CSE 6220 Spring 2021 -- Programming Assignment 1

Due Date: February 12, 2021

This is an introductory programming assignment meant for you to get your bearing in programming in C/C++ and some basic MPI commands. You are encouraged to discuss this in piazza in order to work out how to solve the problem.

Problem Description

In this programming assignment, you will use p MPI ranks to find the L2-Norm of a size N array x such that:

$$L2 = \sqrt{\sum_{i=1}^{N} x_i^2}$$

You can implement a parallel algorithm by slightly modifying the parallel sum algorithm that we discussed in the class as part of *CSE6220_03_Analysis-Model.pdf*.

Code Framework and Programming Guidelines

We are not providing you with a code framework for this project. However, there are some rules that you MUST follow:

- Name your code as *prog1.cpp* (only one source file is required for this project).
- We will generate the x array of type `double` using given seed C.
- The size of the array N and the seed for the random number generator C will be given as program arguments to the executable (*hint: you do not need to broadcast these arguments*). You can simply parse these arguments using `atoi/stoi` functions.
- You can use $srand48$ to set the seed and $drand48$ to generate the random numbers. The seed for each MPI rank should be $rank + C$.
- The number N should be a multiple of p. So, N/p will be an integer.
- Each MPI rank needs to generate N/p random numbers by using its $rank + C$ as its seed.
- Only one MPI rank should print the output as a single line at the end. See below for output format.
- Your code should only use point-to-point communications.

Hint: You may want to test your code first by summing up numbers from 1 to N. As a reminder, the sum should be

$$S = \sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

This formula can help you debug your code.

Input Format

Your executable must have two command line arguments:

- the number of elements N (try ≥ 500000), and
- the seed C for generating random numbers.

Again, you can assume N is a multiple of p, and also p is a power of 2. In command line, your code will be executed as: `mpirun -np p ./prog1 N C` such as:

```
$ mpirun -np 4 ./prog1 5000000 111
```

Output Format

Your submission should include an output file named `output.txt`, which contains five lines such that for each number of MPI ranks, in following format:

```
#N #p #C #L2 #Time
```

Example:

```
2048 4 1 25.842 0.000160
```

Note that:

- Time is the **total time** to generate the random numbers and compute $L2$-norm.
- You must keep N and C same while changing p.
- Your L2 output should be **rounded to 3 digits after the decimal point**
- Expected output format should be strictly adhered, and additional output (e.g., debug print statements you have forgotten to remove) will prevent you from getting full points. *Any additional output from your implementation should be avoided by the time of your submission.*

Compile and Run

You may implement, run, and test your solutions in your local machine, but you MUST use *Pace-ICE* cluster to report your results. The following commands are used to compile and run your code:

```
$ mpicxx prog1.cpp -o prog1
$ mpirun -np p ./prog1 N C
```

Teams

You must form teams up to three members. Only one person per team should submit your programming

assignment.

Grading

Your submission will be graded based on successful compilation, the accuracy of the output, and your analysis given in the report.

Submission Guideline

The submission to the Gradescope should be as single zip file called `prog1.zip` should include:

1. `output.txt` : output file specified earlier.
2. `report.pdf` : A **short report (not to exceed 2 pages)** that shows the speedup as you change N and P.
    - Your report must contain two plots:
        1. One should show the runtime while changing the number of processors p from 1 up to 16
        2. The other plot should show the runtime as you are changing the value of N (you must have at least 5 values for N).
    - You must **analyze your results** and write your **interesting observations** from the results.
3. `team.txt` : A text file containing the names of all team members.
4. `prog1.cpp`: Your source code.

You should also specify your teammates on your Gradescope submission. The source code should be self contained such that in Pace-ICE cluster, once we extract your code we should be able to compile and execute it without any additional parameters. Your source code must be properly commented.