MPI Implementation Analysis: PI Calculation
CS 7172 Parallel and Distributed Computing - Fall 2024

1. Implementation Overview
This report analyzes two parallel implementations of PI calculation using MPI, compared against a sequential version:
1. Sequential Version: Uses the rectangle method to approximate π.
2. Parallel Version 1 (Simple Distribution): evenly divides work among processes.
3. Parallel Version 2 (Pipeline Distribution): Uses smaller chunks in a pipeline fashion.

Note: The implementations were developed and tested on Windows, which required creating a custom header file to define `u_int64_t` and `CLOCK_MONOTONIC` as these are not natively available in the Windows environment. This adaptation ensures compatibility with timing functions found on UNIX systems.

2. Experimental Results

2.1 Comparison of Delays
NUMSTEPS = 1,000,000

Implementation	Processes	Execution Time (ns)
Sequential	1	4,467,400
Parallel V1	4	2,398,200
Parallel V2	4	4,045,100

2.2 Impact of NUMSTEPS on Accuracy and Latency

NUMSTEPS	Implementation	Execution Time (ns)	PI Value
100,000	Sequential	703,300	3.141592653198889
100,000	Parallel V1	1,235,900	3.141687899239145
100,000	Parallel V2	1,465,200	3.141532651098038
1,000,000	Sequential	11,621,400	3.141592653587158
1,000,000	Parallel V1	5,073,100	3.141602178279290
1,000,000	Parallel V2	11,058,600	3.141586653563944
10,000,000	Sequential	546,389,900	3.141592652515961
10,000,000	Parallel V1	26,635,200	3.141593606134609
10,000,000	Parallel V2	87,478,000	3.141592053564827

3. Analysis of Collected Data

3.1 Delay Comparison
- Parallel Version 1 shows significant performance improvement over the sequential version, achieving approximately 1.86x speedup at 1M steps

- Parallel Version 2 shows minimal performance improvement, running only slightly faster than the sequential version
- Parallel V1 consistently outperforms Parallel V2 across all test cases, likely due to its simpler communication pattern and reduced overhead

3.2 Impact of NUMSTEPS

On Accuracy:

- Increasing NUMSTEPS improves accuracy across all implementations
- At 10M steps, all versions achieve accuracy to 6 decimal places
- The difference between implementations becomes smaller as NUMSTEPS increases

On Latency:

- Execution time increases roughly linearly with NUMSTEPS
- Sequential version shows the most dramatic increase in execution time
- Parallel V1 scales better with increased NUMSTEPS compared to Parallel V2

3.3 Performance Analysis

- Speedup Improvements:
  - Parallel V1 shows best speedup at 10M steps (20.5x faster than sequential)
  - Parallel V2 achieves 6.2x speedup at 10M steps
- Performance Bottlenecks:
  - Parallel V2's pipeline approach introduces additional overhead
  - Communication costs become more significant at smaller NUMSTEPS values

3.4 Scalability

- Parallel V1 shows excellent scalability as NUMSTEPS increases
- Parallel V2's pipeline approach shows diminishing returns with larger NUMSTEPS
- Both parallel versions handle larger workloads more efficiently than the sequential version

3.5 Trade-offs

- Accuracy vs. Computation Time:
  - 1M steps provides a good balance between accuracy and performance
  - 10M steps offers minimal accuracy improvement with significant performance cost
- Optimal Configurations:
  - For quick approximations: Use Parallel V1 with 100K steps
  - For balanced accuracy/performance: Use Parallel V1 with 1M steps
  - For highest accuracy: Use Parallel V1 with 10M steps

4. Conclusion

Based on the experimental results, Parallel Version 1 (Simple Distribution) proves to be the superior implementation, offering better performance across all test cases. The simple distribution approach achieves significant speedup while maintaining accuracy comparable to the sequential version. The pipeline approach (Parallel V2) shows less impressive performance gains, suggesting that the added complexity of the pipeline structure introduces overhead that outweighs its potential benefits for this particular calculation.

The optimal configuration for most use cases would be Parallel Version 1 with 1 million steps, providing a good balance between accuracy (6 decimal places) and performance (5.07ms execution time). For applications requiring higher accuracy, increasing to 10 million steps with Parallel Version 1 provides excellent results while maintaining reasonable execution times.