CSCE 735 Spring 2023

Name: Shweta Sharma
UIN: 433003780

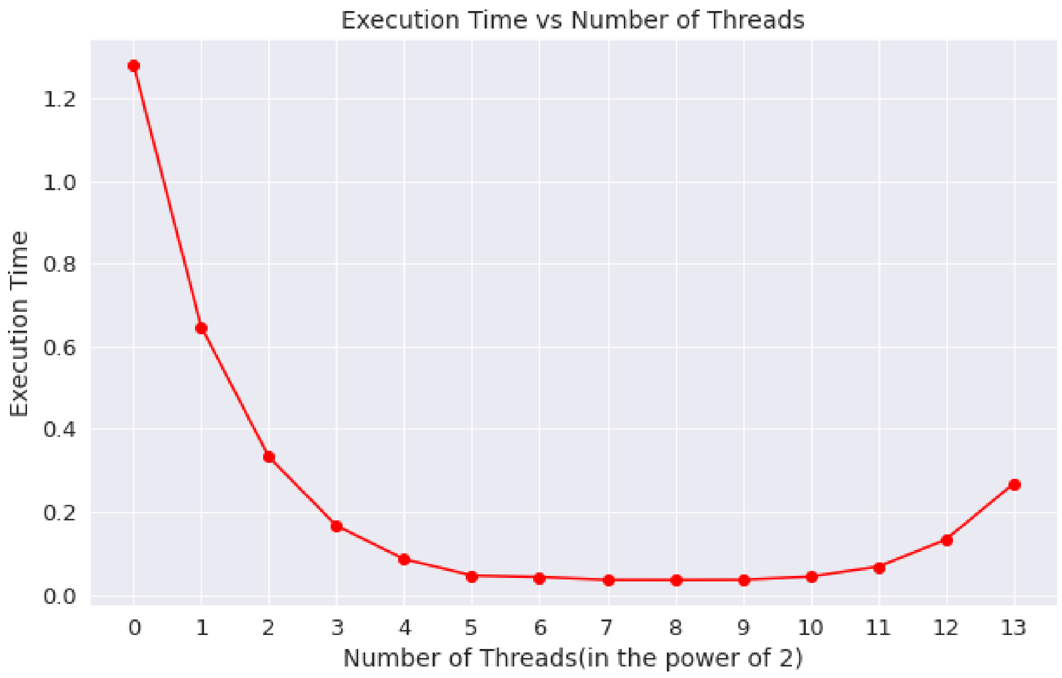
HW 1: Parallel Programming on a Multicore Multiprocessor

Part 1. Shared-Memory Programming with Threads

1. Execute the code for $n=10^8$ with p chosen to be 2k, for k = 0, 1, …, 13. Using the experimental data obtained from these experiments, answer the following questions. For plots, use a logarithmic scale for the x-axis.

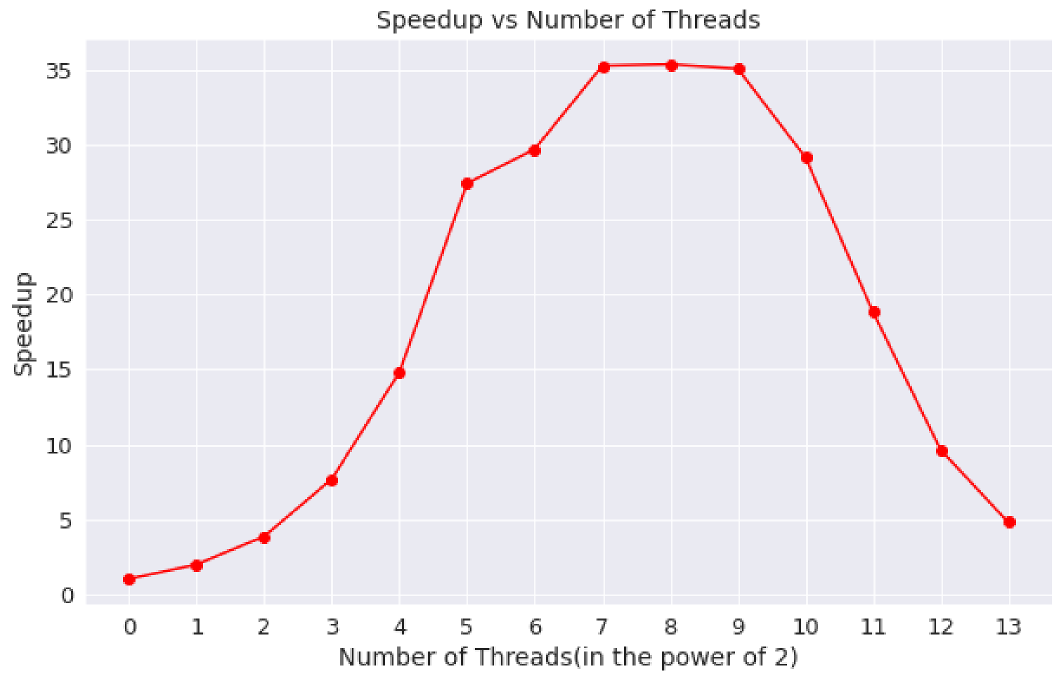
1.1. (10 points) Plot execution time versus p to demonstrate how time varies with the number of threads.

Ans:



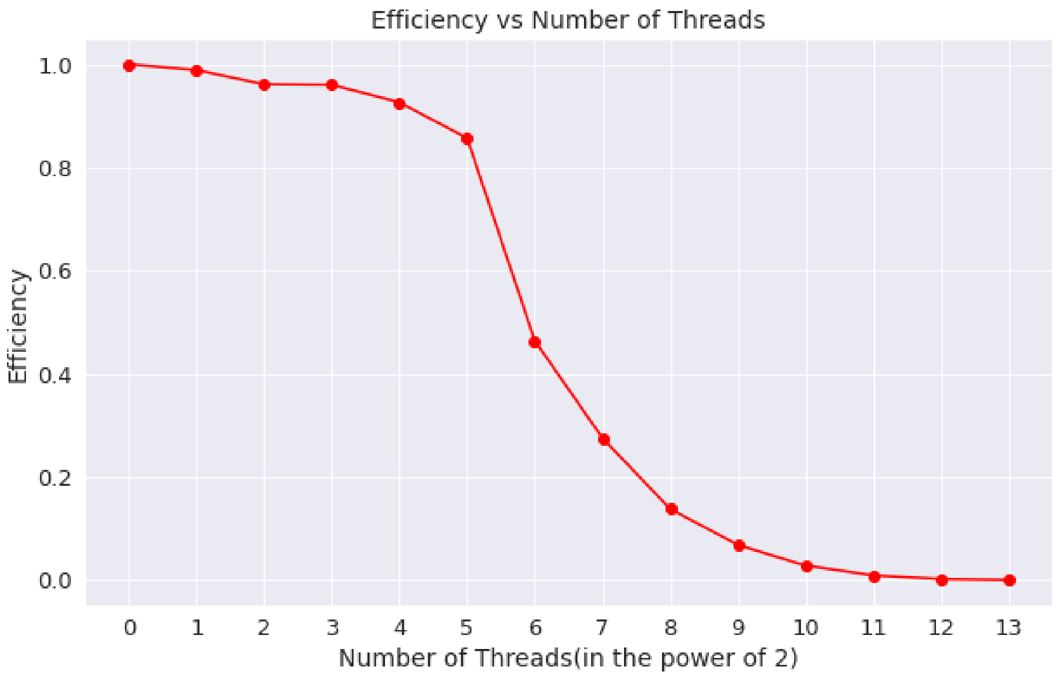
1.2. (10 points) Plot speedup versus p to demonstrate the change in speedup with p.

Ans:



1.3. (5 points) Using the definition: efficiency = speedup/p, plot efficiency versus p to demonstrate how efficiency changes as the number of threads are increased

Ans:



1.4. (5 points) In your experiments, what value of p minimizes the parallel runtime?

Ans: In my experiments, p=258(2^8), minimizes the parallel runtime

2. Repeat the experiments with n=10^{10} to obtain the execution time for p=2k, for k = 0, 1, …, 13.

2.1. (5 points) In this case, what value of p minimizes the parallel runtime?

Ans: In this case, p=1024 (2^{10}), minimizes the parallel runtime

2.2. (5 points) Do you expect the runtime to increase as p is increased beyond a certain value? If so, why? And is this observed in your experiments.

Ans: Yes, this phenomenon is called parallel slowdown. It is when parallelization of a parallel algorithm beyond a certain point causes the program to run slower. It is the result of a communications bottleneck. As more threads are added, each thread spends progressively more time doing communication than useful processing. At some point, the communications overhead surpasses the increased processing power that multithreading, and parallel slowdown occurs.

Yes, we can observe this in our experiment. For our experiment, we see that the maximum speedup is achieved when p=258(2^8). Once we increase the value of p after this point, the speedup decreases significantly.

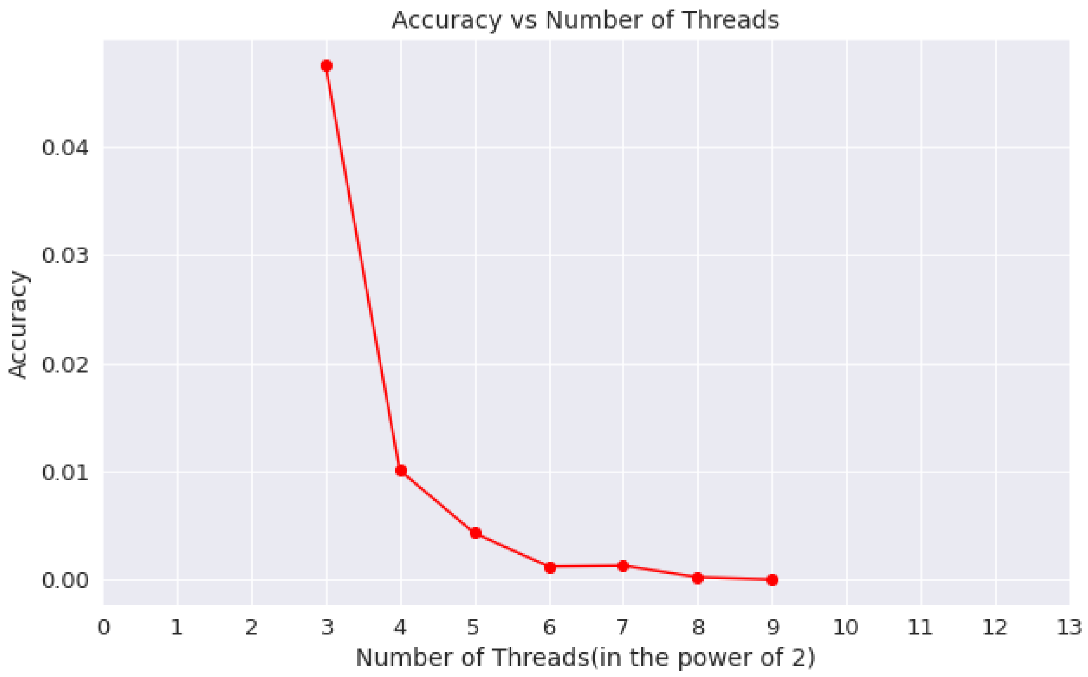
3. (5 points) Do you expect that there would be a difference in the number of threads needed to obtain the minimum execution time for two values of n? Is this observed in your experiments.

Ans: Yes, that there would be a difference in the number of threads needed to obtain the minimum execution time for two values of n. This is because for larger size of data, we can employ more threads to result in more parallelism. For example, given two datasets having 10 and 100 data points, certainly, the number of threads required to achieve maximum parallelism in the dataset having 100 points will be approx 10 times of the number of threads for the dataset having 10 points. Thus, as our size of data increases, the number of threads needed to obtain the minimum execution time also increases.

This is also observed in our experiments.When n=10^8, we see the number of threads needed to minimize runtime was 2^8, however, we see that when n=10^{10}, the number of threads needed to minimize runtime is 2^{10}.

4. (5 points) Plot error versus n to illustrate accuracy of the algorithm as a function of n. You may have to run experiments with different values of n; for example n could be chosen to be 10^k, for k = 3, …, 9. Use p = 48.

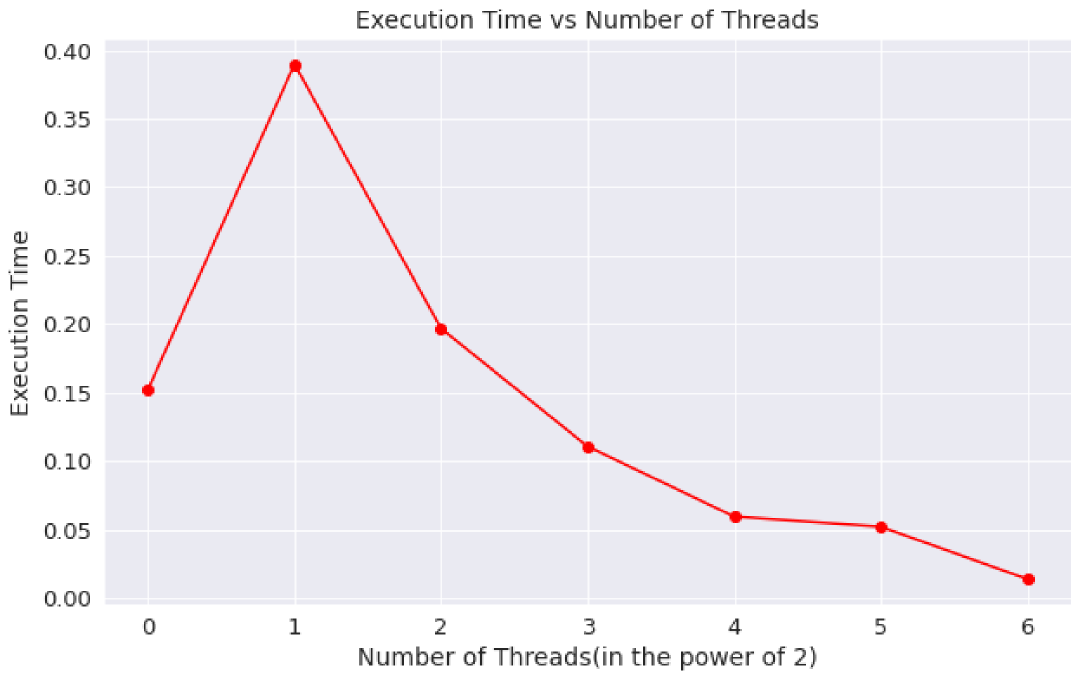
Ans:



Part 2. Distributed-Memory Programming with MPI

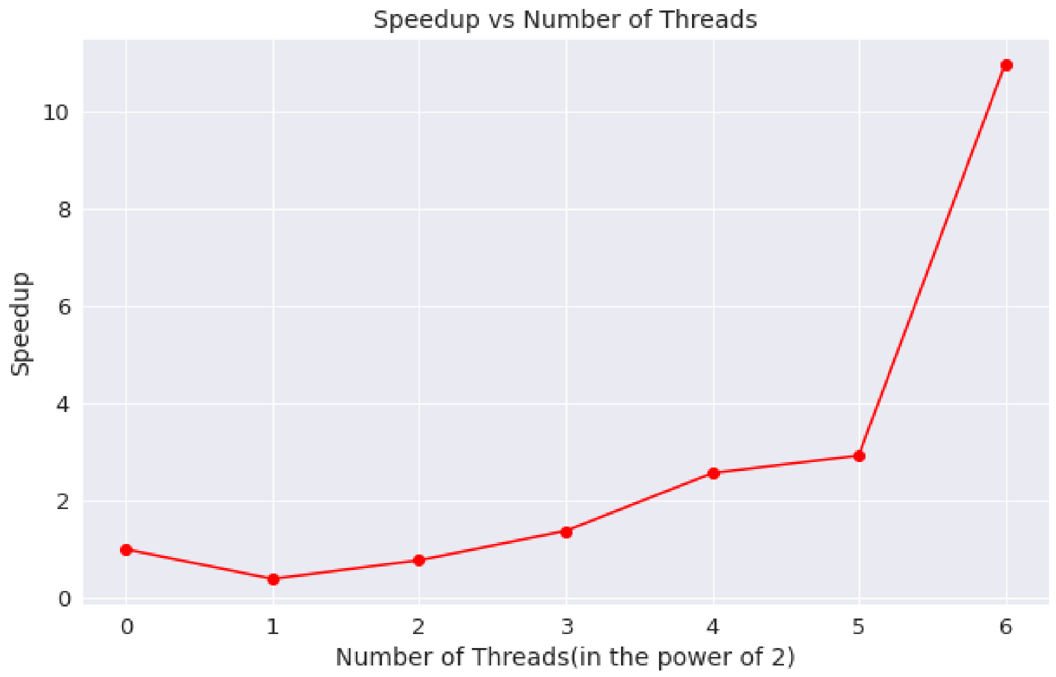
5. Execute the code for n=10^8 with p chosen to be 2k, for k = 0, 1, …, 6. Specify ntasks-per-node=4 in the job file. Using the experimental data obtained from these experiments, answer the following questions. For plots, use a logarithmic scale for the x-axis.
5.1. (10 points) Plot execution time versus p to demonstrate how time varies with the number of processes.

Ans:



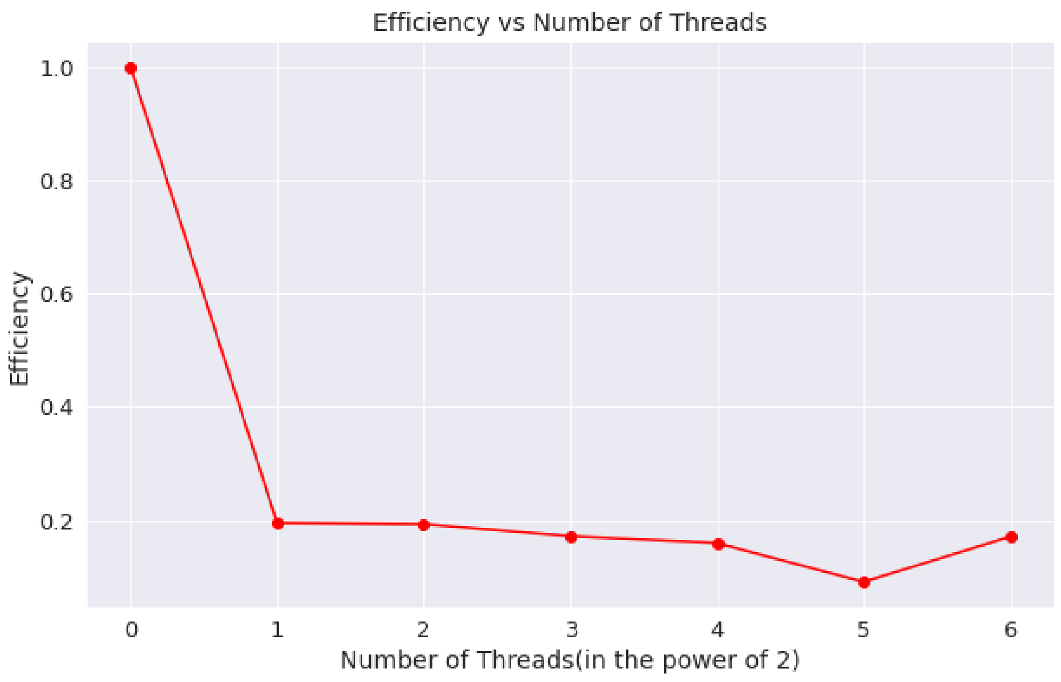
5.2. (10 points) Plot speedup versus p to demonstrate the change in speedup with p.

Ans:



5.3. (5 points) Using the definition: efficiency = speedup/p, plot efficiency versus p to demonstrate how efficiency changes as the number of processes is increased.

Ans:

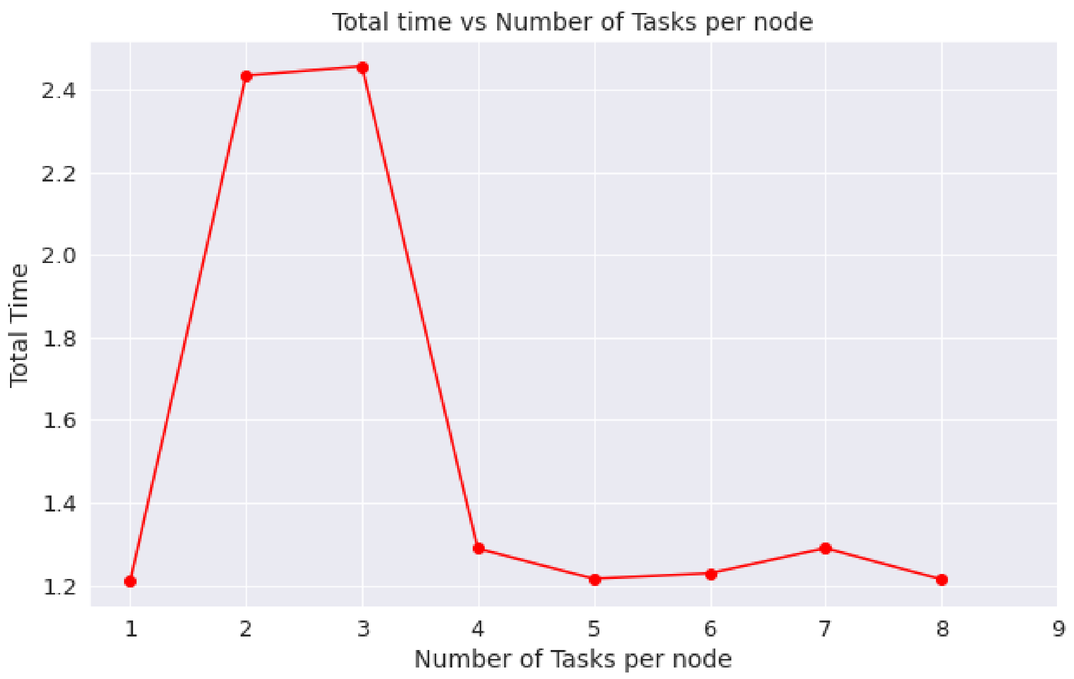


5.4. (5 points) What value of p minimizes the parallel runtime?

Ans: p=64(2^6) minimizes the parallel runtime

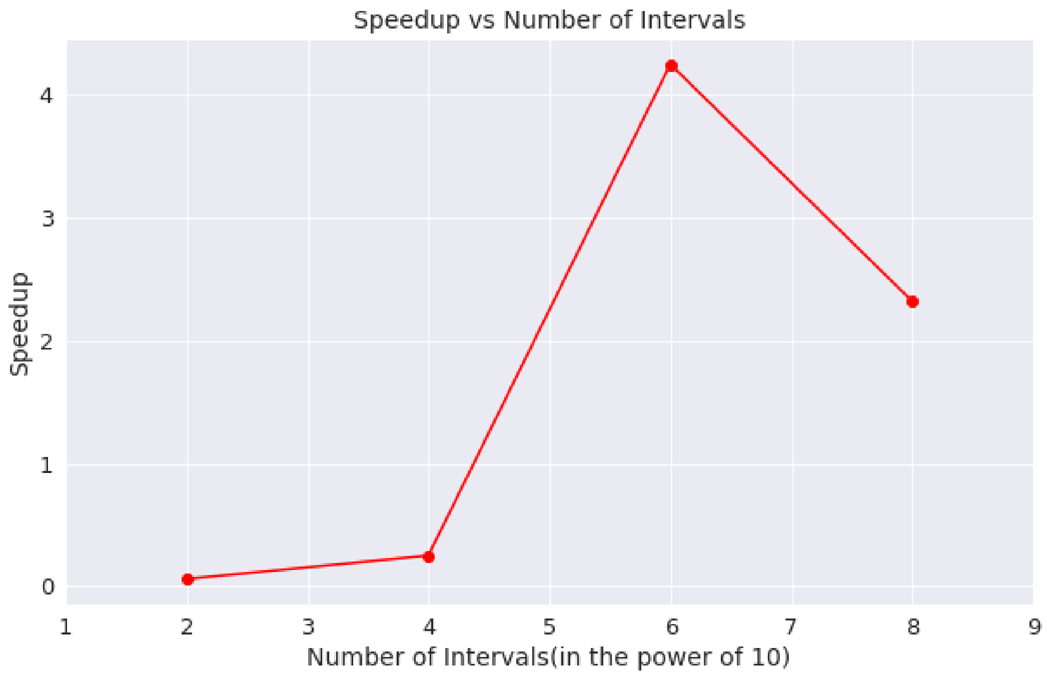
6. (10 points) With n=10^{10} and p=64, determine the value of ntasks-per-node that minimizes the total_time. Plot time versus ntasks-per-node to illustrate your experimental results for this question.

Ans:



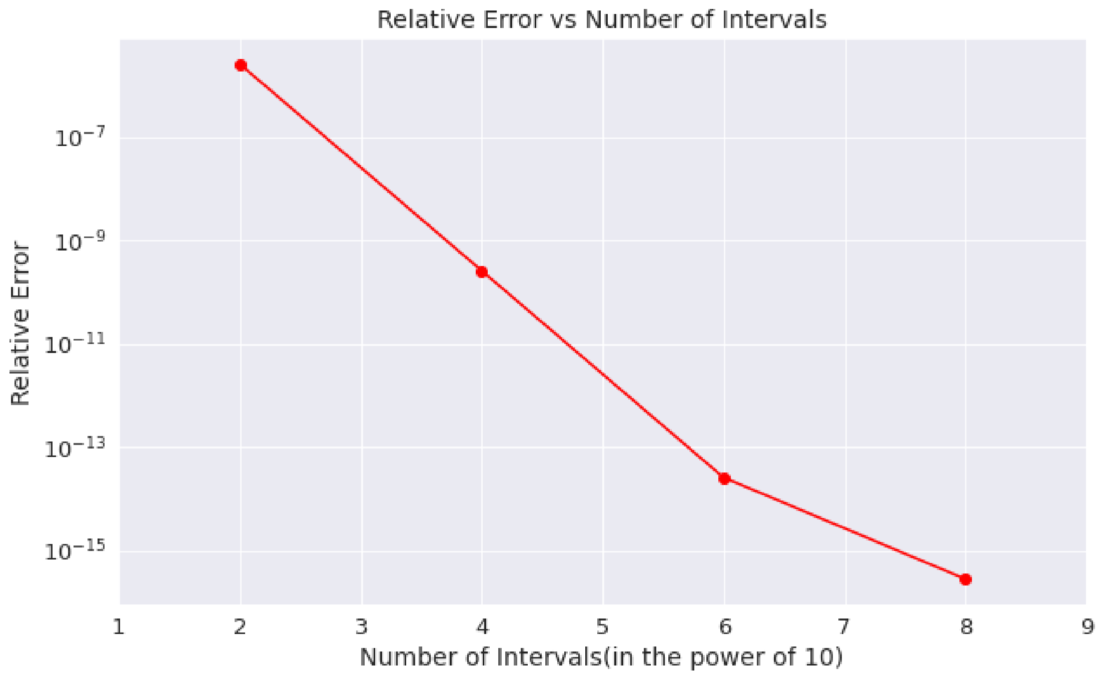
7. Execute the code with p=64 for n=10^2, 10^4, 10^6 and 10^8, with ntasks-per-node=4.
7.1. (5 points) Plot the speedup observed as a function of n on p=64 w.r.t. p=1. You will need to obtain execution time on p=1 for n=10^2, 10^4, 10^6 and 10^8.

Ans:



7.2. (5 points) Plot the relative error versus n to illustrate the accuracy of the algorithm as a function of n.

Ans:


Relative Error vs Number of Intervals
Relative Error
10^{-7}
10^{-9}
10^{-11}
10^{-13}
10^{-15}
Number of Intervals(in the power of 10)