Submitting a Job Through Slurm

In this documentation, we will cover the basics of submitting a job to the cluster using Slurm as our job scheduler. For more in-depth information, please refer to the extensive documentation available online through the official Slurm website or contact us at Systems for any specific questions.

Step 1: Creating a Job Script

First, we must write a job script that Slurm can use to submit the job to the cluster. If you've made a PBS script before, you will find that job scripts for Slurm are very similar. Below is an example of a working script:

```
#!/bin/bash

# set the number of nodes
#SBATCH --nodes=1

# set max wall-clock time (D-HH:MM:SS)
#SBATCH --time=0-12:00:00

# set partition/queue to use
#SBATCH --partition=day-long-cpu

# set name of job
#SBATCH --job-name=test_job

# set name of output file
#SBATCH --output=test.out

# mail alert at start, end and abortion of execution
#SBATCH --mail-type=ALL

# send mail to this address
#SBATCH --mail-user=clusterdemo@clarku.edu

echo "Hello World"
```

This is a sample script will run the user's code/application as "test_job" using one node with a max wall-clock time of 12 hours, and will send email alerts to the address "clusterdemo@clarku.edu". **NOTE: If your time limit exceeds the partition's time limit, your job might be stuck in a PENDING state indefinitely. Please choose your max wall-clock time and partition carefully to avoid this issue. To see a list of available partitions, use the command "sinfo" in the Terminal.**

If your code does not have support for multiple nodes, adding any additional nodes will not make your job faster, and in fact, might hinder its performance. However, if your code does support multiple nodes, by using MPI for example, you can make a few additions to take full advantage of this.

```
#!/bin/bash

# set the number of nodes and processes per node
#SBATCH --nodes=2

# set the number of tasks (processes) per node.
#SBATCH --ntasks-per-node=16

# run code/application

mpirun hello.mpi
```

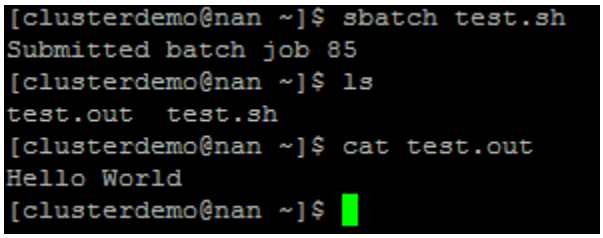
By setting nodes=2 and ntasks-per-node=16, we are setting the environment for an MPI parallel job that can make use of the multiple nodes.

The new cluster also has support for GPU jobs. Please contact us at Systems for more information.

Once we have written a proper script, we can now submit the job to the cluster.

Step 2: Submitting the Job

To submit the job, run "sbatch name_of_job" along with any parameters you might need.



Once you submit the job, you will receive a confirmation that your job was successfully submitted, as well as its job ID. Once the job finishes, you should be able to see the correct output file (STDOUT) or an error file (STDERR) if the code did not run successfully.

Step 3: Keeping Track of Your Job

There are many commands that you can use to keep an eye on how your job is doing. Here are a few commands that can provide some information on the status of your job.

sinfo	Lists the partitions that are available. Flags: -N (nodes) –l (more info)
squeue	Shows list of jobs that are currently running (R) or waiting for resources (PD).
scancel [jobid]	Cancels the job specified by the job ID.
sprio	Shows priority of pending jobs.
sstat [jobid]	Shows detailed information about the listed job.

There are more commands available which include many flags that can provide you with detailed information on the status of your job. Reading over the man pages of Slurm commands and knowing the full extent of the capabilities that Slurm offers will definitely make your life easier on the long run.

Final Words: Etiquette and Shared Resources

Our high performance cluster is a powerful tool used by many faculty members and students. Please remember that this is a shared service with limited resources. On the old cluster, we only had one queue which would sometimes cause short and simple jobs to be listed behind long jobs, causing unnecessarily long queue times. The multiple partitions/queues in the new cluster are a huge improvement in this aspect, if used properly. When submitting a job, please try to use a good guesstimate of max wall-clock time, and submit it to the correct partition depending on the needed wall-clock time. This will help maximize the performance and health of our cluster and its users.