A New Model-Based Approach to Performance Comparison of MPI Collective Algorithms

Emin Nuriyev and Alexey Lastovetsky(✉)

School of Computer Science, University College Dublin, Dublin, Ireland
emin.nuriyev@ucdconnect.ie, alexey.lastovetsky@ucd.ie

Abstract. The performance of collective operations has been a critical issue since the advent of Message Passing Interface (MPI). Many algorithms have been proposed for each MPI collective operation but none of them proved optimal in all situations. Different algorithms demonstrate superior performance depending on the platform, the message size, the number of processes, etc. MPI implementations perform the selection of the collective algorithm empirically, executing a simple runtime decision function. While efficient, this approach does not guarantee the optimal selection. As a more accurate but equally efficient alternative, the use of analytical performance models of collective algorithms for the selection process was proposed and studied. Unfortunately, the previous attempts in this direction have not been successful.

We revisit the analytical model-based approach and propose two innovations that significantly improve the selective accuracy of analytical models: (1) We derive analytical models from the code implementing the algorithms rather than from their high-level mathematical definitions. This results in more detailed models. (2) We estimate model parameters separately for each collective algorithm and include the execution of this algorithm in the corresponding communication experiment.

We experimentally demonstrate the accuracy and efficiency of our approach using Open MPI broadcast algorithms and two different Grid'5000 clusters.

Keywords: Message Passing · Collective communication algorithms · Communication performance modelling · MPI

1 Introduction

The message passing interface (MPI) [1] is the de-facto standard, which provides a reliable and portable environment for developing high-performance parallel applications on different platforms. The study [2] shows that collective operations consume more than eighty percent of the total communication time of a typical

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474.

V. Malyshkin (Ed.): PaCT 2021, LNCS 12942, pp. 11–25, 2021.
https://doi.org/10.1007/978-3-030-86359-3_2

MPI application. Therefore, a significant amount of research has been invested into optimisation of MPI collectives. Those researches have resulted in a large number of algorithms, each of which comes up optimal for specific message sizes, platforms, numbers of processes, and so forth. Mainstream MPI libraries [3,4] provide multiple collective algorithms for each collective routine.

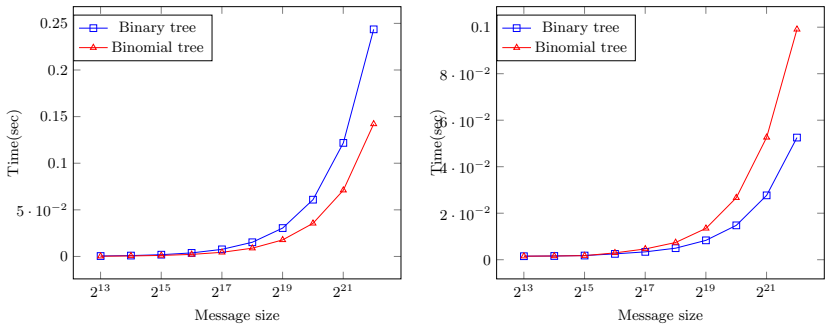

Fig. 1. Performance estimation of the binary and binomial tree broadcast algorithms by the traditional analytical models in comparison with experimental curves. The experiments involve ninety processes (P = 90). (a) Estimation by the existing analytical models. (b) Experimental performance curves.

There are two ways how this selection can be made in the MPI program. The first one, MPI_T interface [1], is provided by the MPI standard and allows the MPI programmer to select the collective algorithm explicitly from the list of available algorithms for each collective call at run-time. It does not solve the problem of optimal selection delegating its solution to the programmer. The second one is transparent to the MPI programmer and provided by MPI implementations. It uses a simple *decision function* in each collective routine, which is used to select the algorithm at runtime. The decision function is empirically derived from extensive testing on the dedicated system. For example, for each collective operation, both MPICH and Open MPI use a simple decision routine selecting the algorithm based on the message size and number of processes [5–7]. The main advantage of this solution is its efficiency. The algorithm selection is very fast and does not affect the performance of the program. The main disadvantage of the existing decision functions is that they do not guarantee the optimal selection in all situations.

As an alternative approach, the use of analytical performance models of collective algorithms for the selection process has been proposed and studied [8]. Unfortunately, the analytical performance models proposed in this work could not reach the level of accuracy sufficient for selection of the optimal algorithm (Fig. 1).

In this paper, we revisit the model-based approach and propose a number of innovations that significantly improve the selective accuracy of analytical models to the extent that allows them to be used for accurate selection of optimal collective algorithms.

The main contributions of this paper can be summarized as follows:

- We propose and implement a new analytical performance modelling approach for MPI collective algorithms, which derives the models from the code implementing the algorithms.
- We propose and implement a novel approach to estimation of the parameters of analytical performance models of MPI collective algorithms, which estimates the parameters separately for each algorithm and includes the modelled collective algorithm in the communication experiment, which is used to estimate the model parameters.
- We experimentally validate the proposed approach to selection of optimal collective algorithms on two different clusters of the Grid'5000 platform.

The rest of the paper is structured as follows. Section 2 reviews the existing approaches to performance modelling and algorithm selection problems. Section 3 describes our approach to construction of analytical performance models of MPI collective algorithms by deriving them from the MPI implementation. Section 4 presents our method to measure analytical model parameters. Section 5 presents experimental validation of the proposed approach. Section 6 concludes the paper with a discussion of the results and an outline of the future work.

2 Related Work

In order to select the optimal algorithm for a given collective operation, we have to be able to accurately compare the performance of the available algorithms. Analytical performance models are one of the efficient ways to express and compare the performance of collective algorithms. In this section, we overview the state-of-the-art in analytical performance modelling, measurement of model parameters and selection of the optimal collective algorithms.

2.1 Analytical Performance Models of MPI Collective Algorithms

Thakur et al. [5] propose analytical performance models of several collective algorithms using the Hockney model [9]. Chan et al. [10] build analytical performance models of *Minimum-spanning tree* algorithms and *Bucket* algorithms for MPI_Bcast, MPI_Reduce, MPI_Scatter, MPI_Gather, MPI_Allgather collectives and later extend this work for multidimensional mesh architecture in [11]. Neither of the studies listed above uses the build models for selecting the optimal collective algorithms. Pjevsivac-Grbovic et al. [8] study selection of optimal collective algorithms using analytical performance models for *barrier*, *broadcast*, *reduce* and *alltoall* collective operations. The models are built up with the traditional approach using high-level mathematical definitions of the collective algorithms.

In order to predict the cost of a collective algorithm by analytical formula, model parameters are measured using point-to-point communication experiments. After experimental validation of their modelling approach, the authors conclude that the proposed models are not accurate enough for selection of optimal algorithms.

2.2 Measurement of Model Parameters

Hockney [9] presents a measurement method to find the α and β parameters of the Hockney model. The set of communication experiments consists of point-to-point round-trips. Culler et al. [12] propose a method of measurement of parameters of the LogP model, namely, L, the upper bound on the latency, o_s, the overhead of processor involving sending a message, o_r, the overhead of processor involving receiving a message, and g, the gap between consecutive message transmission. Kielmann et al. [13] propose a method of measurement of parameters of the PLogP (Parametrized LogP) model. PLogP defines its model parameters, except for latency L, as functions of message size. All approaches listed above to measure model parameters are based on point-to-point communication experiments.

From this overview, we can conclude that the state-of-the-art analytical performance models are built using only high-level mathematical definition of the algorithms, and methods for measurement of parameters of communication performance models are all based on *point-to-point communication experiments*. The only exception from this rule is a method for measurement of parameters of the LMO heterogeneous communication model [14–16]. LMO is a communication model of heterogeneous clusters, and the total number of its parameters is significantly larger than the maximum number of independent point-to-point communication experiments that can be designed to derive a system of independent linear equations with the model parameters as unknowns. To address this problem and obtain the sufficient number of independent linear equations involving model parameters, the method additionally introduces simple collective communication experiments, each using three processors and consisting of a one-to-two communication operation (scatter) followed by a two-to-one communication operation (gather). This method however is not designed to improve the accuracy of predictive analytical models of communication algorithms.

In this work, we propose to use *collective communication experiments* in the measurement method in order to improve the predictive accuracy of analytical models of collective algorithms. A more detailed survey in analytical performance modelling and estimation of the model parameters can be found in [17].

2.3 Selection of Collective Algorithms Using Machine Learning Algorithms

Machine learning (ML) techniques have been also tried to solve the problem of selection of optimal MPI algorithms.

In [18], applicability of the quadtree encoding method to this problem is studied. The goal of this work is to select the best performing algorithm and segment

size for a particular collective on a particular platform. The experimental results show that the decision function performs poorly on unseen data. Applicability of the C4.5 algorithm to the MPI collective selection problem is explored in [19]. The C4.5 algorithm [20] is a decision tree classifier, which is employed to generate a decision function, based on a detailed profiling data of MPI collectives. While the accuracy of the decision function built by the C4.5 classification algorithm is higher than that of the decision function built by quadtree encoding algorithm, still, the performance penalty is higher than 50%.

Most recently Hunold et al. [21] studied the applicability of six different ML algorithms including Random Forests, Neural Networks, Linear Regressions, XGBoost, K-nearest Neighbor, and generalized additive models (GAM) for selection of optimal MPI collective algorithms. First, it is very expansive and difficult to build a regression model even for a relatively small cluster. There is no clear guidance how to do it to achieve better results. Second, even the best regression models do not accurately predict the fastest collective algorithm in most of the reported cases. Moreover, in many cases the selected algorithm performs worse than the default algorithm, that is, the one selected by a simple native decision function.

To the best of the authors' knowledge, the works outlined in this subsection are the only research done in MPI collective algorithm selection using ML algorithms. The results show that the selection of the optimal algorithm without any information about the semantics of the algorithm yields inaccurate results. While the ML-based methods treat a collective algorithm as a black box, we derive its performance model from the implementation code and estimate the model parameters using statistical techniques. The limitations of the application of the statistical techniques (AI/ML) to collective performance modelling and selection problem can be found in a detailed survey [22].

3 Implementation-Derived Analytical Models of Collective Algorithms

As stated in Sect. 1, we propose a new approach to analytical performance modelling of collective algorithms. While the traditional approach only takes into account high-level mathematical definitions of the algorithms, we derive our models from their implementation. This way, our models take into account important details of their execution having a significant impact on their performance. Open MPI uses six tree-based broadcast algorithms to implement MPI_Bcast including Linear tree algorithm, Chain tree algorithm, Binary tree algorithm, Split binary tree algorithm, K-Chain tree algorithm and Binomial tree algorithm. Because of the limited space, we present our analytical modelling approach by applying it to only binomial tree broadcast algorithm implemented in Open MPI.

To model point-to-point communications, we use the Hockney model, which estimates the time $T_{p2p}(m)$ of sending a message of size m between two processes as $T_{p2p}(m) = \alpha + \beta \cdot m$, where α and β are the latency and the reciprocal

bandwidth respectively. For segmented collective algorithms, we assume that $m = n_s \cdot m_s$, where n_s and m_s are the number of segments and the segment size respectively. We assume that each algorithm involves P processes ranked from 0 to $P-1$.

3.1 Binomial Tree Broadcast Algorithm

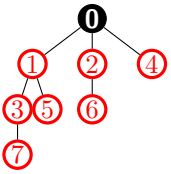

Fig. 2. Balanced binomial tree

In Open MPI, the binomial tree broadcast algorithm is segmentation-based and implemented as a combination of linear tree broadcast algorithms using non-blocking *send* and *receive* operations. The height of the binomial tree is the order of the tree, $H = \lfloor \log_2 P \rfloor$ (Fig. 2).

Figure 3 shows the stages of execution of the binomial tree broadcast algorithm. Each stage consists of parallel execution of a number of linear broadcast algorithms using non-blocking communication. The linear broadcast algorithms running in parallel have a different number of children. Therefore, the execution time of each stage will be equal to the execution time of the linear broadcast algorithm with the maximum number of children. The execution time of the whole binomial broadcast algorithm will be equal to the sum of the execution times of these stages.

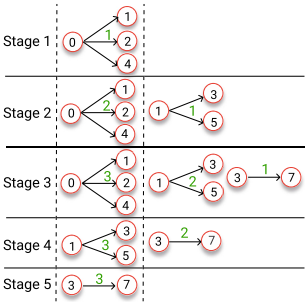

Fig. 3. Execution stages of the binomial tree broadcast algorithm, employing the non-blocking linear broadcast ($P = 8$, $n_s = 3$). Nodes are labelled by the process ranks. Each arrow represents transmission of a segment. The number over the arrow gives the index of the broadcast segment.

In the non-blocking linear broadcast algorithm, $P-1$ non-blocking *send*s will run on the *root* concurrently. Therefore, the execution time of the linear broadcast algorithm using non-blocking point-to-point communications and buffered mode, $T^{nonblock}_{linear\_bcast}(P, m)$, can be bounded as follows:

$$T_{p2p}(m) \leq T^{nonblock}_{linear\_bcast}(P, m) \leq (P-1) \cdot T_{p2p}(m). \quad (1)$$

We will approximate $T^{nonblock}_{linear\_bcast}(P,m)$ as

$$T^{nonblock}_{linear\_bcast}(P,m) = \gamma(P,m)\cdot(\alpha + m\cdot\beta), \tag{2}$$

where

$$\gamma(P,m) = \frac{T^{nonblock}_{linear\_bcast}(P,m)}{T_{p2p}(m)}. \tag{3}$$

In Open MPI, the binomial tree broadcast algorithm employs the balanced binomial tree virtual topology (Fig. 2). Therefore, the number of stages in the binomial broadcast algorithm can be calculated as

$$N_{steps} = \lfloor log_2 P \rfloor + n_s - 1. \tag{4}$$

Thus, the time to complete the binomial tree broadcast algorithm can be estimated as follows:

$$T_{binomial\_bcast}(P,m,n_s) = \sum_{i=1}^{\lfloor log_2 P\rfloor + n_s - 1} \max_{1\le j\le \min(\lfloor log_2 P\rfloor, n_s)} T^{nonblock}_{linear\_bcast}(P_{ij}, \frac{m}{n_s}), \tag{5}$$

where P_{ij} denotes the number of nodes in the j-th linear tree of the i-th stage.

Using the property of the binomial tree and Formula 2, we have

$$T_{binomial\_bcast}(P,m,n_s) = (n_s\cdot\gamma(\lceil \log_2 P\rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P\rfloor - 1} \gamma(\lceil \log_2 P\rceil - i + 1) - 1)\cdot(\alpha + \frac{m}{n_s}\cdot\beta). \tag{6}$$

4 Estimation of Model Parameters

4.1 Estimation of $\gamma(P)$

The model parameter $\gamma(P)$ appears in the formula estimating the execution time of the linear tree broadcast algorithm with non-blocking communication, which is only used for broadcasting of a segment in the tree-based segmented broadcast algorithms. Thus, in the context of Open MPI, the linear tree broadcast algorithm with non-blocking communication will always broadcast a message of size m_s to a relatively small number of processes.

According to Formula 3,

$$\gamma(P) = \frac{T^{nonblock}_{linear\_bcast}(P,m_s)}{T_{p2p}(m_s)} = \frac{T^{nonblock}_{linear\_bcast}(P,m_s)}{T^{nonblock}_{linear\_bcast}(2,m_s)}.$$

Therefore, in order to estimate $\gamma(P)$ for a given range of the number of processes, $P \in \{2,...,P_{max}\}$, we need a method for estimation of $T^{nonblock}_{linear\_bcast}(P,m_s)$. We use the following method:

– For each $2 \leq q \leq P_{max}$, we measure on the root the execution time $T_1(P, N)$ of N successive calls to the *linear tree with non-blocking communication* broadcast routine separated by barriers. The routine broadcasts a message of size m_s.
– We estimate $T^{nonblock}_{linear\_bcast}(P, m_s)$ as $T_2(P) = \frac{T_1(P,N)}{N}$.

The experimentally obtained discrete function $\frac{T_2(P)}{T_2(2)}$ is used as a platform-specific but algorithm-independent estimation of $\gamma(P)$.

From our experiments, we observed that the discrete estimation of $\gamma(P)$ is near linear. Therefore, as an alternative for platforms with very large numbers of processors, we can build by linear regression a linear approximation of the discrete function $\frac{T_2(P)}{T_2(2)}$, obtained for a representative subset of the full range of P, and use this linear approximation as an analytical estimation of $\gamma(P)$.

$$
\begin{cases}
(n_{s_1} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) \cdot (\alpha + \frac{m_1}{n_{s_1}} \cdot \beta) + (P-1) \cdot (\alpha + m_{g_1} \cdot \beta) = T_1 \\
(n_{s_2} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) \cdot (\alpha + \frac{m_2}{n_{s_2}} \cdot \beta) + (P-1) \cdot (\alpha + m_{g_2} \cdot \beta) = T_2 \\
\dots \\
(n_{s_M} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) \cdot (\alpha + \frac{m_M}{n_{s_M}} \cdot \beta) + (P-1) \cdot (\alpha + m_{g_M} \cdot \beta) = T_M
\end{cases}
$$

$$\Downarrow$$

$$
\begin{cases}
\alpha + \beta \cdot \frac{(n_{s_1} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) \cdot \frac{m_1}{n_{s1}} + (P-1) \cdot m_{g_1}}{(n_{s_1} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) + P - 1} = \frac{T_1}{\left((n_{s_1} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) + P - 1\right)} \\
\alpha + \beta \cdot \frac{(n_{s_2} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) \cdot \frac{m_2}{n_{s2}} + (P-1) \cdot m_{g_2}}{(n_{s_2} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) + P - 1} = \frac{T_2}{\left((n_{s_2} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) + P - 1\right)} \\
\dots \\
\alpha + \beta \cdot \frac{(n_{s_M} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) \cdot \frac{m_M}{n_{s_M}} + (P-1) \cdot m_{g_M}}{(n_{s_M} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) + P - 1} = \frac{T_M}{\left((n_{s_M} \cdot \gamma(\lceil \log_2 P \rceil + 1) + \sum_{i=1}^{\lfloor \log_2 P \rfloor - 1} \gamma(\lceil \log_2 P \rceil - i + 1) - 1) + P - 1\right)}
\end{cases}
$$

Fig. 4. A system of M non-linear equations with α, β, $\gamma(\lceil \log_2 P \rceil + 1)$ and $\gamma(\lceil \log_2 P \rceil - i + 1)$ as unknowns, derived from M communication experiments, each consisting of the execution of the binomial tree broadcast algorithm, broadcasting a message of size m_i $(i = 1, ..., M)$ from the root to the remaining $P-1$ processes, followed by the linear gather algorithm without synchronisation, gathering messages of size $m_{g_i} (m_{g_i} \neq m_s)$ on the root. The execution times, T_i, of these experiments are measured on the root. Given $\gamma(\lceil \log_2 P \rceil + 1)$ and $\gamma(\lceil \log_2 P \rceil - i + 1)$ are evaluated separately, the system becomes a system of M linear equations with α and β as unknowns.

4.2 Estimation of Algorithm Specific α and β

To estimate the model parameters α and β for a given collective algorithm, we design a communication experiment, which starts and finishes on the root (in

order to accurately measure its execution time using the root clock), and involves the execution of the modelled collective algorithm so that the total time of the experiment would be dominated by the time of its execution.

For example, for all broadcast algorithms, the communication experiment consists of a broadcast of a message of size m (where m is a multiple of segment size m_s), using the modelled broadcast algorithm, followed by a *linear-without-synchronisation* gather algorithm, gathering messages of size $m_g (m_g \neq m_s)$ on the root. The execution time of this experiment on P nodes, $T_{bcast\_exp}(P, m)$, can be estimated as follows:

$$T_{bcast\_exp}(P, m) = T_{bcast\_alg}(P, m) + T_{linear\_gather}(P, m_g) \tag{7}$$

The execution time of the linear-without-synchronisation gather algorithm, gathering a message size of m_g on the root from $\mathrm{P}-1$ processes where $m_g \neq m_s$, is estimated as follows,

$$T_{linear\_gather}(P, m_g) = (P-1) \cdot (\alpha + m_g \cdot \beta) \tag{8}$$

Using Formula 6 and 8 for each combination of P and m this experiment will yield one linear equation with α and β as unknowns. By repeating this experiment with different p and m, we obtain a system of linear equations for α and β. Each equation in this system can be represented in the canonical form, $\alpha + \beta \times m_i = T_i \ (i = 1, ..., M)$. Finally, we use the least-square regression to find α and β, giving us the best linear approximation $\alpha + \beta \times m$ of the discrete function $f(m_i) = T_i \ (i = 1, ..., M)$.

Figure 4 shows a system of linear equations built for the binomial tree broadcast algorithm for our experimental platform. To build this system, we used the same P nodes in all experiments but varied the message size $m \in \{m_1, ..., m_M\}$ and $m_g \in \{m_{g_1}, ..., m_{g_M}\}$. With M different message sizes, we obtained a system of M equations. The number of nodes, P, was approximately equal to the half of the total number of nodes. We observed that the use of larger numbers of nodes in the experiments will not change the estimation of α and β.

5 Experimental Results and Analysis

This section presents experimental evaluation of the proposed approach to selection of optimal collective algorithms using Open MPI broadcast operation. In all experiments. We use the default Open MPI configuration (without any collective optimization tuning).

5.1 Experiment Setup

For experiments, we use Open MPI 3.1 running on a dedicated Grisou and Gros clusters of the Nancy site of the Grid'5000 infrastructure [23]. The Grisou cluster consists of 51 nodes each with 2 Intel Xeon E5-2630 v3 CPUs (8 cores/CPU), 128 GB RAM, 2x558 GB HDD, interconnected via 10 Gbps Ethernet. The Gros

cluster consists of 124 nodes each with Intel Xeon Gold 5220 (18 cores/CPU), 96 GB RAM, 894 GB SSD, interconnected via 2 × 25 Gb Ethernet.

To make sure that the experimental results are reliable, we follow a detailed methodology: 1) We make sure that the cluster is fully reserved and dedicated to our experiments. 2) For each data point in the execution time of collective algorithms, the sample mean is used, which is calculated by executing the application repeatedly until the sample mean lies in the 95% confidence interval and a precision of 0.025 (2.5%) has been achieved. We also check that the individual observations are independent and their population follows the normal distribution. For this purpose, MPIBlib [24] is used.

In our communication experiments, MPI programs use the one-process-per-CPU configuration, and the maximal total number of processes is equal to 90 on Grisou and 124 on Gros clusters. The message segment size, m_s, for segmented broadcast algorithms is set to 8 KB and is the same in all experiments. This segment size is commonly used for segmented broadcast algorithms in Open MPI. Selection of optimal segment size is out of the scope of this paper.

5.2 Experimental Estimation of Model Parameters

First of all, we would like to stress again that we estimate model parameters for each cluster separately.

Estimation of parameter $\gamma(p)$ for our experimental platforms follows the method presented in Sect. 4.1. With the maximal number of processes equal to 90 (Grisou) and 124 (Gros), the maximal number of children in the linear tree broadcast algorithm with non-blocking communication, used in the segmented Open MPI broadcast algorithms, will be equal to seven. Therefore, the number of processes in our communication experiments ranges from 2 to 7 for both clusters. By definition, $\gamma(2) = 1$. The estimated values of $\gamma(p)$ for p from 3 to 7 are given in Table 1.

Table 1. Estimated values of $\gamma(P)$ on Grisou and Gros clusters.

P	$\gamma(P)$	
	Grisou	Gros
3	1.114	1.084
4	1.219	1.17
5	1.283	1.254
6	1.451	1.339
7	1.540	1.424

After estimation of $\gamma(p)$, we conduct communication experiments to estimate algorithm-specific values of parameters α and β for six broadcast algorithms following the method described in Sect. 4.2. In these experiments we use 40 processes on Grisou and 124 on Gros. The message size, m, varies in the range from 8 KB to 4 MB in the broadcast experiments. We use 10 different sizes for broadcast algorithms, $\{m_i\}_{i=1}^{10}$, separated by a constant step in the logarithmic scale, $\log m_{i-1} - \log m_i = const$. Thus, for each collective algorithm, we obtain a system of 10 linear equations with α and β as unknowns. We use the Huber regressor [25] to find their values from the system.

The values of parameters α and β obtained this way can be found in Table 2. We can see that the values of α and β do vary depending on the collective algorithm, and the difference is more significant between algorithms implementing different collective operations. The results support our original hypothesis that

the average execution time of a point-to-point communication will very much depend on the context of the use of the point-to-point communications in the algorithm. Therefore, the estimated values of the α and β capture more than just sheer network characteristics. One interesting example is the Split-binary tree and Binary tree broadcast algorithms. They both use the same virtual topology, but the estimated time of a point-to-point communication, $\alpha + \beta \times m$, is smaller in the context of the Split-binary one. This can be explained by a higher level of parallelism of the Split-binary algorithm, where a significant part of point-to-point communications is performed in parallel by a large number of independent pairs of processes from the left and right subtrees.

Table 2. Estimated values of α and β for the Grisou and Gros clusters and Open MPI broadcast algorithms.

Collective algorithm	$\alpha(sec)$	$\beta\ (\frac{sec}{byte})$	Collective algorithm	$\alpha(sec)$	$\beta\ (\frac{sec}{byte})$
Broadcast			**Broadcast**		
Linear tree	2.2×10^{-12}	1.8×10^{-8}	Linear tree	1.4×10^{-12}	1.1×10^{-8}
K-Chain tree	5.7×10^{-13}	4.7×10^{-9}	K-Chain tree	5.4×10^{-13}	4.5×10^{-9}
Chain tree	6.1×10^{-13}	4.9×10^{-9}	Chain tree	4.7×10^{-12}	3.8×10^{-8}
Split-binary tree	3.7×10^{-13}	3.6×10^{-9}	Split-binary tree	5.5×10^{-13}	4.5×10^{-9}
Binary tree	5.8×10^{-13}	4.7×10^{-9}	Binary tree	5.8×10^{-13}	4.7×10^{-9}
Binomial tree	5.8×10^{-13}	4.8×10^{-9}	Binomial tree	1.2×10^{-13}	1.0×10^{-9}

5.3 Accuracy of Selection of Optimal Collective Algorithms Using the Constructed Analytical Performance Models

The constructed analytical performance models of the Open MPI broadcast collective algorithms are designed for the use in the MPI_Bcast routines for runtime selection of the optimal algorithm, depending on the number of processes and the message size. While the efficiency of the selection procedure is evident from the low complexity of the analytical formulas derived in Sect. 3, the experimental results on the accuracy are presented in this section.

Figure 5 shows the results of our experiments for MPI_Bcast. We present results of experiments with 50, 80 and 90 processes on Grisou, and 80, 100 and 124 on Gros. The message size, m, varies in the range from 8 KB to 4 MB in the broadcast experiments. We use 10 different sizes for broadcast algorithms, $\{m_i\}_{i=1}^{10}$, separated by a constant step in the logarithmic scale, $\log m_{i-1} - \log m_i = const$. The graphs show the execution time of the collective operation as a function of message size. Each data point on a blue line shows the performance of the algorithm selected by the Open MPI decision function for the given operation, number of processes and message size. Each point on a red line shows the performance of the algorithm selected by our decision function, which uses the constructed analytical models. Each point on a green line shows the performance of the best Open MPI algorithm for the given collective operation, number of processes and message size.

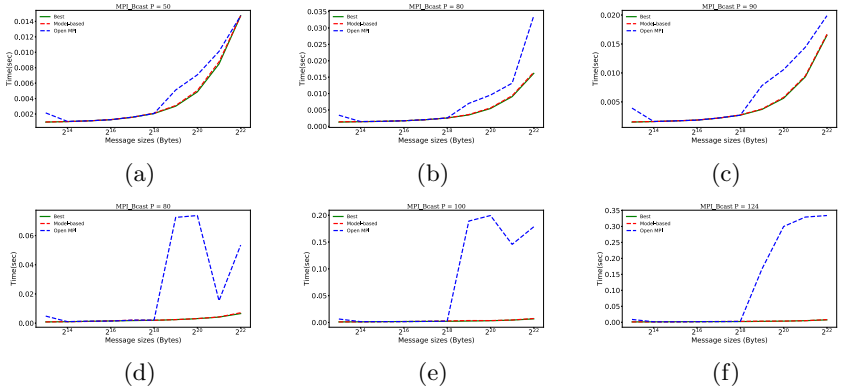

Fig. 5. Comparison of the selection accuracy of the Open MPI decision function and the proposed model-based method for MPI_Bcast. (a–c) and (d–f) present performance of collectives on Grisou and Gros clusters respectively.

Table 3. Comparison of the model-based and Open MPI selections with the best performing MPI_Bcast algorithm. For each selected algorithm, its performance degradation against the optimal one is given in braces.

P=90, MPI_Bcast, Grisou			
m (KB)	**Best**	**Model-based (%)**	**Open MPI (%)**
8	*binomial*	*binary* (3)	*split_binary* (160)
16	*binary*	*binary* (0)	*split_binary* (1)
32	*binary*	*binary* (0)	*split_binary* (0)
64	*split_binary*	*binary* (1)	*split_binary* (0)
128	*binary*	*binary* (0)	*split_binary* (1)
256	*split_binary*	*binary* (2)	*split_binary* (0)
512	*split_binary*	*binary* (2)	*chain* (111)
1024	*split_binary*	*binary* (3)	*chain* (88)
2048	*split_binary*	*binary* (2)	*chain* (55)
4096	*split_binary*	*binary* (1)	*chain* (20)

P=100, MPI_Bcast, Gros			
m (KB)	**Best**	**Model-based (%)**	**Open MPI (%)**
8	*binary*	*binomial* (3)	*split_binary* (549)
16	*binomial*	*binomial* (0)	*split_binary* (32)
32	*binomial*	*binomial* (0)	*split_binary* (3)
64	*split_binary*	*binomial* (8)	*split_binary* (0)
128	*split_binary*	*binomial* (8)	*split_binary* (0)
256	*binary*	*binary* (0)	*split_binary* (6)
512	*binary*	*binary* (0)	*chain* (7297)
1024	*split_binary*	*binary* (7)	*chain* (6094)
2048	*split_binary*	*binary* (4)	*chain* (3227)
4096	*split_binary*	*binary* (9)	*chain* (2568)

Table 3 presents selections made for MPI_Bcast using the proposed model-based runtime procedure and the Open MPI decision function. For each message size m, the best performing algorithm, the model-based selected algorithm, and the Open MPI selected algorithm are given. For the latter two, the performance degradation in percents in comparison with the best performing algorithm is also given. We can see that for the Grisou cluster, the model-based selection either pick the best performing algorithm, or the algorithm, the performance of which deviates from the best no more than 3%. Given the accuracy of measurements, this means that the model-based selection is practically always optimal as the performance of the selected algorithm is indistinguishable from the best performance. The Open MPI selection is near optimal in 50% cases and causes significant, up to 160%, degradation in the remaining cases. For the Gros cluster, the model-based selection picks either the best performing algorithm or the algorithm with near optimal performance, no worse than 10% in comparison with the best performing algorithm. At the same time, while near optimal in

40% cases, the algorithms selected by the Open MPI demonstrate catastrophic degradation (up to 7297%) in 50% cases.

The Open MPI decision functions select the algorithm depending on the message size and the number of processes. For example, the Open MPI broadcast decision function selects the chain broadcast algorithm for large message sizes. However, from Table 3 it is evident that chain broadcast algorithm is not the best performing algorithm for large message sizes on both clusters. From the same table, one can see that the model-based selection procedure accurately picks the best performing binomial tree broadcast algorithm for 16 KB and 32 KB message sizes on the Gros cluster, where Open MPI only selects the binomial tree algorithm for broadcasting messages smaller than 2 KB.

6 Conclusions

In this paper, we proposed a novel model-based approach to automatic selection of optimal algorithms for MPI collective operations, which proved to be both efficient and accurate. The novelty of the approach is two-fold. First, we proposed to derive analytical models of collective algorithms from the code of their implementation rather than from high-level mathematical definitions. Second, we proposed to estimate model parameters separately for each algorithm, using a communication experiment, where the execution of the algorithm itself dominates the execution time of the experiment.

We also developed this approach into a detailed method and applied it to Open MPI 3.1 and its MPI_Bcast. We experimentally validated this method on two different clusters and demonstrated its accuracy and efficiency. These results suggest that the proposed approach, based on analytical performance modelling of collective algorithms, can be successful in the solution of the problem of accurate and efficient runtime selection of optimal algorithms for MPI collective operations.

References

1. A Message-Passing Interface Standard. https://www.mpi-forum.org/. Accessed 8 Mar 2021
2. Rabenseifner, R.: Automatic profiling of MPI applications with hardware performance counters. In: Dongarra, J., Luque, E., Margalef, T. (eds.) EuroPVM/MPI 1999. LNCS, vol. 1697, pp. 35–42. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48158-3_5
3. Open MPI: Open Source High Performance Computing. https://www.open-mpi.org/. Accessed 8 Mar 2021
4. MPICH - A Portable Implementation of MPI. http://www.mpich.org/. Accessed 8 Mar 2021
5. Thakur, R., Rabenseifner, R., Gropp, W.: Optimization of collective communication operations in MPICH. Int. J. High Perform. Comput. Appl. **19**(1), 49–66 (2005)

6. Gabriel, E., et al.: Open MPI: goals, concept, and design of a next generation MPI implementation. In: Kranzlmüller, D., Kacsuk, P., Dongarra, J. (eds.) EuroPVM/MPI 2004. LNCS, vol. 3241, pp. 97–104. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30218-6_19
7. Fagg, G.E., Pjesivac-Grbovic, J., Bosilca, G., Angskun, T., Dongarra, J., Jeannot, E.: Flexible collective communication tuning architecture applied to Open MPI. In: Euro PVM/MPI (2006)
8. Pješivac-Grbović, J., Angskun, T., Bosilca, G., Fagg, G.E., Gabriel, E., Dongarra, J.J.: Performance analysis of MPI collective operations. Clust. Comput. **10**(2), 127–143 (2007)
9. Hockney, R.W.: The communication challenge for MPP: Intel Paragon and Meiko CS-2. Parallel Comput. **20**(3), 389–398 (1994)
10. Chan, E.W., Heimlich, M.F., Purkayastha, A., van de Geijn, R.A.: On optimizing collective communication. In: IEEE International Conference on Cluster Computing 2004, pp. 145–155 (2004)
11. Chan, E., Heimlich, M., Purkayastha, A., van de Geijn, R.: Collective communication: theory, practice, and experience: research articles. Concurr. Comput. Pract. Exper. **19**(13), 1749–1783 (2007)
12. Culler, D., Liu, L.T., Martin, R.P., Yoshikawa, C.: LogP performance assessment of fast network interfaces. IEEE Micro **16**(1), 35–43 (1996)
13. Kielmann, T., Bal, H.E., Verstoep, K.: Fast measurement of LogP parameters for message passing platforms. In: Rolim, J. (ed.) IPDPS 2000. LNCS, vol. 1800, pp. 1176–1183. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45591-4_162
14. Lastovetsky, A., Rychkov, V.: Building the communication performance model of heterogeneous clusters based on a switched network. In: IEEE International Conference on Cluster Computing 2007, pp. 568–575 (2007)
15. Lastovetsky, A., Rychkov, V.: Accurate and efficient estimation of parameters of heterogeneous communication performance models. Int. J. High Perform. Comput. Appl. **23**(2), 123–139 (2009)
16. Lastovetsky, A., Rychkov, V., O'Flynn, M.: Accurate heterogeneous communication models and a software tool for their efficient estimation. Int. J. High Perform. Comput. Appl. **24**(1), 34–48 (2010)
17. Rico-Gallego, J.A., Díaz-Martín, J.C., Manumachu, R.R., Lastovetsky, A.L.: A survey of communication performance models for high-performance computing. ACM Comput. Surv. **51**(6), 1–36 (2019)
18. Pješivac–Grbović, J., Fagg, G.E., Angskun, T., Bosilca, G., Dongarra, J.J.: MPI collective algorithm selection and quadtree encoding. In: Mohr, B., Träff, J.L., Worringen, J., Dongarra, J. (eds.) EuroPVM/MPI 2006. LNCS, vol. 4192, pp. 40–48. Springer, Heidelberg (2006). https://doi.org/10.1007/11846802_14
19. Pješivac-Grbović, J., Bosilca, G., Fagg, G.E., Angskun, T., Dongarra, J.J.: Decision trees and MPI collective algorithm selection problem. In: Kermarrec, A.-M., Bougé, L., Priol, T. (eds.) Euro-Par 2007. LNCS, vol. 4641, pp. 107–117. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74466-5_13
20. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., Burlington (1993)
21. Hunold, S., Bhatele, A., Bosilca, G., Knees, P.: Predicting MPI collective communication performance using machine learning. In: IEEE International Conference on Cluster Computing 2020, pp. 259–269 (2020)
22. Wickramasinghe, U., Lumsdaine, A.: A survey of methods for collective communication optimization and tuning. arXiv preprint arXiv:1611.06334 (2016)

23. Grid5000. http://www.grid5000.fr. Accessed 8 Mar 2021
24. Lastovetsky, A., Rychkov, V., O'Flynn, M.: MPIBlib: benchmarking MPI communications for parallel computing on homogeneous and heterogeneous clusters. In: Lastovetsky, A., Kechadi, T., Dongarra, J. (eds.) EuroPVM/MPI 2008. LNCS, vol. 5205, pp. 227–238. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87475-1_32
25. Huber, P.J.: Robust estimation of a location parameter. In: Kotz, S., Johnson, N.L. (eds.) Breakthroughs in Statistics. Springer Series in Statistics (Perspectives in Statistics), pp. 492–518. Springer, New York (1992). https://doi.org/10.1007/978-1-4612-4380-9_35