WILLIAM & MARY
CHARTERED 1693

Running Jobs with Slurm

Guide to running jobs on main-campus / VIMS HPC cluster

Migrating from Torque

Slurm is the batch system used to submit jobs on all main-campus and VIMS HPC clusters. For those that are familiar with Torque, the following table may be helpful:

Table 1: Torque vs. Slurm commands

action	slurm	torque
submit batch job script	sbatch <script name>	qsub <script name>
launch interactive session	salloc	qstat -I
view current jobs	squeue	qstat
cancel job	scancel	qdel
launch mpi job	srun	N/A (mvp2run, mpiexec, etc.)
check queue status	squeue	qstat

Important differences between running jobs under Torque vs. Slurm:

1. **NEW - now you can ssh into any node you are using within a job from the cluster front-end/login machine.**
2. **NEW - To do mpi over ssh (which some codes require), you will need to create/modify your .ssh config file and add an ssh-key.** See our page on: Using Gaussian 16 on HPC for more information.
3. If you use a shell type other than your usual startup shell (i.e. run a batch script with bash but you use tcsh as your default shell), you will need to add "`source /usr/local/etc/sciclone.bashrc`" or "`source`

`/usr/local/etc/sciclone.cshrc`" if you are using main-campus compute clusters, or "`source /usr/local/etc/chesapeake.cshrc`" or "`source /usr/local/etc/chesapeake.bashrc`". For example, say your default shell is tcsh (the majority of users use tcsh, however you can tell for sure by entering "`echo $0`" at a prompt). To use bash in a batch script, you would add the command "`source /usr/local/etc/sciclone.bashrc`" after your #SBATCH lines in your batch script.

4. All jobs start their life in the submission directory (where you run sbatch or salloc). This is different than Torque which always started in your home directory.
5. sbatch jobs do NOT source your bashrc.XXX and will inherit the environment that exists when you run sbatch.
6. salloc DOES source your startup environment and will load your startup modules.
7. Most clusters (excluding main-campus 'bora/hima' and VIMS campus potomac and pamunkey) do not require a specific type of node (***-C***, ***--constraint***) to be specified.
8. Main-campus clusters require submission from their respective front-end. i.e. run sbatch, salloc from femto for femto jobs, kuro, for kuro jobs.
9. The main-campus 'hima' cluster jobs must be submitting from bora (its front-end / login machine). Use ***-C hi*** to specify hima, use ***-C bo*** to specify bora.
10. VIMS cluster jobs can be submitted to any VIMS cluster from either chesapeake or james using ***-C*** or ***--contstraint*** (see examples below).

Running jobs with SLURM

Like most batch systems, SLURM allows one to request compute resources (nodes, memory, gpus, etc.) and then use these resources to run executables in the compute environment. Under SLURM, there are multiple ways of doing this:

salloc - interactive session on nodes ; same as qsub -I under Torque

sbatch - submit a batch script for execution ; same as qsub under Torque

srun - run MPI job or a single command on a set of resources

The following table lists the common options for selecting compute resources:

Table 2: Controlling slurm resources:

option	description	notes

-N, --nodes	How many nodes	
-n, --ntasks	how many cores	this is the total # cores for the job
--ntasks-per-node	how many cores/node	this is the same as ppn=X in Torque
-c, --cpus-per-task	how many core used per task	For hybrid calcs, how many OpenMP threads per MPI task
-t, --time	specify walltime	"minutes", "minutes:seconds", "hours:minutes:seconds", "days-hours", "days-hours:minutes" and "days-hours:minutes:seconds" The max walltime for most clusters is 72hrs. The exceptions are: kuro - 48 hrs pamunkey, ptotomac - 180 hrs
--mem=<size>[units]	specify the memory required per node	'[units]' can be K,M,G or T.
-J, --job-name	name of job	
--x11	forward X display to job	
-o <file>	change location of stdout	defaults to both stdout and stderr in the same slurm-<JOBID>.out file
-e <file>	change location of stdin	
--mail-type	which events cause email	NONE, BEGIN, END, FAIL, REQUEUE, ALL ; default is NONE
--mail-user	email address for user	gmail addresses will not work.

-C, --constraint	selected a nodes base on node features	Only necessary for VIMS cluster and main-campus bora cluster for now.
-G, --gpus	select the number of gpus	
-a <low>-<high>	create an array job with indicies from <low> to <high>	see detailed example below

salloc:

salloc requests resources for an interactive session. Here are some examples:

Table 3: salloc options

command	request	notes
salloc -N1 -n64 -t 1-0	one node with 64 cores for one day (kuro)	
salloc -N1 -n20 -t 30:00 -C bo	one node with 20 cores for 30 minutes (bora)	-C bo should be used to ensure bora nodes are used
salloc -N10 --ntasks-per-node=32 -t 1:00:00	ten nodes with 32 cores per node for one hour (femto)	
salloc -N1 -n32 -t 1-0 -C hi	1 node with 32 cores for 1 day on a hima node	-C hi should be used to ensure hima nodes are used
salloc -N1 -n8 -t 1-0 --gpus=1	1 node with 8 cores for 1 day with a GPU (astral, gulf)	
salloc -N1 -n8 -t 1-0 --gpus=1 -C hi	1 node with 8 cores for 1 day with a GPU (hima)	

salloc -N1 -n20 -t 1-0 -C bo		
salloc -N1 -n8 -t 1-0 --gpus=1 -C p100	1 node with 8 cores for 1 day and a P100 GPU (hima)	
salloc -N2 -n2 -c 16 -t 1-0	2 nodes with 2 mpi tasks and 16 OpenMP tasks per node for 1 day	
salloc -N1 -n20 -t 1-0 -C pt	1 node with 20 cores and 1 day on a VIMS potomac node	-C pt is necessary to access pt nodes
salloc -N1 -n64 -t 1-0 -C pm	1 node with 64 cores and 1 day on a VIMS pamunkey node	-C pm is necessary to access pm nodes
salloc -N5 -n 100 -t 30:00 -C jm	5 nodes with 100 total cores (20/node) for 30 min on james	-C jm is the default, but can be specified

sbatch:

sbatch submits a script to the batch system that will run once resources are available. All batch scripts consists of two sections, the SLURM directives and the commands the user wants to run. Here are some examples:

Table 4: sbatch examples:

script	notes

`#!/bin/tcsh`
`#SBATCH --job-name=serial`
`#SBATCH -N 1 -n1`
`#SBATCH -t 0:30:00`
 	
./a.out_serial
	Run single core job. Will work on most clusters.

#!/bin/tcsh #SBATCH --job-name=kurotest #SBATCH -N 5 --ntasks-per-node 64 #SBATCH -t 1-0 srun ./a.out_parallel	Run a job on the kuro cluster (only works on kuro since it is the only cluster with 64 cores/node)
#!/bin/tcsh #SBATCH --job-name=boratest #SBATCH -N 10 --ntasks-per-node 20 #SBATCH -t 0:30:00 ./a.out	Run a 10 node / 200 core job on bora.
#!/bin/tcsh #SBATCH --job-name=himatest #SBATCH -N 1 -n 32 -C hi #SBATCH -t 0:30:00 ./a.out	Run a 32 core job on a hima node
#!/bin/tcsh #SBATCH --job-name=potomac_test #SBATCH -N 1 -n 12 -C pt #SBATCH -t 0:30:00 ./a.out	Run a 1 node / 12 core job on potomac
#!/bin/bash #SBATCH --job-names=james_test #SBATCH -N 5 --ntasks-per-node 20 -C jm #SBATCH -t 2-0 ./a.out	Run a 5 node / 100 core job on james for 2 days (-C jm isn't actually necessary since this is the default) Note that in this sript, 'bash' syntax is used instead of tcsh. Both options are available for batch scripts.

srun:

srun is the command for launching mpi jobs within SLURM. It takes the place of mpirun, mvp2run, mpiexec for SLURM systems. srun accepts the same arguments that sbatch and salloc take. However, for the vast majority of cases, srun will not need arguments since the batch script or salloc command line has already selected the resources srun will use.

srun can also be used to run a command (parallel or serial) on a set of resources from the front-end/login machine directly. This second use is mainly for testing purposes

e.g. srun -N1 -n8 hostname # will run 'hostname' from the front-end on one node and 8 cores.

Other useful slurm commands:

Table 5: other useful SLURM commands

Scroll right for more

command	description	notes
sinfo	node statuses	
scontrol show node [nodename]	show details of a particular node	all nodes are shown if no nodename is given
scontrol show job [jobid]	show details of a particular job	all jobs are shown if no jobid is given
seff	show cpu/memory usage of completed job	

array jobs:
Array jobs are a series of jobs that each get a unique array id that can be used within your job script. This is useful for parameters studies where you have a number of different input files to work through. For instance, imagine a scenario in which you have ten input files named INPUT.1 through INPUT.10. You want to submit 10 jobs, each of which runs a different input file. To do this you can submit the following script:

Array job example

```
#!/bin/tcsh
#SBATCH --job-name=array_test
#SBATCH -N 1 -n 1
#SBATCH -t 0:30:00
#SBATCH -a 1-10


./a.out  < INPUT.$SLURM_ARRAY_TASK_ID
```

This will run 10 jobs, each using 1 core on 1 node for 30 minutes. Each array job will substitute its value of $SLURM_ARRAY_TASK_ID.

Standing reservations for debugging / short tests:
There are currently no nodes on any subcluster resevered for debug/short jobs. Please send email to hpc-help@wm.edu if you need some room to run short tests/debug jobs.

Checking load on job
For jobs that use whole nodes, i.e. MPI/parallel jobs, it is useful to be able to check the load before running. The script ***ckload*** can be run at the beginning of a job to report the high load on any nodes in the job and, optionally, kill the job so it can be submitted again.

ckload output and use

```
>> ckload -h

usage: ckload [-h] [-X] [-v] maxload

check load

positional arguments:
maxload check load and warn if too high

optional arguments:
-h, --help show this help message and exit
-X, --kill check load and die if too high
-v, --verbose increase verbosity
```

```
#!/bin/tcsh
#SBATCH --job-name=kurotest
#SBATCH -N 5 --ntasks-per-node 64
#SBATCH -t 1-0

ckload -X 0.05

srun ./a.out_parallel
```

In the above batch script example, the job is killed if any of the allocated five nodes have a 1-min avg load larger than 0.05.