Introduction to the Open Science Grid

HCC Kickstart
September 6, 2017

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 1.5 billion CPU hours per year!!

<table>
<thead>
<tr>
<th>Status Map</th>
<th>Jobs</th>
<th>CPU Hours</th>
<th>Transfers</th>
<th>TB Transferred</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>24 Hours</td>
<td>30 Days</td>
<td>12 Months</td>
</tr>
</tbody>
</table>

In the last 24 Hours
- 432,000 Jobs
- 4,560,000 CPU Hours
- 6,490,000 Transfers
- 795 TB Transferred

In the last 30 Days
- 13,054,000 Jobs
- 133,215,000 CPU Hours
- 196,212,000 Transfers
- 17,450 TB Transferred

In the last 12 Months
- 136,030,000 Jobs
- 1,519,107,000 CPU Hours
- 2,043,436,000 Transfers
- 176,000 TB Transferred
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

- VO = Virtual Organization

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

- About 20% 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?

- 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

Are your jobs OSG-friendly? no

Continue submitting to HCC clusters

Would you like to have access to more computing resources? no

Consider submitting your jobs to OSG

Would you like to have access to more computing resources? yes

Consider submitting your jobs to OSG

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

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:**

```plaintext
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

High Throughput Computing
n cores

Time

job

job

job

job
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
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
Python Brute Force Optimization

Universe = Vanilla

....

....

Queue variables from (list)