# CSCE 735 Spring 2023

## HW 1: Parallel Programming on a Multicore Multiprocessor

### Part 1. Shared-Memory Programming with Threads

Compile and execute the program in the file `compute_pi.c`, which computes an estimate of π using the parallel algorithm discussed in class. It should be compiled and executed on `grace.hprc.tamu.edu`.

Load the Intel software stack prior to compiling and executing the code.

```
module load intel
```

To compile, use the command:

```
icc -o compute_pi.exe compute_pi.c -lpthread
```

To execute the program, use

```
./compute_pi.exe <n> <p>
```

where <n> represents the number of points and <p> represents the number of threads. The output of a sample run is shown below.

```
./compute_pi.exe 1000000 4
```

```
Trials = 1000000, Threads = 4, pi =   3.1433480000, error = 5.59e-04,
time (sec) =   0.0043
```

The run time of the code should be measured when it is executed in dedicated mode. Use the batch file `compute_pi.grace_job` to execute the code in dedicated mode using the following command on Grace:

```
sbatch compute_pi.grace_job
```

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.
   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.
   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?
2. Repeat the experiments with $n=10^{10}$ to obtain the execution time for $p=2^k$, for k = 0, 1, …, 13.
   2.1. (5 points) In this case, what value of p 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.
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.

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.

## Part 2. Distributed-Memory Programming with MPI

Compile and execute the program in the file `compute_pi_mpi.c`, which computes an estimate of $\pi$ using the parallel algorithm discussed in class. It should be compiled and executed `grace.hprc.tamu.edu`.

Load the Intel software stack prior to compiling and executing the code.

```
module load intel
```

To compile, use the command:

```
mpiicc -o compute_pi_mpi.exe compute_pi_mpi.c
```

To execute the program, use

```
mpirun –np <p> ./compute_pi_mpi.exe <n>
```

where <n> represents the number of intervals and <p> represents the number of processes. The output of a sample run is shown below.

```
mpirun -np 4 compute_pi_mpi.exe 100000000

n = 100000000, p = 4, pi = 3.1415926535897749, relative error = 5.80e-15, time (sec) =   0.0608
```

The run time of the code should be measured when it is executed in dedicated mode. Use the batch file `compute_pi_mpi.grace_job`, to execute the code in dedicated mode using the following command on Grace:

```
sbatch compute_pi_mpi.grace_job
```

5. Execute the code for n=$10^8$ with p chosen to be $2^k$, 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.
   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?
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.
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 $10^8$.

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

**Submission:** Upload a single PDF or MSWord document with your answers to Canvas.