Open Science Grid

RDA
RESEARCH DATA ALLIANCE
U.S.

The Landscape of Academic Research Computing

Rob Quick <rquick@iu.edu>

Associate Director Cyberinfrastructure Integration Research Center

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

INDIANA UNIVERSITY

Who Am I?

- Associate Director Cyberinfrastructure Integration Research Center
- NSF PI – Enhanced Robust Persistent Identification of Data (ERPID) Project
- Co-chair CODATA School of Research Data Science Initiative
- Member of the Technical and Organizational Advisory Boards for RDA
- Member of the WDS ITO Technical Advisory Committee and the EOSC Architecture Working Group
- 12+ Years - Chief Operations Officer of the Open Science Grid and the Software Assurance Marketplace

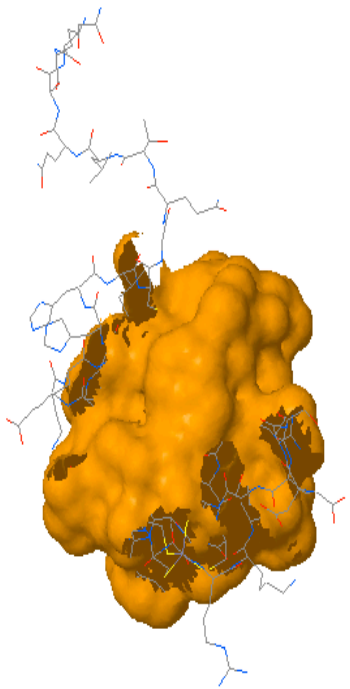
Protein Docking Project at the IU School of Medicine

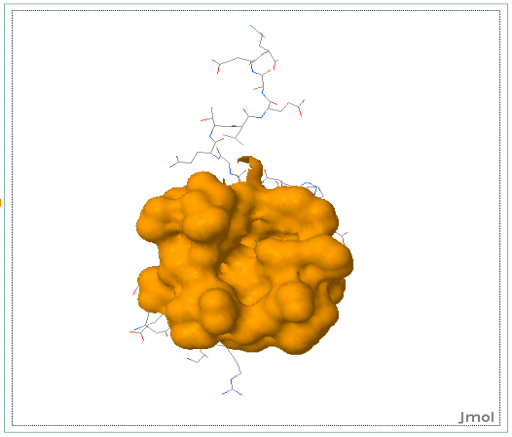
- SPLINTER - Structural Protein-Ligand Interactome
- Used autodock-vina – "…open-source program for drug discovery, molecular docking and virtual screening…"
- Frist run in 2013 - docked ~3900 Proteins with 5000 Ligands for a total of ~19M docked pairs.
- Submitted via command line to Condor using Pegasus on the OSG-XSEDE submission node
- Infrastructure is set and new runs can be easily started
- To date more than 6.3B dockings completed

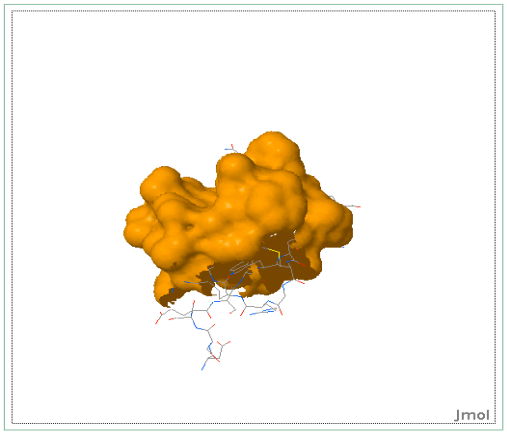
Open Science Grid

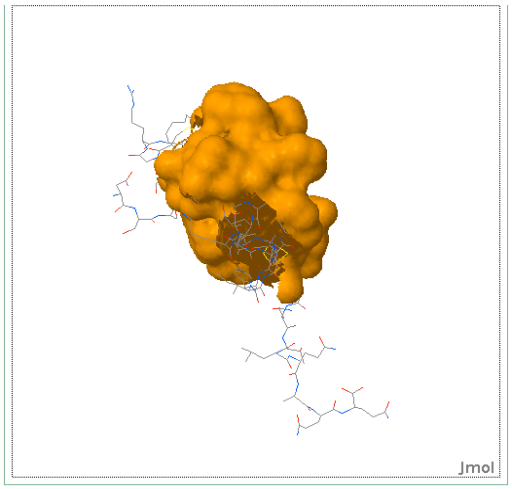
- Various rotations of Protein CBFA2T1 (Cyclin-D-related protein) (Eight twenty one protein) (Protein ETO) (Protein MTG8) (Zinc finger MYND domain-containing protein 2)

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

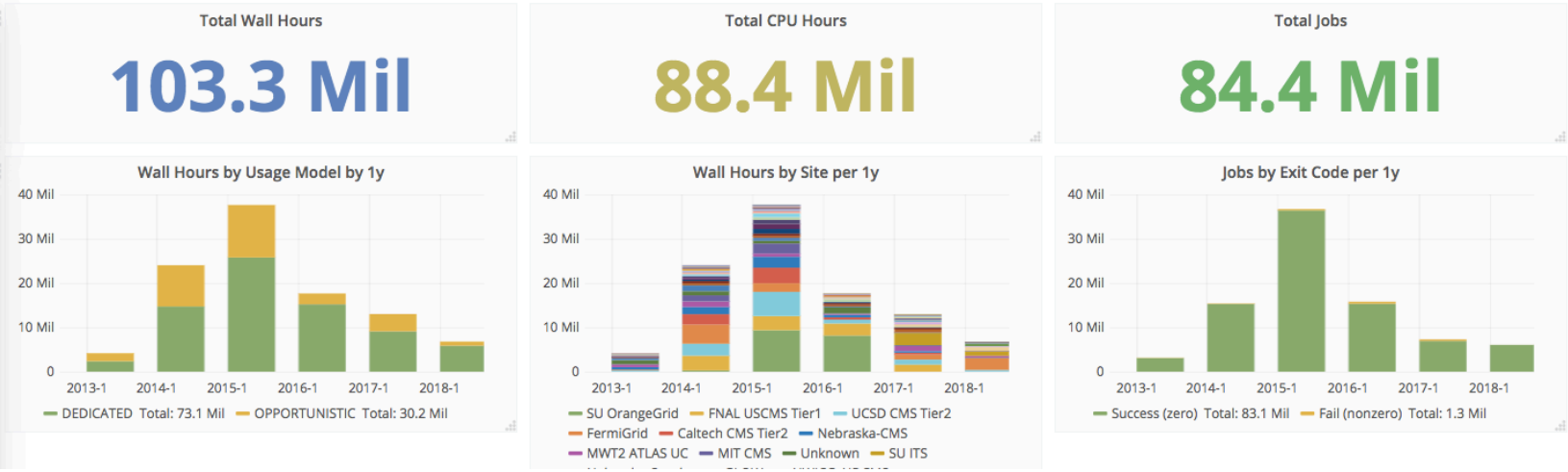








Some Numbers



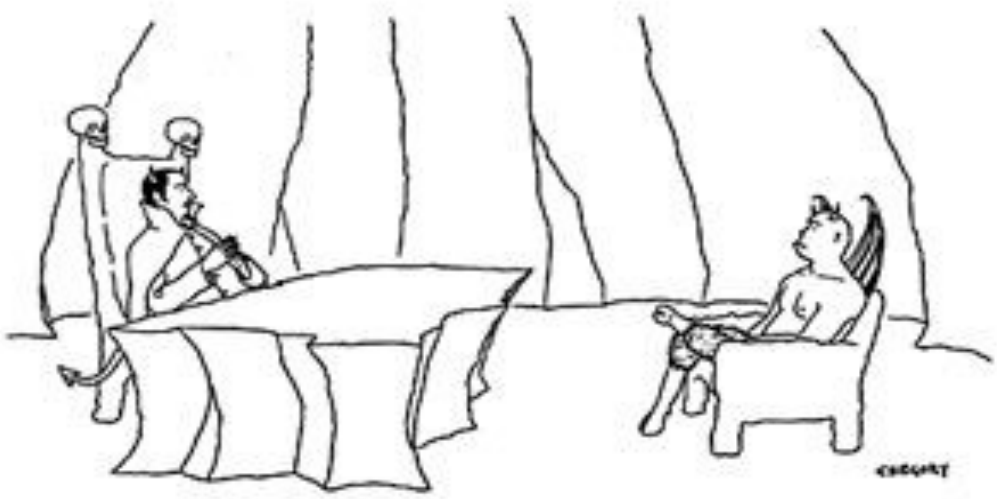
- Amazon EC2 Computing $0.046/hour
- $4.066M Compute Only
- Data Transfer and Storage Not Included

Follow Along at:

https://github.com/CODATA-RDA-DataScienceSchools/Materials/blob/master/docs/DataSanJose2019/CI/Materials.md

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 exercises

- 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), Cloud Computing (XaaS), and Containers
- Shared, Allocated, and Purchased
- What is HTCondor? And why are we learning it?

The setup: You have a problem

- Your science computing is complex!
  - Monte carlo, image analysis, genetic algorithm, simulation, ML or AI algorithms…
- It will take a year (CPU time) to get the results on your laptop, but your paper is due 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 can access 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 week 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 academic clouds are becoming more available.

Open Science Grid

Computing Infrastructures

RDA RESEARCH DATA ALLIANCE U.S.

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

INDIANA UNIVERSITY

Why focus on high-throughput computing? (HTC)

- An approach to distributed computing that focuses on long-term throughput, not instantaneous computing power
- The lessons learned in HTC environments and HTCondor are easily applied to other clustered or HPC systems
- We have access to an international HTC system to show the power of distributed computing

New Technologies

Only that shall happen

Which has happened,

Only that occur

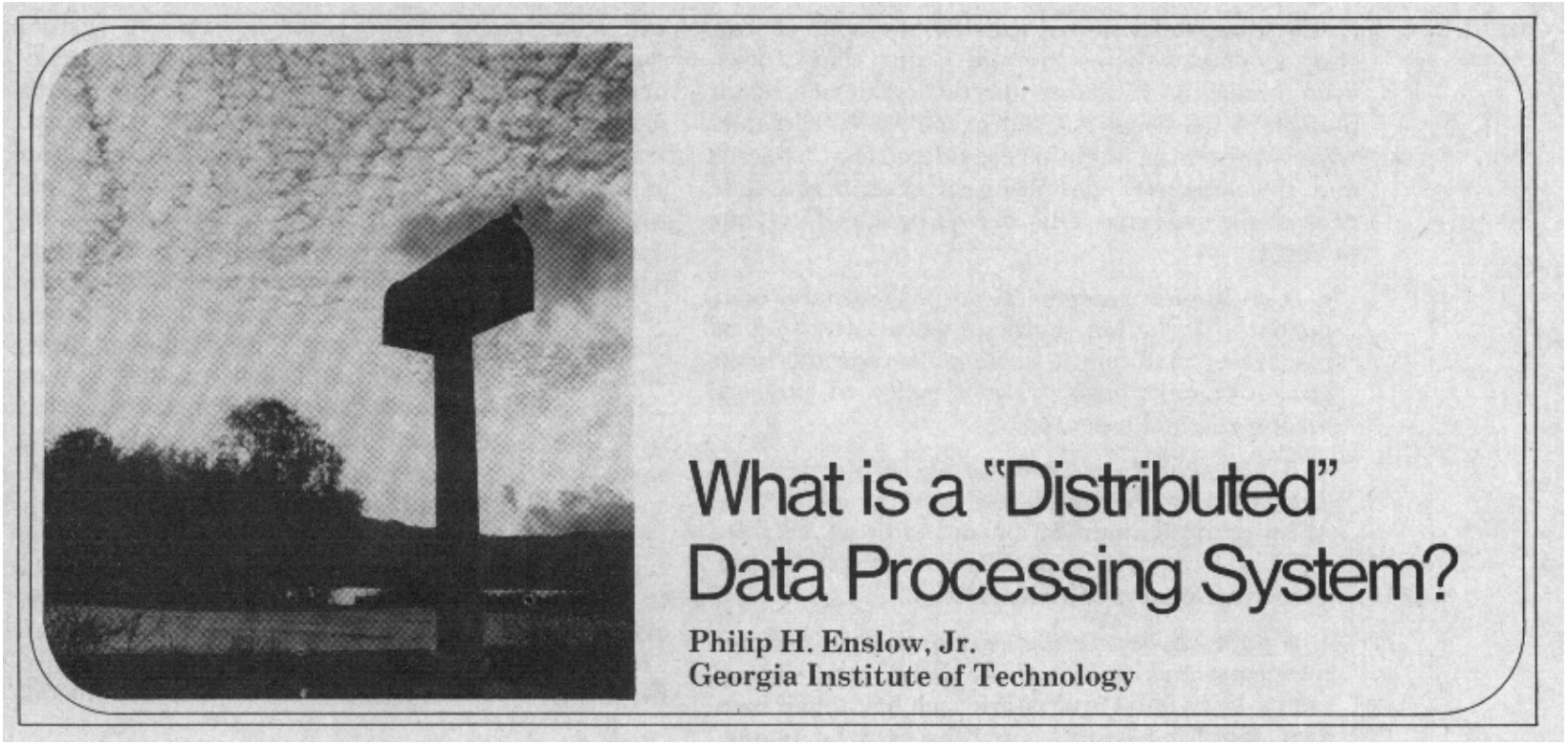
Which has occurred;

There is nothing new

Beneath the sun!

Ecclesiastes Chapter 1 verse 9

Phillip H Enslow



DHTC

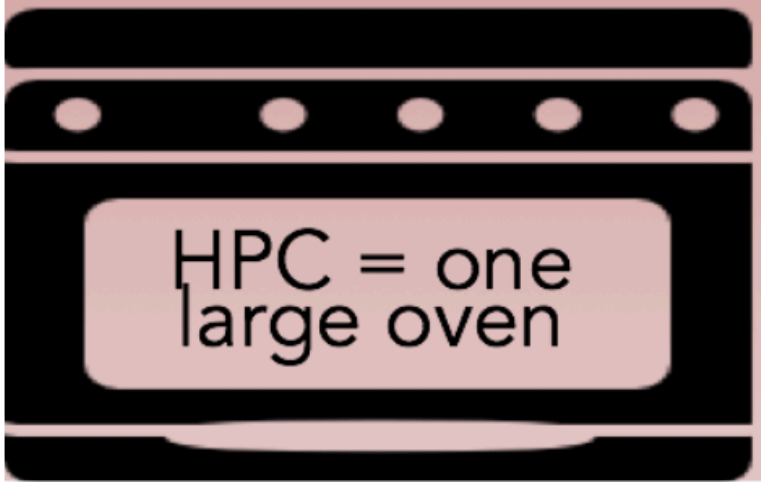
- HTC is about many jobs, many users, many servers, many sites and many workflows.

- What's in a buzzword?
  - "I was doing cloud computing since before it was called grid computing."

Think about a cake

- You've decided you want to bake a cake large enough to break the World's Record for wedding cakes.
- Currently WR 6.818 tonnes (15,032 lb)
- You could approach this two ways…

The HPC Approach



A special oven

Expensive to build & maintain

Scheduling use

The HTC Approach

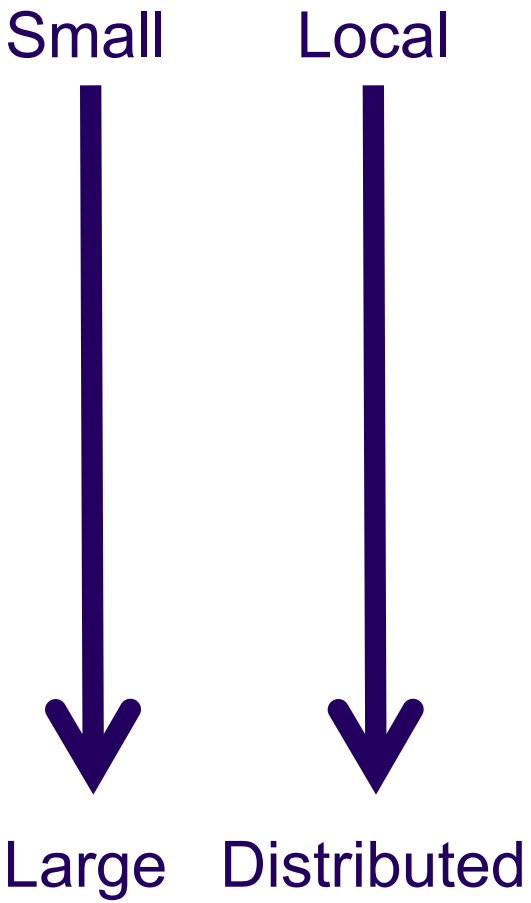
High throughput

Many ovens working independently, outputs assembled afterwards

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 a 100 job parameter sweep 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 scheme of output
- A set of parameters (command-line arguments)
- Requirements:
  - Ex: My job requires at least 2GB of RAM
  - Ex: My job requires 2 CPUs
- 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 resources
- Processes to:
  - Reliably track set of submitted jobs
  - Reliably track set of available resources
  - Decide which job runs on which resource
  - Manage a single computer
  - Start up a set of jobs

Quick UNIX Refresher Before We Start

- **$** - This symbolizes the prompt.
- ssh username@user-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://github.com/CODATA-RDA-DataScienceSchools/Materials/blob/master/docs/DataSanJose2019/CI/Materials.md

Slides are also available from this URL.