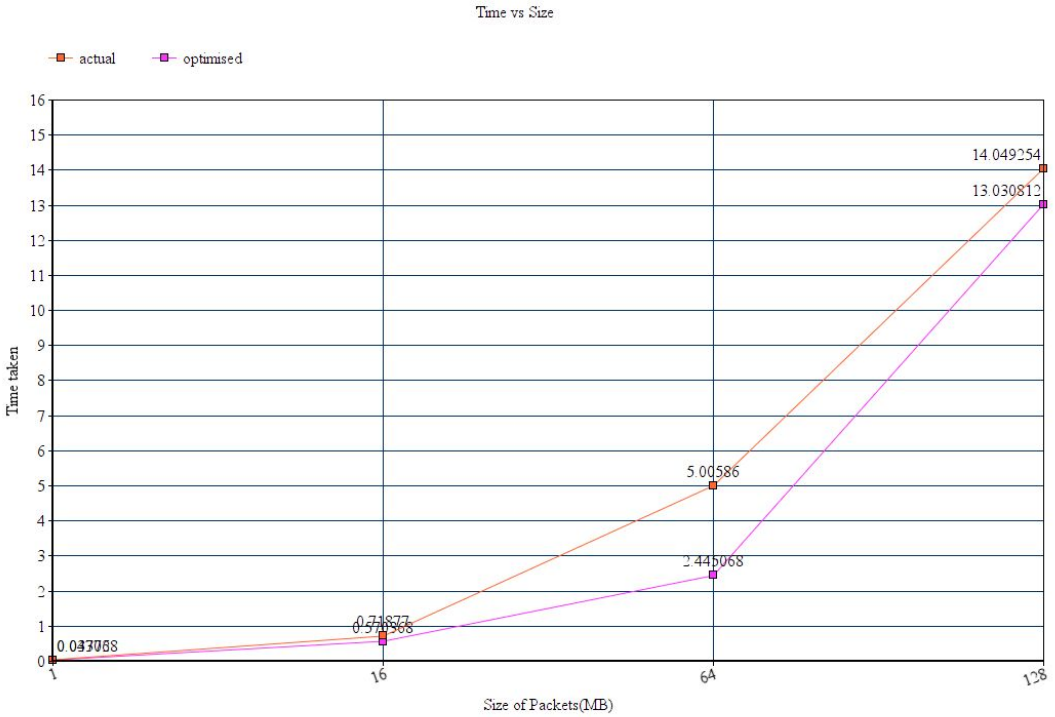
Topology Aware Collectives

Sanket Bodele (18111059)
Rudranil Bhowmik (18111055)

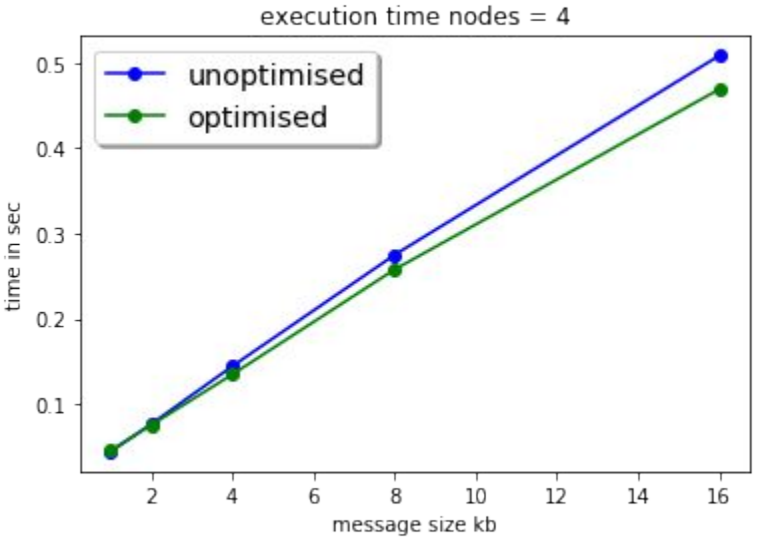
Previous Work

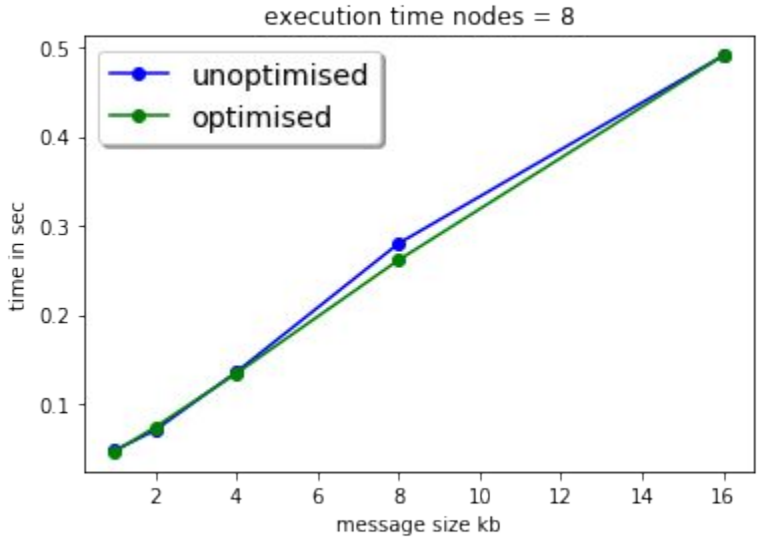
- Implemented topology aware **ring algorithm** for all gather.
- **Compared** it with inbuilt non topology aware ring algorithm.
- **Analyzed and verified** the results in the paper and speed up was within the mentioned range(15-30 %)



Results for Ring Communication pattern

- Results for 4 and 8 nodes for various sized data.





MPI_ALLGATHER

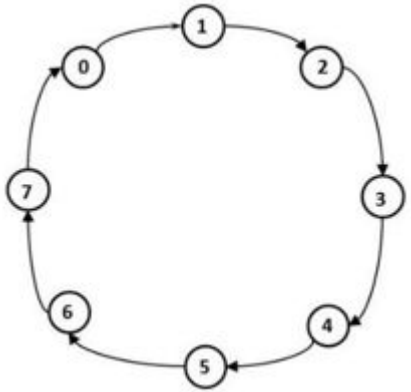
- MPI_Allgather uses different techniques to gather and send data in between nodes.
- It uses a **ring algorithm** for data size < 81920 bytes.
- It uses a **recursive doubling algorithm** for data size <524288 bytes.

```
178                                         recvtype, comm_ptr, errflag);
179             break;
180         case MPIR_CVAR_ALLGATHER_INTRA_ALGORITHM_recursive_doubling:
181             mpi_errno =
182                 MPIR_Allgather_intra_recursive_doubling(sendbuf, sendcount, sendtype, recvbuf,
183                                                         recvcount, recvtype, comm_ptr, errflag);
184             break;
185         case MPIR_CVAR_ALLGATHER_INTRA_ALGORITHM_ring:
186             mpi_errno =
187                 MPIR_Allgather_intra_ring(sendbuf, sendcount, sendtype, recvbuf, recvcount,
188                                           recvtype, comm_ptr, errflag);
189             break;
190         case MPIR_CVAR_ALLGATHER_INTRA_ALGORITHM_nb:
191             mpi_errno =
192                 MPIR_Allgather_allcomm_nb(sendbuf, sendcount, sendtype, recvbuf, recvcount,
193                                           recvtype, comm_ptr, errflag);
194             break;
```

Ring Communication Pattern

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 **reference core**
3. Repeat 2 until complete



Recursive Doubling Communication Pattern

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

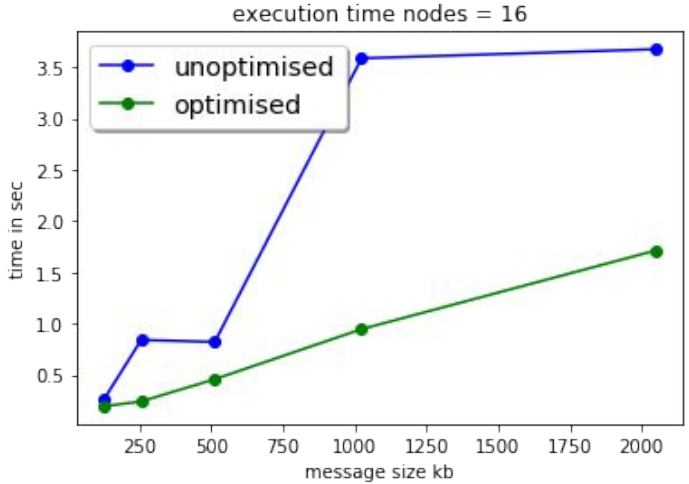
1. Fix rank 0 on its current core and make it the reference rank
2. Starting from the last stage i/2.
    a. While there exists process to map
        i. If i/2 is mapped then i=i/2
    b. new rank = **reference rank (XOR) i**
    c. Find a target core closest to reference core.
    d. Map new process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

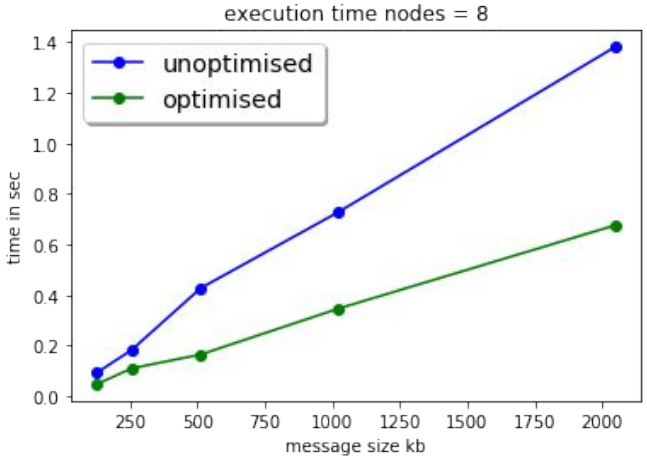
Results for Recursive Doubling algorithm

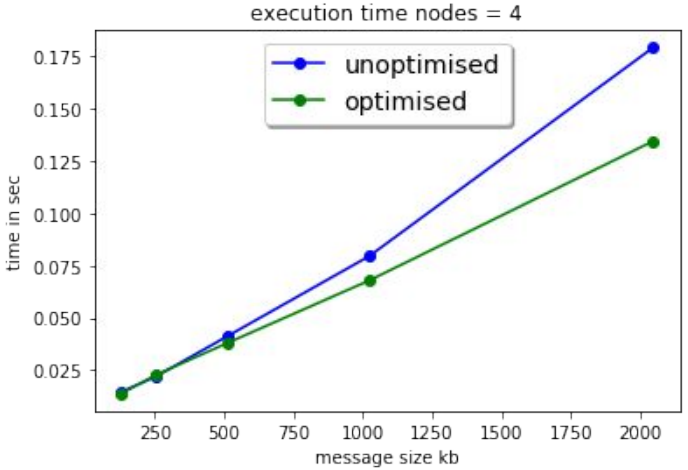
- Recursive Doubling algorithm converged better than the ring algorithm in our testing.
- We got more than 30% speedup for larger data size.
- It is due to the described mapping heuristics for Recursive Doubling was able to map more communicating processes as close as possible.

Results for Recursive doubling

- Results for 4 8 and 16 nodes for various sized data.







Discussion

- The results are produced by running our optimized algorithm multiple times(atleast 100).
- This gives a fairly uniform output that MPI-Allgather algorithm performs better under our mapping conditions.
- We ran optimized and unoptmized versions of these algorithms on cse cluster.

Conclusion

- Here we successfully **exploited rank reordering** to make MPI-Allgather topology aware .
- Experimental results showed that topology-aware rank reordering with our heuristics can provide **considerable** performance improvements
- Recursive doubling was able to scale well than the Ring algorithm with proposed Mapping heuristics.

Future Work

- We can also try various topology aware mapping heuristics for other MPI function such as MPI_Bcast.
- We can also try to test the current results on more complicated intra node topology systems with larger number of cores per node.
- Furthermore, we can use machine learning in which runtime component is used to decide whether the new mapping heuristics should be used or not based upon the load on the system.