CSCE 735 Homework 1

Name: Pranav Anantharam
UIN: 734003886

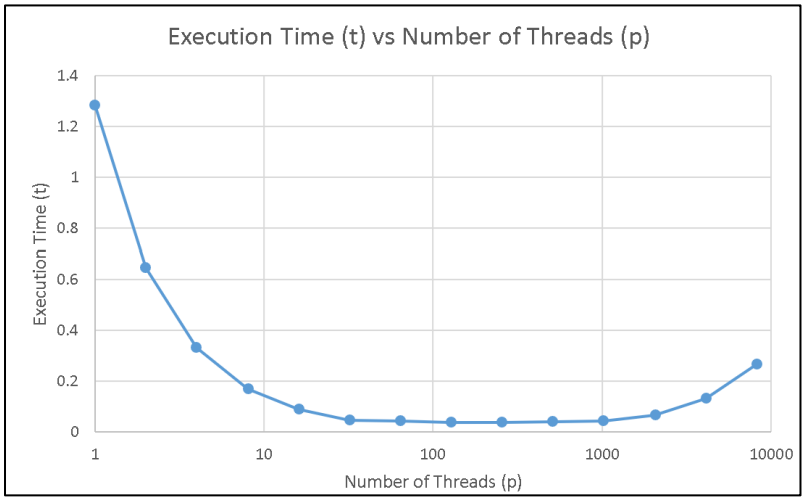
1. Console output: Running compute_pi.exe in dedicated mode

```
Trials = 100000000, Threads =    1, pi =   3.1416149600, error = 7.10e-06, time (sec) =   1.2843
Trials = 100000000, Threads =    2, pi =   3.1417022800, error = 3.49e-05, time (sec) =   0.6451
Trials = 100000000, Threads =    4, pi =   3.1415910800, error = 5.01e-07, time (sec) =   0.3300
Trials = 100000000, Threads =    8, pi =   3.1415947600, error = 6.70e-07, time (sec) =   0.1664
Trials = 100000000, Threads =   16, pi =   3.1415291200, error = 2.02e-05, time (sec) =   0.0863
Trials = 100000000, Threads =   32, pi =   3.1415901200, error = 8.06e-07, time (sec) =   0.0458
Trials = 100000000, Threads =   64, pi =   3.1413512400, error = 7.68e-05, time (sec) =   0.0425
Trials = 100000000, Threads =  128, pi =   3.1429299200, error = 4.26e-04, time (sec) =   0.0358
Trials = 100000000, Threads =  256, pi =   3.1412995200, error = 9.33e-05, time (sec) =   0.0355
Trials = 100000000, Threads =  512, pi =   3.1468450800, error = 1.67e-03, time (sec) =   0.0379
Trials = 100000000, Threads = 1024, pi =   3.1514026000, error = 3.12e-03, time (sec) =   0.0421
Trials = 100000000, Threads = 2048, pi =   3.1453611600, error = 1.20e-03, time (sec) =   0.0654
Trials = 100000000, Threads = 4096, pi =   3.1494162400, error = 2.49e-03, time (sec) =   0.1300
Trials = 100000000, Threads = 8192, pi =   3.1377031200, error = 1.24e-03, time (sec) =   0.2642
Trials = 10000000000, Threads =    1, pi =   1.4236202260, error = 5.47e-01, time (sec) = 127.8758
Trials = 10000000000, Threads =    2, pi =   3.1416070092, error = 4.57e-06, time (sec) =  64.0741
Trials = 10000000000, Threads =    4, pi =   3.1416060276, error = 4.26e-06, time (sec) =  32.0206
Trials = 10000000000, Threads =    8, pi =   3.1416118228, error = 6.10e-06, time (sec) =  16.0239
Trials = 10000000000, Threads =   16, pi =   3.1416066896, error = 4.47e-06, time (sec) =   8.0183
Trials = 10000000000, Threads =   32, pi =   3.1416332068, error = 1.29e-05, time (sec) =   4.0187
Trials = 10000000000, Threads =   64, pi =   3.1416139092, error = 6.77e-06, time (sec) =   3.1337
Trials = 10000000000, Threads =  128, pi =   3.1415943424, error = 5.38e-07, time (sec) =   2.7581
Trials = 10000000000, Threads =  256, pi =   3.1416473120, error = 1.74e-05, time (sec) =   2.7156
Trials = 10000000000, Threads =  512, pi =   3.1415762012, error = 5.24e-06, time (sec) =   2.6977
Trials = 10000000000, Threads = 1024, pi =   3.1414319268, error = 5.12e-05, time (sec) =   2.6909
Trials = 10000000000, Threads = 2048, pi =   3.1412588724, error = 1.06e-04, time (sec) =   2.6989
Trials = 10000000000, Threads = 4096, pi =   3.1407169720, error = 2.79e-04, time (sec) =   2.7186
Trials = 10000000000, Threads = 8192, pi =   3.1412633928, error = 1.05e-04, time (sec) =   2.7667
Trials = 1000, Threads =   48, pi =   2.9920000000, error = 4.76e-02, time (sec) =   0.0018
Trials = 10000, Threads =   48, pi =   3.1732000000, error = 1.01e-02, time (sec) =   0.0017
Trials = 100000, Threads =   48, pi =   3.1550800000, error = 4.29e-03, time (sec) =   0.0023
Trials = 1000000, Threads =   48, pi =   3.1453960000, error = 1.21e-03, time (sec) =   0.0026
Trials = 10000000, Threads =   48, pi =   3.1456372000, error = 1.29e-03, time (sec) =   0.0107
Trials = 100000000, Threads =   48, pi =   3.1409400400, error = 2.08e-04, time (sec) =   0.0390
Trials = 1000000000, Threads =   48, pi =   3.1416179080, error = 8.04e-06, time (sec) =   0.2947
```

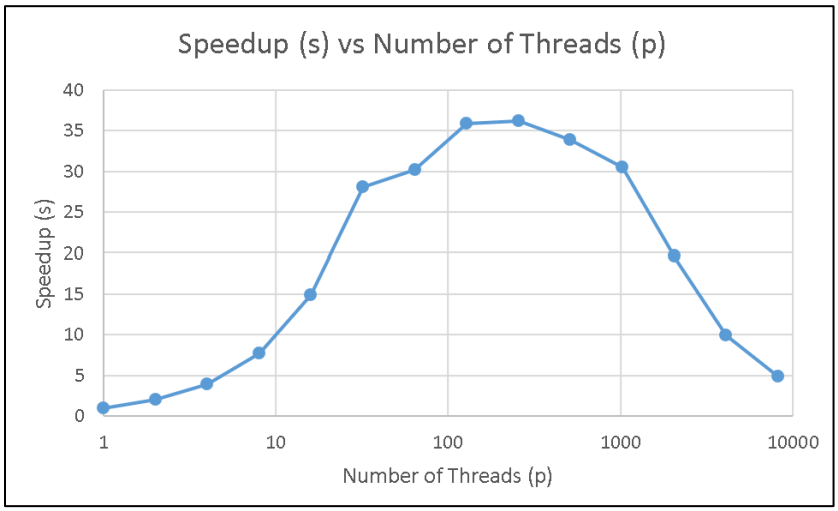
Spreadsheet Calculations:

Number of Threads (p)	Efficiency (e)	Speedup (s)	Execution Time (t)
1	1	1	1.2843
2	0.995427066	1.990854131	0.6451
4	0.972954545	3.891818182	0.33
8	0.96476863	7.718149038	0.1664
16	0.930112978	14.88180765	0.0863
32	0.876296397	28.04148472	0.0458
64	0.472169118	30.21882353	0.0425
128	0.280267982	35.87430168	0.0358
256	0.141318222	36.17746479	0.0355
512	0.066184655	33.88654354	0.0379
1024	0.029790955	30.50593824	0.0421
2048	0.009588679	19.63761468	0.0654
4096	0.002411922	9.879230769	0.13
8192	0.000593395	4.861090083	0.2642

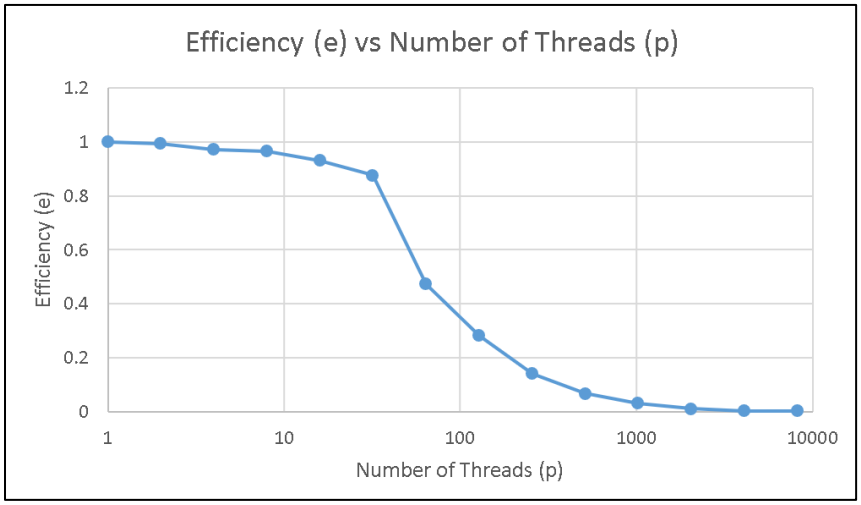
1.1 Execution time versus p to demonstrate how time varies with the number of threads



1.2 Speedup versus p to demonstrate the change in speedup with p



1.3 Efficiency versus p to demonstrate how efficiency changes as the number of threads are increased



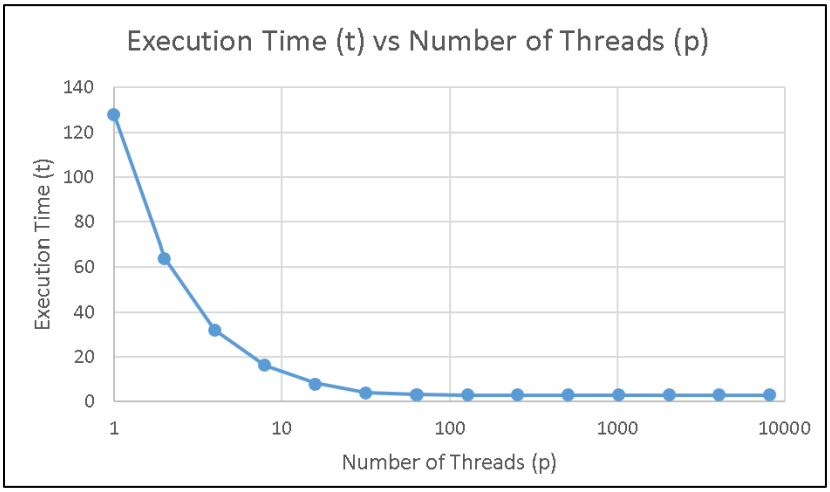
1.4 In your experiments, what value of p minimizes the parallel runtime?

The execution time is at its minimum value **(0.0355 seconds)** when the number of threads (p) = **256**

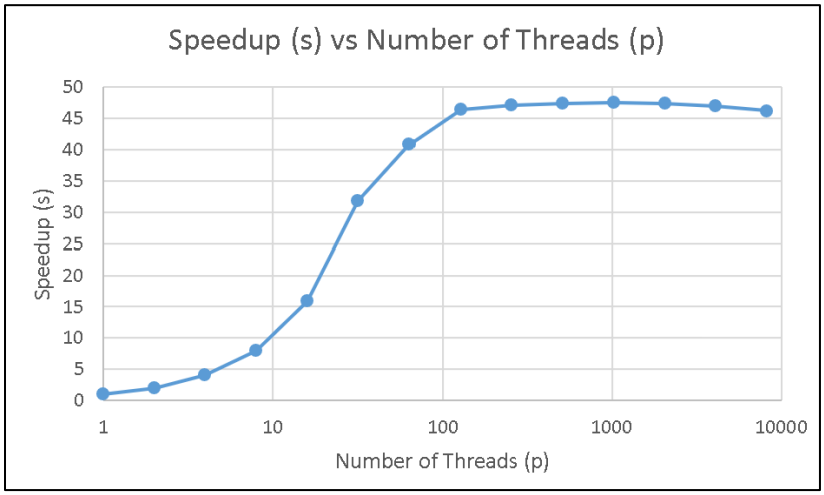
2. Spreadsheet Calculations:

Number of Threads (p)	Execution Time (t)	Speedup (s)	Efficiency (e)
1	127.8758	1	1
2	64.0741	1.995748672	0.997874336
4	32.0206	3.993547904	0.998386976
8	16.0239	7.980316902	0.997539613
16	8.0183	15.94799396	0.996749623
32	4.0187	31.82019061	0.994380957
64	3.1337	40.80665029	0.637603911
128	2.7581	46.36372865	0.36221663
256	2.7156	47.08933569	0.183942718
512	2.6977	47.40178671	0.092581615
1024	2.6909	47.52157271	0.046407786
2048	2.6989	47.38071066	0.023135113
4096	2.7186	47.03737218	0.011483733
8192	2.7667	46.21961181	0.005642042

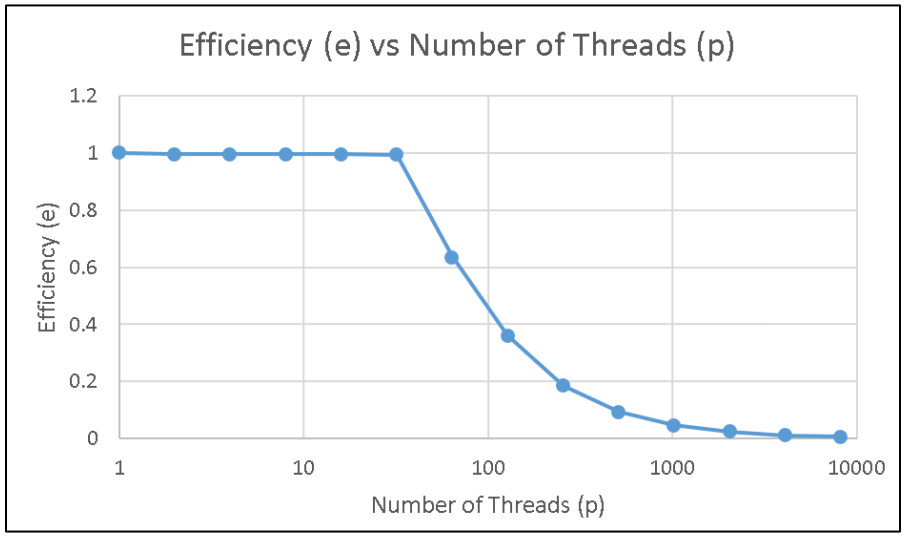
Execution time versus p to demonstrate how time varies with the number of threads



Speedup versus p to demonstrate the change in speedup with p



Efficiency versus p to demonstrate how efficiency changes as the number of threads are increased



2.1 In this case, what value of p minimizes the parallel runtime?

The execution time is at its minimum value **(2.6909 seconds)** when the number of threads (p) = **1024**

2.2 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.

Yes, the execution runtime increases slightly as p is increased beyond a certain value (as observed in experiments). This is because too many threads have been initialized. Each thread consumes system resources, and there is overhead associated with creating, managing and switching between threads which results in an increase in execution runtime. We also observe that beyond a certain point, increasing the number of threads does not lead to proportional speedup (Amdahl's Law).

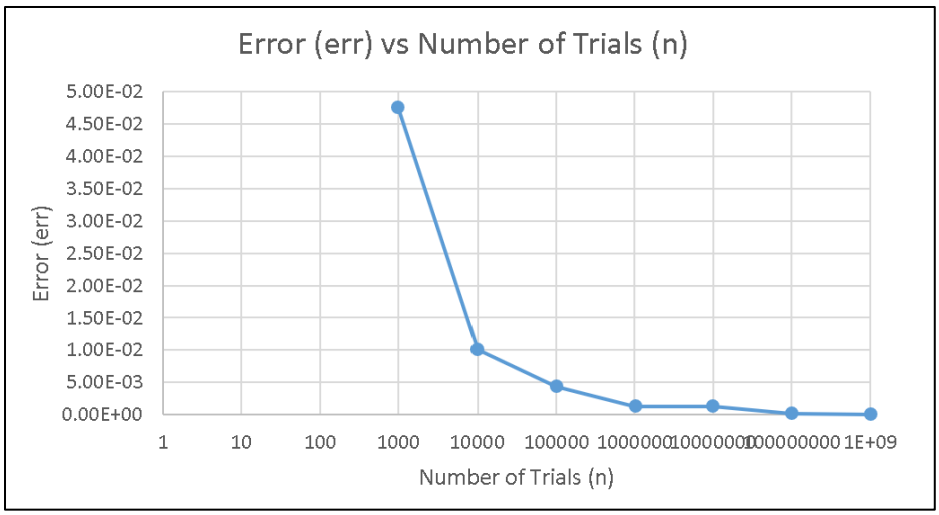
3. Do you expect that there would be a difference in the number of threads needed to obtain the minimum execution time for two values of n? Is this observed in your experiments.

Yes, larger number of threads were required to obtain the minimum execution when the number of trials were increased from $n = 10^8$ to $n = 10^{10}$ (as observed in experiments). This was expected as the problem size was increased from $n = 10^8$ to $n = 10^{10}$, as more threads ran concurrently and performed computations, thereby improving performance and resource utilization.

4. Plot error versus n to illustrate accuracy of the algorithm as a function of n. (p = 48)

Spreadsheet Calculations:

Number of Trials (n)	Error (err)
1000	4.76E-02
10000	1.01E-02
100000	4.29E-03
1000000	1.21E-03
10000000	1.29E-03
100000000	2.08E-04
1000000000	8.04E-06



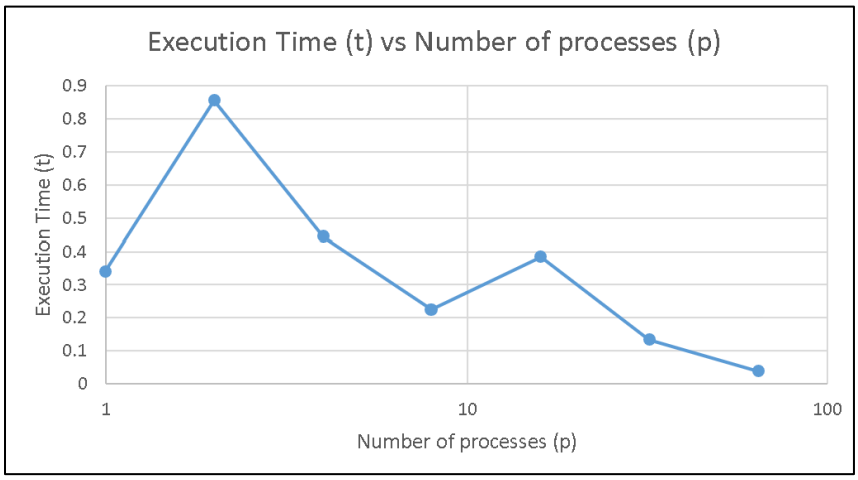
5. Console Output: Running compute_pi_mpi.exe in dedicated mode

```
<----- Q 5 ----->
Processes = 1
n = 100000000, p = 1, pi = 3.1415926535904264, relative error = 2.02e-13, time (sec) =   0.3427
Processes = 2
n = 100000000, p = 2, pi = 3.1415926535900223, relative error = 7.29e-14, time (sec) =   0.8555
Processes = 4
n = 100000000, p = 4, pi = 3.1415926535902168, relative error = 1.35e-13, time (sec) =   0.4450
Processes = 8
n = 100000000, p = 8, pi = 3.1415926535896137, relative error = 5.71e-14, time (sec) =   0.2249
Processes = 16
n = 100000000, p = 16, pi = 3.1415926535897754, relative error = 5.65e-15, time (sec) =   0.3835
Processes = 32
n = 100000000, p = 32, pi = 3.1415926535897736, relative error = 6.22e-15, time (sec) =   0.1339
Processes = 64
n = 100000000, p = 64, pi = 3.1415926535897940, relative error = 2.83e-16, time (sec) =   0.0390
<----- Q 7a ----->
Processes = 64
n = 100000000, p = 64, pi = 3.1415926535897940, relative error = 2.83e-16, time (sec) =   0.0294
Processes = 64
n = 1000000, p = 64, pi = 3.1415926535898753, relative error = 2.62e-14, time (sec) =   0.0010
Processes = 64
n = 10000, p = 64, pi = 3.1415926544231265, relative error = 2.65e-10, time (sec) =   0.0015
Processes = 64
n = 100, p = 64, pi = 3.1416009869231249, relative error = 2.65e-06, time (sec) =   0.0012
<----- Q 7b ----->
Processes = 1
n = 100000000, p = 1, pi = 3.1415926535904264, relative error = 2.02e-13, time (sec) =   0.3419
Processes = 1
n = 1000000, p = 1, pi = 3.1415926535897643, relative error = 9.19e-15, time (sec) =   0.0038
Processes = 1
n = 10000, p = 1, pi = 3.1415926544231341, relative error = 2.65e-10, time (sec) =   0.0001
Processes = 1
n = 100, p = 1, pi = 3.1416009869231254, relative error = 2.65e-06, time (sec) =   0.0000
```

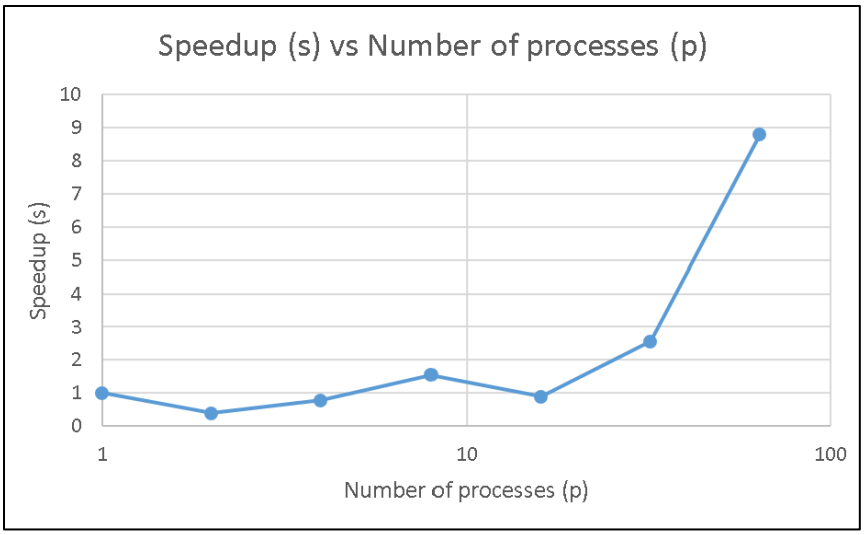
Spreadsheet Calculations:

Number of processes (p)	Execution Time (t)	Speedup (s)	Efficiency (e)
1	0.3427	1	1
2	0.8555	0.400584454	0.200292227
4	0.445	0.77011236	0.19252809
8	0.2249	1.52378835	0.190473544
16	0.3835	0.893611473	0.055850717
32	0.1339	2.559372666	0.079980396
64	0.039	8.787179487	0.137299679

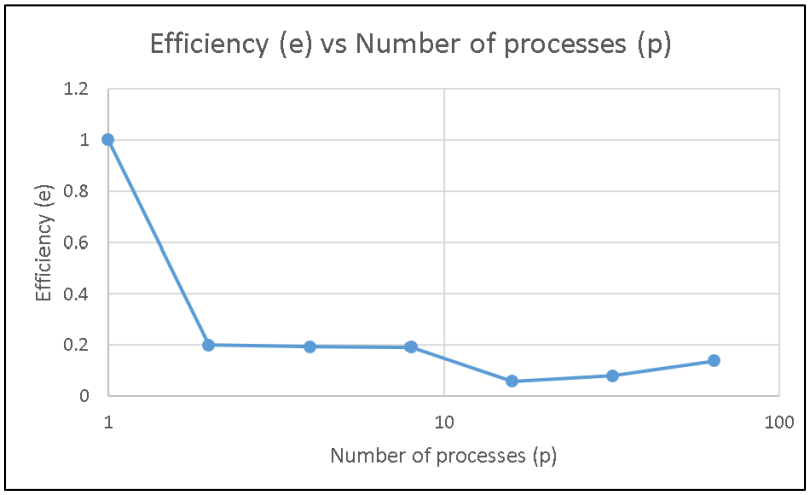
5.1 Execution time versus p to demonstrate how time varies with the number of processes



5.2 Speedup versus p to demonstrate the change in speedup with p



5.3 Efficiency versus p to demonstrate how efficiency changes as the number of processes is increased



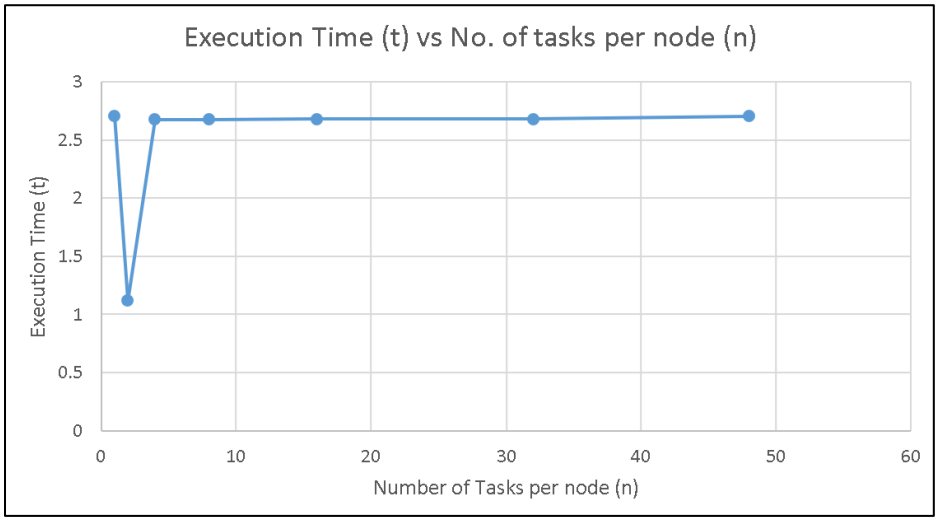
5.4 What value of p minimizes the parallel runtime?

The execution time is at its minimum value **(0.039 seconds)** when the number of processes (p) = **64**

6. 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.

Spreadsheet Calculations:

Number of Tasks per node (n)	Execution Time (t)
1	2.7075
2	1.1177
4	2.6781
8	2.6798
16	2.6821
32	2.6862
48	2.7079

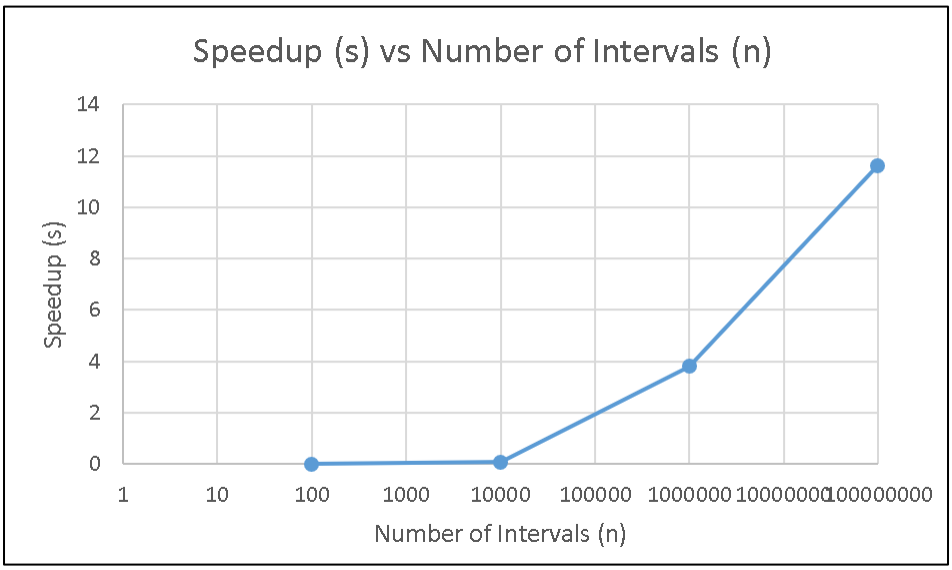


The total time is at its minimum value **(1.1177 seconds)** when the number of tasks per node (n) = **2**

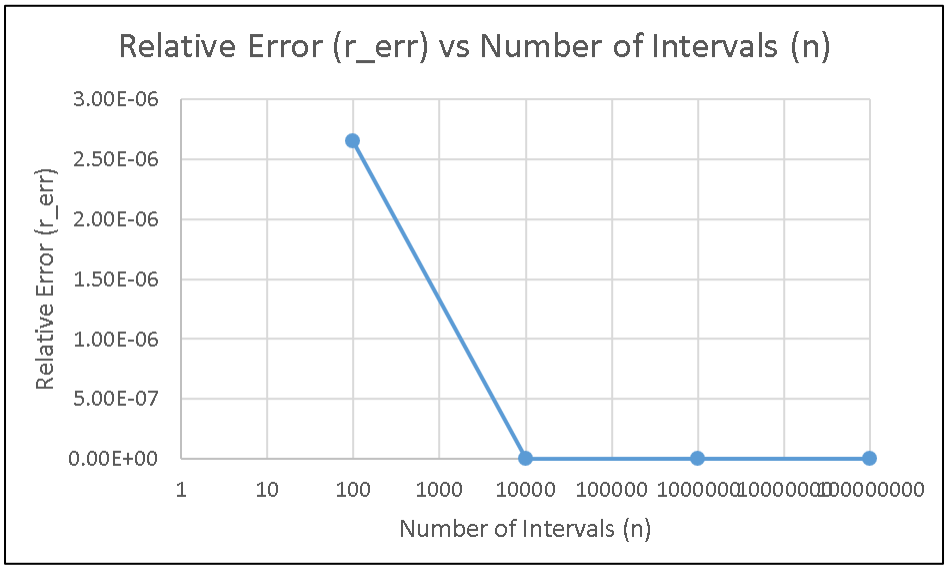
7. Spreadsheet Calculations:

Number of intervals (n)	Execution Time (t) p = 1	Execution Time (t) p = 64	Speedup (s)	Relative Error (r_err) p = 64
100	0	0.0012	0	2.65E-06
10000	0.0001	0.0015	0.066666667	2.65E-10
1000000	0.0038	0.001	3.8	2.62E-14
100000000	0.3419	0.0294	11.6292517	2.83E-16

7.1 Plot the speedup observed as a function of n on p=64 w.r.t. p=1.



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



Code changes for Q4:

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE            #Do not propagate environment
#SBATCH --get-user-env=L         #Replicate login environment
#
##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobName       #Set the job name to "JobName"
#SBATCH --time=0:30:00           #Set the wall clock limit to 0hr and 30min
#SBATCH --nodes=16               #Request 16 node
#SBATCH --ntasks-per-node=1      #Request 4 tasks/cores per node
#SBATCH --mem=8G                 #Request 8GB per node
```

```
#SBATCH --output=output.%j       #Send stdout/err to "output.[jobID]"
#
##OPTIONAL JOB SPECIFICATIONS
##SBATCH --mail-type=ALL              #Send email on all job events
##SBATCH --mail-user=email_address    #Send all emails to email_address
#
##First Executable Line
#
module load intel             # load Intel software stack
#
echo "<----- Q 6 ----->"
echo "Processes = 64"
mpirun -np 64 ./compute_pi_mpi.exe 10000000000
```

Code changes for Q5 and Q7:

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE            #Do not propagate environment
#SBATCH --get-user-env=L         #Replicate login environment
#
##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobName       #Set the job name to "JobName"
#SBATCH --time=0:30:00           #Set the wall clock limit to 0hr and 30min
#SBATCH --nodes=16               #Request 16 node
#SBATCH --ntasks-per-node=4      #Request 4 tasks/cores per node
#SBATCH --mem=8G                 #Request 8GB per node
#SBATCH --output=output.%j       #Send stdout/err to "output.[jobID]"
#
##OPTIONAL JOB SPECIFICATIONS
##SBATCH --mail-type=ALL              #Send email on all job events
##SBATCH --mail-user=email_address    #Send all emails to email_address
#
##First Executable Line
#
module load intel             # load Intel software stack
#
echo "<----- Q 5 ----->"
echo "Processes = 1"
mpirun -np  1 ./compute_pi_mpi.exe 100000000
echo "Processes = 2"
mpirun -np  2 ./compute_pi_mpi.exe 100000000
echo "Processes = 4"
mpirun -np  4 ./compute_pi_mpi.exe 100000000
echo "Processes = 8"
mpirun -np  8 ./compute_pi_mpi.exe 100000000
echo "Processes = 16"
mpirun -np 16 ./compute_pi_mpi.exe 100000000
```

```
echo "Processes = 32"
mpirun -np 32 ./compute_pi_mpi.exe 100000000
echo "Processes = 64"
mpirun -np 64 ./compute_pi_mpi.exe 100000000

echo "<----- Q 7a ----->"
echo "Processes = 64"
mpirun -np 64 ./compute_pi_mpi.exe 100000000
echo "Processes = 64"
mpirun -np 64 ./compute_pi_mpi.exe 1000000
echo "Processes = 64"
mpirun -np 64 ./compute_pi_mpi.exe 10000
echo "Processes = 64"
mpirun -np 64 ./compute_pi_mpi.exe 100

echo "<----- Q 7b ----->"
echo "Processes = 1"
mpirun -np 1 ./compute_pi_mpi.exe 100000000
echo "Processes = 1"
mpirun -np 1 ./compute_pi_mpi.exe 1000000
echo "Processes = 1"
mpirun -np 1 ./compute_pi_mpi.exe 10000
echo "Processes = 1"
mpirun -np 1 ./compute_pi_mpi.exe 100
```
