**Name: Nikhil Kumar Gattu**
**Mail: gattun@oregonstate.edu**
**ONID: 934615235**

1. **What machine you ran this on**
   ➔ I ran on Flip3 server in my personal laptop (MacOS).

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

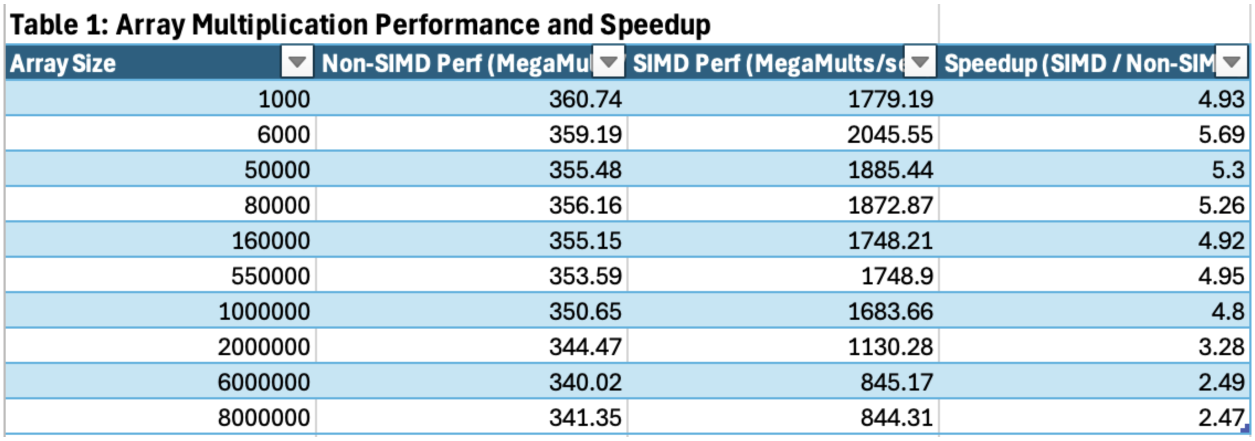


Table 1: Array Multiplication Performance and Speedup

| Array Size | Non-SIMD Perf (MegaMu | SIMD Perf (MegaMults/s | Speedup (SIMD / Non-SIM |
|---:|---:|---:|---:|
| 1000 | 360.74 | 1779.19 | 4.93 |
| 6000 | 359.19 | 2045.55 | 5.69 |
| 50000 | 355.48 | 1885.44 | 5.3 |
| 80000 | 356.16 | 1872.87 | 5.26 |
| 160000 | 355.15 | 1748.21 | 4.92 |
| 550000 | 353.59 | 1748.9 | 4.95 |
| 1000000 | 350.65 | 1683.66 | 4.8 |
| 2000000 | 344.47 | 1130.28 | 3.28 |
| 6000000 | 340.02 | 845.17 | 2.49 |
| 8000000 | 341.35 | 844.31 | 2.47 |

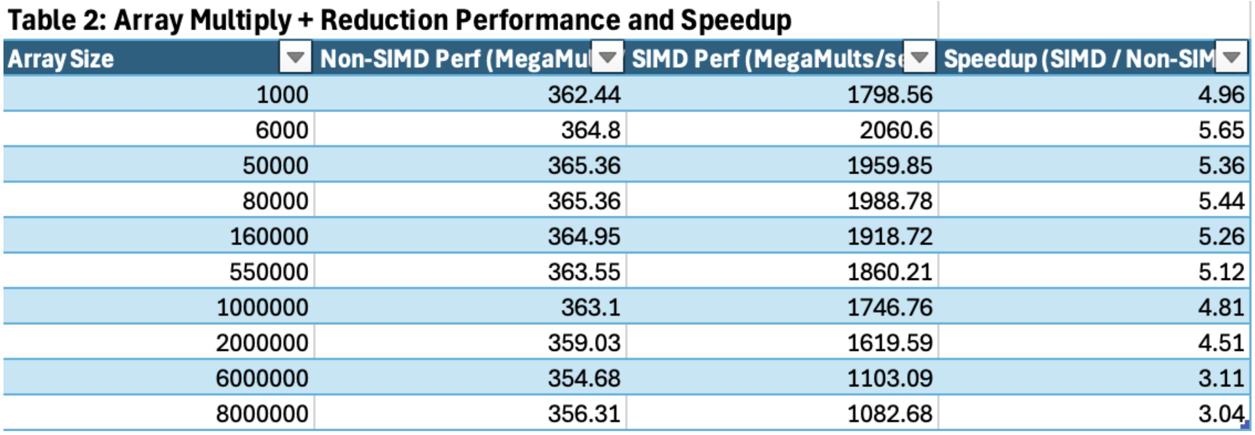
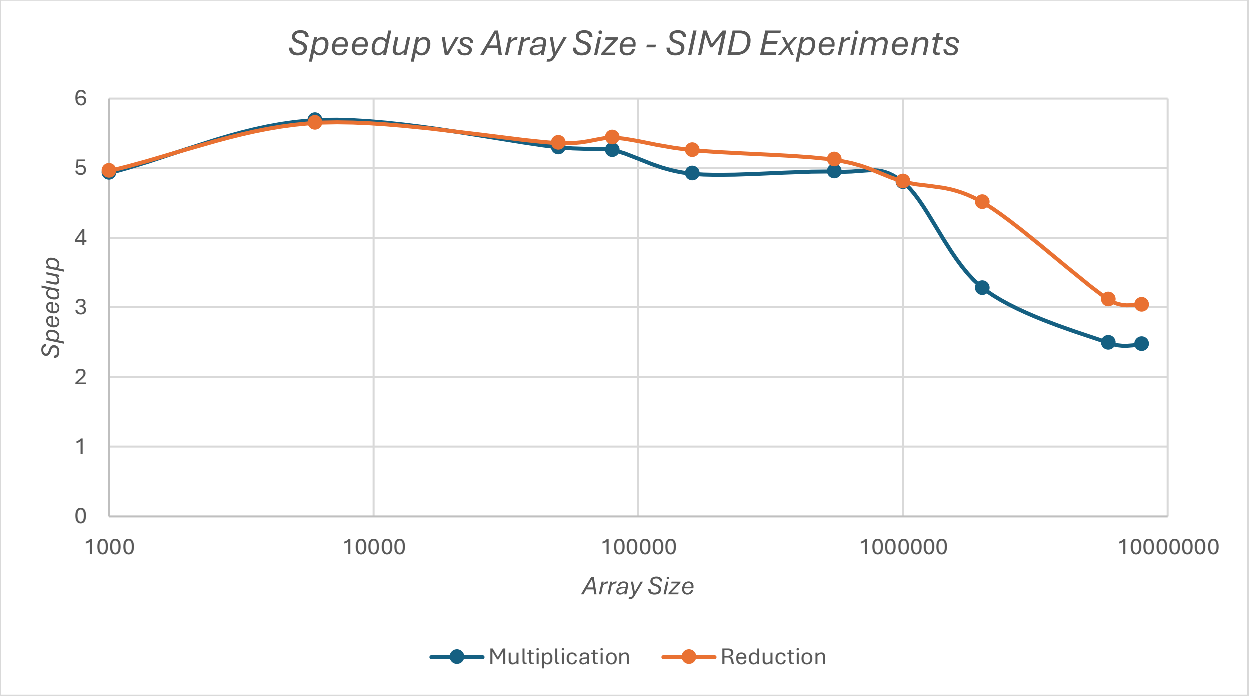


Table 2: Array Multiply + Reduction Performance and Speedup

| Array Size | Non-SIMD Perf (MegaMu | SIMD Perf (MegaMults/s | Speedup (SIMD / Non-SIM |
|---:|---:|---:|---:|
| 1000 | 362.44 | 1798.56 | 4.96 |
| 6000 | 364.8 | 2060.6 | 5.65 |
| 50000 | 365.36 | 1959.85 | 5.36 |
| 80000 | 365.36 | 1988.78 | 5.44 |
| 160000 | 364.95 | 1918.72 | 5.26 |
| 550000 | 363.55 | 1860.21 | 5.12 |
| 1000000 | 363.1 | 1746.76 | 4.81 |
| 2000000 | 359.03 | 1619.59 | 4.51 |
| 6000000 | 354.68 | 1103.09 | 3.11 |
| 8000000 | 356.31 | 1082.68 | 3.04 |

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?**

   ➔ In both results, array multiplication and array multiplication with reduction the speedup from using SIMD is initially high, peaking around 5 to 5.7x. This trend is strongest with small to medium array sizes (from 1K to ~500K). As the array size increases beyond that point, the speedup begins to gradually decline. By the time we reach 8 million elements, speedup drops to around 2.4 to 3.0x, depending on the operation.

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

   ➔ The speedups are relatively consistent within each size range, especially for small and medium arrays. There's a smooth, predictable curve, the speedup starts high and gradually flattens out or declines. This shows that SIMD performance benefit is consistent within resource-friendly sizes, but the trend changes as we scale up.

6. **Why or why not, do you think?**

   ➔ This change in pattern likely stems from memory hierarchy and bandwidth limitations. For small arrays, most data fits in CPU cache, so SIMD instructions benefit from low-latency memory access. As the array size grows, data must be loaded from main memory, increasing latency and reducing the effectiveness of vectorization. Additionally, I think the overhead of loop control, cache misses, and lack of memory parallelism limits the gain from SIMD, even though it's still faster than non-SIMD execution.