Research on elastic resource management for multi-queue under cloud computing environment

Zhenjing CHENG¹,², Haibo LI¹, Qiulan HUANG¹, Yaodong Cheng¹, Gang CHEN¹
¹IHEP computing center, 19B Yuquan Road, Beijing 100049, China
²University of Chinese Academy of Sciences, Beijing, China
E-mail: chengzj@ihep.ac.cn

Abstract. As a new approach to manage computing resource, virtualization technology is more and more widely applied in the high-energy physics field. A virtual computing cluster based on Openstack was built at IHEP, using HTCondor as the job queue management system. In a traditional static cluster, a fixed number of virtual machines are pre-allocated to the job queue of different experiments. However this method cannot be well adapted to the volatility of computing resource requirements. To solve this problem, an elastic computing resource management system under cloud computing environment has been designed. This system performs unified management of virtual computing nodes on the basis of job queue in HTCondor based on dual resource thresholds as well as the quota service. A two-stage pool is designed to improve the efficiency of resource pool expansion. This paper will present several use cases of the elastic resource management system in IHEPCloud. The practical run shows virtual computing resource dynamically expanded or shrunk while computing requirements change. Additionally, the CPU utilization ratio of computing resource was significantly increased when compared with traditional resource management. The system also has good performance when there are multiple condor schedulers and multiple job queues.

1. Introduction
The scale of High Energy Physics computing has been increasing as the origins of the universe and basic material composition are explored. The European Large Harden Colider(LHC), Daya Bay Reactor Neutrino Experiment[1], the Large High Altitude Air Shower(LHAASO)[2], BESIII[3] and many more are all typical compute-intensive applications. Their computing and storage requirements are much larger than before – requiring hundreds of PB storage and hundreds of thousands of CPU cores. Scientists have been seeking different technical means to meet the growing computing needs, such as grid computing, cloud computing and so on.

Cloud computing[4] and virtualization[5] technologies have been selected to manage CPU and other resources in several Grid computing centers. Cloud computing management technology enables computing resources sharing, convenient business deployment and rapid system failure detection. Software applications are isolated from the underlying hardware through virtualization technology. It not only comprises a mode for dividing one single resource into a plurality of virtual resources but also aggregates multiple resources into one virtual resource. Virtualization technology also enables live migration of virtual machines from failed servers to healthy servers. Outdated worker nodes can be gradually and dynamically replaced with freshly updated ones as soon as they phase out (rolling updates)[6].
International High Energy Physics communities such as CERN have implemented private cloud projects to support for scientific computing. A single OpenStack was built which manages tens of thousands of virtual machines[7]. The ALICE Tier-2 in Tornio is based on OpenNebula[8] providing computing and storage resources. In China, IHEPCloud, which was launched in May 2014 based on Openstack[9] and kvm[10], was aimed to provide a self-service cloud platform for IHEP(Institute of High Energy Physics, Beijing) users and a scientific computing platform for HEP experiment applications. It has finished a version upgrade in 2016 from Icehouse to IHEP UMT. There are 28 computing nodes and 672 CPU cores in the scientific computing platform. 768 CPU cores would be added to the cluster by the end of 2016. Virtual machines of the cluster are created using different images for HEP applications and job queues. The status of the cluster and virtual machines are monitored by multiple tools such as Nagios[11] and Ganglia[12]. Batch systems – Torque PBS[13] and HTCondor[14] are in use to control jobs and job queues in the cluster. However, this cluster didn’t support dynamical reconfiguration of computing resource. Different job queues have different usage cases and different peak time of resource usage. It often occurs that computing resources are scarce for some job queues and surplus for others. A large number of jobs are queuing but the overall resource utilization is low (see figure 1)

![Figure 1. CPU Usage of HTCondor physics machine cluster](image)

In this paper, an elastically extended computing resource management mechanism is proposed. The concept of elastic extension was first proposed by Amazon. It is a dynamic extension to the operational resources of cloud applications. The number of VMs which support cloud applications will increase or decrease dynamically. In the IHEPCloud, on-demand elastic virtual computing resource pools were created. The resource pools of each HEP application are dynamically expanded or shrunk according to the number of queued jobs. Upper and lower thresholds were designed to control size of each elastic resource pool. This resource management mechanism would reduce maintenance work load and improve resource utilization.
2. Related work

2.1. HTCondor
HTCondor – the high throughput computing system is widely used in the high energy physics community as the batch system for several Worldwide LHC Computing Grid (WLCG) resources. Moreover, it is the backbone of the GlideInWMS[15], the pilot system used by the computing organization of the Compact Muon Solenoid (CMS) experiment. HTCondor consists of three components: a central manager running the collector and negotiator daemons, one or more submit nodes running the schedd daemon, and one or more worker nodes running the startd daemon. Users submit serial or parallel jobs to HTCondor on submit nodes. The schedd and the negotiator daemon place jobs into a queue and determine when and where to run the jobs based upon a policy. Collector daemon stores semi-structured data such as computing resource information, job submitter information, checkpoint server information and daemon master information. Researchers of IHEP have built a HTCondor cluster of physics machines using 153 blade servers with 2500 CPU cores. Now new resource share policy has been used to support 3 HEP experiments: BES-III, JUNO, CMS.

2.2. kvm performance Optimization and Evaluation
Kvm is a hardware-assisted hypervisor which is supported by VT-x technology[16]. Virtual machine performance will affect application performance to a certain extent. In general, virtual machine performance is evaluated from CPU computing power, DISK I/O and network I/O. HEPSPEC06[17] score was used as the test program of CPU performance. From benchmark tests, researchers in IHEP observed that virtual machine performance was mainly restricted by a CPU performance penalty and DISK IO loss. By default, CPU performance loss rate is about 10% compared to physical machines. To solve this problem, our colleagues tried to bind a virtual processor to a specific physical processor, which proved to reach a higher cache hit rate. It also reduced the migration times of the kvm process between processors. The CPU performance loss on an optimized kvm has dropped to 3%[18]. That can be ignored by sharing resources and improving resource utilization.

3. Elastic resource management algorithm
The above research work provide a strong basis for research on elastic resource management under the IHEPCloud environment. Virtual machines in the IHEPCloud are created using different templates to adapt to different OS and different versions of application software. Users continue to submit Grid jobs using a standard interface while being unaware of the nature of exploited resources. Several job queues and resource pools are built for HEP applications. The elastic resource management algorithm checks the size of the resource pool and the number of queued jobs, analyzes upper and lower thresholds, then increases or decreases batch virtual machines.

Generally speaking, each pilot or virtual machine in IHEPCloud can only run one type of user jobs or one HEP experiment’s jobs. For virtual machines in IHEPCloud, job queue attribute switch, such as condor client reconfiguration, is needed to run a different HEP experiment’s jobs. Thus jobs queuing means this queue’s current computing resource cannot meet its requirements and needed to be extended. The scheduler first calculates how much resource to add, starts a specific number of virtual machines and join them to the HTCondor resource pool. If there are no jobs queued the system will attempt to shrink the cluster, delete idle virtual machines and release computing resource. The scheduler periodically repeat the above process to achieve elasticity of resource pool. In the ith scheduling period, specific steps made by the algorithm are as follows:

- **Step1:** Collect statistics about each HEP application’s resource pool size $\zeta_i$ and the number of queued jobs $\gamma_i$.
- **Step2:** Obtain computing resource upper threshold $\mu_{\text{high}}$ and lower threshold $\mu_{\text{low}}$ of each HEP application.
• **Step3:** Compare the resource pool size $\zeta_i$ and the thresholds ($\mu_{\text{high}}$, $\mu_{\text{low}}$). If $\zeta_i$ is higher than $\mu_{\text{high}}$ the scheduler will generate a list of virtual machines that need to be deleted. Then it processes to step 8. The number of resources that need to be released is

$$\theta_1 = \zeta_i - \mu_{\text{high}}$$

(1)

• **Step4:** If $\zeta_i$ is lower than $\mu_{\text{low}}$ the scheduler needs to expand the resource pool and processes to step 7. The number of resources that needs to be increased is

$$\theta_2 = \mu_{\text{low}} - \zeta_i$$

(2)

• **Step5:** If the job queue is empty and $\delta_i$ virtual machines are idle, the scheduler will delete some virtual machines and release the computing resources. It calculates the number $\theta_1$ of virtual machines that need to be deleted and generate a VM list. Then it proceeds to step 8.

$$\theta_1 = \begin{cases} 
\delta_i, & \zeta_i - \delta_i \geq \mu_{\text{low}} \\
\zeta_i - \mu_{\text{low}}, & \zeta_i - \delta_i < \mu_{\text{low}} 
\end{cases}$$

(3)

• **Step6:** If user jobs are queuing and the cluster has no more free virtual machines the resource pool needs more computing resources. The scheduler calculates the number $\theta_2$ of computing resources that need to be increased. Then it proceeds to step 7.

$$\theta_2 = \begin{cases} 
\gamma_i, & \zeta_i + \gamma_i \leq \mu_{\text{high}} \\
\mu_{\text{high}} - \zeta_i, & \zeta_i + \gamma_i > \mu_{\text{high}} 
\end{cases}$$

(4)

• **Step7:** The scheduler launch a fixed number of virtual machines and adds them to the HTCondor resource pool. Then it processes to step 9.

• **Step8:** Delete the virtual machines which is in the generated VM deletion list and return the computing resource to the cluster. Then it turns to step 9.

• **Step9:** System sleeps for the next scheduling cycle. The elastic resource management algorithm is executed periodically to ensure that the resource pool expands or shrinks on demand. HTCondor constantly sends user jobs to run in the virtual worker nodes.

4. System Design

4.1. The Overall Structure

The front end of the flexible resource management system is the batch processing system HTCondor, middle layer is the flexible cluster scheduler, and the back end consists of virtual machines which can execute the job and the physical machines which provide the resources.

The architecture of the elastic resource management system is shown in figure 2. A user submit jobs to the appropriate queue in HTCondor by issuing the condor_submit command. The elastic cluster scheduler is used to monitor HTCondor status in the batch system to create virtual machines when too many jobs are waiting, or delete them when they turn idle. The backend consists of virtual worker nodes and the physical infrastructure.

![Figure 2. Architecture of the elastic resource management](image-url)
4.2. Elastic cluster scheduler

The elastic cluster scheduler consists of three components: a virtual resource quota service, a resource demand analysis component and a VM schedule component.

a) The resource demand analysis component periodically obtains the attributes of jobs in each HEP application’s job queue and counts the minimum requirements for the number of CPU cores.

b) The VM scheduler component uses the Openstack controller nova to complete virtual machines startup and removal. Then the VMs would join in the HTCondor resource pool or exit from it.

c) The virtual resource quota service limits the number of resources available to each HEP application’s job based on resource allocation policies. It makes sure computing resource is distributed among several job queues fairly and efficiently.

Configuration information which is stored in XML format would be read by the elastic cluster scheduler before it starts to run. Afterwards, the elastic cluster scheduler would complete the following steps in turn:

a) The resource demand analysis component periodically invokes the HTCondor API to obtain per job queue’s status and computing resource demand.

b) The virtual resource quota service returns an upper and lower threshold and calculates available resources, determines resource that can be occupied/released when system extends/shrinks.

c) The VM scheduler component creates VMs in Openstack by Nova API, joins it to the computing resource pool or deletes a VM in Openstack, pulls it out from the resource pool. Figure 3 shows the interaction between three components of the elastic cluster scheduler.

\[ T_i = T_0 + k t \quad (i > 0) \] (5)

The elastic cluster scheduler adopts a loose coupling architecture, improves the flexibility and expansibility of the system in a certain extent. The system is able to provide concurrency support for
multiple HTCondor pools. In addition, it can be ported to OpenNebula, Amazon EC2 and other cloud platforms by increasing a variety of cloud interfaces, which improves scalability and flexibility of the system.

4.3. Resource Quota Service
By default the elastic cluster scheduler allocates computing resources to job queues based on demand. This means that multiple job queues would compete for fixed resources of the underlying cloud platform. The entire resource pool being occupied by one queue leaves much to be desired. Resource quota service limits resources owned by each job queue through dual thresholds. It continuously listens for requests from the elastic cluster scheduler. After receiving a request the resource quota service would return upper and lower thresholds based on the elastic resource management algorithm. To increase the response speed of the resource quota service, it’s better to use a lightweight protocol which has lower latency. The system uses a reliable data-transmission communication mode based on TCP, with JSON as the data-interchange format. Compared to http or other similar protocol, data-transmission is in byte level. For TCP the amount of data transferred is less than http. When parsing a received JSON string the validity of key/value pairs’ format would be checked first. Resource quota service would resend the message to the elastic cluster scheduler if it has been told to do so.

4.4. VM Schedule Component
The VM schedule component receives requests from the elastic cluster scheduler. Then it uses novaclient, a Nova python API to create or delete VMs on Openstack. After the VM is created, internal self-test software within the VM will check the condor client running status, distributed file system and network environment. After that the VM would join the HTCondor resource pool. According to some statistics the VM starts up to 300 seconds or so [19]. It is related to the OS image size, instance type, the number of the instances requested at the same time and so on. When the cloud platform is heavily loaded it may take longer. In IHEPCloud, the VM starts up to 3 minutes at least or 10 minutes at most. In order to shorten the resource pool expansion time and improve the scheduling efficiency a two-level buffer pool is designed.

1) first-level buffer pool: VMs which are active but not in HTCondor resource pool. Those VMs take up computing resources such as VCPUs and they can join in the HTCondor pool and receive user jobs rapidly when needed.

2) Second-level buffer pool: virtual computing, network and storage resources created by the virtualization technology. A hypervisor or virtual machine manager creates a virtual machine on the host hardware and joins it to the first-level buffer pool.

Before the resource pool expands, the elastic cluster scheduler will first determine how many CPU cores to expand, then preferentially select VMs in the first-level buffer pool to join in the HTCondor pool. If there are not enough VMs the scheduler will create some from the second-level buffer pool immediately to replenish the first-level buffer pool. Expanding the number of VMs in the first-level buffer pool helps reducing jobs waiting time. If there are no jobs waiting the resource pool will shrink. Idle VMs will be blocked from receiving new jobs and exit from HTCondor resource pool to join the first-level buffer pool. The elastic cluster scheduler periodically checks the first-level buffer pool, deletes the excess VMs and releases the computing resources into the second-level buffer pool.

5. System deployment
The elastic resource management system has been deployed on the IHEPCloud – a public service cloud platform of IHEP. Data analysis and processing jobs of LHAASO and JUNO are supported. In the future more HEP job queues and a larger pool of virtual computing resource will be supported.
Figure 4. Computing resource utility of LHAASO

Figure 4 shows LHAASO computing resource usage during 2016/05/31 to 2016/07/01. The blue area represents total number of computing resources (VCPUs). The red area represents number of computing resources which are running users’ jobs. And the green bar represents number of queued jobs. As shown in the figure, users submitting jobs to HTCondor resulted in an increase in total jobs number, as well as an increase in computing resources (VCPUs) number within the upper threshold. On the other hand as jobs gradually completed LHAASO’s computing resource number decreased along with the user jobs number, still no less than the lower threshold set by the resource quota service.

In addition, the CPU utilization of computing resources reached 90.86% within 30 days of operation time. It has been greatly enhanced comparing to a traditional static virtual machine cluster which only has 50% CPU utilization.

6. Conclusions and future plans

In this paper, an elastic resource management system based on Openstack and HTCondor was designed and implemented considering HEP practical application characteristics. The scheduler system is able to adjust virtual CPU core numbers of each HEP application on demand dynamically. Comparing to a static virtual machine cluster, flexible expansion of the resource pool is the most significant feature. At present, the system has been deployed on IHEPCloud. From the actual operation it is obvious that CPU utilization of the cluster has been significantly improved. Now the elastic virtual computing cluster size is not so large. In the future we plan to expand the virtual computing cluster resources and provide computing service for more HEP applications.

Acknowledgments

This work was supported by the National key Research Program of China “Scientific Big Data Management System” (No.2016YFB1000605).
This work was supported by the National Natural Science Foundation of China (NSFC) under Contracts No. 11305192.
This work was supported by the National Natural Science Foundation of China (NSFC) under Contracts No. 11605223.

References

[1] Dayabay experiment, http://dayabay.ihep.ac.cn/twiki/bin/view/Public/
[2] Zhen CAO. A future project at Tibet: the large high altitude air shower observatory (LHAASO)[J]. Chinese physics C, 2010, 34(2): 249.
[3] BESIII experiment, http://bes3.ihep.ac.cn/
[4] Foster I, Zhao Y, Raicu I, et al. Cloud Computing and Grid Computing 360-Degree Compared[J]. Grid Computing Environments Workshop Gce, 2009, 5:1-10.
[5] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization[C]//ACM SIGOPS Operating Systems Review. ACM, 2003, 37(5): 164-177.
[6] Berzano D, Blomer J, Buncic P, et al. Lightweight scheduling of elastic analysis containers in a competitive cloud environment: a Docked Analysis Facility for ALICE[C]//Journal of Physics: Conference Series. IOP Publishing, 2015, 664(2): 022005.
[7] CERN openstack, http://cern.ch/openstack
[8] Milojičić D, Llorente I M, Montero R S. OpenNebula: A Cloud Management Tool[J]. IEEE Internet Computing, 2011, 15(2).
[9] Corradi A, Fanelli M, Foschini L. VM consolidation: A real case based on OpenStack Cloud[J]. Future Generation Computer Systems, 2014, 32(1):118-127.
[10] Qumranet A K, Qumranet Y K, Qumranet D L, et al. kvm: the Linux virtual machine monitor[J]. Proc Linux Symposium, 2007.
[11] Josephsen D. Building a monitoring infrastructure with Nagios[M]. Prentice Hall PTR, 2007.
[12] Sacerdoti F D, Katz M J, Massie M L, et al. Wide area cluster monitoring with ganglia[C]//Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on. IEEE, 2003: 289-298.
[13] Staples G. TORQUE Resource Manager[C]// ACM/IEEE Sc2006 Conference on High Performance Networking and Computing, November 11-17, 2006, Tampa, Fl, Usa. 2006:8.
[14] Fienen M N, Hunt R J. High-throughput computing versus high-performance computing for groundwater applications.[J]. Groundwater, 2015, 53(2):180-4.
[15] Balcas J, Belforte S, Bockelman B, et al. Using the glideinWMS System as a Common Resource Provisioning Layer in CMS[J]. J.phys. Conf.ser, 2015, 664(6).
[16] Goto Y. Kernel-based virtual machine technology[J]. Fujitsu Scientific and Technical Journal, 2011, 47: 362-368.
[17] HEPSPEC06 benchmark[OL]. http://w3.hepix.org/benchmarks/doku.php/
[18] Huang Q L, Sha LI, Cheng Y D, et al. Performance Optimization and Application of kvm in HEP Computing Environment[J]. Computer Science, 2015, 42(1).
[19] Mao M, Humphrey M. A Performance Study on the VM Startup Time in the Cloud[C]// IEEE, International Conference on Cloud Computing. IEEE, 2012:423-430.