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

Summarization

- Parallel Computing
- Cloud Computing

Parallel Computing

- Parallel computing divides tasks into smaller sub-tasks that are processed simultaneously, improving efficiency and performance.

Example: Multiple chefs in a kitchen preparing different dishes at the same time to complete a meal faster.

Parallel Computer Memory Architectures

1. **Shared Memory**:
    1. All processors share the same memory space.
    2. **Advantage**: Fast communication since data is directly accessible.
    3. **Challenge**: Requires synchronization mechanisms to prevent data conflicts.
    4. **Example**: OpenMP on a multi-core CPU.
2. **Distributed Memory**:
    1. Each processor has its local memory.
    2. Communication happens via message passing.
    3. **Advantage**: Scalable for large systems.
    4. **Challenge**: More overhead due to communication.
    5. **Example**: MPI on a cluster of servers.
3. **Hybrid Memory**:
    1. Combines shared and distributed memory models.
    2. **Example**: A distributed system with multi-core processors where threads use shared memory, and nodes communicate using MPI.

Parallel Programming Models

Shared Memory Model

- All processors access the same memory space.
- Threads run in parallel, and synchronization ensures consistency.
- **Example**: OpenMP for intra-node communication.

Distributed Memory Model

- Each process has its own memory space.
- Communication happens via explicit message-passing protocols (e.g., MPI).
- **Example**: HPC clusters running distributed simulations.

Hybrid Model

- Combines shared and distributed memory approaches.
- **Example**: MPI between nodes and OpenMP within nodes.

Shared Memory Model

- **Definition**: Memory is accessible to all processors within a system.
- **Key Challenges**:
  - Requires locks or mutexes to avoid race conditions.
  - Limited scalability as the number of processors increases.
- **Example**:
  - Multi-threaded bank transaction system where multiple threads update account balances.

Flynn’s Programming Model

SISD (Single Instruction, Single Data):

• Traditional sequential execution.

• **Example**: A single-core CPU running one task.

SIMD (Single Instruction, Multiple Data):

• Same instruction operates on multiple data simultaneously.

• **Example**: GPUs processing multiple pixels in an image at once.

MISD (Multiple Instruction, Single Data):

• Rarely used; multiple instructions operate on the same data.

• **Example**: Fault-tolerant systems.

MIMD (Multiple Instruction, Multiple Data):

• Different instructions operate on different data concurrently.

• **Example**: Multi-core CPUs or supercomputers.

Pipeline Computations

- **Definition**: A series of stages where data flows sequentially.Each stage processes part of the task while other stages handle other parts, increasing throughput.
- **Pipeline Stages**: Fetch → Decode → Execute → Write Back.
- **Example**: Instruction pipelining in CPUs where multiple instructions are executed simultaneously but at different stages.

Automatic vs Manual Parallelization

Automatic Parallelization:

- Compilers identify and execute parallel code automatically.
- **Advantage**: Saves time and effort for developers.
- **Example**: Intel's ICC compiler.

Manual Parallelization:

- Developers explicitly define parallel tasks.
- **Advantage**: Provides more control and optimization opportunities.
- **Example**: Writing OpenMP code for matrix multiplication.

Comparison:

- **Automatic**: Easier but less efficient for complex tasks.
- **Manual**: More effort but better optimization for large-scale problems.

Data Dependencies

Types of Data Dependencies:

1.True Dependency (Read-after-Write):

1. One instruction depends on the result of a previous one.

```
x = a + b   # Instruction 1
y = x + c   # Instruction 2 depends on Instruction 1
```

2. Anti-Dependency (Write-after-Read):

1. Writing to a location before it is read.
2. **Example**: Writing to x before reading its current value.

3. Output Dependency (Write-after-Write):

1. Multiple instructions write to the same variable.
2. **Example**: Two threads writing to y concurrently.

Breaking Dependencies:

•Use techniques like loop unrolling or reordering instructions.

Load Balancing

Definition: Ensures even distribution of tasks across processors to prevent idle time.

Techniques: **Static Balancing**: Tasks are assigned before execution.

Dynamic Balancing: Tasks are assigned during runtime based on workload.

Example: A web server distributes incoming requests to multiple servers based on their current load.

Conclusion

- **Key Takeaways**:
  - Parallel computing boosts performance by dividing tasks across multiple processors.
  - Understanding memory architectures and programming models is essential.
  - Effective parallelization requires resolving data dependencies and maintaining load balance.
- **Example**:
  - Scientific simulations running on high-performance clusters use parallel computing to achieve faster results.

Cloud Computing

"Cloud computing enables on-demand access to resources and services over the internet, transforming how businesses operate."

Example: Think of Google Drive, where users can store, access, and share files seamlessly, without managing the physical hardware.

Cloud Computing Models

1. Infrastructure as a Service (IaaS):

1. Provides virtualized computing resources like servers, storage, and networking.
2. Example: **AWS EC2**, where users can rent virtual machines.

2. Platform as a Service (PaaS):

1. Offers platforms for developers to build and deploy applications.
2. Example: **Google App Engine**, where developers can deploy apps without worrying about infrastructure.

3. Software as a Service (SaaS):

1. Delivers software applications over the internet.
2. Example: **Google Workspace (Docs, Sheets)**.

Characteristics of Cloud Models

1. IaaS:

On-demand infrastructure scalability.

Example: Expanding storage during peak traffic periods.

2. PaaS:

Simplifies application development with pre-built tools.

Example: Using Firebase for real-time database management.

3. SaaS:

Subscription-based access to applications.

Example: Salesforce for CRM.

Web Application Framework

Definition: Provides a structure for web development by offering pre-written components and libraries.

Popular Frameworks:

- **Django** (Python): Secure, scalable framework for backend development.
- **React.js** (JavaScript): Used for building dynamic, interactive UIs.
- **Ruby on Rails** (Ruby): Framework for database-backed applications.

  Example: Using Django to create an e-commerce website with built-in authentication and admin panels.

Benefits of Web Application Frameworks

Simplifies development by providing pre-built components.

Enhances security with features like input validation.

Boosts scalability for handling high traffic.

Example: Building a social media platform like Instagram using Django or React.js to manage millions of users.

Cloud Web Services

Definition: Services provided by the cloud to enhance web applications, such as storage, databases, or APIs.

Examples:

- **Amazon S3**: Cloud storage for storing images or backups.
- **Google Cloud Functions**: Running serverless functions for real-time processing.
- **Azure Cognitive Services**: Provides AI capabilities like image recognition.

  Example: A ride-hailing app like Uber uses cloud APIs to display maps and calculate routes.

Service-Oriented Architecture (SOA)

1. Definition: A design pattern where services (independent functionalities) communicate over a network.

2. Core Components: **Service Provider**: Hosts the services (e.g., weather API).

3. Service Consumer: Uses the services (e.g., a weather app).

4. Service Registry: Helps discover services.

Example: SOA in a shopping platform where different services handle payments, inventory, and user authentication independently.

SOA Towards Cloud Computing

SOA forms the foundation of cloud services by enabling:

1. Service Reusability: Services like login APIs are used across multiple apps.

2. Scalability: Independent services can scale based on demand.

3. Interoperability: Services communicate regardless of underlying platforms.

Example: A banking app using SOA to integrate payment gateways, fraud detection, and user notifications.

Cloud Integration Using SOA

- **Benefits of Integration**: Streamlines development with reusable services.
- Reduces development costs.
- Enhances flexibility for changes.
- **Challenges**: Managing service dependencies.
- Ensuring data consistency across services.

Example: Netflix uses SOA for its microservices architecture, hosted on AWS, to handle millions of user requests simultaneously.

Conclusion

- Cloud computing revolutionizes IT by enabling scalable and cost-effective solutions.
- Web application frameworks and cloud web services simplify development and deployment.
- SOA plays a pivotal role in building reusable, scalable cloud services.

Example: Leveraging SOA on AWS to develop a healthcare platform for patient data management and real-time analytics.

Thanks & Cheers!!

Small aim is a crime; have great aim.

Bharat-Ratan A. P. J. Abdul Kalam