Submitting Jobs with SLURM

Once you have an executable program and are ready to run it on the compute nodes, you **must** create a job script that performs the following functions:

- Use job batch options to request the resources that will be needed (i.e. number of processors, run time, etc.), and
- Use commands to prepare for execution of the executable (i.e. cd to working directory, source shell environment files, copy input data to a scratch location, copy needed output off of scratch location, clean up scratch files, etc).

After the job script has been constructed you must submit it to the job scheduler for execution. The remainder of this section will describe the anatomy of a job script and how to submit and monitor jobs.

Please note script options are being provided using the long options and not the short options for readability and consistency e.g. *--nodes* versus *-N*.

Per Cluster restrictions
Warning per cluster restrictions may require you to customize the following generic instructions. e.g. NOTS has a maximum of 1 node per job, but DAVinCI does not that restriction. Please refer to the introduction of each cluster for requirements.

SLURM Batch Script Options

All jobs must be submitted via a SLURM batch script or invoking *sbatch* at the command line . See the table below for SLURM submission options.

Option	Description
#SBATCH --account=AccountName **#SBATCH --partition=PartitionName**	Required: You need to specify both the name of the account and partition to schedule jobs especially to use a condo on the cluster. Use the command ***sacctmgr show assoc user=netID*** to show which accounts and partitions with which you have access.
#SBATCH --job-name=YourJobName	Required: Assigns a job name. The default is the name of SLURM job script.
#SBATCH --ntasks=2	Recommended: The number of tasks per job. Usually used for MPI jobs. You can get further explanation here .
#SBATCH --nodes=2	Recommended: The number of nodes requested. You can get further explanation here .
#SBATCH --ntasks-per-node=2	Recommended: The number of tasks per node. Usually used in combination with --nodes for MPI jobs. You can get further explanation here .
#SBATCH --cpus-per-task=4	Recommended: The number processes per task. Usually used for OpenMP or multi-threaded jobs.
#SBATCH --partition=PartitionName	Recommended: Specify the name of the Partition (queue) to use. Use this to specify the default partition or a special partition i.e. non-condo partiton with which you have access.
#SBATCH --time=08:00:00	Recommended: The maximum run time needed for this job to run, in days-hh:mm:ss. If not specified, the default run time will be chosen automatically.

#SBATCH --mem-per-cpu=1024M	Optional: The maximum amount of physical memory used by any single process of the job ([M]ega|[G]iga|[T]era)Bytes. The value of "mem-per-cpu" multiplied by "tasks" (mem X tasks) should not exceed the amount of memory on a node. See our FAQ for more details.
#SBATCH --export=ALL	Required: Exports all environment variables to the job. See our FAQ for details.
#SBATCH --mail-user=YourEmailAddress	Recommended: Email address for job status messages.
#SBATCH --mail-type=ALL	Recommended: SLURM will notify the user via email when the job reaches the following states BEGIN, END, FAIL or REQUEUE.
#SBATCH --nodes=1 --exclusive	Optional: Using **both** of these options will give your job exclusive access to a node such that no other jobs can share the node. This combination of arguments will assign eight tasks to your job and will give it exclusive access to all of the resources (i.e. memory) of the entire node without interference from other jobs. Please see our FAQ for more details on exclusive access.
#SBATCH --output=mypath	Optional: The full path for the standard output (stdout) and standard error (stderr) "slurm-%j.out" file, where the "%j" is replaced by the job ID. Current working directory is the default.
#SBATCH --error=mypath	Optional: The full path for the standard error (stderr) "slurm-%j.out" files. Use this only when you want to separate (stderr) from (stdout). Current working directory is the default.

Serial Job Script

A job script may consist of SLURM directives, comments and executable statements. A SLURM directive provides a way of specifying job attributes in addition to the command line options. For example, we could create a *myjob.slurm* script this way:

myjob.slurm

```
#!/bin/bash
#SBATCH --job-name=YourJobNameHere
#SBATCH --account=commons
#SBATCH --partition=commons
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=1000m
#SBATCH --time=00:30:00
#SBATCH --mail-user=YourEmailAddressHere
#SBATCH --mail-type=ALL

echo "My job ran on:"
echo $SLURM_NODELIST
if [[ -d $SHARED_SCRATCH/$USER && -w $SHARED_SCRATCH/$USER ]]
then
 cd $SHARED_SCRATCH/$USER
 srun /path/to/myprogram
fi
```

This example script will submit a job to the default partition using 1 processor and 1GB of memory per processor, with a maximum run time of 30

minutes.

Definition of --ntasks-per-node
For the clusters the **--ntasks-per-node** option means *tasks per node.*

Accurate run time value is strongly recommended
It is important to specify an accurate run time for your job in your SLURM submission script. Selecting eight hours for jobs that are known to run for much less time may result in the job being delayed by the scheduler due to an overestimation of the time the job needs to run.

How to specify mem
The *--mem* value represents memory per processor core. If your *--mem* value multiplied by the number of tasks (*--ntasks-per-node*) exceeds the amount of memory per node, your job will not run. If your job is going to use the entire node, then you should use the *--exclusive* option instead of the *--mem* or *--ntasks-per-node* options (See Here). It is good practice to specify the *--mem* option if you are going to be using less than an entire node and thus sharing the node with other jobs.

If you need to debug your program and want to run in interactive mode, the same request above could be constructed like this (via the *srun* command):

```
srun --pty --partition=interactive --ntasks=1 --mem=1G --time=00:30:00 $SHELL
```

For more details on interactive jobs, please see our FAQ on this topic.

SLURM Environment Variables in Job Scripts

When you submit a job, it will inherit several environment variables that are automatically set by SLURM. These environment variables can be useful in your job submission scripts as seen in the examples above. A summary of the most important variables are presented in the table below.

Variable Name	Description
$SHARED_SCRATCH	Location of shared scratch space. See our FAQ for more details.
$LOCAL_SCRATCH	Location of local scratch space on each node.
$SLURM_JOB_NODELIST	Environment variable containing a list of all nodes assigned to the job.
$SLURM_SUBMIT_DIR	Path from where the job was submitted.

Job Launcher (srun)

For jobs that need two or more processors and **are compiled with MPI libraries**, you must use *srun* to launch your job. The job launcher's purpose is to spawn copies of your executable across the resources allocated to your job. We currently support *srun* for this task and do not support the *mpirun* or *mpiexec* launchers. By default *srun* only needs your executable, the rest of the information will be extracted from SLURM.

The following is an example of how to use *srun* inside your SLURM batch script. This example will run *myMPIprogram* as a parallel MPI code on all of the processors allocated to your job by SLURM:

myMPIjob.slurm

```
#!/bin/bash
#SBATCH --job-name=YourJobNameHere
#SBATCH --account=commons
#SBATCH --partition=commons
#SBATCH --ntasks=24
#SBATCH --mem-per-cpu=1G
#SBATCH --time=00:30:00
#SBATCH --mail-user=YourEmailAddressHere
#SBATCH --mail-type=ALL

echo "My job ran on:"
cat $SLURM_NODELIST
if [[ -d $SHARED_SCRATCH/$USER && -w $SHARED_SCRATCH/$USER ]]
then
 cd $SHARED_SCRATCH/$USER
 srun /path/to/myMPIprogram
fi
```

This example script will submit a job to the default partition using 24 processor cores and 1GB of memory per processor core, with a maximum run time of 30 minutes.

Your Program must use MPI
The above example assumes that *myMPIprogram* is a program designed to be parallel (using MPI). If your program has not been parallelized then running on more than one processor will not improve performance and will result in wasted processor time and could result in multiple copies of your program being executed.

The following example will run *myMPIprogram* on only four processors even if your batch script requested more than four.

```
srun -n 4 /path/to/myMPIprogram
```

To ensure that your job will be able to access an mpi runtime, you must load an mpi module before submitting your job as follows:

```
module load GCC OpenMPI
```

Submitting and Monitoring Jobs

Once your job script is ready, use *sbatch* to submit it as follows:

```
sbatch /path/to/myjob.slurm
```

This will return a jobID number while the output and error stream of the job will be saved to one file inside the directory where the job was submitted, unless you specified otherwise.

The status of the job can be obtained using SLURM commands. See the table below for a list of commands:

Command	Description

squeue	Show a detailed list of all submitted jobs.
squeue *-j jobID*	Show a detailed description of the job given by *jobID*.
squeue *-- start -j jobID*	Gives an estimate of the expected start time of the job given by *jobID*.

There are variations to these commands that can also be useful. They are described below:

Command	Description
squeue -l	Show a list of all running jobs.
squeue *-u username*	Show a list of all jobs in queue owned by the user specified by *username*.
scontrol *show job jobID*	To get a verbose description of the job given by *jobID*. The output can be used as a template when you are attempting to modify a job.

There are many different states that a job can be after submission: *BOOT_FAIL (BF), CANCELLED (CA), COMPLETED (CD), CONFIGURING (CF), COMPLETING (CG), FAILED (F), NODE_FAIL (NF), PENDING (PD), PREEMPTED (PR), RUNNING (R), SUSPENDED (S), TIMEOUT (TO),* or *SPECIAL_EXIT (SE)*. The *squeue* command with no arguments will list all jobs in their current state. The most common states are described below.

Running (R): These are jobs that are running.

Pending (PD): These jobs are eligible to run but there is simply not enough resources to allocate to them at this time.

Deleting Jobs

A job can be deleted by using the *scancel* command as follows:

```
scancel jobID
```