Sanketh Karuturi
karutusa@oregonstate.edu

CS 575

Project #4
Vectorized Array Multiplication and Multiplication/Reduction using SSE

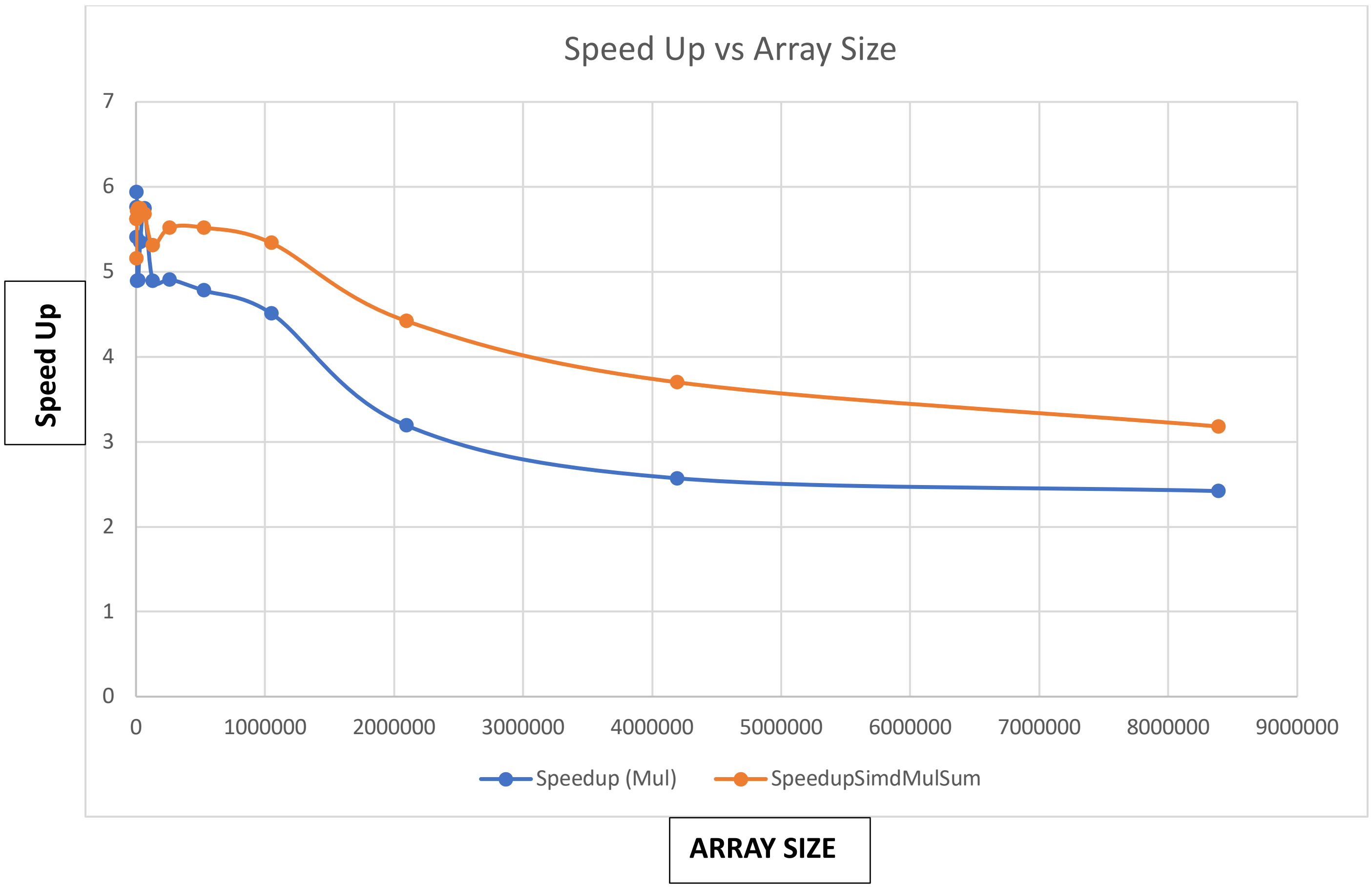
1. What machine you ran this on

→ Flip Server

2. Show the 2 tables of performances for each array size and the corresponding speedups

Array Size	NonSimdMu	SimdMul Tir	Speedup (M	NonSimdMu	SimdMulSu	SpeedupSin
1024	352.44	1906.7	5.41	351.65	1975.57	5.62
2048	315.32	1816.24	5.76	355.29	1834.22	5.16
4096	292.56	1737.8	5.94	356.19	2001.9	5.62
8192	357.31	1747.24	4.89	355.77	2033.45	5.72
16384	358.49	1756.6	4.9	356.6	2049.88	5.75
32768	357.54	1912.84	5.35	356.41	2050.48	5.75
65536	357.68	2056.66	5.75	357.89	2035.16	5.68
131072	356.92	1745.34	4.89	356.21	1890.69	5.31
262144	355.63	1746.14	4.91	355.86	1965.56	5.52
524288	353.14	1688.01	4.78	352.66	1948.85	5.52
1048576	349.26	1575.16	4.51	355.68	1901.42	5.34
2097152	337.35	1076.14	3.19	352.37	1557.53	4.42
4194304	338.84	870.82	2.57	351.29	1299.87	3.7
8388608	331.57	802.4	2.42	349.77	1113.25	3.18

3. Show the graphs (or graph) of SIMD/non-SIMD speedup versus array size (either one graph with two curves, or two graphs each with one curve)



4. What patterns are you seeing in the speedups?

→ Initial High Speedup at Smaller Array Sizes

The initial high speedup observed at smaller array sizes can be primarily attributed to the efficiency of SIMD operations when handling moderate amounts of data that can fit entirely or largely within the processor's cache. This results in faster data retrieval and fewer memory access penalties, maximizing the benefits of SIMD's parallel processing capabilities.

→ Decline in Speedup as Array Size Increases

As the array size increases, the speedup begins to decline. This decline can be explained by several factors that start to play a more significant role as the volume of data exceeds the cache size and requires more frequent memory accesses.

→ **Stabilization of Speedup at Larger Array Sizes**

The speedup stabilizes and levels off as the array sizes become very large. Despite the decline, SIMD operations still maintain a consistent advantage over non-SIMD implementations.

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

→ While the exact speedup value slightly fluctuates, the overall pattern of starting high and then stabilizing at a lower but consistent value is similar across different array sizes.

6. Why or why not, do you think?

- **Memory Hierarchy Influence**: At smaller array sizes, the data likely fits better within the CPU caches, leading to faster access times and higher speedups. As the array size increases, cache misses increase, which may slow down the computations and reduce speedup.

- **Overhead of SIMD Setup**: The initial setup cost of SIMD operations is amortized better over larger datasets, but once the setup is complete, the computational benefits become more uniform, explaining the stabilizing speedup.

- **Limits of Parallel Processing**: SIMD operations process data in parallel up to the width of the SIMD registers. Once the maximum benefit from parallelism is achieved, additional increases in array size do not contribute proportionally to speedups, hence the observed plateau in performance gains.