A VIA support for the LAM/MPI suite[⋆]

Massimo Bertozzi, Marco Panella, and Monica Reggiani

Dipartimento di Ingegneria dell'Informazione
Università di Parma, I-43100 Parma, Italy
{bertozzi,panella,reggiani}@ce.unipr.it

The increasing use of System Area Network (SAN) to interconnect cluster's nodes, for a number of applications ranging from database to numerical computation, has boosted interest in network research. The most used network protocol (TCP/IP) is unable to take fully advantage of SAN features such as low error rates or more performant network hardware. Recently, a consortium leaded by Microsoft, Compaq and Intel authored a new standard, the Virtual Interface Architecture (VIA) specifically designed for fast interprocess communication in SAN environment [1]. The main goals of VIA design are: communication operations featuring low overhead, I/O operations without interrupts, no unnecessary context switches as well as copies of data, no operating system intervention for managing concurrency in communications.

To obtain these targets VIA model eliminates kernel intervention during data transfers. In the VIA model, the *kernel agent* performs the setup and manages the resources needed for communications between processes, thus not taking charge of the actual transmission of data. Each process (*VI consumer*) is provided with the exclusive use of a *virtual interface* (VI) for each other process with which it needs to exchange data. The VIs are featured by two queues, one for sending and one for receiving data. When a process exchanges data with another, it puts a structure describing the required operation (*d*escriptor) on the top of the appropriate VI queue and acquaints the VI Network Interface Card (NIC) with the presence of new data in the queue, writing in a specific register (the *doorbell*). At this point, VI NIC takes charge of the communication and, since it can access directly to process data unnecessary memory transfers are avoided. When data transfer is over, the VI NIC marks the related descriptor, therefore enabling receiving process to access to data. The use of these memory mapped registers allows the transfer to proceed without the use of interrupts, that represent a major penalty for modern super-scalar CPUs [2].

Although the concepts behind the VIA standard are not new [3–8], its main advantage is that it has been carefully designed to be operating system and processing architecture independent, therefore sharing a significant goal with the MPI standard [9] and allowing programmers to write fully portable applications.

In order to design the communication suitable for MPI, VIA provides two different transfer model: Send/Receive and Remote Direct Memory Access (RDMA) [10]. Nonetheless, the plain use of these communication models provided by VIA does not allow a complete implementation of MPI based on VIA, due to the large number of MPI communication flavors (synchronous, asynchronous, blocking, non-blocking, point to point, collective,...) [9]. In fact, VIA Send and Receive operations require to be synchronous: the receiver must initiate the communication before the delivery of sender's data; otherwise the communication will fail. Conversely, in RDMA communication model the receiver process is not in charge of any operation during the transfer and no notification is given to the remote node about completion of request, therefore following a pure asynchronous model. In addition, VIA offers a strict point-to-point model, in which every VI is connected only to another VI. Thus the implementation of a blocking receive from any source requires querying multiple VIs. Therefore, a combination of VIA instructions has been used to develop a protocol supporting the whole set of MPI communication features. This protocol has been inserted in the LAM/MPI

[⋆] The work described in this paper has been carried out under the financial support of the Italian *Ministero dell'Università e della Ricerca Scientifica e Tecnologica* (*MURST*) in the framework of the MOSAICO (Design Methodologies and Tools of High Performance Systems for Distributed Applications) Project.

Message size (byte)	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768
LAM remote on TCP/IP (μs)	114	114	114	116	116	120	126	138	163	213	310	426	609	996	1743	3219
LAM remote on M-VIA (μs)	63	64	64	66	67	69	77	88	120	169	271	401	590	975	1756	3277
LAM local on TCP/IP (μs)	63	63	63	63	63	63	63	64	65	67	72	81	117	175	305	529
LAM local on M-VIA (μs)	10	10	10	10	10	10	11	12	18	21	27	32	49	106	210	407

Table 1. Transmission times for LAM/MPI on VIA or TCP/IP.

library; it allows straightforward implementation of basic MPI functions, but in order to support all the MPI functions, it has been enhanced with some features [10].

The ParMa^2 cluster of PCs has been used as testbed; it is composed of four dual 450 MHz Pentium II PCs with 256 Mbyte RAM each interconnected using a switched full-duplex Fast Ethernet network. In absence of a hardware VIA implementation, M-VIA, a VIA software emulation, has been used [11]; thanks to the simplicity of VIA protocol and to the ability of M-VIA in using specific features of supported network adapters, latency of LAM-MPI on VIA is smaller than the one of LAM-MPI on TCP/IP even if no VIA hardware has been exploited.

In order to evaluate performance, we used a common *ping-pong* test involving two processes, on a single node (LAM local in table 1) and on different nodes (LAM remote in table 1) of the ParMa^2.

VIA succeeds in dropping down latency, which features 45% reduction with LAM/MPI on VIA with respect to the same library with TCP/IP even if Fast Ethernet adapters are used instead of actual VIA interfaces. This ratio is much higher on local communication, in which kernel intervention represents the greatest part of overhead. However, the absolute latency decrease obtained using VIA is similar in local and remote communications, so we can estimate as 50μs the overhead due to operating system intervention on ParMa^2 nodes.

Anyway for larger messages, transmission times are closer, and for 16k message MPI on VIA messages takes more transmission time than TCP ones. In fact our implementation introduce a memory copy that becomes significant and reduces the advantages of using VIA. Different solutions to this drawback are currently under evaluation.

The prototypal LAM/MPI implementation over VIA is freely downloadable as a patch for LAM version 6.3.2 from web page: `http://www.ce.unipr.it/pardis/parma2/via/via.html`.

References

1. D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A. M. Merrit, E. Gronke, and C. Dodd, "The Virtual Interface Architecture," *IEEE Micro*, vol. 18, pp. 66–69, Mar.–Apr. 1998.
2. J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., 1990.
3. T. von Eicken and W. Vogels, "Evolution of the Virtual Interface Architecture," *IEEE Computer*, vol. 31, pp. 61–68, Nov. 1998.
4. T. von Eicken, A. Basu, V. Buch, and W. Vogels, "U-Net: a user-level network interface for parallel and distributed computing," *Operating Systems Review*, vol. 29, pp. 40–53, Dec. 1995.
5. T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser, "Active Messages: A mechanism for integrated communication and computation," in *Procs. 19th Symp. Computer Architecture*, (Gold Coast, Qnd. Australia), May 1992.

6. S. Pakin, V. Karamcheti, and A. A. Chien, “Fast Messages: efficient, portable communication for workstation clusters and MPPs,” *IEEE Concurrency*, vol. 5, p. 90, Apr. 1997.
7. M. A. Blumrich, K. Li, R. D. Alpert, C. Dubnicki, E. W. Felten, and J. Sandberg, “Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer,” in *Procs. International Symposium on Computer Architecture‘94*, pp. 142–153, 1994.
8. L. Prylli and B. Tourancheau, “BIP: A New Protocol Designed for High Performance Networking on Myrinet,” *Lecture Notes in Computer Science*, vol. 1388, pp. 472–485, Mar. 1998.
9. MPI Forum, “MPI A Message Passing Interface Standard,” tech. rep., University of Tennessee, June 1995.
10. M. Bertozzi, F. Boselli, G. Conte, and M. Reggiani, “An MPI Implementation on the Top of the Virtual Interface Architecture,” *Lecture Notes in Computer Science*, vol. 1697, pp. 199–206, Sept. 1999.
11. National Energy Research Scientific Computer Center, “M-via: A high performance modular via for linux.” `http://www.nersc.gov/research/FTG/via/` .