Open Science Grid

RESEARCH DATA ALLIANCE U.S.

The Landscape of Academic Research Computing

Rob Quick <rquick@iu.edu>
Chief Operations Officer - Open Science Grid
Manager High Throughput Computing

Some Slides Contributed by the University of Wisconsin HTCondor Team and Scot Kronenfeld

INDIANA UNIVERSITY

Open Science Grid

Let's jump right in...

$$
\begin{aligned}
E_n^{(1)} &= V_{nn} \\
E_n^{(2)} &= \frac{|V_{nk_2}|^2}{E_{nk_2}} \\
E_n^{(3)} &= \frac{V_{nk_3}V_{k_3k_2}V_{k_2n}}{E_{nk_2}E_{nk_3}} - V_{nn}\frac{|V_{nk_3}|^2}{E_{nk_3}^2} \\
E_n^{(4)} &= \frac{V_{nk_4}V_{k_4k_3}V_{k_3k_2}V_{k_2n}}{E_{nk_2}E_{nk_3}E_{nk_4}} - \frac{|V_{nk_4}|^2}{E_{nk_4}^2}\frac{|V_{nk_2}|^2}{E_{nk_2}} - V_{nn}\frac{V_{nk_4}V_{k_4k_3}V_{k_3n}}{E_{nk_3}^2E_{nk_4}} - V_{nn}\frac{V_{nk_4}V_{k_4k_2}V_{k_2n}}{E_{nk_2}E_{nk_4}^2} + V_{nn}^2\frac{|V_{nk_4}|^2}{E_{nk_4}^3} \\
&= \frac{V_{nk_4}V_{k_4k_3}V_{k_3k_2}V_{k_2n}}{E_{nk_2}E_{nk_3}E_{nk_4}} - E_n^{(2)}\frac{|V_{nk_4}|^2}{E_{nk_4}^2} - 2V_{nn}\frac{V_{nk_4}V_{k_4k_3}V_{k_3n}}{E_{nk_3}^2E_{nk_4}} + V_{nn}^2\frac{|V_{nk_4}|^2}{E_{nk_4}^3} \\
E_n^{(5)} &= \frac{V_{nk_5}V_{k_5k_4}V_{k_4k_3}V_{k_3k_2}V_{k_2n}}{E_{nk_2}E_{nk_3}E_{nk_4}E_{nk_5}} - \frac{V_{nk_5}V_{k_5k_4}V_{k_4n}}{E_{nk_4}^2E_{nk_5}}\frac{|V_{nk_2}|^2}{E_{nk_2}} - \frac{V_{nk_5}V_{k_5k_2}V_{k_2n}}{E_{nk_2}E_{nk_5}^2}\frac{|V_{nk_2}|^2}{E_{nk_2}} - \frac{|V_{nk_5}|^2}{E_{nk_5}^2}\frac{V_{nk_3}V_{k_3k_2}V_{k_2n}}{E_{nk_2}E_{nk_3}} \\
&\quad - V_{nn}\frac{V_{nk_5}V_{k_5k_4}V_{k_4k_3}V_{k_3n}}{E_{nk_3}^2E_{nk_4}E_{nk_5}} - V_{nn}\frac{V_{nk_5}V_{k_5k_4}V_{k_4k_2}V_{k_2n}}{E_{nk_2}E_{nk_4}^2E_{nk_5}} - V_{nn}\frac{V_{nk_5}V_{k_5k_3}V_{k_3k_2}V_{k_2n}}{E_{nk_2}E_{nk_3}E_{nk_5}^2} + V_{nn}\frac{|V_{nk_5}|^2}{E_{nk_5}^2}\frac{|V_{nk_3}|^2}{E_{nk_3}^2} + 2V_{nn}\frac{|V_{nk_5}|^2}{E_{nk_5}^3}\frac{|V_{nk_2}|^2}{E_{nk_2}} \\
&\quad + V_{nn}^2\frac{V_{nk_5}V_{k_5k_4}V_{k_4n}}{E_{nk_4}^3E_{nk_5}} + V_{nn}^2\frac{V_{nk_5}V_{k_5k_3}V_{k_3n}}{E_{nk_3}^2E_{nk_5}^2} + V_{nn}^2\frac{V_{nk_5}V_{k_5k_2}V_{k_2n}}{E_{nk_2}E_{nk_5}^3} - V_{nn}^3\frac{|V_{nk_5}|^2}{E_{nk_5}^4} \\
&= \frac{V_{nk_5}V_{k_5k_4}V_{k_4k_3}V_{k_3k_2}V_{k_2n}}{E_{nk_2}E_{nk_3}E_{nk_4}E_{nk_5}} - 2E_n^{(2)}\frac{V_{nk_5}V_{k_5k_4}V_{k_4n}}{E_{nk_4}^2E_{nk_5}} - \frac{|V_{nk_5}|^2}{E_{nk_5}^2}\frac{V_{nk_3}V_{k_3k_2}V_{k_2n}}{E_{nk_2}E_{nk_3}} \\
&\quad - 2V_{nn}\left(\frac{V_{nk_5}V_{k_5k_4}V_{k_4k_3}V_{k_3n}}{E_{nk_3}^2E_{nk_4}E_{nk_5}} - \frac{V_{nk_5}V_{k_5k_4}V_{k_4k_2}V_{k_2n}}{E_{nk_2}E_{nk_4}^2E_{nk_5}} + \frac{|V_{nk_5}|^2}{E_{nk_5}^2}\frac{|V_{nk_3}|^2}{E_{nk_3}^2} + 2E_n^{(2)}\frac{|V_{nk_5}|^2}{E_{nk_5}^3}\right) \\
&\quad + V_{nn}^2\left(2\frac{V_{nk_5}V_{k_5k_4}V_{k_4n}}{E_{nk_4}^3E_{nk_5}} + \frac{V_{nk_5}V_{k_5k_3}V_{k_3n}}{E_{nk_3}^2E_{nk_5}^2}\right) - V_{nn}^3\frac{|V_{nk_5}|^2}{E_{nk_5}^4}
\end{aligned}
$$

Who Am I?

- Chief Operations Officer of the Open Science Grid – 10+ years
- Manager High Throughput Computing Indiana University (IU)
- IU Institutional OSG PI
- Co-PI Science Node – (Formally - International Science Grid This Week)
- External Advisor to European Grid Infrastructure
- Member of the Organizational Advisory Board for RDA

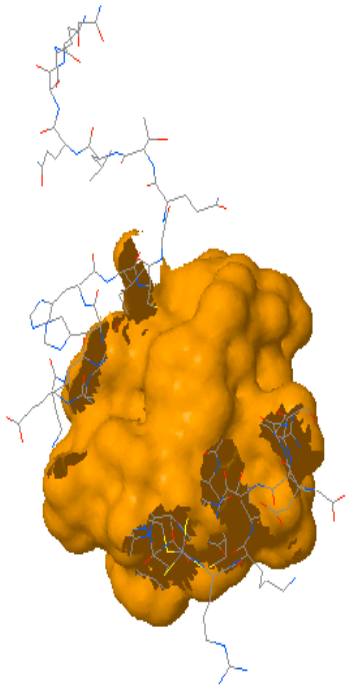
Protein Docking Project at the IU School of Medicine

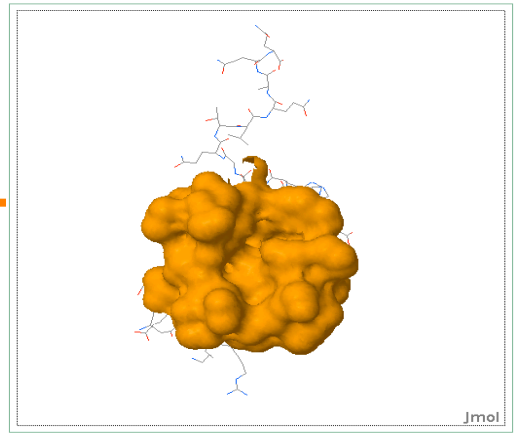
- SPLINTER - Structural Protein-Ligand Interactome
- Used autodock-vina – "…open-source program for drug discovery, molecular docking and virtual screening…"
- Frist run in 2013 - docked ~3900 Proteins with 5000 Ligands for a total of ~19M docked pairs.
- Submitted via command line to Condor using Pegasus on the OSG-XSEDE submission node
- Infrastructure is set and new runs can be easily started

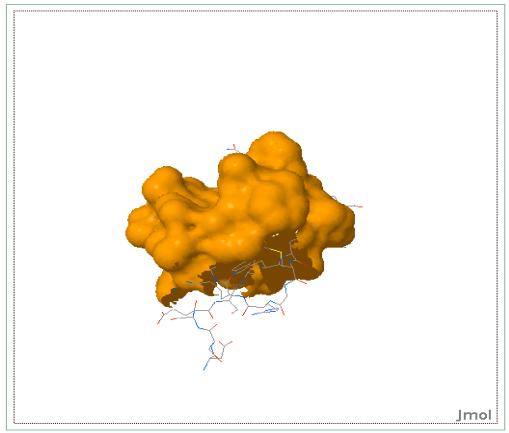
Open Science Grid

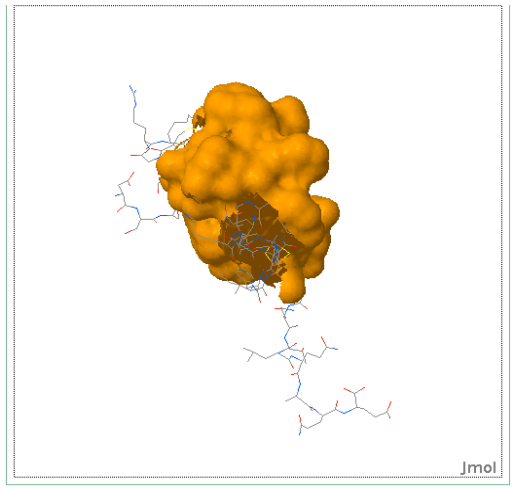
- Various rotations of Protein CBFA2T1 (Cyclin-D-related protein) (Eight twenty one protein) (Protein ETO) (Protein MTG8) (Zinc finger MYND domain-containing protein 2)

Target	Ligand Image	Rank
ZINC27470710 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		1
ZINC01228697 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		2
ZINC04741379 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		3
ZINC09611213 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		4
ZINC02810311 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		5
ZINC02945941 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		6
ZINC00404256 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		7
ZINC20038318 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		8
ZINC04039256 VIEW POSE MOL2 ORDER INFO OTHER TARGETS		9

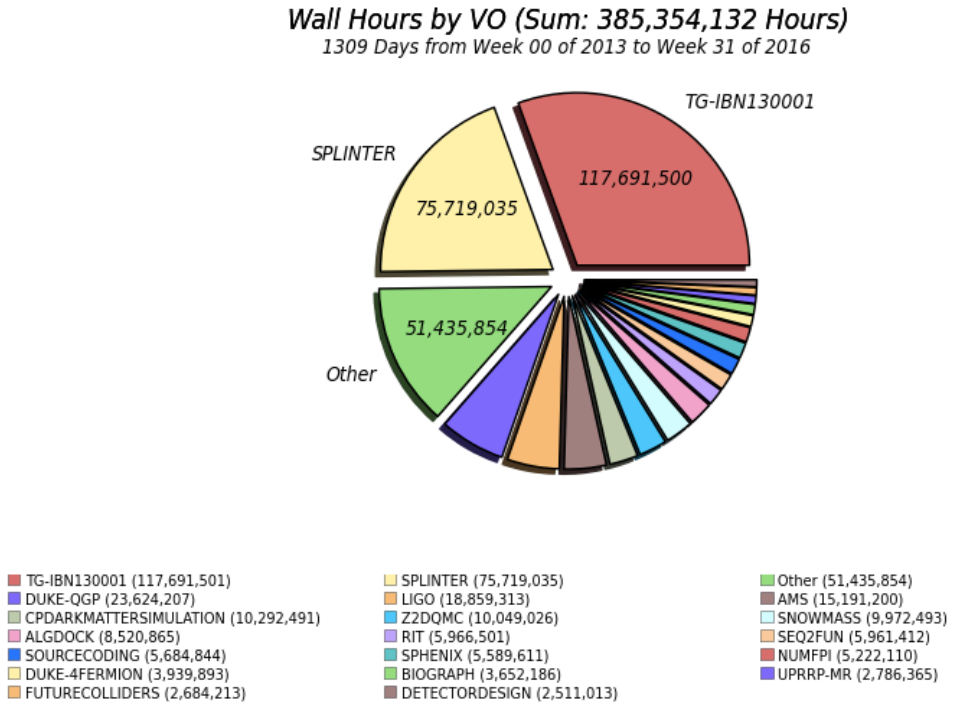








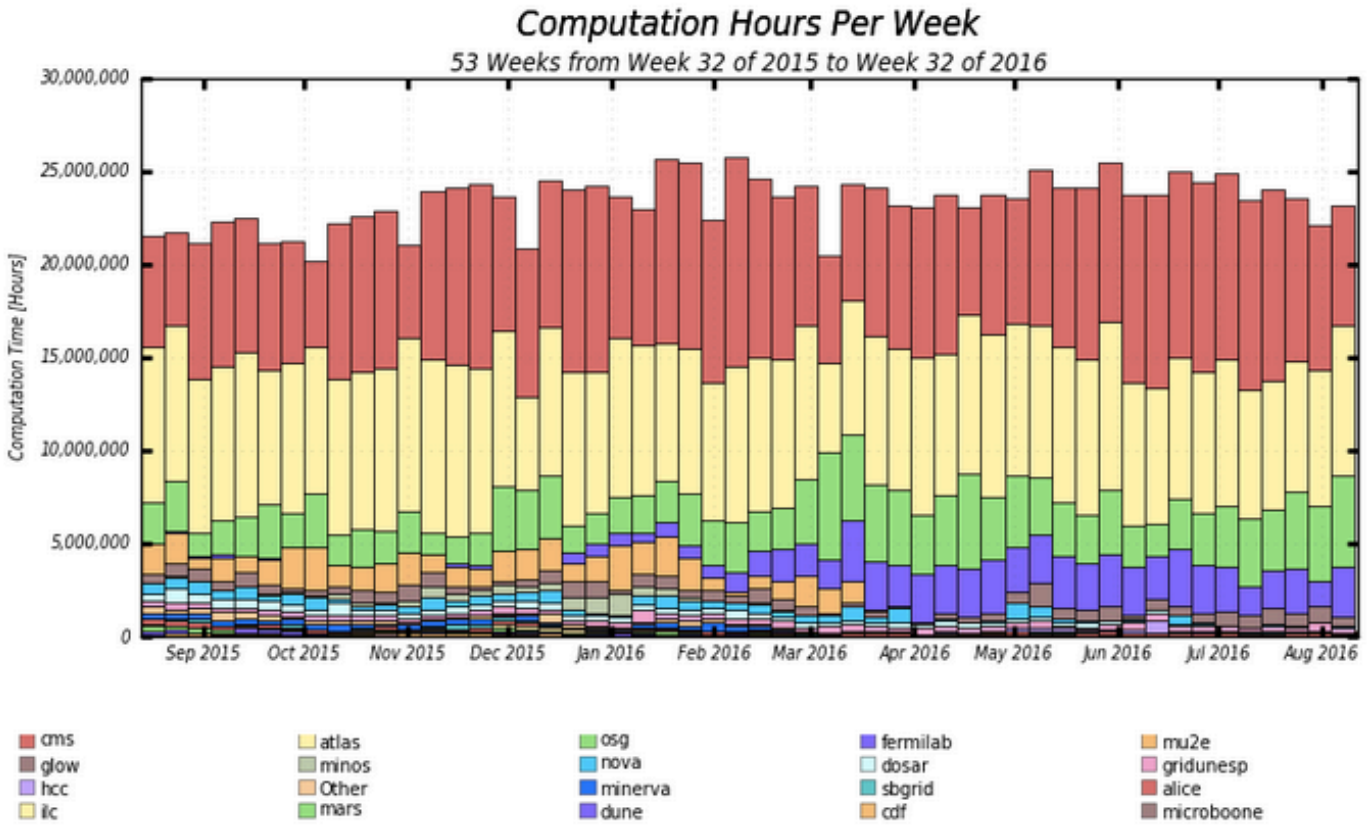
Some Numbers



- Amazon EC2 Computing $0.073/hour
- $5.5M Compute Only
- Data Transfer and Storage Not Included

At what scale?

Job Activity

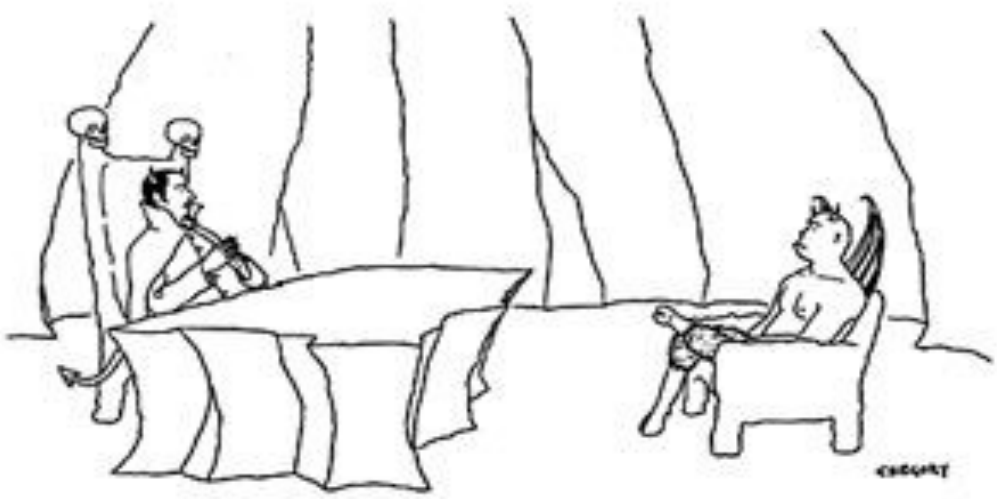


Follow Along at:

https://opensciencegrid.github.io/dosar/Materials/TriesteMaterials/

Overview of day

- Lectures alternating with exercises
  - Emphasis on lots of exercises
  - Hopefully overcome PowerPoint fatigue



"I need someone well versed in the art of torture—do you know PowerPoint?"

Some thoughts on the exercise

- It's okay to move ahead on exercises if you have time
- It's okay to take longer on them if you need to
- If you move along quickly, try the "On Your Own" sections and "Challenges"

Most important!

- Please ask questions!
  - …during the lectures
  - …during the exercises
  - …during the breaks
  - …during the meals
  - …over dinner
  - …via email after we depart (rquick@iu.edu)
- If I don't know, I'll find the right person to answer your question.

Goals for this session

- Define Local, Clustered, High Throughput Computing (HTC), High Performance Computing (HPC), and Cloud Computing (XaaS)
- Shared, Allocated, and Purchased
- What is HTCondor? And why are we using it in this Summer School?

The setup: You have a problem

- Your science computing is complex!
  - Monte carlo, image analysis, genetic algorithm, simulation…
- It will take a year to get the results on your laptop, but the conference is in a week.
- What do you do?

Option 1: Wait a year

Option 2: Local Clustered Computing

- Easy access to additional nodes
- Local support for porting to environment (maybe)
- Often a single type of resource
- Often running at capacity

Option 3: Use a "supercomputer" aka High Performance Computing(HPC)

- "Clearly, I need the best, fastest computer to help me out"
- Maybe you do…
  - Do you have a highly parallel program?
    - i.e. individual modules must communicate
  - Do you require the fastest network/disk/memory?
- Are you willing to:
  - Port your code to a special environment?
  - Request and wait for an allocation?

Option 4: Use lots of commodity computers

- Instead of the fastest computer, lots of individual computers
- May not be fastest network/disk/memory, but you have a lot of them
- Job can be broken down into separate, independent pieces
  - If I give you more computers, you run more jobs
  - You care more about total quantity of results than instantaneous speed of computation
- This is **high-throughput computing**

Option 5: Buy (or Borrow) some computing from a Cloud Provider

- Unlimited resources (if you can afford them)
- Full administrative access to OS of the resources you 'buy'
- Specialized VM images reducing effort in porting
- XaaS Business Model

These are All Valid Options

- Remember the problem you have one month to publish results for your conference
  - Option 1: You will miss your deadline
  - Option 2: You might miss your deadline – But if your lucky you'll make it (or if you know the admin)
  - Option 3: If you have parallelized code and can get an allocation you have a good chance
  - Option 4: If you can serialize your workflow you have a good chance
  - Option 5: You can meet your deadline for a price. Though some efforts are underway to enable academic clouds

Computing Infrastructures

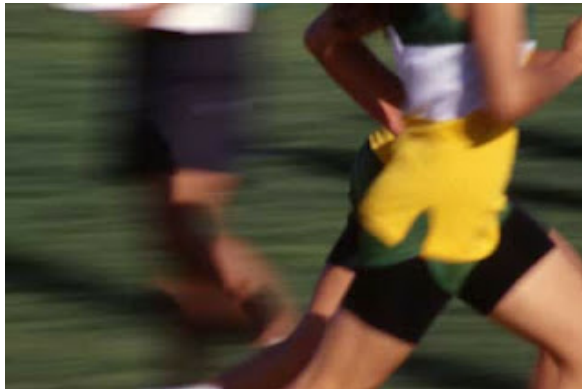
- Local Laptop/Desktop – Short jobs with small data
- Local Cluster – Larger jobs and larger data but subject to availability
- HPC – Prime performance with parallelized code
- HTC – Sustained computing over a long period for serialized
- Cloud – Need deeper permission on an OS and have deeper pockets

Why focus on high-throughput computing? (HTC)

- An approach to distributed computing that focuses on long-term throughput, not instantaneous computing power
  - We don't care about operations per second
  - We care about operations per year
- Implications:
  - Focus on reliability
  - Use all available resources
    - Any Linux based machine can participate

Think about a race

- Assume you can run a four minute mile
- Does that mean you can run a 104 minute marathon?
- The challenges in sustained computation are different than achieving peak in computation speed



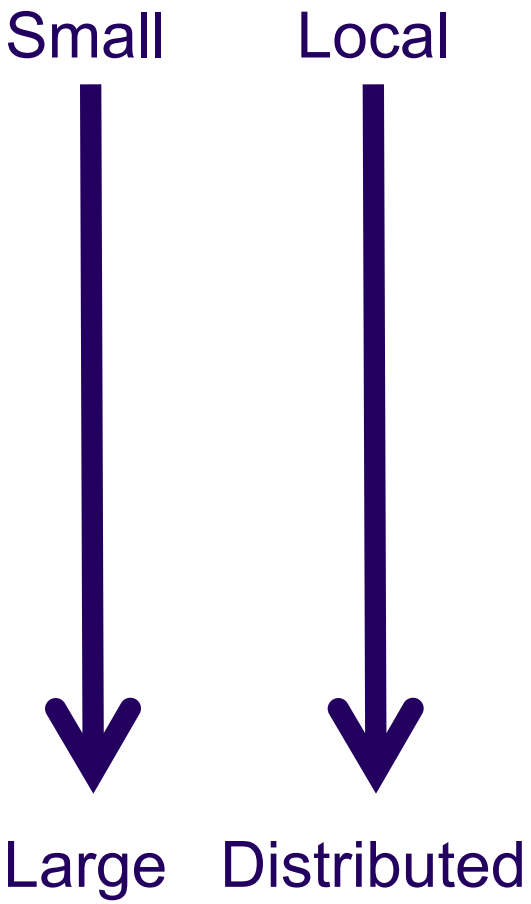
An example problem: BLAST

- A scientist has:
  - Question: Does a protein sequence occur in other organisms?
  - Data: lots of protein sequences from various organisms
  - Parameters: how to search the database.
- More throughput means
  - More protein sequences queried
  - Larger/more protein data bases examined
  - More parameter variation

Why is HTC hard?

- The HTC system has to keep track of:
  - Individual tasks (a.k.a. jobs) & their inputs
  - Computers that are available
- The system has to recover from failures
  - There will be failures! Distributed computers means more chances for failures.
- You have to share computers
    - Sharing can be within an organization, or between orgs
    - So you have to worry about security
    - And you have to worry about policies on how you share
- If you use a lot of computers, you have to handle variety:
  - Different kinds of computers (arch, OS, speed, etc..)
  - Different kinds of storage (access methodology, size, speed, etc…)
  - Different networks interacting (network problems are hard to debug!)

Let's take one step at a time



- Can you run one job on one computer?
- Can you run one job on another computer?
- Can you run 10 jobs on a set of computers?
- Can you run a multiple job workflow?
- How do we put this all together?

This is the path we'll take

Discussion

- For 5 minutes, talk to a neighbor: If you want to run one job in a local environment:
  1) What do you (the user) need to provide so a single job can be run?
  2) What does the system need to provide so your single job can be run?
     - Think of this as a set of processes: what needs happen when the job is given? A "process" could be a computer process, or just an abstract task.

What does the user provide?

- A "headless job"
  - Not interactive/no GUI: how could you interact with 1000 simultaneous jobs?
- A set of input files
- A set of output files
- A set of parameters (command-line arguments)
- Requirements:
  - Ex: My job requires at least 2GB of RAM
  - Ex: My job requires Linux
- Control/Policy:
  - Ex: Send me email when the job is done
  - Ex: Job 2 is more important than Job 1
  - Ex: Kill my job if it runs for more than 6 hours

What does the system provide?

- Methods to:
  - Submit/Cancel job
  - Check on state of job
  - Check on state of available computers
- Processes to:
  - Reliably track set of submitted jobs
  - Reliably track set of available computers
  - Decide which job runs on which computer
  - Manage a single computer
  - Start up a single job

Quick UNIX Refresher Before We Start

- **$** #This symbolizes the prompt.
- ssh username@training.osgconnect.net
- nano, vi, emacs, cat >, etc.
- **which, rpm, ps**, mkdir, cd, **gcc**, ls
- A varitey of condor_* commands

Questions?

- Questions? Comments?
  - Feel free to ask me questions now or later:
    Rob Quick rquick@iu.edu

Exercises start here:

https://opensciencegrid.github.io/dosar/Materials/TriesteMaterials/

Presentations are also available from this URL.