Kae-Yang Hsieh(001092745)

Program Structures & Algorithms

Fall 2021 Assignment No. 5, Parallel Sorting

Tasks

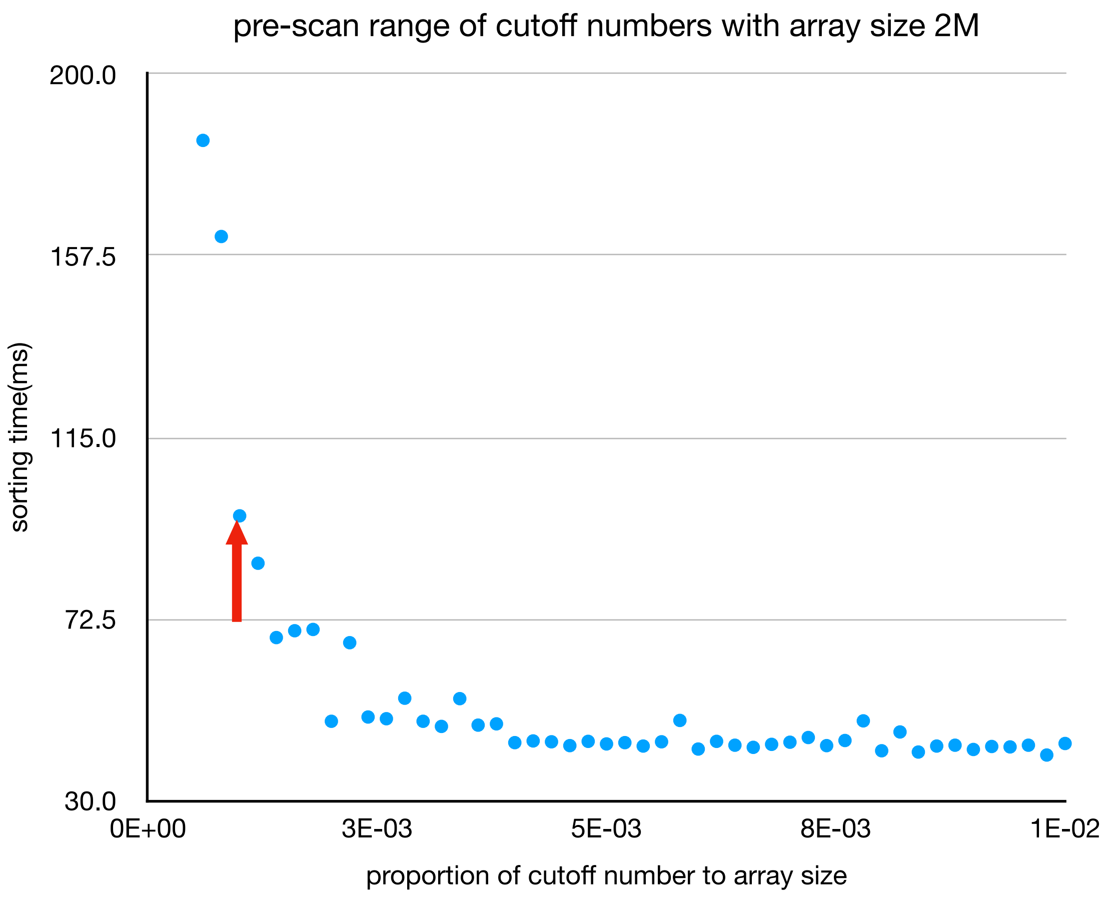
* Experiments setup and design

Two experiments are executed to observe the relationship of cutoff number and thread counts between the sorting time.
The array size will be ranged from 1 million to 8 million.

cutoff number finding:

To find out the best cutoff number, the thread count is fixed, and the independent variable is the array size.

To narrow down the search range of the cutoff number, a pre-scan experiment is executed. The cutoff range starts from 200 to 10000, and the interval is 200.



The result showed that starting from a small cutoff number like 200 will cost too much time, and an interval like 200 is unnecessary; there is no need to get that high resolution. The cutoff number from 1000 will be more suitable(the red arrow). The following experiments will start from 1000.

Thread count finding:

To find out the best thread should be used, the array size is fixed and the independent variable is the thread count, and the cutoff number will be adjusted and measured accordingly.

The thread counts start from 1 to 8, sticked to the power of 2.

*** Results**

The cutoff number determined the ratio of the system sorting time to the total sorting time. The same cutoff number with different array size are plotted in Figure 1.

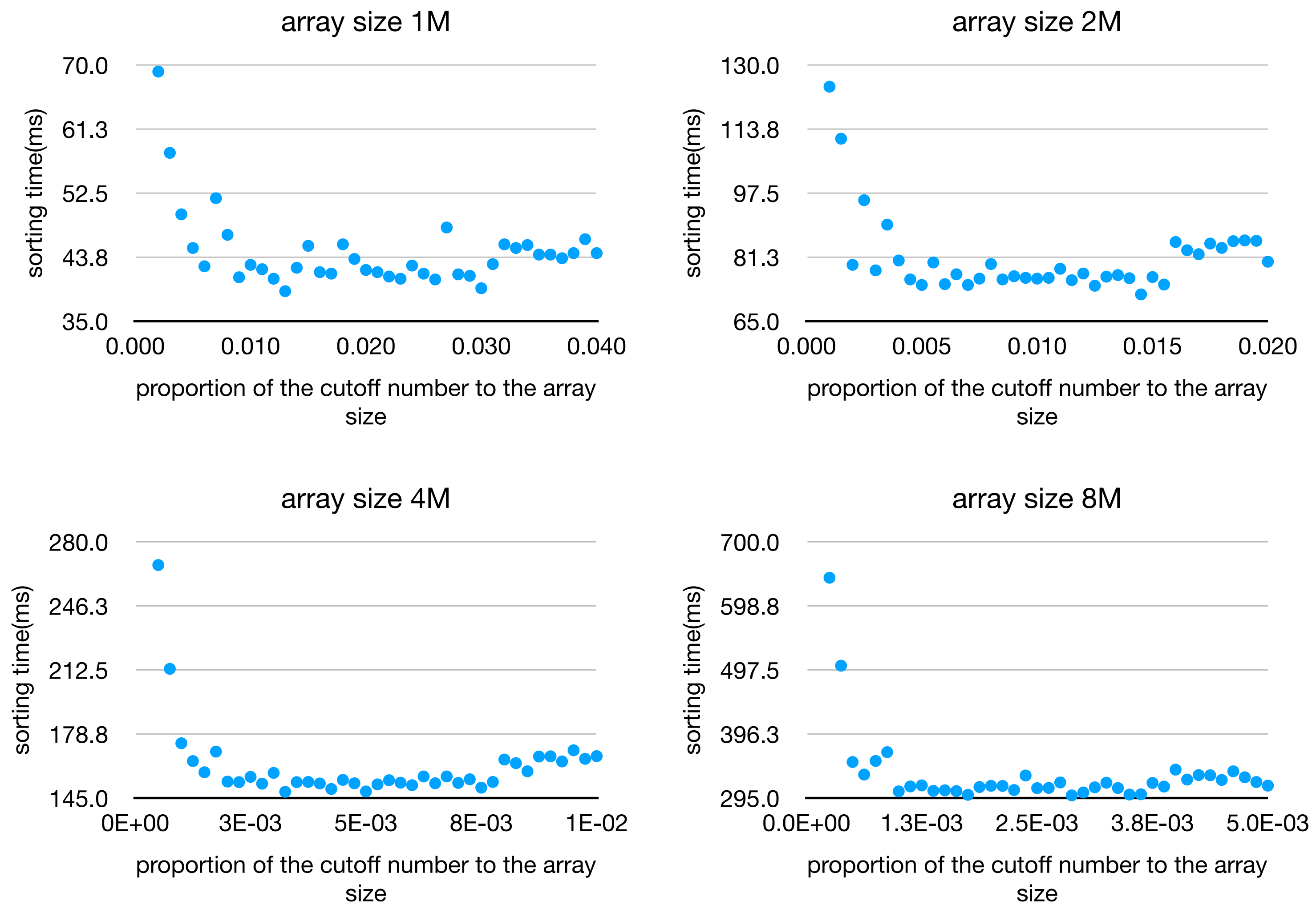


Figure 1 cutoff number finding experiments

The parallelism determined the parallel sorting time. The array sizes are fixed to compare the different parallelism.

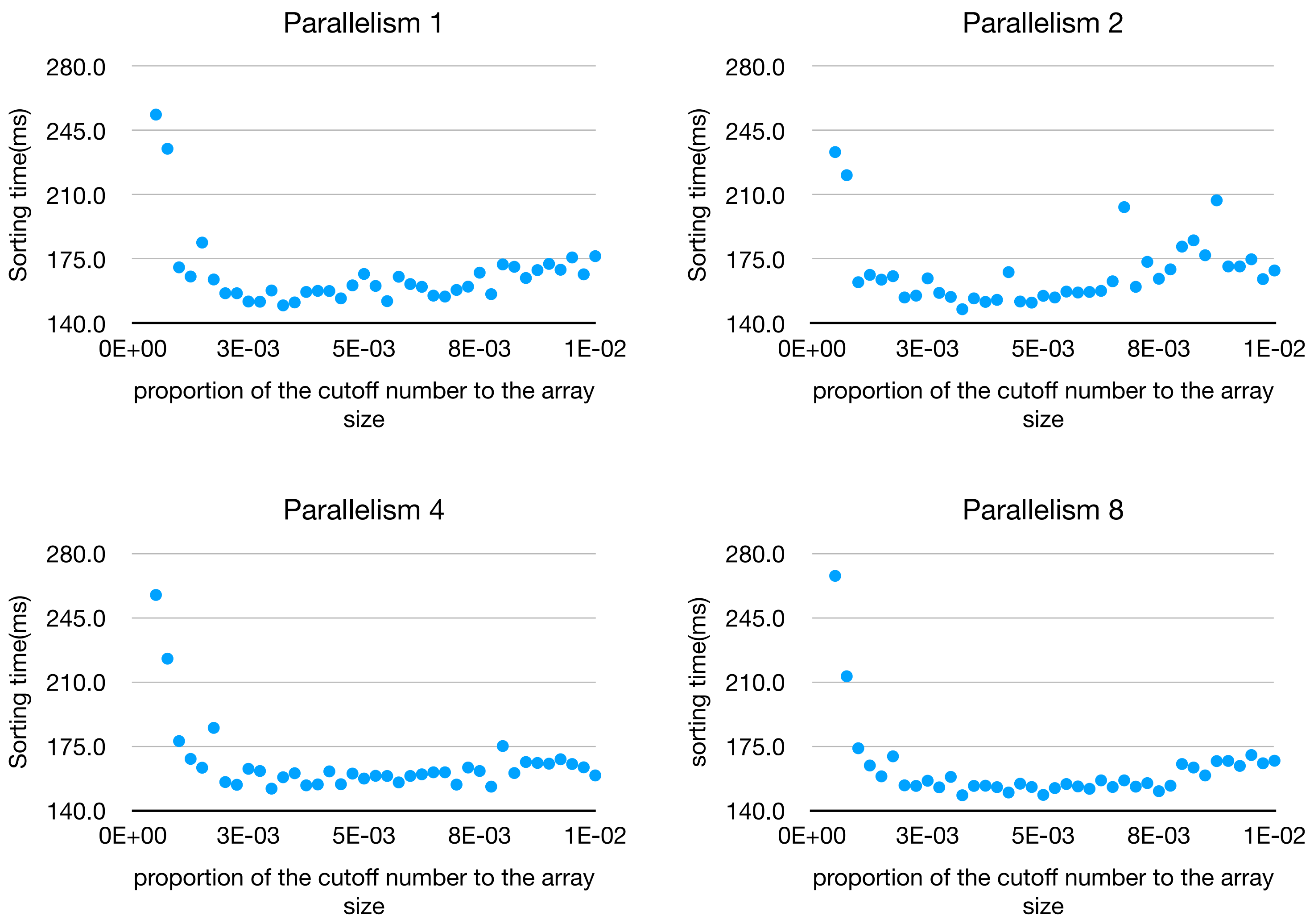


Figure 2 parallelism experiments

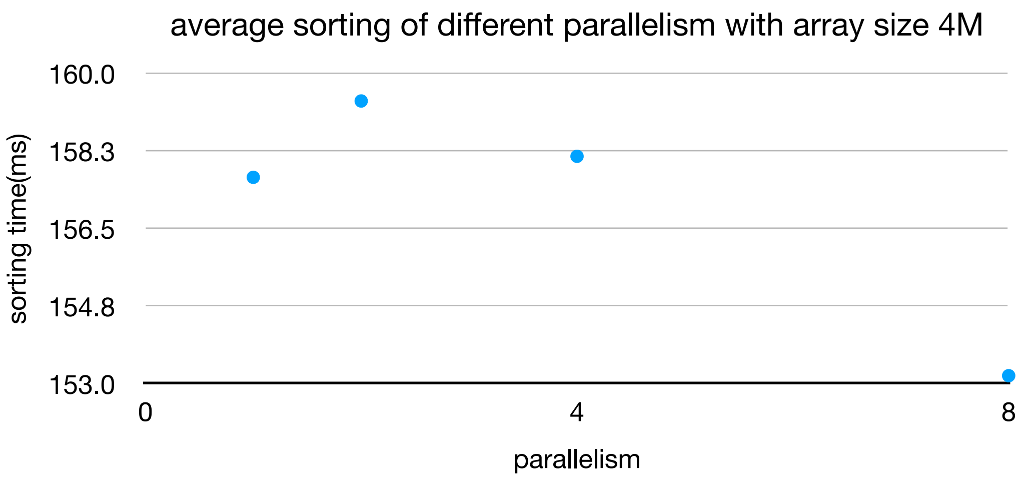


Figure 3

* **Analysis**

Short Summary:
* a cutoff around 8000 is a good choice, increase it has no improvement.
* a cutoff over 30000 increased the sorting time
* increased parallelism should have a greater improvement but the experiment showed that it has few improvement to the sorting time

Total sorting time = ParSort time + SysSort time

* cutoff number experiments:

The cutoff number determined the SysSort time and the ParSort time, and the parallelism affect few for the ParSort time. The total sorting time will be determined by the increase rate and decrease rate of the SysSort and ParSort, respectively. When the cutoff number increased, the SysSort time increased and the ParSort time decreased.

The pre-scan experiments showed that a small cutoff like 200 is much slower than a greater cutoff like 2000, which means that the ParSort time contribute much more than SysSort time.

After calculating the ratio back to the cutoff number, I found that a cutoff number ranging from 8000 to 30000 will not have a great impact on the sorting time. However, if the cutoff number is over 30000 will increase the total sorting time, and the reason is the increasing rate of the SysSort is greater than the decrease rate of the ParSort.

* parallelism experiments:

The parallelism has few improvement to the sorting time(figure 3).
Even on a very large array size like 32 million.

There are many different array size I tried but it seems like have no big improvement to the sorting time. I am not sure if this is because the architecture of my laptop(m1 pro, which has a different cpu architecture) is different to the others.

The result of the parallelism experiments is not match to my expectation.
My expectation of the parallelism experiments is that when the thread count doubles the sorting time should be divided by around 1.5 to 2. (I did some similar experiments before, but it seems like not work on my laptop this time)

When the thread count approached the CPU count, the sorting time will remain almost the same and will not decrease a lot.

The above expectation is based on the environment has enough RAM for the program to use.