Fast Barrier Synchronization for InfiniBand™

Torsten Hoefler, Torsten Mehlan, Frank Mietke and Wolfgang Rehm

Chemnitz University of Technology
Dept. of Computer Science
Chemnitz, 09107 GERMANY
{htor, tome, mief, rehm}@informatik.tu-chemnitz.de

Abstract

The MPI_Barrier() call can be crucial for several applications and has been target of different optimizations since several decades. The best solution to the barrier problem scales with $O(log_2 N)$ and uses the dissemination principle. A new method using an enhanced dissemination principle and inherent network parallelism will be demonstrated in this paper. The new approach was able to speedup the barrier performance by 40% in relation to the best published algorithm. It is shown that it is possible to leverage the inherent hardware parallelism inside the InfiniBand™ network to lower the latency of the MPI_Barrier() operation without additional costs. The principle of sending multiple messages in (pseudo-) parallel can be implemented into a well known algorithm to decrease the number of rounds and speed the overall operation up.

1 Introduction

InfiniBand™ network is an emerging network technology in the HPC sector. More and more clusters utilize the features of this special interconnect, which include very high bandwidth, remote direct memory access and host offloading features. Several implementations exist to map these features to the standardized MPI API. One of these MPI operations is the MPI_Barrier() call which synchronizes all participating nodes. This operation can be quite important for the overall running time of several applications and has been target of optimizations for the last decades. Our goal is to lower the latency of the MPI_BARRIER operation over the InfiniBand™ network. We leverage implicit parallelism of todays InfiniBand™ adapters to speed the barrier operation up.

The next section enumerates the work which has been done in this field before. Section 2 describes a new approach to model the InfiniBand™ network and the differences to well established models like models of the LogP family [4, 2, 16] followed by the working principle for a new dissemination algorithm which leverages network parallelism. Section 4 explains several implementation details and provides performance data and comparison. The last section draws conclusions and shows perspectives for future work on this field.

1.1 Related Work

Several barrier implementations and algorithms have been developed and could be used to perform the MPI_Barrier() operation over the InfiniBand™ network. These include the Central Counter approach [5, 7], several Tree Based barriers as the Combining Tree Barrier [22], the MCS Barrier [17], the Tournament Barrier [9], the BST Barrier [20] and butterfly or dissemination based barriers [3, 9]. All these different approaches have been compared in [12] with the result that the dissemination algorithm is the most promising algorithm for cluster networks. Several studies have also been made to find special barrier solutions, either with additional hardware [18, 10] or with programmable Network Interface Cards [23]. Our approach uses only the InfiniBand™ network which does not offer programmable NIC support or special hardware barrier features. A similar study has been done by Panda et. al. in [14] and implemented in MVAPICH. Our study shows that the use of implicit parallelism of the InfiniBand™ network and maybe also other offloading based networks can lower the barrier latency significantly.

2 A new InfiniBand™ Model

2.1 Benchmarking the Network Capabilities

We benchmarked the InfiniBand™ network with and 1:P and P:1 ping-pong benchmark to evaluate the current barrier algorithms. The benchmark measures all InfiniBand™ transport types with one-byte packets (which fulfil the task of notificating the other nodes) and the following communication scheme:

- node 0 sends to node $1 \ldots P$
- each node $1 \ldots P$ waits for the reception of a message and sends the message immediately back to node 0 (1:P ping-pong)

The scheme is depicted in figure 1 and the respective Round Trip Time is denoted as $RTT(P)$.

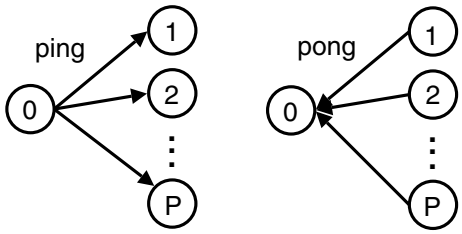

Figure 1. 1:P Ping-Pong Scheme

The benchmark was carried out at different InfiniBand™ based clusters and the results of the biggest system, the Mozart Cluster, located at the University of Stuttgart, Germany are presented in this work. It is populated with 64 nodes and therewith the biggest InfiniBand™ cluster which was used to verify the results of this paper. A single node offers the following configuration:

- Processor: 2x3GHz Xeon
- OS: Red Hat Linux release 9 (Shrike), Kernel: 2.4.27 SMP
- HCA: Mellanox "Cougar" (MTPB 23108)

The nodes are interconnected with a 64 port Mellanox InfiniBand™ MTS 9600 switch and Gigabit Ethernet. It has to be mentioned, that the retrieved benchmark ($RTT(P)$) curves are of the same shape for all probed cluster systems.

The benchmark is designed for all transport types of InfiniBand™ but we describe only the most promising one (with the lowest latencies), which is RDMA-Write, due to space restrictions. Figure 2 shows the minimal and average RDMA-Write $RTT(P)$ measurements for 100 consecutive inline and normal send operations.

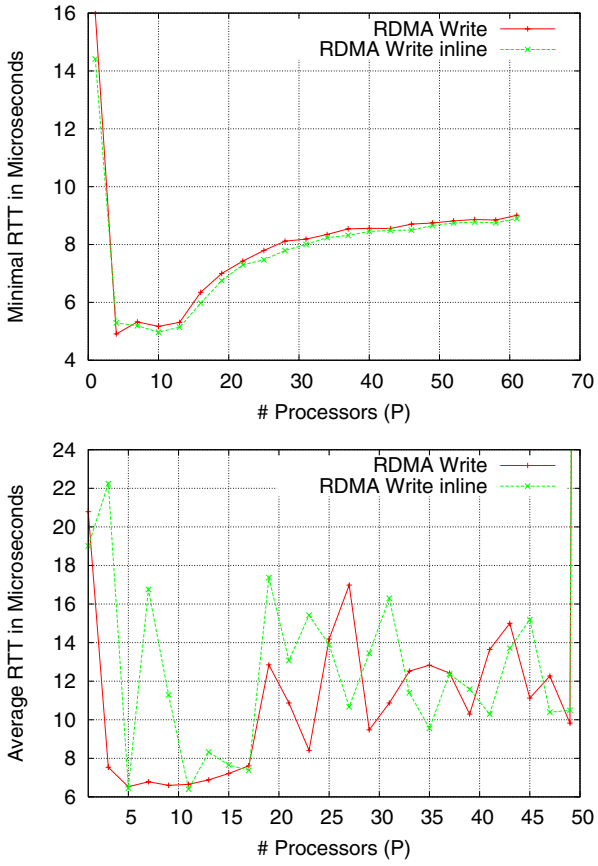

Figure 2. Minimal and Average RDMA-Write $RTT(P)$ Times

It is easy to recognize that the minimal results fit exactly into a single curve but the average values vary extremely. This variation is due to memory congestion inside the North-Bridge, where the CPU and the HCA concur for the memory (the benchmark loops on the memory to test if the data was received).

The main perception of this benchmark is that it is really cheap to send multiple messages in parallel (the price per message transfer is much lower for 4 simultaneous messages as for 1). This observation will be used in the following to enhance the barrier performance for the InfiniBand™ network.

2.2 The Prediction - LogP vs. LoP

The LogP Model, designed by Culler et al. in 1993 was developed to model the behavior of communica-

tion networks. It shows different aspects of the underlying network for coarse grained machines which were mainly a collection of loosely coupled computers (cmp. Network of Workstations). It consist of the four parameters L, o, g and P. L denotes the pure hardware Latency (the maximum for all interfaces), o models the CPU overhead and can be subdivided into o_s for the sender's overhead and o_r for the receiver's overhead. g is the gap to wait between consecutive message sends or receives, which is essentially a bandwidth limitation $bandwidth \sim \frac{1}{g}$. P denotes the number of processors involved in the algorithm. The whole LogP model is linear, which means that is is independent of the number of messages sent over the network before. A LogP prediction curve for $RTT(P)$ normed by P is shown in Figure 3. The LogP model was proven to be quite

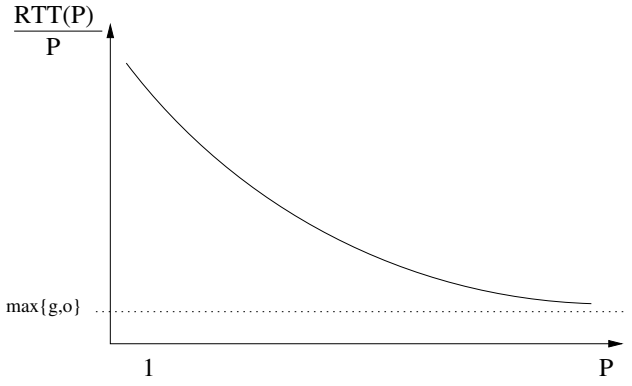

Figure 3. LogP predicted Curve

accurate for small messages on traditional networks as for example Ethernet based architectures [11]. But it seems quite inaccurate for the InfiniBand^{TM} network. The main disadvantage in this context is that it does not model the architectural specialties of modern interconnects, such as RDMA, offloading or Multicast, which can be used to speed-up algorithms on the network. This problem is assumed to be not InfiniBand^{TM} specific, also other offloading based network which offer parallelism in the processing of messages (the NIC is an active element, and not all message processing is done at the host CPU) should not be accurate within the LogP model. Thus, we shortly introduce a new model which could be more accurate for offloading based networks such as InfiniBand^{TM} or Quadrics. The LoP model bases on the LogP model because the parameters and effects which are modelled remain the same, only their dependency of the number of sent messages (the number of addressed hosts in our particular case for the barrier) is changed.

2.3 The LoP Model for InfiniBand^{TM}

The LoP model was introduced in [13] and will be shortly explained in the following. The basic assumption is that the inherent hardware parallelism in the InfiniBand^{TM} network (as defined in the standard [1]) cannot be modeled with a linear model like the LogP model. Thus the overhead and the Latency of each message depends on the history of the interface (mainly the number of messages sent before). To simplify this assumption, only the number of messages which are posted successively is evaluated. The linear LogP model changes to the non-linear LoP model, where each parameter depends on the number of messages to be sent. The L changes to $L(P)$ and the o to $o(P)$ both functions are explained in the following. $L(P)$ can be measured indirectly with the $RTT(P)$ of the system.

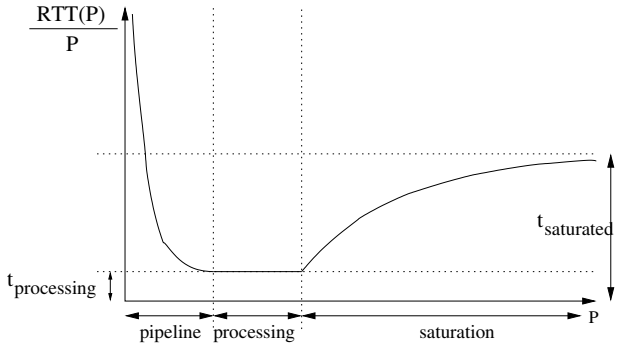

Figure 4. The $RTT(P)$ Model

The model is depicted in figure 4 where a possible explanation of its behavior is given.

For a more detailed description, a parametrized analytic model or runtime prediction refer to [13]. This model should be accurate for all offloading based networks because it models their architectural details. This shows that our barrier algorithm should also be beneficial on other networks than InfiniBand^{TM} . The main idea is to leverage this implicit hardware parallelism (sending 2 messages consecutively is significantly faster than sending two "single" messages with some time between them). This can be done by changing the barrier algorithm to send multiple consecutive messages. This idea leads us to the design of the n-way dissemination barrier. Due to the assumed generality of the model, this approach should increase the barrier performance on different modern offloading based networks architectures.

3 The n-way Dissemination Principle

The Dissemination Barrier, proposed by Hengsen et al. in 1988 [9] was proven to be the best barrier solution for single-port LogP compliant system [11]. The n-way dissemination algorithm is a generalization of the dissemination principle for multi-port networks and can be proven to be optimal for this task. The main change is the additional parameter n which defines the number of communication partners in each round to leverage the hardware parallelism. The n-parameter should refer to the number of messages which can be sent in parallel. The InfiniBand™ network, and probably more offloading based networks do not offer this parallelism explicitely but an implicit parallelism is implemented due to the hardware design. This means that the n-way dissemination barrier can use this parallelism to speed the barrier operation up. The original algorithm typifies the 1-way Dissemination Barrier ($n = 1$) in this context.

The algorithm is described shortly in the following:

Every node p sends n packets to notify n other nodes that it reached its barrier function in each round and waits for the notification of n other nodes. Subsequently, at the beginning of a new round r, node p calculates all its peer nodes (the sendpeer - $speer_i$ and the receive peer - $rpeer_i$, $\{i \in \mathbb{N}; 0 < i \leq n\}$) as follows:

$$speer_i = (p + i \cdot (n+1)^r) \ mod \ P \quad (1)$$

whereby P is the number of nodes participating in the barrier. The peers to receive from are also determined each round:

$$rpeer_i = (p - i \cdot (n+1)^r) \ mod \ P \quad (2)$$

For the original algorithm ($n = 1$), the peer calculation gives the same rules as stated in the original paper [9].

$$speer = (p + 2^r) \ mod \ P \quad (3)$$
$$rpeer = (p - 2^r) \ mod \ P \quad (4)$$

An example for $n = 2$ and $P = 9$ is given in figure 5.

A possible pseudo-code for a RDMA based implementation (e.g. InfiniBand™) is given in listing 1.

```
// parameters (given by environment)
set n = 2 // parameter
set P = number of participating processors
set rank = my local id
// phase 1 - initialization (only once)
// the barrier counter - avoid race conditions
set x = 0
reserve array with P entries as shared
for i in 0..P-1 do
   set array[i] = 0
forend
// barrier - done for every barrier
set round = -1
set x = x + 1
// repeat log_n(P) times
repeat
   set round = round + 1

   for i in 1..n do
      set sendpeer = (rank + i*(n+1)^round) mod P
      set array[rank] in node sendpeer to x
   forend
   for i in 1..n do
      set recvpeer = (rank - i*(n+1)^round) mod P
      wait until array[recvpeer] >= x
   forend
until round = ceil(log(P)/log(n))
```

Listing 1. Pseudocode for the n-way Dissemination Barrier

4 Implementation Details

The n-way dissemination barrier is implemented as an Open MPI collective component. It uses the the Mellanox VERBS API to communicate directly with the InfiniBand™ hardware. The general Open MPI framework and the component framework is introduced in [6, 19]. The collective framework offers space to implement MPI collective routines. The *ibbarr* component is an optimized MPI_BARRIER implementation for the InfiniBand™ architecture. A caching strategy precomputes the communication partners for each round in advance (during the communicator initialization) to reduce the amount of calculation during the critical MPI_BARRIER path.

4.1 Benchmark Results

Different MPI implementations for InfiniBand™ have been taken and compared with regards to their MPI_BARRIER latency to evaluate the new approach. The results show that the Open MPI *ibbarr* component is in faster than every Open Source MPI implementation, even faster as the current leader MVAPICH [15] where much research has been done to enhance the barrier performance of InfiniBand™ [8, 14]. The 1-way dissemination barrier is already faster than the dissemination barrier of MVAPICH. This can be explained with the precomputed communication partners and the lower latency of the Open MPI framework. The results are shown in figure 6 and show that the optimized n-way dissemination algorithm can be up to 40% better than the fastest implementation in MVAPICH. The performance gain from the $n > 1$ parameter

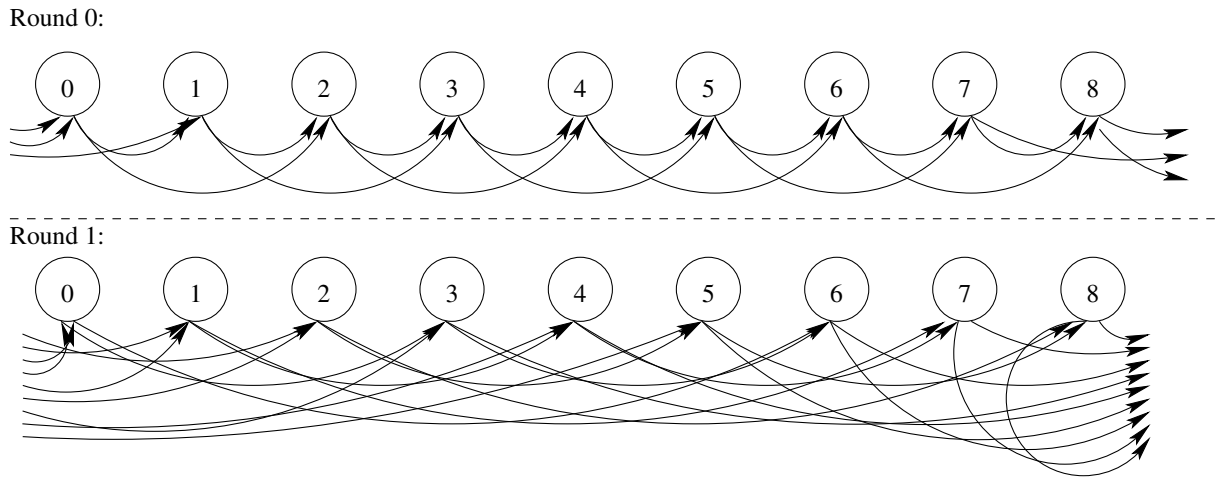

Figure 5. Example of the 2-way Dissemination Barrier

can be seen between the IBBARR-1 and the IBBARR-$n > 1$ graphs. The gap increases with the node count, but if the n parameter is too big, the memory congestion effects on the receiver side increase the latency. Thus, the task to deduce the optimal n-parameter for InfiniBand™ is not easy and will be done in a self-tuned fashion (cp. [21]), where different n parameters are benchmarked during communicator initialization and the best one is chosen for the MPI_BARRIER.

Figure 7 shows the relative speedup of the automated tuned implementation compared to the RDMA-optimized implementation in MVAPICH.

5 Conclusions

It was shown that it is possible to leverage the inherent hardware parallelism inside the InfiniBand™ network to lower the latency of the MPI_BARRIER operation with no additional costs. The principle of sending multiple messages in (pseudo-) parallel could be added to a well known algorithm to decrease the number of rounds and speed the overall operation up. The underlying principle is very simple, but it is extremely hard to model the architecture because indeterminate memory congestion delays the RDMA-Write operation and cannot be modeled precisely. Thus, we use automated tuning to choose the n-parameter. However, the barrier operation could be enhanced up to 40% on a 64 nodes cluster in comparison to the also well tuned implementation of MVAPICH, and the gap is expected to widen for larger node numbers. The precaching technique to calculate the communication partners in advance and the reduced latency of the Open MPI framework were

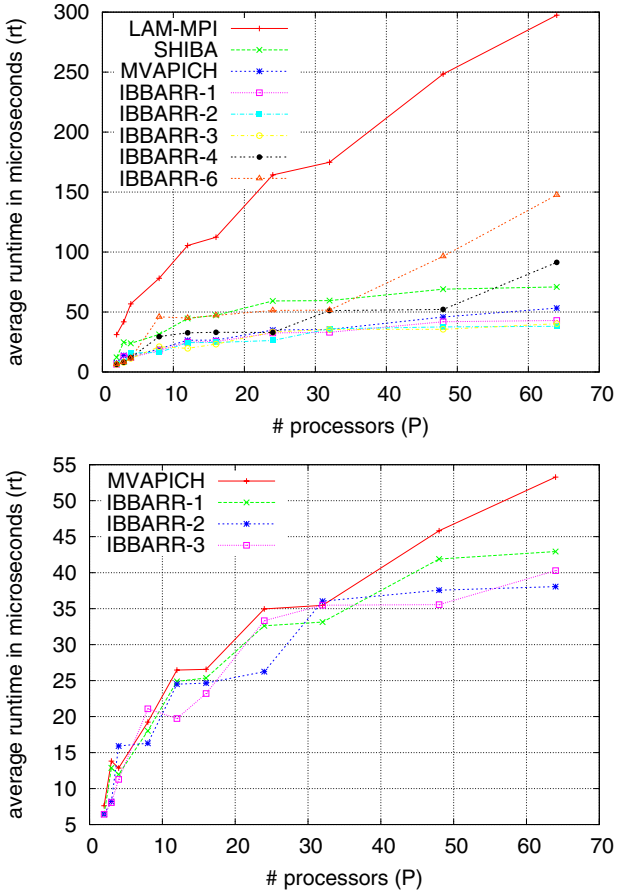

Figure 6. Comparison of different MPI_BARRIER Implementations

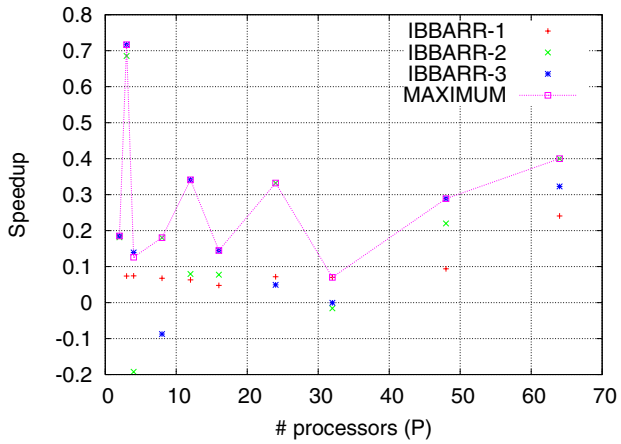

Figure 7. Relative Speedup to MVAPICH

also beneficial to reduce the latency of the standard (1-way) dissemination algorithm.

References

[1] *Infiniband Architecture Specification Volume 1, Release 1.2*.

[2] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating Long Messages into the LogP Model. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1995.

[3] E. D. Brooks. The Butterfly Barrier. *International Journal of Parallel Programming*, 15(4):295–307, 1986.

[4] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: towards a realistic model of parallel computation. In *Principles Practice of Parallel Programming*, pages 1–12, 1993.

[5] E. Freudenthal and A. Gottlieb. Process Coordination with Fetch-and-Increment. In *ASPLOS-IV: Proceedings of the fourth international conference on Architectural support for programming languages and operating systems*, pages 260–268. ACM Press, 1991.

[6] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004.

[7] J. R. Goodman, M. K. Vernon, and P. J. Woest. Efficient Synchronization Primitives for Large-Scale Cache-Coherent Multiprocessors. *SIGARCH Comput. Archit. News*, 17(2):64–75, 1989.

[8] R. Gupta, V. Tipparaju, J. Nieplocha, and D. Panda. Efficient Barrier using Remote Memory Operations on VIA-Based Clusters. In *2002 IEEE International Conference on Cluster Computing (CLUSTER 2002)*, page 83. IEEE Computer Society, 2002.

[9] D. Hengsen, R. Finkel, and U. Manber. Two Algorithms for Barrier Synchronization. *Int. J. Parallel Program.*, 17(1):1–17, 1988.

[10] T. Hoefler. Evaluation of publicly available Barrier-Algorithms and Improvement of the Barrier-Operation for large-scale Cluster-Systems with special Attention on InfiniBand^{TM}Networks. Master's thesis, TU-Chemnitz, 2004. url: http://archiv.tu-chemnitz.de/pub/2005/0073/data/diploma.pdf.

[11] T. Hoefler, L. Cerquetti, T. Mehlan, F. Mietke, and W. Rehm. A practical Approach to the Rating of Barrier Algorithms using the LogP Model and Open MPI. In *Proceedings of the 2005 International Conference on Parallel Processing Workshops*, pages 562–569, June 2005.

[12] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. A Survey of Barrier Algorithms for Coarse Grained Supercomputers. *Chemnitzer Informatik Berichte - CSR-04-03*, 2004. url: http://archiv.tu-chemnitz.de/pub/2005/0074/data/CSR-04-03.pdf.

[13] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. A Communication Model for Small Messages with InfiniBand. *PARS Proceedings*, 2005.

[14] S. P. Kini, J. Liu, J. Wu, P. Wyckoff, and D. K. Panda. Fast and scalable barrier using rdma and multicast mechanisms for infiniband-based clusters. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface,10th European PVM/MPI Users' Group Meeting, Venice, Italy, September 29 - October 2, 2003, Proceedings*, pages 369–378, 2003.

[15] J. Liu, W. Jiang, P. Wyckoff, D. K. Panda, D. Ashton, D. Buntinas, W. Gropp, and B. Toonen. Design and Implementation of MPICH2 over InfiniBand with RDMA Support. In *Int'l Parallel and Distributed Processing Symposium, Proceedings*, 2004.

[16] C. A. Moritz and M. I. Frank. LoGPC: Modelling Network Contention in Message-Passing Programs. *IEEE Transactions on Parallel and Distributed Systems*, 12(4):404, 2001.

[17] M. L. Scott and J. M. Mellor-Crummey. Fast, contention-free combining tree barriers for shared-memory multiprocessors. *Int. J. Parallel Program.*, 22(4):449–481, 1994.

[18] R. Sivaram, C. B. Stunkel, and D. K. Panda. A reliable hardware barrier synchronization scheme. In *11th International Parallel Processing Symposium (IPPS '97), 1-5 April 1997, Geneva, Switzerland, Proceedings*, pages 274–280. IEEE Computer Society, 1997.

[19] J. M. Squyres and A. Lumsdaine. The Component Architecture of Open MPI: Enabling Third-Party Collective Algorithms. In *Proceedings, 18th ACM International Conference on Supercomputing, Workshop on Component Models and Systems for Grid Applications*, St. Malo, France, July 2004.

[20] N.-F. Tzeng and A. Kongmunvattana. Distributed Shared Memory Systems with Improved Barrier Synchronization and Data Transfer. In *ICS '97: Proceedings of the 11th international conference on Supercomputing*, pages 148–155. ACM Press, 1997.

[21] S. S. Vadhiyar, G. E. Fagg, and J. Dongarra. Automatically tuned collective communications. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 3, Washington, DC, USA, 2000. IEEE Computer Society.

[22] P. Yew, N. Tzeng, and D. Lawrie. Distributing Hot Spot Addressing in Large Scale Multiprocessors. *IEEE Trans. Comput.*, 36(4):388–395, 1987.

[23] W. Yu, D. Buntinas, R. L. Graham, and D. K. Panda. Efficient and scalable barrier over quadrics and myrinet with a new nic-based collective message passing protocol. In *18th International Parallel and Distributed Processing Symposium (IPDPS 2004), CD-ROM / Abstracts Proceedings, 26-30 April 2004, Santa Fe, New Mexico, USA*, 2004.