# SR-IOV Support for Virtualization on InfiniBand Clusters: Early Experience

Jithin Jose, Mingzhe Li, Xiaoyi Lu, Krishna Chaitanya Kandalla, Mark Daniel Arnold, and Dhabaleswar K. (DK) Panda
*Department of Computer Science and Engineering, The Ohio State University*
*Email: {jose, limin, luxi, kandalla, arnoldm, panda}@cse.ohio-state.edu*

***Abstract*—High Performance Computing (HPC) systems are becoming increasingly complex and are also associated with very high operational costs. The cloud computing paradigm, coupled with modern Virtual Machine (VM) technology offers attractive techniques to easily manage large scale systems, while significantly bringing down the cost of computation, memory and storage. However, running HPC applications on cloud systems still remains a major challenge. One of the biggest hurdles in realizing this objective is the performance offered by virtualized computing environments, more specifically, virtualized I/O devices. Since HPC applications and communication middlewares rely heavily on advanced features offered by modern high performance interconnects such as InfiniBand, the performance of virtualized InfiniBand interfaces is crucial. Emerging hardware-based solutions, such as the Single Root I/O Virtualization (SR-IOV), offer an attractive alternative when compared to existing software-based solutions. The benefits of SR-IOV have been widely studied for GigE and 10GigE networks. However, with InfiniBand networks being increasingly adopted in the cloud computing domain, it is critical to fully understand the performance benefits of SR-IOV in InfiniBand network; especially for exploring the performance characteristics and trade-offs of HPC communication middlewares (such as Message Passing Interface (MPI), Partitioned Global Address Space (PGAS)) and applications. To the best of our knowledge, this is the first paper that offers an in-depth analysis on SR-IOV with InfiniBand. Our experimental evaluations show that for the performance of MPI and PGAS point-to-point communication benchmarks over SR-IOV with InfiniBand is comparable to that of the native InfiniBand hardware; for most message lengths. However, we observe that the performance of MPI collective operations over SR-IOV with InfiniBand is inferior to native (non-virtualized) mode. We also evaluate the trade-offs of various VM to CPU mapping policies on modern multi-core architectures and present our experiences.**

***Keywords*-SR-IOV; InfiniBand; Clusters; Virtualization; HPC**

## I. Introduction

Over the years, the cloud computing paradigm has become increasingly popular. In this paradigm, organizations offer computing, storage resources, and infrastructure as a service based on a subscription-based access. Customers can utilize such resources for their computing needs and are charged based on their usage. Modern Virtual Machine (VM) technology offers attractive techniques to easily manage various hardware/software components, along with guaranteeing security, performance isolation, consolidation and live migration [1]. Coupled with emerging virtualization technology, the cloud computing paradigm has reduced the cost of computation and storage by several orders of magnitude.

However, running HPC applications on cloud systems still remains a major challenge. One of the biggest hurdles in realizing this objective is the lower performance offered by virtualized computing environments. Recently introduced hardware and software techniques [2], [3], [4] have significantly narrowed the performance gap between VMs and native hardware for CPU and memory virtualization. However, improving the performance of virtualized I/O devices still remains a challenge, especially for high speed networking devices. Since many applications and communication middlewares (such as Message Passing Interface (MPI) [5] and Partitioned Global Address Space (PGAS) [6]) in the HPC domain rely extensively on features offered by modern interconnects, the performance of virtualized I/O devices are likely to be the key driver in the adoption of virtualized cloud computing systems for HPC applications.

HPC systems are increasingly being deployed with modern interconnects such as InfiniBand [7]. Currently, more than 44% of the fastest supercomputing systems rely on InfiniBand for their I/O and networking requirements [8]. InfiniBand offers the Remote Direct Memory Access (RDMA) feature to deliver low latency and high bandwidth to communication middlewares and scientific applications. Moreover, InfiniBand adapters can also be used to run applications that use TCP/IP over the InfiniBand network in the IP-over-InfiniBand (IPoIB) mode [9]. Owing to these reasons, InfiniBand is also being increasingly used to deploy Cloud computing systems.

State-of-the-art I/O virtualization solutions can be broadly classified as software-based and hardware-based approaches. In software-based approaches, several software components such as guest VMs, a virtual machine monitor (VMM) and possibly a special I/O VM, work together to provide virtualized I/O access points to VMs without special hardware support [10], [11]. In these approaches, physical devices on the host cannot be accessed directly by the guest VMs and each I/O operation is virtualized by the multiple software components. Such solutions suffer from significant performance degradation when compared to native high performance network interfaces because of overheads such as context/control switches and memory copies. However, hardware-based I/O virtualization approaches can potentially achieve higher performance by allowing direct hardware access from within a guest VM [12]. In these approaches, performance-critical I/O operations can be carried out in a guest VM by interacting with the hardware directly instead of involving the VMM or special VMs. Recently, the industry has released several standards which specify native I/O virtualization (IOV) capabilities in PCI Express (PCIe) adapters. These include Single-Root IOV (SR-IOV) [13] and Multi-Root IOV (MR-IOV) [14].

In SR-IOV, a PCIe device presents itself as multiple virtual devices and each virtual device can be dedicated to a single VM. MR-IOV enables sharing of PCIe device resources between different physical servers. Recent studies have demonstrated that SR-IOV is significantly better than software-based solutions for GigE and 10GigE networks [15], [16], [17], [18]. However, considering that InfiniBand networks are also being adopted by Cloud computing systems leads us to a broad question: *Is SR-IOV support for InfiniBand networks, ready for "Prime-Time" HPC workloads?*

## II. Motivation

The performance characteristics of the InfiniBand native hardware have been thoroughly evaluated by the HPC community. However, the trade-offs of using InfiniBand in a virtualized environment with SR-IOV support have not been demonstrated. InfiniBand networks offer two main communication modes: memory (RDMA) and channel semantics. Additionally, InfiniBand networks offer two main progress modes: *polling* and *blocking*. While high performance communication libraries usually use polling mode for better communication latencies [19], the blocking mode might be preferable when multiple VMs share the same physical InfiniBand adapter. The network interrupt overheads experienced by middlewares running across different VMs can significantly impact the overall performance of parallel applications. With the emergence of multi-core architectures, multiple VMs can be scheduled on one compute node with different subscription policies, either one VM per core, one VM per CPU Socket (NUMA-node), or one VM per compute node. For HPC applications the performance trade-offs of such VM subscription policies are not obvious. Moreover, in order to facilitate the adoption of the SR-IOV technology in main-stream HPC, it is also imperative to analyze the performance characteristics of running parallel applications based on MPI and PGAS in a virtualized environment. In this paper, we offer an in-depth study of these important challenges to understand the performance characteristics and trade-offs of using SR-IOV with state-of-the-art InfiniBand networks for HPC communication middlewares and applications. To summarize, we address the following critical problems:

1) InfiniBand networks offer various transport mechanisms and communication modes. What are the performance characteristics and trade-offs of using the SR-IOV mechanism, when compared to native (non-virtualized) InfiniBand hardware?
2) MPI and PGAS runtimes offer various point-to-point and collective communication primitives that are widely used in scientific applications. What is the performance of such operations when used with SR-IOV in InfiniBand cluster?
3) On multi-core architectures, VMs can be run with various subscription policies. Can we quantify the impact of such policies on the performance of HPC communication middlewares?
4) Finally, can we offer insights into the performance characteristics of scientific application benchmarks in the environment of SR-IOV capable InfiniBand?

Our experimental evaluations show that for the performance of MPI and PGAS point-to-point communication benchmarks over SR-IOV with InfiniBand is comparable to that of the native InfiniBand hardware, for medium and large message lengths. However, we observe that the performance of MPI collective operations over SR-IOV with InfiniBand is inferior when compared to the native designs. We also study the trade-offs of various VM to CPU mapping policies on modern multi-core architectures.

## III. Background

In this section, we provide an overview of SR-IOV, InfiniBand and HPC Cloud Computing middlewares.

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

Single Root IOV (SR-IOV) specifies native I/O Virtualization (IOV) capabilities in the PCI Express (PCIe) adapters. SR-IOV is applicable when a PCIe interface works in a single server environment and allows a single physical device, or a Physical Function (PF), to present itself as multiple virtual devices, or Virtual Functions (VFs) (Figure 1). 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 realize I/O virtualization. Moreover, VFs are designed to be based on the existing non-virtualized PFs; hence, the drivers that currently drive the current adapters can also be used to drive the VFs in a portable fashion.

### B. InfiniBand

InfiniBand [7] is an industry standard switched fabric that is designed for interconnecting nodes in HEC clusters. It is a high-speed, general purpose I/O interconnect that is widely used by scientific computing centers world-wide. The recently released TOP-500 [8] rankings (November 2012) reveal that more than 44% of the computing systems use InfiniBand as their primary interconnect. InfiniBand is also gaining ground in the commercial domain. InfiniBand offers the RDMA feature, which can be used by communication middlewares to minimize communication overheads through zero-copy communication semantics. In addition, InfiniBand also offers basic Send/Recv channel semantics. Furthermore, InfiniBand adapters can also be used to run applications that use TCP/IP over the InfiniBand network in the IPoIB mode [9]. In addition, InfiniBand networks

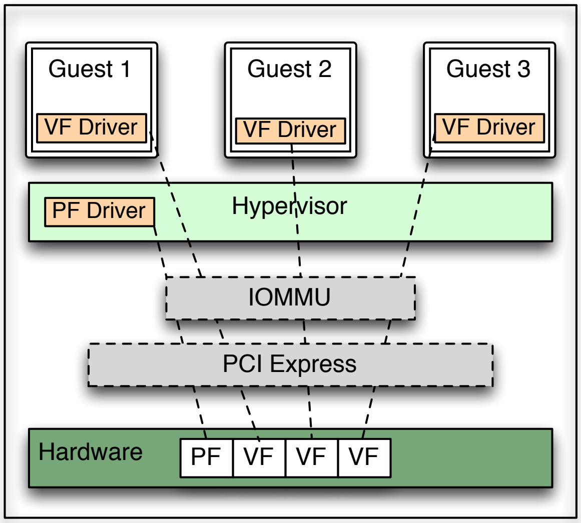


Figure 1. Overview of SR-IOV

also offer polling and event based progress modes. HPC communication libraries typically use the polling mode to minimize communication overheads, but such an approach requires the CPUs to constantly poll various completion queues. On the other hand, the blocking progress mode suspends the execution of a process that is waiting for specific network events and generates an interrupt when the specific condition is satisfied. This mode may lead to poorer communication performance, but it can free up CPU cycles.

### C. HPC Cloud Computing Middlewares

*1) Message Passing Interface (MPI):* MPI has been the de-facto programming model for developing HPC applications. MPI offers various point-to-point, one-sided and collective communication primitives that are widely used across parallel applications. MPI has evolved over the years to include many features to improve performance of parallel applications. The current MPI-3 specification offers various attractive new features including non-blocking collectives and improved support for one-sided operations. MVAPICH2 is a high-performance implementation of the MPI standard over InfiniBand, RoCE, and iWARP [19]. It is widely used across more than 2,000 organizations world-wide and is powering various supercomputer systems, including the 7th ranked TACC Stampede system. MVAPICH2 takes advantage of various hardware features offered by the InfiniBand network interface to deliver best performance, at large scales [20], [21], [22], [23].

*2) Partitioned Global Address Space (PGAS):* While MPI has been very successful with regular, iterative applications, emerging PGAS models offer an attractive alternative to develop irregular applications. PGAS models offer *both* performance and programmer productivity. PGAS models can either be language-based, such as Unified Parallel C (UPC), or library-based, such as OpenSHMEM. UPC runtimes have also been optimized to take advantage of InfiniBand networks [7] to achieve high performance and scalability. However, since PGAS models are still emerging and lack the performance and scalability offered by MPI runtimes, it is unlikely that entire parallel applications will be rewritten with PGAS models. Instead, it is widely believed that next generation applications will be “Hybrid,” relying on both MPI and PGAS models, and take advantage of unified communication runtimes [24]. The MVAPICH2-X software stack [19] provides flexibility to write and use such Hybrid programming models.

## IV. Schemes to Analyze SR-IOV Capabilities

In this section, we discuss the various aspects of using InfiniBand in a virtualized environment. We also explore different dimensions for evaluating the performance characteristics and trade-offs of using SR-IOV with InfiniBand for HPC communication middlewares and applications. Experiment results for each of these dimensions are presented in Section V.

**InfiniBand Communication Modes:** In Section III-B, we discussed the various features offered by current generation InfiniBand network adapters. The performance and scalability features of these modes have been widely evaluated on native hardware. In Section V-B, we analyze the trade-offs of using virtualization technology with SR-IOV support. Specifically, we evaluate the performance of RDMA, Send/Recv, and IPoIB modes on SR-IOV and compare them to the performance obtained on native (non-virtualized) hardware. These are the primitives that are used for implementing different communication middlewares such as MPI and UPC.

**Polling vs. Event modes in InfiniBand:** InfiniBand offers two modes for getting completion events: polling and event based. In polling mode, user application continuously polls the completion queue (CQ). This mode is usually good for applications with many cores, and each core can continuously poll to ensure communication progress. The second mode is based on events, where the user application can register for completion events. High performance libraries such as MVAPICH2 provide both of these modes and enables application developers to choose progress mode based on communication characteristics. In a virtualization environment, CPU cores are likely to be oversubscribed, ie, a host node can have more Virtual Machines (VM) than CPU cores. In such environment, the event based mode is most optimal. Therefore, the performance of blocking versus polling modes and their impact on virtual machine deployment needs to be evaluated.

**HPC Communication Middlewares:** As discussed in Section III-C, MPI and PGAS programming models are commonly used to develop scientific parallel applications. Communication libraries that implement MPI, PGAS specifications are typically highly optimized to take advantage of the various features offered by the low-level hardware and network interface. In Section V-C, we evaluate the performance of various communication primitives in both MPI and UPC communication runtimes with SR-IOV and compare that against the performance achieved on native-hardware.

**Virtual Machine Configurations and Scalability on Multi-Core Architectures:** On multi-core architectures, more than one parallel job can be efficiently scheduled concurrently to achieve improved utilization of processor and network resources [25]. In such scenarios, using virtualization technology can lead to easier system management to offer performance isolation; but, the performance characteristics can vary significantly based on VM configurations. We evaluate the performance characteristics and trade-offs of scheduling more than one VM per node. Specifically, we evaluate various VM to CPU mapping patterns such as one VM per core, one VM per CPU Socket (NUMA-node), and one VM per compute node. Across these configurations, we evaluate the performance of various communication operations defined in the MPI standard. We present a detailed evaluation of these parameters in Section V-E. We also study VM scalability characteristics - the impact of running multiple VMs in a physical node - with SR-IOV and present our performance evaluations in Section V-F.

## V. Performance Evaluation

### A. *Experiment Setup*

Our experimental testbed consists of four compute nodes featuring the Intel Sandy Bridge-EP platform. Each node has dual Intel Xeon E5-2670 2.6GHz eight-core processors with a 20 MB L3 shared cache. Each compute node has 32 GB of main memory and, the platform is equipped with one PCIe 3.0 slot. We use RedHat Enterprise Edition 6.1 (RHEL6) with kernel 2.6.32-131.0.15.el6.x86 64 as the operating system on these nodes. We have compiled various applications used in this study with gcc 4.4.6 compiler. In addition, we use the Mellanox OpenFabrics Enterprise Edition (MLNX_OFED) SRIOV-ALPHA-3.3.0-2.0.0008 [26] to provide the InfiniBand interface with SR-IOV support. All the nodes are equipped with Mellanox ConnectX-3 FDR cards (56 Gbps). The compute nodes are connected to a Mellanox FDR switch SX6036 for all experiments. We used KVM [11] as the Virtual Machine Monitor (VMM).

We have used IB-Verbs benchmarks and NetPerf [27] for IB-Verbs level and socket level experiments. All MPI experiments were run using MVAPICH2 1.9a2 and the UPC experiments were run using MVAPICH2-X 1.9a2 [19]. We report results that were averaged across multiple runs to ensure fair comparisons.

### B. *InfiniBand Network Level Evaluations*

In this section, we present the performance evaluation results of SR-IOV compared to native mode. In native mode, we ran the experiments using two different physical nodes. Similarly, in virtual machine mode, the two VMs were placed in two neighboring physical nodes. We evaluated network level performance using IB-Verbs benchmarks, and IPoIB performance using the Netperf benchmark. The results are illustrated in Figure 3.

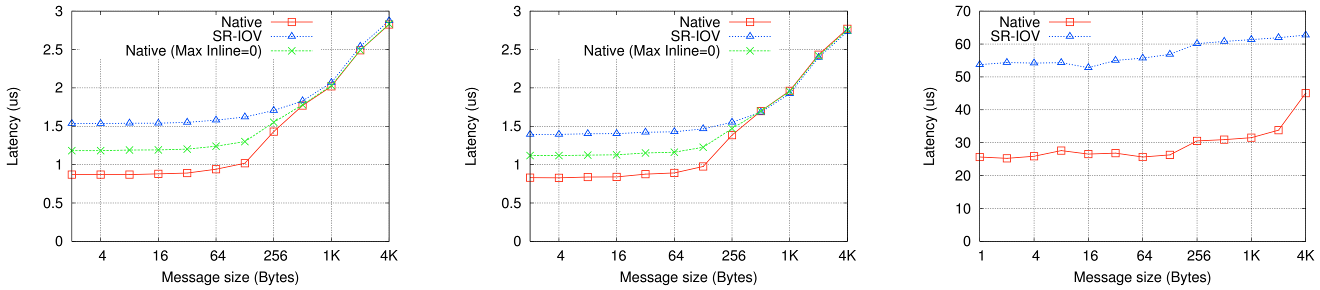
As indicated in Section III-B, InfiniBand has send-recv and memory semantics (RDMA). We present the send-recv performance of IB-Verbs in Figure 2(a). Here, we can see a big difference for small messages. The latency results for 2 byte message size for native and SR-IOV modes are 0.87 and 1.53 $\mu$s, respectively. However, for larger message sizes, the difference becomes nominal. Latency results for the 4 KB message size are 2.82 and 2.87 $\mu$s. We investigated this further and compared the device capabilities of physical and virtual NICs. We found out that the `max_inline_size` for virtual NIC is 0 bytes, while it is 400 bytes for the physical NIC. This explains the difference in performance for message sizes less than 400 bytes. Further, we also evaluated the performance of the native mode with `max_inline_size` set to 0. These results are indicated as 'Native (Max Inline=0)'. We can see that the difference for smaller message size is very minimal compared to the SR-IOV mode.

Figure 2(b) represents the RDMA write performance results. The results are similar to that of send-recv latency. For a 2 byte message size, the latency results observed for native and SR-IOV modes were 0.83 and 1.39 $\mu$s. The impact of `max_inline_size` is also evident in this case. Without this optimization, native mode performance is similar to that of SR-IOV mode.

IP over InfiniBand (IPoIB) results are presented in Figure 2(c). As discussed in Section III-B, socket applications can directly take advantage of InfiniBand using IPoIB. We used the Netperf [27] benchmark to evaluate IPoIB performance. The native mode IPoIB latency for a 1 byte message was 25.653 $\mu$s. With virtualization using SR-IOV, the latency observed was 53.74 $\mu$s. This is more than 2× difference in performance. We believe that the TCP stack overheads are significant in virtualized mode.

### C. *MPI and UPC Level Evaluations*

We also evaluated the MPI and UPC level performance with SR-IOV using OSU Microbenchmarks [28]. MPI latency and bandwidth results are depicted in Figures 3(a) and 3(b). The MPI latency for a 1 byte message was observed as 1.02 and 1.39 $\mu$s for native and SR-IOV modes, respectively. We used MVAPICH2-1.9a2 for our evaluations. MVAPICH2 employs 'RDMA-Fast Path' [29] optimization for small message sizes; which, uses RDMA write for small message sizes, so it matches with the IB-verbs level RDMA write latency. Also, it can be observed that the native and SR-IOV lines in Figures 3(a) converge earlier than in the pure IB-verbs case (Figure 2(b)). This is because the inline size optimization for MVAPICH2 is set as 128 bytes.

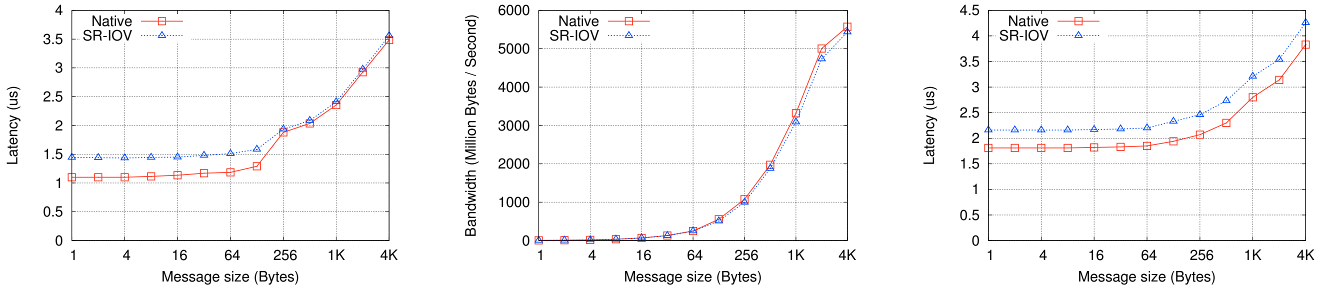


(a) IB-Verbs Send/Recv Latency

(b) IB-Verbs RDMA Write Latency

(c) IPoIB Latency

Figure 2. InfiniBand Network Level Evaluations



(a) MPI Latency

(b) MPI Bandwidth

(c) UPC Get Latency

Figure 3. MPI and UPC Level Evaluations

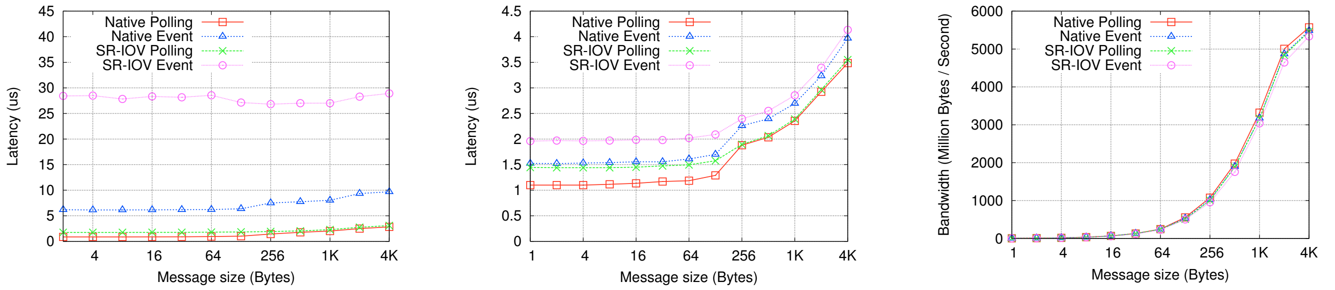
UPC level experiments are presented in Figure 3(c). We evaluated `upc_memget` operation in this experiment. These operations directly rely on RDMA Read operations. The 1 byte latency for `upc_memget` operation was observed as 1.81 and 2.16 $\mu$s, for native and SR-IOV mode, respectively. This indicates that there is a significant performance gap for RDMA Read operation between native and SR-IOV modes.

### *D. Impact of Polling vs. Event Modes*

As discussed in Section III-B, InfiniBand completion events can be handled in two modes: polling and blocking based. In polling mode, a user thread continuously polls the InfiniBand Completion Queue (CQ) for events. In the blocking mode, the user thread registers for receiving completion events and gets suspended when it invokes the `ibv_get_cq_event` function. On receipt of an event, the network interface generates an interrupt to signal the thread and it can get scheduled depending on the OS scheduling policies. Hence, the blocking mode may incur higher overheads due to interrupt and scheduling latencies. In these set of experiments, we evaluate the performance impact of polling versus blocking based modes for SR-IOV.

Figure 4(a) presents the latency results at InfiniBand verbs level under polling and blocking based schemes; we present both native and SR-IOV results here. For the native mode, the 1 byte latencies for polling and blocking based schemes are 0.83 and 6.19 $\mu$s, respectively. But in the SR-IOV mode, the latency results are significantly higher with the blocking scheme. The latency results obtained in SR-IOV mode for 1 byte message size with polling and blocking schemes are 1.53 and 28.43 $\mu$s. We believe that this is because of a lack of optimization related to serving the interrupts from the network interface within the SR-IOV firmware.

Furthermore, we evaluated MPI latency and bandwidth under these two schemes. The results are depicted in Figure 4(b) and 4(c). We used 'MV2_USE_BLOCKING=1' option in MVAPICH2 for enabling the blocking mode. In MVAPICH2 when the blocking mode is enabled, each process polls the completion queues for a fixed number of iterations and then invokes the `ibv_get_cq_event` function. If the required completion events are received within these number of iterations, then we skip the ibv_get_cq_event() function. Due to this design, we observe that the host level latency using default and 'MV2_USE_BLOCKING=1' option are 1.02 and 1.46 $\mu$s, respectively. In SR-IOV mode, these results are 1.39 and 1.89 $\mu$s. MPI bandwidth evaluation reveals that polling modes with both native and SR-IOV achieve near to peak bandwidth: 6,283.36 and 6,283.36 MB/s. For the blocking mode, the results for native and SR-IOV modes are 6,108.05 and 6,060.96 MB/s. However, as we increase the number of processes it is more likely



(a) IB-Verbs Latency

(b) MPI Latency

(c) MPI Bandwidth

Figure 4. Impact of Polling vs. Event Modes

that the MPI processes invoke the `ibv_get_cq_event` function and this can invariably affect the communication performance.

### *E. Impact of Virtual Machine Configurations*

VM deployment configuration has a significant impact on performance. Various modes for deployment are possible, such as VM per host CPU core, VM per host CPU socket, and VM per host node. In this section, we present the evaluation results of these configurations for SR-IOV. We evaluate the performance of MPI collectives using these configurations. We chose MPI_AlltoAll, MPI_AllReduce, and MPI_Bcast as representatives for common communication patterns across HPC applications.

MVAPICH2 uses hierarchical shared-memory based designs to optimize the intra-node phases of the MPI_Bcast and the MPI_Allreduce operations. The MPI_Alltoall operation is implemented directly over basic point-to-point operations. We compare the performance of these MPI collectives using OSU microbenchmarks [28] for both SR-IOV and native modes. In SR-IOV mode, we consider the following configurations - VM per host CPU core, VM per host CPU socket and VM per host node. For native mode, we present results with and without shared memory optimizations. Performance results are presented in Figures 5(a), (b) and (c).

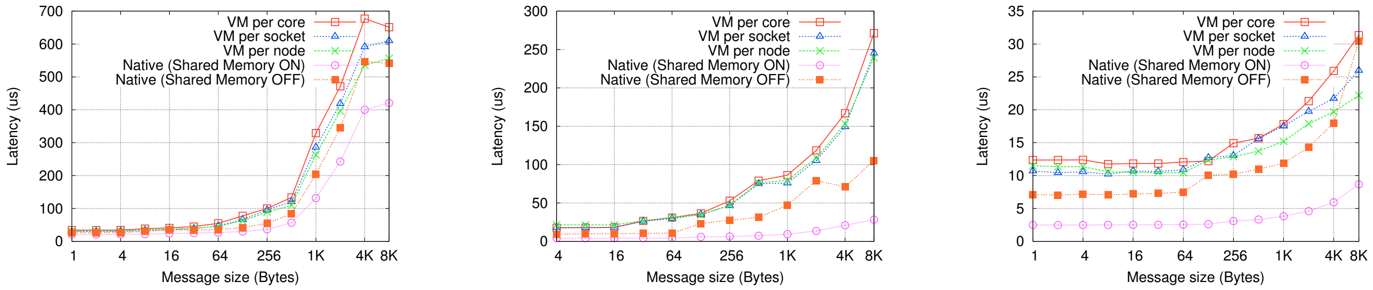
In these experiments, we kept the number of processes same in all the three modes as 32. We used four physical nodes for these experiments, each with 2 sockets. For example, in 1 VM per node mode, each VM hosted 8 processes. Similarly, in 1 VM per socket mode, each node was configured with 2 VMs, and each VM hosted 4 processes. For 1 VM per core, each node was configured with 8 VMs, and each VM hosted 1 process. We configured the VMs with same number of cores as the number of processes that it has to run. Thus, for 1 VM per node, 1 VM per socket and 1 VM per core cases, each VM has 8, 4 and 1 cores, respectively. We also disabled the intra-node shared memory communication in the MPI library for the virtualized modes, so that the comparisons were fair.

The performance evaluation results are presented in Figure 5. In all the three collectives, we observe that 1 VM per node case provides lowest latency. For MPI-Alltoall with message size 8 KB, the latencies observed were 651.16, 609.99 and 556.335 $\mu$s for VM per core, VM per socket and VM per node, respectively. MPI-Bcast results with 8 KB message size for these modes are 31.34, 25.98 and 22.20 $\mu$s. A similar trend was observed with MPI-AllReduce (22.20 $\mu$s for 8 KB message size). Notably these results are without intra-node shared memory communication. Thus, these results can further be improved if the 1 VM per node case uses shared memory for intra-node communication. We also compare these results with that of native mode (with and without intra-node shared memory support). In native mode (shared memory enabled), the latency results for MPI-Alltoall, MPI-Bcast and MPI-AllReduce for 8 KB message size are 405.96, 8.67 and 28.20 $\mu$s. These trends indicate that there is a huge performance difference between native and virtualized modes for collective operations, even with advanced virtualization features such as SR-IOV.

### *F. Impact of Virtual Machine Scalability*

In this section, we present the impact of virtual machine scalability on performance. We present the weak scalability results by using the MPI Graph500 [30] benchmark. The Graph500 benchmark is a Breadth First Search (BFS) benchmark, which reports the BFS traversal time for a given input graph size. In our experiments, we use a graph with 4 million vertices and 64 million edges. We keep the same graph size and increased the number of virtual machines per node, and with four nodes.

The experiment results are presented in Figure 6. The execution time for BFS traversal is reported in Y-axis and number of Virtual Machines (VMs) per node is reported on X-axis. With increase in number of VMs per node, the total number of participating processes also increases. As it can be observed from the figure, the execution time reduces as we increase the number of VMs per node. But, with 16 VMs per node the execution time increases, even though the number of participating processes increase. The 16 VMs per node



(a) MPI Alltoall Latency

(b) MPI AllReduce Latency

(c) MPI Bcast Latency

Figure 5. Impact of Virtual Machine Configurations

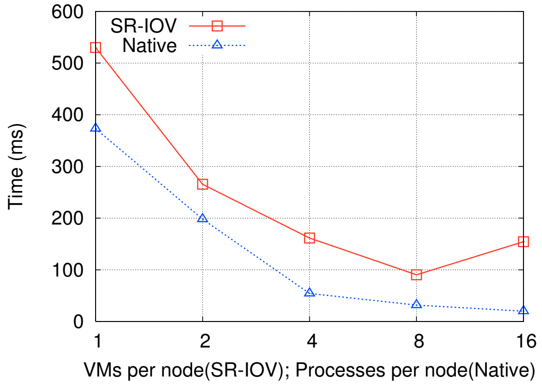


Figure 6. Impact of Virtual Machine Scalability

is ‘fully-subscribed’ mode, as each host node has 16 CPU cores. We compare these results with the native mode. In this configuration, we used four physical nodes and changed the number of processes per node from 1 to 16. We can see that there is significant difference in performance between native and SR-IOV modes. With 16 processes per node, the execution decreases further as compared to 8 processes per node. But for SR-IOV, the execution time increases for 16 VMs per node as compared to 8 VMs per node. These results reveal performance degradation at fully-subscribed mode.

## VI. Related Work

Recent studies about I/O virtualization mainly focus on two categories: one adopts a software-based approach and the other is hardware-based. For example, articles [31] and [32] show network performance evaluation of software-based approaches in Xen. Studies [15], [16], [18] have demonstrated that SR-IOV is significantly better than software-based solutions for 10GigE networks. In [15], the authors provide a detailed performance evaluation on the environment of SR-IOV capable 10GigE Ethernet in KVM. They have studied several important factors that affect network performance in both virtualized and native systems. Authors in [16] have conducted experiments to compare SR-IOV performance with a paravirtualized network driver. The results show that SR-IOV can achieve high performance, high scalability, and with a low CPU overhead at the same time. Article [18] has addressed two important issues: redundant interrupts and single-threaded NAPI, which affect performance and scalability of SR-IOV with 10GigE network. Their results also demonstrate that SR-IOV approach can achieve high performance I/O in a KVM-based virtualized environment. Further, studies [33], [34], [35], [36] with Xen demonstrated the ability to achieve near-native performance in VM-based environment for HPC.

As discussed in the above section, current research on SR-IOV mainly pays attention to the environment of 10GigE network. However, SR-IOV has been introduced for InfiniBand recently. Therefore, it is very important for researchers and engineers to fully understand the performance benefits of SR-IOV in InfiniBand network. This paper concentrates on this scenario, which is different from other works. In this paper, we provide a detail performance evaluation of both IB-level primitives and MPI/UPC-Level benchmarks.

## VII. Conclusions

In this work, we presented our initial evaluation results of using SR-IOV with InfiniBand. We explored the different dimensions for evaluating the performance aspects such as InfiniBand communication modes, communication progress modes and virtual machine configuration modes. Based on these dimensions, we presented the performance characteristics of MPI and UPC communication runtimes.

Our experimental evaluations showed that for the performance of MPI and PGAS point-to-point communication benchmarks over SR-IOV with InfiniBand is comparable to that of the native InfiniBand hardware, for medium and large message lengths. However, we observed that the performance of MPI collective operations over SR-IOV with InfiniBand is inferior when compared to the native designs.

## VIII. Acknowledgment

The authors would like to thank Jonathan Perkins of The Ohio State University for extending help in setting up the experimental systems. This research is supported in part by National Science Foundation grants #OCI-0926691, #OCI-1148371 and #CCF-1213084.

# 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] K. Adams and O. Agesen, "A Comparison of Software and Hardware Techniques for x86 Virtualization," in *ACM SIGOPS Operating Systems Review*, vol. 40, no. 5. ACM, 2006, pp. 2–13.

[3] AMD, "AMD Secure Virtual Machine Architecture Reference Manual, Rev. 3.01, May 2005."

[4] R. Uhlig, G. Neiger, D. Rodgers, A. Santoni, F. Martins, A. Anderson, S. Bennett, A. Kagi, F. Leung, and L. Smith, "Intel Virtualization Technology," *Computer*, vol. 38, no. 5, pp. 48–56, 2005.

[5] MPI Forum, "MPI: A Message Passing Interface," in *Proceedings of Supercomputing*, 1993.

[6] "Partitioned Global Address Space," http://www.pgas.org/.

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

[8] Top500 Supercomputing System, http://www.top500.org.

[9] "IP over InfiniBand Working Group," http://www.ietf.org/html.charters/ipoib-charter.html.

[10] J. Sugerman, G. Venkitachalam, and B. Lim, "Virtualizing I/O Devices on VMware Workstations hosted Virtual Machine Monitor," in *USENIX Annual Technical Conference*, 2001.

[11] Kernel Virtual Machine, http://kvm.qumranet.com/.

[12] J. Santos, Y. Turner, G. Janakiraman, and I. Pratt, "Bridging the Gap Between Software and Hardware Techniques for I/O Virtualization," in *Proceedings of the USENIX08 Annual Technical Conference*, 2008.

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

[14] Multi-Root I/O Virtualization, http://www.pcisig.com/specifications/iov/multi-root/.

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

[16] 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.

[17] J. Suzuki, Y. Hidaka, J. Higuchi, T. Baba, N. Kami, and T. Yoshikawa, "Multi-Root Share of Single-Root I/O Virtualization (SR-IOV) Compliant PCI Express Device," in *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*. IEEE, 2010, pp. 25–31.

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

[19] MVAPICH2/MVAPICH2-X: MPI/PGAS over InfiniBand, 10GigE/iWARP and RoCE, http://mvapich.cse.ohio-state.edu/.

[20] M. Koop, J. Sridhar, and D. Panda, "TupleQ: Fully-Asynchronous and Zero-Copy MPI over InfiniBand," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1–8.

[21] M. J. Koop, J. K. Sridhar, and D. K. Panda, "Scalable MPI Design over InfiniBand using eXtended Reliable Connection."

[22] M. Koop, T. Jones, and D. Panda, "MVAPICH-Aptus: Scalable High-Performance Multi-Transport MPI over InfiniBand," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–12.

[23] S. Sur, M. Koop, and D. Panda, "High-Performance and Scalable MPI over InfiniBand with Reduced Memory Usage: an In-Depth Performance Analysis," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. ACM, 2006, p. 105.

[24] J. Jose, K. Kandalla, M. Luo, and D. Panda, "Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation," in *Parallel Processing (ICPP), 2012 41st International Conference on*. IEEE, 2012, pp. 219–228.

[25] M. Koop, M. Luo, and D. Panda, "Reducing Network Contention with Mixed Workloads on Modern Multicore Clusters," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*. IEEE, 2009, pp. 1–10.

[26] MLNX_OFED (OpenFabrics Enterprise Distribution), http://www.mellanox.com/related-docs/prod_software/PB_OFED.pdf.

[27] NetPerf, http://www.netperf.org/netperf/.

[28] OSU Micro-benchmarks, http://mvapich.cse.ohio-state.edu/benchmarks/.

[29] J. Liu, W. Huang, B. Abali, and D. Panda, "High Performance VMM-bypass I/O in Virtual Machines," in *Proceedings of the annual conference on USENIX*, vol. 6, 2006, pp. 3–3.

[30] The Graph500, http://www.graph500.org.

[31] 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.

[32] 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.

[33] 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, 2007.

[34] 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.

[35] 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. ATEC '06, Berkeley, CA, USA, 2006.

[36] 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.