CS5473 - Project 5

Author: Enzo Durel

Professor: Dr. Chongle Pan

April 11, 2025

GALLOGLY COLLEGE OF ENGINEERING
The UNIVERSITY of OKLAHOMA

Contents

1 Problem 1 . . . **1**
- 1.1 Runtimes . . . 1
- 1.2 Same Node . . . 1
- 1.3 Diff Node . . . 1
- 1.4 Linear Regression . . . 1

2 Problem 2 . . . **3**
- 2.1 Wall-clock Time Table . . . 3
- 2.2 Speedup . . . 3
- 2.3 Efficiency . . . 3
- 2.4 Discussion . . . 3

3 Problem 3 . . . **4**
- 3.1 Runtimes . . . 4
- 3.2 Speedup . . . 4
- 3.3 Efficiency . . . 4
- 3.4 Discussion . . . 4

4 Problem 4 . . . **5**
- 4.1 Runtimes . . . 5
- 4.2 Speedup . . . 5
- 4.3 Efficiency . . . 5
- 4.4 Discussion . . . 5

List of Figures

1 Linear Regression for same 1
2 Linear Regression for diff 2
3 Linear Regression for both methods 2

List of Tables

1 Problem 1 Runtime . . . 1
2 Problem 2 Wall-clock Time Table . . . 3
3 Problem 2 Speedup Table . . . 3
4 Problem 2 Efficiency Table . . . 3
5 Problem 3 Runtimes . . . 4
6 Problem 3 Speedup . . . 4
7 Problem 3 Efficiency . . . 4
8 Problem 4 Runtimes . . . 5
9 Problem 4 Speedup . . . 5
10 Problem 4 Efficiency . . . 5

1 Problem 1

1.1 Runtimes

Table 1: Problem 1 Runtime

Integers	1M	2M	4M	8M
Same RT	0.0002394965	0.0004862981	0.0012506239	0.0032828204
Diff RT	0.0004300362	0.0008670914	0.0017082404	0.0039351337

1.2 Same Node

- Latency $\approx -3.488867e-04s$
- Bandwidth $\approx 9016.07MB/s$

1.3 Diff Node

- Latency $\approx -1.498357e-04s$
- Bandwidth $\approx 7957.72MB/s$

1.4 Linear Regression

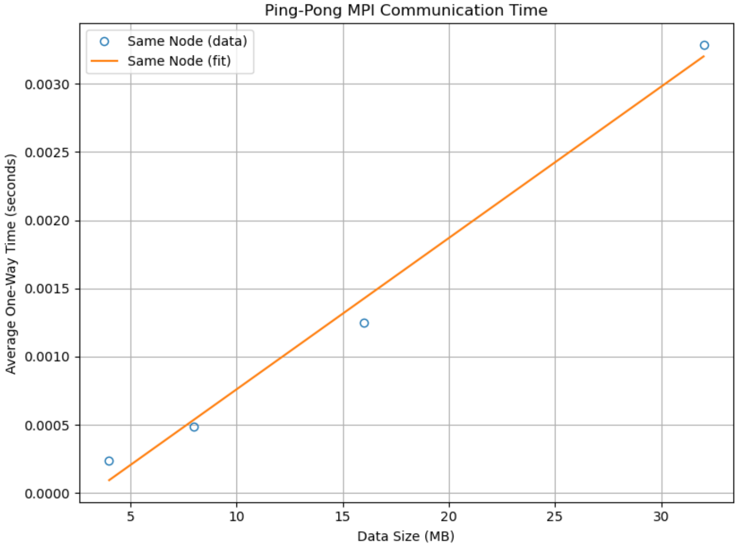

Fig. 1: Linear Regression for same

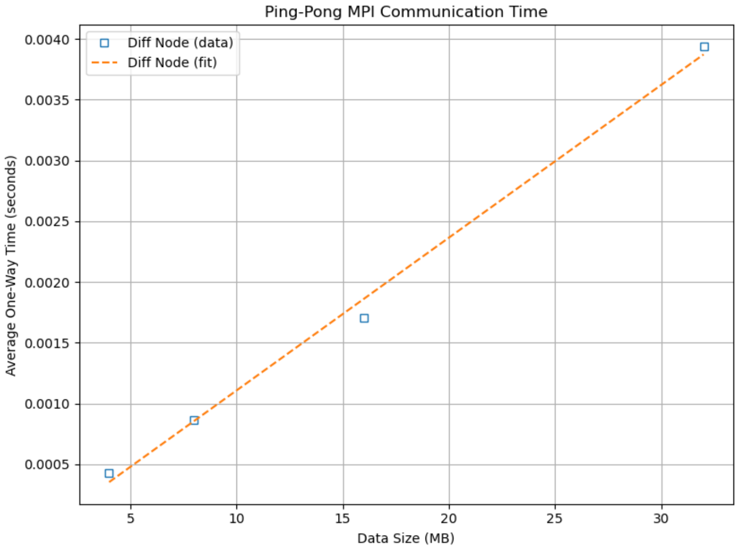

Fig. 2: Linear Regression for diff

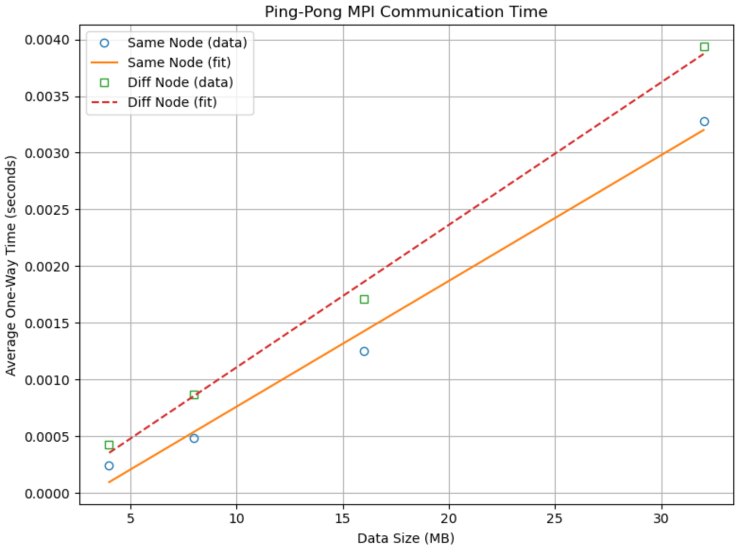

Fig. 3: Linear Regression for both methods

2 Problem 2

2.1 Wall-clock Time Table

Table 2: Problem 2 Wall-clock Time Table

Array size	262144	524288	1048576
serial	0.000902495	0.001672294	0.003332579
2 processes	0.002470	0.005453	0.008972
4 processes	0.003374	0.004037	0.008923
8 processes	0.003583	0.003975	0.008298

2.2 Speedup

Table 3: Problem 2 Speedup Table

Array size	262144	524288	1048576
2 processes	0.3652	0.3067	0.3714
4 processes	0.2673	0.4143	0.3733
8 processes	0.2518	0.4205	0.4016

2.3 Efficiency

Table 4: Problem 2 Efficiency Table

Array size	262144	524288	1048576
2 processes	0.1826	0.1534	0.1857
4 processes	0.0668	0.1036	0.0933
8 processes	0.0315	0.0526	0.0502

2.4 Discussion

Speedup is sub-linear due to communication overhead in MPI. Efficiency drops with increasing process count. With 8 processes and 262144 elements, we've got around 3% efficiency, which suggests that parallel overhead dominates the computation.

The program shows poor scalability especially for small vector sizes. An MPI limitation here is, for lightweight computations, the communication overhead quickly outweighs parallel benefits. Scalability improves slightly with larger input sizes but remains insufficient.

3 Problem 3

3.1 Runtimes

Table 5: Problem 3 Runtimes

Array size	262144	524288	1048576
4 processes on the same node	0.019300	0.040307	0.084053
4 processes on 4 different nodes	0.026653	0.042971	0.112452

3.2 Speedup

Table 6: Problem 3 Speedup

Array size	262144	524288	1048576
Diff vs Same Speedup	0.7245	0.9370	0.7474

3.3 Efficiency

Table 7: Problem 3 Efficiency

Array size	262144	524288	1048576
Efficiency (Diff Node, /4)	0.1811	0.2343	0.1868

3.4 Discussion

MPI Merge Sort shows better scalability on the same node due to reduced communication latency and faster memory access.

Performance drops on different nodes, see in timing and efficiency metrics. This aligns with the other problems results, meaning that distributed-node latency and bandwidth penalty strongly algorithms. Moreover, merge sort algorithm has a significant data exchange.

4 Problem 4

4.1 Runtimes

Table 8: Problem 4 Runtimes

Array size	1	2	4	8	16
Runtime	0.002640	0.001349	0.000798	0.000388	0.000290

4.2 Speedup

Table 9: Problem 4 Speedup

Processes	1	2	4	8	16
Speedup	1.0000	1.9570	3.3075	6.8041	9.1034

4.3 Efficiency

Table 10: Problem 4 Efficiency

Processes	1	2	4	8	16
Efficiency	1.000	0.978	0.827	0.850	0.569

4.4 Discussion

The Monte Carlo pi estimation program in strongly scalable. There is small communication overhead, pracitcal speedup around the theorical speedup and a high efficiency.