INFO 6200 Assignment 4 Parallel sorting
Qixiang Zhou
NUID: 001822974

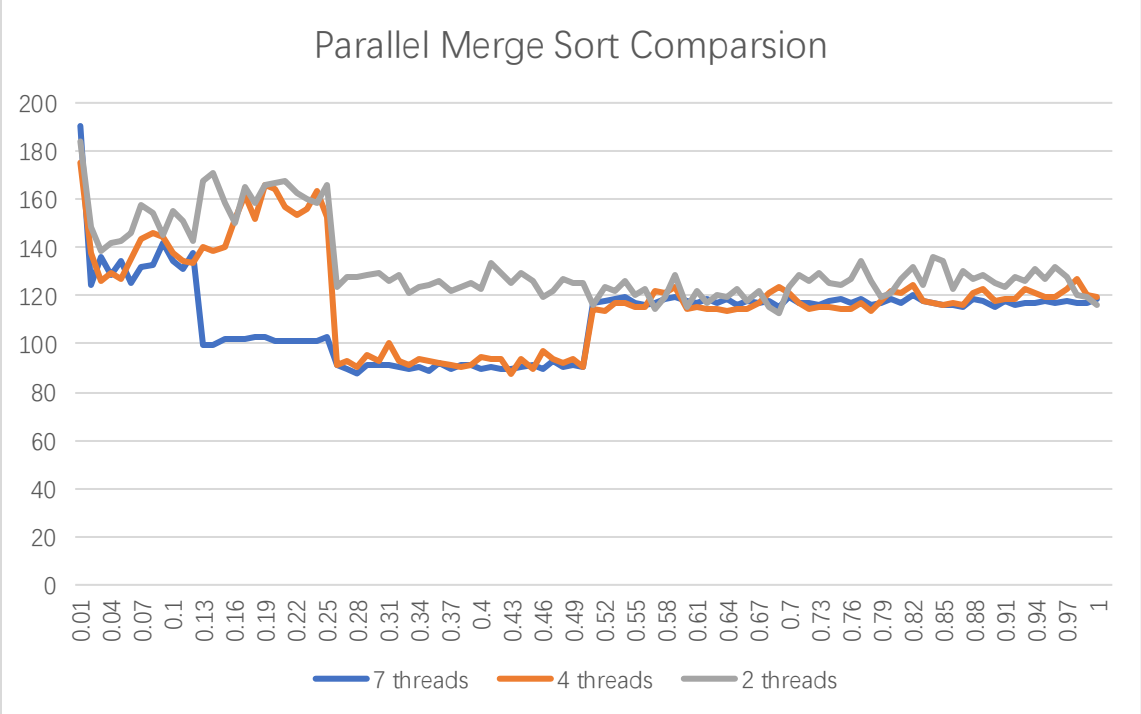
1. Conclusion

   After the experiment, the size of the cutoff should depend on the core, or the number of threads we want to use. For example, if we have an array which length is 1000, and we have 4 threads, the best size of cutoff is 1000/4 = 250.

   Besides, when we have same number of threads, small cutoff size will not present better than the best size of cutoff. And, when we have the same cutoff size, more threads will present almost same as use the proper number of threads, which is length/cutoff.size();

   Therefore, the best performance is: number of threads = Array.length / cutoff.size();

2. Chart



You can see when the cutoff is ¼ to ½ of the array, the run time is the least. And the threads number is equal to array/cutoff.

3. Code

I use this code to set the threads number. In addition, this must did before code compile.

```
System.setProperty("java.util.concurrent.ForkJoinPool.common.parallelism", "2");
```

And when the waiting for sort array's size is smaller than cutoff, I use the system sort method. Else, I will divide it into two parts, and after sort, merge them together.

```java
if (size < cutoff) {
    Arrays.sort(array, from, to);
} else {
    CompletableFuture<int[]> parsort1 = parsort(array, from, to: from + (to - from) / 2);
    CompletableFuture<int[]> parsort2 = parsort(array, from: from + (to - from) / 2, to);
    CompletableFuture<int[]> parsort = parsort1.
            thenCombine(parsort2, (xs1, xs2) -> {
                int[] result = new int[xs1.length + xs2.length];
                // TODO implement me
                int i = 0;
                int j = 0;
                for (int k = 0; k < result.length; k++) {
                    if (i >= xs1.length) {
                        result[k] = xs2[j++];
                    } else if (j >= xs2.length) {
                        result[k] = xs1[i++];
                    } else if (xs2[j] < xs1[i]) {
                        result[k] = xs2[j++];
                    } else {
                        result[k] = xs1[i++];
                    }
                }
                return result;
            });
```