Programming Assignment III: MPI Programming

The purpose of this assignment is to familiarize yourself with MPI programming.

1 Problem Statement

In this problem, you need to use MPI to parallelize the following serial program (`http://www.cs.nctu.edu.tw/~ypyou/courses/PP-f19/assignments/HW3/conduction.c`). The program simluates heat conduction on an isolated 2D long strip and finds the minimum temperature until heat balance or reaching specified iterations, given random temperatures on the strip. There are three `int` arguments (L, which controls the length of the strip, *iteration* and *seed*, which seeds the random number generator) from command line arguments taken as inputs. For convenience, L is assumed to be divisible by the number of threads.

```
#include <stdio.h>
#include <stdlib.h>
#ifndef W
#define W 20                                  // Width
#endif
int main(int argc, char **argv) {
  int L = atoi(argv[1]);                      // Length
  int iteration = atoi(argv[2]);              // Iteration
  srand(atoi(argv[3]));                       // Seed
  float d = (float) random() / RAND_MAX * 0.2; // Diffusivity
  int *temp = malloc(L*W*sizeof(int));        // Current temperature
  int *next = malloc(L*W*sizeof(int));        // Next time step

  for (int i = 0; i < L; i++) {
    for (int j = 0; j < W; j++) {
      temp[i*W+j] = random()>>3;
    }
  }
  int count = 0, balance = 0;
  while (iteration--) {  // Compute with up, left, right, down points
    balance = 1;
    count++;
    for (int i = 0; i < L; i++) {
      for (int j = 0; j < W; j++) {
        float t = temp[i*W+j] / d;
        t += temp[i*W+j] * -4;
        t += temp[(i - 1 < 0 ? 0 : i - 1) * W + j];
        t += temp[(i + 1 >= L ? i : i + 1)*W+j];
        t += temp[i*W+(j - 1 < 0 ? 0 : j - 1)];
        t += temp[i*W+(j + 1 >= W ? j : j + 1)];
        t *= d;
        next[i*W+j] = t ;
        if (next[i*W+j] != temp[i*W+j]) {
          balance = 0;
```

```
      }
    }
  }
  if (balance) {
    break;
  }
  int *tmp = temp;
  temp = next;
  next = tmp;
 }
 int min = temp[0];
 for (int i = 0; i < L; i++) {
  for (int j = 0; j < W; j++) {
    if (temp[i*W+j] < min) {
      min = temp[i*W+j];
    }
  }
 }
 printf("Size: %d*%d, Iteration: %d, Min Temp: %d\n", L, W, count, min);
 return 0;
}
```

Hint: Divide array into several pieces and exchange minimum data to lower the overhead.

2 Requirement

- Your submitted solution contains one source file: `conduction.c` (or `conduction.cpp`).
- Your program takes two command-line arguments.
- You should not modify the output format.

3 Developing and Execution Environment of MPI Programs

3.1 Using the NCTU CS virtual cluster

3.1.1 Login information

Your program will run on these four machines/nodes which have NFS and NIS services that make MPI easy to use.

IP	Port	User Name	Password
140.113.215.195	37031-37034	`<student ID>`	`<student ID>`

3.1.2 SSH from one machine to the others

Handy hostnames corresponding to their private IP are provided in `/etc/hosts`.

Hostname	Original Port
pp01	37031
pp02	37032
pp03	37033
pp04	37034

For example, use the command below to log in to pp2 with the same user name.

```
$ ssh pp2
```

3.1.3 Executing jobs on other machines without entering a password

To make `mpi` work properly, you need to be able to execute jobs on remote nodes without typing a password. You will need to generate an ssh key by yourself.

You can also google "ssh passphrase" for details.

```
$ mkdir -p ~/.ssh
$ ssh-keygen -t rsa
# Then you will be prompted to enter some infomation, you can leave them empty for
    convenience.
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
# Thanks to NFS, you only need to do it once.
```

Then, log in to all machines at least one time to make sure the configuration works and also record them in `.ssh/known_hosts` file.

```
# Log in without entering password
$ ssh pp01
$ ssh pp02
$ ssh pp03
$ ssh pp04
```

3.2 MPI usage

MPI is installed in `/home/PP-f19/MPI`. Check with the command below.

```
$ /home/PP-f19/MPI/bin/mpiexec --version
```

3.2.1 Recording machines with host file

An MPI host file records all available machines with its ability. Attribute `slots` indicates at most what number of threads on that machine will be used by MPI.

```
// hostfile
pp01 slots=4
pp02 slots=4
pp03 slots=4
pp04 slots=4
```

3.2.2 Compiling and running

You may get an MPI hello world program from `https://reurl.cc/ekGDW` and then compile and run the program on the four machines to make sure that MPI works well.

```
$ ls
hostfile mpi_hello_world.c
$ /home/PP-f19/MPI/bin/mpicc mpi_hello_world.c -o mpi_hello_world
$ /home/PP-f19/MPI/bin/mpiexec -npernode 1 --hostfile hostfile mpi_hello_world
Hello world from processor ec037-031, rank 0 out of 4 processors
Hello world from processor ec037-032, rank 1 out of 4 processors
Hello world from processor ec037-033, rank 2 out of 4 processors
Hello world from processor ec037-034, rank 3 out of 4 processors
```

Then, start to write your own program.

4 Submission

Please rename your `conduction.c` to `<your-student-id>.c` and upload it to e-Campus system by the due date.

Due Date: 23:55, November 29, 2019

5 References

- https://computing.llnl.gov/tutorials/mpi/