Introductory hands-on guide to using Condor

Condor was conceived at and is run by the University of Wisconsin http://www.cs.wisc.edu/condor/
In their own words: "Condor is a system that supports High Throughput Computing (HTC) on large collections of distributively owned computing resources"

It does not in itself work out how to split up your huge computing job into collections of smaller jobs: you have to do that, or rely on work by others who have trodden the path and provided the documentation or applications that do this for you.

Condor is a command line system, running on UNIX and Windows, it is *not* point and click! Although Condor allows jobs to be submitted through Windows and UNIX the setup here at Newcastle requires you to submit jobs through UNIX. This guide assumes some basic familiarity with a few UNIX commands and how to write some simple scripts using a UNIX based editor.

See http://www.ncl.ac.uk/iss/research/condor for an introduction to Condor

Submitting jobs via UNIX

Your Condor jobs will need to be submitted from one of the ISS UNIX servers: `condor1.ncl.ac.uk` see: http://www.ncl.ac.uk/iss/unix/ From a PC you need to open up a command shell using the "Secure Shell Client" program [At home you can use the free SSH client PuTTY if you don't have Secure Shell Client]. This is normally in the Communications and Internet folder on the Start menu, if not you need to ask your Computing Officer to install it for you.

You will need to use the following UNIX commands

`mkdir`	create a new directory: `mkdir` *`dirname`*
`cd`	change directory: `cd` *`dirname`*, `cd ..`, `cd` (back to you home directory) [Aside: unlike Windows it is essential to place a space between cd and the directory name]
`ls`	lists contents of current directory (also `ls -alh` gives more detail) depending on client/shell you may need to type `unalias ls` if the colour coding of files makes some of them difficult (or impossible) to read.
`chmod`	change the permissions on directories/files
`nano/pico`	simple visual editor (or use vi if you know it)
`more`	lists a file to the screen, e.g. `more` *`filename`*
`man`	Bring up the manual for a command e.g. `man man`

Further detail on UNIX commands: http://www.ncl.ac.uk/iss/unix/unixhelp/commands.html
You must ensure that you are familiar with the UNIX shell and commands before proceeding.

Windows Command Shell

The largest Condor resource comprises the Windows 7 PC clusters around campus. Your application may run directly on Windows 7, or you may need to "wrap" it in a command shell in order to set up files, environment variables and pass parameters into the application, so you may also need to know some basic Windows Command shell commands. For further details, see:

http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds_shelloverview.mspx

http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.mspx

We'll start with the "hello world" job – all it does is write the words "Hello World" into a file.

Step 1 – Set up a job on the ISS condor server

- Arrange an ISS Unix login
- Open a shell on ISS host condor1.ncl.ac.uk (e.g. using *SSH*)
- In your UNIX home directory create a directory for your Condor job (e.g. *mkdir condor*).
- Change to that directory *(cd condor*) and create a file called `hello.bat` (e.g. using the *vi, pico*, *nano* or GUI based editor – whatever you are happy with and works) containing:

```
echo Hello World > hello.txt
```

- Create a file called `hello.sub` containing the lines:

```
Executable   = hello.bat
Requirements = (Arch == "INTEL") && (OpSys == "WINNT61")
Priority     = high
Universe     = vanilla
should_transfer_files = YES
when_to_transfer_output = ON_EXIT_OR_EVICT
output       = hello.out
error        = hello.err
Log          = hello.log
Queue
```

- ***Important***: Your home directory and any directories containing your Condor job files need to have "execute" permission, and the files that Condor is going to read also need to have "read" permission: Use the *ls -alh* command to check the permissions on your home directory and your condor directory have *"x"* permission for "other" – if not, use *chmod o+x <directory>* to set it. Check the permissions on your condor files have "r" and "x" permission for "other" – if not, use *chmod o+rx <file>* to set it.
  More on file/directory permissions: http://www.ncl.ac.uk/iss/unix/unixhelp/filedir.html

Step 2 – Run your job and monitor it

- Submit your job to Condor by typing the command:
  condor_submit hello.sub
- The system should come back with a few lines, the last one looking something like:

```
1 job(s) submitted to cluster 2345.
```

  Here, 2345 is the identifying number for the (single) job you have just submitted. In your condor directory you will now see (using the command *ls –alh*), as well as `hello.bat` and `hello.sub`, the following files: `hello.log`, and `hello.err`, `hello.out` (both initially empty files)

- Examine the state of your job in the queue by typing the command:
  condor_q 2345

Each time you enter this command it will show your job's status in the queue. When the job has completed it will vanish from the queue, and if you check the contents of your condor directory you will see the file `hello.txt` containing the words `Hello World`. You will also receive an email from "Owner of Condor Daemons" giving details on the job – check your email now.

Congratulations on running your first condor job!

Step 3 – Managing your jobs

Sometimes jobs don't run properly, or indeed at all! To successfully manage your jobs you need to know a little about what to expect in the various log files and how to use a couple more condor commands.

Still in your condor directory, examine the contents of `hello.log`, condor appends job information to the log file each time you submit a job with that log file. In this file you can see "Job executing on host: <IP address>, it also lists the times the job was submitted, ran and terminated, and the bytes of data transferred in/out by the job.

Now add two lines to your hello.bat file so it looks like:

```
echo Hello World > hello.txt
echo This line is written to standard out
ecko This line generates an error!
```

Using *condor_submit* run the job again and, after it completes, examine the contents of `hello.out`, `hello.err`, and `hello.log`

Now make the following change to OpSys in your hello.sub file

```
Requirements = (Arch == "INTEL") && (OpSys == "WINNT40")
```

Using *condor_submit* run the job again – trick question: it *won't* run, ever! ISS do not have any PCs in the Condor cluster running Windows NT.

Step 4 – Analysing your jobs

Type the command *condor_q*, you will see your job `hello.bat` sitting "Idle" (Column "ST" has the value I) in the queue – it isn't going anywhere. If you see your jobs sitting idle indefinitely you probably want to look into why. Use the command *condor_q –analyze <job number>* (or better *condor_q –better-analyze <job number>*) to obtain more information on why your job isn't progressing. Your job may fail to match any available resources, as in this case, or it may match but be rejected for various reasons. Determining the reason for errors in jobs or the reason jobs won't run can be a tricky business! We'll give some more advice on this later...

Now let's clean up after this job, type the condor command:

condor_rm <job number>

The errant job should then vanish from the queue; now have a look at the end of the `hello.log` file.

You should now edit your `hello.bat` and `hello.sub` files so that jobs work again cleanly, we'll leave that to you!

Step 5 -Submitting multiple jobs

Condor won't be much use unless you can run lots of jobs, and you really don't want lots of emails from "Owner of Condor Daemons".

You can turn off email notification by adding the following line to your hello.sub file:

```
notification = never
```

Put this just before the Queue command (which must be the last line of the submit file).
Try this out, check that your job still runs, and that you are **not** sent an email this time.

When you look at the output of the condor_q command you might have noticed the ID number in the first column. These increments by one for each job (or "cluster") submitted [Aside: this is for all users so you may find that your jobs have non-continuous numbers]. You can submit more than one instance in a cluster by changing the queue command in the hello.sub file to tell Condor how many instances you want it to run, for example:

```
Queue 10
```

Try this out, and run the condor_q <job number> command repeatedly from time to time, you will see a cluster of jobs queued up; these will run (and so disappear) in a random order e.g.

```
 ID
1234.0 ...
1234.3 ...
1234.8 ...
```

It's not much use if you can only submit copies of the same job! In reality you need to submit jobs that are different, and you do this by using a parameter `$(Process)` – this process number corresponds to the .0, .1, .2 etc on the Condor job numbers you saw above. Using this you can specify parameters to your executing jobs; specify different input or output files, different initial directories etc.

For example to feed the process number as a "command line argument" to your application add the following line to your submit file:

```
Arguments = ver$(Process)
```

Each job in your cluster will now run with a different command line argument – namely `ver0, ver1, ver2` … which you might use to perform a different calculation for each job.

This will be a waste of time if each job overwrites the output of the others, so you might also want to specify different output files! For example if the job writes to "standard-output" you might add the following line to your submit file:

```
output = $(Process).out
```

You will then get output written to `1.out, 2.out, 3.out ...`

Putting this all together in an example, we have hello.bat

```
echo Hello World > hello.txt
echo This line is written to standard out
echo This job was run on %COMPUTERNAME%
echo This line displays the first command line parameter: %1
```

and hello.sub

```
Executable = hello.bat
Arguments = infile$(Process)
Requirements = (Arch == "INTEL") && (OpSys == "WINNT61")
Priority = high
Universe = vanilla
should_transfer_files = YES
when_to_transfer_output = ON_EXIT_OR_EVICT
output = $(Process).out
error = hello.err
Log =  hello.log
notification = never
queue 3
```

Submit this job and check that you get three files: 0.out, 1.out, 2.out containing the output from running the hello.bat command, notice the different values for the command line parameter and computer names.

If you have little or no control over the files your application reads or writes you might want to specify a set of directories with different names that you can use to hold different data-sets for input files and to store the different output files. You can do this by adding the `InitialDir` line in your submit file:

```
InitialDir = run_$(Process)
```

Create three directories called run_0, run_1 and run_2 – these will contain the input and output files for the jobs. Then submit your cluster of three jobs again. Check that these three directories each contain the files `0.out, 1.out` and `2.out` as well as the `.err, .log` and `.txt` files you originally had all in the same directory. These same directories could for example contain different sets of input data files for processing by the condor job.