1. Assignment 4: MPI + CUDA

1.1 Performance Analysis

We implement MPI + CUDA vector addition. Table 0 demonstrates the total and kernel execution times of our vector addition using a fixed vector size (100,000,000) and one GPU and process. We observe that 512 threads per block gives us the best kernel configuration.

Table 0: Execution times of vector addition using one GPU/process for vector of size 100,000,000

		Total Execution Time (s), Kernel Execution Time (ms)		
Threads Per Block	Grids Per Block	Trial 1	Trial 2	Trial 3
16	6250000	(1.084, 8.726)	(1.084, 8.722)	(1.083, 8.722)
32	3125000	(1.079, 4.380)	(1.074, 4.390)	(1.075, 4.386)
64	1562500	(1.075, 2.044)	(1.072, 2.207)	(1.079, 2.215)
128	781250	(1.077, 1.133)	(1.071, 1.044)	(1.076, 1.139)
256	390625	(1.074, 0.950)	(1.074, 0.947)	(1.071, 0.957)
512	195313	(1.072, 0.927)	(1.078, 0.953)	(1.071, 0.954)
1024	97657	(1.075, 0.994)	(1.071, 0.958)	(1.083, 0.995)

We graph the above table in Figure 1 for Trial 3 and observe an exponential decrease in kernel execution time as the number of threads per block increases.

Figure 2 demonstrates the scalability of the program. We fix the vector size (100,000,000) and use the optimal kernel configurations (512 threads per block). We vary the number of MPI processes and analyze the execution time. We observe that after 4 MPI processes, the execution time increases. In the ideal case, it should linearly decrease while increasing the number of processes. There are a couple reasons why this may be. For large numbers of MPI processes, there is

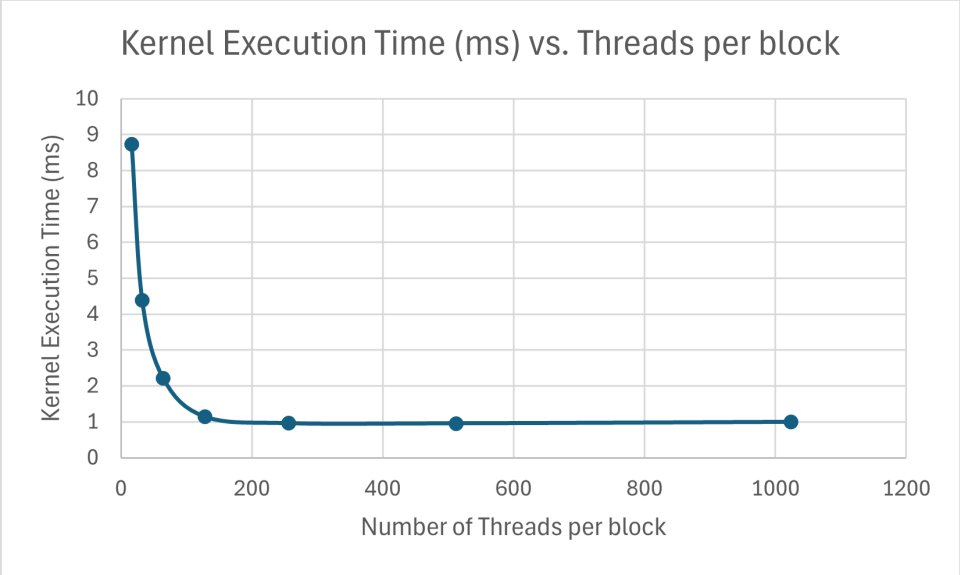

Figure 1: Kernel Time vs the number of threads per block

less data to work with per process. This means it is possible that GPU launches are less efficient, more communication per unit of useful work, and overhead becomes a bigger percentage of total time.

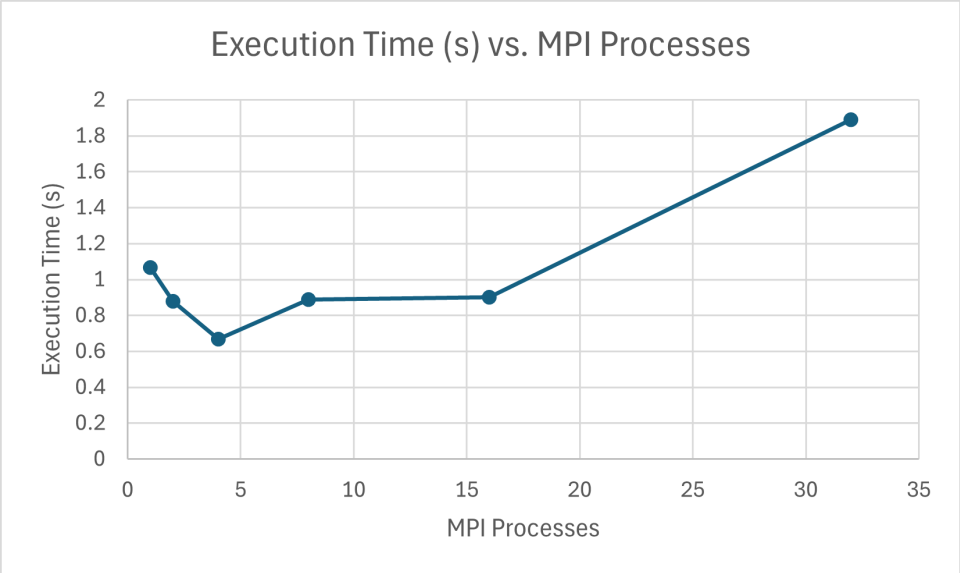

Figure 2: Time vs the number of MPI processes

Finally, we simply analyze the effect of different vector sizes on execution time. For our configurations, we use 4 MPI processes and 512 threads per block. The results are shown in Table 2. We observe that execution time largely doesn't scale until a vector size of 100,000,000.

Table 2: Comparison between vector sizes and execution time

Vector Size	Execution Time (s)
1000	0.52225
10000	0.42206
100000	0.370877
1000000	0.418249
10000000	0.394974
100000000	0.726235