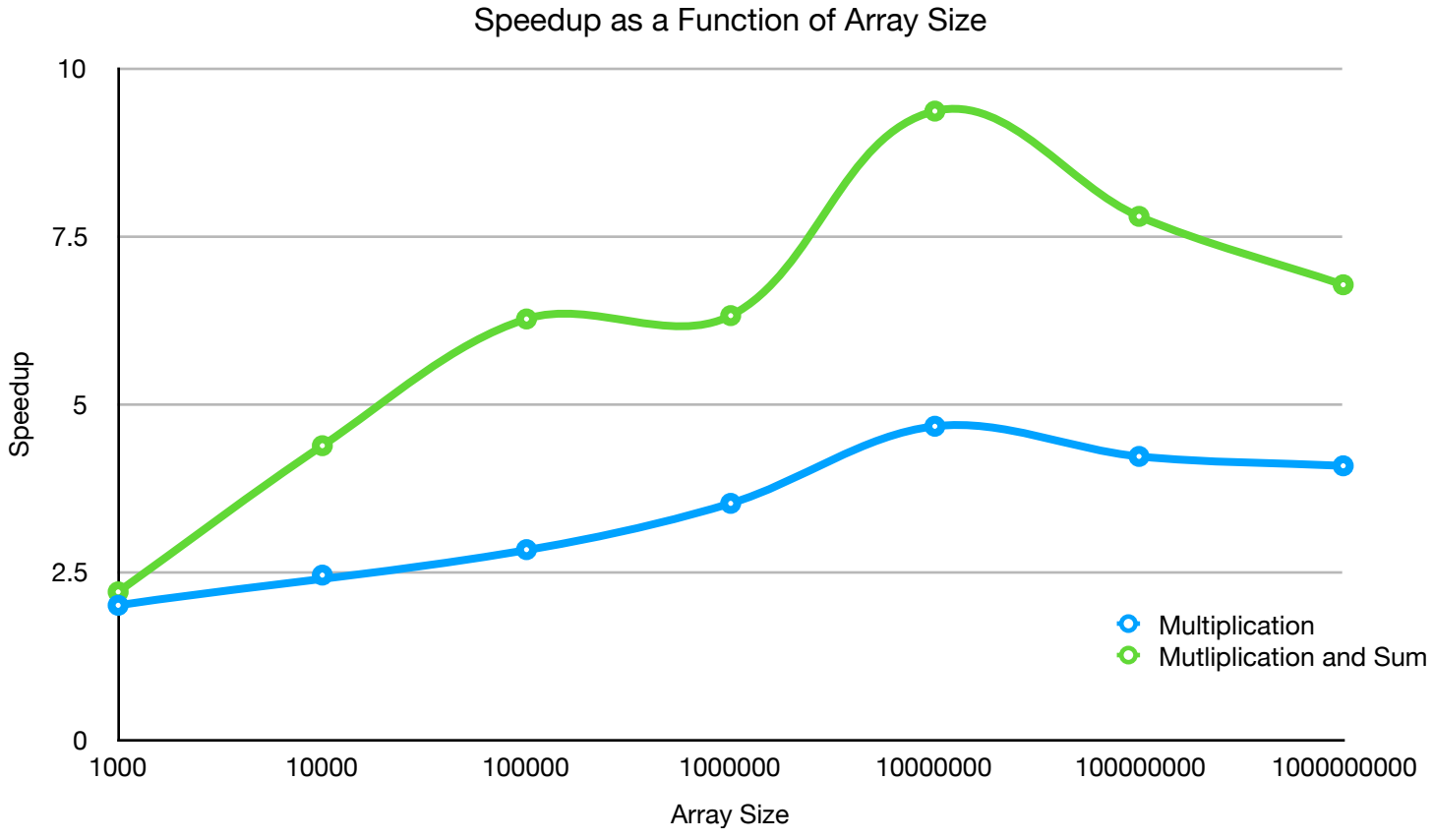
Aaron Berns
Project 5: Vectorized Array Multiplication

Machine
flip1

Results

Array Size	Multiplication Speedup	Multiplication and Sum Speedup
1000	2	2.2
10000	2.45	4.38
100000	2.83	6.27
1000000	3.52	6.32
10000000	4.67	9.37
100000000	4.22	7.8
1000000000	4.08	6.78



Discussion

The array size seems to have a significant effect on the speedup value. The smallest speedup is for the smallest array size. I would guess that even smaller array sizes would lower the speedup value further. The peak speedup for both the multiplication and multiplication + sum operations happens around an array size of 10,000,000. This is most likely because in order for the SIMD architecture to maximize the completion of operations over setup and data transfer overhead, the amount of time spent operating would need to be significant. Operating on several million values would boost the time spent working vs. preparing. The speedup value falls again as the array size gets larger than 10,000,000. This is most likely due to a lack of temporal coherence when caching the array values. They are only operated on once, therefore the caches are loaded and unloaded often.

In the array multiplication comparison, the speedup values ranged from 2 on the low side to 4.67 on the high side. Since four times the amount of data is being operated on when using SIMD, the expected value would be around 4.0. My best guess for why the speedup would be less than 4.0 with smaller array sizes has to do with the overhead of loading and unloading caches as well as other non-parallel tasks within the process. The ratio of time spent calculating vs. time spent doing other logistical tasks would be lower with less data to process. This could also be the reason why the speedup surpassed 4.0 in some of the larger array sizes. The ratio of time spent operating vs. time spend doing other things would favor time spent operating as the large array keep the data flowing. This has echoes of Gustafson's observation: more data increases the parallel fraction.

The range in speedup values was even greater in the multiplication/sum operation with values between 2.2 and 9.4. My best guess here is that the same factors were are play as in the multiplication operation. These effects were exaggerated by the fact that instead of loading the product values into another large array, which would lead to caching issues affecting performance, the product values are simply add to the sum value. This takes advantage of temporal coherence as the sum never leaves the cache.