Introduction to SLURM & SLURM batch scripts

Zhiyu (Drew) Li & Anita Orendt
Research Consulting & Faculty Engagement
Center for High Performance Computing
{zhiyu.li; anita.orendt}@utah.edu
Overview of Talk

- What is SLURM
- Accounts and Partitions
- Basic SLURM Commands
- Node Sharing
- SLURM batch directives
- SLURM Environment Variables
- Running an Interactive Batch job
- Monitoring Jobs
- Where to get more Information
What is SLURM

• Formerly known as Simple Linux Utility for Resource Management
• Open-source workload manager for supercomputers/clusters
  – Manage resources (nodes/cores/memory/interconnect/gpus)
  – Schedule jobs (queueing/prioritization)
• Used by 60% of the TOP500 supercomputers¹
• Fun fact: development team based in Lehi, UT

Partitions & Accounts

- **Partition**: a group of nodes that a job can be scheduled on. A node can belong to more than one partition, and each partition can be configured to enforce different resource limits and policies.

- **Account**: to limit and track resource utilization at user/group level. A user/group can have multiple Slurm accounts – each represents different privileges.

- To run a job on CHPC, you need to specify a pair of a **Partition** and an **Account**. (How to find out? -- There are 3 commands! More on this later)
Basic SLURM commands

- **sinfo** - shows all partitions/nodes state
  - **mysinfo** - info on partitions/nodes and associated accounts you have access to on the cluster *(Method 1)*

- **squeue** - shows all jobs in queue
  - **squeue** -u <username> - shows only your jobs
  - **mysqueue** - shows job queue per partition and associated accounts you have access to on the cluster *(Method 2)*

- **sbatch** <scriptname.sbatch> - launch a batch job

- **scancel** <jobid> - cancel a job

- **salloc** – start an interactive job

*CHPC developed programs. See [CHPC Newsletter 2023 Summer](#)

For **sinfo, mysinfo, squeue, mysqueue** – can use –M <ClusterName> (notchpeak, kingspeak, lonepeak, ash)

Redwood (PE) has own slurm setup, separate from others
Some Useful Aliases

- **Bash** to add to `.aliases` file:
  
  ```bash
  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\""
  ```

- **Csh/Tcsh** to add to `.aliases` file:
  
  ```bash
  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"
  ```

See: [https://www.chpc.utah.edu/documentation/software/slurm.php#aliases](https://www.chpc.utah.edu/documentation/software/slurm.php#aliases)

- si/si2 – check node specifications (CPU, Memory, GPU, PI)
- sq – check job priority, assigned nodes, reason/error…
Partitions & Accounts

- To run a job on CHPC, you need to specify a pair of a **Partition** and an **Account**.
  - Commands to check valid pairs:
    - `myinfo`, `mysqueue`, `myallocation` (Method 3, gives info on all clusters)

- **CHPC Cluster Partition Naming Convention**
  - `<CluserName>`: *notchpeak, kingspeak, lonepeak* → general nodes (allocation required on notchpeak)
  - `<CluserName>-freeycle`: *notchpeak-freeycle* → general nodes - preemptable
  - `<PILastName>-<ClusterCode>`: *baggins-np (-kp; -lp)* → owner nodes (PI/Dept-specific)
  - `<ClusterName>-guest`: *notchpeak-guest* → owner nodes (from all PIs) - preemptable
  - Variants: -gpu; -shared;
    - notchpeak-shared: general nodes on notchpeak run in Shared mode (more on this later)
    - baggins-gpu-kp: owner GPU nodes on kingspeak
## More on Accounts & Partitions

<table>
<thead>
<tr>
<th>Awarded allocations and node ownership status</th>
<th>What resource(s) are available (recommendation high to low)</th>
</tr>
</thead>
</table>
| No awarded general allocation (notchpeak), no owner nodes | Unallocated general nodes (eg kingspeak, lonepeak)  
Guest access on owner nodes  
Allocated general nodes in freecycle mode (notchpeak) - not recommended |
| Awarded general allocation, no owner nodes | Allocated general nodes (notchpeak)  
Unallocated general nodes (eg kingspeak, lonepeak)  
Guest access on owner nodes |
| Group owner nodes, no awarded general allocation | Group owned nodes  
Unallocated general nodes (eg kingspeak, lonepeak)  
Guest access on owner nodes of other groups  
Allocated general nodes in freecycle mode (notchpeak) - not recommended |
| Group owner node, awarded general allocation | Group owned nodes  
Allocated general nodes (notchpeak)  
Unallocated general nodes (eg kingspeak, lonepeak)  
Guest access on owner nodes of other groups |

See [https://www.chpc.utah.edu/documentation/guides/index.php#parts](https://www.chpc.utah.edu/documentation/guides/index.php#parts)
Node Sharing

- A partition can be configured to run jobs in 2 modes: **Exclusive** V.S. **Shared**
- **Exclusive partition**: Slurm gives whole node(s) (all CPU cores) to your job (and you will be charged on whole nodes);
- **Shared partition**: Slurm gives a portion of node (CPU core & Memory) as requested; The remain resources can be used by other jobs; (you will be charged on the portion of the node)
- **How to tell on CHPC clusters** -- Identifiable by partition names
  - **Exclusive**: notchpeak, kingspeak, baggins-np, baggins-kp
  - **Shared**: notchpeak-shared, kingpeak-shared-guest, baggins-shared-kp
  - Exception: GPU partitions are all in Shared mode (even no ‘—shared” in names) on CHPC: notchpeak-gpu
- **Use Shared Partition wherever possible**
  - Save your group allocations/credits
  - Shorten queueing time for You and Others: allow multiple jobs on same node
  - Help increase utilization and save energy/environment
  - CHPC may reach out to you to promote resources sharing

[https://www.chpc.utah.edu/documentation/software/node-sharing.php](https://www.chpc.utah.edu/documentation/software/node-sharing.php)
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) and node feature list
SLURM Batch Directives

#SBATCH --time 1:00:00 ← wall time of a job (or -t) in hour:minute:second
#SBATCH --partition=name ← partition to use (or -p)
#SBATCH --account=name ← account to use (or -A)
#SBATCH --nodes=1 ← number of nodes (or -N)
#SBATCH --ntasks=32 ← total number of tasks (cpu cores) (or -n)
#SBATCH --mem=128GB ← memory per node

#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
Guest Job -- Target on Owner nodes

#SBATCH --time 10:00:00
#SBATCH --partition=notchpeak-shared-guest
#SBATCH --account=owner-guest
#SBATCH --nodes=1
#SBATCH --ntasks=32
#SBATCH --mem=128GB

#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --mail-user=name@example.com
#SBATCH -o slurm-%j.out-%N
#SBATCH -e slurm-%j.err-%N
#SBATCH --constraint "<Owner-Nodes-Label-Found-On-Chart>"
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
#!/bin/bash
#SBATCH --time=02:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --mem=32G
#SBATCH -o slurmjob-%j.out-%N
#SBATCH -e slurmjob-%j.err-%N
#SBATCH --account=owner-guest
#SBATCH --partition=kingspeak-shared-guest

#Set up whatever package we need to run with
module load <some-modules>

#set up the temporary directory
SCRDIR=/scratch/general/vast/$USER/$SLURM_JOB_ID
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

# Save the script as XXXXX.sbatch
# submit it
sbatch XXXXX.sbatch
# slurm returns a <jobid>
squeue -u <jobid>
Basic SLURM script - tcsh

#!/bin/tcsh

#SBATCH --time=02:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --mem=32G
#SBATCH -o slurmjob-%j.out-%N
#SBATCH -e slurmjob-%j.err-%N
#SBATCH --account=owner-guest
#SBATCH --partition=kingspeak-shared-guest

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

#set up the scratch directory
set SCRDIR /scratch/local/$USER/$SLURM_JOB_ID
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
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_JOB_ID
  - $SLURM_SUBMIT_DIR
  - $SLURM_NNODES
  - $SLURM_NTASKS

See: https://slurm.schedmd.com/sbatch.html#SECTION_OUTPUT-ENVIRONMENT-VARIABLES
Slurm for use of GPU Nodes

- GPU nodes are on lonepeak, kingspeak, notchpeak (and redwood in the PE)
- Info on GPU nodes found at https://chpc.utah.edu/documentation/guides/gpus-accelerators.php
- There are both general (open to all users) and owner GPU nodes (available via owner-gpu-guest, with preemption, to all uses)
- At this time, general GPU nodes are run without allocation (no charge)
  - Must get added to the gpu accounts – Request via helpdesk@chpc.utah.edu
- GPU partitions set up in a shared mode only as most codes do not yet make efficient use of multiple GPUs so we have enabled node sharing
- Use only if you are making use of the GPU for the calculation
Node Sharing on GPU nodes

- In addition to submitting to a GPU partition, at least you need to specify flag "--gres=gpu", number of CPU cores, amount of memory.

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>#SBATCH --gres=gpu:p100:1</td>
<td>request one p100 GPU (others types names are titanx, rtx3090, p100, v100, titanv, 1080ti, 2080ti, p40, t4, a40,a100)</td>
</tr>
<tr>
<td>#SBATCH --mem=4G</td>
<td>request 4 GB of RAM (default is 2GB/core if not specified)</td>
</tr>
<tr>
<td>#SBATCH --mem=0</td>
<td>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</td>
</tr>
<tr>
<td>#SBATCH --ntasks=1</td>
<td>request 1 cpu core</td>
</tr>
</tbody>
</table>
GPU Job

#SBATCH --time 10:00:00
#SBATCH --partition=notchpeak-gpu-guest
#SBATCH --account=owner-gpu-guest
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --mem=16G
#SBATCH --gres=gpu:a100:1

#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --mail-user=name@example.com
#SBATCH -o slurm-%j.out-%N
#SBATCH -e slurm-%j.err-%N
Running interactive batch jobs

• An interactive command is launched through the `salloc` command

  `salloc --time=8:00:00 --ntasks=4 --nodes=1 --mem=16G --account=<account> --partition=kingspeak-shared`

  `salloc --time=8:00:00 --ntasks=4 --nodes=1 --mem=16GB --account=notchpeak-gpu --partition=notchpeak-gpu --gres=gpu`

• Use of FastX connection is highly recommended
  – support GUI applications
  – keep your sessions alive

*OpenOnDemand is another option to start interactive sessions*
Strategies for Job Arrays

- [https://www.chpc.utah.edu/documentation/software/slurm.php#jobarr](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](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

- Only 5 jobs per user per slurm account (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, atop
  – Also from interactive node can query job
    • /uufs/chpc.utah.edu/sys/installdir/pestat/pestat
  – Can query node status
    • scontrol show node notch024

• After job complete -- XDMoD Supremm
  – Job level data available day after job ends
  – XDMoD sites https://xdmod.chpc.utah.edu and https://pe-xdmod.chpc.utah.edu
  – usage info:
    https://www.chpc.utah.edu/documentation/software/xdmod.php
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
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

- We use [chpc-hpc-users@lists.utah.edu](mailto:chpc-hpc-users@lists.utah.edu) for sending messages to users