Gestion des réseaux multi-grappes hétérogènes avec la bibliothèque Madeleine III

Olivier Aumage

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Heterogeneous Multi-Cluster Networking with the Madeleine III Communication Library

Olivier Aumage, LIP

No 4365

Février 2002

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4365--FR+ENG

INRIA
RHÔNE-ALPES

Heterogeneous Multi-Cluster Networking with the Madeleine III Communication Library

Olivier Aumage, LIP

Thème 1 — Réseaux et systèmes
Projet ReMaP

Rapport de recherche n˚4365 — Février 2002 — 16 pages

Abstract: This paper introduces the new version of the *Madeleine* portable multi-protocol communication library. *Madeleine* version III now includes full, flexible multi-cluster support associated to a redesigned version of the transparent multi-network message forwarding mechanism. *Madeleine III* works together with a new configuration management module to handle a wide panel of network-heterogeneous multi-cluster configurations. The integration of a new topology information system allows programmers of parallel computing applications to build highly optimized distributed algorithms on top of the transparent multi-network communication system provided by *Madeleine III*'s virtual networks. The preliminary experiments we conducted regarding the new virtual network capabilities of *Madeleine III* showed interesting results with an asymptotic bandwidth of 43 MB/s over a virtual link made of a SISCI/SCI and a BIP/Myrinet physical link.

Key-words: Cluster of cluster, high-speed networks, multi-protocol communication, heterogeneity.

(Résumé : tsvp)

This text is also available as a research report of the Laboratoire de l'Informatique du Parallélisme `http://www.ens-lyon.fr/LIP`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : 04 76 61 52 00 - International: +33 4 76 61 52 00
Télécopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

Gestion des réseaux multi-grappes hétérogènes avec la bibliothèque Madeleine III

Résumé : Cet article présente la nouvelle version de la bibliothèque de communication multi-protocole portable *Madeleine*. La version III de *Madeleine* intègre désormais un support complet et flexible pour les configurations multi-grappes associé à une version redessinée du mécanisme transparent de retransmission multi-réseau. *Madeleine III* travaille de concert avec un nouveau module de gestion de session capable de prendre en charge une très grande variété de configurations multi-grappes hétérogènes au niveau réseau. L'ajout d'un nouveau système d'information sur la topologie permet en outre au programmeur d'applications de calcul parallèle de concevoir des algorithmes hautement optimisés au-dessus du système de communication multi-réseau transparent fourni par les réseaux virtuels de *Madeleine III*. Les premiers résultats des expériences menées sur les nouvelles capacités de réseau virtuel de *Madeleine III* montrent des résultats intéressants, avec une bande passante asymptotique de 43 Mo/s sur un lien virtuel constitué d'un lien physique sur SISCI/SCI et d'un lien physique sur BIP/Myrinet.

Mots-clé : Grappes de grappes, réseaux rapides, communication multi-protocoles, hétérogénéité.

1 Introduction

Cluster Computing Evolution The interest in cluster computing kept growing since the early years of the previous decade. Now, this growing interest is leading many research teams to go further and experiment with new and wider forms of computer aggregates.

At a very large scale, these new experiments mostly concern two categories of configurations. The first category is made of the metacomputing grids. These grids interconnect powerful supercomputers from computation centers disseminated among countries and continents. Metacomputing grids are built using software such as the Globus environment, which handles session management, security and resource allocation issues (among many other ones). The second category embraces the loosely coupled systems such as the one used by the SETI program. These systems use idle CPU cycles on a very large number of workstations to perform complex computations. Each workstation first works on its own piece of the whole computation and then sends the results to a centralized server.

Now, at the scale of clusters interconnected by a SAN network, the main objective is to build high performance computing systems out of several clusters of workstations. The idea is to allow computation centers to interconnect their clusters with high performance networks (such as the ones used for building the clusters themselves), and to get a single powerful multi-cluster computing architecture. This objective is rather challenging for communication libraries. Indeed, the inherent heterogeneity and especially the inherent network heterogeneity found in such architectures is very high. Nevertheless, communication libraries have to compose with that heterogeneity while ensuring the most efficient use of underlying hardware capabilities. Moreover, network links established between clusters may well outperform internal cluster links. As a consequence, communication environments that were designed for grids (MPICH-G [9], PACX-MPI [6]) are just unsuitable for supporting multi-cluster communications. They rely on the use of standard protocol stacks (e.g. TCP) over inter-cluster links which cannot deliver the full networking hardware potential.

Need for Efficient, Flexible Networking Environments It results from this observation that multi-cluster communication platforms have to be more specifically designed. Several issues have to be addressed and can be split into two parts.

The first part that comes to mind groups technical issues involving specific communication support over inter-cluster links in order to extract most of the nominal hardware capabilities up to the application. Most of these issues have to be addressed by the software located on cluster gateways (the nodes that belong to more than a single network, that is, the nodes connected to an inter-cluster link). Indeed, the multi-cluster communication support has to provide an efficient way to transfer messages across cluster boundaries. Yet, it should strive to make that transfer as transparent as possible, both for the application programmer and for the application process(es) running on the processing nodes used as gateways. Moreover, the software running on the gateways should keep messages to be forwarded from going through the whole communication stack twice on each gateway. Yet, whatever the optimization level of the multi-cluster data forwarding mechanism, its overhead cannot be made negligible. Finally, applications may have a need for an access to information about the underlying configuration topology, so as to build clever high-level communication and distribution algorithms (for instance, by analyzing whether having a given message cross the boundaries of a cluster is worth the cost of the overhead or not). Therefore, the communication library should be able to provide the application with mechanisms to perform queries about topology information.

The second kind of issues concerns the building and the management of the session. The session management system has to be flexible enough to handle most real-life cases. As the goal of multi-cluster communication systems is to allow users to easily build powerful computers out of their existing clusters, it would be unacceptable to reject a configuration just because it does not exactly fit some rigid schemes. Indeed, supporting (for instance) such a configuration where two

networks overlap – with a given network hardware (let's say Myrinet) available on every node and another hardware (e.g. SCI) only available on a part of the whole cluster – requires the communication support to consider every node to be a potential gateway. Yet, the increase in flexibility from the session management system should be followed by an increased care in the initialization sequence algorithms to avoid deadlocks while establishing the connections.

Madeleine III The version II of the *Madeleine* communication library already provided an efficient support for cluster-level multi-protocol communication to distributed programming environments ([2]). Yet, its use was restricted to network homogeneous clusters only: each node could only use exactly the same network hardware(s) (i.e. the same set of NICs) as the other ones. This limitation could be considered acceptable on single cluster configurations the library was designed for, but it is incompatible with multi-cluster support.

As an attempt to overcome *Madeleine II* limitations while still fully exploiting its original internal layers features, a preliminary study of an automatic multi-network data forwarding extension to *Madeleine II* was developed and presented at the HCW'01 workshop ([3]). This successful experience led the way to the conception and development of a new version of the *Madeleine* communication library presented in this paper. This third version builds on the *Madeleine II* buffer management and multi-protocol support engines to provide a high-performance multi-cluster networking platform. It integrates an entirely redesigned version of the multi-network data forwarding mechanism. This mechanism now supports any number of gateways and as been designed to suppress the risk of deadlocks that could be caused by processes communicating using a circular scheme, for instance.

It is coupled with a new session management module called *Leonie* and relies on the introduction of an additional data structure hierarchy for configuration representation in the internals of *Madeleine*. The *Leonie* module is responsible for parsing the application configuration file and for building the session configuration by spawning processes on selected networks/nodes and connecting them together. This paper mainly focuses on describing the last developments regarding the multi-cluster configuration management and the topology information system, and their integration with the automatic multi-network message forwarding mechanism. The preliminary performance results of this mechanism are very encouraging (see Section 5), with an asymptotic bandwidth at 43 MB/s over a SCI/Myrinet virtual connection, to be compared with the half bandwidth (the bus on the gateway node both receives data on the first network NIC and send the data again to the second network NIC) of the 132 MB/s PCI bus used.

This paper is organized as follows. The next section depicts the context of our work and the different issues to be addressed in supporting communication onto multi-cluster configurations. The section III details more thoroughly our contribution and presents the new features in *Madeleine III*. The section IV is dedicated to a more in-depth view of its new internals. The section V reports on the performance of the library, both as raw and forwarded data transfers are concerned. The section VI concludes this paper and briefly presents ongoing and future work.

2 Context

2.1 Multi-Cluster Environments

A carefull glance at the current picture of networking platforms research efforts around the world shows that most projects either target wide-area configurations or cluster-scale configurations. Yet, both approaches do not suitably address inbetween situations such as linking two clusters with a fast communication link.

Even though large scale metacomputing environments such as Globus would be able to support multi-cluster configurations, they are not well suited for that particular use. Metacomputing

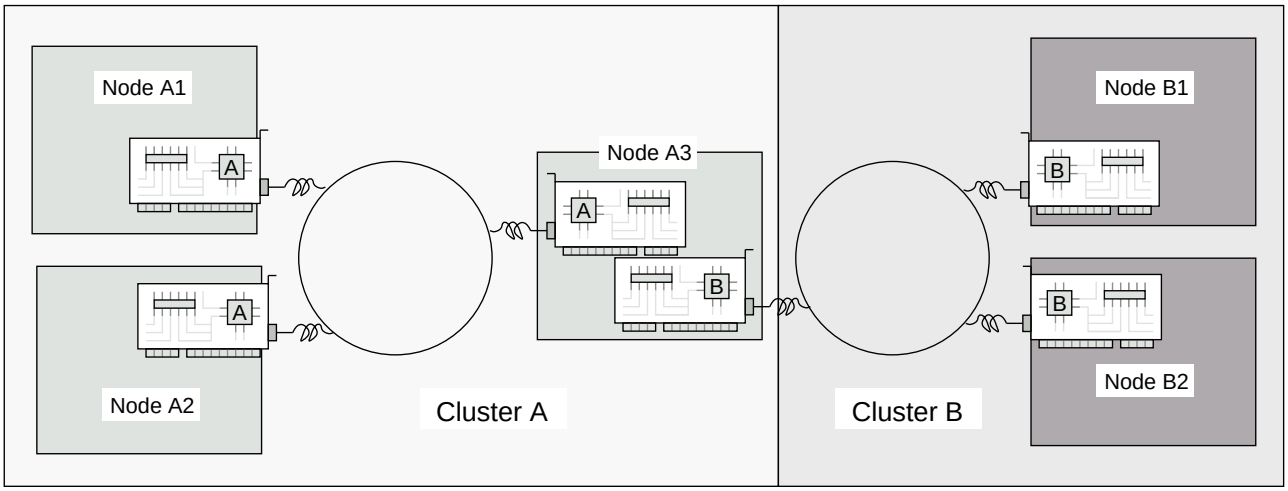

Figure 1: **Multi-cluster link.**

environments are equipped with very advanced mechanisms to address issues like security enforcement, resource reservation and allocation, or fault detection and tolerance. Yet, while these mechanisms are perfectly relevant in the metacomputing context where resources of many computation center have to be managed with the greatest care, it is clear that they would be rather oversized—or even irrelevant—for multi-cluster configuration. Even a light-weight version of such an environment wouldn't be efficient as metacomputing communication systems only use standard protocol stacks (like TCP in MPICH-G and PACX-MPI). Again, this approach is valid for long-distance communication between the nodes of a *metacomputer* which benefits from the added value of such protocol stacks. Yet, it will not allow to efficiently exploit a fast cluster interconnect.

Cluster scale communication libraries are not well suited for multi-cluster support either. Yet, while metacomputing environments are oversized for managing multi-cluster configurations, cluster scale communication libraries are rather undersized for that use. They have not been designed for integrating the increased complexity of such configurations. Consequently, multi-cluster configurations are either simply not supported (if the internal structures do not allow the storage of complex configurations) or they involve the application in managing communication between each network on the gateways. As a result, the former alternative is not usable and the later one implies a lot of extra-work from the application programmer, to be repeated for each new application port.

Nevertheless, research labs now commonly own two or more clusters of workstations. The availability of a fast, portable multi-cluster communication support would provide these labs, mostly for free, with a much higher aggregated computing power.

2.2 Supporting High-Bandwidth Inter-Cluster Networks

The main specific characteristic of multi-cluster configurations is the fact that the performance of the inter-cluster links is in the same order of magnitude as the intra-cluster networks. To put it in another way (see Fig. 1), the feature that can be identified as being common to any multi-cluster configuration is that at least one node of a cluster named *A* (node **A3** on the figure) built around a *A* networking hardware owns a *B* networking hardware NIC linked to the *B* cluster network (networks are symbolized by the circles on the picture). Any multi-cluster configuration is therefore a variation on this general scheme.

Hence, there is no fundamental difference between an intra-cluster link and an inter-cluster link. Therefore, a multi-cluster communication library should not make any difference between both kind

of links at the driver level. Instead, the new role in supporting multi-clusters of the communication layer will be to integrate routing and message steering functionalities to implement the gateways. Let's come back to the Figure 1. The node **A3** can be identified as a gateway node because it belongs to more than one network. Suppose the node **A1** needs to send a message to the node **B2**. Because **A1** and **B2** do not belong to the same network, **A1** will not be able to send the message directly to **B2**. Instead, the message will have to be relayed by the **A3** gateway. The node **A3** has to listen both for messages destinated to itself and for messages to be forwarded from the *A* network to the *B* network and conversely. Moreover, the node **A1** will have to determine that his message for **B2** has to be physically sent to **A3**. Hence the need for routing.

As a conclusion, the communication library has to handle inter-network data forwarding to build the missing physical links, even though networks might be made of different protocol/hardware pairs. The automation of that relaying job would even hide these missing links. This would show the whole multi-cluster architecture as a single fully connected virtual cluster to the applications and allow for the straightforward port of single-cluster applications.

2.3 From Madeleine II to Madeleine III

The version II of the *Madeleine* library ([2]) is a cluster scale communication library. It was designed to provide an efficient and yet portable communication support for distributed computing applications. It allowed multiple networks and multiple NICs per network to be used within the same computing session. Yet, it required each node of the configuration to use exactly the same set of NICS. For instance, suppose each workstation of a cluster owns both a Myrinet card and a Fast-Ethernet card, with the exception of one workstation equipped with a Myrinet card and two Fast-Ethernet cards. The *Madeleine II* library would allow an application to use both the Fast-Ethernet card and the Myrinet card on each *workstation* but only the Myrinet card and *one* among the two Fast-Ethernet cards would have been available for the application on the workstation with the three NICs. Consequently, the *Madeleine II* library as such was not ready for multi-cluster support.

Our preliminary study of an automatic multi-network forwarding extension for gateway support was conducted using an hacked version of *Madeleine II*. The hack basically forced Madeleine to ignore missing NICs and multi-cluster routing used hard-coded routing tables. It was definitely not a suitable version for production use, but the experiments we conducted with the forwarding extension were successful and started the development of the new version of *Madeleine* presented here. This version integrates a redesigned version of the initial fowarding extension which has now been designed with scalability and flexibility in mind. *Madeleine III* now natively supports any combination of union and inclusion of workstations clusters. At least one direct or indirect path must exist between each pair of nodes among the configuration if that configuration is to be seen as a single virtual cluster by the application, but it is not mandatory (a configuration not following this rule will simply generate several disconnected virtual networks instead of a main single one and the application will take the shape of independently running sets of processes). The multi-network forwarding extension handles data transfers in a transparent manner from the application point of view.

In order to replace the hard-coded routing tables used in the testing version of *Madeleine II*, the *Madeleine III* library now works together with a new external tool. This tool (called *Leonie*) is responsible for addressing many session management issues that were handled directly into *Madeleine* before. The initialization code of *Madeleine* has been greatly simplified by this new partition of responsabilities. The *Leonie* tool analyses the application configuration file and build the topology and routing tables from it. It then remotely spawns the session processes and connects them together. *Leonie* is also responsible for controling the session clean-up sequence.

These last developments are described in the next section.

```
application: {
  name: mad_test;

  networks: {
    include: networks.cfg;

    channels: ({
      name: first;
      net: bip_net;
      hosts: (tennessee, faure, ravel);
    }, {
      name: second;
      net: sisci_net;
      hosts: (muddy-waters, debussy,
              ravel);
    });

    vchannels: {
      name: global;
      channels: (first, second);
    };
  };
};
```

Figure 2: Example of application description file.

3 Contribution

Managing Heterogeneous Configurations with Leonie As explained above *Madeleine III* library relies on a new helper program called *Leonie* to perform configuration and session management duties. The *Leonie* tool use the services of a generic configuration file parser library (called *Leoparse*) to read and analyse the application configuration file. The Figure 2 shows an example of an application configuration file to be submitted to *Leonie*. *Madeleine* use the word *channel* for designing the interface abstraction of a network. The main function of the configuration file, beside indicating the application executable, is to map *Madeleine* channels onto physical networks and to map virtual channels (called *vchannels* in the fig. 2) onto regular channels. The difference between a regular *Madeleine* channel and a virtual channel is that the first kind is only made out of a set of direct physical links, while the second kind, the virtual channels, may contain nodes that do not belong to the same network. As such, virtual channels may involve the new forwarding extension to provide virtual point-to-point links between nodes that are not physically directly connected to each other.

From the application point-of-view, there is no difference between regular and virtual channels at the *Madeleine* communication interface level. The application selects the channel to use for sending a message by passing its name as parameter to the interface of *Madeleine*. Then, according to what was analyzed in the application configuration file by *Leonie*, that name is mapped onto its corresponding regular or virtual channel. In the example of the figure 2, if the application passes the string `"global"` to the interface of *Madeleine*, the virtual channel described in the configuration file will be used for the transmission. Changing the definition of the `global` channel (or simply using another configuration file) to the one of a regular channel would have the application communicate onto that regular channel, without even the need of recompiling it.

The *Leonie* tool builds a complete internal representation of the session configuration and of the routing information from the configuration file. It then starts the process spawning sequence. Yet, *Leonie* has to take the special case of network interface launchers into account. Indeed, several communications interfaces require that the processes be launched using a proprietary *black-box* launcher (e.g. `bipload` for the BIP/Myrinet interface, `mpirun` with many MPI implementations). Consequently, *Leonie* must sort the processes of the configuration into several sets according to the launcher needed to spawn it. Moreover, each launcher must have a specific interfacing stub in *Leonie* to handle issues such as building the command line. It is only after the sorting step that processes can effectively be spawned. Each set of processes is passed to its corresponding launcher interfacing stub, or to the common fallback generic loader. Each set is always launched as a whole and *Leonie* can be interfaced with optimized tree-based application launchers (in that case, such a launcher is used in the place of the fallback loader for the nodes where it is available). Once a process is launched, it calls back the *Leonie* server over a TCP link. This link will be used all along the session (but mainly during the initialization and clean-up sequences) to exchange information, but the very first exchange involves two steps. First, the application process identifies itself by sending information about is location, and then, *Leonie* sends the process rank number in the configuration. This rank number is very important as it uniquely identifies a process among the configuration. It should be noted that this rank number cannot be passed to the process as a command line argument during the spawning step because most black-box launchers just do not have support for sending process specific information.

Once the processes are launched, *Leonie* schedules the initialization sequences of each instance of *Madeleine*. Network drivers, network adapters are first initialized. They are followed by the initialization of the regular channels and finally by the initialization of the virtual channels. Such a tight synchronization is mandatory to avoid deadlocks during the setup of connections. The reverse sequence is used for session clean-up.

Beside these session management services, *Leonie* also provides multi-purpose services such as synchronization routines (over the whole session) and centralized display routines.

Multi-Cluster Networking over Virtual Channels As mentionned earlier, *Madeleine III* provides two *public* types of channels to the applications: the *regular* channels and the *virtual* channels. In the figure 2, the channels named `first` and `second` are related to a single physical network driver. These two channels are regular channels. The basic idea of the regular channel concept is that each point-to-point connection of a regular channel is either direct or inexistent. The communication interface of *Madeleine* only supports sending messages over point-to-point connections. Consequently, an application process running on a given node cannot send a message to another node over a regular channel if both nodes are not directly interconnected by a physical network link.

The *virtual* channels have been introduced to overcome that limitation while preserving the communication interface of *Madeleine*. Virtual channels are built over the regular channels. They rely on the use of the multi-network forwarding extension to fill in the gaps with virtual point-to-point connections were physical point-to-point connections are missing. In order to preserve communication consistency, a regular channel selected in a virtual channel is locked. It cannot be used directly by the application anymore. Each virtual channel consumes its regular channels. A regular channel cannot appear in more than one virtual channel, again for ensuring communication consistency. It should be noted however that both limitations could be easily removed by increasing the length of internal message headers. Such an approach would reduce the use of multiplexing resources, at the cost of decreased performances, especially at the regular channel level because the regular channel messages currently do not have headers.

Each regular channel involved in a virtual channel has a twin *forwarding* channel with identical characteristics. Forwarding channels are built automatically from regular channels information and they do not have to be declared in the application configuration file. A forwarding channel is private to *Madeleine* and may never be used by the application because it is permanently locked. Forwarding

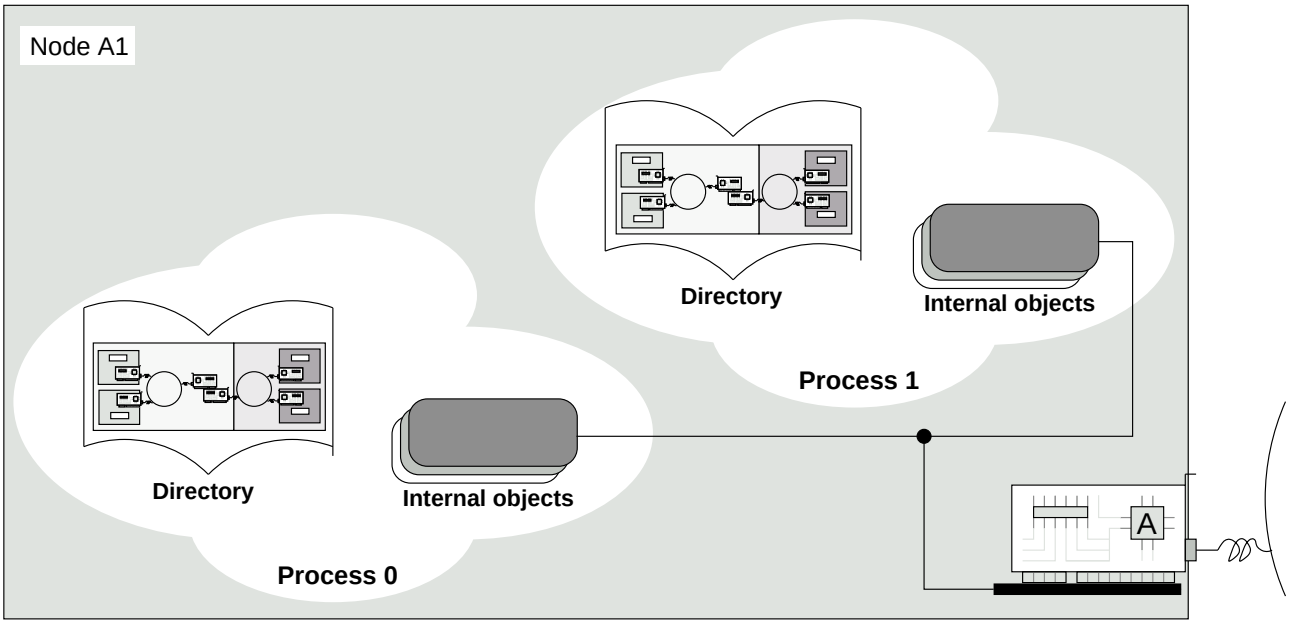

Figure 3: ***Madeleine* data structures.**

channels are used to convey *to-be-forwarded* messages while their twin regular channels only carries *direct* messages. This way, the gateways can easily tell messages targeted to themselves from messages to be relayed. Again, the use of forwarding channels could be replaced by extra information in message headers to reduce multiplexing resource usage, at the cost of a reduced level of performance.

4 Implementation

We now details the implementation of the changes induced by the addition of multi-cluster support into *Madeleine*. Among those changes, internal data structures were partially redesigned and extended to support the specificity and the increased complexity of multi-cluster configurations.

Configuration Directory The internals of the *Madeleine II* communication are built around an object-oriented data structure stack. The function of this structure was to provide an internal representation for concepts such as drivers, adapters, channels and point-to-point connections. Because *Madeleine II* requires each node of the configuration to use exactly the same set of NICS, each node has the same set of objects. As such, each node immediately knows the configuration details of any other node just by looking at its own configuration.

With the introduction of multi-cluster configurations, this is not true anymore. For instance, a given regular channel that would be unavailable on several nodes just wouldn't even appear in the object stack of these nodes. As a consequence a difference must now be made between the internal representation of the information about the whole configuration and the internal representation of the state of the software and hardware communication components (adapters, channels, etc.) The first kind of information represents data describing the entire configuration and can be considered to remain static during the session. The second information is a dynamic and instantaneous view of the state of the communication library instance on a given node. A new object stack was integrated into *Madeleine III* to complete the existing one by storing configuration information separately. This data structure is called the *configuration directory* (or simply just directory) and is represented in the

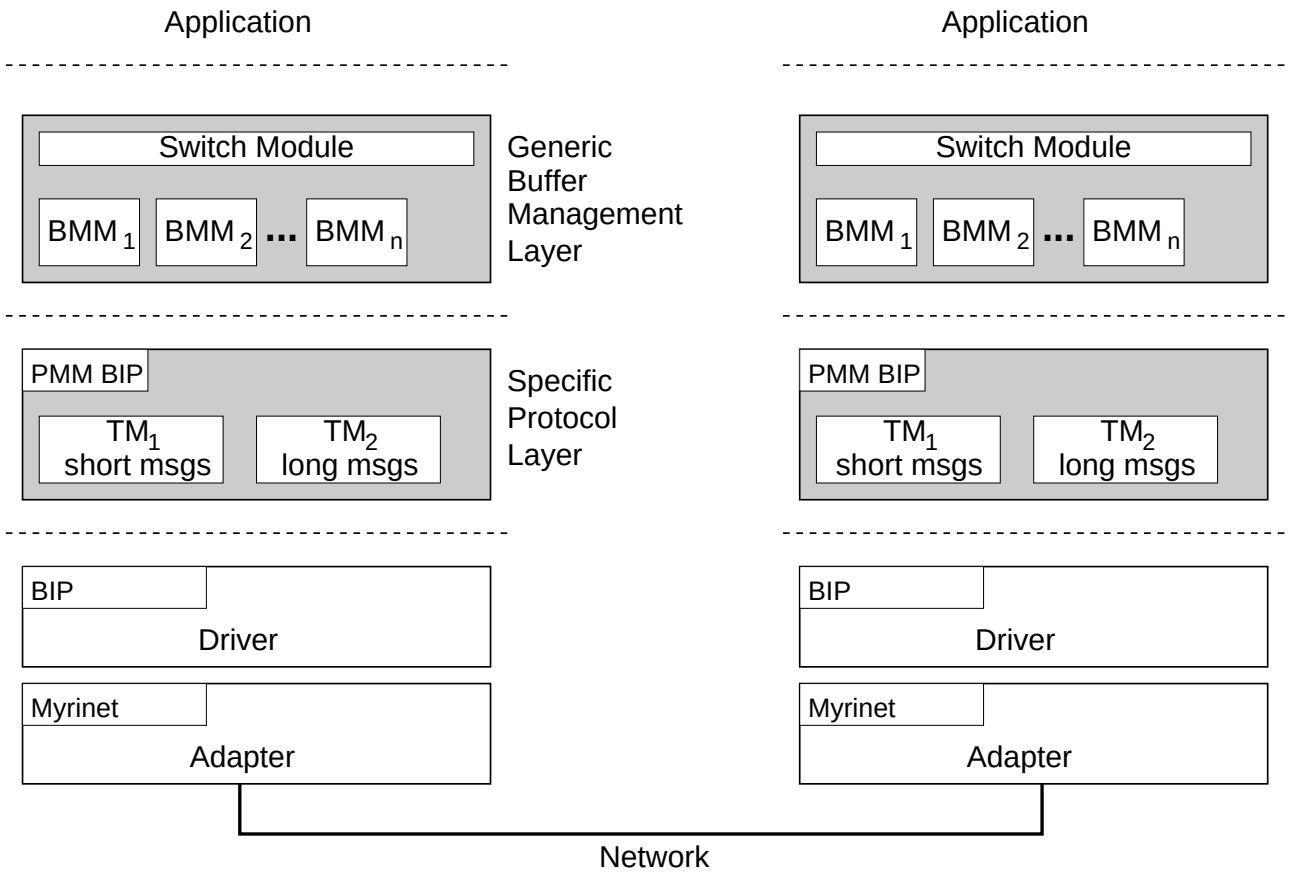

Figure 4: ***Madeleine* internal architecture.**

figure 3. It is made of a hierarchy of tables and lists of objects and can be considered as a kind of knowledge base about the whole configuration.

The *Leonie* server is responsible for building the master copy of the directory out of the information extracted from the application configuration file. Once the directory has been initialized and the processes have been launched, a copy of the directory is sent to each process of the configuration. Most of the information is just copied as-is from the configuration file to the directory. The remaining is essentially made of the routing tables of the virtual channels and is computed by *Leonie*. The algorithm currently used for computing network routes is a simple adaptation of a regular *all shortest path* algorithm. However it could easily be replaced with a more clever algorithm. Such an algorithm could take, for instance, network performance characteristics or gateway balancing into account to build better inter-network routes. Yet, whatever the algorithm chosen, the routing tables cannot currently be changed dynamically during the session in this implementation.

Multi-Threaded Data Forwarding We now describe the implementation of the multi-network forwarding extension included in *Madeleine III*. This part requires a basic knowledge of the internals of the *Madeleine* library so we give a short description of these internals first, before the actual presentation of the extension.

The architecture of *Madeleine* presented in [1] is summarized in figure 4. It uses a traditional approach with a high level portable layer for buffer management and a low level layer for network hardware interfacing. The high level layer is made of several buffer management modules (BMM), each implementing its own management policy. The network layer is also made of several modules called transmission modules (TM) and protocol management modules (PMM). A protocol module

is basically a network driver for *Madeleine*. It must contain at least one transmission module, but there is no upper limit on their number. Each transmission module implements a way to use a given network interface (e.g BIP for Myrinet) or a network protocol (e.g. TCP).

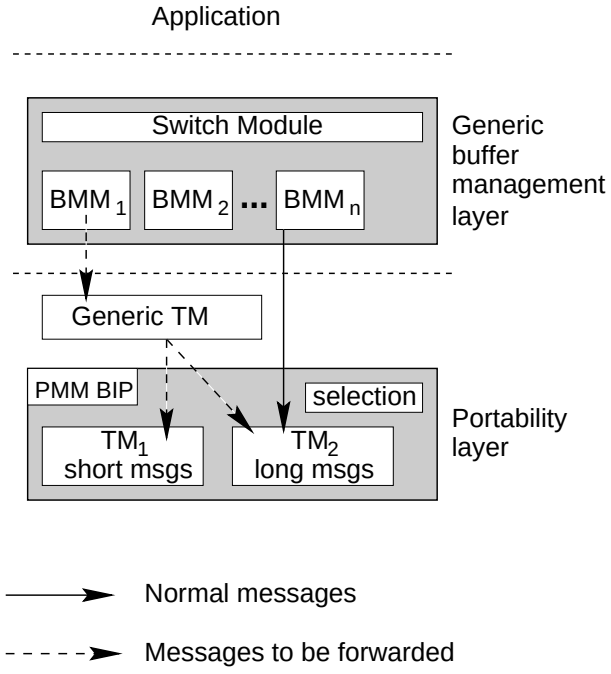

Figure 5: ***Madeleine* forwarding module.**

The whole forwarding module was designed to perfectly mimic the specifications of the transmission module interface and the protocol module interface of *Madeleine* (see figure 5). Indeed, this solution has two major advantages. The first one is that this way, the messages to be forwarded on the gateways are kept from traversing the whole *Madeleine* stack twice (which was one of the main objectives in the design of this extension). The second advantage is that the forwarding module runs un-intrusively with respect to the *Madeleine* buffer management layer: implementing the forwarding extension doesn't required a major redesign of the buffer management layer. Moreover, the forwarding module also use the protocol module interface and the transmission module interface – but now from the *consumer* point-of-view — to control the actual underlying network modules. Another main objective of the forwarding extension was the overall transparency of the forwarding mechanism for the application. The forwarding module included in *Madeleine III* relies on the services of the *Marcel* user-level multi-threading library [5]. The use of threads to receive and retransmit messages allows the mechanism to run un-intrusively with respect to the application code running on the gateway nodes.

The *Madeleine* communication interface is based on the paradigm of incremental message building and extraction. A message is initiated with a call to the `begin_packing` routine, incrementally built with a series of calls to the `pack` routine and finalized by a call to the `end_packing` routine. The message extraction scheme is just the same. The buffer management layer of *Madeleine* is then responsible for virtually aggregating (or not) each packed piece of data together before eventually passing them to a transmission module for actually putting them on the wire. Consequently, a given *Madeleine* message may well require several transactions with a transmission module to be integrally sent over the wire. Moreover, there is no bound over the delay between each successive transaction, and hence there is no bound over the whole message construction duration either. On a regular channel directly used by the application, the corresponding point-to-point connection would be reserved

during the whole message construction, whatever its duration. The same is true for a virtual point-to-point connection of a virtual channel. However, the underlying direct connections that build this virtual connection cannot be locked for so long as they may meanwhile be needed by other virtual connections. Doing so would unavoidably lead to deadlocks. As a consequence, the atomic transmission unit used by the forwarding module internally is not the whole message but a unit simply called *block*. The forwarding module blocks are made of a fixed length header and a variable length payload. The link reservation is limited to a single direct connection (from the sender to the first relay, or between two further transmission relays) and its duration is limited to the time to send a block.

This assertion means that blocks being part of different messages may travel interleaved on the same wire. This is not a problem because these blocks either differ by their source or by their destination. Remember, a virtual connection is locked during the whole message construction and transmission. Therefore, there cannot be interleaved messages on the same virtual connection, and hence there cannot be any interleaved messages with both the same source and destination on the same wire. The last relay of each virtual connection (the one retransmitting a block to its actual destination) is responsible for sorting the blocks. It accumulates the messages blocks for each destination it is responsible for, because the flow of data over the last direct connection of a virtual connection is controlled by the receiver. As a result, a worst case situation where a gateway keeps receiving blocks of data while their actual destination processes never ask for them could end up in the gateway memory being exhausted. To avoid this problem, *Madeleine III* also integrates an optional flow control mechanism over whole virtual lines.

5 Evaluation

This section presents an evaluation of the overall *Madeleine III* performance on the networks we have been using to test the previous version of *Madeleine*, to allow for an easier comparison with previous papers.

The following performance results were obtained using a cluster of dual INTEL PENTIUM II 450 MHz PC nodes with 128 MB of RAM and a 33 MHz 32bit PCI bus running LINUX Kernel 2.2.13. The cluster interconnection networks are 100 Mb/s FAST-ETHERNET for TCP, DOLPHIN SCI ([8], [7]) (D310 NICs) for SISCI and MYRINET (LANai 4.3, 32bit bus, 1 MB SRAM) for BIP [10]. Results were obtained by ping pong tests (half round-trip).

Raw Performances Raw performance results for *Madeleine III* over TCP/Fast-Ethernet, SISCI/SCI and BIP/Myrinet are displayed in figures 6, 7 and 8. The performance of *Madeleine III* over SISCI/SCI shows that the inherent overhead of *Madeleine III* is very low. The minimal latency achieved by *Madeleine III* over this network is around 5.3 μs. As, the raw SISCI/SCI latency is around 2 μs, the *Madeleine III* raw overhead may be estimated to be between 3 and 4 μs. Yet, the *Madeleine III* source code is not currently over-optimized. Indeed, this low overhead is due to the short and rather straightforward critical path of the transmission routines, which also allows for transfer rates to fill most of the available hardware bandwidth on both SISCI/SCI and BIP/Myrinet (and of course TCP/Fast-Ethernet). The internals and optimizations of *Madeleine* SISCI/SCI and BIP/Myrinet drivers have been thoroughly described in [1] and did not change significantly during the transition between the version II and the version III of *Madeleine*. Therefore, they won't be presented in this paper again.

Forwarding Performances The performance of data transmission over *Madeleine III*'s virtual channels is shown in figure 9. The test program performs a ping-pong with messages traveling back and forth over a virtual connection going through a BIP/Myrinet link and a SISCI/SCI link. Expectedly,

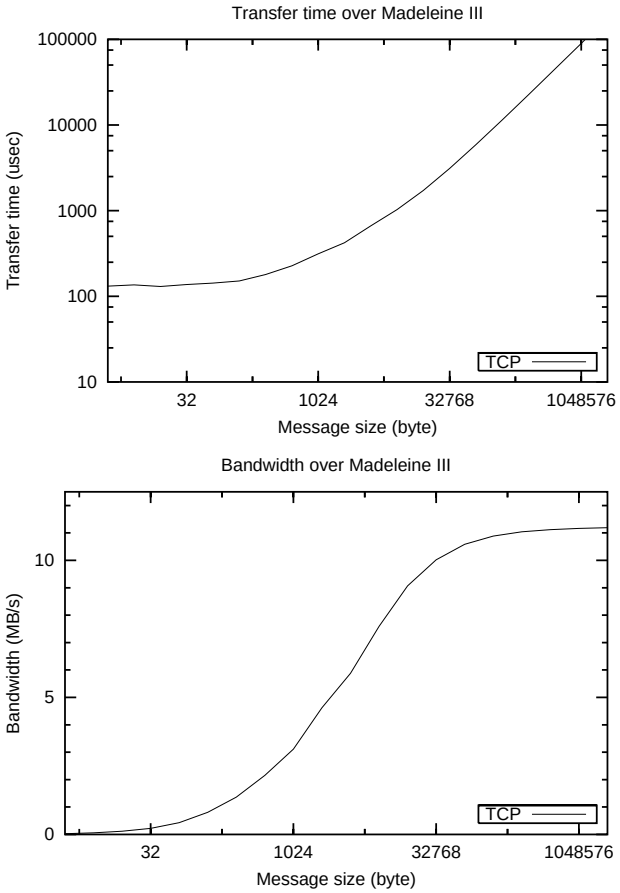

Figure 6: Latency and bandwidth over TCP/FAST-ETHERNET

the 50 μs minimal latency is quite high. There two main reasons for that high latency. The first reason is that messages now have to go through two nodes in the test, instead of only one in the raw experiments with *Madeleine*'s drivers. The second reason is that the data blocks that travel over a virtual connection are made of two pieces of data (the header and the body) while messages sent over regular channels don't have an header. Moreover, the cost of that additional header is also amplified by the additional node to go through. Yet, it should be noted that this 50 μs minimal latency is only half the minimal TCP/Fast-Ethernet latency. However, it is not the main focus of the forwarding mechanism and this interesting result compared to the TCP latency is only a nice side-effect.

The asymptotic bandwidth reaches 43 MB/s, which is a very good result indeed. The gateway node PCI bus has to be shared between both a SCI NIC and a BIP NIC, so the theoretical maximum bandwidth achievable is 66 MB/s (half the 132 MB PCI bus bandwidth). This theoretical limit is for one-way burst data transfers only. Taking into account the fact that in the gateway context, the two NICs unavoidably conflict for accessing the bus, the 43 MB/s bandwidth obtained can be considered an interesting result.

6 Conclusion and Future Work

In this paper, we presented the new version of the *Madeleine* communication library and the modifications induced by the addition of a support for multi-cluster configurations to it. This transition involved design and redesign work inside the library as well as the development of an external helper

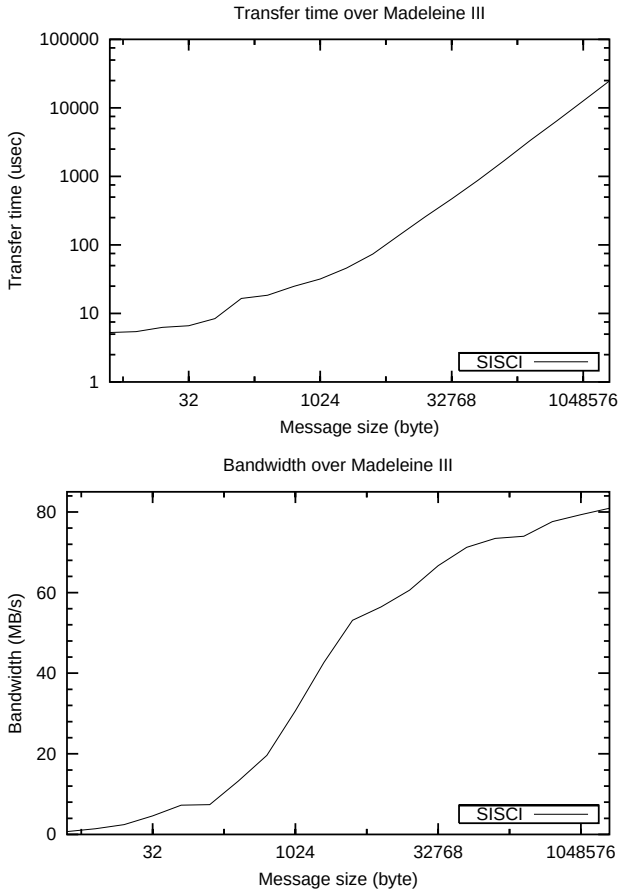

Figure 7: Latency and bandwidth over SISCI/SCI

program. This program, called *Leonie* removes the session management issues from the tasks of the *Madeleine* library. It is in charge of duties such as analyzing application configuration files, building the internal representation of the configuration and the inter-cluster routing tables, launching the session processes and finalizing the session.

Madeleine internals have been extended with a new data structure for representing static information about the network components (drivers, channels, etc.) of the configuration. This data structure called *configuration directory* completes the existing one which stores information about the current state of the *Madeleine* library instance running as part of an application process. The *Madeleine* interface has been left untouched by this transition to multi-cluster support. Cluster networks are still presented as channels although some channels may now be virtual. As a result, an application can run unmodified on this new multi-cluster version of *Madeleine*. Such an application may later be optimized by accessing the configuration directory to retrieve topology information and use it to adopt clever communication schemes. *Madeleine* also includes a redesigned version of its automatic multi-network message forwarding extension. This extension handles gateway traffic in a transparent manner using a multi-threaded approach. This transparent approach is the key part of the virtual channels as it provides for the building of the virtual connections between nodes standing on different networks.

The preliminary experiments we conducted with *Madeleine III* showed the low overhead of the library and the encouraging results of the virtual network support. We now intend to focus our work on our adaptation of MPI over *Madeleine* [4] to efficiently import the new multi-cluster capabilities of *Madeleine III*.

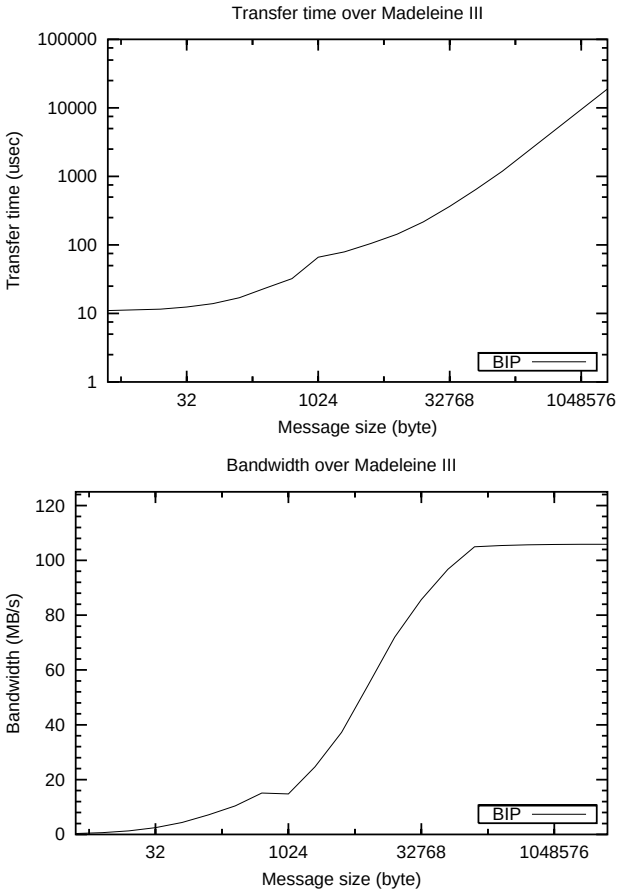

Figure 8: Latency and bandwidth over BIP/MYRINET

References

[1] O. Aumage, L. Bougé, A. Denis, J.-F. Méhaut, G. Mercier, R. Namyst, and L. Prylli. A portable and efficient communication library for high-performance cluster computing. In *IEEE Intl Conf. on Cluster Computing (Cluster 2000)*, pages 78–87, Technische Universität Chemnitz, Saxony, Germany, Nov. 2000.

[2] O. Aumage, L. Bougé, and R. Namyst. A portable and adaptative multi-protocol communication library for multithreaded runtime systems. In *Parallel and Distributed Processing. Proc. 4th Workshop on Runtime Systems for Parallel Programming (RTSPP '00)*, volume 1800 of *Lect. Notes in Comp. Science*, pages 1136–1143, Cancun, Mexico, May 2000. Held in conjunction with IPDPS 2000. IEEE TCPP and ACM, Springer-Verlag. Electronic version available.

[3] O. Aumage, L. Eyraud, and R. Namyst. Efficient inter-device data-forwarding in the Madeleine communication library. In *Proc. 15th Intl. Parallel and Distributed Processing Symposium, 10th Heterogeneous Computing Workshop (HCW 2001)*, page 86, San Francisco, Apr. 2001. Held in conjunction with IPDPS 2001. Extended proceedings in electronic form only.

[4] O. Aumage, G. Mercier, and R. Namyst. MPICH/Madeleine: a true multi-protocol MPI for high-performance networks. In *Proc. 15th International Parallel and Distributed Processing Symposium (IPDPS 2001)*, page 51, San Francisco, Apr. 2001. IEEE.

[5] V. Danjean, R. Namyst, and R. Russell. Integrating kernel activations in a multithreaded runtime system on Linux. In *Parallel and Distributed Processing. Proc. 4th Workshop on Runtime Systems for Parallel Programming (RTSPP '00)*, volume 1800 of *Lect. Notes in Comp. Science*, pages 1160–1167, Cancun, Mexico, May 2000. Held in conjunction with IPDPS 2000. IEEE TCPP and ACM, Springer-Verlag.

[6] E. Gabriel, M. Resch, T. Beisel, and R. Keller. Distributed Computing in a Heterogeneous Computing Environment. In V. Alexandrov and J. Dongarra, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Lecture Notes in Computer Sciences. Springer, 1998.

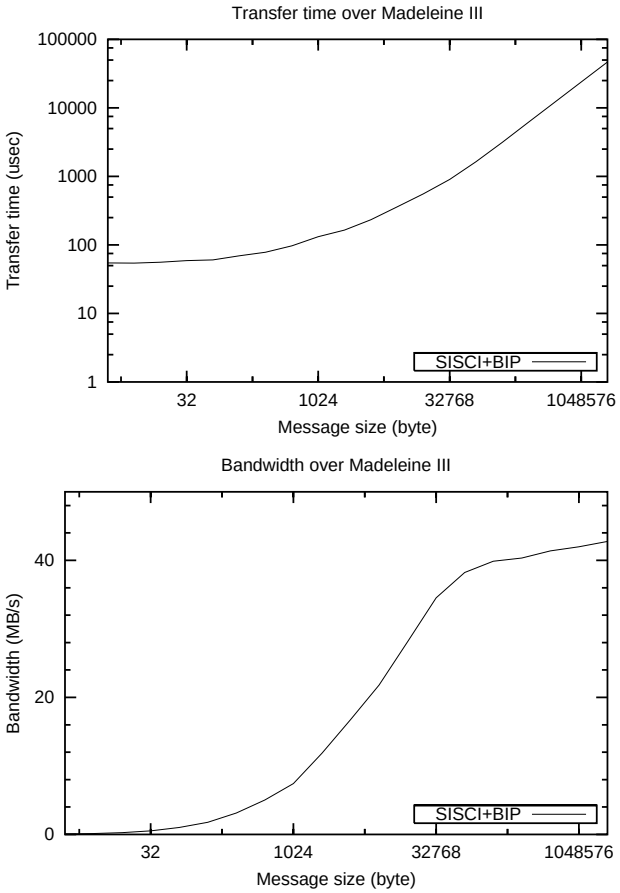

Figure 9: Latency and bandwidth over BIP/MYRINET + SISCI/SCI

[7] H. Hellwagner and A. Reinefeld, editors. *SCI: Scalable Coherent Interface, Architecture and Software for High-Performance Compute Clusters*, volume 1734 of *Lect. Notes in Comp. Science, State-of-the-Art Surveys*. Springer-Verlag, 1999.

[8] IEEE. *Standard for Scalable Coherent Interface (SCI)*, Aug. 1993. Standard no. 1596.

[9] N. Karonis, B. de Supinski, I. Foster, W. Gropp, E. Lusk, , and J. Bresnahan. Exploiting hierarchy in parallel computer networks to optimize collective operation performance. In *Fourteenth International Parallel and Distributed Processing Symposium (IPDPS '00)*, pages 377–384, Cancun, Mexico, May 2000.

[10] L. Prylli and B. Tourancheau. BIP: a new protocol designed for high performance networking on Myrinet. In *1st Workshop on Personal Computer based Networks Of Workstations (PC-NOW '98)*, volume 1388 of *Lect. Notes in Comp. Science*, pages 472–485. Held in conjunction with IPPS/SPDP 1998, Springer-Verlag, Apr. 1998.

7 Biography

Olivier Aumage is a PhD. Student at the *Laboratoire de l'Informatique du Parallélisme* (LIP) computer science laboratory (École Normale Supérieure de Lyon, France). He is the main designer of the *Madeleine II* and *Madeleine III* message passing communication libraries. The *Madeleine III* library is distributed as part of the PM^2 multithreaded distributed programming environment (`http://www.pm2.org`).

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

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