Tanya Khemani
932988007

Project #5: Vectorized Array Multiplication and Reduction using SSE

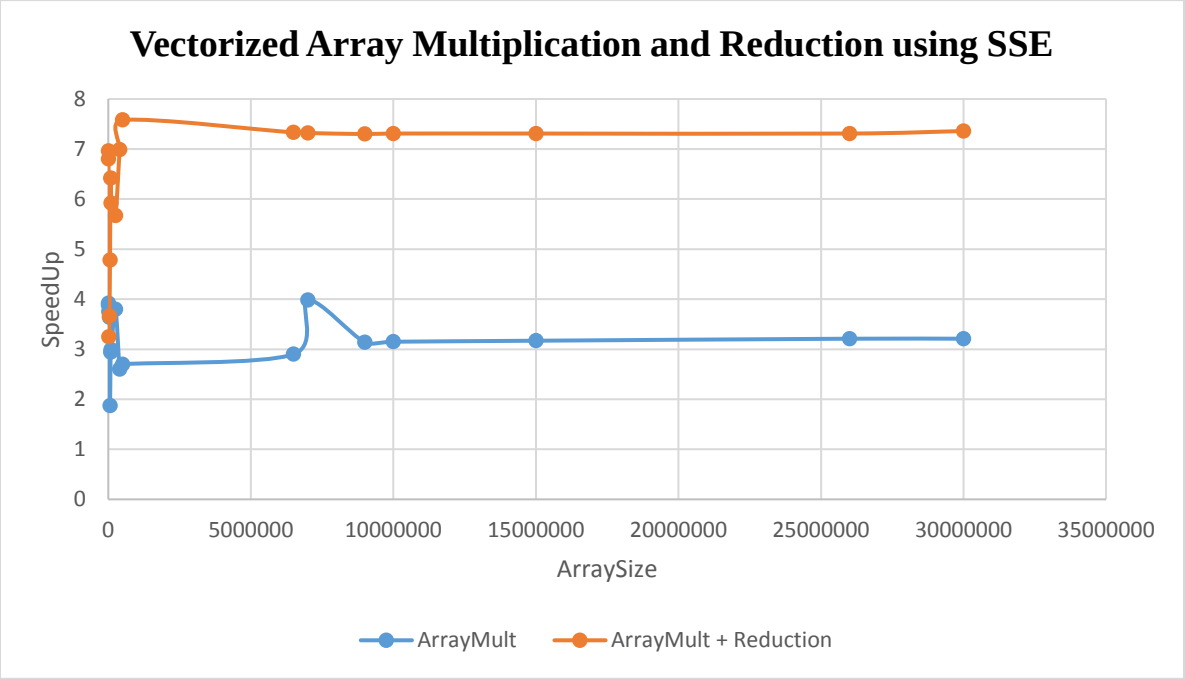
1. What machine you ran this on?

   Flip.engr.oregonstate.edu(Linux)

2. Show the table and graph

ArraySize	ArrayMult	ArrayMult + Reduction	
1000	3.92	6.96	
10000	3.88	6.8	
20000	3.75	3.25	
40000	3.64	3.66	
60000	1.87	4.78	
80000	2.94	6.42	
100000	2.99	5.92	
250000	3.8	5.67	
400000	2.6	6.99	
500000	2.7	7.58	
6500000	2.9	7.33	
7000000	3.98	7.32	
9000000	3.14	7.3	
10000000	3.15	7.31	
15000000	3.17	7.31	
26000000	3.21	7.31	
30000000	3.21	7.36	

Tanya Khemani
932988007



3. What patterns are you seeing in the speedups?

   Talking about ArrayMult, we can see that the speedup first decreases and then shoots up and then decreases again and remains constant (for a variety of array sizes). On the other hand, we can see that the speedup first decreases then increases and then again decreases and remains constant (for a variety of array sizes) for ArrayMult and Reduction.

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

   Yes, from the graph we can see that after a point, the speedups become consistent for a variety of array sizes for ArrayMult as well as for ArrayMult and Reduction.

5. Why or why not, do you think?

   The speedups become consistent for a variety of array sizes(both in ArrayMult and for ArrayMult and Reduction because in this there are no jumps or branches,

Tanya Khemani
932988007

there are no backward loop dependencies and also the elements are having contiguous memory address.

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 mutiplication?

   For array multiplication, I am getting a speedup of less than 4 because in array multiplication, we are calculating, SimdMul: C[0:len] = A[0:len] * B[0:len], and thus for this it takes more execution time since it involves more computations and hence its speedup comes out to be less than 4.

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 mutiplication-reduction?

   For array multiplication-reduction, I am getting speedups greater than 4 because in array multiplication-reduction, we are calculating, SimdMulSum: return (ΣA[0:len] * B[0:len]), and thus for this it takes less execution time(less computation) and hence its speedup comes out to be more than 4.