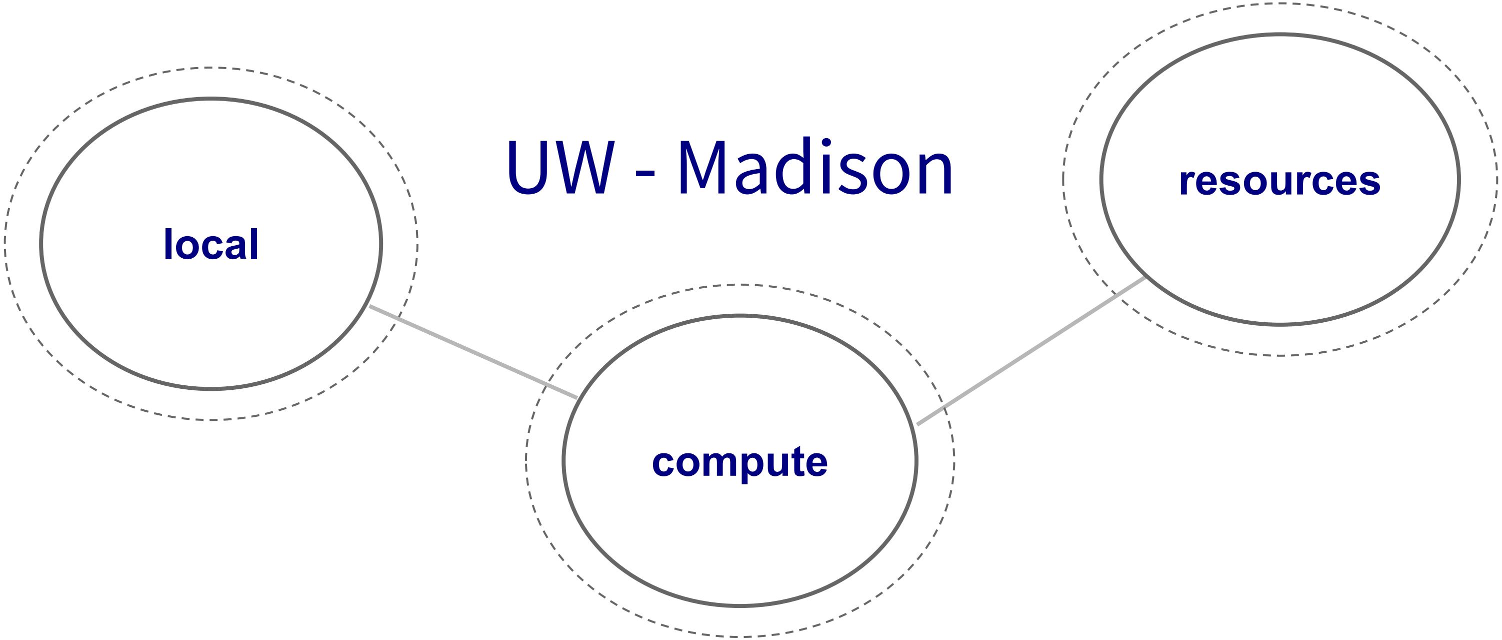
Introduction to DHTC

Brian Lin

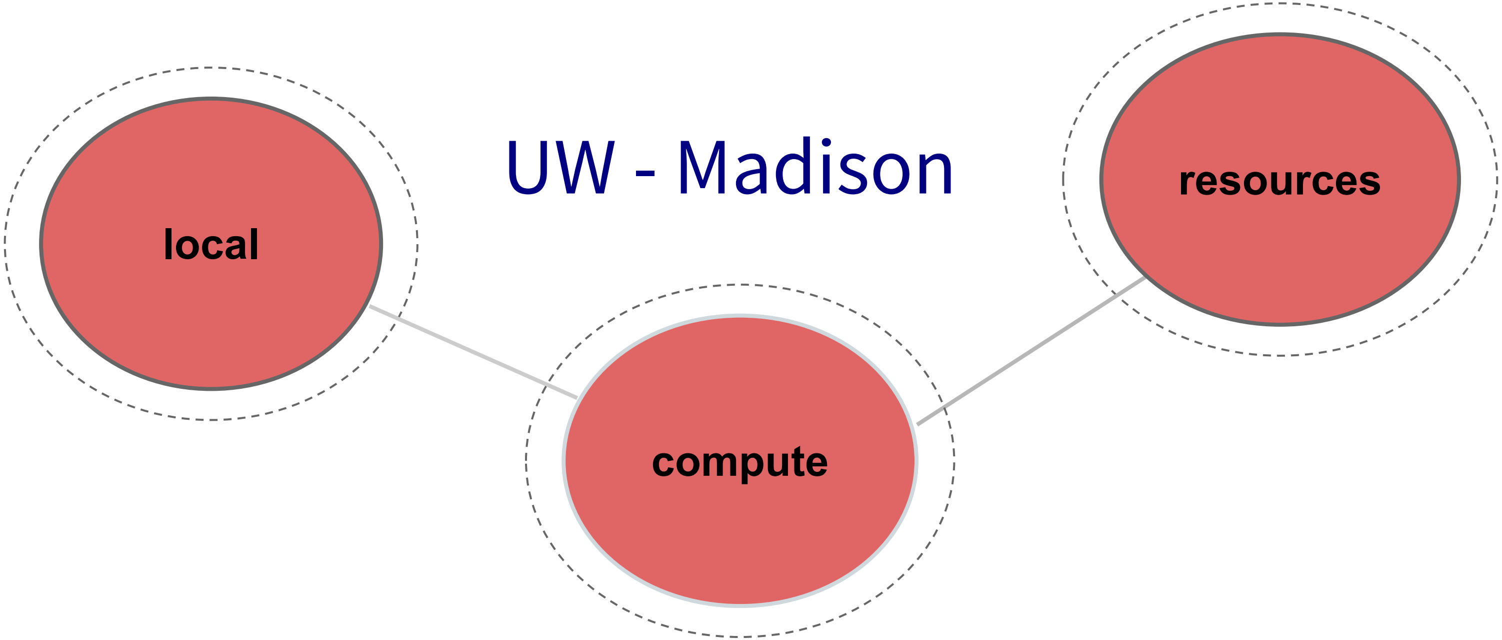
OSG Software Team

University of Wisconsin - Madison

Local High Throughput Computing



Local High Throughput Computing



How do you get more computing resources?

#1: Buy Hardware

#1: Buy Hardware

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

#2: Use the Cloud

#2: Use the Cloud - Pay per cycle

- e.g. Amazon Web Services, Google Compute Engine, Microsoft Azure, Rackspace
- Fast spin-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?

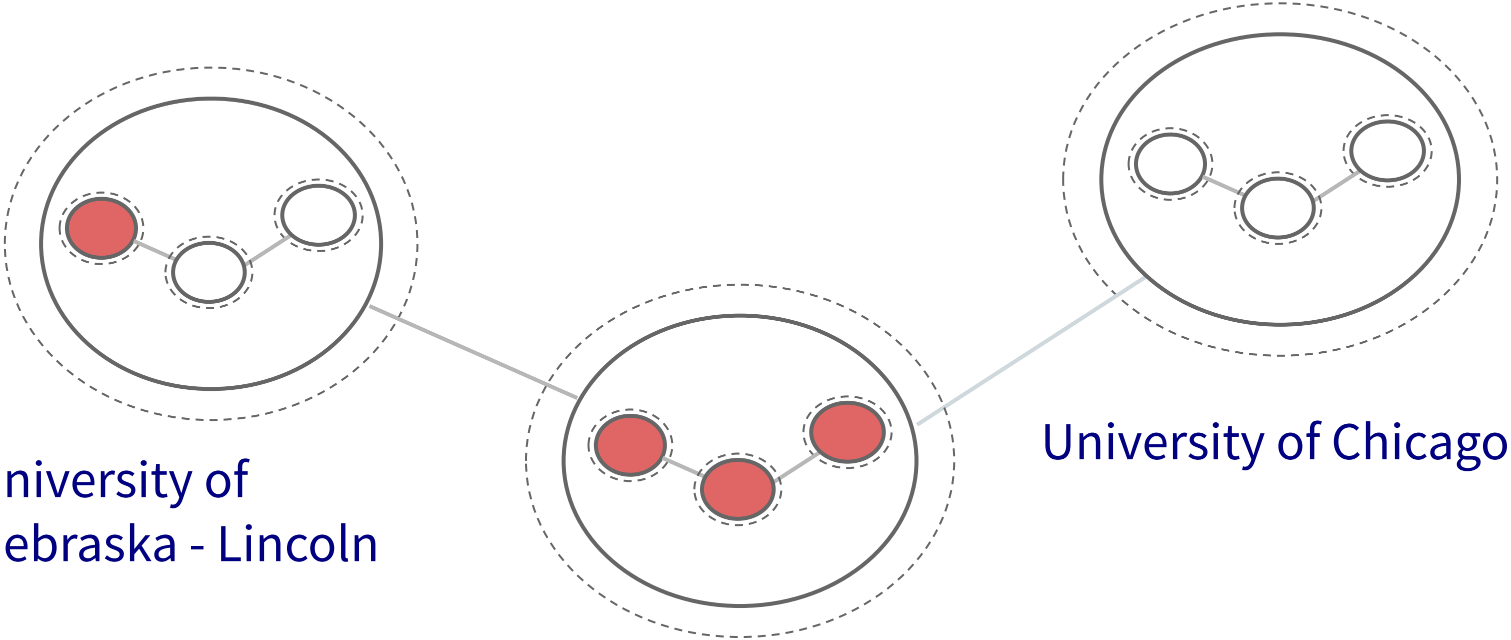
#2: Use the Cloud - ‘Managed’ clouds

- e.g. 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.

Open Science Grid

#3: Share Resources

#3: Share Resources - Distributed HTC



University of Nebraska - Lincoln

University of Chicago

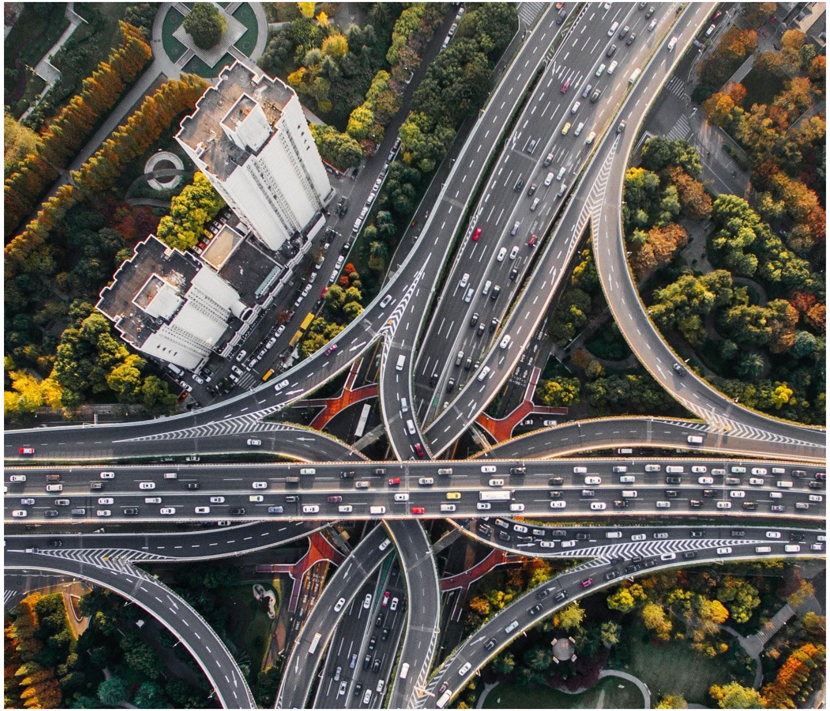
UW - Madison

i.

Split Up Your Jobs Manually

Let’s start sharing!

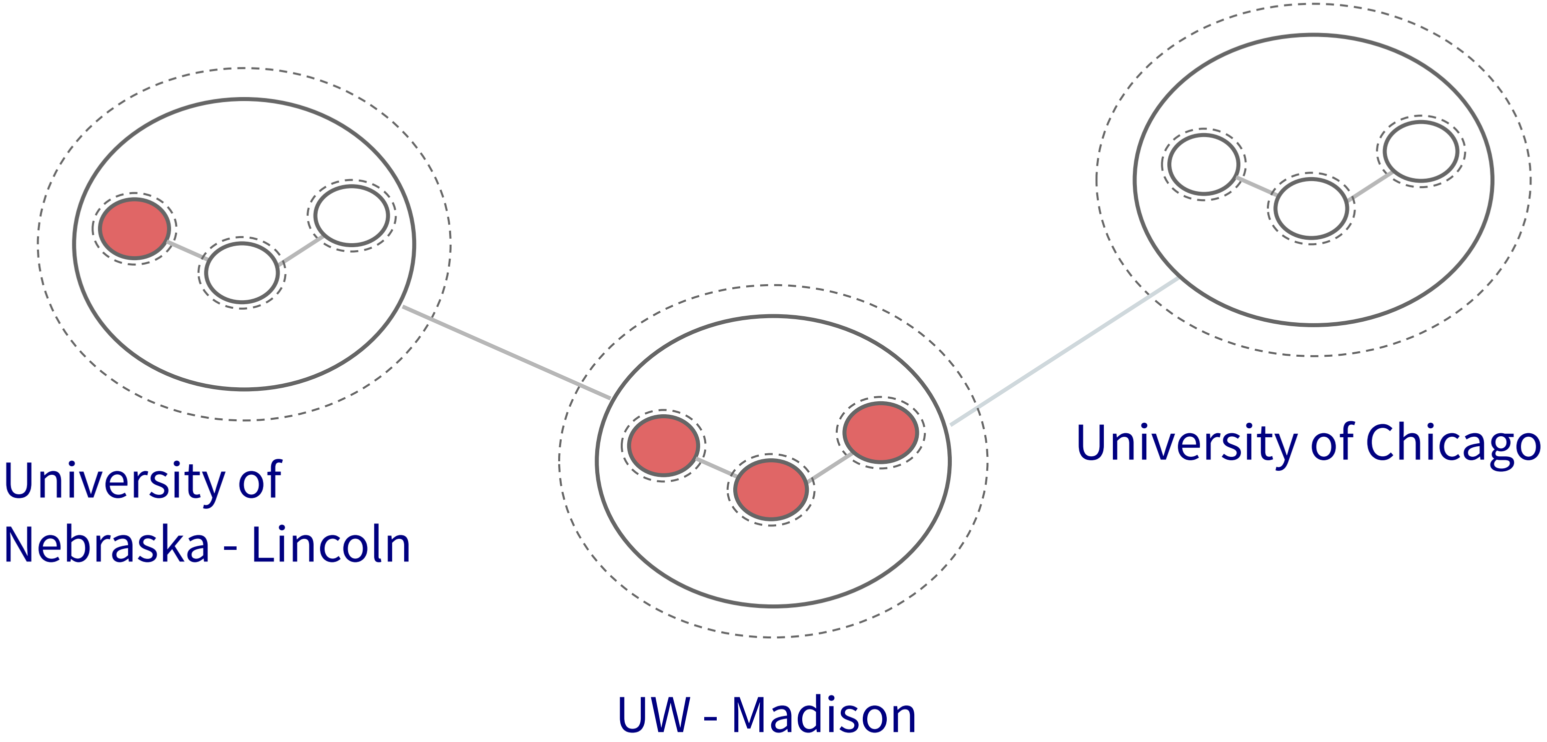
Manual Job Split



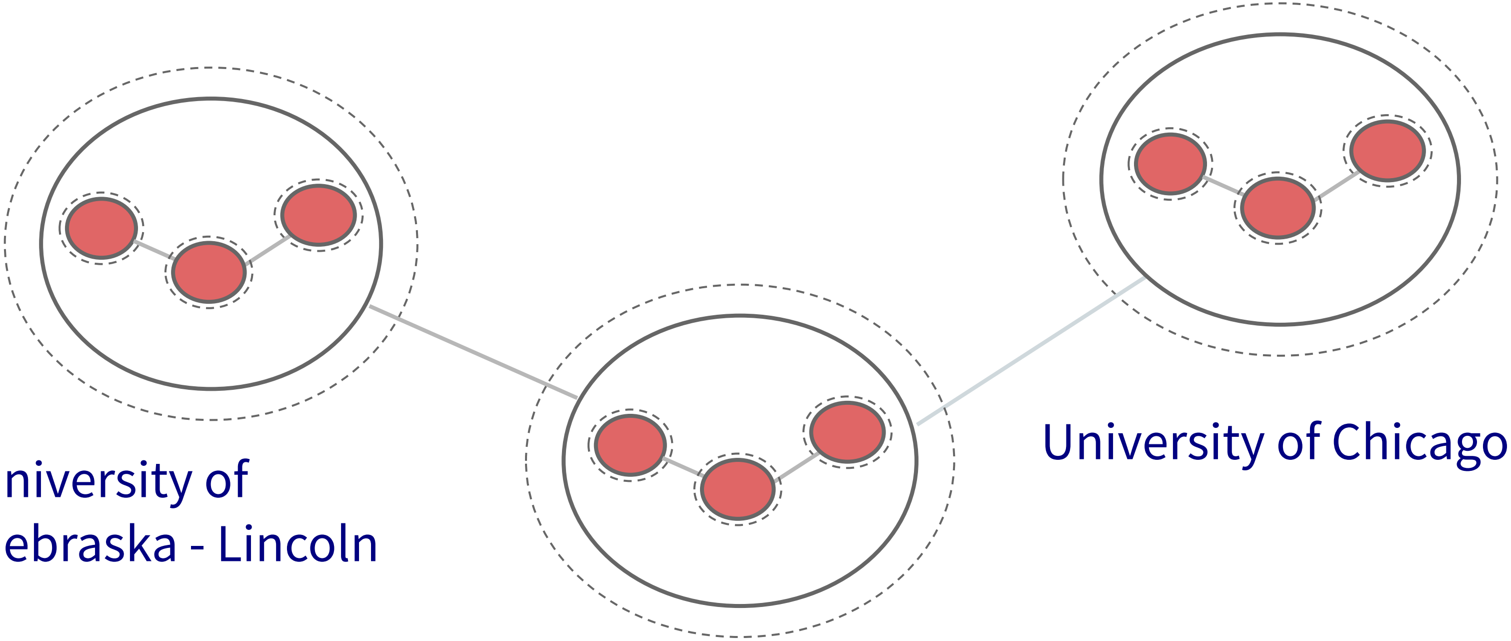
- Obtain login access
- Query each cluster for idle resources
- Split and submit jobs based on resource availability

Photo by Denys Nevozhai on Unsplash

#3: Share Resources - Distributed HTC



#3: Share Resources - Distributed HTC



University of Nebraska - Lincoln

UW - Madison

University of Chicago

Manual Job Split - Shortcomings

- Fewer logins = fewer potential resources
- More logins = more account management
- Why would they give you accounts? Are your friends going to want CHTC accounts?
- Querying and splitting jobs is tedious and inaccurate
- Not all clusters use HTCondor — other job schedulers e.g., SLURM, PBS, etc.
- Pools are independent — workflows must be confined to a single pool

ii.

Split Up Your Jobs Automatically

Let the computers do the work

Automatic Job Split - Shortcomings

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

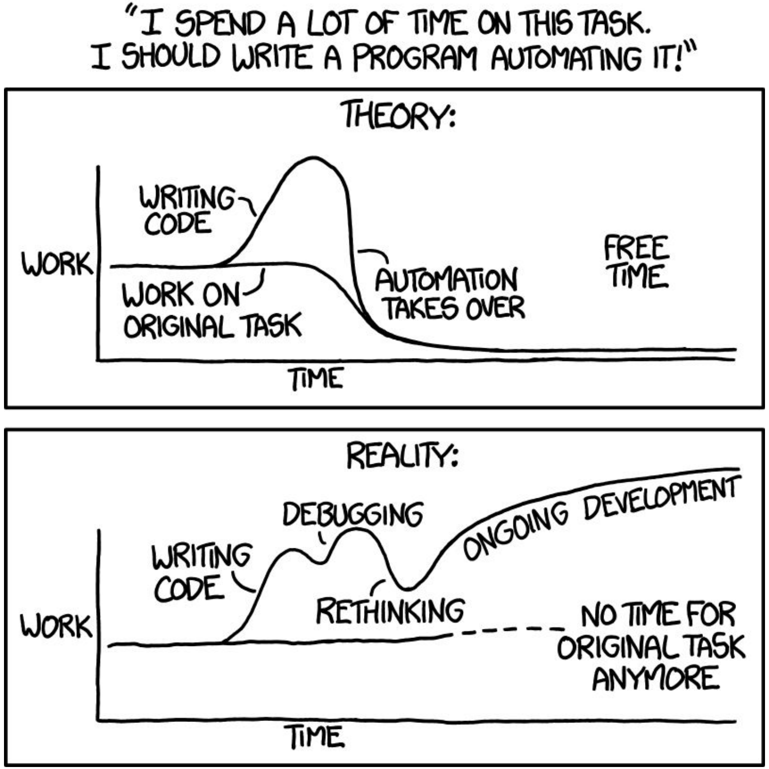
Bart: Isn't that the wrong way?

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

Open Science Grid

Automatic Partitions - Shortcomings



Source: https://xkcd.com/1319/

#3: Share Resources - Requirements

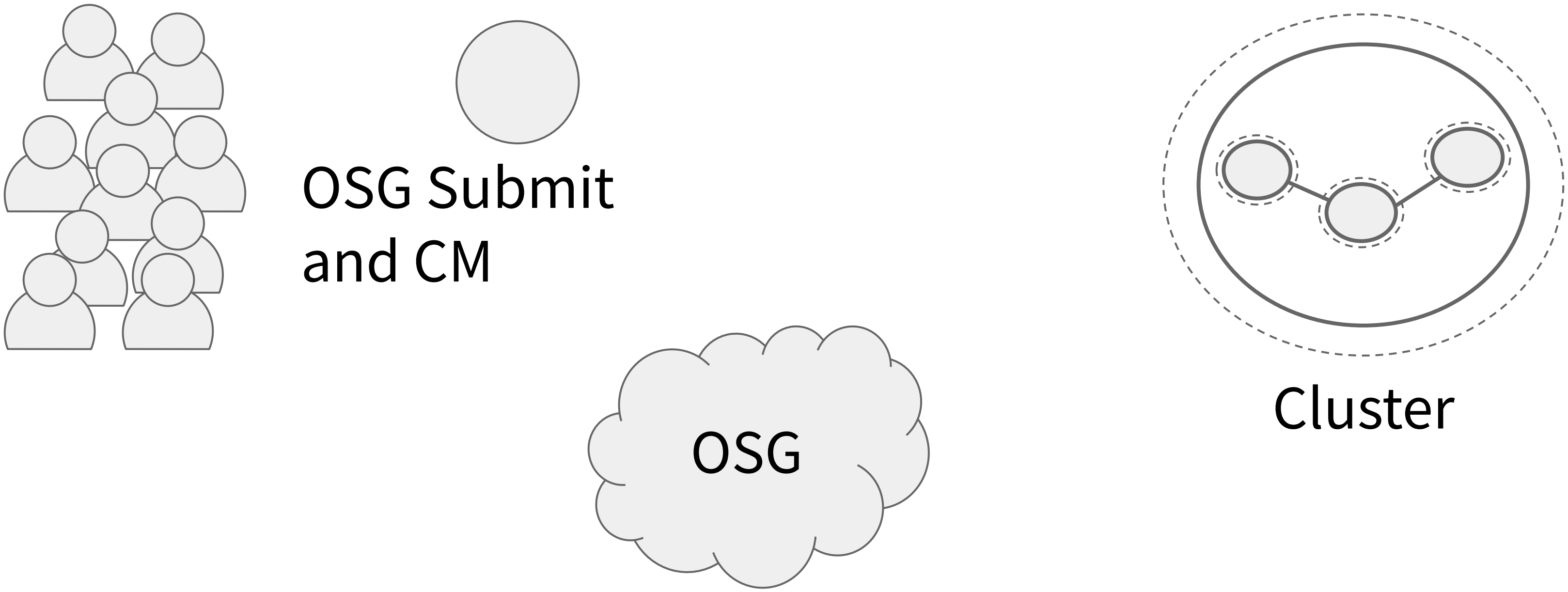
- Minimal account management
- No job splitting
- DAG workflow functionality
- HTCondor only!
- No resource sharing requirements

iii.

Overlay Systems

Let the OSG do the heavy lifting

The OSG Model

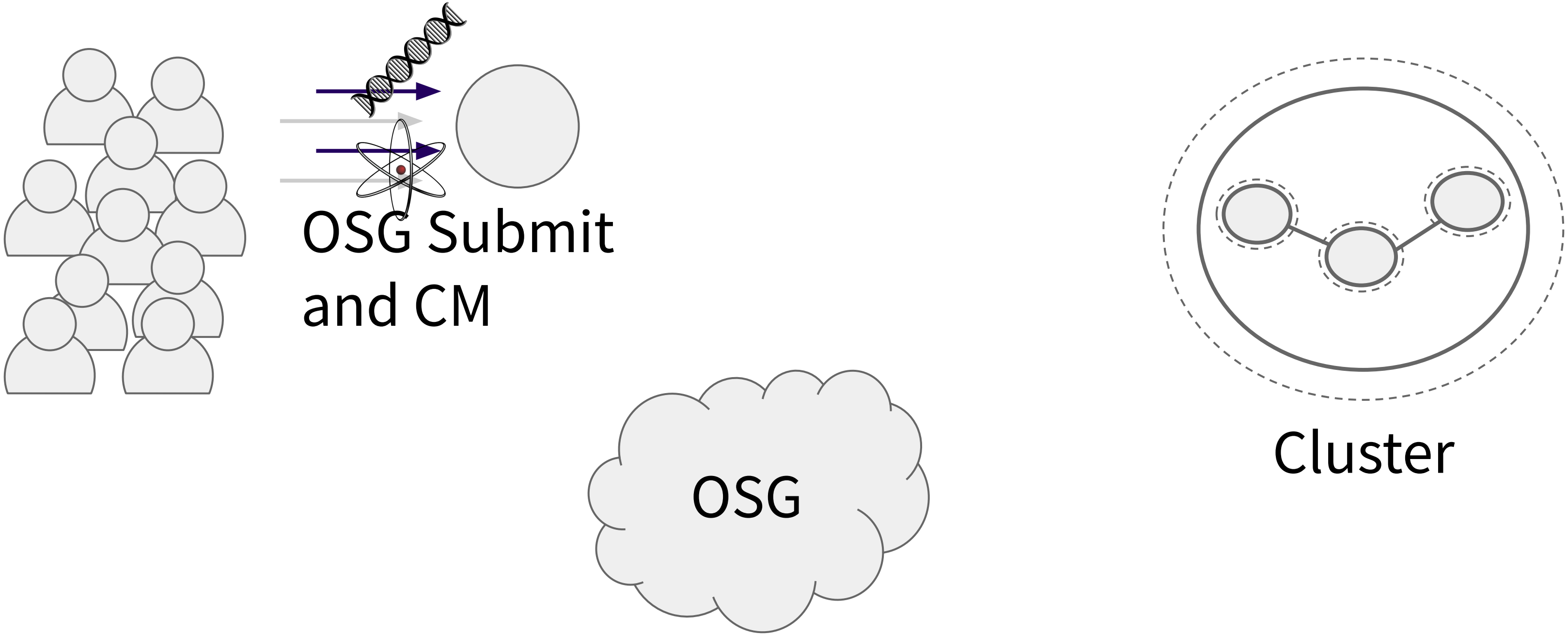


OSG Submit and CM

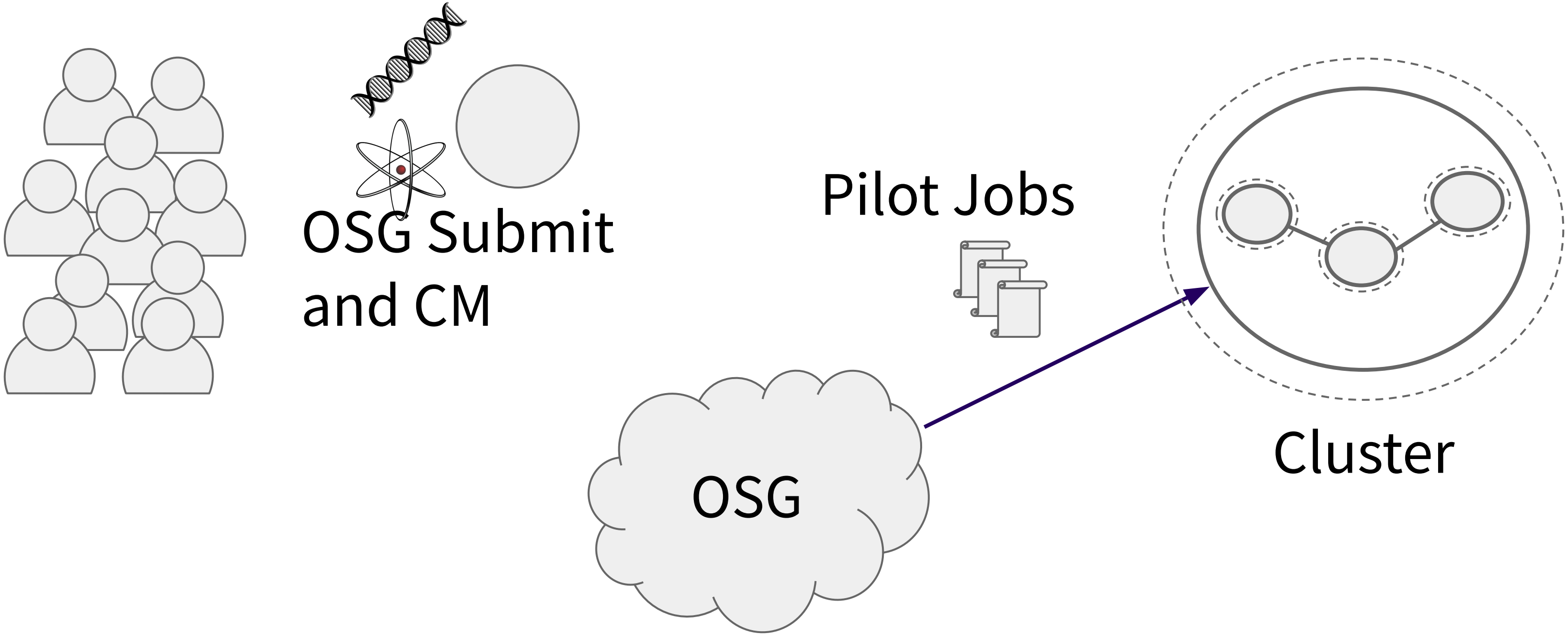
OSG

Cluster

The OSG Model



The OSG Model



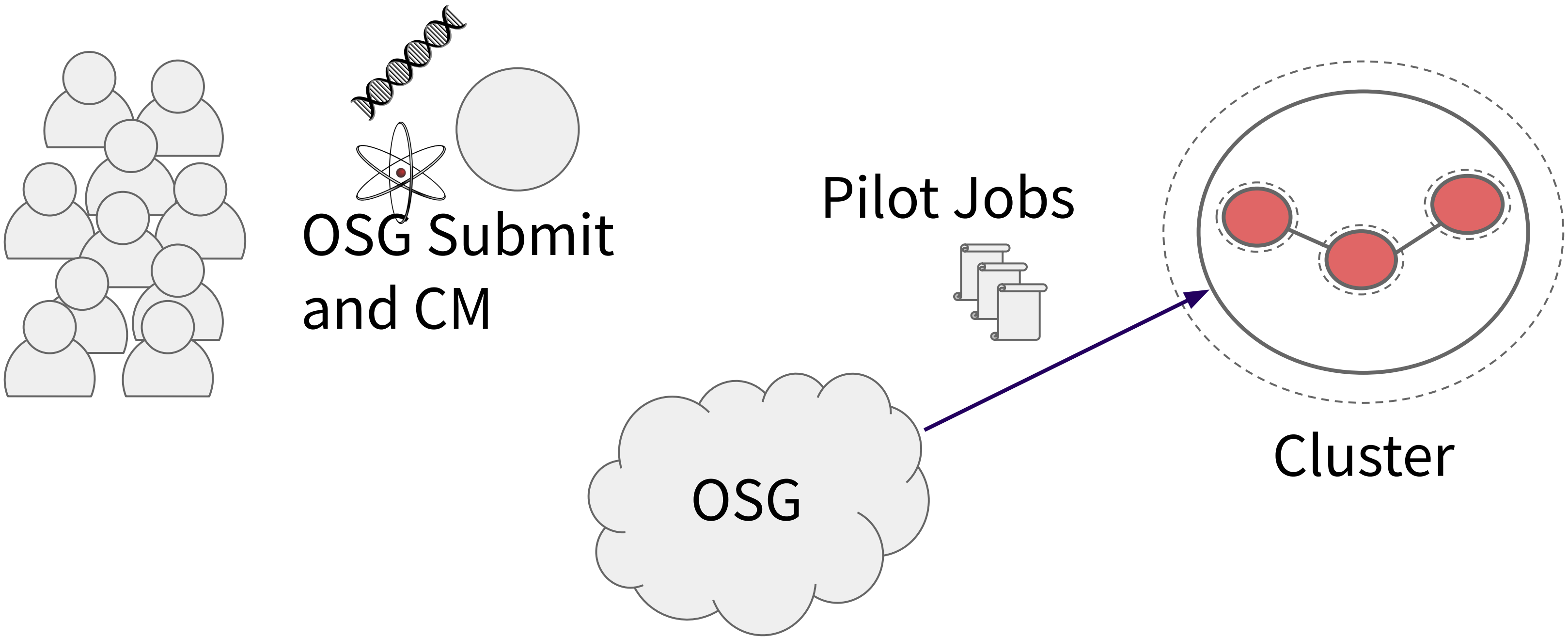
OSG Submit and CM

Pilot Jobs

OSG

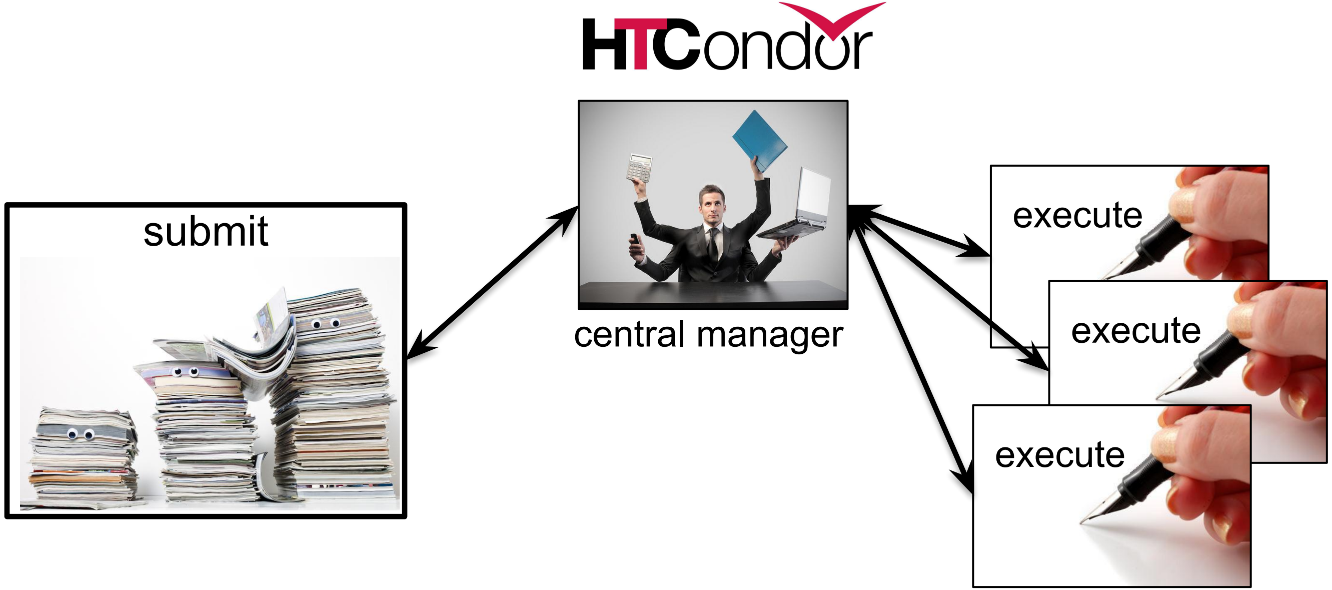
Cluster

The OSG Model

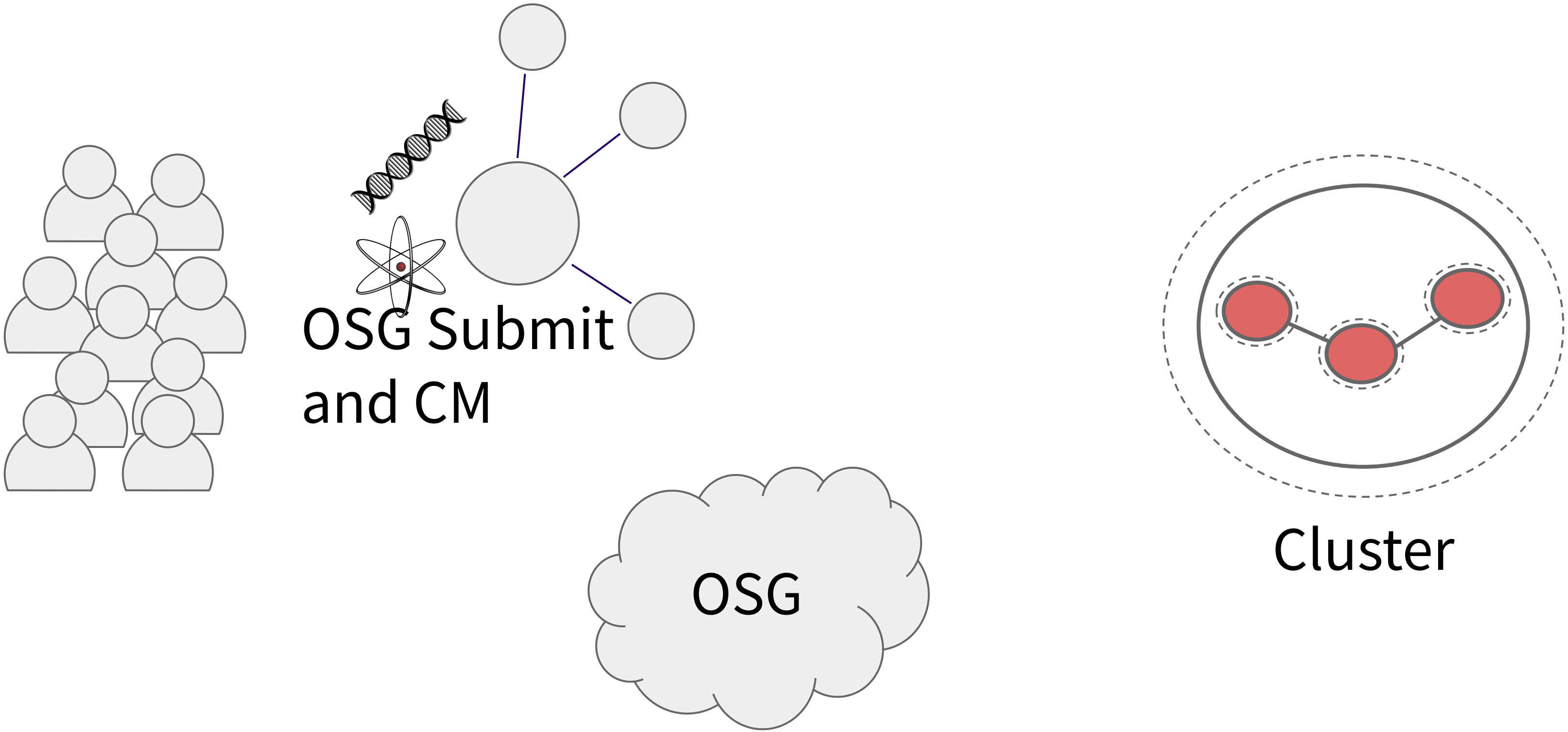


Job Matching

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



The OSG Model



OSG Submit and CM

OSG

Cluster

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)

#3: Share Resources - Requirements

- Minimal account management: only one submit server
- No job splitting: only one HTCondor pool
- DAG workflow functionality: Only one HTCondor pool
- HTCondor only: Only one HTCondor pool
- No resource sharing requirements: the OSG doesn’t require that users “pay into” the OSG

Open Science Grid

The OSG Model - Recap

- Pilot jobs (or pilots) are special jobs
- Pilots are sent to sites with idle resources
- Pilot payload = HTCondor execute node software
- Pilot execute node reports to your OSG pool
- Pilots lease resources:
  - Lease expires after a set amount of time or lack of demand
  - Leases can be revoked!

The OSG Model - Leasing the Cloud

- What if there aren’t enough idle resources?
- Combine overlay system with cloud technology
- Some of your OSG jobs may run in the cloud in the next few years
- … but this should be completely transparent to you

The OSG Model - Collection of Pools

- Your OSG pool is just one of many
- Separate pools for each Virtual Organization (VO)
- Your jobs will run on the OSG VO pool

Photo by Martin Sanchez on Unsplash

The OSG Model - Getting Access

- During the school:
  - OSG submit node at UW (exercises)
  - OSG submit node via OSG Connect (exercises)
- After the school:
  - Both of the above
  - Institution-hosted submit node
  - VO-hosted submit nodes

Questions?

Overlay Systems are Awesome!

Photo by Zachary Nelson on Unsplash

What’s the Catch?

Requires more infrastructure, software, set-up, management, troubleshooting...

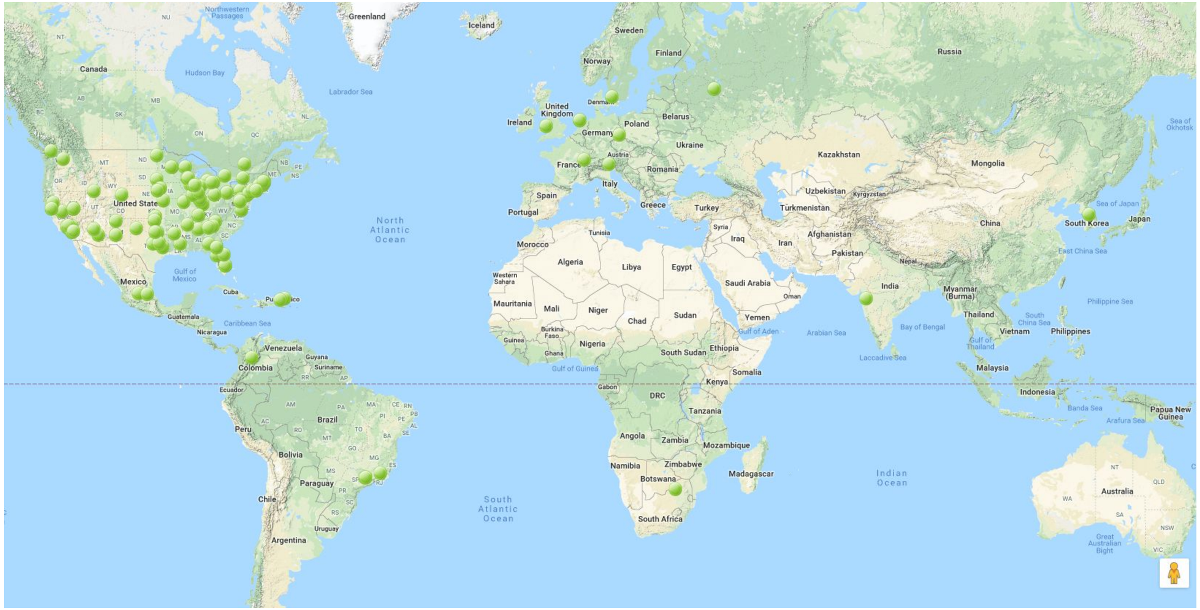
“You know you have a ***distributed system*** *when the crash of a computer you’ve never heard of stops you from getting any work done.”*

- Leslie Lamport

#1: Heterogenous Resources

Accounting for differences between the OSG and your local cluster

Sites of the OSG



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

Heterogeneous Resources - Software

- Different operating systems (Red Hat, CentOS, Scientific Linux; versions 6 and 7)
- Varying software versions (e.g., at least Python 2.6)
- Varying software availability (e.g., no BLAST*)

Solution: Make your jobs more portable: OASIS, containers, etc (more in Wednesday’s talks)

Hetero. Resources - Hardware

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

Solution: 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
- No local system administrator relationships
- No sensitive data (again)!

Photo by Nathan Dumlao on Unsplash

Be a Good Netizen!

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

Solution: Test jobs on local resources with

```
condor_submit -i
```

#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

Open Science Grid

Robustify Your Jobs

Succeeding in the face of failure

Job Robustification

- Test small, test often
- Specify output, error, and log files at least while you develop your workflow
- Use on_exit_hold to catch different failure modes
  - `on_exit_hold = (ExitCode =?= 3)`
  - `on_exit_hold = (time() - JobCurrentStartDate < 1 * $(HOUR))`
- For jobs that run too long:

```
periodic_hold      = (time() - JobCurrentStartDate > 4 * $(HOUR))
periodic_release       = (HoldReasonCode == 3) && (NumJobStarts < 3)
```

`HoldReasonCode` is 3 for any jobs where `on_exit_hold` or `periodic_hold` evaluate to True

Job Robustification

- In your own code:
  - Self checkpointing
  - Different exit codes for use with `on_exit_hold`
  - Defensive troubleshooting (`hostname`, `ls -l`, `pwd`, `condor_version` in your wrapper script)
  - Add simple logging (e.g. print, echo, etc)

Questions?