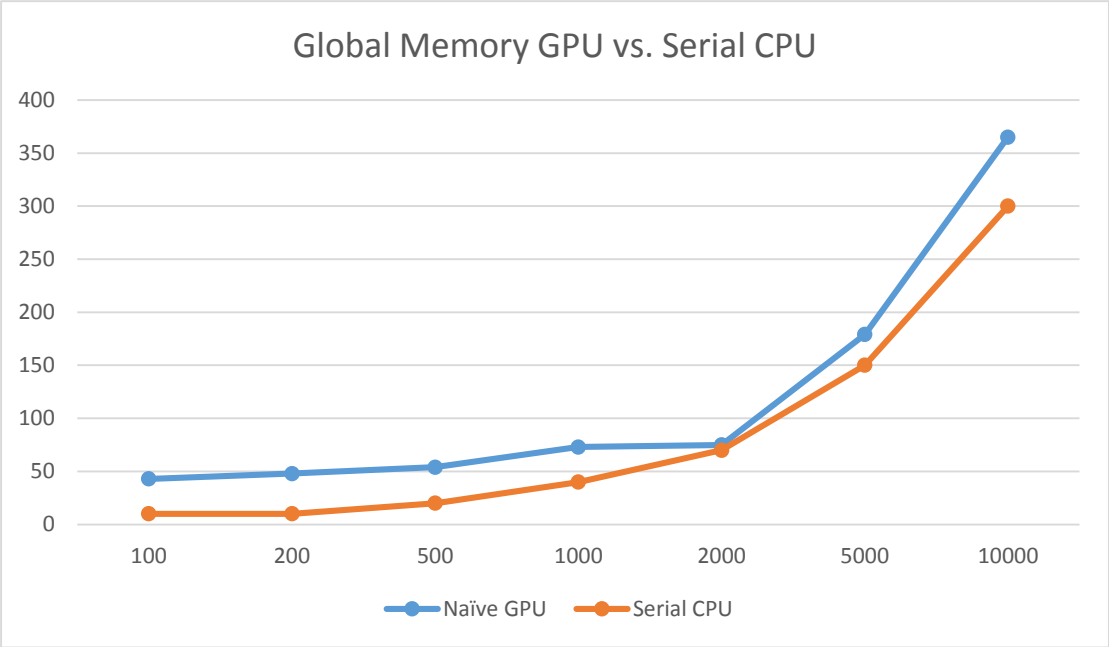
Naive prefix sum vs. Serial version of exclusive prefix scan

Iteration times: 1000, blocksize: 128

n	100	200	500	1000	2000	5000	10000
Naïve GPU	43	48	54	73	75	179	365
Serial CPU	10	10	20	40	70	150	300

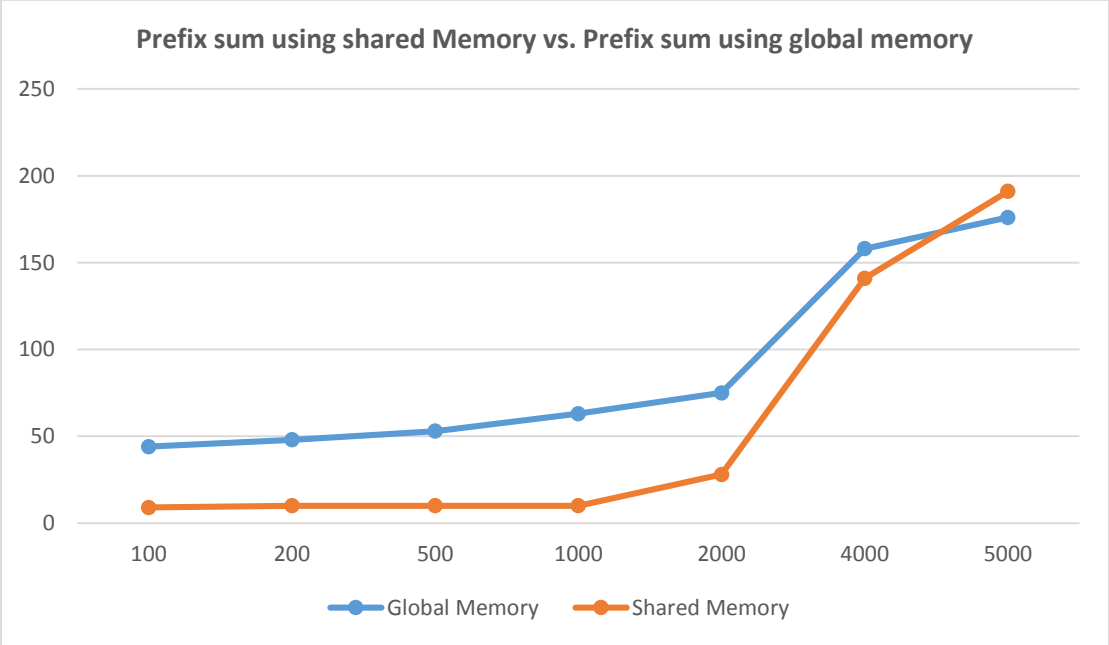


The serial version is much faster than the naïve parallel version when the array size is small, but this difference become less obvious when the array size grow. I expect the parallel version will be more efficient when the array size is bigger than a point.

Prefix sum using shared Memory vs. Prefix sum using global memory

Iteration times: 1000, blocksize: 128

n	100	200	500	1000	2000	4000	5000
Shared memory	9	10	10	10	28	141	191
Global memory	44	48	53	63	75	158	176

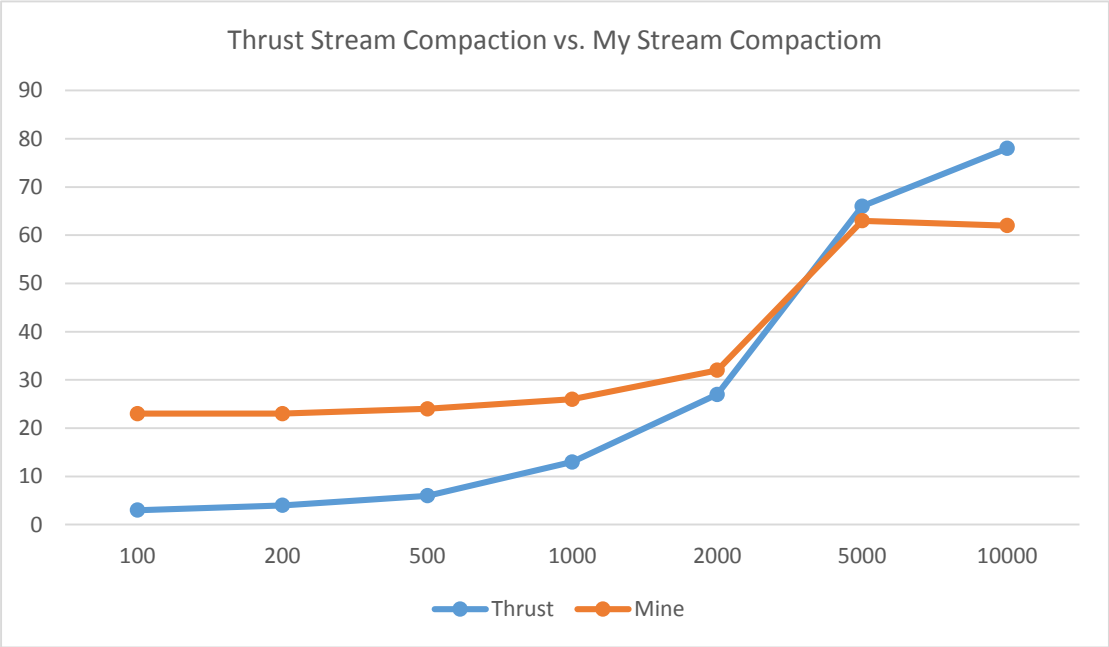


Apparently the shared memory version is more efficient and the computational cost increase slower with the increase of array size. That makes sense because read and write on global memory is slower than in shared memory. But when the array size become really bigger, my shared memory version does not work out very well, I think that may have something to do with how I assigned the volume of shared memory, but I am not sure how to optimize it.

Thrust Stream Compaction vs. My Stream Compaction

Iteration times: 1000, blocksize: 128

n	100	200	500	1000	2000	5000	10000
Thrust	3	4	6	13	27	66	78
Mine	23	23	24	26	32	63	62



The thrust version is faster than my version at the beginning, but the difference becomes smaller and smaller when the array size increase. I guess it related to whether the shared memory size id big enough for the data. My shared memory version still use some global memory so I think it can be optimized by converting it into a through shared memory version.