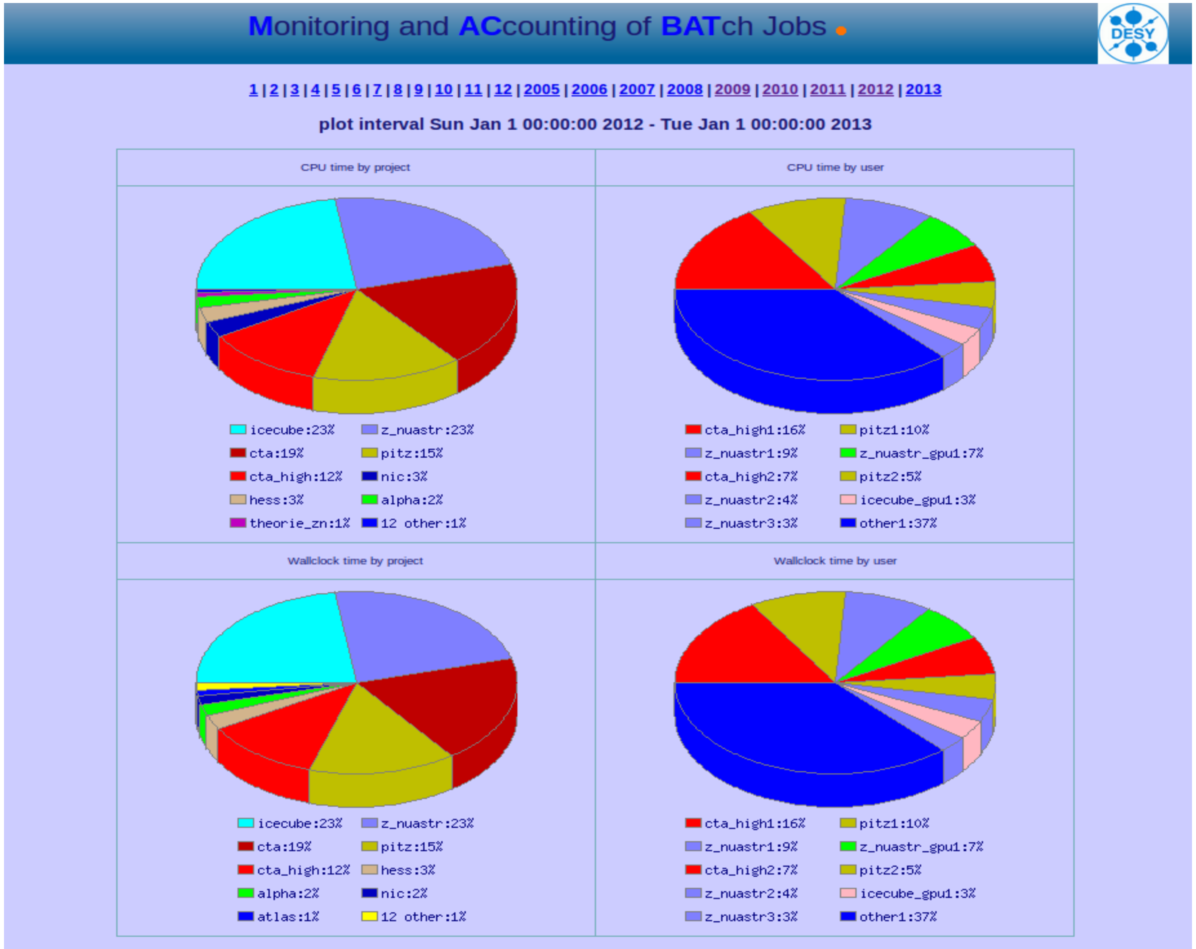
Monitoring and Reporting for Gridengine



Monitoring and ACcounting of BATch Jobs

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013

plot interval Sun Jan 1 00:00:00 2012 - Tue Jan 1 00:00:00 2013

Wolfgang Friebel
April 16, 2013

Contents

- Motivation
- Existing solutions
- Data sources
  - Accounting and reporting
  - Monitoring
- Monitoring and accounting at DESY
  - Used tools
  - Performance considerations
  - Privacy
  - Improvement plans
- Summary

Motivation

- Several batch farms in use at DESY (Bird, NAF, Parallel Farm, …)
  - Need to watch proper function and optimal utilization of the farms
- Several projects compete on the same farm for resources
  - Need to check actual usage of the farm in accordance to the plans
  - Need to check that hardware contributions of projects get properly used
  - Need to verify that the batch system cannot be tricked out
- Resources are precious and need to be optimally used
  - Need to check that users tune their jobs to not waste resources
- Users need to know the status of their waiting, running and ended jobs
  - Frequent queries for the job status affect the scheduling process
  - Interpretation of the raw numbers (e.g. in units of GB*s) and error codes difficult

Existing solutions

- Accounting and reporting
  - Commercial solutions
    - ARCO (Accounting and reporting console based on accounting/reporting files)
    - Unisight (for accounting/reporting, comes with UNIVA Gridengine, uses warehouse techniques)
  - Open source programs
    - Solutions using the ARCO database component (dbwriter) were reported on past GE meetings
    - Generic solutions in the grid context
- Monitoring
  - Commercial solutions
    - Unisight seems to allow monitoring of values reported by qhost (see later)
  - Open source programs
    - Xml-qstat (web based monitoring based on qstat)
    - Several scripts that summarize qstat information and produce text output

Monitoring and accounting at DESY

- No satisfactory open source solution found
  - Is maybe to trivial and everybody is having a home grown solution?
- Commercial solution not appropriate for us
  - Not useable for other batch systems in use (e.g. Torque, Son of GridEngine)
  - Does not fully cover job monitoring (to my opinion)
  - Is fairly complex
- A DESY approach
  - Existing since several years, served only the immediate needs
  - Small CGI program and a simple database in the beginning
  - Tried to improve and enhance the solution, but lack of manpower
  - Recently new concepts being tried (see talk of Th. Finnern)

Data sources for accounting

- File "accounting"
  - Written by gridengine on termination of jobs
  - Contains approx 45 variables characterizing a job (content did vary with GE version)
  - Interpretation of variable values not always straightforward (mem in GB*s, counting of cpu and wallclock time for parallel and array jobs, error codes)
- File "reporting"
  - Not written by default, its generation needs to be configured
  - Contains in addition to information above also records for submitted and started jobs
  - Further information recorded not easily useable for monitoring/accounting
- Usage of these data
  - File accounting not suitable for monitoring, as info is available at job end only
  - File reporting has limited use for monitoring, almost no info for running jobs
  - These files are the only source of information for ARCO and Unisight

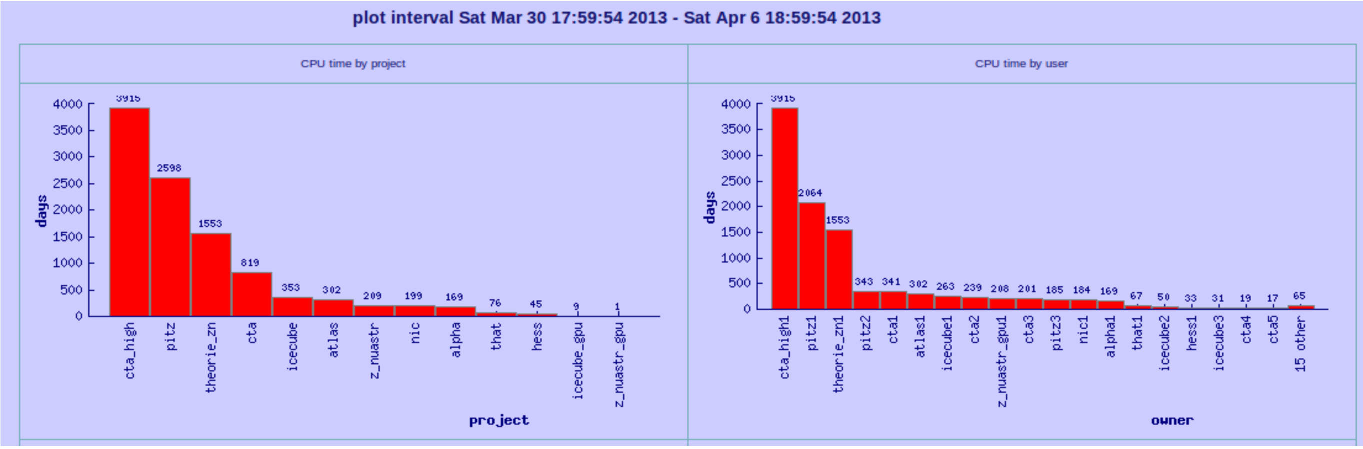
Data sources for monitoring

- Number of machines, cores available, load values and consumables
  - Output of the qhost command
- Summary of running jobs per queue
  - Output of qstat -g c (for quick checks only)
- Detailed information on running and waiting jobs
  - Output of qstat -f -s a -t -ext -xml
  - XML output can be easily parsed, very slow if huge number of jobs waiting
  - Better separate calls for running jobs and waiting/hold … jobs (option -s)
  - Does provide also information on cpu usage, I/O, memory consumption
  - CPU values scaled with cpu_scaling parameter if set, available from qconf -se exechost

Gridengine accounting at DESY (conventional ansatz)

> Typical farm throughput O(10.000.000) jobs/year per farm

- Job results (45 parameters) get written to a (mysql) DB (1 record/job) and deleted from accounting file (faster qstat calls, but few results for ended jobs)
- Results are kept for > 1 year in DB
- Accounting is based on querying this DB, mostly used for summary information
    - Fraction of farm used by a given project during a time period (last day, .. last year etc.)
    - Information is displayed using pie charts (see title page) or bar charts
    - User information is by default anonymous, can be displayed for admins
    - Plots useful for capacity planning and controlling project shares



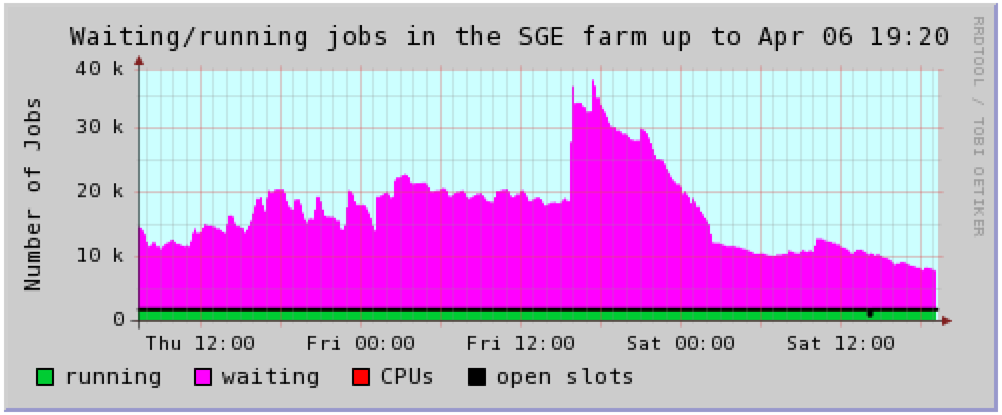
plot interval Sat Mar 30 17:59:54 2013 - Sat Apr 6 18:59:54 2013

Displaying accounting information

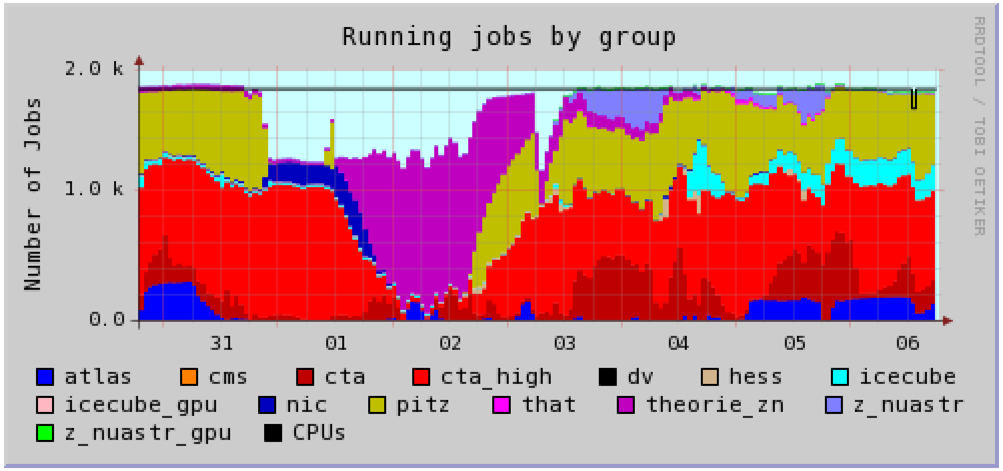
- For users recent farm usage (globally or by project) most useful
  - Number of waiting and running jobs over time to estimate job waiting time
  - Waiting and running jobs by project to see activity periods of projects
  - Ratio of CPU/wallclock time to estimate efficiency of jobs
  - …
- Extraction of these parameters from DB on request nearly impossible
  - Have one table with huge number of entries, queries even with indexing too slow
  - Would need more intelligent storage of job data optimized for fast retrieval
  - We fill the job data in addition into round robin databases (RRD)
    - Is done incrementally
    - Does no longer hold data for individual jobs but data aggregated by user, project, ...
    - Is less precise for historical data, holds data for a time period of one year
    - Plots can be generated very quickly from these rrds

Examples of RRD plots

- Running and waiting jobs

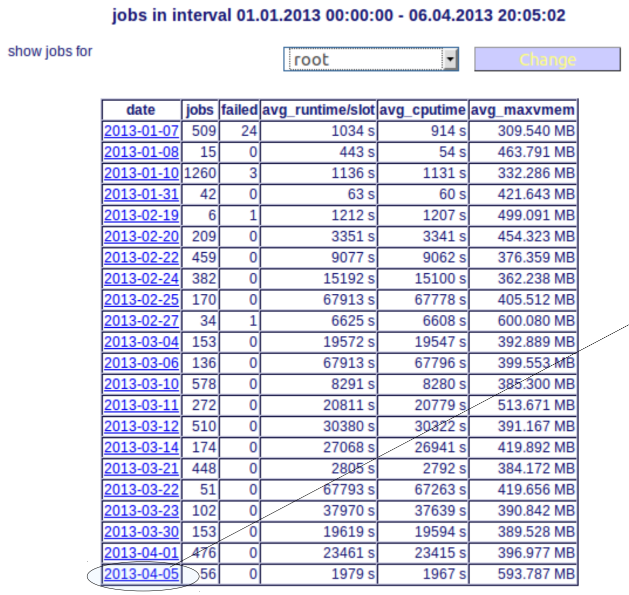


- Running jobs by group



Gridengine monitoring at DESY

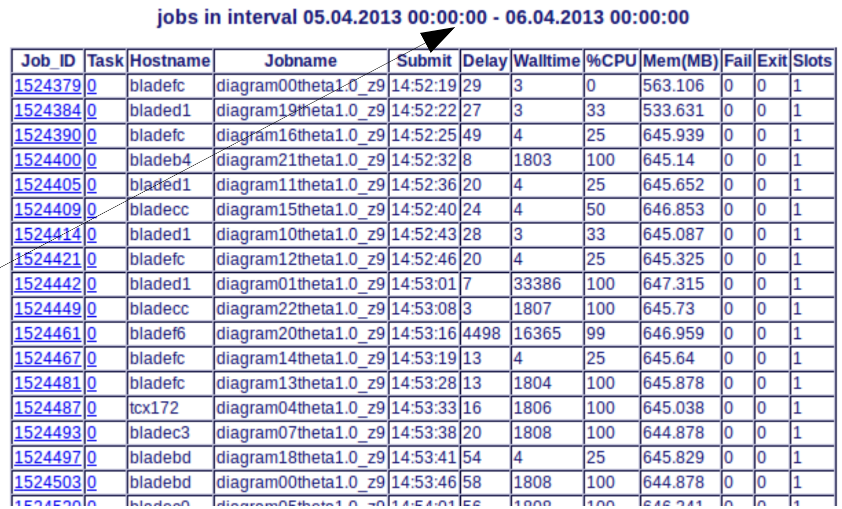
- Current focus on providing job information for users
- Monitoring the batch farm to see the health of the system done differently
- Start with summary information for given user per day
- Continue with listing of all jobs on a given day



jobs in interval 01.01.2013 00:00:00 - 06.04.2013 20:05:02

show jobs for root ▾ Change

date	jobs	failed	avg_runtime/slot	avg_cputime	avg_maxvmem
2013-01-07	509	24	1034 s	914 s	309.540 MB
2013-01-08	15	0	443 s	54 s	463.791 MB
2013-01-10	1260	3	1136 s	1131 s	332.286 MB
2013-01-31	42	0	63 s	60 s	421.643 MB
2013-02-19	6	1	1212 s	1207 s	499.091 MB
2013-02-20	209	0	3351 s	3341 s	454.323 MB
2013-02-22	459	0	9077 s	9062 s	376.359 MB
2013-02-24	382	0	15192 s	15100 s	362.238 MB
2013-02-25	170	0	67913 s	67778 s	405.512 MB
2013-02-27	34	1	6625 s	6608 s	600.080 MB
2013-03-04	153	0	19572 s	19547 s	392.889 MB
2013-03-06	136	0	67913 s	67796 s	399.553 MB
2013-03-10	578	0	8291 s	8280 s	385.300 MB
2013-03-11	272	0	20811 s	20779 s	513.671 MB
2013-03-12	510	0	30380 s	30322 s	391.167 MB
2013-03-14	174	0	27068 s	26941 s	419.892 MB
2013-03-21	448	0	2805 s	2792 s	384.172 MB
2013-03-22	51	0	67793 s	67263 s	419.656 MB
2013-03-23	102	0	37970 s	37639 s	390.842 MB
2013-03-30	153	0	19619 s	19594 s	389.528 MB
2013-04-01	476	0	23461 s	23415 s	396.977 MB
2013-04-05	56	0	1979 s	1967 s	593.787 MB

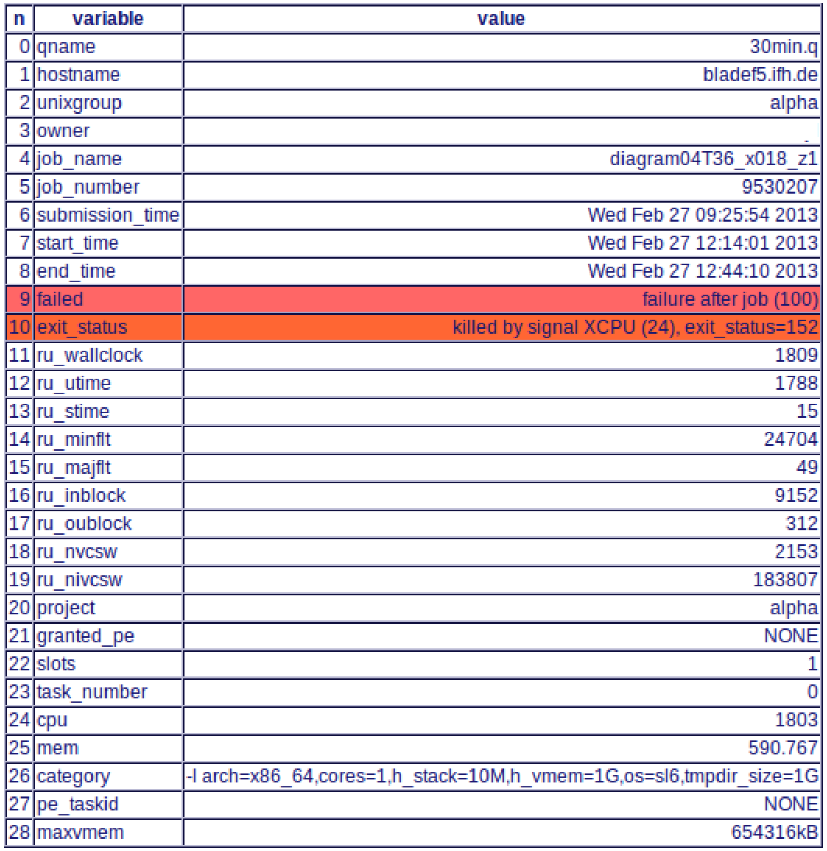


jobs in interval 05.04.2013 00:00:00 - 06.04.2013 00:00:00

Job_ID	Task	Hostname	Jobname	Submit	Delay	Walltime	%CPU	Mem(MB)	Fail	Exit	Slots
1524379	0	bladefc	diagram00theta1.0_z9	14:52:19	29	3	0	563.106	0	0	1
1524384	0	bladed1	diagram19theta1.0_z9	14:52:22	27	3	33	533.631	0	0	1
1524390	0	bladefc	diagram16theta1.0_z9	14:52:25	49	4	25	645.939	0	0	1
1524400	0	bladeb4	diagram21theta1.0_z9	14:52:32	8	1803	100	645.14	0	0	1
1524405	0	bladed1	diagram11theta1.0_z9	14:52:36	20	4	25	645.652	0	0	1
1524409	0	bladecc	diagram15theta1.0_z9	14:52:40	24	4	50	646.853	0	0	1
1524414	0	bladed1	diagram10theta1.0_z9	14:52:43	28	3	33	645.087	0	0	1
1524421	0	bladefc	diagram12theta1.0_z9	14:52:46	20	4	25	645.325	0	0	1
1524442	0	bladed1	diagram01theta1.0_z9	14:53:01	7	33386	100	647.315	0	0	1
1524449	0	bladecc	diagram22theta1.0_z9	14:53:08	3	1807	100	645.73	0	0	1
1524461	0	bladef6	diagram20theta1.0_z9	14:53:16	4498	16365	99	646.959	0	0	1
1524467	0	bladefc	diagram14theta1.0_z9	14:53:19	13	4	25	645.64	0	0	1
1524481	0	bladefc	diagram13theta1.0_z9	14:53:28	13	1804	100	645.878	0	0	1
1524487	0	tcx172	diagram04theta1.0_z9	14:53:33	16	1806	100	645.038	0	0	1
1524493	0	bladec3	diagram07theta1.0_z9	14:53:38	20	1808	100	644.878	0	0	1
1524497	0	bladebd	diagram18theta1.0_z9	14:53:41	54	4	25	645.829	0	0	1
1524503	0	bladebd	diagram00theta1.0_z9	14:53:46	58	1808	100	644.878	0	0	1
1524520	0	bladec0	diagram05theta1.0_z9	14:54:01	56	1808	100	646.341	0	0	1

Monitoring individual jobs

- All parameters of a job get displayed
- Error codes are translated into human readable form
- All tasks of array jobs and parallel jobs can be displayed individually



n	variable	value
0	qname	30min.q
1	hostname	bladef5.ifh.de
2	unixgroup	alpha
3	owner	
4	job_name	diagram04T36_x018_z1
5	job_number	9530207
6	submission_time	Wed Feb 27 09:25:54 2013
7	start_time	Wed Feb 27 12:14:01 2013
8	end_time	Wed Feb 27 12:44:10 2013
9	failed	failure after job (100)
10	exit_status	killed by signal XCPU (24), exit_status=152
11	ru_wallclock	1809
12	ru_utime	1788
13	ru_stime	15
14	ru_minflt	24704
15	ru_majflt	49
16	ru_inblock	9152
17	ru_oublock	312
18	ru_nvcsw	2153
19	ru_nivcsw	183807
20	project	alpha
21	granted_pe	NONE
22	slots	1
23	task_number	0
24	cpu	1803
25	mem	590.767
26	category	-l arch=x86_64,cores=1,h_stack=10M,h_vmem=1G,os=sl6,tmpdir_size=1G
27	pe_taskid	NONE
28	maxvmem	654316kB

Monitoring of still running jobs

- Currently only details for finished jobs are displayed
  - No information on varying resource usage during job execution
- Database change to collect data for running jobs
  - Adding a table holding the job parameters for running (and recently finished) jobs
  - Recording parameters every 5 minutes for all running jobs in a DB table
  - Info gets deleted after a few (3) days to limit size of the table
- New table allows to display status of running jobs
  - In addition history of resource usage since start of job
  - Is done also for recently finished jobs
  - Valuable info for users to better understand how the jobs perform
- Code ready for simple jobs
  - No plots yet, but planned, if job runs for more than 1 hour
  - Not ready yet for array jobs and parallel jobs

Performance considerations

Database design

- All values from accounting entered in DB, several values always constant or of limited use
- Shrinking the DB would result in increased speed
- Job details are usually only important for a limited time, store job summaries instead
- Store even less information for older data (like in RDDs)

Database changes

- New tables storing summary information in 5 minute intervals
- Quantities CPU time, runtime, memory and I/O usage by user, project, host and queue
- Cron job to replace fine grained summaries by more coarse grained ones for older data

Generation of plots

- Plots need to be regenerated in regular intervals to show the images instantly
- Not required for RRDs, generation of RRD graphs is fast

Privacy

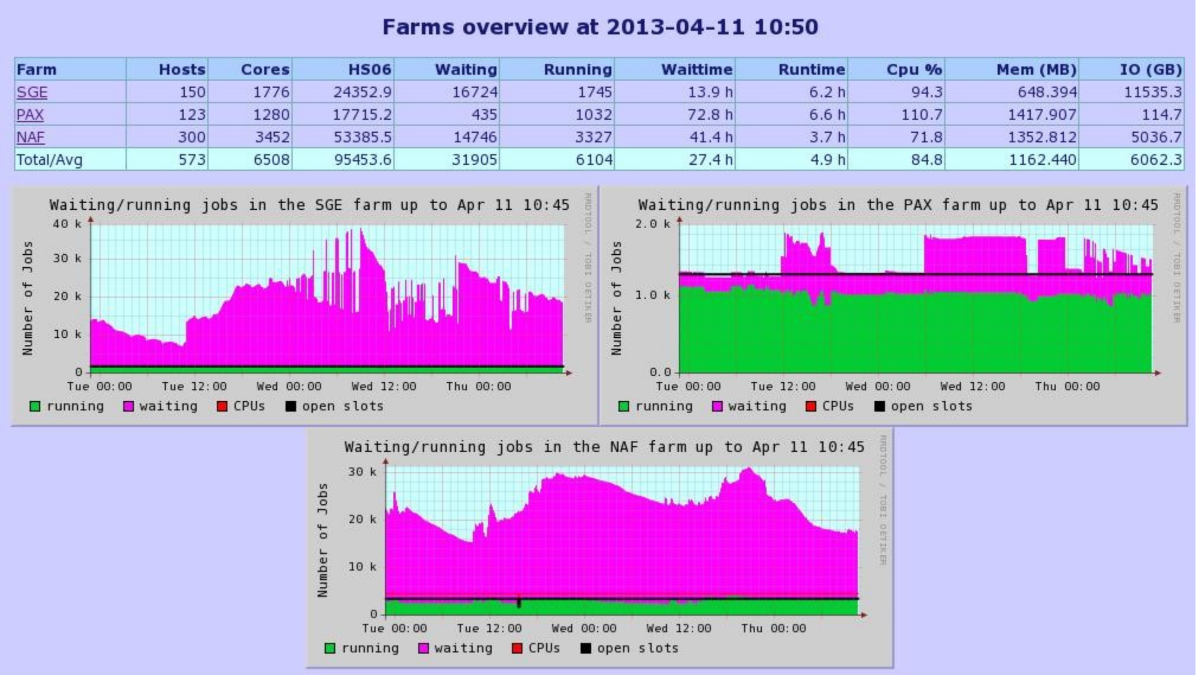
- Batch system information (qstat, qhost,...) accessible without restriction
- Accumulated data can show trends, user behavior etc.
- Decision to let users view only their own jobs
- Decision to hide user names for top n user plots
- Definition of group and global administrators who can see all data
  - Jobs of users within same group
  - User names within plots

Integration of other batch systems

- Torque does provide reporting files similar to GridEngine
  - Torque is used in our Grid cluster
  - We process the reporting file and put the data in tables (almost) identical to GE tables
  - The same scripts are then used to fill RRDs and display plots
- We do not have a mechanism in place yet to monitor current jobs
  - Would be in analogy to calling qstat (collecting data from running jobs)
  - Data would again go into tables compatible with GE tables
  - Display of running Torque jobs would then be covered by existing scripts
- Integration of systems other than Torque would require similar steps
  - Create and fill tables for ended and running jobs
  - Need to make sure same quantities in same units get recorded

Improvements

- Improved monitoring script is in testing phase
  - Added a top page summarizing the status of the farms



Farms overview at 2013-04-11 10:50

Farm	Hosts	Cores	HS06	Waiting	Running	Waittime	Runtime	Cpu %	Mem (MB)	IO (GB)
SGE	150	1776	24352.9	16724	1745	13.9 h	6.2 h	94.3	648.394	11535.3
PAX	123	1280	17715.2	435	1032	72.8 h	6.6 h	110.7	1417.907	114.7
NAF	300	3452	53385.5	14746	3327	41.4 h	3.7 h	71.8	1352.812	5036.7
Total/Avg	573	6508	95453.6	31905	6104	27.4 h	4.9 h	84.8	1162.440	6062.3

  - Includes monitoring of running jobs
  - Allows to fix the color mapping for projects appearing in plots
  - Still issues with speed of DB queries and display of running array or parallel jobs
  - No integration of Torque (GRID farm) yet

Next steps

- Check consistency between data from qstat and accounting records
- Backfill tables from accounting data, if qstat info is missing
- Optimize DB design to speed up queries
- Make the monitoring more robust and production ready

- Look into new paradigms to store and display data

- Try to look for partners who are willing to contribute

Summary

- An accounting and monitoring system for GE is in place at DESY
- Several shortcomings and bottlenecks exist in the present solution
  - ARCO and Unisight cannot cover all of our use cases
  - A new concept using Splunk is being tried (see next talk)
- Improvements to the current traditional solution are planned
- Collaborators welcome