Intro to Slurm Workload Manager at MSI

Minnesota Supercomputing Institute
University of Minnesota

2020-10-20
Preamble: Objectives

- This tutorial is broken up into two sections. You will learn the following:
  a. Section 1: “Crash Course” (1 hour)
     - Timeline of PBS to Slurm transition at MSI
     - Important terminology for Slurm
     - Important commands for using Slurm
     - MSI’s Partitions
     - Running an interactive Slurm job
     - Converting a PBS jobs script to Slurm
  b. Section 2: Advanced Topics (1 hour)
     - Anatomy of a Slurm job
     - Writing new Slurm jobs scripts
     - Viewing accounting info
     - Job arrays
     - Job dependencies
Preamble: Formatting

There will be formatting cues to help you identify important pieces of information in this tutorial:

- **Monospaced text** indicates computer code or literal values that must be entered into a program.

- **Bold text** indicates technical terminology that is being used in a specific context. This is because the technical definitions collide with common language.

- **Italicized text** indicates a special word that you may hear in a computing context, but we are not covering it directly.
Preamble: Reference This Later!

- This tutorial has reference tables integrated into the slides. Please do not try to memorize them during the presentation; refer to the slides or to the website afterward!

- This tutorial has an interactive component that requires command line access to MSI.
  - Be sure you are connected to the UMN OIT VPN
    https://it.umn.edu/services-technologies/virtual-private-network-vpn
  - Be sure you have a way to use ssh to access MSI:
    https://www.msi.umn.edu/content/connecting-hpc-resources#ssh
Overview

- MSI is switching from PBS to Slurm for job management
- **Starting in 2021, all users must be using Slurm.**
  - PBS is going away.
  - The way you submit jobs will change.
- We are providing this workshop and document to help users make the transition from PBS to Slurm so that research work is minimally disrupted
- PBS will continue to function on MSI systems until January 2021.
  - MSI will be moving nodes from PBS management to Slurm management, so if you continue to use PBS, you will experience longer wait times and slower performance...
- For help, email MSI Help: help@msi.umn.edu
Transition Timeline

- October 2020:
  - Partitions (queues) established
- November 1, 2020:
  - >30% of nodes switch from PBS to Slurm
- December 1, 2020:
  - ~80% of nodes switch from PBS to Slurm
- January 6, 2021:
  - PBS goes offline

- See more info here:
  https://www.msi.umn.edu/slurm
Part 1: Resource Managers and Job Schedulers

- Systems to allocate *shared compute resources* to users of a large compute system
- Shared compute resources are often under *contention*
  - There is more compute work to be done than compute resources available at any given moment
- Workload is managed by a *resource manager* and a *job scheduler*
- Resource manager:
  - Monitors node availability and load (usage)
  - Manages CPU, network, disk, memory, etc. in a cluster
- Job scheduler:
  - Sends compute tasks to nodes
  - Manages queues and priority

Somewhat like the *Maître d'hôtel* in a restaurant
Part 1: Resource Managers and Job Schedulers

- Some factors for determining which jobs get run:
  - Current system load
  - Submitting user’s *fair-share usage*
  - Submitted job’s requested resources
- Several solutions to this problem:
  - **Portable Batch System**
  - **Slurm Workload Manager**
  - Sun Grid Engine
  - IBM Load Sharing Facility
  - And others
Part 2: Slurm Overview

- Slurm is both a **resource manager** and a **job scheduler**
- Officially supported by SchedMD:
  - [https://schedmd.com/](https://schedmd.com/)
- Online documentation:
  - [https://slurm.schedmd.com/](https://slurm.schedmd.com/)
- Open source:
  - [https://github.com/SchedMD/](https://github.com/SchedMD/)
- MSI is running **Slurm 20.02.3**
  - If you are looking at the official documentation, be sure that the versions match
  - Also, if you find documentation from a different computing facility, be sure you know what version they are running
  - Slurm is also highly customisable, so we cannot guarantee that what is posted on another facility’s documentation will work at MSI
Part 2: Terminology

- Thankfully most of the terminology for Slurm is very similar to the terminology used by PBS TORQUE/Moab:
  - **Job**: a reservation on the system to run commands
  - **Node**: physical machine that is part of the cluster. The cluster is made up of many connected nodes.
  - **Core/CPU**: single processing unit for computing. One node contains many cores or CPUs (we will discuss this later!)
- There are a couple places where the terminology is different, however:
  - **Partition**: where to run jobs. TORQUE PBS calls this a “queue.” Has resource limits and access controls.
  - **Quality of Service (QoS)**: special limits for a given partition or user. TORQUE PBS implements this with “routing queues” (large, max, widest, on Mesabi, for example).
Part 2: Important Differences

- Besides terminology, there are some functional differences between PBS and Slurm that you should be aware of:
  
  - Slurm combines the `stdout` and `stderr` channels into one file by default (like `-j oe` in PBS). PBS’s default behavior is to write them separately as `.o` and `.e` files, respectively.
    - We will go over how to deal with this!

  - Slurm jobs run in the same directory as the submitted jobscript. PBS jobs, by comparison, run in the submitter’s home directory.

  - Slurm allows you to specify multiple `partitions` for a job. PBS allows you to specify only one queue. More on this later!
Part 3: Interacting with Slurm

- Slurm uses different commands from TORQUE PBS/Moab to handle jobs and view information about a specific job or partition
- The basics are shown in the next tables, but refer to the following guides for more detailed descriptions:

NIH PBS to Slurm guide:
https://hpc.nih.gov/docs/pbs2slurm.html

NREL guide:
## Part 3: Important Commands

- Do NOT memorize this table right now! Use this as a reference for when you need to interact with Slurm.

<table>
<thead>
<tr>
<th>Slurm Command</th>
<th>PBS/Moab Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbatch</td>
<td>qsub</td>
<td>Submit a job to the scheduler</td>
</tr>
<tr>
<td>srun --pty bash</td>
<td>qsub -I</td>
<td>Submit an interactive job to the scheduler</td>
</tr>
<tr>
<td>scancel</td>
<td>qdel</td>
<td>Delete a job</td>
</tr>
<tr>
<td>scancel</td>
<td>mjobctl -c</td>
<td>Delete a job</td>
</tr>
<tr>
<td>scontrol show job</td>
<td>checkjob</td>
<td>Show job information</td>
</tr>
</tbody>
</table>

Also note: you will have to provide options and arguments to these commands. They are not shown in this table.
Part 3: Important Commands

- Do NOT memorize this table right now! Use this as a reference for when you need to interact with Slurm.

<table>
<thead>
<tr>
<th>Slurm Command</th>
<th>PBS/Moab Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scontrol show partition</code></td>
<td><code>qstat -Qf</code></td>
<td>View partition configuration information</td>
</tr>
<tr>
<td><code>squeue -al</code></td>
<td><code>qstat -f</code></td>
<td>Show all job information</td>
</tr>
<tr>
<td><code>squeue --me</code></td>
<td><code>qstat -u $(id -un)</code></td>
<td>Show only your job information</td>
</tr>
<tr>
<td><code>sinfo</code></td>
<td><code>qstat -Q</code></td>
<td>Show partition status</td>
</tr>
</tbody>
</table>

Also note: you will have to provide options and arguments to these commands. They are not shown in this table.
## Part 4: MSI Partitions: Mesabi

<table>
<thead>
<tr>
<th>Partition Name</th>
<th>Node Sharing?</th>
<th>Max. nodes per job</th>
<th>Cores per node</th>
<th>Walltime limit</th>
<th>Total node memory</th>
<th>Advised memory per core</th>
<th>Local scratch per node</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>Yes</td>
<td>9</td>
<td>24</td>
<td>96:00:00</td>
<td>60.4gb</td>
<td>2639mb</td>
<td>390gb</td>
</tr>
<tr>
<td>large</td>
<td>No</td>
<td>48</td>
<td>24</td>
<td>24:00:00</td>
<td>60.4gb</td>
<td>2639mb</td>
<td>390gb</td>
</tr>
<tr>
<td>widest</td>
<td>No</td>
<td>360</td>
<td>128</td>
<td>24:00:00</td>
<td>60.4gb</td>
<td>2639mb</td>
<td>390gb</td>
</tr>
<tr>
<td>max</td>
<td>Yes</td>
<td>1</td>
<td>24</td>
<td>696:00:00</td>
<td>60.4gb</td>
<td>2639mb</td>
<td>390gb</td>
</tr>
<tr>
<td>ram256g</td>
<td>Yes</td>
<td>2</td>
<td>24</td>
<td>96:00:00</td>
<td>248.9gb</td>
<td>10814.3mb</td>
<td>390gb</td>
</tr>
<tr>
<td>ram1t</td>
<td>Yes</td>
<td>2</td>
<td>24</td>
<td>96:00:00</td>
<td>10003.9gb</td>
<td>32649.3mb</td>
<td>228gb</td>
</tr>
<tr>
<td>k40</td>
<td>No</td>
<td>40</td>
<td>24</td>
<td>24:00:00</td>
<td>123.2gb</td>
<td>5365.5mb</td>
<td>390gb</td>
</tr>
<tr>
<td>interactive</td>
<td>Yes</td>
<td>4*</td>
<td>24**</td>
<td>12:00:00</td>
<td>60.4gb*</td>
<td>2639mb*</td>
<td>***</td>
</tr>
</tbody>
</table>

Note: jobs in the **interactive** partition have a limit of four (4) cores total, spread across 1, 2, or 4 nodes. It also targets **ram256g** and **ram1t** nodes, so please refer to per-core memory recommendations for high-memory interactive jobs.

Yellow highlight: **nodes** with GPUs
# Part 4: MSI Partitions: Mangi

<table>
<thead>
<tr>
<th>Partition Name</th>
<th>Node Sharing?</th>
<th>Max. nodes per job</th>
<th>Cores per node</th>
<th>Walltime limit</th>
<th>Total node memory</th>
<th>Advised memory per core</th>
<th>Local scratch per node</th>
</tr>
</thead>
<tbody>
<tr>
<td>amdsmall</td>
<td>Yes</td>
<td>1</td>
<td>128</td>
<td>96:00:00</td>
<td>248.7gb</td>
<td>2027.7mb</td>
<td>429gb</td>
</tr>
<tr>
<td>amdlarge</td>
<td>No</td>
<td>32</td>
<td>128</td>
<td>24:00:00</td>
<td>248.7gb</td>
<td>2027.7mb</td>
<td>429gb</td>
</tr>
<tr>
<td>amd2tb</td>
<td>Yes</td>
<td>1</td>
<td>128</td>
<td>96:00:00</td>
<td>2010gb</td>
<td>16341.8mb</td>
<td>429gb</td>
</tr>
<tr>
<td><strong>v100</strong></td>
<td>No</td>
<td><strong>6</strong></td>
<td><strong>24</strong></td>
<td>24:00:00</td>
<td>376.4gb</td>
<td>16352.7mb</td>
<td><strong>875gb</strong></td>
</tr>
</tbody>
</table>

Note: All Mangi GPU nodes have been placed into the **v100** partition. Jobs in this **partition** will be allocated as follows:
- 1-2 GPUs: **v100** 2-way, 4-way, or 8-way nodes
- 3-4 GPUs: **v100** 4-way or 8-way nodes
- 5-8 GPUs: **v100** 8-way node

Yellow highlight: nodes with GPUs
Part 4: MSI Partitions

- There are a few changes from the TORQUE PBS queues:
  - There is no `amd_or_intel` partition
  - To submit jobs to either Mesabi or Mangi (which is what the PBS `amd_or_intel` queue targeted), use the following in your batch scripts:

    ```bash
    #SBATCH -p small,amdsmall
    ```

- There are no `v100-4` and `v100-8` partitions
  - These queues have all been merged into the `v100` partition.
  - Jobs in this partition will be placed as follows:
    - 1-2 GPUs: v100 2-way, 4-way, or 8-way nodes
    - 3-4 GPUs: v100 4-way or 8-way nodes
    - 5-8 GPUs: v100 8-way node
Next: Hands-on Work

- We will now start the hands-on portion of the tutorial.
- Connect to the login.msi.umn.edu server with your ssh program. Replace X.500 with your UMN internet ID. Be sure you are connected to the UMN OIT VPN!

  ssh X.500@login.msi.umn.edu

- Connect to the mesabi cluster from the login node.

  ssh mesabi
Part 5: Interactive Slurm Jobs

- Use the `srun` command to request an interactive job:

```
srun -N 1 -n 1 -c 1 --mem=2gb -t 20 -p interactive --pty bash
```

This job makes the following request:
- **1 node** (`-N 1`)
- **1 core** (`-n 1 -c 1`)
- **2gb of RAM** (`--mem=2gb`)
- **20 minutes of walltime** (`-t 20`)

Use the **interactive partition** (`-p interactive`)

- The **--pty bash** tells the system that you want to run a **bash** shell (interactively) inside of your allocation.
- When you see your prompt again, you are running a shell in a new interactive **job** allocation
Part 5: Interactive Slurm Jobs

- Let’s check what node we are connected to. Run this command:

  `hostname`

- You should see a name like `cn0007` get printed to the terminal. This is the name of the compute node onto which your allocation was assigned.
  - If you experience issues related to a particular node, be sure to include the name of the node in your messages to the MSI Helpdesk.
  - This will help us identify potential hardware errors or misconfigurations.
Part 5: Interactive Slurm Jobs

- Let’s check the job ID by running this command:

  ```bash
  echo ${SLURM_JOBID}
  ```

- You should see a number like 9620 get printed to the terminal. This is is the ID of the allocation for your job.
  - If you experience issues related to a job, be sure to include the ID of the job in your message to the MSI Helpdesk.

- Exit out of the job:

  ```bash
  exit
  ```

- Interactive jobs in Slurm function identically to interactive jobs in PBS TORQUE/Moab
  - You have full access to the software available in modules
  - You can run interactive R, Perl, Python…
Part 6: Converting a PBS Script to Slurm

- Now, we will convert a pre-written PBS jobscript into a Slurm jobscript.
- Copy the example PBS script into your home directory:

  ```sh
cp /home/msistaff/public/Slurm_Workshop/pbs_example_to_convert.sh ~
  ```

- Open the script in `nano`:

  ```sh
  nano pbs_example_to_convert.sh
  ```
Part 6: Converting a PBS Script to Slurm

- The script looks like the text on the left; edit it to make it look like the text on the right (use your email address, though!):

```bash
#!/bin/bash
#PBS -l nodes=1:ppn=1,mem=2gb,walltime=00:20:00
#PBS -m abe
#PBS -M YOUR.X.500@umn.edu
#PBS -q mesabi
hostname
echo ${PBS_JOBID}
```

```bash
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=2gb
#SBATCH -t 20
#SBATCH --mail-type=ALL
#SBATCH --mail-user=YOUR.X.500@umn.edu
#SBATCH -p small
#SBATCH -o %j.out
#SBATCH -e %j.err
hostname
echo ${SLURM_JOBID}
```
Part 6: Converting a PBS Script to Slurm

- Save the file by pressing [Control] + [X], then pressing [Y], then pressing [Enter]
- Now, submit the job with sbatch:

```bash
sbatch pbs_example_to_convert.sh
```

- Make a note of the job ID that gets written to the terminal.
- Watch out for the emails!
  - They come from msi_slurm@msi.umn.edu; so filter based on that address.
- Check the output from the job; replace job_id with your actual job ID:

```bash
more job_id.out
```
## Part 6: Converting a PBS Script to Slurm

Conversion between commonly-used PBS and Slurm *directives*:

<table>
<thead>
<tr>
<th>PBS Directive</th>
<th>Slurm Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -l nodes=X:ppn=Y</td>
<td>#SBATCH --nodes=X</td>
<td>Request <em>X</em> nodes and <em>Y</em> CPUs per node</td>
</tr>
<tr>
<td></td>
<td>#SBATCH --ntasks-per-node=Y</td>
<td></td>
</tr>
<tr>
<td>#PBS -l walltime=HH:MM:SS</td>
<td>#SBATCH -t HH:MM:SS</td>
<td>Request a total of <em>HH:MM:SS</em> of walltime</td>
</tr>
<tr>
<td>#PBS -l mem=Xgb</td>
<td>#SBATCH --mem=Xgb</td>
<td>Request a total of <em>X</em> gigabytes of memory for the job</td>
</tr>
<tr>
<td>#PBS -q QUEUE</td>
<td>#SBATCH -p QUEUE</td>
<td>Send job to the QUEUE queue or partition</td>
</tr>
</tbody>
</table>

Note: you will have to fill in appropriate values for these directives. The values that need to be replaced are **bold and underlined**
# Part 6: Converting a PBS Script to Slurm

Conversion between commonly-used PBS and Slurm directives:

<table>
<thead>
<tr>
<th>PBS Directive</th>
<th>Slurm Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -M <a href="mailto:USER@umn.edu">USER@umn.edu</a></td>
<td>#SBATCH --mail-user=<a href="mailto:USER@umn.edu">USER@umn.edu</a></td>
<td>Send job emails to <a href="mailto:USER@umn.edu">USER@umn.edu</a></td>
</tr>
<tr>
<td>#PBS -m abe</td>
<td>#SBATCH --mail-type=ALL</td>
<td>Send job emails for start, abort, and completion</td>
</tr>
<tr>
<td>#PBS -e file.err</td>
<td>#SBATCH -e file.err</td>
<td>Write the standard error channel to file.err</td>
</tr>
<tr>
<td>#PBS -o file.out</td>
<td>#SBATCH -o file.out</td>
<td>Write the standard output channel to file.out</td>
</tr>
<tr>
<td>#PBS -N NAME</td>
<td>#SBATCH -J NAME</td>
<td>Set the job name to NAME</td>
</tr>
</tbody>
</table>

Note: you will have to fill in appropriate values for these directives. The values that need to be replaced are **bold and underlined**
Reminder: Transition Timeline

- October 2020:
  - Partitions (queues) established
- November 1, 2020:
  - >30% of nodes switch from PBS to Slurm
- December 1, 2020:
  - ~80% of nodes switch from PBS to Slurm
- January 6, 2021:
  - PBS goes offline
Section 2: More Advanced Slurm

- By now, you should:
  - Know that you will have to use Slurm
  - Have a translation table between PBS commands and Slurm commands
  - Know how to submit interactive and batch jobs to the Slurm scheduler
  - Know how to convert a PBS jobscript to a Slurm jobscript

- Short break (5 min)!

- Next section:
  - More detailed Slurm job terminology
  - Writing new Slurm jobscripts
  - View accounting info
  - Intro to job arrays
  - Intro to job dependencies
Gritty Details: Terminology

- **Job**: Resource request that can be used to perform compute tasks. CPU (and optionally GPU), memory, disk space for a specified time.

- **Step**: A specific command or compute task within a **job**. A job is made up of one or more **steps**.

- **Task**: A compute process that needs to be run. One or more **tasks** make up a **step**.

- **Partition**: Queue for **jobs**. Has resource limits and access controls.
Part 7: General Slurm Jobs

In a schematic:

- A **job** is just a resource allocation request.
  - Made up of one or more **steps**.
  - A step can contain one or more **tasks**.
- Mechanistically, the **steps** in a **job** are subsets of the overall allocation for the **job**.
  - These can be run **sequentially** or **in parallel**

Job 1:
2 nodes, 4 CPUs per node, 2gb RAM per CPU, 4 hours

- Step 1: 1 CPU
- Step 2: 8 CPU
- Step 3: 4 CPU
- Step 4: 4 CPU

**Present**

**Time**
Part 7: General Slurm Jobs

In a script:

```bash
#!/bin/bash

#SBATCH --nodes 2
#SBATCH --tasks-per-node 4

# Step 1
srunk --nodes 1 --ntasks 1 mkdir -p /scratch.global/user

# Step 2
srunk analysis.mpi < input.dat > /scratch.global/user/output_1.txt

# Step 3
srunk --ntasks 4 --nodes 1 analysis_2.mpi < input.dat >
  /scratch.global/user/output_2.txt &

# Step 4
srunk --ntasks 4 --nodes 1 analysis_3.mpi < input.dat >
  /scratch.global/user/output_3.txt &

# Wait for the two backgrounded processes to finish
wait
```

Resource request parameters for the whole job

Steps are made with `srunk`. We will cover this later!

Steps can have their own allocations within a job

Steps can be run in parallel
Part 7: General Slurm Jobs

- Note that it is not necessary to use `srun` to make **steps** within the **job**.
  - You can just use a normal shell script.
  - The division of a **job** into **steps** makes it easier to manage concurrent processes in a **job** and also view more detailed resource usage information for your **job**.
  - You can more tightly control how many compute resources any given **step** is allowed to use.
Part 8: A New Batch Job

- Now we will write a new jobscript for a batch job
- Use `nano` to start a new script:

```
nano example_batch.sh
```

- We are starting a new jobscript here because we will use some of the features of Slurm job management
Part 8: A New Batch Job

- Enter the following text into the file. Be sure to use your actual email address instead of the placeholder!

```bash
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=2gb
#SBATCH -t 20
#SBATCH --mail-type=ALL
#SBATCH --mail-user=YOUR.X.500@umn.edu
#SBATCH -p small
#SBATCH -o %j.out
#SBATCH -e %j.err

srun hostname

srun echo ${SLURM_JOBID}
```
Part 8: A New Batch Job

- Save the file by pressing `[Control] + [X]`, then pressing `[Y]`, then pressing `[Enter]`.
- Now, we will send the job to the scheduler with the `sbatch` command:

  `sbatch example_batch.sh`

- You will see text like “Submitted batch job 9621” get written to the terminal.
- Eventually, you will get some emails from the Slurm scheduler about the start and finish of your jobs.
  - Just like with PBS TORQUE/Moab, set up an email filter to manage these!
  - They come from `msi_slurm@msi.umn.edu`
Part 8: A New Batch Job

- Check the contents of your home directory:

```bash
cd $HOME
ls -ltrh
```

- You should see two files that have names like `9621.out` and `9621.err` (your filenames will have your job ID, rather than my job ID).

- Dump the contents of the .out file to the terminal:

```bash
more 9621.out
```

- The information looks very similar to what we saw during the interactive work!
Part 9: View Accounting Info

- We will use the batch job we submitted in the previous section to view some basic accounting information about the job.
- This is also included in the Slurm email summaries that get sent upon job completion.
- Accounting information includes:
  - Job ID
  - Partition in which the job was run
  - Job name
  - Allocated resources
  - Execution time
  - Nodes that were used
  - And more …!
- Use accounting information to tune your resource request for the job you are running. Request only what you will realistically need; it helps your job run on the system sooner!
- Recall the ID of the batch job. Use it to check the accounting information:

```
sacct -j 10384
```

- What gets printed is something like the following:

<table>
<thead>
<tr>
<th>JobID</th>
<th>JobName</th>
<th>Partition</th>
<th>Account</th>
<th>AllocCPUS</th>
<th>State</th>
<th>ExitCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>10384</td>
<td>batch.sh</td>
<td>small</td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>10384.batch</td>
<td>batch</td>
<td></td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>10384.extern</td>
<td>extern</td>
<td></td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>10384.0</td>
<td>hostname</td>
<td></td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>10384.1</td>
<td>echo</td>
<td></td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
</tbody>
</table>

- There are a lot of pieces here, so we will break them down a bit in the next slide!
Part 9: View Accounting Info

<table>
<thead>
<tr>
<th>JobID</th>
<th>JobName</th>
<th>Partition</th>
<th>Account</th>
<th>AllocCPUS</th>
<th>State</th>
<th>ExitCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>10384</td>
<td>batch.sh</td>
<td>small</td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>10384.batch</td>
<td>batch</td>
<td></td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>10384.extern</td>
<td>extern</td>
<td></td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>10384.0</td>
<td>hostname</td>
<td></td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>10384.1</td>
<td>echo</td>
<td></td>
<td>msistaff</td>
<td>1</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
</tbody>
</table>

- Five entries for this one job:
  a. 10384: Accounting info for the whole job
  b. 10384.batch: Accounting info for the batch script portion of the job
  c. 10384.extern: Accounting info for non-batch script portion of the job, e.g., if you connected to the compute node and ran commands while the job was executing
  d. 10384.0: Accounting info for the first step of the job, hostname (the first srun statement)
  e. 10384.1: Accounting info for the second step of the job, echo (the second srun statement)

- We will see in two slides how using steps makes it easy to keep track of resource usage within a large job!
- You can view many more pieces of accounting information, such as the CPU time, memory used, and total execution time. See the list of fields for the `--format=` option to `sacct`:

  [https://slurm.schedmd.com/sacct.html](https://slurm.schedmd.com/sacct.html)

- You also get this information (and more!) in the email report when your job finishes.
Part 9: View Accounting Info

- Sample email report of job accounting:

<table>
<thead>
<tr>
<th>JobID</th>
<th>JobName</th>
<th>MaxRSS</th>
<th>MaxVMSIZE</th>
<th>Elapsed</th>
<th>AllocCPUS</th>
<th>NTasks</th>
<th>ExitCode</th>
<th>WorkDir</th>
<th>Group</th>
<th>Partition</th>
<th>MaxDiskWrite</th>
<th>MaxDiskRead</th>
<th>NodeList</th>
</tr>
</thead>
<tbody>
<tr>
<td>10384</td>
<td>batch.sh</td>
<td>1188K</td>
<td>182864K</td>
<td>00:00:08</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>/panfs/roc/scratch/konox006/slurm</td>
<td>msstaff small</td>
<td></td>
<td></td>
<td></td>
<td>cn0002</td>
</tr>
<tr>
<td>10384.batch</td>
<td>batch</td>
<td>864K</td>
<td>150804K</td>
<td>00:00:08</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>/panfs/roc/scratch/konox006/slurm</td>
<td>msstaff small</td>
<td></td>
<td></td>
<td></td>
<td>cn0002</td>
</tr>
<tr>
<td>10384.extern.extern</td>
<td>batch</td>
<td>1124K</td>
<td>181.50M</td>
<td>00:00:01</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>/panfs/roc/scratch/konox006/slurm</td>
<td>msstaff small</td>
<td></td>
<td></td>
<td></td>
<td>cn0002</td>
</tr>
<tr>
<td>10384.0.hostname</td>
<td>hostname</td>
<td>1120K</td>
<td>181.50M</td>
<td>00:00:01</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>/panfs/roc/scratch/konox006/slurm</td>
<td>msstaff small</td>
<td></td>
<td></td>
<td></td>
<td>cn0002</td>
</tr>
</tbody>
</table>

Memory usage, by step. Mind the units!
Execution time, by step.

Working directory. You will find the output and error files here.
Volume of data written to and read from disk.
Allocated nodes.

Breaking a job up into steps allows detailed resource tracking! You can tell which steps are the “heavy ones” and adjust them if necessary. You can also estimate the required resources for future job submissions.
Part 10: Job Arrays

- Slurm supports job arrays in a similar fashion as TORQUE PBS/Moab
  - These are useful if you have a workflow that must be run on a collection of input data files
  - For example, an RNA sequencing data workflow that must be run on a collection of single-sample files
- Use the \texttt{--array=} option to \texttt{sbatch} to enable array processing
  - Array indices are inclusive; for example, \texttt{--array=0-10} submits 11 jobs.
- To reference the array index in the job script, use the \texttt{$\{SLURM\_ARRAY\_TASK\_ID\}$} environmental variable
  - The PBS equivalent of this is \texttt{$\{PBS\_ARRAYID\}$}
  - Also note here that the “task” that Slurm is referring to in its variable name is not the same as a task in the resource request context
Part 10: Job Arrays

- We will run an example **job array** with a pre-written Slurm jobscript now.
- Copy the example script into your home directory:

  ```bash
  cp /home/msistaff/public/Slurm_Workshop/slurm_job_arrays_example.sh ~
  ```

- Edit the script in **nano** to replace the dummy email address with your own (line 9):

  ```bash
  nano slurm_job_arrays_example.sh
  ```

- Send the **job array** to the scheduler. There are four (4) input files, so use the `--array=0-3` option to send a **job array** with four **jobs**:

  ```bash
  sbatch --array=0-3 slurm_job_arrays_example.sh
  ```

- Watch out for the emails, then check the outputs!
Part 10: Job Arrays

- Let’s take a look at the input data:

```bash
ls -1 /home/msistaff/public/Slurm_Workshop/array_example_data
```

- The resulting file listing looks like this:

```
01.dat
02.dat
03.dat
04.dat
```

- Notice how the names have a common structure. This is important, and we will cover this in the next slide!
Part 10: Job Arrays

- **Array indices** are just integers. The script is reproduced below:

```bash
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=100mb
#SBATCH -t 5
#SBATCH -p small
#SBATCH --mail-type=ALL
#SBATCH --mail-user=konox006@umn.edu
#SBATCH -o %A_%a.out
#SBATCH -e %A_%a.err

DATA_DIR="/home/msistaff/public/Slurm_Workshop/array_example_data"
DATA_FILES=$(find ${DATA_DIR} -mindepth 1 -maxdepth 1 -type f | sort -V)
CURR_DATA_FILE=${DATA_FILES[${SLURM_ARRAY_TASK_ID}]}
srun echo "This is array index ${SLURM_ARRAY_TASK_ID}. I am processing ${CURR_DATA_FILE}."
```

Orange boxes: these are the array pieces!
Part 11: Dependencies

- Slurm supports **job dependencies**, too. Useful for pipelines:
  - Job 1: quality control of data.
  - IF job 1 succeeds:
    - Job 2, job 3, and job 4 will perform separate analyses
  - IF jobs 2, 3, and 4 succeed:
    - Job 5 will generate a report of the analyses
- If a **job** fails, then the **jobs** that come later in the pipeline (**depend** on it), will be held
  - You can use `scancel` to delete **jobs** that are held due to failed **dependencies**
Part 11: Dependencies

- Use the `--dependency=` option to `sbatch` to supply a dependency, in the form of a job ID.
- Also use the `--parsable` option to make the retrieval of the job ID easier!
- The `--parsable` option makes `sbatch` write only the job ID to the terminal (rather than the full sentence)
- What it would look like in a script:

```
job1=$(sbatch --parsable job1.sh)
job2=$(sbatch --parsable --dependency=afterok:${job1} job2.sh)
job3=$(sbatch --parsable --dependency=afterok:${job1} job3.sh)
job4=$(sbatch --parsable --dependency=afterok:${job1} job4.sh)
job5=$(sbatch --parsable --dependency=afterok:${job2}:${job3}:${job4} job5.sh)
```
Part 11: Dependencies

- There are many types of **dependencies** that are available
  - “afterok” is likely to be the one you will use most in an analytical pipeline
- See the “dependency” section in the `sbatch` manual to see the full list of **dependency** types that you can specify:
  [https://slurm.schedmd.com/sbatch.html](https://slurm.schedmd.com/sbatch.html)
  - Combine them with **arrays** for extra fun and sophisticated pipelines!
Further Reading: Slurm @ MSI

- Slurm official documentation:
  https://slurm.schedmd.com/documentation.html

- Slurm @ MSI overview:
  https://www.msi.umn.edu/slurm

- MSI guide on batch job submission and scheduling:
  https://www.msi.umn.edu/content/job-submission-and-scheduling-slurm

- MSI guide on interactive job submission:
  https://www.msi.umn.edu/content/interactive-queue-use-qsub

- MSI-RIS Slurm quickstart (Requires UMN ID):
  https://github.umn.edu/MSI-RIS/SLURM_Quickstart/blob/master/SLURM_Quic
  kstart.md
Further Reading: MSI Generally

- MSI queues: [https://www.msi.umn.edu/queues](https://www.msi.umn.edu/queues)

- MSI tutorials: [https://www.msi.umn.edu/tutorials](https://www.msi.umn.edu/tutorials)

- MSI interactive HPC resources: [https://www.msi.umn.edu/content/connecting-interactive-hpc-resources](https://www.msi.umn.edu/content/connecting-interactive-hpc-resources)

- MSI software catalogue: [https://www.msi.umn.edu/software](https://www.msi.umn.edu/software)
Further Reading: Nice Things

- NIH has a PBS to Slurm conversion tool: https://hpc.nih.gov/docs/pbs2slurm_tool.html

  If you use this, READ YOUR SCRIPT CAREFULLY! Make sure that the logic of the script is still intact before submitting jobs.
Reminder: Transition Timeline

- October 2020:
  - Partitions (queues) established
- November 1, 2020:
  - >30% of nodes switch from PBS to Slurm
- December 1, 2020:
  - ~80% of nodes switch from PBS to Slurm
- January 6, 2021:
  - PBS goes offline
Thank You!

- If you have feedback on this tutorial, please send it Tom Kono (konox006@umn.edu). I am happy to make the tutorials more useful for you!
- If you have additional questions about the Slurm transition or have difficulties with the Slurm scheduler, please contact the MSI Help Desk (help@msi.umn.edu)
Supplement: Resource Managers and Job Schedulers

- Example: three users want to run jobs on the cluster

- Analogous to three groups want to eat dinner at a restaurant
  - 3 people, 5 people, 16 people with a prior reservation
  - But, there is only one table with four seats available right now
  - Who should get seated?

- Some things to consider:
  - Are there tables that are about to be free?
  - Who was waiting the longest?
  - Not appropriate in a restaurant, but relevant for job scheduling:
    - Who is the hungriest?

*: This is not how the MSI job scheduler actually works; this example is to illustrate why scheduling is important when there is contention for compute/memory/throughput
Gritty Details: Hardware Terminology

- **Cluster**: Set of connected compute resources (hardware!). Made up of multiple **nodes**.
- **Node**: Set of compute resources that are physically connected, i.e., in the same “box” or “server” or “machine.” Multiple **nodes** are connected via network to make a **cluster**.
- **Core**: A single unit of computing hardware. Largely synonymous with “CPU.” A single **node** has multiple **cores**.
Supplementary Background

- MSI’s previous system used the TORQUE fork of the PBS resource manager and the Moab job scheduler
  
  - This is where some of the issues with jobs came from:
    - Jobs are sent to TORQUE with `qsub`
    - Jobs are then assigned an ID by TORQUE and sent to the Moab scheduling daemon
    - Moab monitors job status and communicates changes to TORQUE
  
  - If one of TORQUE or Moab were overloaded or down, then job control or job monitoring would fail.
    - This would lead to `qsub/qstat/qdel` hanging or not being able to report information on a job
    - May also be related to some “zombie” jobs that continually run and drain service units
Supplementary Background

- MSI’s new system uses Slurm for both resource management and job scheduling
  - “Slurmctld” manages available resources and schedules new jobs
    - Typically running multiple instances within a facility: one per “cluster” (Mesabi or Mangi)
  - “Slurmdb” manages accounting information for users/groups/jobs
    - Typically running a single slurmdb instance for all of a site
  - Should be more resilient to downtime or large volumes of requests than PBS TORQUE/Moab because it is more distributed