

GW COLONIAL ONE



HPC Workshop 2

For new Colonial One users!!!

Login to Colonial One using your GW NetID/email address & password

```
$ ssh <netid>@login.colonialone.gwu.edu
```

```
cd ~
```

```
cd skel
```

```
cp .bash* ../
```

```
cd ~
```



HPC Workshop 2

What we're covering:

- Working with SLURM
- Modules (review)
- Simple job submission script
- How to submit job script
- sinfo, salloc, squeue, scancel, sbatch, srun
- Tips for scripting submit files
- Reporting Job Errors



Working with SLURM

Overview: Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters.

- Slurm requires no kernel modifications for its operation and is relatively self-contained. As a cluster workload manager, Slurm has three key functions:
 - First, it allocates exclusive access to resources (compute nodes) for users for some duration of time so they can perform work.
 - Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes.
 - Finally, it arbitrates contention for resources by managing a queue of pending work.



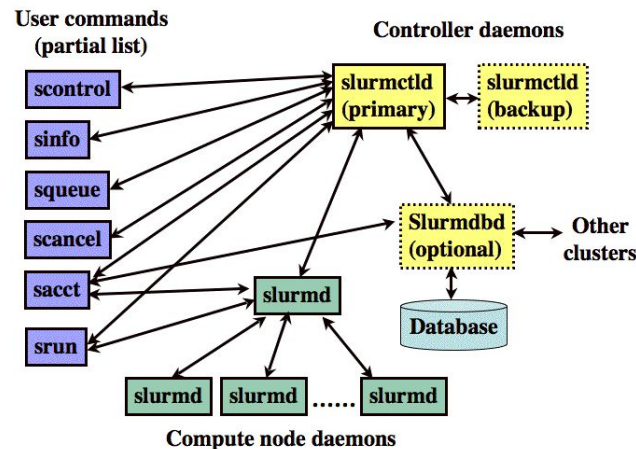
Working with SLURM

Why use SLURM?

- SLURM allows jobs to be scheduled so a user does not have to wait until a node is free to begin work. SLURM does the waiting for you!
- SLURM allows resources to be prioritized for groups that purchase shares on Colonial One.
- SLURM enables efficient use of the cluster since it constantly monitors resources in use and schedules jobs on unallocated resources as they free up

Architecture I

As depicted in Figure 1, Slurm consists of a **slurmd** daemon running on each compute node and a central **slurmctld** daemon running on a management node (with optional fail-over twin). The **slurmd** daemons provide fault-tolerant hierarchical communications. The user commands include: **sacct**, **salloc**, **sattach**, **sbatch**, **sbcast**, **scancel**, **scontrol**, **sinfo**, **smap**, **squeue**, and **srun**. All of the commands can run anywhere in the cluster.



Architecture II

The entities managed by these Slurm daemons, shown in Figure 2, include **nodes**, the compute resource in Slurm, **partitions**, which group nodes into logical (possibly overlapping) sets, **jobs**, or allocations of resources assigned to a user for a specified amount of time, and **job steps**, which are sets of (possibly parallel) tasks within a job. The partitions can be considered job queues, each of which has an assortment of constraints such as job size limit, job time limit, users permitted to use it, etc. Priority-ordered jobs are allocated nodes within a partition until the resources (nodes, processors, memory, etc.) within that partition are exhausted. Once a job is assigned a set of nodes, the user is able to initiate parallel work in the form of job steps in any configuration within the allocation. For instance, a single job step may be started that utilizes all nodes allocated to the job, or several job steps may independently use a portion of the allocation.

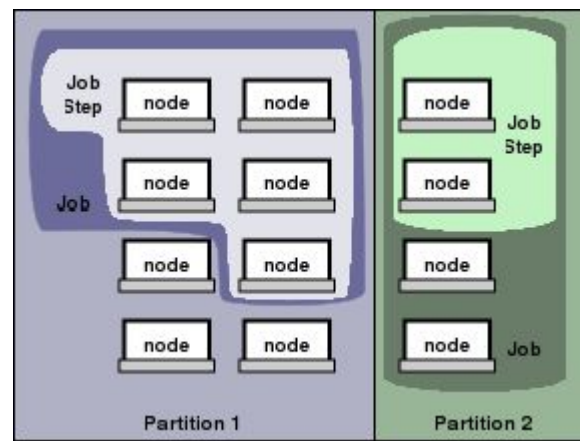


Figure 2. Slurm entities



Software, Modules & Login

- Common software is available to all users via the “module” system
- Can I install my own software?
 - Yes, provided it runs from your home/group directory
 - Users cannot use a package manager (yum) to install software
 - Colonial One users do not have root on Colonial One

Login to Colonial One using your GW NetID/email address & password

```
$ ssh <netid>@login.colonialone.gwu.edu
```




Modules

List and load modules

- `module list`
- `module avail`
- `module load <module name>`
- `module unload <module name>`
- `module spider <search term>`

SLURM Commands

sinfo

- lists node and partition information for the cluster
- useful for finding unallocated nodes

Example: type "sinfo" at the prompt

```
PARTITION    AVAIL  TIMELIMIT  NODES  STATE NODELIST
defq*        up 14-00:00:0  128  alloc node[033-160]
short        up 2-00:00:00   95  alloc node[097-191]
128gb        up 14-00:00:0   24  alloc node[041-064]
256gb        up 14-00:00:0    8  alloc node[033-040]
2tb          up 14-00:00:0    1  alloc node901
gpu          up 7-00:00:00   22  alloc node[003-020,029-032]
gpu          up 7-00:00:00   10  idle node[001-002,021-028] ← Free nodes!
gpu-noecc    up 7-00:00:00   22  alloc node[003-020,029-032]
gpu-noecc    up 7-00:00:00   10  idle node[001-002,021-028] ← Free nodes!
ivygpu       up 7-00:00:00   21  idle node[333-353] ← Free nodes!
ivygpu-noecc up 7-00:00:00   21  idle node[333-353] ← Free nodes!
allgpu-noecc up 7-00:00:00   22  alloc node[003-020,029-032]
allgpu-noecc up 7-00:00:00   31  idle node[001-002,021-028,333-353] ← Free
nodes!
debug        up    4:00:00    2  alloc node[991-992]
debug-cpu    up    4:00:00    1  alloc node992
debug-gpu    up    4:00:00    1  alloc node991
```



SLURM Commands

salloc - Obtain a Slurm job allocation (a set of nodes), execute a command, and then release the allocation when the command is finished. You can use `salloc` to run interactive jobs:

```
salloc -n 16 -p ivygpu -t 5
```

srun - Run a parallel job on cluster managed by Slurm. Use `srun` to identify your allocated nodes after running `salloc`:

```
srun hostname  
ssh nodename
```



SLURM Commands

squeue - view information about jobs located in the Slurm scheduling queue.

List jobs for your user account:

```
squeue -u <username>
```

Estimate when a job will start:

```
squeue -u username --start
```

List by job status:

```
squeue -u username -t RUNNING or PENDING
```



SLURM job status

Check Status of Job by User

```
[user@login4 ~]$ squeue -u <username>
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
2325403	defq	job_OpenMP.sh	user	PD	0:00	1	(Resources)

Jobs typically pass through several states in the course of their execution. The typical states are PENDING, RUNNING, SUSPENDED, COMPLETING, and COMPLETED. An explanation of each state follows.

CA = CANCELLED
CD = COMPLETED
CG = COMPLETING
F = FAILED
NF = NODE_FAIL
PD = PENDING
R = RUNNING
S = SUSPENDED
TO = TIMEOUT



SLURM Commands

sbatch - Submit a batch script to Slurm

```
sbatch <submit_script.sh>
```

scancel - used to signal or cancel jobs, job arrays or job steps.

```
scancel <jobid>
```

sprio - view the factors that comprise a job's scheduling priority

```
sprio -u <username>
```



SLURM COMMANDS

- `sinfo` - view information about Slurm nodes and partitions.
- `salloc` - Obtain a Slurm job allocation (a set of nodes), execute a command, and then release the allocation when the command is finished.
- `squeue` - view information about jobs located in the Slurm scheduling queue.
- `scancel` - Used to signal jobs or job steps that are under the control of Slurm.
- `sbatch` - Submit a batch script to Slurm.
- `sprio` - view the factors that comprise a job's scheduling priority
- `srun` - Run parallel jobs



hello-mpi.py

This small Python program will list an mpi process number and the node it is running on. Copy the text into a document called "hello.py" and save it.

```
#hello.py

from mpi4py import MPI

comm = MPI.COMM_WORLD

rank = comm.Get_rank()

size = MPI.COMM_WORLD.Get_size()

print "hello world from process ", rank, " of ",size
```



How to submit job script I

Load the current MPI module so we can run our program correctly:

```
module load openmpi/1.8/gcc/4.9.2/cpu  
module load python/2.7.6
```

Check your home directory to make sure your compiled program is there:

```
ls -la hello*
```



How to submit job script I

Create a SLURM script using an editor such as vi or emacs using steps 1 through 3. The script (or file) can be called anything you want but should end in .sh (i.e. submit.sh).

Step 1: Resource Specification

```
[user@login4 ~]$ nano submit.sh

#!/bin/sh
#SBATCH --time 5:00
#SBATCH -o testing%j.out
#SBATCH -e testing%j.err
#SBATCH -p defq -n 8
#SBATCH --mail-user=<username>@gwu.edu
#SBATCH --mail-type=ALL
module load openmpi/1.8/gcc/4.9.2/cpu
module load python/2.7.6
mpirun python /home/<username>/hello.py
```

Step 2: Submit job

```
module load slurm
[user@login4 ~]$ sbatch submit.sh
Submitted batch job 2325403
```



How to submit job script I

SLURM will email you when your job has finished. Once the job is finished, SLURM will place a .out file and a .err file (if there are errors) in the directory with your submit script.

```
$> ls slurm*
slurm-2333627.out  slurm-2333631.out  slurm-2333633.out
slurm-2333629.out  slurm-2333632.out  slurm-2333636.out
```

The .out file is the result of your job. Cat or less the file to show you the results.

```
$> cat slurm-2333636.out
Process 1 on node991.cm.cluster out of 8
Process 3 on node991.cm.cluster out of 8
Process 7 on node991.cm.cluster out of 8
Process 5 on node991.cm.cluster out of 8
Process 4 on node991.cm.cluster out of 8
Process 6 on node991.cm.cluster out of 8
Process 0 on node991.cm.cluster out of 8
Process 2 on node991.cm.cluster out of 8
```



Reporting Job Errors

- Email to hpchelp@gwu.edu
- Include .err file
- Include submit script
- Include what modules you loaded