Graphics Processing Units (GPUs): Architecture and Programming
Programming Assignment # 1

In this lab you will write a simple program that processes vectors in CUDA.

The user enters a number n (n is a positive integer). Three arrays, a, b, and c of n elements each are dynamically allocated and initialized with random floating points numbers.
Then the operation c[i] += a[i] * b[i] will be executed both sequentially on the host and on the device (the GPU). The time taken by each are printed on the screen.

We are providing you with a file vectors.cu that reads the input form the user, does the sequential operation, measure the time for both the sequential and GPU parts, and print them on the screen. The only thing missing is the GPU part.
You need to do the following:

1. Read very carefully the code in vectors.cu
2. Pay close attention to the comments in the code.
3. Implement the parts that have a comment with “TODO:” at its start.
4. Read, from the course website, how to access cuda clusters.
5. compile with nvcc -o vectorprog vectors.cu -lm

Note:

- If you compile the initial program vectors.cu without any change, it will compile but you will get three warnings that *ad*, *bd*, and *cd* are declared but never used. These are the pointers to the arrays that you will allocate on the device and use in your kernel.
- The last part in the file vectors.cu compares the sequential and GPU versions in terms of correctness. But since GPU floating point precision is better than CPU, we use the condition **fabsf(temp[i] - c[i]) >= 0.009** that compares the numbers up to the second digit in the floating point only.

Experiments

You will do the following experiments and report your findings:

<u>Experiment 1:</u>

- Fix n to be 1,000,000 (one million)
- Report in a bar graph the speedup (or slowdown), over the CPU version for the following organizations (Note that you may need to adjust your code to make each thread do different amount of work):
    - 4 blocks and 500 threads per block
    - 8 blocks and 500 threads per block
    - 16 blocks and 500 blocks per block
    - 4 blocks and 250 threads per block

    - 8 blocks and 250 threads per block
    - 16 blocks and 250 blocks per block
- The y-axis is the speedup (CPU time / GPU time) while the x-axis shows these configurations, for example, the first configuration can be written as (4, 500) as a point on x-axis.
- Write few lines explaining:
    - What is the behavior you observed in the graph?
    - Why do you think we got this behvaior?

Experiment 2:

- Fix the configuration to 8 blocks and 500 threads each.
- Report, in a bar graph, the speedup (or slowdown) as n takes the values: 100, 1000, 10000, 100000, one million, 10 millions, and 100 millions.
- The y-axis is the speedup (CPU time / GPU time) while the x-axis are shows the different values of n.
- Note: If you run out of memory (i.e. cannot allocate arrays on system memory or global memory) for a value of n, please write so in your report and report the results of the other values of n.
- Write few lines explaining:
    - What is the behavior you observed in the graph?
    - Why do you think we got this behvaior?

What to submit?

Add the source code vectors.cu as well as the pdf file that contains your results to a zip file named: netID.zip where netID is your own netID.

Important: The code you submit must have the version that creates 8 blocks with 500 threads each.

How to submit? Through the assignment sections of Brightspace.

Have Fun!