Distributed job scheduling in MetaCentrum

Šimon Tóth[1] **and Miroslav Ruda**[2]
CESNET a.l.e., Zikova 4, Prague 6, 160 00, Czech Republic
E-mail: [1] simon@cesnet.cz
E-mail: [2] miroslav.ruda@cesnet.cz

Abstract.
MetaCentrum – The Czech National Grid provides access to various resources across the Czech Republic. The utilized resource management and scheduling system is based on a heavily modified version of the Torque Batch System. This open source resource manager is maintained in a local fork and was extended to facilitate the requirements of such a large installation. This paper provides an overview of unique features deployed in MetaCentrum. Notably, we describe our distributed setup that encompasses several standalone independent servers while still maintaining full cooperative scheduling across the grid. We also present the benefits of our virtualized infrastructure that enables our schedulers to dynamically request ondemand virtual machines, that are then used to facilitate the varied requirements of users in our system, as well as enabling support for user requested virtual clusters that can be further interconnected using a private VLAN.

1. Introduction

The Czech National Grid – MetaCentrum is a highly geographically distributed grid providing both computational and storage resources.

The grid contains a heterogeneous set of resources, currently encompassing approximately 10000 CPU cores distributed across 550 nodes, with varied memory capacity (12GB..6TB) and other capabilities (GPU cards, infiniband, high-speed NFS access, SSD scratch,...). Storage resources mainly consist of 1 PB permanent storage and 27 PB of long term (tape) storage. Heterogeneity of the resources is mirrored in the user base and workloads. The system commonly facilitates large job submissions (up to 10000 jobs), or very long jobs (several months of expected runtime).

This paper provides an overview of major features deployed in the Czech National Grid to deal with the varied provided resources.

2. Resource management

Original Torque [1] did not contain significant resource management features apart from managing CPU cores. We have extended Torque feature set significantly with additional resources, such as memory, GPU cards, scratch disk space and software licenses. These resources can all be requested by users in a very detailed manner. Users even have the option to request heterogeneous jobs with multiple machines, each with different resource allocation. For example a user can easily make a request for a job running on one master node with 1 CPU core and 48GB of memory, 10 slave nodes, each with 16 CPU cores and 32 GB memory, all nodes connected using infiniband. Additional features include negative requests, selecting machines based on their performance, exclusive full node allocation, etc.

All resource requests are processed by the scheduler and jobs are only executed on nodes strictly matching the jobs requirements, with all requested resources exclusively allocated to this job. Users can therefore rely on these resource being available throughout the entire jobs execution. For certain resources, in particular software licenses, we cannot provide a strict guarantee, as these resources can be exhausted without the intervention of the scheduler (from outside of the grid). For this style of resource we employ a heuristic to determine when a particular job requesting such resource can be safely executed.

To guarantee fair access to the provided resource, we are employing a complex fairsharing scheme [8] that takes into account utilization of CPU cores and memory [6]. The fairshare scheme is based on a dynamic ordering policy that orders jobs (across queues) based on their users historical resource consumption (with more recent consumption having a higher weight).

3. Distributed scheduling

MetaCentrum has experienced significant growth in the past five years (from 1500 CPU cores in 2009 to over 10000 CPU cores in 2014). Naturally this growth demanded changes in the scheduling infrastructure as the original centralized solution reached its scalability limits.

To address the scalability issue we went through multiple iterations of a distributed architecture [12, 13, 10], with the final version [11] targeting not only scalability, but also providing better resilience against local outages and giving resource providers better control over the local policies they want to enforce over their hardware.

The model is based on the notion of global queues, which can span across multiple servers (see Fig. 1) and are used to distribute information about jobs across these servers. Each scheduler is responsible only for the resources of its local server and follows all locally imposed policies. When a global queue is encountered it is treated the same way as any other queue, with the scheduler also pulling in information about jobs in the remote (on other servers) parts of the global queue. Remote and local jobs are all treated in the same manner.

Cross-server execution is handled in the background (between the servers), with the scheduler only issuing a standard run requests. For the convenience of the users, cross-running jobs have their execution information synchronized between the execution and the originating server.

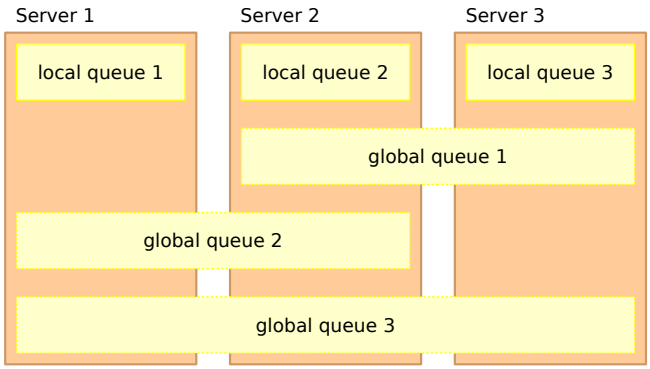

Figure 1. Distributed configuration utilizing global queues spanning across different servers.

This model puts very little constraints on the sites connected to the distributed system as they only need to run a compatible version of the Torque Batch System and can even use a different scheduler than the rest of the system as long as this scheduler is patched with the global queue support.

4. Virtualized infrastructure

Based on the Xen virtualization platform [2] and utilizing our Magrathea abstraction layer [4], the virtualized infrastructure was originally designed to provide full machine preemption. In the simplest case a physical machine is configured to host two virtual machines that both encompass the complete resource pool provided by the machine and are capable of switching resources between themselves (see Fig. 2). One of these machines is dedicated to low-priority preemptible jobs, one is dedicated to high-priority jobs and can preempt (take over the resources) from the low-priority virtual machine. This allows high-priority jobs immediate access to resources while heavily simplifying the scheduling process.

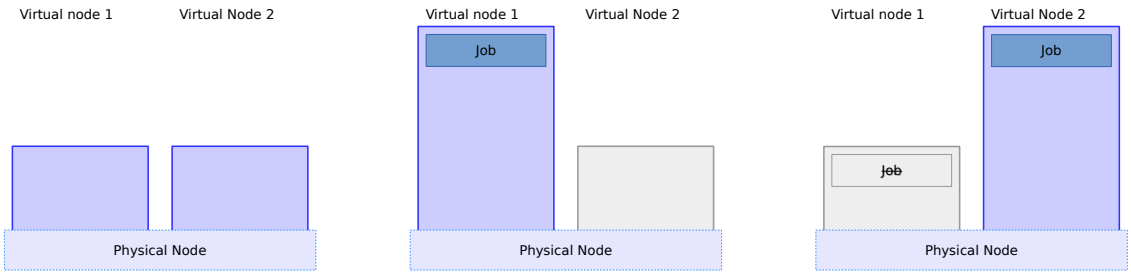

Figure 2. Example of a full machine preemption. Low-priority job claims all resources of the physical machine, but is then preempted by a high-priority job.

An extension of this basic model is the ability to start ondemand virtual clusters on top of our infrastructure. In this case, one of the virtual machines is left in an offline-bootable state and the scheduler is left with the option to start a new virtual machine with the desired (depending on the current overall users' demands) virtual image (see Fig. 3). This allows us to provide a more varied range of software configurations on the grid, without the need to statically dedicate a portion of the machine pool to a particular configuration.

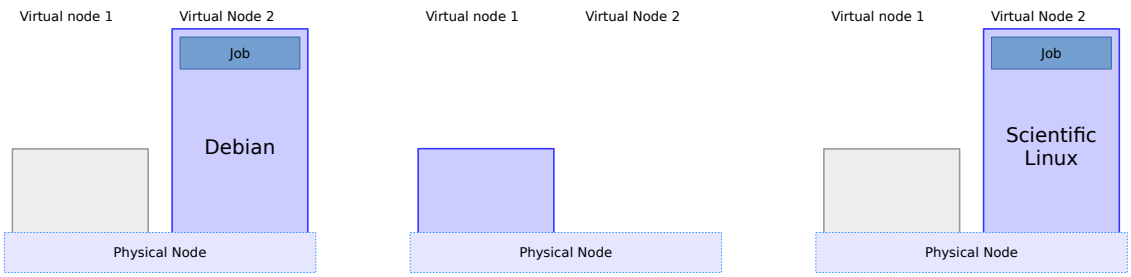

Figure 3. Example of an ondemand cluster. Once an instance with Debian is freed up, the scheduler boots up another instance with Scientific Linux.

Virtual clusters can also be requested directly by the users (including the possibility to connect the machines in the cluster using a VLAN). In this case the user can either request a virtual cluster that connects back to the resource management system, effectively creating a resource reservation, and specifying what users can access this prebuilt virtual cluster. For more complex use cases we are capable of providing a prebuilt image that connects to a preconfigured VLAN, extending the users computational infrastructure with more resources. This use case can cover even users that require access to Windows instances.

5. Management features

Extensive set of features, particularly important for the grid operator, was also implemented into Torque. This includes architectural features such as Kerberos [3] authentication with Kerberos based ACL for both job submission and system management and user oriented features such as the detection of invalid jobs that will never be executed under the current system configuration (these can be both the result of user error, or a maintenance event).

To maintain control over the systems behaviour, we have significantly extended the per-queue, per-group and per-user limits. Queues can be now limited to a particular subset of resources, can be limited in how many CPU cores they are allowed to occupy at any time, which extends to per-group and per-user limits.

To allow easy maintenance of machines, we are providing a dedicated maintenance queue, which disallows any other queues from accessing machines assigned to this queue. Furthermore, future maintenance can be specified for each node in the form of a deadline. No jobs that with runtime exceeding this specified deadline will be allowed to run on this particular node.

For testing and monitoring purposes it is sometimes required to submit small jobs to machines that are already fully utilized. Since we are heavily relying on exclusive resource allocation model, this would not be generally possible. To maintain this convenience, we have extended Torque with the support for admin slots. These represent an additional virtual CPU core that can be used to process small jobs (access to admin slots is limited to grid management and monitoring).

6. Conclusion

In this paper, we have presented an overview of important features that were implemented into our local fork of the Torque Batch System to facilitate the requirements of the Czech National Grid. Our resource management extensions allow users great control over what particular resources will be assigned to their jobs. Support for distributed scheduling allows us to divide the system into more manageable sites while still maintain global scheduling support and quality. Virtualized infrastructure gives our system the flexibility to deal with more exotic user requirements, be it infrequently utilized system configurations (through ondemand clusters)

or dedicated virtual clusters. Our fork of the Torque Batch System is fully opensource and freely available at `https://cesnet.github.io/torque/`.

Acknowledgements. 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] Adaptive Computing Inc. 2014 Torque batch system `http://www.adaptivecomputing.com/products/open-source/torque/`
[2] Barham P, Dragovic B and Fraser K, Hand S, Harris T, Ho A, Neugebar R, Pratt I and Warfield A 2003 Xen and the art of virtualization *SOSP*
[3] Neuman C, Yu T, Hartman S and Raeburn K 2005 The kerberos network authentication service *RFC*
[4] Ruda M, Šustr Z, Sitera J, Antoš D, Hejtmánek L and Holub P 2010 Virtual Clusters as a new service for MetaCentrum, the Czech NGI *CGW'09* pp 64–71
[5] Klusáček D, Rudová H and Jaroš M 2014 Multi resource fairness: problems and challenges *JSSPP'13* To appear.
[6] Klusáček D and Rudová H 2014 Multi-resource aware fairsharing for heterogenous systems *JSSPP'14* To appear.
[7] Tóth Š and Klusáček D 2014 User-aware metrics for measuring quality of parallel job schedules *JSSPP'14* To appear
[8] Klusáček D and Tóth Š 2014 On interactions among scheduling policies: finding efficient queue setup using high-resolution simulations *Euro-Par'14* pp 138–149
[9] Tóth Š and Klusáček D 2013 Tools and methods for detailed analysis of complex job schedules in the Czech National Grid *CGW'13* pp 83–84
[10] Tóth Š and Ruda M 2012 Practical experiences with Torque meta-scheduling in the Czech National Grid *Computer Science* **13** 2 pp 33-45
[11] Tóth Š 2013 Peer-to-peer scheduling in the Czech National Grid and beyond *CGW'13* pp 87–88
[12] Matyska L, Ruda M and Tóth Š 2011 Peer-to-peer cooperative scheduling architecture for national grid infrastructure *Data Driven e-Science* pp 105–118
[13] Tóth Š, Ruda M and Tóth Š 2011 Towards peer-to-peer scheduling architecture for the Czedch National Grid *CGW'11* pp 92–101