Assignment 4, due November 29

- LF In end form
- Check Ass. 3 deductions

1 Synopsis

Dirichlet Problem

Please read the document, ***General Requirements Programming Assignments***, posted on the course website and make sure you follow them when you do this assignment. This is a relatively easy assignment just to give you a bit more practice with MPI. In the lecture notes, I provided a sequential algorithm and code to use random walks to solve the problem of finding the steady state heat distribution for a two-dimensional square plate, given specific initial conditions on the boundaries. In this assignment, you are to write a parallel program to solve this problem more generally.

2 Program Invocation, Input, and Output

In this project, you are to write an MPI program that finds the steady state temperature distribution of a two dimensional plate with heat applied to its boundaries. It expects a single command line argument, which is the pathname of a text file that consists of six numbers in the following order, separated by white space:

- an integer representing the number of rows in the grid
- an integer representing the number of columns in the grid
- a double precision float representing the temperature in Celsius on the north edge
- a double precision float representing the temperature in Celsius on the east edge
- a double precision float representing the temperature in Celsius on the south edge
- a double precision float representing the temperature in Celsius on the west edge

Therefore, if the program is named `steady_state`, and the file `plate` is in the proper form, it would be used as follows

```
steady_state plate
```

The program will output a single number, which is the temperature in steady state of the point in the middle of the plate. If the plate has M rows and N columns, then this is the point at coordinates $(\lfloor (N-1)/2 \rfloor, \lfloor (M-1)/2 \rfloor)$ assuming a coordinate system that is zero-based. The number should be displayed with 2 decimal digits to the right of the decimal point. There should no other output.

precission

The steady state of the plate is defined as the state in which the maximum change in the estimated temperature at every point between successive iterations is under some small value called the ***convergence threshold***. This implies that after every iteration, the program must calculate the difference in the estimates at every point and decide if steady state is reached.

↳ In every Process, then send to root 1 if < tolerance, 0 if not below tolerance

2.1 Error Handling

If the argument is missing or if the file can not be opened for reading, the program should print an error message on ***standard error*** that includes how to use it correctly, and it should exit. If during processing the program results in an error such as being unable to allocate memory or other programmatic failures, it should print a message on ***standard error*** that it failed and it should clean up all MPI processes and exit. The standard error stream in C can be written to using `fprintf`.

Check grade for assign. #3

3 Program Implementation Requirements

- Only the root process can perform input and output and error handling. It is an error if any other process does so.
- The program must produce correct results regardless of the size of the plate, provided that there is memory for it.
- The program must work correctly regardless of how many tasks are run.
- The convergence threshold for determining the steady state must be no larger than 0.05. Just use 0.05
- The program must be self-contained. If it uses any of the code I have provided, then copy that code into a single source code file.
- The program must be documented and written to comply with the requirements stated in the ***General Requirements Programming Assignments*** referred to above
- The program is supposed to be immune to incorrect usage. This means that if the first argument is anything other than a file in the correct format that can be read by the program, it should detect this and exit with an error message.
- The program must run correctly on any `cslab` host.

4 Advice

fake random numbers - eg same random numbers

The program should be tested initially with repeatable random number sequences. You can do this by seeding the random number generator with constants. This way you can run the program repeatedly to find bugs.

Make sure that you run the program with many processes to see the effect once you have debugged it.

The problem you must solve is how to distribute the work evenly. Use the decision tree to decide.

5 Program Grading Rubric

The program will be graded based on the following rubric out of 100 points:

- The program must compile and run on any `cslab` host. If it does not compile and link on any `cslab` host, it loses 80 points.
- **Correctness** (70 points)
  - The program should do exactly what is described above. Incorrect output, incorrectly formatted output, missing output, or output containing other characters are all errors.
  - It must process plates of arbitrary size.
  - The program should produce the same output no matter how many processes it is given, except for the elapsed time.
  - It should handle errors correctly.
- **Performance** (10 points)

  The program should be as efficient as possible. There are ways to solve this problem that are more efficient than others. When run with successively greater numbers of processes, the elapsed time should decrease, except that if the number gets too large (larger than the available processors generally), it will start to increase again. Check that this behavior occurs on average. On any one run, it may not be exactly like this, but over many runs it should behave like this.

- **Compliance with the Programming Rules** (20 points)

  Are all of the rules stated in that document observed? Programs that violate them will lose points accordingly.

6 Submitting the Homework

In these instructions, assume your program file is named `myprog.c`. ***The project must be submitted by November 29 at 19:00.***

Monday ——— 7pm

1. You will use the `submithwk_cs49365` command to submit this assignment. **To submit your file, type the command**[1]

   ```
   $ /data/biocs/b/student.accounts/cs493.65/bin/submithwk_cs49365  -t 4  myprog.c
   ```

   The program will copy your `program` into the directory

   ```
   /data/biocs/b/student.accounts/cs493.65/hwks/hwk4/
   ```

   and if it is successful, it will name it `hwk4_username.c` and display the message, "`File hwk4_`*`username`*`.c successfully submitted.`" where `username` is your actual username.

   You will not be able to read this file, nor will anyone else except for me. But you can double-check that the command succeeded by typing the command

   ```
   ls -l /data/biocs/b/student.accounts/cs493.65/hwks/hwk4
   ```

   and making sure you see a non-empty file named `hwk4_`*`username`*`.c`.

2. ***You can do the preceding step as many times as you want. Newer versions of the file will overwrite older ones.***

[1] If you have modified your PATH variable to include the directory `/data/biocs/b/student.accounts/cs493.65/bin`, then you can just type `submithwk_cs49365`.