Setup Instructions

Please complete these steps for the June 26th workshop before the lessons start at 1:00 PM:

http://hcc.unl.edu/june-workshop-setup#weekfour

And make sure you can log into Crane. OS-specific login directions here: https://hcc-docs.unl.edu/display/HCCDOC/Quick+Start+Guides

(If you don’t have an HCC account, find a helper to give you a demo account.)

If you need help with the setup, please put a red sticky note at the top of your laptop.

When you are done with the setup, please put a green sticky note at the top of your laptop.
June Workshop Series
June 26th: Anvil – HCC’s Cloud and the Open Science Grid
University of Nebraska – Lincoln
Holland Computing Center

Carrie Brown, Natasha Pavlovikj, Emelie Harstad,
Adam Caprez, Jingchao Zhang, Caughlin Bohn
June Workshop Series Schedule

June 5th: Introductory Bash
Learn how to connect to Holland's High Performance Computing cluster, Crane and how to navigate in the Linux command line with the Bash shell. Topics include the use of pipes, filters, loops, and handling of text files. Lesson materials based on Software Carpentry lessons.

June 12th: Advanced Bash and Git
Continue mastering the Bash shell by learning how to write reusable shell scripts to help automate tasks. We will also be looking at how using version control with Git can help streamline collaboration and manage revisions. Lesson materials based on Software Carpentry lessons.

June 19th: Introductory HCC
Learn the basics of working with Holland resources. Topics include an overview of HCC offerings, submitting batch jobs, handling data and tips on how to optimize workflow.

June 26th: Anvil: HCC’s Cloud and the Open Science Grid
Learn about Anvil, HCC's Open Stack resource which allows researchers greater flexibility by offering cloud-based virtual machines which are ideal for applications that are not well suited to the Linux command line environment. Also, learn about the Open Science Grid (OSG), a nation-wide collection of opportunistic resources that are well suited to large high-throughput applications. Learn what jobs are ideal for OSG and how to begin using this service.
Logistics

• Location: Avery Hall Room 347
• Name tags, sign-in sheet
• Sticky notes: Red = need help, Green = all good
• 15 min break at about 2:30
• Link to June 26th attendee portal (with command history):
  http://hcc.unl.edu/swc-history/index.html
• Link to slides:
  https://hcc-docs.unl.edu/display/HCCDOC/June+Workshop+Series+2018
Introduction to the Open Science Grid

HCC June Workshop Series
June 26, 2018

Emelie Harstad <eharstad@unl.edu>
OSG User Support
UNL Holland Computing Center – Applications Specialist
Outline

• What is the OSG?
• Who uses OSG?
• Owned vs. Opportunistic Use
• Characteristics of OSG-Friendly Jobs
• Is OSG Right for Me?
• Hands-on: How to submit jobs to the OSG from HCC clusters
The Open Science Grid

A **framework for large scale distributed resource sharing** addressing the technology, policy, and social requirements of sharing computing resources.

- The OSG is a **consortium** of software, service and resource providers and researchers, from universities, national laboratories and computing centers across the U.S., who together build and operate the OSG project.
- Funded by the NSF and DOE.

> 50 research communities
> 130 sites
> 100,000 cores accessible
The Open Science Grid

Over 1.6 billion CPU hours per year!!

Status Map  Jobs  CPU Hours  Transfers  TB Transferred

Thousands of Hours/Hour

In the last 24 Hours
- 353,000 Jobs
- 4,655,000 CPU Hours
- 459,000 Transfers
- 126 TB Transferred

In the last 30 Days
- 8,721,000 Jobs
- 140,664,000 CPU Hours
- 116,884,000 Transfers
- 15,404 TB Transferred

In the last 12 Months
- 133,311,000 Jobs
- 1,610,781,000 CPU Hours
- 2,247,837,000 Transfers
- 236,000 TB Transferred

OSG delivered across 123 sites
Who is Using the OSG?

- Astrophysics
- Biochemistry
- Bioinformatics
- Earthquake Engineering
- Genetics
- Gravitational-wave physics
- Mathematics
- Nanotechnology
- Nuclear and particle physics
- Text mining
- And more…
OSG Usage

Wall Hours by VO in past 30 days

VO = Virtual Organization

Most OSG use is on dedicated resources (used by resource owners) – ‘atlas’, ‘cms’

About 13% is opportunistic use – ‘osg’, ‘hcc’, ‘glow’
OSG Jobs

• **High Throughput Computing**
  ➢ Sustained computing over long periods of time. Usually serial codes, or low number of cores threaded/MPI.

**vs. High Performance Computing**
  ➢ Great performance over relative short periods of time. Large scale MPI.

• **Distributed HTC**
  ➢ No shared file system
  ➢ Users ship input files and (some) software packages with their jobs.

• **Opportunistic Use**
  ➢ Applications (esp. with long run times) can be *preempted* (or killed) by resource owner’s jobs.
  ➢ Applications should be relatively short or support being restarted.
OSG Jobs

• **High Throughput Computing**
  - Sustained computing over long periods of time. Usually serial codes, or low number of cores threaded/MPI.

  **vs. High Performance Computing**
  - Great performance over relative short periods of time. Large scale MPI.

• **Distributed HTC**
  - No shared file system
  - Users ship input files and (some) software packages with their jobs.

• **Opportunistic Use**
  - Applications (esp. with long run times) can be *preempted* (or killed) by resource owner’s jobs.
  - Applications should be relatively short or support being restarted.
OSG Jobs

• High Throughput Computing
  ➢ Sustained computing over long periods of time. Usually serial codes, or low number of cores treaded/MPI.

vs. High Performance Computing
  ➢ Great performance over relative short periods of time. Large scale MPI.

• Distributed HTC
  ➢ No shared file system
  ➢ Users ship input files and (some) software packages with their jobs.

• Opportunistic Use
  ➢ Applications (esp. with long run times) can be preempted (or killed) by resource owner’s jobs.
  ➢ Applications should be relatively short or support being restarted.
OSG Jobs

- Run-time: 1-12 hours
- Single-core
- Require <2 GB Ram
- Statically compiled executables (transferred with jobs)
- Non-proprietary software
OSG Jobs

- Run-time: 1-12 hours
- Single-core
- Require <2 GB Ram
- Statically compiled executables (transferred with jobs)
- Non-proprietary software

These are not hard limits!
- Checkpointing – for long jobs that are preempted
  - Many applications support built-in checkpointing
  - Job image is saved periodically so that it can be restarted on a new host after it is killed (without losing the progress that was made on the first host)
- Limited support for larger memory jobs
- “Partitionable” slots – for parallel applications using up to 8 cores
- Modules available – a collection of pre-installed software packages
- Can run compiled Matlab executables
Is OSG right for me?

Are your jobs OSG-friendly?

- **Yes**: Would you like to have access to more computing resources?
  - **Yes**: Consider submitting your jobs to OSG
  - **No**: Continue submitting to HCC clusters

- **No**: You will need to change your submit script slightly (to use HTCondor scheduler). Please contact us for help hcc-support@unl.edu
For more information on the Open Science Grid:
https://www.opensciencegrid.org/

For instructions on submitting jobs to OSG:
https://hcc-docs.unl.edu/display/HCCDOC/The+Open+Science+Grid
Quickstart Exercise

Submit a simple job to OSG from Crane

ssh <username>@crane.unl.edu
cd $WORK
git clone https://github.com/unlhcc/HCCWorkshops.git
cd $WORK/HCCWorkshops/OSG/quickstart

Exercise 1:
osg-template-job.submit (HTCondor submit script)
short.sh (job executable)

Exercise 2:
osg-template-job-input-and-transfer.submit
short_with_input_output_transfer.sh
Quickstart Exercise

**HTCondor Commands:**

- `condor_submit <submit_script>`  # submit a job to osg
- `condor_q <username>`  # monitor your jobs
- `condor_rm <jobID>`  # remove a job
- `condor_rm <username>`  # remove all of your jobs

*Everything you need to know and more about HTCondor submit scripts:*
Scaling Up on OSG

Efficient approach to handle independent jobs

Serial

1 core

job

job

job

job

Time

High Throughput Computing

n cores

job

job

job

job

Time
Scaling Up Exercise

cd $WORK/HCCWorkshops/OSG/ScalingUp-Python

scalingup-python-wrapper.sh    # job executable (wrapper)
rosen_brock_brute_opt.py      # Python script

Example1/ScalingUp-PythonCals.submit  # submit script 1
Example2/ScalingUp-PythonCals.submit  # submit script 2
Example3/ScalingUp-PythonCals.submit  # submit script 3
Example4/ScalingUp-PythonCals.submit  # submit script 4
Python Brute Force Optimization

\[ f = (1 - x)^2 + (y - x^2)^2 \]

2-D Rosenbrock function
Used to test the robustness of an optimization method

Python script
`rosen_brock_brute_opt.py` finds the minimum of the function for a set of points (grid) between selected boundary values.

By default, Python script will randomly select the boundary values of the grid that the optimization procedure will scan over. These values can be overridden by user supplied values.

```
python rosen_brock_brute_opt.py x_low x_high y_low y_high
```
Python Brute Force Optimization
Python Brute Force Optimization

Universe = Vanilla
...
....
arguments = Queue
arguments = Queue
...
arguments = Queue

arguments

1 2 • • • N

Jobs
Python Brute Force Optimization

Universe = Vanilla
....
....
Queue variables from (list)

arguments arguments arguments arguments

1 2 3 \ldots N