**CSCE 735 Fall 2022**

**Name: Rohan Chaudhury**

**UIN: 432001358**

**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 2^k, 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.**

**OUTPUT:**

Data obtained from the experiments:

A picture containing table

Description automatically generated

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

**OUTPUT:** The required plot:

Chart, line chart

Description automatically generated

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

**OUTPUT:** The required data and plot:

Table

Description automatically generated

Chart, line chart

Description automatically generated

**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.**

**OUTPUT:** The required data and plot:

Chart

Description automatically generated with low confidence

Chart, line chart

Description automatically generated

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

**ANSWER:**

p = 256 gives the least execution time that is = 0.0346 seconds

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

**OUTPUT:** Data obtained from the experiments:

Table

Description automatically generated with medium confidence

A picture containing chart

Description automatically generated

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

**ANSWER:**

p = 512 gives the least execution time that is = 2.6935 seconds

**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.**

**ANSWER:**

Yes, I expect that the runtime would increase after a certain number of threads (that is, as p is increased beyond a certain value).

We can also observe this is in the first experiment (where n=10^8), where the least time was taken when p=256, and then the time taken gradually increased till p=8192.

This can also be observed in the second experiment (where n=10^10), where the least time was taken when p=512, and then the time taken gradually increased till p=8192.

This behavior can be explained by the following factors:

1. The actual quantity of resources, or processing units, available and the fact that they are in use or not, can affect the final parallelized runtimes. Resource availability has a bigger influence on total runtime than the number of started threads. If more threads are launched than the number of processing units available then the runtimes may deteriorate owing to scheduling delays and overheads.
2. If there are more number of threads than the number of processing units available then **context switching** is employed. In such a case multiple threads would be assigned to the same processing units and then the processing units would need to devise a technique for scheduling the execution of several threads on themselves and switching between the assigned threads on a timely basis. If the time necessary for context switching exceeds the time necessary to run that thread, then the runtime can be expected to show a degradation with an increase in thread count.
3. With an increase in the number of threads, some of the threads will have to wait for other threads to complete their I/O operations as the resources are shared and access to shared resources must be synchronized.

**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.**

**ANSWER:**

Yes, there would be a difference in the number of threads needed to obtain the minimum execution time for two values of n.

This is also observed in my experiments:

p = 256 gives the least execution time (that is = 0.0346 seconds) for n= 10^8 whereas, p = 512 gives the least execution time (that is = 2.6935 seconds) for n= 10^10.

The proportion of work handled by each processing unit is significant here. If we take an example where the processor can accomplish the task of one thread in the time allotted to it by the CPU and where n = 512 and p = 128, we may assume that each thread will handle 4 units of work. But, when n is increased, for example, to n = 2048, each thread is compelled to undertake 16 units of work. Because each thread in this situation does not complete execution within the allotted CPU time, the processor may have to perform more context switches. In such a case, increasing the number of threads would lead to faster runtimes since each thread now interacts with fewer work units that can be run in the assigned CPU time and the number of context switches would reduce.

According to the speed-up formula,

Where,

p = number of processes

n = number of trials

When n is increased, speed up tends to peak at a later value of p.

Furthermore, running the same tests at different times generated differing findings, indicating that the number of threads necessary to attain the shortest runtimes changed for each run's input parameters. This implies that the number of threads necessary to achieve the shortest possible runtime may be potentially impacted by external factors like the task scheduling approaches, present workload of the processing units, and so on.

**4. (5 points) Plot error versus n to illustrate the 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.**

**ANSWER:**

Graphical user interface

Description automatically generated with low confidence

Chart, line chart

Description automatically generated

**Part 2. Distributed-Memory Programming with MPI**

**5. Execute the code for n=108 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.**

**OUTPUT:** Data obtained from the experiments:

A picture containing chart

Description automatically generated

**5.1. (10 points) Plot execution time versus p to demonstrate how time varies with the number of processes.**

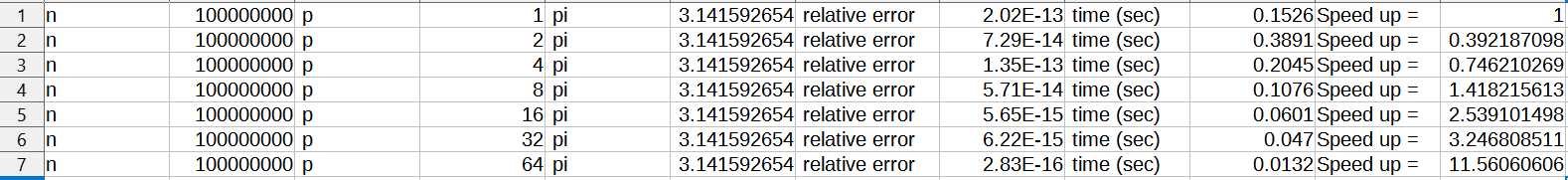
**OUTPUT:** The required plot:

Chart, line chart

Description automatically generated

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

**OUTPUT:** Data obtained from the experiments and the plot:

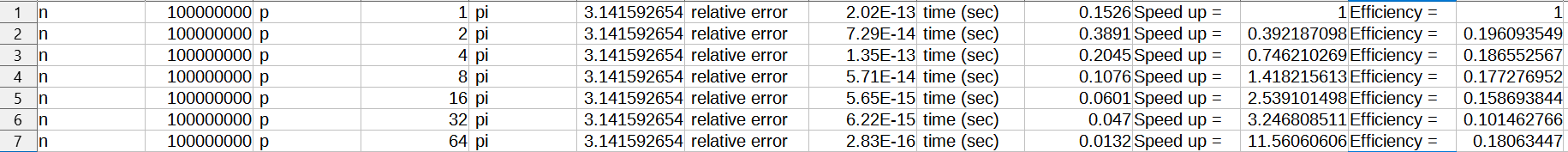


Chart, line chart

Description automatically generated

**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.**

**OUTPUT:** Data obtained from the experiments and the plot:



Chart, line chart

Description automatically generated

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

**ANSWER:**

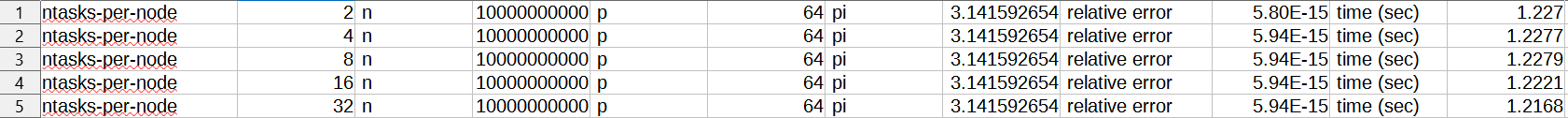
p = 64 gives the lowest runtime of 0.0132 seconds

**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.**

**ANSWER:**

For ntasks-per-node=32, lowest runtime was seen, that is = 1.2168 seconds

Data obtained from the experiments and the plot:



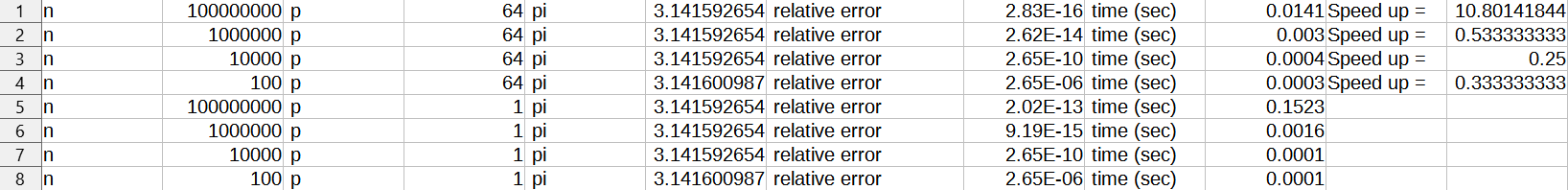
Chart, line chart

Description automatically generated

**7. Execute the code with p=64 for n=102, 104, 106 and 108, 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=102, 104, 106 and 108.**

**OUTPUT:** Data obtained from the experiments and the plot:

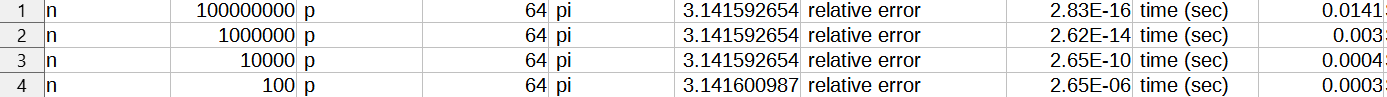


Chart, line chart

Description automatically generated

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

**OUTPUT:** Data obtained from the experiments and the plot:



Chart, line chart

Description automatically generated