Embedded System Design

Assignment 2

This homework corresponds to

- Attendances: Week 4 and 5 (2% of total score)
- Homework 2: (10% of total score)

What to submit: Report and source codes

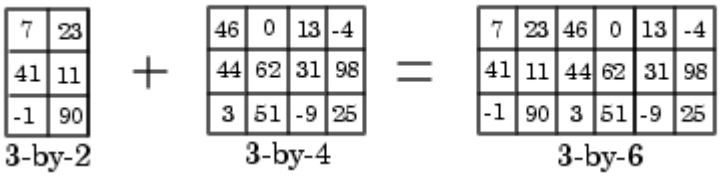
- There is no fixed report format, except for the hard page limit, which is 8 pages.
- Zip your report and source code into a single file (your_student_num.zip, ex: 2015123456.zip) and submit it.

Deadline: Refer to I-Campus

Please read the following and implement source codes accordingly. You will implement source codes for GPUs for i) optimized matrix concatenation and ii) finding a maximum value in a large vector.

1. Profiling and Optimizing Matrix Concatenation (5% of total score)

In the HW assignment 1, you implemented a CUDA code performing the matrix concatenation. Here is a reminder of HW1.



In the assignment of HW1, the sizes of two input matrices were as follows. All numbers in the matrices were integers (just use **`int`** type).

Input matrix 1: 512 by 1024 (# of rows: 512, # of columns: 1024)

Input matrix 2: 512 by 512 (# of rows: 512, # of columns: 512)

Also, you are studying how to profile CUDA codes and how to optimize your source codes. Therefore, this homework is a chance to take a simple review and a test what you are studying.

Based on what you have done in HW1, do the following.

1. Figure out inefficiencies of your CUDA code.
    - First, analyze your source code by hand and find out inefficiencies that cause performance degradation by hand. Do not worry if you find something inefficient inside your code. In HW1, there is no penalty even if you do not need to apply to any optimizations. Note that you did not study anything about optimization at that time. Now, it's time to do it!
    - Choose proper metrics and run `nvprof` with the chosen metrics to prove your theory.
2. Optimize your CUDA code **based on what you studied until week 6**.
    - Implement a new one that addresses the problem that you found in your implementation in HW1.
3. With your new code, measure the GPU execution time and profile the metrics that you picked. And then compare the experimental results to those in HW1.
    - Find the speedup of your new CUDA program over the CPU program.
    - Compare the speedup of your new CUDA program to what you implemented in HW1.

2. Finding a Maximum Value in a Large Vector (5% of total score)

Implement programs that finds the largest elements in a large vector. The vector contains 100,000,000 floating point-type (`float`) elements. You may initialize all the elements in the vector as you want. Do the following.

1. Implement a single-threaded program for CPU.

2. Implement a CUDA program, aiming at minimizing branch divergence and optimizing memory accesses. Include analysis of your own regarding
    A. Are there any possibilities to have branch divergence? How do you resolve it?
    B. Are there any optimizations for better memory accesses? How do you apply them?
3. Measure the execution time with the CPU code and the GPU code. Find the speedup of your CUDA code over the code for CPU.