Assignment No-A1

1 Title

Perform BST using Divide and Conquer on a computer cluster.

2 Problem Statement

Using Divide and Conquer Strategies design a cluster/Grid of BBB or Rasberi pi or Computers in network to run a function for Binary Search Tree using C /C++/ Java/Python/ Scala.

3 Learning Objectives

1. To study Divide and Conquer strategies.
2. To implement binary search using recursive techniques.
3. To form a cluster of computers in network.

4 Learning Outcome

1. We have learnt how to form a cluster of computer network.
2. We have implemented BST using Divide and Conquer strategy.

5 Mathematical Model

Let S be the system that represents the parallel Binary Search Tree program.
Initially,
S ={φ}
Let,
S = {*I,O,F*}
Where,
I – RepresentsInputset
O – RepresentsOutputset
F – RepresentsFunctionset

Inputset − I : *ElementsoftheBinarySearchTree.*
Outputset − O : *SearchedelementandtheInordertraversal.*
Functionset − F : *F* = {*F*1,*F*2,*F*3}
*F*1−*RepresentsthefunctiontoinsertelementintoBinarySearchTree. F*1(*Ri,Ei*) → {*E*1,*E*2.....*En*}∪{*Ei*} = {*E*1,*E*2.....*En,Ei*}
Where,
Ri : *Rootoftheithprocessor.*
Ei : *ithelementoftheBinarySearchTree.*
*F*2 − *RepresentsthefunctiontoprintInordertraversal. F*2(*Ri*) → {*E*1,*E*2.....*En*}*Where,*

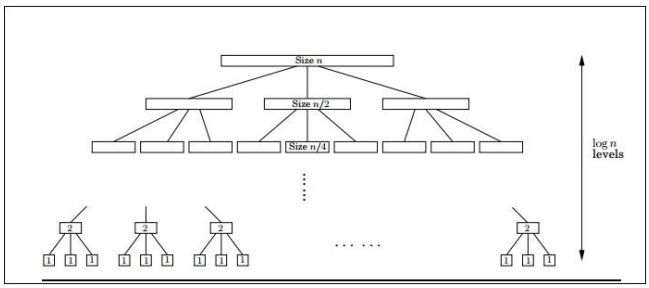
Ri: Root of the ith processor.
Ei: ith element of the Binary Search Tree.
F3 - Represents the function to find element from the Binary Search Tree.
F3(Ri, K) → {*K*}
Where,
Ri : *Rootoftheithprocessor.*
K : *Searchelement. Finally,*
S = {*I,O,F*}

6 Theory

6.1 Divide And Conquer

Divide-and-conquer is a top-down technique for designing algorithms that consists of dividing the problem into smaller sub-problems hoping that the solutions of the sub-problems are easier to find and then composing the partial solutions into the solution of the original problem.
Little more formally, divide-and-conquer paradigm consists of following major phases:

1. Breaking the problem into several sub-problems that are similarto the original problem but smaller in size,
2. Solve the sub-problem recursively (successively and independently), and then
3. Combine these solutions to sub-problems to create a solutionto the original problem.



Divide-and-conquer algorithms often follow a generic pattern: they tackle a problem of size n by recursively solving, say, a sub-problems of size n/b and then combining these answers in O(nd) time, for some a, b, d greater than 0 their running time can be captured by the equation

T(n) = aT(dn/be) + O(n^d)

6.2 Binary Search Tree

Binary Search tree is a binary tree in which each internal node x stores an element such that the element stored in the left subtree of x are less than or equal to x and elements stored in the right subtree of x are greater than or equal to x. This is called binary-search-tree property.
The basic operations on a binary search tree take time proportional to the height of the tree. For a complete binary tree with node n, such operations runs in θ (log$n$)*worst−casetime.Ifthetreeisalinearchainofnnodes,however casetime.TheheightoftheBinarySearchTreeequalsthenumberoflinksfromtherootnodetoth*

6.3 Implementation of Binary Search Tree

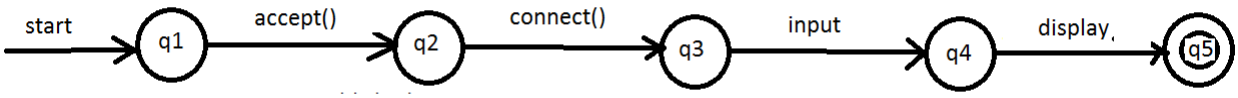
Binary Search Tree can be implemented as a linked data structure in which each node is an object with three pointer fields. The three pointer fields left, right and p point to the nodes corresponding to the left child, right child and the parent respectively NIL in any pointer field signifies that there exists no corresponding child or parent. The root node is the only node in the BST structure with NIL in its p field.

6.4 Parallelism and Scaling

The current setup works well enough on small amounts of data, but at some point data sets can grow sufficiently large that a single

computer cannot hold all of the data at once, or the tree becomes so large that it takes an unreasonable amount of time to complete a traversal. To overcome these issues, parallel computing, or parallelism, can be employed. In parallel computing, a problem is divided up and each of multiple processors performs a part of the problem. The processors work at the same time, in parallel.In a tree, each node/subtree is independent. As a result, we can split up a large tree into 2, 4, 8, or more subtrees and hold subtree on each processor. Then, the only duplicated data that must be kept on all processors is the tiny tip of the tree that is the parent of all of the individual subtrees.
Mathematically speaking, for a tree divided among n processors (where n is a power of two), the processors only need to hold n-1 nodes in common , no matter how big the tree itself is. Throughout the module, a processoraˆs rank is an identifier to keep it distinct from the other processors.

7 State Diagram



q1 : start state
q2: server listening
q3: connection established
q4: search
q5: display and end state

8 Algorithm

Insert:
Insert(Node root, Key k)

1. if (root == null) // insertion position found
2. return new Node(k);
3. if (k h= *root.key*)*//proceedtotheleftbranchroot* = *Insert*(*root.left,k*);

else // k *root.key,i.e.proceedtotherightbranchroot.right* = *Insert*(*root.right,k*); Search: BST-Search(x, k)

15 y ← *xwhiley*! = *nildo* if

key[y] = k then return y

3. else if key[y] h*ktheny* ← *right*[*y*]*elsey* ← *left*[*y*] return

("NOT FOUND")

Inorder-Walk: Inorder-Walk(x)

1. if x = nil then return
2. Inorder-Walk(left[x])
3. Print key[x]
4. Inorder-Walk(right[x])

9 Conclusion

Thus, we have successfully implemented BST using Divide and Conquer strategy on a cluster of computer network.