Trevor Stahl

Project 4

CS475 Spring 2019

Professor Bailey

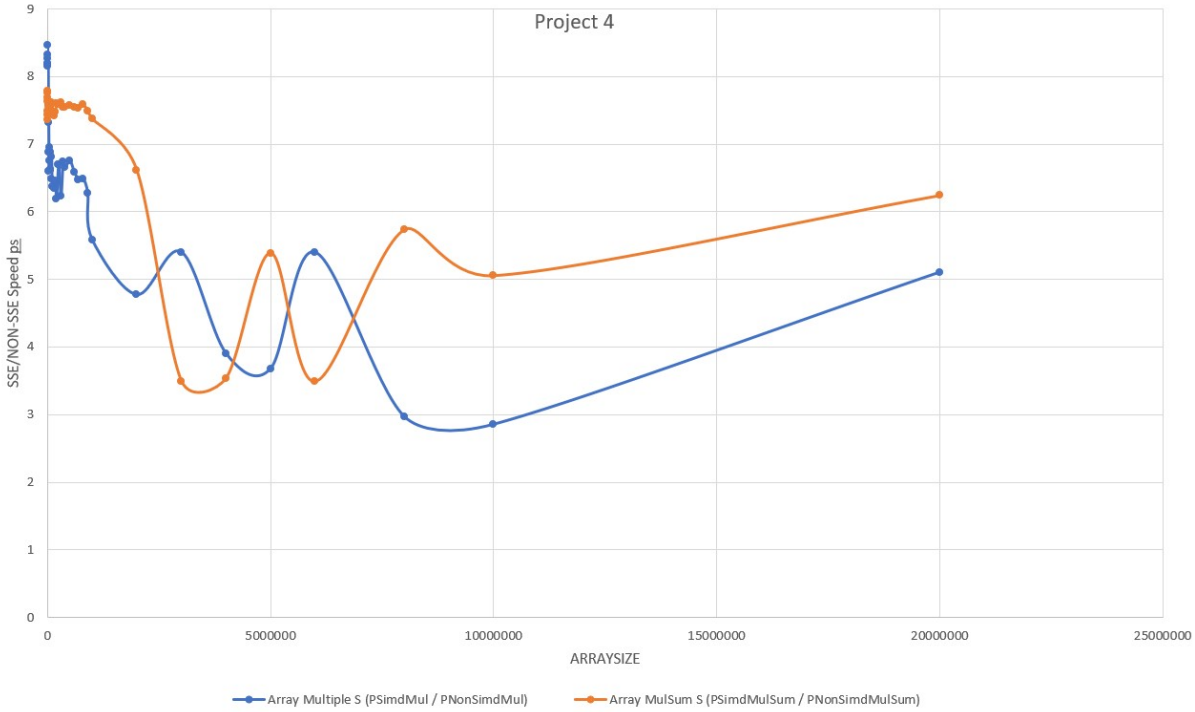
What machine you ran this on

I ran my code on the flip3 OSU server. I compiled with g++ -o proj4 project4.cpp -lm -fopenmp
Uptime load averages before running: 0.96 1.20 1.37

Uptime load averages after running: 1.63 1.62 1.55

Show the table and graph

Array Size	Array Multiple S (PSimdMul / PNonSimdMul)	Array MulSum S (PSimdMulSum / PNonSimdMulSum)
1000	8.1505	7.3617
2000	8.469	7.6906
3000	8.1779	7.7848
4000	8.3231	7.7619
5000	8.2655	7.432
10000	8.198	7.631
15000	8.1684	7.4837
20000	7.5934	7.6193
25000	7.3176	7.4646
30000	6.8853	7.553
35000	6.6047	7.4515
40000	6.7557	7.5565
50000	6.9478	7.5724
60000	6.6126	7.563
70000	6.8771	7.5652
80000	6.6276	7.5308
90000	6.8094	7.6172
100000	6.4796	7.4615
120000	6.3724	7.4509
140000	6.3691	7.4388
160000	6.3451	7.4206
180000	6.4516	7.4764
200000	6.186	7.596
250000	6.698	7.5869
300000	6.2282	7.6123
350000	6.7431	7.5406
400000	6.6584	7.5518
500000	6.7536	7.572
600000	6.5836	7.5491
700000	6.4729	7.5248
800000	6.4887	7.5836
900000	6.2756	7.4931
1000000	5.5813	7.3785
2000000	4.7704	6.6178
3000000	5.4017	3.4964
4000000	3.8998	3.5392
5000000	3.6777	5.3906
6000000	5.3974	3.4893
8000000	2.9636	5.7376
10000000	2.8521	5.0543
20000000	5.1031	6.2405



What patterns are you seeing in the speedups?

- At an array size of 8000000 and above, summing the multiplication of array A and B in a single variable is faster than writing each multiplication to a third array.

- Difference between SSE and Non-SSE is greatest at lower ARRAYSIZE values

- Given larger values data seems to indicate that normal multiplication will get closer and closer to MulSum

- Data seems to indicate that both Mul and MulSum will steadily increase if ARRAYSIZE grew larger.

- In between the upper very consistent region and the lower region, we have a region from 2500000 to 8000000 where the lines seems to trade alternating places for a few data points.

Are they consistent across a variety of array sizes?

- Yes, for all five patterns stated above I produced and recorded many data points to indicate them. Outside of this there seems to be three areas of separate behavior. The lowest ARRAYSIZE values are the first region where normal multiplication shows erratic behavior being sometimes quite high at around 8 and also low at 6.5. Where as MulSum here shows

reliable values of about 7.5. Both lines dip down at around 2500000. From here until 8000000 the lines show erratic behavior fluctuating up and down in an alternating manor.

Why or why not, do you think?

- The data at the lowest region is hard to explain, the difference between SIMD and nonSIMD seems to be greater here. This could be because at the lowest values the speed difference between SIMD and nonSIMD is maximized by not introducing large arrays which 2 or 3 of have to be brought into and out of cache.

- In the lowest region also, MulSum is far more consistent. I have no idea why this would be.

- The behavior of the middle region is bizarre to me. I have no guesses as to why the lines do what they do there.

- The right hand side region with the the largest ARRAYSIZE I can explain largely. MulSum is above normal multiplication between it has only 2 massive arrays that must be brought through cache allowing quick execution.

- And as stated before both lines in the right hand region are below where they were in the left hand region because a big part of the execution time is bringing arrays through cache.

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-mutiplication?

~~Well, for one if you are multiplying 4 things at once instead of 2 you are performing three multiplications instead of 1, which means it wont't be exactly 4 in theory. Another way I can think of it is. This goes for array-mutiplication-reduction as well however.~~

A. In the normal array-multiplication one might not see a speed of 4 because each operation is only ever between 2 floats.
B. Not really sure what I should have here. I don't really know.

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-mutiplication-reduction?

A. Here 4 elements could be multiplied at once.