World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:5, No:6, 2011

Formosa3: A Cloud-Enabled HPC Cluster in NCHC

Chin-Hung Li, Te-Ming Chen, Ying-Chuan Chen, and Shuen-Tai Wang

***Abstract*—**This paper proposes a new approach to offer a private cloud service in HPC clusters. In particular, our approach relies on automatically scheduling users' customized environment request as a normal job in batch system. After finishing virtualization request jobs, those guest operating systems will dismiss so that compute nodes will be released again for computing. We present initial work on the innovative integration of HPC batch system and virtualization tools that aims at coexistence such that they suffice for meeting the minimizing interference required by a traditional HPC cluster. Given the design of initial infrastructure, the proposed effort has the potential to positively impact on synergy model. The results from the experiment concluded that goal for provisioning customized cluster environment indeed can be fulfilled by using virtual machines, and efficiency can be improved with proper setup and arrangements.

***Keywords*—**Cloud Computing, HPC Cluster, Private Cloud, Virtualization

I. Introduction

IN recent years there has been renewal of interest in the relation between virtualization and cloud computing. A growing number of companies try to propose new solutions for users to exploit the computing power in the cloud. Although cloud computing is a buzz word, the computing power of HPC cluster should not be ignored as well.

As Taiwan's only full service HPC service provider, we continue to insist on innovation. By planning and executing the construction of advanced researching environment, the NCHC has dutifully provided HPC services to academia. We also have noticed that the overall computing capacity still falls far behind expectations. Shortage of allocation of advanced machines becomes our major challenge of the year. Experience is that lots of complaints concerning long waiting time before execution. One common feedback is that users need their projects done in a specific platform, but they always got stuck with the currently OS version of supercomputer systems.

Supercomputer systems are not affordable for most academic institutes and colleges in Taiwan. The other problem was that most researchers are not familiar with software stack on HPC systems, not to mention installing them alone. Our academic users always look forward to better facility we could offer. They hope for more OS versions and distributions for them to choose from. Fortunately, virtualization technology brings new meaning to the HPC cluster. In the past, if we installed dedicated Linux distribution on a cluster, there was no reason for system administrator to change the cluster OS except software or kernel upgrade. Now, with the virtualization of worker nodes, system administrators can provide any OS on virtual machines that user needed. In order to meet this kind of requirement, we try to propose a mechanism which could offer a rental-like service for outreach programs.

The main stream as supercomputing facilities is using distributive cluster systems. According to cluster architecture, one typical cluster usually includes login, management, storage and compute nodes. Users are allowed to login and submit jobs from login nodes, and storage servers provide central and accessible working space mounted for all nodes. And, of course, many identical compute nodes act as workhorses solving problems in parallel.

In this paper we propose a flexible approach to utilize traditional HPC clusters. For smartly scheduling compute nodes, new features should be added to cluster batch system. Integrating extra virtual machine management software is also inevitable. This paper explains how we customized the commodity cluster infrastructure and deployed middleware on it to offer dual-purpose function for cloud and HPC.

II. Background

A. Cloud Computing

When it comes to cloud computing, Amazon EC2 [1] might be the most famous service you've heard of. EC2 is a platform offering huge amount of computing power. The computing power of EC2 is resizable. Via AWS web interface, EC2 users can change, increase or decrease the demand of the computing resource at any time. Besides, they can control and manage their own virtual instances so that users can greatly save their time to buy and build physical IT infrastructure. All their projects can be fulfilled in Amazon's mature virtual platform. The on-demand virtual machines concept brought by Amazon EC2 will benefit those who are not affordable for expensive machines [2]. EC2 pioneered a sea change and foster Infrastructure as a Service innovation. Based on this successful model, we try to learn and figure out what else we can do in the high performance computing cluster [3].

B. NCHC Formosa3 HPC Cluster

Formosa3 HPC Cluster is a 64-bit Beowulf cluster located in the Southern Business Unit of NCHC. It consists of 76 IBM System x3550 M3 servers as its compute nodes. This self-made

C. H. Li is with the National Center for High-Performance Computing, Tainan 74147 Taiwan (phone: 886-6-505-0940; fax: 886-6-505-5909; e-mail: OscarLi@nchc.narl.org.tw).

T. M. Chen is with the National Center for High-Performance Computing, Tainan 74147 Taiwan (phone: 886-6-505-0940; fax: 886-6-505-5909; e-mail: gavin@nchc.narl.org.tw).

Y. C. Chen is with the National Center for High-Performance Computing, Tainan 74147 Taiwan (phone: 886-6-505-0940; fax: 886-6-505-5909; e-mail: ycc0301@nchc.narl.org.tw).

S. T. Wang is with the National Center for High-Performance Computing, Tainan 74147 Taiwan (phone: 886-6-505-0940; fax: 886-6-505-5909; e-mail: stwang@nchc.narl.org.tw).

cluster was designed and constructed by the 'HPC Cluster Team' at NCHC for computational science applications and came online in 2011. Each node has two six-core Intel model processors and 48GB of DDR 3 registered ECC SDRAM. All nodes are connected by the 4x QDR InfiniBand high speed network and a private subnet with Gigabit Ethernet. An additional 4 nodes are arranged as login nodes, thus, enabling users' easy access. Its 4 DAS storage servers are responsible for the parallel file-system. The major difference from the previous series clusters is the virtualization support. In Formosa3's hardware specifications, we have added new requirements in processors, motherboards and network cards for cluster compute nodes. Theoretically, native virtualization capability provided by this cluster is shown in Table I.

TABLE I
VIRTUALIZATION SUPPORT OF FORMOSA3 CLUSTER

Component	Model	Virtualization Technology
CPU	Intel Xeon X5660	Intel VT-x
IOH	Intel 5520	Intel VT-d 1. DMA remapping 2. Address translation 3. Interrupt remapping
InfiniBand HCA	Voltaire 700Ex2-Q-1	1. SR-IOV 2. Address translation and protection 3. Dedicate adapter resources 4. Multiple queue per virtual machine 5. Enhanced Qos for vNICs

The first one, the CPU part, it must support hardware-assisted virtualization instruction set extensions. This means that your processor must be Intel VT type or AMD-V type. V stands for virtualization support. The second component, north bridge chipset on motherboard, it must support PCI passthrough—PCI devices performance improvement for virtualized environment. The third one, InfiniBand HCA, it must support the new industry standard, SR-IOV (Single-Root I/O Virtualization) [4]. A virtualization solution is never just one piece. These three parts of virtualization are what we focused on first.

There three ways, full-virtualization, para-virtualization and container-based, for virtual machine in current Linux world. According to the most I/O benchmark results, performance of fully-virtualization is far behind the para-virtualization. The main reason is modified guest operating systems comprise a hypervisor-aware driver in para-virtualization mode. Para-virtualization reduced the degradation of system performance. But only using full-virtualization can fulfill the goal for offering various operating systems on single machine. The most crucial factor of development for this cloud platform is comprehensive support of virtualization in hardware level.

Because virtualization will not work on all regular processors, CPU and related chipsets vendors began supporting virtualization technology in 2006. In Intel side, we saw the VT technologies while AMD offered AMD-V. Their detailed technology may be far different, but their goal was the same to accelerate performance of binary translation for hardware.

If the motherboard supports Intel VT-d or AMD IOMMU and you've enabled it in BIOS, it will allow the host PCI device to be directly accessed by VM as if it was physically attached to the Guest OS. On top of that, more and more HPC clusters on TOP500 List have used InfinBand as inter-connect. Formosa3 has also adopted the 4x QDR HCA so that it could offer very high bandwidth private networking. In this cluster, InfiniBand network is responsible for both MPI and parallel file-system communication. As for cloud service, its Mellanox ConnectX-2 chipset of InfiniBand HCA supports the SR-IOV technology. With SR-IOV, the HCA on a host node can be efficiently multiplexed by many Guest OSes of VMs at the same time.

C.KVM Hypervisor

KVM stands for Kernel-based Virtual Machine [5] which means a full virtualization solution. KVM is for Linux which is working on processors that have capabilities related to virtualization. KVM virtual machine implementation has two loadable kernel modules. One is "kvm.ko" that handles the main virtualization infrastructure and another is the processor specific module, "kvm_intel.ko" or "kvm_amd.ko". In addition, it also needs a modified QEMU to emulate the peripherals for virtual machines.

Linux KVM is open-source software. But to manage your own virtual machine well you will need to add up some special software to it. The main requirement on host is the libvirt library. Libvirt provides a hypervisor-agnostic and comprehensive APIs to manage Guest OSes running on a real host. Fig. 1 presents the components of a typical KVM installation. As you can see that each KVM VM can be treated as a user space process in Linux. This will be helpful for streamlining VM builds within batch system job script if we choose KVM as our hypervisor.

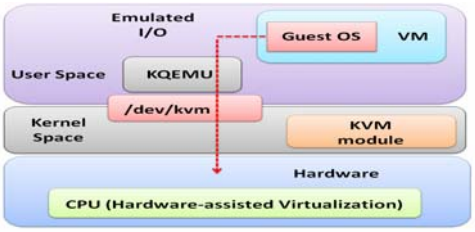

Fig. 1 Linux KVM architecture

D.Torque Resource Manager

Torque [6] is an open source resource manager providing control over batch jobs and distributed compute nodes. To guarantee the fairness of using compute nodes, priority escalading, or resource partitioning, installing a scheduling system is necessary for a large cluster open to many users and organizations. Due to ease of installation, Torque has predominated in batch systems for typical HPC clusters, especially in research institutions. Not only Torque can detect the compute nodes' availability but leverage processes on single node from being over-occupied. There are six Queues on Formosa3, and the detail setting of each Queue was illustrated

in Table II. For better utilization of resources, the backfill algorithm of these Queues is arranged as Best-Fit.

TABLE II
TORQUE QUEUE CONFIGURATIONS ON FORMOSA3

Queue Name	Maximum Nodes	Maximum Wall-Time
Serial	1	168 hours
N4	4	120 hours
N8	8	120 hours
N16	16	96 hours
N32	32	48 hours
N64	64	48 hours

E. Virtualization and HPC

Table III and Table IV list the middleware installed on Formosa3 cluster. Generally speaking, HPC applications have historically tended to exhaust the limits of CPU performance, memory capacity and network bandwidth. On average, MPI job processes optimize over 80 percent workloads on single compute node. Server Virtualization is not fancy technology, but in HPC world is the only one place where virtualization hasn't taken off. HPC consumers are always looking for best performance. Therefore, cluster designers opt to avoid server virtualization due to the overhead that virtualization imposes. However, for HPC sites, cloud computing concept is still attractive with dynamic resource provisioning and live migration capabilities.

TABLE III
HPC MIDDLEWARE OF FORMOSA3

Operating System	CentOS 5.5 x86_64
Message Passing Library	MPICH2, MVAPICH2
Batch System	Torque, maui
Remote Operation	PDSH
Monitoring	Ganglia

TABLE IV
CLOUD MIDDLEWARE OF FORMOSA3

Operating System	CentOS 5.5 x86_64
Virtualization Library	Libvirt
Hypervisor	KVM
Resource Management	OpenNebula
Remote Operation	Virtual Shell
Web Portal	eyeOS, Web Console

III. IMPLEMENTATION

Most researches treat VMs as new resource for clusters or grid environment [7]. They voluntarily neglected the VM overhead while processing jobs on the same compute node at the same time. Coexistence of virtualization and HPC service could be a disaster for less powerful processors because there is no room for virtualization. Fig. 2 shows the distinct concept about how we use virtual machines in HPC batch system. We design a new mechanism for VM allocation inside Torque system. And how we integrate OpenNebula [8]–[9] for managing VMs and VM images will be described in detail as follows.

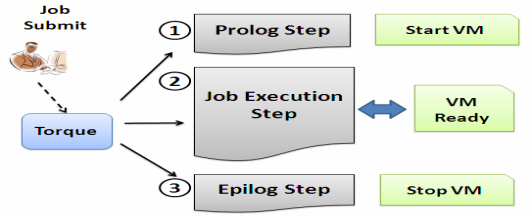

Fig. 2 VM life cycle in batch system

A. Torque Customization

In Torque system, host CPU and host memory are main resources for HPC clusters. But, in virtualization's point of view, the computing resources were extended to software level. These different VM operating systems will be new resource for the batch system. But if we want a virtual machine to attain evenly matched computing power as a physical one, we have to fairly schedule HPC jobs and VM allocation requests at the same time. Unlike other projects [10]–[12] trying to increase job capacity by scheduling VM in the cloud, the rule of thumb in our work is that those nodes processing HPC jobs should not be available for VM allocations, and vice versa. Once a node was dispatched for job execution, no VM will be created on it any more.

The batch system is the most import part for this project. We retained the default interface for user job submission, but just focused on the prolog and the epilog job processing steps of Torque client daemon on compute nodes. We add the extra environment variables to job scripts when cloud users send out the VM requests. These variables used to indicate the path of VM image file, template file and script files for customization, etc.

When parsing the job scripts, the Torque client daemon, pbs_mon, will automatically decide whether or not it has to start the virtual machine. Fig. 2 indicates the VM life cycle and its relation with Torque operation. These VM Guest OSes will be staged during the prolog step and filed away during the epilog step. These Guest OSes will keep up and running until jobs use up their Queue's maximum wall-time.

B. OpenNebula Integration

Installing Xen or KVM in a machine is not that difficult, but working out how to fit different requirements into virtual environment in a cluster could be a challenge [13]. With OpenNebula, it makes the manipulation of multiple VMs on cluster environment easier. OpenNebula contains a set of commands you can apply to VMs, physical hosts, and even virtual network, for example, which are not for the faint of heart in virsh tools. The following are a few examples of host and VM management in OpenNebula.

```
$ onevnet  create private.net
$ onehost add node1 im_kvm vmm_kvm tm_nfs
$ onevm create centos5.one
$ onevm deploy centos5 node1
```

Fig. 3 ONE command examples

One of OpenNebula's strongest points is its ability to monitor hosts, VMs and network in cluster by issuing "one commands" from console. Use the above example feeding the VM profile data into host pool, and you can easily deploy, migrate any VM among cluster nodes.

For practical reasons, imaged should be classified. There are some commands will be helpful when administrators need to do VM image management (to add/delete image, clone image, etc.). The following are examples of image file management in latest OpenNebula.

```
$ oneimage register centos5.img
$ oneimage publish
$ onevm saves
```

Fig. 4 ONE image manipulation examples

C. Security Control

While using the Linux server for internet services, in order to avoid the extra legal problems during operation, sometimes periodically checking the system security is necessary. There are already a lot of security tools from open-source for Linux platform. Before beginning Formosa3's cloud service, we should also take security issues into consideration. There is a lot of possibility that the OS of VM will be tainted after being hacked from outside or changed by some mischievous users [14].

As shown in Fig. 5, in order to detect the OS integrity on VM, we tried to apply effective security tools on them. We have developed the two kinds of mechanisms for intrusion detecting. Firstly, before preparing the VM images for users, we have adopted the host intrusion systems on them in advance. AIDE is what we used for gathering the whole original system files' key fingerprint. Next, when VM woken up for user request, AIDE intrusion systems will alert system administrator to possible intrusion attempts by periodically checking system files during VM's running cycle. Once it detected target file changed, each VM will pipe out AIDE comparison log data to cluster file system and mail to the administrator. This is real-time alerting approach for system administration. The second way we used is passive detection. Each VM image will be mounted (via lomount command) again as a chroot-like partition after VM shutdown and then the Formosa3 system will apply rootkit detection tools (e.g., chkrootkit or rkhunter) hunting for any malware. Both methods will greatly enhance the system security.

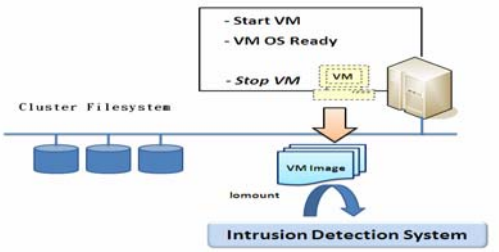

Fig. 5 Security audit for virtual machine instances

D. Deployment and Operation

As of this writing, both IBM GPFS and Lustre parallel file-system [15] now support InfiniBand native mode networking. In Formosa3, we have deployed both Lustre and GPFS as global shared file-systems at the same time. Especially for large files, GPFS brings the best data I/O rate for jobs. Therefore, we considered that GPFS will play well too while accessing the VM images.

The upper part of Fig. 6 is the scenario of HPC service. It sketches the workflows of HPC services in Formosa3. Users login, submit their jobs and get the output after job scheduled and completed.

The cloud service is illustrated in the bottom part of Fig. 6. Cloud users start the web browser and access the Formosa3 web site. This web portal contains several contextualization options that users could select for their virtual environment. In Table V, currently available customization options will be platform type, Linux distribution and target service. Then, cloud users simply choose the dedicated OS version, and fill in how many nodes needed for running their projects. In addition, the target service will help install all necessary packages for the mpich, mvapich or hadoop environment without users' intervention. If the cloud user did not specify it, there will be a raw operating system installed only.

TABLE V
CUSTOMIZATION OPTIONS FOR CLOUD SERVICE

	Options
Platform Category	Cluster (Head Node and Slave Nodes), Single Node
Operating System	CentOS, Fedora, Ubuntu
Target Service	Hadoop, Mvapich (ib), Mpich (tcp)

All requests from Formosa3 web portal will be transformed into formatted Torque job scripts and automatically submitted, and scheduled as usual. When there are free nodes for jobs, Torque client daemon will execute the OpenNebula command to wake up the dedicated OS-type VM image. The default application packages, VM images and related VM template files are stored on GPFS file-system. Each virtual cluster has two kinds of nodes. The first one is head node, and it is the only one node for each request. The rest nodes are all compute nodes.

Cloud head node served as the login node for users. All the necessary customization procedures will be done during VM operating system booting stage. Basically, its tasks includes

adding the user account, exporting the /home and /opt for virtual cluster clients, starting the NIS server daemons, executing the contextualization scripts that user chose from web portal.

The number of cloud compute node is determined by Torque queue. When one requested 4 virtual compute node from web portal, his corresponding Torque job script will try to allocate 4 free real nodes. If enough nodes were scheduled by Torque system, Guest OS will be created by "one command". Just like the cloud head node, all cloud compute nodes will automatically start NIS client service and mount the cloud head node's /opt and /home via NFS protocol. These cloud compute nodes simply served as workhorses, therefore, no contextualization will apply on them.

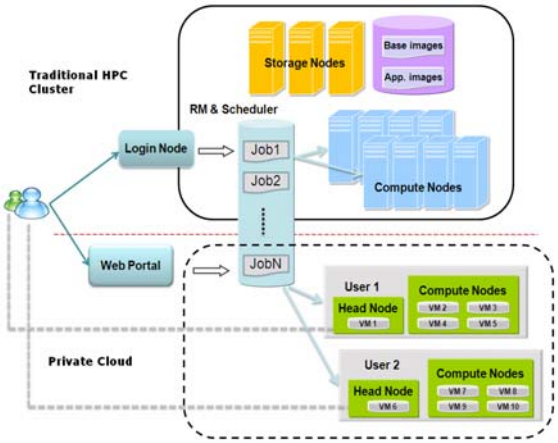

Fig. 6 Scenario of job and VM allocation

These cloud user's dedicated nodes will be in the same subnetwork and all compute nodes will mount the /home and /opt of the head node during OS booting, too. Thus, these VMs finally together form a private cloud. While the virtual environment being ready, user on portal will be prompted a web-based SSH client console page. At last, he or she could freely access this isolated virtual cluster until time lapsed.

These private cloud VMs all are using private network but users are from internet. On the web portal server, we have setup a database counting each user's login time and metering the amount of resource they used. Besides, this database deals with SSH port forwarding and mapping to cloud head nodes.

IV. Experiment Results and Analysis

A. Performance Evaluation of Cloud Service

Although virtualization brings flexibility to OS arrangement, its drawback in performance cannot be ignored [16]–[17]. To understand the impact, we deployed the mechanism on Formosa3 cluster, and ran real world MPI projects on KVM virtual machines.

HPL (High Performance LINPACK) is to solve a dense linear system problem and is the widely used benchmark for evaluating performance of HPC systems. However, the performance of PC cluster is largely application-dependent. We use the NCHC benchmark suit which contains five problems, namely hubksp, nonh3d, bem3d, ns3d and jcg3d, picked from four application domains. The hubksp program comes from physics, and nonh3d is an atmospheric science case. Both bem3d and ns3d are applications of parallel computing in the field of computational fluid dynamics (CFD). The last one, jcg3d is from computational solid mechanics. These particular jobs helped us to distinguish the performance in different problem domains.

B. Performance Analysis

TABLE VI
Sequential Job Performance

Single Core	Physical Node	KVM Node
Bem3d	real 36m36.244s	real 72m0.243s
	user 36m34.750s	user 69m46.451s
	sys 0m0.087s	sys 0m2.829s
Hubksp	real 40m6.238s	real 49m20.904s
	user 40m2.452s	user 48m24.529s
	sys 0m3.213s	sys 0m4.448s
Ns3d	real 39m19.016s	real 45m28.550s
	user 38m10.482s	user 44m25.962s
	sys 0m0.320s	sys 0m3.390s
Nonh3d	real 58m35.250s	real 78m35.344s
	user 56m28.103s	user 74m4.236s
	sys 0m0.904s	sys 0m5.023s
Jcg3d	real 82m43.054s	real 108m14.257s
	user 81m39.085s	user 107m4.393s
	sys 0m0.502s	sys 0m6.140s

TABLE VII
Parallel Job Performance

32 Cores over Gigabit Ethernet	Physical Cluster	KVM Cluster
Hubksp	real 16m11.841s	real 53m2.332s
	user 8m34.624s	user 11m34.943s
	sys 0m32.682s	sys 8m42.274s
Nonh3d	real 42m59.208s	real 66m23.284s
	user 12m28.441s	user 14m16.380s
	sys 1m3.038s	sys 5m45.228s

From sequential jobs' elapsed-time data listed in Table VI, apparently performance results of physical node excels the virtual one while running single core. In terms of parallel processing, we evaluate hubksp and nonh3d jobs over 32 cores. Table VII shows the performance comparison between physical and KVM virtual cluster over Gigabit Ethernet.

It is a comfort to see that the performance drop become not explicit between physical and virtual environment over single core evaluation. According to Table VI, virtual node shows slight performance drop on CPU-intensive cases. On the contrary, performance drops are obvious on parallel hubksp and nonh3d. It is because that both hubskp and nonh3d jobs will output files inside Guest OS during job processing. The other possible cause is poor network I/O rate over emulated Ethernet by KVM.

TABLE VIII
Performance Comparison

24 Cores	Network Interface	Rmax (Gflops)	Efficiency (%)	NetPIPE (Gbps)

Physical Cluster	GbE	152.2	56.62	0.74
	IPoIB	199.5	74.21	4
	Native IB	232.4	86.45	24
KVM Cluster	GbE	84.3	31.36	0.17
	IPoIB	157.8	58.70	4
	Native IB	164.4	61.16	23

Rpeak: 268.8 Gflops

Table VIII are HPL and NetPIPE benchmark results of physical and virtual environment. The testbed here is 24 cores on two Formosa3 compute nodes. According the NetPIPE result, it reveals there is a huge performance drop in communication between KVM VMs hosted by distinct physical nodes over Gigabit Ethernet. The disk and network I/O devices emulated by KQEMU are not capable of high performance computing.

As for the InfiniBand network channel, that is another story. The NetPIPE performance drop of virtual cluster over InfiniBand is much smaller than that over Gigabit Ethernet. These results do prove that PCI passthrough technology really benefits classical HPC applications on virtual machine environment. Unfortunately, the overall LINPACK performance of virtual cluster over InfiniBand native mode contributes 61% efficiency, which means there is still much room for improvement of KVM cluster.

C. VM Allocation Time Analysis

As most research revealed [18]–[19], virtualization does not play well with high performance I/O. In this section, we try to evaluate the suitability of file-system for deploying virtual clusters inside the Formosa3. To check out the possible factor interfering VM allocation speed [20], we put file-backed VMs on different directories for tests. The Formosa3 plans to offer three sorts of cluster-wide file-system for our users. The /home and /opt directories are shared via NFS and compute nodes mount them over Ethernet interface. The two public working space, /work was built by Lustre package, and /gpfswan is from IBM GPFS. The default KVM image repository is /var/lib/libvirt/images where is the local disk partition formatted to ext3 on compute nodes. The Linux-based VM image file size we prepared here is about 4GB. We submitted different cluster scale cases to Torque, and counted the elapsed time spent from submission to private cloud being available. Below, the Fig. 7 shows the average allocation-time of each cluster scale on different file-system.

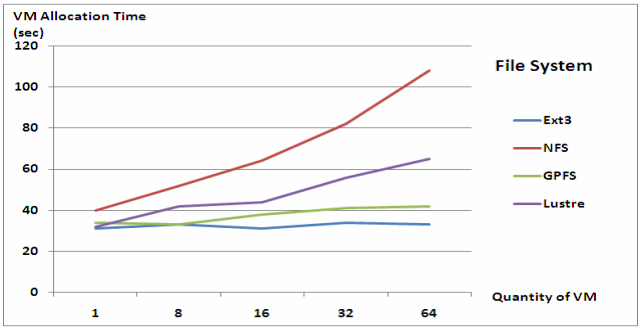

Fig. 7 Average VM allocation time on different file systems

From Fig. 7, it seeks to capture the fact that users have to wait for a longer time till virtual cluster being ready. The allocation-time for a virtual cluster keeps increasing when more VM images demanded. Cloud users need to be more patient when they requested more virtual machines.

From the allocation-time test results, it proves that putting VM images on parallel file-system could bring noticeable effect on allocation rate. According to Fig. 7, housing VM image files on parallel file-system and local disk both are more efficient than on a NFS share. We also found that putting images on local disk can retain very stable deploying rate for online service. This would be expected, since the local disk's maximum read/write transfer rate is the same on every compute node. Putting images on NFS directory was unable to guarantee the service level requirement. NFS performance drops seriously while staging too many VMs. Although, putting VM images on local node really offers stable allocation speed, it potentially consumes unknown file size. Generally speaking, compute node disk size was fixed and limited. If more VM image files should be prepared, putting these images on Lustre or GPFS partitions would be adequate for both users and cluster administrators since it could not only export large disk size but also lower the allocation time. Using parallel file system as the image storage will lead to roughly a two times speedup for 64 VMs' allocation, compared with NFS.

V. CONCLUSION AND FUTURE WORK

Amazon EC2 services raise the level of new service model for business. EC2 allows its users to specify their own OS environments followed by several selection steps. Rather than buying more machines from vendors, this paper gives some approaches for how to greenly using current computing cluster. The approaches in this paper were to describe a harmless solution for providing pre-created cluster environments on a physical cluster. We explored the effect and affect of present initial results. By means of KVM, we made a dual-use cluster for cloud and HPC possible. From our performance evaluation experiment, partitioning HPC jobs and VM allocation makes private cloud service achieve near-native performance result. Furthermore, no matter how HPC-job load changes over time, any virtual cluster running on compute nodes will not be affected. All techniques or ideas mentioned in this paper could be applied to other HPC clusters to present cloud service in data center.

While current approach helps users manage VMs as jobs to create a customized cluster, it needs to be refined to provide more elastic features. Due to the license issue, our implementation currently just offers Linux-based images for our clients. No window-based virtual cluster provided in Formosa3. And a more accurate reservation mechanism for users should be offered. Once virtual cluster finished, those VM image files should be saved for a specific period of time, and then database marked them as reserved state. And these images could be re-invoked for the same user again. The ability

to make these improvements will require advanced integration of OpenNebula and database systems. Preparing window-based image files, a VPN GUI client, and ample contextualization scripts will be also our future work.

REFERENCES

[1] Amazon Elastic Compute Cloud (EC2), http://aws.amazon.com/documentation/ec2/.

[2] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, P. Maechling, "Scientfic Workflow Applications on Amazon EC2," in *Procs. 5th IEEE International Conference on e-Science. 2009*. pp. 59-66.

[3] HPC as a Service, http://www.penguincomputing.com/POD_old/Benefits/.

[4] Neil Smyth, *Red Hat Enterprise Linux 6 Essentials*, 2010, ch. 13.

[5] Kernel-based Virtual Machine, http://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine/.

[6] Torque Resource Manager, http://www.clusterresources.com/pages/products/torque-resource-manager.php/.

[7] H. Kim, Y. el-Khamra, S. Jha, M. Parashar, "An Autonomic Approach to Integrated HPC Grid and Cloud Usage," *the 5th IEEE International Conference on e-Science*, Oxford, UK, pp. 366–373, Dec. 2009.

[8] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Capacity Leasing in Cloud System using the OpenNebula Engine," *Cloud Computing and Applications, 2008*. Chicago, Illinois, USA.

[9] P. Sempolinski and D. Thain, "A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus," in *Proc. CloudCom'*2010, pp.417–426.

[10] V. Buge, H. Hessling, Y. Kemp, M. Kunze,O. Oberst, G. Quast, A. Scheurer, and O. Synge, "Integration of Virtualized Worker Nodes in Standard Batch Systems," in *Journal of Physics: Conference Series, Volume 219, Issue 5*, pp. 052010, 2010.

[11] D. H. van Dok, "Pushing Torque jobs in a chroot environment" Available at: http://www.nikhef.nl/pub/projects/grid/gridwiki/images/3/37/Momchroot.pdf.

[12] W. Emeneker, D. Jackson, J. Butikofer, and D. Stanzione, "Dynamic Virtual Clustering with Xen and Moab," in *Proc. ISPA Workshops'2006*, pp.440–451.

[13] R. Rose, "Survey of System Virtualization Techniques," Available at: http://www. robertwrose.com/vita/rose-virtualization.pdf, 2004.

[14] T. Garfinkel and M. Rosenblum, "When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments," in *Proc. 10th Workshop on Hot Topics in Operating Systems (HOTOS-X)*.

[15] J. Cope, M. Oberg, H. M. Tufo, and M. Woitaszek, "Shared Parallel Filesystems in Heterogeneous Linux Multi-Cluster Environments," in *Proc. 6th LCI International Conference on Linux Clusters: The HPC Revolution*.

[16] T. Deshane, Z. Shepherd, J. N. Matthews, M. Ben-Yehuda, A. Shan, and B. Rao, "Quantitative Comparison of Xen and KVM," *Xen Summit, Boston, MA, USA*, pp. 1–2, 2008.

[17] L. Nussbaum, O. Mornard, F. Anhalt, J. P. Gelas, "Linux-based virtualization for HPC clusters," Available at: http://www.loria.fr/~lnussbau/files/linux-virtualization-mls09.pdf

[18] O. Khalid, I. Maljevic and R. Anthony, "Dynamic Scheduling of Virtual Machines Running HPC Workloads in Scientific Grids," in *Proc. 3rd IEEE International Conference of New Technologies, Mobility and Security*, 2009.

[19] Jeffrey Shafer, "I/O Virtualization Bottlenecks in Cloud Computing Today," *Workshop on I/O Virtualization (WIOV 2010)*, Pittsburgh, PA, March 2010.

[20] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, P. Maechling, "Data Sharing Options for Scientific Workflows on Amazon EC2," *SC'10 Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*., submitted for publication.