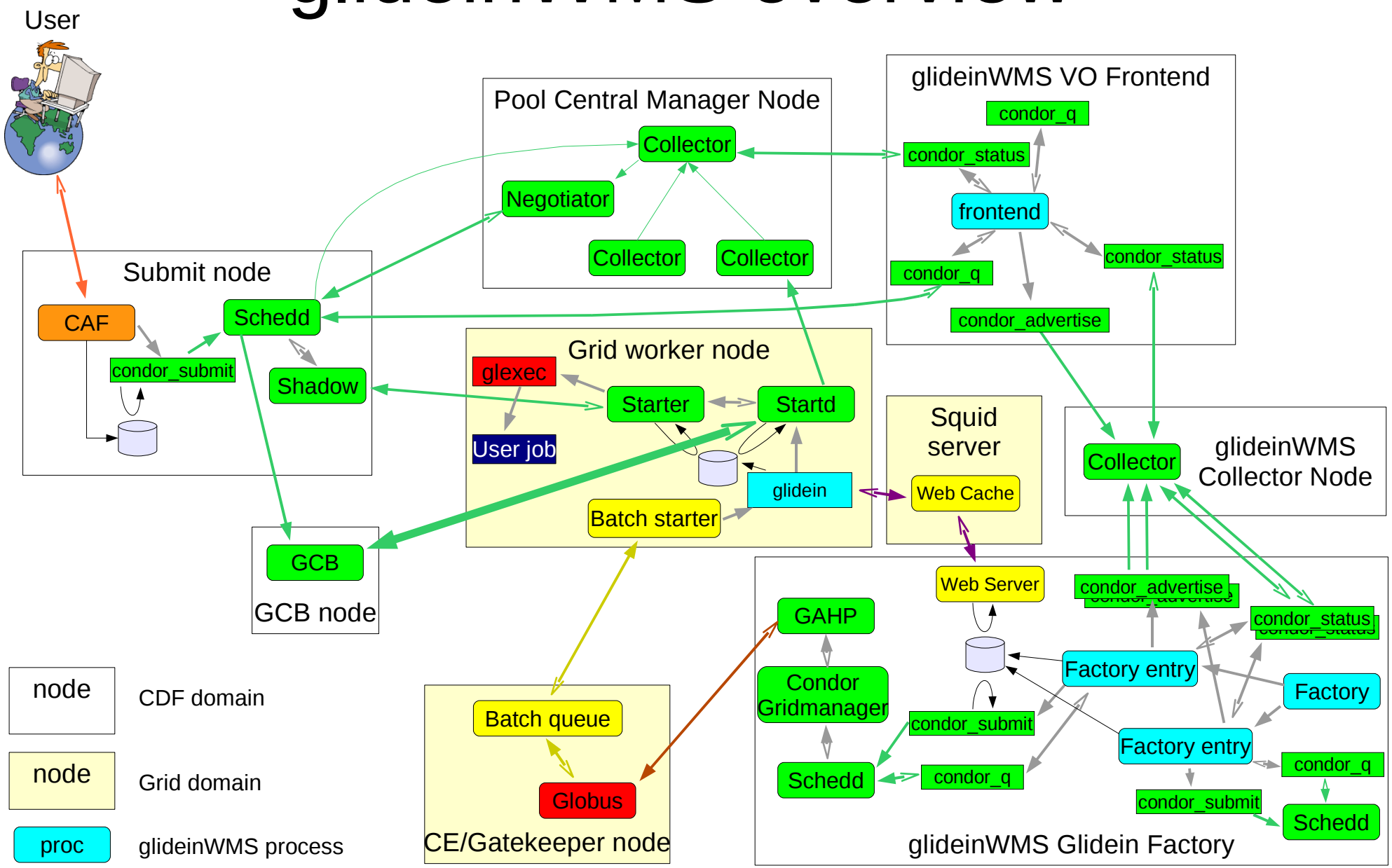
glideinWMS 101

From the CDF point of view

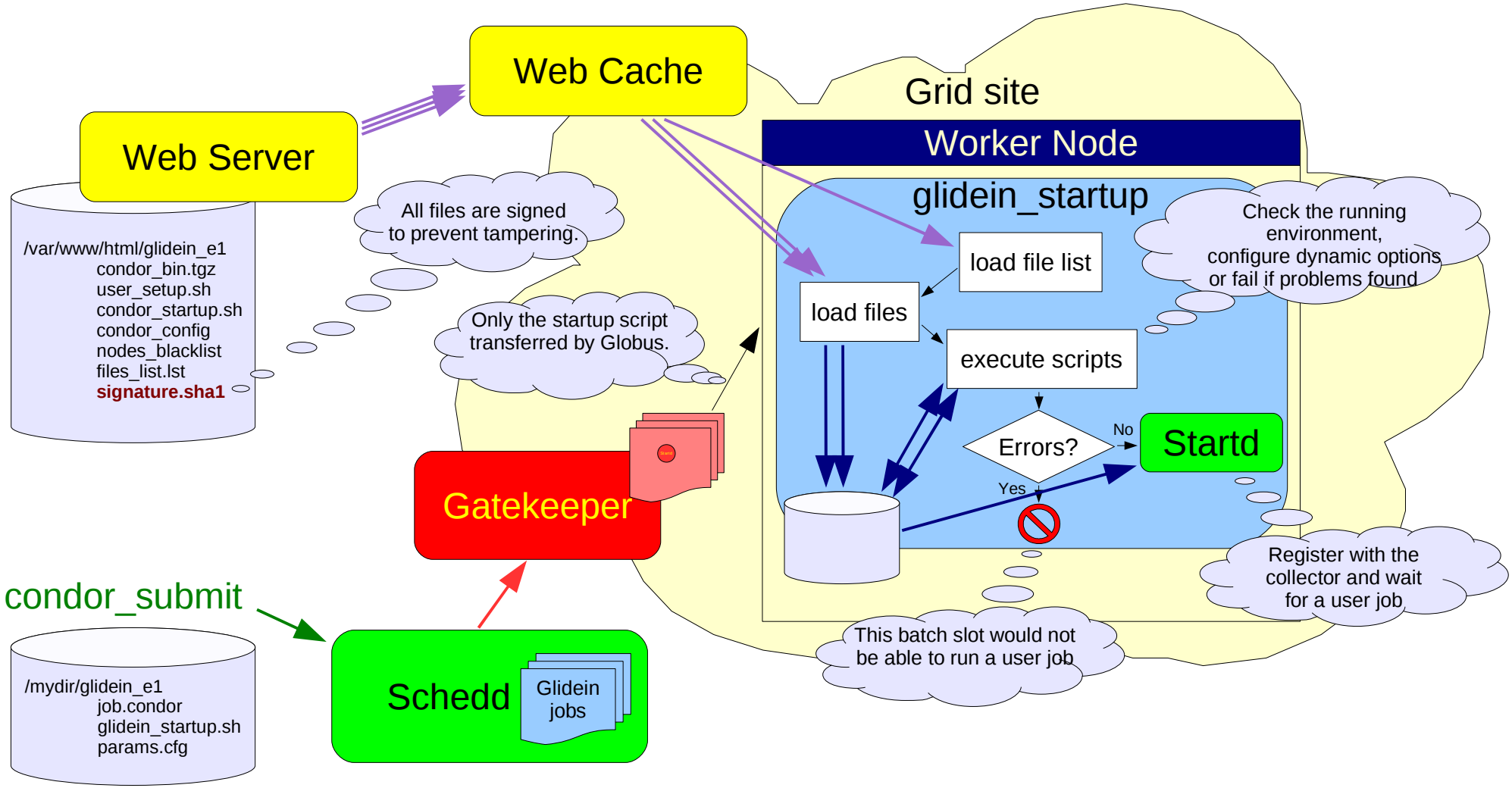
Based on glideinWMS v1.5.2 and Condor v7.2.0

by igor Sfiligoi

glideinWMS overview



Glidein startup script overview



Security considerations

- Between Condor daemons on different nodes
  - Mostly based on GSI (host cert or pilot proxy)
  - Between schedd/shadow and startd/starter a one time shared key is used (startd uploads to collector)
  - No authentication for GCB
- Between processes on the same node
  - Mostly based on UID
  - CDF uses GSI for job submission (robots proxies)

Security considerations (2)

- glideinWMS security
  - GSI used for all communication between nodes
  - GSI (pilot proxy) used for job submission to Globus
  - Pilot proxy passed to condor daemons launched by the glidein
- Web traffic uses two stage hash keys
  - Master hash key delivered as a parameter
    - Used to verify the integrity of the “signatures” file
  - All other files have a hash key in the “signatures” file

Scalability considerations

- Main pool (not glideinWMS)
  - Collector sensitive to latencies
    - Over the Atlantic, can only sustain 0.1Hz new glideins (Due to initial security handshakes)
    - In the US should be better but plan to use a hierarchy for all offsite glideins anyhow
  - Negotiator uses quite a bit of CPU and memory, but was never the bottleneck in my tests
  - Single schedd tested up to ~11k running jobs and ~200k jobs in the queue
    - Stable at that scale (uses all slots, no major job restarts or failures)
    - >12k starts to break down in my tests

Scalability considerations(2)

- Main pool (not glideinWMS) (2)
  - Each shadow uses 2 ports , O(10) file descriptors and O(1.5M) of memory
  - GCB needs 5-6 ports per running glidein
    - The default GCB installation only supports 4k glideins
      GCB_MAX_RELAY_SERVERS*GCB_MAX_CLIENTS_PER_RELAY_SERVER/5
    - The Linux defaults only allow ~28k ports
      /proc/sys/net/ipv4/ip_local_port_range
- glideinWMS Condor pool
  - Collector holds minimal data, not a problem

Scalability considerations (3)

- glideinWMS Condor pool (3)
  - Schedd/GAHP
    - In the past, found that it shows down with more that 7k jobs in the queue
    - But haven't retested recently
    - No other obvious scalability issues found
- VO frontend
  - No limits found yet

Scalability considerations (4)

- Glidein Factory
  - Most of the load coming from monitoring (Disk IO)
  - With default parameters, 40 entry points seem to be the limit for a typical node
    - v1.6 will improve this (at least factor 2)
  - If needed, disable parts of the monitoring

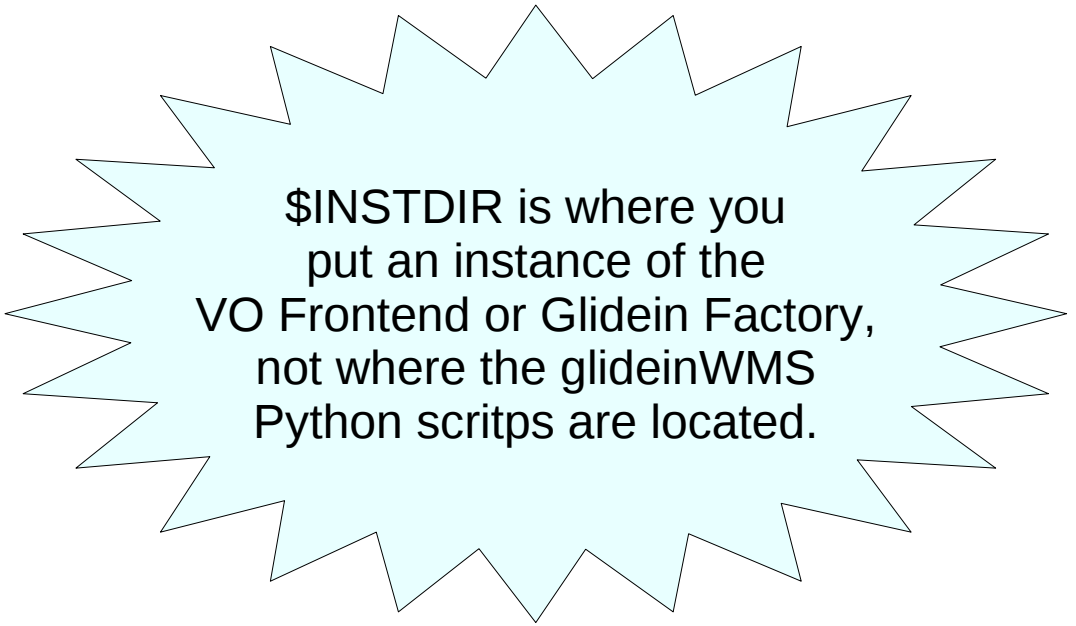
```
<monitor>
    <entry want_infoage_graphs="True" want_split_graphs="True"
want_split_terminated_graphs="False" want_trend_graphs="True"/>
    <factory want_infoage_graphs="True" want_split_graphs="True"
want_split_terminated_graphs="True" want_trend_graphs="True"/>
</monitor>
```

Operations - Startup

- Startup order
  - First start the CDF collector and schedds
  - Next comes glideinWMS collector
  - Finally VO Frontend and the Glidein Factory
- Condor startup (installed by glideinWMS installer)
  - If installed as root
    /etc/init.d/condor start
  - If installed as regular user
    $CONDORDIR/start_condor.sh

Operations – Startup (2)

- VO Frontend
  - $INSTDIR/frontend_startup start
- Glidein Factory
  - $INSTDIR/factory_startup start



Operations – Check status

- Condor
  - condor_q
  - condor_status
- VO Frontend
  - $INSTDIR/frontend_startup status
- Glidein Factory
  - $INSTDIR/factory_startup status
  - $INSTDIR/factory_startup statusdown factory
  - $INSTDIR/factory_startup info -entries

Operations - Shutdown

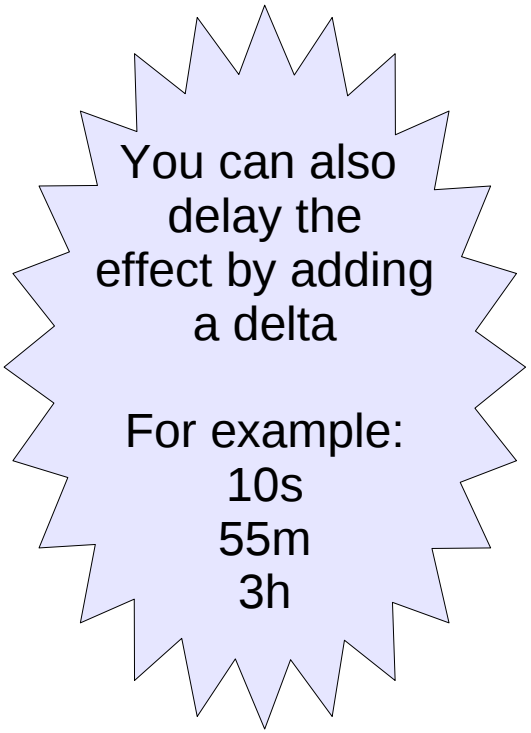
- Condor
  - Send kill to condor_master
    killall condor_master
  - On submit machine, may want to kill all the processes if there are many jobs running (timeouts)
- VO Frontend
  - $INSTDIR/frontend_startup stop
- Glidein Factory
  - $INSTDIR/factory_startup stop

Operations - Reconfig

- Condor
  - Change config and force reload
    condor_reconfig -<daemon name>
- VO Frontend
  - Change config
    $INSTDIR/etc/vofrontend.cfg
  - Restart VO Frontend
    $INSTDIR/frontend_startup reconfig
- Glidein Factory
  - Change **copy of config** (CDF uses: $INSTDIR.cfg/glideinWMS.xml)
    (never modify the main config: $INSTDIR/glideinWMS.xml)
  - Restart the Glidein Factory
    $INSTDIR/factory_startup reconfig <path to copy>

Operations – Downtime handling

- Glidein Factory
  - Temporarily disable an entry point
    $INSTDIR/factory_startup down <entry name>
  - Re-enable a disabled entry point
    $INSTDIR/factory_startup up <entry name>
  - Check what entries are in downtime
    $INSTDIR/factory_startup statusdown entries
  - Temporarily disable the whole factory (but keep monitoring)
    $INSTDIR/factory_startup down factory
  - Re-enable the factory
    $INSTDIR/factory_startup up factory



Operations – Upgrade to new version

- Condor
  - Just replace the binaries (stop before, start after)
- VO Frontend
  - Just replace the binaries (stop before, start after)
- Glidein Factory
  - Must create a new glidein tree
  - Can reuse the same config file
    - **Change the glidein name**!
- Condor for glideins
  - Change the glideinWMS_copy.xml and reconfig
    (CDF glideinWMS_copy.xml: $INSTDIR.cfg/glideinWMS.xml)
  - Remove the link to the old tarball
    <condor base_dir="$STAGE/condor-7.2.0" **~~tar_file~~**~~="$WEB/condor_bin.92okQX.tgz"~~/>

Debugging

- Submit machine
  - Look for error messages in SchedLog and ShadowLog
  - Try to correlate with the logs from glideins
- Each glidein has its log in a different file
  - But they publish enough information to get to the right one

    condor_status
    -format "%s " Name
    -format "%s " **GLIDEIN_Entry_Name** -format "%s " **GLIDEIN_Schedd**
    -format "%s." **GLIDEIN_ClusterId** -format "%s\n" **GLIDEIN_ProcId**

Debugging (2)

- Glidein Factory - glideins
  - To see what an overview of a specific glidein
    cat $INSTDIR/entry_E/log/job.C.P.out
  - To look at StartdLog of a specific glidein
    $SRCDIR/factory/tools/cat_startdlog.py $INSTDIR/entry_E/log/job.C.P.err
  - To look at the StarterLog of a specific glidein
    $SRCDIR/factory/tools/cat_starterlog.py $INSTDIR/entry_E/log/job.C.P.err
  - To see what glideins finished recently
    tail $INSTDIR/entry_E/log/completed_jobs_<date>.log
- Glidein Factory – entries
  - tail $INSTDIR/entry_E/log/factory_err.<date>.log
  - tail $INSTDIR/entry_E/log/factory_info.<date>.log

Debugging (3)

- Fetch log files of running glideins
  - Needs pilot or schedd proxy
  - condor_fetchlog -startd <NAME> STARTD
- Peek at what a user is doing (on Submit node)
  - Needs user proxy
  - $SRCDIR/tools/glidein_ps.py <C.P>
  - $SRCDIR/tools/glidein_cat.py <C.P> <filename>
- Get the factory status (on GlideinFactory)
  - $SRCDIR/tools/wmsXMLView.py
  - $SRCDIR/tools/wmsTxtView.py

Configuration highlights

- Schedd
  - MAX_JOBS_RUNNING
  - SCHEDD_SEND_VACATE_VIA_TCP = True
    STARTD_SENDS_ALIVES = True
    SEC_ENABLE_MATCH_PASSWORD_AUTHENTICATION = True
- Negotiator
  - NEGOTIATOR_INFORM_STARTD = False
  - NEGOTIATOR_MAX_TIME_PER_SUBMITTER=90
    NEGOTIATOR_MAX_TIME_PER_PIESPIN=15
- Collector
  - CONDOR_VIEW_HOST = $(COLLECTOR_HOST)

Scalability optimization

Protect schedd from overload

Collector tree

Configuration highlights (2)

- VO Frontend
  - schedd_names – make sure you list them all
  - glidein_params['GLIDEIN_Collector'] – same here
  - match_string – this decides where to send glideins
  - max_idle_glidein_per_entry – pressure
  - max_running_jobs – abs. value to prevent problems
- Glidein Factory
  - Too much to cover, let's have a look at the manual glideinWMS/doc/manual/factory/entry.html

The end?

If you have additional question,
now is the the time to speak