EDA387 - Lab 4.3: Self-stabilizing α-maximal-partitioning

Jinsong LI, Nathan HAUDOT

October 5, 2021

1 The problem

We have a tree of nodes more or less complex, all nodes are defined as semi-anonymous, the root node runs different program from the others. We want to split our tree into several partitions. The objective of this problem is to design an algorithm to split the tree in order to have partitions that does not exceed α nodes, and the neighboring partitions can not merge into a new valid partition.

2 Algorithm steps

Each edge has a shared register contains following variables, all of them are readable and writable for both nodes:

- bool $r(i,j).inParentsPartition$: indicates whether the node is in the same partition as its parent.
- int $r(i,j).partitionSize$: indicates the size of the partition excluding their higher-level nodes.

We assume that at the beginning, the network tree is already constructed, every node already recognized its parent and children, and the root node is the distinguished processor that runs a different program:

- The node goes through the list of its children nodes, the largest children tree size first (line 3 in the algorithm for the root node, line 5 in the algorithm for the other nodes). We want the largest tree first to directly search if it has a probable α partition or not, the algorithm will then be more efficient in its execution and we can avoid problems that would lead to not discovering other α partitions
- For each child, the node check their partition size
- If joining the child's partition won't make the partition size exceed α, then it will join, and update its partitionSize value.
- After processing a child node, the algorithm-running-node sets the $inParentsPartition$ value to true to the child to indicate that they are in the same partition or not (if the above condition is met).
- At the end of the loop, the node computes his new partition size.

Now, each node will execute the loop program forever, but the register values of all the nodes will not change after they are stabilized.

3 Algorithm

Proposed algorithm for the root node:

1: **while** $True$ **do**
2: $\quad partitionSize = 1$
3: $\quad$ **for each** $child \in node.children.sort(reverse)$ **do**
4: $\quad\quad lr_{child,i} =$ read $(r_{child,i})$

```
        if (partitionSize + lr_{child,i}.partitionSize) ≤ α then
            write r_{i,child}.inParentsPartition = true
            partitionSize = partitionSize + lr_{child,i}.partitionSize
        else
            write r_{i,child}.inParentsPartition = false
        end if
    end for
end while
```

Proposed algorithm for the other the nodes:

```
while True do
    lr_{parent,i} = read (r_{parent,i})
    inParentsPartition = lr_{parent,i}.inParentsPartition
    partitionSize = 1
    for each child ∈ node.children.sort(reverse) do
        lr_{child,i} = read (r_{child,i})
        if (partitionSize + lr_{child,i}.partitionSize) ≤ α then
            writer_{i,child}.inParentsPartition = true
            partitionSize = partitionSize + lr_{child,i}.partitionSize
        else
            write r_{i,child}.inParentsPartition = false
        end if
    end for
    write r_{i,parent}.partitionSize = partitionSize
end while
```

4 Example case

Here is an example with $\alpha = 4$.

On the first figure, we can see that leaf nodes consider their parents as in "their partition". Their parents have the *partitionSize* register value equal to 3, and the α value is not reached.

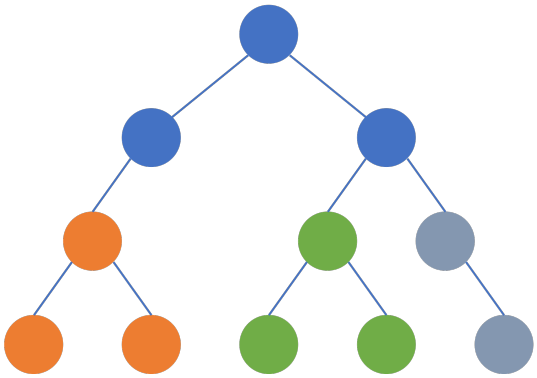

Figure 1: Leaf nodes are in the same partition as their parents

On the second figure, we can see that the parent nodes of the orange and green partitions have included themselves in it. Here, the *partitionSize* register value of the new parents of the orange and green partitions have reached α, these partitions cannot expand anymore, so the states of the processors in these partitions should not change in the future.

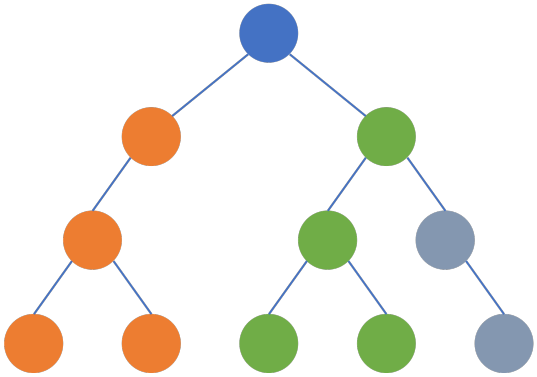

Figure 2: Partition parents have been updated, and α value has been reached for them.

Now both children of the root are in full partitions so the root cannot join any partitions, α-maximal partitioning is constructed, the nodes are stabilized.