Parallel/Distributed Computing (CSEG414/CSE5414) Assignment #1

2020. 10. 10

Due Date: Nov. 1st (Sunday)

1. **(10 Points)** Suppose $T_{serial} = n$ and $T_{parallel} = n/p + \log_2(p)$, where times are in microseconds. If we increase p (# of processors) by a factor of k, find a formula for how much we'll need to increase n in order to maintain constant efficiency. How much should we increase n by if we double the number of processes p from 8 to 16? Determine whether this parallel program is (weakly) *scalable* or not.
   Hint: Suppose we increase the number of processors, p. If we can find a *corresponding rate of increase* in the problem size so that the program always has the same efficiency E, then the program is (weakly) *scalable*.

2. **(40 Points)** Finding **prefix sums** is a generalization of global sum. Rather than simply finding the sum of n values, $X_0 + X_1 + X_2 + \ldots + X_{n-1}$, the prefix sums are the n partial sums X_0, $X_0 + X_1$, $X_0 + X_1 + X_2$, ..., $X_0 + X_1 + \ldots + X_{n-1}$. MPI provides a collective communication function, *MPI_Scan*, that can be used to compute prefix sums.

   (1) Understand the semantics of *MPI_Scan* operation and devise at least **two** parallel prefix sum algorithms (i.e., Explain the algorithms without MPI notation).
   (2) Implement this operation using only MPI send and receive (**blocking** and **non blocking**) calls. When implementing your solutions, make sure that your implementation is not dependent on the number of processors used. Verify your results by generating n random integers and compare the performance with that of **original** *MPI_Scan* as you increase the number of nodes involved and n. Discuss the results.

3. **(75 Points)** It is generally agreed that topics in image processing have the high potential for significant parallelism. In this question, you are to read in a **PPM** (Portable Pix Map) file in **P6 format** (full color), and write sequential and parallel programs written in C and MPI to
   (a) *flip* an image horizontally (mirroring) and
   (b) reduce the image to *grayscale* by taking the average of the red, green, and blue values for each pixel and
   (c) *smooth* the image by calculating the mean of each pixel's value and its eight neighbors (some algorithms consider only the values from the diagonal neighbors or the horizontal and vertical neighbors).
   When implementing your MPI programs, try to use *MPI derived data types* as much as you can. Compare the performance of sequential and parallel versions of the

program and discuss the results over a cluster of workstations. Use different PPM files with *various* data sizes and discuss the scalability aspects of your code as you increase the number of nodes. When you submit your code, include the short report on the questions above, the sample PPM files used and your programs (sequential and parallel versions). Also include the name of the PPM viewer(Linux version) in a *readme* file.

4. **(75 Points)** In this assignment, you are required to implement a calculating server and its corresponding client program using **SUN RPC**. The calculating server will provide service procedures such as addition(+), subtraction(-), multiplication (*), division(/), and x power y(x^y) of any two integer values. In the client side, you can implement a command-line calculator that accepts an expression (consisting of only integer values and five arithmetic operators (+, -, *, /, **)) and prints out the corresponding answer. Here is an example of the client program.

```
assignment> test Enter↵
2+3*5+2**3 Enter↵
The answer is 25
```

< How to submit >

Create a **tar** file with the name of "pdc1_학번.**tar**" that consists of your source file(s) and a makefile, a readme file, and a hwp file containing the answers for each question (Do not zip your files before making a tar file). Then, upload the **tar** file to Sogang Cyber Campus.