Outline of talk

1. Live demo
2. Duke Compute Cluster overview
3. Introduction to SLURM basics
4. Running multi-core and parallel jobs
5. Running jobs with dependencies
6. More live demo
Live Demo notes

- `df -h`
- `srun --pty bash -i`
- `squeue | more`
- `squeue -S S`
- `squeue -S S | grep -v PD`
- `squeue -u (NetID)`
- `sbatch (job script)`
- `scancel (job id)`
- `uname -n, sleep, top`
- `sinfo | grep -v common`
- `scontrol show node (node name)`
- `scontrol show job (job id)`
Accessing the Duke Compute Cluster

• From on-campus (University or DUHS networks):
  `ssh dscr-slogin-01.oit.duke.edu`
  (or `dscr-slogin-02`)

• From off campus, first
  `ssh login.oit.duke.edu`
  and then `ssh` to an `slogin` node

• Alternatively, use the Duke VPN:
  https://oit.duke.edu/net-security/network/remote/vpn/
SLURM commands

sbatch
• Submit a batch job (like “qsub”)

#SBATCH
• Specify job parameters (like “#$”)

squeue (like “qstat”)
• Show lists of jobs

scancel (like “qdel”)
• Delete one or more batch jobs

sinfo (like “qhost”)
• Show info about machines

scontrol
• Show cluster configuration information
SLURM resources

• The DSCR wiki: “SLURM Queueing System”
  https://wiki.duke.edu/display/SCSC
• https://wiki.duke.edu/display/SCSC/SLURM+Queueing+System

“Official” SLURM docs
• http://schedmd.com/slurmdocs/

Older SLURM documentation
• https://computing.llnl.gov/linux/slurm/slurm.html
• Comes up a lot in Google searches
• outdated – use schedmd.com instead
Running an interactive job

No built-in SLURM equivalent to “qrsh“

• Reserve a compute node by typing
  `srun --pty bash -i`

```
tm103@dscr-slogin-02  ~ $ srun --pty bash -i
srun: job 186535 queued and waiting for resources
srun: job 186535 has been allocated resources
tm103@dscr-encode-11  ~ $
```

```
tm103@dscr-encode-11  ~ $ squeue -u tm103
  JOBID  PARTITION   NAME   USER ST TIME  NODES NODELIST(REASON)
  186535  common    bash  tm103 R  0:14   1 dscr-encode-11
```
Job are within “partitions”

- Most partitions are department-owned machines
- Submit to partitions with “--partition=” or “-p”
  E.g. interactively: `srun -p (partition name) --pty bash -i`
  `srun -p econ --pty bash -i` etc.
Duke Compute Cluster partitions

- “Partitions” are comparable to old DSCR “groups”
- Specified with “--partition=...“ or “-p=...”
  - e.g. `#SBATCH --partition=dbchem`
- Partitions can overlap
- The default partition is called “common”
- The common partition includes all nodes
- Submitting to a group partition gives “high-priority”
- A high-priority job can preempt common partition jobs
sbatch

• Use “sbatch” (all lower case) to submit
  `sbatch test.q`
• Use “#SBATCH” (upper case) in your scripts for scheduler directives, e.g.
  
  ```
  #SBATCH --mem=1000
  #SBATCH --output=matlab.out
  ```

• All SLURM directives can be given on the command line instead of the script.

• [http://slurm.schedmd.com/sbatch.html](http://slurm.schedmd.com/sbatch.html)
#!/bin/bash
#
#SBATCH --output=test.out
uname -n  # print hostname

This prints the name of the compute node in the file "test.out".

tm103@dscr-slogin-02  ~/slurm $ sbatch simple.q
Submitted batch job 186554
tm103@dscr-slogin-02  ~/slurm $ cat test.out
dscr-compeb-14
Comparing SLURM to SGE example

• SGE script

```bash
#!/bin/bash
#$ -S /bin/bash -cwd
#$ -n test
#$ -o test.out -j y
#$ -l mem_free=1.0G
#$ -l highprio
uname -n
```
SLURM script (long command names)

!/bin/bash
#SBATCH --job-name=test
#SBATCH --output=test.out
#SBATCH --mem=100  # 100 MB RAM
#SBATCH --partition=dbchem  # e.g. uname -n  # print hostname

• For a user in the “dbchem” group, this job will run in high priority on an “dbchem” node.
SLURM script (short command names)

- SLURM equivalent (short command names)

```
!/bin/bash
#SBATCH -J=test
#SBATCH -o=test.out
#SBATCH --mem=1G  # 1GB RAM
#SBATCH -p dbchem  # only works for owners
uname -n
```
SLURM memory directives

--mem=<MB>
• The amount of memory required per node

--mem-per-cpu=<MB>
• The amount of memory per CPU core
• For multi-threaded jobs
• **This is a hard limit** – always request a little more

Note: --mem and --mem-per-cpu are mutually exclusive
SLURM parallel directives

• All parallel directives have defaults of 1

-N <number>  How many nodes (machines)

-n <number> or --ntasks=<number> How many parallel jobs ("tasks")

-c, --cpus-per-task=<ncpus>

• Use for multi-threaded jobs

• The --ntasks default is one CPU core per task, but the --cpus-per-task option will change this default.
Multi-threaded (multi-core) example

!/bin/bash
#SBATCH -J test
#SBATCH -o test.out
#SBATCH --c 4
#SBATCH --mem-per-cpu=500 (500 MB)
myApplication ...

- This launches a single, multi-threaded job that uses 4 CPU cores and 2 GB (4x500MB) of RAM
Job Arrays

• a mechanism for submitting and managing collections of similar jobs

• Job arrays are only supported for batch jobs and the array index values are specified using the --array or -a option

• http://slurm.schedmd.com/job_array.html
SGE Array job example

#!/bin/tcsh
#$ -S /bin/tcsh -cwd
#$ -o matlab.out -j y
#$ -l mem_free=1.0G
#$ -t 1-100

/opt/apps/MATLAB/R2012b/bin/matlab nojvm -nodisplay -singleCompThread -r
"rank=$SGE_TASK_ID;my_program;quit"
SLURM Job Array equivalent

#!/bin/bash
#SBATCH --output=matlab.out
#SBATCH --array=1-100
#SBATCH --mem=4G

/opt/apps/MATLAB/R2012b/bin/matlab
-nojvm -nodisplay -singleCompThread -r "rank=${SLURM_ARRAY_TASK_ID};my_prog;quit"
Running MPI jobs

• Supported MPI versions
  - Intel MPI
  - OpenMPI
  - LAM MPI

• SLURM MPI jobs - use “--ntasks=“
• https://wiki.duke.edu/display/SCSC/Running+MPI+Jobs
Compiling with OpenMPI

tm103@dscr-slogin-02 ~$ /misc/slurm/openmpi $
export PATH=/opt/apps/slurm/openmpi/bin:$PATH

tm103@dscr-slogin-02 ~$ which mpicc
/opt/apps/slurm/openmpi/bin/mpicc

tm103@dscr-slogin-02 ~$ mpicc -o openhello hello.c
tm103@dscr-slogin-02 ~$ ls -l openhello
-rwxr-xr-x 1 tm103 scsc 9184 Sep 2 12:26 openhello
#!/bin/bash

#SBATCH --ntasks=20
#SBATCH --output=openhello.out

export PATH=/opt/apps/slurm/openmpi/bin:$PATH

mpirun -n $SLURM_NTASKS openhello
OpenMPI example output

tm103@dscr-slogin-02 ~/misc/slurm/openmpi $ cat openhello.out
dscr-core-01, rank 0 out of 20 processors
dscr-core-01, rank 1 out of 20 processors
dscr-core-01, rank 2 out of 20 processors
dscr-core-01, rank 3 out of 20 processors
dscr-core-01, rank 4 out of 20 processors
dscr-core-03, rank 13 out of 20 processors
dscr-core-03, rank 14 out of 20 processors
dscr-core-03, rank 10 out of 20 processors
dscr-core-03, rank 11 out of 20 processors
dscr-core-03, rank 12 out of 20 processors
dscr-core-03, rank 14 out of 20 processors
dscr-core-02, rank 8 out of 20 processors
dscr-core-02, rank 9 out of 20 processors
dscr-core-02, rank 5 out of 20 processors
...
#!/bin/bash
#SBATCH --ntasks=20
#SBATCH --output=intelhello.out

export I_MPI_PMI_LIBRARY=/opt/slurm/lib64/libpmi.so
source /opt/apps/intel/intel/intelvars.sh
srun -n $SLURM_NTASKS intelhello
Job dependencies

Start job “dep2” after job “dep1”

$ sbatch dep1.q

• Make a note of the assigned job ID of dep1

$ squeue -u tm103

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST (REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>230398</td>
<td>common test</td>
<td>tm103</td>
<td>R</td>
<td>0:04</td>
<td>1</td>
<td>dscrtstgrp-07</td>
<td></td>
</tr>
</tbody>
</table>

$ sbatch --dependency=afterok: 230398 dep2.q
# Wait for specific job array elements
sbatch --depend=after:123_4 my.job
sbatch --depend=afterok:123_4:123_8 my.job2

# Wait for entire job array to complete
sbatch --depend=afterany:123 my.job

# Wait for entire job array to complete successfully
sbatch --depend=afterok:123 my.job

# Wait for entire job array to complete and at least one task fails
sbatch --depend=afternotok:123 my.job
Questions and live demo

http://wiki.duke.edu/display/SCSC
http://sites.duke.edu/scsc
rescomputing@duke.edu
tm103@duke.edu