Parallel Processing Project 3 Report

Team Members: Reem Ooka & Nihar Lodaya

1) Architecture/Algorithm Used and Why?

1. Overall System Architecture: The system operates as a parallel database across multiple MPI processes (nodes). Each node runs the same program (SPMD model), with rank 0 as coordinator—reading queries, broadcasting them, and aggregating results. All nodes store and process data, and adding more nodes improves scalability and parallel processing capability.

2. Data Distribution Algorithm (Hash-Based Placement): We use a hash-based method to determine which node stores each inserted tuple. Hashing the "major" field distributes tuples by attribute values rather than randomly or in a round-robin manner, helping avoid skew and creating a more balanced workload.

3. Handling INSERT & SELECT Queries: Rank 0 broadcasts each INSERT query to all nodes. Every node parses it, computes the target node via hashing, and only the target node stores the tuple. This ensures a consistent, deterministic distribution without complex coordination.

SELECT queries are also broadcast. Each node filters its local tuples in parallel, then sends matching results back to rank 0 via point-to-point communication. This distributes the query load, leverages parallelism, and concentrates final aggregation at a single node.

4. MPI Communication Strategies:

- **Collective (MPI_Bcast):** Used for both INSERT and SELECT queries, ensuring all nodes stay synchronized.
- **Point-to-Point (MPI_Send/MPI_Recv):** Used to return SELECT results to rank 0 without unnecessary communication overhead.

5. Query Parsing and Data Structures: A simplified SQL-like syntax and a fixed schema keep parsing and storage straightforward. Tuples are stored in arrays for fast indexing and minimal overhead.

By using hashing to intelligently balance data distribution, broadcasting queries to maintain simple, synchronized logic among all nodes, employing point-to-point communication to minimize overhead in returning results, and relying on simple parsing with a fixed schema for low complexity, the system achieves a scalable, efficient, and balanced parallel database solution that can handle large datasets and queries with high performance.

2) Citations:

None were referred to for this project.

3) Table:

Database of size 1 Million Inserts Statements

No. of Nodes	Runtime	Memory Consumption
1	23.7675 seconds	1560000 bytes
2	19.9208 seconds	1560000 bytes
3	22.0831 seconds	1040000 bytes
4	12.2383 seconds	1170000 bytes
5	10.836 seconds	1248000 bytes
6	12.2077 seconds	1040000 bytes
7	11.7698 seconds	668571 bytes
8	12.2452 seconds	780000 bytes

As the number of nodes increases, we generally expect the system to process the workload faster due to parallelization, which initially explains why moving from 1 node (23.77 seconds) down to 5 nodes (10.84 seconds) reduces runtime. Adding more nodes allows the workload to be split among them, so each node handles fewer tuples, potentially lowering runtime. However, beyond a certain point, communication overhead, synchronization costs, and imperfect load balancing can offset these gains. For example, at 6, 7, or 8 nodes, runtime no longer consistently decreases. This is because more nodes mean more MPI communication and possibly uneven hash-based data distribution, which can reintroduce delays.

Memory consumption similarly fluctuates due to how the hash-based distribution may not perfectly balance the data among all nodes. At certain node counts, the distribution might be more balanced, resulting in each node holding fewer tuples and thus consuming less memory on average. At other node counts, certain nodes might receive more tuples than others, increasing their memory usage. The observed irregular pattern in both runtime and memory usage is a consequence of these combined factors: parallel efficiency, communication overheads, and non-uniform data distribution.

4) Hardware Specs:

We used the Hancock Lab 0004 system to do our project.

- Processor (CPU)
  Intel(R) Xeon(R) w3-2435, 3096 Mhz, 8 Core(s), 16 Logical Processor(s)
- RAM (Memory)
  Installed Physical Memory (RAM)32GB
- BIOS
  LENOVO S0CKT15A, 3/4/20