**Connected Components in Graph using MPI java**

For example consider the following input file:

Vertex : edges

0 : 1,2

1 : 0

2 : 0

3 : 2

4 : 0

5 : 0

6 : 2

7 : 2

|  |
| --- |
| Rank0 |

|  |
| --- |
| Rank1 |

Expected Output: 1

Rank0

**Used functions to get number of connected components:**

To find connected components in graph, I have used Union Find algorithm.

Find(): Determines whether given two elements belong to same root or parent

Union(): Joints two graphs into single graph

getConnectedComponents(): using Union and find, returning the number of connected components

mergeConnectedComponents (): Merge function will merge subsets if they belong to same root, after processed data is received by the master node.

**MPI Strategy:**

Calculated the stripe size according to number of vertices and number of nodes. Also calculate their start and end to send related data to each node.

int stripe = numberOfVertices / size;  
  
int start = myRank \* stripe;  
int end = start + stripe;

Master node i.e. rank0 will send data to each node. Only calculated stripe data is sent to other nodes.

if (myRank == 0) {  
 for (int i = 1; i < size; i++) {  
 MPI.*COMM\_WORLD*.Isend(graph, i \* stripe, stripe, MPI.*OBJECT*, i, i);  
 }  
}

Every node will receive their data and work on their own data set. Nodes will find their own connected components and return that data to master node.

if (myRank > 0) {  
 MPI.*COMM\_WORLD*.Recv(graph, start, stripe, MPI.*OBJECT*, 0, myRank);  
}

if (myRank > 0) {  
 MPI.*COMM\_WORLD*.Isend(connectedComponents, 0, numberOfVertices, MPI.*INT*, 0, myRank);  
}

Master node will merge that data to his calculated connected components to find if there are any subsets that have same root or parent. And returns the number of connected components in graph.

if (myRank == 0) {  
 for (int i = 1; i < size; i++) {  
 int[] target = new int[numberOfVertices];  
 MPI.*COMM\_WORLD*.Recv(target, 0, numberOfVertices, MPI.*INT*, i, i);  
 *mergeConnectedComponents*(connectedComponents, target);  
 }  
 Set<Integer> set = new HashSet<>();  
 for (int root : connectedComponents) {  
 set.add(root);  
 }

**Data Sharing strategy:**

Rank 0 : Send : every other rank with stripe size data

Other Rank : Recv : stripe size data, performs operations returns to master node

Rank 0: Recv data one by one : merge with its own data and finds final #connected components

**Scope of Improvement:**

* Merge function can be parallelizes
* Data rend recv can be done with neighboring node and eventually with master node such as rank0-rank1 and rank2-rank3, eventually rank0-rank2.