CS633A Parallel Computing - Assignment 2

March 29, 2021

Submitted By:

Akash Patel (20111007)

Prasoon Sahu (20111042)

1 How to Use:

- bash Run.sh

2 Files:

- **Makefile:** Compiles all required files.
- **src.c:** Contains all Mpi code.
- **run.sh:** Script to run the complete program in one go.
- **plottingScript.py:** Script to plot the graph, use python3 to run it.
- **NodeAllocator.sh:** Checks the available nodes and saves their information in Data_Temp.txt file.
- **CreateHostfile.py:** Takes Data_temp.txt file as input and produces hostfile. It arranges the nodes in the hostfile also takes care of load on the nodes.
- **hostfile:** Host file to be used by MPICH program.
- **Data_X.txt:** Stores Temporary data.

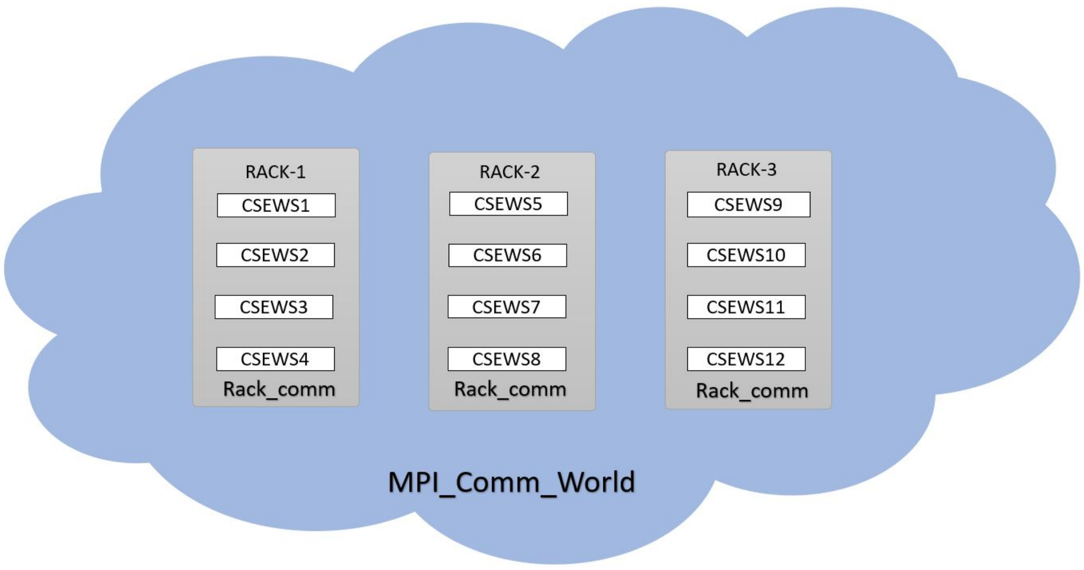
3 Program Documentation:

3.1 Points to be mentioned

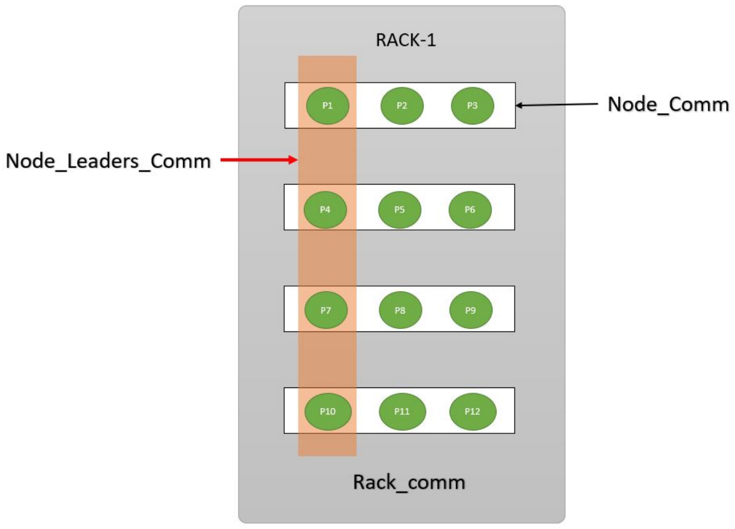
- **Root Selection:** We have Randomly selected the root for every operation, to get more random result.
- **Hostfile creation:** We have used our own hostfile creator script which creates hostfile by ssh each node and arranging them in ascending order of users on that node.

3.2 Communicators:

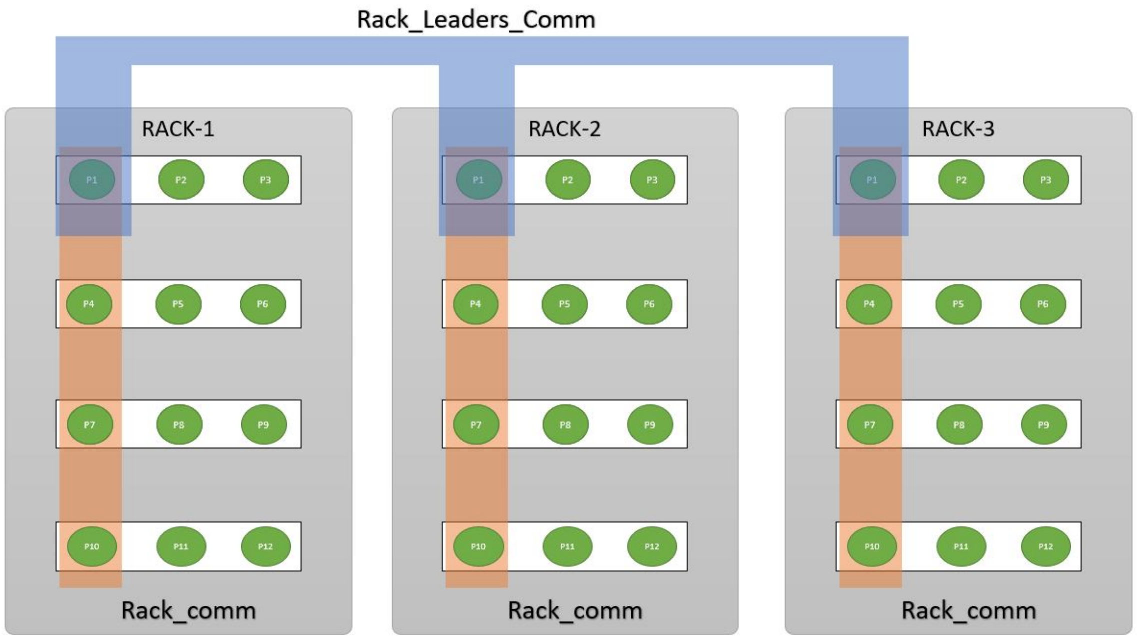
- **MPI_COMM_WORLD**: Contains all ranks.



- Rack_Comm : Every Rack has its own Communicator to communicate inside the Rack.
- Node_Comm : Every Node has its own Communicator to communicate inside the Node. Processes which are assigned to same node will be in same Node_comm.

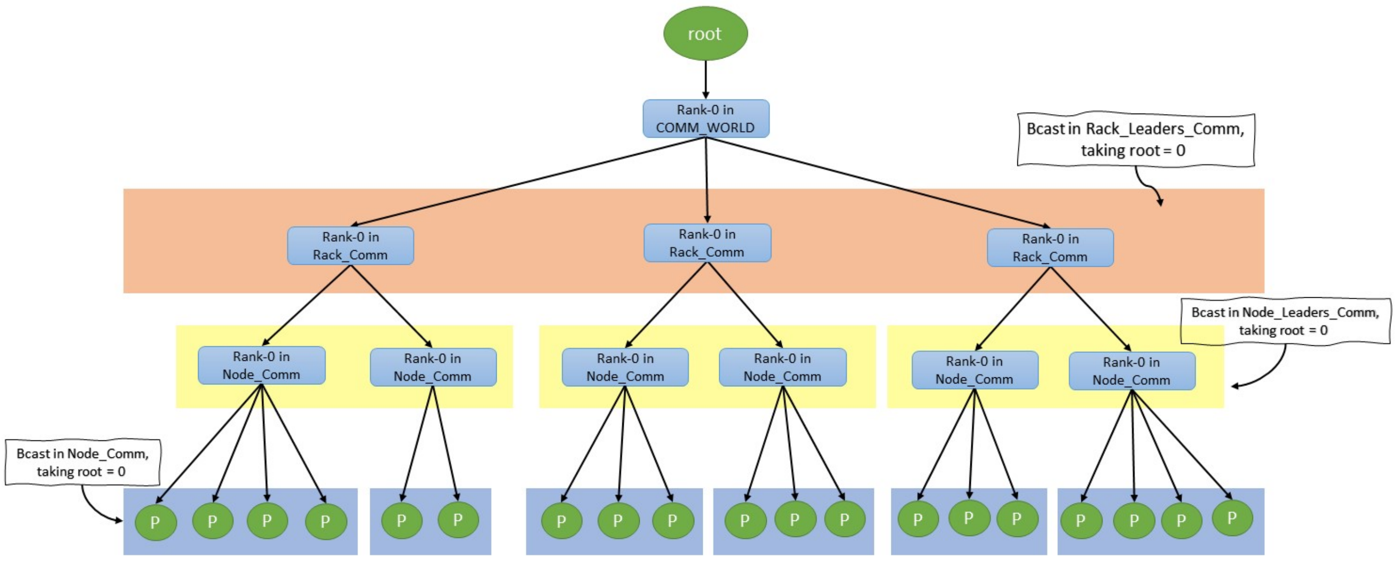


- **Node_Leaders_Comm** : Every node has its leader rank/process, (The processes whose rank is 0 in Node_Comm will become node leader of respective node). And all these leaders of same Rack will come together to make a new communicator called Node_Leaders_Comm.
- **Rack_Leaders_Comm** : Same as leader in each node there is Leader in each rack. Rack Leader has 0 rank in its Rack_Comm. Rack Leader of every Rack come together to form a new Communicator called Rack_Leaders_Comm.
  Leaders will have special integer variable named as *leader* whose value will be 1; This is to check if the process is leader(rack leader/node leader) or not.



3.3 Optimized Communication

3.3.1 BCast


root
Rank-0 in COMM_WORLD
Bcast in Rack_Leaders_Comm, taking root = 0
Rank-0 in Rack_Comm
Bcast in Node_Leaders_Comm, taking root = 0
Rank-0 in Node_Comm
Bcast in Node_Comm, taking root = 0
P

BCast is done in 4 Stages.

- **Stage 1:** Root Process Sends the data to Rank-0.
- **Stage 2:** As the Rank-0 is also the member of Rack_Leaders_Comm it Broadcast the data in this Rack Leaders Communicator.
- **Stage 3:** Till now every Rack Leader has the data. And this Rack Leader is also the one of the member of Node_Leaders_Comm so it will Bcast the data in Node_Leaders_Comm. Hence every Node Leader have data now.

- **Stage 4:** Every Node leader will BCast the data in its respective node using Node_Comm. Hence now every Rank have the data, BCast Completed.

3.3.2 Reduce

It is almost opposite of BCast. It is also completed in 4 steps. And computation is done in every stage by the root.

- **Stage 1:** Reduce is done on node level using Node_Comm and taking root as rank-0, in Node_Comm. Rank-0 is also the Node leader of that Node.
- **Stage 2:** Reduce is done on Node_Leaders_Comm in every Rack and taking root as Rank-0 in Node_Leaders_Comm. Rank-0 is the also the Rack leader of that Rack.
- **Stage 3:** Now we will use Rack_Leaders_Comm to perform reduce operation and taking root as Rank-0. Hence Now the Rank-0 in the MPI_COMM_WORLD has the required data.
- **Stage 4:** In the last stage we will send the resultant data form Rank-0 to the root rank which we have randomly generated in the starting of the program, in this stage we will use MPI_COMM_WORLD.

3.3.3 Gather

Optimization in Gather operation is same as done in Reduce operation. Here we dont have to do calculation in each stage.

- **Stage 1:** Gather is done on node level using Node_Comm and taking root as rank-0, in Node_Comm. Rank-0 is also the Node leader of that Node.
- **Stage 2:** Gather is done by all the node leaders inside every rack using Node_Leaders_Comm and taking root as Rank-0 in Node_Leaders_Comm. Rank-0 is the also the Rack leader of that Rack.
- **Stage 3:** In this we individually send data from rack leader to process having rand-0, Hence for this every rack leader tells the rank 0 that how much data it is going to send and then it sends its Data. here we will use Rack_Leaders_Comm to perform send and Receive operation and taking root as Rank-0. Hence Now the Rank-0 in the MPI_COMM_WORLD has gathered the data from all the the ranks.
- **Stage 4:** In the last stage we will send the gathered data form Rank-0 to the root rank which we have randomly generated in the starting of the program, in this stage we will use MPI_COMM_WORLD.

3.3.4 AlltoAllv

In AlltoAllv every process sends different amount of data to other processes, so it is very important for every process to know that how much other processes are going to send. Hence to take are of this problem we are creating 2D-Array in Rank-0 and BCasting this Array to all other ranks and hence now every process now knows how much other processes are accepting from it and how much it will get from other processes.

- Our Optimal Version Didn't Worked.

4 Problem Faced:

- In AllToAllv it was necessary to tell each and every process that how much other processes are going to send so we have to create 2D matrix and send it to every process so that every process will get to know that how much other is sending. But to convert this process to optimal version became very difficult. Because we can't tell every process that the required process is in which rack.
- In optimal MPI_Gather after gathering every processes's data in rack leader,now the data that Rack leader will send to rank 0 process is different because every rack has different number of nodes. So the rack leaders will first tell the rank 0 that how much data they are going send.
- Creating Rack leaders and Node leaders is difficult to assign any rank to be a leader hence for this we used Real_Node_Leader and Real_Rack_Leader variable to the rank which are rank 0 in their respective nodes or Racks.

5 Experimental Setup

- MPICH should be installed in the system.