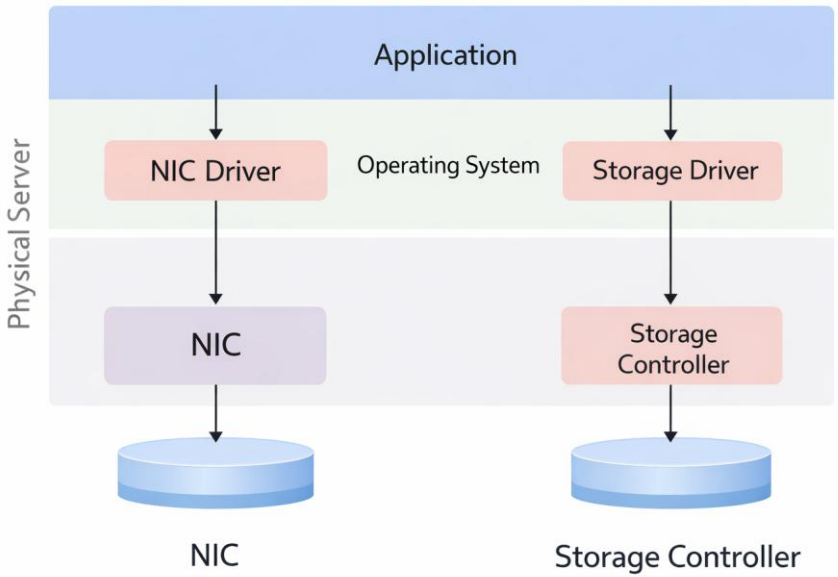
- Virtualization decouples OS from hardware.
- The hypervisor mediates all access.
- Guest OSs believe they run on real hardware.

Native OS Hardware Interaction

- Applications request hardware services.
- Requests pass through the OS.

Figure 3.9 – Native I/O Flow

- Figure shows how applications access hardware.
- The OS manages concurrency.
- Hardware access is controlled.



CPU Rings and Privilege Levels

- x86 architecture uses privilege rings.
- Ring 0 is most privileged.
- Ring 3 is least privileged.
- Ring 1 & 2 are used by device drivers (obsolete in recent hardware)

Role of Ring 0

- Operating system kernels traditionally run in ring 0.
- Privileged instructions execute here.
- This ensures system stability.

Hypervisor Interception

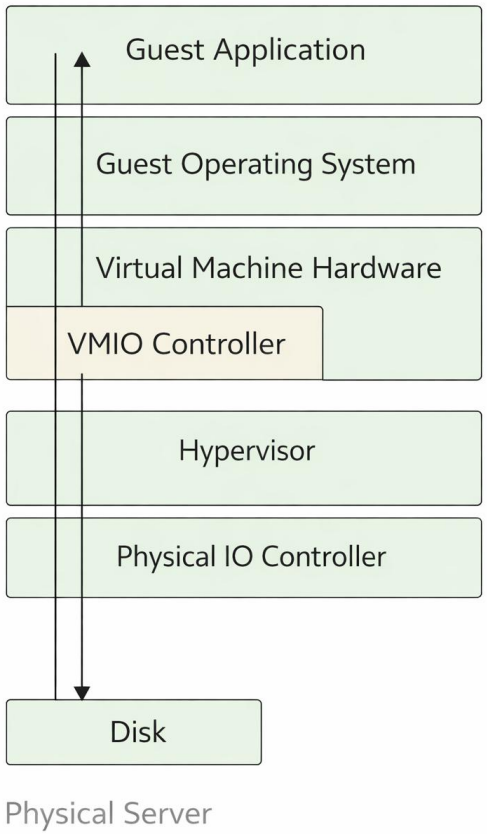
- Hypervisors run at the highest privilege level.
- Guest OSs believe they run in ring 0.
- Privileged instructions are trapped.

Isolation Enforcement

- Instruction trapping enforces isolation.
- Guests cannot affect each other.
- This satisfies Popek and Goldberg requirements.

Figure 3.10 – Virtualized I/O Flow

- Figure illustrates resource flow in virtualization.
- The hypervisor mediates requests.
- Performance remains efficient.



Working with Virtual Machines

- VMs exist as files and runtime instances.
- Administrators manage them remotely.
- Operations resemble physical servers.

VMs as Data Files

- VM files can be copied.
- They can be archived or backed up.
- File operations enable flexibility.

Understanding VM Clones

- Cloning creates a copy of an existing VM.
- It reduces provisioning time dramatically.
- Customization ensures uniqueness.

Provisioning Before Virtualization

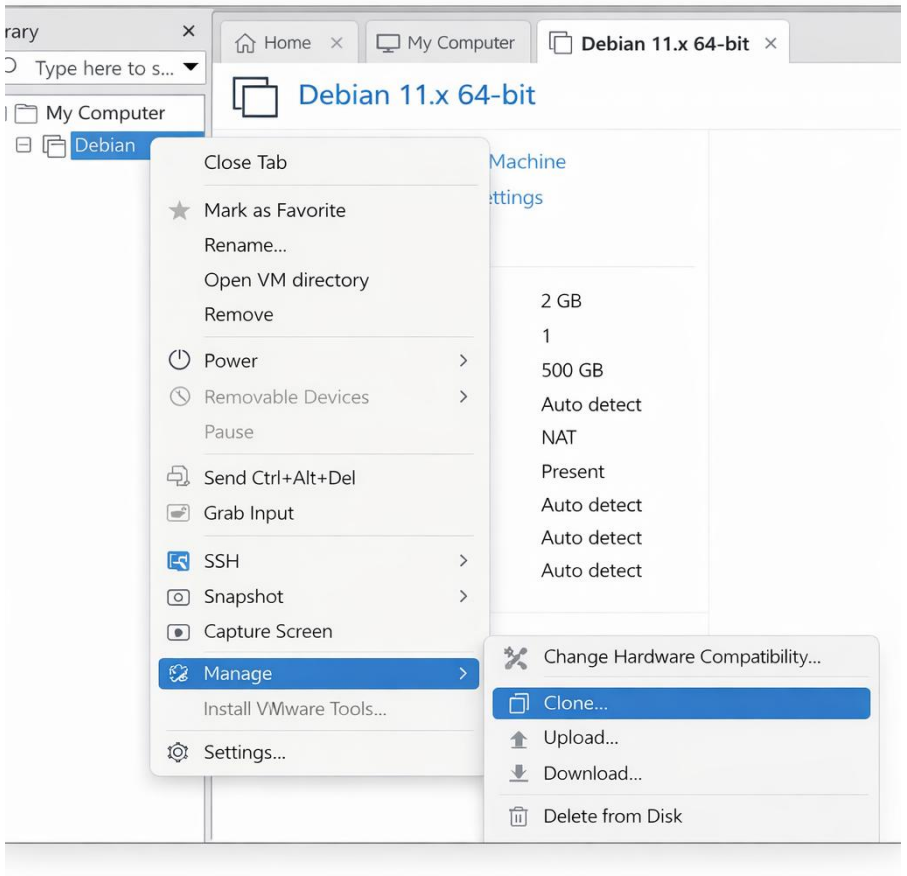
- Physical server provisioning took weeks.
- Hardware acquisition was slow.
- Costs were high.

Provisioning with Virtual Machines

- VMs can be provisioned in minutes.
- Cloning copies existing configurations.
- Operational efficiency improves.

Figure 3.11 – VM Cloning

- Figure illustrates VM cloning.
- Files are duplicated.
- Identity customization is applied.



Understanding Templates

- Templates are non-running VM images.
- They act as molds for new VMs.
- They ensure consistency.

- **Ex.: Book Publishing**
- **Image:** Manuscript (content written, not published)
- **Template:** Printing plate (used to print many copies)
- **VM:** Printed book (real, usable)
- Template = convenience, not requirement. (VMs can be created without temperate also)

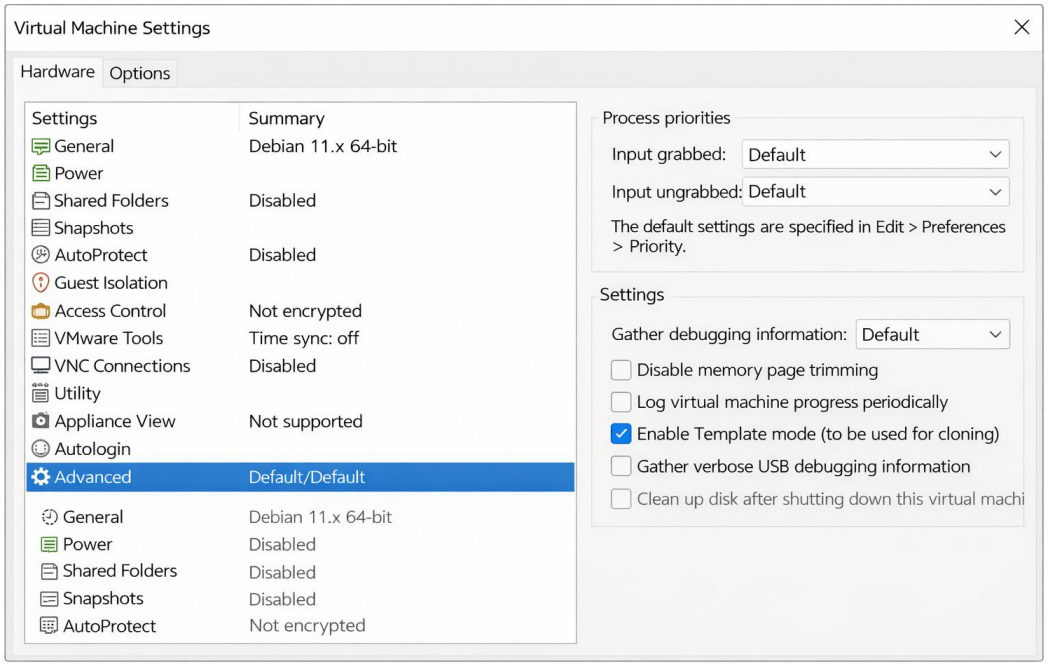
Template Lifecycle

- Templates cannot be run directly.
- They must be converted to VMs.
- Updates require reconversion.

From ↓ \ To →	VM	Template	Image
VM	NO	YES	YES
Template	YES	NO	YES
Image	YES	YES	NO

Figure 3.12 – Template Mode

- Figure shows template usage.
- Templates standardize deployments.
- Provisioning is accelerated.



Understanding Snapshots

- Snapshots capture VM state.
- They act as restore points.
- They support experimentation.

Snapshot Mechanics

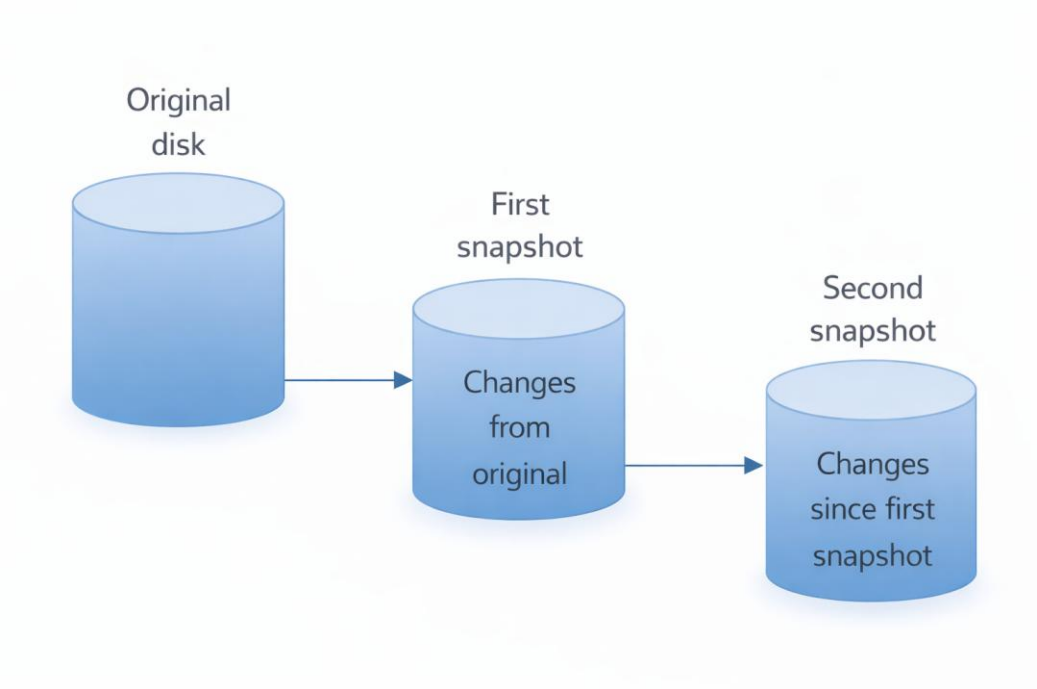
- Changes are written to delta disks.
- Original disk remains unchanged.
- Multiple snapshots form chains.

Snapshot Consolidation

- Consolidation merges delta disks.
- The VM returns to a single disk state.
- Performance is restored.

Figure 3.13 – Snapshot Operation

- Figure illustrates snapshot behavior.
- Snapshots are not backups.
- Overuse causes performance issues.



Understanding OVF

- Open Virtualization Format (OVF) is a vendor-neutral standard for packaging and distributing virtual machines.
- It enables VM portability.
- It is vendor-neutral.

OVF Packaging Formats

- OVF supports multiple files.
- Open Virtual Appliance (OVA) packages everything into one file (file format).
- Transport becomes simpler.

An OVA, or Open Virtual Appliance, is a software package that contains a pre-configured virtual machine image, along with any necessary software, configuration settings, and documentation. OVAs are typically used in virtualized environments, such as VMware, VirtualBox, or Hyper-V, to simplify the deployment and management of virtual machines.

Understanding Containers

- Containers are not virtual machines.
- They abstract at the OS level.
- They package applications efficiently.

Container Limitations

- All containers share the same OS kernel.
- Isolation is weaker than VMs.
- Security trade-offs exist.

Containers vs Virtual Machines

- VMs abstract hardware.
- Containers abstract the OS.
- Use cases differ.

Docker and Container Ecosystem

- Docker is the most popular container platform.
- It promotes standardization.
- Containers are rapidly evolving.

Virtual Appliances

- Virtual appliances bundle OS and application.
- They are preconfigured.
- They simplify deployment.

Evolution of Virtual Appliances

- Early appliances were locked down.
- Modern appliances allow customization.
- Vendor support has expanded.

Summary and Key Takeaways

- Virtual machines are core to virtualization.
- They enable consolidation, flexibility, and isolation.
- Advanced VM management transforms IT operations.