CSCI-GA-3033 GPUs Fall 2024 - Lab 1

Manisha Goyal – mg7609@nyu.edu

Experiment 1: Varying Blocks and Threads, Fixed Input Size

The input size for this experiment was fixed at 1,000,000 elements, and the performance was compared across different configurations of blocks and threads per block. The results are shown in Table 1, Figure 1 and Figure 2 below.

Configuration (Blocks, Threads)	**CPU Time (seconds)**	**GPU Time (seconds)**	**Speedup (seconds)**
4 blocks, 500 threads	0.003279	0.002958	1.108519
8 blocks, 500 threads	0.003285	0.002703	1.215316
16 blocks, 500 threads	0.003285	0.002600	1.263461
4 blocks, 250 threads	0.003291	0.003115	1.056500
8 blocks, 250 threads	0.003286	0.002850	1.152982
16 blocks, 250 threads	0.003287	0.002779	1.182799

Table 1: Results for Experiment 1

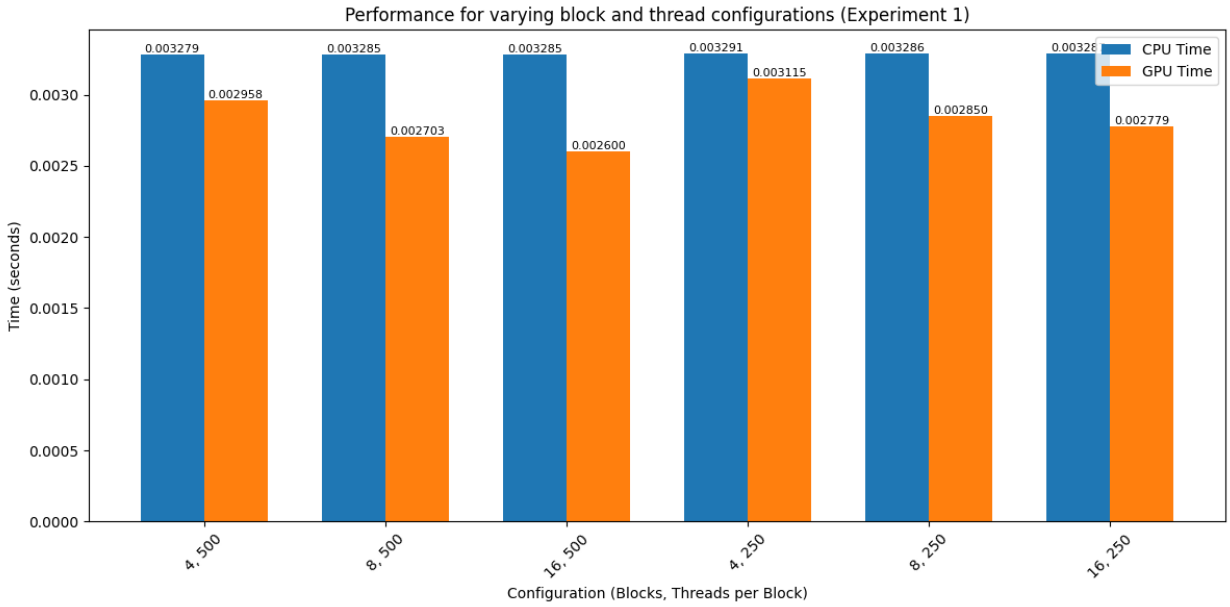

Figure 1: Results (CPU Time and GPU time) for Experiment 1

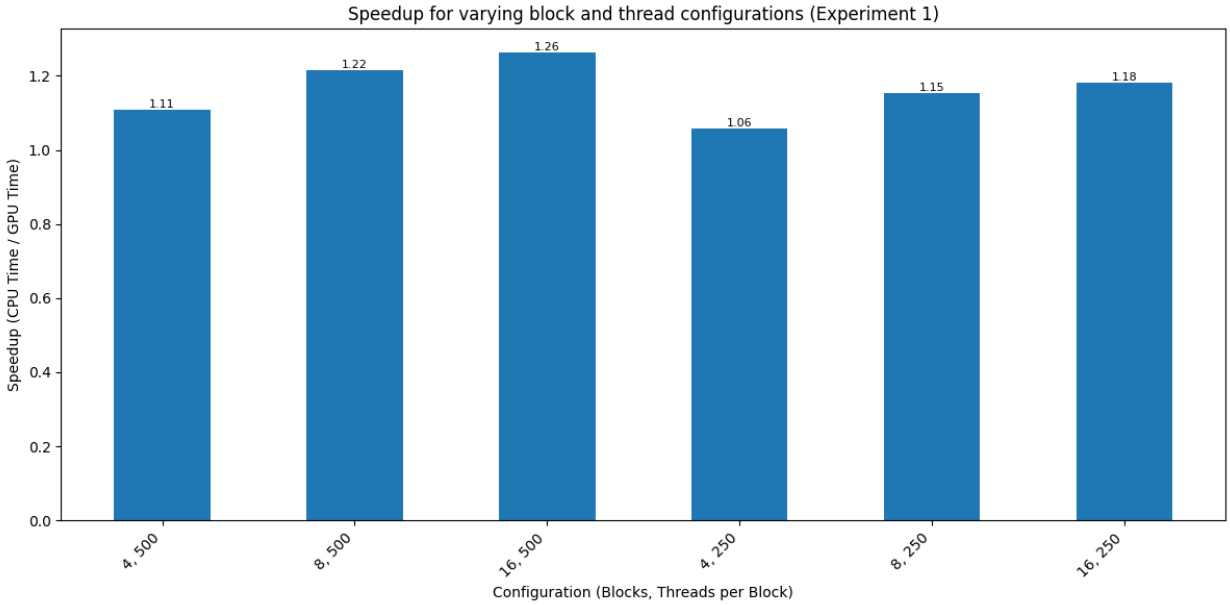

Figure 2: Results (Speedup) for Experiment 1

Behavior observed in the graph: The GPU is consistently faster than the CPU for all configurations. Increasing the number of blocks generally reduces GPU time, particularly when the number of threads per block is small (250).

Reason for the observed behavior: The consistently faster GPU times result from its parallel execution of tasks across many cores, compared to the CPU's sequential execution. The improvement in performance as the number of blocks increases is due to better utilization of the GPU's parallel architecture. When there are more blocks, the workload is spread out more effectively across the available SMs, leading to better parallelization. The best performance was observed with 16 blocks and 250 threads per block, as this configuration strikes a good balance between maximizing parallelization and minimizing underutilization.

Experiment 2: Varying Input Sizes, Fixed Configuration

In this experiment, the block and thread configuration was fixed at 8 blocks with 500 threads per block, while the input size varied from 100 to 100,000,000 elements. The results are shown in Table 2, Figure 3 and Figure 4 below.

Input Size (elements)	**CPU Time (seconds)**	**GPU Time (seconds)**	**Speedup (seconds)**
100	0.000002	0.000503	0.003976
1,000	0.000005	0.000518	0.009652
10,000	0.000034	0.000585	0.058119
100,000	0.000335	0.000915	0.366120
1,000,000	0.003291	0.002750	1.196727
10,000,000	0.033019	0.019949	1.655170
100,000,000	0.330456	0.174408	1.894729

Table 2: Results for Experiment 2

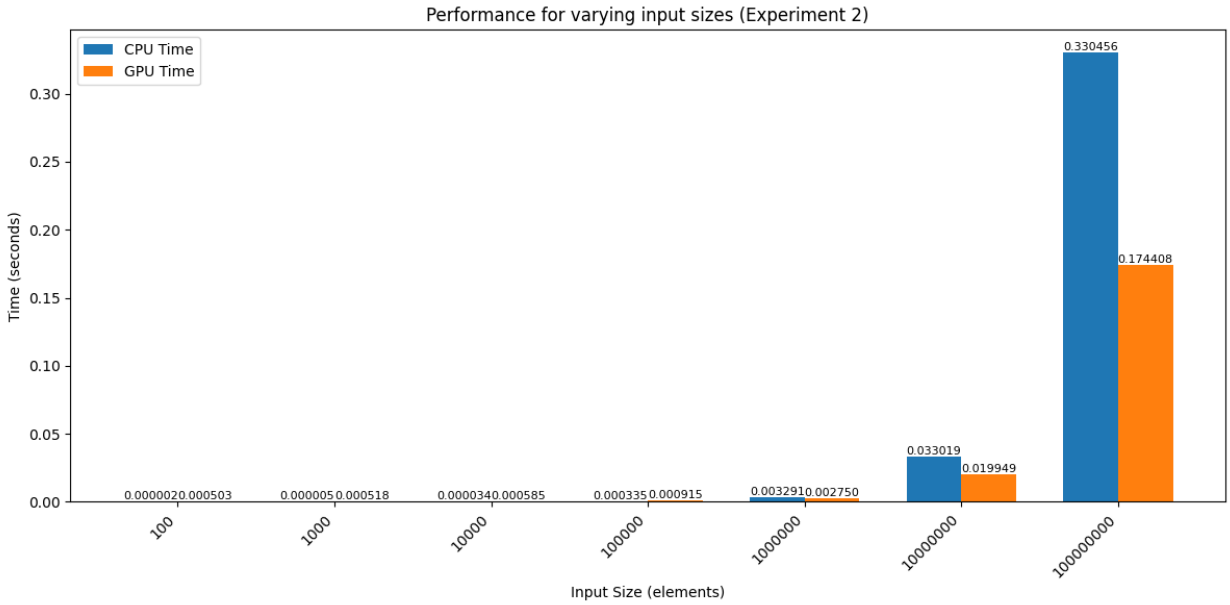

Figure 3: Results (CPU Time and GPU time) for Experiment 2

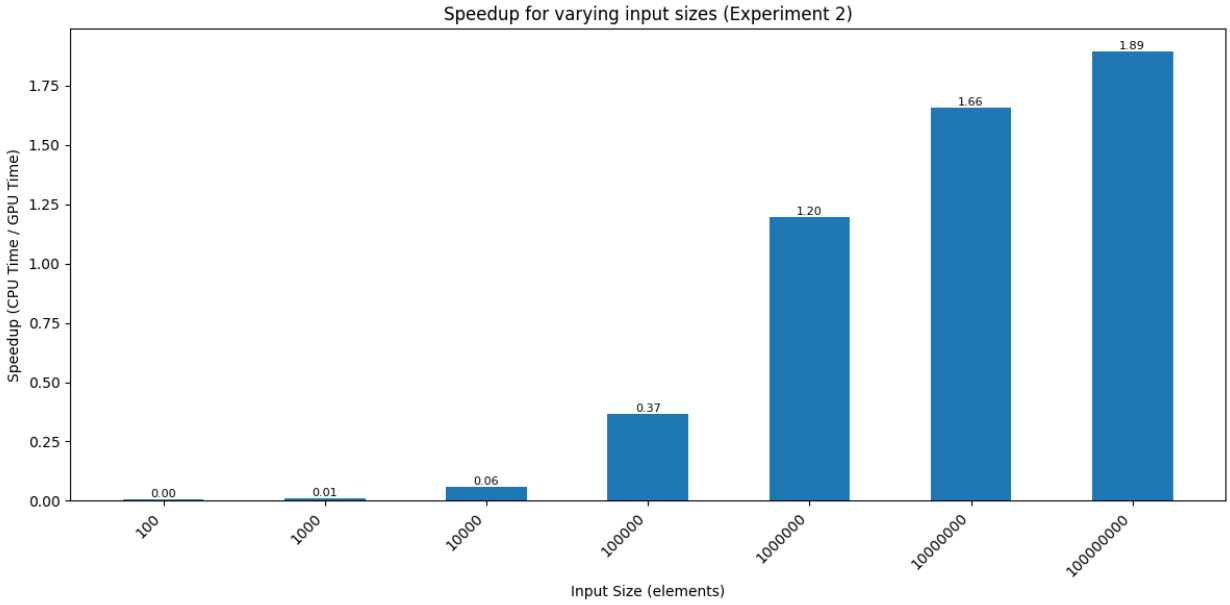

Figure 4: Results (Speedup) for Experiment 2

Behavior observed in the graph: As the input size increases, both the CPU and GPU times increase. However, for small input sizes (up to 100,000 elements), the CPU is actually faster than the GPU. As the input size increases, the GPU begins to outperform the CPU,

and for very large input sizes (1,000,000 elements and above), the GPU's advantage becomes significant.

Reason for the observed behavior: For small input sizes with minimal parallelism, the GPU's overhead (data transfer and kernel launch) outweighs the parallel processing benefits, making the CPU more efficient. However, as the input size grows, the GPU becomes much more efficient due to its highly parallel architecture, which can process many elements simultaneously. Thus, for large input sizes, the parallel execution on the GPU results in much faster processing times, significantly outperforming the CPU, which struggles to handle larger datasets sequentially.