Assessing Contention Effects of All-to-All Communications on Clusters and Grids

Luiz Angelo Steffenel, Maxime Martinasso, Denis Trystram

Assessing Contention Effects of All-to-All Communications on Clusters and Grids

Luiz Angelo Steffenel
CReSTIC-SysCom, Université de Reims Champagne-Ardenne, France
Luiz-Angelo.Steffenel@univ-reims.fr

Maxime Martinasso
Denis Trystram
LIG - Laboratoire d'Informatique de Grenoble, France
{Maxime.Martinasso, Denis.Trystram}@imag.fr

Received: March 31 2008; revised: October 16 2008

***Abstract*— One of the most important collective communication patterns used in scientific applications is the *complete exchange*, also called *All-to-All*. Although efficient algorithms have been studied for specific networks, general solutions like those available in well-known MPI distributions (e.g. the MPI_Alltoall operation) are strongly influenced by the congestion of network resources. In this paper we present an integrated approach to model the performance of the All-to-All collective operation, which consists in identifying a contention signature that characterizes a given network environment, using it to augment a contention-free communication model. This approach, assessed by experimental results, allows an accurate prediction of the performance of the All-to-All operation over different network architectures with a small overhead. We also discuss the problem of network contention in a grid environment, studying some strategies to minimize the impact of contention on the performance of an All-to-All operation.**

***Index Terms*— MPI, all-to-all, total exchange, network contention, performance modeling, computational grid, personalized many-to-many communications**

I. Introduction

One of the most important collective communication patterns for scientific applications is the *total exchange* [4] (also called *All-to-All*), in which each process holds n different data items that should be distributed among the n processes, including itself. An important example of this communication pattern is the All-to-All operation, where all messages have the same size m.

Although efficient All-to-All algorithms have been studied for specific networks structures like meshes, hypercubes, tori and circuit-switched butterflies, general solutions like those found in well-known MPI distributions rely on direct point-to-point communications among the processes. Because all communications are started simultaneously, architecture independent algorithms are strongly influenced by the saturation of network resources and subsequent loss of packets - the network contention.

In the first part of this paper we present a new approach to model the performance of the All-to-All collective operation. Our strategy consists in identifying a *contention signature* that characterizes a given network environment. Using such *contention signature*, we are able to accurately predict the performance of the All-to-All operation, with an arbitrary number of processes and message sizes. To demonstrate our approach, we present experimental results obtained with different network architectures (Fast Ethernet, Gigabit Ethernet and Myrinet). We believe that this model can be extremely helpful on the development of application performance prediction frameworks such as PEMPIs [20] and task schedulers.

Later, we focus on the performance modeling of All-to-All in a grid environment. We observe that contention at the wide-area level not only exists but represents a major component on the communication performance. We propose therefore an algorithm adapted for inter-cluster message exchanges. This algorithm minimizes the impact of network contention and, by extension, allows a better performance modeling. Indeed, we show that this algorithm not only outperforms the traditional All-to-All algorithms on a grid environment but also allows an accurate performance modeling using only elements derived from our local-area models.

This paper is therefore organized as follows: Section II presents a survey of performance modeling under communication contention. Section III presents the network models used in this paper and we formalize the *total exchange* problem, as well as some performance lower bounds. In Section IV we propose a strategy to characterize the *contention signature* of a given network and for instance, to predict the performance of the All-to-All operation. Section V validates our model against experimental data obtained on different network architectures (Fast Ether-

net, Gigabit Ethernet and Myrinet). In Section VI we provide a study case for predicting the performance of the All-to-All algorithm on a grid platform. Finally, Section VII presents some conclusions and the future directions of our work.

II. Related Works

In the *All-to-All* operation, every process holds $m \times n$ data items that should be equally distributed among the n processes, including itself. Because general implementations of the All-to-All collective communication rely on direct point-to-point communications among the processes the network can easily become saturated, and by consequence, degrades the communication performance. Indeed, Chun and Wang [5][6] demonstrated that the overall execution time of intensive exchange collective communications are strongly dominated by the network contention and congestive packet loss, two aspects that are not easy to quantify. As a result, a major challenge on modeling the communication performance of the All-to-All operation is to represent the impact of network contention.

Unfortunately, most communication models like those presented by Christara *et al.* [4] and Pjesivac-Grbovic *et al.* [22] do not take into account the potential impacts of network contention. Indeed, these works usually represent the All-to-All operation as parallel executions of the *personalized one-to-many* pattern [15], as presented by the linear point-to-point model below, where α is the start-up time (the latency between the processes), $\frac{1}{\beta}$ is the bandwidth of the link, m represents the message size in bytes and n corresponds to the number of processes involved in the operation:

$$T = (n-1) \times (\alpha + \beta m) \tag{1}$$

The development of contention-aware communication models is relatively recent, as shown by Grove [11]. For instance, Adve [1] presented one of the first models to take into account the effects of resource contention. This model considers that the total execution time of parallel programs is the sum of four components, namely:

$$\begin{aligned} T = t_{computation} + t_{communication} \\ + t_{resource-contention} + t_{synchronization} \end{aligned} \tag{2}$$

While conceptually simple, this model was non-trivial in practice because of the non-deterministic nature of resource contention, and because of the difficulty to estimate average synchronization delays.

While the non-deterministic behavior of the network contention is a major obstacle to modeling communication performance, some authors suggested a few techniques to adapt the existing models. As consequence, Bruck [2] suggested the use of a *slowdown factor* to correct the performance predictions. Similarly, Clement *et al.* [7] introduced a technique that suggested a way to account contention in shared networks such as non-switched Ethernet, consisting in a contention factor γ that extends the linear communication model T:

$$T = \alpha + \beta \times m \times \gamma \tag{3}$$

where γ is equal to the number of processes. A restriction on this model is that it assumes that all processes communicate simultaneously, which is only true for a few collective communication patterns. Anyway, in the cases where this assumption holds, they found that this simple contention model enhanced the accuracy of their predictions for essentially zero extra effort.

The use of a contention factor was supported by the work of Labarta et *al.* [19], that intent to approximate the behavior of the network contention by considering that if there are m messages ready to be transmitted, and only b available buses, then the messages are serialized in $\lceil \frac{m}{b} \rceil$ communication waves. Also, König *et al.* [18] have shown indeed that some All-to-All algorithms that are optimal with unlimited buffers become less efficient when communications depend on restricted buffers sizes.

A similar approach was followed by Jeannot *et al.* [14], who designed scheduling algorithms for data redistribution through a backbone. In their work, they suppose that at most k communications can be performed at the same time without causing network contention (the value of k depending on the characteristics of the platform). Using the knowledge of the application transfer pattern, they proposed two algorithms to schedule the messages transfers, performing an application-level congestion control that in most cases outperforms the TCP contention control mechanism.

Most recently, some works aimed to design contention-aware performance models. For instance, LoGPC [21] presents an extension of the LogP model that tries to determine the impact of network contention through the analysis of k-ary n-cubes. Unfortunately, the complexity of this analysis makes too hard the application of such model in practical situations.

Another approach to include contention-specific parameters in the performance models was introduced by Chun [5]. In his work, the contention is considered as a component of the communication latency, and by consequence, the model uses different latency values depending on the message size. Although easier to use than LoGPC, this model does not take into account the number of messages passing in the network nor the link capacity, which are clearly related to the occurrence of network contention.

III. Definitions

A. Network Models

In this work we assume that the network is fully connected, which corresponds to 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.

Transmission Model: We use Hockney's notation [12] to describe our transmission model. Therefore, the time to send a message of size $w_{i,j}$ from a process p_i to another process p_j, is $\alpha + w_{i,j}\beta$, where α is the start-up time (the

communication latency between the processes) and $\frac{1}{\beta}$ is the bandwidth of the link. As in this paper we assume that all links have the same latency and bandwidth, and because we only investigate the regular version of the MPI_Alltoall operation where all messages have the same size m, $\forall i, \forall j, w_{i,j} = m$, and therefore the time to send a message from a process p_i to a process p_j is $\alpha + m\beta$.

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 synchronization model corresponds to the execution of the MPI_Alltoall operation, used as reference in this work.

B. Problem Definitions

In the *total exchange* problem, n different processes hold each one n data items that should be evenly distributed among the n processes, including itself. Because each data item has potentially different contents and sizes according to their destinations, all processes engage a total exchange communication pattern. Therefore, a total exchange operation will be complete only after all processes have sent their messages to their counterparts, and received their respective messages.

Formally, the *total exchange problem* can be described using a weighted digraph $dG(V, E)$ of order n with $V = \{p_0, ..., p_{n-1}\}$. This digraph is called a message exchange digraph or MED for short. In a MED, the vertices represent the process nodes, and the arcs represent the messages to be transmitted. An integer $w(e)$ is associated with each arc $e = (p_i, p_j)$, representing the size of the message to be sent from process p_i to process p_j. Note that there is not necessarily any relationship between a MED and the topology of the interconnection network.

The port capacity of a process for transmission is the number of other processes to which it can transmit simultaneously. Similarly, the port capacity for reception is the number of other processes from which it can receive simultaneously. We will concentrate on the performance modeling problem with all port capacities restricted to one for both transmitting and receiving. This restriction is well-known in the literature as *1-port full-duplex*.

C. Notation and lower bounds

In this section, we present theoretical bounds on the minimum number of communications and on the bandwidth for the general message exchange problem. The number of communications determines the number of start-ups, and the bandwidth depends on the message weights.

Given a MED $dG(V; E)$, we denote the in-degree of each vertex $p_i \in V$ by $\Delta_r(p_i)$, and the out-degree by $\Delta_s(p_i)$. Let $\Delta_r = \max_{p_i \in V}\{\Delta_r(p_i)\}$ and $\Delta_s = \max_{p_i \in V}\{\Delta_s(p_i)\}$. Therefore, we obtain the following straightforward bound on the number of start-ups.

Claim 1. *The number of start-ups needed to solve a message exchange problem on a digraph $dG(V; E)$ without message forwarding is at least* $\max(\Delta_s, \Delta_r)$.

Given a MED $dG(V, E)$, the bandwidth bounds are determined by two obvious bottlenecks for each vertex - the time for it to send its messages and the time for it to receive its messages. Each vertex p_i has to send messages with sizes $\{w_{i,j} \mid j = 0 \dots n-1\}$. The time for all vertices to send their messages is at least $t_s = \max_i \sum_{j=0}^{n-1} w_{i,j}\beta$. Similarly, the time for all vertices to receive their messages is at least $t_r = \max_j \sum_{i=0}^{n-1} w_{i,j}\beta$.

Claim 2. *The time to complete a personalized exchange is at least* $\max\{t_s, t_r\}$.

We can combine the claims about the number of start-ups and the bandwidth when message forwarding is not allowed.

Claim 3. *If message forwarding is not allowed, and either the model is synchronous or both maxima are due to the same process, the time to complete a personalized exchange is at least* $\max(\Delta_s, \Delta_r) \times \alpha + \max\{t_s, t_r\}$.

Because in this paper we do not assume messages forwarding, the fan-in and fan-out of a process must be $(n-1)$. Further, as we consider messages to be the same size and the network to be homogeneous, we can simplify Claim 3 so that the following bound holds.

Proposition 1. *If message forwarding is not allowed, and all messages have size* m*, and both bandwidth and latency are identical to any connection between two different processes p_i and p_j, the time to complete a total exchange is at least* $(n-1) \times \alpha + (n-1) \times \beta m$.

Proof. The proof is trivial, as the time to complete a total exchange is at least the time a single process needs to send one message to each other process. ∎

IV. Contention Signature Approach

To cope with this problem and to model the contention impact on the performance of the All-to-All operation, we adopt an approach similar to Clement *et al.* [7], which considers the contention sufficiently linear to be modeled. Our approach, however, tries to identify the behavior of the All-to-All operation with regard to the theoretical lower bound (Proposition 1) on the *1-port* communication model. In our hypothesis, the network contention depends mostly on the physical characteristics of the network (network cards, links, switches), and consequently, the ratio between the theoretical lower bound and the real performance represents a "*contention signature*" of the network. Once identified the *signature* of a network, it can be used in further experiments to predict the communication performance, provided that the network infrastructure does not change.

Initially, we consider communication in a contention-free environment. In this case, a process that sends messages of size m to $n-1$ processes needs at least $(n-1)\times\alpha+(n-1) \times m\beta$ time units. Further, by the properties of the *1-port* communication model, the total communication time of the All-to-All operation must be at least $(n-1) \times \alpha + (n-1)\times m\beta$ time units if all processes start communicating simultaneously, as stated by Proposition 1.

In the case of the All-to-All operation, however, the intensive communication pattern tends to saturate the

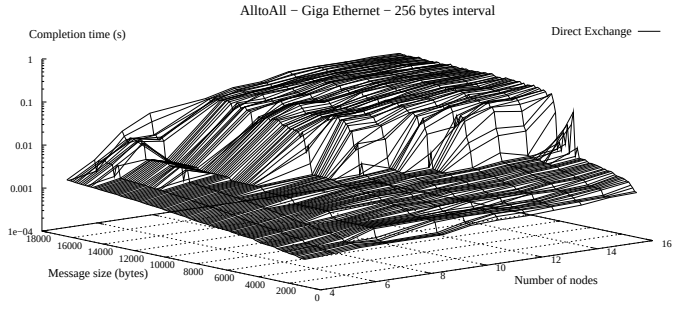

Fig. 1. Non-linearity of communication cost with small messages

network, causing message delays and packet loss that strongly impact on the communication performance of this collective communication. In this network congestion situation, traditional models such as those presented by Christara [4] do not hold anymore, even if the communication pattern has not changed.

Therefore, our approach to model the performance of the MPI_Alltoall operation despite network contention consists on determining a *contention ratio* γ that express the relationship between the theoretical performance (lower bound) and the real completion time. For simplicity, we consider that this *contention ratio* γ is constant and depends exclusively on the network characteristics. Therefore, the simplest way to integrate this *contention ratio* γ in our performance model would be as follows:

$$T = (n-1) \times (\alpha + m\beta \times \gamma) \tag{4}$$

A. Non-linear aspects

Although the performance model augmented by use of the *contention ratio* γ improves the accuracy of the predictions, we observe nonetheless that some network architectures are still subject to performance variations according to the message size. To illustrate this problem, we present in Fig. 1, a detailed mapping of the communication time of the MPI_Alltoall operation in a Gigabit Ethernet network. We observe that the communication time does not increase linearly with the message size, but instead, present a non-linear behavior that prevents our model to accurately predict the performance when dealing with small messages.

To cope with this non-linearity, we propose an extension of the *contention ratio* model to better represent this phenomenon when messages are sufficiently large. Hence, we augment the model with a new parameter δ, which depends on the number of processes but also on a given message size M. As a consequence, the association of different equations helps to define a more realistic performance model for the MPI_Alltoall operation, as follows:

$$T = \begin{cases} (n-1) \times (\alpha + m\beta \times \gamma) & if\, m < M \\ (n-1) \times (\alpha + m\beta \times \gamma + \delta) & if\, m \geq M \end{cases} \tag{5}$$

V. Validation

To validate the approach proposed in this paper, this section presents our experiments to model the performance of MPI_Alltoall operation using three network architectures, Fast Ethernet, Gigabit Ethernet and Myrinet. As previously explained, our approach compares the expected and real performance of the MPI_Alltoall operation using a sample experiment with n' nodes; the relationship between these two measures allows us to define the γ and δ parameters that characterize the "network contention signature".

To obtain these parameters, we compare the sample data obtained from both theoretical lower bound and experimental measure, while varying the message size. Indeed, the lower bound comes from Proposition 1, with parameters α and β obtained from a simple point-to-point measure. The parameters γ and δ are obtained through a linear regression with the Generalized Least Squares method, comparing at least four measurement points in order to better fit the performance curve.

The different experiments presented in this paper represent the average of 100 measures for each set of parameters (message size, number of processes), and were conducted over two clusters of the Grid'5000 network[1]:

The *icluster2* cluster, located at INRIA-Rhone-Alpes, composed of 104 dual Itanium2 nodes at 900 MHz, used for the experiments with the Fast Ethernet network (5 Fast Ethernet switches - 20 nodes per switch - interconnected by 1 Gigabit Ethernet switch) and the Myrinet 2000 network (one 128 ports M3-E128 Myrinet switch). All machines run Red Hat Enterprise Linux AS release 3, with kernel version 2.4.21.

The *GdX* (GriD'eXplorer) cluster, operated by INRIA-Futurs. This cluster includes 216 nodes with dual AMD Opteron processors at 2 GHz running Debian Linux kernel 2.6.8 and a Broadcom Gigabit Ethernet network.

A. Fast Ethernet

Taking as basis the measured performance for a 24 machines network, we were able to approximate the performance of the Fast Ethernet network with a *contention ratio* $\gamma = 1.0195$. Indeed, this relatively small difference must be considered in the light of the retransmission policy: although the communication latency (and therefore the timeouts) is relatively small (around 60 μs), the reduced bandwidth of the links minimizes the impact of the retransmission of a lost packet. More important, we observe that the experimental measures behave like an affine equation, showing a start-up cost usually not considered by the traditional performance model which corresponds to the δ parameter proposed in our model. Therefore, we determined $\delta = 8.23\,ms$ for messages larger than $M = 2\,kB$, which means that each simultaneous communication induces an overload of $8.23\,ms$ to the completion time of the All-to-All operation. Applying both γ and δ parameters to the performance model we were able to approximate our predictions from the performance of the MPI_Alltoall operation with an arbitrary number of processes, as demonstrate in Fig. 2a. We observe indeed

[1] *http://www.grid5000.fr/*

that our error rate is usually smaller than 10% when there are enough processes to saturate the network, as presented in Fig. 2b.

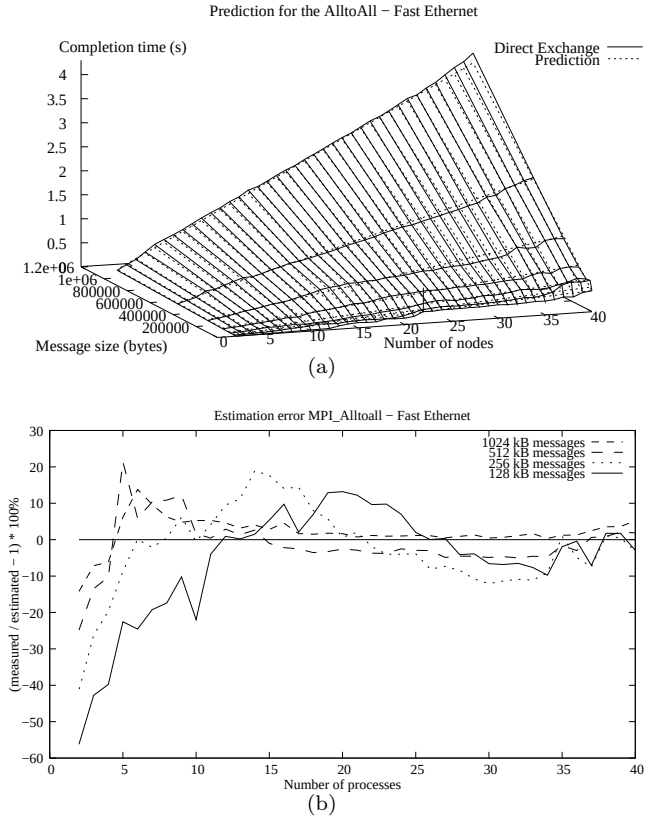

Fig. 2. Performance prediction on a Fast Ethernet network

B. Gigabit Ethernet

To compute the *contention ratio* γ and a *start-up cost* δ, we use sample data for an arbitrary number of processes. Indeed, we chose in this example the results for an execution of the All-to-All operation with 40 processes (one by machine). Using linear regression on these data we obtain $\gamma = 4.3628$ and $\delta = 4.93\,ms$ (to be used only for messages larger than $M = 8\,kB$). As a result, the performance predictions from our model correspond to the curve presented on Fig. 3a. As in the case of the Fast Ethernet network, the error rate is quite small when the network becomes saturate, even when we consider different message sizes (Fig. 3b).

C. Myrinet

Although the two previous experiments give important proofs on the validity of our modeling method, they share many similarities on both network architecture and transport protocol (TCP/IP). To ensure that our method is not bounded to a specific infrastructure, we chose to validate our performance model also in a Myrinet network, using the *gm* transport protocol. Because of the *Myrinet+gm* stack differs considerably from the Ethernet+TCP/IP

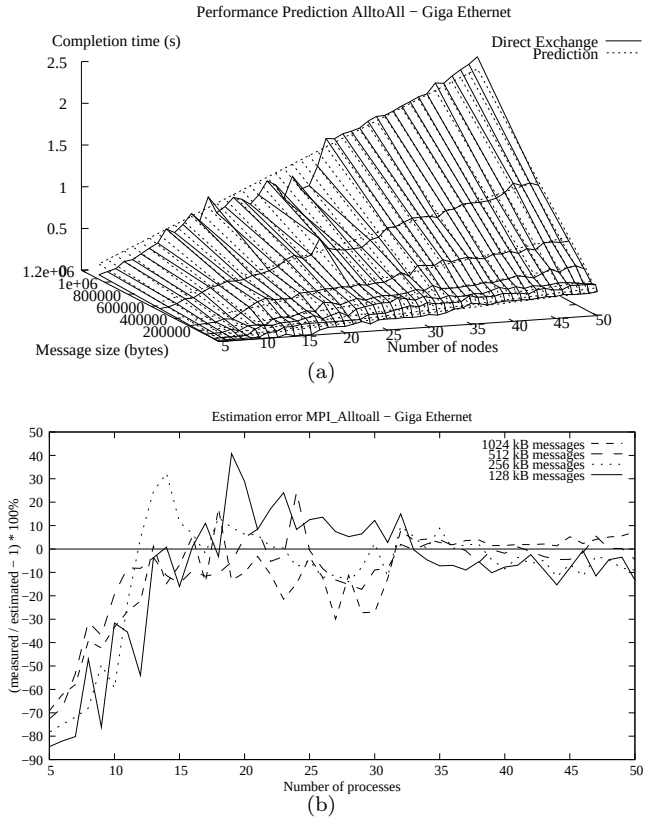

Fig. 3. Performance prediction on a Gigabit Ethernet network

stack, any systematic behavior introduced into our sampling data by these architectures should be exposed.

Indeed, the Myrinet network differs from Ethernet-based architectures due to an *start-up* cost almost inexistent (one of the main characteristics of the *Myrinet+gm* stack). Indeed, we were able to fit the performance of a 24-processes All-to-All operation using only the *contention ratio* $\gamma = 2,49754$ (as the linear regression pointed a *start-up cost* δ smaller than 1 microsecond).

Nevertheless, when applying this factor to an arbitrary number of machines, as presented in Fig. 4a, we observe that our predictions do not follow the experimental data as observed before with Fast Ethernet and Gigabit Ethernet. Actually, a close look at the error rate (Fig. 4b) indicates that network saturation occurs only when there are more then 40 communicating processes (evidenced by the constant error rate from that point). These results demonstrate the limitations of our approach: while a *contention ratio* may provide precise performance predictions, it depends on the data used to define the network signature. By using reference data from a partially saturated network we are subjected to inaccurate approximations (even if they are better than the contention unaware predictions).

VI. Problem of Total Exchange between Two Clusters

Multicluster and computational grids are popular heterogeneous environments used by the HPC community,

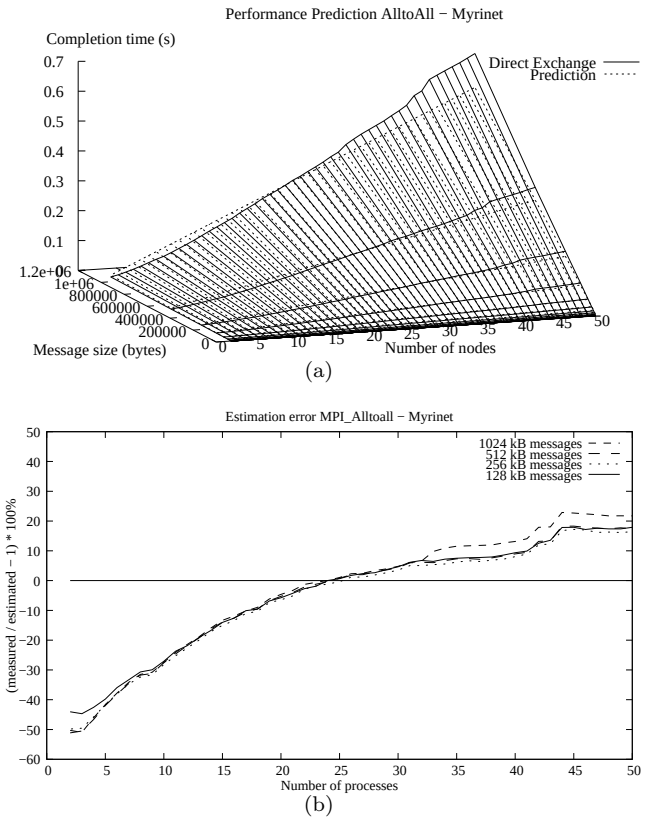

Fig. 4. Performance prediction on a Myrinet network

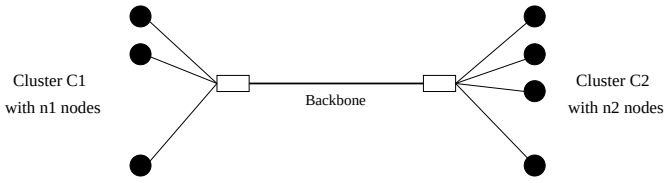

Fig. 5. Architecture for the redistribution problem

and part of our efforts concentrate on the development of efficient algorithms and performance models for these environments. More specifically, we must deal with communications costs that differ significantly if they are local or remote. Without loss of generality, let us assume two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ with respectively n_1 nodes and n_2 nodes, as represented in Figure 5. A network, called a backbone, interconnects the two 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. Based on that topology inter cluster communications are never faster than communication within a cluster.

Let us suppose that an application is running and using both clusters (for example, a code coupling application). One part of the computation is performed on cluster $\mathcal{C}_1$ and the other part on cluster $\mathcal{C}_2$. During the application, data must be exchanged from $\mathcal{C}_1$ to $\mathcal{C}_2$ using the *all-toall* pattern. Altogether, this means that we will have to transfer $(n_1 + n_2)^2$ messages over different network environments. The data of all these messages are different

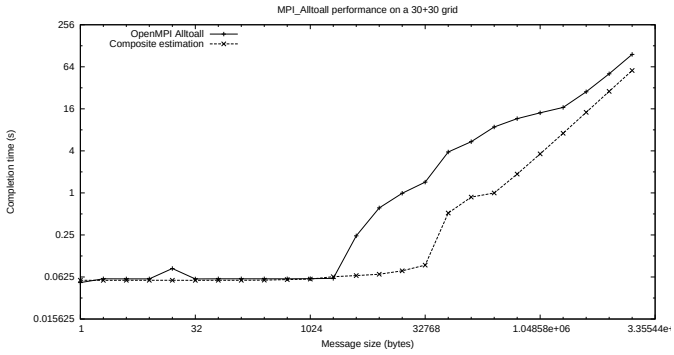

Fig. 6. Measured and predicted performance for the standard MPI_Alltoall in a grid

but the size of the messages are the same and is given and called m (in bytes). Several MPI libraries (OpenMPI, MPICH2, etc.) implement the *allltoall* routine 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 $\mathcal{C}_1$ or from $\mathcal{C}_2$ to $\mathcal{C}_2$) or between the two clusters.

From the homogeneous cluster prediction model presented in the first part of this paper, which allows a quite accurate representation on the performance of local-area networks, we initially proceeded our multi-cluster performance modelling by composing local (contention-aware) and remote communications. Estimate completion time corresponds to the sum of the contention-corrected local-area cost ($T_{\mathcal{C}_n}$) and the wide-area communication cost (a function of wide-area latency α_w and bandwidth β_w) :

$$T = max(T_{\mathcal{C}_1}, T_{\mathcal{C}_2}) + max(n_1, n_2) \times (\alpha_w + \beta_w \times m) \quad (6)$$

Unfortunately, this simple strategy fails to represent the operation of the MPI_Alltoall in a grid. Figure 6 compares estimate and measured completion time of the MPI_Alltoall implementation from OpenMPI in a grid with two clusters of 30 machines each. From this experience we observe that not only contention affects also wide-area links but also that local area communications play a small role in the overall performance. Of course, one could try to define additional parameters for the wide-area communications, but the final model would be too complex to be useful in real situation. Instead, we decided to address this problem by redefining the All-to-All problem against the challenges that characterize a grid environment.

A. Minimizing the impact of contention on the backbone

When dealing with wide-area networks, the most important factor to be considered is the time a message takes to be delivered. Indeed, in addition to the geographical distance, message are subjected to network protocols heterogeneity, message routing and transient interferences on the backbone.

Actually, popular algorithms for collective communications on grids (such as the ones implemented in PACX MPI [9] , GridMPI [10] and MagPIe [16]) try to minimize communications over the wide-area network by defining a single coordinator in every cluster, which participates in the inter-cluster data transfers across the wide-area backbone. By minimizing the number of WAN communication steps, we reduce the probability of inducing contention and accumulating transmission delays on the messages.

However, a single communication between each cluster is an approach inappropriate for the MPI_Alltoall operation. First, it induces additional communication steps to/from the cluster coordinator, which becomes a bottleneck. Second, this approach is not optimal concerning the usage of the wide-area bandwidth, as wide-area backbones are designed to support simultaneous transfers [3]. Hence, in order to improve the performance in a WAN, we need to change the MPI_Alltoall algorithm strategy.

B. The Local Group($\mathcal{LG}$) algorithm

To cope with this problem, we try to minimize wide-area communication steps in a different way. Actually, most of the complexity of the All-to-All problem resides on the need to exchange *different* messages through different networks (local and distant). The traditional implementation of the MPI_Alltoall operation cannot differentiate these networks, leading to poor performances. However, if we assume that communications between clusters are slower than intra-clusters ones, it might be useful to collect data in the local level before sending it in parallel through the backbone, in a single communication step.

As a consequence, we propose in [13] a grid-aware solution which performs on two phases. In the first phase only local communications are performed. During this phase the total exchange is performed on local nodes on both cluster and extra buffers are prepared for the second (inter-cluster) phase. During the second phase data are exchanged between the clusters. Buffers that have been prepared during the first phase are sent directly to the corresponding nodes in order to complete the total exchange.

More precisely, our algorithm called *Local Group* or simply $\mathcal{LG}$ works as follow. Without loss of generality, let us assume that cluster $\mathcal{C}_1$ has less nodes than $\mathcal{C}_2$ ($n_1 \leq n_2$).

Nodes are numbered from 0 to $n_1 + n_2 - 1$, with nodes from 0 to n_1-1 being on $\mathcal{C}_1$ and nodes from n_1 to n_1+n_2-1 being on cluster $\mathcal{C}_2$. We call $\mathcal{M}_{i,j}$ the message (data) that has to be sent from node i to node j. The phases are sum-up in Algorithm 1.

1) First phase: During the first phase, we perform the local exchange: Process i sends $\mathcal{M}_{i,j}$ to process j, if i and j are on the same cluster. Then it prepares the buffers for the remote communications. On $\mathcal{C}_1$ data that have to be send to node j on $\mathcal{C}_2$ is first stored to node j mod n_1. Data to be sent from node i on $\mathcal{C}_2$ to node j on $\mathcal{C}_1$ is stored on node $\lfloor i/n_1 \rfloor \times n_1 + j$.

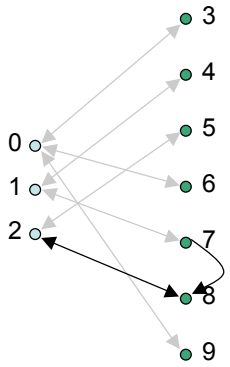

Fig. 7. Example of the 2 phases of the algorithm

Algorithm 1 The $\mathcal{LG}$ (*Local Group*) algorithm when $n_1 \leq n_2$

```
// Local Phase
  for i = {0, ..., (n1 + n2) − 1} do in parallel
    for j = {0, ..., (n1 + n2) − 1} do
      if i < n1 // the sender is on C1
          send M_{i,j} to j mod n1
      else // the sender is on C2
          if j ≥ n1 // the receiver is on C2
              send M_{i,j} to j
          else // the receiver is on C1
              send M_{i,j} to ⌊i/n1⌋ × n1 + j
// Inter-cluster Phase
for s = {1, ..., ⌈n2/n1⌉}
    for i = {1, ..., n1 − 1} do in parallel
      if (i + s × n1 < n1 + n2)
          exchange messages between i and
j = i + s × n1
```

2) Second phase: During the second phase only n_2 inter-cluster communications occurs. This phase is decomposed in $\lceil n_2/n_1 \rceil$ steps with at most n_1 communications each. Steps are numbered from 1 to $\lceil n_2/n_1 \rceil$ During step s node i of $\mathcal{C}_1$ exchange data stored in its local buffer with node $j = i+n_1 \times s$ on $\mathcal{C}_2$ (if $j < n_1+n_2$). More precisely i sends $\mathcal{M}_{k,j}$ to j where $k \in [0, n_1]$ and j sends $\mathcal{M}_{k,i}$ to i where $k \in [n_1 \times s, n_1 \times s + n_1 - 1]$.

3) Example: Suppose that $n_1 = 3$ and $n_2 = 7$. What happens to the message $\mathcal{M}_{7,2}$ (*i.e* the messages that goes from node 7 on cluster $\mathcal{C}_2$ to node 2 on $\mathcal{C}_1$)? This is illustrated in Figure 7. During the first phase it is stored on node $\lfloor 7/3 \rfloor \times 3 + 2 = 8$ on $\mathcal{C}_2$. Then during the second phase it is sent to node 2 during the step $s = 2$: node 8 sends $\mathcal{M}_{6,2}$, $\mathcal{M}_{7,2}$ and $\mathcal{M}_{8,2}$ to node 2. During this step nodes 1 and 7 and node 0 and 6 exchange data as well, while in the previous step node 0 and 3, 4 and 1 and 5 and 2 exchange data and in the last step only 0 and 9 exchange data.

C. Comparison with the standard Total Exchange algorithm

As our algorithm tries to minimize the number of inter-cluster communications between the clusters, we need only $2 \times \max(n_1, n_2)$ messages of size $m \times \min(n_1, n_2)$ in both directions against $2 \times n_1 \times n_2$ messages of size m in the traditional algorithm. For instance, the exchange of data between two clusters with the same number of processes will proceed in one single communication step of the second phase. At the other hand, if $n_2 \gg n_1$, the total number of communication steps will be similar to the traditional algorithm.

An important difficulty that arises in the implementation of the algorithm is how to determine if a node belong to the first cluster or to the second cluster. Indeed, this information is the only extra information required to run the algorithm compared to the standard one. We propose several approaches:

- a simple solution is to extend the MPI API by adding a multi-cluster *alltoall* call where the users passes the value of n_1 and n_2 assuming that the first n_1 nodes of the machine file belong to cluster 1 and the n_2 nodes to cluster 2. The problem of such an approach is to port old MPI code to the multi-cluster version.
- At the beginning of the execution MPI can run some network benchmark in order to try to guess the topology of the infrastructure. This is what already does OpenMPI in the case of multiple network interfaces.
- GridMPI has an interesting approach that consists in launching a different MPI "*client*" on each cluster, each client being interconnected to the other through a GridMPI server. Based on that, for each MPI collective communication call GridMPI is able to determine the cluster a node belongs to.
- another solution consists in modifying the MPI machine file by adding keywords that tells to which cluster a node belongs to, such as in MagPIe [16].

As our algorithm minimizes the number of inter-cluster communications, it is also wide-area optimal since it ensures that a data segment is transferred only once between two clusters separated by a wide-area link. Additionally, wide-area transmissions pack several messages together, reducing the impact of transient interferences on the backbone. Hence, Figure 8 presents a comparison between the traditional algorithm used by OpenMPI and the $\mathcal{LG}$ algorithm. We observe that $\mathcal{LG}$ improves the performance of the MPI_Alltoall operation, reaching over than 50% of performance improvement comparing to the traditional strategy.

D. Modeling approach

As shown above, the algorithm we propose to optimize All-to-All communications in a grid environment rely on the relative performances of both local and remote networks. Indeed, we extend the total exchange among nodes in the same cluster in order to reduce transmissions through the backbone.

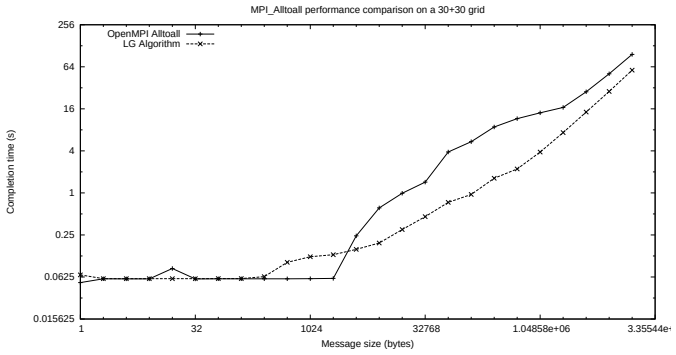

Fig. 8. Performance comparison between OpenMPI and $\mathcal{LG}$ algorithms

This approach has two consequences for performance prediction: first, it prevents contention in the wide-area links, which are hard to model. Second, the transmission of messages packed together is less subjected to network interferences. For instance, we can design a performance model by composing local-area predictions obtained with our contention ratio model and wide-area predictions that can be easily obtained from traditional methods. Hence, an approximate model would consider the following parts, where $\mathcal{T}_{C_n}$ corresponds to Equation 5:

$$T = max(T_{\mathcal{C}_1}, T_{\mathcal{C}_2}) + \lceil n_2/n_1 \rceil \times (\alpha_w + \beta_w \times m \times n_1) \quad (7)$$

E. Experimental validation

To validate the algorithm we propose in this paper, this section presents our experiments to evaluate the performance of the MPI_Alltoall operation with two clusters connected through a backbone.

These experiments were conducted over two clusters of the Grid'5000 platform[2], one located in Nancy and one located in Rennes, approximately 1000 Km from each other. Both clusters are composed of identical nodes (dual Opteron 246, 2 GHz) locally connected by a Gigabit Ethernet network and interconnected by a private backbone of 10 Gbps. All nodes run Linux, with kernel 2.6.13 and OpenMPI 1.1.4. The measures were obtained with the *broadcast-barrier* approach [8].

To model the communication performance of both *inter-cluster* and *intra-cluster* communications we use the *parameterised LogP* model (*pLogP*) [16]. The *pLogP* parameters for both local and distant communications were obtained with the method described in [17]. To model the contention at the local level we used $\gamma = 2.6887$ and $\delta = 0.005039$ for $M >= 1KB$ parameters obtained from the method of the least squares as described in [23].

Therefore, in Figure 9 we compare the performance predictions obtained with Equation 7 against the effective completion time of the $\mathcal{LG}$ algorithm. We observe that prediction fit with a good accuracy to the real

[2] *http://www.grid5000.fr/*

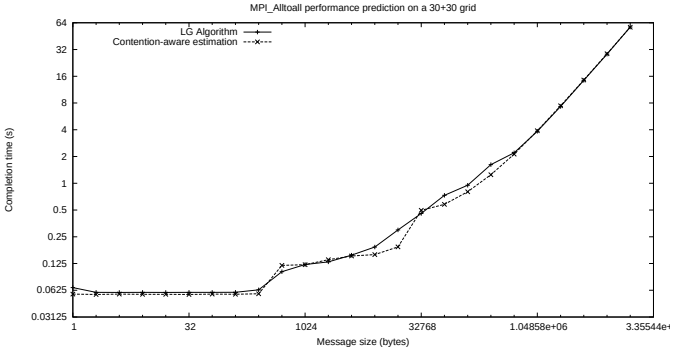

Fig. 9. Performance predictions for the $\mathcal{LG}$ algorithm

execution times, which is not possible with the traditional MPI_Alltoall algorithm. Indeed, the new algorithm minimizes the impact of distant communications, concentrating the contention problems at the local level. Because we are able to predict the performance of local communications even under contention, we can therefore establish an accurate performance model adapted to grid environments.

VII. Conclusions and Future Works

In this paper we address the problem of modeling the performance of *Total Exchange* communication operations, usually subject to important variations caused by network contention. Because traditional performance models are unable to predict the real completion time of an All-to-All operation, we try to cope with this problem by identifying the *contention signature* of a given network. In our approach, two parameters γ and δ are used to augment a linear performance model in order to fit the performance of the MPI_Alltoall operation. Because these parameters characterize the network contention and are independent of the number of communicating processes, they can be used to accurately predict the communication performance when communications tend to saturate the network. Indeed, we demonstrate our approach through experiments conducted on popular network architectures, Fast Ethernet, Gigabit Ethernet and Myrinet.

We also address the problem of modeling the MPI_Alltoall operation in a grid environment represented by two clusters. We demonstrate that network contention also affects wide-area communications but, instead of developing a new complex performance model, we observed the behaviour of MPI_Alltoall and proposed a grid-aware algorithm that reduces drastically the impact of contention, improving drastically the communication performance. Additionally, we are able to predict the overall performance of this new algorithm by simply extending our local-area contention-aware model, which represents an important advantage when comparing with the traditional MPI_Alltoall implementation.

Our efforts now are concentrating on the development of grid-optimized algorithms for the generalized total exchange problem (represented by the MPI_Alltoallv operation), where processes exchange messages of different sizes. By developing algorithms that minimize the network contention we may be able to improve the performance of the operation at the same time as we allow a simple and accurate performance modeling.

Acknowledgments

We are grateful to the anonymous referees for many comments and helpful suggestions which helped us improve the focus of the paper.

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).

References

[1] Vikram Adve. *Analysing the Behavior and Performance of Parallel Programs*. PhD thesis, University of Wisconsin, Computer Sciences Department, 1993.

[2] Jehoshua Bruck, Ching-Tien Ho, Shlomo Kipnis, Eli Upfal, and Derrick Weathersby. Efficient algorithms for all-to-all communications in multiport message-passing systems. *IEEE Transactions on Parallel and Distributed Systems*, 8(11):1143–1156, November 1997.

[3] Henri Casanova. Network modeling issues for grid application scheduling. *International Journal of Foundations of Computer Science*, 16(2):145–162, 2005.

[4] Christina Christara, Xiaoliang Ding, and Ken Jackson. An efficient transposition algorithm for distributed memory computers. In *Proceedings of the High Performance Computing Systems and Applications*, pages 349–368, 1999.

[5] Anthony Tam Tat Chun. *Performance Studies of High-Speed Communication on Commodity Cluster*. PhD thesis, University of Hong Kong, 2001.

[6] Anthony Tam Tat Chun and Cho-Li Wang. Realistic communication model for parallel computing on cluster. In *Proceedings of the International Workshop on Cluster Computing*, pages 92–101, 1999.

[7] Mark Clement, Michael Steed, and Phyllis Crandall. Network performance modelling for PM clusters. In *Proceedings of Supercomputing*, 1996.

[8] Bronis R. de Supinski and Nicholas T. Karonis. Accurately measuring MPI broadcasts in a computational grid. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC'99)*, August 1999.

[9] Edgar Gabriel, Michael Resch, Thomas Beisel, and Rainer Keller. Distributed Computing in a Heterogenous Computing Environment. In *EuroPVMMPI'98*, number 1497 in LNCS, Liverpool, UK, sep 1998. Springer-Verlag.

[10] Gridmpi. www.gridmpi.org.

[11] Duncan Grove. *Performance Modelling of Message-Passing Parallel Programs*. PhD thesis, University of Adelaide, 2003.

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

[13] Emmanuel Jeannot and Luiz Angelo Steffenel. Fast and efficient total exchange on two clusters. In *Proceedings of the 13th International Conference on Parallel Computing (EURO-PAR 2007)*, LNCS 4641, pages 848–857, August 2007.

[14] Emmanuel Jeannot and Frédéric Wagner. Two fast and efficient message scheduling algorithms for data redistribution through a backbone. In *Proceedings of the IPDPS*, 2004.

[15] S. Lenart Johnssonn and Ching-Tien Ho. Optimum broadcasting and personalized communication in hypercubes. *IEEE Transactions on Computers*, 38(9):1249–1268, September 1989.

[16] Thilo Kielmann, Henri Bal, Sergey Gorlatch, Kees Verstoep, and Rutger Hofman. Network performance-aware collective communication for clustered wide area systems. *Parallel Computing*, 27(11):1431–1456, 2001.

[17] Thilo Kielmann, Henri Bal, and Kees Verstoep. Fast measurement of LogP parameters for message passing platforms. In *Proceedings of the 4th Workshop on Runtime Systems for Parallel Programming*, LNCS Vol. 1800, pages 1176–1183, 2000.

[18] J.-C. König, P. S. Rao, and D. Trystram. Analysis of gossiping algorithms with restricted buffers. *Parallel Algorithms and Applications*, 13(2):117–133, feb 1998.

[19] Jess Labarta, Sergi Girona, Vincent Pillet, Toni Cortes, and Luis Gregoris. DiP: A parallel program development environment. In *Proceedings of the 2nd Euro-Par Conference*, volume 2, pages 665–674, 1996.

[20] Edson T. Midorikawa, Helio M. Oliveira, and Jean M. Laine. PEMPIs: A new metodology for modeling and prediction of MPI programs performance. In *Proceedings of the SBAC-PAD 2004*, pages 254–261. IEEE Computer Society/Brazilian Computer Society, 2004.

[21] Csaba A. Moritz and Matthew I. Frank. LoGPC: Modeling network contention in message-passing programs. *IEEE Transactions on Parallel and Distributed Systems*, 12(4):404–415, 2001.

[22] Jelena Pjesivac-Grbovic, Thara Angskun, George Bosilca, Graham E. Fagg, Edgar Gabriel, and Jack 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] Luiz Angelo Steffenel. Modeling network contention effects on alltoall operations. In *Proceedings of the IEEE Conference on Cluster Computing (CLUSTER 2006)*, Barcelona, Spain, Sep 2006. IEEE Computer Society.

Luiz Angelo Steffenel is an Assistant-Professor at the Université de Reims Champagne-Ardenne, France. He obtained a Ph.D. in Computer Science in 2005 at the Institut National Polytechnique de Grenoble, France, and a MSc. in Communication Systems in 2002 from the École Polytechnique Fédérale de Lausanne, Switzerland. His research interests include parallel and distributed systems, grid computing, scheduling algorithms, fault tolerance and pervasive computing. Dr. Steffenel actively contributes with french Grid'5000 and european CoreGRID projects.

Maxime Martinasso s a High Performance Computing (HPC) consultant working for the oil and gaz company Shell International, The Netherlands. He obtained a Ph.D. in Computer Science at University Joseph Fourier of Grenoble in 2007, in cooperation with Bull SAS and INRIA, and a MSc. in Computation Fundaments at University Bordeaux 1. His main research activities are related to general performance aspects of scientific applications over cluster and grid. Therefore, his research interests are focused on understanding behaviors of resource utilisations, which includes behavior analysis of network communications, modelling, simulation and tracing.

Denis Trystram is Professor at INP Grenoble since 1991. He obtained a first PhD in Applied Mathematics at INPG in 1984 and a second one in Computer Science in 1988 from the same university. He is currently Regional Editor for Europe for the Parallel Computing Journal, belongs to the board of IEEE Transactions on Parallel and Distributed Systems and participates regularly to the Program Committee of the major conferences of the field. His research activities concern all aspects of the study on the design of efficient parallel algorithms. Today, his major interest is approximation algorithms for scheduling, multi-objective analysis and Game theoretic approaches with a special emphasis on grid computing.