A web workbench system for the Slurm cluster at IHEP

Ran Du[1,*], *Jingyan* Shi[1,**], *Xiaowei* Jiang[1,***], and *Chaoqi* Guo[1,****]

[1]Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China, 100049

Abstract. Slurm REST APIs are released since version 20.02. With those REST APIs one can interact with slurmctld and slurmdbd daemons in a RESTful way. As a result, job submission and cluster status query can be achieved with a web system. To take advantage of Slurm REST APIs, a web workbench system is developed for the Slurm cluster at IHEP. The workbench system consists with four subsystems including dashboard, tomato, jasmine and cosmos. The dashboard subsystem is used to display cluster status including nodes and jobs. The tomato subsystem is developed to submit special HTCondor glidein jobs in the Slurm cluster. The jasmine system is used to generate and submit batch jobs based on workload parameters. The cosmos subsystem is an accounting system, which not only generates statistical charts but also provides REST APIs to query jobs. This paper presents design and implementation details of the Slurm workbench. With the help of workbench, administrators and researchers can get their work done in an effective way.

1 Introduction

The Slurm cluster at IHEP is founded in 2017. As the number of applications, resources and users are increased year by year, the Slurm cluster scales up gradually. By the middle of 2023, the Slurm cluster has provided parallel and heterogeneous computing services to nearly 20 applications and 230 users with 10000 CPU cores and 250 GPU cards approximatly.

To better support a cluster at such a scale, a web workbench system based on Slurm REST APIs is developed. Slurm REST APIs are released since version 20.02. With those REST APIs one can interact with slurmctld and slurmdbd daemons in a RESTful way. As a result, job submission and cluster status query can be achieved with web systems. The workbench system consists with four subsystems: dashboard, tomato, jasmine and cosmos. We will present design and implementation of these four subsystems in this paper. The paper is organized in the following order. Section 2 presents related works and section 3 gives POST and GET examples on Slurm REST APIs. Section 4 presents design and implementation of the four subsystems, and current status and conclusion can be found in section 5 and section 6 respectively. In section 7, we express our acknowledgement to fund projects.

*e-mail: duran@ihep.ac.cn
**e-mail: shijy@ihep.ac.cn
***e-mail: jiangxw@ihep.ac.cn
****e-mail: guocq@ihep.ac.cn

2 Related Works

The idea of developing web-based systems for Slurm clusters is not new. Many work is done on this topic. Comparing with Slurm REST API, FirecREST[1] is another REST API used to submit jobs and move data for the CSCS supercomputing center. The Cornell University Center for advanced computing (CAC) developed a CJRS[2] tool to provide web based interactive computing environment for Slurm jobs. Matthew Li et al.[3] presents a system consisted of Slurm plugins and an HTTP API to enforce cluster usage policy based on accounting statistics. Besides, some open source projects are trying to provide a common solution system for slurm clusters. For example, the SCOW[4] developed by Peking University is a web-based portal and management system for HPC users and administrators, and the slurm-web[5] tries to develop more functionalities in addition to dashboard. From the above projects we can find that it's preferable to have a web-based system for HPC clusters. Moreover, although some projects tried to provide common solution to web-base systems, there are normally customized solutions for different clusters. Take our case for example, the dashboard and cosmos(accounting) subsystems can be used as a common solution, but the tomato and jasmine subsystems are developed based on special administrative requirements.

3 Slurm REST API

Slurm provides a new daemon slurmrestd in addition to the original slurmctld and slurmdbd daemons. As shown in figure 1, slurmrestd is located between clients and the other two daemons, so that users can send requests to slurmctld and slurmdbd daemons through slurmrestd with Slurm REST APIs.

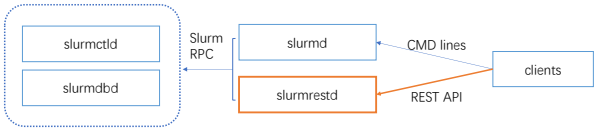

Figure 1. Slurmrestd and Slurm REST API.

Based on which daemon slurmrestd interacts with, Slurm REST APIs can be classified into two categories as shown in table 1. The URLs started with 'slurm' indicates that slurmrestd is talking to slurmctld daemon, while the others started with 'slurmdb' means that the slurmrestd is interacting with slurmdbd daemon. Because there are multiple versions of Slurm support REST API, one can replace the <version> with the right string according to the version of Slurm. Table 1 only lists part of Slurm REST APIs, the full list can be found on the Slurm official web site[6]. To use Slurm REST APIs, user name and JWT token must be provided with headers of X-SLURM-USER-NAME and X-SLURM-USER-TOKEN respectively, and the value of X-SLURM-USER-TOKEN can be generated with scontrol command.

4 Design and Implementation

The workbench system is implemented based on the Slurm GET and POST REST APIs shown in Figure 2. As mentioned earlier, the workbench as the client, is able to interact with slurmctld and slurmdbd through slurmrestd. As a result, administrators and researchers can work on the Slurm cluster with the help of workbench. The workbench system consists with dashboard, tomato, jasmine and cosmos. The frontend interface of these four subsystems

Table 1. Examples of Slurm REST APIs.

Index	Slurm REST APIs interacted with slurmctld	Slurm REST APIs interacted with slurmdbd
1	GET /slurm/<version>/jobs	GET /slurmdb/<version>/jobs
2	GET /slurm/<version>/job/{job_id}	GET /slurmdb/<version>/job/{job_id}
3	GET /slurm/<version>/nodes	GET /slurmdb/<version>/accounts
4	POST /slurm/<version>/job/submit	POST /slurmdb/<version>/accounts
5	DELETE /slurm/<version>/job/{job_id}	DELETE /slurmdb/<version>/user/{user_name}

is implemented under the Python Flask framework, while the backend is implemented with Python. Some historic data is required for long-term store and MariaDB is adopted to save these data.

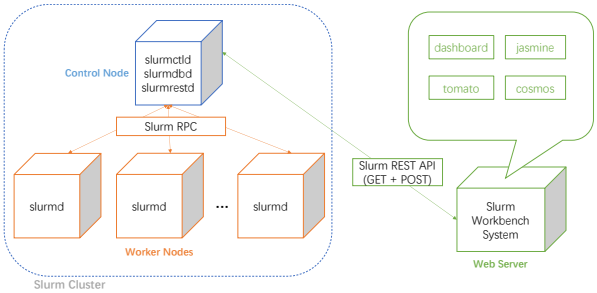

Figure 2. Architecture of the Slurm workbench system.

4.1 Tomato

The tomato subsystem is used to submit HTCondor glidein jobs to the Slurm cluster. There are two batch clusters at IHEP, one is the HTC cluster whose scheduler is HTCondor, the other is the HPC cluster whose scheduler is Slurm. To make resource utilization of the Slurm cluster more effective, some HTC jobs will be scheduled and run in the Slurm cluster.

To run HTC jobs in the Slurm cluster, glidein jobs are submitted to the Slurm cluster to start HTCondor startd daemons. With startd daemons running, the Slurm worker nodes get registered in the HTCondor cluster, so that HTC jobs could be scheduled and run in these nodes. Figure 3 shows the submission interface. When the generate button is clicked, a Slurm POST REST API will be called to submit glidein jobs.

4.2 Jasmine

Subsystem jasmine can be used to generate and submit batch jobs to the Slurm cluster according to specified parameters. We use jasmine to do an overall test during summer maintenance and test Slurm cluster configuration. Similar with the tomato subsystem, the key functionality of the jasmine subsystem is to submit batches of jobs. Parameters of user, time points and parallelism degree can be specified in the form shown in figure 4.

4.3 Dashboard

After glidein jobs or workload jobs are submitted to the Slurm cluster, the dashboard subsystem will be used to display Slurm cluster status including nodes, jobs and users. Figure

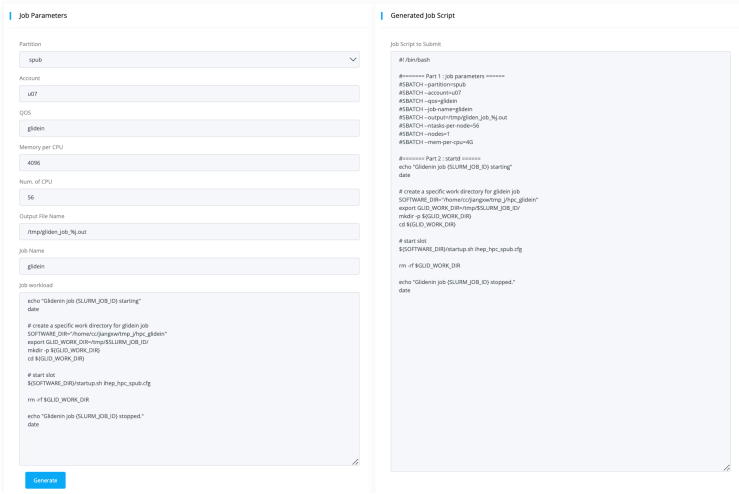

Figure 3. Submit glidein jobs with the tomato subsystem.

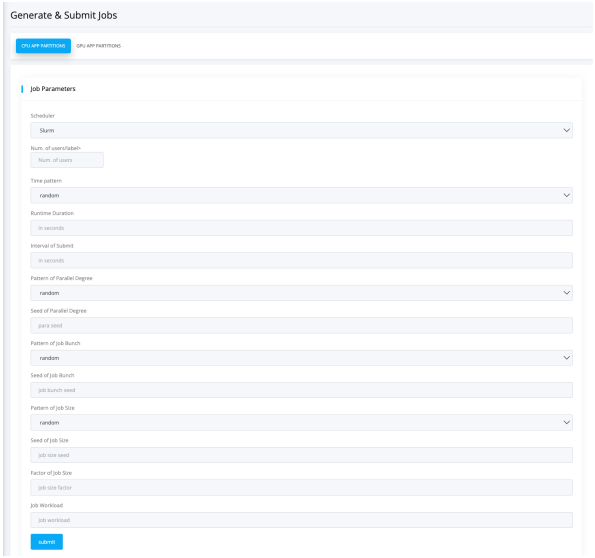

Figure 4. Submit job batch with jasmine.

5 shows the overview of dashboard subsystem, all the information are collected with Slurm GET REST APIs.

4.4 cosmos

The cosmos subsystem is a cluster accounting system. Because accounting is more complex comparing with other three subsystems, only part information are get with Slurm REST APIs, meanwhile, other data resources are also adopted. Figure 6 shows an example chart generated by cosmos, which displays the statistics of weekly jobs from multiple groups.

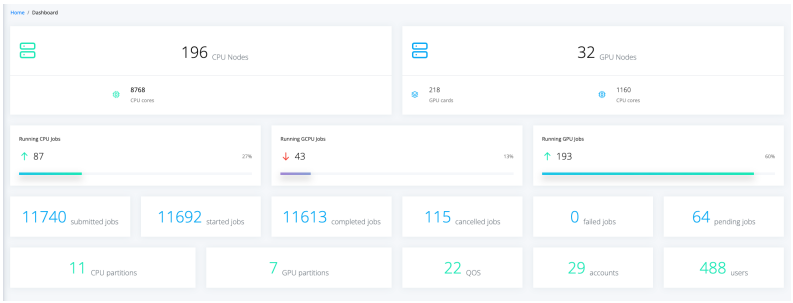

Figure 5. Overview of the dashboard subsystem

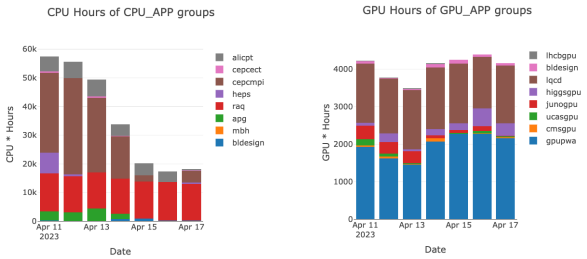

Figure 6. Sample weekly job statistics generated by the cosmos subsystem.

5 Workbench Status

The workbench prototype has been implemented. Here with the word prototype it means that the Slurm REST APIs are adopted by four subsystems of the workbench. Some subsystems are already used in production, for example, the dashboard and cosmos subsystems, while the others will have more functionalities developed based on requirements. Besides, we are trying to deploy the workbench system in containers, so that the workbench can be run everywhere even if the host server is down for some reason.

6 Conclusion

Slurmrestd daemon provides REST APIs for clients to interact with slurmctld and slurmdbd daemons. We implement a workbench system using Slurm REST APIs so that administrators and researchers can manipulate the Slurm cluster in an easy way. In this paper we make a brief introduction to the Slurm REST APIs and describe four subsystems of the workbench system implemented with Slurm REST APIs. Some subsystems are in production, but others still need more development. Besides, containerized workbench is under progress.

7 Acknowledgement

Slurm workbench is developed with the support of the following projects: Chinese National Natural Science Foundation(No. 12105303, No. 12175255) and Youth Innovation Promotion Association of Chinese Academy of Sciences(No. 2020017).

References

[1] F.A. Cruz, A.J. Dabin, J.P. Dorsch, E. Koutsaniti, N.F. Lezcano, M. Martinasso, D. Petrusic, *FirecREST: a RESTful API to HPC systems*, in *2020 IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud)* (IEEE, 2020), pp. 21–26

[2] F. León, G. Diaz, Revista UIS Ingenierías **18**, 95 (2019)

[3] M. Li, N. Chan, V. Chandra, K. Muriki, in *Practice and Experience in Advanced Research Computing* (2020), pp. 232–238

[4] *Scow*, https://github.com/PKUHPC/SCOW, open source project, accessed on 23-09-2023

[5] *slurm-web*, https://github.com/rackslab/slurm-web, open source project, accessed on 23-09-2023

[6] *Slurm rest api reference*, https://slurm.schedmd.com/rest_api.html, official document, accessed on 18-April-2022