Distributed resource management and glide-ins

Wed July 21st, 1:15pm

Igor Sfiligoi, isfiligoi@ucsd.edu

OSG Scalability Area coordinator and
OSG glideinWMS factory manager

University of California San Diego

What your learned until now

- Scientific problems need a lot of CPU time
  - Using a batch system like Condor is a must
- Local compute resources are often not enough
  - Get some/most of the needed CPU cycles from the Grid

Moving to the Grid

- What resources are out there?
- How do I know if they have the needed software installed?
- How do I know where will my jobs finish the soonest?

Open Science Grid

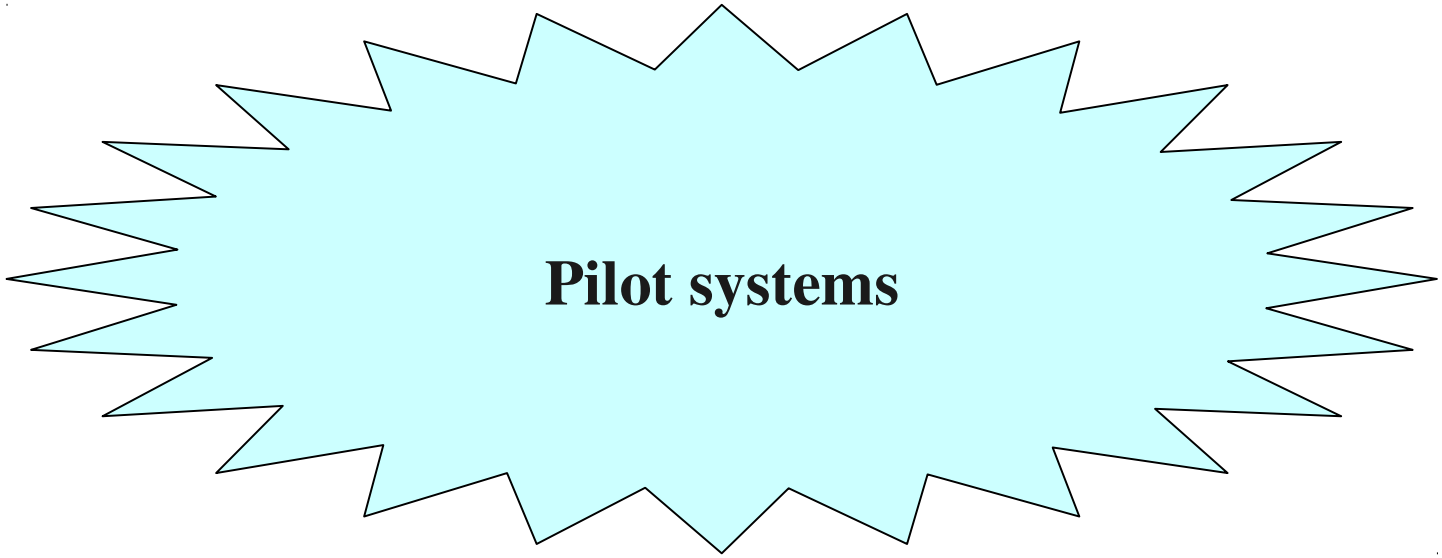
Using OSG MM/Condor-G

- Limited functionality
  - Need to specify output file names
  - Limited monitoring
  - No checkpointing
- Very heterogeneous environment
  - Discover which apps are installed
  - Discover which libs are installed

Can we make life easier?

- More Condor (without G) like?

Yes



Open Science Grid

What is a pilot system?

- An infrastructure that creates a virtual-private batch system
  - It hides “the Grid” from the users
- For users like a local system
- All Grid interaction taken care of in a separate layer (hidden from users)

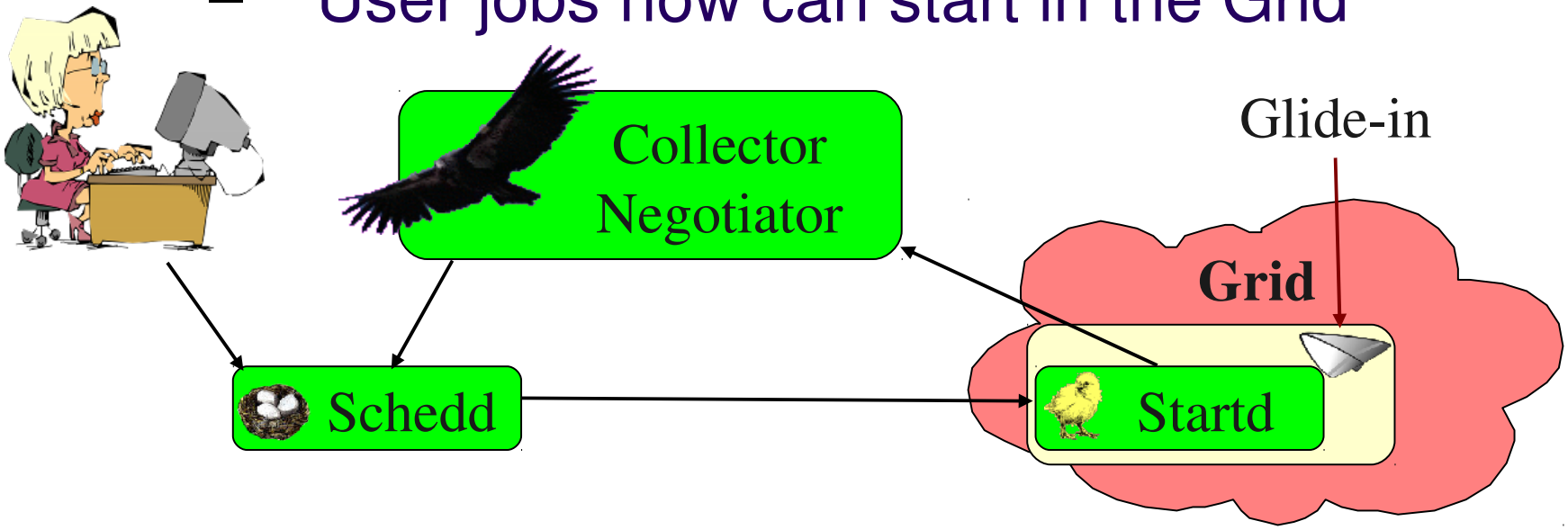
Open Science Grid

Pilot principles

- Never send user jobs directly to the Grid
  - Send pilots instead
- After a pilot lands on a Grid resource
  - Detect local resources
  - Stop execution if resource is bad
- Pilot fetches a user job
  - Fetch currently the most important job (late binding)
  - Start and monitor the job

Glide-ins

- Glidein-ins are Condor-based pilots
- A glide-in is a Condor startd sent as a Grid job
  - When it starts, it joins the Condor pool
  - User jobs now can start in the Grid



Glide-ins (2)

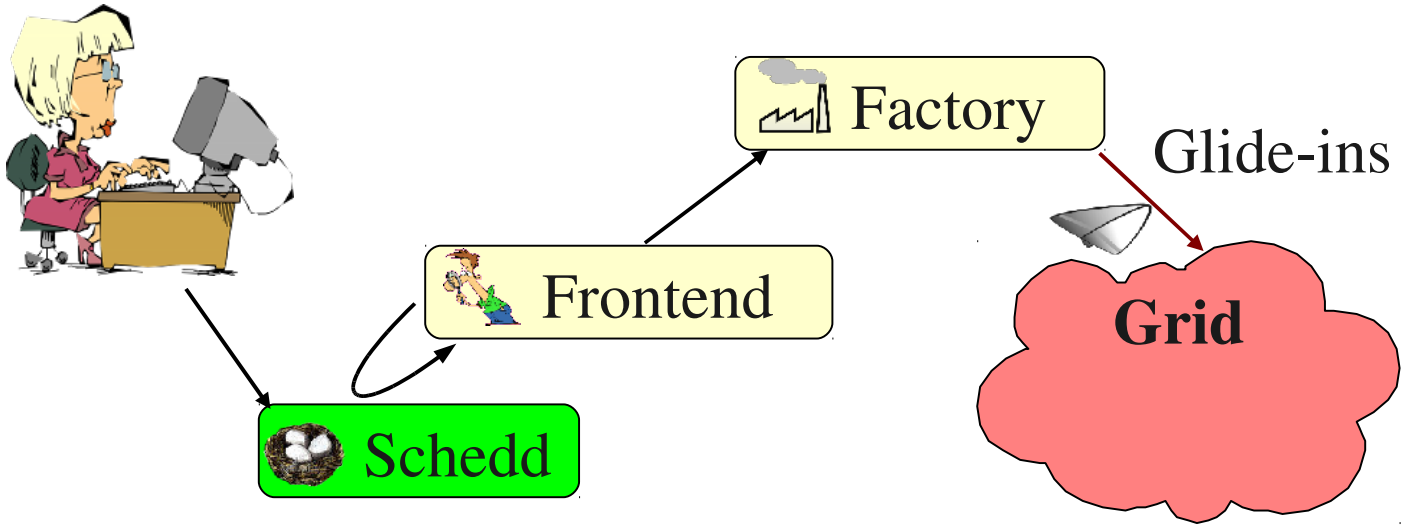
- From the user point of view, just a regular Condor pool
  - Like the one you learned about Mon morning
- All Condor functionality available
  - Detailed matchmaking
  - Detailed monitoring
  - Can use checkpointing

glideinWMS

- Someone has to submit the glideins to the Grid resources
  - User could do it him/herself, but that can be tedious
- In OSG users can use the glideinWMS
  - Provides for an automated glidein submission system

glideinWMS (2)

- Split in two pieces
  - A glidein factory (submits glideins)
  - A VO frontend (regulates how many)



- Glideins are submitted when idle user jobs in a schedd found

GlideinWMS (3)

- Every community/VO supposed to host its own Condor system and frontend
  - Knows best its own user needs
  - Provides the Grid credentials needed to submit the pilots
- Factory can be shared
  - OSG hosts a glidein factory at UCSD
  - Able to submit glideins both to OSG as well as European grids

Other pilot systems

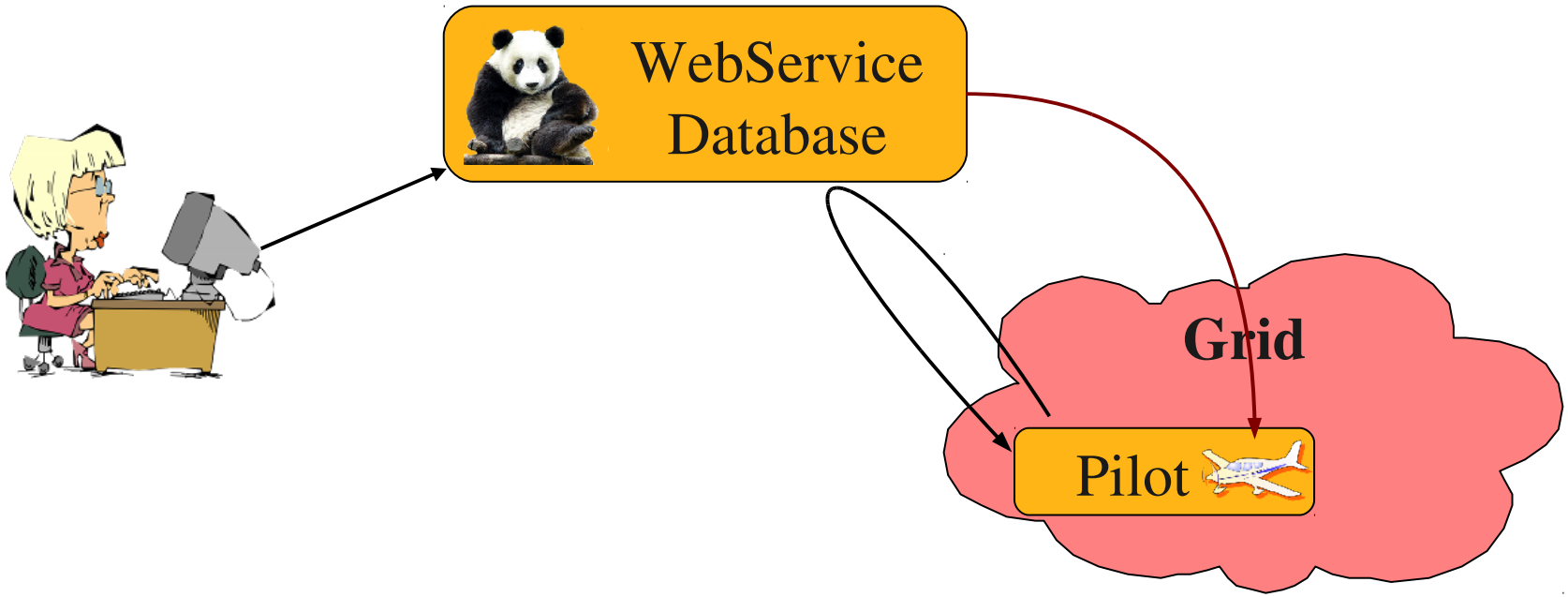
- Two other major pilot systems used in the US
  - Corral
  - PANDA

Corral

- Based on glide-ins
  - Similar to glideinWMS
- Uses explicit glidein provisioning
  - i.e. users must explicitly ask for a certain number of glideins
  - Unlike glideinWMS automated requests
- Mostly used on Teragrid resources
  - Fits that operation mode better

PANDA

- A Web Services based pilot system
  - Database driven
- Same concept, different implementation



Pilot system drawbacks

- Requires networking from the worker nodes
  - Pilots need to “call home”
  - Regular Grid jobs can live without
- Potentially wasting resources
  - When a pilot starts, there may not be any more user jobs waiting to be run

Open Science Grid

Part 1 Summary

- Pilot systems can hide most of the Grid complexity from the users
- Glide-ins are a Condor implementation of a pilot system
  - With glideinWMS providing automation
- Other pilot systems available

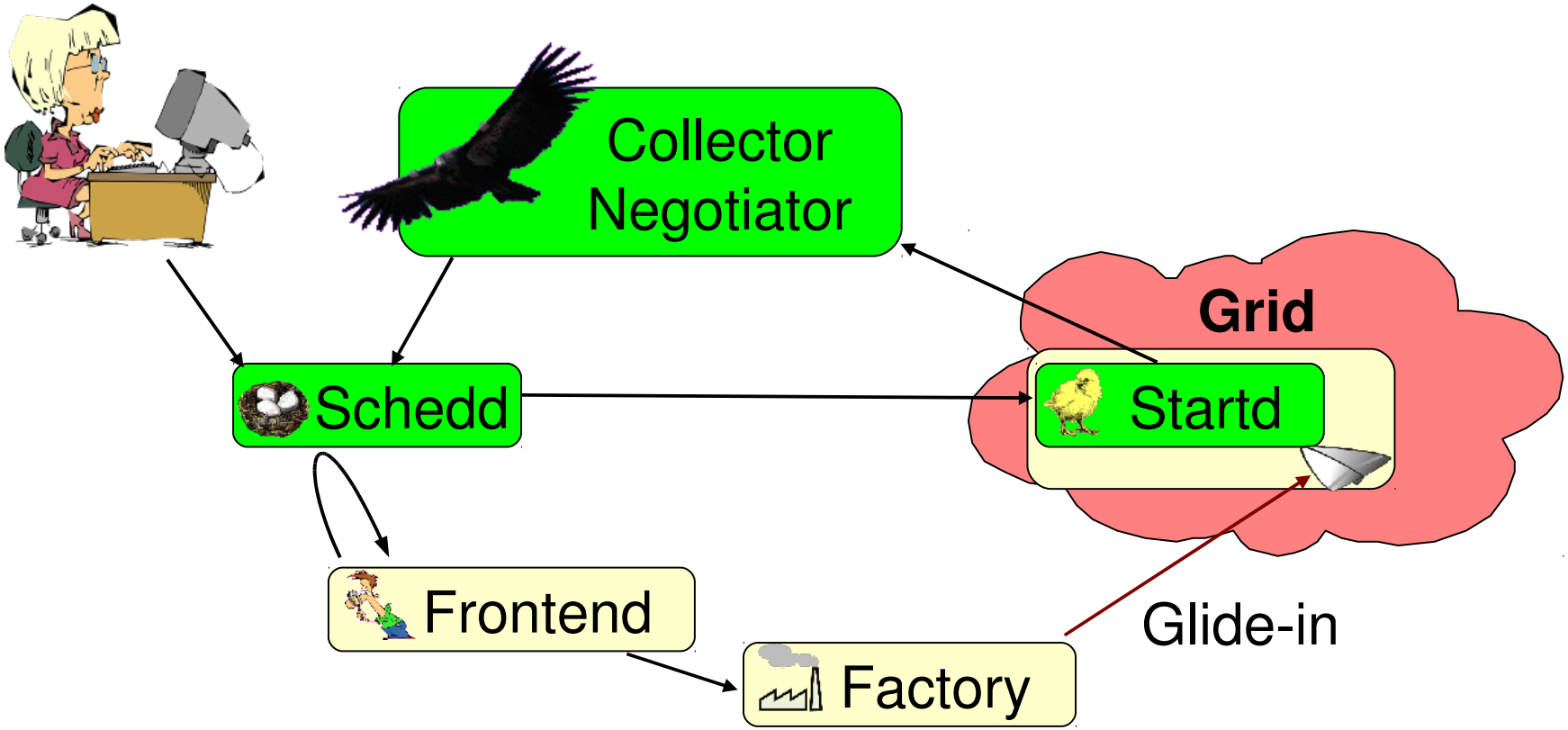
Open Science Grid

Part 2

A closer look at glideinWMS

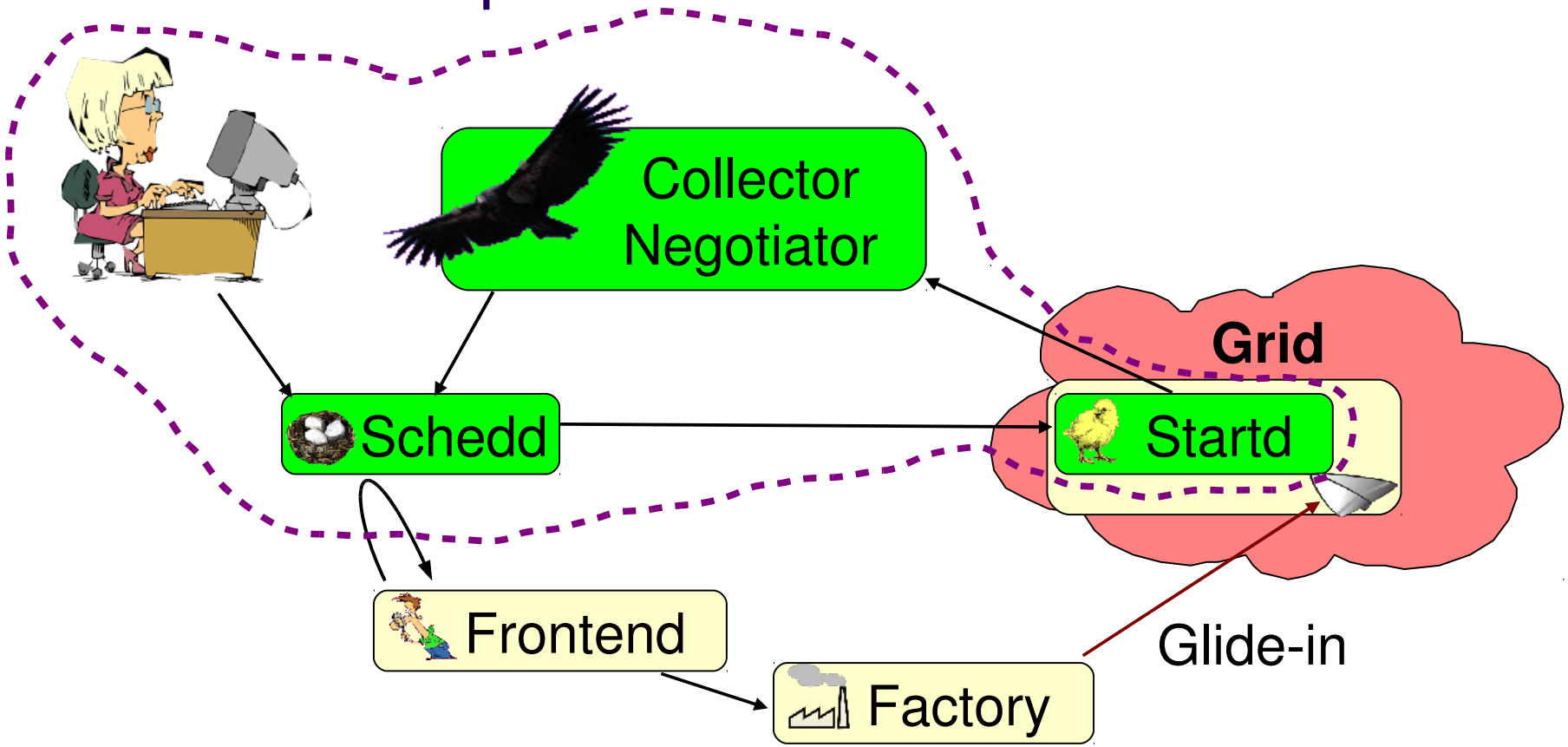
glideinWMS

- At a glance



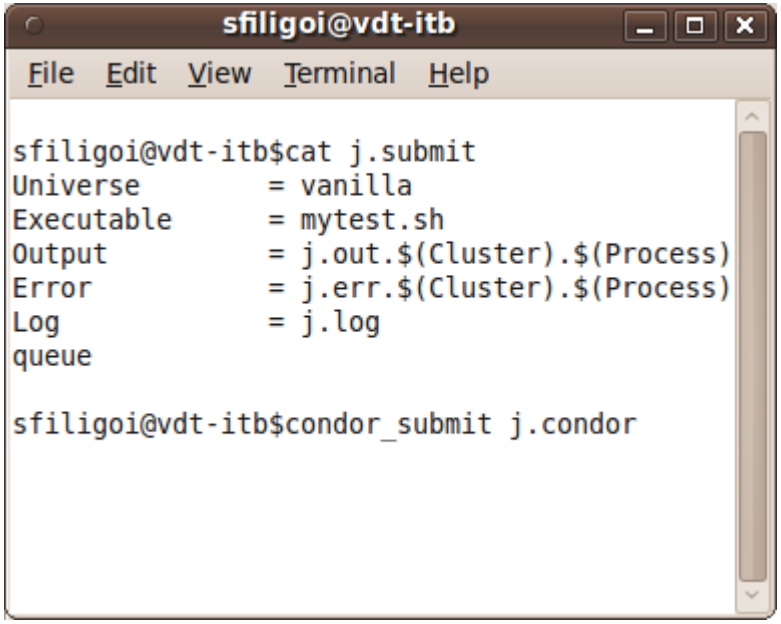
glideinWMS – for the user

- The user really only sees the regular Condor pool

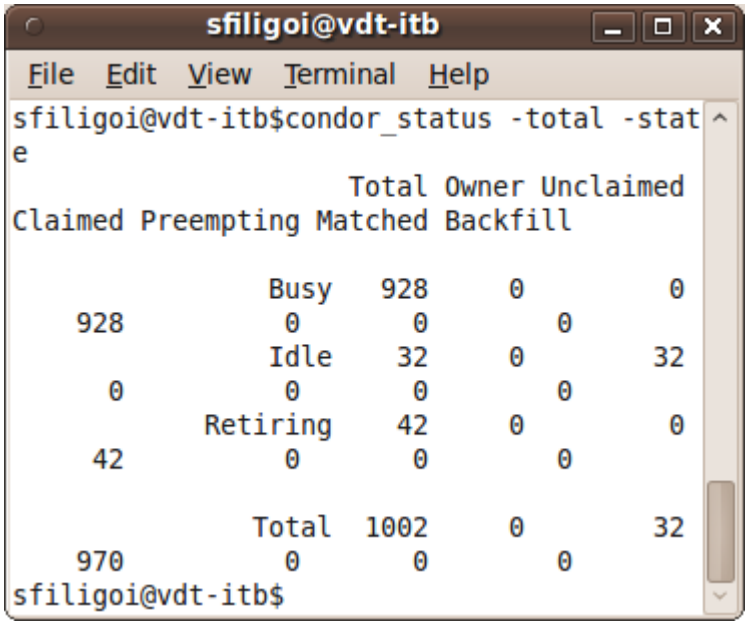


Just a regular-looking pool

- Submit vanilla jobs

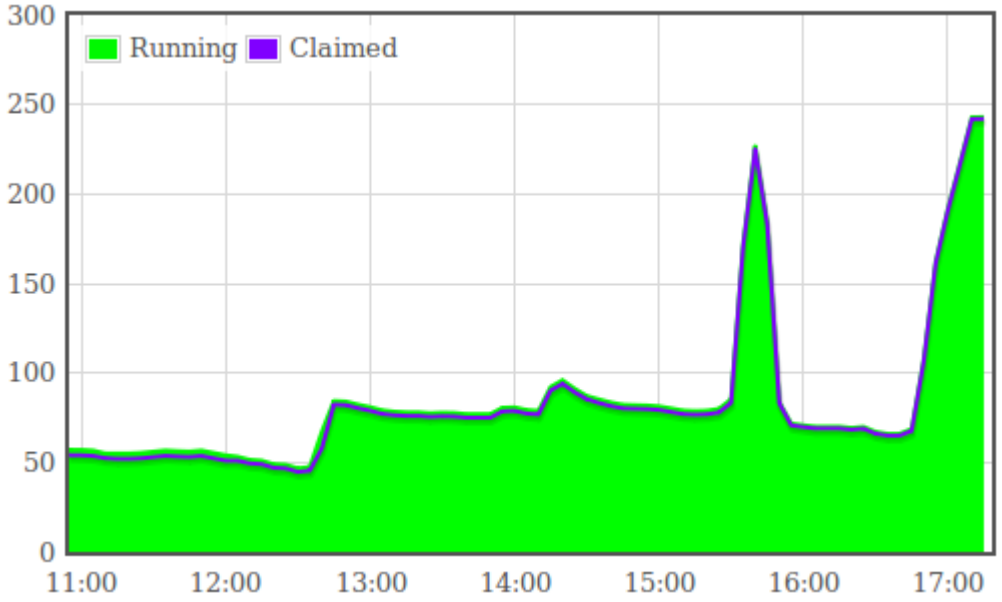


- Monitor resources



Just a dynamic one

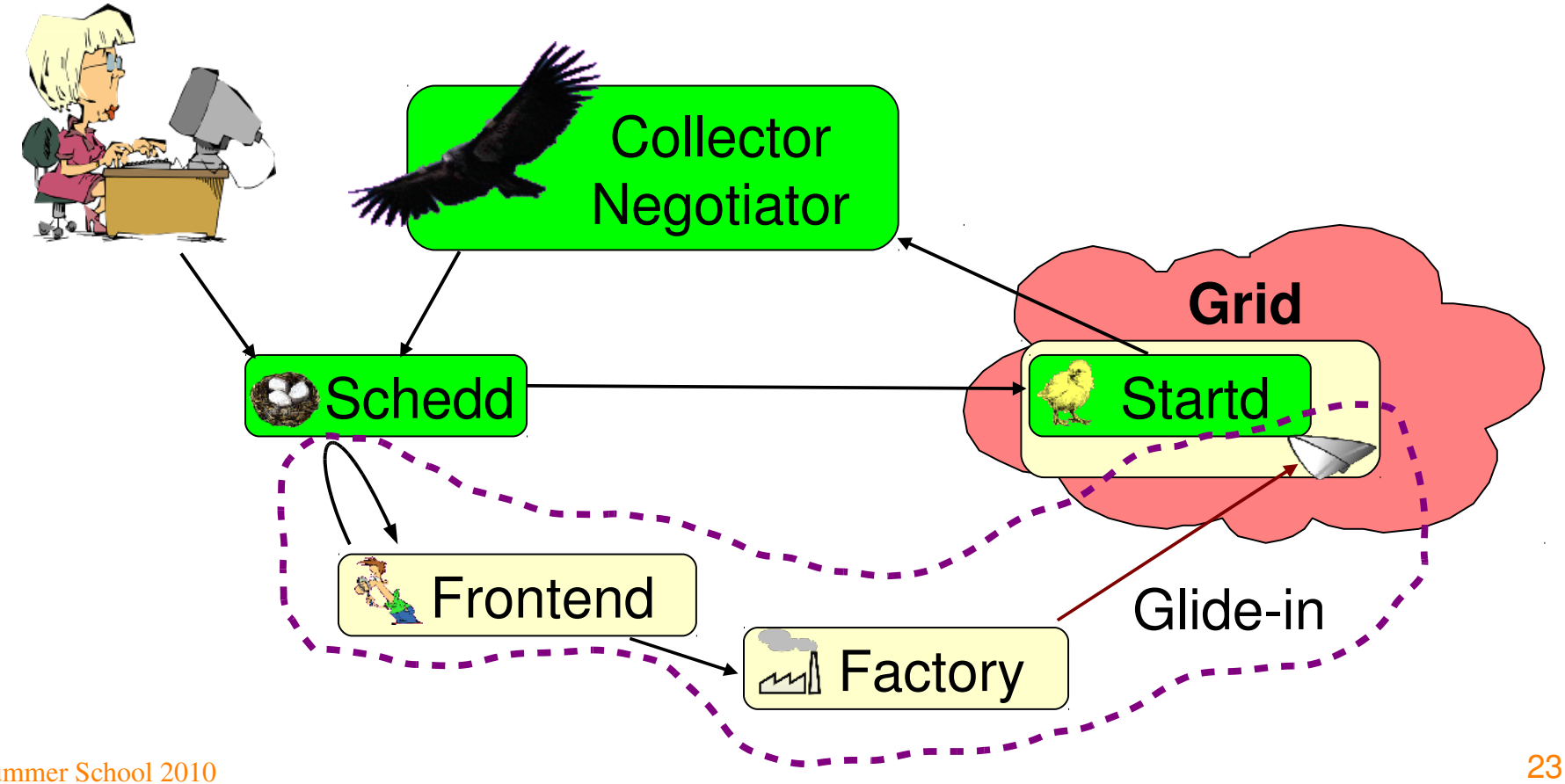
- Resources come and go



Courtesy of the UCSD CMS frontend

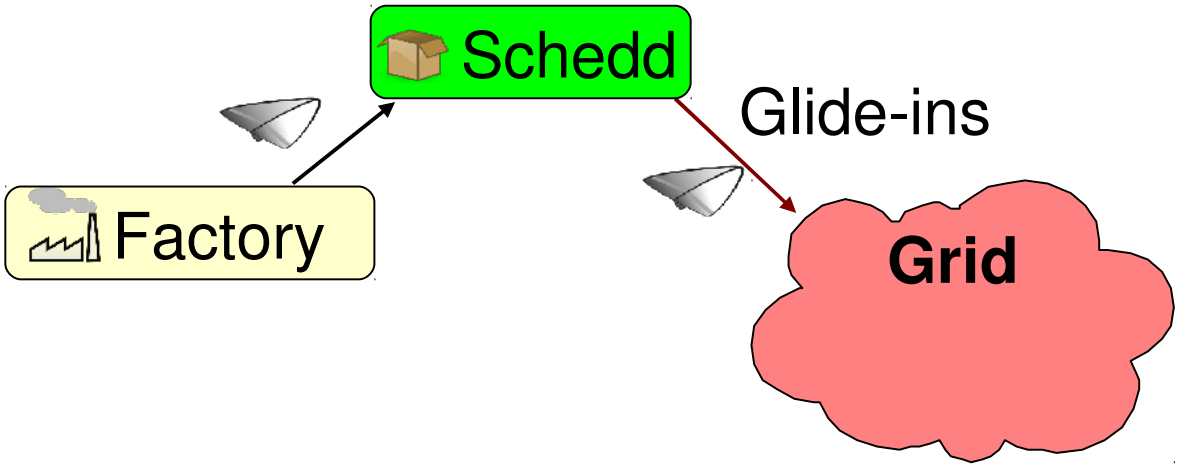
glideinWMS backend

- glideinWMS processes make sure users have resources when needed



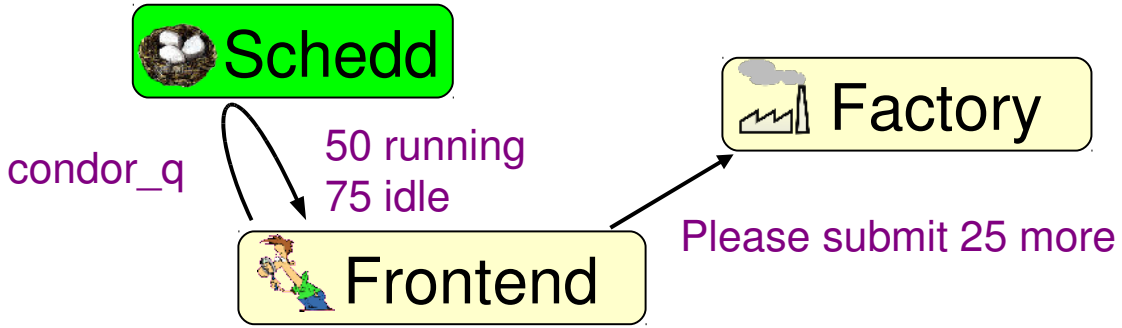
Glidein Factory

- The glidein factory actually submits the Grid jobs – the glideins
- Uses Condor-G for the task



Frontend

- The frontend decides how many glideins to submit
- The decision based on how many jobs need more resources and where



The glideins

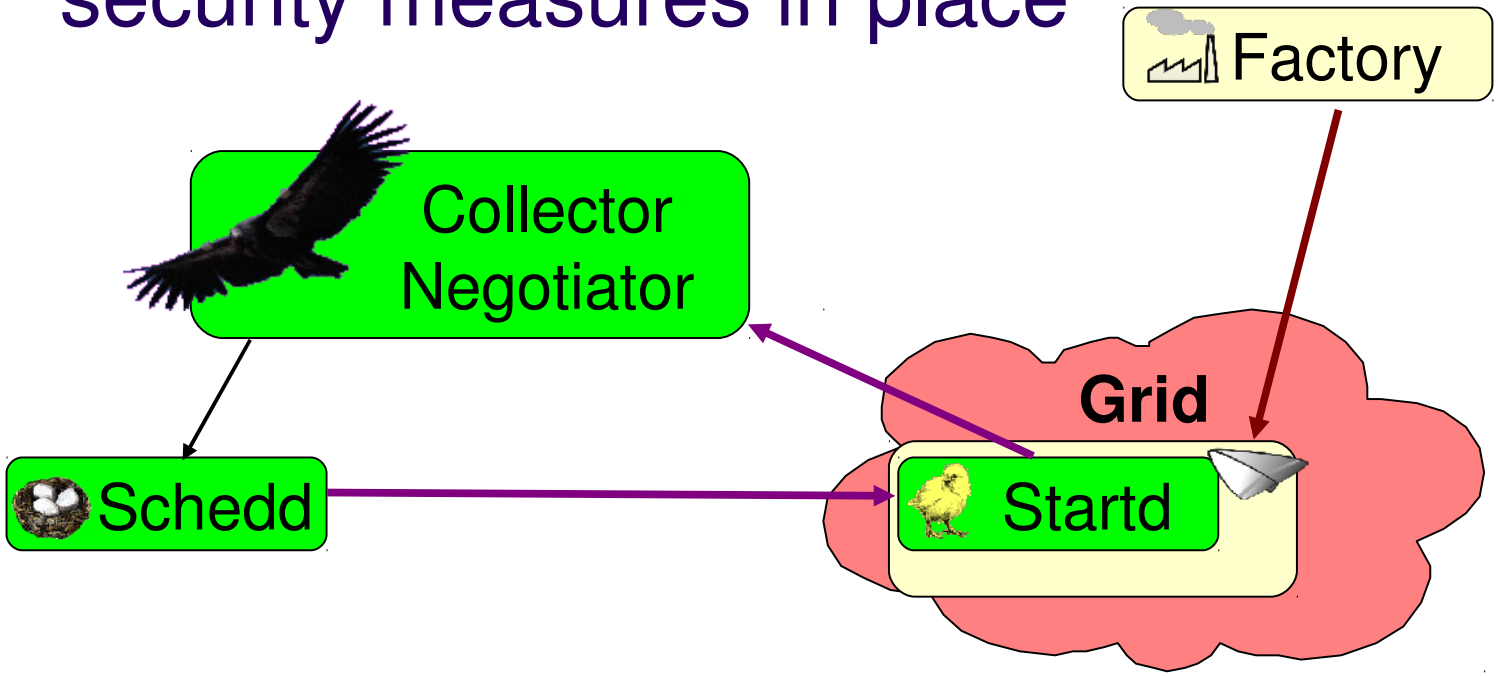
- Glideins are regular Grid jobs
  - Nothing special about this
- Will configure a Condor instance
  - Get the binaries
  - Discover/validate WN environment
  - Create condor config
- Finally start a Condor startd
  - Condor takes care of itself after that

The glideins (2)

- Glideins are not specific to a user
  - The startd will pick the highest priority job that matches
  - Just like in a regular pool
- A glidein can run many jobs
  - Possibly from many users
  - Just like in a regular pool
- Uses its own credential for Grid submission
  - Usually a service proxy

Security considerations

- Traffic to and from a Grid site flows over untrusted Internet
- All WAN connections must have security measures in place



Security considerations (2)

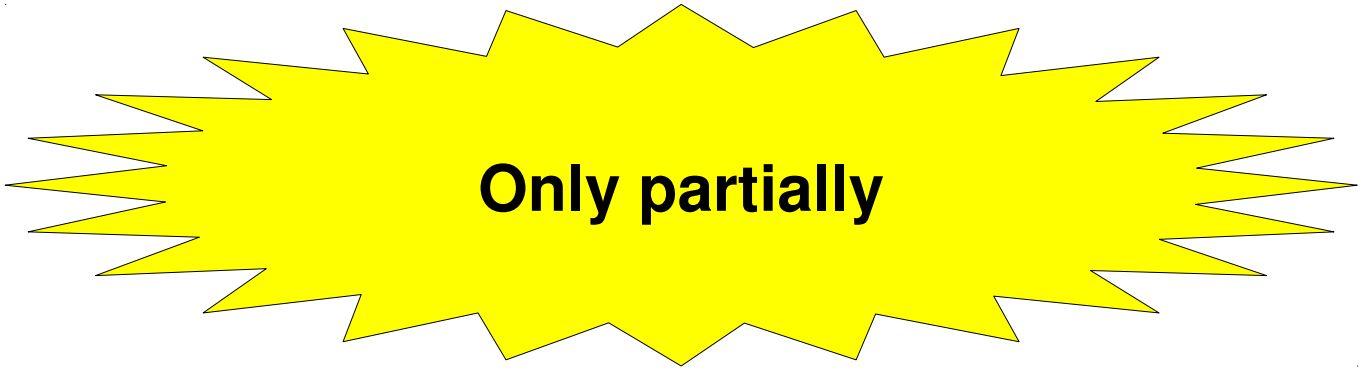
- The glidein will have its own proxy when landing
  - Will use it to authenticate to the rest of the system
- The other processes have a certificate as well
  - Mutual authentication
- Using whitelist authorization
- Also enabling integrity checks in Condor

Part 3

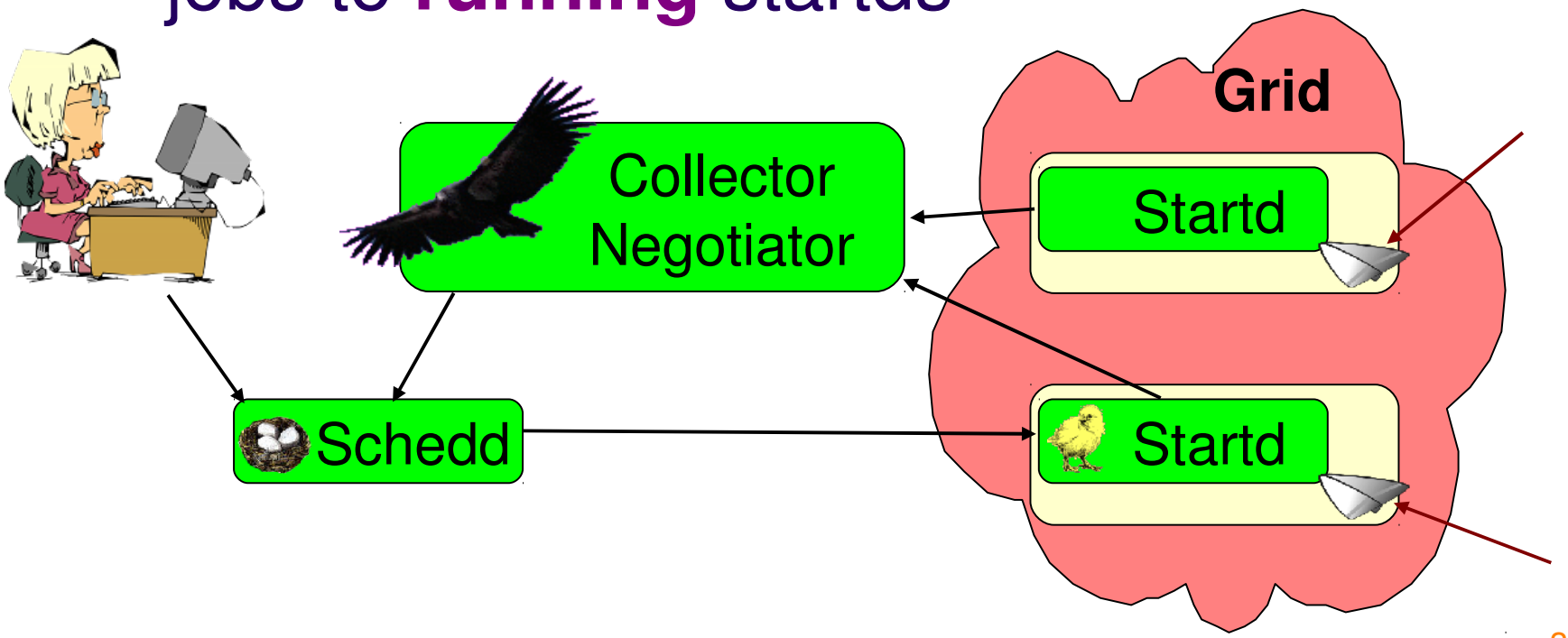
Resource selection

Anything glideinWMS specific?

- From the user's point of view,
  isn't this just a regular Condor pool?
- Can't I just set the
  "**requirements**"
  and be done?
  (i.e. the negotiator will take care of it)

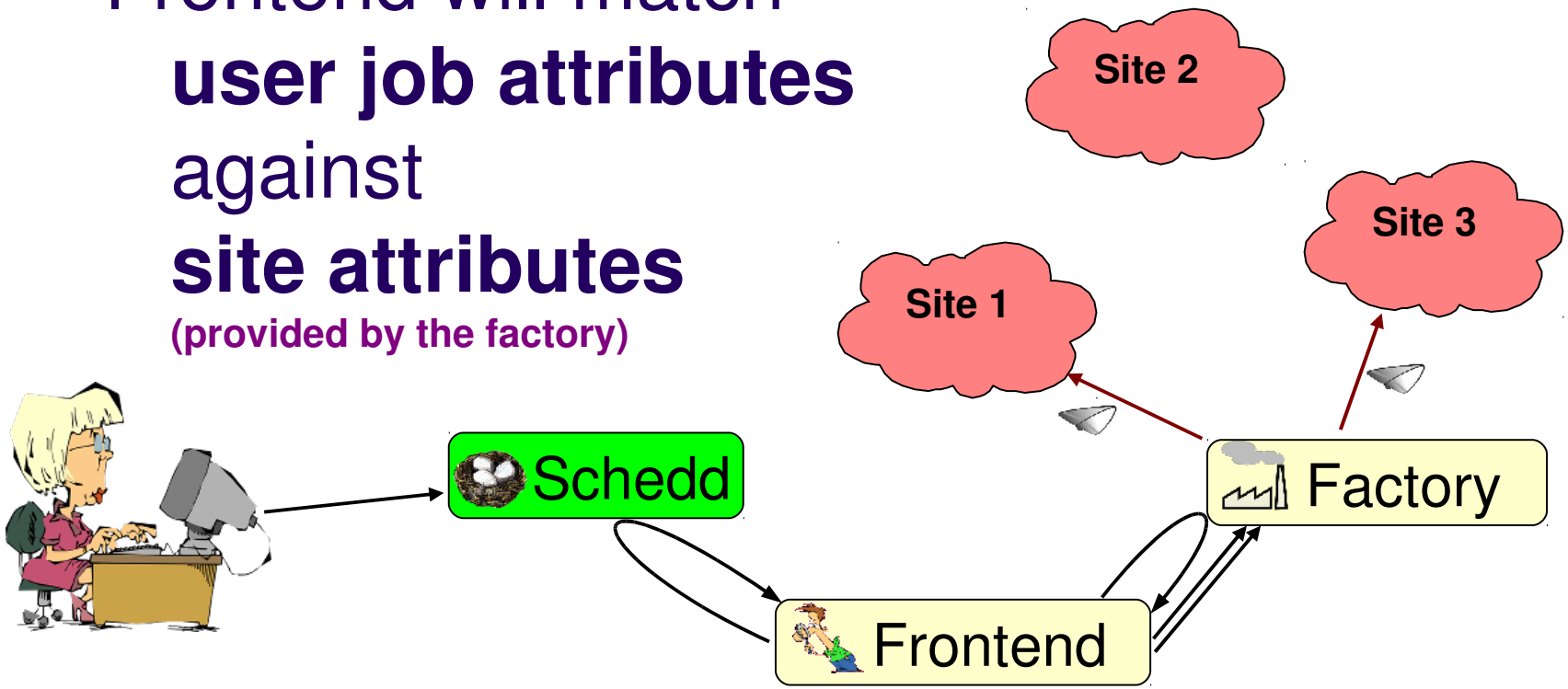


- The "**requirements**"
  only used by the negotiator
- The negotiator only matches
  jobs to **running** startds



Where will the glideins run?

- There may be many Grid sites
  - OSG g.factory @UCSD serves ~160
- Frontend will match
  user job attributes
  against
  site attributes
  (provided by the factory)

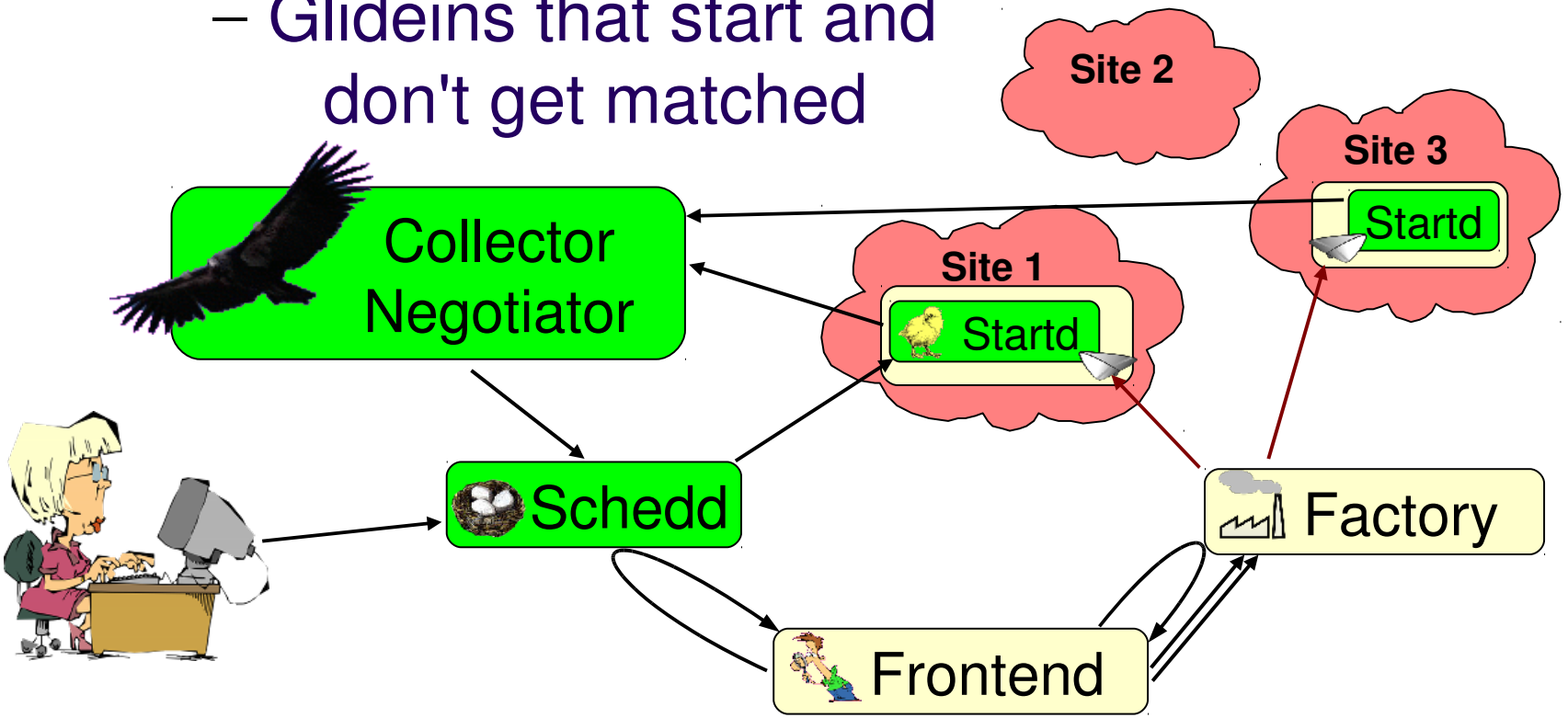


Frontend matchmaking

- The standard "requirements" attribute cannot be used by the frontend
  - The glidein factory attributes are likely different than the glidein-startd attributes
- The factory has only generic information about the site
  - Glideins can augment it with information discovered on the worker nodes

Two level matchmaking

- Based on different information
- Can potentially be a problem
  - Glideins that start and don't get matched



An example

- Site selection criteria:
  - Site=!=”Nebraska” && HasBLAST
- Startd requirements:
  - BLAST_version>2 && Disk>100000

Open Science Grid

Parts 2 & 3 summary

- glideinWMS looks almost like a regular Condor pool to the users
  - Most of the Grid details taken care of by the glideinWMS admins
- Users may need to add additional attributes for Grid site selection

Questions?

- Questions? Comments?
- Feel free to ask me questions later:

  Igor Sfiligoi, isfiligoi@ucsd.edu

- Upcoming sessions
  - 2:00pm-3:00pm
    - Hands-on exercises with glide-ins
  - 3:00pm – 3:15pm
    - Break
  - 3:15pm - 5:00pm
    - Dealing with real resources

Get ready for practice

Glide-in hands-on session next