UNIVERSITY OF Nebraska®

HOLLAND COMPUTING CENTER

High Performance Computing Core

Important Links

- Workshop Website
  - http://hcc.unl.edu/hcc-kickstart-2020
- Command History
  - https://hcc.unl.edu/swc-history/20201021.html
- HCC Documentation
  - http://hcc.unl.edu/docs/
- Job Examples
  - https://github.com/unlhcc/job-examples

Learning Objectives

Session 1 – Wednesday, October 21st

- Introduction to high performance computing
- What is a supercomputing cluster?
- Who is HCC and what services do we provide
- Using the SLURM job scheduler
  - Interactive jobs
  - Submitting batch jobs

Session 2 – Thursday, October 22nd

- Data storage on Crane
- Transferring files to and from Crane
  - scp
  - Introduction to Globus
- Using Applications on Crane
  - module system

What is High Performance Computing?

High Performance Computing most generally refers to the practice of

aggregating computing power

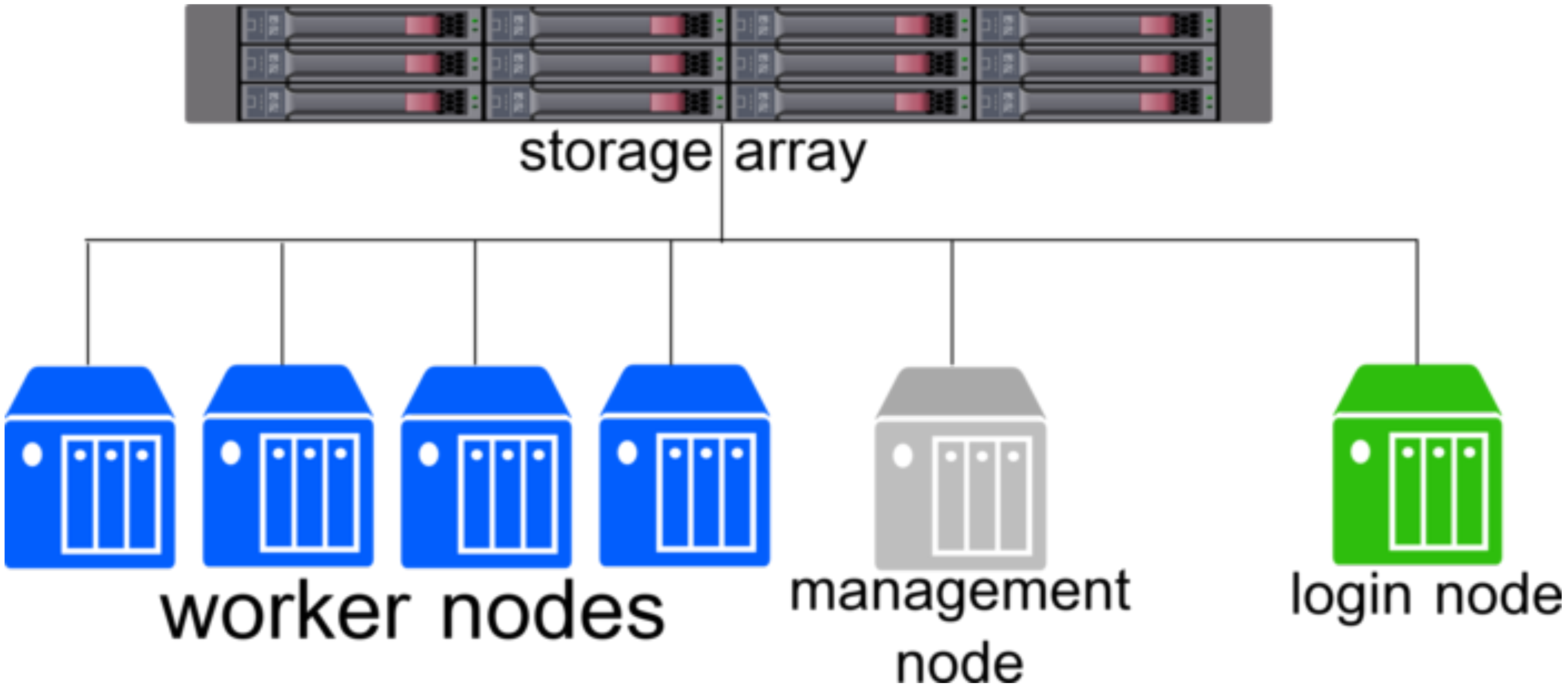
in a way that delivers much **higher performance**

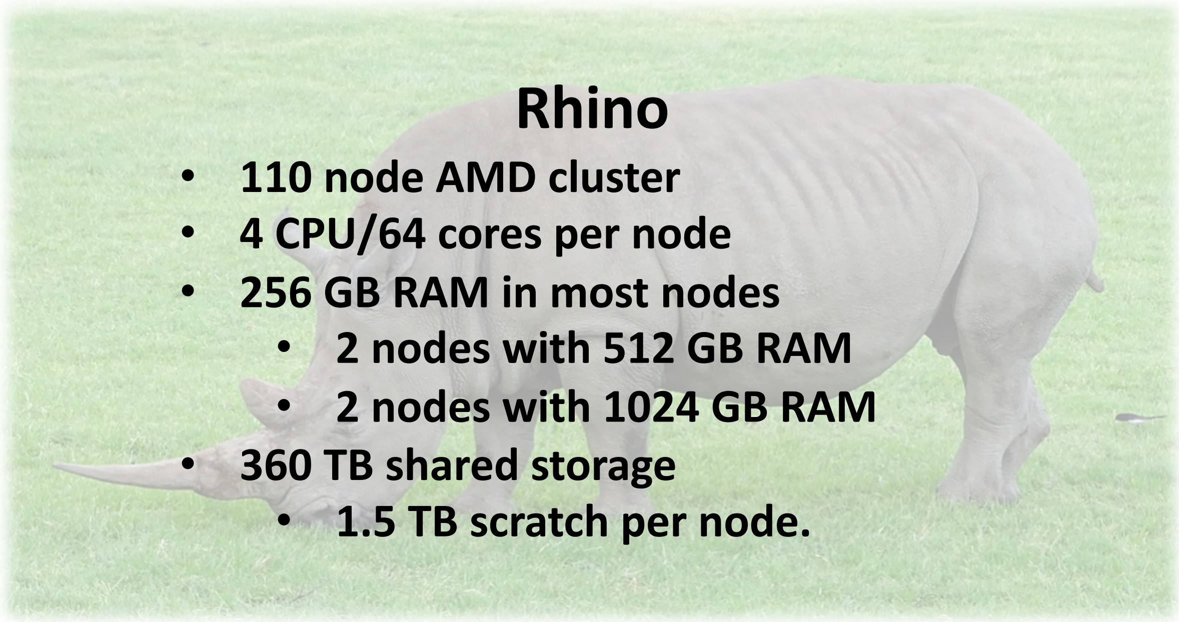
than one could get out of a typical desktop computer or workstation

in order **to solve large problems** in science, engineering, or business.

https://insidehpc.com/hpc-basic-training/what-is-hpc/

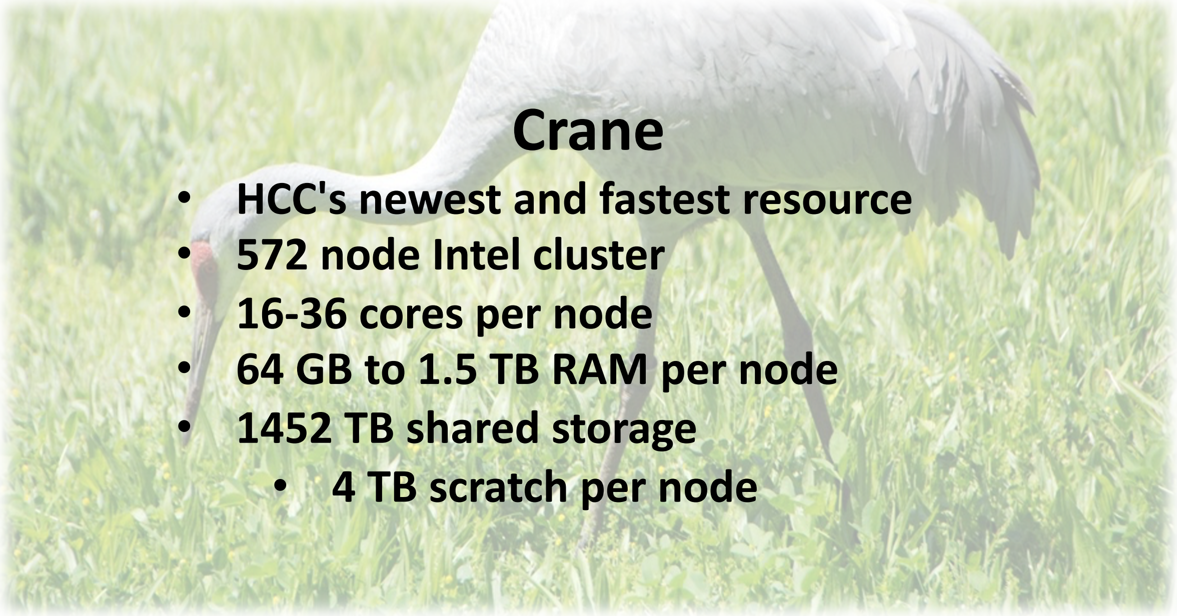
What is a Computing Cluster?





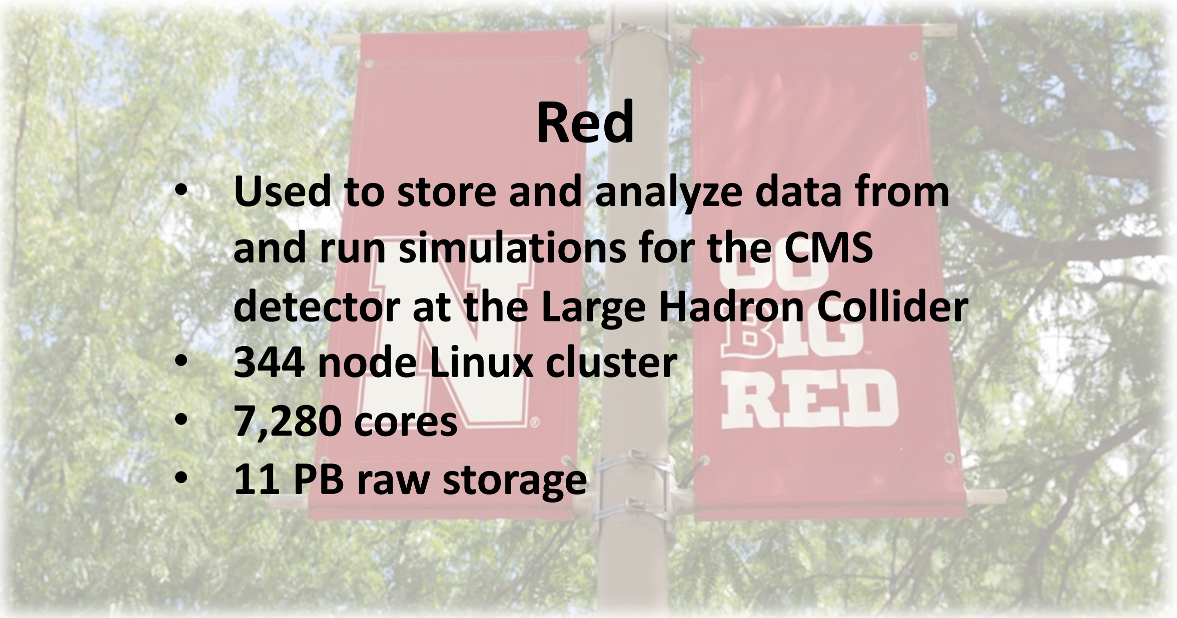
Rhino

- 110 node AMD cluster
- 4 CPU/64 cores per node
- 256 GB RAM in most nodes
  - 2 nodes with 512 GB RAM
  - 2 nodes with 1024 GB RAM
- 360 TB shared storage
  - 1.5 TB scratch per node.



Crane

- HCC's newest and fastest resource
- 572 node Intel cluster
- 16-36 cores per node
- 64 GB to 1.5 TB RAM per node
- 1452 TB shared storage
  - 4 TB scratch per node



Red

- Used to store and analyze data from and run simulations for the CMS detector at the Large Hadron Collider
- 344 node Linux cluster
- 7,280 cores
- 11 PB raw storage

Total Resources

almost 30k cores
approximately 15 PetaBytes of storage
64 GB to 1 terabyte memory per node

Anvil

- HCC’s Cloud: uses the OpenStack framework
- Customizable virtual machines
- For projects not well served by a traditional Linux environment:
  - interactive environments or alternate operating systems
  - projects that require root access or dedicated resources
  - test cluster environments

Attic

- Near-line data archive
- Backed up in Lincoln and Omaha for disaster tolerance
- 10 Gb/s transfer speed to and from the clusters when using Globus Connect
- Cost lower than commercial cloud services

What we provide

- Free shared resources to all NU students, staff and faculty
- Dedicated resources maintained by HCC's System Administrators
- Educational services through hands-on workshops and group tutorials
- Consultations with Research Computing experts
- Extended computing resources offered through partnerships with:

Open Science Grid

High Throughput (Grid) Computing spanning over 125 Institutions nationwide

XSEDE

Extreme Science and Engineering Discovery Environment

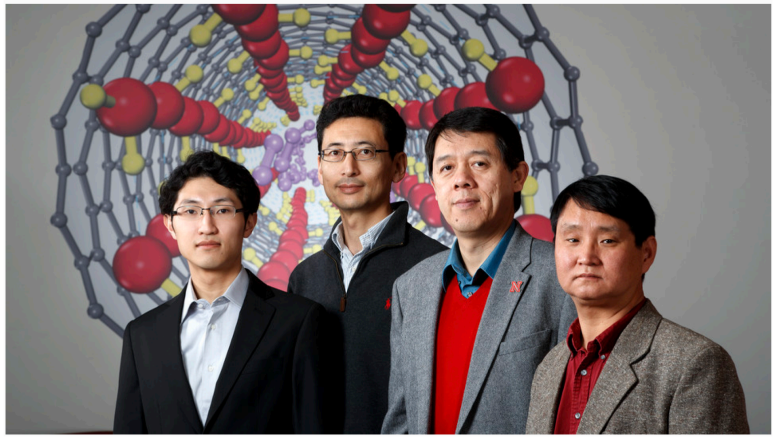
Petascale HPC Computing, Training and Collaborative Support Service

Research Done at HCC

April 3, 2017

Study finds support for new forms of liquid water

by Scott Schrage | University Communication



July 12, 2016

NU agencies lead project to help MENA region respond to drought

by Shawna Richter-Ryerson | Natural Resources

February 12, 2016

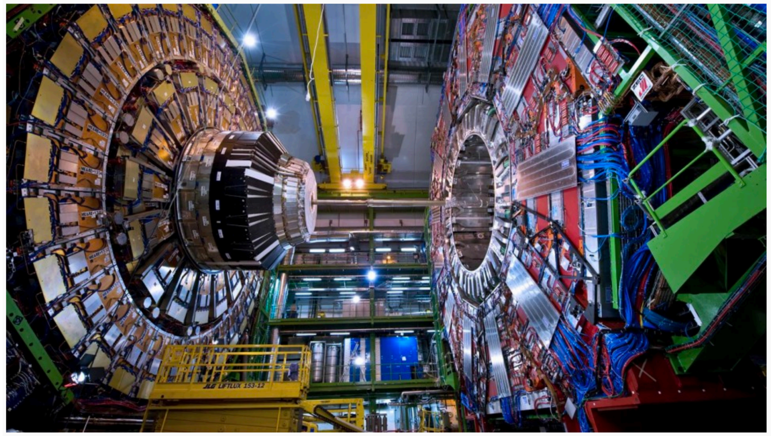
Husker supercomputers help gravitational wave discovery

by Scott Schrage | University Communication

April 6, 2015

UNL physicists expand roles with reboot of Large Hadron Collider

by Scott Schrage | University Communication



September 12, 2016

What makes a bestselling book? Ask Tusker

NEBRASKA'S MATT JOCKERS USES SUPERCOMPUTER TO CRACK BESTSELLER CODE

by Leslie Reed | University Communication

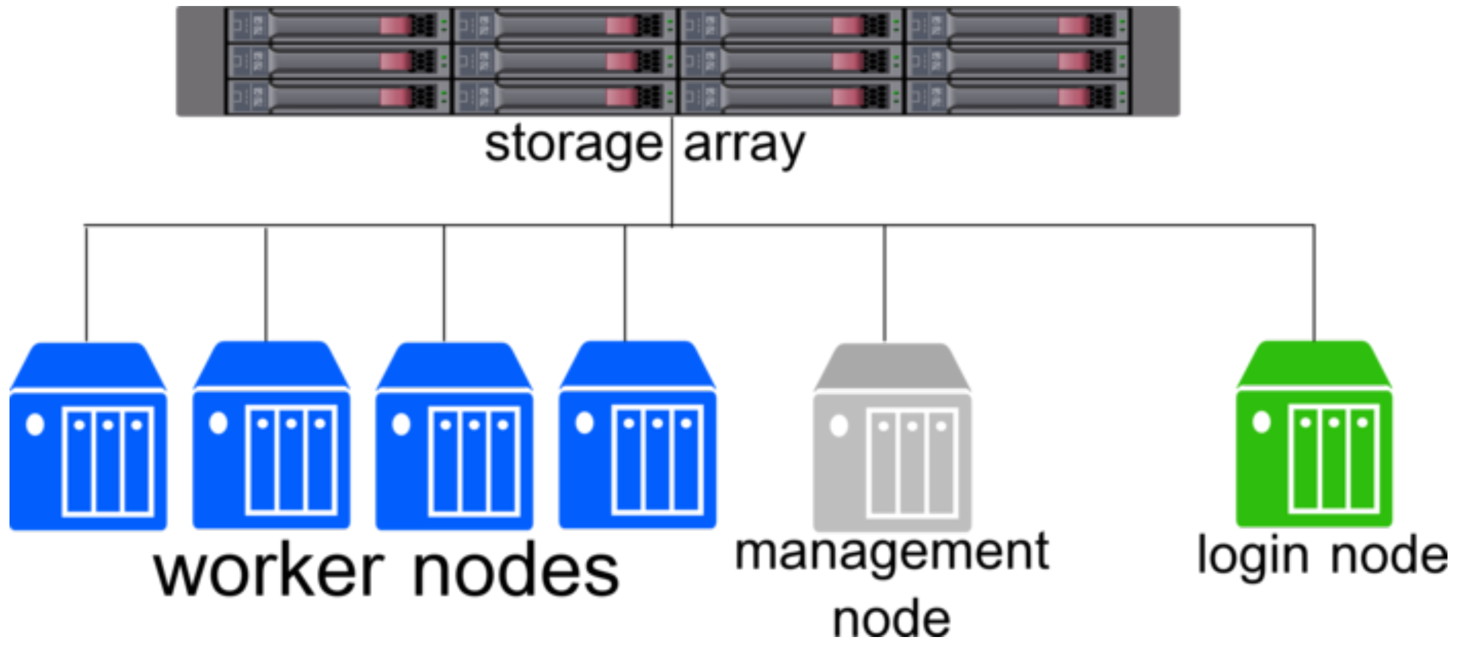
December 22, 2017

Nebraska Unwrapped: CB3 builds on momentum

by Deann Gayman | University Communication

Running Jobs

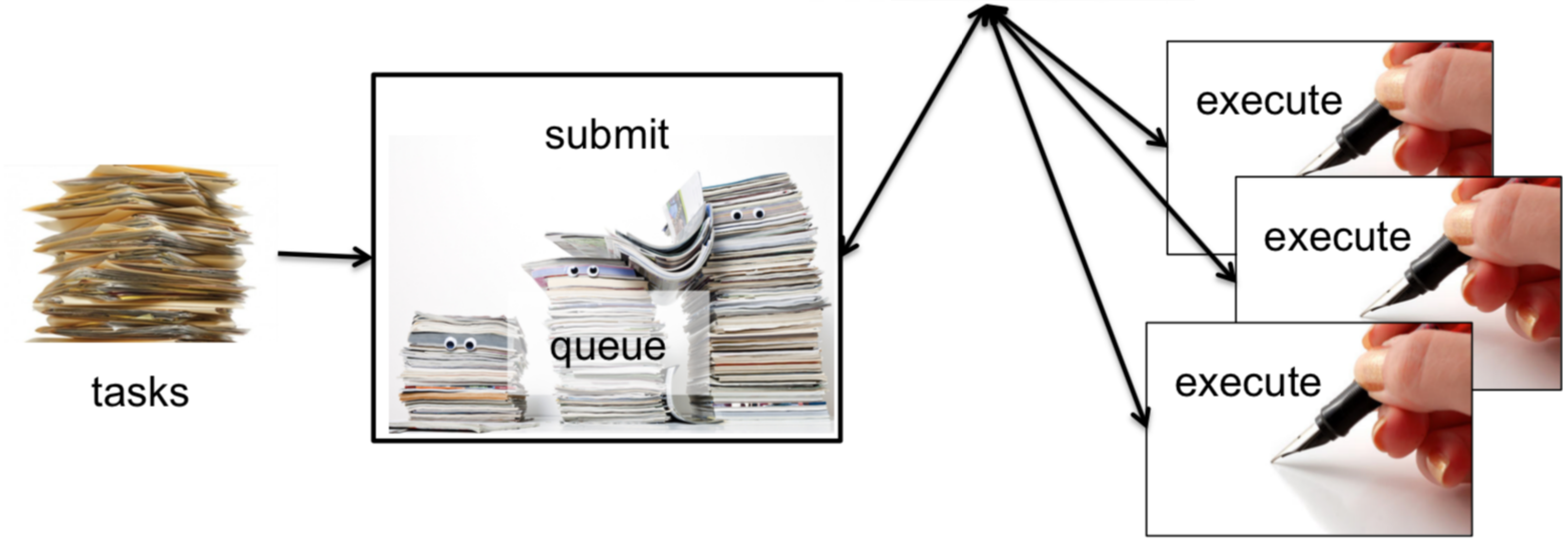
- All software and analysis must be done on the worker nodes
- Processes started on the login node will be killed
    - Limit usage to brief, non-intensive tasks like file management and editing text files



SLURM

Simple Linux Utility for Resource Management

- Open source, scalable cluster management and job scheduling system
- Used on ~60% of the TOP500 supercomputers
- Jobs are assigned based on user priority



Interactive vs Batch Jobs

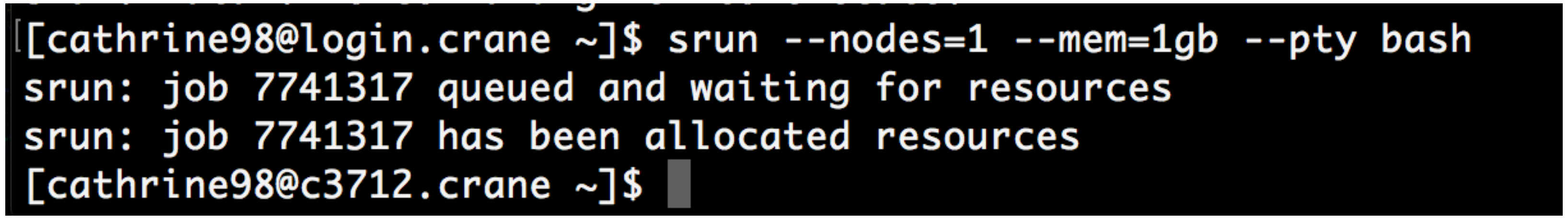
Interactive Jobs

- Run commands and see output immediately on screen
- Uses **`srun`**

Batch Jobs

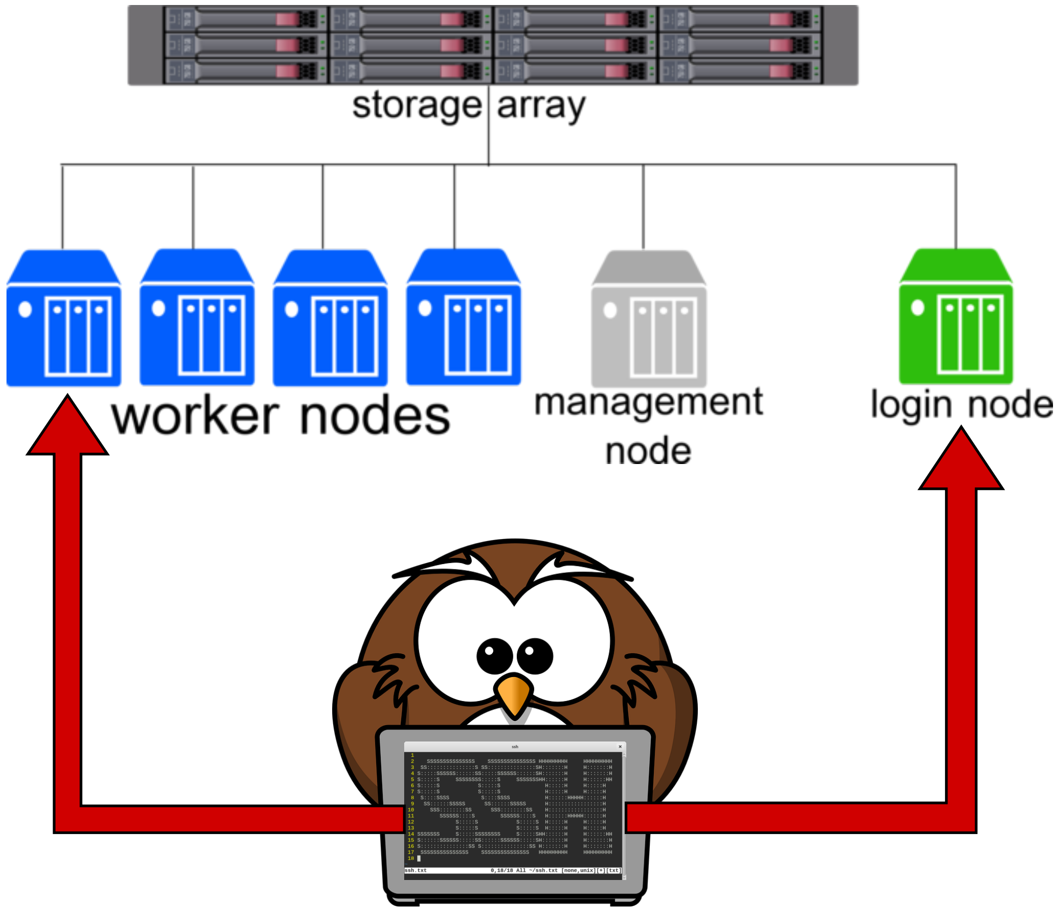
- Submit a script containing commands, wait for the job to finish, then view output from a file
- Uses **`sbatch`**

Interactive Jobs



```
[cathrine98@login.crane ~]$ srun --nodes=1 --mem=1gb --pty bash
srun: job 7741317 queued and waiting for resources
srun: job 7741317 has been allocated resources
[cathrine98@c3712.crane ~]$ 
```

- Once resources are allocated, commands can be input interactively
- Output is directed to the screen
- Once the requested time is up, you will receive a 32 second warning before the job is killed.

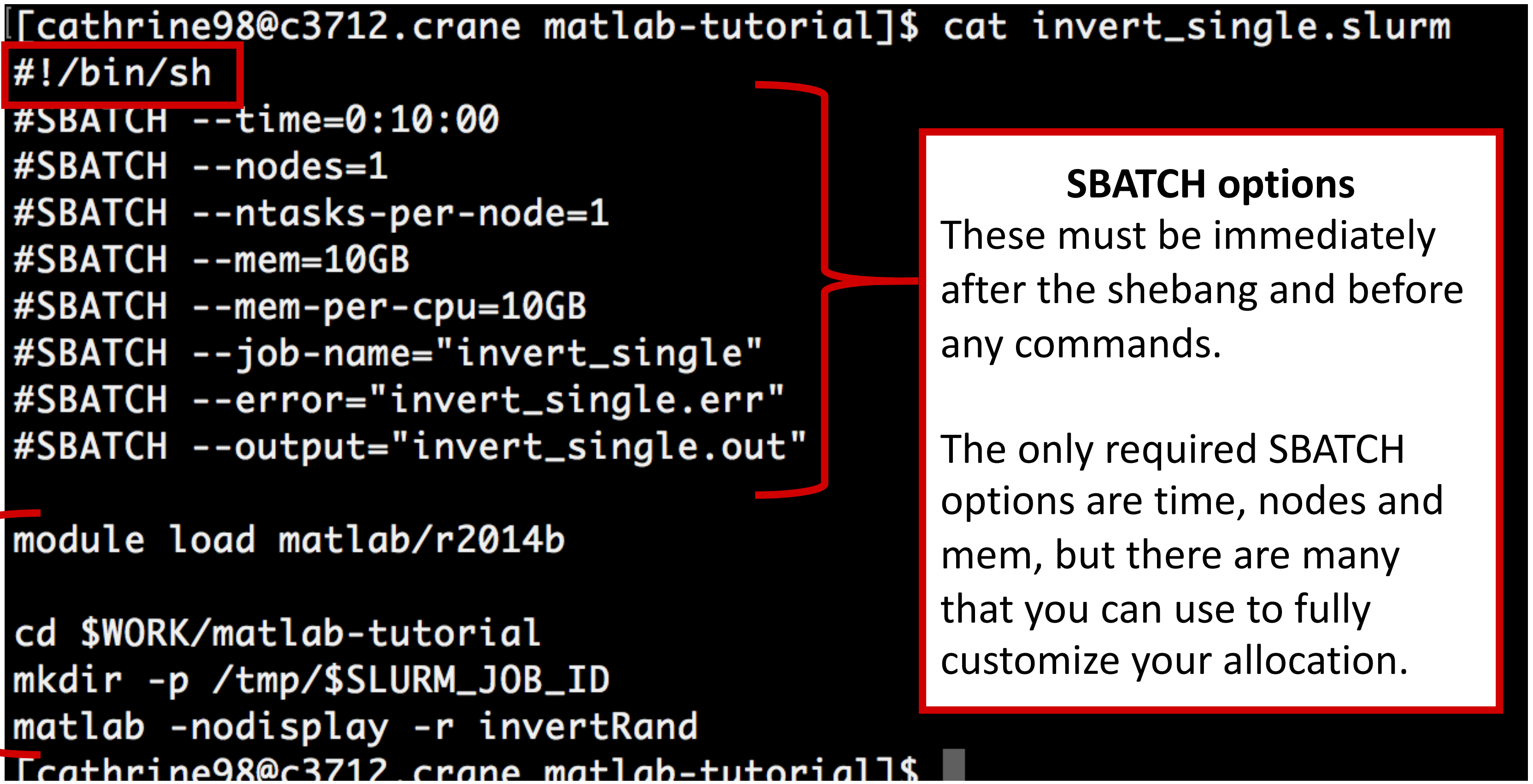


Batch Jobs

- To create a batch job, the user must first make a submit script
  - Submit scripts include all job resource information and necessary commands
  - **If the job exceeds the requested memory or time, it will be killed.**
- Submit script is then added to the job queue using the **`sbatch`** command
- **`squeue`** will show queued and running jobs
- **`sacct`** can be used to find information about completed jobs

Submit Scripts

Shebang
The shebang tells Slurm what interpreter to use for this file. This one is for the shell (Bash)



```
[cathrine98@c3712.crane matlab-tutorial]$ cat invert_single.slurm
#!/bin/sh
#SBATCH --time=0:10:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem=10GB
#SBATCH --mem-per-cpu=10GB
#SBATCH --job-name="invert_single"
#SBATCH --error="invert_single.err"
#SBATCH --output="invert_single.out"

module load matlab/r2014b

cd $WORK/matlab-tutorial
mkdir -p /tmp/$SLURM_JOB_ID
matlab -nodisplay -r invertRand
[cathrine98@c3712.crane matlab-tutorial]$
```

SBATCH options
These must be immediately after the shebang and before any commands.

The only required SBATCH options are time, nodes and mem, but there are many that you can use to fully customize your allocation.

Commands
Any commands after the SBATCH lines will be executed by the interpreter specified in the shebang – similar to what would happen if you were to type the commands interactively

Common SBATCH Options

Command	What it does
`--nodes`	Number of nodes requested
`--time`	Maximum walltime for the job – in DD-HHH:MM:SS format – maximum of 7 days on batch partition
`--mem`	Real memory (RAM) required per node - can use KB, MB, and GB units – default is MB Request less memory than total available on the node - The maximum available on a 512 GB RAM node is 500, for 256 GB RAM node is 250
`--ntasks`	Number of tasks – used to request a specific number of cores
`--mem-per-cpu`	Minimum of memory required per allocated CPU – default is 1 GB
`--output`	Filename where all STDOUT will be directed – default is slurm-<jobid>.out
`--error`	Filename where all STDERR will be directed – default is slurm-<jobid>.out
`--job-name`	How the job will show up in the queue

For more information:

- `sbatch –help`
- SLURM Documentation: https://slurm.schedmd.com/sbatch.html

Determining Parameters

How many nodes/memory/time should I request?

- **Short answer:** We don’t know.
- **Long answer:** The amount of time and memory required is highly dependent on the application you are using, the input file sizes and the parameters you select.
  - Talk with someone else who has used the software before
  - Trial and error
    - Start with a bit more than what you’ve used before and increment as needed
    - Check utilization after each job: https://hcc.unl.edu/docs/submitting_jobs/monitoring_jobs/

Additional SBATCH Options

`--begin:<time>`	**The controller will wait to allocate the job until the specified time** • Specific Time: HH:MM:SS • Specific Date: MMDDYY or MM/DD/YY or YYY-MM-DD • Specific Date and Time: YYYY-MM-DD[THH:MM:SS] Keywords can be used – now, today, tomorrow – Can also be relative in format “now+<time>”
`--deadline=<time>`	**Remove the job if it cannot finish before the deadline** Valid time formats: • HH:MM[:SS] [AM|PM] • MMDD[YY] or MM/DD[/YY] or MM.DD[.YY] • MM/DD[/YY]-HH:MM[:SS] • YYYY-MM-DD[THH:MM[:SS]]]
`--hold`	**Will hold the job in “held state” until released manually using the command `scontrol release <job_id>`**
`--immediate`	**Will only release the job if the resources are immediately available**
`--mail-type=<type>`	**Notify user by email when certain event types occur.** Valid type include: BEGIN, END, FAIL, ALL, TIME_LIMIT, TIME_LIMIT_X (When X% of the time is up, where X is 90, 80, or 50)
`--mail-user=<user_email>`	**Specify an email to send event notifications to**
`--open-mode=<append|truncate>`	**Specify how to open output files** – default is truncate
`--test-only`	**Validates the script and returns a starting estimate based on the current queue and job requirements** Does not submit the job
`--tmp=<MB>`	**Minimum amount of temporary disk space on the allocated node**

Environmental Variables

- Can be used in the command section of a submit file (passed to scripts or programs via arguments)
- Cannot be used within an #SBATCH directive
    - Use Replacement Symbols instead

Environment Variable	Description
$SLURM_JOB_ID	batch job id assigned by Slurm upon submission
$SLURM_JOB_NAME	user-assigned job name
$SLURM_NNODES	number of nodes
$SLURM_NODELIST	list of nodes
$SLURM_NTASKS	total number of tasks
$SLURM_QUEUE	queue (partition)
$SLURM_SUBMIT_DIR	directory of submission
$SLURM_TASKS_PER_NODE	number of tasks per node

Replacement Symbols

Symbol	Value
%A	Job array’s master job allocation number
%a	Job array ID (index) number
%j	Job allocation number (job id)
%N	Node name – will be replaced by the name of the first node in the job (the one that runs the script)
%u	User name
%%	The character “%”

- A number can be placed between % and the following character to zero-pad the result
- For example:
  - job%j.out would create job7773455.out for job_id=7773455
  - job%9j.out would create job007773455.out for job_id=7773455

Advanced SLURM Options

Job Arrays: https://hcc.unl.edu/docs/submitting_jobs/submitting_a_job_array/

- Submits a specified number of identical jobs
- Usage: **`#SBATCH --array=<array numbers or ranges>`**
- To cancel array jobs: **`scancel <job_id>_<array numbers>`**

Dependencies: https://hcc.unl.edu/docs/submitting_jobs/job_dependencies/

- Queues jobs that depend on the completion of previous job(s)
- Usage: **`#SBATCH --dependency=<when_to_execute>:<job_id>`**
    - After successful completion – afterok:<job_id>
    - After non-successful completion – afternotok:<job_id>

Canceling Jobs: `scancel`

- Use to cancel jobs prior to completion
- Usage: **`scancel <job_id>`**
- Use other arguments to cancel multiple jobs at once or combine both to prevent accidentally canceling the wrong job
- Other arguments:

Argument	Cancel…
`--name=<job_name>`	jobs with this name
`--partition=<partition>`	jobs in this partition
`--user=<user_name>`	jobs of this user
`--state=<job_state>`	jobs in this state Valid states: PENDING, RUNNING, and SUSPENDED

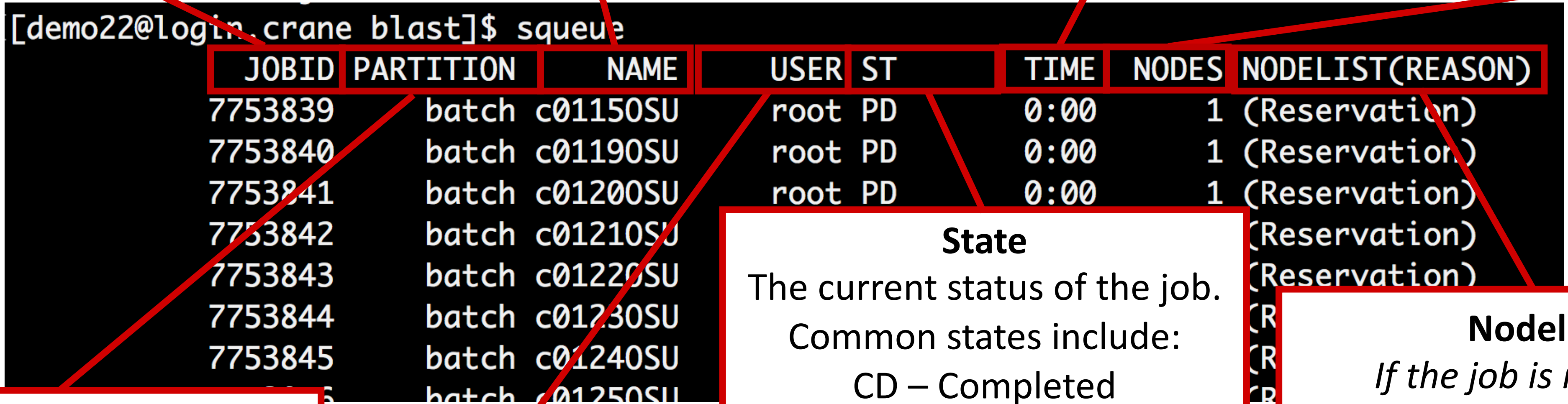
squeue

Job ID
The ID number assigned to your job by Slurm

Name
The name you gave the job as specified in the submit script

Time
The length of time the job has been running

Nodes
The number of nodes the job is running on



```
[demo22@login.crane blast]$ squeue
   JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
 7753839     batch c01150SU     root PD       0:00      1 (Reservation)
 7753840     batch c01190SU     root PD       0:00      1 (Reservation)
 7753841     batch c01200SU     root PD       0:00      1 (Reservation)
 7753842     batch c01210SU                               (Reservation)
 7753843     batch c01220SU                               (Reservation)
 7753844     batch c01230SU                               (R
 7753845     batch c01240SU                               (R
             batch c01250SU                               (R
```

Partition
The partition the job is running on or assigned to

User
The user that owns the job

State
The current status of the job.
Common states include:
CD – Completed
CA – Canceled
F – Failed
PD – Pending
R – Running

Nodelist
If the job is running:
the names of the nodes the job is running on
If the job is pending:
the reason the job is pending

For more information: https://slurm.schedmd.com/squeue.html

Common Reason Codes

Job Reason Codes	Description
Dependency	This job is waiting for a dependent job to complete.
NodeDown	A node required by the job is down.
PartitionDown	The partition (queue) required by this job is in a DOWN state and temporarily accepting no jobs, for instance because of maintainance. Note that this message may be displayed for a time even after the system is back up.
Priority	One or more higher priority jobs exist for this partition or advanced reservation. Other jobs in the queue have higher priority than yours.
ReqNodeNotAvail	No nodes can be found satisfying your limits, for instance because maintainance is scheduled and the job can not finish before it
Reservation	The job is waiting for its advanced reservation to become available.

More information:

- squeue --help
- https://slurm.schedmd.com/squeue.html

Common `squeue` Options

Option	Displays information about…
-u <user_name> --user=<user_name>	jobs owned by the specified user_name(s) *
-j <job_list>	specified job(s) *
-p <part_list>	jobs in a specified partition(s) *
-t <state_list>	jobs in the specified state(s) – {PD, R, S, CG, CD, CF, CA, F, TO, PR, NF} *
-i <interval> --interate= <interval>	jobs repeatedly reported at intervals (in seconds)
-S <sort_list> --sort=<sort_list>	jobs sorted by specified field(s) *
--start	pending jobs and scheduled start times

* Indicates arguments that can take a comma-separated list

For more options: https://slurm.schedmd.com/squeue.html

Using srun to monitor batch jobs

1. Connect to the node running the job:
   - **`srun -j <job_id> --pty bash {or top}`**
   - **`srun -nodelist=<node_id> --pty bash {or top}`**
2. Monitor:
   - **`top`** (if not already running)
     - Use to monitor core use – ideal for multi-core processes
     - Press 'u' to search for your username
   - **`cat /cgroup/memory/slurm/uid_<uid>/job_<job_id>/memory.max_usage_in_bytes`**
     - Use to monitor memory use
     - To determine your uid use: **`id -u <user_name>`**
     - Match with **`watch`**
       - -n to specify a refresh interval - default is 2 seconds
       - CTRL + C to exit

Exercises

1. Clone the job-examples repo in your $WORK directory with the command:

```
git clone http://github.com/unlhcc/job-examples
```

2. `` `**cd**` `` into the job-examples directory and look at the subdirectories. How many different job examples are there? What applications do you recognize or use?

3. `` `**cd**` `` into one of the example directories (such as \`R\` or \`python\`) and examine the contents of the files there, focusing on the one ending in \`.submit\`. Identify what resources are being requested. Can you interpret what this job is going to do?

Once you have finished, please click "yes" in the Zoom participants panel.

If you have issues, please click "no" in the Zoom participants panel.