Designing QoS Aware MPI for InfiniBand - Techinical Report *

Hari Subramoni, Ping Lai, and Dhabaleswar K. Panda

Department of Computer Science and Engineering, The Ohio State University

{subramon, laipi, panda}@cse.ohio-state.edu

1 Abstract

As computing clusters grow in size and core count, congestion at the host as well as the network level is becoming a real problem in modern high performance computing. This is especially true in the popular cluster of cluster scenario where bottleneck links are readily available. With the advent of long range capable InfiniBand devices, we are now looking at Cluster of cluster scenarios that may span across continents. Providing Quality of Service for all applications in such scenarios pose a formidable challenge for cluster designers as well as administrators. On the other hand, modern high performance network interconnects like InfiniBand now provides us with advanced features like fine grained network level bandwidth provisioning. Such features give us the ability to create networks where we can minimize common network problems such as small message starvation and the like. The emerging trends in modern high performance computing mentioned above, gives us a strong motivation to explore QoS provisioning for high performance networks from a quantitative angle. Though current designs allow us to specify the priority for a job at the job launch time, there exists no way to ensure that this priority is enforced once the job is actually running on the network. Through our research, we aim to make the MPI Stack *QoS-aware* and ensure that the priority settings are adhered to throughout the life cycle of the job resulting in an overall improvement in application performance. To the best of our knowledge, this is the first such QoS-aware MPI design and evaluation for InfiniBand-based clusters. We explore various design alternatives such as Inter-Job and Intra-Job bandwidth partitioning by use of multiple service levels and effective use of NIC level buffers by utilizing multiple Virtual Lanes for data transmission. Our experiments show a 220% performance improvement in small message latency and 90% performance increase in large message latency can be achieved on giving increased priority

*This research is supported in part by DOE grants #DE-FC02-06ER25749 and #DE-FC02-06ER25755; NSF grants #CNS-0403342 and #CCF-0702675.

to the messages. Our application level evaluation with the NAS FT benchmark shows a marked improvement in performance when using multiple virtual lanes over just a one.

2 Introduction

The presence of multiple small clusters in a geographically distributed manner, has resulted in scientists looking for ways to interconnect these smaller clusters and aggregate them into larger, and potentially more powerful computing systems. With the advent of long range capable IB devices like the Obsidian Longbow [17] series of routers, cluster of clusters are now capable of having inter-continental spans. Providing Quality of Service for all applications in such a scenario poses a formidable challenge for cluster designers as well as administrators. In this context, we use advanced features like fine grained network level bandwidth provisioning, provided by modern network interconencts like InfiniBand to create networks where we can minimize network congestion. Given the emerging trends in modern high performance computing mentioned above, this gives us a strong motivation to explore QoS provisioning for high performance networks from a quantitative angle. To the best of our knowledge, this is the first such QoS-aware MPI design and evaluation for InfiniBand-based clusters.

In this paper, we explore multiple options to effectively utilize the QoS concepts to enhance the cluster performance. Our schemes are broadly classified into Inter-Job and Intra-Job. Inter-Job schemes rely on giving one job priority over another allocating it a higher priority service level. Intra-Job schemes look at enhancing the performance of an application by properly provisioning the network bandwidth and ensuring that smaller messages do not get starved by getting queued up behind larger messages. We also look at schemes to properly utilize he available hardware resources in terms of NIC level buffers allocated to multiple virtual lanes. As evidenced by our NAS performance results, properly provisioning the network resource not only gives good performance improvement for small messages, but for large messages as well. From our micro benchmark level evaluation, we can see an improvement of up to 220% for small messages and up to 90% for medium to large messages using a higher priority in scenarios where there is congestion in the network.

Another key observation made is that, proper utilization of available hardware resources like NIC level buffers allocated for each virtual lane results in appreciable performance improvement as demonstrated by our micro benchmark and application numbers. We see a performance improvement of up to 45% for all messages sizes from the micro benchmark level tests.

The rest of the paper is organized as follows. Section 3 gives a brief overview of InfiniBand, QoS in InfiniBand and MPI. Section 4 explains the design methodology we followed. Our experimental setup and results are described in Section 5. Section 6 gives an overview of the related work in the field. Finally we summarize our conclusions and possible future work in Section 7.

3 Overview of InfiniBand and MPI

3.1 InfiniBand and QoS

InfiniBand Architecture (IBA) [5] defines a switched network fabric for interconnecting processing nodes and I/O nodes. It uses a queue-based model. A Queue Pair (QP) consists of a send queue and a receive queue. The send queue contains instructions for transmitting data and the receive queue contains the instructions describing where the receive buffer is. At the low level, InfiniBand supports different transport services including Reliable Connection (**RC**) and Unreliable Datagram (**UD**).

IB QoS Support: IBA supports QoS at two levels, using SL (service level) at switch level and TClass (traffic class) at router level. SL is a field in the LRH (local route header) of a packet. The QoS mechanism in IBA consists of three components: virtual lane, virtual lane arbitration and link level flow control. IBA defines that each port must have a minimum of two and a maximum of 16 virtual lanes. Depending on the type of implementation, each virtual lane can have different transmitting and receiving buffers.

Every network element in the subnet maintains a Service Level to Virtual Lane (SL2VL) mapping table that specifies to which virtual lane we need to send packets belonging to one service level. It also has a virtual lane arbitration (VLArb) table which defines two strict priority levels - low priority and high priority. In each priority level, we define a weighted round-robin scheme of arbitration between the virtual lanes. These two tables together ensure that each packet will be forwarded according to its agreed upon service level (priority) across the entire network. The VLArb low priority and high priority tables contain values which indicate the number of 64byte units/credits that can be sent out through a virtual lane before yielding to the next one. The VLArb high priority table is processed before the VLArb low priority table. We can also put a cap on the number of credits that can be sent by all the high priority entries combined before we yield to the low priority entries. This ensures that the applications using the low priority queues are not starved. ConnectX [13] is the fourth generation InfiniBand adapter from Mellanox Technologies. It has two ports, with 8 InfiniBand virtual lanes for each. It provides fine-grained end-to-end QoS and congestion control. Latest driver OFED 1.3 [18] has included the capability of enabling QoS support.

3.2 MPI

Message Passing Interface (MPI) [14] is one of the most popular programming models for parallel computing. MVAPICH [16] is a high-performance MPI implementation over InfiniBand clusters, used by more than 800 organizations world-wide. Generally, small message (including small data message and control message) and large message passing have different performance requirements in terms of latency. Although many MPI designs [16, 15] employ different protocols for them, they do not differentiate their priorities. As the hardware-based QoS support is maturing in IBA at

multiple layers, including verbs level, it is possible that we can utilize this feature to provide QoS provisioning to different messages in MPI.

4 Design and Methodology

As computing clusters grow in size and number, providing Quality of Service for jobs throughout its life cycle becomes critical. This is all the more important in large clusters where chances for congestion are higher. In large computing clusters, it is usually the scheduler's responsibility to select user requests from job queues and launch them. We can provide job prioritization at this stage, in the sense that, the jobs belonging to a particular queue may be launched before jobs in other queues. But once the jobs are actually running on the machines, there exists no way in current MPI stacks to enforce the initial priority settings defined by the administrator. Through our research, we aim to fill this void in the current design and make the MPI stack *QoS aware*.

The InfiniBand specifications define some very useful features to provide QoS to an InfiniBand Network. Out of these, the two key features are the mapping of Service Levels to Virtual Lanes and the varying levels of priority given to each virtual lane. While the former allows us to smoothly direct the flow of packets over the network, the latter enables us to allocate varying priority or weights to the packets sent through these queues. In this paper, we explore multiple design alternatives, aiming to provide QoS to the various jobs running in an InfiniBand network.

4.1 Inter Job QoS

The jobs running in a large scale computing system can be various types and have different QoS requirements, i.e., short-term or long-term jobs, jobs requiring high bandwidth and jobs sensitive to latency etc. Treating all jobs equally will result in performance degradation for some, or all of the jobs.

Consider the scenario where an application, such as the NAS [1] LU benchmark which is extremely sensitive to latency, is running along side a large application such as the NAS FT benchmark which sends out many large messages. As seen in in earlier works [] relevant to congestion management, the large application will create various congestion points in the network, resulting in bad performance for the delay sensitive application. To address this, our solution is to provide job level QoS.

We define multiple *Service Levels* with varying performance metrics. Jobs can be mapped to different service levels. In particular, at the job launch time, the scheduler assigns a specific priority to a job. The MPI library internally translates this priority to an InfiniBand service level, which will subsequently be used for all InfiniBand network operations. We use the opensm [18] infrastructure to configure the *SL2VL* and the *VLArb* tables. Based on this, the network elements will prioritize and classify the packets, ensuring the same QoS for the packet anywhere on the

network.

4.2 Intra Job QoS

It is well known that in large congested networks, small message latency gets adversely affected by large volume of data going through the network. In this context, we modify the design of the MPI library to utilize different service levels for small and large messages.

Our initial design is based on the Reliable Connected (RC) transport. According to the InfiniBand specification, the service level parameter can be changed only when initializing the Queue Pair (QP). Given this constraint, we create two QPs for each process and associate one with the higher priority service level and another with the lower priority service level. The QP associated with the higher priority service level will be used exclusively for small message transmission while the other will be used for large message transmission.

As most of the control messages in MPI are small messages, giving high priority to small messages will not only enhance their performance, but also the performance of large messages whose progress relies on small control messages.

4.3 Utilizing multiple Virtual Lanes

As mentioned in Section 3.1, InfiniBand offers link layer Virtual Lanes (VLs) to support multiple logical channels on the same physical link. Authors [4] have done theoretical work indicating that virtual lanes provide a good mechanism to avoid Head of Line (HoL) blocking [10] and give us the ability to support Quality of Service. Currently, most MPI libraries do not make use of multiple virtual lanes for data transmission, which is usually done through the default virtual lane - *0*.

Each virtual lane is allocated a set of buffers used to hold the packets for transmission. Depending on the implementation, there are various ways in which the buffers can be split up among the virtual lanes. Usually, each virtual lane is allocated a set of private buffers which is not shared with other virtual lanes. Apart from this, there is also a common buffer pool which, depending on demand, is used on a shared basis by all virtual lanes. By using just one virtual lane for all communication, the current MPI libraries prevent themselves from using the private buffers reserved for the other virtual lanes. This will lead to contention between the packets being sent by various processes sharing the same virtual lane. This contention is expected to increase with the explosion in core count, as evidenced by the current multi-core/many-core architectures. In this context we propose a QoS aware MPI which will perform load balancing across multiple virtual lanes by sending packets through them in a round robin fashion. Such a mechanism will help to alleviate contention by the proper utilization of hardware resources in order to extract maximum performance.

In networks that are not highly congested, the use of such a QoS aware MPI library will enable us to provide a higher quality of service to all traffic in the network. However, it is to be noted

that in highly congested networks, due to sheer volume of network traffic, we will have to perform prioritization.

5 Performance Evaluation and Results

We detail the results of our experimental evaluation in this section. For all the experiments, we introduce some background MPI traffic in order to generate enough network congestion so that the QoS provisioning will really take effect. In the interest of fairness, the same QoS settings were used for background traffic and the traffic that we are measuring.

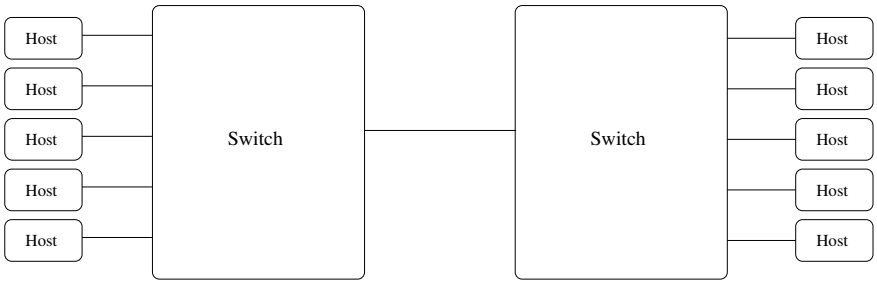

Figure 1: Experimental Setup

5.1 Experimental Setup

Figure 1 shows the basic setup we used to run our experiments. All numbers were taken on 4 dual quad core Intel Xeon processors running at 2.33 GHz. As the figure shows, we had some additional nodes in the system to simulate the network load experienced by machines on large computing clusters. We used Mellanox ConnectX HCA's on all machines where the numbers were taken.

5.2 Performance of Inter-job QoS Provisioning

In this section, we look at the advantage in performance we gain by giving one job priority over another when there is load on the system. The micro benchmark experiments are running between two of the hosts while the other hosts are running a bandwidth intensive application which we use to create background traffic. As we can clearly see from the micro benchmark results shown in Figure 2, there is significant performance advantage to be gained from using a different priority level. The application level numbers also show a similar trend in performance. Due to lack of space, we do not show these numbers here.

5.3 Performance of Intra-job QoS Provisioning

In this section, we characterize the performance of assigning intra-job QoS, i.e., provision different priorities to small and large messages in a MPI job. We first measure the basic latency given that small messages have higher priority. Then we measure the impact of adjusting the threshold for large messages. Finally we use NAS benchmark to present the performance with various priority ratios.

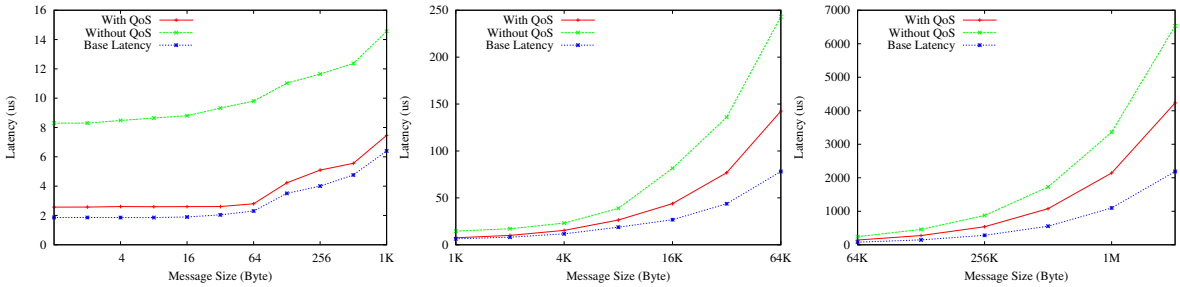

Figure 2: Impact of QoS provisioning on latency of: (a) Small messages, (b) Medium messages and (c) Large messages

As mentioned in Section 4, it may be quite beneficial to assign high priority to small messages. In this experiment, we configure the QoS ratio of small messages to large messages to be 1 Kilo Bytes, and measure the basic latency as shown in Figure 3. We compare the performance of three cases: (i) the baseline latency; (ii) the performance with background traffic while no QoS provisioning; and (iii) the performance with background traffic and QoS provisioning. From Figure 3 (a), we see that higher priority to small messages can effectively accelerate the transmission, resulting in much lower latency as compared to case (ii). On the other hand, we observe from Figure 3 (b) that large message transmission is not affected much due to its already high latency. These results present the potential benefits in applications by assigning higher priority to small messages which we will present shortly.

Following the first set of experiments, we then varied the threshold between small messages and large messages from 1 Kilo Bytes to 8 Kilo Bytes. The same QoS provisioning parameters are used. Due to lack of space we only show the results with the threshold set at 2 Kilo Bytes. As we can observe from Figure 3 (c), there is subtle change for different thresholds, i.e., smaller latency holds until the message size is larger than threshold. This indicates us to adjust the threshold when provisioning QoS priorities according to different needs. This threshold need not be static, we can have multiple virtual lanes configured to handle multiple message sizes. Thus, we could have one priority for small, one for medium and one for large message sizes.

We finally use NAS benchmark to evaluate the impact of intra-job QoS provisioning. The message threshold is again set to the default value of 1 Kilo Byte. We observe, for applications

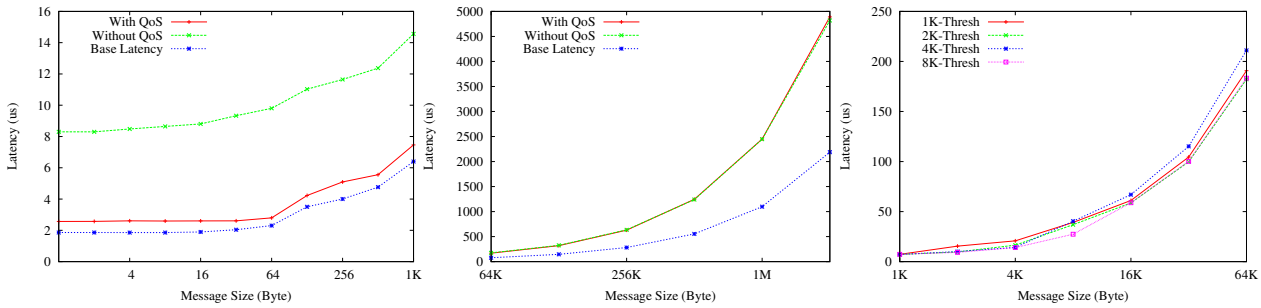

Figure 3: Micro benchmark latency with Intra-Job QoS (a) Small messages, (b) Large Message and, (c) With varying small-to-large message threshold

like FT, using mostly large messages, ensuring that small control messages are not starved due to contention from large messages increases overall performance. As we can see from Figure 4, the priority for small and large messages needs to be carefully set. Too high a priority for small (S100-L10) or large (S100-L200) messages result in bad performance. The best results are obtained when we give equal priority to small and large messages (S100-L100), ensuring that small messages are not starved by large messages and at the same time, the large messages are not unduly queued up in favor of small messages. Due to in-sufficient load on the system, we are not able to see the benefits of Intra-Job QoS on applications which use a higher percentage of small messages (LU) and approximately equal percentage of large and small messages (CG, MG).

5.4 Performance of Optimizing VL Utilization

In this section, we optimize the QP and VL utilization according to Section 4 in order to improve the application performance. By default, MPI library uses one QP and one particular VL for a job. Both QP and VL may be choked by large number of consecutive transmission requests. This prompted us to utilize multiple QPs and VLs based on the QoS provisioning capability.

Figure 5 (a) and (b) compare the latency performance using various number of QPs and VLs. When we use multiple QPs (e.g., 2QP, 4QP, 8QP) and one VL, the performance degrades as compared to the baseline case. This tells us that the fundamental bottleneck is at VL. Then we increase the number of VL to match the number of QPs. As we see from the cases of 2 QPs with 2 VLs, 4 QPs with 4VLs, and 8 QPs with 8 VLs, the latency decreases continuously. If there is very large volume of traffic, we can expect that the performance will be improved by using more QPs and VLs. It is to be noted that we do not show the performance for small messages, considering that it cannot generate enough traffic to show the benefit. As a matter of fact, we see fluctuations in its performance.

We repeated the same set of configurations for running NAS benchmark, attempting to charac-

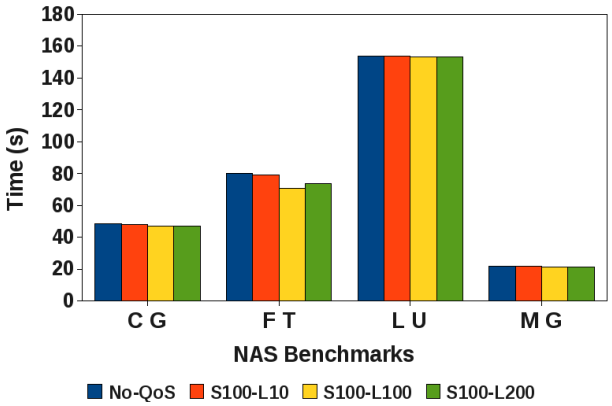

Figure 4: NAS Performance with varying QoS provisioning

terize the application-level benefits. As illustrated in Figure 5 (c), FT has apparent improvement by using more QPs and VLs due to its high percentage of large messages, which is consistent with the trend in the micro-benchmark experiment. LU has no variation in performance since we see a lot of fluctuations in performance of small messages and LU has almost 99% small messages. CG and MG both have fair amount of small and large messages, so we cannot see obvious trends due to the mitigation of effects on small and large messages.

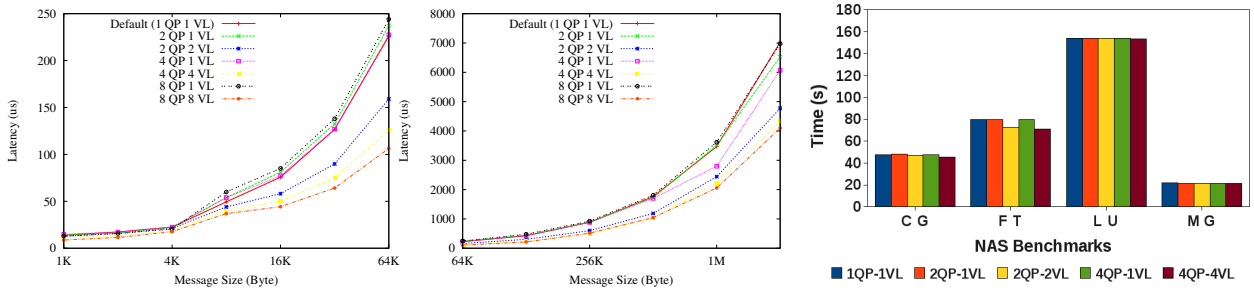

Figure 5: Impact of workload distribution to QPs and VLs: (a) Micro benchmark latency - Medium messages, (b) Micro benchmark Latency - Large messages and (c) NAS performance

6 Related Work

Some researchers have investigated QoS problem over InfiniBand. In [19], authors first described the QoS problem and a generalized approach to its solution using IBA mechanisms. The papers [9, 7] studied how to configure InfiniBand VL arbitration table to achieve required or optimized QoS, and how to deal with the time sensitive traffic through IBA QoS support, respectively. The

same group also improved the arbitration mechanism in [8],maximizing the number of requests to be allocated in the arbitration table that the output ports have. They further made a lot of efforts [6, 12] to improve the cluster QoS based on IBA supported mechanism and try to establish a unified system architecture. Although all of these research stems from InfiniBand QoS specification, they mainly focus on the theoretical exploitation. On the contrary, in [20] authors made investigation on real IB cluster and experimented its QoS provisioning capability by assigning different QoS level to different traffic. The paper also presents the possible sources of bottleneck that prevents the application from obtaining desired QoS performance. Other works in the area of congestion and flow control include [3, 11, 2].

7 Conclusions and Future Work

In this paper we look at the possible benefits that can be obtained by providing appropriate Quality of Service to applications in congested networks. Over the course of our work, we made few key observations about MPI applications. As evidenced by our micro benchmark and application level evaluation, we can enhance the performance of large messages as well as small messages by ensuring that small messages do not get queued up behind large messages. We also see that proper utilization of available hardware resources like the NIC level buffers allocated to each virtual lane can significantly enhance performance of applications using large as well as small messages.

One possible issue that we foresee with the current implementation is the memory requirements imposed by having to use multiple RC QPs for each process in the system. As part of future work, we plan to look at using the UD communication protocol which allows us to set the service level at message transmission time. The concept of QoS is not limited to the field of MPI, it has wide applications in Data Center as well as File System research.

8 Acknowledgement

We would like to thank Jonathan Perkins, our system administrator, for all the help he has done to make this paper possible. We would also like to thank Matthew Koop for his valuable input during the design of various experiments. We would also like to thank Todd Wilde and Jim Mott from Mellanox for the help they extended.

References

[1] NAS Parallel Benchmarks. http://www.nas.nasa.gov/Resources/Software/npb.html.

[2] C Minkenberg and M Gusat. Speculative Flow Control for High-Radix Datacenter Interconnect Routers. In *IPDPS*, 2007.

[3] Frank Olaf Sem-Jacobsen, Sven-Arne Reinemo, Tor Skeie, Olav Lysne. Achieving Flow Level QoS in Cut-Through Networks Through Admission Control and DiffServ. In *HPCC*, 2006.

[4] M. E. Gmez, J. Flich, A. Robles, P. Lpez, and J. Duato. A methodology to reduce hol blocking in infiniband networks. *Parallel and Distributed Processing Symposium, International*, 0:46a, 2003.

[5] Infiniband Trade Association. http://www.infinibandta.org.

[6] F. Joslfaro, J. S. andand M. Mendui, and J. Josuato. A Formal Model to Manage the InfiniBand Arbitration Tables Providing QoS. In *IEEE Trans. Computers*, 2007.

[7] F. Joslfaro, J. Sh, and J. and. A Strategy to Manage Time Sensitive Traffic in InfiniBand. In *16th International Parallel and Distributed Processing Symposium*, April 2002.

[8] F. Joslfaro, J. Sh, and J. and. A New Proposal to Fill in the InfiniBand Arbitration Tables. In *International Conference on Parallel Processing*, Sep 2003.

[9] F. Joslfaro, P. J. Shez, Josuato, and C. R. Das. A Strategy to Compute the InfiniBand Arbitration Tables. In *16th International Parallel and Distributed Processing Symposium*, April 2002.

[10] M. Karol, M. Hluchyj, and S. Morgan. Input Versus Output Queuing on a Space-Division Packet Switch. In *IEEE Transactions on Communications*, Dec 1987.

[11] M Gusat, D Craddock, W Denzel, A Engbersen, N Ni, G Pfister, W Rooney, J Duato. Congestion Control in InfiniBand Networks. In *HotInterconnects*, 2005.

[12] A. Martz-Vicente, G. Apostolopoulos, F. Joslfaro, J. Sh, and Josuat. Efficient Deadline-Based QoS Algorithms for High-Performance Networks. In *IEEE Trans. Computers*, 2008.

[13] Mellanox Technologies. ConnectX Host Channel Adapters.

[14] MPI Forum. MPI: A Message Passing Interface. In *Proceedings of Supercomputing*, 1993.

[15] MPICH2: High Performance portable MPI implementation. http://www.mcs.anl.gov/research/projects/mpich2.

[16] MVAPICH2: High Performance MPI over InfiniBand and iWARP. http://mvapich.cse.ohio-state.edu/.

[17] Obsidian Research Corp. http://www.obsidianresearch.com/.

[18] Open Fabrics Enterprise Distribution. http://www.openfabrics.org/.

[19] J. Pelissier. Providing Quality of Service over InfiniBand Architecture Fabric. In *Proc. Symp. High Performance Interconnects (Hot Interconnects 8)*, Aug 2000.

[20] A. A. R. R. Grant, M. J. Rashti. An Analysis of QoS Provisioning for Sockets Direct Protocol vs IPoIB over Modern InfiniBand Networks. In *P2S2 Workshop, in conjunction with ICPP*, 2008.