



## Module Handout

**Module Title:** Distributed and Cloud Computing

**Module Code:** CSC422

**School:** School of Computing and Information Technology

**Lecturer:** Faustin KWIZERA

**Year:** 4      **Trimester:** 2

## Lessons Outline:

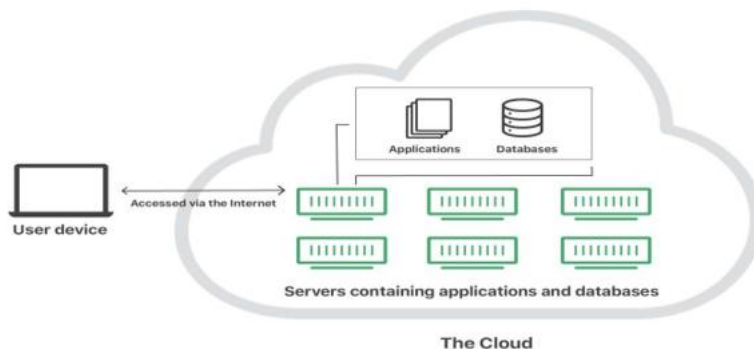
1. Introduction to Cloud Computing
2. Introduction to distributed computing concepts
3. Distributed Systems and Cloud Middleware
4. Cloud Computing Services Model
5. Big Data Analytics in Cloud Computing
6. Robust Distributed Systems and Supercomputing Foundations in Cloud Environments

INDEX		
UNIT	Topics	Page No
I	Definition of Cloud computing	3
	Characteristics of Cloud Computing	4
	Challenges and Risks of Cloud Computing	5
	Types of Clouds	6
	Cloud Computing Architecture	7
II	Introduction to distributed and parallel computing	9
	Types of Distributed Systems	9
	Advantages & disadvantages of Distributed Systems	10
	Characterization of Distributed systems	11
	Modes of Inter-Process Communication (IPC)	12
	Approaches to Inter Process Communication	13
	Remote Invocation in Distributed Systems & Cloud Computing	15
	Failure and timeouts	16
III	Introduction to Distributed Systems and Cloud Middleware	17
	Importance of Middleware in Distributed System	17
	Types of Middleware in Distributed Systems	17
	Consistency and Replication	18

	Large Scale Distributed Systems	19
	Security Considerations for Large Scale Distributed Systems	20
IV	Introduction to Cloud Computing Service Models	21
	Software as a Service (SaaS) & Characteristics	22
	Platform as a Service (PaaS) & Characteristics	22
	Infrastructure as a Service (IaaS) & Characteristics	23
	Essential criteria for selecting the best cloud service provider	24
V	Introduction to Big-Data Analytics	25
	5V's of Big Data	25
	Challenges and Benefits of Big Data Analytics in Cloud Computing	27
	Big Data Analytics Technologies and Tools	28
	Hadoop Architecture	29
	Google File System vs. Hadoop Distributed File System	30
	MapReduce Architecture and Components	31
VI	Introduction to Robust Distributed Systems	32
	Types of Testing to ensure Robustness of Test Suites	33
	Advantages of Robustness Testing	34
	Introduction to High performance Computing (Supercomputing)	34
	Importance of High-performance Computing	35
	Challenges with HPC	35
	Applications of HPC	36

## UNIT 1. Introduction to Cloud Computing

"**The cloud**" refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are in data centers all over the world. By using cloud computing, users and companies do not have to manage physical servers themselves or run software applications on their own machines.



The cloud enables users to access the same files and applications from almost any device, because the computing and storage takes place on servers in a data center, instead of locally on the user device.

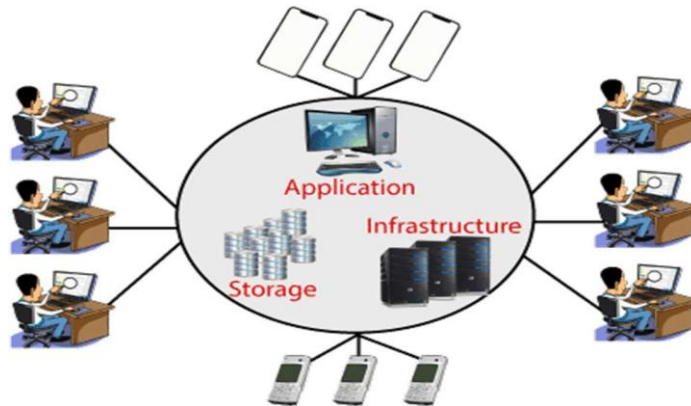
**Therefore**, a user can log into their Instagram account on a new phone after their old phone breaks and still find their old account in place, with all their photos, videos, and conversation history. It works the same way with cloud email providers like Gmail or Microsoft Office 365, and with cloud storage providers like Dropbox or Google Drive.

**For businesses**, switching to cloud computing removes some IT costs and overhead: for instance, they no longer need to update and maintain their own servers, as the cloud vendor they are using will do that. This especially makes an impact for small businesses that may not have been able to afford their own internal infrastructure but can outsource their infrastructure needs affordably via the cloud. The cloud can also make it easier for companies to operate internationally, because employees and customers can access the same files and applications from any location.

### Definition of Cloud Computing:

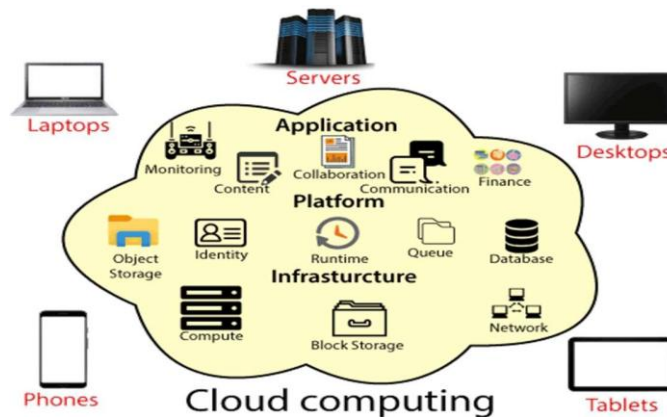
The term “Cloud Computing” refers to services provided by the cloud that is responsible for delivering of computing services such as servers, storage, databases, networking, software, analytics, intelligence, and more, over the Cloud (Internet).

Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and data sets dynamically



Cloud Computing provides an alternative to the on-premises data center. With an on-premises data center, we must manage everything, such as purchasing and installing hardware, virtualization, installing the operating system, and any other required applications, setting up the network, configuring the firewall, and setting up storage for data. After doing all the set-up, we become responsible for maintaining it through its entire lifecycle.

However, if we choose Cloud Computing, a cloud vendor is responsible for the hardware purchase and maintenance. They also provide a wide variety of software and platform as a service. We can take any required services on rent. The cloud computing services are charged based on usage.



## 1.1 Characteristics of Cloud Computing

- ✓ Cost reductions, Centralization of infrastructure in locations with lower costs.
- ✓ Device and location independence, which means no maintenance, required.
- ✓ Pay-per-use means utilization and efficiency improvements for systems that are often only 10–20% utilized.
- ✓ Performances are being monitored by IT experts i.e., from the service provider end.
- ✓ Productivity increases which results in multiple users who can work on the same data simultaneously.



- ✓ Time may be saved as information does not need to be re-entered when fields are matched
- ✓ Availability improves with the use of multiple redundant sites
- ✓ Scalability and elasticity via dynamic ("on-demand") provisioning of resources on a fine-grained, self-service basis in near real-time without users having to engineer for peak loads.
- ✓ Self-service interface.
- ✓ Resources that are abstracted or virtualized.
- ✓ Security can improve due to centralization of data

The National Institute of Standards and Technology's definition of cloud computing identifies "five essential characteristics":

1. On-demand self-service.
2. Broad network access.
3. Resource pooling.
4. Rapid elasticity.
5. Measured service.

## 1.2 Challenges and Risks of Cloud Computing

Despite the initial success and popularity of the cloud computing paradigm and the extensive availability of providers and tools, a significant number of challenges and risks are inherent to this new model of computing. Providers, developers, and end users must consider these challenges and risks to take good advantage of cloud computing. Issues to be faced include user privacy, data security, data lock-in, availability of service, disaster recovery, performance, scalability, energy- efficiency, and programmability.

- **Security, Privacy, and Trust:** Security and privacy affect the entire cloud computing stack, since there is a massive use of third-party services and infrastructures that are used to host important data or to perform critical operations. In this scenario, the trust toward providers is fundamental to ensure the desired level of privacy for applications hosted in the cloud. Legal and regulatory issues also need attention. When data are moved into the Cloud, providers may choose to locate them anywhere on the planet. The physical location of data centers determines the set of laws that can be applied to the management of data.

**For example,** specific cryptography techniques could not be used because they are not allowed in some countries. Similarly, country laws can impose that sensitive data, such as patient health records, are to be stored within national borders.

- **Data Lock-In and Standardization:** A major concern of cloud computing users is about having their data locked-in by a certain provider. Users may want to move data and applications out from a provider that does not meet their requirements. However, in their current form, cloud computing infrastructures and platforms do not employ standard methods of storing user data and applications. Consequently, they do not interoperate, and user data are not portable.



The answer to this concern is standardization. In this direction, there are efforts to create open standards for cloud computing. The Cloud Computing Interoperability Forum (CCIF) was formed by organizations such as Intel, Sun, and Cisco to “enable a global cloud computing ecosystem whereby organizations are able to seamlessly work together for the purposes for wider industry adoption of cloud computing technology.” The development of the Unified Cloud Interface (UCI) by CCIF aims at creating a standard programmatic point of access to an entire cloud infrastructure. In the hardware virtualization sphere, the Open Virtual Format (OVF) aims at facilitating packing and distribution of software to be run on VMs so that virtual appliances can be made portable—that is, seamlessly run on hypervisor of different vendors.

- **Availability, Fault-Tolerance, and Disaster Recovery:** It is expected that users will have certain expectations about the service level to be provided once their applications are moved to the cloud. These expectations include availability of the service, its overall performance, and what measures are to be taken when something goes wrong in the system or its components. In summary, users seek for a warranty before they can comfortably move their business to the cloud. SLAs, which include QoS requirements, must be ideally set up between customers and cloud computing providers to act as warranty.

Additionally, metrics must be agreed upon by all parties, and penalties for violating the expectations must also be approved.

- **Resource Management and Energy-Efficiency:** One important challenge faced by providers of cloud computing services is the efficient management of virtualized resource pools. Physical resources such as CPU cores, disk space, and network bandwidth must be sliced and shared among virtual machines running potentially heterogeneous workloads. The multi-dimensional nature of virtual machines complicates the activity of finding a good mapping of VMs onto available physical hosts while maximizing user utility.

## 1.3 Types of clouds

A cloud deployment model represents a specific type of cloud environment, primarily distinguished by ownership, size, and access.

There are four common cloud deployment models:

### 1. Public Cloud:

A public cloud is a publicly accessible cloud environment owned by a third-party cloud provider. The IT resources on public clouds are usually provisioned via cloud delivery models and are generally offered to cloud consumers at a cost or are commercialized via other avenues (such as advertisement) eg: Google, Oracle, Microsoft

### 2. Community Clouds:

A community cloud is like a public cloud except that its access is limited to a specific community of cloud consumers. eg: Government agency

### 3. Private Clouds:

A private cloud is owned by a single organization. Private clouds enable an organization to use cloud



computing technology as a means of centralizing access to IT resources by different parts, locations, or departments the organization. eg: HP data center, Ubuntu

#### 4. Hybrid clouds:

hybrid cloud is a cloud environment comprised of two or more different cloud deployment models.  
eg: Amazon Web service

### 1.4 Private Cloud and Infrastructure Services

A private cloud aims at providing public cloud functionality, but on private resources:

1. Maintaining control over an organization's data and resources to meet security and governance's requirements in an organization.
2. Private cloud exhibits a highly virtualized cloud data center located inside your organization's firewall.
3. It may also be a private space dedicated for your company within a cloud vendor's data center designed to handle the organization's workloads.

#### Private clouds exhibit the following characteristics:

1. Allow service provisioning and compute capability for an organization's users in a self-service manner.
2. Automate and provide well-managed virtualized environments.
3. Optimize computing resources, and servers' utilization.
4. Support specific workloads.

### 1.5 Cloud Computing Architecture

In current time Cloud Computing is giving a new shape to every organization by providing on demand virtualized services/resources. Starting from small to medium and medium to large, every organization use cloud computing services in storing information and accessing that from anywhere and anytime only with the help of internet.

#### Cloud Computing Architecture:

The cloud architecture is divided into 2 parts i.e.

- Frontend
- Backend



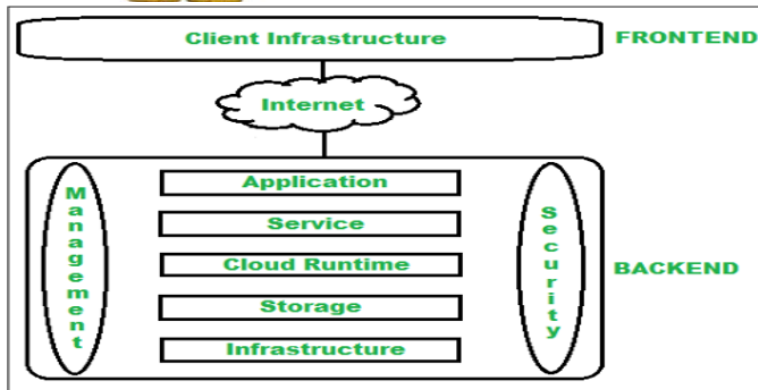


Figure - Internal Architectural view of Cloud Computing

### 1. Front- end:

Frontend of the cloud architecture refers to the client side of cloud computing system. Means it contains all the user interfaces and applications which are used by the client to access the cloud computing services/resources.

#### Client Infrastructure:

Client Infrastructure refers to the frontend components. It contains the applications and user interfaces which are required to access the cloud platform.

### 2. Internet:

Internet connection acts as the medium or a bridge between frontend and backend, it establishes the interaction and communication between frontend and backend.

### 3. Back- end:

Backend refers to the cloud itself which is used by the service provider. It contains the resources as well as manages the resources and provides security mechanisms. Along with this it includes huge storage, virtual applications, virtual machines, traffic control mechanisms, deployment models etc.

#### ➤ Application:

Application in backend refers to a software or platform to which client accesses. Means it provides the service in backend as per the client requirement.

#### ➤ Service:

Service in backend refers to the major three types of cloud-based services like SaaS, PaaS and IaaS. Also manages which type of service the user accesses.

#### ➤ Cloud Runtime:

Runtime cloud in backend refers to provide the execution and runtime platform / environment to the virtual machine

#### ➤ Storage:

Storage in backend refers to provide flexible and scalable storage service and management of stored data.

#### ➤ Infrastructure:

Cloud Infrastructure in backend refers to hardware and software components of cloud like it includes



servers, storage, network devices, virtualization software etc.

➤ **Management:**

Management in backend refers to management of backend components like application, service, runtime cloud, storage, infrastructure, and other security mechanisms etc.

➤ **Security:**

Security in backend refers to implementation of different security mechanisms in the backend for secure cloud resources, systems, files, and infrastructure to end-users.

## UNIT 2. Introduction to distributed computing concepts

### 2.1 Introduction to distributed and parallel computing

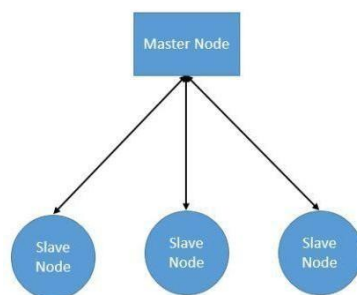
A distributed system contains multiple nodes that are physically separate but linked together using the network. All the nodes in this system communicate with each other and handle processes in parallel. Each of these nodes contains a small part of the distributed operating system software.

#### Types of Distributed Systems:

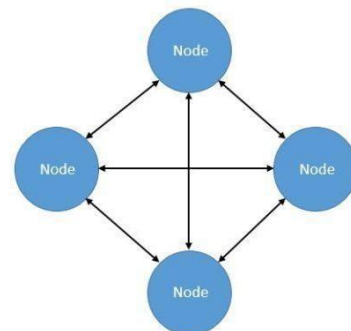
The nodes in the distributed systems can be arranged in the form of client/server systems or peer to peer systems. Details about these are as follows –

**Client/Server Systems:** In client server systems, the client requests a resource, and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network and so they are a part of distributed systems.

**Peer to Peer Systems:** The peer-to-peer systems contains nodes that are equal participants in data sharing. All the tasks are equally divided between all the nodes. The nodes interact with each other as required as share resources. This is done with the help of a network.



Master-Slave Model



Peer-to-Peer Model

### Advantages of Distributed Systems:

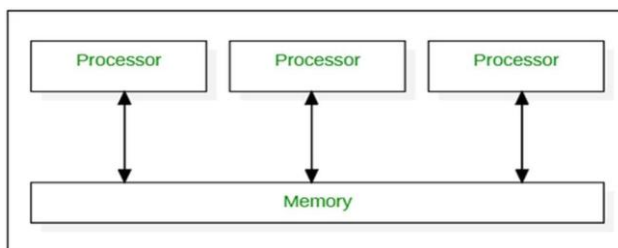
- All the nodes in the distributed system are connected to each other. So, nodes can easily share data with other nodes.
- More nodes can easily be added to the distributed system i.e. it can be scaled as required.
- Failure of one node does not lead to the failure of the entire distributed system. Other nodes can still communicate with each other.
- Resources like printers can be shared with multiple nodes rather than being restricted to just one.

### Disadvantages of Distributed Systems:

- It is difficult to provide adequate security in distributed systems because the nodes as well as the connections need to be secured.
- Some messages and data can be lost in the network while moving from one node to another.
- The database connected to the distributed systems is quite complicated and difficult to handle as compared to a single user system.
- Overloading may occur in the network if all the nodes of the distributed system try to send data at once.

**Parallel Computing** involves the simultaneous use of multiple processors within a single machine to solve a computational problem. Tasks are broken down into smaller sub-tasks, which are executed concurrently to reduce execution time. Used in high-performance computing applications such as scientific simulations, image processing, and large-scale data analysis.

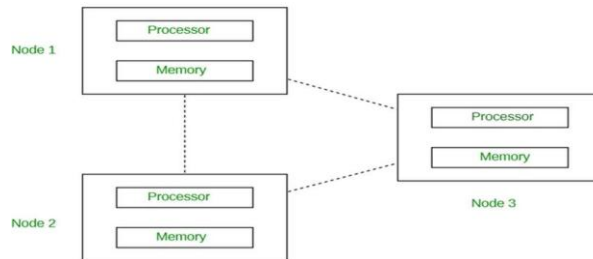
Here in the below diagram, you can see how the parallel computing system consists of multiple processors that communicate with each other and perform multiple tasks over a shared memory simultaneously.



**Distributed Computing** refers to the use of multiple independent computers, often geographically scattered, that work together to achieve a common goal. These systems communicate over a network, coordinate task execution, and share resources. Distributed computing is the foundation of technologies such as cloud computing, peer-to-peer systems, and web services.

Distributed computing is defined as a type of computing where multiple computer

systems work on a single problem. Here all the computer systems are linked together, and the problem is divided into sub-problems where each part is solved by different computer systems. The goal of distributed computing is to increase the performance and efficiency of the system and ensure fault tolerance. In the below diagram, each processor has its own local memory, and all the processors communicate with each other over a network.



## 2.2 Characterization of Distributed systems

### 1. Resource Sharing

- ✓ Multiple users and applications can share hardware (e.g., printers), software (e.g., databases), and data.
- ✓ Resources are located on different networked computers but appear as a single system.

### 2. Concurrency

- ✓ Many processes run simultaneously, often on different nodes.
- ✓ Synchronization and coordination are essential to avoid conflicts and ensure consistency.

### 3. Scalability

- ✓ The system can grow (e.g., adding more nodes or users) without performance degradation.
- ✓ Supports geographic distribution and high-volume services.

### 4. Fault Tolerance

- ✓ The system can continue functioning even when some components fail.
- ✓ Techniques like replication, redundancy, and failover mechanisms are used.

### 5. Transparency

- ✓ Distributed systems aim to hide the complexity of the underlying infrastructure from users and developers. Several types of transparency include:
  - **Access transparency** – users access resources uniformly.
  - **Location transparency** – resource location is hidden.
  - **Replication transparency** – multiple copies of data appear as one.
  - **Concurrency transparency** – multiple users access without interference.
  - **Failure transparency** – users are unaware of failures.

### 6. Openness

- ✓ Based on standard protocols and interfaces.
- ✓ Easily extensible and interoperable with other systems.

### 7. Heterogeneity

- ✓ Consists of different types of hardware, operating systems, networks, and programming languages.
- ✓ Middleware or standard interfaces help manage this diversity.

### 8. Security

- ✓ Must ensure authentication, authorization, confidentiality, and data integrity across multiple nodes and networks.



## 9. Latency and Network Issues

- ✓ Communication over the network can introduce delays.
- ✓ Systems must manage timeouts, dropped messages, and varying bandwidth.

## 10. Autonomy of Components

- ✓ Each node (computer or device) operates independently and makes its own decisions while cooperating to complete system tasks.

### 2.3 Modes of Inter-Process Communication (IPC)

Inter-Process Communication (IPC) in distributed systems and cloud computing refers to the mechanisms and protocols that allow different processes that often running on separate machines or virtual instances to exchange data and coordinate with each other.

IPC is the set of techniques used to enable communication and data exchange between multiple processes, which can be:

- **On the same machine** (e.g., threads or processes in an OS)
- **On different machines** (as in distributed systems and cloud computing)

**Shared memory and message passing** are the two modes through which processes can communicate with each other. The process is divided into several sections.

#### Shared Memory

To establish a shared memory region, require communicating process which can run through the shared memory model. Every process has its own memory region as well as they have its own address space to communicate with each other, the other process which want to communicate with them, they need to attach their address space to this shared memory segment

#### Message Passing

The shared memory model is very useful model for transferring the data, but it is not giving the suitable outcome for some processes. some process we can't achieve through this for example if the data is at various system and the process needs to exchange the data through different computer systems in distributed computing, they don't have straightforward way to communicate with each other in a shared memory region.

To successfully exchange messages, there needs to be a communication link between the processes. There are several techniques through which these communication links are established. Following points discussed the mechanisms:

**Direct Communication:** In this type of the direct sender and receiver is including in the process, every process has explicit the recipient or the sender. For example, if process A needs to send a message to process B, it can use the primitive

**Indirect communication** refers to a form of inter-process communication (IPC) where the sender and receiver do not communicate with each other directly. Instead, they interact through an intermediary like a message queue, broker, or shared data store.

#### Synchronization:



In this type of the additional option of synchronization is there which can have extension to direct and indirect communication. There is block to send and receive message which can be based on the need of a process can choose to block while sending or receiving messages Besides, it can communicate without any blocking is called asynchronous.

### Buffering:

The temporary queue for exchanged messages is called as buffering. These queues can be of zero, bounded, and unbounded capacity.

### Approaches to Inter Process Communication

- Pipe:

A unidirectional channel of data transfer is called as pipe. In this one pipe is used for one-way data channel if both the way data transfer then we have to use two pipes for two processes.

- File:

A set of data record is called as file it is stored on a disk or acquired on demand by a file server. As per the requirement multiple processes can access a file. Files can be used for data storage in all operating systems.

- Signal:

In a limited way of inter-process communication, the signals are used. The messages are sent from one process to another. Normally, signals are not used to transfer data but are used for remote commands between processes.

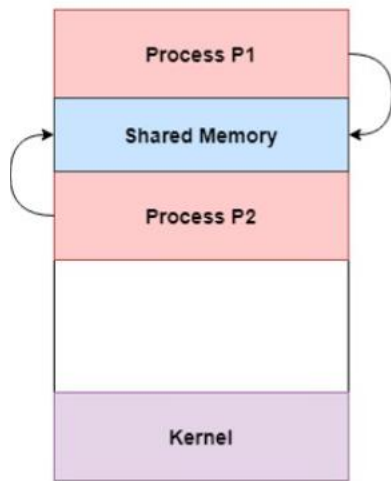
- Shared Memory:

Multiple process at the same time shares the resources with the help of Shared memory. Due to this all process can communicate with one another in easiest way.

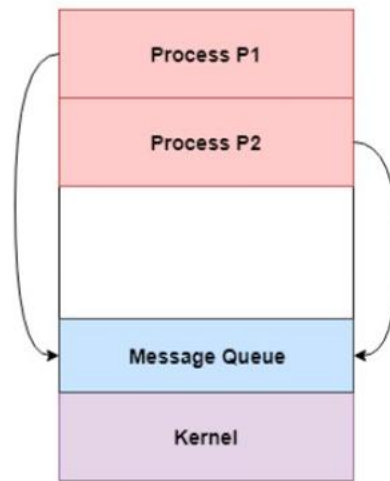
- Message Queue:

Without being connected to each other multiple processes can read and write data to the message queue. Until their recipient retrieves, messages are stored in the queue. In inter-process communication Message queues are quite useful and are used by most operating systems.

Approaches to Interprocess Communication



Shared Memory



Message Queue

## 2.3 Remote Invocation in Distributed Systems & Cloud Computing

Remote Invocation is a mechanism that allows a program (client) to invoke a function or procedure on a remote machine (server) as if it were a local call (request-reply communication).

Difference between Remote and Local procedure call:

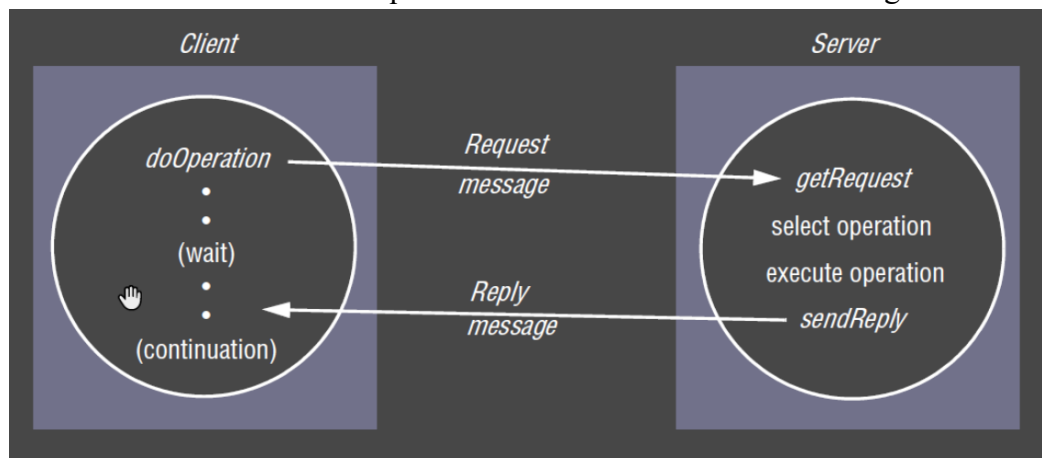
- ✓ Local procedure call: is done either by call by value or call by address/reference.
- ✓ Remote procedure call: address space of procedure calling and one that is called are different

### Request-Reply Protocol (RPP)

Most common exchange protocol for remote invocation

Why RRP?

- Establishing connection requires extra message other than send and receive
- Acknowledgements are unnecessary if request and reply are done immediately
- Flow control is not required in remote invocations if small arguments and results are passed



Operations

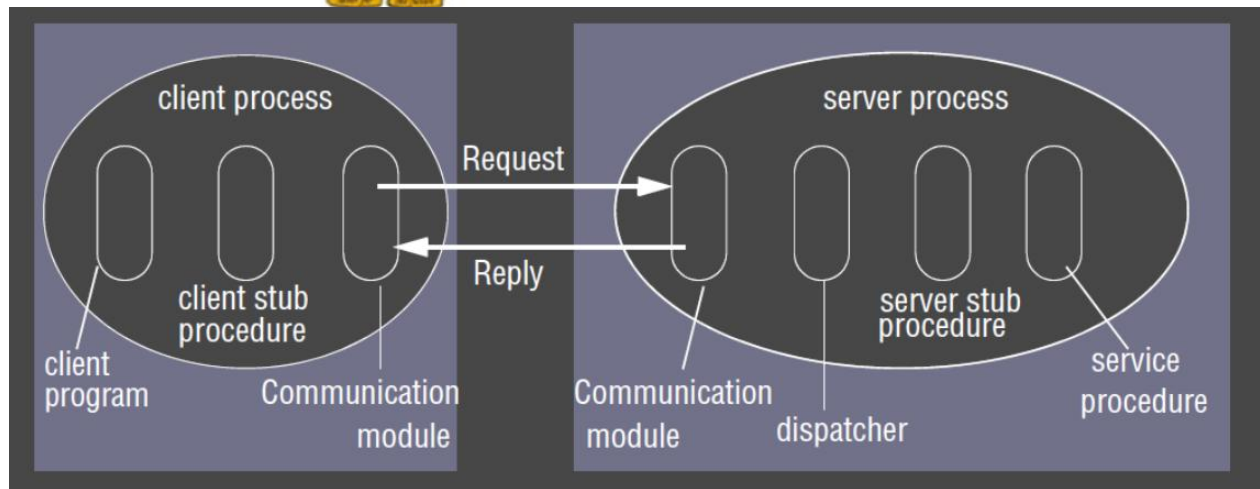
- **doOperation():** send request to remote object, and returns the reply received
- **getRequest():** acquire client request at server port
- **sendReply():** sends reply message from server to client

### 2-way exchange of messages as in inter-process communication

- **Remote Procedure Call (RPC):** The extended form of RPC is Remote Procedure Call. The use of RPC is in used for client-server applications. RPC mechanisms are used when a computer program causes a procedure or subroutine to execute in a different address space, which is coded as a normal procedure call without the programmer specifically coding the details for the remote interaction.

This procedure call also manages low-level transport protocol, such as User Datagram Protocol, Transmission Control Protocol/Internet Protocol etc. It is used for carrying the message data between programs.





**Remote Method Invocation (RMI):** is a way that a programmer, using the Java programming language and development environment, can write object-oriented programming in which objects on different computers can interact in a distributed network. RMI is the Java version of what is generally known as a remote procedure call (RPC), but with the ability to pass one or more objects along with the request. The object can include information that will change the service that is performed in the remote computer. Sun Microsystems, the inventors of Java, calls this "moving behavior."

### Failure and timeouts

**Failures** indicate that a task or process has encountered an error and cannot complete successfully.

- **Examples:**
  - A database query returning an error code.
  - A file operation failing due to insufficient permissions.
  - A network request timing out due to a broken connection.
- **Handling:**

Failures are typically handled through error handling mechanisms like try-except blocks, retries, or fallback strategies.

**Timeouts** occur when a task exceeds a predefined time limit and is automatically terminated.

- **Examples:**
  - A network request waiting for a response exceeding the timeout.
  - A long-running computation exceeding the allowed execution time.
- **Handling:**

Timeouts are often used to prevent a slow or unresponsive task from blocking other operations and to ensure the overall system remains responsive.



## UNIT 3. Distributed Systems and Cloud Middleware

It simplifies the development and deployment of applications by abstracting away the complexities of distributed computing and enabling seamless communication between different parts of a system.

In distributed systems, middleware is a software component that provides services between two or more applications and can be used by them. Middleware can be thought of as an application that sits between two separate applications and provides service to both.

### 3.1 Importance of Middleware in Distributed System

Below is the importance of Middleware in Distributed Systems:

- **Facilitates Communication**
  - **Seamless Interaction:** Middleware enables different components, often running on separate machines, to communicate effectively, overcoming the challenges posed by network heterogeneity and protocol differences.
  - **Standardized Interfaces:** It provides standardized communication protocols and APIs, simplifying the development of distributed applications by abstracting complex communication details.
- **Supports Integration**
  - **Service Integration:** Middleware integrates diverse services and applications, ensuring that various system components can work together, even if they were developed using different technologies or platforms.
  - **Data and Application Integration:** It enables the integration of various data sources and applications, facilitating the creation of composite services and systems.
- **Ensures Data Consistency**
  - **Transaction Management:** Middleware provides mechanisms for managing transactions across distributed databases, ensuring data consistency and integrity through features like two-phase commit protocols.
  - **Data Synchronization:** It helps in synchronizing data across different components and databases, maintaining consistency and coherence in distributed environments.

### 3.2 Types of Middleware in Distributed Systems

In distributed systems, middleware is crucial for enabling communication, integration, and management of various software components and services. Here's an overview of the primary types of middleware:

#### 1. Communication Middleware

- **Message-Oriented Middleware (MOM):** Manages the communication between distributed applications using messages. It supports asynchronous message passing, queuing, and publish-subscribe models. Examples: RabbitMQ, Apache Kafka.
- **Remote Procedure Call (RPC) Middleware:** Facilitates remote procedure calls where a program can execute procedures on a remote server as if they were local. Examples: gRPC, Apache Thrift.



## 2. Database Middleware

- **Object Request Brokers (ORBs):** Facilitates communication between distributed objects in an object-oriented system. ORBs handle requests and responses between objects across different locations. Examples: CORBA (Common Object Request Broker Architecture).
- **Database Connectivity Middleware:** Provides a layer that manages connections between applications and database systems, supporting features like connection pooling and transaction management. Examples: JDBC (Java Database Connectivity), ODBC (Open Database Connectivity).

## 3. Transaction Middleware

- **Transaction Processing Monitors (TPMs):** Manage transactions across multiple systems, ensuring that all parts of a distributed transaction are completed successfully. They handle transaction coordination, recovery, and rollback. Examples: IBM CICS, BEA Tuxedo.
- **Two-Phase Commit Protocol:** Ensures atomicity in distributed transactions by coordinating commit decisions across multiple systems. It involves a preparation phase and a commit phase to ensure consistency.

A **distributed transaction** is a transaction that involves multiple nodes or databases in a distributed system, ensuring that either all operations succeed or none are applied, While **Concurrency Control in Distributed Systems** Ensures correct execution of concurrent transactions by preserving isolation.

- **Example:** A bank transfer from Account A (in one database) to Account B (in another database) must debit and credit both accounts successfully—or roll back entirely.

## 4. Application Middleware

- **Enterprise Service Bus (ESB):** Provides a communication backbone for integrating various applications and services within an enterprise. It supports message routing, transformation, and service orchestration. Examples: MuleSoft, Apache ServiceMix.
- **Web Middleware:** Supports the development and deployment of web applications and services, handling tasks like request routing, session management, and web service integration. Examples: Apache Tomcat, Microsoft IIS.

## 3.3 Consistency and Replication

An important issue in distributed systems is the replication of data. Data are generally replicated to enhance reliability or improve performance. One of the major problems is keeping replicas consistent. Informally, this means that when one copy is updated, we need to ensure that the other copies are updated as well; otherwise, the replicas will no longer be the same. Main questions here are “why replication is useful? and how it relates to scalability? **What consistency actually means**”.



First, we start with concentrating on managing replicas, which takes into account not only the placement of replica servers, but also how content is distributed to these servers.

The second issue is how replicas are kept consistent. In most cases, applications require a strong form of consistency. Informally, this means that updates are to be propagated more or less immediately between replicas.

**There are two primary reasons for replicating data including reliability and performance:**

**1) Data are replicated to increase the reliability of a system.**

If a file system has been replicated it may be possible to continue working after one replica crashes by simply switching to one of the other replicas. Also, by maintaining multiple copies, it becomes possible to provide better protection against corrupted data. For example, imagine there are three copies of a file and every read and write operation is performed on each copy.

We can safeguard ourselves against a single, failing write operation, by considering the value that is returned by at least two copies as being the correct one.

**2) Replication for performance**

➤ **Scaling in numbers:** Replication for performance is important when the distributed system needs to scale in numbers and geographical area. Scaling in numbers occurs, for example, when an increasing number of processes needs to access data that are managed by a single server. In that case, performance can be improved by replicating the server and subsequently dividing the work.

➤ **Scaling in geographical area:** The basic idea is that by placing a copy of data in the proximity of the process using them, the time to access the data decreases. Therefore, the performance as perceived by that process increases. The benefits of replication for performance may be hard to evaluate. Although a client process may perceive better performance, it may also be the case that more network bandwidth is now consumed keeping all replicas up to date.

### **3.4 Large Scale Distributed Systems**

Large Scale Distributed Systems refer to systems composed of multiple interconnected computers or nodes, typically geographically dispersed and working together to achieve a common goal. These systems are designed to handle massive amounts of data, support high throughput, and ensure reliability and fault tolerance.

#### **Architectural Patterns for Large-Scale Distributed Systems**

Architectural patterns for Large Scale Distributed Systems provide structured approaches and guidelines for designing systems that can handle massive scale, high availability, and distributed nature. These patterns help address common challenges such as scalability, fault tolerance, consistency, and performance optimization.



Below are some key architectural patterns commonly used in Large Scale Distributed Systems:

1. **Microservices Architecture:** In this pattern, a large application is broken down into smaller, loosely coupled services that can be developed, deployed, and scaled independently
2. **Service-Oriented Architecture (SOA):** Similar to microservices, SOA breaks down an application into services, but typically with a focus on enterprise-level services that are often larger in scope.
3. **Event-Driven Architecture (EDA):** This pattern emphasizes the production, detection, consumption, and reaction to events that occur within a system.
4. **Distributed Caching:** Involves caching data in-memory across distributed nodes to reduce latency and improve performance.
5. **Load Balancing:** Techniques for distributing incoming network traffic across multiple servers or resources to optimize resource utilization, maximize throughput, and minimize response time.

### 3.5 Security Considerations for Large Scale Distributed Systems

Security considerations for Large Scale Distributed Systems (LSDS) are crucial due to their complex nature, involving multiple interconnected components and potentially spanning across different geographical locations. Addressing security concerns in LSDS involves mitigating risks related to data breaches, unauthorized access, denial-of-service attacks, and ensuring compliance with regulations. Here are key security considerations:

#### 1. Authentication and Authorization:

- **Challenge:** Ensuring that only legitimate users and services can access resources within the distributed system.
- **Solution:** Implement strong authentication mechanisms such as multi-factor authentication (MFA), OAuth, or JWT (JSON Web Tokens). Use centralized identity providers or federated identity management for consistent authentication across nodes.

#### 2. Data Encryption:

- **Challenge:** Protecting data both in transit and at rest to prevent unauthorized access or tampering.
- **Solution:** Use encryption protocols like TLS (Transport Layer Security) for securing communications between nodes and clients. Employ encryption mechanisms for data storage, such as AES (Advanced Encryption Standard) for sensitive data stored in databases or distributed file systems.

#### 3. Secure APIs and Interfaces:

- **Challenge:** Ensuring that APIs and interfaces used for communication between distributed components are secure and not vulnerable to attacks like injection or improper authentication.
- **Solution:** Validate and sanitize input data to prevent injection attacks (e.g., SQL injection, XSS). Implement rate limiting and throttling to protect against API abuse and denial-of-





service attacks. Use API gateways with built-in security features for centralized API management and enforcement of security policies.

#### 4. Data Integrity and Consistency:

- **Challenge:** Maintaining data integrity and consistency across distributed nodes, especially in the presence of concurrent updates and replication.
- **Solution:** Use cryptographic techniques like digital signatures and hashes to verify data integrity. Implement distributed transaction protocols or eventual consistency models depending on application requirements. Ensure proper synchronization and conflict resolution mechanisms for replicated data.

## UNIT 4. Cloud Computing Service Models

SaaS, PaaS, and IaaS are the three main cloud computing service model categories. You can access all three via an Internet browser or online apps available on different devices. The cloud service model enables the team to collaborate online instead of offline creation and then share online.

### 4.1 Software as a Service (SaaS)

Software as a Service (SaaS) is a web-based deployment model that makes the software accessible through a web browser. SaaS software users don't need to care where the software is hosted, which operating system it uses, or even which programming language it is written in. The SaaS software is accessible from any device with an internet connection. This cloud service model ensures that consumers always use the most current version of the software. The SaaS provider handles maintenance and support. In the SaaS model, users don't control the infrastructure, such as storage, processing power, etc.





## Characteristics of SaaS

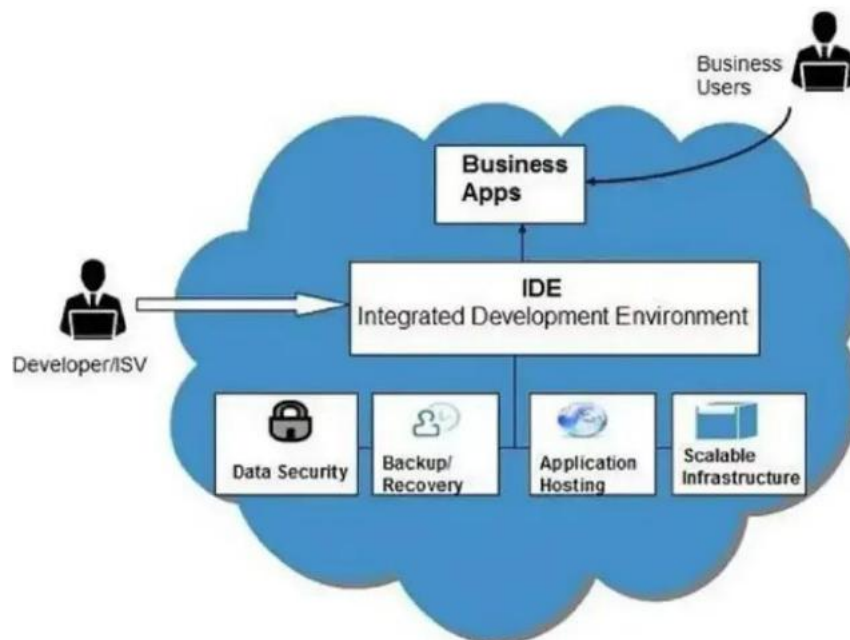
There are the following characteristics of SaaS:

- It is managed from a central location.
- Hosted directly on a remote server.
- It is accessible over the Internet.
- SaaS users are not responsible for hardware and software updates.
- The services are purchased on a pay-as-per-use basis.

## 4.2 Platform as a Service (PaaS)

Platform-as-a-Service (PaaS) provides a cloud computing framework for software application creation and deployment. It is a platform for the deployment and management of software apps. This flexible cloud computing model scales up automatically on demand. It also manages the servers, storage, and networking, while the developers manage only the application part. It offers a runtime environment for application development and deployment tools.

This Model provides all the facilities required to support the complex life cycle of building and delivering web applications and services entirely for the Internet. This cloud computing model enables developers to rapidly develop, run, and manage their apps without building and maintaining the infrastructure or platform.



## Characteristics of PaaS

- Builds on virtualization technology, so computing resources can easily be scaled up (Auto-scale) or down according to the organization's needs.
- Support multiple programming languages and frameworks.
- Integrates with web services and databases.



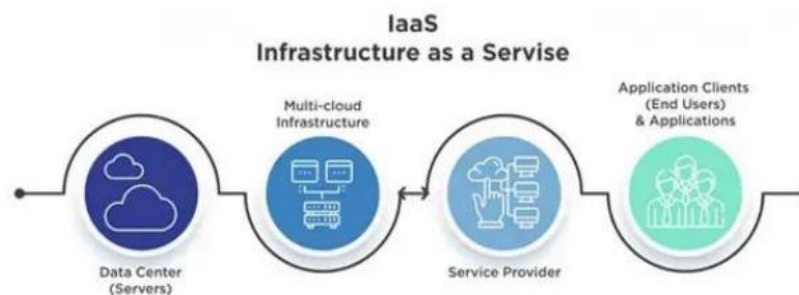
**Here are essential things you need to consider before PaaS implementation:**

- Analyze your business needs, decide the automation levels, and decides whether you want a self-service or fully automated PaaS model.
- You need to determine whether to deploy on a private or public cloud.
- Plan through the customization and efficiency levels.

### **4.3 Infrastructure as a Service (IaaS)**

Infrastructure-as-a-Service (IaaS) is a cloud computing service offering on-demand computing, storage, and networking resources. It usually works on a pay-as-you-go basis. Organizations can purchase resources on-demand and as needed instead of buying the hardware outright.

The IaaS cloud vendor hosts the infrastructure components, including the on-premises data center, servers, storage, networking hardware, and the hypervisor (virtualization layer)



This Model contains the basic building blocks for your web application. It provides complete control over the hardware that runs your application (storage, servers, VMs, networks & operating systems). IaaS model gives you the best flexibility and management control over your IT resources.

#### **Characteristics of IaaS**

There are the following characteristics of IaaS:




- Resources are available as a service
- Services are highly scalable
- Dynamic and flexible Cloud Service Model
- GUI and API-based access
- Automate the administrative tasks

#### **Consider Before IaaS Implementation**

Here are some specific considerations you should remember before IaaS Implementation:

- You should clearly define your access needs and your network's bandwidth to facilitate smooth implementation and functioning.
- Plan out detailed data storage and security strategy to streamline the business process.

- Ensure that your organization has a proper disaster recovery plan to keep your data safe and accessible.

Service Class	Main Access & Management Tool	Service content
 SaaS	Web Browser	<b>Cloud Applications</b> Social networks, Office suites, CRM, Video processing
 PaaS	Cloud Development Environment	<b>Cloud Platform</b> Programming languages, Frameworks, Mashups editors, Structured data
 IaaS	Virtual Infrastructure Manager	<b>Cloud Infrastructure</b> Compute Servers, Data Storage, Firewall, Load Balancer

17

#### 4.4 Some essential criteria for selecting the best cloud service provider

- ✓ **Financial stability:** Look for a well-financed cloud provider that has steady profits from the infrastructure. If the company shuts down because of monetary issues, your solutions will also be in jeopardy.
- ✓ **Industries that prefer the solution:** Before finalizing cloud services, examine its existing clients and markets. Your cloud service provider should be popular among companies in your niche or neighboring ones.
- ✓ **Datacenter locations:** To avoid safety risks, ensure that cloud providers enable your data's geographical distribution.
- ✓ **Encryption standards:** You should make sure the cloud provider supports major encryption algorithms.
- ✓ **Check accreditation and auditing:** The widely used online auditing standard is SSAE (Statement on Standards for Attestation Engagements). This procedure helps you to verify the safety of online data storage. ISO 27001 certificate verifies that a cloud provider complies with international safety standards for data storage.
- ✓ **Backup:** The provider should support incremental backups so that you can store offsite and quickly restore.



## UNIT 5. Big Data Analytics in Cloud Computing

Big Data Analytics uses advanced analytical methods that can extract important business insights from bulk datasets. Within these datasets lies both structured (organized) and unstructured (unorganized) data. Its applications cover different industries such as healthcare, education, insurance, AI, retail, and manufacturing. This unit will discuss in greater detail the concept of big data analytics and how it impacts the decision-making process in many parts of the corporate world.

### 5.1 Introduction to Big-Data Analytics?

Big Data Analytics is all about processing massive amounts of information to uncover hidden trends, patterns, and relationships. It is the process of examining large and complex data sets known as big data to uncover hidden patterns, correlations, trends, and other valuable insights that can support decision-making.

**Big Data** is a collection of data that is huge in volume yet growing exponentially with time. It is a data with so large size and complexity that none of traditional data management tools can store it or process it efficiently. Big data is also a data but with huge size.

Big Data Analytics is a powerful tool which helps to find the potential of large and complex datasets.

### 5V's of Big Data

#### 1. Velocity

Velocity refers to how quickly data is generated and how fast it moves. This is an important aspect for organizations that need their data to flow quickly, so it's available at the right times to make the best business decisions possible.

An organization that uses big data will have a large and continuous flow of data that's being created and sent to its end destination. Data could flow from sources such as machines, networks, smartphones or social media. Velocity applies to the speed at which this information arrives -- for example, how many social media posts per day are ingested -- as well as the speed at which it needs to be digested and analyzed -- often quickly and sometimes in near real time.

**As an example**, in healthcare, many medical devices today are designed to monitor patients and collect data. From in-hospital medical equipment to wearable devices, collected data needs to be sent to its destination and analyzed quickly.

In some cases, however, it might be better to have a limited set of collected data than to collect more data than an organization can handle -- because this can lead to slower data velocities.

#### 2. Volume

Volume refers to the amount of data that exists. Volume is like the base of big data, as it's the initial size and amount of data that's collected. If the volume of data is large enough, it can be considered big data. However, what's considered to be big data is relative and will change depending on the available computing power that's on the market.



**For example**, a company that operates hundreds of stores across several states generates millions of transactions per day. This qualifies as big data, and the average number of total transactions per day across stores represents its volume.

### 3. Value

Value refers to the benefits that big data can provide, and it relates directly to what organizations can do with that collected data. Being able to pull value from big data is a requirement, as the value of big data increases significantly depending on the insights that can be gained from it.

Organizations can use big data tools to gather and analyze the data, but how they derive value from that data should be unique to them. Tools like Apache Hadoop can help organizations store, clean and rapidly process this massive amount of data.

A great example of big data value can be found in the gathering of individual customer data. When a company can profile its customers, it can personalize their experience in marketing and sales improving the efficiency of contacts and garnering greater customer satisfaction.

### 4. Variety

Variety refers to the diversity of data types. An organization might obtain data from several data sources, which might vary in value. Data can come from sources in and outside an enterprise as well. The challenge in variety concerns the standardization and distribution of all data being collected.

As noted above, collected data can be unstructured, semi-structured or structured.

- **Unstructured data** is data that's unorganized and comes in different files or formats. Typically, unstructured data isn't a good fit for a mainstream relational database because it doesn't fit into conventional data models.
- **Semi-structured data** is data that hasn't been organized into a specialized repository but has associated information, such as metadata. This makes it easier to process than unstructured data.
- **Structured data** is data that has been organized into a formatted repository. This means the data is made more addressable for effective data processing and analysis.

Raw data also qualifies as a data type. While raw data can fall into other categories -- structured, semi-structured or unstructured -- it's considered *raw* if it has received no processing at all. Most often, raw applies to data imported from other organizations or submitted or entered by users. Social media data often falls into this category.

**A more specific example** could be found in a company that gathers a variety of data about its customers. This can include structured data called from transactions or unstructured social media posts and call center text. Much of this might arrive in the form of raw data, requiring cleaning before processing.

### 5. Veracity

Veracity refers to the quality, accuracy, integrity and credibility of data. Gathered data could have missing pieces, might be inaccurate or might not be able to provide real, valuable insight. Veracity, overall, refers to the level of trust there is in the collected data.



Data can sometimes become messy and difficult to use. A large amount of data can cause more confusion than insights if it's incomplete. For example, in the medical field, if data about what drugs a patient is taking is incomplete, the patient's life could be endangered.

Both value and veracity help define the quality and insights gathered from data. Thresholds for the truth of data often -- and should -- exist in an organization at the executive level, to determine whether it's suitable for high-level decision-making.

Where might a red flag appear over the veracity data? It could, for instance, be lacking in proper data lineage -- that is, a verifiable trace of its origins and movement.

## 5.2 What Big Data Analytics involves:

- **Collecting Data:** Such data is coming from various sources such as social media, web traffic, sensors and customer reviews.
- **Cleaning the Data:** Imagine having to assess a pile of rocks that included some gold pieces in it. You would have to clean the dirt and the debris first. When data is being cleaned, mistakes must be fixed, duplicates must be removed, and the data must be formatted properly.
- **Analyzing the Data:** It is here that the wizardry takes place. Data analysts employ powerful tools and techniques to discover patterns and trends. It is the same thing as looking for a specific pattern in all those rocks that you sorted through.

## 5.3 Benefits of Big Data Analytics

Big Data Analytics offers a host of real-world advantages, and let's understand with examples:

1. **Informed Decisions:** Imagine a store like Walmart. Big Data Analytics helps them make smart choices about what products to stock. This not only reduces waste but also keeps customers happy and profits high.
2. **Enhanced Customer Experiences:** Think about Amazon. Big Data Analytics is what makes those product suggestions so accurate. It's like having a personal shopper who knows your taste and helps you find what you want.
3. **Fraud Detection:** Credit card companies, like MasterCard, use Big Data Analytics to catch and stop fraudulent transactions. It's like having a guardian that watches over your money and keeps it safe.
4. **Optimized Logistics:** FedEx, for example, uses Big Data Analytics to deliver your packages faster and with less impact on the environment. It's like taking the fastest route to your destination while also being kind to the planet.





## 5.4 Challenges of Big data analytics in cloud computing

While Big Data Analytics offers incredible benefits, it also comes with its set of challenges:

- **Data Overload:** Consider Twitter, where approximately 6,000 tweets are posted every second. The challenge is sifting through this avalanche of data to find valuable insights.
- **Data Quality:** If the input data is inaccurate or incomplete, the insights generated by Big Data Analytics can be flawed. For example, incorrect sensor readings could lead to wrong conclusions in weather forecasting.
- **Privacy Concerns:** With the vast amount of personal data used, like in Facebook's ad targeting, there's a fine line between providing personalized experiences and infringing on privacy.
- **Security Risks:** With cyber threats increasing, safeguarding sensitive data becomes crucial. For instance, banks use Big Data Analytics to detect fraudulent activities, but they must also protect this information from breaches.
- **Costs:** Implementing and maintaining Big Data Analytics systems can be expensive. Airlines like Delta use analytics to optimize flight schedules, but they need to ensure that the benefits outweigh the costs.

## 5.5 Use of Big Data Analytics

Big Data Analytics has a significant impact in various sectors:

- **Healthcare:** It aids in precise diagnoses and disease prediction, elevating patient care.
- **Retail:** Amazon's use of Big Data Analytics offers personalized product recommendations based on your shopping history, creating a more tailored and enjoyable shopping experience.
- **Finance:** Credit card companies such as Visa rely on Big Data Analytics to swiftly identify and prevent fraudulent transactions, ensuring the safety of your financial assets.
- **Transportation:** Companies like Uber use Big Data Analytics to optimize drivers' routes and predict demand, reducing wait times and improving overall transportation experiences.
- **Agriculture:** Farmers make informed decisions, boosting crop yields while conserving resources.
- **Manufacturing:** Companies like General Electric (GE) use Big Data Analytics to predict machinery maintenance needs, reducing downtime and enhancing operational efficiency.

## 5.6 Big Data Analytics Technologies and Tools

Big Data Analytics relies on various technologies and tools that might sound complex, let's simplify them:

- **Hadoop:** Think of Hadoop as an enormous digital warehouse. It's used by big companies like Amazon, Microsoft etc to store tons of data efficiently. For instance, when Amazon suggests products you might like, it's because Hadoop helps manage your shopping history.
- **Spark:** Think of Spark as the super-fast data chef. Netflix uses it to quickly analyze what you watch and recommend your next binge-worthy show.



- **NoSQL Databases:** NoSQL databases, like MongoDB, are like digital filing cabinets that Airbnb uses to store your booking details and user data. These databases are famous because of their quick and flexible, so the platform can provide you with the right information when you need it.
- **Tableau:** Tableau is like an artist that turns data into beautiful pictures. The World Bank uses it to create interactive charts and graphs that help people understand complex economic data.

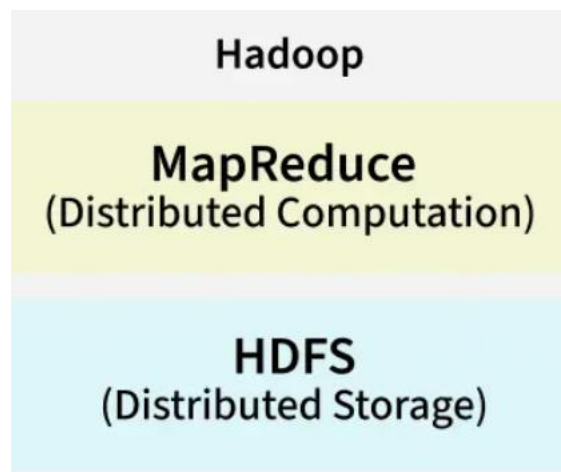
### 5.6.1 Introduction to Hadoop

Hadoop is an open-source software framework that is used for storing and processing large amounts of data in a distributed computing environment. It is designed to handle big data and is based on the MapReduce programming model, which allows for the parallel processing of large datasets. Hadoop is designed to process large volumes of data (Big Data) across many machines without relying on a single machine. It is built to be scalable, fault-tolerant and cost-effective.

#### Hadoop Architecture

Hadoop has two main components:

- **Hadoop Distributed File System (HDFS):** HDFS breaks big files into blocks and spreads them across a cluster of machines. This ensures data is replicated, fault-tolerant and easily accessible even if some machines fail.
- **MapReduce:** MapReduce is the computing engine that processes data in a distributed manner. It splits large tasks into smaller chunks (map) and then merges the results (reduce), allowing Hadoop to quickly process massive datasets.



#### ➤ Hadoop Distributed File System (HDFS)

HDFS is the storage layer of Hadoop. It breaks large files into smaller blocks (usually 128 MB or 256 MB) and stores them across multiple Data Nodes. Each block is replicated (usually 3 times) to ensure fault tolerance so even if a node fails, the data remains available.

##### Key features of HDFS:

- **Scalability:** Easily add more nodes as data grows.
- **Reliability:** Data is replicated to avoid loss.



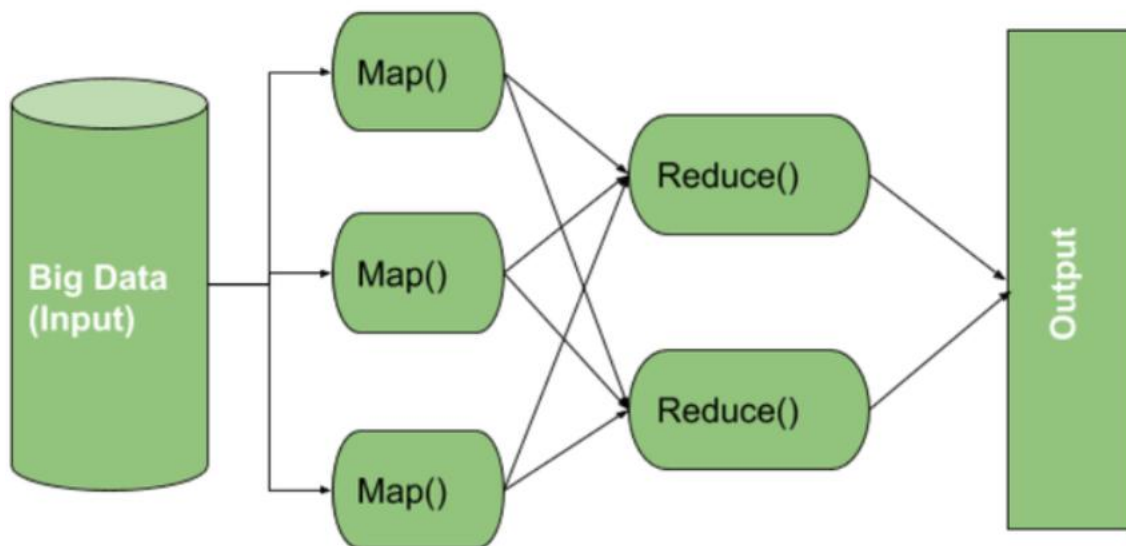


- **High Throughput:** Designed for fast data access and transfer.

### ➤ MapReduce

MapReduce is the computation layer in Hadoop. It works in two main phases:

1. **Map Phase:** Input data is divided into chunks and processed in parallel. Each mapper processes a chunk and produces key-value pairs.  
As the name suggests its main use is to map the input data in key-value pairs. The input to the map may be a key-value pair where the key can be the id of some kind of address and value is the actual value that it keeps. The Map() function will be executed in its memory repository on each of these input key-value pairs and generates the intermediate key-value pair which works as input for the Reducer or Reduce() function.
2. **Reduce Phase:** These key-value pairs are then grouped and combined to generate results.  
The intermediate key-value pairs that work as input for Reducer are shuffled and sort and send to the Reduce() function. Reducer aggregate or group the data based on its key-value pair as per the reducer algorithm written by the developer.



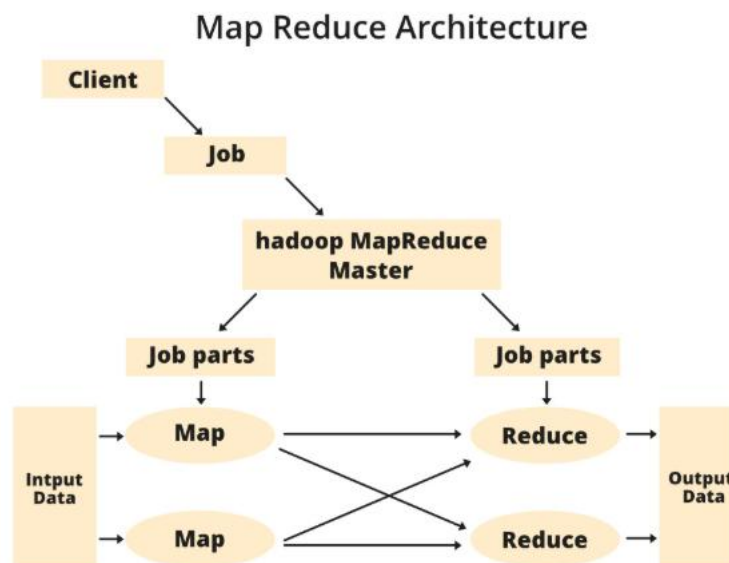
### 5.6.2 Google File System (GFS) vs. Hadoop Distributed File System (HDFS)

In distributed file systems, Google File System (GFS) and Hadoop Distributed File System (HDFS) stand out as crucial technologies. Both are designed to handle large-scale data, but they cater to different needs and environments.

Google File System(GFS)	Hadoop Distributed File System (HDFS)
<ul style="list-style-type: none"> <li>Developed by Google for their internal applications.</li> <li>Default chunk size of 64 MB.</li> <li>Optimized for write-once, read-many access patterns.</li> <li>Focus on computation close to data for efficiency.</li> </ul>	<ul style="list-style-type: none"> <li>Developed by Apache for open-source big data frameworks.</li> <li>Default block size of 128 MB (configure).</li> <li>Also optimized for write-once, read-many workloads.</li> <li>Provides data locality by moving computation to where the data is stored.</li> </ul>

### 5.6.3 MapReduce Architecture

MapReduce and HDFS are the two major components of Hadoop which makes it so powerful and efficient to use. MapReduce is a programming model used for efficient processing in parallel over large datasets in a distributed manner. The data is first split and then combined to produce the final result. The library for MapReduce is written in so many programming languages with various different-different optimizations. The purpose of MapReduce in Hadoop is to Map each of the jobs and then it will reduce it to equivalent tasks for providing less overhead over the cluster network and to reduce the processing power.



### 5.6.4 Components of MapReduce Architecture:

1. **Client:** The MapReduce client is the one who brings the Job to the MapReduce for processing. There can be multiple clients available that continuously send jobs for processing to the Hadoop MapReduce Manager.
2. **Job:** The MapReduce Job is the actual work that the client wanted to do which is comprised of so many smaller tasks that the client wants to process or execute.



3. **Hadoop MapReduce Master:** It divides the particular job into subsequent job-parts.
4. **Job-Parts:** The task or sub-jobs that are obtained after dividing the main job. The result of all the job-parts combined to produce the final output.
5. **Input Data:** The data set that is fed to the MapReduce for processing.
6. **Output Data:** The final result is obtained after the processing.

## UNIT 6. Robust Distributed Systems and Supercomputing Foundations in Cloud Environments

Robustness testing is a quality assurance methodology focused on testing the robustness of the software and helps in removing the reliability of the software by finding the corner cases by inputting the data that mimics the extreme environmental conditions to determine whether or not the system is robust enough to deliver the required functionality to the user.

### What is Robustness Testing?

Robustness testing is a type of testing that is performed to assess the ability of a system or component to function correctly when it is subjected to invalid or unexpected inputs, or when it is operating outside of its specified operating conditions. It is typically used to test for the presence of memory leaks or other types of errors that can cause a system to crash. Robustness testing is also sometimes referred to as reliability testing, stress testing, or endurance testing.

- The purpose of robustness testing is to identify the parts of the system that are most vulnerable to failure and to determine how the system can be made more resistant to failure.
- Robustness testing is typically conducted by subjecting the system to a variety of stressful conditions, such as high temperatures, high humidity, high pressure, and high levels of vibration.
- Robustness testing is typically conducted during the later stages of software testing after the software has been shown to work correctly under normal conditions.
- A common example of robustness testing is testing how a system responds to unexpected input values.
- For example, if a system is designed to accept numerical input values between 1 and 10, a robustness test would involve trying to input values outside of this range, such as 0, 11, or -5, to see how the system responds.
- Another example of robustness testing would be testing how a system responds to unexpected environmental conditions, such as excessive heat, cold, or humidity.

### 6.1 Why is Robustness Testing important?

- **Handle Unexpected Inputs:** Robustness testing is important because it helps ensure that a system can handle unexpected or abnormal inputs without crashing.



- **Uncover Potential Issues:** This type of testing can help uncover potential issues that could cause a system to fail in unexpected ways. By uncovering these issues early on, they can be fixed before the system is put into production.
- **Test Limits:** It allows developers to test the limits of their software and ensure that it can handle unexpected inputs and situations.
- **Uncover Hidden Bugs:** This type of testing can help uncover hidden bugs that could cause major problems in the field.
- **Stability:** This can help to prevent software failures and crashes and can also help to improve the overall stability of your software.

## 6.2 Types of Testing to ensure Robustness of Test Suites

There are many types of testing that can ensure the robustness of test suites. Below are some of the testing types that ensure the robustness of test suites:

- **Regression testing:** It's a type of testing that is used to find bugs in software that have already been fixed. This is done by running the software with different inputs and comparing the output to the expected output. This type of testing is used to verify that a software program continues to function properly after it has been modified or updated. This type of testing is typically performed after a new version of the software has been released, or after a change has been made to the code.
- **Functional testing:** It's a type of testing that is used to verify that a system or software performs as expected. Load testing is a type of testing that is used to verify that a system or software can handle a heavy load or traffic. This type of testing is typically performed by running the software through a series of tests that exercise the various functions of the software.
- **Load testing:** It's a type of testing that is used to find bugs in software by running it with different inputs and checking if the output is as expected. It is used to verify that a software program is able to handle the load that is expected to be placed on it. This type of testing is typically done by running the software through a series of tests that simulate the load that the software will experience in production.
- **Negative testing:** Negative testing involves deliberately providing invalid or incorrect inputs to a system in order to see how it responds. This can help to uncover errors in input validation or handling that could lead to security vulnerabilities or data loss.
- **Fuzz testing:** It's a software testing technique, usually automated or semi-automated, that involves providing invalid, unexpected, or random data to the inputs of a computer program. The program is then monitored for exceptions such as crashes, failing built-in code assertions, or potential memory leaks. Fuzz testing is effective in finding coding errors and security vulnerabilities.
- **Use case testing:** It involves testing the system with realistic scenarios to see how it responds. This can help to identify potential usability issues.



- **Security testing:** Involves testing the system for security vulnerabilities. This can help to identify potential security risks.
- **Black-box testing:** It's a method of software testing that examines the functionality of a software program without knowing the internal code structure. The tester is only aware of what the software is supposed to do but not how it does it. Black-box testing can be used to test the functionality of a software program, the usability of a user interface, and the compliance of a program with external standards.
- **Mutation testing:** It's a type of software testing that involves modifying a program's source code or its inputs and then testing to see if the program still behaves as expected. The goal of mutation testing is to find faults in a program's code or inputs that can cause the program to produce incorrect results.
- **Fault injection testing:** It's a method of testing software by introducing faults into the software program to see if the program can detect and handle the faults. Fault injection can be used to test the robustness of a program's error-handling capabilities.

### 6.3 Advantages of Robustness Testing

- **Reduced false positives:** By testing for a wider range of conditions, false positives can be reduced, saving time and resources.
- **Improved test accuracy:** Robustness testing can help to ensure that test cases are more accurate, leading to improved quality assurance.
- **Greater flexibility:** Robustness testing can be used to test for a wide range of conditions, making it more flexible than other types of testing.
- **Increased efficiency:** Robustness testing can save time and resources by allowing for increased test coverage in a shorter amount of time.
- **Improved software quality:** By identifying and addressing errors early on in the development process, robustness testing can help improve the overall quality of the software.
- **Reduced development costs:** By catching errors early, robustness testing can save on development costs by avoiding the need for expensive rework or redesign.
- **Increased customer satisfaction:** By ensuring that the software meets the customer's expectations in terms of functionality and reliability, robustness testing can help increase customer satisfaction.

### 6.4 High performance Computing (Supercomputing)

It is the use of parallel processing for running advanced application programs efficiently, relatively, and quickly. The term High-performance computing is occasionally used as a synonym for supercomputing. Although technically a supercomputer is a system that performs at or near currently highest operational rate for computers. Some supercomputers work at more than a petaflop ( $10^{12}$ ). The most common HPC system all scientific engineers & academic institutions. Some Government





agencies particularly military are also relying on APC (Advanced Process Control) for complex applications.

High Performance Computing (HPC) generally refers to the practice of combining computing power to deliver far greater performance than a typical desktop or workstation, in order to solve complex problems in science, engineering, and business. High Performance Computing (HPC) generally refers to the practice of combining computing power to deliver far greater performance than a typical desktop or workstation, in order to solve complex problems in science, engineering, and business.

#### 6.4.1 Importance of High-performance Computing:

1. It is used for scientific discoveries, game-changing innovations, and to improve quality of life.
2. It is a foundation for scientific & industrial advancements.
3. It is used in technologies like IoT, AI, 3D imaging evolves & amount of data that is used by organization is increasing exponentially to increase ability of a computer, we use High-performance computer.
4. HPC is used to solve complex modeling problems in a spectrum of disciplines. It includes AI, Nuclear Physics, Climate Modelling, etc.
5. HPC is applied to business uses, data warehouses & transaction processing.

#### 6.4.2 Why do we Need of High-performance Computing?

1. It will complete a time-consuming operation in less time.
2. It will complete an operation under a light deadline and perform a high numbers of operations per second.
3. It is fast computing, we can compute in parallel over lot of computation elements CPU, GPU, etc. It set up very fast network to connect between elements.

#### 6.4.3 Challenges with HPC

1. **Cost:** The cost of the hardware, software, and energy consumption is enormous, making HPC systems exceedingly expensive to create and operate. Additionally, the setup and management of HPC systems require qualified workers, which raises the overall cost.
2. **Data Management:** Data management can be difficult when using HPC systems since they produce and process enormous volumes of data. These data must be stored and accessed using sophisticated networking and storage infrastructure, as well as tools for data analysis and visualization.
3. **Programming:** Parallel programming techniques, which can be more difficult than conventional programming approaches, are frequently used in HPC systems. It might be challenging for developers to learn how to create and optimize algorithms for parallel processing.



4. **Support for software and tools:** To function effectively, HPC systems need specific software and tools. The options available to users may be constrained by the fact that not all software and tools are created to function with HPC equipment.
5. **Power consumption and cooling:** To maintain the hardware functioning at its best, specialized cooling technologies are needed for HPC systems' high heat production. Furthermore, HPC systems consume a lot of electricity, which can be expensive and difficult to maintain.

#### 6.4.4 Applications of HPC

High Performance Computing (HPC) is a term used to describe the use of supercomputers and parallel processing strategies to carry out difficult calculations and data analysis activities. From scientific research to engineering and industrial design, HPC is employed in a wide range of disciplines and applications. Here are a few of the most significant HPC use cases and applications:

1. **Scientific research:** HPC is widely utilized in this sector, especially in areas like physics, chemistry, and astronomy. With standard computer techniques, it would be hard to model complex physical events, examine massive data sets, or carry out sophisticated calculations.
2. **Weather forecasting:** The task of forecasting the weather is difficult and data-intensive, requiring sophisticated algorithms and a lot of computational power. Simulated weather models are executed on HPC computers to predict weather patterns.
3. **Healthcare:** HPC is being used more and more in the medical field for activities like medication discovery, genome sequencing, and image analysis. Large volumes of medical data can be processed by HPC systems rapidly and accurately, improving patient diagnosis and care.
4. **Energy and environmental studies:** HPC is employed to simulate and model complex systems, such as climate change and renewable energy sources, in the energy and environmental sciences. Researchers can use HPC systems to streamline energy systems, cut carbon emissions, and increase the resilience of our energy infrastructure.
5. **Engineering and Design:** HPC is used in engineering and design to model and evaluate complex systems, like those found in vehicles, buildings, and airplanes. Virtual simulations performed by HPC systems can assist engineers in identifying potential problems and improving designs before they are built.