CSCI 3366 (Parallel and Distributed Programming), Fall 2021
Homework 1

Credit: 10 points.

1 Reading

Be sure you have read, or at least skimmed, the first few sections of the updated Appendix A (before writing your OpenMP program) and the first few sections of the updated Appendix B (before writing your MPI program).

2 Overview

First a general comment: For this assignment, please *do not* discuss the problem or possible solutions with each other or anyone else. I want you to discover any potential pitfalls yourselves!

In class we briefly discussed approximating the value of π by "throwing darts" at a square of side 2 enclosing a circle of radius 1, counting how many darts fall within the circle, and then dividing that by the total number of darts to get the ratio between the area of the circle (π) and the area of the square (4). For this assignment, your mission is to write, for two different programming environments (OpenMP and MPI), a parallel program that performs this calculation. (The next assignment will add additional programming environments.)

3 Details

3.1 Sequential starter programs

To get you started, I have written a sequential program in C that performs the desired calculation and prints appropriate results:

- C program: monte-carlo-pi.c. Also requires timer.h.

Start by downloading this code, compiling it, and running it a few times to get a sense of what inputs you need to get a good approximation of π. (Note that you will need the flag `-std=c99` or `-std=c11` to compile the C code.)

3.2 Parallel programs

(5 points)

The programs you write (one each using OpenMP and MPI for this assignment, additional environments in the next assignment) should accept the same command-line input and produce the same output as my sequential program, except that:

- The OpenMP program should get the number of threads to use from environment variable `OMP_NUM_THREADS`.
- The MPI Program should get the number of processes in the usual way MPI programs do (parameter to `mpirun`).

- Rather than printing "sequential C program results",
  - the OpenMP program should print "parallel OpenMP program results with N threads" (where N is replaced with the number of threads);
  - the MPI program should print "parallel MPI program results with N processes" (where N is replaced with the number of processes); and

  (You can make other changes to the output format if you really want to, but be sure your code prints the same information mine does.)

You can also make any changes you like to how the programs work internally.

It's up to you how you choose to parallelize the sequential code, but note that in many respects the calculation here strongly resembles the one in the numerical integration example, so the approaches we used for that example might work well here too. The only thing that's tricky is deciding what to do about the random numbers (should all the processes/threads generate the same sequence of random numbers? should they generate different ones? if so, how?). For this assignment, I want you to make your best guess about what would be reasonable, implement that, and see how it works. After everyone has turned something in, we'll discuss in class your results and possible improvements.

You can make my grading job a bit easier by using the following names for your programs:

- `monte-carlo-pi-openmp.c` for the OpenMP program.
- `monte-carlo-pi-mpi.c` for the MPI program.

3.3 Performance experiments

After you get each program working, you should try running it repeatedly, varying the number of threads or processes, to see whether more threads/processes really do help. Good machines to use for this purpose may be `dione` for OpenMP (more processing elements than other machines) and the `pandora` cluster (names `Pandora01` through `Pandora08`) for MPI (Linux-only so shouldn't be rebooted out from under you).

3.4 Discussion of results

(5 points)

In addition to turning in your source code, briefly answer the following questions about each of your programs:

- Does the program seem to generally give better results as the number of samples increases? Can you say anything interesting or useful about how the seed affects the quality of the results?
- For the same number of samples and seed, does the program give the same results no matter how many processes or threads you use? If not, why do you think it doesn't?

- Does parallelism seem to help the program's performance? You will probably need a fairly large number of samples to observe any benefit, but up to a point more processes or threads should give faster execution. Is this true for your program? Support this with some observations (output of your program showing number of samples, number of processes/threads, etc., plus the name(s) of the machine(s) used). If it's not true, what do you think is going wrong?

Keep in mind that *this assignment is a first pass at producing good programs for this problem.* Your programs should compile and produce output that's more or less reasonable, but you will have another chance (in Homework 2) to produce something that really works well, so it's okay this time if the output seems slightly off, or the performance is disappointing.

3.5 Hints and tips

- Feel free to borrow code from any of the sample programs linked from the course sample programs page. This page also contains links to my writeups about compiling and running programs on the lab machines. The course "useful links" page has pointers to documentation on OpenMP and MPI.

- You can develop your programs on any system that provides the needed functionality, but I will test them on the department's Linux classroom/lab machines, so you should probably make sure they work in that environment before turning them in.

4 What to turn in and how

Submit your program source code and discussion of results by putting them in the "turn-in" folder I set up for each of you on Google Drive (it should be in your "shared with me" with a name that starts with CSCI3366), specifically in the HW01 subfolder.

(Yes, this is different from how I asked students to submit work in previous semesters. I figure try it out, and if it goes badly we can go back to e-mail, though that's not without problems either.)

5 Essay and pledge

Include with your assignment the following information.

For programming assignments, please put it a separate file. (I strongly prefer plain text, but if you insist you can put it in a PDF — just no word-processor documents or Google Drive links please). For written assignments, please put it in your main document.

5.1 Pledge

This should include the Honor Code pledge, or just the word "pledged", *plus* at least one of the following about collaboration and help (as many as apply). Text *in italics* is explanatory or something for you to fill in; you don't need to repeat it!

- I did not get outside help *aside from course materials, including starter code, readings, sample programs, the instructor.*

- I worked with *names of other students* on this assignment.

- I got help with this assignment from *source of help — ACM tutoring, another student in the course, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

5.2 Essay

This should be a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what if anything you think you learned from the assignment, and what if anything you found interesting, difficult, or otherwise noteworthy.