CS 575

Project #5

Vectorized Array Multiplication and Reduction using SSE

Zhouxiang Meng

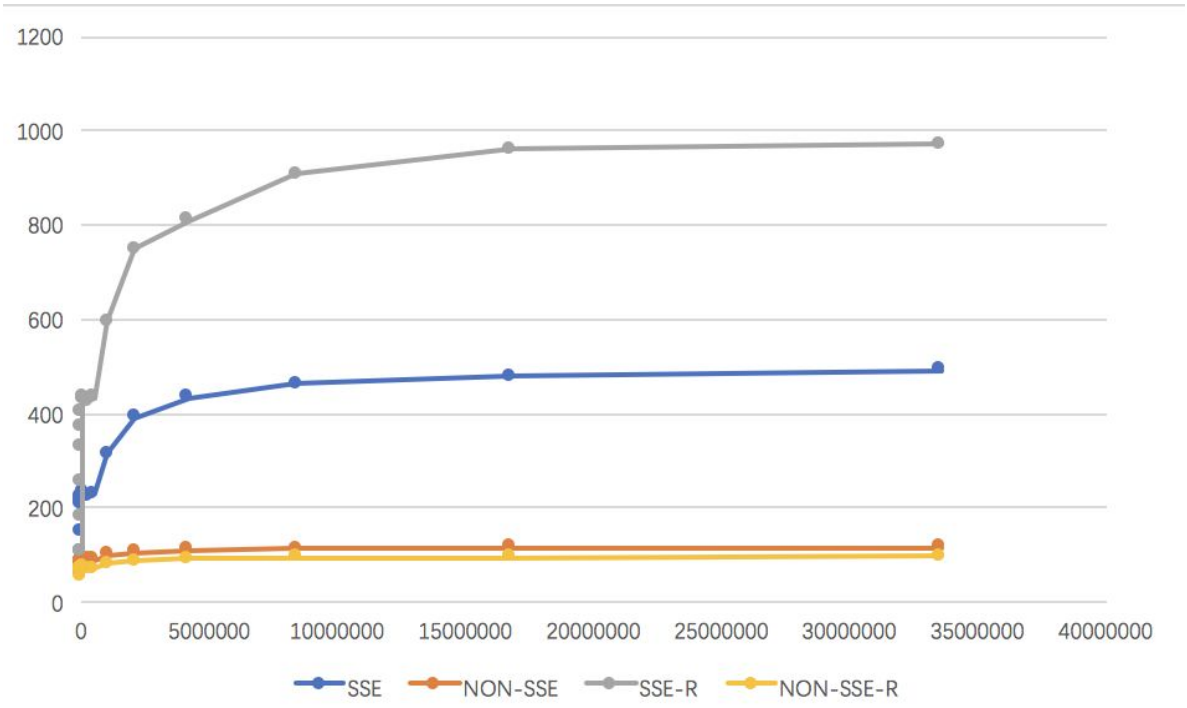
mengz@oregonstate.edu

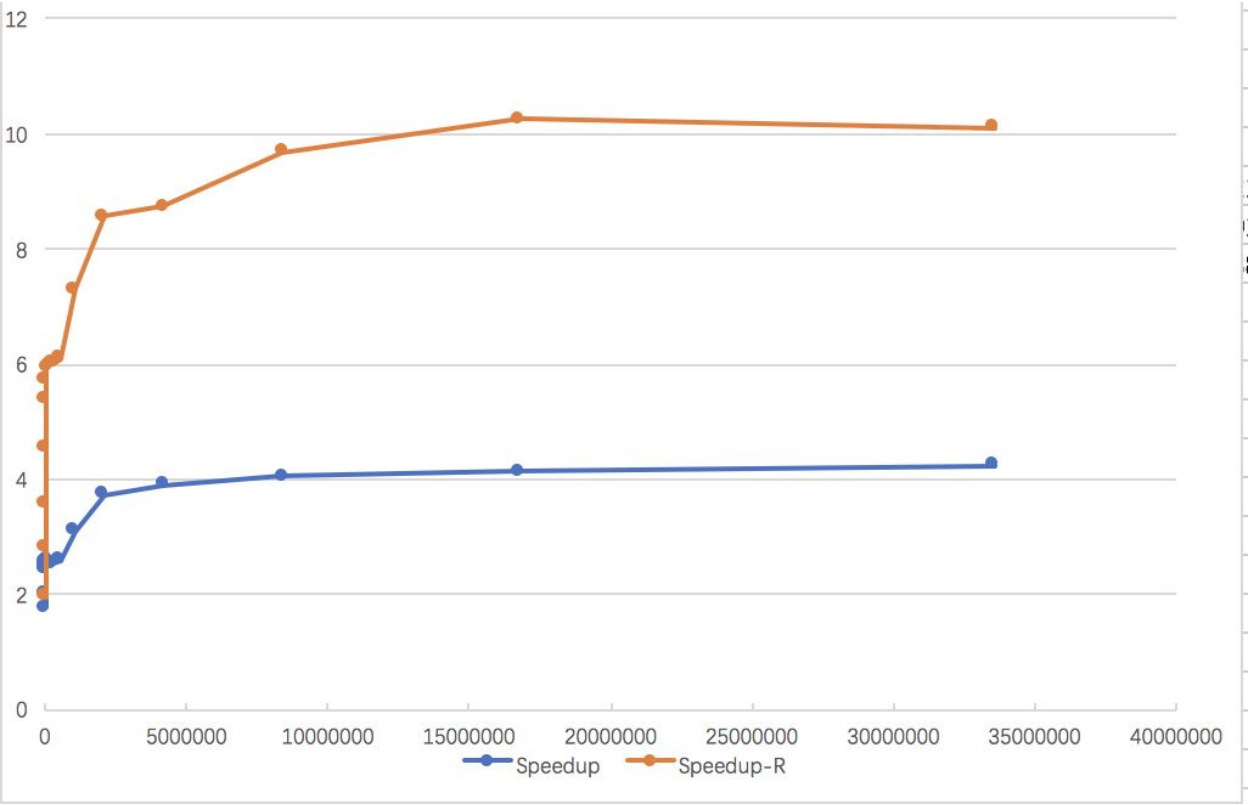
1. What machine you ran this on

   I run this project on OSU server.

2. Show the table and graph

Array_size	1024	2048	4096	8192	16384	32768	65536	131072
SSE	107.93	149.81	208.42	212.42	219.71	220.74	234.91	226.09
NON-SSE	61.2	74.32	84.63	87.36	86.733	86.66	89.68	89.07
SSE-R	107.65	180.61	255.01	328.26	372.8	405.22	427.15	434.18
NON-SSE-R	55.14	64.26	71.09	71.96	69.22	70.78	72.13	72.42
Speedup	1.76356209	2.01574273	2.46272008	2.43154762	2.53317653	2.54719594	2.61942462	2.53834063
Speedup-R	1.95230323	2.81061313	3.58714306	4.56170095	5.38572667	5.72506358	5.92194649	5.99530516
Array_size	262144	524288	1048576	2097152	4194304	8388608	16777216	33554432
SSE	223.76	228.99	313.77	391.55	433.01	463.1	478.31	492.4
NON-SSE	88.34	88.66	101.29	105.22	110.77	114.55	115.95	116.42
SSE-R	425.73	433.08	595.91	749.49	808.86	907.06	960.22	972.19
NON-SSE-R	70.73	71.07	81.57	87.57	92.54	93.65	93.66	96.24
Speedup	2.53294091	2.58278818	3.09773917	3.72125071	3.90909091	4.04277608	4.12514015	4.22951383
Speedup-R	6.01908667	6.09371043	7.30550448	8.558753	8.74065269	9.68563801	10.2521888	10.1017249





3. What patterns are you seeing in the speedups?

   From the graph, with the array size going up, SSE code does much better than traditional c code. From the graph I know that if the array size is bigger than about 10M, the speedup will remain consistent.

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

   The speedup of multiplication function isn't consistent. The speedup is consistent variety of array size.

5. Why or why not, do you think?

   Because the prefetching is used to take place a cache line in memory before we use it for reduction function. But multiplication function does not use prefetching, so when the size of array increase, the performance will decrease.

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/mutiplication-reduction?

   In my mutiplication result, the speed-up is around 4. But in my mutiplication-reduction result, the speed-up is larger than 4. I think this is because the SimdMul() has to create space for 3 arrays but SimdMulSum() only need to create space for 2 arrays. That is, the assembly code spends most of time doing malloc.