Name – Pulkit Agrawal
Reg.No. – 23BCE10735

Mapping Techniques for Load Balancing

Load balancing ensures that workload is distributed evenly across processors/nodes in a distributed system to improve performance and resource utilization.

A. Static Mapping

- Decisions made **before execution**
- Based on prior knowledge of tasks
- No runtime changes

Examples:

- Round Robin allocation
- Random allocation
- Hash-based mapping

Advantages:

- Simple implementation
- Low overhead

Disadvantages:

- Not suitable for dynamic workloads
- Cannot adapt to failures

B. Dynamic Mapping

- Decisions made **during execution**
- Uses current system load information
- Tasks may migrate between nodes

Types:

1. **Centralized Load Balancing**
    - One node decides distribution
    - Simple but single point of failure
2. **Distributed Load Balancing**
    - Each node participates in decision making
    - More scalable and fault tolerant
3. **Hierarchical Load Balancing**
    - Nodes grouped in clusters
    - Load balancing within and between clusters

Advantages:

- Adaptive
- Better performance under varying load

Disadvantages:

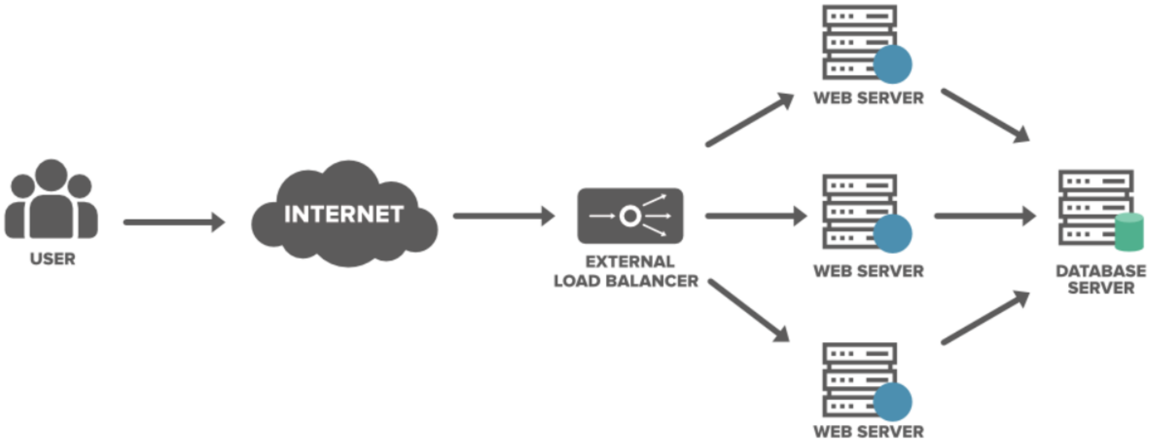
- Communication overhead
- More complex

C. Sender-Initiated vs Receiver-Initiated

Technique	Description
Sender-Initiated	Overloaded node sends tasks to others
Receiver-Initiated	Idle node requests tasks
Symmetric	Combination of both

D. Task Assignment Strategies

- Immediate assignment
- Batch assignment
- Threshold-based assignment



Design Issues in Distributed Systems

Designing distributed systems involves multiple challenges because components are geographically separated and communicate over networks.

1. Transparency

System should appear as a **single unified system**.

Types:

- Access transparency
- Location transparency
- Replication transparency
- Failure transparency
- Migration transparency

2. Scalability

System should handle growth in:

- Users
- Resources
- Geographic distribution

Approaches:

- Decentralization
- Partitioning
- Replication

3. Fault Tolerance

System must continue functioning despite failures.

Techniques:

- Replication
- Checkpointing
- Consensus algorithms

4. Consistency

Maintaining correct data across multiple nodes.

Models:

- Strong consistency
- Eventual consistency

5. Communication

- Message passing
- Remote Procedure Calls (RPC)
- Middleware

Challenges:

- Latency
- Bandwidth limitations
- Network failures

6. Security

- Authentication
- Authorization
- Encryption

7. Concurrency Control

Multiple users accessing shared resources.

Solutions:

- Locks
- Timestamps
- Distributed transactions

8. Heterogeneity

Different:

- Hardware
- Operating systems
- Networks
- Programming languages

Middleware helps solve this issue.