Running Jobs with SLURM (Puma)

Overview

SLURM

The new HPC system, Puma, uses SLURM as a job scheduler rather than PBS Pro. SLURM has several advantages:

- It provides more robust support for a larger number of jobs in queue.
- It is used by national HPC groups (XSEDE and TACC) making it easier for users to scale out to those systems.
- It has more sustainable support.

Allocations and Job Partitions (Queues)

Using Puma with SLURM is similar to using ElGato and Ocelote with PBS. Users will still receive a monthly allocation of cpu hours associated with their PI's group which will be deducted when they run their jobs in standard. Users will also still be able to use windfall to run jobs without consuming their monthly allocations. As on Ocelote and Puma, jobs run using windfall will still be subject to preemption when resources are requested by higher-priority jobs.

To request a specific partition (standard, windfall, or high_priority), see Job Partition Requests below.

Modules and Software

The process of finding, loading, and using software as modules will not change on the new system. Users will still be able to utilize the standard commands described in the Software section in our User Guide. However, in a departure from our previous systems, modules will not be available to load and utilize on the login nodes. To load, use, and test software for job submissions, users will need to request an interactive session. Interactive sessions may be requested by simply using the command "interactive".

Interactive command

When you are on a login node, you can enter interactive and that will provide a default session on a compute node. This is useful for checking available modules, testing submission scripts, compiling software for example.

The interactive command actually does this:

```
srun --nodes=1 --ntasks=1 --ntasks-per-node=1 --mem=per-cpu=4GB --time=01:00:00 --job-name=interactive --account=windfall --pty bash -i
```

If you find that this session is insufficient, then run the command above, but change what you need. Perhaps you need 2 hours, not 1 hour.
GPU Jobs

To run a job on a GPU you request the resource name that is assigned to the GPU. So in this example interactive job:

```
srun --nodes=1 --ntasks-per-node=1 --mem-per-cpu=1GB --time=01:00:00 --job-name=interactive --account=windfall --gres=gpu:1  --pty bash -i
```

The resource is gpu and the quantity is 1.

PBS  SLURM Rosetta Stone

In general, SLURM can translate and execute scripts written for PBS. This means that if you submit a PBS script written for Ocelote or ElGato on Puma (with the necessary resource request modifications), your script will likely run. However, there are a few caveats that should be noted:

- You will need to submit your job with the new SLURM command, e.g. sbatch instead of qsub
- There may be some PBS directives that do not directly translate to SLURM which cannot be interpreted
- The environment variables specific to PBS and SLURM are different. If your job relies on these, you will need to update them. Common examples are PBS_O_WORKDIR and PBS_ARRAY_INDEX

To help with the transition to SLURM, we've also installed software that converts some basic PBS Pro commands into SLURM commands automatically called pbs2slurm.

To get acquainted with the new scheduling system, refer to the following list of common PBS commands, directives, and environment variables and their SLURM counterparts.

<table>
<thead>
<tr>
<th>PBS</th>
<th>SLURM</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>qsub &lt;options&gt;</td>
<td>sbatch &lt;options&gt;</td>
<td>Batch submission of jobs to run without user input</td>
</tr>
<tr>
<td>qsub -I &lt;options&gt; (note the upper case 'I')</td>
<td>srun &lt;options&gt; --pty bash -i</td>
<td>Request an interactive job</td>
</tr>
<tr>
<td>qstat</td>
<td>squeue</td>
<td>Show all jobs</td>
</tr>
<tr>
<td>qstat &lt;jobid&gt;</td>
<td>squeue --job &lt;jobid&gt;</td>
<td>Check status of a specific job</td>
</tr>
<tr>
<td>qstat-u &lt;netid&gt;</td>
<td>squeue -u &lt;netid&gt;</td>
<td>Check status of jobs specific to user</td>
</tr>
<tr>
<td>tracejob &lt;jobid&gt;</td>
<td>sacct -j &lt;jobid&gt;</td>
<td>Check history of a completed job</td>
</tr>
<tr>
<td>qdel &lt;jobid&gt;</td>
<td>scancel &lt;jobid&gt;</td>
<td>Delete a specific job</td>
</tr>
<tr>
<td>qdel-u &lt;netid&gt;</td>
<td>scancel -u &lt;netid&gt;</td>
<td>Delete all user jobs</td>
</tr>
<tr>
<td>qstat-Q</td>
<td>sinfo</td>
<td>View information about nodes and queues</td>
</tr>
<tr>
<td>qhold &lt;jobid&gt;</td>
<td>scontrol hold &lt;jobid&gt;</td>
<td>Places a hold on a job to prevent it from being executed</td>
</tr>
<tr>
<td>qrls &lt;jobid&gt;</td>
<td>scontrol release &lt;jobid&gt;</td>
<td>Releases a hold placed on a job allowing it to be executed</td>
</tr>
</tbody>
</table>

### Job Directives

<table>
<thead>
<tr>
<th>PBS</th>
<th>SLURM</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -W group_list=group_name</td>
<td>#SBATCH --account=group_name</td>
<td>Specify group name where hours are charged</td>
</tr>
<tr>
<td>#PBS -q standard</td>
<td>#SBATCH --partition=standard</td>
<td>Set job queue</td>
</tr>
<tr>
<td>#PBS -l walltime=HH:MM:SS</td>
<td>#SBATCH --time HH:MM:SS</td>
<td>Set job walltime</td>
</tr>
<tr>
<td>#PBS -l select=&lt;N&gt;</td>
<td>#SBATCH --nodes=&lt;N&gt;</td>
<td>Select N nodes</td>
</tr>
</tbody>
</table>
#PBS -l ncpus=<N>  #PBS: Select N cpus
#SBATCH --ntasks=<N>  #SLURM: Each task assume to require one cpu. Optionally, you may include cpus-per-task if more are required. Requests NxM cpus
Note: Puma has 94 cpus available on each node

#PBS -l mem=<N>gb  #PBS: Select N gb of memory
#SBATCH --mem=<N>gb  
Note: Puma defaults to 5GB per cpu

#PBS -l pcmem=<N>gb  #PBS: Select N gb of memory per cpu
#SBATCH --mem-per-cpu=<N>gb  
Note: Puma defaults to 5GB per cpu

#PBS J N-M  #PBS: Array job submissions where N and M are integers
#SBATCH --array=N-M  
Optional: Request a GPU

#PBS -N JobName  #PBS: Optional: Set job name
#SBATCH --job-name=JobName  
Optional: Combine stdout and error

#PBS -j oe  (default)  Optional: Combine stdout and error
#SBATCH --array=N-M  (default)  Optional: Separate stdout and stderr
(SLURM: %j is a stand-in for $SLURM_JOB_ID)

#PBS -o filename  #PBS: Optional: Standard output filename
#SBATCH --output=filename  
Optional: Error filename

#PBS -e filename  #PBS: Optional: Error filename
#SBATCH --error=filename  
Optional: Combine all output into single file. Note: If this is selected, each job run will append to that filename, including preexisting files with that name

#PBS -v var=<value>  #PBS: Optional: Export single environment variable var to job
#SBATCH --export=var  
Optional: Export all environment variables to job
#SBATCH --export=none  
Optional: Do not export working environment to job

#PBS -m be  #PBS: Optional: Request email notifications
#SBATCH --mail-type=BEGIN|END|FAIL|ALL  
Beware of mail bombing yourself

#PBS -M <netid>@email.arizona.edu  #PBS: Optional: email address used for notifications
#SBATCH --mail-user=<netid>@email.arizona.edu  
Optional: Request exclusive access to node

#PBS -l place=excl

## Environment Variables

$PBS_O_WORKDIR $SLURM_SUBMIT_DIR  Job submission directory
$PBS_JOBID $SLURM_JOB_ID  Job ID
$PBS_JOBNAME $SLURM_JOB_NAME  Job name
$PBS_ARRAY_INDEX $SLURM_ARRAY_TASK_ID  Index to differentiate tasks in an array
$PBS_Q_HOST $SLURM_SUBMIT_HOST  Hostname where job was submitted
$PBS_NODEFILE $SLURM_JOB_NODELIST  List of nodes allocated to current job

## Terminology

Queue  Partition
Group List  Association
PI  Account

Job Partition Requests
SLURM partition requests are slightly different from PBS. Use the following table as a guide for how to use the partition that is relevant to you:

<table>
<thead>
<tr>
<th>Partition</th>
<th>SLURM</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>#SBATCH --account=&lt;PI GROUP&gt; #SBATCH --partition=standard</td>
<td>Consumes your group's standard allocation.</td>
</tr>
<tr>
<td>windfall</td>
<td>#SBATCH --partition=windfall</td>
<td>Does not consume your group's standard allocation. Jobs may be interrupted and restarted by high-priority jobs. The --account flag needs to be omitted or an error will occur.</td>
</tr>
<tr>
<td>high_priority</td>
<td>#SBATCH --account=&lt;PI GROUP&gt; #SBATCH --partition=standard #SBATCH --qos=user_qos_&lt;PI GROUP&gt;</td>
<td>Available for groups who have purchased compute resources. The partition flag is left as standard and requires the additional qos flag. Replace &lt;NETID&gt; with your own UArizona NetID.</td>
</tr>
</tbody>
</table>

SLURM Output Filename Patterns

Unlike PBS, SLURM offers ways to make your job's output filenames more customizable through the use of character replacements. A table is provided below as a guide with some examples. Variables may be used or combined as desired. Note: character replacements may also be used with other SBATCH directives such as error filename, input filename, and job name.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
<th>Example Slurm Directive(s)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>%A</td>
<td>A job array's main job ID</td>
<td>#SBATCH --array=1-2 #SBATCH -o %A.out #SBATCH --open-mode=append</td>
<td>12345.out</td>
</tr>
<tr>
<td>a</td>
<td>A job array's index number</td>
<td>#SBATCH --array=1-2 #SBATCH -o %A_%a.out</td>
<td>12345_1.out 12345_2.out</td>
</tr>
<tr>
<td>%J</td>
<td>Job ID plus stepid</td>
<td>#SBATCH -o %J.out</td>
<td>12345.out</td>
</tr>
<tr>
<td>j</td>
<td>Job ID</td>
<td>#SBATCH -o %j.out</td>
<td>12345.out</td>
</tr>
<tr>
<td>%N</td>
<td>Hostname of the first compute node allocated to the job</td>
<td>#SBATCH -o %N.out</td>
<td>r1u11n1.out</td>
</tr>
<tr>
<td>u</td>
<td>Username</td>
<td>#SBATCH -o %u.out</td>
<td>netid.out</td>
</tr>
<tr>
<td>x</td>
<td>Jobname</td>
<td>#SBATCH --job-name=JobName #SBATCH -o %x.out</td>
<td>JobName.out</td>
</tr>
</tbody>
</table>

Job Examples

Single serial job submission

PBS Script
#!/bin/bash

#PBS -N Sample_PBS_Job
#PBS -l select=1:ncpus=1:mem=1gb
#PBS -l walltime=00:01:00
#PBS -q standard
#PBS -W group_list=<group_name>

cd $PBS_O_WORKDIR

pwd; hostname; date

module load python

python --version

---

SLURM Script

#!/bin/bash

#SBATCH --job-name=Sample_Slurm_Job
#SBATCH --ntasks=1
#SBATCH --mem=1gb
#SBATCH --time=00:01:00
#SBATCH --partition=standard
#SBATCH --account=<group_name>

# SLURM Inherits your environment. cd
#$SLURM_SUBMIT_DIR not needed

pwd; hostname; date

module load python

python --version

---

Array Submission

IMPORTANT:

When submitting jobs with named output files (i.e. with the line #SBATCH -o=Job.out) as arrays, SLURM will write every array element to that filename leaving you with only the output of the last completed job in the array. Use one of the following SLURM directives in your script to prevent this behavior:

1. Differentiates output files using array indices. Similar to PBS default. See SLURM Output Filename Patterns above for more information.

#SBATCH --output=Job-%a.out

2. Appends the output from all tasks in an array to the same output file. Warning: if a file exists with that name prior to running your job, the output will be appended to that file.

#SBATCH --open-mode=append

---

PBS Script
#!/bin/bash
#PBS -N Sample_PBS_Job
#PBS -l select=1:ncpus=1:mem=1gb
#PBS -l walltime=00:01:00
#PBS -q standard
#PBS -W group_list=<group_name>
#PBS -J 1-5

cd $PBS_O_WORKDIR
pwd; hostname; date

>./sample_command input_file_${PBS_ARRAY_INDEX}.in

SLURM Script

#!/bin/bash
#SBATCH --output=Sample_SLURM_Job-%a.out
#SBATCH --ntasks=1
#SBATCH --mem=1gb
#SBATCH --time=00:01:00
#SBATCH --partition=standard
#SBATCH --account=<group_name>
#SBATCH --array 1-5

# SLURM Inherits your environment. cd
# $SLURM_SUBMIT_DIR not needed
pwd; hostname; date

>./sample_command
input_file_${SLURM_ARRAY_TASK_ID}.in

MPI Example
For openmpi the important variables are set by default, so you do not need to include them in your scripts.

export SBATCH_GET_USER_ENV=1
export OMPI_MCA_btl_openib_opc_include=rdmacm
export OMPI_MCA_btl_openib_if_include=bnxt_re1
export OMPI_MCA_btl_openib_roce_enable=1
export OMPI_MCA_btl=vader,self,openib
export OMPI_MCA_oob_tcp_if_include=eth1

For Intel MPI, these variables are set for you:

export I_MPI_FABRICS=shm:ofi
export FI_PROVIDER=verbs
export FI_VERBS_IFACE=eth1

PBS Script
#!/bin/bash
#PBS -N Sample_MPI_Job
#PBS -l select=1:ncpus=16:mem=16gb
#PBS -l walltime=00:10:00
#PBS -W group_list=<group_name>
#PBS -q standard

cd $PBS_O_WORKDIR
pwd; hostname; date
module load openmpi
#On ocelote, openmpi does not load by default
/usr/bin/time -o mpit_prog.timing mpirun -np 16 a.out

SLURM Script

#!/bin/bash
#SBATCH --job-name=Sample_MPI_Job
#SBATCH --ntasks=16
#SBATCH --ntasks-per-node=16
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=1gb
#SBATCH --time=00:10:00
#SBATCH --account=<group_name>
#SBATCH --partition=standard
#SBATCH --output=Sample_MPI_Job_%A.out
#SBATCH --error=Sample_MPI_Job_%A.err

# SLURM Inherits your environment. cd
$SLURM_SUBMIT_DIR not needed
cd
pwd; hostname; date
# module load openmpi3 is not necessary. On Puma it
# is loaded by default
/usr/bin/time -o mpit_prog.timing mpirun -np 16 a.out

rosetta_min.pdf