ECE 60827: CUDA Programming Assignment 1

Ronit Nallagatla February 1, 2025

1 SAXPY

The results of running the `saxpy_gpu` kernel on various input vector sizes is shown below. Figure 1 shows the overall execution time including both GPU activity and API calls for input vector sizes between $[2^5, 2^{29} - 1]$. The `cudaMalloc` API call takes up most of the execution time for the smaller input sizes. This overhead overshadows the actual kernel execution time. The execution time remains similar for the vector sizes upto 2^{20}. As the vector size increases significantly (2^{25}), the proportion of time spent copying the data to/from the host increases, which also highlights the overhead of transferring data and setting up the GPU.

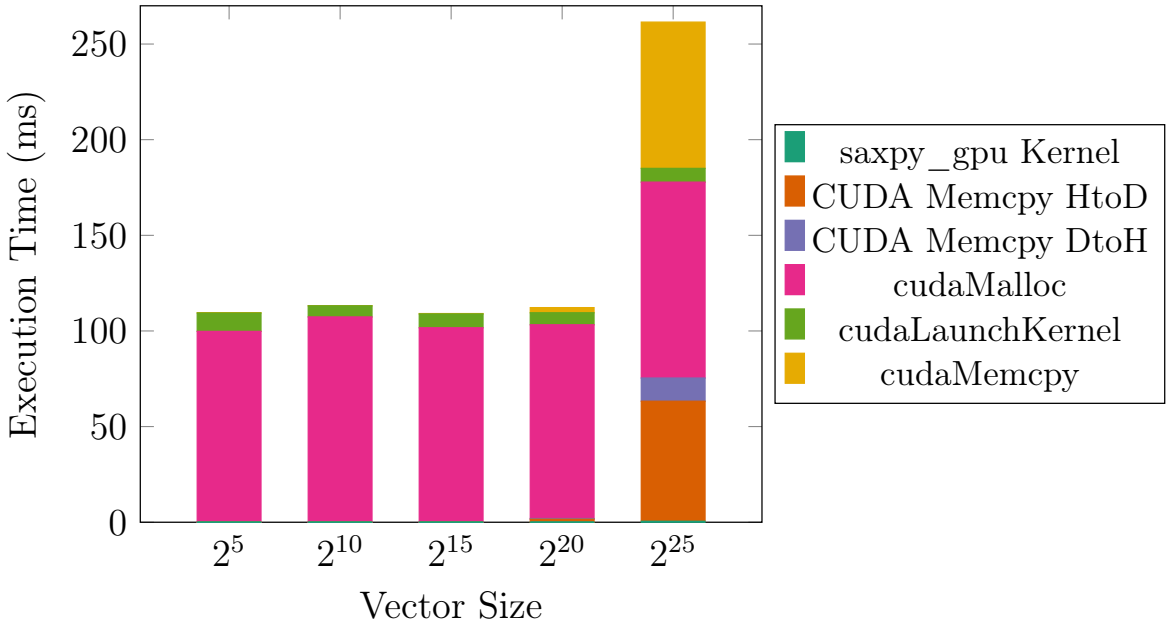

Figure 1: SAXPY - Execution Time for Various Input Vector Sizes

Figure 2 shows the breakdown of GPU activity execution time for the different input vectors. For the smaller input vectors, about an equal amount of time is spent running the `saxpy_gpu` kernel and copying memory to/from host. As the vector size increases, the overhead of moving data dominates.

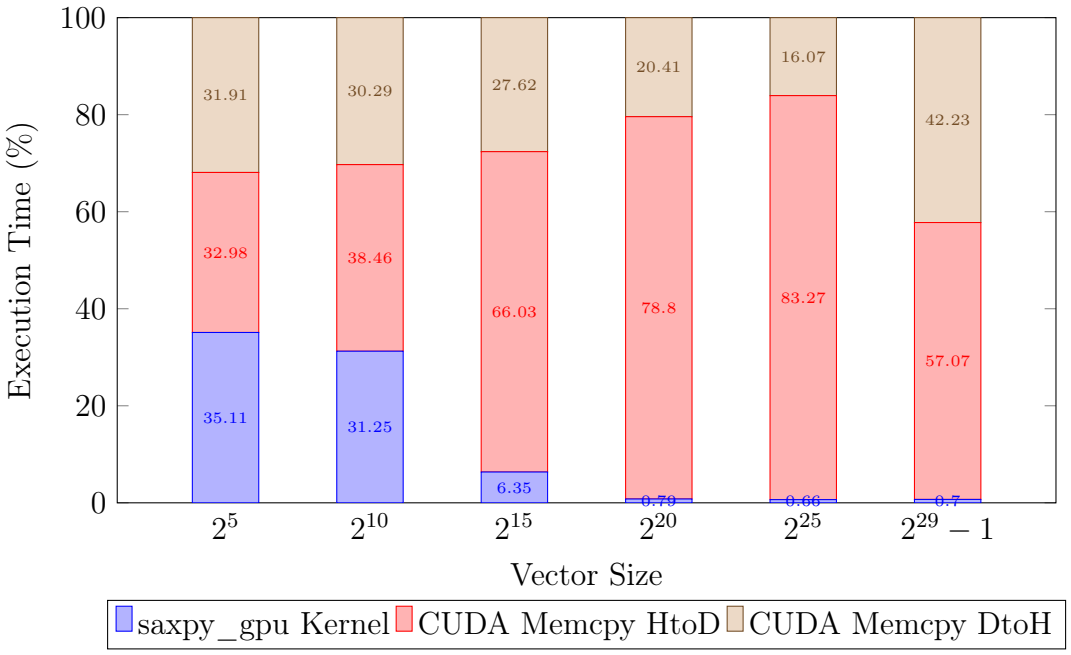

Figure 2: SAXPY - GPU Activity Execution Time % Breakdown

Figure 3 shows that as the vector size increases the `memcpy` overhead increases significantly.

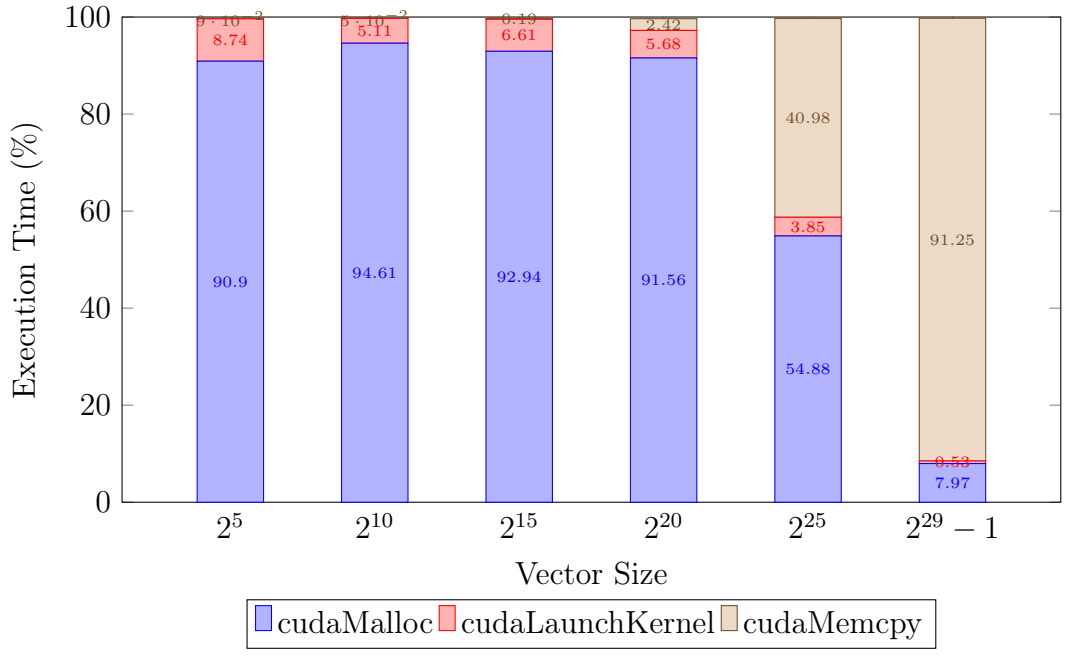

Figure 3: SAXPY - API Call Execution Time % Breakdown

Keeping the input vector size constant at 2^{15} and changing the threads per block in Figure 4 shows that using 128 threads per threadblock slows down the execution time slightly as the higher threadblock count could degrade TB to SM mapping.

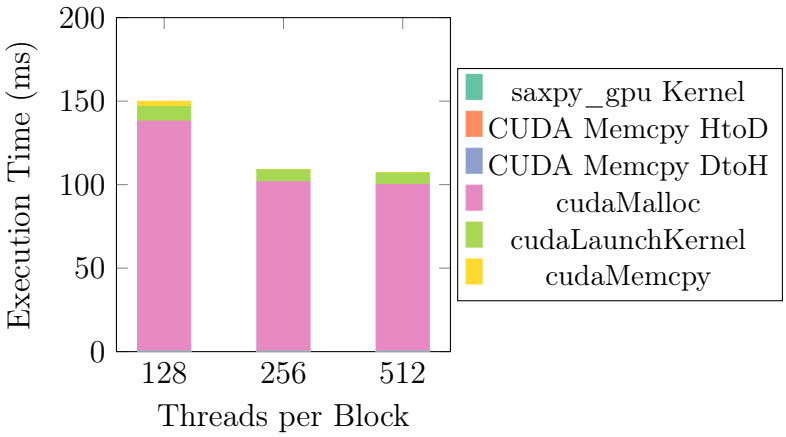

Figure 4: Execution Time with Varying Threads/Block

2 Monte Carlo π Estimation

The results of the Monte Carlo π estimation GPU implementation are described below. Figure 5 shows the execution time breakdown with varying sample sizes between $[10^4, 10^7]$. As the sample size increases, the GPU spends more time running the `generatePoints` kernel since more samples need to be generated and tested with the unit circle. Similarly, the data transfer overhead also increases.

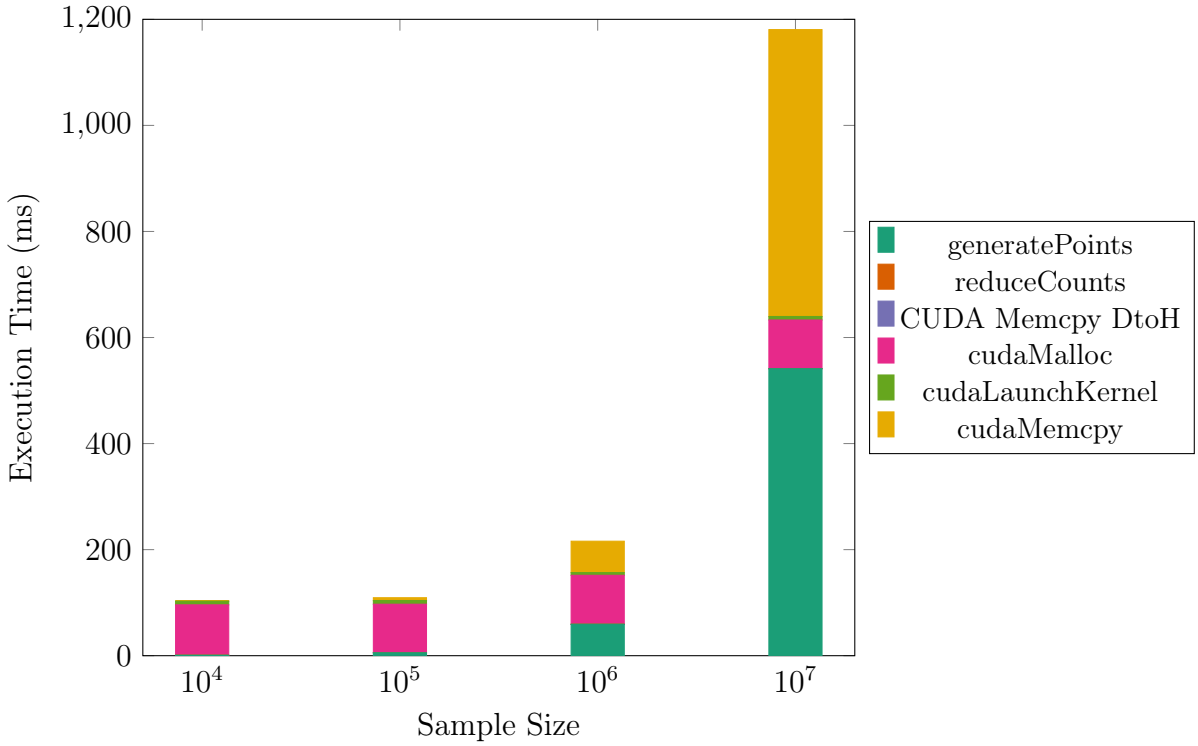

Figure 5: Monte Carlo π Estimation Execution Time with Varying Sample Sizes

In the following figures, the sample size is kept fixed at 10^6. Figure 6 shows the execution time plotted for different generate thread counts. As the thread count increases, the execution time increases. The data transfer overhead also increases, notably the `cudaMemcpy` API call.

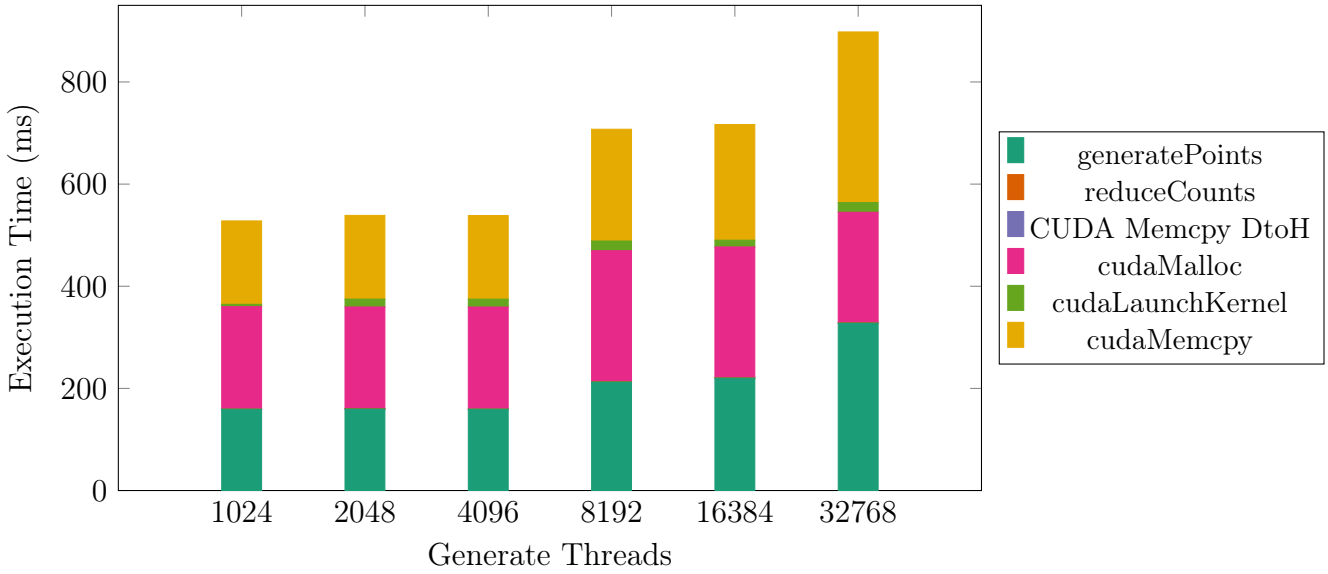

Figure 6: Monte Carlo π Estimation Execution Time with Varying Generate Thread Count

Changing the Reduce Size as shown in Figure 7 does not have a major impact on execution time as most of the exeuction time is spent in the `generatePoints` kernel for this example with an input size of 10^6. For very large input sizes, the reduction will benefit.

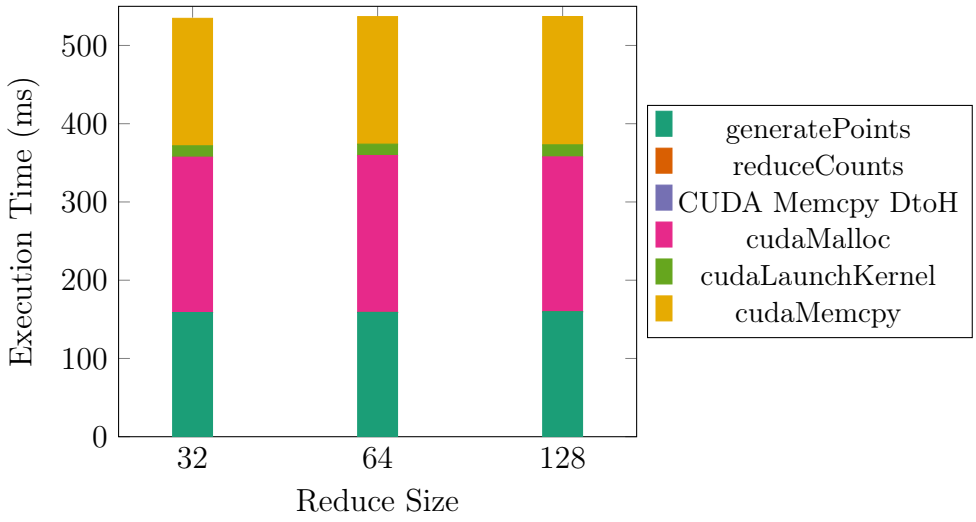

Figure 7: Monte Carlo π Estimation Execution Time with Varying Reduce Size