2011 OSG Summer School

An introduction to

Overlay systems

Also known as

Pilot systems

by Igor Sfiligoi
University of California San Diego

Summary of past lessons

- HTC is maximizing CPU use over long periods
  - And getting lots of computation done
- DHTC is HTC over many sites
- Grid sites have a CE with an abstract API
- Direct Grid submission requires job partitioning
  - Job partitioning is hard

Overlay systems

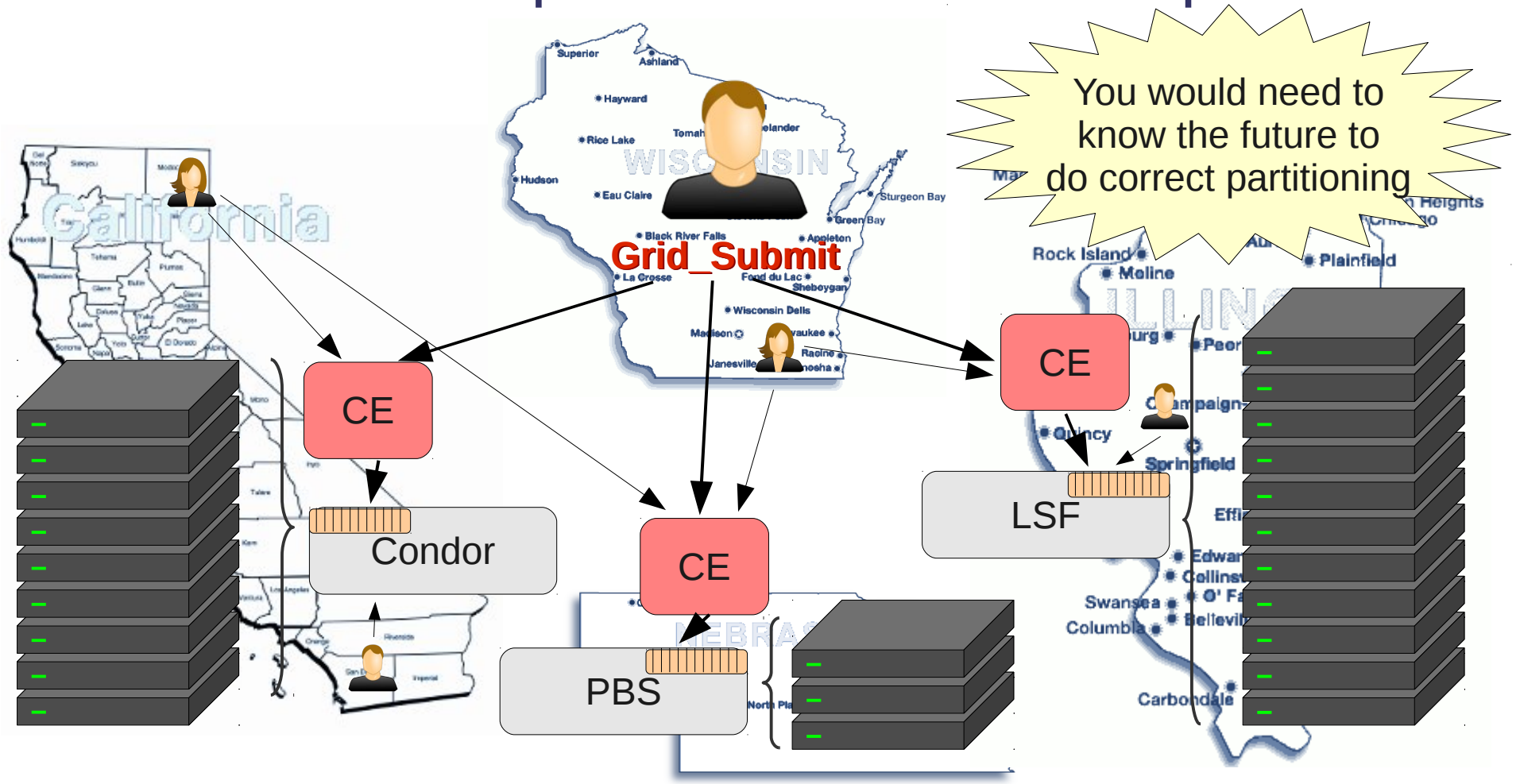
Introduction to

Overlay systems

in the DHTC context

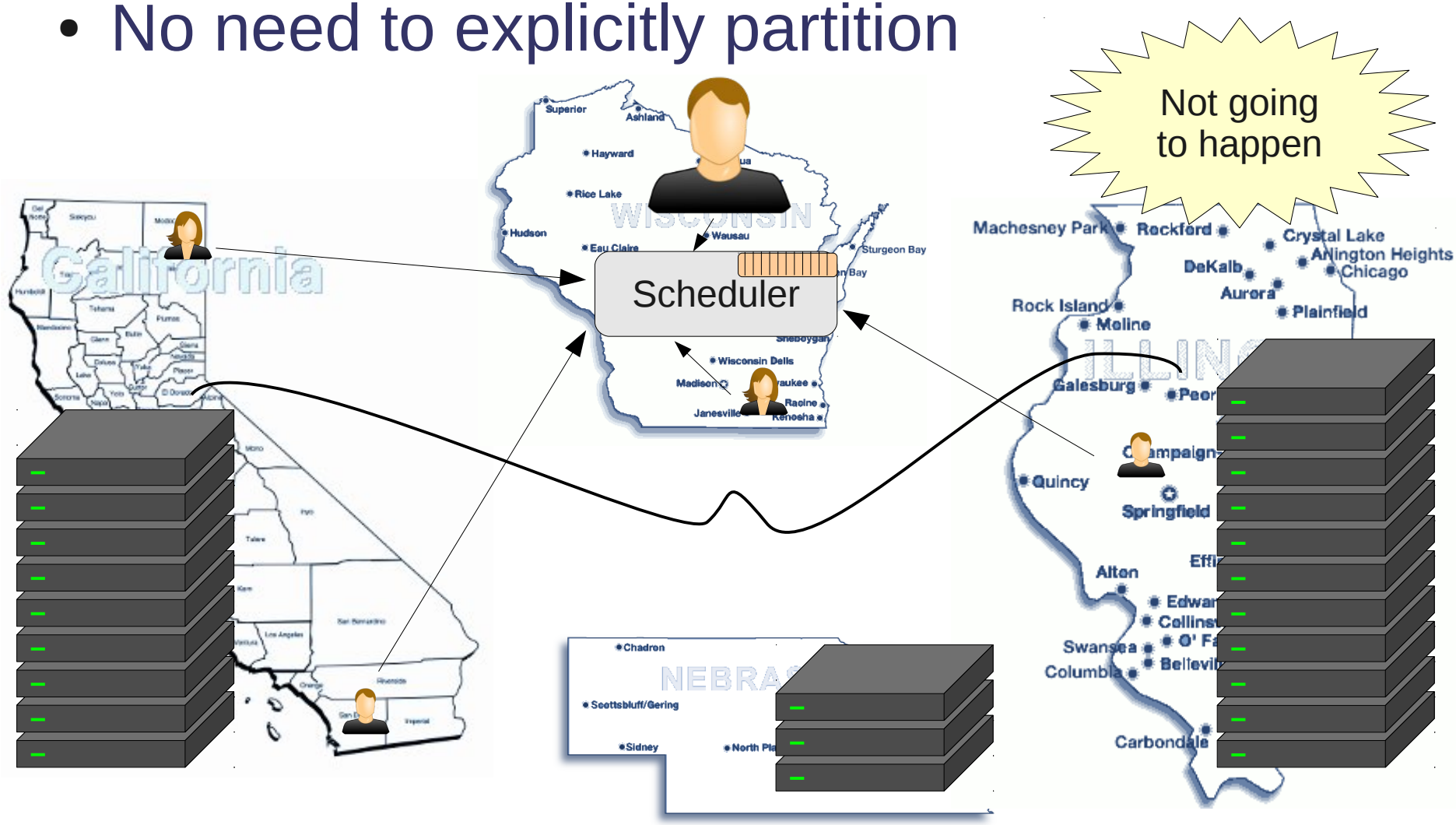
Why is job partitioning hard?

- Intermediate queues with unknown policies



If only we could have a global scheduler

- No need to explicitly partition

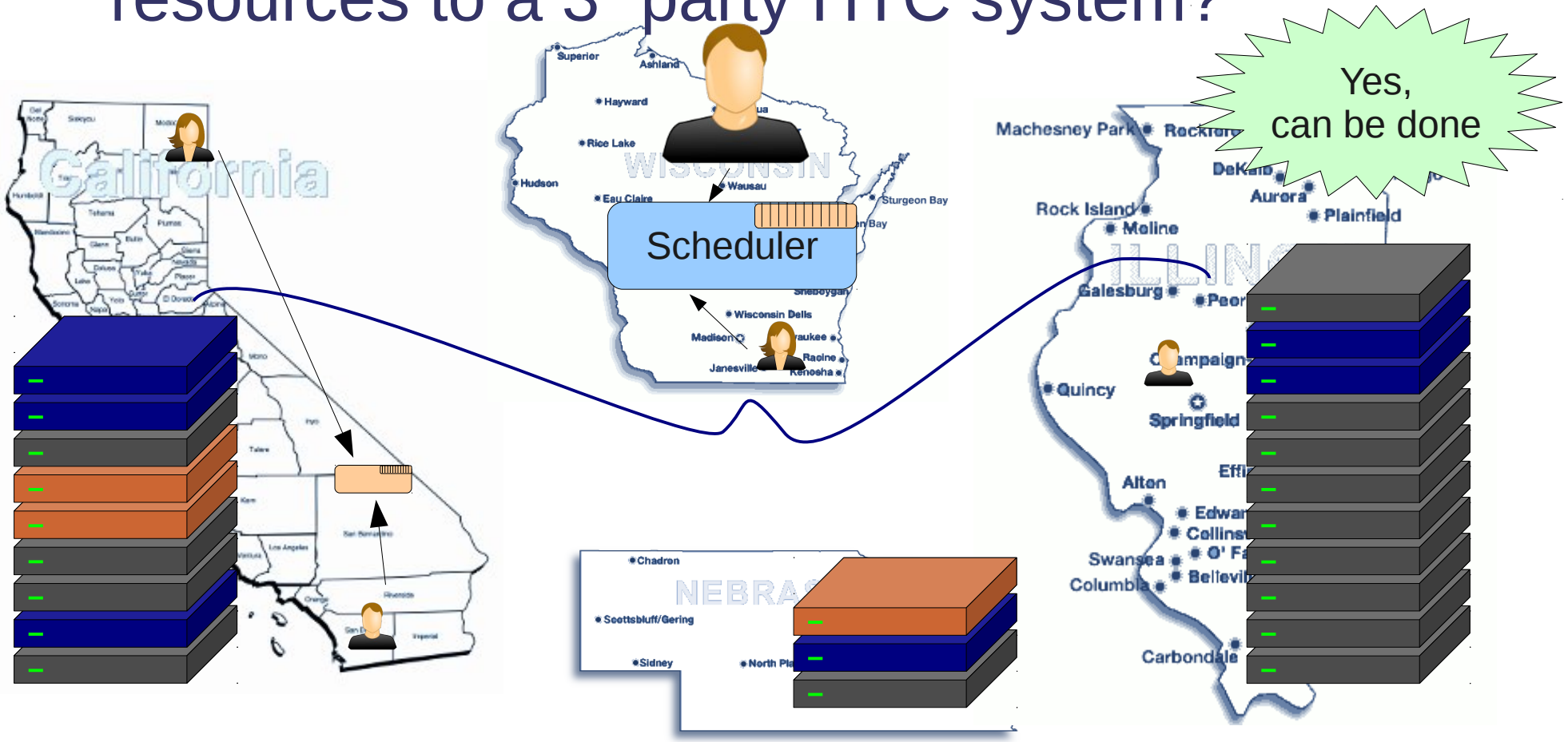


Why we cannot have a global scheduler

- Existing infrastructure
- Local users, local policies
- Money & politics
- Being able to work when WAN goes down
- ...

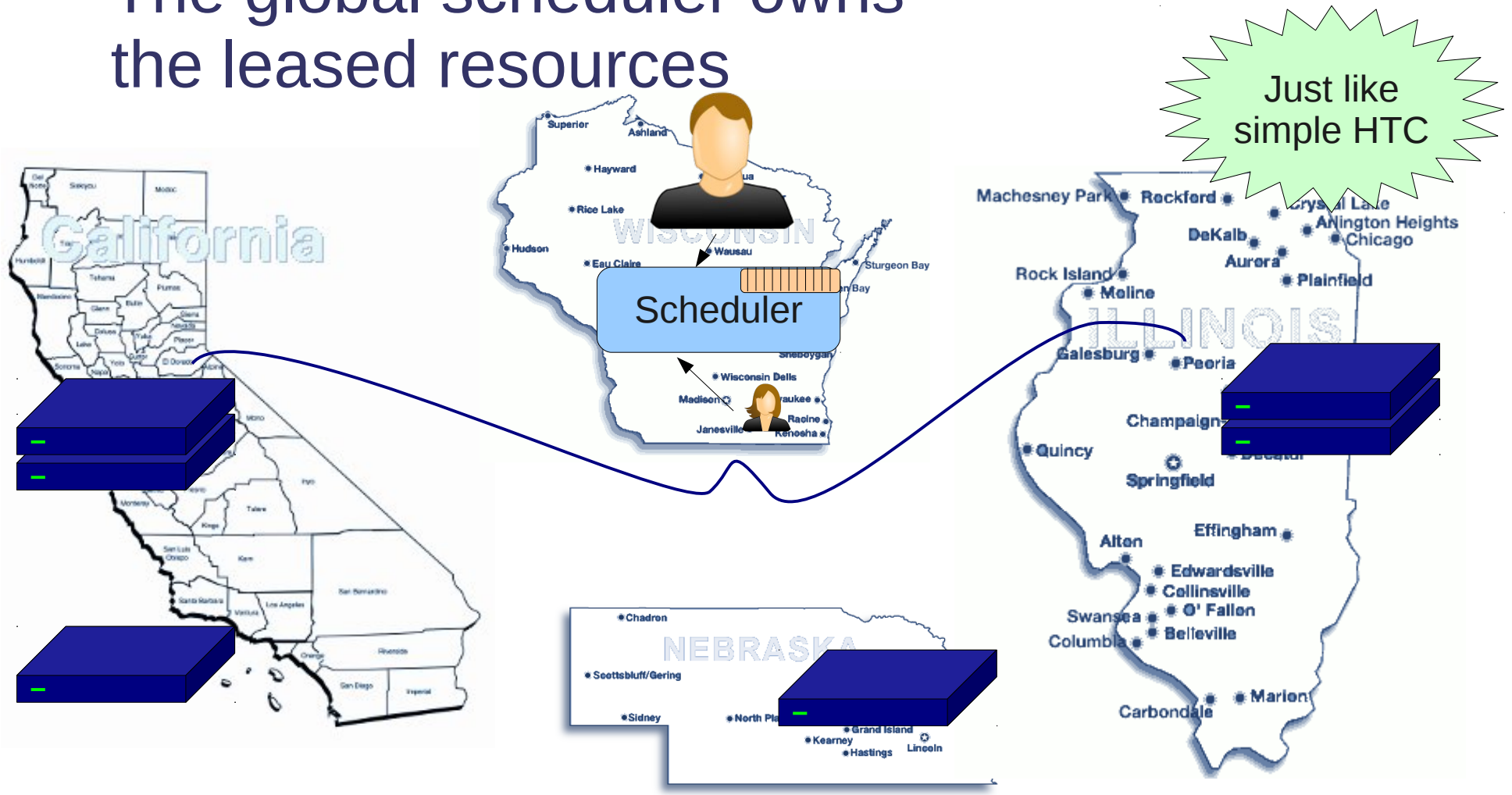
What about a subset?

- Can we convince the sites to lease some of the resources to a 3rd party HTC system?



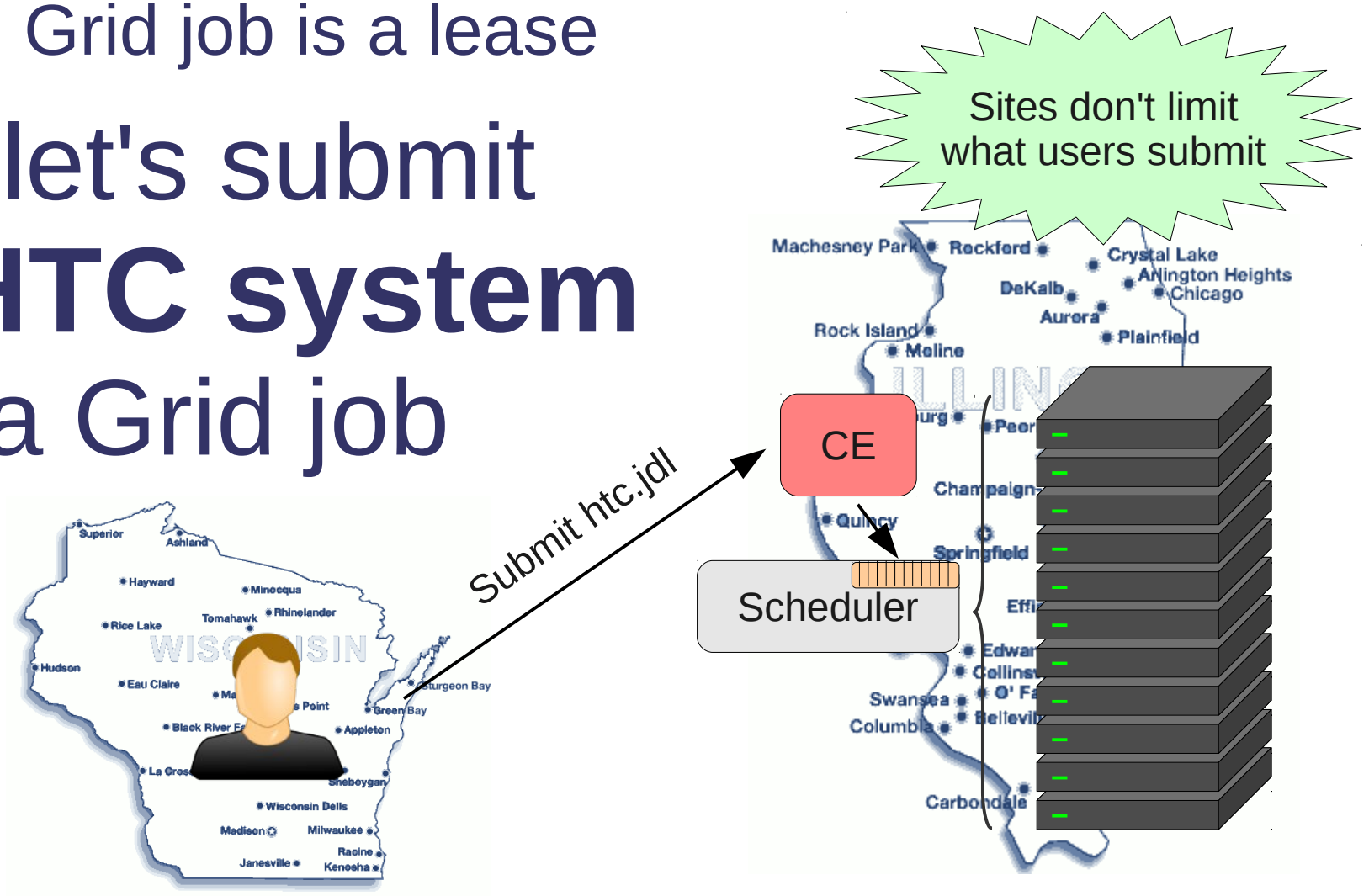
Resource leasing

- The global scheduler owns the leased resources



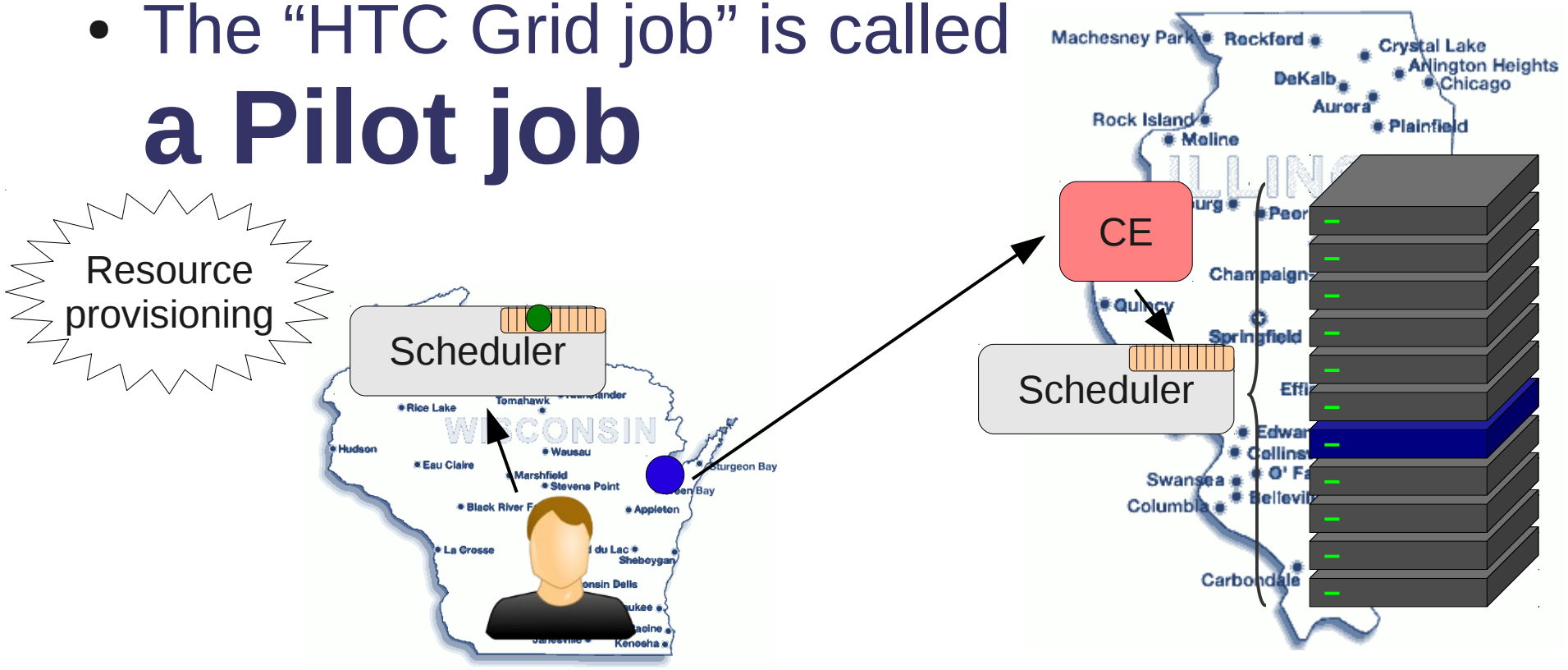
How do we lease in the Grid

- Each Grid job is a lease
- So let's submit **a HTC system** as a Grid job



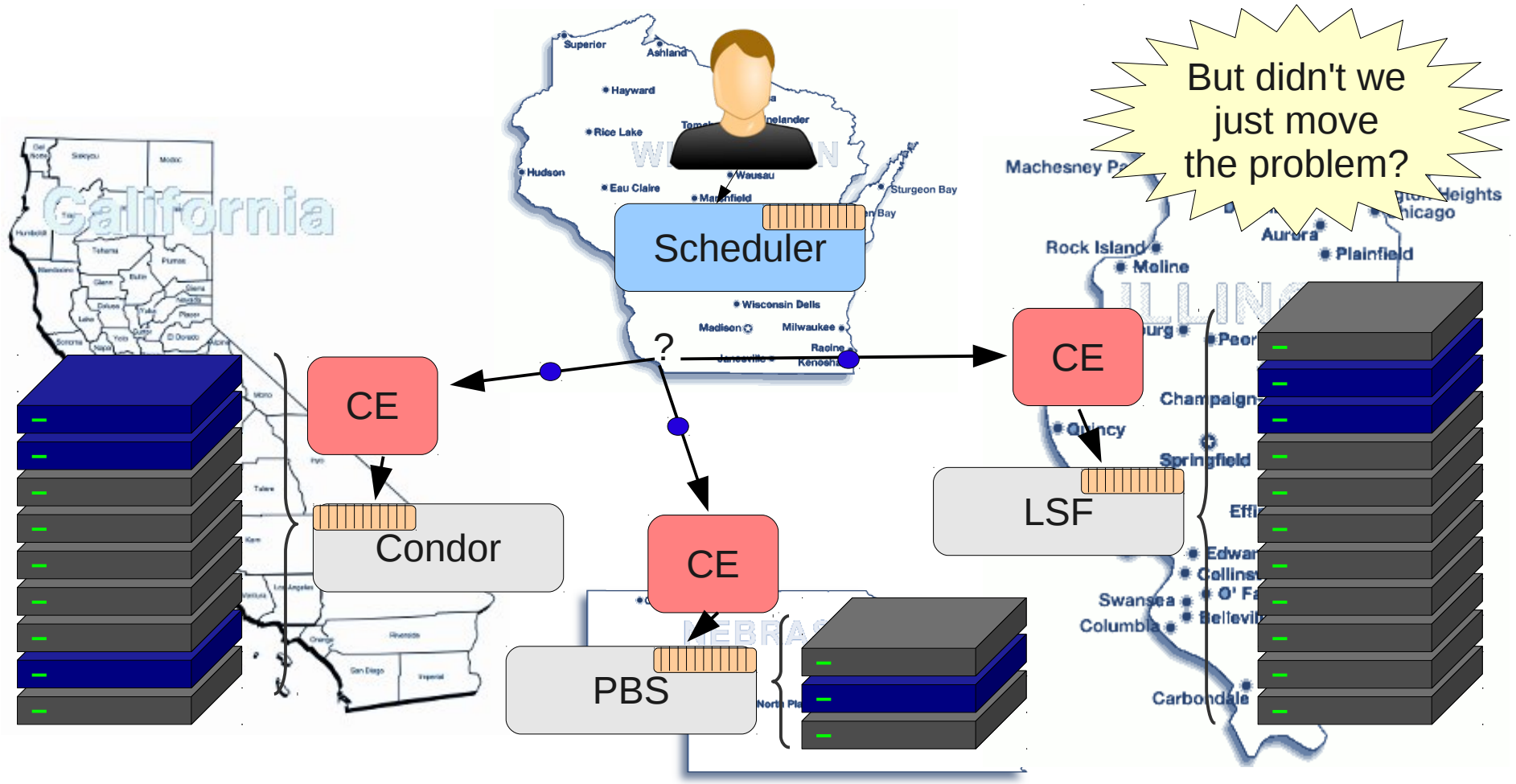
Overlay system

- We effectively create an **overlay system**
  - A HTC system on top of another HTC system
- The “HTC Grid job” is called **a Pilot job**



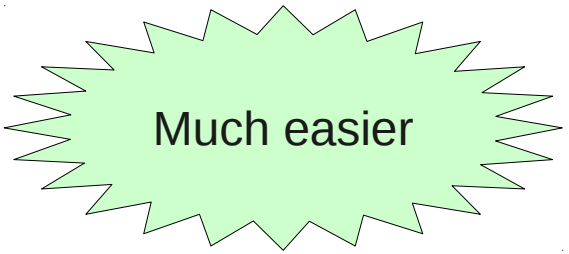
Hiding the D from DHTC

- Just a simple HTC from a user point of view



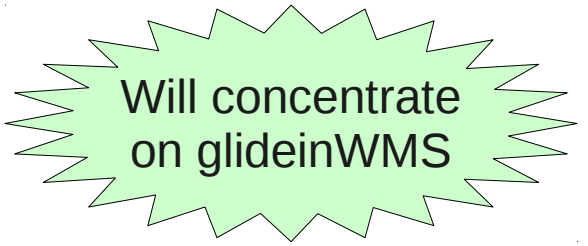
Provisioning not as hard

- Main problem in **user job** partitioning
  - **All jobs are important!**
  - User interested in when the **last job** finishes
- In pilot job “partitioning”
  - **All jobs are the same**
  - User interested in the **total number** of resources provisioned



Pilot systems in real life

- glideinWMS
  - Used by several OSG VOs, including CMS
- PANDA
  - Used mostly by ATLAS
- DIRAC
  - Not used in OSG, used by LHCb



Overlay systems

A high level overview of

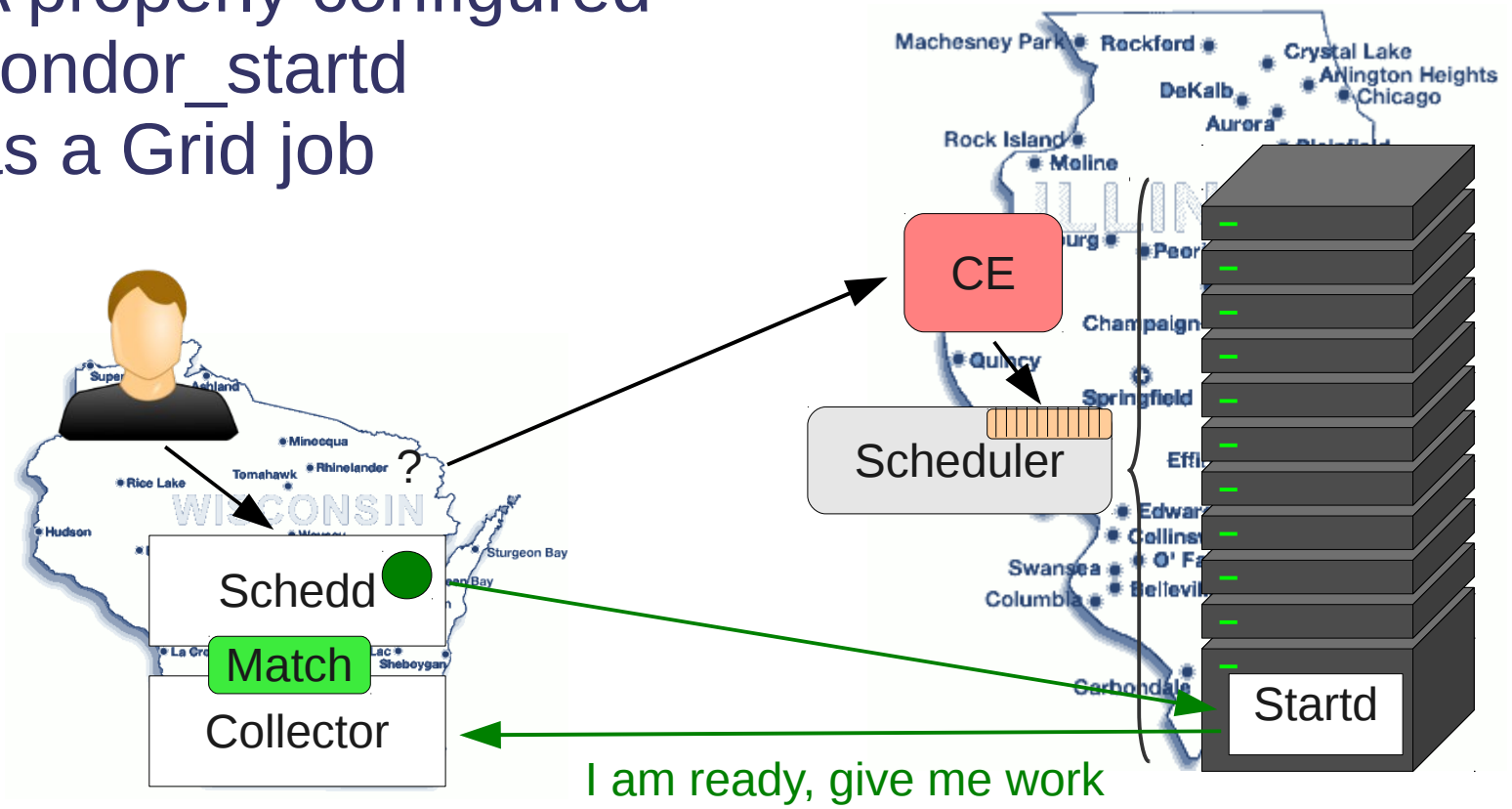
glideinWMS

What is glideinWMS

- A pilot system **based on Condor**
  - Condor as a global HTC system
  - Additional glideinWMS processes used to create and submit the pilot jobs
- Developed by CMS (as a generalization of CDF work)
  - Based on original Condor **glidein** work
- Home page: http://tinyurl.com/glideinWMS

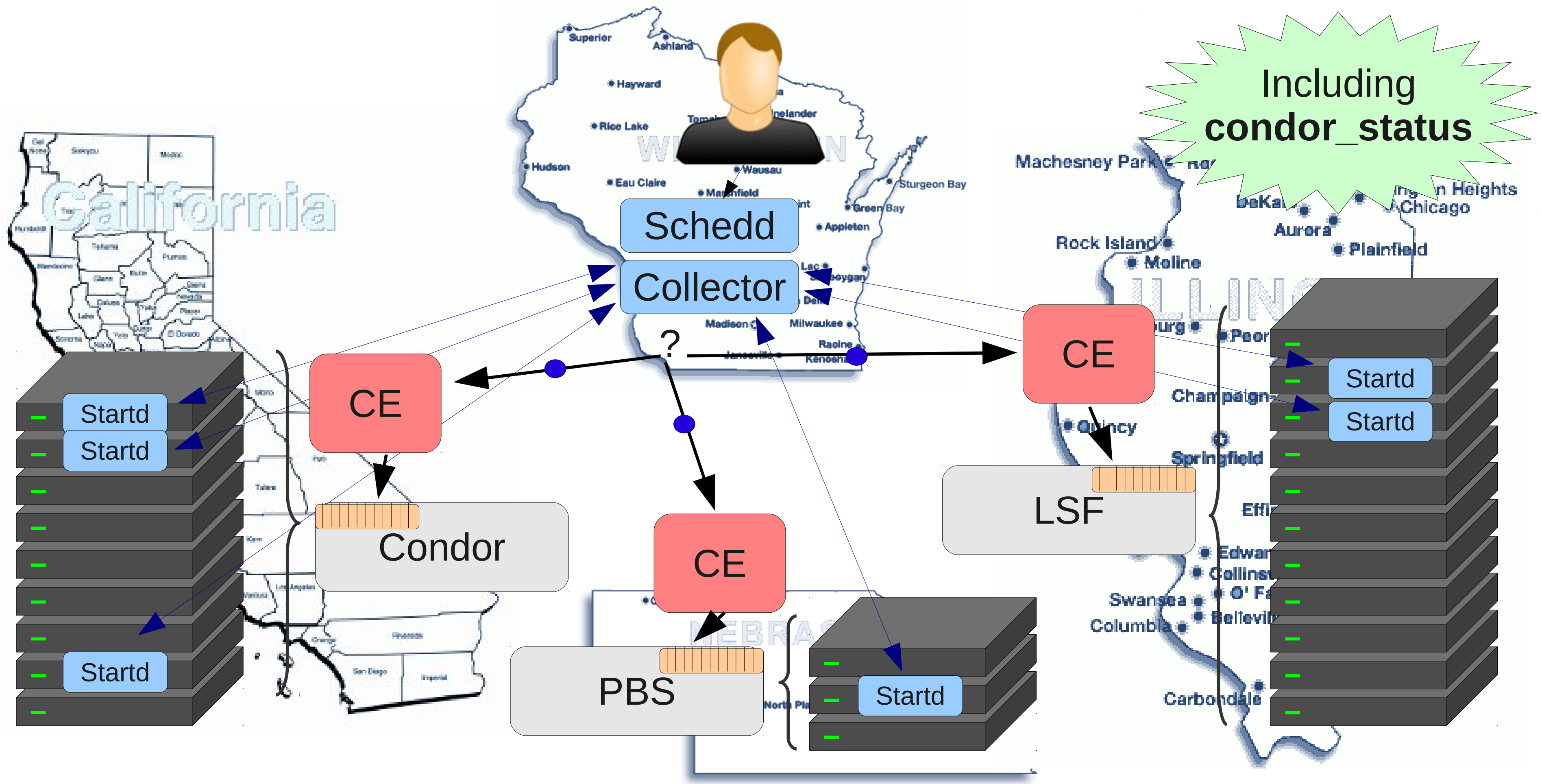
Glidein = Condor pilot

- Glidein
  - A properly configured condor_startd as a Grid job



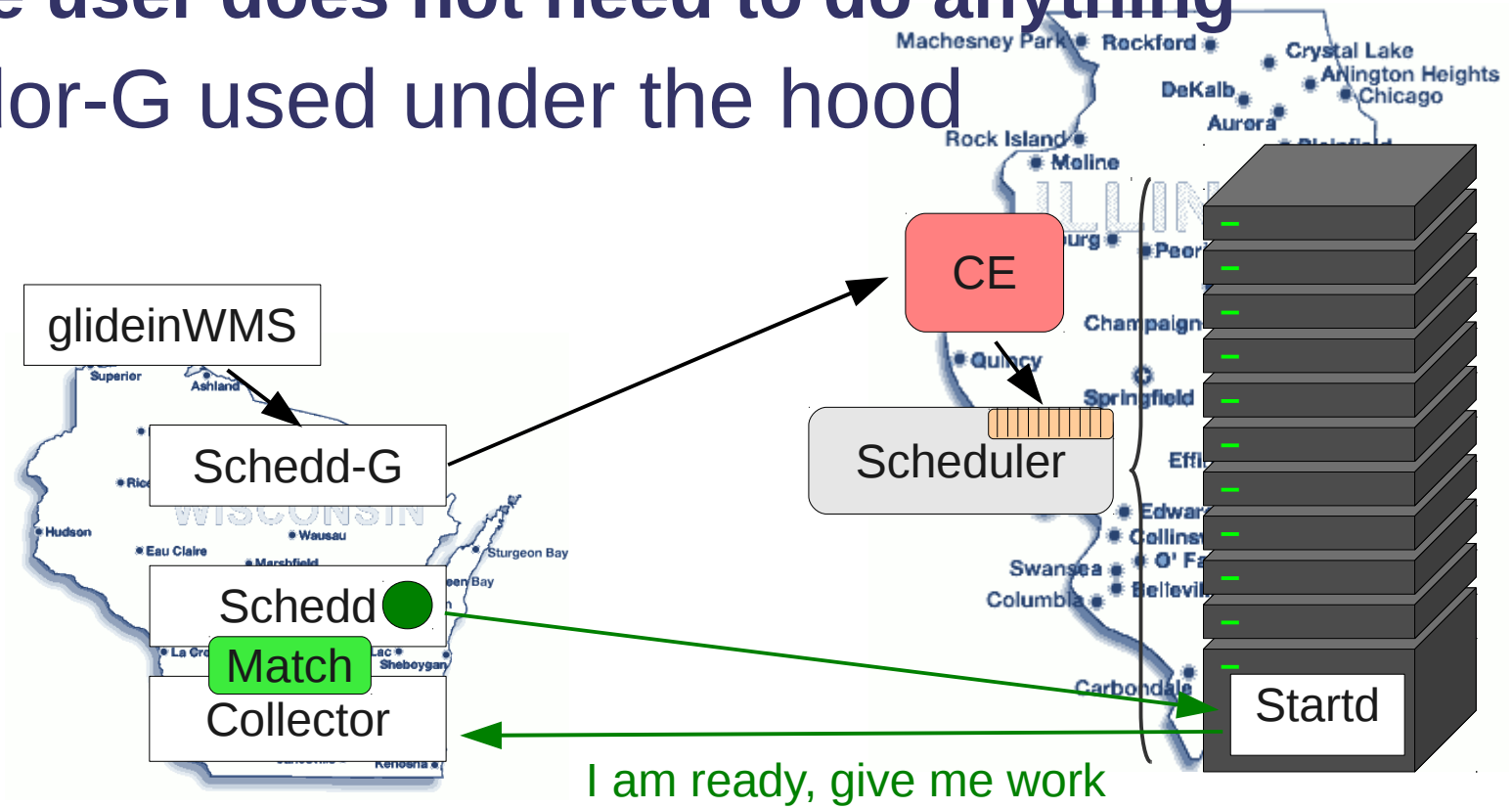
Glidein pool

- Just like a regular Condor pool for the user



glideinWMS

- glideinWMS processes are the ones that actually configure and submit the glideins
  - **The user does not need to do anything**
- Condor-G used under the hood



Resource selection

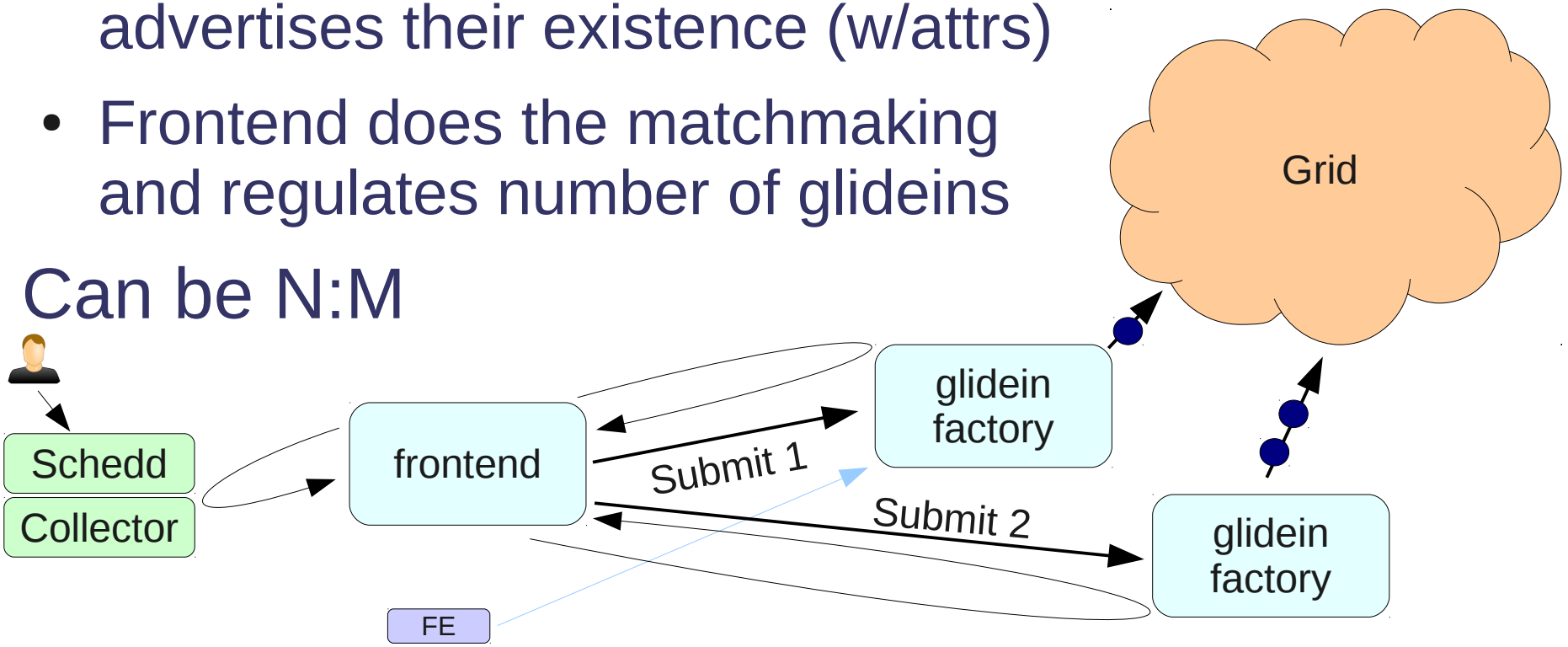
- Users may want to run only on a subset of resources
  - i.e. have some requirements
- You don't want to provision resources that user jobs will not use!
  - glideinWMS thus does matchmaking

glideinWMS matchmaking

- Not as sophisticated as the rest of Condor
- Policy centralized in glideinWMS
  - No “requirements” expression in job ClassAd
- On the plus side, very easy on users
  - Just add an attribute
  - Typical basic setup has
    +DESIRED_Sites=”...”
    (startd requirements contain stringListMember(GLIDEIN_Site,DESIRED_Sites)=?=True)

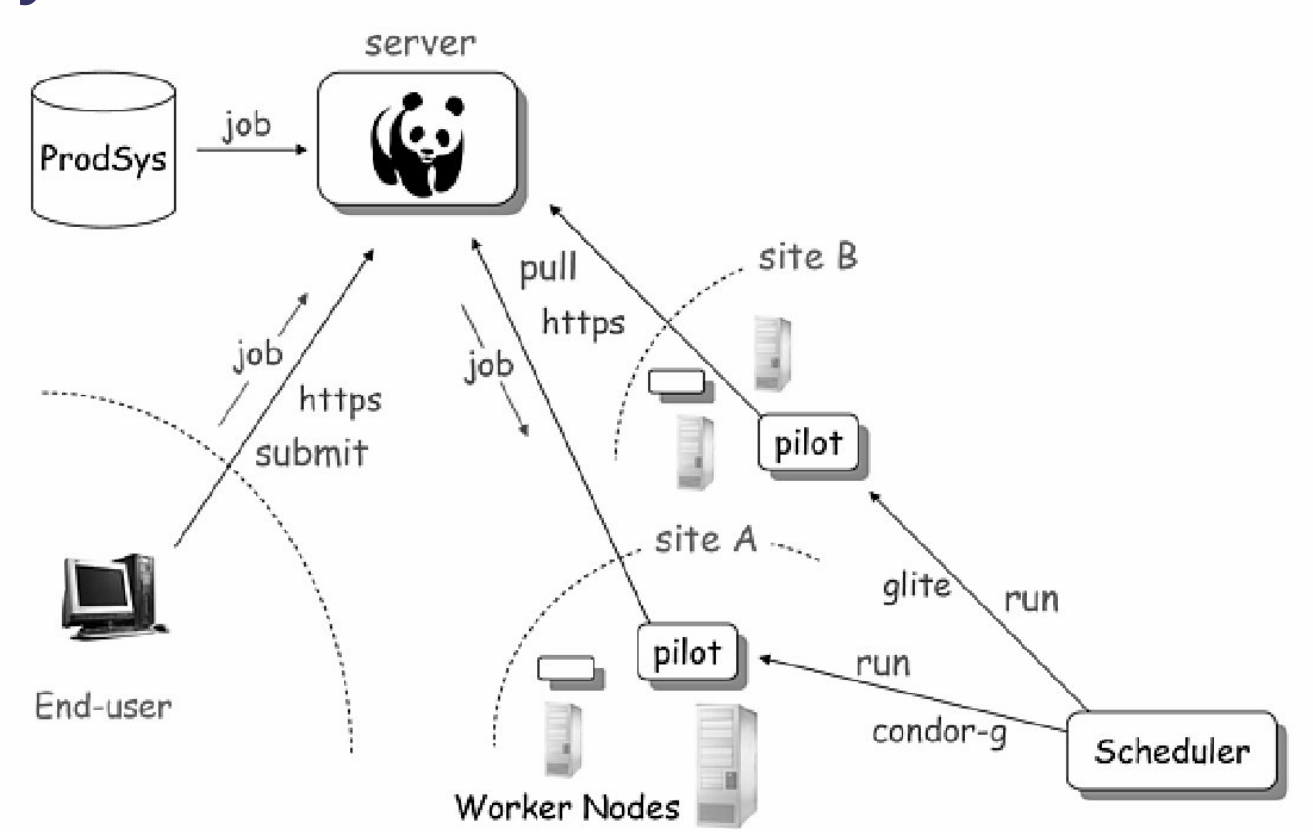
Architecture

- Separates glidein submission from matchmaking
  - Factory knows about sites and advertises their existence (w/attrs)
  - Frontend does the matchmaking and regulates number of glideins
- Can be N:M



PANDA

- High level overview, just for comparison
- Heavily based on Web standards



Get your hands dirty

- This is all the theory you need to know for now
- Exercise time

- Feel free to ask question