Open Science Grid

RDA
RESEARCH DATA ALLIANCE
U.S.

The Landscape of Academic Research Computing

Jaehoon Yu <jaehoonyu1@gmail.com>
Original slides by: Rob Quick <rquick@iu.edu>

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

INDIANA UNIVERSITY
INDIANA UNIVERSITY

Follow Along at:

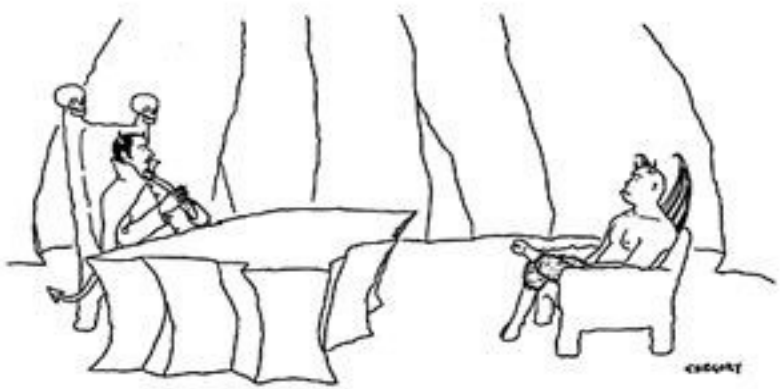
https://osg-htc.org/dosar/ASP2022/ASP2022_Materials/

DOSAR: Distributed Organization for Scientific & Academic Research

2

Overview of day

- Lectures alternating with exercises
  - Emphasis on lots of hands-on exercises
  - Hopefully overcome PowerPoint fatigue
- Note: Power Shedding expected to start at noon ➔ We go for an early lunch and return by 1:45pm.



"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
  - …via email after we depart

  If we don't know the answers, we'll find the right people to answer your questions.

Goals for this lecture

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

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 work-flow 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 Infrastructure

- 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 2-minute km
- Does that mean you can run a 80 minute marathon?
- The challenges in sustained computation are different than achieving peak in computation speed

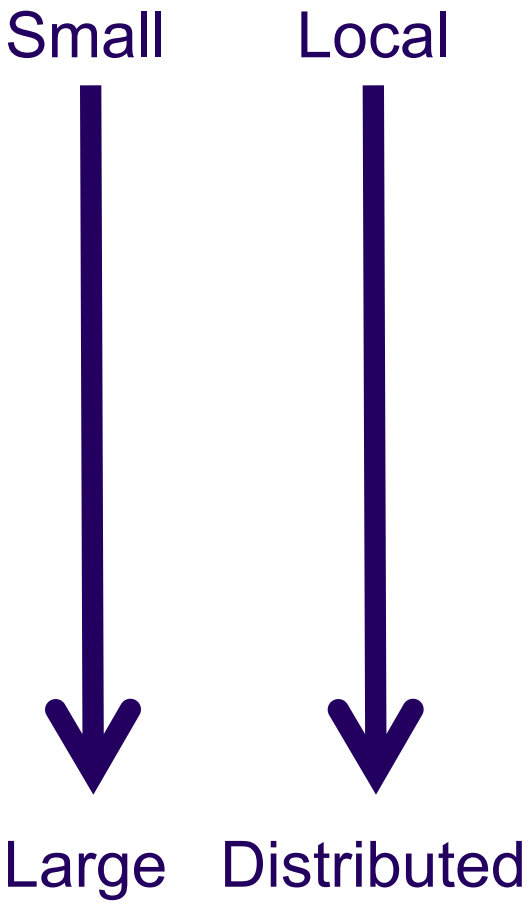
An example problem: BLAST

- 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 a multi-job workflow in a distributed 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 to 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

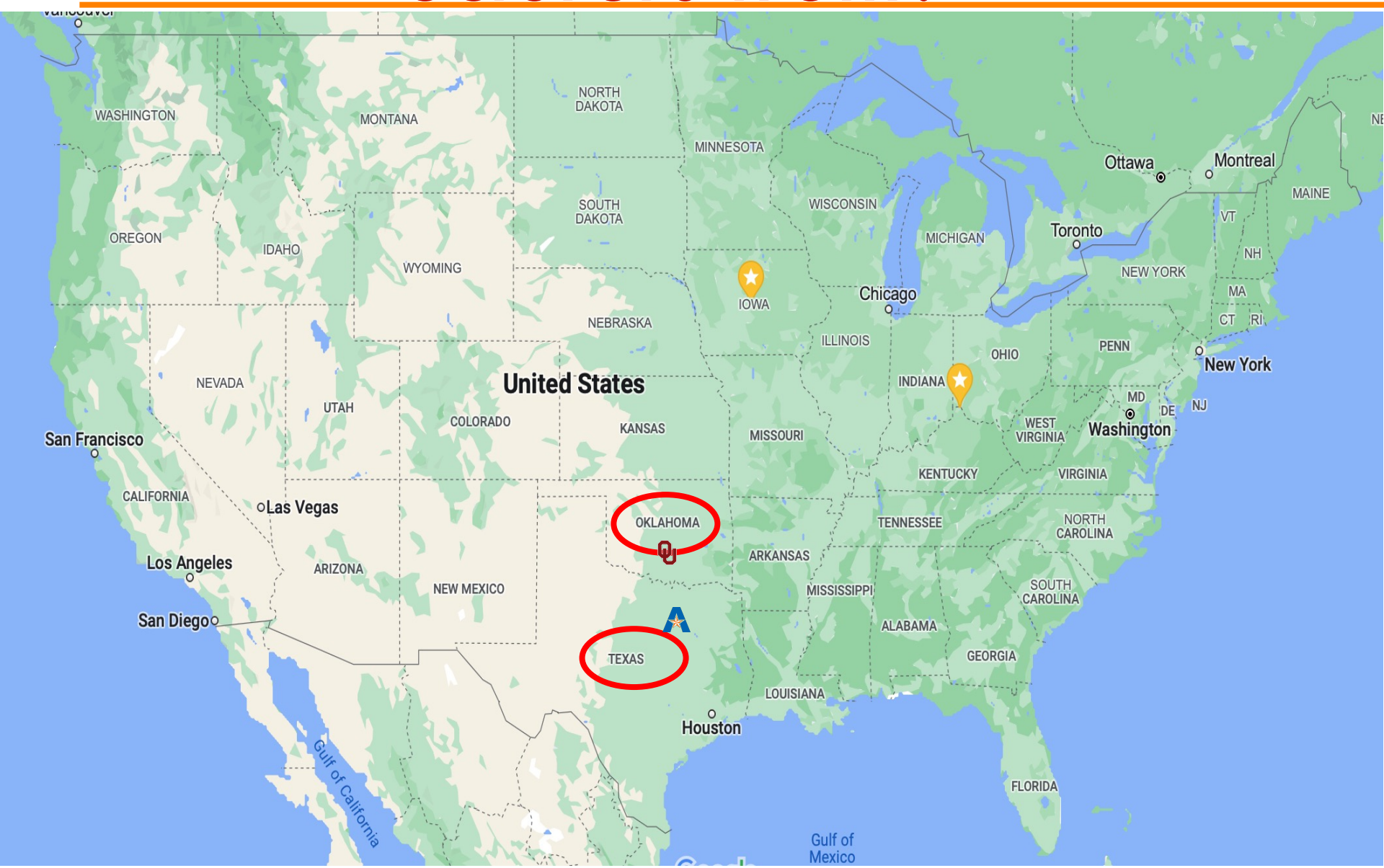
What does the system provide?

- 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.
- nano, vi, emacs, cat >, etc.
- **which, rpm, ps**, mkdir, cd, **gcc**, ls
- A variety of condor_* commands

Where are your lecturers from?



Questions?

- Questions? Comments?
  - Feel free to ask us questions now or later:
  - Jaehoon Yu jaehoonyu1@gmail.com
  - Horst Severini hs@nhn.ou.edu
  - Pat Skubic pskubic@ou.edu

  Exercises start here:

  https://osg-htc.org/dosar/ASP2022/ASP2022_Materials/

  Presentations are also available from this URL.