Integration of Openstack cloud resources in BES III computing cluster

Integration of Openstack cloud resources in BES III computing cluster

Haibo Li[1], Yaodong Cheng[1], Qiulan Huang[1], Zhenjing Cheng[1,2] and Jingyan Shi[1]

[1]IHEP computing center, 19B Yuquan Road, Beijing 100049, China
[2]University of Chinese Academy of Sciences, Beijing, China

E-mail: lihaibo@ihep.ac.cn

Abstract.Cloud computing provides a new technical means for data processing of high energy physics experiment. However, the resource of each queue is fixed and the usage of the resource is static in traditional job management system. In order to make it simple and transparent for physicist to use, we developed a virtual cluster system (vpmanager) to integrate IHEPCloud and different batch systems such as Torque and HTCondor. Vpmanager provides dynamic virtual machines scheduling according to the job queue. The BES III use case results show that resource efficiency is greatly improved.

1. Introduction

As a new computing model, cloud computing has an important impact on the high energy physics (HEP) community. As HEP has great demands for high throughput computing (HTC) workloads, cloud computing usingvirtualization technology can provide flexible computing resources. Such a situation often occurs when the resources of one experimental group are not enough, the resources of other resources are still idle. Cloud computing makes it possible for the sharing of computing resource among multiple experiment group.

The IHEP Public Service Cloud Computing Platform (IHEPCloud) [1] is an IaaS platform developed by the IHEP computing center based on Openstack. One of IHEPCloud's function is to provide virtual machine (VM) resources for HEP experiments in IHEP. Now IHEPCloud has a cluster with more than 1,000 cores. Studies have shown that particle physics application code run equally well in a VM or on the native system [2]. However, it lacks dynamic scheduling interface between job scheduling system and virtual resources. The commonly used job scheduling system in IHEP now is Torque [3] and HTCondor [4]. So we provide a tool and technique to integrate the IHEPCloud and job scheduling system (Torque, HTcondor), making the resources shared between different experiment groups and dynamically scheduling according to the status of jobs and finally greatly improving the resource utilization rate.

The rest of the paper is organized as follows. Section 2 presents the related work about the integration between different cloud systems and job management systems. Section 3 illustrates the design and implementation of cloud scheduler, which allocates and reclaims VMs dynamically according to the status of TORQUE and HTCondor queues and the design of resource pool to improve resource utilization. Section 4 gives a BESIII use case which illustrates the effectiveness of our mechanism. And we conclude the paper in Section 5.

2. Related work

With the development of cloud computing and related technologies, international research work is gradually expanding from different levels. Many cloud computing infrastructure engines are emerging, such as open source OpenStack[5], OpenNebula[6], Nimbus [7] and other cloud computing infrastructure engine. More and more institutes began to integrate the HTC environment with cloud resources. Here are some unique and specialized solutions being actively developed.

Cloud Scheduler [8] is an object oriented python-based package run alongside Condor to manage VMs for jobs based on the available cloud resources and job requirements. Cloud scheduler boots and manages the user-customized virtual machines in response to a user's job submission. A user creates their VM and stores it in the VM image repository. They write a job script that includes information about their VM and submits it to the condor job scheduler. The cloud scheduler reads the queues of the condor job scheduler, request that one of the available cloud resources boot the user VM, the VM advertises itself to the condor job scheduler which then dispatches the user job to that VM. Once jobs are finished, if there are no more jobs in the condor queue that user them, Cloud Scheduler shuts down the worker nodes, and returns the VMs to the cloud.

The grid-middleware project: "Distributed Infrastructure with Remote Agent Control" (DIRAC) [9] provided a software framework to submit jobs to EC2 compatible IaaS clouds. The pilot agent is preinstalled in the OS of the virtual machines (VMs) deployed on the IaaS clouds. Once VM starts, the agent will contact the DIRAC server and request jobs. User must use the DIRAC client commands to submit jobs, so it is not transparent to users who use Torque and HTCondor as their familiar tools in IHEP.

Dynamic Torque [10] is a tool developed to integrate Torque/Maui batch system and Openstack cloud service. Dynamic Torque provides active and passive mode of communication with Toque/Maui batch system. In Dynamic Torque, the work nodes in the cloud are grouped as 'static' and 'dynamic': static means these worker nodes stay up forever, no matter what current workload is; dynamic means the work nodes will be shut down if no job is in the queue, and they will be launched if jobs are waiting in the queue.

3. System design and implementation

3.1. Architecture

Our goal is to build a dynamic virtual resource pool to support different experiments to improve the resource utilization. The architecture is composed of four layers including physical machines, virtual machines, resource pool manager and job management system. The first layer is physical machines which are bought by different experiments. The second layer is virtual machines managed by Openstack, which don't belong to any experiment. The third layer is resource pool manager, which dynamically allocate virtual machines to different experiments depending on current computing tasks. This layer has fine-grained resource allocation policies to balance the resource sharing and physical machine invest. The fourth layer is job management system. Currently, Torque and HTCondor are supported. Different job queues are assigned to different experiments, and user can use these queues like traditional computing cluster. The different point is that the resource in the queue is dynamic, which can be expanded or be shrunk automatically.

VPMangerisa suite of python script to schedule virtual machines dynamically according to the workload of job management system. The middleware lies between resource management system (such as Torque and HTCondor) and Openstack. The overall architecture of the system is shown in Figure 1.

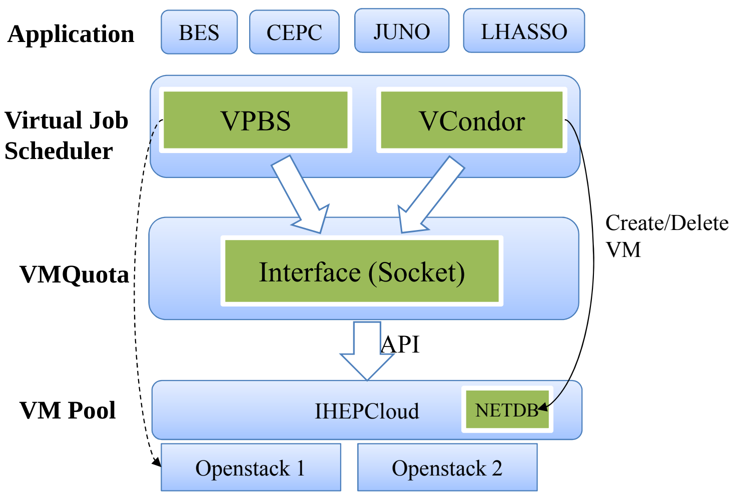


Figure 1. Virtual computing cluster architecture

3.2. Modules
VPManager is composed of different components, they are VM Pool, VM Quota, Virtual Job manger, VM node manager and Accounting system. VM Pool manages one or more Openstack deployments, which hides the detailed information of Openstack from upper applications. VM Pool makes it possible to deploy multiple and different versions of Openstack. VM Quota checks the information of VM Pool and requirements of different applications to allocate or reserve resources. Virtual job manager including VPBS and VCondor checks the status of different queue and get the available VM number and create new VMs or destroy existing VMs. VM node manager checks and controls all the VM run environment such network status, affiliated job queue by an agent running in the virtual machine. Accounting system keeps all the usage information of each virtual machine and generates bills to user.

Virtual job scheduler checks the status of different queue and create or destroy VMs according to the job queue. To support two different job manage systems, we divide this part to two sub modules: VPBS and VCondor. VPBS is responsible for the Torque queue and VCondor is responsible for the HTCondor queue. Job status monitoring system communicates by command lines or APIs to get the current status of each job queue. We developed an external component to use the OpenStack API to control the creation and deletion of VMs.

VM quota checks the information of VM Pool and requirements of different applications to allocate or reserve resources. Load balance system provides an interface to get the information of available virtual resources for each experiment from VM Quota. The VM Quota tells load balance system how many virtual machines one experiment can use and reserve them for a period of time such as 30 minutes. The daemon component asks load balance system to decide how many available virtual resources.

VM node manager checks and controls all VM run environment such network status, affiliated job queue by an agent running in the virtual machine. Computing node management component communicates with Openstack to launch or destroy virtual machines.

3.3. Workflow
This system performs unified management of virtual machines on the basis of job queue. A VM will be created automatically when a job is waiting to run. It will be destroyed when the job is finished and there are no more jobs in queue. After a VM is created, it will be added to the resource pool of corresponding experiment group. Then the VM can get a job to run. After the job finishes, the virtual machine will be shutdown. When the VM shutdown in Openstack, it will be removed from the resource pool. Meanwhile, the computing node management system provides an interface to query

virtual resources usage. It also communicates with job status monitoring system to get the number of queued jobs. Finally, it calls computing node management system to launch or destroy a few of virtual computing nodes. The workflow is illustrated in Figure 2 and described below:

1. HEP users submit a job using ‘qsub’ or ‘condor_qsub’ command.
2.Jobs come into the RMS and are captured by Virtual Job Scheduler.
3. Virtual Job Scheduler requests VM from VM quota management.
4. VM quota calculates the available VM number to reply.
5. Virtual Job Scheduler starts the VM according to the job queues.
6. Once VM starts, the client can get jobs from RMS.

Virtual Computing Cluster middleware software has been developed and deployed, and the detailed development solution can refer to the vpmanager web site [11].

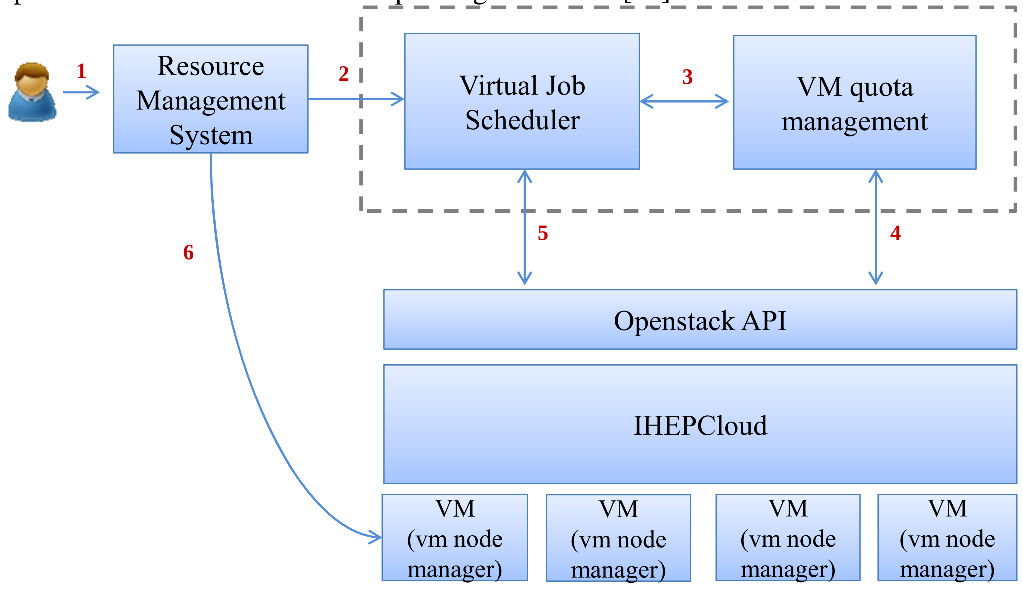
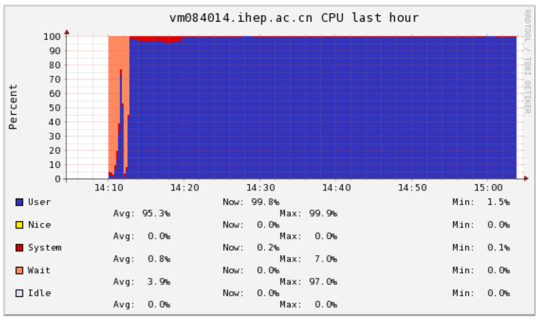


Figure 2. Virtual cluster components and workflow

4. A BESIII use case

The BES III Experiment [12] at the BEPC-2 electron-positron collider at IHEP in Beijing (China) uses collision energies from 3.5 GeV to 4.7 GeV for the spectroscopy and the investigation of the structure of light and heavy hadrons and is in operation since 2008. The total volume of experimental data is already about 0.9 PB, of which about 300 TB is event summary data for physics analysis (DSTs). This amount of data is rather large to be processed in a single computing centre. Use of cloud computing looks like an attractive option to increase the computing resources efficiency and to speed up the data analysis.

Figure 3a shows that the CPU utilization of a BESIII job running on a virtual machine is 99.98%. This result confirms that the job runs the same on both virtual machines and physical machines. Figure 3b shows the effect of the dynamic scheduling of the BESIII experiment in the virtual computing cluster system. In this system, the virtual machine start and stop by the user's job number and the number of virtual machines running the current decision, basically ensure that the number of virtual machines running and the number of jobs are equal, Figure 3b illustrates the system based on the current number of virtual machines and operations Quantity, dynamic scheduling of the effectiveness of resources, which ensure the full sharing of resources between the experimental group. In the virtual computing cluster, the utilization rate of resources is as high as 95%. Compared with the traditional computing cluster, the utilization rate from the average 50% is twice as fast as that of the traditional computing cluster, and the resource utilization rate in the HEP computing environment is greatly improved.



```
■ User                          Now:  99.8%                     Min:  1.5%
             Avg:  95.3%                    Max:  99.9%
■ Nice                          Now:   0.0%                     Min:  0.0%
             Avg:   0.0%                    Max:   0.0%
■ System                        Now:   0.2%                     Min:  0.1%
             Avg:   0.8%                    Max:   7.0%
■ Wait                          Now:   0.0%                     Min:  0.0%
             Avg:   3.9%                    Max:  97.0%
□ Idle                          Now:   0.0%                     Min:  0.0%
             Avg:   0.0%                    Max:   0.0%
```

Figure 3a. CPU efficiency of BESIII

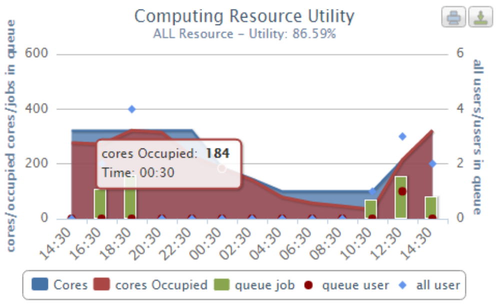


Figure 3b. Dynamic resource schedule of BESIII

5. Conclusion and future work

In this paper, a set of HEP virtual computing cluster system is designed based on Openstack, which realizes the dynamic scheduling of virtual resources, and the system is transparent to the users. From the system analysis and the actual operation, we can conclude that the virtual computing cluster system to ensure HEP computing operations under the premise of correct, the computing environment resource utilization showed great advantages, operating efficiency and physical machine Which proves that the system can meet the requirements of HEP. The current system size is not great, the next step will have more experimental group of resources to join the system. In the large-scale use of continuous improvement and perfection, such as resource preemption, dynamic scheduling algorithm optimization, billing system, support for more batch scheduling system.

Acknowledgments
This work was supported by the National Natural Science Foundation of China(No. 11605223) and the National key Research Program of China "Scientific Big Data Management System" (No.2016YFB1000605).

References

[1] IHEPCloud: http://ihepcloud.ihep.ac.cn/.
[2] Agarwal A, Desmarais R, Gable I, et al. Deploying hep applications using xen and globus virtual workspaces[C]. Journal of Physics: Conference Series. IOP Publishing, 2008, 119(6): 062002.
[3] Staples G. TORQUE resource manager[C]. Proceedings of the 2006 ACM/IEEE conference on Supercomputing. ACM, 2006: 8.
[4] HTCondor: https://research.cs.wisc.edu/htcondor/.
[5] Openstack:https://www.openstack.org/.
[6] Milojičić D, Llorente I M, Montero R S. Opennebula: A cloud management tool[J]. IEEE Internet Computing, 2011, 15(2): 11-14.
[7] The Nimbus Cloud: http://workspace.globus.org/clouds/nimbus.html.
[8] Armstrong P, Agarwal A, Bishop A, et al. Cloud Scheduler: a resource manager for distributed compute clouds[J]. arXiv preprint arXiv:1007.0050, 2010.
[9] DIRAC: http://diracgrid.org/.
[10] Zhang S, Boland L, Coddington P, et al. Dynamic VM provisioning for Torque in a cloud environment[C]. Journal of Physics: Conference Series. IOP Publishing, 2014, 513(3): 032107.
[11] VPManager: https://github.com/hep-gnu/VCondor.git/.
[12] BESIII experiment: http://bes3.ihep.ac.cn/.