Intro to Parallel Programming

Amine Gaizi
gaizia@oregonstate.edu

Project n°4 – Vectorized Array Multiplication/Reduction Using SSE

What machine was this run on ?

This project was ran on the university's flip servers. The load average at the moment of the execution was: 4,55, 4.80, 7,94.

Performance Table:

ArraySize/Computational Method	1 thread	2 threads	4 threads	8 threads	only SIMD	SIMD with 2 threads	SIMD with 4 threads	SIMD with 8 threads
1024	0,859823743	1,04337	1,709003	1,164921	2,17138414	1,519682046	2,41880076	1,243027475
2048	0,924828061	1,3783	2,389237	2,182393	2,19188446	2,17892022	3,518019257	2,670667125
4096	0,952443041	1,65423	3,014548	2,63296	2,20158523	2,951447982	5,246225638	4,118103212
8192	0,974874889	1,80171	3,247755	4,881093	2,20357167	3,559676424	5,598957976	7,234661
16384	0,987664051	1,89655	3,568276	3,559093	2,20693554	3,961518692	6,897680156	7,27255594
32768	0,991932438	1,96697	3,781592	6,818978	2,20234131	4,266332521	7,827113873	12,78924783
65536	0,992373983	1,97469	3,877163	5,347787	2,19841324	4,304151563	8,246426373	8,995143971
131072	0,997460274	1,98452	3,941655	7,65621	2,20626695	4,337474689	8,526306758	15,91526238
262144	0,995636189	1,99533	3,968319	7,810947	2,20667972	4,385321101	8,655190187	16,78864722
524288	0,998543437	2,01033	4,025975	7,987585	2,2111323	4,435963239	8,8273279	17,31995838
1048576	0,981469038	2,01542	4,101567	8,182976	2,19209959	4,47190296	8,994801429	17,90214662
2097152	0,999504074	2,02798	4,062735	8,034396	2,11349628	4,162628409	8,089868934	12,89277365
4194304	0,999575311	1,99713	3,994798	8,045336	2,11572763	3,963335221	6,98559598	9,781356172
8388608	1,002862365	1,94896	3,89092	7,752664	2,07731834	3,870193468	6,629271994	7,733558644

Table 1: Speedup values for various array sizes and computational methods

Array Sizes/Computational Method	No threads No SIMD	1 thread	2 threads	4 threads	8 threads	only SIMD	SIMD with 2 threads	SIMD with 4 threads	SIMD with 8 threads
1024	289,35	248,79	301,9	494,5	337,07	628,29	439,72	699,88	359,67
2048	290,8	268,94	400,81	694,79	634,64	637,4	633,63	1023,04	776,63
4096	291,44	277,58	482,11	878,56	767,35	641,63	860,17	1528,96	1200,18
8192	291,74	284,41	525,63	947,5	1424,01	642,87	1038,5	1633,44	2110,64
16384	291,83	288,23	553,47	1041,33	1038,65	644,05	1156,09	2012,95	2122,35
32768	291,29	288,94	572,96	1101,54	1986,3	641,52	1242,74	2279,96	3725,38
65536	292,42	290,19	577,44	1133,76	1563,8	642,86	1258,62	2411,42	2630,36
131072	291,37	290,63	578,23	1148,48	2230,79	642,84	1263,81	2484,31	4637,23
262144	291,03	289,76	580,7	1154,9	2273,22	642,21	1276,26	2518,92	4886
524288	288,35	287,93	579,68	1160,89	2303,22	637,58	1279,11	2545,36	4994,21
1048576	282,77	277,53	569,9	1159,8	2313,9	619,86	1264,52	2543,46	5062,19
2097152	282,3	282,16	572,5	1146,91	2268,11	596,64	1175,11	2283,77	3639,63
4194304	282,56	282,44	564,31	1128,77	2273,29	597,82	1119,88	1973,85	2763,82
8388608	289,97	290,8	565,14	1128,25	2248,04	602,36	1122,24	1922,29	2242,5

Table 2: Performance (in Mega Operation Per Second) values for various array sizes and computational methods

SIMD and Non SIMD Speedup VS Array Size Graphs:

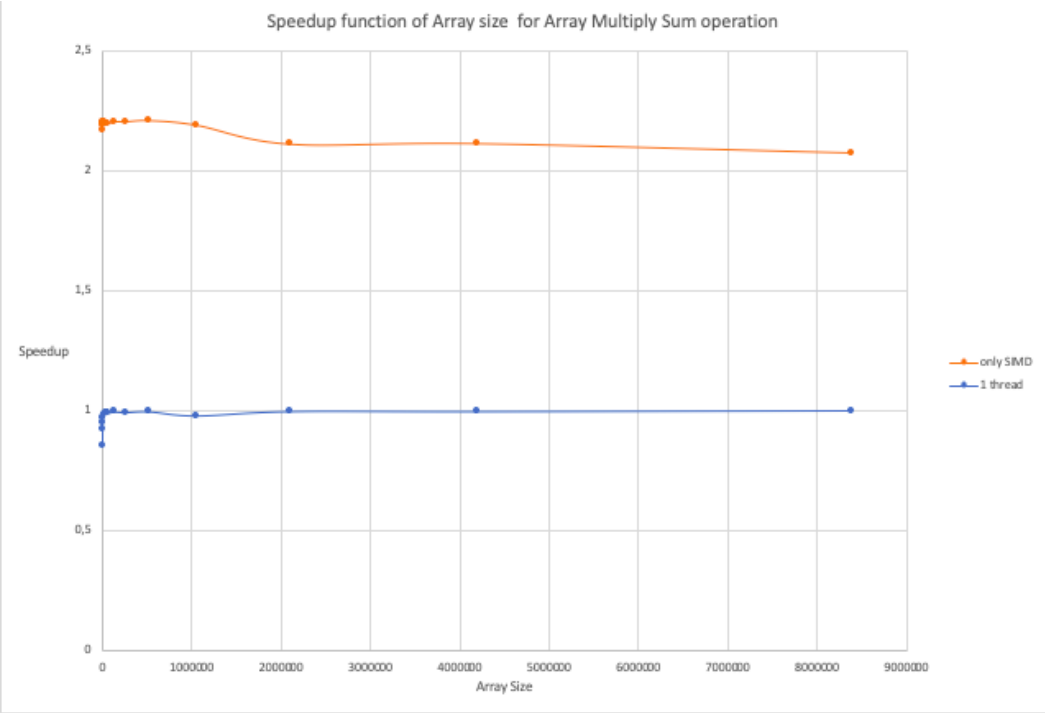

*Figure 1: SIMD and Non SIMD Speedup vs Array Size Curves**

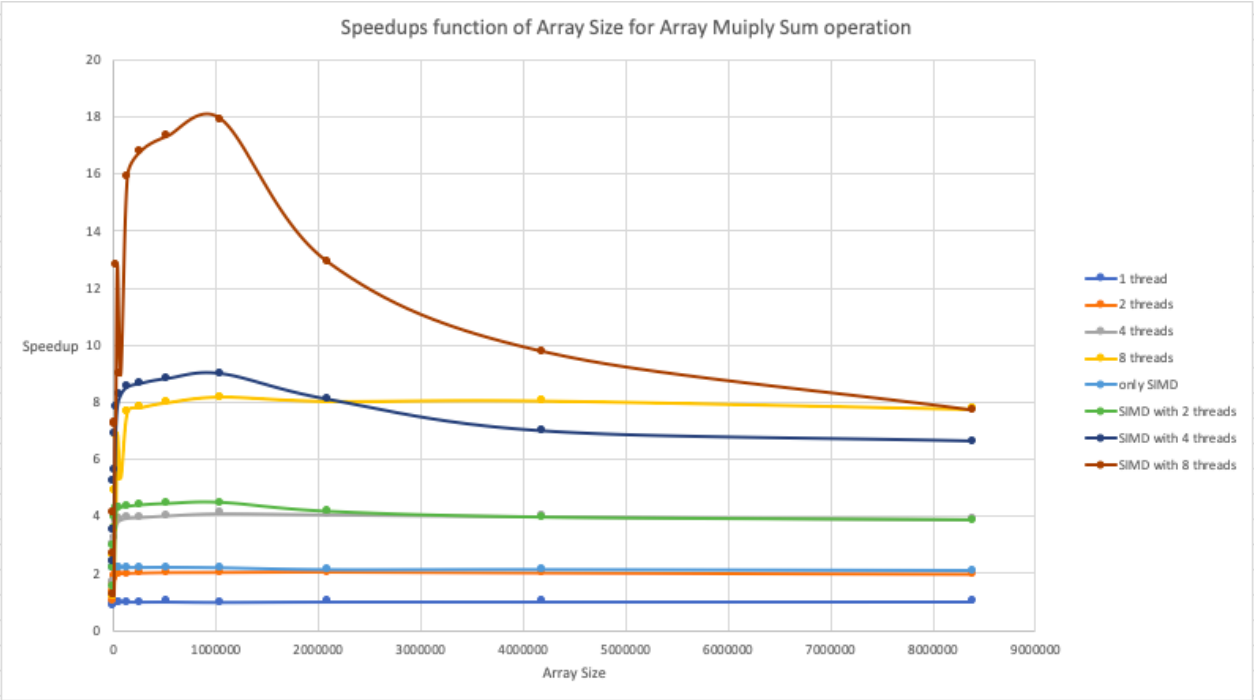

*Figure 2: Multithreading and SIMD Speedups vs Array Size Curves**

*: For a better visualization, the pictures of the graphs are available in the "Graphs" folder in the compressed folder of this project's submission.

<u>What patterns is visible in the speedups ? Are the patterns consistent across a variety of array sizes ? Why ?</u>

From figure 1 and figure 2, what we can observe is that the speedup when using SIMD (with and without multithreading) is always higher at small dataset size. It seems that the speedup increases until reaching a maximum value (usually around an array size of 1048576) and then slowly decreases and settles down.

The reason the speedup is higher with smaller dataset sizes is because we are using the intel intrinsics to program the SIMD operations. These instructions have a tighter coupling to the setting up of the registers. Therefore, a smaller setup time makes the smaller dataset size speedup look higher than when the dataset size increases.

Also, the speedup values seem to decrease and stabilize as the dataset size gets bigger (array size higher than 1048576). This might be due to the fact that with a larger dataset size and such a high performance, the operations are being performed faster than the memory access allows to fetch the data. Ultimately, the speedup value ends up stabilizing because the performance is bounded by the speed of loading data from cache to memory. According to the material taught in class, we could prevent the decrease in speedup by prefetching manually the data from cache to the memory before it is to be used.

Another trend we can observe when using SIMD with multithreading (it is very obvious for the SIMD + 8 threads) is that for low array sizes, the speedup dips to a lower value before increasing and reaching its maximum speed up. The decreasing trend is specific to the speedup because it is relative, the performance on the other hand keeps on increasing.

When it comes to the multithreading (no SIMD) speedups, they seem to increase with the dataset size until reaching a maximum speedup value (around the array size 131072) and stabilizing around that speedup. This behavior is the same as the one observed in previous projects. The performance increases with dataset sizes when using multiple threads.