TCP/IP versus VIA on Network of Workstations

Marcelo Lobosco, Anderson Faustino da Silva, Vítor Santos Costa e Claudio Amorim[1]
[1]Programa de Engenharia de Sistemas e Computação, COPPE, UFRJ
Centro de Tecnologia, Cidade Universitária, Rio de Janeiro, Brazil
{lobosco, faustino, vitor, amorim@cos.ufrj.br}

Abstract—

This paper evaluates the performance gains provided by VIA, an user-level communication protocol, when compared to TCP/IP, a traditional, multilayered communication protocol. To achieve this purpose we run five distinct applications using the same network card and switch, and just change the communication protocol. For all but one application, the speedup of TCP/IP was between 28% and 97% that of achieved by VIA. We further run the same applications with a hardware implementation of VIA, to evaluate the gains that a hardware implementation could offer over a software implementation. For applications with high bandwidth demand, the hardware support helped to improve performance from 20% to 114%. Surprisingly, for one application, TCP/IP performs equally to the hardware implementation.

Keywords— **VIA, User-Level Communication Protocols, TCP/IP, Communication Protocols Evaluation**

I. Introduction

In the last few years, we have seen a continuous improvement in the performance of network; both in reduced latency and in increased bandwidth. These improvements have motivated interest in the development of applications that can take advantage of parallelism in clusters of standard workstations. Often, these applications use standard message passing libraries, based in standards such as MPI [MPI]. However, not all applications have benefited from network improvements. The main reason is that the software protocol overhead has remained. Initial efforts to reduce the software overhead of traditional system networking stacks [ABB 93, BRA 95, KAY 93] were not sufficient. An alternative solution is to remove the operating system from the critical path of communication, so that the application has direct access to the network card. In this case, the operating system is just called to set up the communication and mappings required to provide the application with direct and protected access to the network interface. From this time onwards, the application can send and receive data without further operating system involvement, which contributes to reduce the communication overhead. This approach is known as user-level communication protocol.

Several academic user-level communication protocols have been developed for clusters to achieve low-latency and high-bandwidth communication [EIC 92, PAR 97, EIC 95, BLU 94]. More recently, Intel, Compaq and Microsoft have decided to standardize such user-level communication protocols, creating the Virtual Interface Architecture (VIA) [VIA]. This new protocol is being integrated with a new interconnection technology, InfiniBand [IBA].

The goal of this paper is to evaluate the performance gains provided by VIA, when compared to TCP/IP, a traditional, multilayered communication protocol. To achieve this purpose we run five distinct applications using the same network card and switch, and just change the communication protocol: Water from the SPLASH benchmark suite [SIN 92]; 3-D FFT, IS and EP from NAS benchmark [BAI 91]; and SOR, from [LU 95]. We ran SOR and Water with two different input sets: SOR internal elements of the matrix initialized to either zero (SOR-Z) or nonzero values (SOR-NZ); and Water with 288 and 1728 molecules. For all but one application, the speedup of TCP/IP was between 28% and 97% that of achieved by VIA. TCP/IP outperformed VIA in just one application, SOR, with both input sets.

We further run the applications with a hardware implementation of VIA, to evaluate the gains that a hardware implementation could offer over a software implementation. In two out seven executions, SOR-NZ and 3D-FFT, hardware support helped to improve performance from 20% to 114%. In four applications, SOR-Z, Water-1726, Water-288, and IS (without the RDMA feature) the software performance is within 6% of the hardware. And finally, for the remaining application, EP, the VIA implementation in software actually outperforms the hardware implementation by 10%. Surprisingly, for both SOR data sets, TCP/IP performs equally to the hardware implementation. We believe these results stem from inefficiencies in the VIA implementation of MPI.

This paper is organized as follows. Section 2 gives an overview of TCP/IP and the VI architecture. Section 3 presents the applications used in the study and their results. Section 4 concludes the work.

II. Communication Protocols

A. TCP/IP

TCP/IP was developed by the United States Department of Defense (DoD) research project in order to connect a

number of different networks designed by different vendors into a network of networks. The DoD designed TCP/IP to be robust and automatically recover from any node or network failure. However, the reliability provided by TCP/IP has a price, paid as overhead to the communication; to ensure reliable data transfer, protocol stack implementations like TCP/IP usually require data to be copied several times among layers and that communicating nodes exchange numerous protocol-related messages during the course of data transmission and reception.

TCP/IP is composed of IP, responsible for routing individual packet of data from node to node; and TCP, responsible for breaking up the message into datagrams, reassembling them at the other end, and putting things back in the right order. TCP is also responsible for verifying the correctness of data delivery. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.

In a typical implementation, a single data packet passes through these protocol layers. Each layer in the stack adds protocol information to the data packet. When the packet arrives at its destination, it must again pass through the protocol stack, this time in reverse.

The number of protocol layers that are traversed, data copies, context switches and interrupts contributes directly to the software overhead. Also, the multiple copies of the data that must be maintained by the sender node and intermediate nodes until receipt of the data-packet is acknowledged, contributes to reduce memory resources and further slows down transmission.

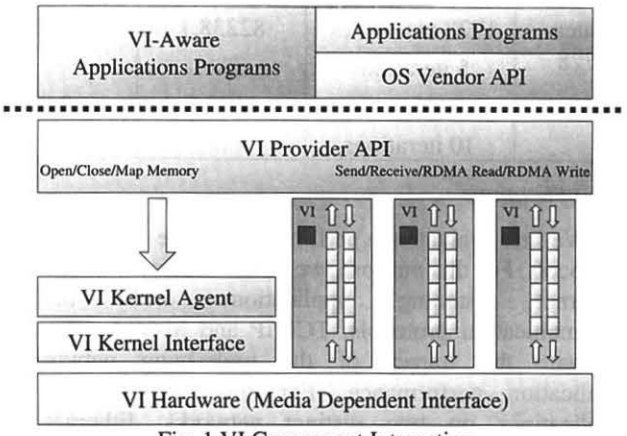

Fig. 1 VI Component Interaction

B. VI Architecture

The Virtual Interface Architecture (VIA) is a user-level memory-mapped communication architecture developed by the industry that aims to achieve low latency and high bandwidth communication within clusters of servers and workstations. The main idea is to remove the critical path of communication from the operating system kernel. The operating system is called just to control and setup the communication. Data is transferred directly to the network by the sending process and is read directly from the network by the receiving process. Even though VIA allows applications to bypass the operating system for message passing, VIA works with the operating system to protect memory so that applications use only the memory allocated to them.

The basic components of VIA as defined by the VIA Specification [VIA] are:

- VI consumer - The VI consumer is a software process that communicates through a VI. An application program, a standard operating system communication facility (such as an application program), and a VI user agent (see below) are examples of VI consumers. The VI consumer posts requests, in the form of descriptors, to the VI provider to send or receive data.
- VI provider - The VI provider consists of a physical network adapter - the Network Interface Card (NIC) - and a kernel agent, such as a device driver. The NIC implements the VIs, work queues, and completion queues. Beside this, it also performs the data transfer functions. The VI kernel agent (described below) performs the resource management necessary to maintain the VI.
- VI user agent - The VI user agent is a software component that enables an operating system communication facility to utilize a particular VI provider. The VI user agent issues commands to create, manage, and destroy VI instances under the control of the VI kernel agent. The VI user agent abstracts the details of the underlying VI NIC hardware in accordance with an interface defined by the operating system communication facility.
- VI kernel agent - The VI kernel agent is a part of the operating system that acts as a device driver for the VI NIC. It includes the kernel software necessary to register communication and manage VIs.
- Virtual interface - A virtual interface is the interface between a NIC and a process (such as a data transfer operation) that allows the VI direct access to the process' memory. The VI consists of two work queues: a send and a receive queue.
- Completion queue - A completion queue contains information about completed descriptors. It can be used to create a single point of completion notification for multiple queues.

Figure 1 depicts the interaction between VI components and the application program.

VIA supports two types of data transfers: Send-Receive, that is similar to traditional message-passing model, and Remote Direct Memory Access (RDMA), where the source and destination buffers are specified by the sender, and no receiver is required. VIA defines two RDMA operations, RDMA Write and RDMA Read, to respectively write and read remote data.

III. Performance Evaluation

A. Experimental Platform

Our experiments were performed on a cluster of 16 SMP PCs. Each PC contains two 650 MHZ Pentium III processors. For the results presented in this paper, we used just one processor on each node. Each processor has a 256 Kb L2 cache and each node has 512 Mb of main memory. All nodes run Linux 2.2.14-5.0.

Each node has two network cards: (a) Intel EtherExpress Pro 10/100; and (b) Giganet cLAN NIC. The EtherExpress Pro 10/100 card is connected to a Micronet SP624C 10/100Mbps Dual-Speed Switch. The Giganet cLAN card is connected to a Giganet cLAN 5300 switch.

To run VIA on the Intel EtherExpress Pro 10/100 card, we use M-VIA version 1.0 [MVI], a software implementation of VIA for Linux.

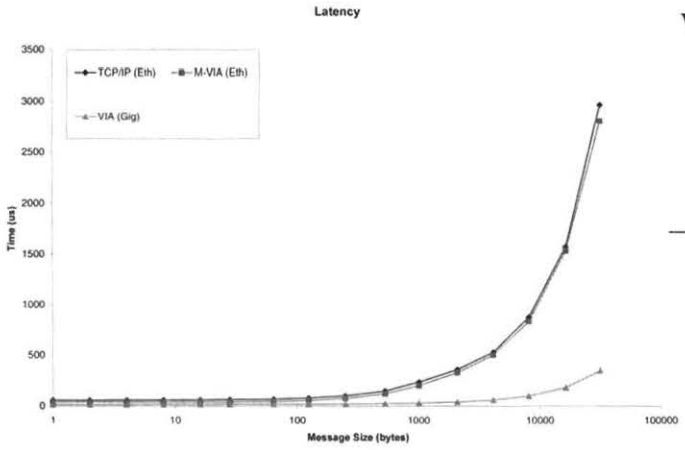

Fig. 2 – Latency

Figures 2 and 3 show the latency and bandwidth of TCP/IP and M-VIA on Intel EtherExpress Pro 10/100 card and VIA on Giganet. The figures show that the latency of M-VIA is 70% of the TCP/IP. Giganet's latency is an order of magnitude smaller than on Ethernet.

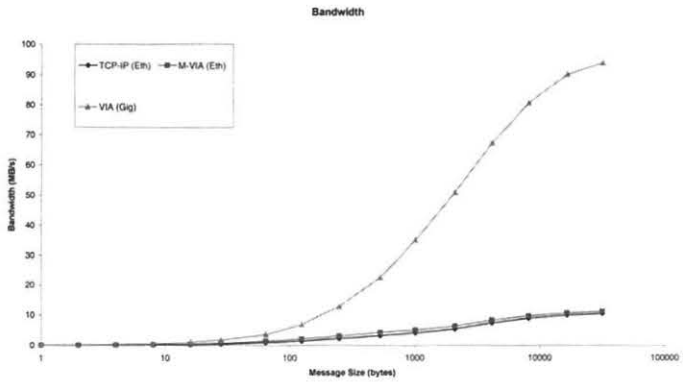

Fig. 3 – Bandwidth

TABLE I
Sequential Times of Applications

Program	Program Size	Sequential Time (msec)	Standard Deviation (%)
EP	2^{28}	433724,3	0,01
SOR-Zero	4096 X 4096, 50 iterations	48620,3	0,01
SOR-Nonzero	4096 X 4096, 50 iterations	47593,6	0,02
IS Class A	$N = 2^{23}$, $B_{max} = 2^{15}$, 10 iterations	15102,5	0,13
Water-288	288 molecules, 5 iterations	2285,4	0,07
Water-1728	1728 molecules, 5 iterations	82238,1	0,1
3-D FFT	64 x 64 x 64, 10 iterations	6155,7	0,05

B. Applications

We first evaluate the performance of the communication protocols. For this purpose, we used the same network card, Ethernet, running applications with different communication protocols, TCP/IP and M-VIA. Then, we evaluate the impact of the underlying network in application performance. For this purpose, we run applications on two distinct networks, Ethernet and Giganet. We used five applications for both experiments: Water from the SPLASH benchmark suite [SIN 92]; 3-D FFT, IS and EP from NAS benchmark [BAI 91]; and SOR, from [LU 95]. We ran SOR and Water with two different input sets: SOR internal elements of the matrix initialized to either zero or nonzero values; and Water with 288 and 1728 molecules. The parallel applications were ported to MPI [MPI] from a previous version [LU 95] that used PVM [GEI 92]. We chose MPI because it has an implementation

to VIA, called MVICH [MCH], which supports both M-VIA and Giganet. Application code is exactly the same on all possible configurations, not requiring any code adaptation.

We used MPICH 1.2.0 in our experiments for TCP/IP and MVICH 1-alpha 5 for VIA. The implementation of MVICH is in the early stages, so it is neither completely stable nor optimized for performance yet. Thus the VIA results presented here can become better with an optimized version of MVICH. We run the applications with and without RDMA (refer to section 2) for the tests with VIA. This option is enabled during the compilation of MVICH.

The performance of the applications running MPICH – so TCP/IP – on Giganet was poor. The execution times, when compared with MPICH on Ethernet, were between 1,12 and 12,4 times slower. So we do not report the results of this configuration on this paper.

TABLE II
MESSAGES AND DATA AT 8 PROCESSORS

Program	Num of Messages	Total Transfers (Kilobytes)	Md Msg Size (Kb/m)	BW per CPU (MB/s)
EP	7	0.3	0,04	0
SOR-Zero	1372	10981	8,00	0,10
SOR-NZ	1372	10981	8,00	0,14
IS Class A	2024	360453	178,09	2,76
Water-288	620	1483	2,39	0,36
Water-1728	620	8908	14,37	0,07
3-D FFT	686	39424	57,47	1,28

The sequential execution times for the applications are presented in table I. Each application was run 10 times; the times presented are an average of these values. We also present the standard deviation of the times.

Table II shows the total amount of data and messages sent by the applications, as well as the medium message size (in kilobytes per message) and the bandwidth per processor, when running on 8 nodes.

C. Results

Table III shows the application's speedups for 8 processors. More detailed results are given in figures 4 to 10. We present the speedups for 8 processors because IS and FFT did not run for 16 processors with all configurations. We also could not run IS Class A for 4 and 8 CPUs using MVICH on Ethernet with RDMA. We believe it occurred because a bug in the interface between MVICH and M-VIA, since the version on Giganet runs.

The table shows 3 classes of applications:

- Applications insensitive to both communication protocol and hardware platform. SOR performs equal or better in TCP/IP than in VIA, even with a better hardware support.
- Applications sensitive to the communication protocol but insensitive to hardware platform. EP and Water perform equal or better in Ethernet than in Giganet, when using the same communication protocol.
- Applications sensitive to both communication protocol and hardware platform. 3-D FFT and IS Class A performs better with VIA than IP. Also, Giganet is effective in improving performance.

TABLE III
SPEEDUPS – 8 PROCESSORS

Application	MPICH	MVICH			
	Eth	Eth	Eth RDMA	Gig	Gig RDMA
EP	7,48	8,48	8,48	8,31	8,29
SOR-Zero	3,66	3,15	3,23	3,59	3,52
SOR-Nonzero	4,84	3,88	3,84	4,51	3,97
Water-288	4,59	9,16	9,04	8,92	8,96
Water-1728	5,33	6,21	8,42	5,42	5,40
3-D FFT	1,63	1,68	2,42	3,61	3,47
IS Class A	0,94	3,18	NA	3,22	4,89

C.1 Applications Insensitive to Both Communication Protocol and Hardware Platform

Red-Black SOR

Red-Black Successive Over-Relaxation (SOR) is a method for solving parallel differential equations. The program divides the red and the black array into roughly equal size bands of rows, and each band is assigned to a processor. Communication occurs across the boundary rows between bands: each processor sends the boundary rows to its neighbors.

We run Red-Black SOR with a 4096 X 4096 matrix for 50 iterations. In SOR-Z, edge elements are initialized to 1 and all other to 0, whereas in SOR-NZ all elements are initialized to non-zero numbers. The results are shown on figure 4 and 5. The improvement in speedup obtained by all communication protocols in SOR-NZ is due its better load balancing, when compared to SOR-Z. Load imbalance occurs in SOR-Z because floating-point computations involving zeros take longer than those involving non-zeros, causing the processors working on the middle parts of the array to take longer between iterations [LU 95].

Surprisingly, VIA was not effective in SOR. For SOR-Z, the speedup of MVICH is 91%-97% that of MPICH. For SOR-NZ, the speedup of MVICH is 82%-101% that of

MPICH. Note that, for 2 processors, the speedup of both MVICH on Ethernet and Giganet are near linear, while the speedup of MPICH is between 1.18 and 1.24. Nevertheless, for 4 processors, the speedup of both MVICH and MPICH maintains near 2, while the speedup of MPICH grows to 2.2. We decided to examine why this occurs. The Multiprocessing Environment library (MPE) was used to profile the application. Jumpshot, a graphical visualization tool for parallel programs, was used to visualize the results. Both applications are distributed together with MPICH.

The result showed that the MPI_Recv primitive implementation on MVICH is slower than on MPICH for any number of nodes. For 2 nodes, the speedup is better because this primitive is invoked less time than for 4 and 8 nodes, so the other MPI primitives, which are faster on MVICH than on MPICH, hides the overhead to this call.

C.2 Applications Sensitive to the Communication Protocol But Insensitive to Hardware Platform.

EP

The Embarrassingly Parallel program is part of the NAS parallel benchmark suite. It evaluates an integral by means of pseudorandom trials. This kernel requires virtually no inter-process communication; the only communication is summing up an integer list at the end of the program. The processor 0 receives the lists from each processor and sums them up.

The class A problem, that generates 2^{28} pairs of random numbers, was solved in our experiments. The results are shown on figure 6. The super linear speedup achieved by all configurations is due the cache effect.

The use of VIA resulted in a small improvement in performance, since the communication is insignificant: For 16 processors, the speedup of MPICH was 88% that of MVICH on Ethernet and 97% that of MVICH on Giganet. For the same reason, the use of RDMA was not effective in improving performance.

When comparing VIA on Ethernet and Giganet, Giganet's lower latency and higher bandwidth do not contribute to reduce computation time. In fact, Giganet's speedup was 90% that of Ethernet using VIA. Again, the insignificant communication among processes is responsible by this result.

Water

Water is a N-body molecular application that evaluates forces and potentials in a system of water molecules in liquid state. The algorithm equally divides the water molecules among processors. The algorithm has two key phases: (a) the force computation phase, where a processor update forces due to the interaction of its molecules with those of other processors; and (b) the displacement computation phase, where a processor updates the displacements of its molecules based on the forces calculated in the previous phase.

In our experiments we run Water with two different data sets, 288 and 1728 molecules. The results are shown on figures 7 and 8. Again, the super linear speedup achieved by all configurations is due the cache effect. The low computation / communication ratio helps to explain the better speedup achieved by Water-1728 for all configurations, when compared to Water-288. Table III illustrates this situation. While the bandwidth per process is 0,36 MB/s in Water-288, the bandwidth in Water-1728 is 0,07 MB/s.

Again, MVICH outperformed MPICH: for Water-1728, the MPICH speedup was just 59% that MVICH on Ethernet without/without and 63% that of MVICH on Giganet with/without RDMA. For Water-288, the MPICH speedup was 47% that MVICH on Ethernet without/without and 45% that of MVICH on Giganet with/without RDMA.

For Water-1728 running on 8 CPUs, the RDMA Write support was quite effective in improving performance: both MVICH on Ethernet and Giganet performs better with the RDMA support than without it. For 16 processors, the performances of the configurations with and without RDMA are similar. This occurs because the average message size drops from 14,37 Kb when running on 8 processors to 8,02 Kb, when running on 16 CPUs. For Water-288, RDMA did not improve performance because the medium message size is too small.

When comparing VIA on Ethernet and Giganet, we see that Giganet's performed worst for Water-1728 than Ethernet, and better for Water-288 than Ethernet. The explanation is the higher bandwidth per process required by Water-288. As the bandwidth required by Water-1728 is low, the difference between Ethernet and Giganet is small (less than 5%).

Water-1728 has less data communication and should therefore achieve better performance than Water-288. We would also expect similar performance for all MVICH configurations. For smaller number of processors, 2 and 4, we did achieve quite good performance with all MVICH configurations. For larger numbers of processors, we surprisingly only found consistent good results with MVICH-RDMA on Fast Ethernet. Fast Ethernet MVICH without RDMA performs well on 16 processors, as expected, but also shows bad performance at 8 processors. Giganet MVICH also performs badly for 8 processors. For 16 it performs slightly worse than Fast Ethernet, as had been the case for EP. We believe that all these problems lie in issues with the current implementation of MVICH.

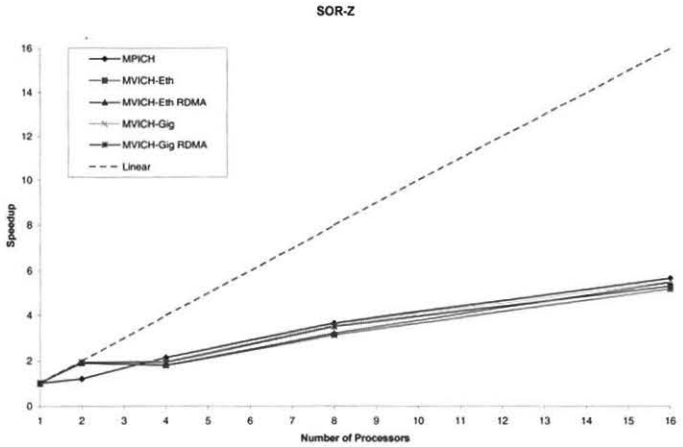

Fig. 4 - SOR-Z

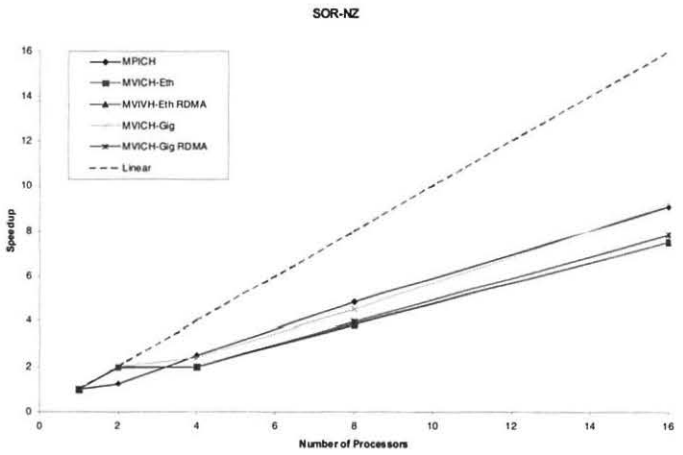

Fig. 5 - SOR-NZ

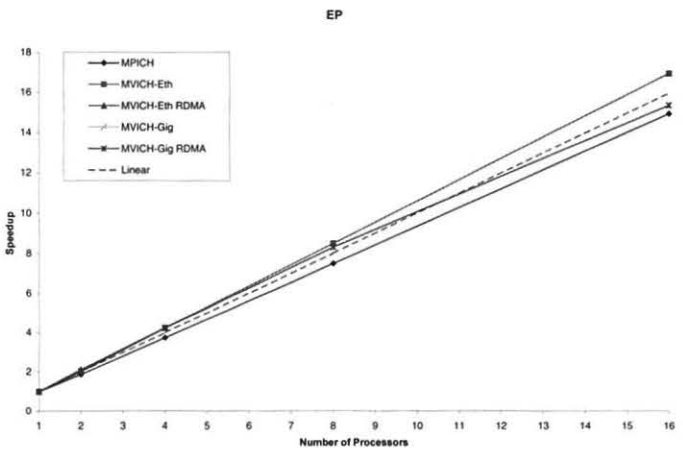

Fig. 6 – EP

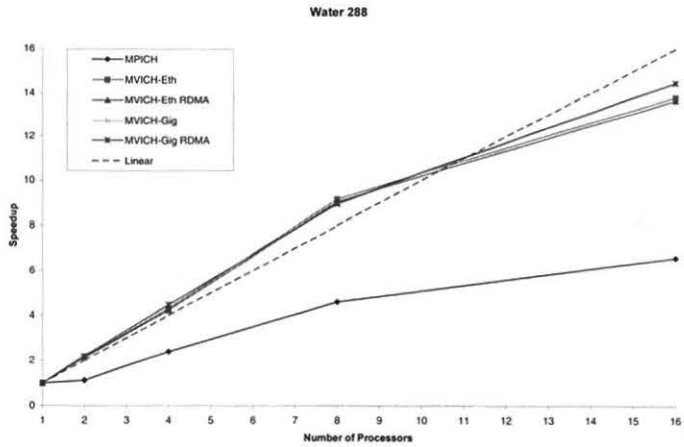

Fig. 7 – Water 288

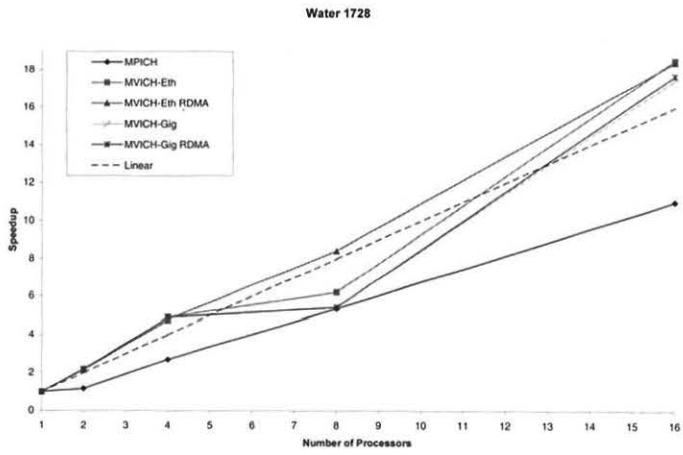

Fig. 8 – Water 1728

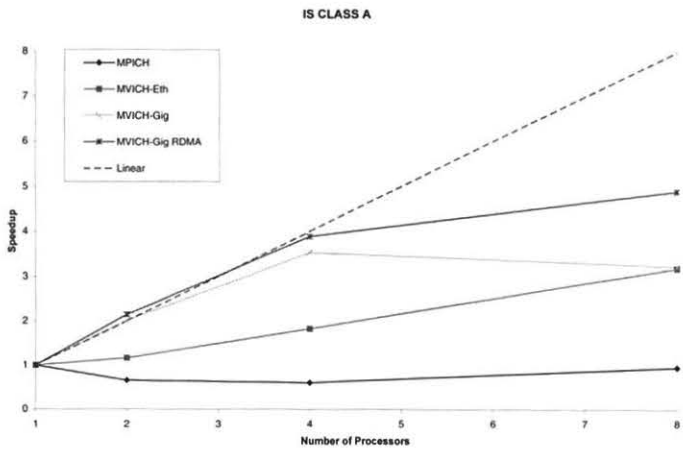

Fig. 9 – IS Class A

C.3 Applications Sensitive to Both Communication Protocol and Hardware Platform

IS

Integer Sort (IS), from the NAS benchmark, uses bucket sort to rank an unsorted sequence of keys. The algorithm divides up the keys among processors and each processor counts its keys, writing the result in a private array of buckets. After counting their keys, the processors form a chain, in which processor n send its local array of buckets to processor n+1, which adds the values received in its local array of buckets and forwards the result to the next processor. The last processor calculates the final result and broadcasts it. So, this problem requires significant data communication, as Table II shows.

In our experiments we run IS Class A, that sorts 2^{23} keys ranging from 0 to 2^{15} for 10 iterations. Recall that MVICH on Ethernet using RDMA Writing did not run. We believe this occurred because a bug in the interface between MVICH and M-VIA, since the version on Giganet runs. The results are shown on figure 9.

MVICH outperformed MPICH because of the significant data communication required by the benchmark. This is the same reason why Giganet performed better than Ethernet. IS is a communication bound application. Consequently, IS is bandwidth dependent. The MPICH speedup was just 29% that MVICH on Ethernet and 19% that of MVICH on Giganet with RDMA. The large medium message size combined with the high bandwidth requirement of IS explains way the RDMA support was effective in improving performance, when compared with the version without RDMA.

3D-FFT

3D-FFT solves a partial differential equation using three-dimensional forward and inverse FFT's. The input array A is n_1 x n_2 x n_3, organized in row-major order. The 3-D FFT first performs an n_3-point 1-D FFT on each of the n_1 x n_2 complex vectors, performing then an n_2-point 1-D FFT on each of the n_1 x n_3 vectors. Next, the resulting array is transposed into an n_2 x n_3 x n_1 complex array and an n_1-point 1-D FFT is applied to each of the n_2 x n_3 complex vectors

The computation on the array elements along the first dimension of A is distributed so that for any i, all elements Aijk $0<=j<n_2$, $0<=k<n_3$ are assigned to a single processor. No communication is needed in the first two phases, because a single processor computes each of the n_3-point FFTs or the n_2-point FFTs. The communication occurs at the transpose phase, because each processor accesses a different set of elements afterwards.

The input used in the experiment was a 64 x 64 x 64 array of double precision complex numbers for 10 iterations. Results are shown on figure 10.

Again, MVICH outperformed MPICH: the MPICH speedup was 97% that MVICH on Ethernet; 67% that of MVICH with RDMA; and 45% that of MVICH on Giganet with/without RDMA. The high bandwidth requirement, 1,28 MB/s, helps to explain why M-VIA performed better than TCP/IP, and why Giganet performed better than Ethernet.

The medium size of each message, 58 Kbytes, helps to explain why MVICH outperformed MPICH. Again, with a large amount of data to be transferred, the bandwidth becomes decisive for achieve a good performance.

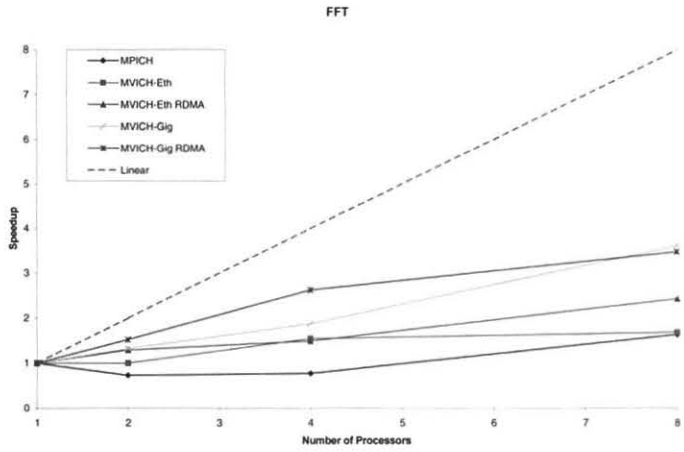

Fig. 10 – 3D-FFT

IV. CONCLUSIONS

This work presented (a) a performance evaluation of two communication protocols, TCP/IP, still widely used in cluster architectures, and VIA, a new user-level communication protocol, and (b) the impact of the underlying network in application performance. For this purpose, we run applications on two distinct networks, Ethernet and Giganet. We run five applications for all possible configurations: Water, 3-D FFT, IS, EP, and SOR. We ran SOR and Water with two different input sets: SOR internal elements of the matrix initialized to either zero or nonzero values; and Water with 288 and 1728 molecules. The performance of the applications running MPICH on Giganet was poor, so we do not report the results of this configuration on this paper.

Our experiments identified 3 distinct classes of applications:

- Applications insensitive to both communication protocol and hardware platform. For SOR, TCP/IP performed between 3% and 20% better than VIA and equally to the hardware implementation. This occurs because

of the MPI_Recv primitive implementation on MVICH that is slower than on MPICH for any number of nodes. For 2 nodes, the speedup is better because this primitive is invoked less time than for 4 and 8 nodes, so the other MPI primitives, which are faster on MVICH than on MPICH, hides the overhead to this call.

- Applications sensitive to the communication protocol but insensitive to hardware platform. EP and Water perform equal or better in Ethernet than in Giganet, when using the same communication protocol. For Water, the hardware support was not effective in improving performance because the data sets were too small, so the applications do not benefit from a better hardware support. For EP this occurs because bandwidth is quite small, thus after removing TCP/IP stack latency there is little room for the application to benefit from a faster network.
- Applications sensitive to both communication protocol and hardware platform. 3-D FFT and IS Class A performs better with VIA than IP. Also, Giganet is effective in improving performance. This occurred because of the high communication bandwidth required by these applications.

And finally, we showed that the RDMA support for VIA tends to be useful just when the medium message size of application is large.

ACKNOWLEDGMENTS

We would like to thank Sandhya Dwarkadas from Rice University for giving us the source code of applications used in this paper. We would also thank Wagner Meira from UFMG and Raquel Pinto and Lauro Whately from UFRJ for their helpful suggestions.

REFERENCES

[ABB 93] ABBOT, M.; PETERSON, L. *Increasing Network Throughput by Integrating Protocol Layers.* IEEE/ACM Transactions on Networking. Vol. 1, (no.5), Oct. 1993, pp 600-610.

[BRA 95] BRAUN, T.; DIOT, C. *Protocol implementation using integrated layer processing.* In ACM SIGCOMM '95, Cambridge, MA, USA, August. 1995.

[KAY 93] KAY, J.; PASQUALE, J. *The Importance of Non-Data Touching Processing Overheads in TCP/IP.* Computer Communication Review, vol. 23, (no. 4), Oct. 1993, pp. 259-268.

[EIC 92] EICKEN, T. von; CULLER, D.; GOLDSTEIN, S.; SCHAUSER, K. *Active messages: a Mechanism for Integrated Communication and Computation.* In Proceedings of the 19th Annual International Symposium on Computer Architecture, Gold Coast, Australia, May 1992, pp.256-266.

[PAR 97] PAKIN, S.; KARAMCHETI, V.; CHIEN, A. *Fast messages: efficient, portable communication for workstation clusters and MPPs.* IEEE Concurrency, vol.5, (no.2), April-June 1997, pp.60-72.

[EIC 95] EICKEN, T. von; BASU, A.; BUCH, V.; VOGELS, W. *U-Net: A User-level Network Interface of parallel and Distributed Computing.* In Proc. of the 15th ACM Symposium of Operating Systems Principles, vol. 29, (no.5), December 1995, pp. 40-53.

[BLU 94] BLUMRICH, M. et al. *Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer.* In Proceedings of the 21st International Symposium on Computer Architecture, April 1994, pp. 142-153.

[LU 95] LU, H.; DVARKADAS, S.; COX, A.; ZWAENEPOEL, W. *Message Passing Versus Distributed Shared Memory on Networks of Workstations.* In Supercomputing '95, 1995.

[VIA] Compaq, Intel, and Microsoft. *Virtual Interface Architecture Specification, Version 1.0.* Available at http://www.viarch.org.

[IBA] Compaq, Dell, Hewlett-Packard, IBM, Intel, Microsoft and Sun Microsystems. *InfiniBand Trade Association.* http://www.infinibandta.org

[SIN 92] SINGH, J.; WEBER, W.; GUPTA, A. *SPLASH: Stanford Parallel Applications for Shared-Memory.* Computer Architecture News, 20(1):2-12, March 1992.

[BAI 91] BAILEY, D.; BARTON, J.; LASINSKI, T.; SIMON, H. The NAS Parallel Benchmarks. Technical Report 103863, NASA, July 1993.

[MVI] National Energy Research Scientific Computing Center. *M-VIA: A High Performance Modular VIA for Linux.* http://www.nersc.gov/research/FTG/via/.

[MCH] National Energy Research Scientific Computing Center. *MVICH: MPI for Virtual Interface Architecture.* http://www.nersc.gov/research/FTG/mvich.

[GIG] Giganet Inc. http://www.giganet.com/

[MPI] Message Passing Interface Forum. http://www.mpi-forum.org/.

[GEI 92] GEIST, G.; SUNDERAM, V. *Network based concurrent computing on the PVM system.* Concurrency: Practice and Experience, June 1992, pp. 293-311.