Running Jobs with SLURM

Overview

All three clusters, Puma, Ocelote, and ElGato, use SLURM for resource management and job scheduling.

Additional SLURM Resources and Examples

Link	Description
Official SchedMD User Documentation	Official SchedMD user documentation. Includes detailed information on SLURM directives and commands.
PBS SLURM Rosetta Stone	Table for converting some common PBS job directives to SLURM syntax.
Puma Quick Start	HPC Quick Start guide. If you have never submitted a batch job before, this is a great place to start.
Job Examples	Basic SLURM example scripts. Includes PBS scripts for comparison.
Even More Job Examples!	Growing repository of example SLURM submission scripts
Intro to HPC	A recorded video presentation of our Intro to HPC workshop. Keep your eyes peeled for periodic announcements in the HPC listserv on upcoming live sessions!

SLURM and System Commands

Command	Purpose	Example(s)
	Native Slurm Commands	
`sbatch`	Submits a batch script for execution	`sbatch script.slurm`
`srun`	Run parallel jobs. Can be in place of mpirun/mpiexec. Can be used interactively as well as in batch scripts	`srun -n 1 --mpi=pmi2 a.out`

`salloc`	Requests a session to work on a compute node interactively	see: Interactive Sessions section below

`squeue`	Checks the status of pending and running jobs	`squeue --job $JOBID`
		`squeue --user $NETID`

`scancel`	Cancel a running or pending job	`scancel $JOBID`
		`scancel -u $NETID`

`scontrol hold`	Place a hold on a job to prevent it from being executed	`scontrol hold $JOBID`
`scontrol release`	Releases a hold placed on a job allowing it to be executed	`scontrol release $JOBID`
System Commands		
`va`	Displays your group membership, your account usage, and CPU allocation. Short for "view allocation"	`va`
`interactive`	Shortcut for quickly requesting an interactive job. Use "interactive --help" to get full usage.	interactive -a $GROUP_NAME
`job-history`	Retrieves a running or completed job's history in a user-friendly format	`job-history $JOBID`
`seff`	Retrieves a completed job's memory and CPU efficiency	`seff $JOBID`
`past-jobs`	Retrieves past jobs run by user. Can be used with option "-d N" to search for jobs run in the past N days.	`past-jobs -d 5`
`job-limits`	View your group's job resource limits and current usage.	`job-limits $GROUP`
`nodes-busy`	Display a visualization of nodes on a cluster and their usage	`nodes-busy --help`
`system-busy`	Display a text-based summary of a cluster's usage	`system-busy`
`cluster-busy`	Display a visualization of all three cluster's overall usage	`cluster-busy --help`

Batch Job Directives

Command	Purpose
`#SBATCH --account=group_name`	Specify the account where hours are charged. Don't know your group name? Run the command "va" to see which groups you belong to
`#SBATCH --partition=partition_name`	Set the job partition. This determines your job's priority and the hours charged. See Job Partition Requests below for additional information
`#SBATCH --time=DD-HH:MM:SS`	Set the job's runtime limit in days, hours, minutes, and seconds
`#SBATCH --nodes=N`	Allocate N nodes to your job. For non-MPI enabled jobs, this should be set to "–-nodes=1" to ensure access to all requested resources and prevent memory errors.

`#SBATCH --ntasks=N`
 	
`#SBATCH --cpus-per-task=M`
	ntasks specifies the number of tasks (or processes) the job will run. For MPI jobs, this is the number of MPI processes. Most of the time, you can use ntasks to specify the number of CPUs your job needs. However, in some odd cases you might run into issues. For example, see: Using Matlab
	
	By default, you will be allocated one CPU/task. This can be increased by including the additional directive --cpus-per-task.
	
	The number of CPUs a job is allocated is cpus/task * ntasks, or M*N

`#SBATCH --mem=Ngb`	Select N gb of memory **per node**. If "gb" is not included, this value defaults to MB. Directives --mem and --mem-per-cpu are mutually exclusive.
`#SBATCH --mem-per-cpu=Ngb`	Select N GB of memory per CPU. Valid values can be found in the Node Types/Example Resource Requests section below. If "gb" is not included, this value defaults to MB.
`#SBATCH --gres=gpu:N`	Optional: Request N GPUs.

`#SBATCH --constraint=hi_mem`	Optional: Request a high memory node (Ocelote and Puma only).
`#SBATCH --array=N-M`	Submits an array job from indices N to M
`#SBATCH --job-name=JobName`	Optional: Specify a name for your job. This will not automatically affect the output filename.
`#SBATCH -e output_filename.err` `#SBATCH -o output_filename.out`	Optional: Specify output filename(s). If -e is missing, stdout and stderr will be combined.
`#SBATCH --open-mode=append`	Optional: Append your job's output to the specified output filename(s).
`#SBATCH --mail-type=BEGIN|END|FAIL|ALL`	Optional: Request email notifications. **Beware of mail bombing yourself**.
`#SBATCH --mail-user=email@address.xyz`	Optional: Specify email address. If this is missing, notifications will go to your UArizona email address by default.
`#SBATCH --exclusive`	Optional: Request exclusive access to node.
#SBATCH --export=VAR	Optional: Export a comma-delimited list of environment variables to a job.
`#SBATCH --export=all (default)`	Optional: Export your working environment to your job.
`#SBATCH --export=none`	Optional: Do not export working environment to your job.

Job Partition Requests

Partition	SLURM	Details
standard	`#SBATCH --account=<PI GROUP>` `#SBATCH --partition=standard`	Consumes your group's standard allocation. These jobs cannot be interrupted.
windfall	`#SBATCH --partition=windfall`	Does not consume your group's standard allocation. Jobs may be interrupted and restarted by higher-priority jobs. The `--account` flag needs to be omitted or an error will occur.
high_priority	`#SBATCH --account=<PI GROUP>` `#SBATCH --partition=high_priority` `#SBATCH --qos=user_qos_<PI GROUP>`	Available for groups who have purchased compute resources.
qualified	`#SBATCH --account=<PI GROUP>` `#SBATCH --partition=standard` `#SBATCH --qos=qual_qos_<PI GROUP>`	Available for groups that have submitted a special project request.

SLURM Output Filename Patterns

SLURM offers ways to make your job's output filenames 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.

Variable	Meaning	Example Slurm Directive(s)	Output

%A	A job array's main job ID	`#SBATCH --array=1-2`
		`#SBATCH -o %A.out`
		`#SBATCH --open-mode=append`
			12345.out

%a	A job array's index number	`#SBATCH --array=1-2`
		`#SBATCH -o %A_%a.out`
			12345_1.out
			12345_2.out

%J	Job ID plus stepid	`#SBATCH -o %J.out`	12345.out
%j	Job ID	`#SBATCH -o %j.out`	12345.out
%N	Hostname of the first compute node allocated to the job	`#SBATCH -o %N.out`	r1u11n1.out
%u	Username	`#SBATCH -o %u.out`	netid.out

%x	Jobname	`#SBATCH --job-name=JobName`
		`#SBATCH -o %x.out`
			JobName.out

Node Types/Example Resource Requests

Standard Nodes

Cluster	Max CPUs	Mem/CPU	Max Mem	Sample Request Statement

ElGato	16	4gb	62gb	`#SBATCH --nodes=1`
				`#SBATCH --ntasks=16`
				`#SBATCH --mem-per-cpu=4gb`

Ocelote	28	6gb	168gb	`#SBATCH --nodes=1`
				`#SBATCH --ntasks=28`
				`#SBATCH --mem-per-cpu=6gb`

Puma	94	5gb	470gb	`#SBATCH --nodes=1`
				`#SBATCH --ntasks=94`
				`#SBATCH --mem-per-cpu=5gb`

GPU Nodes

During the quarterly maintenance cycle on April 27, 2022 the ElGato K20s and Ocelote K80s were removed because they are no longer supported by Nvidia.

GPU jobs are requested using the generic resource, or `--gres`, SLURM directive. In general, the directive to request N GPUs will be of the form: `--gres=gpu:N`

Cluster	Max CPUs	Mem/CPU	Max Mem	Sample Request Statement

Ocelote	28	6gb	168gb	`#SBATCH --nodes=1`
				`#SBATCH --ntasks=28`
				`#SBATCH --mem-per-cpu=6gb`
				`#SBATCH --gres=gpu:1`

Puma[1]	94	5gb	470gb	`#SBATCH --nodes=1`
				`#SBATCH --ntasks=94`
				`#SBATCH --mem-per-cpu=5gb`
				`#SBATCH --gres=gpu:1`

1 Up to four GPUs may be requested on Puma on a single GPU node with --gres=gpu:1, 2, 3, or 4				

High Memory Nodes

When requesting a high memory node, include **both** the memory/CPU and constraint directives

Cluster	Max CPUs	Mem/CPU	Max Mem	Sample Request Statement

Ocelote	48	41gb	2015gb	`#SBATCH --nodes=1`
				`#SBATCH --ntasks=48`
				`#SBATCH --mem-per-cpu=41gb`
				`#SBATCH --constraint=hi_mem`

Puma	94	32gb	3000gb	`#SBATCH --nodes=1`
				`#SBATCH --ntasks=94`
				`#SBATCH --mem-per-cpu=32gb`
				`#SBATCH --constraint=hi_mem`

Interactive Jobs

When you are on a login node, you can request an interactive session on a compute node. This is useful for checking available modules, testing submission scripts, compiling software, and running programs directly from the command line. We have a built-in shortcut command that will allow you to quickly and easily request a session by simply entering: **`interactive`**

When you request a session, the full salloc command being executed will be displayed for verification/copying/editing/pasting purposes. For example:

```
(ocelote) [netid@junonia ~]$ interactive
Run "interactive -h for help customizing interactive use"
Submitting with /usr/local/bin/salloc --job-name=interactive --mem-per-cpu=4GB --nodes=1    --ntasks=1 --
time=01:00:00 --account=windfall --partition=windfall
salloc: Pending job allocation 531843
salloc: job 531843 queued and waiting for resources
salloc: job 531843 has been allocated resources
salloc: Granted job allocation 531843
salloc: Waiting for resource configuration
salloc: Nodes i16n1 are ready for job
[netid@i16n1 ~]$
```

Notice in the example above how the command prompt changes once your session starts. When you're on a login node, your prompt will show "junonia" or "wentletrap". Once you're in an interactive session, you'll see the name of the compute node you're connected to.

If no options are supplied to the command `interactive`, your job will automatically run using the windfall partition for one hour using one CPU. To use the standard partition, include the flag "-a" followed by your group's name. To see all the customization options:

```
(ocelote) [netid@junonia ~]$ interactive -h
Usage: /usr/local/bin/interactive [-x] [-g] [-N nodes] [-m memory per core] [-n ncpus per node] [-Q optional
qos] [-t hh::mm:ss] [-a account to charge]
```

You may also create your own `salloc` commands using any desired SLURM directives for maximum customization.

MPI Jobs

OpenMPI

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

Default OpenMPI variables

```
export SBATCH_GET_USER_ENV=1
export OMPI_MCA_btl_openib_cpc_include=rdmacm
export OMPI_MCA_btl_openib_if_include=bnxt_re1
export OMPI_MCA_btl_openib_rroce_enable=1
export OMPI_MCA_btl=vader,self,openib
export OMPI_MCA_oob_tcp_if_include=eth1
```

Intel MPI

For **Intel MPI,** these variables are set for you:

Default Intel MPI variables

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

Because the modules gnu8 and openmpi3 are loaded by default, these should be manually unloaded in your submission script:

```
module unload openmpi3 gnu8
```

If you're using Intel MPI with mpirun and are getting errors, try replacing `mpirun -np $NPROCESSES` with:

```
srun -n $NPROCESSES --mpi=pmi2
```

Parallel Work

To make proper use of a supercomputer, you will likely want to use the benefit of many cores. Puma has 94 cores in each node available to Slurm. The exception to that is running hundreds or thousands of jobs using High Throughput Computing.

We have a training course which explains the concepts and terminology of parallel computing with some examples. Introduction to Parallel Computing

This practical course in Parallel Analysis in R is also useful