


UC San Diego
Local Impact, National Influence, Global Reach

Adaptive 2011

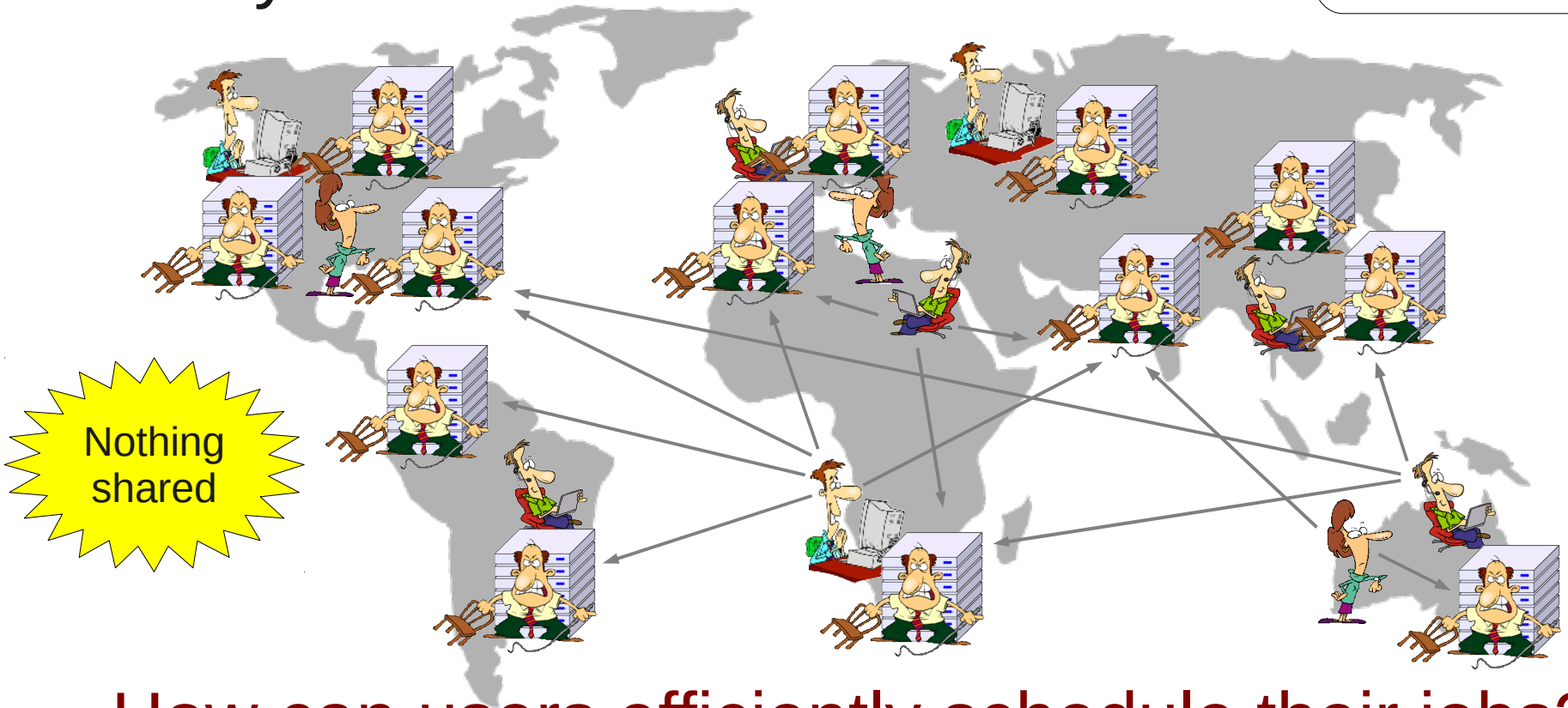
Adapting to the Unknown With a few Simple Rules: The glideinWMS Experience

by Igor Sfiligoi[1],
Benjamin Hass[1], Frank Würthwein[1], and Burt Holzman[2]
[1]UCSD [2]FNAL

The Grid landscape

- Many highly autonomous Grid sites
- Many diverse user communities

Within Scientific Grid environments (e.g. OSG, EGI)



- How can users efficiently schedule their jobs?

Scheduling problem

- Grid sites expose only partial information
  - Access to finer details restricted to site admins
- Each user community wants independence
  - **No centralized, Grid-wide job scheduling**
- As a result
  - **Cannot accurately predict even the near future**
  - Partitioning across sites mostly a guesswork
- Adapting to the ever-changing state a must


Traditional approaches

- Force sites to expose as much info as possible
  - Sites end up publishing lots of garbage
- Implement retries
  - Long tail before ALL jobs in a workflow finish
- Start at many sites concurrently, then kill some
  - Wasteful and with semantic problems

- **Mediocre results and complex code**

The glideinWMS

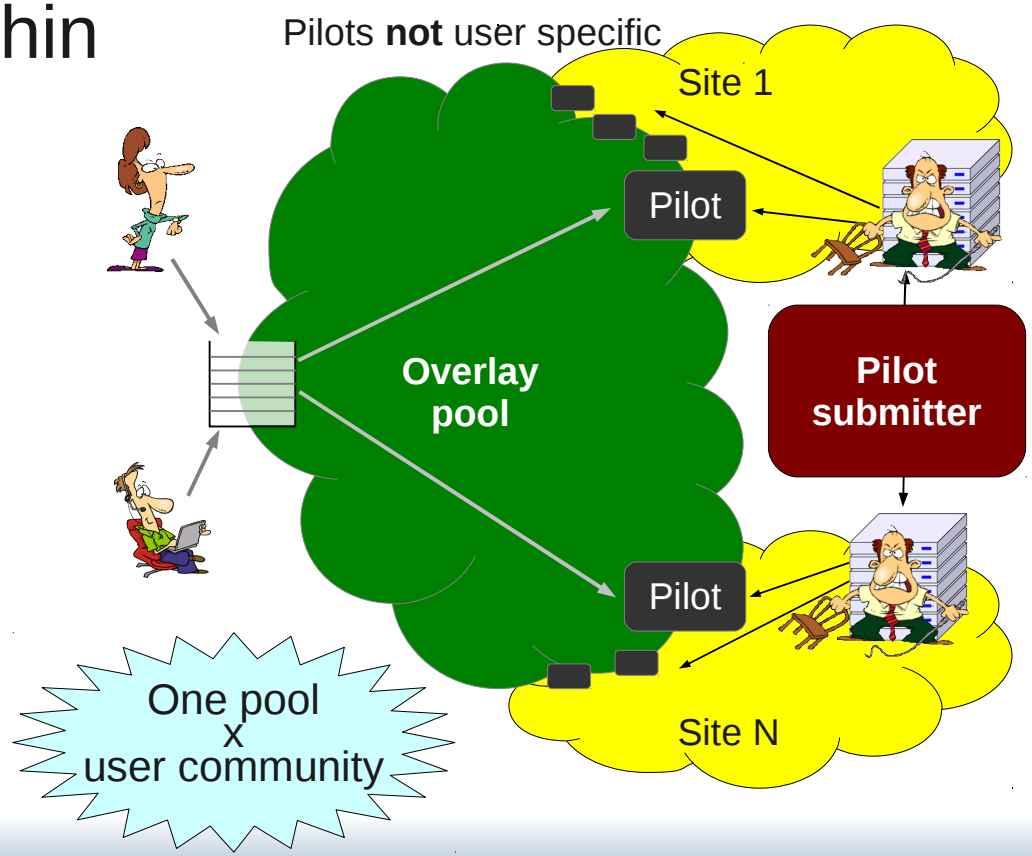
- The glideinWMS approach to the problem
  - **Use the pilot paradigm**
  - **Pressure based scheduling**
  - **Avoid using external information**
  - **Range reduction**

The glideinWMS is a Grid job scheduler initially developed at FNAL by the CMS experiment

- Based on the CDF glideCAF concept
- With contributions from several other institutes
- Widely used in OSG, with a large instance at UCSD

The pilot paradigm

- Send pilots to Grid sites (never user jobs)
  - Create a dynamic overlay pool of compute resources
  - Jobs scheduled within this overlay pool
- Scheduling in the overlay pool easy
  - Complete info
  - Full control
- Problem moved to the pilot submitter



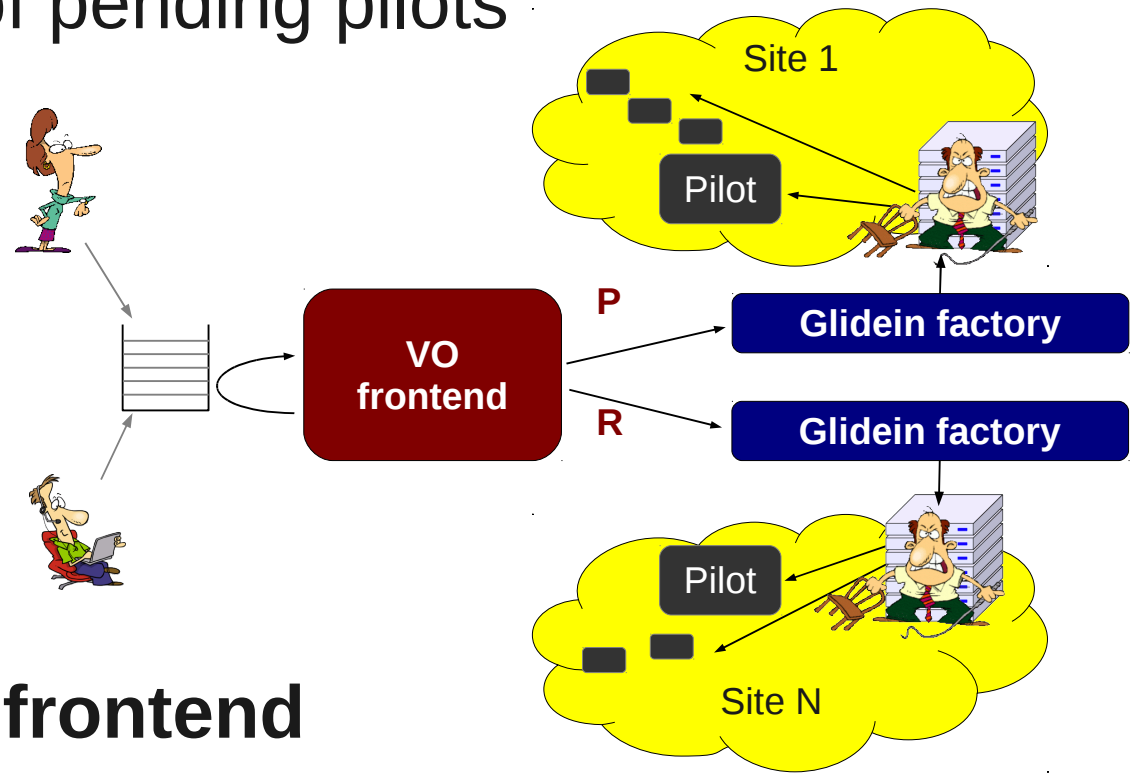
Is pilot scheduling easier?

- User jobs
  - Every job is important => users wait for last to finish
  - A failed job is a problem for the user
  - Many users => priority handling
- Must handle each and every one

- Pilot jobs
  - All the same
  - A failed pilot job is just wasted CPU time
  - Single credential => no need to prioritize between them
- Only number of pilot jobs counts

Pressure based scheduling

- The glideinWMS pilot scheduling based on the concept of **pilot pressure**
  - Keep a fixed no. of pending pilots in remote queues
  - Site by site
- Furthermore, split pilot scheduling from pilot submission
  - Scheduling in **VO frontend**



Determining the pressure

- Calculating the proper pressure important
  - Too low => small overlay pool => long job wait
  - Too high => on jobs when pilot starts => wasted CPU
- Must be recalculated often
- Each site has its own pressure
- Input to pressure calculation
  - **Only no. matching pending**(i.e. idle) **user jobs**
  - Grid status incomplete and unreliable
- Some jobs that can run on multiple Grid sites
  - Count them as the appropriate fraction against each

$$P_s(t)=f(I_s(t))$$

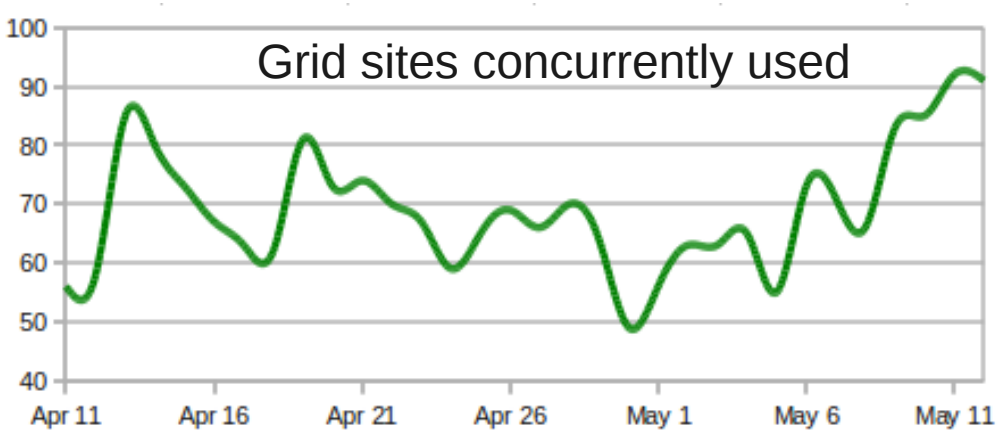
Simple pressure function

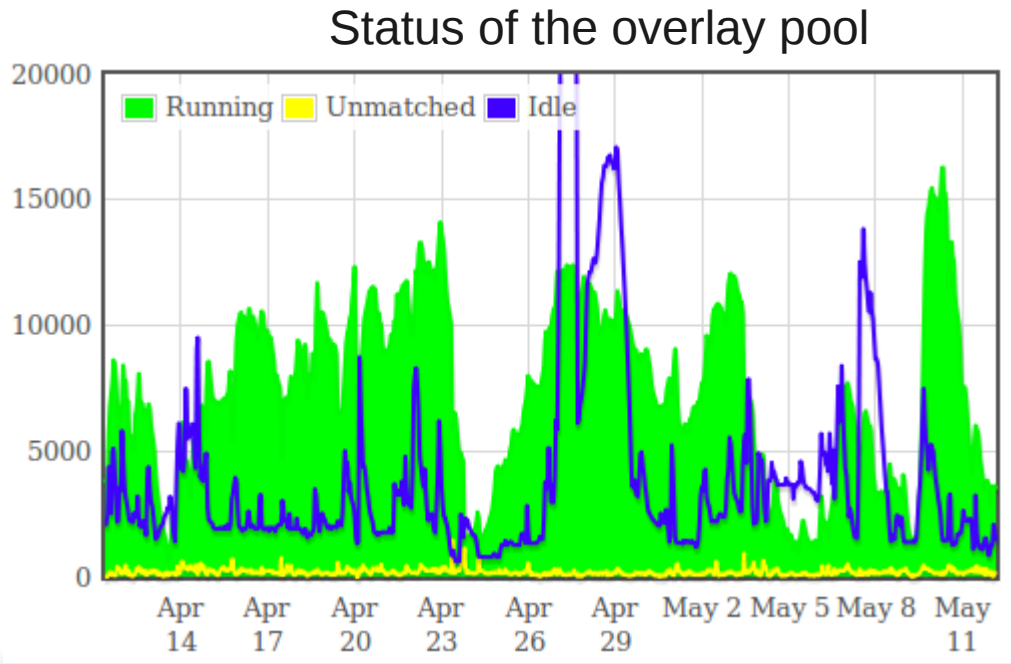
- Experience tells us Grid jobs have relatively flat start and terminate rates
  - Typical O(10/few mins), max O(100/few mins)
  - So pressure can be capped in the O(10) range
- Small range => tuning only when few jobs
  - Using simple heuristic of dividing by 3
  - Just to have a reasonable edge-case policy

$$f(I_s(t)) = \lceil min(I_s(t)/3, C_s) \rceil$$

Operational experience (1)

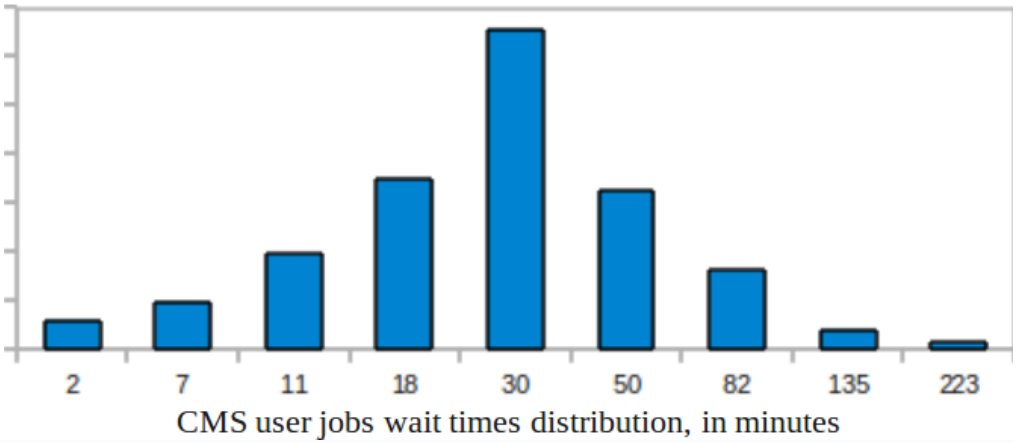
- CMS@UCSD has 2 years of experience
  - Serving O(4k) users
  - Using about O(100) Grid sites
    located in the Americas, Europe and Asia

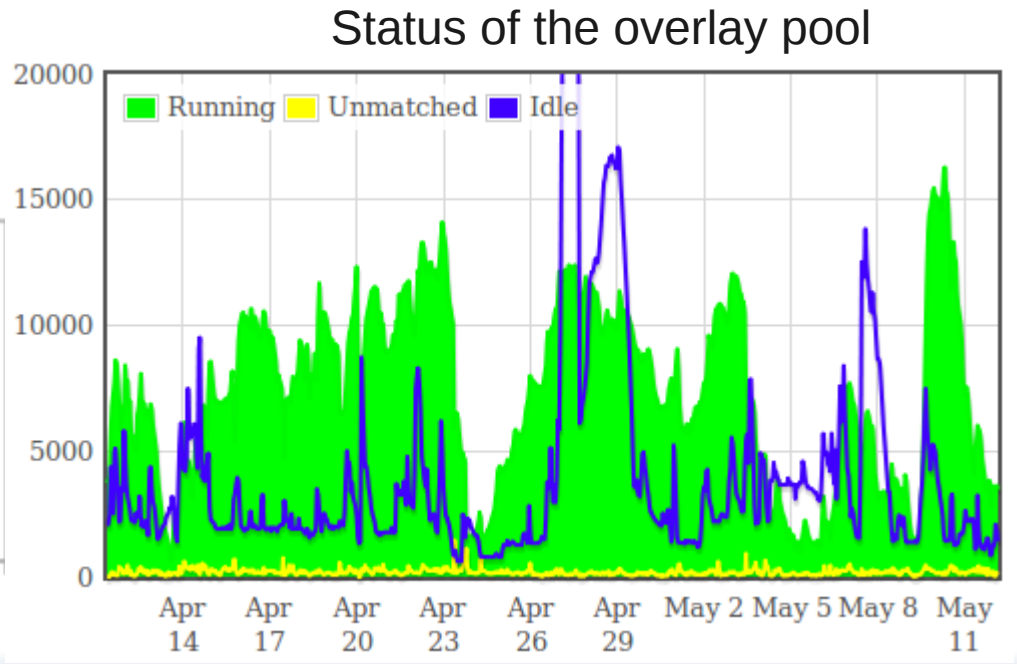




Operational experience (2)

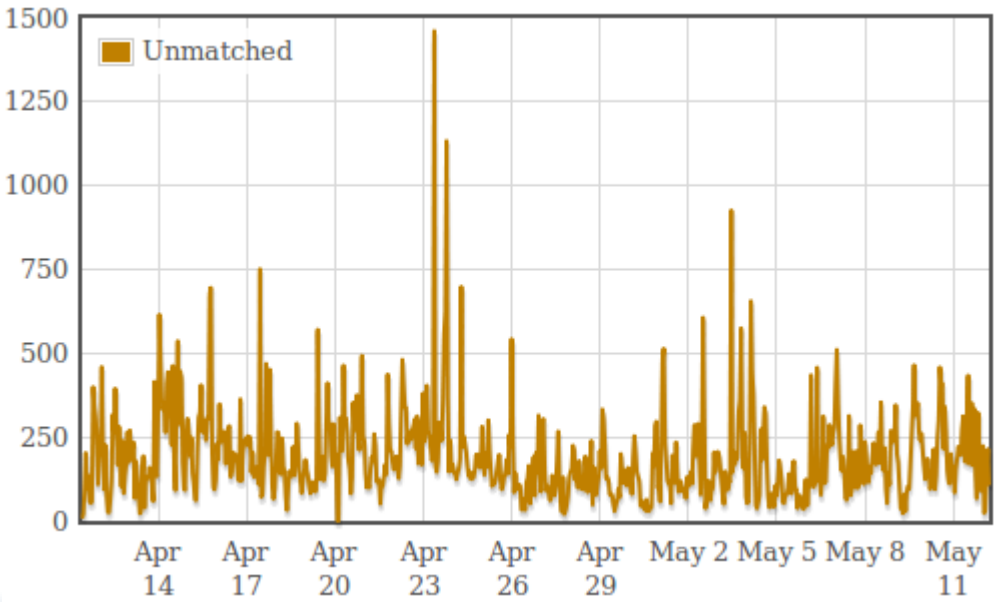
- CMS@UCSD has 2 years of experience
- The glideinWMS logic works very efficiently
  - Quick job startup times



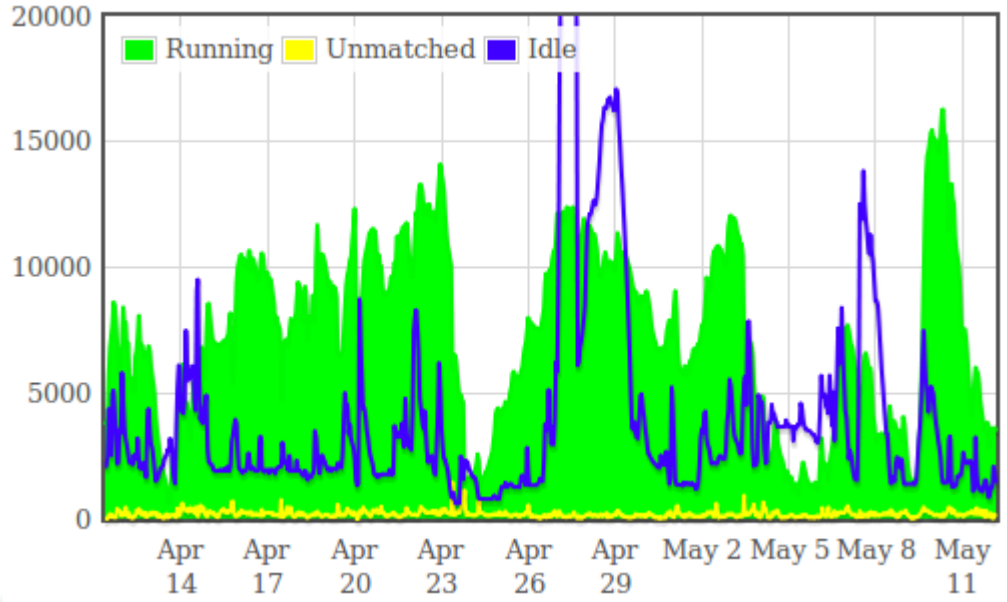


Operational experience (3)

- CMS@UCSD has 2 years of experience
- The glideinWMS logic works very efficiently
  - Quick job startup times
  - Little over-provisioning (~5%)



Status of the overlay pool



Related work

- Non-pilot WMS (i.e. direct submission)
  - gLite WMS and OSG MM
  - More complex and brittle since they require accurate and complete info from Grid sites
- Pilot WMS
  - PANDA
    - Pressure based, with basically constant pressure over time => high load on sites
  - DIRAC and MyCluster
    - Require services at Grid sites to gather site state => many Grid sites do not allow this

Summary

- Direct Grid-wide job scheduling is hard
- **Pilot paradigm** simplifies it by making it **uniform**
- The glideinWMS use **pressure logic**
  - Based on number of pending user jobs only
- **Pressure function capped** => simple rules
- CMS experience at UCSD shows it works
  - **and it works well**

For more information

- The glideinWMS home page
  http://tinyurl.com/glideinWMS
- Relevant papers:
  - I. Sfiligoi et al.,
    "The pilot way to grid resources using glideinWMS,"
    CSIE, WRI World Cong. on, vol. 2, pp. 428-432, 2009,
    doi:10.1109/CSIE.2009.950
  - The CMS Collaboration et al.
    “The CMS experiment at the CERN LHC,”
    J. Inst, vol. 3, S08004, pp. 1-334, 2008,
    doi:10.1088/1748-0221/3/08/S08004

Acknowledgment

- This work is partially sponsored by
  - the US Department of Energy under Grant No. DE-FC02-06ER41436 subcontract No. 647F290 (OSG), and
  - the US National Science Foundation under Grant No. PHY-0612805 (CMS Maintenance & Operations).

Copyright notice

- This presentation contains graphics copyright of Toon-a-day
  that was licensed to
  Igor Sfiligoi
  for use in this presentation
- Any other use strictly prohibited