Changing the batch system in a Tier 1 computing center: why and how

2014 J. Phys.: Conf. Ser. 513 032018

(http://iopscience.iop.org/1742-6596/513/3/032018)

Journal of Physics: Conference Series **513** (2014) 032018 doi:10.1088/1742-6596/513/3/032018

Changing the batch system in a Tier 1 computing center: why and how

Andrea Chierici
INFN-CNAF
E-mail: andrea.chierici@cnaf.infn.it

Stefano Dal Pra
INFN-CNAF
E-mail: stefano.dalpra@cnaf.infn.it

Abstract. At the Italian Tier1 Center at CNAF we are evaluating the possibility to change the current production batch system. This activity is motivated mainly because we are looking for a more flexible licensing model as well as to avoid vendor lock-in.

We performed a technology tracking exercise and among many possible solutions we chose to evaluate Grid Engine as an alternative because its adoption is increasing in the HEPiX community and because it's supported by the EMI middleware that we currently use on our computing farm. Another INFN site evaluated Slurm and we will compare our results in order to understand pros and cons of the two solutions.

We will present the results of our evaluation of Grid Engine, in order to understand if it can fit the requirements of a Tier 1 center, compared to the solution we adopted long ago. We performed a survey and a critical re-evaluation of our farming infrastructure: many production softwares (accounting and monitoring on top of all) rely on our current solution and changing it required us to write new wrappers and adapt the infrastructure to the new system.

We believe the results of this investigation can be very useful to other Tier-1s and Tier-2s centers in a similar situation, where the effort of switching may appear too hard to stand. We will provide guidelines in order to understand how difficult this operation can be and how long the change may take.

1. Introduction

Batch system is a key element of INFN Tier1, allowing flexible and fine grained control on computing resources. Our center adopted LSF since the very beginning, a very robust and reliable solution widely adopted in HEPiX community. Lately new software (free and commercial) gained our attention, thanks to interesting features as well as attractive pricing. INFN has a nation-wide contract in place to use LSF that will expire by 2014: during this time we would like to evaluate alternatives. In this article we explain our investigation on UNIVA Grid Engine a solution that has gained consensus in our community. Another INFN site evaluated SLURM in a very similar way and discussion is ongoing inside INFN on what could be a suitable alternative to be adopted by our computing centers.

2. Grid Engine

Grid Engine is an open source batch-queuing system, originally developed and supported by Sun Microsystems. Univa Corporation provides a commercial and supported evolution of Grid Engine, which recently has been adopted by many computing centers in HEP community. We evaluated this product in order to understand if it can fit our requirements.

2.1. Grid Engine Setup

Grid Engine (from now on GE) suggested setup is rather simple and comparable to what we have now in our farm: a master server is required, with a possible shadow server to grant high availability. All hosts that are available for GE can either be set up in a single cluster, or they can be split up into multiple groups of hosts where each group defines a cluster. These smaller sub-clusters are named *cells*. Each host can adopt one or more roles, but each host should belong to only one cell.

2.2. Classic vs BDB Spooling

Grid Engine supports two different spooling methods on the master host: classic spooling and BDB spooling. With classic spooling, the master service creates files containing the configuration objects of a Grid Engine installation in human readable format (a flat file). When BDB server spooling is enabled, a BDB database will be used to make data persistent. Both methods have different requirements and characteristics. Advantages of classic spooling are mainly that data is stored in plaintext files and that it's easy to backup and to achieve master server high availability. Advantages of BDB spooling are mainly on performance. It's generally suggested that sites should start with classic spooling and in case of performance problems, simply reinstall with DBD, since capturing current GE configuration is simple.

2.3. Differences between GE and LSF

Although LSF and GE have been independently designed and developed, they have many similarities, differentiating in a few concepts and operational procedures. A few worth of mention are:

- the "queue" concept in GE is traditionally referred to a compute node while in LSF "queue" refers to a specific area for pending jobs. Later evolution of GE added "cluster wide queues", making migration from LSF simpler
- there is no "configuration file" in GE. Everything is done through shell commands, possibly opening an editor. One benefit of this approach is that it prevents to some extents accidental corruption of the configuration. For example, defining the equivalent of a LSF queue in GE is done by issuing a few `qconf` commands, each one with the desired options
- on execution hosts, GE requires only one daemon running while LSF requires three

2.4. Evaluation Goals

In order to be able to fit our needs the new batch system must be compatible with our use-cases:

- effective integration with grid middleware
- support for hierarchical fair-share
- support for multicore jobs
- allow for implementation of priorities and advance reservation
- current LSF configuration should be translated to GE without loss of features
- master node must support high availability
- quick and safe recover after hardware failure is not an option

- scalability: on 18K cores we currently run 80 to 130 Kjobs/day
- jobs are heterogeneous (submission rate, lifetime, CPU intensive or I/O intensive or both, etc.)
- impact of the change to our user community. Although the change would be transparent to users from the Grid community, local submission mechanisms should be adapted. Effort from our side would be needed to reduce such an impact to a minimum.

During the tests in our lab we were able to verify that all these requirements are met.

2.5. Evaluation Timeline
Following is the timeline we adopted for the evaluation of GE:

(i) setup a mini cluster for self-training and get confident with GE admin and maintenance tasks
(ii) integrate GE with EMI middleware (cream CE) and run test jobs on it
(iii) evaluate Customer support (answer readiness, ticket management)
(iv) evaluate different setups for performance and HA:
- classic spooling (flat-files on shared directory)
- DB spooling
- master with shadow

(v) evaluate integration with accounting and monitoring:
- data collection for our local accounting and monitoring system must be adapted
- dgas and apel compatibility

(vi) conversion and emulation of real use cases
- adaptation of current LSF "pre_exec" and "post_exec" scripts
- evaluation of special configurations, such as "Job Packing" [3] or GPU management

(vii) test using production farm
(a) install one cream CE with GE extensions to provide an access point for the grid to our farm
(b) share computing nodes with LSF
(c) gradually migrate computing resources from LSF to GE. Two ways to achieve this:
- by nodes: a node is entirely managed either by GE or by LSF
- by slots: x slots in a node are assigned to GE and $N - x$ to LSF. Gradually increasing x from 0 to N realizes a complete switch.

We chose the latter, as it offers a finer control and because it also generalizes the former.

3. Evaluation summary

The overall result of our evaluations was satisfactory. The support looks adequate for our needs, and the key requirements are fairly easy to implement with the new system. Not everything is straightforward, though. Some important parameters in LSF have no direct equivalent on GE. The EXIT_RATE parameter, for example, provides a mean to disable a misbehaving host, preventing the so called *black–hole* syndrome: currently this is a "feature request" in Univa GE: the ability to ban sick hosts can be however emulated by post execution scripts. We are interested in the capability to have jobs dispatched on the smallest possible set of nodes, depending on their family: this can be easily achieved. Another point of interest is about GPU management since there is an increasing demand for this kind of resources: suggestions and guidelines are available already for a number of use case scenarios.

4. Integration with higher level middleware

The INFN-T1 centre is mainly devoted to grid computing for the LHC experiments where jobs get submitted to Computing Elements, hosts acting as the entry point to the site; local job submission is a minor component (approx. 20%) of the overall workload.

Grid jobs are usually unaware about the locally adopted batch system, since the CE provides a generic interface to it. At INFN-T1 we currently run EMI3 cream CE with LSF plugin and we are going to test the GE plugin [1] against our Univa GE test cluster. We need to evaluate and compare reliability, performance and scalability when adopting this plugin.

4.1. Local Monitoring and Accounting

Recently we rewrote all our local monitoring and accounting system, making it more flexible and scalable. Before this, we relied on LSF shell commands, parsing the output with perl scripts and graphing it conveniently. Now we use a more reliable approach, querying LSF directly through its APIs with a set of python scripts. These pieces of code must be adapted when interfacing to GE:

monitoring if using BDB spooling we plan to directly retrieve data via SQL from the PostgreSQL database. If using flat file spooling then the DRMA specification will be used. Currently there does not appear to be a production quality piece of code available to bind DRMAA on GE to python, and this means we need to use the C++ implementation or either the Java one.

accounting Univa GE comes with the Accounting Report COnsole (ARCO) module. It collects raw events into a report file, which is then read at regular intervals, imported into a RDBMS and deleted thereafter. The administrator can then define a set of derived values to express its metric of interest, that can be added to the predefined ones.

4.2. Grid Accounting

Grid accounting is obtained by matching batch side accounting records with grid ones. These records are produced in the Computing Element (*cream-ce*) by the *blah* component, and provide a mean to map the local job id with the external informations (virtual organization, job owner and its credentials). The accounting service currently deployed at INFN-T1, DGAS [2] and its alternative, APEL, have both a module for GE: we plan to verify the functionality against Univa GE, and adapt the code to our requirements, if necessary.

4.3. WNODeS integration

Currently WNODeS, the Worker Nodes on Demand Service developed and adopted by INFN, supports natively Torque/MAUI, LFS and lately SLURM. Being it an internally developed product, we do not expect any major problem in adding support for this batch system, but it's important to notice that right now, a migration to GE would break our compatibility with WNODeS and would require a significant effort in order to provide a testbed and adequate support to developers to allow for the integration.

5. How to make the switch

Switching from LSF to GE is not really difficult, because both systems have common requirements and do not need many specific interventions on the worker nodes. The switch must be implemented at two different levels: grid level and worker node level. At grid level, we install Cream Computing Elements with GE extensions and point them to our Master Node, that shares the same WNs set. When all the WNs have turned to GE only job execution, it's safe to remove the LSF client and turn the LSF CEs off.

At worker node level, both LSF and GE clients run at the same time on the node and the number of slots is modified in order for GE to take over the LSF slots. An alternative approach is to allocate a subset of WNs on the farm to GE, once all the LSF jobs on them are terminated.

6. Slurm

INFN Bari tested SLURM, the Simple Linux Utility for Resource Management, a free and open-source job scheduler for the Linux kernel. SLURM implements a fairly traditional cluster management architecture, rather similar to the one we found in LSF and GE.

SLURM passed all the tests that were made, and is natively supported by WNODeS.

7. Future work

We expect GE to work without major problems with our production use-cases. Should we actually change our current batch system and adopt GE, we expect that we may need to get and provide support (among our community) to solve any compatibility and reliability issues concerning the CREAM CE module interfaced to GE, especially with respect to the Univa flavor. Testing the reliability and adaptability of this component within a Tier1 is in fact one of the major key points to ensure a safe migration and to keep undesired side effects away.

8. Conclusions

Grid Engine (UNIVA flavor) proved to be a very reliable and stable batch system. We verified it can quite easily take the place of our current batch system in our environment, thanks to its scalability and since all the software that rely on it is at production quality level (or can be with small effort). INFN Bari proved SLURM to be a good alternative too, giving us another software to choose from. Our current batch system contract will finish next year and now we know that switching it is possible in a safe way.

References

[1] R. Dopazo Rosende, Á. Simón García, E. Freire García, C. Fernàndez Sànchez (CESGA), A. López García, P. Orviz Fernàndez, E. Fernàndez Del Castillo (IFCA) and G. Borges (LIP)"Grid Engine batch system integration in the EMI era", Proceedings of Science, p. 7, 2012.

[2] R.M.Piro, A.Guarise, G.Patania, A.Werbrouck: "Using historical accounting information to predict resource usage of grid jobs", Future Generation Computing System (2008), doi:10.1016/j.future.2008.11.003.

[3] S. Dal Pra "Job packing: optimized configuration for job scheduling", Oral Presentation at HEPiX Spring 2013 Workshop.