Protocols and Software for Exploiting Myrinet Clusters

P. Geoffray[1], C. Pham, L. Prylli[2], B. Tourancheau[3], R. Westrelin

Laboratoire RESAM, Université Lyon 1
[1]Myricom Inc., [2] ENS-Lyon, [3] SUN Labs France
Congduc.Pham@ens-lyon.fr

Abstract. A cluster, by opposition to a parallel computer, is a set of separate workstations interconnected by a high-speed network. The performances one can get on a cluster heavily depend on the performances of the lowest communication layers. In this paper we present a software suite for achieving high-performance communications on a Myrinet-based cluster: BIP, BIP-SMP and MPI-BIP. The software suite supports single-processor (Intel PC and Digital Alpha) and multi-processor machines, as well as any combination of the two architectures. Additionally, the `Web-CM` software for cluster management that cover job submissions and node monitoring is presented as the high-level of the software suite.

1 Introduction

In the past 5 years, there has been a tremendous demand, and offer, on cluster architectures involving commodity workstations interconnected by a high-speed network such as Fast Ethernet, Gigabits Ethernet, Giganet, SCI and Myrinet. These architectures are often referred to as Network Of Workstations (NOW) or high-performance clusters (HPC). Several research teams have launched projects dealing with NOWs used as parallel machines. Previous experimentations with IP-based implementations have been quite disappointing because of the high latencies of both the interconnection network and the communication layer. Therefore, the goal of most research groups is to design the software needed to make clusters built with commodity components and high-speed networks really efficient. The NOW project of UC Berkeley [1] was one of the first projects.

The performances one can get on a cluster heavily depend on the performances of the lowest communication layers. Previous experiences have shown that efficient communication layers and fast interconnection networks must be present altogether to build a high-performance cluster. The availability of HPCs, at an affordable price, and adequate communication software is a great opportunity for the parallel processing community to bring these techniques to a larger audience.

In this paper we present a software suite for achieving high-performance communications on a Myrinet-based cluster: BIP, BIP-SMP and MPI-BIP. The software suite supports single-processor (Intel PC and Digital Alpha) and multi-processor machines, as well as any combination of the two architectures. It can be viewed as a collection of highly optimized components that contribute at each level of the cluster communication architecture to provide the maximum of performance to the end-user. It is also possible for the end-user to choose at which level he wants to program, knowing as a rule of thumb that the lowest level usually provides high-performances but less functionalities. Each of these components has been previously described in the literature so the main motivation for this paper is to present to the parallel computer users a bottom-up approach for efficiently exploiting on a daily basis a Myrinet cluster. Of course, the problems covered by cluster-based computing is much broader than the communication-oriented problems we focused on. Issues such as cluster management, check-pointing and load-balancing are also very important. In many cases, these features are desperately lacking on cluster-based environments, as opposed to traditional (expensive and mainly proprietary) parallel computer environments. In this first attempt, however, we will address more specifically the high-performance communication issues as we believe that this point may drive the first motivation to move from massively parallel computer to a cluster-based solution. Additionally, the `Web-CM` software for cluster management that cover job submissions and node monitoring is presented as the high-level of the software suite.

The rest of the paper is organized as follows. Section 2 presents the hardware characteristics of the interconnection network. Section 3 presents the low-level communication layers and Section 4 presents the customized MPI communication middle-ware. Related works are presented in Section 5 and performance measures in Section 6. `Web-CM` is described in Section 7 and we present our conclusions in Section 8.

2 The Myrinet hardware

The Myrinet communication board uses a PCI slot to connect a node to a Myrinet switch [2]. The bandwidth provided by the network is approximately 160 MBytes/s (1.2Gbits/s), but the PCI bus (32 bits, 33 Mhz) limits the maximum throughput to 132 MBytes/s. All links are full-duplex and the Myrinet switch is a full cross-bar operating a source-based routing algorithm. There are several features that make this kind of technology much more suitable than a traditional commodity network:

- The hardware provides an end-to-end flow control that guarantees a reliable delivery and alleviates the problem of implementing in software the reliability on top of an lossy channel. As message losses are exceptional, it is possible to use algorithms that focus on very low overheads in the normal case.
- The interface card has a general purpose processor that can be programmed in C. The code, called Myrinet Control Program (MCP), is downloaded at the initialization of the board. It is powerful enough to handle most of the communication activity without interrupting the main processor.

– The interface card has up to 8 megabyte memory for buffers (LANai 9). This memory compensates in some cases (contention on I/O bus, on network...) the throughput difference between the communication components.

3 Low-level communication layers

3.1 BIP

At the lowest level of the communication architecture, BIP (Basic Interface for Parallelism) [3, 4] provides an efficient access to the hardware and allows zero memory-copy communications. It has been optimized for low latency, high throughput and a rapid throughput increase. As BIP supplies very limited functionalities, it is not meant to be used directly by the parallel application programmer. Instead, higher layers (such as MPI) are expected to provide a development environment with a higher functionality/performance ratio.

The BIP's API is a classical message-passing interface: it provides both blocking and non-blocking communication (`bip_send`, `bip_recv`, `bip_isend`, `bip_irecv`, `bip_wait`, `bip_probe`...). Communications are as reliable as the network, errors are detected and in-order delivery is guaranteed. BIP is composed of a user library, a kernel module and a NIC program. The key points of the implementation are:

A user level access to the network. Avoiding system calls and memory copies implied by the classical design becomes a key issue: the bandwidth of the network (160 MBytes/s in our case and 132 MBytes/s for the I/O bus) is equivalent to the bandwidth of the memory (300 MBytes/s for memory read and 160 MBytes/s for a copy on a computer with a BX chip set).

Long messages follow a rendez-vous semantic: the receive statement must be posted before the send is completed. Messages are split into chunks and the different steps of the communication are pipelined.

Small messages. Since initializations and handshakes between the host and the NIC program are more expensive than a memory copy for small messages so they are written directly in the network board memory on the sending side, and copied in a queue in main memory on the receiving side. The size of this queue is statically fixed and the upper layers must guarantee that no overflow occurs.

Highly optimized, the raw communication performance for BIP is about $5\mu s$ latency one-way. The maximal bandwidth is 126 MBytes/s for a LANai 4 on a PCI 32 bits 33 Mhz (95% of the theoretical hardware's limit residing, in our case, in the PCI bottleneck). Half of the maximum bandwidth is reached with a message size of 4 KBytes. There is a distinction between small and large messages at 1024 bytes.

3.2 BIP-SMP

In the context of SMP nodes, BIP is not able to exploit all of the hardware performance as only one process per node can gain access to the Myrinet board while the other processors must remain idle for communication. BIP-SMP [5] provides the support of several processes per node. The difficulties in doing so are: (*i*) to manage the concurrent access to the hardware and the Myrinet board and, (*ii*) to provide local communications with the same level of performance as BIP over Myrinet. The key points of the implementation are:

Handling the concurrent access to the network. The concurrent access to the send request queue is managed by a lock. In the current implementation on the Linux OS, the lock uses a specific function provided by the kernel (**test_and_set_bit** in kernel 2.2.x). This function guarantees the atomicity of the memory operation. The cost of the lock operation is small compared to IPC system V locks or the `pthread` library locks. With BIP-SMP two processes can overlap the filling of a send request queue. The only operation that needs serialization is to obtain an entry in the send request queue.

Managing Internal communication. For efficiency reasons, BIP-SMP uses both shared memory to implement mailboxes and direct data transfer from user space to user space. The shared memory scheme moves small buffers with two memory copies but small latency, and the direct copy scheme moves large messages with a kernel overhead but a large sustained bandwidth. The shared-memory strategy needs one queue per communicating peer and the amount of shared memory needed increases by the square of the number of local processes. However, as commodity SMP nodes usually contain 2 or 4 processors, the implementation is justified. The direct-memory copy feature is provided by implementing a Linux kernel module to move data from one user space to another user space.

Multi-protocol layer. Another part of this work is to enable BIP-SMP to simultaneously use both remote communications and local communications while hiding this new feature in the BIP's API. We use two independent pools of receive queues per node: one for the internal communications and the other one for remote communications on Myrinet. Then we can allow the receipt of a message from the Myrinet network and the receipt of a message from another process in shared memory at the same time without any synchronization. The use of BIP-SMP is completely transparent as each process receives a different logical number. Everything else is hidden by the BIP-Multi-protocol layer. Variables are available to provide the information about the location of the other logical nodes, the number of processes on the same physical node, etc.

4 MPI-BIP: the communication middle-ware

MPI-BIP [6] is the privileged middle-ware for the end-user. It is a high performance implementation of MPI [7] for Myrinet-based clusters using the BIP protocol. The current MPI-BIP implementation is based on the work done by L. Prylli on MPI-GM. Most of the code is now shared between MPI-GM and MPI-BIP.

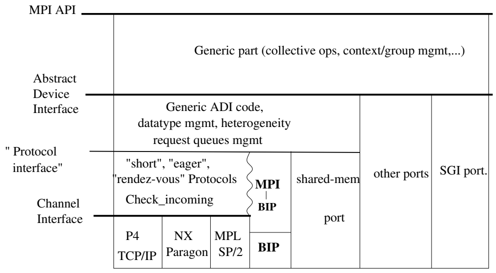

Fig. 1. The architecture of MPI-BIP

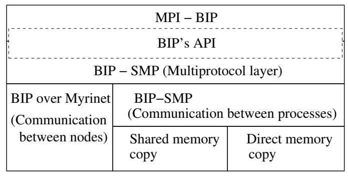

Fig. 2. Architecture of BIP-SMP.

MPICH is organized in layers and designed to facilitate the porting to a new target hardware architecture. Figure 1 presents our view of the MPICH framework and shows at what level we inserted the MPI-BIP specific part. Different ports choose different strategies depending on which communication system they use. We implemented our network specific layer at a non-documented interface level that we will call the "Protocol Interface". This API allows us to specify custom protocols for the different kinds of MPI messages. Each MPI message of the application is implemented with one or several messages of the underlying communication system (BIP in our case). The main contribution of MPI-BIP are:

- As BIP's flow control for long messages relies on the hardware flow-control, it is not sufficient when one side is not able to receive for a long time. For small messages, MPI-BIP uses a credit-based flow control taking into account the size of the BIP's queues.
- MPI-BIP uses request FIFO to allow multiple non-blocking operations.

Figure 2 shows the architecture of the BIP-SMP module within MPI-BIP. The complete view of the communication software architecture can be obtained by replacing the MPI-BIP and BIP blocks in figure 1 by figure 2. We chose to maintain the split view for simplicity.

5 Related works

First introduced in 1997, BIP was more an incremental step in a large family of software than a complete new design. Especially on Myrinet, we can find many other communication systems: Active Messages from Berkeley[8], Fast Messages from Illinois [9] and U-Net from Cornell University [10]. All these systems bypass the operating system to shorten the communication critical path and to limit, or avoid completely, memory copies for bandwidth improvements. BIP is however a very efficient implementation.

More recently, the efficient usage of clusters of shared-memory multi-processors (CLUMPs) has gain attention. We have investigated issues related to a multi-protocol message passing interface using both shared memory and the interconnection network within a CLUMP. Several projects have proposed solutions for this problem in the last few years and BIP-SMP is in the same research line. Projects like MPI-StarT [11] or Starfire SMP Interconnect use uncommon SMP nodes and exotic networks but performances are limited. Multi-Protocol Active Messages [12] is an efficient multi-protocol implementation of Active Messages using Myrinet-based networks and Sun Enterprise 5000 as SMP nodes. Multi-Protocol AM achieves 3.5 μs of latency and 160 MBytes/s of bandwidth. The main restriction is the use of the Sun Gigaplane memory system instead of a common PC memory bus. The polling is also a problem in Multi-Protocol AM as polling for external messages is more expensive than for internal messages. However, Multi-Protocol AM is the first message passing interface to efficiently manage CLUMPs. Finally, one of the first message-passing interfaces to manage CLUMPs as a platform is the well-known device P4 [13] used by MPICH. P4 provides mechanisms to start multiple processes on hosts and uses either message passing or shared memory copies to communicate between these processes. However, the programmer must explicitly select the appropriate library calls. We can also cite implementations of MPI limited to a single SMP node, like the MPICH devices **ch_shmem** or **ch_lfshmem** . The device **ch_lfshmem** is a lock-free shared memory device that achieves very good performance (2.4 μs and 100 MB/s on one of our SMP nodes). Regarding the shared memory management, some works about concurrent access for shared memory Active Messages [14] presents very efficient solutions such as a lock-free algorithm and a high performance lock implementation.

6 Performance measures

For all measures, the operating system is Linux. Figure 3 and 4 shows the performances of BIP and MPI-BIP for LANai 7 and LANai 9 (the latest board available). The additional cost of MPI-BIP is approximately 4 μs for a 0-byte messages (mainly CPU) over BIP for the latency on our test-bed cluster. Note that the latency of MPI-BIP depends in a large part on the processor speed as the network part of the latency is very small. For instance, as the processor speed is increased, the latency of MPI-BIP is decreased. The results obtained with a LANai 9 on a 2Gbits/s link are still experimental and performed with a back-to-back configuration as no switch was available yet. The jumps in the latency curves come from the BIP distinction between small and large messages at 1024 bytes. For the LANai 9, this distinction is beneficial for the latency. In figure 3, one jump in the MPI-BIP curve comes from BIP; the other one, occurring a bit earlier, comes from the way MPI-BIP switches from short to a three-way eager strategy.

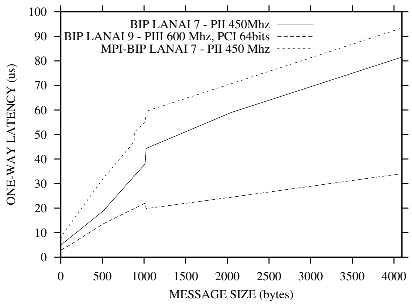

Fig. 3. BIP and MPI-BIP latency.

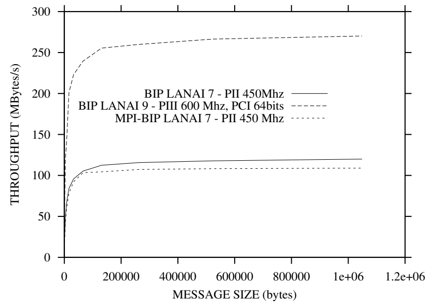

Fig. 4. BIP and MPI-BIP bandwidth.

Table 1 shows the raw point-to-point communications performance of BIP-SMP and MPI-BIP/BIP-SMP. The experimental platform consists in a cluster of 4 dual Pentium II 450 MHz 128 MBytes SMPs interconnected by a LANai 7 Myrinet network. We measured the latency and the bandwidth between two processes using ping-pong communications, on the same node and on two different nodes of the cluster.

Table 1. Pt-to-pt communications with BIP-SMP and MPI-BIP/BIP-SMP

Architecture	*BIP-SMP*	*MPI-BIP/BIP-SMP*
Intra-node latency (Shared memory)	1.8 μs	3.3 μs
Inter-node latency (Myrinet network)	5.7 μs	7.6 μs
Intra-node bandwidth (Shared memory)	160 MBytes/s	150 MBytes/s
Inter-node bandwidth (Myrinet network)	126 MBytes/s	107 MBytes/s

We then compared the latency and the bandwidth of MPI-BIP/BIP-SMP with several other related works: the **ch_shmem** and **ch_lfshmem** MPICH devices; MPICH over GM and MPI-PM/CLUMP (with `mpizerocopy` flag). We used the benchmark program **mpptest** included in the MPICH distribution. This software measures the latency and the bandwidth of a network architecture using a round-trip communication with blocking calls. The tests (cf. figure 5 and 6) are performed by varying the size of the packets sent between two processes on the same node node for the **intra-node** tests and between two processes on two different nodes for the **inter-node** tests.

7 `Web-CM`: Executing programs on the Myrinet cluster

A number of steps must be performed before programs can be executed on a Myrinet cluster. These steps are, in a chronological order: install the Myrinet hardware, install the BIP software suite, and run `biproute` to determines the topology of the cluster. We will not describe these steps as they are very specific to the user's hardware and operating system configuration. We will describe

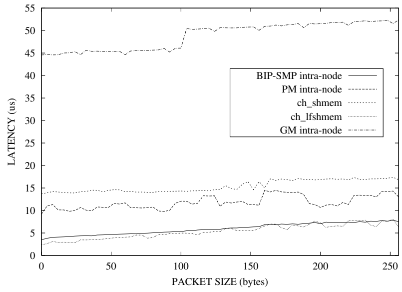

Fig. 5. Intra-node MPI.

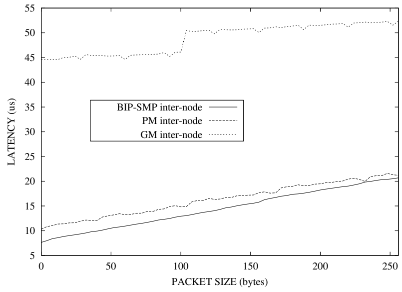

Fig. 6. Inter-node MPI.

below the main steps for running parallel programs on an operational Myrinet cluster: linking the libraries and submitting jobs. Then the `Web-CM` tool is presented.

7.1 Linking the libraries

Depending on the choice of the end-user, the program may only need the `bip` library, or also the `bipsmp` library if multi-processor support is needed. If MPI is used then the `mpi` library must be added. All the required low-level libraries are automatically included with the `bipcc` command but the user must include the `mpi` library in its `makefile` file. `bipcc` internally calls `gcc` by default but any other compiler can be used (such as g++) with the `-comp` flag.

7.2 Submitting jobs and monitoring nodes

The BIP software comes with a few Perl scripts such as `bipconf`, `myristat` and `bipload` that respectively configures a virtual parallel machine, shows the status of the Myrinet board and launches a program on a virtual machine. For the moment, the utilization of a Myrinet board is exclusive to a user. Therefore, the typical way to submit jobs was to (*i*) run `myristat` to know how many nodes, and which one, are available, (*ii*) run `bipconf` to select the available nodes (if they are in a sufficient number), (*iii*) run `bipconf` to build the virtual machine, and (*iv*) call `bipload` with the program to be executed.

7.3 `Web-CM`: the integrated Web-based Cluster Management tool

`Web-CM` is our first attempt to ease the utilization of a cluster. The main goals of `Web-CM` are to facilitate the submission of jobs and to offer a graphical view of the resources on the cluster. However `Web-CM` must be viewed as an integrated web-based environment and not as a new package for job submission or graphical visualization. `Web-CM` integrates existing packages into the web framework and interacts with them through a number of CGI-bin scripts (mainly Perl and shell scripts).

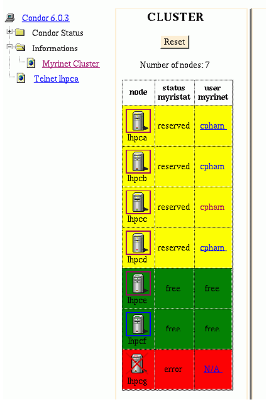

Fig. 7. Main screen.

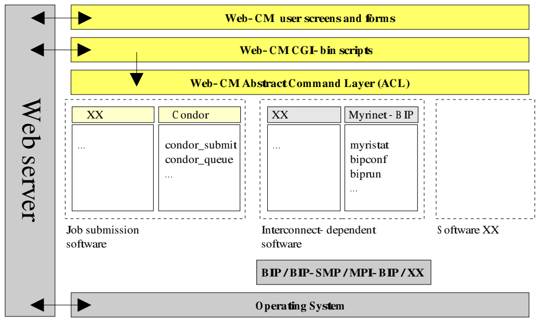

Fig. 8. Architecture of `Web-CM`.

The choice of a web-based environment makes it easy to gain access to remote clusters through a regular web page, and this in the whole Internet. For the moment, it supports Myrinet-based clusters and the `condor` job submission package (`http://www.cs.wisc.edu/condor/`) but an abstraction layer (ACL in fig. 8) makes it possible to adapt it to another type of hardware and software configurations. This ACL simply maps predefined functionalities into specific platform-dependent commands. Such predefined functionalities are for example the list of available nodes, the submission of a job in a batch queue, the status of a job... The realization of the functionality is left to an existing package.

For the moment, `Web-CM` allows a user to graphically view the available nodes and to interactively create a virtual machine by selecting the desired nodes. Then the user can run interactively the job or submit it with the `condor` package. Further implementations will allow an automatic virtual machine configuration just by indicating the number of nodes needed (mainly for queued jobs).

The modular view of `Web-CM`, and the fact that it integrates existing packages rather that developing new ones, allows for an quick increase of its functionalities. For instance, the ACL can be easily configured for another kind of interconnect network technology (SCI, Giganet...) and the new dedicated software package integrated into the common framework. We plan to add additional job submission packages since `condor` was initially chosen because of its free availability but appears to be too complex.

8 Conclusions

HPCs represent a serious alternative to expensive parallel computers. In this paper, we mainly focus on a Myrinet-based cluster and have presented a software suite exploiting on a daily basis such a cluster. The software suite is used intensively in our group for several projects involving genomic simulations and parallel simulations of large scale communication networks. BIP is used worldwide by several universities.

One of the principal aims of this paper is to show the maturity of the more recent technologies. Generalizing the use of the faster systems available will provide to the end-user community a way to reach a higher level of scalability, and to make parallel solutions usable for a wider range of applications, especially those that require fine grain decomposition.

Acknowledgements

The initial version of `Web-CM` was developed by S. Oranger and F. Goffinet (as part of their undergraduate work) and, L. Lefèvre and C. Pham.

References

1. Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW Team. A case for networks of workstations: Now. *IEEE Micro*, Feb 1995.
2. Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet - a gigabit-per-second local-area network. In *IEEE Micro*, volume 15.
3. Loïc Prylli and Bernard Tourancheau. BIP: a new protocol designed for high performance networking on myrinet. In *Workshop PC-NOW, IPPS/SPDP98.*
4. L. Prylli, B. Tourancheau, R. Westrelin, An Improved NIC Program for High-Performance MPI, *in : International Conference on Supercomputing (ICS'99)*, N. P. Carter, S. S. Lumetta (éditeurs), *Workshop on Cluster-based Computing.*
5. P. Geoffray, L. Prylli, B. Tourancheau:, BIP-SMP: High Performance message passing over a cluster of commodity SMPs. *in : Supercomputing'99 (SC99).*
6. L. Prylli, B. Tourancheau, and R. Westrelin. The design for a high performance mpi implementation on the myrinet network. In *EuroPVM/MPI'99*, 1999.
7. 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, September 1996.
8. T. von Eicken. *Active Messages: an Efficient Communication Architecture for Multiprocessors.* PhD thesis, University of California at Berkeley, November 1993.
9. Pakin, S., Karamcheti, V., Chien, A,: Fast messages (FM): Efficient, portable communication for workstation clusters and massively-parallel processors. *IEEE Concurrency*, 1997.
10. von Eicken, T., Basu, A., Welsh, M.: Incorporating memory management into user-level network interfaces. TR CS Dept., Cornell University, 1997.
11. Parry Husbands and James C. Hoe. Mpi-start: Delivering network performance to numerical applications. In *SuperComputing (SC'98)*, Orlando, USA.
12. Steven S. Lumetta, Alan M. Mainwaring, and David E. Culler. Multi-protocol active messages on a cluster of smp's. In *SuperComputing (SC'97).*
13. Ralph M. Butler and Ewing L. Lusk. Monitors, messages, and clusters : the p4 parallel programming system. TR, University of North Florida and ANL, 1993.
14. Steven S. Lumetta and David E. Culler. Managing concurrent access for shared memory active messages. In *International Parallel Processing Symposium*, 1998.