UNIT-III

Virtualization

Definition

Virtualization is the process of creating virtual versions of physical resources such as servers, storage devices, operating systems, and networks.

It allows multiple virtual machines (VMs) to run on a single physical machine.

Types of Virtualization

1. Server Virtualization

Multiple virtual servers run on one physical server.

Advantages

- Better hardware utilization
- Reduced cost
- Improved performance
- Easy management

2. Storage Virtualization

Combines multiple storage devices into a single storage system.

Benefits

- Efficient storage management
- Faster backup and recovery
- Increased flexibility

3. Network Virtualization

Creates virtual networks independent of physical hardware.

Applications

- Virtual LANs (VLANs)
- Software Defined Networking (SDN)

4. Desktop Virtualization

Desktop environment is stored on a central server and accessed remotely.

Advantages

- Remote access
- Centralized security
- Easy maintenance

5. Application Virtualization

Applications run without being directly installed on the operating system.

Virtual Machine (VM)

A Virtual Machine is a software-based computer that runs an operating system and applications like a physical computer.

Components of VM

- Virtual CPU
- Virtual Memory
- Virtual Storage
- Virtual Network Interface

Hypervisor

A Hypervisor is software that creates and manages virtual machines.

Types of Hypervisors

Type 1 Hypervisor (Bare Metal)

Runs directly on hardware.

Examples

- VMware ESXi
- Microsoft Hyper-V
- Xen

Type 2 Hypervisor (Hosted)

Runs on top of an operating system.

Examples

- Oracle Virtual Box
- VMware Workstation

Advantages of Virtualization

- Better resource utilization
- Reduced hardware cost
- Energy efficiency
- Easy backup and recovery
- Improved disaster recovery
- Scalability

Disadvantages of Virtualization

- Initial setup cost
- Performance overhead
- Security concerns
- Requires skilled management

3. Programming Models for Cloud Computing

Programming models help developers create distributed applications that can run efficiently on cloud platforms.

Need for Programming Models

- Handle large-scale data
- Improve parallel processing
- Simplify distributed computing
- Increase scalability and fault tolerance

4. Map Reduce

Definition

Map Reduce is a programming model developed by Google for processing large amounts of data in parallel across distributed systems.

It divides the task into two major phases:

1. Map Phase
2. Reduce Phase

Working of Map Reduce

Step 1: Input Splitting

Large data is divided into smaller chunks.

Step 2: Map Function

Processes input data and generates key-value pairs.

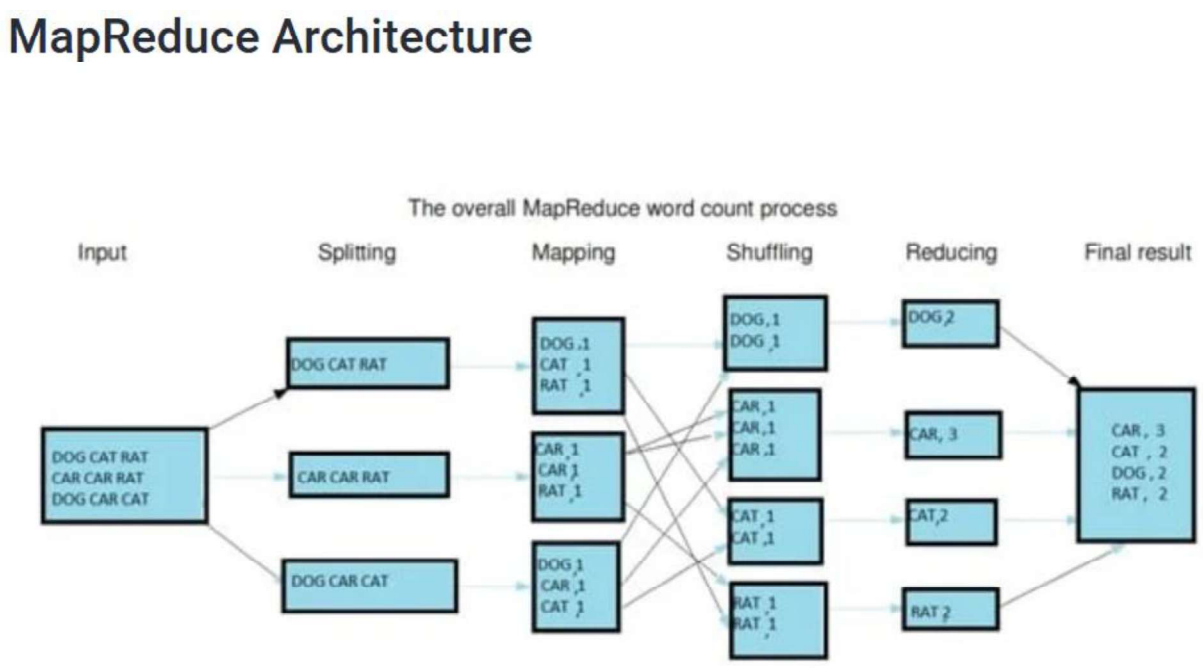
Step 3: Shuffle and Sort

Groups all similar keys together.

Step 4: Reduce Function

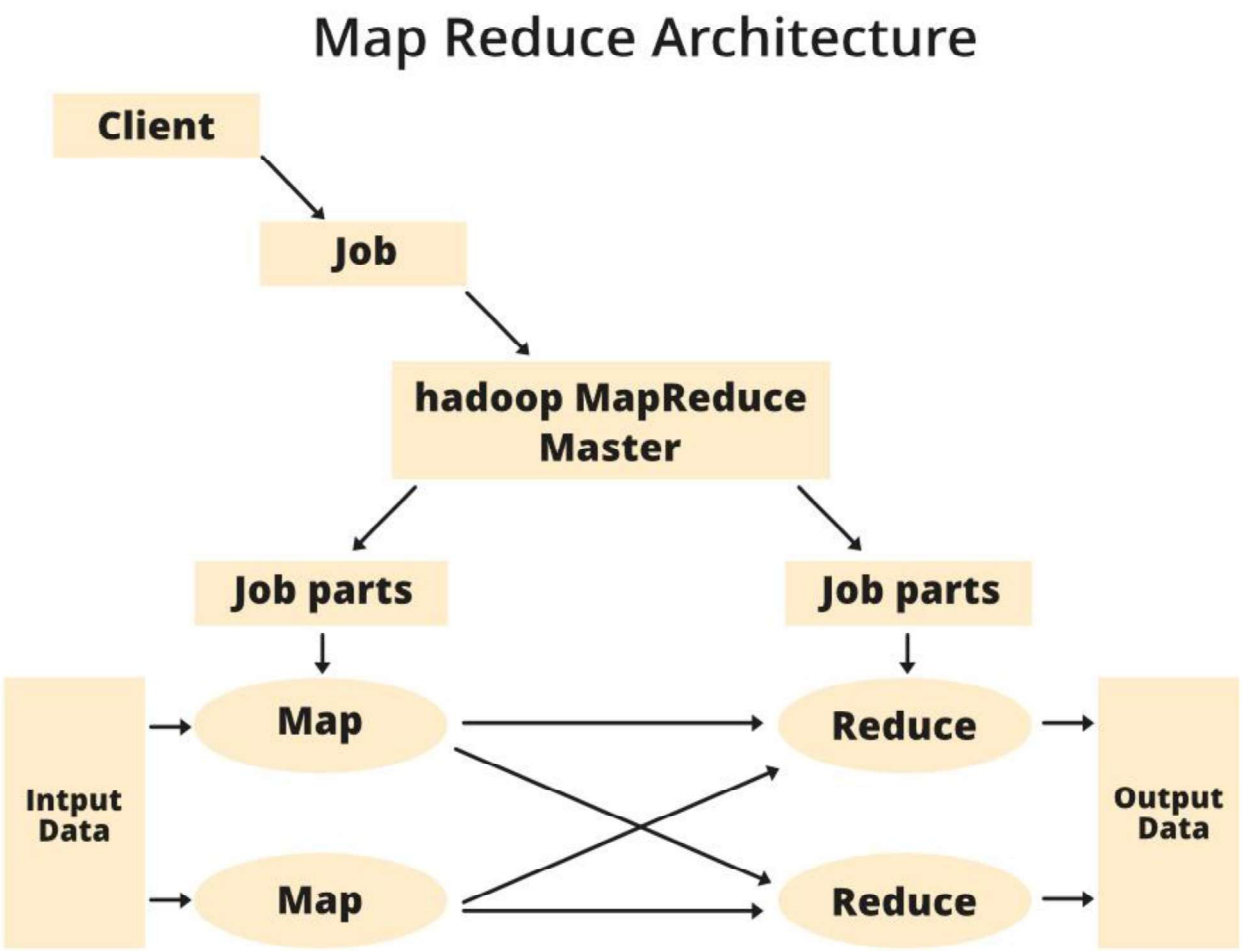
Processes grouped data and produces final output.

Map Reduce Architecture



Components

- Client
- Job Tracker
- Task Tracker
- Distributed File System



Example of Map Reduce

Word Count Problem

Input

"Cloud computing is powerful and cloud services are scalable"

Map Output

(cloud,1) (computing,1) (is,1) (powerful,1) (cloud,1) (services,1) (are,1) (scalable,1)

Reduce Output

(cloud,2) (computing,1) (is,1) (powerful,1) (services,1) (are,1) (scalable,1)

Advantages of Map Reduce

- Handles huge data sets
- Automatic parallelization
- Fault tolerance
- Scalability
- Cost effective

Disadvantages of Map Reduce

- Complex programming for some tasks
- High latency
- Not suitable for real-time processing

Applications of Map Reduce

- Search engines
- Data analytics
- Log processing
- Machine learning
- Social media analysis

5. Cloud Haskell

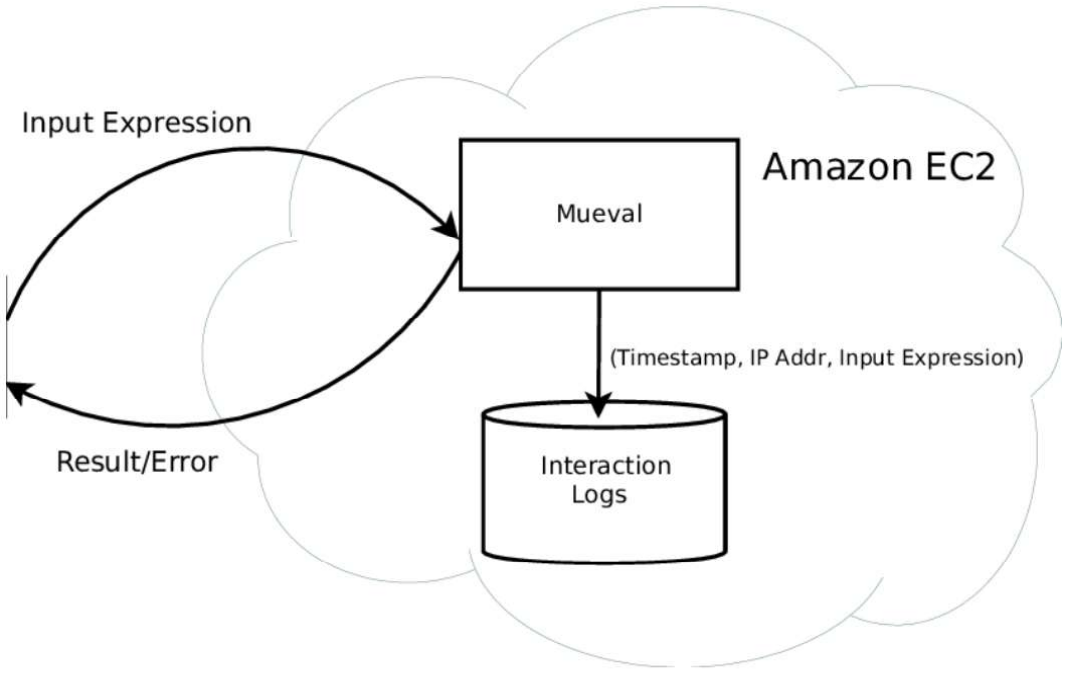
Definition

Cloud Haskell is a distributed programming platform based on the Haskell programming language. It provides support for building scalable and fault-tolerant distributed cloud applications.

Features of Cloud Haskell

- Message passing communication
- Distributed processing
- Fault tolerance
- Scalability
- Strong type safety

Architecture of Cloud Haskell



Architectural diagram of our Haskell code evaluation infrastructure, which was hosted on Amazon Elastic Compute Cloud (EC2)

Main Components

- Nodes
- Processes
- Channels
- Message Passing

Working Principle

1. Multiple nodes communicate over the network.
2. Processes exchange messages.
3. Tasks are distributed among nodes.
4. Fault tolerance mechanisms recover failed processes.

Advantages of Cloud Haskell

- Reliable distributed computing
- Strong error handling

- Efficient concurrency
- Easy scalability

Disadvantages

- Complex syntax for beginners
- Limited industry adoption
- Requires understanding of functional programming

Applications of Cloud Haskell

- Distributed systems
- Cloud applications
- Parallel data processing
- Real-time analytics

6. Software Development in Cloud

Definition

Software development in cloud refers to designing, developing, testing, deploying, and maintaining applications using cloud platforms.

Cloud-Based Software Development Lifecycle

1. Requirement Analysis

Understanding project requirements.

2. Design

Creating architecture and application design.

3. Development

Writing application code.

4. Testing

Testing software for bugs and performance.

5. Deployment

Deploying software on cloud platforms.

6. Maintenance

Updating and monitoring the software.

Cloud Development Models

1. SaaS Development

Software is delivered over the Internet.

Example

- Google Docs
- Gmail

2. PaaS Development

Developers use cloud platforms to build applications.

Example

- Google App Engine
- Microsoft Azure

3. IaaS Development

Infrastructure resources are rented from cloud providers.

Example

- Amazon EC2
- Microsoft Azure VM

Tools Used in Cloud Software Development

- Docker
- Kubernetes
- GitHub
- Jenkins
- Visual Studio Code
- AWS
- Microsoft Azure
- Google Cloud Platform

Advantages of Cloud Software Development

- Faster development
- Reduced infrastructure cost
- Easy collaboration
- Automatic updates
- Global accessibility
- Better scalability

Challenges

- Security issues
- Internet dependency
- Data privacy concerns
- Vendor lock-in

7. Comparison: Traditional vs Cloud Software Development

Traditional Development	Cloud Development
Requires physical hardware	Uses virtual resources
High setup cost	Pay-as-you-use model
Limited scalability	Highly scalable
Manual updates	Automatic updates
Difficult remote collaboration	Easy online collaboration