Lab Assignment 2 – CSE 591, Spring 2015

Due: Thursday, May 14, 11:59pm

This lab will allow you to play around with all the material you have learned about so far -- hold nothing back. Experiment with:

- different numbers of blocks, warps, and threads
- different memory levels (global vs. shared) and management schemes (collaborative loading, loop unrolling, etc.)
- different algorithmic constructs

Use NVIDIA Nsight to optimize your programs. Upon completion of the assignment please submit the following, via blackboard:

- the complete software (all that is needed to build the executable: source code, project files, etc)
- an executable (.exe file) of your work
- a comprehensive report that illustrates with (1) narrative text, (2) code snippets, (3) program output, and (4) run time of all aspects of your work

This lab deals with the computation of histograms. The aim is to make each of the implementations run as fast as possible (ignore overhead due to GPU data upload/download). Your program should first allocate an array of N integers and then assign random values within the range 0 to 999 to them. Have your program vary N to be 2M, 8M, and 32M (M=2^{20}). Make sure that you use the same set of values when you compare your different implementations in the results section.

1. Implement the CPU base version

This will also serve as a reference application to confirm the accuracy of your GPU codes.

2. Implement two alternative GPU-based strategies

Optimize each of these two strategies as much as possible.

(a) Use atomic adds for the update of the bins. This will have limitations due to the possible serialization of the threads on collisions.

(b) Use a pre-binning scheme similar to that used in parallel Sample Sort. This will avoid the atomic adds and its disadvantages. You would first distribute the numbers into a set of bins, each spanning an interval of numbers. Then you would compute the histograms for each of these bins and finally append them for the full histogram.

3. Further optimization methods

You have learned of other optimization methods, such as instruction-level parallelism, loop unrolling, and so on. Will any of these help speed up your program more?

4. Report results on comparative charts

Take all the results you obtained and compile charts/plots that demonstrate the performance measured in the programs you developed. Plot speed over size for the various implementations you did so they can be easily compared. In addition, create plots for speedup = time_CPU / time_GPU for each of the GPU versions (in a single chart for ease of comparison).