2011 OSG Summer School

An introduction to

Distributed High-Throughput Computing

with emphasis on

Grid computing

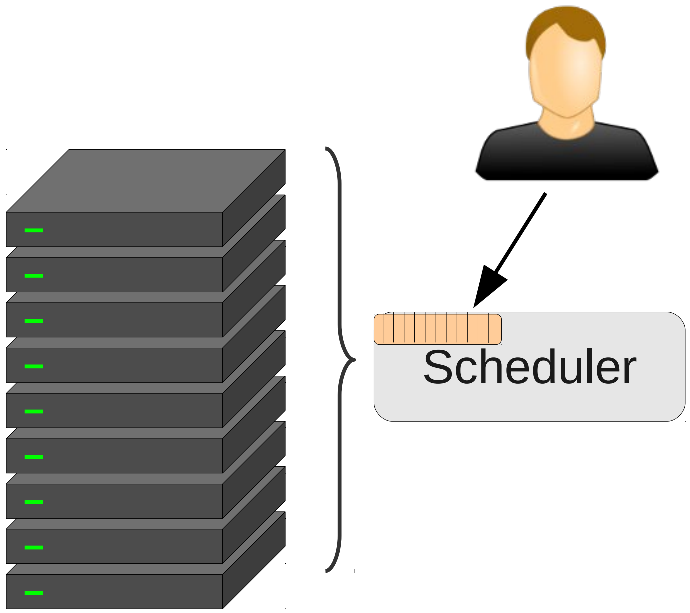
by Igor Sfiligoi
University of California San Diego

Me and DHTC

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

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.

DHTC

Introduction to

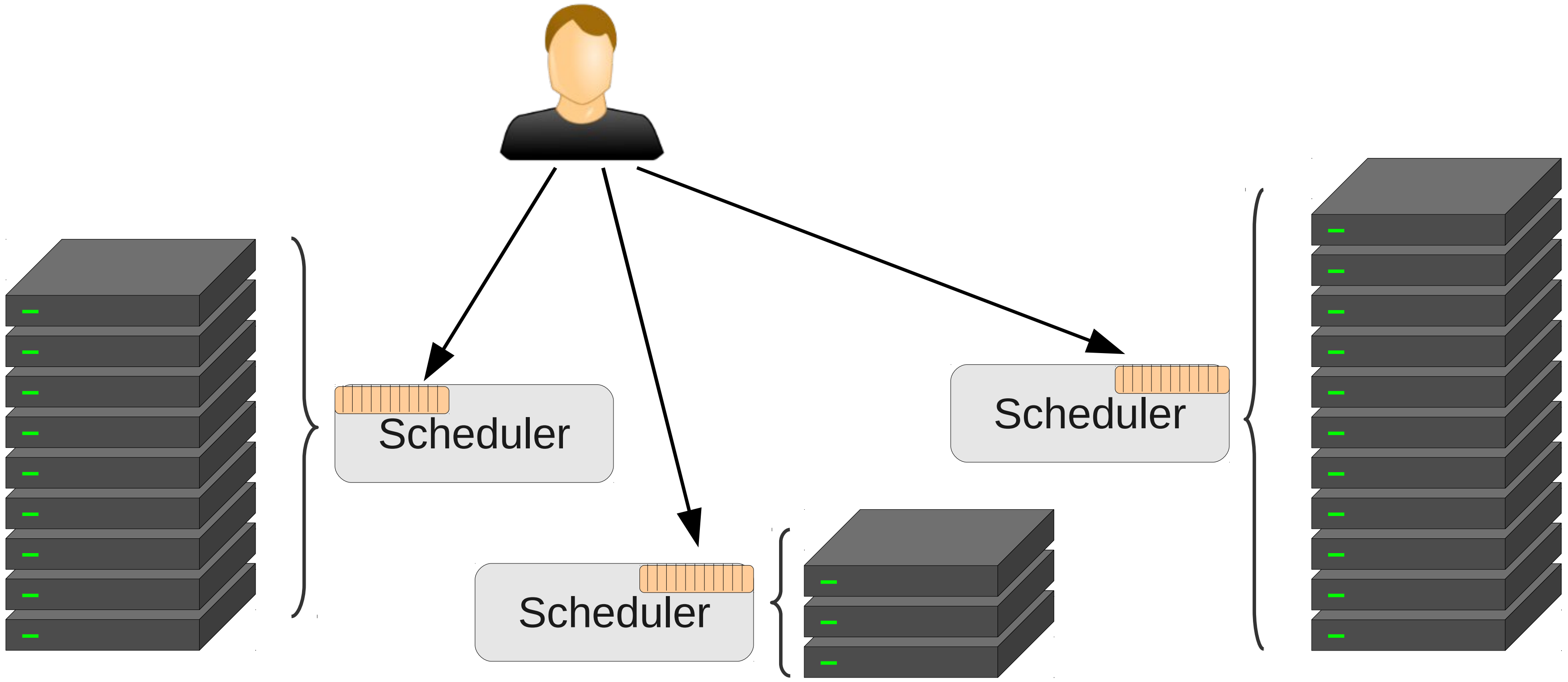
Distributed High Throughput Computing

What is 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 is it different?

- Not a single system anymore
  - Most likely does not have a shared file system
  - Different clusters likely operated by different people
    - You may have a different account
  - Different clusters may use different technologies (e.g. Condor vs PBS)
- Usually harder to use than regular HTC
  - Just as using a HTC cluster is harder than using a single desktop

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))

DHTC in real life

- Campus-wide scheduling
  - e.g. U.Wisc Condor flocking
- Scientific Grids
  - e.g. OSG, Teragrid, EGI
- Hosted servers
  - e.g. Rackspace
- Cloud computing
  - e.g. Amazon EC2

Campus-wide scheduling

- Connects HTC clusters ran by different departments
- High trust between them
  - Sysadmins can talk to each other
  - Trust may be imposed from above (e.g. CIO)
- May use the same technology, which makes life much easier
  - Condor flocking
    an excellent example

Won't go in more detail.

Scientific Grids

- Widely distributed
  - OSG is US wide
- Moderate trust
  - Too many participants
  - OSG – O(100) sites and O(1k) users
- Many technologies
  - Joining sites may have existing infrastructure

Will spend most time on this

Hosted computing

- Commercial offering
- Lease a server for $$/month
- Similar to buying and hosting your own HW
  - Install whatever you want on them
  - But not in your LAN
- May be a good solution if you need a boost in capacity for a few months

Won't go in more detail.

Cloud computing

- A mix between hosted computing and Grids
- Job-based like a Grid
  - But “jobs” are Virtual Machines, not just processes
- You get your own machines like in hosted cmp.
  - They just happen to be VMs
  - You install whatever you want in them
  - There is an economic factor
    (although there is a push for scientific clouds
    as well)

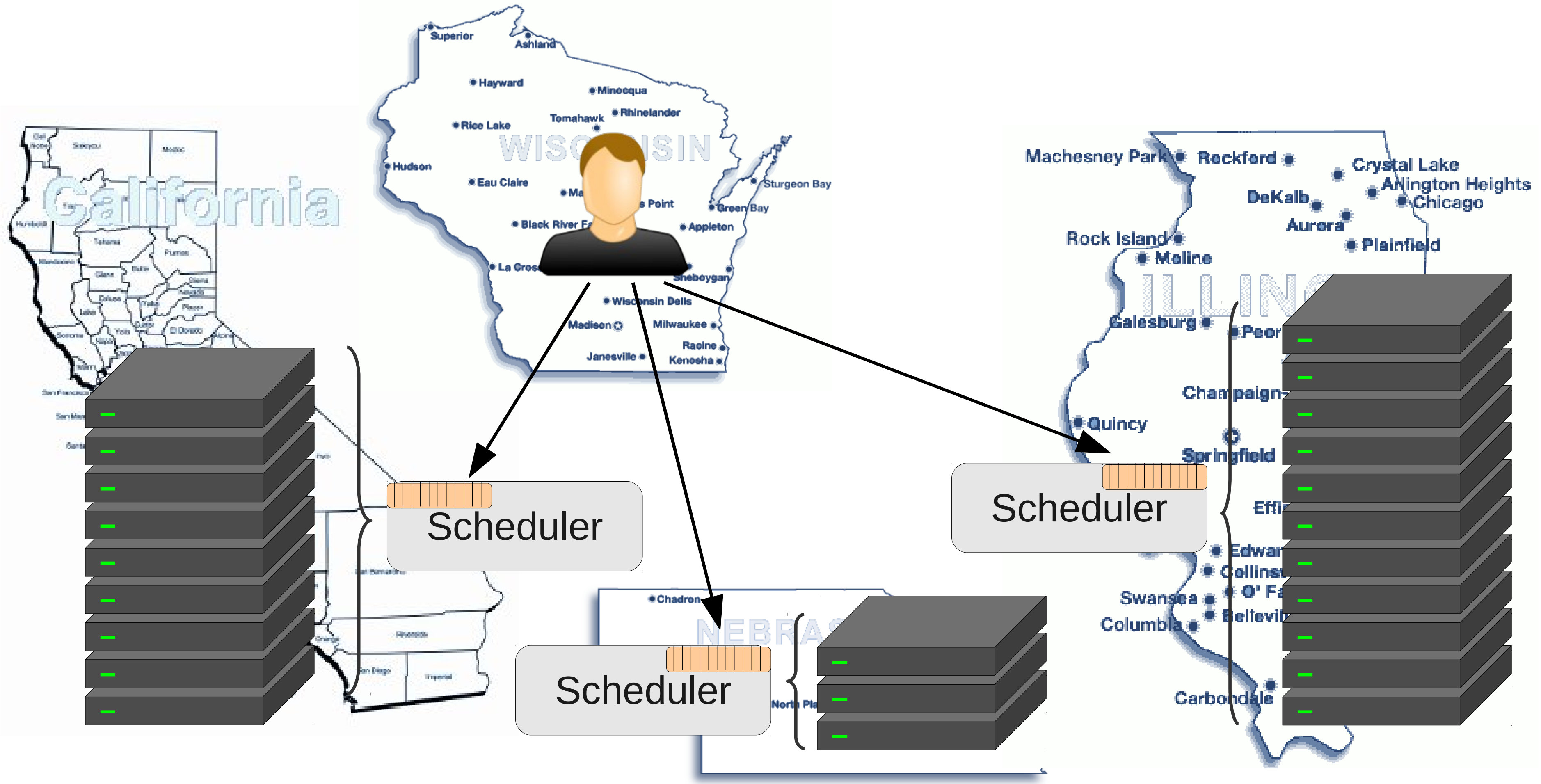
Won't go in more detail.

DHTC

Grid computing

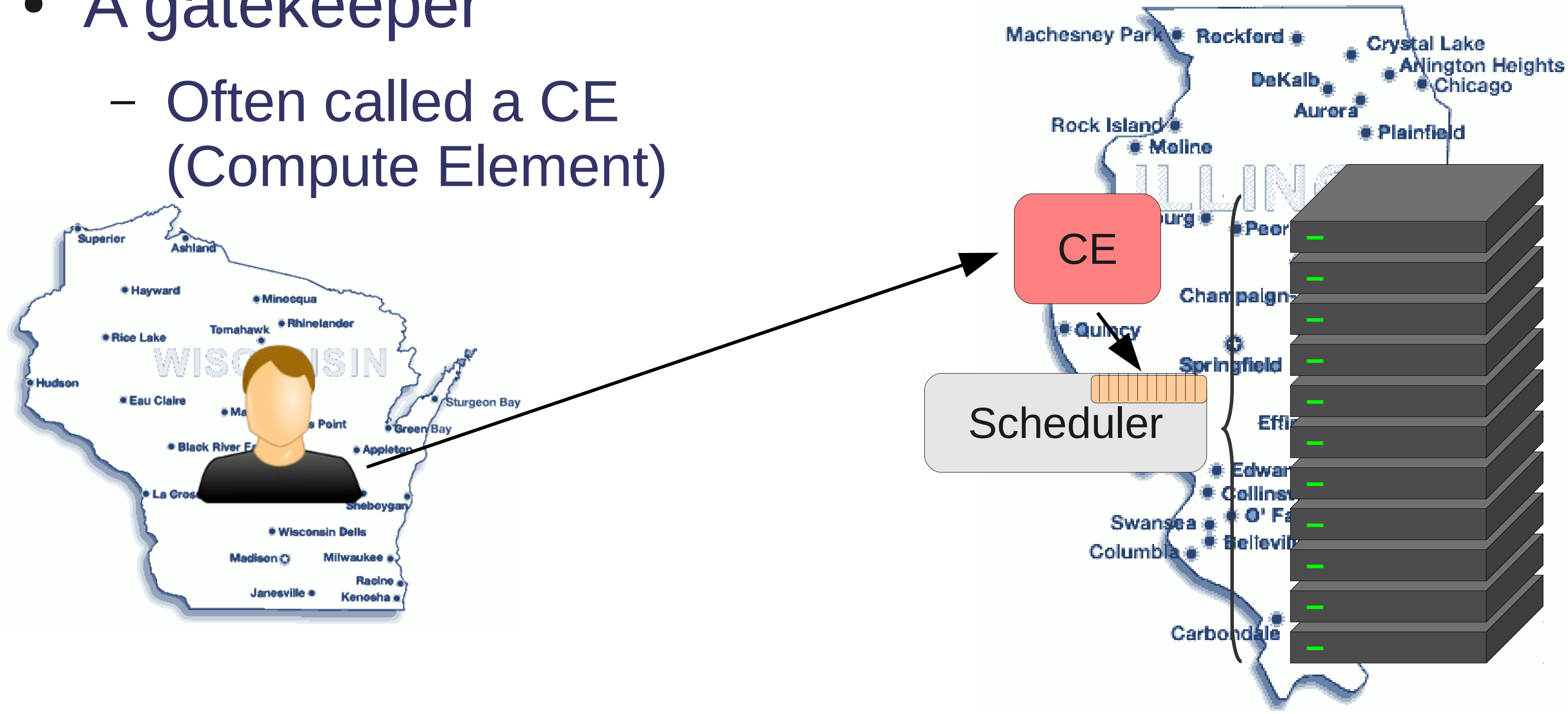
Grid computing

- Think of it as DHTC over WAN



Remote access

- Major difference compared to HTC is that we need remote access
  - A gatekeeper
    - Often called a CE (Compute Element)



Many options

- Batch system native
  - e.g. Condor-C
- SSH
  - Then locally condor_submit, bsub, ...
- Grid gatekeepers
  - Globus
  - CREAM
  - ARC

Will hide site details

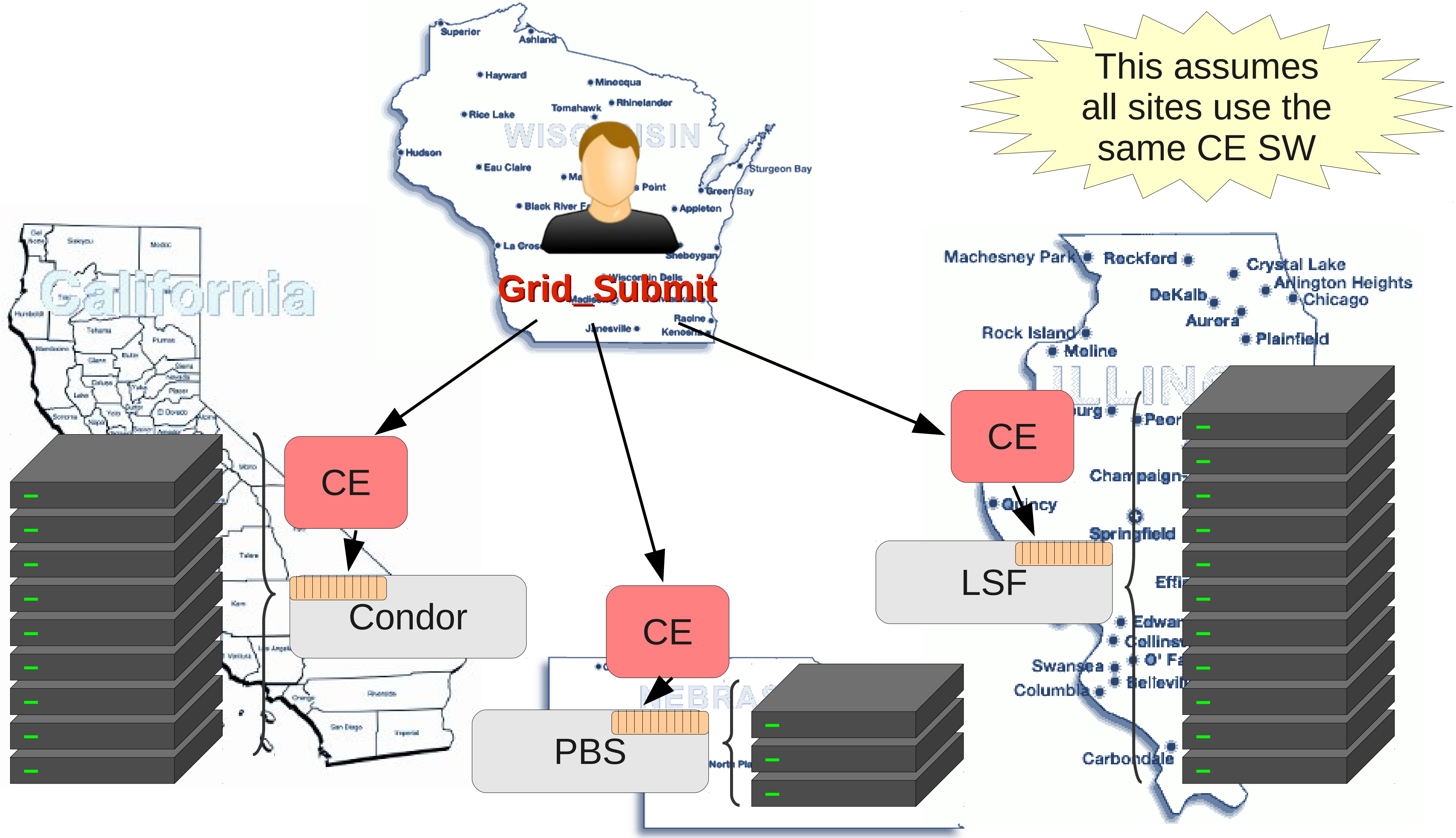
Grid gatekeepers

- Each site potentially uses a different local HTC system
- A Grid gatekeeper abstracts the API
  - Same remote calls for site-local Condor, PBS, ...
- Makes life easier for users
  - No need to know site details

Just learn "Grid" commands

- Used by most major Grid communities
  - e.g. OSG, Teragrid, EGI

Grid CE = Abstraction layer



This assumes all sites use the same CE SW

Condor as a Grid client

- Condor can be used as a “universal client”
- Can submit jobs to remote HTC systems
  - Using the “Grid” universe
  - Supports most CE APIs

Known as
Condor-G

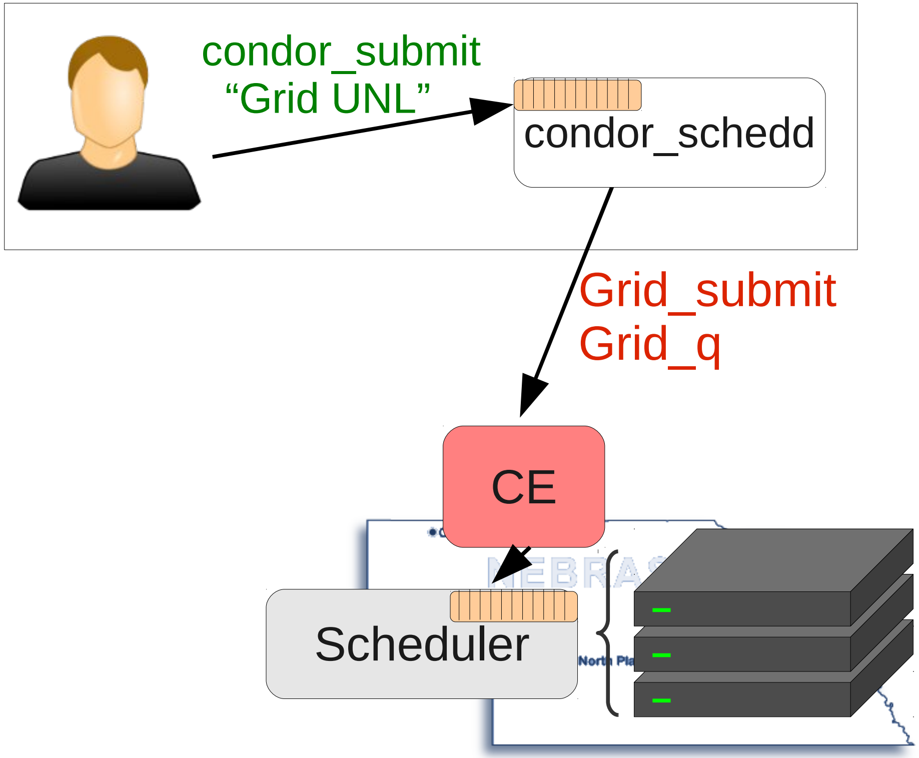
- Same job commands as for Condor HTC
  - condor_submit, condor_q, etc.
- But very different under the hood!

See next slide

Condor-G

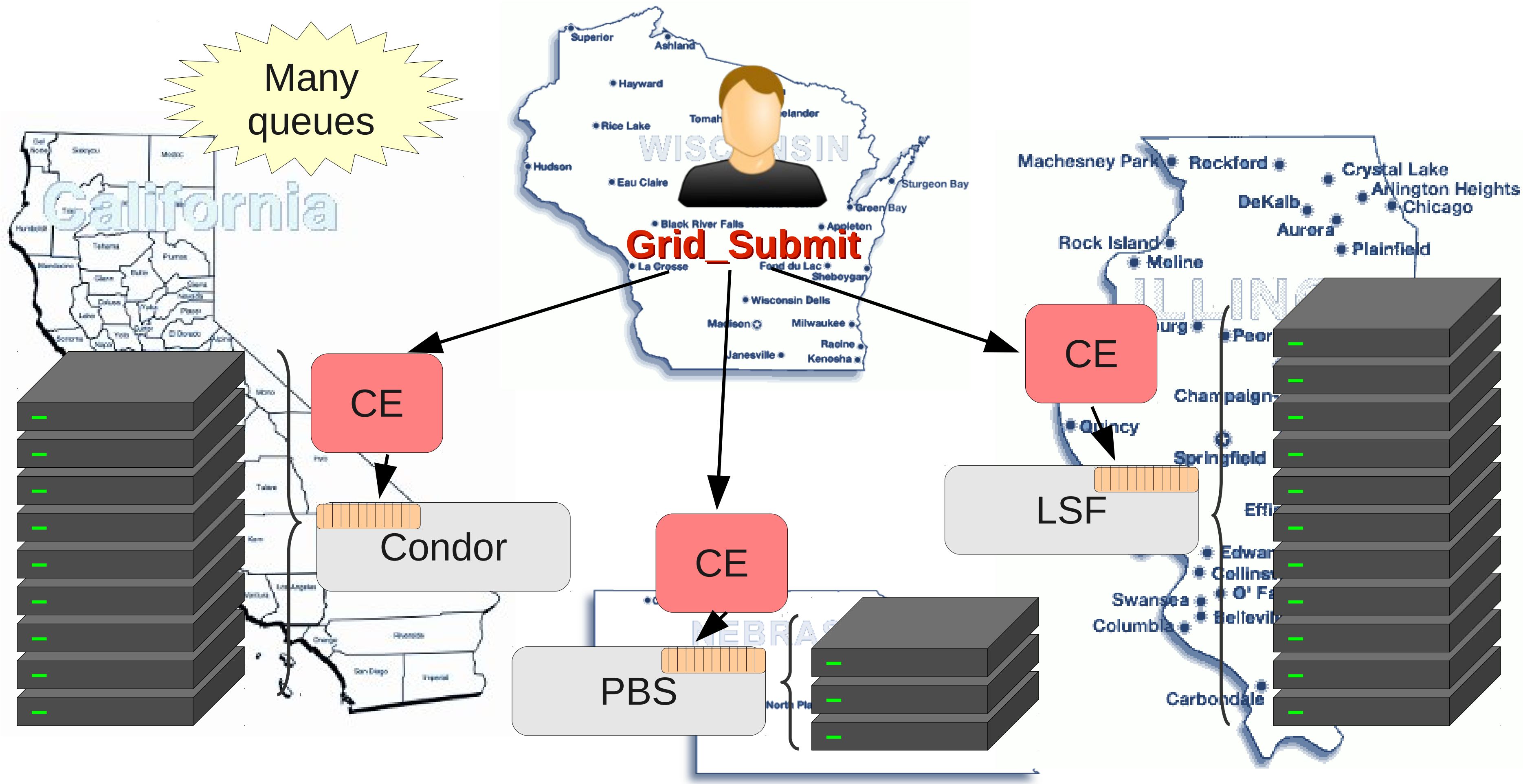
- Condor-G does not manage remote resources
  - It just forwards and monitors jobs to remote HTC sys
  - No condor_status
- No matchmaking
  - User explicitly specifies API and site to use

Limited resource selection available with external support



Job partitioning problem

- Pure Grid computing requires job partitioning



Job partitioning = Hard problem

- Job partitioning is a hard problem
- Especially in the Grid
  - Many different technologies (e.g. Condor vs PBS)
  - Owned by many different admins
  - With an abstracted API in between
- Some automation highly desirable
  - gliteWMS
  - OSG MM

No really good solutions

Get your hands dirty

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

- Feel free to ask question