OpenSees OSG Integration Details

Marko Slyz

11 February 2011

Contents

(Up)

1 Introduction

OSG consistes of many *sites*, each of which is a network of computers administered by a single organization. The computers at any particular site typically have the same software configuration, and have access to a shared file system for the site.

We are going to be running jobs using the Condor distributed computing system. More specifically, in this case Condor will actually be running on top of a system called glideinWMS, which makes it easier to run at many sites, but that is mostly transparent to us.

Background reading: UsingTheGrid.

2 Get a proxy certificate

A proxy certificate is necessary to have permission to run on OSG. The commands to get one are:

```
. /opt/osg/osg-1.2/setup.sh
```

This command sets up the environment so that condor and its associated commands (including the ones to get a proxy) work.

```
voms-proxy-init -valid 72:00 --voms Engage:/Engage/NEES
```

This gets a proxy certificate good for 3 days good for running as the Engage VO in the NEES group. Can add the "-debug" option if there are problems.

```
voms-proxy-info --all
```

This lists some information about the new proxy.

3 Create a Job Submission file

3.1 Manual Approach

There is a kind of template file, *opensеesA.csf*, included in the distribution. The three variables that need to change from run to run are at the top of the file:

3.1.1 opensees_input_file

This has to be in gzipped tar format. At the top level there should be an OpenSees executable, and a single directory that contains the OpenSees input files. Both the executable and the directory can have any valid name, but there must be only one of each.

3.1.2 opensees_output_directory

This is a directory on the submit host where the output from all of the runs from a particular condor_submit command (see below) should go. It just needs to be different from any other directory names where output goes.

3.1.3 opensees_num_runs

This is the number of separate runs that gmMP.tcl will do. Typically it's the number of lines in the SAClaRecords.txt file, but could be less.

3.2 (Mostly) Automatic Approach

There is a script called setup_opensees.pl that partially automates this process. Typing "setup_opensees.pl -h" will print out its usage, which currently says

```
             ./setup_opensees.pl
This script modifies a template submit file to have
the proper parameters for running opensees, and also
creates the tar file and a directory for the output.
  This script should be run from the directory where
the submit will occur. All arguments are required
except -a.

-c [filename]
    Input condor submit file to be used as a template
    to run the job.  This can be an actual working
    submit file, but needs to have the
    opensees_input_file, opensees_output_directory,
    and opensees_num_runs macros at the top.

-o [filename]
    Name of condor submit file that will
    incorporate the changes.

-i [name of directory]
   Name of directory that contains the input
   data for OpenSees, like exampleTeraGridCode/.

-e [filename]
```

```
   Path to the OpenSees executable.

-t [filename]
   Name of tar file to later be used as input
   to runOpenSeesA.sh.

-d [name of directory]
    Directory on the submit host to store output. A
    relative path is ok.

-a [filename]
   Name of the dag file to produce. If the -a
   argument is not present then this script produces
   a regular condor submit file whose name is
   indicated by -o.
```

An example of this command is:

```
./setup_opensees.pl -c openseesA.csf -i exampleTeraGridCodeZZZ \
  -e ./OpenSees  -o test4.csf -t opensees_distA2.tgz -d test4_C \
  -a test4.dag
```

Here the inputs (files or directories that must already exist) are

openseesA.csf is a template submission file,

exampleTeraGridCodeZZZ has the input data for OpenSees,

OpenSees is the executable to run on the worker node,

and the outputs (files or directories that the script creates) are:

test4L.csf will be the resulting submission file with the fields filled in,

opensees_ dist2A.tgz will have the OpenSees executable and input data in the form (a gzipped tar file) that we send it to the worker node,

test4_ C is the name of the directory where the the output should go.

test4.dag is the name of a DAG input file to create.

The script will also print a suggested command to run the job, in this case:

```
condor_submit_dag -usedagdir test4_C/test4.dag\\
```

or without the -a option the command would be

```
condor_submit test4L.csf
```

Further reading: http://www.cs.wisc.edu/condor/manual/v7.5/condor_submit.html
For the -a option: http://www.cs.wisc.edu/condor/manual/v7.5/condor_submit_dag.html
http://www.cs.wisc.edu/condor/manual/v7.5/2_10DAGMan_Applications.html#sec:DAGLotsaJobs

4 While the Job is Running

The status of the jobs while they are running can be checked with a command like

```
condor_q [username]
```

The status of the jobs can also be monitored by checking the log file(s) that condor creates in the submit directory.
It is possible to stop the jobs before they're done by issuing a command like

```
condor_rm [cluster #]
```

where the cluster number is printed by condor_submit, or is listed as the number before the dot by condor_q.

5 Automated Tests

There is a script called test_opensees.pl that will try running a couple OpenSees test cases. It expects the directory exampleTeraGridCodeZZZ and the executable OpenSees to be in the same directory from which it is run. It may be run as follows:

```
./test_opensees.pl
```

and doesn't need any arguments.

6 A More Realistic Test Case

1. Suppose the directory with the input data is called `exampleTeraGridCode_long`. Make sure to to modify userDir in gmMP.tcl to say

    ```
    set userDir "[pwd]/exampleTeraGridCode_long"
    ```

    (This is specific to this user's OpenSees setup.)

2. Run this command to set things up:

    ```
    ./setup_opensees.pl -c openseesA.csf -o test_fnpB.csf \
      -t opensees_distA2_fnpB.tgz -i exampleTeraGridCode_rl \
      -e ./OpenSees -d test_fnpB -a test_fnpB.dag
    ```

3. If necessary, fix the dag file to have the runs you'd like.

    ```
    cd test_fnpB
    nano test_fnpB.dag
    ```

4. Submit job using the command from setup_opensees.pl:

    ```
    condor_submit_dag -usedagdir test_fnpB/test_fnpB.dag
    ```

5. Check its progress with

    ```
    condor_q [username]
    ```

7 Appendix: Untarring multiple output files at once

Here is an example with comments that uses a for loop to untar several output files at once. (The setup_opensees.pl command line here was edited to split it into several lines so it fits.)

```
# Set up a quick run:

glidein:opensees$ ./setup_opensees.pl -c openseesA.csf -o test4.csf \
  -t opensees_distA2.tgz -i exampleTeraGridCodeZZZ -e ./OpenSees \
```

```
  -d test4_C $extra_params
Using SAClaRecords_test.txt as the SACla file.
num_runs: 2
Suggested command:  condor_submit test4.csf
glidein:opensees$

# Start jobs going:

glidein:opensees$ condor_submit test4.csf
Submitting job(s)..
2 job(s) submitted to cluster 2471.
glidein:opensees$

# ... Wait for them to finish.

glidein:opensees$ condor_q mslyz


-- Submitter: glidein.unl.edu : <129.93.239.145:49510> : glidein.unl.edu
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 idle, 0 running, 0 held


glidein:opensees$ cd test4_C

# Use a bash for loop to list all the output files:

glidein:test4_C$ for x in OpenSees_output_*.tgz; do echo $x; done
OpenSees_output_0.tgz
OpenSees_output_1.tgz

# Now add the untar command to the for loop:

glidein:test4_C$ for x in OpenSees_output_*.tgz; do echo $x; tar -x -f $x; done
OpenSees_output_0.tgz
OpenSees_output_1.tgz


# See that the la29GM and la28GM directories were created:
```

```
glidein:test4_C$ ls -lt
total 1924
-rw-r--r-- 1 mslyz bockelman   1324 Feb 10 11:35 openseesA_2471.log
-rw-r--r-- 1 mslyz bockelman  24585 Feb 10 11:35 debug_output_for_1.txt
-rw-r--r-- 1 mslyz bockelman     60 Feb 10 11:35 openseesA_stderr.2471.1.txt
-rw-r--r-- 1 mslyz bockelman   5874 Feb 10 11:35 openseesA_stdout.2471.1.txt
-rw-r--r-- 1 mslyz bockelman 843226 Feb 10 11:35 OpenSees_output_1.tgz
-rw-r--r-- 1 mslyz bockelman  29230 Feb 10 11:35 debug_output_for_0.txt
-rw-r--r-- 1 mslyz bockelman     60 Feb 10 11:35 openseesA_stderr.2471.0.txt
-rw-r--r-- 1 mslyz bockelman   5835 Feb 10 11:35 openseesA_stdout.2471.0.txt
-rw-r--r-- 1 mslyz bockelman 973440 Feb 10 11:35 OpenSees_output_0.tgz
drwxr-xr-x 3 mslyz bockelman   4096 Feb 10 11:35 la29GM
drwxr-xr-x 3 mslyz bockelman   4096 Feb 10 11:35 la28GM


# Another variant:

# Clean up the just-produced output:

glidein:test4_C$
glidein:test4_C$
glidein:test4_C$ rm -r la*GM


# Put the numbers of the output files that you'd like to
# untar in a list in braces (called "brace expansion"):

glidein:test4_C$
glidein:test4_C$ for x in OpenSees_output_{0,1}.tgz; do echo $x; tar -x -f $x; done
OpenSees_output_0.tgz
OpenSees_output_1.tgz

# In this case using the * is less typing, but brace expansion is
# useful if you'd like to untar more than one output file but not all
# of them.

glidein:test4_C$ ls -lt
total 1924
-rw-r--r-- 1 mslyz bockelman   1324 Feb 10 11:35 openseesA_2471.log
```

```
-rw-r--r-- 1 mslyz bockelman  24585 Feb 10 11:35 debug_output_for_1.txt
-rw-r--r-- 1 mslyz bockelman     60 Feb 10 11:35 openseesA_stderr.2471.1.txt
-rw-r--r-- 1 mslyz bockelman   5874 Feb 10 11:35 openseesA_stdout.2471.1.txt
-rw-r--r-- 1 mslyz bockelman 843226 Feb 10 11:35 OpenSees_output_1.tgz
-rw-r--r-- 1 mslyz bockelman  29230 Feb 10 11:35 debug_output_for_0.txt
-rw-r--r-- 1 mslyz bockelman     60 Feb 10 11:35 openseesA_stderr.2471.0.txt
-rw-r--r-- 1 mslyz bockelman   5835 Feb 10 11:35 openseesA_stdout.2471.0.txt
-rw-r--r-- 1 mslyz bockelman 973440 Feb 10 11:35 OpenSees_output_0.tgz
drwxr-xr-x 3 mslyz bockelman   4096 Feb 10 11:35 la29GM
drwxr-xr-x 3 mslyz bockelman   4096 Feb 10 11:35 la28GM
glidein:test4_C$
glidein:test4_C$

# Clean-up:

glidein:test4_C$ cd ..
glidein:opensees$ rm -r test4_C opensees_distA2.tgz test4.csf
glidein:opensees$
```

8 Appendix: Monitoring a running job

1. `condor_ssh_to_job [jobid]`
   This should start a login session at the computer where the job is running. Then it's possible to check the output files etc using standard unix commands like less/more, tail, etc.

2. `glidein_ls [jobid]`
   This lists the files in the directory where condor is running.

   `glidein_interactive [jobid] [command]`
   This runs an arbitrary command in that directory.

   It may be that one or the other of these monitoring options doesn't work for any particular job, but hopefully at least one does.

   Here is a sample session that uses these commands:

```
# On the submit host, set up the job:

glidein:opensees$ ./setup_opensees.pl -c openseesA.csf -o test_fnpB.csf \
   -t opensees_distA2_fnpB.tgz -i exampleTeraGridCode_rl -e ./OpenSees \
   -d test_fnpB -a test_fnpB.dag
Using SAClaRecords.txt as the SACla file.
num_runs: 60
Suggested command:  condor_submit_dag -usedagdir test_fnpB/test_fnpB.dag
glidein:opensees$
glidein:opensees$
glidein:opensees$


# Start it running.

glidein:opensees$ condor_submit_dag -usedagdir test_fnpB/test_fnpB.dag

-----------------------------------------------------------------------
File for submitting this DAG to Condor           : test_fnpB/test_fnpB.dag.condor.sub
Log of DAGMan debugging messages                 : test_fnpB/test_fnpB.dag.dagman.out
Log of Condor library output                     : test_fnpB/test_fnpB.dag.lib.out
Log of Condor library error messages             : test_fnpB/test_fnpB.dag.lib.err
Log of the life of condor_dagman itself          : test_fnpB/test_fnpB.dag.dagman.log

Submitting job(s).
1 job(s) submitted to cluster 2403.
-----------------------------------------------------------------------


# Wait a little while for jobs to start running.

# Do "condor_q" to figure out what the job ids are.
# 2406.0 is one of them in this case.


======================================================================
# Using condor_ssh_to_job:

glidein:opensees$ condor_ssh_to_job 2406.0
Welcome to glidein_20209@red-d8n6!
Your condor job is running with pid(s) 20793.

# We are now logged into the machine where the
# job is running.
```

```
bash-3.2$ ls -lat
total 13004
drwxr-xr-x 6 hcc grid    4096 Feb 10 00:05 .
drwxr-xr-x 2 hcc grid    4096 Feb 10 00:05 .condor_ssh_to_job_3
-rw-r--r-- 1 hcc grid    3948 Feb 10 00:05 .job.ad
-rw-r--r-- 1 hcc grid    5215 Feb 10 00:05 .machine.ad
-rw-r--r-- 1 hcc grid   59748 Feb 10 00:05 openseesA_stdout.2406.2.txt
drwxr-xr-x 2 hcc grid    4096 Feb  9 23:57 .condor_ssh_to_job_2
drwxr-xr-x 2 hcc grid    4096 Feb  9 23:42 .condor_ssh_to_job_1
drwxr-xr-x 3 hcc grid    4096 Feb  9 23:34 ..
-rwxr-xr-x 1 hcc grid    2038 Feb  9 23:34 condor_exec.exe
-rw-r--r-- 1 hcc grid       0 Feb  9 23:34 debug_output_for_2.txt
-rw-r--r-- 1 hcc grid      60 Feb  9 23:34 openseesA_stderr.2406.2.txt
-rw-r--r-- 1 hcc grid 4082242 Feb  9 23:34 opensees_distA2_fnpB.tgz
-rw-r--r-- 1 hcc grid       0 Feb  9 23:34 OpenSees_output_2.tgz
drwxr-xr-x 4 hcc grid    4096 Feb  4 00:41 exampleTeraGridCode_rl
-rwxr-xr-x 1 hcc grid 9099208 Jan 12 16:49 OpenSees
bash-3.2$

# Look at the end of the output file that OpenSees would
# usually print to the screen. "tail" is a standard
# unix command the lists the last few lines.

bash-3.2$ tail openseesA_stdout.2406.2.txt
DirectIntegrationAnalysis::analyze() - the Algorithm failed at time 4.03793
OpenSees > analyze failed, returned: -3 error flag
Trying Reducing This Time Step by 2 ..
WARNING: CTestEnergyIncr::test() - failed to converge
after: 70 iterations
AcceleratedNewton::solveCurrentStep() -The ConvergenceTest object failed in test()
DirectIntegrationAnalysis::analyze() - the Algorithm failed at time 4.03743
OpenSees > analyze failed, returned: -3 error flag
Trying Reducing This Time Step by 10 ..
Converged at this step, continuing....
bash-3.2$
bash-3.2$


# Look the info about the running processes.
```

```
bash-3.2$ ps uaxw | grep -i opensees
hcc       4129  0.0  0.0  61208   816 pts/0    SN+  00:06   0:00 grep -i opensees
hcc      20793  0.0  0.0   8928  1140 ?        SN   Feb09   0:00 /bin/bash -x /var/lib/
hcc      20806 99.9  0.4 130828 104188 ?       RNl  Feb09  31:51 ./OpenSees exampleTera
hcc      24587  0.0  0.0   8928  1140 ?        SN   Feb09   0:00 /bin/bash -x /var/lib/
hcc      24600 99.8  0.4 130828 104052 ?       RNl  Feb09  27:47 ./OpenSees exampleTera
bash-3.2$
bash-3.2$

# Done looking around.

bash-3.2$ logout
Connection to condor-job.red-d8n6 closed.

=============================================================
# Using the glidein_* tools:


# List files in the directory where the job # 2406.0 is running:

glidein:opensees$ glidein_ls 2406.0
OpenSees
OpenSees_output_2.tgz
condor_exec.exe
debug_output_for_2.txt
exampleTeraGridCode_rl
openseesA_stderr.2406.2.txt
openseesA_stdout.2406.2.txt
opensees_distA2_fnpB.tgz
glidein:opensees$


# See the last few line of the output using the tail command again:

glidein:opensees$ glidein_interactive 2406.0 tail openseesA_stdout.2406.2.txt
DirectIntegrationAnalysis::analyze() - the Algorithm failed at time 4.06923
OpenSees > analyze failed, returned: -3 error flag
Trying Reducing This Time Step by 10 ..
Converged at this step, continuing....
```

```
WARNING: CTestEnergyIncr::test() - failed to converge
after: 70 iterations
AcceleratedNewton::solveCurrentStep() -The ConvergenceTest object failed in test()
DirectIntegrationAnalysis::analyze() - the Algorithm failed at time 4.06983
OpenSees > analyze failed, returned: -3 error flag
Trying Reducing This Time Step by 2 ..
glidein:opensees$


# Could do "glidein_cat openseesA_stdout.2406.2.txt"
# to see the whole output so far.


# If the runs were just to experiment with the job
# monitoring tools then it's probably best to
# do a condor_rm on the first job at this point.
```

(Up)