GlideinWMS

Marco Mambelli
DUNE Computing Model Workshop
9 September 2019
GlideinWMS

GlideinWMS is a pilot based resource provisioning tool for distributed High Throughput Computing

- Provides reliable and uniform virtual clusters
- Submits Glideins to unreliable heterogeneous resources
- Leverages HTCondor
  - Provides HTCondor pools
  - Uses HTCondor capabilities
Glidein: node testing and customization

• Scouts for resources and validates the Worker node
  – Cores, memory, disk, GPU, …
  – OS, software installed
  – CVMFS
  – VO specific tests

• Customizes the Worker node
  – Environment, GPU libraries, …
  – Starting containers (Singularity, …)
  – VO specific setup

• Provides a reliable and customized execute node to HTCondor
Factory

- A Glidein Factory knows how to submit to sites
  - Sites are described in a local configuration
  - Only trusted and tested sites are included
- Each site entry in the configuration contains
  - Contact info (hostname, resource type, queue name)
  - Site configuration (startup dir, OS type, …)
  - VOs authorized/supported
  - Other attributes (Site name, core count, max memory, …)
  - Glideins can also auto-detect resources
- Configuration can be auto-generated (e.g. from CRIC), admin curated, stored in VCS (e.g. GitHub)
- Condor does the heavy lifting of submissions.
Factory: Supported resources

- Remote or local clusters:
  - Can have batch systems other than HTCondor: PBS, SGE, Slurm, all supported.
- Grid sites (CREAM, ARC, HTCondor-CE)
- Hosted CEs
- Commercial cloud (AWS, Google)
- Open Source Cloud (OpenStack, OpenNebula)
- HPC sites
  - Uses an ssh-based system to ssh into HPC sites and submit directly from their login nodes.
Frontend

- Monitors jobs to see how many Glideins are needed
- Compares what entries (sites) are available
- Requests Glideins from the Factory
- Requests Factory to kill Glideins if there are too many
- Pressure-based system
  - Works keeping a certain number of Glideins running or idle at the sites
  - Glideins requests are gradual to avoid spikes and overloads
- Manages credentials and delegates them to the Factory.
Distributed

- N-to-M relationship
  - Each Frontend can talk to many Factories
  - Each Factory may serve many Frontends
- Multiple User Pools
- High Availability replicas
Major GlideinWMS Deployments

- Beta version was called “GlideCAF” in CDF
  - Began testing in 2005
- CMS Global Pool—regularly 200000+ cores
  - Redundant master nodes at CERN and Fermilab
  - Combines production and analysis jobs
- FIFEBATCH / FermiGrid
  - Integrates 18000 on-site cores of FermiGrid with up to 12000 offsite cores.
  - This is what DUNE is using now for standard production and analysis
  - Demonstrated a pool with 2.01 million cores (NOVA 2018)
- Open Science Grid
  - Multi-VO structure shares the same Factory at UCSD
- HEPCloud
  - More in next talk
How it is used?

• Can be used directly
  - HTCondor

• Integrates well in hybrid systems
  - OSG-Connect
  - FermiGrid

• Used by workload/workflow managers:
  - JobSub
  - ProdAgent, CRAB
  - POMS
  - Pegasus
  - HEPCloud
Development and collaborations

• Add support for tokens
  – Use HTCondor token-auth for Glideins authentication
  – Support submission of Glideins to sites using sci-token
• Support of both Python 3 and Python 2
• Support more Frontend-like services
• Better and wider HPC support (e.g. multi-node jobs, LCF)
• Improved monitoring
• Strong collaboration with HTCondor
  – Blackhole prevention
  – Singularity invocation via HTCondor and condor_ssh_to_job
  – Use of tokens (security without x509 certificates)
Shared operations

- Factories are shared
  - Common resource on-boarding
  - Operators know the sites
  - Can solve site-dependent problems (network, middleware)

- Glideins are a common framework
  - Improvement for/from one VO are passed onto others
    - Adapting to new architectures or technologies
    - New types of containers
    - Black-hole prevention
    - Use of GPUs
  - The same Glidein can run multiple Jobs
GlideinWMS software

• Source on Github: https://github.com/glideinWMS/glideinwms
• Packaged as RPMs for RHEL6 and RHEL7
• Regular releases, compatibility and support statements
• OSG product, distributed via Open Science Grid yum repos
• Multiple project stakeholders (incl. OSG, Fermilab, CMS)

• Development team at Fermilab and UCSD/CERN
• Several groups have expertise in operating GlideinWMS infrastructure (OSG, Fermilab, CMS)

• Contact info:
  • Email: glideinwms-support@fnal.gov
  • Doc: http://glideinwms.fnal.gov/
Thank you

• Acknowledgements
  – Some of the material was contributed by Steve Timm (GlideinWMS presentation for the FNAL-UK Planning Meeting)

• Additional slides follow
  – HTCondor intro
  – Technical description
A HTCondor pool is composed of 3 pieces:

1. **Central manager**
   - Collector
   - Negotiator

2. **Submit node**
   - Schedd

3. **Execution node**
   - Job

---

S.Timm - FNAL-UK Planning Meeting  GlideinWMS
What is a Glidein?

• A Glidein is **a properly configured** execution node submitted as a **Grid** job

---

S.Timm - FNAL-UK Planning Meeting  GlideinWMS
What is GlideinWMS?

• GlideinWMS is an automated tool for submitting Glideins on demand
• Can increase or shrink the number of Glideins based on demand.
GlideinWMS Components

- Glidein – Configures and starts HTCondor execution daemons at sites
- Factory – Knows about the sites and does the submission
- Frontend – Knows about user jobs and the user requests to increase or decrease Glideins.
The Factory works with an HTCondor pool, WMS pool, to submit Glideins to different resources.

The HTCondor Glideins are pilots that launch a startd that registers on a second HTCondor pool, User pool.

User jobs are matched and execute on the resources.

The Frontend monitors the user schedds and notifies the Factory about the need for more Glideins.
Partitioning in an overlay system

- Dimensions: Cores, Memory, Disk, Lifetime
- The resource partitions its Execute nodes
- GlideinWMS further partitions the resources it receives

- E.g. 64 Cores machine, 16 or 32 cores cluster slots, 16 or 12 cores Glideins in 4 or 2 cores slots, 2 or 1 core jobs

- Issues where Glidein can help
  - Fragmentation (unused resources)
  - Flexibility (vs Complexity)
  - Under or over provisioning (overbooking or be prudent)
  - Scaling (big slots, fewer slices)