Asynchronous BFS

Project Report

Distributed Computing

To the

The University of Texas at Dallas

`

Submitted by:

Maxwell Hall

Prashant Prakash

Shashank Adidamu

The University of Texas at Dallas

Dallas – 75252

December, 2015

1. Setup

A Connected Graph is built $G = (V, E)$ with a distinguished source node i. For the shortest paths problem, we also assume that each undirected edge (i, j) has a nonnegative real-valued weight, $weight(i, j)$, known at both the end processes. We assume that the processes do not know the size of diameter of the network and that there are no UIDs.

2. Algorithm

The way Asynch BFS can be solved with the modification in Asynch Spanning Tree. If a process i initially identifies one of its neighbors, say j , as its parent , and later obtains information from another neighbor , say k , along a shorter path , then process i can change its parent designation to k. In this case , process i must inform its neighbors about its correction , so that they might also correct their parent designations.

$AsynchBFS_i$ automaton:

Signature:

Input:
$\quad receive(m)_{j,i}$, $m \in \mathbb{N}$, $j \in nbrs$

Output:
$\quad send(m)_{i,j}$, $m \in \mathbb{N}$, $j \in nbrs$

States:

$dist \in \mathbb{N} \cup \{\infty\}$, initially 0 if $i = i_0$, ∞ otherwise
$parent \in nbrs \cup \{null\}$, initially $null$
for every $j \in nbrs$:
$\quad send(j)$, a FIFO queue of elements of $\mathbb{N}$, initially containing the single element 0 if $i = i_0$, else empty

Transitions:

$send(m)_{i,j}$
Precondition:
$\quad m$ is first on $send(j)$
Effect:
$\quad$ remove first element of $send(j)$

$receive(m)_{j,i}$
Effect:
$\quad$ if $m + 1 < dist$ then
$\quad\quad dist := m + 1$
$\quad\quad parent := j$
$\quad\quad$ for all $k \in nbrs - \{j\}$ do
$\quad\quad\quad$ add $dist$ to $send(k)$

Tasks:

for every $j \in nbrs$:
$\quad \{send(m)_{i,j} : m \in \mathbb{N}\}$

3. Termination

For termination we use “Converge Cast” technique. We add acknowledgements for all messages, convergecasting the acknowledgments back to i as in AsynchBcastAck. This enables i to learn when the system has reached a stable state and then to broadcast a signal to all the processes to perform their parent outputs.

4. System Requirements

To run this Program “**JAVA 8**” must be available in the system. We are extensively using Java 8 features to solve the problem.

5. Sample Input and output

The system takes input from file in the format mentioned below:
Sample Input:

```
12, 7
1  1  0  1  0  0  0  1  0  0  0  0
1  1  1  0  1  0  0  0  0  0  0  0
0  1  1  0  1  1  0  0  0  0  0  0
1  0  0  1  0  0  1  1  0  0  0  0
0  1  1  0  1  1  0  1  1  0  0  0
0  0  1  0  1  1  0  0  0  0  0  1
0  0  0  1  0  0  1  1  0  1  0  0
1  0  0  1  1  0  1  1  0  0  1  0
0  0  0  0  1  0  0  0  1  0  1  1
0  0  0  0  0  0  1  0  0  1  1  1
0  0  0  0  0  0  0  1  1  1  1  1
0  0  0  0  0  1  0  0  1  1  1  1
```

Where 12 is the number of Process and 7 is the source Node. The two dimensional matrix is the Adjacency matrix for the graph.

Output:

```
The root is process 7
Process Parent Distance
1       8      2
2       1      3
3       5      3
4       7      1
5       8      2
6       5      3
7       -1     0
8       7      1
9       5      3
10      7      1
11      8      2
12      10     2
```

```
We also print the adjacency list to represent the output graph.
```

```
Adjacency List:
NODE 11 adjacency list: 8
NODE 1 adjacency list: 2 8
NODE 12 adjacency list: 10
NODE 2 adjacency list: 1
NODE 3 adjacency list: 5
NODE 4 adjacency list: 7
NODE 5 adjacency list: 3 6 8 9
NODE 6 adjacency list: 5
NODE 7 adjacency list: 4 8 10
NODE 8 adjacency list: 11 1 5 7
NODE 9 adjacency list: 5
NODE 10 adjacency list: 12 7
```

6. Challenges

The major problem we faced are following:

a. We faced problem with basic understanding of Algorithm and various steps involved and understanding each nodes participation in different rounds.
b. We faced problem with implementation of ConvergeCast technique.