Advanced Computing Center for Research and Education

SLURM
Resource Management and Job Scheduling

Visit us here: www.accre.vanderbilt.edu
Follow us on Twitter: @ACCREVandy
Visit our GitHub Repos: https://github.com/accre
What is SLURM?

Simple Linux Utility for Resource Management

- SLURM manages resources on the compute nodes.
- SLURM schedules jobs using those resources.
- SLURM is fully open source – GPL v3+.
- SLURM is maintained and supported by SchedMD.
- SLURM is used at academic research compute centers and national labs around the world.
Advanced Computing Center for Research and Education

Using SLURM

SBATCH: Submitting jobs to the cluster

Visit us here: www.accre.vanderbilt.edu
Follow us on Twitter: @ACCREVandy
Visit our GitHub Repos: https://github.com/accre
Login to the cluster:

ssh <your-vunetid>@login.accre.vanderbilt.edu

Microsoft Windows users may use PuTTY.

Create a job script:

A script is used to request resources and run programs. Use one of several editors: nano, emacs, or vi. nano is a good choice for users new to Linux. emacs and vi are challenging to learn, but in the end are satisfying to use.
Name scripts meaningfully:
   Many users use .slurm to clearly identify scripts. For example, monti_carlo_sym.slurm

Use sbatch to submit the script to SLURM:
   sbatch monti_carlo_sym.slurm
   SLURM will respond with a jobid:
      Submitted batch job 3115419
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=1G
#SBATCH --time=0-01:30:00
#SBATCH --output=my_output.txt
#SBATCH --mail user=vunetid@vanderbilt.edu,you@gmail.com
#SBATCH --mail-type=ALL
#SBATCH --job-name="just_a_test"
# The next line will be run once as the one task
/bin/hostname
About the Simple Script

Requesting a single node: \texttt{--nodes=1}
Informing SLURM there will be just one task: \texttt{--ntasks=1}
Requesting one gigabyte of memory: \texttt{--mem=1G}
Requesting one hour and thirty minutes of time: \texttt{--time=0-01:30:00}
Output not sent to a file goes here: \texttt{--output=my\_output.txt}
Send email here: \texttt{--mail-user=vunetid@vanderbilt.edu,you@gmail.com}
Email when job starts, ends, or aborts: \texttt{--mail-type=ALL}
The name of the job in the queue: \texttt{--job-name=\textquote{just\_a\_test}}
This will be run once as the single task: \texttt{/bin/hostname}
#!/bin/bash must be the first line in the script, and the “#” character must be the first character on the first line. Failing to do this can lead to problems that are very hard to diagnose.

Group all #SBATCH directives together. Insert all comments before or after the group.

The format for time is dd-hh:mm:ss and the maximum amount of time is 14 days: 14-00:00:00 and the day specifier is optional. So, one hour can be requested as 01:00:00 or 00:60:00.

Memory may be requested per node: --mem=
Memory may be requested per cpu: --mem-per-cpu=
Use --job-name to make job names meaningful to you.
About the Simple Script

Things to remember

--mail-type maybe any combination of the following:
BEGIN
END
FAIL
ALL (equivalent to BEGIN, END, FAIL)
TIME_LIMIT_90 (reached 90 percent of time limit)
TIME_LIMIT_80 (reached 80 percent of time limit)
TIME_LIMIT_50 (reached 80 percent of time limit)
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=5
#SBATCH --mem-per-cpu=2G
#SBATCH --time=2-12:00:00
#SBATCH --output=multi_threaded_output.txt
#SBATCH --mail-user=vunetid@vanderbilt.edu
#SBATCH --mail-type=ALL,TIME_LIMIT_50,TIME_LIMIT_90
#SBATCH --job-name="job_with_five_threads"

python multi_threaded.py
About the Multi-Threaded Script

Things to remember

Memory can be requested per node as well as per cpu.

Limit threads to no more than 2 per cpu. Requesting more than that will be honored, but might slow your program down.

Request multiple cpus **only** when you know your program is multi-threaded. Most programs are not multi-threaded.

This script requests email when the job begins, fails or ends; and when the job has reached 50% of requested time, and 90% of requested time.
Overview

Job arrays offer a mechanism for submitting and managing collections of similar jobs quickly and easily.

Job arrays with 10 to 20 thousand jobs can be submitted with a single script, and handled by SLURM in a matter of minutes.

All jobs must have the same initial options (e.g. size, time limit).

Users may limit how many such jobs are running simultaneously.

Job arrays are only supported for batch jobs.
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --time=02:00:00
#SBATCH --mem=5G
#SBATCH --array=0-250
#SBATCH --output=matrix_data_%A_%a.out

echo "SLURM_JOBID: " $SLURM_JOBID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID
setpkgs -a ruby2.2.0
matrix_transform matrix_data_${SLURM_ARRAY_TASK_ID}
When `sbatch` is used to submit this script, SLURM will create 251 jobs. Each job is named with its jobid and array index. Suppose the jobid SLURM gave was 7765988, then these files:

- `matrix_data_7765988_0.out`
- `matrix_data_7765988_1.out`
- ...
- `matrix_data_7765988_250.out`

Will be found in the directory where the job was submitted.
Since the script requested 251 jobs in the array the program matrix_transform will need to find the following files:
matrix_data_0
matrix_data_1
matrix_data_2

.
.
.
.matrix_data_250

in the directory where the job was submitted.
Things to remember

All the input files are named exactly the same with just an integer at the end.

The 251 jobs will all have the same resources: 1 node, 1 cpu, 5G of memory, and request 2 hours of wall time.

Every job will have the same jobid, but a different index value. The jobid will appear in the queue as jobid_index. In this case, the jobids will appear as 7765988_0, 7765988_1, 7765988_2, …, 7765988_251
#!/bin/bash

#SBATCH --mail-user=vunetid@vanderbilt.edu
#SBATCH --mail-type=ALL
#SBATCH --ntasks=1
#SBATCH --time=2:00:00
#SBATCH --mem=2G
#SBATCH --array=1-500
#SBATCH –output=python_image_job_slurm_%A_%a.out

arrayfile=`ls data/ | awk -v line=$SLURM_ARRAY_TASK_ID '{if (NR == line) print $0}'`
setpkgs -a python2.7.8
pyarray.py -i data/$arrayfile -o $arrayfile_%A_%a.jpg
Things to remember

Each array job has access to two environment variables:

- `SLURM_ARRAY_JOB_ID` will be set to the first job ID of the array.
- `SLURM_ARRAY_TASK_ID` will be set to the job array index value.

Two additional options are available for file names:

- `%A` will be replaced by the value of `SLURM_ARRAY_JOB_ID`.
- `%a` will be replaced by the value of `SLURM_ARRAY_TASK_ID`.

The command:

```
arrayfile=`ls data/ | awk -v line=$SLURM_ARRAY_TASK_ID '{if (NR == line) print $0}'`
```

picks off a file name in `data/` one file at a time and stores that name in `$arrayfile`. Which file name? Well, the `$SLURM_ARRAY_TASK_ID` one, which is stored in “line”.

All 500 jobs will have the same resources in memory, cpus and walltime.
#!/bin/bash
#SBATCH --mail-user=vunetid@vanderbilt.edu
#SBATCH --mail-type=ALL
#SBATCH --nodes=3
#SBATCH --tasks-per-node=8
#SBATCH --time=7-00:00:00
#SBATCH --mem=4G    # 4 GB RAM per node
#SBATCH --output=mpi_job_slurm.log
setpkgs -a openmpi_1.8.4
echo $SLURM_JOB_NODELIST
srun --mpi=pmi2 ./test
About the Multi-Node Script

Things to remember

--nodes=3 Request three nodes.

--tasks-per-node=8 Request 8 MPI processes per node.

--output=mpi_job_slurm.log Capture standard output in this file.

```
echo $SLURM_JOB_NODELIST
```
Place the list of nodes on which the MPI job is running into the mpi_job_slurm.log file.

```
srun --mpi=pmi2 ./test
```
Use srun to launch an MPI job rather than mpirun or mpiexec.
# SBATCH Directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>--nodes=[count]</td>
<td>Number of nodes requested.</td>
</tr>
<tr>
<td>--tasks-per-node=[count]</td>
<td>Number of processes on each node.</td>
</tr>
<tr>
<td>--ntasks=[count]</td>
<td>Total number of processes requested.</td>
</tr>
<tr>
<td>--cpus-per-task=[count]</td>
<td>Number of cores per process requested.</td>
</tr>
<tr>
<td>--nodelist=[nodes]</td>
<td>Prefer jobs to run here.</td>
</tr>
<tr>
<td>--exclude=[nodes]</td>
<td>Prefer jobs <strong>not</strong> to run here.</td>
</tr>
<tr>
<td>--time=days-hh:mm:ss</td>
<td>Wall-time limit requested.</td>
</tr>
<tr>
<td>--mem=[amount][M or G or T]</td>
<td>Memory per node (avoid --mem-per-cpu)</td>
</tr>
<tr>
<td>--mem-per-cpu=[amount][M or G or T]</td>
<td>Memory per cpu (avoid --mem)</td>
</tr>
</tbody>
</table>
### SBATCH Directives (contd.)

<table>
<thead>
<tr>
<th>Directive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>--output=[[file_name]]</td>
<td>STDOUT goes here.</td>
</tr>
<tr>
<td>--error=[[file_name]]</td>
<td>STDERR goes here.</td>
</tr>
<tr>
<td>--array=[array specification]</td>
<td>Job array index, often of the form [0-XXX]</td>
</tr>
<tr>
<td>--array=[array specification]%[count]</td>
<td>Permit only “count” jobs to run at one time.</td>
</tr>
<tr>
<td>--mail-user=[[email_address]]</td>
<td>Send e-mail notifications here.</td>
</tr>
<tr>
<td>--email-type=[[BEGIN</td>
<td>END</td>
</tr>
<tr>
<td>--account=[[group name]]</td>
<td>Run job under this group.</td>
</tr>
<tr>
<td>--job-name=[[name]]</td>
<td>Name of the job.</td>
</tr>
<tr>
<td>--contstrain=[[attribute]]</td>
<td>Request nodes of a certain type, e.g., “intel”</td>
</tr>
</tbody>
</table>
Using SLURM

SQUEUE: Locating my job
squeue – Show job information

Things to remember

squeue will show all jobs pending and running.

squeue --user your-vunetid will show all of your jobs.

squeue --account your-group will show all jobs for your group.

squeue --start jobid will sometimes show a start time.
<table>
<thead>
<tr>
<th>Reason job is pending</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssocGrpCpuLimit</td>
<td>The limit for the total number of cpus of all running jobs of the group has been reached. One or more running jobs must end before more jobs may start.</td>
</tr>
<tr>
<td>AssocGrpMemLimit</td>
<td>The limit for the total amount of memory of all running jobs of the group has been reached. One or more running jobs must end before more jobs may start.</td>
</tr>
<tr>
<td>AssocGrpTimeLimit</td>
<td>The limit for the total amount of wall time requested by all running jobs of the group has been reached. One or more running jobs must end before more jobs may start.</td>
</tr>
<tr>
<td>Priority</td>
<td>Other jobs in the queue have higher priority, and those will usually run before this one.</td>
</tr>
</tbody>
</table>
Advanced Computing Center for Research and Education

Using SLURM

RTRACEJOB: get information about my job

Visit us here: www.accre.vanderbilt.edu
Follow us on Twitter: @ACCREVandy
Visit our GitHub Repos: https://github.com/accre
rtracejob will show useful information about a job:

- Memory requested and used
- Walltime requested and used
- Exit codes
- Cpus requested and used
- Node list
- Wait time in the queue
- Submit time and date
- Start time and date
- End time and date
### rtracejob – Example 1

<table>
<thead>
<tr>
<th>User: uscms4425</th>
<th>JobID: 6338720</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>cms</td>
</tr>
<tr>
<td>Job Name</td>
<td>bl_67e5efbe2748</td>
</tr>
<tr>
<td>State</td>
<td>Completed</td>
</tr>
<tr>
<td>Exit Code</td>
<td>0:0</td>
</tr>
<tr>
<td>Wall Time</td>
<td>2-00:00:00</td>
</tr>
<tr>
<td>Requested Memory</td>
<td>4000Mn</td>
</tr>
<tr>
<td>Memory Used</td>
<td>1813176K</td>
</tr>
<tr>
<td>CPUs Requested</td>
<td>1</td>
</tr>
<tr>
<td>CPUs Used</td>
<td>1</td>
</tr>
<tr>
<td>Nodes</td>
<td>1</td>
</tr>
<tr>
<td>Node List</td>
<td>vmp617</td>
</tr>
<tr>
<td>Wait Time</td>
<td>0.4 minutes</td>
</tr>
<tr>
<td>Run Time</td>
<td>233.3</td>
</tr>
<tr>
<td>Submit Time</td>
<td>Thu Dec 10 06:59:28 2015</td>
</tr>
<tr>
<td>Start Time</td>
<td>Thu Dec 10 06:59:50 2015</td>
</tr>
<tr>
<td>End Time</td>
<td>Thu Dec 10 10:53:10 2015</td>
</tr>
<tr>
<td>Today's Date</td>
<td>Thu Dec 10 12:50:47 2015</td>
</tr>
<tr>
<td>User: leew8</td>
<td>JobID: 6205733</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Account</td>
<td>cho_lab</td>
</tr>
<tr>
<td>Job Name</td>
<td>con13_3models</td>
</tr>
<tr>
<td>State</td>
<td>Running</td>
</tr>
<tr>
<td>Exit Code</td>
<td></td>
</tr>
<tr>
<td>Wall Time</td>
<td>14-00:00:00</td>
</tr>
<tr>
<td>Requested Memory</td>
<td>8Gn</td>
</tr>
<tr>
<td>Memory Used</td>
<td>Unknown</td>
</tr>
<tr>
<td>CPUs Requested</td>
<td>2</td>
</tr>
<tr>
<td>CPUs Used</td>
<td>2</td>
</tr>
<tr>
<td>Nodes</td>
<td>1</td>
</tr>
<tr>
<td>Node List</td>
<td>vmp619</td>
</tr>
<tr>
<td>Wait Time</td>
<td>0.2 minutes</td>
</tr>
<tr>
<td>Run Time</td>
<td></td>
</tr>
<tr>
<td>Submit Time</td>
<td>Tue Dec 1 15:25:58 2015</td>
</tr>
<tr>
<td>Start Time</td>
<td>Tue Dec 1 15:26:08 2015</td>
</tr>
<tr>
<td>End Time</td>
<td>Not yet known</td>
</tr>
<tr>
<td>Today's Date</td>
<td>Thu Dec 10 10:54:33 2015</td>
</tr>
</tbody>
</table>
Using SLURM

sacct: get detailed information about my job

Visit us here: www.accre.vanderbilt.edu
Follow us on Twitter: @ACCREVandy
Visit our GitHub Repos: https://github.com/accre
Accounting information for jobs invoked with Slurm are saved to the Slurm database.

The `sacct` command displays job accounting data stored in the database in a variety of forms for your analysis. The `sacct` command displays information on jobs, job steps, status, and exitcodes by default. You can tailor the output with the use of the `--format=` option to specify the fields to be shown.

The `sacct` command limits the display of job accounting data to jobs that were launched with their own user identifier (UID) by default. Data for other users can be displayed with the `--allusers`, `--user`, or `--uid` options.
<table>
<thead>
<tr>
<th>AllocCPUS</th>
<th>AllocGRES</th>
<th>AllocNodes</th>
<th>AllocTRES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>AssocID</td>
<td>AveCPU</td>
<td>AveCPUFreq</td>
</tr>
<tr>
<td>AveDiskRead</td>
<td>AveDiskWrite</td>
<td>AvePages</td>
<td>AveRSS</td>
</tr>
<tr>
<td>AveVMSize</td>
<td>BlockID</td>
<td>Cluster</td>
<td>Comment</td>
</tr>
<tr>
<td>ConsumedEnergy</td>
<td>ConsumedEnergyRaw</td>
<td>CPUPTime</td>
<td>CPUTimeRAW</td>
</tr>
<tr>
<td>DerivedExitCode</td>
<td>Elapsed</td>
<td>Eligible</td>
<td>End</td>
</tr>
<tr>
<td>ExitCode</td>
<td>GID</td>
<td>Group</td>
<td>JobID</td>
</tr>
<tr>
<td>JobIDRaw</td>
<td>JobName</td>
<td>Layout</td>
<td>MaxDiskDiskRead</td>
</tr>
<tr>
<td>MaxDiskReadNode</td>
<td>MaxDiskReadTask</td>
<td>MaxDiskWrite</td>
<td>MaxDiskWriteNode</td>
</tr>
<tr>
<td>MaxDiskWriteTask</td>
<td>MaxPages</td>
<td>MaxPagesNode</td>
<td>MaxPagesTask</td>
</tr>
<tr>
<td>MaxRSS</td>
<td>MaxRSSNode</td>
<td>MaxRSSTask</td>
<td>MaxRSSTask</td>
</tr>
<tr>
<td>MaxVMSizeNode</td>
<td>MaxVMSizeTask</td>
<td>MinCPU</td>
<td>MinCPU</td>
</tr>
<tr>
<td>MinCPUTask</td>
<td>NCPUS</td>
<td>NNodes</td>
<td>NodeList</td>
</tr>
<tr>
<td>NTasks</td>
<td>Priority</td>
<td>Partition</td>
<td>QOS</td>
</tr>
<tr>
<td>QOSRAW</td>
<td>ReqCPUFreq</td>
<td>ReqCPUFreqMin</td>
<td>ReqCPUFreqMax</td>
</tr>
<tr>
<td>ReqCPUFreqGov</td>
<td>ReqCPUS</td>
<td>ReqGRES</td>
<td>ReqMem</td>
</tr>
<tr>
<td>ReqNodes</td>
<td>ReqTRES</td>
<td>Reservation</td>
<td>ReservationId</td>
</tr>
<tr>
<td>Reserved</td>
<td>ResvCPU</td>
<td>ResvCPURAW</td>
<td>Start</td>
</tr>
<tr>
<td>State</td>
<td>Submit</td>
<td>Suspended</td>
<td>SystemCPU</td>
</tr>
<tr>
<td>Timelimit</td>
<td>TotalCPU</td>
<td>UID</td>
<td>User</td>
</tr>
<tr>
<td>UserCPU</td>
<td>WCKey</td>
<td>WCKeyID</td>
<td></td>
</tr>
</tbody>
</table>
### sacct – Example

```sh
sacct --format=jobid,elapsed,ncpus,ntasks,state
```

<table>
<thead>
<tr>
<th>Jobid</th>
<th>Elapsed</th>
<th>Ncpus</th>
<th>Ntasks</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>00:01:30</td>
<td>2</td>
<td>1</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>38</td>
<td>00:01:30</td>
<td>2</td>
<td>1</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>41</td>
<td>00:00:00</td>
<td>2</td>
<td>2</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>43</td>
<td>00:00:01</td>
<td>2</td>
<td>2</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>57</td>
<td>00:01:23</td>
<td>2</td>
<td>1</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>59</td>
<td>00:01:31</td>
<td>2</td>
<td>1</td>
<td>COMPLETED</td>
</tr>
</tbody>
</table>
Advanced Computing Center for Research and Education

Using SLURM
scontrol: interacting with my running job
scontrol is used to view or modify a job's configuration including holding and releasing jobs, or changing their run order.

Most of the scontrol commands can only be executed by user root.

If no command is entered on the execute line, scontrol will operate in an interactive mode and prompt for input. It will continue prompting for input and executing commands until explicitly terminated.

All commands and options are case-insensitive, although node names are case-sensitive (node names "LX" and "lx" are distinct).

All commands and options can be abbreviated to the extent that the specification is unique.

The most common uses for scontrol are to hold jobs to prevent them from running. Later to release those jobs so they may be scheduled to run.
<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scontrol show job [jobid]</code></td>
<td>Show job information.</td>
</tr>
<tr>
<td><code>scontrol hold [jobid]</code></td>
<td>Prevent a job from running.</td>
</tr>
<tr>
<td><code>scontrol release [jobid]</code></td>
<td>Release hold so job may be scheduled to run.</td>
</tr>
<tr>
<td><code>scontrol update dependency=jobid</code></td>
<td>Allow job to run after “jobid” completes.</td>
</tr>
<tr>
<td><code>scontrol show nodes</code></td>
<td>Show information about compute nodes.</td>
</tr>
</tbody>
</table>
Advanced Computing Center for Research and Education

Using SLURM

salloc: Testing and Debugging Jobs

Visit us here: www.accre.vanderbilt.edu
Follow us on Twitter: @ACCREVandy
Visit our GitHub Repos: https://github.com/accre
`salloc` is used to allocate a slurm job allocation, which is a set of resources: one or more nodes, one or more cpus, memory and time.

When `salloc` successfully obtains the requested allocation, it then runs the command, or commands you specify. Finally, when your specified command is complete, `salloc` relinquishes the job allocation.

The command may be any program that is available on the cluster, or that you have installed in your `/home`, `/data`, or `/scratch` directories.

This gives you the opportunity to debug programs and/or scripts rather than trying to write output to a log file or to the screen.
salloc: Example

salloc --nodes=1 --ntasks=1 --time=02:30:00 --mem=4G

This will ask for 1 node for 2½ hours. It also asks for 4GB of memory, and notifies slurm that just one cpu will be needed.

When the command is executed on the command line salloc will provide a jobid:

salloc: Granted job allocation 6451342

It will then pause until it is able to honor the request at which time the command line will indicate the node you have been given, and you are placed at the command line of the node.

At that point you may run the programs you wish provide that you do not exceed the 4GB of memory requested. After 2½ hours slurm will kill the job and drop you off the node.
Advanced Computing Center for Research and Education

Using SLURM

Other Useful Commands

Visit us here: www.accre.vanderbilt.edu
Follow us on Twitter: @ACCREVandy
Visit our GitHub Repos: https://github.com/accre
## Other Useful Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>qSummary</td>
<td>Shows the number of user jobs arranged by groups.</td>
</tr>
<tr>
<td>showLimits</td>
<td>Shows all the limits imposed on users and groups.</td>
</tr>
<tr>
<td>SlurmActive</td>
<td>Shows status of the cluster by node type.</td>
</tr>
<tr>
<td>sinfo</td>
<td>Show a shorthand status of all nodes in the cluster.</td>
</tr>
</tbody>
</table>
Advanced Computing Center for Research and Education

Using SLURM

Getting Help from ACCRE

Visit us here: www.accre.vanderbilt.edu
Follow us on Twitter: @ACCREVandy
Visit our GitHub Repos: https://github.com/accre
ACCRE website FAQ and Getting Started pages:
    http://www.accre.vanderbilt.edu/support
Please, read the ACCRE faq. Many questions can be resolved by simply reading the faq.
Suppose you want to download and install an R module. The faq shows the three most common scenarios for doing just that. There are instructions for python, and perl, too.
I want to use Matlab, or SAS on the cluster? Read the faq!
I have forgotten my password! What do I do? Read the faq!
I have submitted the wrong job, and I do not want it to run. How do I delete this job? Read the faq
I want to be on an 8-core node all by myself. How do I do that? Read the faq.
Getting Help from ACCRE

ACCRE Help Desk:

http://www.accre.vanderbilt.edu/support/contact/submit_RT.php

Tickets default to non-rush queue, which is for tickets about an issue which only impacts you and / or which can wait until the next business day.
Rush tickets are for issues which would impact the cluster as a whole. Rush queue pages ACCRE personnel (we normally only have staff on site during normal business hours), so please think before opening up a rush ticket at 3 AM!
If you open up a rush ticket - day or night - please plan on being available to work with ACCRE personnel to resolve the issue. Once the issue (rush or non-rush) is resolved we would appreciate it if you would let us know.
Getting Help from ACCRE

SchedMD website for complete SLURM documentation.

   http://slurm.schedmd.com

Clone our SLURM repo for job many script examples.

   https://github.com/accre/SLURM

If you will learn to use git, you can contribute job script examples, tricks, tips, even complete programs for help other users.

Join our news groups for information about using R, matlab, python and ruby on the cluster. For details visit:

   http://lists.accre.vanderbilt.edu/mailman/listinfo

For a handy list of SLURM commands check out:

   http://slurm.schedmd.com/pdfs/summary.pdf