Slurm Job Scheduler

Basic Slurm Usage

In a Linux cluster there are hundreds of computing nodes inter-connected by high speed networks. The resources are shared among many users for their technical or scientific computing purposes. Users submit jobs, which compete for computing resources. The Slurm software system is a resource manager and a job scheduler, which is designed to allocate resources and schedule jobs. Slurm is an open source software, with a large user community, and has been installed on many top 500 supercomputers.

This tutorial assumes you have a NYU HPC account, and have connected your computer to our facility. Also assumes you are comfortable with Linux command-line environment. If not, you may apply for an account here, or learn about Linux from Tutorial 1.

A typical interaction with Slurm from an user's point of view is as the following:

- Check cluster status
- Submit jobs
- Check job status
- Cancel a job
- Look at job results

We show briefly below how to do these step by step.

Check cluster status

The `sinfo` command gives information about the cluster status, by default listing all the partitions. Partitions group computing nodes into logical sets, which serves various functionalities such as interactivity, visualization and batch processing.

```
$ sinfo
PARTITION   AVAIL TIMELIMIT NODES STATE  Nodelist
  test       up  5:00    3 idle  c[1-3]
 normal*    up 1-00:00:00  1  mix  c5
 normal*    up 1-00:00:00 27  idle  c[6-32]
```
As we can see in the above box, there are two partitions namely test and normal. The partition marked with an asterisk is the default one. Except the node c5 which is in the state 'mix', which means some CPU cores occupied, all other nodes are idle.

The `squeue` command lists jobs which are in a state of either running, or waiting or completing etc.

```
$ squeue

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000006</td>
<td>normal</td>
<td>jobtest</td>
<td>sd3477</td>
<td>PD</td>
<td>0:00</td>
<td>1</td>
<td>JobHeldUser</td>
</tr>
<tr>
<td>10000007</td>
<td>test</td>
<td>job3</td>
<td>sd3477</td>
<td>R</td>
<td>0:02</td>
<td>1</td>
<td>c3</td>
</tr>
<tr>
<td>10000005</td>
<td>normal</td>
<td>job2</td>
<td>sd3477</td>
<td>R</td>
<td>0:42</td>
<td>1</td>
<td>c9</td>
</tr>
<tr>
<td>10000004</td>
<td>normal</td>
<td>job1</td>
<td>sd3477</td>
<td>R</td>
<td>1:37</td>
<td>1</td>
<td>c5</td>
</tr>
<tr>
<td>10000003</td>
<td>normal</td>
<td>sleep_20</td>
<td>devuser</td>
<td>R</td>
<td>18:07</td>
<td>1</td>
<td>c5</td>
</tr>
</tbody>
</table>
```

In the `squeue` output example above, there is a job which is in ‘R’ meaning running, on the node c3, which is in the partition ‘test’. The job ID is 10000007, and its owner is user ‘sd3477’. All other four jobs are in the partition ‘normal’. One job is in ‘PD’ (pending), which is due to the user’s choice (JobHeldUser). All these jobs are on single node, no jobs span more than one node, with the time column showing time durations jobs have been running to now.

Run ‘man sinfo’ or ‘man squeue’ to see detailed usage information of the commands.

**Submit jobs**

Batch job submission can be accomplished with the command `sbatch`. Like in Torque `qsub`, we create a bash script to describe our job requirement: what resources we need, what softwares and processing we want to run, memory and CPU requested, and where to send job standard output and error etc.

```
$ cat runscript.sh
#!/bin/sh
#SBATCH --verbose
#SBATCH --job-name=myFirstTest
#SBATCH --output=slurm_%j.out
#SBATCH --error=slurm_%j.err
#SBATCH --time=01:00:00
#SBATCH --nodes=1
#SBATCH --mem=1GB
/bin/hostname
/bin/pwd

cd /home/devuserd/test_slurm
module load python/intel/2.7.12
srun python -c 'import sys; print sys.version'
```

In the `runscript.sh` we request for one node for one hour, memory per cpu of 700 MB, name the job ‘myFirstTest’, and specify job standard output and error file names being `slurm_%j.out` and `slurm_%j.err` where %j will be replaced with the job ID. Once the script is ready, we can submit the job to the Slurm controller as the following:

```
$ sbatch runscript.sh
Submitted batch job 10000010
```

This job has been submitted successfully. And as the example box showing, its job ID is 10000010. Usually we should let the scheduler to decide on what nodes to run jobs. In cases there is a need to request a specific set of nodes, use the directive nodelist, e.g. `#SBATCH --nodelist=c9`.

**Check job status**

With the job ID in hand, we can track the job status through its life time. The job first appears in the Slurm queue in the PENDING state. Then when its required resources become available, the job gets priority for its turn to run, and is allocated resources, the job will transit to the RUNNING state. Once the job runs to the end and completes successfully, it goes to the COMPLETED state; otherwise it would be in the FAILED state. Use `squeue -j <jobID>` to check a job status. Run the command `sstat` to display various information of running job/step. Run the command `sacct` to check accounting information of jobs and job steps in the Slurm log or database. There is a ‘--helpformat’ option in the latter two commands to help checking what output columns are available.
$ squeue -j 10000012

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000012</td>
<td>normal</td>
<td>sleep_20</td>
<td>devuserd</td>
<td>CG</td>
<td>2:30</td>
<td>1 c9</td>
<td></td>
</tr>
</tbody>
</table>

$sstat -j 10000012 -o JobID,NodeList,ReqCPUFreq,Pids

<table>
<thead>
<tr>
<th>JobID</th>
<th>NodeList</th>
<th>ReqCPUFreq</th>
<th>Pids</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000012.0</td>
<td>c9</td>
<td>Unknown</td>
<td></td>
</tr>
</tbody>
</table>

$sacct -j 10000012

<table>
<thead>
<tr>
<th>JobID</th>
<th>JobName</th>
<th>Partition</th>
<th>Account</th>
<th>AllocCPUS</th>
<th>State</th>
<th>ExitCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000012</td>
<td>sleep_200+</td>
<td>normal</td>
<td>test_acco+</td>
<td>1</td>
<td>CANCELLED+</td>
<td>0:0</td>
</tr>
</tbody>
</table>

Cancel a job

Things can go wrong, or in a way unexpected. Should you decide to terminate a job before it finishes, `scancel` is the tool to help.

$ squeue

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000016</td>
<td>normal</td>
<td>sleep</td>
<td>deng</td>
<td>R</td>
<td>0:40</td>
<td>1 c27-08</td>
<td></td>
</tr>
</tbody>
</table>

$ scancel 10000016

srn: Force Terminated job 10000016
srn: Job step aborted: Waiting up to 32 seconds for job step to finish.
slurmstepd: error: *** STEP 10000016.0 ON c27-08 CANCELLED AT 2016-12-16T09:59:59 ***
srn: error: c27-08: task 0: Terminated

Look at job results

Job results includes the job execution logs (standard output and error), and of course the output data files if any defined when submitting the job. Log files should be created in the working directory, and output data files in your specified directory. Exam log files with a text viewer or editor, to gain a rough idea of how the execution goes. Open output data files to see exactly what result is generated. Run sacct command to see resource usage statistics. Should you decide that the job needs to be rerun, submit it again with sbatch with a modified version of batch script and/or updated execution configuration. Iteration is one characteristic of a typical data analysis!