Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

HW 1: Parallel Programming on a Multicore Multiprocessor

Part 1. Shared-Memory Programming with Threads

SBATCH Script (Questions 1-4)

```
#!/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=HW1_735       #Set the job name to "JobExample2"
#SBATCH --time=1:30:00               #Set the wall clock limit to 6hr and 30min
#SBATCH --nodes=1                    #Request 1 node
#SBATCH --ntasks-per-node=48          #Request 8 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=jsalguero@tamu.edu    #Send all emails to email_address
#
##First Executable Line
#
module load intel               # load Intel software stack
#
./compute_pi.exe 100000000 1
./compute_pi.exe 100000000 2
./compute_pi.exe 100000000 4
./compute_pi.exe 100000000 8
./compute_pi.exe 100000000 16
./compute_pi.exe 100000000 32
./compute_pi.exe 100000000 64
./compute_pi.exe 100000000 128
./compute_pi.exe 100000000 256
./compute_pi.exe 100000000 512
./compute_pi.exe 100000000 1024
./compute_pi.exe 100000000 2048
./compute_pi.exe 100000000 4096
./compute_pi.exe 100000000 8192

./compute_pi.exe 10000000000 1
./compute_pi.exe 10000000000 2
./compute_pi.exe 10000000000 4
./compute_pi.exe 10000000000 8
./compute_pi.exe 10000000000 16
./compute_pi.exe 10000000000 32
./compute_pi.exe 10000000000 64
./compute_pi.exe 10000000000 128
./compute_pi.exe 10000000000 256
./compute_pi.exe 10000000000 512
./compute_pi.exe 10000000000 1024
./compute_pi.exe 10000000000 2048
./compute_pi.exe 10000000000 4096
./compute_pi.exe 10000000000 8192

./compute_pi.exe 1000 48
./compute_pi.exe 10000 48
./compute_pi.exe 100000 48
./compute_pi.exe 1000000 48
./compute_pi.exe 10000000 48
./compute_pi.exe 100000000 48
./compute_pi.exe 1000000000 48
##
```

Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

Question 1

SBATCH RESULTS

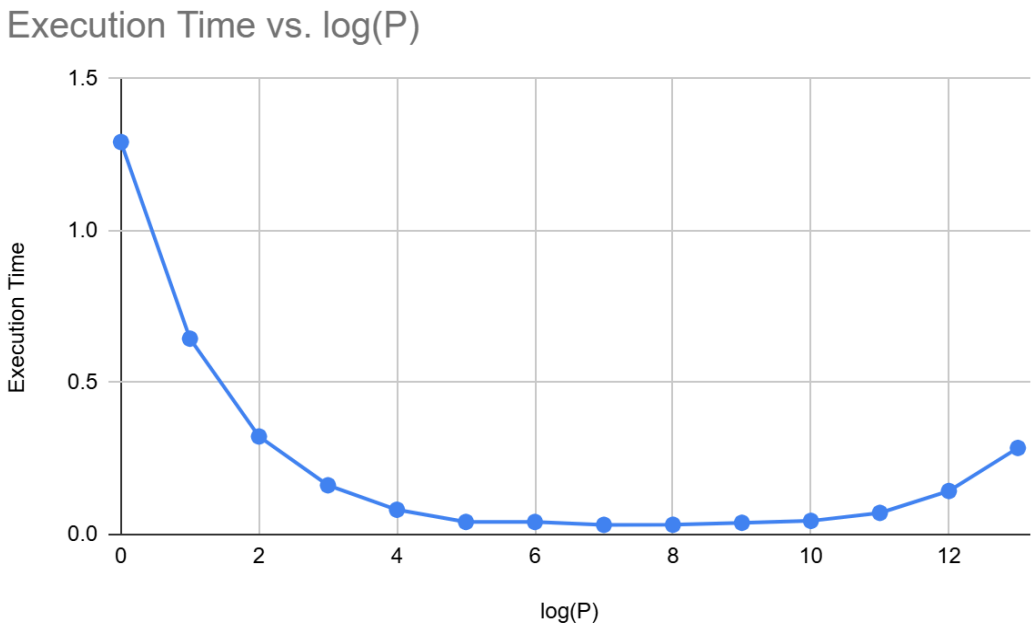
```
Trials = 100000000, Threads =    1, pi =   3.1416149600, error = 7.10e-06, time (sec) =   1.2918
Trials = 100000000, Threads =    2, pi =   3.1417022800, error = 3.49e-05, time (sec) =   0.6445
Trials = 100000000, Threads =    4, pi =   3.1415910800, error = 5.01e-07, time (sec) =   0.3226
Trials = 100000000, Threads =    8, pi =   3.1415947600, error = 6.70e-07, time (sec) =   0.1619
Trials = 100000000, Threads =   16, pi =   3.1415291200, error = 2.02e-05, time (sec) =   0.0814
Trials = 100000000, Threads =   32, pi =   3.1415901200, error = 8.06e-07, time (sec) =   0.0413
Trials = 100000000, Threads =   64, pi =   3.1413512400, error = 7.68e-05, time (sec) =   0.0412
Trials = 100000000, Threads =  128, pi =   3.1429299200, error = 4.26e-04, time (sec) =   0.0317
Trials = 100000000, Threads =  256, pi =   3.1412995200, error = 9.33e-05, time (sec) =   0.0320
Trials = 100000000, Threads =  512, pi =   3.1468450800, error = 1.67e-03, time (sec) =   0.0385
Trials = 100000000, Threads = 1024, pi =   3.1514026000, error = 3.12e-03, time (sec) =   0.0448
Trials = 100000000, Threads = 2048, pi =   3.1453611600, error = 1.20e-03, time (sec) =   0.0711
Trials = 100000000, Threads = 4096, pi =   3.1494162400, error = 2.49e-03, time (sec) =   0.1431
Trials = 100000000, Threads = 8192, pi =   3.1377031200, error = 1.24e-03, time (sec) =   0.2848
```

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.

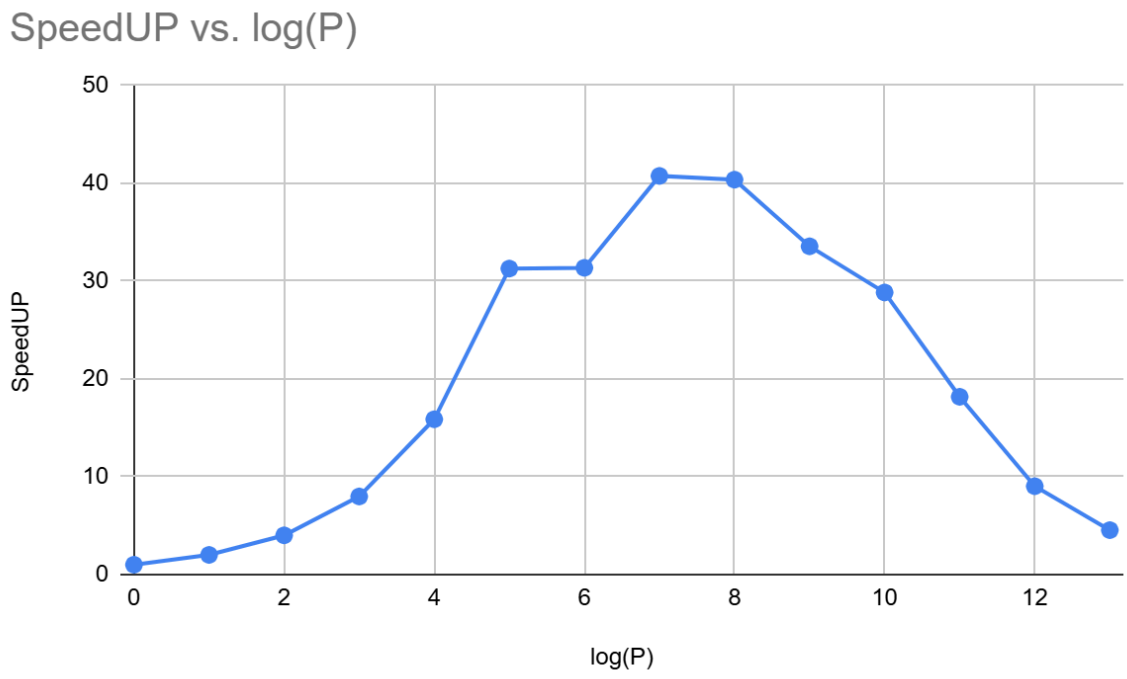
Threads (P)	log(P)	Execution Time	SpeedUP	Efficiency
1	0	1.2918	1	1
2	1	0.6445	2.004344453	1.002172227
4	2	0.3226	4.00433974	1.001084935
8	3	0.1619	7.978999382	0.9973749228
16	4	0.0814	15.86977887	0.9918611794
32	5	0.0413	31.27845036	0.9774515738
64	6	0.0412	31.35436893	0.4899120146
128	7	0.0317	40.75078864	0.3183655363
256	8	0.032	40.36875	0.1576904297
512	9	0.0385	33.55324675	0.06553368506
1024	10	0.0448	28.83482143	0.0281590053
2048	11	0.0711	18.16877637	0.008871472838
4096	12	0.1431	9.027253669	0.002203919353
8192	13	0.2848	4.535814607	0.0005536883065

Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

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

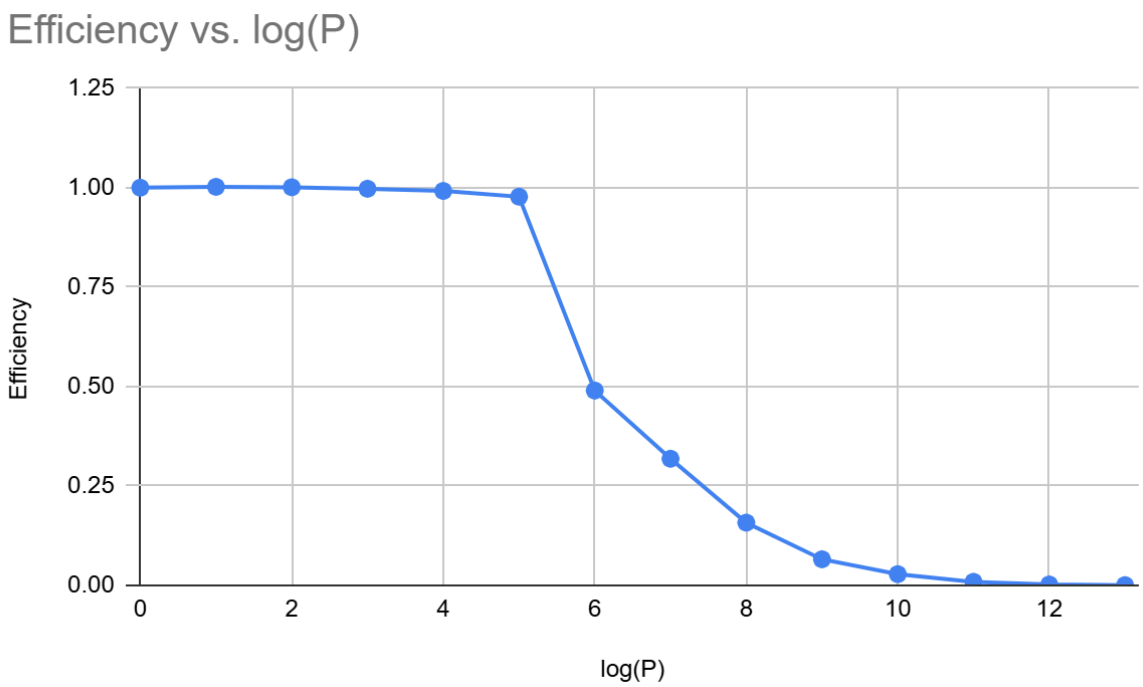


1.2. (10 points) Plot speedup versus p to demonstrate the change in speedup with p.



Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

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.



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

★ In my experiments, parallel runtime was minimized when with p=128.

Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

Question 2

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

SBATCH RESULTS

```
Trials = 10000000000, Threads =    1, pi =   1.4236202260, error = 5.47e-01, time (sec) = 128.6600
Trials = 10000000000, Threads =    2, pi =   3.1416070092, error = 4.57e-06, time (sec) =  64.3644
Trials = 10000000000, Threads =    4, pi =   3.1416060276, error = 4.26e-06, time (sec) =  32.1966
Trials = 10000000000, Threads =    8, pi =   3.1416118228, error = 6.10e-06, time (sec) =  16.0972
Trials = 10000000000, Threads =   16, pi =   3.1416066896, error = 4.47e-06, time (sec) =   8.0529
Trials = 10000000000, Threads =   32, pi =   3.1416332068, error = 1.29e-05, time (sec) =   4.0358
Trials = 10000000000, Threads =   64, pi =   3.1416139092, error = 6.77e-06, time (sec) =   3.0363
Trials = 10000000000, Threads =  128, pi =   3.1415943424, error = 5.38e-07, time (sec) =   2.7502
Trials = 10000000000, Threads =  256, pi =   3.1416473120, error = 1.74e-05, time (sec) =   2.7516
Trials = 10000000000, Threads =  512, pi =   3.1415762012, error = 5.24e-06, time (sec) =   2.7333
Trials = 10000000000, Threads = 1024, pi =   3.1414319268, error = 5.12e-05, time (sec) =   2.7226
Trials = 10000000000, Threads = 2048, pi =   3.1412588724, error = 1.06e-04, time (sec) =   2.7752
Trials = 10000000000, Threads = 4096, pi =   3.1407169720, error = 2.79e-04, time (sec) =   2.7671
Trials = 10000000000, Threads = 8192, pi =   3.1412633928, error = 1.05e-04, time (sec) =   2.8685
```

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

- ★ In this case, the parallel runtime is minimized with p=1024.

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?

- ★ It should be expected that the runtime will increase as p is increased beyond a certain value. When the number of threads gets too high, the overhead associated with managing communication and splitting the work between them begins to outweigh potential benefits of parallel computation. After a certain value, increasing the number of threads no longer leads to a beneficial speedup in computation time and the overhead can actually begin to hinder performance. My experiments were able to observe this happening. When p=1028, it achieved the fastest runtime. However, increasing p=2048, 4096, and 8192 resulted in longer execution times.

Question 3

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?

- ★ Yes, it should be expected that there is a difference in the number of threads needed to obtain the minimum execution time for two values of n. Since larger values of n have more tasks that need to be completed, it is easier to spread out the work amongst various threads and see improved computation times. However, for smaller values of n there are fewer tasks to be spread among the threads; using more threads than needed can result in load imbalance which will hinder execution times since there may be one or more idle threads. Using the same high number of threads for a small and large value of n would likely result in worse execution times for the smaller n.
- ★ My experiments did demonstrate this concept. For n=10^8, minimum execution time was when p=128. However, for n=10^10, the minimum execution time was when p=1024.

Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

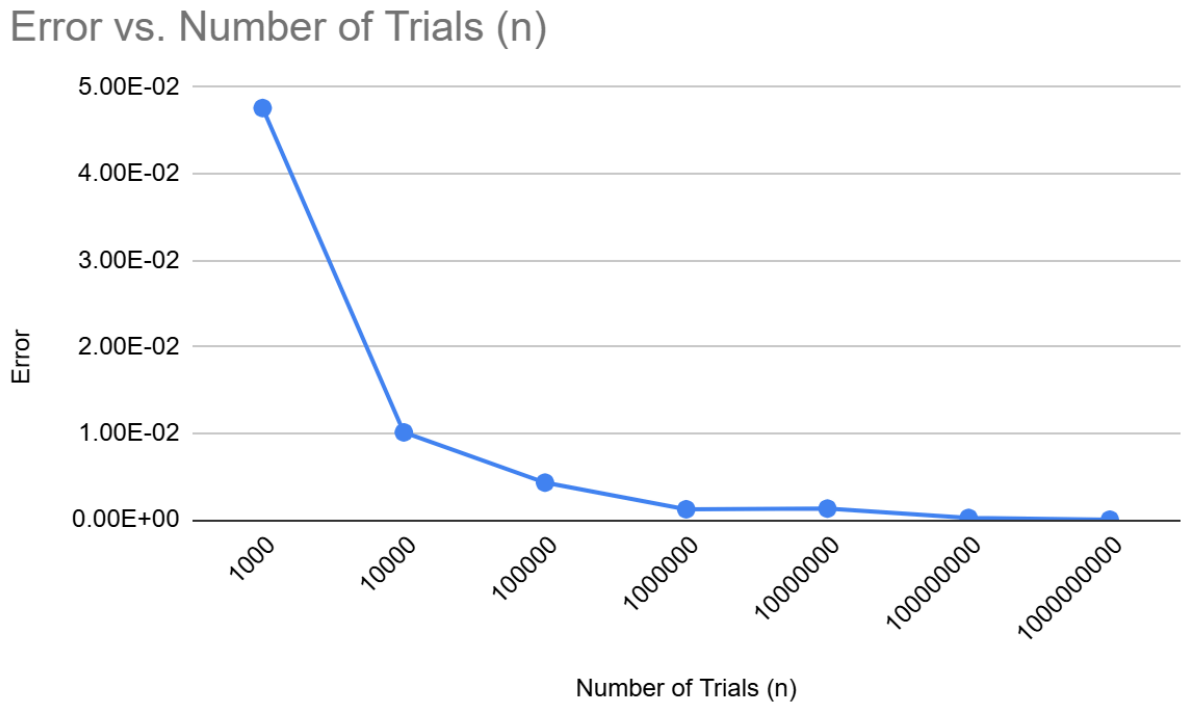
Question 4

SBATCH RESULTS

```
Trials = 1000, Threads =   48, pi =   2.9920000000, error = 4.76e-02, time (sec) =   0.0023
Trials = 10000, Threads =   48, pi =   3.1732000000, error = 1.01e-02, time (sec) =   0.0025
Trials = 100000, Threads =   48, pi =   3.1550800000, error = 4.29e-03, time (sec) =   0.0021
Trials = 1000000, Threads =   48, pi =   3.1453960000, error = 1.21e-03, time (sec) =   0.0022
Trials = 10000000, Threads =   48, pi =   3.1456372000, error = 1.29e-03, time (sec) =   0.0045
Trials = 100000000, Threads =   48, pi =   3.1409400400, error = 2.08e-04, time (sec) =   0.0284
Trials = 1000000000, Threads =   48, pi =   3.1416179080, error = 8.04e-06, time (sec) =   0.2713
```

Number of Trials (n)	Error
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

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.



Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

Question 5

5. Execute the code for n=10^8 with p chosen to be 2^k , for k = 0, 1, …, 6. Specify ntasks-pernode=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.

SBATCH SCRIPT

```
#!/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=735_HW1_Q5       #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.735_HW1_Q5.%j       #Send stdout/err to "output.[jobID]"
#
##OPTIONAL JOB SPECIFICATIONS
##SBATCH --mail-type=ALL              #Send email on all job events
##SBATCH --mail-user=jsalguero@tamu.edu    #Send all emails to email_address
#
##First Executable Line
#
module load intel             # load Intel software stack
#

#Question 5
echo "Question 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
```

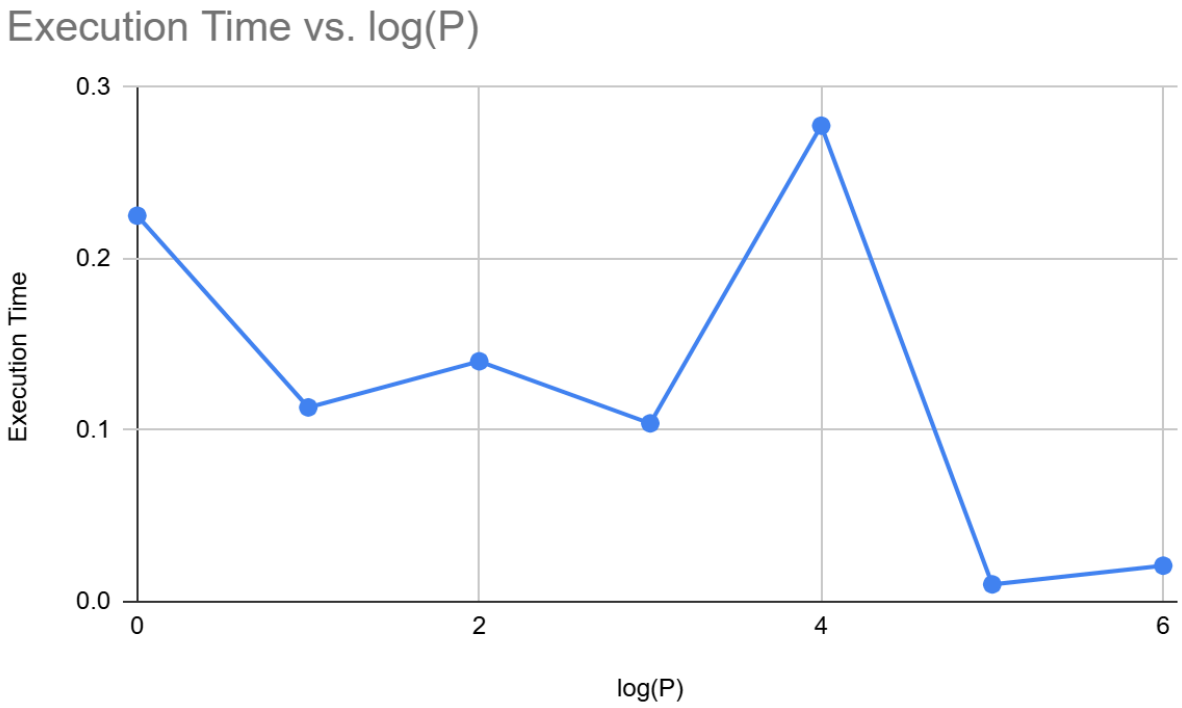
SBATCH RESULTS

```
Question 5
Processes = 1
n = 100000000, p = 1, pi = 3.1415926535900223, relative error = 7.29e-14, time (sec) =   0.2251
Processes = 2
n = 100000000, p = 2, pi = 3.1415926535902168, relative error = 1.35e-13, time (sec) =   0.1133
Processes = 4
n = 100000000, p = 4, pi = 3.1415926535896137, relative error = 5.71e-14, time (sec) =   0.1402
Processes = 8
n = 100000000, p = 8, pi = 3.1415926535897754, relative error = 5.65e-15, time (sec) =   0.1040
Processes = 16
n = 100000000, p = 16, pi = 3.1415926535897740, relative error = 6.08e-15, time (sec) =   0.2776
Processes = 32
n = 100000000, p = 32, pi = 3.1415926535897940, relative error = 2.83e-16, time (sec) =   0.0101
Processes = 64
n = 100000000, p = 64, pi = 3.1415926535897931, relative error = 0.00e+00, time (sec) =   0.0210
```

Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

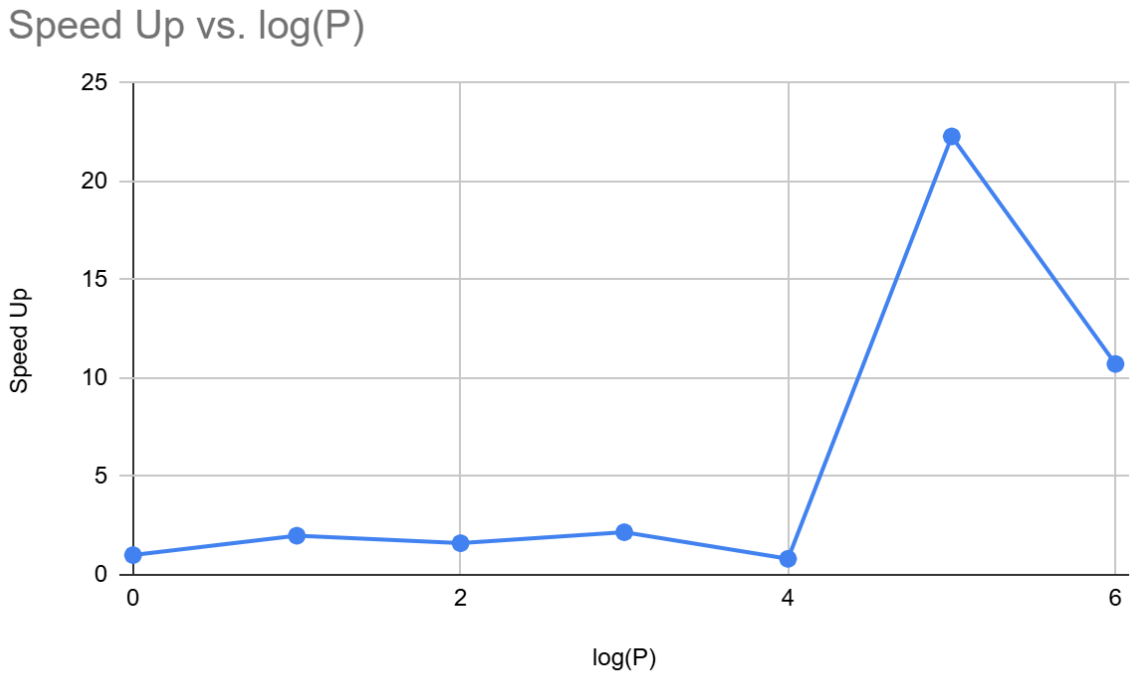
Threads (P)	log(P)	Execution Time	Speed Up	Efficiency
1	0	0.2251	1	1
2	1	0.1133	1.986760812	0.993380406
4	2	0.1402	1.605563481	0.4013908702
8	3	0.104	2.164423077	0.2705528846
16	4	0.2776	0.8108789625	0.05067993516
32	5	0.0101	22.28712871	0.6964727723
64	6	0.021	10.71904762	0.167485119

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

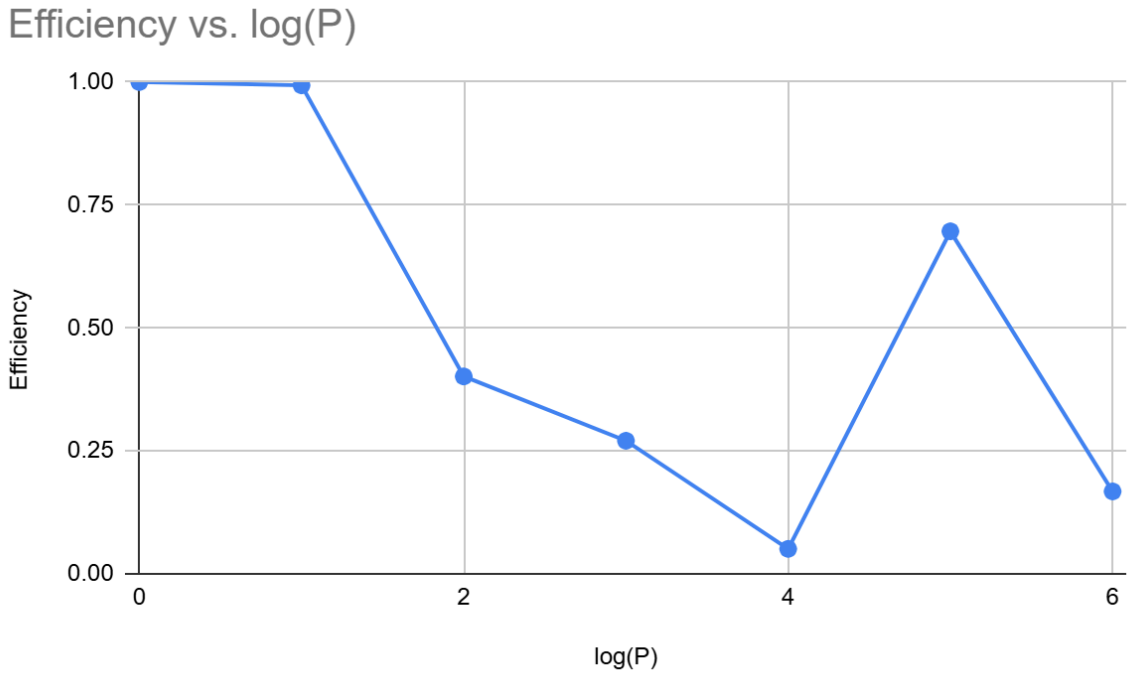


Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

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



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.



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

★ The parallel runtime is minimized when p=32.

Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

Question 6

Tested ntasks-per-node=1, 2, 4, 8, 16, 32. When I tried ntasks-per-node=64, I received this error:

```
[jsalguero@grace1 HW1]$ sbatch compute_pi_mpi_Q6.grace_job
sbatch: error: CPU count per node can not be satisfied
sbatch: error: Batch job submission failed: Requested node configuration is not available
```

SBATCH SCRIPT (ntasks-per-node=32)

```
#!/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=735_HW1_Q6       #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=32      #Request 4 tasks/cores per node <-- #VALUE CHANGING FOR Q6
#SBATCH --mem=8G                 #Request 8GB per node
#SBATCH --output=output.Q6-32.%j       #Send stdout/err to "output.[jobID]"
#
##OPTIONAL JOB SPECIFICATIONS
##SBATCH --mail-type=ALL              #Send email on all job events
##SBATCH --mail-user=jsalguero@tamu.edu    #Send all emails to email_address
#
##First Executable Line
#
module load intel             # load Intel software stack
#
echo "Question 6, n-tasks-per-node=32"
echo "Processes = 64"
mpirun -np 64 ./compute_pi_mpi.exe 10000000000
```

SBATCH RESULT (ntasks-per-node=32)

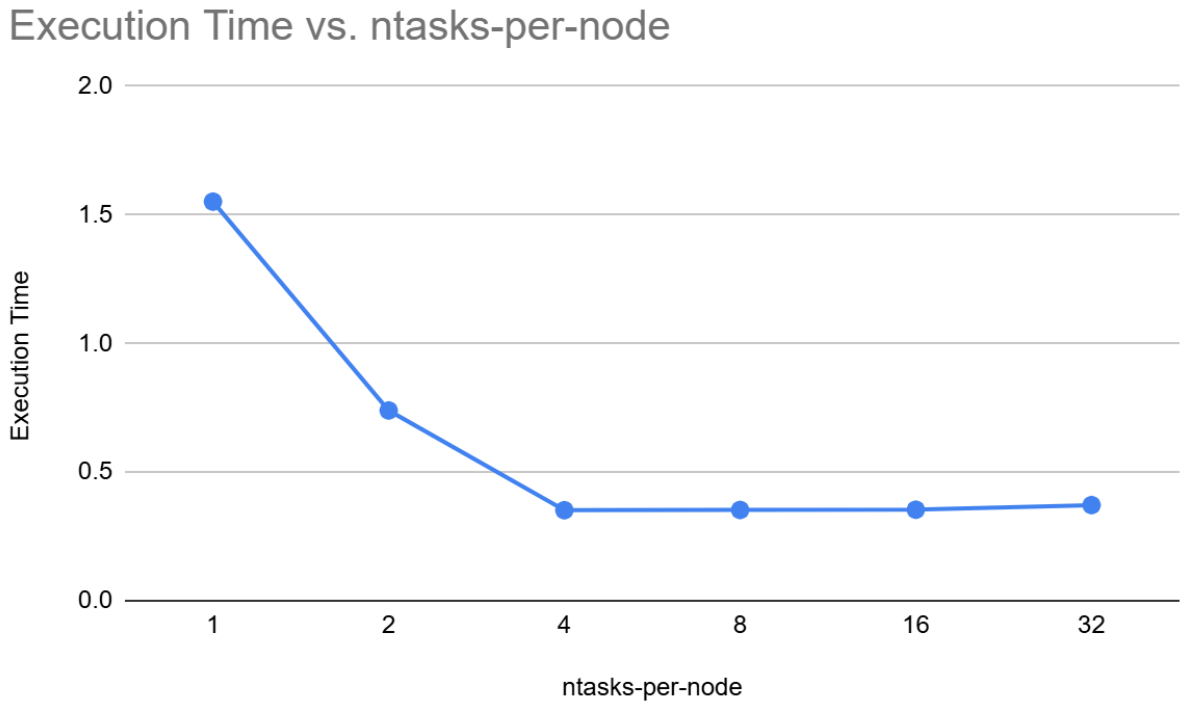
```
Question 6, n-tasks-per-node=32
Processes = 64
n = 10000000000, p = 64, pi = 3.1415926535897949, relative error = 5.65e-16, time (sec) =   0.3717
```

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.

★ The total execution time is minimized when ntasks-per-node=4.

ntasks-per-node	Execution Time
1	1.5515
2	0.7396
4	0.3518
8	0.3527
16	0.3537
32	0.3717

Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025



Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

Question 7

7. Execute the code with p=64 for n=10^2, 10^4, 10^6 and 10^8, with ntasks-per-node=4.

SBATCH SCRIPT

```
#!/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=735-HW1-Q7       #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.Q7.%j       #Send stdout/err to "output.[jobID]"
#
##OPTIONAL JOB SPECIFICATIONS
##SBATCH --mail-type=ALL              #Send email on all job events
##SBATCH --mail-user=jsalguero@tamu.edu    #Send all emails to email_address
#
##First Executable Line
#
module load intel             # load Intel software stack
#
echo "Question 7"
echo "Q7.1"
echo "Processes = 64"
mpirun -np 64 ./compute_pi_mpi.exe 10000000000
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 "Q7.2"
echo "Processes = 1"
mpirun -np 1 ./compute_pi_mpi.exe 10000000000
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
```

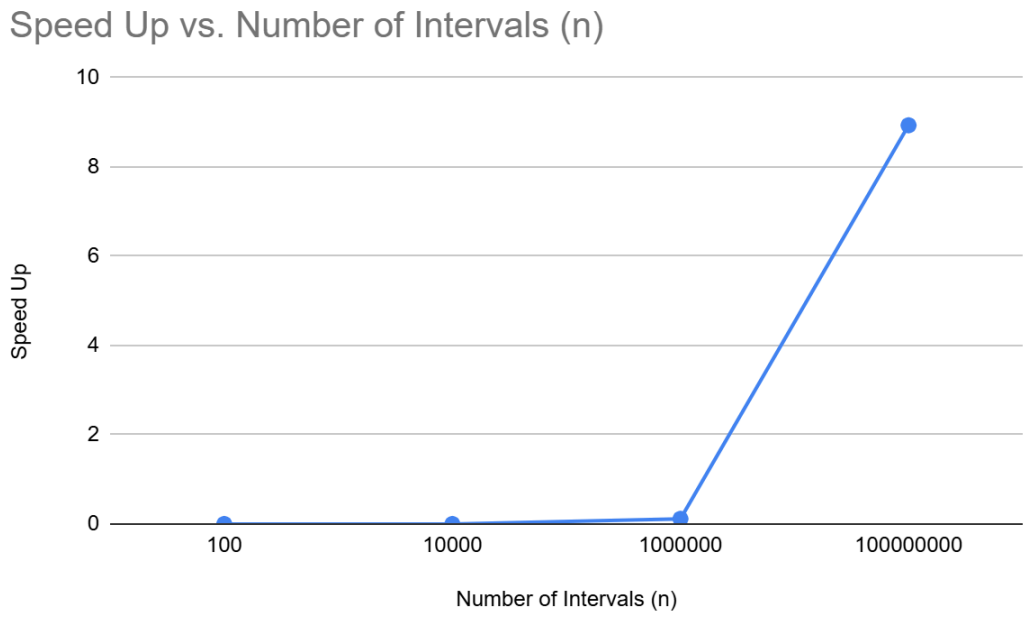
Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

SBATCH RESULTS

```
Question 7
Q7.1
Processes = 64
n = 10000000000, p = 64, pi = 3.1415926535897944, relative error = 4.24e-16, time (sec) =   0.3602
Processes = 64
n = 100000000, p = 64, pi = 3.1415926535897931, relative error = 0.00e+00, time (sec) =   0.0251
Processes = 64
n = 1000000, p = 64, pi = 3.1415926535898757, relative error = 2.63e-14, time (sec) =   0.0199
Processes = 64
n = 10000, p = 64, pi = 3.1415926544231265, relative error = 2.65e-10, time (sec) =   0.0200
Processes = 64
n = 100, p = 64, pi = 3.1416009869231249, relative error = 2.65e-06, time (sec) =   0.0195
Q7.2
Processes = 1
n = 10000000000, p = 1, pi = 3.1415926535897949, relative error = 5.65e-16, time (sec) =  22.4284
Processes = 1
n = 100000000, p = 1, pi = 3.1415926535900223, relative error = 7.29e-14, time (sec) =   0.2242
Processes = 1
n = 1000000, p = 1, pi = 3.1415926535899388, relative error = 4.64e-14, time (sec) =   0.0023
Processes = 1
n = 10000, p = 1, pi = 3.1415926544231318, relative error = 2.65e-10, time (sec) =   0.0000
Processes = 1
n = 100, p = 1, pi = 3.1416009869231245, relative error = 2.65e-06, time (sec) =   0.0000
```

Number of Intervals (n)	Execution Time (p=1)	Execution Time (p=64)	Speed Up	Relative Error (p=64)
100	0	0.0195	0	2.65E-06
10000	0	0.02	0	2.65E-10
1000000	0.0023	0.0199	0.1155778894	2.63E-14
100000000	0.2242	0.0251	8.932270916	0.00E+00

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.



Jessica Salguero, UIN: 937005006
CSCE 735-701, Summer 2025

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