Journal of Physics: Conference Series **513** (2014) 032027 doi:10.1088/1742-6596/513/3/032027

Testing SLURM open source batch system for a Tier1/Tier2 HEP computing facility.

Giacinto Donvito[1], **Davide Salomoni**[2], **Alessandro Italiano**[1 2]

[1] INFN-Bari, via Orabona 4, 70126 Bari (IT)
[2] INFN-CNAF, Viale Berti Pichat 6/2 40127 Bologna (IT)

E-mail: giacinto.donvito@ba.infn.it, davide.salomoni@cnaf.infn.it, alessandro.italiano@cnaf.infn.it

Abstract. In this work the testing activities that were carried on to verify if the SLURM batch system could be used as the production batch system of a typical Tier1/Tier2 HEP computing center are shown. SLURM (Simple Linux Utility for Resource Management) is an Open Source batch system developed mainly by the Lawrence Livermore National Laboratory, SchedMD, Linux NetworX, Hewlett-Packard, and Groupe Bull. Testing was focused both on verifying the functionalities of the batch system and the performance that SLURM is able to offer.

We first describe our initial set of requirements. Functionally, we started configuring SLURM so that it replicates all the scheduling policies already used in production in the computing centers involved in the test, i.e. INFN-Bari and the INFN-Tier1 at CNAF, Bologna. Currently, the INFN-Tier1 is using IBM LSF (Load Sharing Facility), while INFN-Bari, an LHC Tier2 for both CMS and Alice, is using Torque as resource manager and MAUI as scheduler.

We show how we configured SLURM in order to enable several scheduling functionalities such as Hierarchical FairShare, Quality of Service, user-based and group-based priority, limits on the number of jobs per user/group/queue, job age scheduling, job size scheduling, and scheduling of consumable resources. We then show how different job typologies, like serial, MPI, multi-thread, whole-node and interactive jobs can be managed. Tests on the use of ACLs on queues or in general other resources are then described. A peculiar SLURM feature we also verified is triggers on event, useful to configure specific actions on each possible event in the batch system.

We also tested highly available configurations for the master node. This feature is of paramount importance since a mandatory requirement in our scenarios is to have a working farm cluster even in case of hardware failure of the server(s) hosting the batch system.

Among our requirements there is also the possibility to deal with pre-execution and post-execution scripts, and controlled handling of the failure of such scripts. This feature is heavily used, for example, at the INFN-Tier1 in order to check the health status of a worker node before execution of each job. Pre- and post-execution scripts are also important to let WNoDeS, the IaaS Cloud solution developed at INFN, use SLURM as its resource manager. WNoDeS has already been supporting the LSF and Torque batch systems for some time; in this work we show the work done so that WNoDeS supports SLURM as well.

Finally, we show several performance tests that we carried on to verify SLURM scalability and reliability, detailing scalability tests both in terms of managed nodes and of queued jobs.

1. Introduction

The modern evolution of CPU is putting much pressure on the batch system software. Indeed, the ability to pack always more CPU cores on the same silicon piece make possible also for small-

medium site to have thousands of processing slots. The experience carried on with Torque/Maui highlight that it is quite easy to experience scalabilities and stability issues.

2. Experience with Torque and MAUI

The INFN-Bari farm is using Torque/Maui[1] since 2004. Since that date the farm has greately increased in number of CPU Cores reaching about 250 worker nodes, 5000 CPU cores and usually got up to 18'000 queued jobs. The standard version of the Maui scheduler is not able to deal with more than 4000 job in the queue, we patched the code in order to be able to analyse up to 18'000 jobs, but under this condition the Maui scheduler is often saturating the CPU of the batch server, and usually becomes unresponsive to the client interaction. The torque server under high load, often saturated the memory available on the batch server and this could lead to a crush of the pbs_server itself. In case of network problem in connecting to the worker node, the pbs_server could hang and become unresponsive.

3. Requirement for a batch system

In this section we well describe the main requirements for a new batch system in order to fulfil the use case of a HEP grid site.

Scalability
: How the batch system is able to deal and cope with the increasing number of Cores and nodes in a computing farm. How it deals with the increase in number of jobs submitted, running and queued. How it deals with the increasing number of concurrent users interacting with the batch system

Reliability and Fault-tolerance
: high availability features on the server side, that could be able to deal with the server failure, but it is also important to conside the client behavior in case of service failures.

Scheduling functionalities
: The INFN-Bari site is a mixed site, both grid and local users share the same resources. We need complex scheduling rules and full set of scheduling capabilities in order to both use efficiently the available resources and provide to the users a good experience in terms of queue time.

Low TCO
: It is important to choose a batch system that is able to provide all the needed functionalities and the required scalability trying to keep as low as possible the Total Cost of Ownership. This means that we would prefer the solution that are OpenSource. It is also important to take into account the cost of operations, personnel to manage it, etc.

Grid enabled
: As we are dealing with a WLCG site so it is quite important to chose solution that are easily supported by CREAM CE in order to join the official EGI Grid infrastructure.

4. Testing a new batch system: SLURM

A new batch system that could be of interest is SLURM[2]. This solution developed by Lawrence Livermore National Laboratory is used by many of the TOP500 Super Computing in the world. The documentation states that it is possible to configure about 65'000 worker node, to execute more than 120'000 jobs per hours. So we do not really expect to have perfomance issues with the scale of site that we are intented to support. In the next section we will report both functionality and performance test.

4.1. SLURM installation and configuration testing
Behind the idea to test SLURM there is a huge issue. The new batch system must be at least like the one in production in terms of functionality and scalability. Keeping in mind this issue all the test activities have been done in order to verify that slurm can provide the same service levels as the products already in production. As a first step we have tried the installation process and we have verified that it is well documented and the step-by-step guide works; this means that we successfully installed slurm at first attempt.

In order to install the software the site admin can directly download the source code from the official website and follow the installation procedure that provides instruction on how to build the rpm.

This approach could be a plus in our environment as the site admin is able to patch or customize the code as needed.

During this test activity, a new version was released and we were able to try the upgrade process too. It worked fine without any constraints and all the slurm services and data came back to work as desired and all the jobs running successfully ended.

As the second step of compare process we focused on the job priority which is one of the main issues because the computing resource we manage are used by several users belonging to several groups, subgroups and organizations so the priority must be defined in a hierarchical way and must be a function of the resources used in the past in order to grant a fair use of the resource between all the users.

Slurm provides a multi-factor job priority plugin which implements a fair-share algorithm so this means that it can be intended to suite our needs.

The plugin has direct access to the accounting system in terms of gathering the resource usage information, users share configuration and hierarchical groups membership. The Slurm accounting system can be deployed to use a MySQL on a dedicated server. This solution could become a single point of failure in case of a network connectivity problem between slurm master and the server hosting the MySQL server. All the configuration must be done using the command line and it is immediately available without the need of reconfiguring the system.

4.2. SLURM functionality tests
We have tested several configurations of the SLURM batch system in order to replicate the configuration that are already in production both using Torque/Maui at INFN-Bari and with LSF at INFN-CNAF Tier1. This, as we will show, covers a huge list of capabilities.

4.2.1. Accounting One interesting feature of SLURM is that it is quite easy to configure a database backend to host accounting data. Indeed, this is quite important because this means that a site admin could have the possibility to ask for the statistics of usage with a poweful tool based on a Relational Database Management System (RDMS). At the moment both MySQL and PostgreSQL are supported.

4.2.2. Scheduling One of the most important features is the capability to provide hierarchical fair-share capabilities. This is a feature that allows user quotas and priorities to be managed within an administrator-defined hierarchy. For example the site administrator could configure the share on the computational resources used on a group bases, but within the group it is also possible to configure the share for the single users. Indeed, this feature is supported in SLURM but it needs that the accounting on the RDBMS is enabled. We tested also the capability to provide priority based on Quality of Service definitions. Indeed, this could be useful to help the site admin to manage easily the priority over a large variety of users and groups. It is also possibile to implement scheduling priorities based on users/groups/queue etc. We tested also the possibility to enable *"age based scheduling"*: this is quite important in order to avoid job

starvation. This means that a job that stays in the queue for a long period starts gaining priority based on this amount of time. Using SLURM it also possible to use *pluggable "consumable resources"*. In this way the site admin could configure an arbitrary resource on each worker node (like GPU, network, memory, etc.) and providing a plugin to publish the usage for that resource. The batch system will schedule the jobs taking care of the worker node status and the requirements of the jobs. It is also possible to configure a limit on the number of running or queued jobs for: users, groups, etc. This is quite useful when a site admin needs to protect the cluster from misuse by few users/groups. SLURM support also the "job size scheduling", this means that the site admin could choose if the large MPI jobs will have greater or smaller priority.

4.2.3. High Availability It is not possible to rely on a single host to guarantee the services of a whole site. Indeed, if the batch master fails all the computational resources will be unavailable for all the users. In order to avoid this single point of failure it is of great importance to have a fault tolerant configuration, where two o more hosts are or can be used to provide the batch manager server. We have tested the support for fail-over among two different servers. This feature in SLURM is based on the concept of "event". Indeed, it is possible to configure the execution of a predefined script when a particular event occurs. The solution is based on the classical paradigm of active-standby configuration. For example it is possible to use the event of failure of the batch master server to trigger the fail-over on the secondary machine, that is already configured. Both the client and the worker nodes are configured in order to know that there are two servers. They are able to fallback seamless on the secondary server without any need of manual intervention. Moreover, the client is able to deal also with temporary failure in reaching the server. Indeed, the client when it retrieves and error, just sleep for a while and retry afterwards, without the need of any action from the user.

4.2.4. Advanced features With SLURM it is possible to suspend and resume jobs. The batch system is also capable to re-execute jobs that were running on a node after the failure of the node itself. This is quite useful because the users do not see any job failure from his/her point of view. We also successfully tested the support on SLURM for several kind of jobs: MPI, "whole node", multi-thread. It is also possible to submit "interactive jobs" that allow the user to obtain an interactive shell on a worker node. SLURM give also the possibility to associate computing resource to a specific group or set of users, it is also possible to set-up ACLs in order to define wich resources could or could not be used, both in terms of nodes and also in terms of queues. The site admin using SLURM could also configure an hard limit on memory used by jobs on worker node, indeed using this configuration as soon as the job will use more memory the slurm daemon kill the process. Moreover it is possible to configure cgroups[3] in order to have a more granular control on the resource usage on the worker nodes.

4.3. SLURM performance tests
We also tested the behaviour of the SLURM batch system when it is under heavy load condition, in order to measure how it cope with the increasing

- number of jobs in the queue
- number of WNs
- number of concurrent submitting users
- amount of jobs submitted in a small time interval

During the tests, all the job has the same duration and they basically just sleeps on the worker node. This was done in order to avoid overloading teh WN, as this may couse trouble on the

slurm agent and put a bias on the test. They are mainly single CPU jobs, plus a small fraction of multi-CPU jobs. This is in order to reproduce the standard situation in a HEP Tier1-Tier2 farm. The jobs are not trasferring any file as the standard configuration of SLURM is to have a shared file-system among the nodes belonging to the cluster. The first test aims to show how SLURM deals with the job scheduling when the number of jobs in the queue increasing up to 25'000. In this test we run for 24 hours continuously submitting long jobs. The test is carried on measuring the number of queued jobs (the blue line in the graph), the number of executed (the orange line) and accepted (the green line) jobs per minutes. The aim is to prove that the scheduling is not affected by the high number of jobs in the queue, in terms of capability to accept and delivery jobs. In the Figure 1 it is evident that the number of accepted and executed jobs are pratically constant while the number of queued jobs are increasing.

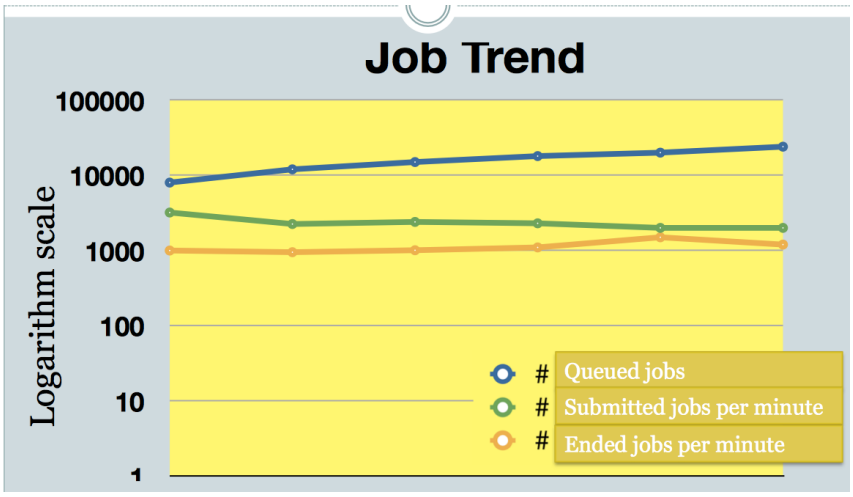

Figure 1. SLURM performance test: number of queued jobs and the submitted and executed job per minutes.

Indeed the SLURM server show a good behaviour: the load on the server is quite low (stable at 1 of loadAverage), the memory usage of the server is about 200MB and it is able to run stable with a scheduling period of about 20 seconds.

We run also another test that aims to check the capability to cope with an high number of worker nodes and batch slots, respectively set at 250 and 6000. In this case the test is executed submitting from 10 concurrent clients against the same server. Each client will submit 10'000 jobs for a total of 100'000 jobs to be executed. In this test the aim is to prove the reliability of the SLURM by means of measuring the number of executed and accepted jobs. The test went fine: the loadaverage on the machine is stable at 1.20. The submission, from a point of view of the users, do not experienced any problems, the memory used on the server always less than 500MB. At the beginning of the test the submission/execution rate is 5'500 jobs per minute. Also during the pick of the load the the rate of submission/execution is about 350 job/minute.

We also executed a stress test to prove the reliability of the server: the test was carried on with 6000 Cores available, with two clients continuosly submitting jobs for 4 days, always keeping in the queue about 20'000 jobs. At the end of the test no crush, or memory leak was found and the load on the server is always under control.

5. Using SLURM in the EGI Grid

As we are involved in the EGI grid computing infrastructure it is of paramount importance to test the support for SLURM in the CREAM Computing Element[4].

The Cream CE is a modular piece of software that is able to accept job submission from the grid and forward to the local batch system. Indeed, using the BLAH[5] component it is possible to implement the support to several batch system with minimal effort, indeed the Cream CE is already supporting: LSF, Torque/PBS, Condor, SGE, BQS. Also SLURM is wall supported, indeed during our test we found that all the needed component are fully usable:

- Blah/job submission: works
- Infoproviders: works
- Accounting (Apel)[6]: works

5.1. Job Submission
More deeply we configured both the SLURM master (toghether with the MySQL used for the accounting) and the worker nodes before installing grid middleware. After this we installed the EMI middleware. The yaim configuration was quite straightforward, and smooth. We created the queues on the batch system by our own associating computational resources and user group allowed to use each of the queues. One interesting feature of the EMI middleware is that all the grid jobs works perfectly also if the nodes do not have a shared file-system among them. Usually SLURM require this in order to copy back the error/output files. In this case the CREAM-CE it self is able to bring back the files, using the wrapper of the job that is submitted to the worker node.

5.2. Infoprovides
In the latest version of the EMI middleware, the SLURM infoproviders released with CREAM is able to correctly read the status of each SLURM queue (in terms of queued and running jobs) and provide to the Site BDII the needed information. From the test that we carried on, the dynamic infoprovider run quite fast and reliable. So also in this case no major problem were found.

6. Conclusions and Future works
During this tests we found that SLURM is a fast and reliable batch system solution, that has the advantage of being completely OpenSource and community driven. Indeed we already interacted successfully with the developers team proposing patch, and asking for support.

We have been able to implement all the needed configuration, replicating exactly what we were able to do using both torque/maui and LSF. Also from the point of view of Cream CE support everything is ready to use this software in a real pre-production environment. In order to be put the solution in production we only need to carry on a full set of stress test using Cream CE, in order to test the reliability of the full chain.

We also plan to test the compatibility layer with Torque and SLURM, that is realized by means of Command Line Tool that emulate all the torque clients. In this way for it will be easier for a local user of our batch cluster to move from torque to SLURM.

References
[1] http://www.adaptivecomputing.com
[2] https://computing.llnl.gov/linux/slurm/
[3] http://en.wikipedia.org/wiki/Cgroups
[4] http://grid.pd.infn.it/cream/
[5] http://www.eu-emi.eu/kebnekaise-products/-/asset_publisher/4BKc/content/blah
[6] https://wiki.egi.eu/wiki/APEL