Lab 5

CMPUT 398

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.

This lab will be submitted as one zipped file through eclass. Details for submission are at the end of the lab.

Summary the problem:

- There are N inputs values. All of them will be from 0 to (NUM_BINS – 1)

- Your task is build a histogram of the input values. The histogram has NUM_BINS bin, each of the bin correspond to a number.

- Finally, do a post-processing task: if a bin value is greater than 127, make it 127.

- You may need to syncthread between these two steps to make sure the program work correctly

Instructions

Edit the code where the TODOs are specified and 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

Local Setup Instructions

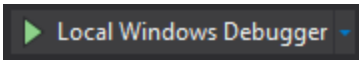
Steps:

1. Download “Lab5.zip”.

2. Unzip the file.
3. Open the Visual Studios Solution in Visual Studios 2013.
4. Build the project. Note the project has two configurations.
    a. Debug
    b. Submission

But make sure you have the "Submission" configuration selected when you finally submit.

5. Run the program by pressing the following button:



Make sure the "Debug" configuration is selected. Running the program in Visual Studios will run one of the tests located in "Dataset/Test".

Testing

To run all tests located in "Dataset/Test", first build the project with the "Submission" configuration selected. Make sure you see the "Submission" folder and the folder contains the executable.

To run the tests, click on "Testing_Script.bat". This will take a couple of seconds to run and the terminal should close when finished. The output is saved in "Marks.js", but to view the calculated grade open "Grade.html" in a browser. If you make changes and rerun the tests, then make sure you reload "Grade.html". You can double check with the timestamp at the top of the page.

The testing script only tests the correctness of your histogram implementation. You will be given bonus marks for an optimized implementation using shared memory. To get the additional marks you will need to use shared memory and write a report on the speedup of the shared memory version compare to the global memory version. That means you will have to do two versions to get 100%. See the Mark Breakdown for more information.

To show the speedup between the two versions, you must write a report that includes 2 screenshot of the NSIGHT CUDA summary for them**. The speed test must be ran on test number 6.** Copy and paste these screenshots into the report just like you did in the last labs, no other writing are needed. For this lab, as long as your optimized version runs faster than the normal version, you will get the other 20%. **If do the optimized version but do not include this report, you won't get the 20% mark.**

As a reference, you will be provided with an executable that you may want to compare the speed of your program with named "Lab5_reference.exe". **Do NOT submit this executable or you will fail all the test!**

Submission

For this submission, submit your best kernel (i.e. submit the optimized version if you done it, otherwise, submit the normal version).

After you build your project with the “Submission” configuration selected (see above) you will see a new folder called “Submission” in the project directory. If you open the folder you will see your executable. Put the report file (if you do the also do the shared memory version) to “report.pdf”. Finally zip the folder “Submission” and upload the zipped folder to eclass.

If you don’t see this folder or are missing then executable, then one of two things happened. First your build could have failed and you should check for any errors in your build log in Visual Studios. Also you might have had the “Debug” configuration selected (see above). Make sure the “Submission” configuration is selected.

Mark Breakdown

We will be running your implementation of the histogram against our own. In order to get the “Optimized” marks you must have similar or better performance than our implementation. To have similar performance to our implementation you will need to use shared memory in your implementation.

It is important that you **do not** add or remove any print statements or wb functions. We use them for part of your grading so if you make changes the marking script could fail.

Part	Breakdown (%)
Histogram	80 %
Optimized	20 %