DISTRIBUTED SYSTEM MODELS

TYPES OF DISTRIBUTED SYSTEM MODELS (ARCHITECTURE OF DISTRIBUTED SYSTEMS)

- It describes the architecture, behaviour and design principles of distributed system.
- Each model has own strengths, weakness and use cases. The choice of distributed model depends on the specific requirements and constraints of the distributed system.
- Below are the commonly used models in distributed system.

1. Client-Server Model (Client Server Architecture)

- Here clients request the services and servers provide them.
- It divides the tasks between service providers(servers) and requesters (clients)
- It is centralized architecture which means that single of point of a control which is server.
- **Examples**
  - Web applications, Email services
- Use cases – Web browsing, database access.

Server Role

- The server is main controller in the client server architecture. It has the authority to make decisions about:
  - Resource allocation
  - Data access
  - Service provision
  - Authentication and authorization.

Client Role

- It requests services or resources from the server.
- It follows the server's instructions and protocols.
- Clients depend on the server for services and resources.

Summary

- The server is the main controller and decision maker in a client server model, making it a centralized system.

Characteristics

- Centralized server processes all client requests (A centralized server will provide services to multiple clients).

Advantages

- Centralized control simplifies management.
- Easy to deploy and maintain.

Disadvantages

- Single point of failure at the server.
- Scalability issues with a high number of clients.

2. Peer-to-Peer Model (Peer -to-Architecture) (P2P)

- It is decentralized architecture which means that multiple points of control.
- Here all the nodes (computers) are considered as equal and communicate with each other directly.
- Each node act as both a client and a server (All nodes act as both clients and servers).
- Here resources / services are directly shared between nodes (Peers share resources without depending on a centralized server).
- **Examples**
  - BitTorrent systems, Blockchain.
- Use cases – File sharing, cryptocurrency systems

3. Cluster Computing Model

- A group of connected computers works as a single unit, sharing a high-speed network.
- **Examples**
  - Apache Hadoop, High-Performance Computing (HPC).
- Use Cases – Data analysis, machine learning, simulations.

4. Grid Computing Model

- Popular distributed model for achieving a big task via with the power of multiple independent nodes connected over the network.
- Resources are distributed across multiple administrative domains but work together to solve a single problem.
- **Unlike cloud model, it is decentralized system.**
- **Examples**
  - BOINC (Berkeley Open Infrastructure for Network Computing).
- Use Cases – Scientific research, weather forecasting, genome analysis.

Working Operation of Grid Model

- The grid model involves the following operations. They are
  - Task decomposition
  - Task distribution
  - Parallel processing
  - Data aggregation.

Task decomposition

- Here, a large task is broken down into smaller subtasks.

Task distribution

- In this phase, subtasks are distributed across the grid nodes geographically.

Parallel processing

- Here nodes execute their assigned subtasks concurrently.

Data aggregation

- Finally, results from individual nodes are collected and combined to produce the final output.

Characteristics of grid computing

Fault tolerance

- Grids can continue to operate even if some nodes fail, ensuring high availability.

Resource sharing

- Grids enable the sharing of computing power, storage, and applications across multiple domains.

Heterogeneity

- Grids can incorporate a variety of hardware and software platforms.

Scalability

- Grids can easily scale to accommodate increasing workloads by adding more resources to the network.

Applications of grid computing

Scientific research

- Grids are widely used in scientific fields such as biology, physics, and astronomy for tasks like genome sequencing, drug discovery, and climate modelling.

Business applications

- Grids can be used for data warehousing, financial modelling, and e-commerce.

Education

- Grids can provide access to high-performance computing resources for educational purposes.

5. Cloud Computing Model

- Resources and services are delivered over the internet
- Here resources are dynamically allocated based on the user demands
- It offers scalability and flexibility.
- It is a centralized architecture which means that services are delivered from a centralized cloud service provider(s).

Types

1. IaaS (Infrastructure as a Service) - Provides virtualized computing resources.
2. PaaS (Platform as a Service) - Provides development platforms.
3. SaaS (Software as a Service) - Provides software applications.

- **Examples**
  - Google Cloud, Amazon AWS, Microsoft Azure.
- **Use Cases**
  - Web hosting and application development,
  - Data Storage and backup
  - AI processing
  - Software development and testing
  - Business application and collaboration.

6. Service Oriented Architecture

- Systems are organized as a collection of services that communicate through APIs.
- **Examples**
  - Microservices architecture using REST or gRPC.
- Use Cases – Enterprise applications, modular systems.

7. MapReduce Model

- It is a parallel programming model
- It processes large datasets in parallel by dividing tasks into a map (data distribution) and reduce (aggregation) phase.

- **Examples**
  - Hadoop MapReduce, Apache Spark.
- Use Cases – Big data processing, data analytics.

8. Hybrid Model

- It combines features of different models (Ex. client-server with peer-to-peer).
- **Examples**
  - Content Delivery Networks (CDNs), hybrid cloud systems.
- Use Cases - Video streaming, scalable web services.

9. Three-Tier Model (Three – Tier Architecture)

- Here It divides systems into three layers like presentation, logic, and data layers.
- **Examples**
  - Web-based applications.

10. Multi-Tier Model (Mult-Tier Architecture)

- It extends three-tier systems by adding layers for scalability.
- **Examples**
  - Large enterprise systems.

11. Event-Based Model (Event Based Architecture)

- It uses message queues for asynchronous communication.
- **Examples**
  - Apache Kafka, RabbitMQ.

12. Master Slave Model

- Centralized architecture means that all the nodes are not equal priority. Only master node will control and coordinate slave nodes.
- Here master node is the main controller and slave nodes are worker nodes.
- Slave nodes perform tasks assigned by master node.

13. Distributed Shared Model (DSM)

- Here memory spaces are shared across nodes
- Nodes are shared memory for data exchange.

14. Message Passing Model

- Here nodes are communicated with each other through message passing
- Here memory is not shared.
- Nodes are used to exchange data between them through message.

15. Actor Model

- Nodes (actors) communicate with each other via asynchronous messages
- Actors can create new actors, send messages and find behaviour.

Characteristics

1. Fault tolerance

- The Actor Model provides built-in support for fault tolerance through supervisor hierarchies.

2. Concurrency

- Actors run concurrently, enabling efficient use of system resources.

3.Asynchronous communication

- Actors communicate through asynchronous messages, reducing latency and improving responsiveness.

4. Decoupling

- Actors are decoupled from each other, allowing for greater flexibility and scalability.

Use Cases

1. Distributed systems

- The Actor Model is well-suited for distributed systems, where actors can run on different nodes or machines.

2.Real-time systems

- The Actor Model's asynchronous communication and concurrency features make it suitable for real-time systems.

3. Reactive systems

- The Actor Model's focus on event-driven programming and asynchronous communication aligns well with reactive system principles.

Popular Implementations

1. Akka (Java/Scala)

- A popular Actor Model implementation for building concurrent and distributed systems.

2.Erlang/OTP

- A programming language and framework that provides built-in support for the Actor Model.

3. Actor Model in Python

- Several libraries, such as Pykka and Thespian, provide Actor Model implementations for Python.