CS 475/575 -- Spring Quarter 2017

Project #5

Vectorized Array Multiplication and Reduction using SSE

Professor Mike Bailey

Xiao Tan

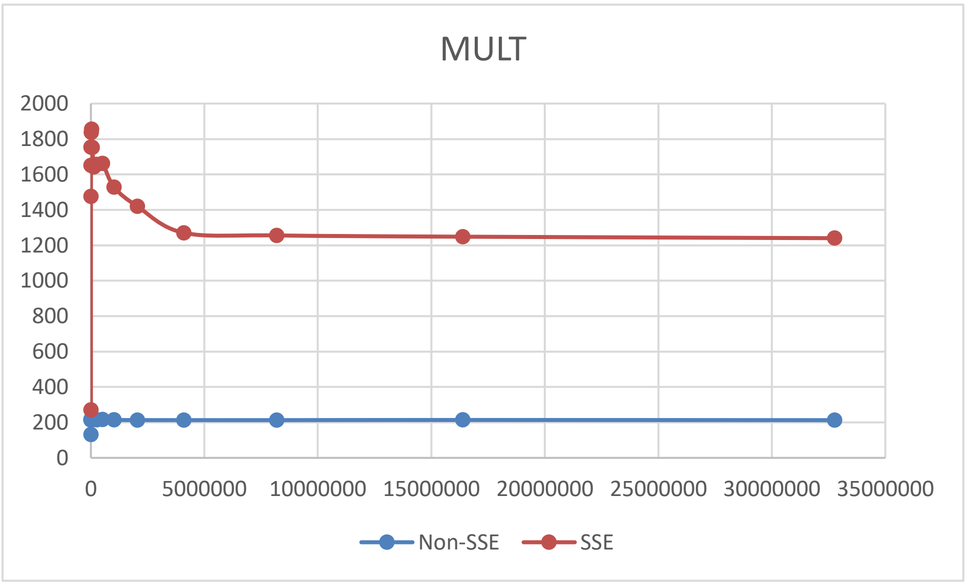
I. Runtime environment

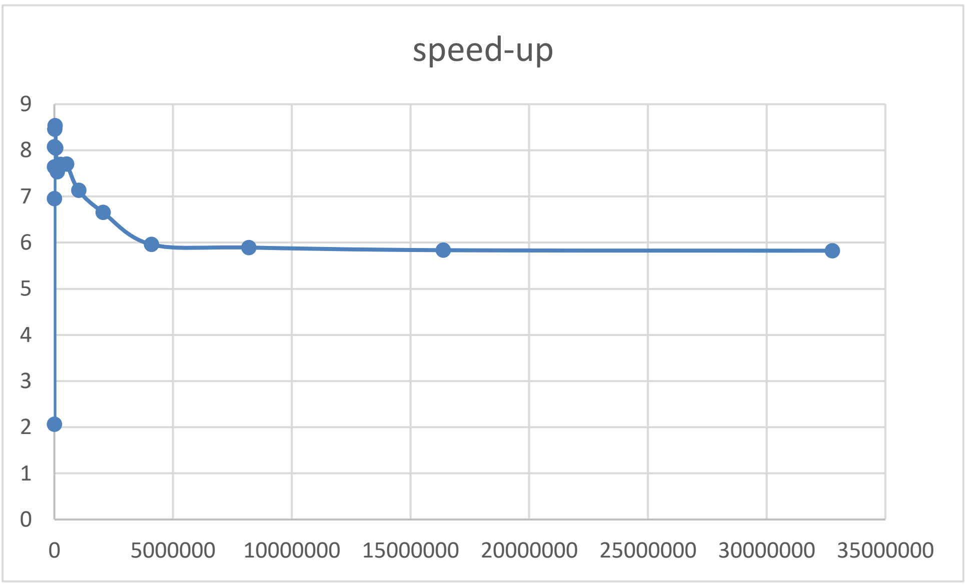
In this project, I ran my program on OSU's server flip.engr.oregonstate.edu. Then I got the following results, and graph.

II. Results and graph

First, the following table is for multiplication.

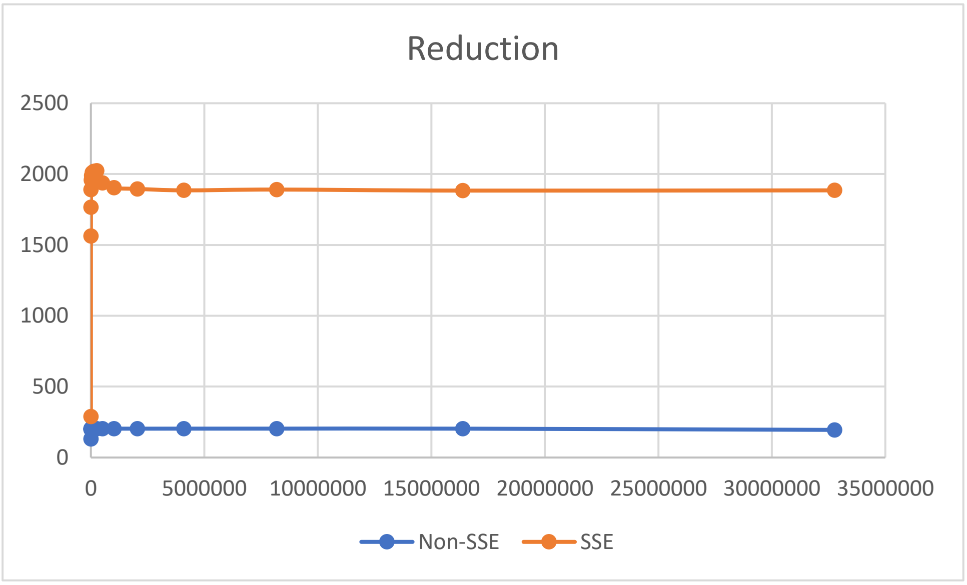
MULT	Non-SSE	SSE	speed-up
1000	131.22	270.86	2.064167048
2000	212.34	1476.19	6.952010926
4000	216.16	1650.8	7.636935603
8000	217.11	1753.94	8.07857768
16000	217.02	1836.34	8.461616441
32000	217.39	1855.7	8.536271218
64000	217.61	1751.84	8.050365332
128000	218.02	1642.79	7.535042657
256000	215.27	1656.18	7.693501185
512000	215.71	1661.55	7.702702703
1024000	214.25	1528.04	7.132042007
2048000	213.37	1420.06	6.655387355
4096000	213.14	1270.41	5.960448531
8192000	213.06	1255.76	5.893926593
16384000	213.93	1248.63	5.836628804
32768000	212.89	1240.23	5.825684626

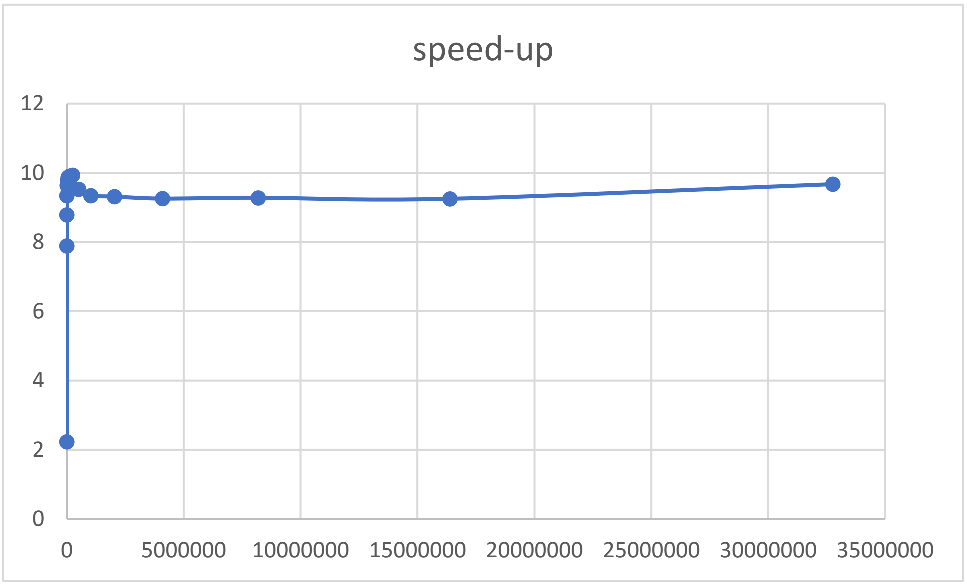




Then, there is showed the multiplication-reduction result.

REDUCTION	Non-SSE	SSE	speed-up
1000	129.68	288.68	2.226095003
2000	198.16	1562.66	7.885849818
4000	200.96	1766.02	8.787917994
8000	202.41	1890.81	9.341485104
16000	203.15	1957.65	9.636475511
32000	203.52	1988.7	9.771521226
64000	203.76	2010.24	9.865724382
128000	203.84	2019.18	9.905710361
256000	203.62	2022.45	9.932472252
512000	203.49	1938.36	9.525578653
1024000	203.57	1902.44	9.34538488
2048000	203.54	1895.48	9.312567554
4096000	203.66	1885.16	9.256407738
8192000	203.73	1890.74	9.280616502
16384000	203.61	1883.4	9.250036835
32768000	194.9	1885.07	9.671985634





III. Pattern analysis

In these previous graphs, I can see speedups are almost horizontal lines. For multiplication, the speedup went down at the beginning, then it becomes horizontal. Moreover, for multiplication-reduction, it didn't have really big changes except array size is 1000, so is multiplication.

It should be like this pattern, because the speed-up equation is $S = \frac{P_{sse}}{P_{non-sse}}$, their performance graphs are also showed before, so speed-up pattern should follow their performance. Because this is a single thread program, its performance will not have any big changes due to array size. Therefore, we got graph like this.

Furthermore, as we known, SSE SIMD is 4-floats-at-a-time, running under single thread, SSE program speed-up should be 4 times than non-SSE, however, I got a higher speed-up. That is because SSE is run with SIMD code, it is using assembly language, there is no need for compilers to compile C++ code to assembly language, so that machine will get their instruction quickly. In the other hand, C++ compiler should translate C++ code into assembly code, so that it won't get the full potential speed-up. Both two speed-ups have the same reason.