SLURM 101

or How I Learned to Stop Worrying and Love the Scheduler

Josh Sonstroem
Sr. Infrastructure Engineer
jsonstro@ucsc.edu
What is SLURM

https://slurm.schedmd.com/

SLURM is an open source, fault-tolerant, highly scalable cluster management and job scheduling system for *NIX clusters.

**SLURM has 3 distinct features**

1. Allocates access to compute node resources for some period of time
2. Offers a framework for starting, executing and monitoring (often parallel) jobs on those allocated resources
3. Arbitrates contention for those resources by utilizing a queueing system to manage pending work

**What’s in the name?** Simple Linux Utility for Resource Management (SLURM)
SLURM Architecture

SLURM’s architecture supports a distributed management model with hierarchical design and high-availability failover capabilities.

It also provides a general-purpose plugin mechanism available to easily support various infrastructures.

One or more physical and/or logical clusters can be managed by a single controller.
SLURM Components

The 4 elements of SLURM’s distributed architecture:

1. A centralized manager, `slurmctld`, to monitor resources and work
2. A backup manager in HA configurations
3. Each compute server (node) has a `slurmd` daemon, which can be compared to a remote shell awaiting work, executing it and returning
4. An optional `slurmdbd` (SLURM DataBase Daemon) which can be used to record accounting information

*HB* uses the backfill scheduler calculation to fit properly sized and time-limited jobs into the queues
SLURM Vocab:

- Nodes
- Partitions
- Jobs
- Job Steps
SLURM Vocab:

- **Nodes**
- **Partitions**
- **Jobs**
- **Job Steps**

**Nodes:**
- Baseboards, Sockets, Cores, Threads
- CPUs (Core or thread)
- Memory size
- Generic resources
- Features
- State
  - Idle
  - Mix
  - Alloc
  - Completing
  - Drain / ing
  - Down
SLURM Vocab:

- Nodes
- Partitions
- Jobs
- Job Steps

Partitions:

- Associated with specific set of nodes
- Nodes can be in more than one partition
- Job size and time limits
- Access control list
- State information
  - Up
  - Drain
  - Down
SLURM Vocab:

- Nodes
- Partitions
- Jobs
- Job Steps

Jobs:

- ID (a number)
- Name
- Time limit
- Size specification
- Node features required
- Other Jobs Dependency
- Quality Of Service (QoS)
- State (Pending, Running, Suspended, Canceled, Failed, etc.)
SLURM

Vocab:
- Nodes
- Partitions
- Jobs
- Job Steps

Jobs Step:
- ID (a number)
- Name
- Time limit (maximum)
- Size specification
- Node features required in allocation
On *Hummingbird* the SLURM command line (CLI) tools are loaded by default upon user login.

These include, but are not limited to:

1. **srun** and **sbatch** to initiate different types of compute jobs
2. **scancel** to terminate queued or running jobs
3. **sinfo** to report system status
4. **squeue** to report the status of jobs
5. **sacct** to get information about jobs and job steps that are running or have completed
In addition to the userland tools every user has access to some administrative information via the SLURM CLI admin tools

1. **scontrol** is the tool available to monitor and/or modify configuration and state information on the cluster such as partitions (queues) and other settings as well as detailed job information and accounting

2. **sacctmgr** is tool used to manage quality of service settings as well as the bank accounting and user database
Partitions (aka Queues) and Accounts

On *Hummingbird* the SLURM command line tools are loaded by default upon login

- Accounts are access groups allowing use of resources on the cluster
- Partitions are a logical unit which break up the cluster into different usable units based on the qualities or traits of different nodes or users/groups

When examining the system:

- To look at the partition definitions use **scontrol**
- To look at the available partitions use **sinfo**
- To look at the current and scheduled jobs use **squeue**
- To stop or cancel a current job use **scancel**
- To modify a current job use **scontrol**
sinfo - A quick way to check the cluster

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction*</td>
<td>up</td>
<td>4:00:00</td>
<td>3</td>
<td>idle</td>
<td>hbcomp-[000,004-005]</td>
</tr>
<tr>
<td>Course</td>
<td>up</td>
<td>4:00:00</td>
<td>2</td>
<td>idle</td>
<td>hbcomp-[001-002]</td>
</tr>
<tr>
<td>256x44</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>idle</td>
<td>hbcomp-003</td>
</tr>
<tr>
<td>128x24</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>down*</td>
<td>hbcomp-008</td>
</tr>
<tr>
<td>128x24</td>
<td>up</td>
<td>infinite</td>
<td>9</td>
<td>mix</td>
<td>hbcomp-[010-014,018,020,024-025]</td>
</tr>
<tr>
<td>128x24</td>
<td>up</td>
<td>infinite</td>
<td>6</td>
<td>alloc</td>
<td>hbcomp-[006-007,009,015-017]</td>
</tr>
<tr>
<td>128x24</td>
<td>up</td>
<td>infinite</td>
<td>3</td>
<td>idle</td>
<td>hbcomp-[019,023,026]</td>
</tr>
<tr>
<td>96x24gpu4</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>idle</td>
<td>hbcgpu-021</td>
</tr>
<tr>
<td>1024x28</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>idle</td>
<td>hbcomp-028</td>
</tr>
</tbody>
</table>

Some queues are restricted and are not for general use
sinfo - Use command line options to customize

<table>
<thead>
<tr>
<th>NODELIST</th>
<th>NODES</th>
<th>PARTITION</th>
<th>STATE</th>
<th>CPUS</th>
<th>S:C:T</th>
<th>MEMORY</th>
<th>TMP_DISK</th>
<th>WEIGHT</th>
<th>AVAIL_FE</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbcgpu-021</td>
<td>1</td>
<td>96x24gpu4</td>
<td>idle</td>
<td>48</td>
<td>2:12:2</td>
<td>92000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-000</td>
<td>1</td>
<td>Instruction*</td>
<td>idle</td>
<td>64</td>
<td>8:8:1</td>
<td>256000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-001</td>
<td>1</td>
<td>Course</td>
<td>idle</td>
<td>64</td>
<td>8:8:1</td>
<td>256000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-002</td>
<td>1</td>
<td>Course</td>
<td>idle</td>
<td>48</td>
<td>8:6:1</td>
<td>192000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-003</td>
<td>1</td>
<td>256x44</td>
<td>down*</td>
<td>44</td>
<td>2:22:1</td>
<td>256000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>Not responding</td>
</tr>
<tr>
<td>hbcgpu-004</td>
<td>1</td>
<td>Instruction*</td>
<td>idle</td>
<td>48</td>
<td>8:6:1</td>
<td>192000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-005</td>
<td>1</td>
<td>Instruction*</td>
<td>idle</td>
<td>64</td>
<td>8:0:1</td>
<td>192000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-006</td>
<td>1</td>
<td>128x24</td>
<td>allocated</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-007</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-008</td>
<td>1</td>
<td>128x24</td>
<td>down*</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>Not responding</td>
</tr>
<tr>
<td>hbcgpu-009</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-010</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-011</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-012</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-013</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-014</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-015</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-016</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-017</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-018</td>
<td>1</td>
<td>128x24</td>
<td>idle</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-019</td>
<td>1</td>
<td>128x24</td>
<td>mixed</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-020</td>
<td>1</td>
<td>128x24</td>
<td>mixed</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-021</td>
<td>1</td>
<td>128x24</td>
<td>allocated</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-022</td>
<td>1</td>
<td>128x24</td>
<td>mixed</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-023</td>
<td>1</td>
<td>128x24</td>
<td>allocated</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-024</td>
<td>1</td>
<td>128x24</td>
<td>mixed</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-025</td>
<td>1</td>
<td>128x24</td>
<td>mixed</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>hbcgpu-026</td>
<td>1</td>
<td>128x24</td>
<td>allocated</td>
<td>24</td>
<td>2:12:1</td>
<td>128000</td>
<td>0</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
</tbody>
</table>
```bash
[rootparson@hb ~]$ squeue -l
Wed Dec  2 13:15:07 2020

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>STATE</th>
<th>TIME</th>
<th>TIME_LIMI</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>88174</td>
<td>128x24</td>
<td>6_first_</td>
<td>blufox</td>
<td>RUNNING</td>
<td>16-20:12:26</td>
<td>62:00:00:00</td>
<td>1</td>
<td>hbcomp-024</td>
</tr>
<tr>
<td>88297</td>
<td>128x24</td>
<td>mapping_</td>
<td>jharenc</td>
<td>RUNNING</td>
<td>12-02:44:22</td>
<td>UNLIMITED</td>
<td>3</td>
<td>hbcomp-[012-014]</td>
</tr>
<tr>
<td>88370</td>
<td>128x24</td>
<td>7_first_</td>
<td>blufox</td>
<td>RUNNING</td>
<td>7-21:54:56</td>
<td>62:00:00:00</td>
<td>1</td>
<td>hbcomp-020</td>
</tr>
<tr>
<td>88493</td>
<td>128x24</td>
<td>8_first_</td>
<td>blufox</td>
<td>RUNNING</td>
<td>1-00:48:10</td>
<td>62:00:00:00</td>
<td>1</td>
<td>hbcomp-025</td>
</tr>
<tr>
<td>88498</td>
<td>128x24</td>
<td>PMZ7bi2</td>
<td>vroger</td>
<td>RUNNING</td>
<td>21:38:24</td>
<td>20-20:00:00</td>
<td>1</td>
<td>hbcomp-006</td>
</tr>
<tr>
<td>88499</td>
<td>128x24</td>
<td>P7unfre</td>
<td>vroger</td>
<td>RUNNING</td>
<td>21:35:09</td>
<td>20-20:00:00</td>
<td>1</td>
<td>hbcomp-007</td>
</tr>
<tr>
<td>88501</td>
<td>128x24</td>
<td>P7cunfre</td>
<td>vroger</td>
<td>RUNNING</td>
<td>21:28:56</td>
<td>20-20:00:00</td>
<td>1</td>
<td>hbcomp-009</td>
</tr>
<tr>
<td>88507</td>
<td>128x24</td>
<td>75898_3p</td>
<td>amstahl</td>
<td>RUNNING</td>
<td>20:07:49</td>
<td>INVALID</td>
<td>3</td>
<td>hbcomp-[010-011,018]</td>
</tr>
<tr>
<td>88511</td>
<td>128x24</td>
<td>extsemim</td>
<td>oross</td>
<td>RUNNING</td>
<td>5:01:56</td>
<td>20-10:00:00</td>
<td>3</td>
<td>hbcomp-[015-017]</td>
</tr>
</tbody>
</table>
```

Much easier to parse!
scontrol - Understanding Partition Limits/Size

Important fields to note:
- MaxNodes
- TotalNodes
- TotalCPUs
- Max wall clock time
- AllowGroups
- State
Before submitting ask yourself...

- How many jobs will you need to run?
- How large will those jobs be?
  - How many cores will each job require (-n)
  - How much memory will each job require total or per cpu
  - How many nodes to accommodate these memory/core/size requirements? (-N)
- How long will each job take? (Wall time)
- What kind of execution is required -- serial or parallel?
- What is a reasonable max time to allocate to contain runaway or failure potentials?

Accordingly, what job type best fits my workflow: single-run, interactive, batch or array

- Is the software I need already available as a module?
- Are there sufficient compute resources currently available?
- Do I need access to any additional Generic RESouces (GRES) such as a GPU?

Finally, choose your partition based on these requirements and current availability
Interlude - Laying out a scientific work space

HB is a shared research system and as such has some limitations and some baseline expectations for how users will behave and manage both their custom software and research data on the cluster.

HB IS NOT INTENDED AS A LONG TERM or REDUNDANT STORAGE SYSTEM FOR DATA OR CODE -- it is YOUR responsibility to backup and maintain additional copies of important bits.

There are 4 primary types of data we deal with in High Performance Computing (HPC):

1. Initial data: Often sync’d from external sources (no preservation, easily replicated)
2. Intermediate data: intermediary data generated by your work (no preservation, able to be replicated)
3. Custom code/documentation: preserved/versioned in a Source Control system like git or svn
4. Results: Final results of one’s own work (long term preservation, MULTIPLE offsite backups)

Each of these types of data require different data protection, backup and long-term storage strategies and it is important to layout your workspace in a way that administrators, collaborators and “future yous” can quickly and succinctly differentiate between the types.
Interlude - Example directory structure for HPC data
Types of Jobs - Single-Run

Single use or interactive jobs use \texttt{srun}

This can be helpful when trying to debug a batch script or problematic job step

To request an “interactive” job, if needed for debugging, use the \texttt{--pty} argument like so:

\begin{verbatim}
% srun -N 1 -n 1 -p 128x24 --pty bash
\end{verbatim}

Then you will be able to ssh directly to the node and access your allocation

However, \textbf{DO NOT run your software from here manually} or your job will be killed since the management of the resources are not under SLURMs control
Types of Jobs - Batch

Batch use or array jobs use `sbatch`

`Sbatch` can take its arguments either from command line flags or from special comments in the batch submit script or mix & match (command line args take priority)

You should use a batch job anytime you have parallel work that needs doing or have an actual HPC workload that is heavy duty
Submitting your job

```bash
sbatch your_job_script.slurm
```

Verifying your job's settings

```bash
scontrol show job <jobid>
```

That's a lot of details!
Types of Jobs - Array

- Array jobs are a special type of batch job allowing for simultaneous execution.
- A job array is a collection of jobs that differ from each other by only a single index parameter.
- Job arrays provide an easy way to group related jobs together.
- To specify that only a certain number of sub-jobs in the array can run at a time, use the percent sign (%) delimiter.
- To submit a specific set of array sub-jobs, use the comma delimiter in the array index list.
- You can specify a step size using a colon (:) in the array range for how many jobs to step.

```
# Submit a job array with index values between 0 and 31
$ sbatch --array=0-31 -N1 tmp

# Submit a job array with index values of 1, 3, 5 and 7
$ sbatch --array=1,3,5,7 -N1 tmp

# Submit a job array with index values between 1 and 7
# with a step size of 2 (i.e. 1, 3, 5 and 7)
$ sbatch --array=1-7:2 -N1 tmp
```
Array job example

We have a file with a list of paths to different files that you wish perform the same action on.

You could submit a job that loops through the file on 1 node and does said action, or you could submit an **array job with an index range** with however many lines the file contains.

For this example, our file contains 1000 lines:

```bash
#!/bin/bash
#
#SBATCH --ntasks=1
#SBATCH --partition sixhour
#SBATCH --time=6:00:00
#SBATCH --array=1-1000

LINE=$(sed -n "${SLURM_ARRAY_TASK_ID}p File.txt")
echo $LINE
call-program-name-here $LINE
```
What are Job Steps?

- All SLURM jobs can be broken down into steps.
- Job steps are sets of (possibly parallel) tasks within a job

To limit the number of simultaneous tasks in an array use the % delimiter

- For example "--array=0-15%4" will limit the number of simultaneously running tasks from the example job array to 4

```
$ sbatch --array [1-1000]%5 testarray.sh
```

```
516540_1
516540_2
516540_3
516540_4
516540_5
```
HB Template SLURM scripts

Premade templates for your convenience. Just make a copy to your working directory and edit to meet your needs.

If you have a template that is useful to more than just yourself, let us know and we can include it here.
Anatomy of a basic `sbatch` Script (Preamble)

- Name of Partition to use
- Name to give your run
- Name for output/error logs (use for troubleshooting)
- Number of nodes to request
- Number of tasks your program will require
- Amount of RAM requested for your program
- Time limit for the run (!!!)
- Get status emails about your job

```bash
#!/bin/bash
#SBATCH -p 128x24
#SBATCH -j example_job
#SBATCH -o job.%j.out
#SBATCH -e job.%j.err
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --mem=600mb
#SBATCH --time=00:05:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=rkparson@ucsc.edu
module load python-3.6.5
export EXMP_VAR="foo"
python my_python_script.py
```
Anatomy of a basic `sbatch` Script

- Any modules your program requires in order to run
- Any system variables your program might require
- The program to run

```bash
#!/bin/bash
#SBATCH -p 128x24
#SBATCH -j example_job
#SBATCH -o job.%j.out
#SBATCH -e job.%j.err
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --mem=600mb
#SBATCH --time=00:05:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=rkparson@ucsc.edu
module load python-3.6.5
export EXMP_VAR="foo"
python my_python_script.py
```
Some helpful detailed sinfo/squeue aliases

BASH version

# User specific aliases and functions
alias si="sinfo -o "%20P %5D %14F %8z %10m %10d %11l %32f %N\""
alias si2="sinfo -o "%20P %5D %6t %8z %10m %10d %11l %32f %N\""
alias sq="squeue -o "%8i %12j %4t %10u %20q %20a %10g %20P %10Q %5D %11l %11L %R\""
alias sq2="squeue --long -o "%.8i %.12P %.12a %.12q %.8j %.8u %.8T %.10M %.12l %.6D %.10Q %.20R\""

TCSSH version

alias si 'sinfo -o "%20P %5D %14F %8z %10m %11l %32f %N"'
alias si2 'sinfo -o "%20P %5D %6t %8z %10m %10d %11l %32f %N"
alias sq 'squeue -o "%8i %12j %4t %10u %20q %20a %10g %20P %10Q %5D %11l %11L %R"
alias sq2 'squeue --long -o "%.8i %.12P %.12a %.12q %.8j %.8u %.8T %.10M %.12l %.6D %.10Q %.20R""
SLURM with MPI, threaded, or OpenMP (shared memory) Jobs

SLURM supports multiple frameworks for parallel execution including the industry standard OpenMPI (message passing interface) and OpenMP for threaded or shared memory workloads. There are a number of options available to optimize these types of workloads (and the details get complex quick):

OpenMPI jobs can use multiple processors that may, or may not, be on multiple compute nodes:
- The SLURM --ntasks flag specifies the number of MPI tasks created for your job, which can end up on different nodes
- For more control over how SLURM lays out your job, you can add the --nodes and --ntasks-per-node flags
  - --nodes specifies how many nodes to allocate to your job and SLURM will allocate your requested number of cores to a minimal number of nodes on the cluster

For OpenMP Shared memory applications they can only run on a single node:
- You must set --ntasks=1, and then set --cpus-per-task to the number of threads you wish to use
- You must make the application aware of how many processors to use which depends on the application:
  - For some applications, set OMP_NUM_THREADS to a value less than or equal to the # of --cpus-per-task you set
  - For some applications, use a command line option when calling that application
Knowing your Hummingbird variants

hb.ucsc.edu
- Cluster login node
- Used for:
  - Compiling your code
  - Submitting jobs
  - Checking the status on your submitted jobs

hbfeeder.ucsc.edu
- Cluster storage node and data transfer node (DTN)
- Used for:
  - Accessing your data when you don't need the cluster
  - Transferring your data to or from the cluster
Thank you!!! Questions, Comments???
DO Set time limits on your jobs

In the event of a run-away job, this keeps it contained and keeps us admins happy
DON'T Overprovision

Request only the resources you need

This is especially true for the GPU node! There are four GPUs, but if you're only using one, you need to declare it so that the other three are available to others!
DON'T Leave data in Scratch

You are encouraged to use the scratch space when compiling your code or post-processing your results. Note however, that the scratch storage is NOT GUARANTEED. We regularly purge the contents of scratch based on age - if your stuff is languishing in there being unused, it will be deleted.