CS426
Spring 2020
Project 1 v2
Due 11/03/2020 – 23:59

Input files can be found on this link:

http://www.cs.bilkent.edu.tr/~ozturk/cs426/input.txt

http://www.cs.bilkent.edu.tr/~ozturk/cs426/image.txt

PART A: Find the Min

The objective of the problem is to write a serial, then a parallel program that takes as input an array of integers, stored in 'input.txt' with one integer per line, and prints out the min of the elements in the file.

```
% cat input
7
28
9
5
19
12
% my_program input
5
```

You will write several versions of this program in serial and in MPI.

- **Serial:** Write a serial program to solve this problem. Name the source code min-serial.c.
- **MPIv1:** Write an MPI implementation of the above program in which a master process reads in the entire input file and then dispatches pieces of it to workers, which these pieces being of as equal size as possible. The master must also perform computation. Each processor computes a local min and results are then collected by the master. Master then finds the real min value from the local min values. This implementation should not use any collective communications, but only point-to-point. Name the source file min-mpi-v1.c.
- **MPIv2:** This is similar to MPIv1 except that all processors should have the computed overall min. In this implementation, you are expected to use collective communication features of MPI.(MPI_Allreduce()/MPI_Bcast()) Name the source file min-mpi-v2.c.

PART B: RGB to Grayscale Conversion

RGB to grayscale conversion is done by applying a dot product over the red, green and blue channels, i.e, color dimensions, of an image using the following algorithm:

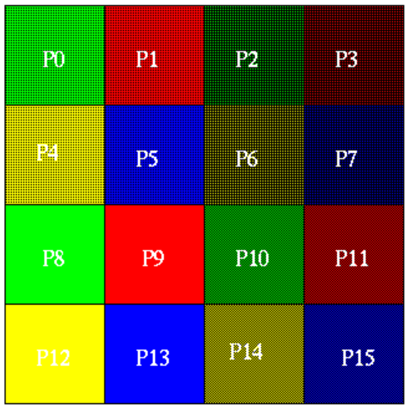
```
For each pixel [r, g, b] at positions [i, j]
    grayPixel[i,j] = 0.21*red + 0.71*green + 0.07*blue
```

You will move on the image and find the dot product of the RGB values with [0.21, 0.71, 0.07]. The input matrix which is square is stored in 'image.txt' whose first line gives the number of rows and the following lines are the comma separated RGB values in the row major order. The program writes the resulting matrix to the output file in the same format as the input without dimension.

You will write several versions of this program in serial and in MPI.

- **Serial:** Write a serial program to solve this problem. Name the source code greyscale-serial.c.
- **MPIv1:** Write an MPI implementation of the above program in which a master process reads in the entire input file and then dispatches rows of it to workers, which these rows being of as equal size as possible. The master must also perform computation. Each processor makes the computation for its rows and results are then collected by the master. This implementation should not use any collective communications, but only point-to-point. You also have to send neighbor rows for the borders of row patches. Name the source file greyscale -mpi-v1.c.
- **MPIv2:** Write an MPI implementation of the same problem with a different partitioning scheme. Master process reads in the entire input file and then dispatches it to workers. The master must perform computation as well. Assume that the number of processors used is a perfect square (4, 9, etc.), and that the matrix dimensions are perfectly divisible by the square root of the number of processors (e.g., if matrices are 100x100, then we use 16 processors). The processors are thought of, logically, as organized in a 2-D "processor grid".

  As seen in the figure, each processor works on a grid only given to it, and returns the result to the master. Then master will dispatch again, computations performed and sent back to master. The master then writes the output file. Don't forget to send necessary row and column segments for the grid borders while dispatching. Name this source code greyscale -mpi-v2.c.



- **MPIv3:** Can you improve the performance of MPIv2? One improvement could be not to send the calculations to master, thereby eliminating the need for broadcasting the data. (Hint: inter-process communication.). Name this source code greyscale -mpi-v3.c.

Submission

- Send a single zip file (yourname_lastname_p1.zip) that includes:
- Your implementation with source files and necessary input files for the following
  - min-serial.c
  - min-mpi-v1.c
  - min-mpi-v2.c
  - greyscale -serial.c
  - greyscale -mpi-v1.c
  - greyscale -mpi-v2.c
  - greyscale -mpi-v3.c
- Your outputs generated by your implementation. It should be clear to which program this output belongs (e.g greyscale -serial-output.txt, greyscale -mpi-v1-output.txt etc.), and mention how many processes used and which input file was fed to it in the report for each output file.
- Run your code with various number of processes
- Your report (3 pages at most):
  - Explain the implementation and design choices
  - Plot a graph with various process numbers indicating the performance of your implementation. Use sequential implementation as a baseline.
  - Your observations about the performance of your implementation, how you interpret the results etc.
- We will not test your codes with the given inputs, so run your code with different dimensions.

Email: berkay.gulcan@bilkent.edu.tr
Email subject: CS426_HW1 (Without this subject, your project will not be evaluated).
Zip File name: yourname_lastname_p1.zip (Without this name, will not be evaluated).
No Late Submission Allowed!