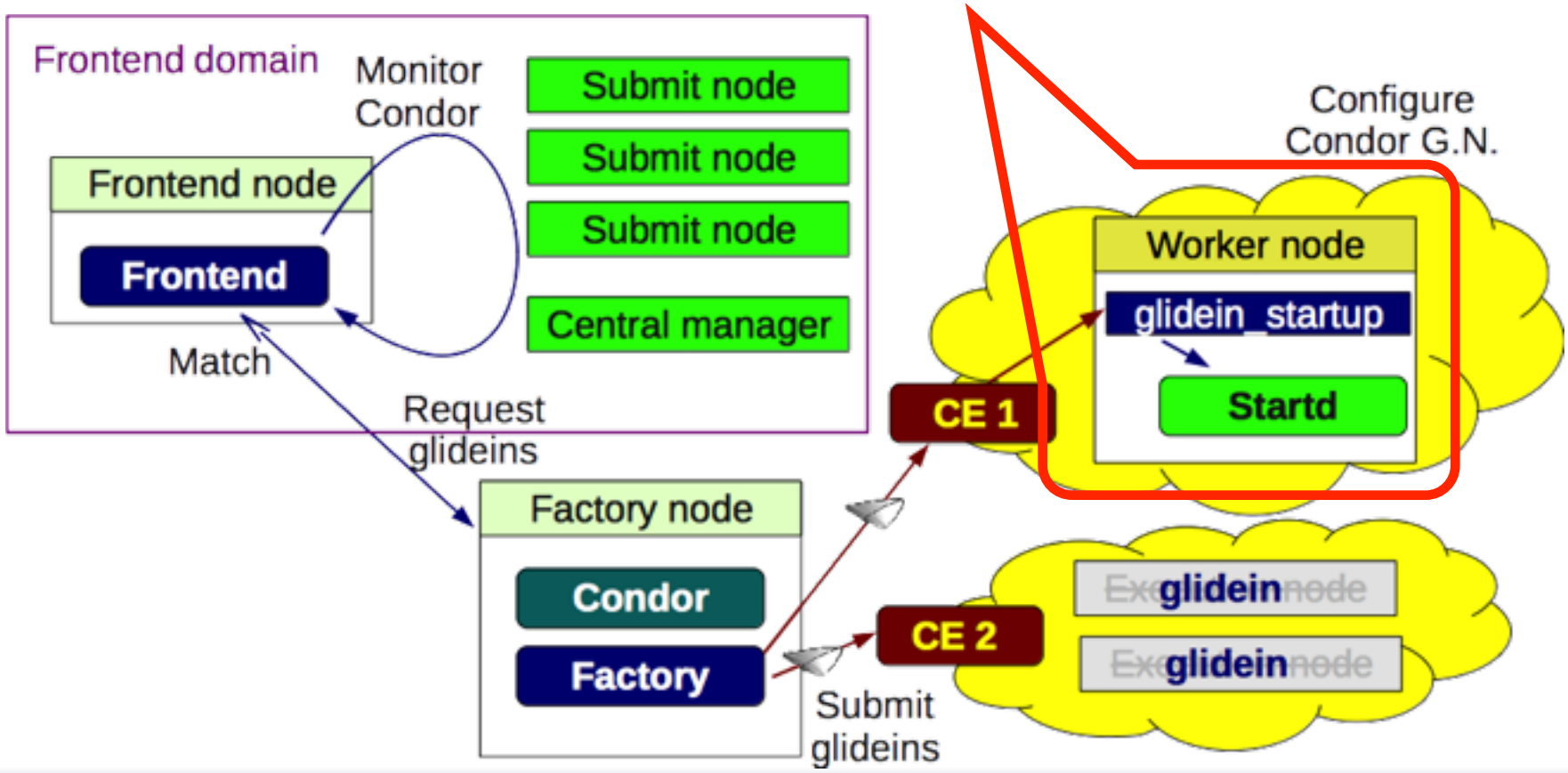
Glidein Internals

How they work and why

by Igor Sfiligoi, Jeff Dost (UCSD)

Refresher – glidein_startup

- the glidein_startup script configures and starts Condor on the worker node



Refresher - glidein_startup tasks

- Download scripts, parameters and Condor binaries
- Validate node (environment)
- Configure Condor
- Start Condor daemon(s)
- Collect post-mortem monitoring info
- Cleanup

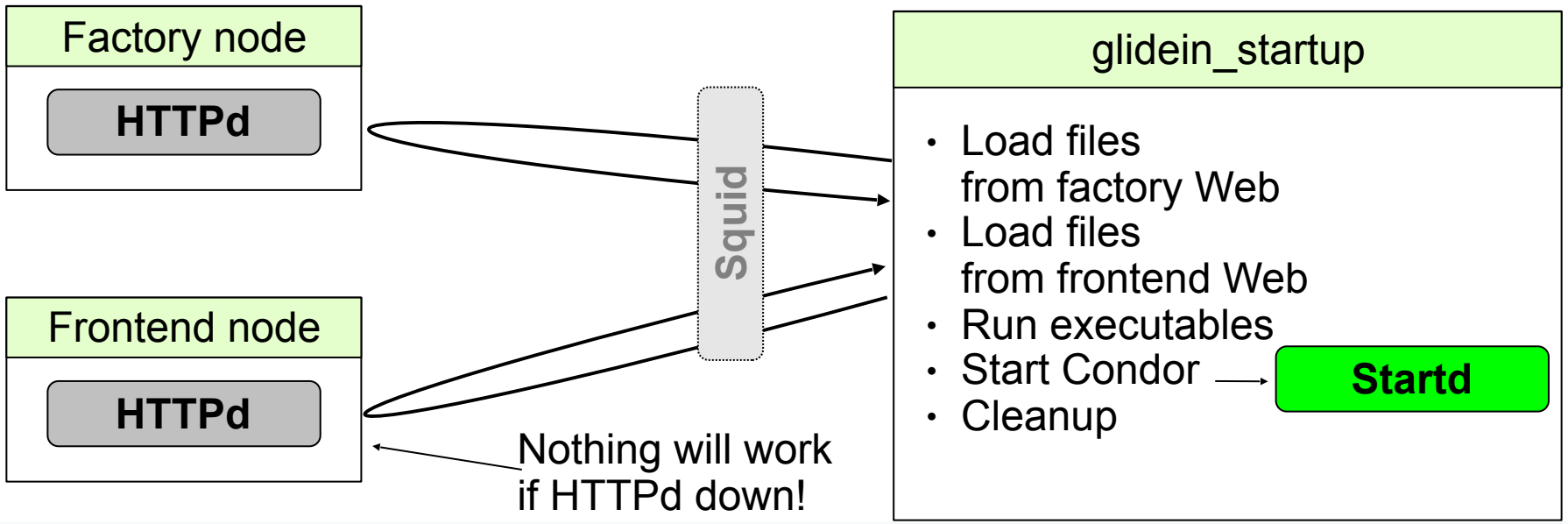
glidein_startup details

Work area

- Before writing any files, a work area must be chosen
  - CWD is often not the right choice
- Where to cd into is provided as an argument
- A unique tmp dir is created there
  - So several glideins running on the same node don't step on each other toes

Refresher - Downloading files

- Files are downloaded via HTTP
  - From **both the factory and the frontend** Web servers
  - Can use local Web proxy (e.g. Squid) if available at site
  - Mechanism is tamper proof and cache coherent



URLs

- All URLs are passed to glidein_startup as arguments
  - Factory Web server
  - Frontend Web server
  - Squid, if any
- glidein_startup arguments are defined by the factory
  - Frontend passes Web URL to the Factory via request ClassAd

Cache coherence

- **Files never change** once uploaded to the Web area
- If the source file changes, a file with a **different name** is created on the Web area
  - Essentially **fname.date**
  - pilots map logical to physical file name to access correct version
- glideins submitted before reconfig don't see changes
  - factory / FE reconfigs only affect newly submitted glideins

Making the download tamper proof

- **SHA1 hashes** are calculated on factory and FE for all files
  - written into **hash file** published on web sever
- pilots calculate SHA1 of each downloaded file
  - compares with published hash file to verify integrity
- factory and FE take hash of the hash files and send them as a glidein_startup arguments
  - this ensures hash files themselves weren't corrupted

Node validation

- glidein_startup runs scripts / plugins provided by both factory and frontend
  - A map file has a flag to distinguish scripts from “regular” files

# Outfile	InFile	Exec/other
constants.cfg	constants.b8ifnW.cfg	**regular**
cat_consts.sh	cat_consts.b8ifnW.sh	**exec**
set_home_cms.source	set_home_cms.b8ifnW.source	**wrapper**

- If a script returns with **exit_code !=0**, glidein_startup stops execution
  - Condor was never started → no user jobs ever pulled
  - Will sleep for 20 min, then terminate → blackhole protection

Condor configuration

- Condor config values come from files downloaded by glidein_startup
  - Static values come from both factory and frontend config files
  - More details at http://glideinwms.fnal.gov/doc.prd/factory/custom_vars.html
- Scripts can add, alter or delete any attribute
  - Based on dynamic information
  - Either discovered on the WN
    or by combining info from various sources
  - More details at http://glideinwms.fnal.gov/doc.prd/factory/custom_scripts.html

Example (pseudo-)script

```
# check if CMSSW installed locally
if [ -f "$CMSSW" ]; then
   # source the environment
   source "$CMSSW"
else
   # fail validation
   echo "CMSSW not found!\n" 1>&2
   exit 1
fi

# publish to Condor
add_config_line "CMSSW_LIST" "$CMS_SW_LIST"
add_condor_vars_line "CMSSW_LIST" "S" "-"  "+" "Y" "Y" "-"

# if I got here, passed validation
exit 0
```

More details at http://glideinwms.fnal.gov/doc.prd/factory/custom_scripts.html

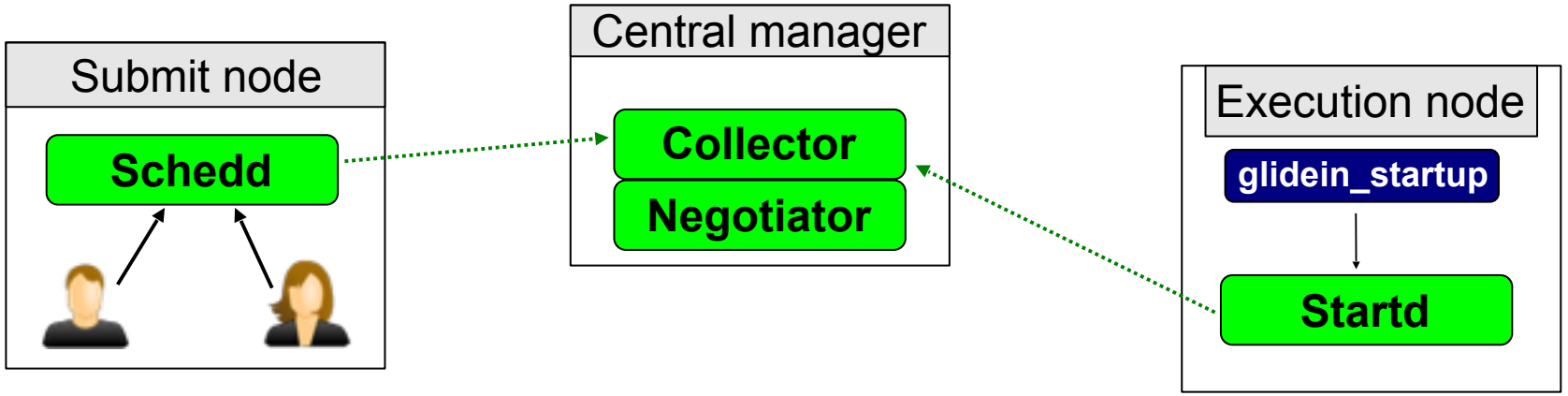
Example attributes

- Factory defining num cores available to glidein
  - GLIDEIN_CPUS = “8”
- Frontend requesting partitionable glideins
  - SLOTS_LAYOUT = “partitionable”

More details at glideinwms.fnal.gov/doc.prd/factory/custom_scripts.html

Condor startup

- After validation and configuration, glidein just start `condor_master`
  - Which in turn starts `condor_startd`
- It is just regular Condor from here on
  - Any policy the VO needs must be part of Condor configuration



Post mortem monitoring

- After a glidein terminates, the logs are sent back to the Factory
- The logs are mined for job information
  - Number jobs it ran
  - Job exit codes, exit by signal?
  - Efficiency (time spent running user jobs over total wall time)
- Condor logs are compressed and copied into stderr
  - Most reliable way to retrieve the logs
  - File transfer over the Grid is never guaranteed

Cleanup

- glidein_startup will remove all files before terminating
  - Unless it's killed by the OS, of course
- Condor cleans up the user job disk area after every job
  - Users should not expect anything to survive their jobs

Glidein lifetime

Glidein lifetime

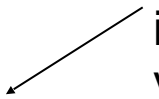
- **Glideins are temporary resources**
  - Must go away after some time
- **We want them to go away by their own will**
  - So we can monitor progress and clean up
  - As opposed to being killed by Grid batch system
- Condor are daemons configured to die by themselves
  - **Just need to tell them when**

Limiting glidein lifetime

- Hard End-of-Life deadline
  - Condor will terminate if it is ever reached
  - Any jobs running at that point in time will be killed
    - Results in waste
- Deadline is site specific
  - Set in the Factory config
  - Controlled by `GLIDEIN_Max_Walltime`
- Condor advertises it in the glidein ClassAd
  - Attribute **`GLIDEIN_ToDie`**

Deadline for job startup

- Starting jobs just to kill them is wasteful
  - Glideins set an earlier deadline for job startup
  - After the deadline, Condor will terminate after current job completes
  - Advertised as **`GLIDEIN_TORETIRE`**
- Need to know how long the expected job lifetime is
  - Should be **provided by the Frontend**
  - Parameter **`GLIDEIN_Job_Max_Time`**
  - ToRetire=Max_Walltime-Job_Max_Time



Termination due to non-use

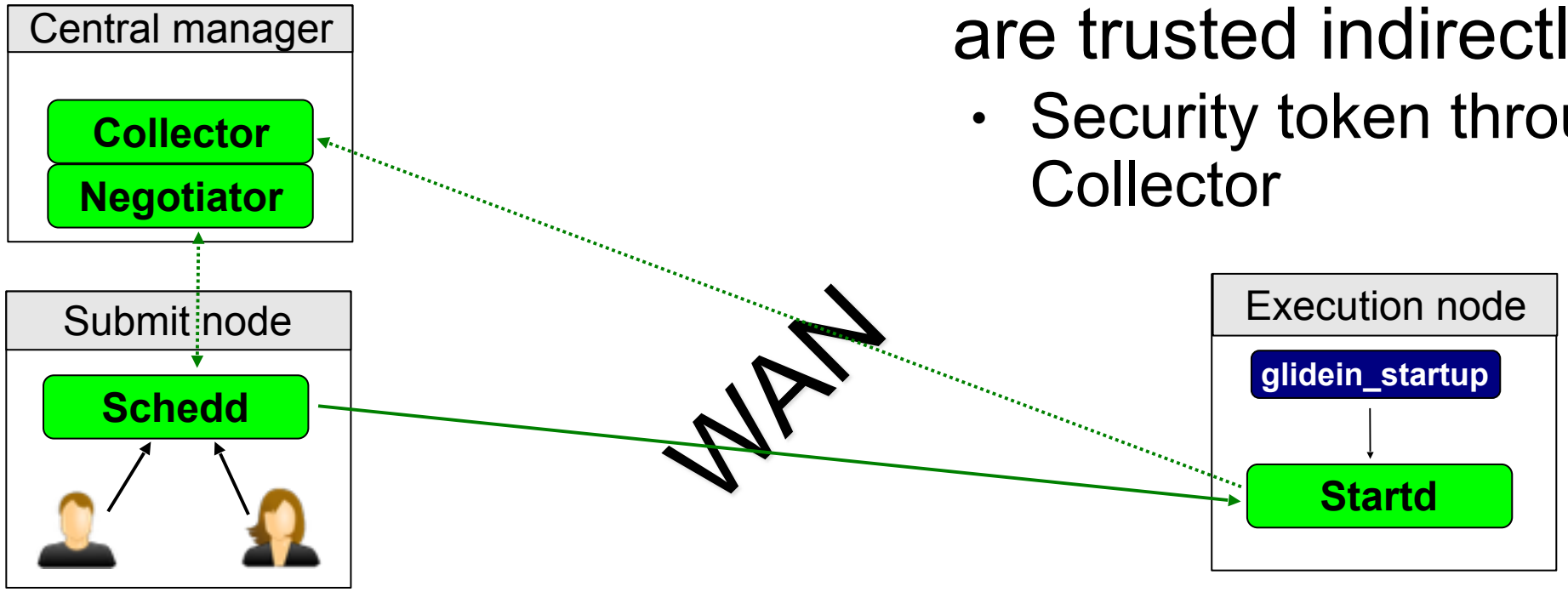
- **glideins will self-terminate early if not used**
  - To limit waste
- Reasons for hitting this
  - No more user jobs (spikes)
  - Policy problems (Frontend vs Negotiator)
- Typically set as single value across the Condor pool
  - **Can be set by either Frontend** or Factory
  - Controlled by **`GLIDEIN_Max_Idle`**
  - Defaults to **20 mins**

To give Negotiator the time to match the glidein

Security considerations

Talking to the rest of Condor

- Glideins will talk over WAN to the rest of Condor
  - Strong security a must
- Collector and glidein must whitelist each other
- Schedd and glidein are trusted indirectly
  - Security token through Collector



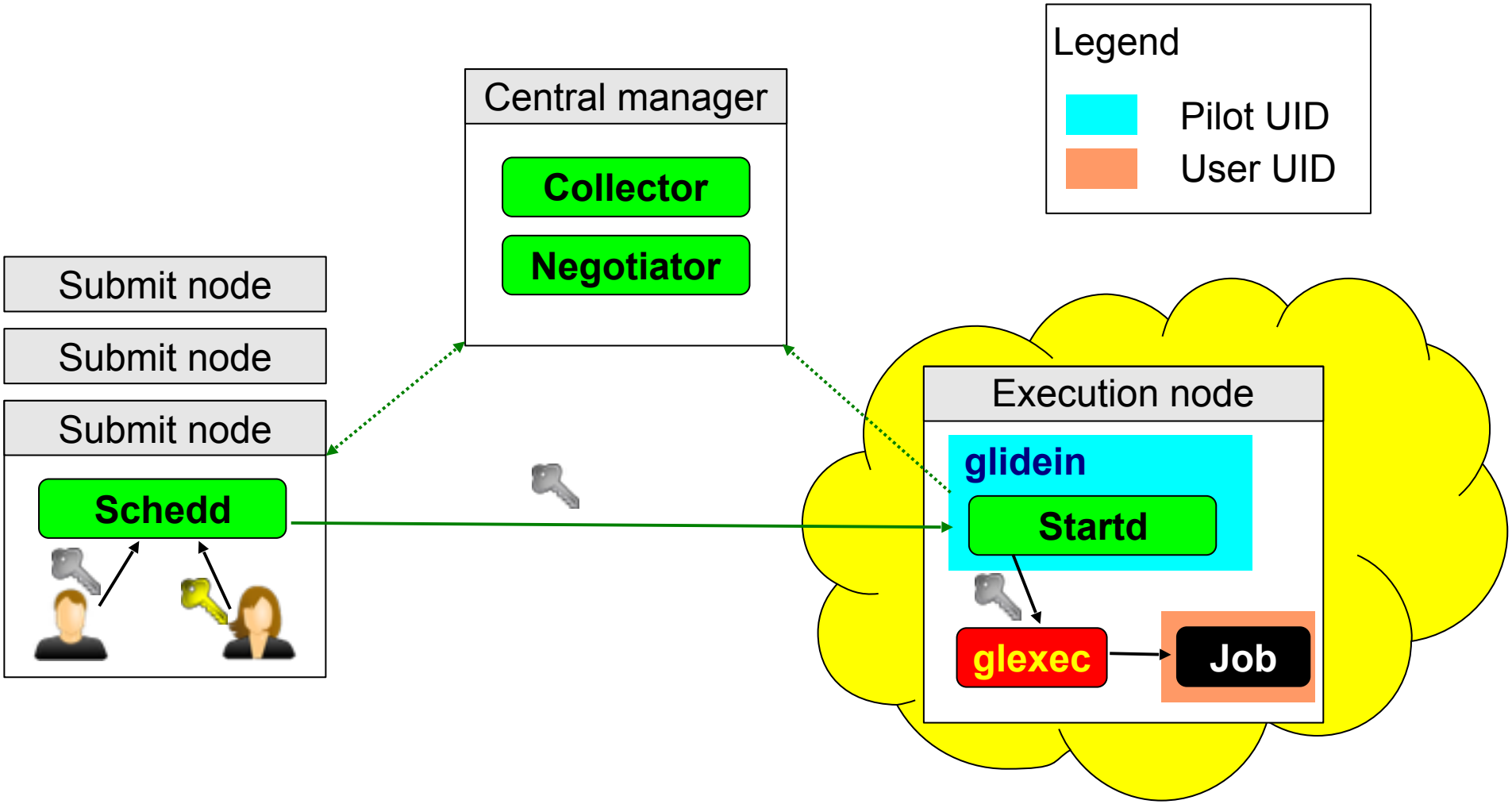
Multi User Pilot Jobs

- A glidein typically starts with a proxy that does not represent a specific user
- Three potential security issues:
  - Site does not know who the final user is (and thus cannot ban specific users, if needed)
  - If 2 glideins land on the same node, 2 users may be running as the **same UID** (no system protection)
  - **User and pilot code** run as the same UID (no system protection)

glexec

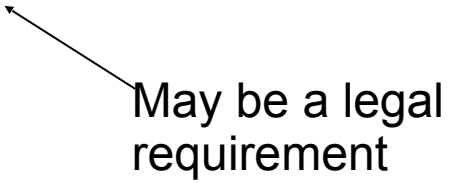
- We have one tool that can help us with all 3
  - Namely **glexec**
- glexec provides UID switching
  - But must be installed by the Site on WNs
- glexec requires a valid proxy from the final users on the submit node
  - Or jobs will not match
  - Not a problem for long time Grid users, like CMS, but it may be for other VOs

glexec in a picture



glexec, cont

- It is in VO best interest to use glexec
  - Only way to have a secure Multi User Pilot Job system
- Some sites require glexec for their own interest
  - To cryptographically know who the final users are
  - To easily ban users
- Note on site banning
  - Condor currently does not handle glexec failures well – VO admin must look for this



Turning on glexec

- Factory provides information about glexec installation at site
  - Attribute **`GLEXEC_BIN`** = `NONE | OSG | glite`
- Frontend decides if it should be used
  - Parameter **`GLIDEIN_Glexec_Use`**
  - Possible values:
    - NEVER - Do not use, even if available
    - OPTIONAL - Use whenever available
    - REQUIRED - Only use sites that provide glexec

Standard attributes

Standard attributes - Factory

- Factory provides
  - Max_walltime, optionally max_idle
  - Glexec location
  - Condor binaries (including os arch and version)
  - High level network setup (CCB, TCP, etc.)
- Anything not explicitly defined is provided by Factory defaults
  - glideinWMS/creation/web_base/condor_var*

Standard attributes - Frontend

- Frontend provides
  - Collector location
  - START expression
- Frontend should also provide
  - job_max_time
  - Should glexec be used

THE END

Pointers

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

Acknowledgments

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