Introduction to Distributed and Parallel Computing
CS-401

Dr. Sanjay Saxena
Visiting Faculty, CSE, IIIT Vadodara
Assistant Professor, CSE, IIIT Bhubaneswar
Post doc – University of Pennsylvania, USA
PhD – Indian Institute of Technology(BHU), Varanasi

What is a Distributed System?

- **Definition**: A distributed system is a collection of independent computers that appear to the users as a single coherent system.
- **Characteristics**:
  - Resource sharing
  - Concurrency
  - Fault tolerance

  Examples:

1. Google Docs: Multiple users editing a document simultaneously.
2. Online multiplayer games: Players interact in real-time across the globe.

Goals of Distributed Systems

- **Primary Goals**:
  - **Transparency**: Access, location, migration, replication.
  - **Scalability**: Handle growing demand efficiently.
  - **Reliability**: Ensure fault tolerance and availability.
    Examples:

1.Netflix: Seamless video streaming across devices.

2.Uber: Real-time ride-sharing with location transparency.

Hardware in Distributed Systems

- **Types of Systems**:
  - **Homogeneous Systems**: Identical hardware and software.
  - **Heterogeneous Systems**: Diverse hardware and software platforms.
- **Interconnects**:
  - LAN, WAN, wireless networks.
    Examples:

1.Homogeneous: Raspberry Pi clusters for educational research.

2.Heterogeneous: A mobile app communicating with a cloud server.

Software Concepts

- **Middleware**: Software that acts as a bridge between applications and hardware.
- **Resource Management**: Load balancing, fault detection, and recovery.
  Examples:

1.Kubernetes: Managing containers in distributed systems.

2.Apache Kafka: Distributed event streaming platform.

The Client-Server Model

- **Client**: Sends requests.
- **Server**: Processes requests and sends responses.
- **Advantages**: Centralized control, easier maintenance.
  Examples:

1. Gmail: Client app interacts with Google’s email servers.
2. DNS: Domain name resolution service.

Communication in Distributed Systems

- **Layered Protocols**:
  - Application Layer: HTTP, FTP.
  - Transport Layer: TCP, UDP.
  - Network Layer: IP.
- **Purpose**: Enables interoperability between systems.
  Examples:

1.HTTP for web browsing.

2.UDP for video streaming.

Remote Procedure Call (RPC)

- Allows a program to execute code on a remote server as if it were local.
- **Steps**:
  - Client makes a call.
  - Server executes and sends a response.
    Examples:

1.Microservices communicating in a distributed system.

2.Weather apps fetching data from remote APIs.

Remote Object Invocation

- Extends RPC by invoking methods on remote objects.
- **Techniques**:
  - Java RMI
  - CORBA
    Examples:

1. Distributed database systems.
2. A banking system where client applications interact with remote account objects.

Message-Oriented Communication

- **Asynchronous Communication**: Send and receive messages independently.
- **Synchronous Communication**: Sender waits for acknowledgment.
  Examples:

1. Asynchronous: Email systems.
2. Synchronous: Chat applications like WhatsApp.

Stream-Oriented Communication

- Continuous flow of data.
- Examples: Video streaming, real-time audio.
  Examples:

1. Netflix: Continuous video delivery.
2. Spotify: Real-time audio streaming.

Threads in Processes

- Lightweight processes running within a program.
- **Advantages**: Faster context switching, better resource utilization.
  Examples:

1.Web servers handling multiple requests using threads.

2.Multithreaded gaming applications.

Types of Servers

Content:

- **Iterative Servers**: Handle one request at a time.
- **Concurrent Servers**: Handle multiple requests concurrently.
  Examples:

1.Iterative: Simple HTTP server.

2.Concurrent: Database servers like MySQL.

Code Migration

- Moving code execution to another machine for performance or resource reasons.
- **Types**:
  - Static: Pre-defined.
  - Dynamic: On-the-fly.
    Examples:

1.Java Applets.

2.Cloud-based app deployment.

Software Agents

- Autonomous entities performing tasks on behalf of users.
- **Types**:
  - Mobile agents.
  - Reactive agents.
    Examples:

1. Chatbots like ChatGPT.
2. Virtual assistants like Alexa.

Naming Entities

- **Goals**: Identify and locate resources.
- **Techniques**:
  - DNS for websites.
  - IP addresses for devices.
    Examples:

1.Resolving domain names to IP addresses.

2.Mapping MAC addresses in local networks.

Locating Mobile Entities

- **Challenges**: Tracking moving devices or users.
- **Solutions**: Home agent, foreign agent.
  Examples:

1.Mobile IP protocol.

2.GPS-based apps like Google Maps.

Removing Unreferenced Entities

- Ensures memory efficiency by cleaning unused objects.
- **Garbage Collection**: Automatic cleanup in Java.
  Examples:

1.Java’s JVM handling unused objects.

2.Python’s reference counting.

Clock Synchronization

- **Techniques**:
  - NTP (Network Time Protocol).
  - Logical clocks.
    Examples:

1.NTP syncing all servers to UTC.

2.Google’s Spanner database.

Logical Clocks

- Used to maintain event order.
- **Lamport Timestamps**: Ensures consistency without real-time clocks.
  Examples:

1.Coordinating tasks in distributed systems.

2.Event order in blockchain.

Election Algorithms

- Used to select a leader in distributed systems.
- **Algorithms**:
  - Bully Algorithm.
  - Ring Algorithm.
    Examples:

1.Distributed databases electing a coordinator.

2.Leader selection in distributed chat apps.

Distributed Coordination

- Coordination across nodes for consistency.
- Techniques: Consensus algorithms like Paxos, Raft.
  Examples:

1.Distributed file systems.

2.Consensus in blockchain.

Fault Tolerance

1.Techniques: Replication, redundancy.

Examples:RAID for data redundancy.

1.Cloud providers replicating data across regions.

Security in Distributed Systems

- **Threats**: Eavesdropping, denial of service.
- **Techniques**: Encryption, firewalls.
  Examples:

1. SSL for secure web communication.
2. Authentication tokens in APIs.

Summary

- Distributed systems rely on efficient communication, naming, and synchronization.
- Challenges like fault tolerance and security require robust solutions.
  Examples:

1. Cloud services combining fault tolerance with scalability.
2. Distributed applications like video conferencing ensuring synchronization.

Thanks & Cheers!!

Small aim is a crime; have great aim.
Bharat-Ratan A. P. J. Abdul Kalam