Assignment #2

CS 398 GPGPU Programming, Fall 2020

Due Date:	As specified on the moodle
Topics covered:	Tiled Matrix Computation, Shared Memory
Deliverables:	The submitted project file including source code and the report `A2Report.pdf`. These files should be put in a folder and subsequently zipped according to the stipulations set out in the course syllabus.
Objectives:	Learn how to use shared memory to write more efficient kernel functions.

Programming Statement

This is the second CUDA C programming assignment. Students are expected to finish the programming for a given computation problem in Assignment 1 and the learnt Tiled Matrix Multiplication. This requires the measurements done on Lab PC with NVIDIA GeForce GTX 1060 6GB. Please note that there are 48 KB shared memory per SM for our GPU device.

Problem 1 Statement

This problem statement is the same as described in Assignment 1 but you are required to use shared memory to store the data for tiled computation. Assume $n < 30,000$. You are required to submit the file kernel.cu that defines a kernel function

```
__global__ void heatDistrCalcShm(float *in, float *out, uint nRowPoints);
```

Please refer to the updated Assignment 1 Reference Code on the moodle.

You are required to turn in your homework that is compilable under the Windows environment using Visual Studio.

Problem 2 Statement

In the second problem of this assignment you are required to implement a floating point matrix multiplication program using CUDA C/C++ and study the effect of block size and thread size on performance. You can consult the sample CUDA code and lecture notes provided in class.

Your submission should have two mode of operations: CPU mode and GPU mode. In the CPU mode with three numerical arguments (l, m, n), the program will create two randomly initialized matrices A and B. The size of A is $l \times m$, and that of B is $m \times n$. Another matrix C is created and calculate $C = A \times B$. C should be calculated both on the CPU and the GPU. If the two solutions match (with a certain tolerance e.g. relative error small than e.g $\epsilon = 10^{-6}$)[1] , it will print out "Test PASSED". In the GPU mode with

[1]For error tolerance, you can use both absolute error and relative error to test. E.g. if X is obtained from the CPU version and Y is from the GPU version, then the test is considered as failure if $(|X - Y| > \epsilon)$ && $(|X - Y| > \epsilon * |Y|)$ is true. You may use double precision to achieve lower error tolerance.

three strings as arguments, the program will read its inputs (two matrices) from the files provided by the first two arguments, and write its output (the resulting matrix) to the file provided in the third. The matrix file should be in ASCII text. The first line contains two integers (row and col) to define the matrix size. In all cases, your program should be able to handle very large matrices (e.g., tiling is needed).

You should begin the implementation of the program with just one thread block (e.g., grid = (1,1)). After the program is functionally correct, you are asked to explore different block size and its impact on performance. Note that the performance is only measurable when you have a large matrix (e.g. 2000 × 2000). In particular, for a given N × N matrix (e.g. N = 2,000), you should vary the block/tile size to 1×1, 4×4, 16×16, and 32×32 and report the performance difference on your hardware. It is expected that some of the configurations may not work on your particular hardware (due to hardware resource limitation). This is fine and should just be documented in your report.

You are required to turn in your homework that is compilable under the Windows environment using Visual Studio in our lab.

Grading Guideline

Your submission will be graded on the following parameters.

1. Problem 1: 20%
   (a) Functional Correctness: 0 marks if it produces incorrect result output for test inputs
   (b) Coding: Correct usage of thread id's in data idx computation for copying data into shared memory.
   (c) Performance: the latency close to that in Assignment 1 (e.g. about 0.35 s for n = 2,000 and 500 iterations)
2. Problem 2: 80%
   (a) Functional Correctness: 30%: Produces correct result output for test inputs.
   (b) Coding: 30%
      i. Correct usage of CUDA library calls and C extensions.
      ii. Correct usage of thread id's in matrix computation.
   (c) Report: 20% Describe the performance effect you have observed using different grid size. Performance: the latency for CPU version and GPU version should be less than 14 s and 0.04 s respectively if N = 2000.