# Topology aware Collectives

**Rudranil Bhowmik[1] and Sanket Bodele[1]**

[1] *Indian Institute of Technology,Kanpur*

April 22, 2019

**Abstract:- In this Exa-scale era of of parallel computing there is always an increasing complexity in the heterogeneity of the communication channel ,this leads to variation in the performance of the machine because the communication channel cannot look into the inner workings as there is quite some loss due to packet collision and resistance .This can be avoided as because MPI allows us to fix the rank of the processes in a hierarchical fashion so there is much less collision and transmission don't take much time to travel through a slower channel.So in this project we propose two heuristics in order to tune the MPI-Allgather algorithm where we try to exploit the underlying architecture for communication patterns . In our experimental result there is a 12-40 percent increase in speed up for the MPI-ALLGather implementation.**

## 1 Introduction

In this exascale era of parallel computing the number of nodes are increasing exponentially as a result of this in the modern HPC systems it is becoming more and more complex at intra and inter level layers. The combination of multi level hierarchies in the processor and the way they are arranged in a supercomputer setting really leads to a non uniform memory access which effects multiple cache and hierarchies.
Previous works have used a the topology aware mapping of processes in order to exploit the mapping scheme in MPICH and thus creating a efficient utilization of systems in a modern large scale system. Topology awareness is linked with both hardware and process topology which represents communication in an application. Topology aware mapping can be specifically geared towards collective communications and since collective communication generally takes a major portion of time in parallel application the deduction in an overhead time for each collective process can drastically improve time for a parallel application.The core merit of such algorithm can only be achieved by appropriate mapping of processes which is quite challenging for different mapping processes.

In our work we try to to remap the processes of the underlying topology so that the we get a time improvement during the transfer of data.We propose fine tuned algorithm for finding the most efficient mapping in a given scenario and then use it to modify the initial layout so that the MPI-ALLGATHER runs in the most efficient manner in any given scenario.In our case since we map the processes in a matter of bandwidth present in between two at a given scenario the algorithm works most of the time.

## 2 Related Work

Early work on collective communication focused on developing optimized algorithms for particular architectures, such as hypercube, mesh, or fat tree, with an emphasis on minimizing link contention, node contention, or the distance between communicating nodes [1][2].More recently, Vadhiyar et al.have developed automatically tuned collective communication algorithms .They run experiments to measure the performance of different algorithms fora collective communication operation under different conditions (message size, number of processes) and then use the best algorithm for a given set of conditions [3]. Benson et al. studied the performance of the allgather operation in MPICH on Myrinet and TCP networks and developed a dissemination allgather based on the dissemination barrier algorithm [4].Lastly Mirsadeghi et al. computed a distance matrix using hwloc library and then used a mapping algo to implement allgather [5].

```
8 processes
 0 0.000000 0.012235 0.012359 0.022103 0.021609 0.033389 0.038791 0.050430
 1 0.012235 0.000000 0.034503 0.020846 0.034099 0.038954 0.050440 0.038700
 2 0.012359 0.034503 0.000000 0.047565 0.046227 0.050572 0.038688 0.032320
 3 0.022103 0.020846 0.047565 0.000000 0.096737 0.038809 0.032245 0.022932
 4 0.021609 0.034099 0.046227 0.096737 0.000000 0.067622 0.024339 0.023499
 5 0.033389 0.038954 0.050572 0.038809 0.067622 0.000000 0.046626 0.013748
 6 0.038791 0.050440 0.038688 0.032245 0.024339 0.046626 0.000000 0.026340
 7 0.050430 0.038700 0.032320 0.022932 0.023499 0.013748 0.026340 0.000000
minimum overhead mapping (#processNo->#processNo)
0 1 3 7 5 6 4 2
```

**Figure 1:** *Mapping heuristic based on nearest neighbour*

# 3 Implementation

Topology awareness collective is generally done in two ways

1. Changing the underlying pattern of communicatiom
2. Changing the mapping of the processes

In our algorithm we try to change the mapping of the underlying algorithm by using how much each and every processes on each nodes take time to communicate within one another.Having found the mapping this way , a new communicator is created with with with process ranks based on the mapping results and thus we create a new mapping for our program to run on.

## 3.1 Mapping Heuristics

In this algorithm we use a ping pong algorithm in order to find a total physical topology distance of the nodes from one another.In order to find this we need to start the by setting up all the nodes and then pass data packets in between them and then find the time for the data packets to reach in between nodes.

The algorithm for this is mentioned below.

**Algorithm for Mapping heuristics**

1. Start with a set of nodes for mapping it.
2. Repeat until all nodes completes a to a
3. Select a single node and then send similar size message to all other nodes
4. Wait for the message to complete transmission
5. Find the time taken to complete transmission.
6. Repeat step 2-5 multiple times and find the mean.

## 3.2 Recursive doubling

In case of recursive doubling algorithm we use the fact that the message passed in the last stages of the communication will be largest so we try to map the stages which will communicate in the later parts of the algorithm will be kept closer.Since rank x is our reference core and p denotes the total number of processes $x \bigoplus \frac{p}{2}$ is mapped close as possible to rank 0.Now we need to choose betwwen 0 or $\frac{p}{2}$ to be our reference core the explanation to it is given in the algorithm described below

**Algorithm for Recursive doubling**

**Input:-**Number of processes p, Physical distance matrix D from Ping Pong Benchmark.
**Output:-**Mapping of the new rank for each process.

1. Fix rank 0 on its current core and make it the reference rank
2. Starting from the last stage i/2.
   (a) While there exists process to map
      i. If i/2 is mapped then i=i/2
   (b) new rank = reference rank (XOR) i
   (c) Find a target core closest to reference core.
   (d) Map new process onto the target
   (e) If mapped two process with reference map then
      i. Update reference core
      ii. Restart from last stage.

## 3.3 Ring Communication

In case of ring communication our algorithm is straight forward as because in case of the ring algorithm the nodes communicates with its adjacent nodes so if we start mapping from a given reference and continue mapping until we exhaust all our nodes then the communicating pattern which we get will be the most efficient of all.So if we start with rank 0 and make all the way to rank 1 as close as possible similarly 2 placed as close to 1 and so on.

**Algorithm for Ring Communication**

**Input:-**Number of processes p, Physical distance matrix D from Ping Pong Benchmark.
**Output:-**Mapping of the new rank for each process.

1. We take process 0 as a reference rank
2. While there exists new nodes.
   (a) Find the closest node to the reference rank.
   (b) Allocate it the new rank
   (c) Make this the refence core
3. Repeat 2 until complete

This way we try to map and then using MPIAllgather for the same layout would yield different result from before as we would optimise the algorithm for a given mapping in our network.

# 4 Experiments

We conduct the experiments by implementing MPI-Allgather on our mapped heuristic and on the default heuristic and emunerating it for a couple hundred times and then finding the mean of the time taken for gathering 1,2,4,8,16 Kb and 128-2048 kb of data for the ring algorithm and recursive doubling .

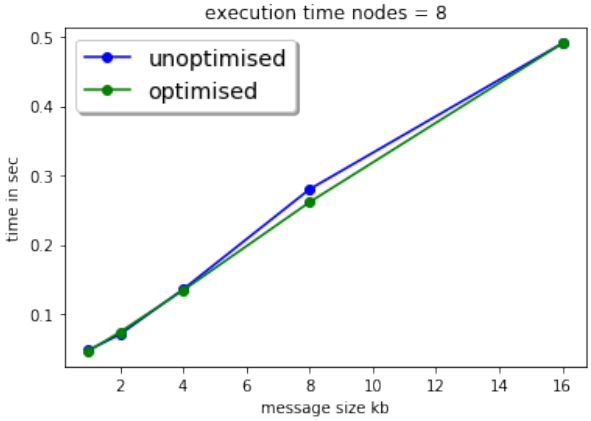


**Figure 2:** *message size vs time for ring on 8 nodes*

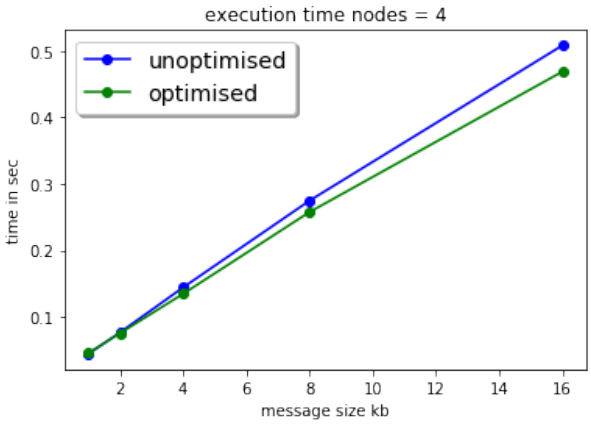


**Figure 3:** *message size vs time for ring on 4 nodes*

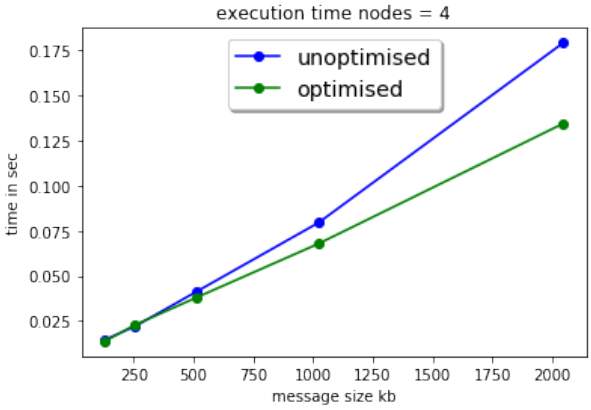


**Figure 4:** *message size vs time for recursive doubling on 4 nodes*

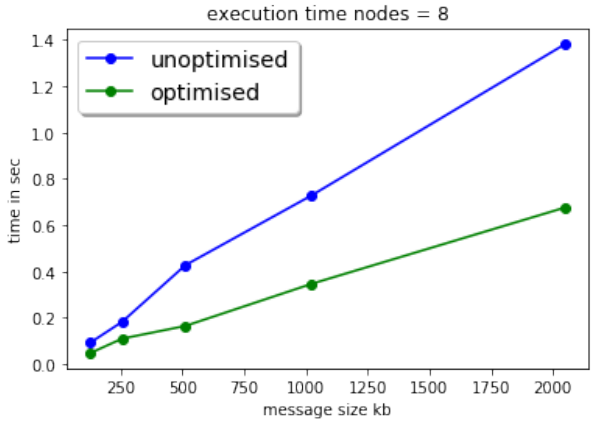


**Figure 5:** *message size vs time for recursive doubling on 8 nodes*

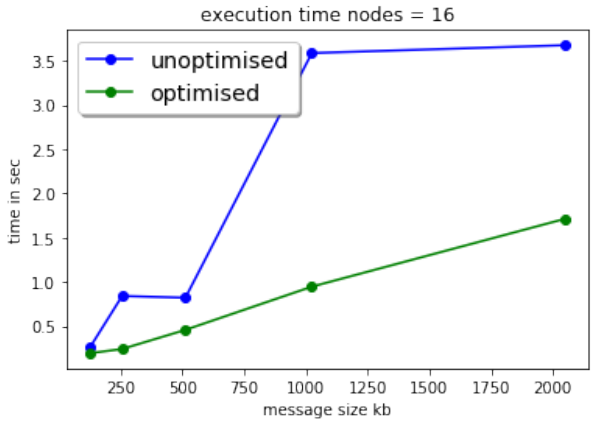


**Figure 6:** *message size vs time for recursive doubling on 16 nodes*

# 5 Conclusion

Our new mapping heuristics showed considerable speed improvement with lower overhead over non topology aware inbuilt algorithms for MPI all gather. Experimental results were calculated multiple times to confirm the results. We used cse cluster to run the algorithms as the inter node bandwidth is very less as compared to hpc. So, we were able to get good results as distances between the two nodes affected more in cse cluster than in the hpc. We also observed that recursive doubling showed better results than the ring algorithm with proposed heuristics. This is due to the underlying algorithm for recursive doubling scales well with the new heuristics. This fact was not mentioned in the paper.

Finally, The inbuilt algorithm for mpi all gather also uses some heuristics for optamization but it is limited

to intra node level. The new heuristics exploits the intra as well as inter node communications and able to scale the already implement version of MPI allgather.

## References

[1] M. Barnett, R. Littlefield, D. Payne, and R. van deGeijn. *Global combine on mesh architectures with-wormhole routing.* Proceedings of the 7th International Parallel Processing Symposium,April 1993.

[2] S. Bokhari. *Complete exchange on the iPSC/860.Technical Report*. NASA Langley Research Center, 1991.

[3] Sathish S. Vadhiyar, Graham E. Fagg, and Jack Dongarra. Automatically tuned collective communications Proceedings of SC99: High PerformanceNetworking and Computing, November 1999.

[4] Gregory D. Benson, Cho-Wai Chu, Qing Huang,and Sadik G. Caglar. A comparison of MPICH all-gather algorithms on switched networks Recent Advances in Parallel Virtual Ma-chine and Message Passing Interface, 10th EuropeanPVM/MPI Users' Group Meeting, pages 335–343.Lecture Notes in Computer Science 2840, Springer,September 2003.

[5] Seyed H. Mirasdeghi and Ahmad Afsahi Topology-Aware Rank Reordering for MPI 2016 IEEE International Parallel and Distributed Processing Symposium Workshops .