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:

- Deque
- Number of yeilds the stack has
- Number of takes the stack has
- left
- 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 char my_deque
2 int number_takes ← 0
3 int number_yields ← 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 [illegible].

Algorithm 4: Assign Workers assign workers t[illegible]les and repeat util there is just a single Node

```
Input: A finite set $lis$[illegible]$Node_1, Node_2, \ldots, Node_n\}$ of Node pointers
Output: A Node with reduced deque (theoritically stack)
[illegible]$sent = s$[illegible]$flist$
$last\_yields = present$
while not single Node in $list$ do
    while $present > 0$ do
        if $Node[present].number\_takes < Node[present].number\_yields$ then
            $last\_yields =$[illegible]$t$
        else if $Node[present].number\_takes >= Node[present].number\_yields$ & [illegible]$st\_yields$ $!$[illegible] then
            $Node[present] \leftarrow Assignworkerswhich$[illegible]$ergepresent$-$last\_yieldswihchreturnaNodepointer$
            $last\_yields \leftarrow$ [illegible]
        $present \leftarrow Node[present].left - 1$
    wait unti[illegible] workers are done their tasks
return $list$
```