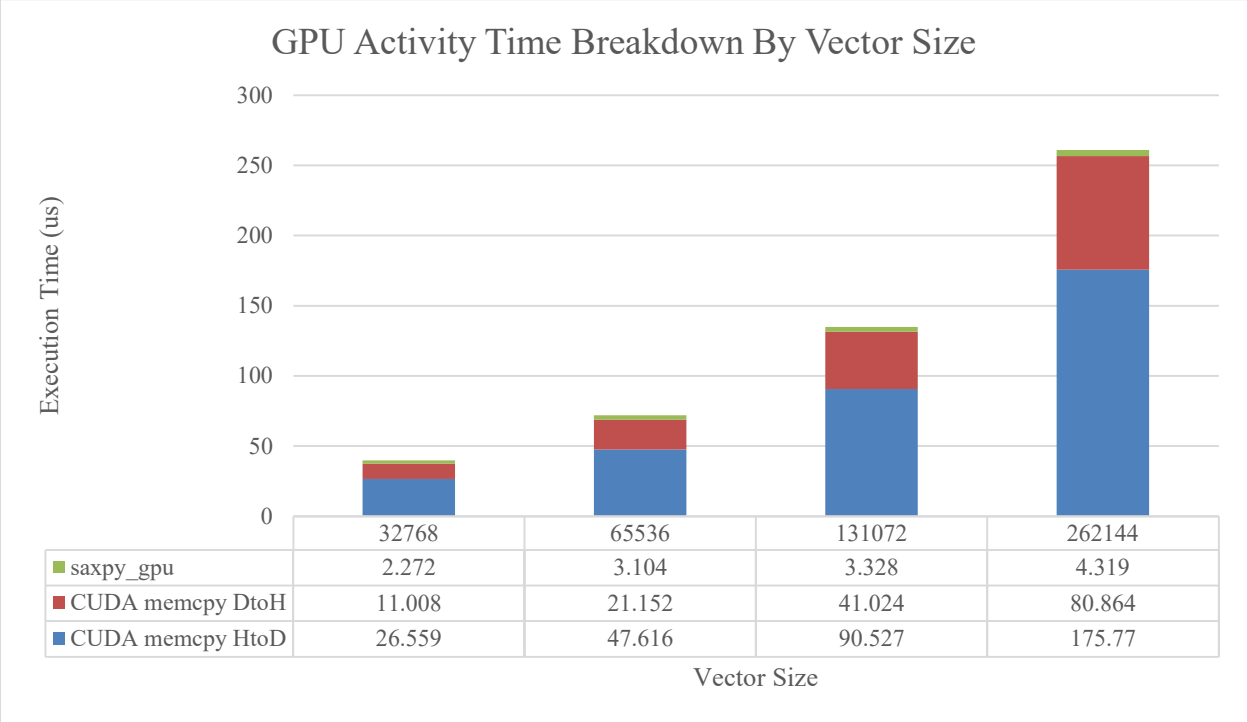
ECE 60827 CUDA Programming Part 1

Rick Kittelson

Part A

The controlled variable for GPU timing in Part A is the size of the vector on which SAXPY is performed. Fig. 1 demonstrates the timing breakdown for 4 different vector sizes doubling in size. Table 1 displays the increase in time taken for each major GPU activity compared to the previous iteration. Table 2 displays the total execution time for each test and the percent increase from test to test.



	32768	65536	131072	262144
saxpy_gpu	2.272	3.104	3.328	4.319
CUDA memcpy DtoH	11.008	21.152	41.024	80.864
CUDA memcpy HtoD	26.559	47.616	90.527	175.77

Figure 1: GPU Timing Breakdown for Varied Vector Sizes

Table 1: Percent Increase of GPU Activity Times Over Previous Test

Vector Size	32768	65536	131072	262144
CUDA memcpy HtoD	-	79%	90%	94%
CUDA memcpy DtoH	-	92%	94%	97%
saxpy_gpu	-	37%	7%	30%

Table 2: Total GPU Time and Percent Increase Across Tests

Vector Size	32768	65536	131072	262144
Total GPU Time (us)	39.839	71.872	134.879	260.953
Total GPU Time Increase	-	80%	88%	93%

While memory operations seem to grow roughly proportionally to the size of the data, it is apparent that the growth of the kernel execution time is not necessarily growing at the same rate, indicating the value of optimizing to minimize data transfer and maximize time spent in the kernel for applications. This is demonstrated with the reduction in Part B.

Part B

Fig. 2 Displays a GPU timing breakdown for varying the number of samples per thread in the Monte Carlo algorithm. Fig. 3 displays the same data but varying the number of threads while keeping the sample count constant. Tables 3 and 4 display the relevant timing data for each of these tests.

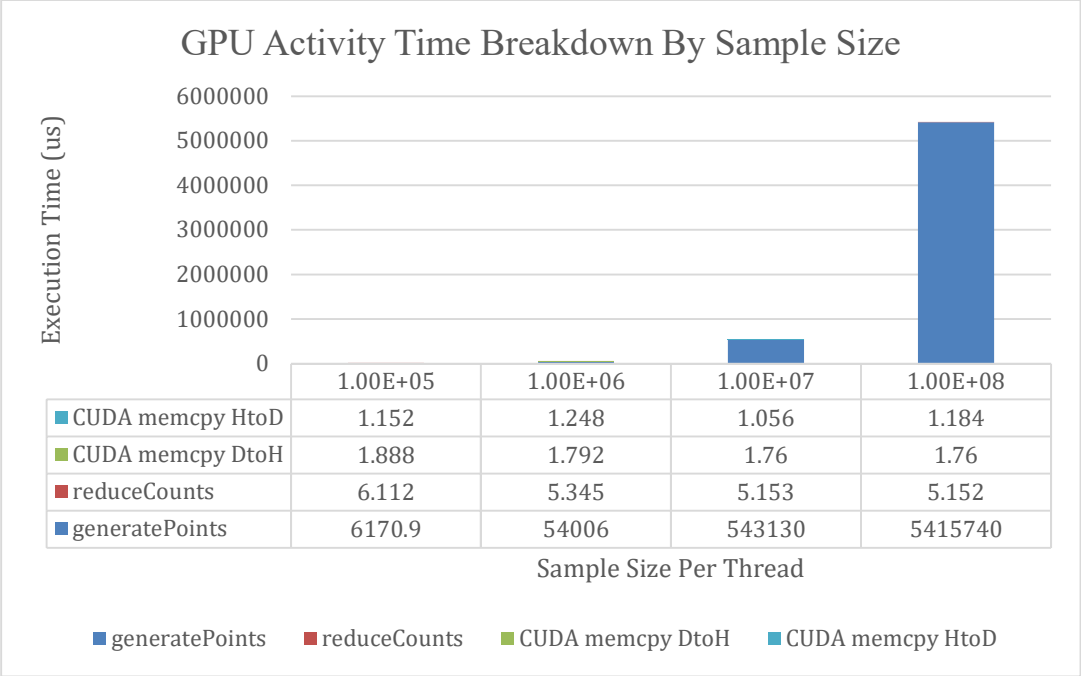


Figure 3: GPU Timing Breakdown for Varying Sample Count Per Thread

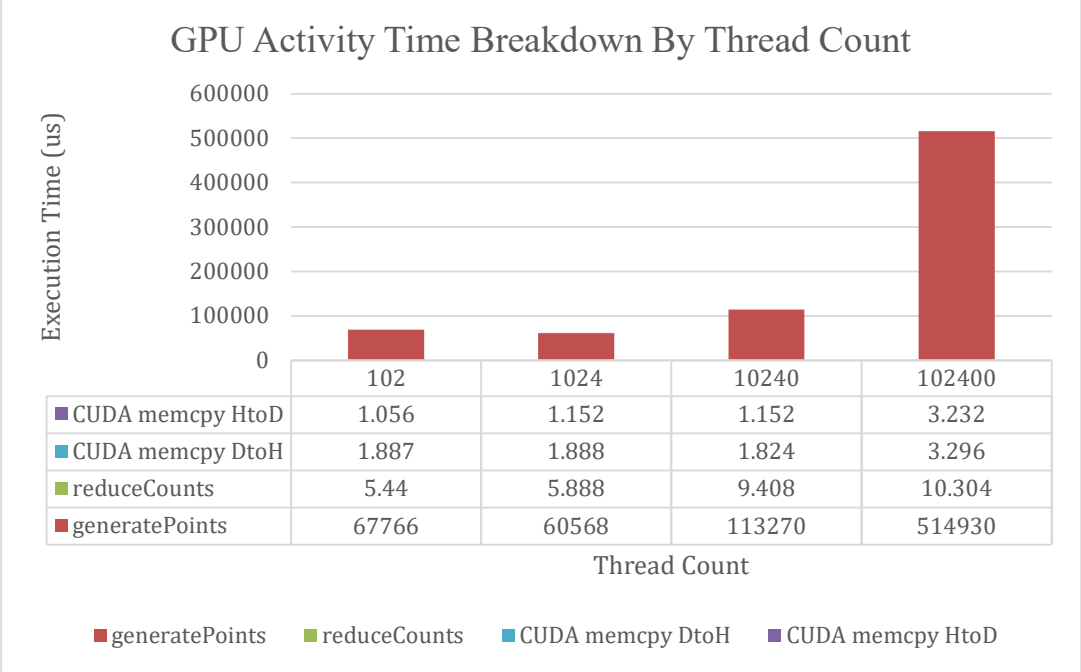


Figure 4: GPU Timing Breakdown for Varying Sample Count Per Thread

Table 3: Total Execution Time and Samples for Varying Samples Per Thread

MC Sample Size	Total Execution Time (us)	Total Samples
1.00E+05	6180.052	1.02E+08
1.00E+06	54014.385	1.02E+09
1.00E+07	543137.969	1.02E+10
1.00E+08	5415748.096	1.02E+11

Table 4: Total Execution Time and Samples for Varying Thread Count

MC Thread Count	Total Execution Time (us)	Total Samples
102	67774.383	1.02E+08
1024	60576.928	1.02E+09
10240	113282.384	1.02E+10
102400	514946.832	1.024E+11

An interesting observation is that for lower total sample counts, increasing the samples per thread results in the fastest execution time, but for larger total sample counts, increasing the number of threads works better. This is because for large total sample counts, a 10x increase in samples per thread lead to a roughly 10x increase in execution time, but the same was not true for a 10x increase in the number of threads. Certainly, this may vary with the hardware used to execute an application, but optimizing large datasets to use more threads rather than more iterations is intuitively the better decision for problems similar to this.