

UNIT 3

Cloud Virtualization Technology

(10 Hours) — Cloud Computing (CACS402)

Disclaimer: These notes were prepared as a study aid and may contain errors or omissions. Please cross-check important definitions, figures and product details (especially vendor services such as Google App Engine, Microsoft Azure and Aneka, which change often) against your prescribed textbook, class notes and official documentation before relying on them in an exam.

3.1 Overview of Virtualization Techniques

Virtualization is the creation of a virtual (rather than physical) version of a resource — a server, storage device, network or operating system — by adding a software layer between the resource and its users. It allows one physical machine to run several isolated **virtual machines (VMs)**, each with its own OS and applications.

Virtualization is the foundation of cloud computing because it enables resource pooling, multi-tenancy, rapid provisioning and elasticity.

Example 3.1 — Utilisation gain

A company has 10 physical servers, each only 10% utilised. Using virtualization, how many hosts could be needed if a host can safely run at 70% load?

Solution

Total useful load = $10 \times 10\% = 100\%$ of one server.

Hosts required $\approx 100 / 70 \approx 1.43$, so **2 hosts** (one extra for redundancy is common practice).

Result: 10 servers consolidated into about 2 → less hardware, power and space.

Main virtualization techniques

Technique	Description
Full virtualization	Hypervisor fully emulates hardware; unmodified guest OS runs unaware it is virtual (e.g., VMware ESXi with binary translation or hardware assist).
Para-virtualization	Guest OS is modified to make “hypercalls” to the hypervisor, giving better performance (e.g., Xen PV).
Hardware-assisted virtualization	CPU extensions (Intel VT-x, AMD-V) let the hypervisor run guests efficiently without binary translation (e.g., KVM).
OS-level (container) virtualization	Multiple isolated user-space instances share the host kernel (Docker, LXC).

3.2 Types of Virtualization

Type	What is virtualized	Example
Server virtualization	One physical server split into many VMs.	VMware vSphere, Hyper-V, KVM
Desktop virtualization (VDI)	User desktops run in the data centre and are accessed remotely.	Citrix Virtual Apps, Azure Virtual Desktop
Storage virtualization	Multiple physical storage devices pooled and presented as one logical store.	SAN virtualization, software-defined storage
Network virtualization	Physical network resources abstracted into virtual networks (VLANs, VPNs, SDN).	VMware NSX, OpenFlow-based SDN
Application virtualization	Applications run in isolated environment without full install.	Microsoft App-V
Data virtualization	Unified view of data from many sources without moving it.	Denodo
Memory / Hardware virtualization	Physical memory and hardware exposed as virtual resources.	Virtual memory, virtual RAM in VMs

3.3 Implementation Levels of Virtualization Structures

Virtualization can be implemented at different levels of the computer system stack. The lower the level, the more complete (but heavier) the virtualization.

Level	How it works	Example
1. Instruction Set Architecture (ISA) level	Emulates one ISA on another by translating instructions (interpretation or dynamic binary translation).	QEMU, Bochs

Level	How it works	Example
2. Hardware abstraction layer (HAL) level	Virtualizes CPU, memory and I/O directly on hardware using a hypervisor (VMM).	VMware, Xen, KVM
3. Operating system level	Isolated containers share the same OS kernel; very lightweight.	Docker, LXC, OpenVZ
4. Library (user-level API) level	Intercepts and re-maps library/API calls so apps written for one OS run on another.	Wine, Cygwin
5. Application level	Application runs inside a virtual machine that hides the OS/hardware.	Java Virtual Machine (JVM), .NET CLR

Note: Trade-off: higher levels (application/OS) give better performance and density but weaker isolation; lower levels (ISA/hardware) give strong isolation and flexibility but more overhead.

3.4 Virtualization Benefits

- **Server consolidation** — fewer physical machines, lower cost, power and cooling.
- **Better resource utilisation** — idle capacity is shared among VMs.
- **Isolation** — a failure or attack in one VM does not directly affect others.
- **Rapid provisioning** — new VMs are created from templates in minutes.
- **Flexibility and portability** — VMs can be copied, snapshotted and live-migrated between hosts.
- **High availability and disaster recovery** — restart VMs on another host after failure.
- **Testing and development** — safe sandbox environments.
- **Supports cloud elasticity and multi-tenancy.**

Drawbacks: hypervisor overhead, a single host failure affects many VMs, VM sprawl (uncontrolled growth), licensing complexity and new security risks (e.g., VM escape).

3.5 Server Virtualization

Server virtualization partitions one physical server into multiple isolated virtual servers using a hypervisor. Each virtual server has its own virtual CPU (vCPU), memory, disk and network interface.

Approaches

- **Hypervisor-based (full / para / hardware-assisted)** — VMs with separate OS kernels.
- **OS-level (containers)** — shared kernel, faster start-up, higher density.
- **Kernel-level** — the OS kernel itself acts as the hypervisor (KVM in Linux).

Key capabilities

- **Live migration** — move a running VM between hosts with almost no downtime (e.g., vMotion).
- **Snapshots and cloning** — save VM state and duplicate quickly.
- **Resource overcommit** — allocate more virtual resources than physically exist, relying on statistical sharing.
- **Load balancing and HA** — automatically restart or relocate VMs.

Example 3.2 — Container vs VM

Explain why a container starts faster than a VM.

Solution

A VM must boot a full guest operating system on virtual hardware (tens of seconds, GBs of disk). A container is just an isolated process group on the host kernel; there is no OS boot (often under a second, MBs of disk).

Feature	Virtual Machine	Container
Isolation	Strong (separate kernel)	Process-level (shared kernel)
Size	GBs	MBs
Start-up time	Seconds–minutes	Milliseconds–seconds
Density per host	Lower	Higher
Guest OS	Any OS	Same kernel as host

3.6 Hypervisor Management Software

A **hypervisor** or **Virtual Machine Monitor (VMM)** is the software that creates and runs virtual machines by allocating physical CPU, memory, storage and network to them and keeping them isolated.

Type 1 — Bare-metal

Type 2 — Hosted

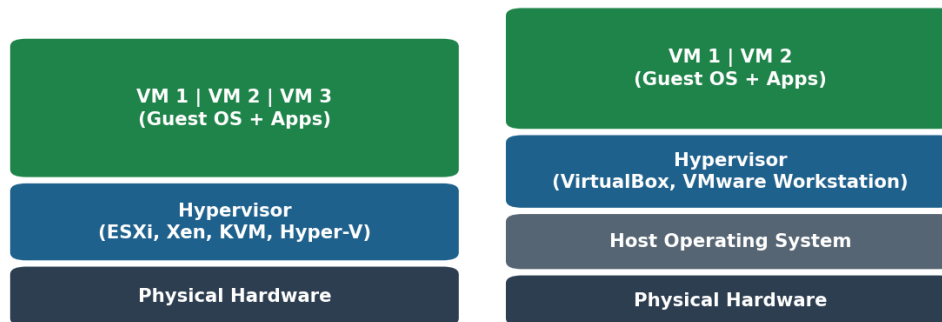


Figure 3.1 — Type 1 (bare-metal) and Type 2 (hosted) hypervisors

Feature	Type 1 (Bare-metal)	Type 2 (Hosted)
Runs on	Directly on hardware	On top of a host OS
Performance	High	Lower (extra OS layer)
Use	Data centres, cloud	Desktops, testing, learning
Examples	VMware ESXi, Microsoft Hyper-V, Xen, KVM	Oracle VirtualBox, VMware Workstation, Parallels

Hypervisor management software

Large clouds manage hundreds of hosts and thousands of VMs, so management software provides a central console and automation:

Product	Vendor / type	Main functions
VMware vCenter Server	VMware (manages ESXi hosts)	VM provisioning, vMotion, HA, DRS load balancing
Microsoft System Center VMM	Microsoft (Hyper-V)	Manage hosts, VMs and private clouds
Citrix XenCenter / XenServer	Citrix (Xen)	Pool management, live migration
libvirt / virt-manager / oVirt	Open source (KVM)	VM lifecycle and storage/network management
OpenStack (Nova)	Open-source cloud platform	Cloud orchestration across hypervisors

- **Typical functions:** creating, starting, pausing and deleting VMs; resource allocation; monitoring; templates and snapshots; live migration; access control; automated scheduling

and load balancing; alerting and reporting.

3.7 Virtual Infrastructure Requirements

Building a reliable virtual infrastructure requires careful planning of the following components:

Requirement	Details
Physical servers (hosts)	Multi-core CPUs with virtualization support (VT-x / AMD-V), large RAM, redundant power supplies.
Hypervisor and management software	Choose type 1 hypervisor and central management console with licences.
Shared storage	SAN / NAS / distributed storage so VMs can move between hosts; enough IOPS and capacity.
Networking	Adequate bandwidth, redundant NICs and switches, VLANs, separation of management, storage and VM traffic.
Security	Hardened hypervisor, isolation, firewalls, patching, role-based access control.
High availability and backup	Clustering, failover, snapshots, backup and disaster-recovery plans.
Monitoring and capacity planning	Track CPU, memory, disk and network to avoid overcommit problems and VM sprawl.
Skilled staff and licensing	Trained administrators; operating-system and hypervisor licence compliance.

Example 3.3 — Memory sizing

A host must run 20 VMs each needing 4 GB RAM. Keeping 20% of RAM reserved for the hypervisor and headroom, how much physical RAM is needed?

Solution

VM demand = $20 \times 4 \text{ GB} = 80 \text{ GB}$.

Usable RAM should be 80% of physical: $\text{Physical} = 80 / 0.80 = \mathbf{100 \text{ GB}}$. In practice choose 128 GB.

Key Takeaways — Unit 3

- ✓ Virtualization abstracts hardware so many isolated VMs share one physical machine; it is the base technology of the cloud.
- ✓ Techniques: full, para-virtualization, hardware-assisted and OS-level (containers).
- ✓ Types: server, desktop, storage, network, application and data virtualization.
- ✓ Implementation levels: ISA, hardware abstraction (HAL), OS, library, application.
- ✓ Type 1 hypervisors run on bare metal (ESXi, Xen, KVM, Hyper-V); Type 2 run on a host OS (VirtualBox).
- ✓ A virtual infrastructure needs capable hosts, shared storage, resilient networking, management software, security and HA/backup.

— *End of Unit 3* —

bcalink.com