An Analysis of QoS Provisioning for Sockets Direct Protocol vs. IPoIB over Modern InfiniBand Networks

Ryan E. Grant Mohammad J. Rashti Ahmad Afsahi
Department of Electrical and Computer Engineering
Queen's University
Kingston, ON, CANADA K7L 3N6
{*ryan.grant, mohammad.rashti*}*@ece.queensu.ca ahmad.afsahi@queensu.ca*

Abstract

The introduction of Quality of Service (QoS) features for socket-based communication over InfiniBand networks provides the opportunity to enact service differentiation for traditional socket-based applications over high performance networks for the first time. The effectiveness of such techniques in providing control over the quality of service that individual connections experience is important in managing traffic in modern data centers. In this paper, we quantitatively analyze the performance benefits of QoS provisioning in InfiniBand networks for Sockets Direct Protocol (SDP) and IPoIB. We find that QoS provisioning can provide prioritized service for sockets-based streams, with more apparent impact on SDP traffic than IPoIB.

1. Introduction

InfiniBand (IB) [8] is a standardized System Area Network used in both high performance clusters and data center environments. The IB networking technology allows for very high-speed data transfer at low latency. To be able to serve legacy IP-based applications over IB in data centers, IPoIB protocol is used as an interface on top of the IB verbs layer [8, 9] that allows socket-based applications to utilize the host's TCP/IP protocol stack, which is then translated into native IB verbs in a manner transparent to the application. Sockets Direct Protocol (SDP) [9] is an optimization of the sockets based interface that allows the system to bypass the TCP/IP stack and translate socket-based packets directly into the verbs layer RDMA operations [9], while maintaining existing TCP stream socket semantics. SDP has the advantage of bypassing several software layers that are necessary in IPoIB, resulting in SDP having better latency and throughput than IPoIB [4]. Nevertheless, both SDP and IPoIB have kernel-level modules imposing some overhead on them.

The application of InfiniBand technology in modern computing centers is of great benefit in reducing communication latency as well as providing much higher available bandwidth to nodes within the local data center network. Data center applications are typically built upon traditional socket-based semantics. Utilizing SDP, such applications can leverage the superior performance of InfiniBand networks while requiring no alterations to the original software.

The management of network traffic is of great concern within modern networks. Quality of Service (QoS) provisioning can be used to regulate traffic such that, for example intra-network traffic can have priority over incoming inter-network traffic. This can be achieved by assigning greater weights to traffic that would typically be the source of a system-wide performance bottleneck, allowing for great flexibility in fine-tuning the performance of the network. Such a system could significantly boost the performance of existing data centers that rely on Ethernet Best Effort service, with little or no need to modify existing socket-based applications. As such, it is important to analyze the behaviour of the emerging hardware-level QoS provisioning for InfiniBand networks with a concentration on the optimized socket-based protocols such as SDP. This work represents a first step towards using new techniques to harness high-speed interconnection technologies for use in back-end support of existing Internet applications.

In this paper, we show how and to what extent the performance of data flows can be boosted by assigning them higher priority in an IB network. In essence, our assessment results show that the QoS provisioning has a definite effect on the performance of socket-based streams. We also observe that performance bottlenecks exist in our servers that prevent us from

exploiting the full potential of QoS support in ConnectX [10] InfiniBand networks. In addition, the overhead associated with extra data copies and kernel involvement in the socket-based protocols is another factor that prevents socket-based streams from taking advantage of the network's QoS capabilities. To the best of our knowledge, this is the first work concerning the effect of QoS provisioning on socket-based communication over real InfiniBand networks.

We provide a brief background on IB networks, ConnectX and the SDP protocol in Section 2. The operation of standard QoS provisioning over IB networks is discussed in Section 3. Section 4 reviews some recent research in the area of SDP and QoS in IB networks. Section 5 introduces our experimental platform. In Section 6, we present baseline single and multiple stream results for SDP and IPoIB based benchmarks, and then we thoroughly examine the functionality and performance of ConnectX InfiniBand networks utilizing QoS provisioning for socket-based traffic. In Section 7, we investigate the sources of performance bottlenecks. Finally, Section 8 concludes the paper and discusses plans for future work.

2. Background

InfiniBand [8] is a leading cluster interconnect with very low latency and high throughput. Native InfiniBand verbs (IB verbs) form the lowest software layer for the IB network, and allow direct user-level access to IB host channel adaptor (HCA) resources while bypassing the operating system. At the IB verbs layer, a queue pair model is used for communication supporting both Send/Receive and RDMA semantics [9]. InfiniBand requires user buffers to be registered prior to being used for communication.

ConnectX [10] is the latest generation of InfiniBand HCAs from Mellanox Technologies. It has two ports that could operate as 4X InfiniBand or 10-Gigabit Ethernet. The ConnectX architecture includes a stateless offload engine for network interface card (NIC) based protocol processing. Moreover, ConnectX supports a number of extended features [10] on top of the IB standard, including enhanced QoS support. Software support is not yet available.

Sockets Direct Protocol [9] was introduced to improve the performance of sockets by exploiting the RDMA abilities of the InfiniBand networks. SDP is a byte-stream protocol based on TCP stream socket semantics. SDP implements a protocol switch inside the operating system kernel that transparently switches between kernel TCP/IP stack over IB (IPoIB) and the SDP over IB (which bypasses the kernel TCP/IP stack) [9]. SDP has two modes of data transfer. In the buffered-copy mode, the socket data is copied in a pre-registered buffer before the network transfer. In the zero-copy mode, the user buffer is directly registered for communication to avoid data copy [3]. A buffered-copy SDP has been implemented in the Open Fabrics enterprise software distribution (OFED-1.3) [14].

3. Quality of Service for IB Networks

QoS management in InfiniBand is done through the use of multiple Virtual Lanes (VL) that allow separation of traffic flows going through a shared link. A Service Level (SL) is assigned to each traffic flow in order to differentiate among different flows. Each SL indicated in the IB packet is mapped to a VL through an SL2VL mapping table, which assigns the packet to one of up to 15 possible VL queues [8, 6].

There are two differentiation mechanisms: *weighted* QoS and *priority queue (blocking)* QoS. In the weighted mechanism, each VL is assigned a weight that identifies the amount of traffic from the flows assigned to that VL, compared to other flows. This is essentially a Weighted Fair Queue (WFQ) mechanism and the weights can vary from 0 (no service) to 255.

In the blocking priority queue mechanism, VLs are assigned to two separate priority levels. The high priority traffic is capable of blocking low priority traffic until either all high priority packets have been sent (queue length of zero) or a set high priority packet limit (*high_limit*) is reached. Once the high_limit is reached, a packet from the low priority VLs is serviced, and then the high priority queue resumes its blocking behaviour. All high priority and low priority VLs have their own weights and one VL can exist in both priority levels. The VL arbiter is responsible for choosing the VL from which the next data packet will go over the wire. Figure 1 depicts a high-level view of the described QoS mechanism.

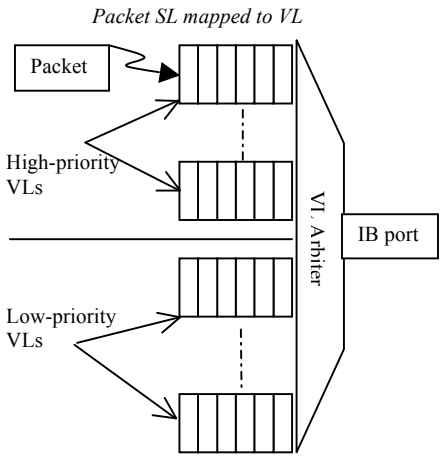

Figure 1. QoS arbitration mechanism

4. Related Work

The performance of the ConnectX architecture, operating with MPI [11] and IB verbs has been analyzed in detail by Sur et al. [17]. The performance of SDP relative to native IB verbs and IPoIB has been documented by Balaji et al. [4]. They found that SDP was capable of outperforming IPoIB by 2.7 times on the previous generation of IB cards from Mellanox.

The benefits to utilizing a zero-copy (as opposed to buffered-copy) SDP protocol have been explored by Balaji et al. [3] as well as Goldenberg et al. [7]. Asynchronous zero-copy SDP sockets offload memory transfers from the CPU to a direct DMA-IB data transfer, allowing for much higher data transmission speeds and reduced CPU overhead. Goldenberg's implementation of zero-copy sockets [7] saw an average improvement of 29% over traditional buffered-copy methods, while Balaji's approach [3] saw improvements of 35%-200% depending on the protocol's application. Balaji et al. [5] have also explored the potential memory bottlenecks that exist in socket-based and RDMA-based communications for 10-Gigabit Ethernet networks, including a comparison to RDMA-based InfiniBand networks.

The QoS performance of native IB verb traffic over IB networks has been explored in [2] on a simulation basis. In addition, Alfaro et al. [1] have explored the impact of message sizes on the ability of IB networks to provide guaranteed QoS to applications. Reinemo et al. [16] have published a survey covering the QoS capabilities of IB networks while comparing them to the QoS provisions in many other modern network architectures.

5. Experimental Platform

Our experiments were conducted on two Dell PowerEdge 2850 SMP servers. The PowerEdge 2850 has two dual-core 2.8GHz Intel Xeon EM64T processors with a 2MB L2 cache for each core on a chip. There are 4GB of DDR-2 SDRAM on an 800 MHz Front Side Bus. Each SMP server is equipped with a Mellanox ConnectX 4X InfiniBand HCA, with firmware version 2.4.938, connected through a Mellanox 24-port MT47396 Infiniscale-III switch.

The operating system used is a Fedora Core 5 Linux kernel 2.6.20 implementation. The Open Fabrics distribution OFED-1.3 was used as the software stack for IB while using the mvapich-1.0.0 [12] implementation of MPI for the MPI background traffic. Oprofile 0.9.1 [15] was used to gather the performance counter results, using the applicable kernel support, compiled directly into our Linux kernel.

6. Performance Results

In this section, we will first present the basic performance results for single-stream and multi-stream communication, and then evaluate the effect of IB QoS provisioning on SDP based traffic compared to traditional TCP/IP on top of InfiniBand (IPoIB) under both weighted and priority queue QoS policies.

6.1 Baseline Performance Results

6.1.1. Single-stream Tests. Using the netperf [13] suite, the baseline performance of SDP and IPoIB protocols is shown in Figure 2 for large messages. We have used timed TCP stream tests for all of our socket-based benchmarks in this paper. SDP and IPoIB utilize a maximum of only 30.7% and 18.1% of the available hardware bandwidth on our platform (9828Mb/s and 5801Mb/s), respectively. Clearly, SDP has a smaller CPU utilization than IPoIB for large messages.

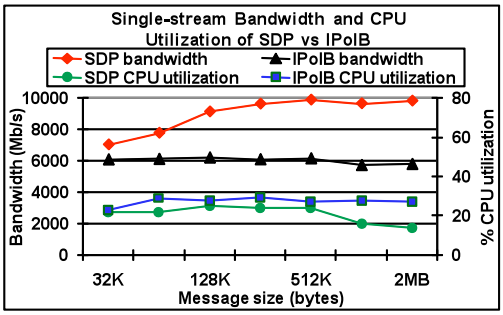

Figure 2. Single-stream SDP vs. IPoIB

6.1.2. Multi-stream Tests. The bandwidth results for multi-stream IPoIB and SDP are shown in Figure 3. Tests were performed with multiple streams of SDP and IPoIB using netperf with a 512KB message size. SDP was found to have the best overall aggregate performance with 5 streams or more, while the peak bandwidth of the IPoIB was found to be achievable with just two streams.

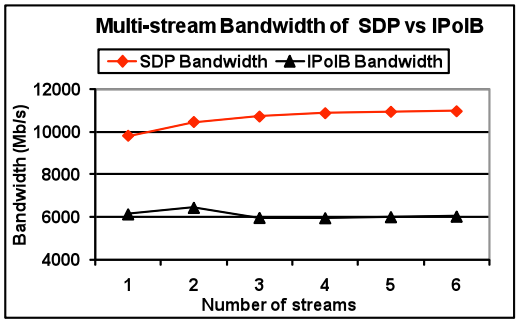

Figure 3. Multi-stream SDP vs. IPoIB

The performance of SDP and IPoIB when co-existing on a link is also of interest. To that end, the performance of SDP and IPoIB with varying numbers

of streams, evenly distributed, running over the same link is shown in Figure 4. For instance, the bandwidths shown for two streams represent the case of a single SDP stream running at the same time with one stream IPoIB over the link. Figure 4 shows that SDP is able to utilize a greater proportion of the available bandwidth than IPoIB, especially when more than two streams (one IPoIB, and one SDP) are functioning at the same time.

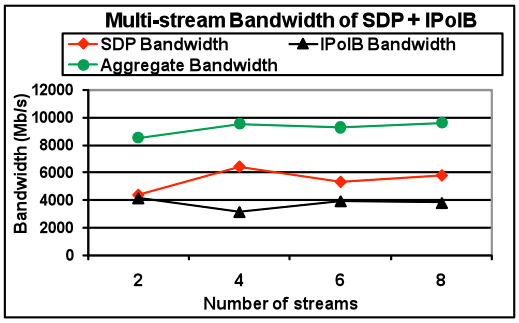

Figure 4. Multi-stream SDP and IPoIB when running together

6.2. Quality of Service Behaviour

The performance of SDP and IPoIB protocols can be adjusted in favour of either protocol using the available QoS provisioning, recently included in the latest version of the OFED software. In order to utilize the QoS features of the network, the network traffic must be sufficiently busy to create queues of packets at the NIC. Due to the inability of the socket-based streams to fully saturate the link on our platform (see the results in Section 6.1), an MPI both-way stream is run over the link during QoS testing in the following sections, providing a large amount of background traffic with minimum overhead, over which SDP and IPoIB traffic could be prioritized. MPI provides a low overhead communication mechanism, as fast as 1.7µs for small messages, while achieving more than 22Mb/s or ~70% of the maximum available network bandwidth on our platform when operating in IB DDR mode.

To observe whether the QoS ability of the ConnectX cards can be activated when the traffic is able to saturate most of the available card capacity, we use two identical MPI both-way streams over two nodes. We prioritize one stream over the other using both weighted and blocking QoS mechanisms. Figure 5 shows the bandwidth ratio of the high-priority stream over the low-priority stream, as well as the aggregate bandwidth of both streams. The X-axis (QoS factor) in Figure 5 is translated as high-priority stream weight for the weighted QoS mechanism, and high-priority queue's high_limit for the blocking QoS mechanism. For instance, a QoS factor of 4 means a queue weighting of 4 to 1.

We can clearly see the effect of QoS in Figure 5. In both QoS mechanisms, the bandwidth ratio of the higher-priority flow to the lower priority flow closely follows the QoS weighting for small QoS factor values (up to 16). However, as the QoS factor increases, the blocking QoS mechanism yields no significant effect. On the other hand, the weighting mechanism shows higher effect on the differentiated data flows and its ratio increases up to 70 for the maximum weight of 255. One important observation is that the aggregate bandwidth of these two streams remains constant, and is close to 22Mb/s regardless of the QoS factor. As we will analyze in Section 7, this is the maximum throughput that our nodes' chipsets can generate.

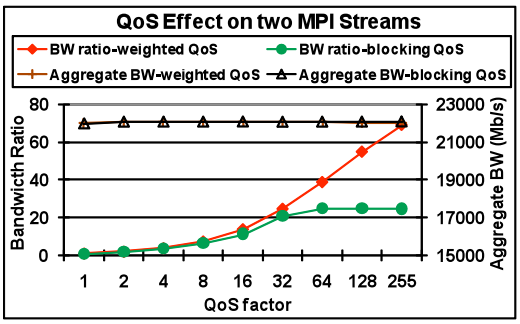

Figure 5. Effect of QoS on two MPI streams

We will focus on socket-based streams, which are not able to generate a high volume of traffic to put enough pressure on the cards. To do so, we employ one MPI stream as the background traffic and use three netperf TCP streams for SDP or IPoIB running over the MPI traffic. All different streams in our tests are running on different cores of the system. We examine the effect of enacting QoS on socket-based traffic and the total aggregate bandwidth using both weighted and blocking QoS mechanisms for a 512KB message size. For weighed QoS, we show the results for up to a 64 weighting of prioritized traffic over background traffic. Similarly, for blocking QoS, we go up to the high priority queue size of 64. The stream being assigned to the higher priority (blocking) queue has weightings only in that high priority queue, and is not serviced in the low priority queue.

6.2.1. QoS Effects on SDP Traffic. The results for enacting a higher QoS for 3-stream SDP traffic is shown in Figure 6 where we can see that the SDP performance improves by 33.0% between queue weightings of 1 and 4. Assigning a priority weighting above 4 results in a diminishing return, yielding a flat performance. For blocking QoS, this is the case for high_limit larger than 2.

The impact of prioritizing SDP over MPI on the aggregate traffic (SDP + MPI) is significant. We observe a 33.0% increase in SDP performance in exchange for a 15.6% decrease in the aggregate bandwidth for the 4 to 1 weighting. This drop in aggregate bandwidth is most likely due to the increased overhead of the SDP protocol over the MPI background traffic, resulting in more system related delays when SDP has higher priority.

From these results we can conclude that the average queue length at the NIC is of limited size, rarely growing to have more than 4 packets for weighted QoS case, and more than 2 packets for the blocking QoS case, at any one time. We will explore the reasons behind this in detail in Section 7.

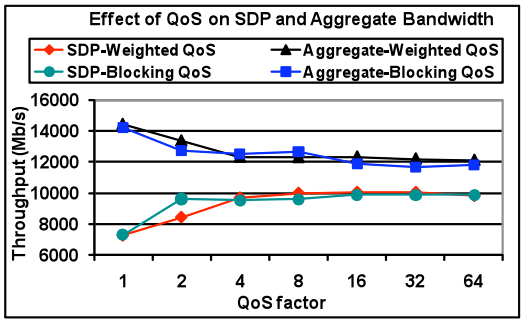

Figure 6. Effect of QoS on SDP, with MPI background traffic

6.2.2. QoS Effects on IPoIB Traffic. Here we examine the network performance when prioritizing IPoIB traffic over MPI background traffic. As shown in Figure 7, there is a 72.1% increase in IPoIB performance in exchange for a 36% decrease in the aggregate bandwidth for the 4 to 1 weighting. Using blocking QoS mechanism we see 64.2% improvement in IPoIB bandwidth accompanied by 38% drop in the aggregate bandwidth. As in the SDP case, we see diminishing returns after the weight value of 4 for weighted QoS and high_limit of 2 for blocking QoS.

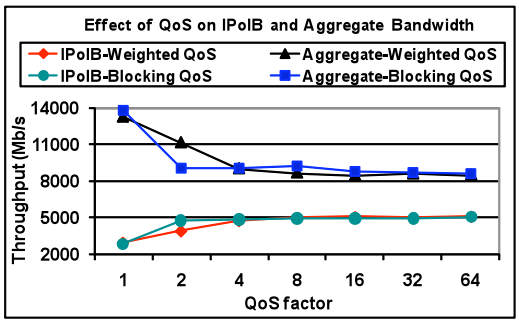

Figure 7. Effect of QoS on IPoIB, with MPI background traffic

6.2.3 Combined SDP and IPoIB QoS Behaviour. It is desirable to investigate the performance of SDP and IPoIB integrated traffic, as it could be used in a real data center. Due to space limitation, we only present the weighted QoS results for this case. For the case in which SDP is the higher priority traffic, we can see the effect on the bandwidth of the SDP streams in Figure 8. In the 2-stream case (in which 1-stream SDP and 1-stream IPoIB are running), SDP sees a maximum bandwidth improvement of 37.7%, while for the 4-stream case, a maximum improvement of 4.5% is seen.

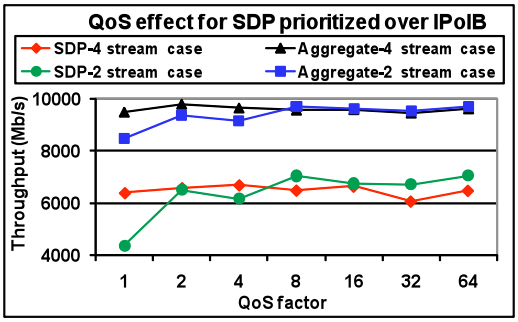

Figure 8. SDP performance when prioritized over IPoIB

IPoIB exhibits interesting behaviour when it is prioritized over SDP, as shown in Figure 9. In the 2-stream case, IPoIB performance drops by 30.8% when it is given higher priority over SDP traffic, while the aggregate bandwidth of the combined SDP and IPoIB traffic sees a 13.2% improvement. For the 4-stream case, the effect of prioritizing IPoIB is a maximum improvement of 3.8%. The 2-stream IPoIB performance drop is odd and we are currently investigating the reasons behind this.

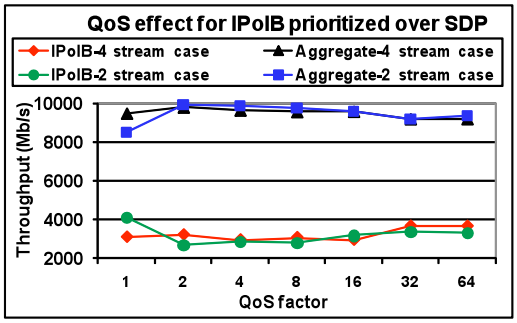

Figure 9. IPoIB performance when prioritized over SDP

Given that the QoS results for socket-based streams on our system show diminishing returns with relatively low overall QoS factors, we are prompted to investigate the performance bottleneck that is preventing the system from reaching full utilization and limiting the size of the queues at the NIC. Section 7 details the approach taken in identifying the performance bottlenecks.

7. Analysis of Performance Bottlenecks

In order to investigate the reasons behind the apparent performance bottleneck observed in Section 6, several different studies were undertaken to find the source of the performance bottleneck. Section 7.1 examines the memory pressure of the system. Section 7.2 studies chipset, and front-side bus behaviour with regards to memory access, and Section 7.3 examines the performance penalties incurred by using a buffered-copy mechanism.

7.1 Determining Memory Pressure

In order to rule out CPU utilization as a possible bottleneck, the systems were profiled using built-in CPU performance counters and the Oprofile [15] profiling application. All profiles were run for a set duration, covering the time to complete a timed bandwidth stream test. Therefore the number of clock ticks for different protocols are directly comparable as their execution time is identical.

Figure 10 compares the executed CPU micro-operations (μ-ops) and L2 cache misses for SDP and IPoIB streams, with and without background traffic. It is important to note that the overall number of cache misses in proportion to the total number of clock ticks within the time period is relatively low. This indicates that the total memory pressure created due to overhead associated with the protocol processing is fairly low. We would typically expect numbers in the millions if memory pressure were the cause of the performance bottleneck.

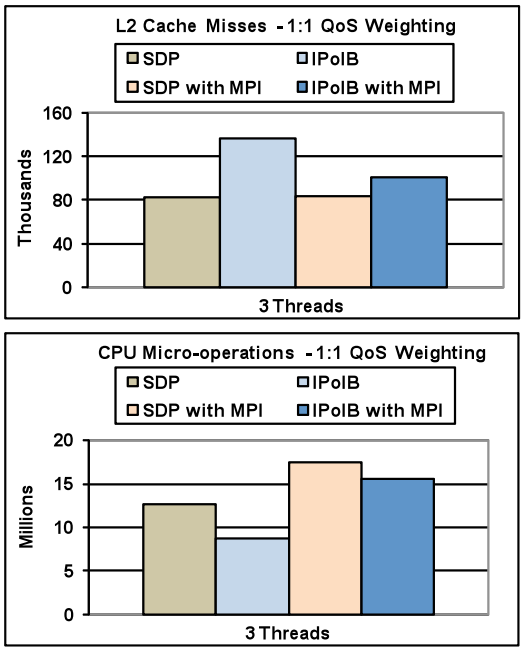

Figure 10. CPU micro-operations and L2 cache miss comparison

By examining the number of μ-ops that each protocol is performing at the CPUs we can observe the ratios of processing overhead that the given protocols incur. The 3-stream SDP with MPI combination produces the largest number of processed μ-ops, but even this level of μ-ops only creates an approximate 50% CPU load on the system. Examining a variety of other factors such as total instructions retired and ITLB behaviour indicate that there are no apparent sources of performance bottleneck at the CPU.

7.2 Chipset and Memory Access Architecture

Given the results from section 7.1, we devised a number of new tests to determine the possible source of the performance bottleneck. It was observed that introducing extra requests to memory had an immediate effect on the network performance. Testing the system with large message MPI traffic and small applications that made occasional requests to memory showed a marked performance drop in the MPI bandwidth. Given that we had already determined that the memory requests from the CPU operations themselves were not taxing the system, the DMA controller and memory bus were placed under scrutiny.

MPI works over IB verbs by registering DMA requests with the DMA controller, which then communicates directly with the IB NIC in delivering the requested data, given that the message size is greater than 64 bytes [17]. As performance counter analysis of the DMA controller is impossible with the system under test, experiments were performed to bypass the DMA controller and send data directly between the CPU and the NIC, as well as testing the effect of potential front-side bus blocking given that the chipset has two FSBs shared amongst four cores.

Running an MPI communication between cores on the same physical chip has been shown to degrade the performance of an inter-node communicating MPI process running on the other physical chip in the system. This occurs even when the message size of the intra-node communicating MPI process is small enough to fit into the local caches of the two cores of the same chip (not a shared cache). Therefore, the second physical chip's front side bus (FSB) is somehow blocked, whenever a cache miss is observed, and a DMA request is issued to the controller before any cores have a chance to respond with a cache hit.

We have conducted another test to find the source of this blocking. This test takes place with two MPI communications on the system at the same time, one intra-chip on the same physical chip, and another to a remote process through the NIC. We have altered the message sizes to determine the effect of the 4-port DMA controller on the performance of the system.

With the message sizes to the NIC being only 8 bytes, to bypass the use of the DMA controller, we find that the performance degradation of a 1KB intra-chip communication suffers by 13% drop in bandwidth. When the size of inter-node messages through the NIC is increased to 1MB (to use DMA) we find a performance decrease in the intra-chip communication of 47%. Given the assumption that the intra-chip communication is blocking only the FSB from the NIC communicating process, this should not have had a significant effect on the performance of the DMA based transfers to the NIC, therefore the intra-chip communication must be blocking the DMA-NIC transfer of data. In addition, the cache misses incurred by the inter-chip communication causes blocking of the FSB of both of the processor chips, as coherence messages are broadcast to the other physical processor via its FSB. This blocks any DMA requests from that chip during the time in which the FSB is in use.

This behaviour makes sense when compared to the performance bottleneck results for the IPoIB and SDP testing. From that testing, we were unable to observe a QoS effect for any queue sizes larger than 4 to 1. From this we can infer that queues very rarely or never build to levels exceeding 4 packets in the queue. The 4-ported DMA can theoretically serve 4 transactions that all complete to the IB NIC at very closely temporally located positions. Therefore, the maximum queue size for the system is a queue size of 4. IB Verbs and MPI layer bandwidth results show that all these chipset limitations have constrained the maximum achievable bandwidth through this chipset to be less than 23Gb/s.

7.3 Zero-copy vs. Buffered-copy Protocols

Kernel involvement and extra data copies used in socket protocols introduce other sources of overhead. SDP uses a buffered-copy mechanism in OFED-1.3 (a zero-copy SDP is planned to be released in OFED-1.4). Studies of zero-copy techniques on the previous generation of IB cards [3] have found up to a 2 times improvement in bandwidth utilizing asynchronous zero-copy mechanism for SDP. Therefore, if one extrapolates our buffered-copy performance of SDP, seeing a maximum of ~10000 Mbps, we can anticipate a throughput of zero-copy SDP close to that of MPI.

A throughput comparison of the buffered-copy SDP and IPoIB with both a zero-copy (Rendezvous) and a buffered-copy (Eager) MPI is shown in Figure 11 for 512KB messages. When MPI is configured to use buffered-copy for large messages the performance of MPI drops accordingly such that the performance of the buffered-copy SDP protocol is better than that of the Eager MPI and IPoIB. However, the buffered-copy SDP provides only slightly better than half the zero-copy MPI throughput.

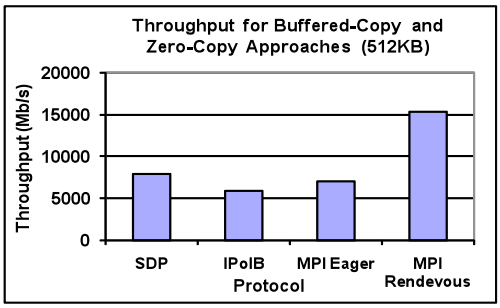

Figure 11. Buffered-copy vs. MPI zero-copy

8. SDR Mode Results

Given the potential limitation of our systems with regards to their ability to supply data at a sufficient rate to the NICs, the ConnectX HCAs were reduced in operational speed by a factor of 2 to see how QoS engages when the machine is theoretically able to saturate a slower network. This should allow us to create larger queues of packets at the NIC.

The new Single Data Rate (SDR) configuration results show a performance gain for more queue weightings than the DDR cases. Examining the QoS effect on two MPI streams in SDR configuration (similar to the test in Figure 5 for DDR), we find that the prioritized MPI stream achieves more than 80 times the non-prioritized MPI traffic when using weighted QoS with a weighting of 255. The aggregate bandwidth of both MPI streams reaches around 96% of the NIC capacity, compared to 70% for DDR.

Figure 12 shows the effect of weighted QoS on 3-stream SDP and 3-stream IPoIB when running over background MPI traffic. IPoIB and SDP are capable of occupying 36.6% and 49.7% respectively, of the maximum available bandwidth in SDR mode. In this mode, SDP and IPoIB bandwidths are increased by QoS weighting of up to 16 to 1, better than the 4 to 1 weighting in the DDR case. Similar to the DDR results, more loss in aggregate bandwidth is observed for IPoIB than for SDP. The results of blocking QoS are not shown, as they are very similar to those of weighted QoS, and thus better than the blocking QoS results in DDR. This is because in DDR configuration, queue lengths rarely build to levels that cause large amounts of blocking to occur for the MPI traffic, while in the SDR mode, blocking occurs more frequently as queue lengths are longer. As in the DDR mode, the low utilization by SDP and IPoIB relative to MPI can be partially attributed to their use of a buffered-copy mechanism, while MPI utilizes a zero-copy approach. We conclude that a faster node architecture coupled with a zero-copy SDP protocol will enhance the QoS impact in the DDR mode.

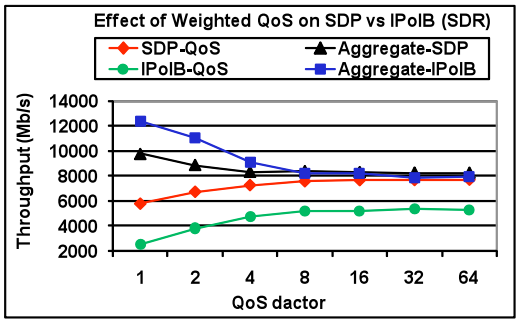

Figure 12. Effect of QoS on SDP vs. IPoIB (SDR), with MPI background traffic

8. Conclusions and Future Work

In this paper, we have analyzed the behaviour of SDP and IPoIB when managed using either of two IB QoS mechanisms. We investigated the performance gains that can be achieved when prioritizing SDP and IPoIB versus MPI background traffic as well as combined SDP and IPoIB traffic. We observed that if a reasonable amount of traffic is loading the NIC, the effect of QoS policies is clearly observable. However, performance bottlenecks in our systems inhibit the overall effectiveness of QoS due to not being able to saturate the link. We have identified the sources of such bottlenecks to be the chipset and specifically the memory access architecture, as well as the socket-based protocols' overhead such as extra data copies and kernel involvement. Despite the existence of bottlenecks, QoS provisioning has a definite effect on the performance of socket-based streams, with more apparent impact on SDP traffic than IPoIB.

QoS provisioning can be of great use on emerging systems capable of taking advantage of all of the available network bandwidth while handling the requisite processor and memory required for high-speed networks. We intend to investigate the performance of (zero-copy) SDP streams and data center applications with QoS provisioning on the newly released multi-core servers with more technologically advanced architectures.

9. Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), Canada Foundation for Innovation (CFI), Ontario Innovation Trust (OIT), Queen's University, and Mellanox Technologies. The authors would like to thank Mellanox for their technical support. We are grateful to the anonymous reviewers for their insightful comments.

10. References

[1] F.J. Alfaro, J.L. Sanchez and J. Duato. Studying the influence of the InfiniBand packet size to guarantee QoS. In *10th IEEE Symposium on Computers and Communications (ISCC'05),* pages 989-994, 2005.

[2] F.J. Alfaro, J.L. Sánchez and J. Duato. QoS in InfiniBand Subnetworks. *IEEE Transaction on Parallel and Distributed Systems,* 15(9):810-823, Sept. 2004.

[3] P. Balaji, S. Bhagvat, H. Jin and D.K. Panda. Asynchronous zero-copy communication for synchronous sockets in the Sockets Direct Protocol (SDP) over InfiniBand. In *6th Workshop on Communication Architecture for Clusters,* 2006.

[4] P. Balaji, S. Narravula, K. Vaidyanathan, S. Krishnamoorthy, J. Wu and D.K. Panda. Sockets Direct Protocol over InfiniBand in clusters: is it beneficial? In *2004 IEEE International Symposium on Performance Analysis of Systems and Software,* pages 28-35, 2004.

[5] P. Balaji, H.V. Shah and D.K. Panda. Sockets vs RDMA interface over 10-Gigabit networks: an in-depth analysis of the memory traffic bottleneck. In *Workshop on Remote Direct Memory Access (RDMA): Applications, Implementations, and Technologies,* 2004.

[6] D. Crupnicoff, S. Das and E. Zahavi. Deploying quality of service and congestion control in InfiniBand-based data center networks. Mellanox White Paper, http://www.mellanox.com/pdf/whitepapers/deploying_qos_wp_10_19_2005.pdf

[7] D. Goldenberg, M. Kagan, R. Ravid, M. Tsirkin. Transparently achieving superior socket performance using zero copy Socket Direct Protocol over 20Gb/s InfiniBand links. In *2005 IEEE International Conference on Cluster Computing,* pages 1-10, 2005.

[8] InfiniBand Trade Association, http://www.InfiniBandta.com/.

[9] InfiniBand Trade Association. InfiniBand Architecture Specification, Volume 1, October 2004.

[10] Mellanox Technologies: ConnectX Architecture, http://www.mellanox.com/products/connectx_architecture.php/.

[11] Message Passing Interface Forum: MPI, A Message Passing Interface standard, Version 1.2, 1997

[12] MVAPICH. http://mvapich.cse.ohio-state.edu/.

[13] Netperf Network Performance Benchmarking Suite, http://www.watson.org/~robert/freebsd/netperf/.

[14] Open Fabrics Alliance. http://www.openfabrics.org/.

[15] Oprofile. http://oprofile.sourceforge.net/.

[16] S.A. Reinemo, T. Skeie, T. Sødring, O. Lysne and O. Tørudbakken. An overview of QoS capabilities in InfiniBand, advanced switching interconnect, and Ethernet. *In IEEE Communications Magazine*, 44(7):32-38, July 2006.

[17] S. Sur, M.J. Koop, L. Chai and D.K. Panda. Performance analysis and evaluation of Mellanox ConnectX InfiniBand architecture with multi-core platforms. In *15th Annual IEEE Symposium on High-Performance Interconnects,* pages 125-134, 2007.