Fall 2021
ECE 5770: GPU Accelerated Computing
Prof. Shadi G. Alawneh
Deadline: Tue Nov 2 11:59 pm

Assignment 3

Histogram

Description

The purpose of this assignment is to implement an efficient histogramming algorithm (using privatization technique) 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 assignment. 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.

Instructions

Edit the code 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.

Local Setup Instructions

Requirements

-=-

libwb:

https://github.com/abduld/libwb

Compile instructions (Unix)

-=-

You will need libwb (a library by the authors of the teaching kit) for all assignments in this course.

The easiest way to get everything working would be to create a directory for class assignments, and place libwb in that directory alongside the individual assignment directories.

To build libwb:
cd [path_to_libwb]
mkdir build
cd build
cmake ..
make

To build Histogram:
cd [path_to_Histogram]
mkdir build
cd build
cmake ..
make

This should just work if the Histogram and libwb directories are in the same parent directory.

Testing
-=-
The "Dataset" directory contains some sample input / output data to test your implementation. To test a particular input, run (from the Hisotgram root directory):

./build/Histogram -i Dataset/##/input.raw -e Dataset/##/output.raw
-o output.raw -t integral_vector

This will produce some output, part of which should indicate whether or not your output matches the expected output.

To test all 6 examples at once, run (again from the Histogram root directory):

python test.py

Upon completion, this should tell you how many of the samples you produced the correct output for, and indicate which ones (if any) were incorrect.

Submission
-=-
You can simply submit your solution on Moodle. There is no need to include the dataset or any build files, just the Histogram.cu is fine!