3460:677 LAB 2: VECTOR ADDITION

Objective

The purpose of this lab is to introduce the student to the CUDA API by implementing vector addition in two ways. First, the student will implement vector addition by writing the GPU kernel code as well as the associated host code. Next, the student will implement vector addition using Thrust.

Prerequisites

Before starting this lab, make sure that:

- You have completed all of Online Module 2
- You have completed Lab 1 (Getting Started)

Local Setup Instructions

The most recent version of source code for this lab can be found on the class web page. As in Lab 1, download it to your Windows account on a lab machine, then copy it to your `tesla` account. Log into `tesla` and transfer the files there. (Note that this has been designed to work on and tested on `tesla`. You may do your work on other CUDA-capable equipment but you are on your own altering the seed files so that they execute.)

Overview

The **`Lab2`** directory contains five files:

- **`run.sh`** is a Linux batch file that compiles and executes a named CUDA program;
- **`vectorAdd.cu`** is the code template for the CUDA vector addition program;
- **`thrustVectorAdd.cu`** is the code template for the Thrust vector addition program; and
- **`input0.raw`** and **`input1.raw`** are input data files to test your program, with **`output.raw`** the correct answer used to check your work.

The code templates are provided as a starting point. The code handles the import and export of data files as well as the checking of the solution. You are expected to insert your code in the sections demarcated with `//@@`. Leave the other code unchanged. To compile and execute use the commands **`./run.sh vectorAdd`** and **`./run.sh thrustVectorAdd`**.

Instructions, Part 1

Edit the code in **`vectorAdd.cu`** to perform the following:

- Allocate device memory
- Copy host memory to device
- Initialize thread block and kernel grid dimensions
- Invoke CUDA kernel
- Copy results from device to host
- Free device memory
- Write the CUDA kernel

Instructions about where to place each part of the code is demarcated by the `//@@` comment lines.

Instructions, Part 2

Edit the code in **`thrustVectorAdd.cu`** to perform the following:

- Generate a `thrust::device_vector<float>` for host input arrays
- Copy host memory to device
- Invoke `thrust::transform()`
- Copy results from device to host

In all cases use `thrust` commands. Instructions about where to place each part of the code is demarcated by the `//@@` comment lines.

Important note: Because **`hostInput1`** and **`hostInput2`** are not declared as **`thrust::host_vector`**s you cannot just copy them to the device with an equals sign, as seen in the lecture slides. You must use the **`thrust::copy`** command:

```
thrust::copy(hostInput, hostInput + inputLength, deviceInput.begin());
```

Lab Report and Submission

When finished write a one-to-two page lab report (single-spaced, 1-inch margins, 12- to 14-point font size) summarizing your work and findings. In your report answer these questions:

(1) How many floating operations are being performed in each of your vector add kernels? EXPLAIN.
(2) How many global memory reads are being performed by each of your kernels? EXPLAIN.
(3) How many global memory writes are being performed by each of your kernels? EXPLAIN.
(4) Describe what possible optimizations can be implemented to your kernel in **`vectorAdd.cu`** to achieve a performance speedup.
(5) In what ways did Thrust make developing a functional vector addition code easier or harder?
(6) Name three applications of vector addition.

Submit the two edited `.cu` files and your lab report by the due date for grade.