Zixuan.Zhu (002196924)

**Program Structures & Algorithms**

**Fall 2021**

**Assignment No. 4**

* **Task**

Please see the presentation on Assignment on Parallel Sorting under the Exams. etc. module.

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).

An appropriate combination of these.

There is a Main class and the ParSort class in the sort.par package of the INFO6205 repository. The Main class can be used as is but the ParSort class needs to be implemented where you see "TODO..." [it turns out that these TODOs are already implemented].

Unless you have a good reason not to, you should just go along with the Java8-style future implementations provided for you in the class repository.

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:**

Firstly, we have to consider the scheme that when the length of array is less than the cutoff, we use system sort instead. Actually, it’s easy to make it. We can test different cutoffs but we have to know that the cutoff is related to the original length of array. I means that when the given array is huge, the cutoff we have to come up with is also big. So in my program, I pass the value of cutoff to Parsort’s static method to sort the randomly generated array. And the formula to set the cutoff is “cutoff = 10000 \* (j + 1)” (50<j<100).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2 thread | 4 thread | 8 thread | system sort |
| 0.005 | 149.8 | 136.5 | 131.9 | 243.9 |
| 0.01 | 131.6 | 105.5 | 99.9 | 243.1 |
| 0.015 | 139.7 | 97.5 | 107.6 | 248.6 |
| 0.02 | 141.4 | 104.9 | 99.2 | 245.8 |
| 0.025 | 140.7 | 106.7 | 99.9 | 269.8 |
| 0.03 | 128.4 | 106.4 | 95.7 | 271 |
| 0.035 | 136.3 | 97.8 | 94.5 | 258.9 |
| 0.04 | 147.2 | 102.7 | 97.4 | 256.8 |
| 0.045 | 134.7 | 111 | 106.7 | 255.7 |
| 0.05 | 137.2 | 102 | 103.3 | 249.1 |
| 0.055 | 129.2 | 97.2 | 97.6 | 247.4 |
| 0.06 | 127.5 | 97.5 | 95.5 | 248.6 |
| 0.065 | 130.2 | 98.1 | 95.8 | 272.2 |
| 0.07 | 137.2 | 101.3 | 94.8 | 287.2 |
| 0.075 | 137 | 97.8 | 94.4 | 238.5 |
| 0.08 | 131.4 | 99.2 | 106.3 | 257.9 |

This is the result I got. And you can see that the second column is the result when the threadcount is 8, the third one is the result when the count is 4,and finally the fourth is the result when the count is 2. The results of the second one and the third one are close. But the difference between the first one’s result and the second one’s is quite big.

Generally speaking, when the parallelism is bigger, the result should be more satisfying.

And I also show the efficiency of parsort so I use Arrays.sort() (system sort method) to make a comparison.

As you can see, parsort is two times faster than system sort. So we can see that we can deal with the problems using the parallelism scheme which is faster than the normal methods. And in some cases, when the length of array is small enough, just use the system sort rather than use a complicated method.

1. **Graphical Representation**

Graph:

* **Output:**

文本

描述已自动生成

* **Github Links:**

Main.java:

<https://github.com/ZixuanZhu-faye/INFO-6205/blob/bf2651b93b15557d8dfc38354c07607b9cde3e4a/INFO6205-Fall2021/src/main/java/edu/neu/coe/info6205/sort/par/Main.java>

ParSort.java:

<https://github.com/ZixuanZhu-faye/INFO-6205/blob/bf2651b93b15557d8dfc38354c07607b9cde3e4a/INFO6205-Fall2021/src/main/java/edu/neu/coe/info6205/sort/par/ParSort.java>