Bowen Jiang (NUID: 001582174)

INFO 6205

Program Structures & Algorithms

Fall 2020

Assignment 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.

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.

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).

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

With different arraysize and under the same degree of parallelism the out put shows below:

```
Degree of parallelism: 11
The size of Array = 500000
cutoff : 2500        10times Time:425ms
cutoff : 5000        10times Time:323ms
cutoff : 10000       10times Time:188ms
cutoff : 20000       10times Time:228ms
cutoff : 40000       10times Time:241ms
cutoff : 80000       10times Time:162ms
cutoff : 160000      10times Time:232ms
cutoff : 320000      10times Time:253ms
cutoff : 640000      10times Time:397ms
cutoff : 1280000     10times Time:397ms
```

```
Degree of parallelism: 11
The size of Array = 1000000
cutoff : 5000        10times Time:854ms
cutoff : 10000       10times Time:508ms
cutoff : 20000       10times Time:355ms
cutoff : 40000       10times Time:375ms
cutoff : 80000       10times Time:400ms
cutoff : 160000      10times Time:335ms
cutoff : 320000      10times Time:372ms
cutoff : 640000      10times Time:536ms
cutoff : 1280000     10times Time:841ms
cutoff : 2560000     10times Time:844ms
```

```
Degree of parallelism: 11
The size of Array = 2000000
cutoff : 10000       10times Time:1364ms
cutoff : 20000       10times Time:779ms
cutoff : 40000       10times Time:714ms
cutoff : 80000       10times Time:770ms
cutoff : 160000      10times Time:715ms
cutoff : 320000      10times Time:697ms
cutoff : 640000      10times Time:810ms
cutoff : 1280000     10times Time:1100ms
cutoff : 2560000     10times Time:1756ms
cutoff : 5120000     10times Time:1760ms
```

```
Degree of parallelism: 11
The size of Array = 4000000
cutoff : 20000       10times Time:2015ms
cutoff : 40000       10times Time:1693ms
cutoff : 80000       10times Time:1397ms
cutoff : 160000      10times Time:1423ms
cutoff : 320000      10times Time:1401ms
cutoff : 640000      10times Time:1373ms
cutoff : 1280000     10times Time:1573ms
cutoff : 2560000     10times Time:2198ms
cutoff : 5120000     10times Time:3625ms
cutoff : 10240000        10times Time:3696ms
```

With different threads number shows the relationship between cutoff and time, about thread = 16 we can find it performs best.

ArraySize = 1000000		
Threads	Cutoff/size	Time/ms
1	0.01	66.6
2	0.02	58.4
4	0.04	37.1
8	0.16	32.8
16	0.16	31.8
32	0.16	35.6
64	0.16	33.2
128	0.16	32.5
256	0.16	34.0
512	0.16	32.8

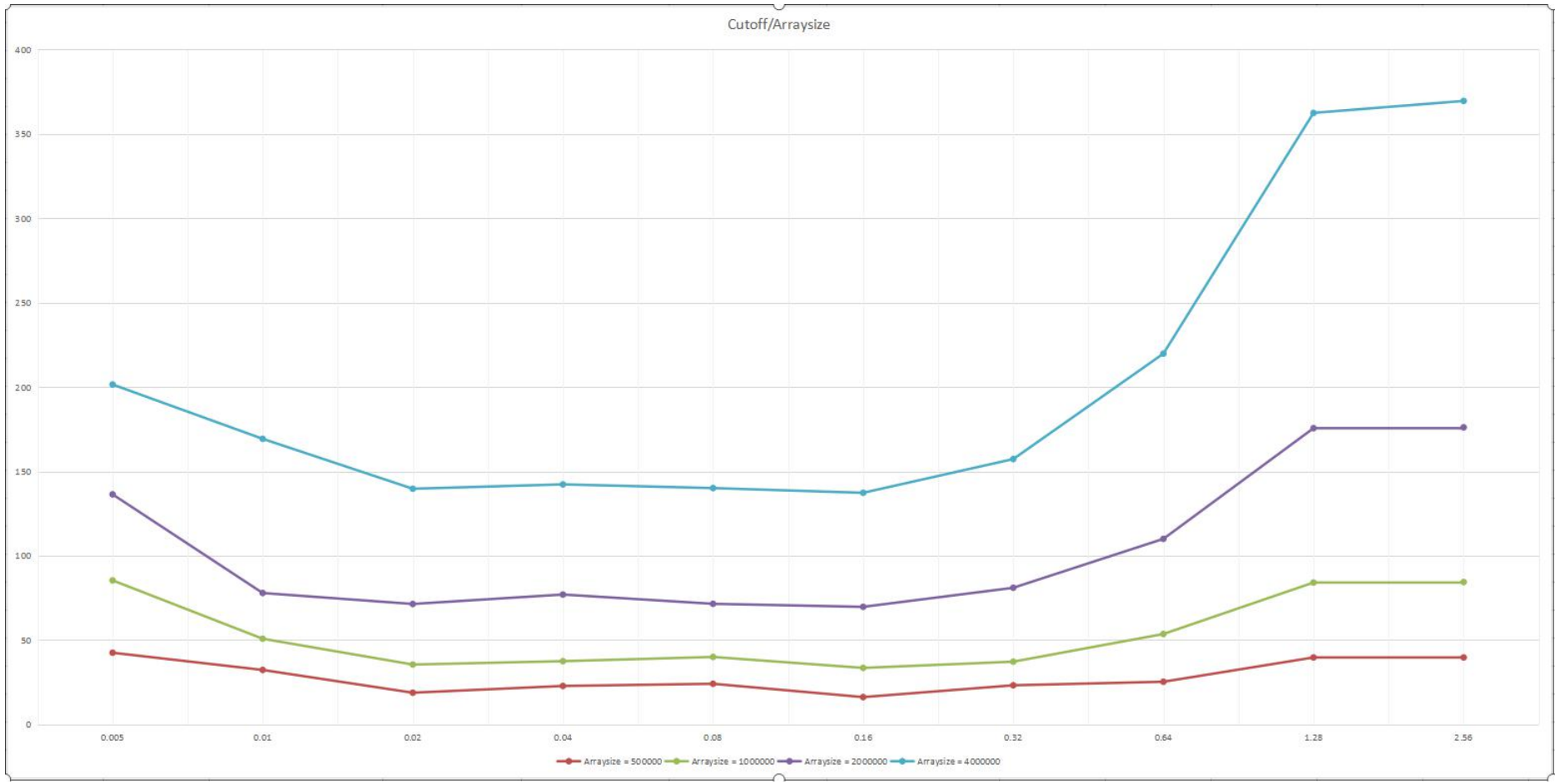
Relationship Conclusion

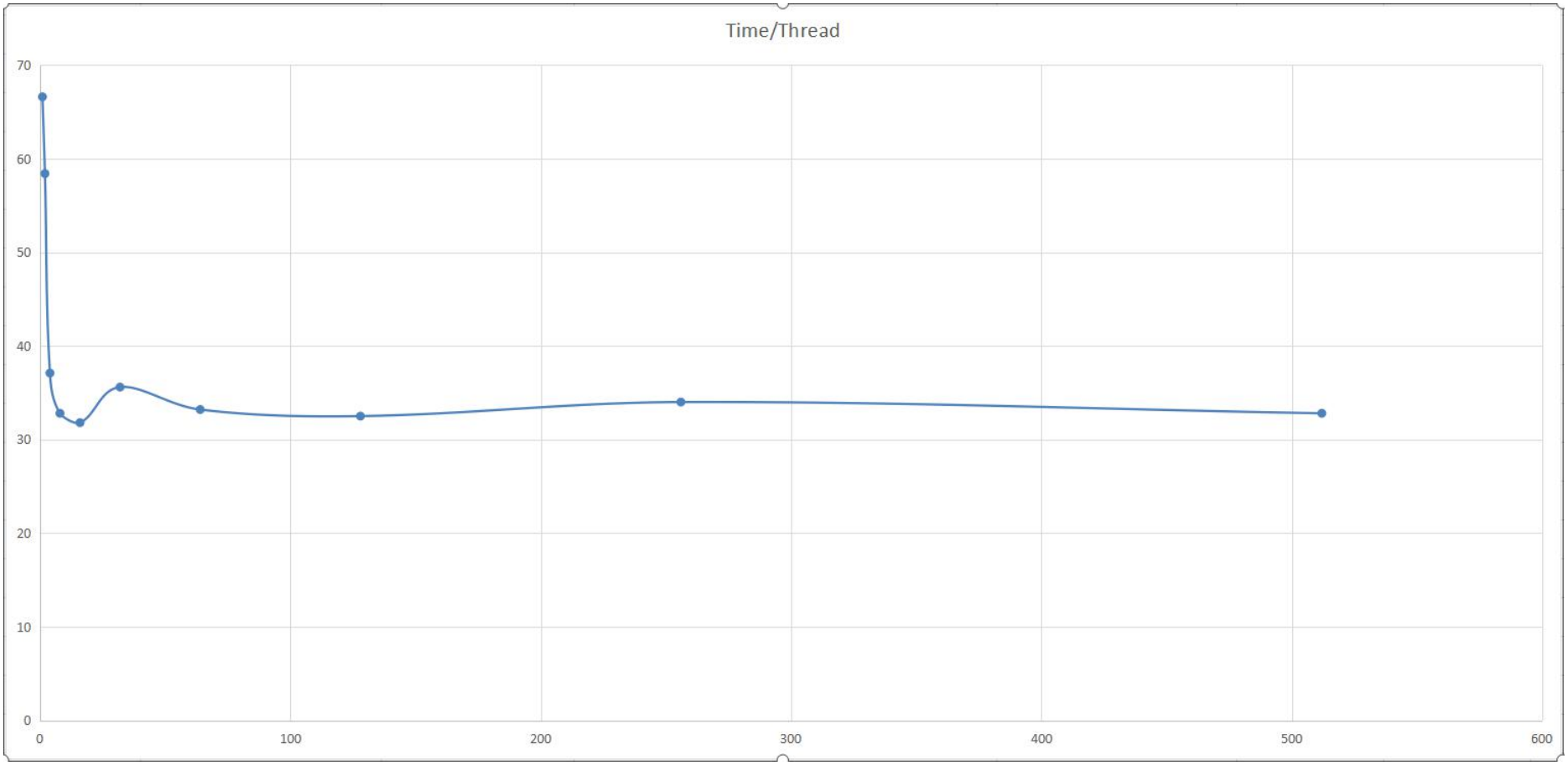
Conclusion: With the different value of Arraysize and Cutoff, the degree of parallelism = 11, on my laptop, about 16% - 32% of the Arraysize perform better than other times. And the most proper thread number is 16.

Evidence to support relationship

This table shows when Arraysize = {500000, 1000000, 2000000, 4000000}, the relationship between cutoff/Arraysize and different SortTime.

cutoff/arraysize	Arraysize = 500000	Arraysize = 1000000	Arraysize = 2000000	Arraysize = 4000000
0.005	42.5	85.4	136.4	201.5
0.01	32.3	50.8	77.9	169.3
0.02	18.8	35.5	71.4	139.7
0.04	22.8	37.5	77	142.3
0.08	24.1	40	71.5	140.1
0.16	16.2	33.5	69.7	137.3
0.32	23.2	37.2	81	157.3
0.64	25.3	53.6	110	219.8
1.28	39.7	84.1	175.6	362.5
2.56	39.7	84.4	176.1	369.6


Cutoff/Arraysize
400
350
300
250
200
150
100
50
0
0.005
0.01
0.02
0.04
0.08
0.16
0.32
0.64
1.28
2.56
Arraysize = 500000
Arraysize = 1000000
Arraysize = 2000000
Arraysize = 4000000


Time/Thread
70
60
50
40
30
20
10
0
0
100
200
300
400
500
600