**Parallelization of Maximum Flow Algorithm**

**Using OpenMP**

Done by V.Sai Venkata Krishnan CSE BE Loyola-ICAM College

under the guidance of Prof. Dr.Rupesh Nasre

Vertices : Numbered from 0 to V-1

1. **Sequential Algorithm :** This algorithm is based on Ford-Fulkerson Algorithm .

Graph is represented using the Flow Matrix G[i][j] represents the flow from ith vertex to jth vertex **.**

BFS is used to find the shortest augmenting path , from source to destination

With the shortest augmenting path found , the flow along the path is increased .

1. **Parallel Algorithm :**

**Idea for Parallelization :**

The BFS to find the paths from the source to destination is inherently Parallelizable

This can be implemented by employing a  **Level Order Traversal of the Graph**

All the vertices in the same level are processed simultaneously to find the next level of vertices .

**Constraints :**

1. **Queue** is inherently sequential . To ensure correctness w.r.t push and pop operation **Critical Constructs** are employed which increase the overhead on the execution and results in degradation of performance rather then upgradation , when compared to the Sequential Version .
2. **Parent** vector -> Parent[k] = j represents that kth vertex was discovered from the jth vertex in BFS routine .(updated in the BFS routine)

In sequential version using this will help us to find the flow along the shortest augmenting path efficiently.

But implementing this in Parallel program will require the use of **critical constructs to avoid Data Race .** This will again degrade the performance of the parallel program w.r.t sequential version .

**Solution :**

1. **Bag Data Structure : To implement the Parallel Breadth First Search[1]**
2. **Pennant** – tree of nodes (k>0)

Bag is implemented using Pennants.

Each node in the pennant has a left child and right child

Root of the pennant has left child only

Operations on Pennant : Union , Splitting the Pennant of size into two pennants of size

**Bag**  - collection of pennants , no two of which have the same size

Operations on Bag – Union , Insertion of Element in Bag

**Bag is a Multiset data structure** that can contain multiple copies of the same element . Thus any data race that occurs will have no affect on the correctness of the program . This is a  **Benign Data Race**

1. **Distance :** Distance[i] represents the level of the ith vertex w.r.t source vertex (computed in the Parallel BFS Routine)

Updating the distance vector can also occur simultaneously .

Irrespective of the vertex being processed in the same level , *l* , the adjacent vertex in the next level that are discovered will be assigned a

Level has *l+1* . So this does not result in data race as the value being written is the same independent of the vertex through which it is accessed .

This is also Benign Data Race .

1. **AllPaths :** It is computed using the Distance vertex. Starting with the destination vertex , all the vertices that are at a lower level than the destination are examined recursively till the source vertex is reached .

AllPaths consists of all the paths from the source to destintion

Then the minimum capacity available along each path is computed and the flow is increased .

The Parallel BFS is executed till there is a path from source to destination with a positive capacity .

1] A Work-Efﬁcient Parallel Breadth-First Search Algorithm (or How to Cope with the Nondeterminism of Reducers)

Charles E. Leiserson Tao B. Schardl

MIT Computer Science and Artiﬁcial Intelligence Laboratory