Job Scheduling in Grid Farms

A. Gellrich for the Grid Team at DESY*, Germany

DESY

DESY, a member of the German Helmholtz Association (**HGF**), is one of the world-wide leading **centers** for research with **particle accelerators** and **synchrotron light**.

DESY is a **WLCG Tier-2** center for LHC experiments **ATLAS** and **CMS** and participates in the EU-projects EGI in the federation NGI_DE.

Grid at DESY

The Grid site **DESY-HH** is the home to 10 VOs and supports in total 20 VOs, incl. ATLAS and CMS. All VOs are using one common EMI3-based Grid infrastructure.
In addition to the **7724** job slots (2GB mem/slot, 15GB scratch/slot) with a total of **62kHS06** and the 3 dCache-SEs with a total of **4PB** of disk space, all Grid services which make up a complete Grid infrastructure are provided, incl. multiple instances of **BDII**, **LFC**, **PX**, **ARGUS**, **VOMS**, and **WMSLB**.
The Grid services run in virtual machines hyper-vised by Xen.

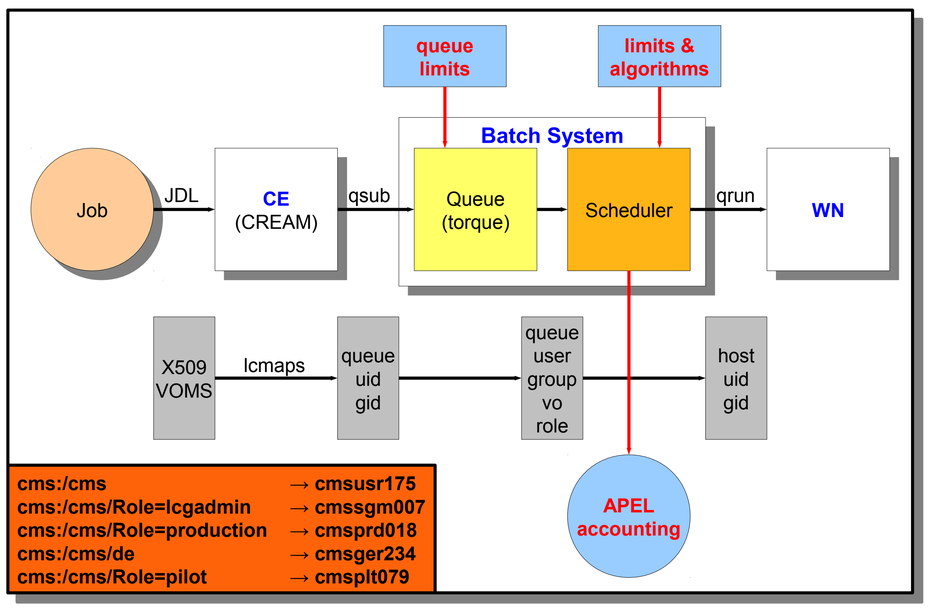
Computing Resources

Stable operations:
Avoidance of crashes due to exhausted resources, usually many more than jobs the causing one are effected. Crucial components are memory usage, network utilization, and local scratch space usage.

VO-requirements:
In particular large VOs have contracts (MoU, VO-cards) with participating sites in which resource pledges are specified. This includes a number of job slots normalized to specs, memory per slot, and local scratch space per job.

Worker Nodes (WN):
The Farm is heterogenous containing nodes with 16 to 64 cores, >=3GB/core memory, and >=20GB/core disk space.

Resource utilization:
In order to run a Grid resources efficiently, all job slot should be occupied and the cpu-time / wall-time ration should be close to 1. Since resources are not independent, bottlenecks must be avoided, e.g. massive local disk I/O might leave the CPU idling as well as the usage of swap space due to exhausted memory.



Classifying Jobs

Jobs can be classified by their resource requirements to CPU, memory, network, and local scratch space in two main classes:

MC jobs:
CPU-bound, little input, moderate output
organized and submitted centrally

Analysis jobs:
I/O-bound, stream data files, big output on disk
coded and submitted individually

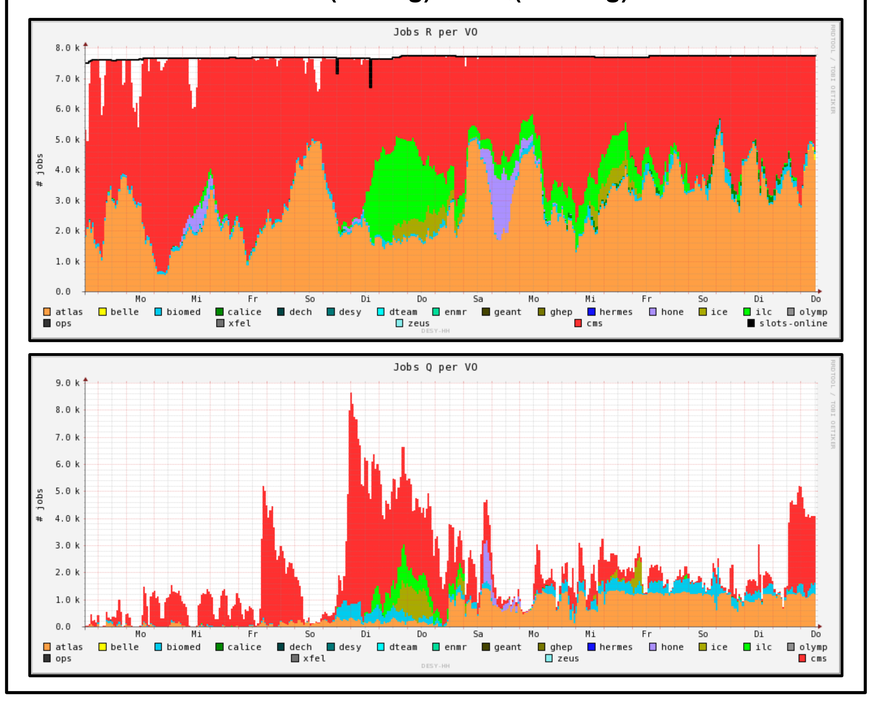
Mapping Jobs

In the Grid, jobs are sent to the batch system by means of Computing Elements (CE). *Authentication* and *authorization* is based on **X509** proxies with **VOMS**-extensions. Job submissions contain the user's **VOMS**-proxies which are mapped to **POSIX** users/groups by the CE. Jobs are then submitted to the batch system with user/group credentials.
The mapping of the **VOMS**-proxies to **POSIX** uid/gid is the key to distinguish job classes.

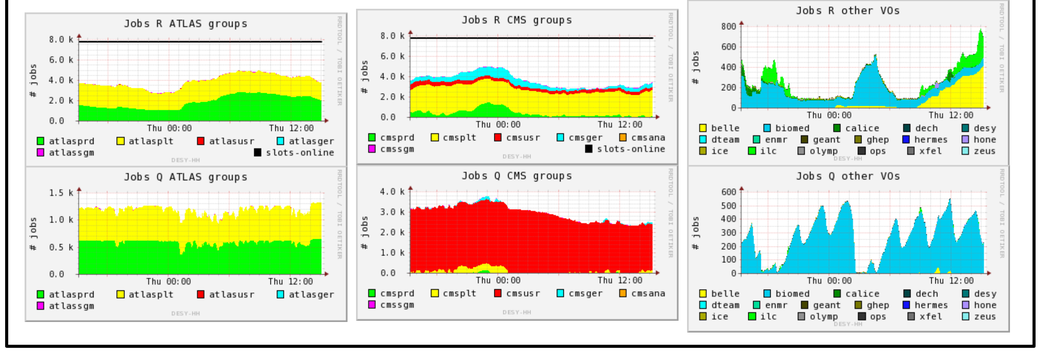
The Batch System

The batch system consists of two parts: The job *queuing system* and the *scheduler*. In gLite / EMI the combination **torque**/**maui** is widely used. It is possible though to use the C-API of torque to implement an custom schedule. In the past, DESY-HH had continuous problems to concurrently guarantee stable operations and maximal occupancy. It was impossible to configure the scheduler *maui* appropriately.
In February 2012 DESY-HH started to test a simple home-grown scheduler which make use of the PBS C-API. It is tailored to the current needs at DESY-HH and allows to configure limits and shares for the VOs and groups. A set of simple algorithms creates mixtures of jobs according to their classification on the WNs in order to optimize the resource utilization.

Jobs R(unning) and Q(ueueing)



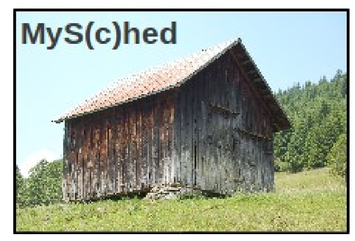
ATLAS, CMS, and other VOs: R(unning) & Q(ueueing) jobs



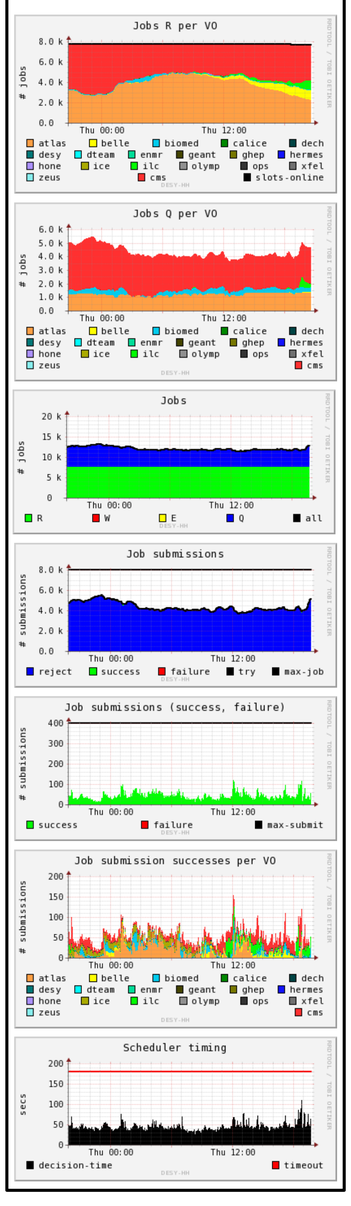
MySched: Algorithm

Jobs are handled on the basis of their **uid/gid**.
The *queueing system* applies limits to running and waiting jobs in the queues, typically one per VO.
The *scheduler* uses limits, priorities, and accounting data to intelligently distribute jobs to the worker nodes.

- Create node list
- Create job list
- Sort job list according to share
- Loop:
  - Treat job in list
  - Check each job for limits and rules
  - Find appropriate WN
  - Start job on WN
  - Update node list
- Find node:
  - online, not-busy, free slots, node limits
  - least occupancy and max diversity



Job Submission



MySched: Limits and Rules

General:
- jobMaxTotal (max number of jobs to be considered) [=8000]
- jobMaxSubmit (max number of jobs to be submitted) [=400]
- maxNodeSub (max number of job submits per node) [=4]
- musecSleep (sleep after submission in musec) [=100000]
- ndays (usage statistics of last n*24h) [=2]
- timeout (timeout after which scheduler quits in sec) [=180]

By-pass ('*hot*'):
- hotVo (*hot* VO) [="ops"]
- hotGroup (*hot* group) [="desytst"]
- hotRole (*hot* role) [="tst"]
- hotUser (*hot* user) [="cmsusr165"]
- hotQueue (*hot* queue) [="emi"]

Node:
- enable/disable (node)
- type (node type)
- queues (allow list of queues)
-

VO/group:
- enable/disable (whole VO or group)
- max (absolute number of jobs)
- fraction (maximal fraction of number of jobs of online slots)
- nodemax (maximal number of jobs per node=WN)
- nodefrac (maximal fraction of number of jobs per node)
- type (meet type of node)
- share (relative usage with past time interval)

Abstract-ID: 284, CHEP2013, Amsterdam, The Netherlands, 14-18.10.2013

*contact: http://gridcenter.desy.de/