



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS site report

Massimo Benini
Slurm User Group
September 23 - 24, 2014
Lugano





CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Agenda

- **Slurm history at CSCS**
- **Slurm at CSCS systems overview**
- **Current status**
- **Future work**



Slurm history at CSCS

- June 2010: First working port of Slurm to a Cray
- April 2011: CSCS go live with Slurm
- From Spring to Fall 2011: start the migration from PBS to Slurm
- November 2011: Rosa upgraded to XE6 running Slurm 2.3.0-pre5
- March – June 2012: The last PBS systems (Buin and Dole) are replaced with Albis and Lema running Slurm 2.3.4 (MCH systems)
- Fall 2012: preparing Slurm code for the upcoming XC30 System Piz Daint
- December 2012: Slurm successfully running on Piz Daint



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Slurm history at CSCS

- April 2013: Piz Daint Update, hybrid system (GPU-CPU) running in production with Slurm
- Spring 2013: General update to slurm 2.5.4
- March 2014: start testing new version 14.03
- April 2014: Common Compute Currency decommissioning
- Summer 2014: Cray Slurm Support contract
- October 2014: general site update to version 14.03.07

Slurm@CSCS, principal systems

1/2

Machine	Arch Type	# of nodes	# of cores	Node Layout	GPU	Node memory	Slurm version
Daint	XC30	5272	42176	1x8x1	5272 Tesla K20X	32GB	2.5.4
Rosa	XE6	1496	47872	2x16x1	None	32GB	2.5.4
Todi	XK7	272	4352	1x16x1	272 Tesla K20X	32GB	2.5.4
Julier	non-Cray	12	288	2x6x2	None	10-48GB 2-256GB	2.5.4
Pilatus	non-Cray	44	704	2x16x2	None	64G	2.5.4
Albis	XE6	72	1728	2x12x1	None	32GB	2.5.4
Lema	XE6	168	4032	2x12x1	None	32GB	2.5.4



Slurm@CSCS, principal systems

Machine	Arch Type	# of nodes	# of cores	Node Layout	GPU	Node memory	Slurm version
Castor	non-Cray	32	384	2x6x1	2 Fermi M2090 per Node	24GB	2.5.4
Dom & dommic (R&D cluster)	non-Cray	16	512	2x8x1	K20c, K20X, Xeon Phi (MIC)	32	14.03.2
Monch	non-Cray	360	7200	2x10x2	None	312*32GB 24*64GB 24*256GB	2.6.1
Phoenix	non-Cray	93	1104 * SNB 480 * IVB	2x8x2 2x10x2	None	69*64GB 24*128GB	2.6.9



The systems miscellaneous

- Ela—main gateway from outside to systems.
- External and Internal login nodes.
- Fojorina01/Fojorina02—Hosts the common slurmdbd for all principal systems. Associations limits.
- db.cscs.ch—hosts the central CSCS DB and SLURM DB.



The test systems

- TDS systems that mirrors production systems:
 1. Gele -> Rosa test system (iLogin and Elogin).
 2. Santis -> Daint test system (iLogin and Elogin).
- Dolomite cluster, a set of non-Cray blades, currently 4 nodes with 2 Sockets, 6 Cores per Sockets, 2 Threads per Core. Main development platform.
- Virtual Machines (Slurm simulators). Mainly used to simulate the workload and tune parameters. Slurm compiled with **-enable-multiple-slurmd** and **-have-front-end flags**.
- Currently developing scripts for taking screenshot of the current workload on a machine and reproduce within our simulator systems.



Slurm features used

- **Basic**

- partition/node configuration options
- backfill scheduler type
- Cray and cons_res modules for node selection
- Priority Multifactor

- **Plugins**

- Lua scripts
- def_partition (CSCS custom)
- get_group_prio (CSCS custom)

- **Others**

- Task/Affinity
- GRES (gpus)
- Prologues and Epilogues (set variables, manage GPUs, CCM)
- PAM module
- SPANK module
- Job chaining
- Least Loaded Node (some systems)

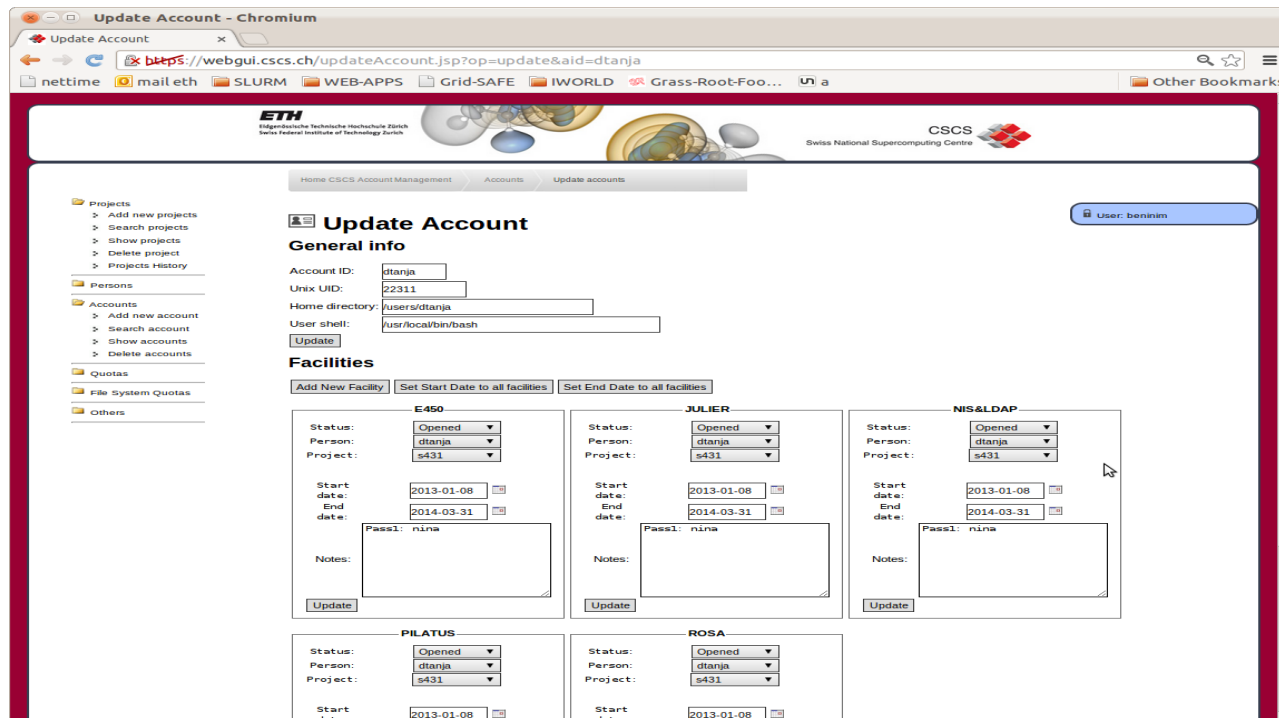


Slurm features NOT used

- Fairshare (except on Monch)
- QoS
- Gang Scheduling
- Preemption
- CGroups
- Command wrappers on Cray

Slurm ecosystem

- SLURM “ecosystem” consists of SLURM and various scripts and utilities built around it
- CSCS’s Web GUI to insert id’s, accounts, allocations... in the site's general DB and then propagate this data into slurm via cronscripts





Slurm ecosystem

cluster_populate_slurmdb.pl

- Migrate accounts, quotas, organizations, projects to SLURM DB
- Use SQL queries to retrieve data and *sacctmgr* to update SLURM DB

cluster_accounting.pl

- Data From SLURM DB to CSCS DB
- Run every hour on each system.

cluster_set_priority.pg

- Generate priorities flat file (run on every system roughly every hour)
- Factors which affect group priorities: project type, global quota, global quota used, local quota, local quota used and time left to the end of the allocation period.

<i>Group1</i>	<i>prioA</i>
<i>Group2</i>	<i>prioB</i>
....



CSCS's custom priority

- Maintain concept of local quota-usage for a given group. Per cluster-group allocation.
- Deny jobs from over-budget accounts or bottom-feeding jobs allowed (over-budget but have the lowest of possible priorities) depending on the cluster-group.
- Use cron script to periodically update priorities for pending jobs.
- Uses the “nice” value component of multifactor priority equation.



Future works

- “Vanilla” Slurm: try to use as much as possible Slurm embedded functionalities to manage group/users priorities. (e.g. QoS + Fairshare)
- Remove CSCS old customizations that are now available with Slurm.
- Now that we have a support contract -> work closely with Cray
e.g. CCM & MPI
- Slurm Intercluster Project.



Conclusions

- CSCS has diverse size systems. Successfully manage these resources with SLURM.
- Maintain a SLURM “ecosystem” of SLURM instances, DB and scripts to provide both.
- CSCS and users with the desired resource management functionality.



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Q+A
