Danlin Lu (001567176)

Program Structures & Algorithms

Fall 2021

Assignment No. 5

- **Task**

  Your task is to 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.

  You must prepare a report that shows the results of your experiments and draws a conclusion (or more) about the efficacy of this method of parallelizing sort. Your experiments should involve sorting arrays of sufficient size for the parallel sort to make a difference. You should run with many different array sizes

(they must be sufficiently large to make parallel sorting worthwhile, obviously) and different cutoff schemes.

- **Relationship Conclusion:**

1. When cutoff ratio(cutoff/size) chosen is quite small, around 0.1, we do not need to use sort in parallel as line programming performs well. However, when the ratio gets larger, parallel sorting can be a good choice.
2. No matter what size is an array and no matter how many numbers of available threads, the roughly good range of the cutoff is from 0.15 to 0.5. And 0.3 seems a general good choice for the cutoff.
3. The better number of available threads depends on the size of arrays. For the arrays with large size, like 8 million, 8 threads can be a better choice of parallel sorting; while for that with small size, like 1 million, 16 threads can be a better choice.

- **Evidence to support the conclusion:**

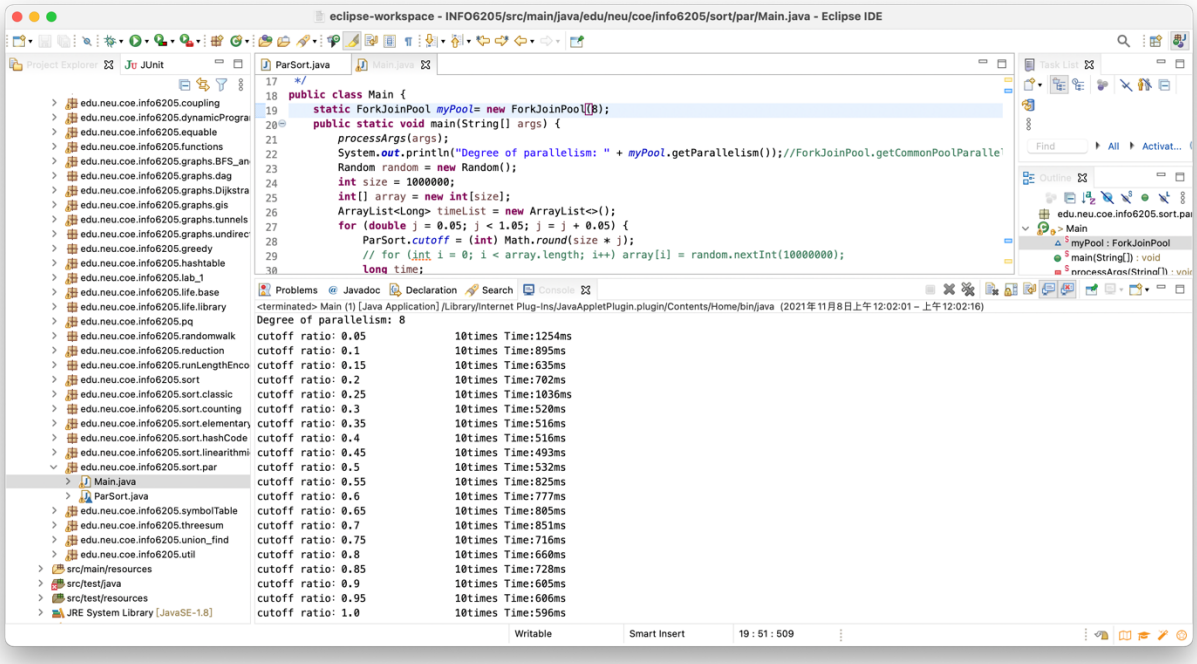
1. **Output**

Array sizes chosen are: 1 million, 2 million, 3 million, 4 million, 5 million, 6 million, 7 million, 8 million.

Cutoff schemes are: 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.1.

Number of threads chosen are: 1 (non-parallel), 2, 4, 8, 16, 32, 64

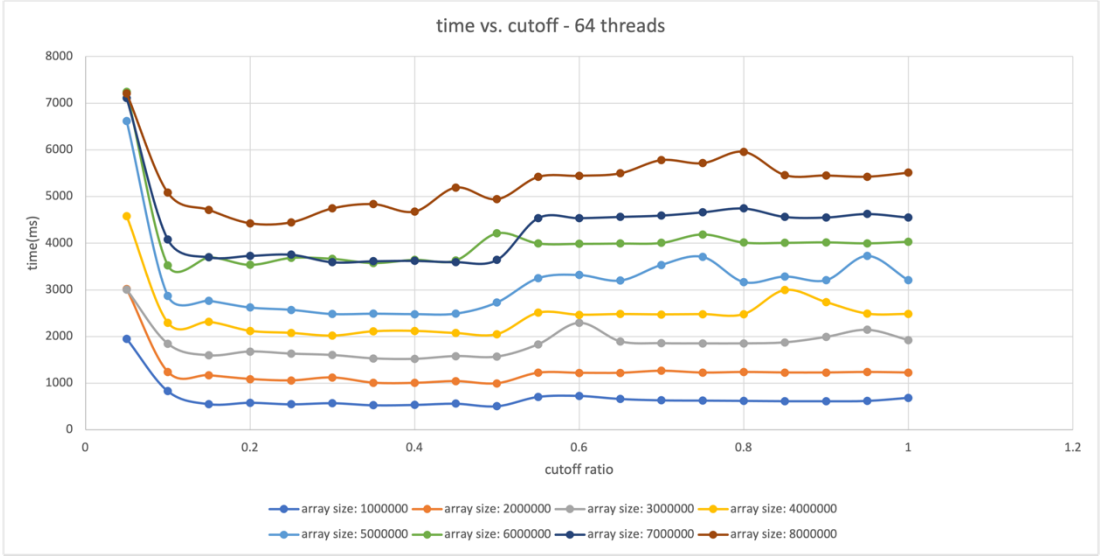
Here is one sample with 8 threads and 1 million size to show my code can run successfully:



2. Graphical Representation

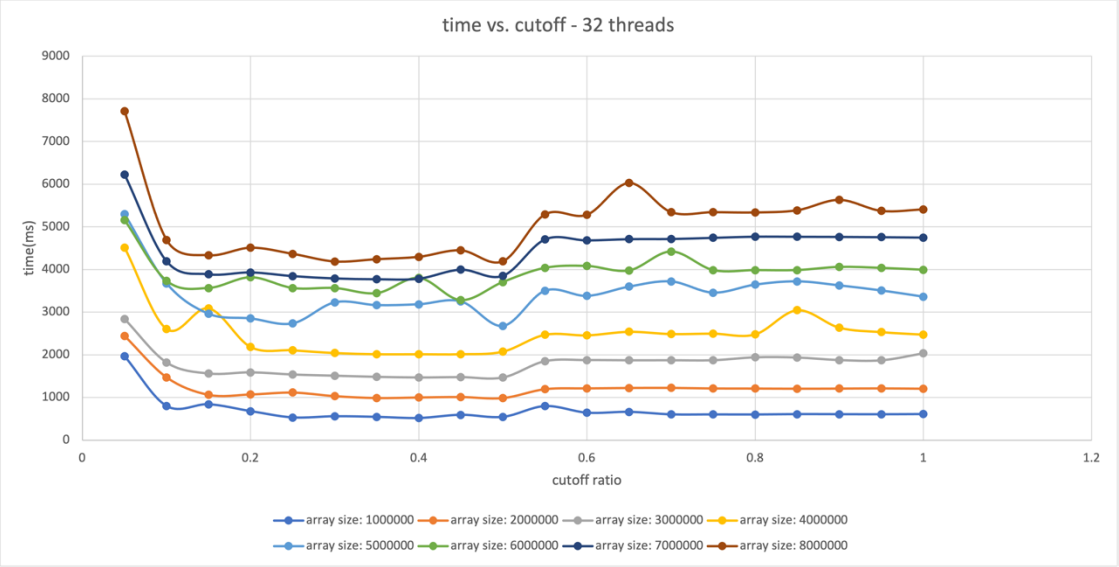
Firstly, I fix the number of threads to find good value of the cutoff for arrays with different sizes. Here are the results:

64 threads								
ratio	array size: 1000000	array size: 2000000	array size: 3000000	array size: 4000000	array size: 5000000	array size: 6000000	array size: 7000000	array size: 8000000
0.05	1950	3017	3005	4579	6615	7246	7112	7211
0.1	828	1242	1844	2296	2870	3526	4079	5082
0.15	551	1168	1596	2313	2765	3697	3699	4714
0.2	581	1088	1677	2121	2621	3536	3730	4427
0.25	548	1057	1631	2077	2571	3685	3754	4447
0.3	572	1120	1602	2018	2481	3665	3593	4747
0.35	527	1008	1529	2112	2489	3575	3615	4837
0.4	535	1004	1519	2120	2479	3644	3623	4678
0.45	561	1042	1579	2074	2490	3631	3596	5193
0.5	506	996	1568	2050	2729	4212	3642	4943
0.55	707	1222	1831	2509	3249	3995	4538	5421
0.6	728	1220	2293	2466	3317	3986	4538	5445
0.65	662	1220	1895	2481	3199	3993	4565	5498
0.7	633	1264	1855	2473	3534	4006	4594	5782
0.75	628	1225	1851	2480	3705	4188	4663	5716
0.8	622	1238	1851	2480	3166	4014	4745	5960
0.85	615	1227	1873	2996	3284	4008	4564	5460
0.9	614	1227	1990	2737	3205	4019	4552	5452
0.95	621	1238	2143	2490	3726	3997	4627	5426
1	687	1226	1924	2483	3205	4032	4551	5512



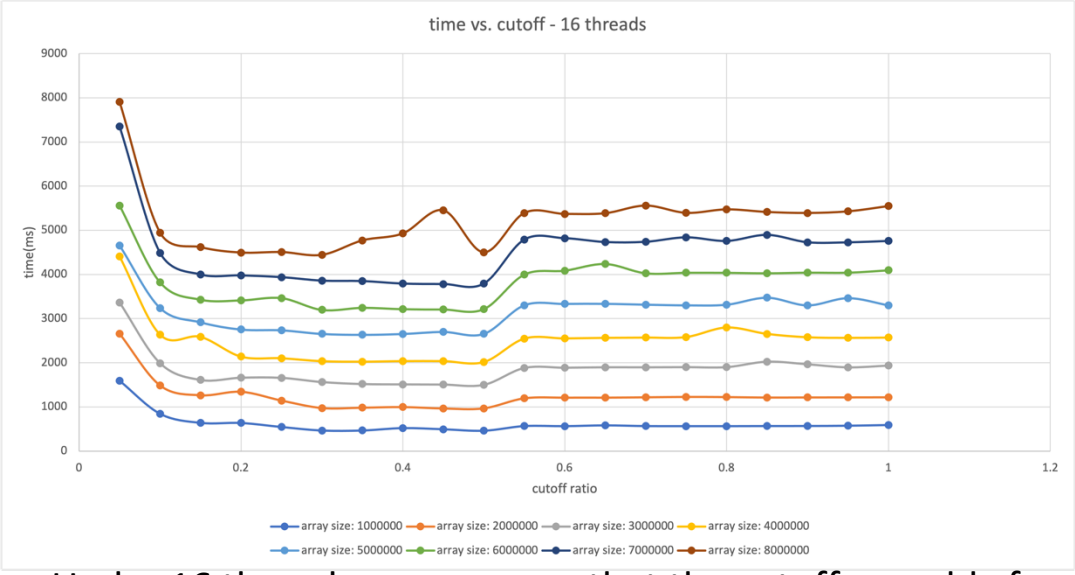
Under 64 threads, you can see that the cutoffs roughly from 0.15 to 0.5 achieve the best performance, more accurately, 0.3.

32 threads								
ratio	array size: 1000000	array size: 2000000	array size: 3000000	array size: 4000000	array size: 5000000	array size: 6000000	array size: 7000000	array size: 8000000
0.05	1970	2445	2841	4514	5297	5155	6228	7711
0.1	799	1471	1818	2606	3673	3732	4191	4689
0.15	840	1061	1560	3091	2961	3566	3888	4333
0.2	678	1070	1588	2183	2852	3816	3932	4509
0.25	530	1115	1536	2108	2737	3564	3845	4365
0.3	558	1030	1508	2043	3230	3564	3790	4186
0.35	543	986	1481	2015	3164	3444	3771	4240
0.4	516	999	1466	2015	3185	3801	3784	4295
0.45	592	1010	1475	2015	3252	3280	3996	4447
0.5	543	985	1466	2076	2677	3706	3852	4191
0.55	797	1194	1850	2471	3503	4037	4706	5290
0.6	641	1213	1876	2457	3381	4081	4683	5282
0.65	661	1223	1871	2541	3598	3976	4712	6027
0.7	602	1225	1872	2488	3720	4420	4716	5343
0.75	601	1211	1872	2498	3457	3985	4743	5343
0.8	599	1209	1941	2480	3645	3984	4770	5336
0.85	608	1205	1936	3046	3722	3985	4768	5387
0.9	606	1208	1876	2638	3627	4059	4765	5631
0.95	604	1213	1870	2533	3508	4036	4758	5373
1	609	1205	2034	2474	3363	3992	4749	5407



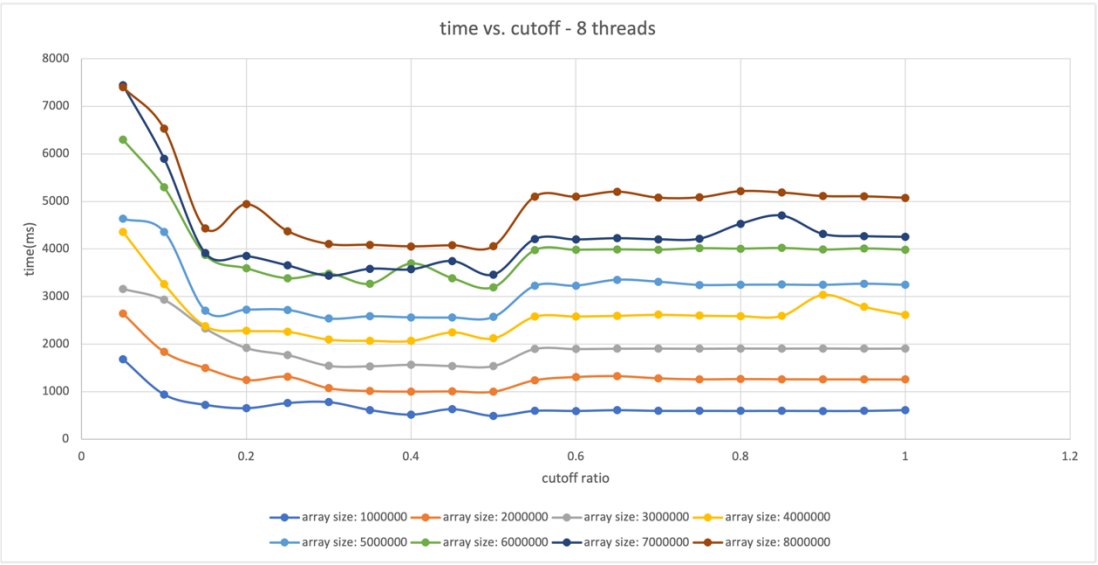
Under 32 threads, you can see that the cutoffs roughly from 0.15 to 0.5 achieve the best performance, more accurately, 0.25.

16 threads								
ratio	array size: 1000000	array size: 2000000	array size: 3000000	array size: 4000000	array size: 5000000	array size: 6000000	array size: 7000000	array size: 8000000
0.05	1592	2656	3363	4405	4654	5559	7352	7907
0.1	845	1486	1991	2637	3240	3821	4489	4947
0.15	640	1264	1612	2587	2917	3430	3999	4618
0.2	637	1343	1662	2140	2754	3414	3978	4495
0.25	546	1144	1659	2104	2737	3466	3938	4507
0.3	466	975	1564	2037	2655	3198	3859	4444
0.35	467	983	1519	2026	2635	3246	3849	4771
0.4	521	997	1511	2039	2654	3215	3792	4929
0.45	495	965	1507	2039	2701	3208	3782	5452
0.5	462	968	1501	2018	2656	3216	3791	4501
0.55	567	1195	1883	2548	3303	3998	4790	5392
0.6	565	1210	1889	2552	3334	4087	4820	5371
0.65	583	1210	1899	2565	3336	4238	4734	5386
0.7	567	1218	1898	2571	3317	4027	4739	5559
0.75	565	1227	1902	2583	3302	4039	4840	5394
0.8	565	1223	1902	2799	3316	4038	4758	5473
0.85	568	1212	2025	2654	3474	4027	4896	5414
0.9	569	1215	1966	2578	3301	4042	4728	5392
0.95	575	1216	1897	2565	3464	4041	4727	5428
1	591	1218	1937	2571	3302	4095	4758	5548



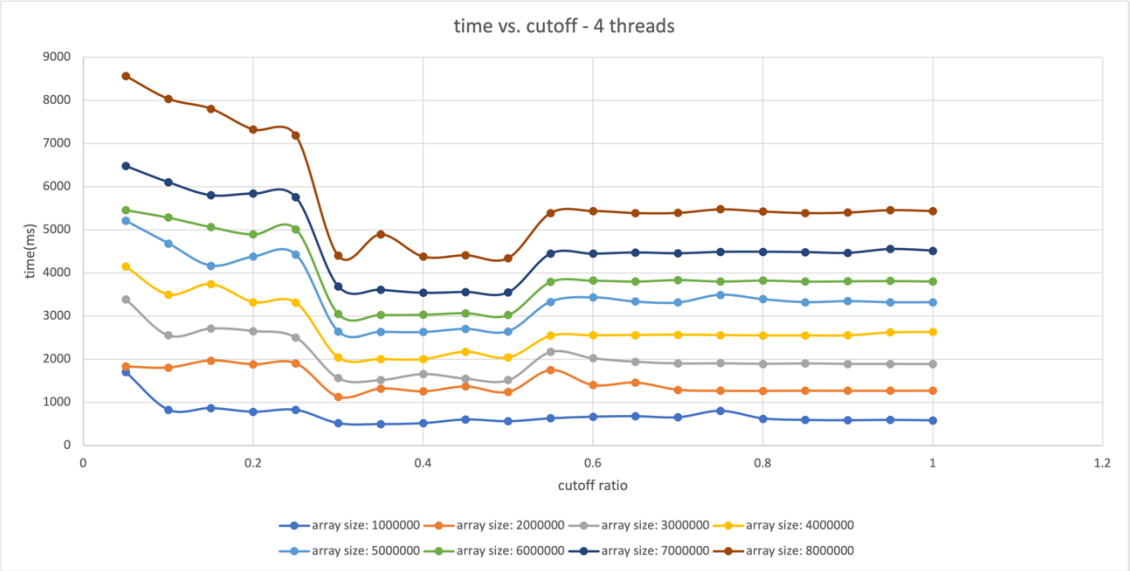
Under 16 threads, you can see that the cutoffs roughly from 0.15 to 0.5 achieve the best performance, more accurately, 0.3.

8 threads								
ratio	array size: 1000000	array size: 2000000	array size: 3000000	array size: 4000000	array size: 5000000	array size: 6000000	array size: 7000000	array size: 8000000
0.05	1683	2645	3163	4361	4635	6300	7442	7401
0.1	942	1838	2932	3262	4362	5297	5898	6529
0.15	724	1498	2330	2379	2705	3872	3914	4433
0.2	654	1244	1919	2281	2725	3594	3854	4943
0.25	762	1315	1772	2261	2720	3384	3655	4375
0.3	781	1073	1544	2095	2538	3483	3440	4105
0.35	615	1013	1535	2071	2586	3267	3585	4089
0.4	518	1002	1567	2068	2563	3695	3573	4055
0.45	633	1007	1540	2249	2559	3386	3748	4079
0.5	490	1000	1539	2122	2572	3196	3462	4059
0.55	598	1241	1899	2578	3226	3976	4210	5103
0.6	595	1310	1897	2582	3230	3984	4200	5102
0.65	612	1328	1905	2596	3354	3993	4228	5206
0.7	597	1282	1908	2619	3313	3985	4205	5079
0.75	597	1260	1904	2600	3245	4017	4217	5086
0.8	596	1267	1909	2589	3249	4007	4532	5216
0.85	597	1262	1907	2593	3252	4022	4703	5188
0.9	595	1260	1910	3036	3246	3988	4317	5113
0.95	596	1259	1905	2785	3271	4012	4271	5107
1	613	1257	1908	2615	3247	3986	4256	5077



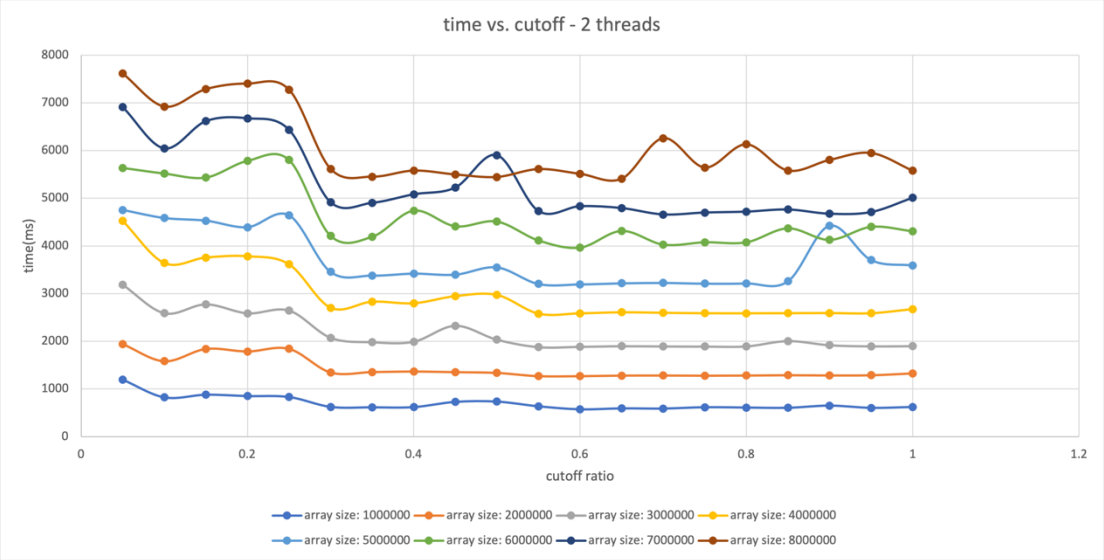
Under 8 threads, you can see that the cutoffs roughly from 0.2 to 0.5 achieve the best performance, more accurately, 0.4.

4 threads								
ratio	array size: 1000000	array size: 2000000	array size: 3000000	array size: 4000000	array size: 5000000	array size: 6000000	array size: 7000000	array size: 8000000
0.05	1696	1831	3382	4145	5211	5455	6474	8558
0.1	826	1804	2551	3500	4688	5281	6101	8033
0.15	864	1969	2710	3741	4172	5061	5798	7804
0.2	779	1879	2653	3324	4380	4891	5835	7321
0.25	823	1903	2503	3311	4428	5016	5751	7184
0.3	514	1128	1559	2040	2643	3049	3685	4399
0.35	493	1318	1517	2002	2633	3026	3606	4893
0.4	515	1254	1656	2001	2627	3031	3535	4379
0.45	600	1366	1550	2172	2701	3068	3556	4410
0.5	559	1241	1512	2037	2646	3021	3547	4345
0.55	627	1745	2167	2542	3334	3790	4446	5388
0.6	664	1399	2022	2552	3434	3820	4442	5435
0.65	676	1457	1941	2560	3335	3800	4471	5387
0.7	651	1289	1902	2569	3314	3835	4452	5390
0.75	801	1266	1905	2558	3495	3801	4485	5473
0.8	625	1264	1894	2549	3391	3824	4488	5423
0.85	589	1268	1903	2549	3322	3800	4480	5386
0.9	582	1267	1893	2551	3346	3806	4462	5398
0.95	590	1265	1892	2620	3320	3812	4551	5455
1	580	1268	1889	2630	3320	3802	4516	5433



Under 4 threads, you can see that the cutoffs roughly from 0.25 to 0.5 achieve the best performance, more accurately, 0.3.

2 threads								
ratio	array size: 1000000	array size: 2000000	array size: 3000000	array size: 4000000	array size: 5000000	array size: 6000000	array size: 7000000	array size: 8000000
0.05	1195	1942	3183	4524	4753	5635	6911	7615
0.1	824	1582	2588	3641	4584	5517	6044	6922
0.15	879	1840	2773	3756	4529	5433	6616	7287
0.2	851	1787	2586	3782	4392	5782	6676	7403
0.25	831	1842	2643	3614	4641	5801	6436	7277
0.3	624	1346	2074	2703	3456	4211	4914	5615
0.35	615	1355	1981	2830	3379	4192	4902	5451
0.4	619	1366	1992	2797	3421	4740	5081	5579
0.45	729	1353	2325	2946	3399	4412	5223	5498
0.5	735	1337	2034	2972	3551	4511	5903	5445
0.55	636	1269	1879	2579	3203	4115	4729	5611
0.6	573	1269	1885	2586	3195	3965	4831	5511
0.65	593	1281	1898	2608	3217	4315	4792	5408
0.7	589	1283	1893	2597	3226	4028	4658	6259
0.75	617	1279	1890	2588	3210	4077	4698	5643
0.8	610	1282	1894	2587	3215	4076	4717	6130
0.85	605	1289	2005	2589	3260	4369	4762	5580
0.9	651	1284	1918	2592	4422	4127	4673	5802
0.95	604	1288	1895	2588	3704	4400	4708	5948
1	622	1327	1899	2674	3591	4309	5008	5577

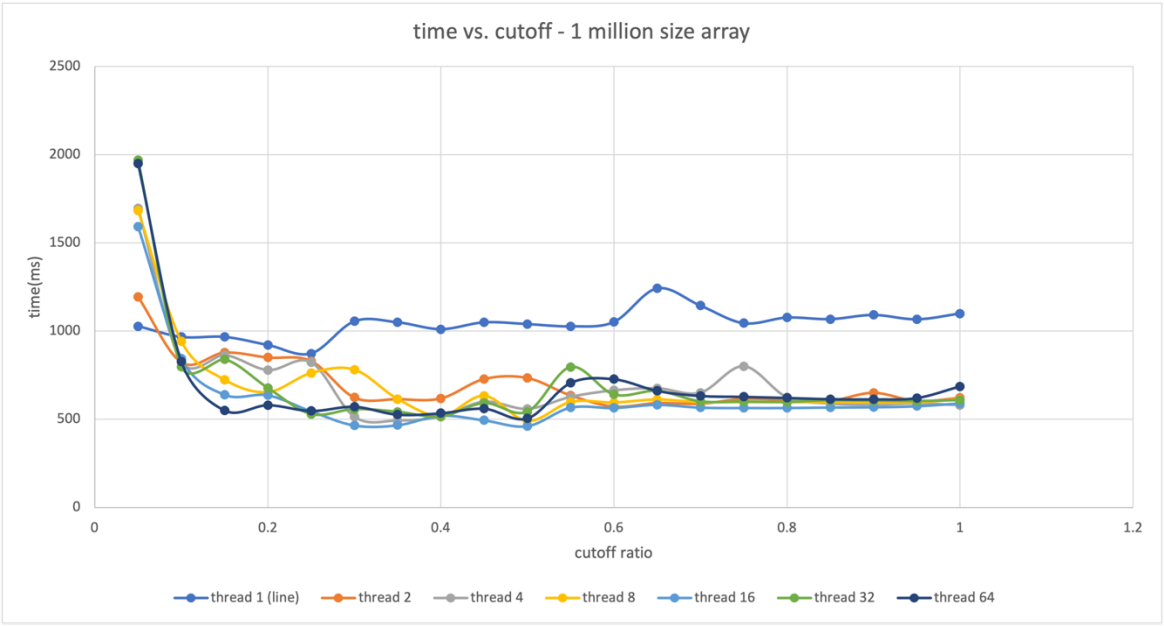


Under 2 threads, you can see that the cutoffs roughly from 0.25 to 0.5 achieve the best performance, more accurately, 0.3.

In conclusion, the roughly good range of the cutoff is from 0.15 to 0.5. And 0.3 seems a general good choice for the cutoff.

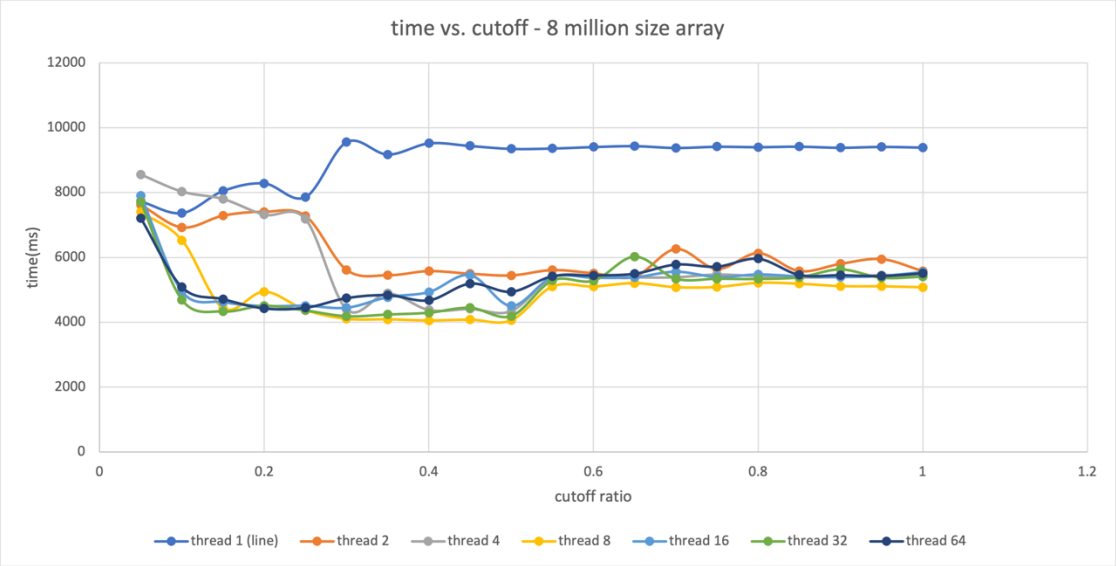
Secondly, to find better number of available threads, I fix the size of array. And to test whether the number of threads is related to the size, I choose two sizes: 1 million and 8 million.

The result of 1 million array:



We can find that 16 threads performs better than others.

The result of 8 million array:



We can find that 8 threads performs better than others.

And from those two images, we can find that when cutoff is small enough, around 0.1, we do not need to choose parallel sorting.