CONDOR and MUGrid

How to submit jobs to a queue

Lars E. Olson
lars.olson@marquette.edu

Marquette University
Department of Biomedical Engineering

July 14, 2010
Why Use MUGrid in the First Place?

- The goal of high performance and high throughput computing is to:
  - A–Break up a big computational task into parts so it can be done in a reasonable amount of time.
  - B–Doing a computational task many times, usually with varying inputs.
- Examples of A include CFD, image analysis, weather, some informatics.
- Examples of B include exploring parameter space for a model and Monte Carlo simulation.
Père offers a large computing resource to do HTC and HPC.
We have to learn how to share Père.
A shared resource needs a mechanism to share: a job manager or queue.
- Condor
- PBS
Everyone needs to submit computational jobs to a queue.
- Condor lets you submit type A and B jobs
- This session will help you learn how to do basic submits to the queue
Two Examples for Condor Submission

- Breaking up a big job into pieces: Mandelbrot image
- Doing a computational task a bunch of times: Monte Carlo seeding
Example 1: Making a Mandelbrot Set

- Making a Mandelbrot set is an iterative task for each pixel
- We can break up an imaging task into pieces row-wise or column-wise or other.
Mandelbrot Programs

- Log onto Père: ssh username@pere.mu.edu or equiv.
- in the mandelbrot directory.....

Three Programs

$ ./mandelbrot     #makes mandelbrot.bmp image
                  #scp it back to your machine
                  #to look at it.

$ ./mandelbrotp 0 16  #makes the zeroth slice of 16
                      #slices of the raw image

$ ./mandelbrotc 16  #connects all the raw slices
                    #into a bmp image.
Working on head node is actually not that desirable.
So let’s send mandelbrot program to the Condor queue.

```
$ condor_submit submitfile

# submitfile is the name of a file
# you create, like mb1.condor
```
The format for a condor submission file is pretty standard.

```
universe = vanilla
executable = mandelbrot
transfer_output_files = mandelbrot.bmp
output = mandelbrot.out
error = mandelbrot.err
log = mandelbrot.log
should_transfer_files = true
when_to_transfer_output = on_exit
queue
```
submit mb1.condor

- Submit a job and watch it.

**Condor Submission cont’d**

```
$ condor_submit mb1.condor

$ condor_q username
```

When your job is done and not on the queue anymore, scp mandelbrot.bmp back to your local machine and have a look.
If there’s a problem......

- How to remove a job from the queue.

**Condor Submission removal**

$ condor_rm username  #or job number

$ condor_q username    #should be gone
The 5 Basic Condor Commands

- (four really....)

<table>
<thead>
<tr>
<th>Condor commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ condor_status</td>
<td>#how’s the cluster doing</td>
</tr>
<tr>
<td>$ condor_q</td>
<td>#show everybody in line</td>
</tr>
<tr>
<td>$ condor_q username</td>
<td>#show my jobs that are in line</td>
</tr>
<tr>
<td>$ condor_submit file</td>
<td>#submit the jobs outlined in file</td>
</tr>
<tr>
<td>$ condor_rm username</td>
<td>#remove my jobs from the queue</td>
</tr>
</tbody>
</table>
Things to learn about Condor

- Passing in arguments to executable
- Using the process number

What we want to do with Condor is break up a big job into smaller ones to go faster. mandelbrotp and mandelbrotc can do that for us, but we need to learn how to implement that in a condor submit file.
Here we learn about arguments and process number.

```condor
universe = vanilla
executable = mandelbrotp
transfer_output_files=mandelbrot$(Process).raw
output = mandelbrot$(Process).out
error = mandelbrot$(Process).err
log = mandelbrot.log
Arguments = $(Process) 16
should_transfer_files = true
when_to_transfer_output = on_exit
queue 16
```
Now submit all 16 jobs with one command.

**Condor Submission**

```
$ condor_submit mb2.condor

$ condor_q username
```

16 jobs show up on the queue. When they are done, you’ll have 16 raw files in your directory.
use mandelbrotc to unite the raw files

- Doing this on head not is not great (more later).

**Concatenate raw files**

```
$ ./mandelbrotc 16

# Then scp mandelbrot.bmp back to your local machine.
# Also, we need to clean up all the files.
```
Monte Carlo simulations do the same task many times with different random seeds.

In this task, we take a number from a file based on process number. (We do nothing with that number, but you could do something.)

It applies to Monte Carlo AND parameter space exploration.
Monte Carlo Programs

- in the montecarlo directory.....

**Two Programs**

```bash
$ ./makerand randseed.txt 1000
    #makes the seed file with 1000 values in it

$ ./getrandno 11   #gets the 11th number in the seed file.
    #and outputs a file out_11.txt.
```
More things to learn about Condor

- Transfering input files through Condor. (also using tar to bundle up files)
- Using a shell script instead of an executable.

Shell scripts are ways to do more than just the executable.
Prep Work before Condor

Things to do first

#optional
$ gcc -o makerand makerand.c  #if you edit it.
$ gcc -o getrandno getrandno.c  #ditto.

$ ./makerand randseed.txt 1000  #make randseed.txt file
$ tar -cvzf in.tar.gz getrandno randseed.txt
    #tar up all needed file into one.

(make getrandno.sh....see next slide)
$ chmod 755 getrandno.sh
    # or chmod +x
getrandno.sh does two things.

```bash
#!/bin/bash
tar xzvf in.tar.gz  # unpacks the tarball
./getrandno $1     # runs getrandno with the first argument in shell call.
```
Two new things.

```
universe = vanilla
executable = getrandno.sh
transfer_input_files = in.tar.gz
transfer_output_files = out_$(Process).txt
output = mc_$(Process).out
error = mc_$(Process).err
log = mc.log
Arguments = $(Process)
should_transfer_files = true
when_to_transfer_output = on_exit
queue 10
```
run getrandno 10 times.

Condor Submission

$ condor_submit mc1.condor

$ condor_q username

When done, there will be out_0.txt through out_9.txt
Other things in condor submit files
Things learned about Condor

- How to submit single and multiple jobs to Condor Q
  - queue vs. queue 16
  - Process number starts at zero, making sure arguments work that way.
- Can call an executable and a shell script.
- Transferring input and output files via condor (tar trick).
Things NOT learned about Condor

- Heterogeneous Condor Pools
  - requirements line in condor submit file.
  - Using the jobrouter.
- Controls on submission like maximum idle jobs setting.
- Optimizing, managing data issues, etc.
- Controlling overall workflow (doing no job on head node).