PROCEEDINGS OF SCIENCE

AIST Super Green Cloud: lessons learned from the operation and the performance evaluation of HPC cloud

Ryousei Takano[*], **Yusuke Tanimura, Akihiko Oota, Hiroki Oohashi, Keiichi Yusa, and Yoshio Tanaka** [†]
National Institute of Advanced Industrial Science and Technology (AIST)
Central 2, 1-1-1 UmezonoTsukuba, Ibaraki 305-8568, Japan
E-mail: {takano-ryousei, yusuke.tanimura, akihiko-oota, hiroki.oohashi, keiichi.yusa, yoshio.tanaka}@aist.go.jp

This paper describes a private cloud platform on our recently introduced supercomputer system, the AIST Super Green Cloud (ASGC). This is a fully virtualized High Performance Computing (HPC) system. ASGC is a 155 node-InfiniBand cluster with a theoretical peak performance of 69.44 TFLOPS. The software stack is built on the integration of cloud computing technologies including Apache CloudStack, KVM, Ceph cluster storage, and Zabbix monitoring system. To provide users with virtual clusters with near bare-metal performance, our extended CloudStack allows the user's guest OS to directly access an InfiniBand HCA by using PCI passthrough or SR-IOV. ASGC has been in operation since July 2014 with over 30 users and we report lessons learned from the first six months of operation. We can launch a highly productive system in a short development time. We also found several serious problems mainly related to Apache CloudStack, and we are improving it to achieve stable operation. We also report the elapsed time of a virtual cluster deployment and preliminary experimental results using InfiniBand HCA with SR-IOV. Our results show that 1) although the performance of SR-IOV is comparable to that of PCI passthrough, some unexpected performance degradation is often observed, and 2) VCPU pinning is effective for both improving the performance and reducing performance fluctuations.

International Symposium on Grids and Clouds 2015
15-20 March 2015
Academia Sinica, Taipei, Taiwan

[*]Speaker.
[†]This work was partly supported by JSPS KAKENHI Grant Number 24700040.

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

PoS(ISGC2015)004

1. Introduction

A network of supercomputers, including XSEDE, PRACE, and HPCI, is an attractive HPC platform that is capable of scale out. In order to make such an HPC platform practical, deployment of the applications across the platforms must be easy to do. We have proposed a multi-cloud deployment method in which a customized development and execution environment for applications can be deployed across the HPC platforms, in a "build once, run everywhere" manner, as shown in Figure 1. An application environment is contained in a set of virtual machine (VM) templates, which is shared among clouds. A virtual cluster created from the templates is deployed to any number of clouds, including a private cloud and public clouds like Amazon EC2.

Based on our vision of "build once, run everywhere," we have designed a fully virtualized HPC system, the AIST Super Green Cloud (ASGC). This HPC Cloud platform is built on the integration of state-of-the-art cloud computing technologies including the Apache CloudStack [1], KVM hypervisor [2], RADOS cluster storage [3], and Zabbix monitoring system [4]. We provide users with an Infrastructure as a Service (IaaS) type HPC Cloud, where they can freely customize a cluster environment for their applications. Unlike a common IaaS service, we provide not only a single VM but also *a virtual cluster*, which is a set of VMs interconnected by a high-speed network. To provide users with virtual clusters with near bare-metal performance, our extended CloudStack allows the user's guest OS to directly access an InfiniBand HCA by using PCI passthrough or SR-IOV [5]. Our previous work has reported that a virtual cluster achieves competitive performance as in the case of using the same physical resources [6]. Although we found some performance issues in a microscopic benchmark programs, the negative impact is limited on application level benchmark programs. The virtualization overhead is about 5%, even when the number of nodes increases up to 128.

This paper shows that firstly, lessons learned from the operation of ASGC, secondly, a performance evaluation of a virtual cluster deployment, and finally, a preliminary performance evaluation of SR-IOV. While our HPC Cloud employs PCI passthrough to enable a VM to directly access InfiniBnad HCA, we now consider the use of SR-IOV instead of PCI passthrough. SR-IOV enables multiple VMs to simultaneously access an InfiniBand HCA. Although it can help to improve the utilization of server resources, the performance impact is not clear. Therefore, we compared

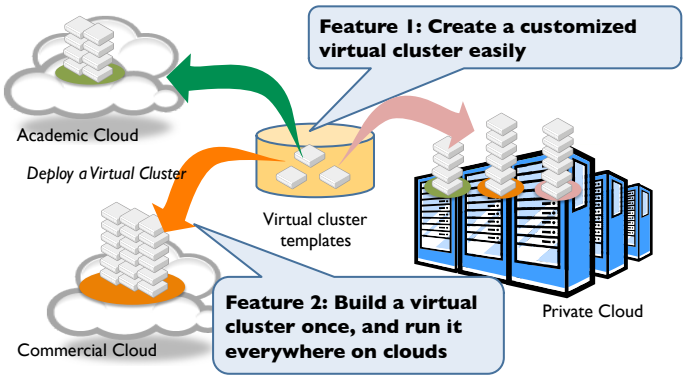


Figure 1: Vision of AIST HPC Cloud: "Build once, run everywhere."

PoS(ISGC2015)004

the performance using the several benchmark programs, including Intel Micro Benchmark and LAMMPS Molecular Dynamics Simulator. Our results show that 1) the performance of SR-IOV is comparable to that of PCI passthrough but unexpected performance degradation is often observed, and 2) VCPU pinning is effective for both improvement of the performance and reduction in the performance fluctuation.

The rest of the paper is organized into the following sections: Section 2 presents the overview of AIST Super Green Cloud and our HPC Cloud service, Section 3 discusses lessons learned from the operation of ASGC, Section 4 shows preliminary experimental results on a virtual cluster with SR-IOV, and finally Section 5 summarizes the paper and briefly mentions future work.

2. AIST Super Green Cloud

2.1 Usage Model of AIST HPC Cloud

As illustrated in Figure 2, for HPC users, we provide computing resources not only as a single standalone VM but also as a virtual cluster. The first step of the usage model is the user selects a template of a virtual cluster, which contains typical libraries and utilities for applications such as HPC, Hadoop, and so on. Next, the user installs additional software packages into the template and stores it in the repository. After that, the user can deploy a virtual cluster anytime from this template.

2.2 Compute Nodes

The AIST Super Green Cluster (ASGC) is the latest supercomputer at AIST. Its theoretical peak performance is 69.44 TFLOPS. It is composed of 155 nodes of Cray H2312 blade servers. Each compute node has two Intel Xeon E5-2680 v2 (Ivy Bridge EN) 2.8 GHz processors, 128 GB of memory (DDR3-1866), a 600 GB SSD, a Mellanox ConnectX-3 FDR InfiniBand HCA, and a 10 Gigabit Ethernet NIC. The Intel Xeon E5-2680 v2 supports the Intel-VT technology for hardware virtualization. The Hyper Threading capability was disabled. All nodes are connected by a full bisection bandwidth InfiniBand network, in which Mellanox SX6025 FDR switches are

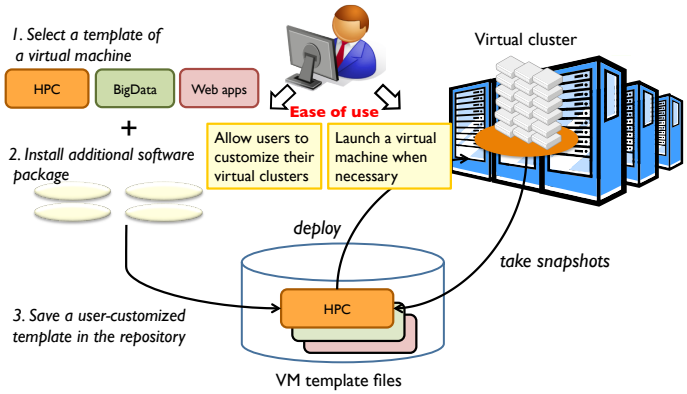


Figure 2: Usage model of AIST HPC Cloud.

PoS(ISGC2015)004

Table 1: AIST Super Green Cluster specifications.

Compute Node	
CPU	Dual 10-core Intel Xeon E5-2680 v2/2.8GHz
Chipset	Intel C600
Memory	128 GB DDR3-1866
InfiniBand	Mellanox ConnectX-3 (FDR)
Ethernet	Intel X520-DA2 (10GBASE-T)
Disk	Intel SSD DC S3500 600 GB
Switch	
InfiniBand	Mellanox SX6025
Ethernet	Extreme Networks BlackDiamond X8

used. The 10 Gigabit Ethernet network is also used for the console service, storage network, and so on. Table 1 summarizes the specifications of the node PC and the switch.

2.3 Software Stack

Our HPC Cloud platform is built based on Apache CloudStack 4.3.2 and the QEMU/KVM hypervisor version 0.12.1.2. The Apache CloudStack is a popular open source cloud infrastructure software suite, equipped with a comprehensive set of features to orchestrate VMs, networks, and storage. We have extended CloudStack to allow users to construct virtual clusters with direct access to an InfiniBand HCA without intervention from the hypervisor.

We basically provide two types of VM instance: a standard VM and an HPC VM. A standard VM has only one VCPU core. On the other hand, an HPC VM has 20 VCPU cores, 120 GB of memory, and Infiniband FDR HCA. Only one HPC VM is running on the physical node, because an HPC VM is not over-subscribed and dedicates the resources for sustain high performance. A virtual cluster consists of a set of HPC VMs and all VMs are connected through a full bisection InfiniBand network. In detail, a virtual cluster consists of a frontend node and compute nodes, as shown in Figure 3. The frontend node is an entry point for users to submit MPI jobs, which are then processed on the compute nodes. Typical HPC cluster software, including MPI library and compiler suite, TORQUE job scheduler, NFS, and NIS, are available on the cluster. Each virtual cluster is isolated by VLAN. The user also can dynamically increase and decrease the number of compute nodes according to the workload.

To manage such a virtual cluster easily, we have developed a virtual cluster construction utility suite, in the form of a set of command line utilities, called *sgc-tools*. Users can set up their own application environments based on it and instantly start their applications. Sgc-tools also enable users to dynamically scale in and scale out the virtual cluster, depending on their requirements. Users can create VM templates from their virtual clusters and take them to other clouds like Amazon EC2 to deploy clones.

The detail of sgc-tools is shown in Table 2. To create a virtual cluster, a user issues `sgc-cluster create` command with a virtual cluster configuration file as follows:

```
1 [frontend]
2 template = cluster_frontend-20141016.1
```


PoS(ISGC2015)004

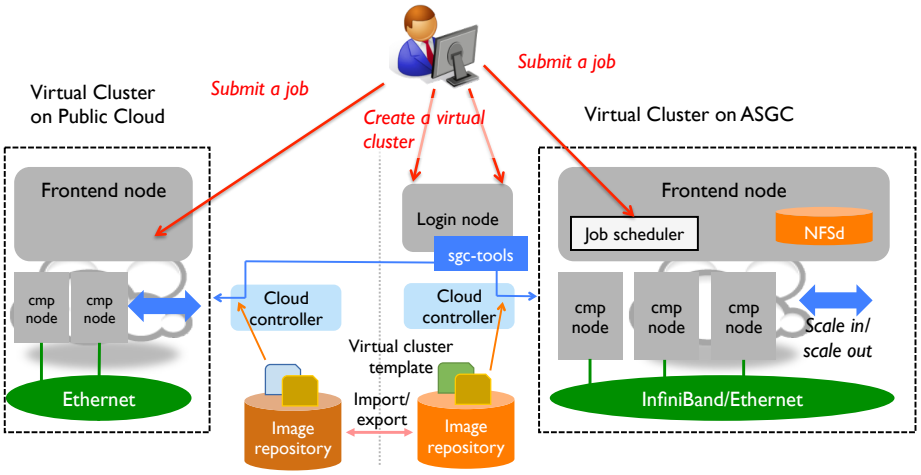


Figure 3: Elastic Virtual Cluster.

Table 2: Sgc-tools: Virtual Cluster Construction Tools

Command	Sub-command	Description
sgc-init	-	Initialization
sgc-vm	list|create|destroy|start|stop|show	Operations for virtual machines
sgc-cluster	list|create|destroy|start|stop|show|add-node|delete-node	Operations for virtual clusters
sgc-network	list|create|acquire	Operations for user networks
sgc-sshkey	list|register|delete	Registration of SSH public keys

```
3  compute_offering = HPC
4  disk_offering = Large
5
6  [cmpnode]
7  number = 16
8  template = cluster_compute-20141016.1
9  compute_offering = HPC
10 disk_offering = Large
11
12 [common]
13 name = myCluster
14 zone = zone01
15 network = myNetwork
16 sshkey = myKey
```

2.4 CloudStack and Storage Architecture

CloudStack provides two types of storage: primary storage and secondary storage. In CloudStack terms, a *cluster* is a set of physical machines running a hypervisor and each cluster has a dedicated *primary storage* where the virtual cluster instances are hosted. *Secondary storage* is used to store virtual machine templates, ISO images, and snapshots.

Unlike a typical design of CloudStack based systems, ASGC uses only local SSD storage for primary storage instead of a shared file system or storage, in order to become free from many

PoS(ISGC2015)004

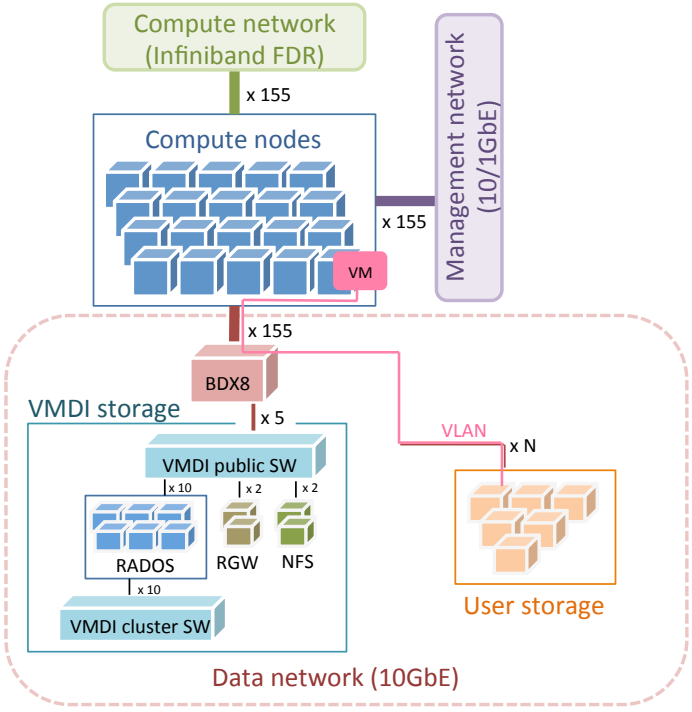


Figure 4: ASGC Network and the Storage system.

issues related to shared storage, such as costs, fairness among users, etc. The benefits are higher than support of VM live migration. ASGC provides the other two storage mechanisms: Virtual Machine Disk Image (VMDI) storage and user attached storage without relying on shared storage. The former is used for secondary storage. Regarding the latter, a user can bring their own storage to use from their virtual cluster, where the storage network is isolated from other users. This is because the capacity of local SSD is insufficient for some users. Figure 4 shows a diagram of the ASGC storage architecture. ASGC and the storage system connect with each other through a 10 Gigabit Ethernet data network. VMDI storage is based on a scale out object store, RADOS [3]. Although a RADOS gateway (RGW) provides the Amazon S3-compatible API, CloudStack does not directly access files on RGW/RADOS. Therefore, a file has to be staged from RADOS to a NFS secondary staging server before CloudStack accesses it. The effective capacity is about 48 TB (the physical capacity is 160 TB) due to the fact that RADOS keeps three replicas of each object.

3. Lessons Learned from the Operation of ASGC

3.1 CloudStack on a Supercomputer

We are operating an HPC Cloud service on a supercomputer, which is not designed for such a use case. For instance, ACE cluster management software from Cray strongly assumes a traditional HPC cluster platform. We installed CloudStack by using an ACE feature that mounts a local disk partition as read only, except for `/work` directory. This is not a problem on a traditional HPC platform, however, CloudStack has issues because it modifies some files under `/etc` and `/var/lib/libvirt`. We had to redirect access to such files by using bind mount and symbolic

PoS(ISGC2015)004

links to `/work` directory step by step. In addition, this makes software updates difficult. A well-designed provisioning tool for cloud middleware is strongly required.

We discovered and resolved several serious problems mainly related to CloudStack and we are improving it to achieve stable operation. For brevity, we are omitting many of the details, but a full list of the problems is provided in Appendix A. These issues mainly come from our storage architecture described in 2.4, that is, we use local SSD disk as primary storage, and S3-compatible RADOS object store as secondary storage. This situation may not be well tested in the community resulting in some obscure errors to be solved. For example, the DeploymentPlanner that determines which physical node to start a VM on, relies on VM migration. When multiple VMs are launched in parallel like `sgc-cluster`, DeploymentPlanner may assign VMs to the same physical node. VM migration allows a VM to move another physical node if the node is occupied, otherwise, the deployment fails. Generally speaking, however, DeploymentPlanner should not rely on VM migration. Now `sgc-cluster` inserts one second-sleep delay between subsequent requests for launching a VM to avoid this problem.

3.2 Cloud Services for HPC users

If many users require running a few regular applications, SaaS can be the best solution. The research fields of users in ASGC are quite broad, therefore SaaS is not realistic for us. On the other hand, IaaS is quite flexible. However, it is difficult to set up and maintain an HPC environment from scratch for application users. To bridge this gap, we provide sgc-tools on top of an IaaS service. We believe it works well, although some minor problems remain. For example, users cannot check exact availability of HPC VM in real-time.

To improve the ability to maintain VM templates, we are investigating the notion of "infrastructure as a code." Some useful configuration management and orchestration tools have been proposed to this end, including Packer, Chef, Terraform, and Amazon CloudFormation. We are currently considering introducing such technologies into our HPC Cloud.

3.3 Utilization

Currently, the utilization of ASGC is under 70%. Note that it is the ratio of the number of user-assigned CPU cores. A virtual cluster provides a dedicated environment for individual users where they can run their application whenever they want with no interference once it is created. This is an advantage over traditional supercomputing, however from the viewpoint of administrators, the precious resource is occupied whether the user fully utilizes it or not. This situation may be ignorable in hyper-scale cloud data centers like Amazon and Google. However, smaller-scale private cloud providers, such as the ASGC, cannot overlook such wasteful use of resources. Moreover, sgc-tools do not support queuing requests at system wide. Therefore, the users have to check the availability before they launch a virtual cluster. Introducing a global scheduler can be a solution for this problem. We are considering an approach that integrates a global scheduler and virtual clusters, like Condor VM Universe [7] and Vaccum model [8].

In terms of VM instance types, 95% of the total usage time is consumed for running HPC VM instances. This is another reason of low utilization. CloudStack launches system VMs, including a virtual router, secondary storage VM, and console VM, in addition to the user VMs. For example,

PoS(ISGC2015)004

37 system VMs are distributed to 9 nodes at a certain moment. If there is no standard VM, about 140 CPU cores remain idle. Server consolidation using VM migration is a possible solution for this problem, however, we cannot implement it for the several reasons. First of all, VM migration is not supported on our platform. Second, even if VM migration was supported, the effect of consolidation is limited because it depends on the number of clusters, where cluster is defined as in Section 2.4. ASGC is divided into ten clusters and a VM cannot migrate between clusters. When the operation is suspended, e.g., scheduled maintenance, CloudStack sequentially turns a node to the maintenance mode at each cluster. In the case of a single cluster, the state transition takes quite long time. We expected standard VMs fill such CPU slots, however, the demand for standard VMs is lower than we expected. We should reinvestigate user demands and/or obtain new users for a standard VM, for example, web server.

4. Preliminary Experiment on a Virtual Cluster with SR-IOV

4.1 Experimental Setting

The following software settings are identical for both a physical cluster and virtual clusters. The Operating System installed is CentOS version 6.5. Each node has the Intel Cluster Studio version SP1 1.2.144, which includes compiler suite, MPI runtime, and math kernel library, the Mellanox OpenFabrics Enterprise Distribution (MLNX_OFED) version 2.1 installed. The firmware version of ConnectX-3 is 2.30.8000 and supports an SR-IOV feature.

We enabled a VCPU pinning feature, which statically binds a virtual CPU core to a physical CPU core. On a NUMA system like the latest x86-64 architecture, memory affinity is an important concern in enhancing the performance of HPC applications. It can be more effective if a process is pinned to a CPU core in such a way that its memory allocations are always local to the CPU socket it is running on. This avoids inter-socket memory transfer, which has less bandwidth and can significantly degrade performance.

4.2 Deployment Time of Virtual Cluster

Figure 5 shows the deployment time of a virtual cluster by using sgc-tools, where the number of nodes varied from 1 to 32. A virtual cluster requires two VM templates: one for a frontend node and another for a compute node. In this experiment, the file size of a frontend node and a compute node are 18.3 and 13.7 GiB, respectively. The deployment time depends on the cache state of the VM templates and the state is classified in one of the following three categories: 1) “Cache on local node” denotes that the cache already exists on a local node, therefore, no network traffic is involved during deployment; 2) “Cache on NFS/SS” denotes that the cache exists on the staging server (SS), thus VM templates are transferred from SS to local nodes at the deployment and the traffic volume increases proportionally as the number of nodes increases; and 3) “No cache” denotes there is no cache anywhere and the deployment time includes “cache on NFS/SS” in addition to the time for transferring VM templates from RADOS to SS.

The time (a) for transferring from RADOS to SS is almost constant, that is, about 220 seconds. The time (b) for transferring from SS to local nodes linearly increases as the number of nodes increases. The time (c) for deploying from cache on local nodes is almost constant, independent of

PoS(ISGC2015)004

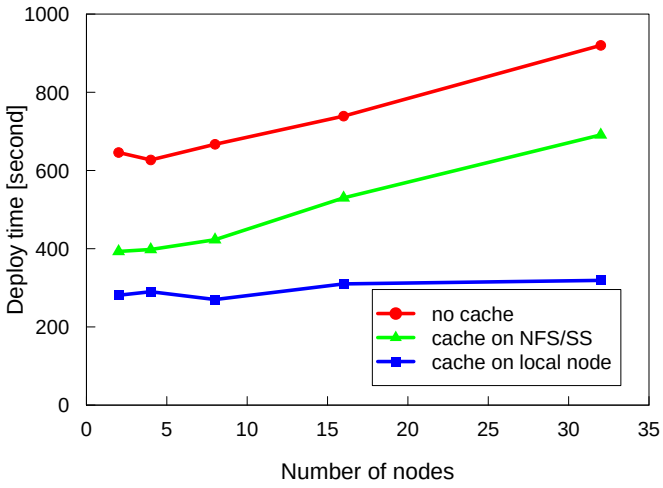


Figure 5: Deployment time of a virtual cluster.

the number of nodes. The times (a) and (b) fluctuate due to background traffic caused by sharing 10 Gigabit Ethernet with other users. We can reduce the deployment time by performance tuning of a secondary storage VM and trunking of the link between ASGC and user-attached storage. The time (c) divides into PCI device attach time (90 seconds), OS boot time (90 seconds), and file system creation `mkfs` time at only the first boot time (90 seconds). In PCI device attach processing before OS boot, QEMU issues `ioctl(KVM_ASSIGN_PCI_DEVICE)` and KVM does not respond for such a long time. We believe the latest KVM fixes this issue.

4.3 MPI Communication Performance

The performance of basic MPI operations using Intel Micro Benchmark (IMB) version 3.2.4 [9] was measured. We have compared the performance among the following three configurations: physical cluster (BM), a virtual cluster with PCI passthrough (PCI passthrough), and a virtual cluster with SR-IOV (SR-IOV). In this experiment, we used a 16 node-cluster.

Figure 6 presents MPI point-to-point bandwidth. The message size varies from 1 byte to 4 mega bytes. The virtualization overhead decreases as the message size increases. With small message size, the performance is reduced by up to 30%. Conversely, with large message size, the performance degradation is reduced less than 5%. Comparing SR-IOV with PCI passthrough, although the performance of SR-IOV is slightly less than that of PCI passthrough with short messages, both configurations yield results are almost comparable.

Figure 7 presents the performance of several MPI collective operations, including Allgather, Allreduce, Alltoall, Bcast, and Reduce. These results are almost identical with Pingpong, however, we did observe some unexpected performance degradations in SR-IOV. For example, both the 100 KB message in Allgather and 256 KB message in Alltoall suffered large, reproducible reductions in performance and the cause of this is currently under investigation. In addition, we measured the performance of Barrier and found that the execution time of BM, PCI passthrough, and SR-IOV were 6.87, 8.07, and 9.36 microseconds, respectively. In other words, the virtualization overhead of PCI passthrough and SR-IOV are 17% and 36%, respectively.

PoS(ISGC2015)004

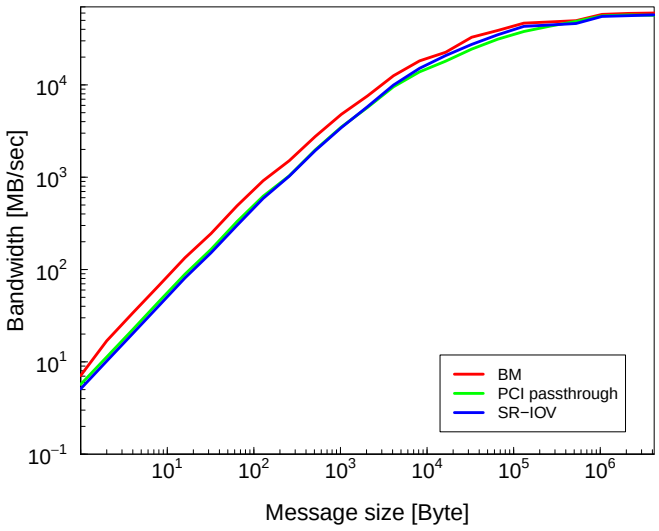


Figure 6: MPI point-to-point bandwidth.

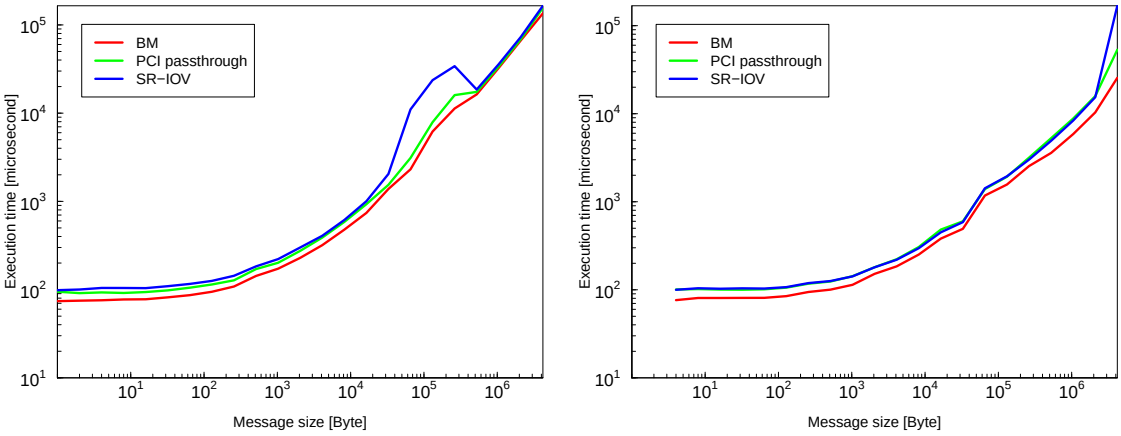
4.4 LAMMPS

LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) is a classical molecular dynamics simulator developed by Sandia National Laboratory [10]. We used an embedded atom method (EAM) metallic solid benchmark included in the package. Figure 8 shows the execution times for 1M atom-problem runs with different numbers of processes. We repeated this ten times and the average execution time with standard deviation is plotted. The difference between PCI passthrough and SR-IOV is negligible, and the virtualization overhead is about 13% without relying on the number of nodes. In this experiment, we also confirmed the effect of VCPU pinning on a virtual cluster. On both PCI passthrough and SR-IOV, the performance fluctuations are reduced if VCPU pinning is enabled. This can be considerable because VCPU scheduling on the host OS has a negative impact on the application behavior on a virtual cluster.

5. Conclusions and Future Work

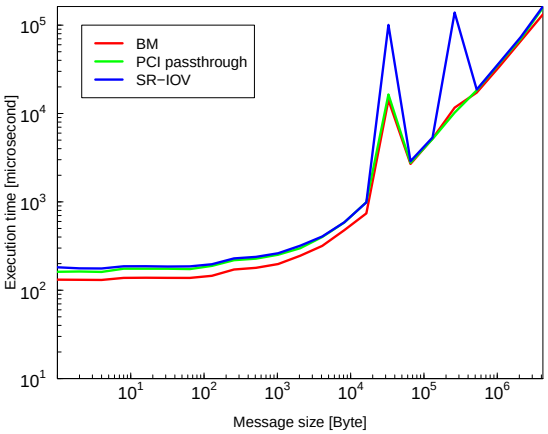
99% of HPC jobs running on NSF supercomputing centers fit into one rack [11]. On such a scale, HPC Cloud is feasible enough because the overhead of virtualization can be dramatically reduced with the help of both hardware level and system software level virtualization mechanisms that are mature enough for practical use. In the future, HPC Clouds are heading toward more hybrid-cloud and multi-cloud systems, where the user can execute their application anytime and anywhere he/she wants. Our vision "build once, run everywhere" is shared with the PRAGMA community [12] and both ASGC, which is a fully virtualized HPC system, and PRAGMA Cloud, which is an inter-cloud computing testbed, are two approaches toward the same goal. This paper presents our lesson learned from our HPC service and preliminary experimental results using SR-IOV. We can launch a highly productive system in a short development time by leveraging state-of-the-art open source system software. Although SR-IOV is a useful I/O virtualization technology

PoS(ISGC2015)004

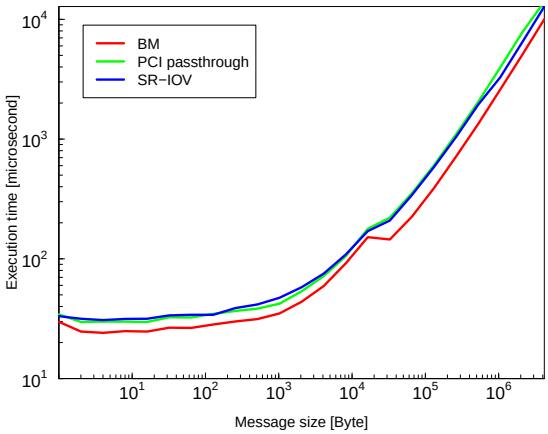


(a) Allgather

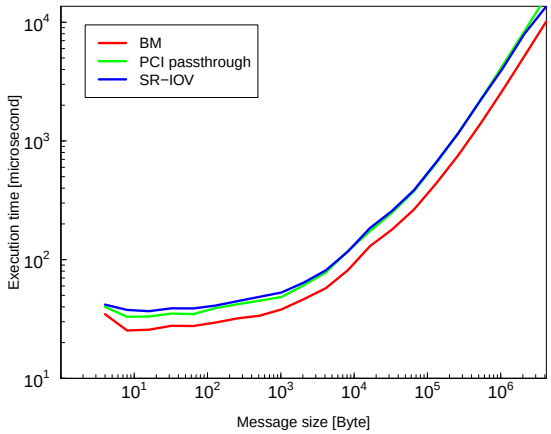
(b) Allreduce



(c) Alltoall



(d) Bcast



(e) Reduce

Figure 7: MPI Collective operations.

PoS(ISGC2015)004

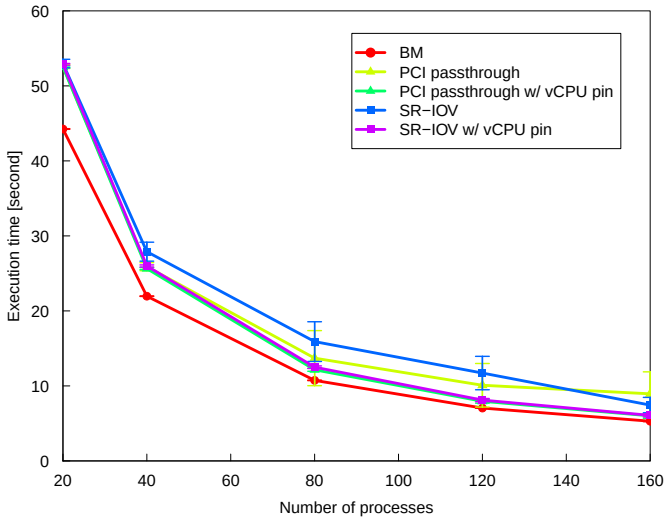


Figure 8: Performance comparison of LAMMPS.

to improve server utilization, the software stack is not mature enough at this time. On a multi-cloud environment, data movement is key and in keeping with our vision, we plan to investigate efficient data management and transfer methods to improve multi-cloud environment performance, in addition to federated identity management.

A. Issue List

We have faced several issues related to the Apache CloudStack, as shown in Table 3. The issue number is corresponding to the number on the CloudStack JIRA [1]. The details of each item are found in this site. We have reported some of them and our patches were accepted to the main trunk of the Apache CloudStack.

References

[1] Apache CloudStack. [Online]. Available: `http://cloudstack.apache.org/`

[2] A. Kivity, Y. Kamay, D. Laor, U. Lublin, *KVM: the Linux Virtual Machine Monitor*, in proceedings of *the Linux Symposium*, Vol. 1, pp. 225–230, 2007.

[3] A. S. Weil, A. W. Leung, S. A. Brandt, and C. Maltzahn, *RAODS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters*, in proceedings of *International Workshop on Petascale Data Storage*, pp. 35–44, 2007.

[4] Zabbix Monitoring System. [Online]. Available: `http://www.zabbix.org`

[5] P. Pornkitprasan, V. Visoottiviseth, and R. Takano, *Engaging Hardware-Virtualized Network Devices in Cloud Data Centers*, in proceedings of *ICT International Student Project Conference (ICT-ISPC2014)*, pp. 1–4, 2014.

[1] https://issues.apache.org/jira

PoS(ISGC2015)004

Table 3: Issue List related to the Apache CloudStack on ASGC.

Issue Number	Title	Status
	Action	
7951	cloudstack-agent jsvc gets too large virtual memory space	Fixed
	Our patch is merged to the main trunk.	
6472	listUsageRecords generates NullPointerExceptions for expunging instances	Fixed
	Our patch is merged to the main trunk.	
2625, 2401	Duplicate usage records when listing large number of records / Small page sizes return duplicate results	Fixed
	Backporting is done.	
6869	Public key content is overridden by template's meta data when you create a instance	Fixed
	We reported this bug, and backporting is done.	
6810	Migration of a VM with volumes in local storage to another host in the same cluster is failing	Fixed
	Backporting is done.	
6236	Negative ref_cnt of template(snapshot/volume)_store_ref results in out-of-range error in MySQL	Fixed
	An existing patch does not fix this bug. We applied another work around patch.	
7539	[S3] Parallel deployment makes reference count of a cache in NFS secondary staging store negative(-1) (JIRA #7539)	Unresolved
	Our patch is pending.	
7412	Can't create proper template from VM on S3 secondary storage environment	Fixed
	Our patch is merged to the main trunk.	
8085	Fails to attach a volume (is made from a snapshot) to a VM with using local storage as primary storage	Unresolved
	We reported this bug.	

[6] N. Chakthranont, P. Khunphet, R. Takano, and T. Ikegami, *Exploring the Performance Impact of Virtualization on an HPC Cloud*, in proceedings of *IEEE International Conference on Cloud Computing Technology and Science (CloudCom2014)*, pp. 426–432, 2014.

[7] The Condor Team, *Condor Manual: Virtual Machine Applications*. [Online]. Available: `http://www.cs.wisc.edu/condor/manual`

[8] A. McNab, F. Stagni, and M. U. Garcia, *Running Jobs in the Vacuum*, in proceedings of *20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013)*, 2014.

[9] Intel Micro Benchmark. [Online]. Available: `http://software.intel.com/en-us/articles/intel-mpi-benchmarks`

[10] LAMMPS Molecular Dynamics Simulator. [Online]. Available: `http://lammps.sandia.gov/`

[11] M. Norman, *Data-intensive HPC at SDSC*, Keynote, ON*VECTOR2015, 2015.

PoS(ISGC2015)004

[12] Y. Tanaka, N. Yamamoto, R. Takano, A. Ota, P. Papadopoulos, N. Williams, C. Zheng, W. Huang, Y. Pan, C. Wu, H. Yu, J.H. S. Shiao, K. Ichikawa, T. Tada, S. Date, S. Shimojo, *Building Secure and Transparent Inter-Cloud Infrastructure for Scientific Applications*, *Advances in Parallel Computing*, vol. 23, pp.35-52, 2013.

PoS(ISGC2015)004