Tutorial 2: Pulling it all together - an R example

HPC at NYU

Back to the NYU HPC Wiki
Back to Tutorials index
Back to the Introduction

Prequel: preparing your workstation for the HPC access (and this tutorial)

Accessing software with Environment Modules

Job scripts and how to reserve resources

Introduction to job scheduling

Submitting a job with qsub

Requesting resources

R equi ng G P Us

Using compute nodes interactively
Pulling it all together - Preparing, submitting and monitoring a job on Mercer

In this section we will prepare, submit and monitor a small R job. Our test case comes from the NYU Data Services “Introduction to R” tutorial.

Exercise
Start a terminal session on Mercer and replicate this example in it.
We're using R, so first we'll look for available modules. On mercer:

```
$ module avail r

-------------------------------------- /share/apps/modules/modulefiles
--------------------------------------
r/intel/3.0.3                       raxml/intel/8.0.23
rseqc/intel/2.3.9                   repeat_masker/4.0.5
rstudio/0.98.1028                   raxml/intel/7.3.0
randfold/intel/2.0                  rsem/intel/1.2.15
raxml/intel/7.3.0                   ruby/gnu/2.1.1
```

There's a few modules starting with r, and a couple of versions of R. We'll use the latest version, 3.1.2.

```
$ module purge
$ module list
No Modulefiles Currently Loaded.
$ module load r/intel/3.1.2
```

Take a look at what it did:

```
$ module list
Currently Loaded Modulefiles:
  1) intel/14.0.2        5) cairo/gnu/1.12.16      9) hdf5/intel/1.8.12   13)
 r/intel/3.1.2
  2) jdk/1.7.0_60        6) libxml2/intel/2.9.1    10) netcdf/intel/4.3.1.1
  3) expat/intel/2.1.0   7) openssl/gnu/1.0.1g     11) centos/bin
  4) zlib/intel/1.2.8    8) curl/intel/7.38.0      12) jags/intel/3.4.0
```

... clearly, R uses a lot of other packages. The modulefile has looked after loading the correct ones.

Choose your own example
After - or instead of - following this example through, prepare and submit a run of something genuinely relevant to your research. This way, if you are doing this tutorial in a classroom, the presenter will be available should you have questions or strike difficulties
For our example, we'll get some code and data from /share/apps/examples:

Hint: there are usage examples for a few common packages here

```bash
$ mkdir $SCRATCH/R-example
$ cd !$
$ cp /share/apps/examples/r/r-tut/* .
```

Take a look at the job script:

```bash
$ cat my_R_job.q
#!/bin/bash
#PBS -l nodes=1:ppn=1
#PBS -l walltime=5:00
#PBS -l mem=1GB
#PBS -N jobname
#PBS -M bob.smith@nyu.edu
#PBS -j oe

module purge
module load r/intel/3.1.2
RUNDIR=$SCRATCH/R-example
mkdir -p $RUNDIR
cp /share/apps/examples/r/r-tut/* $RUNDIR

cd $RUNDIR
R --vanilla < IntroToR-Syntax-HPC.R
```
There are a few steps we can try here:

1. Start an interactive batch session, and run the IntroToR-Syntax-HPC.R script interactively
2. Close the interactive session, and submit the batch script as a job:

```
$ qsub my_R_job.q
```

You'll get a job id returned.

Is it running yet?

```
$ qstat -u $USER
```

You could watch the output in the run directory:

```
$ ls -l ${SCRATCH}/R-example
```

Finally, when the job finishes, you should see a .o12345 and .e12345 file in the directory you submitted from.

**Exercise**

Experiment with qsub options for the job name, output and error file merging and location, resource limits.