



CANCER
RESEARCH
UK

CAMBRIDGE
INSTITUTE

CRUK CI HPC cluster introduction (II of III)

Using the scheduler for job submission



UNIVERSITY OF
CAMBRIDGE

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:

- **sinfo**
- **squeue**
- **sbatch**
- **scancel**
- **salloc**
- **srun**
- **scontrol**
- **sstat**
- **sacct**
- **sacctmgr**

Type command followed by -h for usage details.

```
man <COMMAND>
<COMMAND> --usage, lists options
<COMMAND> --help, explain options
```

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

SLURM documentation: <https://slurm.schedmd.com/>

SLURM summary: <https://slurm.schedmd.com/pdfs/summary.pdf>

A Job

Job = SLURM instructions + your program

srun [OPTIONS...] executable [args...]

Can be a single command or a complex piece of BASH programming

In CI, many jobs are done in parallel

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**

clust1-headnode \$ **squeue -u <USERNAME>**

```
[adm-ct@clust1-headnode ~]$ squeue
   JOBID PARTITION   NAME   USER ST      TIME  NODES NODELIST(REASON)
476136 general detect_a morris01 R 19:49:27 1 clust1-node-22
476137 general detect_a morris01 R 19:49:27 1 clust1-node-23
476138 general detect_a morris01 R 19:49:27 1 clust1-node-24
476139 general detect_a morris01 R 19:49:27 1 clust1-node-25
476140 general detect_a morris01 R 19:49:27 1 clust1-node-26
480480 general bash sawle01 R 44:34 1 clust1-node-29
480482 general bash lukk01 R 21:45 1 clust1-node-1
177846 general HLL2VBBX solexa R 25-21:50:29 1 clust1-node-10
480479 general bash sammuto1 R 1:32:53 1 clust1-node-16
480478 general bash sammuto1 R 1:33:43 1 clust1-node-3
480476 general bash sammuto1 R 1:40:56 1 clust1-node-7
480483 general wrap portel01 R 18:41 1 clust1-node-28
475686 general bKO(CG_1 portel01 R 20:30:18 1 clust1-node-17
475687 general bKO(CG_1 portel01 R 20:30:18 1 clust1-node-19
475688 general bKO(CG_1 portel01 R 20:30:18 1 clust1-node-20
475689 general bKO(CG_1 portel01 R 20:30:18 1 clust1-node-21
475690 general bKO(CG_1 portel01 R 20:30:18 1 clust1-node-22
475691 general bKO(CG_1 portel01 R 20:30:18 1 clust1-node-23
475692 general bKO(CG_1 portel01 R 20:30:18 1 clust1-node-24
475693 general bKO(CG_2 portel01 R 20:30:18 1 clust1-node-25
475694 general bKO(CG_2 portel01 R 20:30:18 1 clust1-node-26
475673 general bKO(CG_0 portel01 R 20:30:21 1 clust1-node-19
475674 general bKO(CG_1 portel01 R 20:30:21 1 clust1-node-30
475675 general bKO(CG_2 portel01 R 20:30:21 1 clust1-node-31
475676 general bKO(CG_3 portel01 R 20:30:21 1 clust1-node-2
475677 general bKO(CG_4 portel01 R 20:30:21 1 clust1-node-3
475678 general bKO(CG_5 portel01 R 20:30:21 1 clust1-node-4
475679 general bKO(CG_6 portel01 R 20:30:21 1 clust1-node-5
475680 general bKO(CG_7 portel01 R 20:30:21 1 clust1-node-11
475681 general bKO(CG_8 portel01 R 20:30:21 1 clust1-node-12
475682 general bKO(CG_9 portel01 R 20:30:21 1 clust1-node-13
475683 general bKO(CG_1 portel01 R 20:30:21 1 clust1-node-14
475684 general bKO(CG_1 portel01 R 20:30:21 1 clust1-node-15
475685 general bKO(CG_1 portel01 R 20:30:21 1 clust1-node-16
475587 general bKO(CG_0 portel01 R 20:30:52 1 clust1-node-6
475588 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-7
475589 general bKO(CG_2 portel01 R 20:30:52 1 clust1-node-8
475590 general bKO(CG_3 portel01 R 20:30:52 1 clust1-node-11
475591 general bKO(CG_4 portel01 R 20:30:52 1 clust1-node-12
475592 general bKO(CG_5 portel01 R 20:30:52 1 clust1-node-14
475593 general bKO(CG_6 portel01 R 20:30:52 1 clust1-node-16
475594 general bKO(CG_7 portel01 R 20:30:52 1 clust1-node-18
475595 general bKO(CG_8 portel01 R 20:30:52 1 clust1-node-20
475596 general bKO(CG_9 portel01 R 20:30:52 1 clust1-node-21
475597 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-22
475598 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-23
475599 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-24
475600 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-27
475601 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-32
475602 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-33
475603 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-4
475604 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-5
475605 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-6
475606 general bKO(CG_1 portel01 R 20:30:52 1 clust1-node-7
475607 general bKO(CG_2 portel01 R 20:30:52 1 clust1-node-8
475608 general bKO(CG_2 portel01 R 20:30:52 1 clust1-node-9
475609 general bKO(CG_2 portel01 R 20:30:52 1 clust1-node-10
[adm-ct@clust1-headnode ~]$
```

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
```

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
Clust1-headnode > squeue
Clust1-headnode > sacct

```
[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]
```

Checking the status of hosts within the cluster

SLURM

Clust1-headnode > sinfo
Clust1-headnode > squeue
Clust1-headnode > sacct

```
[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]
```

Submitting a simple command job

SLURM

<COMMAND> [OPTIONS...] executable [args...]

Clust1-headnode > srun -n2 -l hostname

Allocate Resources

The more specific your jobs are, the more likely they are to be run soon

sbatch option

--mem-per-cpu

--time=<walltime>

-t <walltime>

--nodes=<number>

-N <number>

--ntasks=<number> -n <number>

-n <number>

--output=<path>/<file pattern>

-o <path>/<file pattern>

--error=<path>/<file pattern>

-e <path>/<file pattern>

--mail-user=<email>

--job-name=<jobname>

-J <jobname>

--account=<project account>

-A <project_account>

--requeue or --no-requeue

Submitting a batch script job

SLURM

Clust1-headnode > `sbatch myscript.sh`

```
#!/bin/bash#
#SBATCH -p general # partition (queue)
#SBATCH -N 1 # number of nodes
#SBATCH -n 1 # number of cores
#SBATCH --mem 100 # memory pool for all cores
#SBATCH -t 0-2:00 # time (D-HH:MM)
#SBATCH -o slurm.%N.%j.out # STDOUT
#SBATCH -e slurm.%N.%j.err # STDERR
for i in {1..100}; do
echo $RANDOM >> SomeRandomNumbers.txt
Done
sort SomeRandomNumbers.txt
```

Killing jobs with scancel (jing)

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

Simple Parallel Computing

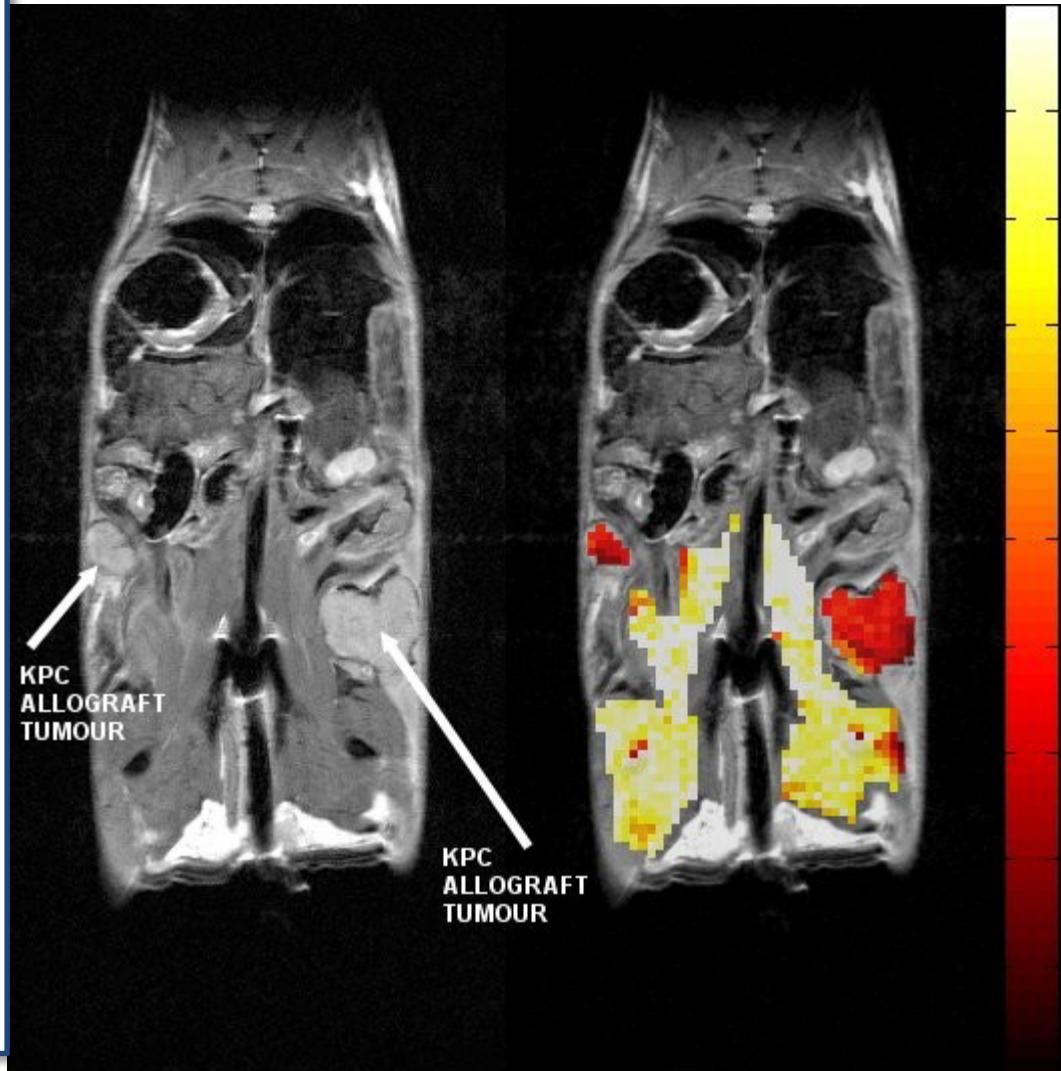
(marc)

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)



Simple Parallel Computing

Example

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