# High Performance MPI Library over SR-IOV Enabled InfiniBand Clusters*

Jie Zhang, Xiaoyi Lu, Jithin Jose, Mingzhe Li, Rong Shi, Dhabaleswar K.(DK) Panda
Department of Computer Science and Engineering
The Ohio State University
Email: {zhanjie, luxi, jose, limin, shir, panda}@cse.ohio-state.edu

***Abstract*—Virtualization has become a central role in HPC Cloud due to easy management and low cost of computation and communication. Recently, Single Root I/O Virtualization (SR-IOV) technology has been introduced for high-performance interconnects such as InfiniBand and can attain near to native performance for inter-node communication. However, the SR-IOV scheme lacks locality aware communication support, which leads to performance overheads for inter-VM communication within a same physical node. To address this issue, this paper first proposes a high performance design of MPI library over SR-IOV enabled InfiniBand clusters by dynamically detecting VM locality and coordinating data movements between SR-IOV and Inter-VM shared memory (IVShmem) channels. Through our proposed design, MPI applications running in virtualized mode can achieve efficient locality-aware communication on SR-IOV enabled InfiniBand clusters. In addition, we optimize communications in IVShmem and SR-IOV channels by analyzing the performance impact of core mechanisms and parameters inside MPI library to deliver better performance in virtual machines. Finally, we conduct comprehensive performance studies by using point-to-point and collective benchmarks, and HPC applications. Experimental evaluations show that our proposed MPI library design can significantly improve the performance for point-to-point and collective operations, and MPI applications with different InfiniBand transport protocols (RC and UD) by up to 158%, 76%, 43%, respectively, compared with SR-IOV. To the best of our knowledge, this is the first study to offer a high performance MPI library that supports efficient locality aware MPI communication over SR-IOV enabled InfiniBand clusters.**

***Keywords*—SR-IOV, IVShmem, Virtualization, InfiniBand, MPI**

## I. Introduction

Virtualization offers attractive techniques for Cloud Computing, given the capabilities to scale resources on virtual machines with the ease of system management and administration. They provide desirable features for users to meet their various resource utilization requirements, including server consolidation, performance isolation, security and live migration [1]. The advantages of Cloud Computing with virtualization are certainly attractive to HPC users. High-performance computing in the Cloud has already been widely adopted in the industry. For instance, Cloud providers, such as Amazon's Elastic Compute Cloud (EC2) [2], rely on virtualization to consolidate computing, storage and networking resources for applications with required Quality of Service guarantees on the shared underlying infrastructure.

Even though virtualization has gained significant momentum in Cloud Computing, running HPC applications on Cloud systems with good performance is challenging. One of the biggest hurdles is the lower performance of virtualized I/O devices [3]. This limits the adoption of virtualized Cloud Computing systems for HPC applications. In the traditional HPC domain, high performance MPI libraries such as MVAPICH2 [4] and OpenMPI [5], are able to provide sub-microsecond latencies. However, recent studies [3] have shown that achieving similar performance in virtualization based Cloud Computing systems is still a challenge.

To address this issue, the community has recently introduced an enhanced networking capability, Single Root I/O Virtualization (SR-IOV) [6], which offers a high-performance alternative for virtualizing I/O devices in Cloud Computing systems. SR-IOV provides higher I/O performance and lower CPU utilization compared to the traditional software-based virtualization solutions. SR-IOV enables a PCIe device to present itself as multiple virtual devices and each virtual device can be dedicated to a single VM. Our earlier study [3] shows that SR-IOV can attain near to native performance for inter-node MPI point-to-point communication. Currently, SR-IOV has been already used in production Cloud Computing systems, such as the C3 and I2 instance types (using 10GigE) in Amazon EC2 [2], where this feature shows higher packet per second performance and lower network jitter.

Even though SR-IOV is enabled in these systems, inter-VM communications within the node also have to use SR-IOV, leading to performance overheads, which is the main drawback of SR-IOV that it does not support VM locality aware communication. On the other hand, high performance MPI libraries in the HPC domain typically use shared memory based schemes for intra-host communication. Similarly, inter-VM shared memory (IVShmem) [7] is a novel feature proposed for inter-VM communication, and offers shared memory backed communication for VMs within a given host. Based on the features that IVShmem provides, our recent performance evaluations [8] exhibit that IVShmem can dramatically improve the performance of intra-node inter-VM communications compared to SR-IOV on virtualized InfiniBand clusters.

Based on these new features provided by both SR-IOV and IVShmem, current MPI libraries need to be redesigned or enhanced to fully exploit the benefits of these features on SR-IOV and IVShmem enabled InfiniBand clusters, so that they can meet the demand of high performance communication on virtualization environments. In addition, as bridges between MPI libraries and virtualized I/O devices, SR-IOV and IVShmem channels may have different performance characteristics compared to native network and shared memory channels, which need to be explored further at the MPI level to deliver optimal communication performance. All of these issues lead to the following broad challenges:

*This research is supported in part by National Science Foundation grants #OCI-1148371, #CCF-1213084 and #CNS-1347189.

1) How to design a high performance MPI library to efficiently support locality aware MPI communication over SR-IOV enabled InfiniBand clusters?
2) How to further optimize communications in MPI libraries for the new channels of SR-IOV and IVShmem in virtualized InfiniBand clusters?
3) What are the performance impacts of different InfiniBand transport protocols (RC and UD), when run in SR-IOV mode?
4) How much performance improvement can be achieved by new design and enhancements on point-to-point operations, collective operations and applications?

To address these challenges, we first propose a high performance MPI library based on MVAPICH2 for SR-IOV enabled InfiniBand clusters. Our new design is able to dynamically detect VM locality information and coordinate data movement between SR-IOV and IVShmem channels. Through these, the enhanced MVAPICH2 library can efficiently support locality aware communication and improve the overall MPI communication performance further on virtualized environments. Then, based on our proposed design, we further optimize data communications in SR-IOV and IVShmem channels by investigating the performance impact of core mechanisms and parameters inside the MPI library. Finally, we conduct comprehensive performance evaluations for our proposed design on InfiniBand clusters featuring SR-IOV. The results show that our proposed design can improve the performance of point-to-point and collective operations, and applications by up to 158%, 76%, 43%, respectively. We further explore the impact of different InfiniBand transport protocols (RC and UD) on MPI communications over SR-IOV enabled InfiniBand cluster. In summary, this paper makes the following key contributions:

1) Analyze multiple VM locality detection alternatives (static or dynamic) for MPI libraries on virtualized environments and propose a dynamic VM locality detection design to achieve locality-aware MPI communication
2) Analyze and optimize data communications in SR-IOV and IVShmem channels in the proposed design
3) Explore the impact of different InfiniBand transport protocols (RC and UD) on MPI communications over SR-IOV enabled InfiniBand clusters
4) Comprehensive performance evaluations showing the benefits of the proposed design

To the best of our knowledge, this is the first paper that attempts to offer a high performance MPI library, which can support efficient locality aware MPI communication over SR-IOV enabled InfiniBand clusters.

The rest of the paper is organized as follows. Section II provides an overview of SR-IOV, IVShmem, and InfiniBand. Section III presents our proposed design, and discusses the detailed designs of key components. At the end of this section, we discuss communication optimizations for IVShmem and SR-IOV channels in MPI library. Section IV presents the evaluation methodology, and then presents the performance evaluation results on point-to-point, collective operations, different InfiniBand transport protocols and representative applications. We discuss the related work in Section V, and conclude this paper in Section VI.

## II. Background

### A. Single Root I/O Virtualization (SR-IOV)

Single Root I/O Virtualization (SR-IOV) [6] is a PCI Express (PCIe) standard which specifies the native I/O virtualization capabilities in PCIe adapters. SR-IOV is applicable when the PCIe interface works in a single server environment. As shown in Figure 1(a), SR-IOV allows a single physical device, or a Physical Function (PF), to present itself as multiple virtual devices, or Virtual Functions (VFs). Each virtual device can be dedicated to a single VM through the PCI pass-through, which allows each VM to directly access the corresponding VF. Hence, SR-IOV is a hardware-based approach to implement I/O virtualization. Furthermore, VFs are designed based on the existing non-virtualized Physical Functions (PFs); hence, the drivers of the current adapters can also be used to drive the VFs in a portable manner.

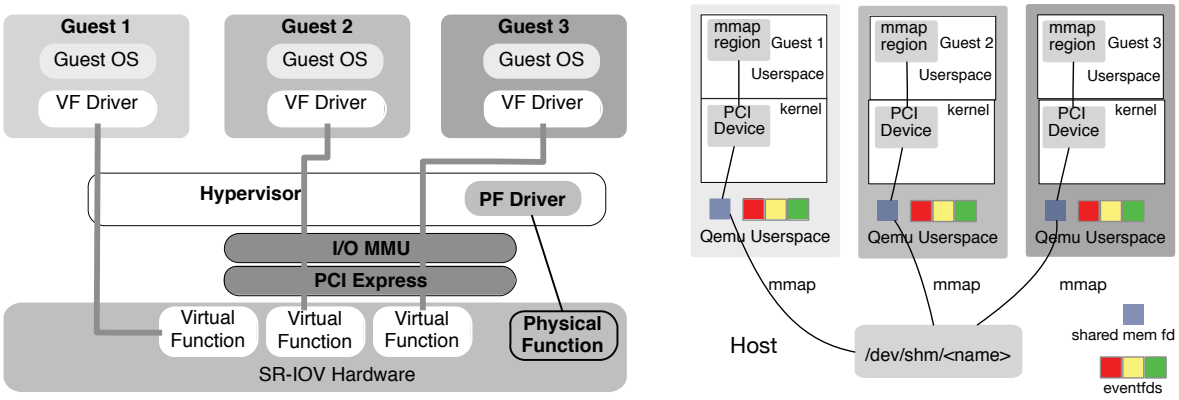
### B. Inter-VM Shared Memory (IVShmem)

IVShmem (e.g. Nahanni) [7] provides zero-copy access to data co-resident on VM shared memory, for guest-to-guest and host-to-guest communications on the KVM platform. IVShmem is mainly designed and implemented in system calls layer and its interfaces are visible to user space applications as well. Figure 1(b) shows that IVShmem contains three components: the guest kernel driver, the modified QEMU supporting PCI device, and the POSIX shared memory region on the host OS. The shared memory region is allocated by host POSIX operations and mapped to QEMU process address space. The mapped memory can be used by guest applications by being mapped to guest user space. Through supporting zero-copy, IVShmem can achieve better performance.

### C. InfiniBand

InfiniBand [9] is an industry standard switched fabric designed for interconnecting nodes in HPC clusters. The TOP500 rankings released in June 2014 indicate that more than 44% of the computing systems use InfiniBand as their primary interconnect. Remote Direct Memory Access (RDMA) is one of the main features of InfiniBand, which allows software to remotely access memory contents of another remote process without any involvement at the remote side.

When a connection between two channel adapters is established, five kinds of transport layer communication protocols defined by the InfiniBand specification can be selected: Reliable Connection (RC), Reliable Datagram (RD), Unreliable Connection (UC) , Unreliable Datagram (UD) and Raw Datagram. RC and UD are two common protocols. RC is the most popular transport service for implementing MPI over InfiniBand. For connection-oriented RC, a QP must be dedicated to communicating with only one other QP. That is to say, each peer communicating with N other peers thus needs to create at least N QPs. RC provides RDMA capability, atomic operations, and reliable service. Data transfer between two entities using RC receives acknowledgment. UD is a connection-less and unreliable transport without acknowledgement. It is the most basic transport specified for InfiniBand. The advantage is that a single UD QP can communicate with any number of other UD QPs. However, UD does not guarantee reliability or message ordering.




(a) SR-IOV Mechanism

(b) Inter-VM Shmem Mechanism [7]

Fig. 1: Overview of Inter-VM Shmem and SR-IOV

## III. Proposed Design

In this section, we first present the overview of our proposed design for high performance MPI library. Then we discuss and analyze different locality detection approaches and describe our design of locality detector. Next, we present the design details of communication coordinator. In Section III-D and Section III-E, we discuss the communication optimization for IVShmem and SR-IOV channel, respectively.

### A. Design Overview

Our design is based on MVAPICH2, an open-source MPI library over InfiniBand. For portability reasons, it follows a layered approach, as shown in Figure 2(a). The Abstract Device Interface V3 (ADI3) layer implements all MPI-level primitives. Multiple communication channels provide basic message delivery functionalities on top of communication device APIs. There are two types of communication channels available in MVAPICH2: a shared memory channel communicating over user space shared memory to peers hosted in the same host and a network channel communicating over InfiniBand user-level APIs to other peers.

Without any modification, default MVAPICH2 can run in virtualization environment. However, VMs running on same host cannot use shared memory channel (SMP) for communication, which can lead to severe performance limitations. In our proposed high performance MPI library, as shown in Figure 2(b) we add two components, which are 'Communication Coordinator' and 'Locality Detector' between ADI3 layer and channel layer. In the channel layer, we integrate IVShmem channel into the library as well as the SR-IOV channel. Communication Coordinator is responsible for selecting communication channel in lower channel layer, while Locality Detector maintains the information of local VMs on the same host. Communication Coordinator makes a decision on going through a channel by utilizing Locality Detector to identify whether the communicating VMs are co-resident on the same host or not. If they are co-resident in a given host, Communication Coordinator will select IVShmem channel for the communication between these co-located VMs. Otherwise, it will go through SR-IOV channel.

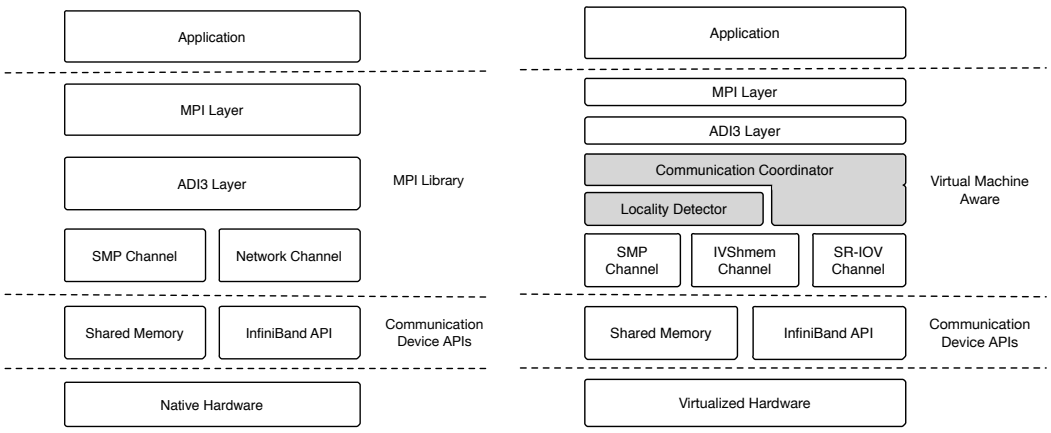
The locality detector will further identify whether there are multiple processes running in the same VM, then the Communication Coordinator will select default SMP channel in VM (not host) for the communication between those processes. Since this paper mainly focuses on communication optimization for co-resident VMs, and also default SMP channel in VM is similar with the one in host, we will not discuss this channel in details.

### B. Locality Detector

Given the above functionality provided by IVShmem in Section II-B, the way to identify co-resident VMs among all VMs becomes a critical problem. Basically, there are two locality identification alternatives we can evaluate.

The first one is a static method, which is mainly used when the information of co-resident VMs is preconfigured by the administrator, and it is assumed that the membership of co-resident VMs do not change during the communication afterwards. Thus, the VM locality information is already available when launching the MPI jobs. The advantage of this approach is that the processes can be directly re-mapped in the VM layer based on the above information, with little overhead. But the problem is that without intervention from the administrator, the static information cannot be dynamically updated.

The other one is dynamic detection, that is MPI jobs will dynamically detect the VMs running on the same host. And according to who initiates the process, there are two ways to implement it: Since privileged domain plays a center role in virtualization environment, we can use it to periodically gather VM information on the same host. VM peers advertise their membership information, such as presence and absence to all other VMs running on the same host. This approach is asynchronous and needs centralized management from privileged domain. However, the period between two periodical gather operations needs to be configured properly. If the period is set longer than needed, it cannot bring accurate co-residency information in time. If it is too short, it might lead to unnecessary probing and thus waste undesirable CPU cycles. The second approach works in synchronous mode. When a VM takes a significant action, it will notify related VMs to update the co-residency information. Thus, the updates are immediate upon the occurrence of the corresponding events. In comparison, the first approach periodically collects the status from co-resident VMs and thus introduces delayed update wasting CPU cycles, and also potential inconsistency, while for the




(a) Native Environment (b) Virtualization Environment

Fig. 2: MVAPICH2 Stack Running in Native and Virtualization Environments

second approach, it is possible that co-residency information of multiple VMs changes concurrently [10].

To take advantage of the dynamic detection approach, we propose a locality detector component. Based on IVShmem support, we create a VM list structure on the shared memory region of each host. And each process will write its own membership information into this shared VM list structure according to its global rank. For example, consider launching an 8-process MPI job, one process per VM. Let ranks 0, 1, 4, and 5 run on the same host (e.g. host1), as shown in Figure 3, and other 4 ranks run on another host (e.g. host2). Then the four VMs (ranks 0, 1, 4 and 5) will write their own membership information into positions 0, 1, 4 and 5 of VM list on host1 correspondingly. Other positions will be left blank. Similarly, other four VMs write at positions 2, 3, 6 and 7 of VM list on host2. In this case, the local number of processes on host1 can be acquired by checking and counting whether the membership information has been written or not. Similarly, their local ordering will be maintained by their positions in VM list. Therefore, the written membership information on the same VM list indicates that they are co-resident.

Since byte is the smallest granularity of memory access without lock. In our proposed design, the VM list is designed by using multiple bytes. Each byte will be used to tag each VM. This guarantees that multiple VMs on the same host are able to write membership information on their corresponding positions concurrently without introducing lock&unlock operations. This approach reduces the overhead of locality detection procedure. Moreover, the proposed approach will not introduce much overhead of traversing the VM list. Take a one million processes MPI job for instance, the whole VM list only occupies 1 Mega bytes memory space. Therefore, it brings good scalability on virtualized MPI environment.

### C. Communication Coordinator

The default MVAPICH2 stack, as shown in Figure 2(a), can also be deployed in virtualization environment, whereas the processes running on different VMs can not communicate through shared memory channel, even though they are co-located on the same host. However, with the help of Locality Detector and VM lists which are created and maintained in

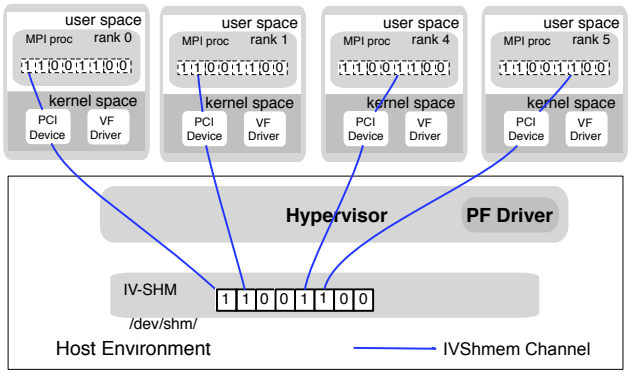
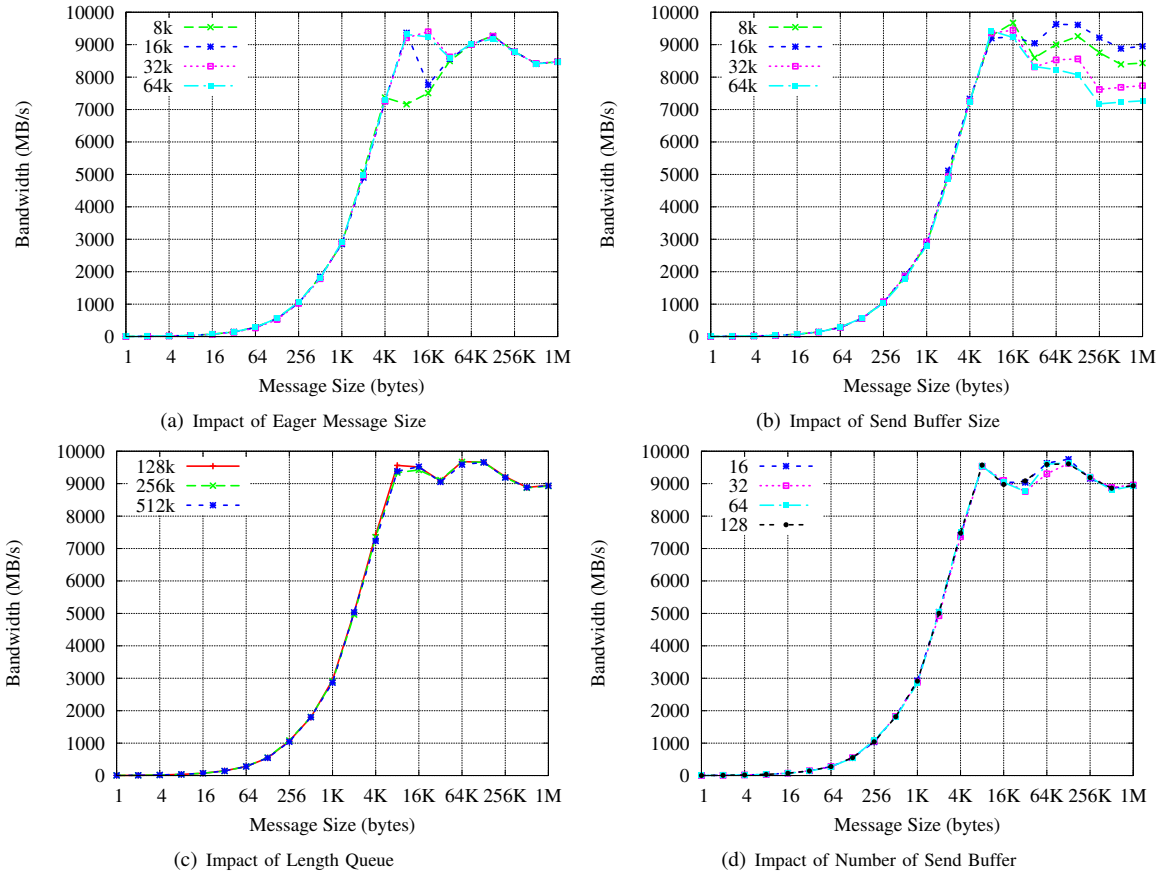



Fig. 3: Virtual Machine Locality Detection

shared memory region, the co-resident VMs can be dynamically identified. Another key component in our proposed design is called Communication Coordinator, as shown in Figure 4. It is responsible for capturing the communication channel requests coming from the upper layer and carrying out the channel selection by checking the membership information provided by the Locality Detector. If the communicating processes are co-resident, Communication Coordinator will schedule them to communicate through IVShmem channel. Otherwise, they will go through SR-IOV channel. For example, we can see in Figure 4 that, Guest 1 and Guest 2 are co-located on the same host. MPI process rank 1 and rank 4 are running on Guest 1 and Guest 2, respectively. They can access the same VM list located in IVShmem region by mapping the IVShmem region to their own user space. By checking the flag at position 4 of VM list, the communication coordinator finds that the flag has been set, which means process rank 4 is on the same host. Thus, communication coordinator will schedule the communication between rank 1 and rank 4 to go through IVShmem channel, as presented in the solid line. If the communication coordinator finds that the flag is not set (e.g. position 6), then it will coordinate the communication between rank 1 and rank 6 to go through SR-IOV channel as shown in dashed line. Same thing happens on the Guest 2 side also, rank 4 will be scheduled to communicate with rank 6 using SR-IOV channel.




(a) Impact of Eager Message Size

(b) Impact of Send Buffer Size

(c) Impact of Length Queue

(d) Impact of Number of Send Buffer

Fig. 5: Communication Optimization for IVShmem Channel

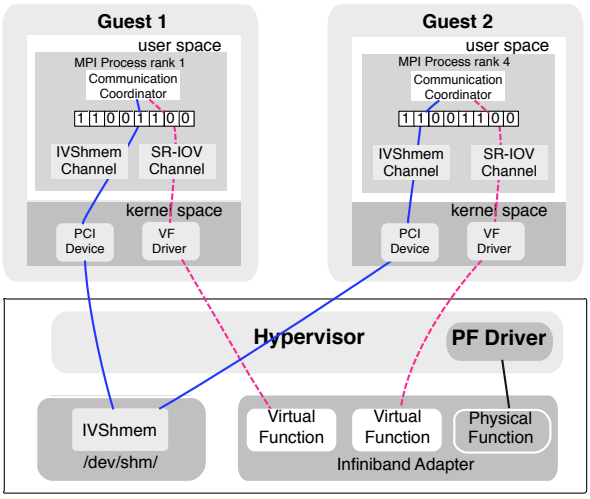


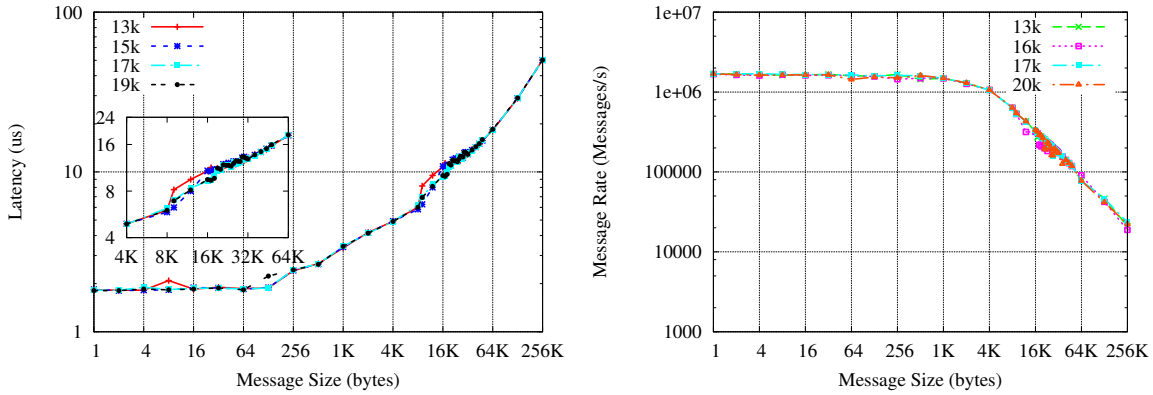

Fig. 4: Communication Coordinator

### D. Optimizing Communication for IVShmem Channel

When IVShmem channel is selected by Communication Coordinator, the default environment setting, which is optimized for native environment, may not be able to benefit MPI communication by the greatest extent, therefore, we need to optimize the IVShmem channel further in order to achieve high performance message passing for intra-host inter-VM communication. There are four related parameters need to be optimized, which are SMP_EAGER_SIZE, SMP_SEND_BUF_SIZE, SMPI_LENGTH_QUEUE and SMP_NUM_SEND_BUFFER. SMP_EAGER_SIZE defines the switch point between Eager protocol and Rendezvous protocol. SMPI_LENGTH_QUEUE is the size of the shared memory buffer which is used to store outstanding small and control messages. Messages larger than SMP_EAGER_SIZE are packetized and sent out in a pipelined manner. SMP_SEND_BUF_SIZE is the packet size. SMP_NUM_SEND_BUFFER is the number of send buffers. Figure 5 shows the optimization result. Here we only show bandwidth optimization result because there is no clear difference in terms of latency and buffer space (memory footprint) constraints. As we can see in Figure 5(a), the optimal bandwidth performance, which is more than 9.3Gb/s, is delivered when SMP_EAGER_SIZE is set to 32k. Even though 64k also delivers similar bandwidth performance, we still select 32k in order to reduce memory footprint. For large size message transfer, it can be observed in Figure 5(b) that based on optimized SMP_EAGER_SIZE value, bandwidth performance can achieve 9.6Gb/s when SMP_SEND_BUF_SIZE is set to 16k. Similarly, SMPI_LENGTH_QUEUE and SMP_NUM_SEND_BUFFER are set to 128k and 16, respectively.

### E. Optimizing Communication for SR-IOV Channel

For the optimization of SR-IOV channel, we need to consider parameter MV2_IBA_EAGER_THRESHOLD, which




(a) Impact of Eager Threshold on Latency

(b) Impact of Eager Threshold on Message Rate

Fig. 6: Communication Optimization for SR-IOV Channel

specifies the switch point between eager and rendevous protocol. If the threshold is too small, then it could incur additional overhead of RTS/CTS exchange during rendezvous transfer between sender and receiver for many message sizes. If it is too large, then it will require larger amount of memory space for the library. Therefore, we need to optimize this channel to have optimal threshold for inter-host inter-VM communication. We measure the performance by setting MV2_IBA_EAGER_THRESHOLD to different values from 13k to 20k. In Figure 6, just some representative values are shown to make the contrast more clear. We can see in Figure 6 that it delivers the optimal performance in terms of latency and message rate, when MV2_IBA_EAGER_THRESHOLD is set to 17k.

## IV. Performance Evaluation

In this section, we describe our experimental testbed and discuss the evaluation results on different dimensions based on the optimization results mentioned in Section III-D and Section III-E. We evaluate the performance of our proposed design on SR-IOV enabled InfiniBand clusters from four dimensions, which are point-to-point communication, collective operations, different InfiniBand transport protocols (RC and UD), and representative HPC applications.

### A. Experiment Setup

Our testbed is an InfiniBand cluster consisting of four physical nodes, where each node has dual 8-core 2.6 GHz Intel Xeon E5-2670 (Sandy Bridge) processors with 20MB L3 shared cache, 32 GB main memory and equipped with Mellanox ConnectX-3 FDR (56 Gbps) HCAs with PCI Express Gen3 interfaces. We use RedHat Enterprise Linux Server release 6.4 (Santiago) with kernel 2.6.32-279.19.1.el6.x86_64 as the host and VM OS. In addition, we use the Mellanox OpenFabrics Enterprise Distribution MLNX_OFED_LINUX-2.1-1.0.0 to provide the InfiniBand interface with SR-IOV support and use KVM as the Virtual Machine Monitor (VMM). Each VM is pinned to a single core and has 1.5 GB main memory. All applications and libraries used in this study are compiled with gcc 4.4.6 compiler.

All experiments are conducted by comparing our proposed design with MVAPICH2-2.0. We choose OSU Micro-Benchmarks (OMB) 4.3 to do the evaluations. Over all four physical nodes, we allocate 8 VMs per node to conduct experiments for collectives, different transport protocols and applications. For point to point experiments, we select two VMs from them.

### B. Point-to-Point Communication Performance

In this section, we evaluate MPI point-to-point communication performance for inter-VM in terms of latency and bandwidth.

Figure 7(a) and Figure 7(b) show the point-to-point performance for intra-host inter-VM communication. From these two figures, we can observe that compared to SR-IOV, our proposed design can significantly improve the point-to-point performance by up to 84% and 158% for latency and bandwidth, respectively. If we compare the performance of our design with that of native MPI, we can see that our design only has 3%-8% overheads, which are much smaller than the overheads of SR-IOV. For example, at 1KB message size, MPI point-to-point latency of SR-IOV is around 2.36$\mu$s, while the latencies of our design and native mode are 0.52$\mu$s and 0.5$\mu$s, respectively. In this case, our design just shows about 4% overhead. Through this comparison, we can clearly see the performance benefits by incorporating locality-aware communication into MPI library over virtualized environments.

For inter-host inter-VM point-to-point communication, as shown in Figure 7(c) and Figure 7(d), our proposed design has similar performance with SR-IOV in terms of latency and bandwidth. This is because the communication coordinator in the proposed design will select the SR-IOV channel for inter-host inter-VM data movement. These results also show that the newly introduced components in our proposed design do not cause extra overhead. If we compare the performance with native MPI, we can see that the overheads of both our proposed design and SR-IOV are very small. For example, the bandwidth of native MPI is about 6.3Gb/s at the message size of 256KB, while both SR-IOV and our design can achieve 6.2 GB/s bandwidth.

From the above discussion, we can see that our proposed design can achieve near-native performance for both intra-host inter-VM and inter-host inter-VM communications. This is because our design can fully exploit the benefits of locality-aware communication for intra-host inter-VM data movement,

while maintaining similar performance behavior as SR-IOV channel for inter-host inter-VM communication.

### C. Collective Communication Performance

We select four widely used collective communication operations in our evaluations: Broadcast, Allgather, Allreduce and Alltoall. As shown in Figures 8(a)-Figure 8(d), we can clearly observe that, compared with SR-IOV, the proposed design effectively cuts down the latency for each collective operation across 32 VMs.

We show the Broadcast performance in Figure 8(a). The latency of SR-IOV scheme is 6.73 $\mu$s at 4 bytes message size, while it is 4.44 $\mu$s for proposed design, with 34% improvement. The performance benefit comes from locality-aware based communication in our proposed design instead of IB loopback in SR-IOV. The Allgather performance is shown in Figure 8(b), the latencies of SR-IOV scheme and proposed design at 4 bytes message size are 15.77 $\mu$s and 11.2 $\mu$s, respectively. The proposed design improves the performance by 29%. Figure 8(c) shows us the latency of Allreduce operation. We can see that at 4 bytes message size the latency values are 17.29 $\mu$s and 6.97 $\mu$s for SR-IOV scheme and proposed design, respectively. The performance improvement at 4 bytes message size achieves 60%. With respect to Alltoall operation, as shown in Figure 8(d), SR-IOV and proposed design deliver 32.38 $\mu$s and 27.20 $\mu$s latencies, respectively. The proposed design helps reduce the latency of alltoall at 4 bytes message size by 16%. For different message sizes, the proposed design can improve latency of the above four collective operations (Brocast, Allgather, Allreduce, Alltoall) by up to 68%, 76%, 61%, 29%, respectively.

Based on the experimental evaluation results, our proposed design can gain remarkable improvement for MPI collective operations compared to SR-IOV.

### D. Different InfiniBand Transport Protocol (RC & UD)

In Section II-C, we give the introduction of different InfiniBand transport protocols. In this section, we evaluate the performance of SR-IOV and proposed design on different InfiniBand transport protocols. The point-to-point and collective results are shown in Figure 9 and Figure 10, respectively. Figure 9 shows that the RC protocol performs better than UD for SR-IOV scheme in terms of latency for intra-host inter-VM point-to-point communication. The latency difference can be up to 60%. This is because MVAPICH2 library enables Fast Path feature by default for RC protocol, which supports RDMA based communication, while the UD scheme does not support this feature. Moreover, the reliability support for UD in MPI library costs some additional overhead. Therefore, we see performance differences between RC and UD protocols on SR-IOV scheme. Since our proposed design uses IVShmem based communication instead of SR-IOV within the same node, it does not have this kind of performance differences as SR-IOV scheme.

Based on the above discussion of point-to-point communication, we can reasonably explain the performance behavior of following collective operations. For Broadcast operation in Figure 10(a), the proposed design outperforms SR-IOV scheme for both RC and UD protocols. In addition, there exists clear performance difference between RC and UD protocols for SR-IOV scheme. Compared with RC, UD protocol increases the broadcast latency by up to 206%. Whereas the proposed design delivers similar performance for these two transport protocols. This is also because the intra-host inter-VM communication goes through IVShmem channel instead of SR-IOV channel in the proposed design and it does not get much affected by different transport protocols. However, as the proportion of inter-host inter-VM communication to total amount of communication increases, the influence of different transport protocols will be more obvious accordingly. Similar to Broadcast, other three collective operations, Alltoall, Allgather and Allreduce for SR-IOV scheme on the UD protocol, incur up to 173%, 76%, 118% latency increasing, respectively, compared with the RC protocol. On the contrast, our proposed design still delivers close performance when switching from RC to UD protocol.

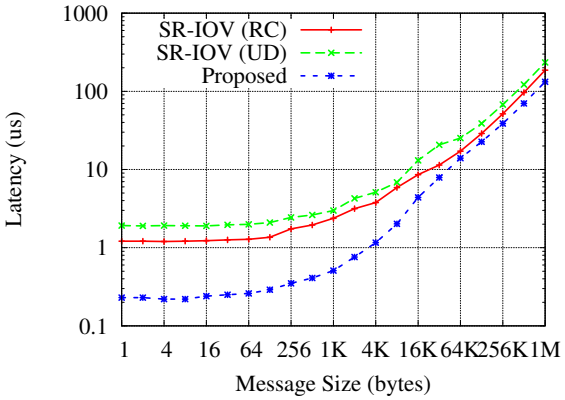



Fig. 9: Intra-host Inter-VM Point-to-Point Performance on RC and UD Protocols

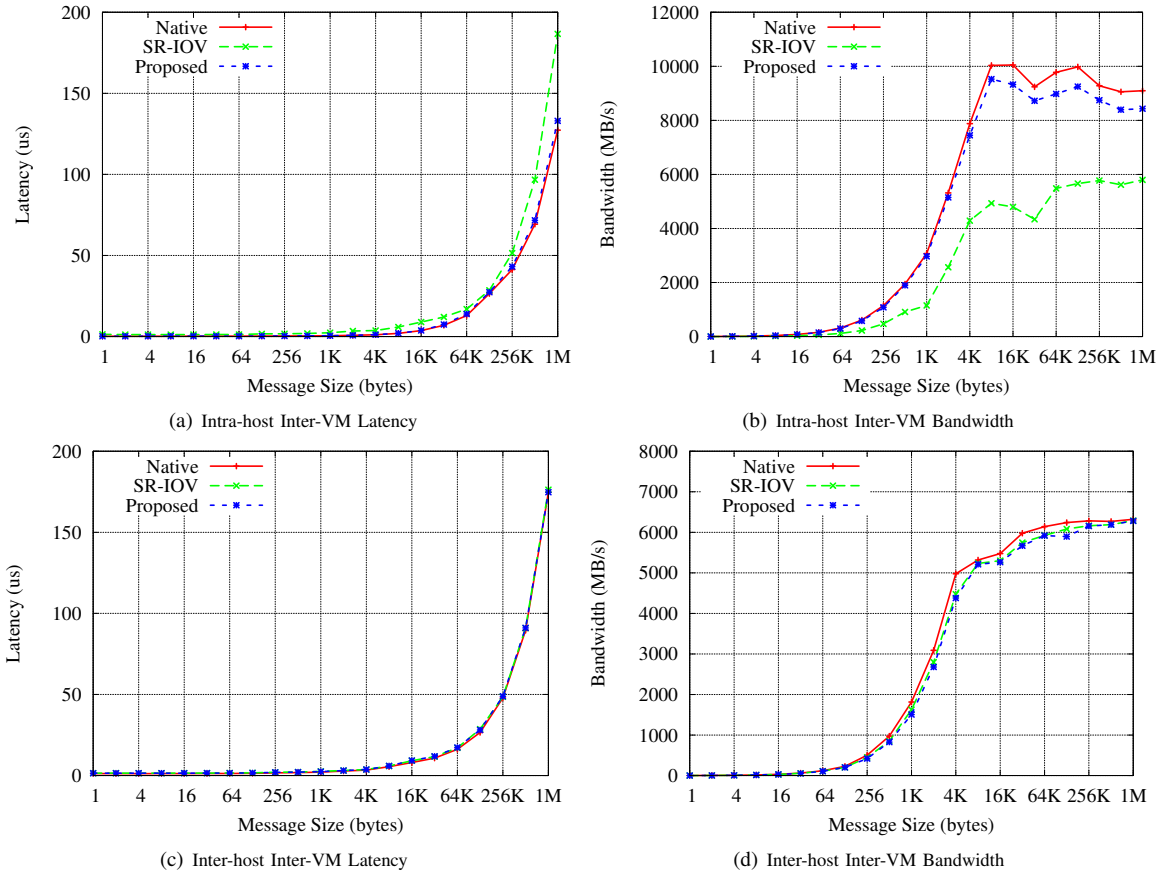
### E. Application Performance

In this section, we evaluate our proposed design and SR-IOV scheme with two end applications: P3DFFT and NAS Parallel Benchmarks (NPB) .

Figure 11 shows the performance comparison of SR-IOV scheme and proposed design on Class B NAS benchmarks on 32 VMs across 4 nodes. Figure 11(a) shows that our proposed design improves the performance for NAS by up to 43% over SR-IOV based scheme. For IS benchmark, SR-IOV scheme needs 2.84s, whereas our proposed design takes only 1.61s. In Figure 11(b), we show the performance of SR-IOV scheme and our proposed design with P3DFFT. We run all 5 tests with the same input size $512 \times 512 \times 512$. From the results, we can see that our proposed scheme outperforms the SR-IOV scheme in all the cases. The improvements for `INVERSE`, `RAND`, `SINE` and `SPEC` are 29%, 33%, 29% and 20%, respectively. The performance benefits for P3DFFT comes from shared memory collective operations that are not available in SR-IOV scheme.

## V. Related Work

In general, I/O virtualization schemes can be classified into software based and hardware based schemes. Earlier studies such as [11], [12] have shown network performance evaluations of software-based approaches in Xen. Studies [13], [14], [15] have proved that SR-IOV demonstrates significantly




(a) Intra-host Inter-VM Latency

(b) Intra-host Inter-VM Bandwidth

(c) Inter-host Inter-VM Latency

(d) Inter-host Inter-VM Bandwidth

Fig. 7: Point-to-point Performance

better performance compared to that of software-based solutions for 10GigE networks. Liu et. al [13] provides a detailed performance evaluation on the environment of SR-IOV capable 10GigE Ethernet in KVM. Also, they studied several important factors that affect network performance in both virtualized and native environments. Studies [16], [17], [18], [19] with Xen demonstrate the ability to achieve near-native performance in VM-based environment for HPC applications. In addition, the work [7] first presented the framework of Nahanni and gave its introduction in details. Based on it, MPI-Nahanni user-level library was developed, which ported MPICH2 library from Nemesis channel that used memory-mapped shared memory to Nahanni in order to accelerate inter-VM communication on the same host.

In our earlier studies, we proposes designs to improve intra-node point-to-point communication operations using an Inter-VM Communication Library (IVC) and re-designed the MVAPICH2 library to leverage the features offered by the IVC [16]. However, this solution is based on the Xen platform and does not show the studies with SR-IOV enabled InfiniBand clusters. Our early evaluation of using SR-IOV with InfiniBand [3] shows that while SR-IOV enables low-latency communication, MPI libraries need to be redesigned carefully in order to provide advanced features to improve intra-node inter-VM communication. Within a single node, our recent evaluation [8] reveals the fact that the performance of intra-node inter-VM communications can be dramatically improved through IVShmem, compared to SR-IOV scheme on virtualized InfiniBand clusters. This exhibits significant performance potential to optimize MPI communication across nodes further.

Therefore, based on our previous evaluation, we present a solution for MPI inter-node communication in this paper. We propose a new design using high performance MPI library with the support of KVM and IVShmem [7], which dynamically detects the VM locality information and coordinates communications between IVShmem and SR-IOV channels to offer effective locality-aware communication on SR-IOV enabled InfiniBand clusters. The evaluation results show promising results of our proposed design with regard to point-to-point, collective benchmarks and end-applications.

## VI. Conclusion and Future Work

In this paper, we analyzed multiple VM locality detection approaches and proposed a high performance design of MPI library over SR-IOV enabled InfiniBand clusters, which can dynamically detect co-located VMs and coordinate communications between SR-IOV and IVShmem channels. The proposed design efficiently supports locality-aware communication across VMs. We further analyzed and optimized MPI library level core mechanisms and design parameters in both SR-IOV and IVShmem channels for virtualized environments. Based on our new design, we conducted comprehensive performance evaluations by using point-to-point, and collective benchmarks and representative HPC applications.

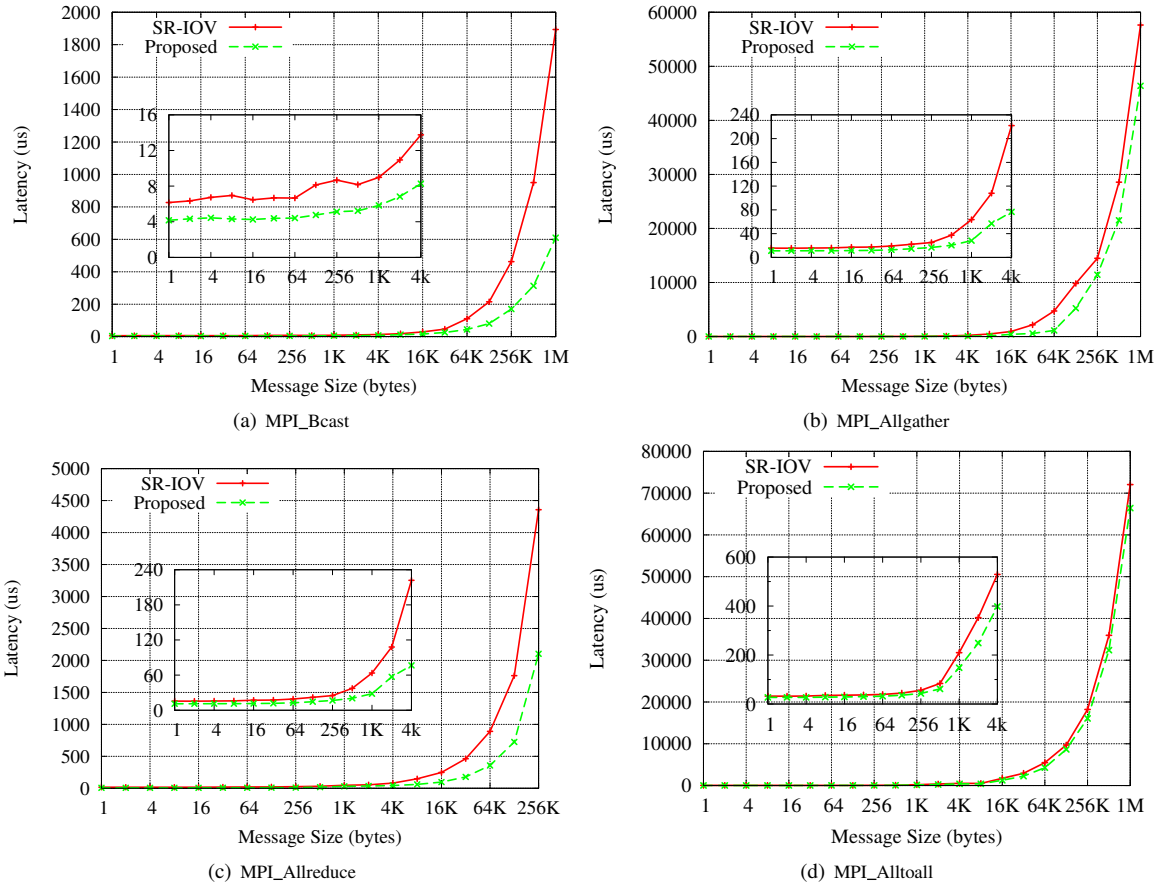



Fig. 8: Collective Communication Performance on 32 VMs (8 VMs per node)

Our performance evaluations show that, compared to SR-IOV, our proposed design can significantly improve the performance of intra-host inter-VM communication by up to 84% and 158% for latency and bandwidth, respectively, while the proposed design only introduces 3%-8% overhead compared with native mode. The evaluations also show that our proposed design effectively integrates SR-IOV channel for inter-host inter-VM communication. On the aspect of collective operations, the proposed design can achieve up to 68%, 76%, 61%, and 29% performance improvements for Broadcast, Allgather, Allreduce, and Alltoall, respectively, compared to SR-IOV. In addition, the evaluations for different InfiniBand transport protocols (RC and UD) indicate that SR-IOV incurs performance degradation when switching the protocol from RC to UD in MPI library. Whereas based on locality-aware communication, our proposed design delivers similar performance between these two protocols. Finally, compared to SR-IOV, our design outperforms NAS and P3DFFT by up to 43% and 33%, respectively.

In the future, we plan to explore the effect of different collective algorithms on virtualization environment and improve current high performance MPI library to support live migration on SR-IOV enabled InfiniBand clusters.

## REFERENCES


[1] M. Rosenblum and T. Garfinkel, "Virtual Machine Monitors: Current Technology and Future Trends," *Computer*, vol. 38, no. 5, pp. 39–47, 2005.

[2] "Amazon EC2," http://aws.amazon.com/ec2/.

[3] J. Jose, M. Li, X. Lu, K. Kandalla, M. Arnold, and D. Panda, "SR-IOV Support for Virtualization on InfiniBand Clusters: Early Experience," in *On 13th IEEE/ACM International Symposium Cluster, Cloud and Grid Computing (CCGrid)*, May 2013, pp. 385–392.

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

[5] OpenMPI: Open Source High Performance Computing, http://www.open-mpi.org/.

[6] Single Root I/O Virtualization, http://www.pcisig.com/specifications/iov/single_root.

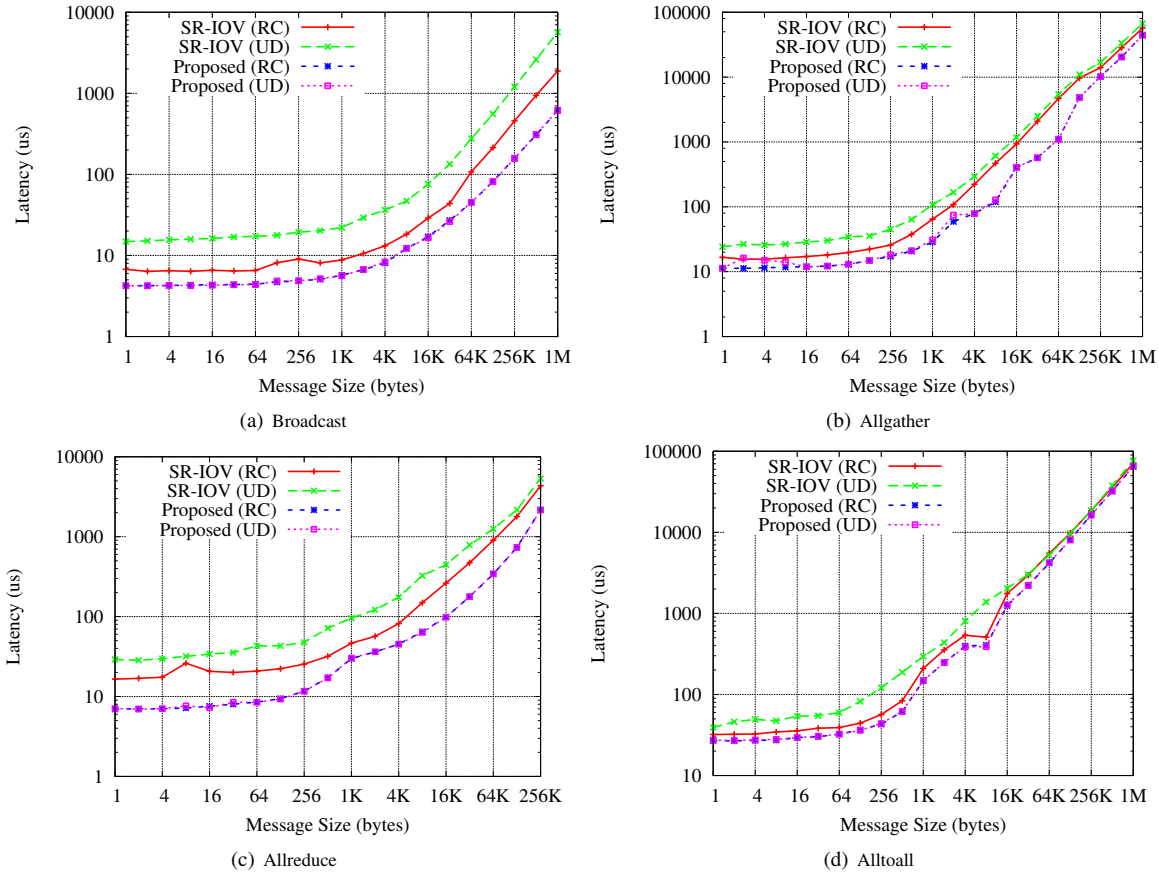
[7] A. C. Macdonell, "Shared-Memory Optimizations for Virtual Machines," PhD Thesis. University of Alberta, Edmonton, Alberta, Fall 2011.

[8] J. Zhang, X. Lu, J. Jose, R. Shi, D. K. Panda, "Can Inter-VM Shmem Benefit MPI Applications on SR-IOV based Virtualized InfiniBand Clusters?" in *Proceedings of 20th International Conference Euro-Par 2014 Parallel Processing (to be appear)*, Porto, Portugal, August 25-29 2014.

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

[10] Y. Ren, L. Liu, Q. Zhang, Q. Wu, J. Yu, J. Kong, J.Guan, H. Dai, "Residency-Aware Virtual Machine Communication Optimization: Design Choices and Techniques," in *Proceedings of IEEE 6th International Conference on Cloud Computing (Cloud 2013)*, Santa Clara Marriott, CA, USA, June 27-July 2 2013.

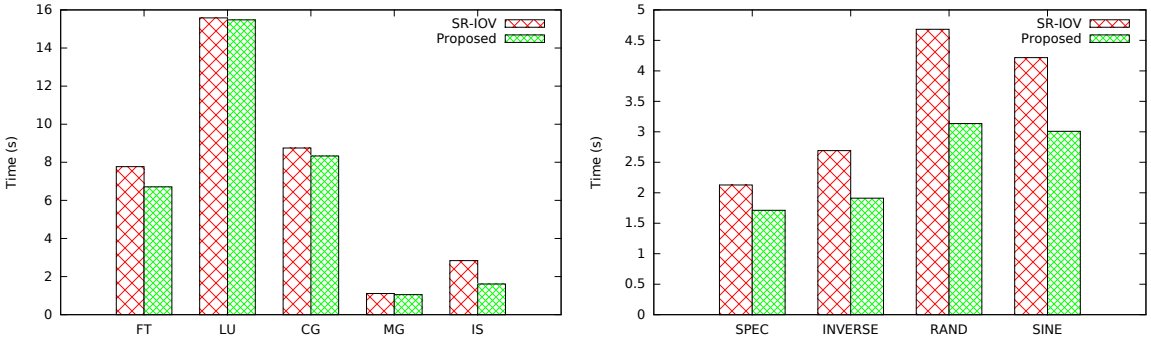
[11] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel, "Diagnosing Performance Overheads in the Xen Virtual Machine Environment," in *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, ser. VEE '05. New York, NY, USA: ACM, 2005, pp. 13–23.




(a) Broadcast (b) Allgather

(c) Allreduce (d) Alltoall

Fig. 10: 32 VMs (8 VMs per node) Collective Performance on RC and UD Protocols




(a) NAS Performance (b) P3DFFT Performance

Fig. 11: Application Performance


[12] P. Apparao, S. Makineni, and D. Newell, "Characterization of Network Processing Overheads in Xen," in *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, ser. VTDC '06. Washington, DC, USA: IEEE Computer Society, 2006.

[13] J. Liu, "Evaluating Standard-Based Self-Virtualizing Devices: A Performance Study on 10 GbE NICs with SR-IOV Support," in *Proceeding of 2010 IEEE International Symposium Parallel & Distributed Processing (IPDPS)*. IEEE, 2010, pp. 1–12.

[14] Y. Dong, X. Yang, J. Li, G. Liao, K. Tian, and H. Guan, "High Performance Network Virtualization with SR-IOV," *Journal of Parallel and Distributed Computing*, 2012.

[15] Z. Huang, R. Ma, J. Li, Z. Chang, and H. Guan, "Adaptive and Scalable Optimizations for High Performance SR-IOV," in *Proceeding of 2012 IEEE International Conference Cluster Computing (CLUSTER)*. IEEE, 2012, pp. 459–467.

[16] W. Huang, M. J. Koop, Q. Gao, and D. K. Panda, "Virtual Machine aware Communication Libraries for High Performance Computing," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, ser. SC '07. New York, NY, USA: ACM, 2007, pp. 9:1–9:12.

[17] W. Huang, J. Liu, B. Abali, and D. K. Panda, "A Case for High Performance Computing with Virtual Machines," in *Proceedings of the 20th Annual International Conference on Supercomputing*, ser. ICS '06, New York, NY, USA, 2006.

[18] J. Liu, W. Huang, B. Abali, and D. K. Panda, "High Performance VMM-bypass I/O in Virtual Machines," in *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ser. ATC '06, Berkeley, CA, USA, 2006.

[19] W. Huang, J. Liu, M. Koop, B. Abali, and D. Panda, "Nomad: Migrating OS-bypass Networks in Virtual Machines," in *Proceedings of the 3rd International Conference on Virtual Execution Environments*, ser. VEE '07, New York, NY, USA, 2007.