Trying to Parallelize after Algorithm 2

Prasanth R

June 8, 2018

We need to define a class which the alorightm 1 and 2 should work on.
Let name of the class be Node and contains the following memeber variables:

- D

- Number of yeilds the stack has

- Number of takes the stack has

- l

- right

Algorithm 1: ALGORITHM 1 FROM THE PAPER

Algorithm 2: ALGORITHM 2 FROM THE PAPER

1 Splits the input in to k arbitary sets and send it to Algorithm 1 which returns a Node class.

Algorithm 3: CLASS Node

1 deque type $check\_my\_deque$
2 int $number\_takes \leftarrow 0$
3 int $number\_yields \leftarrow 0$
4 int $left$
5 int $right$

Assuming that after algorithm 2 we have a list of Nodes (each Node is an instance of its respective stack).

Algorithm 4 will choose the Nodes which can be reduced and write them to the respective positions of the initial.

Algorithm 4: ASSIGN WORKERS assign workers to nodes and repeat util there is just a single Node

Input: A finite set $list = \{Node_1, Node_2, \ldots, Node_n\}$ of Node pointers

Output: A Node with reduced deque (theoritically stack)

1 $present = size \ of \ list$

2 $last\_yields = present$

3 **while** ***not single Node in*** $list$ **do**

4   **while** $present > 0$ **do**

5   **if** $Node[present].number\_takes < Node[present].number\_yields$ **then**

6 $last\_yields = present$

7   **else if**

          $Node[present].number\_takes >= Node[present].number\_yields$

          $\& \ last\_yields \ != present$ **then**

8 $Node[present] \leftarrow Assign workers which mergepresent-$

             $last\_yields wihch return a Node pointer$

9 $last\_yields \leftarrow present$

10 $present \leftarrow Node[present].left - 1$

11   **wait until** **workers are done their tasks**

12 **return** $list$