Open Science Grid

Introduction to Distributed HTC and overlay systems

Tuesday morning, 9:00am

Igor Sfiligoi <isfiligoi@ucsd.edu>
Leader of the OSG Glidein Factory Operations
University of California San Diego

About Me

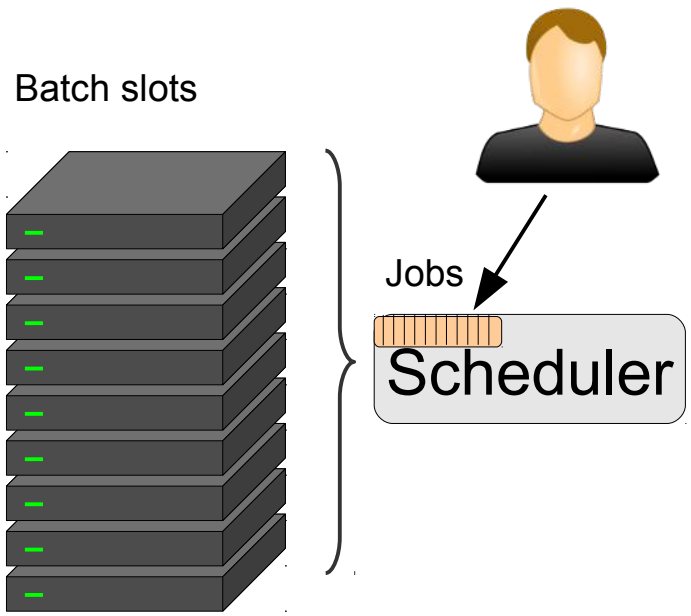
- Working with distributed computing since 1996
- Working with Grids since 2005
- Leader of the OSG glidein factory ops since 2009
- Deeply involved in overlay system development and deployments
- Mostly worked with Physics communities (KLOE, CDF, CMS)

Logistical reminder

- It is OK to ask questions
  - During the lecture
  - During the demos
  - During the exercises
  - During the breaks
- If I don't know the answer,
  I will find someone who likely does

High Throughput Computing

- Alain yesterday introduced you to HTC
  - The concept of getting as many CPU cycles as possible over the long run



- Based on batch job processing
  - No interactive access to resources

HTC in words

- As our esteemed Miron would put it

HTC is about extending the compute power of my own machine.

I **could** run my work on my own machine, but then it would take a very large number of calendar days/months/years to complete.

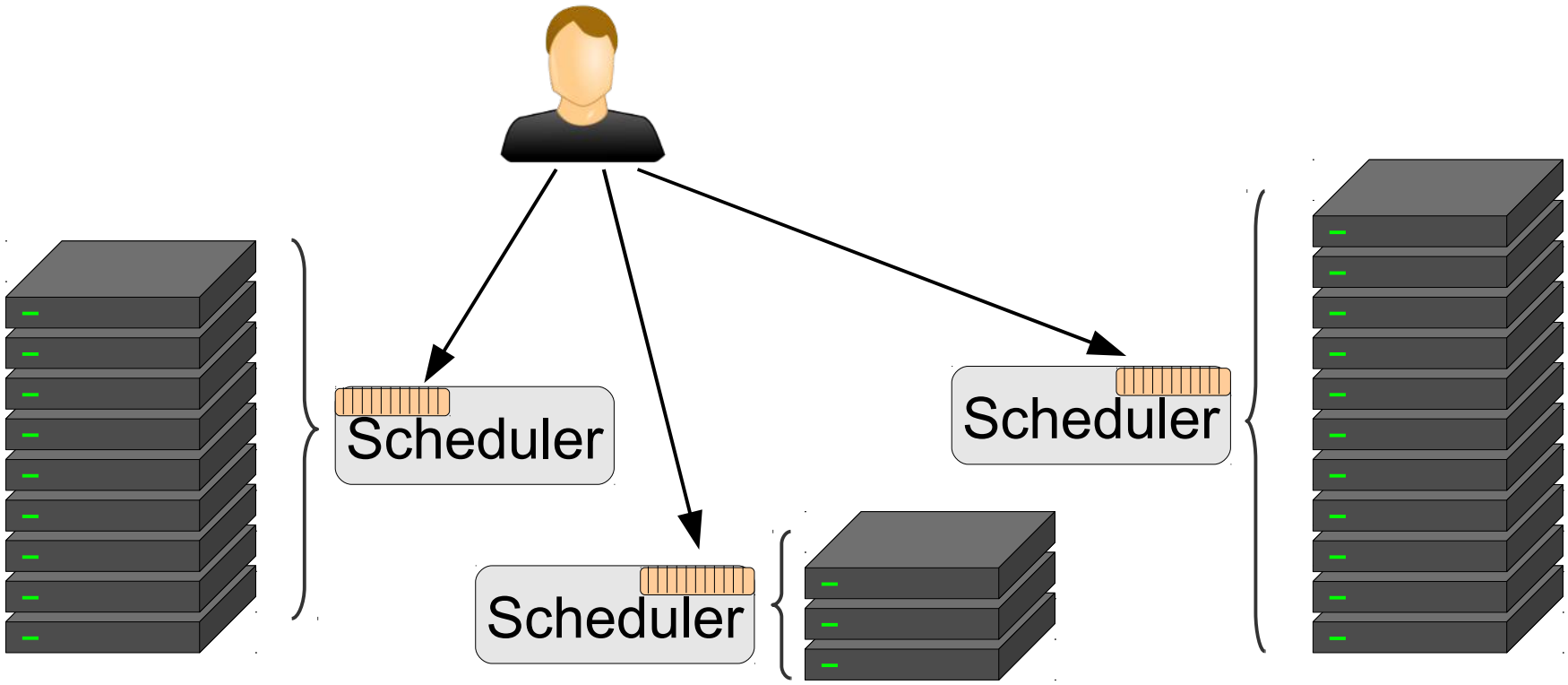
To finish the computation in a reasonable time, I have to expand the capacity of my own machine by obtaining and using temporary resources.

Introducing DHTC

- So what is **Distributed** HTC???
  - HTC is always distributed, right?
- What we mean here is **MASSIVELY distributed**
  - i.e. more than you can afford to host and operate in one place

Anatomy of DHTC

- So DHTC is about computing on **more than one** HTC system



Why DHTC

- Many reasons:
  - Practical
    (a site has a limit to how much HW can host)
  - Political
    (you only get money for HW if it is hosted at X)
  - Economical
    (hosting and operating HW myself is too expensive)
    (someone else can offer you hosted HW for less)
  - Opportunistic
    (owners of site X have temporarily no jobs, might as well allow others to use them (for free or for pay))

Why is DHTC different?

- Not a single system anymore
  - How do I partition my jobs?
- Different clusters likely operated by different people
  - Leads to variations in compute environment
- Likely no globally shared file system
- Typically Wide Area Networking
  - Likely lower bandwidth and high RTT

Why is DHTC different?

- Not a single system anymore
  - How do I partition my jobs?
- Different clusters likely operated by different people
  - Leads to variations in compute environment
- Likely no globally shared file system
- Typically Wide Area
  - Likely lower bandwidth

Will concentrate on this part today.

Why is DHTC different?

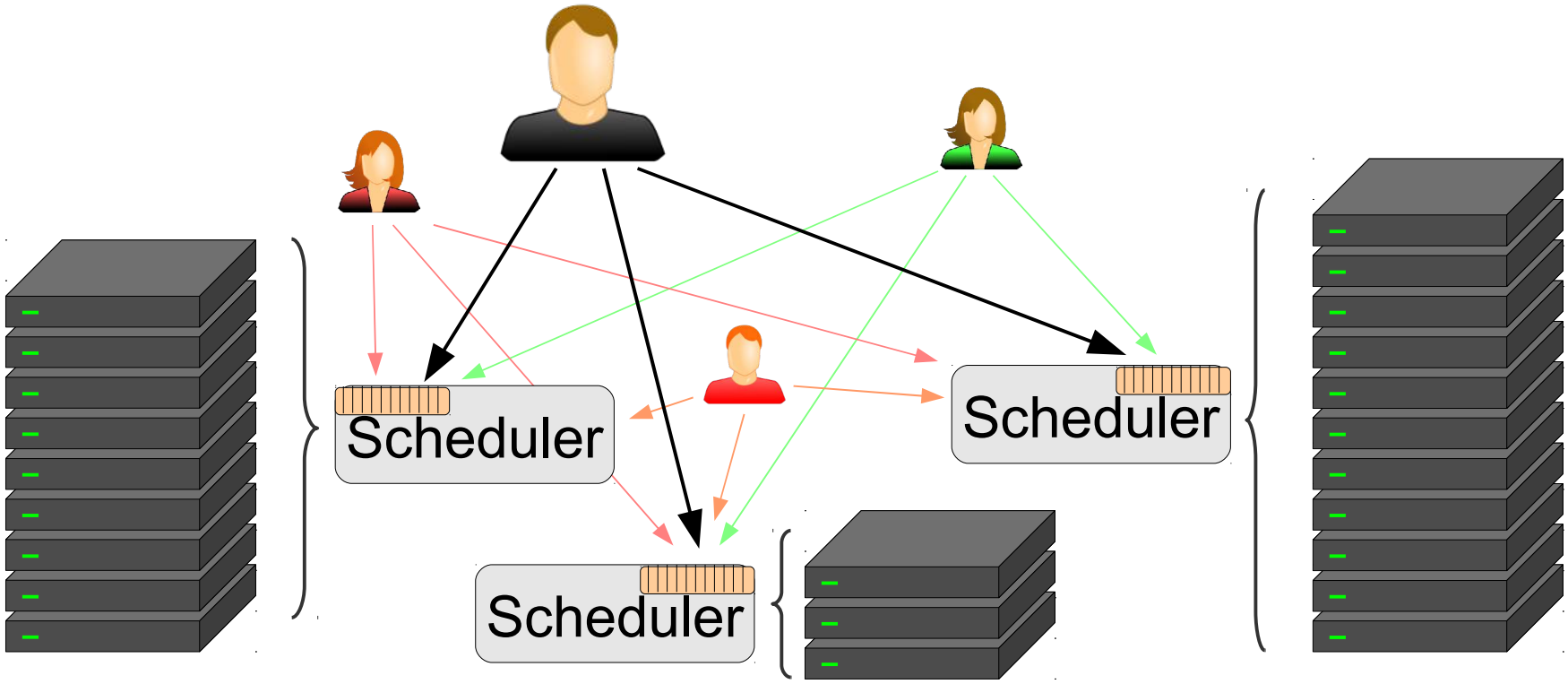
- Not a single system anymore
  - How do I partition r
- Different clusters
  operated by diffe
  - Leads to var tions in compute environment
- Likely no globally shared file system
- Typically Wide Area Networking
  - Likely lower bandwidth and high RTT

Tomorrow's topic.

i.e. you will have to do explicit data movement.

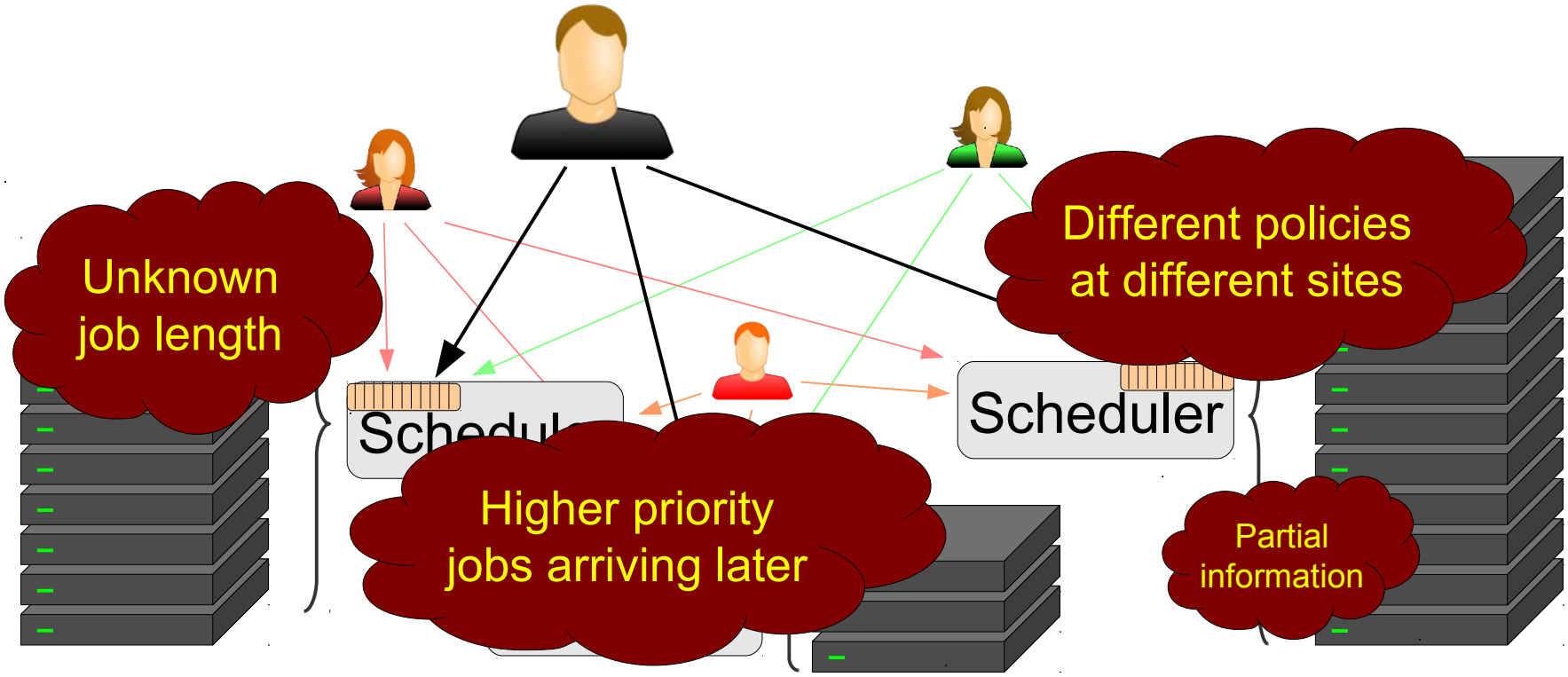
Job partitioning?

- Used naively, DHTC requires job partitioning
  - Very hard to do it right in a multiuser world!



Job partitioning?

- Used naively, DHTC requires job partitioning
  - Very hard to do it right in a multiuser world!

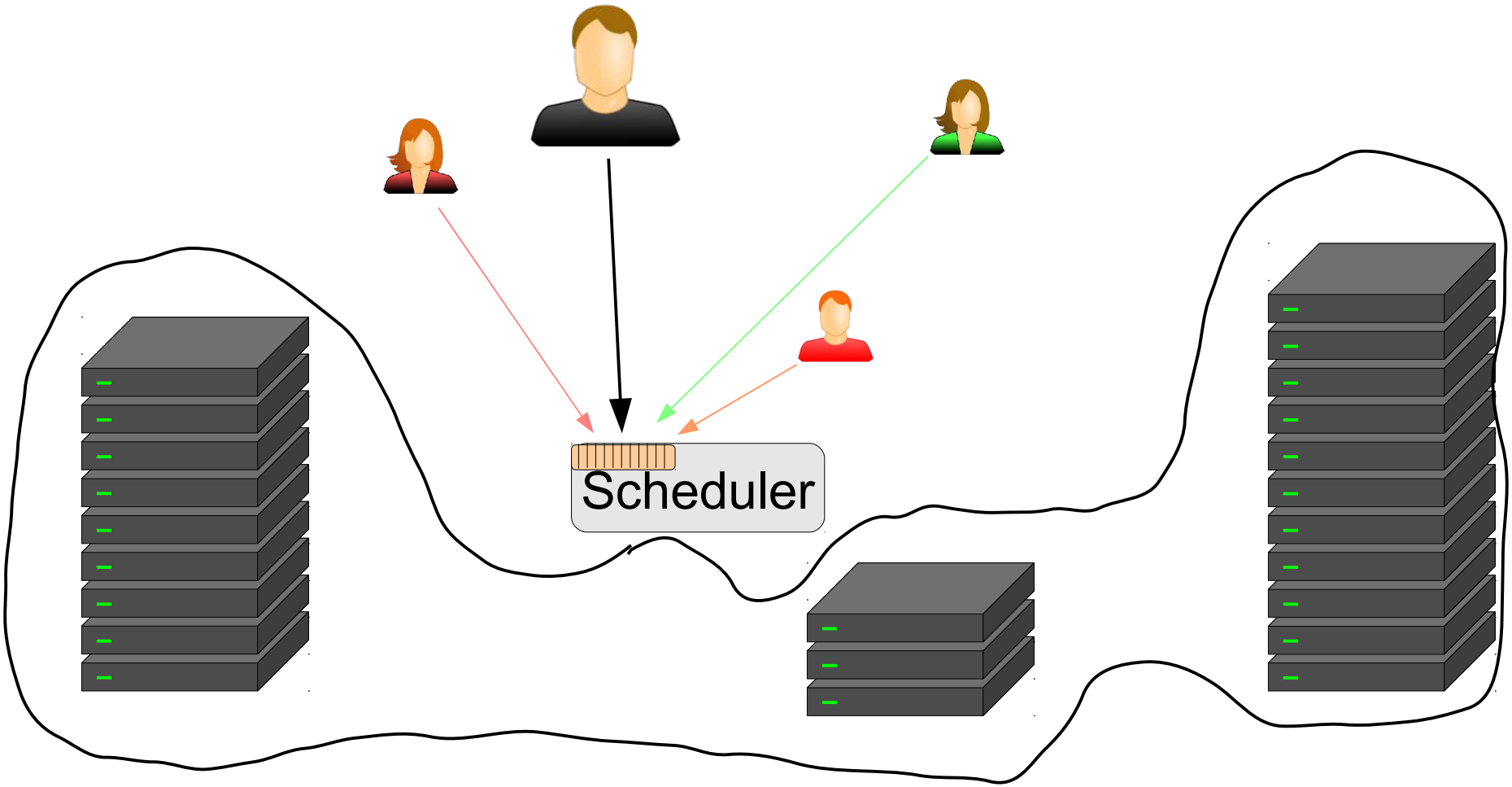


Job partitioning?

- Used naively, DHTC requires job partitioning
  - Very hard to do it right in a multiuser world!

If only we had a global scheduler

- This would make life easy again



If only we had a global scheduler

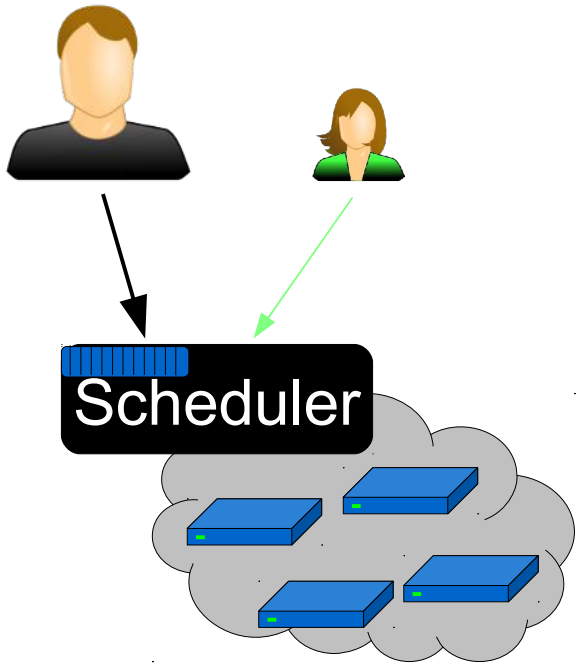
- This would make life easy again

Why we cannot have it?

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

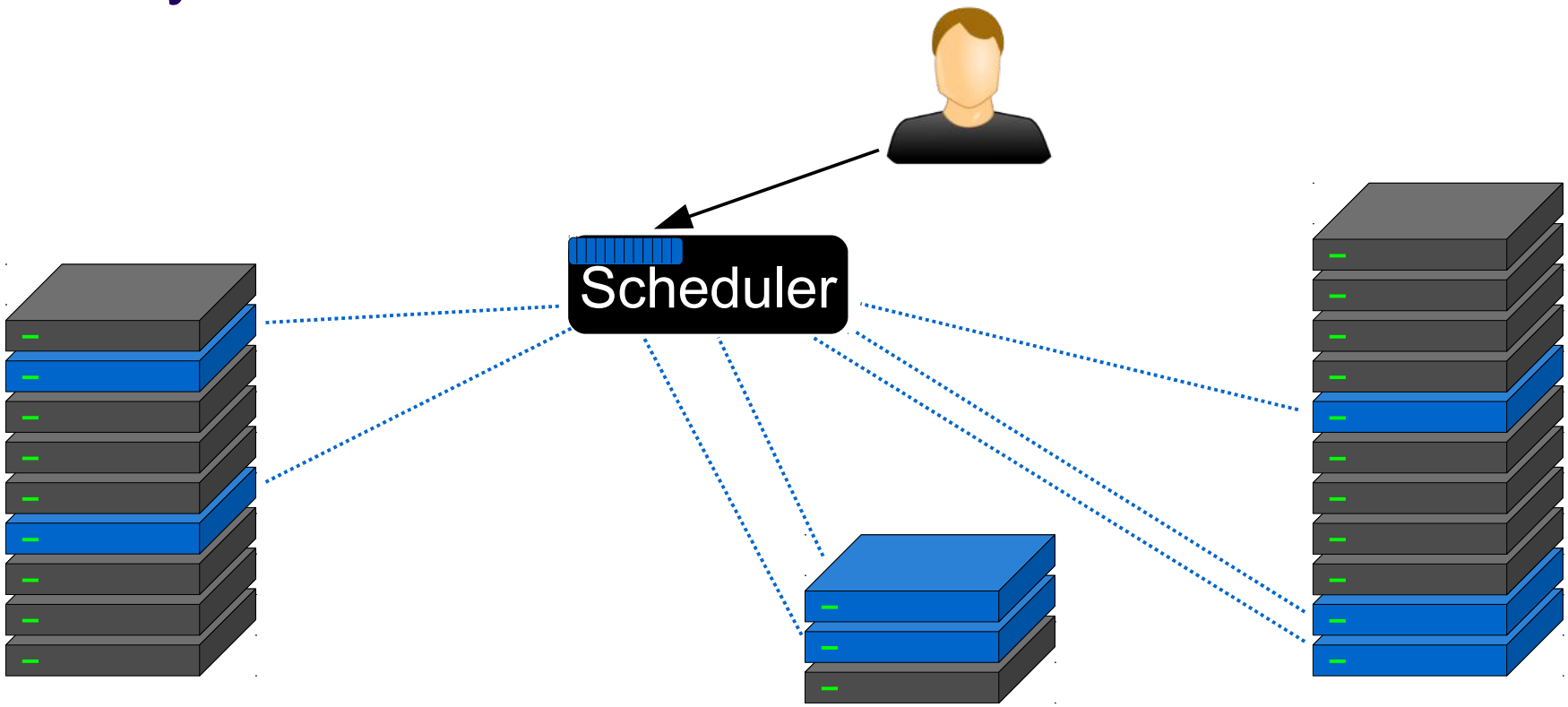
Idea – Create virtual-private HTC

- What if we **collected** a set of batch slots and **only then scheduled** jobs on them?
  - Possibly for a set of users
- From the user point of view, **a single, global** scheduler
  - But the available batch slots change in time



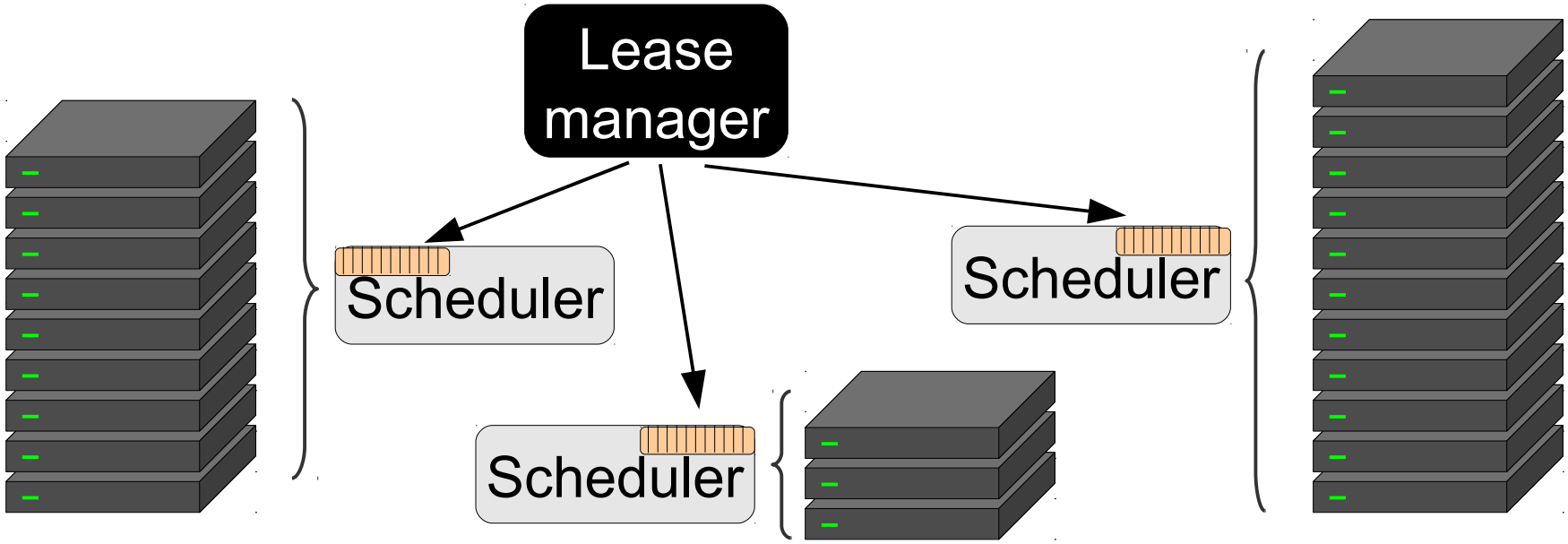
A leasing model

- Imagine leasing some of the batch slots
  - Once you have them,
    you decide what to do with them



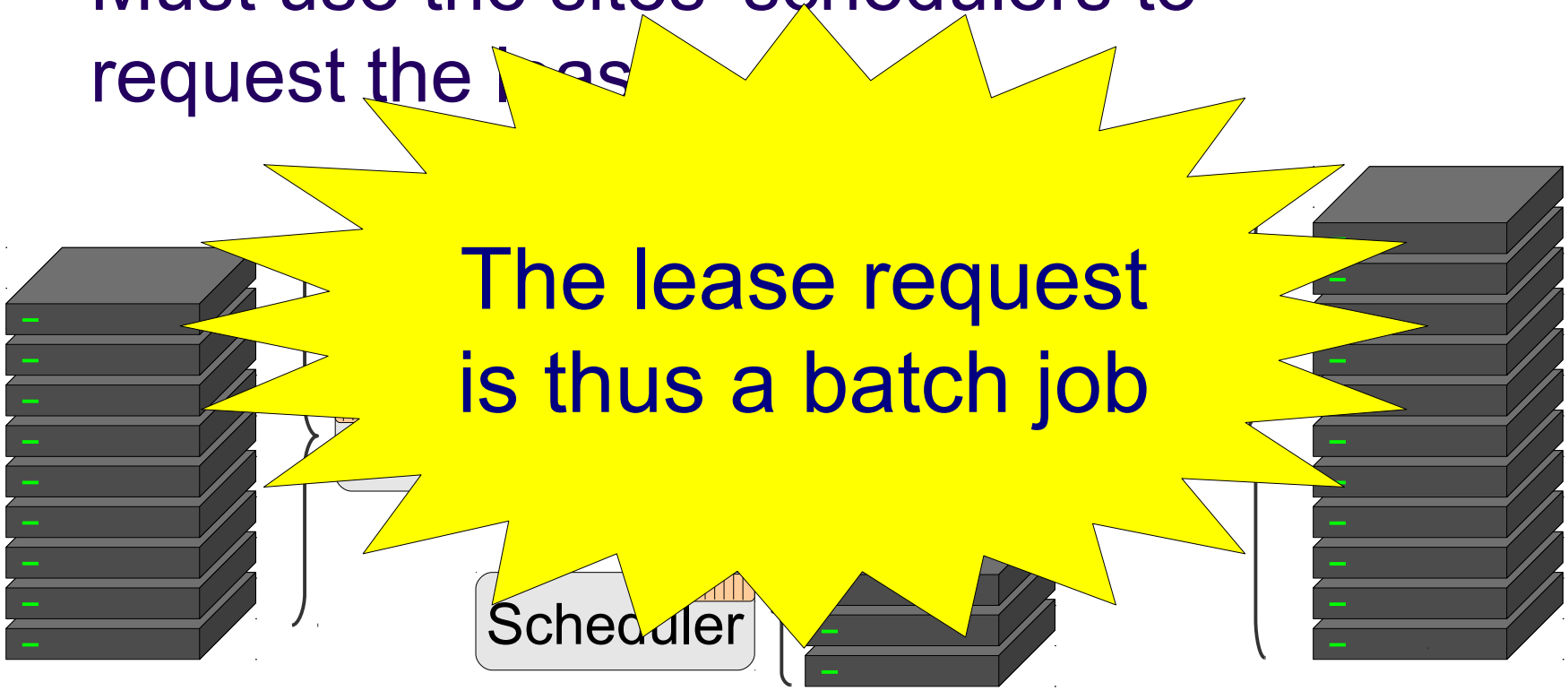
Batch leasing

- Sites still own the resources
  - So we have to play by sites' rules
- Must use the sites' schedulers to request the lease



Batch leasing

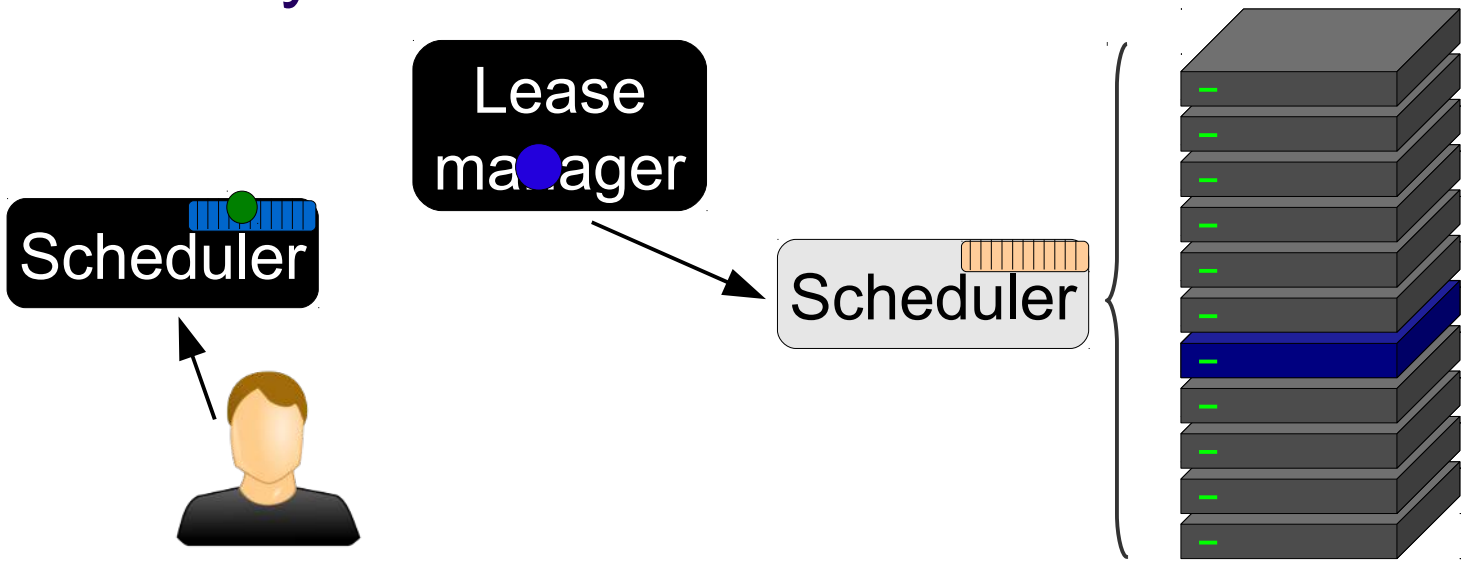
- Sites still own the resources
  - So we have to play by sites' rules
- Must use the sites' schedulers to request the leas



Overlay or Pilot systems

Also known as **resource provisioning**

- We submit **pilot jobs** to sites
- Each pilot job holds the lease on the batch slot
  - Creating an **overlay HTC system**
  - Overlay == 2nd level



Overlay or Pilot systems

Also known as **resource provisioning**

- We submit **pilot jobs** to sites
- Each pilot job holds the lease on the batch slot
  - Creat
  - C

But didn't we just move the problem?

Provisioning not as hard

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

Fighting heterogeneity

- Pilot jobs can tweak the environment before starting user jobs
  - So users see a much more homogeneous system
- Cannot do miracles, of course
  - Usually limited to unprivileged-user tweaks (e.g. cannot replace the kernel)
  - But this is enough most of the time

Fighting heterogeneity

- Pilot jobs can tweak the environment before s
  - So
- O
  - B

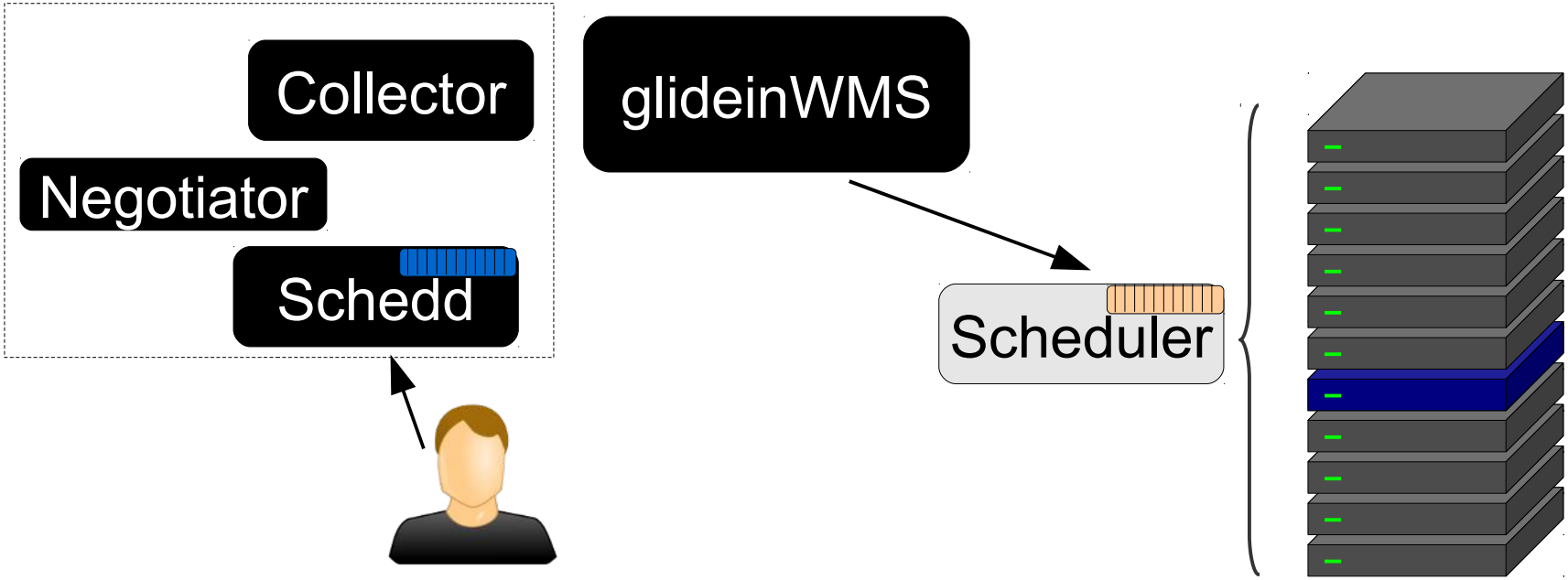
Only a few pilot system administrators need to worry about heterogeneity

Pilot systems

- Many possible implementations
- We will concentrate on **glideinWMS**
  - Based on Condor
  - The one used by most user communities on OSG
- Others available
  - PANDA, DIRAC, ALIEN, ...

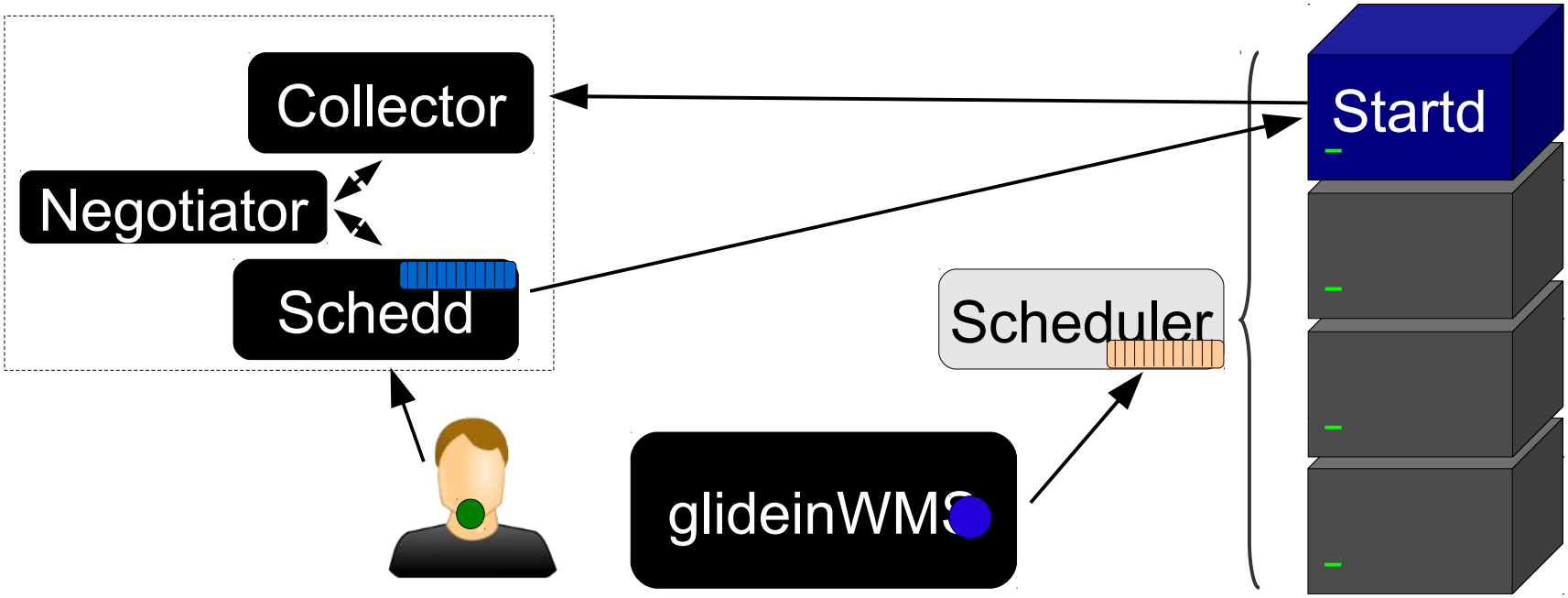
glideinWMS

- A Condor based overlay system
  - i.e. looks like a regular Condor system to the users
  - Adds a resource provisioning service
    (i.e. the lease manager)



Condor pilots

- Condor pilot == A glidein
- A properly configured Condor Startd



Two level matchmaking

- The system now has **two matchmaking points**
  - The **glideinWMS** decides when and where to send glideins
  - The **Condor negotiator** decides which job runs on which glidein
- The two must treat jobs the same way
  - Or we end up with either unused glideins or jobs that never start

Moving policy in glideinWMS

- In glideinWMS, **user jobs never have requirements**
- All policy is implemented by system administrators
  - Users **just provide parameters**

Example policy

```
(DESIRED_SITES=?=”any”) || stringListMember(GLIDEIN_Site,DESIRED_SITES)
```

Example job JDL

```
+DESIRED_SITES = “UCSD,Wisconsin”
queue
```

Example job JDL

```
+DESIRED_SITES = “any”
queue
```

Know your system

- The matchmaking is thus less flexible
  - You can only work within the frame of the system policy
- But arguably easier to use
  - No complex boolean expressions to write
- Be sure to ask for the system policy of your system

Down to practice

- This is all for the theoretical part
- Next we have the hands-on session

Questions?

- Questions? Comments?
  - Feel free to ask me questions later:
    Igor Sfiligoi <isfiligoi@ucsd.edu>
- Upcoming sessions
  - Now - 11:00am
    - Hands-on exercises
  - 11:00am – 11:15am
    - Break
  - 11:15am –
    - Next lecture – The Grid and glideinWMS architecture

Beware the power

Courtesy of fanpop.com