

# Introduction to SLURM & SLURM batch scripts

Anita Orendt

Assistant Director

Research Consulting & Faculty Engagement

[anita.orendt@utah.edu](mailto:anita.orendt@utah.edu)

16 June 2020

# Overview of Talk

- Basic SLURM commands
- SLURM batch directives
- Accounts and Partitions
- SLURM Environment Variables
- SLURM Batch scripts
- Running an Interactive Batch job
- Monitoring Jobs
- Where to get more Information

# Basic SLURM commands

- **sinfo** - shows partition/node state
- **sbatch <scriptname>** - launches a batch script
- **squeue** - shows all jobs in the queue
  - **squeue -u <username>** - shows only your jobs
- **scancel <jobid>** - cancels a job

## Notes:

For **sinfo**, **squeue** – can add **-M all** to see all clusters using given slurm installation (notchpeak, kingspeak, lonepeak, ember, ash)

Can also add **-M cluster** OR use full path

/uufs/<cluster>.peaks/sys/pkg/slurm/std/bin/<command> to look at the queue, or submit or cancel jobs for a different cluster

# Some Useful Aliases

- Bash to add to .aliases file:

```
alias si="sinfo -o \" %20P %5D %14F %8z %10m %10d %11l %16f %N\""
```

```
alias si2="sinfo -o \" %20P %5D %6t %8z %10m %10d %11l %16f %N\""
```

```
alias sq="squeue -o \" %8i %12j %4t %10u %20q %20a %10g %20P %10Q %5D %11l %11L %R\""
```

- Tcsh to add to .aliases file:

```
alias si 'sinfo -o " %20P %5D %14F %8z %10m %11l %16f %N"
```

```
alias si2 'sinfo -o " %20P %5D %6t %8z %10m %10d %11l %N"
```

```
alias sq 'squeue -o " %8i %12j %4t %10u %20q %20a %10g %20P %10Q %5D %11l %11L %R"
```

Can add **-M** to **si** and **sq** also

You can find these on the CHPC Slurm page

<https://www.chpc.utah.edu/documentation/software/slurm.php#aliases>

# SLURM Batch Directives

- #SBATCH --time 1:00:00 ← wall time of a job (or -t)
- #SBATCH --partition=name ← partition to use (or -p)
- #SBATCH --account=name ← account to use (or -A)
- #SBATCH --nodes=2 ← number of nodes (or -N)
- #SBATCH --ntasks 32 ← total number of tasks (or -n)
- #SBATCH --mail-type=FAIL,BEGIN,END ← events on which to send email
- #SBATCH --mail-user=name@example.com ← email address to use
- #SBATCH -o slurm-%j.out-%N ← name for stdout; %j is job#, %N node
- #SBATCH -e slurm-%j.err-%N ← name for stderr; %j is job#, %N node
- #SBATCH --constraint "C20" ← can use features given for nodes (or -C)

# Accounts & Partitions

- You need to specify an **account** and a **partition** to run jobs
- You can see a list of partitions using the `sinfo` command
- For general allocation usage the partition is the cluster name
- If no allocation (or out of allocation) use *clustername*-freecycle for partition
- Your account is typically your PI's name (e.g., if your PI is Baggins, use the "baggins" account) – there are a few exceptions!
- Private node accounts and partition have the same name – PI last name with cluster abbreviation, e.g., baggins-kp, baggins-np, etc
- Private nodes can be used as a guest using the "owner-guest" account and the *cluster*-guest partition
- Remember general nodes on notchpeak need allocation; general nodes on kingspeak , lonepeak and tangent are open to all users without allocation

# More on Accounts & Partitions

Allocations and node ownership status	What resource(s) are available
No general allocation, no owner nodes	<u>Unallocated general nodes</u> <u>Allocated general nodes in freecycle mode</u> - not recommended <u>Guest access on owner nodes</u>
General allocation, no owner nodes	<u>Unallocated general nodes</u> <u>Allocated general nodes</u> <u>Guest access on owner nodes</u>
Group owner nodes, no general allocation	<u>Unallocated general nodes</u> <u>Allocated general nodes in freecycle mode</u> - not recommended <u>Group owned nodes</u> <u>Guest access on owner nodes of other groups</u>
Group owner node, general allocation	<u>Unallocated general nodes</u> <u>Allocated general nodes</u> <u>Group owned nodes</u> <u>Guest access on owner nodes of other groups</u>

See <https://www.chpc.utah.edu/documentation/guides/index.php#parts>

# Query your accounts & partitions

```
~]$ sacctmgr -p show assoc user=u0028729 format=cluster,account,partition,qos
```

```
Cluster|Account|Partition|QOS|
```

```
notchpeak|chpc||notchpeak|
```

```
kingspeak|kingspeak-gpu||kingspeak-gpu|
```

```
ember|ember-gpu||ember-gpu|
```

```
ash|smithp-guest||ash-guest,ash-guest-res|
```

```
lonepeak|chpc||lonepeak|
```

```
kingspeak|chpc||kingspeak|
```

```
lonepeak|owner-guest||lonepeak-guest|
```

```
ember|owner-guest||ember-guest|
```

```
kingspeak|owner-guest||kingspeak-guest|
```

```
kingspeak|owner-gpu-guest||kingspeak-gpu-guest|
```

```
ember|chpc||ember|
```

***Note that partition field is empty – for the most part partition and qos are paired***



# Query your allocation

~]\$ myallocation

You have a **general** allocation on **kingspeak**. Account: **chpc**, Partition: **kingspeak**

You can use **preemptable** mode on **kingspeak**. Account: **owner-guest**, Partition: **kingspeak-guest**

You can use **preemptable GPU** mode on **kingspeak**. Account: **owner-gpu-guest**, Partition: **kingspeak-gpu-guest**

You have a **GPU** allocation on **kingspeak**. Account: **kingspeak-gpu**, Partition: **kingspeak-gpu**

You have a **general** allocation on **notchpeak**. Account: **chpc**, Partition: **notchpeak**

You can use **preemptable GPU** mode on **notchpeak**. Account: **owner-gpu-guest**, Partition: **notchpeak-gpu-guest**

You can use **preemptable** mode on **notchpeak**. Account: **owner-guest**, Partition: **notchpeak-guest**

You have a **GPU** allocation on **notchpeak**. Account: **notchpeak-gpu**, Partition: **notchpeak-gpu**

You have a **general** allocation on **ember**. Account: **chpc**, Partition: **ember**

You can use **preemptable** mode on **ember**. Account: **owner-guest**, Partition: **ember-guest**

You have a **GPU** allocation on **ember**. Account: **ember-gpu**, Partition: **ember-gpu**

You have a **general** allocation on **lonepeak**. Account: **chpc**, Partition: **lonepeak**

You can use **preemptable** mode on **lonepeak**. Account: **owner-guest**, Partition: **lonepeak-guest**

# Node Sharing

- Use the shared partition for a given set of nodes (using normal account for that partition)

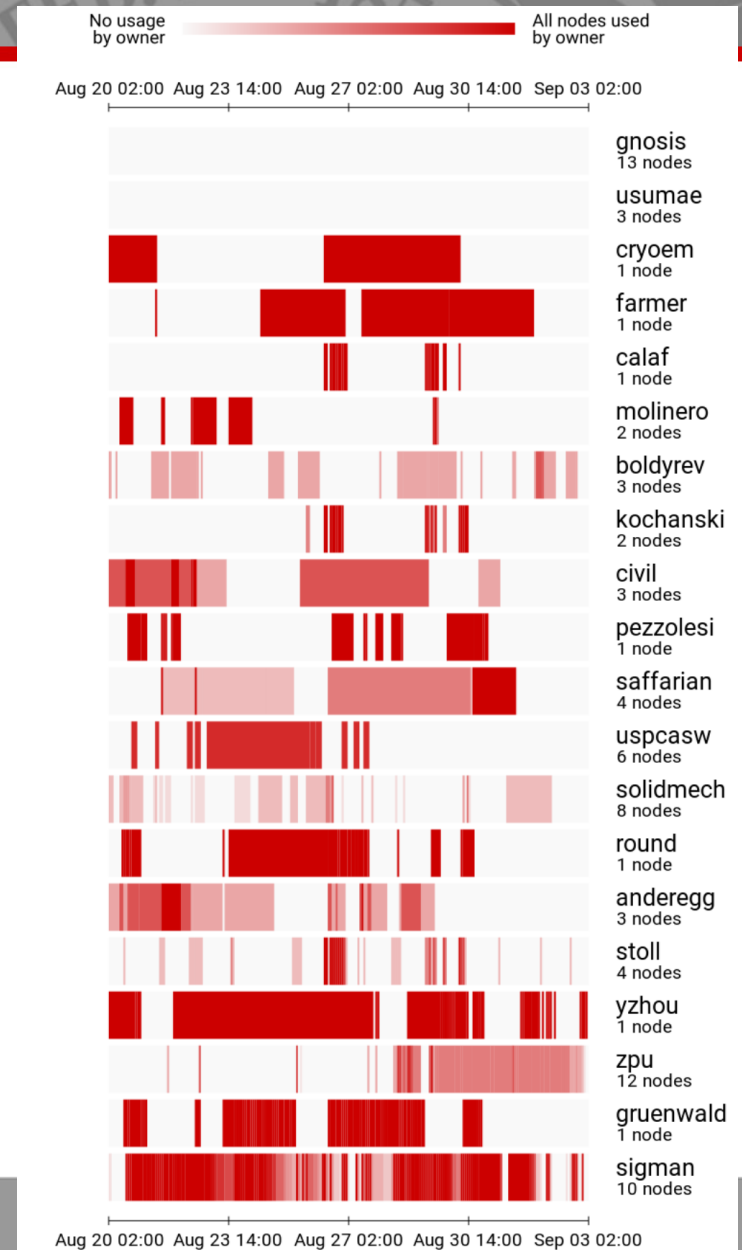
– notchpeak*	4	4/0/0/4	2:16:2	95000	1800000	3-00:00:00	chpc,c32,m96	notch[005-008]
– notchpeak-shared	4	4/0/0/4	2:16:2	95000	1800000	3-00:00:00	chpc,c32,m96	notch[005-008]
– molinero-np	2	2/0/0/2	2:16:2	95000	1800000	3-00:00:00	molinero,c32,m96	notch[024-025]
– molinero-shared-np	2	2/0/0/2	2:16:2	95000	1800000	3-00:00:00	molinero,c32,m96	notch[024-025]

- In script:
  - #SBATCH --partition=cluster-shared
  - #SBATCH --ntasks=2
  - #SBATCH --mem=32G
- If there is no memory directive used the default is that 2G/core will be allocated to the job.
- Allocation usage of a shared job is based on the percentage of the cores and the memory used, whichever is higher

<https://www.chpc.utah.edu/documentation/software/node-sharing.php>

# Owner/Owner-guest

- CHPC provides heat maps of usage of owner nodes by the owner over last two weeks
- <https://www.chpc.utah.edu/usage/constraints/>
- Use information provided to target specific owner partitions with use of constraints (more later)



# SLURM Environment Variables

- Depends on SLURM Batch Directives used
- Can get them for a given set of directives by using the `env` command inside a script (or in a `srun` session).
- Some useful environment variables:
  - `$SLURM_JOBID`
  - `$SLURM_SUBMIT_DIR`
  - `$SLURM_NNODES`
  - `$SLURM_NTASKS`

# Basic SLURM script flow

1. Set up the #SBATCH directives for the scheduler to request resources for job
2. Set up the working environment, by loading appropriate modules
3. If necessary, add any additional libraries or programs to \$PATH and \$LD\_LIBRARY\_PATH, or set other environment needs
4. Set up temporary/scratch directories if needed
5. Switch to the working directory (often group/scratch)
6. Run the program
7. Copy over any results files needed
8. Clean up any temporary files or directories

# Basic SLURM script - bash

```
#!/bin/bash
#SBATCH --time=02:00:00
#SBATCH --nodes=1
#SBATCH -o slurmjob-%j.out-%N
#SBATCH -e slurmjob-%j.err-%N
#SBATCH --account=owner-guest
#SBATCH --partition=kingspeak-guest
#Set up whatever package we need to run with
module load somemodule
#set up the temporary directory
SCRDIR=/scratch/general/lustre/$USER/$SLURM_JOBID
mkdir -p $SCRDIR
#copy over input files
cp file.input $SCRDIR/.
cd $SCRDIR
#Run the program with our input
myprogram < file.input > file.output
#Move files out of working directory and clean up
cp file.output $HOME/.
cd $HOME
rm -rf $SCRDIR
```

# Basic SLURM script - tcsh

```
#!/bin/tcsh
#SBATCH --time=02:00:00
#SBATCH --nodes=1
#SBATCH -o slurmjob-%j.out-%N
#SBATCH -e slurmjob-%j.err-%N
#SBATCH --account=owner-guest
#SBATCH --partition=kingspeak-guest

#Set up whatever package we need to run with
module load somemodule

#set up the scratch directory
set SCRDIR /scratch/local/$USER/$SLURM_JOBID
mkdir -p $SCRDIR

#move input files into scratch directory
cp file.input $SCRDIR/.
cd $SCRDIR

#Run the program with our input
myprogram < file.input > file.output

#Move files out of working directory and clean up
cp file.output $HOME/.
cd $HOME
rm -rf $SCRDIR
```

# Parallel Execution

- MPI installations at CHPC are SLURM aware, so mpirun will usually work without a machinefile (unless you are manipulating the machinefile in your scripts)
- If machinefile or host list needed, create the node list:
  - `srun hostname | sort -u > nodefile.$SLURM_JOBID`
  - `srun hostname | sort > nodefile.$SLURM_JOBID`
- Alternatively, you can use the srun command instead, but you need to compile with a more recently compiled MPI
- Mileage may vary, and for different MPI distributions, srun or mpirun may be preferred (check our slurm page on the CHPC website for more info or email us)



# Running interactive batch jobs

- An interactive command is launched through the `srun` command

```
srun --time=1:00:00 --nodes=1 --account=chpc  
      --partition=ember --pty /bin/tcsh -l
```
- Launching an interactive job automatically forwards environment information, including X11 forwarding
- "`--pty`" must be set to shell preferred for the session (either `/bin/tcsh` or `/bin/bash`)
- `-l` (lower case "L") at the end required

*OpenOnDemand is another option (Thursday March 5)*

# Slurm for use of GPU Nodes

- Ember – 8 GPU nodes
  - All are general nodes, each with M2090 cards
- Kingspeak – 8 GPU nodes
  - 4 general nodes, two with 4 Tesla K80 cards (8 GPUs) each, two with 8 GeForce TitanX cards each
  - 4 owner nodes each with 2 Tesla P100 cards (owned by School of Computing)
- Notchpeak – 13 GPU nodes
  - 3 general nodes each with 3 Tesla V100 cards; 3 general nodes each with 4 RTX2080Ti cards; 1 general node with 2 RTX2080Ti plus 1 P40 card
  - 1 owner node with 4 TitanV cards
  - 1 owner node with 8 GTX1080Ti cards
  - 4 owner nodes each with 4 RTX2080Ti cards
- Use partition and account set to **cluster-gpu** (for general) or **cluster-gpu-guest** for guest jobs on owner
- Must get added to the gpu accounts – request via [helpdesk@chpc.utah.edu](mailto:helpdesk@chpc.utah.edu)
- Use only if you are making use of the GPU for the calculation
- Most codes do not yet make efficient use of multiple GPUs so we have enabled node sharing
- See <https://www.chpc.utah.edu/documentation/guides/gpus-accelerators.php>

# Node Sharing on GPU nodes

- Need to specify number of CPU cores, amount of memory, and number of GPU
- Core hours used based on highest % requested among cores, memory and GPUs

Option	Explanation
#SBATCH --gres=gpu:k80:1	request one K80 GPU (others types names are titanx, m2090, p100, v100, titanv, 1080ti, 2080ti, p40)
#SBATCH --mem=4G	request 4 GB of RAM (default is 2GB/core if not specified)
#SBATCH --mem=0	request all memory of the node; use this if you do not want to share the node as this will give you all the memory
#SBATCH --tasks=1	requests 1 core

# Strategies for Serial Applications

- <https://www.chpc.utah.edu/documentation/software/serial-jobs.php>
- When running serial applications (no MPI, no threads) unless memory constraint, you should look to options to bundle jobs together so using all cores on nodes
- There are multiple ways to do so
  - `srun --multi-prog`
  - submit script
- Also consider OpenScienceGrid (OSG) as an option (especially if you have a large number of single core, short jobs)

# Strategies for Job Arrays

- <https://www.chpc.utah.edu/documentation/software/slurm.php#jobarr>
- Useful if you have many similar jobs when each use all cores on a node or multiple nodes to run where only difference is input file
- `sbatch --array=1-30%n myscript.sh` – where `n` is maximum number of jobs to run at same time
- In script: use `$SLURM_ARRAY_TASK_ID` to specify input file:
  - `./myprogram input$SLURM_ARRAY_TASK_ID.dat`

# Job Priorities

- <https://www.chpc.utah.edu/documentation/software/slurm.php#priority>
- **sprio** give job priority for all jobs
  - sprio -j JOBID for a given job
  - sprio -u UNID for all a given user's jobs
- Combination of three factors added to base priority
  - Time in queue
  - Fairshare
  - Job size
- New – only 5 jobs per user per qos will accrue priority based on time on queue

# Checking Job Performance

- With an active job – can ssh to node
  - Useful commands, top, ps, sar
- Also from interactive node can query job
  - `/uufs/chpc.utah.edu/sys/installdir/pestat/pestat`
- Can query node status
  - `scontrol show node notch024`
- The day after the job ends – XDMMoD Supremm

# Checking job efficiency

```
~]$ pestat -u $USER
```

Hostname	Partition	Node	Num_CPU	CPUload	Memsize	Freemem	Joblist
		State	Use/Tot		(MB)	(MB)	JobId User ...
kp016	kingspeak*	alloc	16 16	1.00*	64000	55494	7430561 u0123456
kp378	schmidt-kp	alloc	28 28	.00*	256000	250656	7430496 u0123456



# Slurm Documentation at CHPC

<https://www.chpc.utah.edu/documentation/software/slurm.php>

<https://www.chpc.utah.edu/documentation/software/serial-jobs.php>

<https://www.chpc.utah.edu/documentation/software/node-sharing.php>

<https://www.chpc.utah.edu/usage/constraints/>

<https://www.chpc.utah.edu/documentation/guides/index.php#GenSlurm>

## Other good documentation sources

<http://slurm.schedmd.com/documentation.html>

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

<http://www.schedmd.com/slurmdocs/rosetta.pdf>

# Getting Help

- CHPC website
  - [www.chpc.utah.edu](http://www.chpc.utah.edu)
    - Getting started guide, cluster usage guides, software manual pages, CHPC policies
- Service Now Issue/Incident Tracking System
  - Email: [helpdesk@chpc.utah.edu](mailto:helpdesk@chpc.utah.edu)
- Help Desk: 405 INSCC, 581-6440 (9-6 M-F)
- We use [chpc-hpc-users@lists.utah.edu](mailto:chpc-hpc-users@lists.utah.edu) for sending messages to users