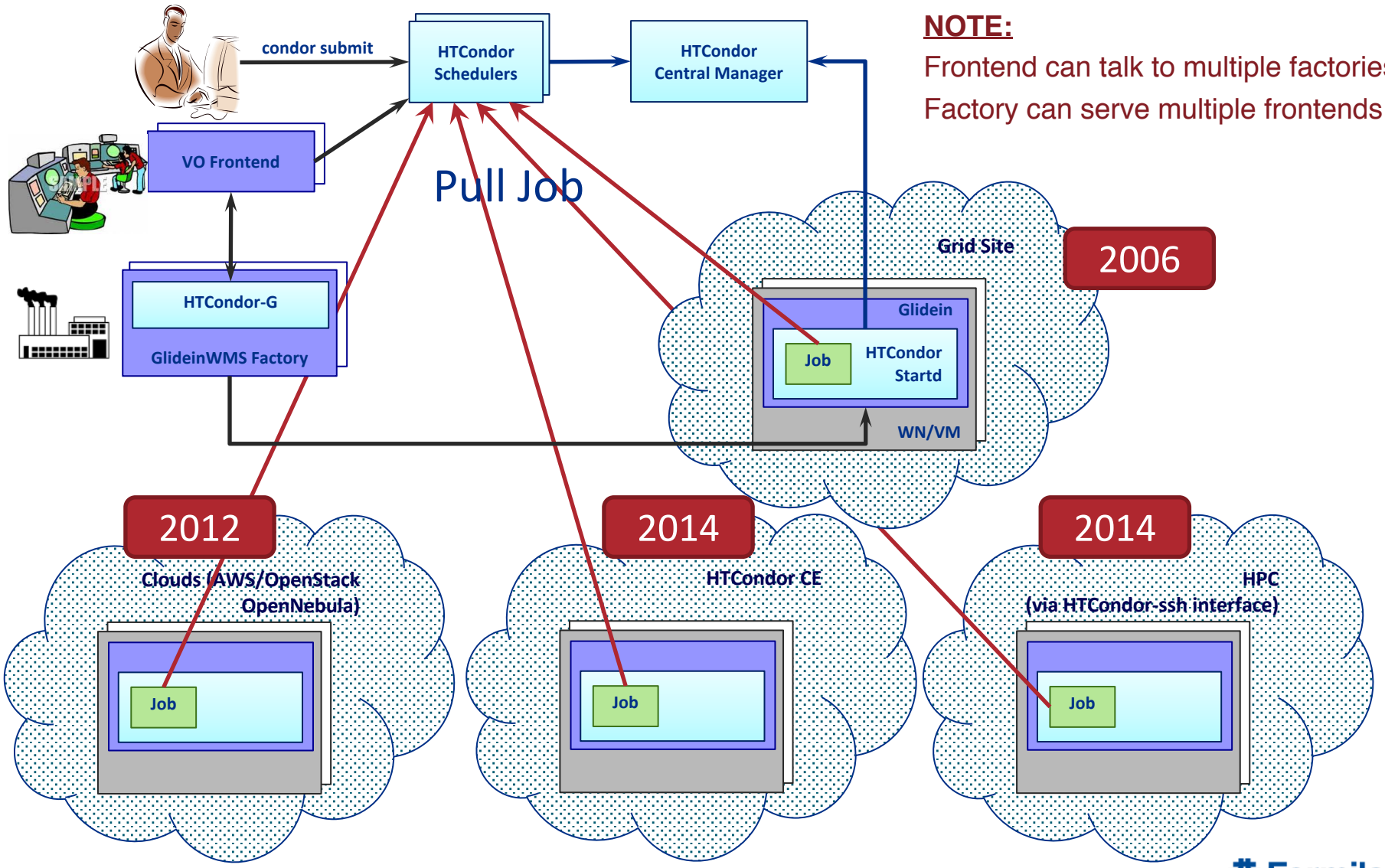
GlideinWMS

Parag Mhashilkar

OSG WMS Blueprint Meeting

February 21, 2018

GlideinWMS Architecture



NOTE:

Frontend can talk to multiple factories

Factory can serve multiple frontends

Fermilab

Target Resource Types

- Focus
  - Provision compute/worker nodes and make them available as batch slots in HTCondor pool
  - GlideinWMS can be made to support any compute resource types supported by HTCondor-G
- Currently supports
  - Grid CEs
    - HTCondor-CE
    - Globus Gatekeeper
    - CREAM-CE
    - Nordugrid
  - Clouds
    - Amazon Web Services (AWS)
    - Google Compute Engine (GCE)
    - OpenNebula using EC2 interface
    - OpenStack using EC2 interface
  - HPC
    - HPC sites supported using HTCondor ssh interface to site's batch system
    - Supports requesting multiple nodes through a single glidein
    - Does not support forming clusters for MPI jobs yet
- Support condor_annex in future

Allocation Models

- GlideinWMS does not keep track of
  - Allocations similar to that on HPCs
  - Budget and finances for Commercial Clouds
- GlideinWMS does not allocate resources, it generates resource requests based on the current demand in the system
- Glideins can be configured to support opportunistic & dedicated model
  - Opportunistic: Glidein can run any job in the queue
  - Dedicated: Glidein can run one or more class of jobs
- GlideinWMS supports multiple VOs to be serviced by a single VO Frontend
  - Resource requests are VO specific
  - VO-level isolation: Resource requests are VO specific
  - User-level isolation: Support for glexec and singularity shields multi-user jobs running in a multicore/whole node glidein

Resource Request Policies

- VO can write complex frontend configuration policies to generate resource requests
  - Requests have equal distribution across multiple sites
  - Resource request distribution is configurable and supports following models
    - Equal distribution across multiple VO credentials
    - Priority based distribution based on the VO credentials
  - Time based resource requests
    - Example:
      - Jobs, ***idle_time < 30 min***, request opportunistic resources
      - Jobs, ***30 ≦ idle_time ≦ 120 min***, request dedicated resources
      - Jobs, ***idle_time > 120 min***, request cloud resources
  - Look at the budget and allocation to make resource requests
    - Non-trivial

Resource Allocations

- Once the resource allocation decision is made, jobs start running on the worker node
- Push v/s pull model
  - Glideins are always pushed into the system
  - User jobs are pulled by running glideins
- Resource allocation is site-based
- Currently, adding new production-ready site usually requires some manual intervention and testing but can be easily automated
  - Steps to automate
    - A new production-ready site advertises relevant information to Information System
    - Factory periodically fetches this information and automatically adds the site to the list of configured sites
    - Once added, site is available for frontends to make resource requests on factory reconfiguration (few seconds – minutes)
- GlideinWMS support adding following types of resources to pool
  - Single core
  - Multicore or whole node
  - Group of nodes (for HPC sites)

Runtime Environment

- Runtime environment is highly customizable
- Glidein starts with a set of well define properties configured through the Factory and Frontend services
- GlideinWMS supports scripts available from Factory & Frontend that are executed as job prologues

Difference w.r.t. Traditional GlideinWMS Model

Adding New Resource Targets

- GlideinWMS (factory) expects that
  - The resource administrator has setup a CE in front of a batch system
  - HTCondor-G supports job submission to this CE
    - Grid or Cloud or SSH-based submission
- To successfully submit a glidein to a resource
  - Configure the GlideinWMS Factory with site's entry
    - CE endpoint and its type
    - Authentication methods to use for the glidein submission

Example: Adding HTCondor-CE Resource

```
<entry name="HTCondorCE_Example" enabled="True" schedd_name="cms-xen6.fnal.gov"
       auth_method="grid_proxy" gatekeeper="fermicloud121.fnal.gov"
       gridtype="condor" trust_domain="bosco"
       verbosity="std" work_dir=".">
    <config>
        <max_jobs glideins="3" held="2" idle="1">
            <max_job_frontends></max_job_frontends>
        </max_jobs>
        <release max_per_cycle="20" sleep="0.2"/>
        <remove max_per_cycle="5" sleep="0.2"/>
        <restrictions require_voms_proxy="False"/>
        <submit cluster_size="10" max_per_cycle="100" sleep="0.2"/>
    </config>
    <allow_frontends></allow_frontends>
    <attrs>
        <!— Any additional attributes for customization -->
        <attr name="NERSC_PBS_Site" const="True" glidein_publish="True"
              job_publish="True" parameter="True" publish="True"
              type="string" value="HTCondorCE_Sample_Site"/>
    </attrs>
    <files></files>
    <infosys_refs></infosys_refs>
    <monitorgroups></monitorgroups>
</entry>
```

Example: Adding EC2 Cloud Resource

```
<entry name="Amazon_Example" enabled="True" schedd_name="cms-xen6.fnal.gov"
       auth_method="key_pair+vm_id+vm_type" gridtype="ec2" trust_domain="Cloud"
       gatekeeper="https://us-east-1.ec2.amazonaws.com"
       verbosity="std" work_dir=".">
    <config>
        <max_jobs glideins="3" held="2" idle="1">
            <max_job_frontends></max_job_frontends>
        </max_jobs>
        <release max_per_cycle="20" sleep="0.2"/>
        <remove max_per_cycle="5" sleep="0.2"/>
        <restrictions require_voms_proxy="False"/>
        <submit cluster_size="10" max_per_cycle="100" sleep="0.2"/>
    </config>
    <allow_frontends></allow_frontends>
    <attrs>
        <!— Any additional attributes for customization -->
        <attr name="GLIDEIN_Site" const="True" glidein_publish="True"
              job_publish="True" parameter="True" publish="True"
              type="string" value="Amazon_Sample_Site"/>
    </attrs>
    <files></files>
    <infosys_refs></infosys_refs>
    <monitorgroups></monitorgroups>
</entry>
```

Example: Adding HPC Resource

```
<entry name="HPC_Example" enabled="True" schedd_name="cms-xen6.fnal.gov"
       auth_method="key_pair" gatekeeper="cmsuser@cavergrid.nersc.gov"
       gridtype="batch pbs" trust_domain="bosco"
       verbosity="std" work_dir=".">
    <config>
        <max_jobs glideins="3" held="2" idle="1">
            <max_job_frontends></max_job_frontends>
        </max_jobs>
        <release max_per_cycle="20" sleep="0.2"/>
        <remove max_per_cycle="5" sleep="0.2"/>
        <restrictions require_voms_proxy="False"/>
        <submit cluster_size="10" max_per_cycle="100" sleep="0.2"/>
    </config>
    <allow_frontends></allow_frontends>
    <attrs>
        <!— Any additional attributes for customization -->
        <attr name="NERSC_PBS_Site" const="True" glidein_publish="True"
              job_publish="True" parameter="True" publish="True"
              type="string" value="HPC_Sample_Site"/>
    </attrs>
    <files></files>
    <infosys_refs></infosys_refs>
    <monitorgroups></monitorgroups>
</entry>
```

Accounting & Monitoring

- GlideinWMS accounting information is available through HTCondor classads
- Monitoring information (budget, HPC allocations, CPU hour usage, etc)
  - Available through the HTCondor classad mechanism
  - Available as xml format through web server
- Information can be fed into the graphite or influxdb for creation of custom monitoring dashboards