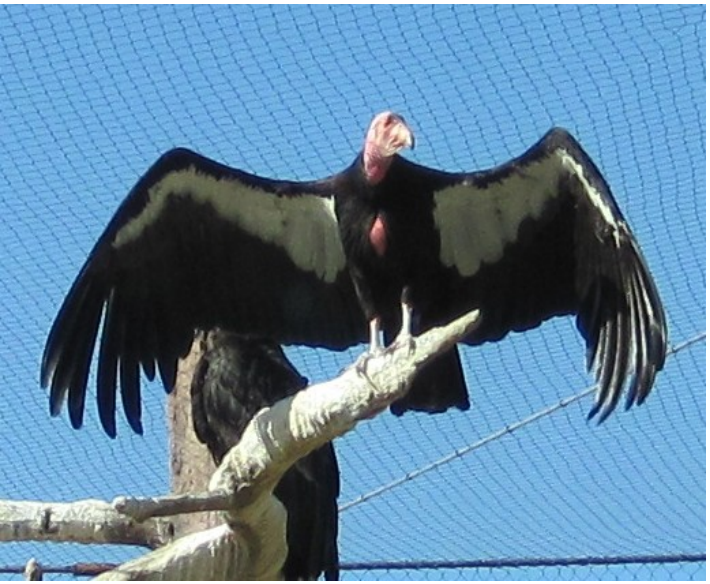
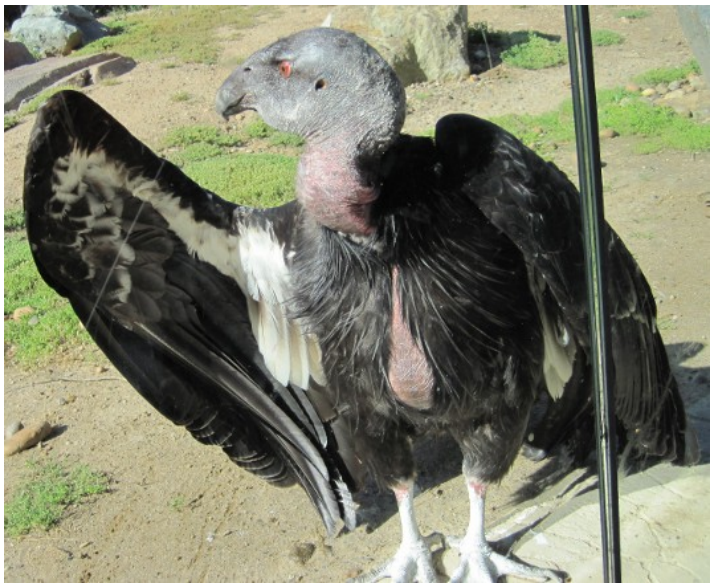
An argument for moving the requirements out of user hands

-

The CMS experience



presented by Igor Sfiligoi
UC San Diego



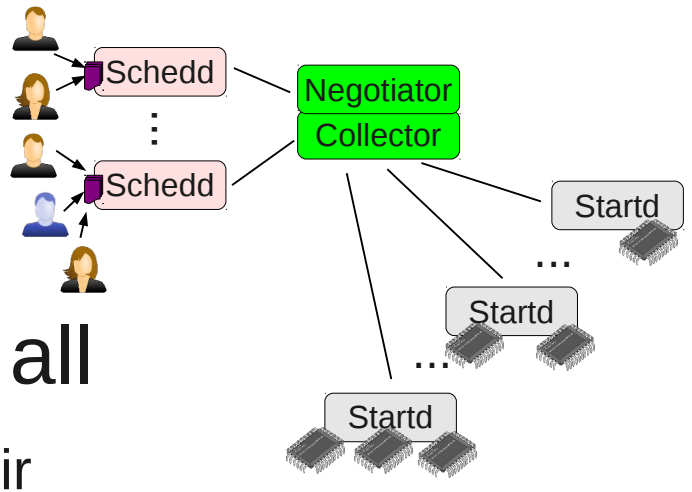
The basics

- Condor architecture clearly separates
  - **Resource providers** from
  - **Resource consumers**

Machines (aka worker nodes)
CPUs, Memory, IO,...

Job queues (aka submit nodes)
Jobs submitted by users

- Each has a daemon process to represent it
  - **Startd** for resource provides
  - **Schedd** for resource consumers



- A central service connects them all
  - Managed by a Collector/Negotiator pair

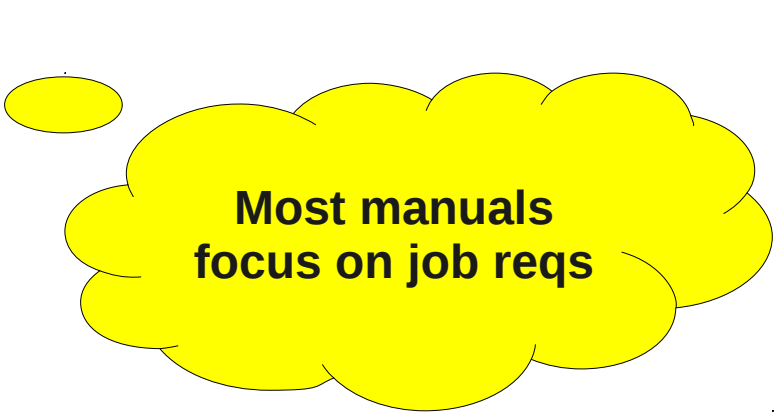
Matchmaking

- In order for a job to start running on a resource
  - The **job requirements** must evaluate to True
  - The **machine requirements** must evaluate to True

There is also the ranking,
but that's 2nd level optimization

Matchmaking

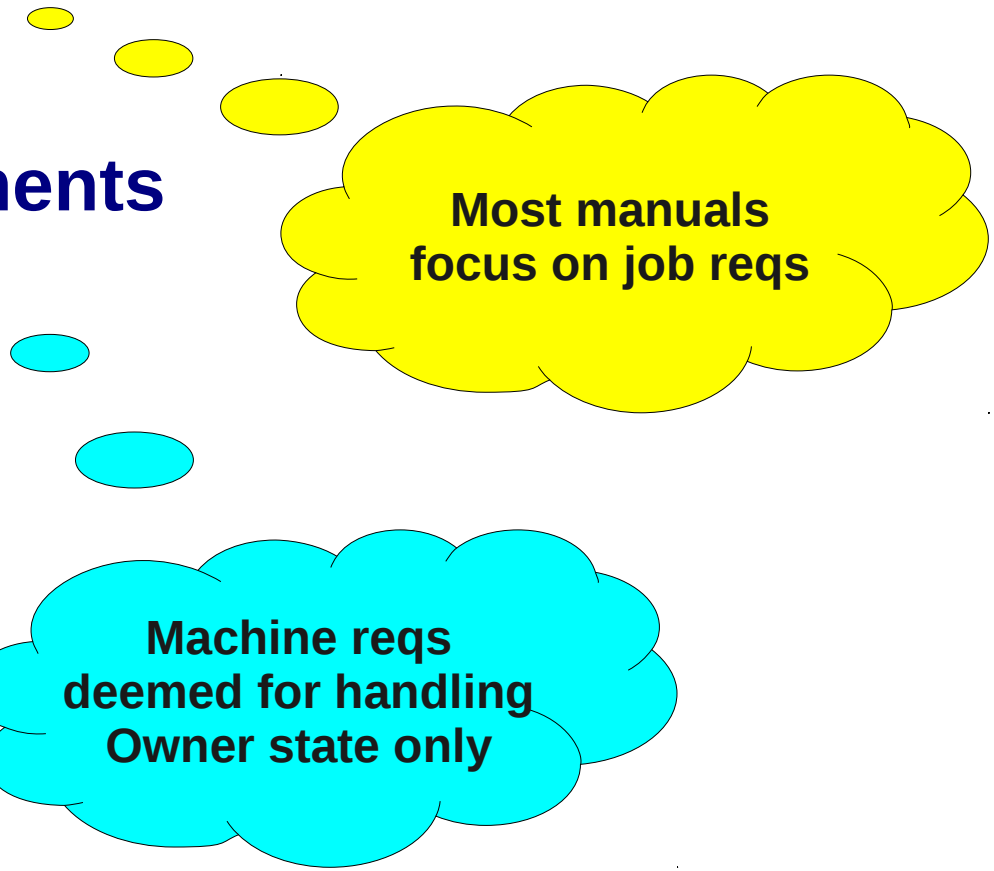
- In order for a job to start running on a resource
  - The **job requirements** must evaluate to True
  - The **machine requirements** must evaluate to True



There is also the ranking,
but that's 2nd level optimization

Matchmaking

- In order for a job to start running on a resource
  - The **job requirements** must evaluate to True
  - The **machine requirements** must evaluate to True



There is also the ranking,
but that's 2nd level optimization


My argument

Get rid of
Job Requirements

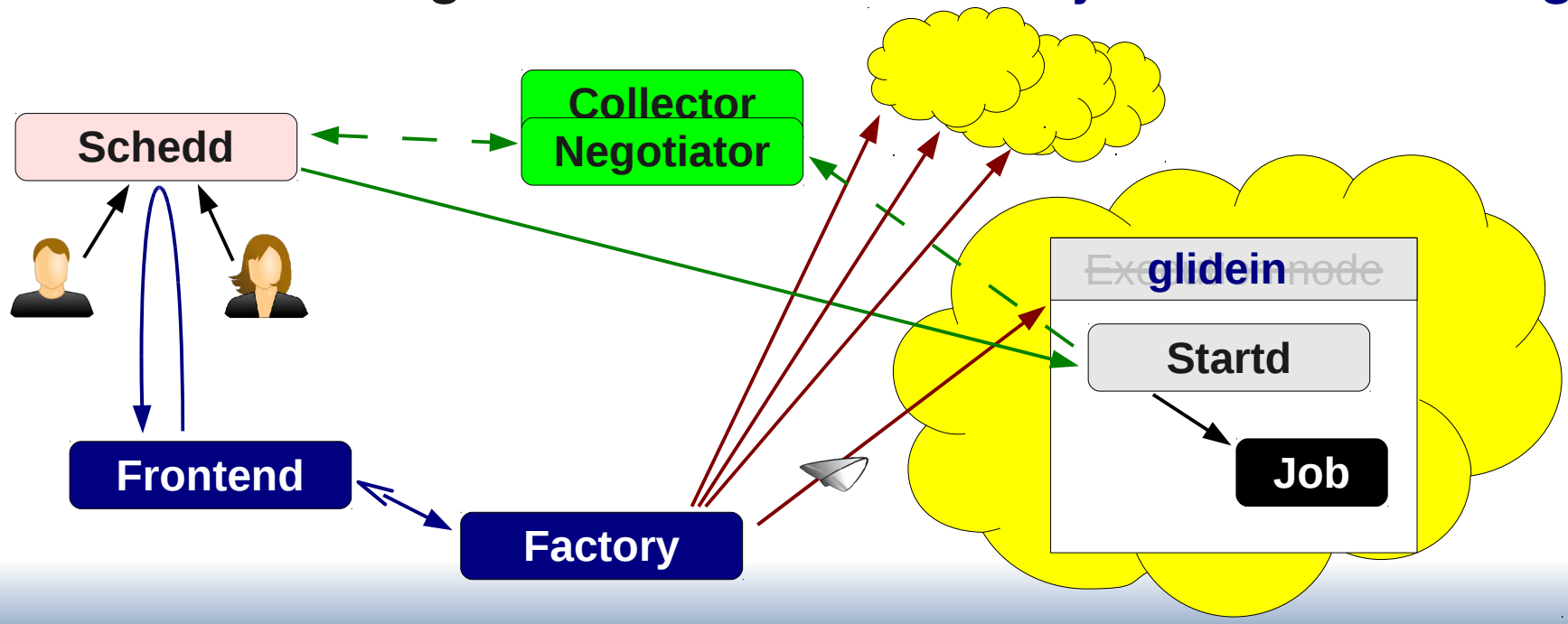
Put all logic in the
Machine Requirements

Some background

- I am a big glideinWMS user

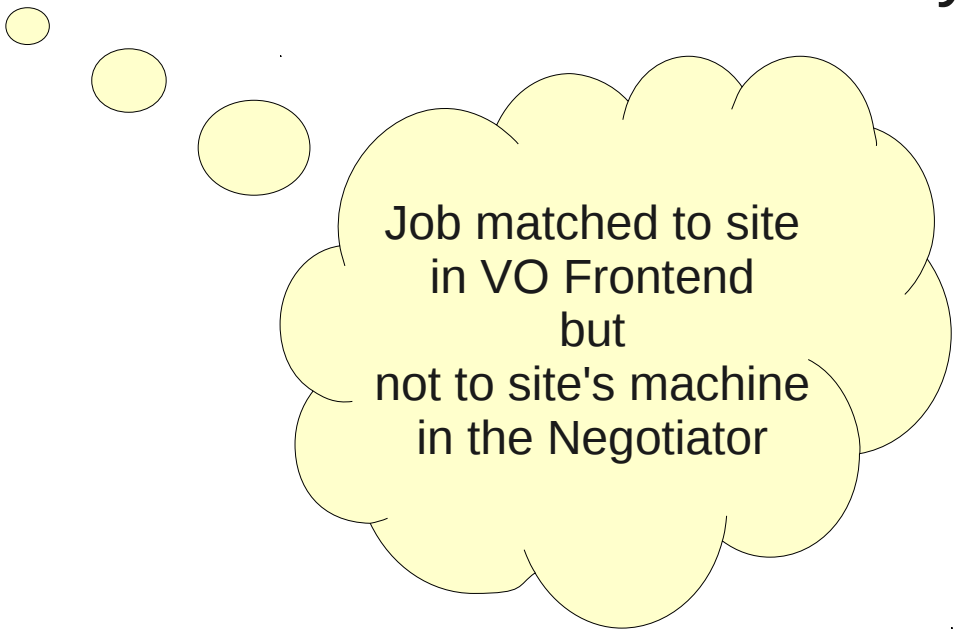
http://tinyurl.com/glideinWMS

- And glideinWMS has 2 level matchmaking
  - One at VO Frontend level – where to send glideins
  - One at Negotiator level – which job to start in a glidein



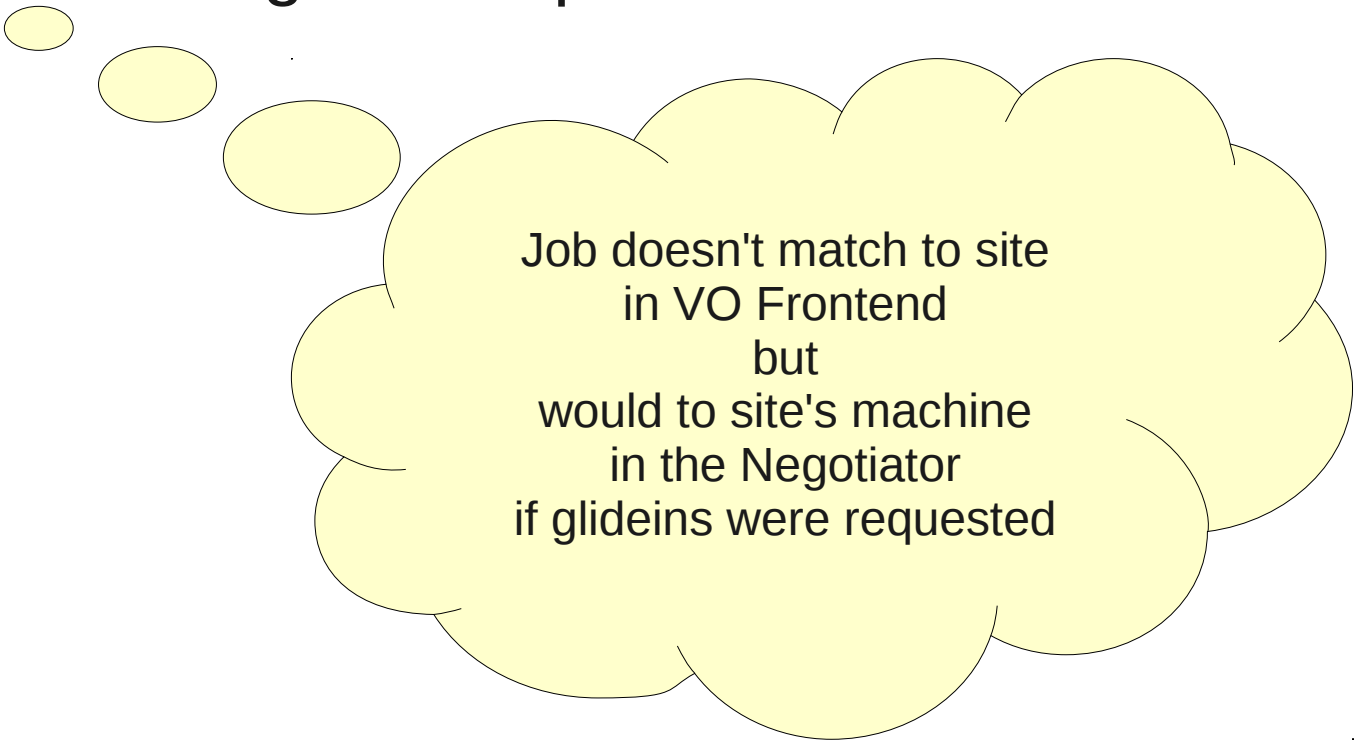
Matchmaking problem

- Both levels must be in sync, or you either
  - Ask for glideins which never match any jobs



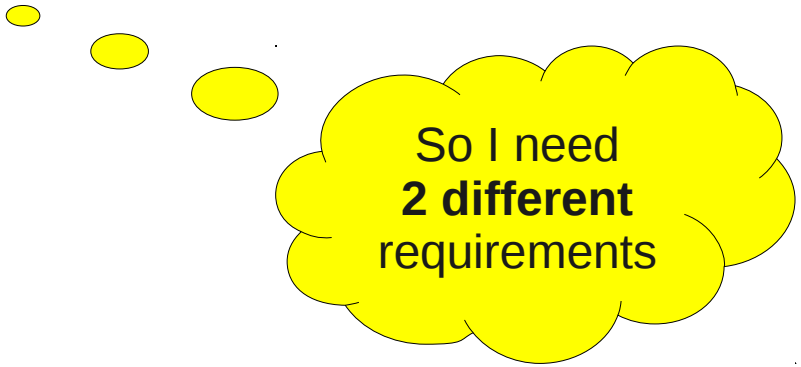
Matchmaking problem

- Both levels must be in sync, or you either
  - Ask for glideins which never match any jobs, or
  - Have job waiting in the queue when site available



Matchmaking problem

- Both levels must be in sync, or you either
  - Ask for glideins which never match any jobs, or
  - Have job waiting in the queue when site available
- But site and machine adds have different attributes
  - Not all machines on a site are exactly the same




The usual dilemma

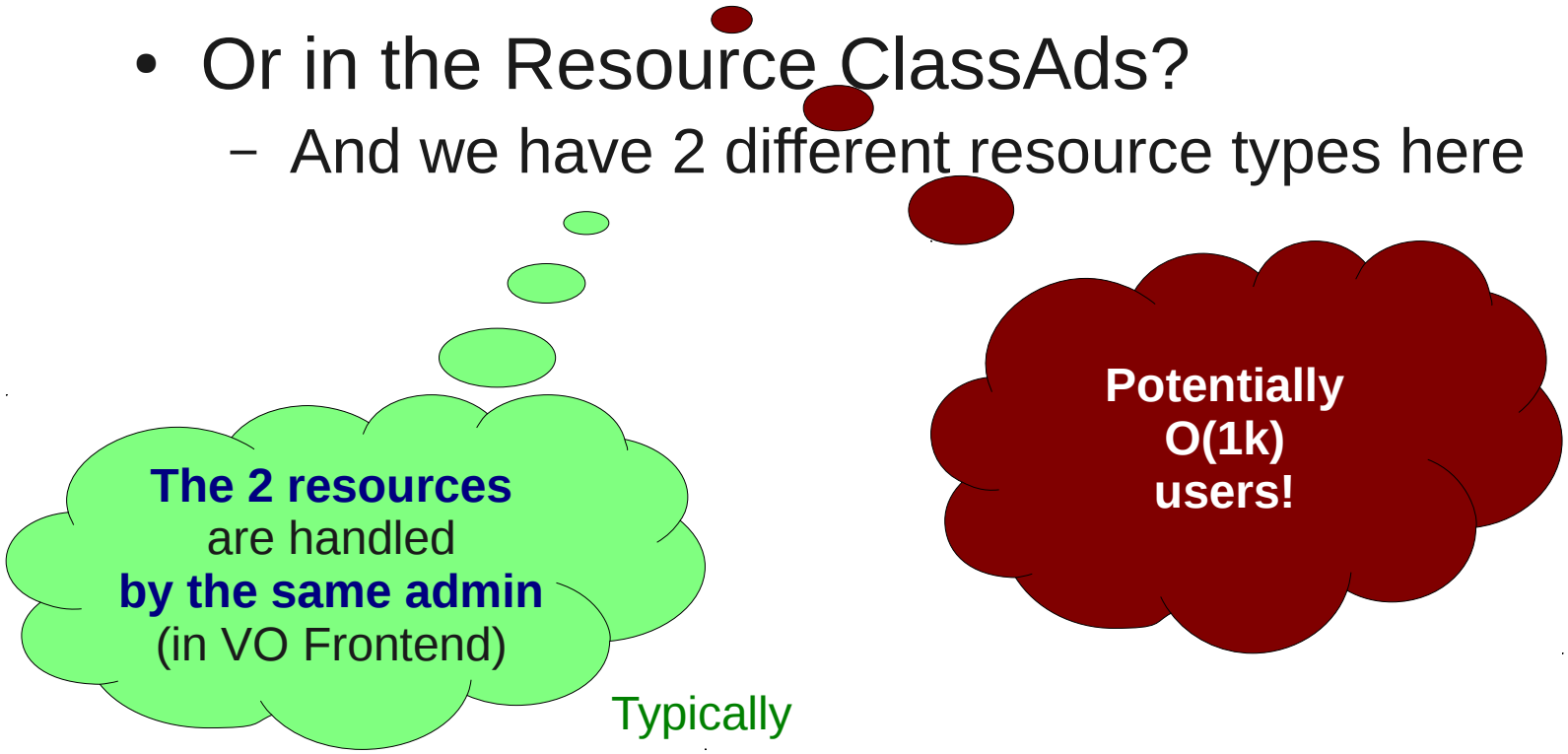
- Where do I define these requirements?
  - In user job ClassAd?
  - Or in the Resource ClassAds?
    - And we have 2 different resource types here

The usual dilemma

- Where do I define these requirements?
  - In user job ClassAd?
  - Or in the Resource ClassAds?
    - And we have 2 different resource types here

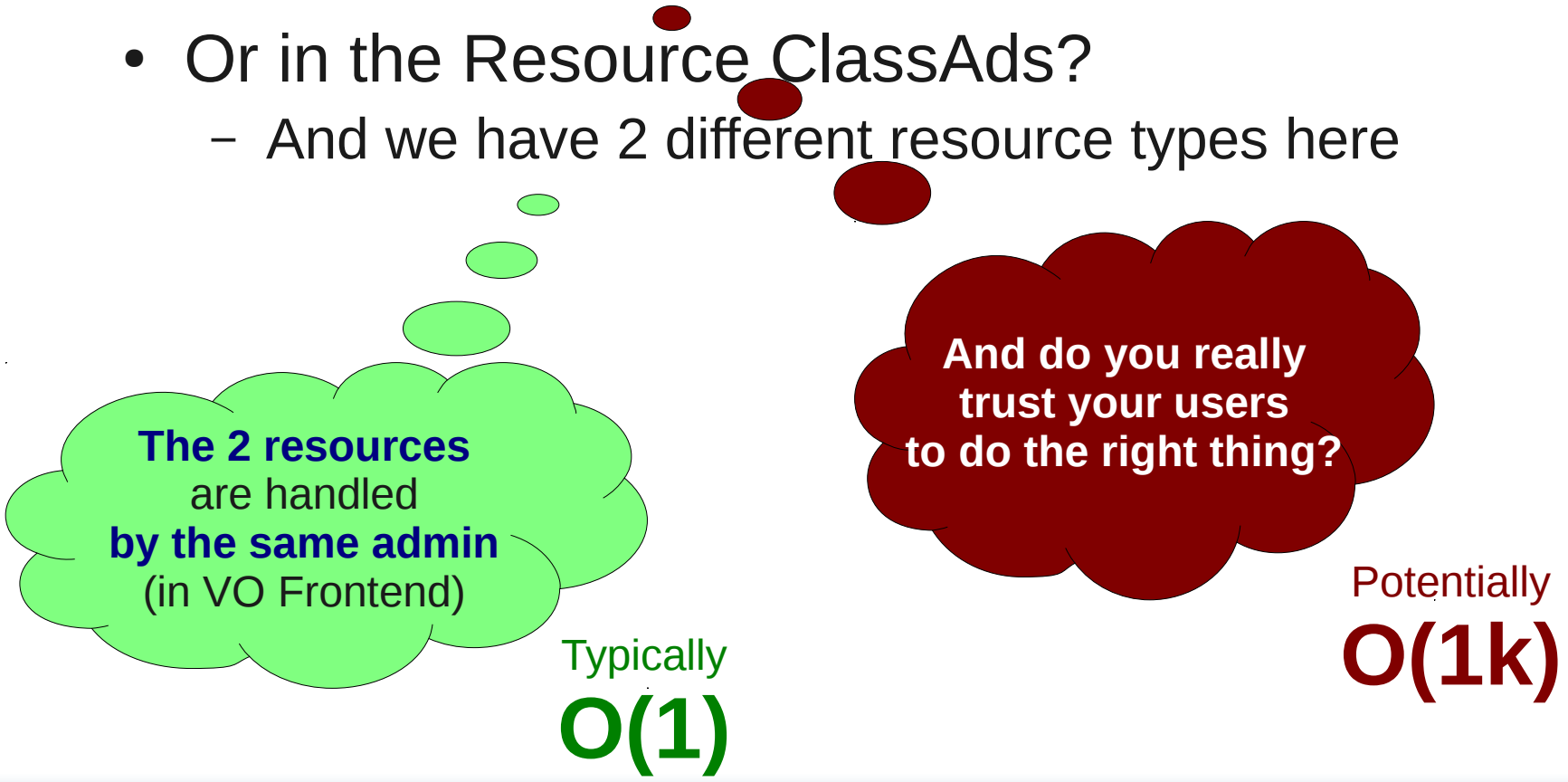
The usual dilemma

- Where do I define these requirements?
  - In user job ClassAd?
  - Or in the Resource ClassAds?
    - And we have 2 different resource types here



The usual dilemma

- Where do I define these requirements?
  - In user job ClassAd?
  - Or in the Resource ClassAds?
    - And we have 2 different resource types here




Moving reqs to the resources

- So we went for setting the requirements in the resources themselves


Moving reqs to the resources

- So we went for setting the requirements in the resources themselves
- But users still need a way to select resources!
  - How do they do it???

Moving reqs to the resources

- So we went for setting the requirements in the resources themselves
- But users still need a way to select resources!
  - How do they do it???

Fixed schema

- The resource provider defines the requirements
  a.k.a startd + VO Frontend
  - The VO Frontend admin in our case
- Those requirements look for **well-defined** user-provided attributes

Site attribute
Machine attribute
Job attribute

Example

```
entry_req   =   stringListMember(GLIDEIN_Site,DESIRED_Sites)&&
                ((GLIDEIN_Min_Mem>DESIRED_Mem)=!=False)

Start       =   stringListMember(GLIDEIN_Site,DESIRED_Sites)&&
                ((Memory>DESIRED_Mem)=!=False)
```

Simple user job submit file

- No complex requirements to write
- Very little user training
- Low error rate

a.submit

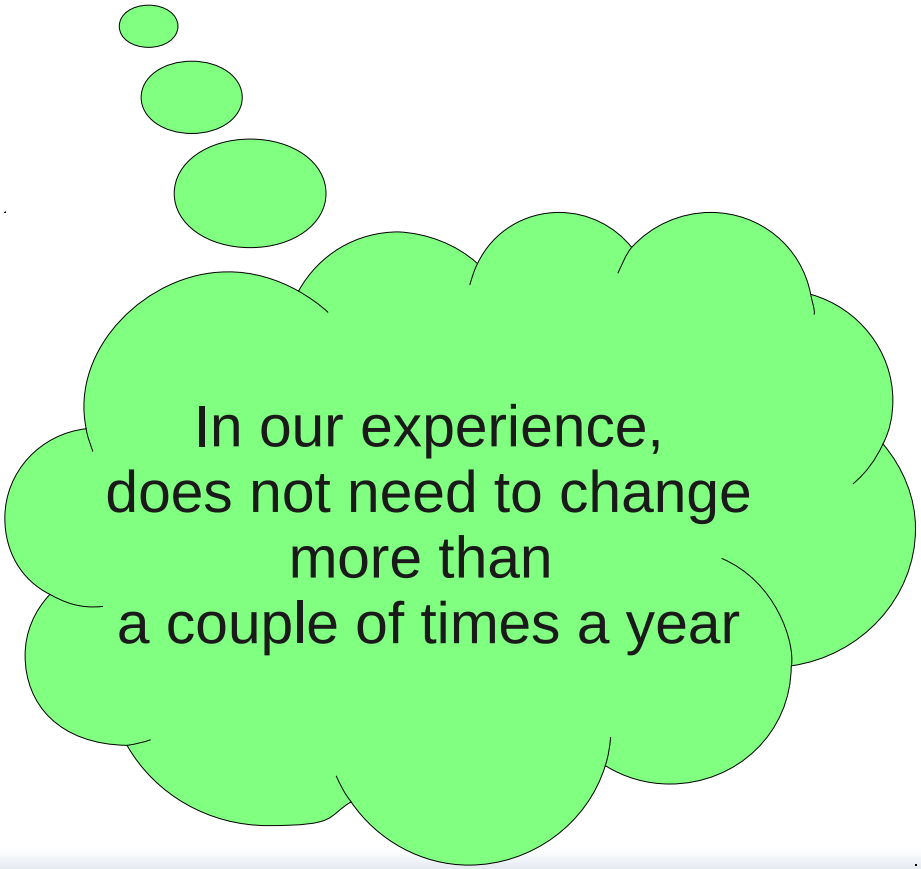
```
Executable = a.sh
Output = a.out
+DESIRED_Sites=”UCSD,Nebraska”
Requirements=True
queue
```

How well does it work?

- Advantages
  - Easy to keep the two levels in sync
  - Easy to define reasonable defaults
  - Easy on the users
  - And more...
    (wait for later slides)
- Disadvantages
  - Rigid schema

How well does it work?

- Advantages
  - Easy to keep the two levels in sync
  - Easy to define reasonable defaults
  - Easy on the users
  - And more...
    (wait for later slides)
- Disadvantages
  - Rigid schema



Is this glideinWMS specific?

- Adva
  - F
    - Ie

Don't think so!

- Disadvantages
  - Rigid schema

- **Easy to define reasonable defaults**
- **Easy on the users**

The ease of use is there for any Condor setup

Is this glideinWMS specific?

- Adva
  - F
    - Is

Don't think so!

- Disadvantages
  - Rigid schema

- **Easy to define**

I would argue it should become standard practice

The ease of use is there for any Condor setup


And there is still more to it

Side effect

- Discovered one unexpected nice side effect

Side effect

- Discovered one unexpected nice side effect

The overflow use case

- Normally, CMS jobs run near the data
  - So users provide a whitelist of sites to run on
  - And we have the appropriate glideinWMS expression

a.submit

```
Executable = a.sh
Output = a.out
+DESIRED_Sites=”UCSD,Nebraska”
Requirements=True
queue
```

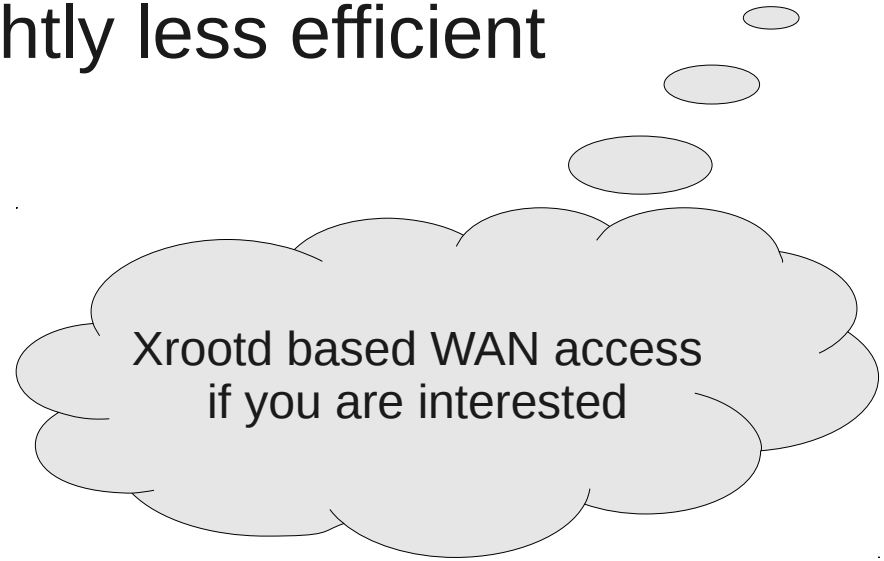
Examp

```
DEIN_Site,DESIRED_Sites)&&
>DESIRED_Mem)=!=False)

Start    =    stringListMember(GLIDEIN_Site,DESIRED_Sites)&&
              ((Memory>DESIRED_Mem)=!=False)
```

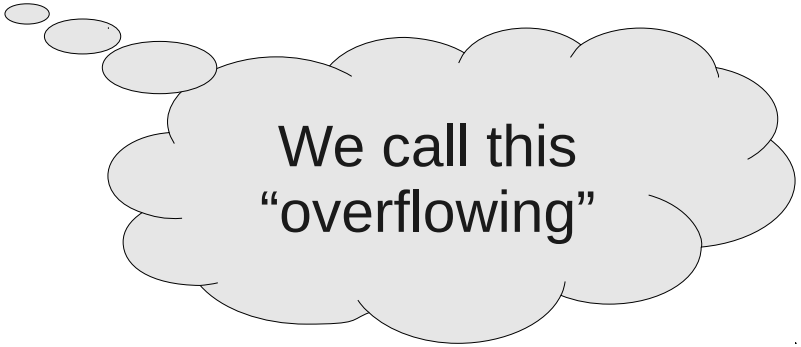
The overflow use case

- Normally, CMS jobs run near the data
- But some jobs could run over the WAN
  - It is just slightly less efficient



The overflow use case

- Normally, CMS jobs run near the data
- Jobs could run over the WAN
  - It is just slightly less efficient
- But if some CPUs are idle due to low demand
  - **A low efficiency job is still better than no job!**
  - As long as it results in users getting their results sooner
    - i.e. only if "optimal resources" are not available



The overflow use case

- Normally, CMS jobs run near the data
- Jobs could run over the WAN
  - It is just slightly less efficient
- But if some CPUs are idle due to low demand
  - **A low efficiency job is still better than no job!**
  - As long as it results in users getting their results sooner
    - i.e. only if "optimal resources" are not avail

Overflow configuration

- Essentially, we change the rules!
  - Without involving the users
- We write the requirements based on where the data is, not where the CPUs are
  - Since not all sites have xrootd installed

Example

```
entry_req   =   ((CurrentTime-QDate)>21600)&&(Country=?="US")&&
                stringListsIntersect(DESIRED_Sites,"UCSD,Wisc")&&
                ((GLIDEIN_Min_Mem>DESIRED_Mem)=!=False)
Start       =   ((CurrentTime-QDate)>21600)&&
                stringListsIntersect(DESIRED_Sites,"UCSD,Wisc")&&
                ((Memory>DESIRED_Mem)=!=False)
```

Overflow configuration

- Essentially, we change the rule
  - Without involving the users
- We write the requirements ba Os are

No change to the user submit file!

a.submit

```
Executable = a.sh
Output = a.out
+DESIRED_Sites=”UCSD,Nebraska”
Requirements=True
queue
```

bootd installed

Exam

```
Start   =   ((CurrentTime-QDate)>21600)&&(Country=?=”US”)&&
            ESIRED_Sites,”UCSD,Wisc”)&&
            >DESIRED_Mem)=!=False)
```

```
Start   =   ((CurrentTime-QDate)>21600)&&
            stringListsIntersect(DESIRED_Sites,”UCSD,Wisc”)&&
            ((Memory>DESIRED_Mem)=!=False)
```

Overflow configuration

And we can decide where to overflow from hours after the jobs were submitted

No change to the user submit file!

e the rule

sers

ements ba

Os are

botd installed

```
Executable = a.sh
Output = a.out
+DESIRED_Sites="UCSD,Nebraska"
Requirements=True
queue
```

"UCSD,Wisc"

Exam

```
Start   =   ((CurrentTime-QDate)>21600)&&
            stringListsIntersect(DESIRED_Sites,"UCSD,Wisc")&&
            ((Memory>DESIRED_Mem)=!=False)
```

Looking at the future


Looking at the future

- The **attribute schema now a fixed one**
  - At least regarding matchmaking
  - Opens up new interesting possibilities


Looking at the future

- The **attribute schema now a fixed one**
  - At least regarding matchmaking
- **Can consider RDBMS techniques**
  - Example uses
    - RDMBS driven matchmaking
    - Tracking of job and/or machine history in a DB

Looking at the future

- The **attribute schema now a fixed one**
  - At least regarding matchmaking
- **Can consider RDBMS techniques**
  - Example uses
    - RDMBS driven matchmaking
    - Tracking of job and/or machine history in a DB
  - Will likely need code changes in Condor
    - UCSD committed to try it our in the near future
    - If the results end up as good as hoped for, will push for official adoption

Conclusions

- Matchmaking is made of requirements
  - But there are two places where they can be defined
- Usually, users expected to set requirements
  - CMS glideinWMS setup moves it completely into the resource domain
- Experience with **no-user-req** setup very positive
  - **Advocating that this should be the recommended way for all Condor deployments**
- Also opens up interesting new venues


Acknowledgments

- This work is partially sponsored by
  - the US National Science Foundation under Grants No. PHY-1104549 (AAA) and PHY-0612805 (CMS Maintenance & Operations)

    and
  - the US Department of Energy under Grant No. DE-FC02-06ER41436 subcontract No. 647F290 (OSG).