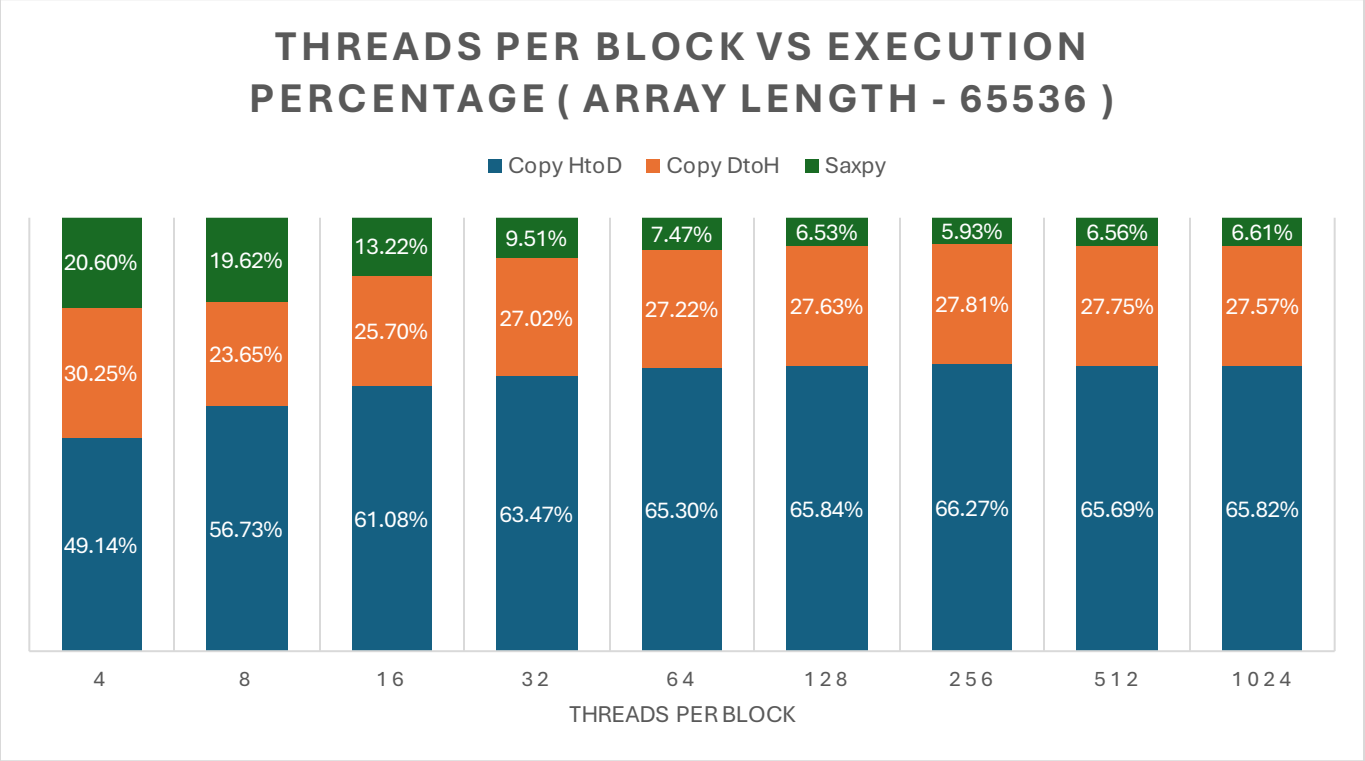
Name: Jacob Humphrey

Purdue Email: humphr31@purdue.edu

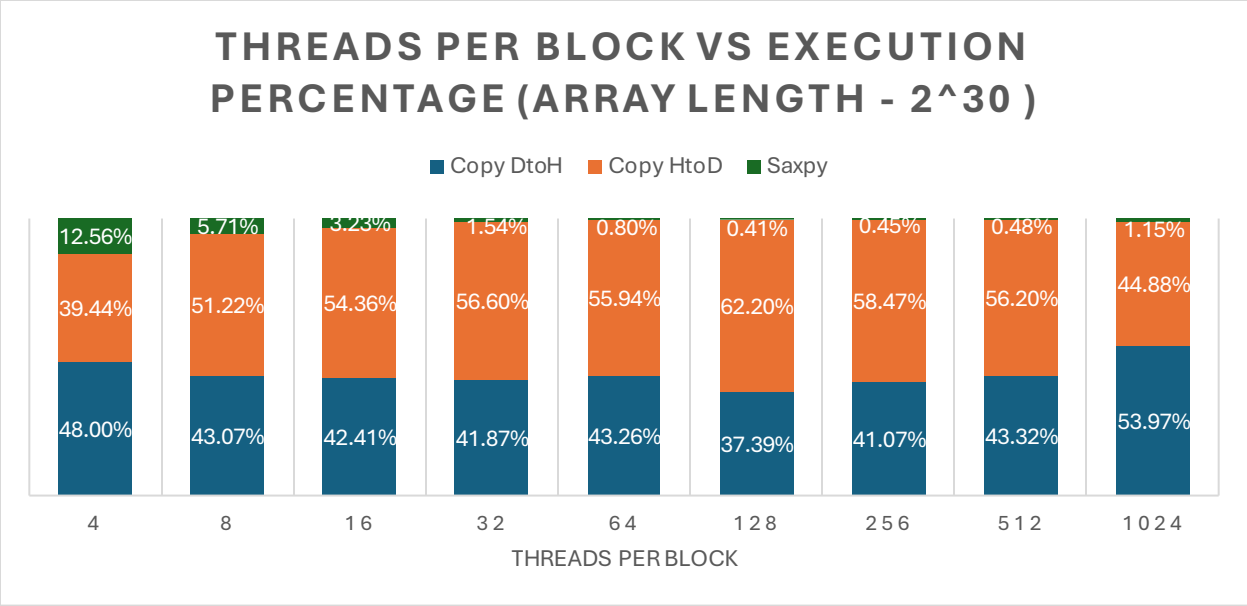
Github ID: humphr31

ECE 60827 Cuda Assignment 1 Report



Threads Per Block	4	8	16	32	64	128	256	512	1024
Total Time(us)	54.05	46.82	43.33	42.05	40.67	40.19	39.94	40.48	40.16

In the parallel execution of SAXBY with the default length of floats calculated we can see that the execution is largely taken up by copying the arrays between the Host and the Device. However, even with the relatively small dataset we can see a correlation between increasing the threads per block and decreasing the overall percentage of time in the SAXBY kernel as well as the total time.



Threads Per Block	4	8	16	32	64	128	256	512	1024
Total Time(s)	3.33	3.75	3.40	3.41	3.30	3.84	3.48	3.29	2.89

Increasing the amount of data that is computed, we can see just how memory bound this algorithm is. Even though the sample size has increased from 65536 values to 2^30. The percentage of execution is overwhelmingly taken up with the memory copying operations as the threads per block increase. This supports the proposition that you should use algorithms that are compute bound with high reuse as the hardware is obviously very capable of large amounts of quick computation if fed with enough data.

ThreadsPerBlock	4	8	16	32	64	128	256	512	1024
Copy DtoH	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.01%	0.00%	0.00%
GeneratePoints	100.00%	100.00%	100.00%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%
ReduceCounts	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%	0.00%
Seconds	277.517	138.713	69.7632	38.601	38.685	38.625	38.564	60.503	50.708

Moving onto the second algorithm for estimating π. This dataset was gathered when executing 2^24 threads with 10^6 datapoints each. All the iterations spent 99.99%+ of their time within the generatePoints kernels so we can conclude this that is likely a compute bound application. We can first see a trend of time reducing until it reaches 32 threads per block which is the same size as a warp. This is likely due to the structure and way that memory is delivered to each block, and causes a large slowdown when memory is trying to query below a block size of 32 threads. Once it reaches 512 threads per block it seems as if there is a different type of constraint that is hit. Likely inefficiencies with how many blocks can fit into a single SM.

Overall, with this assignment, the implementation of SAXBY shows how powerful a GPGPU can be to accelerate largely parallel applications compared to sequential CPU programs, but also shows the bottlenecks having to copy large amounts of data back and forth over data busses can be. Also, it shows how relatively small reuse of data really bottlenecks performance, as each of the datapoints was only used a single time. In the π estimation, we get to see how useful it is to generate the used data on the GPU rather than generating on CPU, then transferring. This results in a much higher utilization of time spent on computation rather than memory transfer.