CS802 (B) Cloud Computing

Unit 1

What is Cloud Computing?

Cloud computing is the **delivery of computing services** over the internet ("the cloud")—including **servers, storage, databases, networking, software, analytics, and intelligence**—to offer faster innovation, flexible resources, and economies of scale.

Instead of owning their own computing infrastructure or data centers, companies can **rent access** to anything from applications to storage from a cloud service provider.

□ Key Characteristics

- **On-demand self-service:** Resources like computing power and storage can be provisioned as needed.
- **Broad network access:** Services are available over the network and accessed through standard mechanisms (e.g., browsers, APIs).
- **Resource pooling:** Multiple users share a pool of computing resources.
- **Rapid elasticity:** Scale up or down based on demand.
- **Measured service:** Usage is monitored and billed accordingly.

☁ Types of Cloud Deployment Models

1. **Public Cloud** – Services are offered over the public internet (e.g., AWS, Azure, Google Cloud).
2. **Private Cloud** – Services are maintained on a private network.
3. **Hybrid Cloud** – Combines public and private cloud elements.

□ Service Models in Cloud

1. **IaaS (Infrastructure as a Service):**
   Provides virtualized computing resources via the internet (e.g., Amazon EC2, Google Compute Engine).
2. **PaaS (Platform as a Service):**
   Provides a platform to develop, run, and manage applications (e.g., Google App Engine, Microsoft Azure App Services).

3. **SaaS (Software as a Service):**
   Provides software applications over the internet (e.g., Gmail, Dropbox, Salesforce).

□ Cloud Technologies

- **Virtualization:** Enables multiple virtual machines (VMs) on a single physical machine.
- **Containers (e.g., Docker):** Lightweight alternatives to VMs for application deployment.
- **Serverless Computing:** No need to manage infrastructure—just run code (e.g., AWS Lambda).

□ Security in Cloud Computing

- Data encryption (at rest and in transit)
- Identity and access management (IAM)
- Firewalls and intrusion detection systems
- Compliance standards (ISO, GDPR, etc.)

□ Storage Services

- **Object Storage:** Stores unstructured data (e.g., AWS S3).
- **Block Storage:** Like traditional disk drives (e.g., AWS EBS).
- **File Storage:** Shared file systems (e.g., Amazon EFS).

□ Popular Cloud Platforms

- **Amazon Web Services (AWS)**
- **Microsoft Azure**
- **Google Cloud Platform (GCP)**
- **IBM Cloud**
- **Oracle Cloud**

□ Applications of Cloud Computing

- Hosting websites and applications
- Big data analytics
- Backup and disaster recovery

- Internet of Things (IoT)
- Machine learning and AI services

Unit 2

Utility Computing, Elastic Computing, AJAX, Mashups, Virtualization & Multitenancy

---

□ Utility Computing

Definition:
Utility computing is a **service provisioning model** where computing resources like processing power, storage, and networking are provided and billed based on usage—like electricity or water.

Key Features:

- Pay-per-use model
- Scalable resources
- Delivered over the internet
- Core concept behind cloud computing

Example:
Amazon EC2 charges based on CPU usage per hour, similar to paying for electricity based on units consumed.

□ Elastic Computing

Definition:
Elastic computing refers to the **ability to scale** computing resources **up or down automatically** based on demand.

Why it's important:

- Helps handle traffic spikes efficiently (e.g., during Black Friday sales)
- Saves cost during low usage
- Core feature of cloud platforms like AWS Auto Scaling, Azure VM Scale Sets

Example:
An e-commerce site adding servers automatically as traffic increases.

□ AJAX (Asynchronous JavaScript and XML)

Definition:
AJAX is a web development technique for creating **asynchronous** and **dynamic user interfaces** that do **not require full-page reloads**.

Key Features:

- Improves user experience
- Data can be exchanged with the server and updated dynamically
- Used in rich web applications (like Google Maps, Gmail)

How it works: AJAX uses:

- JavaScript to make asynchronous calls
- XML/JSON to exchange data
- HTML/CSS for presentation

□ Mashups

Definition:
A **mashup** is a web application that **combines data or functionality from multiple sources** to create a new service.

Example:

- A real estate website that uses:
    - Google Maps API for locations
    - Weather API for climate data
    - Real estate listings from another service

Used For:

- Quick innovation
- Reusability of services
- Custom dashboards and applications

□ User Interface in Cloud

Cloud-based user interfaces need to be:

- Responsive and dynamic (thanks to AJAX)
- Lightweight and fast
- Often built using web technologies like HTML5, JavaScript, React, etc.

These interfaces allow users to manage cloud resources, monitor services, and more.

□ Virtualization Technology

□ Virtualization Applications in Enterprises

- **Server Consolidation:** Run multiple VMs on one physical server.
- **Disaster Recovery:** Snapshots and backups of VMs.
- **Development & Testing:** Isolated environments for developers.
- **Application Isolation:** Separate services to prevent conflicts.

Popular tools: VMware, KVM, Hyper-V, VirtualBox

□ Pitfalls of Virtualization

- **Overhead:** Virtualization adds performance overhead.
- **Security Risks:** One compromised VM can affect others if not isolated.
- **Complex Management:** Requires skilled staff to maintain.
- **License Costs:** Can increase with commercial tools.

□□□ Multitenant Software

Definition:
Multitenancy means a **single software instance serves multiple customers (tenants)**.

□ Multi-Entity Support

- Each customer (tenant) is logically separated.
- Same app instance, but data is isolated.

- Efficient use of resources.

□ Multischema Approach

- Each tenant has a separate database schema.
- Offers better data isolation.
- Harder to scale compared to a single schema.

□ Multitenancy using Cloud Data Stores

- Use of **NoSQL** or **relational cloud databases** (e.g., Amazon DynamoDB, Google Cloud Firestore, Azure SQL).
- Shared resources with tenant identifiers to isolate data.
- Provides scalability + flexibility.

□ Summary:

Concept	Purpose
Utility Computing	Pay-as-you-use IT model
Elastic Computing	Auto-scale resources based on demand
AJAX	Build interactive web interfaces
Mashups	Combine APIs/services into new apps
Virtualization	Run multiple OS instances on the same hardware
Pitfalls	Overhead, complexity, security
Multitenancy	One app, multiple customers with isolated data

Concept **Purpose**

Unit 3

Data in the Cloud

□ Relational Databases in the Cloud

Relational databases store data in **structured formats using tables**, rows, and columns with relationships between them.

Examples in the Cloud:

- **Amazon RDS** (supports MySQL, PostgreSQL, Oracle, etc.)
- **Google Cloud SQL**
- **Azure SQL Database**

Features:

- Scalability
- High availability
- Automated backups
- Security and compliance

□ Cloud File Systems

□ GFS (Google File System)

What it is:
A scalable distributed file system developed by Google for large data-intensive applications.

Features:

- Stores data in large **chunks** (typically 64 MB)
- Each chunk replicated (usually 3 times)
- **Master node** manages metadata
- Optimized for **read-heavy workloads**

□ HDFS (Hadoop Distributed File System)

What it is:
An open-source file system inspired by GFS, used with Hadoop.

Features:

- Data split into blocks (default 128MB/256MB)
- Blocks replicated across DataNodes (default: 3 copies)
- **NameNode** for metadata + **DataNodes** for actual storage
- Fault-tolerant, scalable, and handles large datasets

□ GFS vs. HDFS Comparison

Feature	GFS	HDFS
Ownership	Google	Apache
Block/Chunk Size	64 MB	128–256 MB
Metadata Server	Master	NameNode
Use Case	Google's internal systems	Big data applications
Data Replication	Default 3	Default 3
Availability	Very high	High with secondary nodes

□ Big Table, HBase, Dynamo

□ Bigtable (by Google)

- Distributed storage system for managing **structured data**.
- Stores data in a **sparse, distributed, multi-dimensional sorted map**.
- Used by services like Google Search, Maps, YouTube.

□ HBase (by Apache)

- Open-source implementation of Bigtable.
- Runs on top of HDFS.
- Integrates well with Hadoop and MapReduce.

- Good for **random, real-time read/write** access to big data.

□ Dynamo (by Amazon)

- Key-value storage system developed for Amazon's e-commerce platform.
- Focus on **high availability, scalability, and partition tolerance**.
- NoSQL model; eventually consistent.
- Inspired newer databases like **Amazon DynamoDB**.

□ MapReduce and Extensions

□ What is MapReduce?

A **programming model** for processing large datasets in **parallel** across a cluster of machines.

Phases:

1. **Map:** Process input data and output key-value pairs.
2. **Shuffle:** Redistribute data based on keys.
3. **Reduce:** Aggregate values with the same key.

⚡ Parallel Computing in MapReduce

- Executes operations in parallel across many nodes
- Efficient for processing **terabytes or petabytes of data**
- Reduces computation time significantly

□ Parallel Efficiency of MapReduce

- High efficiency when tasks are **independent and uniform**
- Data locality reduces overhead
- Bottlenecks: Network shuffling, straggler tasks, skewed data

---

□ Relational Operations with MapReduce

You can implement traditional SQL operations like:

- **Selection**
- **Projection**
- **Join**
- **Group By**
- **Aggregation**

All using **Map** and **Reduce** functions.

□ Enterprise Batch Processing

- Use MapReduce for periodic tasks like:
  - Log analysis
  - ETL (Extract, Transform, Load)
  - Fraud detection
- Frameworks like **Apache Hadoop** or **Apache Spark** (extension of MapReduce)
- **Example / Application of MapReduce**
- **Example – Word Count:**
- `plaintext`
- `CopyEdit`
- `Input: "Cloud computing is cool. Computing is fun."`
-
- `Map:`
- `["cloud", 1], ["computing", 1], ["is", 1], ["cool", 1], ["computing", 1], ["is", 1], ["fun", 1]`
-
- `Shuffle & Sort:`
- `["cloud", [1]], ["computing", [1,1]], ["is", [1,1]], ["cool", [1]], ["fun", [1]]`
-
- `Reduce:`
- `["cloud", 1], ["computing", 2], ["is", 2], ["cool", 1], ["fun", 1]`
-

- **□ Summary Table:**

Component	Description
GFS/HDFS	Distributed file systems for large-scale storage
Bigtable/HBase	Column-oriented storage for structured data
Dynamo	Key-value NoSQL store, highly available
MapReduce	Distributed data processing model
Relational Ops	SQL-like logic on large datasets
Enterprise Usage	Batch jobs, analytics, reporting

Unit 4

Cloud Security Fundamentals

□ Why Cloud Security Matters:

Cloud environments are **shared, distributed, and internet-facing**, which makes them **vulnerable to threats** like:

- Data breaches
- Insider threats
- Insecure APIs
- Account hijacking

Goal: Ensure **confidentiality, integrity, and availability (CIA)** of data in the cloud.

□ Vulnerability Assessment Tools for Cloud

These tools help identify **weaknesses** in cloud-based infrastructure, applications, and services.

Common Tools:

- **Nessus** – Scans for known vulnerabilities
- **OpenVAS** – Open-source vulnerability scanner
- **Qualys Cloud Security** – Monitors cloud security posture
- **AWS Inspector** – Assesses AWS-hosted apps
- **ZAP (OWASP)** – Web app vulnerability scanner

Features:

- Scans ports, misconfigurations, outdated libraries
- Detects missing patches
- Helps maintain **compliance and auditing**

□ Privacy and Security in Cloud

□ Cloud Computing Security Architecture

Key Layers:

1. **Network Security** – Firewalls, VPNs, intrusion detection/prevention
2. **Host Security** – Securing VMs, operating systems
3. **Application Security** – Authentication, authorization, input validation
4. **Data Security** – Encryption at rest & in transit, access control

Core Elements:

- Identity & Access Management (IAM)
- Encryption & Key Management
- Monitoring & Logging
- Secure APIs

□ General Issues in Cloud Security

1. **Data Breaches**
2. **Account Hijacking**
3. **Data Loss**
4. **Denial of Service (DoS)**
5. **Insider Threats**
6. **Weak Authentication**

✅Trusted Cloud Computing

Refers to cloud systems that users can **trust** with their sensitive data.

Principles:

- Transparent operations
- Verified identity of cloud providers
- Secure multi-tenancy
- Third-party security certifications (ISO 27001, SOC 2)

□ Security Challenges in Virtualized Environments

□ Virtualization Security Management

When multiple VMs share physical infrastructure, new threats arise.

□ Virtual Threats

- **VM Escape** – Malware breaks out of a VM to access the host
- **VM Sprawl** – Uncontrolled VM creation increases attack surface
- **Inter-VM Attacks** – One VM affects another on the same host
- **Snapshot Vulnerabilities** – Cloned VMs can have unpatched flaws

---

□ VM Security Recommendations

1. **Isolate VMs** properly
2. Keep hypervisors **updated and patched**
3. Use **role-based access control** (RBAC)
4. Implement **logging & monitoring**
5. **Restrict snapshot access** to admins

□ VM-Specific Security Techniques

- **Hardening VMs:** Remove unnecessary services & open ports
- **Use of Virtual Firewalls**
- **Encryption of VM images**
- **Integrity checks** for deployed instances

□ Secure Execution Environments & Communication

□ Secure Execution Environments

- Use of **Trusted Execution Environments (TEEs)** like Intel SGX
- Run code in **isolated, protected memory areas**
- Prevents unauthorized access from OS or other VMs

□ Secure Communication

- **TLS/SSL** for secure data transfer
- **VPNs** for private cloud access
- **End-to-end encryption** for sensitive data
- **Key Management Services (KMS)** for handling encryption keys securely

---

Summary Table

Concept	Description
Cloud Security Architecture	Layered protection (network, host, app, data)
Vulnerability Assessment Tools	Identify & fix security flaws
Trusted Cloud Computing	Secure, transparent, and reliable cloud services
Virtual Threats	Risks specific to virtual machines
VM Security	Best practices for protecting VMs
Secure Execution Environments	Isolated, encrypted code execution
Secure Communication	Protecting data in transit

Unit 5

Issues in Cloud Computing

Common Issues:

1. **Security & Privacy**
2. **Data Lock-in** – Difficulty moving data between providers
3. **Downtime/Availability**
4. **Limited Control** – Less transparency from providers
5. **Compliance** – GDPR, HIPAA, etc.

Implementing Real-Time Applications in Cloud

Real-time apps (like video conferencing, gaming, online trading) require:

- **Low latency**
- **High availability**
- **Scalable infrastructure**

Challenges:

- Network latency
- Unpredictable workloads
- Synchronization issues
- Resource contention

Solutions:

- Edge computing
- Auto-scaling
- Load balancing
- Using high-performance virtual machines (VMs)

□ QoS Issues in Cloud (Quality of Service)

Key QoS Parameters:

- **Latency**
- **Bandwidth**
- **Throughput**
- **Availability**
- **Jitter (variation in delay)**

Problems in Cloud:

- Variable performance due to shared resources
- SLA (Service Level Agreement) breaches

Approach:

- Use **QoS-aware scheduling and resource allocation**
- Monitor workloads and performance dynamically

□ Dependability in Cloud

Dependability = Reliability + Availability + Maintainability + Security

Challenges:

- Ensuring uptime
- Fault tolerance in distributed systems
- Redundancy in storage & compute

Solution: Use techniques like:

- Replication
- Backup & Recovery
- Monitoring & Auto-healing tools

□ Data Migration in Cloud

What is it?
Moving data between cloud platforms or from on-premise to cloud.

Issues:

- Downtime
- Data integrity
- Data loss
- Security risks

Tools: AWS Snowball, Google Transfer Service, Azure Data Box

□ Streaming in Cloud

Used for **real-time data processing** like video streaming, sensor data, financial feeds.

Challenges:

- Low latency requirement
- Bandwidth management
- Real-time analytics

Solutions:

- Apache Kafka
- Amazon Kinesis
- Google Dataflow

□ Cloud Middleware

Middleware = Software layer between OS and applications.

In cloud:

- Helps manage **communication, authentication, scaling, and APIs**
- Examples: Docker, Kubernetes, OpenStack middleware layers

Roles:

- Resource abstraction
- Service orchestration
- Performance monitoring

□ Mobile Cloud Computing (MCC)

What is it?

Cloud computing capabilities extended to mobile devices.

Benefits:

- Battery & storage offloading
- Access powerful cloud processing
- Platform-independent apps

Example: Mobile apps using Firebase backend or iCloud for storage

---

□ Inter-Cloud Issues

Inter-cloud = Multiple cloud systems working together

Problems:

- **Interoperability** between providers
- **Data portability**
- **Security policy mismatches**
- **Different APIs and standards**

Solution: Use standardized interfaces, federated cloud models

---

☁️ A Grid of Clouds / Sky Computing

Grid of Clouds:

- Multiple clouds combined for large-scale processing
- Like grid computing + cloud computing

Sky Computing:

- Vision of **globally interconnected clouds** (like the internet)
- Users can move workloads freely between cloud providers

⚖️ Load Balancing in Cloud

Distributes incoming traffic across multiple servers/resources.

Benefits:

- Avoids overload
- Improves performance & reliability

Tools: AWS ELB, NGINX, HAProxy

⚙️ Resource Optimization & Dynamic Reconfiguration

Resource Optimization:

- Allocating resources efficiently based on demand
- Minimizes cost and waste

Dynamic Reconfiguration:

- Reallocating VMs, storage, and network dynamically based on performance needs or failure recovery

Examples:

- Auto-scaling groups in AWS
- Kubernetes pod rebalancing

☁ Monitoring in Cloud

Helps track:

- CPU usage
- Memory
- Network traffic
- Service health

Tools:

- AWS CloudWatch
- Azure Monitor
- Prometheus + Grafana

☁ Installing Cloud Platforms & Performance Evaluation

Popular Cloud Platforms:

- **OpenStack** (open-source private cloud)
- **CloudStack**
- **Eucalyptus**

Steps:

- Install base OS (e.g., Ubuntu)
- Install platform (e.g., OpenStack components: Nova, Neutron, Swift)
- Configure networking & storage
- Launch VMs

Performance Evaluation:

- Measure **latency, throughput, uptime, resource usage**
- Use benchmarking tools like Apache JMeter, Iperf, Sysbench

☁ Features & Functions of Cloud Computing Platforms

Core Features:

- **Scalability**

- **Elasticity**
- **High Availability**
- **Self-service provisioning**
- **Pay-as-you-go**

Functions:

- Hosting VMs/containers
- Providing storage & networking
- Identity and Access Management
- Analytics & Machine Learning services

□ Quick Summary:

Concept	Key Point
QoS Issues	Performance metrics in cloud apps
Data Migration	Move data securely & reliably
Cloud Middleware	Software glue between cloud layers
Mobile Cloud	Cloud capabilities on mobile
Sky Computing	Global cloud integration
Load Balancing	Efficient traffic distribution
Monitoring	Track system health & resources
Cloud Installation	Deploying platforms like OpenStack
Performance Evaluation	Benchmarking & resource tracking