High Performance Computing Tutorial: Exercises

Research and Specialist Computing Support

V1.2 November 2013

Connect to Grace.uea.ac.uk

Using PuTTY connect to grace.uea.ac.uk and log in with your UEA username and password.

These exercises guide you through starting an interactive session to submitting sequential and parallel tasks to the task scheduler to be run automatically. Copy the file /gpfs/grace/samplescripts/Training.tar.gz to your home directory with the command `cp /gpfs/grace/samplescripts/Training.tar.gz .`

```
[cc@login00 ~]$ cp /gpfs/grace/samplescripts/Training.tar.gz .
[cc@login00 ~]$ ls Training.tar.gz
Training.tar.gz
```

The file you have copied is a compressed tar file - tar files are an archive of a number of files or directories into a single file. The contents can be viewed by running `tar tf Training.tar.gz` (t - list, f - file)

```
[cc@login00 ~]$ tar tf Training.tar.gz
Training/
Training/HPL.dat
Training/HPLrun.bsub
Training/average.R
Training/R.bsub
Training/montecarlo.m
Training/xhpl
Training/hostname.bsub
Training/fortran_host.f90
Training/resource.bsub
Training/stress
```

You should extract the file using the following command `tar xvzf Training.tar.gz` (x - extract, v - verbose, z - compressed):

```
[cc@login00 ~]$ tar xvzf Training.tar.gz
Training/
Training/HPL.dat
Training/HPLrun.bsub
Training/average.R
Training/R.bsub
Training/montecarlo.m
Training/xhpl
Training/hostname.bsub
Training/fortran_host.f90
Training/resource.bsub
Training/stress
```

This will create a folder 'Training' inside which are the files required for the rest of the exercises

```
[cc@login00 ~]$ ls
Training Training.tar.gz
[cc@login00 ~]$ ls Training
fortran_host.f90 hostname.job HPL.dat HPLrun.job matlab.job montecarlo.m average.R xhpl
resource.bsub stress
```

Interactive

All subsequent tasks should be run in an interactive session, rather than from the login node. Start an interactive session on one of the compute nodes by using the **interactive** command:

```
[cc@login00 ~]$ interactive
Job <1432928> is submitted to queue <interactive>.
<<Waiting for dispatch ...>>
<<Starting on cn136>>
[cc@cn136 ~]$
```

Modules

Modules allow you to select software options and setup the relevant environment variables. Check what modules are available with the **module avail** command which lists all application, tools and library modules. The command **module show modulename** shows more information about a module:

```
[cc@cn136 ~]$ module show matlab/2013a
-------------------------------------------------------------------
/gpfs/grace/modules/apps/matlab/2013a:

module-whatis    adds Matlab 2013a to your environment variables
prepend-path     PATH /gpfs/grace/matlab/2013a/bin
conflict         matlab
set-alias        matlabcli matlab -nosplash -nodesktop -nojvm -singleCompThread
set-alias        matlabcli-mt matlab -nosplash -nodesktop -nojvm
set-alias        matlab matlab -singleCompThread
set-alias        matlab-mt matlab
system           /gpfs/grace/lsf/pre_exec/app_stat --module=matlab/2013a --cluster=grace
-------------------------------------------------------------------
```

For one of the later exercises, we will need to use Matlab, so add a Matlab module with **module add matlab/2013a** for example

```
[cc@cn136 ~]$ which matlab
which: no matlab in (/opt/lsf/7.0/linux2.6-glibc2.3-x86_64/etc:/opt/lsf/7.0/linux2.6-
glibc2.3-
x86_64/bin:/opt/kusu/bin:/opt/kusu/sbin:/usr/kerberos/bin:/bin:/usr/bin/:/usr/mbin:/loca
l/bin:/usr/local:/usr/ucb)
[cc@cn136 ~]$ module add matlab/2013a
[cc@cn136 ~]$ module list
Currently Loaded Modulefiles:
  1) matlab/2013a
 [cc@cn136 ~]$ which matlab
/gpfs/grace/matlab/2013a/bin/matlab
```

We will also be trying compiler modules and R, so add either the Intel compiler (icc/intel/12.1) or Portland Group (pgi/12.6) and R/2.15.1.

Exercise 1 - R

With PuTTY, start a new session on Grace.uea.ac.uk and start a Xinteractive session. You will also need XMing started on your local Windows PC. Make sure you have an R module added and check that R is available:

```
[cc@cn136 ~]$ which R
/gpfs/grace/R-2.15.1/bin/R
```

You can start R by just running **`R`**

```
[cc@cn136 ~]$ R

R version 2.15.1 (2012-06-22) -- "Roasted Marshmallows"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

>
```

There is a simple demonstration R script, average.R, included in Training.tar.gz - you can try running these commands within an interactive R session. You can run the script average.R within R by running **`source("average.R")`**. Use **`q()`** to quit

Try running R in batch mode to run the commands in the script by entering **`R CMD BATCH average.R`** on the Linux command line. The output from this command will be written to average.Rout. When submitted to run in batch mode, the output graph is written to a pdf file, which can be viewed on the **login node** (grace.uea.ac.uk not an interactive session on a compute node) with the command **`xpdf filename.pdf`**.

Use the included job script R.job to submit the R script to the queues using **`bsub < R.bsub`**

```
[cc@cn136 ~/Training]$ bsub < R.bsub
Job <1432962> is submitted to queue <short>.
```

You can monitor the progress of your job by using **`bjobs`**.

Exercise 2 - Matlab

Matlab can be used in an interactive session either on the command line or by using the GUI, however using the GUI can be slow or even unreliable depending on the connection. Instead in this exercise we will submit a Matlab batch job to run automatically without interaction.

The Matlab script montecarlo.m simulates 10million flips of 20 coins and outputs the number out of 20 that the coin lands on heads.

Using an editor such as nano, vi or emacs, modify R.bsub used in the previous exercise to submit this Matlab command to the queue. In your submission script you will need to load a Matlab module (matlab/2013a) and Matlab is launched with the command `matlab -nodisplay -nodesktop -nosplash -r "montecarlo"`

Submit the job. You should find the 10 million simulations in the Matlab job will take about 5 minutes to run.

The Matlab script, montecarlo.m is able to run across multiple cores. In order to make use of additional cores, using an editor such as nano, vi or emacs modify the submission script and montecarlo.m files as follows:

At the top of the submission script, add the following directives:

```
#BSUB –n 9
#BSUB -R 'span[ptile=9]'
```

Matlab montecarlo.m file add:

Before the tic command, add `matlabpool open local 8`
After the toc command, add `matlabpool close`

In the job submission script the –n 9 flag is requesting 9 slots, and the span[ptile=9] flag is requesting that these all be on the same node. In the montecarlo.m file we are asking for 8 additional workers as well as the main Matlab process (therefore 9 slots).

Resubmit the job and see how long the 10 million simulations take to run.

Exercise 3 – Compiling Code

The file fortran_host.f90 is a short and simple piece of Fortran code. There are a number of compilers available on Grace: the Portland Group Compiler suite, the Intel Compiler and the built in Gnu Compiler Collection, each with a Fortran compiler. Loading the appropriate module(s), try compiling the code and running the program.

Gnu compiler - `gfortran fortran_host.f90 -o fortran_host`

Intel compiler - `ifort fortran_host.f90 -o fortran_host`

PGI compiler - `pgf90 fortran_host.f90 -o fortran_host`

Using the hostname.bsub script, submit the job to the queues and check on the progress of the job with `bjobs -X`. This is an example of an array job and produces multiple output files. You can show the contents of multiple files at once using the `cat` command along with the name and wildcard `*`.

Exercise 4 – Resource Limits

The submission script resource.bsub is setup to request 8GB of memory (`#BSUB -R "rusage[mem=8000]”`), so will only be allocated to a node that meets the requirement of having 8GB of RAM available. The submission script also has a memory limit set at 5GB (`#BSUB -M 5000`), meaning the scheduler should automatically terminate the job if the memory usage exceeds this level for more than a short period of time.

Submit the job in its original state. While the job is running, you can get information about the resource usage by running `bjobs -l` ***`jobid`*** depending what other tasks are running on the node your job is allocated to, you may be able to observe the memory usage increase on the cluster monitoring website at http://grace-head00.uea.ac.uk/ganglia/ go to GRACE and choose your node from the drop down box. You can then follow the memory usage graph.

When the job stops check the output and error logs for the reason the job didn’t complete. Modify the job submission script and resubmit so that the job completes successfully.

Exercise 5 - HPL

High Performance Linpack (HPL) is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers. This software package is used to measure the maximum performance of a computer, returning a GFlops figure.

Two files are needed for this exercise - HPLrun.bsub is job script set up for HPL, but can be used as the basis of a generic parallel job script. The file HPL.dat is a configuration file for the run - for the purposes of this exercise, HPL.dat has been configured to produce a relatively quick run. Changing the value of Ns in HPL.dat will increase the measured performance, but the run will take longer and require more memory - please do not increase the value of Ns above 20000 (this may stop others from working and can also cause nodes to crash).

`bjobs` will only show you that you have a 16 slot parallel job running, if you use `bjobs -X` you can see what nodes your job is running on.

You can follow the progress of your job by running `tail -f HPL.out`

This job performs 4 runs, getting progressively larger. The first should take about than 10-15 seconds, the fourth around a minute and a half.

Review

Having completed the HPC exercises, you have used the cluster to work interactively, tunnel graphical windows back to your desktop PC, run batch and array jobs and run a parallel jobs. You have also used, albeit to a basic level, some of the more common programs and compilers used on the cluster

You can use the jobs scripts from these exercises as the basis for your own jobs.