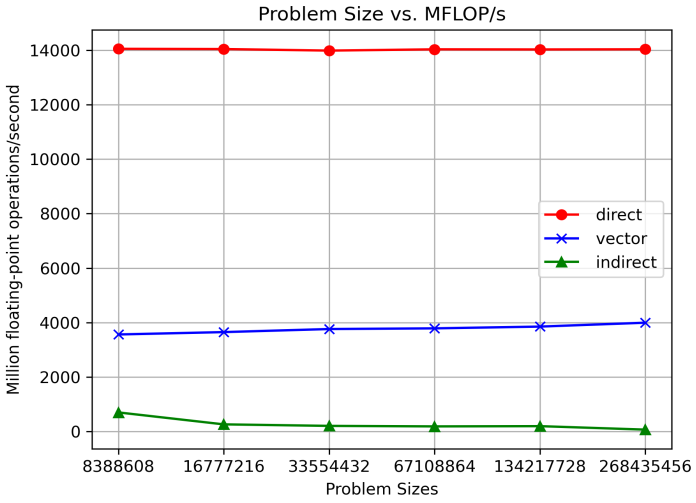
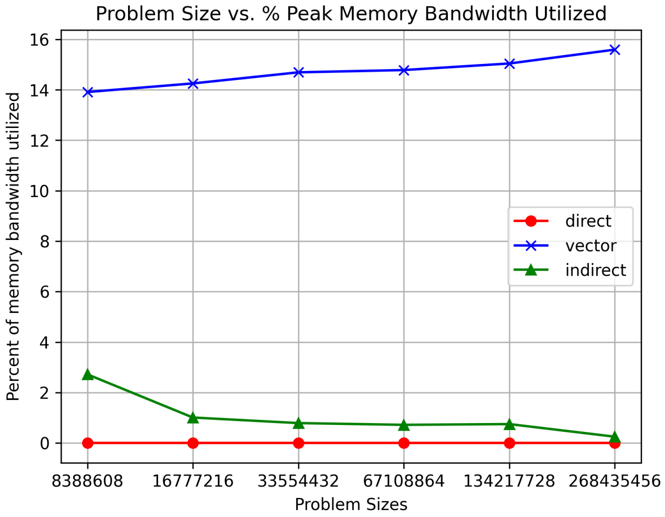
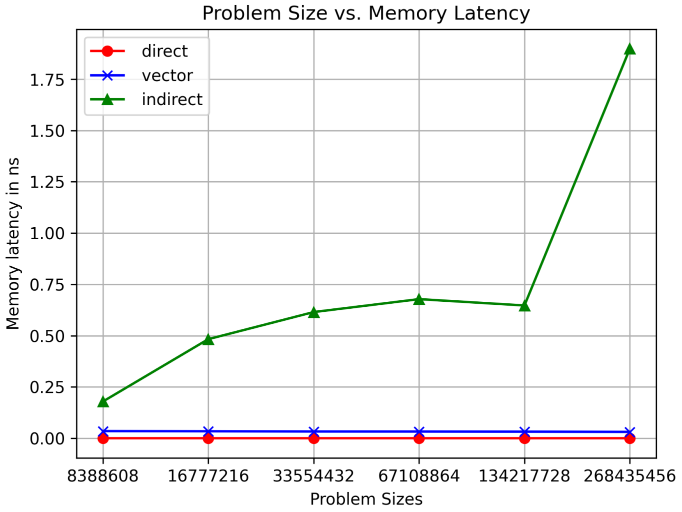

Problem Size vs. MFLOP/s


Problem Size vs. % Peak Memory Bandwidth Utilized


Problem Size vs. Memory Latency

What types of operations are more expensive and why?

Memory access operations are more expensive because getting data from memory is similar to us getting a book from the library. The CPU would first check the cache for the data, if it's a cache miss, then there would be a missed penalty, then finding the data in main memory and the distance the data has to travel to get to the CPU. All those steps would contribute to memory access operations being more expensive.

Which of the codes is performing a larger number of more expensive operations?

I think the indirect sum is performing a larger number of more expensive operations. Even though both algorithms perform one addition operation and one memory access operation, the difference is how both algorithms function. The vector sum algorithm is more cache-friendly which leads to a higher cache hit rate than the indirect sum algorithm, making the indirect sum algorithm more expensive.

Computational rate. Which of the 3 methods has the best computational rate (MFLOP/s)? Why?

The direct sum method has the best computational rate since it is only executing a simple addition operation. The other two are accessing memory which takes up valuable time that could have been used to execute another operation.

Memory bandwidth usage. Of the 2 methods vector sum and indirect sum, which has higher levels of memory bandwidth utilization? Why?

The vector sum method has a higher level of memory bandwidth utilization since all of the memory addresses that the CPU needs to read are in a linear fashion, the CPU can just preemptively get all the data and cache as much of them as it can. Then when the CPU tries to get the data for the next slot in the array, it will check the cache first, and it would be a cache hit because the data is already there. This would cause the memory bandwidth utilization to spike when caching for the first time, then drop to near zero, then spike again when it needs to repeat the process for the data later down the array.

Memory latency. Of the 2 methods vector sum and indirect sum, which shows lower levels of memory latency? Why?

The vector sum method has a lower level of memory latency since the CPU can preemptively cache the data, leading to a lot more cache hits and lowering the number of times it has to access memory. With both a higher cache hit and a lower number of times the CPU has to access memory, the memory latency would also be lower.