Group Assignment
Parallel Sorting Algorithms for Sections 1 and 3
Semester 1/2022

Due date: Saturday, October 22, 2022 at time 23:59

A group of 3 students is assigned to implement one parallel sorting algorithm. One group will be assigned one of the 8 parallel algorithms to work as we studied all of them in class. The list of all parallel algorithms and the details of the implementation are briefly explained below.

1. Parallel Quicksort via OpenMP
2. Parallel Quicksort via MPI
3. Parallel Mergesort via OpenMP
4. Parallel Mergesort via MPI
5. Parallel Bucketsort via OpenMP
6. Parallel Bucketsort via MPI
7. Parallel Ranksort via OpenMP
8. Parallel Ranksort via MPI

For each parallel sorting algorithm, you need to do the followings.

1. Generate 10000, 100000, and 1 million (1,000,000) random integer numbers (ranging from 0 to 9999) for running the sorting.
2. The number of compute nodes (MPI) or the number of threads (OpenMP) is varied from 1, 4, 8, 12 and 16. Also, measure the execution time for each data group and for each number of compute nodes. Put the results into a table and also plot one graph having all results.
3. If the number on each node is below a threshold, say 1000, you can call another sorting function to sort those numbers locally.

What to submit to the link provided on MyCourses

All submissions must have Project name and ID of all members.

You need to submit only one file contains 2 parts (zipped them together).

1. **Code Part (50 points)**
   - One zip file includes source code, executable code and a readme file telling how to run your code, e.g. an example of a command and what parameters to put in.
   - All source codes will be compiled and run on the ICT cluster system.

2. **Report Part (50 points)**
   - One PDF report file having the following:
     - Topic and members (names and IDs)
     - Explanation of your program as a diagram or a flowchart, but don't draw the flowchart to represent every line of the code since it is useless. You need to explain how the algorithm was implemented in details; for

example, the outline of your code and data structures used. There is no need to explain the algorithm (as we already did in class).

- Testing results as a table with sample capture screen shots showing part of the output and the execution time.
- Speedup graph showing for all data sets with varying compute nodes.
- Explain the results as shown in the graph.