# Glidein Based WMS

**A pilot-based (PULL) approach to the Grid**

An overview

by Igor Sfiligoi

## Why use pilots on the Grid?

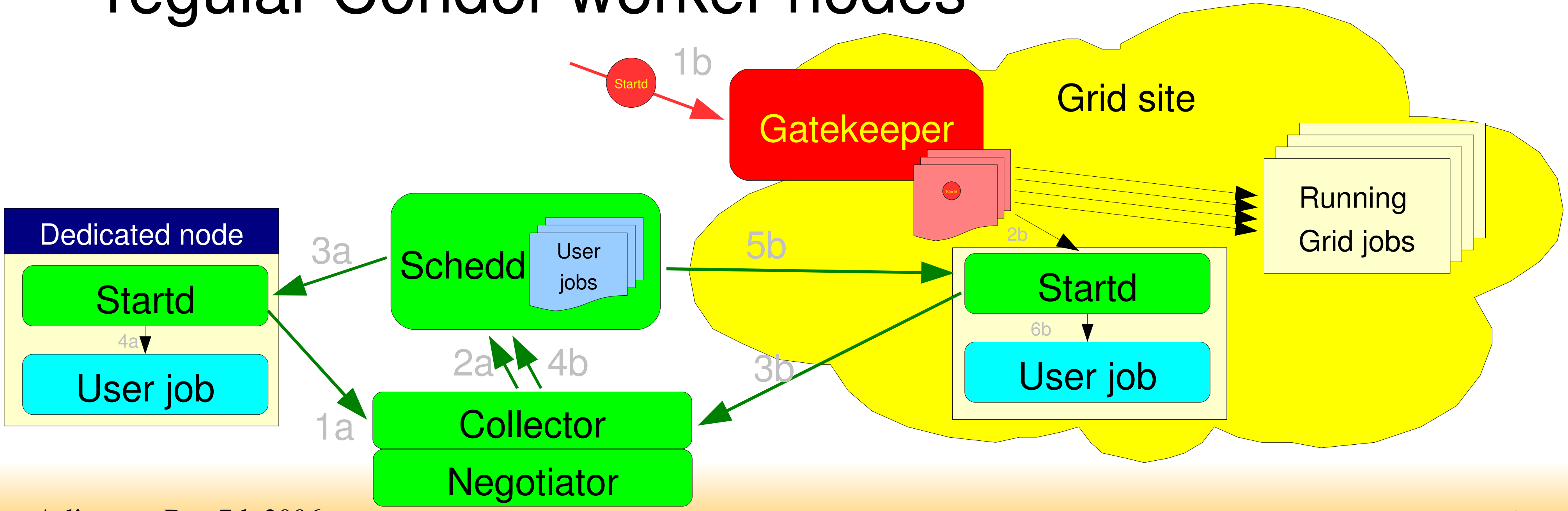
- Grid based on a loose aggregation of resources
  - Users have no say on how the resources are managed
  - The probability of at least one resource being broken at any given time is very high
- Pilots can verify the resource before pulling a user job
  - Difficult to have complete and reliable information about the resources
  - Managing priorities inside the resource pools difficult if not impossible
- Pilots can do full matchmaking based on complete info

## Why using Condor for pilots?

- 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
- Condor architecture distributed by design
  - Started as a project to gather unused CPU cycles
  - A perfect fit for the Grid world

## What are the “glideins”?

- Glideins are just properly configured, regular Condor daemons submitted as batch jobs
- From the user point of view, they look like regular Condor worker nodes

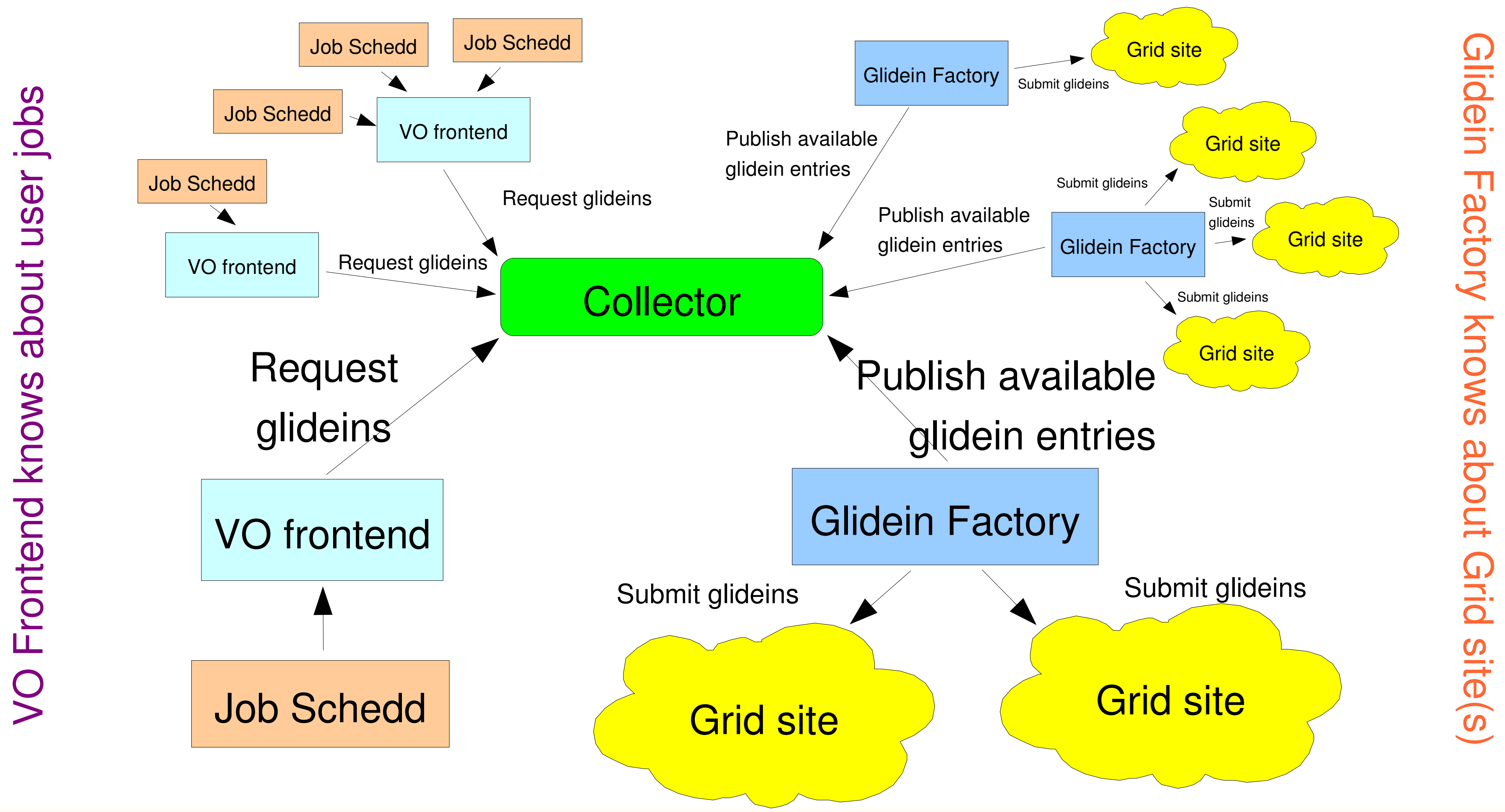


## How do we create a WMS out of it?

- 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

# Dividi et impera

- Split the task between different processes



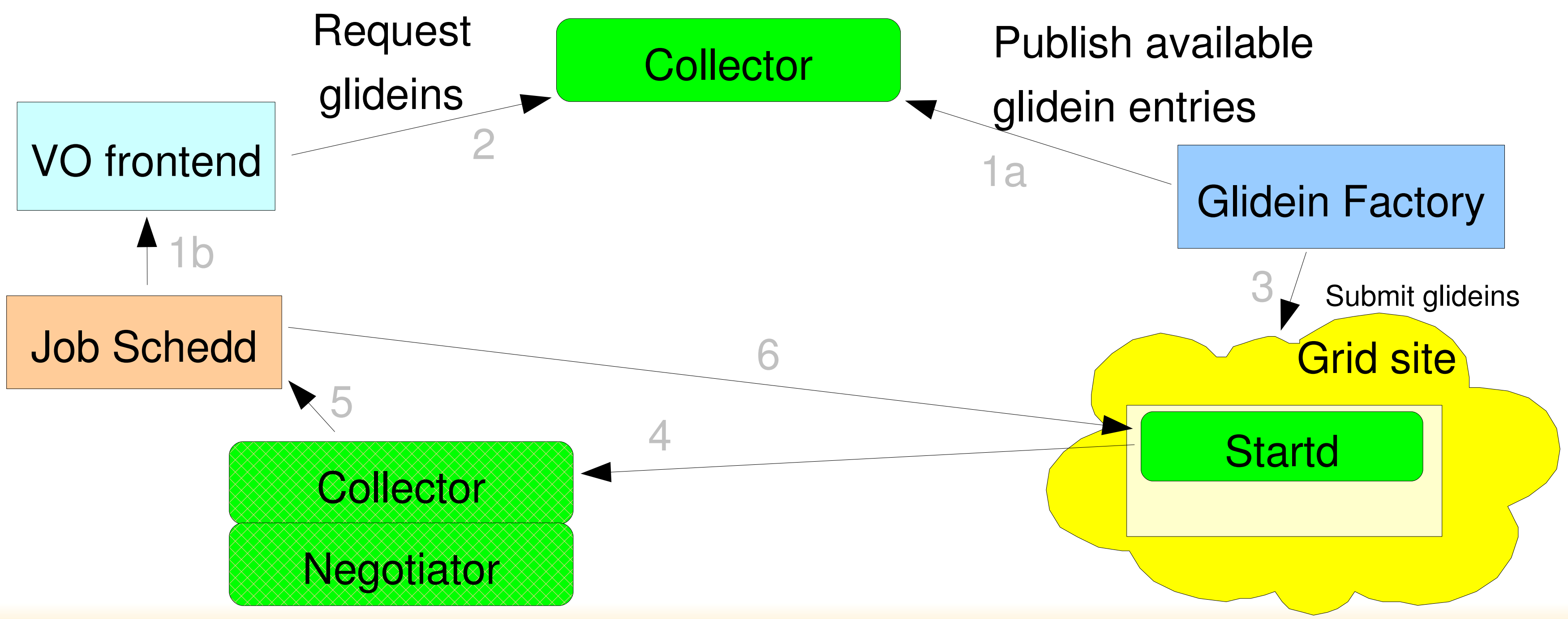
## Dividi et impera (2)

- VO frontend
  - User/VO specific
  - A reasonably generic implementation provided for simple use cases
- Customizable
  - Looks only for a known subset of jobs
  - Can process custom attributes

- Glidein Factory
  - Completely generic
  - The default implementation should suit most tasks
  - The same factory could in principle serve several VOs

## Comunication between processes

- Using standard Condor Class-Ads
  - The Collector defines the WMS
  - This Collector is usually different than the glidein Collector



# Glidein factory ClassAd

Due to Condor limitations, define also GlideinMyType

**Published classad**

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

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"
```

Due to Condor limitations, define also GlideinMyType

Trying to keep a steady stream of glideins starting

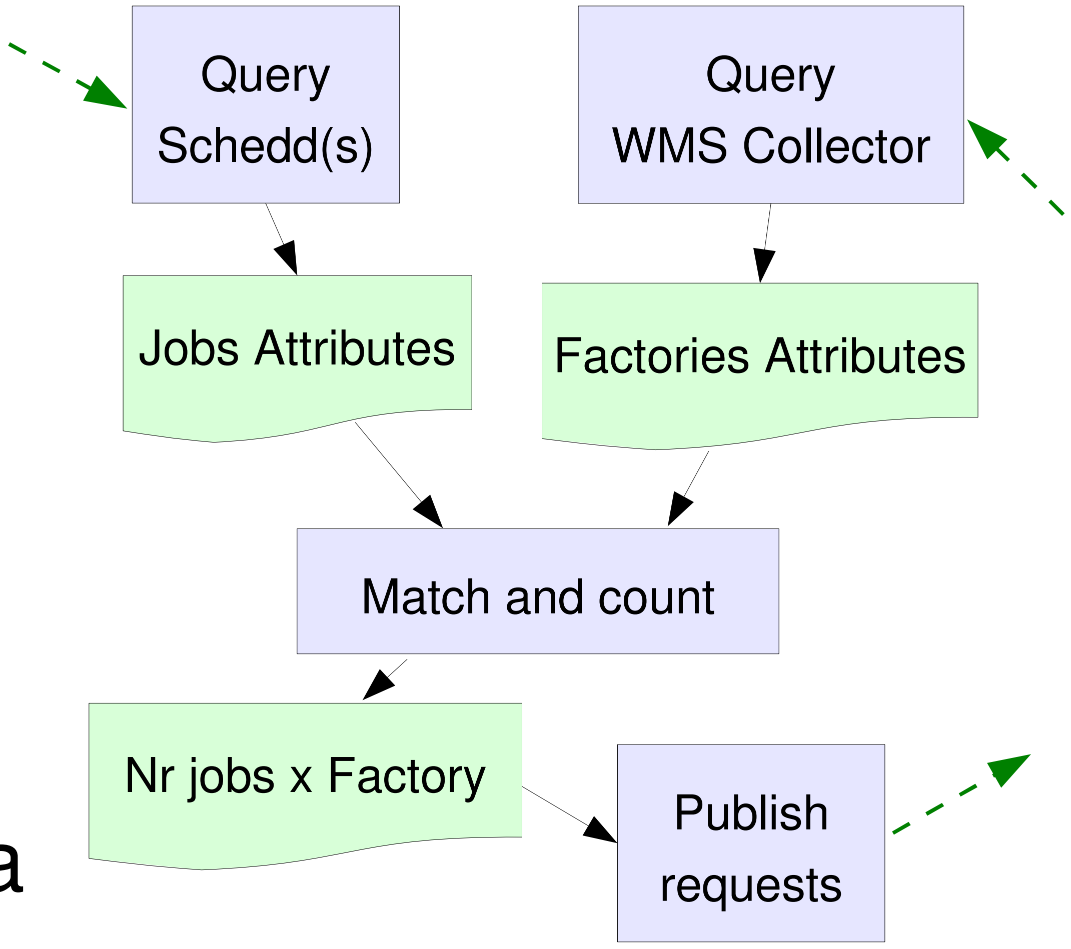
Limits on the amount of glideins submitted envisioned but not yet implemented

GlideParamXXX must match the names published by the factory

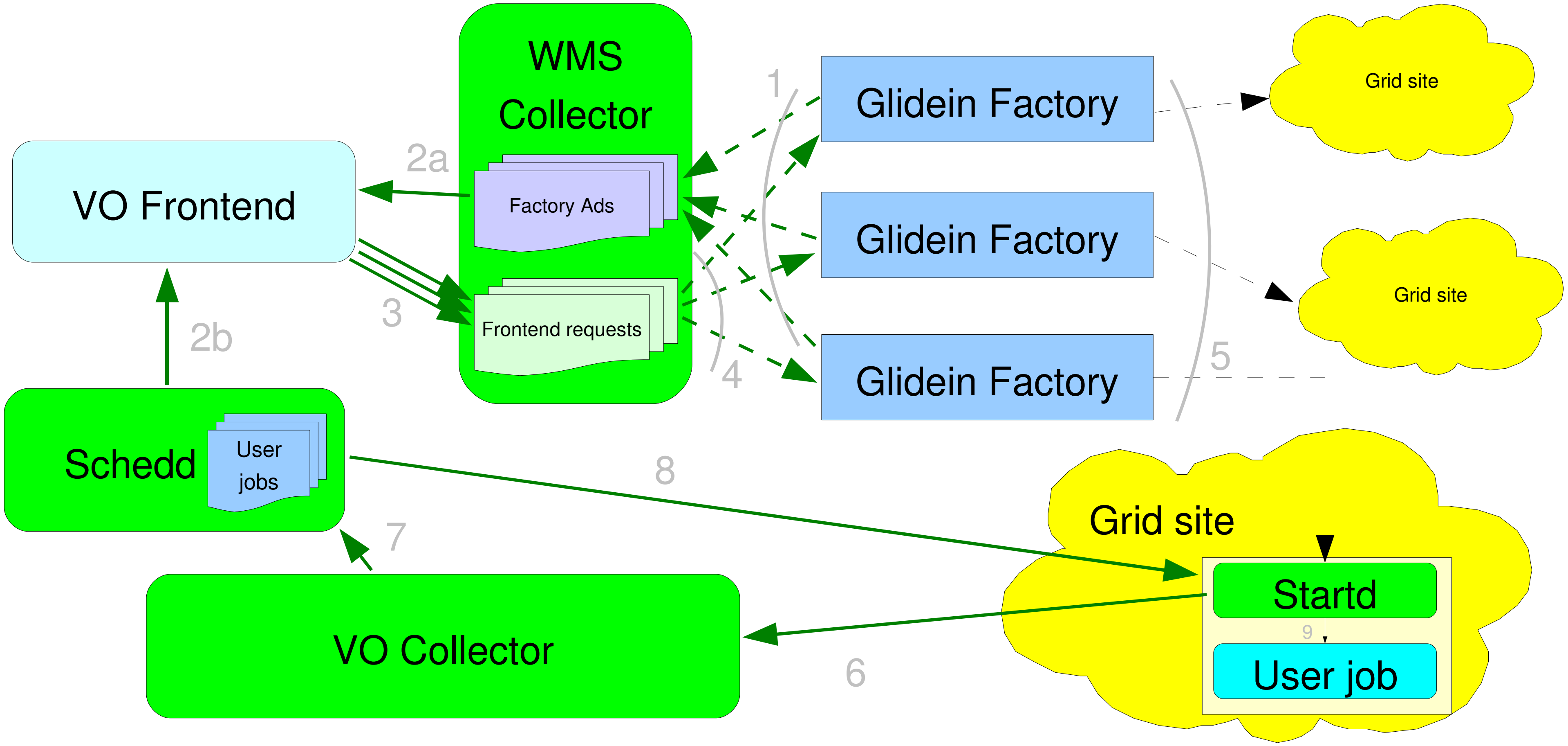
## Implementation details

# Frontend logic

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

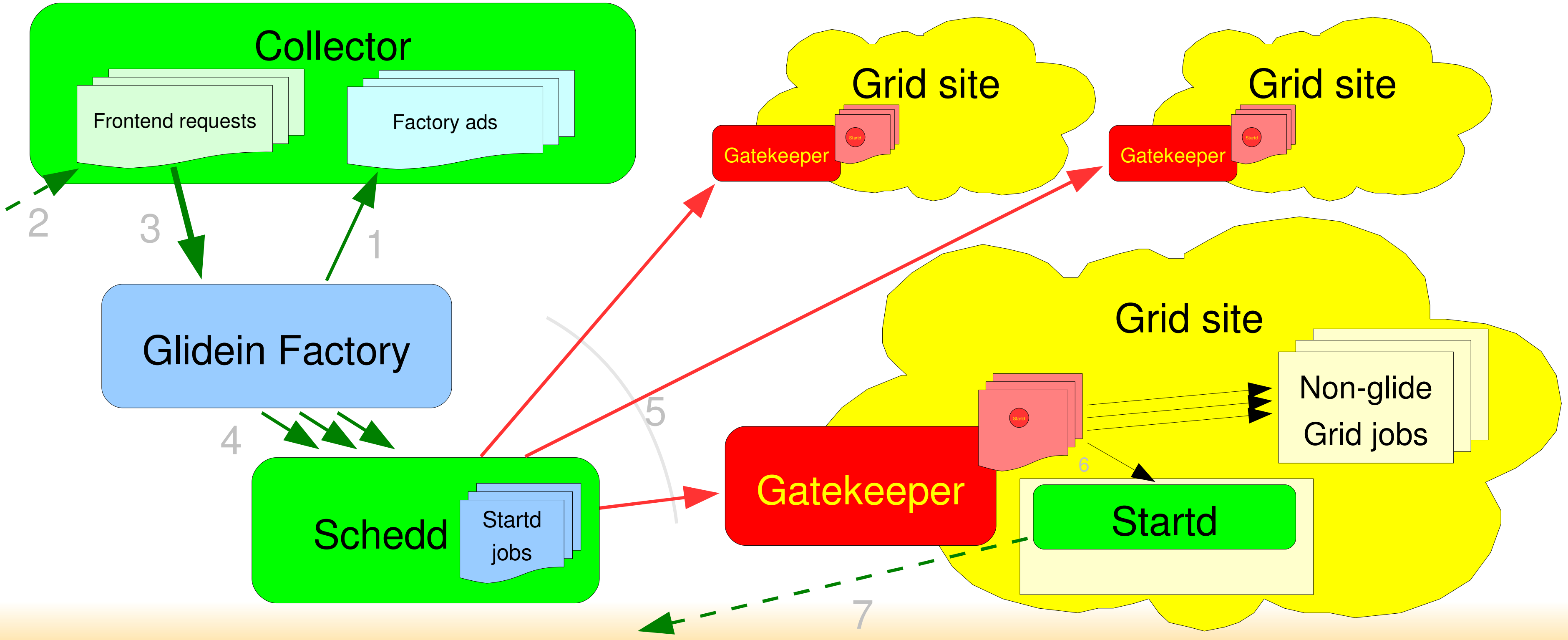


# Frontend in the big picture



# Glidein Factory Logic

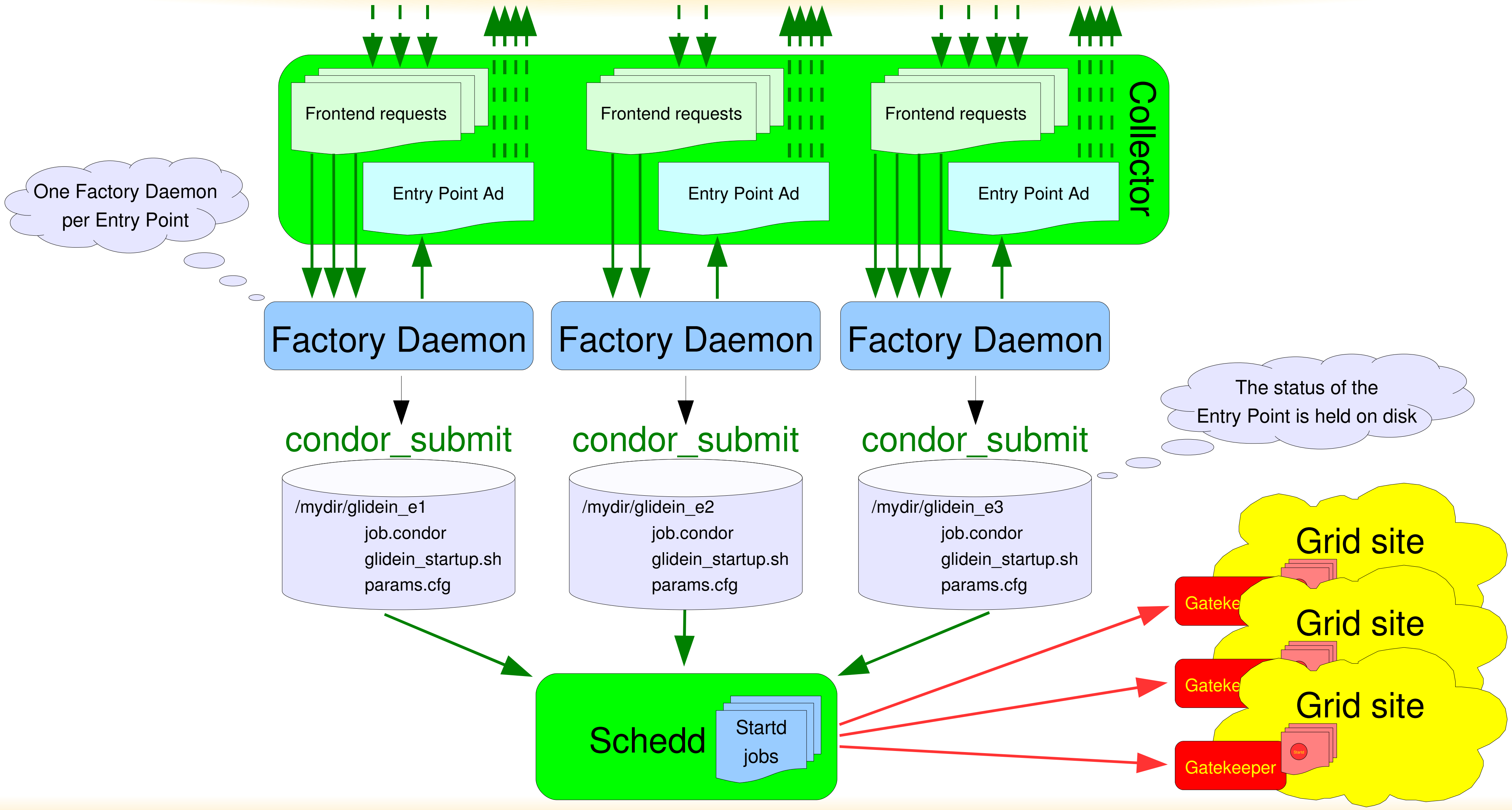
- Essentially a publish-read-submit loop
  - Blindly execute orders
  - Security implemented at Collector level



## 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



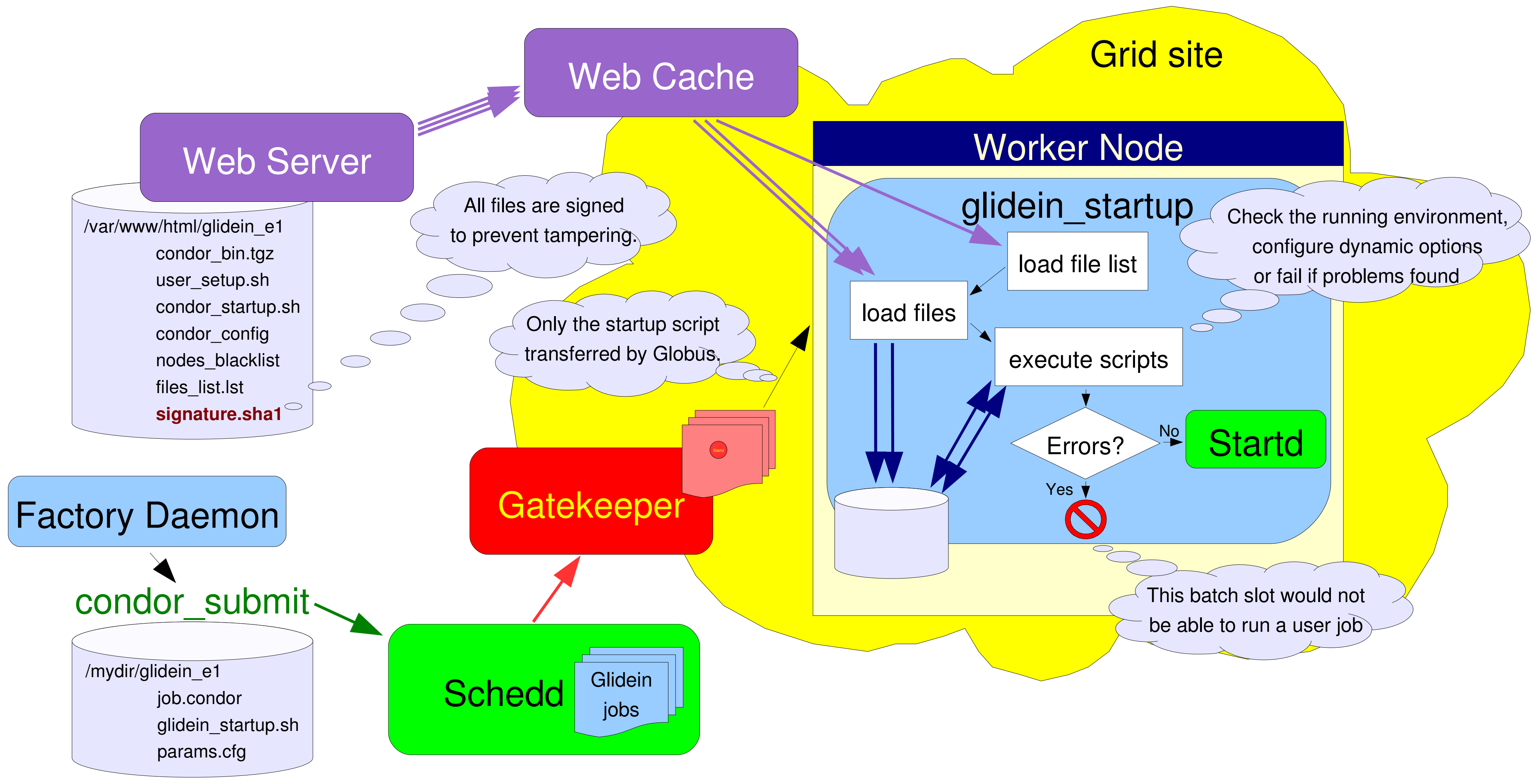
## Glidein implementation

- 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

# Glidein at a glance



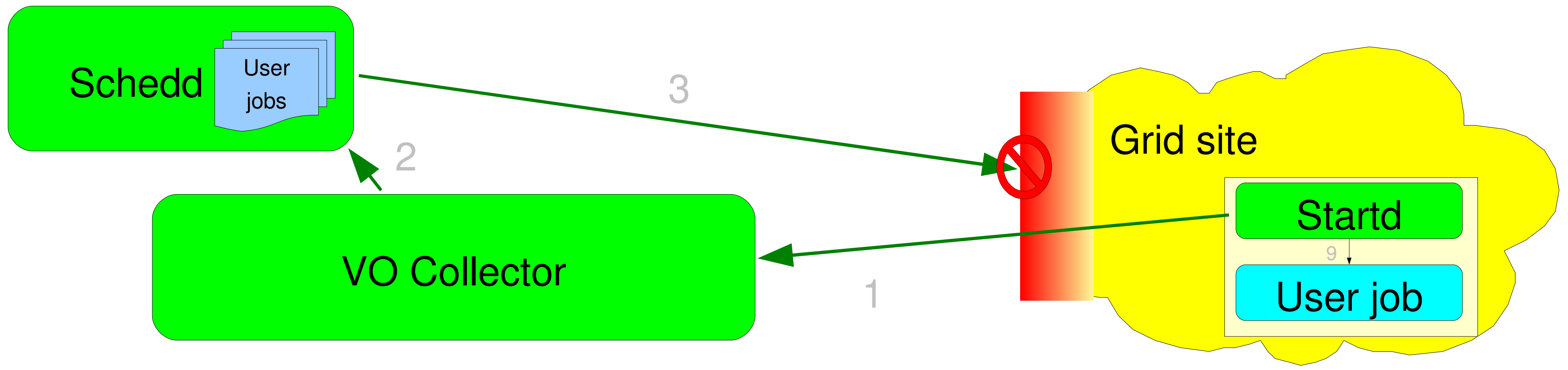
## Advanced topics

## 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

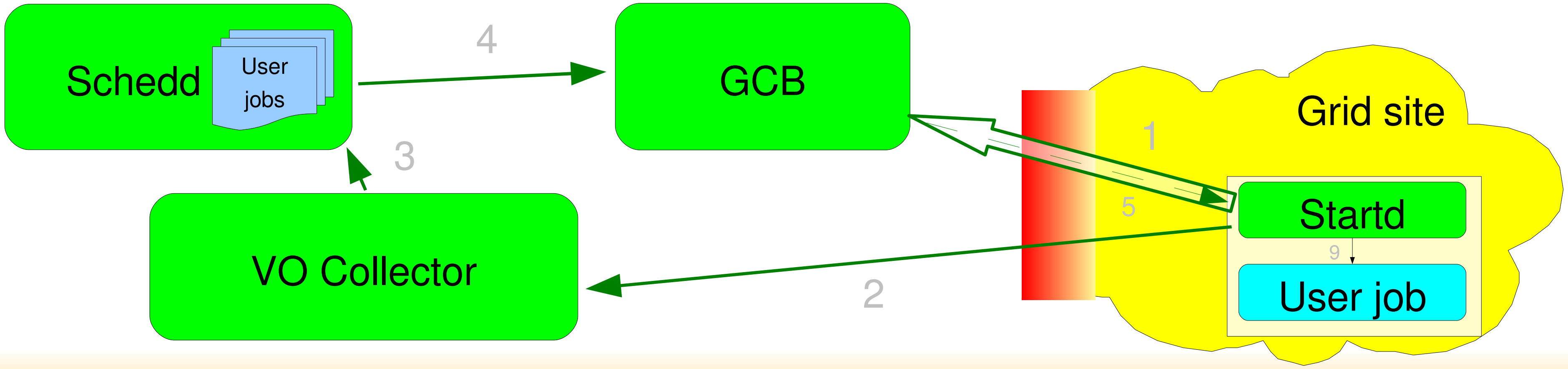
# Private networks and firewalls

- 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



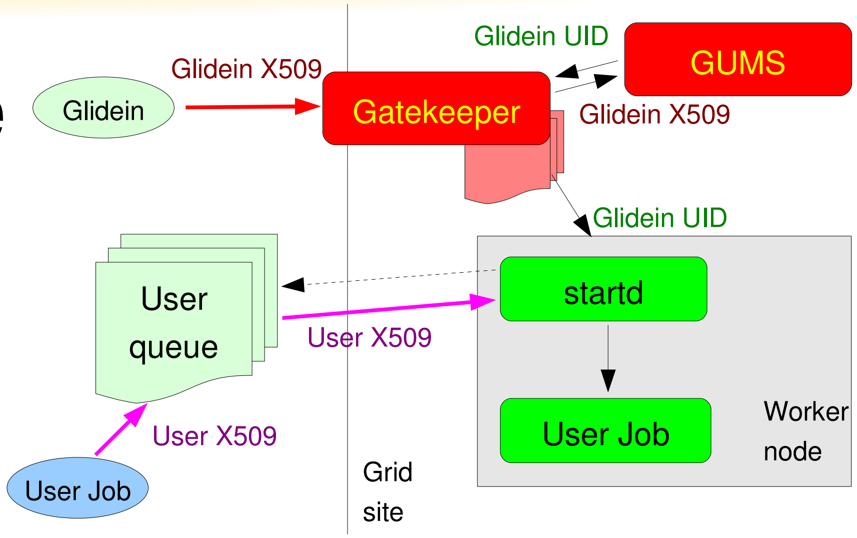
# Using Condor GCB

- 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 ~1k glideins



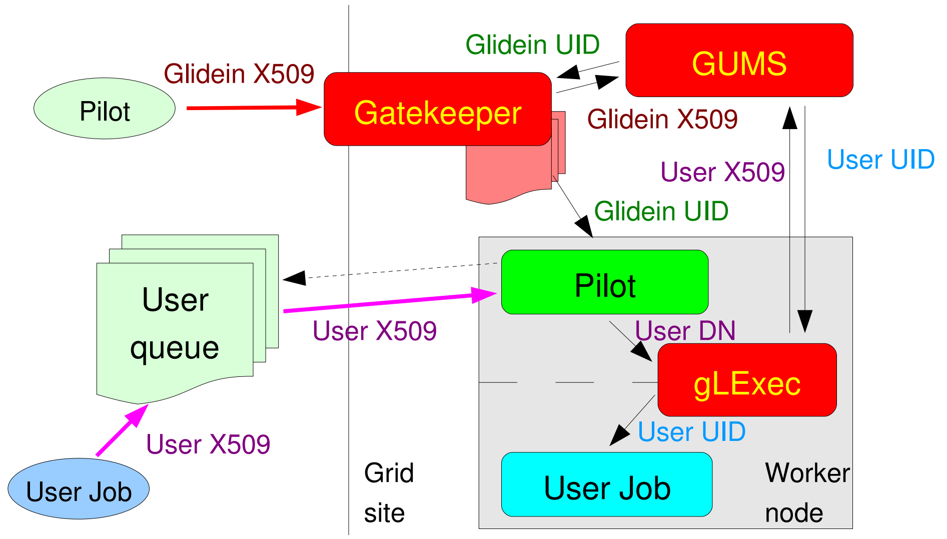
## Integration with Grid-local security infrastructure

- 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

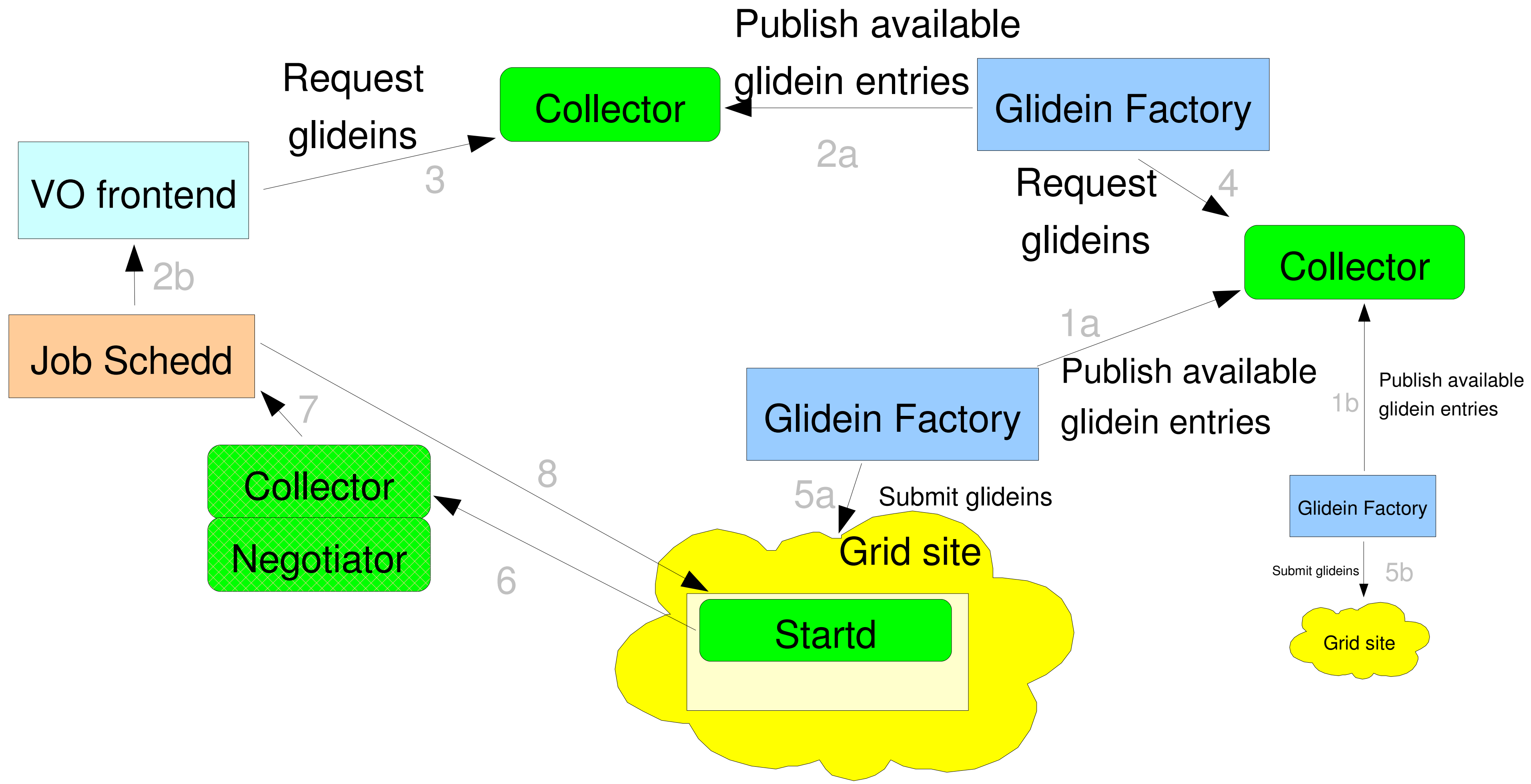


## Future work

- Need to implement several pieces that have been mentioned before
  - Waiting for some real users to bump up priorities
  - Current set satisfies the needs of CMS test harness
- With real users will certainly come new requests
  - I do not dare forecast what they might be
- As a wish, I would not mind if Condor natively implemented most of the functionality

# An example of possible future work

- Hierarchies of glidein factories


Publish available glidein entries
Request glideins
Collector
Glidein Factory
2a
VO frontend
3
Request glideins
4
Collector
2b
Job Schedd
1a
Publish available glidein entries
Glidein Factory
Publish available glidein entries
1b
7
Collector
Negotiator
8
5a
Submit glideins
Glidein Factory
Grid site
6
Startd
Submit glideins
5b
Grid site