A PC cluster with high speed network interconnects

R.Innocente [1,2] , M.Corbatto [1,2], S.Cozzini [1,3]
[1] *Abdus Salam ICTP, 34014 Trieste(Italy)*
[2] *SISSA, Via Beirut 2/4, 34014 Trieste(Italy)*
[3] *INFM, Sissa Trieste Unit*

August 10,2000

Abstract

Following a previous project of a lightweight cluster with 20 single processor nodes interconnected by Fast Ethernet on which the feasibility of large scientific computations on PC clusters has been explored, we present a report on the ongoing realization and evaluation of a PC cluster with multiple high speed interconnects. [1]

[1] *This work has been partially supported by a grant from Regione Friuli Venezia-Giulia.*

1 Cluster configuration

The cluster is composed of 8 dual processor computing nodes interconnected by 3 different network technologies: Fast Ethernet, Gigabit Ethernet and Myrinet. A service node is connected only to the Fast Ethernet network and performs as a dhcp/boot/file server and batch scheduler. On the computing nodes there are 2 Pentium III (Katmai) processors running at 550 Mhz.

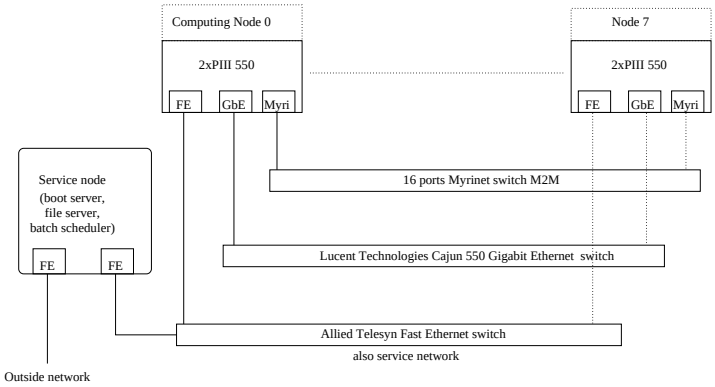

Figure 1: Network interconnections

These processors have an on chip 16 KB + 16 KB separate instruction and data level 1 cache and an off chip (but in-package) discrete 512 KB level 2 cache on a separate bus (Back Side Bus) at f/2 Mhz. The processor bus or Front Side Bus runs at 100 Mhz. On the Front Side Bus, the

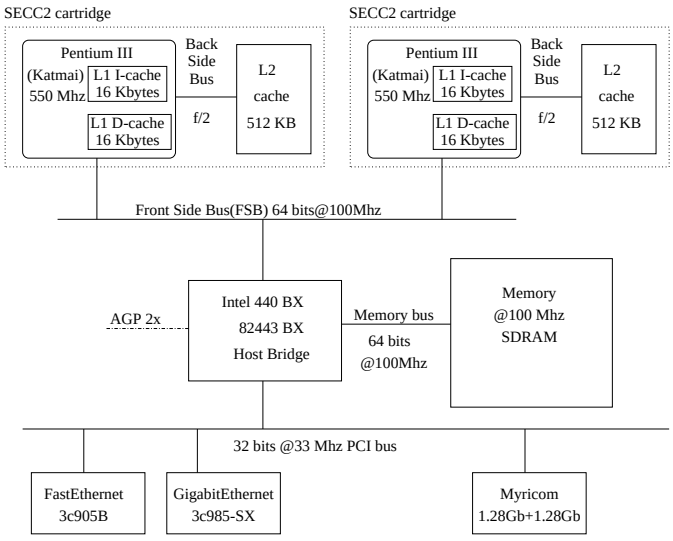

Figure 2: Computing node 2xPIII-550 + 440BX

Intel 440BX chipset manages the memory and the peripheral buses(a 32 bits @33 Mhz PCI). Memories are commodity 100 Mhz SDRAM. Each computing node has 256 MB of memory.

2 Software setup

Because of the requirement to frequently switch kernels, drivers and software setup, since the beginning we planned to reduce at a minimum the system management overhead. For this reason we installed netboot eproms on the Fast Ethernet cards and all the computing nodes

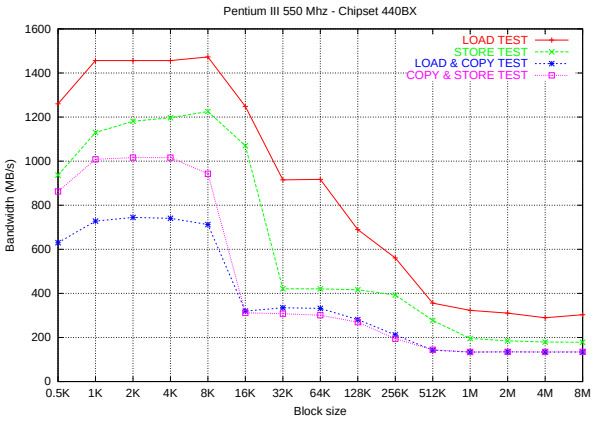

Figure 3: P3-550 Cache/Memory performance

access their root and usr partition on the service node via NFS. The nodes are now running a linux 2.2.14 smp-enabled kernel. Computing nodes have a local disk that is mainly used as a temporary scratch area and swap disk. We chose to install PBS (Portable Batch System) as our batch system. We mainly use the fortran compilers available from the Portland Group (PGI).

3 Network hardware

The service network - a Fast Ethernet - supports remote boot of the computing nodes, a common file system through NFS, remote logins, etc. For the service network we have used 3COM 3C905B cards and an Allied Telesyn switch. For this network the most strict requirement was the possibility to have netboot eproms on the cards. We also used the performance of this network in some comparisons as a frame of reference.

The aim of the cluster was the characterization of the performance of multiple high speed interconnects and software combinations. As high performance interconnects we installed Myrinet and Gigabit Ethernet.

- Gigabit Ethernet This technology promises to make available during the next years NICs working at 1 Gbit/s as commodity hardware. Nevertheless many of the devices available in the present, with prices comparable to those of other Gigabit/s technologies, have serious difficulties in obtaining results near what can be expected from the nominal performance of this technology. Two techniques have been developed to alleviate the problem : *interrupt coalescing* and *jumbo frames*. *Interrupt coalescing* refers to the possibility for the NIC to group together multiple interrupts issuing one host interrupt for multiple events. *Jumbo frames* refers to the possibility to use "big" frames up to 9000 bytes (this frames are not compatible with the Ethernet/802.3 standards and therefore could'nt be generally employed outside a restricted evnironment of switches and hosts). The cost of this solution is about 500 US$ for each NIC plus at least 7000 US$ for a small switch. As a representative of this technology we chose the 3com 3c985 cards. These cards are made out of an ASIC chip with 2 on board 32 bits 88 Mhz RISCs and 2 independent DMA engines. They have 1 MByte of memory on board. These cards support receive hardware TCP/UDP/IP checksumming, jumbo frames and receive/transmit interrupts coalescing. The cards can also support ethernet flow control in hardware.

- Myrinet It's a proprietary net that evolved from USC/ISI ATOMIC LAN. At the moment the speed is 1.28 Gigabit/s. Crossbar switches up to 16 gates are available; for an higher

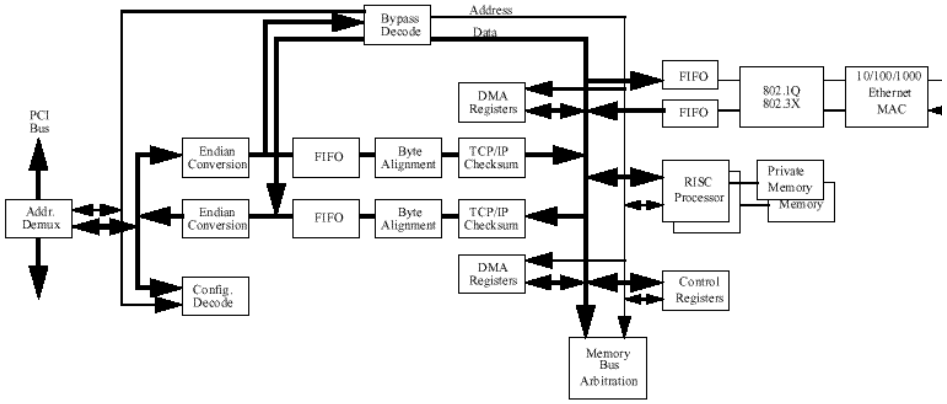

Figure 4: 3com 3c985 Gigabit Ethernet NIC

number of nodes multi-stage nets can be used. The net card is equipped with a RISC chip that can be programmed to reduce the central unit (processor and memory) load. The specifications of the card and of the RISC chip are available as well as various software like optimized drivers and communication libraries (GM, mpich over myrinet). The actual cost is about 1000 US$ per card plus 4000 US$ for a 16 port switch.

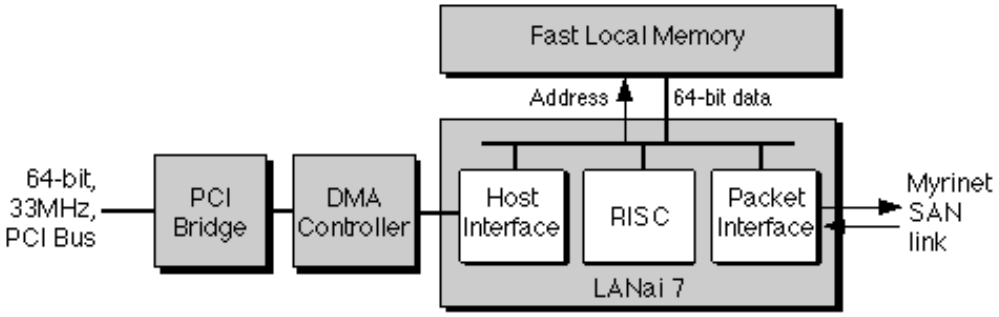

Figure 5: Myrinet M2M NIC

4 Protocols

The quality of communication software has a great weight in the overall performance of a cluster. Various protocols are available:

- Link Layer We tried to use directly the link layer. The canonical way to accomplish this on recent Linux kernels is using the socket API specifying the PF_PACKET protocol family. Thus we implemented a small library to perform a reliable ordered delivery of packets. In this way we were able to go under 100 usec of latency for all the network cards on an SMP-enabled kernel, but we did'nt reach the latency of the user-level efforts. This interface requires a system call for each packet . We are currently investigating the possibility to include fragmentation/defragmentation and to improve the performance of the device driver.

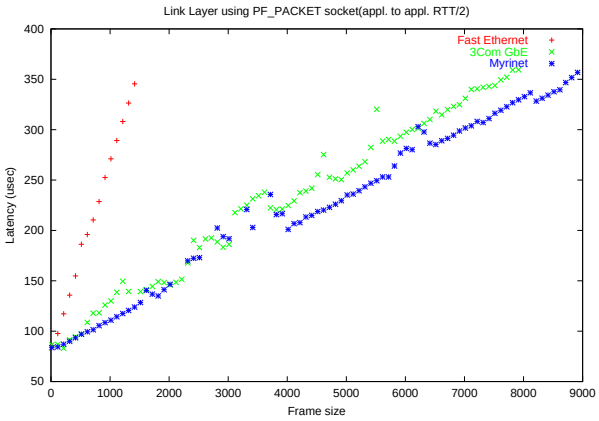

Figure 6: Link layer latency (usec)

- VIA The Virtual Interface Architecture is a Compaq, Intel and Microsoft joint proposal for a standard architecture for communications within clusters in an SAN (System Area Network) environment. This proposal tries to exploit ideas emerged in the user-level networking research efforts. In this architecture the kernel is not involved in data transfers that occur directly between the interface and user processes, but only to assist with protection and sharing mechanism. We have carried out some experiments with the prototype M-VIA (Modular-VIA) from LBL. We will try to develop a VIA kernel agent for one of the Gigabit/s NICs(Network Interface Card).

- TCP/IP This general purpose protocol have characteristics that usually make it not suitable for high performance System Area Networks. The main problem is due to the intervention of the kernel in all the network operations. (in-kernel networking). In particular it has been found that the standard implementation over Unix which realizes a copy of the data from user memory space to a kernel buffer penalizes a lot the performance when the net throughput has the same order of magnitude of memory bandwidth. This represents an unsurmountable limit to the decrease of the latency time too. To get over this bottleneck, several systems has been developed where the net card achieve read/write operations from the net to a memory space directly accessible from the user process (user-level networking). It is difficult to optimize the protocol for the SAN and at the same time for the WAN environments. We have produced a kernel patch that allows the dynamic activation and deactivation of the delayed ack algorithm. We are investigating different possible implementations of the slow start, congestion avoidance, fast retransmit and fast recovery algorithms in such a way to obtain a fair behaviour in the SAN environment.

- GM GM is a lightweight communication system for Myrinet that supports reliable ordered delivery and protected access, provided directly by Myricom. It requires the use of DMAable memory (registered with the system or allocated through the system). It supports messages up to 2^{31}-1 bytes if the OS allows such amount of DMAable memory. It has 2 levels of message priority to help avoiding deadlocks in an efficient way . In the GM programming model a reliable connection is established between hosts, while communication endpoints do not need any communication establishment to communicate(connectionless). It supports up to 10000 nodes. Sends and receives are regulated by implicit tokens representing space allocated to the client by the system in its queues.

- BIP BIP (Basic Interface for Parallelism) is a thin software layer aimed at providing a basic interface for message passing that can achieve near the maximum performance. For this

LATENCY TIME (us)	SMP Kernel	UP kernel
TCP loop-back	53	29
TCP 100Mb direct link	98	53
TCP 100Mb switched	105	61
TCP 1Gb direct link	108	70
TCP over Myrinet	103	63
GM over Myrinet	15	15
BIP over Myrinet	5	n.a.
MPICH/TCP over 100Mb	172	n.a.
MPICH/TCP over 1 Gb	175	n.a.
MPICH/GM over Myrinet	15	n.a.

reason it implements all communications in a user level library with zero-memory copies and direct access to the hardware without system calls. It has been implemented for the Linux/x86 platform with Myrinet. Short and long messages are managed differently. Short messages (less than a configurable parameter normally between 100 and 400 bytes) are stored on a queue even if no receive has been posted on the receving node. Long messages require the receive to be posted before or in a very short time (20 msec).

5 Base Performance

The tests reported have been performed mainly using four bi-processor nodes. We used BIP 0.99, GM 1.2, MPICH 1.2.0 both over TCP and over GM. During the tests a considerable difference in the TCP latency time has been observed enabling/disabling the SMP capability of the Linux kernel: this is due to the overhead in locking/unlocking of the internal structures to assure mutually exclusive access in case of multiple CPU. The reported latency time over Myrinet is comprehensive of the switch delay while in the case of fast ethernet both values were measured.

The bandwidth obtained using directly TCP and GM are here summarized:

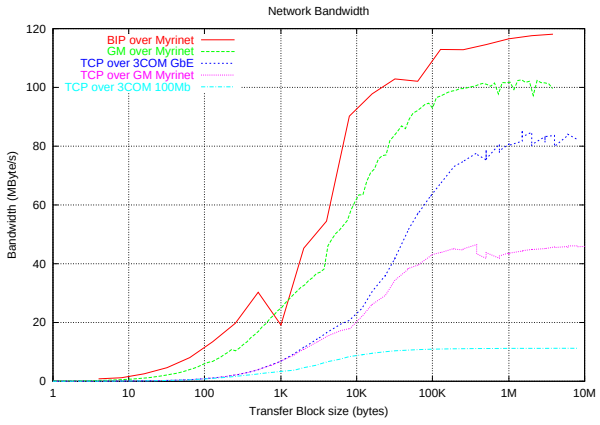

Figure 7: Best Bandwidths on Beowulf : BIP/GM/TCP

At the MPI level it's evident the difference of performance when communication take place between two process in the same node.

Another performance index that is frequently cited is the Scalapack performance. This

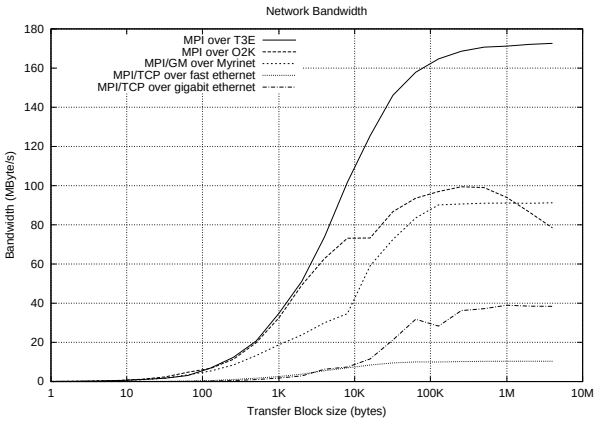

Figure 8: MPI bandwidth

library is the parallel counterpart of LAPACK. We compiled our version with the BLAS library produced running the ATLAS package, in this way we obtained a performance of 2.063 Gflops/s on 4 nodes (8 processors).

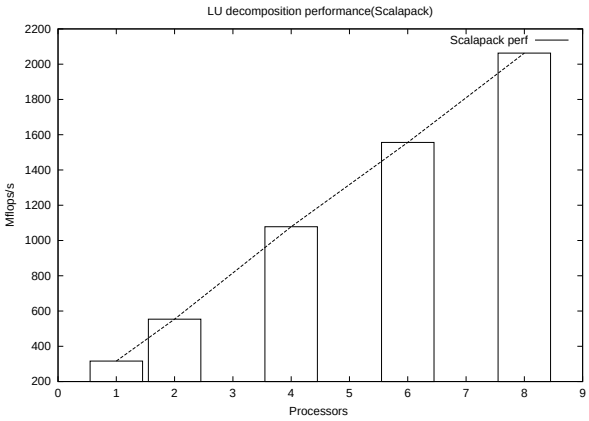

Figure 9: LU decomposition performance

6 Compared Performance on scientific applications

In this final section we present some benchmarking results obtained on some parallel scientific applications routinely used at Sissa. Performances are compared against parallel machines (T3E and Origin 2000) that are in this moment our production platforms.

There are three main classes of codes used at Sissa: Ab-initio codes, classical molecular dynamics and Quantum Montecarlo Code. We can compare these three families looking at specific aspects of high performance computing (HPC). Table 1 presents such a comparison. The three columns represents the three classes of codes we discussed above while each row indicates a peculiar characteristic of the HPC. An adjective defines the behavior of each class on that specific subject. From the table emerges clearly that the computational requirements of the three classes are quite different.

We report in table 2 a short overview of the performances obtained by all the codes on the different machines; for each code we define two benchmark values: the first, named Scalar Benchmark (SB) is given by summing times over all the serial runs. The second, named Parallel

aspect	Ab-initio	Classical MD	Quantum MC
`Communications`	high	high	negligible
`Memory requirements`	high	moderate	moderate
`Parallelization strategy`	MPI	MPI/Shared Memory	MPI
`Scalability`	high	not scalable	linear
`Use of Linear Algebra kernel`	high	almost null	moderate
`Accessing Cache`	good	bad	moderate

Table 1: Comparison between codes on different aspects of HPC.

Benchmark (PB), is similar but the summed value refers to 8 processors runs.

code	SCALAR (SB)			PARALLEL (PB)		
	T3E	O2K	Beowulf	T3E	O2K	Beowulf
PWSCF(ab-initio)	1094	775	1485	663	-	1325
FPMD(ab-initio)	56	42.1	119	6.5	5.4	17.4
DLPROTEIN_2.0(MD)	4249	1781	2831	801	412	693
AMBER-5.0(MD)	-	168	309	-	61	99
QMC	67	-	71	537	-	569

Table 2: scalar and parallel benchmark values for all the codes. See text for definition.

These benchmark values help to draw some observations about the overall behavior of different classes of codes on the parallel platform we tested.

We consider first ab-initio codes; for this group the T3E machine is the actual production machine and we look at performances of the Beowulf machine with respect to this one. Scalar performances between T3E and Beowulf cluster ($SB_{beowulf}/SB_{t3e} = 1.35 - 2.12$) are better than the parallel ones ($SB_{beowulf}/SB_{t3e} = 1.99 - 2.67$). We can not test at this moment parallel performances on larger configurations (16-32 nodes) that could allow to run real size systems but it seems that the Beowulf network will be unable to cope with large number of nodes. Therefore it could be rather difficult to run real simulation with the network technology at our disposal in this moment. It has noted however that increasing the scalar performances by means of more recent INTEL CPU, could reduce the need of large number of processors allowing to run at least medium size system in a range where the network behavior is still acceptable. This makes the Beowulf cluster a valid support platform where to run small/medium size applications.

Concerning Classical MD codes the target machine is Origin-2000: the ratio in performances between Beowulf cluster and Origin-2000 is 1.65 -1.83 and maintains almost the same values also for the parallel case; this is a good result that makes the Beowulf machine a interesting computational resource for this kind of code with a excellent performance/price ratio.

Scalar and parallel performances obtained with the QMC code are the same due to the embarrassing parallelism of the code itself. The overall performances obtained confirm that this family of codes is perfect to be run on a Beowulf cluster.

References

[1] R. Innocente, Proceedings of the Workshop PC-NETS 1999, MPI Performance of a PC Cluster at the ICTP, INFN-LNGS 1999 *(http://www.ictp.trieste.it/parallel/performance/)*

[2] S.Cozzini, R.Innocente, M.Corbatto, 6th European SGI/Cray MPP Workshop, Comparing scientific codes on different parallel platforms