Assessing Single-Message and Multi-Node Communication Performance of InfiniBand

Torsten Hoefler Carsten Viertel Torsten Mehlan Frank Mietke
Wolfgang Rehm
Technical University of Chemnitz
Dept. of Computer Science
Chair of Computer Architecture
{htor,vca,tome,mief,rehm}@cs.tu-chemnitz.de

Abstract

We present a micro benchmark suite to evaluate InfiniBand^{TM} implementations with regards to single message performance and the addressing of many hosts. We use a 1:n communication pattern to assess the latency and bandwidth for all different combinations of InfiniBands^{TM} transport services and functions. The results gathered in this study are used to optimize MPI collective communication operations where 1:n communication schemes are not used widely today. We show that applications as well as collective algorithms can benefit from sending multiple messages in a single round. Moreover, the results will be used to choose the transport service and function to develop InfiniBand^{TM} optimized collective communication functions. Our study compares all available transport options and shows that single packet sends can be very expensive compared to multi packet sends.

1. Introduction

Many newly deployed cluster-based supercomputers use InfiniBand^{TM} [22] as interconnect network and require an efficient MPI [5, 18] implementation. The InfiniBand^{TM} nework is able to offer latencies down to $1.3\mu s$ [4] and a bandwidth of up to 60Gb/s[21]. The main goal of the MPI library is to preserve the low latency and the hign bandwidth and to leverage the network as efficient as possible for collective communication. The usual way of implementing collective operations is to model the operation as a collection of point-to-point messages, for example in a tree-like fashion. Usually point-to-point benchmarks are used to determine the speed of a specific network and parametrize a network model. Many different models could be used to assess the network performance; the most common are the Hockney model [7] and the LogP model family [3, 1, 17, 11]. This simple modelling approach ignores architectural details of offloading-based networks like InfiniBand^{TM}.

Several studies [9, 10] have shown that a node to multi-node communication pattern can be used to optimize collective communication atop InfiniBand^{TM}. Our new InfiniBand^{TM} barrier [9], implemented as a collective module [20] in Open MPI [6], performs up to 40% better (with 64 nodes) than the optimized InfiniBand^{TM} barrier in the current version of MVAPICH [15]. It uses the n-way dissemination principle [8] which sends more than one message per node in a single round. We also proposed a new model to assess the 1:n-n:1 communications atop InfiniBand^{TM} [10]. These results indicate that the importance of fully analyzing node to multi-node communication can be crucial for the optimized implementention of MPI collective operations.

Our study contributes to the understanding of the InfiniBand hardware in terms of collective multi-node communication. We analyze the performance of 1:n and n:1 communications atop InfiniBand^{TM} and provide detailed benchmark results for a specific cluster system.

Section 1.1 comments related work in the field of microbenchmarking cluster performance. We continue in section 2 with a short explanation of InfiniBand's different transport types as each of them could be chosen to perform collective operations atop. The general benchmark principle for the different transport types is explained in section 3 followed by detailed benchmark results on a 64 node cluster system. The last section summarizes the results gathered in this paper and summarizes future steps to optimize collective communication.

1.1. Related Work

Only a few benchmarks are available for InfiniBand^{TM}. The most common practice is to test the performance atop MPI with MPI-level benchmarks (e.g. [19]). We had to implement a new benchmark because we want to evaluate the hardware in all its details and with a special benchmark procedure. The available MIBA Microbenchmark Suite [2] is able to measure point-to-point communication performance for InfiniBand^{TM}. Additional studies [13] used this benchmark suite to measure communication and application performance on different systems. However, it does not encounter the effects of a node to multi-node communication pattern.

2. InfiniBand Transport Types

The InfiniBand^{TM} standard offers different transport services like stream or datagram to the user. Each transport service has special features and shows different communication times and overheads. Additionally, different transport functions can be performed atop each transport service (see table 36 at page 245 in [22]). A combination of transport function and transport service will be called transport type in the following. All different transport types have to be benchmarked and analyzed in order to draw an accurate conclusion as to which type should be used to gain the best result for the implementation of a specific MPI Collective operation. The different transport types are explained shortly in the following.

2.1. Transport Services

This section explains the different transport services defined in the InfiniBand^{TM} specification briefly.

Unreliable Connection The Unreliable Connection (UC) offers an unreliable connection based point-to-point transmission without flow control. Packets may be silently discarded during transmission. A queue pair (QP) has to be created at each host and connected to each other in order to transmit any data. The queue pair acts as a connection and endpoint identifier in this way.

Reliable Connection The Reliable Connection (RC) is basically identical to UC despite the fact that the correct in order transission of packets is guaranteed by the InfiniBand^{TM} hardware. An automatic retransmit and flow control mechanism is used to ensure correct delivery even if slight network errors occur. The role of the QPs remains the same as for UC.

Unreliable Datagram The Unreliable Datagram (UD) transport type offers connectionless data delivery. The reception or in order delivery is not guaranteed in this case. QPs are only used to send or receive packets and each packet can have a different destination (they do not denote a virtual channel as for UC/RC). A single datagram must not exceed the MTU of the underlying network.

Reliable Datagram The Reliable Datagram (RD) transport type adds a guaranteed reliable in order delivery to UD. QPs can be used to reach any target as in the UD case. RD is currently not supported in our test system.

RAW The RAW Transport type can be used to encapsulate other transmission protocols as ethernet or ipv6. It has nearly the same characteristics as UD and will thus not be discussed in the following.

2.2. Transport Functions

In this section, the possible transport functions are discussed and bound to a specific transport type for the benchmark.

Send The simple send function is available for all IBA transport services and will be measured for all of them explicitely.

RDMA Write The Remote Direct Memory Access Write (RDMAW) can write to explicitely registered memory at a target without the need to interrupt the target's CPU. RDMAW can be used atop RC, UC and RD. We present results for RDMAW atop RC because RD is currently not supported.

RDMA Read The Remote Direct Memory Access Read (RDMAR) can read from registered memory at a target system without influencing the remote CPU. RDMAR can be used atop RC and RD. We present results for RDMAR atop RC because RD is currently not supported.

Atomic Operations Atomic Operations (AO) can be very useful to support the MPI_BARRIER collective call. Our current system does not support the optional AOs and we cannot present any measurement values.

Multicast Multicast (MC) is available for UD only and could be used to enhance MPI collective calls like MPI_BCAST. Several studies [14, 12] have been conducted to leverage the multicast features for different collectiv operations. Our measurements will provide a tool to give a theoretical proof of their efficiency.

3. Benchmark Principle

We use four different benchmark scenarios to test the four different InfiniBand™ transport functions. We use notified receives in the send case that each received packet creates a CQ entry which can be consumed via the poll CQ VAPI call. The simple send-receive is tested with the following 1:n n:1 principle between n nodes numbered from $0..n-1$:

1. node 0 posts $n-1$ send requests to each node $1..n-1$ (ping)
2. node 0 polls it's CQ until all nodes answered
3. node $1..n-1$ waits for the reception of a message (poll CQ)
4. node $1..n-1$ sends the message back to node 0

The RDMAW benchmark uses basically the same principle but the receive (step 2 and 3 above) is unnotified. The only way to detect the last byte of the receive memory region is to poll a counter. The RDMAW operation has been finished when the last byte of the receive buffer is changed (in order delivery is guaranteed). Checking the whole buffer would introduce too much memory congestion and the performance would decrease significantly for larger data sizes. The receive buffer layout in node 0 is depicted in figure 1.

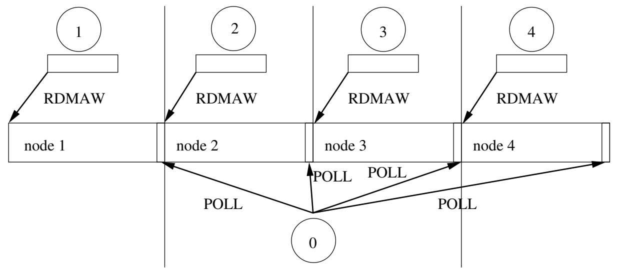

Figure 1. RDMAW Receive Buffer Layout at Node 0

Node 0 polls $n-1$ bytes to check the receiption from $n-1$ peers. Polling introduces a memory congestion overhead on node 0 which has to be accepted because there is no faster method of testing whether a message has been received or not (interrupts or CQ elements are to slow in this context).

RDMAR performance is measured in two stages. First node 0 reads from all $n-1$ nodes and takes the time. The time for $n-1$ nodes to read from a single node is measured in a slightly more complicated way as described in the following.

1. node 0 sends via RDMAW to node $1..n-1$
2. node $1..n-1$ wait via polling for the RDMAW from node 0
3. node $1..n-1$ read via RDMA Read from node 0
4. node $1..n-1$ take the used time for their RDMAR operation

The Multicast benchmark is performed by sending a single UD Multicast packet from node 0 to all other nodes. All other nodes wait for the CQ entry via polling the CQ. Each node returns a unicast UD packet to node 0.

All round-trip-times (RTT) are usually measured at node 0 between the first send and the last receive (except in the second stage of RDMAR). The RTT is divided by two and denotes the latency for the specific communication pattern.

This benchmark also tests the performance of the InfiniBand™ implementation (the switch as well as the HCA and the software stack) under heavy load and maximum congestion (n nodes send/receive to/from a single node).

To enable statistical analysis as well as the assessment of minimal and maximal transmission times we use a different measurement scheme than most other benchmarks. Common benchmarks (e.g. [19, 2]) often perform a defined number of repetitions s (e.g. $s = 1000$) in a loop and divide the measured time of all tests by s afterwards. This prohibits a fine-grained statistical analysis (to find mavericks as well as determine absolute hardware limited minima). Our approach measures each packet separately and stores the result in an array. This makes it possible to find minimum and maximum values and to calculate the average. This measurement scheme has also some impact on the measured values themselves. We measure each packet, which means that the whole pipeline startup (in the network cards itself and in the network) has to be charged to this. This results in very poor performance compared to the usual fully pipelineable benchmarks (1000 repetitions). However, because parallel applications do not communicate 1000 messages between two hosts (usually this is done in a single bigger message), our scheme represents the reality better.

4. Benchmark Results

We present the benchmark Results for a 64 node InfiniBand™ system in this section. The system consists of:

- Processor: 3 GHz dual Xeon
- Memory: 4GB
- OS: Red Hat Linux release 9 (Shrike)

- Kernel: 2.4.27 SMP
- HCA: Mellanox "Cougar" (MTPB 23108)

A single run returns measurement values for varying message sizes and a varying number of processors. The $RTT/2$ latency scaling with regards to the message size as well as the scaling with regards to the number of participating processors is described in the following section.

4.1. Scaling with Message Size

First, we analyze the latency scaling with the message size for 1:1 communication (normal ping-pong benchmark). The latency scaling for increasing small message sizes up to 1k is shown in Figure 2. The MTU has been adjusted to 2k so that every message fits into a single packet. The latency scales linearly with the message size as expected. A LogGP [1] modelling of this scenario would predict accurate communication times as the G in the LogGP model scales linearly with the message size. The second analysis determines the latency scaling with the message size for a 1:15 communication (16 nodes). Figure 3 shows the latency scaling up to 1k. The measurement results show the very interesting result that the single packet latencies to address 16 hosts are lower than the according latencies in the host-to-host (1:1) case. The only exception is the multicast result, which uses a single multicast in the 1:15 direction and 15 UD in the 15:1 direction. This leads us to the conclusion that the interface uses a rather deep pipeline on the sender side which benefits multiple sucessive packets.

The bandwidth scaling ($bandwidth = messagesize/latency$) for larger message sizes up to

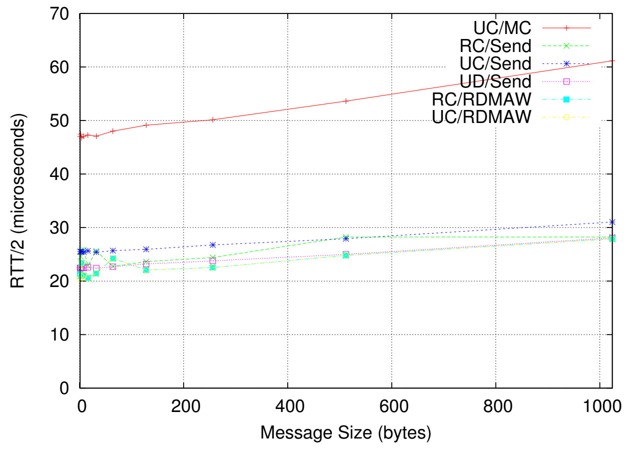

Figure 2. 1:1 $RTT/2$ with varying message sizes

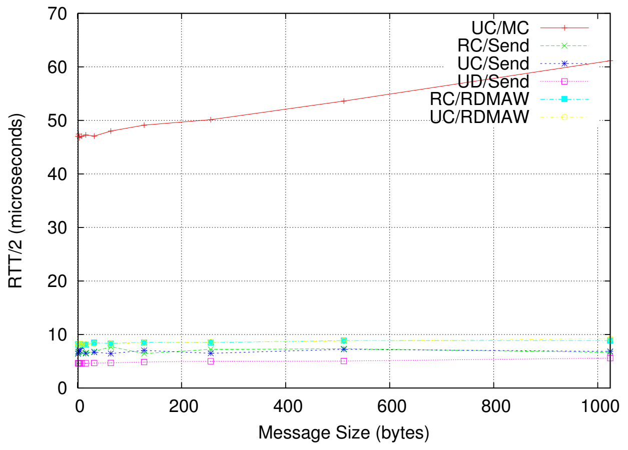

Figure 3. 1:15 Bandwidth with varying message sizes

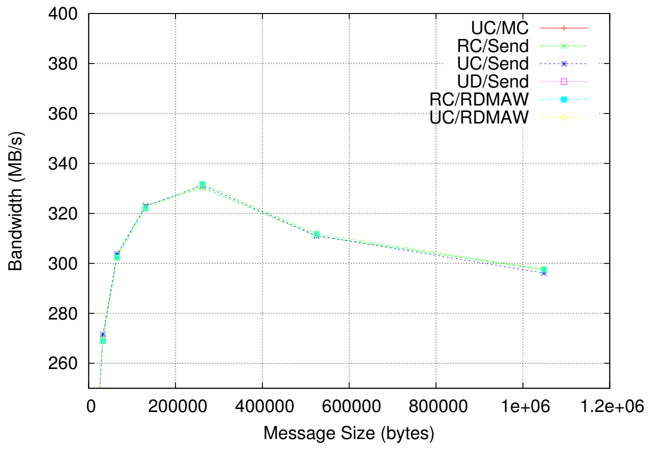

Figure 4. 1:1 Bandwidth with varying message sizes

1M in the 1:1 case ist shown in Figure 4. And the according bandwith for the 1:15 case is shown in Figure 5. Again,

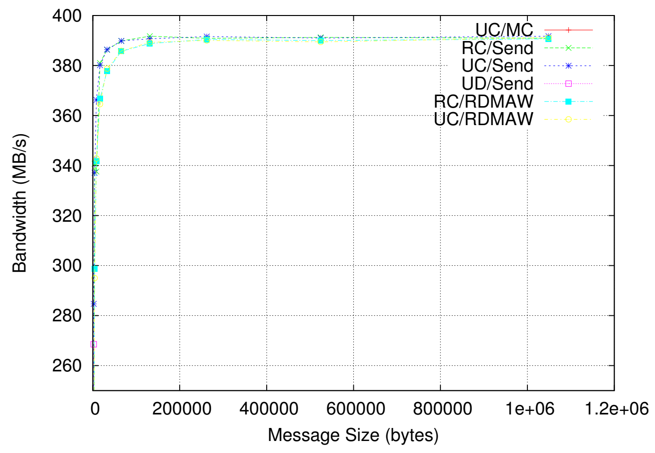

Figure 5. 1:1 Bandwidth with varying message sizes

this benchmark shows clearly that the full power of the HCA cannot be reached with single packets and that a 1:n communication pattern should be preferred over a 1:1 communication. This shows that the LogGP model cannot be very accurate for this type of communication because its predicted communication times are directly proportional to the number of addressed hosts (a single g is accounted for each host). These results are quite interesting for the optimization of collective communication because they show that the sending of multiple packets in a single communication round can increase the throughput and lower the latency of single packets. Today's collective algorithms [16] usually use only a single communication partner per round.

4.2. Scaling with the Number of addressed Hosts

The number of addressed hosts (successively sent packets) seems to play an important role. This will be analyzed and shown in the following. Figure 6 shows the measured latency in relation to the host-number for the transmission of a single byte message. We see that the latency is decreasing with a growing the number of addressed hosts for MC and UD. Other transport services seem to have a local minimum around $n = 10$ and the latency is increasing for higher processor counts. This may be due to local congestion (polling the memory or creating and/or polling CQ entries). This result shows clearly that one should send up to 10 messages in parallel to achieve best results. This discovery has already been used to optimize the InfiniBand^{TM}

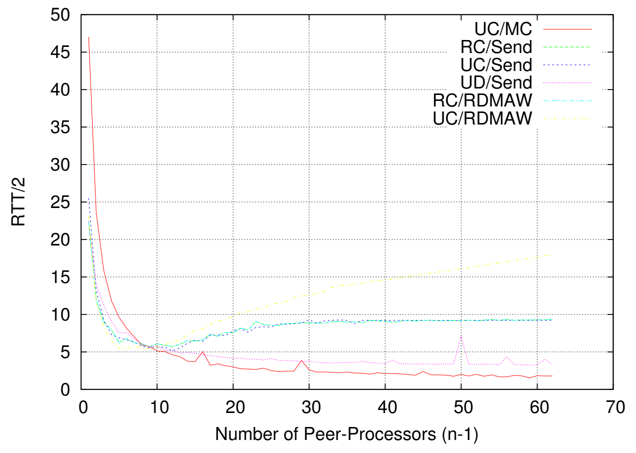

Figure 6. Single byte Transmission Latency

MPI_BARRIER implementation [9] with a new algorithm which sends more than one packet per round (n-way dissemination - [8]). We will analyze in the following if this behavior can also be used to optimize MPI collective operations which need to communicate bigger data chunks (as MPI_BCAST). Figure 7 shows the appropriate scaling for a single 1M message. We see the same results for bigger

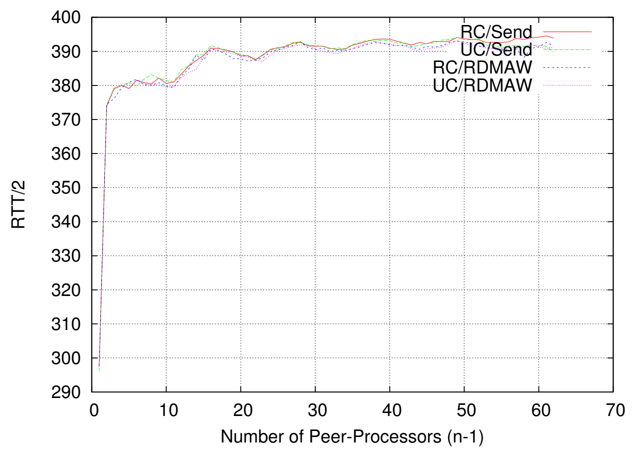

Figure 7. 1M Message Transmission Latency

message sizes. This leads to the conclusion that one should send more than one message per collective communication round regardless of the message size.

All measured parameters are highly system dependent. We have conducted these measurements on different InfiniBand^{TM} systems (consult [10] for details) and saw the same results. The shape of all benchmarked curves stayed the same, only some parameters varied. This makes it possible to use our benchmark suite to evaluate and compare the performance of different InfiniBand$^{\text{TM}}$ implementations and to find bottlenecks during InfiniBand$^{\text{TM}}$ development. The benchmark suite is also able to measure parameters like send and receive overhead and latencies to poll the completion queue under different circumstances. But these have been left out due to space restrictions.

5. Acknowledgements

Thanks to Dr. A. Lumsdaine and Dr. L. Hopkins at the Indiana University for helpful comments to improve this paper.

6. Conclusions and Future Work

We propose a new 1:n n:1 benchmarking principle to assess the performance of the InfiniBand$^{\text{TM}}$ network for all different transport types. Our proposed solution uses time measurement for each packet to enable detailed statistical analysis afterwards. We use this benchmark to evaluate hardware InfiniBand$^{\text{TM}}$ implementations and to search for new optimization possibilities for MPI collective operations atop InfiniBand$^{\text{TM}}$. The benchmark results are very helpful to design new collective algorithms. Especially the discovery that single message sends are quite slow compared to multi message sends is very important for collective algorithms (as most of theme use mainly single message sends). We can conduct a statistical analysis on our benchmark results to find mavericks and mean values.

References

[1] 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.

[2] B. Chandrasekaran, P. Wyckoff, and D. K. Panda. Miba: A micro-benchmark suite for evaluating infiniband architecture implementations. *Computer Performance Evaluation / TOOLS*, pages 29–46, 2003.

[3] 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.

[4] L. Dickman. An introduction to the pathscale infinipath htx adapter. *Pathscale White Papers*, November 2005.

[5] M. P. I. Forum. MPI: A Message Passing Interface Standard. *www.mpi-forum.org*, 1995.

[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] R. Hockney. The communication challenge for mpp: Intel paragon and meiko cs-2. *Parallel Computing*, 20(3):389–398, March 1994.

[8] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. An optimal synchronization algorithm for multi-port networks. 2005. submitted to the Journal of Parallel Processing .

[9] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. Fast Barrier Synchronization for InfiniBand. In *Accepted at the IPDPS06 CAC Workshop*, 2006.

[10] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. LogfP - A Model for small Messages in InfiniBand. In *Accepted at the IPDPS06 PMEO-PDS Workshop*, 2006.

[11] F. Ino, N. Fujimoto, and K. Hagihara. LogGPS: A Parallel Computational Model for Synchronization Analysis. In *PPoPP '01: Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming*, pages 133–142. ACM Press, 2001.

[12] 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.

[13] J. Liu, B. Chandrasekaran, W. Yu, J. Wu, D. Buntinas, S. Kini, D. K. Panda, and P. Wyckoff. Microbenchmark performance comparison of high-speed cluster interconnects. *IEEE Micro*, 24(1):42–51, January 2004.

[14] J. Liu, A. Mamidala, and D. Panda. Fast and scalable mpi-level broadcast using infiniband's hardware multicast support, 2003.

[15] J. Liu, J. Wu, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. *Int'l Journal of Parallel Programming, 2004*, 2004.

[16] G. I. Massimo Bernaschi. Collective communication operations: experimental results vs. theory. *Concurrency - Practice and Experience 10*, 5:359–386, 1998.

[17] 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.

[18] M. P. I. F. MPIF. MPI-2: Extensions to the Message-Passing Interface. Technical Report, University of Tennessee, Knoxville, 1996.

[19] Pallas GmbH. Pallas MPI Benchmarks - PMB, Part MPI-1. Technical report, Pallas GmbH, 2000.

[20] 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.

[21] M. Technologies. Infiniband - industry standard data center fabric is ready for prime time. *Mellanox White Papers*, December 2005.

[22] The InfiniBand Trade Association. *Infiniband Architecture Specification Volume 1, Release 1.2*. InfiniBand Trade Association, 2003.