A framework for adaptive collective communications for heterogeneous hierarchical computing systems

Luiz Angelo Steffenel, Grégory Mounié

A Framework for Adaptive Collective Communications for Heterogeneous Hierarchical Computing Systems

Luiz Angelo Steffenel[1], Grégory Mounié[2]

[1] *Université Nancy 2/LORIA, Nancy, France*

[2] *Laboratoire ID-IMAG, Grenoble, France*

Abstract

Collective communication operations are widely used in MPI applications and play an important role in their performance. However, the network heterogeneity inherent to grid environments represent a great challenge to develop efficient high performance computing applications. In this work we propose a generic framework based on communication models and adaptive techniques for dealing with collective communication patterns on grid platforms. Toward this goal, we address the hierarchical organization of the grid, selecting the most efficient communication algorithms at each network level. Our framework is also adaptive to grid load dynamics since it considers transient network characteristics for dividing the nodes into clusters. Our experiments with the broadcast operation on a real-grid setup indicate that an adaptive framework allows significant performance improvements on MPI collective communications.

Key words: Grid computing; Performance modeling; Adaptive techniques; Polyalgorithms; Collective communication; MPI

1 Introduction

In the last years, there was a huge development in the field of parallel and distributed processing, especially at the architectural level leading to a wide variety of execution supports. The major innovation was the phenomenal spread of architectures like clusters and grids. These platforms represent a reasonable alternative to traditional parallel machines and have become the most cost-effective computing supports for solving a large range of high performance computing applications due the good cost/performance ratio that they provide. However, the introduction of such parallel systems has a major impact on the design of efficient parallel algorithms. Indeed, new characteristics have to be taken into account including scalability and portability. Moreover, such parallel systems are often upgraded with new generation of processors and network technologies. For instance, adaptability becomes crucial because of the frequent changes of the system hardware. These different elements require to revise the classical parallel algorithms which consider only regular architectures with static configurations and to propose new approaches.

Our objective in this work is to propose a generic framework based on communication models and scheduling techniques to deal with communication scheduling in heterogeneous environments such as computational grids. More precisely, this paper proposes a communication schedule methodology with two adaptation levels. At the first level we proceed at the intra-cluster level, by determining the most efficient communication algorithm from a set of well known algorithms from the literature. At a second level, our framework de-

Email address: [1]Luiz-Angelo.Steffenel@univ-nancy2.fr,
[2]Gregory.Mounie@imag.fr (Luiz Angelo Steffenel[1], Grégory Mounié[2]).

termines an inter-cluster communication schedule that minimizes the overall execution time of a collective communication. Therefore, our framework differs significantly from other works, as existing adaptive approaches presented in the literature [1,2,3] proceed by simply scheduling communications at the inter-cluster level, i.e., long-distance links. At the other side, works like [4,5,6] only try to minimize the execution time of collective communication operations in the context of intra-cluster environments. To the best of our knowledge, our framework provides the first general methodology to automatically associate efficient intra-cluster algorithms with inter-cluster communication heuristics, reducing the overall execution time of a collective communication.

The remainder of the paper is organized as follows. We begin in Section 2 by describing our assumptions for the communication environment. In Section 3 we first define the concept of polyalgorithm, presenting our framework for adaptive communications and detailing its components. Section 4 describes the platform partitioning phase, where we organize the grid into homogeneous logical cluster. Hence, in Section 5 we present a case study where we apply the second part of our framework for the development of a grid-aware `MPI_BCast` communication operation. To validate the framework contributions, we conduct both practical experiments on a grid environment (Section 6) and numerical simulations (Section 7). These results concern both the evaluation of the optimization overhead and the scalability of the algorithms, proving the interest of this work. Finally, Section 8 concludes the paper and discusses some perspectives to extend this work.

2 Description of the Environment

Heterogeneity Model: We assume a generic platform composed by heterogeneous clusters as described in [7]. The platform studied enjoys heterogeneity along three orthogonal axes: ***(i)*** the processors that populate the clusters may differ in computational powers, even within the same cluster; ***(ii)*** the clusters are organized hierarchically and are interconnected via a hierarchy of networks of possibly differing latencies and bandwidths. At the level of physical clusters, the interconnection networks are assumed to be heterogeneous; ***(iii)*** the clusters at each level of the hierarchy may differ in sizes.

Communication Model: We assume that the network is fully connected. The links between pairs of processes are bidirectional, and each process can transmit data on at most one link and receive data on at most one link at any given time. This model is well-known in the literature as *1-port full-duplex*.

Transmission Model: The literature contains several parallel communication models [8,9,10,11,12,3]. These models differ on the computational and network assumptions, such as latency, heterogeneity, network contention, etc. In this work we adopted the *parameterized LogP* model (*pLogP*) [3]. Our choice on the *pLogP* model comes from the fact that we can experience different transmission rates according to the message size, as a consequence of transport protocols and hardware policies. Hence, all along this paper we shall use $\boldsymbol{L}$ as the communication latency between two nodes, $\boldsymbol{P}$ as the number of nodes and $\boldsymbol{g(m)}$ for the gap of a message of size m. The gap of a message m represents the time required to transmit a message through the network (excluding the latency), which is inversely proportional to the bandwidth of the link. In the case of message segmentation, the segment size s of the message m is a

multiple of the size of the basic datatype to be transmitted, and it splits the initial message m into k segments.

3 An Adaptive Framework for Grid-Aware Communications

In this section, we describe our framework for adaptive communication scheduling in an execution environment characterized by its heterogeneity and its hierarchical organization. We consider a grid environment composed by different clusters $\mathcal{C}_1$ to $\mathcal{C}_n$ with respectively n_1, n_2, ..., n_n nodes. A wide-area network, called a backbone, interconnects these clusters. We assume that a cluster use the same network card to communicate to one of its node or to a node of another cluster, although each cluster may use different network technologies (Fast Ethernet, Gigabit Ethernet, Myrinet, etc.). Based on that topology inter-cluster communications are never faster than communication within a cluster.

Most MPI libraries (LAM-MPI, OpenMPI, MPICH2, etc.) implement collective communications assuming that all the nodes are on the same clusters, which means that all communications have the same weight. However, in our case, some messages are transferred within a cluster (from a node of $\mathcal{C}_1$ to a node of $\mathcal{C}_1$, for example, or between the two clusters. In the first case, bandwidth and latency are faster than in the second case. Therefore, we need to associate different tools to model the overall performance. We assume that communication performances can be predicted based on communication cost models (for instance, the pLogP model [3]) and benchmarks on the real system.

An overview of the framework is sketched in Figure 1. Since the target system may experience heterogeneity at different levels (computing performance, net-

work capacity, etc), it is too difficult to manage the entire platform towards a high performance computing. One way to circumvent this problem is to subdivide the network in homogeneous subnets (or logical clusters), handling each cluster individually to subsequently aggregate them at the grid level. Therefore, the framework is separated in two successive phases. During the first one, we aim to partition the execution platform into subnets with homogeneous characteristics. Then, when executing the second phase, we determine for each subnet (i.e., for each cluster) the communication algorithm that performs better in that cluster. Indeed, using pLogP, we are able to predict the communication performance on each different cluster, allowing us to compare different communications algorithms. In the same way, pLogP is used to define efficient wide-area communication schedules adapted to a heterogeneous grid environment.

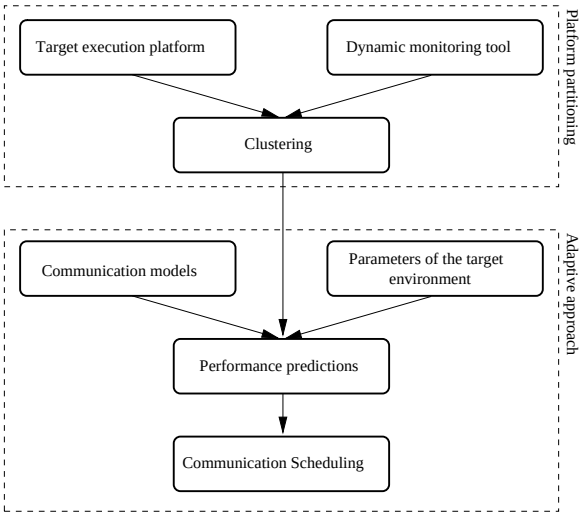


Figure 1. Conceptual framework of the adaptive mechanism

Once the platform is partitioned in separated homogeneous hierarchical clusters we determine, for each cluster, an algorithm which performs better in that network environment. Actually, we compare the expected performance of different algorithms from the literature (each algorithm being previously mod-

eled with *pLogP*), in terms of the size of data to be transmitted, the network characteristics and the number of nodes.

Through the analysis of the inter-clusters and intra-cluster performance predictions we are able to define a communication schedule that minimizes the overall execution time. Once again we can compare different schedule policies (heuristics), which are chosen according to their estimated termination time. The framework allows, indeed, to implement scheduling heuristics that act on different communication levels, be it at inter-cluster level (mostly appropriate to collective operations like *broadcast* [2] and *reduce* [13]) or at node-to-node level (for operations such as the *all-to-all* [4]).

4 Platform Partition

We propose a method to automatically discover the network topology, allowing the construction of optimized multilevel collective operations. We prefer automatic topology discovery instead of a predefined topology because if there are hidden heterogeneities inside a cluster, they may interfere with the communication and induce a non negligible imprecision in the models. The automatic discovery we propose should be done in two phases: the first phase collects reachability data from different networks. The second phase, executed at the application start-up, subdivides the networks in homogeneous logical clusters and finally acquires pLogP parameters to model collective communications.
Several specialized tools can be used to gather connectivity information through network monitoring. These tools may acquire data from direct probing, like NWS [14], from SNMP queries to network equipments, like REMOS [15], or even combine both approaches, like TopoMon [16]. NWS seems to be the best candidate to our needs: as a *de facto* standard in the grid community,

NWS can be configured to provide information like communication latency, throughput, CPU load and available memory. For instance, we may identify groups of machines with similar communication characteristics using latency and throughput data obtained from NWS.

4.1 Clustering

One reason to construct logical clusters is that even machines in the same network may behave differently, in spite of their physical location. Indeed, such differences introduce undesirable heterogeneities that may invalidate the performance models used to optimize collective communications. For instance, we are interested in grouping machines with similar performances into "logical clusters" to reduce the scheduling complexity.

Clustering may be performed according different approaches. The most known approach try to define a spanning tree such that each node connects to the closest node in the network. This approach can be implemented through agglomerative construction of the spanning tree from a given parameter, but also can be implemented by pruning the full interconnection graph [17]. Another approach consists on defining a "closeness" parameter ρ, which indicates the maximum variance among nodes in the same group. In the specific case of our work, the last technique seems to be the most appropriate, as at this point we are simply interested on the definition of homogeneous clusters.

Therefore, we may consider a weighted digraph $dG(V, E)$ of order n with $V = \{p_0, ..., p_{n-1}\}$ to represent our network. In this digraph, the vertices represent the process nodes and the edges represent the link between two nodes. An integer $w_{i,j}$ is associated with each edge $E_{i,j}$, representing the distance between nodes p_i and p_j (communication latency, for example), and we define ρ as the

maximal distance variation between two nodes in the same cluster. Hence, this digraph corresponds to the distance matrix M defined by:

$$M = \begin{cases} w_{i,j} & if\ there\ is\ a\ local\ link\ between\ \{i,j\} \\ 0 & otherwise \end{cases} \tag{1}$$

For instance, a trivial algorithm to solve this problem initially sorts the outgoing edges from each node in increasing order of their weights. By proceeding from the smallest weighted edge $w_{x,y}$, we define an initial group $\{x, y\}$. At each step we select a candidate node a and compare its distance to any node within a group S. If distance does not vary more than ρ, node a can be included in group S. Otherwise, if node a does not fit into any existent group, it becomes the first node of a new group S'. The algorithm terminates after all outgoing edges have been evaluated. Indeed, this algorithm can be defined by the expression:

$$\forall x, \forall y \in S, x \neq y,\ a \in S \Rightarrow |w(a,x) - w(x,y)| \leq \rho \tag{2}$$

Because we need to compare node a to each node from group S, this algorithm executes in $O(N^2)$ steps. Therefore, Lowekamp [18] presented a greedy algorithm, which was implemented within the ECO library and is also adopted in our work. More specifically, Lowekamp's algorithm compares a candidate node a with the smallest edge $wmin$ within a group S. This algorithm, which requires only $O(N)$ steps, corresponds to the following expression:

$$\forall x, \forall y \in S, x \neq y,\ a \in S \Rightarrow |w(a,x) - wmin(S)| \leq \rho \tag{3}$$

Although the distance between two nodes can be expressed with the help of different parameters (latency, bandwidth, hops, etc.), we considered latency as

the main parameter to be evaluated in our topology discovery implementation. Indeed, latency has proved to be sufficiently accurate to distinguish nodes in connected to different switches in a local network. Further, latency can be easily measured in a wide area network without disturbing the ongoing traffic, contrarily to a bandwidth measurement.

In addition, the topology discovery process may be detached from the application, minimizing the overhead in the application performance. Indeed, the most expensive part of the process consists on contacting each other node to compose a distance matrix, while the clustering part is quite simple. An offline topology discovery is recommended for such applications, following the principles used by MagPIe [2], which reads the topology description from a file. A *daemon* process may conduct regular updates on the description file, inducing almost no overhead to the application.

4.2 Efficient Acquisition of pLogP Parameters

Once identifying the logical cluster organization of our grid, we must other network parameters such as the bandwidth (or the gap, for the pLogP model). Hopefully, there is no need to execute $n(n-1)$ pLogP measures, one for each possible interconnection. Using the topology information we can get pLogP parameters in an efficient way by considering a single process to represent each cluster. As one single measure may represent the entire subnet, the total number of pLogP measures is fairly reduced. If we sum up the measures to obtain the parameters for the inter-clusters connections, we shall execute at most $C \times (C-1) + C$ experiments, where C means the number of cluster. Further, if we consider symmetric links, only half of the probes are need, minimizing the interference on the network.

5 Case Study - Broadcast Operations

5.1 Intra-cluster Communication Strategy Selection

With Broadcast, a single process, called *root,* sends the same message of size m to all other $(P-1)$ processes. Classical implementations of the Broadcast operation rely on d-ary trees characterized by two parameters, d and h, where d is the maximum number of successors a node can have, and h is the height of the tree, the longest path from the root to any of the tree leaves. Therefore, most MPI implementations rely on the Binomial Tree broadcast, an algorithm that is optimal on homogeneous networks if we assume that messages cannot be segmented.

Barnett *et al.* [19] demonstrate, however, that better performances can be obtained if we compose a pipeline among the processes. This strategy benefits from message segmentation, as recent works indicate [3][20]. In a Segmented Chain Broadcast, the transmission of a segment k overlaps with the reception of segment $k+1$, reducing the overall time.

To fully benefit from the pipeline effort, the segment size must be chosen according to the network environment. Indeed, too small messages pay more for their headers than for their content, while too large messages do not explore enough the pipeline. Therefore, an efficient method to identify an adequate segment size s consists in searching through all values of s where $s = m/2^i, i \in [0 \dots log_2 m]$ such that s minimizes the predicted performance of the communication operation. To refine the search, we can also apply some heuristics like local hill-climbing, as proposed by Kielmann *et al.* [3].

In our work we developed the communication models for some current techniques, which are presented on Table 1. From these models, we are able to easily determine the broadcast algorithm that best performs on each cluster. Indeed, using the pLogP parameters obtained during the topology discovery phase, we can predict the broadcast execution time with a good accuracy and select the fastest algorithm for each cluster, as we presented in [21].

Table 1
Some communication models for the *Broadcast* operation

Algorithm	**Communication Cost**
Flat Tree	$L + (P-1) \times g(m)$
Segmented Flat Tree	$L + (P-1) \times (g(s) \times k)$
Chain	$(P-1) \times (g(m) + L)$
Segmented Chain (Pipeline)	$(P-1) \times (g(s) + L) + (g(s) \times (k-1))$
Binary Tree	$\leq \lceil log_2 P \rceil \times (2 \times g(m) + L)$
Binomial Tree	$\lceil log_2 P \rceil \times L + \lfloor log_2 P \rfloor \times g(m)$
Segmented Binomial Tree	$\lceil log_2 P \rceil \times L + \lfloor log_2 P \rfloor \times g(s) \times k$
k-chain [22] with a degree d	$(d + \lceil \frac{P-(2^d+1)}{(2^d+1)} \rceil) \times (g(s) + L) + (g(s) \times (k-1))$
Scatter/Collection [23]	$(log_2 P + P - 1) \times L + 2 \times (\frac{p-1}{p}) \times g(m)$

5.2 Grid-aware Communication Scheduling

The literature presents several works that aim to optimize collective communications in heterogeneous environments. While some works just focus on the search for the best broadcast tree of a network [17], most authors such as Banikazemi [24], Bhat [4], Liu [5], Park [25], Mateescu [26] and Vorakosit [27] try to generate optimal broadcast trees according to a given *root* process.

Unfortunately, most of these works were designed for small-scale systems. One of the first works on collective communication for grid systems was the ECO library proposed by Lowekamp [18], where machines are grouped according to

their location. Later, the same principle was used by the MPI library MagPIe [2], where processes are hierarchically organized in two levels with the objective to minimize the exchange of wide-area messages.

A common characteristic of these two implementations is that only *inter-cluster* communications are optimized. Hence, to improve communication performances, we must also improve *inter-cluster* communications. One of the first works to address this problem was presented by Karonis [1], who defined a multilevel hierarchy that allows communication overlapping between different levels. While this structure on multiple levels allows a performance improvement, it relies on flat trees to disseminate messages between two wide area levels, the same strategy as ECO or MagPIe. It is important to note that a flat tree is far from being optimal on heterogeneous systems. Because the exhaustive search of the optimal tree is expensive, we decided to employ different optimization heuristics. For instance, in this work we explore a different approach to improve communication efficiency.

We consider that wide-area latency is no longer the single parameter that may contribute to the broadcast time. Indeed, the communication cost inside a cluster may represent an important factor to the overall completion time. For example, let us consider two clusters from Grid'5000, one located at Orsay and the other at Grenoble (approximately 700km from each other). The transmission of 1MB between these clusters with a private backbone of 1Gbit/s needs 350 milliseconds. At the same time, a binomial-tree broadcast with 50 nodes interconnected by a Gigabit Ethernet network for the same message size requires almost 600 milliseconds. Ignoring the intra-cluster time may lead to inefficient communication schedules if the clusters are not well balanced.

Hence, we propose a smart schedule of wide-area collective communications, which considers both *inter* and *intra-cluster* times to minimize makespan.

5.2.1 Description Formalism and Performance Model

To describe the heuristics presented in the next sections, we use a formalism similar to the one used by Bhat [4]. We consider that clusters are divided in two sets, **A** and **B**. The set **A** contains the clusters that already received a message (i.e., the coordinator of the cluster receives it). In set **B** we found all clusters that shall receive the message. At each communication round, two clusters are chosen from sets **A** (a sender) and **B** (a receiver). After communicating, the receiver cluster is transferred to set **A**. When a coordinator does not participate in any other *inter-cluster* communication, it can finally broadcast the message inside its cluster.

5.2.2 Baseline Algorithm - Flat Tree

This strategy uses a flat tree to send messages at the *inter-cluster* level, i.e., the *root* process sends the message to the coordinators of all other clusters, in a sequential way. Formally, the *root* process, which belongs to the set **A**, chooses a different destination among the clusters in set **B** at each communication round (with a complexity $O(n)$). Once a cluster coordinator receives a message, it broadcasts the message inside the cluster using a binomial tree technique. Although easy to implement, this strategy is far from being optimized as the diffusion of messages does not take into account the performance of different clusters or the interconnection speed.

5.2.3 Fastest Edge First - FEF

Proposed by Bhat *et al.* [4], the *Fastest Edge First* heuristic considers that each link between two different processes i and j, corresponds to an edge with

weight T_{ij}. Usually, this edge weight T_{ij} corresponds to the communication latency between the processes. To schedule the broadcast communications in a heterogeneous environment, the FEF heuristics order nodes from the set **A** according to their smallest outgoing edge weight. Once this smallest edge is selected, it implicitly designates the sender and receiver processes. When a receiver is chosen, it is transferred from set **B** to set **A**, and the minimal outgoing edge list is updated. Hence, this technique maximizes the number of available senders that can proceed in parallel for a complexity of $O(n^2)$.

5.2.4 Early Completion Edge First - ECEF

In the previous heuristics, once the receiver is assigned, it is immediately transferred to the set **A** and can take part in the next communication round. This model is not realistic as communication delays may prevent a receiver process from having the message immediately. The *Early Completion Edge First* heuristic [4] keeps an account of the moment in which a message becomes available to the processes in the set **A**. This way, a *Ready Time* (RT_i) parameter is evaluated conjointly with the transmission time between the processes, which leads to a complexity of $O(n^2)$ (similar to the previous algorithm). The choice of the sender-receiver pair depends on the earliest possible moment when this transmission may effectively be finished, minimizing the sum:

$$T = RT_i + g_{i,j}(m) + L_{i,j} \tag{4}$$

5.2.5 Early Completion Edge First with look-ahead - ECEF-LA

While the precedent heuristic efficiently solves the problem of the effective readiness of a sender process, it does not verify if these processes would be efficient senders on their turn. Bhat [4] proposed the use of *look-ahead* evaluation functions to make a deep analysis on the scheduling choices.

In the variant called *Early Completion Edge First with look-ahead* - ECEF-LA, the algorithm uses a *look-ahead function* F_j to characterize each process in set $\mathbf{B}$. A possible strategy considers that F_j represents the minimal transmission time from process j to any other process in set $\mathbf{B}$, which leads to an overall complexity of $O(n^3)$. Indeed, this function evaluates the utility of a process P_j if it is transferred to set $\mathbf{A}$. This way, the sender-receiver pair will be the one that minimizes the sum:

$$T = RT_i + g_{i,j}(m) + L_{i,j} + F_j \; with \; F_j = \min_{P_k \in B} (g_{j,k}(m) + L_{j,k}) \tag{5}$$

5.2.6 ECEF-LA variants

We also evaluate two different heuristics especially adapted to grid environments, both with complexity $O(n^4)$. These heuristics expand the ECEF-LA heuristic by considering the broadcast time inside each cluster i on the look-ahead function. More precisely, we call T_k the intra-cluster broadcast time. Further, we can reduce the complexity of the heuristics to $O(n^3)$ if we reuse the broadcast time T_i computed during the intra-cluster optimization phase (where we choose the fastest broadcast algorithm).

For instance, the first heuristic, called ECEF-LA**t**, tries to find a schedule that minimizes the overall communication time to a distant cluster, including the broadcast time inside each cluster i. As a result, the look-ahead function for this heuristic considers the following elements:

$$F_j = \min_{P_k \in B} (g_{j,k}(m) + L_{j,k} + T_k) \tag{6}$$

Although similar to the precedent strategy, the ECEF-LAT strategy differs in the objectives of the look-ahead function. We observed that the previous techniques tend to select the fastest clusters (a *min-min* optimization). In a

grid environment, however, this behavior penalizes the slower clusters, with a potential impact on the overall termination time. Therefore, the ECEF-LAT strategy gives priority to the clusters that need more time to finish theirs internal broadcasts. For instance, this heuristic tries to maximize the sum of the following parameters:

$$F_j = \max_{P_k \in B} \left(g_{j,k}(m) + L_{j,k} + T_k \right) \tag{7}$$

6 Practical Evaluation

To evaluate the previous heuristics in a real situation, we implemented these techniques on top of a modified version of the MagPIe library [2]. Indeed, we extended MagPIe with the capability to acquire pLogP parameters and to predict the communication performance of homogeneous clusters, as explained in [28]. Therefore, we conducted a practical experiment using 88 machines from three different clusters on the Grid'5000 network, all interconnected by a 1Gbit/s VLAN backbone. Figure 2 shows the location of the clusters, while Table 2 lists the main characteristics from each cluster.

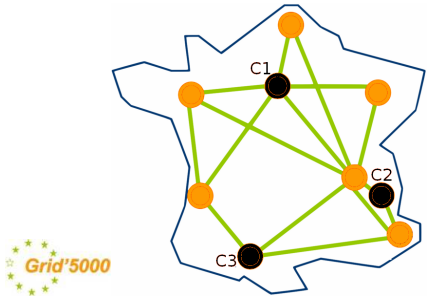


Figure 2. Grid'5000 sites

These machines were split into homogeneous clusters according to cluster map provided by Lowekamp's algorithm [18] (with a tolerance rate $\rho = 30\%$). As a result, the network was divided in six homogeneous clusters: C1-1 (29 machines at Orsay) and C1-2 (30 machines at Orsay), C2-1 (8 machines at Grenoble,

Table 2
Characteristics from the experimental testbed

	C1 - Orsay	C2 - Grenoble	C3 - Toulouse
Number of Nodes	60	8	20
Processor Type	Opteron 246	Xeon IA-32 2.4GHz	Opteron 248
Gigabit Network Adapters	Broadcom	Broadcom/Intel*	Broadcom
Memory	2GB	2GB	2GB
Software Environment	Linux 2.6.8	Linux 2.4.26	Linux 2.6.8
	LAM 7.2beta	LAM 7.2beta	LAM 7.2beta

* Intel cards present important performance problems.

Table 3
Intra and inter-cluster latencies (microseconds).

	C1-1	C1-2	C2-1	C2-2	C2-3	C3
	31 x Orsay	29 x Orsay	6 x Grenoble	1 x Grenoble	1 x Grenoble	20 x Toulouse
C1-1	47.56	62.10	12181.52	12187.24	12197.49	5210.99
C1-2	62.10	47.92	12181.52	12198.03	12195.22	5211.47
C2-1	12181.52	12181.52	35.52	60.08	60.08	5388.49
C2-2	12187.24	12198.03	60.08	0*	242.47	5393.98
C2-3	12197.49	12195.22	60.08	242.47	0*	5394.10
C3	5210.99	5211.47	5388.49	5393.98	5394.10	27.53

* these "logical clusters" have only one machine each.

Broadcom adapter), C2-2 (1 machine at Grenoble, Intel adapter) and C2-3 (1 machine at Grenoble, Intel adapter), and C3 (20 machines at Toulouse) with two levels of hierarchy distributed over three sites in France. The intra and inter-clusters latencies are presented in Table 3.

Indeed, Figure 3 present the broadcast time when varying the message size and the scheduling heuristics. The times represent the average of 10 individual runs synchronized by barriers, each one performing both intra and inter-cluster optimization steps (online optimization) based on a topology description file. Further, to better evaluate the performance speed-up obtained with the use of scheduling heuristics and the overhead caused by the optimization steps,

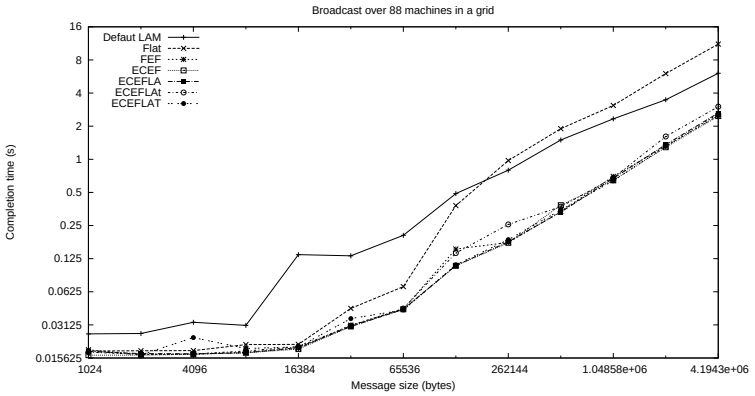


Figure 3. MPI_Bcast performance on a 88 machine grid

we compare the results with the standard `MPI_Bcast` operation provided by LAM-MPI, which uses a binomial tree.

We observe that the scheduling heuristics allow a performance improvement of at least 50% in comparison with the standard `MPI_Bcast` binomial tree. One exception is the baseline algorithm, which uses a flat tree scheduling. Because this algorithm follows a fixed scheduling that does not take into account the communication performance at the grid level, its performance is limited by the weight of the network latency. For instance, the baseline algorithm is able to minimize the communication time only when the latency dominates the transfer time (the gap), leading to a poor network performance when message sizes are more important. Indeed, in a broadcast with a higher inter-cluster transfer time, it is important to multiply the number of data sources, spreading the message to all clusters as fast as possible (somehow similar to the behavior of the binomial tree algorithm on homogeneous network). Another important point is that all other heuristics behave quite similarly. Indeed, these heuristics seem to produce optimal or quasi-optimal schedules, as observed by Bhat in his work [4]. To verify these properties and to compare these heuristics under harder conditions than the experimental testbed allows, we designed a software

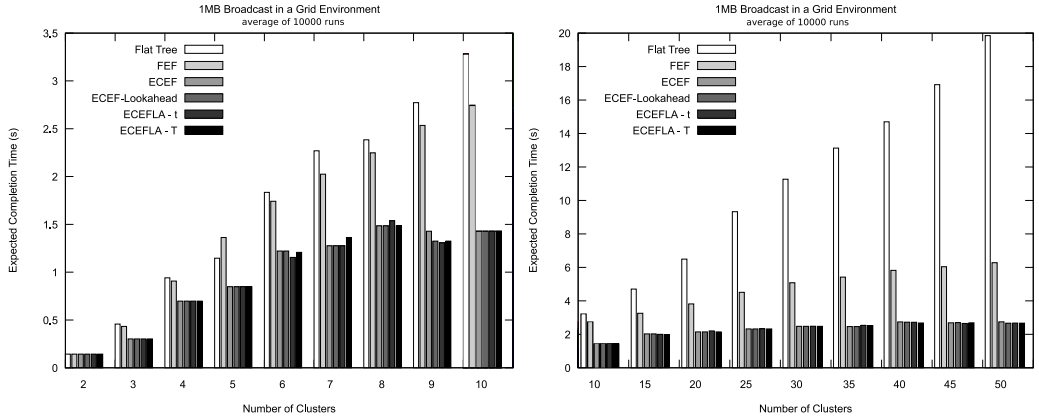
simulator where we are able to change the number of interconnected clusters and the interconnexion parameters, as presented in the following section.

7 Simulation and Scalability Concerns

While the previous section demonstrates that the use of scheduling heuristics may help to reduce the execution time of a broadcast in a heterogeneous network, we must also be concerned by the scalability of these heuristics. Although working in a grid environment such as Grid'5000, our experiments are still limited to a few clusters and network architectures. In order to evaluate the scalability and the efficiency of the heuristics presented above, we decided to compare these heuristics in a simulated environment.

We have developed a software simulator that executes the heuristic algorithms of Section 5.2, and calculates the completion time for each of them. The inputs to the simulator are the number of clusters, the size of the message to be broadcast, and the range of latencies and bandwidths (gap) in the inter-cluster network. Additionally, we provide a range of T_k values for the algorithms that consider the intra-cluster broadcast time (ECEF-LAt and ECEF-LAT). The simulator generates a random communication matrix based on these parameters. The simulator then executes the steps in the heuristic algorithms for 10000 random input configurations. Finally, the simulator reports the average completion time for each heuristic.

Figure 4 compares the performance of the different communication scheduling heuristics for the broadcast problem with a message size of 1 MB: the inter-cluster network latencies and bandwidths are chosen in the ranges of 1 ms to 15 ms and 1 MB/s to 100 MB/s respectively. Finally, T_k ranges from 200 ms to 3000 ms. Comparatively, the average latency between Grid'5000 clusters is



(a) (b)

Figure 4. Simulation results for a broadcast with different number of clusters

in the order of 5-8 ms, while the average throughput with LAM-MPI between two clusters is 50MB/s. Similarly, a broadcast of 1MB over 50 nodes in a Myrinet network takes 200 ms with the pipeline algorithm, while we need up to 3000 ms to broadcast a message in a Fast Ethernet network with the flat tree algorithm. The graph shows the completion time for the baseline algorithm, the FEF, ECEF, and look-ahead heuristics.

Initially, we evaluate the behavior of the heuristics in a grid with a reduced number of clusters, which corresponds to the majority of grid environments in use today. For instance, Figure 4(a) shows the average completion time of the `MPI_Bcast` operation with up to 10 clusters. Later, concerned by the scalability of the algorithms, we extended our simulations to evaluate the broadcast with up to 50 interconnected clusters, as represented in Figure 4(b).

In both cases, the Flat Tree schedule presents the worst performance as it does not adapts the scheduling to the *inter-cluster* communication. We also observe the limitations from the FEF heuristic, corroborating the problems pointed in section 5.2. Indeed, FEF considers that sender nodes are immediately avail-

able, while in reality there is a transmission gap that must be respected (the Ready Time parameter).

While Flat Tree and FEF heuristics clearly show their limitations, all other heuristics (ECEF, look-ahead, ...) present good results. Because these techniques are able to start communications from different clusters in parallel and therefore minimizing the execution time, the number of clusters has a small influence on the overall communication time. Another interesting point is that all these heuristics present similar results, being aware of the intra-cluster broadcast time (T_k) or not. The fact that the intra-cluster broadcast time hardly influences the overall termination time has two main reasons: first, inter-cluster communications are far more expensive, and optimizing the inter-cluster schedule reduces considerably the execution time. Second, intra-cluster communications are already optimized in our framework, reducing their impact on the overall execution. Hence, the association of two optimization levels (intra and inter-cluster) seems to be fair sufficient to obtain good communication performances. The choice of the scheduling heuristic reposes therefore on the complexity of the scheduling heuristic and the heterogeneity of the environment, for which the software simulation environment can help to compare.

8 Concluding Remarks and Future Works

In this paper we presented a grid-aware communication framework based adaptive approaches for predicting and optimizing the performances of collective communication algorithms on heterogeneous hierarchical grids. We defined the concept of polyalgorithmic optimization, and proposed a methodology that proceeds in two adaptation levels to dynamically associate the fastest

algorithm for a give cluster and a communication schedule that minimizes the termination time. In this work we present a case study on an important collective communication pattern, the broadcast operation, proving the interest of the proposed multi-level adaptive scheme. Both experimental and simulated results are used to illustrate the operation of the framework and the benefits to the collective communications performance. Indeed, this framework is implemented in our grid-aware MPI communication library LaPIe, in which we intend to integrate other communication patterns and scheduling algorithms based on the principles from this framework.

Acknowledgments

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see https://www.grid5000.fr). We are also grateful to all anonymous referees for their helpful comments and suggestions that helped us improving this document.

References

[1] N. T. Karonis, B. Supinski, I. Foster, W. Gropp, E. Lusk, J. Bresnahan, Exploiting hierarchy in parallel computer networks to optimize collective operation performance, in: Proceedings of the 14th International Conference on Parallel and Distributed Processing Symposium, IEEE Computer Society, 2000, pp. 377–384.

[2] T. Kielmann, R. Hofman, H. Bal, A. Plaat, R. Bhoedjang, Magpie: MPI's collective communication operations for clustered wide area systems, in: Proceedings of the 7th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 1999, pp. 131–140.

[3] T. Kielmann, H. Bal, S. Gorlatch, K. Verstoep, R. Hofman, Network performance-aware collective communication for clustered wide area systems, Parallel Computing 27 (11) (2001) 1431–1456.

[4] P. B. Bhat, C. Raghavendra, V. Prasanna, Efficient collective communication in distributed heterogeneous systems, Journal of Parallel and Distributed Computing 2003 (63) (2003) 251–279.

[5] P. Liu, D.-W. Wang, Y.-H. Guo, An approximation algorithm for broadcast scheduling in heterogeneous clusters, in: Proceedings of the Real-Time and Embedded Computing Systems and Applications, 9th International Conference (RTCSA 2003), LNCS 2968, Springer-Verlag, 2004, pp. 38–52.

[6] O. Hartmann, M. Kuhnemann, T. Rauber, G. Runger, Adaptive selection of communication methods to optimize collective mpi operations, in: Proceedings of the 12th Workshop on Compilers for Parallel Computers (CPC'06), La Coruna, Spain, 2006.

[7] F. Cappello, P. Fraigniaud, B. Mans, A. Rosenberg, An algorithmic model for heterogeneous hyper-clusters: Rationale and experience, International Journal of Foundations of Computer Science 16 (2) (2005) 195–215.

[8] L. G. Valiant, A bridging model for parallel computation, Communications of the ACM 33 (8) (1990) 103–111.

[9] A. Bar-Noy, S. Kipnis, Designing broadcasting algorithms in the postal model for message-passing systems, Math. Systems Theory 27 (5) (1994) 431–452.

[10] R. Hockney, The communication challenge for MPP: Intel paragon and meiko cs-2, Parallel Computing 20 (1994) 389–398.

[11] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, T. von Eicken, LogP - a practical model of parallel computing, Communication of the ACM 39 (11) (1996) 78–85.

[12] A. Alexandrov, M. Ionescu, K. Schauser, C. Scheiman, LogGP: Incorporating long messages into the LogP model - one step closer towards a realistic model for parallel computation, in: Proceedings of the 7th Annual Symposium on Parallel Algorithms and Architecture (SPAA'95), 1995.

[13] P. Liu, D.-W. Wang, Reduction optimization in heterogeneous cluster environments, in: Proceedings of the 14th IPDPS, 2000, pp. 477–482.

[14] R. Wolski, N. Spring, C. Peterson, Implementing a performance forecasting system for metacomputing: The network weather service, in: Proceedings of the Supercomputing, 1997.

[15] P. Dinda, T. Gross, R. Karrer, B. Lowekamp, N. Miller, P. Steenkiste, D. Sutherland, The architecture of the remos system, in: Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC-10), 2001.

[16] M. den Burger, T. Kielmann, H. Bal, TopoMon: a monitoring tool for grid network topology, in: Proceedings of the International Conference on Computational Science'02, LNCS Vol. 2330, Springer-Verlag, 2002, pp. 558–567.

[17] O. Beaumont, L. Marchal, Y. Robert, Broadcasts trees for heterogeneous platforms, in: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2005), 2005.

[18] B. Lowekamp, Discovery and application of network information, Ph.D. thesis, Carnegie Mellon University (2000).

[19] M. Barnett, D. Payne, R. van de Geijn, J. Watts, Broadcasting on meshes with wormhole routing, Journal of Parallel and Distributed Computing 35 (2) (1996) 111–122.

[20] R. Thakur, R. Rabenseifner, W. Gropp, Optimization of collective communication operations in MPICH, International Journal of High Performance Computing Applications 19 (2005) 49–66.

[21] L. Barchet-Steffenel, G. Mounie, Performance characterisation of intra-cluster collective communications, in: Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2004), Foz do Iguacu, Brazil, 2004, pp. 254–261.

[22] J. Pjesivac-Grbovic, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, J. J. Dongarra, Performance analysis of MPI collective operations, in: Proceedings of the Wokshop on Performance Modeling, Evaluation and Optimisation for Parallel and Distributed Systems (PMEO), in IPDPS 2005, 2005.

[23] R. Thakur, W. Gropp, Improving the performance of collective operations in MPICH, in: Proceedings of the Euro PVM/MPI 2003, LNCS Vol. 2840, Springer-Verlag, 2003, pp. 257–267.

[24] M. Banikazemi, V. Moorthy, D. K. Panda, Efficient collective communication on heterogeneous networks of workstations, in: Proceedings of the International Conference on Parallel Processing (ICPP'98), 1998, pp. 460–467.

[25] J.-Y. L. Park, H.-A. Choi, N. Nupairoj, L. M. Ni, Construction of optimal multicast trees based on the parameterised communication model, in: Proceedings of the International Conference on Parallel Processing (ICPP 1996), 1996, pp. 180–187.

[26] G. Mateescu, A method for MPI broadcast in computational grids, in: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2005), 2005.

[27] T. Vorakosit, P. Uthayopas, Generating an efficient dynamic multicast tree under grid environnement, in: Proceedings of the Euro PVM/MPI 2003, LNCS 2840, 2003, pp. 636–643.

[28] L. Barchet-Steffenel, G. Mounie, Identifying logical homogeneous clusters for efficient wide-area communication, in: Proceedings of the Euro PVM/MPI 2004, LNCS Vol. 3241, Budapest, Hungary, 2004, pp. 319–326.