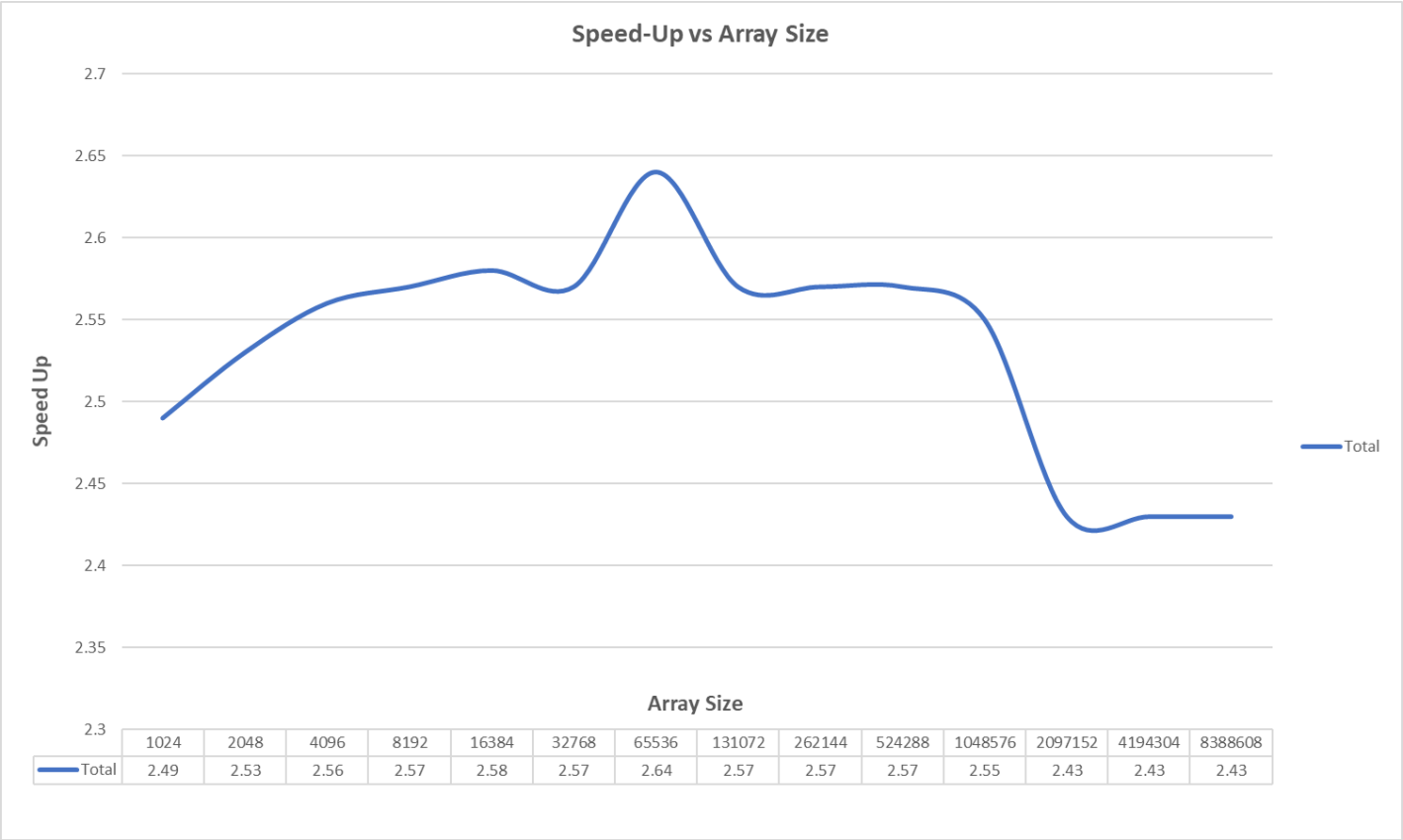
Clinton Hawkes
hawkesc@oregonstate.edu
CS475-400
Project #4

1. This was run on my personal laptop:
   Lenovo X1 Extreme
   Core i7 9750H (6 cores/12 threads)
   32GB ram
   Nvidia GTX 1650
   Running Linux kernel 5.5.9

2. Here is the table of performance figures I obtained.

Array Size	SIMD Performance	No SIMD Performance	Speedup:
1024	1122.81	450.31	2.49
2048	1175.66	463.87	2.53
4096	1186.56	464.19	2.56
8192	1189.83	463.58	2.57
16384	1194.26	463.49	2.58
32768	1187.89	462.15	2.57
65536	1190.03	418.91	2.64
131072	1192.55	463.62	2.57
262144	1188.77	461.83	2.57
524288	1164.43	452.59	2.57
1048576	1143.51	448.15	2.55
2097152	1074.01	442.36	2.43
4194304	1123.85	463.25	2.43
8388608	1161.08	476.85	2.43

3. Graph of SIMD/NON-SIMD speed-up:



	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288	1048576	2097152	4194304	8388608
Total	2.49	2.53	2.56	2.57	2.58	2.57	2.64	2.57	2.57	2.57	2.55	2.43	2.43	2.43

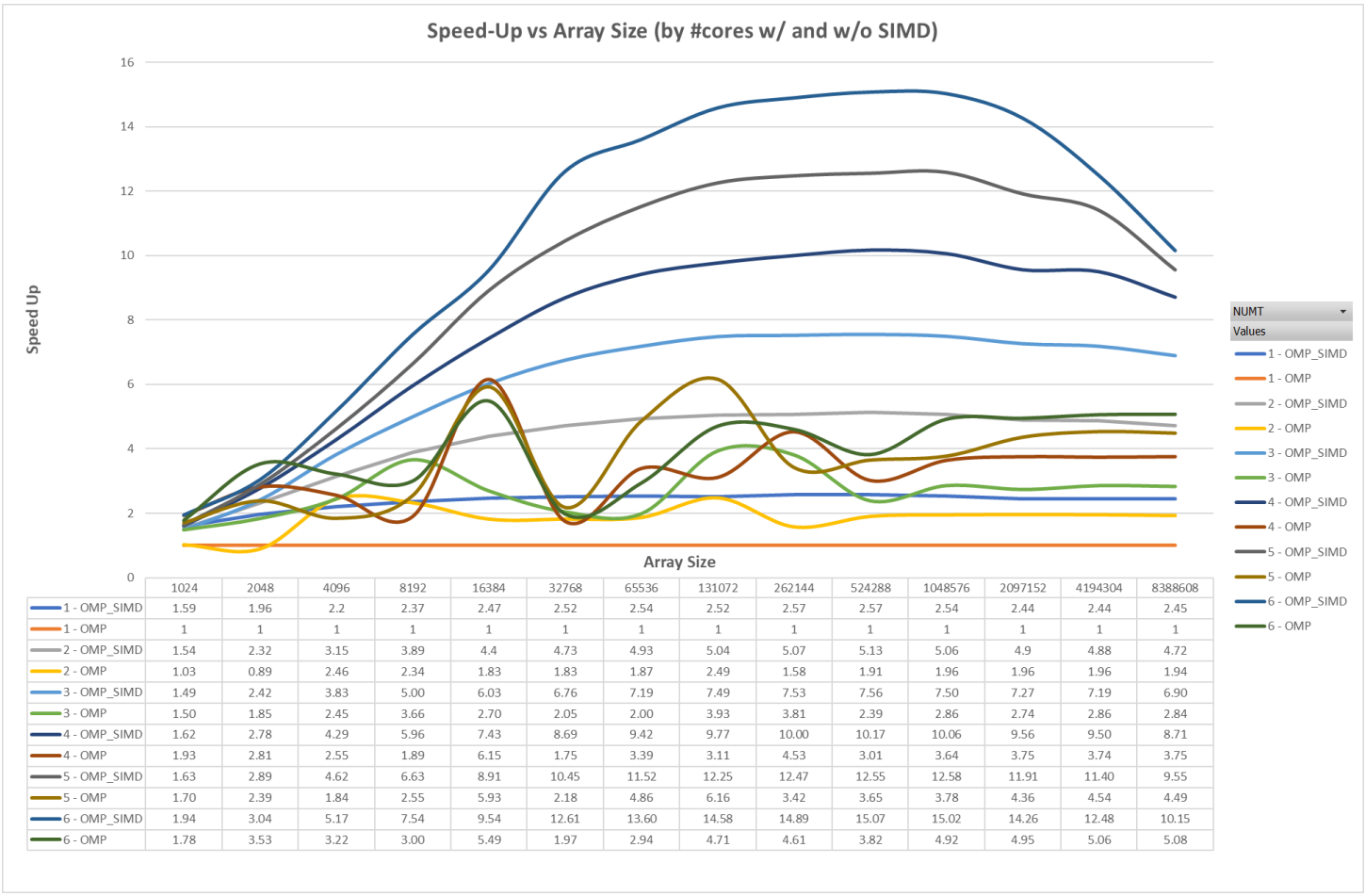
4. The pattern I'm seeing in the speedup is it starts low, rises up and peaks when array size is 65536, and then decreases and levels off at a speedup of 2.43. I think the graph looks a little weird with its abrupt peak and sharp drop-off, but I ran the simulation several times, and there was always this type of variation in speedups.

5. The speed ups are not consistent across the different array sizes. I was expecting to get similar results to the lecture slides. That was not the case for me.
    a. First, my max speedup for SIMD using one core was 2.64. The lectures showed a speedup of almost 4, so this threw me off a bit. I read on Piazza that most of the student's test produces figures similar to mine, so it may just be that the simulation that was used in the lectures was different from the simulation we ran.
    b. Speedup for a small array size (1024) was 2.49. The speedup slowly increases as the size of the input array increases to 65536, which is where the speedup peaks. As the input array size increases beyond 65536, the speedup declines and appears to level off at around 2.43.
    c. The difference in speedup between the lowest and highest is .21, which is not that impactful unless you are working with a ton of data. Given that SIMD is "free" performance that takes minimal time to implement, I think you should always use SIMD if your calculations allow. Even if the speedup decrease as the dataset grows larger, you will still benefit from using SIMD.

6. Since there is a little overhead of calling the function to multiply and sum the two array values using SIMD, I suspect that is why there is a reduced speedup when arrays are small. There is less time to "catch up" when arrays are small.
   We have learned that performance begins to decrease as the dataset size grows large, due to getting values from memory off the chip. We are seeing this in the graph above. Once the array size grows to be over one million, the performance (speedup) drops.

7. Extra Credit
   |
   |
   |
   |
   |
   |
   |
   |
   V
   |
   |
   |
   |
   |
   |
   V
   |
   |
   |
   |
   |
   |
   |
   V
   |
   |
   |
   |
   |
   |
   |
   V
   |
   |
   |
   |
   |
   |
   |
   V

Cores	Array Size	SIMD w/ OMP	OMP	Speedup_OMP_SIMD	Speedup_OMP
1	1024	769.93	321.1	1.59	1.00
1	2048	931.33	297.46	1.96	1.00
1	4096	1044.1	312.43	2.2	1.00
1	8192	1148.46	289.46	2.37	1.00
1	16384	1195.48	228.14	2.47	1.00
1	32768	1193.13	481.49	2.52	1.00
1	65536	1202.5	406.7	2.54	1.00
1	131072	1214.56	308.51	2.52	1.00
1	262144	1218.24	319.84	2.57	1.00
1	524288	1276.72	496.57	2.57	1.00
1	1048576	1255.25	495.65	2.54	1.00
1	2097152	1180.24	482.84	2.44	1.00
1	4194304	1172.24	479.94	2.44	1.00
1	8388608	1174.77	478.82	2.45	1.00
2	1024	745.27	329.37	1.54	1.03
2	2048	1104.64	265.56	2.32	0.89
2	4096	1497.08	767.9	3.15	2.46
2	8192	1845.88	675.96	3.89	2.34
2	16384	2081.57	416.44	4.4	1.83
2	32768	2241.93	879.37	4.73	1.83
2	65536	2331.16	762.51	4.93	1.87
2	131072	2384.17	767.66	5.04	2.49
2	262144	2429.53	506.18	5.07	1.58
2	524288	2483.61	948.2	5.13	1.91
2	1048576	2442.74	969.31	5.06	1.96
2	2097152	2314.06	947.9	4.9	1.96
2	4194304	2283.31	940.74	4.88	1.96
2	8388608	2225.52	926.76	4.72	1.94
3	1024	703.3	480.98	1.49	1.50
3	2048	1122.19	550.09	2.42	1.85
3	4096	1818.02	766.47	3.83	2.45
3	8192	2368.31	1060.45	5	3.66
3	16384	2856.35	616.13	6.03	2.70
3	32768	3200	988.33	6.76	2.05
3	65536	3414.76	811.77	7.19	2.00
3	131072	3555.17	1213.38	7.49	3.93
3	262144	3575.83	1219.55	7.53	3.81
3	524288	3586.86	1188.6	7.56	2.39
3	1048576	3548.78	1419.42	7.5	2.86
3	2097152	3378.26	1324.83	7.27	2.74
3	4194304	3287.35	1374.23	7.19	2.86
3	8388608	3214.06	1358.42	6.9	2.84
4	1024	750.18	618.73	1.62	1.93
4	2048	1258.76	836.94	2.78	2.81
4	4096	1987.38	798.13	4.29	2.55
4	8192	2695.62	548.22	5.96	1.89
4	16384	3436.24	1403.82	7.43	6.15
4	32768	4028.52	840.44	8.69	1.75
4	65536	4365.57	1377.82	9.42	3.39
4	131072	4528.94	958.64	9.77	3.11
4	262144	4634.55	1447.72	10	4.53
4	524288	4717.45	1496.72	10.17	3.01
4	1048576	4648.19	1803.64	10.06	3.64
4	2097152	4342.12	1810.86	9.56	3.75
4	4194304	4256.07	1793.19	9.5	3.74
4	8388608	3990.84	1797.28	8.71	3.75
5	1024	734.58	547.3	1.63	1.70
5	2048	1308.63	710.37	2.89	2.39
5	4096	2093	576.41	4.62	1.84
5	8192	2996.34	737.75	6.63	2.55
5	16384	4032.49	1352.71	8.91	5.93
5	32768	4731.16	1050.16	10.45	2.18
5	65536	5212.44	1978.33	11.52	4.86
5	131072	5545.44	1901.69	12.25	6.16
5	262144	5643.94	1095.17	12.47	3.42
5	524288	5681.49	1814.24	12.55	3.65
5	1048576	5664.12	1871.42	12.58	3.78
5	2097152	5286.04	2106.79	11.91	4.36
5	4194304	5000.08	2177.32	11.4	4.54
5	8388608	4307.6	2149.68	9.55	4.49
6	1024	854.76	570.16	1.94	1.78
6	2048	1211.83	1051.33	3.04	3.53
6	4096	2284.44	1004.66	5.17	3.22
6	8192	3324.68	868.07	7.54	3.00
6	16384	4205.34	1252.98	9.54	5.49
6	32768	5570.9	949.25	12.61	1.97
6	65536	6003.66	1194.6	13.6	2.94
6	131072	6440.57	1452.19	14.58	4.71
6	262144	6573.65	1474.22	14.89	4.61
6	524288	6655.43	1897.96	15.07	3.82
6	1048576	6616.08	2440.55	15.02	4.92
6	2097152	6173.23	2390.9	14.26	4.95
6	4194304	5365.98	2430.13	12.48	5.06
6	8388608	4541.86	2430.37	10.15	5.08



	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288	1048576	2097152	4194304	8388608
1 - OMP_SIMD	1.59	1.96	2.2	2.37	2.47	2.52	2.54	2.52	2.57	2.57	2.54	2.44	2.44	2.45
1 - OMP	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2 - OMP_SIMD	1.54	2.32	3.15	3.89	4.4	4.73	4.93	5.04	5.07	5.13	5.06	4.9	4.88	4.72
2 - OMP	1.03	0.89	2.46	2.34	1.83	1.83	1.87	2.49	1.58	1.91	1.96	1.96	1.96	1.94
3 - OMP_SIMD	1.49	2.42	3.83	5.00	6.03	6.76	7.19	7.49	7.53	7.56	7.50	7.27	7.19	6.90
3 - OMP	1.50	1.85	2.45	3.66	2.70	2.05	2.00	3.93	3.81	2.39	2.86	2.74	2.86	2.84
4 - OMP_SIMD	1.62	2.78	4.29	5.96	7.43	8.69	9.42	9.77	10.00	10.17	10.06	9.56	9.50	8.71
4 - OMP	1.93	2.81	2.55	1.89	6.15	1.75	3.39	3.11	4.53	3.01	3.64	3.75	3.74	3.75
5 - OMP_SIMD	1.63	2.89	4.62	6.63	8.91	10.45	11.52	12.25	12.47	12.55	12.58	11.91	11.40	9.55
5 - OMP	1.70	2.39	1.84	2.55	5.93	2.18	4.86	6.16	3.42	3.65	3.78	4.36	4.54	4.49
6 - OMP_SIMD	1.94	3.04	5.17	7.54	9.54	12.61	13.60	14.58	14.89	15.07	15.02	14.26	12.48	10.15
6 - OMP	1.78	3.53	3.22	3.00	5.49	1.97	2.94	4.71	4.61	3.82	4.92	4.95	5.06	5.08

Wow! Look at that speedup I got when using OMP and SIMD. I think it is amazing

As expected, when OMP was coupled with SIMD, the speedup what much greater. If you take a look at the graph in problem 3 above, you will see that we got a speedup of around 2.5 (estimated average overall). This speedup was from using SIMD alone, without OMP. So, it makes sense that when OMP and SIMD are used together, one can estimate that the likely speedup will be 2.5 times the speedup seen when using OMP alone. The graph above follows this logic. (roughly) Look at the curves representing 6 cores. When the array size is 262,144 the speedup of OMP alone is 4.61. When SIMD is added to the mix, the speedup for the same array size is 14.89. The speedup gained by utilizing SIMD is actually closer to 3 than the 2.5 estimated, but this was at the peak performance. As the array size increases, the performance gained over OMP alone nears 2.5.

You will notice that speedup for the SIMD w/ OMP curves peak when the array size is about 524,288. After this, when the arrays grow larger, it is apparent that the CPU has a difficult time retrieving the values from memory quick enough to keep up with the calculations. This becomes more and more evident as the speedup increases. If you look at the curve for 6 cores using OMP and SIMD, you can see the dramatic drop in speedup right after the array size of one million. It looks like speedups of around 5 do not cause this slowdown seen with the higher speedups.

The curves for the OMP only runs has a lot of sporadic behavior. I have no idea why the lines are so, seemingly, random. My guess is my computer had a background process that interfered with scheduling, which is why the speedups are all over the place. Interestingly though, the OMP w/ SIMD curves do not seem to be impacted.