Noname manuscript No.
(will be inserted by the editor)

`vcluster`: A Framework for Auto Scalable Virtual Cluster System in Heterogeneous Clouds

Seo-Young Noh · Steven C. Timm · Haengjin Jang*

Received: date / Accepted: date

Abstract Cloud computing is an emerging technology and is being widely considered for resource utilization in various research areas. One of the main advantages of cloud computing is its flexibility in computing resource allocations. Many computing cycles can be ready in very short time and can be smoothly reallocated between tasks. Because of this, there are many private companies entering the new business of reselling their idle computing cycles. Research institutes have also started building their own cloud systems for their various research purposes. In this paper, we introduce a framework for virtual cluster system called `vcluster` which is capable of utilizing computing resources from heterogeneous clouds and provides a uniform view in computing resource management. `vcluster` is an IaaS (Infrastructure as a Service) based cloud resource management system. It distributes batch jobs to multiple clouds depending on the status of queue and system pool. The main design philosophy behind `vcluster` is cloud and batch system agnostic and it is achieved through plugins. This feature mitigates the complexity of integrating heterogeneous clouds. In the pilot system development, we use FermiCloud and Amazon EC2, which are a private and a public cloud system, respectively. In this paper, we also discuss the features and functionalities that must be considered in virtual cluster systems.

Seo-Young Noh
National Institute of Supercomputing and Networking, Korea Institute of Science and Technology Information, Daejeon, 305-806, Korea
Tel.: +82-42-869-0860
Fax: +82-42-869-0789
E-mail: rsyoung@kisti.re.kr

Steven C. Timm
Grid and Cloud Computing Department, Fermi National Accelerator Laboratory, Batavia, IL, 60510, USA

Haengjin Jang (Corresponding author)
National Institute of Supercomputing and Networking, Korea Institute of Science and Technology Information, Daejeon, 305-806, Korea

Keywords vcluster · Virtual Cluster · Cloud Resource Management · Heterogeneous Clouds

1 Introduction

Cloud computing is one of the hottest keywords in the IT world and is being widely adapted in various research fields [1–3]. Such a trend is no exception in the high energy physics (HEP) area. In experiments in HEP, much storage and many computing cycles are generally required to analyze data produced from accelerators in order to find new physics [4,5]. Building huge computing power is always considered a top priority in the research area. Specifically in HEP, adapting cloud computing technologies is getting popular because of its flexible computing resource management.

Cloud computing can be considered as a type of HTC (High Throughput Computing) like Grid, but there is a big difference between Cloud and Grid. Grid focuses on **orchestration** of scattered physical computing resources while cloud focuses on utilizing idle physical computing cycles through **virtualization**. One of main reasons that cloud attracts many researchers is its flexibility in computing resource allocations. By using virtualization technology such as KVM [6,7] and Xen [8,9], multiple homogeneous or heterogeneous virtual machines can be created on physical machines. Since cloud uses virtualization technology for high throughput computing[1], some people have expressed concerns that there will be performance degradation. However, many studies show that overhead for virtualization is marginal compared to the performance of bare metal. According to [13], KVM has achieved over 90% of performance compared to the bare metal.[2]

Cloud infrastructure based on virtualization will gain its importance in HEP applications which require many computing cycles in general. Therefore, it will be useful to develop a methodology to combine scattered cloud computing resources in a single virtual cluster for fast analysis.

In this paper, we introduce a virtual cluster system called `vcluster` which is automatically scalable in heterogeneous cloud systems like private and public. `vcluster` is agnostic with respect to its underlying systems, not only its underlying cloud systems, but also its batch systems. Such agnostic approach is achieved by adapting plugin architecture. Our approach provides many flexibilities. Most importantly, it can provide a uniform view for both cloud systems and batch systems. Regardless of its underlying systems, users will have a single view. We have built a pilot system running on top of FermiCloud [14] and Amazon EC2 [15], where FermiCloud is a private cloud built

[1] There are many research groups that have been utilizing virtualization technology in HPC (High Performance Computing) [10,11]. Fermilab also evaluated the MPI performance using virtual machines communicating via Infiniband [12] and found similar performance.

[2] In some applications like SAP, it shows up even better performance than bare metal. Despite the small overhead for virtualization, flexibility in computing resource allocation compensates for it and provides many advantages in resource management perspective.

with OpenNebula [16] and Amazon EC2 is public. In this paper, we also discuss considerations and functional features which should be implemented when managing multiple cloud systems in a single virtual cluster system.

2 Related Work

Many publications on cloud computing can be found in various research areas. David et. al [17] proposed cloud computing based administrative architecture for grid applications which small groups can dynamically create, join and manage grid infrastructure. Gable et. al [18] introduced a dynamic multi-cloud site batch system which uses distributed IaaS clouds for HEP applications. In their approach, Cloud Scheduler is used to communicate with Condor Scheduler [19]. Cloud Scheduler [20] checks the status of the queue and boots virtual machines in various cloud systems. User submits regular condor jobs, but each condor job includes custom condor attributes for Cloud Scheduler such as `VMType`, `VMLoc`, and so forth. In the attribute description, locations for virtual machine images are described in order for Condor Scheduler to launch the virtual machine images.[3] Their approach is similar to `vcluster` in that Cloud Scheduler provides a mechanism to bridge multiple cloud sites, but there are also differences in that `vcluster` is not restricted to its underlying scheduler and no additional attributes are required for condor jobs.

One would be very curious about the performance of virtualization because virtual machines are running on physical machines. Alef and Gable [21] showed that performance gap is marginal when they benchmarked three types of configurations such as virtual machine per core, virtual machine with multiple cores, and bare metal with benchmark suite HEP-SPEC06.[4] In general, the configuration for a virtual machine per core has slightly less performance compared to a virtual machine with multi cores because of extra overhead associated with running multiple virtual machines on the same physical hardware. They also observed that their results are quite close to the performance evaluation conducted by a research group at the UCSB [23].

There were also many surveys to understand the quality and cost benefits of cloud computing. Hussain [24] introduced an architecture to aid the design and implementation of attestation for cloud computing service. Marcos [25] investigated seven scheduling strategies and tried to find how the strategies achieve a balance between performance and cost. There was also an attempt to evaluate cost and performance of a commercial cloud compared to a typical HPC, showing that using commercial cloud is a reasonable approach for scientific computing communities because their computing powers are comparable to HPC systems. However, one recommendation is not to transfer large data from outside which causes more cost rather than storing data inside [26].

[3] Transmission delay of virtual machine images is one of challenges. According to [18], it took 5 hours to boot virtual machine of 16 GB transferred from Ottawa to Victoria.

[4] The benchmark used in their evaluation is originally come from SPEC CPU2006 [22] and specific to HEP.

3 Conceptual Design of Virtual Cluster System

Applications of HEP require many computing resources in general. Therefore, high throughput computing technologies like Grid are always considered important and naturally adopted in HEP research. In addition to Grid, cloud computing has been gaining attentions from HEP researchers because of its ability of quickly augmenting computing powers. Therefore, it is natural for HEP researchers to utilize cloud computing in their research. To this end, it is required to combine scattered cloud computing cycles and build a single **virtual cluster system** running over multiple clouds. This is the motivation of `vcluster` which is able to manage cloud computing resources focusing on HEP research area.

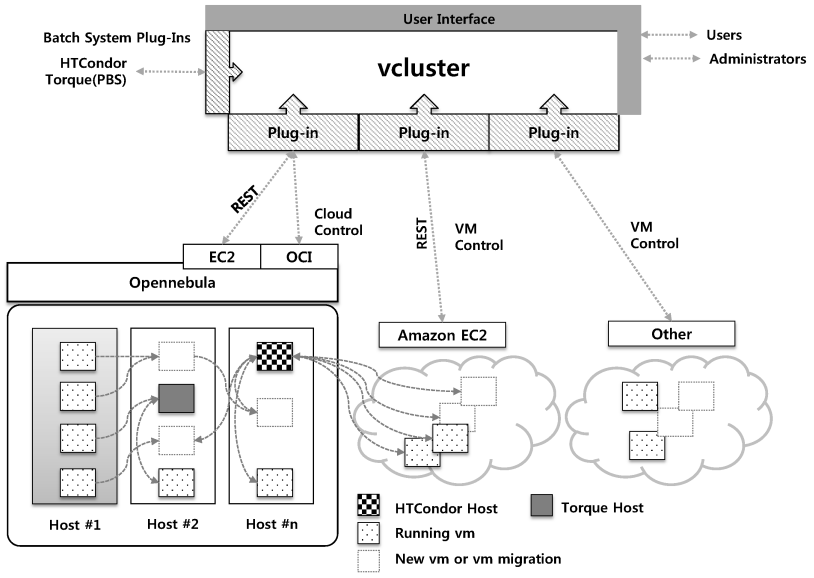

Fig. 1 Conceptual Diagram of `vcluster`

`vcluster` is implemented on top of IaaS-based cloud systems. It combines existing private and public cloud systems to make an automatically scalable cluster system, utilizing idle computing powers and providing great flexibility to handle grid jobs. For a private cloud system, it can directly control worker nodes running as virtual machines; therefore it can reallocate virtual worker nodes in the private cloud system, and migrate virtual worker nodes to specific host machines for a certain condition in order to save power consumption. If there are no more available slots for new virtual worker nodes, `vcluster`

redirect jobs to public clouds. Such worker nodes in private and public cloud systems automatically join the cluster pool managed by `vcluster`.

Fig. 1 shows a conceptual diagram of `vcluster`. It is agnostic with respect to cloud system. Each cloud system communicates with `vcluster` using its corresponding cloud plugin. It is also agnostic with respect to batch systems. Whenever there is a proper plugin for a batch system, for example, HTCondor [19,27] or Torque [28], it can simply dock to `vcluster` providing a uniform view to users. `vcluster` provides a single view for different cloud systems and batch systems. Regardless of underlying cloud and batch systems, all jobs are handled in a uniform way.

4 A Framework of Virtual Cluster System

In this section, we will discuss the framework of `vcluster` to implement the conceptual design introduced in Section 3. `vcluster` has three main component blocks: **Controller**, **Load Balancer** and **Monitoring**, as shown in Fig. 2. `vcluster` is plugin based. All external information is obtained through the plugins in order to flexibly increase and decrease its capabilities of handling jobs.

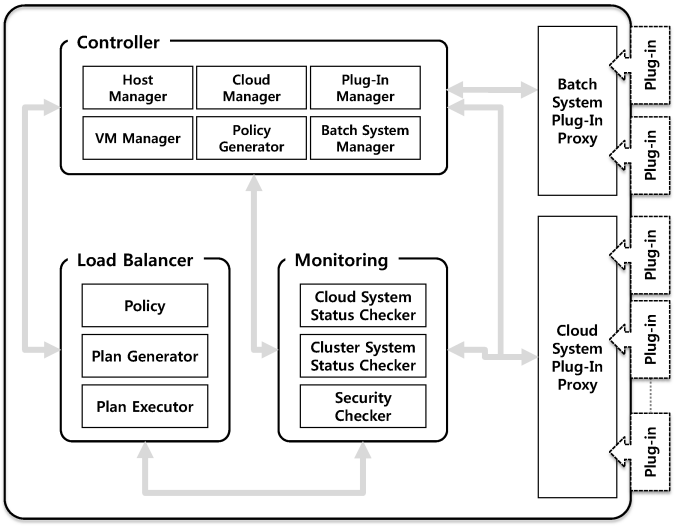

Fig. 2 Components of vcluster

Each component consists of multiple modules and they communicate with each other through message passing mechanism.

4.1 Controller

Controller consists of six modules and provides management functionalities for `vcluster` system. The following describes the functionalities of modules in Controller component.

- **Host Manager:** It manages host systems. Virtual machines can be migrated from one host to another according to a migration policy. Host systems can be turned off or on in order to save power consumption or process more jobs.

- **Cloud Manager:** It provides a single viewpoint of underlying cloud systems regardless of private or public. It keeps track of available clouds for service. For example, it prioritizes cloud resources depending on available slots and charging rates.

- **VM Manager:** It launches and manages virtual machines. It can determine which cloud system has to be selected when launching virtual machines. In order to manage running virtual machines and find a cloud system, it maintains a list of virtual machines and requests a proper cloud system to Cloud Manager.

- **Plugin Manager:** It handles plugins for cloud and batch systems. It provides a feature to switch one plugin to another for proper communication with a underlying system.[5]

- **Policy Generator:** It makes a policy for jobs, running and idle virtual machines. A created policy is passed to **Load Balancer** component. It is first analyzed by Policy Analyzer and executed by Executor, respectively.

- **Batch System Manager:** It communicates with an underlying batch system. It can directly send commands to a batch system through a corresponding plugin. `vcluster` provides batch system related commands which are uniform regardless of batch systems. Such commands are transformed to the commands of real batch systems in the corresponding batch system plugins.

[5] We do not consider multiple plugin supports at the same time. Therefore, if current batch plugin is for HTCondor, it has to be unplugged and replaced with a plugin for Torque in order to communicate with Torque batch system. This approach simplifies our implementation because it is unreasonable to support multiple batch systems in a single cluster system in HEP applications.

4.2 Load Balancer

Load Balancer manages loads on `vcluster`. It makes an action plan based on a policy generated by Policy Manager and executes the plan. Such a plan contains information about how many virtual machines should be launched and where virtual machines to be distributed to which hosts. The following describes modules in Load Balancer.

- **Policy:** It contains policy information which is used when distributing loads over the clouds. Such information provides a clue when `vcluster` moves loads to specific hosts in order to reduce power consumption or take care of system failures
.
- **Plan Generator:** It makes an action plan how to apply the policy to `vcluster`. The plan contains an allocation scheme such that which virtual machines are allocated to which cloud system.

- **Plan Executor:** It performs the execution of a plan generated by Plan Generator. It simply calls relevant functions provided by Controller. Isolating the execution from analysis gives one advantage such that `vcluster` can adaptively make a decision in virtual machine management with latest information considering current running environment.

4.3 Monitoring

Monitoring component is responsible for checking the health of cloud and batch systems. It regularly probes the status to check if any required action is necessary. For example, if there are too many jobs in a waiting queue, it notifies Controller in order to make VM Manager launch virtual worker nodes. The following describes each module in Monitoring component.

- **Cloud System Status Checker:** It monitors the heartbeat of cloud systems to check if they are reachable. If there is any unexpected symptom found, then it makes a corresponding message and sends the message to Controller for proper actions.

- **Cluster System Status Checker:** It checks the status of a cluster system including its queue status and pool status. In case there are too jobs waiting, it generates a corresponding message and sends the message to Controller to launch more virtual worker nodes.

- **Security Checker:** It monitors any security vulnerability of `vcluster` from users or from outsides.

5 Implementation

In this section, we will discuss the implementation of `vcluster`. Not all of functionalities are implemented, but our implementation at least shows the feasibility of our conceptual idea which makes a virtual cluster system running over heterogeneous cloud systems.

5.1 Status of Development

A pilot system for `vcluster` has been developed and its feasibility test has been performed on top of FermiCloud and Amazon EC2 using HTCondor batch system.[6] In order to test its functional feasibility, we first implemented key functionalities of three modules such as Cloud Manager, VM Manager and Cluster System Status Checker in the system architecture shown in Fig. 2. Fig. 3 shows the graphical user interface (GUI) of `vcluster`. In addition to a command line interface, system administrator can manage a virtual cluster system through GUI.[7]

(a) `vcluster` main screen (b) VM status in a cloud

Fig. 3 Graphical User Interface of `vcluster`

Fig. 4 shows the implemented components in our pilot system and how internal modules are interacting with each other.[8]

Cloud Manager loads information about cloud systems which will be used in `vcluster`. Each cloud system is defined as **cloud element** as shown in Fig. 5, where `max` is the maximum virtual machines to be launched in a corresponding cloud system. It is worth noting that cloud elements can be various depending on its type, e.g. private and public. In case of public cloud,

[6] In our experiment system, we use HTCondor batch system, but not limited to a specific batch system.

[7] `vcluster` GUI, managing a virtual cluster system, has been demonstrated at Supercomputing Conference in 2012.

[8] Such modules are simplified version and including minimal functionalities to evaluate the concept of `vcluster`.

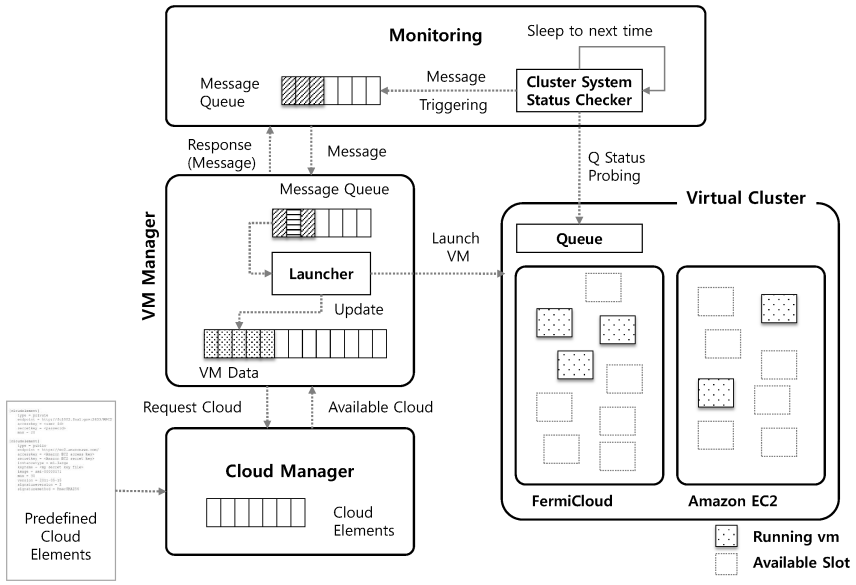

Fig. 4 Interactions among implemented modules

`vcluster` communicates with the cloud element through Amazon EC2 API style queries, which are REST (REpresentational State Transfer) [29] protocol-based and encrypted with a secret key. However, in case of private cloud, `vcluster` directly communicates with the cloud element with APIs provided by the cloud system, e.g., OCA (Open Cloud API) [30] in OpenNebula.[9]

5.2 Ratio based VM Determination Algorithm

In order to automatically start up virtual worker nodes in cloud systems, it is required to periodically check the status of the queue of a cluster system and determine how many worker nodes are necessary. Finding the number of worker nodes is based on the ratio between running jobs and idle jobs. In our feasibility test, we use a simplified approach such that the ratio, I/R, has to be less than or equal to 1, where I is the number of idle jobs in a queue while R is running jobs. If $I/R = k$ and $k > 1$, then we have to find f such that

[9] OpenNebula provides three different types of accesses to the cloud system such as OCA, OCCI (Open Cloud Computing Interface) [31], and Amazon EC2 [32]. In our implementation, we directly use OCA API because as of writing time, OpenNebulas OCA API functions are a more complete set of functions to manage the cloud system compared to the OCCI and EC2 APIs.

```
[cloudelement]
    type = private
    endpoint = http://fcl002.fnal.gov:2633/RPC2
    accesskey = <user id>
    secretkey = <password>
    max = 20

[cloudelement]
    type = public
    endpoint = https://ec2.amazonaws.com/
    accesskey = <Amazon EC2 access key>
    secretkey = <Amazon EC2 secret key>
    instancetype = m1.large
    keyname = <my secret key file>
    image = ami-00000171
    max = 30
    version = 2011-05-15
    signatureversion = 2
    signaturemethod = HmacSHA256
```

Fig. 5 Examples of cloud elements describing FermiCloud and Amazon EC2

$I/(R+f) \leq 1$, where f can be defined as the number of jobs to be executed more. Therefore, we can set up a formula as follow:

$$\frac{I}{R} = k \qquad \Longrightarrow \qquad \frac{I}{R+f} \leq 1$$

Let v and n be the number of virtual machines launched and the number of cores for each virtual machine, respectively.[10] Since R is proportional to v, we can express R and I in terms of v and f as shown in Eq. 1.

$$\begin{aligned} 1 &\geq \frac{I}{R+f} \\ f &\geq I - R \\ f &\geq R \cdot (k-1) && \because \frac{I}{R} = k \\ f &\geq n \cdot v \cdot (k-1) && \because R \cong n \cdot v \end{aligned} \tag{1}$$

Since f is the number of new jobs to start on virtual machines and each virtual worker node can execute n jobs, respectively. f/n is the new number of worker nodes for `vcluster` to launch. Therefore, the minimum worker nodes can be expressed in terms of v and k as shown in Eq. 2. Whenever VM Manager received a message from Monitoring which requests more worker nodes, VM Manager will launch virtual worker nodes according to the value of $v \cdot (k-1)$.

[10] In order to simplify our problem, we assume all virtual worker nodes have same virtual cores.

$$f/n = v \cdot (k-1) \tag{2}$$

By using Eq. 2, we can develop an algorithm to find the number of virtual worker nodes. Algorithm 1 shows the logic implemented in `vcluster`.

Algorithm 1 Ratio-based Virtual Machine Determination Algorithm

```
1: procedure probeQueue(v, n)          ▷ v: running vms, n: number of cores
2:     R ← QStatus.getRunningJob()                          ▷ f: running jobs
3:     I ← QStatus.getIdleJob()                              ▷ i: idle jobs
4:     if R ≤ 0 then
5:         R ← 1                                            ▷ initialize R
6:     end if
7:     k ← R / I                                            ▷ find the ratio
8:     if k > 1 then
9:         f ← n × v × (k − 1)                              ▷ find new vms
10:        msg ← MonMessage(f)                              ▷ create a message
11:        msgQueue.put(msg)                    ▷ send a message to Monitoring
12:    else
13:        sleepSec+ = Config.SLEEP_SEC_INC                 ▷ next probing time
14:        if sleepSec > Config.MAX_SLEEP_SEC then
15:            sleepSec = Config.MAX_SLEEP_SEC              ▷ probing time limit
16:        end if
17:    end if
18: end procedure
```

Depending on the ratio of f and i, it tries to find how many additional virtual worker nodes are required to process idle jobs. Once finding the value, it generates a message to Monitoring which triggers the message to VM Manager to launch additional worker nodes. In our pilot implementation, our approach uses simple ratio information which might not be proper in reality. In real production environment, there are many variables to consider. For example, history of complete jobs, average running hours, and job submission patterns will be those variables. Considering various aspects to find f is out of scope of this paper and it requires further research.

In the algorithm, if $I/R < 1$ is satisfied, then the cluster system status checker enters sleeping mode for a while. In our implementation, we use a stepwise function to find the next sleeping period. For example, if $I/R < 1$ for 5 seconds, then next probing time will be 10 seconds later, then 20 and so forth. However, if its sleeping time reaches a limit, e.g. 30 minutes, then it remains. Notice that current function to find next sleep duration is simple and needs to be improved in order to achieve gracefully handle complex circumstances.

5.3 Considerations in VM Management

When a message has been triggered by the status checker, notifying that there are idle jobs more than expected, such a message is redirected to VM Manager with values—k and R. VM Manager finds proper cloud systems by querying Cloud Manager. It also generates commands to launch virtual worker nodes depending on the types of cloud systems. Once virtual worker nodes are successfully launched, such worker nodes are managed in VM data structures. It is worth noting that VM Manager takes the following cases into account:

1. The number of virtual worker nodes may not be launched in a single cloud system if there are no more available slots in the cloud system.
2. Multiple cloud systems recommended from Cloud Manager may not be same cloud type.
3. There must be a uniform way to manage virtual worker nodes regardless of cloud system types.

Current `vcluster` manages virtual worker nodes launched in different cloud systems in a common data structure called `VM Data` which is the corner stone to manage virtual worker nodes in a uniform way.

We have also noticed that it is important to have a capability of decreasing the number of virtual worker nodes depending on the status of queue and virtual worker nodes. `vcluster` is designed to recall idle virtual worker nodes and finally turns off host systems which do not have any worker node running. In public cloud, `vcluster` just turns off idle virtual worker nodes while it turns off physical systems in case of private cloud.

6 Conclusions

Cloud environment has many advantages not only in a system management, but also in utilizing computing resources. Cloud computing provides ease of management and rapid preparation of computing cycles. If there are available computing resources, such resources can be claimed on demand to improve utilization of computing resources. Because of its flexibility, many private companies and research institutes have been building their own cloud systems for their new businesses and researches. We can summarize the advantages of cloud environment such as 1) failure isolation, 2) green IT by energy saving[11], 3) high throughput computing by utilizing unclaimed computing resources, 4) fast deployment, 5) live migration from a host to another, 6) testbed for new product and development.

In this paper, we have introduced `vcluster` which is a framework to build a virtual cluster system in different types of cloud systems. The fundamental design approach of `vcluster` is agnostic on its underlying cloud systems and batch systems. In addition, it provides a uniform view on a cluster system built

[11] It can be achieved by virtual worker node migration technology to specific host systems.

on various cloud systems. Such a design concept could be achieved by adapting plugin architecture. Independent modules are communicated through message queues. We have also noticed that a virtual cluster system utilizing various cloud systems must take into account: 1) virtual worker nodes might not be launched in a single cloud system, 2) different types of cloud systems could be involved when launching virtual worker nodes, and 3) there must be a uniform way to manage such virtual worker nodes being deployed over multiple cloud systems.

While building `vcluster` on top of FermiCloud and Amazon EC2, we have found that there are many interesting research topics such as finding suitable time intervals to probe the status of queue, which is one of fundamental keys to achieve fast response time, finding a proper number of virtual worker nodes, how to deploy them to scattered cloud systems, and how to avoid secret key information in cloud elements.

`vcluster` is at the early stage of full production; therefore, there are many things to be implemented and improved for stable production. Before directly applying `vcluster` for production level, we will first make vcluster operate alongside a real cluster system as a shadow cluster. Extra jobs waiting in the queue could be redirected to `vcluster`. Such an approach as an intermediate stage will be valuable to determine if `vcluster` could be working stably in production level. We hope that `vcluster` introduced in this paper provides a good insight to researchers to utilize cloud infrastructures in their research.

Acknowledgement

This research was supported by the National Research Foundation (NRF) of Korea through contract N-13-NM-IR04.

References

1. Ian T. Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. *CoRR*, abs/0901.0131, 2009.
2. Jin-Mook Kim, Hwa-Young Jeong, Ilkwon Cho, SunMoo Kang, and JongHyuk Park. A secure smart-work service model based openstack for cloud computing. *Cluster Computing*, pages 1–12, 2013.
3. Xu Ma, Jin Li, and Fangguo Zhang. Outsourcing computation of modular exponentiations in cloud computing. *Cluster Computing*, pages 1–10, 2013.
4. Heinz Stockinger, Asad Samar, Koen Holtman, Bill Allcock, Ian Foster, and Brian Tierney. File and object replication in data grids. *Cluster Computing*, 5(3):305–314, 2002.
5. Adriana Iamnitchi, Shyamala Doraimani, and Gabriele Garzoglio. Workload characterization in a high-energy data grid andimpactonresource management. *Cluster Computing*, 12(2):153–173, 2009.
6. Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.
7. KVM. `http://www.linux-kvm.org/`, May 2013.
8. Xen. `http://xen.org/`, May 2013.

9. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Timothy L. Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP*, pages 164–177, 2003.
10. Yacine Kessaci, Nouredine Melab, and El-Ghazali Talbi. A pareto-based metaheuristic for scheduling hpc applications on a geographically distributed cloud federation. *Cluster Computing*, pages 1–18, 2012.
11. Haibao Chen, Song Wu, Xuanhua Shi, Hai Jin, and Yu Fu. Lcm: A lightweight communication mechanism in hpc cloud. In *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*, pages 443–451, 2011.
12. Keith Chadwick. Fermigrid and fermicloud update. In *2012 International Symposium on Grids and Clouds*, Taipei, Taiwan, 2012.
13. RedHat. KVM - KERNEL BASED VIRTUAL MACHINE. `www.redhat.com/f/pdf/rhev/DOC-KVM.pdf`, May 2013.
14. FermiCloud. `http://www-fermicloud.fnal.gov/`, May 2013.
15. Amazon EC2. `http://aws.amazon.com/ec2/`, May 2013.
16. The Open Source Toolkit for Cloud System. `http://opennebula.org/`, May 2013.
17. David Isaac Wolinsky, Panoat Chuchaisri, Kyungyong Lee, and Renato Figueiredo. Experiences with self-organizing, decentralized grids using the grid appliance. *Cluster Computing*, pages 1–19, 2012.
18. I. Gable, A. Agarwal, M. Anderson, P Armstrong, A Charbonneau, R Desmarais, K Fransham, D Harris, R Impey, C Leavett-Brown, M Paterson, D Penfold-Brown, W Podaima, R Sobie, and M Vliet. A batch system for hep applications on a distributed iaas cloud. In *proceedings of Computing in High Energy Physics 2010*, Taipei, Taiwan, 2010.
19. HTCondor. `http://www.cs.wisc.edu/condor/`, May 2013.
20. Patrick Armstrong, Ashok Agarwal, A. Bishop, Andre Charbonneau, Ronald J. Desmarais, K. Fransham, N. Hill, Ian Gable, S. Gaudet, S. Goliath, Roger Impey, C. Leavett-Brown, J. Ouellete, M. Paterson, C. Pritchet, D. Penfold-Brown, Wayne Podaima, D. Schade, and Randall J. Sobie. Cloud scheduler: a resource manager for distributed compute clouds. *CoRR*, abs/1007.0050, 2010.
21. M Alef and I Gable. Hep specific benchmarks of virtual machines on multi-core cpu architectures. *Journal of Physics: Conference Series*, 219(5):052015, 2010.
22. Spec cpu2000: measuring cpu performance in the. *New Millennium Comput.*, 33(7):28, 2000.
23. Lamia Youseff, Rich Wolski, Brent Gorda, and Chandra Krintz. Evaluating the performance impact of xen on mpi and process execution for hpc systems. In *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, VTDC '06, pages 1–, Washington, DC, USA, 2006. IEEE Computer Society.
24. Mohammed Hussain and HanadyM. Abdulsalam. Software quality in the clouds: a cloud-based solution. *Cluster Computing*, pages 1–14, 2012.
25. MarcosDias Assuno, Alexandre Costanzo, and Rajkumar Buyya. A cost-benefit analysis of using cloud computing to extend thecapacity of clusters. *Cluster Computing*, 13(3):335–347, 2010.
26. G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B.P. Berman, and P. Maechling. Scientific workflow applications on amazon ec2. In *E-Science Workshops, 2009 5th IEEE International Conference on*, pages 59 –66, dec. 2009.
27. Rajesh Raman, Miron Livny, and Marv Solomon. Matchmaking: An extensible framework for distributed resource management. *Cluster Computing*, 2(2):129–138, 1999.
28. Michael Jackson. Moab and torque achieve high utilization on flagship nersc xt4 system. In *CUG 2008 Proceedings*, 2008.
29. Representational State Transfer. `http://en.wikipedia.org/wiki/Representational_State_Transfer`, May 2013.
30. Java OpenNebula Cloud API 3.0. `http://opennebula.org/documentation:rel3.0:java`, May 2013.
31. Open Cloud Computing Interface. `http://occi-wg.org/`, May 2013.
32. OpenNebula Cloud. `http://opennebula.org/cloud:cloud`, May 2013.