Introduction to DHTC

Brian Lin
OSG Software Team
University of Wisconsin - Madison
Local High Throughput Computing

UW - Madison

local
compute
resources
Local High Throughput Computing

UW - Madison

local

compute

resources
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 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

- 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
#3: Share Resources - Distributed HTC

University of Nebraska - Lincoln

UW - Madison

University of Chicago
Manual Job Split

- Obtain login access
- Query each cluster for idle resources
- Split 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

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?
• Not all clusters use HTCondor — other job schedulers e.g., Slurm, PBS, etc.
• Querying and splitting jobs is tedious and inaccurate
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!

Automatic Partitions - Shortcomings

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

**Theoretical Scenario:**
- **Writing Code:** peaks initially.
- **Automation Takes Over:** decreases, freeing up time.
- **Time:**

**Practical Scenario:**
- **Writing Code:** remains steady.
- **Debugging:** peaks.
- **Rethinking:** peaks.
- **No Time for Original Task Anymore:**
- **Ongoing Development:**
- **Time:**

Source: [https://xkcd.com/1319/](https://xkcd.com/1319/)
#3: Share Resources - Requirements

- Minimal account management
- No job splitting
- HTCondor only!
- No resource sharing requirements
The OSG Model

OSG Submit and CM

OSG

Cluster
The OSG Model

OSG Submit and CM

OSG

Cluster
The OSG Model

OSG Submit and CM

Pilot Jobs

Cluster
The OSG Model

OSG Submit and CM

Pilot Jobs

Cluster
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

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)
The OSG Model - Details

- 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!
#3: Share Resources - Requirements

- Minimal account management: only one submit server
- No job splitting: only one HTCondor pool
- HTCondor only: pilots report back as HTCondor slots, you’ll be using an HTCondor submit host
- No resource sharing requirements: the OSG doesn’t require that users “pay into” the OSG
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 in the OSG VO pool
The OSG Model - Getting Access

• During the school: learn and training submit host (exercises)
• After the school:
  − learn.chtc.wisc.edu for 1 year!
  − training.osgconnect.net for 1 month!
  − Register for OSG Connect
  − Institution-hosted submit node
  − VO-hosted submit nodes
Quick Break: Questions?
Pilot jobs are awesome!
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 talks later this week)
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!
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
Job Robustification

- Test small, test often
- Specify output, error, and log files at least while you develop your workflow
- In your own code:
  - Self checkpointing
  - Defensive troubleshooting (hostname, ls -l, pwd, condor_version in your wrapper script)
  - Add simple logging (e.g. print, echo, etc)
Hands-On

• Questions?
• Dynamic pool demo!
• Exercises
  – 4.1 - 4.3: Submitting jobs in the OSG
  – 4.4 - 4.5: Identifying differences in the OSG
• Remember, if you don’t finish, that’s ok! You can make up work later or during evenings, if you’d like.