CX 4220 / CSE 6220-A Introduction to High Performance Computing
Spring 2018
Programming Assignment 1
Due February 22, 2018

This is a simple assignment meant for you to get your baring in programming in C and some basic MPI commands. You are encouraged to work in groups or discuss this in piazza in order to workout how to solve the problem.

Introduction

A university conducts courses for hundreds of thousands of students. At the end of each semester for each course an online system gathers student scores for determining students final grade. These grades are calculated using a loan-out computing cluster called Pace at Georgia Tech.

Task

Using this cluster and based on a students final weighted score for the course, you are to calculate the final grade of each student and output how many students achieved each grade.
If the score of student i is $Score_i$, then:

$$\begin{aligned}
\left(Score_{avg} + \frac{2}{3}Score_{d1}\right) \leq & Score_i \leq Score_{max} && \Rightarrow \mathbf{A} \\
\left(Score_{avg} + \frac{1}{3}Score_{d1}\right) \leq & Score_i < \left(Score_{avg} + \frac{2}{3}Score_{d1}\right) && \Rightarrow \mathbf{B} \\
Score_{avg} \leq & Score_i < \left(Score_{avg} + \frac{1}{3}Score_{d1}\right) && \Rightarrow \mathbf{C} \\
\left(Score_{min} + \frac{1}{5}Score_{d2}\right) \leq & Score_i < Score_{avg} && \Rightarrow \mathbf{D} \\
& \text{Otherwise} && \Rightarrow \mathbf{F}
\end{aligned}$$

$Score_{max} = \max_{i \in n} Score_i$,
$Score_{min} = \min_{i \in n} Score_i$,
$Score_{avg} = 1/n \times \sum_{i \in n} Score_i$,
$Score_{d1} = Score_{max} - Score_{avg}$,
$Score_{d2} = Score_{avg} - Score_{min}$

Guidelines/Rules

- You code should be written in C language
- **Compilation and execution should work out of the box at Pace-ICE cluster.**
- Expected output format should be strictly adhered since the grading for this is automated and additional output (eg: debug print statements you've forgotten to remove) will prevent you from getting full points.
- **Only for this Programming Assignment** you are encouraged to discuss the solutions openly on piazza. The TAs will also help you with what needs to be done and any problems you have in programming in C and/or MPI.

Code Framework

- We are providing you with a code framework that will have a driver to run your code for a particular input and a header file defining the functions which you need to implement in order to solve this problem. Download the code framework provided at `https://github.gatech.edu/ucatalyurek7/cse6220-Sp18-src/tree/master/prog1-template`.
- Implement the functions defined in `mpi_score_analyser.h` header file inside `mpi_score_analyser.c` source file using `C` language (avoid having any `C++` functions or data structures in your final submission).
- Additionally in order for you to test the correctness of the results from running your parallel algorithm you may implement the serial version of the score analyser by implementing the functions defined in `score_analyser.h` header file inside `score_analyser.c` source file.
- Please refer to the `README.md` provided with the code framework for more details about the structure of the framework.
- You may add new files and update the Makefile as needed. However Do not change the `mpi_score_analyser.h` header file.
- You should maintain the same project structure resembling the code framework.

Input Format

- The student scores for your program can be provided via a file containing the scores of each student. Each score should be separated by a new line.
  Note: Your implementation of the function `mpi_distribute_scores(...)` will be triggered only through this method of input.

```
$ mpirun -np 4 ./grader_cal -f scores_100.txt
```

Listing 1: scores_10.txt

```
41.7022
72.0324
0.0114
30.2333
14.6756
9.2339
18.626
34.5561
39.6767
53.8817
```

- You can also pass an n value for the number of students and the code framework will randomize the scores for those students.

  ```
  $ mpirun -np 4 ./grades-cal -n 10000
  ```

  Note: For the same value n the random numbers generated for serial version is NOT as same as for the parallel version.

Output Format

In the provided code framework, the output will be printed by processor 0. It will also print the the execution time for you to compare against the serial algorithm.
(The printing of these values is done by the code in `main.cpp`. Any additional output from your implementation should be avoided at the time of your submission)
eg:

```
$ mpirun -np 4 ./grades-cal -n 10000
5.071e-05
0: 1655
1: 1633
2: 1701
3: 4026
4: 985
```

Teams

- You may form teams of up to 3 members. However we encourage you to do this assignment as an individual assignment to test and improve your programming skills before the next

programming assignments which would take more effort.

- If you do decide to form teams to work on this assignment, we expect you be responsible for managing the workload among yourselves. However each of the team member needs to be knowledgeable of completing other team member(s) tasks.
- Regardless whether you worked this assignment individually or as part of a team, please update the file `team.txt` with the `gtusername` of each team member one per each line.

Listing 2: team.txt

eg:

```
ayasar3
dwijeratne3
```

Testing

You may implement, run and test your solutions in your local machine (see the references section to see how), but you MUST test your code in the Pace-ICE cluster for the following.

1. Compilation should be done via the `Makefile`. You should call the compilation and execution through scheduling a pbs job or via an interactive node. See the "References and Resources" section on how to do this.
2. Use the modules `gcc` and `mvapich2/2.2` for compilation and execution.
3. Fresh copy of your solution should compile out of the box without any changes to the `Makefile` you have provided.
4. Generate input files for small values of n and verify that your parallel solution gives correct results. You may do this manually or by comparing against your serial implementation. For your convenience we have also introduced unit testing for both your serial and parallel implementations. You are welcome to add your own unit tests to these files. Refer the `README` on how to build and execute the unit tests.
5. Your solution should execute and give results for very large values for n. We will let you work-out what is the largest n that your algorithm supports.
6. Test your parallel algorithm at least up to 64 processors. We want you to test the limits of your solution, find-out how to stretch it further if it need be. You may assume $n \geq p$.
7. Resources in Pace-ICE cluster is limited. And this class has over 150 students. In order to avoid delays in testing your solution in the cluster, we urge you to not way until the last day.

Grading [10pt]

- The highest score you can get for this assignment is 10pt
- Your submission will be graded based on successful compilation and the accuracy of the output on the Pace-ICE cluster.
- A submission made by a team will receive the same grade for all members of the team.

Since this assignment is meant for you to get some hands on experience in programming in `C` and using the MPI framework, we will not be checking for efficient implementations. But you should strive to do that nonetheless.

Submission Guideline

- The submission to the T-Square should be in zip format. 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.
- If you are in a team, only one of the team members need to submit the assignment. Double check to see if the `team.txt` file is updated correctly.
- This assignment does not require you to compile a report. However your source code must be properly commented.

Important:
Since the directory `gtest/` contains large files, before submitting please REMOVE this directory from your final submission.

References and Resources

- If you are new to programming in MPI and working with a cluster, `PaceIceCluster.pdf` and `GettingStartedwithMPIlocally.pdf` is a good place to start. You will find these documents under the `Resources` section in T-Square.
- We recommend you code your first MPI hello world program and execute it locally and in the `Pace-ICE` cluster before you start working on the assignment.