Parallel and Distributed Computing

Assignment – 01 (ALL PDC SECTIONS)

Deadline: Monday 20th April 2021 by: 11:59pm (not extendable)
Max points: 5 Absolutes

Instructions:

1. For this assignment you are required to write programs in either C or C++.

2. You MUST use OpenMP and/or MPI to complete the tasks. Do not use pthreads (for multi-threading).

3. You can only use the OpenMP directives studied in the class and/or point-to-point communication primitives (MPI_Send and MPI_Recv) of MPI. If you are coding the assignment in MPI then use of collective and asynchronous calls are not allowed.

4. Your programs must compile without warnings and execute correctly for full credit. Use good programming style, including the use of an appropriate amount of comments and suitable naming conventions.

5. In submitting this work, you are agreeing that it can be checked for plagiarism. **Any plagiarized submission will be dealt with course plagiarism policy (zero marks for all current and future assignments)** as announced and mentioned in the course outline.

6. You MUST submit the screenshots of code execution. All the outputs (screenshots) should include your name and roll number. All the screenshots must be included in a single zip file. The name of the zip file should be as follows (this carries weight): ***PDC_Spring2021_Assign1_[YOUR_ROLLNO]_ [YOUR_SECTION].zip***

7. The breakdown and weights of different sub tasks is mentioned below. It is your responsibility to submit as many screenshots as needed (place them in a single file) to convince the course instructors that you have successfully completed the task. For instance, to convince that the processes abort the search once abort message is received from the 0-ranked process, you can make some particular process (not expected to find the number) to sleep in each iteration while searching. This will make it take a lot of time to search and once the abort message is received it cancels the search.

8. **Late submission (even 1 minute) means No-Submission**. To avoid any potential technical issue at the submission day, submit it as soon as possible.

Problem Statement:

The process ranked 0 (let’s call it ***master*** and not to be confused with the machine named master) initializes and distributes a large list of numbers amongst other processes (let’s call them slaves). The list size and distribution depend on the number of slave processes and each one is assigned an equal share of the numbers to be searched. **[This step carries one absolute]**

The master process then waits for the slave processes to search and report back. Each slave process is doing the similar tasks. First, it is comparing each number in the chunk assigned to the number being searched and secondly at the same time, the slave process is waiting for the abort message from the master process (in case any other process finds the number). For the first case, if the process finds the number, it breaks the search and sends a message to the master that it has found the number. The master process is waiting for exactly this message, it then notifies all the other processes to abort the search.

The important thing here is that a slave processes should abort the search as soon as the abort message is received from the master process. For example, if a process is busy searching 100k numbers and have say searched 5 as yet, and if it receives the abort message it should abort searching immediately rather than the complete the search within 100k numbers. **[This step (abort while searching) carries three absolutes. See instructions to present your implementation using screenshots].**

Please properly comment your code, submit the detailed screenshots and carefully and completely follow the instructions. **[This step carries one absolute but points and can ONLY be awarded if you significantly complete the steps above].**

A sample execution flow is as follows:

```
Master: The number to search is 39
Process 0 has input data: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
Process 2 has local data: 21 23 25 27 29 31 33 35 37 39
Process 1 has local data: 1 3 5 7 9 11 13 15 17 19
Process 2:  I have found the number :-)
Master: Process 2 has found the number!
Master: Informing all processes to abort!
Process 1: Aborting search!
```

Start early, best of luck!