PARALLEL QUICKSORT

Prepared and Presented by: Steven Pittaro

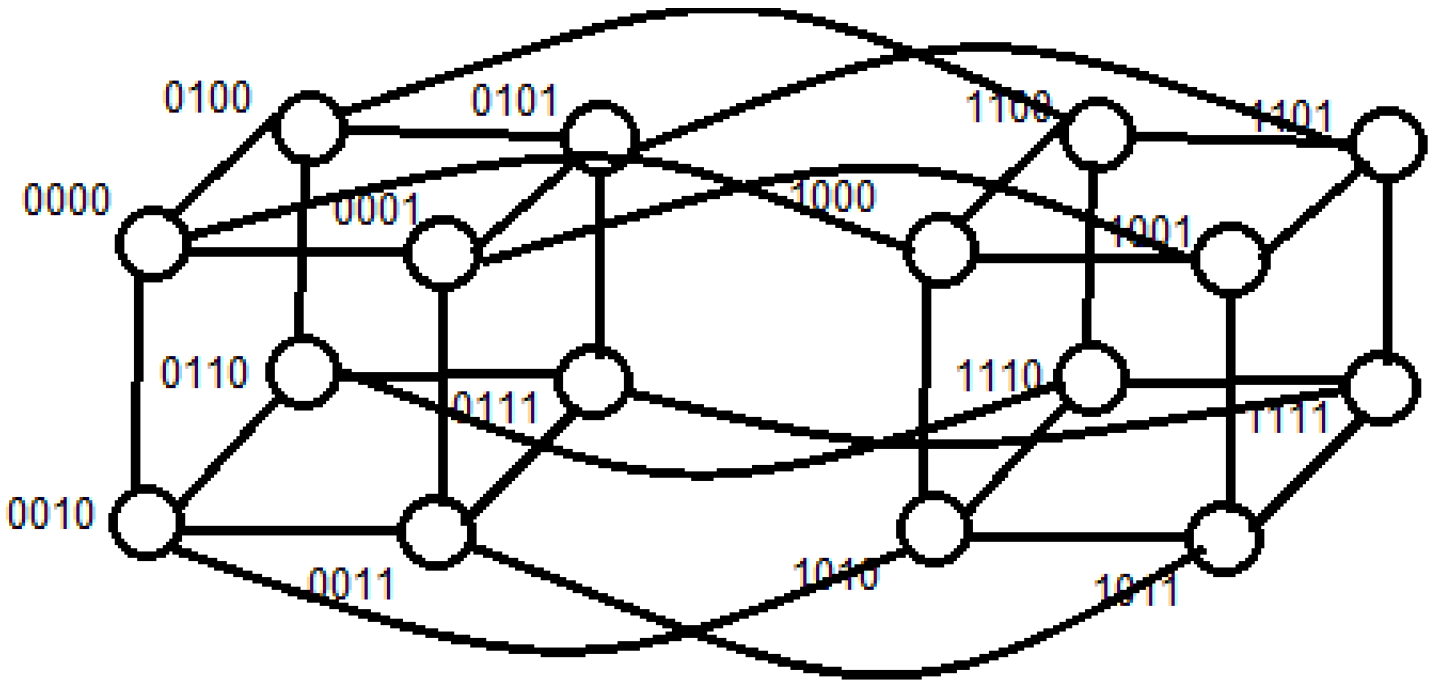
University at Buffalo The State University of New York

- Parallel Implementation
- Amdahl Time Charts
- Gustafson's Time Charts
- Observations
- Questions

Proposed Parallel Implementation

- Split your data into two halves according to the splitter you have
- Check to see if you are in the upper or lower half of processes for your division (EX: if you are >31 or <=31 for 64 processes)
- If you are in the upper half send your lower data to your partner in the lower half, and vice versa for if you are in the lower half.
- Combine the new data on each process.
- Repeat, step 1 going down each significant bit as you recurse.

Visualized…


0100
0101
1100
1101
0000
0001
1000
1001
0110
0111
1110
1111
0010
0011
1010
1011

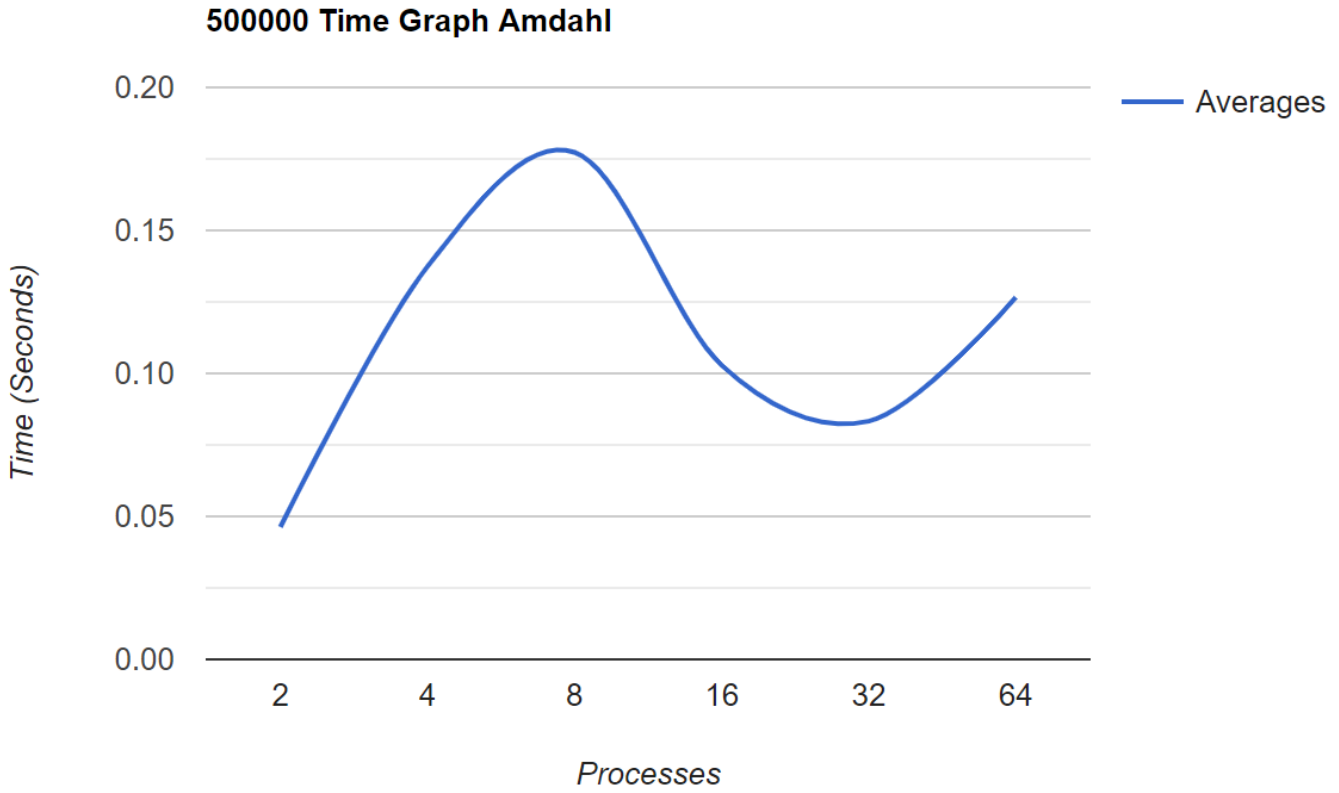
How Data Was Gathered

- Run On CCR using the intel-mpi/4.1.3 module
- Ran each Data Point/process number pair 6 times
- Got a time output from each process and averaged them together for one runtime
- Averaged together the 6 runtime totals to get the final average runtime for the chart
- Requested exclusive process by running

  fisbatch --nodes=8 --ntasks-per-node=8 --time=2:30:00
  – partition=general-compute --qos=general-compute --exclusive

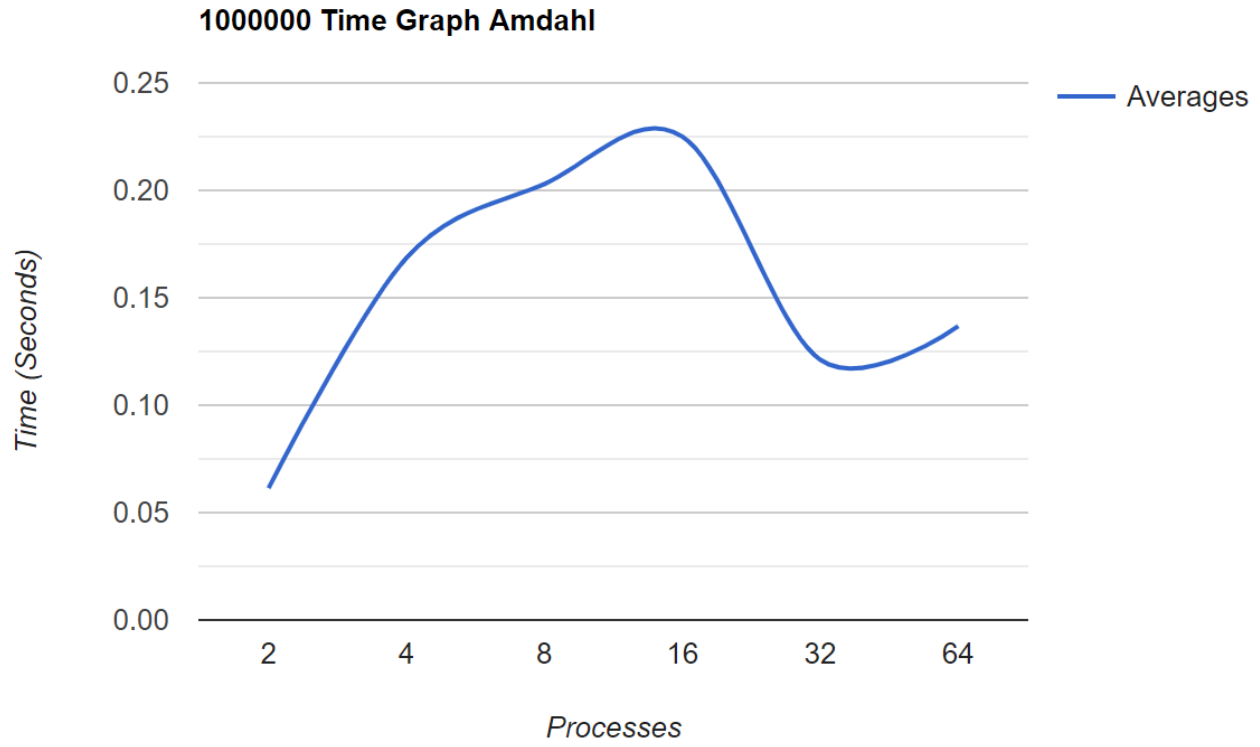
500,000 Data Points

2	0.04631
4	0.137358
8	0.177292
16	0.102867
32	0.0833
64	0.126603



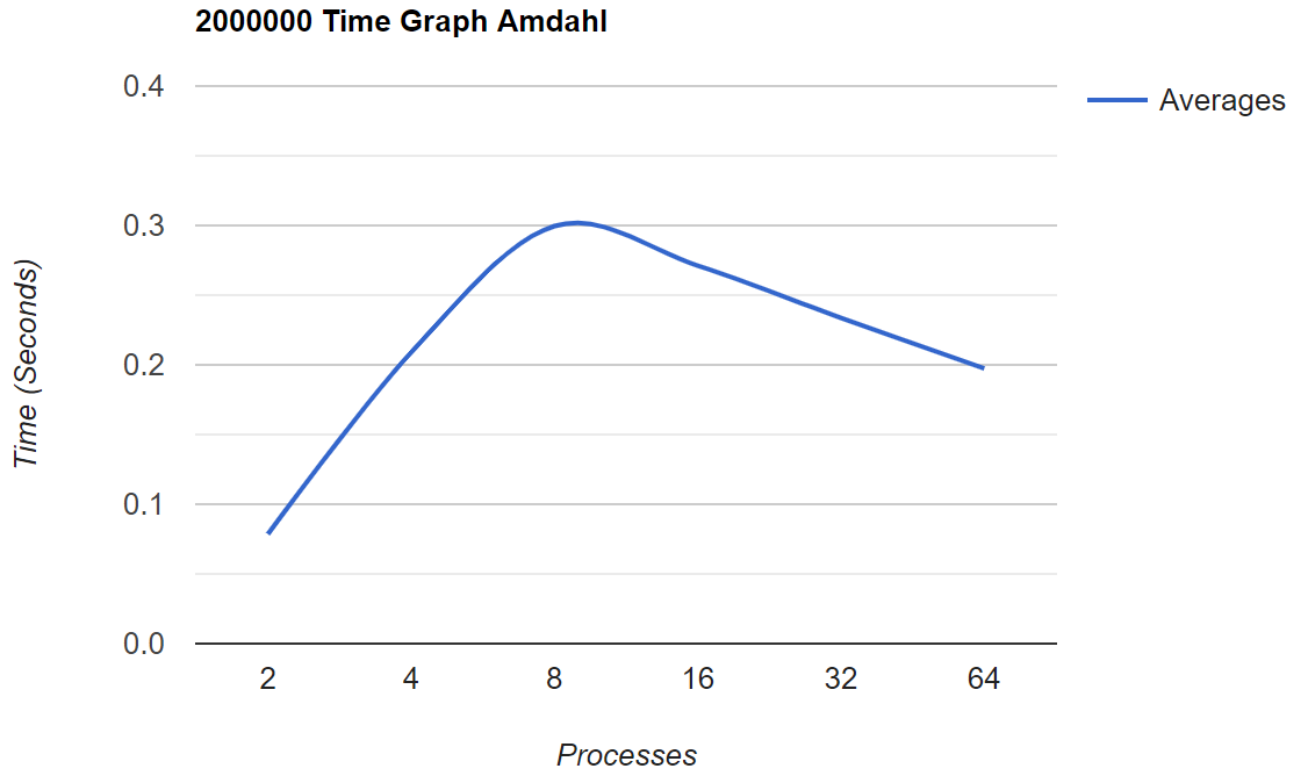
1,000,000 Data Points

2	0.061422
4	0.168529
8	0.202968
16	0.225062
32	0.121137
64	0.136793



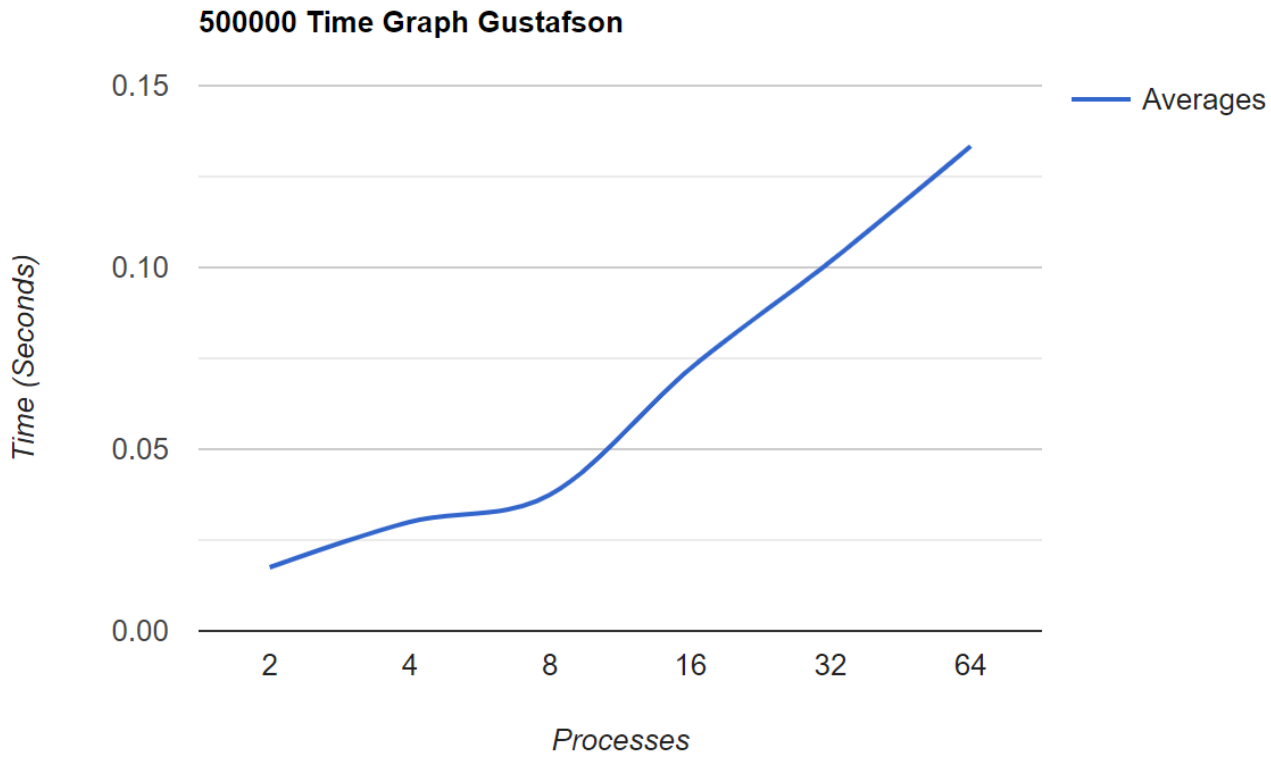
2,000,000 Data Points

2	0.078498
4	0.208954
8	0.299576
16	0.271205
32	0.233671
64	0.197424



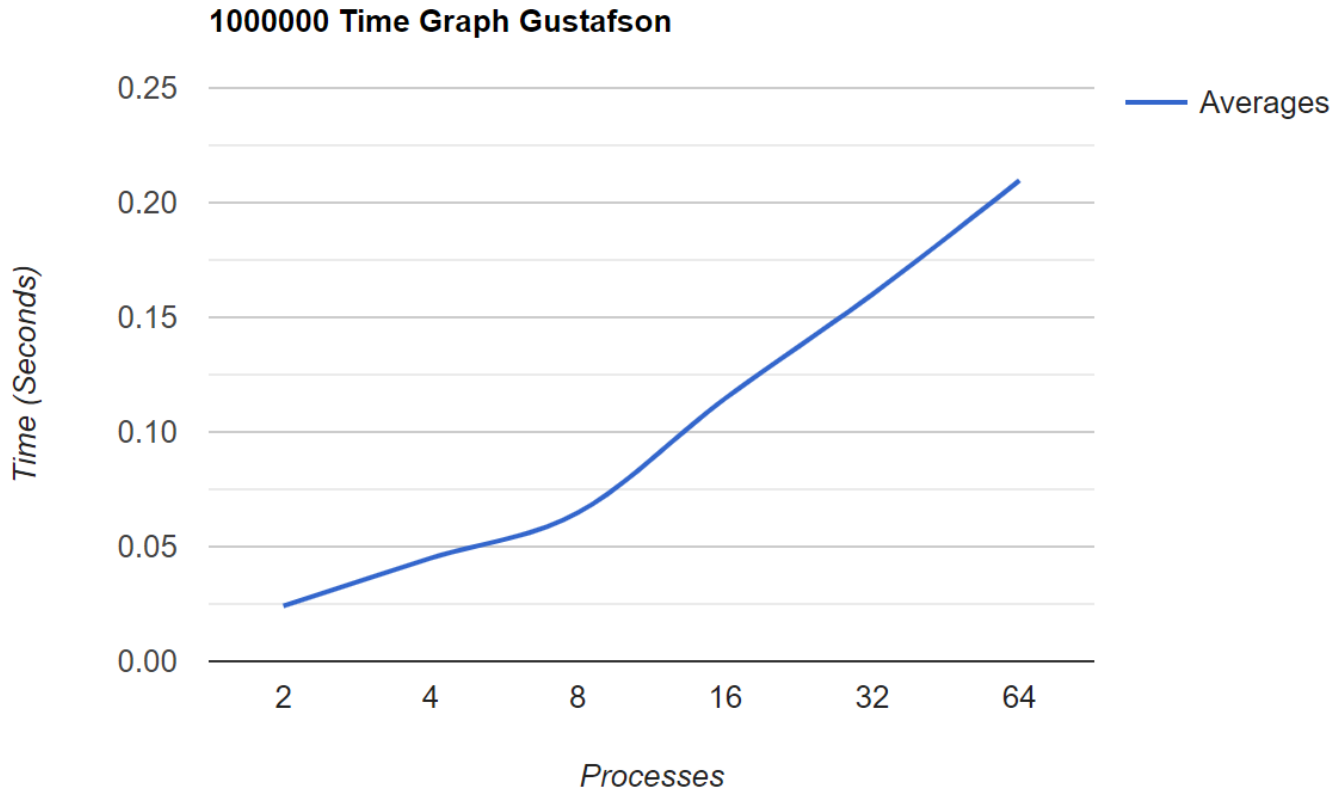
500,000 Data Points

2	0.0175
4	0.03
8	0.0375
16	0.072188
32	0.101667
64	0.133385



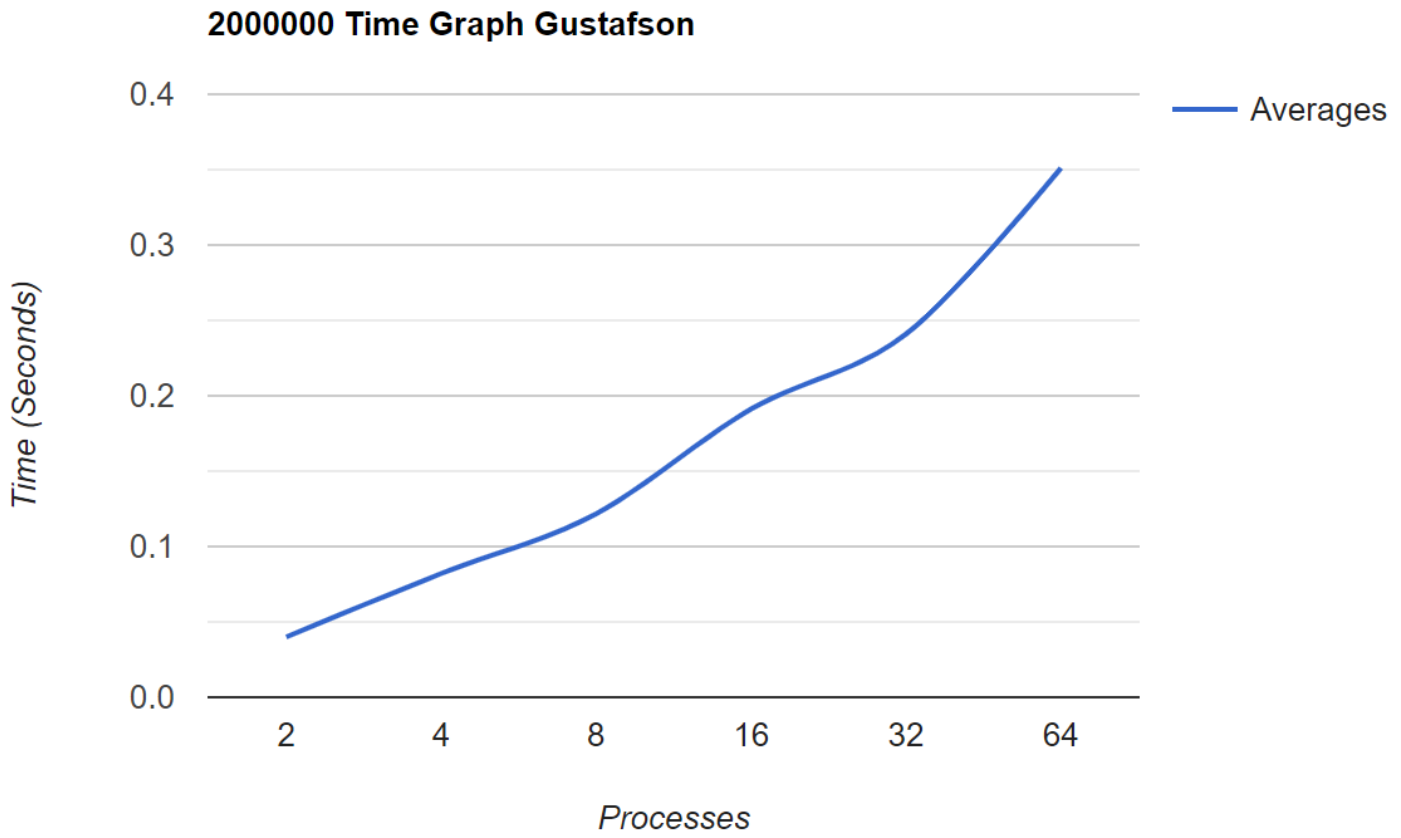
1,000,000 Data Points

2	0.024167
4	0.045
8	0.064792
16	0.114583
32	0.16
64	0.209714



2,000,000 Data Points

2	0.04
4	0.082083
8	0.121667
16	0.19125
32	0.240887
64	0.351094



Observations

- Code can likely be optimized further because of the data trends
- Sending overhead is large, so that can affect results depending on data distribution
- Data distribution is not optimal, as in you cannot tell how much data will end up on each process
- A reason that the two processes could run faster than the higher processes is due to it being less steps to separate the data into its halves as in the implementation the maximum amount of runs is $\log_{10}(n)$ where n is the number of processes or if there is 1 process per division, as such for the lower amount of processes, they will finish in fewer steps.

References used during project

- Algorithms Sequential and Parallel: A Unified Approach
  - Russ Miller & Laurence Boxer

QUESTIONS?

University at Buffalo The State University of New York