GlideIns - The Future of GRID Computing

How and Why do we use GlideinWMS in CMS

Jeff Dost UCSD

What is GlideinWMS

- A WMS or Workload Management System that uses a **pilot** submission model to run jobs on the grid
- A **pilot** is a grid job that acquires a batch slot at a site, then calls home and fetches a real user job to start on it

But First Some Background

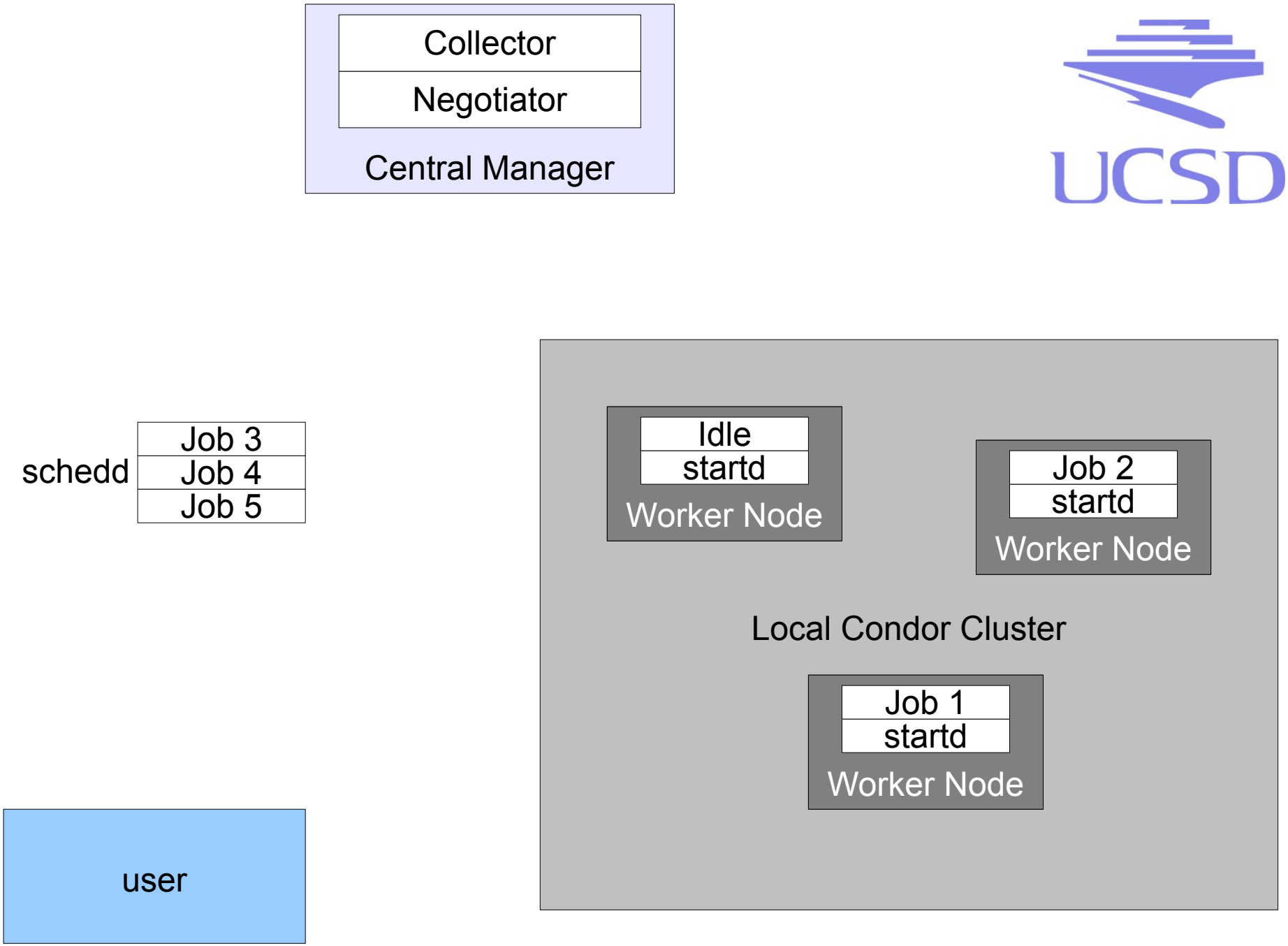
- In order to fully understand how GlideinWMS works it helps to have some basic knowledge of Condor first
- Condor is a widely used batch system
- Many sites on OSG deploy it
- GlideinWMS is heavily built on and dependent on the Condor Architecture

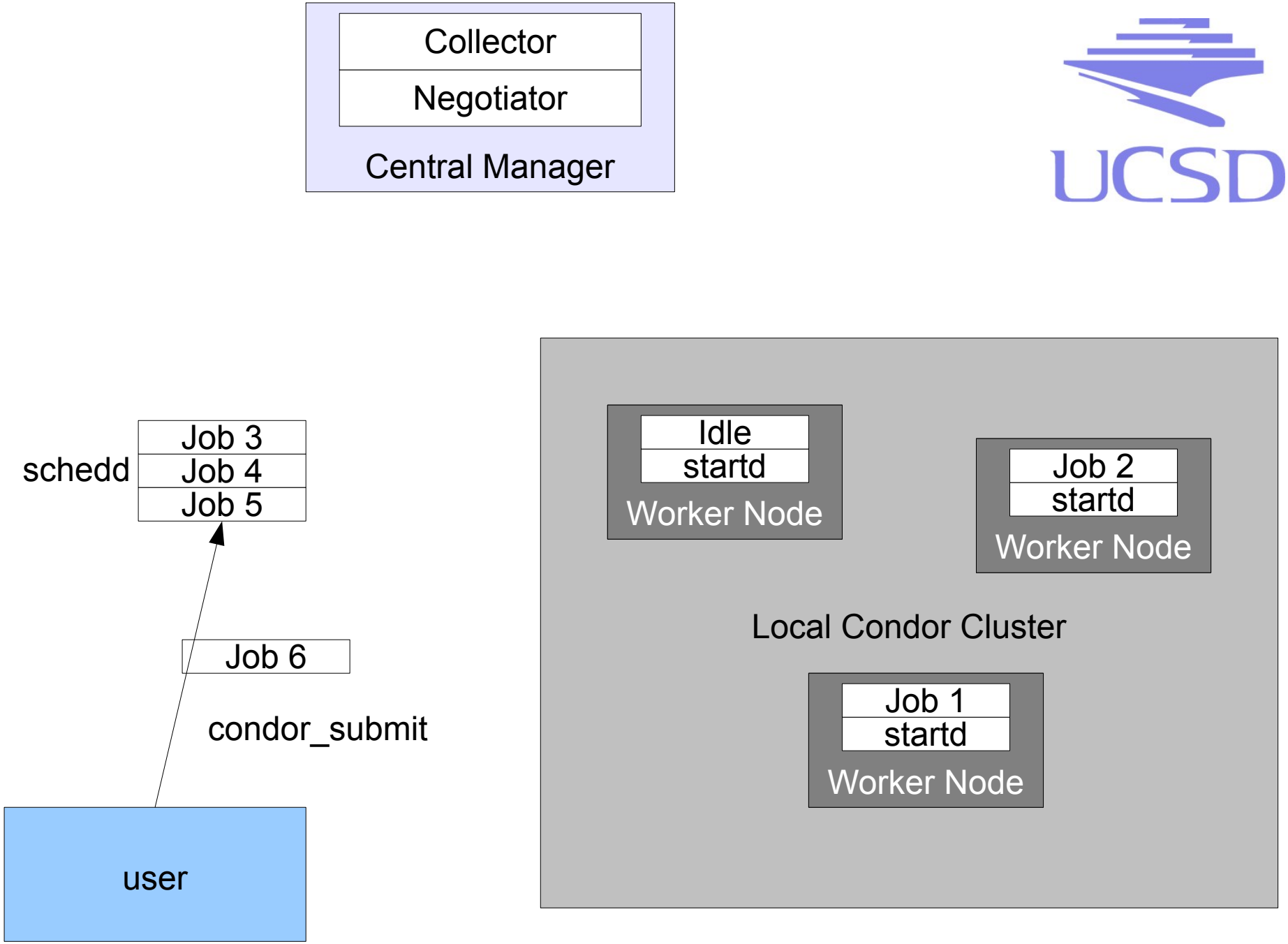
Some Condor Definitions

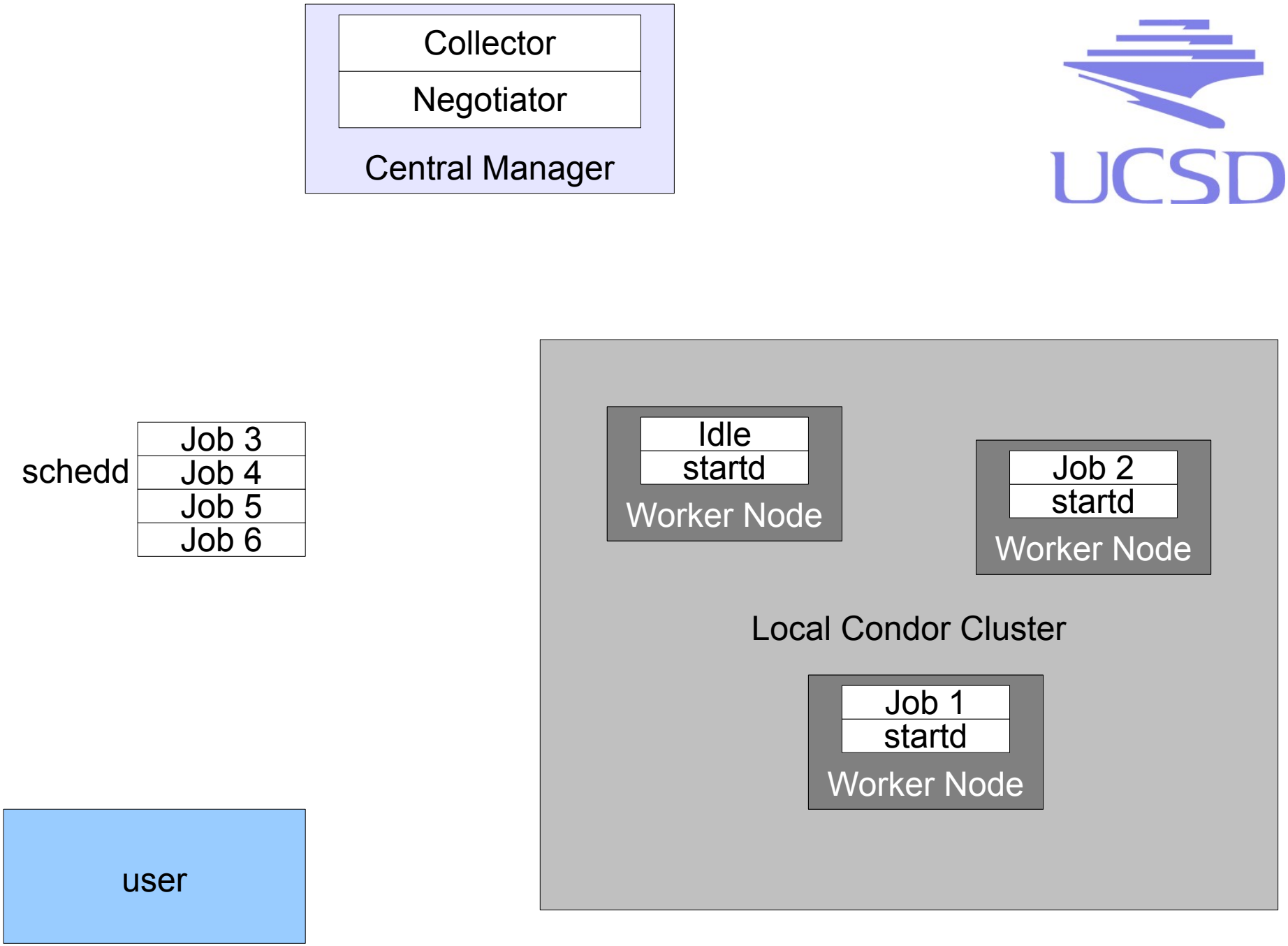
- **Resource** – a machine that accepts user jobs (Worker Node)
- **Job** – a batch program submitted to run a cluster of resources
- **ClassAd** – a list of attributes to describe a resource or a user job
  - For resources a ClassAd might state wether or not it is available – set in machine condor config
  - For a job it is a list of required or desired attributes for the resource it should run on – set by user on job submission

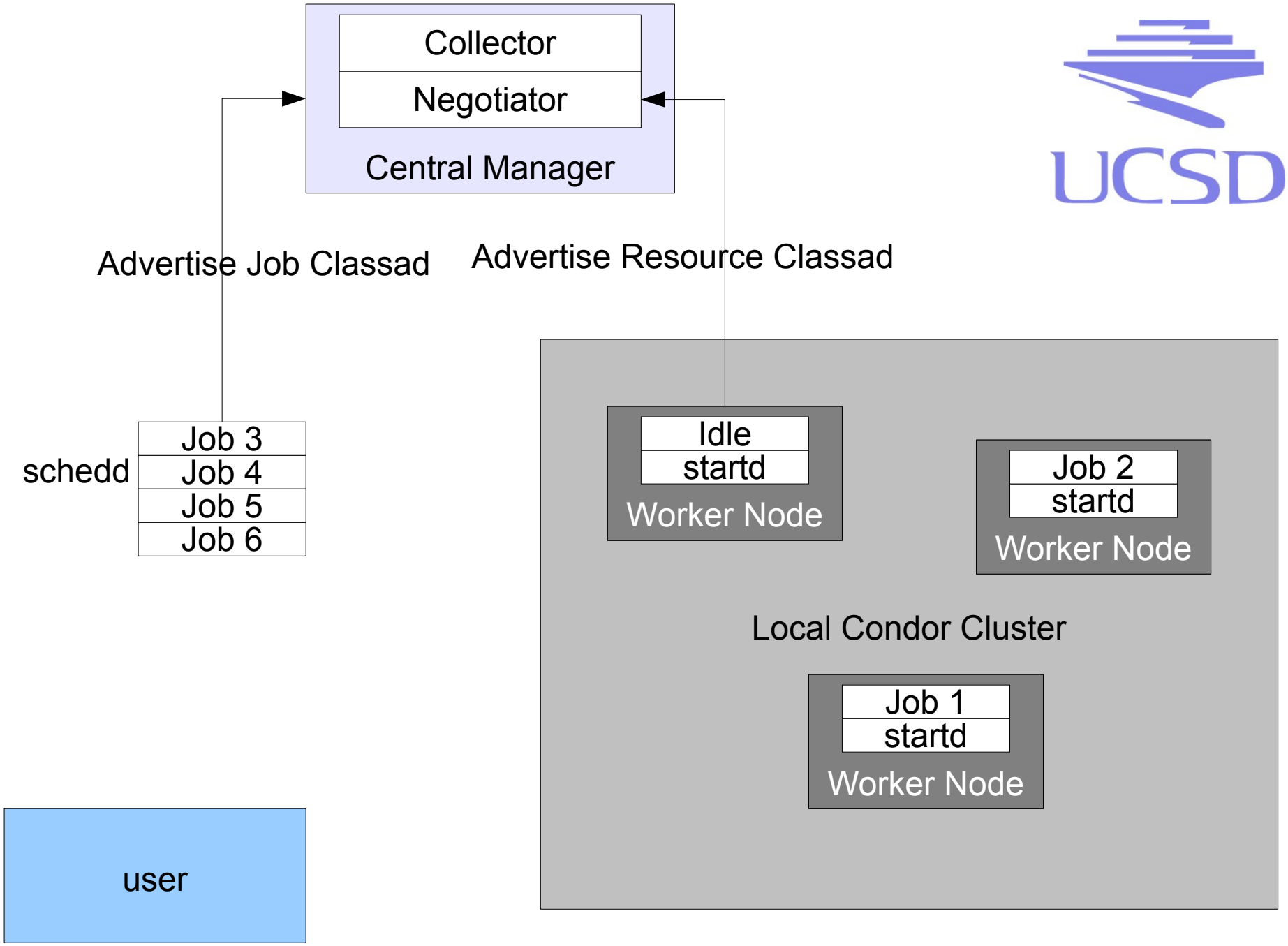
Condor Daemons

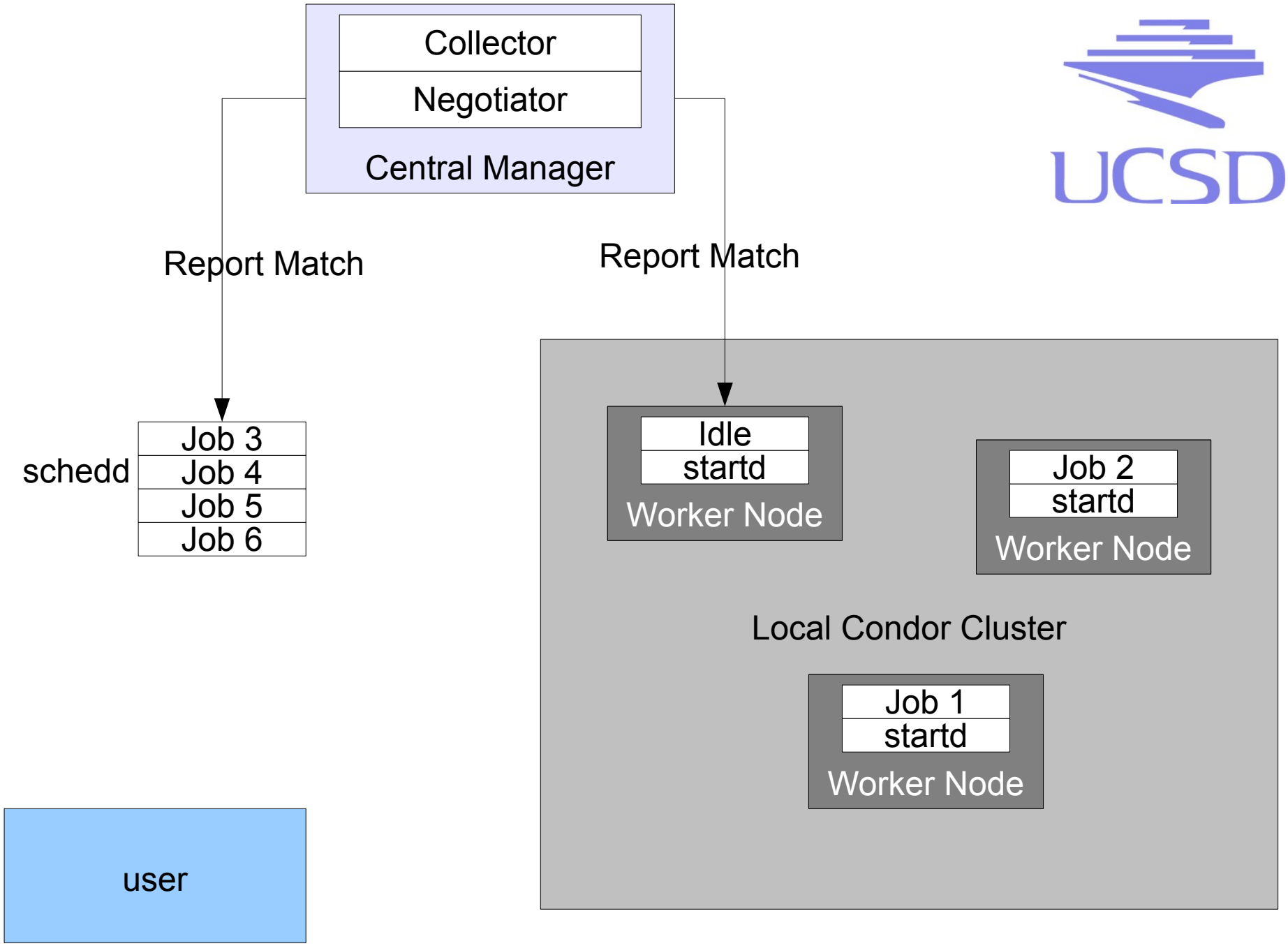
- **collector –** Like a whiteboard that keeps track of job and resource ClassAds
- **schedd** – Manages user job queue, advertises job ClassAds to collector
- **startd** – Represents a single resource in the condor pool, advertises resource ClassAds to collector, responsible for starting user job when schedd claims it
- **negotiator** – Responsible for matchmaking. Traverses collector and reports a match to schedd and startd when found

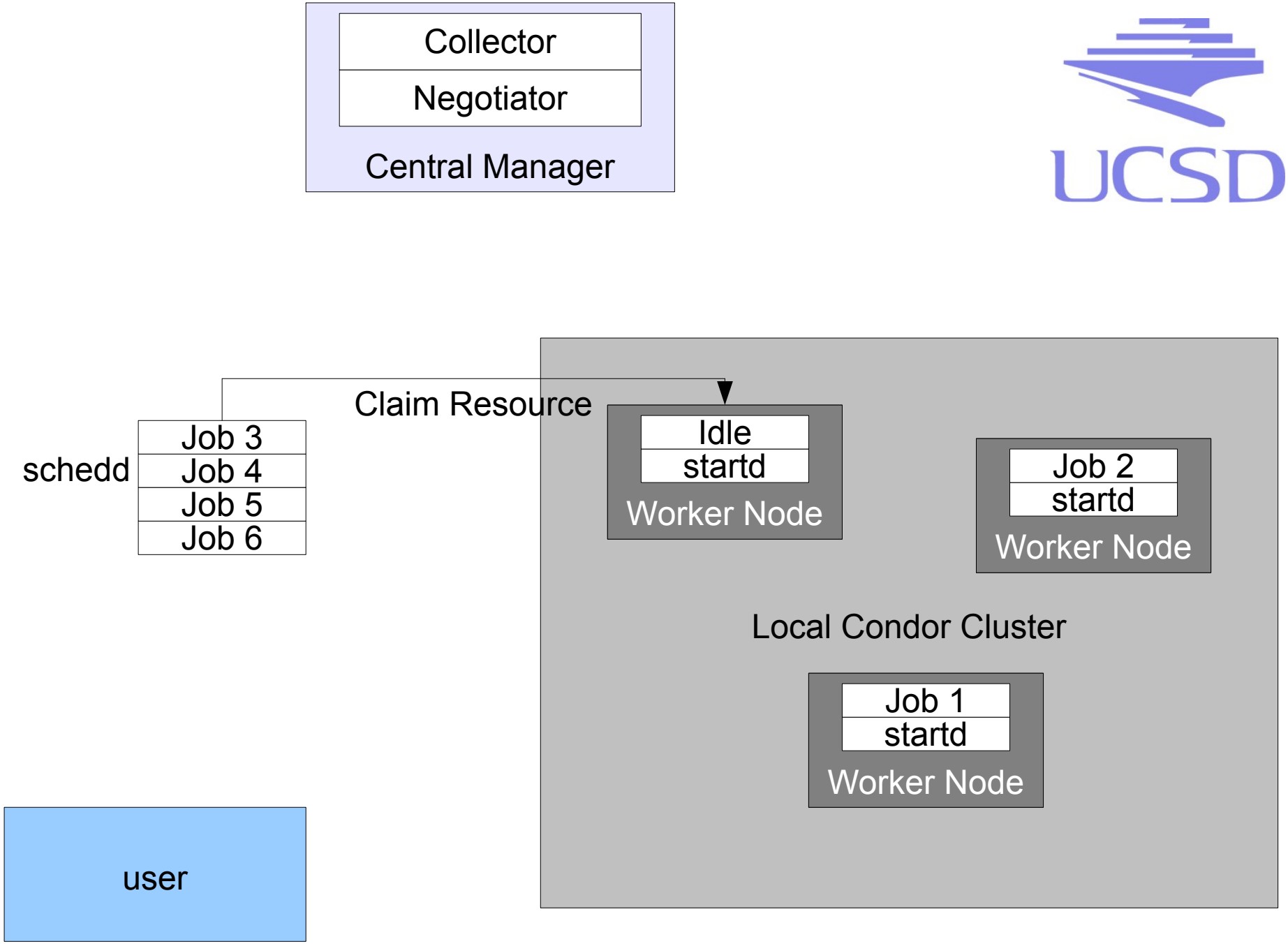

Collector
Negotiator
Central Manager
UCSD
schedd
Job 3
Job 4
Job 5
Idle
startd
Worker Node
Job 2
startd
Worker Node
Local Condor Cluster
Job 1
startd
Worker Node
user

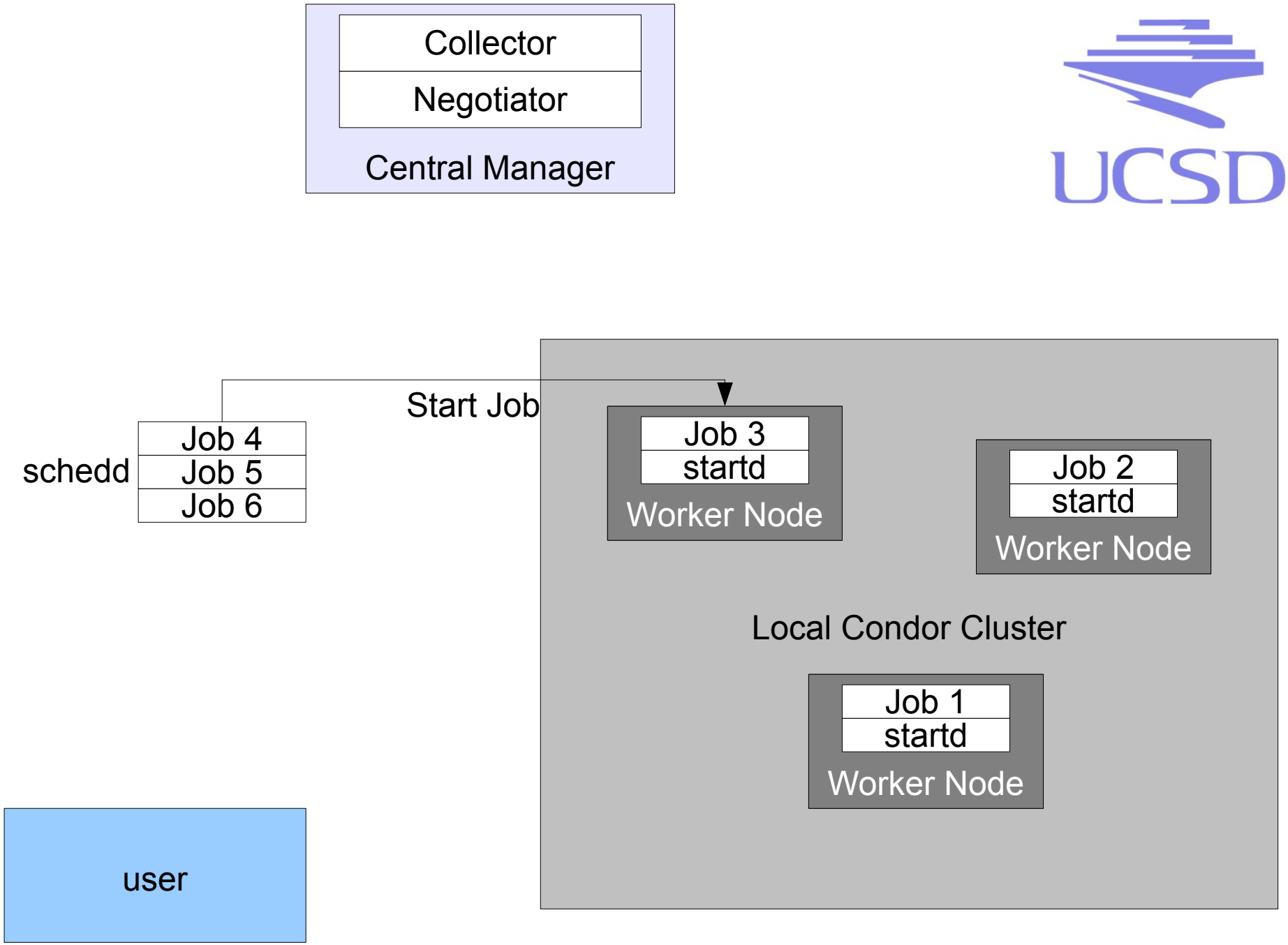

Collector
Negotiator
Central Manager
UCSD
schedd
Job 3
Job 4
Job 5
Job 6
condor_submit
user
Idle
startd
Worker Node
Job 2
startd
Worker Node
Local Condor Cluster
Job 1
startd
Worker Node


Collector
Negotiator
Central Manager
UCSD
schedd
Job 3
Job 4
Job 5
Job 6
Idle
startd
Worker Node
Job 2
startd
Worker Node
Local Condor Cluster
Job 1
startd
Worker Node
user


Collector
Negotiator
Central Manager
Advertise Job Classad
Advertise Resource Classad
schedd
Job 3
Job 4
Job 5
Job 6
Idle
startd
Worker Node
Job 2
startd
Worker Node
Local Condor Cluster
Job 1
startd
Worker Node
user


Collector
Negotiator
Central Manager
Report Match
Report Match
schedd
Job 3
Job 4
Job 5
Job 6
Idle
startd
Worker Node
Job 2
startd
Worker Node
Local Condor Cluster
Job 1
startd
Worker Node
user


Collector
Negotiator
Central Manager
Claim Resource
schedd
Job 3
Job 4
Job 5
Job 6
Idle
startd
Worker Node
Job 2
startd
Worker Node
Local Condor Cluster
Job 1
startd
Worker Node
user

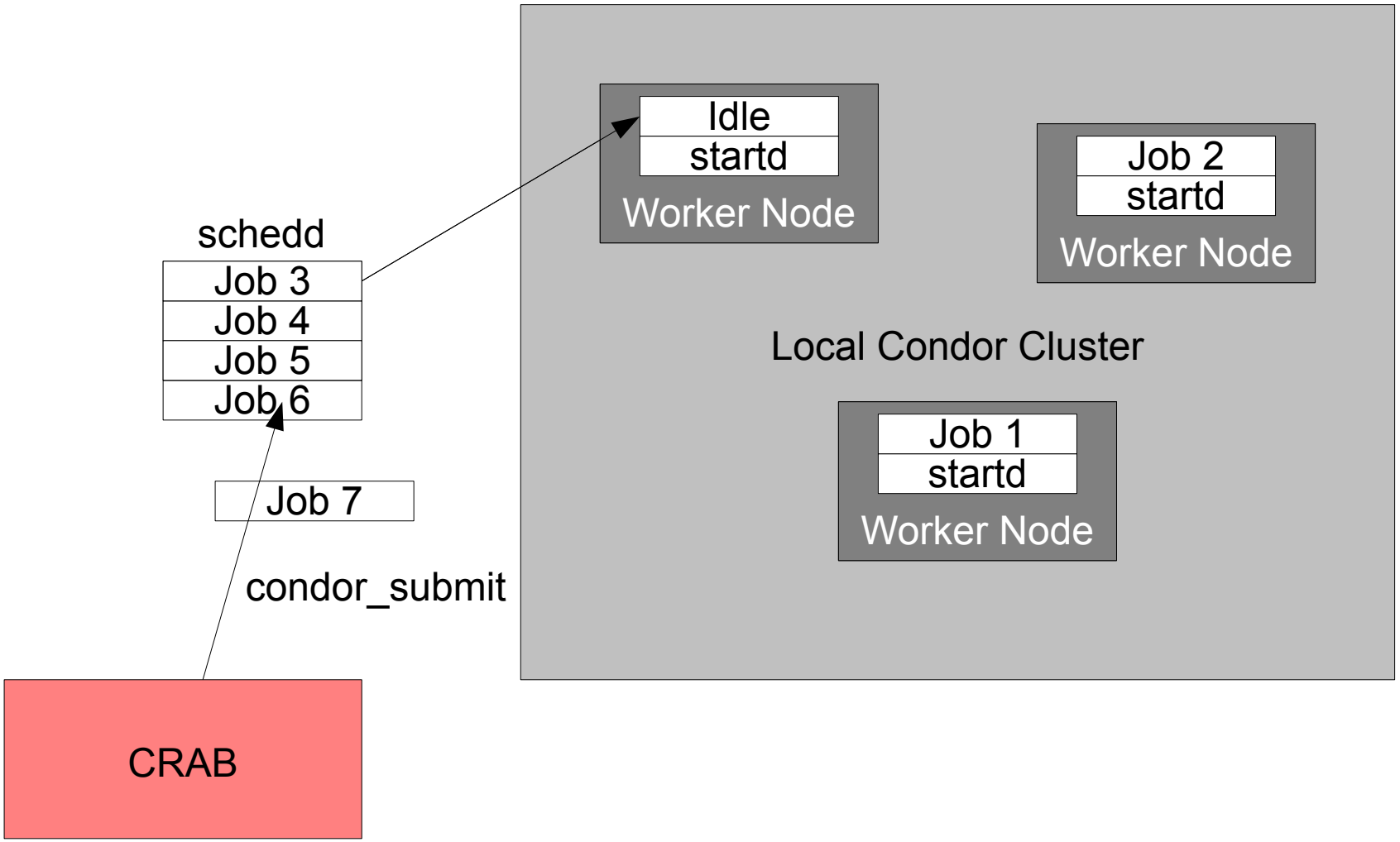

Collector
Negotiator
Central Manager
Start Job
schedd
Job 4
Job 5
Job 6
Job 3
startd
Worker Node
Job 2
startd
Worker Node
Local Condor Cluster
Job 1
startd
Worker Node
user

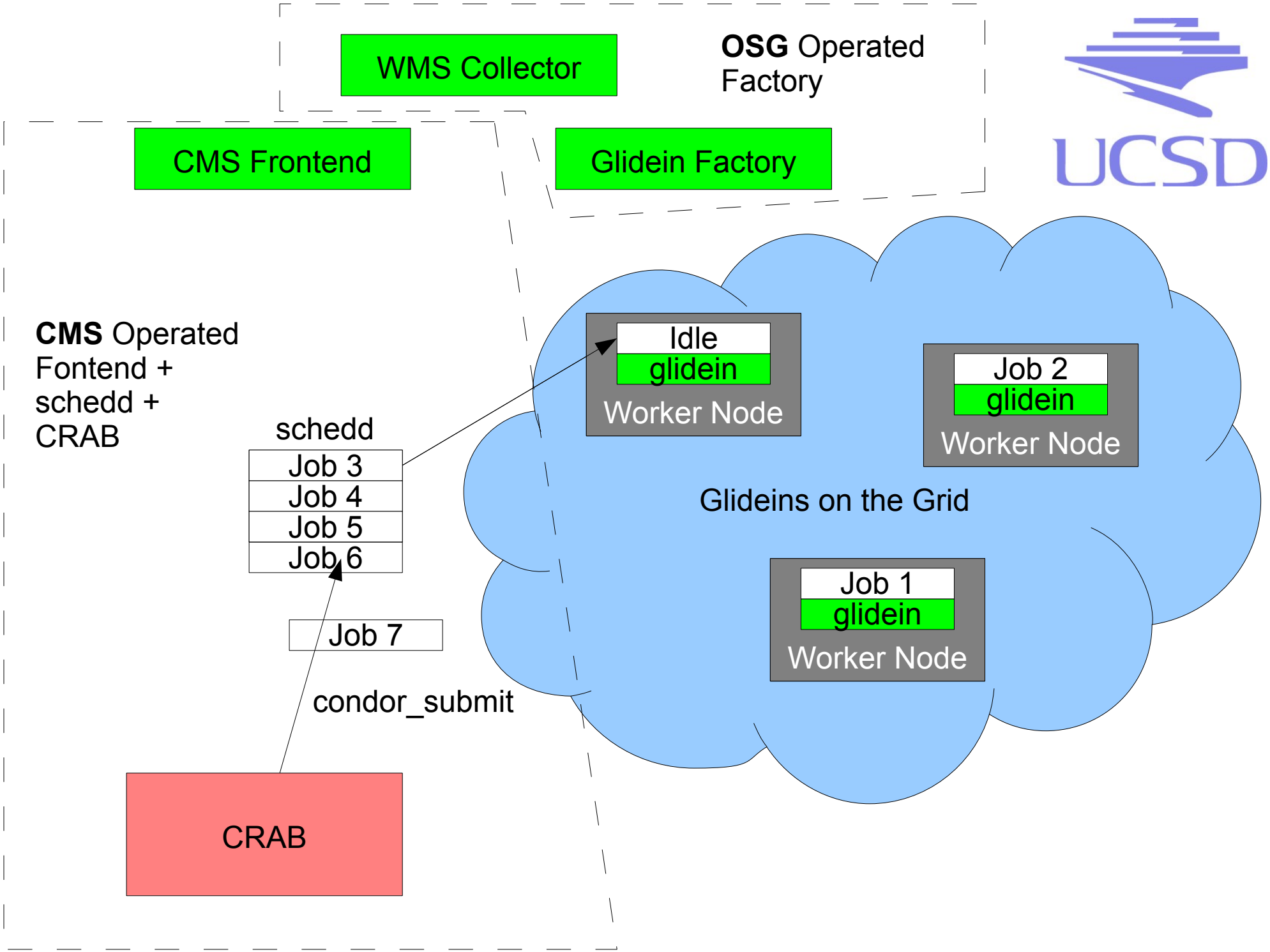
How is GlideinWMS used in CMS

- The slides that follow describe each component and explain its purpose

CMS User Submits CRAB Job

- CRAB submits to glidein pool schedd
- From CRAB service point of view its no different than submitting a job to a condor cluster


Idle
startd
Worker Node
Job 2
startd
Worker Node
schedd
Job 3
Job 4
Job 5
Job 6
Local Condor Cluster
Job 1
startd
Worker Node
Job 7
condor_submit
CRAB


WMS Collector
OSG Operated Factory
UCSD
CMS Frontend
Glidein Factory
CMS Operated Fontend + schedd + CRAB
schedd
Job 3
Job 4
Job 5
Job 6
Job 7
condor_submit
CRAB
Idle
glidein
Worker Node
Job 2
glidein
Worker Node
Glideins on the Grid
Job 1
glidein
Worker Node

Some GlideinWMS Definitions

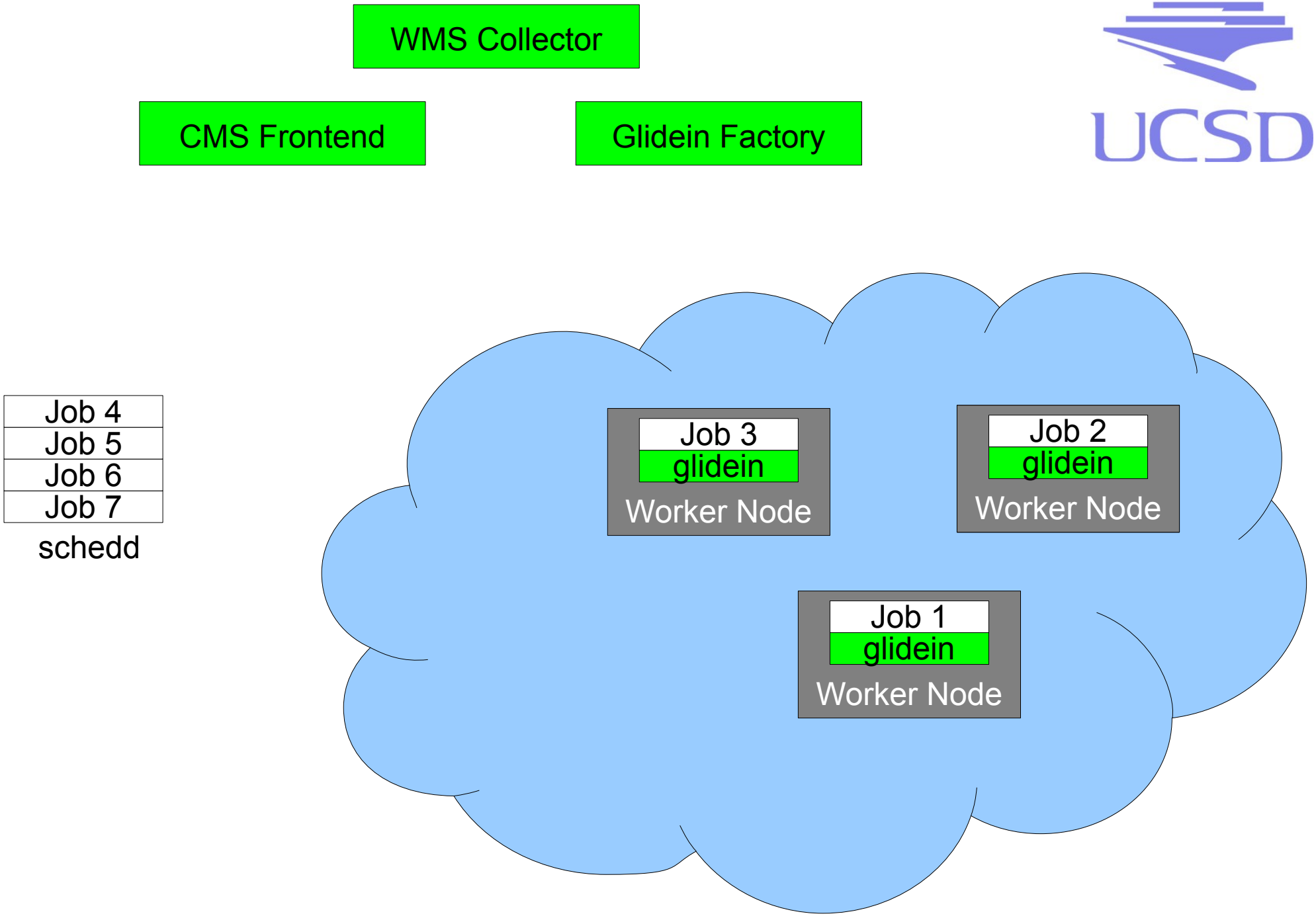
- **Glidein** – a pilot job that starts a Condor startd on the grid
- **Frontend** – component that watches over pending user jobs and makes sure glideins are available
- **Factory** – component that submits glideins to the grid when it receives Frontend requests
- **WMS Collector** – a condor collector that keeps Factory entry ClassAds and Frontend request ClassAds

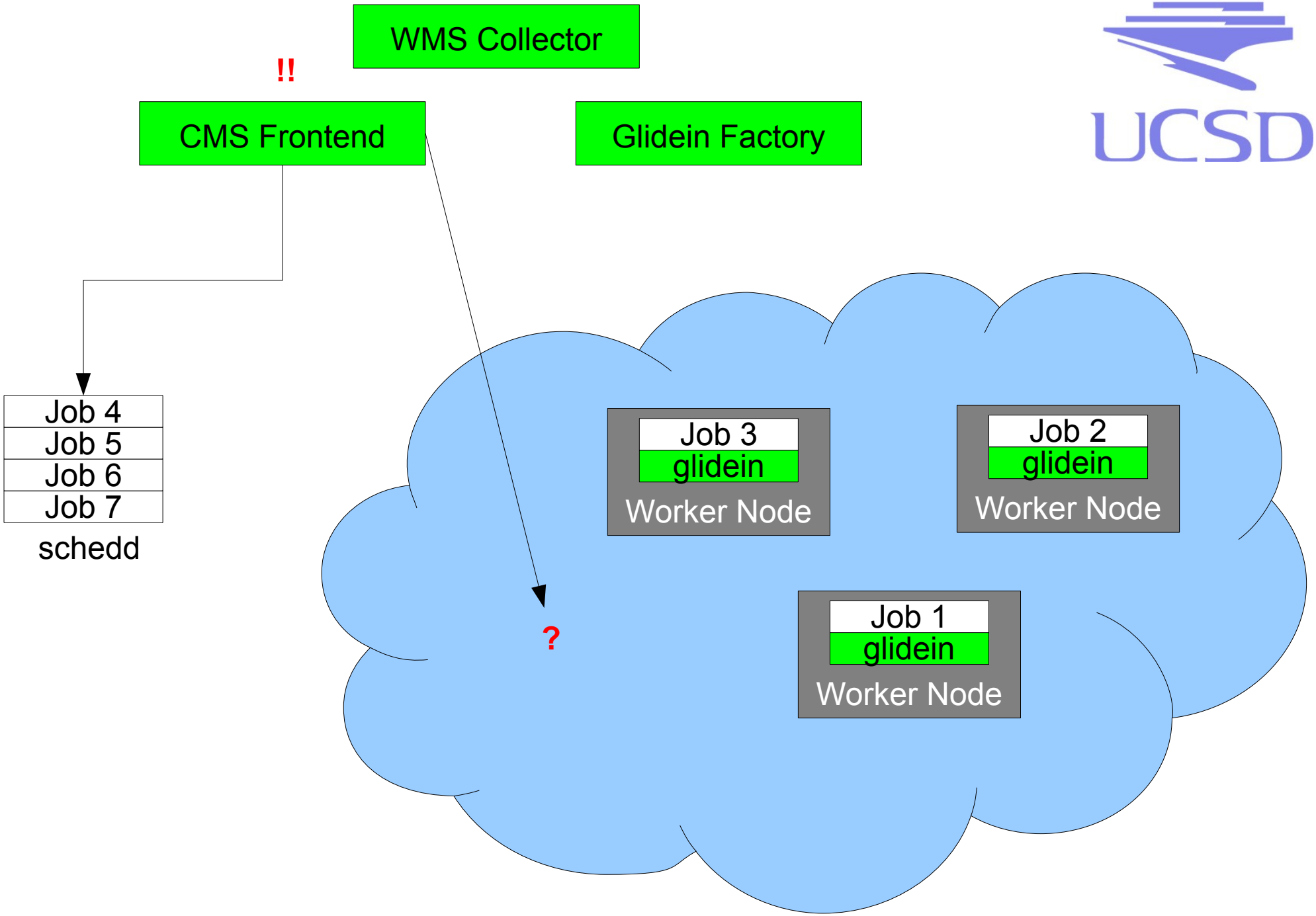
GlideinWMS Matchmaking

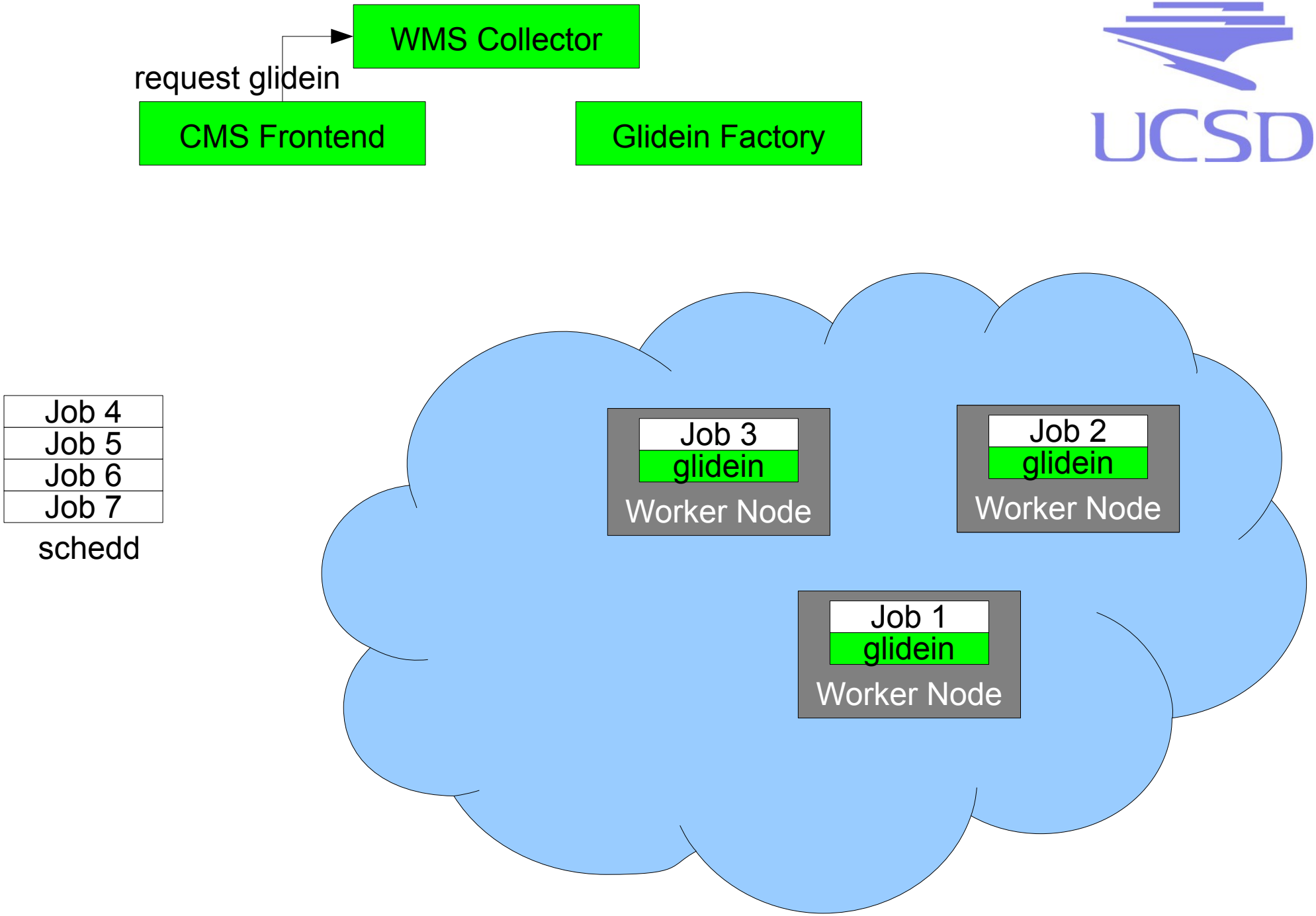
- The Factory advertises **entries** to the WMS Collector
- An entry is a ClassAd describing the kind of glidein a factory is able to submit to a particular grid site resource
- Custom **attributes** are defined in the entry in the factory config file

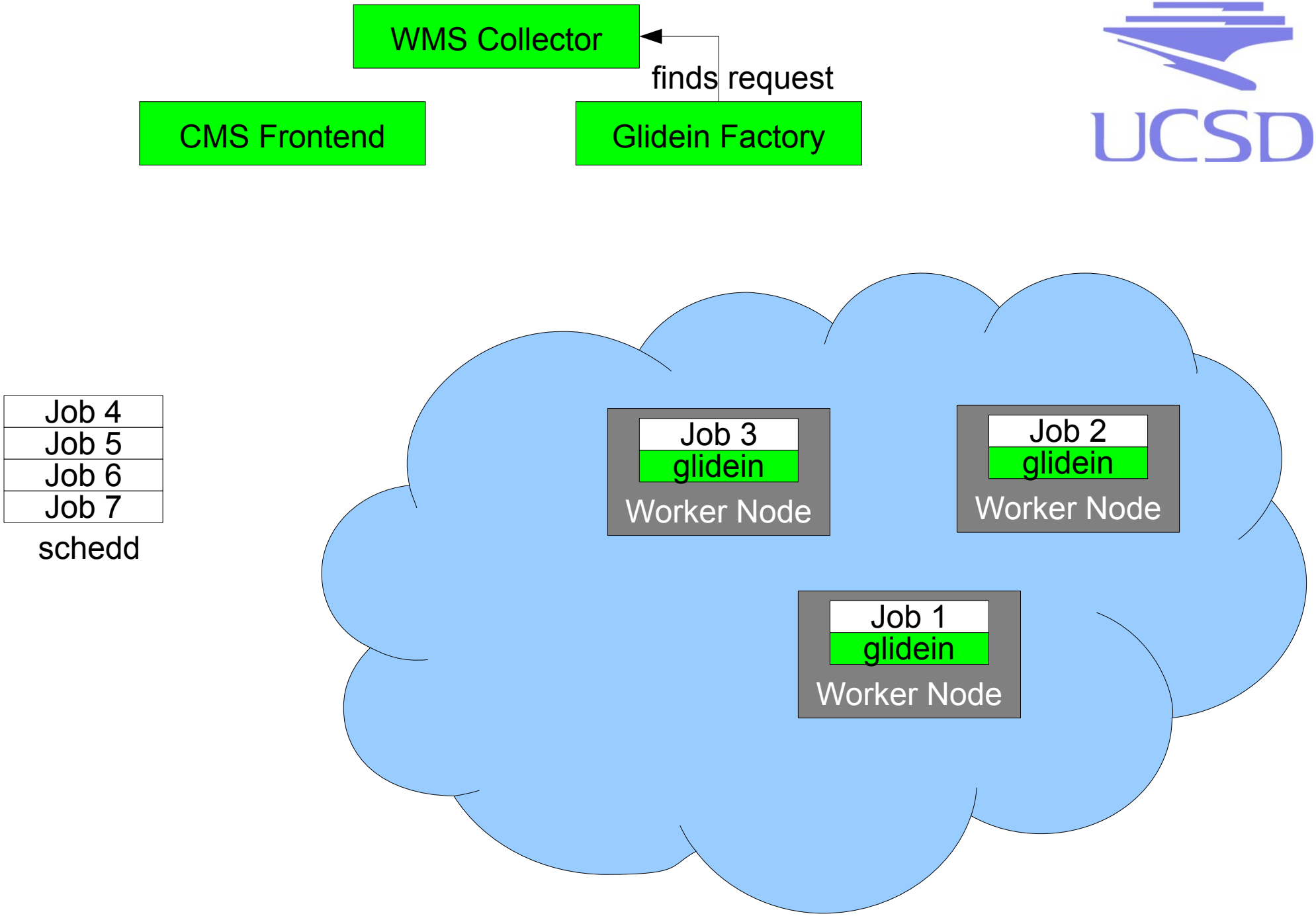
GlideinWMS Matchmaking

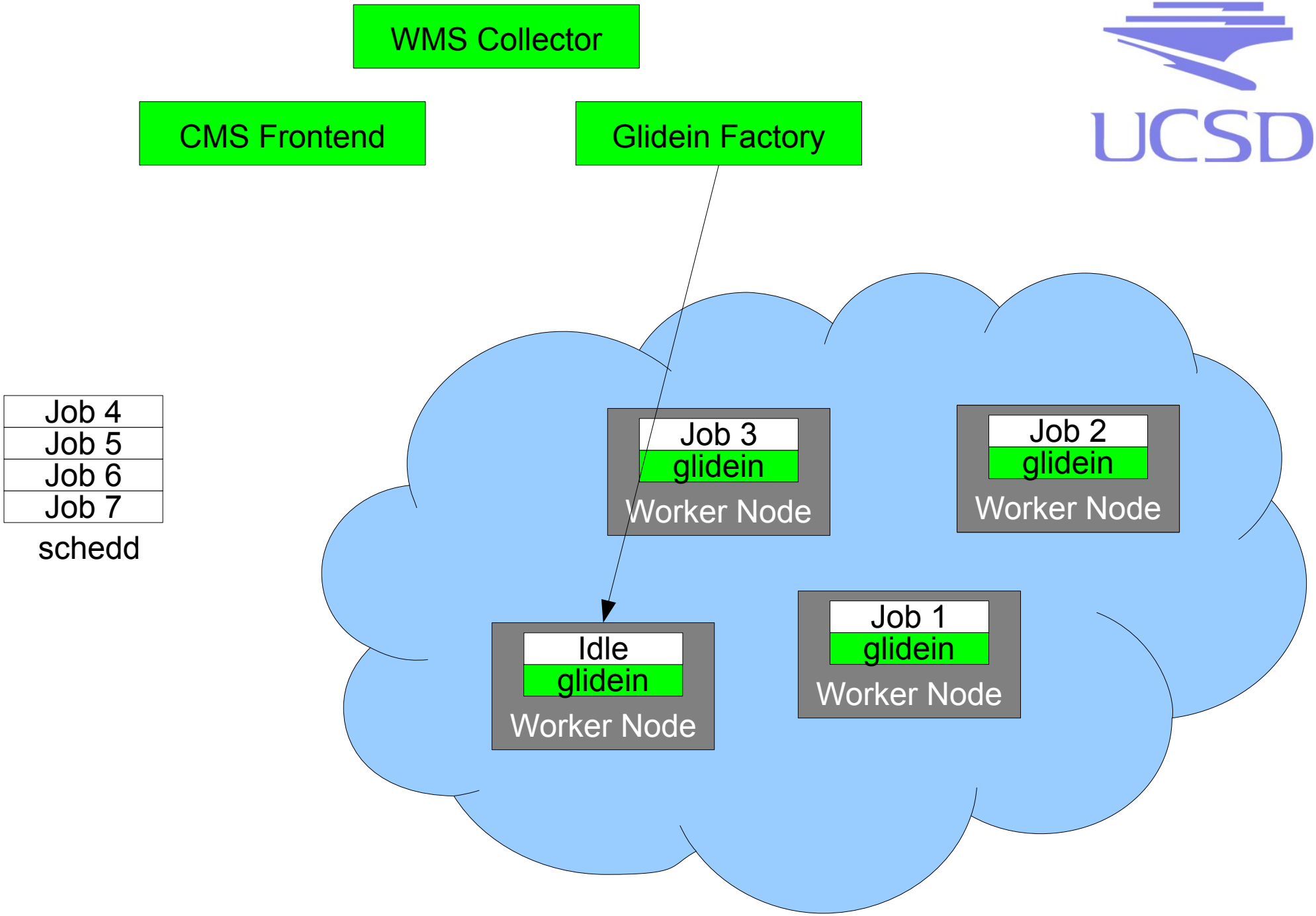
- The frontend reads the WMS Collector and uses a **match expression** against the factory entries and user jobs
- The match expression is defined in the Frontend config and allows a VO to decide which sites particular user jobs should run on
- If all existing glideins are already in use, the frontend posts a request to submit more glideins

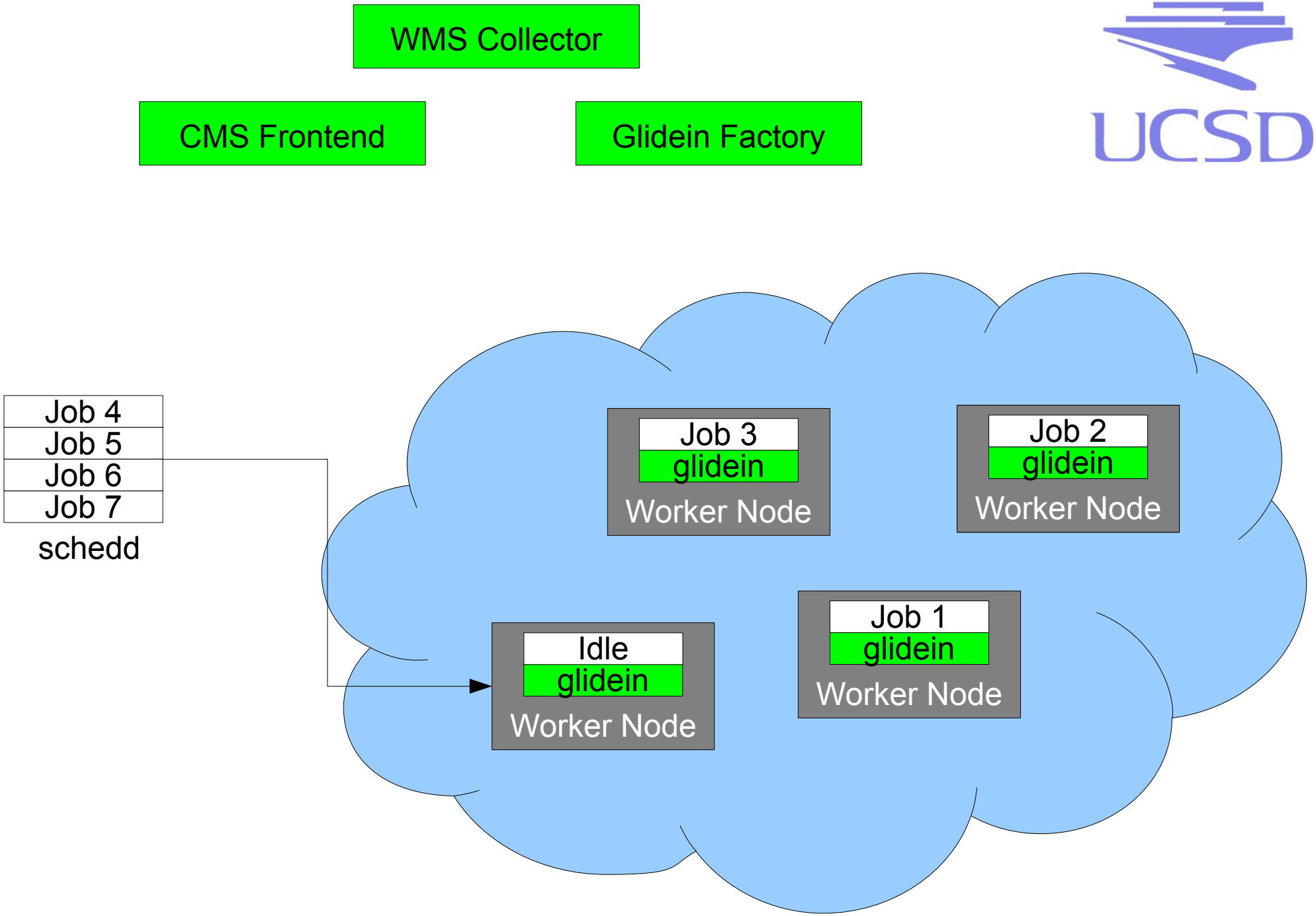

WMS Collector
CMS Frontend
Glidein Factory
UCSD
Job 4
Job 5
Job 6
Job 7
schedd
Job 3
glidein
Worker Node
Job 2
glidein
Worker Node
Job 1
glidein
Worker Node

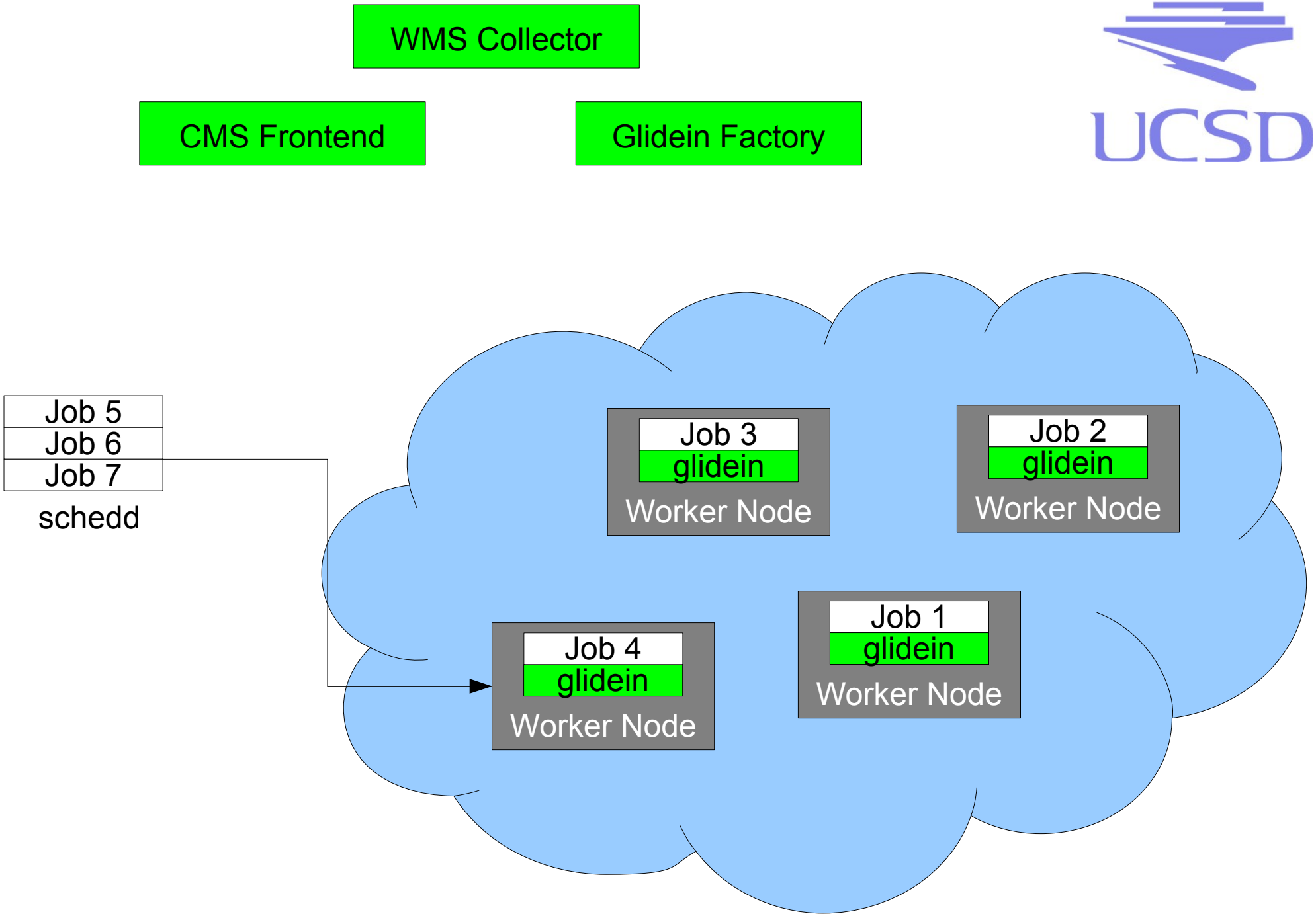

WMS Collector
!!
CMS Frontend
Glidein Factory
Job 4
Job 5
Job 6
Job 7
schedd
?
Job 3
glidein
Worker Node
Job 2
glidein
Worker Node
Job 1
glidein
Worker Node


WMS Collector
request glidein
CMS Frontend
Glidein Factory
UCSD
Job 4
Job 5
Job 6
Job 7
schedd
Job 3
glidein
Worker Node
Job 2
glidein
Worker Node
Job 1
glidein
Worker Node


WMS Collector
finds request
CMS Frontend
Glidein Factory
UCSD
Job 4
Job 5
Job 6
Job 7
schedd
Job 3
glidein
Worker Node
Job 2
glidein
Worker Node
Job 1
glidein
Worker Node


WMS Collector
CMS Frontend
Glidein Factory
UCSD
Job 4
Job 5
Job 6
Job 7
schedd
Job 3
glidein
Worker Node
Job 2
glidein
Worker Node
Idle
glidein
Worker Node
Job 1
glidein
Worker Node


WMS Collector
CMS Frontend
Glidein Factory
UCSD
Job 4
Job 5
Job 6
Job 7
schedd
Job 3
glidein
Worker Node
Job 2
glidein
Worker Node
Job 1
glidein
Worker Node
Idle
glidein
Worker Node


WMS Collector
CMS Frontend
Glidein Factory
Job 5
Job 6
Job 7
schedd
Job 3
glidein
Worker Node
Job 2
glidein
Worker Node
Job 1
glidein
Worker Node
Job 4
glidein
Worker Node

CMS Frontend Match Expression

- Factory defines entry attributes:
  - GLIDEIN_Site
  - GLIDEIN_Gatekeeper
  - GLIDEIN_SEs
- CRAB submission accepts lists:
  - DESIRED_Sites
  - DESIRED_Gatekeepers
  - DESIRED_SEs
- If any of the entry attributes are in any of the submission lists it is a match

What does a glidein do when it starts?

- First a glidein runs validation scripts to ensure it can run on the Worker Node environment
  - User jobs only start on pilots that pass validation
  - User jobs never see a broken node
- The glidein then reserves the slot and starts fetching user jobs to run (one at a time)
- **Key Concept**: Glideins sequentially run more than one user job and are not limited to running jobs from a single user over its lifetime

Implications

- How CMS benefits from glideins running more than one user job in its lifetime:
  - Increased throughput
  - Minimizes startup overhead
  - Especially beneficial for short lived jobs on massive clusters
- Advantage for grid site administrators:
  - User jobs won't be lost when worker nodes fail, they just restart on another matching glidein

Implications

- Caveat:
  - You can no longer determine the user identity from the CE because from there you only see the pilot proxy
  - At any given time you can't know what user will end up on a glidein at a WN
  - Requires some digging to figure out who the user behind the pilot is (more later)

Centralization in the CMS VO Frontend

- User priorities can now be handled from the CMS Glidein Admin
- VO no longer needs to request a site admin to handle priorities of O(1k) users
- All priority accounting centrally takes place in the CMS Condor instance instead of per-site requests

Security Implications

- The identity of the glidein job is that of the pilot proxy not the user proxy
- A pilot with a single proxy running multiple user jobs means users could potentially access each other's data
- A malicious user could also tamper with the glidein itself since mapped to the same UID
- The solution is to install **glexec** at your site

Glexec

- Glexec is like a proxy based equivalent of sudo for pilots
- A pilot uses the user proxy that the user delegated to it to change into the correct unix UID
- This requires glexec to have setuid permissions
- Now you retain user separation as well as separation from the glidein itself.
- Glexec makes it easier for a site admin to track down the real user behind the pilot (will cover more on this in demonstration)

Summary

- Using GlideinWMS is inherently more efficient than non-pilot grid submission techniques by reducing startup overhead
- GlideinWMS expands Condor by spreading the startds across the grid
- Submitting jobs on the grid becomes as simple as submitting to any other condor pool
- Site admins no longer have to micromanage O(1k) users. This is taken care of in the glidein negotiator by the CMS glidein admins

Acknowledgements

- This work is partially sponsored by
  - the US Department of Energy under Grant No. DE-FC02-06ER41436 subcontract No. 647F290 (OSG), and
  - the US National Science Foundation under Grants No. PHY-0612805 (CMS Maintenance & Operations), and OCI-0943725 (STCI).
- Special thanks to the glideinWMS and Condor teams