Lab 7.2 - Parallelizing techniques

December 11, 2018

1 Goal

The goal of this lab is to implement a simple but non-trivial parallel algorithm.

2 Requirement

Add n big numbers. We want the result to be obtained digit by digit, starting with the least significant one, and as soon as possible. For this reason, you should use n-1 threads, each adding two numbers. Each thread should pass the result to the next thread. Arrange the threads in a binary tree. Each thread should pass the sum to the next thread through a queue, digit by digit.

3 Computer Specification

- *CPU:* Intel Core i7-7500U, 2.90GHz
- *RAM:* 8 GB
- *System type:* 64-bit

4 Short Description of the Implementation

Algorithms used in order to check the performance:

- *regular linear algorithm*
- *binary tree algorithm*

4.1 Regular linear algorithm

Complexity: O(n) time
Iterate through the input list, and add the current number to the sum.

4.2 Binary tree algorithm

Complexity: O(n) time

Create N - 1 threads that add 2 numbers, and arrange them in a binary tree, communicating with each other with queues. (Parallelization - Used the Thread class along with the ArrayBlockingQueue.)

5 Performance Tests

note: by level 'x' i am referring that N = x * 10 input numbers, each one having x digits

Algorithm	Level 1	Level 10	Level 15
regular linear algorithm	0ms	0ms	0ms
binary tree algorithm parallel	13ms	53ms	56ms

Throughout the tests I've put those algorithm through, the results were inconclusive. The binary tree implementation might be optimized to yield better results in the parallel form than in the sequential form, however, in this instance I was not able to get a better performance out of it. The parallel version is on average worse than the sequential version.

6 Conclusion

Implementing such a complex algorithm for such a trivial problem might not be in our favour.