A Multi-group and Preemptable Scheduling of Cloud Resource Based on HTCondor

X.W.Jiang, J.H.Zou, Y.D.Cheng, J.Y.Shi
Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China

Introduction

Virtual machines have many features ął flexibility, easy controlling and customized system environments. More and more organizations and enterprises deploy virtualization technology or cloud computing to build the distributed system with the virtual resources. Cloud computing is widely used in high energy physics.
We design and implement a method used in high energy physics that supports multiple resource groups and preemptable resource scheduling policy based on virtual machines and HTCondor – a high throughput computing system. It makes resources control more flexible and more efficient.

Motivation

In IHEP, the resources belong to different experiments, and each experiment has one or more groups. Users in one group only can use the resources belonging to the same group. So we must schedule jobs depending on groups.
We have V-CONDOR(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 if the resource requirement of other groups became large. So the appropriate preempting policy is very important for collecting resources back. So we design and implement two components for multi-group scheduling and preemptible resource scheduler.

Multi-group Scheduling

we designed permission controlling component(PCC) to ensure the different resource groups getting the suitable jobs which the job owner's user-group must be permitted to running jobs on the resources.
PCC is a component at the front of HTCondor's schedd. The HTCondor's default structure and the structure with PCC are shown as figure 1.

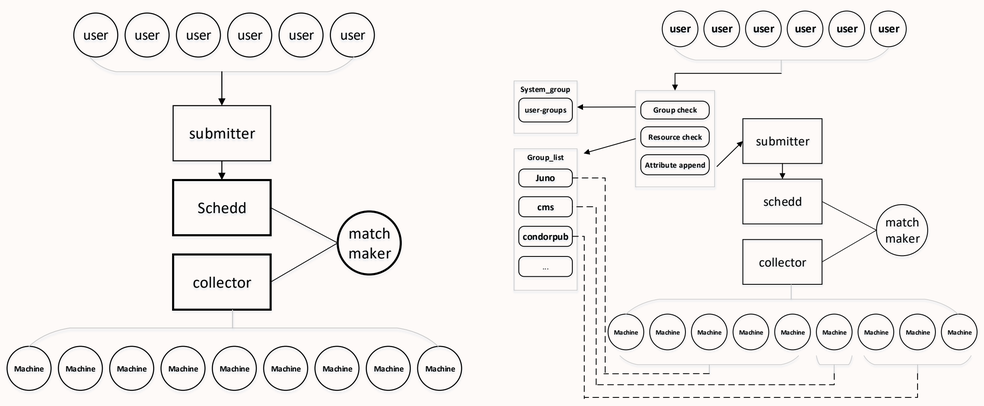

Figure 1: HTCondor (left); HTCondor with PCC(right)

PCC have three parts. one is group check for conforming user's group are exist in the system-group; second is resource check for checking whether there are available resources in the resource group which maps to user group; third is attribute append for parsing and formatting the job classad attributes, then append the formatted classad to htcondor schedd. Util now, the job submitted by PCC declares the resources in which group it wants.

Preemptable Resource Scheduler (1)

Preemptable resource scheduler (PRS) maintenances a virtual resource queue. All resources in the resource queue have their own priority which decides their order in the queue. The resource with lower priority will be preempted more possibly. The work flow of PRS is shown as figure 2.

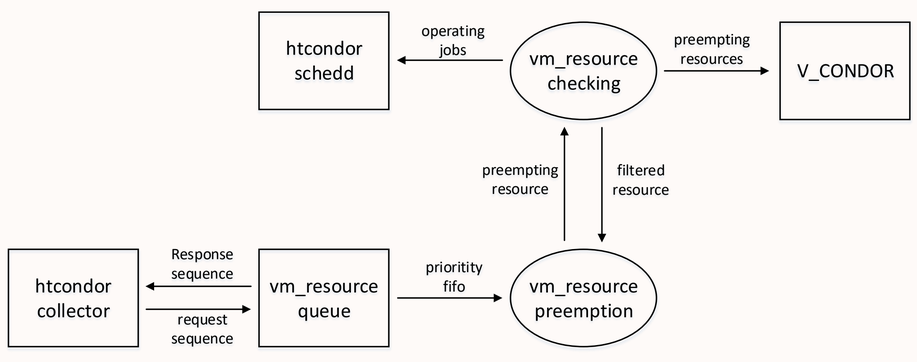

Figure 2: preemptable resource scheduler

Preemptable Resource Scheduler (2)

PRS keeps the synchronizing resource information with the HTCondor collector, and real-timely sorts the resources based on priority. When some resources need to be preempted, vm_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 vm_resource preemption, then vm_resource preemption will pull out one new resource. 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.

$$resource_{priority} = \alpha_{prio} + k * time_{occupied} \quad (1)$$

α_{prio} presents the initial group priority depend on the importance of groups. $k * time_{occupied}$ presents the cumulate priority which the lower value causes more possibility of preemption and the $time_{occupied}$ 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 $k * time_{exec}$ must be less than 1000.

$$k = 1000/time_{estimate} \quad (2)$$

$$resource_{priority} = \alpha_{prio} + 1000 * time_{occupied}/time_{estimate} \quad (3)$$

The resource priority is the equation 3, and $time_{estimate}$ is an estimate value of the max occupied time.

Results Heading

The permission controlling component have been used in the HTCondor cluster of IHEP, supporting for experiment JUNO, CMS, LHAASO etc. The statistics of completed jobs during 20160701-20160731 is shown as table 1.

Experiment	Completed jobs	Total walltime (h)
JUNO	1051918	205944.1
CMS	270298	9252.3
LHAASO	28504	172880.0

Table 1: Jobs Statistics of IHEP in 20160701-20160731

The preemptable resource scheduler have been tested effectively. We simulated 4000 jobs with different submitting time and different executing time and three groups including LHAASO, BES and JUNO. Among the three experiments, BES's initial group priority is 3000.0, LHAASO's is 2000.0, JUNO's is 1000.0. $time_{estimate}$ is set as 2080. And the queuing time of each group is shown as table 2.

Experiment	Without PRS (s)	With PRS (s)
BES	60052.0	130.0
LHAASO	71529.0	132.0
JUNO	86627.0	365636.0

Table 2: Queuing Time without PRS and with PRS

As table 2 showing, scheduling BES and LHAASO jobs with PRS take shorter time to idle than without PRS, that coincides with the higher importance of BES and LHAASO.

Conclusion

The results indicate that permission controlling component and preemptable resource scheduler can complete to controlling the cloud resources and users by groups and make resource allocation more flexible by self-defined resource priority policy. It has been used in some experiments and we plan expand the scalability soon.