CHAPTER 1

OPTIMIZATION OF COLLECTIVE COMMUNICATION FOR HETEROGENEOUS HPC PLATFORMS

KIRIL DICHEV[1] AND ALEXEY LASTOVETSKY[1]

[1]School of Computer Science and Informatics, University College Dublin, Belfield 4, Dublin, Ireland

1.1 Introduction

Communication plays a central role in parallel computing algorithms. For collective communication, significant gains in performance can be achieved by implementing topology- and performance-aware collectives. In this chapter, we offer a comprehensive overview of the existing research in this area. We observe both MPI collectives as well as alternatives from the distributed computing domain. The existing challenges in analytical and experimental solutions for heterogeneous platforms are presented and discussed.

This work describes the existing methods of optimizing communication for modern platforms. The most basic communication operation is the point-to-point communication involving a sender and a receiver. As a higher level of abstraction, col-

High Performance Computing on Complex Environments.

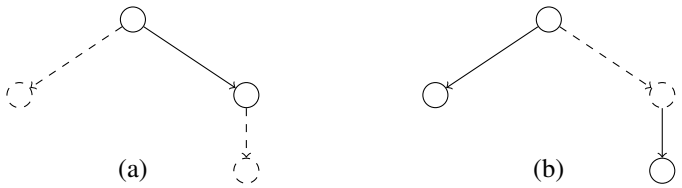

Figure 1.1 Broadcast schedule through a binomial tree. (a) In step 1, the root communicates to its right child. (b) In step 2, two independent point-to-point exchanges can be performed in parallel to complete the operation.

lective communication, which involves the exchange of data between a group of processes, is fundamental in many areas of computing. Parallel implementations of fundamental mathematical kernels like dense matrix multiplication [1] or three-dimensional Fast Fourier Transformation rely on collective operations. More recently, the MapReduce construct [2] widely used by Google for indexing of web pages is gaining in popularity, and also requires collective operations.

In this work, we cover the different research directions for optimizing collective communication operations. This review chapter asks following seemingly simple questions: What are the best practices of implementing an efficient collective operation for a heterogeneous and/or hierarchical platform? How do they differ from efficient implementations for homogeneous platforms? Can the best practices be classified and presented in a clear way? Is there a trend which shows what is and what is not possible in the area of such optimizations for today's complex platforms?

When answering these questions, we focus both on classical high performance approaches, which invariably revolve around MPI collectives, and on more recent and efficient communication libraries. The wide range of collective operations (broadcast, reduce, scatter etc.) and the efficient algorithms for each of them cannot be covered in detail in this work. However, it is helpful to demonstrate the common optimizations for heterogeneous networks based on a use case. In this work, we consider the popular broadcast operation as such a use case. In a broadcast, a root process sends the same data to a group of other processes. The broadcast operation is complete when the last process in the group has received the data. The efficiency of the operation is measured by how fast the broadcast completes.

Even for this fundamental collective operation, there is already a large variety of popular implementations [3]. To name a few, there are the linear tree (or chain or pipeline) algorithm, the binomial tree or the flat tree algorithm. In this work, our algorithm of choice is the binomial tree algorithm. We show a broadcast of a message in a trivial binomial tree in Fig. 1.1. The main advantage of such an algorithm is that (as shown in Fig. 1.1(b)) different point-to-point exchanges can be parallelized. While not always optimal, this algorithm is common for small message broadcasts on trivial networks like single-switch Ethernet clusters.

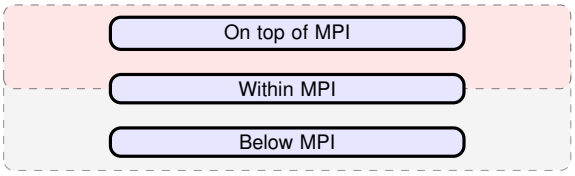

Figure 1.2 Optimizations of collective operations in regard to their relation to MPI - on top, within, or below MPI. In theory, generic optimizations stand above MPI, but as indicated in red, in practice their implementation is either above or within an MPI library.

1.2 Overview of Optimized Collectives and Topology-Aware Collectives

The general area of communication optimization is very broad and includes a number of different directions. Intuitively, the goal of all such optimization is to reduce the global runtime of the communication operations. But there are different ways to achieve this goal. The vastness of optimizations of MPI collectives has obstructed, rather than helped for any division of the different types of optimizations into categories. For clarity, in this section we specify a few categories of such optimizations in regard to the software layer they are embedded in. MPI [4] is still the most used communication library for high performance computing, and we classify all existing approaches in their relation to this library. We present this MPI-centric view in Fig. 1.2. With such a categorization, it is easier to talk of the particular area of interest in this chapter and differentiate it from other work which is also concerned with achieving better performance, but in a different manner.

Optimizations below the MPI layer include tuning of parameters that affect the performance of the underlying protocol. An important example of such tuning [5] demonstrates that the TCP window size has a significant impact on MPI communication on links with a high bandwidth-delay product. Modern grid infrastructures employing fibre optics over very long distances have these properties.

Collective optimizations **within the MPI layer** can be very broad. Some of these are implemented within the MPI library because they require access to hardware-related interfaces. For example, optimizations for Infiniband networks include the use of RDMA calls within MPI [6, 7, 8]. Other such optimizations include accessing kernel modules like CPU affinity to control the migration of MPI processes on cores, and others. Also, some protocols like eager and rendezvous [4], which affect point-to-point and collective operations, are intrinsic to the MPI communication library.

More generic optimized MPI collective algorithms can be implemented **on top of MPI**. The most obvious example is reimplementing a collective based on existing MPI point-to-point calls or MPI collectives.

Still, such generic optimizations are not always implemented on top of MPI, but sometimes are embedded within the MPI layer. The decision to embed an optimization within an MPI implementation in such cases is driven by software development

or even political issues rather than strict requirements. For example, some of the optimizations of collectives we list as generic in the next sections are in fact implemented within Open MPI [9].

The main focus of this work are:

- **Generic optimizations for collective operations** (Sect. 1.6, 1.7), i.e. optimizations not dependent on low-level MPI internals like hardware or low-level protocol details. As highlighted in Fig. 1.2, they can either be implemented within or on top of the MPI layer.

- **Alternative communication libraries** (Sect. 1.8) offer interesting optimizations of communication not present, and often not possible, in existing MPI middleware.

In the following section we present the state-of-the-art in such optimizations of collectives on homogeneous clusters. Then we attempt a structured introduction of the specific area of communication optimizations on heterogeneous networks.

1.3 Optimizations of Collectives on Homogeneous Clusters

Most optimizations of collective operations on homogeneous clusters focus on finding an efficient algorithm on top of point-to-point primitives. Detailed work analyzing an array of collectives [10] considers the general approaches – analytical (through models) and empirical (through experiments). The model of choice is Hockney (see Sect. 1.7.1), and its point-to-point predictions are used to build prediction formulas for more complex collective operations. However, even seemingly different algorithms often produce predictions of similar time complexity. For this reason, the support of experiments is often needed. The decision making process is difficult, and depending on message size and process number, a range of algorithms can be used for a single collective operation. On the example of the broadcast algorithm, binomial tree broadcast is used for small messages, and a combination of a "scatter" and "allgather" operation (in MPI semantics) for large messages. The decision is hard-coded within the MPICH library[1] based on the process count and message size.

A more sophisticated process of optimization can be performed through a more intelligent, system-specific, selection [11]. Clearly, the selection process is good if the selected algorithm is optimal. The optimization consists of the selection from a large set of predefined collective methods. A collective method is considered the coupling of a particular collective algorithm with a segment size. Formally, a relation D is defined which maps "the set of all possible collective operation input parameters PIN^c" to the set of available methods M^c while minimizing collective duration:

$$D : PIN^c \rightarrow M^c \tag{1.1}$$

[1]MPICH web page: `www.mpich.org`

The work proceeds to search for a mapping that selects the fastest collective method for the input parameters on a given platform. Finding such a mapping is extremely difficult in practice. The work proposes an analytical approach (Sect. 1.7.1 discusses these models), and an empirical approach (graphics encoding schemes and statistical learning methods). Both approaches have advantages and drawbacks, and can be used on homogeneous networks.

The empirical solutions have the advantage of being accurate for different platforms. But they have a significant disadvantage – to build a decision map a number of dimensions need to be examined. Typically, the process number *p* and the message size *m* are two orthogonal dimensions of the input. Additionally, different MPI implementations and different platforms make the empirical process very expensive and require some heuristics in most cases.

1.4 Heterogeneous Networks

In recent years, the term "heterogeneity" has been heavily linked with the advent of many-core processors and accelerators like GPUs. In this work, we consider the network heterogeneity rather than the processor heterogeneity. A variety of examples for network heterogeneity can be found in today's computing platforms. Distributed systems traditionally provide a high level of network heterogeneity – popular examples of such heterogeneous high performance systems include grid and cloud infrastructures. But even on supercomputers like the IBM/BlueGene, the torus topology is a heterogeneous network in regard to the latency, with node to node communication depending on the number of hops between sender and receiver. On the other hand, supercomputers with thousands of nodes – in which each node consists of many-cores and possibly accelerators – can also clearly be characterized as heterogeneous. Communication on different levels shows different properties for these modern resources – e.g. intra-node vs. inter-node communication.

1.4.1 Comparison to Homogeneous Clusters

The most significant challenge in optimization of collectives for heterogeneous networks, compared to homogeneous networks, is in the increase in complexity. If we trivially assume the cost of each communicating pair of nodes i and j to be C_{ij}, then "the problem of finding the optimal broadcast schedule in such a heterogeneous system is NP-complete" [12]. We already observed how analytical and empirical solutions can be used to support the selection of an efficient collective algorithm. With the introduction of this new complexity, the empirical approach becomes more challenging. It is not sufficient to run a set of benchmarks $p_i * m_j$ for a selection of process counts and message sizes for a handful of fixed communication schedules. For example, on a homogeneous cluster a binomial tree algorithm would yield the same performance independent of the reordering of processes along the tree. On a heterogeneous cluster, the differing cost per link C_{ij} means that an exponential number of possible binomial trees can be formed, with potentially different total

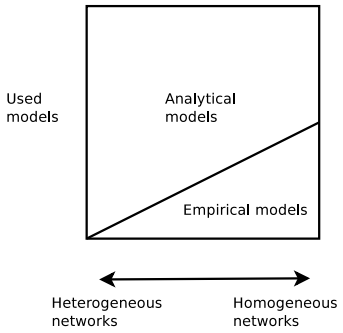

Figure 1.3 While both empirical and analytical solutions are used for homogeneous clusters, analytical solutions are the only option for heterogeneous and more complex networks.

communication cost. This exponential increase in complexity is a challenge both for empirical and analytical solutions, but analytical solutions promise more flexibility in how they can be used, and are hugely more efficient than running communication benchmarks. Therefore, analytical solutions seem like the only viable option for optimizations on complex networks. We visualize this limitation in Fig. 1.3. While we can try both empirical and analytical models on homogeneous networks, with increasing network heterogeneity more empirical, or "lookup table" based solutions, give way to analytical solutions.

1.5 Topology- and Performance-Aware Collectives

Optimized collectives for heterogeneous networks generally follow two main phases as shown in Fig.1.4. In a first phase, a network model is created which characterizes the underlying network in some form and way. In a second phase, this model is used for efficient collective communication.

However, two different categories of collective communication for heterogeneous platforms can be identified – topology-aware and performance-aware collectives (see also [13]).

The now common term **topology aware** seems to originate from the networking domain (e.g. [14]), where information from routing tables can help reduce the number of hops when transferring packets. The central property of this type of optimization is that it does not rely on actual performance data, but rather on the network topology – which is often synonymous to hierarchy and structure. The network properties for topology-aware communication are configured either manually or automatically.

The other popular direction of optimization is **performance aware** communication. In this case, network properties are reconstructed with performance measurements. This approach is useful when topology information cannot be provided or is not sufficient to determine the performance.

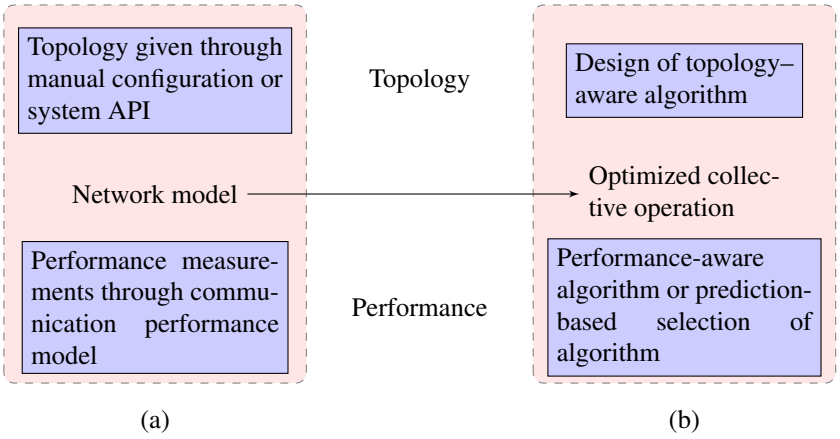

Figure 1.4 General phases of topology or performance-aware collective communication. (a) A network model represents some properties of the network. (b) The network model is used either in a network-aware algorithm, or in a prediction-based selection from a pool of network-unaware algorithms.

1.6 Topology as Input

Naturally, the first use of topology-aware collectives was through manual configuration. Early work of this kind includes different MPI libraries for distributed computing, where the hierarchy information is provided by the user. The hierarchy in the test cases typically consists of two-levels – intra- and inter-cluster information. Some work reimplements efficiently both the intra- and inter-cluster communication on top of MPI point-to-point communication [15]. Other work only reimplements the inter-cluster communication through TCP channels, but relies on the optimized MPI collectives within clusters [16]. Naturally, the main design goal of such middleware is to minimize the expensive cross-site communication for various heterogeneous settings. More recently, with the advent of multi-core machines, similar ideas were introduced for topology-aware communication on high-performance clusters. A simple step is a remapping of MPI processes to follow the topology of the resources [17]. More advanced approaches create new collectives with role assignment of processes according to placement [18, 19].

In MPI, collectives are usually implemented through a single spanning tree. However, using multiple spanning trees (MST) offers performance gains for heterogeneous networks, particularly for large-message collectives. One notable exception in the MPI community is work on optimizing collectives for the meshes/tori topology on the IBM BlueGene supercomputer [20]. On the example of a large-message broadcast “the basic idea is to find a number of non-overlapping spanning trees in the rectangular mesh/tori. The broadcast message is split into components, and each component is pipelined separately (and simultaneously) along one of the spanning

trees". This approach is also an example of a topology-aware optimization. We will discuss alternative MST optimizations in Sec 1.8.

It is worth noting that the often tedious manual configuration of topology or hierarchy has recently given way to useful APIs. This is an important and logical step, since topology in itself – other than performance – rarely changes. A useful API for accessing such topology information gaining popularity in high-performance libraries is *hwloc* [21]. It provides information for the hierarchy and topology within nodes. Automated solutions are emerging also for an inter-node topology, based for example on Infiniband APIs [13].

For further optimizations based on topology, sometimes it is possible to redesign a parallel application to make better use of topology-aware collectives [22].

1.7 Performance as Input

Topology is often sufficient to design an efficient collective algorithm. But there are two main cases when topology-aware communication can not be used. First, with increasing network complexity, the topology might be unknown or difficult to describe. If we book compute resources connected through complex networks, it is not possible anymore to easily represent the network. Second, the topology may only reflect partial performance information – e.g. the number of hops, which are related to latency, but unrelated to the bandwidth.

When performance is used to characterize network properties, it is common to use communication performance models. But such performance models face significant challenges. As described in Fig. 1.2, a number of layers exist for the communication library, and components of each layer impact the performance in some way. Therefore, it is unrealistic to look for "one fits all" model – its complexity and number of parameters would be overwhelming. Instead, it is reasonable to make the assumption that the low-level configuration of software and runtime is optimized, and to focus on the communication as something generic. In many cases, this ideal notion is not possible – e.g. misconfiguration of the underlying hardware or software (including MPI) is possible, and then incorporation of additional parameters is necessary. These technicalities are not the subject of this chapter.

A significant advantage of an accurate communication performance model is that it can be efficiently used for a wide range of optimized collective operations. The use of the model consists of two important phases (see Fig. 1.4):

- In a first phase, the model parameters are estimated.
- In a second phase, some form of optimization is targeted – either through prediction-based selection, or through a design of new algorithm.

For clarity, each time we introduce a model we will briefly address the above points of estimation, and how the models can be used on the example of a broadcast operation.

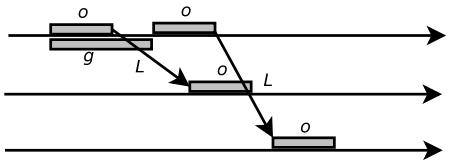

Figure 1.5 LogP example: Even basic predictions for collectives require consideration. Depending on o and g, completion can either take $(g + 2 * o + L)$ or $(3 * o + L)$.

1.7.1 Homogeneous Performance Models

The simple Hockney model [23] is the most comprehensive performance model of point-to-point communication, and is the common starting point for modelling collective algorithms. If the latency is α and the reciprocal value of the bandwidth is β, the time *T* to transfer a message of size *m* is given as:

$$T(m) = \alpha + \beta * m \tag{1.2}$$

The estimation of model parameters is trivially done with ping-pong benchmarks with different message sizes, and tools like NetPIPE [24] can be used.

As a simple example of predicting collectives, let us consider the binomial tree broadcast operation. It can be trivially predicted [10] as

$$T(m) = \lceil log(p) \rceil * (\alpha + m * \beta) \tag{1.3}$$

Numerous early efforts exist to design efficient collective operations on networks with heterogeneous links with the Hockney model. The common feature of all of them is the use of a heuristic to provide an efficient communication schedule rather than an optimal one. An intuitive idea is to use minimal spanning tree algorithms and modifications thereof, using the communication cost as edge property [12]. Other heuristics of trees with binomial or other structure also exist, for example considering overlap of communication [25]. Interestingly, it is not always the case that complex heuristics result in better efficiency – some evidence suggests that even a simple heuristic based on a fixed tree structure with reordering of processes can produce efficient communication trees [26].

A more advanced model is the LogP model [27], which has an upper bound *L* on latency, overhead *o*, gap per message *g*, and the number of processors *P*. The increase in parameters allows separate contributions for the network and processors at each machine – with *g* and *L* being network-dependent, and *o* being processor-dependent. And yet, a number of questions arise. While conceptually we can differentiate between the processor- and network-dependent contributions *o* and *g*, it is unclear where to draw the line between these contributions and what benchmarks should be performed in order to accurately estimate these parameters. This might be unproblematic for point-to-point communication, but is more important for collectives.

There are also other challenges to these parameters. The gap *g* and the overhead *o* parameters overlap in time. Consider for example a trivial broadcast between 3

processors as shown in Fig. 1.5. The prediction depends on the relation between g and o, since they overlap in time at each node.

Let us use this model to predict the familiar binomial tree broadcast for small messages. If we consider that for small message size *m* the gap *g* is small, we make the assumption $g < 2 * o + L$, resulting in [28]:

$$T = \lceil log(p) \rceil * (2 * o + L) \tag{1.4}$$

An extension of this model – LogGP model [29] – introduces the additional parameter gap per byte *G* for long messages. The extra parameter accounts for the overhead of sending one long message, where the prediction for a binomial tree broadcast is

$$T(m) = \lceil log(p) \rceil * (2 * o + L + G * (m - 1)) \tag{1.5}$$

The PLogP model [30], or the *Parametrized* LogP model, is another model related to LogP/LogGP model. It has the same 5 parameters, but they capture slightly different properties – we refer to the information provided in the original work for details. An important feature is that the parameters g and m are not constant, but functions – $g(m)$ and $o(m)$, and do not need to be linear, but only piecewise linear. This, in principle, allows to capture non-linear behaviour for varying message sizes, and such nonlinearities are sometimes observed in MPI (e.g. at the switch point between eager and rendezvous protocol).

The developers of the model provide a software tool for estimating its parameters. The original work introducing LogP/LogGP does not provide such software, and only micro benchmarks have been developed for these models. By using the provided PLogP software, its parameters can be evaluated, and can then in turn be translated into the LogP/LogGP parameters. The estimation of the parameters is much more complex than with simple models like Hockney. The authors claim that their model can be efficiently evaluated, because only $g(0)$ benchmarks need to saturate the network. However, this does not account for non linear behaviour of the network, when the cost of estimating the parameters increases significantly. In such cases, PLogP benchmarks are increased for more message sizes to extrapolate the non linear runtime more accurately using $g(m)$ and $o(m)$. For example, the authors acknowledge that $g(m > 0)$ with saturation of a link can take up to 17 times longer per link.

1.7.2 Heterogeneous Performance Models

The motivation for more complex performance models is that predictions for collective operations are not accurate based on traditional point-to-point models. Simply put, even if the individual contributing factors can be ignored for point-to-point predictions, these factors are needed when modelling collective communication. Performance models of heterogeneous networks can follow one of two approaches – either homogeneous communication models can be applied separately for each link, or new heterogeneous models can be introduced. To avoid the introduction of an

entirely new model, a simple first step is the slight modification of an existing model to represent at least some of the heterogeneity of the used platform. On the example of LogP, it has been recognized early that on sender and receiver side, contributions can differ for different nodes, and the constant overhead o can be subdivided into separate sender and receiver overheads o_s and o_r [31]. New heterogeneous communication models have been proposed [32, 33] with the idea to have more parameters which give more expressive power and, potentially, better accuracy. Parameters for constant and variable contributions of the network and sender and receiver are introduced. Here, we show the point-to-point prediction as given in [32]:

$$T(m) = S_c^{sender} + S_m^{sender} * m + X_c + X_m * m + R_c^{receiver} + R_m^{receiver} * m \quad (1.6)$$

In this formula, the components S_c, X_c and R_c are the constant parts of the send, transmission and receive costs respectively. m is the message size, with S_m, X_m, and R_m being the message dependent parts. Prediction formulas are provided for various collective operations – but with more expressiveness of different contributions to the runtime than homogeneous models. However, the prediction formulas are significantly more complex. If we consider the binomial tree broadcast, the prediction is:

$$T(m) = max\{T_{recv}^0(m), T_{recv}^1(m), \ldots, T_{recv}^{n-1}(m)\} \quad (1.7)$$

with

$$\begin{aligned} T_{recv}^i(m) = T_{recv}^{parent(i)} &+ childrank(parent(i), i) \\ &*(S_c^{parent(i)} + S_m^{parent(i)} * m) \\ &+X_m * m + X_c + R_m^I * m + R_c^i. \end{aligned} \quad (1.8)$$

$parent(i)$ is the parent of node i in the broadcast tree, and $childrank(parent(i), i)$ is the order, among its siblings, in which node i receives the message from its parent.

Unfortunately, the maximum operator cannot be eliminated, and a simpler prediction is impossible in such cases. The reason behind this is that it cannot be determined in advance which tree path is overall slower – and dominating the runtime – on heterogeneous networks.

1.7.3 Estimation of Parameters of Heterogeneous Performance Models

A significant challenge when increasing the number of parameters of heterogeneous models is the estimation phase. A model with a large number of parameters capturing separate contributions in communication is useless if the parameters cannot be practically established. After all, in real experiments it is the estimation phase that gives meaning to the model parameters – not an abstract description of what they should represent. There is good reason to be cautious – the presented model in previous section claims that two sets of experiments, ping-pong and consecutive sends, are sufficient to capture all 9 parameters. This is not plausible. For example, it is assumed that it is easy to estimate the component S_c in isolation, but how this can be

done within a node is not clear. Also, the constant network contribution is sometimes ignored during the estimation phase.

The proper estimation of model parameters is addressed in more recent work [33]. One important requirement is that n model parameters require the estimation phase to provide benchmarks which can be formulated as a system of linear equations with a single unique solution. It is difficult to design an estimation procedure providing such a system of equations. However, under certain assumptions it is feasible and is demonstrated for Ethernet clusters. For a number of collectives, the resulting predictions are shown to be more accurate than simple model predictions.

1.7.4 Other Performance Models

Performance models are not limited to capturing point-to-point or collective operations under "ideal" conditions. Another potential use case for such models is capturing contention and/or congestion. The topic is important, with the increase in networking and memory bus capacity lagging behind the increase of processing units like cores. We only give a short overview of some related work here. Simple approaches suggest introducing a factor to the familiar Hockney model, which slows down performance proportionally to the process number [34]. Other work in this direction introduces more complex extensions to LogP/LogGP to capture network contention [31]. The communication pattern of an application as well as the underlying network are analyzed. While more accurate for the given applications, the model uses a much larger number of parameters. There are also efforts for new contention models – for example, a simple experiment-oriented model which estimates penalty coefficients for Infiniband [35], or a model capturing the congestion on Ethernet clusters for gather operations [36].

1.8 Non-MPI Collective Algorithms for Heterogeneous Networks

While communication research in HPC increasingly focuses on the complex hierarchical structure of clusters – consider the presence of cores and accelerators within nodes – the related area of distributed computing is often overlooked. And yet algorithms and ideas from distributed computing have a strong background in optimizing communication for heterogeneous networks. Many of the solutions in this area do not suffer the limitations of MPI. First, we observe a multiple spanning tree algorithm which, unlike previously presented algorithms, is bandwidth-oriented and not topology-oriented. Then, we focus on adaptive algorithms of distributing data, which are orthogonal to the fixed schedule of communication in MPI. We show two useful applications of such algorithms – efficient alternatives to MPI collectives, or new ways of designing performance models.

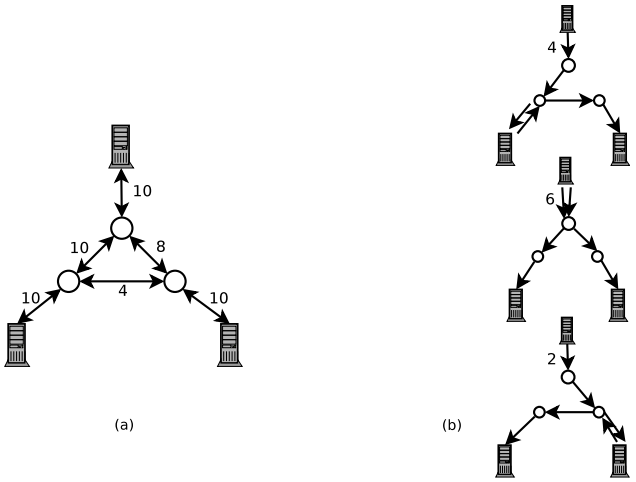

Figure 1.6 A network model [37] and multiple spanning trees for improving throughput. (a) Available bandwidth for each independent link. (b) Three possible spanning trees and predicted throughput for each.

1.8.1 Optimal Solutions with Multiple Spanning Trees

A recent work in distributed computing [37] uses MST without the constraints of MPI to implement efficient collective operations. As discussed earlier, the only known MPI solution builds multiple trees following the known network topology.

The problem of finding an optimal set of communication trees is very complex – n^{n-2} different trees exist alone for n hosts according to Cayley's formula [38]. Naturally, the use of heuristics is required to find a good solution in reasonable time, and the above work uses following steps:

- Start with a network model to describe the properties of a heterogeneous network – an example is given in Fig. 1.6.
- Generate a good (but not optimal) set of trees for given network using a heuristic algorithm [39].
- Translate network model and set of trees into a set of constraints.
- Use linear programming algorithm to find maximum throughput.

It is crucial that a heuristic algorithm provides spanning trees for the given problem. These trees can be seen merely as "sensible suggestions". As any heuristic, they have their limitations, which are discussed in detail in related work. In the last steps, the maximum throughput provided by the linear programming is not used for the algorithm design. It serves as a measure of the efficiency of the designed algorithm.

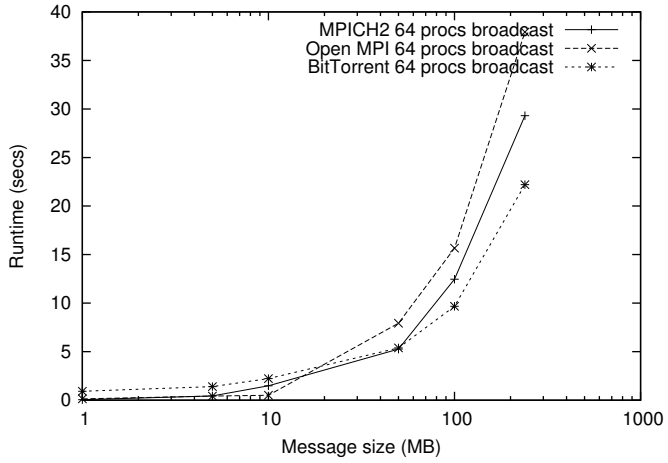

Figure 1.7 Experiments on a hierarchy of 3 Ethernet clusters within a site in Bordeaux. BitTorrent broadcasts outperform MPI broadcasts for very large messages.

1.8.2 Adaptive Algorithms for Efficient Large-Message Transfer

In middleware like MPI, messages are communicated through a fixed schedule of point-to-point operations. The messages are pushed across the network following this schedule, hence this class of algorithms is called *sender-initiated multicasts*. The orthogonal class of collective operations is called *receiver-initiated multicasts*. In this case, receivers pull the data from senders. A very popular protocol in this category is BitTorrent [40]. While initially more popular in distributed and peer-to-peer computing, BitTorrent has shown promising performance even in the context of high performance networks with some level of heterogeneity. Experiments on different emulated wide-area networks show that BitTorrent-inspired protocols can perform very well [41]. More surprisingly, even on a hierarchy of clusters within a single computing site, broadcasts of very large messages using the original BitTorrent protocol can often outperform state-of-the-art MPI algorithms [42]. Fig. 1.7 shows that for message sizes in the range of Megabytes, BitTorrent can outperform state-of-the-art MPI implementations on a hierarchy of Ethernet clusters.

While the analysis of BitTorrent communication is not trivial, one fundamental feature of receiver-initiated multicasts is that data can be propagated adaptively. To support this, BitTorrent opens multiple TCP channels in parallel. Early work [43] has also shown that in this case the throughput approaches the maximum. Therefore, there has recently been strong evidence that dynamic and adaptive scheduling of point-to-point communication can be efficient when transferring large data volumes in heterogeneous networks.

1.8.3 Network Models Inspired by BitTorrent

One interesting application of adaptive and dynamic multicast protocols (e.g. BitTorrent) which deserves increased research effort in the future is the generation of a network model. To some extent, this idea has been demonstrated [37] by allowing dynamic stealing of bandwidth within clusters while providing the overall network topology.

Another entirely BitTorrent-based solution is proposed in a network tomography method to identify logical clusters without any a-priori knowledge of the network [44]. Network tomography, even though an area of research more popular in Internets and unknown networks, bears a close resemblance to the performance models used in high performance computing. The general goal in this area is the logical reconstruction of a network using only end-to-end measurements. Network tomography follows two distinct phases:

- Measurement phase – end-to-end measurements with some level of noise
- Reconstruction phase – remove noise through statistical methods

In its goal, the method is quite similar to the estimation of model parameters. But instead of isolated point-to-point experiments, adaptive BitTorrent-based multicasts are used for network discovery. The multicasts are repeated a number of times until the network properties are reliably reconstructed. In the end, a logical view of the network in the form of bandwidth clusters is built. The resulting accurate clustering of geographically distributed sites according to their bandwidth is shown in Fig. 1.8. The provided clustering can be used as a topology input for any topology-aware collective algorithm.

Furthermore, the presented tomography seems even more suitable for performance-aware collective communication. As shown in Fig. 1.9, such algorithms can provide an entirely performance-based network model, and have the potential to replace the traditional performance measurement procedures. The two types of performance measurements differ strongly. Traditional measurement procedures typically use point-to-point operations to reconstruct link model parameters. Each of these measurements needs to be repeated a number of times. On the other side, the measurement procedures inspired e.g. by the BitTorrent protocol can be very efficient, and can be entirely based on collective operations like broadcasts. In a separate phase, the statistical algorithm removes noise and randomness in the measurements till the accuracy levels are good enough. It is significant that collective operations can be used instead of point-to-point operations. This allows for an increased accuracy of the model when targeting predictions of collective operations. For example, it is impossible for point-to-point operations to detect the presence of bottlenecks in most scenarios. However, the use of flexible measurement procedures can detect such bottlenecks [44].

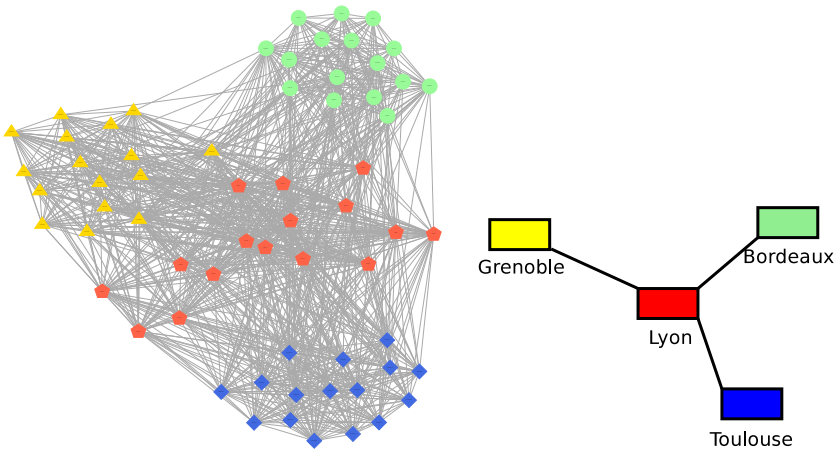

Figure 1.8 Network tomography as proposed in [44]. (a) Reconstructed bandwidth clusters. (b) Underlying physical network.

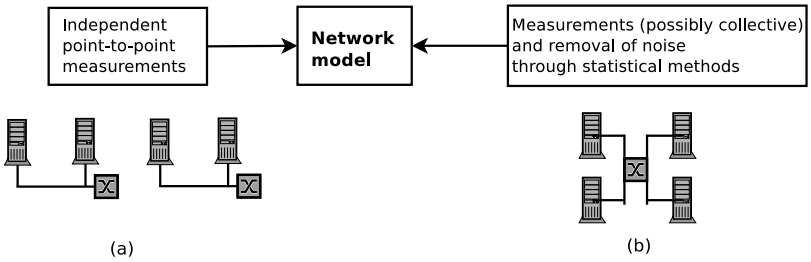

Figure 1.9 Performance-aware network models. (a) Traditionally, a number of isolated point-to-point experiments are used in HPC. (b) Inspired by network tomography, measurements based on collective operations can be performed.

1.9 Conclusion

This work reviewed the main generic optimization techniques for collective communication. With increasingly heterogeneous networks, empirical approaches to optimizing communication become unfeasible due to the exponential growth of the already huge test space. Therefore, we need some sort of a network model. Such a network model can be based on topology or performance. Topology-aware collectives are a relatively straight-forward and popular approach to optimization. They offer solid performance gains in case we know the network topology in advance. More advanced network models are based on the performance. In this case, performance communication models are used. While these models strive to capture the network heterogeneity, they are difficult to use. For example, heterogeneous models capturing various contributions produce complex prediction formulas. In addition to that, their parameters are difficult to estimate. More advanced models would be even more complicated to use, and that limits their practical importance. This means that one of the classic cases of using models on homogeneous networks – prediction-based selection of optimal algorithms – is difficult to apply with advanced models for heterogeneous and hierarchical networks. From a scientific point of view, this outcome is not satisfactory. After all, accurate predictions are the only scientific validation of any model. But if we take a more practical approach, even for a "relatively accurate" model, there is a significant potential for designing efficient performance-aware algorithms. The simple Hockney model – even though proven to not always provide accurate predictions – has been successfully used on heterogeneous networks. It has been observed that minimal spanning trees based on the per-link Hockney model provide efficient broadcast for small messages. For larger messages, the same approach can be used successfully for binomial trees broadcasts. For very large messages, receiver-initiated multicasts are gaining popularity in the HPC domain. The adaptive nature of these algorithms makes them suitable even for very complex networks.

We are grateful to COST Action IC0805 "Open European Network for High Performance Computing on Complex Environments" for their support.

REFERENCES

1. L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, *et al.*, *ScaLAPACK Users' Guide*, vol. 4. Society for Industrial and Applied Mathematics, 1987.
2. J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
3. D. Wadsworth and Z. Chen, "Performance of MPI broadcast algorithms," in *International Parallel and Distributed Processing Symposium. IPDPS 2008.*, pp. 1–7, 2008.
4. W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message passing interface*, vol. 1. MIT press, 1999.
5. L. Hablot, O. Gluck, J.-C. Mignot, S. Genaud, and P.-B. Primet, "Comparison and tuning of MPI implementations in a grid context," in *IEEE International Conference on Cluster Computing*, pp. 458 –463, September 2007.
6. A. Mamidala, A. Vishnu, and D. Panda, "Efficient shared memory and RDMA based design for MPI_Allgather over InfiniBand," *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pp. 66–75, 2006.
7. J. Liu, J. Wu, and D. K. Panda, "High performance RDMA-based MPI implementation over InfiniBand," *International Journal of Parallel Programming*, vol. 32, pp. 167–198, June 2004.
8. T. Hoefler, C. Siebert, and W. Rehm, "A practically constant-time MPI broadcast algorithm for large-scale InfiniBand clusters with multicast," in *International Parallel and Distributed Processing Symposium. IPDPS 2007*, pp. 1 –8, March 2007.

9. E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, (Budapest, Hungary), pp. 97–104, September 2004.

10. R. Thakur and R. Rabenseifner, "Optimization of collective communication operations in MPICH," *International Journal of High Performance Computing Applications*, vol. 19, pp. 49–66, 2005.

11. J. Pjesivac-Grbovic, *Towards Automatic and Adaptive Optimizations of MPI Collective Operations*. PhD thesis, The University of Tennessee, Knoxville, 2007.

12. P. B. Bhat, C. Raghavendra, and V. K. Prasanna, "Efficient collective communication in distributed heterogeneous systems," *Journal of Parallel and Distributed Computing*, vol. 63, no. 3, pp. 251 – 263, 2003.

13. H. Subramoni, K. Kandalla, J. Vienne, S. Sur, B. Barth, K. Tomko, R. Mclay, K. Schulz, and D. K. Panda, "Design and evaluation of network topology-/speed- aware broadcast algorithms for InfiniBand clusters," in *IEEE International Conference on Cluster Computing*, pp. 317–325, 2011.

14. M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *12th international workshop on network and operating systems support for digital audio and video*, pp. 127–136, ACM, 2002.

15. T. Kielmann, H. E. Bal, and S. Gorlatch, "Bandwidth-efficient collective communication for clustered wide area systems," in *Workshops of International Parallel and Distributed Processing Symposium. IPDPS 2000*, pp. 492–499, 2000.

16. E. Gabriel, M. Resch, T. Beisel, and R. Keller, "Distributed computing in a heterogeneous computing environment," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface* (V. Alexandrov and J. Dongarra, eds.), vol. 1497 of *LNCS*, pp. 180–187, Springer Berlin / Heidelberg, 1998. 10.1007/BFb0056574.

17. R. Rabenseifner, G. Hager, and G. Jost, "Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes," in *17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, 2009*, pp. 427–436, IEEE, 2009.

18. J. Ladd, M. G. Venkata, R. Graham, and P. Shamis, "Analyzing the effects of multicore architectures and on-host communication characteristics on collective communications," in *40th International Conference on Parallel Processing Workshops. ICPPW 2011*, pp. 406–415, 2011.

19. T. Ma, G. Bosilca, A. Bouteiller, B. Goglin, J. M. Squyres, and J. J. Dongarra, "Kernel assisted collective intra-node MPI communication among multi-core and many-core CPUs," in *International Conference on Parallel Processing. ICPP 2011*, pp. 532–541, 2011.

20. G. Almási, P. Heidelberger, C. J. Archer, X. Martorell, C. C. Erway, J. E. Moreira, B. Steinmacher-Burow, and Y. Zheng, "Optimization of MPI collective communication on BlueGene/L systems," in *International Conference on Supercomputing. ICS 2005*, pp. 253–262, ACM, 2005.

21. F. Broquedis, J. Clet-Ortega, S. Moreaud, N. Furmento, B. Goglin, G. Mercier, S. Thibault, and R. Namyst, "hwloc: A generic framework for managing hardware

affinities in HPC applications," in *Euromicro International Conference on Parallel, Distributed and Network-Based Processing. PDP2010*, pp. 180–186, 2010.

22. C. Coti, T. Herault, and F. Cappello, "MPI applications on grids: A topology aware approach," in *The 15th International European Conference on Parallel and Distributed Computing. Euro-Par 2009*, vol. 5704 of *LNCS*, pp. 466–477, Springer Berlin / Heidelberg, 2009.
23. R. W. Hockney, "The communication challenge for MPP: Intel Paragon and Meiko CS-2," *Parallel Computing*, vol. 20, pp. 389–398, Mar. 1994.
24. Q. O. Snell, A. R. Mikler, and J. L. Gustafson, "NetPIPE: A network protocol independent performance evaluator," in *IASTED International Conference on Intelligent Information Management and Systems*, 1996.
25. J. Hatta and S. Shibusawa, "Scheduling algorithms for efficient gather operations in distributed heterogeneous systems," in *International Conference on Parallel Processing Workshops. ICPPW 2000*, pp. 173–180, 2000.
26. K. Dichev, V. Rychkov, and A. Lastovetsky, "Two Algorithms of Irregular Scatter/Gather Operations for Heterogeneous Platforms," in *17th European MPI users' group meeting conference on Recent advances in the message passing interface*, vol. 6305 of *LNCS*, (Stuttgart, Germany), pp. 289–293, Sept. 2010.
27. D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a realistic model of parallel computation," *Fourth ACM SIGPLAN symposium on principles in practices of parallel programming*, vol. 28, pp. 1–12, July 1993.
28. T. Hoefler, L. Cerquetti, T. Mehlan, F. Mietke, and W. Rehm, "A practical approach to the rating of barrier algorithms using the LogP model and Open-MPI," in *International Conference on Parallel Processing Workshops. ICPPW 2005*, pp. 562–569, June 2005.
29. A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, "LogGP: incorporating long messages into the LogP model – one step closer towards a realistic model for parallel computation," in *Proceedings of the seventh annual ACM symposium on parallel algorithms and architectures*, pp. 95–105, ACM, 1995.
30. T. Kielmann, H. Bal, and K. Verstoep, "Fast measurement of LogP parameters for message passing platforms," *Workshops on International Parallel and Distributed Processing Symposium. IPDPS 2000*, pp. 1176–1183, 2000.
31. C. Moritz and M. Frank, "LoGPG: Modeling network contention in message-passing programs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 4, pp. 404–415, 2001.
32. M. Banikazemi, J. Sampathkumar, S. Prabhu, D. K. Panda, and P. Sadayappan, "Communication modeling of heterogeneous networks of workstations for performance characterization of collective operations," in *8th Heterogeneous Computing Workshop*, HCW '99, (Washington, DC, USA), pp. 125–133, IEEE Computer Society, 1999.
33. A. Lastovetsky, V. Rychkov, and M. O'Flynn, "Accurate heterogeneous communication models and a software tool for their efficient estimation," *International Journal of High Performance Computing Applications*, vol. 24, pp. 34–48, 2010.
34. L. Steffenel, "Modeling network contention effects on all-to-all operations," in *IEEE International Conference on Cluster Computing, 2006*, pp. 1–10, IEEE, 2006.

35. M. Martinasso and J.-F. Mehaut, "A contention-aware performance model for HPC-based networks: A case study of the InfiniBand network," in *17th International European Conference on Parallel and Distributed Computing. Euro-Par 2011*, pp. 91–102, 2011.

36. A. Lastovetsky and M. O'Flynn, "A performance model of many-to-one collective communications for parallel computing," in *International Parallel and Distributed Processing Symposium. IPDPS 2007*, pp. 1–8, IEEE, 2007.

37. M. den Burger, *High-throughput multicast communication for grid applications*. PhD thesis, Vrije Universiteit Amsterdam, 2009.

38. A. Cayley, "A theorem on trees," *Quarterly Journal of Mathematics*, vol. 23, no. 376-378, p. 69, 1889.

39. R. Izmailov, S. Ganguly, and N. Tu, "Fast parallel file replication in data grid," in *Future of Grid Data Environments workshop (GGF-10)*, 2004.

40. B. Cohen, "Incentives build robustness in BitTorrent," in *1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

41. M. den Burger and T. Kielmann, "Collective receiver-initiated multicast for grid applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 231 –244, February 2011.

42. K. Dichev and A. Lastovetsky, "MPI vs BitTorrent : Switching between large-message broadcast algorithms in the presence of bottleneck links," in *10th International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms. HeteroPar 2012*, (Rhodes Island, Greece), pp. 185–195, LNCS, 7640, Springer, August 2012.

43. P. Rodriguez and E. W. Biersack, "Dynamic parallel access to replicated content in the Internet," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 455–465, Aug. 2002.

44. K. Dichev, F. Reid, and A. Lastovetsky, "Efficient and reliable network tomography in heterogeneous networks using BitTorrent broadcasts and clustering algorithms," in *ACM/IEEE International Conference on High Performance Computing, Networking, Storage and Analysis. SC'12*, (Salt Lake City, UT, USA), 2012.