CANCER RESEARCH UK | CAMBRIDGE INSTITUTE

CRUK cluster introduction (II of III)

Using the scheduler for job submission

UNIVERSITY OF CAMBRIDGE

Overview

Platform LSF (cluster0)

In their own words:

> '*LSF provides the capabilities to manage and accelerate workload processing across heterogeneous distributed compute environments. It is comprised of a comprehensive set of intelligent scheduling policies to ensure that the right resources are automatically allocated to the right jobs, for maximum application performance and efficiency.*'
>
> *Version 7.0.4*

SLURM (Simple Lightweight Unix Resource Manager)

LSF and SLURM will allow you to:

- Submit jobs to the cluster.
- Specify which queue/account to submit your jobs to.
- Request memory resources for your jobs.
- Set memory limits for your jobs.
- Check the status of the jobs you have submitted.
- Check the status of the hosts within the cluster.
- Kill jobs that you have submitted to the cluster.

The most useful SLURM commands

These are:

- **srun**
- **sbatch**
- **sacct**
- **squeues**
- **scancel**

Type command followed by -h for usage details.

```
Usage: srun [OPTIONS...] executable [args...]

Parallel run options:
  -A, --account=name          charge job to specified account
      --acctg-freq=<datatype>=<interval> accounting and
profiling sampling
                              intervals. Supported datatypes:
                              task=<interval> energy=<interval>
                              network=<interval> filesystem=<interval>
…
Help options:
  -h, --help                  show this help message
      --usage                 display brief usage message

Other options:
  -V, --version               output version information and exit
```

The most useful LSF commands

These are:

- **bsub**
- **bjobs**
- **bhosts**
- **bqueues**
- **bkill**

Type command followed by --help for usage details (some accept –h).

Many LSF commands have shared syntax, i.e. the **-l** (long output), **-u** (User) and **-m** (host) flags.

A Job

A command or series of commands submitted to the cluster with associated resource requirements and limits.

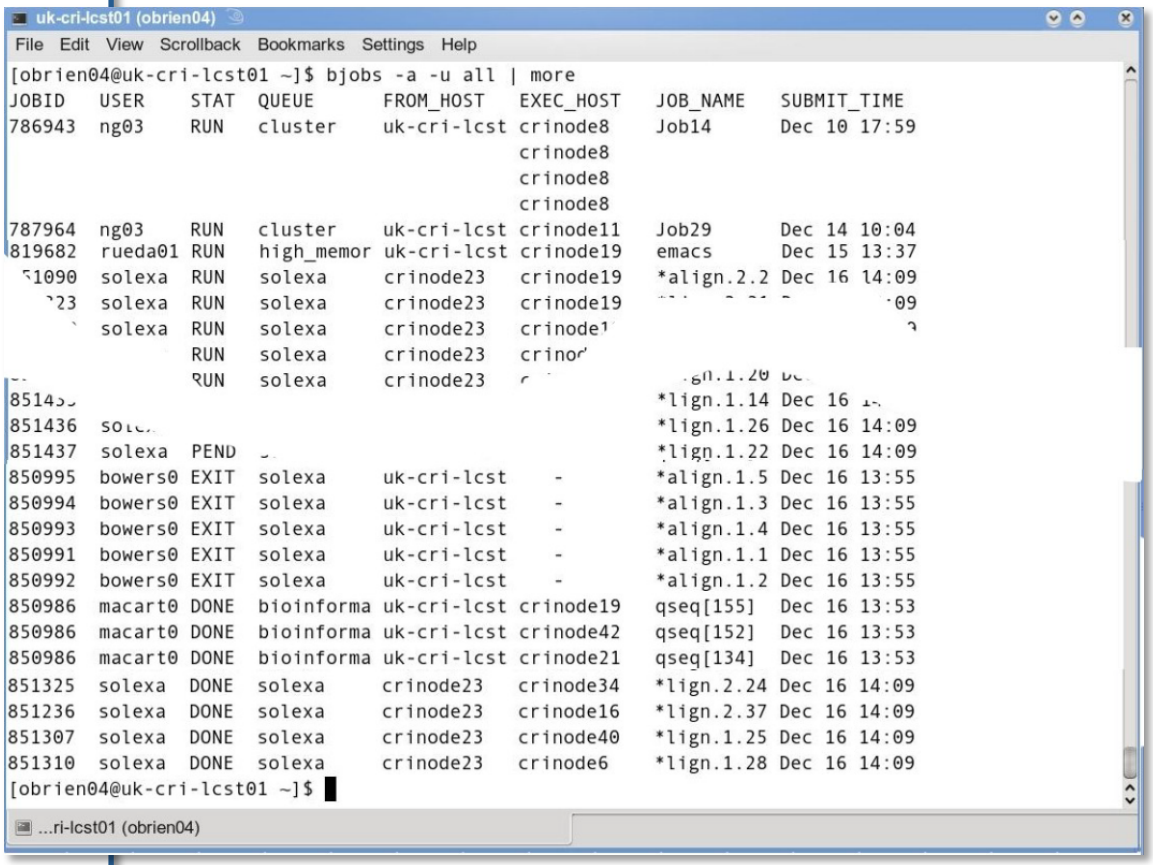
Status of jobs running on the cluster can be seen with the command

LSF

uk-cri-lcst01 > bjobs -a -u all

SLURM

clust1-headnode > squeue



Output from SLURM

clust1-headnode > squeue

```
[obrien04@clust1-headnode ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            427309   general RK307bam   lukk01 PD       0:00      1 (Dependency)
            427311   general RK309bam   lukk01 PD       0:00      1 (Dependency)
…
            436869   general mutect2.    wan01  R    4:25:25      1 clust1-node-16
```

The bsub command

A monster, even by unix standards.

```
bsub [ -h ] [ -V ] [ -H ] [ -x ] [ -r ] [ -N ] [ -
B ] [ -I | -Ip | -Is | -K ]
[ -T time_event ]
[ [ -X "exception_cond([params])::action" ] ... ]
[ -w depend_cond ]
[ -q queue_name ... ] [ -a application_name ]
[ -m host_name[+[pref_level]] ... ]
[ -n min_proc[,max_proc] ]
[ -R res_req ]
[ -J job_name_spec ] [ -b begin_time ] [ -t
term_time ]
[ -i in_file ] [ -o out_file ] [ -e err_file ]
[ -u mail_user ] [ [ -f "lfile op [ rfile ]" ] ... ]
[ -E "pre_exec_command [ argument ... ]" ]
[ -c cpu_limit[/host_spec ] ] [ -W run_limit[/
host_spec ] ]
[ -F file_limit ] [ -M mem_limit ] [ -D data_limit ]
[ -S stack_limit ] [ -C core_limit ]
[ -k "chkpnt_dir [ chkpnt_period ]" ] [ -w
depend_cond ]
[ -L login_shell ] [ -P project_name ]
[ -G user_group ] [ command [ argument ... ] ]
```

LSF

A Queue:

A queue for job submissions associated with specified users and cluster hosts, and providing specified default resources.

You specify the queue to use by adding the
-q <queuename>
option to the bsub command.

SLURM

We have a single general 'partition', with few restrictions (currently).

Most CRI cluster queues have a default memory resource limit of 2GB per job. You can override the default by adding the
-M <memory size in KB>
to the bsub
command.

An example submission request to the queue named cluster to run a job which overrides the 2GB memory limit allowing the job to use 4GBs and requesting that the host has/hosts have 4GB memory available, may look like this:

uk-cri-lcst01 > bsub -q cluster -M 4194304 -R "rusage[mem=4096]" <command>

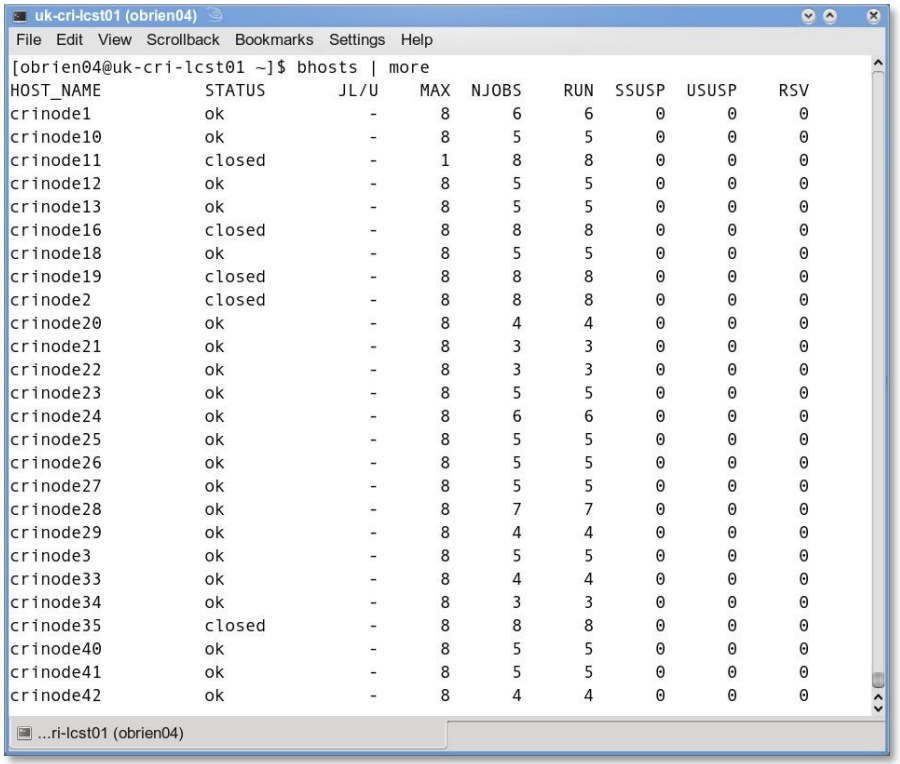
Checking the status of hosts within the cluster

LSF

Check hosts status using the bhosts command
uk-cri-lcst01 > bhosts

SLURM

Clust1-headnode > sinfo




```
[obrien04@clust1-headnode ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
general*    up   infinite     15    mix clust1-node-[1-7,9,12-16,18,21]
general*    up   infinite     18   idle clust1-node-[8,10-11,17,19-20,22-33]
```

LSF
Killing jobs with bkill

You can kill your own jobs if they appear not to be running as intended.

LSF also allows for jobs to be stopped and restarted (provided they were submitted with the **bsub -r** flag).

SLURM
Killing jobs with scancel

clust1-headnode > scancel <job-id>

```
bstop [ -h ] [ -V ] [ -q queue_name ] [ -m host_name ]
[ -u user_name | all ] [ -J job_name ] [ jobId |
"jobId[index_list]" ... ]

bresume [ -h ] [ -V ] [ -q queue_name ] [ -m
host_name ]
[ -u user_name | all ] [ -J job_name ] [ jobId |
"jobId[index_list]" ... ]

bkill [ -h ] [ -V ] [ -l ] [ -s (signal_value |
signal_name ) ]
[ -q queue_name ] [ -m host_name ] [ -u (user_name |
all) ]
[ -J job_name ] [ jobId | "jobId[index_list]" ... ]
```

Running scripts with LSF

Script must be executable, the linux chmod command can be used
to set the executable attribute

uk-cri-lcst01> chmod u+x /lustre/xxlab/xxuser/xxscript.sh

The script can then also be run by redirecting it to the bsub command using one of the linux redirection operators "<“

uk-cri-lcst01> bsub -q cluster -M 4194304 -R "rusage[mem=4096]" < xxscript.sh

SLURM (See practical exercise II)…

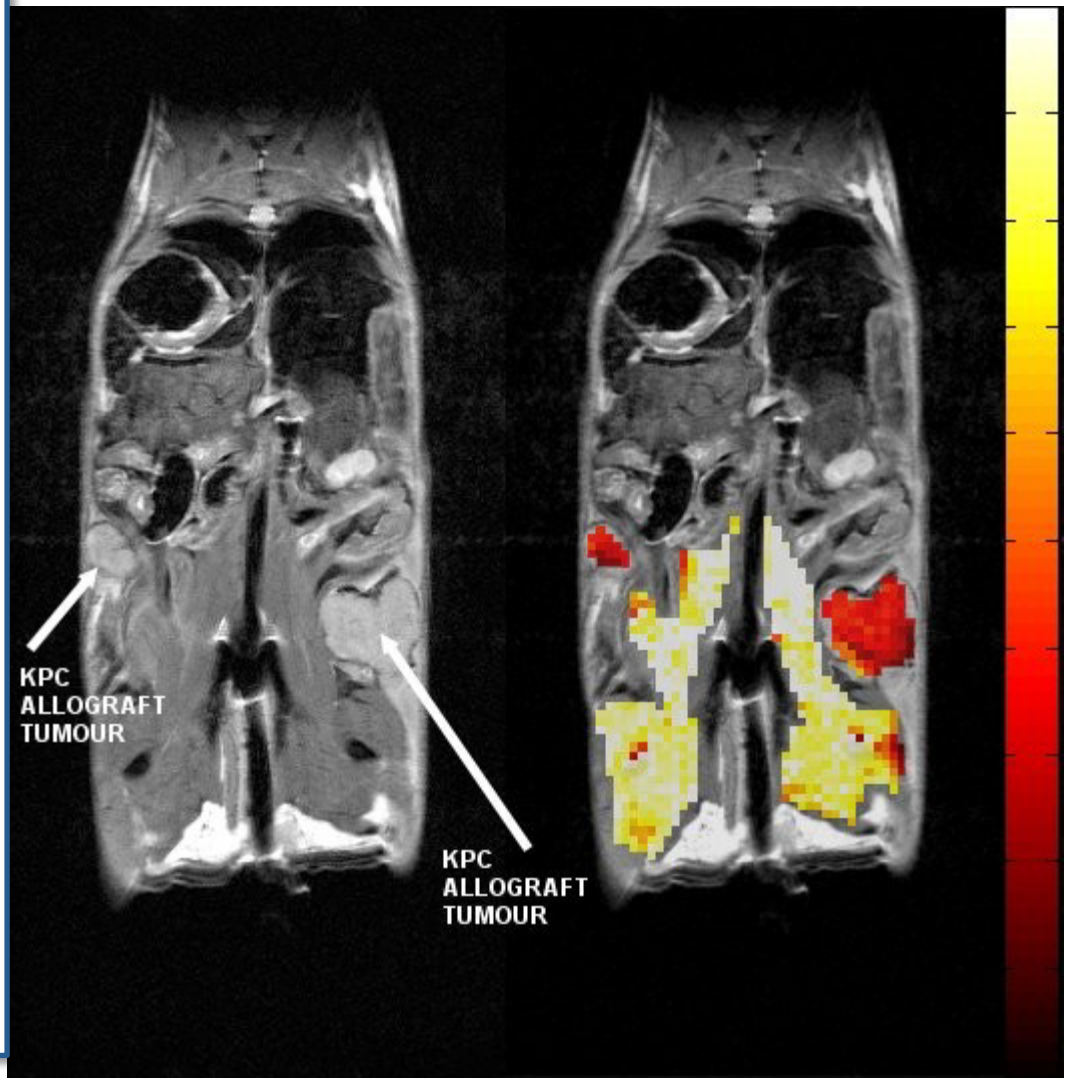
You can also include BSUB options within the script i.e.

```
#!/bin/sh
#BSUB -o myoutput.log
#BSUB -e myerror.log
#BSUB -a R
cd /lustre/xxlab/xxuser
<command>
```

Simple Parallel Computing

Most HPC at CRI consists of breaking up your dataset into chunks and firing off a separate job for each chunk.

> *‘ Magnetisation Transfer Map Example*
>
> *Here small groups of pixels from T2 weighted image slices are processed to calculate the elements of the MT map.’*
>
> (Example courtesy of Dominick McIntyre, Griffiths Group)



Message Passing parallel jobs

Uses the generic PJL (Parallel Job Launcher) framework. You can easily recognise it because of the use of the -a openmpi flag and mpirun.lsf

```
uk-cri-lcst01 > bsub -o %J.out -e %J.err -n 4 -R "span[ptile=1]"
-a openmpi mpirun.lsf ./test
```

In recent versions of LSF, another framework is also available, and it permits a tight (native) integration with the MPIs (this is why there is the OpenMPI integration)

```
uk-cri-lcst01 > bsub -o %J.out -e %J.err -n 4 -R "span[ptile=1]"
mpirun ./test
```

A job submission example

from Ben Davis, bioinformatics

'This is a shell command line loop I ran recently... quite a good example as the actual thing running is just creating checksum files so doesn't distract.'

```
for f in `seq 1 16`; do bsub -n 1 -M 1048576 -R 'span[hosts=1]
select[mem>=1024 && tmp>=2000] rusage[mem=1024, tmp=2000]' -o md5cs-%J.out
-J md5cs$f -q solexa md5sum -c SJD_$f.md5;done;
```

'Here multiple named jobs are submitted to the solexa queue. Each checks a single file specified by numeric indices incorporated in the file title and creates a named output file where the title contains the job ID.'

Job array example

from Stewart MacArthur, bioinformatics

'Here is a self contained example of the basics of using job arrays. The main benefit in this case is not speed but the ability to control the number of jobs running at any one time, using the %50 notation in the bsub ... I find job arrays particularly useful for running lots of small jobs, as there is only a single job submission there is little LSF overhead, compared to submitting 1000 separate jobs, which takes some time. Also being able to control the number of running jobs stops you swamping your queue with jobs and leaves space for others to get jobs running.'

```
### Generate a random big file that we want to sort, 10 Million lines
perl -e 'for (1..1E7){printf("%.0f\n",rand()*1E7)};' > bigFile

### Split the file up into chunks with 10,000 lines in each chunk
split -a 3 -d -l 10000 bigFile split

### rename the files on a 1-1000 scheme not 0-999
for f in split*;do mv ${f} $(echo ${f} |perl -ne 'm/split(0*)(\d+)/g;print
"Split",$2+1,"\n";');done

### submit a job array, allowing 50 jobs to be run at anyone time
bsub -J "sort[1-1000]%50" "sort -n Split\$LSB_JOBINDEX > Split\
$LSB_JOBINDEX.sorted"

### merge the sorted files together once all the jobs are finished
sort -n -m *.sorted > bigFile.sorted

### Delete the temp files
rm -f Split*
```

LSF Job dependency

-w 'dependency_expression'

LSF does not place your job unless the dependency expression evaluates to TRUE. The dependency expression is a logical expression composed of one or more dependency conditions.

To make dependency expression of multiple conditions, use the following logical operators:
&& (AND) || (OR) **!** (NOT)

Use the * with dependency conditions to define one-to-one dependency among job array elements such that each element of one array depends on the corresponding element of another array. The job array size must be identical.

For example:

```
bsub    -w "done(myarrayA[*])"
        -J "myArrayB[1-10]"
        myJob2
```

SLURM Job dependency

```
[clust1-headnode ~] $ sbatch job1.sh
11254323
[clust1-headnode ~] $ sbatch --dependency=afterok:11254323 job2.sh
```

CANCER RESEARCH UK | CAMBRIDGE INSTITUTE

Practical session II

UNIVERSITY OF CAMBRIDGE