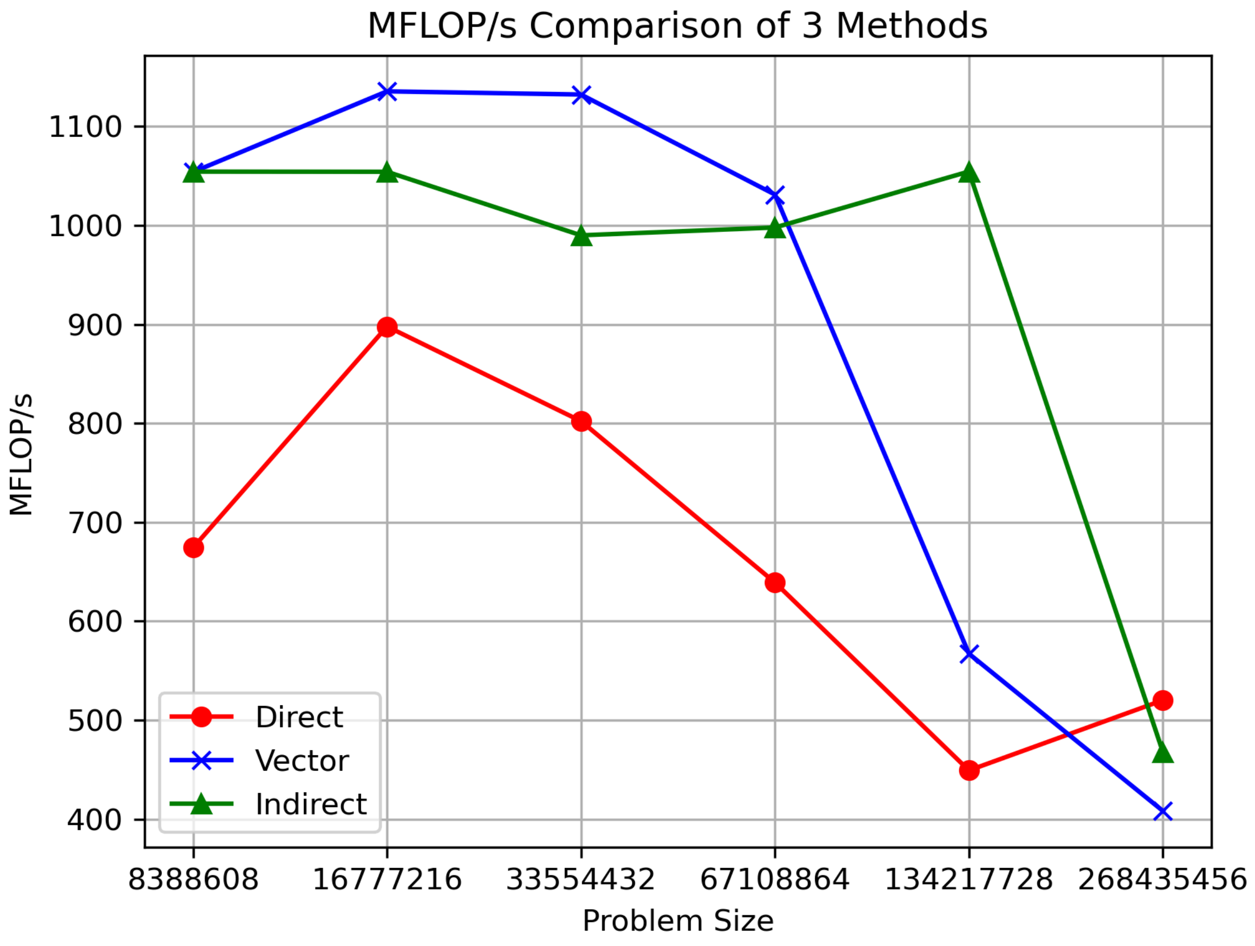
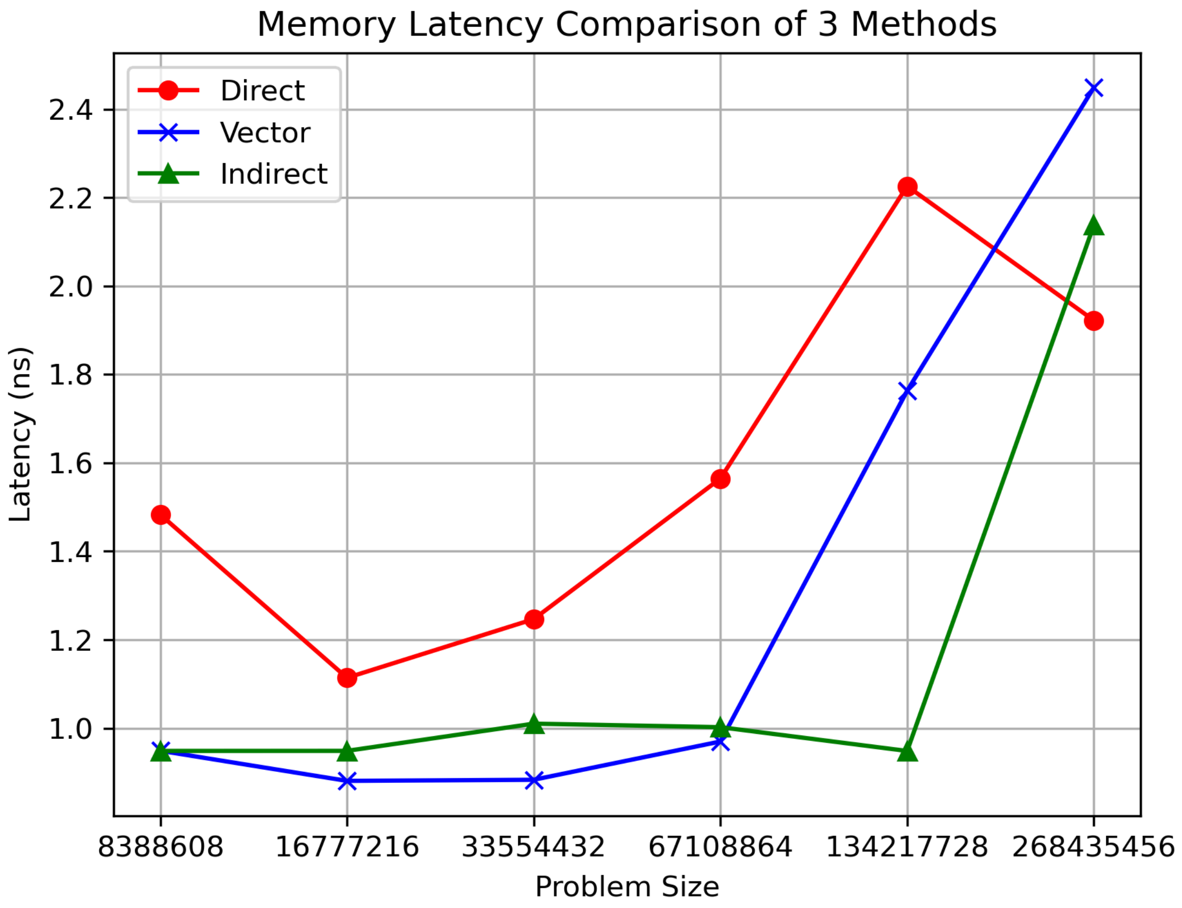
Ken Chika

Professor E. Wes Bethel

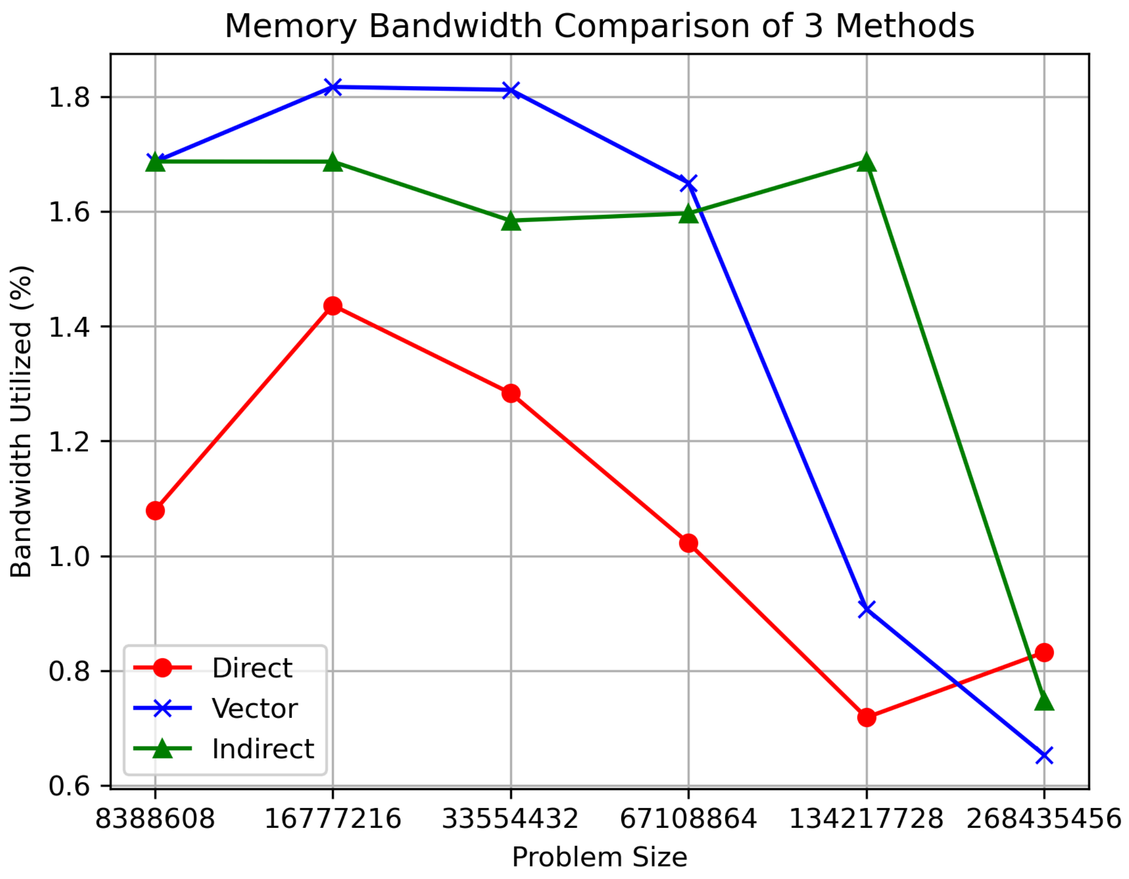
CSC 656-01 F24 CP #2

28 March 2025





Memory Latency Comparison of 3 Methods
Direct
Vector
Indirect
Latency (ns)
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
8388608
16777216
33554432
67108864
134217728
268435456
Problem Size


Memory Bandwidth Comparison of 3 Methods
Bandwidth Utilized (%)
1.8
1.6
1.4
1.2
1.0
0.8
0.6
8388608
16777216
33554432
67108864
134217728
268435456
Problem Size
Direct
Vector
Indirect

Analysis

(a) Expensive Operations:

Based on the latency data in my CSV files and corresponding graphs, the "cost" of a memory access is primarily determined by its average latency (in nanoseconds). For instance, at a problem size of 8,388,608, the Direct method shows an average latency of 1.483083 ns, compared to about 0.949 ns for both the Vector and Indirect methods—indicating that, at this size, Direct incurs more expensive memory accesses. However, at a problem size of 134,217,728, the Indirect method exhibits a lower latency (0.948451 ns) than both Direct (2.226032 ns) and Vector (1.763120 ns). This suggests that, while theory (as discussed in CSC 656 Lecture 15 and Hennessy & Patterson Section 5.2) predicts that random accesses should be expensive, the actual measured cost depends on how well the system's cache and prefetching mechanisms mitigate these penalties.

(b) Computational Rate (MFLOP/s):

Based on the data, the Vector method achieves the highest MFLOP/s for problem sizes 16,777,216 (1,135.36 MFLOP/s), 33,554,432 (1,132.14 MFLOP/s), and 67,108,864 (1,031.10 MFLOP/s). In contrast, the Indirect method outperforms Vector at 8,388,608 (1,054.24 MFLOP/s vs. 1,053.85 MFLOP/s) and 134,217,728 (1,054.35 MFLOP/s vs. 567.18 MFLOP/s), while the Direct method leads only at 268,435,456 (520.20 MFLOP/s vs. 467.82 and 408.31 MFLOP/s). This suggests that, overall, the Vector method exhibits the best computational rate in most cases—likely because its sequential access pattern enables more efficient caching and prefetching, as emphasized in CSC 656 Lecture 15 and Hennessy & Patterson (Section 5.1 on Performance Analysis).

(c) Memory Bandwidth Usage (Vector vs. Indirect):

According to the Memory Bandwidth plot, for problem sizes 16,777,216, 33,554,432, and 67,108,864 the Vector method achieves higher memory bandwidth utilization (up to about 1.8%), consistent with CSC 656 Lecture 16 and H&P Section 5.3 that emphasize the benefits of sequential access for efficient prefetching. However, at 8,388,608 and 134,217,728 the Indirect method records comparable or slightly higher bandwidth usage, suggesting that its random accesses still retain enough locality to leverage the system's memory bandwidth effectively.

(d) Memory Latency (Vector vs. Indirect):

From the Memory Latency plot, the Vector method shows lower latency (approximately 0.88–0.97 ns) at problem sizes 8,388,608 through 67,108,864, which is expected for sequential accesses due to better cache utilization (CSC 656 Lecture 15 and H&P Section 5.4). Interestingly, at 134,217,728 the Indirect method records a significantly lower latency (~0.95 ns), and at 268,435,456 the Direct method has the lowest latency (~1.92 ns), indicating that the actual memory latency is influenced by the interplay of access patterns and cache behavior, which may vary with problem size.