

# PSA Assignment - 5 Parallel Sorting

## 1) Screenshots of Parallel Sorting

The screenshot shows an IDE with a project named 'INFO6205-Spring2023'. The project structure includes a 'sort' package with a 'par' sub-package. The 'Main' class is selected in the 'Run' tab. The code in 'Main.java' is as follows:

```
package edu.neu.coe.info6205.sort.par;

import ...

/**
 * This code has been fleshed out by Ziyao Qiao. Thanks very much.
 * CONSIDER tidy it up a bit.
 */
// Sneha Govindarajan
public class Main {

    // Sneha Govindarajan
    public static void main(String[] args) {
        int size = 1000000;
        processArgs(args);
        int thread = 2;
        System.out.println("Available cores in Machine "+Runtime.getRuntime().availableProcessors());
        while (thread < 128) {
            System.out.println("Degree of parallelism: " + ForkJoinPool.getCommonPoolParallelism());
            ForkJoinPool pool = new ForkJoinPool(thread);
            System.out.println("Degree of parallelism: " + pool.getParallelism());
        }
    }
}
```

The 'Run' output shows the following results:

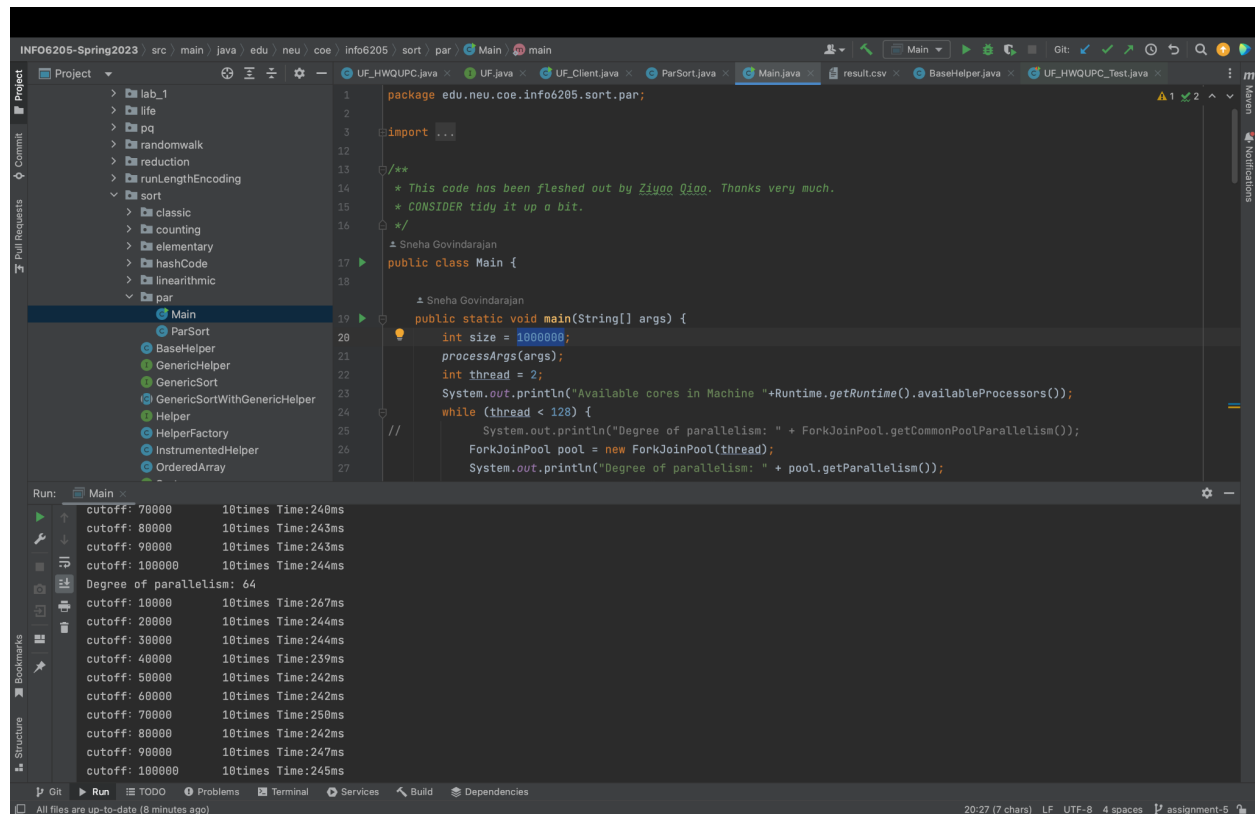
```
Available cores in Machine 8
Degree of parallelism: 2
cutoff: 10000 10times Time:520ms
cutoff: 20000 10times Time:329ms
cutoff: 30000 10times Time:315ms
cutoff: 40000 10times Time:310ms
cutoff: 50000 10times Time:317ms
cutoff: 60000 10times Time:318ms
cutoff: 70000 10times Time:337ms
cutoff: 80000 10times Time:338ms
cutoff: 90000 10times Time:329ms
cutoff: 100000 10times Time:329ms
Degree of parallelism: 4
cutoff: 10000 10times Time:294ms
```

The screenshot shows the same IDE as the first screenshot, but with the 'thread' variable set to 8. The code in 'Main.java' is the same as before. The 'Run' output shows the following results:

```
cutoff: 10000 10times Time:268ms
cutoff: 20000 10times Time:268ms
cutoff: 30000 10times Time:273ms
cutoff: 40000 10times Time:261ms
cutoff: 50000 10times Time:269ms
cutoff: 60000 10times Time:264ms
cutoff: 70000 10times Time:272ms
cutoff: 80000 10times Time:267ms
cutoff: 90000 10times Time:267ms
cutoff: 100000 10times Time:267ms
Degree of parallelism: 8
cutoff: 10000 10times Time:297ms
cutoff: 20000 10times Time:242ms
cutoff: 30000 10times Time:239ms
cutoff: 40000 10times Time:237ms
cutoff: 50000 10times Time:236ms
cutoff: 60000 10times Time:237ms
```

# PSA Assignment - 5

# Parallel Sorting



The screenshot shows an IDE with a project named 'info6205-Spring2023'. The project structure includes a 'par' package with a 'Main' class. The 'Main' class contains a 'main' method that sorts an array of size 1000000 using a parallel sorting algorithm. The code uses a 'ForkJoinPool' to parallelize the sorting process. The output of the program is displayed in the 'Run' console, showing the degree of parallelism and the execution time for different cutoff values.

```
package edu.neu.coe.info6205.sort.par;

import ...

/**
 * This code has been fleshed out by Ziyou Qiao. Thanks very much.
 * CONSIDER tidy it up a bit.
 */
public class Main {

    public static void main(String[] args) {
        int size = 1000000;
        processArgs(args);
        int thread = 2;
        System.out.println("Available cores in Machine " + Runtime.getRuntime().availableProcessors());
        while (thread < 128) {
            System.out.println("Degree of parallelism: " + ForkJoinPool.getCommonPoolParallelism());
            ForkJoinPool pool = new ForkJoinPool(thread);
            System.out.println("Degree of parallelism: " + pool.getParallelism());
        }
    }
}
```

Run: Main

cutoff	Time
70000	10times Time:240ms
80000	10times Time:243ms
90000	10times Time:243ms
100000	10times Time:244ms
Degree of parallelism: 64	
10000	10times Time:267ms
20000	10times Time:244ms
30000	10times Time:244ms
40000	10times Time:239ms
50000	10times Time:242ms
60000	10times Time:242ms
70000	10times Time:258ms
80000	10times Time:242ms
90000	10times Time:247ms
100000	10times Time:245ms

## 2) Relationship Conclusion:

The relationship between the cutoff value and the execution time is not straightforward. Increasing the cutoff value generally results in faster execution times, but there is a point beyond which further increases in the cutoff value provide no performance benefits and may even degrade performance. This threshold is determined by the characteristics of the input data and the machine on which the algorithm is executed.

The relationship between the number of threads and the execution time is generally inverse, i.e., increasing the number of threads leads to faster execution times. However, this relationship also depends on the characteristics of the input data and the machine on which the algorithm is executed. There is a limit to the number of threads that can be used efficiently on a given machine. Beyond this limit, adding more threads may lead to diminishing returns, or even performance degradation due to increased overheads and contention for system resources.

# PSA Assignment - 5

## Parallel Sorting

### 3) Evidence to Support Relationship

Array size: 500000

Cut-off - [10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000]

Thread 2 - [320, 178, 156, 162, 163, 159, 171, 174, 171, 170]

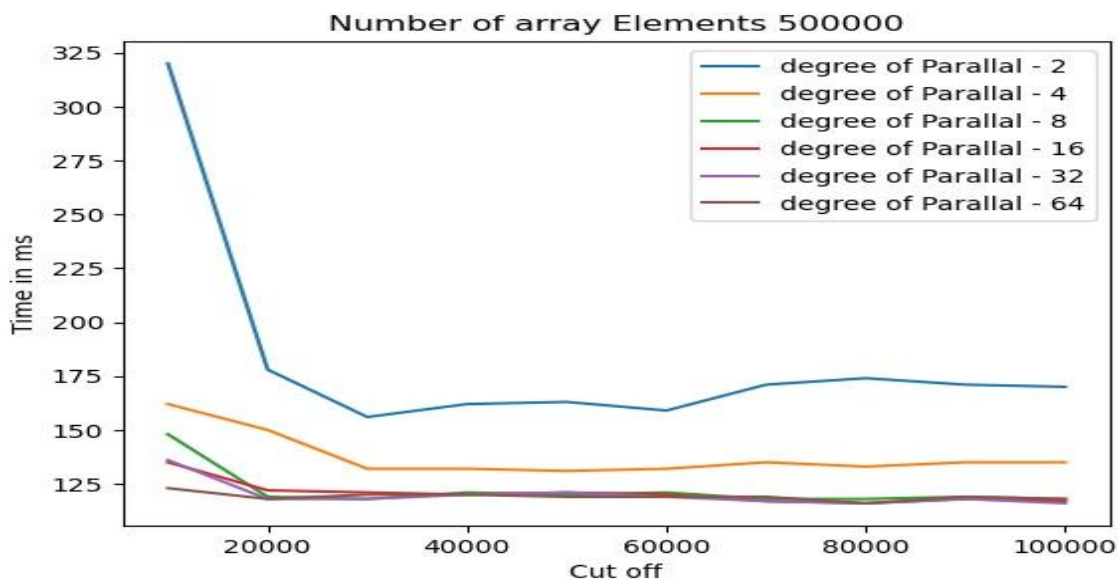
Thread 4 - [162, 150, 132, 132, 131, 132, 135, 133, 135, 135]

Thread 8 - [148, 119, 118, 121, 120, 121, 118, 118, 119, 118]

Thread 16 - [135, 122, 121, 120, 121, 120, 117, 116, 118, 117]

Thread 32 - [136, 118, 118, 120, 121, 119, 117, 116, 118, 116]

Thread 64 - [123, 118, 120, 120, 119, 119, 119, 116, 119, 118]



Array size: 600000

Cut off - [10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000]

Thread 2 - [351, 200, 198, 196, 199, 200, 209, 214, 210, 200]

Thread 4 - [203, 163, 156, 156, 158, 158, 157, 159, 159, 160]

Thread 8 - [187, 146, 148, 146, 146, 142, 145, 141, 143, 141]

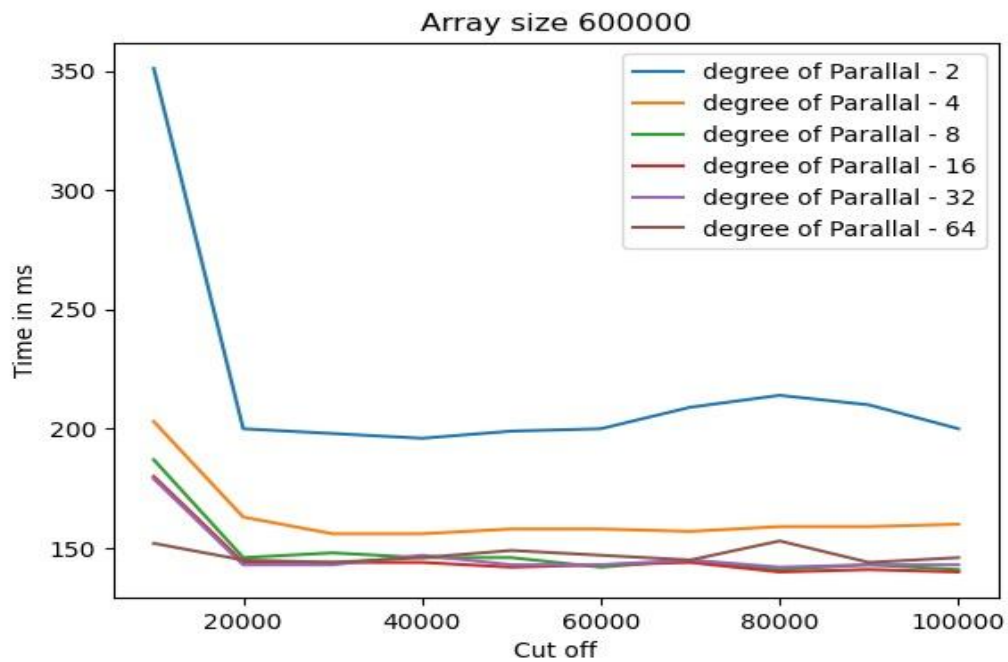
Thread 16 - [180, 144, 144, 144, 142, 143, 144, 140, 141, 140]

Thread 32 - [179, 143, 143, 147, 143, 143, 145, 142, 143, 143]

Thread 64 - [152, 145, 144, 146, 149, 147, 145, 153, 144, 146]

## PSA Assignment - 5

## Parallel Sorting



Array size - 1000000

Cut off - [10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000]

Thread 2 - [500, 317, 308, 315, 322, 321, 331, 339, 335, 342]

Thread 4 - [308, 254, 261, 256, 261, 259, 267, 266, 270, 266]

Thread 8 - [297, 238, 241, 236, 236, 236, 242, 242, 242, 241]

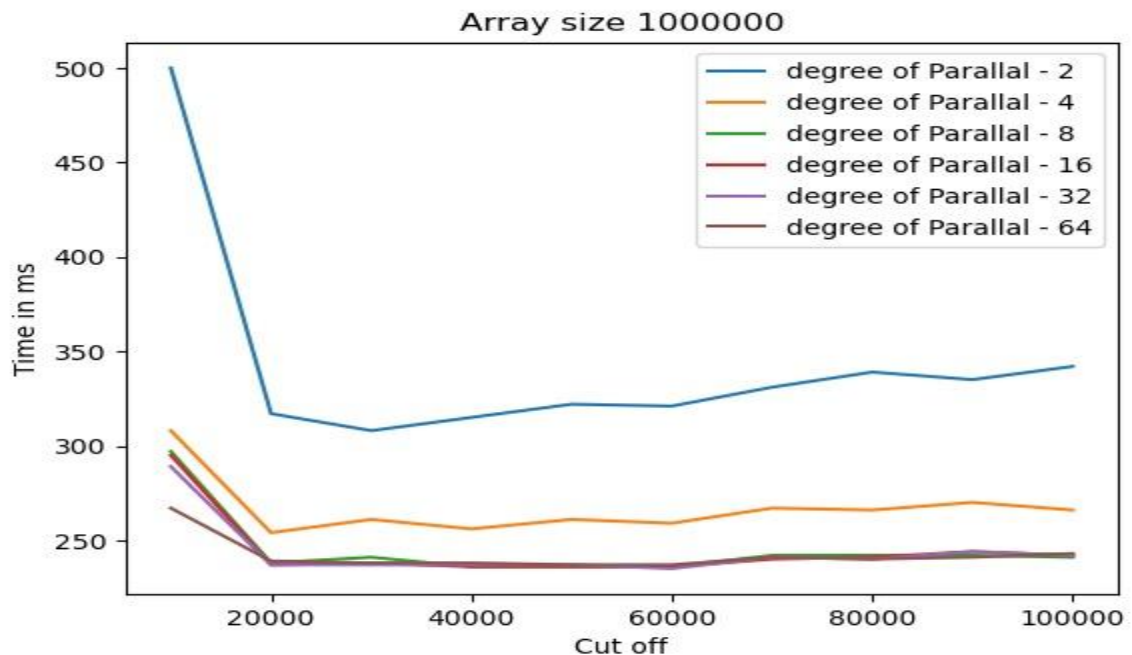
Thread 16 - [295, 237, 238, 236, 236, 236, 240, 241, 244, 242]

Thread 32 - [289, 237, 237, 237, 237, 235, 241, 240, 244, 242]

Thread 64 - [267, 239, 238, 238, 237, 237, 241, 240, 241, 243]

## PSA Assignment - 5

## Parallel Sorting



### Conclusion

- After varying the cutoff value and the number of threads for various array sizes, the number of threads greater than four does not improve performance. Keeping four threads is thus the best option.
- According to the graph, the lowest performance time is achieved for a cutoff value of 25% of the array size.
- Thus, with a cutoff value of 25% and a thread count of 4, optimal performance can be observed.