Homework 33

Brian Knotten, Brett Schreiber, Brian Falkenstein

November 13, 2018

21

Consider a sequential algorithm for this problem on a binary tree T as follows:

1. Let $i \leftarrow 1$.
2. Perform an Eulerian tour on the tree.
3. If the node is a leaf, label the node with i and increment i.

The result is a tree with each leaf node labelled in-order.
The outline for the parallel version of this algorithm is as follows:

1. Perform an Eulerian tour on T to return a $3n$ linked-list L.
2. Let the nodes in L representing a leaves in the tree (only one visit on the tour) be set to 1 and let their corresponding processor keep a pointer to its node.
3. Let all other nodes in L be set to 0.
4. Perform the linked-list parallel prefix algorithm on L.
5. Each processor corresponding to a leaf node in T looks at its pointer to the node in L to see a prefix sum. This number represents the number of the leaf in T in an in-order traversal.
6. Have each processor label their leaf node in T with this number.

This algorithm takes $O(\log(n))$ time, since performing an Eulerian tour takes a constant number of steps with n processors, and performing the parallel prefix sum algorithm takes $O(\log(n))$ steps. This algorithm is EREW, because the Eulerian tour algorithm is EREW, the parallel prefix algorithm is EREW, and each processor reads and writes exclusively to its corresponding nodes in the steps particular to this algorithm.

Formally, the algorithm is as follows:

Algorithm 1 EREW $O(\log(n))$ algorithm for In-Order labeling of leaf nodes with n processors.

Require: An n-sized binary tree T.

$t_i \leftarrow$ the node in T which p_i points to.
$t_p \leftarrow t_i$'s parent node.
if t_i is a leaf node **then**
    Allocate a copy of t_i for creating L: l_{i_1}
    if t_i is a left child **then**
        Link $l_{i_1} \rightarrow l_{p_2}$
    else if t_i is a right child **then**
        Link $l_{i_1} \rightarrow l_{p_3}$
    end if
else
    $t_l \leftarrow t_i$'s left child.
    $t_r \leftarrow t_i$'s right child.
    Allocate three copies of t_i for creating L: l_{i_1}, l_{i_2}, l_{i_3}.
    Link $l_{i_1} \rightarrow l_{l_1}$
    Link $l_{i_2} \rightarrow l_{r_1}$
    if t_i is a left child **then**
        Link $l_{i_3} \rightarrow l_{p_2}$
    else if t_i is a right child **then**
        Link $l_{i_3} \rightarrow l_{p_3}$
    end if
end if