Converting Your (Simple) Job Scripts from PBS to SLURM on \textit{discover}

NASA Center for Climate Simulation
\textit{High Performance Science}

July 31, 2014
Introduction

• **Portable Batch System**
  – Developed at Ames for NASA
  – Commercial version: PBS Pro (Altair Engineering)

• **Simple Linux Utility for Resource Management**
  – Developed at LLNL
  – Open-source (supported by SchedMD)

• **PBS->SLURM on discover in October 2013.**
What’s the difference?

• Concepts and commands have new names.
• Overall script design remains essentially the same.
• A PBS “queue” is equivalent to a SLURM “partition”.

NCCS Brown Bag 7/31/2014

NASA Center for Climate Simulation
Why did we switch?

- Quality of Service (QoS)
  - Eliminates need for dedicated queues
- Great reduction in cost
- But PBS is still used at NAS….
  - … so we use a PBS emulation layer.
PBS emulation with SLURM

- SchedMD provided wrapper scripts (in Perl).
- We modified the wrappers for discover.
- Most changes were folded back into baseline.
- Wrapped tools: `qsub`, `qalter`, `qdel`, `qhold`, `qrerun`, `qrls`, `qstat`, `xsub`
- Wrappers handle command-line options only.
- `#PBS script directives are translated to #SBATCH and processed by sbatch.`
Emulation “gotchas”

• Not all PBS features can be emulated.
• SLURM exports user environment by default.
• SLURM runs in the current directory.
• SLURM combines stdout and stderr.
Batch job submission

• For simple cases, just replace `qsub` with `sbatch`.

```
$ qsub myjob.sh
```

becomes

```
$ sbatch myjob.sh
```
Naming your job

- Naming the job makes it easier to find.

```
#PBS -N job_name
```
becomes

```
#SBATCH -J job_name
```
or

```
#SBATCH --job-name=job_name
```
Specifying the account

• Make sure the proper account is charged.

```
#PBS -A account_name
```
becomes
```
#SBATCH -A account_name
```

or
```
#SBATCH --account=account_name
```
Specifying the partition

• Only if you *have to* ....

```
#PBS -q destination
```

becomes
```
#SBATCH -p destination
```
or
```
#SBATCH --partition=destination
```
Specifying the number of nodes

• Specify how many nodes you need.

#PBS -l select=num

becomes

#SBATCH -N num

or

#SBATCH --nodes=num

• A range can also be specified as nmin-nmax.
Specifying processes per node

• Use one process per core on each node by default, but may want less.

#PBS -l mpiprocs=num

becomes

#SBATCH --ntasks-per-node=num
Specifying processor type

- Choices are:
  - Sandy Bridge (sand – 16 cores/node)
  - Westmere (west – 12 cores/node)

#PBS -l proc=proc_type

becomes

#SBATCH --constraint=proc_type

or

#SBATCH -C proc_type
stdout and stderr streams

• Specify where the output streams are written.

```bash
#PBS -o opath -e epath
```

becomes

```bash
#SBATCH --output=opath --error=epath
```

or

```bash
#SBATCH --output=opath --error=epath
```

• Streams are joined in SLURM by default (./slurm-NNNNNNNNN.out), which required -j oe or -j eo in PBS.
Mail notification

- Use to get a message when your job is done, or when something bad happens…

#PBS -M user_list

becomes

#SBATCH --mail-type=type
#SBATCH --mail-user=user

- Type can be BEGIN, END, FAIL, ALL (any state change).
- Default user is the submitter.
Your working directory

• PBS jobs ran in a spool directory.
• SLURM jobs run in the current directory.
• Can be changed with `cd` command, or:
  
  #SBATCH -D path
  
  or
  
  #SBATCH --workdir=path
Exporting environment variables

- PBS exported nothing by default.
- SLURM exports everything by default.
- Change with one or more of:
  
  ```bash
  #SBATCH --export=names
  #SBATCH --export=ALL
  #SBATCH --export=NONE
  #SBATCH --export-file=path
  ```
Threads and MPI

- Set up and run threads as you always have.
- `mpirun/mpiexec/mpiexec.hydra` are not part of PBS, so no changes needed.
- The SLURM tool `srun` provides additional features that are SLURM-specific.
  - Provides features similar to those of MPI tools.
  - Differences in job step control and signal propagation.
A simple example

• User *inigo* has an old PBS script:
  – The job name is *revenge*.
  – Runs in the default PBS queue.
  – Runs the application *sword.x*.
  – Uses 8 Westmere nodes
The simple script (PBS)

```bash
#PBS -N revenge
#PBS -l select=8:proc=west

mpirun sword.x
```
#SBATCH --job-name=revenge
#SBATCH --nodes=8
#SBATCH --constraint=west

mpirun sword.x

# Could also use:
# srun sword.x
The simple results...

• Program runs in current directory, not the spool directory.
• User environment is exported.
• Standard output and standard error together in ./slurm-NNNNNNNNNN.out.
A not-so-simple example

- User westley has an old PBS script:
  - The job name is pirate.
  - Charge the account roberts.
  - Runs in the PBS queue dread.
  - Uses 12 Sandy Bridge nodes.
  - Uses 8 cores per node.
  - Export only the variable BUTTERCUP.
  - Runs the application sword.x.
The NSS Script (PBS)

#PBS -N pirate
#PBS -A roberts
#PBS -q dread
#PBS -l
select=12:proc=sand:mpiprocs=8
#PBS -v BUTTERCUP

mpirun sword.x
The NSS Script (SLURM)

```bash
#SBATCH --job-name=pirate
#SBATCH --account=roberts
#SBATCH --partition=dread
#SBATCH --nodes=12
#SBATCH --constraint=sand
#SBATCH --ntasks-per-node=8
#SBATCH --export=NONE,BUTTERCUP

mpirun sword.x
# or
# srun sword.x
```
The NSS results…

• Program runs in current directory, not the spool directory.
• User environment is exported.
• Standard output and standard error together in ./**slurm-NNNNNNNNN.out**.

NOTE: If you have an environment variable named NONE, and use **--export=NONE**, nothing is exported. But if you have NONE, and use **--export=NONE, OTHER, NONE and OTHER** are exported with everything else! So don’t do that….
Much more to come…

• Using `mpirun/mpiexec/mpiexec.hydra` vs. using `srun`.
  – Differing behavior for signal propagation and job control commands.

• Job dependencies with `strigger`.

• Copying files with `sbcast`.

• Attaching to running jobs with `sattach`.
Questions?