GPU Project: Modern CUDA Optimizations by Manish Chugani

Student ID: 862388066

- Overview of project

  This project implements CUDA Streams and Unified Memory for Histograms on GPUs for improved performance and parallel execution.

- Technical description of your implementation. Such as, what optimizations did you use? Which libraries were used? How was your algorithm designed/implemented? The technical description would vary based on your project goals.

  The code initially uses cudaHostAlloc to allocate Pinned Memory on the CPU. Using cudaMemcpyAsync coupled with Stream based execution, the code achieves high level parallelism. It uses 3 streams by splitting the input array into 3 segments and executes each segment on a single stream. The queues are also optimized by executing hostToDevice function calls first followed by kernel function calls and finally ending with deviceToHost copy function calls.

- Status of your project. Is it feature complete? Does everything work? What does not work? What are limitations of your project? (e.g. only works on square matrix) What major technical challenges did you encounter?

  The project is feature complete and all tests pass if the steps in the Readme are followed.

- Evaluation / Results. Timing of CUDA code vs Numba code. Timing of various implementations. Bottleneck analysis using profiling. Screenshots of your project running. etc...

  Synchronous and Serial Code inefficiencies are removed and the parallelized code executes nearly 7 times faster than the original non-stream based code.

```
bender /home/csgrad/mchugani/GPU_Project/final-project-manish-chugani $ ./histogram 32

Setting up the problem ... 0.104319 s
    Input size = 32
    Number of bins = 8
Allocating device variables ... 0.000075 s
Copying data from host to device and launching kernel ... 0.000250 s
Copying data from device to host ... 0.000011 s
Verifying results ... 0: 2/2, 1: 4/4, 2: 6/6, 3: 7/7, 4: 3/3, 5: 2/2, 6: 4/4, 7: 4/4,
TEST PASSED
```

```
Setting up the problem ... 0.135162 s
    Input size = 1000000
    Number of bins = 4096
Allocating device variables ... 0.000151 s
Copying data from host to device and launching kernel ... 0.001653 s
Copying data from device to host ... 0.000015 s
```

- Documentation and outline of how to compile and run your project. Description of expected results when running your project. Which input should be used for testing?

  Steps to execute the code:

  1: Clone the repository into a local system that has a GPU Device.

  2: Run the make command.

  3: Run the "./histogram" command with command line arguments for number of elements and number of bins for the histogram.

  4: The code implements Unified Memory using CUDAMallocManaged & CUDAHostAlloc and CUDA Streams using CUDAMemcpyAsync function calls for histograms. The tests pass showing that the project is complete in its functionality.