Using Job Arrays to submit large number of jobs via SLURM

Introduction

It is sometimes necessary to submit large numbers of jobs, especially serial (single core) jobs for High Throughput Computing (HTC). One way of doing this is by preparing many job submission scripts and submitting them with *sbatch* one at a time. This can be cumbersome even when scripted. Our resource manager supports Job Arrays, a mechanism to submit many jobs with a single job script and single job submission.

Using Job Arrays

When you invoke *sbatch* with a Job Array, you will have access to a Job index number stored in the $SLURM_ARRAY_TASK_ID environment variable. This allows a single job script to be used for multiple jobs. Here is a sample script:

array.slurm

```
#!/bin/bash
#SBATCH --job-name=JobNameHere
#SBATCH --partition=PartitionNameHere
#SBATCH --ntasks=1
#SBATCH --time=01:00:00
#SBATCH --export=ALL
#SBATCH --mail-user=YourEmailAddress
#SBATCH --mail-type=ALL

cd $SHARED_SCRATCH/SomeDir/

/path/to/myProgram < $SHARED_SCRATCH/SomeDir/job.$SLURM_ARRAY_TASK_ID.in
```

This example demonstrates how the input files can be a function of the job index number. To submit the above job using a job array of 100 jobs, for example, you would use *sbatch* as follows.

```
sbatch --array=0-99 array.slurm
```

In this example the $SLURM_ARRAY_TASK_ID value would be 0 through 99 and would result in 100 jobs in the queue waiting to run. The jobs would appear in the queue in a fashion similar to the following example as shown by *squeue*:

```
Mon Jul 21 13:42:21 2014
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1931_10   commons    ARRAY     NETID PD       0:00      2 (Resources)
            1931_8   commons    ARRAY     NETID  R       0:32      2 cn-[0125-0126]
            1931_9   commons    ARRAY     NETID  R       0:32      2 cn-[0127-0128]
```

Therefore, the output will be written to a unique file for each job within the array using the templated SLURM output file names i.e. SLURM-*jobID_TASK_ID*.out.

```
ls -Al
-rw-------  1 NETID GROUPID  747 Jul 21 13:48 array.slurm
-rw-------  1 NETID GROUPID 3087 Jul 21 13:49 slurm-1953.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:50 slurm-1954_0.out
-rw-------  1 NETID GROUPID 3137 Jul 21 13:51 slurm-1954_10.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:50 slurm-1954_1.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:50 slurm-1954_2.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:50 slurm-1954_3.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:50 slurm-1954_4.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:50 slurm-1954_5.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:51 slurm-1954_6.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:51 slurm-1954_7.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:51 slurm-1954_8.out
-rw-------  1 NETID GROUPID 3135 Jul 21 13:51 slurm-1954_9.out
```

Best Practice
Using Job Arrays would imply that your jobs can be submitted in some sort of predictable order with predictable file names for input. You will need to provide some mechanism by which each job within the array will know which input files to use.

It is possible to use Job Arrays with an exotic numbering scheme. For example:

```
sbatch array.slurm --array=0-10,50
```

This job submission would result in 12 jobs (indexed 0 through 10 and a job with an index of 50).

Additionally, the array index can be placed in the SLURM batch script via the ***--array*** parameter.

array.slurm

```
#!/bin/bash
#SBATCH --job-name=JobNameHere
#SBATCH --partition=PartitionNameHere
#SBATCH --ntasks=1
#SBATCH --time=01:00:00
#SBATCH --export=ALL
#SBATCH --mail-user=YourEmailAddress
#SBATCH --mail-type=ALL
#SBATCH --array=0-10,50

cd $SHARED_SCRATCH/SomeDir/

/path/to/myProgram < $SHARED_SCRATCH/SomeDir/job.$SLURM_ARRAY_TASK_ID.in
```

For More Information
For even more examples of job array submissions, please see the man page for sbatch i.e. **man sbatch**

Managing Job Arrays

Removing jobs within an array uses the *scancel* command.

If you want to delete **all** jobs in the array, use *scancel* as follows:

```
scancel JobID
```

JobID in the example above is your JobID number of the array.

If you only want to remove the job indexed by the number 5, then the command would be invoked as:

```
scancel JobID_5
```