Optimizing Resource Utilization in Grid Batch Systems

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, the successor of EGEE, in the federation NGI_DE.
DESY was founding partner of the German Grid initiative **D-GRID** and played a leading role in the HEP community project (HEPCG) and in the integration project (DGI-2).

Grid at DESY

The Grid site **DESY-HH**, which is the home to 10 VOs and supports in total 20 VOs, incl. ATLAS and CMS. All VOs are using one common Grid infrastructure.
In addition to the **4784** job slots (2GB mem/slot, 15GB scratch/slot) with a total of **38kHS06** 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**, **LB**, **LFC**, **PX**, **SCAS**, **VOMS**, and **WMS**. Most of the Grid services run in virtual machines.

Optimizing Resource Utilization

The site administrators view and the user's view might differ. Users and VOs focus on their specific needs and tend to ignore the existence of other users and VOs on the site.

Stable operations:
The main goal of any Grid site is to guarantee stable operations. If services die or compute nodes crash 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.

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.

An intelligent distribution of jobs to the compute nodes helps to optimally utilize resources.

Classifying Jobs

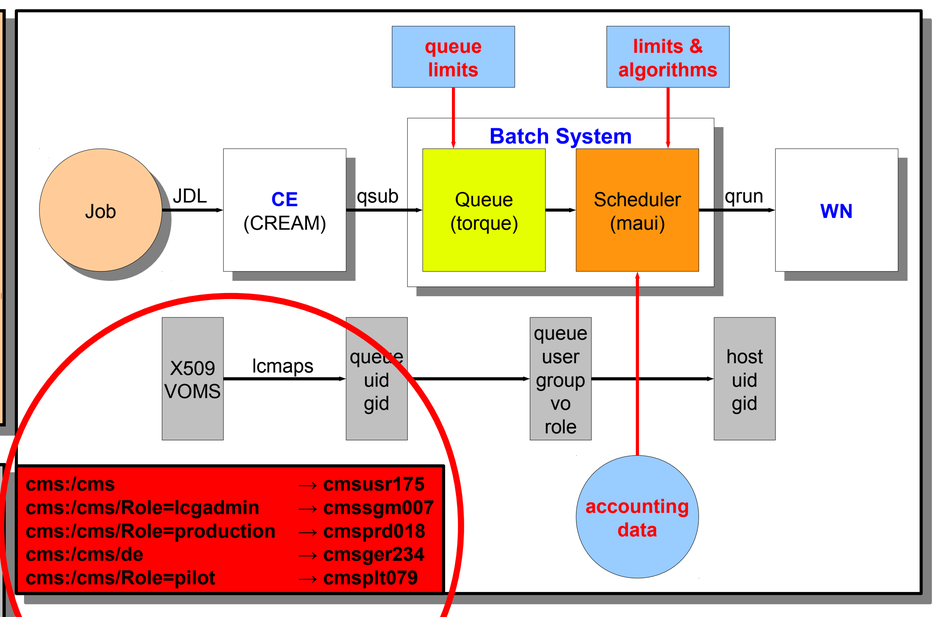
Jobs can be classified by their resource requirements to CPU, memory, network, and local scratch space in two main classes:

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.

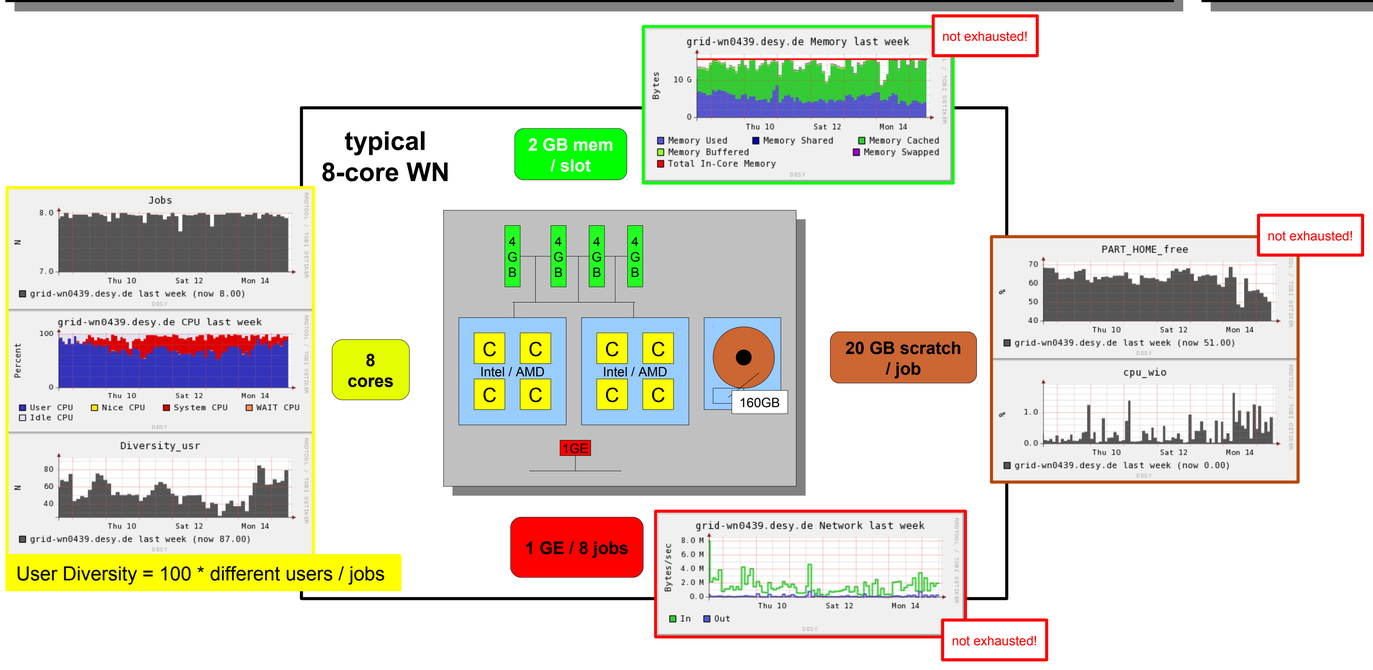


Limits, Algorithms, Data

Jobs are handled on the basis of their **uid/gid**.
The *queuing 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. The scheduler must be configurable.

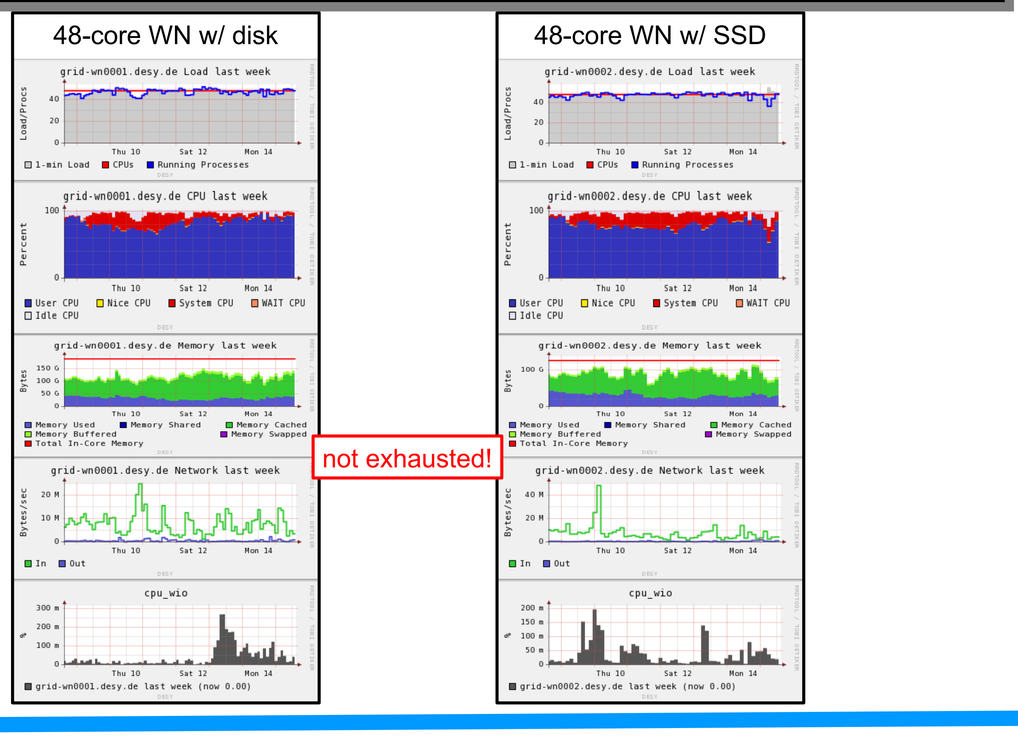
Resource Utilization

In a typical Worker Nodes (WN) with 8 cores and 8 job slots, a well-balanced mixture of jobs allows to efficiently utilize the computing resources (CPU, memory, network, disk), even if single jobs exceed specs. The hardware meets the specs, e.g. 8*2GB memory, 8*20GB scratch.



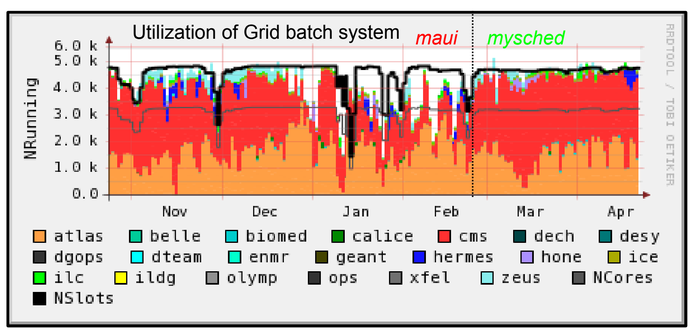
Heterogeneous WN Farm

WNs with 48-cores can be efficiently operated as well. The average memory and network utilization is on average well below the hardware limits which were chosen to meet the VO requirement per job, e.g. 48*2 GB = 96GB memory. SSDs can replace the disks to improve performance (cpu_wait_io).



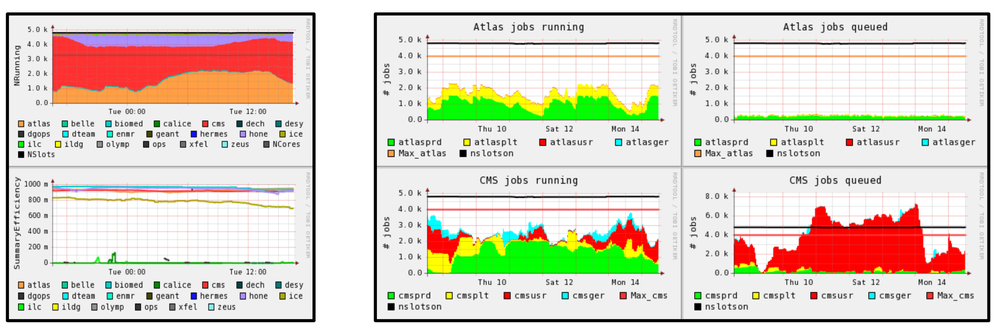
The Batch System at DESY-HH

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.



The *MySched* Attempt

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.



Abstract-ID: 290, CHEP2012, NYC, New York, U.S.A., 22.05.2012

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