

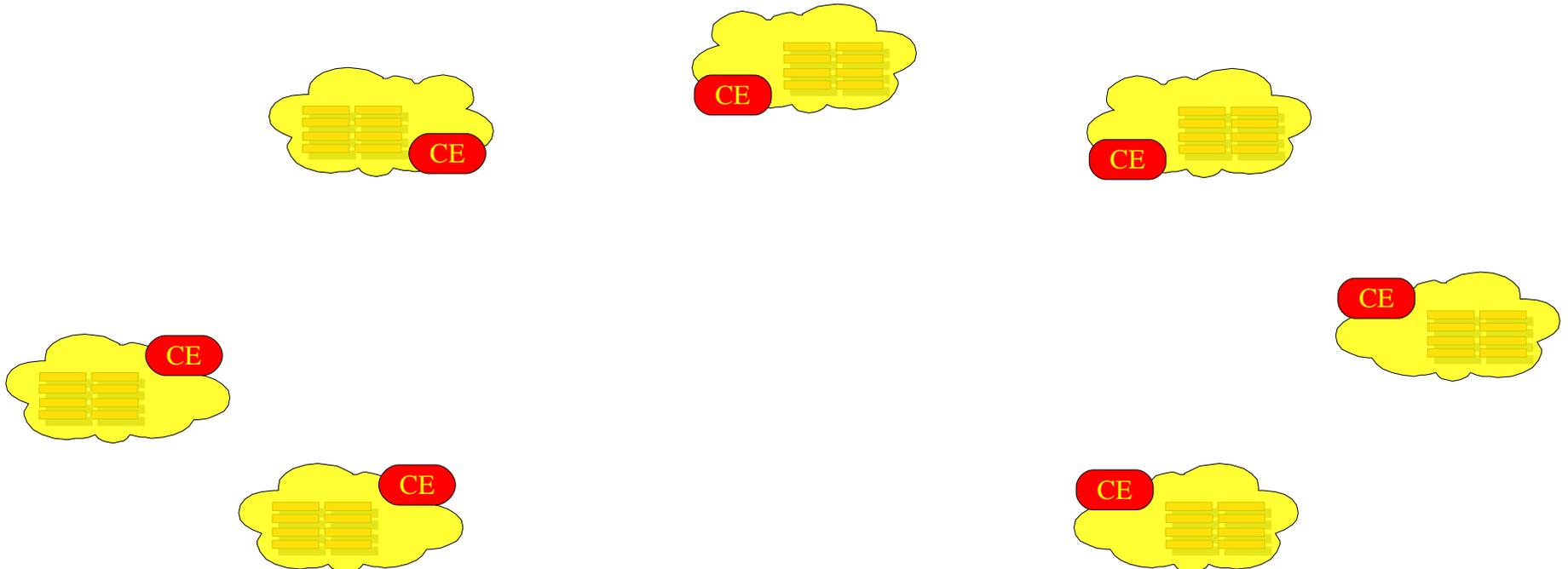
Pilots and the Grid

The CMS glideinWMS

by Igor Sfiligoi

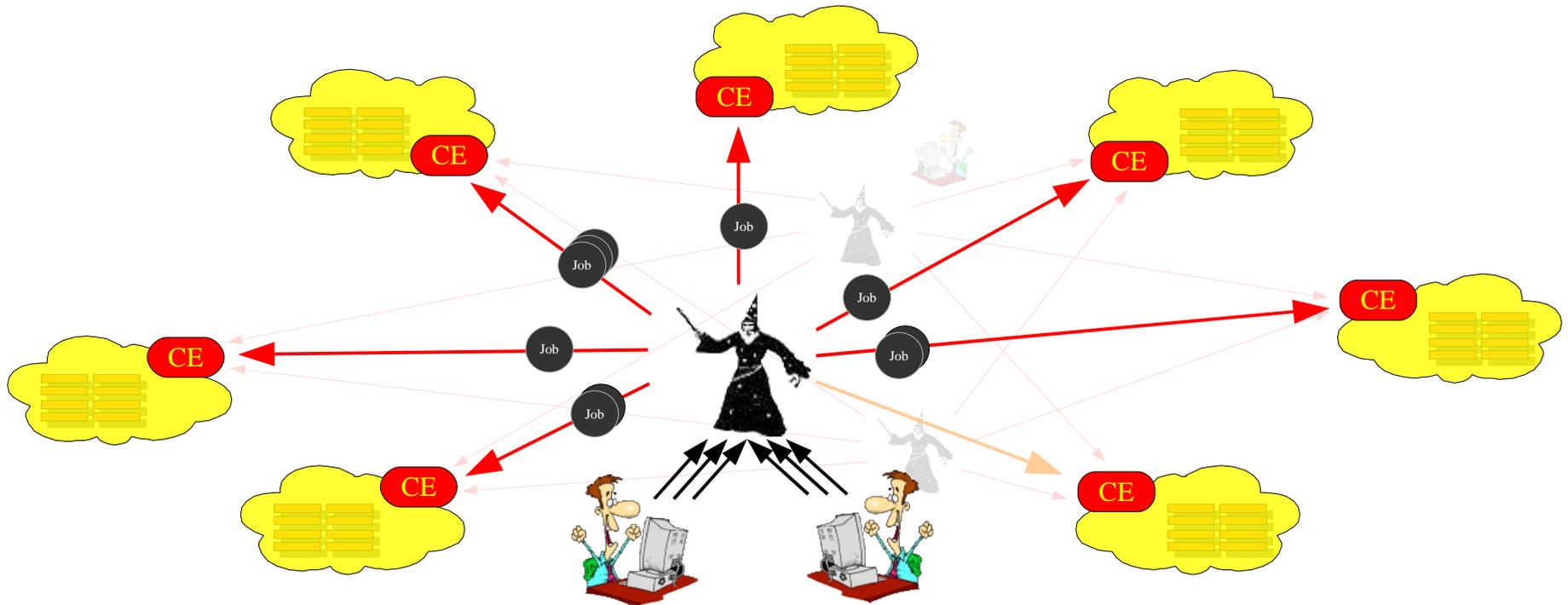
Topology of the Grid

- A loose aggregation of resources
 - Every site essentially independent



Running jobs on the Grid

- A higher level resource selection mechanism needed to efficiently use resources



Why use pilots on the Grid?

- The probability of at least one resource being broken at any given time is very high
 - Pilots can verify the resource before pulling a user job
- Difficult to have complete and reliable information about the resources
 - Pilots can do full matchmaking based on complete info
- Managing priorities inside the resource pools difficult if not impossible
 - Pilot aggregator can internally manage priorities

glideinWMS

The Condor based pilot WMS

Why using Condor for pilots?

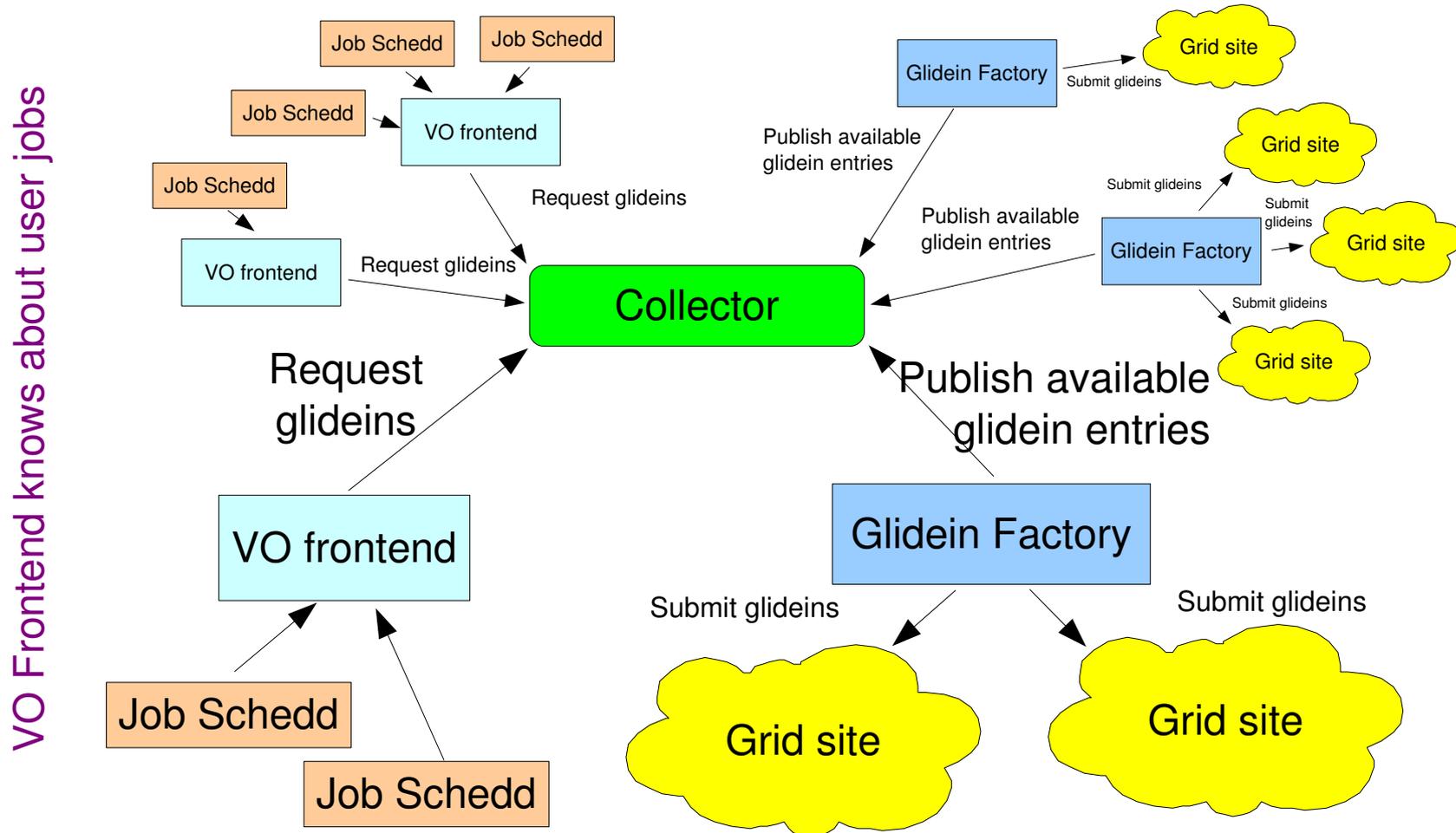
- Condor architecture distributed by design
 - Started as a project to gather unused CPU cycles
 - A concept similar to the Grid paradigm
- Condor is a very widely used system
 - Mature and well known
- Has an active development team
 - ... that listen to their users
 - A new condor release every few months

How do we create a WMS out of it?

- Glideins should be submitted on demand
 - The WMS must know about the characteristics of the user jobs
- Different Grid sites may need different configurations
 - The WMS must know the details of the Grid sites

Divide et impera

- Split the task between different processes

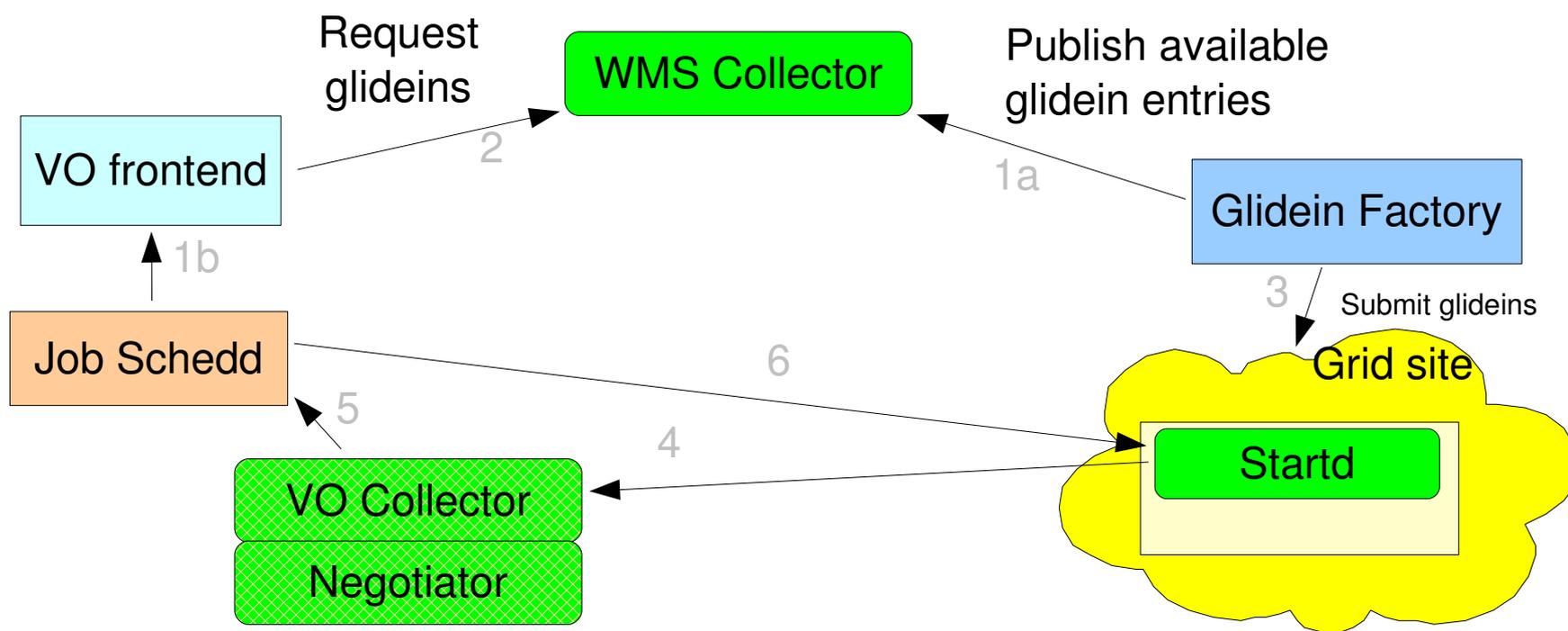


Divide et impera (2)

- VO frontend
 - User/VO specific
 - Customizable to VO specific needs
- Glidein Factory
 - Completely generic
 - The same factory could serve several VOs

Communication between processes

- Using standard Condor Class-Ads
 - The Collector defines the WMS
 - Security implemented at Condor level
 - This Collector is usually different than the VO Collector



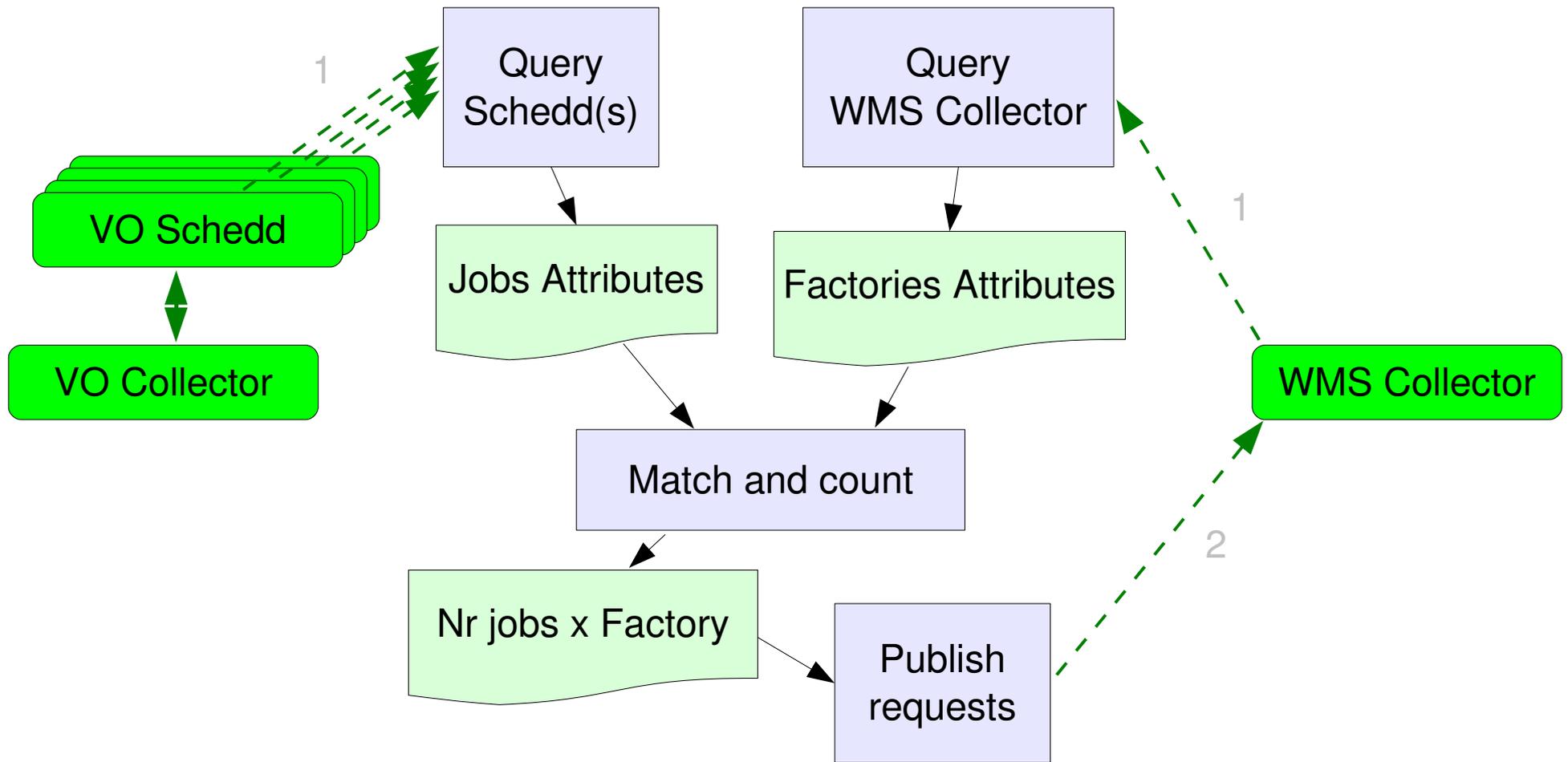
Communication between processes⁽²⁾

- Each Glidein Factory Entry Point publishes:
 - CE attributes
 - list of parameters it accepts
- Each VO Frontend replies a ClassAd containing:
 - The target Entry Point
 - VO parameters (a subset of the above)
 - Number of idle glideins to keep in the queue
 - In the future, max running glideins and max submission rate

```
Published classad
MyType="glidefactory"
Name="entry@factory"
FactoryName="factory"
GlideinName="entry"
Attribute1="..."
...
AttributeN="..."
GlideinParamXXX="val1"
...
GlideinParamYYY="valN"
```

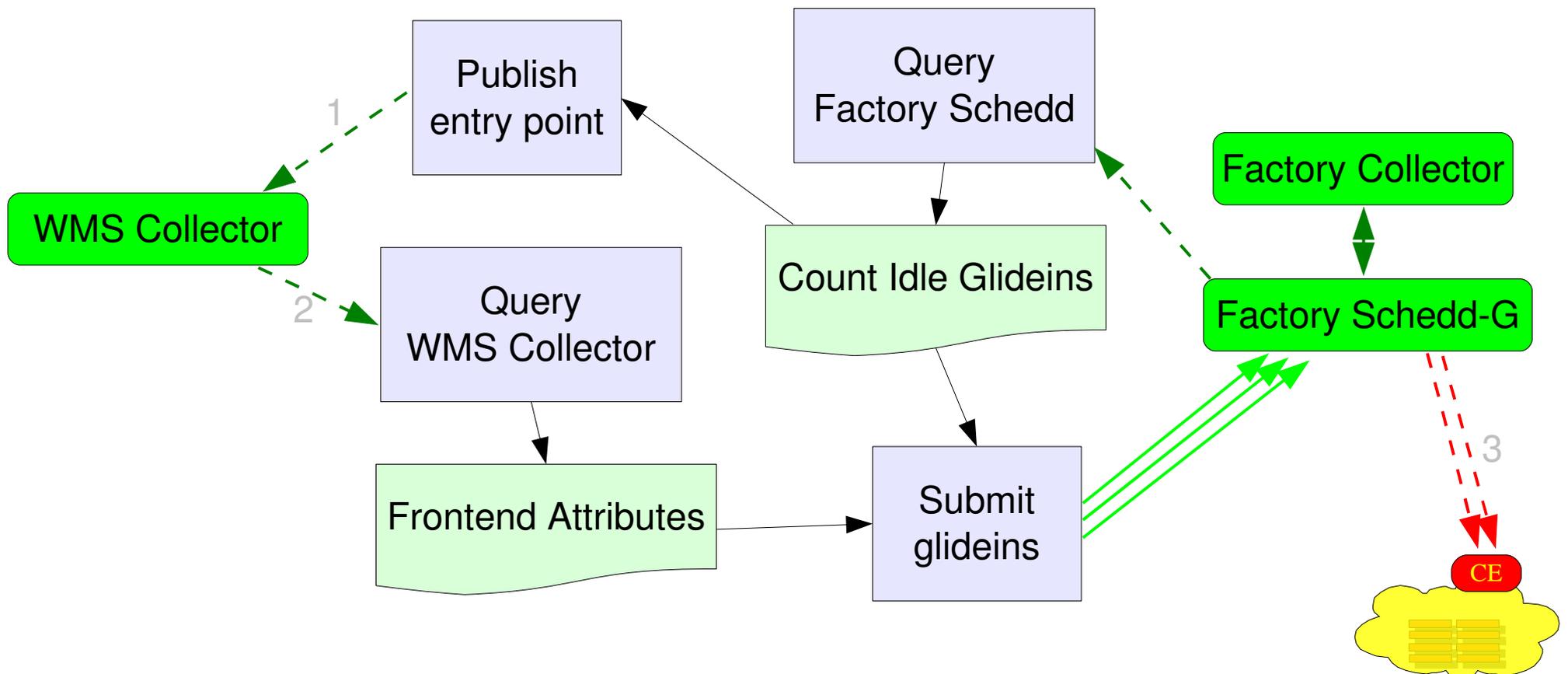
```
Published classad
MyType="glideclient"
Name="reqX@client"
ClientName="client"
ReqName="reqX"
ReqGlidein="entry@factory"
ReqIdleGlideins=nr
ReqMaxRun=nr
ReqMaxSubmitXHour=nr
GlideinParamWWW="val1"
...
GlideinParamZZZ="valY"
```

VO Frontend Acts as a Matchmaker

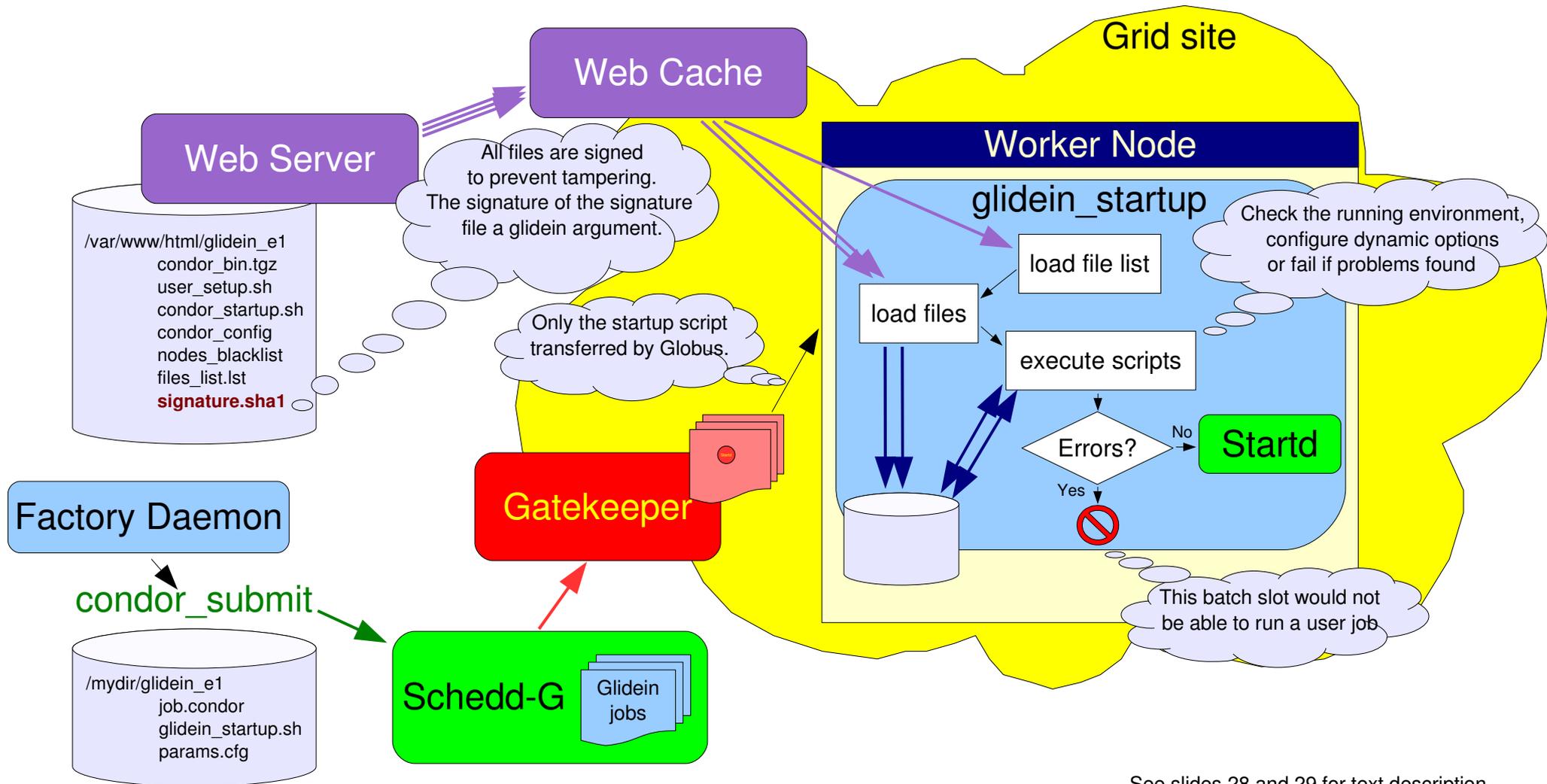


Glidein Factory blindly executes orders

- Essentially a publish-read-submit loop



Glidein implementation



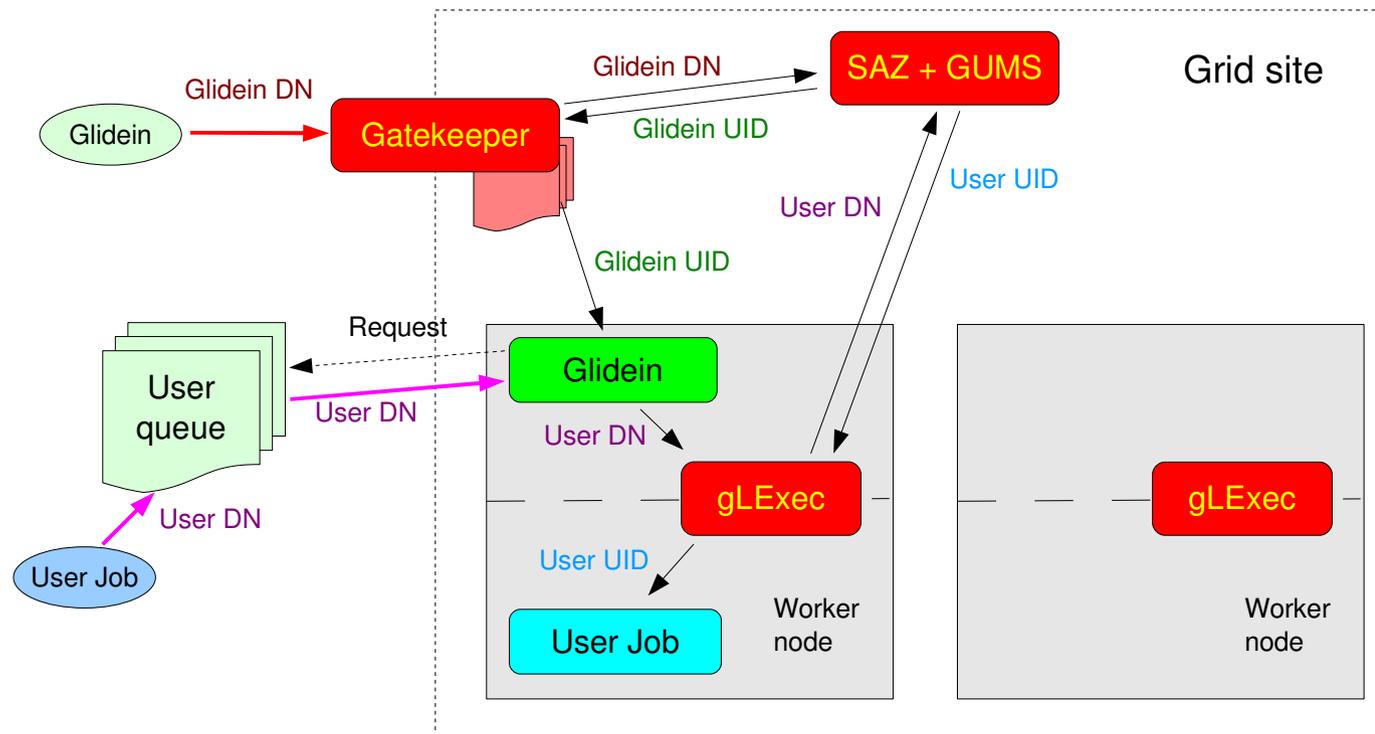
See slides 28 and 29 for text description.

Network security

- Most security handled by Condor
 - Both between
 - VO Frontend and Glidein Factory
 - Startd and Collector/Schedd
 - Condor supports strong authentication, integrity checks and encryption for privacy
- HTTP-accessed data checked via SHA1 checksums

Security on the WN

- gLExec needed to ensure proper security
 - Grid site controls who is really running
 - Pilot infrastructure shielded from the user



Backup slides

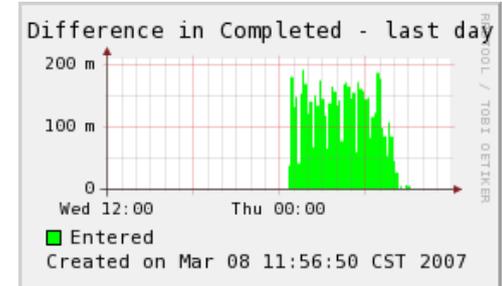
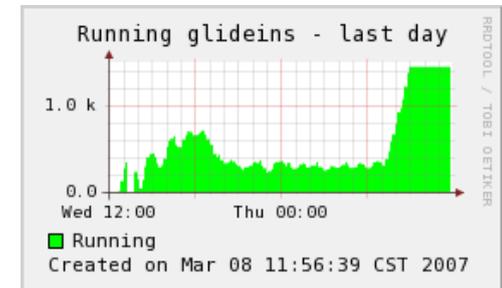
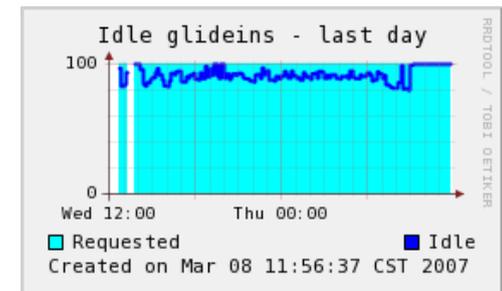
Glidein factories and Grid sites

- The current implementation requires manual configuration of the Glidein Factory
 - Factory admin must provide CE name and other site specific info
 - Thus a site can be verified before being used
- May not scale with hundreds of sites
 - Some integration with Grid information services will be needed
 - Will try to reuse ReSS information gathering tools

glideinWMS Monitoring

- Glidein Factory monitoring implemented
 - VO frontend will follow soon
- Two time dimensions:
 - Last snapshot
 - Historical archives, up to one year
- Three formats
 - XML
 - rrd databases
 - images

```
<frontends updated="1173376587">  
<frontend name="G1_v9_ultralight8@cmssrv13.fnal.gov@cms11_20">  
<Status Held="2" Idle="100" Running="1440"/>  
<Requested Idle="100"/>  
</frontend>  
<frontend name="G1_v9_ultralight8@cmssrv13.fnal.gov@cms11_21">  
<Status Held="0" Idle="100" Running="0"/>  
<Requested Idle="100"/>  
</frontend>  
<frontend name="G1_v9_ultralight8@cmssrv13.fnal.gov@cms11_22">  
<Status Held="0" Idle="100" Running="0"/>  
<Requested Idle="100"/>  
</frontend>  
</frontends>
```



Conclusion

- glideinWMS in good shape
 - Ready to be integrated with USCMS production and analysis tools
 - USATLAS Panda expressed interest in using it
- Counting on having gLExec deployed with the next OSG release
- With more real-world experience will certainly come new requirements

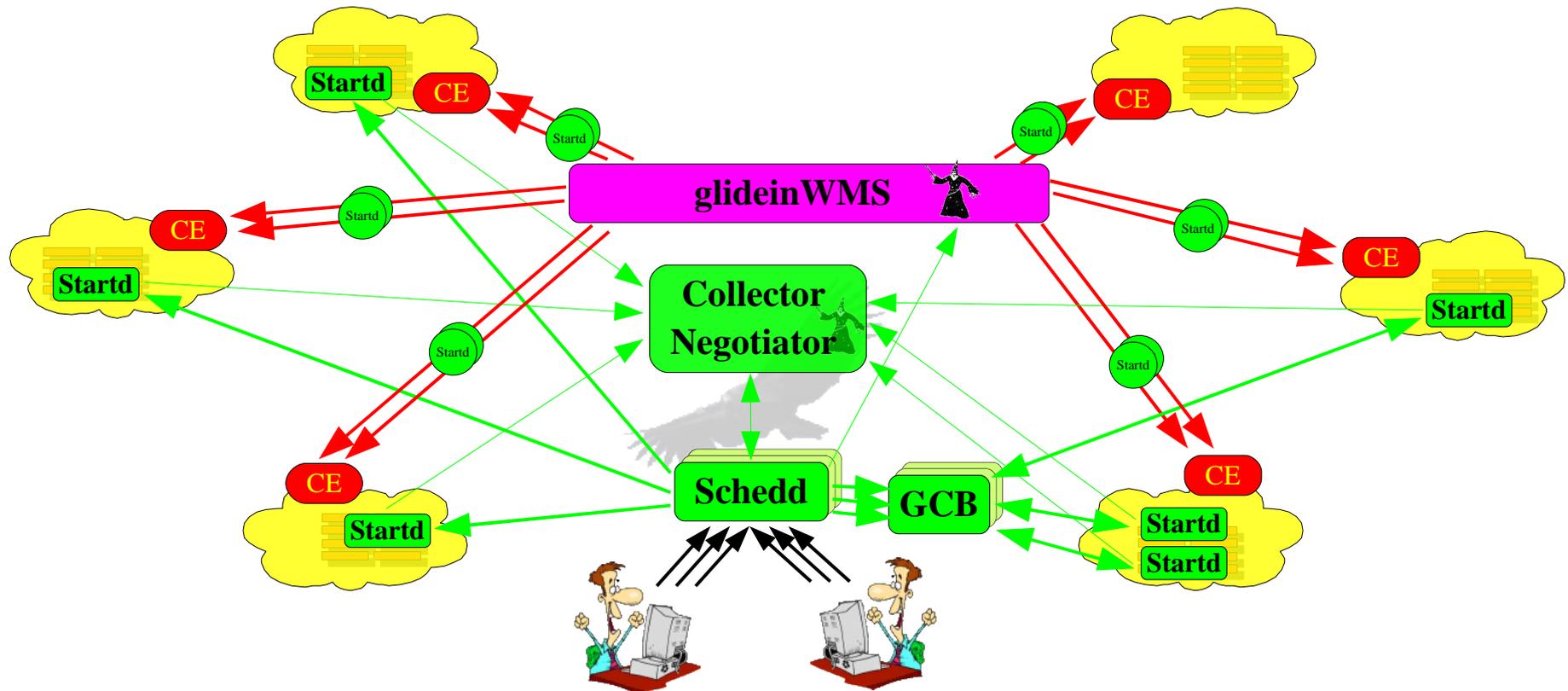
Backup slides

Project overview

- Sponsored by USCMS
- Effort involved: 0.25FTE (I. Sfiligoi)
- Currently being used with USCMS Test Harness only
 - Evaluating feasibility and scalability
- About to start integration with other USCMS tools
 - Evaluate feasibility and scalability

The glideinWMS

- Essentially a standard Condor pool, with startds started in a dynamic way



Glidein factory ClassAd

Published classad

```
MyType="glidefactory"  
Name="entry@factory"  
FactoryName="factory"  
GlideinName="entry"  
Attribute1="..."  
...  
AttributeN="..."  
GlideinParamXXX="val1"  
...  
GlideinParamYYY="valN"
```

One ClassAd x Entry point

Attributes describe the glidein

like:

ARCH="INTEL", MaxHours=72, Site="Florida"

Parameters set glidein parameter defaults

like:

CONDOR_HOST="UNDEFINED", SEC_DEFAULT_ENCRYPTION=OPTIONAL
MinDisk=16G, CheckFilesExist="/tmp/CMS,\$DATA/OSG"

Frontend ClassAd

Published classad

```
MyType="glideclient"  
Name="reqX@client"  
ClientName="client"  
ReqName="reqX"  
ReqGlidein="entry@factory"  
ReqIdleGlideins=nr  
ReqMaxRun=nr  
ReqMaxSubmitXHour=nr  
GlideinParamWWW="val1"  
...  
GlideinParamZZZ="valY"
```

Each client publishes
one ClassAd x Entry point

Targets a specific entry point

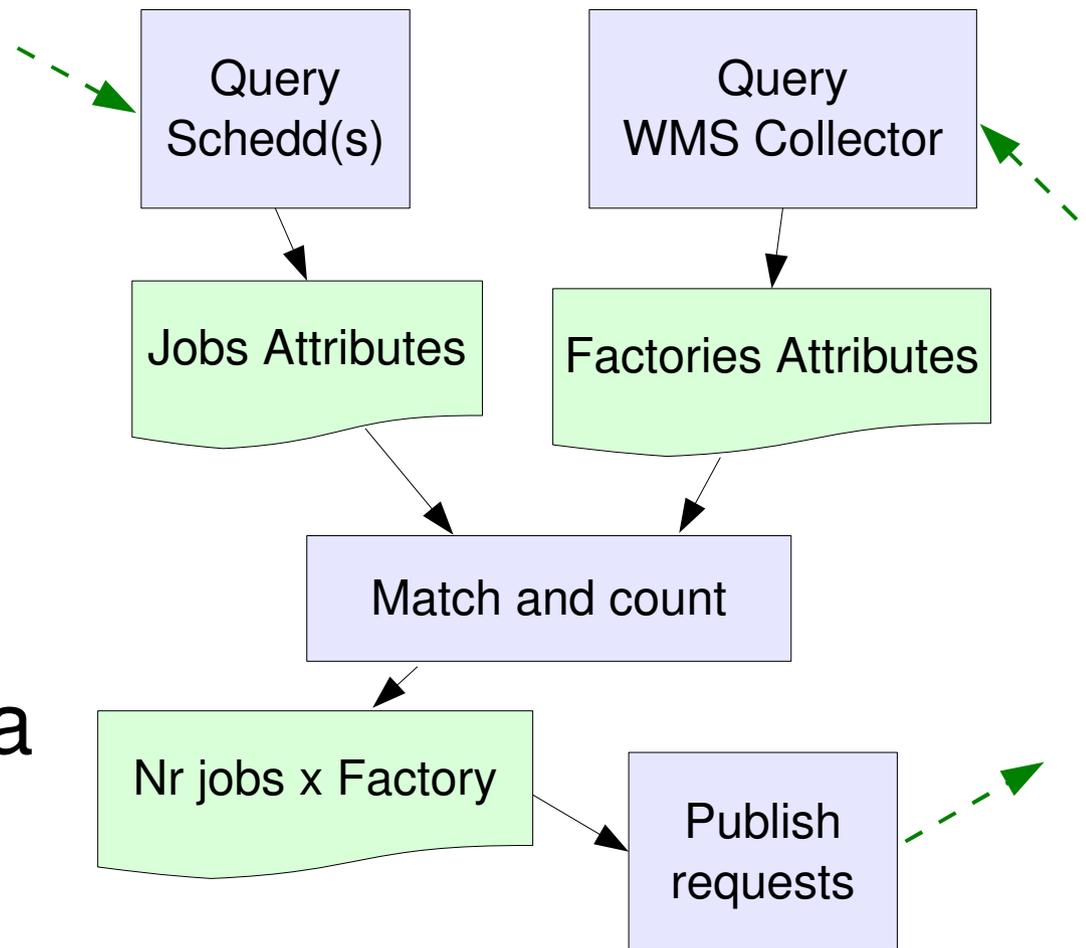
Trying to keep a steady stream
of glideins starting

Limits on the amount of
glideins submitted envisioned
but not yet implemented

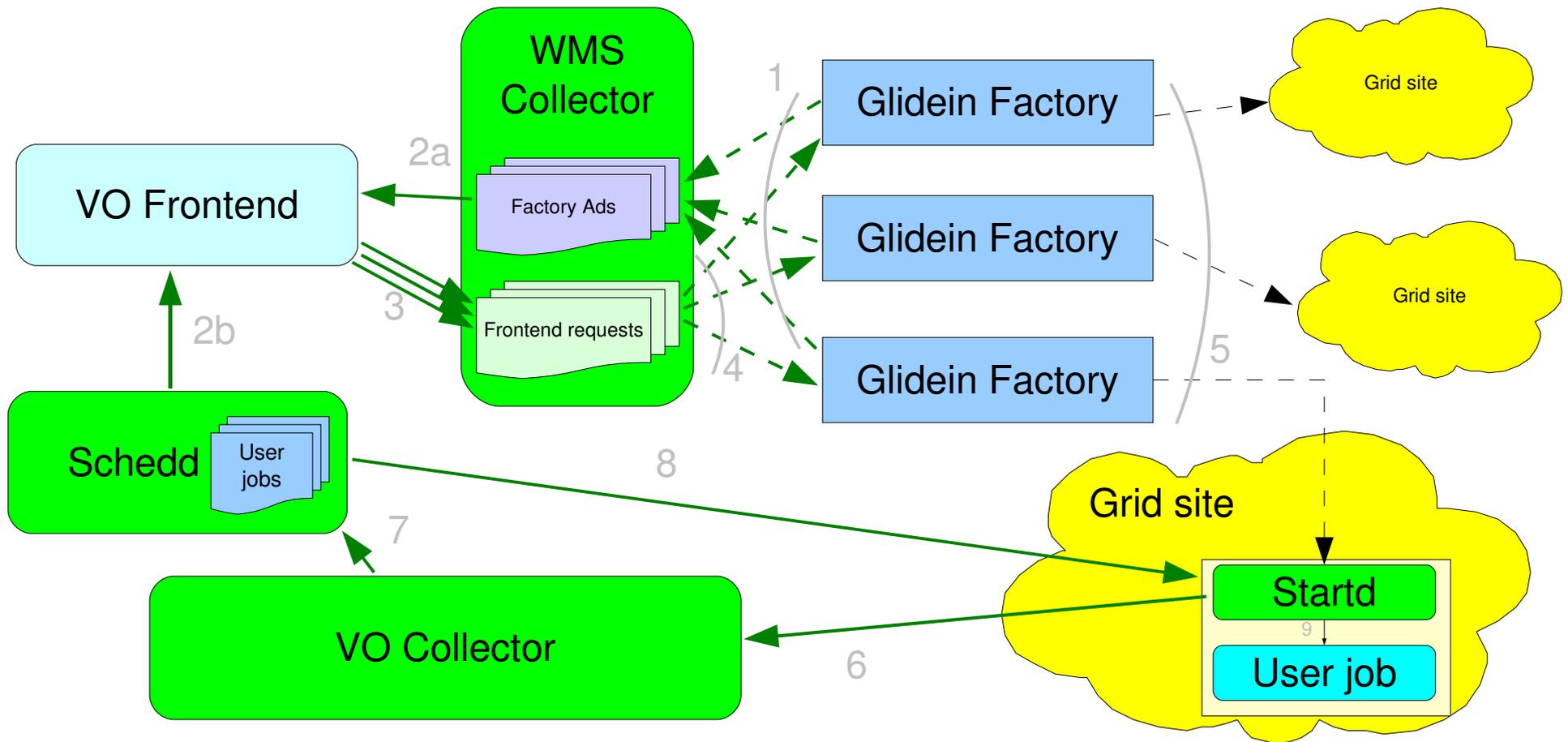
GlideParamXXX must match the names
published by the factory

VO Frontend logic

- Essentially a Matchmaker
- Will match user job attributes to factory attributes
- A scaled back count is then published as a request

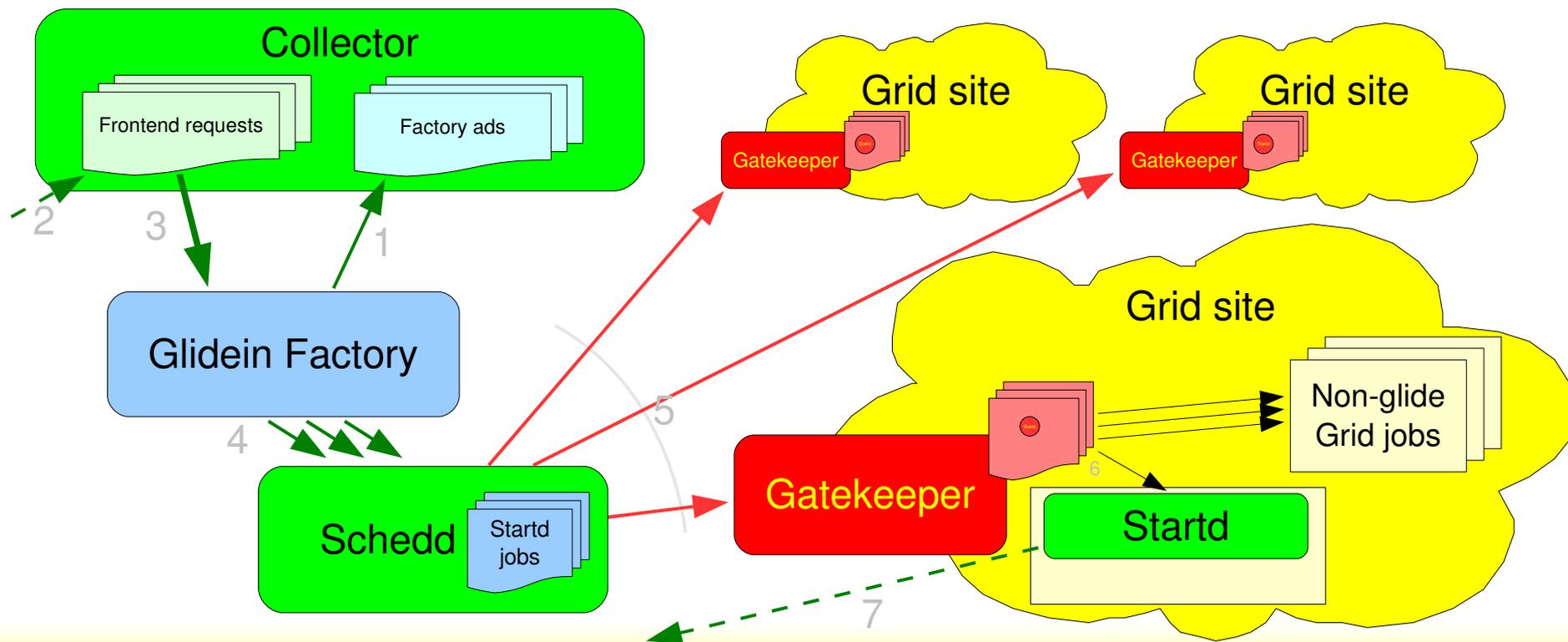


VO Frontend in the bigger picture



Glidein Factory Logic

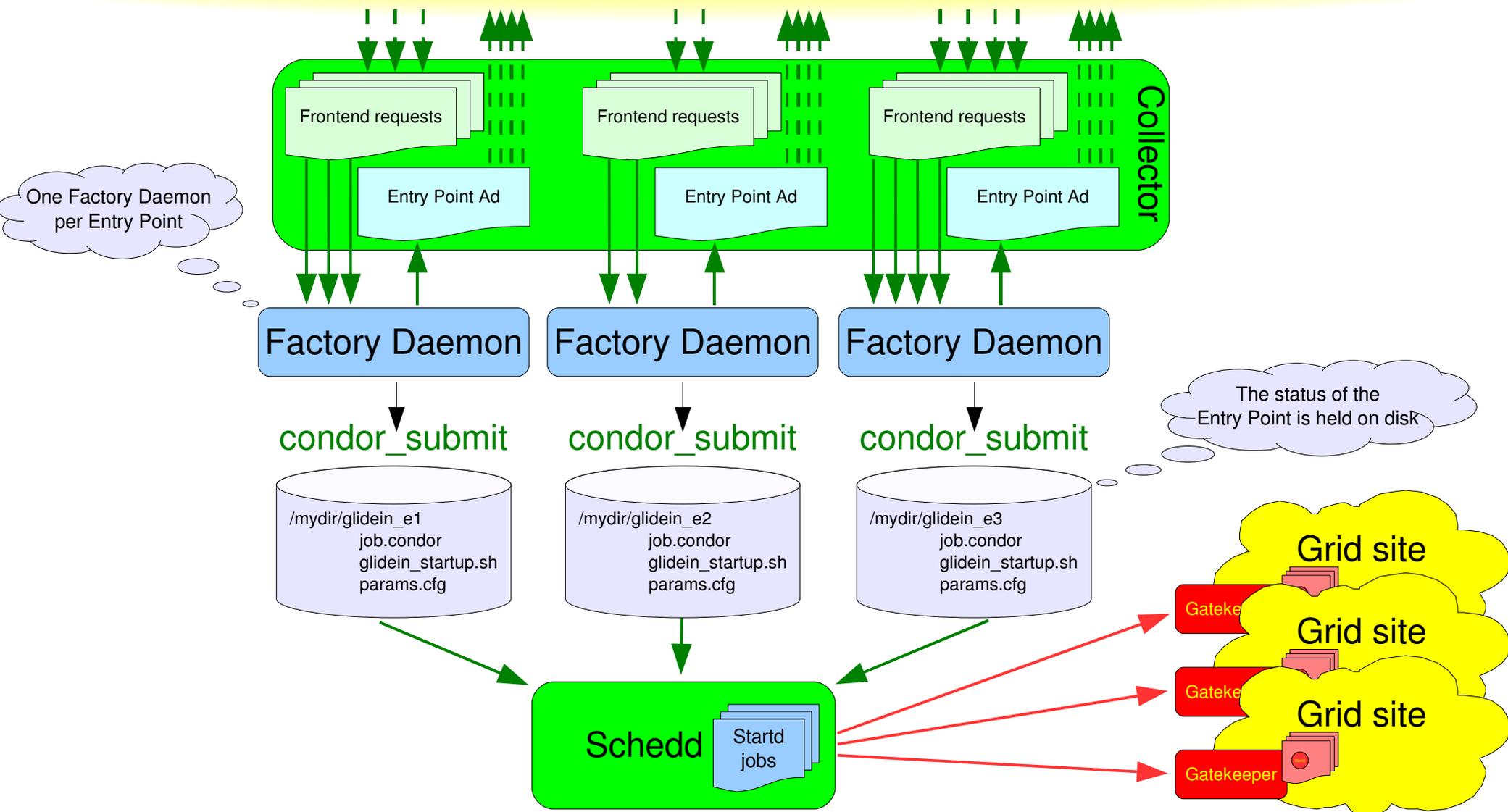
- Essentially a publish-read-submit loop
 - Blindly execute orders
 - Security implemented at Collector level



Glidein Factory Internals

- Made of multiple Entry Points
 - Each Entry Point an independent process
 - Implements the publish-read-submit loop
- Entry Points do the real job
 - The Glidein Factory is just a logical object
 - Management tools that act on the whole factory (i.e. several entry points at a time) envisioned, but not yet implemented

Entry points at a glance



Glidein implementation

- Dummy startup script
 - Just loads other files and execute the ones marked as executable
- File transfer implemented using HTTP
 - Easy cacheable, standard tools available (Squid)
 - Proven to scale, widely used in Industry
- All sensitive file transfers signed (SHA1)
 - Prevent tampering, as HTTP travels in clear

Glidein implementation (2)

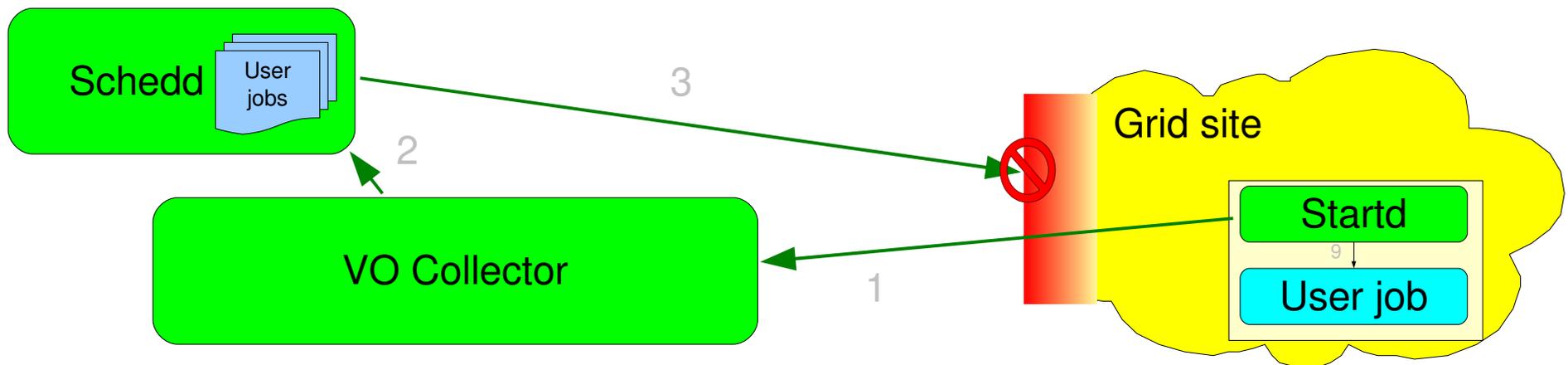
- Standard sanity checks provided
 - Disk space constraints
 - Node blacklisting
- Generic Condor configure and startup script provided, too
- Factory admins can easily add their own customization scripts (both for checks and configs)
 - Allowing Frontends to add custom scripts envisioned, but not yet implemented

Security

- Traffic between the Collector and Schedd, and the Grid sites cannot be trusted
 - WAN is insecure by definition
- Strong authentication needs to be used
 - ★ The startd has access to the service X509 proxy, so that is used for authentication
 - ★ Supported natively by Condor

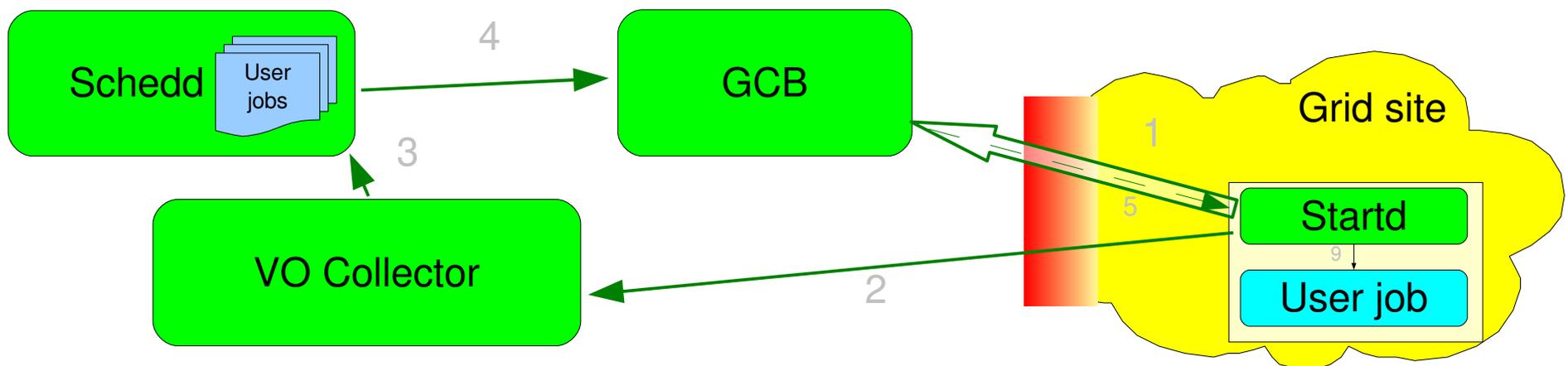
Private networks and firewalls

- Condor developed with LAN in mind
 - All daemons require bi-directional networking
- Default mode fails with firewalls and private networks
 - Incoming traffic on WNs almost never allowed



Using Condor GCB

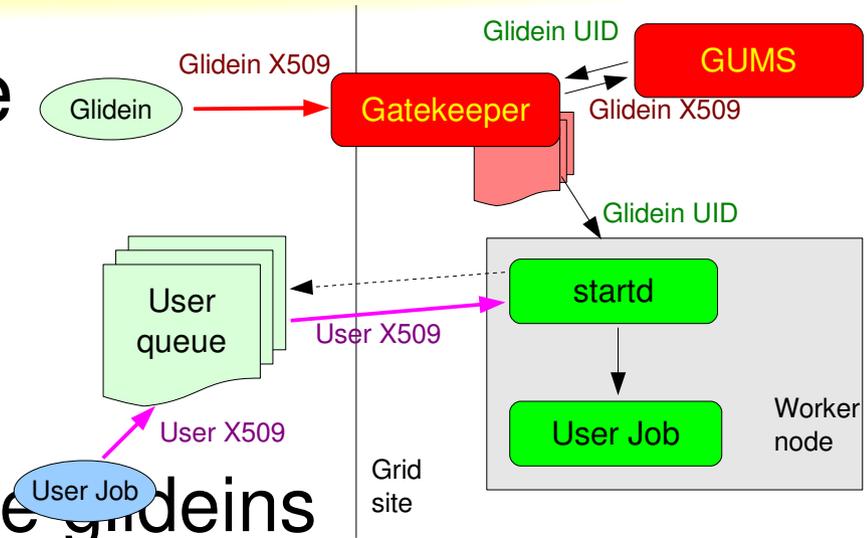
- Adding Condor GCB to the mix solves the problem
 - Startd keeps a permanent TCP connection with GCB
 - Only outgoing connectivity used from WN
- Adds complexity to the system, but is needed
 - One GCB server can handle ~2k glideins



Integration with Grid-local security infrastructure

- Default pilot operation mode breaks Grid security rules

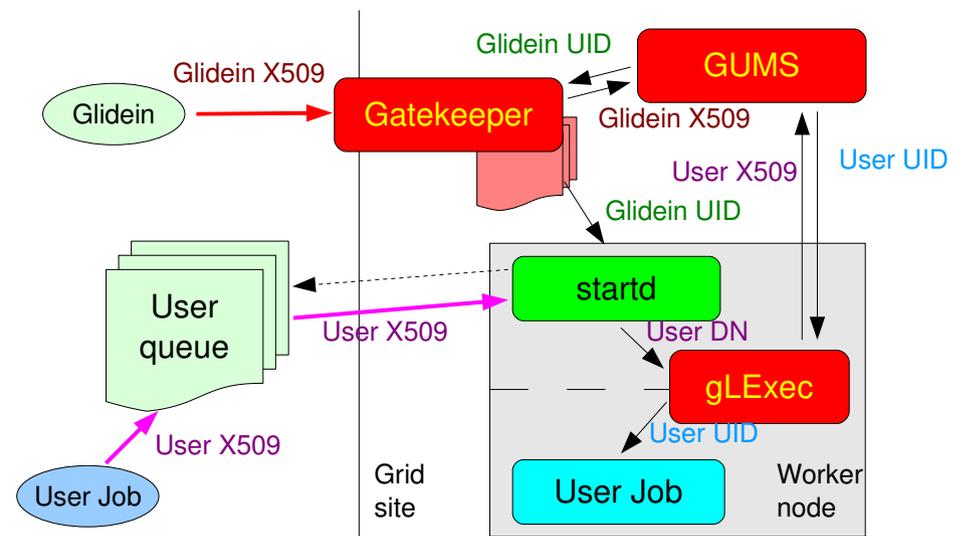
- The real user job is pulled in without local consent
- Grid site knows only about the glideins



- The default mode also a security risk for the VO
 - User jobs run under the same UID as the condor daemons
 - ♦ **Users have access to the service X509 proxy**

Condor integration with gLExec

- gLExec started to be deployed on OSG WNs
 - Will change UID based on X509 proxy
- Condor natively supports it as of v6.9.0
 - ★ Glidein factory have the necessary config script
- Solves both problems at once



Monitoring data

- When last updated
- List of all served requests, and for each request
 - What attributes were requested
 - What is the status of the requested attributes
 - Other attributes can be present, but not required

Entry Point Static Example

```
<frontends updated="1173376587">
  <frontend name="G1_v9_ultralight8@cms13.fnal.gov@cms11_20">
    <Status Held="2" Idle="100" Running="1440"/>
    <Requested Idle="100"/>
  </frontend>
  <frontend name="G1_v9_ultralight8@cms13.fnal.gov@cms11_21">
    <Status Held="0" Idle="100" Running="0"/>
    <Requested Idle="100"/>
  </frontend>
  <frontend name="G1_v9_ultralight8@cms13.fnal.gov@cms11_22">
    <Status Held="0" Idle="100" Running="0"/>
    <Requested Idle="100"/>
  </frontend>
</frontends>
```

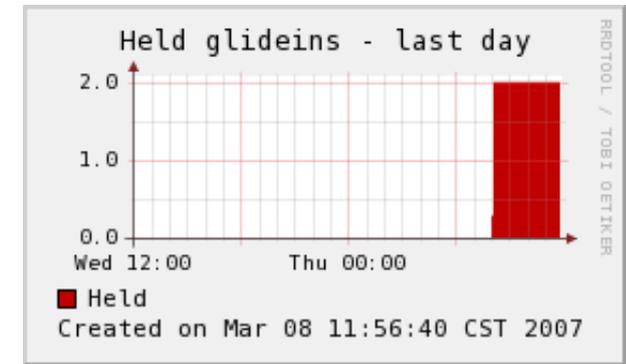
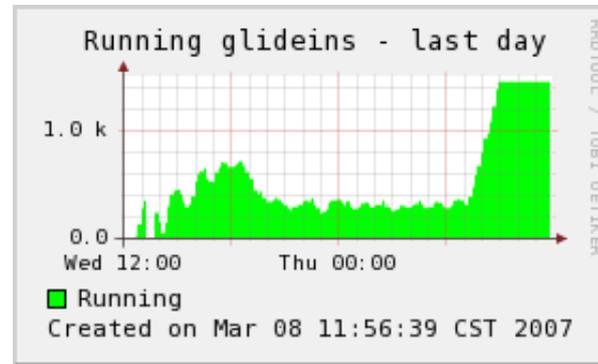
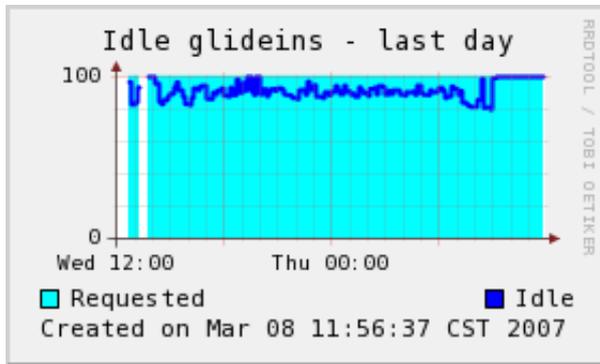
Dynamic example

Status_Attribute_Running.hour.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?> <data>
<xport>
  <meta>
    <start>1173373200</start>
    <step>300</step>
    <end>1173376800</end>
    <rows>13</rows>
    <columns>1</columns>
    <legend>
      <entry>Running</entry>
    </legend>
  </meta>
  <row><t>1173373200</t><v>1.4400000000e+03</v></row>
  <row><t>1173373500</t><v>1.4300000000e+03</v></row>
  <row><t>1173373800</t><v>1.4200000000e+03</v></row>
  <row><t>1173374100</t><v>1.4150000000e+03</v></row>
  <row><t>1173374400</t><v>1.4200000000e+03</v></row>
  <row><t>1173374700</t><v>1.4100000000e+03</v></row>
  <row><t>1173375000</t><v>1.4300000000e+03</v></row>
  <row><t>1173375300</t><v>1.4400000000e+03</v></row>
  <row><t>1173375600</t><v>1.4400000000e+03</v></row>
  <row><t>1173375900</t><v>1.4300000000e+03</v></row>
  <row><t>1173376200</t><v>1.4250000000e+03</v></row>
  <row><t>1173376500</t><v>1.4300000000e+03</v></row>
  <row><t>1173376800</t><v>NaN</v></row>
</data>
</xport>
```

Graphs

Absolute Values



Rates

