

glideinWMS Training @ IU

Glidein startup Internals and Glidein configuration

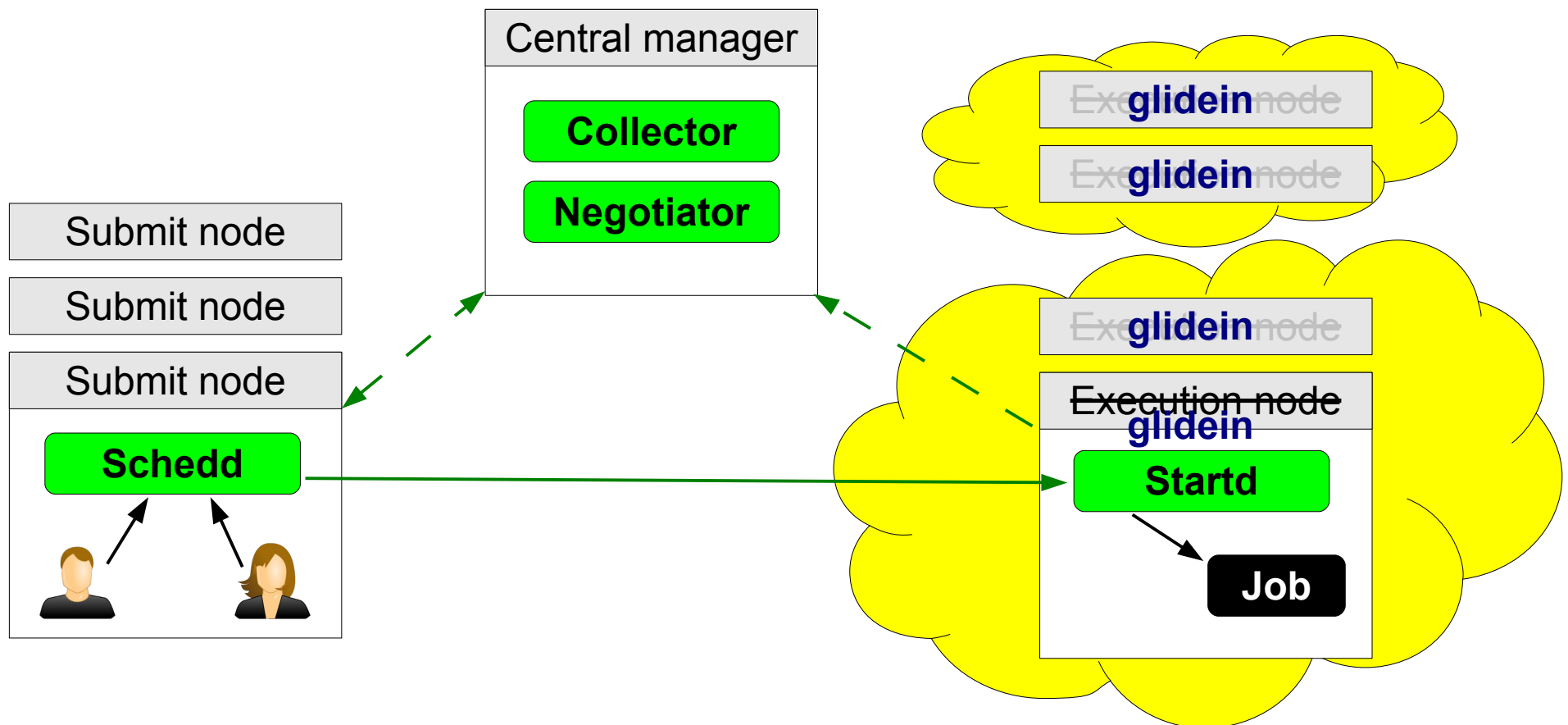
by Igor Sfiligoi (UCSD)

Outline

- Glidein_startup internals
- Glidein lifetime
- Security considerations

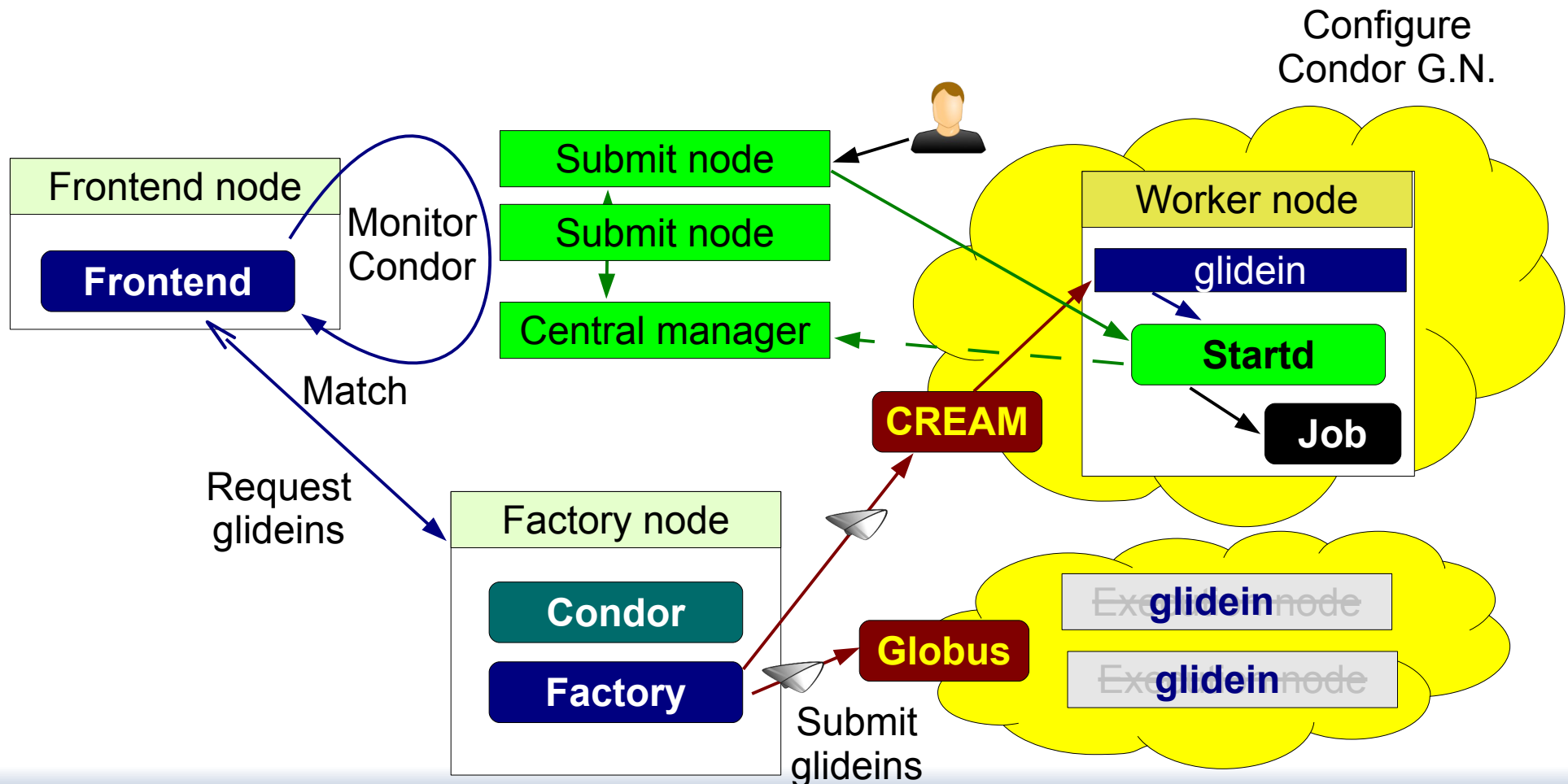
Refresher - What is a glidein?

- A glidein is just a properly configured execution node submitted as a Grid job



Refresher – Glidein startup

- glidein_startup configures and starts Condor



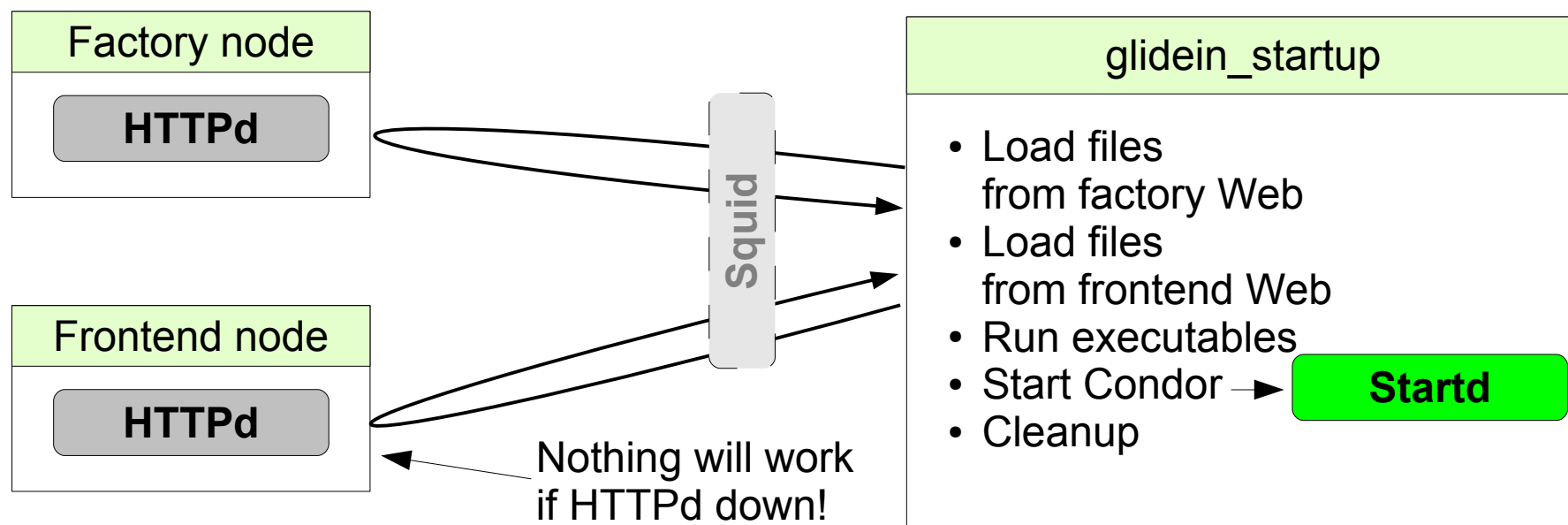
glidein_startup Internals

glidein_startup tasks

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

Downloading files

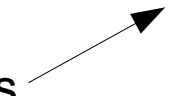
- Files downloaded via HTTP
 - From **both the factory and the frontend** Web servers
 - Can use local Web proxy (e.g. Squid)
 - Mechanism tamper proof and cache coherent



URLs

- All URLs passed to glidein_startup as arguments
 - Factory Web server
 - Frontend Web server
 - Squid, if any
- glidein_startup arguments defined by the factory
 - Frontend Web URL passed 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`
 - There is a (set of) logical to physical name map files
 - `*_file_list.id.lst`
 - The names of the list files are in
 - `description.id.cfg`
 - The id of this list is passed as an argument to `glidein_startup`
- Frontend ids
passed to the
factory via ClassAd
- 

Making the download tamper proof

- **Verifying SHA1 hashes**

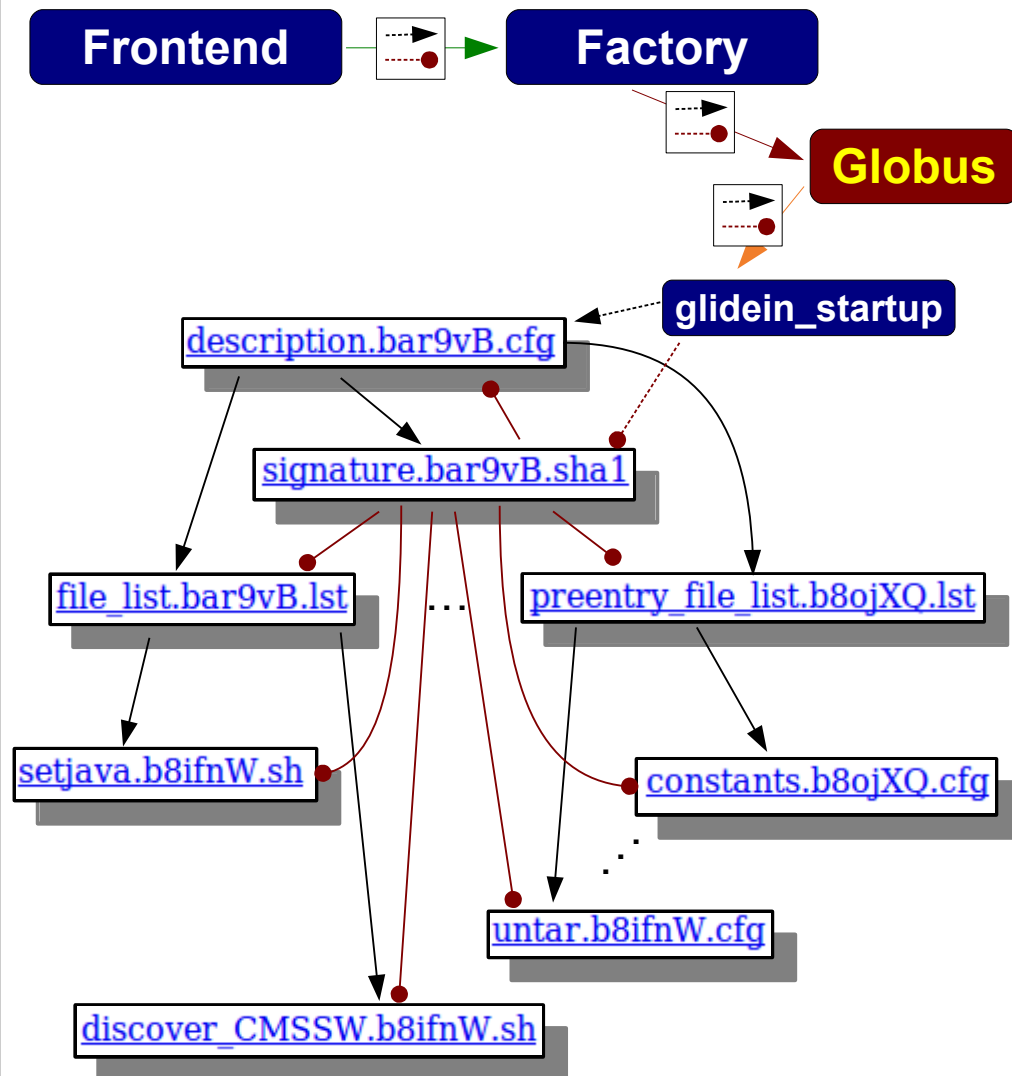
- Each file has a SHA1 associated with it
- The hashes delivered in a single file
 - `signature.id.sha1`
 - List of (fname,hash) pairs
- Signature file id in the description file

- The hash of the signature file a parameter of `glidein_startup`
 - This is guaranteed to be secure with GRAM
- Description file hash in the signature file
 - Still safe even if loaded out of order

Frontend hash
passed to
the factory
via ClassAd

Example Web area

? aftergroup_file_list.b8ifnW.lst	18-Aug-2011 15:23	181
? aftergroup_preentry_file_list.b8ifnW.lst	18-Aug-2011 15:23	190
? cat_consts.b8ifnW.sh	21-Nov-2011 16:04	1.1K
? check_blacklist.b8ifnW.sh	21-Nov-2011 16:04	1.4K
? condor_vars.b8ifGe.lst	18-Aug-2011 15:42	692
? condor_vars.b8ifnW.lst	18-Aug-2011 15:26	656
? condor_vars.b8ojXQ.lst	21-Nov-2011 16:04	692
? constants.b8ifGe.cfg	18-Aug-2011 15:42	83
? constants.b8ifnW.cfg	18-Aug-2011 15:26	61
? constants.b8ojXQ.cfg	21-Nov-2011 16:04	83
? description.b8ifGe.cfg	18-Aug-2011 15:42	274
? description.b8ifnW.cfg	18-Aug-2011 15:23	274
? description.b8ojXQ.cfg	24-Aug-2011 19:59	274
? description.bar9vB.cfg	27-Oct-2011 09:31	274
? discover_CMSSW.b8ifnW.sh	25-Oct-2011 16:18	1.9K
? file_list.b8ifnW.lst	18-Aug-2011 15:23	355
? file_list.bar9vB.lst	27-Oct-2011 09:31	292
? grid-mapfile.b8ifnW	21-Nov-2011 16:04	78
? group_itb/	18-Aug-2011 15:23	-
? nodes.blacklist	21-Nov-2011 16:04	269
? preentry_file_list.b8ifGe.lst	18-Aug-2011 15:42	617
? preentry_file_list.b8ifnW.lst	18-Aug-2011 15:23	617
? preentry_file_list.b8ojXQ.lst	24-Aug-2011 19:59	617
? set_home_cms.b8ifnW.source	21-Nov-2011 16:04	72
? setjava.b8ifnW.sh	21-Nov-2011 16:04	40
? signature.b8ifGe.sha1	18-Aug-2011 15:42	965
? signature.b8ifnW.sha1	18-Aug-2011 15:23	965
? signature.b8ojXQ.sha1	24-Aug-2011 19:59	965
? signature.bar9vB.sha1	27-Oct-2011 09:31	898
? untar.b8ifnW.cfg	21-Nov-2011 16:04	27



Node validation

- Run scripts / plugins provided by both factory and frontend
 - The 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 never started → no user jobs ever pulled
 - Will sleep for 20 mins → blackhole protection

Condor configuration

- Condor config values coming from files downloaded by glidein_startup
 - Static from both factory and frontend config files
 - More details at http://tinyurl.com/glideinWMS/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://tinyurl.com/glideinWMS/doc.prd/factory/custom_scripts.html

Example (pseudo-)script

```
# check if CMSSW installed locally
if [ -f "$CMSSW" ]; then
    # get the other env
    source "$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://tinyurl.com/glideinWMS/doc.prd/factory/custom_scripts.html

Example Frontend attr

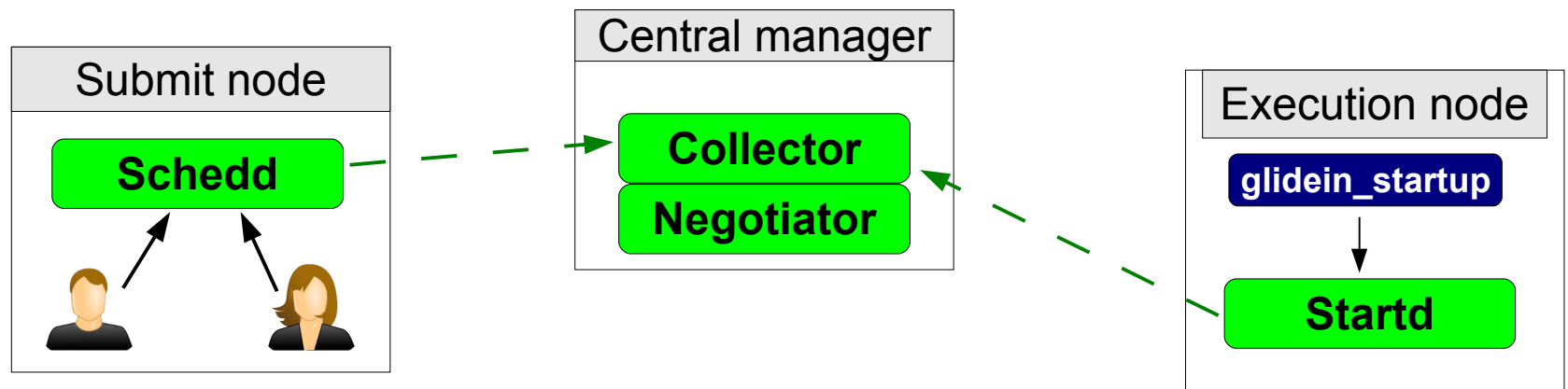
- START expression
 - As GLIDECLIENT_Group_Start
 - Automatically **&&** with Factory provided one

```
GLIDECLIENT_Group_Start =  
  (stringListMember (GLIDEIN_SEs,  
                     DESIRED_SEs)=?=True) ||  
  (stringListMember (GLIDEIN_Site,  
                     DESIRED_Sites)=?=True) )
```

More details at http://tinyurl.com/glideinWMS/doc.prd/factory/custom_vars.html

Condor startup

- After validation and configuration, glidein just start `condor_master`
 - Which in turn start `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
- Frontend does not have access to them
 - Work in progress for future glideinWMS release
 - For now, ask Factory admins if debugging

Cleanup

- glidein_startup will remove all files before terminating
 - Unless killed by the OS, of course
 - Condor does something similar for the user jobs 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 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
 - Resulting in waste
- Deadline site specific
 - Set by the Factory
 - 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 wasteful
 - Glideins set an earlier deadline for job startup
 - **After the deadline, Condor will terminate at job end**
 - Advertised as **GLIDEIN_TORETIRE**
- Need to know how long is the expected job lifetime
 - Should be **provided by the Frontend**
 - Parameter **GLIDEIN_Job_Max_Time**
 - $\text{ToRetire} = \text{Max_Walltime} - \text{Job_Max_Time}$

Default exists but
don't count on it



Termination due to non-use

- **glideins will self-terminate if not used**, too
 - To limit waste
 - Reasons for hitting this
 - No more user jobs (spikes)
 - Policy problems (Frontend vs Negotiator)
 - Typically uniform 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

Finer grained policies

- Job_Max_Time is the typical expected job lifetime
 - What if not all jobs have the same lifetime?
- Use Condor matchmaking to define finer grained policies
 - To prevent long jobs from starting if they are expected to be killed before terminating
 - This allows (lower priority) shorter jobs to use the CPU
 - Or waste just 20 mins (instead of hours)
- Job lifetime can be difficult to know
 - Even users often don't know
 - Can refine after restarts (assuming they are deterministic)

Example START expression

- Have two different thresholds
 - One for first startup, one for all following
 - Useful when wide dynamic range, unpredictable

```
GLIDECLIENT_Start =  
  ifthenelse(  
    LastVacateTime=?=UNDEFINED,  
    NormMaxWallTime < (GLIDEIN_ToDie-MyCurrentTime),  
    MaxWallTime < (GLIDEIN_ToDie-MyCurrentTime)  
  )
```

Other sources of waste

- **Glideins must stay within the limits** of the resource lease
 - Typical limit is on memory usage
OSG Factory defines **GLIDEIN_MaxMemMBs**
- If user jobs exceed that limit, the glideins may be **killed by the resource provider** (e.g. Grid batch system)
 - Resulting in the user job being killed, too
 - Thus waste

Example START expression

- Prevent startup of jobs that are known to use too much memory
 - Unless user defines it, known only at 2nd re-run
- Also kill jobs as soon as they pass that
 - So we don't have to start a new glidein for other jobs

```
GLIDECLIENT_Start =  
  ImageSize <= (GLIDEIN_MaxMemMBs * 1024)
```

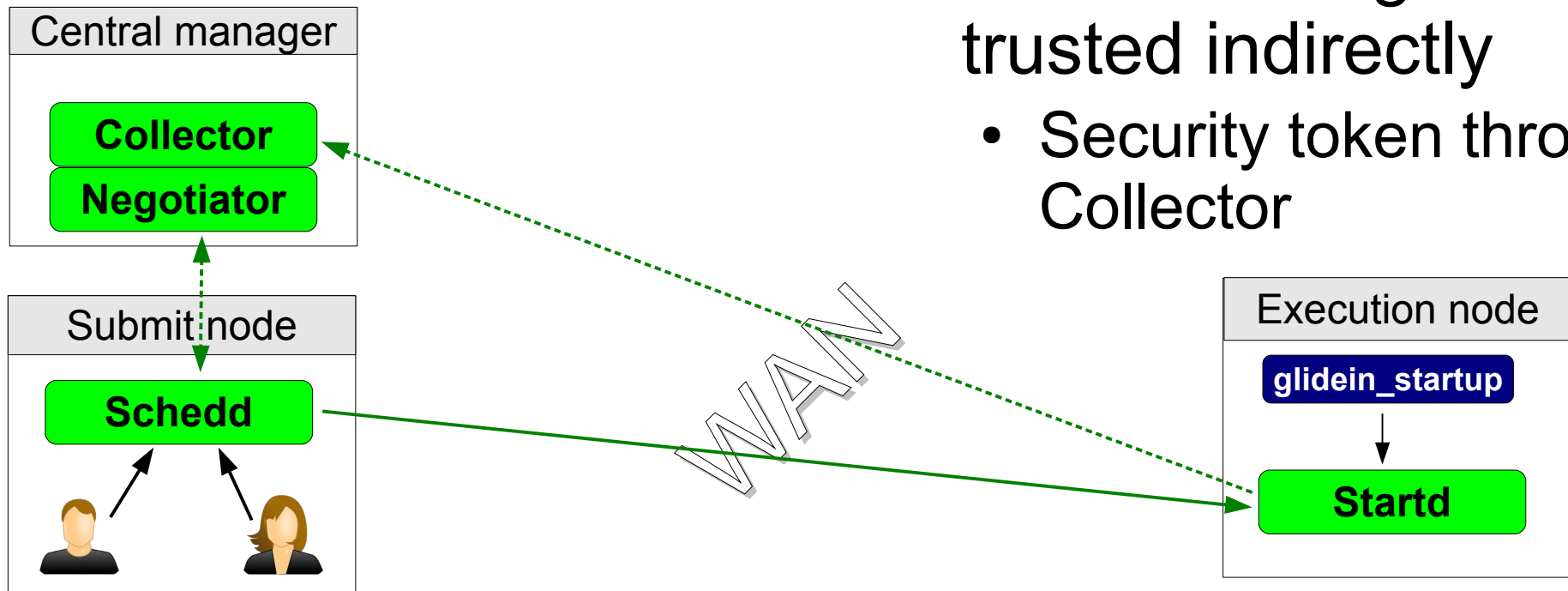
```
PREEMPT =  
  ImageSize > (GLIDEIN_MaxMemMBs * 1024)
```

More details at http://research.cs.wisc.edu/condor/manual/v7.6/10_Appendix_A.html#82447

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 trusted indirectly
 - Security token through Collector



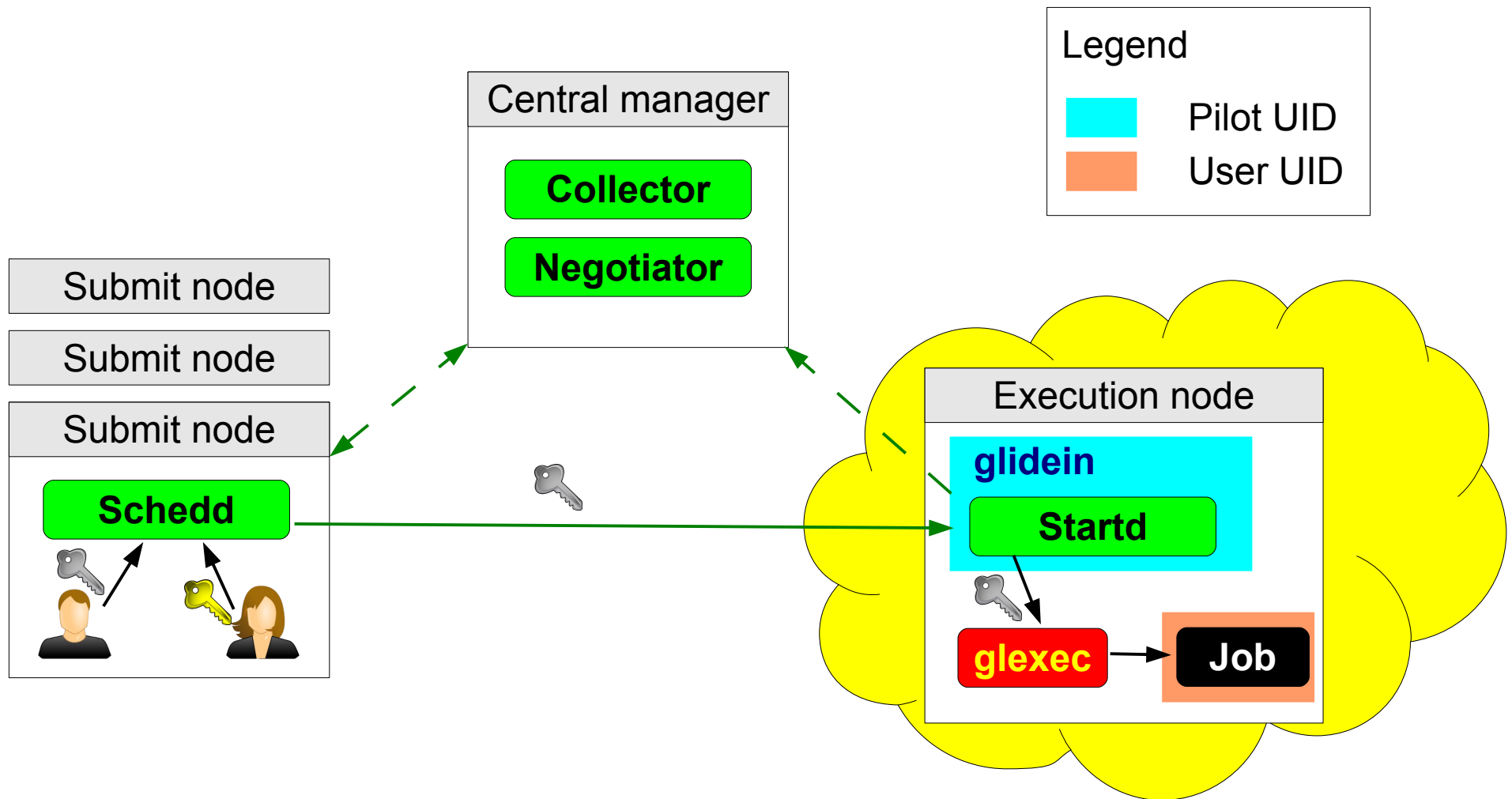
Multi User Pilot Jobs

- A glidein typically starts with a proxy that is not representing a specific user
 - Very few exceptions (e.g. CMS Organized Reco)
- 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
 - Only a subset of sites have done it so far
- **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

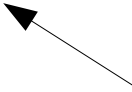
glexec in a picture



glexec, cont

- It is in VO best interest to use glexec
 - Only way to have a secure MUPJ system
 - Should push sites to deploy it
- 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 well glexec failures – VO admin must look for this

May be a legal
requirement



Turning on glEXEC

- Factory provides information about glEXEC installation at site
 - Attribute **GLEEXEC_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

THE END

Pointers

- The official project Web page is <http://tinyurl.com/glideinWMS>
- glideinWMS development team is reachable at glideinwms-support@fnal.gov
- CMS AnaOps Frontend at UCSD
http://glidein-collector.t2.ucsd.edu:8319/vofrontend/monitor/frontend_UCSD-v5_2/frontendStatus.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