Parallel programming II

J. Guo, R. Rajagopalan, N. Semmler, C. Tsakiroglou

May 15, 2013

Exercise 1 – Sequential Overspecification

(a) Analysis

The algorithm finds the single highest sum of any subsequence of the given sequence. E.g. if the given sequence would be $(1, 2, -3, 5, -10, 6, -2, 8, -5, -6, 0)$, positive sums of subsequences would be $[3, 5, 12]$. Hence the program would return the sum 12.

(b) Parallel Algorithm

(b) Pseudo Code:

Listing 1: pseudo code

```
seq = (1, 2, -3, 5, -10, 6, -2, 8, -5, -6, 0)
max = length(seq)
number_parallel = 0

int find_max_sum(seq, max) {
    // starts accumulating max sum whenever a positive number follows a negative
        one
    result = [] // should be safe for parallel write
    for (i = 1; i < max; i++) {
        if (seq[i] > 0 && (i == 0 || seq[i-1] <= 0)) {
            spawn_parallel(cumulative_sum(seq, i, max, result)));
            i++; // can ignore one additional number
        }
    return max(result)
}

void cumulative_sum(seq, index, max, result) {
    maxsum = 0
    sum = 0

    for ( ; i < max; i++) {
        sum += seq[i]
        if (sum < 0) { result += maxsum; return }
        if (sum > maxsum) maxsum = sum
    }
    result += maxsum
}
```

Exercise 2 – Conway's Game of Life

(a) Problem decomposition

We compare variant A1 to B2

The advantage of variant A1 (our choice for implementation) is that all processes run in a completely decentralized matter. They communicate only the most important information (that is border regions) and require no knowledge about what the rest of the nodes are doing. After an initial division of the playing field during initialization no central coordinator is required. Even presenting their results can be done in decentralized manner if an output channel is shared by all. (Of course you need some kind of synchronization)

What variant B2 looses when it comes to speed it wins in flexibility. This variant enables a much more flexible handling of space. Additional fields can be added and existing parts of the field can be deleted at runtime. Idle processors can be added to the program and removed when other higher priority jobs appear.

In a constant environment A1 is to be preferred in a dynamic environment B2 should win your favor.

(b) Implementation

Please take a look into the folder *GameOfLife*.