Name: Brijesh Mavani
CWID: A20406960
University: Illinois Institute of Technology
Course: Parallel and Distributed Processing

Final Project - Result Report

This document only contains results and their analysis. The implementation details are provided in design document (CS546_Mavani_Brijesh_Project_DesignReport.pdf). The results of the execution of each implementation are below. The speed up (Sp) can be calculated by Ts/Tp.

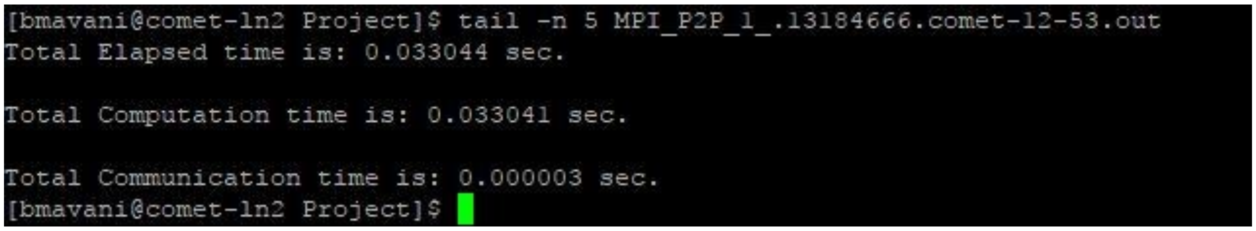
Efficiency (Ep) = Sp/P where P is the number of processors.

- **MPI send and receive (Part A):**

The MPI communication operations were performed using the MPI_Send and MPI_Recv functions. The MPI function MPI_Wtime() has been used to capture the timing information. The communication and computation timing details are provided below:

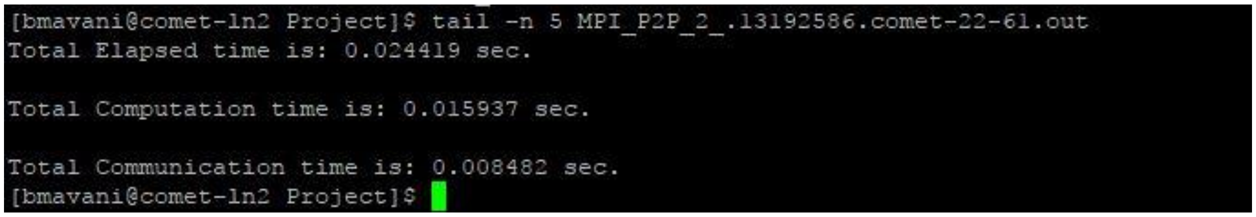
Number of processors	Total Elapsed Time (ms)	Computation Time (ms)	Communication Time (ms)	Speedup (Ts/Tp)	Efficiency (Sp/P)
1	33.044	33.044	0	-	-
2	24.419	15.937	8.482	1.35	0.675
4	20.047	8.235	10.811	1.65	0.413
8	18.506	6.533	11.973	1.79	0.224

1) 1 processor: As this is executed on a single processor, we can consider this as Ts. So, Ts= 33.044 ms.



```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_P2P_1_.13184666.comet-12-53.out
Total Elapsed time is: 0.033044 sec.

Total Computation time is: 0.033041 sec.

Total Communication time is: 0.000003 sec.
[bmavani@comet-ln2 Project]$
```

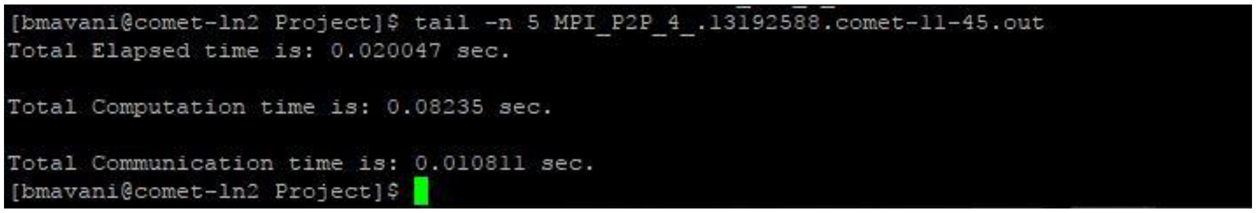
2) 2 processors: As this is executed in parallel, we can consider it as a Tp. Tp = 24.419 ms.



```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_P2P_2_.13192586.comet-22-61.out
Total Elapsed time is: 0.024419 sec.

Total Computation time is: 0.015937 sec.

Total Communication time is: 0.008482 sec.
[bmavani@comet-ln2 Project]$
```

Sp = 33.044 /24.419 = 1.35
Ep = 1.35/2 = 0.675

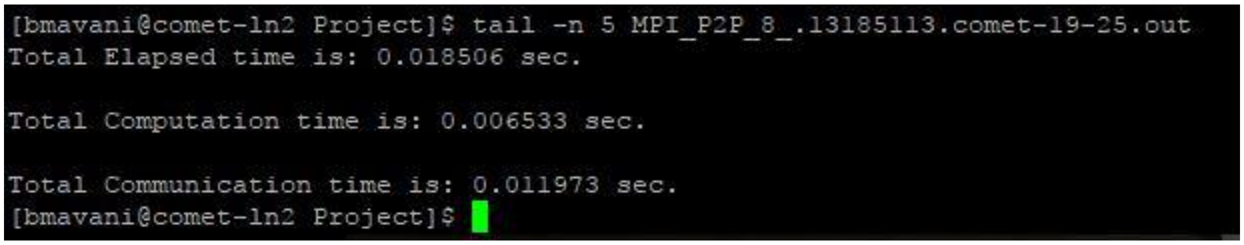
3) 4 processors: As this is executed in parallel, we can consider it as a Tp. Tp = 20.047 ms.



```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_P2P_4_.13192588.comet-11-45.out
Total Elapsed time is: 0.020047 sec.

Total Computation time is: 0.08235 sec.

Total Communication time is: 0.010811 sec.
[bmavani@comet-ln2 Project]$
```

Sp = 33.044/20.047 = 1.65
Ep = 1.65 /4 = 0.413

4) 8 processors: As this is executed in parallel, we can consider it as a Tp. Tp = 18.506 ms.



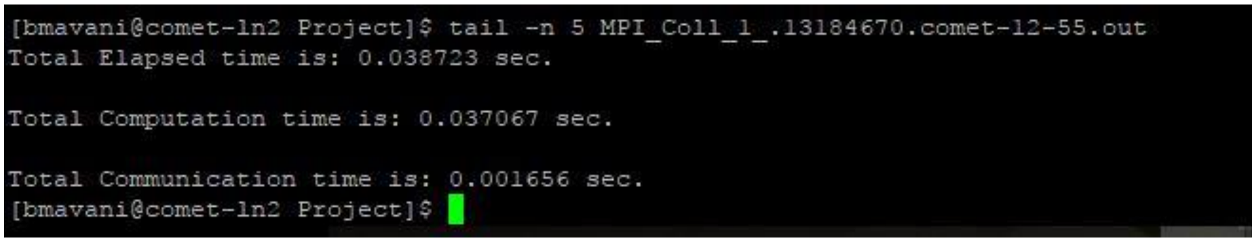
```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_P2P_8_.13185113.comet-19-25.out
Total Elapsed time is: 0.018506 sec.

Total Computation time is: 0.006533 sec.

Total Communication time is: 0.011973 sec.
[bmavani@comet-ln2 Project]$
```

Sp = 33.044/18.506 = 1.79
Ep = 1.79 /8 = 0.224

- **MPI Collective (Part B):**
  As we can see in both tables (Part A and Part B), there is not much difference in timings while implementing the program using MPI point to point communication and MPI Collective communications.
  Some differences can be seen in the communication. From the experiments, it can be inferred that the collective calls improve communication performance when the number of processes grows, but as execution done with maximum 8 processors, this is a too small sample to reach conclusions.

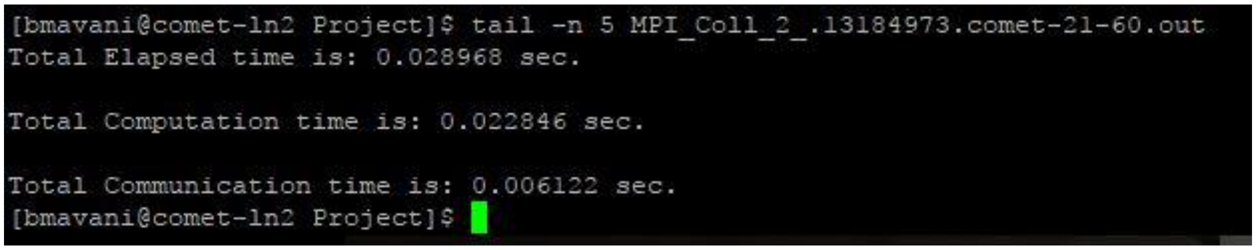
Number of processors	Total Elapsed Time (ms)	Computation Time (ms)	Communication Time (ms)	Speedup(Ts/Tp)	Efficiency (Sp/P)
1	38.723	37.067	1.656	-	-
2	28.968	22.846	6.122	1.34	0.67
4	23.768	15.593	10.575	1.63	0.408
8	21.513	10.001	11.512	1.8	0.225

1) 1 processor: As this is executed on the single processor, we can consider this as Ts. So, Ts= 38.723 ms.



```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_Coll_1_.13184670.comet-12-55.out
Total Elapsed time is: 0.038723 sec.

Total Computation time is: 0.037067 sec.

Total Communication time is: 0.001656 sec.
[bmavani@comet-ln2 Project]$
```

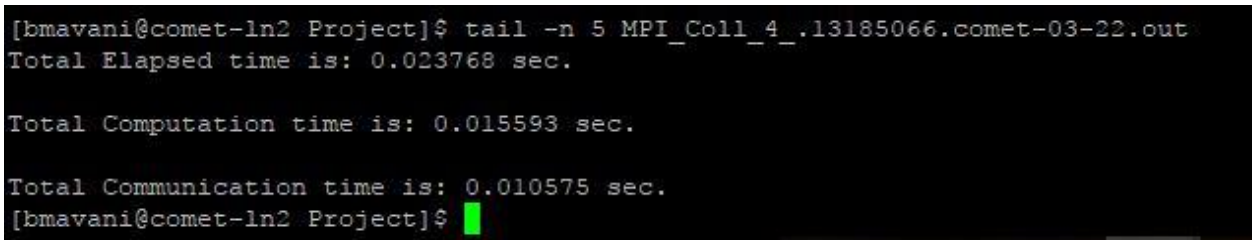
2) 2 processors: As this is executed in parallel, we can consider it as a Tp. Tp = 28.968 ms.



```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_Coll_2_.13184973.comet-21-60.out
Total Elapsed time is: 0.028968 sec.

Total Computation time is: 0.022846 sec.

Total Communication time is: 0.006122 sec.
[bmavani@comet-ln2 Project]$
```

Sp = 38.723 /28.968 = 1.34
Ep = 1.34/2 = 0.67

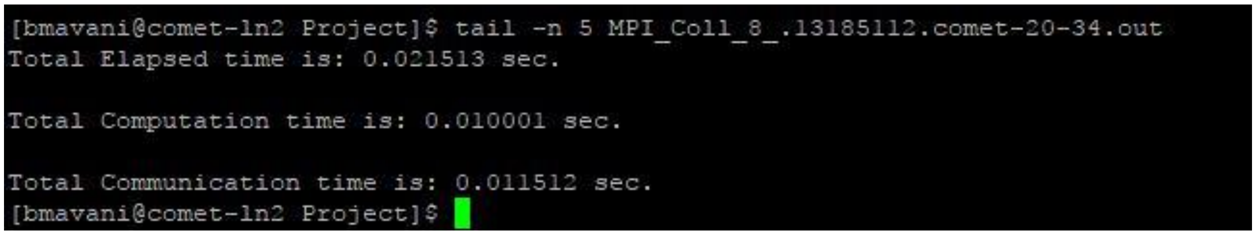
3) 4 processors: As this is executed in parallel, we can consider it as a Tp. Tp = 23.768 ms.



```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_Coll_4_.13185066.comet-03-22.out
Total Elapsed time is: 0.023768 sec.

Total Computation time is: 0.015593 sec.

Total Communication time is: 0.010575 sec.
[bmavani@comet-ln2 Project]$
```

Sp = 38.723 / 23.768 = 1.63
Ep = 1.63 /4 = 0.408

4) 8 processors: As this is executed in parallel, we can consider it as a Tp. Tp = 21.513 ms.



```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_Coll_8_.13185112.comet-20-34.out
Total Elapsed time is: 0.021513 sec.

Total Computation time is: 0.010001 sec.

Total Communication time is: 0.011512 sec.
[bmavani@comet-ln2 Project]$
```

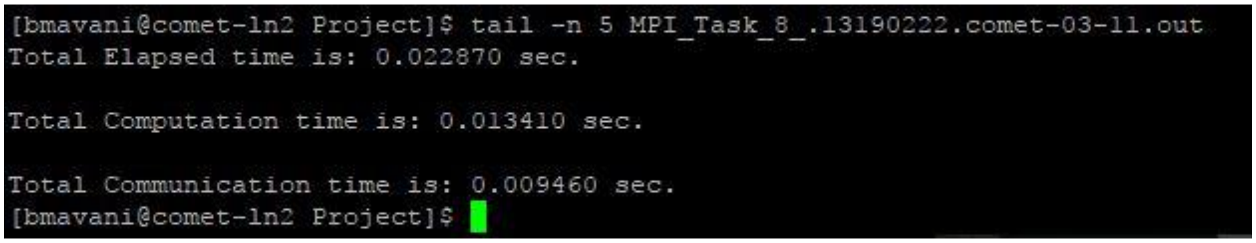
Sp = 38.723 /21.513 = 1.8
Ep = 1.8 /8 = 0.225

- **Task and Data Parallel Model (Part C):**
  From the algorithm provided in design document, it can be inferred that it requires more communication. The processors in P3 needs the input from P1 and P2 both processor groups while earlier there wasn't any need for this communication. Also, some processors will be idle while other processors busy computing or communicating with another processor group.

  Computation and communication timings for P1=P2=P3=P4=2 are as below:

Number of processors	Total Elapsed Time (ms)	Computation Time (ms)	Communication Time (ms)	Speedup (Ts/Tp)	Efficiency (Sp/P)
P1=P2=P3=P4=2	22.870	13.410	9.460	33.044/22.870 = 1.44	1.44/8 = 0.18

1) 8 processors: As this is executed in parallel, we can consider it as a Tp. Tp = 22.870 ms.



```
[bmavani@comet-ln2 Project]$ tail -n 5 MPI_Task_8_.13190222.comet-03-11.out
Total Elapsed time is: 0.022870 sec.

Total Computation time is: 0.013410 sec.

Total Communication time is: 0.009460 sec.
[bmavani@comet-ln2 Project]$
```

Here, we consider the Ts will be the same as the Ts for part A i.e. Ts = 33.044

Sp = 33.044 /22.870 = 1.44
Ep = 1.44 /8 = 0.18

- **Comparison between cases A and C (Part D):**
  As we can see from the results of cases A and C, case A performs better in this particular implementation. This is due to the fact that in case A all processors are busy in either computing or communicating. Whereas in C some processors will be idle/free while other processors busy computing or communicating with another processor group. The performance of the case C could be different if there are multiple jobs to work on and all processor groups work on different jobs. This will make all processor busy with some task. However, this needs to be evaluated further but we can suppose the performance will be improved as it will make all processors to work on jobs and keep them busy.

- **Conclusion:**
  As shown above MPI point to point communication and Task & data parallelism provides approximately same performance in this implementation. The best speedup has been seen in

collective communication with speedup of 1.8. However, the point to point communication is also providing approximately same speedup with 1.79. In theory, the best speedup would be equal to the number of processors. Here, we are getting speedup of 1.79/1.8. This is a sub-optimal speedup yet compared to sequential algorithm this is still better.