ECE 60827 Programming Assignment 1

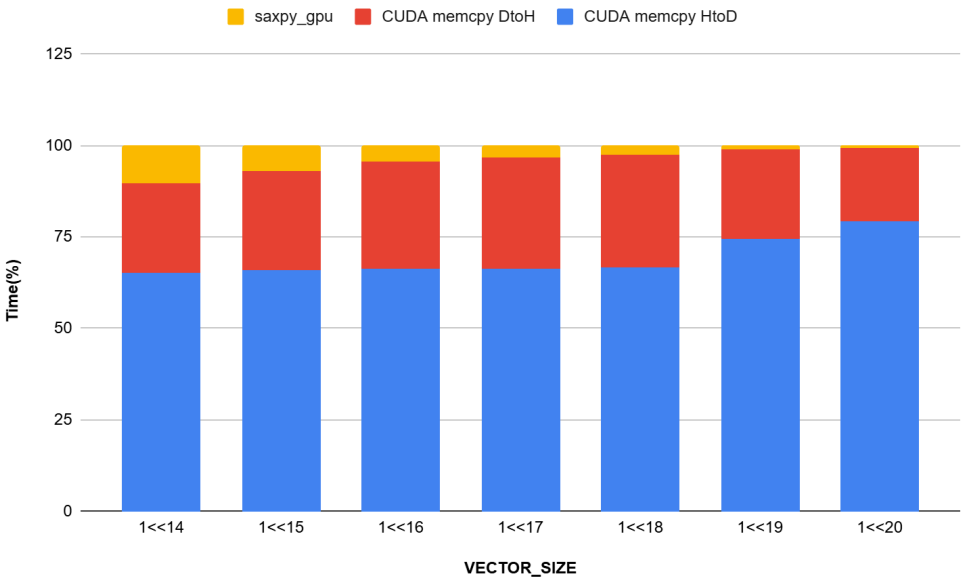
Abhilash Ashok Achary

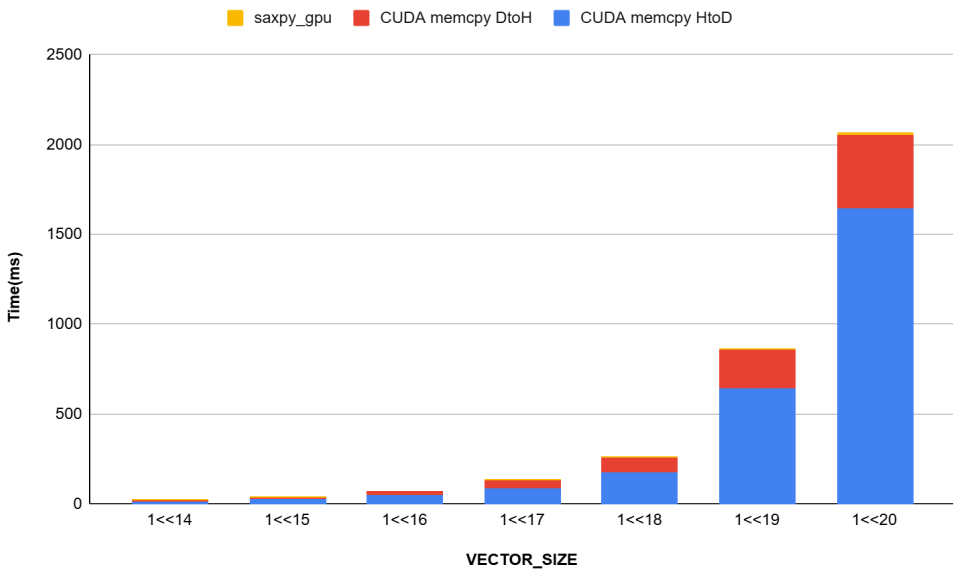
PART A: Single-precision A · X Plus Y (SAXPY)

The **VECTOR_SIZE** is iterated over 1 << 14 to 1 << 20 and the GPU activities and API calls are profiled.

GPU Activities:

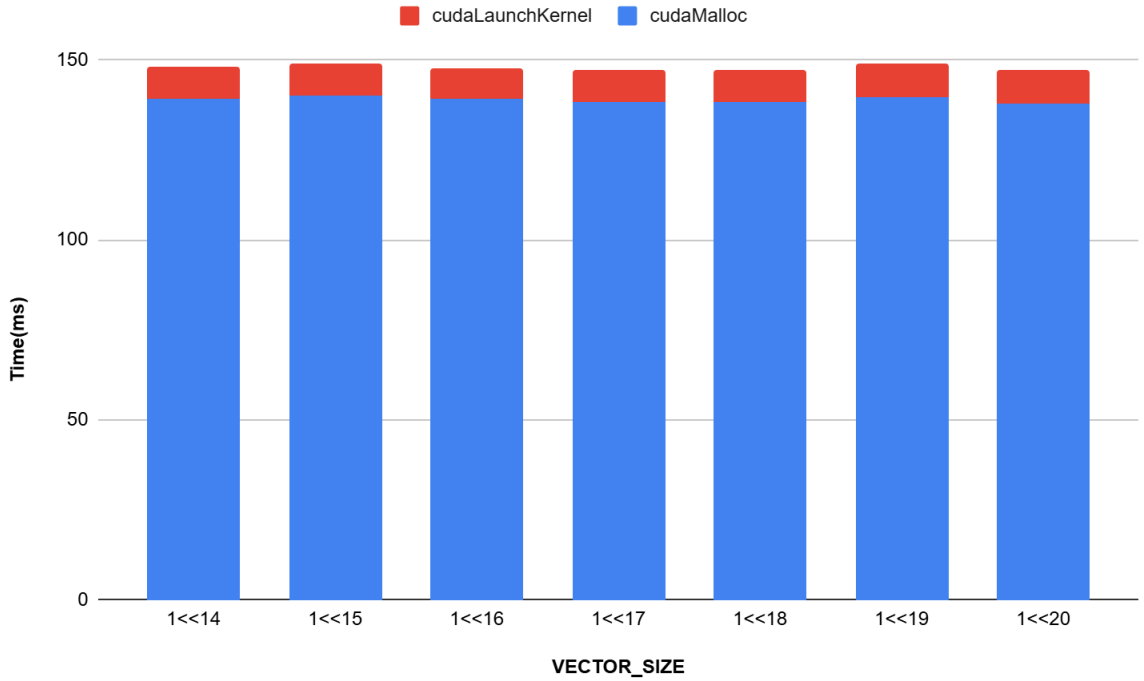
It can be seen that as VECTOR_SIZE increases, the execution time of the program also increases CUDA memcpy Host to Device contribute the most to execution time, and its percentage increases from 65% at 1<<14 to 80% at 1<<20. It is also noted that percentage of execution time by saxpy goes down from 10% to 0.86%





API Calls:

As the VECTOR_SIZE increases, it can be seen that there is not much of a difference in the in cudaMalloc and cudaLauchKernal, aprt from 2% dip at 1 << 16 and 1<< 18.



PART B: Monte Carlo estimation of the value of π

Experiment 1. Varying `SAMPLE_SIZE`

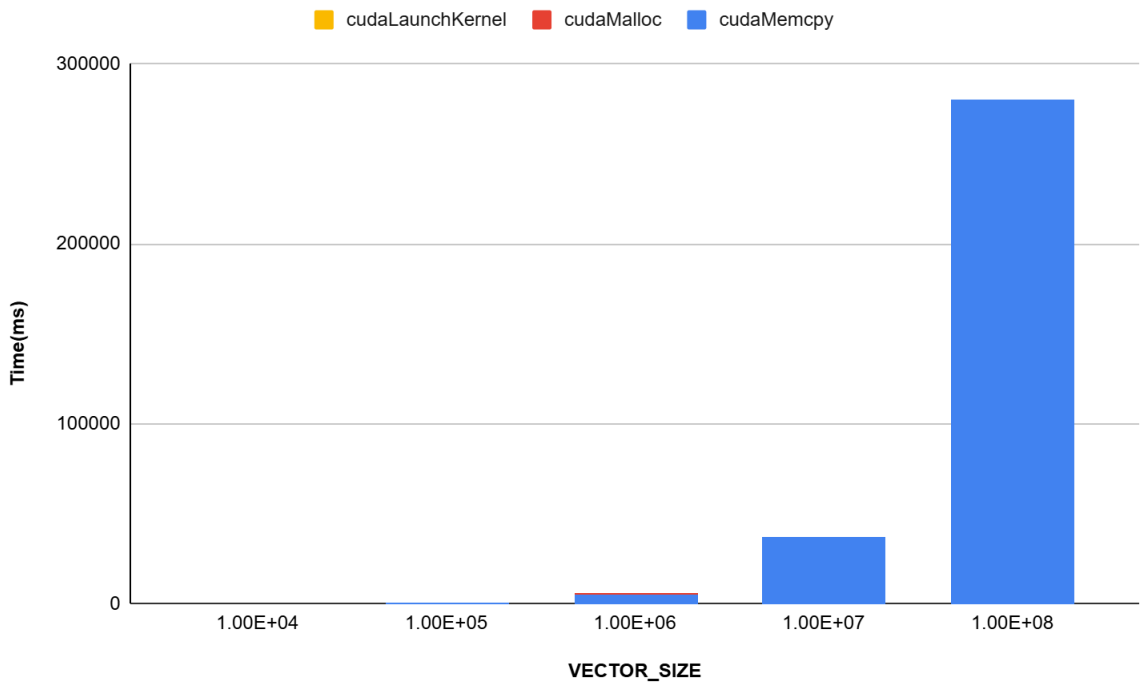
The **SAMPLE_SIZE** is iterated over 1e4 to 1e8 and the GPU activities and API calls are profiled.

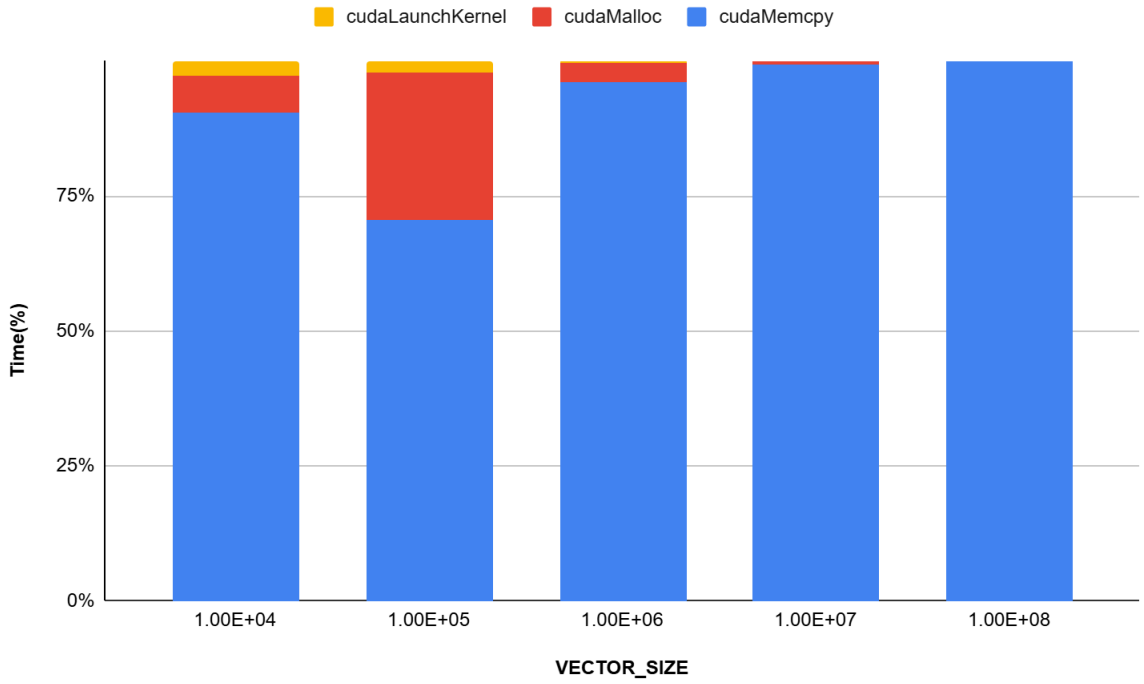
GPU Activities:

The GeneratePoints function accounts for approximately 99% of the execution time, regardless of changes in SampleSize, indicating it as the primary bottleneck, whereas, ReduceCounts and CUDA memcpy DtoH contribute less than 1% to the GPU activities.

API Calls:

Execution time of cudaMemcpy drastically increases when the SAMPLE_SIZE is increased. It starts from 90% with 140ms at SAMPLE_SIZE 1e4 to 99% with 279s at SAMPLE_SIZE 1e8.

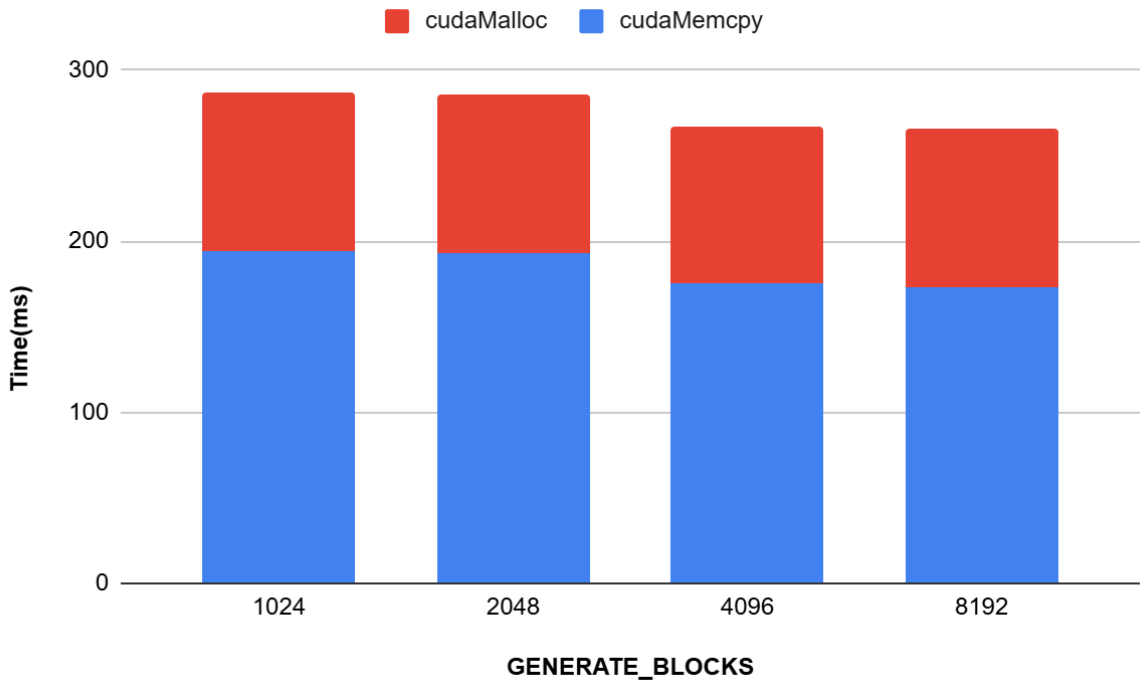




Experiment 2. Varying `GENERATE_BLOCKS`

The **GENERATE_BLOCKS** is iterated over 1e4 to 1e8 and the GPU activities and API calls are profiled.
The GeneratePoints function still accounts for approximately 99% of the execution time, regardless of changes in SampleSize

API Calls:
It can be noted that cudaMalloc and cudaMemcpy do not fluctuate much on changes in the GENERATE_BLOCK. They stay consistent at around 63% and 33% respectively.



Experiment 3. Varying `REDUCE_SIZE`

The **REDUCE_SIZE** is iterated over 32 to 512 and the GPU activities and API calls are profiled.

GPU Activities:
It is noted that as REDUCE_SIZE increases the percentage of execution time by reduceCount also increases. This can be attributed to the fact the each thread has to execute a bigger for loop. Contribution of execution time from reduceCount() increases from 0.02% with REDUCE_SIZE of 32 to 0.28% with REDUCE_SIZE of 512.