University of East Anglia

High Performance Computing

Tutorial: Exercises

Research and Specialist Computing Support

V1.1 December 2012

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/Random.R
Training/R.bsub
Training/montecarlo.m
Training/xhpl
Training/hostname.bsub
Training/fortran_host.f90
```

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/Random.R
Training/R.bsub
Training/montecarlo.m
Training/xhpl
Training/hostname.bsub
Training/fortran_host.f90
```

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 Random.R xhpl
```

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`** or **`Xinteractive`** 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/2012a
-------------------------------------------------------------------
/gpfs/grace/modules/apps/matlab/2012a:

module-whatis    adds Matlab 2012a to your environment variables
prepend-path     PATH /gpfs/grace/matlab/2012a/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/2012a --cluster=grace
-------------------------------------------------------------------
```

For one of the later exercises, we will need to use Matlab, so add a Matlab module with module add matlab/2011a 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/2012a
[cc@cn136 ~]$ module list
Currently Loaded Modulefiles:
  1) matlab/2012a
 [cc@cn136 ~]$ which matlab
/gpfs/grace/matlab/2012a/bin/matlab
```

We will also be using the Intel compiler and R, so add the modules icc/intel/12.1 and a version of R.

R

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]

>
```

Use **`q()`** to quit

There is a simple demonstration R script, Random.R, included in Training.tar.gz - you can try running these commands within an interactive R session. You can use the command cat Random.R to show the contents of the file then start R and run each line of code before quitting.

Try running R in batch mode to run the commands in the script by entering **`R CMD BATCH Random.R`** on the Linux command line. The output from this command will be written to Random.Rout - rename this file using the mv command to a different name.

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`**. When your job has completed compare the output file Random.Rout to the renamed output file from the previous step (you can use the command **`diff`** for this).

Matlab

With PuTTY, start a new session on Grace.uea.ac.uk and start a Xinteractive session. Make sure you have a matlab module added for this exercise. You will also need XMing started on your local Windows PC.

Try starting Matlab from within your interective session by running the command **`matlab`**. With the Matlab GUI up and running, you can use Matlab as you would on your desktop PC.

Matlab can also be used from the command line, without the need of the GUI. The Matlab script montecarlo.m simulates the repeated flipping of 100 coins - try running the montecarlo script with **`matlab -nodisplay -nodesktop -nosplash -r "montecarlo(100)"`**

Try increasing the argument passed to montecarlo from 100 up to 10000 or higher.

If you use montecarlo.m within a GUI Matlab session, comment out the exit command at the end of the script, otherwise your Matlab session will exit!

The Matlab code produces a graph monte.png - hint: ImageMagick has a viewer called display.

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.

Compilation

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. 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. 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.

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 12 seconds, the fourth around a minute and a half minutes.

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.