An automatic solution to make HTCondor more stable and easier

Jingyan Shi[1], *Jiaheng* Zou[1], *Qingbao* Hu[1], *Xiaowei* Jiang[1], *Ge* Ou[1]

[1]Computing Center, Institute of High Energy Physics, Chinese Academy of Science

Abstract. HTCondor has been widely adopted by HEP clusters to provide high-level scheduling performance. Unlike other schedulers, HTCondor provides loose management of the worker nodes. We developed a maintenance automation tool called “HTCondor MAT” that focuses on dynamic resource management and automatic error handling. A central database records all worker node information, which is sent to the worker node for the startd configuration. If an error happens for the worker node, the node information stored in the database is updated and the worker node is reconfigured with the new node information. The new configuration stops the startd from accepting error-related jobs until the worker node recovers. The MAT has been deployed in the IHEP HTC cluster to provide a central way to manage the worker nodes and remove the impacts of errors on the worker nodes automatically.

1 Introduction

The Institute of High Energy Physics in China runs a 17,000 CPU core HTCondor cluster supporting more than 10 HEP experiments such as BESIII[1], LHAASO[2], JUNO[3], ATLAS[4], CMS[5], etc. Each experiment has its storage file system for the experiment data and this is only accessible to the experiment’s user. Scratch, AFS, and CVMFS are the public shared file systems provided by the computing centre to all users. Puppet[6] is the tool that automates software installation and upgrades. Nagios[7] monitors every service running on the servers and worker nodes.

Each experiment has its own “Linux group”. which is assigned to the attribute “AcctGroup” of every job. The HTCondor daemon “startd” has the attribute “START”, and its value is used to select the job for the worker node. The experiments’ “Linux group” is set to the “START” attribute by the administrator based on the experiment’s requirements. Most of the worker nodes are shared by all the experiments and a small part of them are set to run some dedicated tasks. This is done by adjusting the worker node attribute “START”.

An unexpected error that happens to the worker node might cause a black hole worker node to cause the queueing job to fail. This means that the job fails as soon as it is dispatched to the error worker node. A large number of jobs would fail in a short time at the same worker node due to the same error. Different errors cause different types of jobs to fail. It is important to identify and block the related jobs to be dispatched to the error worker nodes. After the worker node is recovered, the block should be removed automatically.

The ClassAds language in HTCondor provides an extremely flexible and expressive framework [8] making HTCondor have high-level scheduling performance, but its simple advertisement policy leads to less central management functions to the worker nodes. The worker node configuration file is manually modified one by one by default. We developed a maintenance automation tool called "MAT" that automatically adjusts the worker node attribute to stop accepting the jobs that would fail due to an error.

2 Design

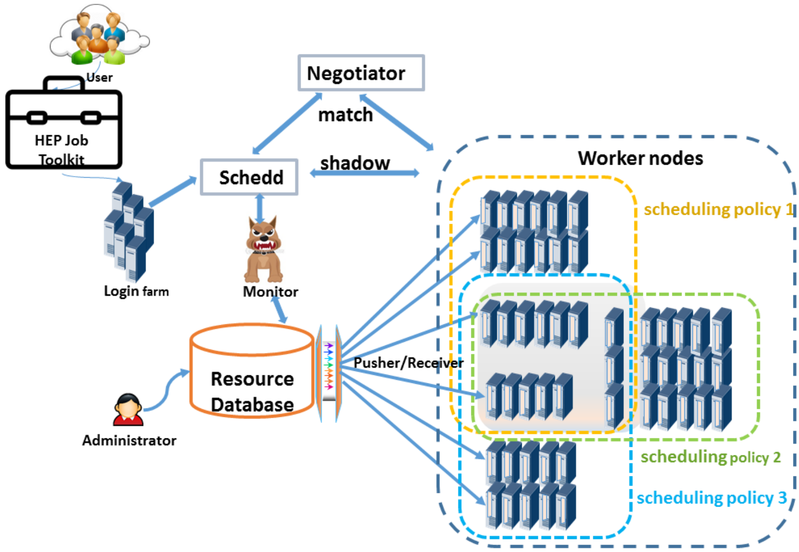

Fig. 1. MAT architecture.

The design is shown in Figure 1. The resource database stores the information of all the worker nodes, such as the memory size, the machine age, and the type of experiment job the worker node accepts. Both software and hardware errors are classified based on the failures of the different experiments' jobs. The classification and related jobs are also stored in the database.

The HEP Job Toolkit is developed based on the HTCondor Python API[9] and provides user job management functionality. Additionally, the MAT uses it to add attributes to the job that are necessary for the new scheduling policy.

The web portal is a GUI for the administrator to adjust the worker nodes' attributes stored in the resource database. The administrator modifies the worker node attribute via the web portal instead of via manual modification. The web portal provides statistics functionality as well.

Nagios is deployed on the cluster as the main monitoring tool. The monitoring dog analyses the monitoring results and returns the "experiment list" of the worker nodes. The jobs belonging to the experiment in the list should not be dispatched to the malfunctioning worker node. The experiment list is switched to the "Linux group list" by the monitoring dog.

The pusher sends the blocked “Linux group list” to each malfunctioning worker node. The receiver is the program running at each worker node that gets the message from the pusher. The receiver reconfigures the “startd” service with the newly generated configuration file based on the “Linux group list” message that is received from the pusher.

The workflow of the MAT is as follows:

Step 1

Case 1: The administrator changes the worker nodes’ information saved in the database via the web portal.

Case 2: As soon as an error is detected by Nagios, the monitoring dog analyses the error and provides a “Linux group list” whose jobs are related to the error. The changes to the “Linux group list” of each worker node are stored in the resource database.

Step 2

Once the worker node information is changed in the database, the monitoring dog calls the pusher to send the new “Linux group list” message to the worker node. After the receiver running at each worker node gets the message of the worker node, it generates a new “startd” configuration file and reconfigures the “startd” service. A job that does not belong to the “Linux group” set in the startd configuration file is unacceptable to the node.

All the processes are transparent to the user. The period from the time that the error happens to the time that “startd” reconfiguration is finished is a few seconds. It is short enough to avoid the worker node becoming a black hole.

3 Approach

3.1 Resource Database

MySQL is used for the database. Figure 2 is the Entity Relationship diagram of the database. Three kinds of tables are defined. The first kind of table records the basic information of the experiments, Linux group, error, and worker nodes. The second type of table includes the relationships defined among the tables, such as the Linux groups that belong to the experiment, the relationships between the experiment jobs, and the error. The content of the first and the second tables are the static information that is added and modified by the administrator manually via the web portal. The third type of table stores the current attributes of each worker node, the history of the worker node attributes that changed, and all the database logs. Unlike the first two kinds of tables, the content of the third kind of table is changed dynamically based on the cluster’s running status and the scheduling policy deployed.

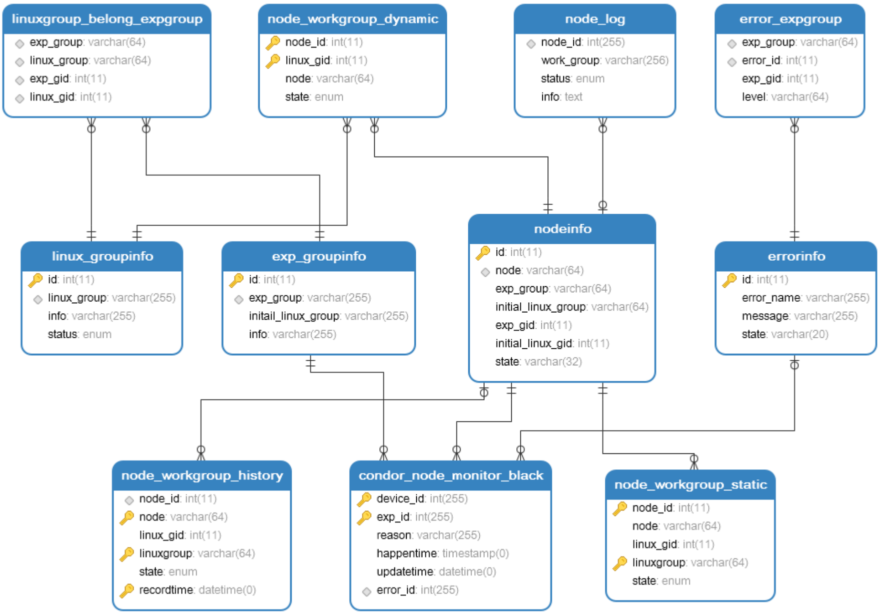

Fig. 2. E-R diagram of the resource database.

There are two ways to modify the resource database. One way is that the administrator uses the web portal to adjust the experiment jobs that they would such as the worker node such as to run. The table "node_workgroup_static" saves all the modifications from the administrator. The other way is the monitoring dog. Once an error happens to a worker node, the monitoring dog analyses the error and provides the "Linux group list" of the jobs that could not run correctly on the node due to the error. The "node_workgroup_dynamic" table is modified automatically based on this list.

3.2 Monitoring Dog

The monitoring dog quickly responds to an error that happens to a cluster. Nagios monitors every service running on the worker node. All the monitoring results from Nagios are written into a disk file. Filebeat is a lightweight open source shipper that stores the monitoring status file. The monitoring dog analyses the file and gets the "error-related Linux group" from the resource database. The "error-related Linux group" indicates the related jobs that should not be dispatched to the error worker node. Apache Spark is a unified computing engine and a set of libraries for massive parallel data processing on computer clusters. All the analysis work is done on the IHEP Apache Spark platform. The monitoring dog removes the "error-related Linux group" from the group list set at the worker node. Before the modification, the monitoring dog compares the current status with the status saved in the database. Only a new error needs to be recorded in the database. If an error is corrected, the monitoring dog adds the corresponding Linux group into the "acceptable Linux group" to the worker node again. As soon as Nagios generates the new status file, the monitoring dog starts its analysis.

3.3 Pusher & Receiver

The pusher sends the new attributes to the worker node. At first, we published all the worker node attributes through an httpd server. Each worker node fetches its attributes via a URL address that has the worker node's IP. A crontab task set at each worker node fetches the node attributes every 5 minutes. Once an error happened at the worker node, the longest time to remove the malfunctioning worker node from the cluster was 5 minutes. Although the error could be removed, there was still a 5-minute black hole and that could cause a large number of job failures.

The optimized way to address this is to push the error message to the worker node from the monitoring dog. NRPE is the protocol used to send the message to the worker node used by Nagios. The monitoring dog calls the pusher to send the node information via NRPE to the worker nodes whose information is changed in the database by the monitoring dog or the administrator. We use coroutine-based concurrency with the non-blocking IO mode to send messages via NRPE to hundreds of worker nodes simultaneously. The message to be sent is a long string. To control the length of the message, the string is composed of the Linux group sequence number, which is defined in the resource database. After the message is sent, a finished flag would be written into the database.

The receiver is the program running on each worker node. After the message string is received from the pusher via NRPE, the receiver switches the message to the "Linux group list" and reconfigures the "START" attribute of the worker node. The pusher/receiver based on the NRPE protocol makes the worker node respond to the error in a few seconds. The startd is reconfigured in "peaceful" mode, which does not kill the running job and makes the configuration available to the new incoming jobs.

3.4 HEP Job Toolkit

HTCondor has a Python module that provides binding to client-side APIs for HTCondor and the ClassAd language. The HEP Job Toolkit is developed for user job management functionality based on the module. A user runs a HEP Job Toolkit command with some simple parameters to submit a job instead of preparing a job 'submit file'. This decreases the user's preparation work. In addition, the HEP Job Toolkit helps the administrator to deploy a new scheduling policy. The new attribute corresponding to the job start-condition set at the worker node could be easily added to the HEP Job Toolkit and this is transparent to the user.

3.5 Web Portal

The MAT provides the administrator with a tool that can be used to manipulate the worker node manually in an easy way. A bunch of attributes of the worker nodes saved in the resource database can be modified via the web portal. All the modification history is logged into a log table of the resource database. Some query functions are provided.

The web portal is developed based on a microservice. The architecture is integrated with Spring Boot 2.0[10]. We use Apache Shiro[11] for the authorization, MyBatis[12] to implement the SQL database access, and Ehcache for caching. In addition, it provides RESTful web service interfaces to the underlying database, which provides an easy way to use the API.

4 Deployment

We choose MySQL 5.7 for the resource database running on a dedicated machine. The machine runs Scientific Linux 6.9. The web portal is deployed using Apache Tomcat 8.0.37 and Spring Boot 2.1.7 running on the same machine as the resource database.

Nagios has been the main monitoring tool run by HTCondor for many years. The current version is 4.3.4. The Filebeat is set at the Nagios server to collect the Nagios monitoring results file and sends it to the Spark[13] analysis platform. It is important to filter the “OK” status using Filebeat and only “warning” and “critical” errors are streamed into the Spark analysis platform. This reduces the number of status messages from 30,000 to less than 100.

The monitoring dog is submitted as a loop task to the Spark analysis and continually analyses the error.

The pusher is submitted as a loop task of the Apache Spark platform. It rewraps the information in a simple short string and sends it to the receiver via NRPE protocol. It is called by the monitoring dog once the new attributes of the worker nodes are available.

The NRPE support belongs to Nagios and does not need extra deployment work.

The receiver is the program deployed with the HTCondor package to each worker node by a puppet. It keeps listening for a message from the pusher and reconfigures the worker node startd service. The “condor” user runs the receiver so that the startd service can be restarted correctly if necessary. As soon as the receiver gets the string from the pusher, it switches it into the START attribute of the startd and upgrades the startd configuration file.

The HEP Job Toolkit is a set of Python programs that provide user job management functions, such as job submission, job query, etc. It is stored in a cvmfs file system.

5 Conclusion

The MAT has been running at the IHEP HTCondor cluster for several months. It gives the administrator a convenient way to manage HTCondor worker nodes centrally. The MAT provides a mechanism to adjust the scheduling policy. The “Linux group” of the job accessible to each worker node can be adjusted via the MAT by the administrator conveniently. Other new scheduling policies such as “long job” and “fast job” can be deployed by adding new attributes to both jobs and worker nodes. The error response time is reduced to a few seconds by the MAT and no black holes appear in the cluster anymore.

6 Acknowledgement

We would like to thank all of our colleagues for their contributions. The work presented is supported by the National Nature Science Foundation of China “Container Virtualization Applied to High Energy Physics Computing” (No.11775250), the National Science Foundation of China “The Two-staged Job Scheduling Algorithms and Resource Management Research about Workload integration between HTC Cluster and HPC Cluster” (No. 11805225), and the Youth Innovation Promotion Association of the Chinese Academy of Sciences.

References

1. BESIII experiment. [online] Available at: http://bes3.ihep.ac.cn [Accessed 3 Oct. 2020]
2. LHAASO experiment. [online] Available at: http://english.ihep.cas.cn/lhaaso/index.html [Accessed 2020-10-03]
3. JUNO experiment. [online] Available at: http://juno.ihep.cas.cn [Accessed 2020-10-03]

4. The ATLAS Collaboration, The ATLAS Experiment at CERN LHC, JINST 3, (2008) S08003
5. CMS Collaboration, The CMS experiment at the CERN LHC, JINST 3 (2008) S08004
6. Puppet Open Source. [online] Available at: http://puppetlabs.com/puppet/puppet-open-source [Accessed 2020-10-03]
7. Syed Ali Monitoring with Nagios and Trend Analysis with Cacti, pages 183, 2014
8. Douglas Thain, Todd Tannenbaum, and Miron Livny, Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, 2005. "Distributed Computing in Practice: The Condor Experience"
9. HTCondor project, “HTCondor Python API”, [online] Available at : https://htcondor.readthedocs.io/en/latest/apis/python-bindings/api/htcondor.html [accessed 2020-06-15]
10. Spring Boot 2.0 [software], version 2.3.1, [online] Available at: https://spring.io/projects/spring-boot [accessed 2020-06-15]
11. Apache Shiro, [software] [online] Available at : https://shiro.apache.org/ [accessed 2020-06-15]
12. Mybatis, [software], version 3.5.5, [online] Available at: https://github.com/mybatis/mybatis-3/tree/master/src/site [access 2020-06-15]
13. Diogo Castro, Prasanth Kothuri1, Piotr Mrowczynski1, Danilo Piparo1, and Enric Tejedor 23rd International Conference on Computing in High Energy and Nuclear Physics (CHEP 2018) Volume 214, 2019 “Apache Spark usage and deployment models for scientific computing”