RDMA Performance in Virtual Machines using QDR InfiniBand on VMware vSphere® 5

RESEARCH NOTE

vmware®

Table of Contents

Introduction

This report details the results of remote direct memory access (RDMA) performance experiments using Mellanox QDR InfiniBand adapters in Red Hat Linux guests running on the VMware vSphere 5 virtualization platform. We demonstrate that latencies under 2µs and bandwidths of 26Gb/s can be achieved using guest-level RDMA with passthrough mode on VMware's ESXi hypervisor.

While networking, interconnect, and storage latencies have long been of concern within the High Performance Computing (HPC) community, latency is becoming increasingly important in the modern Enterprise as well. This is primarily due to the increasing importance of horizontally-scaled, high performance services like memcached, VMware vFabric GemFire, Oracle Coherence, Oracle RAC, IBM PureScale, and a wide variety of new, scale-out NOSQL storage layers.

VMware has published *Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs*[1], which provides detailed guidance for running latency-sensitive applications on vSphere. The interconnect aspects of that paper focus primarily on lowering latencies associated with virtualized Ethernet networks. This companion paper takes a different approach. Specifically, we begin with the lowest latency hardware and software used for high performance parallel distributed computing and assess how well this performs in a virtual environment.

InfiniBand

The use of InfiniBand is widespread within High Performance Computing environments due to its high bandwidth, low latency, and price-performance advantage. In native, non-virtualized environments (also called *physical* or *bare-metal* environments) we have measured QDR InfiniBand latencies as low as 1.35µs and bandwidths as high as 26Gb/s.

The low latencies cited above are achieved using polling to handle message completions rather than interrupts, which introduce additional overheads. While polling requires more CPU use, this is often justified in high performance environments in which fast message completion is critical to overall application performance. Since, in practice, communication and computation phases are not often overlapped, using CPU resources in this way to reduce messaging latencies is a sensible strategy.

InfiniBand hardware uses RDMA to directly copy data to or from an application's memory without the involvement of the operating system, caches, or CPU by using an on-board DMA controller. The RDMA protocol connects two communicating host channel adapters (HCAs) and then performs the transfer, informing the local HCA when the transfer completes. With the exception of the notifications sent to the application to signal the completion of a transfer, this is done without the knowledge of the operating system or hypervisor.

The InfiniBand standard defines two programming methods that are exposed to upper software layers: a channel and a memory semantic. The channel semantic is embodied in the Send operation which requires the participation of both endpoints to transfer a message: A receive operation is posted at the destination to specify a target buffer for the transfer and a send operation is posted at the source to initiate the transfer. In contrast, the memory semantic, which is embodied in RDMA Read and Write operations, allows the initiator of the transfer to specify the source or destination location without the involvement of the other endpoint.

In this paper we measured InfiniBand performance using both semantics and report in detail our Send/Receive and RDMA Read results. We also measured RDMA Write performance but it was generally found to be very similar to that of Send/Receive and so we have not reported those results here.

VMDirect Path I/O

In HPC environments, InfiniBand is most commonly accessed via a Message Passing Interface (MPI) user-level messaging library. The library makes use of the OpenFabrics Enterprise Distribution (OFED) Verbs layer, which mediates access to the hardware. Once buffer registration and other initializations are complete, RDMA transfers across an InfiniBand link can be queued and executed directly from user space with the InfiniBand hardware working

autonomously to perform data transfers without consuming host CPU resources. It is this kernel bypass mechanism, coupled with efficient RDMA InfiniBand hardware, which enables InfiniBand to deliver very low user-to-user messaging latencies.

To approximate the kernel bypass approach in a virtual environment, our tests were performed using VMDirect Path I/O[2], which is sometimes called *passthrough mode*. This ESXi hypervisor feature allows PCI devices to be directly exposed to a Guest OS rather than having access to the device mediated by the virtual networking stack as with a paravirtualized or emulated device. Figure 1 illustrates the concept.

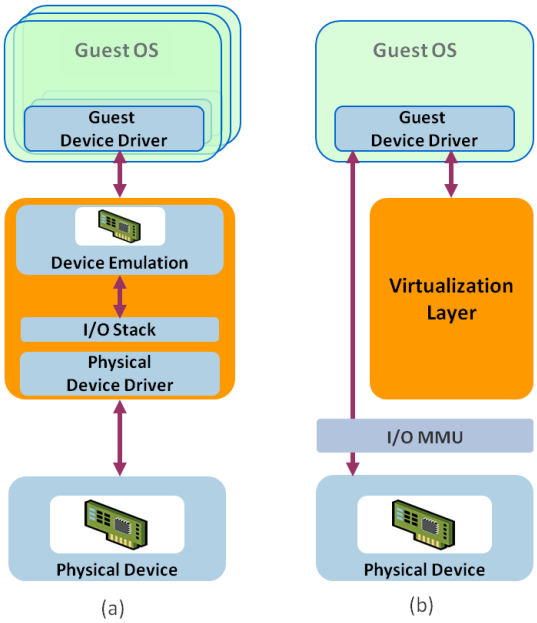

Figure 1: Paravirtualized (a) versus passthrough (b) device models

Note that while passthrough mode will deliver the lowest possible latencies, several virtualization features are disabled as a result of exposing real hardware to the virtual machine, most notably vMotion (live migration) and Snapshots. The VMware Office of the CTO is exploring another RDMA approach: the creation of a virtualized RDMA device that supports RDMA semantics while still allowing the use of vMotion and Snapshots. We expect to report on this effort in a later paper.

Performance Tests

We performed a series of bandwidth and latency tests using two hosts connected back-to-back with Mellanox ConnectX-2 QDR InfiniBand adapters. Native-to-native tests were run using Red Hat Linux on each host while the virtual-to-virtual tests were run with an ESXi hypervisor running on each host and along with a single virtual machine running the same Red Hat version as the Guest OS. All tests were run with both polling and interrupt-based I/O completions.

Full details on hardware and software configurations are given in the Configuration Details section.

Bandwidth Results

Bandwidths were measured over a range of message sizes using the OFED *send_bw* (Send/Receive) and *read_bw* (RDMA Read) tests between two four-vCPU virtual machines, each using an InfiniBand HCA in passthrough mode. Both tests were run using interrupt and polling completions.

Polling

Using polling, virtual-to-virtual bandwidths were essentially identical to those seen in the native-to-native cases for both Send and RDMA Read, as illustrated in Figure 2.

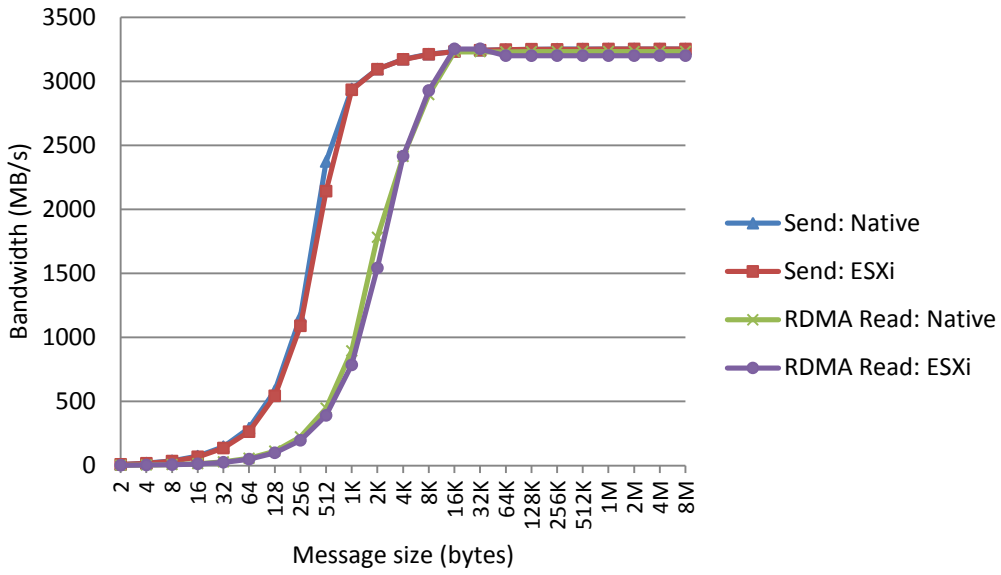

Figure 2: Send and RDMA Read bandwidths using polling completions

Interrupts

Using interrupts, we measured the bandwidths shown in Figure 3. In both the Send and RDMA Read cases we see similar behavior: a slight increase in the half-power point (the smallest message size that can achieve half the peak bandwidth) in the virtual case and a distinct sag in delivered bandwidth at high message sizes that recovers and then matches the native asymptotic maximum bandwidth for 8MB messages. The half-power point changes are small: approximately 350 bytes versus 512 bytes in the Send case and 1725 bytes versus 2200 bytes in the RDMA Read case.

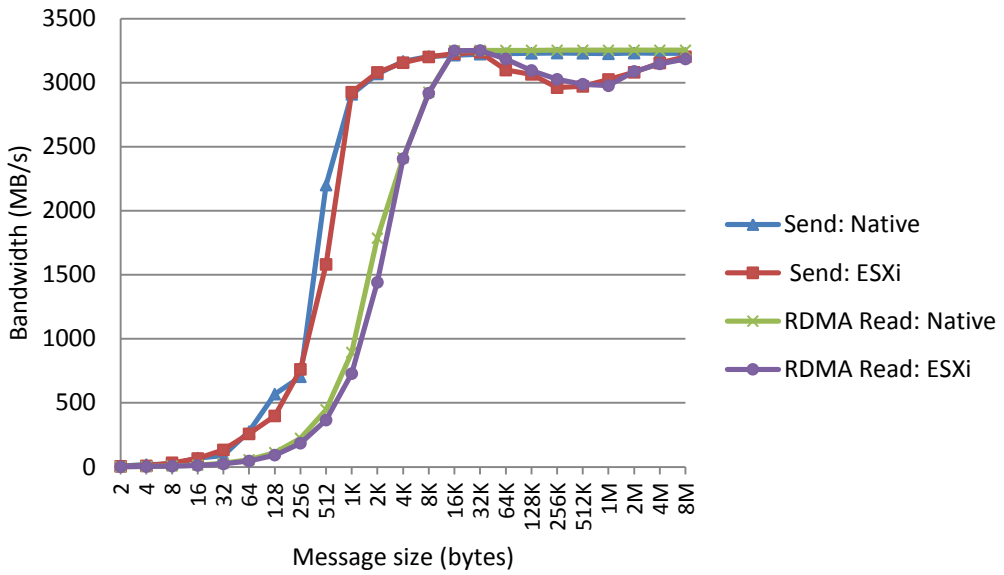

Figure 3: Send and RDMA Read bandwidths using interrupt completions

Latency Results

Latencies were measured over a range of message sizes using the OFED *send_lat* (Send/Receive) and *read_lat* (RDMA Read) tests between two virtual machines, each using an InfiniBand HCA in passthrough mode.

Polling results were generated using an experimental version of ESXi, designated as "ESXi ExpA" and described in the Configuration Details section. Polling results were measured between two four-vCPU virtual machines.

Interrupt results were generated first with the released version of ESXi and then with a second experimental version of ESXi, designated as "ESXi ExpB" in the following graphs and tables. Single vCPU virtual machines were used for the interrupt completion tests to keep the environment as simple as possible for those experiments.

Polling

Applications that are extremely sensitive to latency will most often rely on polling for I/O completions so as to achieve the lowest possible latencies. Figure 4 shows Send latency results using polling and Table 1 provides more detailed information for small message sizes. It can be seen that the virtual case begins with a 0.4µs overhead relative to native for the smallest message sizes which then increases to about 2.6µs before disappearing for message sizes of 256 bytes and greater.

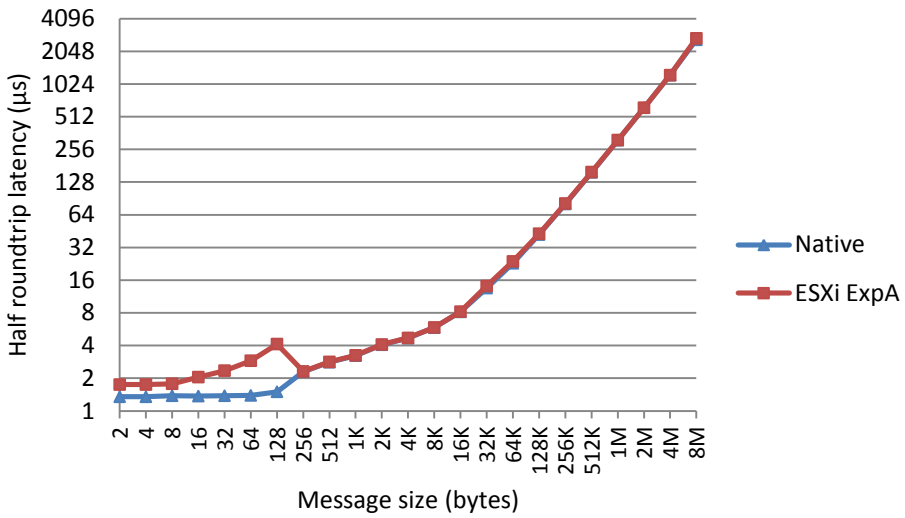

Figure 4: Send latencies using polling completions

The anomalous behavior shown for message sizes from 16 to 128 bytes – where virtual performance increasingly diverges from native performance – is believed to be due to an unoptimized setting in the InfiniBand implementation for the VMware virtual environment. While we were unable to verify this prior to publication, Mellanox testing suggests that this is the case and parameter changes would smooth the latency curve and remove the peak currently seen at 128 bytes.

MsgSize (bytes)	Native	ESXi ExpA
2	1.35	1.75
4	1.35	1.75
8	1.38	1.78
16	1.37	2.05
32	1.38	2.35
64	1.39	2.9
128	1.5	4.13
256	2.3	2.31

Table 1: Send latencies in microseconds for small message sizes

RDMA Read performance (Figure 5) shows a constant 0.7µs overhead in the virtual case for message sizes from two bytes to 32KB. The absolute overhead then increases for messages larger than 32KB, although the relative overhead for these message sizes remains under 1.7%. This overhead had no discernible effect on delivered bandwidth at those message sizes, as was illustrated in Figure 2. Latencies for message sizes from 2 to 256 bytes are shown in Table 2.

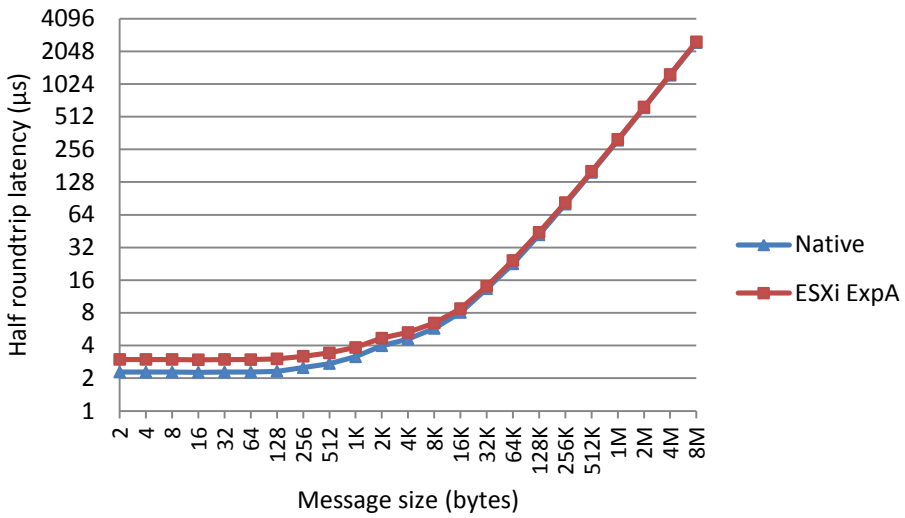

Figure 5: RDMA Read latencies using polling completions

MsgSize (bytes)	Native	ESXi ExpA
2	2.28	2.98
4	2.28	2.98
8	2.28	2.98
16	2.27	2.96
32	2.28	2.98
64	2.28	2.97
128	2.32	3.02
256	2.5	3.19

Table 2: RDMA Read latencies in microseconds for small message sizes

Interrupts

While polling will often be used for applications requiring lowest possible latencies, it is also important to understand performance using interrupt completions for several reasons. First, some applications – those that can overlap computation and communication – can use interrupt-driven completions to allow the CPU to perform useful work while waiting for communication operations to signal their completion. Second, measuring interrupt overheads in passthrough mode can help gain insight into basic overheads introduced by the virtualization layer because relatively little of the virtualization stack is actually active when using passthrough, giving a clearer view of the causes of these overheads. In addition, reductions in these overheads will improve VMware's handling of low latency workloads in the general case in which passthrough mode is not used.

We tested interrupt completions in several virtual configurations as shown in Figure 6. Because the Send results were similar, they have been omitted.

The Native curve represents bare-metal, non-virtualized performance using OFED *read_lat* with Red Hat 6.1. The ESXi curve shows the virtual-to-virtual latencies achieved using the hypervisor described in the Configuration Details section with virtual machines running Red Hat 6.1 and with InfiniBand in passthrough mode. This configuration was

tuned as described in the Configuration Details section. The virtualization overhead measured was roughly 7.5µs for message sizes from two bytes to 32Kbytes. Table 3 shows the data for the smallest message sizes.

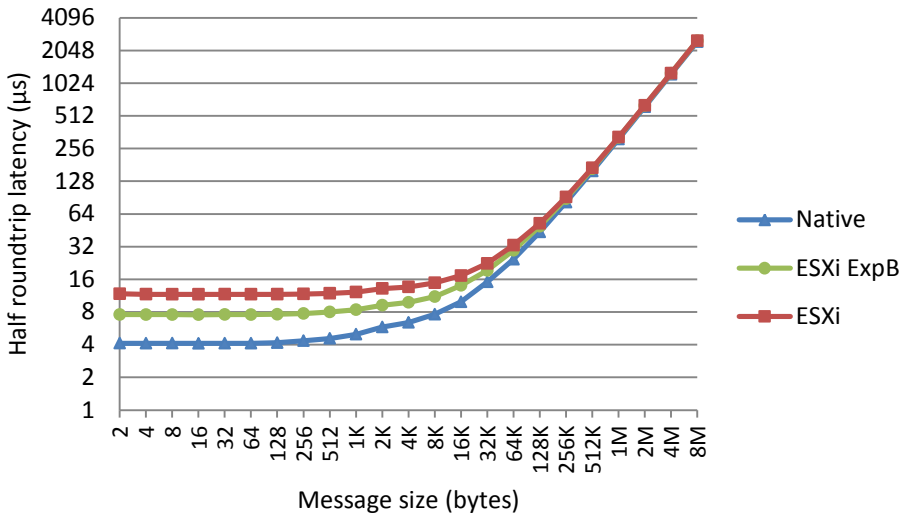

Figure 6: RDMA Read latencies using interrupt completions

The final curve in Figure 6 – labeled ESXi ExpB -- represents the best virtual-to-virtual performance we have measured to date using InfiniBand in passthrough mode with interrupt completions. To achieve these improvements over our reported vSphere 5 results, several ESXi changes that optimized interrupt delivery were prototyped by members of VMware's product engineering teams. These changes reduced the virtual-to-virtual latencies by roughly 4.1µs to bring them to within about 3.4µs of native latencies for message sizes from two to 8K bytes.

MsgSize (bytes)	Native	ESXi ExpB	ESXi
2	4.13	7.57	11.76
4	4.12	7.56	11.66
8	4.13	7.56	11.64
16	4.11	7.55	11.64
32	4.12	7.58	11.66
64	4.12	7.56	11.65
128	4.17	7.61	11.66
256	4.35	7.74	11.73

Table 3: RDMA Read latencies in microseconds for small message sizes

Summary

We have demonstrated that by using InfiniBand with VMDirect Path I/O (passthrough mode), vSphere 5 can deliver bandwidths close to those seen in the native case over a very wide range of message sizes. In addition, we have shown that very low latency communication can be achieved with vSphere: We measured latencies as low as 7.57µs

using interrupt completions and 1.75µs using polling.

While the achieved virtual latencies represent significant overhead relative to native performance, the absolute latencies are expected to be adequate for a significant number of HPC applications.

The work reported here should be considered our initial results. As VMware continues to work to reduce latencies, we expect the performance of passthrough RDMA to improve, further expanding the applicability of virtualization for High Performance Computing.

Configuration Details

Hardware

HP ProLiant ML350 G6, two-socket E5620 (Westmere) 2.4 GHz, 12GB

Mellanox ConnectX-2 MT26428 QDR HCA in PCIe 2.0 8x slot

QSPF cable direct-connected between hosts (no switch)

BIOS Settings

Power Regulator for ProLiant: HP Static High Performance Mode

Intel QPI Link Power Management: disabled

Processor Hyper-Threading: enabled

Intel Virtualization Technology: enabled

Intel VT-d: enabled

Software

64-bit RHEL 6.1 (native and virtual) with OFED 1.5.3 and with nohz=off.

InfiniBand device configured in ESX as a passthrough device using VMDirect Path I/O.

All measurements were made using the OpenFabrics Enterprise Distribution (OFED) Performance Tests in Reliable Connected (RC) mode. Each reported data point represents the mean of 10000 benchmark iterations at that message size.

Bandwidth tests were run with ESXi 5.0: Release Candidate build 414385 (ESXi 5.0 was officially released using build 469512 after these experiments concluded) using vSphere 5 with default settings. We designate this configuration as "ESXi".

Two experimental versions of ESXi were used for latency tests. The polling tests were run with a version of ESXi that included one unreleased patch to improve latency in passthrough mode. We designate this as the "ESXi ExpA" configuration. Interrupt-based testing was done first with the "ESXi" configuration to establish a baseline and then with an experimental version of ESXi that included a number of additional, unreleased interrupt-related optimizations beyond what was included in the ExpA configuration. We designate this configuration as "ESXi ExpB".

For both ExpA and ExpB tests, ESXi was configured so as not to de-schedule the VM when its vCPU was idle to avoid incurring any latency overhead due to rescheduling the VM. This can be accomplished from within the vSphere Client by setting monitor_control.halt_desched=false under *VM Settings -> Options tab -> Advanced General -> Configuration Parameters*.

References

1. *Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs*

   http://www.vmware.com/files/pdf/techpaper/VMW-Tuning-Latency-Sensitive-Workloads.pdf

2. *Configuration Examples and Troubleshooting for VMDirectPath*

   http://www.vmware.com/pdf/vsp_4_vmdirectpath_host.pdf

About the Authors

Josh Simons is a Senior Staff Engineer in the Office of the CTO, leading an effort to bring the values of virtualization to High Performance Computing (HPC). He has over 20 years of HPC experience, primarily at Sun Microsystems and Thinking Machines Corporation, where he held a wide variety of technical positions.

Adit Ranadive is a 4th year PhD student in Computer Science at the Georgia Institute of Technology. His research focuses on using newer interconnects like InfiniBand in virtualized systems and effectively managing these resources to meet application SLAs. His work can be found on his website through the College of Computing: http://www.ccgatech.edu/~adit262.

Bhavesh Davda works in the Office of the CTO, focusing on SR-IOV, VMDirectPath I/O, RDMA and the virtualization of applications that have historically been challenging to virtualize: low latency, real-time/low jitter, high-rate packet processing and HPC applications. Previously, he was Senior Manager for the vSphere Networking R&D team , working on VMware's end-to-end virtual networking stack, including guest OS drivers for paravirtual NICs, virtualization of paravirtual and emulated NICs in the hypervisor, and the vmkernel-based TCP/IP stack used by vMotion, Fault-Tolerance, NFS and iSCSI.

Acknowledgements

The bulk of the work reported here was performed by Adit Ranadive during his 2011 internship in the Office of the CTO at VMware. Thanks as well to Ben Serebrin, Garrett Smith, and Boon Seong Ang for their guidance and contributions during the tuning phase of this project.