Slurm at UPPMAX
How to submit jobs with our queueing system
Jessica Nettelblad
sysadmin at UPPMAX

Open source!
https://github.com/SchedM/slurm

Free!

Watch!
Futurama S2 Ep.4
Fry and the Slurm factory

Simple Linux Utility for Resource Management

Popular!
Used at many universities all over the world
Slurm at UPPMAX

1. Queueing
   - Running
   - Analyzing
2. Monitoring
3. Testing
4. Scripting

Queuing
Submit jobs to Slurm
Access the fancy nodes!

Login nodes

Compute nodes

Slurm

Hey Slurm!!
I've got a job for you!

Which job? This one!

sbatch myjob.sh
Hey Slurm!!
I’ve got a job for you!

`sbatch -A g2018014 -t 10 -p core -n 1 myjob.sh`

Which job? This one!

Flags with extra info!

-A as in project name

`sbatch -A g2018014 -t 10 -p core -n 1 myjob.sh`

- Default: None
- Typical: snic2017-9-99
- Example: -A g2018014

What is my project name??
- https://supr.snic.se/
- Find with projinfo
-t as in time

sbatch -A g2018014 -t 10 -p core -n 1 myjob.sh

10 minutes is enough this time!

- What’s a good time limit?
  - Default: 01:00 (1 minute)
  - Typical: Varies
  - Max: 10-0 (10 days)

- What's a good time limit?
  - Estimate! Add 50%
  - Testing
  - Colleagues

dd-hh:mm:ss

-t as in time: Examples

sbatch -A g2018014 -t 10 -p core -n 1 myjob.sh

- 00:00:02 00:00:02 00:02
- 00:10:00 00:10:00 10:00 10
- 12:00:00 12:00:00
- 00:00:00 3-0
- 12:10:00

dd-hh:mm:ss
-t as in time: Examples

```
sbatch -A g2018014 -t 10 -p core -n 1 myjob.sh
```

- `0:00:00:02` 1 minute!
- `0:00:10:00` 10 minutes
- `0:12:00:00` 12 hours
- `3:00:00:00` 3 days
- `3:12:10:00` 3 days, 12 hours, 10 minutes

### Partition and tasks

```bash
sbatch -A g2018014 -t 10 -p core -n 1 myjob.sh
```

- **Default:** core
- **Typical:** core
- **Options:** core, node, devcore, devel

- **Which partition?**
  - <20 cores: core
  - >20 cores: node
  - Short jobs: devcore/core
Partition and **tasks**

```bash
sbatch -A g2018014 -t 10 -p core -n 1 myjob.sh
```

- Default: 1
- Typical: 1
- Max: 20 for core partition

- How many cores do I need?
  - 1 – jobs without parallelism
  - More – jobs with parallelism or high memory usage

---

Partition and **tasks - memory**

```bash
sbatch -A g2018014 -t 10 -p core -n 10 myjob.sh
```

- Request more cores to get more memory
- One node is 128GB and has 20 cores
- One core has 6.4GB

Do you need 64GB of memory?
10 cores * 6.4GB/core = 64GB
Partition and tasks – parallel

```
sbatch -A g2018014 -t 10 -p core -n 8 myjob.sh
```

- Parallel job = more than one core
  - Ask Slurm for cores: `-p core -n 8`
  - Instruct program to use all cores:

```
/…/
bwa aln -t 8 refseqpath resultpath
/…/
```

Flags in the script

```
sbatch -A g2018014 -t 10 -p core -n 1 myjob.sh
```

cat myjob.sh

```
#!/bin/bash
#SBATCH -A g2018014
#SBATCH -p core
#SBATCH -n 1
#SBATCH -t 10:00
start
Do this
Do that
end
```
Flags in script - override

```
#!/bin/bash
#SBATCH --A g2018014
#SBATCH --p core
#SBATCH --n 1
#SBATCH --t 10:00
```

Start
Do something
Done
End

- Typical use
- More static
- Dynamic use
- Changing from job to job
- Overrides flags in comments

More flags

- Job name
  - `-J testjob`
- Qos
  - `--qos=short`
  - `--qos=interact`
- Email notifications
  - `--mail-type=FAIL`
  - `--mail-type=TIME_LIMIT_80`
  - `--mail-type=ALL`
  - `--mail-user=jessica.nettelblad@it.uu.se`
- Output redirections
  - Default: work directory
  - `--output=/proj/g2018014/nobackup/private/jessine/testjob/output/`
  - `--error=/proj/g2018014/nobackup/private/jessine/testjob/output/`
More flags

- Features – memory
  - C thin / C 128GB
  - C fat / C 256GB, C 1TB
- Dependencies
  - --dependency
- Job array
  - --array
- Noder
  - -w r[100] for submit to Rackham node number 100.
- Set working directory
  - --cd
- Time flags
  - --begin, --deadline, --immediate, --time-min

https://slurm.schedmd.com/sbatch.html
- Most, but not all options are available at every center
Monitoring

Keep track of the job
- In queue
- While running
- When finished

In queue - jobinfo

- jobinfo
  - Shows all jobs in queue. Modified squeue.
  - https://slurm.schedmd.com/squeue.html

- How many jobs are running?
  - jobinfo | less
  - Type q to exit

- When are my jobs estimated to start?
  - jobinfo -u jessine

- How about all jobs in the same project?
  - jobinfo |grep g2018014
Priority - starting

---qos
Elevated 200 000
Normal 190 000

Normal 100 000

Bonus!

---qos
Elevated 200 000
Normal 190 000

70 000
60 000
50 000
40 000
30 000
20 000
10 000
Bonus jobs

- Project quota
  - Most jobs have 2000 core hours/month
  - projinfo
  - https://supr.snic.se

- Running out of core hours? No problems!
  - No limit
  - Just lower priority

- Slurm counts 30 days back
  - In 30 days, all your hard work is forgotten

---

Bonus? No problem!

- --qos
- Elevated
- Normal
- Bonus!
- 100 000

- Make the most of our resources!
- Without disturbing others too much
Aging in the queue

--qos
100 000
200 000
00:00

Normal
100 000

Elevated
190 000

Bonus!

00:00

Aging in the queue

--qos
100 005
200 005
00:05

Normal
100 005

Elevated
190 005

Bonus!

00:05
Aging in the queue

--qos
Elevated
Normal
Bonus!

00:10

00:15
How long has it been waiting??

100065?

--qos

Elevated

Normal

100 000

Waited 65 minutes.
100 000 = normal job

Bonus!

How has it been waiting??

190030?

--qos

Elevated

Normal

100 000

Waited 30 minutes.
>100 000 = elevated job

Bonus!
How has it been waiting??

70015?

- Normal
- Elevated
- Bonus!

Waited 15 minutes.
<100 000 = bonus job

Priority – more info

- FAQ
Monitoring

Keep track of the job
- In queue
- While running
- When finished

Check progress

- `./uppmx_jobstats -raw table`
- `less /sw/share/slurm/rackham/uppmx_jobstats/*<job id>`
- Shows memory and core usage
- Every 5 minutes

- `jobstats`
  - Tool based on uppmx_jobstats
  - Plot: jobstats -p
  - `scontrol show job <job id>`

- Output file
  - `tail -f` (on result file)

- ssh to the compute node
  - `top`
  - `htop`
Monitoring

- Keep track of the job
  - In queue
  - While running
  - When finished

Check finished job

- `slurm.out` / `error.out` / `custom name`
- Check it for every job
- Look for error messages
- `upmax_jobstats`
- `finishedjobinfo -s today`
Jobstats exercise

Log in (my username is jessine)
ssh _X jessine@rackham.uu.se

Move to one of your folders
cd /home/jessine/testscripts/g2018014

Look at the file jobids.txt
cat /proj/g2018014/labs/jobids.txt

Run jobstats for those job ids
jobstats -p 1803863 <job id> <job id>

Show the resulting plot
eog rackham-g2018014-marcusl-1803863.png &

Testing
Test using the
- interactive command
- dev partition
- fast lane
Testing in interactive mode

- interactive instead of sbatch
- All sbatch options work
- No script needed
- interactive -A g2018014 -t 15:00

Example:
- A job script didn’t work. I start an interactive job and submit line for line.

Testing in devel partition

- p develcore -n 1 -t 15:00
  - Typical: 1 devel core for 15 minutes
  - Max: 60 minutes, 1 node (20 cores on Rackham), 1 job submitted.

- p devel -n 1 -t 15:00
  - Typical: 1 devel node for 15 minutes
  - Max: 60 minutes, 2 nodes, 1 job submitted.

Job starts quickly!

Example:
- I have a job I want to submit. But to make sure it’s actually fit to run, I first submit it to devcore and let it run for 15 minutes. I monitor the job output.
- Option: Run a simplified version of the program, or time a specific step.
Testing in a fast lane

- **--qos=short**
  - Max: 15 minutes, four nodes, 2 jobs running, 10 jobs submitted

- **--qos=interact**
  - Max: 12 hours, one node, 1 job running

**Example:**
- I have a job that is shorter than 15 minutes. I add qos short, and my job get super high priority, even if I’ve run out of core hours in my project so that my project is in bonus.
Script example

```bash
#!/bin/bash
#SBATCH -A g2016011
#SBATCH -p core
#SBATCH -n 1
#SBATCH -t 10:00:00
#SBATCH -J day3
module load bioinfo-tools samtools/0.1.19 bwa/
export SRCDIR=$HOME/baz/run3
cp $SRCDIR/foo.pl $SRCDIR/bar.txt $SNIC_TMP/
cp $SNIC_TMP/.
/foo.pl bar.txt
cp *.out $SRCDIR/out2
```

Script example explained

- `#!/bin/bash` - starts the bash interpreter
- `#SBATCH` - starts a comment that bash ignores
- `#SBATCH` is a special signal to SLURM
- `#SBATCH` is a special signal to SLURM
- `-A` specifies which account = project will be "charged".
- `#SBATCH -p core` - sets the partition to core, for jobs that uses less than one node.
- `#SBATCH -n 1` - requests one task = one core
Script example explained

- #SBATCH -t 10:00:00
  - Time requested: 10 hours.
- #SBATCH –J day3
  - day3 is the name for this job
  - mainly for your convenience
- module load bioinfo-tools samtools/0.1.19 bwa
  - bioinfo-tools, samtools version 0.1.19 and bwa is loaded.
  - can specify versions or use default (risky)
- export SRCDIR=$HOME/run3
  - Environment variable SRCDIR is defined
  - Used for this job only (as other variables)
  - Inherited by process started by this job (unlike other variables)

- cp $SRCDIR/foo.pl $SRCDIR/bar.txt $SNIC_TMP/
- cd $SNIC_TMP
  - Copy foo.pl and bar.txt to $SNIC_TMP then go there.
  - $SNIC_TMP is a job specific directory on the compute nodes.
  - Recommended! Can be much faster than home.
- ./foo.pl bar.txt
  - Actual script with code to do something.
  - Call one command, or a long list of actions with if-then, etc.
- cp *.out $SRCDIR/out2
  - $SNIC_TMP is a temporary folder. It’s deleted when job is finished.
  - Remember to copy back any results you need!
Group commands - principle

```bash
#!/bin/bash
#SBATCH -A g2018014
#SBATCH -p core
#SBATCH -n 4
#SBATCH -t 2-00:00:00
#SBATCH -J 4commands

while.sh &
while.sh &
while.sh &
while.sh &
wait
```

Group commands - explained

- `while.sh &
while.sh &
while.sh &
while.sh &
wait`
  & means don’t wait until while.sh has finished, go ahead with next line. This way four parallel tasks are started.

- `wait`
  When one task has finished, the script still has to wait until all of the tasks are finished.
Spawn jobs

#!/bin/bash
for v in {1..5}
do
  sbatch myscript.sh $v
Done

In myscript.sh:
#SBATCH -A g2018014
#SBATCH –p core
#SBATCH –n 4
#SBATCH –t 2-00:00:00
#SBATCH -J spawning

Spawn jobs - explained

for v in {1..5}
do
  sbatch myscript.sh $v
Done

Loops from 1 to 5. Meaning it will start five myscript.sh with different input arguments:
sbatch myscript.sh 1
sbatch myscript.sh 2
sbatch myscript.sh 3
sbatch myscript.sh 4
sbatch myscript.sh 5

myscript.sh has to have necessary flags defined, either in myscript.sh
#SBATCH -A g2018014
#SBATCH –p core
#SBATCH –n 4
#SBATCH –t 2-00:00:00
#SBATCH -J spawning

... or add them to the sbatch command:
sbatch -A g2018014 -p core -n 4 -t 2-00:00:00 -J spawning myscript.sh $v
We’re here to help!

- If you run into problems after this course? Just ask someone for help!
- Check userguides and FAQ on uppmax.uu.se
- Ask your colleagues
- Ask UPPMAX support: support@uppmax.uu.se