README

Ankita Dey, 20111013 P J Leo Evenss, 20111038

This assignment helps us perform optimizations on the collectives, MPI_Bcast, MPI_Reduce, MPI_Gather and MPI_Alltoallv based on CSE cluster topology information that has been provided to us.

Code Explanation and Optimizations:

1 Apart from data size, additional arguments, number of groups, ppn is also passed. This helps us do topology aware optimization of collectives.

2 All the variables needed are initialized appropriately with P and rank indicating total number of processes and rank of current process in MPI_COMM_WORLD respectively; and D, no_groups and ppn passed through command line indicating data size, number of groups and processes per node respectively.

3 Intra-group subcommunicator, newcomm and and inter-group subcommunicator containing one node per group, headcomm are created using MPI_Comm_split. Only headcomm consisting of processes divisible by size of each group (i.e. 0th rank process of newcomm) are used in Bcast, Reduce and Gather. We will call these processes, head process in the rest of the text.

4 Default MPI_BCAST is implemented.

5 Optimized MPI_BCAST is implemented in 2 steps using MPI_Bcast:

5.1 Message is broadcasted from root node to head nodes of all group using headcomm subcommunicator. This requires inter group communication.

5.2 Message is broadcasted from head node to other nodes of each group using newcomm subcommunicator. This requires no intra group communication among all process giving us an advantage.

6 Default MPI_REDUCE is implemented.

7 Optimized MPI_REDUCE is implemented in 2 steps using MPI_Reduce:

7.1 Message is reduced from all nodes of each group to the respective head node of that group using newcomm subcommunicator. This requires no intra group communication among all process giving us the advantage.

7.2 Message is reduced from head nodes of all groups to the root node using headcomm subcommunicator. This requires inter group communication.

8 Default MPI_GATHER is implemented.

9 Optimized MPI_GATHER is implemented in 2 steps using MPI_Gather:

9.1 Message is gathered from all nodes of each group to the respective head node of that group using newcomm subcommunicator. This requires no intra group communication among all process giving us the advantage

9.2 Message is gathered from head nodes of all groups to the root node using headcomm subcommunicator. This requires inter group communication.

10 Default MPI_ALLTOALLV is implemented.

11 For Optimized MPI_ ALLTOALLV, all the processes of headcomm subcommunicator (created using MPI_split with (size %groupsize) being the color) is used. This is because we require all the nodes to get internode messages in alltoallv (every node must be able to get an unique message from every other node). It is implemented in 2 steps using MPI_Gather:

11.1 Message is gathered from all nodes of each group to every other node of that group using newcomm subcommunicator. This requires no intra group communication among all process giving us the advantage

11.2 Message is gathered from head nodes of all groups to the root node using headcomm subcommunicator. But this time all nodes are made root node in a loop. This requires inter group communication.

11.3 Each process becomes root in the gather collective and the starting point of the buffer is different for each of them. This allows each process to gather unique data from all the other processes which is the what happens in alltoallv.

12 Each of the default and optimized version of the collectives are run 5 times and the time taken for each run is noted. Finally the average of this 5 runs and time taken to create the sub communicators is written to output.csv file. Here src.c ends.

13 Barcharts with error bars for every data point obtained from respective *.txt , * is {Bcast,Reduce,Gather,Alltoallv} is plotted using plot.py.

Optimization Reason:

P=nodes/group ; ppn=cores/process per node;

The Data is shared among the nodes of inter groups, so the number of communications becomes very high increasing the traffic in the cluster (which follows a star topology with inter-group distance as 4 hops and intra-group distance as 2 hops) .

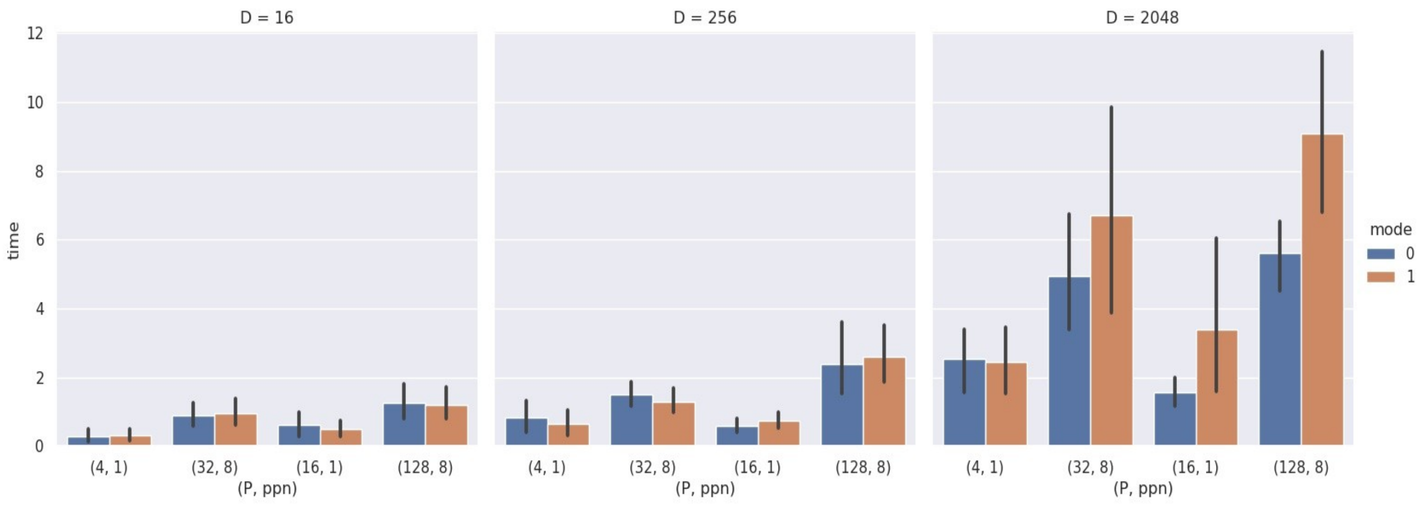
The Groups are chosen as 4. The Communication between process is done as all nodes communicate to a leader node using a new communicator and all head nodes perform internode communication via a head communicator, forming a tree structure at top two levels.

This structure reduces the traffic to a single message among inter group instead of ppn*p messages and gives a topology aware optimisation.

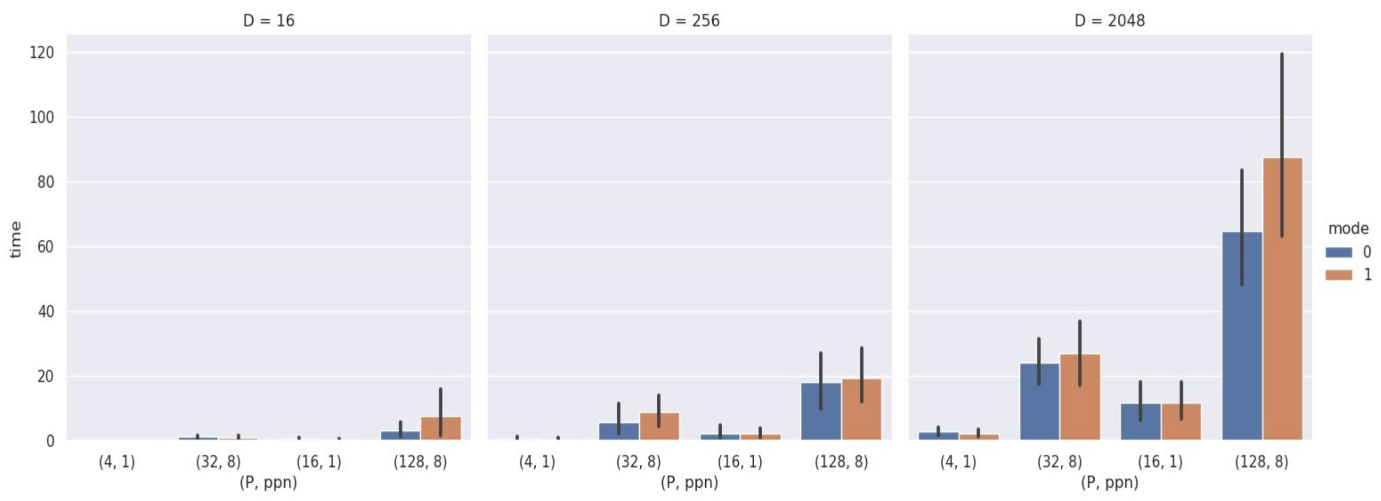
Plots:

1. Barcharts with error bars is plotted with time (in seconds) for each data size per node count per core count for every collective function.
2. Mode 0 represents Default collective and mode 1 represents Optimization applied.
3. The 4 plots for Bcast , Reduce , Gather , Alltoallv are shown below.
4. Note: The outputs and graphs can change according to the traffic on the cluster

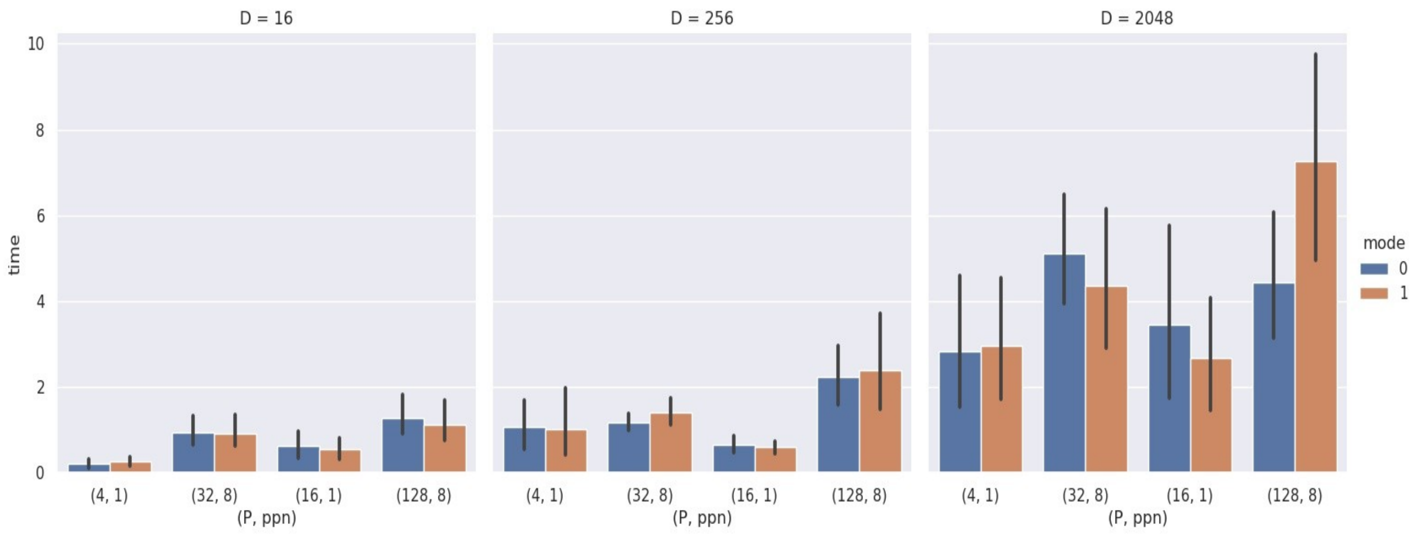
Bcast Plot:



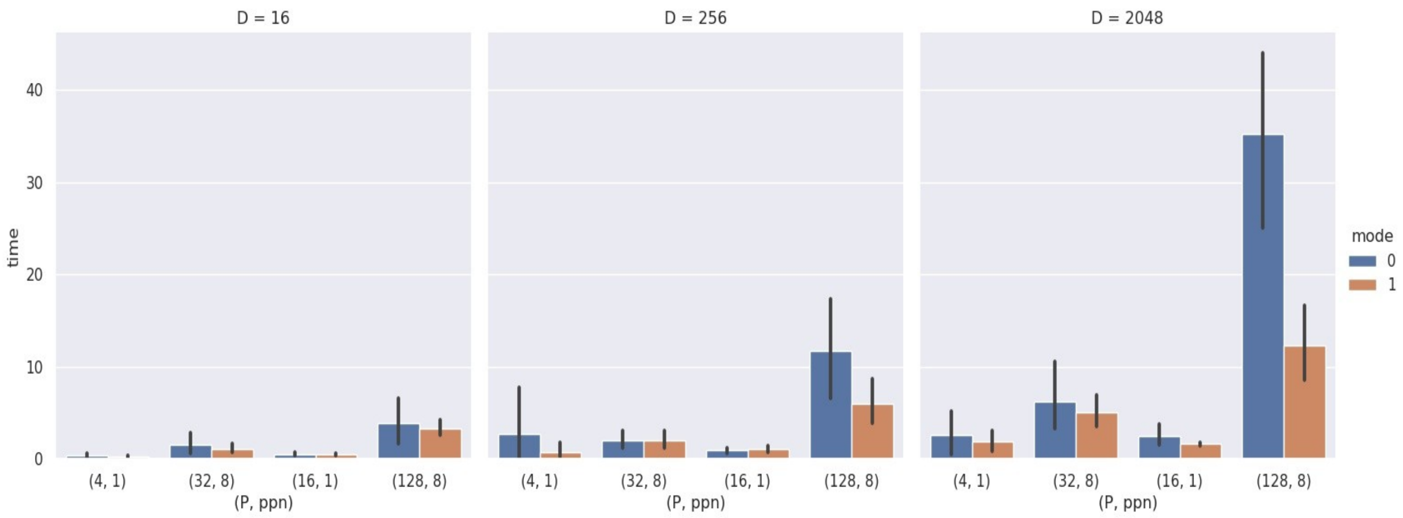
Reduce Plot:



Gather Plot:



Alltoallv Plot:



Observations:

4.1 The Bcast plots show good optimizations because the number of message transfer is reduced , reducing the number of hops to be traversed. But when executed on the last days of submission, the nodes because of high traffic, reduced the optimization to some extent for large data size of 2048Kb as there was more queuing in the nodes of the cluster.

4.2 The Reduce plots also gave good results which reflected well for 2048Kb message sizes , because instead of transfer of individual messages across the nodes and then reducing , leading to lot of inter-node message transfer, by using sub communicators , the number of messages were reduced among the inter group nodes , following a tree like structure with leaders commincating information of entire group to root, thereby reducing the hops and in turn the traffic.

4.3 The Gather plots showed considerable optimizations for the small number of processes , but as the number of processes increased , each process sends D size of data to the nodes, and hence each nodes have D*ppn size of messages , which in turn communicate D*ppn*P size of messages across groups , thereby increasing the queuing time and transmission time for nodes. So for large messages and large processes spun across groups, Gather performed as expected taking more time than default which sends individual messages of d sizes.

4.4 The Alltoallv plots showed good optimizations , because in all to all every process communicates a variable amount of data to every other process, leading to n^2 message transfers, but the optimisations based on tree reduces it.

4.5 Some collectives has implicit optimizations performed by mpi_library for different message sizes , so the default cases for higher message sizes are faster.

Experimental Setup:

1 The groups are fixed as 4 in the aim of maintaining the equality and obtaining a overall insight of the optimization .

2 The communication among nodes of the cluster provided a complete graph based structure for default cases, so we aimed to reduce it to a tree like structured by maintaining a leader process among the nodes and the groups.

3 We tried maintaining a leader at 3 levels as intra node , inter node and inter group. But that worsened the time for the some configurations due to bottleneck at the nodes due to increase in transmission time.

Code Documentation:

1 Run chmod u+x run.sh

2 Now run ./run.sh (it starts running whole configuration, time to complete depends on traffic on nodes, from 2 to 2.5hrs).

3 The code starts compiling using make command and output of src.x is obtained.

4 The hostfiles is generated on the fly using script.py from the helper scripts in the resources section.

5 The src.x is executed using mpiexec command with for loop execution sequence as specified in assignment.

6 The *.txt file (* = Bcast, Reduce,Gather,Alltoallv) is generated which captures the timing of all the configurations as per the execution sequence.

7 The plots script is executed using python3 plot.py and generates four plots with names plot_X.jpg where X is the respective collective.