Optimizing Resource Utilization in Grid Batch Systems

Andreas Gellrich[1]

DESY, Notkestr. 85, 22607 Hamburg, Germany

E-mail: Andreas.Gellrich@desy.de

Abstract. On Grid sites, the utilization of computing, storage, and network requirements of the Grid computing tasks (jobs) differ widely. For instance *Monte Carlo* production jobs are almost purely CPU-bound, whereas physics analysis jobs demand high data rates. In order to optimize the utilization of the compute node resources, jobs must be distributed intelligently over the nodes. Although the jobs resource requirements cannot be deduced directly, jobs are mapped to POSIX UID/GID according to their VOMS-proxy. The UID/GID allows to distinguish jobs, assuming VOs make use of the VOMS group and role mechanism. It is possible to set-up and configure batch systems (queuing system and scheduler) at Grid sites based on this considerations. Moreover, a light-weight implementation of a home-grown scheduler was developed and tested.

1. Introduction

Grid computing has become the key technology for providing computing resources in e-science. The LHC [1] experiments depend on the world-wide *LHC Computing Grid* (WLCG) [2] which was set up in the context of the EU-project EGEE [3] and which is currently operated by EGI [4]. The same Grid infrastructure is being used very successfully by groups in other fields of e-science such as non-LHC High-Energy Particle Physics (HEP) experiments, astro-particle physics, life-sciences, and others. These collaborations and groups founded *Virtual Organizations* – a key concept in Grid Computing [5] – in which the provision and usage of resources is organized based on common sharing rules.

The EGI Grid site *DESY-HH* [6], which is located at DESY [7] in Hamburg, Germany, operates a multi-VO Grid infrastructure and serves as a Tier-2 centre for the ATLAS and CMS experiments within WLCG. The Grid resources are federated among ≈ 20 VOs. They are opportunistically used according to the agreements and contracts such as the WLCG Tier-2 *Memorandum of Understanding* (MoU) in which certain *pledges* per VO are defined. DESY-HH, as one of the world-wide largest Tier-2 centres, provides a total of 4800 job slots.

In order to use the computing resources most efficiently with respect to the usage of CPU cycles, physical and virtual memory, local disk space, and network bandwidth, the computing tasks *jobs* must be distributed intelligently on compute nodes by the batch system.

We describe here the concept and the technical means to achieve stable operations and guaranteed shares at maximal occupancy of the computing resources.

[1] for the Grid Team at DESY

Table 1. Job classification.

Purpose	Class	Submitter	Characteristics
Monte Carlo	Production	central	CPU-dominated, little input, moderate output
Analysis	Analysis	users	I/O-dominated, massive scratch disk usage

Table 2. VOMS mapping for CMS with arbitrary pool account numbers.

purpose	VOMS	UID	GID
user analysis	/cms	cmsusr007	cmsusr
German user	/cms/de	cmsger032	cmsger
priority analysis	/cms/role=analysis	cmsana010	cmsana
software manager	/cms/role=lcgadmin	sgmcms	cmssgm
MC production	/cms/role=production	cmsprd099	cmsprd
pilot jobs	/cms/role=pilot	cmsplt029	cmsplt
unknown to site	/cms/any/role=any	cmsusr199	cmsusr

2. Jobs

In HEP the computing requests (*jobs*)are parallelized at the job-level because applications do not (yet) make use of multi-threading technologies and/or use multi-core architectures. Hence jobs are independent and self-contained and can be treated individually.

2.1. Job classes

In the computing models of the LHC experiments [1], Tier-2 centres are responsible for *Monte Carlo* production and for user specific analysis. Those *job classes* have fundamentally different characteristics with respect to their resource requirements as listed in table 1.

2.2. Job identification

Jobs are submitted to the *Computing Elements* (CE) which act as gateways to the Grid site's batch system. The jobs contain the credentials of the job submitter which are encoded in the X509-formatted VOMS-proxy. Individual users as well as centrally managed submitters make use of VOMS groups and roles [8] which are managed by the VOs.
If such VOMS groups and roles are consequently used, job classes can be identified.

2.3. Mapping

The CEs map the VOMS-proxy to configurable site-specific local accounts (POSIX: UID/GID). The accounts are chosen uniquely from pools to ensure individuality per user. A typical example for the VO CMS is listed in table 2. These unique accounts and groups are used throughout all subsequent processes and allow to distinguish classes of jobs. Caution is required with so-called *pilot* jobs. Such jobs are submitted from a central agent using a standard VOMS-proxy with the role 'pilot'. The real ownership of the job's pay-load, which is started by the pilot job, is not revealed before run-time and precludes the batch system from identifying the job class.

3. Resources

The computing resources are provided by *Worker Nodes* (WN) which are operated as batch clients of a batch system. Most Grid sites purchased hardware in stages over many years, which leads to rather heterogeneous set-ups. DESY-HH operates 400 WNs including 8 to 24-core Intel Xeon machine with and without hyper-threading (HT) as well as 48-core AMD Opteron hosts leading to a total of 4800 job slots delivering 38 $kHS06$. For each job slot 2 GB of physical plus 2 GB of virtual memory and 20 GB of local scratch space is provided. Each WN is equipped with a 1 $Gbit/s$ network link. The optimization the utilization of the computing resources incorporates three aspects:

3.1. Operations

The main goal of any Grid site is to guarantee stable operations. If farm nodes crash due to exhausted resources, usually many more jobs than the responsible one are effected; at least all jobs on the particular node.

3.2. Utilization

In order to minimize costs for procurement, maintenance, and operations, resources should be utilized efficiently. A good measure is the ratio of cpu-time and wall-time which is preferably close to 1. Even if individual jobs have good efficiency, running many of them on one WN might lead to performance losses:

- Too many I/O intensive jobs per WN might exhaust the network and leave the CPU idle.
- Many CPU dominated jobs per WN may lead to a waste of network bandwidth capacities.
- Jobs which extensively exploit the scratch space on the local disk might cause high CPU wait times for all jobs on that node.

3.3. Pledges

In particular large VOs such as the WLCG Tier-2 VOs have MoUs with the sites to specify resource requirements and to define pledges per VO. This includes the number of guaranteed job slots normalized to specs as well as memory and scratch space per slot. Central accounting information is checked against the pledges monthly and sites are notified of imbalances.

4. Batch Systems

Jobs are submitted to the batch system by the site's CEs using the POSIX UID/GID of the mapping. Batch systems queue and schedule jobs according to configurable rules based on the jobs' UID/GID. DESY-HH as well as many WLCG Grid sites deploy *PBS/torque* [9] as the queuing system with the scheduler *maui* [10]. Both these open source products are supported by the gLite/EMI [11][12] middleware.

4.1. Queuing

At DESY-HH a configuration with four queues was chosen to allow for easy operations and maintenance which contains one for the each Tier-2 VOs (ATLAS, CMS), one for all other VOs, and one for monitoring purposes. Table 3 lists the queue properties. This set-up allows to manage the two big VOs independently.

4.2. Scheduling

The simplest ansatz is a first-in-first-out (FIFO) algorithm with only one queue. The scheduler would simply try run the jobs in reverse order of appearance. Obviously it is not possible to give priorities in the order or number of jobs of the various VOs and groups in this scenario.

Table 3. Job queues with the limits.

Queue	VO	cpu time	wall time	max running	max queuable
atlas	atlas, ops	60 *h*	90 *h*	4000	10000
cms	atlas, ops	60 *h*	90 *h*	4000	15000
desy	non-Tier-2	60 *h*	90 *h*	4000	10000
operations	ops	15 *min*	1 *h*	50	1000

Moreover, a congestion of similar job classes on a WN could not be avoided. Hence, schedulers have to act more intelligently to obey sharing rules and achieve efficient distribution of jobs over the WNs.
The scheduler *maui* was configured to guarantee a long term distribution of the resources based on the used wall-time according to the pledges [1]. *maui* also took care of the job class mix per WN by limiting the number of jobs of the same job class in terms of GID per WN. This set-up was able to handle up to 4800 running jobs plus up to 10000 queued jobs waiting for execution. Typically an occupancy of above 90% of the job slots was achieved and resource bottlenecks at the WNs could be avoided.

5. The *MySched* project

Long term experiences have shown that the scheduler *maui* can be configured to apply sharing rules for VOs and groups and concurrently distribute jobs among the WNs for optimal resource utilization. It turned out though that the complexity of the scheduling algorithm of *maui* as well as the many configuration options could hardly be controlled.
At DESY-HH the performance of *maui* did not scale beyond 10000 queued jobs which lead to blocked queues and unused job slots. In view of the increasing demand for computing resources by the LHC experiments which will lead to a significant increase of the number of WNs and job slots, an alternative to *maui* was investigated.

5.1. Considerations

DESY-HH has been studying a home-made scheduler implementation which is based on the C-API of *torque* and replaced *maui* (working title: *mysched*). It is tailored to the rather simple use-case of HEP which currently does not make use of parallelization on application level and multi-core jobs. This ansatz avoids the need for node reservations which complicates the scheduling process considerably.
The following requirements were made to *mysched*:

- Achieving hundred percent occupancy.
- Being light-weight, e.g. short processing times.
- Guarantee VO and group specific shares.
- Fairly distribute free job slots.
- Optimize resource utilization by intelligently distributing jobs to the WNs.
- Allow to configure limits, shares, and distribution algorithms.

Two step scenario:

- Ordering jobs in a list according to the target shares.
- Finding the best node job-by-job from the job list by applying limits and conditions.

A first implementation was written in C++ and contains roughly 4000 lines of program code. Limits and shares are read at run-time from a configuration file. The scheduler process runs is automatically started every 2 minutes. The scheduler logs comprehensive information for monitoring purposes.

5.2. Results

The test implementation met the requirements denoted above. Full occupancy at reasonable processing times (O(30s)) could be achieved. The system was successfully operated with up to 4800 running plus 20000 queued jobs. Figure 1 shows the job slot usage per VO with *maui* and *mysched*.

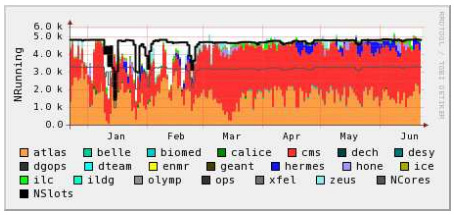

Figure 1. Job slot usage per VO with *maui* (until Feb) and *mysched* (from Mar on).

6. Summary

The classification of jobs due to their requirements on the WNs is essential for an optimal utilization of computing resources in Grid farms. The key to job classification are the user credentials of the jobs which can be mapped to distinguishable local accounts. On the basis of these accounts it is possible for batch systems to intelligently distribute jobs to the compute nodes in order to avoid bottlenecks and exhaustion of local resources.

It is obvious though that the usage of pilot jobs may spoil the described mechanisms if the job class remains hidden to the batch system.

Our home-grown implementation of a light-weight scheduler which is tailored to the needs of HEP jobs, will enable DESY-HH to meet the computing demands of the experiments at optimal occupancy and resource utilization.

Acknowledgments

We would like to thank the DESY Computer Centre for their support in installing and operating the Grid infrastructure.

The EGI project is co-funded by the European Commission's 7th Framework Programme (contract number: RI-261323).

References

[1] http://cern.ch/lhc/
[2] http://cern.ch/lcg/
[3] http://www.eu-egee.eu/
[4] http://www.egi.eu/
[5] http://en.wikipedia.org/wiki/Grid_computing
[6] http://grid.desy.de/
[7] http://www.desy.de/
[8] http://www.globus.org/grid_software/security/voms.php
[9] http://www.adaptivecomputing.com/products/open-source/torque/
[10] http://www.clusterresources.com/products/maui-cluster-scheduler.php/
[11] http://glite.web.cern.ch/
[12] http://www.emi.eu/