ECE 60827 CUDA Assignment 1
Zekun Wu (wu1326@purdue.edu) (Github ID: zwu225)

Part A:

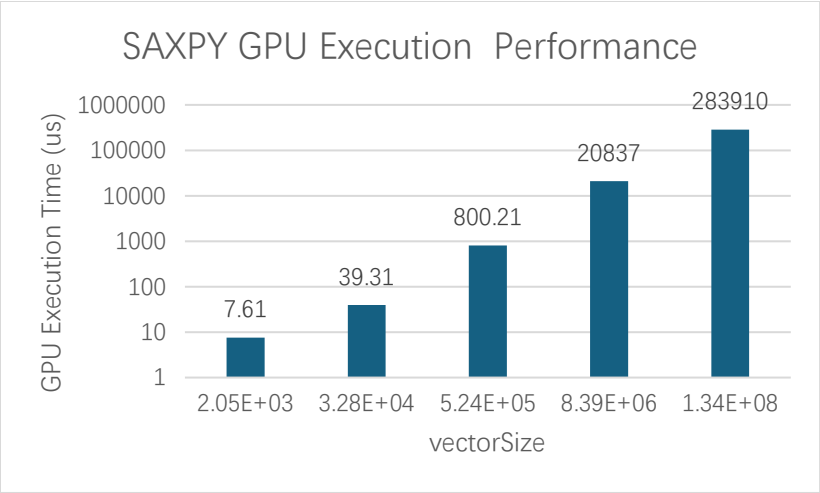

Fig 1. SAXPY GPU execution time in microseconds vs vectorSize input.

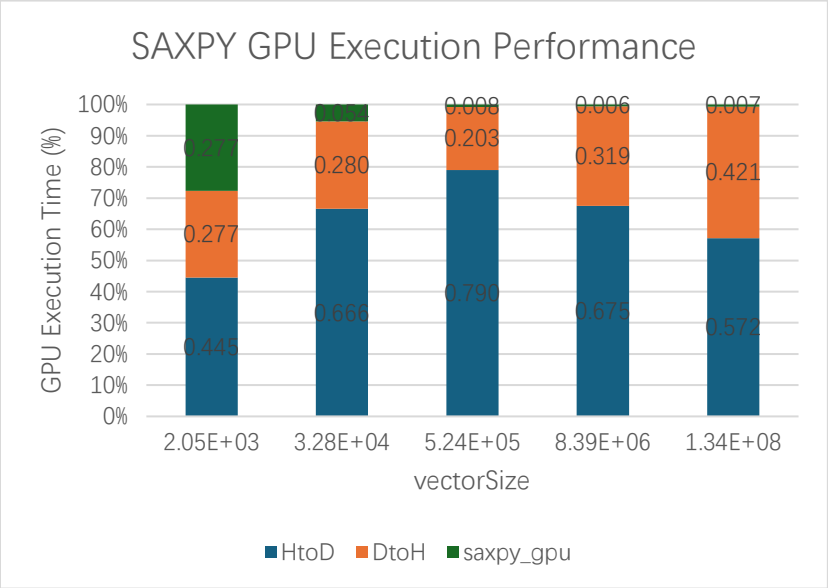

Fig 2. SAXPY GPU execution time percentage for each major time-consumers.

vectorSize	HtoD (us)	DtoH (us)	saxpy_gpu (us)	total (us)
2.05E+03	3.39	2.11	2.11	7.61
3.28E+04	26.17	11	2.14	39.31
5.24E+05	631.8	162.27	6.14	800.21
8.39E+06	14072	6640	125	20837
1.34E+08	162340	119590	1980	283910

Table 1. SAXPY GPU execution time for each major time-consumers.

There are several observations from the above profiling results. First, when the vectorSize is relatively small (i.e. 2.05E+03 and 3.28E+04), the percentage of execution time from saxpy_gpu is significantly higher than the rest. Another observation is that for the majority of the time of execution the GPU is performing I/O from and to the CPU. When running the code, the actual time of execution (from start of the program to finish), which combines the CPU and GPU load, is significantly longer than the pure GPU execution time.

Part B:

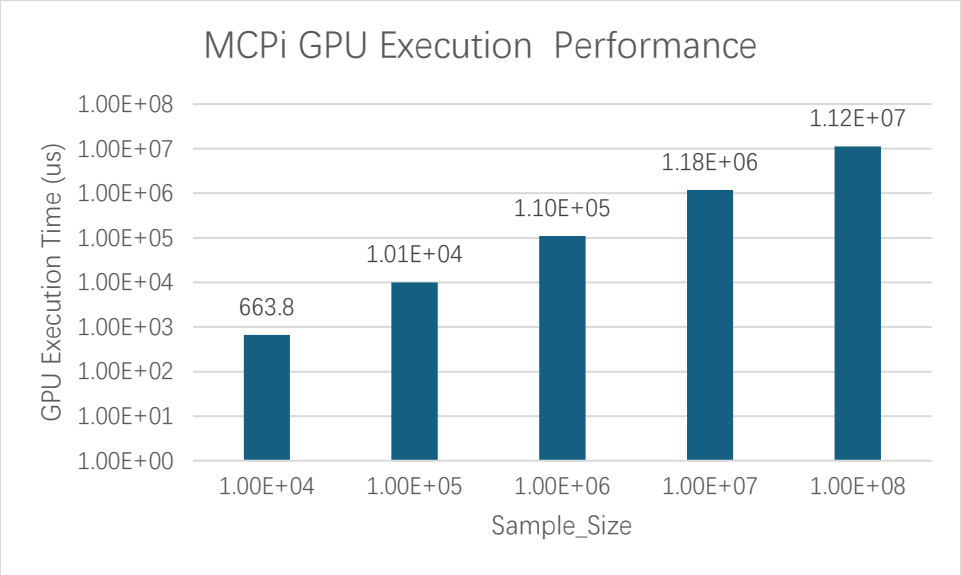

Fig 3. MCPi GPU Execution performance with varying Sample_size.

When varying Sample_Size for MCPi program, it is observed that the DtoH have consistent 1.92 microseconds execution time. The generatePoints spend increasing time as sample size increases.

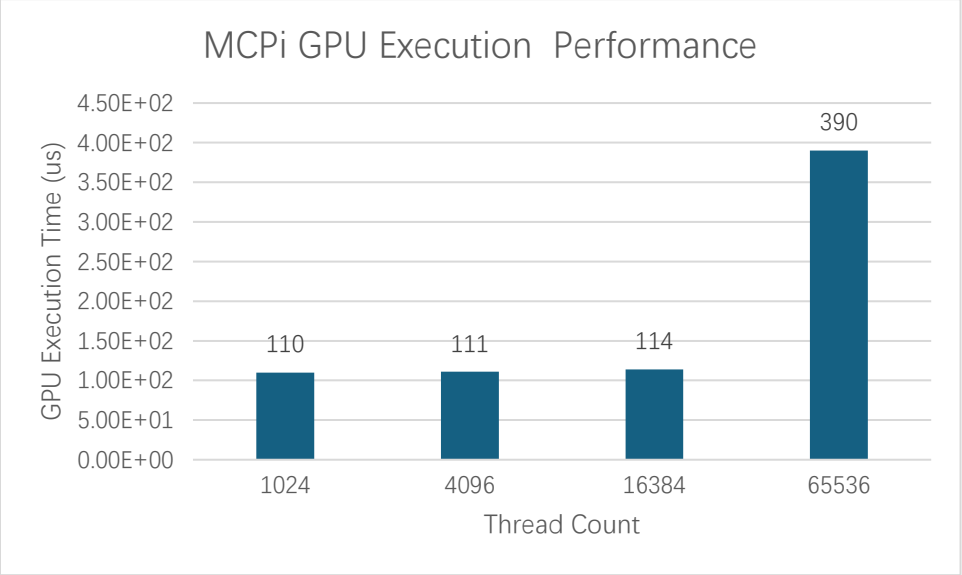

Fig 4. MCPi GPU Execution performance with varying Thread Count

When varying launched thread count, the GPU performance is basically the same from 1024 to 16384 threads, with 256 threads per thread block. The 65536 thread case has significant increase of execution time due to possible exceeding maximum thread count allowed by this GPU.