glideinWMS architecture

by Igor Sfiligoi (UCSD)

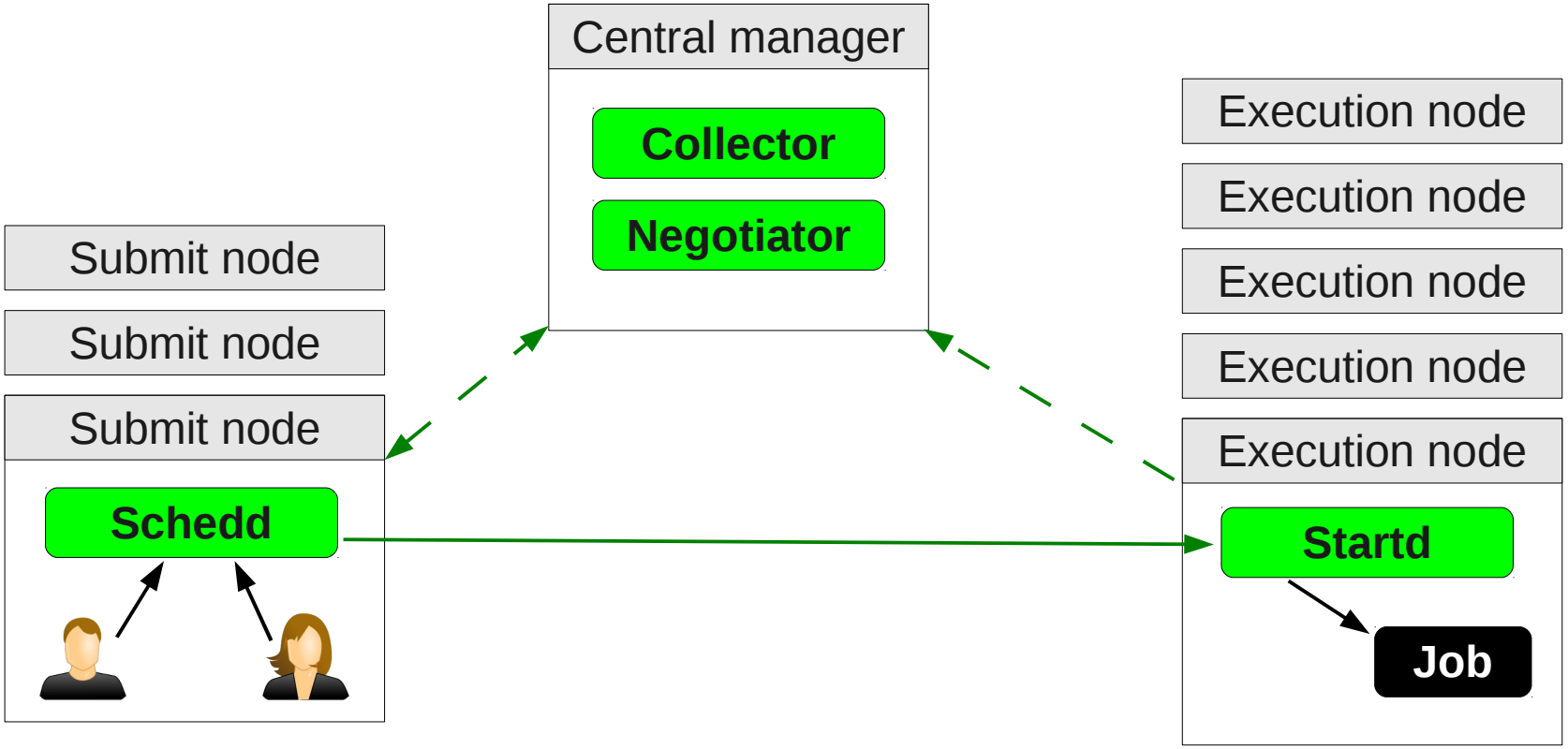
Outline

- A high level overview of the glideinWMS
- Description of the components

glideinWMS from 10km (30k feet)

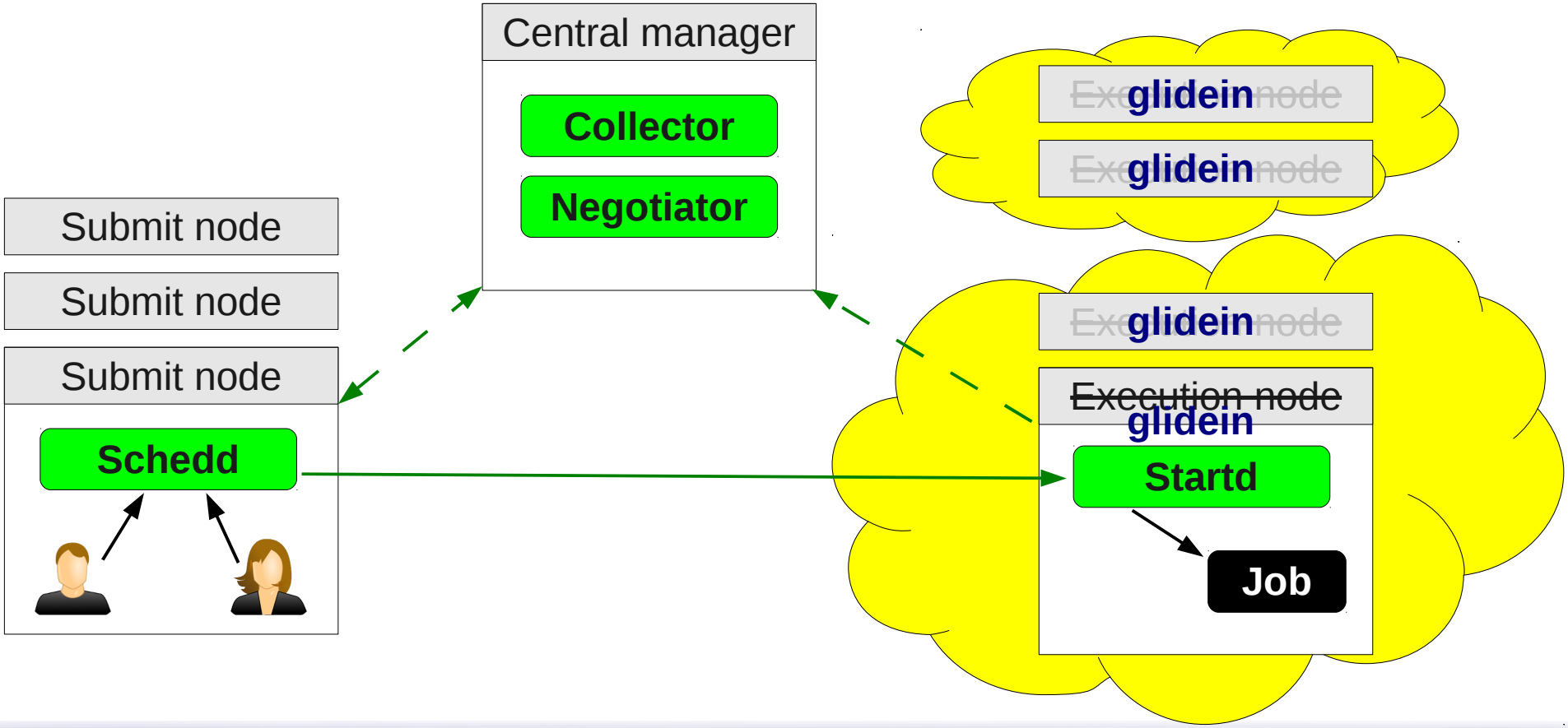
Condor

- A Condor pool is composed of 3 pieces



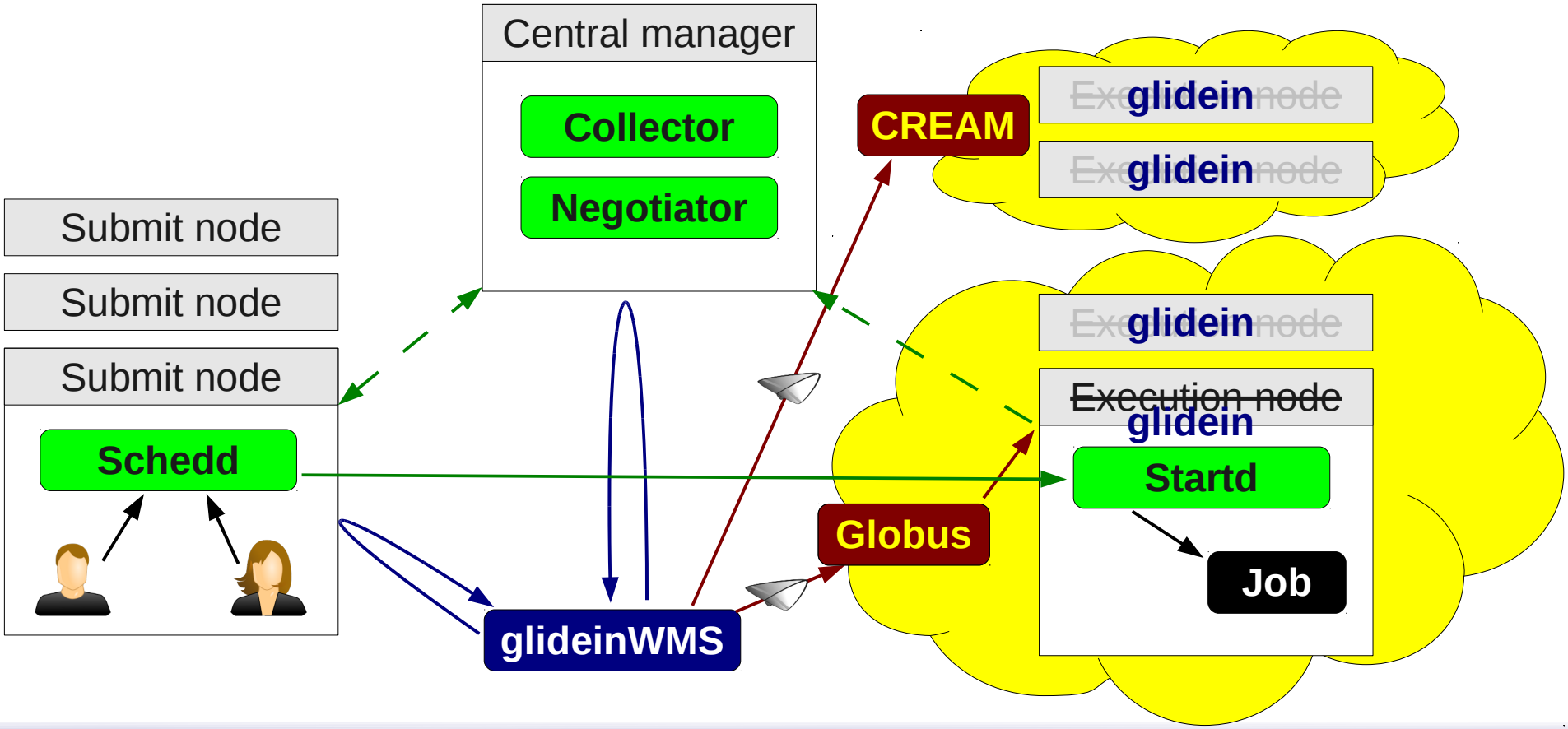
What is a glidein?

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



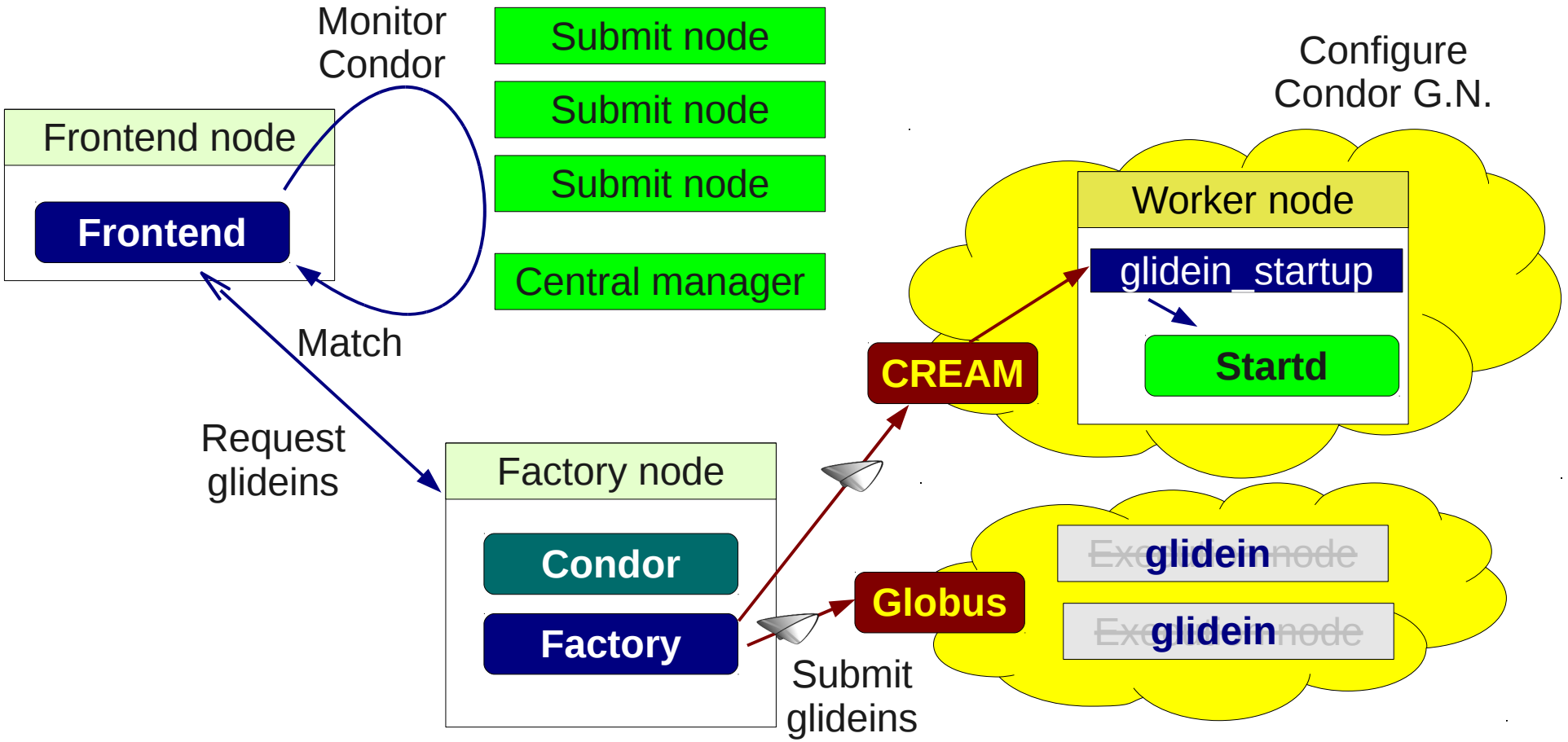
What is glideinWMS?

- glideinWMS is an automated tool for submitting glideins on demand



glideinWMS architecture

- glideinWMS has 3 logical pieces



glideinWMS architecture

- glideinWMS has 3 logical pieces
  - glidein_startup – Configures and starts Condor execution daemons
  - Factory – Knows about the sites and does the submission
  - Frontend – Knows about user jobs and requests glideins

Runtime environment discovery and validation

Grid knowledge and troubleshooting

Site selection logic and job monitoring

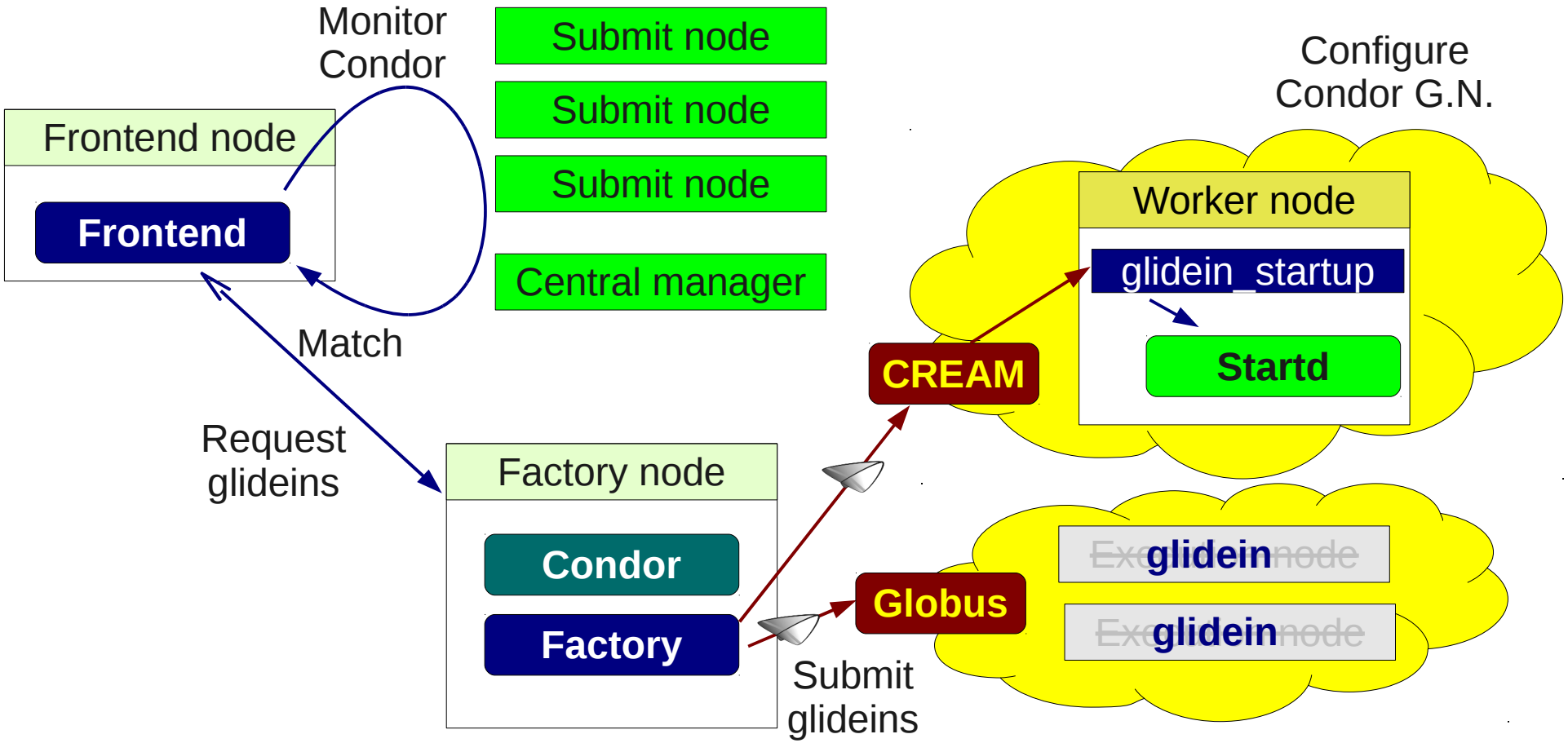
glideinWMS

A (sort of) detailed view of

glidein_startup

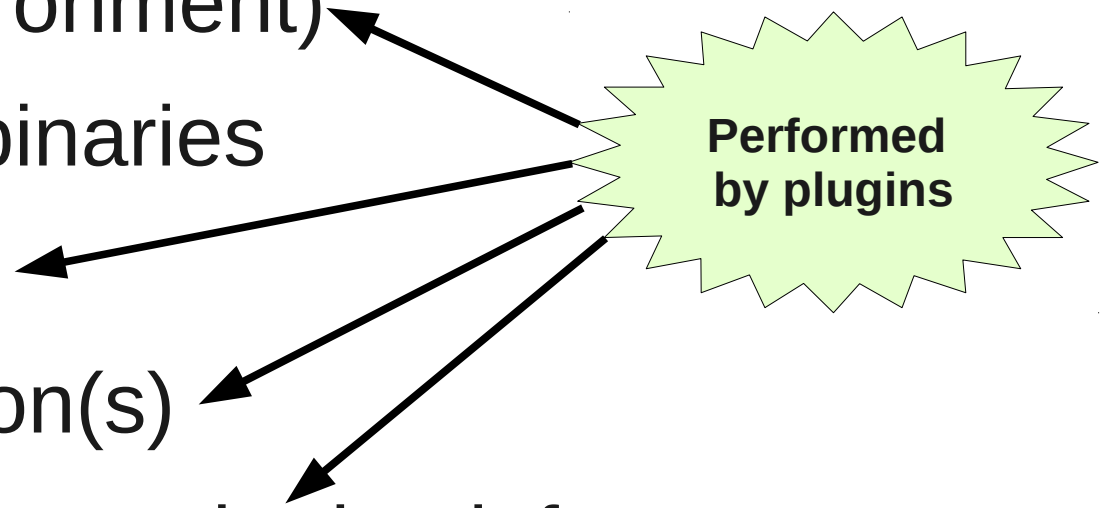
Refresher – glideinWMS arch.

- glidein_startup configures and starts Condor



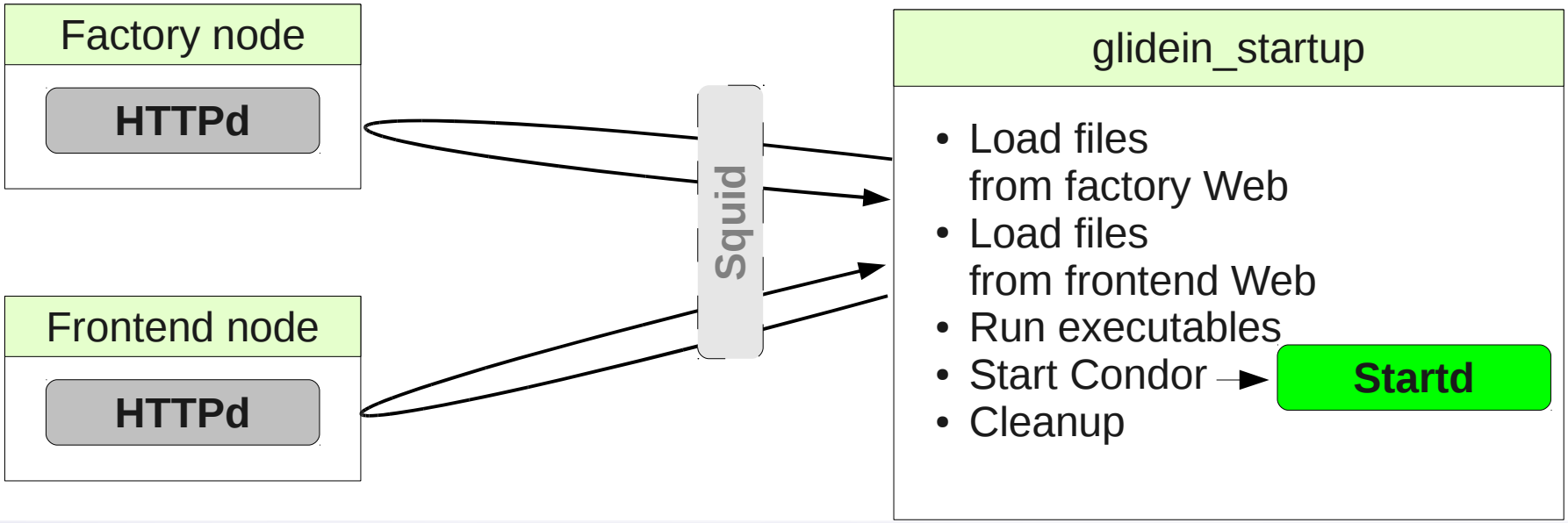
glidein_startup tasks

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



glidein_startup plugins

- Config files and scripts loaded 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



glidein_startup scripts

- Standard plugins
  - Basic Grid node validation (certs, disk space, etc.)
  - Setup Condor (glexec, CCB, etc.)
- VO provided plugins
  - Optional, but can be anything
  - CMS@UCSD checks for CMS SW
- Factory admin can also provide them
- Details about the plugins can be found at http://tinyurl.com/glideinWMS/doc.prd/factory/custom_scripts.html

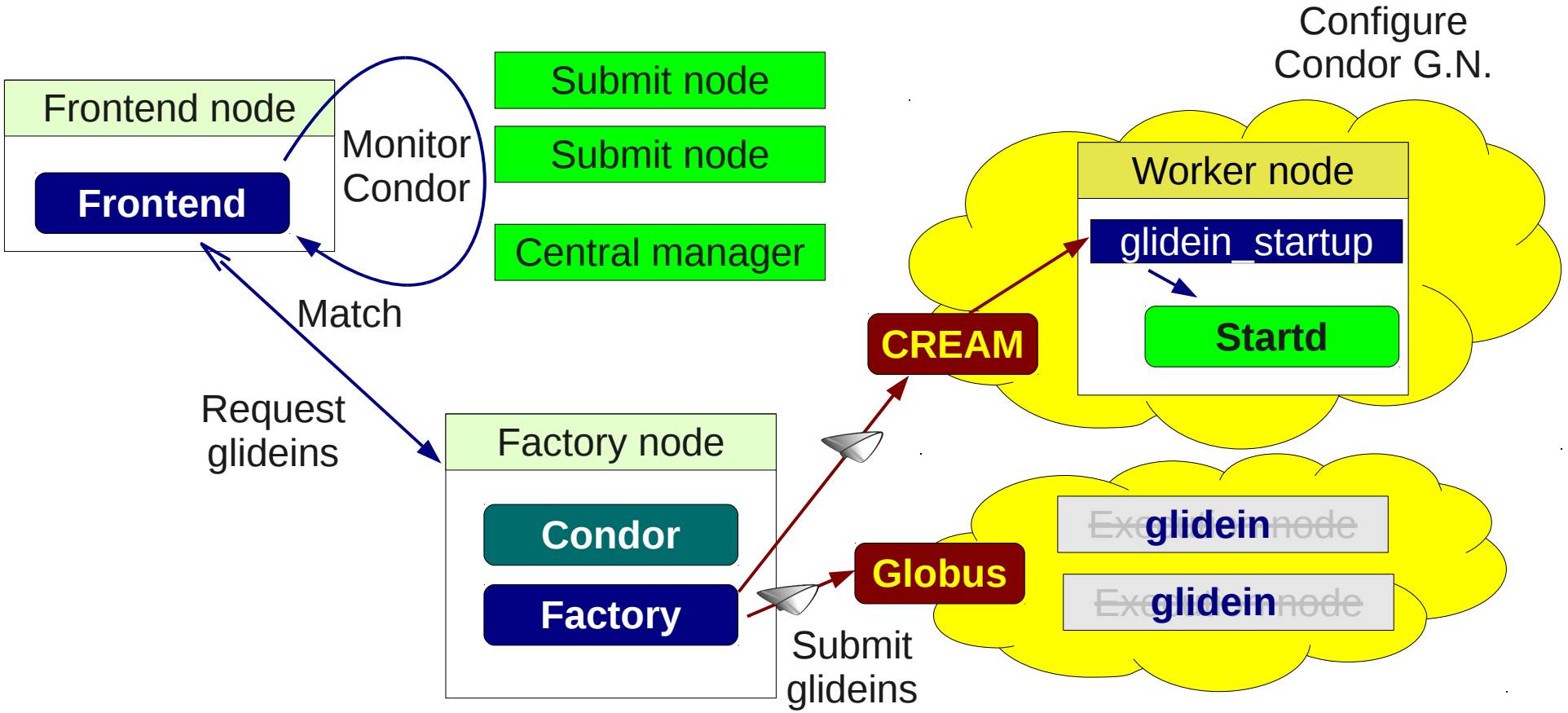
glideinWMS

A (sort of) detailed view of the

glidein factory

Refresher – glideinWMS arch.

- The factory knowns about the grid and submits glideins

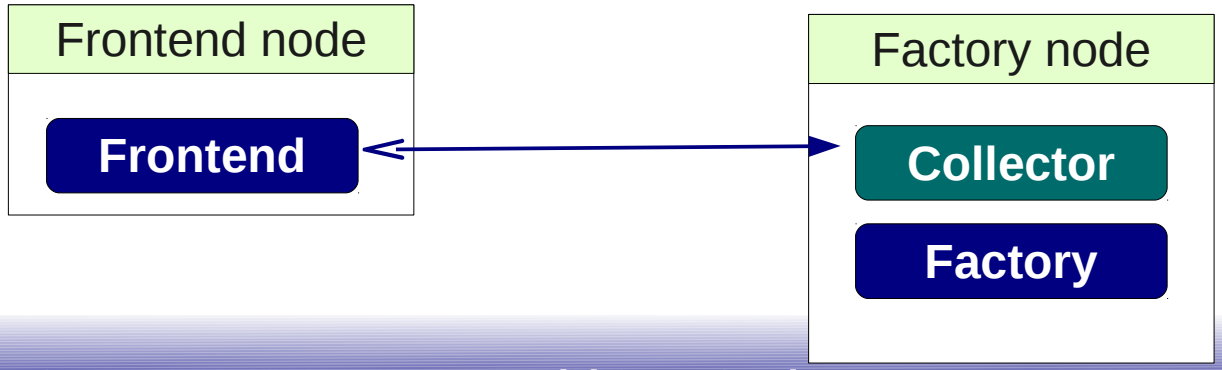


Glidein factory

- Glidein factory knows how to contact sites
  - List in a local config
  - Only trusted and tested sites should be included
- For each site (called **entry**)
  - Contact info (Node, grid type, jobmanager)
  - Site config (startup dir, glexec, OS type, …)
  - VOs supported
  - Other attributes (Site name, closest SE, ...)
- Admin maintained, XML format
  http://tinyurl.com/glideinWMS/doc.prd/factory/configuration.html

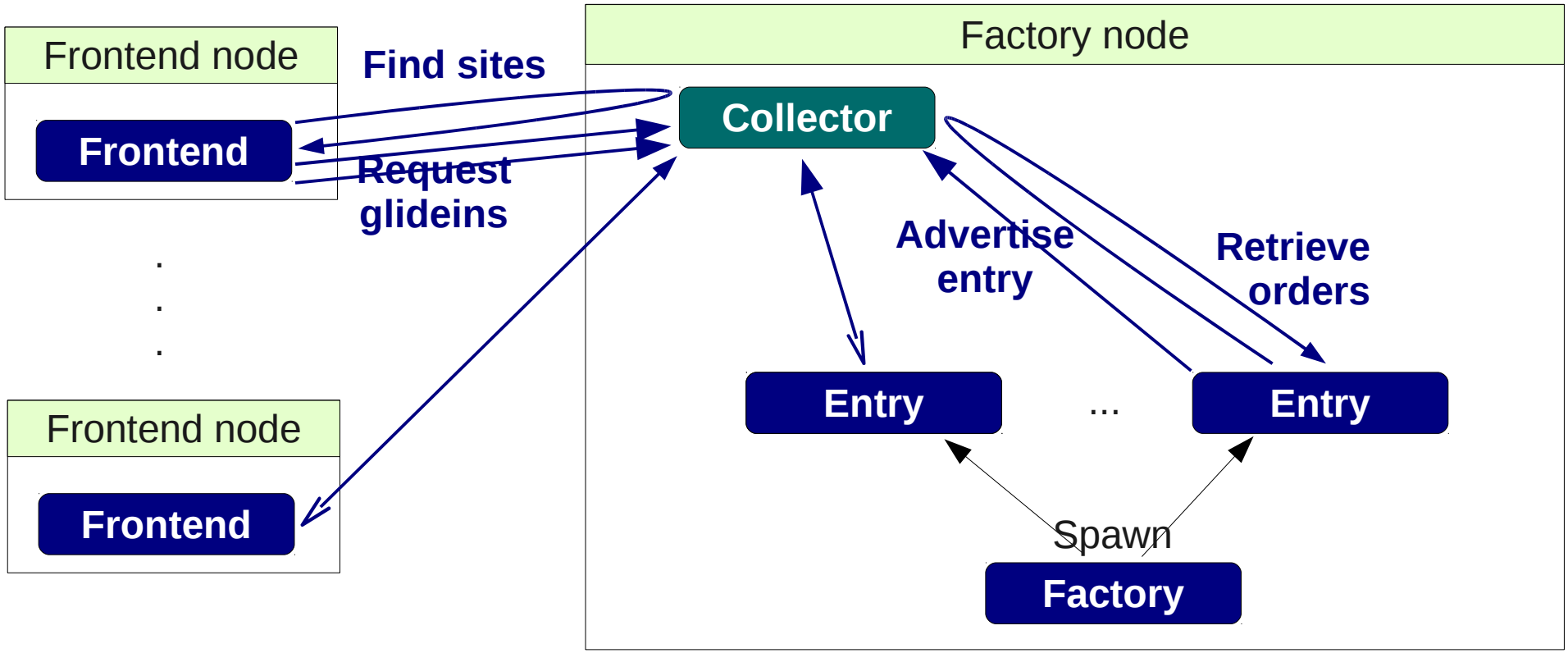
Glidein factory role

- The glidein factory is just a slave
  - The frontend(s) tell it how many glideins to submit where
  - Once the glideins start to run, they report to the VO collector and the factory is not involved
- The communication is based on ClassAds
  - The factory has a Collector for this purpose



Factory collector

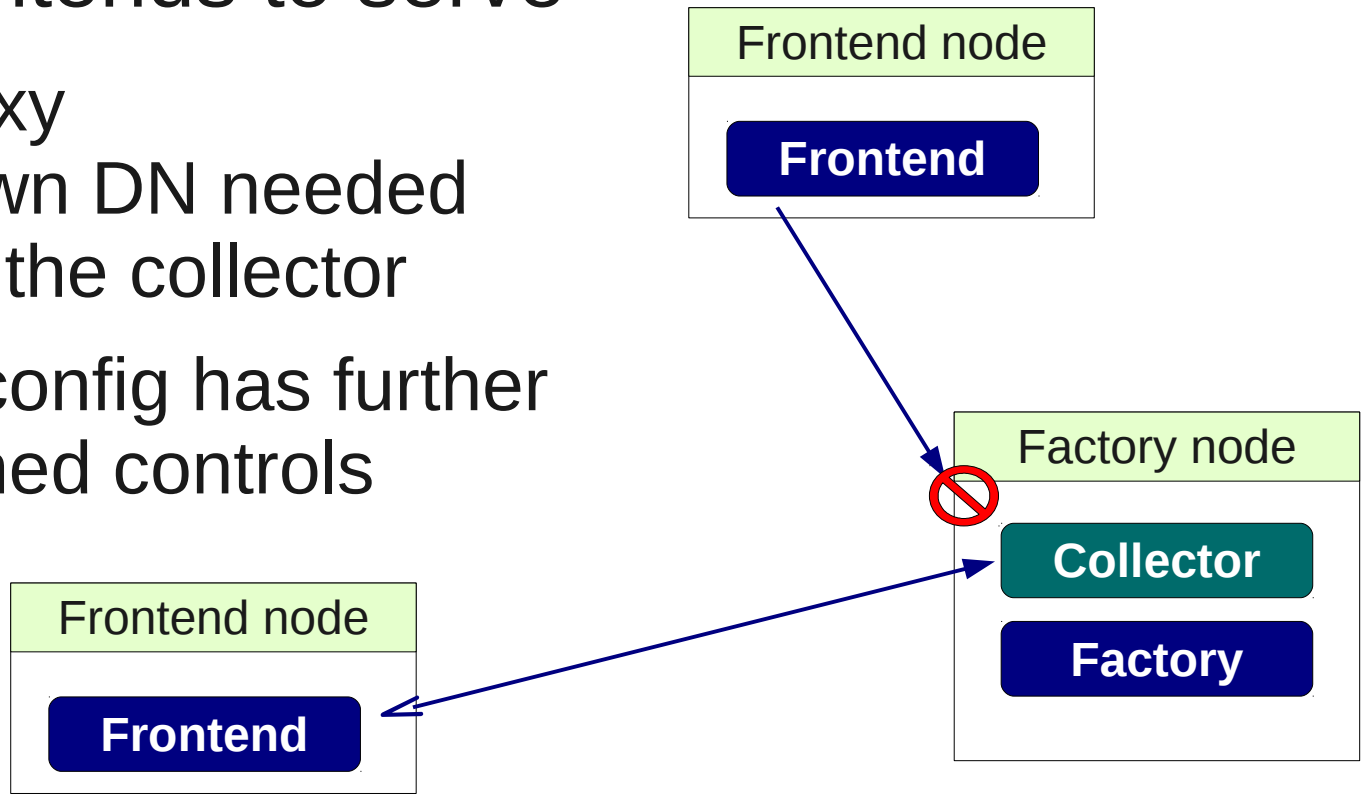
- The factory collector handles all communication



http://tinyurl.com/glideinWMS/doc.prd/factory/design_data_exchange.html

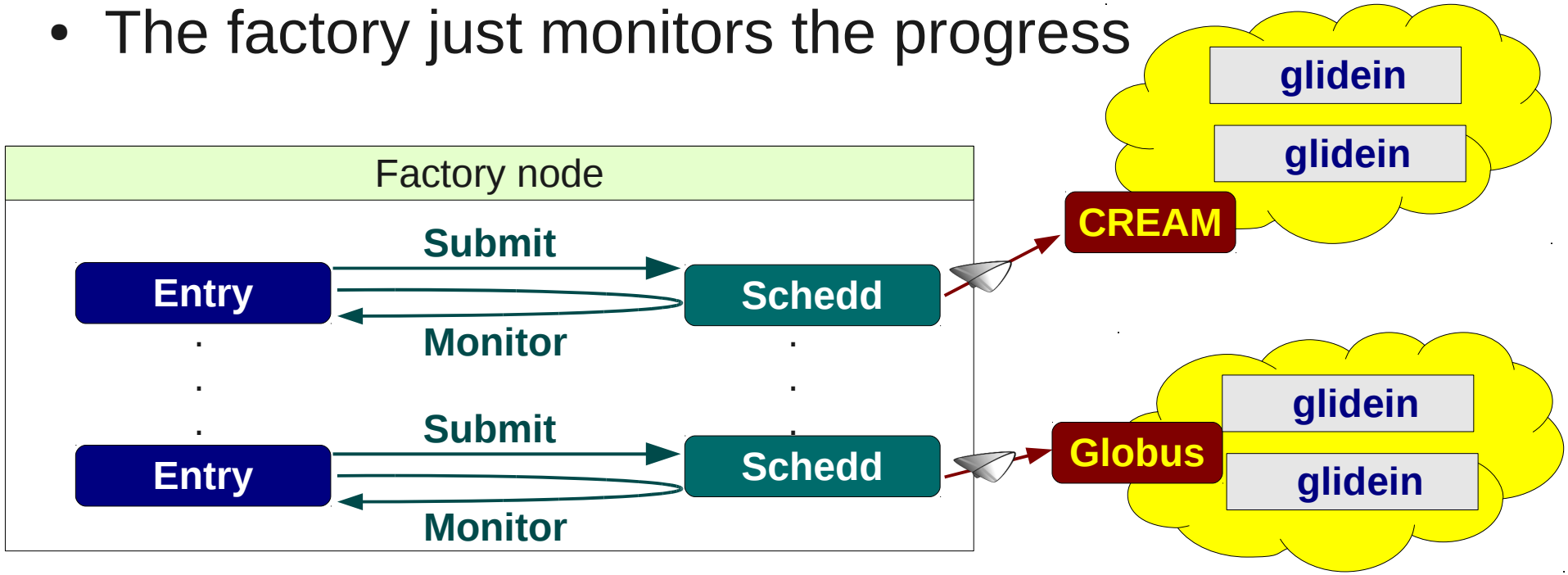
Frontends

- 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



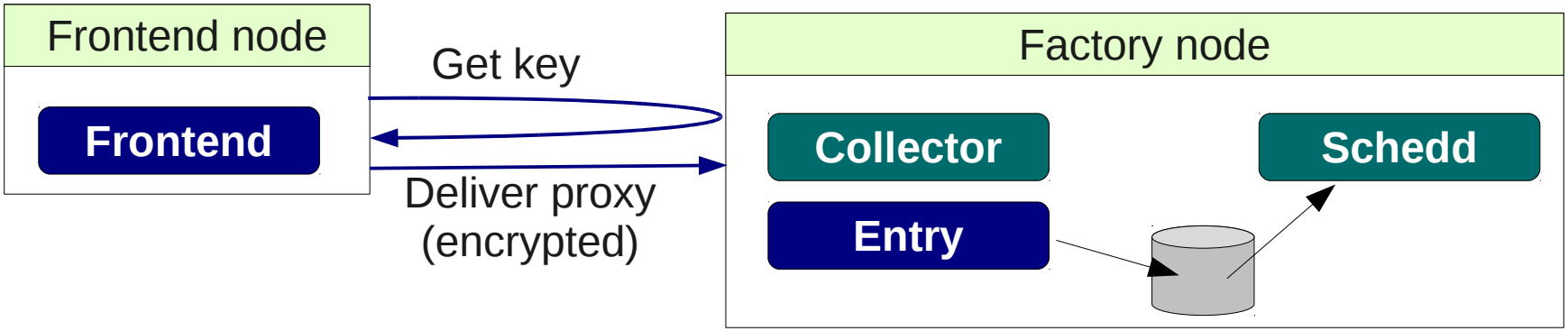
Glidein submission

- The glidein factory (entry) uses Condor-G to submit glideins
  - Condor-G does the heavy lifting
  - The factory just monitors the progress



Credentials/Proxy

- Proxy typically provided by the frontend
  - Although the factory can provide a default one (rarely used)
- Proxy delivered encrypted in the ClassAd
  - Factory (entry) provides the encryption key (PKI)
- Proxy stored on disk
  - Each VO mapped to a different UID



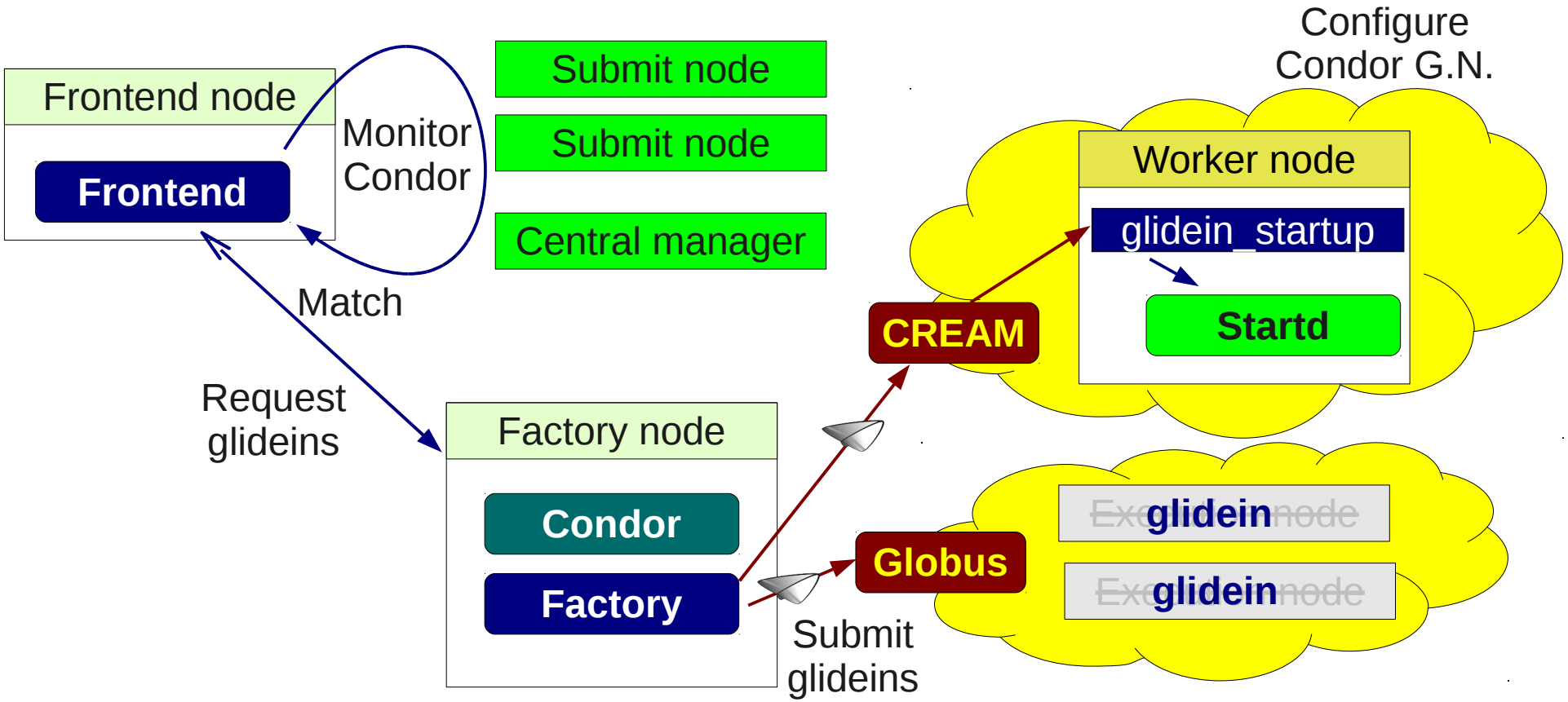
glideinWMS

A (sort of) detailed view of the

VO frontend

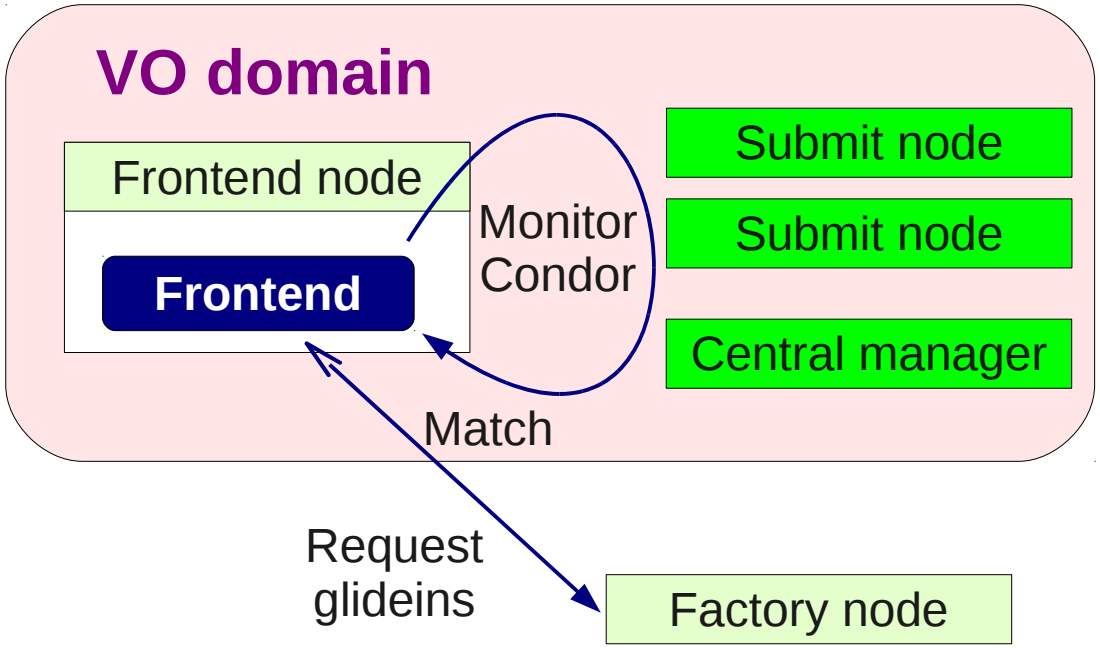
Refresher – glideinWMS arch.

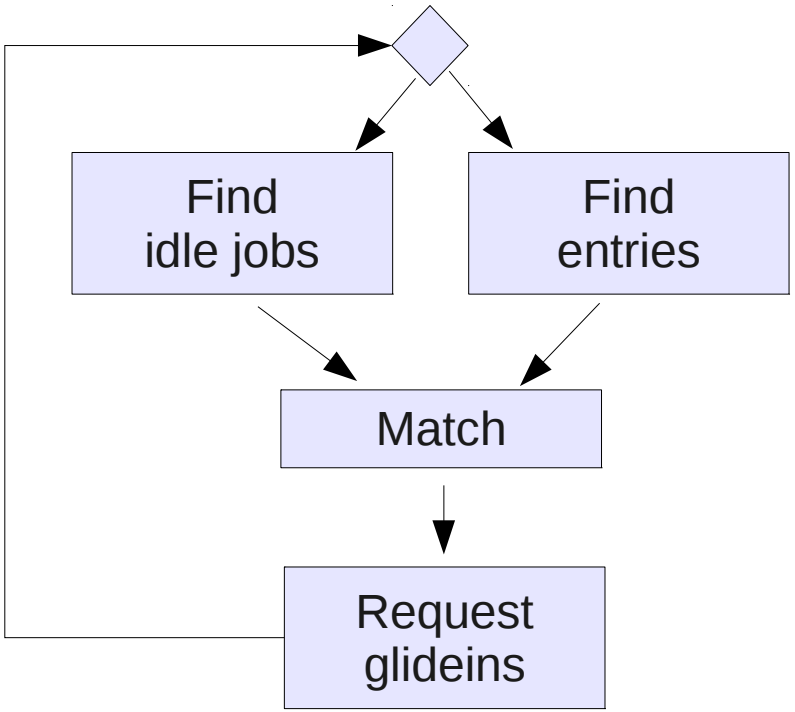
- The frontend monitors the user Condor pool, does the matchmaking and requests glideins



VO frontend

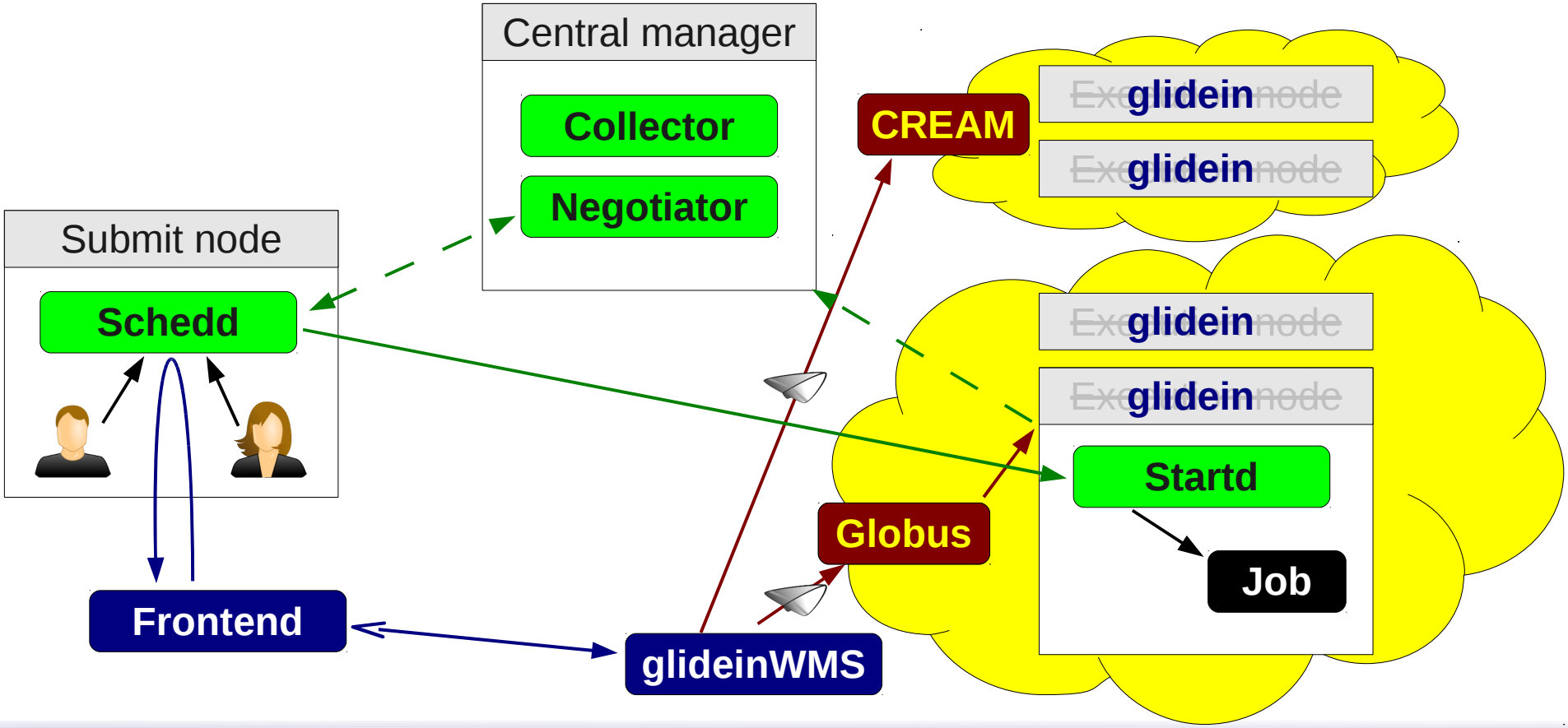
- The VO frontend is the brain of a glideinWMS-based pool
  - Like a site-level “negotiator”





Two level matchmaking

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



Frontend logic

- The glideinWMS glidein request logic is based on the principle on “constant pressure”
  - Frontend 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 and the frontend are in a polling loop and talk to each other indirectly

Frontend logic

- 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)

Frontend config

- The frontend owns the “glidein proxy”
  - And delegates it to the factory(s) when requesting glideins
  - Must keep it valid at all times (usually at OS level)
- The VO frontend can (and should) provide VO-specific validation scripts
- The VO frontend can (and should) set the glidein start expression
  - Used by the VO negotiator for final matchmaking

And the

summary

Summary

- Glideins are just properly configured Condor execute nodes submitted as Grid jobs
- The glideinWMS is a mechanism to automate glidein submission
- The glideinWMS is composed of three logical entities, two being actual services:
  - Glidein factories know about the Grid
  - VO frontend know about the users and drive the factories

Pointers

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