INFO 6205 - Program Structures and Algorithms
Spring 2024

NAME: Neha Devarapalli
NUID: 002883137
GITHUB LINK: https://github.com/nehadevarapalli/INFO6205

Task Assignment 5 (Parallel Sorting) :
Conduct experiments to assess the efficacy of the parallel sorting method. Experiment with varying array sizes and cutoff values. Analyze scalability with available threads and assess the impact of array size on sorting time.

Relationship Conclusion:
Based on the outputs, we can observe that as the degree of parallelism increases (from 2 to 64), the sorting time generally decreases for all cutoff values. This reduction in sorting time indicates that utilizing more threads concurrently leads to faster sorting of the array.

For each degree of parallelism, there seems to be an optimal cutoff value that minimizes the sorting time. For example, when the array length is 200000 with a degree of parallelism of 8, the lowest sorting time (62ms) is achieved when the cutoff value is 10000. This suggests that selecting an appropriate cutoff value based on the available threads can significantly improve the efficiency of parallel sorting.

However, while increasing the degree of parallelism initially leads to a significant reduction in sorting time, the improvement becomes less substantial as the degree of parallelism continues to increase. Beyond a certain point, further increasing the degree of parallelism may not result in significant improvements in sorting time and could even lead to diminishing return or increased overhead. Hence, it is important to strike a balance and avoid excessive parallelism, which may lead to diminishing returns or increased overhead.

For recursion depth (D) and number of available threads (t), we have:

$$t = 2^D$$

Hence, the maximum depth (D) would be:

$$\log_2(N/V)$$

where N = array size
V = cutoff

Evidence:

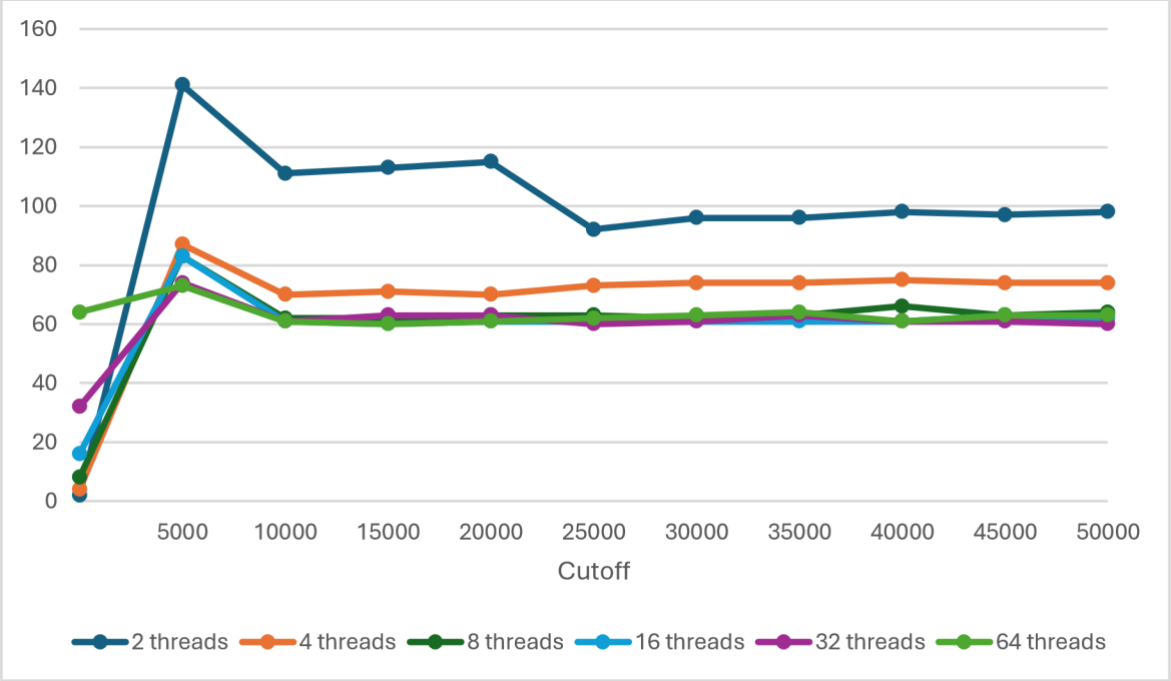
The runtimes (in milliseconds) and the respective graphs for different combinations of lengths of array sizes, number of threads and cutoffs are given as follows:

Array Length = 200000

Tabular Representation:

Cutoff	Number of Threads and Time (ms)					
	2	4	8	16	32	64
5000	141	87	83	83	74	73
10000	111	70	62	61	61	61
15000	113	71	62	60	63	60
20000	115	70	63	61	63	61
25000	92	73	63	61	60	62
30000	96	74	62	61	61	63
35000	96	74	63	61	63	64
40000	98	75	66	61	61	61
45000	97	74	63	63	61	63
50000	98	74	64	62	60	63

Graphical Representation of the above:

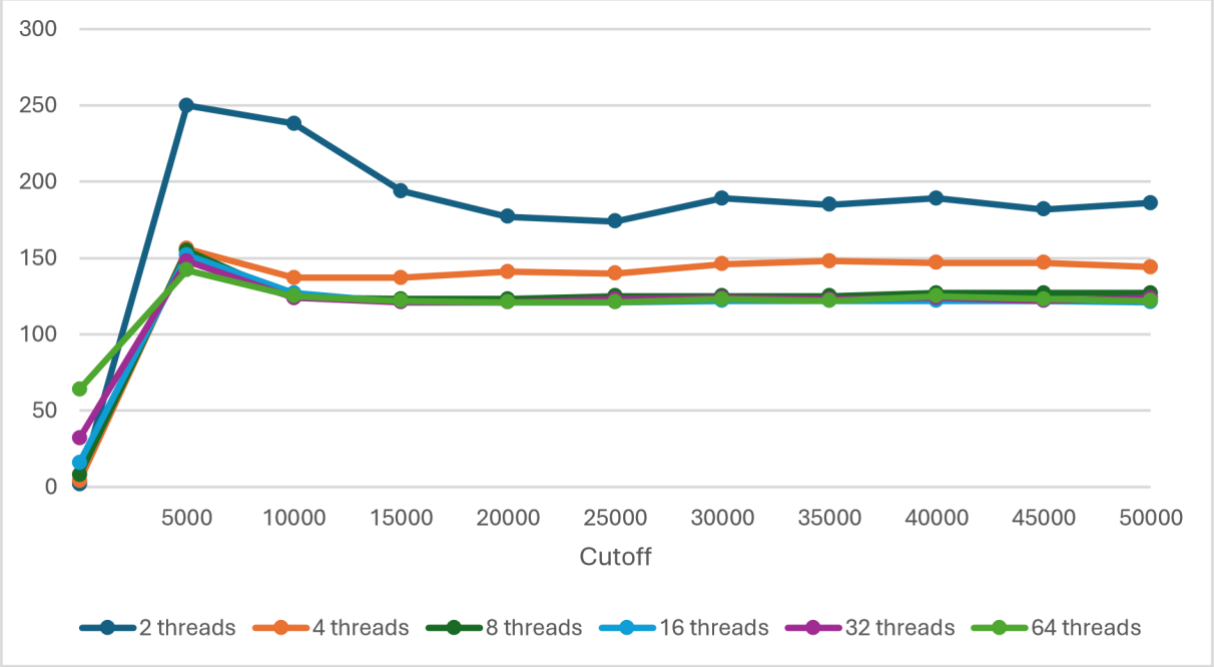


Array Length = 400000

Tabular Representation:

Cutoff	Number of Threads and Time (ms)					
	2	4	8	16	32	64
5000	250	156	155	152	148	142
10000	238	137	124	127	124	125
15000	194	137	123	121	121	122
20000	177	141	123	121	121	121
25000	174	140	125	121	123	121
30000	189	146	125	122	124	123
35000	185	148	125	122	123	122
40000	189	147	127	122	124	125
45000	182	147	127	122	122	123
50000	186	144	127	121	124	122

Graphical Representation:

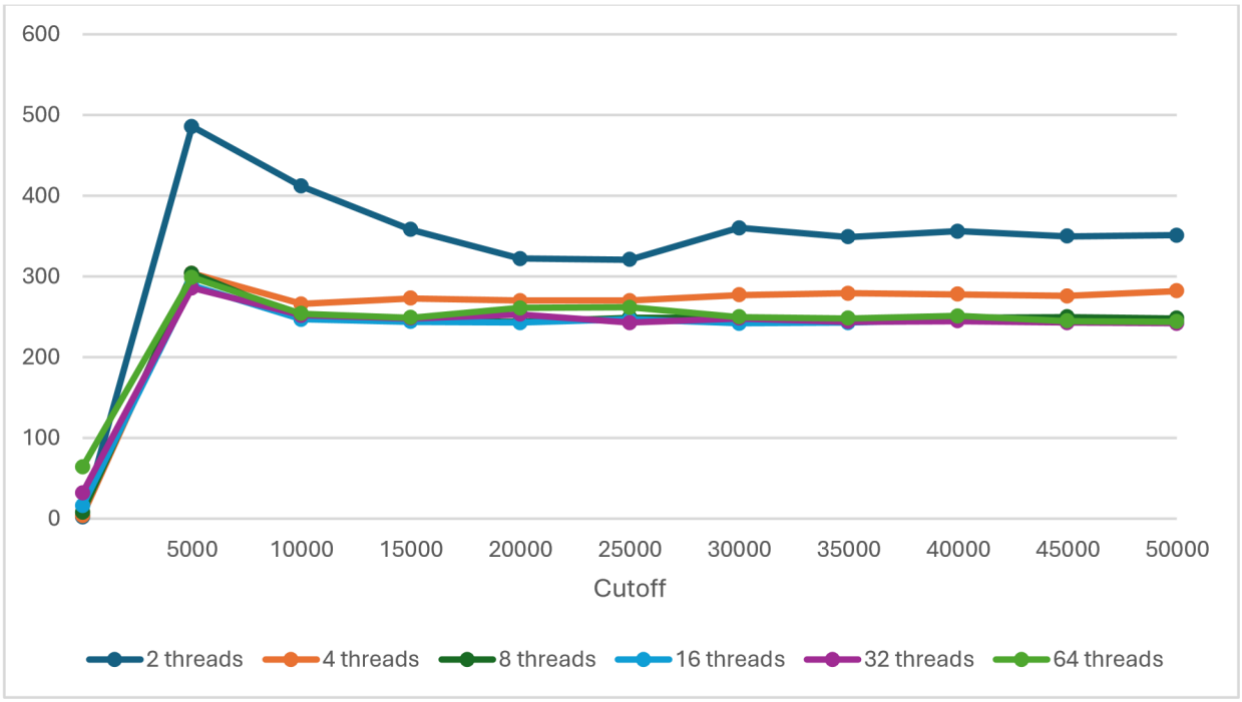


Array Length = 800000

Tabular Representation:

Cutoff	N ımber of Threads and Time (ms)					
	2	4	8	16	32	64
5000	486	304	304	289	286	299
10000	412	266	249	247	252	254
15000	358	273	245	244	248	249
20000	322	270	244	243	253	261
25000	321	270	249	247	243	262
30000	360	277	249	242	248	250
35000	349	279	246	243	244	248
40000	356	278	249	246	245	251
45000	350	276	250	244	243	245
50000	351	282	248	243	242	244

Graphical Representation:


600
500
400
300
200
100
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
Cutoff
2 threads
4 threads
8 threads
16 threads
32 threads
64 threads