Comparing MPI Performance of SCI and VIA

Friedrich Seifert, Daniel Balkanski, Wolfgang Rehm
{sfr,danib,rehm}@informatik.tu-chemnitz.de

Technische Universität Chemnitz
Fakultät für Informatik
Straße der Nationen 62, 09111 Chemnitz

Abstract— **Both the Scalable Coherent Interface (SCI) and the Virtual Interface Architecture (VIA) aim at providing effective cluster communication. While the former is a standardized subnet technology the latter is a generic architecture which can be applied to a variety of physical medias. Both approaches provide user level communication, but they achieve it on different ways and thus, have different characteristics that are independent of the actual implementation. In this paper we report and compare the raw network speed of an SCI and a VIA implementation as delivered by MPI and show how it affects application performance by means of the NAS Parallel Benchmark Suite.**

Keywords—**NetPIPE, NAS Parallel Benchmarks, SCI, VIA, MPI**

I. Introduction

High performing interconnect systems are of key importance for effective cluster computing. Several solutions have been introduced, one of which is the Scalable Coherent Interface (SCI), a technology to build distributed shared memory as well as message passing systems on top of it. Its outstanding characteristic, however, is the shared memory. SCI allows nodes to share parts of their physical address spaces. Hence, in combination with an appropriate paging system processes are able to share their virtual address spaces. Communication can thus take place by the processors issuing simple load and store instructions. That means there is no need for additional protocol layers to transfer data from one node to another, not to mention operating system kernel calls. This gives SCI a clear advantage in terms of latency. Dolphin ICS states 2.3μs for their PCI–SCI–bridges (type D310), which is the version we used for our tests.

The Virtual Interface Architecture [2] also aims at reducing latencies, however, at a higher level. It defines a generic system architecture independently from the physical layer. The main objective of VIA is to move the operating system out of the time critical communication path. For this a VIA NIC provides applications direct access to the network through so called *Virtual Interfaces* or *VI*s. VIs are connected to each other in a point-to-point fashion. The VI Architecture is based on descriptor processing. A VI comprises two *work queues*, one for send descriptors and one for receive descriptors, and a pair of appendant *doorbells*. In order to start a data transfer the sender prepares a descriptor that contains the virtual address of the data to be sent, puts it on the send work queue and rings the doorbell. This is referred to as *posting* a descriptor. Thereupon the NIC starts processing this descriptor, i.e. it reads the data from the user buffer via DMA and sends it through the network to NIC hosting the peer VI. By this time the receiver must have posted a receive descriptor pointing to the destination buffer. When the message arrives the NIC takes the next descriptor from the head of the specified VI's receive queue and writes the data, again by DMA, into memory right into the user buffer. The work queues reside in user memory and the doorbells are mapped into user address space as well, thus there is no kernel call needed to start a data transfer. The completion of a descriptor can be checked either by polling its status or by blocking the process by means of a kernel trap. VIA also provides a remote DMA (*RDMA*) mechanism where the sender specifies both the local and the remote address.

Another characteristic of VIA is that all memory which is to be used to hold descriptors or data buffers must be *registered* in advance. That means that all involved memory pages are locked into physical memory and the addresses are stored in the NIC's *Translation and Protection Table* (TPT).

There are several VIA implementations around. While Berkeley-VIA [9], Compaq's ServerNet I and M-VIA [10] emulate the VIA functionality in software[3], Giganet's cLAN [11], ServerNet II [12], Finisar's FC-VI host bus adapter [13] and Fujitsu's Synfinity CLUSTER are native implementations.

It is important to mention that a combination of both technologies, SCI and VIA, is also possible. In February 1998 Dolphin announced a VIA implementation based on their PCI–SCI bridge [3]. Although they did not publish any technical details of their system, it is very likely that it is a case of software-emulated VIA, as they said no modifications to the existing hardware were needed [4]. Unfortunately, there have been no more public announcements about progress in this direction, neither has this software been made available for testing. In contrast to Dolphin's pure software solution our research group is working on a native, i.e. hardware supported, extended VIA implementation based on SCI [5], [6], [7], [8]. However, until now the project is in an too early stage to be suitable for comprehensive measurements.

In this paper we want to investigate the performance differences between these technologies and how they affect the run time of parallel applications by means of one specific implementation of each, Dolphin's SCI hardware and Giganet's VIA hardware. For comparison to conventional networking technology we also included FastEthernet into our tests. Since all tests are based on MPI [16] the results give an overall assessment

The work presented in this paper is sponsored by the SMWK/SMWA Saxony ministries (AZ:7531.50-03-0380-98/6). It is also carried out in strong interaction with the project GRANT SFB393/B6 of the DFG (German National Science Foundation). We also thank Giganet Inc. and MPI Software Technology Inc. for providing us with their cLAN VIA hardware and MPI/Pro software. Further thanks are extended to Scali Computer AS for providing us with their Scali Software Platform.

[3] M-VIA 2.0 is intended to support native implementations as well.

of the communication hardware and software, which is what an application programmer experiences.

II. The test bed

A. Hardware

Out test environment consisted of nine Pentium III 450 MHz machines. Half the machines were equipped with 512 MB, the others with 384 MB. All of them were connected by switched FastEthernet (using a 3com SuperStack switch) and by SCI in a ring topology based on Dolphin's D310 PCI–SCI–bridges. Using our SCI switch was impossible because it is not supported by Scali's Software (see below). Moreover, eight machines had a cLAN 1000 VIA adapter from Giganet Inc. connected through an eight-port cLAN5000 switch.

B. Software

All machines were running RedHat Linux 6.0 resp. 6.1. For the SCI test we used Scali's SSP version 2.0 [14]. It has been designed for 2D-torus topologies, which rings are a subset of. MP-MPICH from Aachen[15] would have been an alternative MPI implementation but at the time of the tests it was still in an early stage and didn't run stable on our cluster. Besides, it showed performance problems on Linux platforms.

The measurements on Giganet and FastEthernet were done using MPI/Pro by MPI Software Technology Inc., which is the only MPI with support for Giganet's cLAN hardware at the moment.

The NAS Parallel Benchmarks were compiled using the Fujitsu C++ Express Compiler version 1.0 and Fujitsu Fortran 95 Express Compiler version 1.0.

III. Measuring message passing performance using NetPIPE

To measure the pure MPI performance we chose the NetPIPE (Network Protocol Independent Performance Evaluator) benchmark [18]. It was designed as a tool to evaluate different types of networks and protocols and gives answers to the following questions:

- How long does it take to transfer a data block of a given size to its destination?
- What's the maximal bandwidth?
- Which network and protocol transfer a block of given size the fastest?
- What overhead is incurred by different protocols on the same network?

As we examine only one protocol on all networks, which is MPI, the last question is of less interest for us, and we concentrate on throughput and latency.

A. NetPIPE's measuring method

The heart of NetPIPE is a ping-pong loop. A message of a given size is sent out. As soon as the peer receives it, it sends a message of equal size back to the requester. This is repeated several times for each size and the smallest round trip time is recorded. The amount of data is increased with each pass. The number of repetitions depends on it. It is calculated in a way that the transfer lasts for a fixed time (0.5 seconds by default). The program produces a file containing transfer time, throughput, block size and transfer time variance for each message size.

B. Throughput

Figure 1 shows the MPI send/receive throughput of Giganet, SCI and FastEthernet (in Mbit/sec) depending on the message size. There is, as expected, a clear difference between FastEthernet and the other systems.

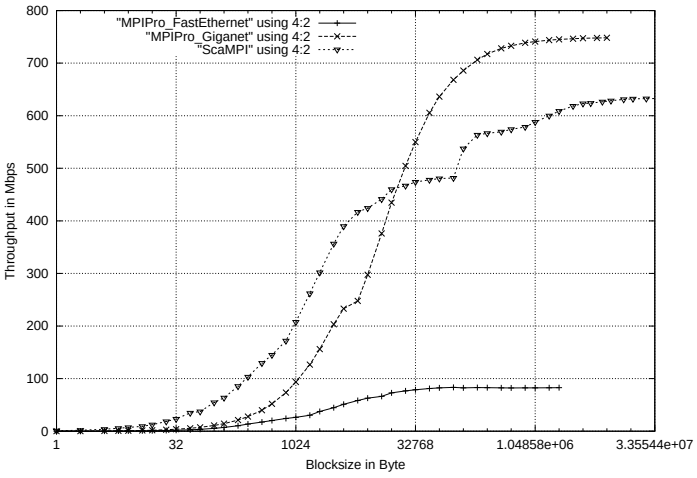

Fig. 1. NetPIPE throughput chart

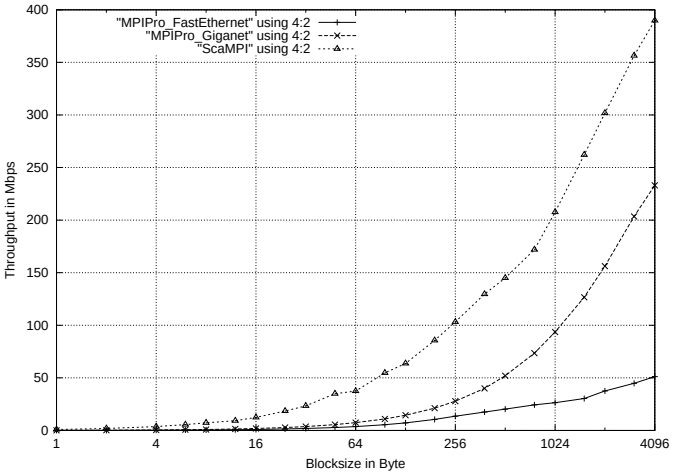

Fig. 2. Small message detail of throughput chart

For large messages MPI/Pro on Giganet delivers the highest performance at 748 Mbit/s or 93.5 MB/s. The kink at 4 KB is caused by switching from eager to long protocol. The former sends messages immediately using VIA send/receive mode which are then buffered as unexpected messages at the receiver. There is an additional copy needed from the MPI-internal buffer to the user buffer. The long protocol is a rendezvous protocol. The sender waits until the receive operation is started. The receiver registers the destination buffer with the VI NIC and sends the destination address to the sender which then transfers the data by RDMA. If the protocol switch point, which is at 4 KB by default, is moved to 8 KB the curve becomes smoother, i.e.

the throughput between 4 and 8 KB is improved a bit. Above that message size the copy operation of the eager protocol is more expensive than the additional message and the memory registration of the long protocol.

ScaMPI's peak performance is about 20 percent lower at 608 Mbit/s (76 MB/s). There is one major kink in the curve at 128 KB. It results from switching from eager protocol to transport protocol, by analogy to MPI/Pro. The default eager buffer size of ScaMPI is just 128 KB. There is one more protocol, the *inline* mode, which is used for messages up to 560 bytes, but this switch point cannot be seen in the curve. This suggests that the ScaMPI developers chose the optimal value in order to get an even performance curve.

MPI/Pro on TCP is able to exploit about 83 percent of FastEthernet's wire speed, i.e. 10.3 MB/s.

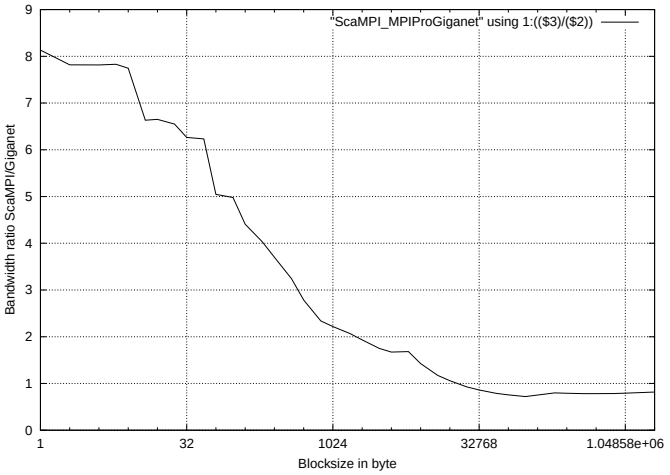

Fig. 3. Bandwidth ratio of SCI and VIA

For small messages the relation between the networks is different. While for large messages Giganet is the fastest for messages up to 16 KB SCI is clearly ahead. As it can be seen from figure 3 SCI is up to eight times faster than VIA. An explanation for this will be given in the following section.

C. Latency

Besides the bandwidth chart NetPIPE offers a so called *saturation* chart. It shows what amount of data can be transfered within a given time. The minimal transfer time represents the latency of the network and the software layers. The following table summarizes the times measured:

Network	Latency
FastEthernet	125 μsec
SCI	8 μsec
VIA	65 μsec

SCI shows the best latencies by far due to its shared memory concept. A simple store operation by the CPU is needed to transfer a data item. The PCI-SCI-bridge transparently translates it into an SCI request and sends it to the target node. In contrast to the value given by Scali (see section I) the lowest raw latencies we could measure on SCI shared memory were about 5 μsec, i.e. MPI imposes an overhead of 3 μsec for very small messages.

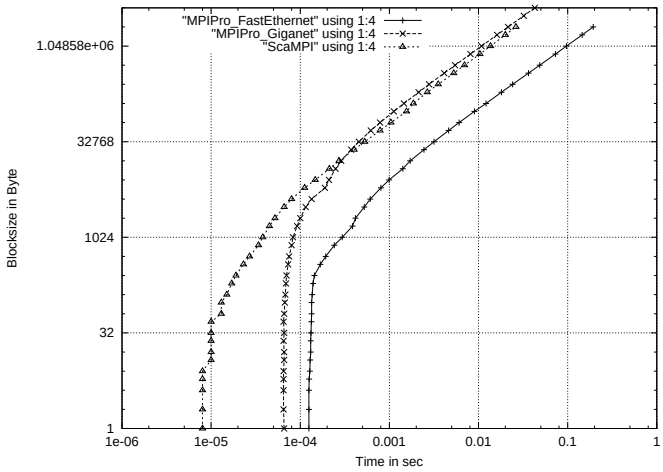

Fig. 4. NetPIPE saturation

The reason for the higher latencies of Giganet can be found in the VI Architecture itself. As described above, VIA uses descriptor based communication. A descriptor must be prepared and posted to the NIC. Then the hardware starts reading the descriptor from main memory by means of DMA. After retrieving the data address it must perform another DMA cycle in order to get the actual data. VIA also provides a mechanism to save the second DMA action: a descriptor may contain up to four bytes of *immediate data*, which is transfered from the send descriptor directly to the receive descriptor. However, this cannot reduce latency essentially since, on the one hand the amount of immediate data is rather small and, on the other hand, the DMA transfer for reading the descriptor remains. Thus, SCI shared memory will always have better short message performance than any VIA implementation on a comparable physical media. With SCI the data itself is written to the NIC, not a one or even two step reference to the data.

But that does not really explain the eight times higher latencies of VIA. The hardware latency of Giganet is as low as 7 μsec after all. Another reason is the synchronization needed between the sender and the receiver. The VI Architecture requires that a receive descriptor be posted before the peer starts the send operation. Otherwise the message would be dropped and the even connection would be broken if the VI connection was established in reliable mode. That's why an MPI implementation for VIA must unconditionally avoid such situations by synchronizing sender and receiver. These synchronizaton messages increase communication startup times.

A third reason for the higher latencies of VIA is the way MPI/Pro checks completion of send and receive descriptors. The version we tested uses waiting mode where the communicating thread is blocked until a descriptor has been completed. Reawakening a process is, of course, more expensive than polling on a local memory location as it it done in ScaMPI. Future versions of MPI/Pro will also be able to use polling mode. A prototype implementation has already shown latencies below 20 μsec [19].

D. Summary

The NetPIPE benchmark revealed clear performance differences between the high speed networks examined. SCI's extremely low startup times provide superior bandwidth for messages up to 16 KB. For larger messages Giganet is faster, but not significantly. In the following section we will examine how these differences impacts particular parallel applications.

IV. NAS Parallel Benchmark

Latency and bandwidth figures are often used for marketing purposes to convince potential customers of the capabilities of a certain network technology. However, the performance gain for parallel applications cannot be derived directly from them. It depends extremely on the communication pattern of the algorithms used. Of course, a high speed network is useless if the tasks exchange only a few bytes every now and then. But even for intensely communicating programs the influence of the network depends on the size of the messages. We have chosen the NAS Parallel Benchmark (NPB) suite version 2.3, which consists of five parallel kernels and three simulated applications from the field of Computational Fluid Dynamics (CFD). They have been designed by NASA Ames Research Center. The benchmarks, which are derived from computational fluid dynamics codes, have gained wide acceptance as a standard indicator of supercomputer performance [20]. Of course, the NAS benchmark suite is limited to specific numerical problems and hence is not representative for other kinds of applications, such as databases or datamining. However, since our group is involved in a Collaborative Research Center "Numerical Simulation on Massively Parallel Computers", we've found this benchmark suitable. We will give the results of all these tests at the end of this document (see figure 7), and we will discuss one benchmark from each group in more detail in this section.

In order to find explanations for the results gained we analyzed the communication of the programs by means of the MPI profiling interface. We wrote wrappers for each MPI data transfer function used by the benchmarks and counted their calls. That was done separately for each power of two of the message size. The findings were quite instructive. So, the EP test, for instance, is really inappropriate for evaluating cluster interconnects as its entire communication consists of three allreduce operations of four to eight byte and one allreduce of 64 to 128 byte, independently on the problem class and number of processes.

A. The IS benchmark

The IS (Integer Sort) benchmark is one of the parallel kernels of the NPB suite. It performs a sorting operation that is important in particle method codes. In such kind of applications particles are assigned to cells and may drift out. They are reassigned to the appropriate cells by means of the sorting operation [21].

A.1 The results

Figure 5 shows the performance for the IS test on FastEthernet, Giganet (both with MPI/Pro) and SCI with ScaMPI. All NPB tests can be run for several sizes: W (Workstation), A, B and C. We were not able to run class C as it required too much memory. IS runs at a power-of-two number of processes, i.e. 2, 4 and 8. We also added the numbers for the serial version to see the benefit of parallelization. The charts show the total million operations per second rate (Mop/s) rate over number of processes.

For all cases MPI/Pro on Giganet is the winner. However, the distance to ScaMPI shrinks as the problem size increases, so that for class B ScaMPI and gets very close to Giganet. There is a performance improvement of factor 2.5 in comparison to FastEthernet. For the latter the two process version is even slower than the serial implementation, and the speedup for eight processes is not even two. It is even worse for the smaller classes. The workstation test with eight nodes is not faster than the serial version which suggests that the communication overhead eats up all the acceleration of the computation by the additional processors.

A.2 The communication pattern

Size	Count	Message Type
4	1	send
4	1	reduce
8	1	reduce
8192	11	allreduce
4	11	alltoall
16777216	11	alltoallv

TABLE I
COMMUNICATION STATISTICS FOR IS CLASS B USING 8 TASKS

As table I shows the IS algorithm mainly relies on *allreduce*, *alltoall* and *alltoallv*, which is a vectorized variant of the all-to-all operation. *allreduce* performs an certain operation on a set of data distributed across all processes whereat all processes get the result. In an all-to-all operation, as the name suggests, each task sends a piece of data to all others, i.e., one such operation comprises $n(n-1)$ individual messages with n as the number of tasks.

The only parameter that changes with the number of processes and the problem class is the size of the *alltoallv* messages. It increases with the problem size, but it is the smaller the more processes are involved since each processor gets a smaller piece of the data as shown in table II.

ntasks:	2	4	8
Class W	2 MB	1 MB	0.5 MB
Class A	16 MB	8 MB	4 MB
Class B	64 MB	32 MB	16 MB

TABLE II
MESSAGE SIZES FOR *alltoallv*

The huge all-to-all messages are most probably the reason for the dramatic differences between FastEthernet and the two high speed networks. For the B class with eight processes a total amount of 896 MB must be transferred at each all-to-all exchange. In our opinion the *allreduce* and *alltoall* operations are of nearly no importance compared with those voluminous ones.

The difference between SCI and VIA are difficult to explain by the measurements given in section III. The performance rise of ScaMPI from class W to A at eight processes could be attributed to the higher bandwidth of about 8.5 percent for 4 MB messages as against 512 KB messages. However, this does not hold true for four processes although the transfer rate for 8 MB is higher than for 1 MB. Hence there must be other nonobvious factors influencing overall performance.

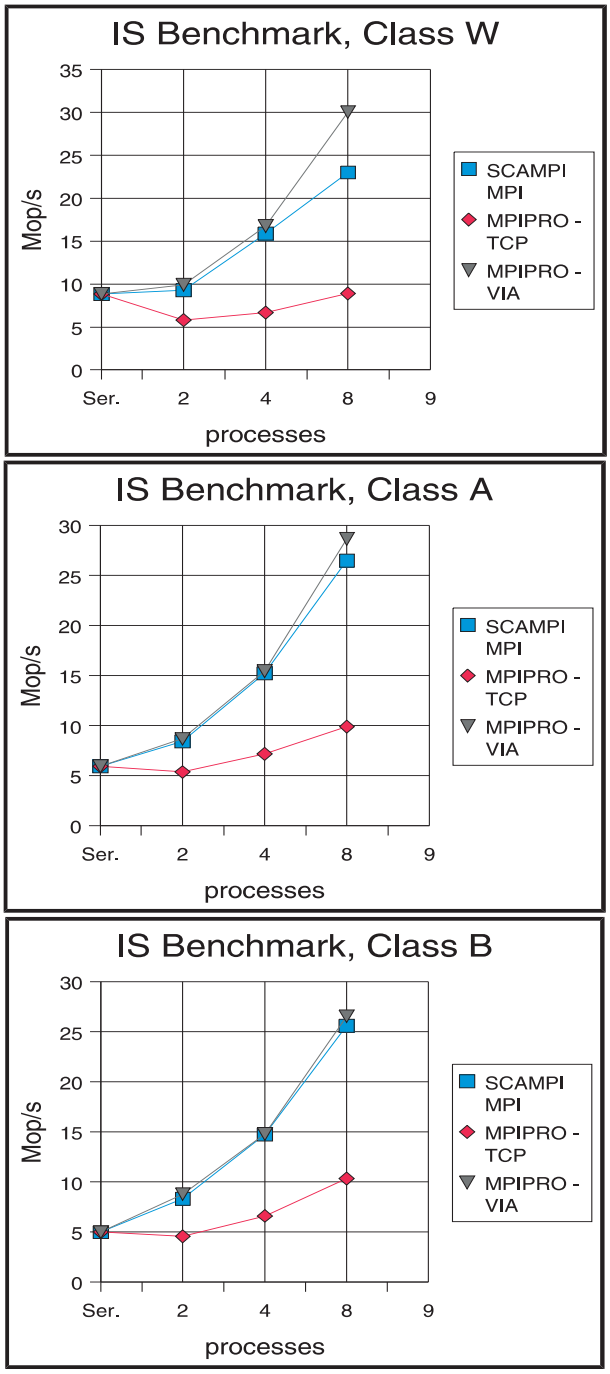

Fig. 5. IS results

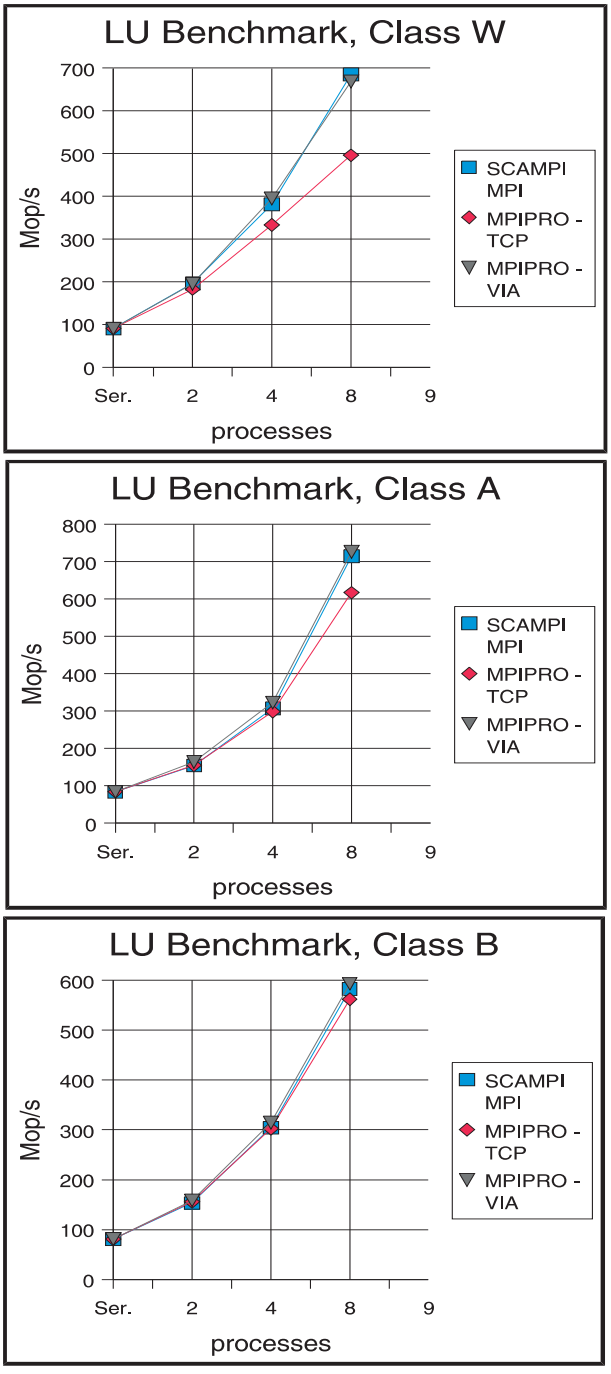

Fig. 6. LU results

B. The LU benchmark

Unlike the name would suggest the LU benchmark doesn't perform an LU factorization but instead employs a SSOR(symmetric successive over-relaxation) numerical scheme to solve a regular-sparse, 5x5 lower and upper triangular system. This problem represents the computations associated with a newer class of implicit CFD algorithms, typified at NASA Ames by the code INS3D-LU [21]. LU is one of the simulated application benchmarks.

B.1 The results

Figure 6 shows the results. Like IS it runs on power-of-two numbers of processes.

All in all there is nearly no difference between the high speed interconnects. For Workstation class on eight processes SCI

is slightly faster than Giganet whereas the things are opposite in the other cases. Further, the advantage of SCI and Giganet over traditional FastEthernet becomes the smaller the bigger the problem size is. For class B there is no significant difference between them any more. This behavior can explained if one considers communication pattern of the algorithm.

B.2 The communication pattern

Size	Count	Message Type
512	9300	send/recv
1024	9300	send/recv
32768	302	send/recv
65536	302	send/recv
4	3	bcast
8	5	bcast
64	1	bcast
8	4	allreduce
64	4	allreduce
	1	barrier

Size	Count	Message Type
1024	25000	send/recv
2048	25000	send/recv
262144	252	send/recv
524288	252	send/recv
4	3	bcast
8	5	bcast
64	1	bcast
8	4	allreduce
64	4	allreduce
	1	barrier

TABLE III
COMMUNICATION STATISTICS FOR LU CLASS W (TOP) AND CLASS B USING 8 TASKS

As can be seen from table IV-B.2 the communication of LU is dominated by send/receive operations. The few *allreduce* and *broadcast* operations can be neglected. For smaller process groups the message sizes grow but the total amount of data transferred remains relatively the same. When the problem size is increased the message sizes as well as the number of messages rise.

Table IV gives the total execution times for ScaMPI in seconds.

ntasks:	2	4	8
Class W	92.0	47.4	26.4
Class A	769	388	167
Class B	3250	1630	856

TABLE IV
TOTAL EXECUTION TIMES FOR LU ON SCAMPI (IN SEC)

The times for the maximum number of tasks are in relation $1 : 6.3 : 32$ for W, A and B. The number of messages, however, rises far slower. The relation is $1 : 1.6 : 2.6$. This explains why FastEthernet achieves nearly the same overall Mop/s rate as SCI and VIA, since runtime is mainly determined by computation in this case.

LU's communication comprises small messages where SCI is faster as well as larger messages where VIA is faster. The total amount of data sent in large messages (above 32 KB) is approximately twice this sent in small messages (below 2 KB). But the bandwidth for the small messages is only half of the large message bandwidth. Thus none of the high speed technologies can take pure advantage.

C. The other benchmarks

The results of all NPB tests are given in figure 7. The numbers shown represent the total Mop/s rate. The BT and SP tests run on a square number of processors while the others need a power of 2 number of processors. This limitation and the fact that we only had an eight port Giganet switch explain the gaps in the table. Although we did not analyze the results of the other tests in the suite [21], we will give a brief explanation of them for the reader's convenience:

Conjugate Gradient (CG) Benchmark In this benchmark, a conjugate gradient method is used to compute an approximation to the smallest eigenvalue of a large, sparse, symmetric positive definite matrix. This kernel is typical of unstructured grid computations in that it tests irregular long-distance communication and employs sparse matrix-vector multiplication.

The Embarrassingly Parallel (EP) Benchmark In this kernel benchmark, two-dimensional statistics are accumulated from a large number of Gaussian pseudo-random numbers, which are generated according to a particular scheme that is well-suited for parallel computation. This problem is typical of many Monte Carlo applications. Since it requires almost no communication, in some sense this benchmark provides an estimate of the upper achievable limits for floating-point performance on a particular system.

3-D FFT PDE (FT) Benchmark In this benchmark a 3-D partial differential equation is solved using FFTs. This kernel performs the essence of many spectral methods. It is a good test of long-distance communication performance.

Multigrid (MG) Benchmark The MG benchmark is a simplified multigrid kernel, which solves a 3-D Poisson PDE. This problem is simplified in the sense that it has constant rather than variable coefficients as in a more realistic application. This code is a good test of both short and long distance highly structured communication.

SP Simulated CFD Application (SP) Benchmark This simulated CFD application is called the scalar pentadiagonal (SP) benchmark. In this benchmark, multiple independent systems of nondiagonally dominant, scalar pentadiagonal equations are solved.

BT Simulated CFD Application (BT) Benchmark This simulated CFD application is called the block tridiagonal (BT) benchmark. In this benchmark, multiple independent systems of non-diagonally dominant, block tridiagonal equations with a 5x5 block size are solved. SP and BT are representative of computations associated with the implicit operators of CFD codes such as ARC3D at NASA Ames. SP and BT are similar in many respects, but there is a fundamental difference with respect to the communication to computation ratio.

NPB results for other systems can be found at the NAS Parallel Benchmark home page [21] and also at [22].

V. Conclusions

Summarizing we must say that both high speed interconnect technologies show significant improvements over traditional FastEthernet connections in terms of raw MPI bandwidth and latency. Comparing the fast networks among themselves ScaMPI is clearly ahead with the bandwidth from 2 to 8 times higher for messages less than 1 KB. It also shows the lowest latency for very small messages. MPI/Pro on Giganet has got its advantages above 16 KB messages.

As the results from the NAS Parallel Benchmark tests show improving overall application performance depends on many factors. The resulting application performance cannot directly be derived from the speed of the cluster interconnection. Only intensely communicating applications can benefit from a faster network. Although ScaMPI has a clear advantage in the small message range the NAS tests do not take advantage of it. Even more VIA shows better results in most of the cases. This is probably because traditional applications tend to use large messages. So in order to decide which network to chose one should know the communication behavior of the applications intended to be run on the cluster. A not less important criteria of course is the price of the entire system.

References

[1] Dolphin Interconnect Solutions AS: *PCI SCI-64 Adaptor Card, Features and Benefits*, http://www.dolphinics.com/dics_pci-sci64.htm.

[2] *The Virtual Interface Architecture Specification. Version 1.0*, Compaq, Intel and Microsoft Corporations, Dec 16, 1997, Available at http://www.viarch.org.

[3] *Dolphin Interconnect Server Cluster Test shows optimal Virtual Interface Architecture (VIA) Performance*, Dolphin Interconnect Solutions Inc., Press Release, San Jose, California, February 1998.

[4] *Dolphin Interconnect Unveils VI Architecture Roadmap*, Dolphin Interconnect Solutions Inc., Press Release, Santa Clara, California, June 1998.

[5] M. Trams, *Design of a system-friendly PCI-SCI bridge with an optimized user-interface*, Diploma Thesis, Chair of Computer Architecture, Chemnitz University of Technology, 1998.

[6] F. Seifert, *Development of system software to integrate the Virtual Interface Architecture (VIA) into the Linux operating system kernel for optimized message passing*, Diploma Thesis, Chair of Computer Architecture, Chemnitz University of Technology, October 1999.

[7] M. Trams, W. Rehm, D. Balkanski, S. Simeonov, *Memory Management in a combined VIA/SCI Hardware*, In proceedings to PC-NOW 2000, International Workshop on Personal Computer based Networks of Workstations held in conjunction with the International Parallel and Distributed Processing Symposium (IPDPS 2000), May 1-5 2000, Cancun/Mexico.

[8] Sven Schindler, Wolfgang Rehm, Carsten Dinkelmann, *An optimized MPI library for VIA/SCI cards*, In: Proceedings of the Asia-Pacific International Symposium on Cluster Computing (APSCC'2000) held in conjunction with the Fourth International Conference/Exhibition on High Performance Computing in Asia-Pacific Region (HPCAsia2000), May 14-17, 2000, Beijing, China.

[9] P. Buonadonna, A. Geweke, and D. Culler, *An Implementation and Analysis of the Virtual Interface Architecture*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, May 1998.

[10] *M-VIA: A High Performance Modular VIA for Linux*, http://www.nersc.gov/research/FTG/via.

[11] *cLAN for Linux*, http://www.giganet.com/products/indexlinux.htm.

[12] *Compaq's ServerNet II*, http://www.servernet.com.

[13] *New Fibre Channel Virtual Interface Architecture Host Bus Adapter for Windows NT Cluster Computing from Finisar Corporation*, Finisar Corporation, Press Release, Mountain View, California, November 1998.

[14] *SCALI - Scalable Linux Systems*, http://www.scali.com.

[15] textslMP-MPICH: Multi-Platform MPICH, http://www.lfbs.rwth-aachen.de/users/joachim/MP-MPICH.

[16] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, 1994, http://www.mpi-forum.org/docs.

[17] Message Passing Interface Forum, *MPI-2: Extensions to the Message-Passing Interface*, 1997, http://www.mpi-forum.org/docs/mpi-20.ps.

[18] Quinn O. Snell, Armin R. Mikler and John L. Gustafson, *NetPIPE: A Network Protocol Independent Performance Evaluator*, Ames Laboratory/Scalable Computing Lab, Ames, Iowa 50011, USA.

[19] Rossen Dimitrov and Anthony Skjellum, *Impact of Latency on Applications Performance*, MPIDC 2000.

[20] David Bailey, Tim Harris, William Saphir, Rob van der Wijngaart, Alex Woo, and Maurice Yarrow, *The NAS Parallel Benchmarks 2.0*, Report NAS-95-020, December, 1995.

[21] Subhash Saini and David H. Bailey, *NAS Parallel Benchmark(Version 1.0) Results 11-96*, Report NAS-96-18, November, 1996, http://www.nas.nasa.gov/Software/NPB.

[22] *NAS Serial Benchmark Performance at NERSC*, http://www.nersc.gov/research/FTG/pcp/performance.html.

ScaMPI	W						A						B					
	Ser.	1	2	4	8	9	Ser.	1	2	4	8	9	Ser.	1	2	4	8	9
BT	69.48	64.22		249.62		539.10	66.49			243.30		539.37				248.00		537.21
CG	32.56	31.11	50.18	88.98	151.36		32.11		50.74	94.72	195.88		22.13		53.03	88.74	199.48	
EP	1.35	1.26	2.52	4.84	9.24		1.34		2.52	5.14	10.28		1.35		2.54	5.04	10.40	
FT	48.18	38.72	63.39	119.38	216.72		50.26		62.36	118.72	248.12					125.86	238.24	
IS	8.85	5.66	9.28	15.90	23.00		5.94		8.45	15.22	26.48		5.00		8.35	14.74	25.60	
LU	90.30	82.71	196.25	381.50	685.68		84.09		154.82	307.70	714.72		81.14		153.45	304.38	583.04	
MG	63.16	49.93	94.43	157.72	304.36		52.91		86.23	156.62	349.92		56.72		92.36	166.58	375.44	
SP	55.90	50.96		206.22		441.36	55.11			190.22		400.05	53.27			183.82		421.43

MPI/PRO TCP	W						A						B					
	Ser.	1	2	4	8	9	Ser.	1	2	4	8	9	Ser.	1	2	4	8	9
BT	69.48	65.12		225.02		422.15	66.49			238.12		494.46				248.58		512.96
CG	32.56	27.27	49.12	55.12	80.44		32.11		55.23	81.04	110.68		22.13		45.42	72.36	148.56	
EP	1.35	1.31	2.55	5.08	10.20		1.34		2.54	5.10	10.24		1.35		2.56	5.10	10.24	
FT	48.18	47.15	60.75	93.06	150.64		50.26		68.65	113.22	187.72					117.04	201.36	
IS	8.85	7.10	5.78	6.68	8.92		5.94		5.38	7.18	9.92		5.00		5.38	6.62	10.36	
LU	90.30	88.44	182.69	333.40	495.88		84.09		155.99	299.12	617.12		81.14		155.54	302.50	562.56	
MG	63.16	57.69	81.60	109.86	172.36		52.91		93.22	160.94	335.68		56.72		99.63	173.80	360.56	
SP	55.90	53.07		173.86		309.65	55.11			177.78		298.80	53.27			181.46		357.71

MPI/PRO VIA	W						A						B					
	Ser.	1	2	4	8	9	Ser.	1	2	4	8	9	Ser.	1	2	4	8	9
BT	69.48			261.40			66.49			253.14						251.14		
CG	32.56		58.28	98.68	152.80		32.11		60.23	108.64	172.56		22.13		47.09	89.52	204.40	
EP	1.35		2.56	5.10	10.20		1.34		2.56	5.12	10.20		1.35		2.56	5.12	10.16	
FT	48.18		74.27	139.42	265.64		50.26		81.26	153.88	299.08						302.04	
IS	8.85		9.90	16.78	30.04		5.94		8.70	15.46	28.64		5.00		8.60	14.82	26.56	
LU	90.30		194.87	395.44	669.28		84.09		164.20	322.96	726.76		81.14		158.72	314.86	593.48	
MG	63.16		106.21	167.12	284.96		52.91		99.93	181.26	403.64		56.72		106.54	192.10	430.16	
SP	55.90			217.22			55.11			201.76			53.27			190.18		

Fig. 7. Overview of NAS Parallel Benchmark results. The numbers represent the total Mop/s rate. The "Ser." columns contain the results of the purely serial versions of the tests, while the columns labeled "1" are for result from running the parallel program on one processor.