Using SLURM on COSMA

Or job scheduling and
how to use it and control it

ICC Theory Lunch
September 2019
Peter W. Draper

- SLURM is a job scheduling system that controls access to all the compute nodes of the three COSMA clusters.
- We have a single SLURM server that manages all the nodes, so you can submit and view jobs from anywhere in the system.
- Time is given to jobs on the basis of priority, this comes from two sources, the priority given to a user and the priority given to a queue.
- All users have the same fairshare of 1, which is adjusted according to several factors, like how successful they have been at getting time, how long the job has queued and how large is the job. Large jobs are preferred over small jobs. The current, system wide, job priorities can be seen using the sprio command.

Main COSMA queues/partitions

COSMA5:

- *cosma*: ~300 exclusive nodes for MPI jobs
- *cordelia*: 9 shared nodes for serial jobs
- *cosma-shm*: 1 node for large shared memory jobs

COSMA6:

- *cosma6*: ~500 exclusive nodes for MPI jobs

COSMA7:

- *cosma7*: 452 exclusive nodes
- *cosma7-shm*: 1 node for large shared memory jobs

Dirac allocations and the pauper queues

On COSMA6 and COSMA7 when a project (namely dp004) exceeds its quarterly allocation all jobs submitted to the main queues are automatically demoted to a *pauper* queue. These run at a lower priority for a maximum of 24 hours.

To check how our project is performing this quarter you can look at the local usage pages:

- https://virgodb.cosma.dur.ac.uk/usage/login.php

Need username and password (ask if you'd like one set), or you can generate a token from the command line to get an one-off email login.

COSMA7 quarterly usages by main project

Note the current quarter normally runs to the end of yesterday. The Cumulative percentage is normalised to the number of days in the quarter that have been available up to yesterday, so will be 100 if a project has used all the time available up to yesterday. The Percent figure is counted for the whole quarter. The separated current quarter figures include times for jobs that are still running, so are just better estimates, they also include the host institution allocation adjustment.

Current quarter

Show 50 entries Search:

Project	Quarter	Start	End	Available CPU hours	Used	Remaining	Percent	Cumulative
di004	2019/Q3	201907	201909	1000.000	0.000	1000.000	0.00	0.00
dp002	2019/Q3	201907	201909	340000.000	0.000	340000.000	0.00	0.00
dp004	2019/Q3	201907	201909	23078889.600	11576368.410	11502521.190	50.16	65.00
dp012	2019/Q3	201907	201909	1140000.000	1129131.242	10868.758	99.05	128.34
dp016	2019/Q3	201907	201909	430000.000	5081658.258	-4651658.258	1181.78	1531.32
dp034	2019/Q3	201907	201909	980000.000	928308.881	51691.119	94.73	122.74
dp036	2019/Q3	201907	201909	1670000.000	30.054	1669969.946	0.00	0.00
dp104	2019/Q3	201907	201909	1500000.000	148974.117	1351025.883	9.93	12.87
dr006	2019/Q3	201907	201909	0.000	14.179	-14.179	100.00	100.00
ds007	2019/Q3	201907	201909	5000.000	4582.911	417.089	91.66	118.77

Showing 1 to 10 of 10 entries Previous Next

Projects whose percent figure is 100 or greater will be considered over budget for the quarter and will be automatically demoted to the cosma7-pauper queue, until the start of the next quarter or they have an increased allocation.

All quarters

Show 50 entries Search: dp004

Project	Quarter	Start	End	Available CPU hours	Used	Remaining	Percent
dp004	2018/Q2	201805	201806	2611000.000	1050147.385	1560852.615	40.22
dp004	2018/Q3	201807	201809	3916600.000	3038811.022	877788.978	77.59
dp004	2018/Q4	201810	201812	3916600.000	4885639.615	-969039.615	124.74
dp004	2019/Q1	201901	201903	10588000.000	11629695.641	-1041695.641	109.84
dp004	2019/Q2	201904	201906	17490000.000	12387257.780	5102742.220	70.82
dp004	2019/Q3	201907	201909	17490000.000	11392909.128	6097090.872	65.14

Showing 1 to 6 of 6 entries (filtered from 43 total entries) Previous Next

Submitting jobs

Jobs can be submitted from any node, login and compute, and the main command to submit a jobs is:

sbatch

Which has a lot of options (man sbatch), so it is usual to use a submission script for anything more than trivial commands.

There are examples of scripts in the directory:

/cosma/home/sample-user/

To submit a job use the command:

sbatch submission-script.sh

Type of batch jobs

SLURM can run any type of (non-interactive) command, but the most likely ones are MPI based, shared memory, hybrid, that is shared memory with MPI ranks, and plain one-core serial jobs.

MPI jobs use many single core ranks on more than one node. Shared memory jobs use a number of process threads on a node (OpenMP or pthreads). Hybrid jobs use many MPI ranks with process threads at the scale of nodes.

Each require different submission script parameters and maybe queues.

Example MPI submission script

```
#!/bin/bash -l
#SBATCH --ntasks 512
#SBATCH -J job_name
#SBATCH -o %J.out
#SBATCH -e %J.err
#SBATCH -p cosma
#SBATCH -A durham
#SBATCH --exclusive
#SBATCH -t 72:00:00
#SBATCH --mail-type=END
#SBATCH --mail-user=<email address>

module purge
module load intel_comp intel_mpi hdf5

mpirun -np $SLURM_NTASKS your_program your_inputs
```

Note we require -p, -A and -t.

Example hybrid submission script (1)

```
#!/bin/bash -l
#SBATCH -A durham
#SBATCH -p cosma
#SBATCH --nodes 4
#SBATCH --tasks-per-node=1
#SBATCH -o %J.out
#SBATCH -e %J.err
#SBATCH -t 72:00:00
#SBATCH -J job_name
#SBATCH --exclusive
#SBATCH --mail-type=END
#SBATCH --mail-user=<email address>

module purge
module load intel_comp intel_mpi hdf5
mpirun -np $SLURM_NTASKS your_program your_inputs
```

Allocates 4 nodes with one task per node, so you have all the threads on node available. This is best for an OpenMP job that wants to use all the cores on each node (including hyperthreads).

Example hybrid submission script (2)

```
#!/bin/bash -l
#SBATCH -A durham
#SBATCH -p cosma
#SBATCH --cpus-per-task=16
#SBATCH --ntasks=4
#SBATCH -o %J.out
#SBATCH -e %J.err
#SBATCH -t 72:00:00
#SBATCH -J job_name
#SBATCH --exclusive
#SBATCH --mail-type=END
#SBATCH --mail-user=<email address>
```

Gives same as previous page (for COSMA5 and COSMA6). The advantage of this method is that if you want to play with using more memory per core for normal MPI jobs then you can use `--cpus-per-task=2` to get twice the usual 8GB per core on COSMA5 and COSMA6 etc. For an OpenMP job you should also add

```
export omp_threads=$SLURM_CPUS_PER_TASK
```

Example hybrid submission script (3)

```
#!/bin/bash -l
#SBATCH --ntasks=28
#SBATCH --ntasks-per-node=7
#SBATCH --nodes=4
#SBATCH -J test
#SBATCH -o test.out
#SBATCH -e test.err
#SBATCH -p cosma7
#SBATCH -A dp004
#SBATCH --exclusive
#SBATCH -t 0:10:00
#SBATCH --mail-type=END
#SBATCH --mail-user=<userid>
```

This final example places 7 ranks per node on 4 nodes. Note the --ntasks value is also required although it seems superfluous (example 2 is easier to understand).

Serial jobs

Serial jobs are those that use a single core with no threads.

On COSMA5 these can easily be submitted to the *cordelia* queue, usually as array jobs. These are simply jobs that have an argument `--array`.

Each job from now has the variable `SLURM_ARRAY_TASK_ID` set to the index of the job.

Example serial submission script

```
#!/bin/bash -l

#SBATCH --ntasks=1
#SBATCH -A durham
#SBATCH -p cordelia
#SBATCH --array=1-16
#SBATCH -o %A_%a.out
#SBATCH -e %A_%a.err
#SBATCH -t 72:00:00
#SBATCH -J array_job
#SBATCH --mail-type=END
#SBATCH --mail-user=<email address>

echo "My SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID

myprogram $SLURM_ARRAY_TASK_ID
```

Obviously `SLURM_ARRAY_TASK` is set to the values 1 through 16.

Serial jobs on other queues

The other main queues only allow exclusive use of single nodes, so if you want to use these for serial work, then extra effort is required as you need to run 16/28 instances of the program at the same time using one submission.

One way to do this (by John Helly) can be found at:

```
/cosma/home/sample-user/parallel_tasks/
```

Check the README for instructions. It solves the problem of how you can run a number of programs when the runtimes may vary, so simple background and wait techniques fail. Similar results are possible with Python subprocesses and, for the UNIX purists, `xargs -P`.

Inspecting and controlling the state of jobs

Once submitted you can view the state of your jobs using the command:

```
squeue
```

This has many command line arguments. So see just your jobs on a queue you'd use:

```
squeue -u <username> -p <partition>
```

For instance to see all the jobs running on COSMA7:

```
squeue -p cosma7,cosma7-pauper,cosma7-prince
```

See the man page for many more options. To stop a job you use:

```
scancel <jobid>
```

It is also possible to change the properties (runtime, queue, hold status) of a pending job using:

```
scontrol update jobid=<jobid> <property>=<value>
```

Overall state of queues

The simplest command to see the state of all the queues you have access rights for is:

```
sinfo -s
```

```
PARTITION     AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
cosma7           up 3-00:00:00     420/30/2/452  m[7001-7452]
cosma7-pauper    up 1-00:00:00     420/30/2/452  m[7001-7452]
cosma6*          up 3-00:00:00     526/0/36/562  m[6001-6562]
cosma6-pauper    up 1-00:00:00     526/0/36/562  m[6001-6562]
cosma            up 3-00:00:00    273/17/12/302  m[5109-5410]
cordelia         up 30-00:00:0          7/2/0/9  m[5411-5419]
```

The `NODES` fields show how many nodes are Active, Idle, Other and Total. Active nodes are running jobs, Idle nodes are not (so are interesting if you are looking for time). Other nodes are down for various reasons, ignore those.

Finding time, i.e. backfilling

When developing code or testing configurations it is usually the case that you don't need a lot of time. When that is true and the queues are busy *backfilling* is your friend.

If there are idle nodes then that means they are available to run jobs, or that they are being kept so that a job can run in the future. The time between when a job needs those nodes and now is the backfill window and jobs that need less than that time will be scheduled to run.

To see the backfill windows for a system use the local commands:

```
c5backfill, c6backfill or c7backfill
```

Depending on the queue family you are interested in.

```
Backfill availability for cosma queue:
HOSTS/SLOTS:   16 / 256
RUNTIME:       00 hours 51 minutes 51 seconds
HOSTNAMES:     m5278 m5284 m5237 m5406 m5405 m5404 m5384 m5385 m5184 m5259 m5256 m5244 m5188
m5260 m5261 m5403
HOSTS/SLOTS:   1 / 16
RUNTIME:       00 hours 38 minutes 15 seconds
HOSTNAMES:     m5391
```

At the time of writing this was the only backfill available (ouch), so we only have 16 nodes available for 51 minutes and 17 for 38 minutes.

Submit quickly as this time is reducing and others may jump in!

When will my job run

If no backfill is suitable, or you are in production then the next question is when will my job run?

That depends on a number of factors, like jobs completing early, people with higher priorities submitting after you and so on, but you can see what SLURM thinks at any moment using the command:

```
scontrol show job <jobid>
```

Look closely and you'll see an attribute like:

```
StartTime=2019-09-09T18:54:08
```

If nothing changed this is when your job would run.

Since this is not easy to see, you can use our local command `showq` that also gives a better overview of the state of the queues than `squeue` and shows when jobs are estimated to run.

The output is too wide for a slide, but if you use the options:

```
showq -q cosma -l
```

That shows all the running jobs with the fields:

```
JOBID USER ACCOUNT STATE CORE NODE QUEUE SUBMIT-TIME START-TIME TIME-REMAINING JOBNAME
```

And pending jobs with:

```
JOBID USER ACCOUNT STATE CORE QUEUE SUBMIT-TIME START-TIME TIME-REQ JOBNAME
```

It also has an option `-f` that expands a system queue name into all the others in the same family as well, so you can see the pauper and prince queues all in one listing. The `-u` option just shows your jobs, so I tend to use:

```
showq -u -l -o
```

What resources is my job using?

There are a number of ways, other than inspecting the output logs, to see what resources a job is using.

The standard method is to ask SLURM using the commands `sstat` or `sacct`.

Use `sstat` for running jobs and `sacct` for completed ones. You can use these commands to get the maximum memory used, the MaxRSS value:

```
sstat -j 1266635 -o MaxRSS
```

(this is the maximum resident set size, i.e. the core memory in use per node).

To see what load a node has use the command `sinfo`:

```
sinfo -O cpusload -n m7142
```

There are many more node attributes you can see, of varying interest.

To make things easier the loads and memory use of a job can be seen using the local command:

```
c5jobload -l <jobid>

# JOB: <jobid>
NODELIST      CPU_LOAD   MEMORY   ALLOCMEM  FREE_MEM
m5225         16.01      127000   127000    94758
m5226         16.01      127000   127000    95157
m5228         16.01      127000   127000    94692
m5229         16.01      127000   127000    94753
m5250         16.01      127000   127000    94928
m5251         16.01      127000   127000    93447
m5265         16.03      127000   127000    91375
m5266         16.03      127000   127000    95423
m5267         16.02      127000   127000    95533
m5272         16.02      127000   127000    94214
m5273         16.06      127000   127000    94635
```

In this case the job is correctly balanced (for normal MPI) as each node has a load of 16. It is also not using a lot of memory. There `c6jobload` and `c7jobload` variations.

Interactive sessions

On rare occasions it can be useful to run interactively on the compute nodes, but you are not allowed to login (unless you have special privilege). This can be done by submitting an interactive job to the queues and waiting for the job to be scheduled.

```
srun -p cosma -A durham -t 1:00:00 --pty bash -l
```

That will login you into a compute node where you can run and monitor your job interactively. Note that time is charged for this just as for a normal batch job, it is also possible to use more than one node and run MPI jobs, just type the `mpirun` command and the job will launch as if from a submission script.

The end