Yanhong Chen (NUID: 001814656)

INFO 6205

Program Structures & Algorithms

Spring 2020

Assignment 4

- **Task**

This task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. 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.

- **Output** (few outputs to prove relationship)

Performance based on different cutoffs and arrays sizes (time: millisecond)

Cutoff	2*10^4	4*10^5	8*10^5	2*10^6	4*10^6
10000	105	490	934	1922	2894
11000	36	316	473	1016	1975
12000	39	245	473	1014	2005
13000	34	224	419	986	1880
14000	10	231	414	962	1879
15000	9	227	456	946	2053
16000	26	234	460	808	2047
32000	23	287	533	890	2030
64000	21	192	463	725	2208
128000	12	207	345	871	2361
256000	14	242	322	880	2344
512000	18	312	426	760	2565
600000	24	342	487	870	2637
700000	18	319	450	734	2737

Yanhong Chen (NUID: 001814656)

Performance based on different threads

Cutoff	2 Threads	4 Threads	8 Threads	16 Threads
510000	149ms	120ms	128ms	124ms
520000	107ms	84ms	84ms	83ms
530000	112ms	75ms	75ms	73ms
540000	108ms	78ms	77ms	77ms
550000	107ms	79ms	76ms	75ms
560000	106ms	76ms	74ms	73ms
570000	103ms	78ms	74ms	72ms
580000	103ms	82ms	79ms	77ms
590000	96ms	85ms	84ms	81ms
600000	89ms	75ms	78ms	72ms

- **Relationship conclusion**

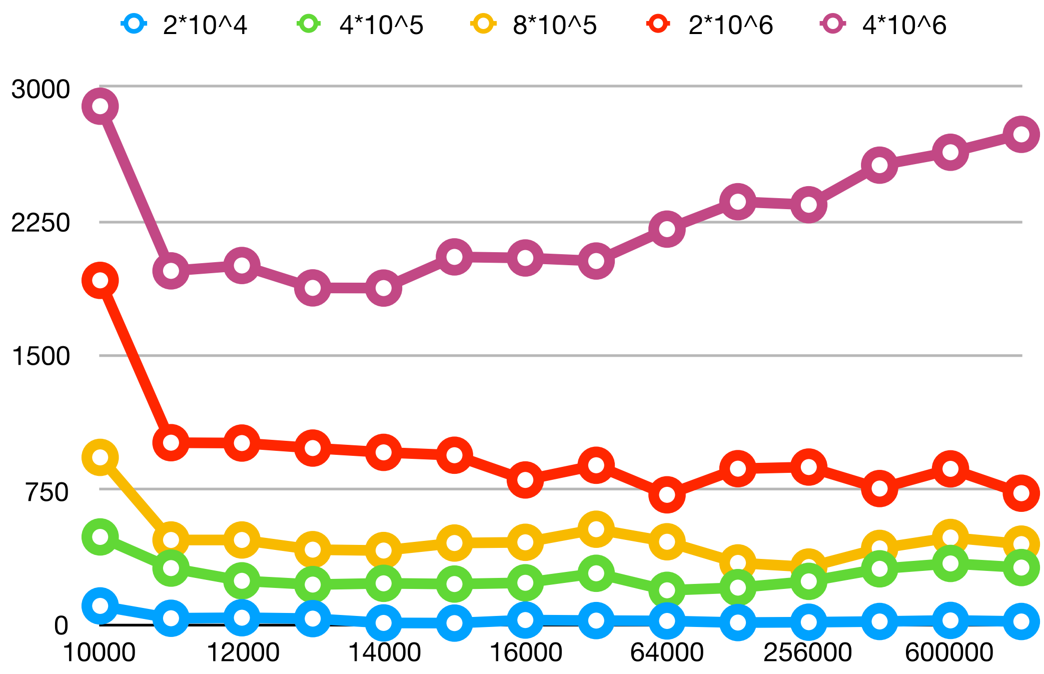
1. For lower cutoff values, the system sort is more efficient than the parallel sort.

2. When we sort a small array, the size of the cutoff has little effect on our efficiency. But when we gradually increase the size of the random array, the larger the cutoff size will cause the sorting efficiency to become lower, which means that when we sorting larger data, we need to split it into smaller arrays by parallel sort and then sort it by system sort, this is much more efficient way.

3. When I increased the thread from 2 to 4, it showed a significant reduction in runtime. which means multiple threads bring better performers. but when I gradually increased thread to 8 and 16, the runtime remains the same with 4 threads. Probably because there will be a certain cost when switching execution between multiple threads.

- **Evidence to support relationship** (screen shot and/or graph and/or spreadsheet)

Yanhong Chen (NUID: 001814656)


2*10^4
4*10^5
8*10^5
2*10^6
4*10^6
3000
2250
1500
750
0
10000
12000
14000
16000
64000
256000
600000