Holly Straley
straleyh@oregonstate.edu
CS475 – Spring 2018
Project 5

Function Decomposition

1. Platform

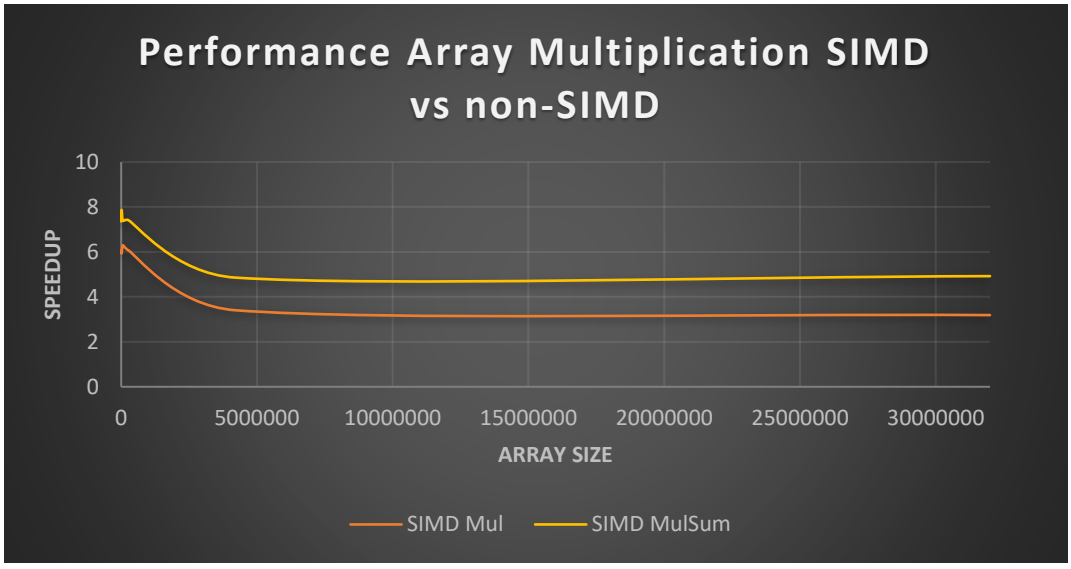
I ran this program on the flip1 server.

2. Performance Data

2.1 Table

Array Size	Non-SIMD type	Performance	SIMD type	Performance	Speedup
1024	Non-SIMD Mul	126.93	SIMD Mul	769.69	6.063893
16384	Non-SIMD Mul	124.95	SIMD Mul	740.79	5.928691
65536	Non-SIMD Mul	127.56	SIMD Mul	803.6	6.29978
262144	Non-SIMD Mul	127.11	SIMD Mul	772.05	6.073873
4194304	Non-SIMD Mul	230.89	SIMD Mul	786.54	3.406557
32000000	Non-SIMD Mul	229.89	SIMD Mul	733.31	3.18983
1024	Non-SIMD MulSum	131.05	SIMD MulSum	964.17	7.357268
16384	Non-SIMD MulSum	130.93	SIMD MulSum	1030.66	7.87184
65536	Non-SIMD MulSum	131.91	SIMD MulSum	973.43	7.379501
262144	Non-SIMD MulSum	131.66	SIMD MulSum	976.29	7.415236
4194304	Non-SIMD MulSum	240.76	SIMD MulSum	1169.82	4.858864
32000000	Non-SIMD MulSum	240.96	SIMD MulSum	1186.26	4.923058

2.2 Graph



3. Patterns

What patterns are you seeing in the speedups?

The speedup values for both the SIMD multiplication and the SIMD multiplication + reduction are the greatest with smaller array sizes then reduce and stabilize for larger array sizes. SIMD multiplication stabilizes around a speedup of 3 and SIMD multiplication + reduction stabilizes around a speedup of 5. The multiplication + reduction process follows a trend of consistently having a higher speedup by around 1.5X or higher.

Are the patterns you're seeing consistent across a variety of array sizes? Why or why not?

One pattern that is definitely consistent across a variety of array sizes is that the multiplication + reduction process has a higher speedup than the multiplication process. This is expected from reduction as we learned early in the quarter, doing reduction during multiplication will always make the process more efficient.

4. SSE SIMD Analysis

Knowing that SSE SIMD is 4-floats-at-a-time, why could you get a speed-up of < 4.0 or > 4.0 in the array multiplication? Knowing that SSE SIMD is 4-floats-at-a-time, why could you get a speed-up of < 4.0 or > 4.0 in the array multiplication-reduction?

We saw earlier this quarter that a speed-up of < 4.0 is likely caused by overhead from the functionality of the process including cache misses and set-up but a speedup > 4.0 is new. One reason for this is that the SIMD code in this experiment is written in assembly language and is 4-floats-at-a-time while the non-SIMD code (which is the baseline for the speedup calculation) is in C++, so I think this makes the caching fetching faster for the SIMD process. This increase in efficiency is not factored into the calculations for speedup so you end up with a speedup higher than 4.0.