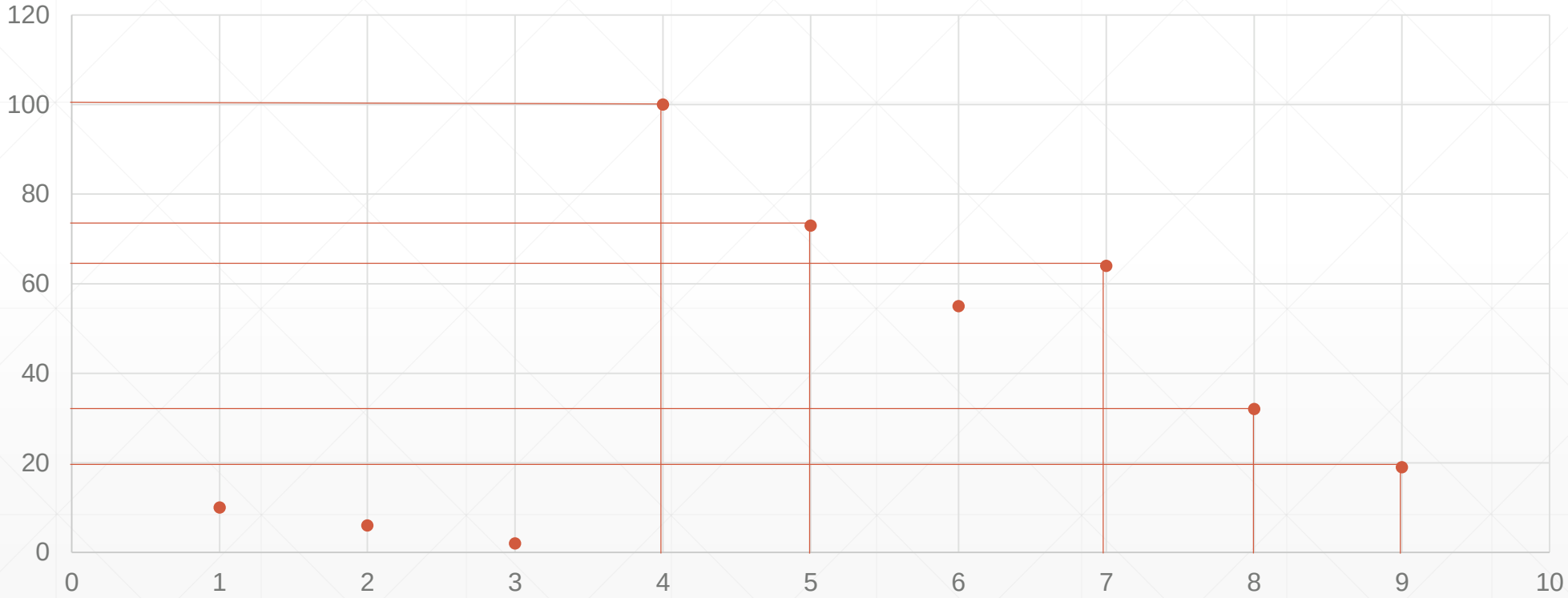
Point Domination Query

Hariharan Kumar

What is Point Domination Query??

- Subset of computational geometry.
- The algorithm gives the dominating points from a given set of points.
- A point P(x1, y1) is said to be dominating another point Q(x2, y2) if and only if the x1>x2 and y1>y2.

Point Domination Query



Input and Output

- Input: The input to the system is n number of points which contains x and y coordinates
- Output : The output from the system is the x and y coordinates of the points that are not dominated by any other points in the given set.

Assumptions & Implementation

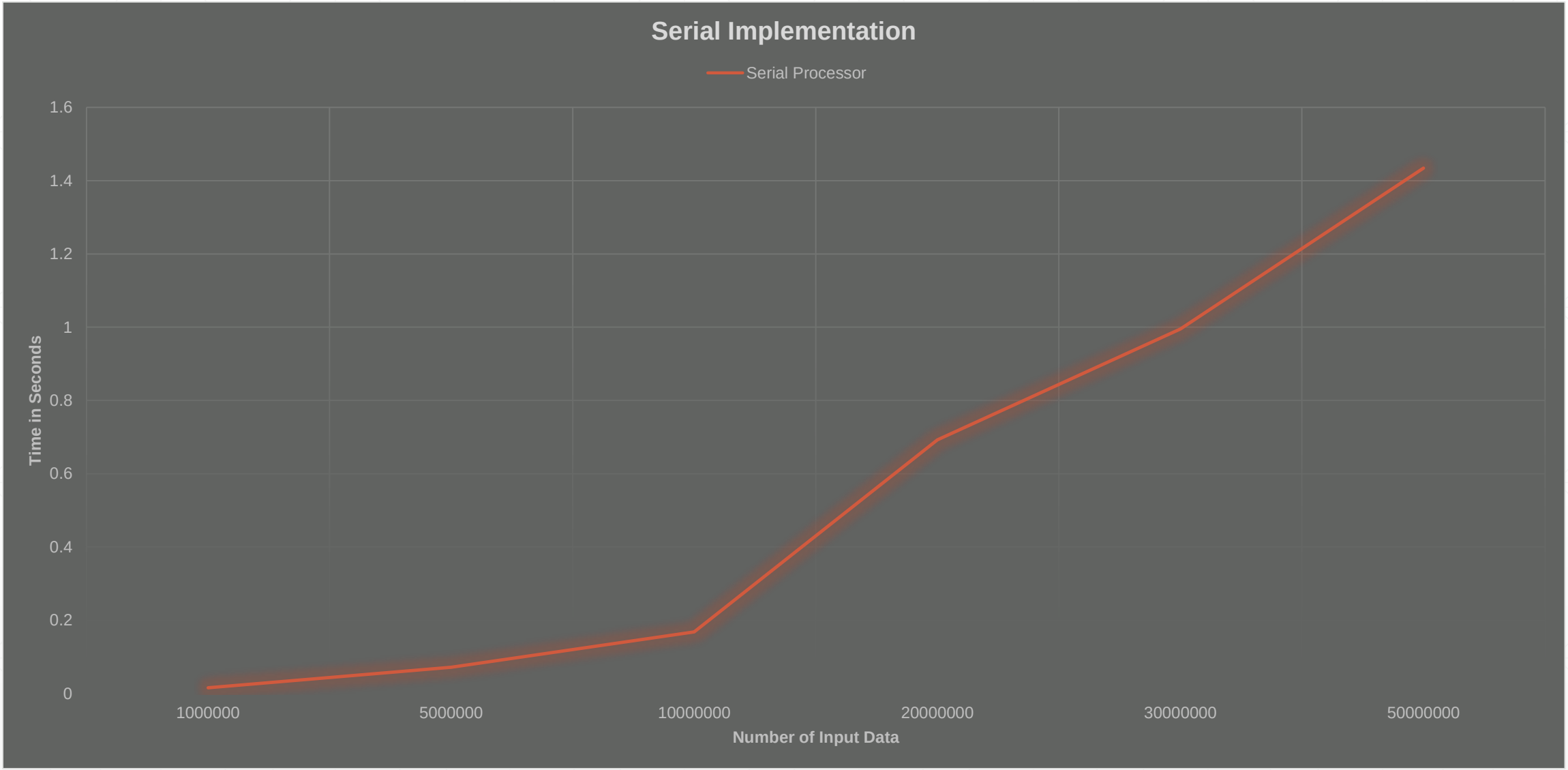
- No two points have same x and y coordinates.
- The input points are sorted with respect to the x coordinates and fed into the system.
- Two ways of implementing the algorithm:
  - Sequential
    - The algorithm is implemented in a sequential machine and run for different number of input datasets.
  - Parallel
    - In parallel implementation the algorithm is executed on P processors. The time taken is calculated and compared for varying P and number of input datasets.

Implementation continued…

- The implementation of the algorithm is done by parallel postfix operation on the given input set.
- The running time of the algorithm in RAM is O(N).
- In parallel implementation each processors gets equal number of data and they execute the algorithm sequentially and passes the result to the master processor which produces the final result.
- In parallel implementation the algorithm is implemented by performing parallel postfix from the last processor and the output is passed on to the next processor and so on to find the global output.
- The parallel implementation is done using MPI.

Serial Implementation Results

Number of Data	Running Time (sec)
1000000	0.0155
5000000	0.0714
10000000	0.1679
20000000	0.6922
30000000	0.9976
50000000	1.4345


Serial Implementation
Serial Processor
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Time in Seconds
1000000
5000000
10000000
20000000
30000000
50000000
Number of Input Data

Master/Worker Result

Number of Data	1 Master & 1 Worker	1 Master & 2 Worker	1 Master & 3 Worker	1 Master & 4 Worker	1 Master & 8 Worker
1000000	0.019354	0.009689	0.008404	0.007761	0.007386
5000000	0.075218	0.044784	0.034091	0.030230	0.031270
10000000	0.149679	0.087725	0.070841	0.061385	0.059326
20000000	0.301983	0.172144	0.142272	0.125485	0.119081
30000000	0.437095	0.259760	0.211865	0.183825	0.178045
50000000	0.769939	0.432947	0.357049	0.300703	0.298823

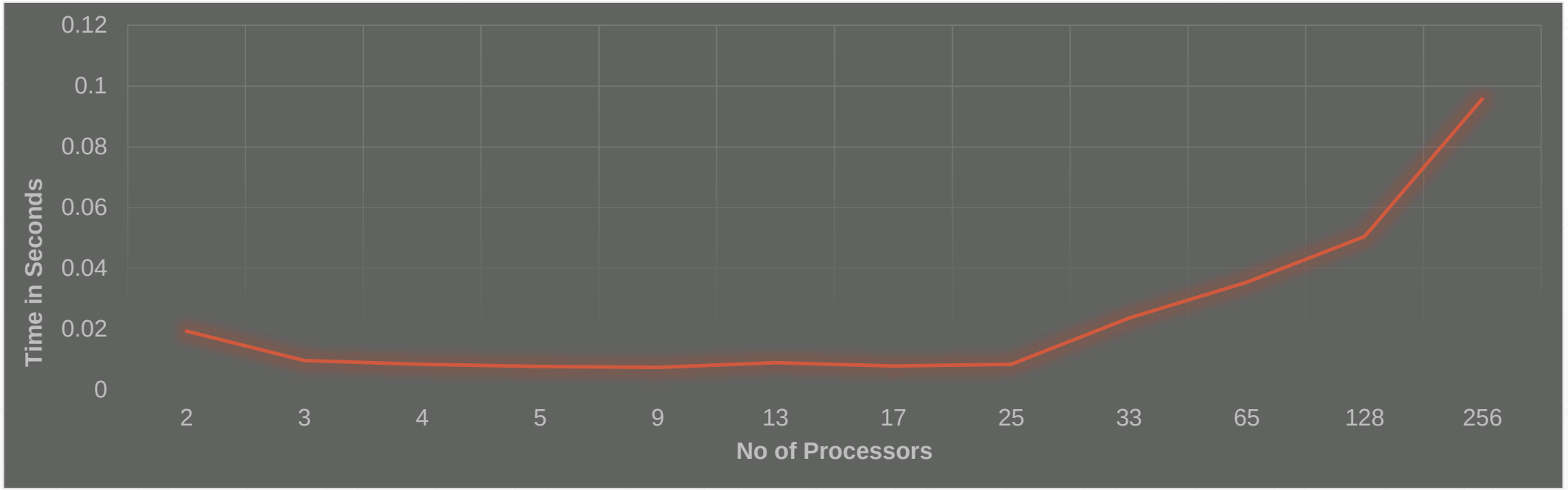
Results Continued…

Number of Data	1 Master & 12 Worker	1 Master & 16 Worker	1 Master & 24 Worker	1 Master & 32 Worker	1 Master & 64 Worker
1000000	0.009002	0.007867	0.008407	0.023659	0.035437
5000000	0.031031	0.035673	0.048335	0.096559	0.158106
10000000	0.059447	0.065279	0.070414	0.112851	0.196655
20000000	0.118770	0.121944	0.131146	0.205695	0.331923
30000000	0.182642	0.188615	0.188579	0.324964	0.484273
50000000	0.309990	0.310099	0.317954	0.462323	0.730854

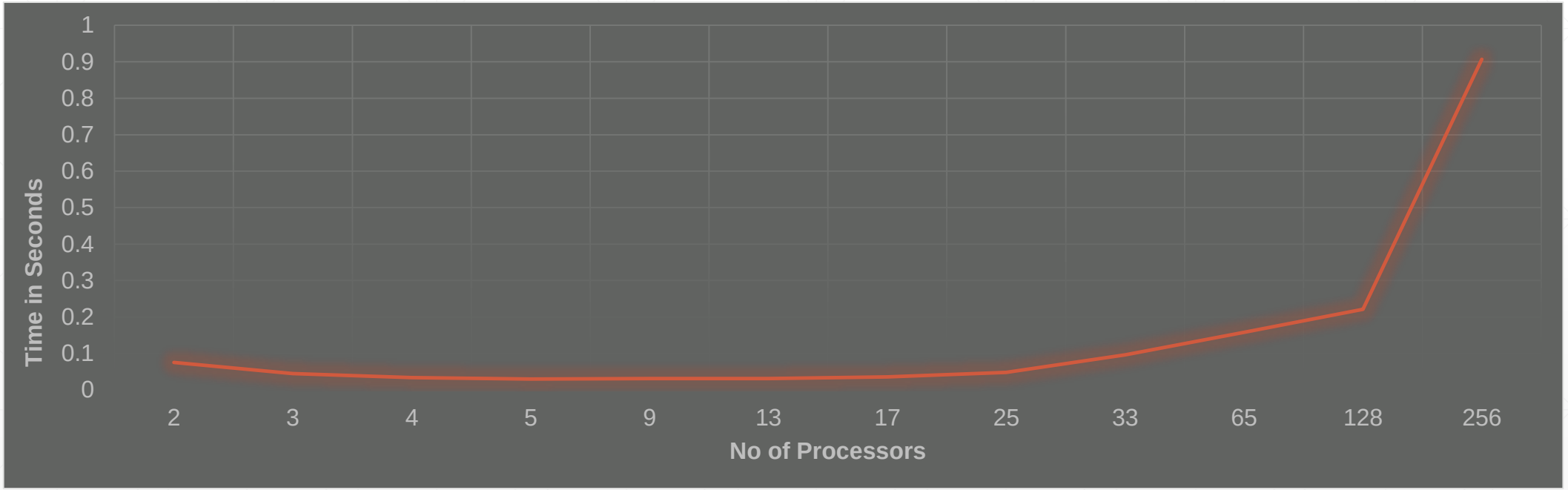
Results Continued…

Number of Data	1 Master & 127 Worker	1 Master & 255 Worker
1000000	0.050494	0.095789
5000000	0.220975	0.906955
10000000	0.317941	0.730254
20000000	0.660369	1.548903
30000000	1.078213	1.903219
50000000	1.627626	3.982638

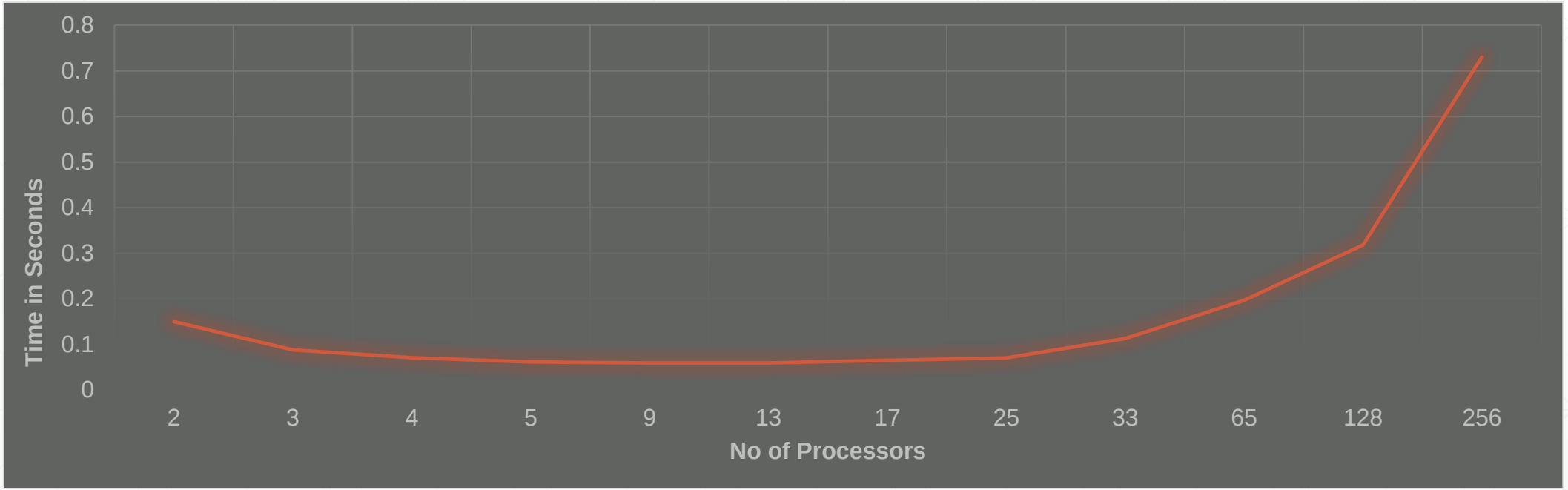
Master Worker Results for 1000000 data items



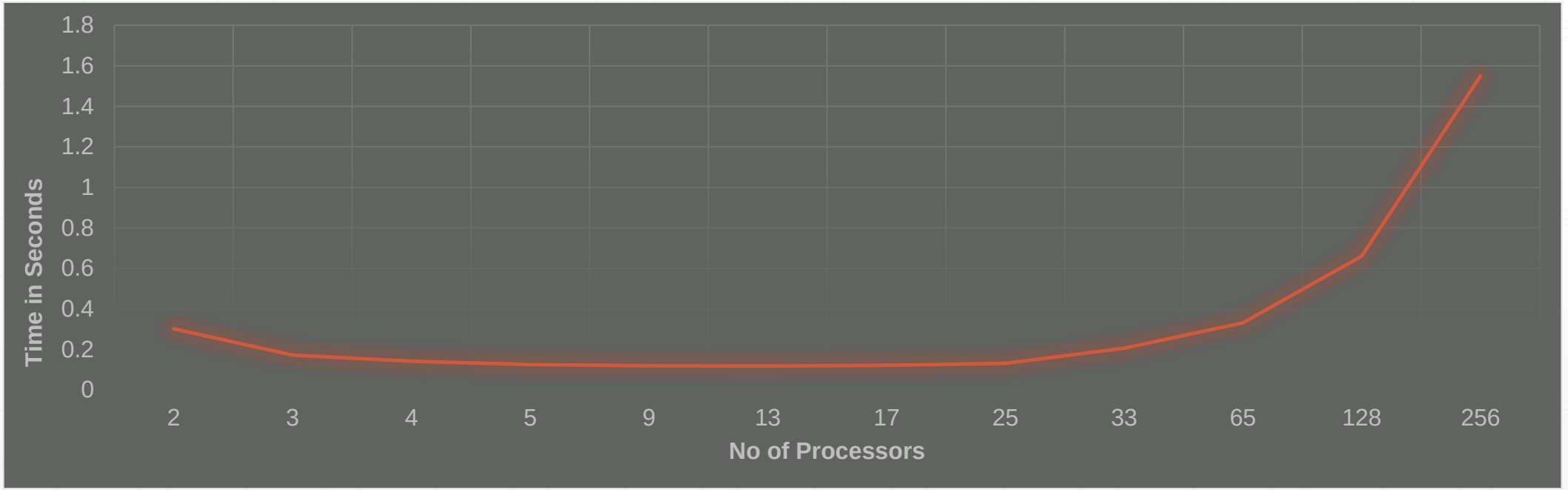
Master Worker Results for 5000000 data items



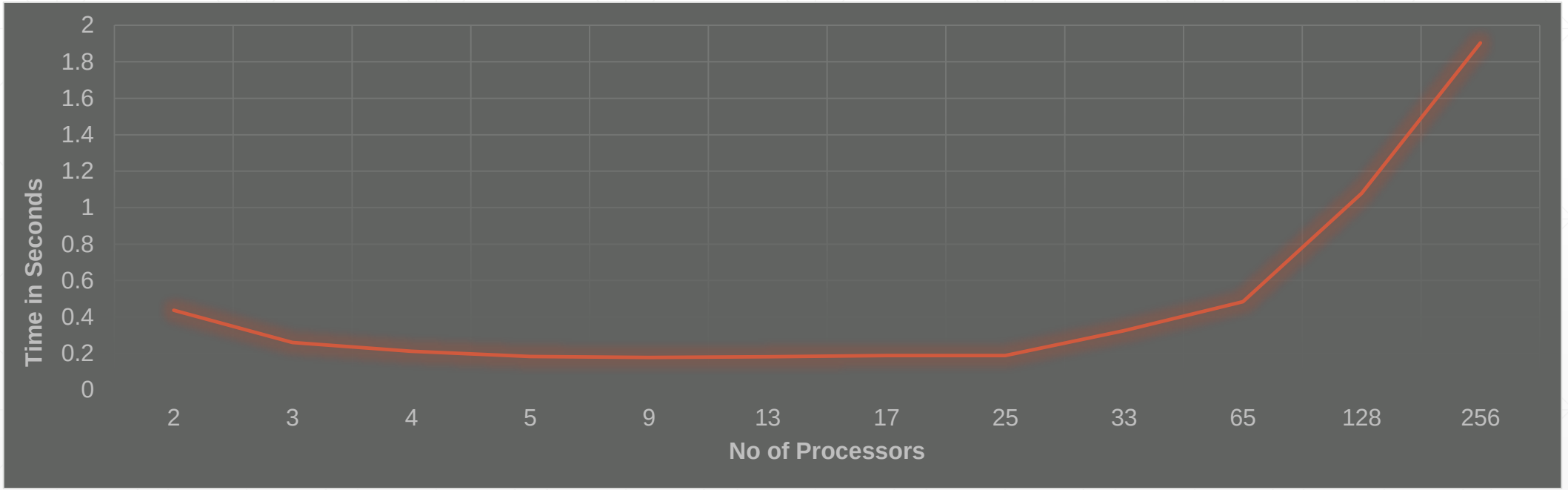
Master Worker Results for 10000000 data items



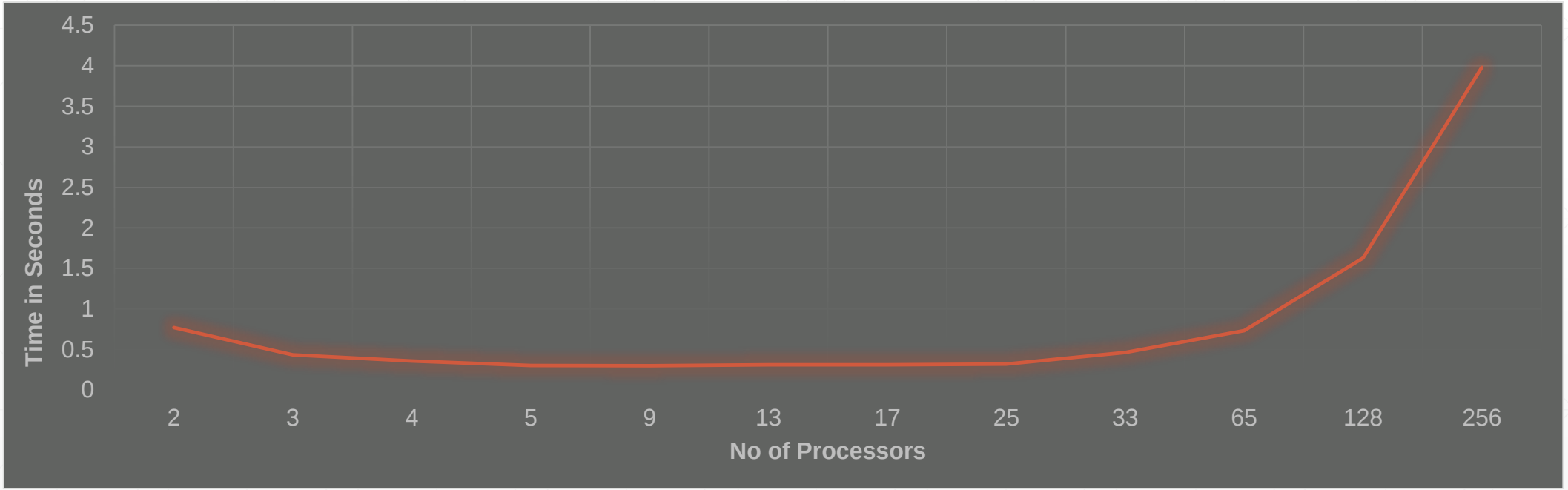
Master Worker Results for 20000000 data items



Master Worker Results for 30000000 data items



Master Worker Results for 50000000 data items



Master/Worker Result Analysis

- Initially regardless the number of the data processed the running time decreased when the number of processors increased.
- But it reached the lowest when the number of processor was 25 (24 Workers + 1 Master)
- This is evident as when the number of processors are increased more than the previous stated value the communication time over-shadows the actual processing time of the problem.
- This increase in time is contributed by the bottleneck that exists at the Master node that needs to receive all the prefix results from all the Workers.
- As the Master can only receive the results of one Worker at a time the other Worker have to wait and hence will result in wastage of time.
- And also the time taken by each processors to send the prefix results to the Master is higher than the each Workers' prefix operation time.

Improvements in Parallelization

- From the previous Master/Worker architecture's result it is clear that the increase in processor will result in decrease in the running time up to a certain number of processors.
- Increasing the number of processor more than the optimal value will result in increase in running time which is contributed by communication time needed to send the result of each Worker to the Master.
- This disadvantage is handled by parallelizing the problem by computing the prefix operation involving all the processors.
- For this method a Linear Array is simulated and the problem is solved by performing Postfix operation on the data in each processor in parallel.

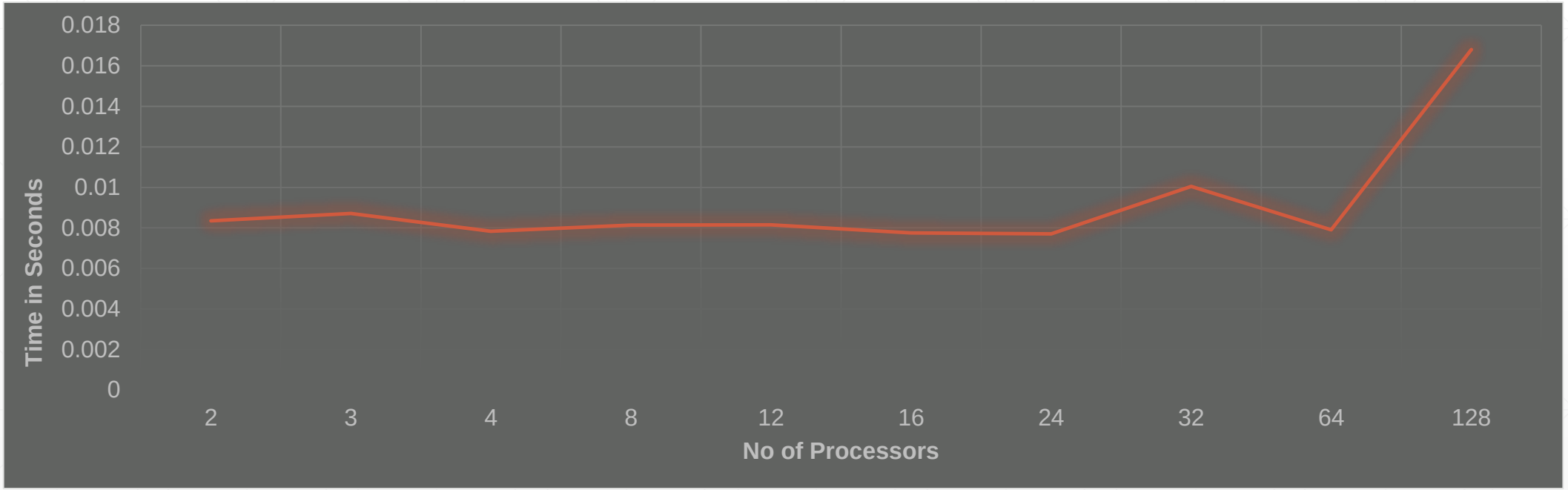
Parallel Postfix Result

Number of Data	Number of Processors - 2	3	4	8	12
1000000	0.008353	0.008715	0.007830	0.008143	0.008153
5000000	0.040988	0.040405	0.039386	0.039224	0.038282
10000000	0.079154	0.078881	0.079095	0.077840	0.081121
20000000	0.159838	0.158074	0.157444	0.155752	0.154681
30000000	0.239582	0.235706	0.235154	0.232836	0.206049
50000000	0.402122	0.394601	0.394381	0.387180	0.342925

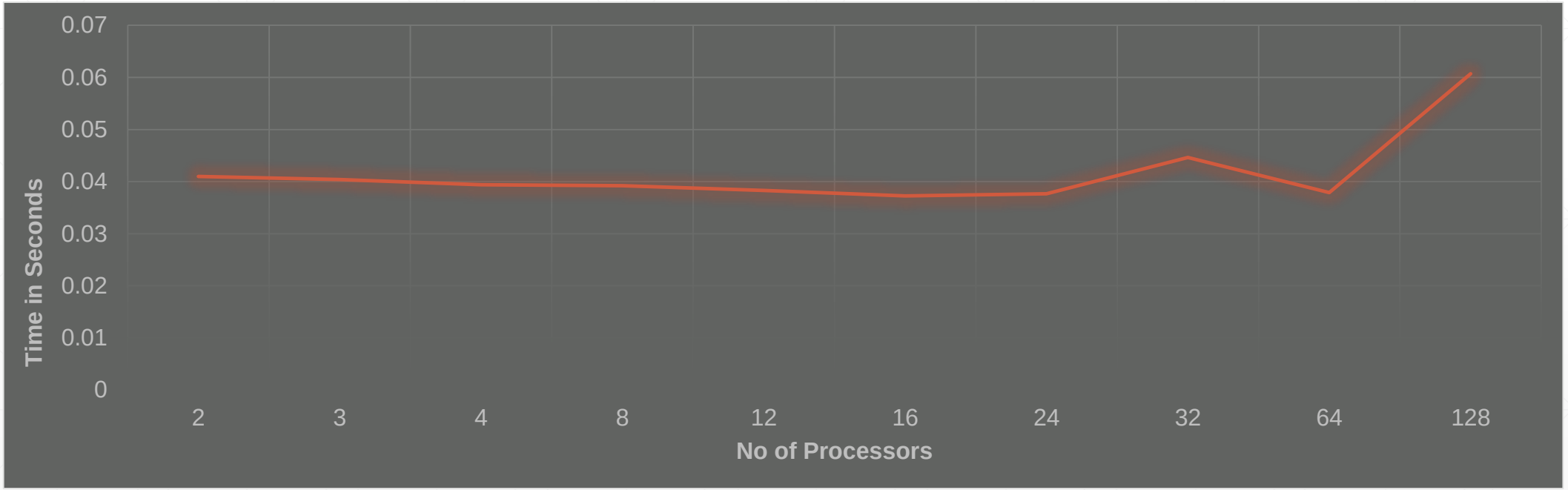
Results Continued…

Number of Data	No of Processors: 16	24	32	64	128
1000000	0.007755	0.007701	0.010043	0.007906	0.016810
5000000	0.037264	0.037659	0.044611	0.037893	0.060687
10000000	0.074712	0.073996	0.088027	0.074891	0.089128
20000000	0.149584	0.149476	0.149421	0.149352	0.159838
30000000	0.224234	0.221895	0.223822	0.224217	0.345100
50000000	0.358662	0.369374	0.373798	0.376584	0.343175

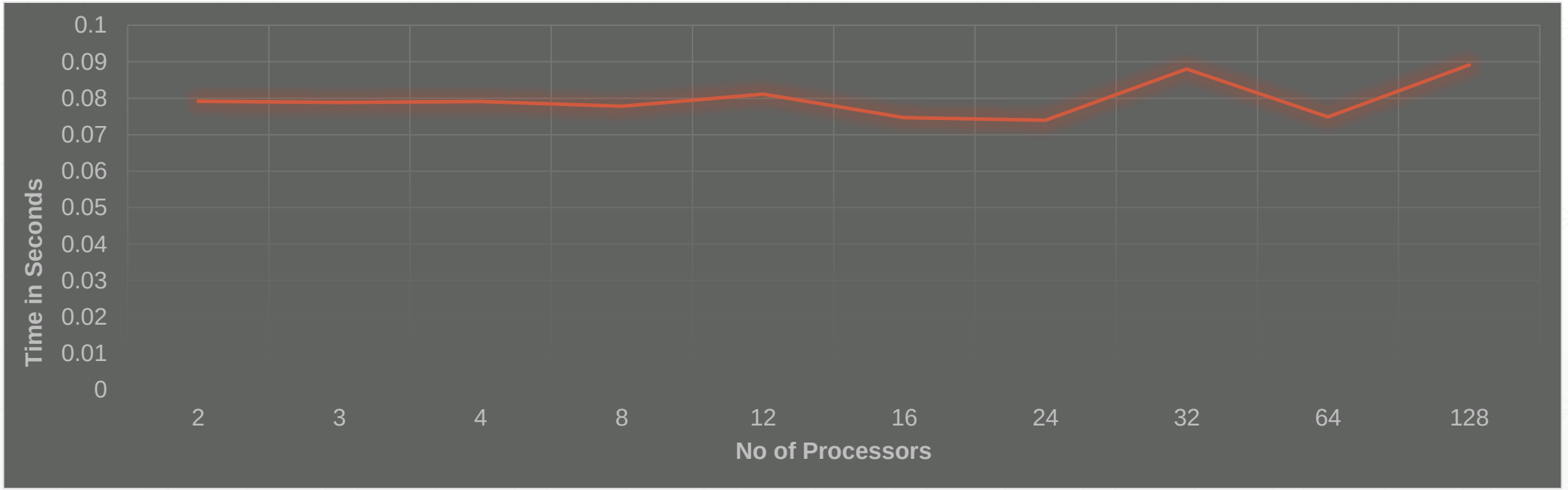
Parallel Postfix Results for 1000000 data items



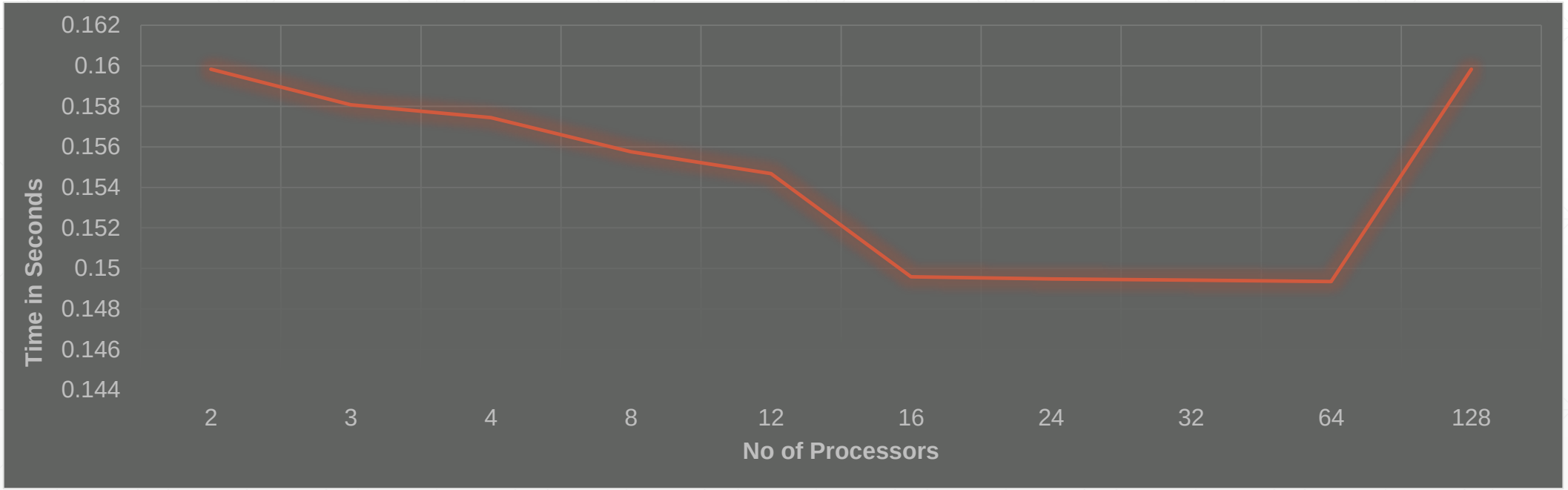
Parallel Postfix Results for 5000000 data items



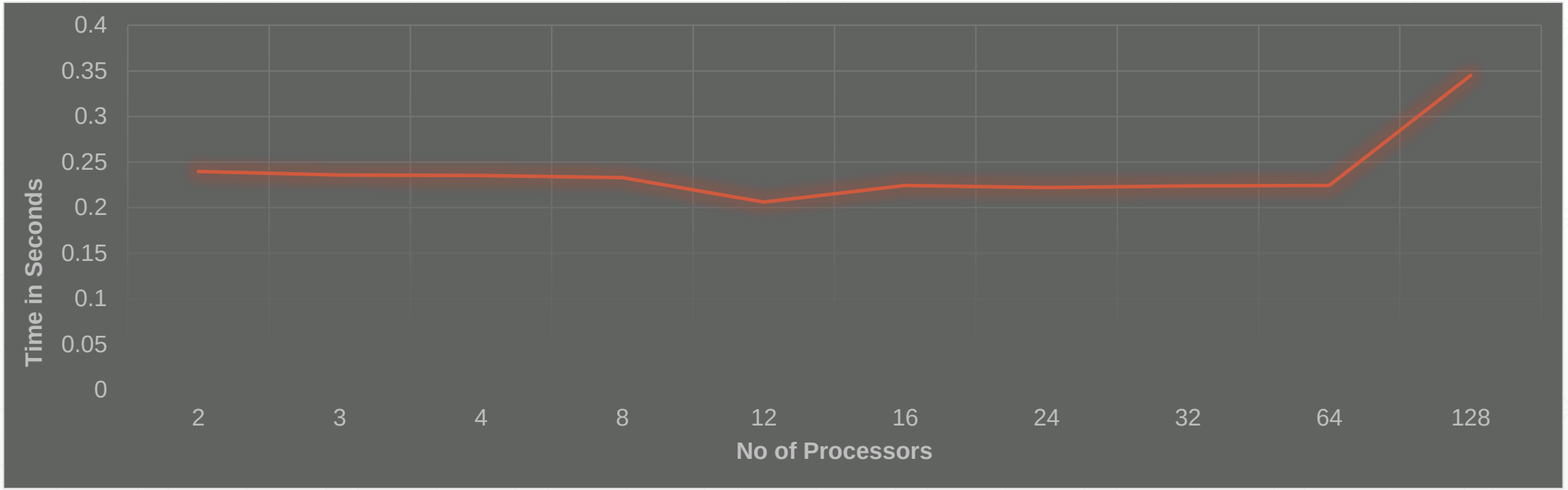
Parallel Postfix Results for 10000000 data items



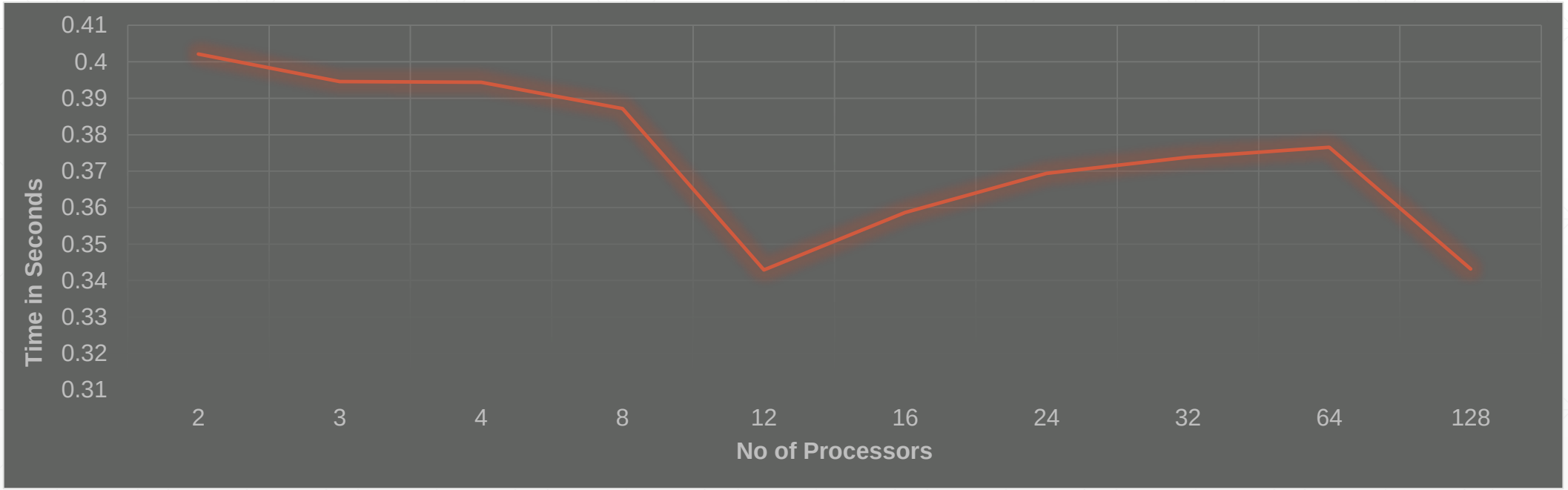
Parallel Prefix Results for 20000000 data items



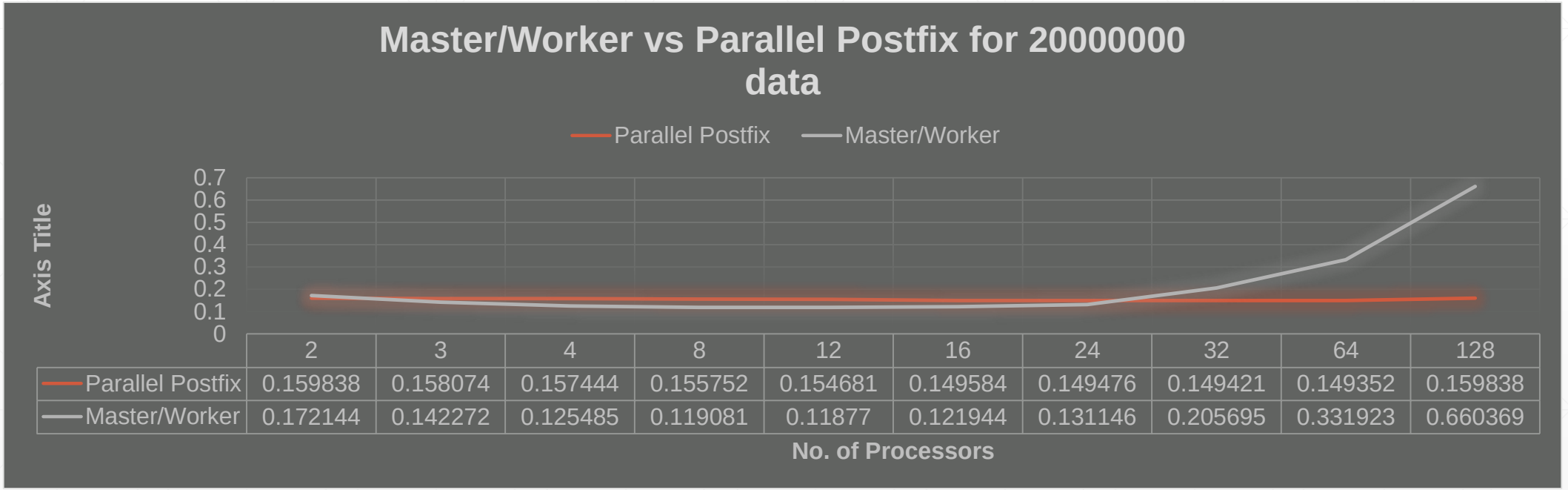
Parallel Postfix Results for 30000000 data items



Parallel Postfix Results for 50000000 data items



Comparison of the two Implementation



	2	3	4	8	12	16	24	32	64	128
Parallel Postfix	0.159838	0.158074	0.157444	0.155752	0.154681	0.149584	0.149476	0.149421	0.149352	0.159838
Master/Worker	0.172144	0.142272	0.125485	0.119081	0.11877	0.121944	0.131146	0.205695	0.331923	0.660369

Parallel Postfix Result Analysis

- From the previous graph it can be seen that the time taken when the number of processor is less in Parallel Postfix implementation tend to be more than the time taken when the problem is implemented in Master Worker Architecture.
- But when the number of processors are increased the running time decreases up to certain value and then it increases. But the increase in the running time is not so high as compared to the Master Worker Architecture.
- This is because the communication among processors involves only a single prefix value of the processor to be sent to the next processor as compared to the Master Worker implementation which needs all the prefix result to be sent to the Master node.

Conclusion

- Master Worker Implementation is good when the number of processor and data is moderate.
- Parallel postfix implementation is better when the number of processor and the data is high.
- The running time decreases up to certain number of processors and then it increases.

Future Work

- Implementation using OpenMPI.
- Implementation using CUDA.
- Analysis with wider range of data.

References

- Algorithms Sequential and Parallel: A Unified Approach, Third Edition Russ Miller and Laurence Boxer
- http://www.cse.buffalo.edu/faculty/miller/Courses/CSE633/Eric-Nagler-Fall-2011.pdf

Questions?