Computer Science • 13 (2) 2012 http://dx.doi.org/10.7494/csci.2012.13.2.33

Šimon Tóth
Miroslav Ruda

PRACTICAL EXPERIENCES WITH TORQUE META-SCHEDULING IN THE CZECH NATIONAL GRID

Abstract *The Czech National Grid Infrastructure went through a complex transition in the last year. The production environment has been switched from a commercial batch system PBSPro, which was replaced by an open source alternative Torque batch system.*

This paper concentrates on two aspects of this transition. First, we will present our practical experience with Torque being used as a production ready batch system. Our modified version of Torque, with all the necessary PBSPro exclusive features re-implemented and further extended with new features like cloud-like behaviour, was deployed across the entire production environment, covering the entire Czech Republic for almost a full year.

In the second part, we will present our work on meta-scheduling. This involves our work on distributed architecture and cloud-grid convergence. The distributed architecture was designed to overcome the limitations of a central server setup, which was originally used and presented stability and performance issues. While this paper does not discuss the inclusion of cloud interfaces into grids, it does present the dynamic infrastructure, which is a requirement for sharing the grid infrastructure between a batch system and a cloud gateway.

We are also inviting everyone to try out our fork of the Torque batch system, which is now publicly available.

Keywords torque, grid scheduling, virtualization, cloud

1. Introduction

The Czech National Grid Infrastructure is composed from a heterogeneous set of computational and storage resources. These are mostly clusters that are spread across the country, concentrated in several geographical sites (Figure 1). Currently, the Czech National Grid Infrastructure includes approximately 3800 CPU cores with clusters in four cities, in 9 total sites. The grid processes over 750 000 jobs per year, with 500 concurrently running jobs on average.

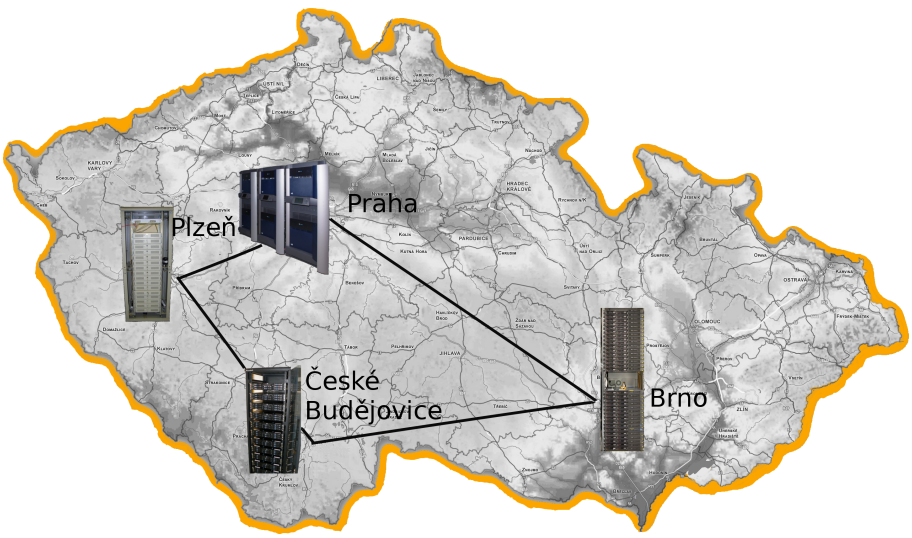

Figure 1. MetaCentrum sites

The entire system was originally governed by a single instance of a commercial batch system – PBSPro [1]. This solution was chosen to provide maximum interoperability with both scientific software (where PBSPro is widely supported) and to allow for the use of various middleware solutions (in the form of gateways into the grid). By being a centralised solution it also provided high scheduling quality and natively supported cross-cluster execution of jobs.

But due to the centralised nature of this solution, it was experiencing several issues. Firstly, there were licensing issues, when connecting new sites into the grid. PBSPro is licensed per CPU core, and each new cluster required new license negotiations, which made the process very inflexible. Secondly, the system was suffering from scalability issues and was very sensitive to hardware failures. Localised outages were resulting in large parts of the grid not being available, affected users becoming unable to submit new jobs into the system and servers unable to execute new jobs on the affected part of the grid.

[1] `http://www.pbspro.com`

To overcome these limitations, we have chosen a replacement for the PBSPro batch system, in the form of a custom solution based on the open source batch system – Torque [2]. By using an open source system, licensing issues were immediately solved and by enhancing Torque with our implementation of distributed architecture we were able to eliminate the centralised system limitations.

This of course required a reimplementation of all features that were PBSPro exclusive and also our local extensions implemented into PBSPro. While work on porting these features took almost a full year [11], at the end we were ready to switch our production system from PBSPro to Torque. Initial evaluation of the production ready implementation of Torque Batch System, its stability and performance, and an overview of the features that were backported from PBSPro were presented at the Cracow Grid Workshop 2010 [13].

Work on the Torque batch system culminated with a transition to this system in the production environment. Now, with almost a full year of production use, we are able to present our experiences with Torque and its evaluation as a production ready batch system.

Apart from our experiences with Torque, we will also present our work on the meta-scheduling architecture. This includes both our work on the distributed architecture that was designed to overcome the centralised server limitations and our work with scheduling the dynamic infrastructure that was designed to deal with cloud-grid convergence.

2. Torque batch system

Torque is a batch system. That means that it is responsible for managing the life cycle of computation jobs. From job submission, through scheduling and execution, monitoring to completion.

While Torque is an open-source software, it is being maintained by a commercial company "Adaptive Computing Enterprises, Inc.". As such, Torque is mostly maintained to be used as a back-end for external schedulers (Moab[3], Maui [7]).

One of the prerequisites of switching to Torque batch system was the reimplementation of features that were either supported in PBSPro and missing in Torque, or were our local extensions of PBSPro.

Both Torque and PBSPro have common code-base roots in OpenPBS [6]. This allowed us to port the custom changes we implemented into PBSPro relatively easily. The modified version of Torque batch system, as it is deployed in the production environment, is currently maintained as a separate fork of the original project.

Since the switch to production, Torque had to be further enhanced with new features to provide the desired quality of service and to satisfy user and administrative requirements.

[2] `http://www.clusterresources.com/products/torque-resource-manager.php`
[3] `http://www.adaptivecomputing.com/products/moab-hpc.php`

We are using a custom developed scheduler that is based on the original FIFO scheduler distributed with Torque, but which was since almost completely rewritten. To further improve the quality of scheduling, we are also exploring more complicated scheduling paradigms, like constrain based forward planning [2].

2.1. Kerberos

Unlike most other grid providers, which use certificates to manage user access to the grid, the Czech NGI is a long term user of the Kerberos protocol [10]. This greatly simplifies access to the grid for users because they no longer need to distribute their keys across the clusters.

We still support even users without Kerberos, but this use case is discouraged because Kerberos is also used to access network filesystems like NFS4 and AFS and other machines. To facilitate access to these resources during the entire job lifetime, Torque maintains an active Kerberos ticket for each running job. Support for non-Kerberos access is maintained mostly for external interfaces like Glite [8] and Globus [5].

Access to the server is controlled using three ACLs, one for users without Kerberos, one for users with Kerberos, and one extra for submitting new jobs into the system. The last ACL is present to allow read-only access to the server from multiple realms (job submittion is currently allowed only with one Kerberos realm).

2.2. Resource semantics

Resource semantics represent an important part of a batch system. Computational jobs request (at least partial) guarantees concerning available resources. A batch system therefore has to do static allocations of resources, depending on the amounts of resources requested by individual jobs and refuse the execution of jobs that would breach the resource limits on computational nodes. This is, of course, coupled with the enforcement of these limitations, during the job execution (both on OS level and batch system level).

We are currently supporting multiple types of resources, the most common being counted resources on nodes. The scheduler is planning jobs according to the following resources on nodes: processors, memory, virtual memory, and GPU cards.

Another type of resource are dynamic resources. These represent the current state of resources that either change values so fast that it would be infeasible to track them directly, or can be affected from outside of the grid. Into this category fall software licenses and scratch disk space.

Original Torque implementation did not support any resource semantics with the exception of CPU counting. Resource semantics were left to be implemented in the scheduler, which then became the authoritative part of the system.

Since we are supporting a distributed configuration with multiple schedulers, we had to re-implement resource semantics in the server. Each server is responsible for

it's own computational nodes and can therefore act as a guardian and block requests that do not fit the current state of the clusters.

This is also important for performance reasons because it would be infeasible to enquire the state of nodes by connecting to each of the nodes. This was the original scheduler model in Torque. The server in the role of a guardian can provide all state information required by the scheduler in one status request.

GPU and other physical cards. With the inclusion of GPU enabled clusters into MetaCentrum, we had to cope with the issues of scheduling GPU and eventually other physical cards/devices.

Resource semantics for GPU cards are similar to processor cores, but unlike processor cores, we have to do strict system level enforcement of the GPU card assignments.

GPU cards (specifically NVIDIA) support multiple computing modes (thread exclusive, process exclusive, and shared). Unfortunately, different applications require different GPU modes to operate efficiently.

Setting all GPU cards into the process exclusive modes would prevent any two jobs sharing a GPU card, but we cannot do that due to the mentioned efficiency issues.

Therefore, instead of setting the cards into process exclusive mode, we are using UNIX file ownership to dedicate GPU cards to job owners, for the duration of the job execution. This way, jobs can choose the appropriate GPU mode, without interfering with other users jobs.

Over-subscribing of resources. While we generally do not support the over-subscribing of resources, we still want to cover one specific use case of resource over-subscribing; administrative and monitoring jobs are required to run on nodes even when all resources are already assigned to standard jobs.

For this specific class of jobs we have implemented a new feature called admin slots. Admin slots are configurable on a per-node basis and will add one special CPU slot for jobs submitted through a specially marked admin queue. Such jobs ignore all resource limitations and only consider the availability of the admin slot.

2.3. Limiting node access

Policies for accessing nodes can be configured using two paradigms. The first one is information about user accounts on nodes and clusters (job cannot run on nodes/cluster when users have no account there). The second, and currently preferred way of enforcing access policies are limitations of queues. This limitation can be enforced on both sides. Queues can be configured to access only a subset of nodes and nodes be exclusively assigned to a specific queue.

For preemption support in the virtualized infrastructure, we also support a configuration option to disallow these machines to participate in a multi-node job.

2.4. Extensions to node specification

To satisfy a user requirements for avoiding specific parts of the grid, we have implemented support for negative requests. Properties prefixed with "^" will be matched to nodes that do not have that property. For example, to avoid a specific cluster of machines, users can specify a negative request in the following form: `-l nodes=1:^cl_skirit`.

For an alternative format of specifying the required resources, we support exclusive node requests. These requests always match the entire node and allow the user to specify the properties of the requested node, instead of required resources. For example, instead of requesting a machine with 4 CPU (`-l nodes=1:ppn=4`), the user would request a node with `quadcore` property (`-l nodes=1:quadcore#excl`).

2.5. Torque evaluation

With almost a full year of production use of our modified Torque version, we are now capable providing an evaluation of Torque usability as a production ready batch system.

Our stable environment processed cca. 500 000 jobs during the evaluation period with a satisfactory uptime. Most system outages were caused by external events, such as DNS, Kerberos, NFS, or physical network/hardware outages.

Node outages were quite common, but Torque batch system is very tolerant towards node outages, therefore even prolonged downtimes did not affect the system and in some cases (software outages) Torque was even able to recover the jobs from affected nodes.

Of course the system also has some issues. We had to continuously work on stabilising and bug-fixing the system. Torque contains a lot of race conditions, which usually are not triggered in non-modified installations, but our extensions implemented into Torque caused these race conditions to manifest. Incorrect memory manipulation is also quite common (both memory leaks and out of bounds access).

Another issue with Torque adoption into a stable environment is the upstream policy towards stable branches. Even branches marked as fixes-only are still accepting new features, and so the overall stability of the system is very problematic to rely on.

Some of these issues can be avoided by using our fork of the Torque batch system. This fork contains the current stable version used in our production environment. The source code is accessible through a git repository on `http://github.com/CESNET/torque`.

3. Distributed architecture

A distributed architecture was designed to overcome the limitations of the central server setup, a detailed analysis was presented in the technical report [11] and the evaluation of the implementation integrated into Torque was presented at the Cracow Grid Workshop 2010 [13].

In this paper we will therefore only present a short overview of the distributed architecture with an update on the current state of design and implementation.

The proposed architecture (Figure 2) 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.

When a situation is encountered, which cannot be resolved locally, the schedulers cooperate to achieve the global goal. These situations range from simple local server saturation to cross-server job execution.

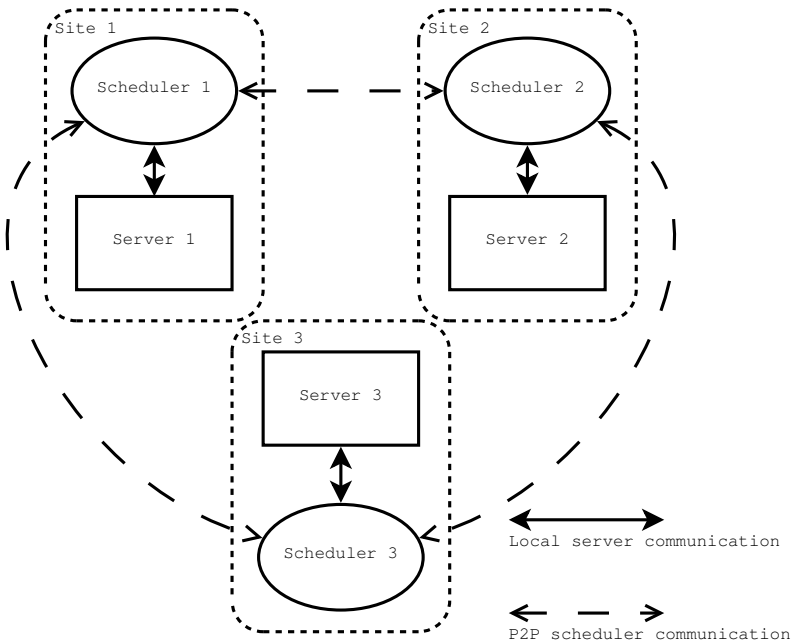

Figure 2. The distributed architecture with 3 sites

While this target architecture is designed to scale well for hundreds (up to thousands) sites, it does present big design and implementation challenges which have not been yet resolved.

As an intermediate solution a simplified version of this architecture was also presented. In this case, each scheduler maintains the full world state by communicating with each server, a performance analysis was presented on the CGW'10 [13] which led to the conclusion that this solution will scale satisfactory for at least tenths of sites.

The newest modification to the architecture is a new atomic move-and-run operation that reduced the amount of job movement in the system, limiting the maximum distance from the original server to one hop.

Current and future work is concentrated on bringing the intermediate architecture into the production environment. This a requirement for connecting the first site that is using a separate Torque instance, CERIT Scientific Cloud[4].

4. Dynamic Infrastructure

Cloud services are slowly gaining popularity even in the area of high performance computing. Although MetaCentrum is a purely non-commercial provider, we still have to compete with the services provided by commercial cloud providers to maintain our users.

From a grid provider viewpoint, the challenge of satisfying users in the advent of cloud computing are the following:

- software environment flexibility,
- easy integration with existing user workflows,
- support for third party computational systems,
- user requests satisfied promptly,
- without the need for additional negotiations.

While it is possible to satisfy user requests in the first three areas, this almost always requires the intervention of grid administrators and non-trivial negotiations between resource providers and users.

The only possibility to provide all the mentioned features in a prompt manner is to extend the infrastructure itself, so that it can support these requests without the need for manual intervention.

In this section, we will discuss our work at the dynamic infrastructure [9], and its support implemented into the Torque batch system.

4.1. Virtualized infrastructure

The initial implementation of virtualized infrastructure was originally introduced to provide preemption support on the machine level [3]. This feature was required to utilise resources connected into MetaCentrum.

Some resource providers are willing to share their computational resources, but require immediate access to these resources when requested. From a scheduling algorithm point, this is an impossible situation. Since new jobs can arrive at any point in time, there can never be any other running jobs utilising these resources.

By adding virtualization into the infrastructure we solve this issue using machine level preemption. Each physical node is split into three parts. The original physical node and two virtual nodes, each representing the entire node. One of the virtual machines is configured as low priority and one as high priority. Jobs can arrive into both of these machines, but when one of the machines has claimed the nodes resources, only the high priority jobs can preempt and reclaim node resources (see Fig. 3).

[4] `http://www.cerit-sc.cz`

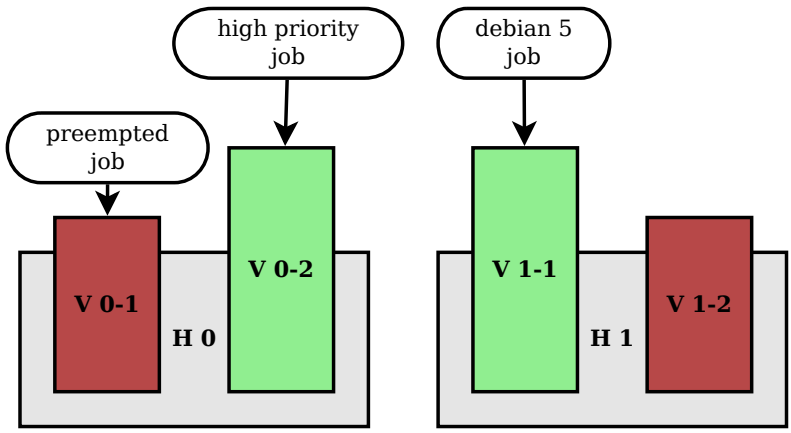

Figure 3. A state snapshot of two host machines, one configured for preemption, one configured for two OS versions

Using the same infrastructure we can provide support for two different OS versions on a single machine. In this case both machines can accept all jobs, but still only one virtual machine can claim the resources of the host at a time (see Fig. 3). In this way we can directly satisfy both users that require stable software and users that require fresh versions.

4.2. Virtual clusters

The virtualized infrastructure does allow for two operating systems available on a single machine. Although that is enough to provide two versions of a single operating system, it is not flexible enough to cover the entire user base.

Users requiring custom images require quick deployment and a unified initial state from which these images are executed. To cover this use case, we have implemented support for virtual clusters [12] on top of our virtualized infrastructure.

The machine configuration remains the same. Each host is still supporting two virtual machines, each of which represents the entire host. The new element is that one of these machines starts in an offline state, in which it can be reinstalled and booted up.

Provided software images are selected from a set of pre-registered images. Registering a new image into the system still requires the assistance of grid admins.

We are currently supporting two types of software images: Torque enabled software images, which after boot-up connect back to the Torque server and allow job submittion into the virtual cluster and custom images, that usually connect back to the users infrastructure.

The first type of images is mostly used to provide a dedicated virtual clusters for a user or a group, but in this case the software image has to be Linux based. The

second type of images is mostly used to provide a scalable computational architecture for users with special requirements. In this case, there are no requirements on the software image and we are currently supporting a group with a MS Windows based image.

For users that want network separation, or want the virtual cluster to connect back into their network infrastructure, we provide the possibility to create VLANs (both L2 VLAN and VPN) [4] and connect the constructed virtual cluster into these VLANs.

4.3. On-demand virtual clusters

Using virtual clusters, we are able to provide a wide range of Linux based software images. Unfortunately, for users that just want a specific software image that will connect back into the Torque server, and do not require any kind of network separation, virtual clusters are too complicated to use.

For this specific use we are able to provide a simplified version of virtual clusters. We can deduce the requested image directly from the jobs properties and can build the virtual cluster on the background, without involving the user.

The scheduler is capable of deciding whether a machine has to be rebooted with new software image, or should remain as is. Leaving these decisions to the scheduler we can also provide the balancing of installed software images. For example we can require that Debian 5 will be installed on at least 30% of the machines, while Scientific Linux will be installed on no more then 15%.

4.4. Light virtualization

All the previously mentioned examples rely on heavy duty virtualization technology like XEN to be installed on the host machine. Unfortunately, such technology is not optimal for all types of machines. For example, machines with GPU cards or SMP machines cannot be effectively virtualized using heavy duty virtualization technologies (XEN [1], KVM[5]).

The virtualization of physical cards like infiniband or GPU cards is a common problem across all virtualization technologies. Although modern CPUs support virtualization technologies (Vt-d, AMD-Vi) that allow the host system to pass a physical card into a Guest system, software support for this feature is still lacking production grade quality, especially in the area of GPU cards.

The solution to this problem that we are exploring is based on the LXC technology. A novel Linux based virtualization technology that is using system level encapsulation instead of hardware level virtualization.

[5] `http://www.linux-kvm.org/page/Main_Page`

LXC[6] provides an alternative to classical virtualization methods. Unlike XEN, KVM and other heavy weight virtualization technologies, LXC is not based on hardware virtualization, but instead on process separation provided by CGROUPs[7].

While not providing the hardware level separation, LXC is still providing full encapsulation at the system level. This has the advantage of extremely low overhead, therefore allowing for the concurrent execution of thousands of virtual machines on a single desktop grade machine. Due to system level encapsulation, it also provides a trivial solution for physical devices passing from host to guest.

5. Future work

Throughout this paper, we have already mentioned several areas that will be advanced in the near future. Distributed architecture will be moved into a stable environment, with the first organisation being connected into the Czech NGI in this matter. Our work with cloud-grid convergence will continue, with on-demand cluster creation entering the stable environment and also with the introduction of virtualization to machines, where it was previously impossible (using technologies like LXC).

Our work on the virtualized infrastructure does not end with the currently provided features. We are also already working on new features that will remove the remaining limitations of our infrastructure. One of the limitations of our infrastructure is the base design that splits each physical machine into three parts, without the possibility to allocate only a subset of resources for one of the virtual machines. To move away from this limitation, we are reworking our infrastructure into a more dynamic setup, where virtual machines can be both pre-configured statically, but can also be constructed on-demand from resource pools. This will allow us (in the extreme) to provide a fully encapsulated environment for each job, while the users will have the ability to specify the desired software image for this job, along the standard resource requests.

Another area that requires immediate attention is the detection of jobs that either cannot be run, or will be waiting in a queue for a significant amount of time (for example, due to low priority). This problem, while challenging on its own, becomes very complicated in a distributed environment.

6. Conclusion

In this paper we have presented our experiences with Torque batch system. With almost a full year of production use, we can now safely state that Torque is a very solid alternative to commercial systems (like PBSPro). While we had to put significant effort into modifying Torque to fully suit the needs of the Czech NGI, less demanding sites, can usually use Torque without any modifications.

[6] `http://lxc.sourceforge.net/`
[7] `http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt`

Acknowledgements

Access to the MetaCentrum computing facilities provided 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 is highly appreciated.

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] Barham P., Dragovic B., Fraser K., Hand S., Harris T., Ho A., Neugebar R., Pratt I., Warfield A.: *Xen and the Art of Virtualization.* [in:] *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[2] Chlumský V., Klusáček D., Ruda M.: *The Extension of Torque Scheduler Allowing the Use of Planing and Optimization Algorithms in Grids.*

[3] Denemark J., Ruda M.: *Magrathea – Scheduling Virtual Grids with Preemption.* [in:] *Cracow'08 Grid Workshop*, 2009.

[4] et al. D. A.: *VirtCloud: Virtualising Network for Grid Environments – First Experiences.* [in:] *AINA*, 2009.

[5] Foster I.: *Globus Toolkit Version 4: Software for Service-Oriented Systems.* [in:] *IFIP International Conference on Network and Parallel Computing*, pp. 2–13, 2006.

[6] Henderson R., Tweten D.: *Portable Batch System: External reference Specification.* NASA, Ames Research Center, 1996.

[7] Jackson D., Snell Q., Clement M.: *Core Algorithms of the Maui Scheduler.* [in:] *Proceedings of 7th Workshop on Job Scheduling Strategies for Parallel Processing*, 2001.

[8] Laure E., Hemmer F., Prelz F., Beco S., Fisher S., Livny M., Guy L., Barroso M., Buncic P., Kunszt P., Di Meglio A., Aimar A., Edlund A., Groep D., Pacini F., Sgaravatto M., Mulmo O.: *Middleware for the next generation Grid infrastructure.* [in:] *Computing in High Energy Physics and Nuclear Physics (CHEP 2004)*, 2004.

[9] Matyska L., Ruda M., Tóth Š.: *Peer-to-peer Cooperative Scheduling Architecture for National Grid Infrastructure.* [in:] *Data Driven e-Science*, pp. 105–118, 2011.

[10] Neuman C., Yu T., Hartman S., Raeburn K.: *The Kerberos Network Authentication Service (V5).* RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113.

[11] Ruda M., Tóth Š.: *Transition to Inter-Cluster Scheduling Architecture in MetaCentrum.* Technical Report 21, Cesnet, 2009.

[12] Ruda M., Šustr Z., Sitera J., Antoš D., Hejtmánek L., Holub P., Mulač M.: *Virtual Clusters as a New Service of MetaCentrum, the Czech NGI.* [in:] *Cracow'09 Grid Workshop*, 2010.

[13] Tóth Š., Ruda M., Matyska L.: *Towards Peer-to-Peer Scheduling Architecture for the Czech National Grid.* [in:] Bubak M., Turała M., Wiatr K., editors, *Cracow'10 Grid Workshop*, pp. 92–101. ACC CYFRONET AGH, Kraków, 2011.

Affiliations

Šimon Tóth
CESNET z.s.p.o., Zikova 4, 160 00 Praha 6, Czech Republic, simon@cesnet.cz

Miroslav Ruda
CESNET z.s.p.o., Zikova 4, 160 00 Praha 6, Czech Republic, ruda@ics.muni.cz

Received: 9.12.2011
Revised: 01.02.2012
Accepted: 23.04.2012