2016 IEEE International Parallel and Distributed Processing Symposium Workshops

Performance Characterization of Hypervisor- and Container-based Virtualization for HPC on SR-IOV Enabled InfiniBand Clusters

Jie Zhang, Xiaoyi Lu and Dhabaleswar K. (DK) Panda
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH USA 43210
Email: {zhanjie, luxi, panda}@cse.ohio-state.edu

***Abstract*—Hypervisor (e.g. KVM) based virtualization has been used as a fundamental technology in cloud computing. However, it has the inherent performance overhead in the virtualized environments, more specifically, the virtualized I/O devices. To alleviate such overhead, PCI passthrough can be utilized to have exclusive access to I/O device. However, this way prevents the I/O device from sharing with multiple VMs. Single Root I/O Virtualization (SR-IOV) technology has been introduced for high-performance interconnects such as InfiniBand to address such sharing issue while having ideal performance. On the other hand, with the advances in container-based virtualization (e.g. Docker), it is also possible to reduce the virtualization overhead by deploying containers instead of VMs so that the near-native performance can be obtained. In order to build high-performance HPC cloud, it is important to fully understand the performance characteristics of different virtualization solutions and virtualized I/O technologies on InfiniBand clusters. In this paper, we conduct a comprehensive evaluation using IB verbs, MPI benchmarks and applications. We characterize the performance of hypervisor- and container-based virtualization with PCI passthrough and SR-IOV for HPC on InfiniBand clusters. Our evaluation results indicate that VM with PCI passthrough (VM-PT) outperforms VM with SR-IOV (VM-SR-IOV), while SR-IOV enables efficient resource sharing. Overall, the container-based solution can deliver better performance than the hypervisor-based solution. Compared with the native performance, container with PCI passthrough (Container-PT) only incurs up to 9% overhead on HPC applications.**

Keywords*—*Virtualization, Container, Cloud Computing, PCI Passthrough, SR-IOV, InfiniBand

I. Introduction

Virtualization technology has been developing rapidly over the past few decades. During the period, several different virtualization solutions, such as Xen [29], VMWare [27], and KVM [15], are proposed and improved by community. The main benefits of virtualization include hardware independence, high availability, isolation, security and easy management. Therefore, it is considered as one of the foundation of cloud computing. Virtualization technology has already been widely adopted in industry computing environments, especially data-centers. For instance, the data-center provider, Amazon's Elastic Compute Cloud (EC2) [1], relies on virtualization to consolidate computational resources for applications from different customers, with required Quality of Service guarantees on the same underlying hardware.

Despite virtualization offering many benefits and widely being used in enterprise computing domain, it is still being treated as a second-class citizen in most HPC contexts due to its inherent performance overhead. Past studies [17], [6], [10] have shown that the traditional hypervisor-based virtualization has considerable overhead, especially in terms of I/O, which constrains its applicability in HPC. To alleviate such high I/O overhead, PCI passthrough mechanism can be utilized by the hypervisor and hardware support from Intel (VT-d) [11] or AMD (AMD-V) [4]. PCI passthrough allows virtual machines (VMs) to have exclusive access to the particular I/O devices so that I/O performance can be maximized. However, the I/O device can not be shared by multiple VMs through PCI passthrough. Further, Single Root I/O Virtualization (SR-IOV) [22] is proposed to deal with such sharing issue, while having ideal I/O performance among VMs.

High performance interconnects, such as InfiniBand, as a type of high speed I/O device has been widely used on many modern HPC clusters. The statistics of TOP500 [26] supercomputers on November 2015 indicate that InfiniBand is the most used underlying interconnects on the TOP500 with 237 systems (47.4%). As InfiniBand provides low latency and high throughput for data transfer with both PCI passthrough and SR-IOV support, it has potential to become a high performance interconnect infrastructure for HPC cloud.

On the other hand, container-based virtualization (e.g. Docker) has been popularized recently as a lightweight virtualization solution [23]. There are several studies focusing on performance characterization of different virtualization solutions in the HPC context. We summarize and compare these existing studies in Table I. Studies [7], [31], [20] present the performance evaluation on Ethernet, instead of InfiniBand, and none of them covers SR-IOV. Studies [14], [16] focus on InfiniBand platform. However, study [14] only provides the performance evaluation on different VM configurations under KVM with SR-IOV, while study [16] only shows the Docker performance in terms of different versions of compiler and MPI library. Thus, there exists little previous work providing a comprehensive evaluation to cover all of the aspects in Table I, even though such an evaluation is very important for building high performance HPC cloud.

Therefore, we conduct a comprehensive performance characterization in this paper for the two virtualization solutions (VM and container) and two virtualized I/O technologies (PCI passthrough and SR-IOV) on InfiniBand clusters using a collection of IB verbs level, MPI level benchmarks and HPC

*This research is supported in part by National Science Foundation grants #CNS-1419123, #IIS-1447804, and #ACI-1450440.

TABLE I: Comparison with Existing Studies

	InfiniBand	KVM	Docker	PCI passthrough	SR-IOV
[7], [31], [20]	×	√	√	√	×
[12], [13], [14]	√	√	×	×	√
[16]	√	×	√	√	×
This paper	√	√	√	√	√

applications.

We try to answer the following three questions through our study:

1) How much performance difference can be observed between VM and container and what are the overheads compared to the native environment?
2) How does PCI passthrough and SR-IOV impact the communication performance?
3) Can container deliver near-native performance for HPC end applications and reveal the promising future of the adoption in HPC cloud, especially with high performance interconnects, such as InfiniBand?

Through the performance evaluation, we observe that VM with PCI passthrough mode (VM-PT) outperforms VM with SR-IOV (VM-SR-IOV), while VM-SR-IOV can enable efficient resource sharing. Overall, the container-based solution can deliver better performance than the hypervisor-based solution. Compared with the native performance, container with PCI passthrough (Container-PT) only incurs up to 9% overhead on HPC applications.

The rest of the paper is organized as follows. Section II describes two types of virtualization solutions (hypervisor based and container based virtualization) with PCI passthrough and SR-IOV technology. Section III describes our evaluation methodology. Section IV presents the performance evaluation results using micro-benchmarks and applications and the comparison with the native performance. We discuss the related work in Section V, and conclude our work in Section VI.

II. Background

A. Hypervisor-based Virtualization

A hypervisor, also called a virtual machine manager, as shown in Figure 1(a), is a program that allows multiple guest operating systems to share a single hardware host. Each guest operating system appears to have the host's processor, memory, and other resources all to itself. We can also see that hypervisor is actually interacting with underlying host operating system or hardware to control the host processor and resources, allocate what is needed to each guest operating system in turn and make sure that the VMs cannot disrupt each other. Hypervisor provides a powerful tool through which to consolidate multiple servers and take better advantage of the available physical computing resources. However, it introduces additional workload-dependent overhead as they faithfully replicate true hardware behaviors. The common hypervisors include VMware vSphere ESXi, Citrix XenServer, open source KVM (Kernel-based Virtual Machine), Microsoft's Hyper-V, Oracle VM VirtualBox, etc. KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V), which allows a user space program to utilize the hardware virtualization features of various processors [15], [19]. In this paper, we use KVM as a representative hypervisor to run VMs.

B. Container-based Virtualization

In container-based virtualization, the host kernel allows the execution of several isolated userspace instances that share the same kernel but possibly run a different software stack (system libraries, services, applications), as shown in Figure 1(b). Container-based Virtualization does not introduce a layer of virtual hardware, however, it provides self-contained execution environments, effectively isolating applications that rely on the same kernel in the Linux operating system. Containers are built around two core mature Linux technologies. First, cgroups (control groups) can be used to group processes and limit their resources usage. Second, namespace isolation can be used to isolate a group of processes at various levels: networking, filesystem, users, process identifiers, etc. Several container-based solutions have been developed, such as LXC, Googles lmctfy, Docker. Docker [5] is a popular open-source platform for building and running containers and offers several important features, including portable deployment across machines, versioning and reuse of container image and a searchable public registry for images. We deploy Docker to run containers in this paper.

C. PCI Passthrough and SR-IOV

PCI passthrough allows giving control of the physical devices to guests: that is, you can use PCI passthrough to assign a PCI device (NIC, disk controller, HBA, USB controller, sound card, etc.) to a guest domain, giving it full and direct access to the PCI device. This provides two important benefits. One is the near-native performance that can be achieved using device passthrough. This is perfect for the networking applications (or those that have high disk I/O) that have not adopted virtualization because of contention and performance degradation through the hypervisor (to a driver in the hypervisor or through the hypervisor to a user space emulation). The other one is the exclusive use of a device that is not inherently shareable. To efficiently share PCI device with multiple VMs, Single Root I/O Virtualization (SR-IOV) [22] has been proposed. It is a PCI Express (PCIe) standard which specifies the native I/O virtualization capabilities in PCIe adapters. In SR-IOV, a PCIe device presents itself as multiple virtual devices and each virtual device can be dedicated to a single VM.

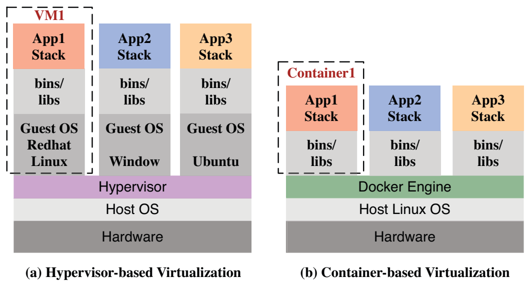


Fig. 1: Hypervisor-based and Container-based Virtualization

III. Evaluation Methodology

We follow a three-dimensional approach to conduct the performance characterization of hypervisor- and container-based virtualization on SR-IOV enabled InfiniBand clusters, as shown in Figure 2. The three dimensions are virtualization solution, virtualized I/O technology and benchmarks and applications, respectively.

Virtualization Solution In this dimension, we choose KVM as a representative hypervisor based solution, and Docker as a representative container based solution. The native performance is also provided to evaluate the related overhead. This will show us the performance of different virtualization solutions and their respective overheads.

Virtualized I/O Technology We use PCI passthrough and SR-IOV in this dimension. Through PCI passthrough mechanism, a VM/container will be assigned a physical HCA device of host, and have direct control of it. However, this dedicated way of passthrough considerably constrains the use of physical HCA from multiple VMs/containers on a single node. On SR-IOV enabled InfiniBand clusters, each Virtual Function (VF) of a physical HCA device can be attached to a VM. Therefore, multiple VMs on the same node are able to share the physical HCA device through VFs. We evaluate the container performance with PCI passthrough mechanism. The evaluation on this dimension helps us learn the performance of these virtualized I/O technologies.

Benchmarks & Applications We use different levels of micro-benchmarks and applications in this dimension. It contains IB verbs level, MPI level micro-benchmarks and applications. Table II lists all tests in the corresponding micro-benchmarks and applications in detail. We start the performance studies with IB verbs level tests, followed by point-to-point and collective operations at MPI level. Point-to-point operations are further categorized into two-sided and one-sided operations. Finally, four representative HPC applications are selected to integrally evaluate the performance under different virtualization solutions and virtualized I/O technologies. By following the above methodology, we present comprehensive performance characterization of hypervisor- and container-based virtualization for HPC on SR-IOV enabled InfiniBand clusters.

Deploying HPC Cloud Testbed with MVAPICH2-Virt Appliance We make a dedicated appliance in order to quickly and efficiently build HPC cloud on bare metal InfiniBand nodes, as shown in Figure 3. The IOMMU module is enabled to support hypervisor based virtualization, more specifically the hardware-assisted virtualization. The Mellanox OFED is pre-installed to enable InfiniBand communication capability and SR-IOV support. In order to maximize the performance, CPU frequency scaling module is disabled to have CPU run at full speed. Inside this appliance, we provide a VM disk image in the format of qcow2 and a corresponding VM launch script. To keep the size of VM disk image small, we utilize an initialization script to automatically detect, download and install Mellanox OFED, MVAPICH2-Virt library and other necessary packages during the system boot. In addition, IVSHMEM is enabled to optimize the intra-node inter-VM MPI communication. Therefore, the users who deploy this appliance in their HPC cloud can easily launch VMs to carry out MPI level experiments with improved communication performance. This appliance is publicly available on Chameleon [2] Cloud.

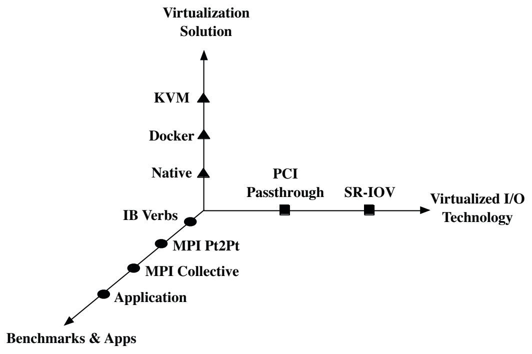


Fig. 2: Evaluation Dimensions

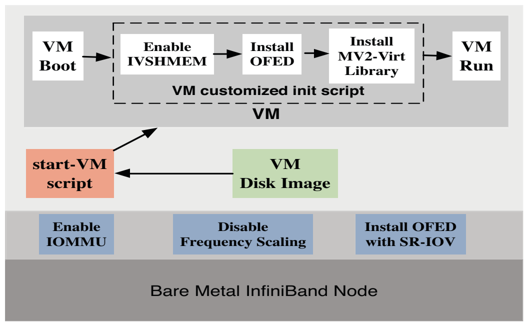


Fig. 3: MVAPICH2-Virt Appliance

IV. Performance Evaluation

In this section, we describe our experimental testbed and discuss our evaluation results in terms of virtualization solution, virtualized I/O technology and benchmarks and applications, as discussed in Section III.

A. Experiment Setup

We use eight bare metal InfiniBand nodes on **Chameleon Cloud** as our testbed, where each node has 24-core 2.3 GHz Intel Xeon E5-2670 processors with 128 GB main memory and equipped with Mellanox ConnectX-3 FDR (56 Gbps) HCAs with PCI Express Gen3 interfaces. We use CentOS Linux release 7.1.1503 (Core) with kernel 3.10.0-229.el7.x86_64 as the host OS and MLNX_OFED_LINUX-3.0-1.0.1 as the HCA driver to have SR-IOV support.

On hypervisor based virtualization front, KVM is used as the Virtual Machine Monitor (VMM). We deploy single VM per node, because PCI passthrough can only have one VM with the IB device. Each VM has 24 cores, 32 GB memory and the same OS as host OS. HCA is attached to VM by PCI passthrough and SR-IOV technology, respectively, which are presented as 'VM-PT' and 'VM-SR-IOV' in the following section.

On container based virtualization side, Docker 1.8.2 is deployed as the engine to build and run Docker containers. To

TABLE II: Benchmarks and representative HPC Applications for Evaluation

IB Verbs	Pt2Pt-2Sided	Pt2Pt-1Sided	Collective	Application	
ib_send_lat	lat	put-lat	bcast	**Graph500**	Graph500 [25] is one of the representative benchmarks of Data intensive supercomputer applications. It exhibits highly irregular communication pattern.
ib_send_bw	bw	put-bw	allgather	**NAS**	NAS [3] contains a set of benchmarks which are derived from the computing kernels, which is common on Computational Fluid Dynamics (CFD) applications. These represent the class of regular iterative HPC applications.
ib_read_lat	bibw	put-bibw	allreduce	**LAMMPS**	LAMMPS stands for Large-scale Atomic/Molecular Massively Parallel Simulator [21]. It is a classical molecular dynamics simulator from Sandia National Laboratory.
ib_read_bw		get-lat	alltoall	**SPEC MPI 2007**	The SPEC MPI 2007 benchmark suite [24]is for evaluating MPI-parallel, floating point, compute intensive performance across a wide range of cluster and SMP hardware.
ib_write_lat		get-bw			
ib_write_bw		acc-lat			

have a fair comparison, we also run single container on each node. The privileged option is enabled to give the container access to the host HCA, which is presented as 'Container-PT' in the following evaluation.

All applications and libraries used in this study are compiled with gcc 4.8.3 compiler. We use Perftest as the benchmark to carry out IB Verbs level experiments. All MPI communication performance experiments use MVAPICH2-2.2b and OSU micro-benchmarks v5.0. Experimental results are averaged across multiple runs to ensure fair comparison.

B. IB Verbs Level Performance

The IB verbs level tests can be used to demonstrate the basic performance of the IB device on data transfer. Figure 4 shows the evaluation results using IB-Send-Lat/BW, IB-Read-Lat/BW and IB-Write-Lat/BW. In IB-Send-Latency test, as shown in Figure 4(a), we observe that VM passthrough performs better than VM SR-IOV mechanism as the introduction of virtualization layer, especially in the range of small message sizes. The benefit can be up to 24%. For example, the latency of VM-PT is 0.91μs, while it is 1.20μs for VM-SR-IOV at 8 bytes message size. We also find that VM-PT has minor overhead, within 3%, compared with native performance. On the container side, the results of Container-PT indicate that it also has minor overhead compared to the native performance on small and large message sizes. However, as we can see in Figure 4(a), there exists more overhead, around 17% on medium message sizes ranging from 256 bytes to 8 Kbytes. In IB-Send-BW test, as shown in Figure 4(d), the evaluation results present similar performance among Container-PT, VM-PT and VM-SR-IOV. The peak bandwidth can achieve 6 Gbytes/sec, and they all have minor overhead, within 8%, compared to the corresponding native performance. In IB-Write-Lat/BW tests, we can observe from Figure 4(b) and 4(e) that, they show similar performance as IB-Send-Lat/BW. In IB-Read-Latency test, as shown in Figures 4(c) and 4(f), where Container-PT and VM-PT have minor overhead, around 7%, compared with native performance. VM-PT outperforms VM-SR-IOV by around 14%. And they have similar performance on IB-Read-BW test.

C. Point-to-Point Communication Performance

Figure 5 shows the performance of two-sided point-to-point communication at MPI level. In latency test, as shown in Figure 5(a), we see that VM-PT has lower latency, compared with VM-SR-IOV. For example, the latency of VM-PT is 1.09μs, while it is 1.41μs for VM-SR-IOV at 8 bytes message size. The difference can be up to 23%. The experimental results indicate that VM-PT has less than 4% overhead compared to the native performance. For Container-PT, it performs similarly as native environment for most message sizes. However, there is still around 12% overhead on medium message sizes. The overhead is similar as verbs-level numbers, which means no extra overhead is being introduced in MPI layer design. This could be the inherent overhead in current generation hardware, hypervisor and vendor drivers.

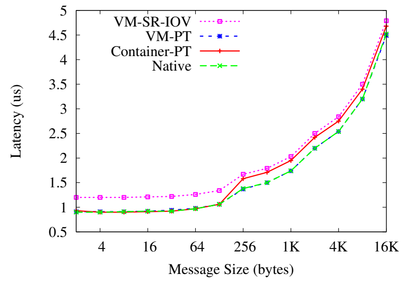
In bandwidth and bidirectional-bandwidth tests, as shown in Figures 5(b) and 5(c), the evaluation results indicate that VM-PT, Container-PT and VM-SR-IOV can achieve similar performance as native environment with around 10% overhead. The peak bandwidth for these two tests are around 6 Gbytes/sec and 10 Gbytes/sec, respectively. The results are consistent with the verbs-level evaluation as well.

Figures 6(a)-6(f) present the evaluation results of MPI one-sided point-to-point communication. In the tests of put operations, as shown in Figures 6(a)-6(c), the evaluation results indicate that VM-PT outperforms VM-SR-IOV by around 20%, 18% and 16% in terms of put latency, put bandwidth and put bidirectional bandwidth. For example, the put latency of VM-PT is 1.49μs, while it is 1.88μs for VM-SR-IOV at 8 bytes message size. VM-PT can achieve near-native performance, within 7% overhead for all message sizes. For Container-PT, we still see around 13%, 18% and 13% overhead for medium message sizes in terms of put latency, put bandwidth and put bidirectional bandwidth.

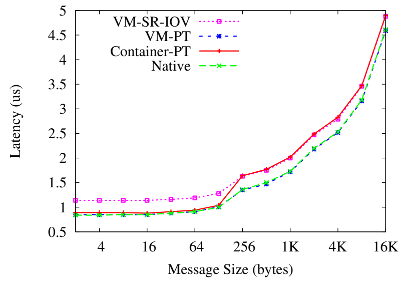
Similarly, VM-PT performs better than VM-SR-IOV by around 14%, 19%, 18% in terms of get latency, get bandwidth and accumulate latency. From Figures 6(d)-6(f), we can also see that both VM-PT and Container-PT can deliver near-native performance with less than 9% overhead.

D. Collective Communication Performance

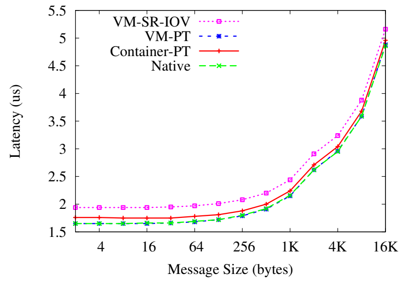
We select four commonly used collective communication operations Broadcast, Allgather, Allreduce and Alltoall, and run them in full subscription mode for our evaluations, as shown in Figures 7(a)-7(d). The evaluation results indicate that, compared to VM-SR-IOV, VM-PT brings up to 20%, 20%, 23%, 31% performance improvement, respectively. The results also show that Container-PT has less overhead than VM-PT. Compared to native performance, VM-PT incurs up to



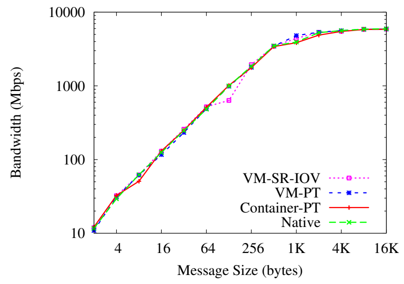
(a) IB Send Latency



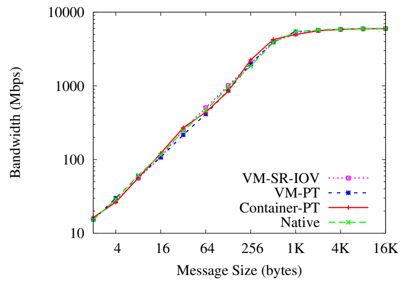
(b) IB Write Latency



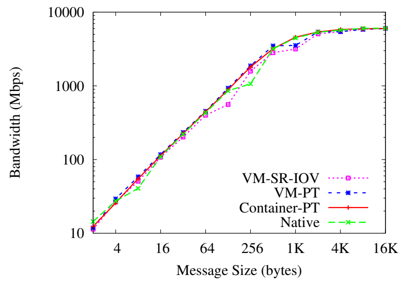
(c) IB Read Latency



(d) IB Send Bandwidth

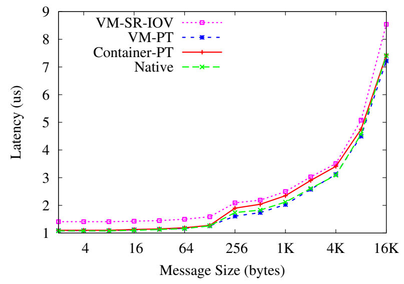


(e) IB Write Bandwidth

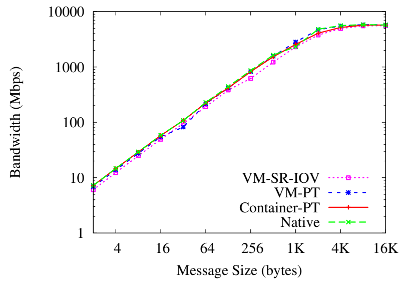


(f) IB Read Bandwidth

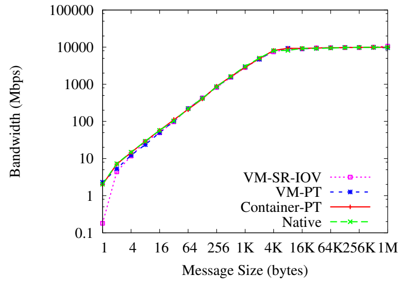
Fig. 4: IB Verbs Level Performance



(a) MPI Point-to-Point Latency



(b) MPI Point-to-Point Bandwidth



(c) MPI Point-to-Point Bi-Bandwidth

Fig. 5: MPI Two-Sided Point-to-Point Communication Performance

20%, 15%, 15%, 20% overhead respectively, while Container-PT only has less than 10% overhead in all four collective operations.

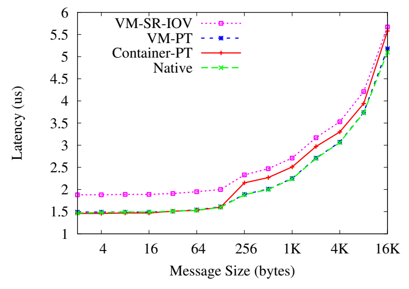
E. Application Performance

Figures 8(a)-8(c) depict the evaluation results of Graph500, SPEC, Class C NAS and LAMMPS with 128 processes. Compared to VM-SR-IOV, VM-PT reduces the total execution time by up to 27%(24,10), 24%(milc), 22%(LU) and 20%(lj, 40% reduction on communication), respectively. As Graph500 is a communication-intensive workload, we see more benefits on it. The results also indicate that compared with VM-PT, Container-PT performs better, which is able to achieve near-native performance. The overheads are only up to 9% in all four applications, which are 9%(24,10), 9%(leslie3d), 5%(EP) and 9%(chain). Although we see more overhead in VM-SR-IOV case, it is needed to emphasize that SR-IOV is still necessary on flexibly building large scale HPC cloud, since it delivers the capability to efficiently share I/O devices with multiple VMs.

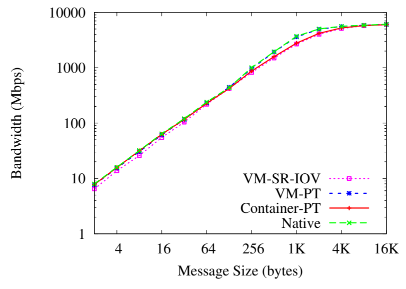
V. Related Work

There has been a large number of studies on optimizing and improving the hypervisor-based virtualization for HPC. Studies [17], [6], [10] demonstrate that SR-IOV is significantly better than software-based solutions for 10GigE networks. In [17], the authors provide a detailed performance evaluation on the environment of SR-IOV capable 10GigE with KVM. Further, studies [8], [18], [9] with Xen demonstrate the ability to achieve near-native performance in VM-based environment for HPC.

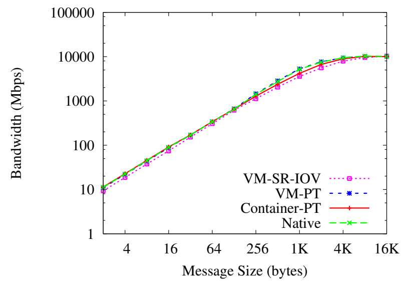
Our initial study of the performance characteristics of using SR-IOV with InfiniBand has shown that while SR-IOV enables low-latency communication [14], the MPI libraries need to be designed carefully and offer advanced features for improving intra-node, inter-VM communication [13], [12]. Based on these



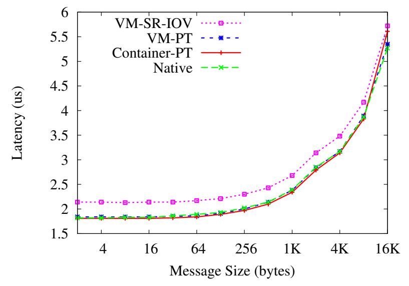
(a) MPI Put Latency



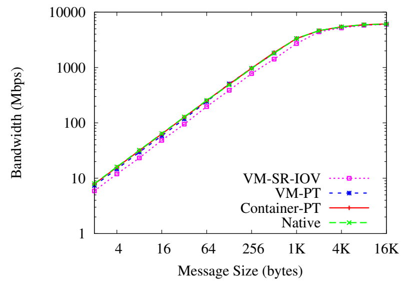
(b) MPI Put Bandwidth



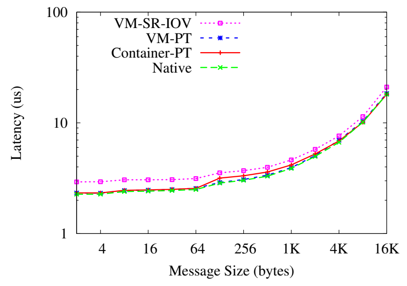
(c) MPI Put Bi-Bandwidth



(d) MPI Get Latency

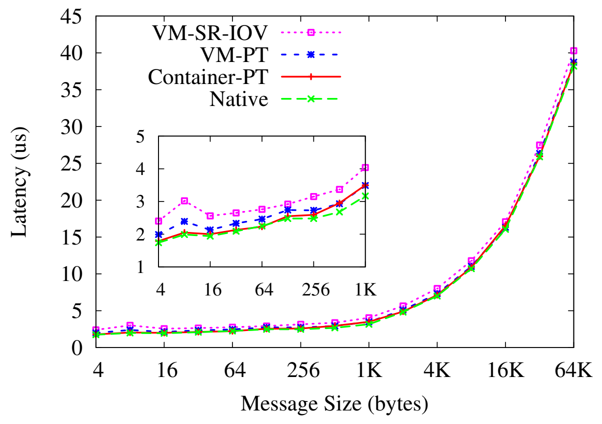


(e) MPI Get Bandwidth

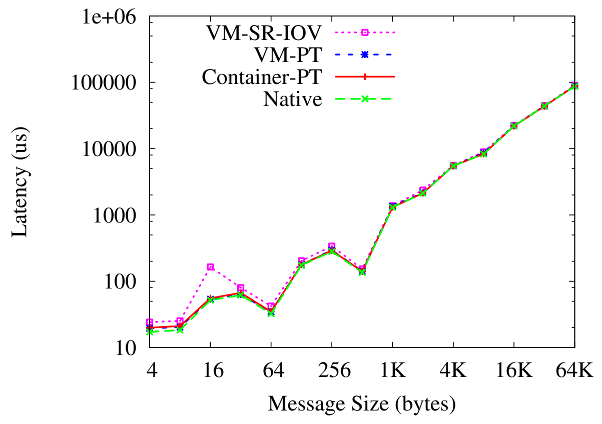


(f) MPI Accumulate Latency

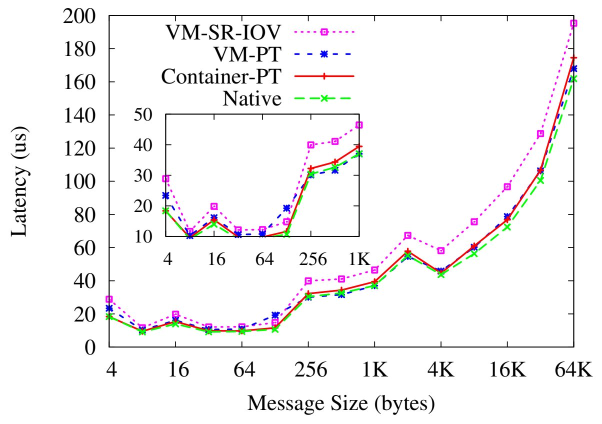
Fig. 6: MPI One-Sided Point-to-Point Communication Performance



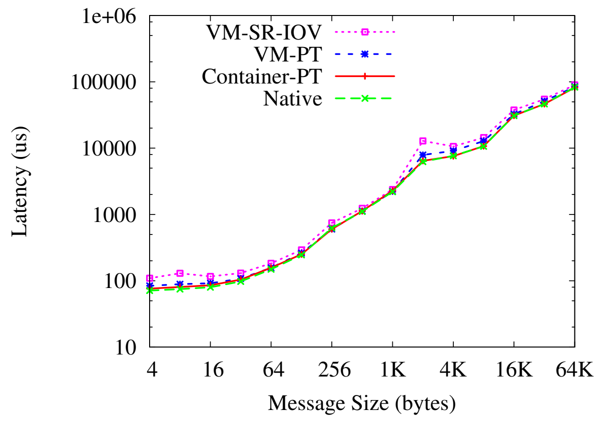
(a) Broadcast



(b) Allgather

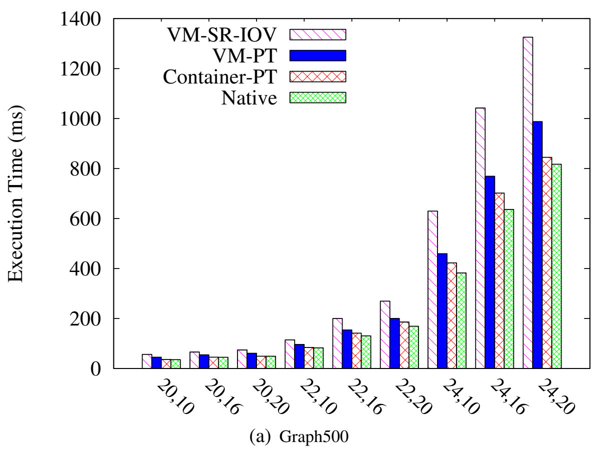


(c) Allreduce

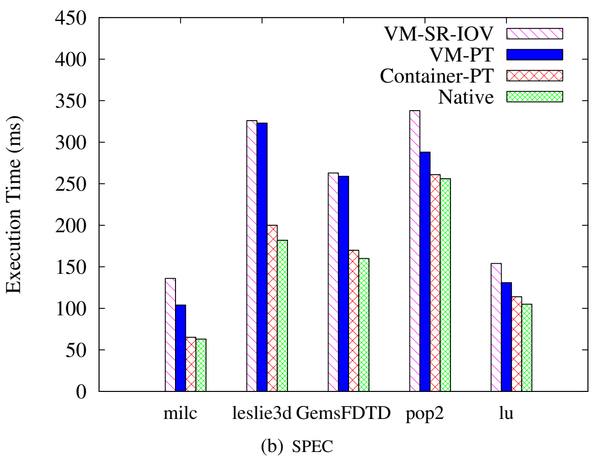


(d) Alltoall

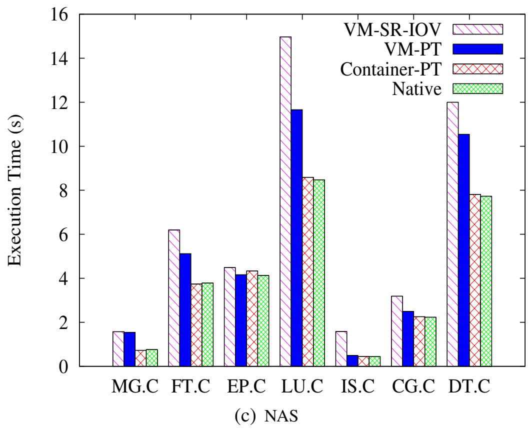
Fig. 7: Collective Communication Performance with 192 Processes



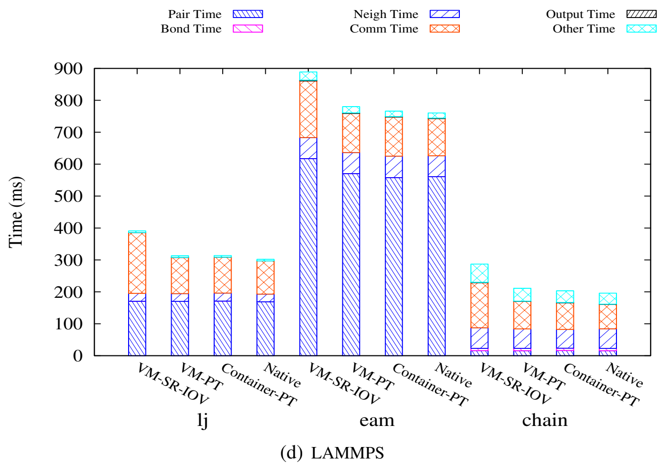
(a) Graph500



(b) SPEC



(c) NAS



(d) LAMMPS

Fig. 8: Application Performance with 128 Processes

studies, we provide an efficient approach to build HPC cloud with OpenStack over SR-IOV enabled InfiniBand clusters [30].

As an interesting alternative to the hypervisor-based solutions, container technology has been popularized during the last several years as a lightweight virtualization solution. More and more studies focus on evaluating the performance of different hypervisor-based and container-based solutions for HPC. Wes Felter et al. [7] explores the performance of traditional virtual machine deployments (KVM), and contrasts them with the use of Docker. They use a suite of workloads that stress CPU, memory, storage, and networking resources. Their results show that containers result in equal or better performance than VMs in almost all cases. In addition, they found that both VMs and containers require tuning to support I/O-intensive applications [7]. Xavier et al. [28] conduct an in-depth performance evaluation of container-based virtualization (Linux VServer, OpenVZ and LXC) and hypervisor-based virtualization (Xen) for HPC in terms of computing, memory, disk, network, application overhead and isolation. In the work of Cristian et al. [20], the authors evaluate the performance of Linux-based container solutions using the NAS parallel benchmarks, in various way of container deployment. The evaluation shows the limits of using containers, the type of application that suffer the most and until which level of oversubscription containers can deal with without impacting considerably the application performance. Yuyu et al. [31] compare the virtualization (KVM) and containerization (Docker) techniques for HPC in terms of features and performance using up to 64 nodes on Chameleon testbed with 10GigE networks.

However, the performance evaluation for different virtualization solutions and different virtualized I/O technologies is missing on InfiniBand clusters, which is a key component in HPC field. Therefore, we conduct a comprehensive performance evaluation using IB Verbs, point-to-point, collective micro-benchmarks and several representative HPC applications on InfiniBand clusters.

VI. Conclusion and Future Work

In this paper, we conduct a comprehensive performance evaluation for two virtualization solutions on SR-IOV enabled InfiniBand clusters. Figure 9 provides a summary of the evaluation.

Our evaluation in the dimension of virtualization solution shows that the container-based solution (Container-PT) can deliver better performance than the hypervisor-based solution

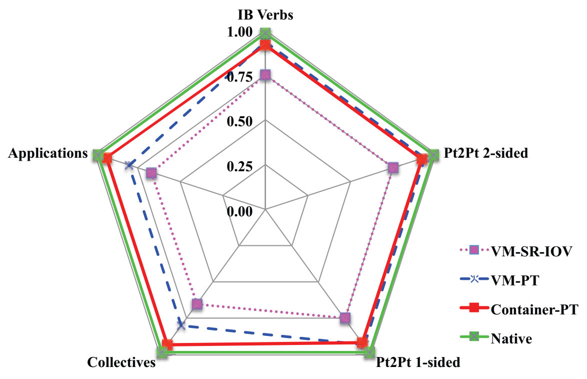


Fig. 9: Performance Evaluation Summary

(VM-PT and VM-SR-IOV) overall, as shown in Figure 9. Compared with the native performance, Container-PT has only up to 9% overhead on HPC applications. In addition, it is worth noting that container introduces a little more overhead on medium message sizes, compared to VM.

In the dimension of virtualized I/O technology, we compare PCI passthrough and SR-IOV mechanism. For the hypervisor-based virtualization, our evaluation results indicate that passthrough performs better than SR-IOV in almost all micro-benchmarks and applications we tested as shown with blue and magenta lines in Figure 9. Whereas we can not ignore the fact that SR-IOV is able to support multiple VMs to efficiently share the IB device, which is necessary on flexibly building large scale HPC cloud. For the container-based virtualization, we utilize PCI passthrough throughout the evaluation. We find that container with PCI passthrough is able to deliver near-native performance for the end applications.

In the future, we plan to have locality aware design for MPI over containers to improve the communication performance for intra-node inter-container case.

References

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

[2] "Chameleon," http://chameleoncloud.org/.

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

[4] AMD-V, http://www.amd.com/br/products/technologies/virtualization/.

[5] Docker, https://www.docker.com/.

[6] 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.

[7] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," Tech. Rep. RC25482 (AUS1407-001), 2014.

[8] 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.

[9] 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.

[10] 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.

[11] Intel Virtualization Technology, http://ark.intel.com/Products/VirtualizationTechnology.

[12] J. Zhang, X. Lu, J. Jose, M. Li, R. Shi, D. K. Panda, "High Performance MPI Library over SR-IOV Enabled InfiniBand Clusters," in *Proceedings of International Conference on High Performance Computing (HiPC)*, Goa, India, December 17-20 2014.

[13] 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*, Porto, Portugal, August 25-29 2014.

[14] 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.

[15] Kernel-based Virtual Machine (KVM), http://www.linux-kvm.org/page/Main_Page.

[16] C. Kniep, "Containerization of High Performance Compute Workloads using Docker," Tech. Rep., 2014.

[17] 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.

[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] qemu-kvm, http://wiki.qemu.org/KVM.

[20] C. Ruiz, E. Jeanvoine, and L. Nussbaum, "Performance Evaluation of Containers for HPC," in *10th Workshop on Virtualization in High-Performance Cloud Computing (VHPC)*, Vienna, Austria, Aug 2015.

[21] S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics," *J Comp Phys, 117, 1-19*, 1995.

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

[23] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors," in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, ser. EuroSys '07, Lisbon, Portugal, 2007, pp. 275–287. [Online]. Available: http://doi.acm.org/10.1145/1272996.1273025

[24] SPEC MPI 2007, https://www.spec.org/mpi2007/.

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

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

[27] VMWare, http://www.vmware.com/.

[28] M. Xavier, M. Neves, F. Rossi, T. Ferreto, T. Lange, and C. De Rose, "Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments," in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, Belfast, Northern Ireland, Feb 2013, pp. 233–240.

[29] Xen, http://www.xen.org/.

[30] J. Zhang, X. Lu, M. Arnold, and D. Panda, "MVAPICH2 over OpenStack with SR-IOV: An Efficient Approach to Build HPC Clouds," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, May 2015, pp. 71–80.

[31] Y. Zhou, B. Subramaniam, K. Keahey, and J. Lange, "Comparison of Virtualization and Containerization Techniques for High Performance Computing," in *Proceedings of the 2015 ACM/IEEE conference on Supercomputing*, Austin, USA, Nov 2015.