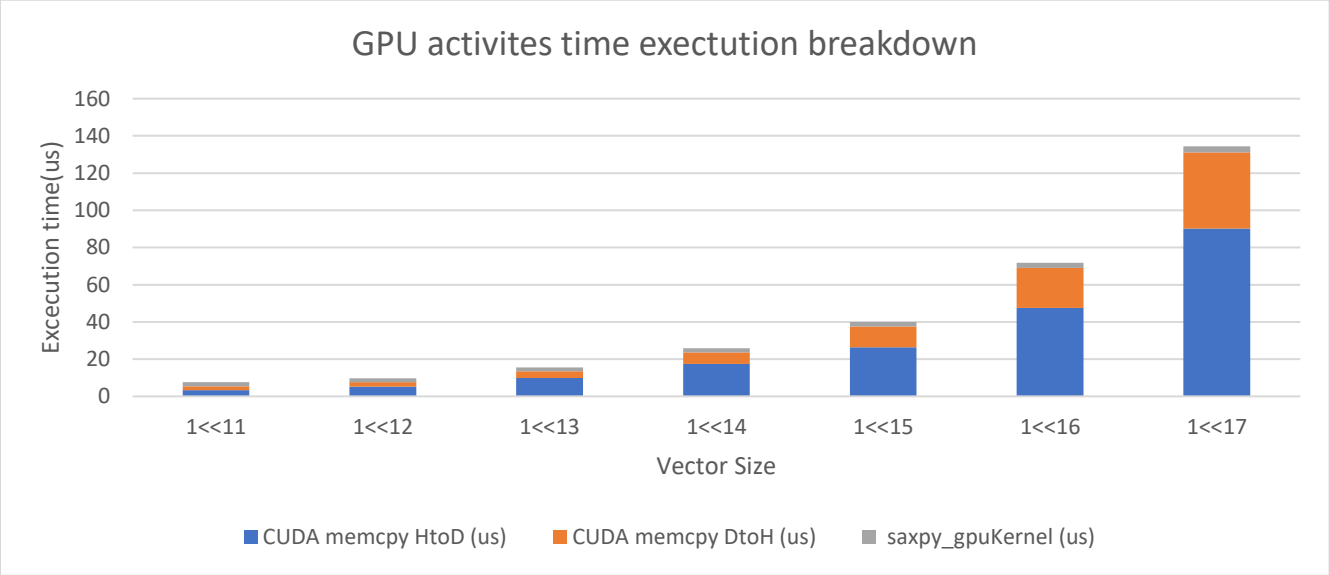
ECE60827 PA-1

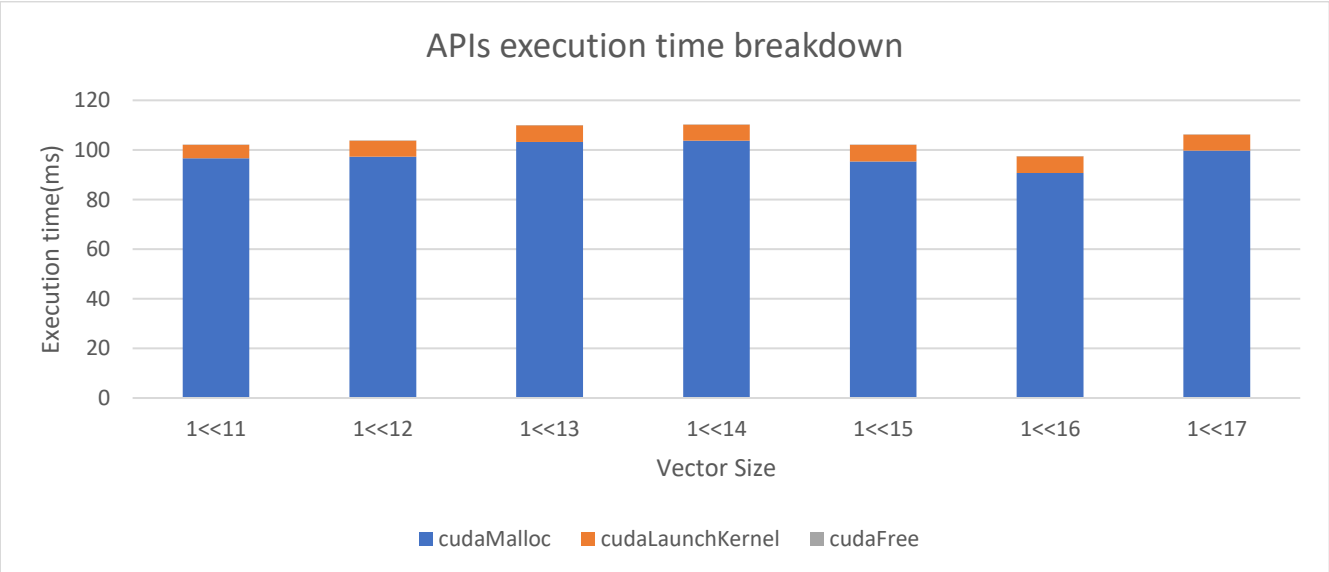
Jagdish Kurdiya (GitHub ID: jkurdiya)

Part A.

As we can see from the plot below that time required for data transfer (Host to device and Device to Host) increases with vector size because more data needs to be transferred through PCIe. Overall saxpy kernel execution time is less as it can perform vector operation efficiently. Thus, we can conclude that data transfer impacts overall performance and it slows down the operation.

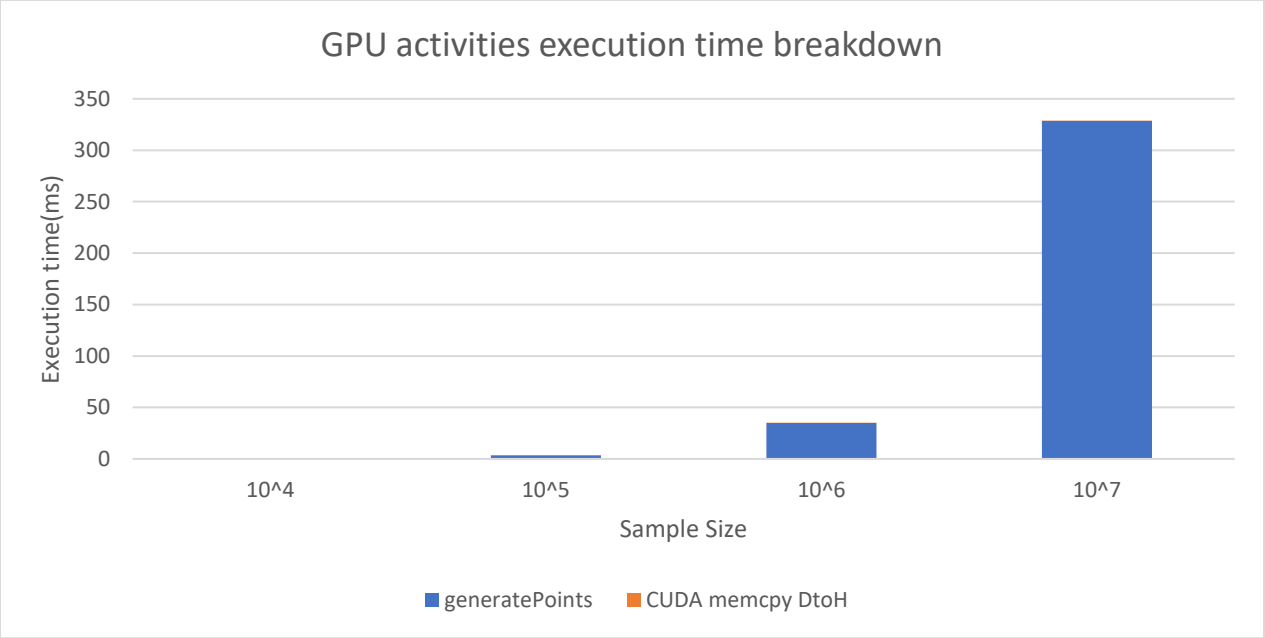


APIs – As we can observe that average time taken by APIs is around 100ms for all vector sizes. Vector sizes majorly affect execution time for cudaMalloc which is obvious for larger vector sizes.

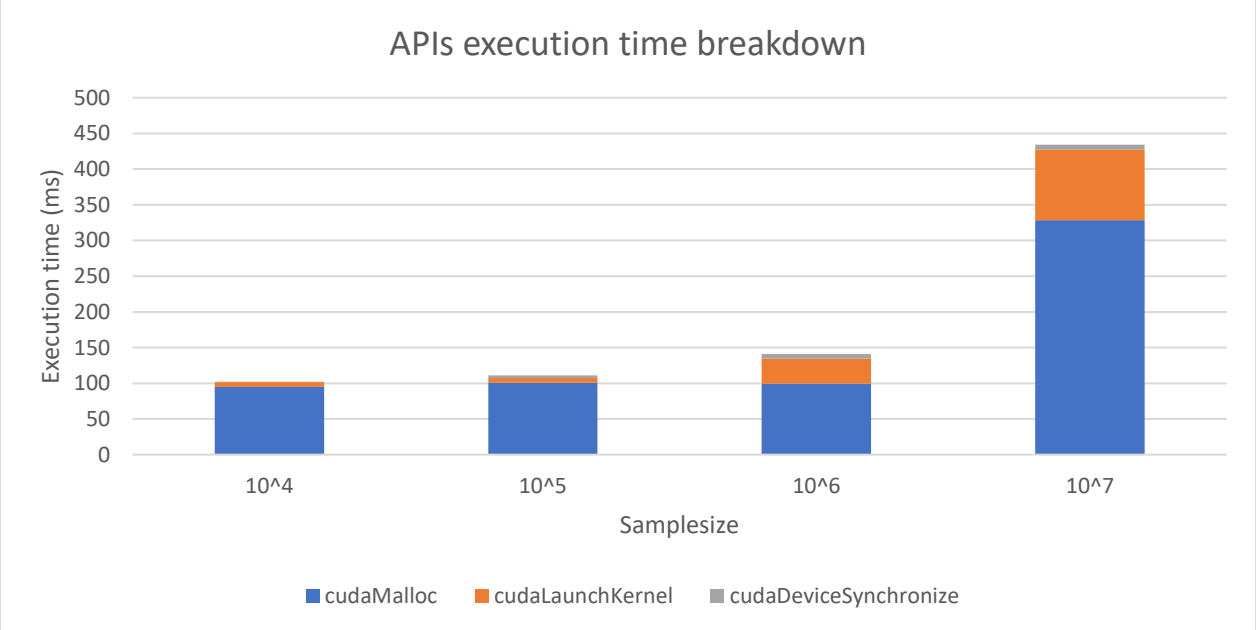


Part B

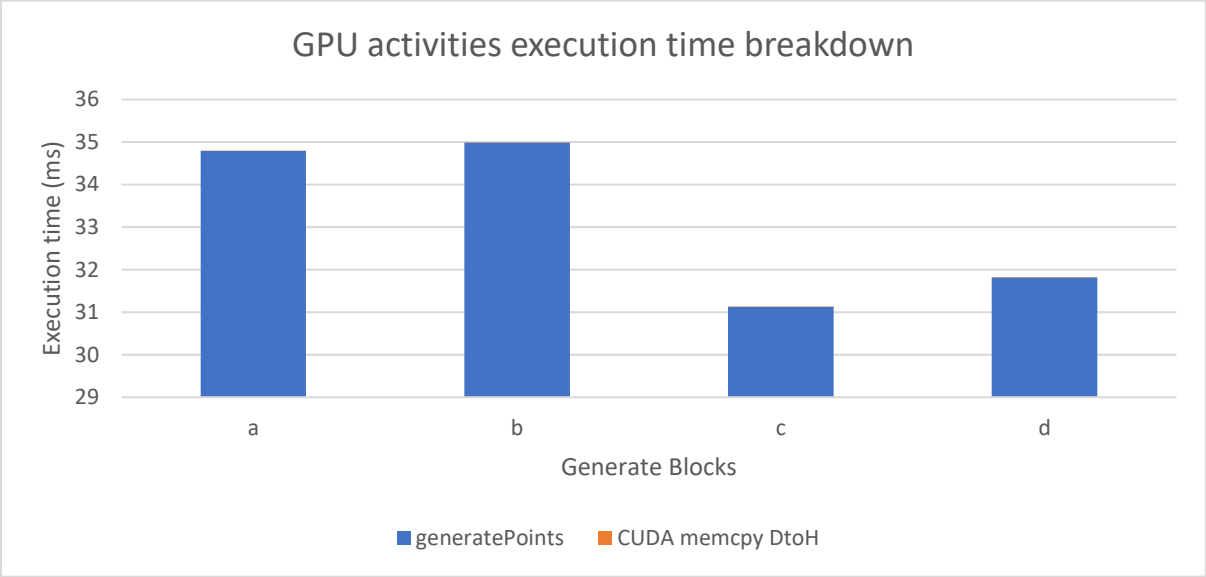
a) For different values of sample size
As is clearly seen from the plot below that execution time increases with sample size. Since Every thread has to generate points sequentially (not much parallelism per thread) from all the sample points the execution time increases rapidly. The execution time is noticeably high for larger sample sizes when compared with smaller sample sizes.



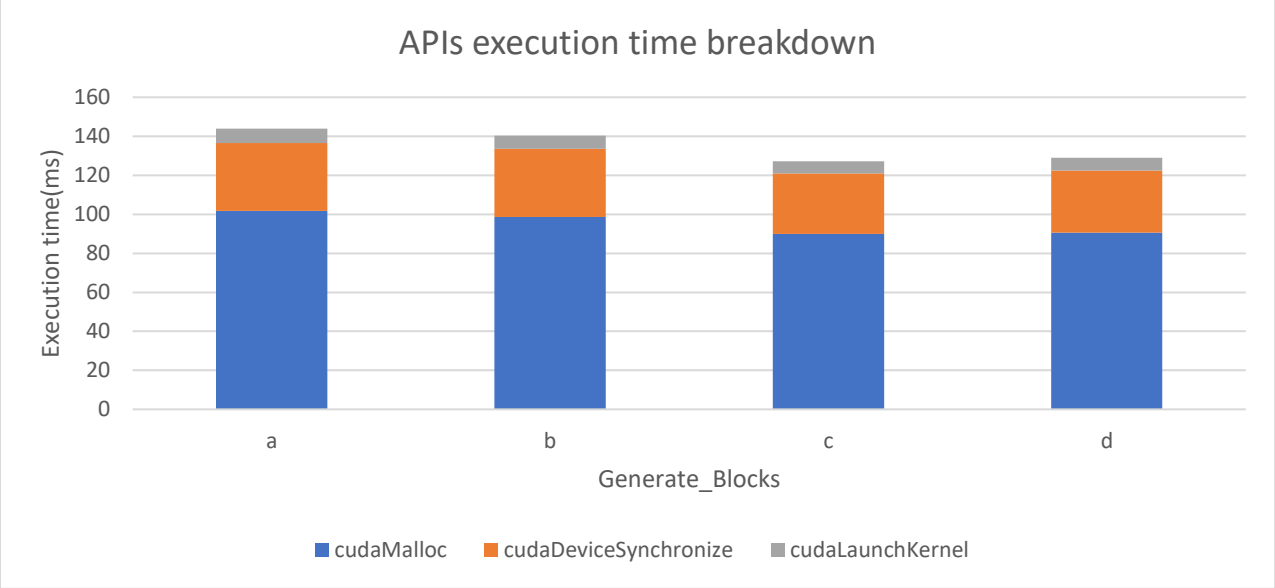
The analysis above is applicable for the execution time of APIs. It shows that memory allocation will be very crucial for larger sample sizes.



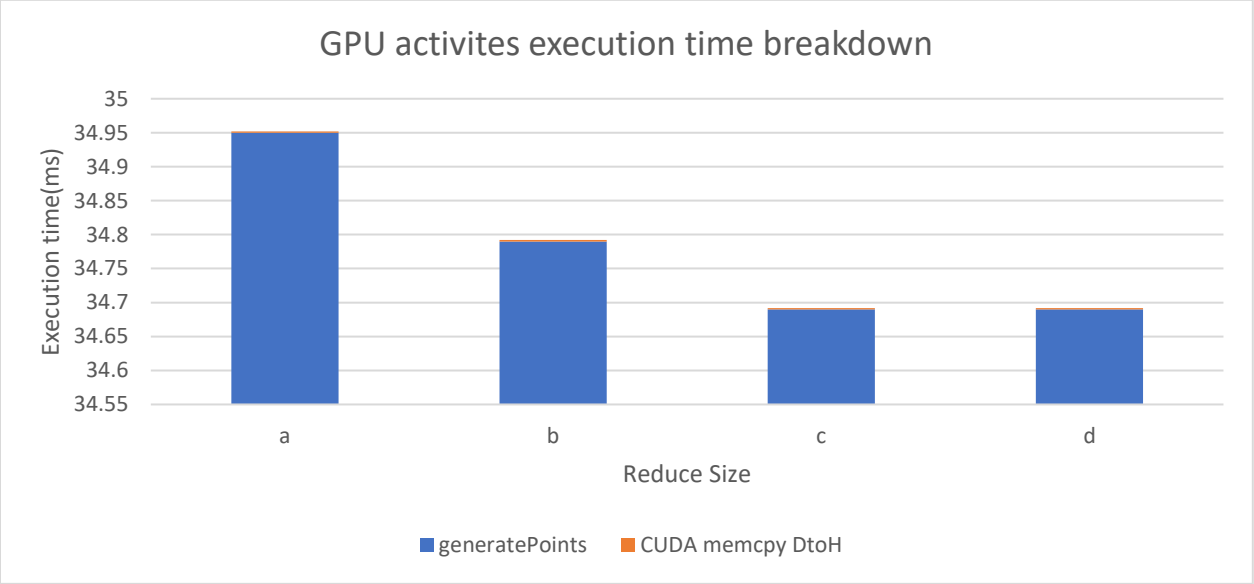
b) For different values of Generate Blocks(threads) where a, b, c, d are 1024, 2048, 3072, 4096 respectively
As it can be seen that majority of the execution time is taken by genratepoints kernel and very less time is required for the data transfer. As we increase the no of threads, utilizations of HW resources will improve and we can hide latency with excessive overlapped computation.



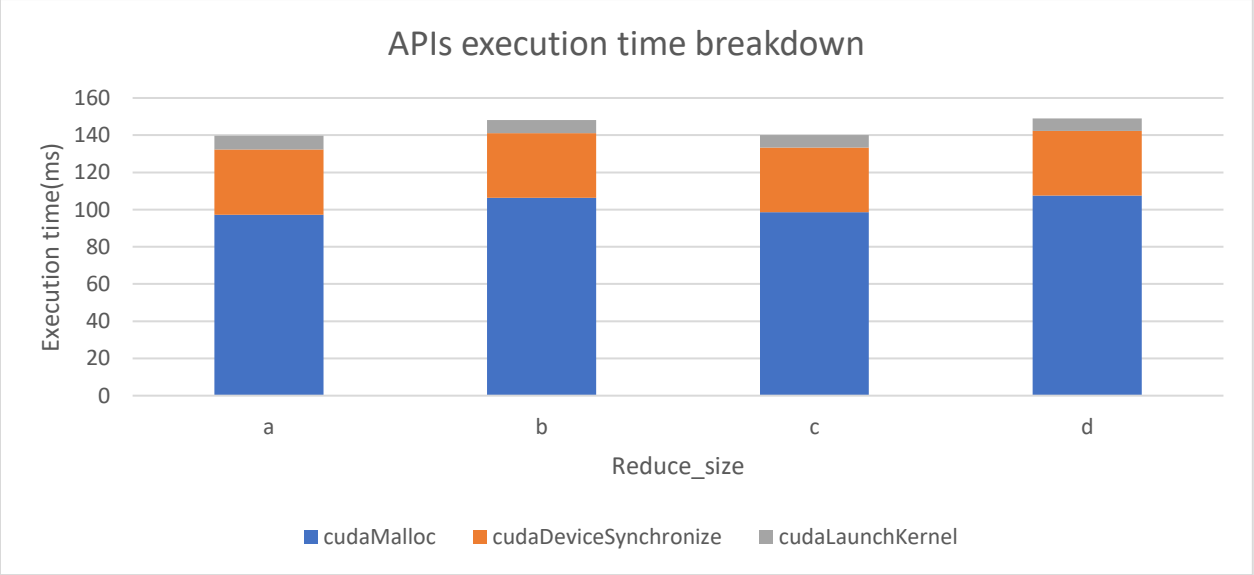
The average execution time required for the below APIs does not change much with respect to the number of threads. There is a slight change in cudaDeviceSynchronize' execution time. It might have happened due to more context switching or scheduling inefficiencies.



c) For different values of reduce size where a, b, c, d are 8, 16, 32, 64.
As it is evident from execution time on y-axis that, it does not change much when a thread's workload is changed.

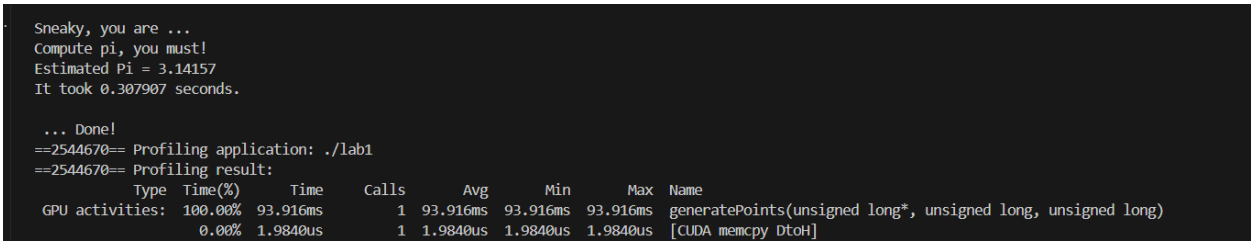


The time required for APIs execution remains stable here.

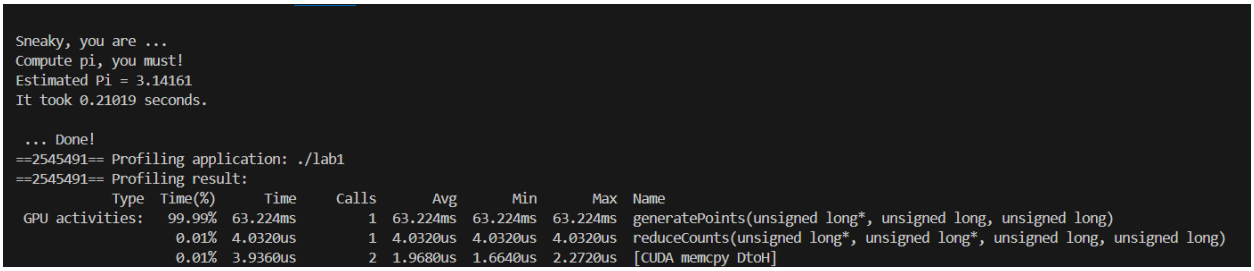


Optional part:

With only generatepoints kernel:



After implementation of reducecounts kernel:



Reducecounts kernel is more efficient as it uses GPU to reduce the partial products and then transfers the array to the CPU memory and it improves overall performance.