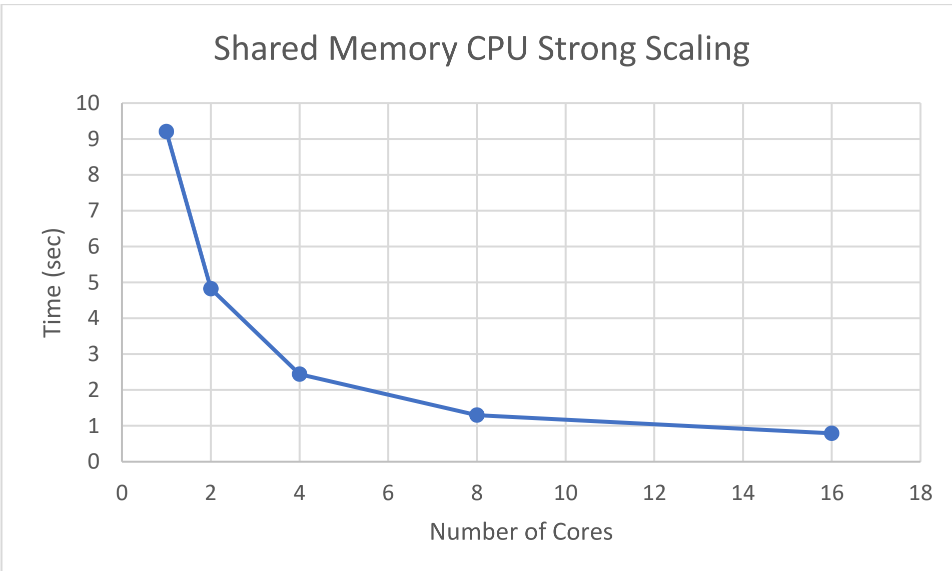
Shared Memory CPU Strong Scaling Study		
Cores	Time	Speedup
1	9.206	1
2	4.824	1.9083748
4	2.441	3.7714052
8	1.302	7.0706605
16	0.792	11.623737

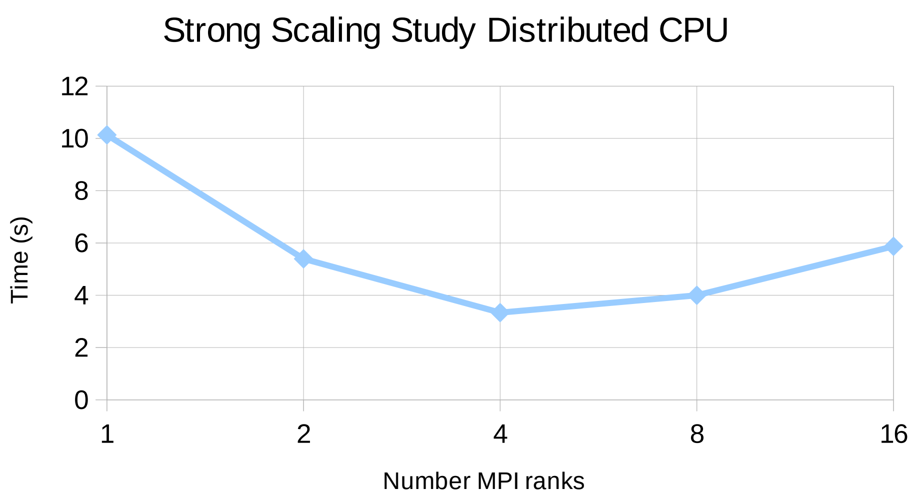


This scales well, but the speedup begins to decrease with more than eight cores.

Serial Time: 11.157 (s)

Distributed CPU Strong scaling study

Number MPI ranks	Time (s)	Speedup
1	10.134	1
2	5.394	1.878754171
4	3.335	3.03868066
8	3.997	2.535401551
16	5.873	1.725523582



Increasing the number of MPI ranks is beneficial up to a point, but too may ranks and it starts to take more time, not less!

Shared GPU block size study

Block Size	Time (ms)
8	754
16	646
32	769

This was run on a NVIDA GeForce GTX TITAN X with a max thread count per SM of 1024. Based on this a 16 sized bock would best utilize the GPU.

Distributed mpi ranks analysis (run with block size 16)

Number MPI r	Time (s)	Speedup
1	2.102	1
2	2.35	0.894468085

Block Size Analysis (run with mpiexec -n 2)

Block Size	Time (s)
4	3.659
8	2.785
16	2.495
32	2.462
64	2.248
128	2.44

This was run with NVIDIA GeForce GT 1030s. The max number of threads per block is 1024.

With 2 GPUs it seems to benefit from a larger block size, in particular 64.

Analysis and Thoughts

The most efficient solution was the shared GPU solution. Adding a second GPU didn’t seem to speed it up at all. Both GPU implementations benefitted slightly by increasing the block size, but only up to a point. Overall, the block size didn’t have a huge impact on performance unless it was excessively small.

For both the shared memory and distributed CPU implementations they ran in a time similar to serial if they were running on 1 thread, which makes sense. But for the shared memory, increasing the number of threads used to 16 continued to improve performance, while for the MPI solution at that point performance starts to decrease!