# 并行化快速排序——中序遍历算法

并行化快速排序算法主要包括两个步骤：1. 按进程级别构造二叉搜索树；2. 对前一步构造的二叉搜索树进行中序遍历即可得到排序结果。

进行中序遍历的流程同样包含两个步骤：1. 以每个进程节点的父节点为根节点并发地统计其左右子树的节点数量；2. 自根节点开始，每个进程并发地计算其对应节点所处中序遍历结果的位置。上述两个步骤对应的算法伪代码分别为：count_descendants、find_order。

经过 count_descendants 算法后，我们可以得到二叉树中每个节点的左右子树的数量。之后，我们可以在 find_order 算法中，自二叉树的根向下计算每个节点在中序遍历结果中的位置编号。二叉树的根节点的编号即为：根节点左子树的节点数量加 1。而当前节点为父节点的左孩子时，节点的编号为：父节点的编号减去该节点右子树的节点数量，再减去 1。当前节点为父节点的右孩子时，节点的编号为：父节点的编号加上该节点左子树的节点数量，再加上 1。

具体算法伪代码如下：

```
procedure count_descendants:
begin
    for each processor i do
    begin
        leftnumber[i] := rightnumber[i] := Yp := 0;
    end for
    repeat for each processor i != root do
    begin
        if leftchild[i] = rightchild[i] = n+1 then Yp := 1;
        if leftchild[i] = n+1 and rightchild[i] <= n and rightnumber[i] > 0
            then Yp := rightnumber[i] + 1;
        if leftchild[i] <= n and leftnumber[i] > 0 and rightchild[i] = n+1
            then Yp := leftnumber[i] + 1;
        if leftchild[i] <= n and leftnumber[i] > 0 and rightchild[i] <= n and rightnumber[i] > 0
            then Yp := leftnumber[i] + rightnumber[i] + 1;
        if i != root and Yp != 0 then
            if i = leftchild[parenti]
                then leftnumber[parenti] := Yp;
            else rightnumber[parenti] := Yp;
            exit
    end for
end count_descendants
```

```
procedure find_order:
begin
    for each processor i do
        order[i] := 0
    end for
    order[root] := leftnumber[root] + 1;
    repeat for each processor i != root do
        Zp := order[parenti];
        if Zp != 0 then
            if i = leftchild[parenti] then
                order[i] := Zp - rightnumber[i] - 1;
            else order[i] := Zp + leftnumber[i] + 1;
        exit
    end for
end find_order
```

在 count_descendants 算法中，我们并发地自叶子节点向各自的父节点计算子树节点数量。由于二叉树同一层的节点可以并发地计算节点数量，因此该步骤的时间复杂度与二叉树的高度相关。该步骤的时间复杂度为：$O(\log n)$。

在 find_order 算法中，我们并发地计算自父节点向下计算各进程节点在中序遍历结果的位置。由于在 count_descendants 算法中，我们已经统计了各节点的左右子树的节点数量。因此，在该步骤中，我们只需要根据当前节点与父节点的关系即可计算得到所处位置编号。而处于同一层的节点可以并发地计算节点位置，因此该步骤的时间复杂度同样与二叉树的高度相关。该步骤的时间复杂度为：$O(\log n)$。

综上所述，并行化快速排序中的中序遍历算法的时间复杂度为：$O(\log n)$。