A Framework for Adaptive Collective Communications on Heterogeneous Hierarchical Networks

Luiz Angelo Steffenel

A Framework for Adaptive Collective Communications on Heterogeneous Hierarchical Networks

Luiz-Angelo Steffenel

N° 6036

Novembre 2006

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6036--FR+ENG

A Framework for Adaptive Collective Communications on Heterogeneous Hierarchical Networks

Luiz-Angelo Steffenel[*]

Thème NUM — Systèmes numériques
Projet AlGorille

Rapport de recherche n° 6036 — Novembre 2006 — 29 pages

Abstract: Today, due to the wide variety of existing parallel systems consisting on collections of heterogeneous machines, it is very difficult for a user to solve a target problem by using a single algorithm or to write portable programs that perform well on multiple computational supports. The inherent heterogeneity and the diversity of networks of such environments represent a great challenge to model the communications for high performance computing applications. Our objective within this work is to propose a generic framework based on communication models and adaptive techniques for dealing with prediction of communication performances on cluster-based hierarchical platforms. Toward this goal, we introduce the concept of polyalgorithmic model of communications, which correspond to selection of the most adapted communication algorithms and scheduling strategies, giving the characteristics of the hardware resources of the target parallel system. We apply this methodology on collective communication operations and show that the framework provides significant performances while determining the best algorithm depending on the problem and architecture parameters.

Key-words: Cluster computing, Performance modeling, Adaptive techniques, Polymodels of communications, Collective communication operations

[*] IUT Nancy Charlemagne - Université Nancy 2

Un Framework pour des Communications Collectives Adaptées aux Réseaux Hétérogènes Hierarchiques

Résumé : À nos jours, la variété et l'hétérogénéité des systèmes de calcul parallèle rendent très difficile le développement d'algorithmes qui sont à la fois portables et efficaces sur tous les environnements de calcul distribué. Parmi les facteurs d'hétérogénéité nous retrouvons notamment les aspects liés à la communication, dont la diversité d'architectures et de performances constitue un sérieux défi à la modélisation et l'optimisation des applications réparties. C'est dans ce cadre de travail que nous proposons l'utilisation d'une approche polyalgorithmique, capable d'évaluer l'environnement de travail (à travers la modélisation et la prédiction des coûts de communication) et d'indiquer les algorithmes les plus adaptés à chaque grappe ou grille de calcul. Ainsi, nous illustrons cette méthodologie à travers la modélisation et l'optimisation des opérations de communication collective, qui sont fortement dépendantes de la performance du réseau. Nous démontrons en effet que la prise en compte des facteurs d'hétérogénéité et l'utilisation d'algorithmes adaptés à chaque sous-réseau permet des gains de performance considérables par rapport aux implémentations classiques.

Mots-clés : Grilles de calcul, Modèlisation de performances, Adaptativité, Polymodèles de communication, Communication collective

1 Introduction

1.1 Evolution of High Performance Computing

The major innovation in parallel processing in the last years was the huge spreading of architectures like grids and clusters. These platforms, made by aggregating PCs or work stations, represent a reasonable alternative and have become the most cost-effective computing supports for solving a large range of high performance computing applications because they provide many advantages, such as a better cost/performance ratio compared to traditional parallel machines. 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. Today, as the systems are composed of collections of heterogeneous machines, it is very difficult for a user to choose an adequate algorithm because the execution supports are continuously evolving: one version will be well-suited for a parallel configuration and not for another one. For instance, portability issue becomes crucial because of the frequent changes of the components of the systems. These different elements require to revise the classical parallel algorithms which consider only regular architectures with static configurations and to propose new approaches.

1.2 Adaptive Approaches

The adaptive approaches are a promising answer to this problem. The idea is to adapt algorithms together with their execution to the target architecture. They propose adaptive algorithms for solving the same problem which have different behavior and performances and to derive the best one for a target parallel system. These algorithms may be automatically adapted to the execution context (data and support). In a parallel context, the adaptive algorithm should be scalable. Indeed, it should maintain a given efficiency when the size of the problem and the number of processors grow. Other approaches are possible for adapting the algorithms to new supports. For instance, adequate software can be developed in the middleware. We are interested in this work in the adaptivity at the algorithmic level.

1.3 Contribution of this Work

Our objective within this work is to propose a generic framework based on communication models and scheduling techniques for dealing with prediction and communication scheduling, and integrating scalability and portability issues. More precisely, the contribution of this paper is to propose a methodology with two levels of adaptation. Indeed, at the first level we proceed, given a target architecture, by determining the most appropriate communication strategy from a set of selected ones. Our framework differs from the other works in a significant aspect. Indeed, existing adaptive approaches presented in the literature like those of [1, 2, 3] proceed by simply scheduling communications at the wide-area range, i.e., long-distance links (for example, the communication between two distant clusters). At the other side, works like those of [4] 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 for automatically associate the most appropriate algorithm among multiple algorithmic options and the corresponding communication schedule to fit the best performance requirements. It determines the best combination strategy-schedule and computes an efficient execution scheme that minimizes the overall execution time of a parallel application depending on collective communications.

1.4 Organization of the Paper

The remainder of the paper is organized as follows. We begin, in Section 2, by presenting the architectural model of the target parallel and distributed system and the performance evaluation models of a parallel algorithm. In Section 3, we first define the concept of polyalgorithm, presenting our adaptive framework for polyalgorithmic communications and detailing its components. Sections 4 and 5 are devoted to a case study where we apply our adaptive framework on the development of a grid-aware `MPI_BCast` collective communication operation. Therefore, we validate the framework by evaluating its performance both through numerical simulations (Section 6) and real experiments on a grid environment (Section 7), proving the interest of this work. Finally, Section 8 concludes the paper and discusses some perspectives to extend this work.

2 Preliminaries

2.1 Description of the Architectural Model

We assume in this work a generic model of a platform composed by heterogeneous hierarchical clusters as described in [5]. The platform studied enjoys heterogeneity along three orthogonal axes:

1. The processors that populate the clusters may differ in computational powers, even within the same cluster.
2. The clusters comprising the platform are organized hierarchically and are interconnected via a hierarchy of networks of possibly differing latencies, bandwidths and speeds. At the level of physical clusters, the interconnection networks are assumed to be heterogeneous.
3. The clusters at each level of the hierarchy may differ in sizes.

2.2 Communication Models

In this section we present the communication, transmission, and synchronization models used in this work. We assume that the network is fully connected. These models can be used to approximately model most current parallel machines with distributed memory.

Communication Model: 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 restriction is well-known in the literature as *1-port full-duplex*.

Transmission Model: There are many parallel communication models in the literature that analyze performances based on system parameters [6, 7, 8, 9, 10, 3, 11]. These models differ on the assumption about the computational support parameters, such as latency, heterogeneity, network contention, etc. Hence, we adopted in this paper the *parameterized LogP* model (*pLogP*) [3], an extension of the LogP performance model [9] that can accurately handle both small and large messages with a low complexity. Hence, all along this paper we shall use the same terminology from pLogP's definition, such as $\boldsymbol{g(m)}$ for the gap of a message of size m, $\boldsymbol{L}$ as the communication latency between two nodes and $\boldsymbol{P}$ as the number of nodes. 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. Thus, $g(s)$ represents the gap of a segment with size s.

Synchronization Model: We assume an asynchronous communication model, where transmissions from different processes do not have to start at the same time. However, all processes start the algorithm simultaneously. This model corresponds to the execution of the `MPI_BCast` operation, used as reference in this work.

The total time for an algorithm is the difference between the start time and the time at which all processes are finished.

3 An Adaptive Framework for Grid-Aware Collective Communications

In this section, we describe our framework for adaptively modeling communications in an execution environment characterized by its heterogeneity and its organization hierarchically. An overview of the methodology is sketched in Figure 1. The processing is separated in two successive phases. During the first one, we aim to partition the target execution platform to form subnets of similar characteristics by automatically discover the network topology. Then, when executing the second phase, we initially determine for each subnet (i.e. cluster) the most appropriate communication strategy. Indeed, through the use of performance models we are able to predict the communication performance for each different cluster. These predictions are therefore used to obtain a wide-area communication schedule that minimizes the overall communication time.

Since the target parallel system may be heterogeneous at many levels (computing powers, interconnection network performances, etc), it is very difficult to manage such platform towards a high performance computing. One way to answer this problem and to minimize the inherent heterogeneity, and thus facilitating the execution, is to subdivide the network in homogeneous subnets (or logical clusters), as described below. At the end of this phase, we

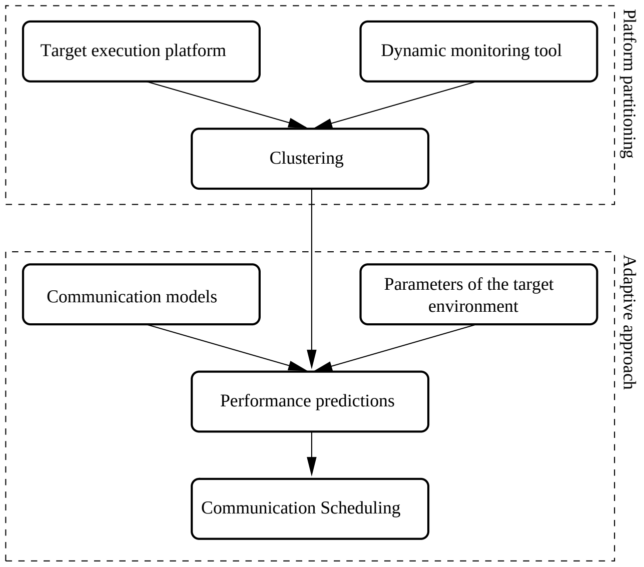

Figure 1: Conceptual framework of the adaptive mechanism

will get a set of logical clusters of homogeneous nodes and accurate interconnection network, which will be used to adaptively modeling communications inside each cluster during the second phase of the framework.

Once the platform is partitioned in separated homogeneous hierarchical clusters, we determine, using an adaptive approach, an adequate algorithm from a set of selected ones for each cluster. Indeed, we modeled and implemented several algorithms from the literature, which perform differently according to the network environment. By selecting the best adapted algorithm to each different cluster in our grid, we contribute to a polyalgorithmic modeling of communications in a grid environment. We recall that the algorithm selection is made in terms of information which is interesting to the problem, such as the size of data to communicate, the type of interconnection network, the number of nodes, etc.

Through the analysis of the inter-clusters communication performance and the intra-cluster performance prediction we are able to define a communication schedule that minimizes the overall execution time. Once again we implement different schedule policies, which are selected 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* and *reduce*) or at node-to-node level (for operations such as the *all-to-all*).

4 Implementing the Framework

4.1 First Phase: Network Partition

We propose a method to automatically discover network topology in order to allow 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 communication 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 startup, identifies SMP nodes (or processes in the same machine), subdivides the networks in homogeneous logical clusters and finally acquires pLogP parameters to model collective communications.

As the first step is independent from the application, it can use information from different monitoring services, which are used to construct a distance matrix. This distance matrix does not need to be complete, in the sense that a cluster does not need to monitor its interconnection with other clusters, and several connectivity parameters can be used to classify the links and the nodes as, for example, latency and throughput. When the network is subdivided in homogeneous subnets, we can acquire pLogP parameters, necessary to model the collective communications and to determine the best communication schedule or hierarchy. Due to the homogeneity inside each subnet, pLogP parameters can be obtained in an efficient way, which reflects in a small impact on the application initialization time.

At the end of this process we have logical clusters of homogeneous machines and accurate interconnection parameters, which can be used to construct an interconnection tree (communicators and sub-communicators) that optimizes both inter and intra-cluster communication.

4.1.1 Obtaining Network Metrics

There exist many tools specialized on network monitoring that can be used to gather connectivity information. These tools may acquire data from direct probing, like NWS [12], from SNMP queries to network equipments, like REMOS [13], or even combine both approaches, like TopoMon [14]. NWS seems to be the best candidate to our needs, as it is a *de facto* standard in the Grid community and can be configured to provide information like communication latency, throughput, CPU load and available memory. For instance, the communication latency and throughput, obtained from NWS, may be used to identify groups of machines with similar communication characteristics.

4.1.2 Clustering

One reason to construct logical clusters is that even machines in the same network may behave differently, in spite of their physical location or network subnet. Indeed, such differences introduce undesirabl e 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".

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 [15], but also can be implemented through the pruning of the full interconnection graph [16]. Another approach consists on defining a "closeness" parameter ρ, which indicates the maximum variance among nodes in the same group. Contrarily to the precedent techniques, this one does not define the full hierarchy of nodes.

In the specific case of our work, the last technique seems to be the most appropriate. Indeed, we are simply interested on the definition of clusters of homogeneous nodes, as we intend to optimize inter-cluster communications in a further moment.

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. Note that there is not necessarily any relationship between this digraph and the topology of the interconnection network. Hence, this digraph corresponds to the distance matrix M defined by:

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

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 varies 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:

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

Because we need to compare node a to each node from group S, this algorithm executes in $O(N^2)$ steps. Therefore, Lowekamp [17] presented a greedy algorithm, which was implemented within the ECO library (Algorithm 1) 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:

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

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

Algorithm 1 Lowekamp's algorithm [17] for partitioning the network in subnets ($\rho = 20\%$)

```
initialize subnets to empty
for all nodes
  node.min_edge = minimum cost edge incident on node
sort edges by nondecreasing cost
for all edges (a,b)
  if a and b are in the same subnet
    continue
  if edge.weight>1.20 * node(a).min_edge or edge.weight>1.20 * node(b).min_edge
    continue
  if node (a) in a subnet
    if (edge.weight>1.20 * node(a).subnet_min_edge)
      continue
  if node (b) in a subnet
    if (edge.weight>1.20 * node(b).subnet_min_edge)
      continue
  merge node(a).subnet and node(b).subnet
  set subnet_min_edge to min(edge,node(a).subnet_min_edge, node(b).subnet_min_edge)
```

to be evaluated in our topology discovery implementation. Indeed, latency has prove to be sufficiently accurate to distinguish between different network architectures (Fast Ethernet, Giga Ethernet, Myrinet) in a local network, as well as between network cards from different manufacturers. Further, latency measures can be easily performed in a wide area network without disturbing the ongoing traffic, contrarily to a bandwidth measurement. Hence, latency would allow us to approximate the network logical topology without incurring too much initial cost. At the other hand, only latency is not very useful in our framework, as we need bandwidth data to estimate the performance of the different communication patterns (and optimize them). Fortunately, our knowledge on the network topology can help to efficiently obtain the necessary interconnection parameters, as explained below.

4.1.3 Efficient Acquisition of pLogP Parameters

While the logical clusters generated by our framework allow a better understanding of the network effective structure, we are still unable to model communications with precision. This first reason is that interconnection data may be incomplete. Indeed, if we strictly consider the procedure presented above, only the communication latency would be measured.

Besides this, the data acquired by the monitoring tools may not be the compatible with the data used in our models. For example, the latency, which originally should have the same meaning to the monitoring tool and the application, is obtained differently by NWS and pLogP. In NWS, the latency is obtained directly from the round-trip time, while pLogP separates the round-trip time in latency and gap, as depicted by Figure 2, with differences that may interfere on the communication model.

Hence, to model the communication in our network, we need to obtain parameters specifically for pLogP. Hopefully, there is no need to execute $n(n-1)$ pLogP measures, one for each possible interconnection. The first reason is that processes belonging to the same machine were already identified as SMP processes and grouped in specific sub-communica tors. And second, the subnets are relatively homogeneous, and thus, we can get pLogP parameters in

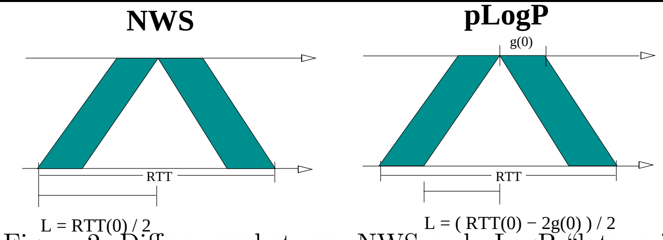

Figure 2: Differences between NWS and pLogP "latency"

an efficient way by considering a single measure inside each subnet as a sample from the pLogP parameters common to the entire 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 subnets. Further, if we suppose symmetrical links, we can reduce this number of measures by half, as $a \rightarrow b = b \rightarrow a$.

4.2 Intra-cluster Communication Strategy Selection - Broadcast Operations

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. While an optimal tree shape can be deduced from the network parameters and from $d,\ h \in$[1...P-1] for which $\sum_{i=o}^{h} d^i \geq P$ is true, most MPI implementations usually rely on two fixed shapes, the Flat Tree, for small number of nodes, and the Binomial Tree.

Because most MPI implementations rely only on Flat and Binomial Broadcast , some techniques were developed to improve its efficiency. This way, it is usual to apply different strategies according to the message size, as for example, the use of a *rendezvous* message that prepares the receiver to the incoming of a large message, or the use of non-blocking primitives to overlap communication and computation. Unfortunately, such techniques bring only minimal improvements to the final performance, and their efficiency still depends mostly on the network characteristics.

Another possibility, however, is to compose a Chain among the processes, pipelining messages [18]. This strategy benefits from the use of message segmentation, presenting many advantages as recent works indicate [3][19]. In a Segmented Chain Broadcast, the transmission of messages in segments allows a node to overlap the transmission of segment k and the reception of segment k+1, reducing the overall *gap* time.

However, the size of the segments should be carefully 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 network bandwidth. Therefore, an efficient method to identify an adequate segment size s consists in searchin g through all values of s where $s = m/2^i, i \in [0 \ldots log_2 m]$ such that s minimizes the predicted performance of the communicati on 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 and their "flavors", which are presented on Table 1. Most of these variations are clearly expensive, while others have only an "historical" interest. Hence, we chose to compare in this paper two of the most efficient techniques, the Binomial and the Segmented Chain Broadcasts, and the simplest one, the Flat Tree Broadcast.

Strategy	**Communication Model**
Flat Tree	$L + (P-1) \times g(m)$
Flat Tree Rendez-vous	$3 \times L + (P-1) \times g(m) + 2 \times g(1)$
Segmented Flat Tree	$L + (P-1) \times (g(s) \times k)$
Chain	$(P-1) \times (g(m) + L)$
Chain Rendez-vous	$(P-1) \times (g(m) + 2 \times g(1) + 3 \times L)$
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	$\lceil log_2 P \rceil \times L + \lfloor log_2 P \rfloor \times g(m)$
Binomial Tree Rendez-vous	$\lceil log_2 P \rceil \times (2 \times g(1) + 3 \times L) + \lfloor log_2 P \rfloor \times g(m)$
Segmented Binomial Tree	$\lceil log_2 P \rceil \times L + \lfloor log_2 P \rfloor \times g(s) \times k$
Scatter/Collection [20]	$(log_2 P + P - 1) \times L + 2 \times (\frac{p-1}{p}) \times g(m)$

Table 1: Communication models for the *Broadcast* operation

4.3 Models Validation

To evaluate the accuracy of our models, we present some results on the evaluation of the Flat, Binomial and the Segmented Chain Broadcasts in real experiments, comparing these results with the model predictions. Although Flat tree is not adequate for a large number of processes, we included it because its simplicity is a good parameter to evaluate other algorithms that use more complex strategies. Hence, Figures 3, 4 and 5 present each strategy compared to its performance model's predictions. Despite some performance variations found mostly in the Segmented Chain and the Binomial Broadcast, we can observe that predictions seem to follow the real experiments general behavior. Actually, as these variations are much less important in the case of Flat and Binomial Broadcast, we think that they are related to communication delays introduced in some machines and further propagated by the message forwarding of the Chain broadcasts.

It is important to note, however, that Figures 3, 4 and 5 are not in the same scale due to the different performance level of each algorithm. To compare these algorithms and to better observe the models' accuracy, we present on Figure 6 the results obtained for a group of 16 machines. Here, we observe that the Segmented Chain Broadcast is the better adapted strategy for our cluster, even if the models predictions have slightly underestima ted the communication cost.

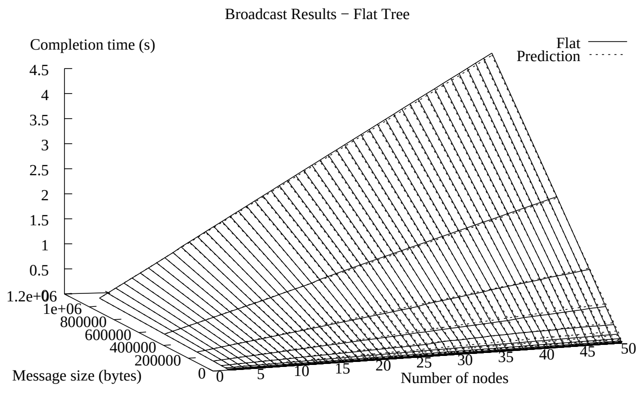

Figure 3: Real and expected performance for the Flat Tree Broadcast

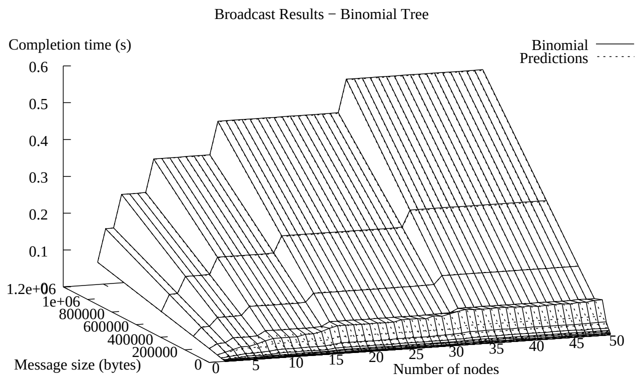

Figure 4: Real and expected performance for the Binomial Broadcast

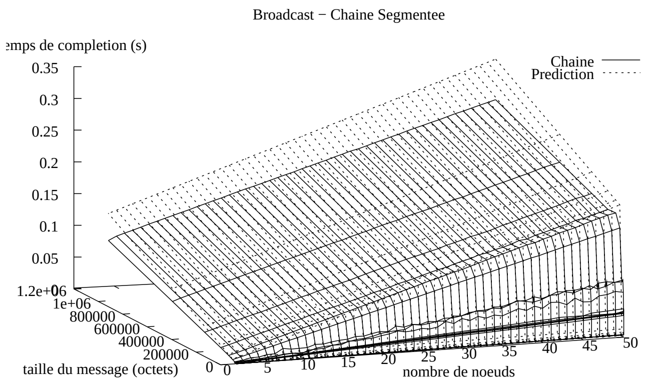

Figure 5: Real and expected performance for the Segmented Chain Broadcast

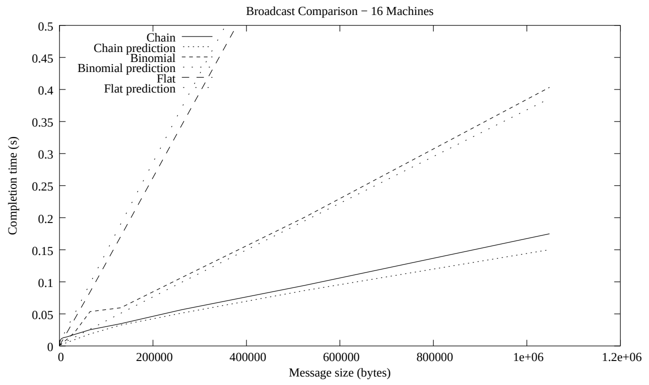

Figure 6: Comparison between models and real results, for 16 machines

Table 2: Communication levels according to their latency [29]

Level 0	Level 1	Level 2	Level 3, 4, ...
WAN-TCP	LAN-TCP	localhost-TCP	shared memory Myrinet Vendor MPI

4.4 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 [16], most authors such as Banikazemi [21], Bhat [22], Liu [23], Park [24], Mateescu [25] and Vorakosit [26] try to generate optimal broadcast trees according to a given *root* process, which corresponds to the `MPI_Bcast` operation.

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 [27, 17], 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 *inter- cluster* communications are structured in a flat tree, while *intra-cluster* communications benefit from efficient strategies like binomial trees. 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, 28]. In his work, Karonis defined a multilevel hierarchy that allows communica tion overlapping between different levels (cf. Table 2). While this structure on multiple levels allows a performance improvement, it still relies on flat trees to disseminate messages between two wide area levels (levels 0 and 1), the same strategy as ECO or MagPIe.

However, a flat tree is far from being an optimal tree shape for heteroge neous systems. Because network heterogeneity does not allow trivial solutions, we are constrained to generate specific broadcast trees for each environm ent. Due to complexity concerns, we cannot rely on exhaustive search of the optimal tree, which is exponential; we must then rely on optimization heuristics. In this paper we use as reference some heuristics from Bhat [22], which try to construct efficient communication schedules that minimize the broadcast execution time.

Nevertheless, 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 contributes to the communication time, as the *intra-cluster* communication time may represent an important optimization parameter. Indeed, current clusters may be composed by several hundred nodes and, even with high performance network interconnections, a broadcast in a local area network may take several milliseconds [30], which sometimes is higher than a long distance transmission. Hence, we propose a smart schedule of wide-area collective communications, which considers both *inter* and *intra-cluster* times to minimize makespan.

4.5 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 [22]. Hence, 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. This way, set **A** initially contains only the cluster from the *root* process, while all other clusters are listed on the set **B**. 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 cluster does not participate in any other *inter-cluster* communication, it can finally broadcast the message among the cluster processes. This strategy improves the multiplication of data sources, giving more choices to the optimization heuristics.

To model the communication performance of *intra-cluster* communicat ion, we also use the *parameterized LogP* model (*pLogP*) [3]. Hence, in this section we use $g_{i,j}(m)$ to represent the communication gap of a message with size m between two clusters i and j. Similarly, we use $L_{i,j}$ to represent the communication latency between these clusters.

5 Scheduling Heuristics

5.1 Baseline Algorithm - Flat Tree

Used by ECO and MagPIe libraries, 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 destination among the clusters in set **B**. At each communication round, the *root* process chooses a new cluster from set **B**, despite the presence of other (potential) sources in set **A**. Once a cluster coordinator receives a message, it broadcasts it inside the clusters using a binomial tree technique.

Although easy to implement, this strategy is far from being optimized. Indeed, the diffusion of messages does not take into account the performance of different clusters, neither the interconnexion speeds. Further, this technique depends on how the clusters list is arranged with respect to the root process, and important performance variations can be observed on applications that rotate the role of the broadcast root.

5.2 Fastest Edge First - FEF

Proposed by Bhat *et al.* [22], 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 sorted again. Hence, the strategy behind this technique is to maximize the number of sender processes, augmenting the number of possible paths that can be explored to reach the more distant processes.

5.3 Early Completion Edge First - ECEF

In the previous heuristics, once the receiver was assigned it was immediately transferred to the set **A**, and could take part in the next communication round. This model, however, is not realistic, as communication delays may prevent a receiver process from having the message immediately. Indeed, it is possible that a process from set **A** is chosen to send a message before it has the message available for retransmission; in this case, communications are blocked until the message becomes available at the sender.

To avoid such situations, Bhat proposed the heuristic called *Early Completion Edge First*, which tries to keep 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, and the choice of the sender-receiver pair depends on the earliest possible moment when this transmission may effectively be finished, as stated by:

$$RT_i + g_{i,j}(m) + L_{i,j}$$

Hence, the final objective of the heuristic is to augment the number of sources that can *effectively* retransmit message to the other processes.

5.4 Early Completion Edge First with lookahead - ECEF-LA

While the precedent heuristic efficiently solves the problem of multiplication of sources that can effectively retransmit a message in a next communication round, it does not offers a guarantee that these new sources would be as efficient to transmit messages as well. To increase the efficiency of the ECEF heuristic, Bhat [22] proposed the use of *lookahead* evaluation functions to make a deep analysis on the scheduling choices.

In the variant called *Early Completion Edge First with lookahead* - ECEF-LA, the algorithm uses a *lookahead function* F_j to characterize each process in set **B**. This way, the sender-receiver pair will be the one that minimizes the sum:

$$RT_i + g_{i,j}(m) + L_{i,j} + F_j$$

To define the *lookahead function* we can use several strategies. Bhat [22] proposed, for example, that F_j represents the minimal transmission time from process j to any other process in set **B**. Indeed, this function can be described as:

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

Hence, this lookahead function evaluates the utility of a process P_j if it is transferred to set **A**. Nevertheless, Bhat suggest some other *lookahead* functions like the average latency between P_j and the other processes in **B** or the average latency between processes in sets **A** and **B**, if P_j was transferred to set **A**.

5.5 ECEF-LA*t*

Because previous heuristics were not designed for wide area systems, we propose in this paper three different heuristics, especially adapted to grid environments. The first one is an extension of the ECEF-LA heuristic, but it uses a *lookahead* function that evaluates both the communication cost at the *inter-cluster* level and T_i, the broadcast time inside a cluster i. Hence, we try to find a schedule that minimizes the overall communication time to a distant cluster (the small t in the name indicates that we are looking for the minimum), and we use the lookahead function:

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

The reasoning of this strategy is that the receiver should be choose not only because it can efficiently retransmit messages to other clusters, but also because the clusters it can reach will likely complete their broadcasts within a reduced interval of time.

5.6 ECEF-LAT

Although its similarity with the precedent strategy, the ECEF-LAT strategy differs in the objectives of the lookahead function. Indeed, this heuristic tries to maximize the sum of the parameters from the lookahead function F_j:

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

This approach comes from the observation of the previous techniques, which tend to select fastest clusters over slowest or more distant ones. In a grid environment, however, this behavior may introduce extra retards to the termination of the slowest clusters, which impacts directly the communication makespan. Therefore, in the ECEF-LAT strategy, we give priority to the clusters that need more time to finish theirs internal broadcasts. We assume that this strategy will limit the termination delay of these clusters, while communication overlap at the *inter-cluster* level will guarantee that all clusters will finish in a bounded time.

5.7 BottomUp

A close analysis on the previous heuristics reveals that besides the use of different lookahead functions, the ECEF-LA* heuristics rely on *min-max* or *min-min* optimization techniques. From the ECEF-LAT technique we observe that sometimes it is interesting to distribute messages to slow clusters first, as a mean to reduce the overall slowdown. Nevertheless, it is still necessary to multiply the number of sources, whose best strategy consists on contacting fast clusters first. Although these strategies seem to be in opposition, they are not mutually exclusive. Indeed, we propose a new heuristic to combine these two strategies. This heuristic, called BottomUp, relies on a *max-min* optimization strategy to choose the sender that can reach the slowest cluster in the minimum time:

$$\max_{P_j \in B} (\min_{P_i \in A} (g_{i,j}(m) + L_{i,j} + T_j))$$

Therefore, this method should allow the available sources to contact slowest clusters as soon as possible, while trying to select potential senders that can contribute to a fast message distribution.

6 Simulation

In order to evaluate the efficiency of the heuristics presented above, we chose in a first moment to simulate the execution of the `MPI_Bcast` operation. For instance, we provide the heuristics with realistic communication parameters, obtaining a communication schedule that is used to calculate the makespan. Therefore, at each iteration, the parameters L, g and T are randomly chose among the values presented in Table 3. These parameters correspond to average values measured over the French national grid GRID5000[1], and the results presented below correspond to the average of 10000 simulation runs.

Table 3: Performance parameters used in the simulations

	minimum	maximum
L	1 ms	15 ms
g	100 ms	600 ms
T	20 ms	3000 ms

Initially, we evaluate the behavior of the heuristics in a grid with a reduced number of clusters, ranging from 2 to 10. This number corresponds to the majority of grid environments in use today. Indeed, Figure 7 shows the average completion time for a reduced number of clusters. As expected, a Flat Tree schedule presents the worst performance, as it does not try to adapt the *inter-cluster* communication. Further, we also observe that the BottomUp heuristic presents a better performance than the FEF technique; this indicates that in a grid system it is sometimes more important to take into account the performance of slow clusters than the pure interconnection speed. We believe that this technique can be improved with the use of *lookahead* functions similar to those used by the ECEF-LA heuristics; indeed, a *lookahead* function can be used to guarantee that a receiver will be a good sender at his turn.

If Flat Tree and FEF heuristics clearly show their limitations and the BottomUp technique illustrates some interesting research directions, the best performance levels in our simulations were achieved by the ECEF* techniques. We observe that the overall communication time does not increases linearly with the number of clusters, evidence that these techniques are able to interleave communications from different clusters and therefore minimizing the execution time.

These results are validated in Figure 8, which presents the expected performances for grids with up to 50 clusters . Although most grid systems are still composed by a small group of clusters, this number tends to increase in the next years, and therefore it is important to identify efficient techniques to meet these new constraints. Hence, Figure 8 demonstrates that the Flat Tree approach is clearly inefficient for a large number of clusters. Similarly, the greedy algorithm FEF does not achieve good performance levels, as communication latency is not a sufficient parameter to balance communication times and minimize the makespan.

It is interesting to note that all ECEF-like heuristics behave quite identically. This average behavior confirms their efficiency, and in especial their ability to reschedule communications at the inter-cluster level. Therefore, all these techniques are good candidates for a real application, and the selection of the best techniques depends mostly on the working environment, which can benefit from specific evaluation approaches.

[1] http://www.grid5000.org

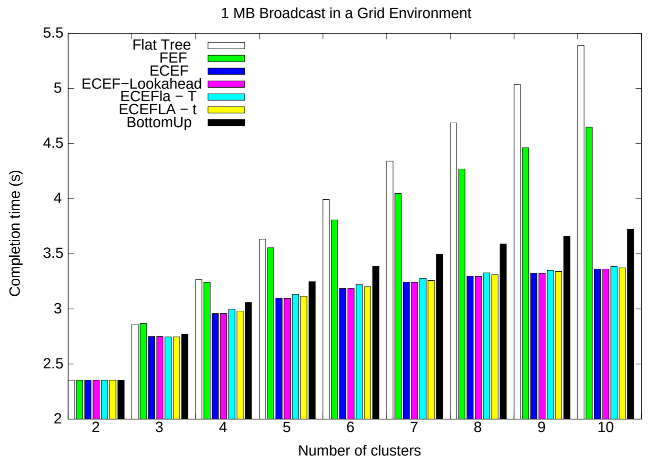

Figure 7: Simulation results for a broadcast with a reduced number of clusters

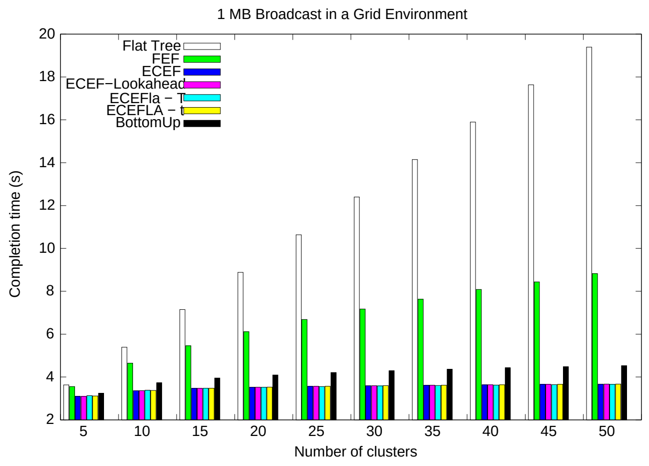

Figure 8: Simulation results for a broadcast

7 Practical Evaluation

While the previous section provides valuable information on the expected efficiency of different communication schedule heuristics, we still cannot evaluate the impact of these techniques on the performance of real implementations. In fact, most of the techniques presented in this paper may induce a scheduling cost that can affect the performanc e of the `MPI_Bcast` operation.

Table 4: Latency between different clusters (in microseconds)

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
	31 x Orsay	29 x Orsay	6 x IDPOT	1 x IDPOT	1 x IDPOT	20 x Toulouse
Cluster 0	47.56	62.10	12181.52	12187.24	12197.49	5210.99
Cluster 1	62.10	47.92	12181.52	12198.03	12195.22	5211.47
Cluster 2	12181.52	12181.52	35.52	60.08	60.08	5388.49
Cluster 3	12187.24	12198.03	60.08	-	242.47	5393.98
Cluster 4	12197.49	12195.22	60.08	242.47	-	5394.10
Cluster 5	5210.99	5211.47	5388.49	5393.98	5394.10	27.53

In order to evaluate the performance of the heuristics studied in this paper, we implemented these techniques on top of a modified version of the MagPIe library [2]. We improved MagPIe by extending it with the capability to acquire pLogP parameters and to predict the communication performance of homogeneous clusters, as explained in a previous paper [31].

Hence, to evaluate the real performance of the different heuristics, we run a test experiment using 88 machines from the GRID5000 environment, split in 6 homogeneous clusters, according to cluster map provided by Lowekamp's algorithm [17] with a tolerance rate $\rho = 30\%$. As a result, Table 4 indicates the latency between every two clusters or between two machines in the same cluster (except for the clusters that have only one single machine). Therefore, some clusters like IDPOT were subdivided in differen t homogeneous clusters, according to their real communication performance [31].

From these data, we initially try to predict the communication performanc e using different scheduling heuristics. It should be note that our framework automatically performs these predictions, combining the communication performances from both network and intra-cluster broadcast strategies. Indeed, Figure 9 presents the performance predictions for these heuristics, while in Figure 10 we present the measured times. To better evaluate the performance speed-up obtained with the use of scheduling heuristics we compare the performance of the standard `MPI_Bcast` operation provided by LAM-MPI, which uses a simple "grid unaware" binomial tree.

We observe therefore that ECEF-like heuristics achieved the best performa nce levels, with less than 3 seconds for a 4 MB message; at the opposite side, the Flat Tree strategy required almost six times more time to execute the broadcast, giving a performance that is even worst than the "grid-unaware" binomial tree algorithm traditionally used by MPI. In addition to the effective performance gain, we can also observe from these figures that performance predictions fit with a good precision the practical results. Indeed, these results demonstrate the importance of grid-aware collective communications. It also demonstra ted that our approach, which consists on combining intra and inter-cluster optimization, allows a precise performance modeling while relying on low complexity optimization techniques.

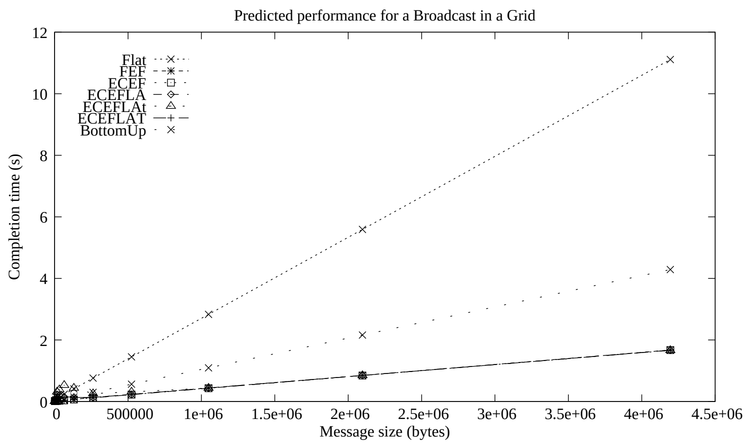

Figure 9: Predicted time for a 88 machine grid

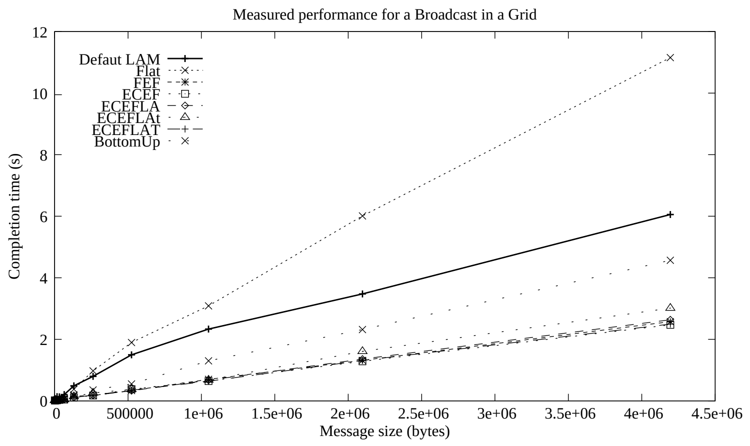

Figure 10: Measured completion time for a 88 machine grid

8 Concluding Remarks and Future Works

We have presented in this paper a new adaptive framework based on communication models and adaptive approaches for predicting and modeling performances of parallel algorithms on heterogeneous hierarchical clusters. We have defined the concept of polyalgorithmic optimization, and proposed a methodology that proceeds in two levels of adaptation to automatically and dynamically associate the most appropriate algorithm for each different cluster with efficient inter-cluster scheduling heuristics. The framework proposed determines therefore the best combinat ion algorithm-schedule and computes an efficient execution scheme that minimizes the overall execution time of a target problem. This approach was applied on an important collective communication pattern, the broadcast operation, proving the powerful of the proposed multi level adaptive scheme and the worthy of this work. As future prospects, we intend to keep validating this approach by achieving experiments on other grid environments. We also plan to integrate other existing adaptive approaches to our framework to benefit well from the powerful of these techniques, be it related to different algorithms for already supported operations or related to still unsupported collective communication primitives.

References

[1] N. T. Karonis, B. Supinski, I. Foster, W. Gropp, E. Lusk, and 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*, pp. 377–384, IEEE Computer Society, 2000.

[2] T. Kielmann, R. Hofman, H. Bal, A. Plaat, and 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*, pp. 131–140, 1999.

[3] T. Kielmann, H. Bal, S. Gorlatch, K. Verstoep, and R. Hofman, "Network performance-aware collective communication for clustered wide area systems," *Parallel Computing*, vol. 27, no. 11, pp. 1431–1456, 2001.

[4] O. Hartmann, M. Kuhnemann, T. Rauber, and 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.

[5] F. Capello, P. Fraigniaud, B. Mans, and A. Rosenberg, "An algorithmic model for heterogeneous hyper-clusters: Rationale and experience," *International Journal of Foundations of Computer Science*, vol. 16, no. 2, pp. 195–215, 2005.

[6] A. Bar-Noy and S. Kipnis, “Designing broadcasting algorithms in the postal model for message-passing systems,” *Math. Systems Theory*, vol. 27, no. 5, pp. 431–452, 1994.

[7] R. Hockney, “The communication challenge for MPP: Intel paragon and meiko cs-2,” *Parallel Computing*, vol. 20, pp. 389–398, 1994.

[8] M. Clement, M. Steed, and P. Crandall, “Network performance modelling for PM clusters,” in *Proceedings of Supercomputing*, 1996.

[9] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, “LogP - a practical model of parallel computing,” *Communication of the ACM*, vol. 39, no. 11, pp. 78–85, 1996.

[10] A. Alexandrov, M. Ionescu, K. 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 7th Annual Symposium on Parallel Algorithms and Architecture (SPAA’95)*, 1995.

[11] C. A. Moritz and M. I. Frank, “LoGPC: Modeling network contention in message-passing programs,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 4, pp. 404–415, 2001.

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

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

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

[15] R. Jain. Wiley Computer Publishing, John Wiley and Sons Inc., 1991.

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

[17] B. Lowekamp, *Discovery and Application of Network Information.* PhD thesis, Carnegie Mellon University, 2000.

[18] M. Barnett, D. Payne, R. van de Geijn, and J. Watts, “Broadcasting on meshes with wormhole routing,” *Journal of Parallel and Distributed Computing*, vol. 35, no. 2, pp. 111–122, 1996.

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

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

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

[22] P. B. Bhat, C. Raghavendra, and V. Prasanna, "Efficient collective communication in distributed heterogeneous systems," *Journal of Parallel and Distributed Computing*, vol. 2003, no. 63, pp. 251–279, 2003.

[23] P. Liu, D.-W. Wang, and 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, pp. 38–52, Springer-Verlag, 2004.

[24] J.-Y. L. Park, H.-A. Choi, N. Nupairoj, and 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)*, pp. 180–187, 1996.

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

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

[27] B. Lowekamp and A. Beguelin, "ECO: Efficient collective operations for communication on heterogeneous networks," in *Proceedings of the 10th International Parallel Processing Symposium*, pp. 399–405, 1996.

[28] N. T. Karonis, I. Foster, B. Supinski, W. Gropp, E. Lusk, and S. Lacour, "A multilevel approach to topology-aware collective operations in computational grids," tech. rep., Mathematics and Computer Science Division, Argonne National Laboratory, 2002.

[29] S. Lacour, N. T. Karonis, and I. Foster, "MPICH-G2 collective operations performance evaluation, optimizations," tech. rep., Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

[30] L. Barchet-Steffenel and 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), pp. 254–261, 2004.

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

Contents

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399