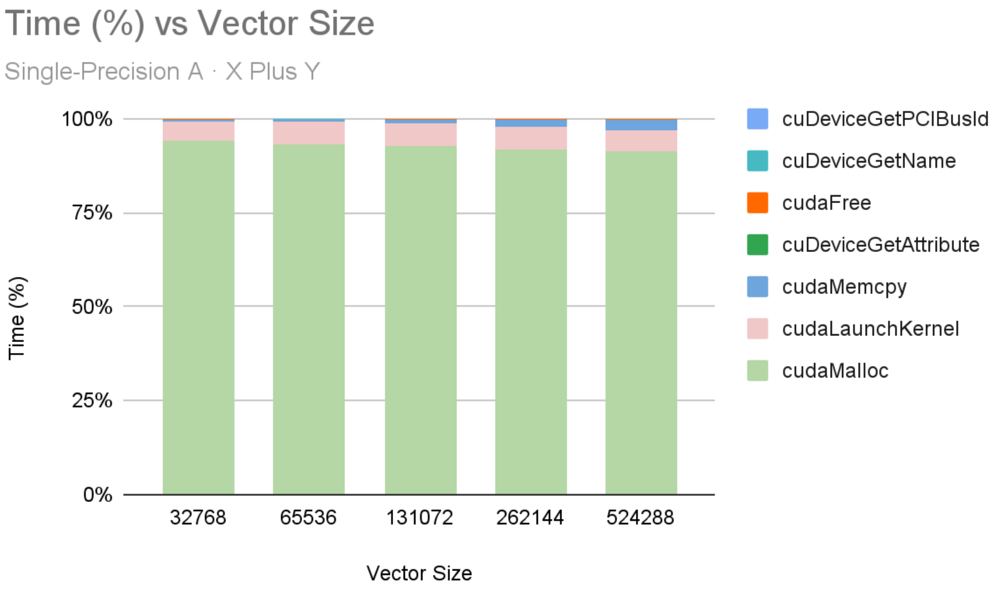
Henry Chu (chu298)
ECE-60827

CUDA Programming Part 1

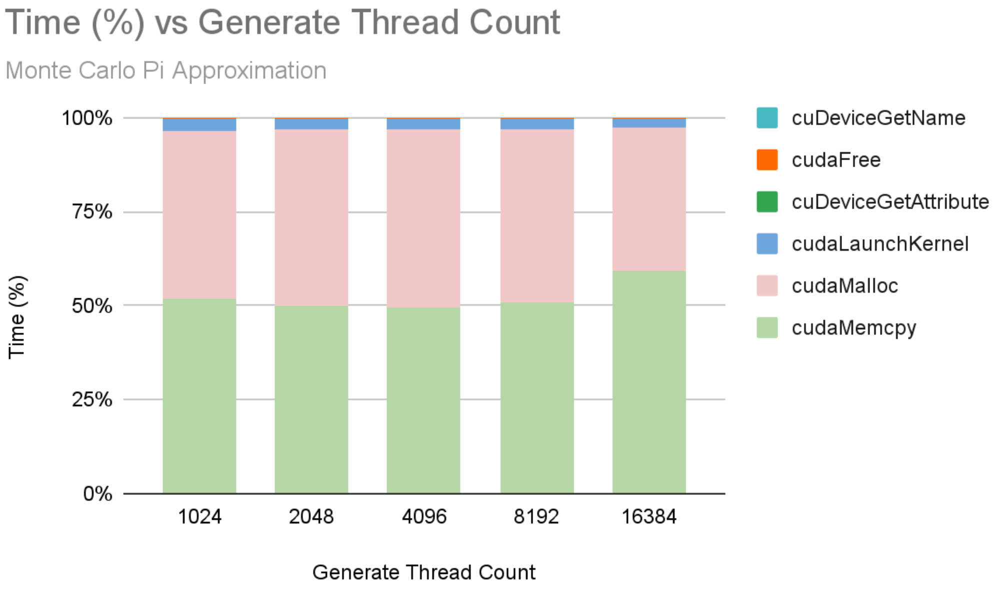
Part A:

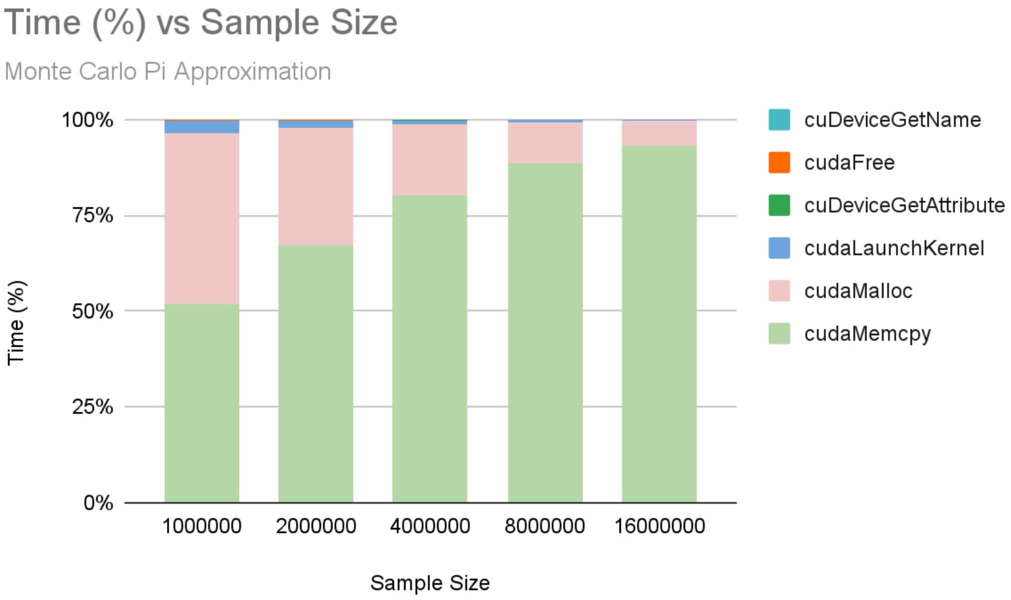
For the part of the lab, a Single-Precision A * X + Y (SAXPY) equation was implemented with variable vector sizes. The data for the below graph was taken from Nvidia's nvprof, a performance profiler used on our Lab1 executable. The API calls as a percentage of time are plotted against the size of the vector. Vector size was doubled every trial starting at 32768. The most visible trend was that cudaMemcpy's percentage of time also doubled with each subsequent trial. Thus, there is a linear trend between vector size and cudaMemcpy.



Part B:

For the second part of the lab, a Monte Carlo pi approximation was implemented. Randomly generated points in a unit square superimposed on a unit circle in Quadrant I were used to approximate pi. The points were within a 2 dimensional space {(0,1], (0, 1]}, where each (x,y) value fell within the unit square. The Pythagorean Theorem was used to determine if a point was within or outside of the circle. The total number of hits divided by the total number of points generated would approximate the area of a quarter of a unit circle (pi/4). Multiplying the area by 4 would thus yield an approximation of pi. Below are two figures showing API calls as a percentage of time plotted against the number of threads generated and the sample size, respectively. Isolating the number of threads generated, we do not see any significant trend. However, evaluating the sample size, you can see that cudaMemcpy increases logarithmically. Thus, sample size directly correlates to the execution time for cudaMemcpy.


Time (%) vs Generate Thread Count
Monte Carlo Pi Approximation
Time (%)
100%
75%
50%
25%
0%
1024
2048
4096
8192
16384
Generate Thread Count
cuDeviceGetName
cudaFree
cuDeviceGetAttribute
cudaLaunchKernel
cudaMalloc
cudaMemcpy


Time (%) vs Sample Size
Monte Carlo Pi Approximation
Time (%)
100%
75%
50%
25%
0%
1000000
2000000
4000000
8000000
16000000
Sample Size
cuDeviceGetName
cudaFree
cuDeviceGetAttribute
cudaLaunchKernel
cudaMalloc
cudaMemcpy