VERA C. RUBIN OBSERVATORY

GlideinWMS: Your New Best Friend?

Ken Herner, FNAL

Another subtitle here or date

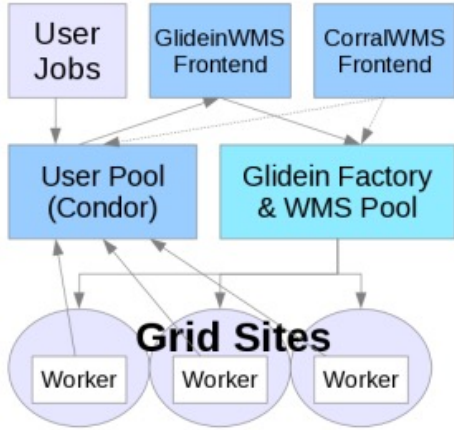
NSF AURA U.S. DEPARTMENT OF ENERGY SLAC CHARLES AND LISA SIMONYI FUND FOR ARTS AND SCIENCES LSST CORPORATION

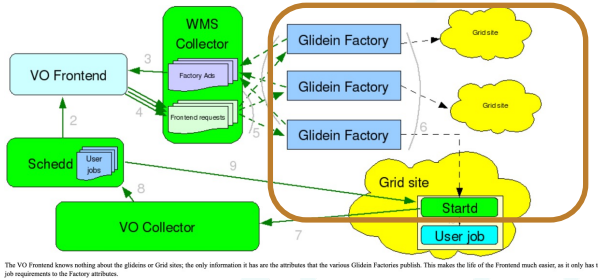
Current Job submission w/HTCondor on S3DF

- Nowadays we run allocateNodes.py to provision glideins on the Slurm queues (e.g. roma and milano). Glideins (also "pilots") with the requested amount of CPU and memory start up (what you see in the Slurm queues) and advertise HTCondor slots, which are what your bps jobs see.
  - User has to decide how many cores and memory to request. Defaults seem to be 4 GB/core
  - At the moment glideins are isolated between users: that's good and bad
  - After no work comes in after a certain amount of time, glideins self-terminate
- All works well-- as long as you remember to run it! Maybe multiple times for longer campaigns...
- Could we remove the need to keep manually refreshing glideins while being efficinet with memory and CPU choices? Fortunately, this is a (largely) solved problem!

Enter GlideinWMS

- Infrastructure to interface between a batch scheduler and one or more remote sites (many different local batch system interfaces supported, including via HTCondor-CE)
- User doesn't need to know anything about the setup at the remote cluster(s). **Slot provisioning is fully automatic**
  - Same (tunable time) principle of self-shutdown if no work
- Used extensively by CMS, DUNE, OSG Open Science Pool, etc.
  - Usual model is with a single shared pool for all users (glideins can be shared and inefficiency minimized)
  - Pools stable to order hundreds of K jobs





What allocateNodes.py does

More info at https://glideinwms.fnal.gov/doc.prd/index.html

More details

- What GWMS does do for you
  - No need to manually provision slots
  - A shared pool would minimize waste and inefficiency (let HTCondor do the slot partitioning)
  - No user-facing parts; no need to change existing submission methods
  - Very easy to expand out to new clusters/sites (just add factory entries)
  - Interface to batch system unchanged, so e.g. bps report would work as it does now
- What GWMS does NOT do
  - Actual job launches (that's still bps)
  - Bookkeeping (also still bps/whatever tools)
  - Recovery launches (you guessed it...)

More details

- Needs a frontend, factory (could join an existing factory or run it internally), collector, condor bits (already in place)
- No need to run infrastructure at a specific place (i.e. could all be at USDF)
- Would suggest the usual setup with a shared pool and small number of frontend groups
    - Downside of single pool is a naughty job can take out an entire pilot (same as now but you're only causing problems for yourself at the moment), but efficiency advantages are considerable
    - Whole-node pilots allow for maximum efficiency; could be split up as needed; other users could come in when your jobs are done without any extra intervention
    - Could just run SLAC only to start, then scale up as needed

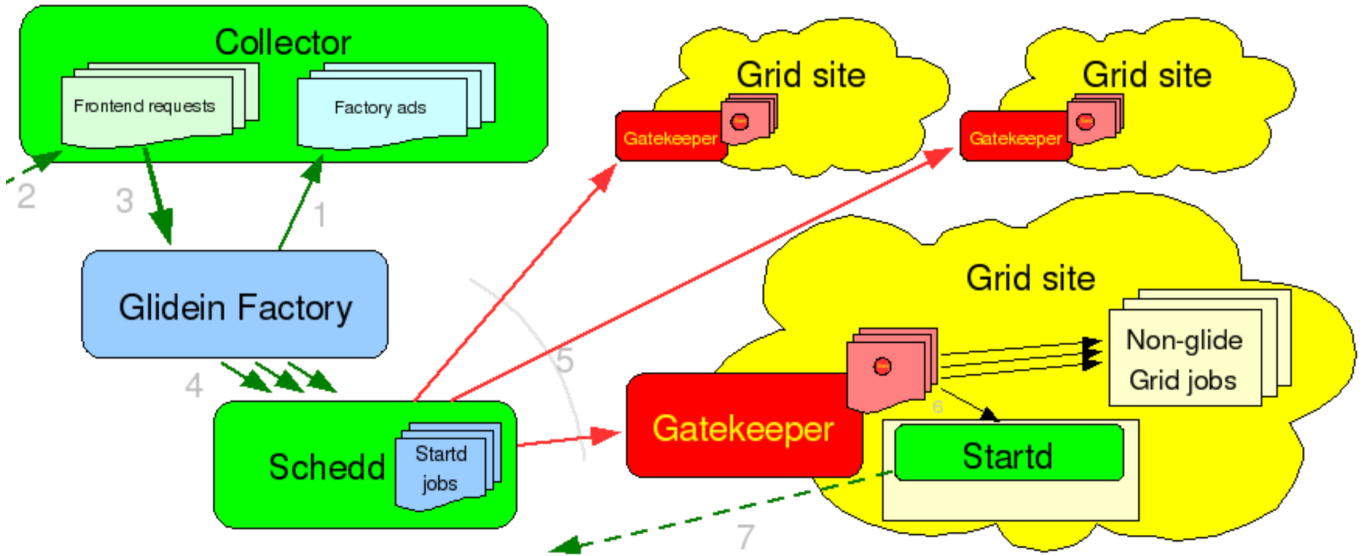
Sound good?

- Would need to convince the USDF Powers That Be to set it up
- Expect significant assistance at FNAL, OSG, elsewhere could be found

Backup

Factory-centric point of view

- Link to active factory at UCSD for OSG/DUNE/CMS: http://gfactory-2.opensciencegrid.org/factory/monitor/



single glidein Factory can handle multiple kinds of glideins, also called **glidein entry points** (as they usually point to different Grid resources). For each entry point, it will advertise a different class-ad. imilarly, each request from a Frontend client will affect a single entry point; a Frontend will need to advertise several requests in order to have glideins submitted to all the desirable resources.