

Assignment 5

Parallel sorting

Assignment 5: Implement a parallel sorting method

- Your assignment is to modify *mergeSort* so that each partition is sorted in parallel.
- In order to do that, you will have to learn about programming asynchronous function calls in Java8.
- So, you will need to create two classes: *Main.java* and *ParSort.java*.
- You will run experiments using different parallelizing strategies: involving cutoff and number of threads. Please see the instructions on Blackboard for more detail on this.
- I will give you a sample *Main.java*:

Assignment 5: Main.java (sample)

- **package** edu.neu.coe.info6205;

import java.util.Random;

public class Main {

public static void main(String[] args) {

if (args.length>0) ParSort.cutoff = Integer.parseInt(args[0]);

 Random random = **new** Random(0L);

int[] array = **new int**[2000];

for (**int** i = 0; i < array.length; i++) array[i] = random.nextInt(10000);

 ParSort.sort(array, 0, array.length);

if (array[0]==11) System.out.println("Success!");

 }

}

Assignment 5: *CompletableFuture*

- Creating a *CompletableFuture* that returns a result (template):

```
CompletableFuture.supplyAsync(  
    () -> {  
        int size = ...  
        int[] result = new int[size];  
        // get the values for result  
        return result;  
    }  
);
```

Assignment 5: composing *CompletableFutures*

- Composing two *CompletableFutures* (cf1 and cf2) into one (template):

```
cf1.thenCombine(cf2, (xs1, xs2) -> {  
    int[] result = new int[xs1.length + xs2.length];  
    // set up result  
    return result;  
});
```

Assignment 5: waiting for *CompletableFutures*

- Waiting for a *CompletableFuture* that returns a result (template):

```
cf.whenComplete((result, throwable) -> // do something with result  
cf.join());
```

Assignment 5: Sample *ParSort.java* (template)

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

import java.util.concurrent.CompletableFuture;

class ParSort {
    public static int cutoff = 1000;
    public static void sort(int[] array, int from, int to) {
        int size = from - to + 1;
        if (size < cutoff) Arrays.sort(array, from, to);
        else {
            CompletableFuture<int[]> cf1 = ...
            CompletableFuture<int[]> cf2 = ...
            CompletableFuture<int[]> cf = cf1.
                thenCombine(cf2, (xs1, xs2) -> {...});
            cf.whenComplete((result, throwable) -> ...);
        }
    }
}
```