On the Hellgate Cluster, we utilize SCHEDMD’s Slurm, a work load manager. Slurm is a job scheduling system for Linux clusters, which allocates resources to users, manages job execution and handles job queues.

More information can be found at [Slurm’s Documentation](https://slurm.schedmd.com/documentation.html).

## SLURM Commands

### Sbatch

Used to submit a job script for later execution. The script will typically contain one or more commands.

We recommend using Sbatch, it allows users to more easily specify resource allocation, schedule job times, create error logs and keep the Slurm job running regardless of the user closing the terminal.

Here we created an Sbatch script named [cpu.sh](http://cpu.sh):

#!/bin/bash

#SBATCH --job-name=cpu\_job

##SBATCH --mail-type=END,FAIL # Mail events (NONE, BEGIN, END, FAIL, ALL)

##SBATCH --mail-user= # Email to notify

#SBATCH --partition='cpu(all)'

#SBATCH --nodes=1 # Number of nodes

#SBATCH --ntasks=1 # Number of simultaneous commands

#SBATCH --cpus-per-task=1 # Number of cores per command

#SBATCH --mem=8G # Memory (RAM)

#SBATCH --time=0-8 # Days-Hours

#SBATCH --output=out.txt

#SBATCH --error=err.txt

## Command(s) to run:

echo "This is a CPU job"

sleep 30

To run the script, we would reference the file after calling the sbatch command.

sbatch cpu.sh

If we would like to use a GPU resource instead, we change the partition to ask for gpu(all) to search for possible GPU nodes and add #SBATCH --gres=gpu:# to request a specific number of GPUs.

Here we create an Sbatch script named [gpu.sh](http://gpu.sh):

#!/bin/bash

#SBATCH --job-name=gpu\_job

##SBATCH --mail-type=END,FAIL # Mail events (NONE, BEGIN, END, FAIL, ALL)

##SBATCH --mail-user= # Where to send mail

#SBATCH --partition='gpu(all)'

#SBATCH --gres=gpu:1 # General Resource = type:# of resouces

#SBATCH --nodes=1 #Number of nodes

#SBATCH --ntasks=1 # Number of simultaneous commands

#SBATCH --cpus-per-task=1 # Number of simultaneous processes

#SBATCH --mem=8G # Memory (RAM)

#SBATCH --time=0-8

#SBATCH --output=out.txt

#SBATCH --error=err.txt

## Command(s) to run:

nvidia-smi # Nvidia's GPU info command

sleep 30

### SLURM Array

Sbatch has the ability to request multiple jobs using arrays, allowing us to run one script and execute multiple jobs. NOTE: Resource specifications will be for each job.

Here is an example script named [array.sh](http://array.sh) which requests 8 jobs, each with 1 CPU and 8GB of Memory:

#!/bin/bash

#SBATCH --job-name=array\_job\_%a # %A is job id, %a is array id

##SBATCH --mail-type=END,FAIL # Mail events (NONE, BEGIN, END, FAIL, ALL)

##SBATCH --mail-user= # Email to notify

#SBATCH --partition='cpu(all)'

#SBATCH --nodes=1 # Number of nodes

#SBATCH --array=1-8 # Number of jobs to run, %2 limits 2 simultaneous jobs

#SBATCH --ntasks=1 # Number of simultaneous commands

#SBATCH --cpus-per-task=1 # Number of cores per command

#SBATCH --mem=8G # Memory (RAM) per job

#SBATCH --time=0-8 #Days-Hours

#SBATCH --output=./out\_%A/out\_job%a.txt

#SBATCH --error=./err\_%A/err\_job%a.txt

### COMMAND TO RUN ###

echo "This is array job $SLURM\_ARRAY\_TASK\_ID"

sleep 30

### Srun

Used to submit a job for execution or initiate job steps in real time, srun has a wide variety of options to specify resource requirements.

Here is a command to create a shell on a node with the specified resources, allowing you to manually run commands on the node. Note the job will terminate if we leave the shell.

srun --job-name=srun\_job --nodes=1 --ntasks=1 --cpus-per-task=1 --mem=8G --time=0-8 --pty /bin/bash

Creating a job that terminates when we leave may seem pointless, but it is helpful for specifying resources for commands within a Sbatch script due to its ability to overrule the Sbatch specifications and create job steps in parallel.

In this example script, we will run two tasks at the same time and give these tasks different CPUs amounts using srun:

#!/bin/bash

#SBATCH --job-name=srun\_job

##SBATCH --mail-type=END,FAIL # Mail events (NONE, BEGIN, END, FAIL, ALL)

##SBATCH --mail-user= # Email to notify

#SBATCH --partition='cpu(all)'

#SBATCH --nodes=1 # Number of nodes

#SBATCH --ntasks=4 # Number of simultaneous commands

#SBATCH --cpus-per-task=4 # Number of CPUs

#SBATCH --mem-per-cpu=1G # Memory (RAM)

#SBATCH --time=0-8 # Days-Hours

#SBATCH --output=out.txt

#SBATCH --error=err.txt

## Command(s) to run:

# 2 CPU job

srun --ntasks=1 --cpus-per-task=2 bash -c 'echo "2 CPUs was run at $(date +\\"%H:%M:%S.%N\\") on CPU affinity: $(taskset -pc $$)"' >> out1.txt &

# 4 CPU job

srun --ntasks=1 --cpus-per-task=4 bash -c 'echo "4 CPUs was run at $(date +\\"%H:%M:%S.%N\\") on CPU affinity: $(taskset -pc $$)"' >> out1.txt &

# 8 CPU job

srun --ntasks=1 --cpus-per-task=8 bash -c 'echo "8 CPUs was run at $(date +\\"%H:%M:%S.%N\\") on CPU affinity: $(taskset -pc $$)"' >> out1.txt &

# 12 CPU job

srun --ntasks=1 --cpus-per-task=12 bash -c 'echo "12 CPUs was run at $(date +\\"%H:%M:%S.%N\\") on CPU affinity: $(taskset -pc $$)"' >> out1.txt &

wait

### Squeue

Reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options.

squeue

Using watch will keep a persistent window which updates every 2s

watch squeue

Squeue can be filtered to only your job using:

squeue --me

### Scancel

Used to cancel a pending or running job or job step.

scancel <JOB\_ID>

### Sacct

Used to report job or job step accounting information about active or completed jobs.

sacct --jobs=<JOB\_ID> --format=JobID,JobName,Elapsed,systemcpu,totalcpu,avecpu,cputime,maxvmsize,maxrss

### SSHing to Nodes

It is possible to SSH into a node that is currently running one of our jobs. This can provide us with useful information on what is going on inside the node.

In this example, we have a job running on hgcpu1-1 and would like to see what processes are running. SLURM allows us to easily SSH with:

ssh hgcpu1-1

We will then use ‘top’ to provide us with resource usage on the node:

top