Advanced Computing Center for Research and Education

SLURM Job Array Support

Opportunities for using arrays
Their strengths and limitations

Visit us here: www.accre.vanderbilt.edu
Follow us on Twitter: @ACCREVandy
Visit our GitHub Repos: https://github.com/accre

Strengths of Using Job Arrays

Speed of submission

- Up to 30,000 jobs can be created in one or two milliseconds.
- This is orders of magnitudes faster than a bash script.

Ease of management

- The job array can be handled as a whole.
- Individual jobs can be handled independently.
- Number of jobs running may be controlled.

Job dependencies by array or single job

- after, afterok, afterany, afternotok

Limitations of Using Job Arrays

All jobs in the array will request identical resources

- Memory
- Number of nodes
- Number of cpus
- Wall time
- Email notifications belong to the array as a whole, and not the individual jobs in the array.

Typical Use Cases for Job Arrays

1. A single program analyzes multiple data files.

2. A single program must be run repeatedly analyzing a single data file.

3. Multiple programs must be run to analyze a single data file.

Typical Job Script Entries

Submit a job array with index values between 0 and 31

```
#SBATCH --array=0-31
```

Submit a job array with index values of 1, 3, 5 and 7

```
#SBATCH --array=1,3,5,7
```

Submit a job array with index values between 1 and 31 with a step size of 2 (i.e. 1, 3, 5, 7, 9 … 31)

```
#SBATCH --array=1-31:2
```

Typical Job Script Entries

Submit a job array with index values between 0 and 3000, and limit the number of simultaneously running jobs to no more than 50

```
#SBATCH --array=0-3000%50
```

The starting value, ending value, and step value must be integers; and are chosen by the user.

Job ID and Environment Variables

Job array scripts will have two additional environment variable set.

SLURM_ARRAY_JOB_ID
This will be set to the first job ID of the array.

SLURM_ARRAY_TASK_ID
This will be set to the job array index value.

Environment Variables Examples

#SBATCH --array=1-3 will create an array with three jobs. If the sbatch command returns a value of 36, then the environment variables will be set like this

SLURM_JOBID=36
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=1

SLURM_JOBID=37
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=2

SLURM_JOBID=38
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=3

File Names

#SBATCH --array=1-3 (from our previous slide) will also create two variables **%A** and **%a** which may be used to name the files that catch stdin and stdout. So,

#SBATCH --output=slurm-%A_%a.out

will create three files:

slurm-36_1.out
slurm-36_2.out
slurm-36_3.out

Job Arrays and squeue

$ squeue -u mac

```
JOBID     PARTITION  NAME  USER  ST  TIME  NODES NODELIST(REASON)
1080_[5-1024]  debug   tmp     mac     PD  0:00      1        (Resources)
1080_1    debug         tmp     mac     R    0:17      1        vmp512
1080_2    debug         tmp     mac     R    0:16      1        vmp443
1080_3    debug         tmp     mac     R    0:03      1        vmp1010
1080_4    debug         tmp     mac     R    0:03      1        vmp317
```

Slurm has not actually created 1024 jobs and placed them in the queue, but waits for the resources to become available before creating the job and placing it in the queue.

Job Arrays and scancel

An individual job in the array may be killed:

```
scancel 2341_7
```

A subset of the array may be killed:

```
scancel 2341_[8-17]
```

The complete job array may be killed:

```
scancel 2341
```

Job Array Script Example 1

```
#!/bin/bash
#SBATCH –mail-user=johns276@accre.vanderbilt.edu
#SBATCH --mail-type=ALL
#SBATCH --ntasks=1
#SBATCH --time=00:15:00
#SBATCH --mem=1G
#SBATCH --array=1-1002%100
#SBATCH --output=hsv_transform_%A_%a.out

cd /scratch/johns276/slurm/hsv/data

echo "SLURM_JOBID: " $SLURM_JOBID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID

arrayfile=`ls | awk -v line=$SLURM_ARRAY_TASK_ID '{if (NR == line) print $0}'`

../hsv_transform $arrayfile
```

Notes on Job Array Script Example 1

#SBATCH –array=1-1002%100

This line will create 1002 jobs, but it instructs slurm to limit the total number of simultaneously running jobs to 100. This avoids swamping the queue, and shares bursting level with others in the group.

#SBATCH –output=hsv_transform_%A_%a.out

This will create 1002 files to catch stdin, stdout and stderr for each respective job in the array. If the array job ID is 23678, we will fine 1002 files starting with hsv_transform_23678_1.out … hsv_transform_23678_1002.out

echo "SLURM_JOBID: " $SLURM_JOBID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID

“echo” sends its output to stdout, so the values of these three environment variables will be captured in the hsv_tansform_%A_%a.out files.

Notes on Job Array Script Example 1

cd /scratch/johns276/slurm/hsv/data
We move to this directory for doing our work.

arrayfile=`ls | awk -v line=$SLURM_ARRAY_TASK_ID '{if (NR == line) print $0}'`
This uses “awk” to select a single file name from a list created by “ls”, the file name chosen will be the one whose position in the list matches the value of $SLURM_ARRAY_TASK_ID. The file name is stored into “arrayfile”.

../hsv_transform $arrayfile
Transforming the file stored in $arrayfile

Job Array Script Example 2

```
#!/bin/bash
#SBATCH --mail-user=johns276@accre.vanderbilt.edu
#SBATCH --mail-type=ALL
#SBATCH --ntasks=1
#SBATCH --time=00:15:00
#SBATCH --mem=1G
#SBATCH --array=1-15
#SBATCH --output=moby_dick_%A_%a.out

cd /scratch/johns276/slurm/word_freq

echo "SLURM_JOBID: " $SLURM_JOBID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID

./word_freq moby_dick.txt $SLURM_ARRAY_TASK_ID
```

Notes on Job Array Script Example 2

./word_freq moby_dick.txt $SLURM_ARRAY_TASK_ID

word_freq is passed
- A file name: moby_dict.txt
- An integer: **$SLURM_ARRAY_TASK_ID**

So the command line for the jobs will iterate over the task_id's

./word_freq moby_dick.txt **1**
./word_freq moby_dick.txt **2**
./word_freq moby_dick.txt **3**
.
.
.
./word_freq moby_dick.txt **15**

Job Array Script Example 3

```
#!/bin/bash
#SBATCH --mail-user=johns276@accre.vanderbilt.edu
#SBATCH --mail-type=ALL
#SBATCH --ntasks=1
#SBATCH --time=00:15:00
#SBATCH --mem=1G
#SBATCH --array=1-26
#SBATCH --output=moby_dick_%A_%a.out

echo "SLURM_JOBID: " $SLURM_JOBID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID

arrayfile=`ls programs/ | awk -v line=$SLURM_ARRAY_TASK_ID '{if (NR == line) print $0}'`

programs/$arrayfile moby_dick.doc
```

Notes on Job Array Script Example 3

As in Example 1, awk is used to select files one at a time, but this time the files are all programs:

```
arrayfile=`ls programs/ | awk -v line=$SLURM_ARRAY_TASK_ID '{if (NR == line) print $0}'`
```

The selected program is then given a file to “analyze”:

```
programs/$arrayfile moby_dick.doc
```