# Algorithm Selection of MPI Collectives Considering System Utilization

Majid Salimi Beni[1], Sascha Hunold[2], and Biagio Cosenza[1]

[1] Department of Computer Science, University of Salerno, Salerno, Italy
[2] Faculty of Informatics, TU Wien, Vienna, Austria

**Abstract.** MPI collective communications play an important role in coordinating and exchanging data among parallel processes in high performance computing. Various algorithms exist for implementing MPI collectives, each of which exhibits different characteristics, such as message overhead, latency, and scalability, which can significantly impact overall system performance. Therefore, choosing the suitable algorithm for each collective operation is crucial to achieve optimal performance. In this paper, we present our experience with MPI collectives algorithm selection on a large-scale supercomputer and highlight the impact of network traffic and system workload as well as other previously-investigated parameters such as message size, communicator size, and network topology. Our analysis shows that network traffic and system workload can make the performance of MPI collectives highly variable and, accordingly, impact the algorithm selection strategy.

**Keywords:** High Performance Computing · MPI · Collectives · Broadcast · Algorithm Selection · Tuning.

## 1 Introduction

MPI (Message Passing Interface) is a widely-used standard for programming parallel and high performance computing (HPC) systems that allows efficient communication among distributed processes over the network [2]. MPI collective communication operations are fundamental building blocks for developing parallel applications, and a big share of HPC applications' runtime is spent while performing collective communications [13].

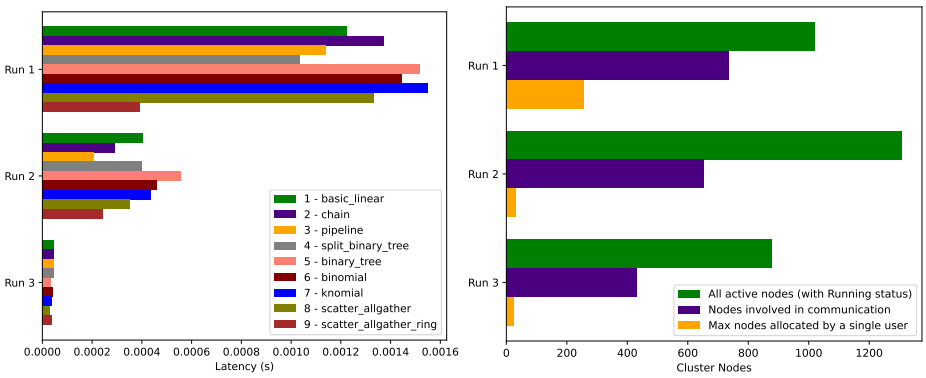
In recent MPI implementations, several algorithms have been implemented for each collective operation, each of which owns distinct internal characteristics such as communication costs and scalability attributes. An efficient algorithm selection for MPI collectives is crucial in achieving optimal performance and significantly impacts the overall scalability, communication overhead, and resource utilization in a parallel application. Hence, researchers have explored different parameters that impact the algorithm selection [8, 11]. Considering the network as a shared resource among users in supercomputers, network elements are subject to congestion; accordingly, different collective algorithms may perform differently under different network conditions since they have different strategies

for data transmission and message chunking. Therefore, network traffic is a determinative factor affecting the decision-making process for the best algorithm.

In this paper, we first analyze the behavior of different implementations of an MPI collective, Broadcast, on a large-scale cluster; and show the impact of network traffic on each algorithm's performance. Then, by monitoring the network, we propose a workload-aware algorithm selection method for MPI collectives that considers network conditions as well as data and communicator size. In the rest of the paper, we perform a preliminary analysis in Sect. 2. Related work is presented in Sect. 3; Sect. 4 describes the proposed algorithm selection method, and a summary of the current status and future directions are in Sect. 5.

## 2 Motivation

Large-scale clusters are usually utilized by many users at the same time, and several resources, including the network, are shared among them. Sharing the network with other users can degrade the communication performance of our job and make it variable, especially in communication-intensive applications. It is shown that collective operations may behave very differently under heavy network traffic, and their performance can be several times variable from run to run [1, 12].




(a) (b)

Fig. 1: (a) Latency of running all algorithms of `MPI_Bcast` three times with 512 processes on 16 nodes, allocated across different islands of Marconi100, with 10KB message size. (b) Cluster utilization data for the three runs extracted from SLURM's job queue.

The collective communications used in the paper are taken from the ReproMPI benchmark [7] and the MPI implementation is OpenMPI 4.1. We conducted our analyses on Marconi100 compute cluster [10] at the CINECA supercomputing center, which characterizes 980 nodes divided into 4 groups called islands. Each node comes with two 16 cores of IBM POWER9 AC922.

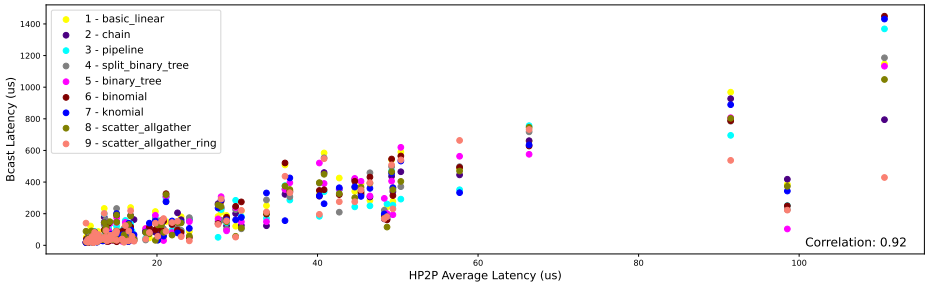
Figure 1a shows the latency of running different algorithms of `MPI_Bcast()` three times on three different days. All the algorithms in each run are performed at once, one after the other. Although the message size and process count are the same in this figure for the three runs, their behavior changes. First, there is high variability between runs: in run 3, algorithm 1 is 25X faster than run 1. Second, the best algorithm is not the same for the three runs: algorithms 4, 3, and 8 are the best for the three runs, consecutively. Figure 1b presents the job scheduler's monitored data before the execution of the benchmark. As shown, when there are users that allocated many nodes for their jobs or more nodes are allocated by single jobs (more nodes are involved in the communication), the latency of our benchmark is higher, and the best algorithm changes.

From figure 1, it is clear that another player is impacting the algorithm selection besides message size and the number of processes. Since the node allocation strategy has been the same for the three runs and nodes are allocated on different islands of the cluster in all runs, the only changing factor is the network condition and cluster utilization. In this paper, we quantify the network traffic between the currently allocated nodes and show which algorithms perform better under different network traffic.

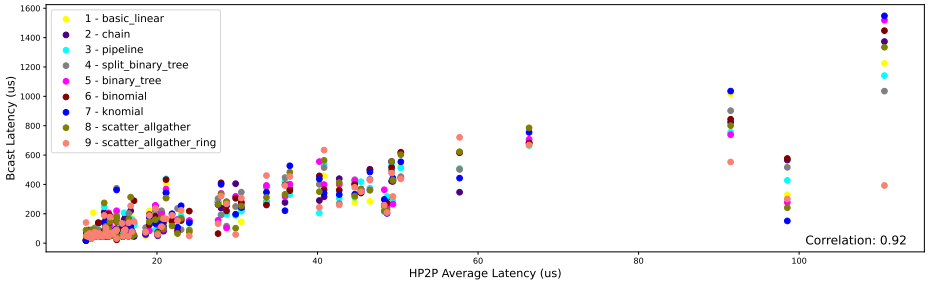
## 3 Related Work and Background

Since the collective algorithm selection can highly improve the performance of MPI collectives and impacts the communication-intensive applications, this problem has been investigated and several algorithms are tied with some MPI implementations such as OpenMPI and IntelMPI. OpenMPI uses a hard-coded decision tree and chooses the best algorithm based on the communicator and message size. IntelMPI, on the other hand, performs an exhaustive search to find the best algorithm for a given set of message sizes and nodes. Other than MPI implementations, recent work has investigated algorithm selection for MPI collectives. Researchers have focused on online [8], offline [3], machine learning [14, 5, 15], and modelling-based [11, 6] approaches to facilitate the algorithm selection process.

Apart from the methodology of choosing the best algorithm, current works consider several parameters that can impact this process. These factors include message size, process count, network topology, and available hardware resources [9]. The message size determines whether a certain algorithm, such as a binomial tree or a ring-based algorithm, is more suitable for small or large messages, respectively. The process count affects the algorithm's scalability, ensuring efficient communication considering the number of processes involved. Network topology helps determine whether a hierarchical algorithm or a non-blocking algorithm would be more suitable for minimizing communication overhead. Finally, considering the available hardware resources, such as the presence of specialized communication hardware or network features, can guide the selection of algorithms optimized for specific architectures.




(a) 100 B




(b) 10 KB

Fig. 2: The correlation between latencies of HP2P and Bcast

While running MPI jobs on supercomputers, selecting the best algorithm while ignoring the network traffic can lead to an improper algorithm selection since the algorithms may perform differently while operating on a network with changing background traffic. Despite the above-mentioned parameters' impact on collectives algorithm selection, to the best of our knowledge, network traffic's effect is not investigated by the literature.

## 4 Workload-Aware Algorithm Selection

In this section, we provide our preliminary results about the impact of network traffic on algorithm selection. In order to show the impact of network traffic on the performance variability of MPI collective algorithms, we use HP2P benchmark [4] that measures the peer-to-peer latency and bandwidth between the pairs by exchanging asynchronous messages. In our experiments, and before running the main benchmark, we run HP2P for 1000 iterations with 4KB and monitor the network status.

Figure 2 shows the latency of different algorithms of Broadcast on 512 processes for two different message sizes correlated with HP2P's latency. Each set of algorithms is executed once, one after the other, and HP2P measures the latency before running each set. The figure includes 100 series of runs executed on different days and hours of the days, three runs per day. For the two sizes,

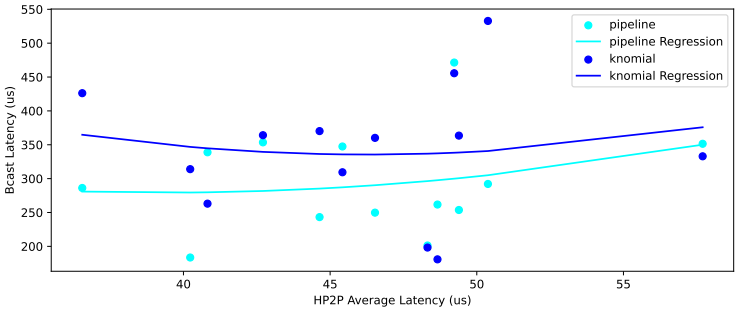



Fig. 3: The performance distribution of Pipeline and Knomial between the range of 35 to 60 us.

the latencies of HP2P and Broadcast are highly correlated (92 percent), and it is possible to accurately estimate the execution time of the main benchmark by quickly checking the network status before its execution. On top of that, when the network is congested, the different algorithms perform differently and deliver various performances. Indicating that first, the network traffic is impacting algorithms performance, and second, the right algorithm selection in higher network traffic can highly improve the communication performance.

To highlight network traffic's impact on the algorithms, in figure 3, we focus on a subset of runs between 35-60 (average cases) of figure 2a. As shown for this subset, on average, Pipeline has shown a higher performance (around 15% on average) than Knomial (to which the Default algorithm is mapped). However, for HP2P average latency smaller than 35, the average-best algorithm is a different one. Therefore, for each range of network traffic, different algorithms have diverse behavior, and the best algorithm may change.

## 5 Summary

In this work, we highlighted the impact of network traffic on the algorithms. We proposed a workload-aware algorithm selection method for MPI collectives that monitors the network traffic and chooses the best algorithm according to the network traffic between the allocated nodes. For future work, we plan to carry out the following activities:

- Collecting data from the job scheduler as well as other microbenchmarks to better characterize the cluster's workload and network utilization.
- Combining the statistical with regression and machine learning methods to provide a more accurate algorithm selector for MPI collectives and then automate the selection process.

## References


[1] Majid Salimi Beni and Biagio Cosenza. "An Analysis of Performance Variability on Dragonfly+ topology". In: *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. 2022, pp. 500–501.

[2] Sudheer Chunduri, Scott Parker, Pavan Balaji, Kevin Harms, and Kalyan Kumaran. "Characterization of mpi usage on a production supercomputer". In: *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2018, pp. 386–400.

[3] Ahmad Faraj, Xin Yuan, and David Lowenthal. "STAR-MPI: self tuned adaptive routines for MPI collective operations". In: *Proceedings of the 20th annual international conference on Supercomputing*. 2006, pp. 199–208.

[4] *GitHub - cea-hpc/hp2p: Heavy Peer To Peer: a MPI based benchmark for network diagnostic*. `https://github.com/cea-hpc/hp2p`. [Accessed 15-May-2023].

[5] Sascha Hunold, Abhinav Bhatele, George Bosilca, and Peter Knees. "Predicting MPI collective communication performance using machine learning". In: *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. 2020, pp. 259–269.

[6] Sascha Hunold and Alexandra Carpen-Amarie. "Autotuning MPI collectives using performance guidelines". In: *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*. 2018, pp. 64–74.

[7] Sascha Hunold and Alexandra Carpen-Amarie. "Reproducible MPI benchmarking is still not as easy as you think". In: *IEEE Transactions on Parallel and Distributed Systems* 27.12 (2016), pp. 3617–3630.

[8] Sascha Hunold and Sebastian Steiner. "OMPICollTune: Autotuning MPI Collectives by Incremental Online Learning". In: *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE. 2022, pp. 123–128.

[9] Wilton Jaciel Loch and Guilherme Piêgas Koslovski. "Sparbit: Towards to a Logarithmic-Cost and Data Locality-Aware MPI Allgather Algorithm". In: *Journal of Grid Computing* 21.2 (2023), p. 18.

[10] *Marconi100, The new accelerated system*. [Accessed 15-May-2023]. URL: `https://www.hpc.cineca.it/hardware/marconi100`.

[11] Emin Nuriyev, Juan-Antonio Rico-Gallego, and Alexey Lastovetsky. "Model-based selection of optimal MPI broadcast algorithms for multi-core clusters". In: *Journal of Parallel and Distributed Computing* 165 (2022), pp. 1–16.

[12] Majid Salimi Beni and Biagio Cosenza. "An analysis of long-tailed network latency distribution and background traffic on dragonfly+". In: *Benchmarking, Measuring, and Optimizing*. LNCS, Springer, 2022.

[13] Majid Salimi Beni, Luigi Crisci, and Biagio Cosenza. "EMPI: Enhanced Message Passing Interface in Modern C++". In: *2023 23rd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE. 2023.

[14] Michael Wilkins, Yanfei Guo, Rajeev Thakur, Peter Dinda, and Nikos Hardavellas. "ACCLAiM: Advancing the Practicality of MPI Collective Communication Autotuning Using Machine Learning". In: *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. 2022, pp. 161–171.

[15] Michael Wilkins, Yanfei Guo, Rajeev Thakur, Nikos Hardavellas, Peter Dinda, and Min Si. "A FACT-based approach: Making machine learning collective autotuning feasible on exascale systems". In: *2021 Workshop on Exascale MPI (ExaMPI)*. IEEE. 2021, pp. 36–45.