The Campus Client

Connecting campus researchers to the OSG

David Champion • University of Chicago

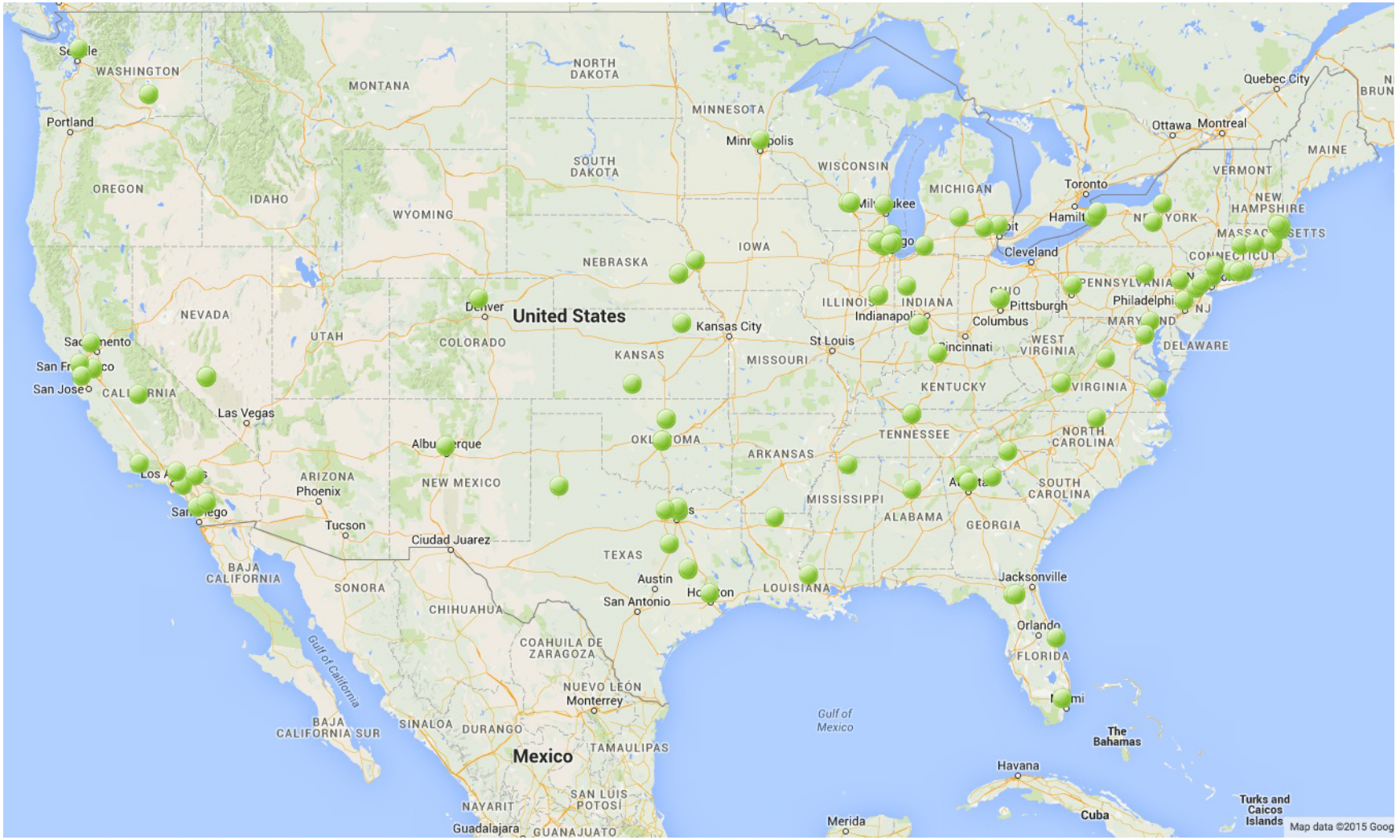
Open Science Grid All Hands Meeting
Northwestern University
March 24, 2015

THE UNIVERSITY OF CHICAGO

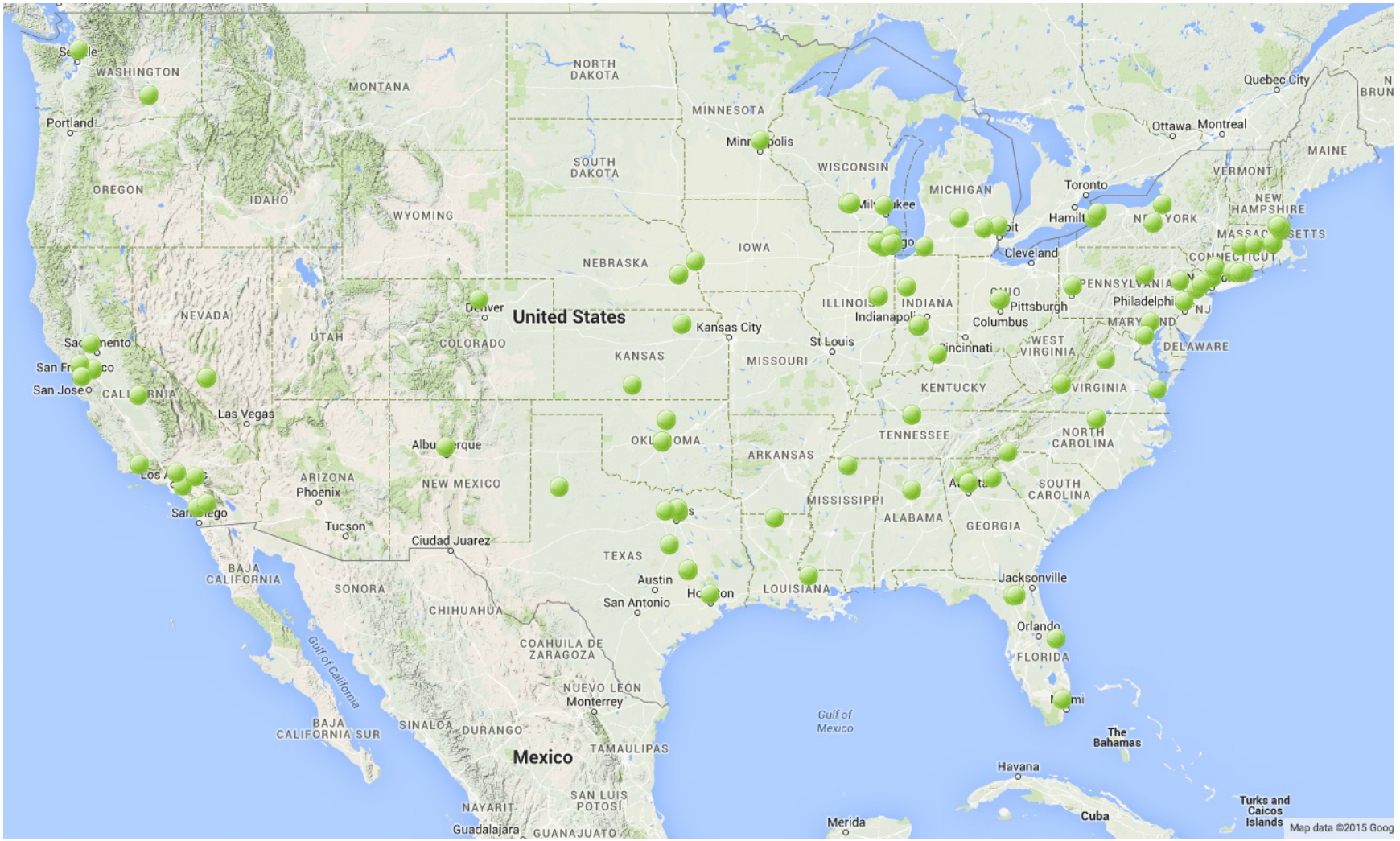
Open Science Grid

- A distributed computing partnership for data-intensive research
- 140+ resource providers in the Americas

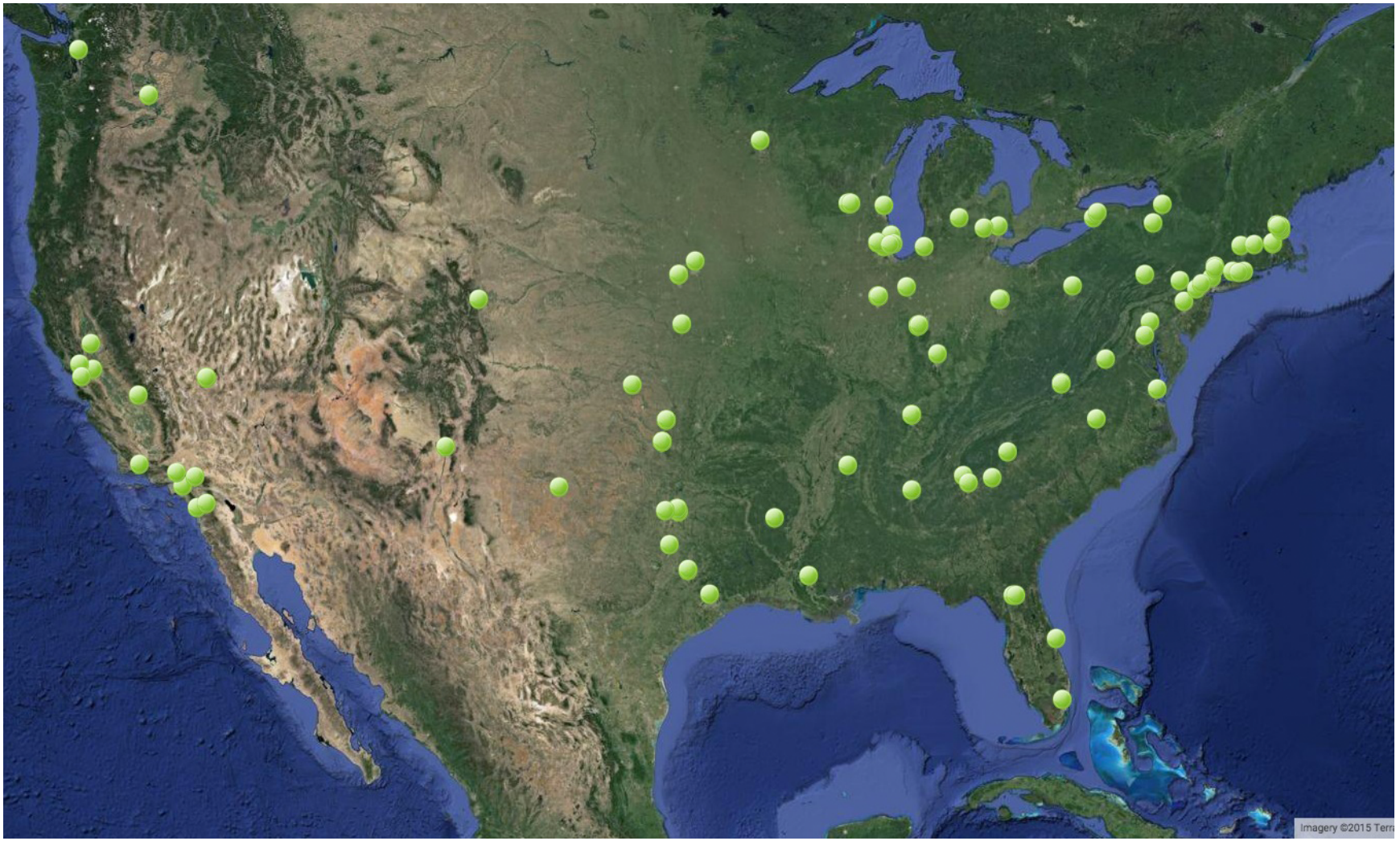
OSG: 140 sites



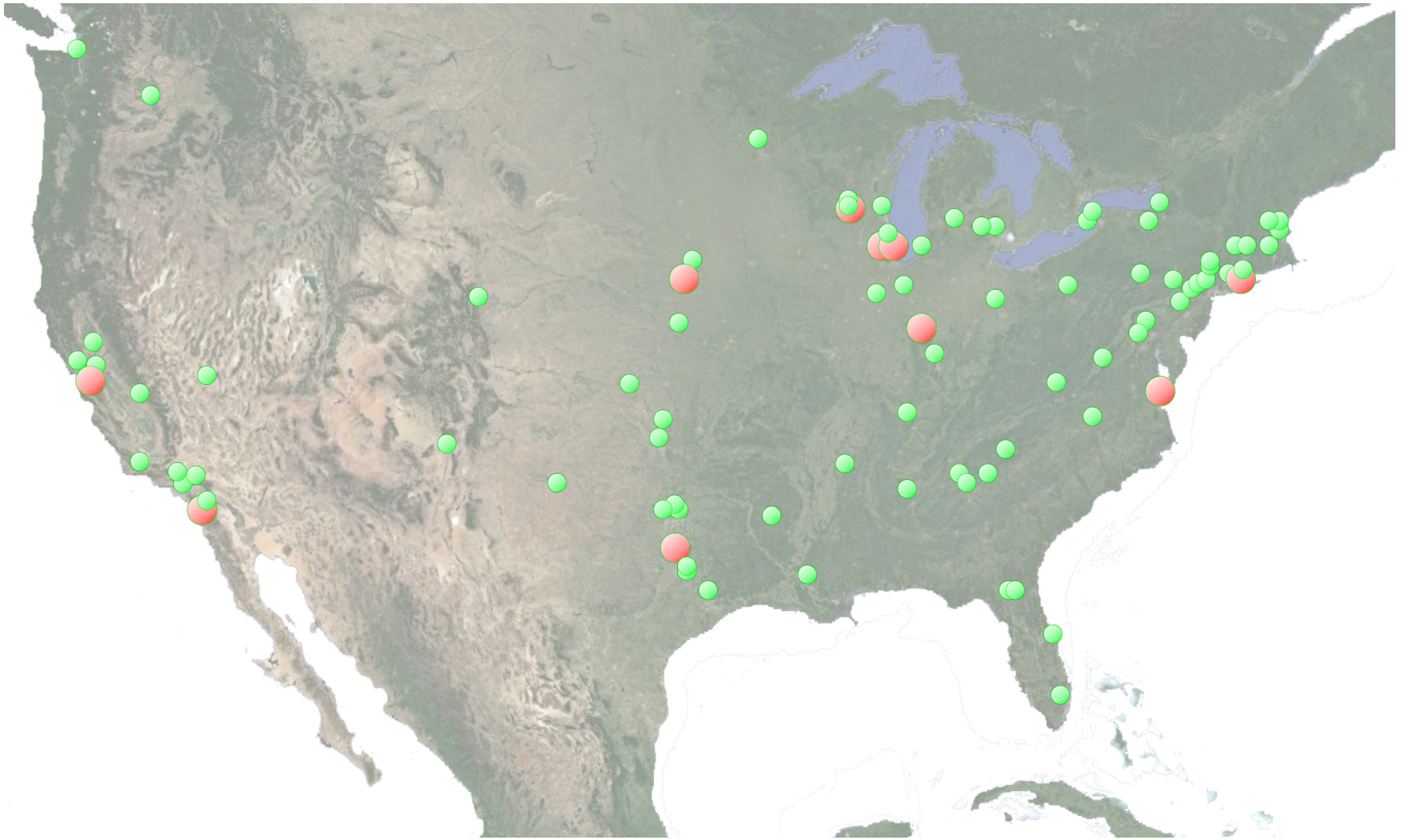
OSG: 140 sites



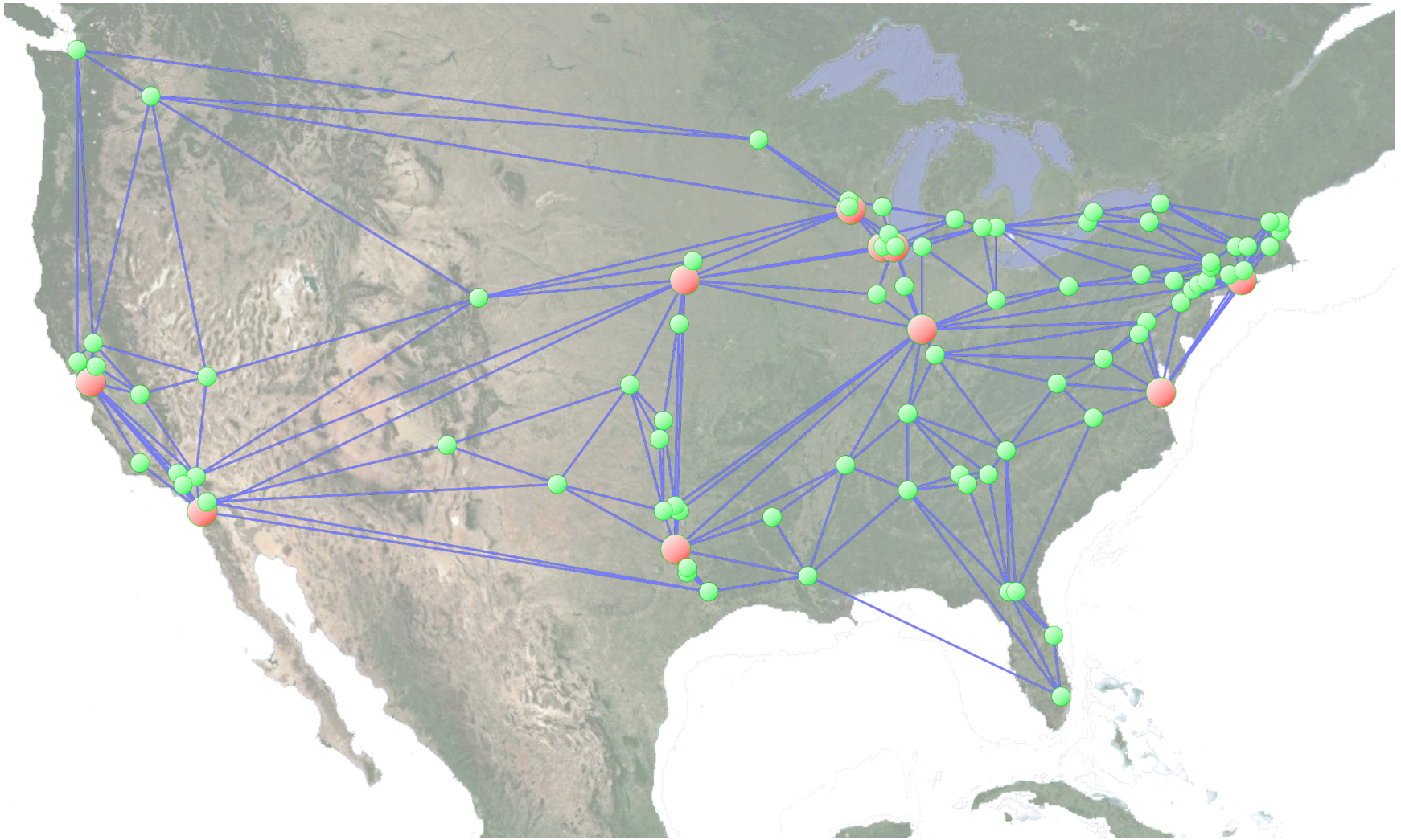
OSG: 140 sites



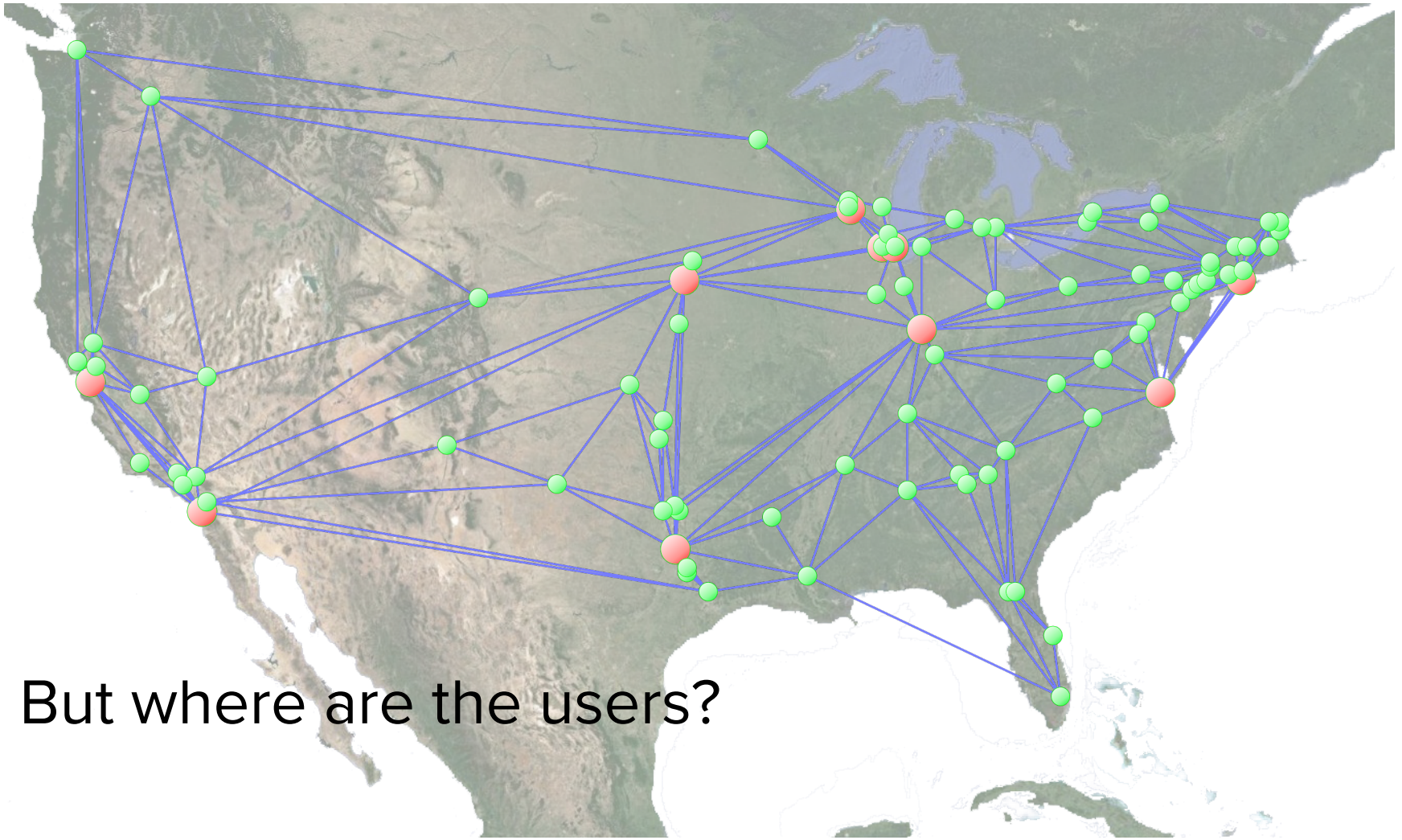
OSG: 140 sites



OSG: 140 *connected* sites…

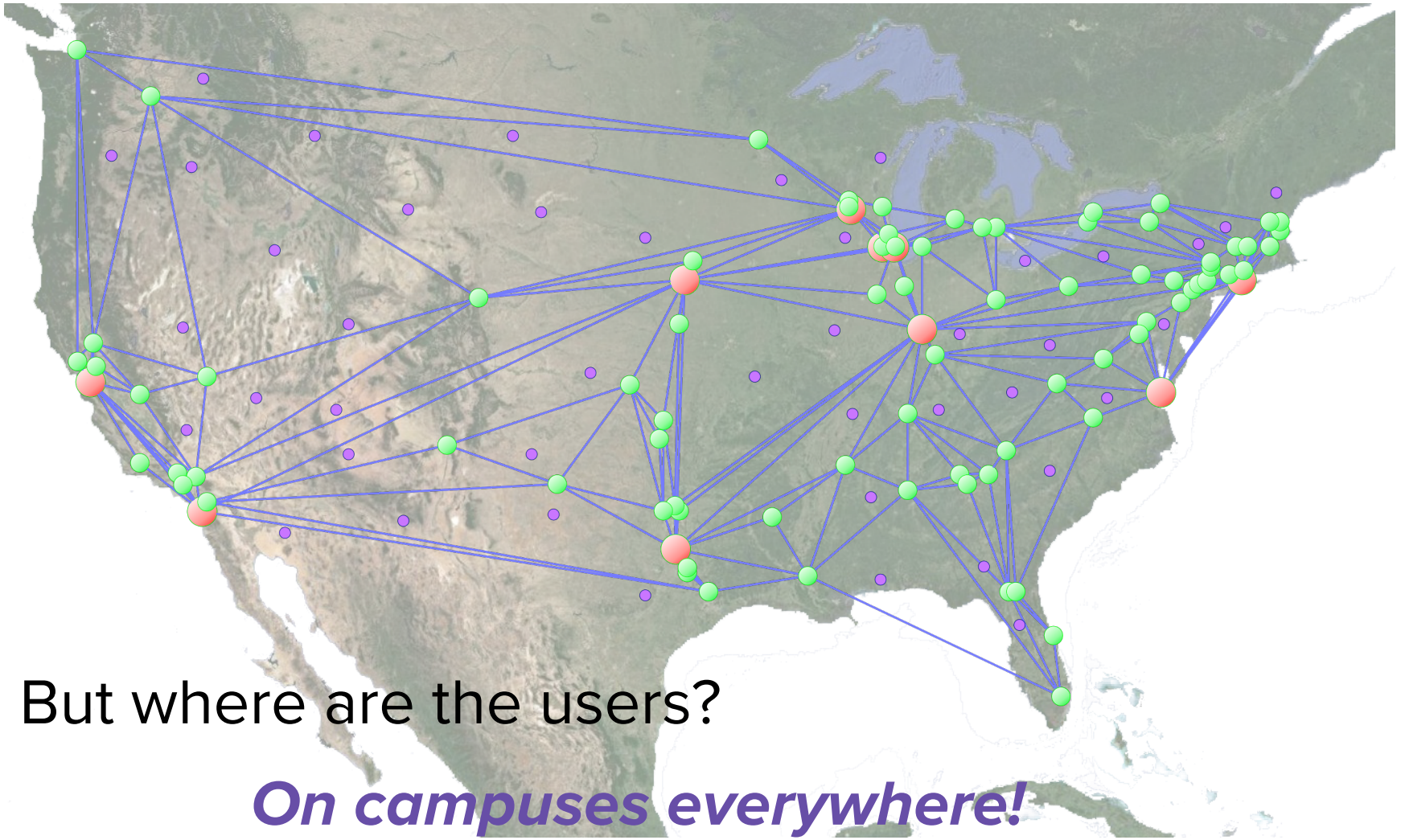


OSG: 140 *connected* sites…



But where are the users?

OSG: 140 *connected* sites…



But where are the users?

On campuses everywhere!

Points of opportunistic entry

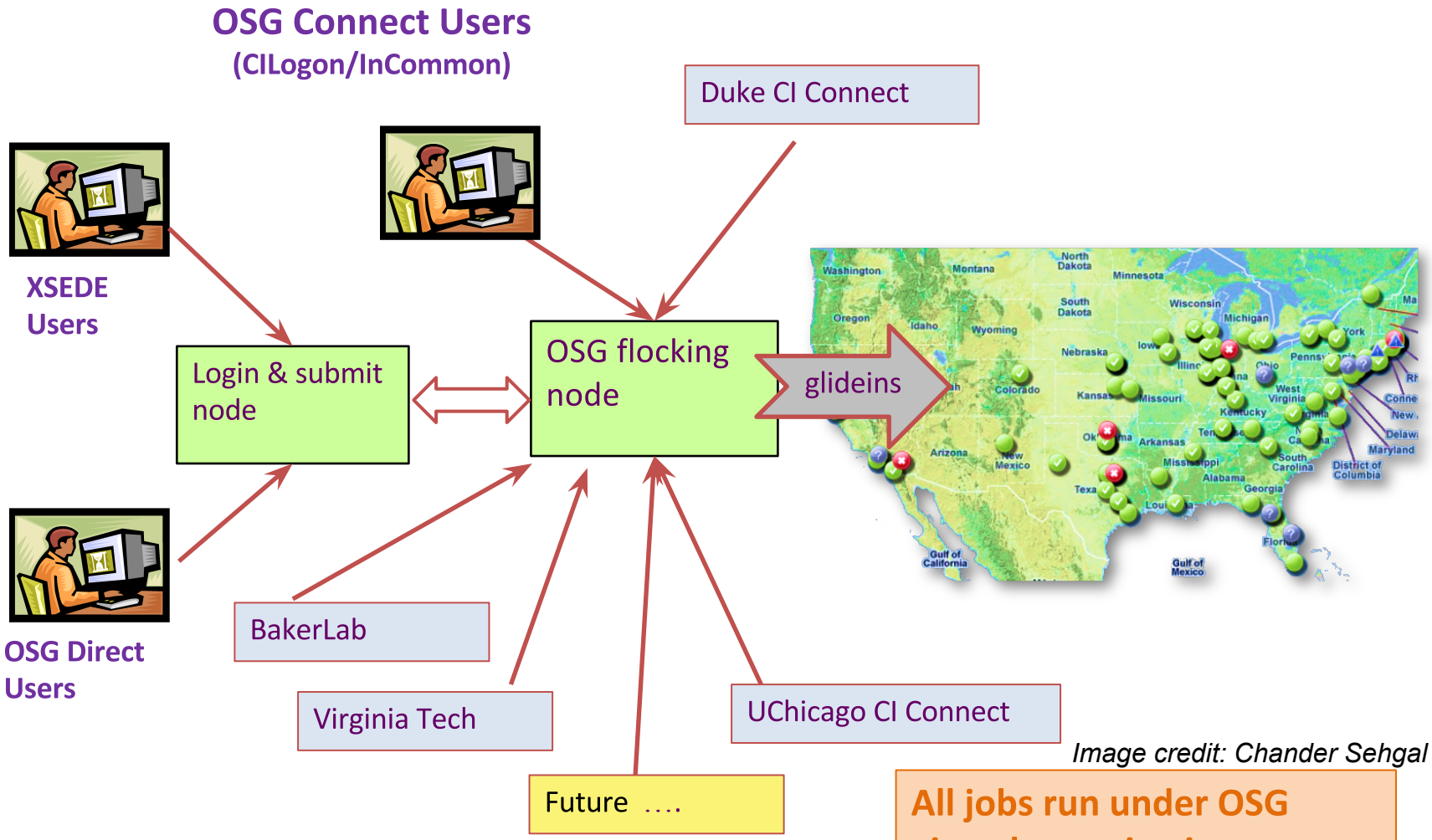


Image credit: Chander Sehgal

All jobs run under OSG virtual organization

Proximity of users to OSG resources

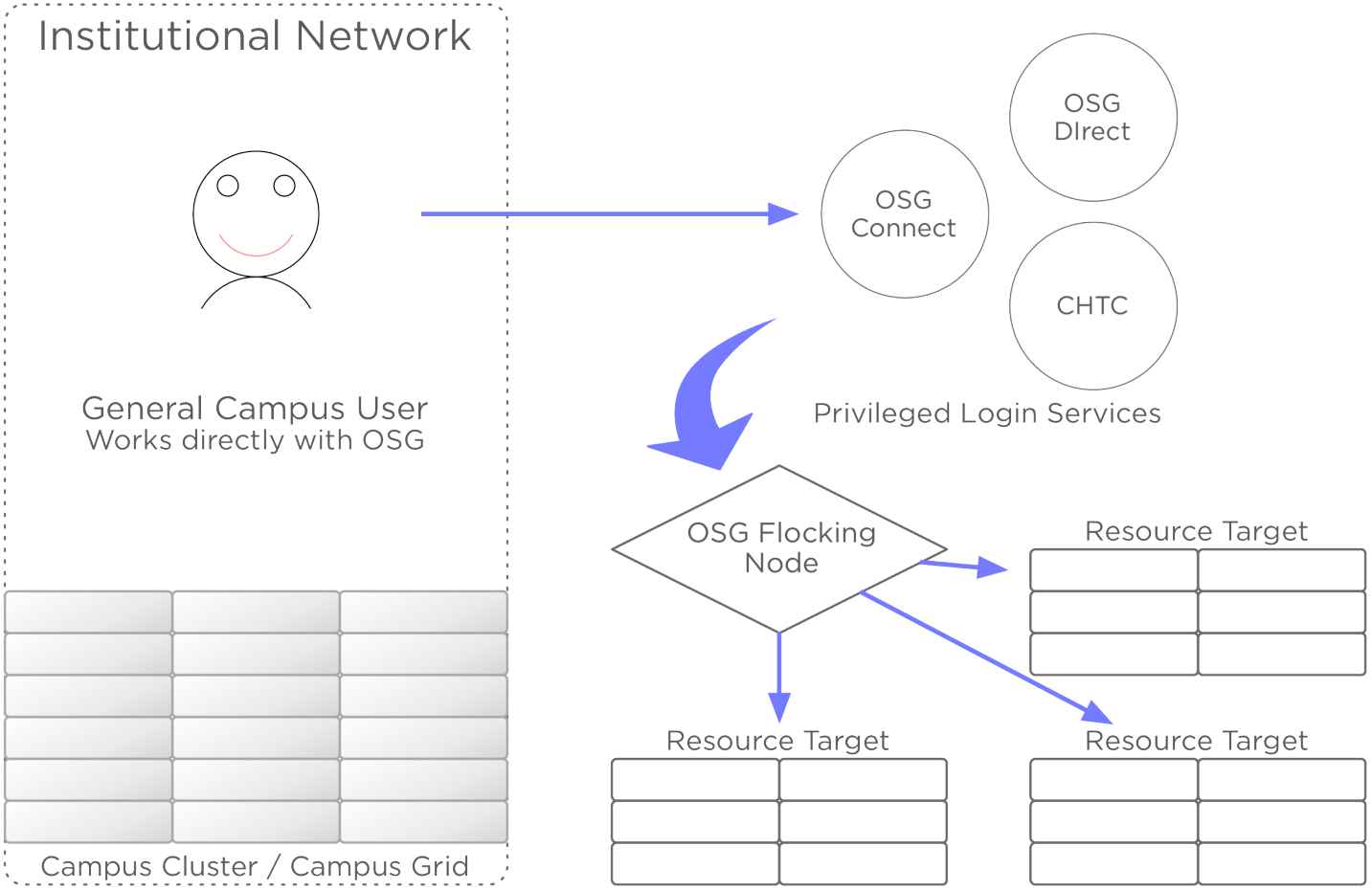
How do we provide access to OSG?

- OSG Connect login
- OSG Direct
- OSG XD (XSEDE allocation)
- Various campus grids
  - CHTC / GLOW
  - Baker Lab
  - Duke Connect
  - ISI…

Is this enough?

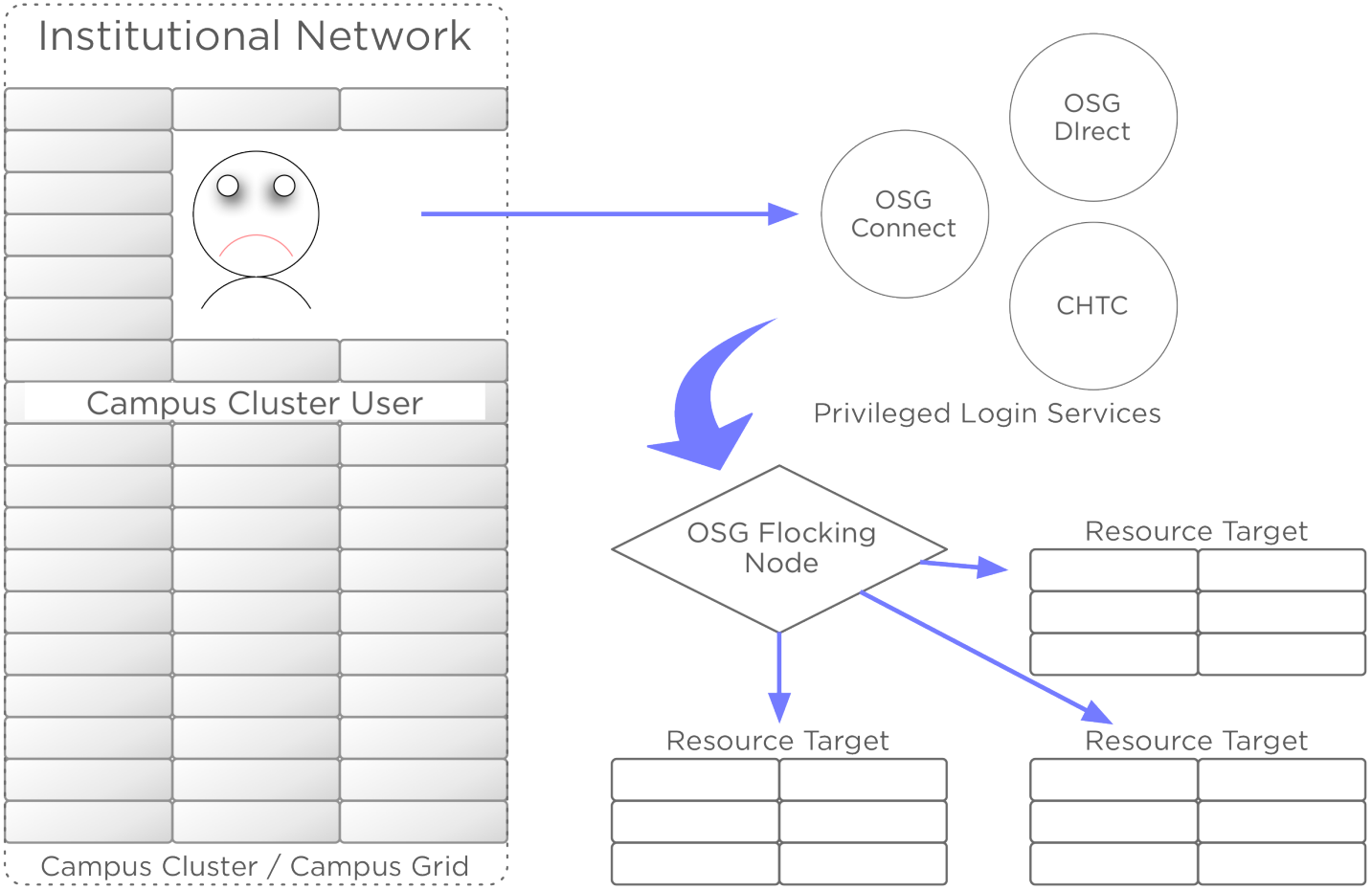
It depends!

Independent users are largely satisfied…



It depends!

...but users of campus resources may be stymied.



What's wrong?

Problem: campus grid users are sheltered within a fully functional ecosystem.

- Local standard configuration
- Local standard data management
- Local standard software access tools
- Local support structure
  - May be less welcoming of obvious institutional interdependence than those of us who work in OSG.
  - "You want my users to do what? At the University of where?"

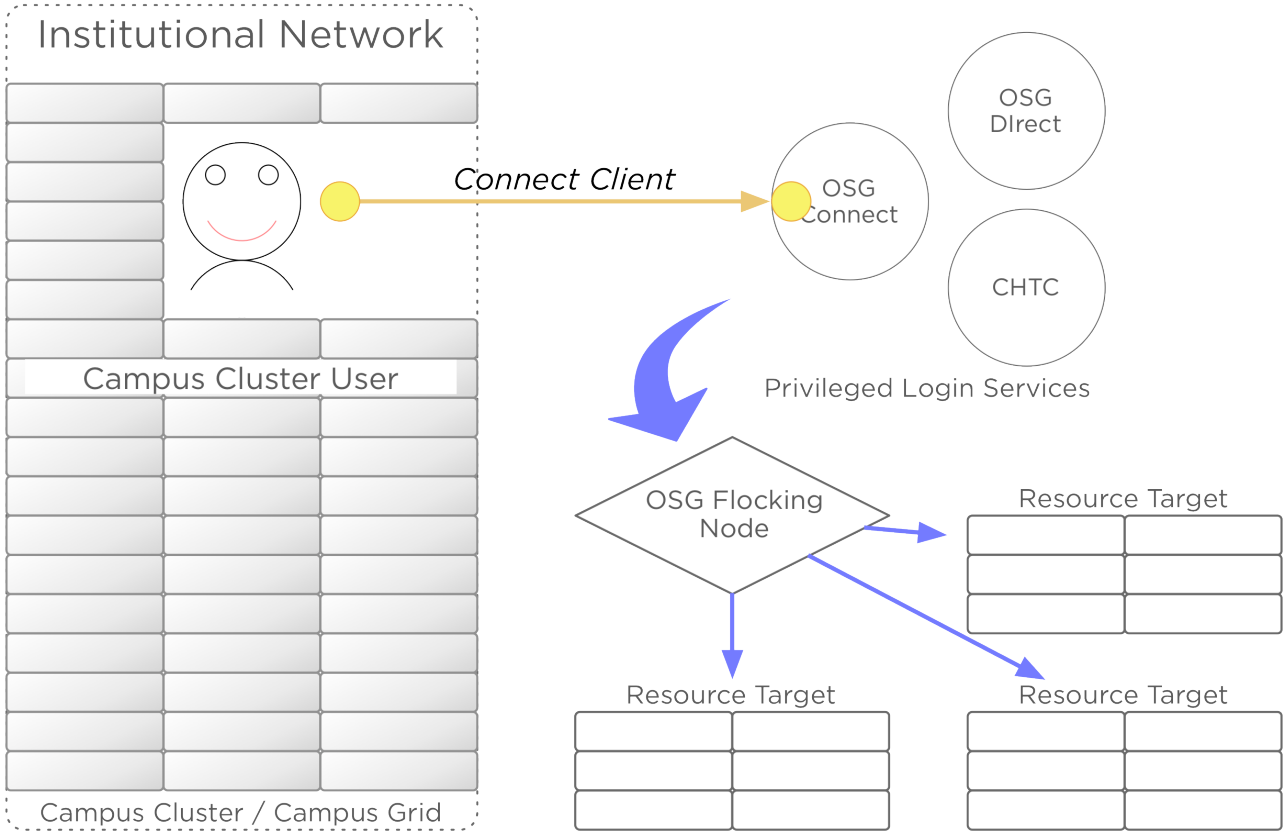
What can we do about this?

Idea: A campus Connect Client to bridge the gap between the campus resource and the national grid gateway.

The Connect Client will co-locate key parts of the interaction within a user's own environment. She can take on the usual OSG entry points at her own readiness, but still *use* the grid before that.

Connect Client bridge

No longer need to “escape” the campus grid environment — at least, not overtly.



What is the Connect Client (v1)?

We started with Bosco.

- Campuses do not want to admin HTCondor
- Users do not want a new submission platform
- Bosco lets your desktop or laptop mimic an HTCondor submission node using SSH and BLAHP

➠ So let's make a client toolset around Bosco, giving each campus grid user a personal Condor system on their home cluster.

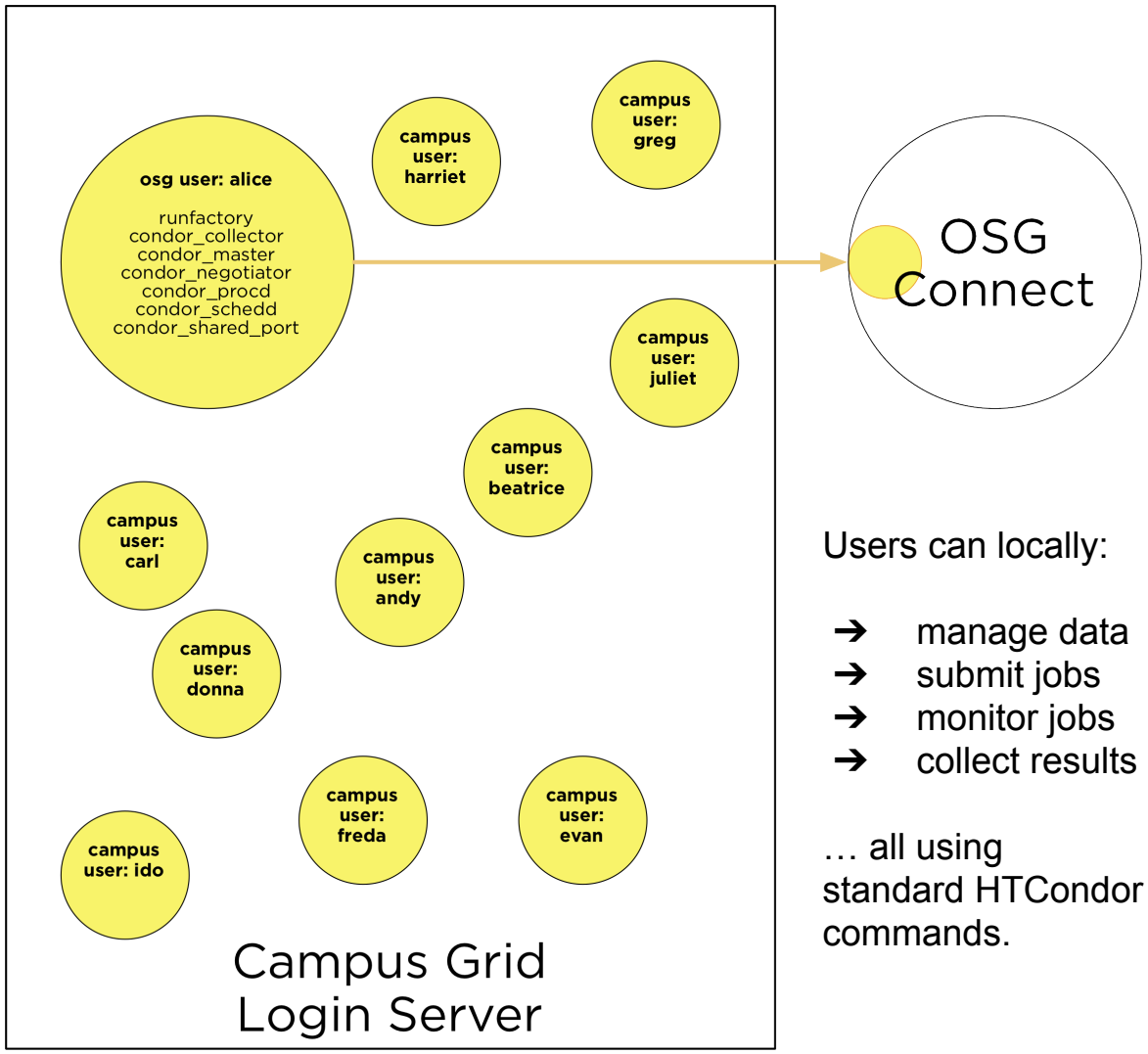
Local setup / installation

- Installs as a `module` for sites that use them.
- Managed by site admin — no download & install steps for user (can be installed by user if necessary)
- Commands:
  - `module load connect`
  - `connect setup`
  - `connect addsite` *`login.osgconnect.net`*
  - `bosco_start`
  - `condor_submit`
  - `condor_etc…`
  - `bosco_stop`

Simple?

Not too bad. The HTCondor/Bosco stack works, of course.

There's a little management overhead for users: starting Bosco, shutting down Bosco when not in use.



Users can locally:

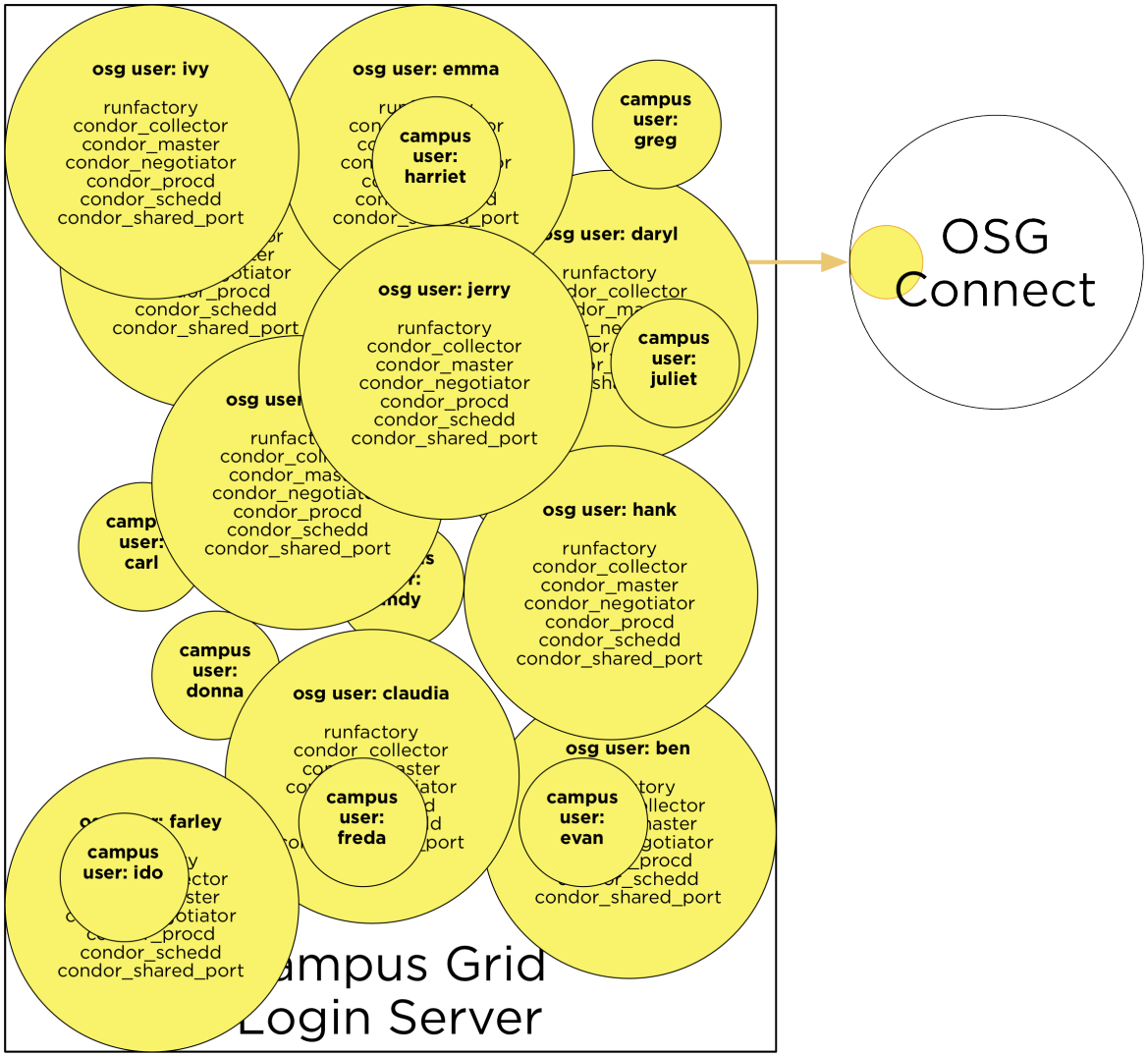
- ➔ manage data
- ➔ submit jobs
- ➔ monitor jobs
- ➔ collect results

… all using standard HTCondor commands.

Scalable?

No:

- 7 long-running processes, ~340 file descriptors (idle)
- condor_shadow processes with vanilla universe
- limited data management with grid universe
- easily outscales the intended design of the typical HPC login node.



Scalable?

Running the full HTCondor submit node stack *per user* turns out to cost a lot of overhead.

- Assorted memory problems with the Bosco supervisor (runfactory)
- Heavy impact from Condor processes, especially condor_shadow
- Local site admins routinely needed to kill users' Bosco/Condor stacks. (Memory and CPU use inconveniencing other users.)

Back to the drawing board

What we need is a *lightweight* Connect Client:

- No large software stacks
- No long-running supervisors
- Able to interact with and exchange files with a full-fledged OSG submission node (e.g. `login.osgconnect.net`)
- Transparent access using established credentials
- Feel like a natural part of the local platform

What is the Connect Client (v2)?

OSG Connect encapsulates many common but unique operations under the *connect* command:

- `connect status`
- `connect watch`
- `connect histogram`
- `connect project`

So we have an umbrella tool for Connect interactions already!

What is the Connect Client (v2)?

Connect Client adds *connect remote*:

- `connect remote setup`
- `connect remote pull / pull / sync`
- `connect remote submit`
- `connect remote q`
- `connect remote history`
- `connect remote rm`
- `connect remote status`
- `...`

Command summary

- `connect remote setup`
  - one-time authorization setup. Creates a new SSH key pair and uses your password to authorize it.
  - *connect remote test* can validate access at any time
- `connect remote push / pull / sync`
  - lightweight access means no supervisors can monitor file readiness for transfer
  - instead, we have explicit commands for uni- or bi-directional file synchronization between local and remote (the “connected” server). The sync occurs over a secure SSH channel.

Command summary

- `connect remote submit`
  - like condor_submit, submits a job from a job control file (submit script). Implicitly performs a *push* beforehand.
- `connect remote q`
  - runs condor_q remotely
- `connect remote history`
- `connect remote status`
- `connect remote rm`
  - also condor_* wrappers

Command summary

- Very straightforward, very simple. Adding wrapped commands is trivial. Bracketing those commands with file sync is also trivial.
- No intention to simply replace “ssh server”. We want to limit access to make the remote capability more direct, more like an extension of the local service. (Advanced users can get regular login accounts.)

Short-term future development

- all necessary capabilities of the base *connect* command should become available as remote commands.
- only *a priori* knowledge is server *user@host*. Other configuration/metadata should be retrieved on demand.
- file synchronization is currently SFTP-based. Smarter file sync (e.g. rsync) should be possible.
- other remote data access support as appropriate
- needs *connect* command on server; eliminate this requirement

Demonstration

```
[402/0]$ module list
Currently Loaded Modulefiles:
 1) vim/7.4          3) emacs/24         5) use.own
 2) subversion/1.8   4) env/rcc          6) slurm/current

[403/0]$ module avail connect-client
--------------------------- /home/dgc/privatemodules ---------------------------
connect-client/1.1
---------------------------- /software/modulefiles -----------------------------
------------------------------ /etc/modulefiles ------------------------------

[404/0]$ module load connect-client

[405/0]$ connect
usage: connect <subcommand> [args]
       connect remote
```

Demonstration

```
[406/2]$ connect remote
usage: connect remote <subcommand> [args]
       connect remote history <condor_history arguments>
       connect remote pull [[localdir] remotedir]
       connect remote push [[localdir] remotedir]
       connect remote q <condor_q arguments>
       connect remote rm <condor_rm arguments>
       connect remote run <condor_run arguments>
       connect remote setup [--replace-keys] [servername]
       connect remote status <condor_status arguments>
       connect remote submit <submitfile>
       connect remote sync [[localdir] remotedir]
       connect remote test [servername]
       connect remote wait <condor_wait arguments>
```

Demonstration

```
[407/0]$ connect remote q dgc
error: SSHError: No key file available.
error: Did you run "connect remote setup"?

[408/10]$ connect remote setup
Password for dgc@login.osgconnect.net:
notice: Ongoing remote access has been authorized at login.osgconnect.net.
notice: Use "connect remote test" to verify access.

[409/0]$ connect remote test
You already have remote access to login.osgconnect.net. There is no need to run setup.

[411/0]$ tutorial quickstart
Installing quickstart (osg)...
Tutorial files installed in ./tutorial-quickstart.
Running setup in ./tutorial-quickstart...
```

Demonstration

```
[412/0]$ cd tutorial-quickstart

[413/0]$ ls
total 192
32 log/       32 short.sh*          32 tutorial02.submit
32 README.md  32 tutorial01.submit  32 tutorial03.submit

[414/0]$ connect remote submit tutorial01.submit
notice: sending README.md as tutorial-quickstart/README.md...
notice: sending short.sh as tutorial-quickstart/short.sh...
notice: sending tutorial01.submit as tutorial-quickstart/tutorial01.submit...
notice: sending tutorial02.submit as tutorial-quickstart/tutorial02.submit...
notice: sending tutorial03.submit as tutorial-quickstart/tutorial03.submit...
notice: sending log/.gitignore as tutorial-quickstart/log/.gitignore...
Submitting job(s).
1 job(s) submitted to cluster 7062512.
```

Demonstration

```
[415/0]$ connect remote q dgc

-- Submitter: login01.osgconnect.net : <192.170.227.195:56133> : login01.osgconnect.net
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
7062512.0   dgc              3/23 23:50   0+00:00:00 I  0   0.0  short.sh

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended

[416/0]$ connect remote q dgc

-- Submitter: login01.osgconnect.net : <192.170.227.195:56133> : login01.osgconnect.net
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
```

Demonstration

```
[417/0]$ connect remote pull
notice: fetching tutorial-quickstart/README.md as README.md...
notice: fetching tutorial-quickstart/short.sh as short.sh...
notice: fetching tutorial-quickstart/tutorial01.submit as tutorial01.submit...
notice: fetching tutorial-quickstart/tutorial02.submit as tutorial02.submit...
notice: fetching tutorial-quickstart/tutorial03.submit as tutorial03.submit...
notice: fetching tutorial-quickstart/job.log as job.log...
notice: fetching tutorial-quickstart/job.output as job.output...
notice: fetching tutorial-quickstart/job.error as job.error...
notice: fetching tutorial-quickstart/log/.gitignore as log/.gitignore...
```

URLs

Connect client

https://github.com/CI-Connect/connect-client/tree/lightweight

OSG Connect

https://osgconnect.net/