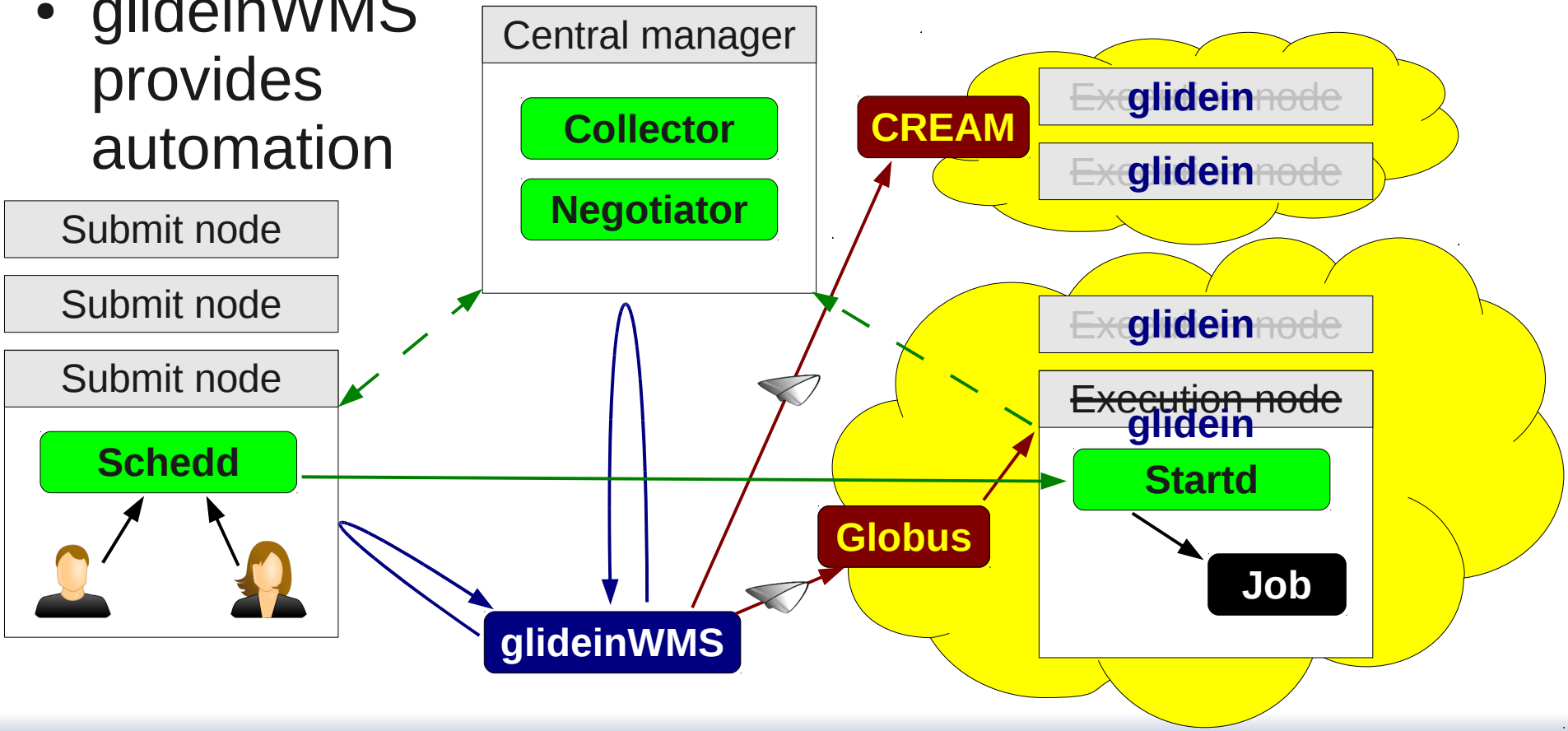
glideinWMS frontend Internals

by Igor Sfiligoi (UCSD)

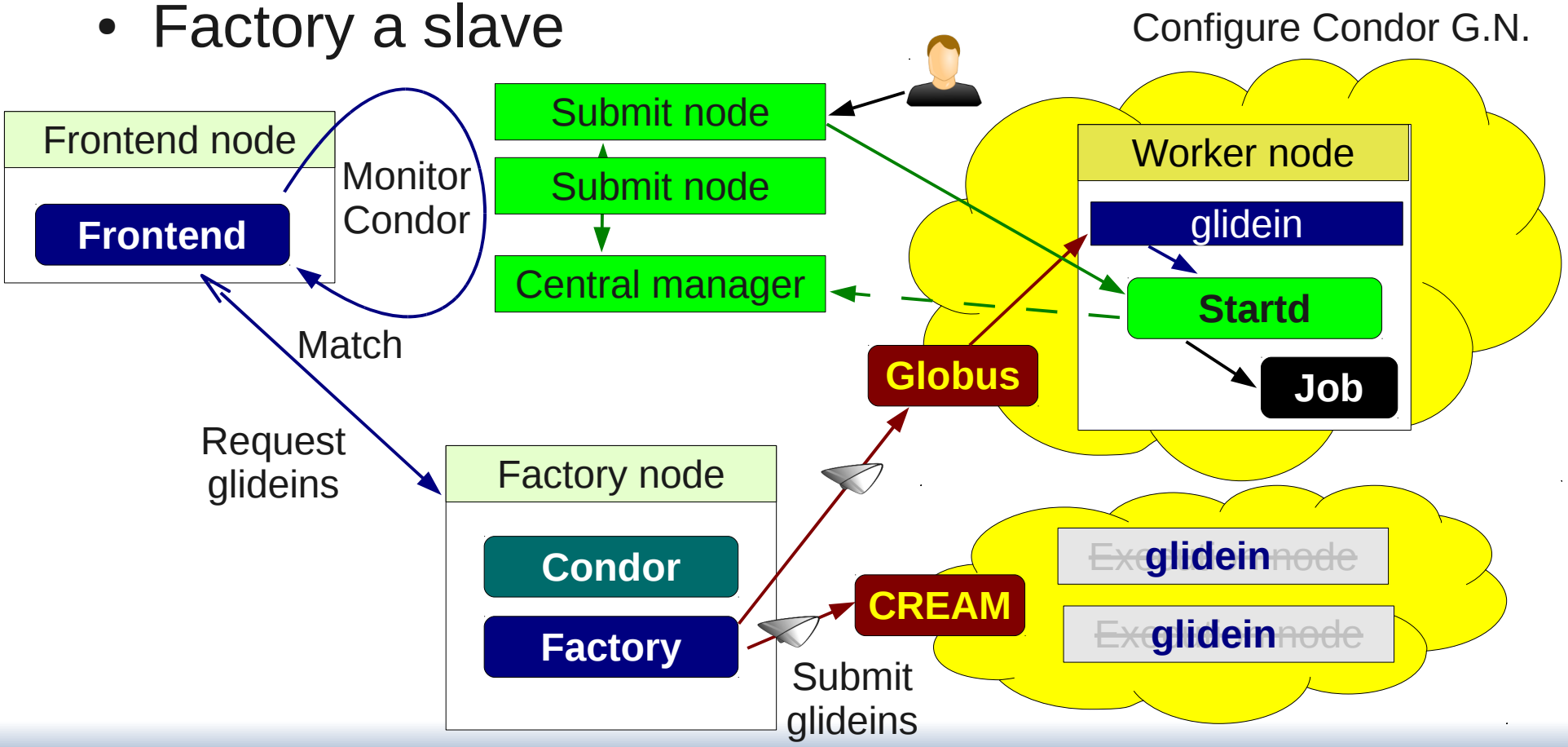
Refresher - Glideins

- A glidein is just a properly configured Condor execution node submitted as a Grid job
  - glideinWMS provides automation



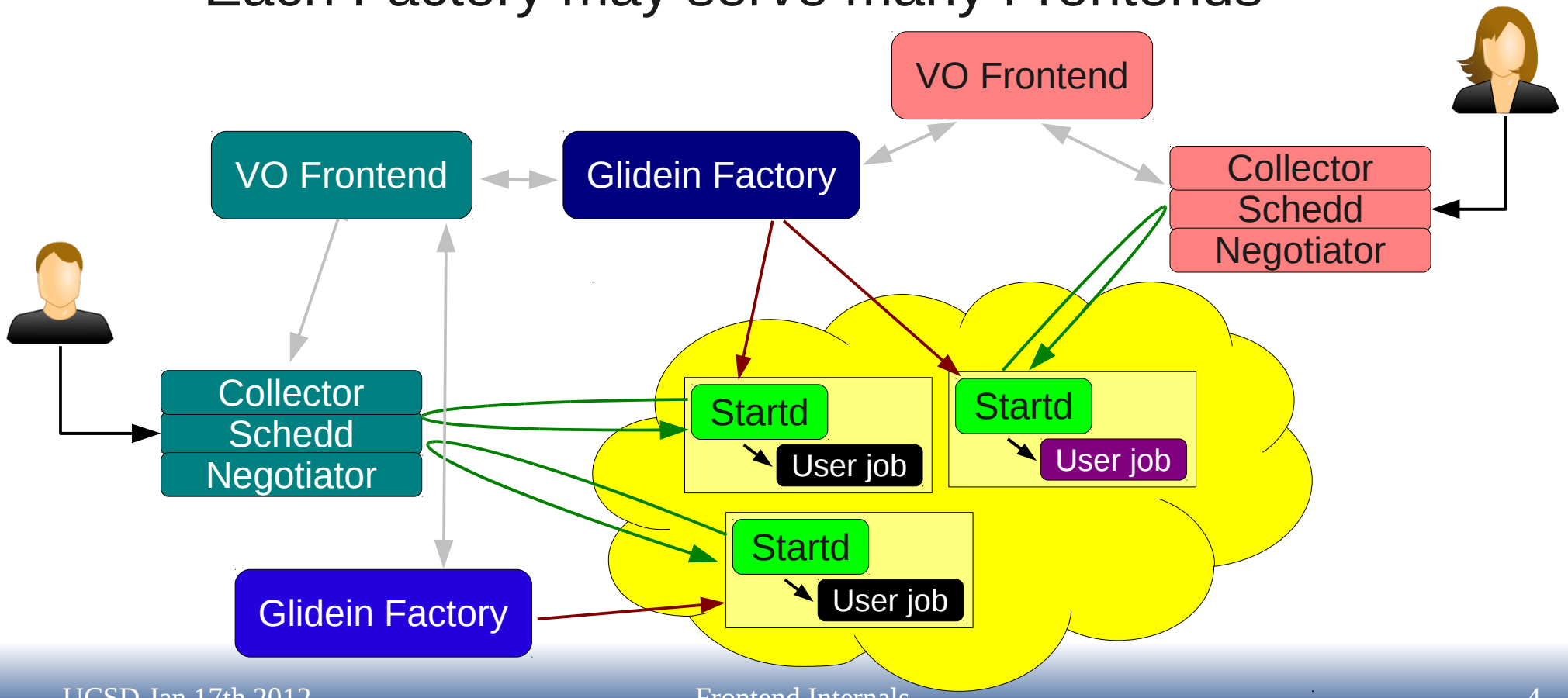
Refresher – Glidein Frontend

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



Refresher - Cardinality

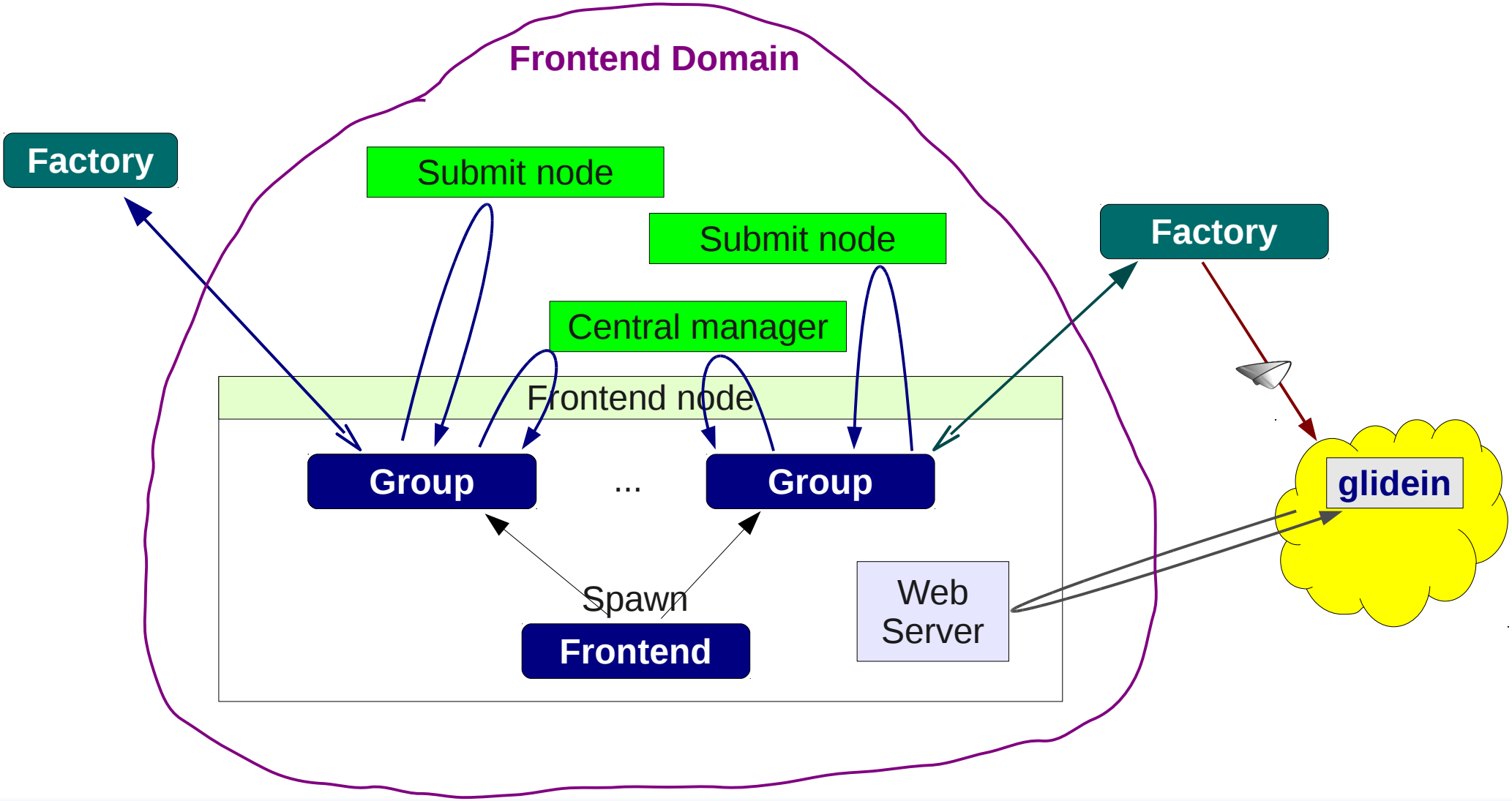
- N-to-M relationship
  - Each Frontend can talk to many Factories
  - Each Factory may serve many Frontends



Frontend architecture

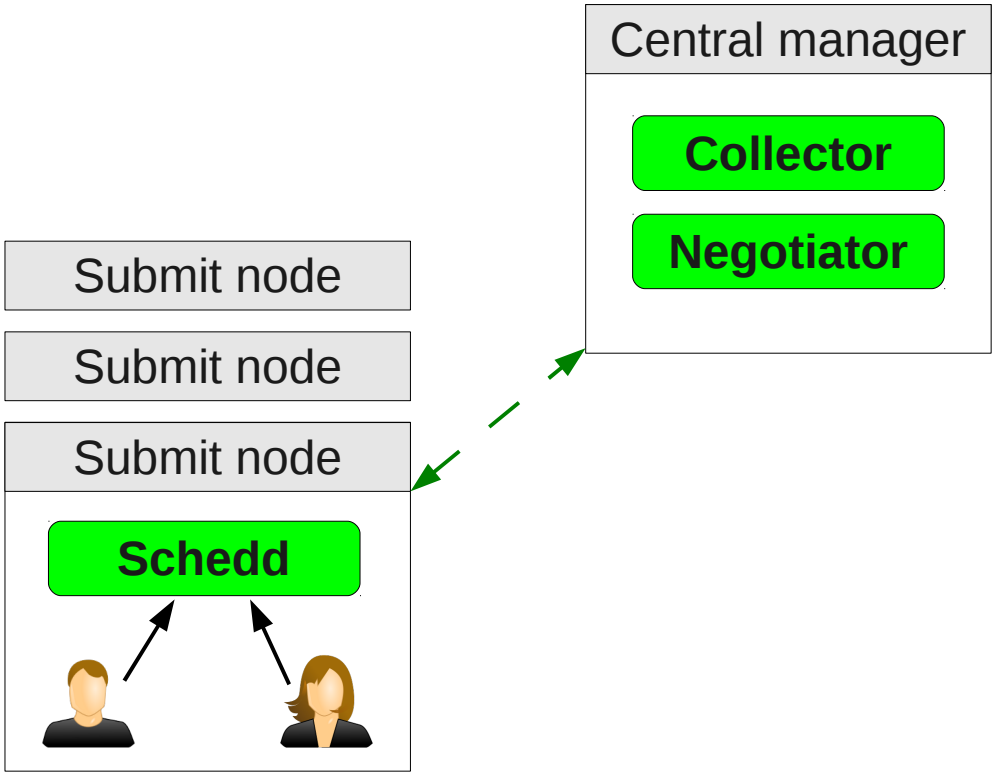
- The frontend is composed of:
  - The Condor daemons
  - The glideinWMS frontend proper
  - Condor client – to talk to the factories
  - Web server – deliver code and data to glideins
    + monitoring
- The glideinWMS frontend itself composed of:
  - Group processes – do the real work
  - Master frontend – controls the others and
    aggregates monitoring

Frontend arch - Picture



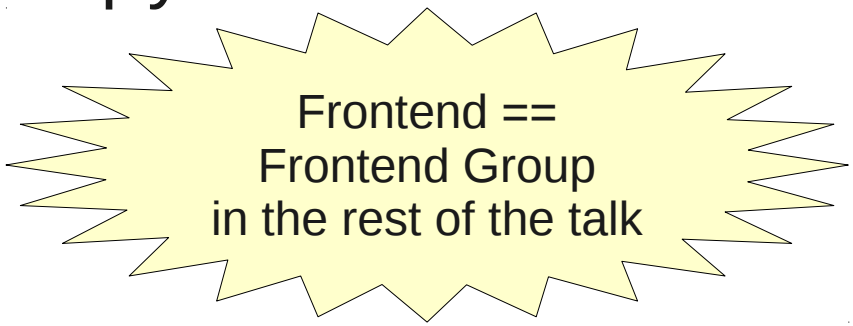
Condor processes

- Explained in enough detail in previous talk
  - Will not repeat myself



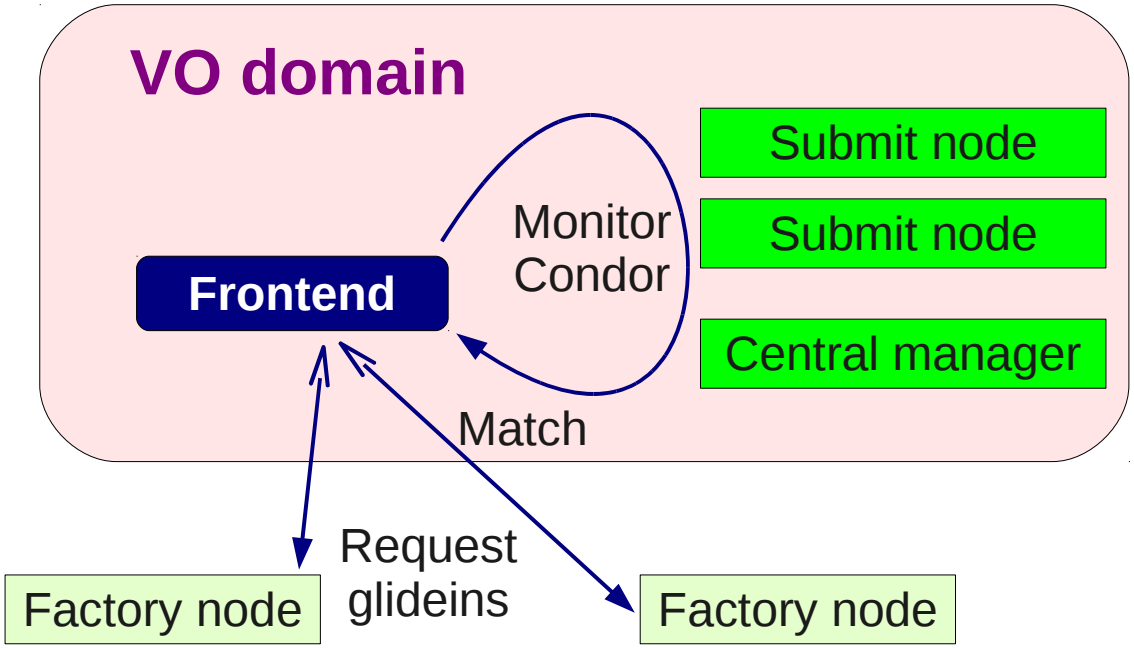
Frontend processes

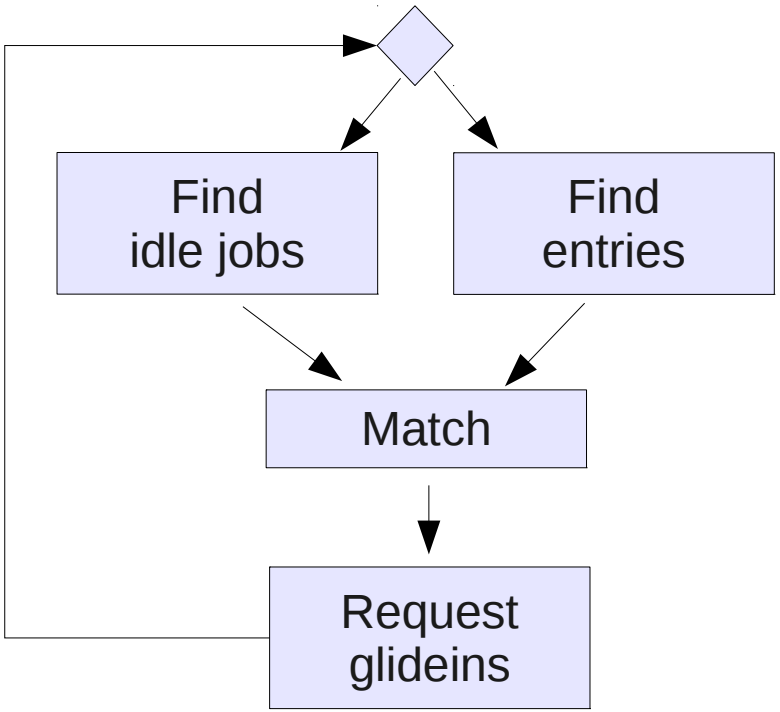
- Real work performed by Group process
  - `glideinFrontendElement.py`
  - One process x Group
- They are controlled by master Frontend
  - `glideinFrontend.py`
  - Starts the other processes
  - Aggregates monitoring



Frontend role

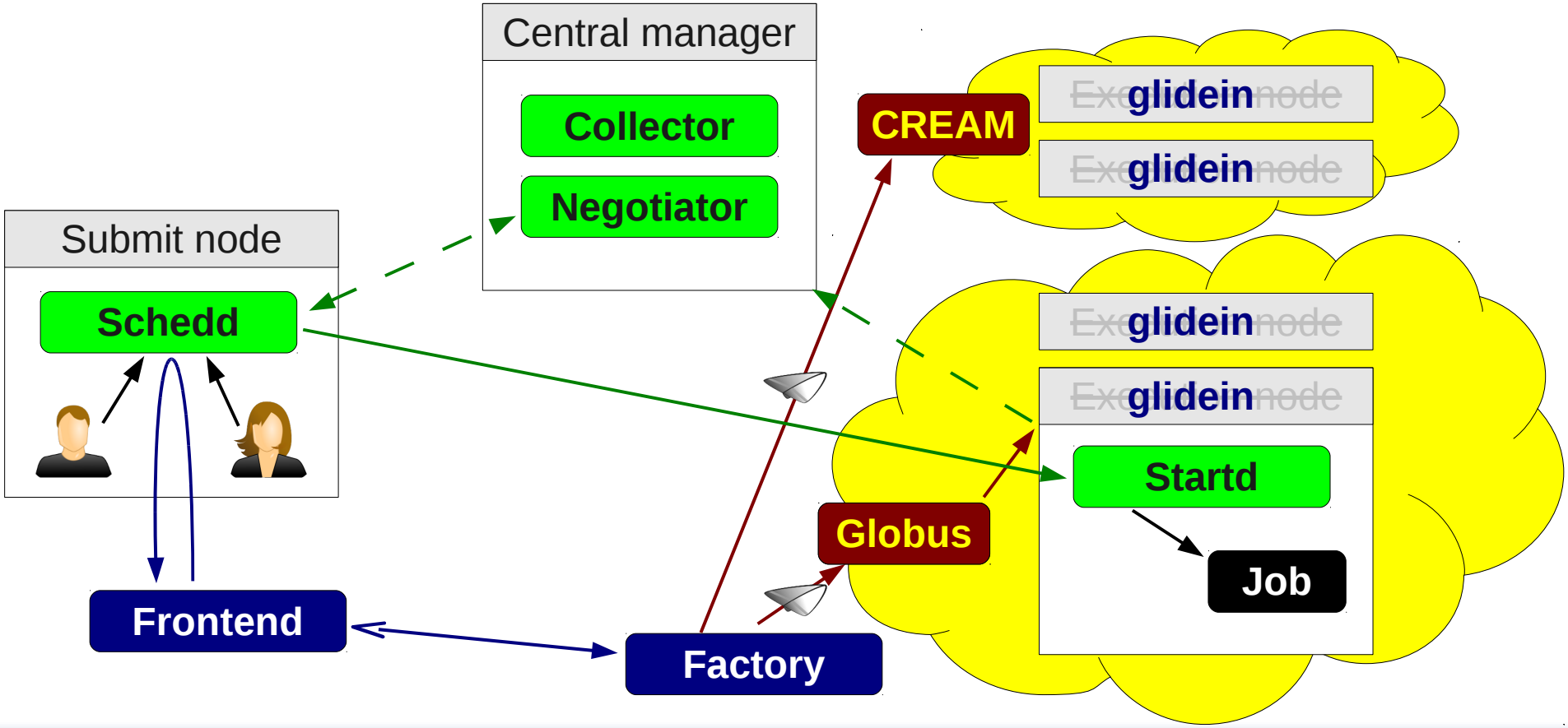
- The VO frontend is the brain of a glideinWMS-based pool
  - Like a site-level “negotiator”





Reminder - Two level matchmaking

- The frontend triggers glidein submission
  - The “regular” negotiator matches jobs to glideins



Matchmaking logic

- The Frontend matchmaking policy is implemented centrally
  - By the VO admin – not by the users
- It can use the attributes from both the job and Factory ClassAds
- Should be kept in sync with Negotiator policy
  - Which is not centralized
  - One way to define in the glidein START expression
  - Unfortunately, one python expression other ClassAds

Example matchmaking logic

- Frontend

```
job.has_key("DESIRED_Sites") &&
glidein["attrs"].get("GLIDEIN_Site")
  in job["DESIRED_Sites"].split(",")
```

- Negotiator (via glidein START)

```
GLIDECLIENT_Start =
  stringListMember(GLIDEIN_Site,
                   DESIRED_Sites,",")=?=True
```

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

Communication Protocol

- No listen sockets
  - All communication one way (Frontend->Factory)
- Each Factory provides a Collector
  - Communication based on ClassAds
  - All security implemented in the Collector
- Use standard cmdline tools for communication
  - condor_status and condor_advertise

Protocol sequence

- Polling loop
  - Read Factory ClassAds from all factory Collectors
  - Match against jobs
  - Advertise own existence and requests
- Frontend sends 4 types of info
  - Own identity
  - Glidein submission regulation instructions
  - Glidein parameters
  - Pilot Proxy

Glidein submission regulation

- The glideinWMS glidein request logic is based on the principle on "constant pressure"
  - Frontend Group requests a certain number of "idle glideins" in the factory queue at all times
  - It **does not** request a specific number of glideins
- This is done due to the asynchronous nature of the system
  - Both the factory entries and the frontend groups are in a polling loop and talk to each other indirectly

Glidein requests

- Frontend matches job attrs against entry attrs
  - It then counts the matched idle jobs
  - A fraction of this number becomes the "pressure requests" (up to 1/3)
  - This number is then capped (~20)
  - The attribute in the ClassAd is **`ReqIdleGlideins`**
- The Frontend also advertises **`ReqMaxRunningGlideins`**
  - Emergency break

Scaling back

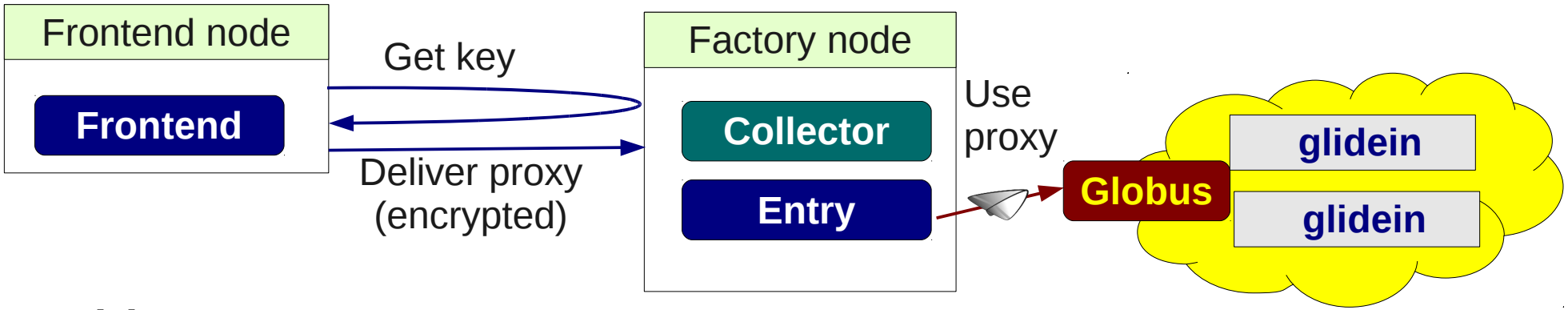
- The Frontend can also request that existing glideins in the Factory queues are removed **`ReqRemoveExcess`**
  - NO – Default, never remove
  - WAIT – Remove any glidein not yet at a site
  - IDLE – Remove any glidein that has not started yet
  - ALL – Remove all glideins
- Frontend pretty conservative
  - Only requests removal if no user jobs in the queues

Parameters

- Frontend can send attributes to glideins:
  - Dynamically – as parameter in the ClassAd
  - Statically – as entry in a config file
- Attributes typically static
  - Current Frontend implementation does not really have much support for dynamicity

Pilot proxy delegation

- Pilot proxy is encrypted with factory pub key
  - Then published in the ClassAd
  - Only owner of priv. key can decrypt it



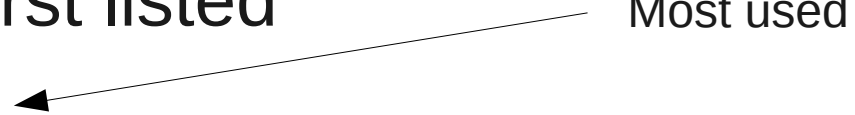
- However
  - Must make sure we are talking to a trusted Factory!
    - not just anyone providing a pub key
  - More details in a few slides

Pilot proxy selection

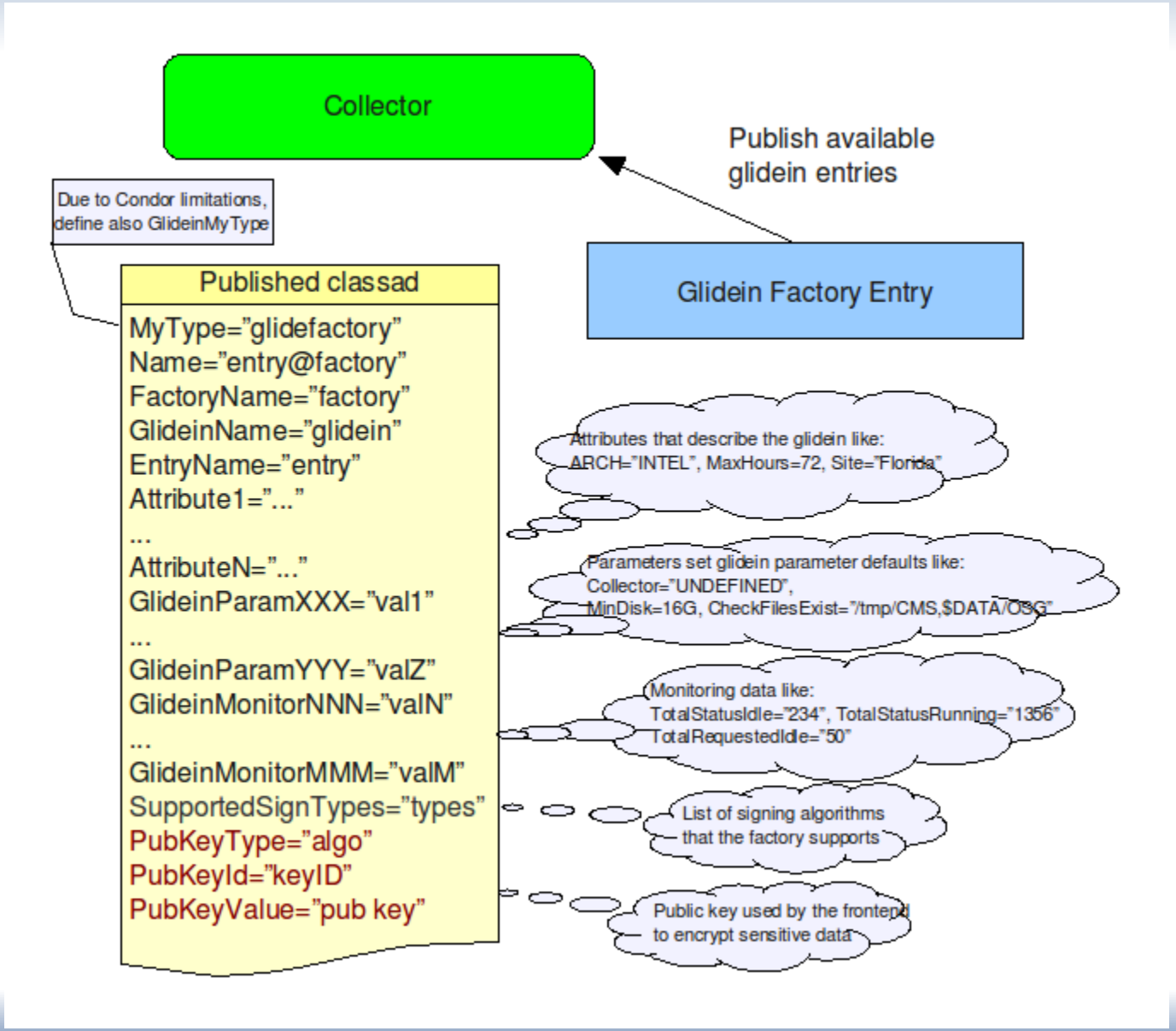
- A Frontend must have at least one pilot proxy
  - But can have more than one
- Many proxies can be used for priority reasons
  - When competing with non-pilot submission
  - Want to have as many proxies as users served
- Proxy selection plugin based

Pilot proxy plugins

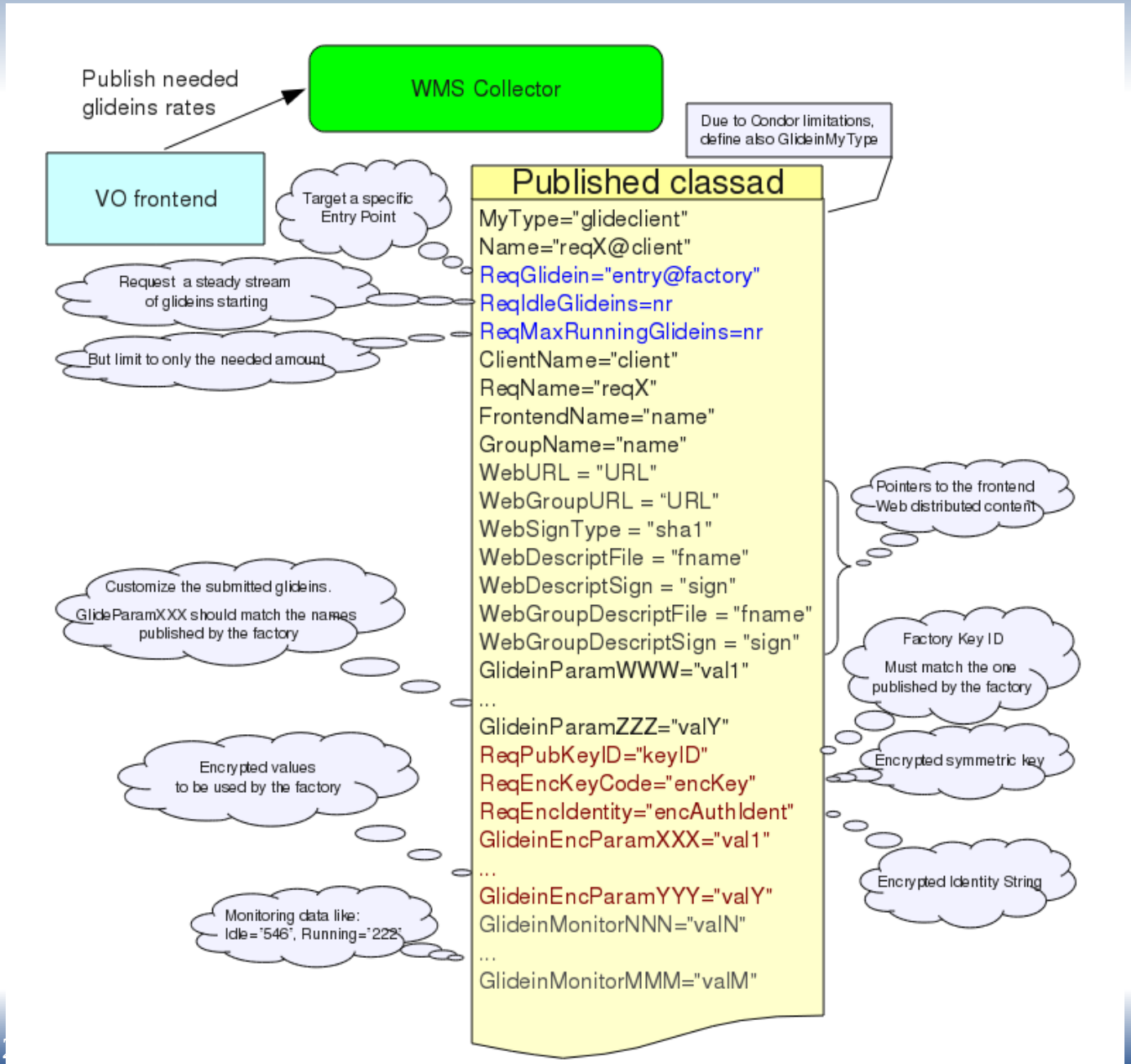
- Several standard plugins
  - ProxyFirst – Only the first listed
  - **ProxyAll – All listed**
  - ProxyUserCardinality – First N, with N=#users
  - ProxyUserMapWRecycling – N, with pilot-to-user mapping
- VO admin could implemented his own, if desired



Factory ClassAd

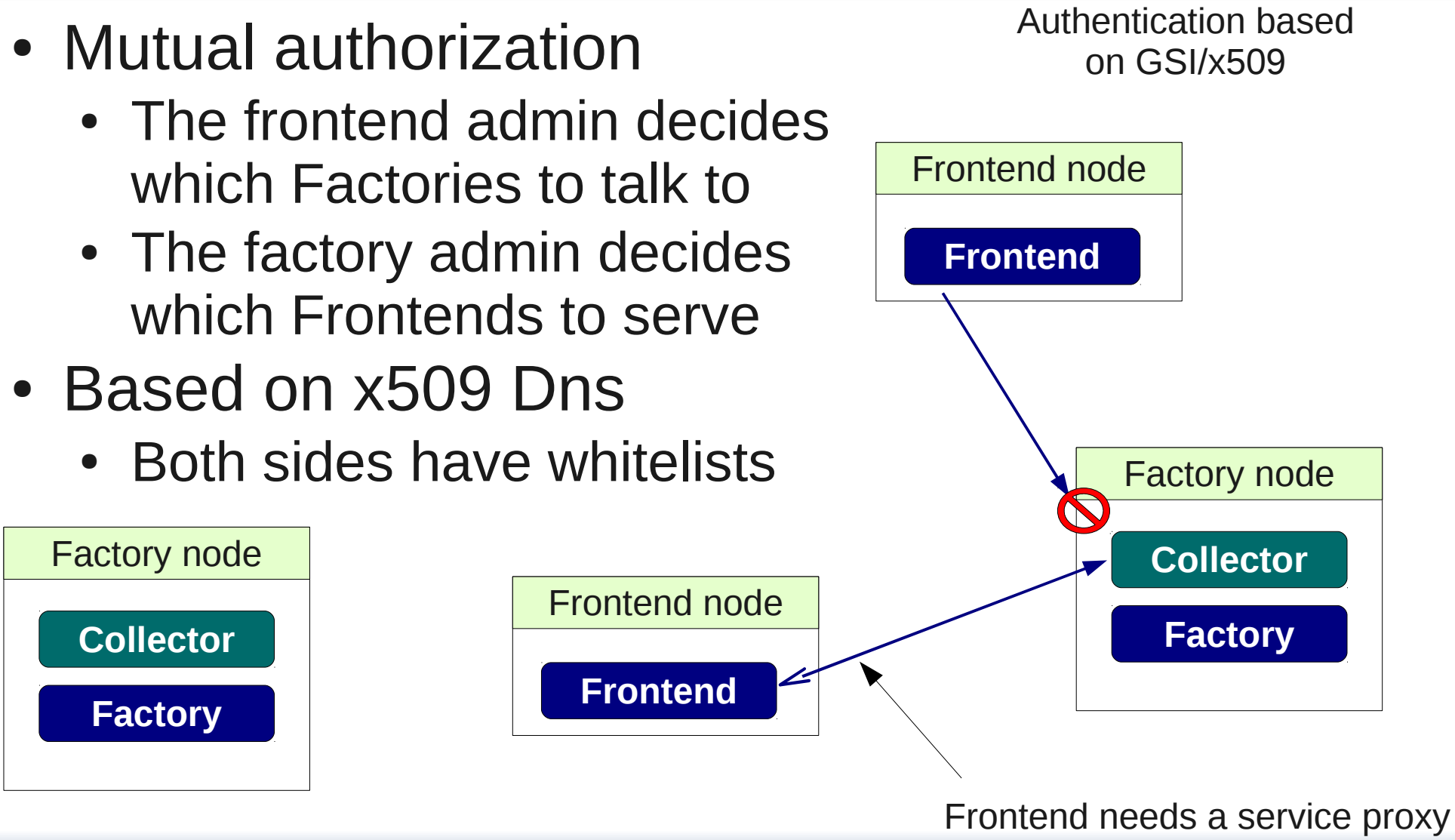


Frontend ClassAd



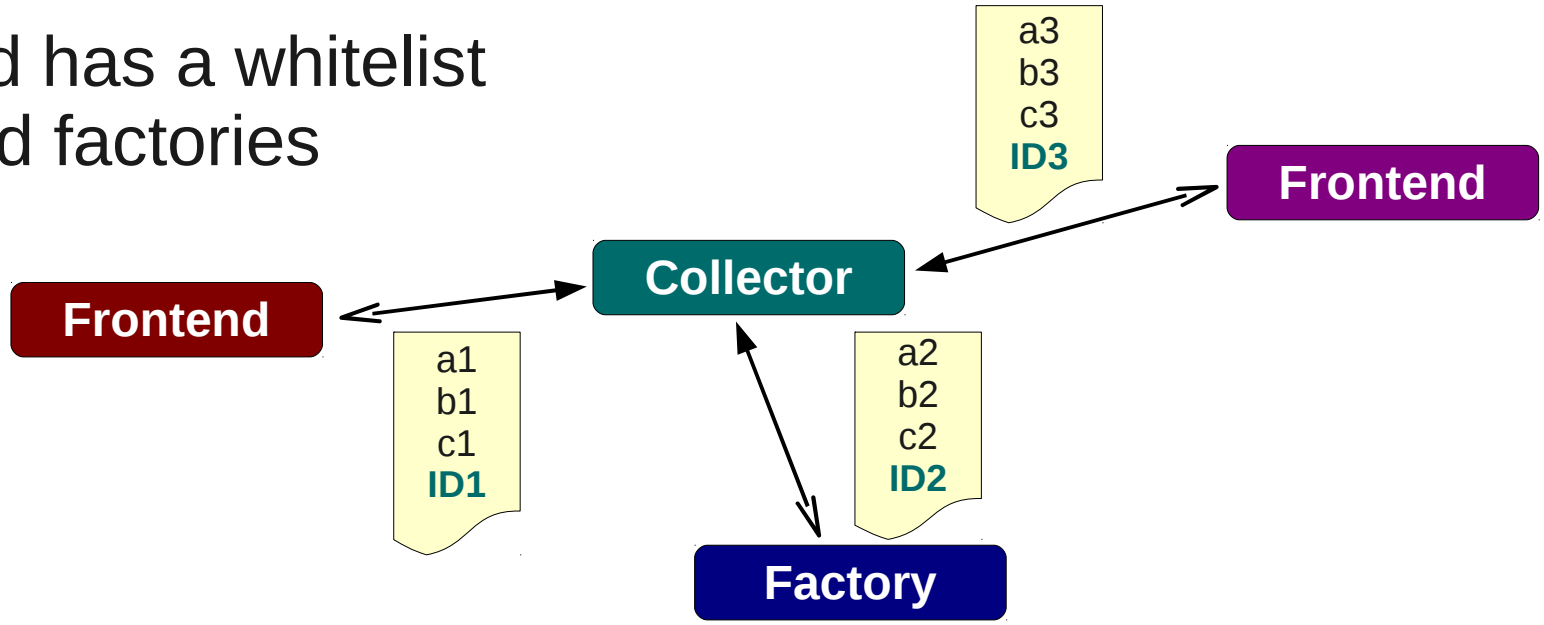
Security - Authorization

- Mutual authorization
  - The frontend admin decides which Factories to talk to
  - The factory admin decides which Frontends to serve
- Based on x509 Dns
  - Both sides have whitelists



Trusting the factory key

- It is all just ClassAds!
  - Anyone can publish a ClassAd and declare to be a factory
- However, Factory Collector knows who published it
  - And advertises it as the attribute **`AuthenticatedIdentity`**
  - Cannot be faked by the client
- Frontend has a whitelist of trusted factories



Security handles

- As we said, mutual authentication with Factory
- Frontend provides (and Factory whitelists)
  - Service Proxy to talk to Factory Collector
  - Frontend Security name

  } One set for whole Frontend (all Groups)

  - Proxy Security Class

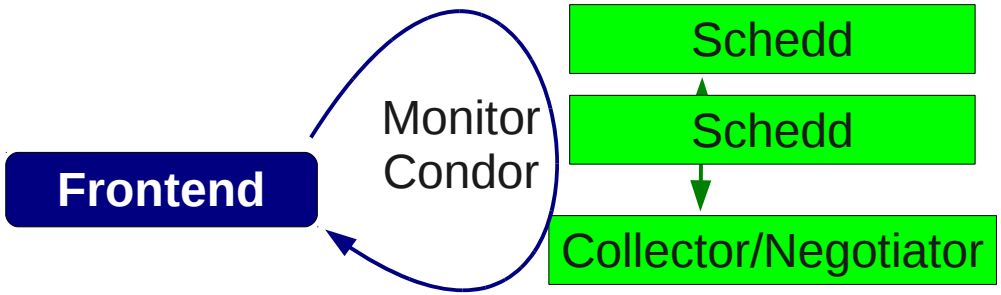
  } One per pilot proxy
- Frontend whitelists (obtained from Factory admins)
  - Factory Collector DN
  - Own mapping @Factory
  - Factory mapping @Factory

  } One set per factory collector

Security within the VO domain

- Frontend process, Collector and schedds often not on the same node
  - Need network security
- **All processes must whitelist each other**
  - Again, GSI based

Could be even over WAN
CMS setup has nodes in CA, IL and Europe



THE END

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