Lab3 Report Steve Deng Zishi

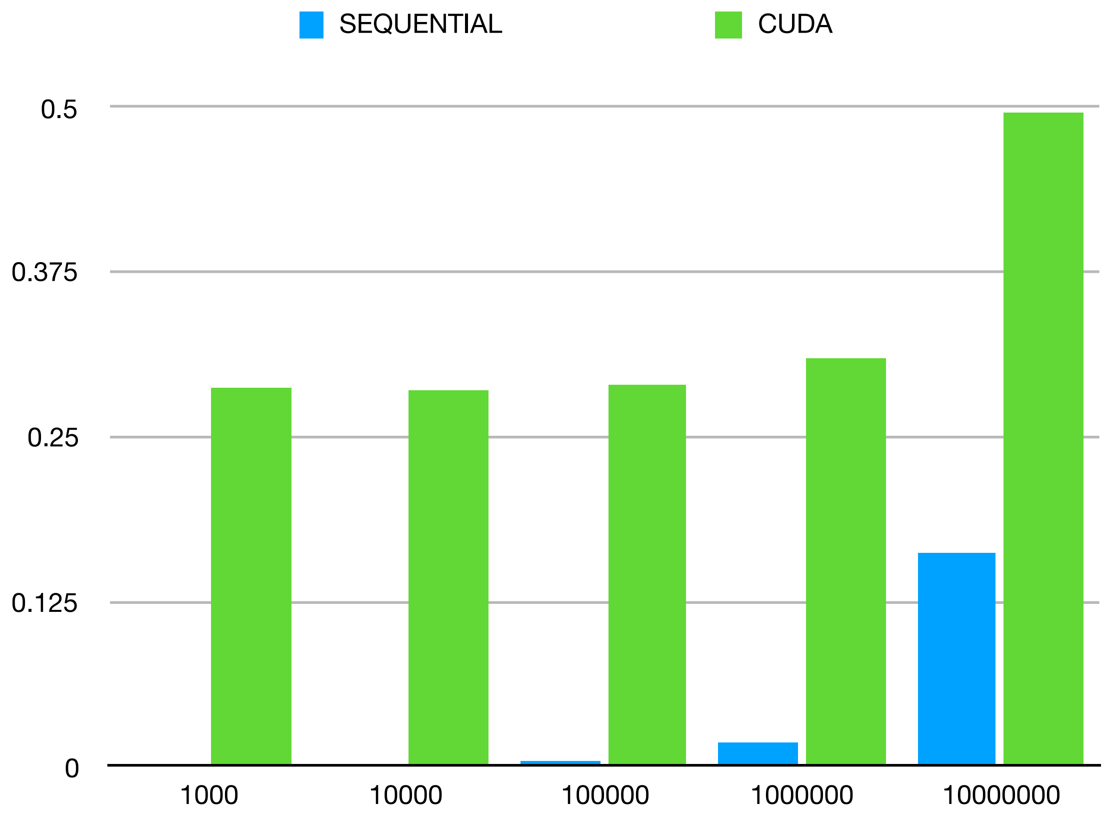
Machine: I used cuda5 TITAN Z for conducting the experiment

1. I picked block size of 1024 threads with 1 dimension and 1 dimensional grid with ceil(size/blocksize) blocks. It is because the max threads per block for the device is 1024 and since we are doing a reduction on the block to get the maximum, we want to have as many threads as possible in a block to maximize the efficiency of reduction. As we are dealing with one dimensional data array, hence using one dimension block and one dimensional grid make sense to deal with the data.

2. nvcc maxseq.cu -o max_cuda

3.

	1000	10000	100000	1000000	10000000
Sequential	0.002	0.002	0.003	0.0185	0.162
CUDA	0.288	0.286	0.290	0.309	0.495



4. As we can see in the graph, CUDA running on all dataset takes longer time than the sequential version. However the ratio of CUDA/SEQUENTIAL is decreasing as the size of the dataset becomes larger. The inefficiency is because finding maximum across is not fully parallelizable as there will be many explicit synchronization barriers to make sure the data is fully loaded to shared memory by all threads and the previous round of reduction has finished across all threads, branch divergence exists in updating max in the global device memory across different blocks. However the ratio is improving as data size get larger because the time gained through parallel reduction of multiple threads offset the lost of efficiency to manage all threads by GPU. As the problem size continues to become larger, we can possibly see a better performance of CUDA against the sequential version.