Array Sum Problem (array size 1,000,000):

	Sequential	1 thread	2 threads	3 threads	4 threads
Time (milliseconds)	10	8	15	16	16

Matrix Multiplication Problem (for matrix size 1,000:

	Sequential	1 thread	2 threads	3 threads	4 threads
Time (milliseconds)	3359	4215	2403	1895	1612

By these results, it appears that the array sum does not gain any significant speedup from solving the problem in parallel. In fact, adding more threads seems to slightly hinder the total speed. It's possible that there isn't a significant amount of computations to show any significant positive change, but by adding additional overhead by creating and joining threads, the program is slowed down. Maybe if the array size was larger, we would notice an increase in performance instead.

On the other had, the matrix multiplication problem is shown to achieve computational speedup by adding additional threads. Having only one thread caused the overall time to run increase, possibly from the additional overhead caused by pthread creation and management, but adding additional threads had a positive effect on the running time. This might be due to the fact that the matrix multiplication program has significantly more operations:

sumPart ~ 1000000 elements * 1 addition per element = 1 million total operations
matrixMultiplyThread ~ 1000000000 * (1000 multiplications + 999 additions)
= 1.999 billion total operations

Again, I think if the array sum problem had the same total operations, we might see a similar change in performance.