REPORT FOR PDC MID LAB

Zubair Naeem

SP23-BAI-057

1. Performance Comparison Across All Implementations

Implementation	Workers	Time (s)	Speedup
Sequential	1	0.21	1.00
OpenMP Parallel	1	1.14	0.18
OpenMP Parallel	2	1.34	0.16
OpenMP Parallel	4	1.79	0.12
OpenMP-style Parallel	8	3.89	0.05
MPI Distributed	2	0.20	1.05

2**. Implementation Analysis**

Sequential Processing:

- Baseline performance: 0.21 seconds
- Single-threaded, straightforward implementation
- Surprisingly efficient for this workload size
- No overhead from parallelization mechanisms

OpenMP-style Parallel Processing:

- Uses Python multiprocessing with shared memory
- Performance degraded with increasing workers

- Best time with single worker: 1.14 seconds
- Significant overhead from process creation and management
- Poor scaling due to:
  * Small workload size (94 images)
  * I/O bottlenecks
  * Process creation overhead
  * Resource contention

MPI Distributed Processing:

- Two-node simulation: 0.20 seconds
- Matched sequential performance
- Even distribution (47 images per node)
- Minimal communication overhead
- Most efficient parallel implementation

3. Performance Analysis

Unexpected Results:

- Sequential implementation outperformed OpenMP-style parallel
- Adding more workers decreased performance
- MPI implementation maintained efficiency

Root Causes:

1. Workload Characteristics:
   - Small dataset (94 images)
   - I/O-bound rather than CPU-bound
   - Quick individual image processing time

2. Overhead Factors:

   - Process creation/destruction cost
   - Inter-process communication
   - File system contention
   - Memory management overhead

3. Implementation Differences:

   - Sequential: No parallelization overhead
   - OpenMP-style: Heavy process management overhead
   - MPI: Efficient work distribution, minimal communication