Distributed Systems - Assignment 1

Gowlapalli Rohit - 2021101113

Question 1

Distributed BFS

```
The graph is partitioned into equal parts and each process is assigned a
set of vertices. When a vertex is visited, its neighbors are communicated
to the corresponding process.
```

Time Complexity:

- **Factors:**
  - Graph traversal (BFS)
  - Data exchange (MPI communication)
- **Graph Partitioning:** O(V/size) per process
- **BFS Traversal:**
  - O(V/size + E) per process
  - Communication overhead: O(d * V/size)
  - Total: O(V/size + E + d * V/size)
- **Gathering Results:** O(V)

Total Time: O(V/size + E + d * V/size) + O(V)

Space Complexity:

- **Per Process:** O(V/size + E/size + L)
  - Adjacency list
  - Level array
  - Blocked set
- **Total System:** O(V + E + L)

Message Complexity:

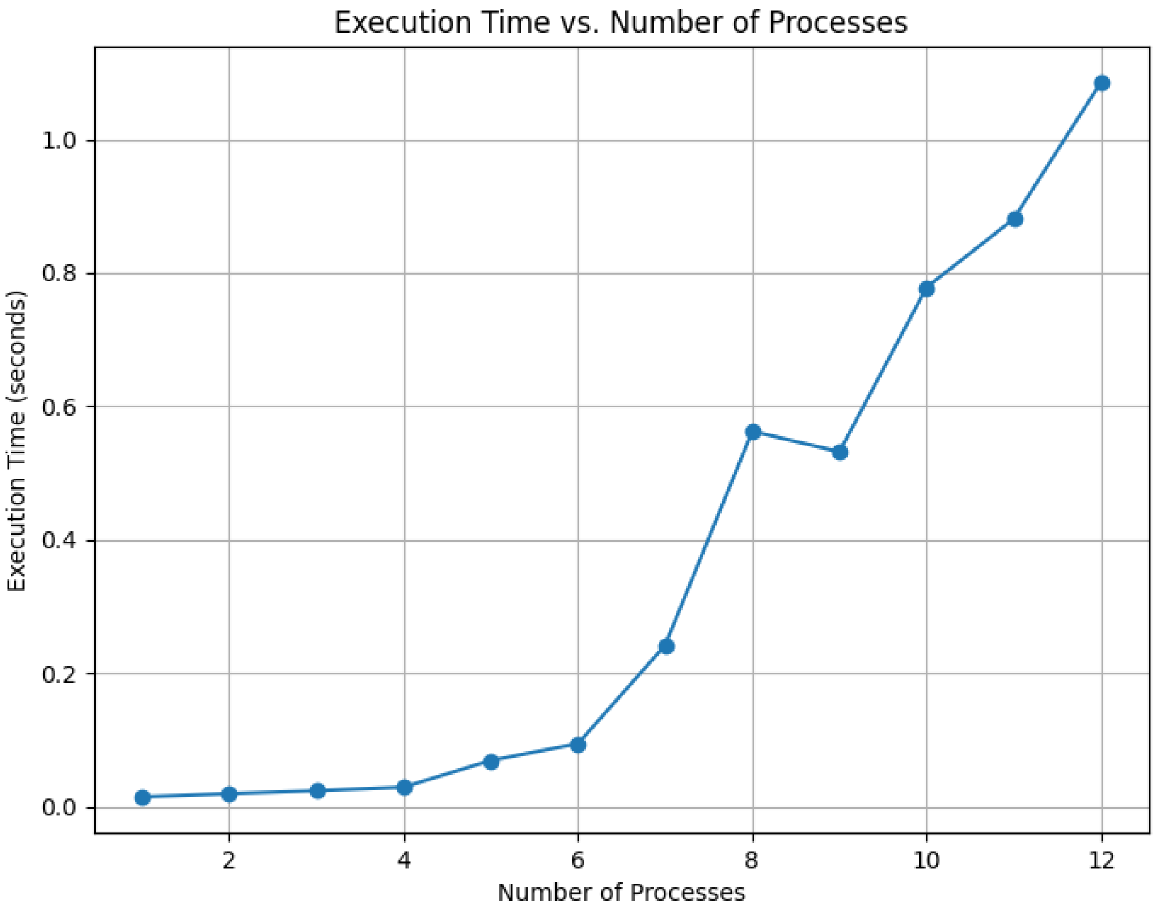
- **Initial Distribution:** O(E)
- **BFS Traversal:** O(V * d)

Total Messages: O(V * d)

Summary:

- **Time:** O(V/size + E + d * V/size) + O(V)
- **Space:** O(V + E + L)
- **Messages:** O(V * d)

Scaling Analysis



Question 2

Distributed Particle Simulation

```
The grid is divided into rows and each process is assigned a set of rows.
When a particle moves to a different row, it is communicated to the
corresponding process.
```

Time Complexity:

- **Initialization:** O(K)
  - Reading K particles and assigning them to processes.
  - Broadcasting N, M, K, T: O(log P)
- **Particle Movement & Collision Handling:** O(T * K)
  - Per iteration:
    - Updating positions: O(K/P)
    - Communication: O(K/P)
    - Erasing invalid particles: O(K/P)
    - Collision handling: O(K/P)
- **Gathering Results:**

  - Sorting particles: O(K log K)
  - Gathering particles: O(K + P)

Space Complexity:

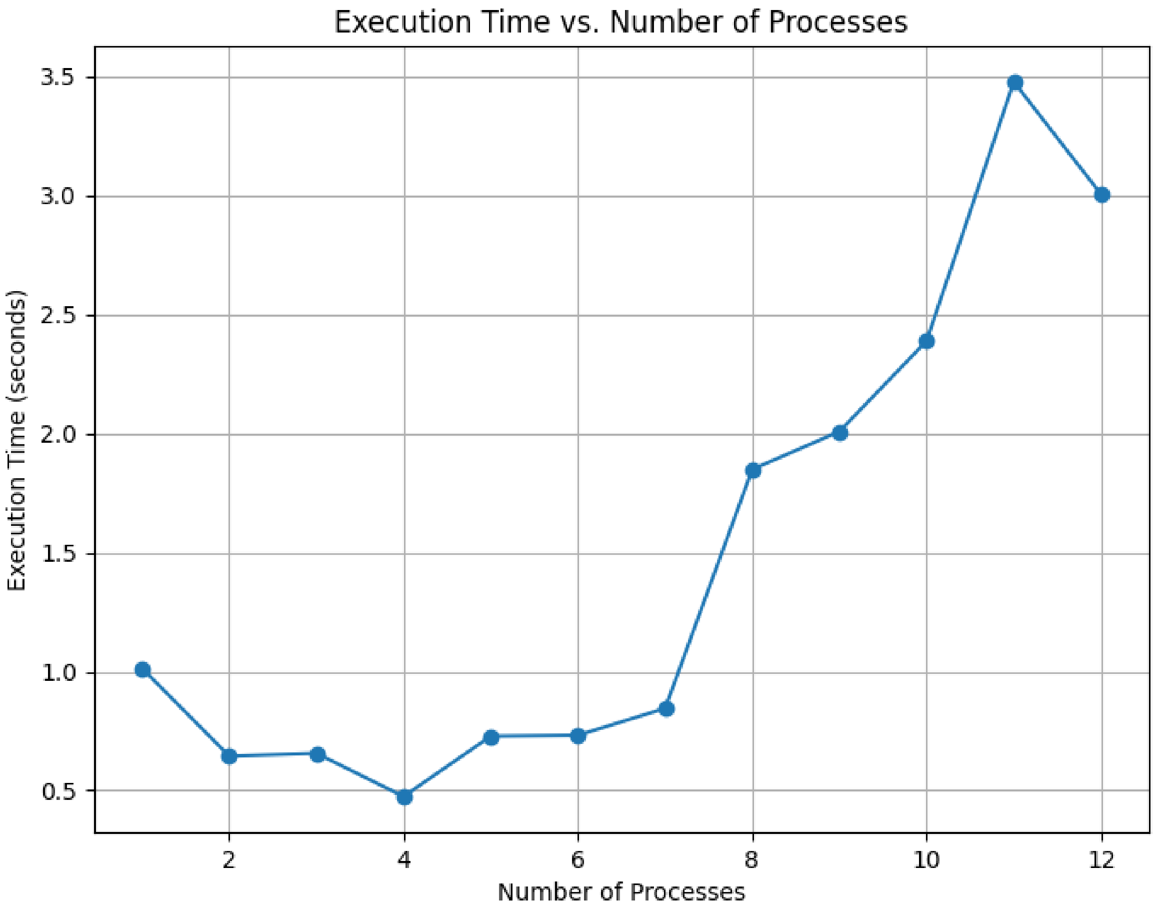
- **Per Process:** O(K/P) (particles, collision handling data)
- **Global (Rank 0):** O(K) (gathered particles)

Message Complexity:

- **Broadcasting Constants:** O(log P)
- **Particle Communication:** O(T * K)
- **Gathering Results:** O(K + P)

Summary:

- Time: O(T * K + K log K)
- Messages: O(T * K + P)
- Space: O(K + K/P)



Question 3

- **Nodes and Chunks:**
  - N: Total number of nodes.
  - C: Number of chunks in a file.

- F: Number of failed nodes.
- Replication: Each chunk has a replication factor of 3.
- Storage: The chunk size is CHUNK_SIZE.
- **Heartbeat:** All nodes send periodic heartbeat messages to the metadata server.

Time Complexity

- **Upload:** O(C * N)
  - Chunk Upload: O(N) per chunk (replica node selection)
- **Retrieve:** O(C)
  - Replica Search and Retrieval: O(1) per chunk
- **Search:** O(C * W)
  - Node Selection: O(C)
  - Word Search: O(W) per chunk (where W is the average words per chunk)
- **Heartbeat:** O(N)
  - All nodes send heartbeats to the metadata server.
- **Failover/Recover:** O(C * N)
  - Node Selection for Re-replication: O(N) per chunk

Space Complexity

- **Metadata Server:** O(F * C)
  - File metadata: O(F)
  - Chunk metadata: O(C) per file
- **Storage Nodes:** O(F * C * CHUNK_SIZE)
  - Chunk data: O(C * CHUNK_SIZE) per file

Message Complexity

- **Upload:** O(C)
  - 12 messages per chunk (4 messages per replica)
- **Retrieve:** O(C)
  - 4 messages per chunk (metadata request + data retrieval)
- **Search:** O(C)
  - 4 messages per chunk (similar to retrieval)
- **Heartbeat:** O(N)
  - N - 1 heartbeat messages
- **Failover/Recover:** O(C)
  - Metadata updates during re-replication

Summary

- **Overall Time Complexity:** O(C * N + C * W + N)
- **Overall Space Complexity:** O(F * C * CHUNK_SIZE + F * C)
- **Overall Message Complexity:** O(C * 12 + N)

Assumptions

Question 1

- 1D grid partitioning of the graph.

Question 2

- 1D grid partitioning of the matrix
- It is assumed that particles cannot collide at the start of the simulation.

Question 3

- Distributed Search is done by fetching only the relevant chunks that have a complete match with search word or if their suffix or prefix is a substring of the search word.
- Heartbeat Interval is 500ms and Failover Interval is 1500ms
- Load balancing is done by always picking storage nodes with the least number of chunks. (When a node fails, the chunks count of the node becomes zero)
- It was assumed that there are no time constraints on the upload and retrieval of chunks. The system is optimized for fault tolerance and load balancing.
- Master Node has maps to store the metadata of files and chunks. It also has a list of all storage nodes.
- Each chunk metadata contains chunk id and the replication nodes.
- Each filemetadata also contains offset information for each chunk.

Distributed File System: Design and Workflow

This distributed file system (DFS) leverages MPI (Message Passing Interface) to distribute file operations across multiple nodes, ensuring fault tolerance, scalability, and efficient resource utilization.

Key Components

1. **File Chunking**:

   - Files are split into fixed-size chunks (`CHUNK_SIZE`), enabling parallel storage and retrieval.

2. **Replication**:

   - Each chunk is replicated across `REPLICATION_FACTOR` nodes to provide redundancy.

3. **Heartbeat Mechanism**:

   - Nodes send periodic heartbeat messages (`HEARTBEAT_TAG`) to indicate availability.
   - Heartbeats are monitored to detect node failures (`FAILOVER_INTERVAL`).

4. **Failover and Recovery**:

   - Failed nodes are excluded from operations.
   - A recovery mechanism re-integrates nodes upon restoration.

5. **MPI Communication Tags**:

   - Tags like `UPLOAD_TAG`, `RETRIEVE_TAG`, and `SEARCH_TAG` coordinate operations between nodes.

6. **Metadata Management**:
    - `FileMetaData` and `ChunkMetaData` structures store file and chunk information, including replication details.

Workflow

1. **File Upload**:
    - **Master Node** (rank 0): Splits the file into chunks and assigns chunks to nodes using a load-balancing strategy.
    - **Worker Nodes**: Store assigned chunks locally.
2. **File Retrieval**:
    - Master node queries workers for chunks of a file.
    - Worker nodes respond with the requested data.
3. **Search Operation**:
    - Searches for a word across file chunks.
    - Worker nodes analyze chunks locally and send results to the master.
4. **Heartbeat Monitoring**:
    - Master node monitors the heartbeats of all nodes.
    - Failure detection triggers failover actions (e.g., halting operations on failed nodes).
5. **Node Failover and Recovery**:
    - Nodes can be marked as failed using a `failover` command.
    - A `recover` command reintegrates restored nodes into the system.
6. **System Exit**:
    - Master sends an `EXIT_TAG` to terminate all worker nodes gracefully.