3460:677 LAB 5: HISTOGRAM

Objective

The purpose of this lab is to implement an efficient histogramming algorithm for an input array of integers within a given range. Each integer will map into a single bin, so the values will range from 0 to (NUM_BINS - 1). The histogram bins will use unsigned 32-bit counters that must be saturated at 127 (i.e. no roll back to 0 allowed). The input length can be assumed to be less than 2^32. NUM_BINS is fixed at 4096 for this lab.

This can be split into two kernels: one that does a histogram without saturation, and a final kernel that cleans up the bins if they are too large. These two stages can also be combined into a single kernel but don't bother.

Prerequisites

Before starting this lab, make sure that you have completed all of the Module 7 lecture videos and materials.

Local Setup Instructions

The most recent version of source code for this lab can be found in a `.zip` file on the class web page. As before, download it to your Windows account on a lab machine, then copy it to your `tesla` account. Log into `tesla` and `unzip` 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 **`Lab5`** directory contains four files:

- **`run.sh`** is a Linux batch file that compiles and executes a named CUDA program;
- **`histogram.cu`** is the code template for the un-optimized dense matrix multiplication; and
- **`input.raw`** is the input data file 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 command **`./run.sh histogram`**.

Instructions

Edit the `main()` code in **`histogram.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
- deallocate device memory

Additionally complete the histogram kernels as described above. Instructions about where to place each part of the code is demarcated by the `//@@` comment lines.

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) Describe all optimizations you tried regardless of whether you committed to them or abandoned them and whether they improved or hurt performance.

(2) Were there any difficulties you had with completing the optimization correctly?

(3) Which optimizations gave the most benefit?

(4) For the histogram kernel, how many global memory reads are being performed by your kernel? Explain.

(5) For the histogram kernel, how many global memory writes are being performed by your kernel? Explain.

(6) For the histogram kernel, how many atomic operations are being performed by your kernel? Explain.

(7) For the histogram kernel, what contentions would you expect if every element in the array has the same value?

(8) For the histogram kernel, what contentions would you expect if every element in the input array has a random value?

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

Last updated 11.1.2016 by T. O'Neil, based on labs licensed by UIUC and NVIDIA (2016) under a Creative Commons Attribution-NonCommercial 4.0 license.