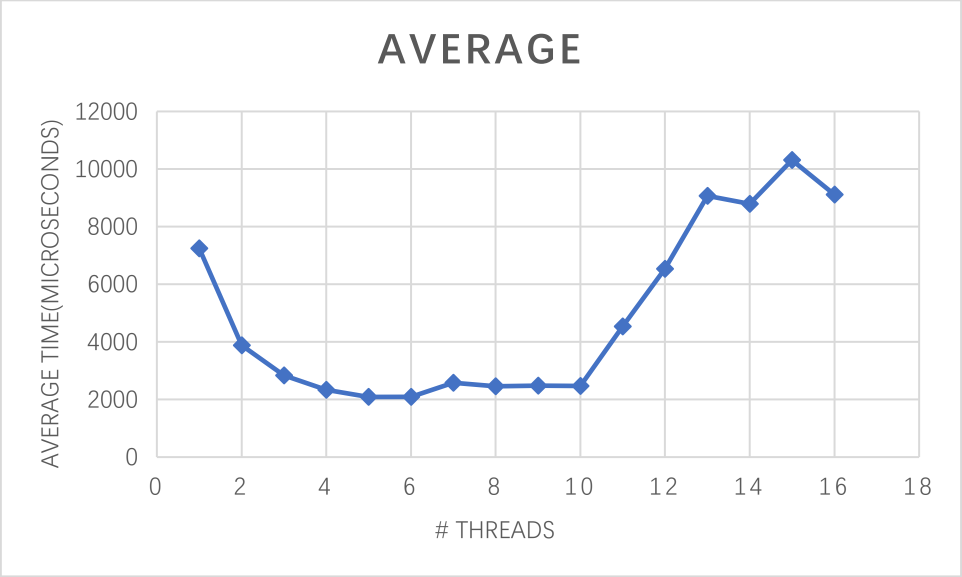
Running Time Report

Shuojiang Liu

Based on the chart (the times are in microseconds):

# Threads	time1	time2	time3	time4	time5	average
1	7149.099	7642.242	6964.003	7263.549	7212.438	7246.266
2	3940.48	3707.124	3735.849	3911.839	4125.386	3884.136
3	3101.793	2548.044	2875.125	2887.033	2756.226	2833.644
4	2387.69	2280.849	2269.964	2247.138	2519.398	2341.008
5	1987.585	1747.324	2685.553	1990.29	2027.356	2087.622
6	1995.366	2287.26	1976.034	2181.354	2002.542	2088.511
7	2570.072	2541.631	2427.725	2970.832	2377.498	2577.552
8	2488.39	2267.753	2803.459	2243.07	2499.718	2460.478
9	2529.057	2313.893	2693.041	2304.426	2584.625	2485.008
10	2492.382	2424.893	2651.09	2302.095	2484.967	2471.085
11	5900.011	3322.242	4534.985	6512.864	2390.324	4532.085
12	9013.23	8832.99	4174.069	6277.056	4388.089	6537.087
13	8800.619	9310.429	9194.478	8895.031	9129.844	9066.08
14	10239.48	9095.921	8109.865	9475.551	7052.357	8794.635
15	10220.19	12306.44	10325.95	8769.476	9929.753	10310.36
16	10869.28	8958.981	9205.321	10828.35	5708.98	9114.183

We can plot this graph:



From the graph, we observe that as the number of threads increases, the average processing time decreases, indicating better performance. The most significant performance improvement occurs when moving from two threads to around eight threads. After that, the performance gain starts to plateau. Notably, as we go beyond 10 threads, the speedup starts to decrease, and efficiency drops significantly. This suggests that adding too many threads introduces overheads that outweigh the benefits of parallelism for this program.