Lab Assignment 5
Due Nov 15, 2023 at 11:59pm

1 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 rollback to 0 allowed). The input length can be assumed to be less than $2\hat{3}2$. 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.

2 Instructions

Edit the skeleton 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 CUDA kernel

Compile the template with the provided `Makefile`. The executable generated as a result of compilation can be run using the following code:

```
./Histogram_Template -e <expected.raw> -i <input.raw> -o <output.raw>
-t integral_vector
```

where `<expected.raw>` is the expected output, `<input.txt>` is the input dataset, and `<output.raw>` is an optional path to store the results.

`README.md` has details on how to build `libgputk`, `template.cpp` and the dataset generator.

3 What to Turn in

Submit a report that includes the following:

1. For the histogram kernel, how many atomic operations are being performed by your kernel? Explain.
2. For the histogram kernel, what contentions would you expect if every element in the array has the same value?
3. For the histogram kernel, what contentions would you expect if every element in the input array has a random value?
4. Your version of `template.cu`.

5. The result as a table/graph of kernel execution times for different input data, with the system information where you performed your evaluation. Run your implementation with the input generated by the provided dataset generator. For time measurement, use `gpuTKTime_start` and `gpuTKTime_stop` functions (You can find details in `libgputk/README.md`).