



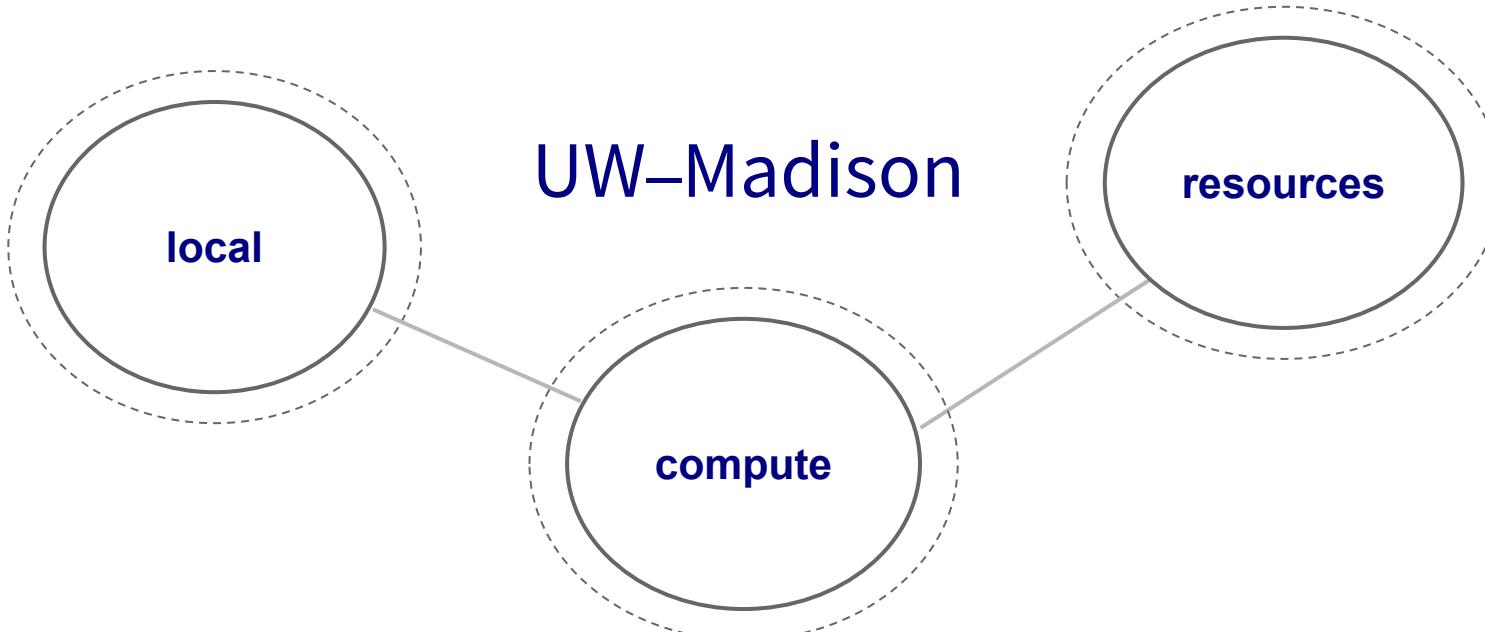
Running Jobs on the Open Science Grid

Brian Lin

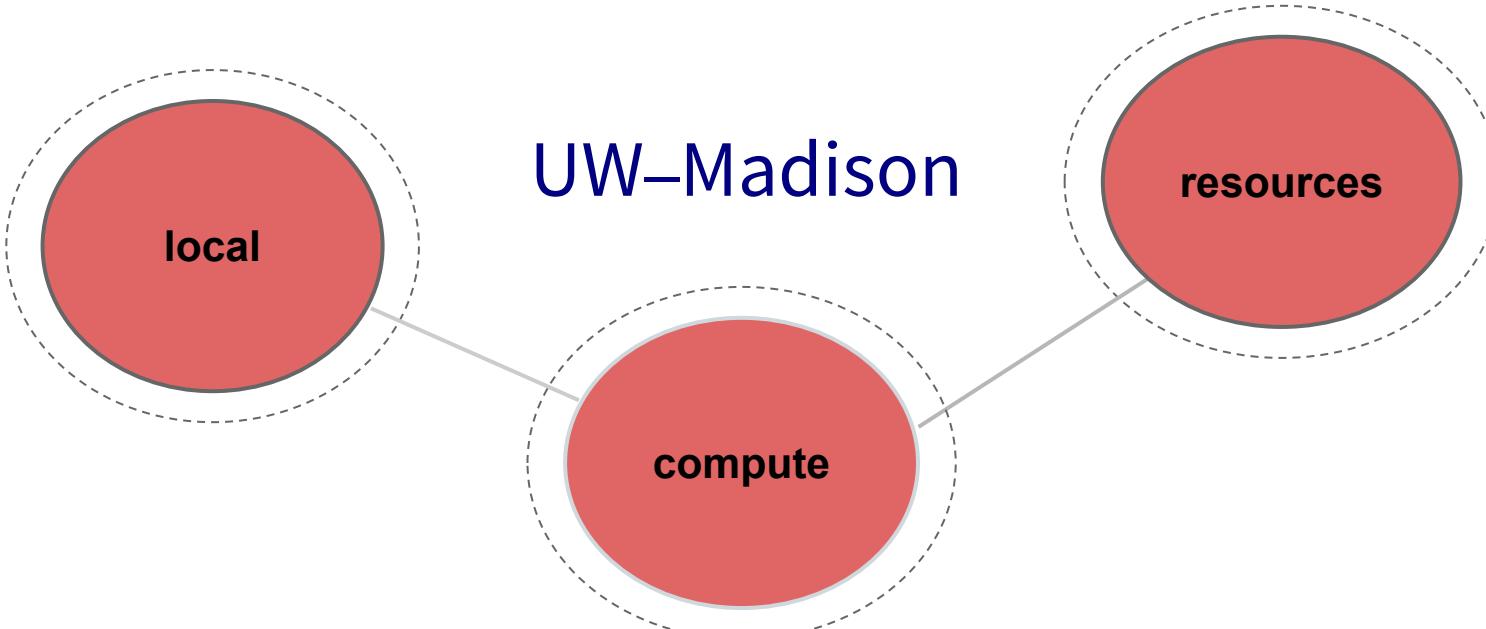
OSG Software Team

University of Wisconsin–Madison

Local High Throughput Computing



Local High Throughput Computing



How do you get more computing resources?

#1: Buy Hardware

- Great for specific hardware/privacy requirements
- Costs \$\$\$
 - Initial cost
 - Maintenance
 - Management
 - Power and cooling
- Rack/floor space
- Obsolescence
- Plan for peak usage, pay for all usage
- Delivery and installation takes time



#2: Use the Cloud - Pay per cycle

- Amazon Web Services, Google Compute Engine, Microsoft Azure, etc.
- Fast start-up
- Costs \$\$\$
- Still needs expertise + management
 - Easier than in the past with the condor_ annex tool
- Does payment fit with your institutional or grant policies?

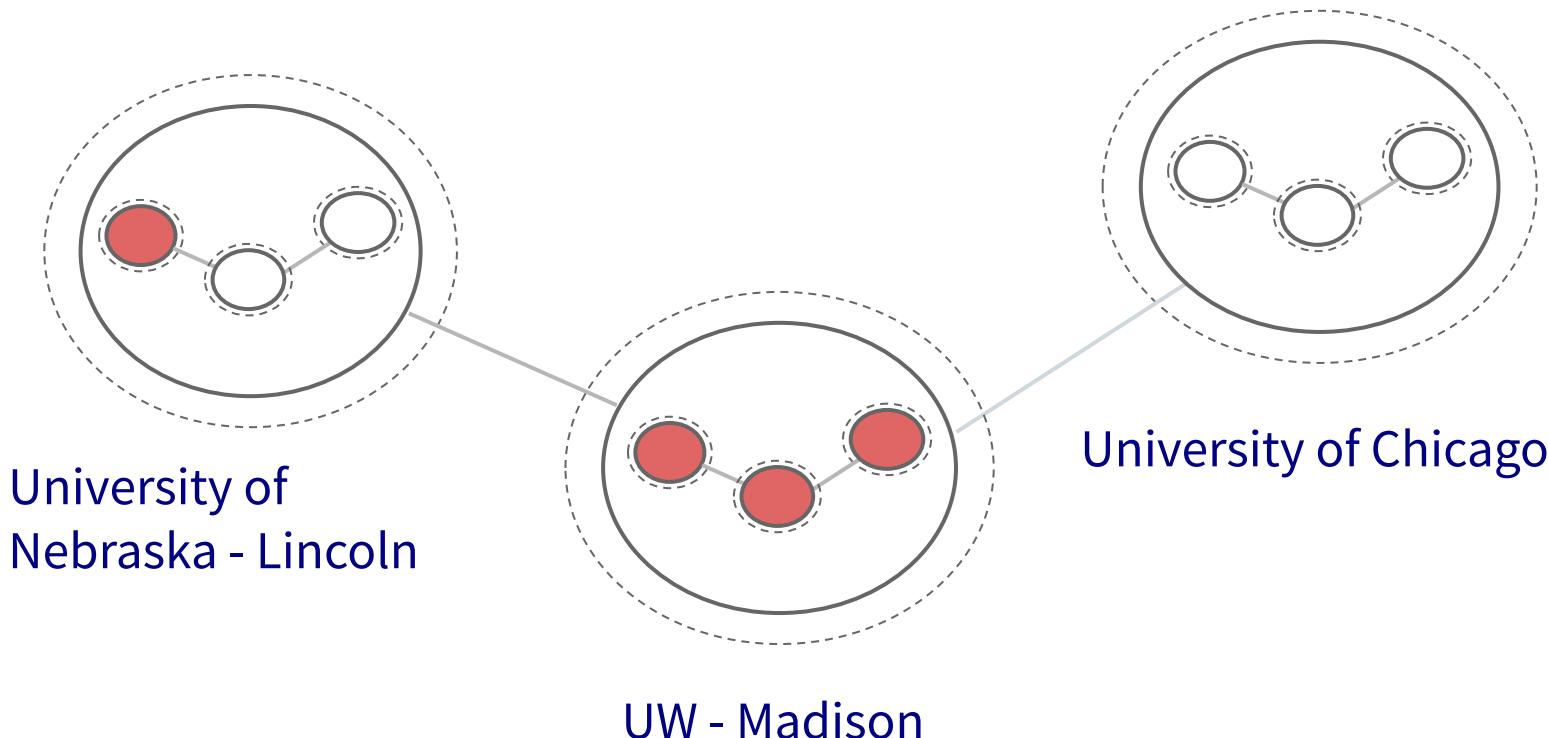


#2: Use the Cloud - ‘Managed’ clouds

- Cycle Computing, Globus Genomics
- Pay someone to manage your cloud resources — still costs \$\$\$
- Researchers and industry have used this to great success
 - Using Docker, HTCondor, and AWS for EDA Model Development
 - Optimizations in running large-scale Genomics workloads in Globus Genomics using HTCondor
 - HTCondor in the enterprise
 - HTCondor at Cycle Computing: Better Answers. Faster.

#3: *Distributed* High Throughput Computing (dHTC)

#3: Share Resources - Distributed HTC

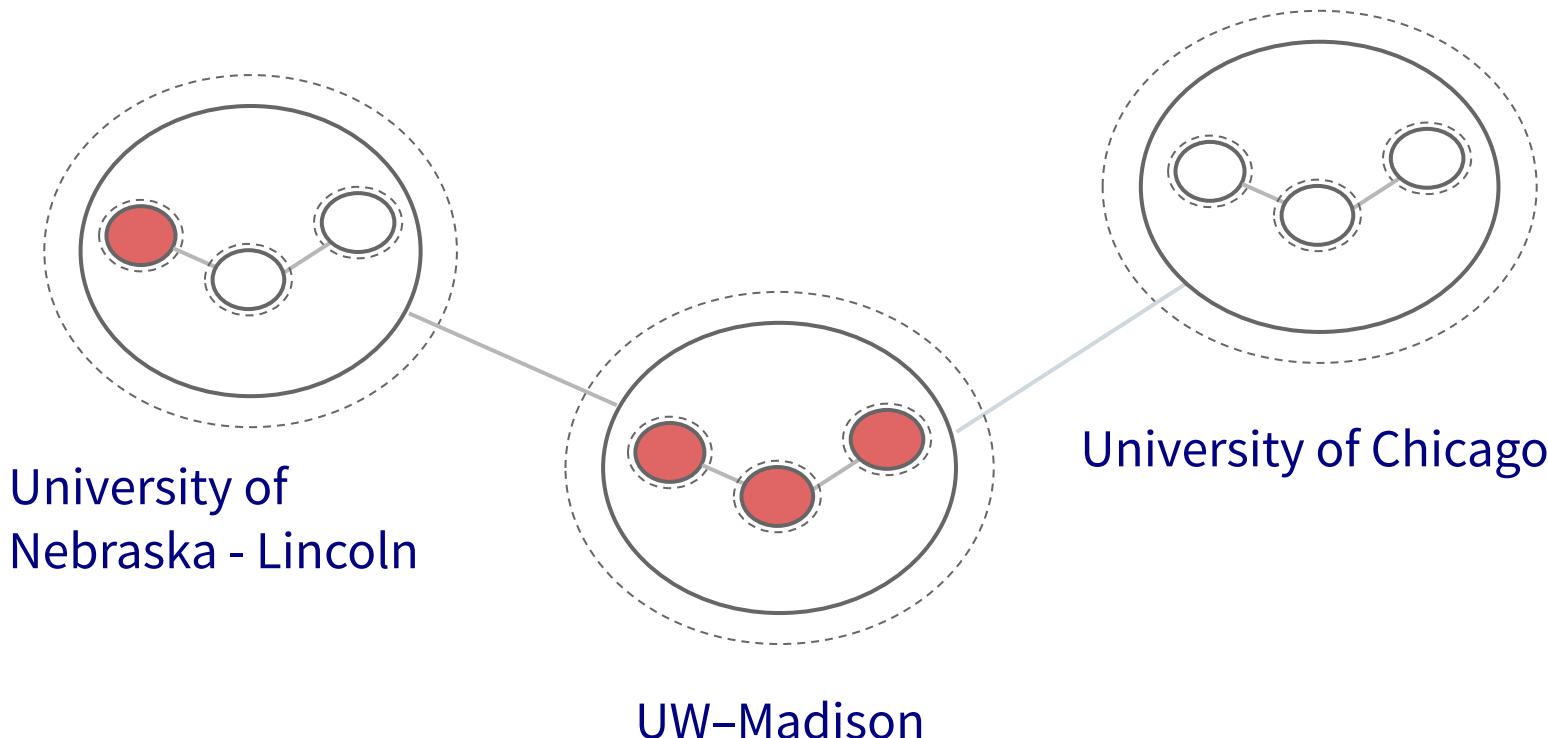


Manual Job Division



- Obtain login access
- Query each cluster for idle resources
- Divide and submit jobs based on resource availability

#3: Share Resources - Distributed HTC

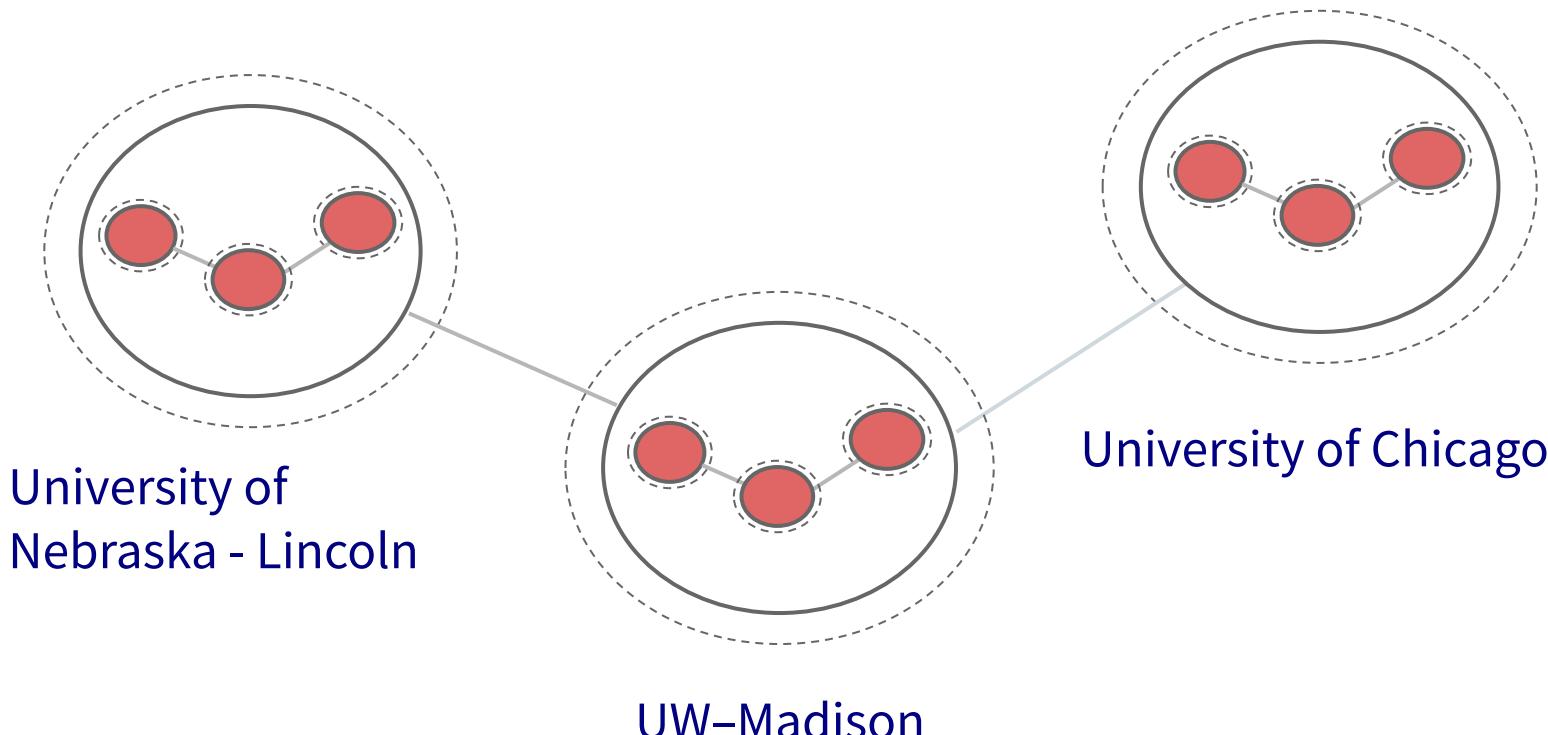


University
of
Nebraska - Lincoln

UW-Madison

University of Chicago

#3: Share Resources - Distributed HTC



Manual Job Division - Shortcomings

- Fewer logins = fewer potential resources, more logins = more account management
- How will you get accounts?
- Not all clusters use HTCondor — other job schedulers e.g., Slurm, PBS/Torque, etc.
- Querying clusters and dividing jobs is tedious and inaccurate

Automatic Job Division - Shortcomings



Homer: Kids: there's three ways to do things; the right way, the wrong way and the Max Power way!

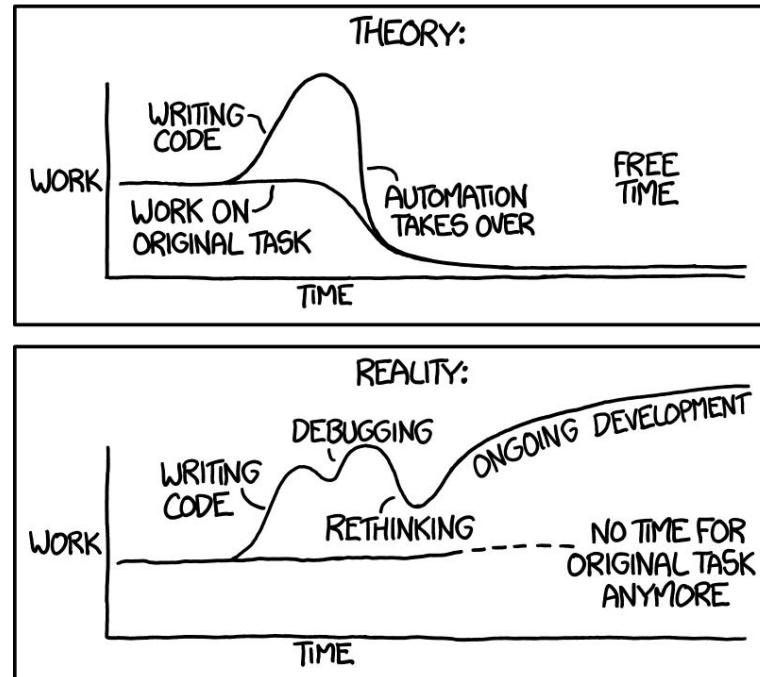
Bart: Isn't that the wrong way?

Homer: Yeah, but faster!

Groening, M (Writer), Michels, P. (Director) . (1999). Homer to the Max [Television Series Episode]. In Scully, M. (Executive Producer), *The Simpsons*. Los Angeles, CA: Gracie Films

Automatic Job Division - Shortcomings

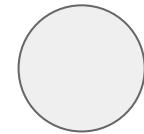
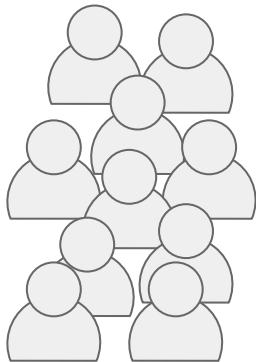
"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"



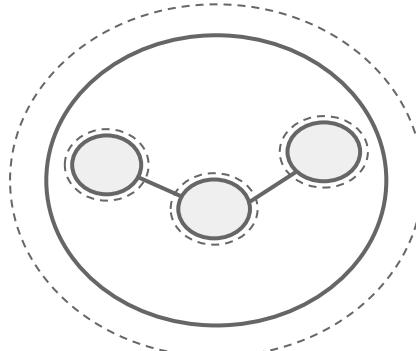
#3: Share Resources - Requirements

- Minimal account management
- No job division
- HTCondor only!
- No resource contribution requirements

The OSG Model

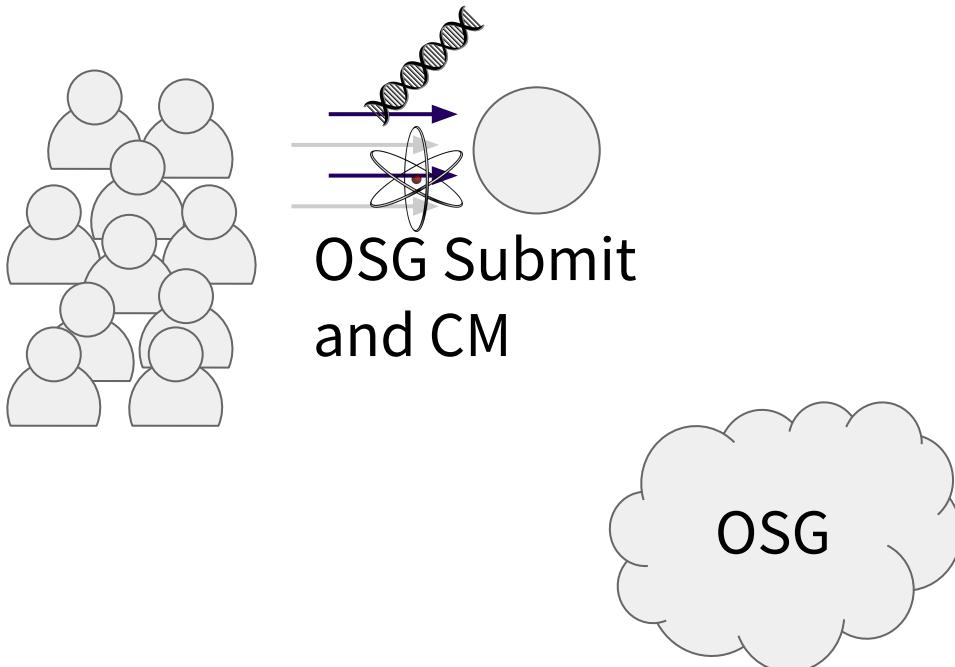


OSG Submit
and CM

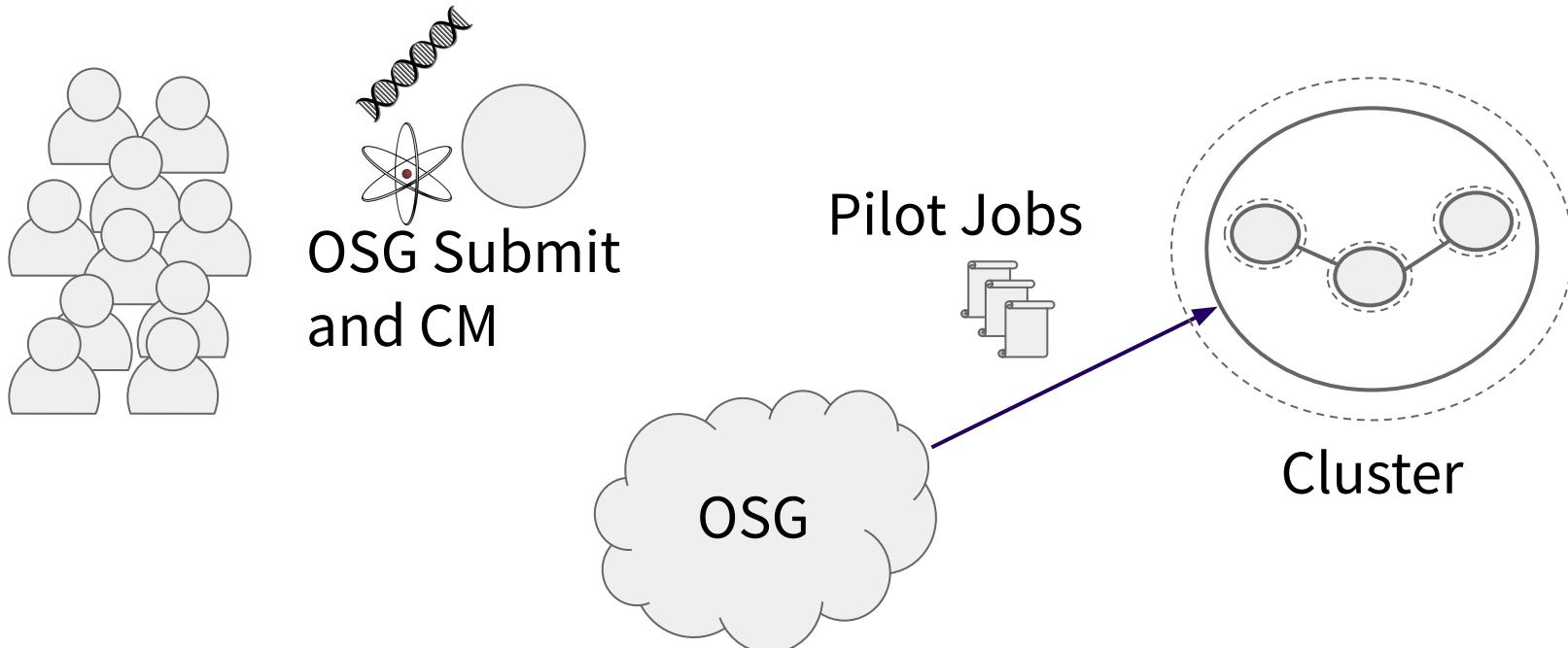


Cluster

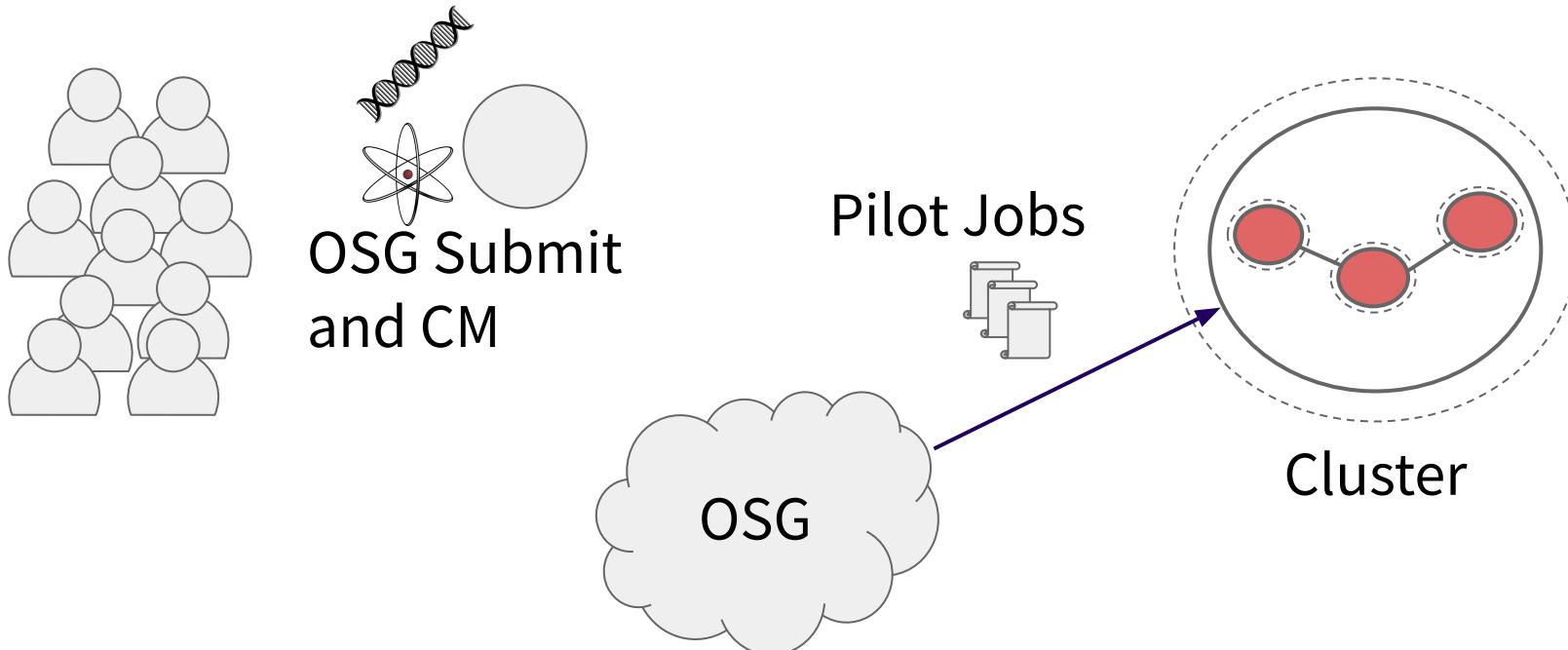
The OSG Model



The OSG Model

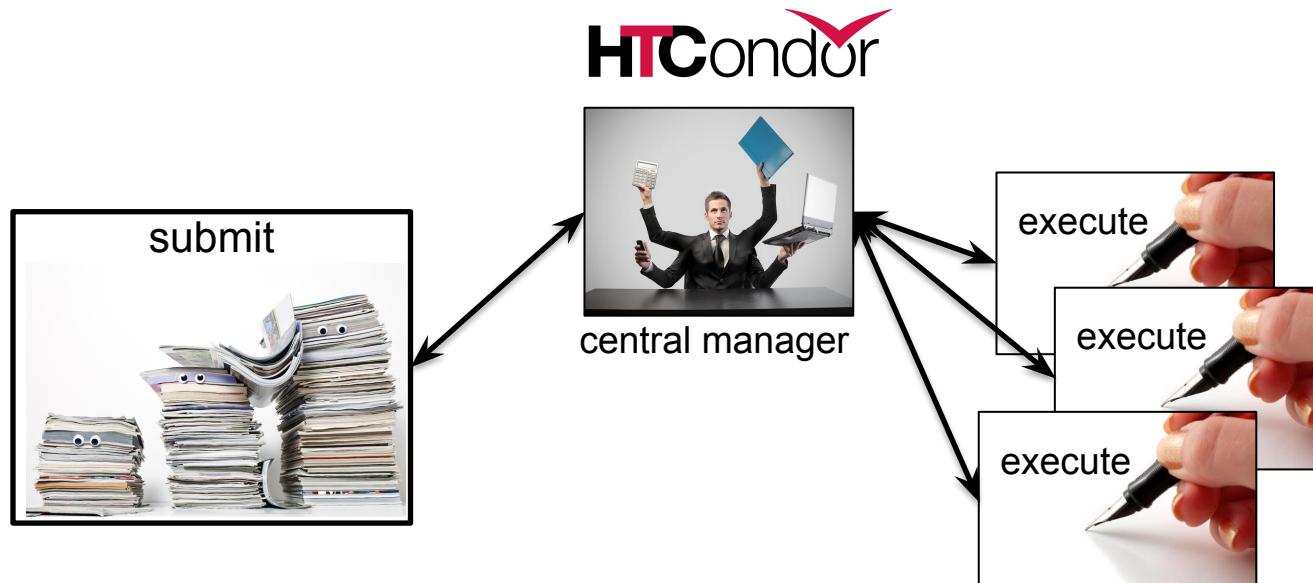


The OSG Model

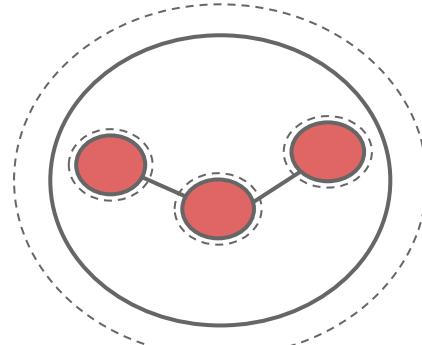
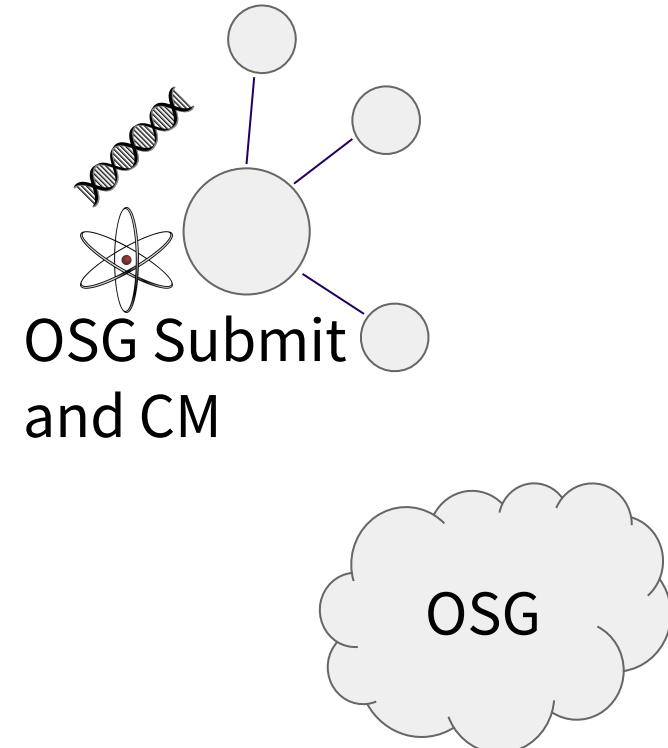
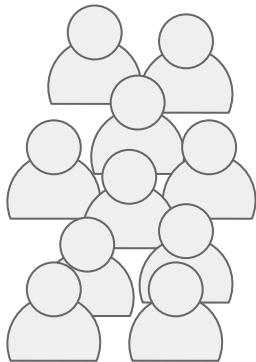


Job Matching

- On a regular basis, the central manager reviews Job and Machine attributes and matches jobs to slots.



The OSG Model



The OSG Model - Jobs in Jobs



Photo Credit: Shereen M, Untitled, Flickr <https://www.flickr.com/photos/shereen84/2511071028/> (CC BY-NC-ND 2.0)

The OSG Model - Details

- Pilot jobs (or pilots) are special jobs
- Pilots are sent to clusters with idle resources
- Pilot payload = HTCondor execute server software
- Execute server reports to the Open Science pool
- Pilots lease resources from OSG clusters:
 - Lease expires after a set amount of time or lack of demand
 - Leases can be revoked!
- On average, the Open Science pool has 10k total cores and most users get 500+ cores at a time!



#3: Share Resources - Requirements

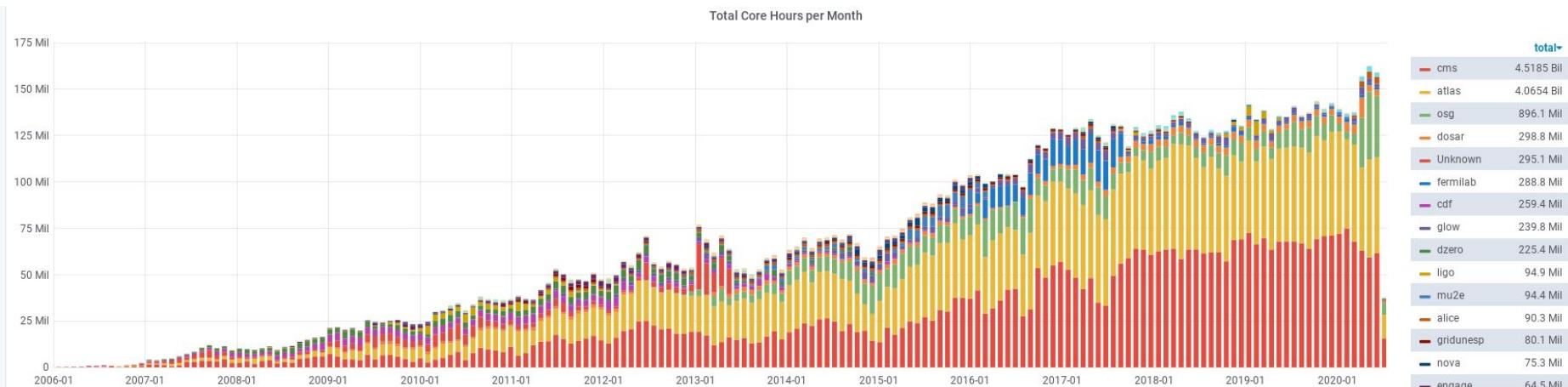
- Minimal account management: only one submit server
- No job division: only one HTCondor pool
- HTCondor only: pilots report back as HTCondor slots, you'll be using an HTCondor submit server
- No resource contribution requirements: the OSG doesn't require that users "pay into" the OSG. Approved researchers can use OSG for free!

The OSG Model - Collection of Pools

- Your jobs will run in the Open Science pool (open to individual researchers and campuses)
- The Open Science pool is one of many!
- Separate pools for each Virtual Organization (VO)



The OSG Model - Collection of Pools



Pilot jobs are awesome!



What's the Catch?

dHTC requires complex machinery but OSG
manages the hard bits so you don't have to!

#1: Heterogenous Resources

Accounting for differences between the
OSG and your local cluster

Clusters of the OSG



Source: <http://display.opensciencegrid.org/>

Het. Resources - Software

- Different variants of Linux (Red Hat based)
- Varying software versions (e.g., at least Python 2.6)
- Varying software availability (e.g., no BLAST*)

Solution: Make your jobs more portable (more in tomorrow's talk and exercises)

Het. Resources - Hardware

- CPU: Mostly single core
- RAM: Mostly < 8GB
- GPU: Limited #s but more being added
- Disk: No shared file system (more next Tuesday)

Solution: Where possible, split up your workflow to make your jobs more high throughput!

#2: With Great Power Comes Great Responsibility

How to be a good netizen

Resources You Don't Own

- Primary resource owners can kick you off for any reason (generally if your job is using too many resources)
- No local relationships
- No sensitive data!



Be a Good Netizen!

- Use of shared resources is a privilege
- Only use the resources that you request
- Be nice to your submit servers

Solution: Test jobs on local resources with
`condor_submit -i` (covered in tomorrow's
exercises)

#3: Slower Ramp Up

Leasing resources takes time!

Slower Ramp Up

- Adding slots: pilot process in the OSG vs slots already in your local pool
- Not a lot of time (~minutes) compared to most job runtimes (~hours)
 - Small trade-off for increased availability
 - Tip: If your jobs only run for < 10min each, consider combining them so each job runs for at least 30min

Job Robustification

- Test small, test often
- Specify output, error, and log files at least while you develop your workflow
- In your own code:
 - Storing intermediate results (i.e., self checkpointing)
 - Defensive troubleshooting (`hostname`, `ls -l`, `pwd`, `condor_version` in your wrapper script)
 - Add simple logging (e.g. `print`, `echo`, etc). Be strategic and don't fill your disk with logs!

#4: dHTC Security

The internet can be a scary place!

dHTC Security

- OSG does its best but security is a game of risk mitigation, not perfection
 - OSG uses secure technologies to verify the identities of distributed servers
 - OSG Security Team tracks software vulnerabilities and responds to security incidents
- Not just any old cluster or user can join the OSG! VOs approve users, cluster owners verify servers, and OSG verifies clusters
- But there are thousands of servers and users!

So What Can You Do?

- You are using a shared computer that you don't own so take basic precautions!
- Protect your data:
 - No files that can be overwritten by other users (i.e., not world writable)
 - No private data or software
- Protect your account
 - Do not share your account
 - Use good passwords (and a password manager)
 - Use SSH keys wherever possible

Questions?

Coming next:

- Grid exercises:

[https://opensciencegrid.org/virtual-school-pilot-2020/
#materials/#grid](https://opensciencegrid.org/virtual-school-pilot-2020/#materials/#grid)

- New submit host: login04.osgconnect.net
 - Set a default project for your login04 account:
\$ connect project
- Tomorrow: Working with real software