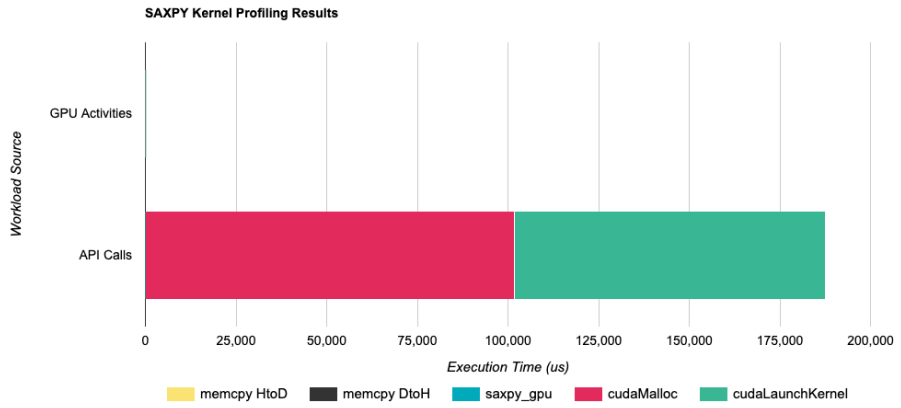
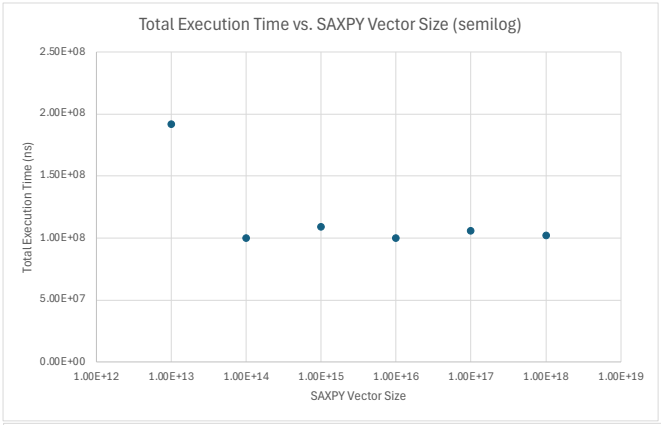
Aidan Prendergast
ECE 60827
Assignment 1
02/03/25

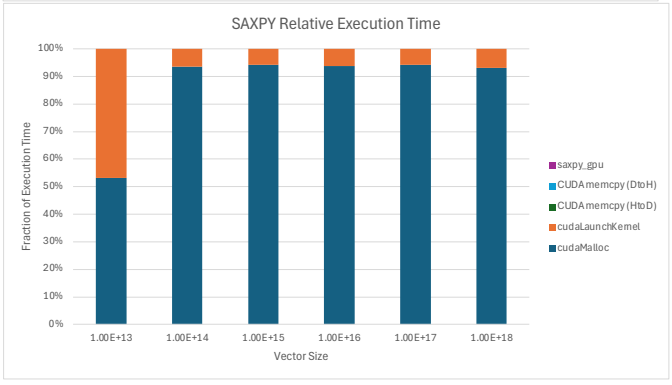
Intro to CUDA: SAXPY and Monte Carlo GPU Kernels

1. Initial impressions of the default size vector show that the SAXPY program is heavily dominated by API-side behavior. This is understandable as each thread does very little work (one fused-multiply-add operation), which fails to amortize the overhead of the launch and malloc done by the CPU.



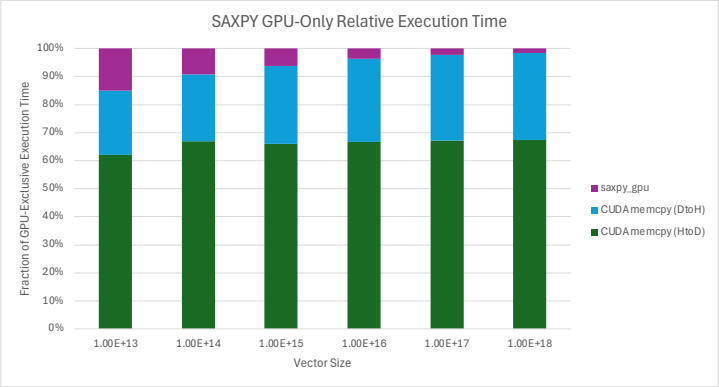
Sweeping the vector size parameter, which modifies the number of threads launched, yields no noticeable effect on the total program time. This demonstrates that, even for these enormous vector sizes, the GPU has enough resources to effectively run the threads in parallel. There is sufficient Fine-Grained Multithreading available that no matter the thread count in this range, these threads can be effectively overlapped.



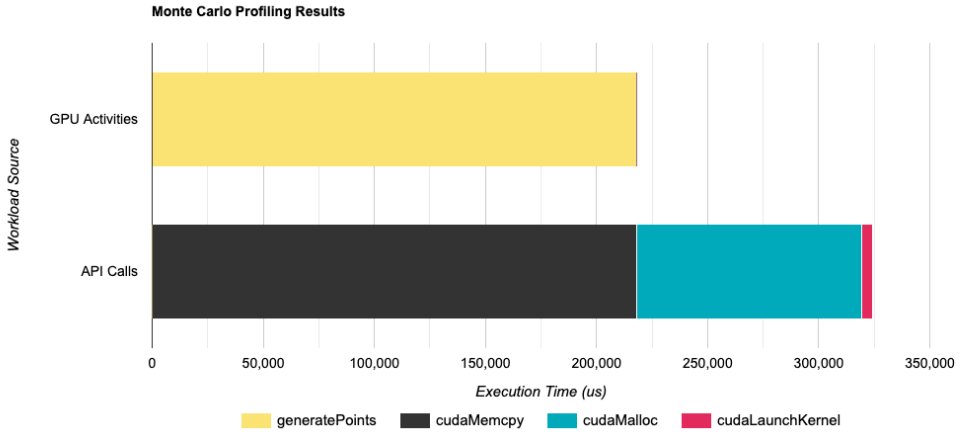


Entirely invisible compared to the API calls, the GPU activity does show more of a trend when compared over a multitude of vector sizes. Specifically, the fraction of execution time taken by the saxpy function decreases as vector size increases, dominated by the memory

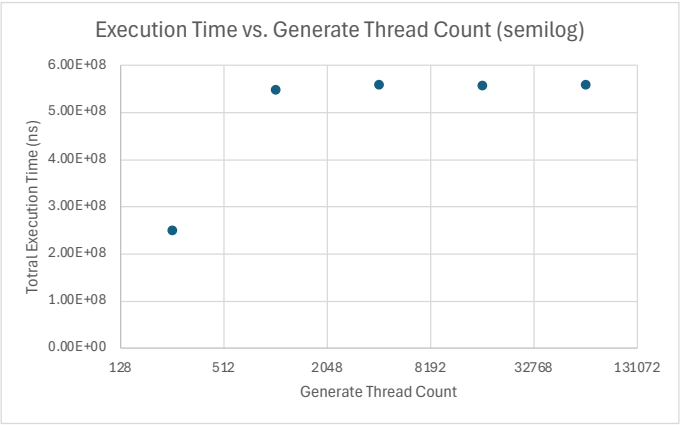
copy operations between the CPU and GPU. This demonstrates that the SAXPY kernel is a memory-bound operation rather than a compute-limited one with the current one-thread, one-element approach, and therefore could likely be better tuned to scale better with size.



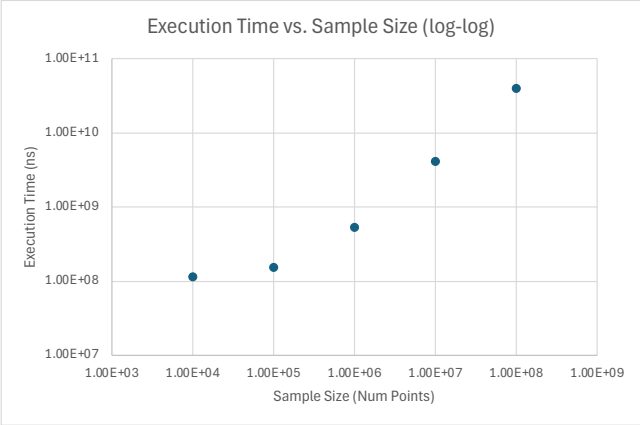
2. Initial impressions of the default settings for the Monte-Carlo Pi Estimation Kernel show a far more balanced workload, where the GPU and API timings are more balanced. This still demonstrates that the GPU is failing to amortize the API calls effectively.



First, the generator thread count was varied. This yielded another flat relation, which is understandable as each thread has an amount of work independent to the number of threads. This result is interesting as a demonstration of fixed thread workload but unhelpful as a dive into the behavior of the MC program's behavior.



The Sample Size parameter is the main one of interest for the Monte Carlo Pi kernel, as this varies the work that each thread must perform. This curve is far more regular, showing the program scaling in the large limit and the fixed costs of kernel launch when sample size is small. The best example to demonstrate this behavior can be seen in the relative execution time of the major components of the program.



Specifically, the plot shows how the Malloc and Kernel Launch dominate when the thread workload is small but become negligible as workload increases. At sample sizes of 10^7 and 10^8, the memory and compute demand of the program (memcpy and generatePoints) are almost exactly balanced 50/50.