Introduction to Distributed System

A distributed system is a system of multiple nodes that are physically separated but linked together using the network.

Each of these nodes includes a small amount of the distributed operating system software.

Every node in this system communicates and shares resources with each other and handles processes in a team.

(or)

A distributed system is a collection of independent computers that appears to its users as a single coherent system.

(or)

A distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved.

Why make a system distributed?

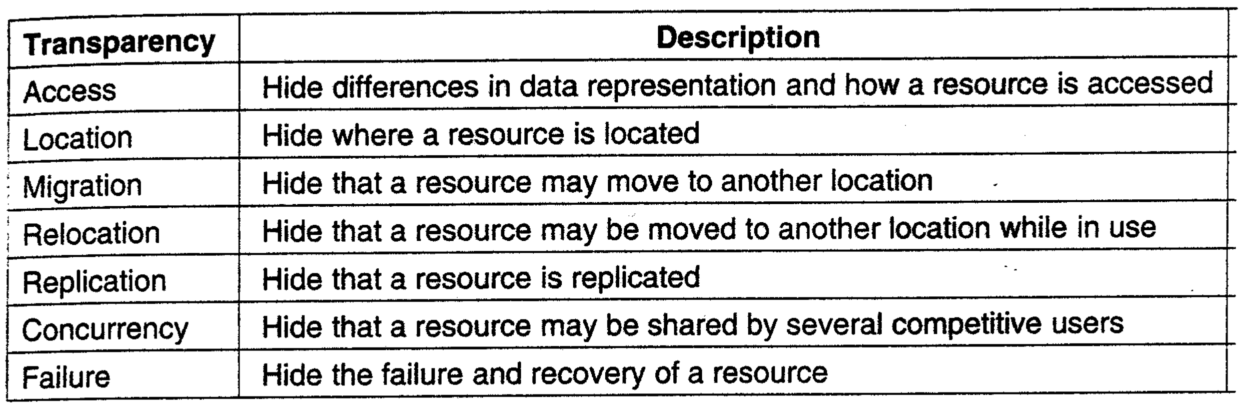
- It's inherently distributed:
  e.g. sending a message from your mobile phone to your
  friend's phone
- For better reliability:
  even if one node fails, the system as a whole keeps functioning
- For better performance:
  get data from a nearby node rather than one halfway round the world
- To solve bigger problems:
  e.g. huge amounts of data, can't fit on one machine

Characteristics of DS

- **Resource Sharing :** It is the ability to use any Hardware, Software, or Data anywhere in the System.

- **Openness :** (i.e., How openly the software is developed and shared with others)
- **Concurrency** : It is naturally present in Distributed Systems, that deal with the same activity or functionality that can be performed by separate users who are in remote locations
- **Scalability:** It increases the scale of the system as a number of processors communicate with more users by accommodating to improve the responsiveness of the system.
- **Fault tolerance:** It cares about the reliability of the system if there is a failure in Hardware or Software, the system continues to operate properly without degrading the performance the system.
- **Transparency** : It hides the complexity of the Distributed Systems to the Users and Application programs as there should be
privacy in every system.

Different types of transparency



Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Disadvantages of DS

- Relevant Software for Distributed systems does not exist currently.
- Security possess a problem due to easy access to data as the resources are shared to multiple systems
- Networking Saturation may cause a hurdle in data transfer i.e., if there is a lag in the network then the user will face a problem accessing data.

- In comparison to a single user system, the database associated with distributed systems is much more complex and challenging to manage.
- If every node in a distributed system tries to send data at once, the network may become overloaded.

Use cases of Distributed System

- **Finance and Commerce:** Amazon, eBay, Online Banking, E-Commerce websites.
- **Information Society:** Search Engines, Wikipedia, Social Networking, Cloud Computing.
- **Cloud Technologies:** AWS, Salesforce, Microsoft Azure, SAP.
- **Entertainment:** Online Gaming, Music, youtube.
- **Healthcare:** Online patient records, Health Informatics.
- **Education:** E-learning.
- **Transport and logistics:** GPS, Google Maps.
- **Environment Management:** Sensor technologies.

Challenges/Cons of DS

Distributed systems (DS) offer many benefits, such as scalability, fault tolerance, and improved performance. However, they also come with several disadvantages and challenges:

1. Complexity:

   System Design: Designing, developing, and maintaining a distributed system is inherently more complex than a centralized system. It involves handling issues like data distribution, synchronization, and communication between nodes.
   Debugging and Testing: Debugging distributed systems can be difficult because bugs may not always be reproducible due to the asynchronous

nature of operations, varying network conditions, and node failures. Testing all possible interactions and failures is also challenging.

2. Network Dependency:

   Latency: The performance of a distributed system heavily depends on network latency. Slow or unreliable network connections can lead to delays, affecting the overall performance of the system.
   Partitioning: Network partitioning, where parts of the system become isolated due to network failures, can lead to inconsistencies, unavailability, or even data loss.

3. Consistency Issues:

   Data Consistency: Ensuring data consistency across distributed nodes is challenging, especially in the face of network partitions or node failures. The CAP theorem highlights the trade-off between consistency, availability, and partition tolerance in distributed systems.
   Event Ordering: In distributed systems, ensuring that events occur in a consistent and correct order across different nodes can be complex, especially when physical clocks are not perfectly synchronized.

4. Fault Tolerance:

   Handling Failures: While distributed systems are designed to be fault-tolerant, handling and recovering from failures is complex. This includes detecting failures, implementing redundancy, and ensuring data is not lost or corrupted.
   Data Recovery: If a node fails, recovering the lost data or ensuring that the system can continue functioning without that node can be difficult.

5. Security Concerns:

   Data Security: Distributed systems often involve multiple nodes communicating over a network, which can expose them to security threats such as eavesdropping, man-in-the-middle attacks, or unauthorized access.
   Authentication and Authorization: Managing and securing authentication and authorization across a distributed system is more challenging compared to centralized systems.

6. Coordination and Synchronization:

Synchronization: Ensuring that all nodes are synchronized, especially in terms of data and state, requires sophisticated algorithms and can introduce overhead.
Coordination Overhead: Coordinating tasks among multiple nodes requires additional communication and processing, which can reduce the overall efficiency of the system.

7. Resource Management:

   Resource Allocation: Efficiently managing and allocating resources across different nodes can be challenging. Overloading one node while others are underutilized can lead to performance bottlenecks.
   Load Balancing: Ensuring that the load is evenly distributed among nodes to avoid overloading certain parts of the system requires careful planning and implementation.

8. Deployment and Upgrades:

   Complex Deployment: Deploying a distributed system requires setting up and configuring multiple nodes, often in different locations, which can be time-consuming and prone to errors.
   Rolling Updates: Upgrading software or hardware in a distributed system without causing downtime or inconsistencies is complex and requires careful coordination.

9. Cost:

   Infrastructure Costs: Distributed systems often require more hardware, networking equipment, and maintenance compared to centralized systems, leading to higher operational costs.
   Management Costs: The complexity of managing a distributed system can lead to higher costs in terms of staffing, training, and maintenance.