Department of Electrical and Computer Engineering
Queen's University

ELEC 374, Digital Systems Engineering

Machine Problem 4

Total Mark: 3%, plus up to 2% bonus mark

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**

Tiled Matrix Multiplication

The objective of this machine problem is to get you familiar with using shared memory to write optimized kernel algorithms by implementing a "tiled" version of matrix multiplication.

A matrix multiplication takes two input matrices M and N and produces an output matrix P. For simplicity, we will only handle square matrices, of which the elements are single-precision floating-point numbers. Write a shared memory based (tiled) matrix multiplication kernel and the host function that can be called with four parameters: pointer to the output matrix P, pointer to the first input matrix M, pointer to the second input matrix N, and the number of elements in each dimension. After the device matrix multiplication 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.

Create randomly initialized matrices M and N, and experiment with different matrix sizes (125 x 125, 250 x 250, 500 x 500, 1000 x 1000, and 2000 x 2000). Write the host code by allocating memory for the input and output matrices, transferring input data to device, launching the kernel, transferring the output data to host, and freeing the device memory for the input and output data. Write a kernel that has each thread producing one output matrix element. Write the execution configuration parameters in your host code considering different TILE_WIDTH of 2, 5, 10, 20, and 25. Do not consider the data transfer time and memory allocation/free time on the device in your timing. Plot the results and discuss your findings. Are the results faster than the baseline matrix multiplication in Machine Problem 3, and why?

Answer the following questions:

1. In your kernel implementation, how many threads can be simultaneously scheduled on your CUDA device, which contains 14 streaming multiprocessors?
2. Can you find the resource usage of your kernel, including the number of registers, shared memory size, number of blocks per streaming multiprocessor, and maximum total threads simultaneously scheduled/executing?

Bonus Mark: Up to 2 bonus marks will be considered if you study a revised tiled matrix multiplication kernel with boundary checks that would remove our two simplifying assumptions: 1) that matrices are square matrices, and 2) that the dimensions of matrices is assumed to be a multiple of the dimensions of the tiles. Experiment with different matrix sizes (350 x 400 matrix M with 400 x 500 matrix N; and 1900 x 1600 matrix M with 1600 x 1300 matrix N) and the tile size of 8 x 15. Do not consider the data transfer time and memory allocation/free time on the device in your timing. Plot the results and discuss your findings.

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**, "MP4_YourLastName.zip", containing your code (e.g., MP4.cu, with your name and student ID on top of the code), and a report in pdf, MP4_YourLastName.pdf, where you present your work by including your CUDA code and the Visual Studio output screenshots, analyses of your results, answers to the questions, 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.