Bhanu Prasanth Konda
934403560
kondab@oregonstate.edu

CS 475/575 -- Spring Quarter 2022

Project #4

Vectorized Array Multiplication/Reduction using SSE

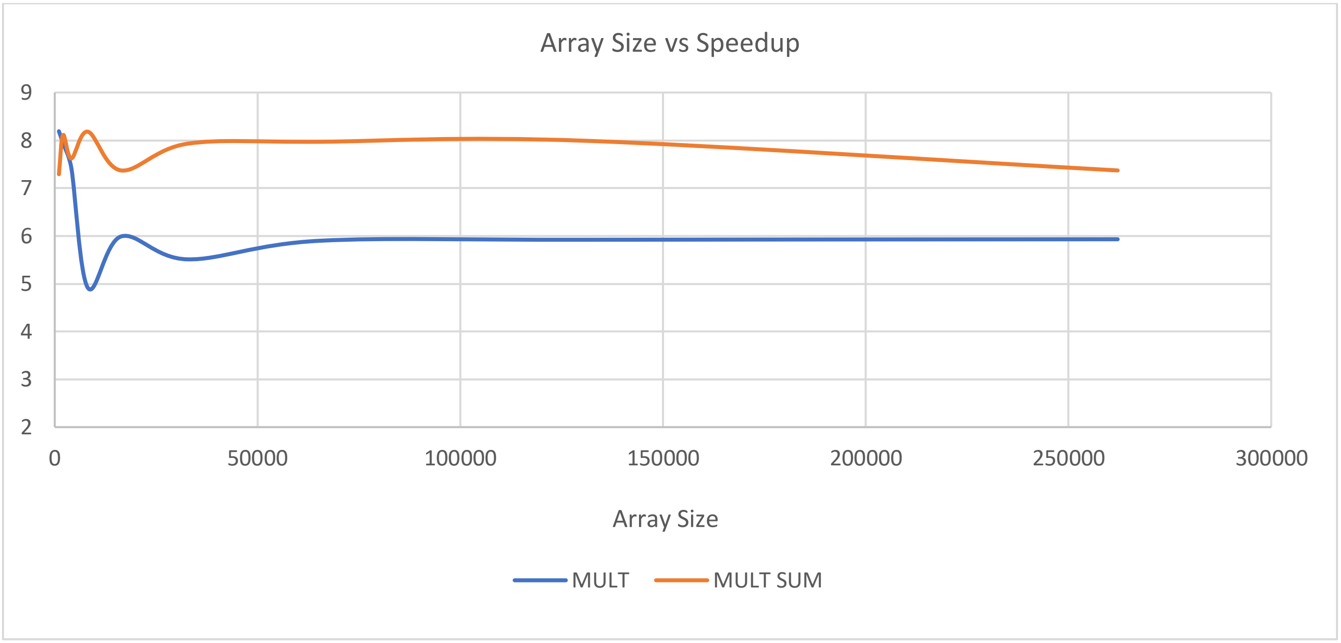
1. What machine you ran this on

   I have used the flip server

2. Show the table of performances for each array size and the corresponding speedups

ArrSize	NON-SIMD Mul	SIMD	MULT	NON-SIMD Mul Sum	SIMD Mul Sum	MULT SUM
1024	221.65	1815.87	8.19	223.52	1630.11	7.29
2048	167.55	1329.92	7.94	173.37	1406.47	8.11
4096	166.94	1238.89	7.42	173.39	1321.33	7.62
8192	207.57	1018.89	4.91	225.8	1846.37	8.18
16384	196.76	1177.6	5.99	176.98	1304.04	7.37
32768	184.99	1019.73	5.51	225.72	1789.05	7.93
65536	221.67	1308.63	5.9	225.78	1798.61	7.97
131072	221.25	1309.65	5.92	225.82	1804.19	7.99
262144	219.07	1298.93	5.93	224.63	1655.95	7.37

3. Show the graph of SIMD/non-SIMD speedup versus array size (either one graph with two curves, or two graphs each with one curve)



Bhanu Prasanth Konda
934403560
kondab@oregonstate.edu

4. What patterns are you seeing in the speedups?

   Based on the graph, array multiplication sum remains constant throughout the array sizes but when coming to array multiplication drops a bit initially but remained constant.

5. Are they consistent across a variety of array sizes?

   Matrix multiplication sum was constant all across, but Matrix multiplication was constant initially and started decreasing between 1000000 to 5000000 and came to a constant state after 5000000

6. Why or why not, do you think?

   The dip in the array multiplication might be due to the constant right to the array location, whereas in case of array multiplication sum all the values are multiplied and added to the local variable which might be stored in the cache as it is being re-used constantly and hence the constant performance.