Using Job Arrays on Hydra (slides)

In the intro portion of the workshop you will learn:

- What are jobs arrays, when and why use them?
- How to write job arrays scripts.
- How to submit jobs arrays: task range, increment and limit concurrent tasks.
- Job arrays tips and tricks.
- How to write parallel job arrays.
- How to consolidate small tasks in job arrays.
- How to manage job arrays: `qstat[+]`, `qdel`, `qacct[+]`.
- HPC wiki on job arrays:
  https://confluence.si.edu/display/HPC/Job+Arrays

What are jobs arrays, when and why use them?

- Job arrays allow you to run the same job file multiple times in a single job submission.
    - job arrays have each a unique job id and multiple task ids
- They are typically for running a given analysis on different input files or parameters.
- They allow you to use the same job file and a single `qsub` to run a type of analysis instead of writing a myriad of very similar job files.

How to write job arrays scripts: a trivial example

- Job array scripts, or job files, are like any other job file, except that
  - they use the task identifier stored in the variable `SGE_TASK_ID`
- A trivial example:

```
echo + `date` $JOB_NAME started on $HOSTNAME in $QUEUE
echo jobID=$JOB_ID and taskID=$SGE_TASK_ID
#
model < model.$SGE_TASK_ID.inp
#
echo = `date` $JOB_NAME for taskID=$SGE_TASK_ID done.
```

> run `model` using the input file `model.N.inp`

How to submit jobs arrays: task range, increment and limit concurrent tasks.

- That trivial example can be queued on 100 tasks with

`qsub -t 1-100 trivial_example.job`
: *queues one job with 100 tasks, the equivalent of 100 job files with*

`model < model.1.inp`

in `test1.job`

`model < model.2.inp`

in `test2.job`, etc. . . , up to

`model < model.100.inp`

in `test100.job` - hence one job file instead of 100.
: *assumes you have 100 input files called `model.1.inp`, `model.2.inp`, . . . , `model.100.inp`*

Job arrays tips and tricks.

- A more complete job array file - csh syntax

```
# /bin/csh
#
#$ -N model-1k -cwd -j y -o model.$TASK_ID.log
#$ -t 1-1000 -tc 100
#
echo + `date` $JOB_NAME started on $HOSTNAME in $QUEUE wit
#
set INPUT  = model.$SGE_TASK_ID.inp
set OUTPUT = model.$SGE_TASK_ID.out
./model < $INPUT > $OUTPUT
#
echo = `date` $JOB_NAME for taskID=$SGE_TASK_ID done.
```

Note: * Task range and max concurrent task embedded in the script * Different log file and output file for each task * :attention:

Job arrays tips and tricks (cont'd).

- The same more complete job array file - `sh` syntax

```
# /bin/csh
#
#$ -N model-1k -cwd -j y -o model.$TASK_ID.log
#$ -t 1-1000 -tc 100
#
echo + `date` $JOB_NAME started on $HOSTNAME in $QUEUE wit
#
INPUT=model.$SGE_TASK_ID.inp
OUTPUT=model.$SGE_TASK_ID.out
./model < $INPUT > $OUTPUT
#
echo = `date` $JOB_NAME for taskID=$SGE_TASK_ID done.
```

Job arrays tips and tricks (cont'd).

- Converting the task id $SGE_TASK_ID

1. formatting, replacing 1,2,. . . ,100 by 001,002,. . . ,100

csh syntax

```
@ i = $SGE_TASK_ID
set I = `echo $i | awk '{printf "%3.3d", $1}'`
```

or sh syntax

```
let i=$SGE_TASK_ID
I=$(echo $i | awk '{printf "%3.3d", $1}')
```

2. using `awk` to extract parameters from a single file

csh syntax

```
@ i = $SGE_TASK_ID
set P = (`awk "NR==$i" parameters-list.txt`)
```

or sh syntax

```
let i=$SGE_TASK_ID
P=$(awk "NR==$i" parameters-list.txt)
```

- the variable `P` will hold the content of the `i`-th line of `parameters-list.txt`, and can be used as:

```
./compute $P
```

assuming `compute` takes parameters.

3. using sed and a template

```
@ i = $SGE_TASK_ID                          // let i=$SGE_TA
sed "s/XXX/$i/" input-template.inp > model.$i.inp
model < model.$i.inp > model.$i.out
```

4. using bc to run models on temperatures starting at 23.72 and by 2.43 increments

```
@ i = $SGE_TASK_ID                          // let i=$SGE_T
set tp = `echo "23.72 + $i*2.43" | bc`      // tp=$(echo "2
sed "s/TP/$tp/" input-template.inp > model.$i.inp
model < model.$i.inp > model.$i.out
```

5. using your own tool, `mytool`, to convert a task id to parameters

```
@ i = $SGE_TASK_ID                  // i=$SGE_TASK_ID
set P = (`./mytool $i`)             // P=`./mytool $i`
```

6. using the <<EOF construct

```
@ i = $SGE_TASK_ID                              // let i=$SGE_T
set tp = `echo "23.72 + $i*2.43" | bc`          // tp=$(echo "2
model <<EOF > model.$.out
$tp
EOF
```

How to consolidate small tasks in job arrays.

- Each task is started like a job, hence has the same overhead as starting one job
- Users should avoid running lots of very short tasks ($<$ 10-30m)
- It is relatively easy to consolidate short tasks into longer ones, using the task increment:
  - `qsub -t 200-500:20` will run tasks with id=200,220,240,. . . ,500
- Examples of job files that consolidate short tasks
- csh syntax

```
@ iFr = $SGE_TASK_ID
@ iTo = $iFr + $SGE_TASK_STEPSIZE - 1
if ($iTo > $SGE_TASK_LAST) @ iTo = $SGE_TASK_LAST
#
echo running model.csh for taskIDs $iFr to $iTo
@ i = $iFr
while ($i <= $iTo)
```

Parallel job arrays.

- Job arrays can be parallel jobs/tasks
- Each task requests a parallel environment, as per the `-pe` specification
    - `-pe mthread N` for multi-threaded
    - `-pe mpich N` or `-pe orte N` for MPI
- Check the HPC wiki for more info

How to manage job arrays: `qstat[+]`, `qdel`, `qacct[+]`.

- job status with `qstat` or `qstat+`
- job deletion with `qdel`
- job accounting with `qacct` or `qacct+`

Also remember

- separate name spaces
- avoid emails (`-m abe`)
- test on a small set of tasks

HPC wiki on job arrays

- https://confluence.si.edu/display/HPC/Job+Arrays

Hands on portion