Accurate heterogeneous communication models and a software tool for their efficient estimation

Alexey Lastovetsky, Vladimir Rychkov, and Maureen O'Flynn

School of Computer Science and Informatics, University College Dublin
Belfield, Dublin 4, Ireland
{alexey.lastovetsky, vladimir.rychkov, maureen.oflynn}@ucd.ie
http://hcl.ucd.ie

Abstract

In this paper, we analyze restrictions of traditional communication performance models affecting the accuracy of analytical prediction of the execution time of collective communication operations on homogeneous and heterogeneous clusters. In particular, we show that the constant and variable contributions of processors and network are not fully separated in these models. Full separation of the contributions that have different nature and arise from different sources would lead to more intuitive and accurate models, but the parameters of such models cannot be estimated from only the point-to-point experiments, which are usually used for traditional models. The paper presents such an intuitive and accurate point-to-point model and describes a set of communication experiments sufficient for estimation of its parameters. It also presents an implementation of the new model in the form of a software tool that automates the estimation of both this model and heterogeneous extensions of traditional communication performance models. We conclude with presentation of experimental results demonstrating that the elaborated model much more accurately predicts the execution time of different algorithms of collective operations than traditional models.

Keywords

Heterogeneous cluster, communication performance model, MPI, communication model estimation.

1 Introduction

The main goal of the presented work is the design and implementation of accurate analytical communication performance models for computational clusters, both homogeneous and heterogeneous. More specifically, we are interested in such models that accurately predict the execution time of any algorithm of any MPI communication operation, whether it is standard or not. The main application of these models, which we consider in the paper, is model-based optimization of MPI collective communication operations when the optimal collective algorithm is the one that will minimize the execution time predicted by the analytical communication model.

The model-based optimization of MPI collective operations for computational clusters is not a new area of research and has been studied in the past. For example, Chan et al., 2004, and Thakur et al., 2005, applied the Hockney model to compare the communication cost of different algorithms of the same collective operation in order to choose the fastest one for different message sizes and numbers of processors.

In the case of heterogeneous clusters, there is another application of communication performance models to the optimization of MPI collective operations. Namely, the performance of a collective operation can be improved by the optimal mapping of heterogeneous processors to the nodes of the communication tree of the operation. Traditional communication performance models are usually homogeneous, with parameters having the same values for all processors and links. Therefore, they will give the same prediction for any mapping. Heterogeneous communication models do distinguish the contributions of different links and processors and hence may be used for this type of optimization. Bhat et al., 2003, and Hatta et al., 2000, built optimal communication trees for collective operations with help of a straightforward heterogeneous extension of the Hockney model.

While the model-based optimization techniques have allowed the researchers to improve at some extent the performance of MPI collective operations, the results, however, appeared not as positive as expected. Many model-based optimization methods, which were supposed to find the optimal solution, returned in practice solutions that were far from optimal. The main reason for this is that the traditional analytical performance models used in these methods appeared quite inaccurate. For example, Pjesivac-Grbovic et al., 2007, show that the predictions provided by the traditional models may significantly differ from the observed communication execution times resulting in a non-optimal decision on switch between algorithms. They report that in some cases the performance penalty of the model-based switching reaches up to 300% due to the inaccuracy.

One limitation of the traditional communication performance models, preventing them from the accurate prediction of the execution time of collective communication operations on homogeneous and, especially, heterogeneous clusters, is that the models combine the contributions in the execution time that have different nature and arise from different sources. Therefore, they do not allow for intuitive analytical expressions of the execution time of most of efficient algorithms of collective communication operations. In the case of heterogeneous clusters, the intuitive models are of the utmost importance as heterogeneous communication performance models have much larger a number of parameters. If the most of these parameters are not intuitive then the model will be absolutely useless.

Any analytical communication performance model only makes sense if it can be implemented, that is, supported by a method of accurate estimation of its parameters on target platforms. Thus, a communication model can be seen as consisting of two parts, the design of the model and the method of its estimation. The estimation part of the model is very important and can exert significant influence on the design part. For example, the estimation methods for all the traditional communication models are based on point-to-point communication experiments. The use of the traditional estimation methods obviously restricts the design of the communication models. Namely, these models cannot include point-to-point parameters that cannot be found from point-to-point communication experiments, even if the additional parameters would make the expression of collective communication algorithms much more intuitive and hence more accurate. In this paper, we show that this is the traditional estimation methods that are the main factor limiting the expressive power of the traditional communication models. They do not allow the models to fully separate the contributions of different nature and originating from different sources in the communication cost.

Heterogeneous communication performance models have not been as intensively studied as homogeneous ones yet. One original heterogeneous model, the design of which cannot be supported by the traditional estimation methods, is proposed in (Lastovetsky et al., 2006). This model separates the variable contributions of the processors and network allowing for more intuitive expressions of collective algorithms than the traditional models or their straightforward extensions. A new estimation method supporting the design of this model is proposed in (Lastovetsky and Rychkov, 2007). The software tool that we will present in this paper implements this new estimation method and therefore can be used for automatic estimation of non-traditional heterogeneous communication models.

In general, there is obvious lack of software for automation of the estimation of communication performance models. The *logp_mpi* library (Kielmann et al., 2000) is a rare exception. It estimates the PLogP parameters for a pair of processors and therefore can be used directly only for homogeneous platforms. The software tool that we will present in this paper automates the estimation of the straightforward heterogeneous extensions of the traditional models, such as Hockney, LogP, LogGP, and PLogP. Thus, this software tool can estimate both traditional and advanced models and can be used for both heterogeneous clusters and homogeneous ones as a particular case of the formers.

This paper is organized as follows. In Section 2, we analyze the design and estimation methods of the traditional communication performance models and their straightforward extensions to heterogeneous computational clusters. In Sections 3, we describe the design of an advanced heterogeneous model, LMO, that allows for intuitive and accurate expression of collective communication algorithms on switched heterogeneous clusters, and outline the method of estimation of its parameters. In Section 4, we present a software tool that implements methods of estimation of both the heterogeneous extensions of the traditional models and the LMO model, and show how the software tool can be used. In Section 5, we present experimental results demonstrating the accuracy of the LMO model.

2 Traditional communication performance models and their straightforward heterogeneous extensions

In this section, we analyze the limitations of traditional communication performance models, preventing them from accurate estimation of the execution time of collective communication operations on computational clusters with a single switch.

The accuracy of the analytical prediction provided by a communication performance model depends on how easy and natural the execution time of collective operations can be expressed via a combination of the model's parameters. An ideal intuitive communication performance model for a homogeneous or heterogeneous cluster with a single switch can be described as follows:

- It would be based on point-to-point parameters that reflect and separate ***constant*** and ***variable*** contributions of ***processors*** and ***network***. This full separation of the contributions that have a different nature and arise from different sources will lead to more intuitive analytical expressions of the communication execution time. In traditional communication performance models, the constant and variable contributions of processors and network are not fully separated.
- The execution time of any collective communication operation could be presented by a combination of ***maximums*** (parallel part) and ***sums*** (sequential part) of the point-to-point parameters. The formula of the execution time might include ***empirical parameters*** that reflect the ***irregular behaviour*** of the collective operation and that are found empirically for a particular platform. Traditional models do not include such empirical parameters.
- There must be ***a set of communication experiments*** that allows for the ***accurate estimation of the parameters***. Traditional models are designed so that their parameters can be estimated from point-to-point communication experiments. The attempts to separate the contributions lead to a model whose parameters cannot be estimated from the point-to-point experiments only.

Next we outline some most popular traditional models and propose their straightforward heterogeneous extensions. For each model we describe both its design and the method of estimation of its parameters.

The parameters of the **Hockney model** (Hockney, 1994) combine the processor and network contributions. The execution time of point-to-point communication is expressed as $\alpha + \beta M$, where α is the latency (constant contributions from processors and network), β is the bandwidth (variable contributions from processors and network) and M is the message size. The Hockney parameters are estimated with help of series of roundtrips $i \underset{M}{\overset{M}{\rightleftarrows}} j$ with different message sizes M.

We can extend the Hockney model for heterogeneous clusters and introduce different parameters α_{ij} and β_{ij} for different pairs of processors, which also combine the processor and network contributions. The parameters of both original and extended models are estimated with help of series of the roundtrip communication experiments in one of two ways:

- Two series of roundtrips: one with empty messages, $\left\{ i \underset{0}{\overset{0}{\rightleftarrows}} j \right\}_{k=0}^{R}$ (to get the latency parameter from the average execution time, $\alpha = \bar{T}(0)$), and the other with non-empty ones, $\left\{ i \underset{M}{\overset{M}{\rightleftarrows}} j \right\}_{k=0}^{R}$ (to get the bandwidth, $\beta = \frac{\bar{T}(M) - \alpha}{M}$), or
- One series of roundtrips with messages of different sizes (to perform a linear regression, which fits the execution time into a linear combination of the Hockney parameters and a message size, $T_k \approx \alpha + \beta M_k$): $\left\{ i \underset{M_k}{\overset{M_k}{\rightleftarrows}} j \right\}_{k=0}^{R}$.

A more elaborated model, **LogP** (Culler et al., 1993), predicts the time of network communication for small fixed-sized messages in terms of the latency, L, the overhead, o, the gap per message, g, and the number of processors, P. The latency, L, is an upper bound on the time to transmit a message from its source to destination; it reflects the constant contribution of network. The overhead, o, is the time period during which the processor is engaged in sending or receiving a message (a constant processor contribution). The gap, g, is the minimum time between consecutive transmissions or receptions; it is the reciprocal value of the end-to-end bandwidth between two processors, so that the network bandwidth can be expressed as L/g. According to LogP, the time of point-to-point communication can be estimated by $L + 2o$. The LogP model assumes that a large message is sent as a series of short data units. In the formula for a series, the gap parameter will be used: $L + 2o + (M-1)g$ where M is the number of short data units. Therefore, the gap can be attributed to the variable contributions of processors and network.

The **LogGP** model (Alexandrov et al., 1995), an extension of LogP, takes into account the message size by introducing the gap per byte parameter, G. The point-to-point communication time is estimated by $L + 2o + (M-1)G$. The original gap parameter, g, is also used in the model to represent the delays between consecutive communications. For example, the execution time of m sendings of M bytes is estimated as follows: $L + 2o + (M-1)G + (m-1)g$. Both these gap parameters combine the contributions of processors and network. The gap g represents the constant contribution, while the gap per byte G represents the variable contribution.

In the **PLogP** (parameterized LogP) model (Kielmann et al., 2000), all parameters except for the latency are piecewise linear functions of the message size, and the meaning of parameters differs slightly from LogP. The meaning of the latency, L, is not intuitive; it combines all constant contribution factors such as copying to/from the network interfaces and the transfer over the network. The send, $o_s(M)$, and receive, $o_r(M)$, overheads are the times that the source and destination processors are busy for the duration of communication (variable contributions of processors). They can be overlapped for sufficiently large messages, that is, the receiving can start while the sending has not been finished yet. The gap, $g(M)$, is the minimum time between consecutive transmissions or receptions; it is the reciprocal value of the end-to-end bandwidth between two processors for messages of a given size M. The gap is assumed to cover the overheads ($g(M) \geq o_s(M)$, $g(M) \geq o_r(M)$) and represents mixed processor-network variable contributions. According to the PLogP model, the point-to-point execution time will be equal to $L + g(M)$. Unlike the LogP/LogGP and Hockney models, the PLogP model is not linear and therefore can more accurately approximate the execution time of point-to-point operations. However, this feature in no way can help in more accurate analytical predictions of collectives algorithms because its functional parameters still combine the contributions of different origin. Indeed, larger numbers of such non-intuitive parameters (each piecewise linear function can be considered as a set of constant parameters) do not make analytical expression of the execution time of collective operations more intuitive.

The parameters of the LogP-based models are estimated by more complicated point-to-point experiments. In addition to roundtrips ($i \underset{0}{\overset{M}{\rightleftarrows}} j$ and $i \underset{M}{\overset{0}{\rightleftarrows}} j$), one-direction consecutive sendings with a confirmation ($i \underset{0}{\overset{\overbrace{M \ldots M}^{s}}{\rightleftarrows}} j$) are used to estimate the gap parameter: $g = T_s / s$, where $s = 2^x$. The number of messages s is chosen to be sufficiently large in order to ensure that the point-to-point communication time is dominated by the factor of bandwidth rather than latency. This experiment, also known as saturation, reflects the nature of the gap parameter but takes a long time. The estimation of the functional PLogP parameters, especially $g(M)$, will be time consuming because these experiments are performed for multiple message sizes, which are selected adaptively. For example, if the $g(M_k)$ is not consistent with the linearly extrapolated value based on $g(M_{k-2})$ and $g(M_{k-1})$, then another measurement will have to be performed for the message size $M_k^* = (M_k + M_{k-1})/2$.

The LogP-based models can be applied to heterogeneous clusters in the same way as the Hockney model. Namely, the parameters are first found for all pairs of processors, with the above experiments being performed for each link. Then, these parameters (heterogeneous version) or their average values (homogeneous version) are used in modelling.

Communication performance models that separate the constant and variable contributions of processors and network would lead to more accurate predictions of the communication execution time. In (Lastovetsky et al., 2006; Lastovetsky and Rychkov, 2007), an analytical heterogeneous communication performance model, separating the variable contributions of processors and network, is proposed. The model, called **LMO**, is designed for both homogeneous and heterogeneous clusters based on a switched network. While this LMO model provides more intuitive and accurate expression of the execution time of MPI collective operations, its parameters cannot be estimated from the traditional point-to-point experiments only. A solution of this problem, proposed in (Lastovetsky and Rychkov, 2007; Lastovetsky and Rychkov, 2009), is to introduce additional collective communication experiments involving more than two processors. These experiments are designed to give us sufficient data in order to build and solve simple systems of equations to find the point-to-point parameters.

While separating the variable contributions of processors and network, the model still combines the fixed delays sourced from the processors and the network. An improved version of the LMO model, which additionally separates the constant contributions of processors and network, is proposed by Lastovetsky et al., 2009. In the following section, we describe this model and the method of its estimation.

3 LMO, the advanced heterogeneous communication performance model

The LMO model is based on six parameters characterizing the point-to-point communication: $(C_i, t_i) \xrightarrow{(L_{ij}, \beta_{ij})} (C_j, t_j)$. Like the most of traditional models, it represents the point-to-point communication time by a linear combination of its parameters and the message size. The execution time of sending a message of M bytes from processor i to processor j in a heterogeneous cluster is estimated by $C_i + L_{ij} + C_j + M(t_i + \frac{1}{\beta_{ij}} + t_j)$, where:

- C_i, C_j are the fixed processing delays;
- t_i, t_j are the delays of processing of a byte;
- L_{ij} is the latency;
- β_{ij} is the transmission rate.

The delay parameters, which are attributed to each processor, reflect the heterogeneity of the processors. The latencies and transmission rates, which correspond to each link, reflect the heterogeneity of communications. For networks with a single switch, it is realistic to assume $L_{ij} = L_{ji}$ and $\beta_{ij} = \beta_{ji}$. In terms of the Hockney model, the parameters can be expressed as follows: $C_i + L_{ij} + C_j = \alpha_{ij}^H$, $t_i + \frac{1}{\beta_{ij}} + t_j = \beta_{ij}^H$, which means that we distinguish between the processors' and network contributions in the constant and variable parts of the point-to-point execution time.

This model provides more flexibility to express the execution time of collective algorithms. Namely, the formulas for collectives can include the fixed processor delays and latencies in different combinations of maximums and sums, which will reflect, for example, the cases when the processor delays are serialized, while transmission is performed in parallel. Consider how this model can be used to express the execution time of MPI collective operations for a heterogeneous cluster with a single switch. The formulas are intuitive, including combinations of sums and maximums of the point-to-point parameters. The expression for the linear scatter is the following:

$$T_{scatter}(M) = (n-1)(C_r + Mt_r) + \max_{i=0, i \neq r}^{n-1}\left(L_{ri} + \frac{M}{\beta_{ri}} + C_i + Mt_i\right) \tag{1}$$

The sequential part of this formula, $(n-1)(C_r + Mt_r)$, is attributed to the root processor, which serially processes the messages to be sent to the rest $n-1$ processors. The maximum reflects the parallel transmissions followed by the parallel processing on the receivers. Thus, this formula conforms to the features of network switches, which parallelize the messages addressed to different processors.

The early version of the LMO model (Lastovetsky et al., 2006; Lastovetsky and Rychkov, 2007) reflected some irregularity in the communication time of the linear scatter. On computational clusters with a TCP/IP layer, we did observe a leap in the execution time of the linear scatter and included this feature into the model. However, the scale of the leap appears not that significant and therefore in the improved version of the LMO model presented in this paper, we use a linear approximation of the scatter execution time, which appears sufficiently accurate in practice (as seen, for example, in Figure 3). Despite this, we still estimate the message size, S, for which the leap is observed, as an important parameter used in the design of communication experiments for estimation of the LMO point-to-point parameters.

The execution time of the linear gather is expressed with help of the point-to-point parameters of the LMO model (analytical part) and some additional parameters that reflect the irregularities observed in the execution time of collective operations on switched clusters (empirical part):

$$T_{gather}(M) = (n-1)(C_r + Mt_r) + \begin{cases} \max\limits_{i=0, i \neq r}^{n-1}\left(L_{ri} + \frac{M}{\beta_{ri}} + C_i + Mt_i\right) & M < M_1 \\ \sum\limits_{i=0, i \neq r}^{n-1}\left(L_{ri} + \frac{M}{\beta_{ri}} + C_i + Mt_i\right) & M > M_2 \end{cases} \tag{2}$$

This formula also reflects the serialization of the processing on the root, which receives all messages. The extra threshold parameters, M_1 and M_2, are found from the observations of the execution time of the linear gather and categorize the message sizes for which the performance of the linear gather differs. For the small (less than M_1) and large (larger than M_2) messages the execution time increases linearly with the increase of message size, while for the medium size messages the non-linear and non-deterministic escalations of the execution time are observed (Lastovetsky and O'Flynn, 2007; Lastovetsky et. al, 2007). These parameters depend on the particular cluster and MPI implementation. For example, on the 16-node heterogeneous cluster specified in Table I, we observed $M_1 = 4KB$, $M_2 = 65KB$ for LAM 7.1.3 and $M_1 = 3KB$, $M_2 = 125KB$ for MPICH 1.2.7. The maximum in formula (2) reflects the parallel processing delays on the processors and parallel transmissions supported by network switch. However, the sending of large messages to one destination is serialized and is hence expressed by the sum.

The point-to-point parameters of the LMO model cannot be estimated from the point-to-point experiments only. The point-to-point experiments with different messages sizes cannot give us sufficient independent data in order to build and solve systems of equations and find the point-to-point parameters. Unlike the traditional communication performance models, this model requires more complicated communication experiments. The full procedure is described in Lastovetsky et al., 2009. It requires minimum three processors. In addition to roundtrips, $i \underset{M}{\overset{M}{\rightleftarrows}} j$, $j \underset{M}{\overset{M}{\rightleftarrows}} k$, $k \underset{M}{\overset{M}{\rightleftarrows}} i$, with empty and non-empty messages M, it includes the communications between the three processors, $i \underset{0}{\overset{M}{\rightleftarrows}} j,k$, $j \underset{0}{\overset{M}{\rightleftarrows}} i,k$, $k \underset{0}{\overset{M}{\rightleftarrows}} i,j$, which consist of parallel sending of empty or non-empty messages M from one of the processors to others (a scatter) and receiving of empty replies (a gather). For reliable estimation of the parameters, these communication experiments are repeated multiple times and average execution times are used. Then, equations (1) and (2) are used to approximate the execution time of these communication experiments, which can be represented as a sum of the execution times of the linear scatter and gather, $T_{ijk}(M) = T_{scatter}(M) + T_{gather}(0)$:

$$T_{ijk}(0) = 2(2C_i + \max_{x=j,k}(L_{ix} + C_x)) \tag{3}$$

$$T_{ijk}(M) = 2(2C_i + Mt_i) + \max_{x=j,k}(2(L_{ix} + C_x) + M(\frac{1}{\beta_{ix}} + t_x)) \tag{4}$$

Then, the systems of equations are built for each triplet of processors. The communication experiments with empty messages give us the solution for the fixed point-to-point parameters, (5), while their counterparts with non-empty messages do the same for the variable parameters, (6):

$$\begin{cases} C_i = (T_{ijk}(0) - \max_{x=j,k} T_{ix}(0))/2 & L_{ij} = T_{ij}(0)/2 - C_i - C_j \\ C_j = (T_{jik}(0) - \max_{x=i,k} T_{jx}(0))/2 & L_{jk} = T_{jk}(0)/2 - C_j - C_k \\ C_k = (T_{kij}(0) - \max_{x=i,j} T_{kx}(0))/2 & L_{ik} = T_{ik}(0)/2 - C_i - C_k \end{cases} \tag{5}$$

$$\begin{cases} t_i = (T_{ijk}(M) - \max_{x=j,k}(T_{ix}(0) + T_{ix}(M))/2 - 2C_i)/M \\ t_j = (T_{jik}(M) - \max_{x=i,k}(T_{jx}(0) + T_{jx}(M))/2 - 2C_j)/M \\ t_k = (T_{kij}(M) - \max_{x=i,j}(T_{kx}(0) + T_{kx}(M))/2 - 2C_k)/M \\ \frac{1}{\beta_{ij}} = (T_{ij}(M)/2 - C_i - L_{ij} - C_j)/M - t_i - t_j \\ \frac{1}{\beta_{jk}} = (T_{jk}(M)/2 - C_j - L_{jk} - C_k)/M - t_j - t_k \\ \frac{1}{\beta_{ik}} = (T_{ik}(M)/2 - C_i - L_{ik} - C_k)/M - t_i - t_k \end{cases} \tag{6}$$

Their solution consists of several comparisons and simple arithmetic operations.

We send the messages of medium size, $M < S$, to avoid the possible leap in the execution time of the scatter, and receive empty replies to eliminate the escalations in the execution time of the gather (Lastovetsky et. al, 2007). Therefore, the procedure of the estimation of the point-to-point parameters must be preceded by the estimation of the threshold parameters, which are used to select the message size for the point-to-two communication experiments. To estimate the threshold parameters, we use the linear scatter and gather benchmarks for different message sizes. The execution time of both communication operations can be approximated by a linear function, with slight breaks in the case of linear scatter and significant escalations in the case of linear gather (see Section 5). We use the algorithm proposed by Bai et al., 2003, which allows for detecting the first break in the execution time of the scatter, S, and the range of large messages for the gather, M_2. Then, we find the M_1 threshold parameter as the minimum message size for which the escalation of the execution time is observed: $M_1 \approx M^k, k = \min\left\{i : T^{i+1}/T^i > 10\right\}$.

We implement two techniques to minimize the total time of the estimation of the point-to-point parameters. First, we use the fact that all processors participate in communication experiments in more than one triplet. Therefore, the values of parameters are found independently from different independent experiments, and these redundant values are used in the statistical analysis to reduce the number of repetitions of the computational experiments needed for reliable estimation of the parameters. Second, on clusters based on a single switch, two communications over the non-overlapping sets of processors do not affect each other. Therefore, to accelerate the estimation procedure for this platform, we perform the communication experiments over non-overlapping triplets of processors in parallel.

The presented LMO model allows us to build the formulas that accurately reflect different aspects of communications in the algorithms of collective operations that are performed on our target platform, switched clusters. The accuracy of the prediction with the LMO model will be demonstrated in Section 5.

4 The design of the software tool

In this section, we describe the design of the CPM (Communication Performance Modelling) software tool that automates the estimation of parameters of both traditional and advanced heterogeneous communication performance models. It implements the estimation procedures presented in Sections 2 and 3. The software tool consists of a library and command-line utilities. The library provides an API for measurement of the execution time of communication experiments (the Measurement module) and for estimation of models and prediction of the execution time of collective algorithms with the models (the Models module). Its structure is shown in Figure 1. The library is implemented in C/C++ on the top of MPI with help of some third-party software. The third-party software is mainly used for statistical and regression analysis, and for estimation of traditional homogeneous models.

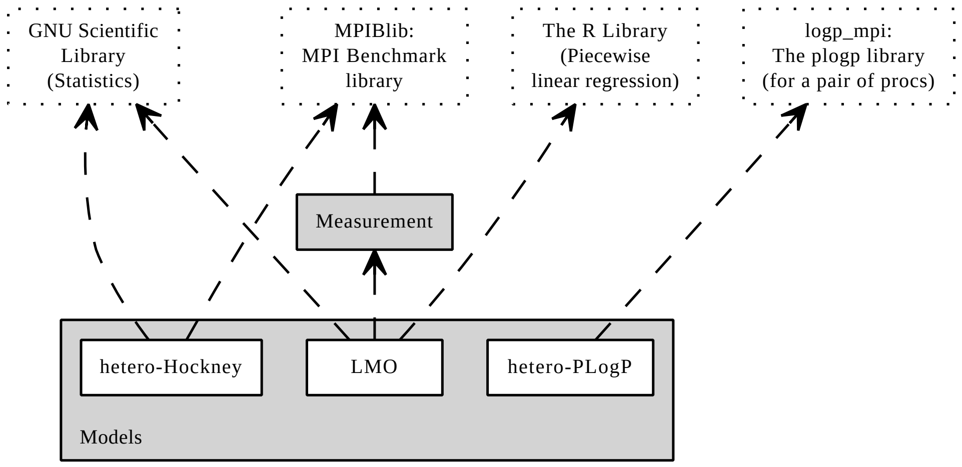
GNU Scientific Library (Statistics)
MPIBlib: MPI Benchmark library
The R Library (Piecewise linear regression)
logp_mpi: The plogp library (for a pair of procs)
Measurement
hetero-Hockney
LMO
hetero-PLogP
Models

Figure 1. The main modules of the library.

The CPM software tool provides a command-line utility that estimates the heterogeneous communication performance models with given accuracy and efficiency, and stores the estimates in files. It allows for using both static and dynamic model parameters. Although the software tool is designed for heterogeneous clusters, it can be applied to homogeneous clusters as well. The main target platform is a computational cluster with a network switch.

4.1 Benchmarking of communication experiments

In this subsection, we describe the API for benchmarking of communication experiments required for estimation of parameters of heterogeneous communication performance models. This API consists of two parts: the MPIBlib benchmarking library (Lastovetsky et al., 2008) and the Measurement module. MPIBlib provides a set of point-to-point and collective benchmarks, some of which are used for estimation of the heterogeneous Hockney and LMO models. The Measurement module implements the point-to-two communication experiments for the LMO model and measures their execution time.

The following MPIBlib function, which measures the execution time of a point-to-point communication operation, is used in the Models module for the estimation of the Hockney and LMO point-to-point parameters:

```
void MPIB_measure_p2p(MPIB_p2p_container* container,
  MPI_Comm comm, int M, int parallel,
  MPIB_precision precision, MPIB_result** results);
```

The `container` argument encapsulates the point-to-point communication operation to be measured. For the model estimation, we use the roundtrip, $i \underset{M}{\overset{M}{\rightleftarrows}} j$, based on the blocking MPI send and receive in standard mode. Function `MPIB_measure_p2p`, like all benchmarking functions, has several mandatory arguments, which represent an MPI communicator, `comm`, a message size, `M`, a precision and the result of the measurement.

The precision argument, `precision` (Figure 2), specifies how many times to repeat the communication experiment in order to obtain an accurate result. It is a data structure that consists of four parameters, which are used in the following way:

- If parameters `min_reps` and `max_reps` (which represent minimum and maximum number of repetitions) have the same value, the communication experiment will be repeated a fixed number of times, equal to this value. This controls efficiency of benchmarking but does not provide a certain level of accuracy.
- If `min_reps < max_reps`, the communication experiment will be repeated until the sample of the measured execution times satisfies the Student's t-test (GNU Scientific Library, Galassi et al., 2009), with the confidence level, `cl`, and relative error, `eps`, or the number of repetitions reaches its maximum, `max_reps`. In this case, the number of repetitions for different pairs of processors and for different message sizes will vary, but a certain accuracy of benchmarking will be guaranteed.

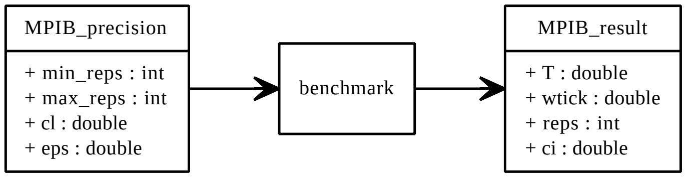

Figure 2. The MPIBlib interface of benchmarking.

Consider how this function and, in particularly, its precision argument are used in the estimation of the heterogeneous Hockney model. The first method of estimation (see Section 2) measures two roundtrips (with empty and non-empty messages) and therefore requires a very high accuracy of measurements. The benchmarking function will be called twice, first time with the message size argument equal to zero and second time with a message size $M > 0$. To guarantee the high precision of the measurements, we set the maximum number of repetitions to be sufficiently large, with the confidence level 95% and the relative error 2.5%. The second method uses multiple roundtrips with different message sizes and therefore requires a high-speed measurement of each individual roundtrip. The accuracy of this method depends on the number of message sizes, for which roundtrips are measured, returning a satisfactory estimate even with low precision of individual measurements. In this latter case, the `MPIB_measure_p2p` function will be called multiple times with different message sizes in the low precision mode: `min_reps = max_reps = 1`.

The `results` argument is an output argument of the function. It is an array of data structures, containing the result of the measurement for each pair of processors. This data structure includes the following fields:

- the average measured communication execution time, `T`,
- the MPI clock resolution of the processor on which the time has been measured, `wtick`,
- the number of repetitions that the benchmark has actually taken, `reps`, and
- the confidence interval within which the time has been measured, `ci`.

The message size and the execution time are used in the calculation of model parameters.

The point-to-point benchmarking function is optimized for clusters with a single switch. As network switches are capable of forwarding packets between sources and destinations without contention, several point-to-point communications can be run in parallel, with each process being involved in no more than one communication. This decreases the execution time the benchmark takes and returns quite accurate results for clusters with a single switch. For other platforms, this benchmark can be performed in the sequential mode, when the `parallel` argument is set to zero.

Another MPIBlib function, which measures the communication execution time of collective operations, is used for the estimation of the threshold parameters of the LMO model (see Section 3):

```
void MPIB_measure_coll(MPIB_coll_container* container,
  MPI_Comm comm, int M, int root,
  MPIB_precision precision, MPIB_result* result);
```

Its `container` argument encapsulates the collective operation to be measured. When finding the LMO threshold parameters, the containers for linear scatter and gather are used. The method of estimation of these parameters is based on the piecewise linear regression and requires a large number of single measurements, that is, measurements made without any repetitions. Indeed, in this method, we are looking for any escalation in the communication execution time, in order to detect the structural changes in the linear regression models as accurately as possible. Measuring with higher precision, in this case, would take longer and provide the average execution time, which may lead to a wrong regression model.

The Measurement module implements on the top of the MPIBlib library a function for benchmarking of the point-to-two communications required for the estimation of the LMO point-to-point parameters (as described in Section 3):

```
void CPM_measure_p2pp(MPI_Comm, int M, int parallel,
  MPIB_precision precision, MPIB_result** results);
```

One point-to-two communication experiment consists of a combination of the blocking send and receive operations. Similarly to the point-to-point benchmark, measurements can be performed either serially or in parallel, which is specified by the `parallel` argument. In the parallel mode, as many as possible point-to-two communications on independent triplets of processors are performed simultaneously. During the estimation of the LMO point-to-point parameters, this function is called twice, with an empty and non-empty messages, in the high-precision mode. We describe this estimation procedure in detail in Section 4.2.

The outlined benchmarking functions are used in the Models module, which provides a set of functions for estimation of different models (model estimators). The model estimators call benchmarking functions, collect measurement results, build and solve the systems of equations in order to calculate the values of parameters.

4.2 API for heterogeneous communication performance models

The Models module provides an API for estimation of the heterogeneous communication performance models and prediction of the execution time of collective algorithms with these models. Apart from the LMO model, this module supports the heterogeneous extensions of traditional models, Hockney, LogP, LogGP, and PLogP. For each model, it provides a function for estimation of the model and a set of functions for prediction of the execution time of different MPI communication operations.

The **heterogeneous extension of the Hockney model** can be estimated by one of the following functions implementing both methods discussed in Section 2:

```
void Hockney_estimate(MPI_Comm comm,
  int M, MPIB_precision precision,
  int parallel, Hockney_model** model);
```

```
void Hockney_estimate_regression(MPI_Comm comm,
  int min_size, int max_size, int max_diff, int max_num,
  int parallel, Hockney_model** model);
```

In the first function, the point-to-point parameters are found directly from two series of roundtrips with `0` and `M` bytes, and the accuracy of the returned estimate solely depends on the precision argument, which specifies the number of repetitions for each roundtrip. In the second function, based on linear regression analysis, the parameters are found from a series of single roundtrips, each with a different message size. In this case, the accuracy of the returned estimate is determined by the set of message sizes. The message sizes are found adaptively, during the execution of the function. The corresponding adaptive procedure is controlled by the following arguments:

- The minimum and maximum message sizes, `min_size` and `max_size`,
- The maximum relative difference between each new measured communication execution time and the linear model, `max_diff`, and
- The maximum number of message sizes, `max_num`.

This procedure can be summarised as follows:

- The execution time of the roundtrip with the message size M_k is compared with the linear model based on the times for smaller messages $M_{k-1},...,M_0$.
- If the difference exceeds `max_diff`, the next measurement will be performed for the message size $(M_{k-1}+M_k)/2$, otherwise, for $M_k+2(M_k-M_{k-1})$.
- The number of message sizes is limited by the `max_num` argument. Linear regression analysis is implemented with help of the GNU Scientific Library (Galassi et al., 2009).

In both estimation functions, the measurements can be preformed in parallel.

The output of the estimation functions, `model`, is a data structure containing the parameters of the estimated model, which can be used, in particular, as the input argument of the predicting functions. A function predicting the execution time of a communication operation Y has the following interface:

```
double Hockney_predict_Y(Hockney_model* model, args);
```

For example, the point-to-point execution time will be predicted by the following function:

```
double Hockney_predict_p2p(Hockney_model* model,
  int i, int j, int M)
{
  int index = IJ2INDEX(model->n, i, j);
  return model->a[index] + model->b[index] * M;
}
```

The API for the **heterogeneous extensions of LogP and LogGP models** is similar to the one for the heterogeneous Hockney model. The following functions estimate these models:

```
void LogP_estimate(MPI_Comm comm,
  int m, MPIB_precision precision,
  int parallel, LogP_model** model);
```

```
void LogGP_estimate(MPI_Comm comm,
  int m, int M, MPIB_precision precision,
  int parallel, LogGP_model** model);
```

The input argument `m` specifies the message size that will be used in the communication experiments required for the estimation of the LogP parameters. According to the estimation procedure, this message must be small; default value is 1 byte. Another argument `M` specifies a large message size for the communication experiment estimating the gap per byte parameter, G.

The **heterogeneous extension of the PLogP model** is straightforward and therefore for its estimation, we can use the *logp_mpi* library (Kielmann et al., 2000), namely, a function estimating the PLogP parameters for a pair of processors following the procedure described in Section 2. This function is used in the implementation of the following estimator of the heterogeneous PLogP model:

```
void PLogP_estimate(MPI_Comm comm,
  int min_size, int max_size, int max_diff, int max_num,
  MPIB_precision precision, int parallel, PLogP_model** model);
```

It calls the *logp_mpi* estimation function for all pairs of processors either sequentially or in parallel. In the parallel mode, the estimations for as many as possible independent pairs of processors will be performed simultaneously. The `precision` argument of the `PLogP_estimate` function defines the precision of measurements. The message set for which the parameters (the piecewise linear functions) are estimated, is defined by the four integer arguments, similar to the `Hockney_estimate_regression` function. The only difference is that the `max_diff` argument is used to compare each new measurement with the linear model based on two previous measurements only. The output of the estimation function, `model`, is used in the set of the functions that predict the execution time of different communication operations, using the heterogeneous extension of the PLogP model.

The following function automates the estimation of the **LMO model**, which has been described in Section 3:

```
void LMO_estimate(MPI_Comm comm,
  int min_size, int max_size, int max_diff, int max_num,
  MPIB_precision precision, int parallel, LMO_model** model);
```

The procedure implemented in this function consists of the following steps:

1. First, the linear scatter and gather benchmarks are performed once for each adaptively selected message size.
2. Then, the obtained results of measurement are used to find the threshold parameters with help of the R *strucchange* library (Zeileis et al., 2002). The library automates the detection of the structural changes in piecewise linear regression models.

3. After that, the point-to-point and point-to-two benchmarks are performed in the high-accuracy mode for an empty message and a message of the size slightly less than the value of the threshold parameter S.
4. Finally, the point-to-point parameters are found by building and solving the systems of linear equations based on the results of the benchmarks performed at step 3.

The output of the estimation function, `model`, contains the LMO parameters and is used in the functions predicting the execution time of different communication operations. For example, `LMO_predict_p2p` predicts the execution time of the point-to-point communication operation, `LMO_predict_Scatter_linear` and `LMO_predict_Gather_linear` predict the execution time of the linear algorithms of scatter and gather. Unlike the predicting functions for other models, the LMO ones use not only analytical but also empirical parameters, which allows for reflecting irregularities in the communication execution time (see experimental results in Section 5).

4.3 Use of the software tool

The CPM software tool can be used for accurate and efficient communication performance modelling on computational clusters, homogeneous and heterogeneous. One important application is in the model-based optimization of MPI collective communication operations. The tool automates the estimation of the heterogeneous communication models and therefore facilitates the implementation of different optimization methods based on these models, such as the optimal switching between algorithms, the optimal mapping of processors in the communication trees, and the implementation of the original model-based collective algorithms, which, for example, use irregular communication trees. For this purpose, it can be integrated into an MPI implementation that provides the model-based optimized versions of the collective operations, or it can be used autonomously by the application programmers for application-level optimizations. In both cases, the estimation can be performed either once, for example, upon installation of the corresponding MPI implementation on the cluster, or at runtime, during each individual execution of the program. The first approach is much more efficient as it removes the estimation cost from the application. At the same time, this approach assumes that the communication performance characteristics of the cluster are the same whenever the application is executed. If this is not the case and the communication performance characteristics can significantly differ for different runs of the application, then the second approach allows the application to automatically tune to the changes and use more accurate communication models. However, the penalty to pay is that the estimation cost will be included into the application.

Consider how the software tool can be used by application programmers to improve the performance of their parallel applications. Suppose we have a program designed for heterogeneous computational clusters, which intensively scatters and gathers the irregularly partitioned data and performs some parallel computations:

```
1:  #include "mpi.h"
2:  int main(int argc, char** argv) {
3:    MPI_Init(&argc, &argv);
4:    Loop {
5:      MPI_Scatterv(data, root, MPI_COMM_WORLD);
6:      Computations;
7:      MPI_Gatherv(data, root, MPI_COMM_WORLD);
8:    }
9:    MPI_Finalize();
10: }
```

Using the CPM software tool, the programmer can improve the communication performance of this application. In the following code, the LMO model is used for the optimization:

```
1:  #include "cpm.h"
```

```
2:  int main(int argc, char** argv) {
3:    MPI_Init(&argc, &argv);
4:    LMO_model* model;
5:    if (to_build) {
6:      LMO_estimate(MPI_COMM_WORLD, msgset, precision, parallel,
7:        &model);
8:    }
9:    else {
10:     if (rank == 0)
11:       LMO_read(istream, &model);
12:     LMO_bcast(model, MPI_COMM_WORLD);
13:   }
14:   Loop {
15:     if (LMO_predict_Scatterv_linear(model, data, root) <
16:         LMO_predict_Scatterv_binomial(model, data, root))
17:       MPIB_Scatterv_linear(data, root, MPI_COMM_WORLD);
18:     else
19:       MPIB_Scatterv_binomial(data, root, MPI_COMM_WORLD);
20:     Calculations;
21:     if (LMO_predict_Gatherv_linear(model, data, root) <
22:         LMO_predict_Gatherv_binomial(model, data, root))
23:       MPIB_Gatherv_linear(data, root, MPI_COMM_WORLD);
24:     else
25:       MPIB_Gatherv_binomial(data, root, MPI_COMM_WORLD);
26:   }
27:   MPI_Finalize();
28: }
```

In line 4, a variable for the LMO model is defined. Then, the model is estimated at runtime (lines 5-8) or read from the file and broadcast to all processors (lines 9-13). In the optimized code, the native irregular scatter and gather are replaced by the if-statements (lines 15-16 and 21-22), which compare the predictions of the LMO model for the linear and binomial algorithms, and invoke the fastest (from the point of view of the LMO model) one. As demonstrated in Section 5, the LMO model is sufficiently accurate to make the correct decision on switching between the linear and binomial algorithms. The linear and binomial algorithms are provided by the MPIBlib library (lines 17, 19, 23, 25). In the modified version, some additional cost will be incurred if the model is estimated at runtime. However, this cost will be well compensated by the benefits from the optimized communications in the case of large-scaled applications.

5 Experimental results

In this section, we present experimental results obtained with help of the CPM software tool on a 16-node heterogeneous cluster with a single Ethernet switch and LAM 7.1.3 specified in Table 1.

Table 1. Specification of seven node types and the parameters of the heterogeneous Hockney model of the 16-node heterogeneous cluster.

Node type	Model	OS	Processor	Front Side Bus	L2 Cache	Number of nodes
1	Dell Poweredge SC1425	FC4	3.6 Xeon	800MHz	2MB	2
2	Dell Poweredge 750	FC4	3.4 Xeon	800MHz	1MB	6
3	IBM E-server 326	Debian	1.8 AMD Opteron	1GHz	1MB	2
4	IBM X-Series 306	Debian	3.2 P4	800MHz	1MB	1
5	HP Proliant DL 320 G3	FC4	3.4 P4	800MHz	1MB	1
6	HP Proliant DL 320 G3	FC4	2.9 Celeron	533MHz	256KB	1
7	HP Proliant DL 140 G2	Debian	3.4 Xeon	800MHz	1MB	3

We compare the execution time of scatter and gather communication operations with the predictions provided by the heterogeneous communication performance models. The communication execution time is measured with help of the MPIBlib collective benchmarks. In Figure 3, one can see a deviation from the linear response in the execution time of the linear scatter observed on our cluster for the message sizes beginning from 64KB. Nonetheless, the LMO model gives a satisfactorily accurate linear approximation. The PLogP prediction is less accurate. While providing the same accuracy for medium size messages and, in addition, reflecting the leap in the execution time, after this leap it diverges significantly from the observations. The predictions of the other traditional models are even more inaccurate.

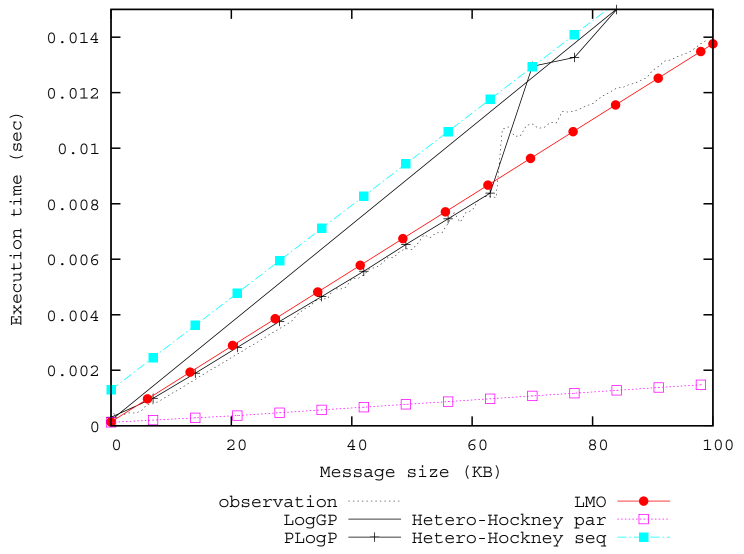

Figure 3. The prediction of the execution time of linear scatter on the 16-node heterogeneous cluster.

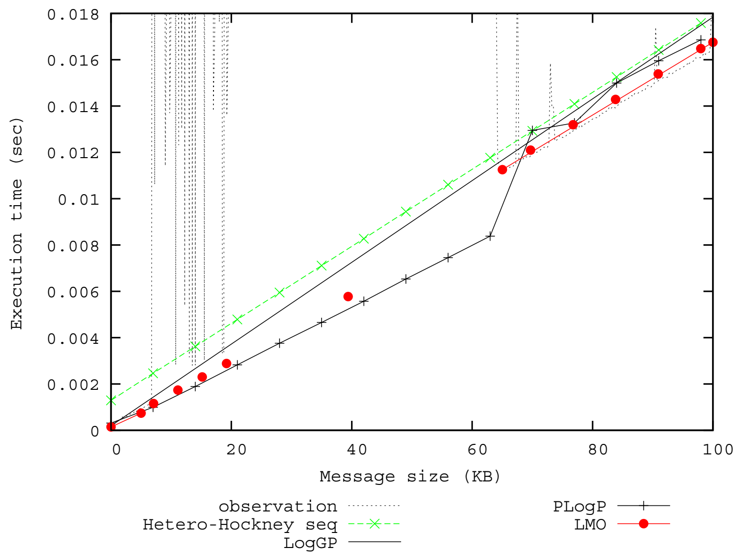

Figure 4. The prediction of the execution time of linear gather on the 16-node heterogeneous cluster.

Figure 4 shows the performance of the linear gather algorithm and the predictions given by the different communication performance models. Only the LMO model reflects the irregular behaviour of the linear gather. For both small (less than 4KB) and large (more than 65KB) messages, this model represents the execution time of the linear gather by straight lines but with different slopes. For medium-sized messages, it reflects the irregular behaviour of the operation in the form of significant escalations of the execution time and gives the most frequent values of escalation and their probability. The escalations are non-deterministic and reaching 0.25 sec. When no escalation happens for a medium-sized message, the execution time is accurately approximated by the continuation of the linear model for small messages as predicted by the LMO model. As to predictions for small and large messages, the LMO model is also more accurate than the other models.

Figure 3 and Figure 4 demonstrate that the intuitive predicting formulas provided by the LMO model are more accurate than the predicting formulas of the traditional models. The accurate prediction of the execution time allows for making correct decisions when switching between algorithms of a collective communication operation. Figure 5 shows the predictions of the heterogeneous Hockney and LMO models for the linear and binomial algorithms of scatter for messages $100KB < M < 200KB$. One can see that the Hockney model mispredicts that the binomial algorithm outperforms the linear one, switching in favour of the first, whereas the decision based on the LMO approximation will be correct.

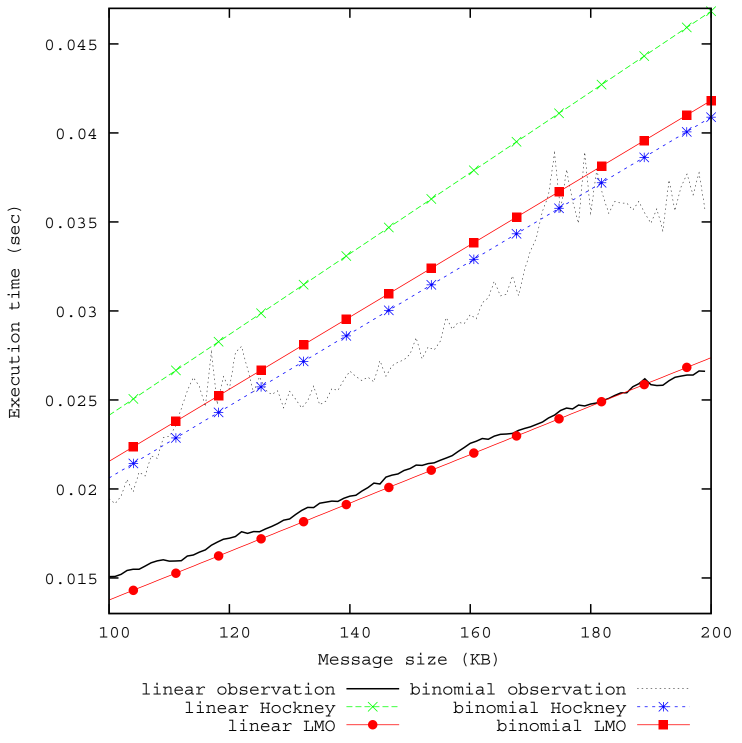

Figure 5. The performance of the linear and binomial algorithms of scatter vs the heterogeneous Hockney and LMO predictions.

The accurate prediction can be a basis for the model-based optimization of collective operations. Figure 6 shows the performance of a simple optimized version of gather that is implemented on the top of its native counterpart by splitting the messages of medium size and performing a series of gathers in order to avoid the escalations. Using the empirical parameters of the LMO model for the linear gather, a 10-fold speedup has been achieved.

The software tool described in this paper provides accurate and efficient estimation of parameters of heterogeneous communication models. The accuracy and efficiency are achieved by using different benchmarking techniques. The user can control the precision of the model parameters and the total estimation time. The software tool supports both traditional and advanced communication performance models. It can be used on both homogeneous and heterogeneous clusters. One important application of the software tool is the implementation of the model-based optimal algorithms of MPI collective communication operations.

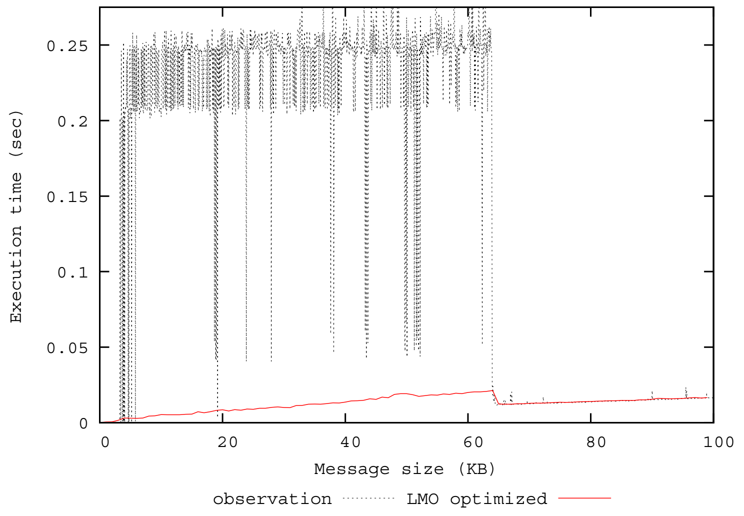

Figure 6. The LMO model-based optimization of the linear gather on the 16-node heterogeneous cluster.

Acknowledgements

This work is supported by the Science Foundation Ireland and in part by the IBM Dublin CAS.

References

1. Alexandrov, A., Ionescu, M., Schauser, K., Scheiman, C. (1997). LogGP: incorporating long messages into the LogP model for parallel computation. J. Parallel Distrib. Comput. **44**(1):71-79.
2. Bai, J., Perron, P. (2003). Computation and analysis of multiple structural change models, J. Appl. Econometrics, **18**(1):1-22.
3. Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K., Santos, E., Subramonian, R., Eicken, T. von. (1993). LogP: Towards a realistic model of parallel computation. In proc. of PPoPP 1993, ACM, pp. 1-12.
4. Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., Rossi, F. (2009). GNU Scientific Library Reference Manual. Bristol: Network Theory Limited.
5. Hockney, R. (1994). The communication challenge for MPP: Intel Paragon and Meiko CS-2. Parallel Comput. **20**(3):389-398.
6. Kielmann, T., Bal, H., Verstoep, K. (2000). Fast measurement of LogP parameters for message passing platforms. In proc. of IPDPS 2000. Springer, pp.1176-1183.
7. Lastovetsky A., Mkwawa I., O'Flynn M. (2006). An Accurate Communication Model of a Heterogeneous Cluster Based on a Switch-Enabled Ethernet Network. In proc. of ICPADS 2006, IEEE Computer Society, vol. 2, pp.15-20.
8. Lastovetsky, A., O'Flynn, M. (2007). A Performance Model of Many-to-One Collective Communications for Parallel Computing. In proc. of IPDPS 2007, IEEE.
9. Lastovetsky, A., O'Flynn, M., Rychkov, V. (2007). Optimization of collective communications in HeteroMPI. In proc. of EuroPVM/MPI 2007. Springer, pp. 135-143.
10. Lastovetsky, A., O'Flynn, M., Rychkov, V. (2008). MPIBlib: Benchmarking MPI communications for parallel computing on homogeneous and heterogeneous clusters. In proc. of EuroPVM/MPI 2008. Springer, pp. 227-238.
11. Lastovetsky, A., Rychkov, V. (2007). Building the communication performance model of heterogeneous clusters based on a switched network. In proc. of Cluster 2007. IEEE, pp. 568-575.

12. Lastovetsky, A., Rychkov, V. (2009). Accurate and efficient estimation of parameters of heterogeneous communication performance models. Int. J. High Perform. Comput. Appl. **23**(2): 123-139.
13. Lastovetsky, A., Rychkov, V., O'Flynn, M. (2009). Revisiting communication performance models for computational clusters. In proc. of IPDPS 2009. IEEE.
14. Pjesivac-Grbovic, J., Angskun, T., Bosilca, G., Fagg, G., Gabriel, E., Dongarra, J. (2007). Performance analysis of MPI collective operations. Cluster Comput. **10**(2):127-143.
15. Zeileis, A., Leisch, F., Hornik, K., Kleiber, C. (2002). Strucchange: An R package for testing for structural change in linear regression models. J. Statistical Software. **7**(2):1-38.