# Designing and Building Efficient HPC Cloud with Modern Networking Technologies on Heterogeneous HPC Clusters

Jie Zhang *(http://www.cse.ohio-state.edu/∼zhanjie)*
*Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, zhang.2794@osu.edu*
*Advisor: Dhabaleswar K. (DK) Panda, The Ohio State University, Columbus, OH, panda.2@osu.edu*

## I. Introduction

To meet the increasing demand for computational power, HPC clusters have grown tremendously in size and complexity. As the prevalence of high-speed interconnects, multi-/many-core processors, and accelerators continue to increase, efficient sharing of such resources is becoming more important to achieve faster turnaround time and reduce the cost per user. Furthermore, a large number of users, including many enterprise users, experience large variability in workloads depending on business needs, which makes predicting the required resources for future workloads a difficult task. For such users, cloud computing can be an attractive solution that offers on-demand resource acquisition, high configurability, and high performance at a low cost. This demand has been evidenced by the plethora of vendors offering such solutions including Amazon, Google, Microsoft, etc.

Virtualization technology, as one of the foundations of cloud computing, has been developing rapidly over the past few decades. Several different virtualization solutions, such as Xen [20], VMware ESX/ESXi [19], and KVM [12] are proposed and improved by the community. These hypervisor-based virtualization solutions bring several benefits, including hardware independence, high availability, isolation, and security. On the other hand, as a lightweight virtualization solution [18], container-based virtualization (such as Linux-VServer [14], Linux Containers (LXC) [13] or Docker [1]) has attracted considerable attention recently. In the container-based virtualization, the same host OS kernel is shared across containers. It thus leads to a more efficient way to provide virtualized computing environment to end users. With the emergence of container-based virtualization technology in clouds, another type of usage paradigm, which is called "nested virtualization", is becoming more and more popular in clouds. As a typical example, many end users choose to run their applications encapsulated by Docker containers over Amazon EC2 virtual machines. Such an approach of running containers nested in virtual machines can bring easy deployment benefit for end users while making the cloud easy-to-manage for administrators.

Even though virtualization has gained significant momentum in Cloud Computing, running HPC applications on Cloud systems with good performance is still challenging. One of the biggest hurdles is the lower performance of virtualized I/O devices [11], which limits the adoption of virtualized Cloud Computing systems for HPC applications. To address this issue, the community has recently introduced an enhanced networking capability, Single Root I/O Virtualization (SR-IOV) [17], 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. 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. Although SR-IOV is enabled in these systems, the communications between the the co-located instances also have to use SR-IOV, resulting in the performance overheads, which is the main drawback of SR-IOV that it does not have locality aware communication support. On the other hand, high performance MPI libraries in the HPC domain typically use shared memory based schemes for intra-host communication. Inter-VM Shared Memory (IVShmem) [15] has been proposed and can be hot-plugged to a VM as a virtualized PCI device to support shared memory backed intra-node-inter-VM communication. Therefore, MPI runtime needs to be redesigned to provide high performance virtualization support for virtual machines, containers and nested virtualization environments on the HPC clouds.

SR-IOV is able to provide efficient sharing of high-speed interconnect resources to VMs. However, as an essential virtualization capability towards high availability and resource provisioning, virtual machine migration with SR-IOV devices is still facing challenges. For instance, to successfully migrate a VM with SR-IOV-enabled InfiniBand virtual device, it is required to handle the challenges of detachment of an active IB device during migration and reestablishment of the IB connections after migration. And all of these need to be done transparently and efficiently for applications. Although several initial prototypes [2, 16, 21] have been proposed to support VM migration with SR-IOV devices, our investigations show that these approaches need to modify either hypervisors [21] or drivers of InfiniBand/Ethernet adapter [2] or both [16]. In the HPC environment, it will be very hard to request HPC resource administrators to run the modified version of the hypervisor in their clusters due to security concerns.

Obviously, it will also limit the usage of their designs on

*This research is supported in part by National Science Foundation grants #CCF-1519123, #ACI-1450440, #CNS-1513120, and #CCF-1565414.

HPC clouds with different adapters from different vendors. Consequently, if a hypervisor-independent and InfiniBand adapter driver-independent approach for virtual machine migration can be designed over SR-IOV enabled InfiniBand clusters, it will significantly benefit the HPC clouds.

In addition, for improved flexibility and resource utilization, it is important to manage and isolate virtualized resources of SR-IOV and IVShmem to support running multiple concurrent MPI jobs on the shared HPC systems. As this requires knowledge of and some level of control over the underlying physical hosts, it is difficult to achieve this with the MPI library alone, which is only aware of the virtual nodes and resources inside. Thus, extracting the best performance from virtualized clusters requires the proper support from other middleware like resource management and job scheduling systems, which have a global view of the VMs and the underlying physical hosts.

## II. Problem Statement

To deliver the near-native performance to the end HPC applications, the communication runtime needs to be able to provide high performance virtualization support in terms of different types of virtualization environments, such as virtual machines, containers, and nested virtualization. As the live migration is an essential feature in the cloud computing domain, it is important to provide high performance and scalable fault-tolerance/resilience capabilities on SR-IOV enabled HPC cloud. In addition, it is also necessary to co-design with resource management and job scheduling systems in order to achieve efficiently sharing of resources on modern HPC systems. To summarize, it addresses the following broad challenges:

1) Can MPI runtime be redesigned to provide high performance virtualization support for virtual machines and containers when building HPC clouds?
2) How much benefits can be achieved on HPC clouds with redesigned MPI runtime for scientific kernels and applications?
3) Can fault-tolerance/resilience (VM Live Migration) be supported on SR-IOV enabled HPC clouds?
4) Can the co-design with resource management and scheduling systems on modern HPC systems be provided to improve the resource utilization on HPC clouds?

## III. Research Highlights

Figure 1 depicts the research framework that I address the challenges highlighted above.

To summarize, it includes the following items:

1) The locality-aware MPI runtime is proposed to provide high performance virtualization support for different types of virtualization environments including hypervisor-based VMs, containers, nested virtualization on HPC cloud.

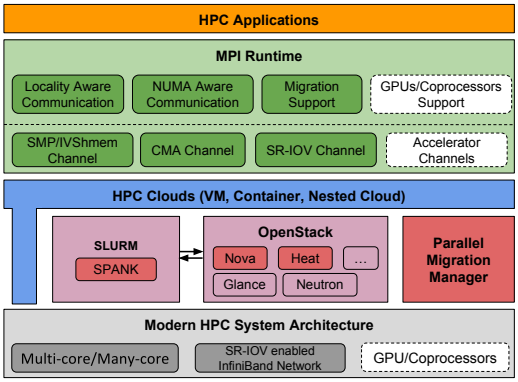


Figure 1. Research Framework

2) The hypervisor- and device driver-independent high performance virtual machine migration framework is proposed for MPI applications on SR-IOV enabled HPC cloud, which could overlap the migration with compuation.
3) The **Slurm-V** framework is proposed, which extends Slurm with virtualization-oriented capabilities such as job submission to dynamically created VMs with isolated SR-IOV and IVShmem resources.

### *A. High Performance Locality-aware MPI Runtime on HPC Clouds*

As introduced in Section I, SR-IOV still lacks high performance virtualization support for those co-resident instances, such as locality-aware and NUMA-aware support, which incurs the severe performance degradation. We propose the designs in MPI runtime to provide high performance virtualization support for different types of virtualization environments on HPC cloud. For hypervisor-based virtualization solution, I propose a high performance locality-aware MPI library, which can dynamically detect co-located VMs and coordinate communications between SR-IOV and IVShmem channels. Two important components are designed, namely Locality Detector and Communication Coordinator. Locality Detector maintains the information of local VMs on the same host, while 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 on 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. We further optimize these two different communication channels in VM environment to extract the optimal communication performance. The performance 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 [7–9].

For container-based virtualization, the locality-aware design and the IPC namespaces sharing are utilized to dynamically and efficiently detect co-resident containers at the communication runtime, so that the shared memory and CMA based communication can be executed to improve the communication performance across the co-resident containers. Compared with the default case, our proposed design can significantly improve the communication performance by up to 191%, 86%, and 16% in terms of MPI point-to-point and collective operations, and MPI applications, respectively [3, 4].

Nested virtualization is getting a lot of attraction. In this pattern, two or more levels of virtualization mechanisms are deployed. The upper-level virtual machines or containers run inside the guest operating systems of the lower-level virtual machines. However, it still has some performance issues because of the redundant call stacks and isolated physical resources. Further, I propose a high performance Two-layer Locality-aware and NUMA-aware MPI library for nested virtualization environment on HPC cloud. Two new components are added in the MPI library, which are Two-Layer Locality-aware Detector and Two-Layer NUMA-aware Communication Coordinator. The Two-Layer Locality Detector further contains three function units, VM Locality Detector, Container Locality Detector, and Nested Locality Combiner, respectively. Through the two-layer locality-aware design, MPI library is able to dynamically and efficiently detect co-resident containers in the same VM as well as co-resident VMs in the same host at runtime. Two-Layer NUMA-aware Communication Coordinator also contains three function units, which are Nested Locality Loader, NUMA Loader, and Message Parser, respectively. Through the NUMA-aware design, the MPI runtime is also able to adapt the different VM/container placement schemes on modern multi-core architecture and deliver the optimal communication performance to the end HPC applications. The evaluation results indicate that compared with the default performance, our proposed two-layer locality-aware and NUMA-aware design delivers up to 184%, 85% and 16% performance improvement in terms of MPI point-to-point and collective operations, and MPI applications, respectively. [6] The proposed high performance virtualization support for the co-resident instances is able to deliver the near-native performance for the end MPI applications with minor overheads on different types of virtualization environments.

### *B. Hypervisor- and Device Driver-independent High Performance VM Migration Framework on HPC Clouds*

SR-IOV is able to provide efficient sharing of high-speed interconnect resources and achieve near-native I/O performance. However, SR-IOV-based virtual networks prevent virtual machine migration, which is an essential virtualization capability towards high availability and resource provisioning. Current solutions have many restrictions, such as depending on specific network adapters and/or hypervisors, which will limit the usage scope of these solutions on HPC environments. We present a high-performance virtual machine migration framework for MPI applications on SR-IOV enabled HPC cloud. The framework is hypervisor-independent and host/guest device driver-independent. It includes a redesigned MPI runtime and a high performance and scalable external controller. The redesigned MPI runtime helps to suspend and reactivating IB connection before and after migration. The external controller is responsible for coordinating VM status and executing VM migration related operations. We propose two different migration designs inside the redesigned MPI runtime: Progress Engine (PE) based design and Migration Thread (MT) based design. The Progress Engine based design for VM migration has less overhead for MPI application but it can not overlap VM migration and application computation. The Migration Thread based design has a slightly higher overhead for MPI application but it allows overlapping between VM migration and application computation. Users can choose either one based on their applications' characteristics. The external controller utilizes multiple parallel libraries to notify MPI applications during migration, detach SR-IOV and IVShmem devices from VMs, migrate VMs, attach SR-IOV and IVShmem devices to the migrated VMs, and monitor migration status. It works seamlessly with the redesigned MPI runtime to significantly improve the efficiency of virtual machine migration. We systematically evaluated the proposed framework with MPI level micro-benchmarks and real-world HPC applications. At application level, for NPB LU benchmark running inside VM, our proposed design could completely hide the overhead of VM migration through computation and migration overlapping [5].

### *C. **Slurm-V** Framework*

To alleviate the cost burden, the HPC cluster resources need to be efficiently shared by the end users through virtualization. In this context, some critical HPC resources among VMs, such as SR-IOV enabled virtual functions and IVShmem devices, need to be enabled and isolated to support efficiently running multiple concurrent MPI jobs on HPC clouds. However, original Slurm is not able to supervise VMs and associated critical resources, such as VFs and IVShmem. We propose a novel framework, **Slurm-V**, which extends Slurm with virtualization-oriented capabilities such as job submission to dynamically created VMs with isolated SR-IOV and IVShmem resources. We propose three alternative designs for Slurm-V: Task-based design, SPANK plugin-based design, and SPANK plugin over OpenStack-based design, to manage and isolate IVShmem and SR-IOV resources for running MPI jobs. With Task-based design,

the end-user needs to implement corresponding scripts and explicitly insert them in the job batch file. After the job being submitted, `srun` will execute these three tasks on allocated nodes. With SPANK plugin-based design, I utilize the SPANK plugin architecture to dynamically extend the functions in different contexts during a Slurm job execution. It is more transparent to the end user compared to the Task-based design. With SPANK plugin over OpenStack-based design, some functionalities, such as VM launching and VM reclaiming, will be accomplished by offloading those tasks to underlying OpenStack infrastructure. The evaluation results indicate that the VM startup time can be reduced by up to 2.64X by using snapshot scheme. Compared with the single-threading scheme, multi-threading scheme reduces the VM startup time by up to 87%. In addition, Slurm-V framework shows good scalability and is able to support running multiple MPI jobs under different scenarios on HPC clouds [10].

## IV. Conclusion

In my research, I address several challenges on designing and building efficient HPC cloud with modern networking technologies on heterogeneous HPC clusters First, I proposed high performance locality-aware and NUMA-aware MPI runtime, which provides high performance virtualization support for various types of cloud environments including VMs, Containers, and nested virtualization platform. Second, I propose a high performance VM migration framework for MPI applications on SR-IOV enabled HPC cloud. The framework is hypervisor-independent and host/guest device driver-independent and could potentially overlap the migration overhead with computation. In addition, I propose a novel framework, **Slurm-V**, which extends Slurm with virtualization-oriented capabilities such as job submission to dynamically created VMs with properly isolated critical virtualized resources.

## References


[1] Docker. https://www.docker.com/.

[2] Wei Lin Guay, Sven-Arne Reinemo, Bjrn Dag Johnsen, Chien-Hua Yen, Tor Skeie, Olav Lysne, and Ola Trudbakken. Early Experiences with Live Migration of SR-IOV Enabled InfiniBand. *Journal of Parallel and Distributed Computing*, 2015.

[3] J. Zhang and X. Lu and D. K. Panda. Performance Characterization of Hypervisor-and Container-Based Virtualization for HPC on SR-IOV Enabled InfiniBand Clusters. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2016.

[4] J. Zhang, X. Lu, D. K. Panda. High Performance MPI Library for Container-based HPC Cloud on InfiniBand Clusters. In *Proceedings of the 45th International Conference on Parallel Processing (ICPP)*, Philadelphia, USA, August 16-19 2016.

[5] J. Zhang, X. Lu, D. K. Panda. High-Performance Virtual Machine Migration Framework for MPI Applications on SR-IOV enabled InfiniBand Clusters. In *Proceedings of the 31st IEEE International Parallel & Distributed Processing Symposium (IPDPS '17)*, Orlando, USA, 2017.

[6] J. Zhang, X. Lu, D. K. Panda. Designing Locality and NUMA Aware MPI Runtime for Nested Virtualization based HPC Cloud with SR-IOV Enabled InfiniBand. In *Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '17)*, Xi'an, China, 2017.

[7] 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.

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

[9] J. Zhang, X. Lu, M. Arnold, D. K. 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*, pages 71–80, May 2015.

[10] J. Zhang, X. Lu, S. Chakraborty, D. K. Panda. SLURM-V: Extending SLURM for Building Efficient HPC Cloud with SR-IOV and IVShmem. In *Proceeding of the 22nd International European Conference on Parallel and Distributed Computing (Euro-Par '16)*, Grenoble, France, August 2016.

[11] J. Jose, Mingzhe Li, Xiaoyi Lu, K.C. Kandalla, M.D. Arnold, and D.K. Panda. SR-IOV Support for Virtualization on InfiniBand Clusters: Early Experience. In *Proceedings of 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 385–392, May 2013.

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

[13] Linux Containers. https://linuxcontainers.org.

[14] Linux VServer. http://linux-vserver.org.

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

[16] Zhenhao Pan, Yaozu Dong, Yu Chen, Lei Zhang, and Zhijiao Zhang. CompSC: Live Migration with Pass-through Devices. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments*, VEE '12, pages 109–120, 2012.

[17] Single Root I/O Virtualization. http://www.pcisig.com/specifications/iov/single _root.

[18] Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry 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*, EuroSys '07, pages 275–287, Lisbon, Portugal, 2007. ISBN 978-1-59593-636-3. doi: 10.1145/1272996.1273025. URL `http://doi.acm.org/10.1145/1272996.1273025`.

[19] VMware ESX/ESXi. https://www.vmware.com/products/esxi-and-esx/overview.

[20] Xen. http://www.xen.org/.

[21] Xin Xu and Bhavesh Davda. SRVM: Hypervisor Support for Live Migration with Passthrough SR-IOV Network Devices. In *Proceedings of the12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '16, Atlanta, USA, 2016.