MPCS HPC WINTER 2021 – HOMEWORK 2

General Instructions

Due date is 2021-01-29 at 5:00pm CST.

Your solution must be pushed to your GitHub Classroom repo by the due date. **Don't forget to click on Submit for the assignment!** Alternately, if you're experiencing technical issues with GitHub, you may send a compressed file (.zip or .tar.gz) of your solution to rahaman@cs.uchicago.edu by the due date.

See the rubric on Canvas for point totals. If a solution does not compile, you will be penalized an additional 20% off the total points possible for the respective problem.

PROBLEM 1 – ROOFLINE WITH MATRIX-VECTOR MULTIPLICATION

OVERVIEW

In this problem, you will do the following:

1. Construct a roofline model of one of your computers
2. Calculate the arithmetic intensity for matrix-vector multiplication from Homework 1.
3. Compare your results from Homework 1 to your roofline

In this problem, the only deliverable is a report (`problem1/report.pdf` or another text file format).

TASKS

1. The Roofline

In Lecture 2A ("The Roofline Model"), in the section "How to Construct a Roofline", we discussed how to use some simple benchmarks to construct a roofline. Your task is to follow that process and construct a roofline:

- On the same computer that you used for Homework 1, Problem 1, download STREAM to measure peak bandwidth. There are several tests; simply take the maximum bandwidth out of all the tests.
- On the same computer, use either Intel MKL or AMD BLIS to measure peak FLOP/s. There are several tests; simply take the maximum out of all the tests.
- Use the bandwidth and FLOP/s to plot a roofline model.

Give your measurements and show a graph of the roofline model in the report.

2. Arithmetic Intensity

Calculate the arithmetic intensity of matrix-vector multiplication for 32-bit floats (the same data type you used in Homework 1).

Give the arithmetic intensity and briefly explain your reasoning in the report.

3. Comparing Results to Roofline

Take the execution time of your matrix-vector multiplication code that you measured in Homework 1 (alternately, measure the posted reference solution). Use it to measure the FLOP/s of your code.

Plot it the FLOP/s of your code on your roofline model. This is a single (x, y) point where x is the arithmetic intensity that you derived in step 2; and y is the FLOP/s that measured. Briefly discuss the disparity.

Problem 2 – Weak and Strong Scaling with Particle Simulation

Overview

In this problem, you will parallelize your particle simulation from Homework 1 with OpenMP. Then you will assess the strong and weak scaling of your implementation.

Tasks

1. OpenMP Implementation

Take either your struct-of-array or array-of-struct implementation (your choice) from Homework 1. Alternately, you may take the reference solution. **Parallelize the particle-tracking portion with OpenMP.**

Put all the source files you need in the `problem2/` subdirectory, including a `CMakeLists.txt`.

2. Strong Scaling

Choose a fixed number of particles (at least 512) and a fixed number of timesteps (at least 512). Gather timings with varying numbers of threads, from 1 thread up to the maximum number of cores on the given computer. So if the computer has 4 cores, you'll execute the program with parameters such as:

Threads	Particles
1	512
2	512
3	512
4	512

In the report (`problem2/report.pdf` or another text file), show a graph of these results. The x axis is the number of threads; and the y axis is **parallel efficiency**.

3. Weak Scaling

For a fixed number of timesteps (at least 512) and a fixed number of particles *per thread* (at least 256 per thread), measure times with a range threads, from 1 up to the maximum number of cores. For example:

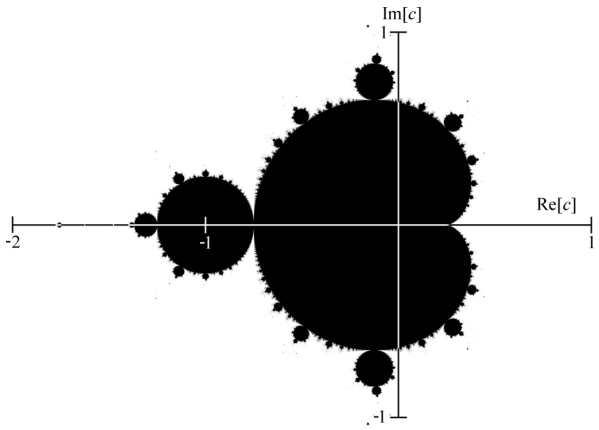
Threads	Particles
1	256
2	512
3	768
4	1024

In the report, show a graph of these results. The x axis is the number of threads; and the y is **execution time**.

PROBLEM 3 – MANDELBROT SET

OVERVIEW

In this problem, you will generate an image of a classic fractal called the Mandelbrot Set. This is a set of complex numbers c for which the iteration $z_{n+1} = z_n^2 + c$ increases without bounds. To generate the image, you take a take the complex number plane (where x is the real component; and y is the imaginary component); evaluate the iteration at each point in the plane; and color the point based on whether the Mandelbrot iteration diverges (within some threshold). For example, in the image on the right, the points for which the iteration diverges are colored in black.



The pseudocode is as follows. Note that i is the imaginary unit; hence, c and z are imaginary numbers.

```
Allocate A as an m × n matrix representing the pixels of the image
Given a threshold T = 1000
Δ_x = 3.0 / m
Δ_y = 2.0 / n
for x in [0, m):
    for y in [0, n):
        c = (−2.0 + x · Δ_x) + (−1.0 + y · Δ_y) · i
        z = 0.0 + 0.0 · i
        in_set = 1
        for t in [0, T):
            z = z^2 + c
            if |z| < 4:
                in_set = 0
                break
        A_{x,y} = in_set
Write A to file as a bitmap
```

TASKS

Using OpenMP, write a parallel program that generates a Mandelbrot set.

- The program should take 2 argument for m and n, the dimensions of the image. All other values can be hardcoded.
- The program should report the wall time for the loop shown above.
- The program should also generate a bitmap image of the set.
- Put all the source files you need in the subdirectory `problem03/`, including the `CMakeLists.txt`

In the report, present a graph of the strong scaling (as in problem 2) using a fixed, large image size and an increasing number of threads. The x axis should be the number of threads; and the y axis should be parallel efficiency.