Towards peer-to-peer scheduling architecture for the Czech National Grid

Š.Tóth, M.Ruda, L.Matyska

CESNET, z.s.p.o., Prague, Czech Republic

Abstract

The Czech National Grid Infrastructure MetaCentrum has been using a central scheduler infrastructure for approximately the past 10 years. This facilitated simple administration and direct support for large jobs running across several geographical sites. The knowledge of complete state allowed the scheduler to provide high quality decision making incorporating features like fairshare. On the other hand, this central setup created a single point of failure issue and also reached its scalability limits.

In this paper we describe our work towards a new distributed architecture that maintains high scheduling quality while solving most of the single server issues.

Our new distributed architecture provides both local autonomy and high scheduling quality. Users can still submit jobs locally even when cross-site connectivity is lost. Individual schedulers work primary with their local server but still maintain global state, that allows them to mimic centralised scheduling features. The architecture still supports central accounting and fairshare across the entire grid.

Implementation is based on the open-source Torque batch system, which replaced the previous commercial PBSPro central server installation. Torque provides a similar codebase as it has a common ancestor with PBSPro in OpenPBS. Torque therefore provides familiar interface for both users and developers.

1 Czech national grid

The Czech national grid infrastructure is composed from mostly computational resources. These are usually clusters, that are spread across the country, concentrated in several geographical sites (figure 1).

MetaCentrum is currently experiencing rapid growth of both computational resources and user base. This was the prime reason for the new distributed architecture proposal.

The expected grow is mostly due to new connected sites. New sites not only increase the instability of the network but new site owners expect certain amount of autonomy from MetaCentrum. Unavailability of the local cluster when connectivity to the central server is lost is usually unacceptable.

Currently[1] the Czech national grid includes approximately 1500 CPUs with

[1]November 2010

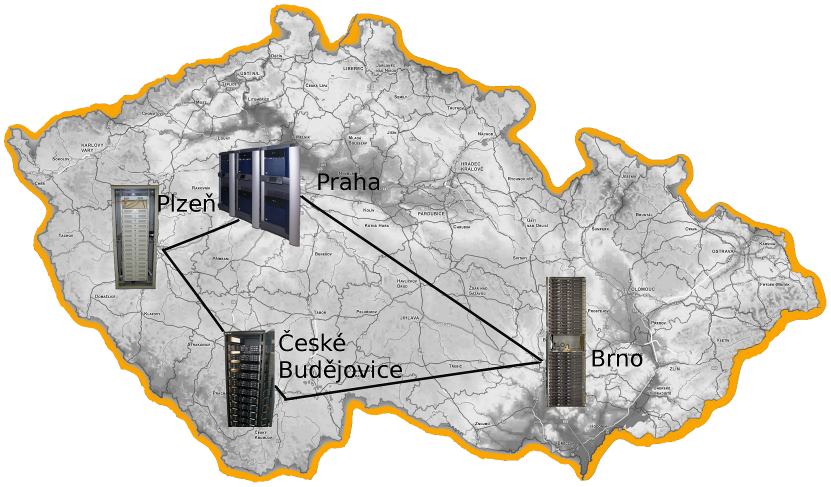

Figure 1: Metacentrum sites

clusters in four cities in 7 total sites. The grid is currently processing over 100 000 jobs each month.

2 Central server architecture

Previously used architecture consisted of a single server installation coupled with a scheduler (figure 2). The system was based on a commercial PBSPro batch system which was modified to facilitate the needs of the Czech national grid.

One central server allows the scheduler to work with full state information from the entire grid. This provides the scheduler with enough information to make informed decision like keeping empty space for starving jobs. Because the scheduler has complete control over the entire grid, it is simple to provide features like fairshare or multi-site jobs with additional effort.

2.1 MetaCentrum PBSPro improvements

The original PBSPro system was greatly enhanced to handle the entire grid and its requirements. Many stability improvements were implemented to limit systems the downtime. The original scheduler was enhanced to support all features required in the Czech national grid.

Detailed analysis of the original system and the possibilities for the transformation into a P2P distributed architecture were discussed in [3] and [4]. We will be discussing the specifics of features ported for the intermediate architecture and the current state of the implementation.

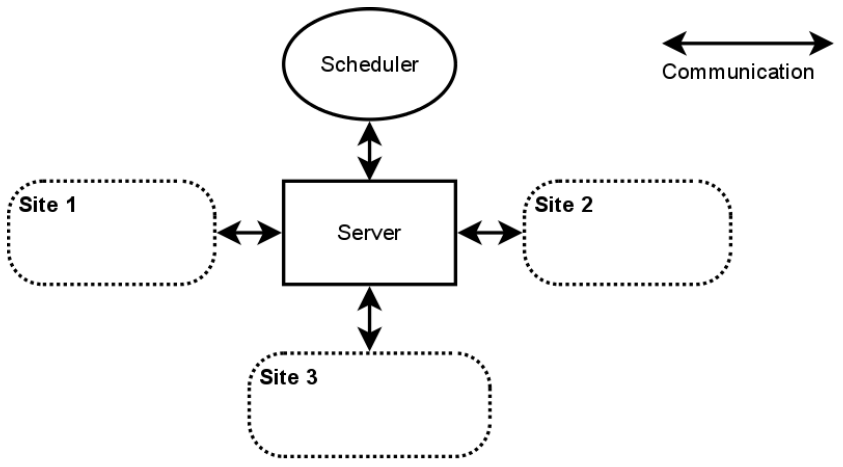

Figure 2: Central server architecture

Configurable support for starving jobs Starving job support was rewritten, server now supports setting a time limit on each queue. If any job is waiting in the queue longer than the specified time limit, it is considered starving. Starving jobs are scheduled onto all nodes (both free and occupied) which are then reserved and cannot be accessed by any non-starving jobs (until the starving job is executed).

Job priorities still take precedence over starving state and therefore a starving job will not be scheduled before a non-starving job with higher priority.

Support for external resources Very dynamically changing resources (like number of software licences, or current disk space) are kept separate from standard resources counted on the server. Special daemon (pbs cache) maintains these values and allows fast readout of current values. Scheduler is using these external resources in conjunction with standard resources obtained from the server.

Nodes dedicated to queues Queues can be configured to imply certain properties and therefore only access a specific subset of the computation nodes. Vice versa nodes can be configured to be dedicated to a certain queue.

Kerberos support Kerberos support was included into the system, upon submit, the users credentials are transferred to the server and upon execution, all job parts have the correct credentials assigned and maintained.

Virtual machines System was enhanced with virtual machine support. Physical machines now host two virtual machines. One of machine represents a high priority machine the second one represents a low priority. Access to these machines is divided using queues. This allows the system to provide preemptible and backfill queues [1, 2].

Automatic node upgrade If a computational node detects a new version of its binary, it will restart at the earliest possible time.

2.2 Disadvantages

Single central server implies several problems. Central solution always introduces a single point of failure issue. If a server outage occurs, the entire grid is offline. If a scheduler outage occurs, no new jobs can be run.

Despite the fact, that the systems stability was improved significantly, network outages still can't be evaded. Each outage cuts out a portion of the grid from the server, disallowing the affected users to submit new jobs and the scheduler to run jobs on the affected site. Because the Czech national grid covers a large geographic area, the batch system has to work with increased timeouts. Each network outage therefore also slows down the server, which has to wait for each of the affected nodes (until they time out).

The scheduler itself was already reaching its scalability limits concerning amount of jobs scheduled and with the expected increase of sites and total job count, we need to create a replacement.

3 Distributed architecture

The new architecture was designed to overcome limitations of the central server setup. Detailed analysis was presented in the technical report [3], we will therefore only concentrate on the current state and its evaluation. But because the central server architecture offered us great scheduling quality, our primary goal was to maintain this high scheduling quality.

Features that require complete cluster state had to be mimicked in the distributed environment. For example fairshare information requires complete cluster state to compute, therefore we have to either use a slightly outdated fairshare information that will be calculated externally, or ignore the imprecision caused by the distributed architecture.

Features that were provided by the concurrent access to the whole grid have to be reimplemented with the distributed architecture in mind. For example cross-site jobs will now require cooperation of several servers. The same applies for running a job on a different site than the one it was submitted onto.

The distributed version provides a natural solution to the single point of failure issue, together with improved scalability. This is critical for coping with the rapid growth of computational capacity and job count.

3.1 Distributed P2P architecture

The proposed architecture (figure 3) divides the grid into a set of semi-independent sites, each with its own scheduler and server and each maintaining its set of computational nodes. Each scheduler only schedules the local servers jobs. If a problem arises that cannot be solved locally, the schedulers will cooperate to solve the issue.

This includes various scenarios from local server saturation (no free computational nodes), to handling requests for features that are not provided by local computational nodes, or running multi-site jobs.

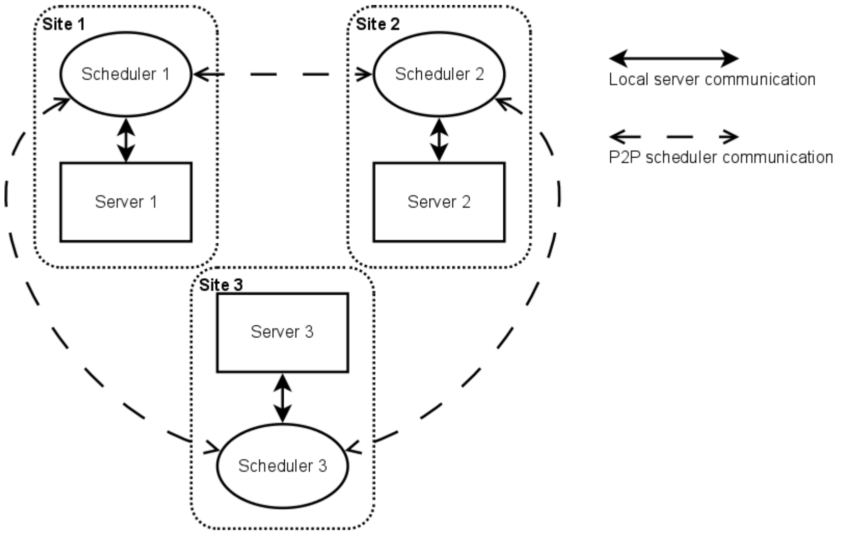

Figure 3: P2P Architecture

This architecture is targeting the scalability of hundreds upto thousands of sites. This relies on the assumption that only a small fraction of jobs will require the cooperation of many sites/schedulers.

3.2 Intermediate M:N architecture

The proposed architecture is a very big architectural jump from the original single central server setup. To solve the immediate needs of the Czech national grid, we proposed a new intermediate architecture (figure 4).

The intermediate architecture, doesn't separate the sites from each other and instead employs a full M:N communication network between schedulers and servers where each scheduler communicates with each server and vice-versa. This allows the schedulers to maintain a solid overview of the global grid state while still scheduling only a subset of the total jobs amount at a time. Network outages

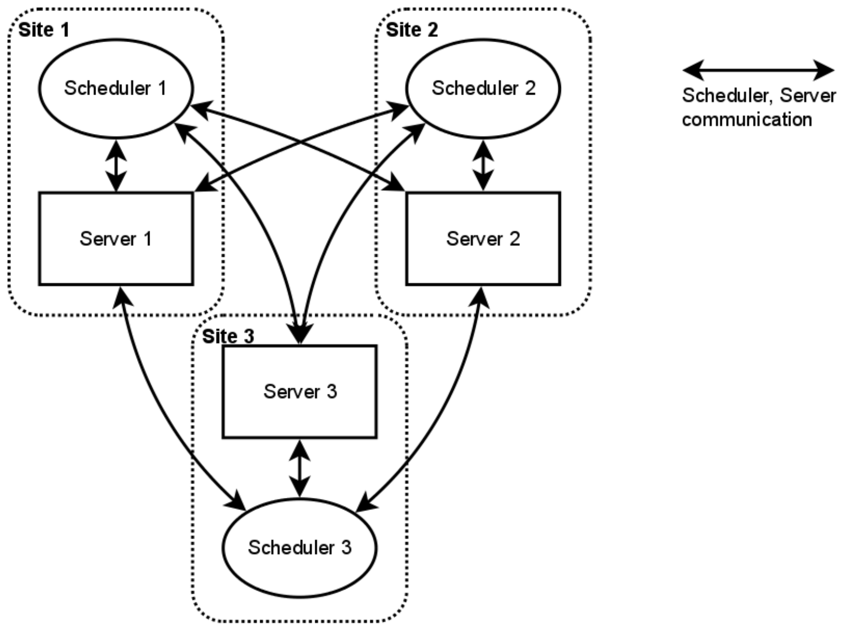

Figure 4: M:N Architecture

only affect cross-site jobs because each server is still fully functional when cross-site functionality is lost and can still manage its own local computational nodes.

4 Transformation towards P2P scheduling architecture

Together with the architecture change, we also decided to change the batch system. PBSPro is a commercial software licensed per CPU. This is something that really cuts into the cost of adding new clusters into the grid.

4.1 Torque

After evaluating several batch systems, we decided to select Torque batch system [7]. One of the major reason why Torque was selected is the fact, that Torque is mostly compatible with PBSPro.

Torque shares a common ancestor with PBSPro [6], both these batch systems are based from OpenPBS [8]. This ensures that both internal and external interfaces are very similar, although the projects did diverge a lot since forked.

Torque is a lively project with patches continuously being submitted from various sources. It is also the most commonly supported backend for grid interfaces like glite [11] and globus [10].

We based our work on top of Torques FIFO scheduler, instead of using an external scheduler like Maui [5]. The provided FIFO scheduler has a very simple and clean codebase, which allowed us to quite rapidly port custom extensions from PBSPro into Torque. The simple codebase allows us to implement even very experimental features like the distributed architecture.

4.2 Prototype

The first action towards the optimal architecture implementation was a prototype of the M:N architecture based on Torque. Server daemon was modified to wake up all schedulers and scheduler daemon was enhanced with the ability to schedule multiple server sequentially.

To improve the initial performance and also to test the suitability of similar modifications we implemented a mutual exclusion protocol, that allows the scheduler to claim the server for a short period of time. This exclusivity is only enforced for job modifying commands. User tool requests for job modifications are processed immediately after the server is released.

The schedulers logic was completely rewritten to support server state caching, job moving and trivial resource counting (which wasn't present in the original FIFO logic).

4.3 Evaluation

After the first prototype was implemented we needed to evaluate the performance of this prototype. This included both the Torque batch system performance (specifically in comparison with PBSPro) and the impact of the new architecture implementation.

To evaluate the performance a testing platform with 205 light virtual machines (5 servers, 200 computational nodes) based on Linux containers [9]. Each measurement consisted of 5000 empty /bin/true jobs being submitted into this virtual cluster.

These tests gave us a very good overview of the initial scalability.

We were primarily concerned with the total time it took to execute all 5000 jobs, but we were also interested in the progress (figure 5). The progress graphs visualise the amount of jobs submitted into the cluster, waiting in queues, running and jobs already finished. The graphs also map the run traffic (amount of run requests), the move traffic (amount of move requests) and state overview of the cluster nodes (free/busy count).

Progress graphs were very valuable for tweaking Torque settings like scheduler wakeup frequency.

4.4 Distributed architecture impact

Multiple servers per scheduler (figure 6) have almost zero speed impact. Schedulers work in an almost completely synchronous manner, waiting for each run requested to be either confirmed or rejected. Therefore processing equivalent amount of jobs (split between servers) will result in almost identical time.

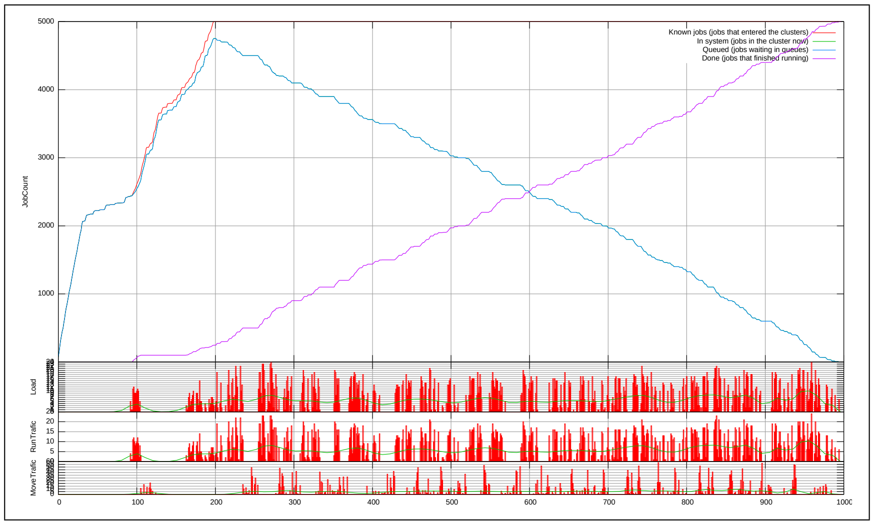

Figure 5: Measurement of job processing

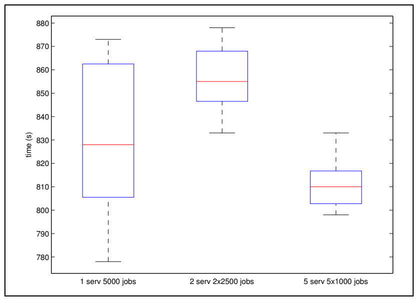

Figure 6: Multiple servers performance impact

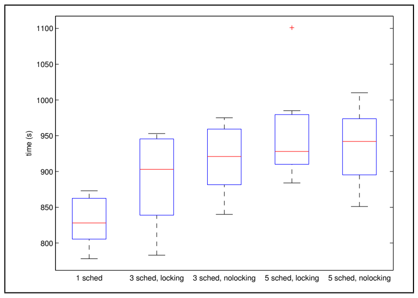

Figure 7: Multiple schedulers performance impact

Adding multiple schedulers (figure 7) to one server causes overhead on the server (more status requests). While the schedulers work in fully synchronous matter the server is very asynchronous, using callbacks and delayed tasks. Fortunately the overhead added by one scheduler is quite reasonable, approximately 1%-3%.

5 Further feature enhancements

With a functional prototype and a good overview of the intermediate architecture scalability and behaviour, we continued with porting of all PBSPro improvements into Torque. This also included features that were missing in Torque itself (implemented in PBSPro after the software was forked from OpenPBS).

Multi node jobs limitation Nodes can be configured not to accept jobs requesting multiple nodes. Scheduler will skip such nodes when scheduling multi-site jobs.

Resource semantics Torque batch system didn't contain internal support for resource counting or scheduling. The only supported resource semantics were processes and resource enforcement on the computational node using the (rlimit) per-process system limits. Both server and scheduler were enhanced with full generic resource support. Server has the ability to read reported resources from nodes, store them and verify each run request against these resources. For convenience the reported resources can also be mapped to a different name on the server.

Beyond the already supported resources, a generic counted resource can be specified on the nodes and the server will keep track of this resources usage.

Virtual clusters and VPN Full support for features already provided by PBSPro was reimplemented into Torque including support for virtual machines [1] and virtual clusters [2]. Active development of new features is now carried on top of Torque.

One of the new features already developed on top of Torque is support of on-demand virtual clusters. The grid now provides an infrastructure of physical machines and virtual machine images (described using provided features). Upon user request request, virtual machine images are selected (fitting the specification provided by the user), installed and executed on the physical machines selected by the scheduler.

If desired a virtual private network can be created between these machines (the machines can also be connected to an already existing VPN).

6 Conclusion

We have designed a new distributed P2P batch system architecture, which will be used as the future base of job scheduling in MetaCentrum.

We implemented an intermediate distributed architecture to satisfy the immediate needs of MetaCentrum as a temporary solution for quick deployment in the current MetaCentrum, that would still solve scalability and stability problems of the original PBSPro installation.

As presented this solution provides solid scalability and will even allow connection of new sites into MetaCentrum.

Deploying this solution on MetaCentrum will give us valuable feedback from real usage that will be extremely useful for the fully distributed P2P architecture.

References

1. Ruda, M.; Denemark, J.; Matyska, L.. *Scheduling Virtual Grids: the Magrathea System*, Second International Workshop on Virtualization Technology in Dis-

tributed Computing, USA, ACM digital library, 2007. p. 1-7. 2007, Reno, USA.

2. Miroslav Ruda, Zdeněk Šustr, Jiří Sitera, David Antoš, Lukáš Hejtmánek, Petr Holub, Miloš Mulač. *Virtual Clusters as a New Service of MetaCentrum, the Czech NGI.* Cracow Grid Workshop '09, 2009. Pages 66-71.
3. Miroslav Ruda, Šimon Tóth. *Transition to Inter-Cluster Scheduling Architecture in MetaCentrum.* CESNET technical report 21/2009.
4. Luděk Matyska, Miroslav Ruda, Šimon Tóth. *Peer-to-peer cooperative scheduling architecture for National Grid Infrastructure.* International Symposium on Grid Computing (ISGC), March 2010, Taiwan.
5. D. Jackson, Q. Snell, and M. Clement. Core Algorithms of the Maui Scheduler. In *Proceedings of 7th Workshop on Job Scheduling Strategies for Parallel Processing, 2001.*
6. The Portable Batch System. `http://www.pbspro.com`
7. Torque Resource Manager. `http://www.clusterresources.com/products/torque-resource-manager.php`.
8. Henderson, R. and D. Tweten. *Portable Batch System: External reference Specification.* NASA, Ames Research Center, 1996.
9. Šimon Tóth. Výkonnostní měření dávkového systému Torque `http://www.metacentrum.cz/export/sites/metacentrum/downloads/techreports/benchmarker.pdf`
10. I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp 2–13, 2006.
11. Laure, E., F. Hemmer, F. Prelz, S. Beco, S. Fisher, M. Livny, L. Guy, M. Barroso, P. Buncic, P. Kunszt, A. Di Meglio, A. Aimar, A. Edlund, D. Groep, F. Pacini, M. Sgaravatto, and O. Mulmo. Middleware for the next generation Grid infrastructure. In: *Computing in High Energy Physics and Nuclear Physics (CHEP 2004).*