Department of Electrical and Computer Engineering
Queen's University

ELEC 374, Digital Systems Engineering

Machine Problem 2

Total Mark: 3%

Posted: Feb 16, 2023
Actual Deadline: **Mar 31, 2023** at 8:30am
3 days Grace Period: **Apr 3, 2023** at 8:30am
After grace period: **25% late penalty per day**

Matrix Addition

The objective of this machine problem is to implement matrix addition routines, where each thread produces one or more output matrix elements. It is also to get you familiar with using the CUDA API and the associated setup code.

A matrix addition takes two input matrices A and B and produces an output matrix C. Each element of the output matrix C is the sum of the corresponding elements of the input matrices A and B. For simplicity, we will only handle square matrices of which the elements are single-precision floating-point numbers. Write a matrix addition kernel and the host function that can be called with four parameters: pointer to the output matrix C, pointer to the first input matrix A, pointer to the second input matrix B, and the number of elements in each dimension. After the device matrix addition is invoked, it will compute the correct output matrix using the CPU and compare that solution with the device-computed solution. If it matches (within a certain tolerance), it will display "Test PASSED" on the screen before exiting.

Follow the instructions below:

1. Write the host code by allocating memory for the input and output matrices, transferring input data to the device, launching the kernel, transferring the output data to host, and freeing the device memory for the input and output data. Leave the execution configuration parameters open for this step.
2. Write a kernel that has each thread producing one output matrix element. Fill in the execution configuration parameters for the design considering 16x16 thread blocks.
3. Write a kernel that has each thread producing one output matrix row. Fill in the execution configuration parameters for the design considering 16 threads per block.
4. Write a kernel that has each thread producing one output matrix column. Fill in the execution configuration parameters for the design considering 16 threads per block.
5. Analyze the pros and cons of each kernel design above.

Create randomly initialized matrices A and B, and experiment with different matrix sizes (125 x 125, 250 x 250, 500 x 500, 1000 x 1000, and 2000 x 2000). In each case, measure the time for the kernel

execution, and using a graph/table compare the GPU performance against the CPU performance. Analyze your results.

Note that in this machine problem we are not concerned about the data transfer cost, which is in fact a real concern in GPU computing and that is something we will consider in Machine Problem 3. Also, do not consider memory allocation/free time on the device in your timing.

For this and other machine problems, you may consult the Lecture Slides on GPU Architectures and Computing and the GPU Server and CUDA Environment Tutorial on the course website. You may also consult the NVIDIA CUDA C Programming Guide. As for the timing, you should create CUDA events for the start and stop times, begin and stop recording, find the elapsed time, and destroy the events (*cudaEventCreate, cudaEventRecord*, *cudaEventElapsedTime*, *cudaEventSynchronize*, etc.). For timing measurements, make sure you repeat your experiments a sufficient number of times, remove any outliers due to the shared environment, and report the average time.

Submission Instructions:

This machine problem is to be done individually. Students must run through the assignment by themselves, include their own solutions and analyses, and turn in their own reports. Submit a **single zip file**, "MP2_YourLastName.zip", containing your code (e.g., MP2.cu, with your name and student ID on top of the code), and a report in pdf, MP2_YourLastName.pdf, where you present your work by including your CUDA code and the Visual Studio output screenshots, analyses of your results, and discussions of any outstanding issues. Students must abide by the academic integrity expectations, and include in their report a statement that says:

- *"I do hereby verify that this machine problem submission is my own work and contains my own original ideas, concepts, and designs. No portion of this report or code has been copied in whole or in part from another source, with the possible exception of properly referenced material.".*

GPU Servers:

Four GPU (Tesla C2075) servers are available for the machine problems. The servers run Windows server operating system and provide the Visual Studio 2015 IDE for building CUDA projects. For remote connection to the GPU servers and information on CUDA and its environment, please consult the GPU Server and CUDA Environment Tutorial. The servers use a load balancer to balance the workload on the GPU servers, and it is possible that you may log in to a different GPU server each time. Therefore, it is very important to back up your files.