glideinWMS frontend system setup

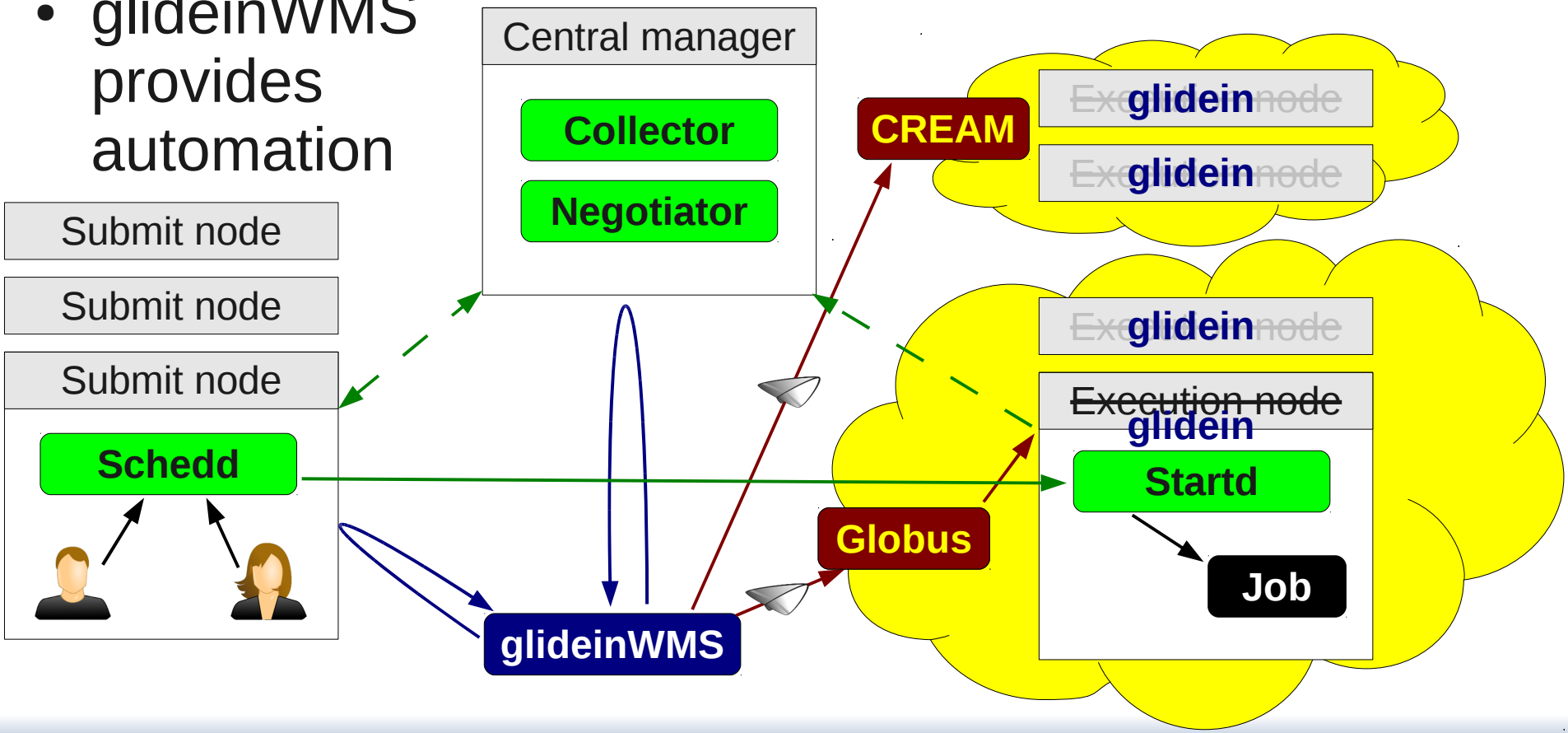
by Igor Sfiligoi (UCSD)

Overview

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

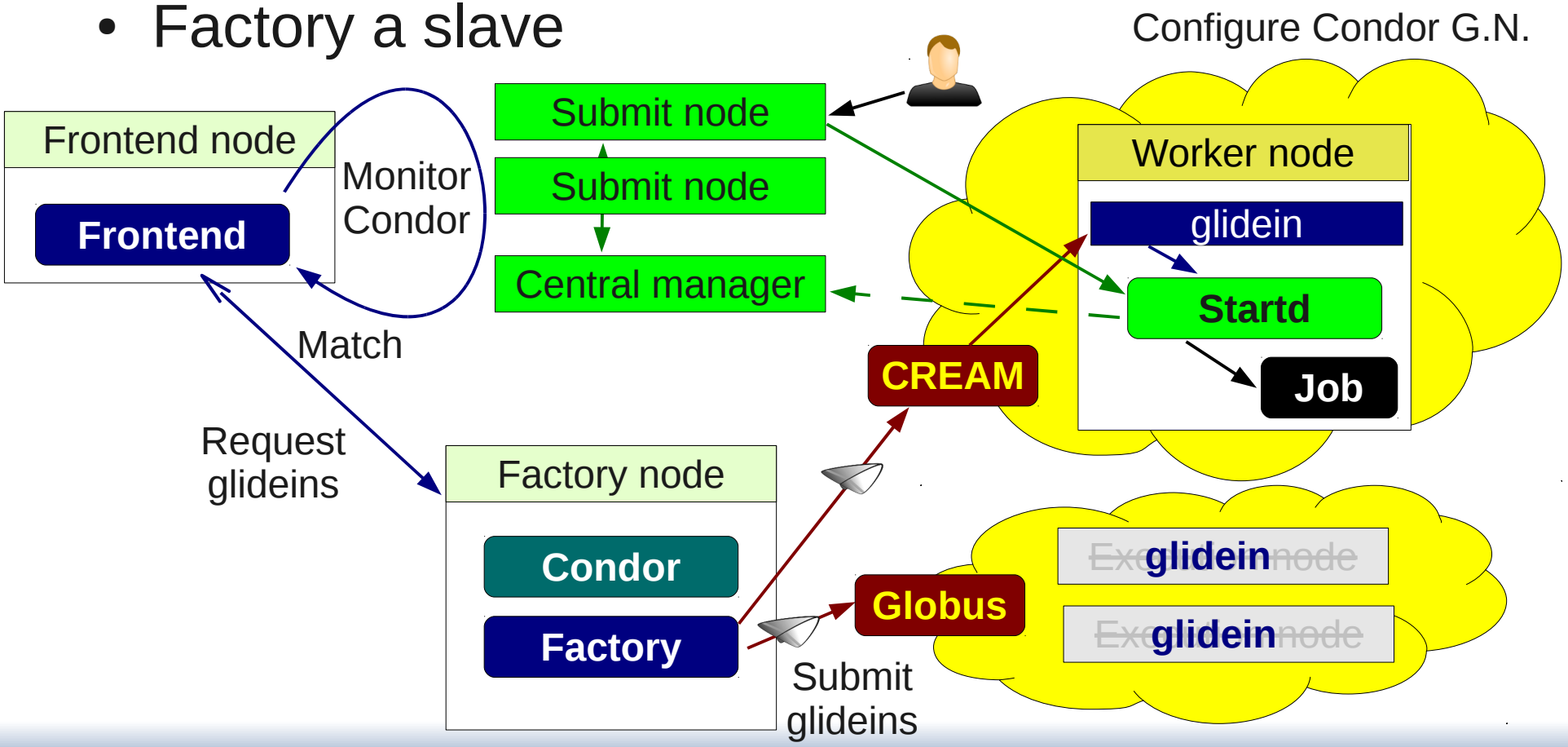
Refresher - Glideins

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



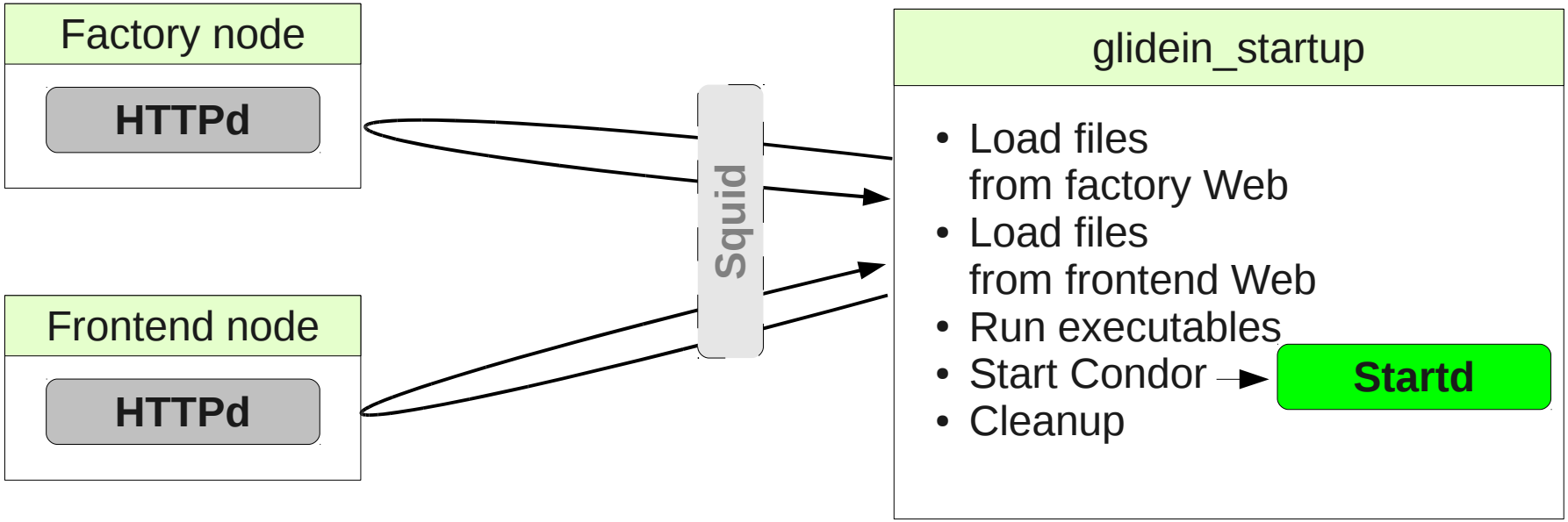
Refresher – Glidein Frontend

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



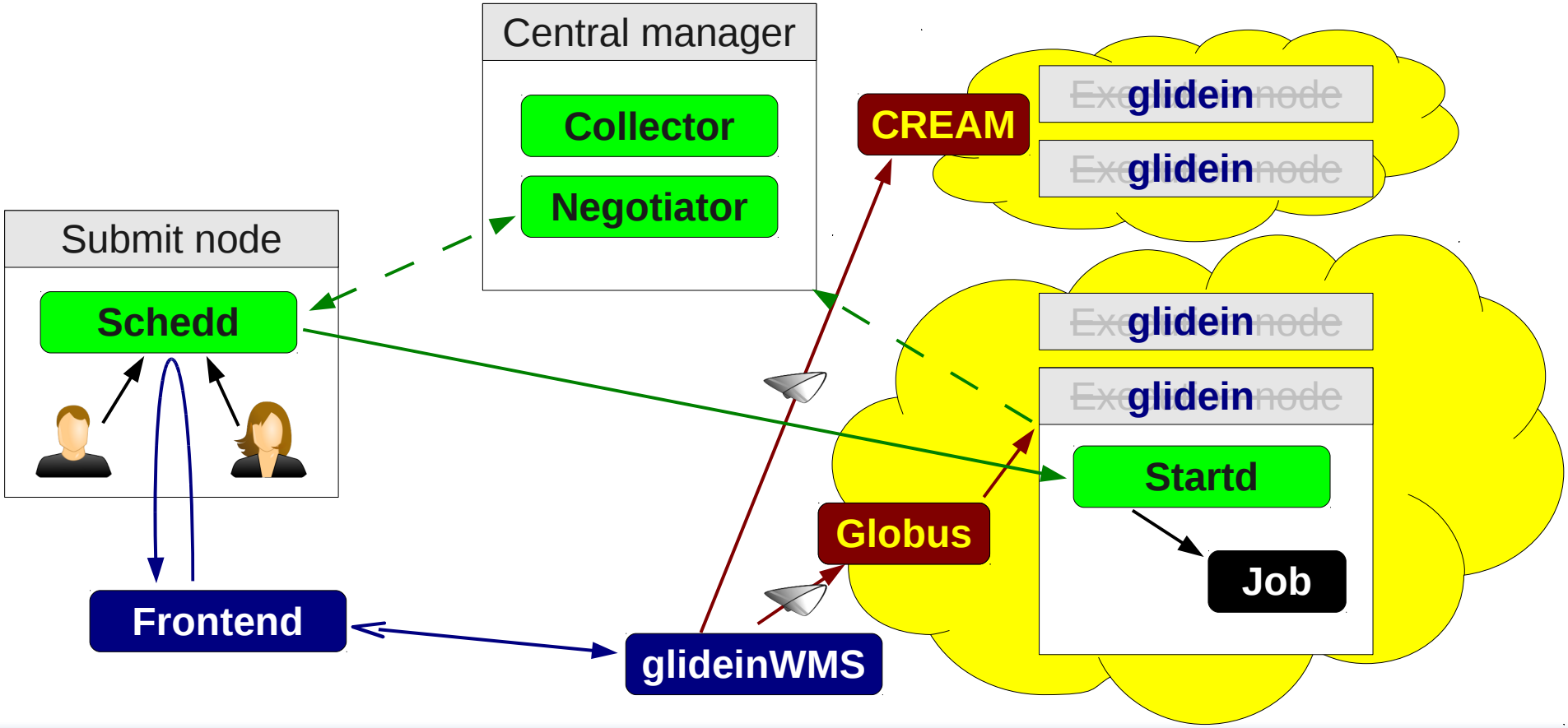
Refresher – Glidein startup script

- Basically an empty shell
- Configuration files and scripts loaded via HTTP
  - From both Factory and Frontend
  - So a Web server needed



Refresher - Two level matchmaking

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



Bottom line

Condor is king!

(glideinWMS just a small layer on top)

Components of a “frontend”

- A VO must run
  - Condor
  - Web server
  - GlideinWMS Frontend
- It also needs:
  - OS
  - GSI security setup (cert & CAs)
  - Libraries (RRDTool, JavascriptRRD, ...)

Components of a “frontend”

- A VO must run
  - Condor
  - Web server
  - GlideinWMS Frontend
- It also needs:
  - OS
  - GSI security setup (cert & CAs)
  - Libraries (RRDTool, JavascriptRRD, ...)

Operating System

- Condor is supported on a wide variety of platforms
  - Including RHEL5, MacOS and Windows
  - The glideinWMS Frontend node needs RHEL5
- Root access is needed to install and operate some of the services
- Public TCP/IP networking needed, both incoming and outgoing
  - Can be restricted to small number of ports
- May need SSHd to allow for job submission

Web server

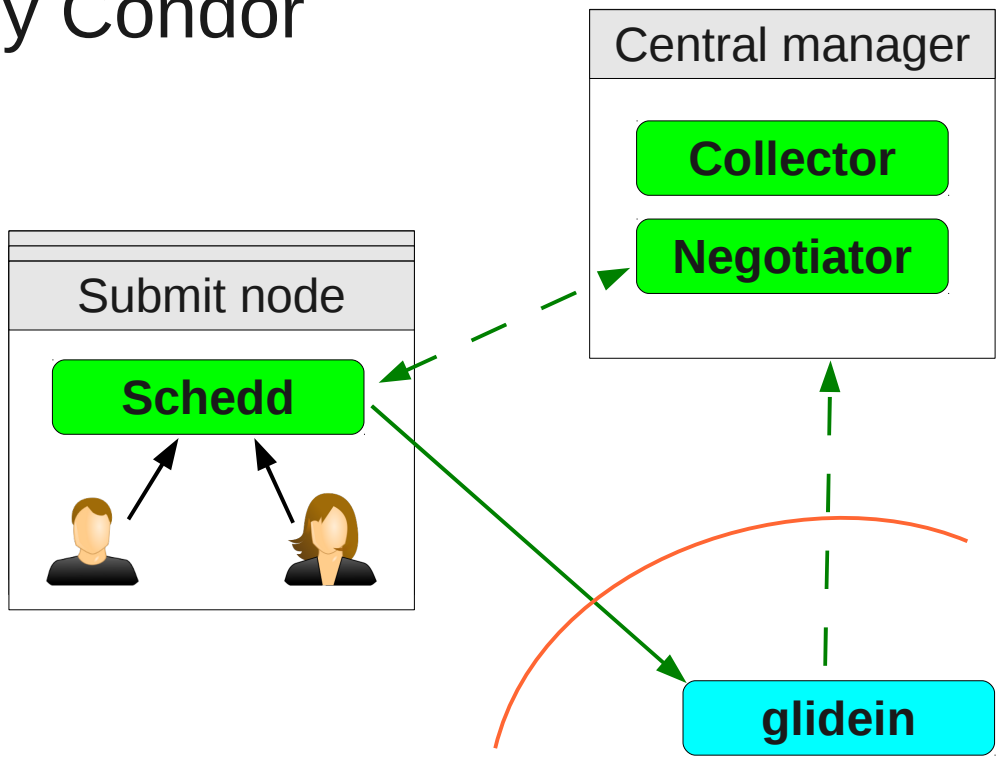
- Used for both delivering payload to worker nodes and for monitoring
  - From the glideinWMS Frontend node only
  - No dynamic content served, only static files
- Any Web server will work
  - System provided Web server recommended
  - The glideinWMS installer will also properly configure it, if desired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

GSI setup

- CA & CRLs needed on all nodes
  - Both for communication with factories and glideins
  - Typically delivered via VDT in OSG (but other means acceptable)
    https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallCertAuth
- Host certs needed for communication between Condor processes
  - Service certs acceptable as well
- Full Grid client software recommended
  https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallOSGClient
  - Not essential, but useful for operations

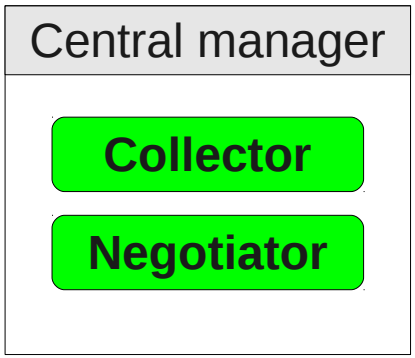
Condor

- As stated before, Condor is the major component
  - Most work performed by Condor
- Two main node types
  - Submit node(s)
  - Central manager
  - (execute nodes are dynamic – glideins)
- Frontend will need client tools, too



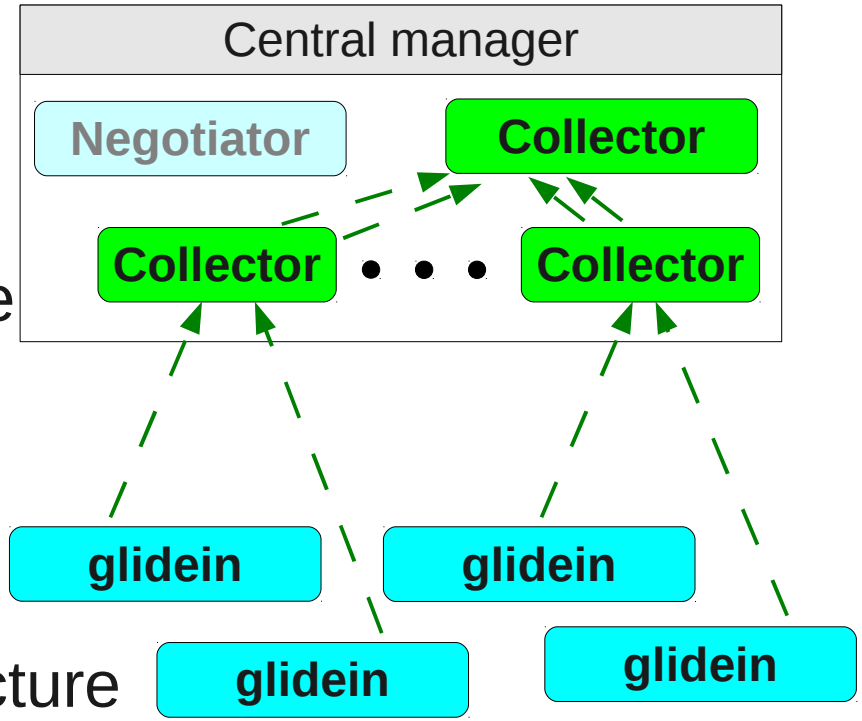
Condor Central Manager

- The Central Manager defines the Condor pool
  - Knows about all the glideins it owns
  - Decides who gets what resources
- Two (groups of) processes
  - Collector
  - Negotiator
- See Condor talk about how matchmaking works



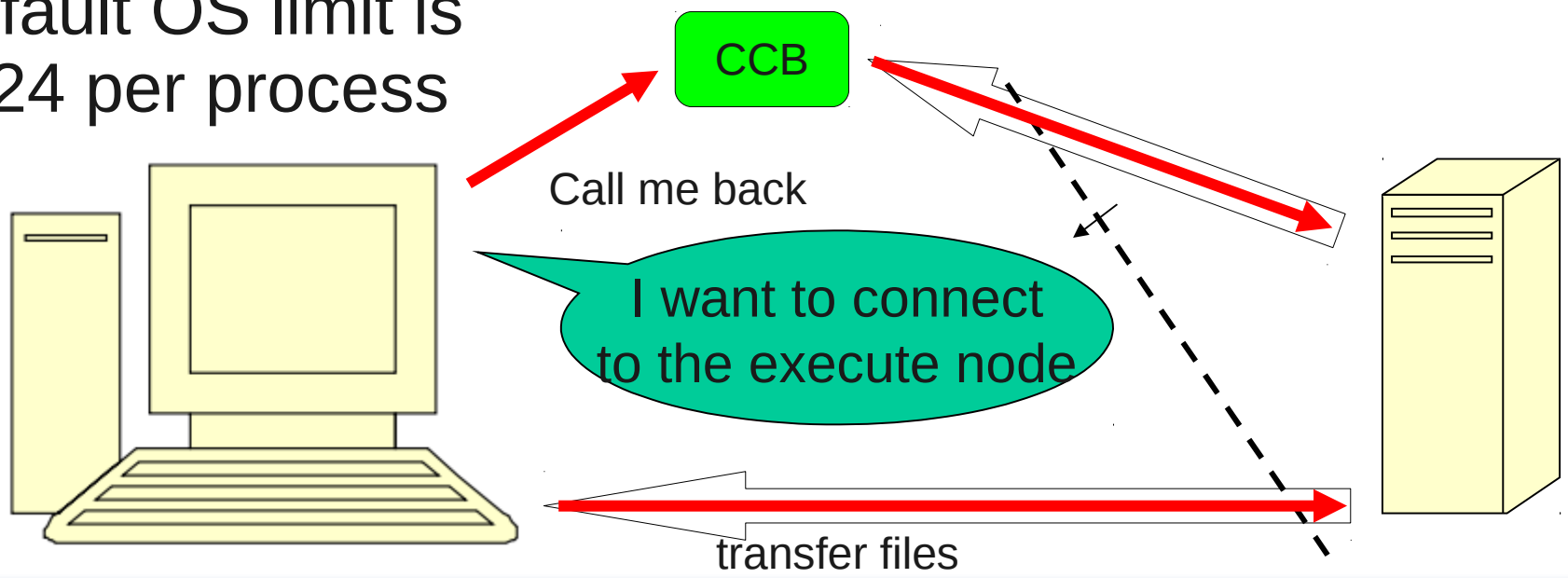
Condor Collector

- The Collector is the **repository of all knowledge**
  - All other daemons report to it
  - Including the glideins, who get its address at run-time
- Must process lots of info
  - One update every 5 mins from each and every daemon
  - With strong security → expensive
- Typically deployed as a **tree of collectors**
  - All security handled in leafs
  - Top one still has the complete picture



CCB

- The Condor collectors are also acting as CCBs
  - Each glidein will open 5+ long-lived TCP sockets
- Make sure you have enough file descriptors available
  - Default OS limit is 1024 per process



High availability

- Central manager can be a single point of failure
  - If it dies, the Condor pool dies with it!
- To avoid this, one can deploy multiple CMs
  - All daemons will advertise to 2 (or more) collectors
  - All CMs will have the same view of the world
- There can only be one Negotiator, though
  - One negotiator will be Active, all others in standby
  - More details on Condor man page
    http://www.cs.wisc.edu/condor/manual/v7.6/3_11High_Availability.html#SECTION004112000000000000000

Installing the CM

- Should be a separate node from the submit node
  - For both performance and security reasons
- Does **not** need to run as root (although it can)
  - Make sure the host/service cer&/key are readable by that user
- The collector is central trust point
  - The DNs of all other daemons are whitelisted here (schedds, glideins and Frontend)

Installing the CM, cont

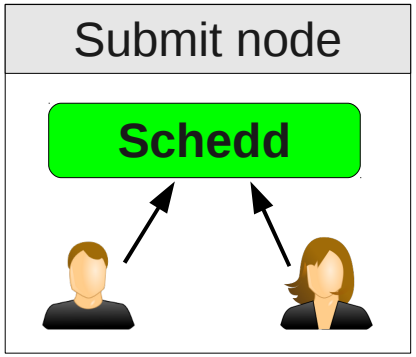
- Setting up the Collector tree can be tricky
  - Better to use the glideinWMS installer
- You may also want to set user priorities
  - Relative priorities (priority factors)
    - A priority factor of 10x will give a user 1/10 of resources
    - Default is 1x, better change it if you want privileged users
  - User groups and quotas
    - Can cap groups of users
    - Can even be hierarchical
    - But needs collaboration on the submit side

HW needs

- Tree of collectors spreads the load over multiple processes
  - But may still need a few CPUs for big pools
- Negotiator can benefit from fast CPU
  - Single threaded
- Memory usage not terrible
  - O(10k) per glidein to store ClassAds
- Negligible disk IO
- Many network sockets (may need firewall tuning)

Submit node(s)

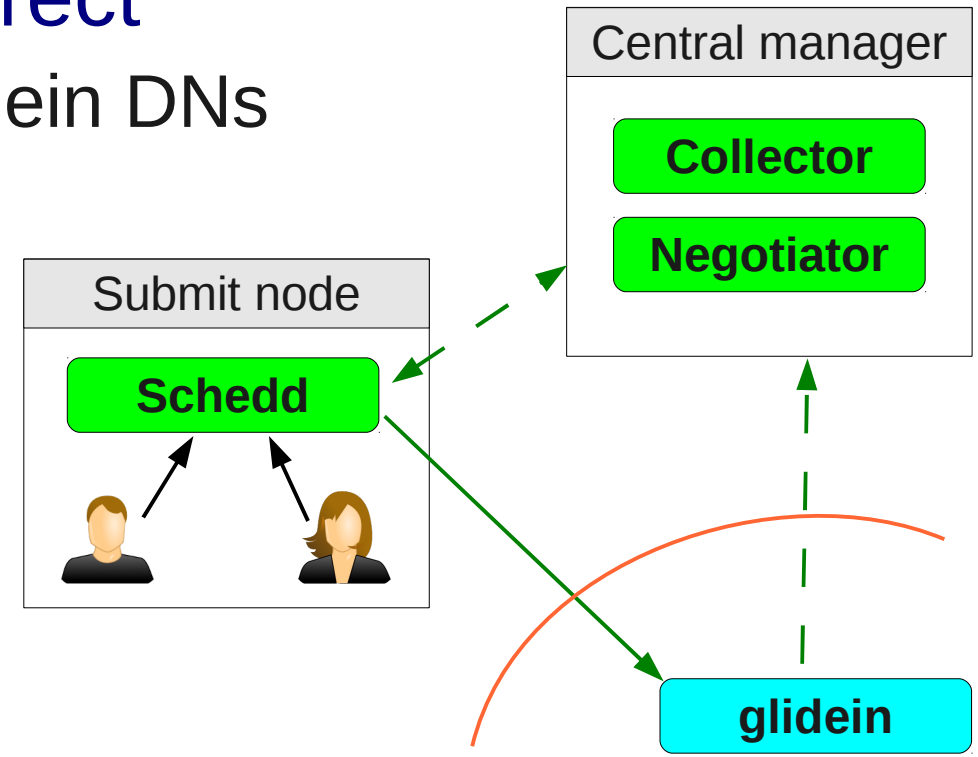
- At least one submit node is needed
  - But there may be many
- Submit node defined by the schedd
  - All user jobs submitted on this node will be handled by it
  - Shadows will be started as the jobs are matched to glideins



- Must be installed as root
  - To switch UID for R/W user files
  - As a side effect, all Auth with users UID based (FS)

Schedd security

- Schedd and CM must whitelist each other
  - Proxy DN based
- AuthZ with glideins indirect
  - No need to whitelist glidein DNs
  - Collector trusts glidein, Schedd trusts Collector
- Schedd also must whitelist Frontend DN
  - Frontend cannot use indirect AuthZ



Network use

- Glideins must contact the submit node in order to run jobs
  - Both due to standard protocol and CCB
- Each shadow normally uses 2 ports
  - Can be a problem over O(10k) jobs
- Newer versions of Condor support "**shared port daemon**"
  - Listens on a single port
  - Forwards the sockets to the appropriate local process

Installing the schedd

- Nothing special about it
  - Just GSI setup & SEC_ENABLE_MATCH_PASSWORD_AUTHENTICATION
- May want to tune the scalability
  - Shared port daemon
  - MAX_JOBS_RUNNING
  - JOB_START_DELAY, JOB_START_COUNT, …
  - SCHEDD_QUERY_WORKERS
- May want to define reasonable defaults
  - APPEND_REQ_VANILLA
  - PERIODIC_*

HW requirements

- Submit node is memory hungry
  - 1M per running jobs due to shadows
  - O(10k) per job in queue for ClassAds
- Jobs may put substantial IO load on node
  - Depends on how much data is being produced
  - Depends how short are the jobs
- Lots of network sockets open
  - May need firewall tuning

User account considerations

- Users must be able to launch **condor_submit** locally on the submit node
  - Remote submission not recommended (and disabled by default)
- VO must decide how to do it
  - SSHd
  - Portal (CRAB, WMAgent, etc.)
- Will need one UID per user

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