Cloud Computing

MUCPD – DSIC - UPV

Virtualization

Overview

- Introduction to virtualization
- History and evolution
- Types of virtualization
    - Hardware Virtualization
    - Software virtualization
    - OS-Level virtualization
    - Network virtualization
- Hypervisors: VM vs Containers
- Virtualization and the Cloud
- Pros and Cons
- Future trends

Introduction

- **Definition**

  Process of creating a software-based representation of something, rather than a physical one

- **Key Concepts**
  - **Abstraction**
    - Separation of resources and services from the physical delivery environment
    - Abstracts the hardware layer
      - Allows multiple virtual environments to run on a single physical system
  - **Resource Sharing**
    - All virtual environments share available physical resources
      - Potentially maximizes resource utilization
  - **Isolation**
    - Each virtual environment operates independently
      - A virtual environment cannot affect another virtual environment

LINUX
BSD
WINDOWS
VMS

Key Benefits

- **Improved Resource Utilization**
  - Sharing physical resource increases their utilization
    - From 15% utilization to 70-80% utilization
- **Cost Reduction**
  - Hardware reduction made possible, which reduces ***CapEx*** & ***OpEx***
- **Simplified Management**
  - Virtualized resources are managed centrally
    - Easier to monitor performance, set updates, ...
- **Enhanced disaster recovery**
  - Virtual machines can be backed up fully
    - Recovery on different physical hardware
- **Environmental impact**

History and Evolution

- **1960s: IBM Mainframes and Time-Sharing**
  - E.g. CP-40, CP-67. Goal:
    - Maximize utilization of expensive hardware
- **1970s-1980s: Decline**
  - More accessible computing resources (PCs, Workstations).
    - Less focus on mainframes
- **1990s: Server Sprawl. Renewed interest**
  - Organizations accumulate servers (sprawl)
    - Virtualization to consolidate servers and increase efficiency
- **2000s: X86 virtualization**
  - VMWare & VirtualPC
    - Virtualization applied to widely used hardware ➔ Game changer
- **2010-present: Cloud computing & Containers**
  - Orchestration tools,...

Early Virtualization-IBM Mainframes

- **CP-40 and CP-67 Projects**
  - Introduced the concept of Virtual Machines:
    - Multiple Operating Systems could run on a single physical mainframe
- **The idea of virtual machines**
  - A Machine within a Machine
    - Let users and their applications run in multiple ISOLATED environments
      - Sharing the physical resources
- **Time-sharing systems**
  - Allow multiple users simultaneous access to computing resources
    - Maximaze utilization.
    - Improve service being delivered to more users
- **Key: ISOLATION**
  - In various degrees

Decline and Resurgence

- **1970s-1980s Decline**
  - Rise of Personal Computers and Workstations
    - Computing power accessible to individuals with no need to access mainframes
      - Reduced need for shared resources
- **1990s: Resurgence**
  - Multiple servers being deployed
    - Each dedicated to a specific task/app (why?)
      - Thus... underutilized hardware
  - Inefficient hardware utilization
    - Increased cost to operate many physical servers
      - IT departments face increased costs: diversity of hardware and software
  - Need for consolidation
    - Consolidation of servers through virtualization techniques:
      - simplify management.
      - Reduce costs

Virtualizing the X86 architecture

- **2000's breakthrough**
  - Virtualization on standard hardware (The PC architecture)
    - Prior only on proprietary mainframes. Not useful for sprawl of servers
      - The PC architecture was (is) the basis for most PCs and servers
  - Virtual PC (Microsoft)
    - Early fully software-emulated architecture
      - Very inefficient, as many resources employed to emulate a hardware machine
        - Very portable, as it was, essentially, a process interpreting machine code
  - VMWare contribution
    - 1999: VMWare workstation. For workstations/PCs/Servers.
      - 2001: VMWare ESX Server. Type 1 Hypervisor for data centers.
- **Impact on Data Centers**
  - Allowed virtualization of server environments (and consolidation)
  - Groundwork for the development of cloud computing services
    - Efficiency needed to manage large amounts of VMs

Emergence of Cloud Computing

- **2006 Onwards**
  - AWS offered IT infrastructure services
- **Reliance on virtualization**
  - Virtualization allowed a scalable offering of on-demand computing resources as services
  - Key to allow providers to dynamically offer resources on demand
- **Shift in IT Paradigm**
  - Organizations move from OWNING physical devices to CONSUMING virtualized ones
  - Business can focus on their core
    - Outsource infrastructure management to third parties
      - Increased flexibility and reduced costs

Container Technologies

- **2013 Onwards**
  - Introduction of Docker
    - Simplified container management and deployment on a single node
- **Advantages over hardware virtualization**
  - Lightweight and fast start up
    - No boot time. Process launching
  - Improved application deployment
    - Images enabled consistency and portability
- **Impact on devops & microservices**
  - Facilitates CI/CD pipelines
  - Facilitates building microservice-based architectures for services.

Types of virtualization

- **Hardware Virtualization**
  - Creates virtual version of physical devices
    - Enables multiple OSs to run un a single physical machine within their own VM
- **Software Virtualization**
  - Focuses on virtualizing applications/software environments
    - E.g. Desktop virtualization
- **OS level virtualization**
  - Containers
  - Multiple isolated user-space instances running on a single OS kernel
- **Network Virtualization**
  - Abstracts physical network resources
    - Creates multiple virtual networks over a single physical network
  - Includes technologies like SDN (Software Defined Networks).

Hardware Virtualization

- **Full Virtualization**
  - Hypervisor provides a complete simulation of the underlying hardware
  - Guest OS runs unmodified.
    - Unaware it is running virtualized
- **Paravirtualization**
  - Hypervisor does not fully simulate the hardware
    - Guest OS is modified (lightly) to communicate with hypervisor directly
      - Performance gains
- **Use Cases**
  - Server consolidation
    - Multiple loads under same physical server
  - Testing & Development Environments
    - Quickly spin-up/down multiple VMs for testing
      - Potentially on different environments

OS level virtualization

- **Containers**
  - Isolated user spaces
  - Share Host OS Kernel
- **Advantages**
  - Lightweight
  - Fast
  - Scalable
- **Technologies**
  - Docker
  - Podman
  - Containerd
  - LXC

Network Virtualization

- **Various elements**
  - Virtual switches and routers
    - Implemented within the kernel
  - Virtual interfaces
    - Implemented within the kernel
  - VLANs
    - Part of the ethernet protocol
  - Software Defined Networks
    - Separates control-plane from data-plane
      - Improves network programmability
- **Benefits**
  - Improved network management
  - Enhanced security
    - Sets up specific communication paths (0-trust networking on top)
  - Flexibility

Hypervisors

A hypervisor, also known as a Virtual Machine Monitor (VMM), is software that creates and runs virtual machines by abstracting the underlying hardware resources

- **Type1 (Bare Metal)**
  - Runs directly on the host's hardware
    - ESXi, Hyper-V, XenServer,...
- **Type 2 (Hosted)**
  - On top of an OS
    - E.g. VMWare Workstation, VirtualBox, Parallels.
- **Responsibilities**
  - Resource allocation
  - Isolation
  - Management

Type 1 vs Type 2 Hypervisors

Type 1

- **Architecture**:
  - Install directly on the physical hardware.
  - hypervisor manages all hardware resources and provides virtual machines with direct access to physical hardware.
- **Performance**:
  - Offers better performance and efficiency since there's no underlying operating system.
- **Use Cases:**
  - Ideal for enterprise data centers, cloud providers, and environments where performance and scalability are critical.
- **Examples**:
  - VMware ESXi, Microsoft Hyper-V Server, Citrix XenServer.

Type 2

- **Architecture:**
  - Install on top of an existing operating system.
  - Relies on the host OS for device support and resource management.
- **Performance:**
  - May have lower performance due to additional overhead from the host OS.
- **Use Cases:**
  - Suitable for individual users, developers, and testing environments where ease of setup and flexibility are important.
- **Examples:**
  - VMware Workstation, Oracle VirtualBox, Parallels Desktop.

Types of Hypervisors

Characteristic	Type 1 (Bare-metal Hypervisor)	Type 2 (Hosted Hypervisor)
Architecture	Runs directly on hardware	Runs on a host OS
Performance	High performance	Lower performance
Use Cases	Data centers, cloud environments	Development, testing environments
Host OS Requirement	No	Yes
Example Hypervisors	VMware ESXi, Microsoft Hyper-V	VirtualBox, VMware Workstation

Virtual Machine

Software emulation of a Physical Computer

- **Components:**
  - Virtual Hardware
    - Virtual CPU, memory storage
  - Guest OS instance
    - Can be different from the host OS (for Type 2)
  - Applications
    - On top of the guest OS
- **Properties**
  - Isolation ➔ Security, stability
  - Flexibility
    - Different Guest OS
      - Legacy application support

VMs vs Containers

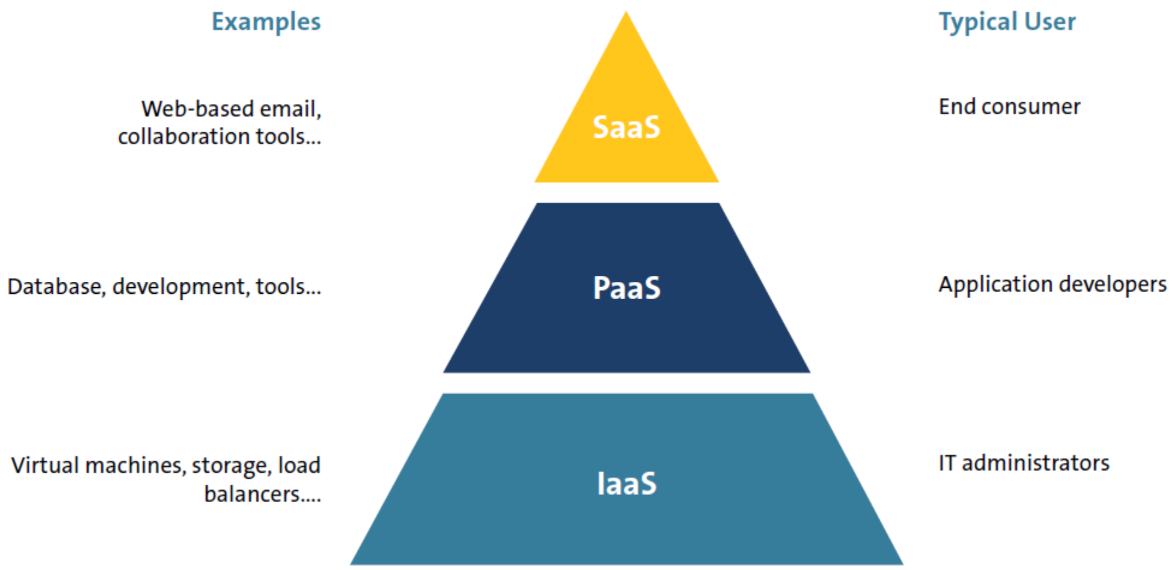
- **Structural:**
  - **VM**: Hypervisor-based, whole OS included
  - **Container**: Shares Kernel
- **Resource Usage**
  - **VM**: Higher overhed
  - **Container**: Lightweight/process-like overhead
- **Use Cases**
  - **VM**: Multi-OS environments. STRONG isolation
  - **Container**: Microservices, rapid deployment.

Virtualization:
the backbone of the cloud

- **Abstraction of resources**
  - Managed by software, thus flexible.
- **Foundation for Cloud Services**
  - Much more difficult without it
- **Enables Scalability**
  - Rapid provisioning.
  - Allows Service providers to adapt to demand
- **Supports Multi-Tenancy**
  - Through Isolation

Cloud Service Models rehash

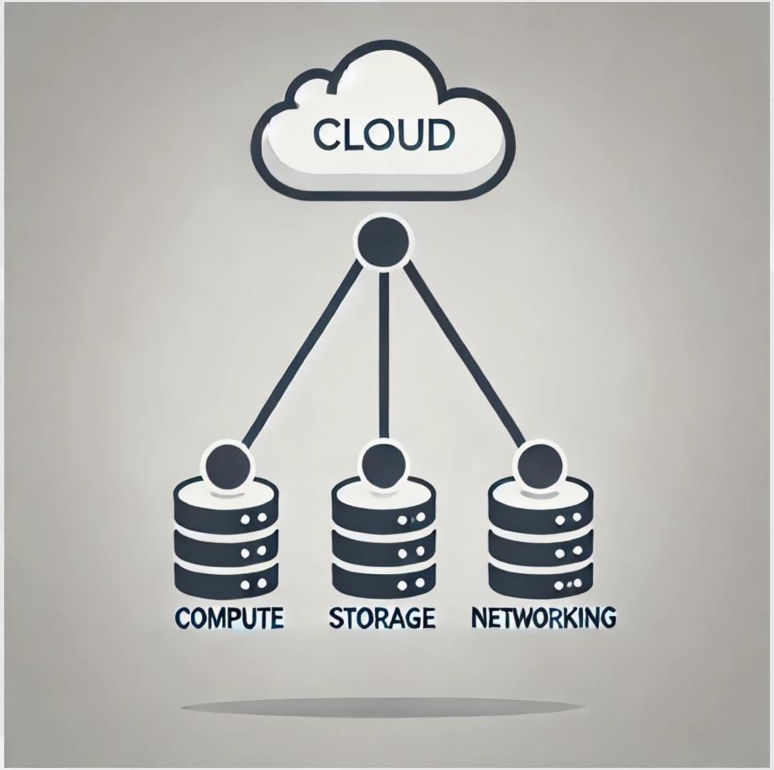
Figure 1: Cloud Pyramid[22]



IaaS impact

- **All resources are virtualized**
- **No virtualization, no IaaS**
- **User controls OS & applications**
  - Full stack
- **Examples**
  - Amazon EC2
  - Microsoft Azure Virtual Machines
  - Google Compute Engine
  - OVH
  - Aire Networks
  - Private CPDs managed with OpenStack
  - ...

IaaS impact



PaaS impact

- **Provides Developer's support**
- **Abstracts OS management**
- **User controls applications**
- **Examples**
  - Heroku
  - Axebow/Kumori
  - Microsoft Azure App Service
  - Google App Engine
  - ...

SaaS impact

- **Delivers an application in the form of a service**
- **Abstracts managing application**
- **Application-specific user interactions**
    - User does not manage the application nor its deployment
- **Examples**
    - Salesforce
    - Office365
    - Google workspace
    - Neflix
    - ... (too numerous to list many)

Case Study: AWS

- **Virtualization technologies**
  - Xen
  - Amazon Nitro System
- **Specific virtualized resources**
  - EC2 instances (VMs)
  - Elastic Block Store (Storage)
  - Virtual Private Cloud/Virtual Private Networking
  - Managed Kubernetes: EKS
- **Impact**
  - Scalable on-demand computing
  - Hybrid Cloud Capabilities

Case Study: Azure

- **Virtualization technologies**
  - Hyper-V
- **Specific virtualized resources**
  - Azure Virtual Machines
  - Azure Disk Storage
  - Azure Virtual Networks
  - Managed Kubernetes: AKS
- **Impact**
  - Integration with the Microsoft ecosystem
  - Hybrid Cloud Capabilities

Case Study: Google GCP

- **Virtualization technologies**
  - KVM hypervisor
- **Specific virtualized resources**
  - Compute Engine (VMs)
  - Google Cloud Storage (Disks)
  - VPC, VPN (Networking)
- **Impact**
  - Integration with the Microsoft ecosystem
  - Hybrid Cloud Capabilities

Summary: Virt pros for cloud

- **Resource Optimization**
  - Improved utilization
  - Dynamic allocation
- **Scalability**
  - On-demand provisioning
  - Rapid deployment
- **Isolation**
  - Fault isolation
  - Enhanced security

Summary: Virt cons for cloud

- **Performance overhead**
  - Latency
  - Resource contention
    - When overcommiting
- **Security risks**
  - Hypervisor vulnerabilities
    - Isolation breakdowns
- **Licensing & Compliance**
  - Conditions imposed by some vendors make it complex
  - Regulatory compliance may require specific controls
    - Not available in the virtualized environment
- **Specialized knowledge**
  - Introduces its own complexity
  - Enhanced security

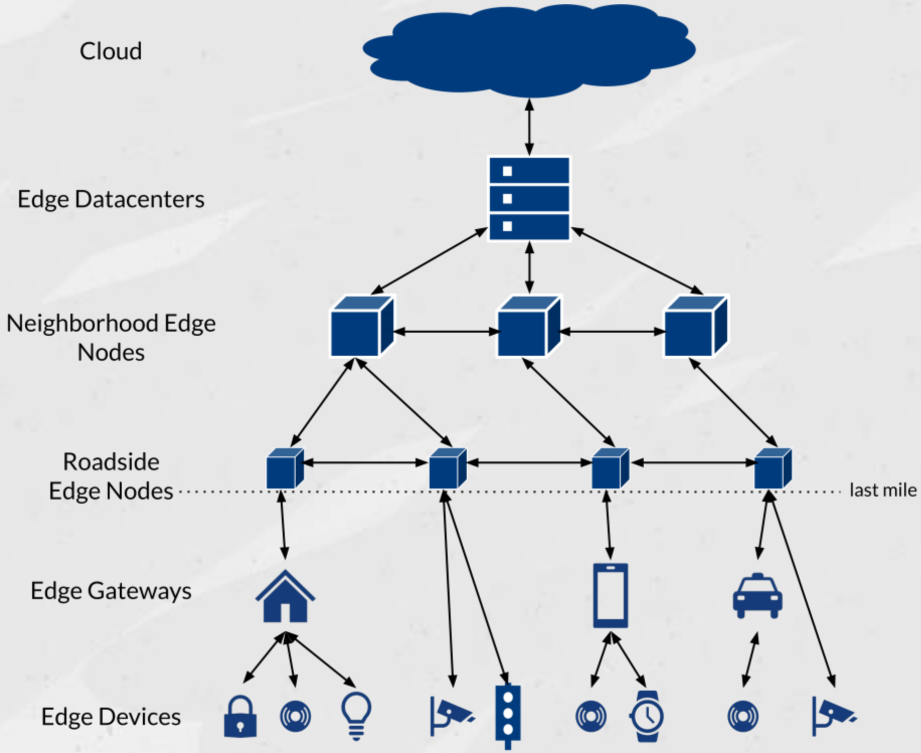
Mitigating challenges

- **Performance optimization**
  - Resource Monitoring
  - Hardware acceleration (VT-x/AMD-V)
    - When overcommiting
- **Security measures**
  - Security best practices
    - Frequent patching
    - Network segmentation
- **Licensing & Compliance**
  - Negotiations and compliance audits
  - Clear documentation
- **Specialized knowledge**
  - Automation tools
  - Training

Future trends

- **Edge computing**
  - Small devices close to the data
- **Serverless architectures**
  - Developer writes functions
    - Does not worry about servers/ state
    - Need fast function activation and setup
- **Hyperconvergence**
  - Integration of compute, storage and networking into a single system
- **AI (of course)**
  - Motivation for innovations to efficiently support AI tasks.
    - E.g., GPU virtualization

Edge computing



Virtualization and Edge Computing

- **Processing data at the edge**
  - Small devices close to the data
- **Lightweight virtualization solutions**
  - Containers for small devices
  - MicroVMs
- **Use Cases**
  - IoT Devices
    - E.g. Autonomous vehicles, industrial sensors
  - Real time analytics
    - For critical applications, needing responsiveness

Serverless

- **Function as a Service (FaaS)**
  - Write and deploy code without caring how it is deployed
  - FaaS in charge of activating code and linking it to whatever services it needs
- **Event-driven execution**
  - Launch when certain events happen
    - E.g., arrival of API call
  - Dormant (not consuming resources) otherwise
- **Benefits**
  - Reduced operational complexity for developer
  - Cost efficiency
  - Automatic scaling

AI impact on virtualization

- **Need to optimize Virtualization for AI workloads**
- **GPU virtualization**
  - Same needs for sharing the resource
  - Same needs to keep an isolated environment
- **Intelligent resource management**
  - Optimize resource allocation
  - Cost efficiency
  - Automatic scaling

Best practices

- **Regular updates & maintenance**
  - Eliminate vulnerabilities
- **Resource monitoring and Optimization**
  - Prevent resource contention
    - Including the network
- **Security policies and procedures**
  - Access control, network segmentations, etc...
- **Disaster Recovery and Planning**
  - Backups, etc...

Key takeaways

- No Cloud Computing without virtualization
- Different types of virtualization
- Need to understand pros and cons for proper decissions
- Technology is evolving