High performance communication libraries for Windows 2000 : from a developer standpoint *

Laurent Lefèvre and Roland Westrelin
RESAM laboratory - INRIA Action RESO
Ecole Normale Supérieure de Lyon
46, allée d'Italie, 69364 LYON Cedex 07, France
Laurent.Lefevre@inria.fr, Roland.Westrelin@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. We developed a software suite for achieving high-performance communications on a Myrinet-based cluster: BIP and MPI-BIP. This software suite originally runs under Linux. In this paper, we present how we ported this layers to Windows 2000 and solved the system administration problems we met.*

Keywords: Cluster computing, Myrinet, Communication software, MPI, Microsoft Windows 2000

1 Introduction

Beowulf clusters are seen increasingly as the future of small-to-medium parallel machines, and some projects are also investigating the use of large clusters. For fine grain applications, the use of a dedicated high-speed interconnects is crucial to the performance. Myrinet [1] is one of this technology.

Experiments show that the bottleneck in such a platform can often be the software part of the communication system. Changes in the methodologies for the design of protocol stacks with the new high-performance networks led to lightweight user-level network interfaces. BIP [2] and MPI-BIP [3] are examples of such software.

BIP was developed with and for the Linux system. This paper introduces our work to port BIP to Microsoft Windows 2000. Our goal is to present the approach we used, knowing that we have a strong Linux background.

The paper is organized as follows. In section 2, our strategy for porting quickly BIP to Windows 2000 is introduced. Section 4 describes the approach we use to make easier the system administration of our Windows cluster. Section 5 presents related works in the Windows cluster field. We finish by some conclusions based on our experience with cluster computing under Windows 2000.

2 Porting BIP and MPI-BIP to Windows

The BIP low level communication layer is composed of a set of components shown on figure 1. We describe then briefly here and introduce the strategy we used to have them working on Windows.

2.1 The kernel module

It must act as a driver and it must also provide some basic services to the BIP library. At the initialization time, it gives direct access to the Myrinet board by the BIP library. It is also used to register/unregister memory (pin down

*This work is supported by Microsoft Research and INRIA RESO Action.

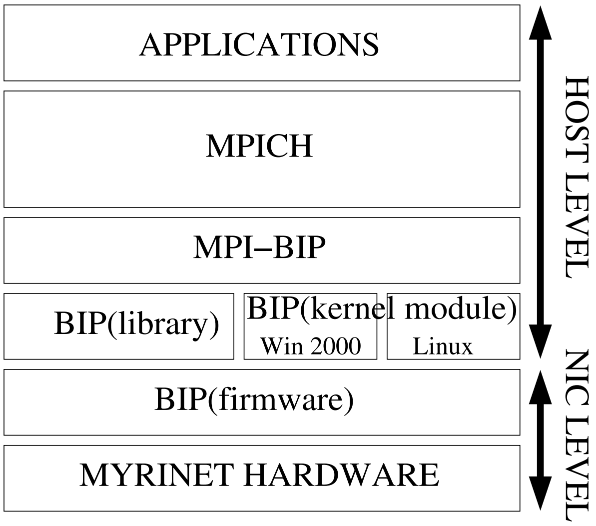

Figure 1: BIP suite architecture

memory pages in physical memory, provide the address translations).

Our idea was to rely on Myricom's GM driver. Indeed this driver already provide functionalities close to what we need. And it is available for a wide range of platforms including Windows 2000. It is designed in two layers. The upper one is generic and is where the core of the treatments are performed. It relies on a lower one which is architecture-dependent. The idea is to compose the services provided by the architecture-specific layer to suit our own needs. So we modified the GM driver so that it provides a new set of services for the BIP library.

2.2 The library

When it was written, it was targeted only to Linux. Thus, even though there is no fundamental limit that prevent a native port to the win32 system, we decided to use the cygwin [4, 5] porting layer which is freely available. Using this library has several advantages. Maintenance of the code is easy. There is only a set of source files with no ugly `#ifdef/#endif`. It comes with a full environment which includes a set of handy tools: make to manage the project, gcc to compile the code, perl to use the script provided with BIP, ssh to access the remote nodes. It is in very active development and is getting better and better at a quick pace. We see very few objections to the use of the cygwin system. It is still possible to use a third party compiler for the applications to ensure top performance. The BIP and MPI library in themselves don't use system calls for any critical tasks and the application writer has the freedom to use win32 calls directly to save the extra overhead introduced by the cygwin layer. Note that even if the cygwin library is a very powerful tool, we still had to rewrite some part of the BIP library using native win32 calls.

2.3 The firmware

Nothing to do here, it is independent of the operating system.

2.4 The runtime environment

The BIP software suite comes with a very basic runtime environment in the form of a set of perl scripts. They are used to automatically discover the routes between the nodes of the cluster (Myrinet uses source routing so it is up to each of the sending nodes to fill correctly the packets' header), set the current configuration (which nodes will be used for the next runs) and to launch the program on the nodes of the current configuration.

These scripts use extensively rsh or ssh to access the remote nodes. Hopefully, cygwin provides its own port of the openSSH [6] implementation (daemon and client). It is even possible to log without a password with public/private key authentication.

2.5 MPI-BIP for Win2000

Since MPI-BIP is a higher level layer, it is less dependent on the underlying operating system and hardware. Porting of MPI on top of BIP was relatively easy and allows us to experiment and gather applications results (NAS benchmarks in Table 1).

3 Experiments

The test bed is composed of 8 dual Intel PIII clocked at 933 MHz with 512 MB of main memory equipped with Myrinet 2000 boards (plugged on a 64bit/66MHz PCI bus which can sustain the 250 MB/s throughput of the physical links). We give the results for both BIP and MPI-BIP running under Linux and Windows. We used GNU compilers in both cases.

Figures 2 and 3 present the results of a ping-pong test in terms of latency and bandwidth. Table 1 gives the results of two of the NAS parallel benchmarks [7]. These are MPI benchmarks. Class A and B designate two problem sizes [1].

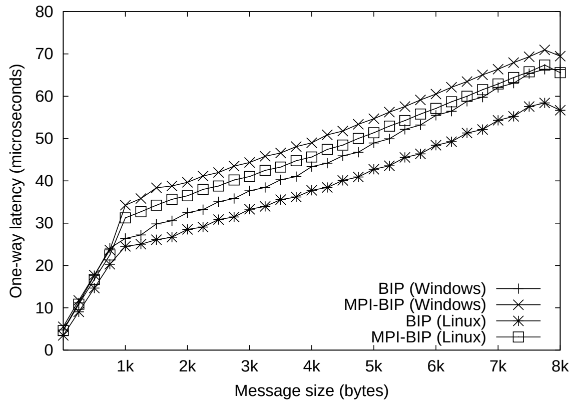

Figure 2: Latencies for a ping-pong test

Both for the point to point micro benchmarks and for the NAS parallel applications, the performance of BIP under Windows is slightly worse the performance of BIP under Linux. This could be explained by the fact that we are using a heavier design under Windows with the cygwin layer and the modified

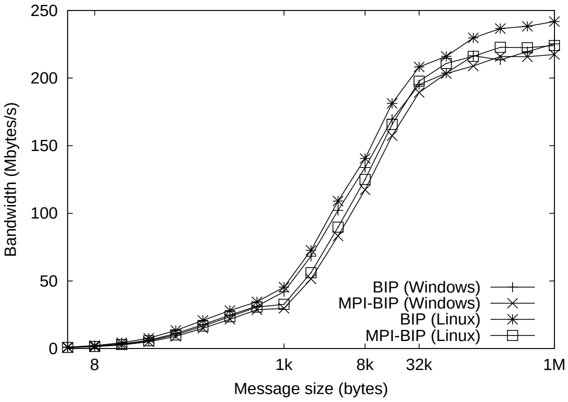

Figure 3: Bandwidths for a ping-pong test

GM driver. This could also be explained by the fact that some calls to a kernel module are more expensive under Windows than it is under Linux.

IS is a C benchmark which is quite communication intensive. The LU benchmark is a Fortran program. Its performance is poorer under Windows even for the sequential case. This tends to prove that the Fortran compiler currently shipped with cygwin generates code less efficient than the one we used under Linux. We believe that this has nothing to do with Windows.

4 Win2000 System Administration : cluster deployment

We believe that the tools provided by Microsoft to ease administration tasks don't suit our needs in a development environment. So we decided to use a simple administration strategy which seems to be widely used in the Windows cluster world: to have a model machine and to perform a raw copy, byte by byte, of its Windows partitions trough the network.

This is the brute-force method. We measured the performance of our SCSI disks and found out that they offer bandwidths in the range of 30 to 50 MB/s. So it appears that copying partitions thought fast Ethernet (12

[1]Note that at the time of writing, we have not been able to run the LU benchmark, in class B, on a single processor under Windows. We are still investigating the problem.

	Sequential		4 processes		8 processes		16 processes	
	Win.	Linux	Win.	Linux	Win.	Linux	Win.	Linux
IS (class A)	9.47	9.46	2.66 (3.6)	2.52 (3.8)	1.53 (6.2)	1.46 (6.5)	1.31 (7.2)	1.27 (7.4)
IS (class B)	38.02	38	10.70 (3.6)	10.31 (3.7)	6.05 (6.3)	5.94 (6.4)	5.34 (7.1)	5.22 (7.3)
LU (class A)	1597	1230	398 (4)	309 (4)	201 (7.9)	156 (7.9)	196 (8.1)	138 (8.9)
LU (class B)		5646	1647	1419 (4)	862	674 (8.4)	536	479 (11.8)

Table 1: Results with the NAS parallel benchmarks: execution time in seconds and speedups (in parenthesis).

MB/s peak performance) would not allow us to do the replication of disks at full speed. Thus we used Myrinet, which offers 250MB/s peak bandwidth. So if done "perfectly", replicating our 5GB Windows partition to all the nodes should take the time to write or read 5GB of raw data from one disk: roughly between two and tree minutes. If this can be achieved, propagation of a master image to all the cluster nodes becomes an almost free (in time) operation which can be perform as often as changes in configuration justify it.

We implemented our own tool to replicate the partitions over Myrinet. Replication is performed from a minimal Linux installation. Propagation of the partition is performed in a pipelined manner. The partitions are split in chunks of 1MB. Data are transfered over TCP/IP over the GM/Myrinet driver. After the replication, all the nodes are strictly identical. They are configured to acquire their IP settings through DHCP so only the name of the machine must be changed. One solution is to reboot under Windows, have a script modify the registry, and reboot one more time. We did not find it practical to set up. So we decided to switch to a more aggressive strategy and directly edit the registry from Linux. This implies that the core Windows 2000 system must be installed on a FAT partition because Linux can only write to FAT and pre-Windows 2000 NTFS partitions. By reusing code from the Wine [2] [8] project we were able to find our way in the Windows 2000 registry.

[2]The goal of this project is to run Windows applications under UNIX.

Indeed, copying 5GB of data (one 2GB partition and one 3GB partition) from one master node to seven slave nodes does only take 2 minutes 45 (including the time to edit the registry but not to reboot the machine under Windows). When the data are sent over Ethernet, the time of the replication is 8 minutes 40.

It is interesting to look at how this kind of mechanism would scale to larger clusters. Using this pipeline strategy to replicate a 5GB partition to 100 machines would only increase the time by 2% (compared to a local copy of the partition). For 1000 machines, it would be 20%, for 10000 machines, 3 times longer and for 100000, 21 times longer. An alternate and more scalable strategy is to rely on a tree for the broadcast of the data chunk to all nodes and to use directly GM instead of TCP/IP/GM.

5 Related works

Compared to Linux in Cluster computing solutions, less research has been conducted in terms of high performance computing with Windows Operating Systems. Academic researchers like the ones from the Concurrent Systems Architecture Group at UCSD and the National Center for Supercomputing Applications (NCSA) [9] and university of Southhampton [10] built large Windows supercomputing cluster experiments. From an industrial point of view, the Cornell Theory Center [11] and Entropia [12] provides windows clusters solutions. Our research is more linked with [13] in order to compare Linux and Windows performance and solutions for high performance cluster computing.

6 Conclusions and future works

In this paper we present how we ported the BIP software suite from Linux to Windows 2000. The cygwin software is a very efficient tools to quickly port a library from UNIX to Windows. But emulating UNIX functionalities with the cygwin layer can have a high cost. However, in our case since BIP avoids system call in the critical path of the communications, it does not have a great impact. The experiments show that even if the performance are slightly worse than under Linux, our Windows cluster remains efficient. Cygwin also provides a set of UNIX tools. The Windows cluster then looks a lot like a UNIX cluster. Our first experiments demonstrate comparable results between Windows 2000 and LINUX in terms of latency and bandwidth. But the administration and deployment of applications and high performance communications libraries with Windows 2000 is relatively complex. To make administration easier, we use extensively a disk replication tool we developed. By sending data over the Myrinet network and by using a proper broadcast algorithm, the replication of a partition is fast enough that it can be performed whenever needed.

Our next step will be to combine OS-heterogeneous clusters, by allowing them to communicate through MPI-BIP.

Acknowledgment

We would like to thank Microsoft Research Ltd for the support during the project and Loic Prylli (LIP / CNRS) for the help he provided with the GM driver.

References

[1] Nanette J. Boden, Dany Cohen, Robert Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov Seizovic, and Wen-King Su. Myrinet: A gigabit per second local area network. *IEEE-Micro*, 15(1):29–36, February 1995.

[2] Loïc Prylli and Bernard 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. IEEE, Springer-Verlag, April 1998.

[3] Loïc Prylli, Bernard Tourancheau, and Roland Westrelin. The design for a high performance MPI implementation on the Myrinet network. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface. Proc. 6th European PVM/MPI Users' Group (EuroPVM/MPI '99)*, volume 1697 of *Lect. Notes in Comp. Science*, pages 223–230, Barcelona, Spain, September 1999. Springer Verlag.

[4] Cygwin: a UNIX environment for Windows. http://sources.redhat.com/cygwin/.

[5] Geoffrey J. Noer. Cygwin: A free Win32 porting layer for UNIX applications. In *1998 Usenix NT Symposium.* Available from http://cygwin.com/usenix-98/cygwin.html.

[6] OpenSSH: a FREE version of the SSH protocol. http://www.openssh.org.

[7] NAS parallel benchmarks. http://science.nas.nasa.gov/Software/NPB/.

[8] Wine: A free implementation of Windows for Unix. http://www.winehq.com/.

[9] CSAG : Concurrent Systems Artchitecture Group. http://www-csag.ucsd.edu.

[10] University of southampton. http://www.soton.ac.uk.

[11] Cornell Theory Center. http://www.tc.cornell.edu.

[12] Entropia : Distributed Computing. http://www.entropia.com.

[13] K. Takeda and D. J. Lancaster. Comparative Performance of a Commodity Alpha Cluster running Linux and Windows NT. In *1st IEEE Workshop on Cluster Computing (IWCC'99)*, Melbourne, Australia, Dec 1999.