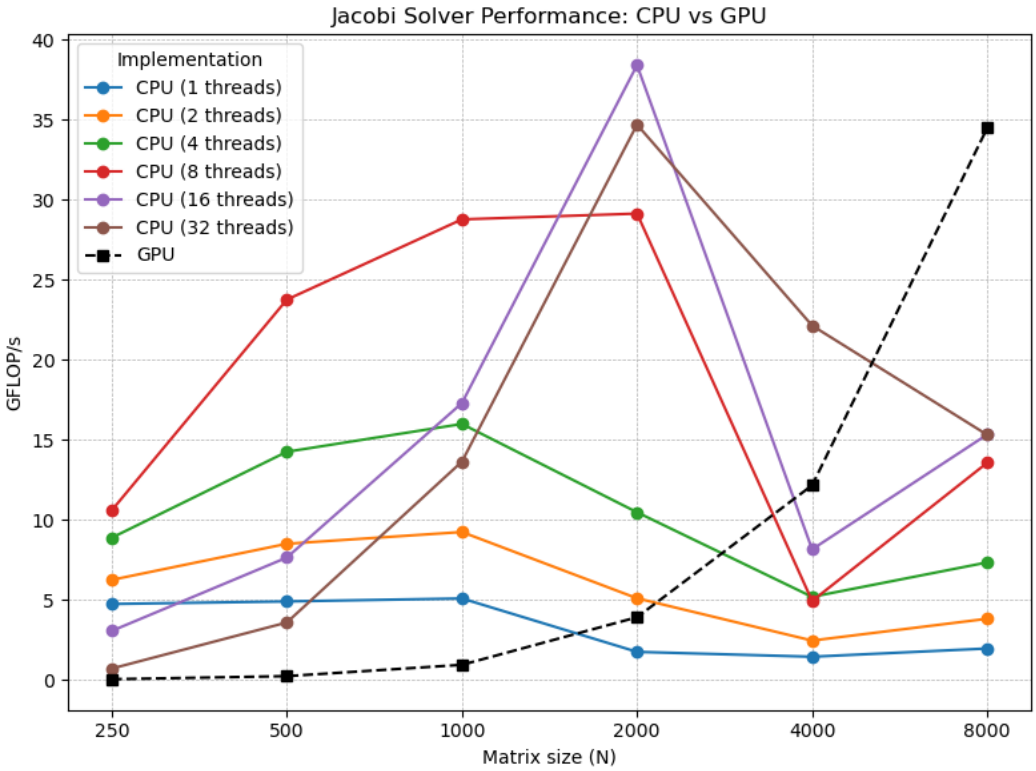
Project 4 Write up

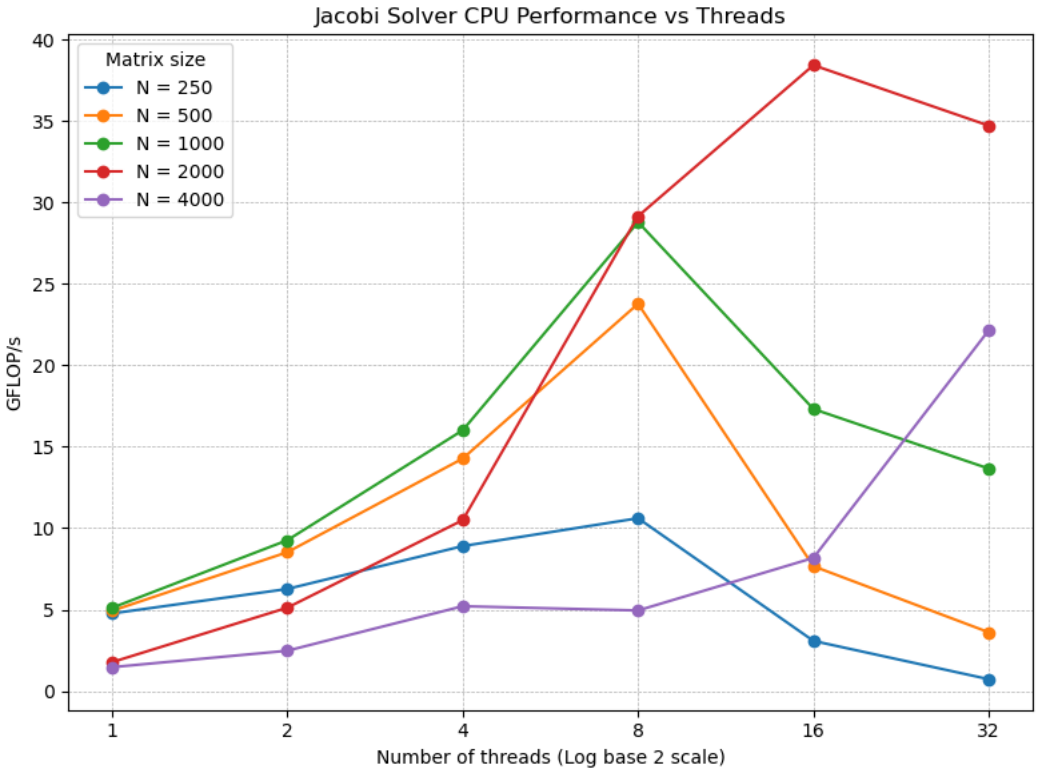
Derek Bowman



This figure shows giga-FLOP rate versus matrix sizes (NxN matrix) for CPU multithreading and GPU offloading for a Jacobi solver.

Starting with CPU multithreading, we see optimal FLOP rate at matrix sizes of 2000 at 16 threads. At larger matrix sizes, we see top FLOP rates with 32 threads, but the performance precipitously drops. This reflects cache-capacity limits, where the working set fills up cache and every matrix update hits DRAM, causing a drop in FLOP rate. On the flip side, for small matrix sizes (250, 500) with larger thread counts (16, 32), the overhead costs completely dominate the actual computation time. This leads to a horrendous performance.

For GPU offloading, this figure illustrates how start-up costs for GPU offloading is critical to consider, especially for smaller job size. It shows explicitly how GPU offloading is really only superior to CPU multithreading at much larger job sizes (8000). Because the start-up costs are so extensive when using GPUs, including starting the GPU kernel and copying data to the GPU, only when the job size (N) becomes large enough does the amortization of these start-up costs pay off. We can see this at matrix size 4000 and especially at 8000. At smaller job sizes, the start-up costs completely dominate the actual computation time. This leads to terrible FLOP rates at smaller matrix sizes.



This figure is illustrates a similar trend to the first figure. We can see that the giga-FLOP rate tends to increase as the number of threads increases for *large* matrix sizes (especially 2000 and 4000). However, the opposite is true for small matrix sizes (250, 500), where giga-FLOP rate plummets at larger thread counts (16 and 32). This demonstrates how overhead costs become disproportionate to the actual computation costs. This leads to drops in FLOP rates.