Unit 2: Cloud Computing - Virtualization

Synthesized from Course Notes

November 15, 2025

1 Introduction to Virtualization

Virtualization serves as the foundational technology for cloud computing [file:1]. Its importance stems from its ability to make computing resources more flexible, scalable, and efficient, which is appealing to users and essential for the financial sustainability of Cloud Service Providers (CSPs) [file:1]. Virtualization involves the creation of a virtual version of something, such as a server, desktop, storage device, operating system, or network resources [file:1].

1.1 Physical vs. Virtual Resources

Understanding the distinction between physical and virtual resources is key to understanding virtualization [file:1]. Physical resources are the tangible hardware components of a system, while virtual resources are their digital, software-based representations [file:1].

Table 1: Comparison of Physical and Virtual Resources

Physical Resource	**Virtual Resource**
Refers to hardware equipment like processors, workstations, servers, networking devices, and storage devices [file:1].	Represents a digital, replicated version of a real physical resource [file:1].
Scalability is limited and often requires time-consuming and costly hardware upgrades [file:1].	Highly scalable; resources can be increased or decreased on demand, often in an automated and cost-effective manner [file:1].
Requires physical space for servers and other infrastructure, which can be a significant limiting factor [file:1].	No physical space is required for the virtual resource itself, reducing the physical footprint and allowing access from anywhere [file:1].
Security can be managed through physical controls like restricted access and surveillance [file:1].	Requires robust digital security measures such as encryption, firewalls, and access controls to protect against cyber threats [file:1].
Data recovery can be complex and time-consuming as it may require physical access to the hardware [file:1].	Data recovery can be faster and more efficient, often utilizing automated, cloud-based backup and replication solutions [file:1].
Flexibility is limited; reconfigurations or upgrades require physical changes to the hardware setup [file:1].	Highly flexible; resources can be easily scaled, modified, or reconfigured as needed, frequently through automated processes [file:1].

2 Virtualization Architecture

A Virtual Machine (VM) is a digital or virtualized replication of a physical machine [file:1]. It runs on a virtualization layer, which is responsible for creating the VM, allocating resources to it, and coordinating between multiple VMs running on the same hardware [file:1].

2.1 Traditional vs. Virtualized Architecture

In a traditional computing architecture, the operating system interacts directly with the hardware, and only a single instance of an OS can run [file:1]. This leads to tight coupling between hardware and software and often results in underutilization of resources [file:1].

In contrast, a virtualized architecture introduces a virtualization layer (the hypervisor) between the hardware and the operating systems [file:1]. This allows multiple VMs, each with its own guest OS, to run in parallel on the same physical hardware [file:1]. This leads to loose coupling and far more efficient resource utilization [file:1].

3 The Hypervisor (Virtual Machine Monitor)

The hypervisor, also known as the Virtual Machine Monitor (VMM), is the software program that creates, runs, and manages virtual machines [file:1]. It allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing [file:1].

There are three main types of hypervisors:

- **Type 1 / Bare-Metal Hypervisor:** This hypervisor runs directly on the host's hardware to control the hardware and to manage guest operating systems [file:1]. This is the most common type in server virtualization, offering high performance and isolation [file:1]. Examples include VMware ESXi and Xen [file:1].
- **Type 2 / Hosted Hypervisor:** This hypervisor runs on top of a conventional host operating system, just like any other software application [file:1]. It has overhead because of the host OS, leading to lower performance compared to Type 1 [file:1]. It is often used for desktop virtualization [file:1]. Examples include VMware Workstation and Parallels Desktop [file:1].
- **Embedded Hypervisor:** In this case, the hypervisor is embedded into the hardware itself, along with the host OS [file:1]. This provides a high level of security and is often used in control-automated devices and manufacturing [file:1]. An example is VMware vSphere [file:1].

4 Levels of Virtualization

Virtualization can be implemented at different levels, primarily distinguished by how the guest OS interacts with the underlying hardware [file:1]. The main models are Full Virtualization and Paravirtualization [file:1].

4.1 Full Virtualization

In a full virtualization environment, the virtual machine is completely unaware that it is virtualized [file:1]. It believes it has direct control over the hardware [file:1]. The hypervisor intercepts and translates privileged instructions from the VM that need to be executed by the physical hardware [file:1]. This process is called **Trap and Emulate** [file:1].

Execution Steps:

1. The VM, unaware of the hypervisor, attempts to execute a privileged instruction as if it were running on physical hardware [file:1].
2. The hypervisor traps this instruction before it reaches the hardware [file:1].
3. The hypervisor inspects the instruction and, if it is safe, emulates the behavior that the hardware would have produced [file:1].
4. The result is returned to the VM, which assumes the instruction was executed successfully by the hardware [file:1].

4.2 Paravirtualization

In a paravirtualization environment, the virtual machine is aware that it is virtualized and cannot execute privileged instructions directly [file:1]. Instead of trapping instructions, the guest OS is modified to make special calls to the hypervisor to request the execution of these instructions [file:1]. These requests are known as **hypercalls** [file:1].

Execution Steps:

1. The VM's guest OS sends a hypercall to the hypervisor, requesting the execution of a privileged operation [file:1].
2. The hypervisor receives the request and executes the operation on behalf of the VM, provided it does not compromise the stability of the underlying system [file:1].

5 Types of Virtualization

Beyond server virtualization, several other types exist, focusing on specific resources like applications and desktops [file:1].

5.1 Application Server Virtualization

This involves hosting multiple virtual instances of an application server on a single physical server [file:1]. This model uses tools like virtual load balancers and firewalls to manage traffic [file:1]. Key features include:

- **Advanced Load Balancing:** Manages traffic across not just homogeneous applications, but heterogeneous ones as well [file:1].
- **Dynamic Scaling and Migration:** Allows for the dynamic scaling of resources, service migration, and server consolidation to optimize workloads [file:1].

5.2 Application Virtualization (Thin Client Technology)

In this model, applications are made independent of the local device's operating system [file:1]. The application is executed remotely, and only its user interface is streamed to the client device, hence the term "Thin Client" [file:1]. This means the application is not installed on the local machine [file:1]. This includes:

- **Remote Desktop / Terminal Services:** The entire desktop environment of a remote server is streamed to the client [file:1]. Example: Citrix Virtual App [file:1].
- **Desktop Virtualization (Sandboxing):** A full desktop environment is run in a secure, isolated "sandbox" in the cloud, protecting the underlying system from changes [file:1]. Example: Citrix Virtual Desktop [file:1].
- **Application Streaming:** Parts of an application are delivered to the client as needed, reducing local storage and processing requirements [file:1]. Example: VMware ThinApp [file:1].