Introduction to
the Open Science Grid

HCC Kickstart
October 16, 2018

Emelie Fuchs <efuchs@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!!
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 8% 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 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

- 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?

- no
  - Continue submitting to HCC clusters

- yes
  - Would you like to have access to more computing resources?
    - no
      - You will need to change your submit script slightly (to use HTCondor scheduler).
      - Please contact us for help hcc-support@unl.edu
    - yes
      - Consider submitting your jobs to OSG
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

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

1 2  ... N  Jobs
Python Brute Force Optimization

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

arguments
arguments
arguments
arguments

1 2 3  .  .  .  N
Additional Slide(s)
To switch to OSG modules on Crane:
    source osg_oasis_init

To analyze job matching:
    condor_q -pool glidein.unl.edu -analyze <jobid>

To view hosts where previous jobs have run:
    condor_history -long <username> | grep LastRemoteHost