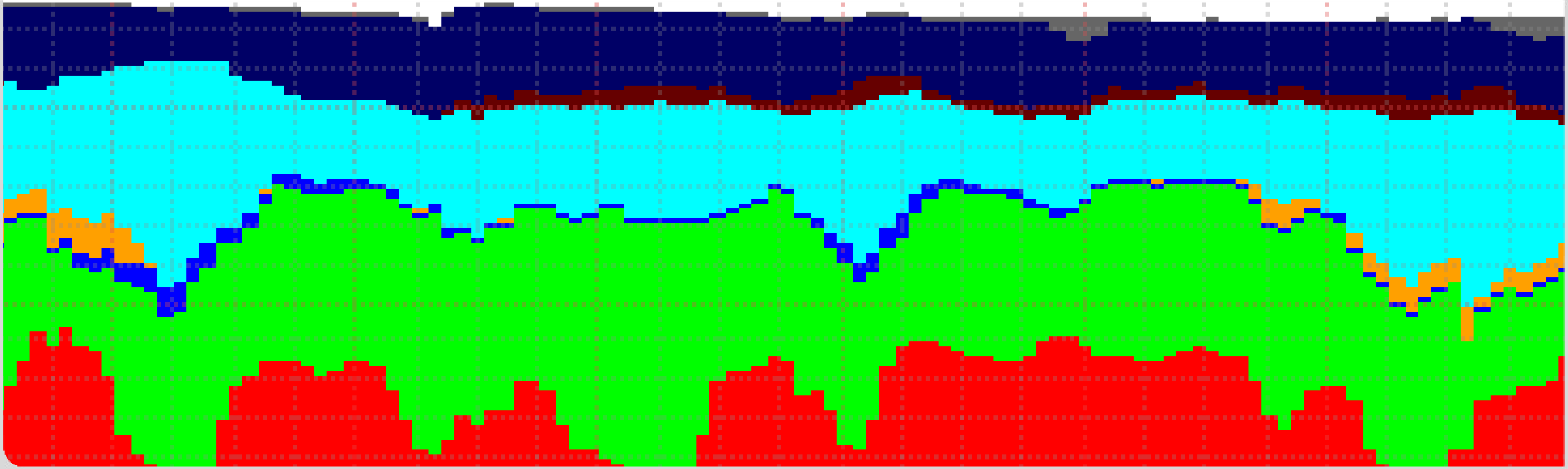
Grid Engine at GridKa

HEPiX Spring 2015, Oxford

Manfred Alef

STEINBUCH CENTRE FOR COMPUTING – SCC



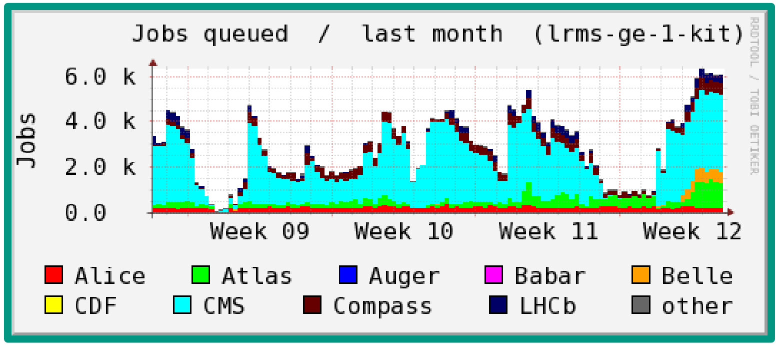
Compute Farm at GridKa

Compute Farm at GridKa

- **Dimensions of the compute farm**
  - HS06: 145,805
  - WNs: 611
  - Job slots: 12,357
  - Number of jobs (2014): 23 M Multi-core: 760 K
  - Average job runtime: 4:40 h

- **Multi-VO support**
  - Alice (23% share), Atlas (33%), Auger, BaBar, Belle II, CMS (13%), Compass, LHCb (15%), ...

- **Very high cluster utilization**
  - Most often waiting jobs in the queue



2 ½ Years of Grid Engine at GridKa

2 ½ Years of Grid Engine at GridKa

- **Grid Engine (Univa) since mid 2012 (until 2017)**
  - Has replaced PBS Professional

2 ½ Years of Grid Engine at GridKa

- **Experiences (1)**
  - Very stable
    - ➔ No crashes (1 unscheduled downtime at the beginning, see next slide)
    - ➔ No hanging daemons
    - ➔ No black hole issues
    - ➔ ...
  - Very fast scheduler
  - Flexible fair-share policies
  - Very quick support team

2 ½ Years of Grid Engine at GridKa

- **Experiences (2)**
  - Documentations, admin guide, manpages, logfiles, error messages, … often very hard to read, a lot of developer's slang
    - ➔ Several tickets have been filed so far like 'Please translate that error message to plain English'
  - Unscheduled draining when an undocumented CRL file had expired
    - ➔ See talk about the Certificate Security Protocol (CSP) by Andreas Haupt https://indico.cern.ch/event/199025/session/5/contribution/18/material/slides/0.pdf
    - ➔ Diagnostic commands from Admin Guide not helpfull because they were cut at the right edge of the printable area (fixed in the newest release 8.2)

2 ½ Years of Grid Engine at GridKa

- **Experiences (3)**
  - Bizarre formatted XML output

```
       <UA_name>cpu</UA_name>
       <UA_value>103926.000000</UA_value>
```

  - No true CPU normalization
    - → Grid Engine supports configuration of `usage_scaling` factor per host ✓
    - → CPU and walltime usage of running jobs are displayed correct ✓
    - → Job accounting doesn’t take care of the normalization factors ✗
    - → No adjustments of runtime limits corresponding to slot performance ✗

Multi-Core Job Support

Multi-Core Job Support

- **Multi-core job usage**
  - Atlas and CMS: continously submitting multi-core jobs since around 1 year
  - No interest so far from other VOs

Multi-Core Job Support

- **Our goal**
  - Enable multi-core job support with minimal wasting of resources

- **No separate queue**
  - Jobs request number of slots by adding `-pe` switch
    - ➔ Parallel environment (PE) has been configured to support multi-core jobs
    - ➔ Any number of slots supported
      (less than or equal to maximal number of slots per host)

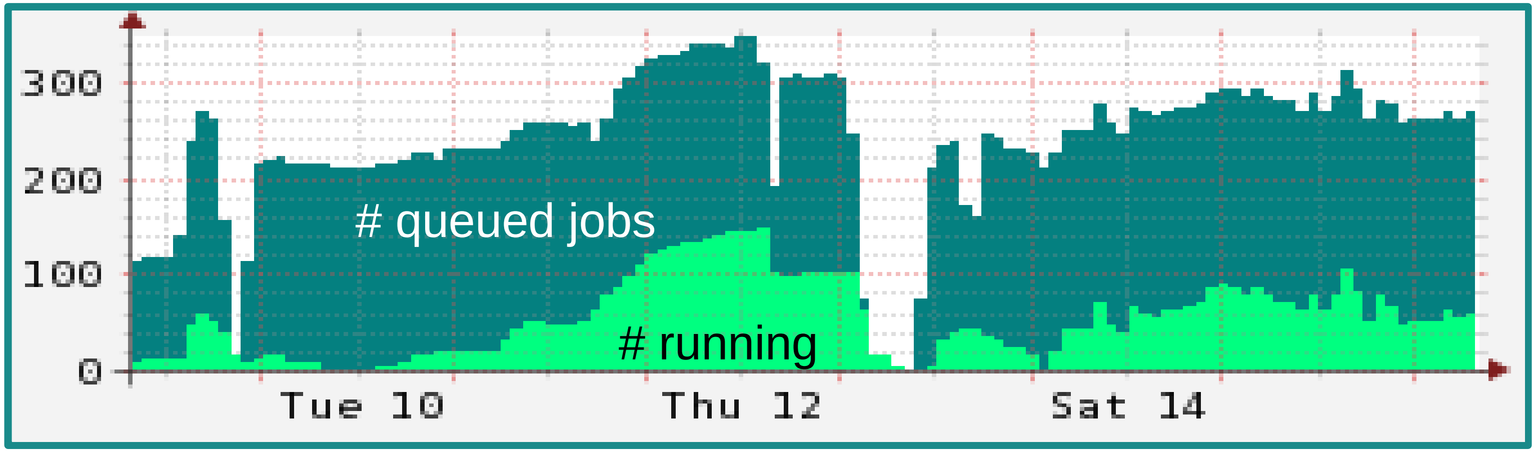
- **Dynamic scheduling**
  - No sub-clusters
    - ➔ Neither VO nor multi-core specific
    - ➔ Neither static nor dynamic partitioning
  - Multi-core jobs can share WN with single-core jobs
    - ➔ Not necessary to drain a whole WN just to boost a single multi-core job

Multi-Core Job Support

- **How prompt does a multi-core job start?**
  - By default (without draining) it most probably won't start within a reasonable timeframe because all slots are continuously occupied by single-core jobs
- **Resource reservations**
  - Max_reservation set to ~10
    - ➔ Per reservation: up to 7 slots idling to boost a pending 8-core job
    - ➔ Max_reservation setting controls the number of pending multi-core jobs with scheduled reservations; <u>no limit on the number of running multi-core jobs</u>
- **Multi-core jobs must use extra option:**
  - Extra submit option:
    - ➔ `qsub  -R y`
  - Alternative (e.g. per cron job):
    - ➔ `qalter  -R y  $list_of_pending_multicore_jobs`

Multi-Core Job Support

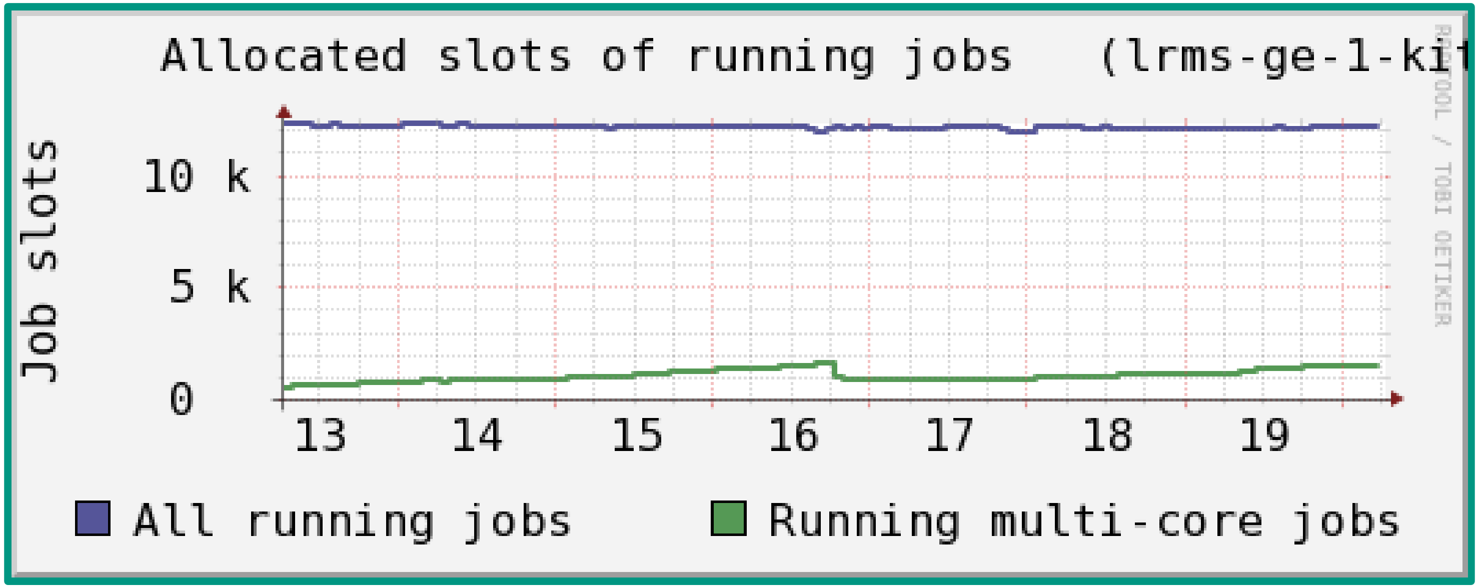
- **Resulting degraded cluster utilization**
  while reservations are scheduled
  - Again and again – no waiting multi-core jobs in the queue
    - ➔ Wavelike submission of multi-core jobs detected
    - ➔ Slots of ending multi-core job become occupied by single-core jobs when no more multi-core jobs are waiting
    - ➔ Frequent reservations, and resulting degradation of cluster utilization



  - Almost no submitted jobs are declaring the expected runtime (and CPU normalization doesn't work as expected) ⇒ no efficient backfilling

Multi-Core Job Support

- **Resulting degraded cluster utilization**
  when reservations become scheduled (8-core-jobs)



Max_reservation	Resulting cluster degradation
10	~ 0.5...1 %
20	~ 1...1.5 %

Linux Control Groups

Linux Control Groups

- **Observations from job monitoring**
  - Huge memory (RSS) consumption of some jobs
  - Orphaned processes (PPID=1)
  - Parallel processes in single-core jobs

Linux Control Groups

- **Testing Cgroups (1)**
  - Configurations according to talk by Daniel Gruber https://indico.cern.ch/event/274555/session/14/contribution/12/material/slides/0.pdf
  - 4 WNs (32 slots)
    - Extended by 10 WNs, including 3 very fat nodes, for bug hunting
  - Job encapsulation:
    - Memory isolation
      - Extra submit option:

        ```
        qsub     -l m_mem_free=$rss_limit
        ```
    - Cpuset
      - Extra submit option:

        ```
        qsub     -mbind nlocal
        ```

Linux Control Groups

- **Testing Cgroups (2)**
  - Job encapsulation works as expected
    - No orphaned processes detected by fabric monitoring
    - Parallel applications cannot get more CPU slots than requested
    - However, memory limitations not fully tested so far
      - Observations at DESY (see site report by Peter ven der Reest): Bugs of features?

Linux Control Groups

- **Testing Cgroups (3)**
  - Crashes (possibly caused by bug of current kernel release)
    - → When many jobs running on a node end at the same time?
    - → Tests at GridKa
      - 20 jobs killed at the same time
        (`pkill -9 -u alef`, `qdel -f -u alef`):
        - Okay
      - WN re-configured with huge number of slots,
        ~ 100 jobs being killed at once
        - WN crashed immediately
    - → Issue may become a showstopper :-(
    - → Not observed in grid production mode so far
      (including some fat nodes with 48 or 64 slots)

Linux Control Groups

- **Testing Cgroups (4)**
  - Issues with Cgroups have also been reported by other sites
    - For instance, see DESY site report https://indico.cern.ch/event/346931/session/1/contribution/75/material/slides/0.pdf
    - Can sites share information?

Conclusions

- 2 1/2 years of Grid Engine at GridKa
- Fully satisfying
  - Very stable operation
  - Few week points, but no showstopper
- Flexible multi-core job support
- Some open questions about configuration and enabling of Cgroups

Questions, Comments

?