1️⃣ Parallel Computing (8 Marks – Detailed)

1. Definition

Parallel computing is a computing technique in which multiple processors execute multiple parts of a program simultaneously to reduce execution time and increase performance.

Instead of solving a problem sequentially, the task is divided into smaller sub-tasks and processed at the same time.

2. Architecture Types

a) Shared Memory

- All processors share a common memory.
- Communication through shared variables.
- Example: Multi-core CPUs.

b) Distributed Memory

- Each processor has its own memory.
- Communication via message passing.
- Example: Cluster systems.

c) Hybrid Model

- Combination of shared and distributed memory.

3. Working

1. Problem is divided into smaller sub-problems.
2. Sub-problems are assigned to multiple processors.
3. Processors execute tasks simultaneously.
4. Results are combined to produce final output.

4. Advantages

✓ High Speed

Tasks execute simultaneously, reducing total execution time.

✓ Efficient Resource Utilization

All processors work together, increasing CPU usage efficiency.

✓ Scalability

More processors can be added to improve performance.

✓ Suitable for Large Problems

Used in scientific computing, simulations, ML training.

5. Disadvantages

✗ Complex Programming

Writing parallel programs is difficult due to synchronization and race conditions.

✗ Cost

Requires multiple processors or GPUs.

✗ Communication Overhead

Processors must communicate, which adds delay.

✗ Debugging Difficulty

Parallel bugs are harder to detect.

6. Applications

- AI & Deep Learning training
- Weather forecasting
- Matrix operations
- GPU computing

2️⃣ Distributed Computing Models (8 Marks – Detailed)

1. Definition

Distributed computing is a system where multiple independent computers communicate over a network and work together as a single system.

2. Characteristics

- No shared memory
- Communication over network
- Fault tolerant
- Scalable

3. Types of Models

a) Client-Server Model

Server provides services to clients.

b) Peer-to-Peer Model

All nodes equal and share resources.

c) Master-Slave Model

Master assigns tasks to workers.

d) Cluster Model

Group of computers working as one system.

4. Advantages

✓ Scalability

New machines can be added easily.

✓ Fault Tolerance

If one node fails, others continue.

✓ Resource Sharing

Data and hardware resources shared.

✓ Geographic Distribution

Systems can operate globally.

5. Disadvantages

✗ Network Dependency

System performance depends on network speed.

✗ Security Risks

More vulnerable to cyber attacks.

✗ Complex Management

Monitoring many nodes is difficult.

✗ Data Consistency Issues

Maintaining synchronization is challenging.

6. Applications

- Cloud computing
- Distributed databases
- Blockchain
- Web services

3️⃣ Message Passing (8 Marks – Detailed)

1. Definition

Message passing is a communication mechanism where processes exchange data by sending and receiving messages.

2. Types

a) Synchronous

Sender waits until message is received.

b) Asynchronous

Sender continues after sending message.

3. Communication Types

- Point-to-point
- Broadcast
- Multicast

4. Advantages

✓ Suitable for Distributed Systems

No shared memory required.

✓ Scalable

Works well across large clusters.

✓ Clear Communication Structure

Explicit send/receive functions.

5. Disadvantages

✗ Communication Delay

Network latency affects performance.

✗ Programming Complexity

Managing messages manually is complex.

✗ Overhead

Frequent communication reduces efficiency.

6. Example

- MPI (Message Passing Interface)
- Cluster computing systems

4️⃣ Distributed File Systems (HDFS & GFS) (8 Marks – Detailed)

Distributed File System stores large files across multiple machines.

A) Hadoop Distributed File System (HDFS)

Architecture

- NameNode → Manages metadata
- DataNode → Stores actual data
- Files split into blocks (128MB+)
- Data replication (default 3 copies)

Advantages

✓ Fault Tolerant (data replication)
✓ High throughput
✓ Suitable for big data
✓ Scalable

Disadvantages

✗ NameNode failure risk
✗ Not suitable for small files
✗ High latency
✗ Requires large storage infrastructure

B) Google File System (GFS)

Architecture

- Master server
- Chunk servers
- Large chunk size (64MB)

Advantages

✓ Optimized for large data
✓ Automatic replication
✓ High performance

Disadvantages

✗ Not open source
✗ Single master bottleneck
✗ Complex implementation

5️⃣ Cluster Computing (AWS, Azure, GCP) (8 Marks – Detailed)

Cluster computing connects multiple computers to work as a single system.

A) AWS

Amazon Web Services

Advantages

✓ Elastic scaling
✓ Pay-as-you-go
✓ Global availability
✓ Large service ecosystem

Disadvantages

✗ Cost can increase quickly
✗ Learning curve
✗ Vendor lock-in

B) Azure

Microsoft Azure

Advantages

✓ Strong enterprise support
✓ Integration with Microsoft tools
✓ Hybrid cloud support

Disadvantages

✗ Complex pricing
✗ Service configuration complexity

C) GCP

Google Cloud Platform

Advantages

✓ Strong AI & ML tools
✓ High-performance network
✓ Competitive pricing

Disadvantages

✗ Smaller market share
✗ Limited enterprise legacy support

6️⃣ Message Brokers & Stream Processing (8 Marks – Detailed)

Message Brokers

Examples:

- Apache Kafka
- RabbitMQ

Advantages

✓ Asynchronous communication
✓ Decouples services
✓ Reliable delivery

Disadvantages

✗ Setup complexity
✗ Maintenance overhead
✗ Message ordering challenges

Stream Processing

Examples:

- Apache Spark
- Apache Flink

Advantages

✓ Real-time processing
✓ Immediate insights
✓ Scalable

Disadvantages

✗ Complex system design
✗ High memory usage
✗ Debugging difficulty

7️⃣ Edge Computing (8 Marks – Detailed)

Definition

Edge computing processes data near the source instead of centralized cloud.

Advantages

✓ Reduced latency
✓ Faster decision-making
✓ Reduced bandwidth usage
✓ Improved privacy

Disadvantages

✗ Limited computational power
✗ Security challenges
✗ Device management complexity
✗ Maintenance cost

Applications

- Smart cities
- IoT
- Healthcare monitoring
- Autonomous vehicles

1️⃣ Data Replication

Definition

Data replication is the process of storing **multiple copies of data on different nodes** in a distributed system to improve availability, fault tolerance, and performance.

Why Replication is Needed

- Improve availability
- Reduce latency
- Increase reliability
- Fault tolerance
- Load balancing

2 Eager Replication (Synchronous Replication)

Definition

Eager replication updates **all replicas immediately and synchronously** before confirming success to the client.

Working

1. Client sends write request.
2. Primary node updates all replicas.
3. All replicas confirm update.
4. Client receives success response.

Advantages

✓ Strong consistency
✓ No stale data
✓ Immediate synchronization

Disadvantages

✗ High latency
✗ Network overhead
✗ Reduced availability (if one replica fails, write may fail)

Example

Banking systems where data correctness is critical.

3 Lazy Replication (Asynchronous Replication)

Definition

Lazy replication updates the primary replica first and then propagates updates to other replicas later.

Working

1. Client writes to primary.
2. Primary confirms success.
3. Updates sent to replicas later (background).

Advantages

✓ Low write latency
✓ Better availability
✓ Faster response time

Disadvantages

✗ Temporary inconsistency
✗ Stale reads possible
✗ Conflict resolution required

Example

Social media posts, caching systems.

4 Quorum-Based Replication

Definition

Quorum replication ensures consistency by requiring a minimum number of nodes (quorum) to agree before a read or write is considered successful.

Working

- N = Total replicas
- W = Write quorum
- R = Read quorum
  Condition:
  $W + R > N$

This ensures overlap between read and write sets.

Advantages

✓ Balanced performance and consistency
✓ Tunable consistency
✓ Fault tolerant

Disadvantages

✗ Configuration complexity
✗ High coordination cost

Example

Distributed databases like NoSQL systems.

5 Consensus-Based Replication

Definition

Consensus-based replication ensures that all nodes agree on the same value using consensus algorithms.

Common Algorithms – Paxos, Raft

Working

1. Leader proposes value.
2. Majority nodes approve.
3. Value committed.

Advantages

✓ Strong consistency
✓ Prevents split-brain
✓ Reliable in distributed systems

Disadvantages

✗ Complex implementation
✗ High communication overhead
✗ Slower writes

Example

Distributed configuration systems, blockchain.

6️⃣ Selective Replication

Definition

Selective replication replicates only selected frequently accessed or critical data across nodes.

Working

- Important data replicated widely.
- Less important data stored in limited nodes.

Advantages

✓ Reduced storage cost
✓ Optimized performance
✓ Efficient bandwidth usage

Disadvantages

✗ Complex decision logic
✗ Risk of unavailability for non-replicated data

Example

Content Delivery Networks (CDNs)

7️⃣ Consistency Models

Consistency defines how up-to-date and synchronized data is across replicas.

8️⃣ Strong Consistency

Definition

After a write, any subsequent read will return the most recent value.

Characteristics

- Immediate consistency
- Linearizability
- No stale reads

Advantages

✓ Reliable data
✓ Suitable for banking & financial systems

Disadvantages

✗ High latency
✗ Reduced availability
✗ Not scalable globally

9️⃣ Eventual Consistency

Definition

If no new updates occur, all replicas will eventually converge to the same value.

Characteristics

- Temporary inconsistency allowed
- Eventually synchronized

Advantages

✓ High availability
✓ Low latency
✓ Highly scalable

Disadvantages

✗ Stale data possible
✗ Conflict resolution needed

Example

Social media likes, comments.

🔟 Read-Your-Writes Consistency

Definition

A client will always see its own updates immediately.

Advantages

✓ Better user experience
✓ Simple session guarantee

Disadvantages

✗ Does not guarantee global consistency

Example

After posting a tweet, user immediately sees it.

1️⃣1️⃣ Consistent Prefix Consistency

Definition

Reads never see out-of-order writes. Updates are seen in the correct sequence.

Advantages

✓ Maintains order
✓ Prevents anomalies

Disadvantages

✗ Does not guarantee latest value

Example

Chat applications showing messages in order.

1️⃣2️⃣ Causal Consistency

Definition

If operation B depends on operation A, then all nodes must see A before B.

Working

Respects cause-and-effect relationships.

Advantages

✓ More realistic consistency
✓ Better than eventual consistency

Disadvantages

✗ Metadata overhead
✗ Complex tracking of dependencies

Example

If you reply to a message, everyone must see original message before reply.

1️⃣ Distributed Hash Tables (DHTs)

1. Definition

A Distributed Hash Table (DHT) is a decentralized distributed indexing system that stores **(key, value) pairs across multiple nodes**, where each node is responsible for a specific portion of the key space determined by a hash function.

It eliminates the need for a centralized index server.

2. Architecture

- Each node has a unique identifier (Node ID).
- A consistent hash function (e.g., SHA-1) maps:
  - Keys → Key ID
  - Nodes → Node ID
- The key is stored at the node whose ID is closest to the key ID.

Common DHT Protocols:

- Chord, Kademlia, Pastry, CAN

3. Working Mechanism

Step 1: Hashing

Key K → Hash(K) → 160-bit identifier.

Step 2: Key Placement

Key stored at node whose ID ≥ Key ID (in circular ring).

Step 3: Lookup

Node forwards request using routing table.
Lookup complexity = O(log N).

Step 4: Node Join/Leave

System redistributes keys automatically using consistent hashing.

4. Design Goals

- Scalability to millions of nodes
- Decentralization
- Fault tolerance
- Load balancing

5. Advantages (Detailed)

✓ **High Scalability -** Lookup grows logarithmically, not linearly.

✓ **Decentralized -** No single point of failure.

✓ **Automatic Load Distribution -**Consistent hashing balances keys across nodes.

✓ **Self-healing -** Nodes can join/leave dynamically.

6. Disadvantages (Detailed)

✗ **Poor Range Queries**

DHT supports exact-match lookup only.

✗ **Network Overhead -** Frequent routing updates required.

✗ **Security Issues -** Malicious nodes can disrupt routing.

✗ **Data Movement During Rebalancing**

When nodes join, key migration required.

7. Real-World Applications

- BitTorrent, IPFS, Blockchain peer discovery ,Distributed caching

2️⃣ Distributed Inverted Indexing

1. Definition

Distributed Inverted Indexing is a technique where keyword-to-document mappings are distributed across multiple nodes to support scalable full-text search.

2. Architecture

Components:

- Document Partitioning
- Index Nodes
- Query Router
- Aggregator

Two Distribution Strategies:

1. Document-based partitioning
2. Term-based partitioning

3. Working Mechanism

Index Creation

Each node:

- Tokenizes documents
- Builds inverted index (term → document list)

Query Processing

1. Query broadcasted.
2. Each node searches local index.
3. Results merged and ranked.

4. Design Goals

- Fast search
- Scalable indexing
- Low latency
- Distributed ranking

5. Advantages (Detailed)

✓ Efficient Full-Text Search

Very fast keyword matching.

✓ Horizontal Scalability

Add more nodes to handle data growth.

✓ Parallel Query Processing

Query executed simultaneously across nodes.

✓ Suitable for Big Data

Handles billions of documents.

6. Disadvantages (Detailed)

✗ Complex Index Maintenance

Updating distributed index is expensive.

✗ Synchronization Cost

Ensuring consistency across partitions is difficult.

✗ High Storage Overhead

Posting lists consume large memory.

7. Real-World Example

- Search engines, E-commerce platforms, Log analytics systems

3️⃣ Range-Based Partitioning

1. Definition

Range-based partitioning distributes data across nodes based on key ranges.

Each node handles a specific interval of values.

2. Architecture

Example:
Node A → 1–1000
Node B → 1001–2000
Node C → 2001–3000

Range metadata stored in directory service.

3. Working

1. Incoming data assigned based on value range.
2. Queries routed to relevant node.
3. If range splits, rebalancing occurs.

4. Design Goals

- Support efficient range queries
- Maintain sorted order
- Enable efficient indexing

5. Advantages (Detailed)

✓ Excellent for Range Queries

Time-series queries become very efficient.

✓ Preserves Order

Data stored in sorted manner.

✓ Simpler Query Planning

Predictable data location.

6. Disadvantages (Detailed)

✗ Load Imbalance (Hotspot Problem)

Some ranges accessed more frequently.

✗ Difficult Rebalancing

Splitting ranges requires data movement.

✗ Central Directory Dependency

Range map often centrally managed.

7. Applications

- Financial transactions, Time-series databases, Distributed SQL databases

4️⃣ Content-Based Indexing

1. Definition

Content-based indexing retrieves data based on actual content features instead of keys.

Used mainly in multimedia systems.

2. Architecture

Steps:

- Feature Extraction
- Feature Vector Storage
- Similarity Matching Engine

Stored in distributed vector databases.

3. Working

1. Extract content features.
2. Store features across nodes.
3. Query processed using similarity search.
4. Top-k nearest matches returned.

4. Design Goals

- Similarity search
- Intelligent retrieval
- AI integration

5. Advantages (Detailed)

✓ Supports AI Applications

Works with ML embeddings.

✓ Flexible Querying

Search by similarity, not exact match.

✓ Powerful Multimedia Retrieval

6. Disadvantages (Detailed)

✗ High Computational Cost

Similarity search expensive.

✗ Large Storage for Vectors

Feature vectors consume memory.

✗ Approximate Results

Not always exact matches.

7. Applications

- Image search
- Recommendation systems
- Face recognition systems

5️⃣ Peer-to-Peer (P2P) Indexing

1. Definition

P2P indexing distributes index responsibilities among peers without central authority.

2. Types

Structured P2P

Uses DHT.

Unstructured P2P

Flooding-based search.

3. Working

- Nodes share index information.
- Query routed via overlay network.
- Peers collaborate to resolve query.

4. Advantages

✓ No central failure
✓ Highly scalable
✓ Low infrastructure cost

5. Disadvantages

✗ High lookup latency
✗ Security risks
✗ Unpredictable performance

6. Applications

- File sharing, Blockchain, Decentralized storage

6️⃣ Hybrid Approaches

1. Definition

Hybrid indexing combines multiple indexing strategies to optimize performance and scalability.

Example:
DHT + Range Partition
Central directory + P2P
Content-based + Hash indexing

2. Architecture

- Frequently accessed data → Fast index
- Rare data → Distributed storage
- Intelligent routing layer

3. Advantages (Detailed)

✓ Balanced Performance

Combines strengths of multiple methods.

✓ Adaptive Scaling

Handles diverse workloads.

✓ Improved Fault Tolerance

4. Disadvantages (Detailed)

✗ High System Complexity

Hard to design and maintain.

✗ Expensive Implementation

✗ Complex Debugging