Rose Whitt

COMP 322 HW 3: Report

Part A.

A summary of your parallel algorithm. You should write a few sentences describing the approach and the algorithm.

My parallel algorithm implements a recursive helper algorithm that utilizes only finish and async constructs to achieve parallelism. The helper is in the same format as the sequential recursive helper but wraps the recursive calls in a finish, with each call in its own async construct.

My thought process behind this was examining the given sequential algorithm and identifying the tasks that could be performed in parallel.

Part B.

An explanation as to why you believe that your implementation is correct and data-race-free.

My implementation is correct and data race free because it always outputs the same expected values and it does not read and write in the same location. More specifically, my algorithm is data race free because it accessed different parts of the array due to the indexed variables (inJ, inI, inN, inM) always being different ranges.

Part C.

An explanation of what values of WORK and CPL (as formulae of n) you expect to see from your algorithm.

I expect the total WORK to be the same as the sequential algorithm but the CPL to be less than the sequential algorithm's CPL. The total WORK will be the best case time complexity of quick sort, nlogn.

WORK = **nlogn**

CPL = O(n) + O(n/2) + O(n/4) + . . . = **O(n)**

Rose Whitt

Testing

The report file should also include test output for sorting an array of size n = 1024 (i.e. the unit test name testRandomDataInput1K). The test output should include the WORK, CPL, and IDEAL PARALLELISM (= WORK/CPL) values.

```
Input array length = 1024
 QuickSort Sequential metrics: WORK = 10335, CPL = 10335
 QuickSort Parallel metrics: WORK = 10335, CPL = 3554, Ideal Parallelism = 2.90799099606
 Speed-Up = 2.9079909960607764
Expected Speedup: 2.7
Actual Speedup: 2.9079909960607764
```