Program Structures and Algorithms
Spring 2024

NAME: Meet Karnik
NUID: 002795334
GITHUB LINK: https://github.com/karnikmeet/INFO6205_meet

Task:

Implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

1. A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.
2. Recursion depth or the number of available threads. Using this determination, you might decide on an ideal number (t) of separate threads (stick to powers of 2) and arrange for that number of partitions to be parallelized (by preventing recursion after the depth of $lg\ t$ is reached).
3. An appropriate combination of these.

Relationship Conclusion:

- As the number of threads increase the amount of time taken to parallel sort is decreasing. This is happening across all array sizes(1000000 to 9000000) in my case.
- The observed cutoff values are all around or below 20 percent of the arraysize.
- The optimal thread count and array size for minimizing completion time can vary depending on the specific task and cutoff. For example, when the cutoff is 160000 , the optimal thread count and array size are different than when the cutoff time is 320000 .
- As the array size increases, the min_time generally increases as well, indicating that the program takes longer to complete the task when working with larger arrays.
- The program's performance is highly dependent on the specific configuration used, including the number of threads, array size, and cutoff time. As a result, optimizing the program's performance requires careful experimentation and tuning of these parameters

Evidence to support that conclusion:

The minimum time(miliseconds) to sort across all array size(1 to 9 million),threads(1,2,4,8(powers of 2)),cutoff (doubling 20000 onwards until less than array size)

min_time	threads_min _time	min_array_	cutoff%	cutoff
319.5	1	1000000	16	160000
316.9	1	3000000	5.33333333	160000

318.6	1	5000000	1.6	80000
312.3	1	7000000	1.14285714	80000
315.1	1	9000000	1.77777778	160000
290.5	2	1000000	16	160000
306.5	2	3000000	2.66666667	80000
303.3	2	5000000	3.2	160000
292.2	2	7000000	2.28571429	160000
292.7	2	9000000	1.77777778	160000
292.1	4	3000000	5.33333333	160000
285.5	4	5000000	3.2	160000
281.1	4	7000000	2.28571429	160000
288.1	4	9000000	1.77777778	160000
269.3	8	3000000	10.6666667	320000
269.4	8	5000000	6.4	320000
274.1	8	7000000	18.2857143	1280000
255.4	8	9000000	14.2222222	1280000

All the data observed across across all times (ms)

Arraysize	Threads	time in ms	cutoff
1000000.00	1	454	20000
1000000.00	1	325	40000
1000000.00	1	337	80000
1000000.00	1	319	160000
1000000.00	1	351	320000
1000000.00	1	472	640000
3000000.00	1	345	20000
3000000.00	1	332	40000
3000000.00	1	317	80000
3000000.00	1	316	160000
3000000.00	1	361	320000
3000000.00	1	439	640000
3000000.00	1	545	1280000
5000000.00	1	345	20000
5000000.00	1	334	40000
5000000.00	1	318	80000
5000000.00	1	323	160000
5000000.00	1	400	320000
5000000.00	1	480	640000

5000000.00	1	567	1280000
5000000.00	1	734	2560000
7000000.00	1	353	20000
7000000.00	1	338	40000
7000000.00	1	312	80000
7000000.00	1	324	160000
7000000.00	1	360	320000
7000000.00	1	448	640000
7000000.00	1	578	1280000
7000000.00	1	776	2560000
7000000.00	1	771	5120000
9000000.00	1	330	20000
9000000.00	1	345	40000
9000000.00	1	315	80000
9000000.00	1	315	160000
9000000.00	1	367	320000
9000000.00	1	452	640000
9000000.00	1	589	1280000
9000000.00	1	680	2560000
9000000.00	1	778	5120000
1000000.00	2	424	20000
1000000.00	2	345	40000
1000000.00	2	297	80000
1000000.00	2	290	160000
1000000.00	2	316	320000
1000000.00	2	357	640000
3000000.00	2	342	20000
3000000.00	2	319	40000
3000000.00	2	306	80000
3000000.00	2	312	160000
3000000.00	2	315	320000
3000000.00	2	397	640000
3000000.00	2	458	1280000
5000000.00	2	331	20000

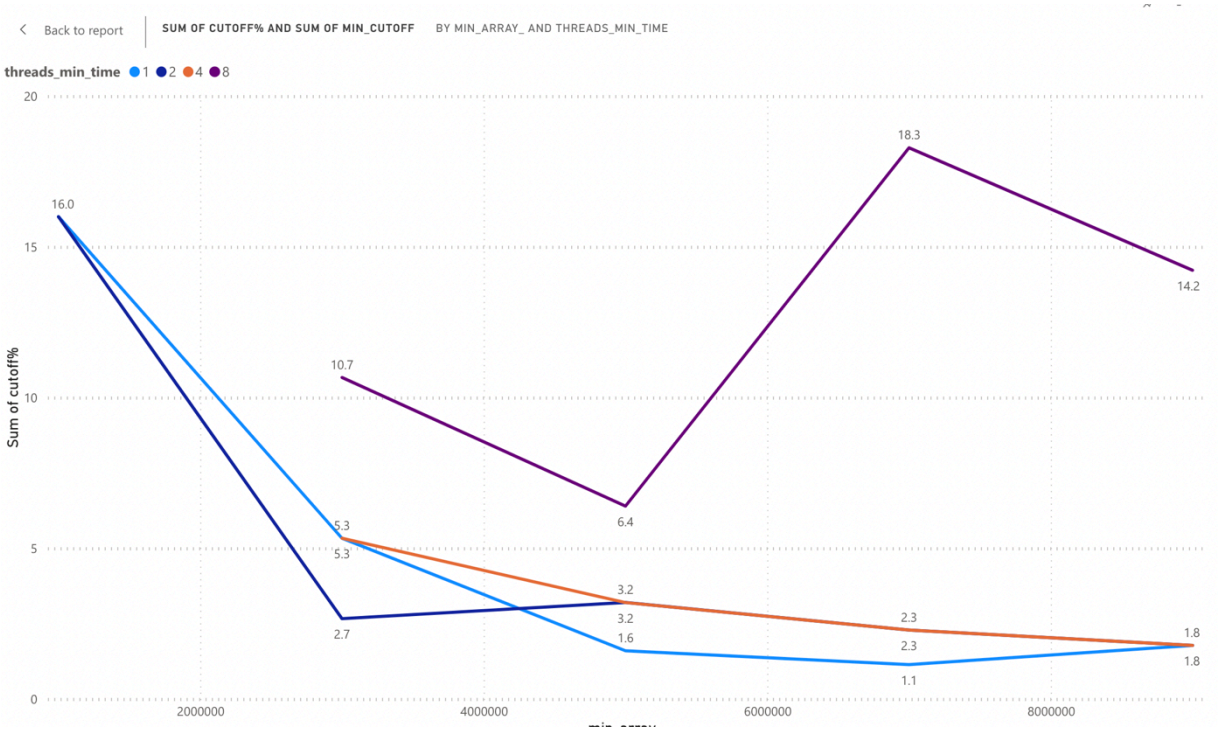
5000000.00	2	327	40000
5000000.00	2	310	80000
5000000.00	2	303	160000
5000000.00	2	311	320000
5000000.00	2	384	640000
5000000.00	2	420	1280000
5000000.00	2	488	2560000
7000000.00	2	341	20000
7000000.00	2	316	40000
7000000.00	2	301	80000
7000000.00	2	292	160000
7000000.00	2	308	320000
7000000.00	2	380	640000
7000000.00	2	406	1280000
7000000.00	2	456	2560000
7000000.00	2	435	5120000
9000000.00	2	327	20000
9000000.00	2	321	40000
9000000.00	2	302	80000
9000000.00	2	292	160000
9000000.00	2	318	320000
9000000.00	2	381	640000
9000000.00	2	426	1280000
9000000.00	2	507	2560000
9000000.00	2	449	5120000
1000000.00	4	353	20000
1000000.00	4	310	40000
1000000.00	4	317	80000
1000000.00	4	296	160000
1000000.00	4	290	320000
1000000.00	4	328	640000
3000000.00	4	355	20000
3000000.00	4	353	40000
3000000.00	4	304	80000
3000000.00	4	292	160000

3000000.00	4	299	320000
3000000.00	4	335	640000
3000000.00	4	377	1280000
3000000.00	4	334	2560000
5000000.00	4	306	20000
5000000.00	4	300	40000
5000000.00	4	290	80000
5000000.00	4	285	160000
5000000.00	4	288	320000
5000000.00	4	319	640000
5000000.00	4	343	1280000
5000000.00	4	336	2560000
7000000.00	4	347	20000
7000000.00	4	317	40000
7000000.00	4	294	80000
7000000.00	4	281	160000
7000000.00	4	283	320000
7000000.00	4	311	640000
7000000.00	4	341	1280000
7000000.00	4	303	2560000
7000000.00	4	437	5120000
9000000.00	4	309	20000
9000000.00	4	313	40000
9000000.00	4	301	80000
9000000.00	4	288	160000
9000000.00	4	315	320000
9000000.00	4	344	640000
9000000.00	4	329	1280000
9000000.00	4	304	2560000
9000000.00	4	449	5120000
1000000.00	8	349	20000
1000000.00	8	316	40000
1000000.00	8	293	80000
1000000.00	8	280	160000
1000000.00	8	269	320000

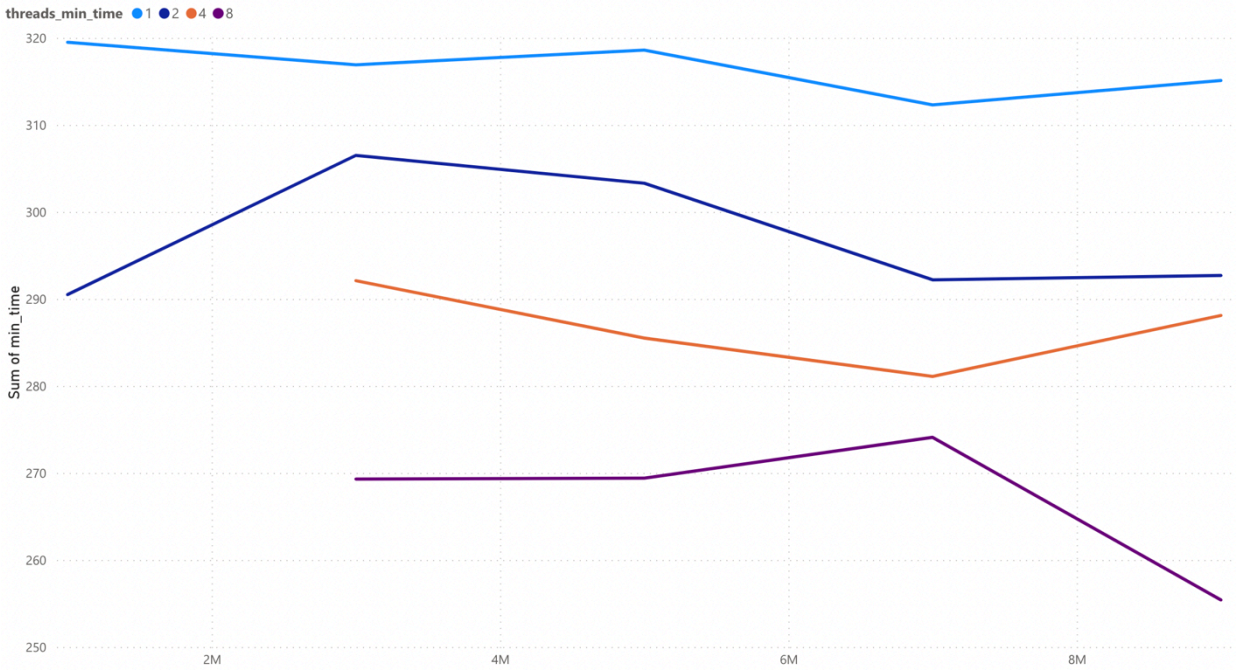
1000000.00	8	291	640000
3000000.00	8	333	20000
3000000.00	8	322	40000
3000000.00	8	291	80000
3000000.00	8	284	160000
3000000.00	8	269	320000
3000000.00	8	288	640000
3000000.00	8	282	1280000
5000000.00	8	339	20000
5000000.00	8	310	40000
5000000.00	8	296	80000
5000000.00	8	277	160000
5000000.00	8	269	320000
5000000.00	8	280	640000
5000000.00	8	273	1280000
5000000.00	8	336	2560000
7000000.00	8	331	20000
7000000.00	8	318	40000
7000000.00	8	291	80000
7000000.00	8	278	160000
7000000.00	8	276	320000
7000000.00	8	294	640000
7000000.00	8	274	1280000
7000000.00	8	328	2560000
7000000.00	8	436	5120000
9000000.00	8	341	20000
9000000.00	8	318	40000
9000000.00	8	283	80000
9000000.00	8	278	160000
9000000.00	8	274	320000
9000000.00	8	287	640000
9000000.00	8	255	1280000
9000000.00	8	324	2560000
9000000.00	8	452	5120000

Cutoff % vs Array size across 1,2,4,and 8 threads

The percentages do not seems to cross 20%. So less than 20% is the cutoff margin.



time in miliseconds vs Array size across 1,2,4,and 8 threads



Unit Test Screenshots:

NIL for this assignment