373

eceye.yuan

October 2015

1 Question2 Efficient Function Calls

Algorithm:

To make one function call efficient, we need to minimize the number of steps to evaluate all it's argument. Since we can make only one function call at the same time, we'd better start evaluating the argument(variables/nested function call) takes longer time, then we can get maximum parallelism, therefore the total number of steps is minimized.

Step 0: To make one function call optimally, we need to choose the best sequence to evaluate all it's argument(variables/nested function call), and each arguments need to be solved optimally too.(which means their arguments need to choose the optimal sequence to evaluate) .

Step 1: Use a tree, where each tree node represent a function call. In the example given, root node represent the outer most function call f(), where all it's immediate children represent the nested function call arguments of f(). (In the tree structure,variables/constant argument not considered since no time is spent on evaluating them). Each tree node record the number of steps to finish that function call.

Step 2: For a tree node $t(arg_1, arg_2, ...arg_k)$, where all arguments are nested function calls, if we evaluate arguments sequentially, the total number of steps on node t should be:

$T(t) = max(1 + T(arg_1), 2 + T(arg_2), ..., k + T(arg_k))$

The reason is quite obvious, we make one function call in each step. Function t() is called in step 0, $arg_1()$ in step 1,..., $arg_k()$ in step k. The finish time is determined by the nested function call that returns last.

Step 3:

Initialization: Do the level order traversal of the tree, set all the leaf node to be 1.(Since there is no more nested call, it only takes 1 step to finish these function call).

Then do the post order traversal(bottom up):

run getNumSteps(root)

```
getNumSteps(node):
    while(node has at least one child){
```

```
    //If it does not have any child, it's a leaf node
    //which is handled in the initialization part
        for each child child_i of node:
            call getNumSteps(child_i)
    }
    sort all the children by running time (descending)
    set node.val=max(1+child_1.val,2+child_2.val,...,k+child_k.val)
```

Step 4:Construct the solution: Another field in the tree node is needed to record the start time of each function call. This time, we need solve it top down. The root function is called in step 0.

set root.start=0;

then run topDown(root)

```
topDown(node){
    print("step"+node.val+" node.function name")
    for every child_i of node{
    //child is indexed from 0
        child_i.start=node.start+i+1
        topDown(child.child_i)
    }
}
```