HPC Final Project

Due: Tue Apr 30 at 1pm (start of final exam period)

For your final project, you will create a program that runs on the CPU in serial, on the CPU in parallel (using OpenMP), and on the GPU in parallel. The exact program you make will be up to your group, submitted to me via Canvas by Mon Apr 15 and approved on a first-come-first-serve basis. You must present your work to the class along with the results and performance analysis.

Program

You must create 3 versions of your program: serial, parallelized with OpenMP, and parallelized with CUDA. You must select the program you want to make, choosing either one of the following examples or coming up with an idea of your own. You must submiit on Canvas by the start of class Wed Apr 10.

- convolutional image filter (demonstrate with several filters of different sizes)
- non-linear image filter (demonstrate with several filters of different sizes)
- histogram equalization / contrast enhancement
- prefix sum and scan (serial version is just cumsum and similar)
- conway's game of life
- mandelbrot, julia, *and* multibrot sets (or some other similar sets)
- FDTD electromagnetic simulator (physics)
- Batcher’s odd-even and bitonic mergesorter sorting (serial version should be shellsort and mergesort)
- k-means clustering (machine learning)
- edit distance (for genomic sequence alignment)
- binomial lattice model (for option pricing)
- raytracer (difficult)

Code Explanation

Most of these problems have been solved already and many times you can find solutions out there or have Copilot generate some decent solutions. In either case you will need to at least tweak them to make them work with how we are doing things in class. You also must reference outside sources (i.e. webpages or Copilot) for large portions used. You must also thoroughly explain every feature that was not covered in class. You must also be prepared to answer questions about any and all lines of code in detail.

Analysis

Run your programs across various sized data set sizes, “quality” of datasets (for example, same size but ones that cause more or less problems), number of cores, and blocks/grids. You should have numerous data points to get nice curves, see why the block sizes are (approximately) optimal, see crossover points, and be able to analyze speedup and efficiency. You should use Expanse to achieve all final timing data. But test on your own machine and mucluster first, only move to Expanse once everything is working.

Presentation

Describe/demonstrate the problem, explain why parallelization is a good choice for the problem, go through all of your analysis to show how well your program did and why certain block/grid sizes were chosen, and then talk about new features learned about. Submit slides to Canvas.

Submission

Your submitted repo must include all necessary files to compile along with sample data. The main 3 files must include instructions for compiling and running at the top of each one. The rest of the code must have high code quality (remember that I will have 0 familitiary with your code, you must make sure I can quickly read the code).

Rubric

15 pts - serial program
15 pts - OpenMP program
15 pts - CUDA program
20 pts - analysis and its presentation
20 pts - code explanation and its presentation
10 pts - code quality
5 pts - other presentation parts