A multi-group and preemptable scheduling of cloud resource based on HTCondor

To cite this article: Xiaowei Jiang et al 2017 J. Phys.: Conf. Ser. 898 092051

View the article online for updates and enhancements.

Related content

- Accessing commercial cloud resources within the European Helix Nebula cloud marketplace
  C Cordeiro, A De Salvo, A Di Girolamo et al.

- Real-time complex event processing for cloud resources
  M Adam, C Cordeiro, L Field et al.

- Geographically distributed Batch System as a Service: the INDIGO-DataCloud approach exploiting HTCondor
  D C Aiftimiei, M Antonacci, S Bagnasco et al.
A multi-group and preemptable scheduling of cloud resource based on HTCondor

Xiaowei Jiang\textsuperscript{1}, Jiaheng Zou\textsuperscript{1}, Yaodong Cheng\textsuperscript{1}, Jingyan Shi\textsuperscript{1}
\textsuperscript{1}Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China

E-mail: jiangxw@ihep.ac.cn

Abstract. Due to the features of virtual machine-flexibility, easy controlling and various system environments, more and more fields utilize the virtualization technology to construct the distributed system with the virtual resources, also including high energy physics. This paper introduce a method used in high energy physics that supports multiple resource group and preemptable cloud resource scheduling, combining virtual machine with HTCondor(a batch system). It makes resource controlling more flexible and more efficient and makes resource scheduling independent of job scheduling.

Firstly, the resources belong to different experiment-groups, and the type of user-groups mapping to resource-groups(same as experiment-group) is one-to-one or many-to-one. In order to make the confused group simply to be managed, we designed the permission controlling component to ensure that the different resource-groups can get the suitable jobs.

Secondly, for the purpose of elastically allocating resources for suitable resource-group, it is necessary to schedule resources like scheduling jobs. So this paper designs the cloud resource scheduling to maintain a resource queue and allocate an appropriate amount of virtual resources to the request resource-group.

Thirdly, in some kind of situations, because of the resource occupied for a long time, resources need to be preempted. This paper adds the preemption function for the resource scheduling that implement resource preemption based on the group priority. Additionally, the way to preempting is soft that when virtual resources are preempted, jobs will not be killed but also be held and rematched later. It is implemented with the help of HTCondor, storing the held job information in scheduler, releasing the job to idle status and doing second matcher.

In IHEP (Institute of high energy physics), we have built a batch system based on HTCondor with a virtual resources pool based on Openstack. And this paper will show some cases of experiment JUNO and LHAASO. The result indicates that multi-group and preemptable resource scheduling is efficient to support multi-group and soft preemption.

Additionally, the permission controlling component has been used in the local computing cluster, supporting for experiment JUNO, CMS and LHAASO, and the scale will be expanded to more experiments at the first half year, including DYW, BES and so on. Its evidence that the permission controlling is efficient.

1. Introduction

In IHEP (Institute of High Energy Physics, Chinese Academy of Sciences), the resources belong to different experiments including BESIII \cite{BESIII}, the DayaBay Reactor Neutrino Experiment \cite{DayaBay}, CMS \cite{CMS}, the Large High Altitude Air Shower Observatory \cite{LHAASO}, the Jiangmen Underground Neutrino Observatory \cite{JUNO} and so on, leading to the resources being isolated. And each experiment has one or more groups, users in one group only can use the resources belonging to the same
group but can not use the resources belonging to other groups. So we must schedule jobs depending on groups.

HTCondor provides the accounting group functions to satisfy the scheduling depending on groups. And for the features of classad and high throughput, we used HTCondor as the batch system of our virtual cluster. However, the HTCondor trusts users so much that users can request the resources belonging to other groups when they set the attribute ”accounting_group” as other groups. So HTCondor can not manage users strictly. Besides, users, groups and experiments must be related by a management system which HTCondor doesn’t provide. So a permission controlling system is necessary.

Around HTCondor, We have V-CONDOR [6] (a dynamic cloud resource management system) to allocate virtual resources to each group. But sometimes, the resources of some groups are occupied for too long time. These resources have to be collected back when the resource requirement of other groups became large. In HTCondor’s preemption, the resources decide whether they are preempted by themselves, through ‘rank’ way. However, there is not a common rank expression suitable for all the preempable situation. So in one case, if we want to allocate resources by HTCondor’s preemption, we still need to find the potential preemptable resources and decide which worker node will be preempted, then change ‘rank’ configuration in every worker node. So we must have a resource queue to maintain a potential preemptable resource sequence, then a component to connect with V-CONDOR to complete preemption operations.

2. Permission controlling component

We designed permission controlling component(PCC) to ensure the resource groups getting the suitable jobs which belong to the owners in these groups. PCC is a component at the front of HTCondor’s schedd. The default HTCondor structure is shown as figure 1.

![Figure 1: default HTCondor structure.](image)

Generally, HTCondors resources are in one share resource pool, and the resources are not divided into different parts. For this case, HTCondor provides the ”accounting group” function to divide resources into different parts. However, HTCondor provides no uniform user information system that automatically sends the user, group and experiments information to the HTCondor. Moreover, the users can freely set his/her groups in the job classad leading to users can use the resources belonging to other groups. So we added the PCC to solve it. PCC’s structure is shown as figure 2.

As the show of figure 2 in PCC, through accounting group, there are two types of group, including user-group and resource-group. User-group indicates which user belonging to which
group, and resource-group indicates which resources belonging to which groups. Then, the user only can use the resources in the same groups. So we developed the management system to manage the users, groups and experiments information in PCC, then added the controlling function which can check the permission and submit jobs by python bindings.

3. Preemptable resource scheduler
For the purpose of elastically adjusting the amount of resources for suitable resource-group, it is necessary to schedule resources by maintaining a resource queue. And the main goal is to implement the preemption function when there is resource competition. In HTCondor, which resources can be preempted would be decide through machine rank. So we need to choose which resources would be preempted, then change the rank on these machines, because there is no common rank expression for all resources. whether the HTCondor preemption is utilized, which resources can be preempted would be decided by ourselves. Given this, this paper designed a preemptable resource scheduler(PSR) which can decide the sequence of potential preemptable resources and complete preemption operations with V-CONDOR.

Due to the dynamic feature of virtual resources, the virtual resources would be deleted or allocated to new jobs by V-CONDOR immediately, when jobs complete. So theoretically, it is unnecessary to consider the empty resources in the resources queue. Additionally, for the consideration of efficiency, the resource queue must be simplified. So PSR only focuses on the maintenance of the occupied resources by the running jobs, that PSR only need to synchronize the Busy resources with condor schedd.

The figure 2 shows the workflow of PRS. PRS keeps the synchronizing resource information with the HTCondor schedd, and real-timely sorts the resources based on priority. When some resources need to be preempted, virtual resource preemption pulls out the resources from queue as the policy of priority FIFO. Then the potentially preempted resources will be check whether there are jobs running on them: if one resource is occupied by job, operate (hold and rematch) the job and send the resource to V-CONDOR, then V-CONDOR will reallocate the resource to
other groups; if not, return the resource to virtual resource preemption, then virtual resource preemption will pull out one new resource.

4. Preemptable policy
In PRS, priority decides the possibility of preemption. So the calculation of priority is the key of PRS. In our environment, some nice group have more possibility to using resource and the resource occupied for shorter time have more possibility to be preempted. So the priority consists of initial group priority and cumulate priority. The priority equation is shown as follow.

\[ \text{resource\ priority} = \alpha_{\text{prio}} + k \times \text{time\ occupied} \]  

\( \alpha_{\text{prio}} \) presents the initial group priority depend on the importance of groups. \( k \times \text{time\ occupied} \) presents the cumulate priority which the lower value causes more possibility of preemption and \( \text{time\ occupied} \) presents the time which resource is occupied. In our environment, we set each initial group priority by interval 1000. For the purpose of reduce the effect by the cumulate priority, the \( k \times \text{time\ exec} \) must be less than 1000.

\[ k = \frac{1000}{\text{time\ estimate}} \]  

\[ \text{resource\ priority} = \alpha_{\text{prio}} + 1000 \times \frac{\text{time\ occupied}}{\text{time\ estimate}} \]  

The resource priority is the equation 3, and \( \text{time\ estimate} \) is an estimate value of the max occupied time.

5. Results Heading
The permission controlling component(PCC) have been used in the HTCondor cluster of IHEP, supporting for experiment JUNO, CMS, LHAASO etc. The statistics of completed jobs in July 2016 is shown as table [1].

Table [1] shows the jobs in different groups are completed normally with PCC. And most of the jobs are simulation jobs. As an example, there are 1051918 juno jobs completed on the resources belonging to JUNO, totally taking 205944.1 hours.

The preemptable resource scheduler(PSR) with the priority policy in this paper have been tested effectively. We simulated 4000 jobs with different submitting time, different executing time and three groups including LHAASO, BES and JUNO. Among these three experiments, BES’s initial group priority is 3000.0, LHAASO’s is 2000.0, JUNO’s is 1000.0. \( \text{time\ estimate} \) is set as 2080. And the queuing time of each group is shown as table [2].
Table 1: Jobs Statistics of IHEP in 20160701-20160731

| Experiment | Group | Completed jobs | Total walltime (h) |
|------------|-------|----------------|-------------------|
| JUNO       | juno  | 1051918        | 205944.1          |
| CMS        | cms   | 270298         | 9252.3            |
| LHAASO     | lhaaso| 28504          | 172880.0          |

Table 2: Queuing Time without PRS and with PRS

| Experiment | Without PRS (s) | With PRS (s) | Group priority |
|------------|----------------|--------------|----------------|
| BES        | 60052.0        | 130.0        | 3000           |
| LHAASO     | 71529.0        | 132.0        | 2000           |
| JUNO       | 86627.0        | 365636.0     | 1000           |

As table 2 showing, BES and LHAASO jobs with PRS take shorter queuing time than without PRS, which indicates BES and LHAASO jobs preempt the resources belonging JUNO. As an example, the queuing time of BES jobs is average 60052 seconds without the priority policy in PRS. But with PRS, due to the higher group priority than LHAASO an JUNO, the BES jobs preempt resources from LHAASO and JUNO. So the BES jobs take shorter queuing time than before.

6. Conclusions
In this proceeding, we show two components including permission controlling component (PCC) and preemptable resource scheduler (PRS) based on HTCondor. PCC is consist of a management system and a checking module. PRS focuses on the cloud resources preemption as a component around V-CONDOR. The simulation results shows the effectiveness of the priority policy in PRS. However, there are still several problems to be modified such as the design of priority policy.

Acknowledgements
This work was supported by the National Natural Science Foundation of China (No.11575223, No.11605221 and No.11475210).

References
[1] Zhen-An Liu. Status of BEPCII/BESIII project. In Accelerator and particle physics. Proceedings, 9th Winter Institute, APPI 2004, Appi, Japan, February 16-20, 2004, pages 86–90, 2004.
[2] F. P. An et al. The muon system of the Daya Bay Reactor antineutrino experiment. Nucl. Instrum. Meth., A773:8–20, 2015.
[3] Univ Pisa Scuola Normale Super Pisa Pisa Tenchini, R. The CMS experiment at the CERN LHC. JOURNAL OF INSTRUMENTATION, 3, 2008.
[4] G. Di Sciascio. The LHAASO experiment: from Gamma-Ray Astronomy to Cosmic Rays. Nucl. Part. Phys. Proc., 279-281:166–173, 2016.
[5] C. Jollet. The JUNO experiment. Nuovo Cim., C39(4):318, 2017.
[6] Zhenjing Cheng Tao Cui. Haibo Li, YaoDong Cheng. Virtual Cluster Computing in IHEPCloud. HEPiX Spring 2016, 2016.