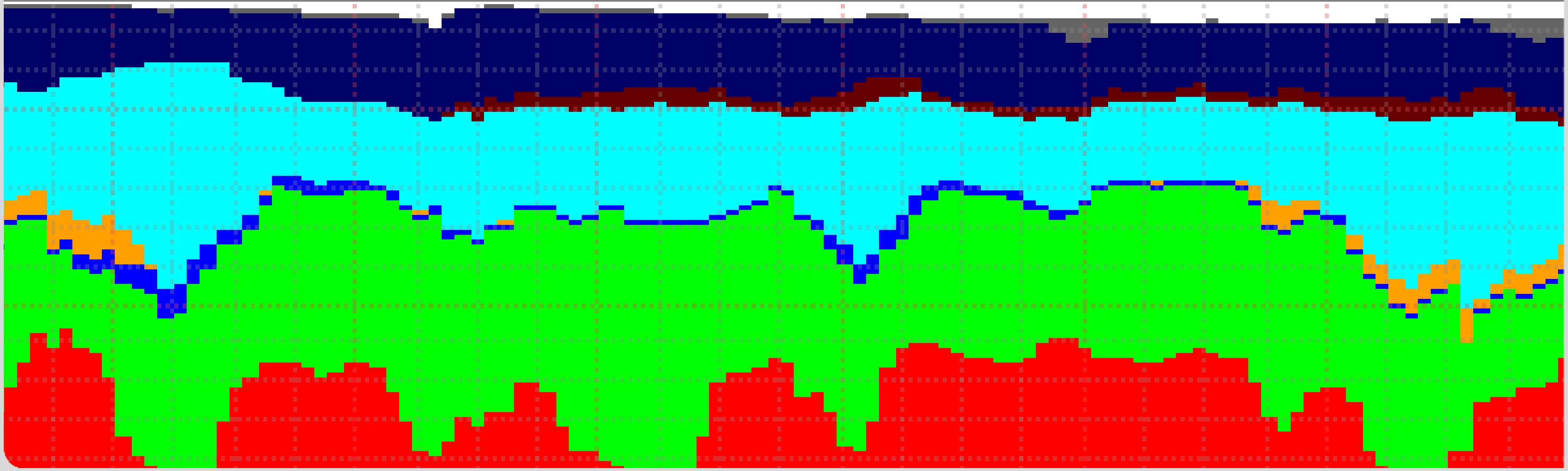
Experiences with Grid Engine (UGE) at GridKa

pre-GDB on Batch Systems, Bologna, 2014-03-11

Manfred Alef

STEINBUCH CENTRE FOR COMPUTING – SCC



LRMS at GridKa

- Multi-VO support
  - Alice, Atlas, BaBar, Belle II, CDF, CMS, Compass, LHCb, ...
- Very high cluster utilization – most often queued jobs waiting
- Dimensions of the compute farm

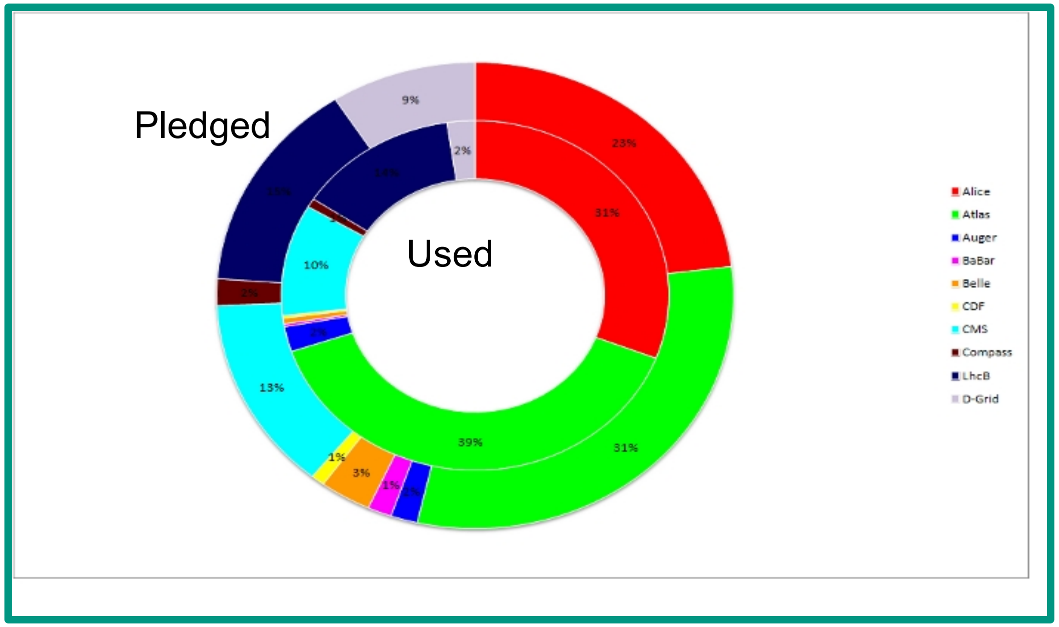
WNs:	613
Job slots:	12,828
Number of jobs (2013):	20 M
➔ Alice:	3.4 M
➔ Atlas:	9.2 M
➔ CMS:	1.9 M
➔ LHCb:	3.2 M
Average job runtime:	5.6 h

LRMS at GridKa

- **Grid Engine (Univa) since mid 2012**
  (has replaced PBS-Professional)
- Some details about configuration:
  - Qmaster using flat files, not Berkeley DB
  - Certificate Security Protocol (CSP) enabled
    - ➔ Some issues with an undocumented CRL file expiring after 30 days by default, causing unscheduled draining of the cluster :-(
  - Single queue supports all VOs as well as single- and multi-core jobs
    - ➔ 120h maximum walltime
    - ➔ In the past (PBS era): short / medium / long / extralong queues
      - Users had submitted either solely to extralong queue,
        or stupidly round-robin like (s, m, l, xl, s, m, l, xl, s, ...)
  - Fair-share and accounting are configured based on reserved usage (aka walltime)
    - ➔ Requires UGE patch do display CPU and wallclock usage in qstat output
    - ➔ Details about fair-share policies see next slide

LRMS at GridKa

- Fair-share scheduling (based on pledged HS06 scores)
  - Share-tree policy (using history, halftime=1000h) – 50% weight
  - Functional ticket policy (not using history) – 50% weight
  - Override tickets (high-priority jobs: OPS, SGM)

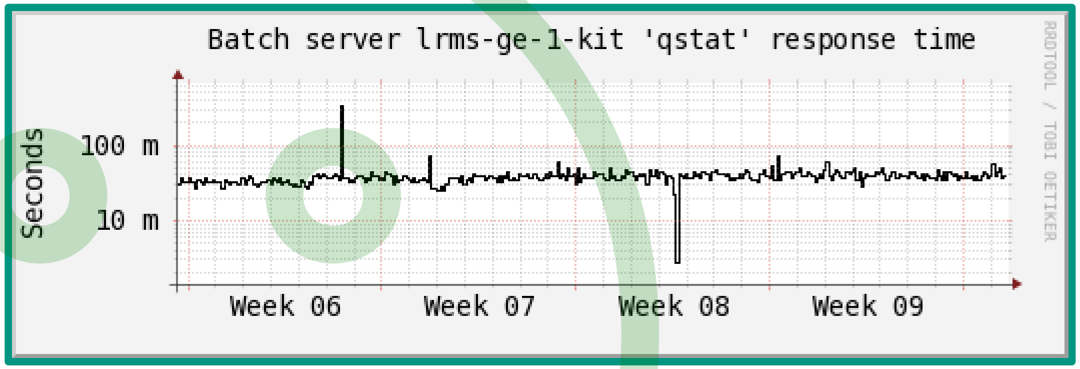


LRMS at GridKa

- **Grid frontends**
  - CREAM

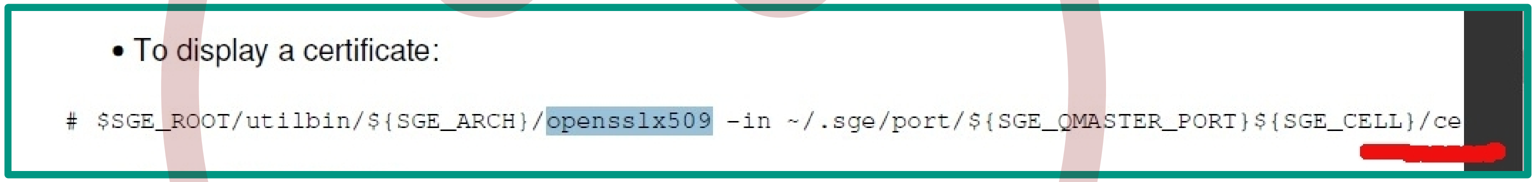
LRMS at GridKa

- **Experiences with UGE**
  - Scheduler:
    - ➔ Very fast
    - ➔ Very stable
  - Very quick support desk



LRMS at GridKa

- **Experiences with UGE**
  - Documentations, admin guide, manpages, logfiles, error messages, ...
    - ➔ Very hard to read, a lot of developers slang
      - Several tickets have been filed so far
        'Please translate that error message into plain English'
    - ➔ Admin guide compiled from internal WIKI, some important command lines cut at the right edge of the printable area



• To display a certificate:

```
# $SGE_ROOT/utilbin/${SGE_ARCH}/openss1x509 -in ~/.sge/port/${SGE_QMASTER_PORT}${SGE_CELL}/ce
```

  - Annoying command line syntax
    - ➔ Example:
      'qstat -j' prints 'various information'
      either for all pending jobs or for some job id's
      - Meaning of 'various' is not specified ...

Multi-Core Job Support

- **Multi-core usage**
  - Atlas: continously since several weeks
  - CMS: only a rew test jobs
  - No interest so far from other VOs

Multi-Core Job Support: LRMS Configurations

- No separate queue
  - Jobs request number of slots in JDL
    - ➔ Parallel environment (PE) has been configured to support multi-core jobs
    - ➔ Any number of slots supported
      (should be less than or equal to maximal number of slots per host :-)
    - ➔ Memory limits set by profile script (according to number of cores), not by UGE
- Dynamic scheduling
  - No sub-clusters (neither VO nor multi-core specific)
- How does a multi-core job start?
  - By default (without draining) it most probably won't because all slots are continuously occupied by single-core jobs

Multi-Core Job Support: LRMS Configurations

- Resource reservations
  - Max_reservation set to ~10...20
    - ➔ Per reservation: up to 7 slots idling to boost 8-core jobs
      - (Example: about 1.0 % of total capacity when max_reservation=20)
    - ➔ Very few jobs which declare their estimated runtime (wallclock usare) pending
    - ➔ Max_reservation setting controls the number of multi-core jobs with scheduled reservations; <u>no limit on the number of running multi-core jobs</u>
- Cron job to provide extra flags to queued multi-core jobs:
  - **`qalter -R y`** `$list_of_pending_multicore_jobs`
    - ➔ '-R y': Add mandatory flag to enable job reservations
      - Should be implemented in /usr/libexec/sge_submit.sh script running on CREAM's

Multi-Core Job Support: LRMS Configurations

- How reservations are handled by the scheduler (snippet of sched_conf manpage):

max_reservation

The maximum number of reservations scheduled within a schedule interval. When a runnable job can not be started due to a shortage of resources a reservation can be scheduled instead. A reservation can cover consumable resources with the global host, any execution host and any queue. For parallel jobs reservations are done also for slots resource as specified in sge_pe(5). As job runtime the maximum of the time specified with -l h_rt=... or -l s_rt=... is assumed. For jobs that have neither of them the default_duration is assumed. Reservations prevent jobs of lower priority as specified in sge_priority(5) from utilizing the reserved resource quota during the time of reservation. Jobs of lower priority are allowed to utilize those reserved resources only if their prospective job end is before the start of the reservation (backfilling). Reservation is done only for non-immediate jobs (-now no) that request reservation (-R y). If max_reservation is set to "0" no job reservation is done.

Note, that reservation scheduling can be performance consuming and hence reservation scheduling is switched off by default. Since reservation scheduling performance consumption to is known grow with the number of pending jobs, the use of -R y option is recommended only for those jobs actually queuing for bottleneck resources. Together with the max_reservation parameter this technique can be used to narrow down performance impacts.

Multi-Core Job Scheduling

- How reservations are handled by the scheduler
  - Limitations:
    - ➔ Configured number of max_reservation
    - ➔ Job priorities – looking at GridKa monitoring dashboard (20 max_reservations):

Project	Queued jobs (or JAT's)	Running jobs	Allocated slots		Nominal share		Normalized tickets	Average CPU efficiency
Alice	41	2939	2939	24 %	24 %	30000 HS06	0.00014	72 %
Atlas-Pilot	24	287	287	2 %	6 %	7975 HS06	0.00036	96 %
Atlas-Prod	212	5077	5322	44 %	25 %	31900 HS06	0.00009	95 %
Auger	0	87	87	1 %	2 %	2182 HS06	0.00036	94 %
Belle	6	53	53	0 %	3 %	4200 HS06	0.00104	80 %
CMS-MCP	2461	2660	2660	22 %	13 %	16625 HS06	0.00009	86 %
CMS-Pilot	2031	147	147	1 %	1 %	875 HS06	0.00008	82 %
D-Grid-2007-HEP	0	29	29	0 %	4 %	5487 HS06	0.00269	93 %
LHCb	2	636	636	5 %	15 %	19200 HS06	0.00042	91 %
SGM	2	2	2	0 %	0 %	0 HS06	1.00000	8 %
Total	4779	11917	12162	100 %	–	–	–	ø 84 %

Multi-Core Job Scheduling

Project	Queued jobs (or JAT's)	Running jobs	Allocated slots		Nominal share		Normalized tickets	Average CPU efficiency
Alice	41	2939	2939	24 %	24 %	30000 HS06	0.00014	72 %
Atlas-Pilot	24	287	287	2 %	6 %	7975 HS06	0.00036	96 %
Atlas-Prod	212	5077	5322	44 %	25 %	31900 HS06	0.00009	95 %
Auger	0	87	87	1 %	2 %	2182 HS06	0.00036	94 %
Belle	6	53	53	0 %	3 %	4200 HS06	0.00104	80 %
CMS-MCP	2461	2660	2660	22 %	13 %	16625 HS06	0.00009	86 %
CMS-Pilot	2031	147	147	1 %	1 %	875 HS06	0.00008	82 %
D-Grid-2007-HEP	0	29	29	0 %	4 %	5487 HS06	0.00269	93 %
LHCb	2	636	636	5 %	15 %	19200 HS06	0.00042	91 %
SGM	2	2	2	0 %	0 %	0 HS06	1.00000	8 %

Cluster utilization by Atlas (role=production) in the past above the pledged share ...

... may result in lower priority than other VOs

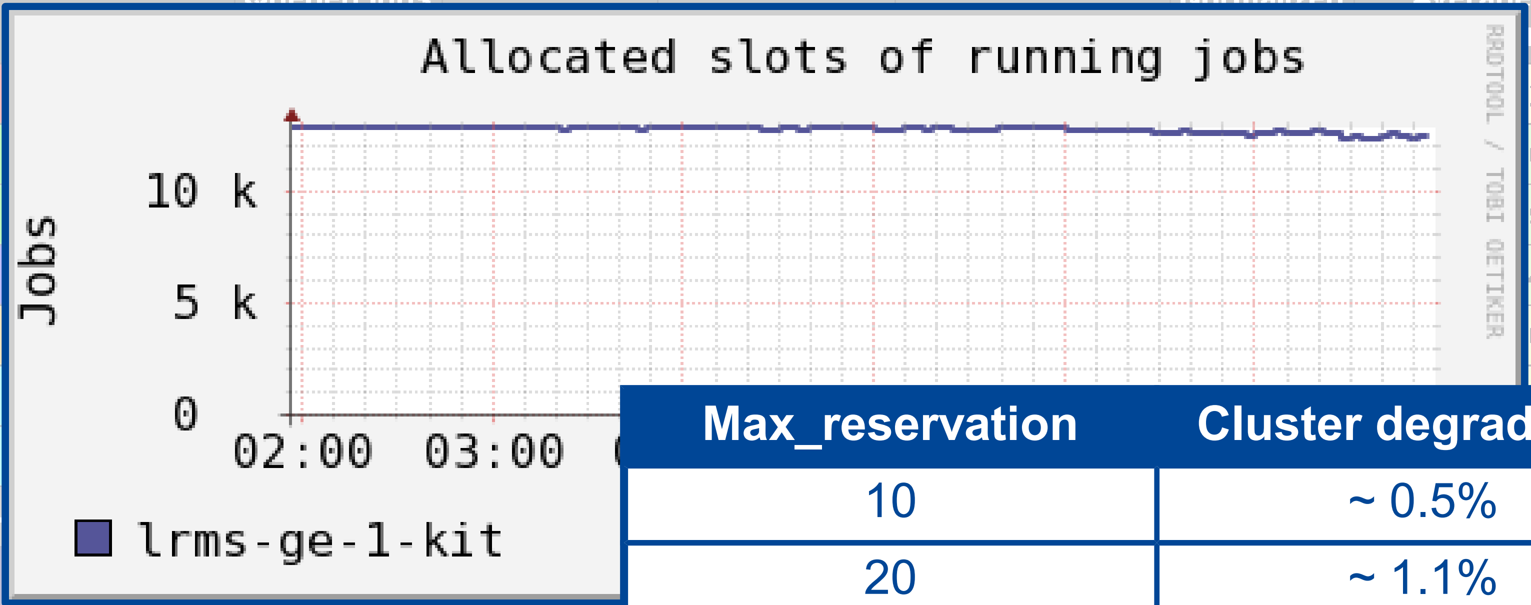
Nevertheless Atlas multi-core production jobs were starting because of low number of pending jobs from other VOs at that time!

Multi-Core Job Scheduling

Project	Queued jobs (or JAT's)	Running jobs	Allocated slots		Nominal share		Normalized tickets	Average CPU efficiency
Alice	41	2939	2939	24 %	24 %	30000 HS06	0.00014	72 %
Atlas-Pilot	24	287	287	2 %	6 %	7975 HS06	0.00036	96 %
Atlas-Prod	212	5077	5322	44 %	25 %	31900 HS06	0.00009	95 %
Auger	0	87	87	1 %	2 %	2182 HS06	0.00036	94 %
Belle	6	53	53	0 %	3 %	4200 HS06	0.00104	80 %
CMS-MCP	2461	2660	2660	22 %	13 %	16625 HS06	0.00009	86 %
CMS-Pilot	2031	147	147	1 %	1 %	875 HS06	0.00008	82 %
D-Grid-2007-HEP	0	29	29	0 %	4 %	5487 HS06	0.00269	93 %
LHCb	2	636	636	5 %	15 %	19200 HS06	0.00042	91 %
SGM	2	2	2	0 %	0 %	0 HS06	1.00000	8 %
Total	4779	11917	12162	100 %	–	–	–	ø 84 %

Decreased cluster utilization when reservations become scheduled, insufficient entropy to backfill gaps (maximal available slots: 12804)
Scheduler configuration: max_reservation=20

Multi-Core Job Scheduling



Max_reservation	Cluster degradation
10	~ 0.5%
20	~ 1.1%

	Queued jobs							Normalized	Average ...ncy
SGM	2	2	2	0 %	0 %		0 HS06	1.00000	8 %
Total	4779	11917	12162	100 %	–		–	–	ø 84 %

Decreased cluster utilization when reservations become scheduled, insufficient entropy to backfill gaps (maximal available slots: 12804)
Scheduler configuration: max_reservation=20

Multi-Core Job Scheduling

- How reservations are handled by the scheduler
  - Backfilling:
    - ➔ Jobs of lower priority are allowed to utilize the reserved resources only if their prospective job end (i.e. the declared wallclock usage) is before the start of the reservation
    - ➔ Job execution time of batch queue: 0...120 h
      Average job runtime: ~5.6 h
    - ➔ Very few jobs which declare their estimated runtime, therefore almost no backfilling in effect

Multi-Core Job Scheduling

- How reservations are handled by the scheduler
  - Ramp-up by 2...5 jobs per hour
    - ➔ If respective VO share at top
    - ➔ If no single-core jobs with higher priority are waiting (FIFO)
  - Again and again – multi-core job queue idling
    - ➔ Wavelike submission of multi-core jobs detected
    - ➔ Slots of ending multi-core job become occupied by single-core jobs when no more multi-core jobs are waiting
    - ➔ Reservations, and degradation of cluster utilization, become permanent

Accounting

- Fairshare configurations and accounting at GridKa are based on reserved (aka wallclock) usage
  - 2 separate configuration items – 'qconf -mconf':

```
execd_params      SHARETREE_RESERVED_USAGE=true \
                  ACCT_RESERVED_USAGE=true
```

- The qacct command doesn't sum up the walltime of multi-core jobs!

```
# qacct -slots -f /tmp/accounting-02-2014
SLOTS     WALLCLOCK           UTIME         STIME           CPU
========================================================================
    0             0           0.000         0.000           0.000
    1   30045925533 19190341698.755 322568683.359 26836528747.830
    2         14574           4.552         1.674       29163.498
    3        113341           3.781         3.935      184506.117
    4        183696          16.268        16.760      735472.000
    8      83796513   513975203.321   1177782.105   605994381.126
#
```

- Multi-core jobs are handled correctly by APEL

Conclusions

- UGE in operation at GridKa since 2012
- Very stable, very fast operation; quick support desk
- User interface (documentations, error messages, ...) less satisfying
- Multi-core job support:
  - Knob to boost multi-core jobs: max_reservations
  - Reservations cause degradation in cluster utilization;
    the job mix arriving at our site prevents effective backfilling
    - ➔ Average job execution time (2013): 5.6 h
    - ➔ No declarations of the estimated job runtime
  - Degratation of utilization depending on max_reservation configurations
    - ➔ Up to around 0.5 % idling per 10 reservations
  - Wavelike submission of multi-core job detected
    - ➔ Slots of ending multi-core job become occupied by single-core jobs when no more multi-core jobs are waiting
    - ➔ Causing new reservations when multi-core jobs are submitted again