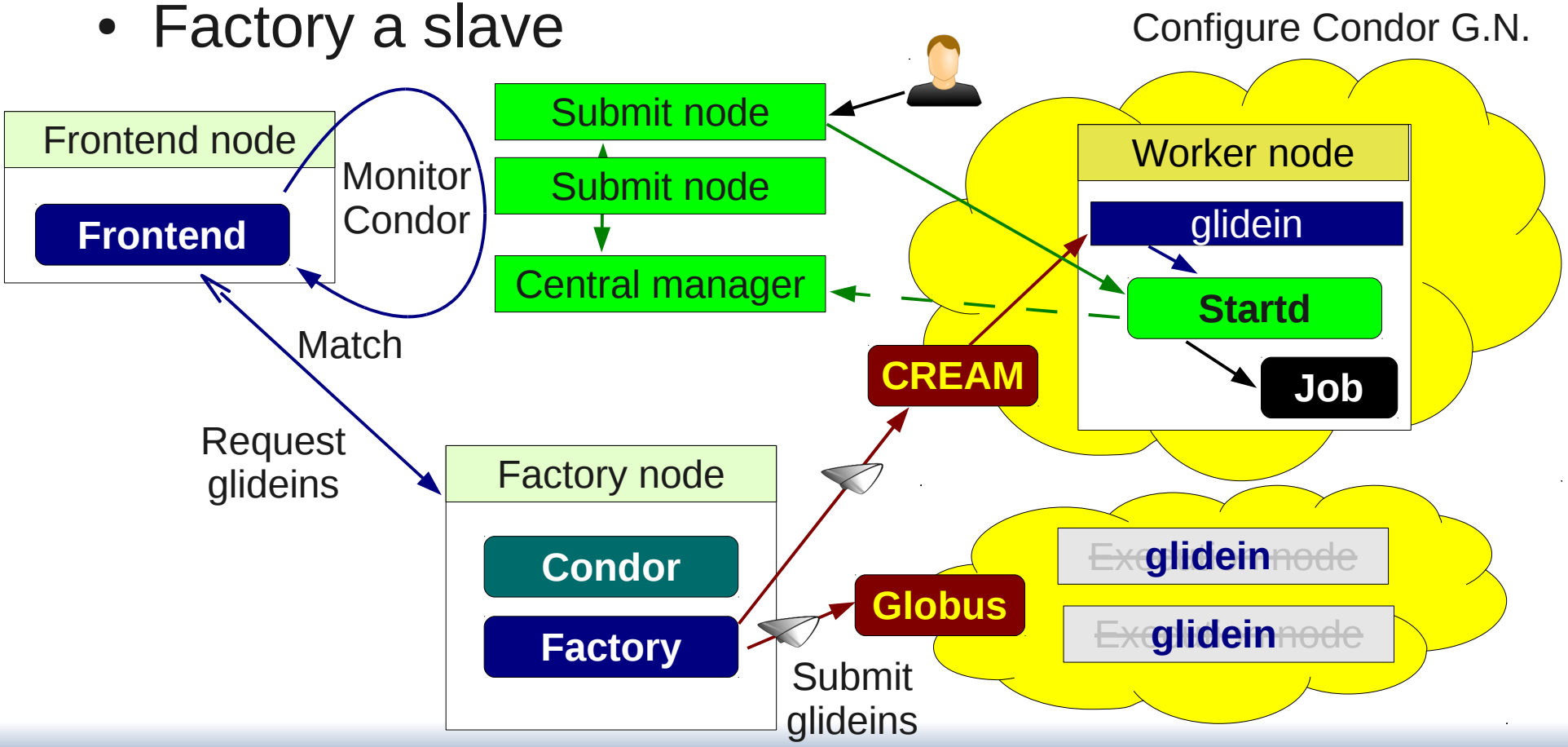
glideinWMS frontend Operations

by Igor Sfiligoi (UCSD)

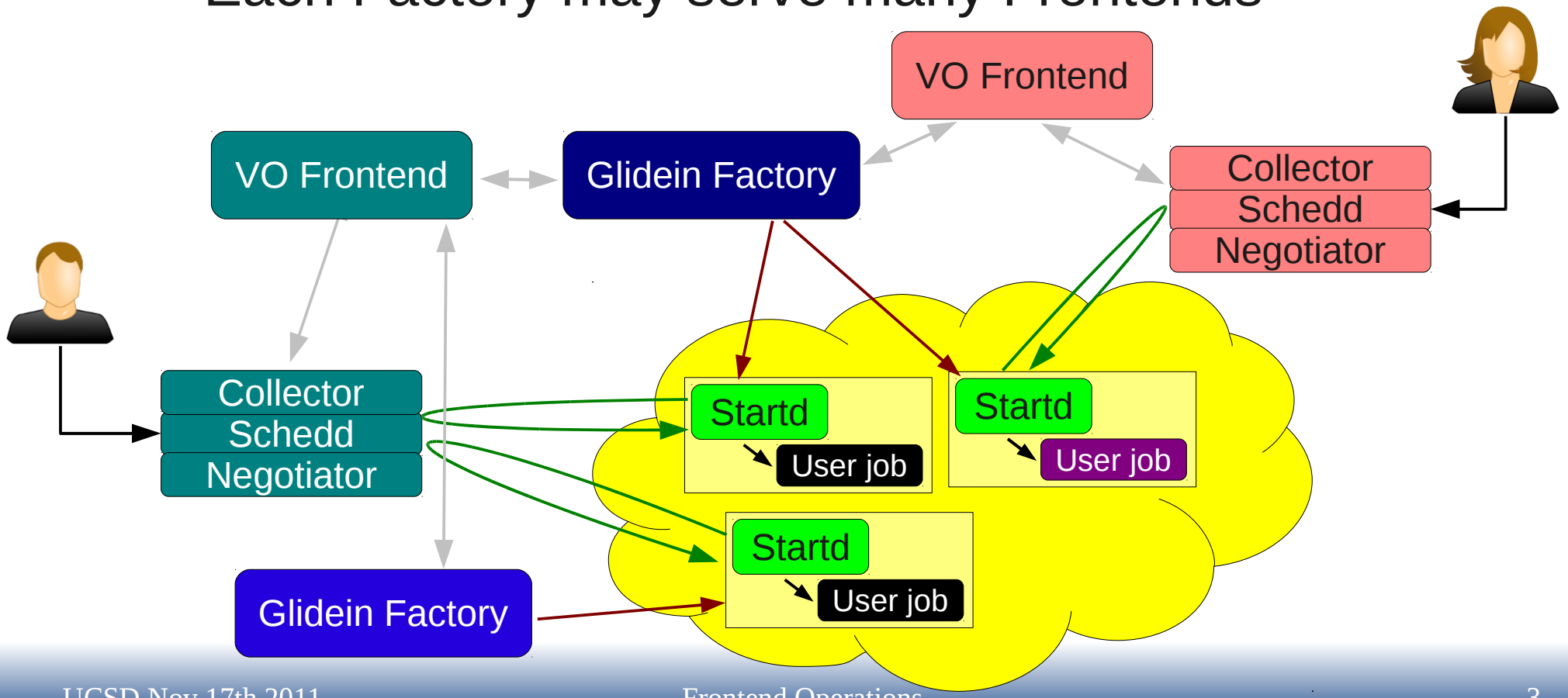
Refresher – Glidein Frontend

- The frontend monitors the user Condor pool, does the matchmaking and requests glideins
  - Factory a slave

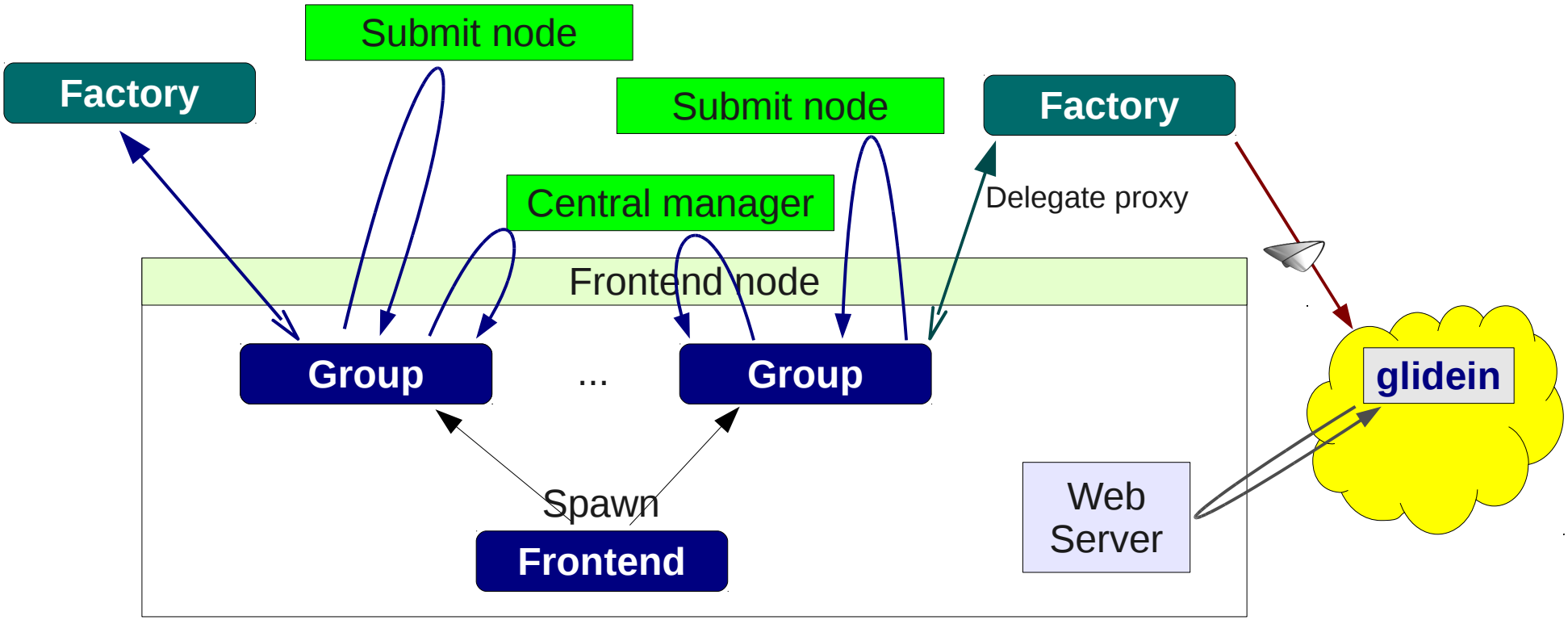


Refresher - Cardinality

- N-to-M relationship
  - Each Frontend can talk to many Factories
  - Each Factory may serve many Frontends

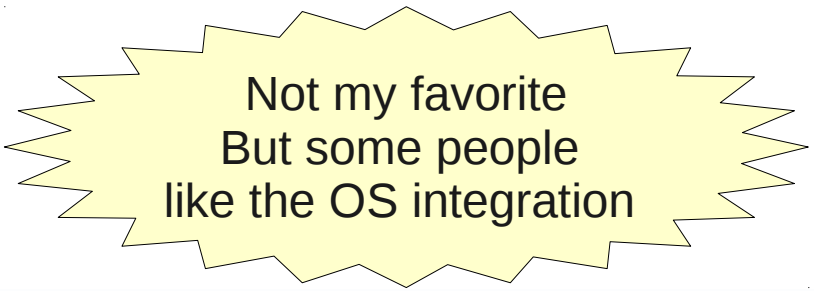


Refresher - Frontend arch



Installation

- Frontend does not need to run as root
  - No UID switching
  - Pick a non-privileged user: e.g. `frontend`
- The glideinWMS team and OSG provide a RPM-based installation
  - Will need to be root to install, configure and start
  - The processes will run as user `nobody`
  - Slightly different ops – will not discuss in this talk



Support packages

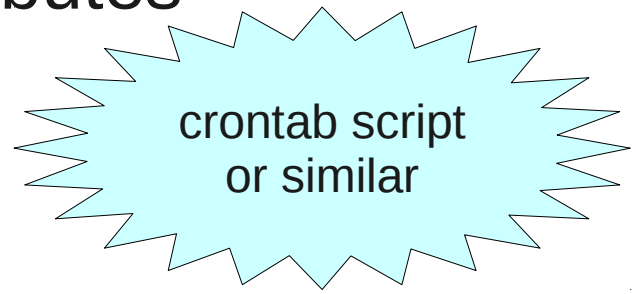
- Will need GSI setup
  (does not need to be root-level, but can be)
- Web server

See Tue talk

- Monitoring libraries
  - Python-rrdtool
    - Usually installed as RPM from EPEL
      (but can be installed in user space as well)
  - javascriptRRD
    - Download from Sourceforge
      http://sourceforge.net/projects/javascriptrrd/
    - Usually just untar, but RPM available, too

Pilot proxi(es)

- Frontend owns the pilot certificate
  - Or several pilot certificates
- Must convert them to a valid proxy
  - Likely with appropriate VOMS attributes
  - Must keep it valid at all times
    (at least 12h left,
    limits glidein lifetime)



- A proxy is also needed to talk to the factory
  - May be different or the same

Configuring the frontend

- glideinWMS installer provides some basic help
  - Good to get the initial template
- Expect to manually configure most options
  - Really not that difficult
  - But readability can be a problem
    - Especially for complex policies
    - A good XML editor would come handy
- More details in Tue talk

Basic ops

- You can have multiple instances installed
  - And running!
  - Each in its own directory tree
- Directory tree created from the XML file
  - Initially by `glideinWMS/creation/create_frontend`
- Two main areas
  - Work area `frontstage/frontend_<name>/`
  - Config area frontstage/instance_<name>.cfg/

Basic ops

- Work area contains init.d-like script
  `./frontend_startup`
- Use it to start, stop and reconfig the frontend
- Reconfig slightly unusual
  - Config file in config area, not work area
    (although there is a similarly named XML file in the work area as well)
  - You must provide the path to the actual config file
    `./frontend_startup reconfig ../instance_`*`blah`*`/frontend.xml`

Monitoring the health of the system

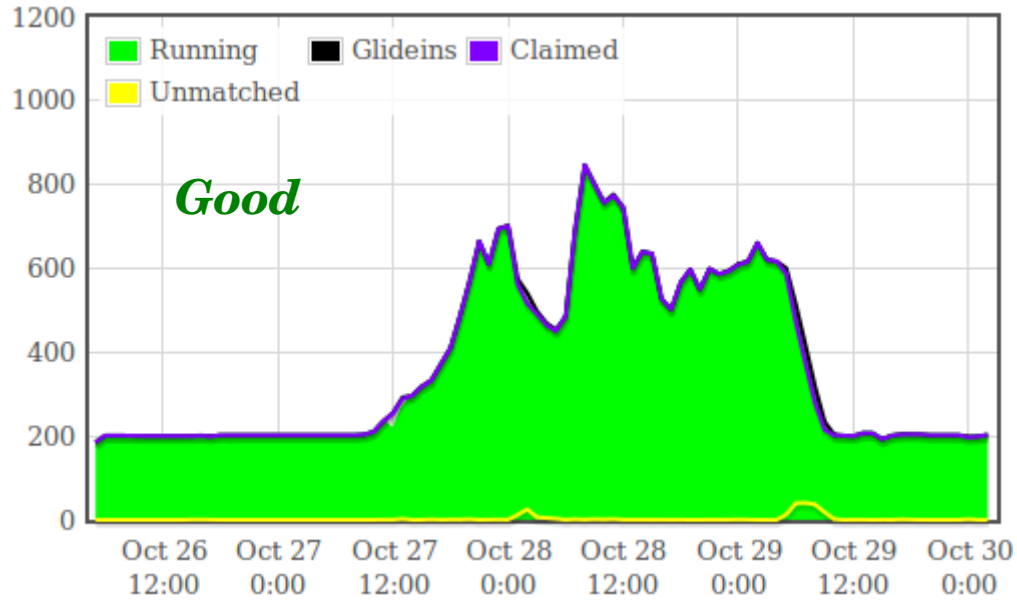
- Six major areas to look after
  - Few unclaimed glideins
    (both globally, and per site)
  - No unmatched jobs
  - Reasonably low restart rate
    (both global, and per site)
  - Reasonably low job failure rate
    (both global, and per site)
  - Negotiation cycle reasonably short
  - Schedd node not overloaded

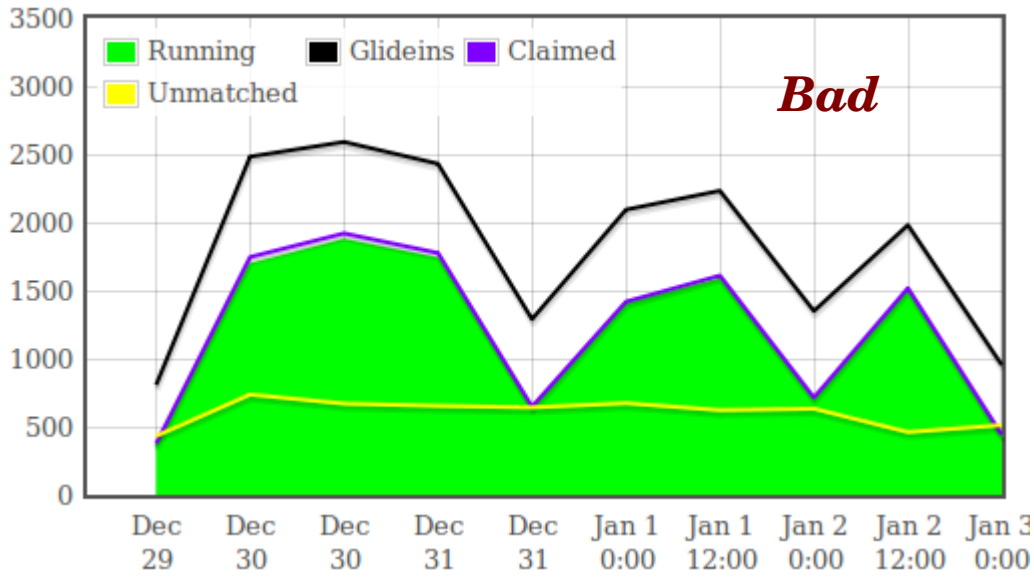
Unclaimed glideins

- Frontend and Negotiator policies are not identical
  - You may end up with glideins that never run any jobs
- The discrepancy can be big enough to be noticed on a global scale
  - But more often it is just for one (or few) sites
- Short spikes are not a problem
  - But long periods are

Unclaimed glideins

- Historical Web monitoring



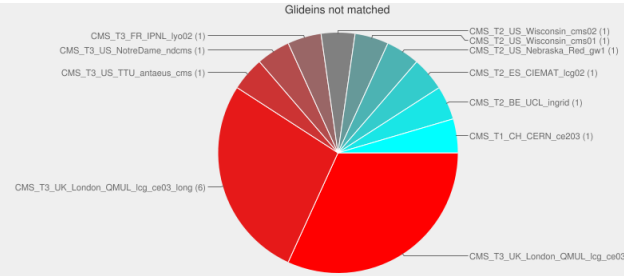


- Daily emails from the Factory
  - Directly from the Fronted in the future

Unclaimed glideins

Needed when debugging

- Latest snapshot
  - Command line tools
    - condor_status and glidein_status.py
  - Web monitoring `frontendGroupGraphStatusNow.html`



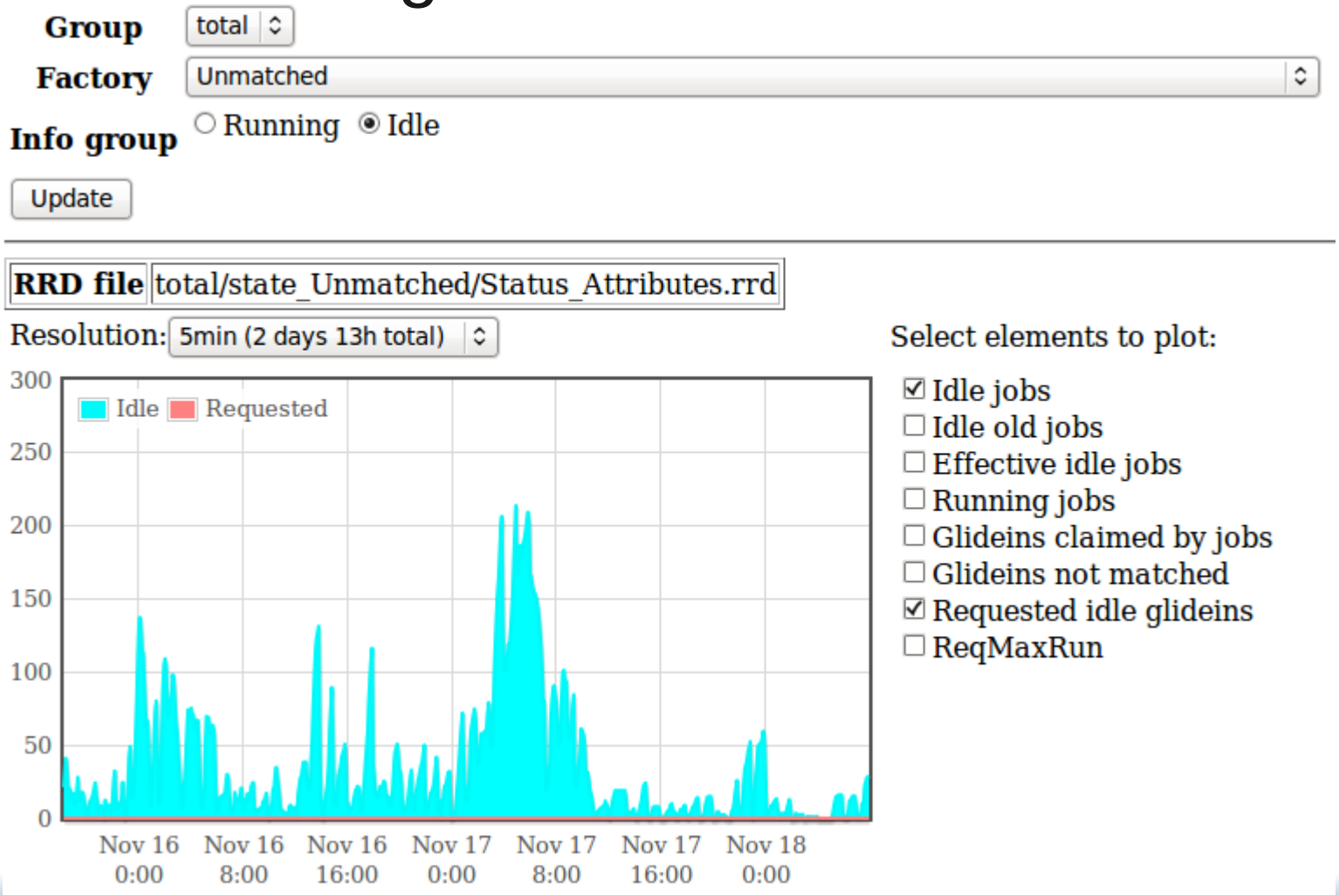
Entry Name	Running Jobs	Idle Jobs	Glideins claimed by jobs	Glideins not matched	% not matched	Requested Max Run	Requested Idle
Totals	3858	31482	3860	22	424.79	263209	8385
CMS_T3_UK_London_QMUL_lcg_ce03_long	0	19	0	7	100	437	8
CMS_T3_UK_London_QMUL_lcg_ce03_long	0	19	0	6	100	437	8
CMS_T1_CH_CERN_ce203	114	23	114	1	0.87	1328	14
CMS_T2_BE_UCL_ingrid	23	200	23	1	4.17	2144	76
CMS_T2_ES_CIEMAT_lcg02	28	200	28	1	3.45	2468	76
CMS_T2_US_Nebraska_Red_gw1	60	205	60	1	1.64	2545	78
CMS_T2_US_Wisconsin_cms01	40	200	40	1	2.44	2990	76
CMS_T2_US_Wisconsin_cms02	44	200	44	1	2.22	2963	76
CMS_T3_FR_IPNL_lyo02	9	29	9	1	10	1072	18
CMS_T3_US_NotreDame_ndcms	0	17	0	1	100	434	10
CMS_T3_US_TTU_antaeus_cms	0	19	0	1	100	437	12
CMS_T1_CH_CERN_ce208	89	23	89	0	0	1328	15
CMS_T1_US_FNAL_ce2	10	17	10	0	0	434	11

Unmatched jobs

- The other side of the problem
  - Glideins never asked for some jobs
- Two possible reasons
  - Wrong Frontend matchmaking policy
  - No available Factory entries to serve the job

Unmatched jobs

- Web monitoring



Unmatched jobs

- Figuring why not trivial
  - You have to actually split the python expression
  - Plans to provide the tool to do this, but nothing yet
- You will need the list of Factory entries
  - Frontend advertise the list in the Collector
    ```
    condor_status -any -const 'GlideinMyType=="glideresource"'
    ```
  - Else you can get it from the Factory Collector(s)
    ```
    glideinWMS/tools/wmsTxtView.py -pool IP
    glideinWMS/tools/wmsXMLView.py -pool IP
    ```
  - Or from Web pages

Restarted jobs

- Any restart == badput
- Several reasons
  - Remote node just died (rare)
  - Site preemption policy (not much you can do)
  - Wrong glidein lifetime (Factory problem)
  - Wrong job lifetime estimate
  - Exceeded resource limits (e.g. memory)
  - Frontend matchmaking logic problem
- condor_q is your friend here

```
condor_q -format '%i\n' NumJobStarts
```

Job failures

- Jobs can fail for any reason
  - You should monitor the ExitCode
    ```
    condor_history -back -const 'JobStatus==5' -format '%i\n' ExitCode
    ```
  - Knowing what users run often needed to interpret errors
- For common WN errors, Frontend admin should create appropriate validation script
  - So glideins fail, not user jobs

Negotiation Time

- The negotiation time should be < 1min
  - If much longer, glideins may terminate without running any jobs
  - Monitor the NegotiatorLog on CM
- Possible causes
  - CPU starvations (e.g. other processes)
  - Autocluster explosion
    - Condor tries to be smart about Matchmaking
    - But if users don't cooperate, cannot do much

Submit node health

- The submit node may have interactive users
  - Can overload the node
  - Portals may have a similar effect (e.g. WMAgent)
- Condor very memory hungry
  - 1M per running job
  - If RAM not enough → trashing → system crash
- Should monitor the system health
  - Ganglia, Cacti, …
  - Possibly with automated alarms

Pointers

- The official project Web page is http://tinyurl.com/glideinWMS
- glideinWMS development team is reachable at glideinwms-support@fnal.gov

Acknowledgments

- The glideinWMS is a CMS-led project developed mostly at FNAL, with contributions from UCSD and ISI
- The glideinWMS factory operations at UCSD is sponsored by OSG
- The funding comes from NSF, DOE and the UC system