CANCER RESEARCH UK | CAMBRIDGE INSTITUTE

CRUK cluster introduction (II of III)

Using the scheduler for job submission

UNIVERSITY OF CAMBRIDGE

Overview

SLURM (Simple Lightweight Unix Resource Manager)

In their own words:

> 'Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. Slurm requires no kernel modifications for its operation and is relatively self-contained. As a cluster workload manager, Slurm has three key functions. First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work..'

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.
- Set time 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**
- **squeue**
- **scancel**

Type command followed by -h for usage details.

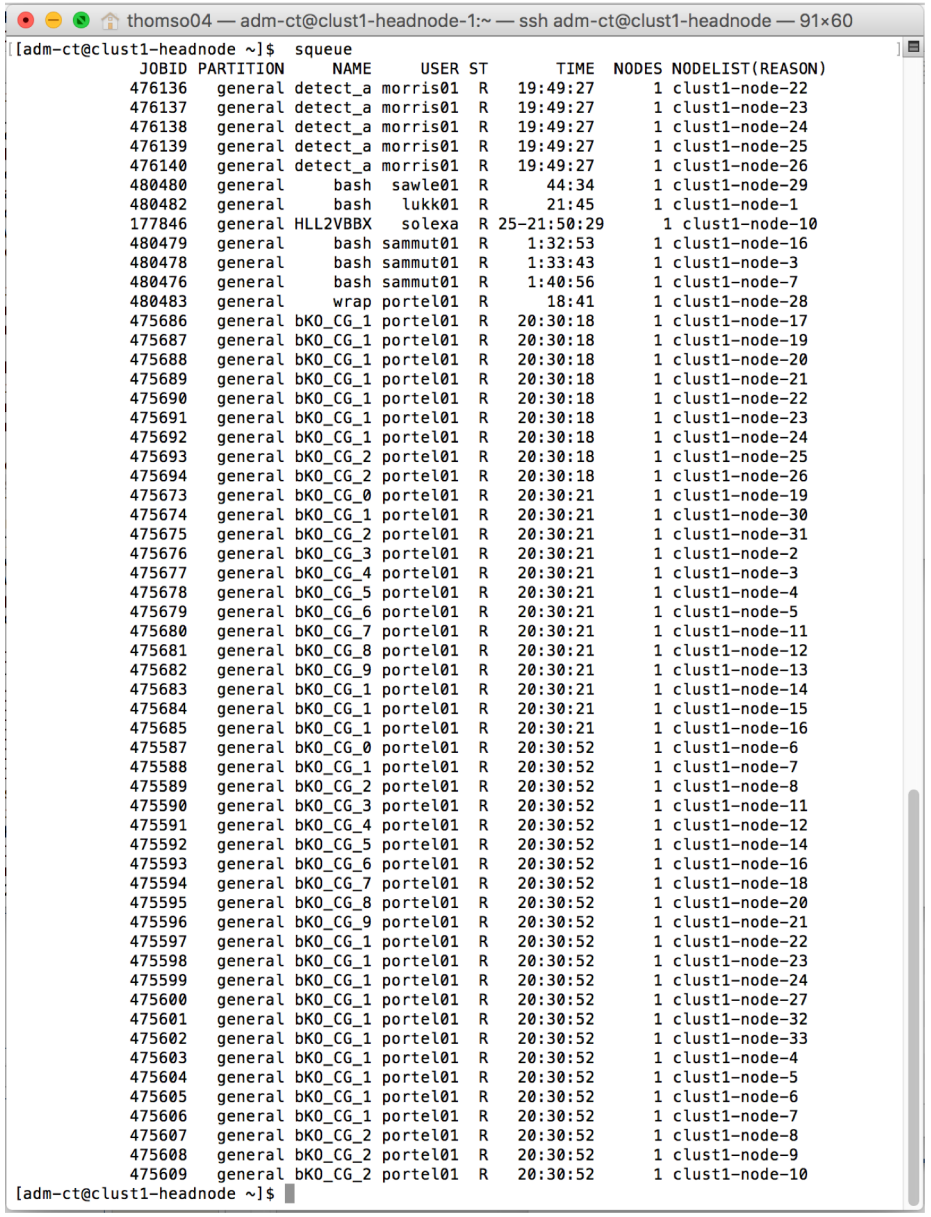
```
Usage: srun [OPTIONS...] executable [args...]

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
```

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

clust1-headnode **$ squeue**



A Queue:

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

SLURM calls queues ‘partitions.’

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

Checking the status of hosts within the cluster

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]
```

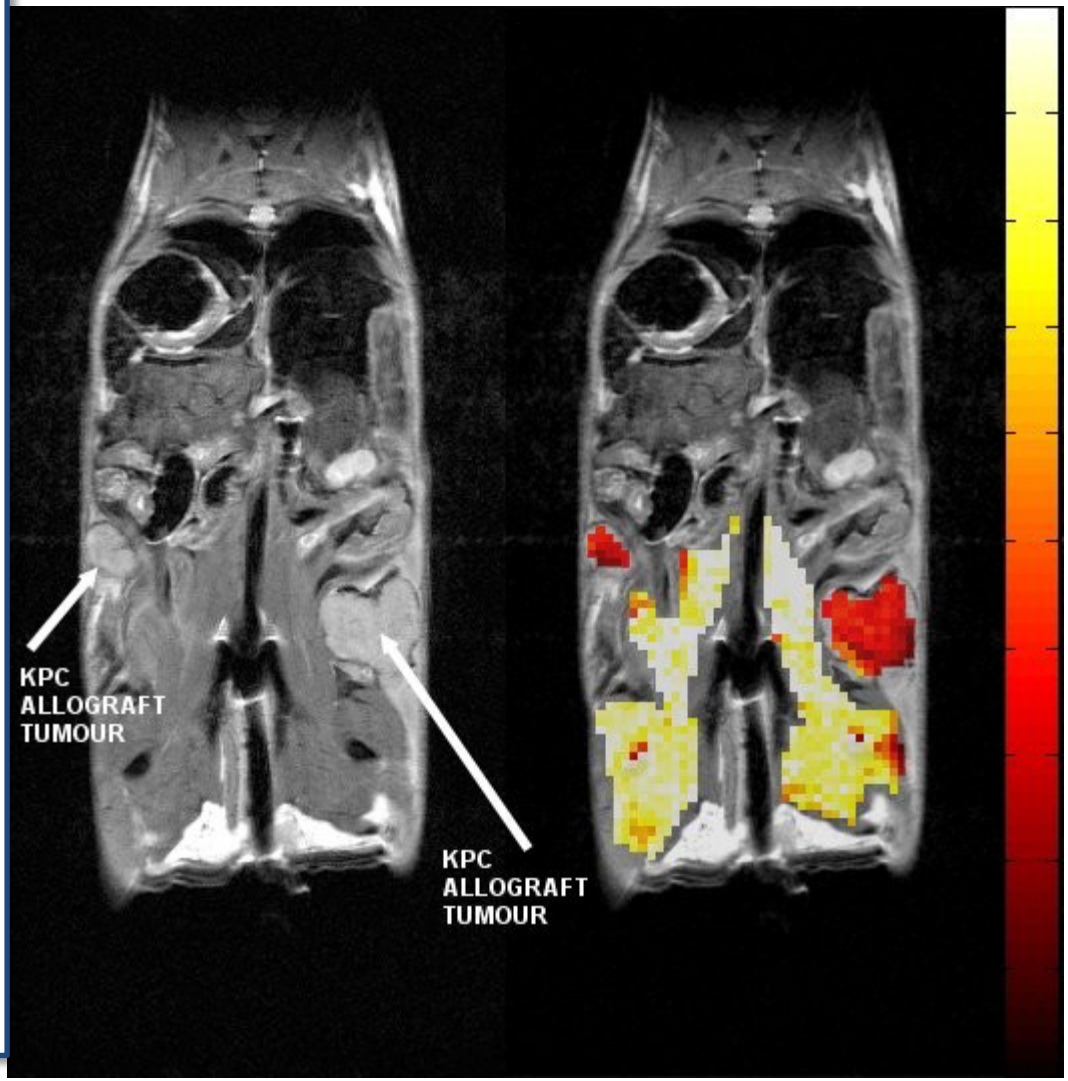
Killing jobs with scancel

clust1-headnode $ **scancel** **<job-id>**

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)



A job submission example

from Ben Davis, Bioinformatics Core Facility

'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 srun --time=0-2 -N1 -o md5cs-%J.out -J md5cs$f
md5sum -c SJD_$f.md5;done;
```

'Here multiple named jobs are submitted. 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

Adapted from an example by Stewart MacArthur, Bioinformatics Core Facility

'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 sbatch... I find job arrays particularly useful for running lots of small jobs, as there is only a single job submission there is little SLURM 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
sbatch --time=1:0 --array=1-1000%50 -N1 ./sort-script.sh

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

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

```
[thomso04@clust1-headnode-1 /home/thomso04]# cat sort-script.sh
#!/bin/bash
/usr/bin/sort -n /scratcha/computing/Split$SLURM_ARRAY_TASK_ID > /scratcha/computing/Split$SLURM_ARRAY_TASK_ID.sorted
```

Job dependency

SLURM allows many different ways to express dependencies, using the `--dependency` switch

Some SLURM dependencies:

`after:job_id[:jobid...]`
This job can begin execution after the specified jobs have begun execution.

`afterany:job_id[:jobid...]`
This job can begin execution after the specified jobs have terminated.

`afterok:job_id[:jobid...]`
This job can begin execution after the specified jobs have successfully executed (ran to completion with an exit code of zero).

```
[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