Model of concurrent MPI communications over SMP clusters

Maxime Martinasso, Jean-François Méhaut

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Model of concurrent MPI communications over SMP clusters

Maxime Martinasso — Jean-François Méhaut

N° 5910

Mai 2006

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--5910--FR+ENG

INRIA
RHÔNE-ALPES

Model of concurrent MPI communications over SMP clusters

Maxime Martinasso* † , Jean-François Méhaut‡

Thème NUM — Systèmes numériques
Projet MESCAL

Rapport de recherche n° 5910 — Mai 2006 — 16 pages

Abstract: SMP clusters are one of the most common HPC platform used by scientific applications. The nodes of SMP cluster contain several computing elements. Scientific applications may be executed over a large number of such nodes introducing complex communication behaviors. Using for instance MPI, communications on a same node with a common interval time create concurrent accesses to resources of nodes. On SMP nodes, concurrent access implies resource sharing depending on the underlying network architecture and the MPI implementation used. This paper presents a model to predict communication times of simultaneous MPI communications over SMP clusters. This model considers the concurrency over resources of nodes and network predicting accurately communication time for many communications in conflict.

Key-words: Performance evaluation, concurrent communications, network conflicts, MPI, SMP clusters.

* INRIA, Lab. ID-IMAG (CNRS, INPG, INRIA, UJF), 51 Av. J. Kunztmann, Montbonnot, France (Email: Maxime.Martinasso@imag.fr).
† Bull-SA,1 Rue de Provence, BP 208 38432 Echirolles Cedex, France.
‡ INRIA, Lab. ID-IMAG (CNRS, INPG, INRIA, UJF), 51 Av. J. Kunztmann, Montbonnot, France (Email: Jean-Francois.Mehaut@imag.fr).

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Modèle de communications concurrentes sur des grappes SMP

Résumé : Les grappes de machines SMPs sont aujourd'hui les infrastructures de calcul haute performance les plus répandues et utilisées par des applications scientifiques. Une grappe SMP est constituée de nœuds comportant plusieurs unités de calcul. Des applications scientifiques s'exécutant sur des dizaines de nœuds SMP introduisent des comportements communicatifs complexes au sein du réseau de la grappe. Par exemple en utilisant MPI, les communications entrantes ou sortantes d'un même nœud induisent des accès concurrents sur les ressources du nœud et du réseau. Ces accès concurrents sur des nœuds SMP impliquent un partage de ressources dépendant du réseau sous-jacent ainsi que de l'implantation MPI utilisée. Ce document présente un modèle de prédiction de performances de communications concurrentes sur des grappes SMP. Ce modèle permet de prédire les temps de communications concurrentes où de nombreux conflits se produisent.

Mots-clés : Evaluation de performances, communications concurrentes, conflits réseau, MPI, grappe SMP.

1 Introduction

With the increasing SMP computation power, cluster communications continue to be an important factor that limit the performance. The cluster's nodes may contain more computing units sharing the access to the network. By adding processors to cluster nodes, execution of parallel applications introduces new events inducing complex behaviors which are difficult to analyze. As an example, one can consider the effect of mapping policies of application processes on cluster's nodes and so the effect of grouping processes on one node. Different mapping policies increase or decrease the distance between processes of distributed applications. Within SMP nodes, processes on a same node (such as one per processor) concurrently access memory, bus or devices like interconnect creating sharing resource phenomena. In a same way communications having the same source or destination must share network links and switches. On performance point of view, resource sharing brings more complexity and then more difficulty to predict application behaviors. Resource sharing prediction can also be viewed as finding the fairness principle regulating resource accesses. For network resource, such principle strongly depends of the underlying network flow control, which regulates accesses to network components. In this article we focus on communication time prediction in the context of shared resources. Our main goal is to give regards to concurrency inside SMP cluster nodes and over network resources. In this paper, the analysis is focused on message passing over two high performance network architectures: Myrinet and Gigabit Ethernet. Beside, this analysis uses common MPI communications with most used MPI methods *MPI_Send* and *MPI_Recv*. A preliminary work was achieved in [1].

After introducing technical aspect of the different network architectures, this paper presents common communication models. Then in the next sections the notion of concurrent communications is introduced followed by our model of prediction. The last section validates the accuracy of the models with a large number of concurrent communications and its usefulness for predicting communiction time of collective operation.

2 Network communications

2.1 Network architectures

Our study of communication latency is based on two high-speed network architectures: Myrinet[2] and Gigabit ethernet (GigaEth)[3].

Myrinet Myrinet is a high speed SAN network developed by Myricom. Myricom provides switches and dual-port interconnects connected with full-duplex links. Network link can deliver 2 Gbits/s bandwidth in each direction with wormhole routing, with flow and error control. Switches are based on 16-port and 32-port crossbar switches with a per-hope delay about 0.5 μs. Network cards access memory through DMA engines and contain a programmable processor. Myrinet network latency is about 5 μs.

Many protocols are built to use Myrinet hardware:

- MX[4] (Myrinet Express) is the new Myricom protocol which decouples the progression of the protocol from the execution of the host application, improving low-level point to point performance and the overlapping of communication and computation.
- GM[5] is the standard Myricom user-level communication protocol providing a reliable ordered delivery of packets.
- BIP[6] and FM[7] from academic teams are other examples of protocols over Myrinet.

Gigabit Ethernet: Ethernet is a frame-based computer networking technology for LANs, defining frame formats, wiring and signaling for the physical layer. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD) scheme to share the communication channel between

interconnects. Network switch has typically a star topology. 1000BASE-T Ethernet or Gigabit Ethernet (GigaEth) has full-duplex links and uses all four pairs of RJ45 cable for simultaneous transmission in both directions. Network latency of Gigabit Ethernet strongly depends of the protocol used and the interconnect manufacturer, but always in order of few tens μs. Network bandwidth is 1 Gbits/s. Common protocols like TCP or UDP may be used over ethernet.

2.2 Flow control

Flow control is one of the most important factors for regulating access (and so concurrency) to network resources. Effectively when a sharing conflict occurs over the network path, it is handled by the flow control of the physical network. Thus different policies of flow control lead to different behaviors of communications and then communication times.

Myrinet flow control Myrinet flow control is based on cut-through routing of packets. Such technology blocks packets while the communication channel is unavailable, avoids the need of packet buffering. Myrinet switches do not introduce concurrency on network path through its pipelined cross-bar topology. Two chips of the NIC performs the flow control. To achieve cut-through routing, Myrinet NIC used a *Stop & Go* flow control protocol. In case of concurrency, receiver injects *Stop* or *Go* control symbols into the network to inform the senders to stop or resume the communication flow.

GigaEth flow control The flow control mechanism of GigaEth was defined by the IEEE 802.3x committee for full-duplex Ethernet. Within this standard, when a receiver becomes congested, it can send a *pause frame* to the source which therefore stop sending packets for a specific period of time. The receiver can also send a frame to inform the source to begin sending data again. However, using TCP over GigaEth, the reliability of packets will be increased thanks to the concept of window size and the TCP's sliding windows mechanism. But such flow control has an important impact on the communication time.

2.3 Communications with MPI

MPI stands for Message-Passing Interface[8], and defines an interface for message-passing communication mechanism. MPI may be based on different low-level protocols to achieve communication, such as TCP or optimized protocols for specific network architectures. However MPI standard introduces a rendez-vous rule to handle large messages. First to send a large message, a rendez-vous message is sent to the receiver to prepare (buffer allocation) the reception of the large message. MPI does not specify a size value to distinct large and small messages, which depends of the MPI implementation. Common MPI implementations are:

- Mpich is a MPI implementation from Argonne National Laboratory[9]. Mpich is a very portable MPI implementation thanks to its design of an abstract device interface (ADI) allowing several implementations of specific protocols. Therefore Mpich-MX is a specific and efficient implementation of MPI for Myrinet and MX protocol.
- Lam/MPI[10] is a MPI implementation from the Trustees of Indiana University. Lam/MPI launches a daemon on each host. This daemon handles and schedules communications.

2.4 SMP cluster platform test

The different tests in this paper are achieved over a cluster of dual processor. The Myrinet/GigaEth cluster is composed of 105 IBM eServer 325 2-processor nodes. Processors are 2 GHz AMD Opteron (64 Bits), Mpich version is 1.2.6 with MX protocol on 1.0.0, Lam-MPI is on 7.1.1. with a linux 2.4.21 kernel. The Gigabit Ethernet network cards are Broadcom Corporation NetXtreme BCM5704 cards.

3 Network communication models

3.1 LogP, LogGP and pLoGP models

Well-known communication models are LogP model [11], LogGP model [12] and pLoGP model [13]. These models predict communication times without contention effect or resource sharing. They predict communication times following linear equation parameterized by several measured parameters. These parameters are mainly: o overheads, G gap per byte (its reciprocal is the available bandwidth per processor) and L the communication latency. The linear form can be written as:

$$t(k) = 2 * o + L + (k-1) * G$$

where $t(k)$ represents the time for sending k bytes over the network. The communication time is split into two parts: an overhead time $(2 * o + L)$ and a communication time $((k-1) * G)$. LogP and LogGP are differentiated by the parameter measurements. LogP model (respectively LogGP model) parameters are measured for short messages (respectively large messages).

pLoGP model expresses overheads and gap in function of the message size: $g(m)$ and $o(m)$. The authors explain a method to obtain these measurements with an available benchmark on their website[1]. In their article, they also express the pLoGP parameters in terms of LogGP's parameters.

3.2 Linear form and MPI communications

This subsection presents the fitness of linear model (derived from pLoGP model) against measured values over different network architectures and MPI implementations. With a free-contention context, prediction of communication times may be achieved following the pLoGP model by a linear form $l(m; g, o) = g(m) + o(m)$, where m describes the message size, $g(m)$ is the variation of the bandwidth reciprocal in function of the message size and $o(m)$ the overhead function depending on m. We simplify the model letting $o(m) = L$ be a constant value , with $L = \lim_{m \to 0} l(m; g, o)$, which represents a fixed service time (Latency). As mentioned before, MPI standard introduces a rendez-vous rule to handle large messages. Therefore the incremental service time $g(m)$ for MPI communications follows a 2-linear equation such as Equ. 1, where m_0 is the message size for which MPI changed to the rendez-vous protocol, with $a = g(m_0)$ and $b = \lim_{m \to \infty} g(m)$.

$$g(m) = \begin{cases} a * m, \ 0 \leq m < m_0 \\ b * m, \ otherwise \end{cases} \quad (1)$$

The values presented in Figure 1,2 and 3 are average values calculated over a sample size of 10000 communications[2]. These figures present the fitness of linear form to predict MPI communication times in a contention-free context. Table 1 displays the values of latency and bandwidth.

These benchmarks are based on the MPI primitive *MPI_Send* and *MPI_Recv*. However the authors of [14] present a short description with linear models for every sending functions defined by the MPI standard. From these experiments, a linear model is accurate to predict communication times without contention.

4 Concurrent communications

This section presents our methodology to analyze concurrency of MPI communications. This methodology is based on different communication patterns and experiments in which two communications share resources with different time intervals and communication sizes. The study of concurrent communications are mainly relevant for large messages. For small messages, the communication cost is too low. Large communications take more time (physical and software)

[1] http://www.cs.vu.nl/albatross/

[2] standard deviation is always lower than 10%.

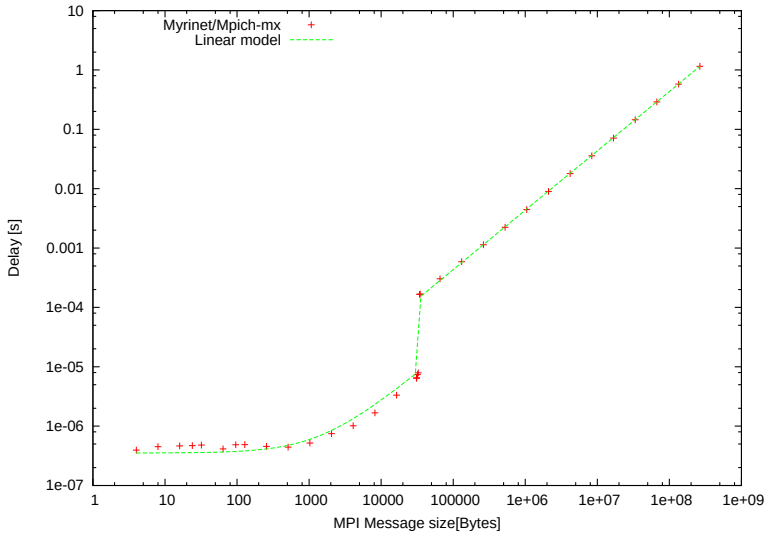

Figure 1: Linear model against measured values for Myrinet/Mpich-mx

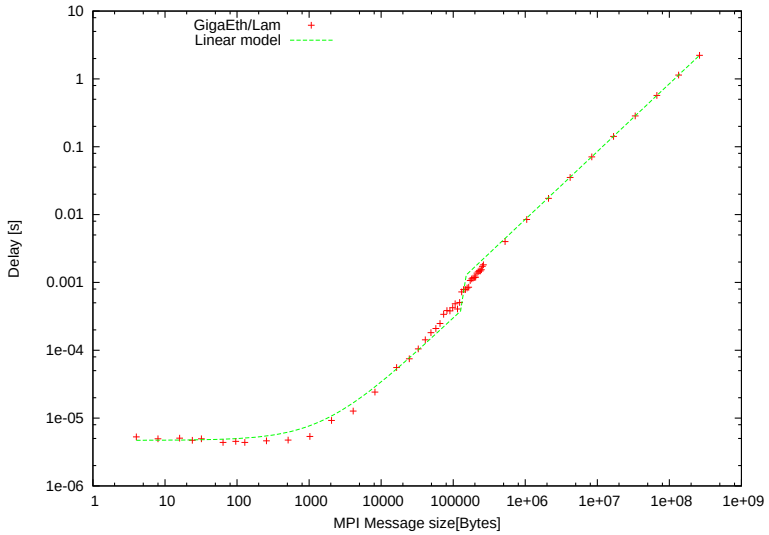

Figure 2: Linear model against measured values for GigaEth/Lam-MPI

Table 1: MPI latency and bandwidth

Network	L [μs]	Bw [MB/s]
Myrinet-Mpich	3.5	219.4
GigaEth-Lam	4.7	112.2
GigaEth-Mpich	5.8	112.1

inside the transfer process and then are more sensitive to concurrency, which have a direct impact

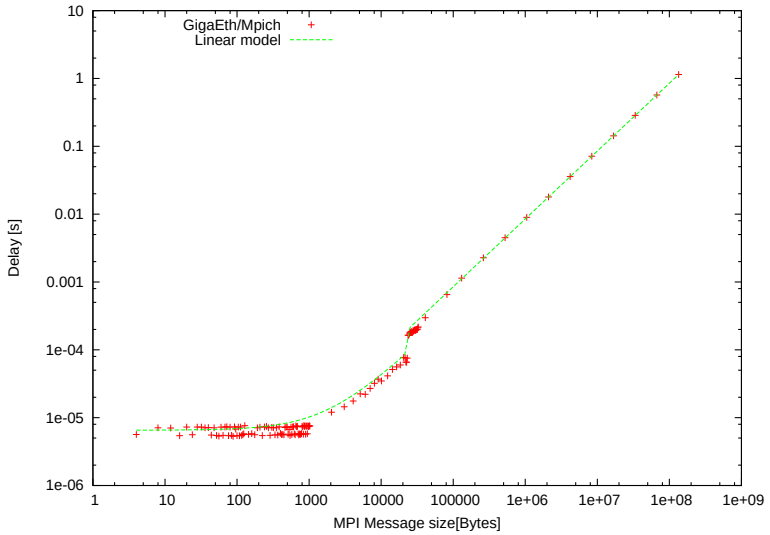

Figure 3: Linear model against measured values for GigaEth/Mpich

on their communication time. Another point of view is that resource sharing can be observed for large communications (no serialization).

4.1 Multiple communications

MPI applications can generate multiple communications in different ways. The following paragraphs introduce our approach to capture most of the different cases of concurrent communications.

We study concurrent communications thanks to synthetic MPI programs, which create communication sets. We associate to synthetic programs different mapping policies.

- **Synthetic MPI programs**: the goal of the synthetic programs is to create communications and to measure their completion times. Communication spreads data with the primitives *MPI_Send* and *MPI_Recv*. Each MPI process is only in sending or receiving state. Times are measured on the send part. Time measure is calcualted dividing the number of processor clock ticks spent during the call of the send primitive by the frequency of the processor.
- **Mapping policies**: different mapping policies of synthetic programs lead to different concurrent behaviors. Therefore, as our tests are only achieved over dual processor nodes, we choose to identify communication patterns by a mesh of arrows represeting communications between nodes. For example, consider the mapping of a 2-communication synthetic program over three dual nodes in which process 0 on node 0 sends message to process 1 on node 1 and process 2 on node 2 sends messages to process 3 on node 1. This mapping will be identified by $0 \rightarrow 1 \leftarrow 2$.

Finally, figure 4 displays examples of some cases of concurrent communications.

5 Concurrent communication models

In this section, we propose new communication models coming from a wide campaign of measures for synthetic MPI programs. As mentioned above, the goal of synthetic MPI programs is to reproduce communication patterns. In a first step, we focused our experiments with synchronous sends and identical large message size for all communications. MPI processes of synthetic programs are mapped to one process per processor.

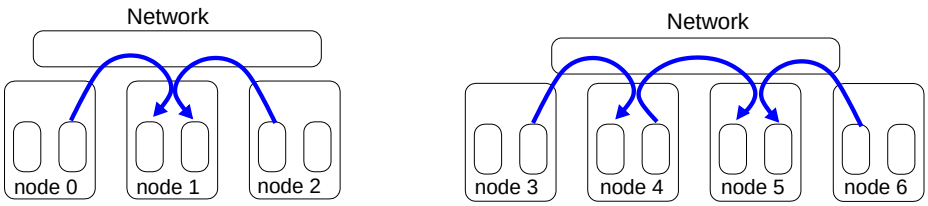

Figure 4: Examples of concurrent communications

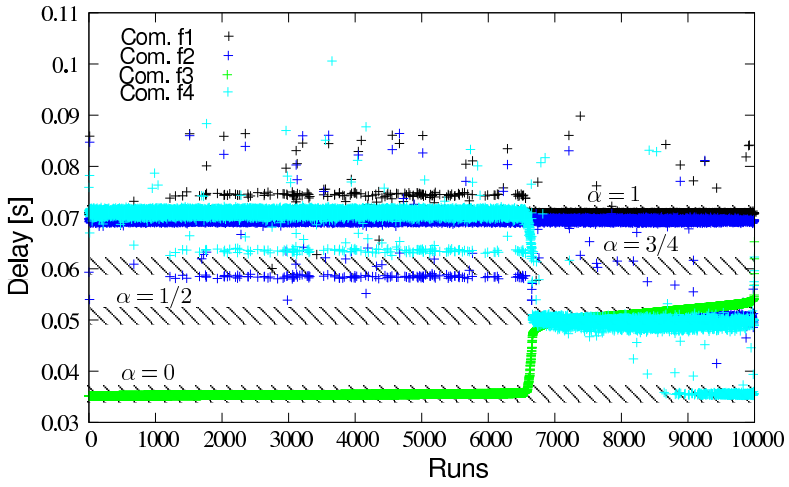

Figure 5: Distribution of communication times, pattern $0 \leftarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, GigaEth / Lam MPI, mesage size 4 MB, values ordered by $f3$.

Our interest is to find a correlation between communication patterns and observed values. Figure 5 is an example of observerd values for pattern $0 \leftarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. For each independant run of the synthetic program we measure a set of four values, one per communication. Communication between node $i-1$ and i is referred by fi. The sets of observed times of each run are ordered by the value of communication $f3$. Indeed displaying ordered sets of values allows to make easier the search of correlation between times and pattern. Effectively, this figure showes that communications $f1$ and $f2$ have mainly a constant communication time and that communication $f3$ and $f4$ vary together.

This experiment and all the other experiments of the campaign describe interesting properties in which communication times vary only over dense groups of values. Moreover the mean value of such groups is a rational multiple $(1+\alpha)$ of the communication time without conflict t_{NC}. In figure 5, stripped areas represent values for which communication times are inside the interval $[(1+\alpha-0.1)*t_{NC}, (1+\alpha+0.1)*t_{NC}]$. In this example, both $f1$ and $f2$ vary over area $\alpha = 1$, $f3$ varies over $\alpha = \{0, \frac{1}{2}\}$ and $f4$ varies over $\alpha = \{0, \frac{1}{2}, 1\}$.

Another interesting phenomena is that dense groups and the mean values associated to are similar for different patterns. We conclude that communication times alternate only above different areas for all patterns, and find a correlation between patterns and α values. With this observation, we base our model by associating such set of α values not directly to complex patterns but to simple patterns of 1-or-2 communications (also called conflict). Then we suggest a method to decompose such complex patterns into simple conflicts.

5.1 Notion of conflict and flow cuts

From the previous analysis, we introduce the notions of conflict and flow cuts.

A conflict represents communications which share resources throught simple pattern. As communications can income or outgo from cluster nodes, one communication can be seized by one of the following elementary conflicts:

- Conflict *Outgo* $C_{\leftarrow X \rightarrow}$ where the communication only outgoes with other outgoing communications from a node X.
- Conflict *Income* $C_{\rightarrow X \leftarrow}$ where the communication incomes with only other incoming communications to a node X.
- Conflict *Outgo/Income* $C_{\rightarrow X \rightarrow}$ or $C_{\leftarrow X \leftarrow}$ in which a communication outgo (resp. income) with other incoming (resp. outgoing) communications.

We consider MPI communications as a flow of packets. A flow f is defined by a data quantity Q_f to send and a network specific communication rate x. A flow cut is associated to a communication flow describing the delay generated by the resource conflict. Our method is to assume for each elementary conflict sets of flow cuts. Each elementary conflict can be written as a set of 2 flow cuts: $C = (\alpha_1, \alpha_2)$. Then we extend linear model adding the impact of communication conflicts. We approximate communication time of flow f_i within a conflict C by:

$$T_{f_i|C} = (1 + \alpha_i) * Q_{f_i} * x, \alpha_i \in C$$

5.2 Synchronous and homogeneous model

Synchronous and homogeneous communications start at the same time with equal message sizes. We remind that our test bed is composed by dual processor therefore with a mapping of one process per processor a communication can be part of maximum two conflicts. This model is based on rules that decompose a pattern into several conflicts. We only decomposed a pattern into elementary conflicts.

Rules:

(R1) A pattern can be divided into 2-communication conflicts (or 1-communication conflicts).

(R2) Pattern or sub-pattern only composed by $C_{\leftarrow X \rightarrow}$ or $C_{\rightarrow X \leftarrow}$ cannot be divided into elementary conflicts.

(R3) A single communication can be part of two different conflicts. In that case the flow cut applied to the communication time is the flow cut of the most dominant conflict. Conflicts are ordered by dominance like:

$$C_{\rightarrow X \leftarrow} \textit{ or } C_{\leftarrow X \rightarrow} > C_{\leftarrow X \leftarrow} \textit{ or } C_{\rightarrow X \rightarrow}$$

This order can be explained by the fact that communications having the same direction share the half part of the full-duplex link in an exclusive manner.

Examples: Consider a simple pattern like $0 \rightarrow 1 \rightarrow 2 \rightarrow 3$. From **(R1)** such conflict is split into: $0 \rightarrow 1 \rightarrow 2$ & $2 \rightarrow 3$. Note that $0 \leftarrow 1 \leftarrow 2 \leftarrow 3$ will be rewritten as $0 \leftarrow 1$ & $1 \leftarrow 2 \leftarrow 3$. Now, as another example, a pattern like $0 \rightarrow 1 \rightarrow 2 \leftarrow 3 \leftarrow 4 \leftarrow 5$ is transformed into $0 \rightarrow 1$ & $1 \rightarrow 2 \leftarrow 3$ & $3 \leftarrow 4 \leftarrow 5$.

5.3 Asynchronous and heterogeneous model

This subsection enhances the previous model with different start times and message sizes of a communication set. The idea remains simple and consists to separate asynchronous and heterogeneous communication conflicts into synchronous and homogeneous conflicts. It is possible to design an algorithm which computes the quantity of data sent by each communications between consecutive start times. Figure 6 is an example of such executive steps of this algorithm. For instance, at the first step between times A and B, communication (1) spreads data without conflict. When communication (2) starts at time B, both communications are in conflict $C_{\rightarrow X\leftarrow}$. Then during the time interval B-C, we consider that communication (1) and (2) are in a homogeneous and synchronous conflict. Therefore with our previous model, we can determine the quantity of data sent for both communications. When there is no more start times, or when a communication ends before a new communication starts, each communication has an remaining data quantity to send within a number of conflicts decreasing. It is the case at the time D, when communication (4) starts. In this time point, we can easily calculate the first communication to end, here communication (3), and then obtain the time point E. Therefore for communication (1),(2) and (4) we are able to calculate the quantity of data sent for each communication during time interval D-E. Such algorithm can be implemented in dynamic way, with a discrete event simulation, where events are a communication start or end. However, we implement statically this algorithm with a recursive program in which recursion is applied on the different homogeneous and synchronous part. As example, consider time point E, we can view the recursion call to the set of communications (1) and (2) and the communication alone (4).

A general aspect of this algorithm is that it does not depend of the synchronous and homogeneous model. One can define another prediction model of synchronous and heterogeneous communication times and enhance these predictions to asynchronous and heterogeneous communications thanks to this algorithm.

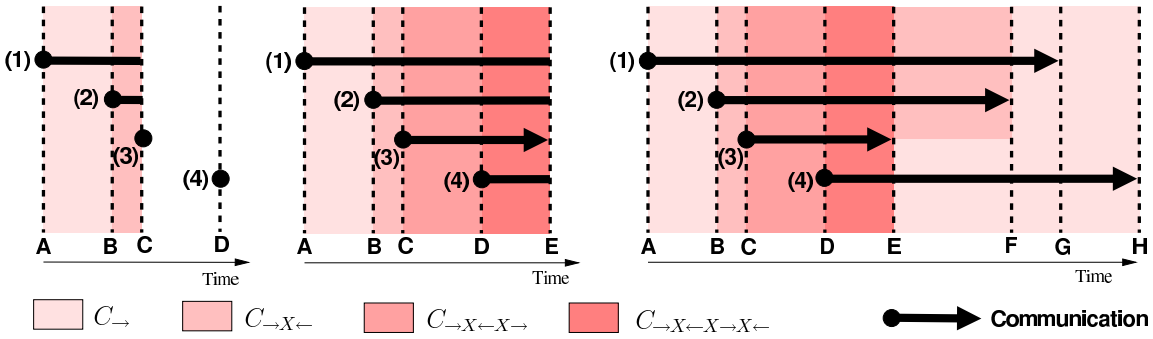

Figure 6: Example of asynchronous and heterogeneous communications divided into synchronous and homogeneous parts.

6 Validation

This section presents the validation of our model, as described in [15]. In the first subsection we express the measured flow cuts, and then we compare measured values versus predicted values. The last subsection shows the usefulness of the model for the time prediction of a collective communication.

6.1 Flow cuts evaluation

To evaluate the value of the flow cuts, we use the previous algorithm with simple communication patterns i.e. elementary conflicts. As an example, consider pattern $0 \rightarrow 1 \rightarrow 2$ which corresponds

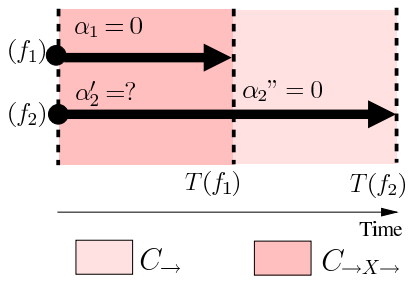

Figure 7: Flow cuts evaluation, example conflict $C_{\rightarrow X \rightarrow}$.

Table 2: Measured flow cuts for conflicts and network architectures.

Network / MPI	Odd # of comm $C_{\leftarrow X \rightarrow}$,$C_{\rightarrow X \leftarrow}$	Even # of comm $C_{\leftarrow X \rightarrow}$,$C_{\rightarrow X \leftarrow}$	$C_{\rightarrow X \rightarrow}$	$C_{\rightarrow}$ Include	$C_{\rightarrow}$ Start/End
GigaEth / Lam-MPI	(1,...,1)	($\frac{1}{2}$,2,$\frac{1}{2}$,2,...,$\frac{1}{2}$)	(0,3)	$\frac{1}{2}$	0
GigaEth / Mpich	(1,...,1)	($\frac{3}{4}$,$\frac{4}{3}$,$\frac{3}{4}$,$\frac{4}{3}$,...,$\frac{3}{4}$)	(0,3)	$\frac{1}{2}$	0
Myrinet / Mpich-MX	(1,...,1)	(1,...,1)	(0,0)	0	0

to conflict $C_{\rightarrow X \rightarrow}$ with communications f_1 (incoming) and f_2 (outgoing), Figure 7. We use homogeneous and synchronous communication, i.e. f_1 and f_2 have the same start time and message size. From experimentation, for example with Lam/MPI and GigaEth, we obtain than $T(f_1|C_{\rightarrow X \rightarrow}) = T(f_1|NC)$ and $T(f_2|C_{\rightarrow X \rightarrow}) = (1+1/2)*T(f_2|NC)$, with $T(f_1|NC) = T(f_2|NC)$ the time without conflict. From this point, we know that both communications spread their data at different rates. Moreover f_1 does not suffer any delay from the conflict. Therefore we can deduce that communication time of f_2 is split into two parts, one suffering delay caused by f_1, and another part without delay (when f_1 ends). Our interest is to calculate the rate α_2' of f_2 suffering conflict because this value represents the flow cut for f_2 of the conflict $C_{\rightarrow X \rightarrow}$, the flow cut of f_1 is obviously 0. This leads us to resolve a simple linear equation, which gives the value 3. We conclude that the flow cut of $C_{\rightarrow X \rightarrow}$ for Lam/MPI and GigaEth is $(0, 3)$ and in an equivalent way the flow cut of $C_{\leftarrow X \leftarrow}$ is $(3, 0)$. Proceeding in the same way, we obtain the flow cuts of each elementary conflict in Table 2.

The flow cut values reveals that GigaEth has a close behavior for both MPI implementation. However Mpich seems to induce better flow cuts for one conflict. The flow cuts of conflict $C_{\rightarrow X \rightarrow}$ show that the outgoing communication is penalized against the incoming communication. This phenomena may be explained by the fact that the node, i.e operating system and interconnect card, has a better control on the outgoing communication and then stop its spreading. Myrinet flow cuts have only value of 0 or 1. Values of 1 in case of communications sharing an half part of the full-duplex link may be explained if we consider a fair application of the flow control protocol, .i.e each communication is stopped and resumed in an equivalent manner. Values of 0 for communications in conflict $C_{\rightarrow X \rightarrow}$ may be interpreted by the hardware design of the Myrinet interconnect. Effectively Myrinet interconnect cards are composed of dual port with separated send and receive buffers each one associated to a specific control chips.

6.2 Model accuracy and scalability

In this subsection we evaluate the accuracy and scalability of our model using a large pattern. For both cases, we compare observed values over 10000 runs for several combinations of MPI implementations and network architectures.

Model accuracy is measured against a pattern with 30 communications. The pattern is chosen randomly with an equal probability to have outgoing or incoming communication. A communication flow f_i has a start time of i milliseconds and message size following the cycle: $m_1 = 1, m_2 = 2, m_3 = 4, m_4 = 2, m_5 = 1, m_9 = 1, m_{10} = 2, m_{11} = 4, m_{12} = 2, m_{13} = 1, \cdots$ in MBytes. We verified that the randomly generated pattern is composed by all most all types of conflicts, moreover the maximum number of communications being in conflict with a common interval time is equal to 13 for Myrinet, 20 for GigaEth with Mpich, and 21 for GigaEth with Lam/MPI. Figures 8 and 9 display the interval times of measured values against predicted values. Two sorted consecutive data points are in the same interval if they do not differ of 0.2 % of the lower one, and only interval times having a number of points greater than 10% of the sample size are plotted. Displayed interval times are characterized by their maximum, minimum and median values.

We compare a set of predicted values against each set of interval median values. Predicted values are calculated by applying the algorithm presented in subsection 5.3. Table 3 resumes different absolute errors. The errors presented are:

- maximum and minimum average absolute error of communication sets, i.e. the sum of absolute error of each communication divided by the number of communications.
- maximum and minimum global absolute error of communication sets, i.e. the absolute error of the sum of communication times.
- the absolute error of the worst and best predicted time over all sets of communications, and the communication numbers associated to.

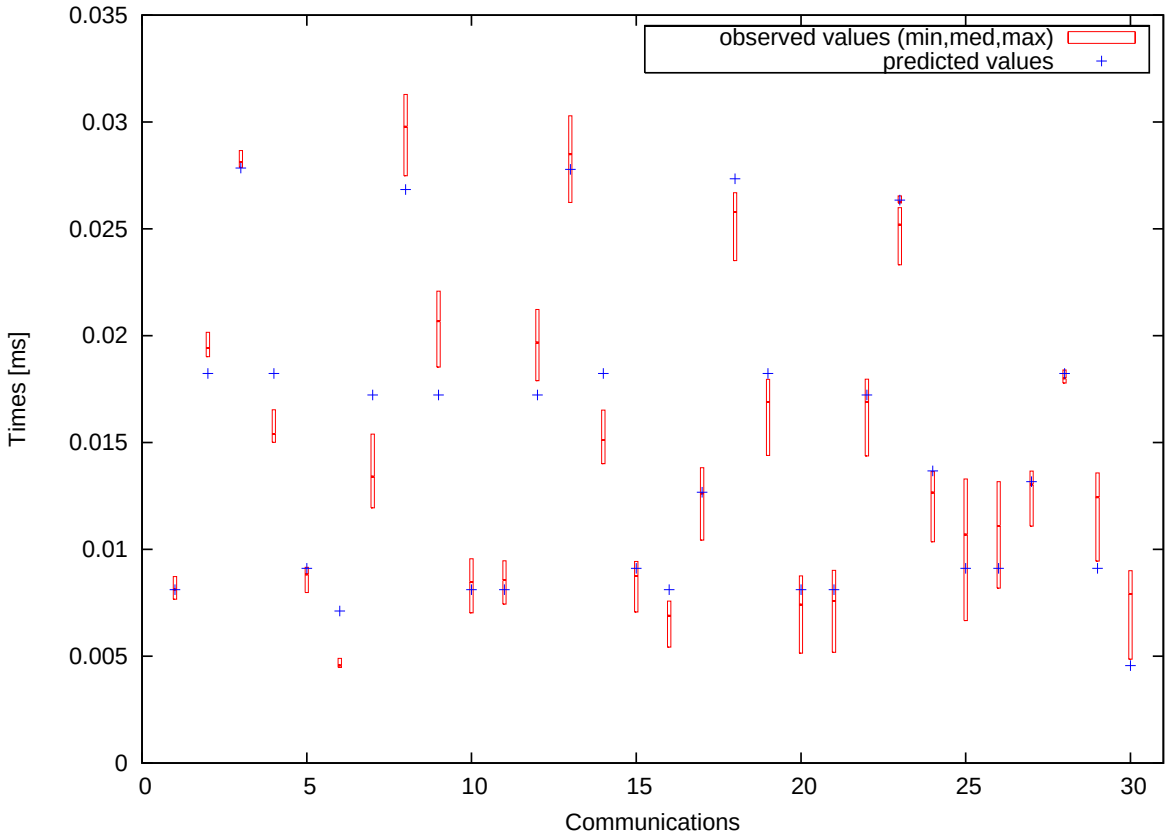

Figure 8: Model accuracy for pattern of 30 communications on Myrinet

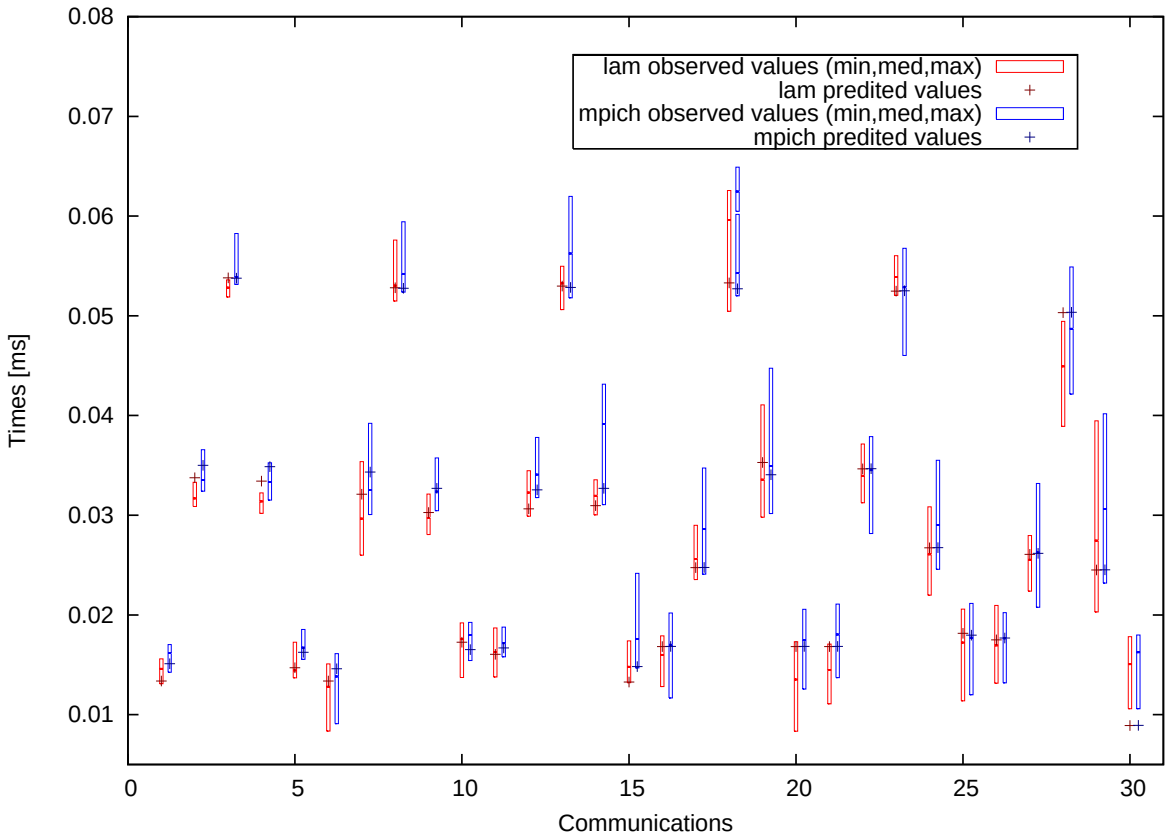

Figure 9: Model accuracy for pattern of 30 communications on GigaEth

Table 3: Absolute error for pattern of 30 communications

Network / MPI	Abs. err. ave. [%] min/max	Abs. global err.[%] min/max	Abs. err. / Coms [%] min/max (# com)
GigaEth / Lam-MPI	6.7/6.7	0.1/0.1	(8) 0.3/40.3 (30)
GigaEth / Mpich	6.6/7.1	4.1/5.0	(3) 0.0/45.6 (30)
Myrinet / Mpich-MX	11.6/11.8	0.7/0.9	(1) 0.1/56.5 (6)

Communications on GigaEth network are predicted with an average absolute error of 7%, and on Myrinet network with 12%. From this experience the presented model predicts with a good accuracy a communication set of 30 concurrent communications.

6.3 Application: broadcast communication

Broadcasting data is a collective operation where one MPI task has data to spread to the entire set of MPI tasks. One of the most efficient way to achieve a broadcast is to use a binomial spanning tree. Using such broadcast operation, our purpose is to accurately predict the completion time of this collective operation depending on the process mapping.

We consider a simple cluster architecture with four dual-processor nodes. Our test program consists of a broadcast based on binomial tree following by a barrier (*MPI_Barrier*) to ensure the completion on each node of the broadcast. All the communications needed by the broadcast are

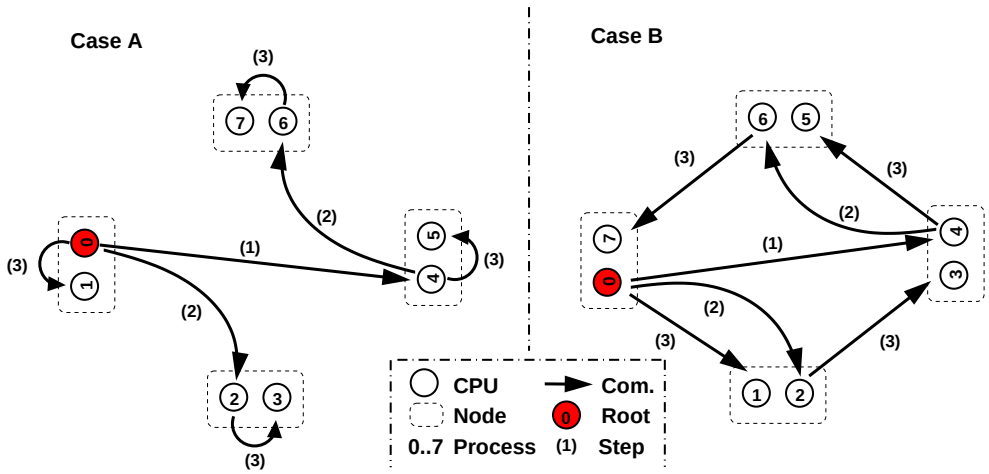

Figure 10: Example of binomial broadcast with different mapping of processes.

homogeneous and denoted as f_b. Figure 10 displays the three steps followed by the broadcast operation for two process mappings. In case A, step 1 involves only one communication, then at step 2 there are two parallel communications without conflict, and third step involves only parallel intra node communications. Thus one can calculate the time of the broadcast for mapping A as $T_A = 2 * T(f_b|NC) + T_{intra}(f_b)$. Proceeding in the same manner, for mapping B, we can calculate $T_B = 2 * T(f_b|NC) + T(f_b|C)$ with $T(f_b|C)$ the maximum communication time in the conflict generated at step three. We also calculate $T_{B_{indep}} = 3 * T(f_b|NC)$ the completion time without considering the impact of the last conflict.

Table 4 presents observed and predicted broadcast times. The time to achieve a *MPI_Barrier* is insignificant against the broadcast time. Such experiments validate our model and introduce the influence of process mapping on performance aspect.

Table 4: Broadcast of 10 Mbytes, time prediction with mapping A and B

Network / MPI	Observed time A [s]	T_A [s]	Observed time B [s]	T_B [s]	$T_{B_{indep}}$ [s]
GigaEth / Lam-MPI	0.205	0.197	0.335	0.334	0.267
GigaEth / Mpich	0.235	0.200	0.377	0.334	0.267
Myrinet / Mpich-mx	0.103	0.104	0.134	0.136	0.136

In a similar way, this model is also useful to determine the best mapping policy for real MPI application or benchmark. The work achieved by the authors of [16] may be augment by the used of our model. In their paper, they reveal the impact on performance of mapping processes of the HPL benchmark[18] on dual SMP nodes. Using Vampir[17], they trace the communication patterns of the HPL benchmark. Then they compare the overall completion time of the benchmark in function of the processes mapping. Their mapping policies depend on the frequency of communication between processes or the communication message sizes. They claim that grouping processes with large communication size enhances the overall performance of the benchmark. Using our model we may find a mapping policy reducing the number of costly conflicts and accurately predict the communication times of large communication size patterns. This model may be a powerful help to easily augment the performance of applications and benchmarks by decreasing the communication times thanks to a better mapping strategy.

7 Conclusion

This paper presents a case study of concurrent MPI communications for GigaEth and Myrinet architectures over SMP (dual processors) clusters. This study introduces a notion of resource sharing by patterns and conflicts and their specific behaviors. The aim of this study is to introduce a communication model, which accurately predicts communication time for combinations of sharing conflicts. This model reduces a chain of conflicts into several elementary conflicts thanks to a set of rules. Communication time is then predicted by the flow cut of each elementary conflict. The model helps to predict times of asynchronous and heterogeneous communications with accuracy and scalability, and can be implemented with discrete event simulations or a static algorithm. Then with traces of MPI applications, one can find mappings of MPI processes reducing the number of most penalizing conflicts and so reducing the overall time spent on communication. Another interesting usefulness of this model is to predict collective communication behaviors and so to deduce new algorithms taking into account network conflicts. As future work we plan to make experiments over SMP (or NUMA) nodes with a greater number of processors and other network architectures like Quadrics, to refine the concurrent communication model.

References

[1] M. Martinasso and J-F. Méhaut. Prediction of communication latency over complex network behaviors on SMP clusters. In *Proceedings of the 2nd European Performance Engineering Workshop (EPEW) 2005, vol. 3670 of LNCS*, 172-186, 2005. Springer-Verlag.

[2] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic and W-K Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29-36, 1995.

[3] R. Seifert. *Gigabit Ethernet Technology And Applications For High Speed LANs*. 1998.

[4] Myricom. Myrinet express (mx): A high-performance, low-level, message-passing interface for myrinet, 2005.

[5] Myricom. Gm reference manual, 2002.

[6] B. Tourancheau. High Speed Networks for Clusters, the BIP-Myrinet Experience. In *Proceedings of the 7th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, 2000.

[7] S. Pakin, M. Lauria and A. Chien. High performance messaging on workstations: Illinois Fast Messages (FM) for Myrinet. Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing, 55, 1995.

[8] Message Passing Interface Forum. MPI: A message-passing interface standard. *International Journal of Supercomputer Applications*, 165-414, 1994.

[9] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789-828, 1996.

[10] G. Burns, R. Daoud and J. Vaigl. LAM: An Open Cluster Environment for MPI. *Proceedings of Supercomputing Symposium*, 379-386, 1994.

[11] D. Culler, R. Karp, D. Patterson, A. Sahay, E. Santos, K. Schauser, R. Subramonian and T. von Eicken. LogP: a practical model of parallel computation. *Commun. ACM*, 39(11):78–85, 1996.

[12] 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. *7th Annual Symposium on Parallel Algorithms and Architectures*, 1995.

[13] T. Kielmann, H. E. Bal, and K. Verstoep. Fast measurement of logp parameters for message passing platforms. In *J. D. P. Rolim, editor, IEEE International Parallel & Distributed Processing Symposium (IPDPS) Workshops, vol. 1800 of LNCS*, 11761183, 2000. Springer-Verlag.

[14] E.W. Chan, M.F. Heimlich, A. Purkayastha, R.A. van de Geijn. On optimizing collective communication. *Cluster Computing, 2004 IEEE*, 145-155, 2004.

[15] M. Martinasso and J-F. Méhaut. Analysis and model of network contention over SMP clusters. *International Meeting on Grid and Parallel Computing*, Beirut Lebanon, Jan. 2006.

[16] T. Leng, R. Ali, J. Hsieh, V. Mashayekhi and R. Rooholamini. Performance impact of process mapping on small-scale SMP clusters- A case study using High Performance Linpack. In *IEEE International Parallel & Distributed Processing Symposium (IPDPS) Workshops*, 0236b, 2002.

[17] W. Nagel, A. Arnold, M. Weber, H. Hoppe and K. Solchenbach. VAMPIR: Visualization and analysis of MPI resources. *Supercomputer*, 12(1):69–80, Jan. 1996.

[18] J.J. Dongarra, J.R. Bunch, C.B. Moler and G.W. Stewart. LINPACK Users' Guide. *SIAM Publications*, Philadelphia, 1979.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

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