Student: Jorge Afonso
Professor: Micah Schuster
Course: PARALLEL COMPUTING COMP-3450
Date: 3/30/2025

PA5:

Prime Number Computation Performance Analysis

Introduction

This project, assigned by Professor Micah Schuster, aims to analyze the performance of a parallelized C program that computes the number of prime numbers up to using MPI. The key objective is to evaluate how different partition sizes affect execution time while maintaining correctness and help us understand who to work and implement full Master/Worker algorithm.

Methodology

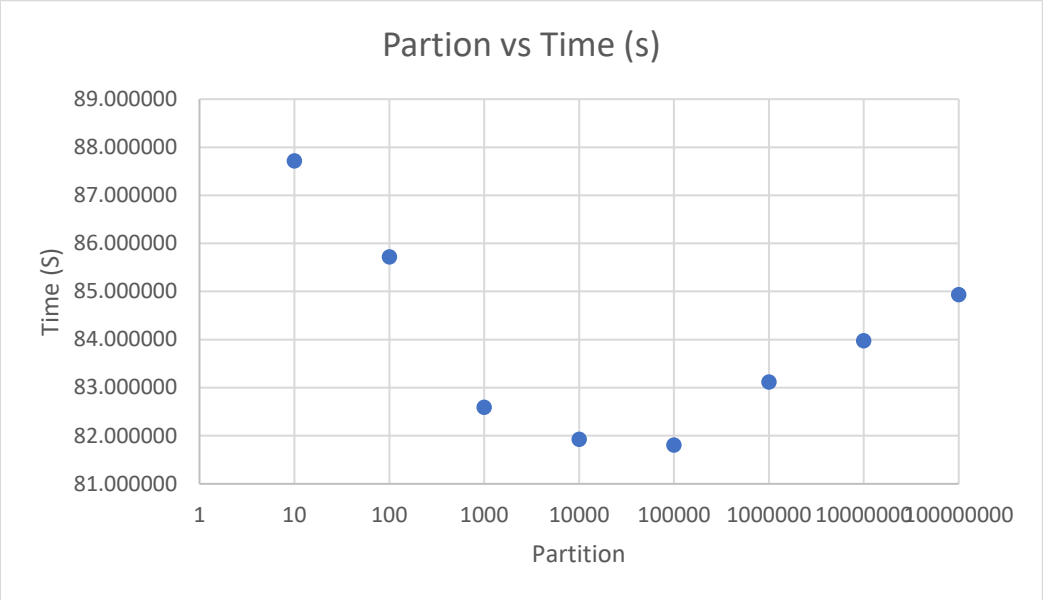
The program is parallelized using MPI, distributing work across five ranks. The workload is divided into partitions ranging from 10 to 100,000,000 numbers per rank, allowing us to measure performance variations and identify the optimal partitioning strategy.

Results and Analysis

The table below summarizes the execution times for different partition sizes:

Number	Ranks	Partition	# Primes	Time (s)
100,000,000	5	10	5,761,455	87.718032
100,000,000	5	100	5,761,455	85.723094
100,000,000	5	1,000	5,761,455	82.593807
100,000,000	5	10,000	5,761,455	81.927598
100,000,000	5	100,000	5,761,455	81.805778
100,000,000	5	1,000,000	5,761,455	83.121908
100,000,000	5	10,000,000	5,761,455	83.976477

100,000,000 5 100,000,000 5,761,455 84.939306

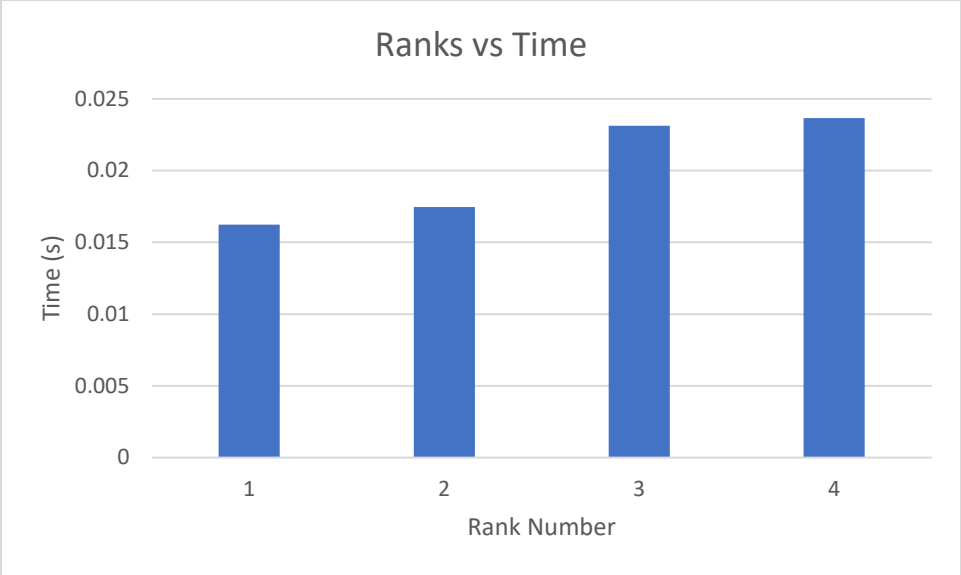


The plot above shows the relationship between the amount of time and Partition. Here we can see execution time changes as the partition size increases. This goes until a certain point, when we hit the optimal partition size leading improved performance before starting taking more time to make to calculate the prime numbers.

From the results, the fastest execution time occurs when the partition size **10,000** and **100,000**, achieving a runtime of **81.927598** and **81.805778** seconds. Beyond this point, increasing partition size results in slightly worse performance due to workload imbalance and MPI communication overhead.

The table below summarizes the execution times for different Ranks:

Number	Ranks #	Partition	Time (s)
100,000,000	1	10,000	0.016236
100,000,000	2	10,000	0.017472
100,000,000	3	10,000	0.023129
100,000,000	4	10,000	0.023654



The graph below demonstrates that the workload is fairly well-balanced across ranks. Ideally, we would want all ranks to have same execution times, but this isn't possible. Calculating primes in smaller ranges, such as 1–10,000, is significantly easier than calculatiung larger values like 100,000–110,000. As a result, some level of imbalance will appear, but the distribution remains efficient overall.

Conclusion

This analysis shows that partitioning significantly impacts performance in parallel prime computation. The optimal partition size for our implementation is around **10,000 and 100,000 numbers per rank**, balancing computational workload and minimizing communication overhead. Future improvements could involve dynamic load balancing to further optimize performance.