Peer-to-Peer Scheduling in the Czech National Grid and Beyond

Šimon Tóth[1]

[1] CESNET, Prague, Czech Republic
emails: simon@cesnet.cz

Keywords: torque, grid scheduling, distributed scheduling

1. Introduction

In this work, we will present our advancements in the Torque Batch System fork used in the Czech National Grid *MetaCentrum*. While the major focus was stabilizing the numerous experimental features implemented during previous years, we have also advanced our feature set noticeably. Our custom scheduler has gone through several architectural changes. We will also touch upon features that have finally entered production, and describe how they had to be changed since first introduced [3].

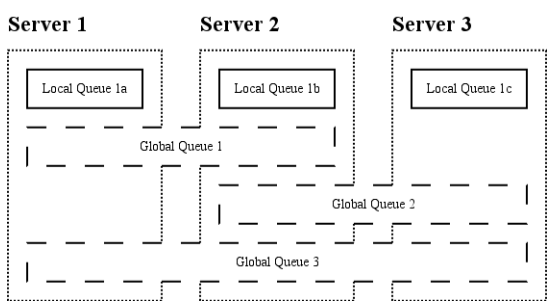
2. Resource Management and Enforcement

One of the notable features of the Czech National Grid was always exclusive resource allocation. Each job was assigned a static allocation of resources with the expectation that the job will not consume more than requested. This was however, for a long time, not enforced. This led to many unfortunate cases of jobs clashing with each other, or even in extreme cases, crashing entire nodes.

To address this issue, we have implemented a configurable support for notifying users and terminating offending jobs. Current support is implemented inside the Torque node daemon `pbs_mom` and includes checking for CPU cores, physical and virtual memory violations. Jobs requesting entire nodes, or running alone on a particular node can be excluded from these checks. After a transitional period, when the system was running in warning-only mode, we are now terminating any offending jobs, thus avoiding most of the job clashing issues.

3. Changes Entering Production

One feature that went through multiple iterations before finally entering the production environment is the Peer-to-Peer scheduling support [1][2][3]. This feature provides support for global queues, a model that allows jobs from one server to be executed on any other server sharing that particular queue. If remotely executed, the job leaves a copy of itself on the original server in a special state. Global queues can span across multiple servers and jobs always follow policies of the server they are executed on.



Second big feature is support for so called on-demand clusters [2]. Using this feature, we are able to switch our infrastructure into a more dynamic environment. When a job re-

quirements cannot be immediately satisfied, the scheduler will also try to consider a reboot of a free node into a suitable configuration. Using this support we can provide a wide range of software configurations without the need to lock particular nodes into such (possibly rare) setups.

4. Scheduler Improvements

While we are continuously improving our scheduler with new features, there are two changes that deserve special attention. First change involves the switch to a client-server model. Originally Torque used a server-server model for communication between the scheduler and server. Communication was always initiated by the server, notifying the scheduler of an important event. While this approach was definitely efficient, the scheduler lacked the ability to easily connect to specific server when needed. Such support was required for the final version of the Peer-to-Peer scheduling support. This is also the first step toward a fully client-based scheduler, running under user UNIX privileges.

Second change is also architectural. Since the very first version of our fork, our scheduler worked with a simple two-value model when determining whether a particular job can, or cannot be currently executed. Unfortunately this led us to an issue where the scheduler was unable to make a distinction between jobs that could not be run temporarily and jobs that could never run. Such jobs would then still trigger anti-starving and other measures, which can result in poor machine utilization levels. Our change switched the model from two values into three values: available now, not available, not possible. Jobs that can never run are now correctly marked as such, skipped and their users notified.

5. Conclusions and Future Work

This paper describes some of the changes implemented into the Torque Batch System fork maintained by MetaCentrum. Our fork is continuing to prove as a valid alternative to the commercial PBS Pro while providing good scalability and stability.

Our future efforts will be concentrated in two main areas, firstly we are working on a large set of scheduler improvements that are aimed at improving machine utilization (backfill, smarter anti-starvation measures, improved software licenses handling, etc.). Second large effort is concentrated on merging our fork with the current version of Torque, which should further improve scalability of our solution.

Source code for our fork is available at http://cesnet.github.io/torque/. RPM packages for various distributions are available at:
https://build.opensuse.org/project/show?project=home:Let_Me_Be.

Acknowledgments. The work presented in this paper was conducted under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" LM2010005 funded by the Ministry of Education, Youth and Sports of the Czech Republic.

References

1. Š. Tóth, M. Ruda and L. Matyska: Towards Peer-to-Peer Scheduling Architecture for the Czech National Grid. In Proc. Cracow'10 Grid Workshop, 2011, pp. 92-101
2. Š. Tóth and M. Ruda: Practical Experiences with Torque Meta-Scheduling in The Czech National Grid. Computer Science, 13 (2):33-45, 2012
3. L. Matyska, M. Ruda and Š. Tóth: Peer-to-Peer Cooperative Scheduling Architecture for National Grid Infrastructure. Data Driven e-Science, 2011, pp. 105-118