Josh Huff, huffj@oregonstate.edu
Professor Bailey
CS 475 – Spring 2017
May 25, 2017

Project 5: Commentary

---

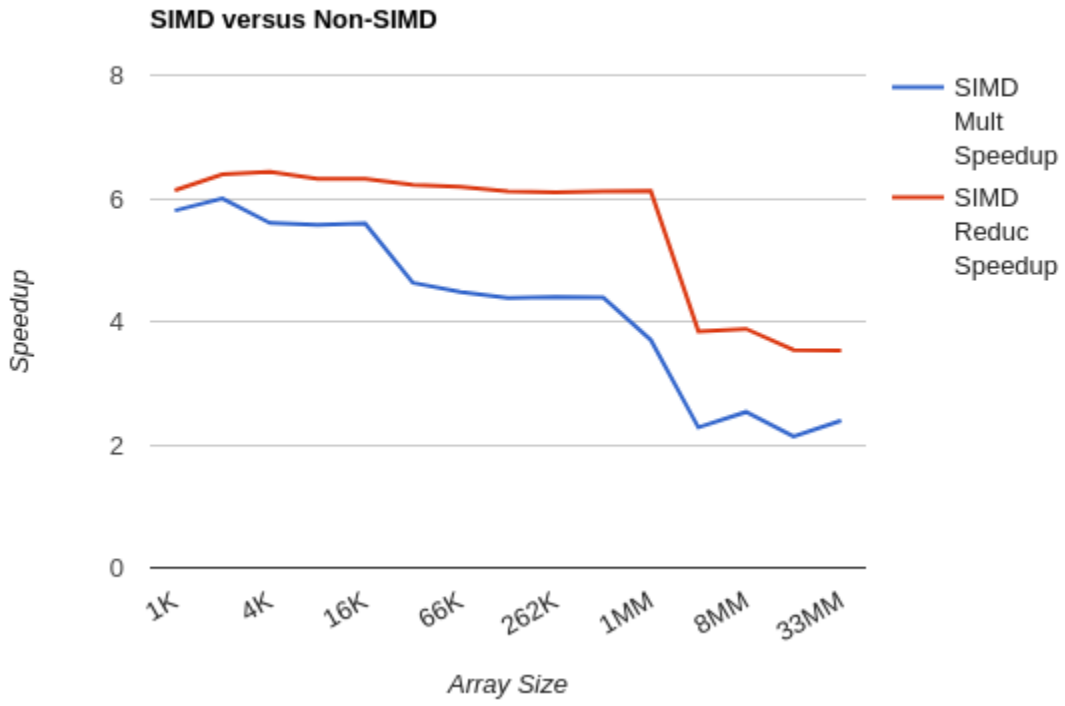
1. What machine you ran this on

My main workhorse, a home-built machine running Ubuntu 16.04 LTS (64-bit). The processor is Intel® Core™ i5-4690K CPU @ 3.50GHz × 4 and it has 15.6 GiB of usable RAM.

The difference between peak and average performance was minimal, and the load averages were very low (less than 1) across the board. However, per Professor Bailey's warning, the SSE code for reduction wasn't 100% portable and was causing segfaults when run locally. Therefore, flip2 was used for this report. At the time, the load averages were in the low 3s.

2. Show the table and graph.

Array Size	Non-SIMD Mult	SIMD Mult	Speedup
1000	307.22	1780.67	5.80
2048	310.55	1862	6.00
4096	312.72	1751.16	5.60
8192	312.82	1743.01	5.57
16384	313.5	1753.08	5.59
32768	313.82	1453.39	4.63
66536	312.37	1399.06	4.48
131072	313.06	1370.13	4.38
262144	312.57	1376.5	4.40
524288	312.05	1369.84	4.39
1048576	306.06	1131.05	3.70
4194304	299.37	681.97	2.28
8388608	298.95	755.02	2.53
16777216	298.47	635.42	2.13
33554432	300.54	718.97	2.39
Array Size	**Non-SIMD Reduc**	**SIMD Reduc**	**Speedup**
1000	288.91	1771.85	6.13
2048	290.78	1858.85	6.39
4096	291.33	1872.31	6.43
8192	291.81	1843.08	6.32
16384	291.95	1846.56	6.32
32768	291.59	1814	6.22
66536	291.39	1803.45	6.19
131072	290.41	1774.18	6.11
262144	291.32	1777.7	6.10
524288	290.73	1776.83	6.11
1048576	289.93	1773.96	6.12
4194304	283.05	1085.52	3.84
8388608	284.2	1103.08	3.88
16777216	283.39	1004.17	3.54
33554432	284.26	1002.38	3.53



3. What patterns are you seeing in the speedups?

For both array multiplication (represented on the graph as “SIMD Mult”) and multiplication-reduction (on the graph as “SIMD Reduc”), speedup is consistently high until some time after the array size crosses one million elements. Beyond that threshold, the speedup of both drops considerably.

After the threshold, SIMD Mult’s SSE alternative offers a speedup factor of only slightly better than 2.

After the threshold, SIMD Reduc’s SSE alternative offers a speedup factor of around 3.5.

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

The speedups appear to be steady before and after the drop-off at one million array elements. SIMD Mult hovers in speedups in the late 5s (with some variance at the 32k mark, when the speedup becomes 4.4), while SIMD Reduc remains in the low 6s. After one million element arrays are encountered, both speedups are slightly better than half the speedups under lighter loads.

5. Why or why not, do you think?

The drop-off is likely due to a hardware limitation. As someone on the discussion board pointed out, flip’s L3 cache can handle roughly three million floats, which means that two float arrays of 1.5 million elements would bring it to capacity. This neatly coincides with the performance hit between the 1M and 4M mark.

6. 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?

and

7. 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?

Though the SSE SIMD expects 4-floats, the compiler notices when not all four spaces are used and can optimize automatically, leading to a greater-than-four speedup. Similarly, in the multiplication-reduction code, there is an FMA, which the compiler uses to its advantage to deliver higher speedups than the code itself would be able to produce (until the hardware bottleneck, of course).