CSCI 503: Parallel Programming Homework 2

Homework Overview

There are two parts to this homework assignment. The first part is a programming assignment and the second part is discussion. Some of the discussion questions may be related to the programming assignment, while others are related to the lecture.

Homework Submission

To help us get your homework graded quickly and efficiently, please observe the following rules when preparing your homework for submission to the Blackboard:

- Your homework package must include:
  - **Makefile** and *all source code*. We recommend that you test out your code on aludra.usc.edu to ensure compatibility.
  - **report.pdf**. Your write up must be in PDF format (as it will include graphics). We strongly recommend that you use LaTeX to do your homework report. A sample LaTeX template is included in your homework package. *Do not submit Microsoft Word documents.*
- Be sure to label everything clearly, e.g. axis, plot title, etc.
- Your homework package must be submitted in a single zip (or tar ball) file named: **LASTNAME_FIRSTNAME_HW3.zip** or **LASTNAME_FIRSTNAME_HW3.tar.gz**. Please note that last name, first name, and hw3 are ALL CAPS. The extension zip or tar.gz are not capitalized.

1. To encourage you (a graduate student) to provide a quality product, and
2. To help us grade your homework efficiently. This filename convention helps us minimize mistakes when assigning your grades and saves us the time it takes to sort ~ 40 files with different formats and generic names such as cs503_homework_2.xxx, cs503-homework2.xxx, hw2-cs503.xxx, and so on.

The bottom line: Should you lose 10 points for not following directions, please exercise extreme restraint by refraining from pleading for a change of grade.

Programming

For this homework assignment, we will write a parallel image processing application using MPI. We've included a serial version of the image processing for your reference. When in doubt, please refer to the demo program as a starting point. Your task is to modify **parallel.cc** and **image.cc** with the appropriate MPI library calls and image manipulation. [25 points]

The idea is that the application will read in an image, partition it into a set of subimages, and distribute the subimages to *N* processors, where each of processors will perform the image processing routine(s). The number of processors will be determined at runtime. The worker processors will return their portion to the master node, where subimages are assembled back to a single large and processed image.

Two things to be on the lookout for: (1) the compute kernel, more specifically how you use it, and (2) the edge condition of the images. Needless to say the resulting image for a parallel filter must product the exact result as that of a serial filter.

Please download homework2.zip package, modify your code and report, and submit one zip or tar ball file to BlackBoard.

Development Environment

Your code will need to run on the USC HPC cluster, it is recommended that you develop and test your code there. You may need to link with the tiff-3.8.2 library. From what we can tell, the software can only be compiled on **hpc-login2.usc.edu**.

Make sure you source: **/usr/usc/mpich2/1.3.1..10/setup.csh** or **/usr/usc/mpich2/1.3.1..10/setup.sh** (for bash) before compiling.

Testing your code

In order to test your code on the HPC cluster, you will need own bake or modify your own custom pbs scripts. Two sample .pbs scripts are provided for your reference.

Note that your code must compile cleanly with the **make** command.

Summary of programming assignment deliverables:

- Modify the code to perform distribute the image processing tasks using MPI
- Your code should support both the **mean** and **median** filters in parallel mode
- Test your code on the HPC cluster to ensure maximum compatibility

Discussion

Answer all problems in as thorough detail as possible. Be sure to include all your work. Partial credit will be given even if the answer is not fully correct.

1. Please discuss the advantages the message passing programming model has over the shared memory programming model; and disadvantages (if any) [3 points].

2. Based on your programming assignment, please discuss what is your understanding of a compute kernel [3 points].

3. Based on your programming assignment, please discuss the edge-condition [3 points].

4. You are not required to comprehensive timing analysis for the programming assignment. However, please discuss how/why a median filter might stand to gain more for the parallel implementation than a mean filter [5 points].

5. Given the unsafe MPI programming practice below, please describe the problem with the code and if applicable please recommend a solution [10 points].

```
...
if (rank == 0) {
  for(i=0; i<4096; i++)
     data[i] =  'x';

  start = MPI_Wtime();
  while (1) {
    MPI_Send(data, 4096, MPI_BYTE, dest, tag, MPI_COMM_WORLD);
    count++;
    if (count % 100 == 0) {
      end = MPI_Wtime();
      printf("Count = %d  Time= %f sec.\n", count, end-start);
      start = MPI_Wtime();
      }
    }
 }
if (rank == 1) {
  result = 0.0;
  while (1) {
    MPI_Recv(data, 4096, MPI_BYTE, source, tag, MPI_COMM_WORLD,
      &status);
    for (i=0; i < 1000000; i++)
      result = result + (double)random();
    }
}
....
```