Program Structures & Algorithms

Spring 2022

Assignment 4 (Parallel Sorting)

Name: Aditya Kinare

(NUID): 001095881

- **Task**
- **Output screenshot**
- **Relationship Conclusion**
- **Evidence / Graph**
- **Unit tests result**

Task:

- Understood and Implemented the ForkJoinPool to manipulate the number of threads
- Modified the Main.java file to initialize different array size, cutoff, and thread values
- Plotted the excel graph for Time taken to perform sort vs Number of Threads for different array sizes and cutoff values

Output screenshot:

Code Implementation for main.java to take input from user for cutoff and threadcount

```java
public class Main {

    public static void main(String[] args) {
        int thread = 2, size = 50000;
        processArgs(args);
        System.out.println("Size of Array: "+size);
        while(thread < 128) {
            ForkJoinPool pool = new ForkJoinPool(thread);
            System.out.println("Degree of parallelism: " + pool.getParallelism());
            Random random = new Random();
            int[] array = new int[size];
            ArrayList<Long> timeList = new ArrayList<>();
            for (int j = 0; j < 10; j++) {
                ParSort.cutoff = 5000 * (j+1);
                long time;
                long startTime = System.currentTimeMillis();
                for (int t = 0; t < 10; t++) {
                    for (int i = 0; i < array.length; i++) array[i] = random.nextInt(10000000);
                    ParSort.sort(array, 0, array.length, pool);
                }
                long endTime = System.currentTimeMillis();
                time = (endTime - startTime);
                timeList.add(time);

                System.out.println("cutoff:" + (ParSort.cutoff) + "\t\t10times Time:" + time + "ms");

            }
```

Code Implementation for ParSort.java

```java
class ParSort {

    public static int cutoff = 1000;

    public static void sort(int[] array, int from, int to, ForkJoinPool parallelPool) {
        if (to - from < cutoff) Arrays.sort(array, from, to);
        else {
            // FIXME next few lines should be removed from public repo.
            CompletableFuture<int[]> parsort1 = parsort(array, from, from + (to - from) / 2, parallelPool); // TO IMPLEMENT
            CompletableFuture<int[]> parsort2 = parsort(array, from + (to - from) / 2, to, parallelPool); // TO IMPLEMENT
            CompletableFuture<int[]> parsort = parsort1.thenCombine(parsort2, (xs1, xs2) -> {
                int[] result = new int[xs1.length + xs2.length];
                // TO IMPLEMENT
                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;
            });

            parsort.whenComplete((result, throwable) -> System.arraycopy(result, 0, array, from, result.length));
//            System.out.println("# threads: "+ ForkJoinPool.commonPool().getRunningThreadCount());
            parsort.join();
        }
    }
```

```
Size of Array: 50000
Degree of parallelism: 2
cutoff:5000             10times Time:94ms
cutoff:10000            10times Time:47ms
cutoff:15000            10times Time:23ms
cutoff:20000            10times Time:23ms
cutoff:25000            10times Time:33ms
cutoff:30000            10times Time:26ms
cutoff:35000            10times Time:21ms
cutoff:40000            10times Time:22ms
cutoff:45000            10times Time:47ms
cutoff:50000            10times Time:22ms
Degree of parallelism: 4
cutoff:5000             10times Time:28ms
cutoff:10000            10times Time:21ms
cutoff:15000            10times Time:16ms
cutoff:20000            10times Time:18ms
cutoff:25000            10times Time:18ms
cutoff:30000            10times Time:21ms
cutoff:35000            10times Time:20ms
cutoff:40000            10times Time:33ms
cutoff:45000            10times Time:23ms
cutoff:50000            10times Time:22ms
Degree of parallelism: 8
cutoff:5000             10times Time:29ms
cutoff:10000            10times Time:20ms
cutoff:15000            10times Time:20ms
cutoff:20000            10times Time:18ms
cutoff:25000            10times Time:20ms
cutoff:30000            10times Time:23ms
cutoff:35000            10times Time:22ms
cutoff:40000            10times Time:23ms
cutoff:45000            10times Time:20ms
cutoff:50000            10times Time:21ms
```

```
Degree of parallelism: 16
cutoff:5000             10times Time:21ms
cutoff:10000            10times Time:19ms
cutoff:15000            10times Time:19ms
cutoff:20000            10times Time:18ms
cutoff:25000            10times Time:18ms
cutoff:30000            10times Time:21ms
cutoff:35000            10times Time:20ms
cutoff:40000            10times Time:22ms
cutoff:45000            10times Time:25ms
cutoff:50000            10times Time:24ms
Degree of parallelism: 32
cutoff:5000             10times Time:17ms
cutoff:10000            10times Time:14ms
cutoff:15000            10times Time:14ms
cutoff:20000            10times Time:16ms
cutoff:25000            10times Time:16ms
cutoff:30000            10times Time:20ms
cutoff:35000            10times Time:21ms
cutoff:40000            10times Time:21ms
cutoff:45000            10times Time:21ms
cutoff:50000            10times Time:21ms
Degree of parallelism: 64
cutoff:5000             10times Time:17ms
cutoff:10000            10times Time:13ms
cutoff:15000            10times Time:14ms
cutoff:20000            10times Time:16ms
cutoff:25000            10times Time:15ms
cutoff:30000            10times Time:18ms
cutoff:35000            10times Time:19ms
cutoff:40000            10times Time:19ms
cutoff:45000            10times Time:19ms
cutoff:50000            10times Time:18ms
```

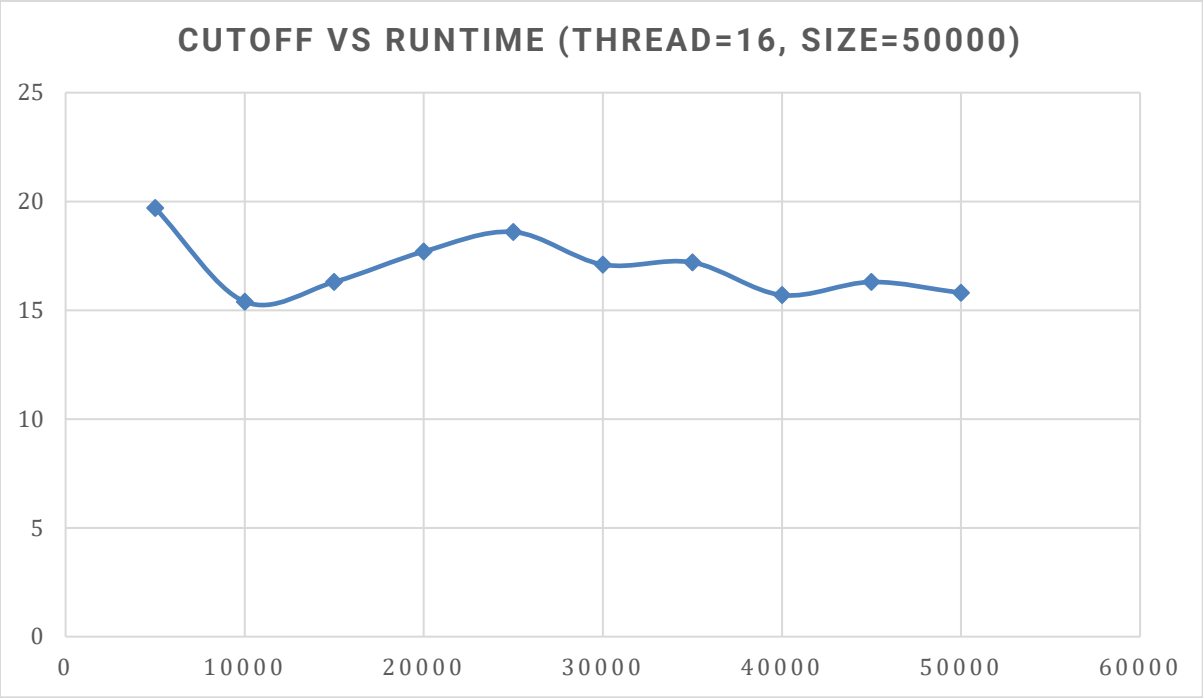
Relationship Conclusion:

After running experiments with different cutoff values and the number of threads for different array sizes, we can conclude that four threads are the optimal choice as the algorithm's performance does not increase significantly beyond four.

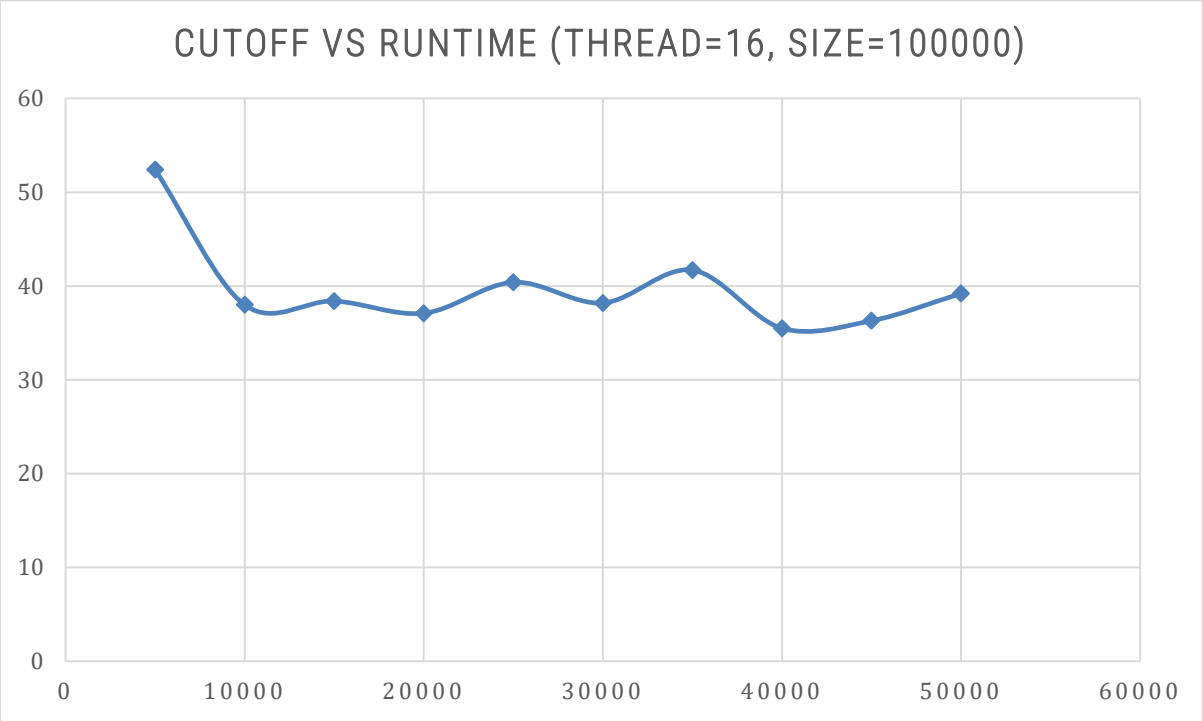
The lowest performance is achieved when the cutoff value is 25% of the array size.

Evidence/Graph:

For Array Size = 50000, Thread = 16



For Array Size = 100000, Thread = 16



Unit Test Results:

```
Size of Array: 50000
Degree of parallelism: 2
cutoff:5000             10times Time:94ms
cutoff:10000            10times Time:47ms
cutoff:15000            10times Time:23ms
cutoff:20000            10times Time:23ms
cutoff:25000            10times Time:33ms
cutoff:30000            10times Time:26ms
cutoff:35000            10times Time:21ms
cutoff:40000            10times Time:22ms
cutoff:45000            10times Time:47ms
cutoff:50000            10times Time:22ms
Degree of parallelism: 4
cutoff:5000             10times Time:28ms
cutoff:10000            10times Time:21ms
cutoff:15000            10times Time:16ms
cutoff:20000            10times Time:18ms
cutoff:25000            10times Time:18ms
cutoff:30000            10times Time:21ms
cutoff:35000            10times Time:20ms
cutoff:40000            10times Time:33ms
cutoff:45000            10times Time:23ms
cutoff:50000            10times Time:22ms
Degree of parallelism: 8
cutoff:5000             10times Time:29ms
cutoff:10000            10times Time:20ms
cutoff:15000            10times Time:20ms
cutoff:20000            10times Time:18ms
cutoff:25000            10times Time:20ms
cutoff:30000            10times Time:23ms
cutoff:35000            10times Time:22ms
cutoff:40000            10times Time:23ms
cutoff:45000            10times Time:20ms
cutoff:50000            10times Time:21ms
```

```
Degree of parallelism: 16
cutoff:5000              10times Time:21ms
cutoff:10000             10times Time:19ms
cutoff:15000             10times Time:19ms
cutoff:20000             10times Time:18ms
cutoff:25000             10times Time:18ms
cutoff:30000             10times Time:21ms
cutoff:35000             10times Time:20ms
cutoff:40000             10times Time:22ms
cutoff:45000             10times Time:25ms
cutoff:50000             10times Time:24ms
Degree of parallelism: 32
cutoff:5000              10times Time:17ms
cutoff:10000             10times Time:14ms
cutoff:15000             10times Time:14ms
cutoff:20000             10times Time:16ms
cutoff:25000             10times Time:16ms
cutoff:30000             10times Time:20ms
cutoff:35000             10times Time:21ms
cutoff:40000             10times Time:21ms
cutoff:45000             10times Time:21ms
cutoff:50000             10times Time:21ms
Degree of parallelism: 64
cutoff:5000              10times Time:17ms
cutoff:10000             10times Time:13ms
cutoff:15000             10times Time:14ms
cutoff:20000             10times Time:16ms
cutoff:25000             10times Time:15ms
cutoff:30000             10times Time:18ms
cutoff:35000             10times Time:19ms
cutoff:40000             10times Time:19ms
cutoff:45000             10times Time:19ms
cutoff:50000             10times Time:18ms
```