COSC 420 - MPI - HW1

Dr. Yaping Jing

Imagine you're a newly hired software/data analyst for an online business such as Amazon or eBay. You're asked to create an MPI program in C (save it as `OnlineRatings.c` under a folder of your name) that computes average ratings on a range of online products (or services). Let `m` be the number of products to be rated, and `n` be the corresponding online ratings with each possible score from 1 to 5 in integer. Your program is to compute the average ratings for each of the products and sort them in the end. Specifically,

- Your program shall take `m` and `n` as *arguments* (i.e. NOT through prompt message dialogue) when user runs your mpi program.
- Each score (in the range of [1, 5]) shall be randomly generated. You may use the current time as the seed (`srand(time(NULL))` and call `rand()` with a simple formula to get those scores.
- Since we only covered `MPI_Send` and `MPI_Recv` for the point to point communication, your program shall be restricted to using these blocking methods for this assignment, so as to show your understanding of these fundamental concepts.
- You should assign one process (corresponding to a core) to be the master whose responsibility include preparing data, and send an array of scores to each worker process to compute the average. Because each array has the same type of data, the tag can be the same when it's sent from the master process to each of it's worker processes. When a worker process finishes it's computation, it sends the result (the average rating) to the master process. Of course, you will use a different type of tag for the reply message. Once the master process receives the results from all the worker process, it sorts them and display to the console in 1 digit after the decimal point from highest rating to the lowest rating for each product indexed from $1...m$.
- For simplicity, for this assignment, you may assume `m`, the number of products to be rated, has its maximum value to be equal to the total number of cores minus `1` of your cluster. For example, if your cluster has a total of `56` cores, then the maximum value of `m` should be `55`. You're welcome to try a larger number, in that case, each worker process will need to compute several rows, depending on how evenly you distribute the workload among the worker processes.

- For n, the number of ratings, it can be as large as a million, or as small as just 5. The reason is that I would like you to test the performance of the cluster if the size is large enough.

The following matrix illustrates some sample data and it's rating results. Again, for simplicity, you can prepare one array at a time and assume they all have the same number of ratings.

1	5	4	3	1	4
2	1	2	3	1	4
3	3	5	1	5	2
4	2	4	3	4	2

```
Sorted Ratings:
1: 3.4
3: 3.2
4: 3.0
2: 2.2
```

To benchmark your parallel performance measures, you may use the following code snippet.

```
double t1 = MPI_Wtime();
...
double t2 = MPI_Wtime();
printf("MPI run time: %4.1f\n", t2-t1);
```

For this assignment, formal performance measure and analysis is optional. If you're interested, you can implement a serial version such that it just uses one process to do all the computation. Then, you compare the measured time with that of parallel version (use various number of available processes). You can further tune the m value to allow each process to compute multiple rows of data in order to observe the power of the cluster. Above all, the interesting question is, at what m and n values, the parallel version is faster than the serial version. Please let me know if you have a chance to test them out. If you do, please submit both the serial version and parallel version, and put the performance measures discussions either inside your program or submit a separate PDF file. Otherwise, you may just submit one file which is `OnlineRatings.c` under MyClass/HW1 Assignment as required. As with any programming assignment, you will lose a significant amount of points if your program cannot pass any test cases.