Reprint from

SCI Based Cluster Computing,
H. Hellwagner and A. Reinefeld, eds.,
Springer, Berlin, 1999.

Table of Contents

List of Contributors

T. Stricker
Laboratory for Computer Systems
Swiss Institute of Technology (ETH)
CH-8092 Zürich
Switzerland

Ch. Kurmann
Laboratory for Computer Systems
Swiss Institute of Technology (ETH)
CH-8092 Zürich
Switzerland

1. A Comparison of two Gigabit SAN/LAN Technologies: Scalable Coherent Interface versus Myrinet

Christian Kurmann[1], Thomas Stricker[1]

Laboratory for Computer Systems, Swiss Institute of Technology (ETH), CH-8092 Zürich, Switzerland
email: {kurmann, tomstr}@inf.ethz.ch
http://www.cs.inf.ethz.ch/CoPs

1.1 Introduction

Five years ago Cray Research announced its T3D MPP (Massively Parallel Processor) line and set high standards for Gigabit/s SAN (System Area Network) interconnects of microprocessor based MPP systems sustaining 1 Gigabit/s per link in many applications. Today the communication speed is still at one Gigabit/s, but major advances in technology managed to drastically lower costs and to bring such interconnects to the mainstream market of PCI based commodity personal computers. Two products are available today: The Scalable Coherent Interface (SCI) and Myricom's Myrinet. Both networking technologies include cabling for SAN and LAN distances and adapter cards that connect to the standard I/O Bus of a high end PC. Both technologies can incorporate crossbar switches to extend point to point links into a network fabric. Myrinet links are strictly point to point while SCI links can be rings of multiple nodes that are possibly connected to a switch for expansion.

What is needed for an evaluation and for a comparison of Gigabit interconnects is a sort of *architectural denominator*. At this time we propose to do this at three different levels: first for a simple and highly optimized remote load/store operation using all the knowledge about the hardware details (DIRECT DEPOSIT), second for an optimized standard message passing library (MPI/PVM) and third for a connection oriented LAN networking protocol (TCP/IP). Specifically in Clusters of PC's (CoPs) we think that an increased performance for TCP/IP LAN emulation can make CoPs a very attractive platforms for many existing PC customers, because several protocol stacks, middleware packages and applications using TCP/IP sockets are readily available. In most cases it would not be economical to adapt this software to each kind of network interface. At a later date we plan to extend the study to higher-level services which might use such UDP/IP or TCP/IP protocols e.g. to remote file systems or high performance web- and database-servers.

Most previous performance studies restrict themselves to the presentation of maximum transfer bandwidth and the minimal latency or simply address the performance of a single application. Unfortunately such studies are still

the state of the art in comparing different network technologies. We think this is inadequate, since scientific application codes for parallel computers or clusters of workstations often deal with large quantities of distributed data for communication. Some applications use the regular layout of data and store their data in distributed arrays, while other applications use a fine grain object store as their distributed data structures. In both cases we encounter a few characteristic communication patterns when arrays or object collections are redistributed or distributed objects are migrated. The regular array transpose is a good generic test case to assess the strengths and weaknesses of the communication- and memory- systems. We therefore extend our benchmarks to cover data types beyond contiguous blocks and incorporate the processing of strided transfers as an example of complicated memory access patterns to our tests.

Our paper is organized as follows: In Chapter 1.2 we describe the three levels at which we compare the performance of the networks. After that we attempt to describe the SCI- and the Myrinet-network interface technologies in uniform architectural terms including a few references to the old T3D technology in Chapter 1.3. Chapter 1.4 explains the options of different transfer modes for each of the three network interfaces. The performance measurements for communication throughput in Chapter 1.5 permit to conclude about the performance of our current SCI and Myrinet evaluation boards in Chapter 1.6.

1.2 Levels of comparison

The key functions of a communication system within a distributed system is to move data and to provide explicit synchronization for consistency. This can be done at different abstraction levels with more or less support by the system. We attempt to find a common denominator for an evaluation and the comparison of different Gigabit interconnects by selecting a few common operations and examine the ways those can be performed with different abstractions and different amount of support from system software. At the lower levels the performance results are highly transparent and we can easily relate them to the specifications of the hardware while at the higher level the performance figures correspond closely to what an application can expect from the system. Therefore we propose to do the comparison at three different levels:

- **DIRECT DEPOSIT:** for simple remote load/store operations. The performance at this level is expected to be closest to the actual hardware performance.
- **MPI/PVM:** for an optimized standard message passing library. Carefully coded parallel applications are expected to see the performance measured at this level.

– **TCP/IP:** for a connection oriented TCP/IP LAN emulation protocol. Users that substitute a Gigabit network for a conventional LAN will see a performance comparable to this benchmark.

Direct Deposit with its simple "no fuzz" remote store semantics allows us to evaluate the fraction of the hardware performance that is sustainable by software in the different technologies. The transfer of non-contiguous blocks exposes the capability of the hardware to handle fine grained communication. If the hardware is unable to execute fine grained transfers efficiently, aggregating copies must be used.

At a higher level we further explore the transfer modes of MPI/PVM message passing communication with full buffering according to clean postal semantics and alternatively with some common "zero copy" shortcuts that typically restrict the semantics of the messaging API. To investigate the performance delivered in a classic, connection oriented networking protocol we picked the "TCP/IP over LAN" protocol emulation. In the latter two cases the transfer of contiguous blocks satisfies the needs of the data representations in the principle API, but additional copies may occur due to the tricky postal semantics of send and receive calls or due to the requirement for retransmission to achieve reliable connections.

1.2.1 Direct Deposit

Conventional message passing programs use the same mechanism (messages) for control and data transfers. We classify control and data messages based on their contents. Control messages are linked to synchronization and are often part of the protocol. Data messages contain a significant amount of data that is moved between nodes and benefit for better communication bandwidth. Control messages are mostly empty and low latency is all that counts for handling them efficiently. Most high performance interconnects permit low latency signaling for control messages. The corresponding issues of their implementation and optimization is beyond the scope of our evaluation in this paper. The major concern is with the data transfers. Typically the source of data is in the virtual memory of a user process on the sender node, and the destination is in the user's memory at the receiver.

In the deposit model only the sender actively participates in the data transfer, "dropping" the data directly into the address space of the receiver, without active participation of the receiver process. The model allows a clean separation of control and data messages. In the deposit model, control messages, barriers or semaphores are used to deal with explicit synchronization, and data messages are sent only when the receiver has signaled its willingness to accept them.

In addition to transferring contiguous blocks the deposit model allows to copy fine grained data directly out of the users data structure at the sender into the users data structure at the receiver, involving complicated access

patterns like indexing or strides. In most applications we encounter a few very characteristic communication patterns when arrays are redistributed. For transposes of distributed arrays and many other redistributions, every processor must exchange data with every other processor i.e. perform and "all-to-all personalized communication" (AAPC) operation. The end-to-end performance of an array transposes is largely determined by the ability of the memory and communication system to collaborate and handle local and remote copy transfers with strides (dense matrices) or indices (sparse matrices) optimally. As we will see in our evaluation different communication technologies have a different amount of hardware support for such transfers. This fact is exposed by our study of deposit transfers.

The deposit transfers can be implemented in software e.g. on top of an active message layer, where the sender node just sends the data, and a handler is invoked on the receiver to move the data to its final destination. However, our understanding of direct deposit suggests that a general control transfer in the form of an RPC should be avoided and that previously asserted synchronization is sufficient to move the data. Furthermore the functionality of the handler is fixed and the deposit operation at the receiver only affects the memory system of the receiver. Unlike in the original active messages a good and efficient implementation of direct deposit therefore mandates that the deposit handler is implemented directly in hardware, e.g. by DMAs or alternatively by a second "communication" co-processor, which executes only handlers and unpacks messages.

Direct deposit strongly resembles a simple remote store on a NUMA architecture. The difference between deposit and NUMA stores is that the deposit model promotes and assumes aggregation despite the non- contiguous access patterns that occur when communication data is placed directly to its destination in user space.

1.2.2 MPI/PVM

MPI and PVM are examples of the classical postal model. Both the sender and receiver participate in a message exchange. The sender performs a send operation and the receiver issues a receive operation. These operations can be invoked in either order, blocking or non-blocking. That is, messages can be sent at any time without waiting for the receiver. This enhancement in functionality forces the system to buffer the data until the receiver accepts it. An optimization, so that no additional copy is needed on the receiver node, delays the transfer and waits until the receive is posted which then pulls the data from the sender. Figure 1.1 shows two possible scenarios of postal semantics. The upper chart shows how data can be transferred directly when restrictions on the use of the send and receive calls are accepted. In the picture the restricted calls are synchronous and mutually block to wait for completion at both ends of the transfer. In the lower chart full postal semantics is assumed and required. The messages can be sent without looking at the receiver. The

messages will be buffered by the communication system until the receiver is eventually ready to accept them.

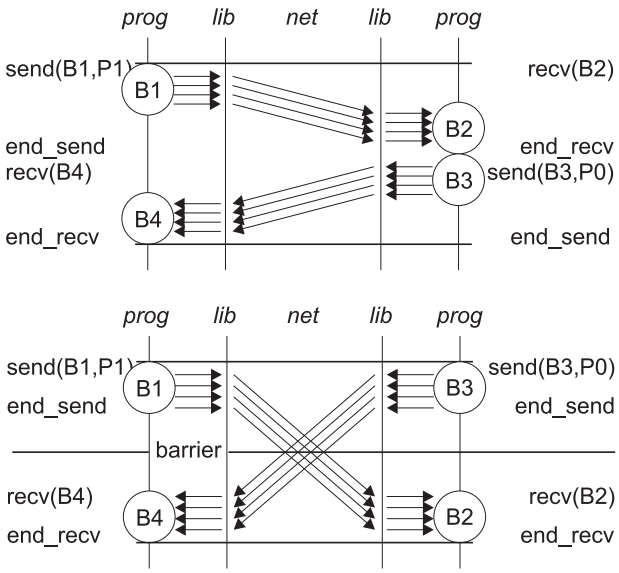

Fig. 1.1. The figure shows two scenarios: First with restricted postal semantics based on synchronous, blocking send calls, which may release the sender only when the receive is executed; Second with full postal semantics where non-blocking calls are always permitted, forcing the communication system to buffer the messages until received.

MPI- and PVM-communication functions must be implemented with the lower level primitives that are offered directly by the communication technology. Often this primitives are just remote load or remote store. For clean higher level abstractions buffering is an important part of the message passing system. The typical amount of data transferred is usually too large to be stored in special purpose registers of the network interface or in the buffers along the path from a sender to the receiver. Therefore buffering is done by a higher level of the message passing library and involves the memory system at the end-points. Many simple libraries just allow to map the implementation of blocking sends directly to a fast direct deposit including synchronization functionality. Those can be compared with the restricted semantics scenario. The proper execution of non-blocking sends need buffering on the receiver side and often leads to an additional copy operation which again largely affects the performance. This mode of operation is characterized by the second message passing scenario. In a serious performance characterization of any MPI/PVM implementation both cases should be evaluated separately.

1.2.3 TCP/IP

Connection oriented LAN network protocols are particularly important for Clusters of PCs and crucial to their commercial viability. Most protocol stacks are provided by the default operating system of the PCs and many software packages using these protocols and the socket interface are available. In our Clusters of PCs (CoPs) project we do not want to limit ourselves to deploying ever cheaper GigaFlops to our research colleagues in computational chemistry or computational biology, but intend to widen the range of parallelizable applications from scientific codes to databases and internet servers. Especially for web-servers or commercial databases and middleware-systems it would not be viable to change the standard communication protocols to restricted high speed messaging. For network file systems on clusters, like NFS or Sprite, both UDP/IP and TCP/IP must be provided. With an optimized IP implementation a Cluster of PCs can provide high performance at a good price for a larger number of programs, than a dedicated cluster with just a message passing system software does.

IP is primarily designed for internet communication and not for messaging in parallel systems. However it can offer an unreliable, connectionless network service fragmenting packets in IP-datagram and deliver them according to the IP address scheme. Transport protocols as UDP and TCP allow to extend communication to different processes of the same end system by a port concept (sockets). TCP further enables full duplex communication over a reliable datastream by implementing flow control and retransmission with a sliding windows protocol. The latter functions of TCP are less important in a cluster interconnect (there should be no loss in the switches) but its API is very common if not ubiquitous.

Because the protocol is implemented without specific knowledge of the used hardware, assuming an unreliable network service like Ethernet or Internet respectively, the performance of IP will rarely match the performance of optimized MPI and direct deposit protocols. Especially the latency for TCP data transfers is much higher due to connection setup, which might be acceptable or unacceptable to certain applications. The maximal bandwidth is lowered by at least a factor of 2-3 by the requirement for reliable transfers with retransmission after communication errors.

1.3 Gigabit Network Technologies

For many years, the rapid advances in processor technology and architecture have consistently outperformed the improvements of interconnect technology in terms of speed, cost and widespread availability. However, recently new interconnect technologies in local-area and system-area networks have been adopted and implemented by interconnect and networking vendors in the form of switches, links and I/O-bus adapter interfaces for workstations and

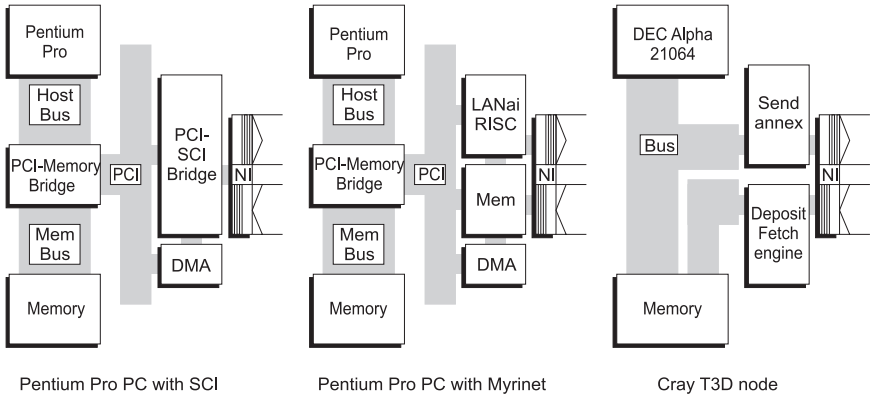

Fig. 1.2. Block diagrams of different network interface architectures. A Pentium Pro PC with either Dolphin SCI Interconnect or Myricom Myrinet PCI adapter and a Cray T3D node with DEC Alpha Processor and deposit-fetch engine.

PC's. They promise to close the gap between the advances in processor and interconnect technology and support the development of new, more powerful, and more effective parallel processing systems.

Two of the most promising new networking technologies for interconnecting compute nodes at gigabit speeds in a high-performance Cluster of PC's (CoPs) are Dolphin's Scalable Coherent Interface (SCI) and Myricom's Myrinet. Of course this list is by no means complete. There are a few old interconnect technologies that are not as actively marketed like e.g. the memory channel or there are other technologies that are newly announced or even shipping and we were just unable to evaluate them (yet). Since the mode of operation of an SCI interface connected to a PC is so similar to the hardware of a processor node in the SGI/Cray T3D system we provide a short description of its communication technology as a reference for mechanisms, services and performance. The SGI/Cray T3D is a 5 year old, non-commodity platform, that reached the end of its life cycle in the mean time, but is still faster (and still much more expensive) than any of its competitors today.

A simplified schematic of the network interface of the Dolphin PCI card, the Myrinet LANai card and the T3D network interface circuitry is shown in Figure 1.2. The main difference between the old T3D and the newer commodity interconnects for PCs is found in the indirect access to the network and the memory through an I/O bus. The "motherboard" chipset of a PC which includes the memory controller and an I/O-Bus bridge assumes the role of a main internal switching hub of the system. All I/O-operations including Gigabit networking must be performed over the PCI-bus, a standard I/O interface with lower bandwidth than the host- and memory-bus. With the advent of games and virtual reality the bandwidth requirements for graphics has outgrown the PCI bus and the PC industry reacted with a separate graphics port (AGP) for the newer PCs. Unfortunately networking seems

less important to the mass market and has not yet been accommodated with a special port by the motherboard chipsets. The SGI/Cray T3D allows direct access to the memory via a highly optimized deposit/fetch engine. In addition to that engine a so called DTB annex supports direct communication from the processors caches to the network via an address translator and a few special FIFOs. The principle difference between the two commodity systems, Myrinet and SCI lays in the large amount of staging memory and the fully functional RISC processor core provided on every Myrinet interface board versus the simple buffer-oriented network interface hardware on the SCI network interfaces.

1.3.1 The 80686 hardware platform of CoPs

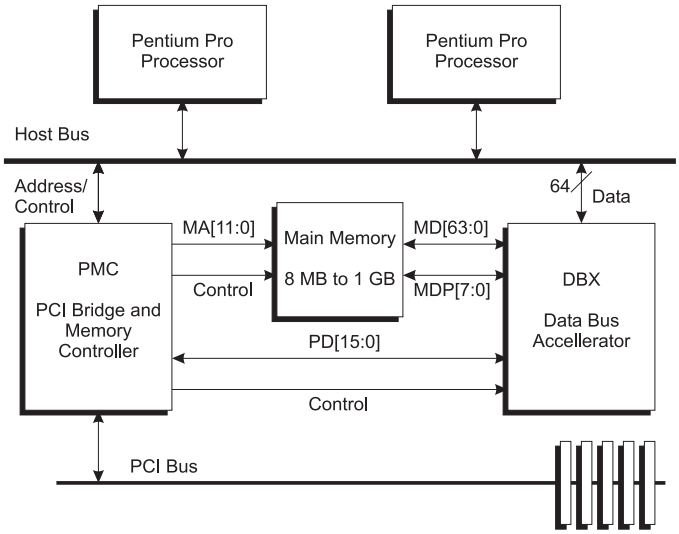

Fig. 1.3. Blockdiagram of the Intel 440FX Chipset. The PCI bridge offers a private bus for memory accesses from the PCI bus which is capable of saturating the 132 MByte/s PCI bandwidth.

Our experimental platform for benchmarking the two commodity Gigabit/s networks is given by the plans for our cluster of PCs. The current node architecture is based on single and twin 200MHz Intel Pentium Pro processors. The memory system design of this PC-Platform is based on a 2-level hierarchy of caches (L1, L2) integrated on the processor chip. The L1 cache consists of separate 8KByte for data and instructions. The instruction cache is 4-way-set-associative, the data cache is dual ported, non-blocking, 2 way-set-associative supporting one load and one store operation per cycle, the cache line sizes are 32Bytes wide. The L2 cache, located on a separate die, is a

256KByte set-associative, non-blocking unified instruction/data cache, which is closely coupled with a dedicated 64-bit full clock-speed backside bus. An Intel 440FX Motherboard Chipset gives the processor access to 64MByte of SDRAM memory and the PCI bus. The processor, the Data Bus Accelerator (DBX) and the PCI Bridge are connected by a proprietary 64-bit 66MHz host bus (512MB/s). The external PCI bus is 32bit wide and runs at 33MHz (132MB/s) (see Figure 1.3 [5]).

1.3.2 Myricom Myrinet Technology

The Myrinet technology is a SAN or LAN networking technology based on networking principles previously used in massive parallel processors (MPPs) [1]. Myrinet networks are built from links that carry a pair of full duplex 1.28 GBit/s channels that connect host and switches point-to-point. Wormhole routing with link level flow control guarantees the delivery of messages despite congestion, the checksums are just for the detection of electrical errors. The 4, 8 or 16 port switches of Myrinet may be connected among each other by links in any topology. Myrinet packets are of arbitrary length and therefore can encapsulate any type of packet (i.e. Ethernet packets, IP packets, MPI messages) and most notably - the maximum length of the packet (MTU) is not limited. In the network link level flow control guarantees integrity of the data transfers at the expense of an increased potential of mutual blocking and deadlocks in the switches.

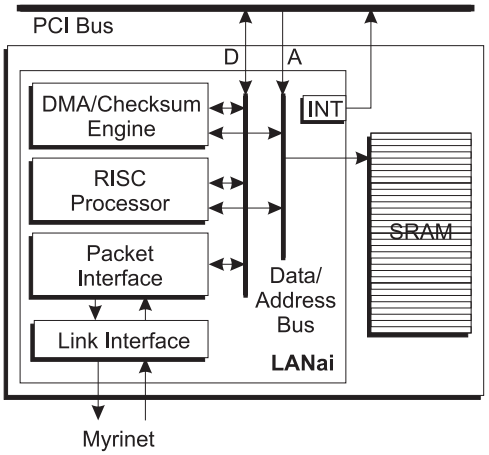

Fig. 1.4. Block diagram of the Myricom Myrinet Adapter.

A Myrinet host adapter (see Figure 1.4) contains a LANai chip with a RISC Processor core, several DMAs and the entire network interface integrated in one VLSI chip. In addition to the LANai there are 512 KByte - 1

MByte of fast SRAM on the adapter card to store a customizable Myrinet Control Program (MCP) and to act as staging memory for buffering packets. Typical MCPs provide routing table management, gather operations, checksumming, send operations, generation of control messages, receive operation, validity checking, scatter operations and interrupts for the receiving processor upon arrival of a message. The RISC processor core is a 32-bit dual-context machine with 24 general purpose registers. One of the two contexts is interruptible which causes the processor to switch to the non-interruptible context. In addition to the RISC processor core the LANai includes three DMA controllers. The LANai can act as a bus master to gather data blocks from or scatter to the host memory over the PCI bus. At the same time, the DMA engine computes an datagram checksum of the data it transfers. The remaining DMAs are used between the network FIFOs and the staging memory on the card. All DMAs can work in parallel which allows pipelined operation. They can either be initialized by the MCP on the LANai or directly through memory mapped special registers with the main processor.

1.3.3 Dolphin PCI SCI Technology

The primary goals of the Scalable Coherent Interface (SCI) technology is to provide bus functionality with point-to-point interconnects including scalable cache coherent shared memory between physical distributed processors and memory systems (IEEE Std 1596-1992 [9]). SCI supports a variety of topologies including rings and switched rings. The current versions of most PCI-SCI adapter card only implement a subset of the IEEE standard excluding the hardware cache coherency protocols. For full coherency there are a few expensive adapters available that replace one Pentium processor in multiprocessor motherboard and connect directly to the processor bus. The most simple full duplex connection can be established with two unidirectional links each allowing 1.6 GBit/s throughput.

We examined Dolphin's PCI-SCI adapters [4] (see Figure 1.5). Those adapters currently support two modes of operation, one mode for per-word, shared memory operation (by transparently forwarding requests and responses between PCI busses) and one mode for block operation in message passing (executed by DMAs). In the first case, a load/store request to the remote memory is sent the adapter card on the PCI bus instead and translated into an SCI read/write request, sent to the remote PCI bus of the receiver and executed there as a memory operation with a potential return of data to the sender (remote reads). At the receiver a Read/Write requests to the remote memory segment is mapped to, is forwarded to the memory system through the local PCI bus.

The card consists of two main parts: the protocol engine and the link controller. The sent data with its management information is stored in eight 64Byte streambuffers selected by the store address. This allows the hardware to gather contiguous data and combine it to a single SCI data packet. The

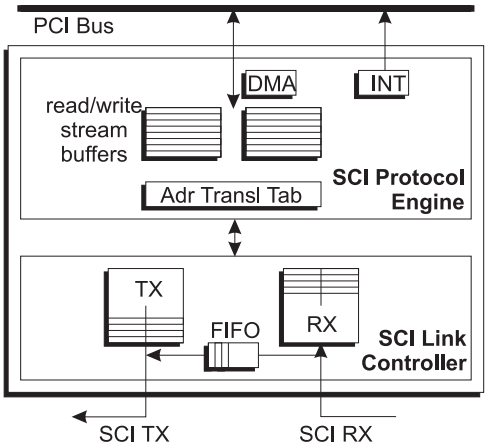

Fig. 1.5. Block diagram of the Dolphin PCI-SCI bridge.

packet is either transmitted when the streambuffer is full, by an explicit buffer flash or an implicit timeout. With this pipelined transmission, high datarates can be achieved for contiguous data. PCI- to SCI memory address mapping is also handled by the protocol engine with an Address Translation Table (ATT) where 32-Bit PCI addresses are converted to global 64-bit SCI addresses. Since SCI protocols are bus protocols, this mapping is seamless much like a PCI bridge and can be performed with low overhead. The most significant 16 bits of the SCI address are used to select between 64K distinct devices. This SCI address space is then mapped into user space.

In addition to the logic for single word/single cache line transfers, blocks can be handled by a DMA engine that moves data transparently from local memory of a sender to the receiver.

The SCI link controller chip contains two FIFOs to send and receive packets from two unidirectional Tx (transmit) and Rx (receive) channels (separate for request and data packets), which operate continuously and concurrently. It provides flow control and guaranteed delivery of SCI packets. A third FIFO stores packets in transit on the ring, when data is sent from the Tx buffer. In SCI data, addresses and SCI commands are all encapsulated into network packets with 0, 16 or 64 byte payloads. A CRC checksum is calculated for each packet so that parity errors can be reported. SCI packets arriving on the receiving link are disassembled and their contents are forwarded to the SCI controller which stores the data at the specified memory address or gets the data on a read request and sends it back.

1.3.4 The SGI/Cray T3D - a reference point

The SGI/Cray T3D node is an interesting example of an old style multiprocessor node architecture specifically designed for a distributed and parallel

system. Although the original design is already retired at this time of emerging PC clusters, it still sets the standards for a good communication interface. The implementation is done in a heat producing, expensive bipolar ECL gate array technology that does not impose any compromises for cost or for standardization. There was no commercial pressure to use a PCI bus between the processor an network interface and Cray even built its own chip foundry to achieve a shorter turn around on the gate arrays used in the network interface.

The processor board comprises a 150MHz 64bit DEC Alpha EV-4 microprocessor (21064), a local memory system, a memory mapped network interface to send remote stores to the network, and a fetch/deposit engine usually names "the annex" according to the construction plans of the vendor and according to other publications. The memory of a T3D node is a simple memory system built from DRAM chips without extensive support for interleaving and pipelined accesses. Unlike DEC Alpha workstations, the node has no virtual memory and runs with a special version of the DEC Microprocessor without the functional units for paged virtual memory.

The interface between the computation agent and the main memory is centered around an 8KB primary cache and a write back queue (WBQ) which are integrated on-chip within the DEC Alpha microprocessor. An overview of the memory system and its interface to the processor and communication system is shown in Figure 1.2 along with the other interfaces. An external read-ahead circuitry (RDAL) can be turned on by the programmer at load-time to improve performance of contiguous load streams; we have measured improvements of approximately 60% with read ahead. For writes, the default configuration of the cache is write-around, and support for writes consists of the write back queue provided by the microprocessor. The documentation of the Cray T3D Application Programmers Course [3] specifies the local read bandwidth at 55 MB/s for non-contiguous single word transfers, and up to 320 MB/s for contiguous reading of cache lines with read-ahead. The latency of a load from main memory is around 150ns.

The interface between the processor and communication system on the Cray T3D consists of the annex, a memory mapped communication port, which maps some range of free address space to the physical memory of another node in the system; this node is then selected as a communication partner. The communication partner can be switched with a fixed overhead by modifying the appropriate annex entry. The significant fixed cost for switching the communication partner justifies our classification of the T3D as a highly advanced distributed memory, message passing machine. Once a store operation is issued to the communication port, the communication subsystem takes over the specified address and data, and it sends a message out to the receiver. Remote loads are handled in a similar way and can be pipelined with an external, 16 element FIFO queue. This queue requires direct coding support by the programmer or compiler and is rarely used.

At the passive end the fetch/deposit engine completes the operation as a remote load/store on behalf of the user at the other node. These accesses happen without involvement of the processor at the receiver node (i.e., there is no requirement to generate an interrupt). This circuitry can store incoming data words directly into the user space of the processing element, since both address and data are sent over the network. The on-chip cache of the main processor can be invalidated line by line as data are stored into local memory, or it can be invalidated entirely when the program reaches a synchronization point.

Transfers from the processor to the communication system can be performed at a rate of approximately 125 MB/s, and if multiple nodes perform remote stores of contiguous blocks to a single node, these transfers can be processed at the full network speed (160 MB/s) [7]. The number of network nodes is only half the number of microprocessors (or processing nodes). If just one of the two processors is communicating at a time, the network can be accessed at up to 125 MB/s, if both processors are communicating the full speed of a network access is available and each processor obtains about 75 MB/s in bandwidth to access the network.

The interconnect topology for data transfers in the T3D is a three dimensional torus with dimension order wormhole routing. Service and IO nodes are inserted into the regular grid in at least two dimension, so that they can be reached by dimension order routing without any problems. Routing is fully deterministic and determined by a global hardware routing table loaded at boot time. There are several sets of virtual channels to permit full torus routing with a dateline and also to permit a complete separation between operating system traffic and user program traffic.

1.4 Transfer Modes

1.4.1 Overview

We focus on the performance of moving data itself and disregard any difference in amount of local or global cache coherency that the diffent technologies offer, since at this point none of the technologies can offer automatic fully coherent shared memory to support a standard shared memory programming in hardware. Such will remain a privilege of much less scalable or more expensive systems like bus based SMPs and directory based CC-NUMAs. Furthermore the implication of different network topologies is an extremely well researched topic and therefore we assume that a sufficient number of switches is used in both systems to provide full bisectional bandwidth as this is the case in most smaller systems. It is also clear the data transfers to remote memory must be pipelined and aggregated into large messages. The pure ping-pong latency of a single word transfer remains of little interest for a comparison of the sustainable end-to-end throughput achieved for different communication

patterns in different transfer modes with processors, co-processors and DMAs all working.

Until recently the maximal performance of the memory- and the I/O-system was rarely achieved by the network interconnects. Therefore neither the performance of the I/O bus designs nor the performance of the common system software was optimized enough to work with Gigabit networking. Those two factors are the principle bottleneck in today's clusters of PC's.

A further bottleneck is the lack of local memory system performance in PCs. Memory system performance is not only important to computational efficiency in applications with large datasets, but it is also the key to good performance of inter-node communication. While high end MPP node designs can afford memory systems with special hooks for inter-node communication at Gigabit/s speeds, low end systems must rely entirely on mass market memory systems and standard I/O interfaces (i.e. a PCI bus) for economic reasons.

The main difference between the SCI- and the Myrinet- network adapters are the default and alternate transfer modes they can operate in. Although the hardware mechanisms involved in transfers between main memory, staging memory and network FIFO queues may be vastly different, the purpose of all data transfers remains the same: Move data from user space of a sending process to the user space of a receiving process. Most interconnect designs can do this with close to peak speed for large blocks of data and for the special semantics of zero copy messaging. To explore these issues in more depth, we require that a direct remote memory operation can also include more complex memory operations at the receiving end e.g. strided stores. A typical application for such an operation would be the boundary exchange of an iterative FEM solver working on large, distributed sparse matrices. Figures 1.7 - 1.8 illustrate two options for transfers for each of the interconnects discussed earlier. One mode is mostly processor driven and utilizes the most direct path from memory to the network FIFOs, while the other mode is DMA driven. The latter mode makes a few additional copies on the way to the network interface but uses DMAs to do them in parallel to the regular activities of the processor.

– **Direct deposit, mapped**: The main processor pushes the data directly into the network FIFOs through regular store operations addressed to a special segment of virtual memory, mapped directly to the network interface and through that port on to the memory of the remote processor. Contrary to a common belief the precise layout of the assembly instructions to trigger this remote store or load operation does not matter. It is well conceivable that a parallelizing compiler handles a remote store as two separate stores for address and data, since the compilers have to know about local and remote for performance optimizations.

This mode of operation is the native mode of operation for the SCI adapter and for the Cray T3D which have both direct hardware support for it. A Myrinet adapter can only map the send registers and the staging memory (SRAM) directly into the user address space of the application, but not the remote memory itself. It seems that direct remote stores are impossible unless the two dedicated MCPs (co-processors) at the sender and the receiver side become involved. A dedicated control program for those co-processors must shadow the adapters staging memories, transfer the data across the network and move the incoming data to the remote memory at the receiving end, so an SCI like remote store operation can be emulated for contiguous and strided blocks of data. This technique does not work too well for an isolated store but performs adequately for an aggregation of multiple stores with indexes or strides.

– **Direct deposit by DMA**: The application stores its data (an potentially also the addresses) into a reserved, pinned address segment of local memory instead of the mapped remote address space. Starting from there the DMA engine of an adapter can pull the data directly into the network FIFO queues interface for transmission.
This mode of operation works very well for *contiguous* blocks of data. SCI in direct message passing mode can send blocks of the mapped memory using its DMA. The DMA controller utilizes the most efficient sequence of SCI transactions to achieve highest possible throughput. Myrinet with its 3 DMAs on the LANai allows a similar mode of operations. Furthermore there is a bit more flexibility since data can be gathered in small portions, stored at the staging memory and sent directly to the packet interface. The DMAs can be supervised and periodically restarted by the Lanai MCP.

– **Buffer packing with processor or DMA at the sender**: The main processor or a DMA of the PC gathers the segmented data into the network FIFO via a segment of mapped main memory. Then, the network card's co-processor or another DMA transfers the message into the network FIFO. In this mode the message is processed by either the main processor or a network processor and a DMA and finally transmitted into the network FIFO. Measurements indicate that this is the best way of transferring data and can therefore be called the *native* sender mode of Myrinet. Depending on the block size of strided and indexed transfers packing with the main processor first can be faster than gathering them directly with the DMA.

– **Buffer unpacking with processor or DMA at the receiver**: The main processor packs the destination store addresses along with the data into a message. The adapter gets the prepared message by a DMA and pushes it into the network FIFO. On the receiver node the main processor

or the network card's message processor reads address and data words and scatters the data via DMA transfers to a segment of main memory.
Again the message is processed by either the main processor or a network processor and a DMA at the receiver side. Measurements indicate that this is the best way of transferring data and can therefore be called the *native* receive mode for Myrinet. Depending on the block size and access pattern unpacking with the main processor can be faster than scattering them with the DMA.

1.4.2 Discussion of the native and alternate transfer modes in the three architectures

To implement the remote memory system performance tests with their complicated strided pattern we can either use the hardware support provided by the communication adapter or the packing/unpacking with the main processor that always leads to an additional copy operation, but avoids the inefficiency of single word transfers across the PCI bus

Myrinet: The Myricom Myrinet Adapter with its own RISC processor and its staging memory allows for different scenarios. Figure 1.6 shows two schematic flows of data, a direct deposit and a buffer packing/unpacking operation. The receiver adapter's processor can be used to unpack and scatter the data without invocation of the main processor. Here the DMA to the main memory is the bottleneck when small amounts of data (words) are transferred because of the PCI bus arbitration overhead for each transfer. For these access patterns the packing/unpacking by the main processor and sending the contiguous packed data with a DMA transfer leads to much higher throughput.

SCI: Figure 1.7 shows a schematic flow of data with the Dolphin PCI-SCI-Adapter. SCI supports a direct mapped mode which enables transparent access to mapped remote memory segments. As the data is first stored in stream buffers, this mode works perfectly well for contiguous blocks whereas for unfavorable access patterns, e.g. strided or indexed access, only one stream buffer can be used and consequently the performance drops below 1/10 of the maximal bandwidth. The performance can be increased in the same way as for Myrinet if the main processor unpacks the data after a fast contiguous transfer.

T3D: The T3D offers hardware support to perform direct user-space to user-space transfers for all communication patterns, contiguous and strided (see Figure 1.8. This capability potentially eliminates all buffer packing at the sender and unpacking at the receiver end even for the more complex access patterns. The Cray SHMEMPUT library (libsma.a) provides a thin layer to cover the hardware details of direct deposit for contiguous transfers. A buffer packing message passing style interface is provided by the Cray PVM

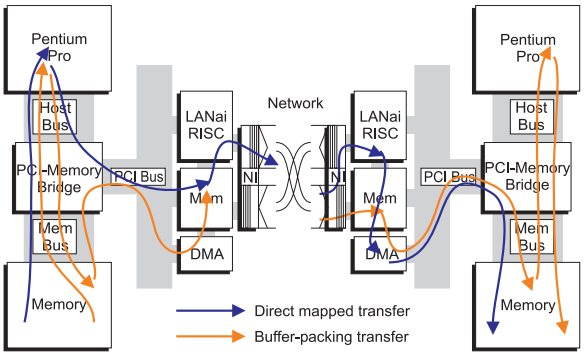

Fig. 1.6. Schematic flow of data with Myrinet. Different scenarios are possible. Some sort of direct mapped mode for chained transfers can be implemented using the SRAM and the LANai processor on the adapter card. The usual mode of operation is the message passing mode where the data is packed and unpacked by the main processor.

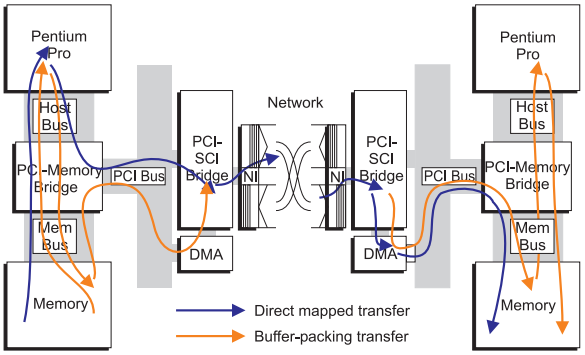

Fig. 1.7. Schematic flow of data with SCI. SCI supports a direct mapped mode which enables transparent access to mapped remote memory segments. For message passing the main processor packs the data which then can be pulled and sent by a bus master DMA of the SCI adapter card.

or MPI libraries for a higher level of messaging in system software. While both libraries contain primitives for direct contiguous block transfers, both libraries fail to provide adequate flexibility for transfers of strided and indexed data without prior copies in local memory.

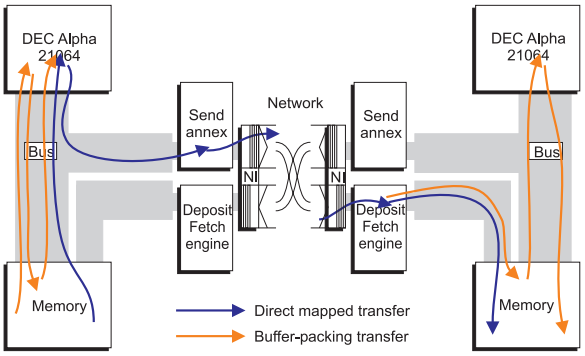

Fig. 1.8. Schematic flow of data on the Cray T3D. Direct transfers can be implemented mapping the entire remote memory into local address space (through annex). A special fetch/deposit circuitry handles incoming remote operations without involvement of the processor. For the message passing mode, the main processor packs the data in the local memory, copies it as a contiguous block to remote memory and unpacks it again.

1.5 Performance Evaluation

In our performance evaluation we discuss the throughput to move data in different application scenarios. For the best understanding of our measurements it is important to keep in mind at which level the benchmark is performed (lowest level of deposit or highest level of LAN emulation) and which data path (mode of operation) is used.

For all modes of operation that involve packing/unpacking operations of messages or buffers , the local memory system performance is very important for communication performance. As a preliminary step to our evaluation we developed a novel memory system microbenchmark. Unlike the simplistic McCalpin loops [6] our test captures all aspects of the memory hierarchy, in particular its performance behavior with temporal- and spatial locality (varying working sets and strides) [11]. Our ECT (extended copy transfer) characterization therefore goes beyond pure loads and stores bandwidth to measure the copy throughput with a simultaneous load and store data stream. Most end-to-end transfers in compiled parallel programs involve fine grain accesses, either with strides into arrays or a large number indexed smaller blocks to gather/scatter data from/into distributed collections of objects.

Therefore we graph access patterns for various strides, from contiguous blocks up to single 64bit entities spaced a constant strides of 64 apart.

The same method is used to measure remote accesses for machines with full, partial or without support for coherent shared memory. For the measurement we use the framework for characterizing local and remote memory system performance [10].

For the best characterization of an interconnect technology, the remote memory system performance figures for different access patterns (strides) and a large working set are the interesting issue. The measured copy performance for the same operation in the local memory system is included into the Figures 1.9 - 1.11 just for a comparison. Since we are only discussing remote deposit (and leave out remote fetch) all transfers in our charts are done by contiguous loads from local memory and strided stores to the same local or remote memory system. The performance numbers in Figures 1.9 - 1.11 show, that the copy bandwidth of the memory system on the Pentium pro PC's of only 45MBytes/s is much less than the peak performance of modern interconnects. Copying data in the same memory system with the processor is therefore always a bottleneck and must be avoided at all cost.

1.5.1 Performance of direct transfers to the remote memory

The performance of direct transfers is measured for contiguous blocks (Stride 1) and for increasing strides (2..64). The first set of performance curves in Figures 1.9 - 1.11 (filled bullet) marks the performance for the most direct transfers by store operations to the mapped remote memory or an emulation thereof. The second curve (triangle) marks the performance for highly optimized buffer packing transfers. In this case the transfers were optimized as well as we could. If the DMA was faster, then DMA was used. The relationship between local and remote memory performance can be understood by comparing the performance curves of local memory for the corresponding copy operation to the direct and buffer-packing remote performance curves (hollow bullets).

Deposit on Myrinet: For Myrinet we observe a uniform picture for strided data. Our emulation for direct deposits for small blocks of data (double words) works with a pipelined transmission of large data blocks with either the LANai or the main processor unpacking the strides (see Figure 1.9). The store by the DMA is very much affected by the size of the chunks transferred by one DMA activation. The buffer packing mode with the main processor seems to perform at about memory copy bandwidth whereas the DMA transfers suffer from the overhead of too many DMA initializations and too many PCI bus arbitrations. The buffer packing, native mode can fully use the DMAs to boost the case of large contiguous blocks.

As DMA transfers are very much affected by the blocksize we compare the performance for different blocksizes for Myrinet and SCI (see Figure 1.12). It turns out, that both adapters have the same problems with small blocks.

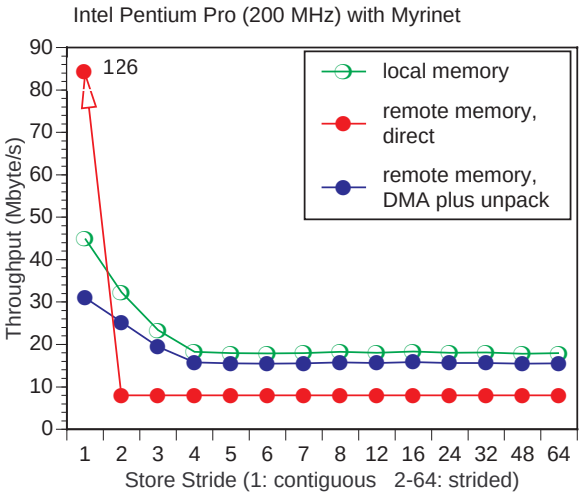

Fig. 1.9. Throughput of the Myrinet Host Adapter for direct mapped emulation and buffer packing transfers. As a reference the performance of the local memory system is given for the same copy operation (contiguous loads / strided stores).

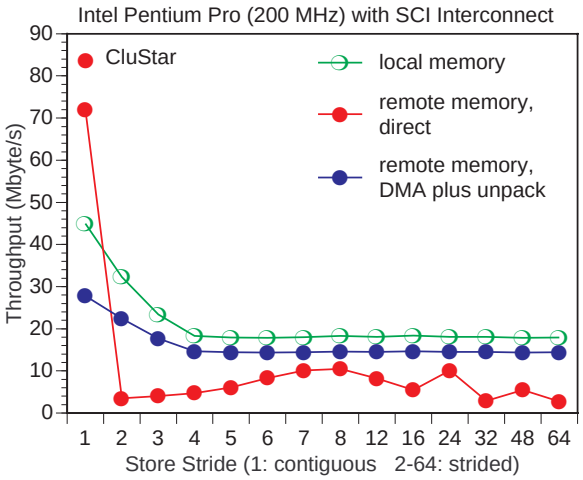

Fig. 1.10. Throughput of the Dolphin PCI SCI Adapter for direct mapped and buffer packing transfers. As a reference the performance of the local memory system is given for the same copy operation (contiguous loads / strided stores).

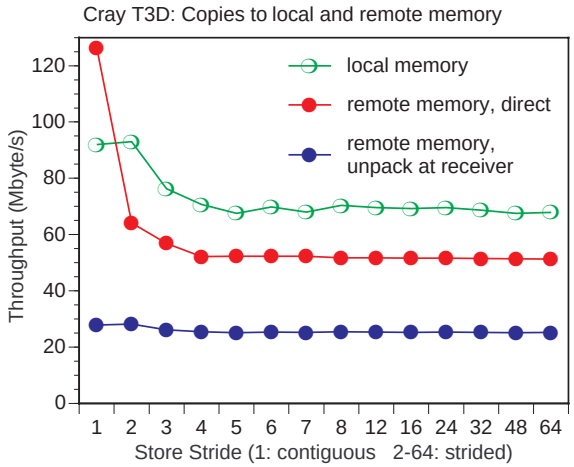

Fig. 1.11. Throughput of the Cray T3D for direct mapped and buffer packing transfers. As a reference the performance of the local memory system is given for the same copy operation (contiguous loads / strided stores). Note that two T3D nodes can exchange data faster than a single node can copy it.

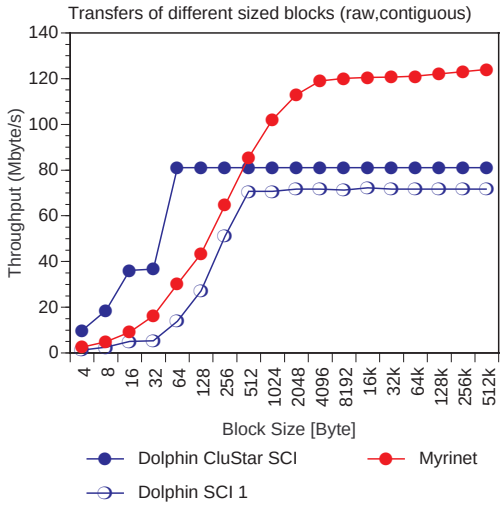

Fig. 1.12. Fastest transfers of different blocksizes for either Myrinet and SCI.

Deposit on SCI: For SCI we also observe good performance for contiguous blocks in direct transfer mode (see Figure 1.10). This is when the eight streambuffers work optimally. For strided data the performance of remote stores on PCI collapses to under 10 MByte/s and appears to be unstable. The sloped curve from stride 2 to stride 8 can be explained by the mechanics of the stream buffers. For stride 2 only one stream buffer is used as the direct mapping of the stride 2 addresses to the same 64 Byte streambuffer leads a sequence of non-pipelined single word transfer. The buffer is always sent directly with only 8 Bytes of data and the next value addressed to the same buffer has to wait, until an acknowledge releases the buffer again.

As with Myrinet it turns out to be faster with SCI to unpack a communication buffer than to execute transfers directly in this case.

Deposit on the Cray T3D: The Cray T3D offers (or better offered) a slightly different performance picture (see Figure 1.11). It turned out that it was always best to execute a data transfer in direct mode. Buffer packing included copies and those slowed down the transfers. For contiguous blocks a direct copy to remote memory was even faster than a local copy from and to memory - this is not surprising since two memory systems - one at the sender side and one at the receiver's side are involved for a single data transfer in the remote case.

1.5.2 Performance of PVM/MPI transfers

The performance of the higher level transfers indicates how well system programmers can work with the hardware. We used a full function standard message passing libraries with buffering for true postal message passing and a reduced zero copy library for reasons explained in the earlier Section. So our evaluations use two different tests exposing the performance at full postal functionality with buffering and at the reduced functionality with direct transfers.

MPI on Myrinet: On Myrinet we use BIP-MPI [8] for tests in message passing. BIP (Basic Interface for Parallelism) is a high performance library with a possibility simplified data transfer semantics. The code was developed at the ENS of Lyon, France. Although this implementation might eventually fall a bit short in terms of compliance with the extensive MPI standard, it seems to provide a stable API for all important basic functions. BIP-MPI is a modified MPICH version using BIP to drive the Myrinet network hardware. The performance of BIP-MPI (see Figure 1.13) matches the raw performance at 126 MByte/s for blocking send and receives measured with large blocks ($>$ 1 MByte). Half of peak performance can be reached with messages of roughly 8 KByte size. The performance results are summarized in Figure 1.16.

For the non-blocking calls, where the sends are posted before the receives, MPICH enforces buffering which drops the performance to the one of the local memory copy. An optimization uses the LANai staging memory wherefore

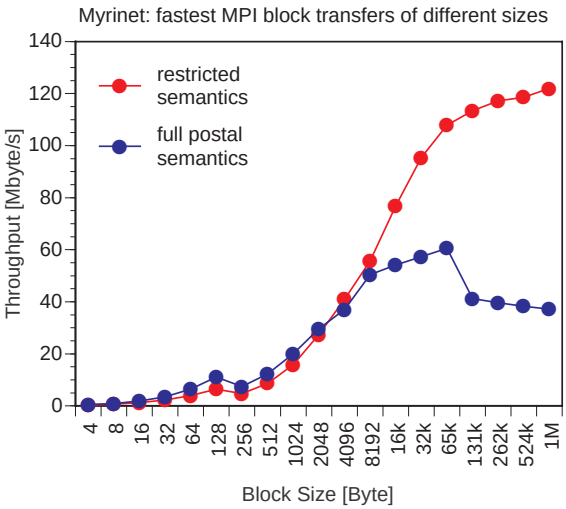

Fig. 1.13. Throughput of BIP-MPI transfers over Myrinet either for restricted and full postal semantics.

blocks until 64KByte can be buffered without an additional local memory copy. When two sends are posted before the receive the peak performance is measured at 32KByte blocks.

MPI on SCI: For SCI we could not run tests and include a number yet, but it appears that MPI for SCI in its fully standardized version is promised as a deliverable of a joint research project between industry and academia.

MPI on the SGI-Cray T3D: For the Cray T3D, we list the performance of PVM 3.0 instead, a highly optimized, fully functional message passing library supplied by the vendor upon delivery of there first T3D system in 1993. A maximum of 30 MByte/s is delivered for large blocks ($>$ 1 MByte) and half of the peak bandwidth is achieved with messages of 2 KByte size.

In addition we show results of a later high performance MPI implementation for the Cray T3D by the Edinburgh Parallel Computing Center in cooperation with SGI/ Cray Research (CRI) [2] (see Figure 1.15). The EPCC MPI provides the full MPI specification and was developed using the SHMEM get/put primitives on the T3D. The measured performance corresponds to the BIP-MPI over Myrinet where blocking calls don't need any buffering whereas non-blocking semantics slows down by an additional copy in the local memory system.

1.5.3 Performance of TCP/IP transfers

For a significant growth of market share for clusters of PCs it is most important to port traditional applications quickly and easily by simply substituting

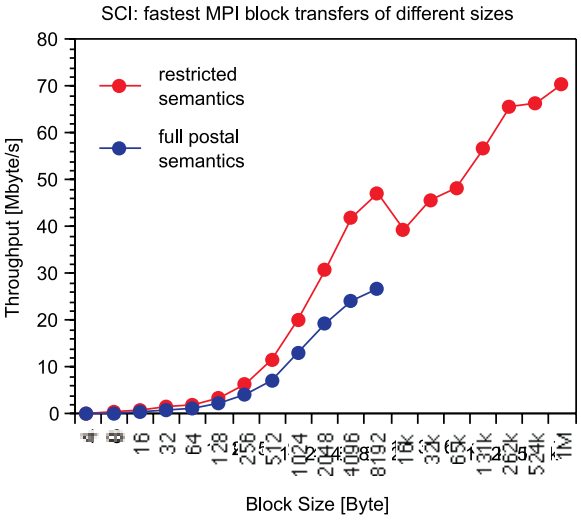

Fig. 1.14. Throughput of ScaMPI transfers with SCI either for restricted and full postal semantics.

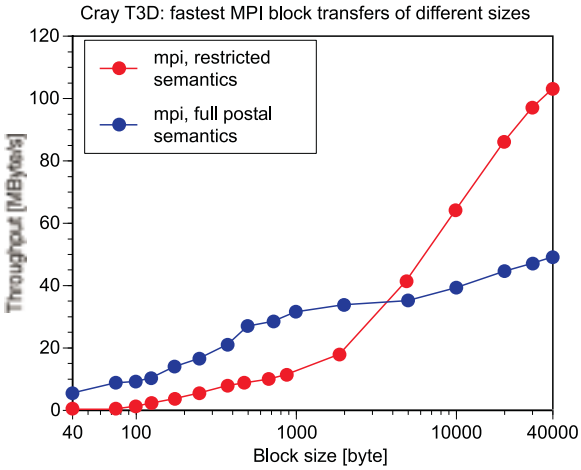

Fig. 1.15. Throughput of EPCC-MPI transfers for the T3D either for restricted and full postal semantics.

a conventional LAN network (e.g. switched 100BaseT) by a gigabit technology interconnect, especially in the booming area of servers for the internet and for the important application of distributed database and middle ware systems. In this mode of operation the performance of a fully standardized protocol stack like TCP/IP is essential. Therefore we measured the standard and the best LAN/IP emulation packages that were available for each interconnect technology (with the exception of the T3D, where IP does not make much sense).

IP on Myrinet: Myricom offers a fully compliant TCP/IP protocol stack that transfers data at 20 MByte/s. BIP-TCP [8] improved this performance by using the "zero copy" BIP interface so that about 40MByte/s are reached. The BIP implementers use the original Linux protocol stack and substituted the transfer mechanism using their BIP message passing system for the lower layers.

IP on SCI: For our SCI-1 cards we could only reproduce 13.2 MByte/s with a TCP/IP protocol stack provided by Dolphin Interconnect running under Microsoft Windows NT. We also tested a TCP/IP implementation similar to the BIP-TCP of PC2 Paderborn running under Linux. It also uses the standard Linux and substituted the transfer mechanism using remote mapped segments. With the Release D Cards from Dolphin the about 22 MByte/s are reached.

IP on the SGI-Cray T3D: The T3Ds were unusually connected to a Cray C90 or J90 vector processor as attached massively parallel multiprocessor. The processing nodes execute only threads in SPMD style controlled by the host and their little runtime system did neither support an IP number nor a socket API. IP would not make much sense in that context.

1.5.4 Summary and comparison

A summary of the measured throughput for the three technologies at different levels is given in Figure 1.16. The raw deposit performance is compared to the MPI bandwidth with either blocking and non-blocking operation. The performance results confirm that the blocking MPI bandwidth matches about the raw performance of the direct deposit whereas the non-blocking calls with buffering semantics force Myrinet and the T3D into buffering. The performance drops down to the copy performance of the local memory system in the corresponding Pentium Pro or DEC Alpha node- architecture. The TCP/IP performance on both PCI based technologies (Myrinet and SCI) reflects the same results and again confirm the overhead of copies in the main memory.

The efficiency of the network interface logic can be evaluated by the ratio of raw data speed on the wires over the maximal throughput sustainable between a sender and a receiver (see Figure 1.17). We assume highly optimized drivers for this test and search for the best measured or published value. The

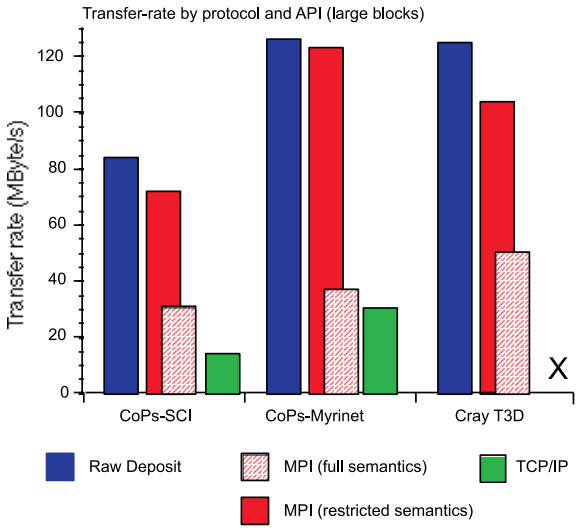

Fig. 1.16. Measured performance of different protocols and APIs over SCI and Myrinet with reference to the Cray T3D. Raw deposit performance is compared to MPI/PVM performance. For the T3D we list the performance of PVM 3.0 library supplied by the vendor. TCP/IP implementations are provided for both PCI adapter cards.

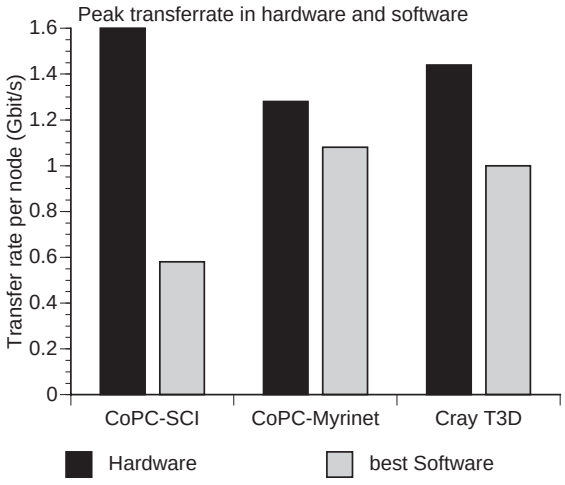

Fig. 1.17. Comparison of communication efficiency in the three evaluated technology. Conclusions about the efficiency are drawn from a comparison of the raw hardware speed vs. best possible throughput in software (low level primitives)

raw hardware speed is specified as 1.6 GBit/s for SCI and 1.28 GBit/s for Myrinet. In both cases full duplex operation at this speed must assumed and is required for a proper characterization. For a real application data transfer it takes the work a sender and receiver to make the entire transfer happen. Doubling the link capacity by counting send- and receive-capacity together would be like charging the passengers separately for the takeoff and the landing for each flight segment of a plane trip. For the SGI-Cray T3D the channel width of its interconnect is known to be 16 bit plus handshake lines but the precise clock on the interconnect remained unpublished. In some experimental settings we could push data at a rate of approximately 160 MByte/s over a single links. We also know that the best packet layout is 4-8 header flits and 16 flits of payload so its raw link capacity is probably between 1.28 GBit/s and 1.92 GBit/s.

1.6 Conclusions

For our study we managed to define three different level of system software support for Gigabit/s networks that permitted a fair comparison of highly different networking technologies based on microbenchmarks. Our methodology of evaluation was able to deal with the many different mode of operation of the networking interfaces involving different kinds of direct hardware support in VLSI, from a DMA agent to subsystems with a message co-processor.

For the performance of direct remote memory operations we notice excellent performance on both low-end Gigabit/s technologies in a few ideal cases. The transfer rates typically peak near the bandwidth limits of the PCI-Bus or the interconnect and are almost comparable the rates seen in traditional MPP supercomputers. But such good performance is only achieved for the simple transfer modes and in communication scenarios like direct remote deposits of contiguous blocks of data or restricted MPI semantics. For strided data or for remote loads of single words the performance of the SCI and Myrinet interconnect collapses, while the traditional MPP can do those cases at acceptable speeds. The DMA transfers suffer from the overhead of too many DMA initializations and too many PCI bus arbitrations which can be bypassed by buffer packing mode with the main processor performing at about memory copy bandwidth. For the implementation of message passing libraries with buffering semantics (e.g. MPI) the performance of the Myrinet interconnect is reduced to the local memory system bandwidth while the traditional MPP can do those cases at better speeds. Similar limitations due to copies in the local memory system will slow down the IP over LAN emulation in most cases.

Given the low cost of these interconnect boards, their performance close to a Gigabit is remarkable, but due to the aforementioned limitations we expect that any Cluster of PCs built with those PCI card interconnects will get

into some difficulties with applications that require complex, dense communication patterns or rely on for standardized high level networking software such a TCP/IP.

1. Nanette J. Boden, 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(1), pages 29–36, February 1995.
2. K. Cameron, L. J. Clarke, and A. G. Smith. CRI/EPCC MPI for T3D. In *Conference paper, 1st European Cray T3D Workshop, EPFL*, Sept 1995. http://www.epcc.ed.ac.uk/t3dmpi/Product/Performance/index.html.
3. Cray Research Inc. *CRAY T3D Applications Programming Course and Cray T3D Hardware Reference Manual*, Nov 1993. TR-T3DAPPL.
4. Dolphin Interconnect Solutions. *PCI SCI Cluster Adapter Specification*, 1996.
5. INTEL Corporation. *INTEL 440 FX PCISET*, 1996.
6. John D. McCalpin. Sustainable memory bandwidth in current high performance computers. Technical report, 1995.
7. R. Numrich, P. Springer, and J. Peterson. Measurement of communication rates on the cray t3d interprocessor network. In *Proc. HPCN Europe '94, Vol. II*, pages 150–157, Munich, April 1994. Springer Verlag. Lecture Notes in Computer Science, Vol. 797.
8. L. Prylli and B. Tourancheau. BIP: A New Protocol Designed for High Performance Networking on Myrinet. Technical report, LHPC and INRIA ReMaP, ENS-Lyon, 1997. http://lhpca.univ-lyon1.fr/.
9. IEEE Computer Society. *IEEE Standard for Scalable Coherent Interface (SCI)*. IEEE, ieee std 1596-1992 edition, August 1993.
10. T. Stricker and T. Gross. Optimizing memory system performance for communication in parallel computers. In *Proc. 22nd Intl. Symp. on Computer Architecture*, pages 308–319, Portofino, Italy, June 1995. ACM/IEEE.
11. T.Stricker and T.Gross. Global Address Space, Non-Uniform Bandwidth: A Memory System Performance Characterization of Parallel Systems. In *Proceedings of the ACM conference on High Performance Computer Architecture (HPCA3)*, 1997.