glideinWMS factory system setup

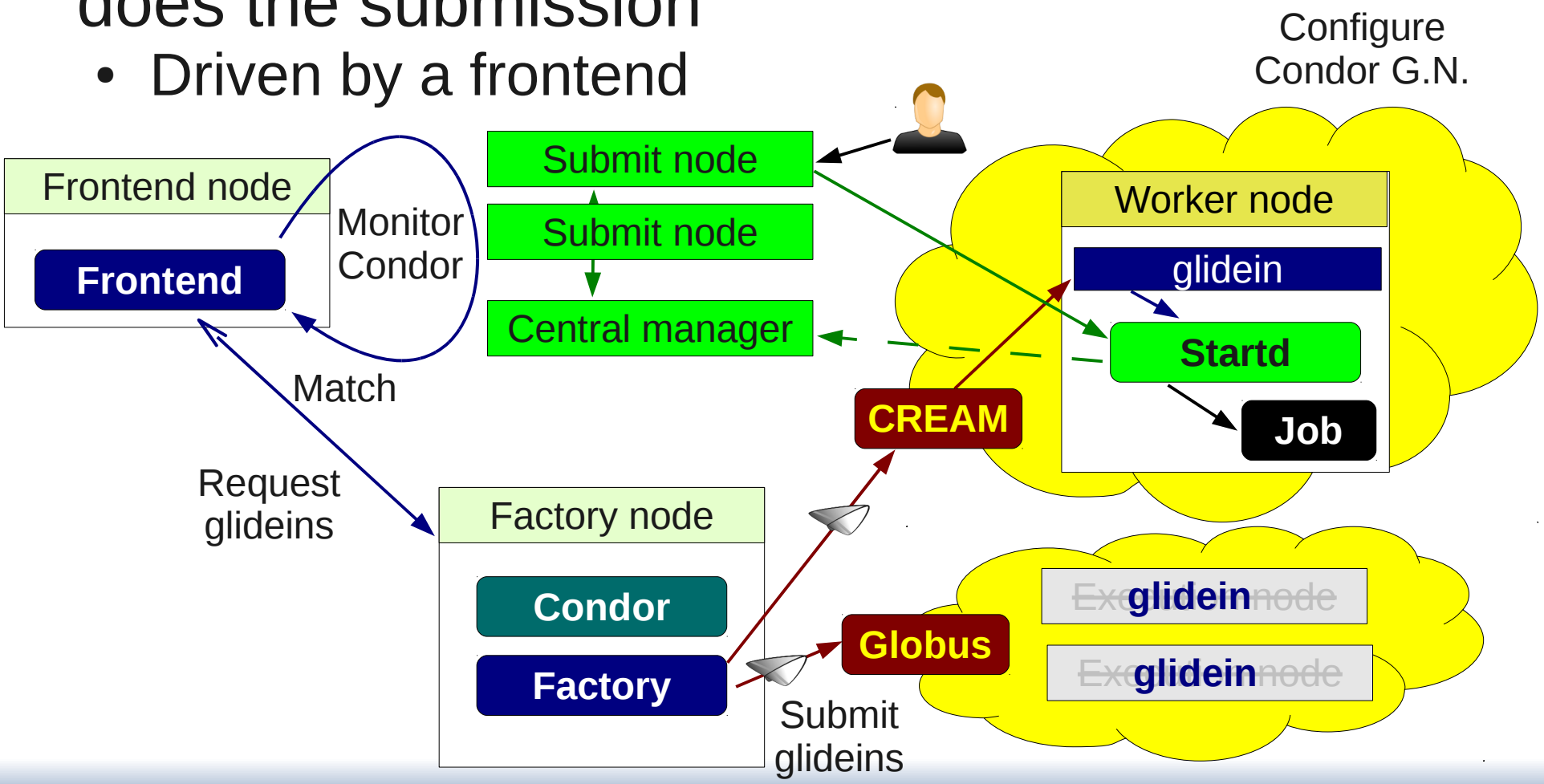
by Igor Sfiligoi (UCSD)

Overview

- Refresher
- Component overview
- Description of the components
- Hardware needs

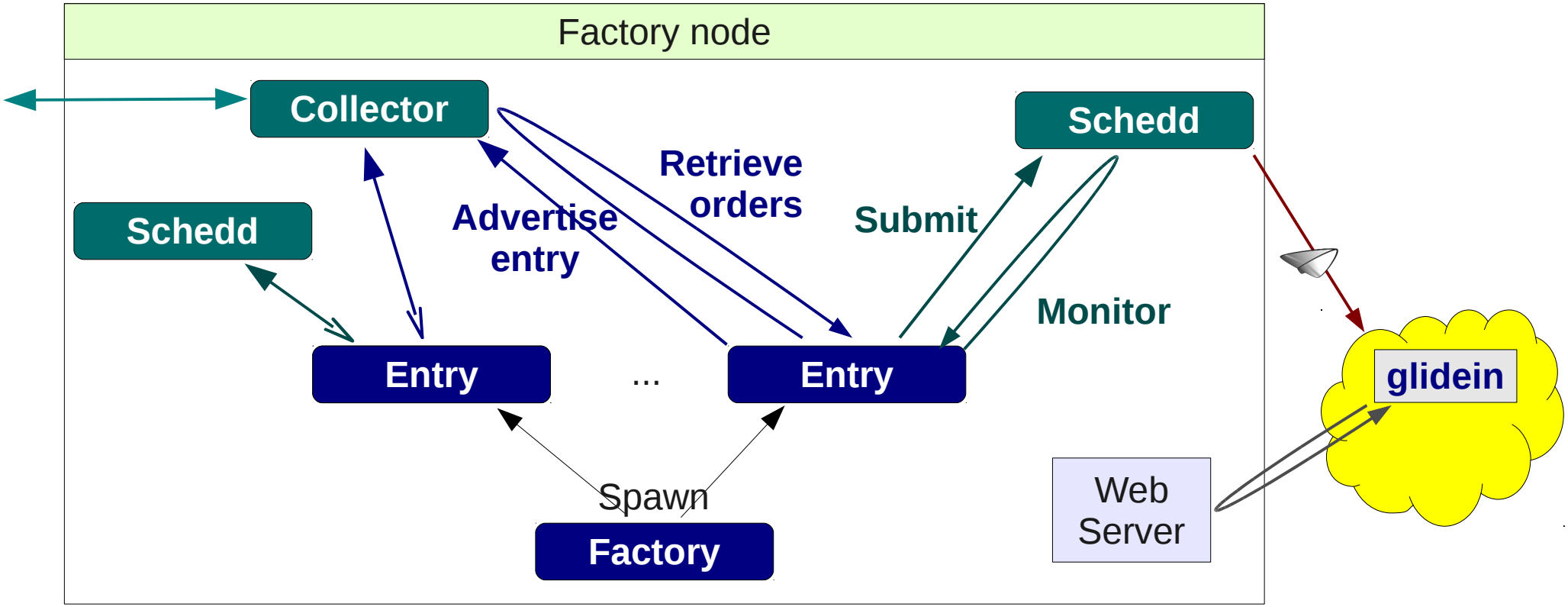
Refresher – Glidein factory

- The glidein factory knows about the sites and does the submission
  - Driven by a frontend



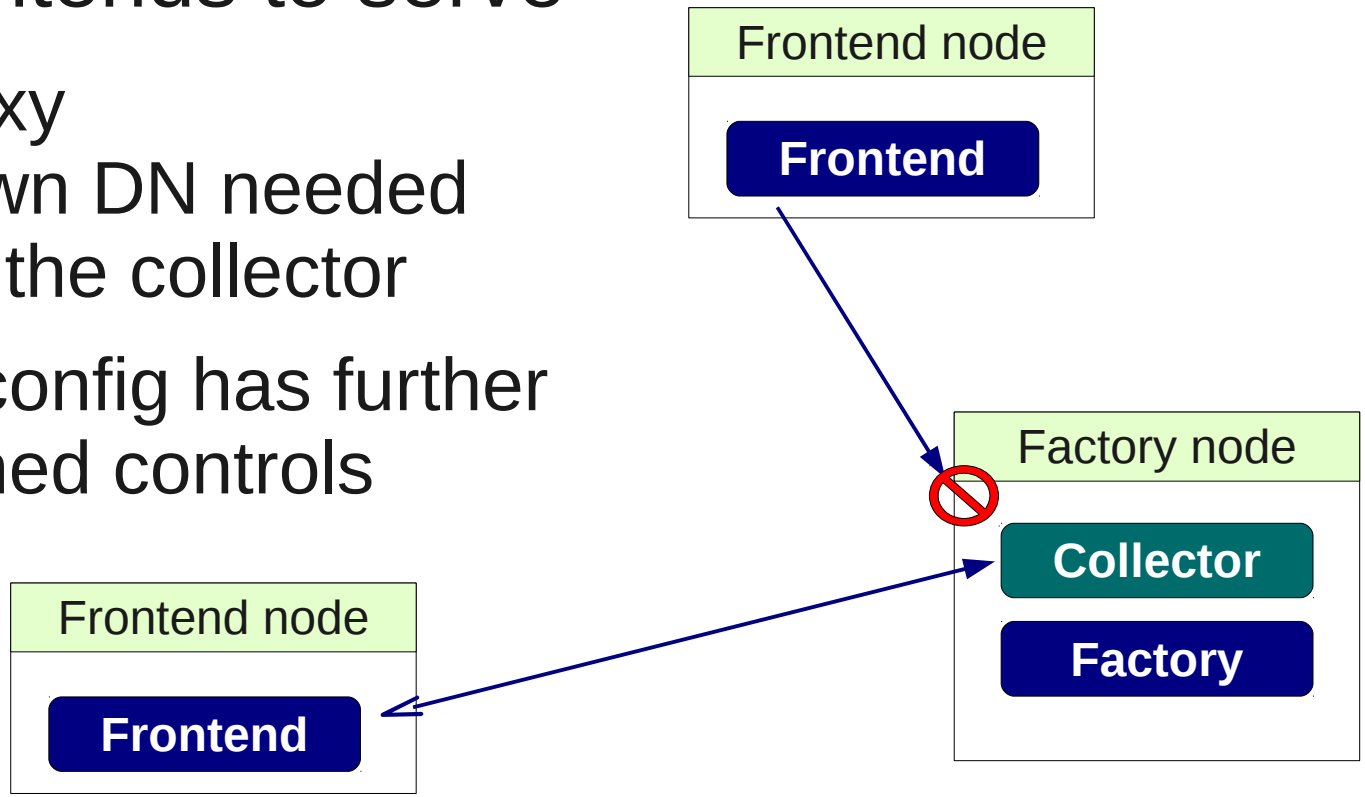
Refresher – Factory arch

- Condor(-G) handles most of the work
- Collector and Web server needed for externals



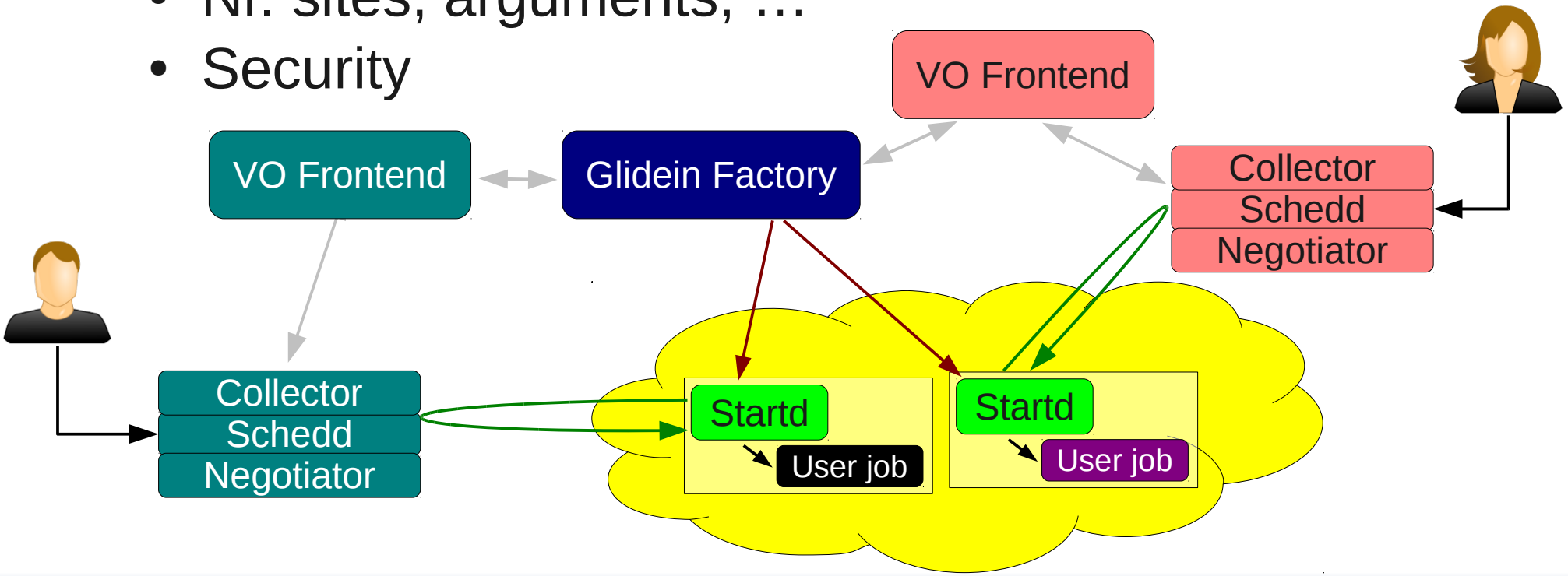
Refresher - AuthZ

- The factory admin decides which Frontends to serve
  - Valid proxy with known DN needed to talk to the collector
  - Factory config has further fine grained controls



Refresher - Cardinality

- A single factory may serve multiple frontends
  - The OSG factory @ UCSD serves O(10)
- Different frontends could be treated differently
  - Nr. sites, arguments, …
  - Security



Components of a factory

- The factory runs:
  - Condor
  - Web server
  - GlideinWMS
- It also needs:
  - OS
  - condor_root_switchboard
  - GSI security setup (cert & CAs)
  - Libraries (RRDTool, JavascriptRRD, ...)

Operating System

- GlideinWMS is supported on RHEL5 compatible systems
  - But *may* run on other *NIX systems as well
- Requires python 2.4.3 or above
- Root access is needed to install and operate some of the services
- Public TCP/IP networking needed, both incoming and outgoing

Libraries

- Needed on top of standard OS libraries:
  - RRDTool – available as RPM in EPEL and other sources
    http://oss.oetiker.ch/rrdtool/
  - JavascriptRRD – available from Sourceforge
    Untarring suffice, but RPM install also available
    http://sourceforge.net/projects/javascriptrrd/

Web server

- Used for both delivering payload to worker nodes and for monitoring
- No dynamic content served
  - Only static files
- Any Web server will work
  - System provided Web server recommended
  - The glideinWMS installer will also properly configure it, if desired (i.e. put it in static mode)
    - De-install existing Web server first, if needed

GSI setup

- CA & CRLs needed
  - Both for communication with frontend and resource providers
  - Typically delivered via VDT in OSG (but other means acceptable)
    https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallCertAuth
- Host cert needed for communication with frontends
  - Service cert acceptable as well
- Full Grid client software recommended
  https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallOSGClient
  - Not essential, but useful for operations

Condor services

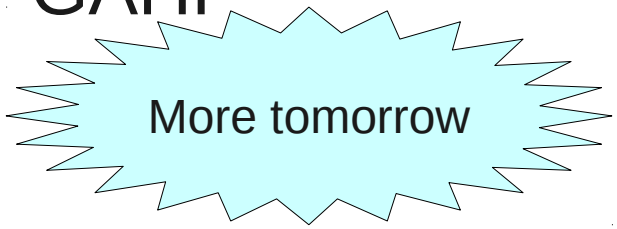
- Condor is used for
  - Collector for communication with frontend
  - Condor-G for communication with resource providers
    - Just a regular Schedd from user point of view
    - Jobs are of the “Grid universe”
- Everything started and controlled by a Condor master
  - As usual for any Condor setup

Condor Collector

- Essentially a whiteboard
  - Factory and frontend advertise and read ClassAds
  - Also used to identify the schedds
- Authorization
  - FS/UID based for local services (factory&schedds)
  - GSI based for network access
    - Typically using host cert for own identification
    - Must whitelist the remote DNs that are allowed
    - Must provide a DN->ID mapping
  - Each classad has the identity (ID) of advertiser
- No need to be root (but usually is)

Condor-G

- Many schedds
  - For scalability and reliability issues
  - Called `schedd_glideins`*X*`@`*nodename*
- Schedds only accessible from the local node
- Schedd really just the interface
  - The gridmanager does all the real work
  - The actual protocol implemented by GAHP
    - One GAHP binary x Grid type
- Schedd must run as root
  - Will switch UIDs for security reasons



condor_root_switchboard

- Simplified sudo-equivalent
  - With a different config file
  - Very restricted
    - From specific UID to other UIDs (no root)
    - Allows for target_uid directory creation within whitelisted directories
- Must be installed by root
  - And with SETUID privileges
- Used by the factory
  - All target UID must be whitelisted

glideinWMS

- The glideinWMS SW can be installed anywhere
- The factory will run as non-privileged user
  - Typically `gfactory`
  - But this user must be a “Condor superuser”
- Factory needs “UID switchable” dirs
  - For user proxies and logs
  - Base dir must be fully owned by root
  - And whitelisted for `condor_root_switchboard`
- A Web area must be writable by gfactory

glideinWMS Installer

- The glideinWMS comes with an installer that can help install all the components (excluding the OS)
- But it is not strictly needed
  - It is just a configuration helper
  - Admins encouraged to use it as a reference, but do the final configuration independently

Hardware needs

- HW needs scale with the number of sites supported
  - Number of glideins a 2nd order effect
  - Disk space scales primarily with number of glideins
- OSG factory experience
  - Using ~12G of RAM (16G avail)
  - Requires SSD; spinning disks IO limited (space: <100G)

Summary

- RHEL5-compatible Linux recommended
- Requires incoming and outgoing TCP/IP
- Most services running as root
- The factory running as unprivileged user
  - Typically `gfactory`
  - Uses condor_root_switchboard for UID switching

Pointers

- The official project Web page is http://tinyurl.com/glideinWMS
- glideinWMS development team is reachable at glideinwms-support@fnal.gov
- OSG glidein factory at UCSD
  http://hepuser.ucsd.edu/twiki2/bin/view/UCSDTier2/OSGgfactory
  http://glidein-1.t2.ucsd.edu:8319/glidefactory/monitor/glidein_Production_v4_1/factoryStatus.html

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