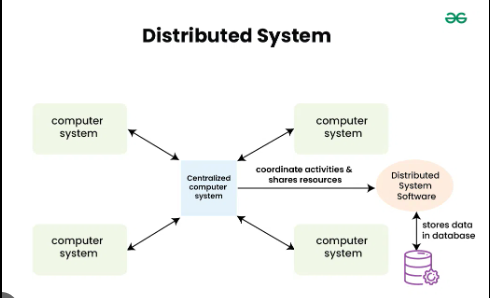
**Distributed System**

* A distributed System is a collection of autonomous computer systems that are physically separated but are connected by a centralized computer network that is equipped with distributed system software.
* These are used in numerous applications, such as online gaming, web applications, and cloud computing.
* These are used in numerous applications, such as online gaming, web applications, and cloud computing. However, creating a distributed system is not simple, and there are a number of design considerations to take into account

**Diagram:**



**Design Issues of Distributed System :**

1. **Scalability in Distributed Systems:**

***Challenges:***

* ***Handling Increased Load****: The system must accommodate more users or requests without experiencing performance degradation.*
* ***Geographic Distribution****: Maintaining low latency and high performance for users across different regions.*
* ***Strategies to Achieve Scalability:***
* ***Horizontal Scaling****: Add more nodes or machines to distribute the workload.*
* ***Vertical Scaling****:* Increase the resources (CPU, RAM, storage) of existing nodes.

1. **Reliability in Distributed Systems:**

***Fault Tolerance:***

* ***Redundancy****: Deploy duplicate components or systems to take over in case of failure, minimizing service disruption.*
* ***Failover Mechanisms****: Automatically switch to a standby system or component when the primary system experiences a failure.*
* ***Redundancy and Replication:***
* ***Data Replication****: Store multiple copies of data across different nodes to ensure data availability and prevent data loss.*
* ***Consensus Algorithms****: Use protocols like* ***Paxos*** *or* ***Raft*** *to maintain consistency and agreement among replicated data in the system.*

**3)Consistency in Distributed Systems:**

***Data Consistency Models:***

* ***Strong Consistency****: Guarantees that all nodes have the same data at the same time after any update, ensuring immediate consistency.*
* ***Eventual Consistency****: Allows temporary discrepancies between nodes, with the assurance that they will converge to a consistent state over time.*

***Trade-offs between Consistency and Availability (CAP Theorem)***

**CAP Theorem**:

States that in the presence of a network partition, a distributed system can provide only two out of three guarantees:

* **Consistency**: All nodes see the same data.
* **Availability**: Every request gets a response (success or failure).
* **Partition Tolerance**: The system continues to operate despite network failures.

**4)Latency in Distributed Systems:**

***Sources of Latency:***

* ***Network Delays****: The time required for data to travel between nodes over the network.*
* ***Processing Delays****: The time taken by servers or nodes to process and respond to requests.*

***Minimization Techniques:***

* **Caching**: Store frequently accessed data closer to users (e.g., in-memory or edge caches) to reduce retrieval time.
* **Data Compression**: Decrease the size of data transferred, reducing transmission time over the network.

**5)Load Balancing in Distributed Systems:**

***Load Distribution Methods:***

* ***Round Robin****: Distributes incoming requests evenly across all available servers in sequential order.*
* ***Least Connections****: Directs traffic to the server with the fewest active connections to balance load dynamically.*

***Dynamic vs. Static Load Balancing:***

* **Dynamic Load Balancing**: Adjusts distribution in real-time based on current server loads and resource availability.
* **Static Load Balancing**: Follows predefined rules or configurations for distributing load, regardless of real-time changes.