

Intro to Monsoon and Slurm

2026-02-12 presentation with Jason Buechler

These slides:

<https://rcdata.nau.edu/hpcpub/workshops/odintro.pdf>



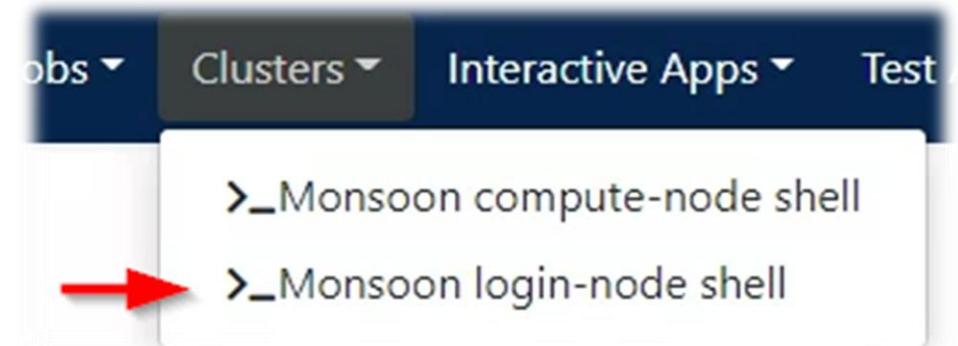
NAU NORTHERN ARIZONA UNIVERSITY

Get logged in!



From a computer:

- Connect to the NAU VPN *if off-campus!*
 - Info: <https://in.nau.edu/its/remote-services>
- Open any web browser
- Login to <http://ondemand.hpc.nau.edu>
 - Standard 'abc123' Louie ID & password
- Optional: try out the command-line
 - Clusters > Monsoon login-node shell



These slides:

<https://rcdata.nau.edu/hpcpub/workshops/odintro.pdf>

- HIT RECORD! 😊

Introductions



- Introduce yourself
 - Name
 - Department / Group
 - What project(s) do you plan to use monsoon for?
 - Linux or Unix experience
 - Previous HPC experience?

Don't worry: few researchers have prior linux/HPC experience!

I am Jason Buechler

- Aero, Mech Engineering
fluid-thermodynamics, renewable energy, & grid-engineering
- With Monsoon since 2019
- Linux/supercomputing since 2000

Agenda

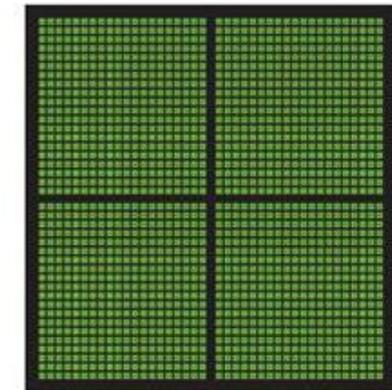
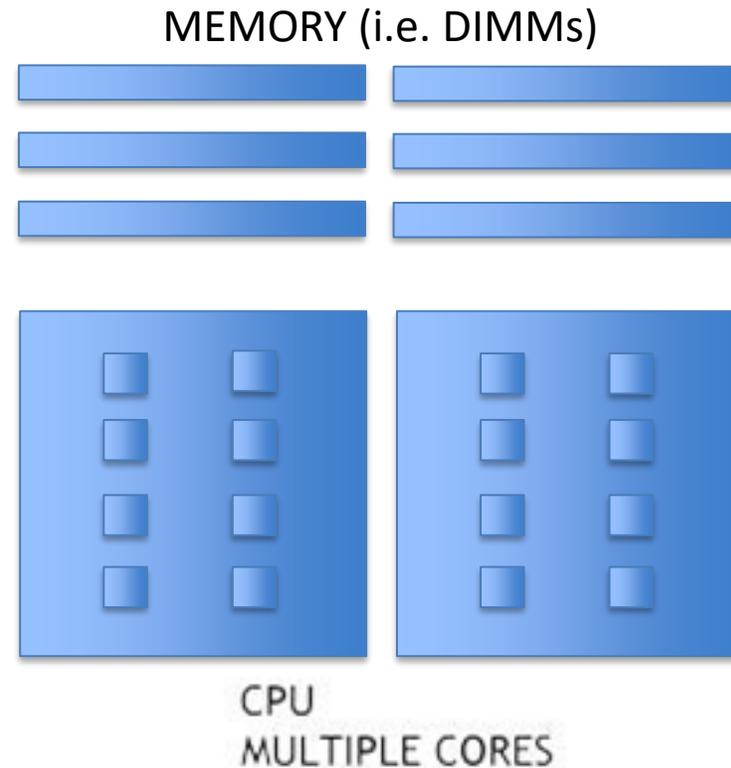


- (High-Performance) Cluster education
 - What is a cluster, exactly?
 - Job-queues, scheduling, and resource management
- Monsoon Cluster orientation
 - How do I use this cluster?
 - Group resource limits
 - Exercises
 - Question and answer

Cluster (of) resources

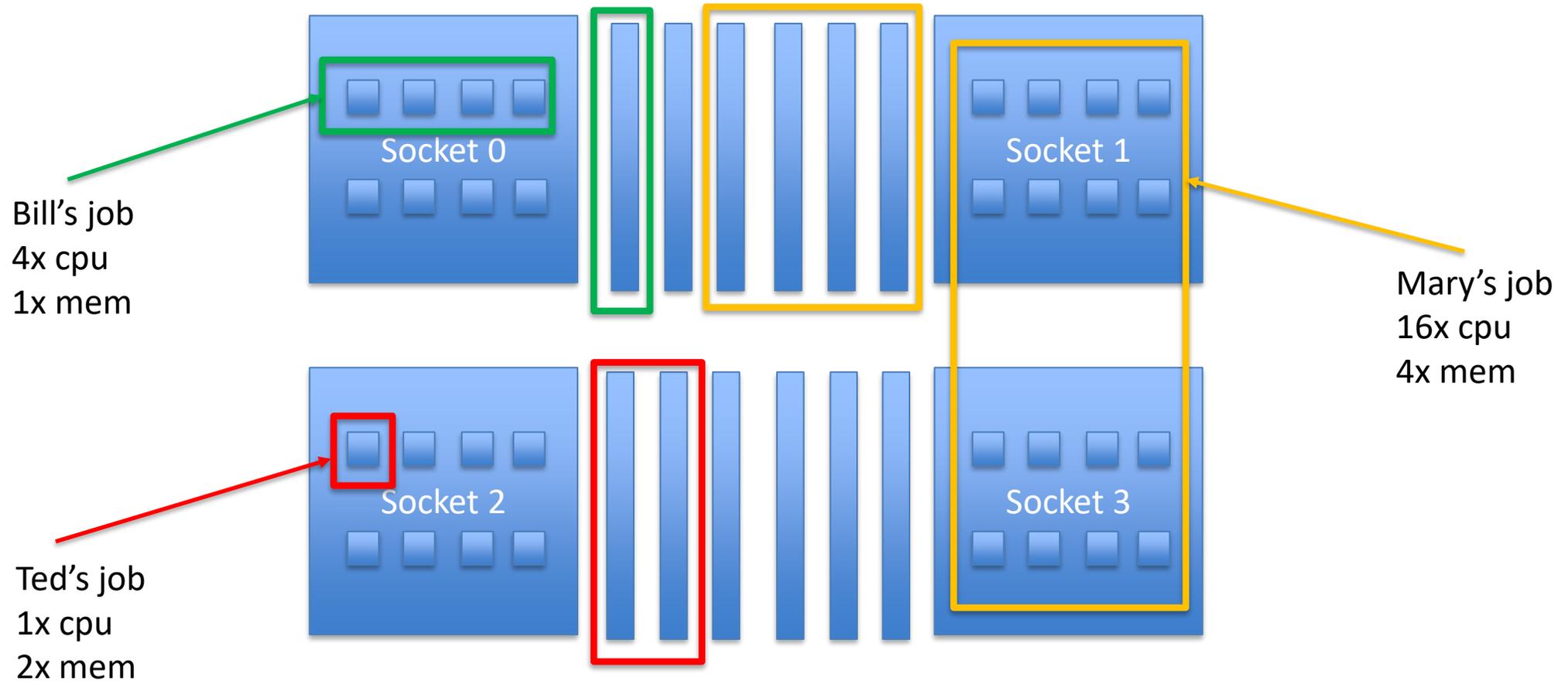


- Node itself
 - Memory
 - CPU's
 - GPU's
- Networking
- Licenses



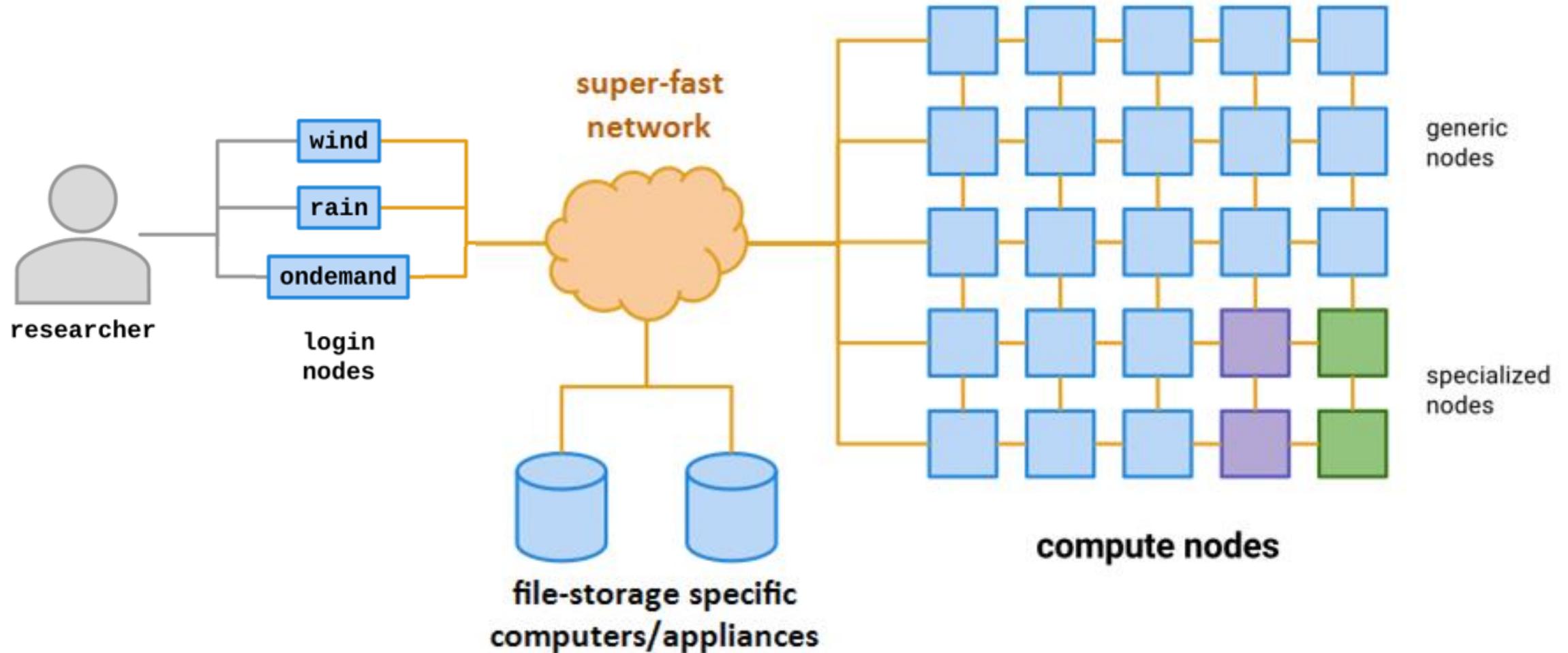
GPU
THOUSANDS OF CORES

Inside a (single) compute-node

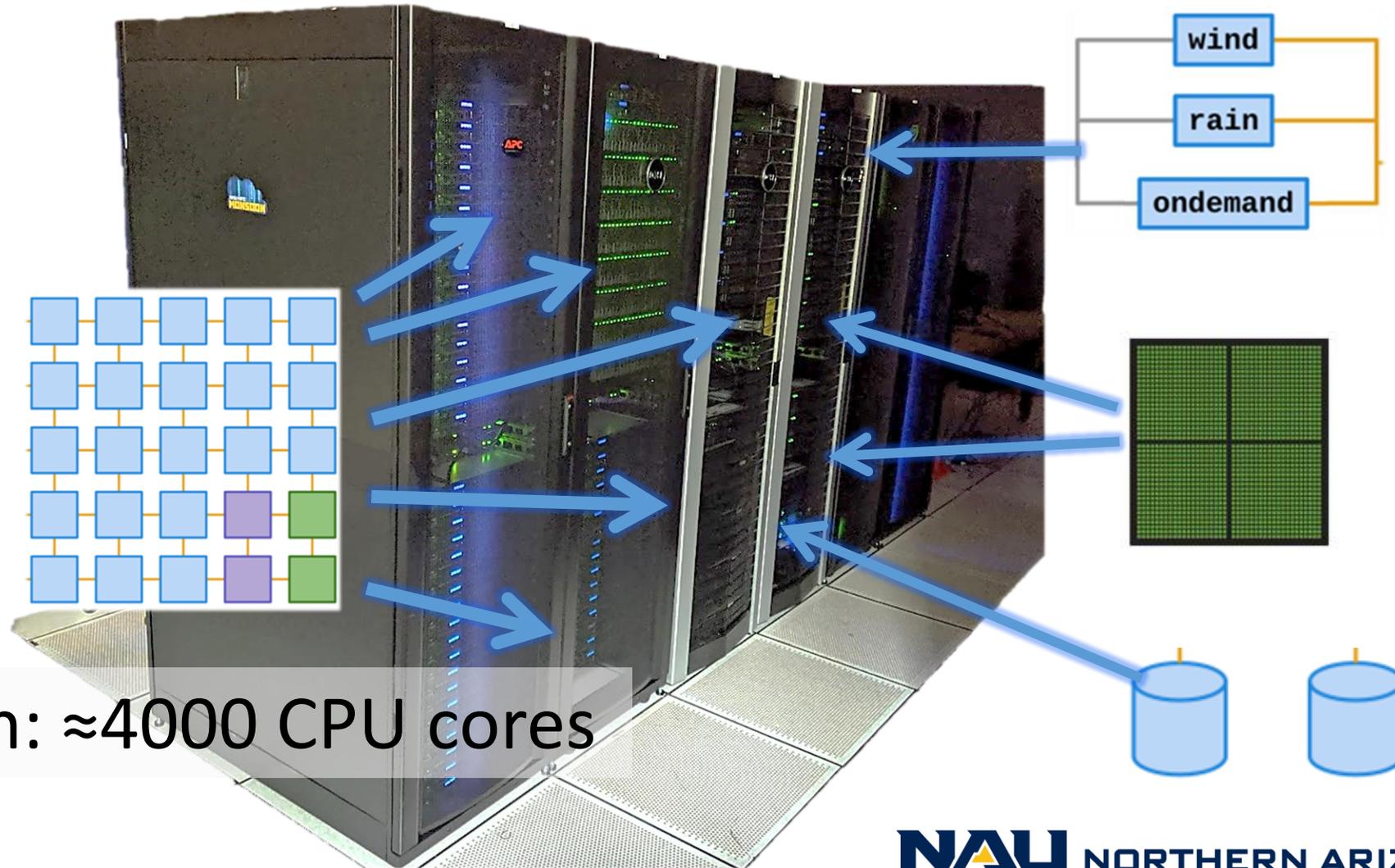


Cluster =

Login-nodes + Compute-nodes + More



Monsoon



Monsoon: ≈ 4000 CPU cores

Largest Cluster!



El Capitan: 11.3M cores

Small Cluster!



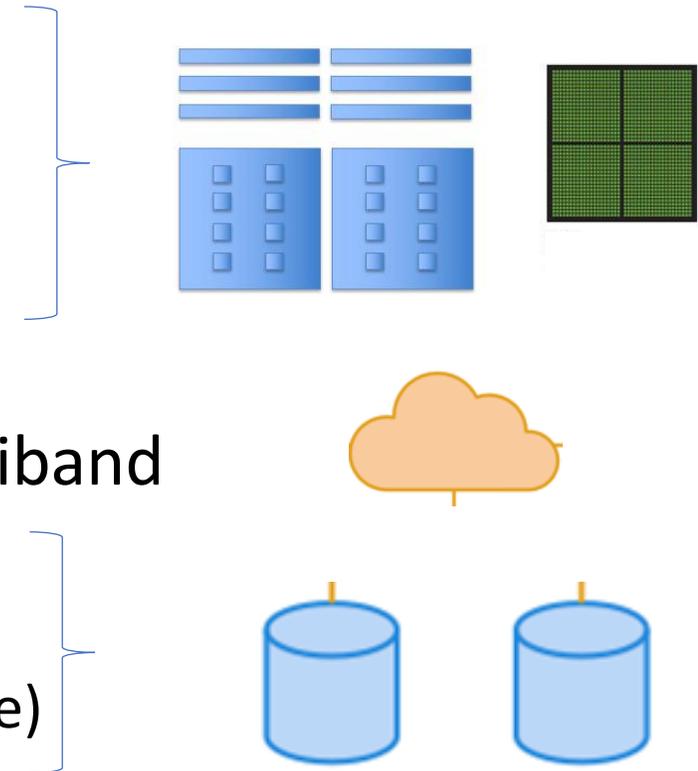
2 cores? 2 jobs!

Monsoon Today

(from <https://in.nau.edu/arc/details>)



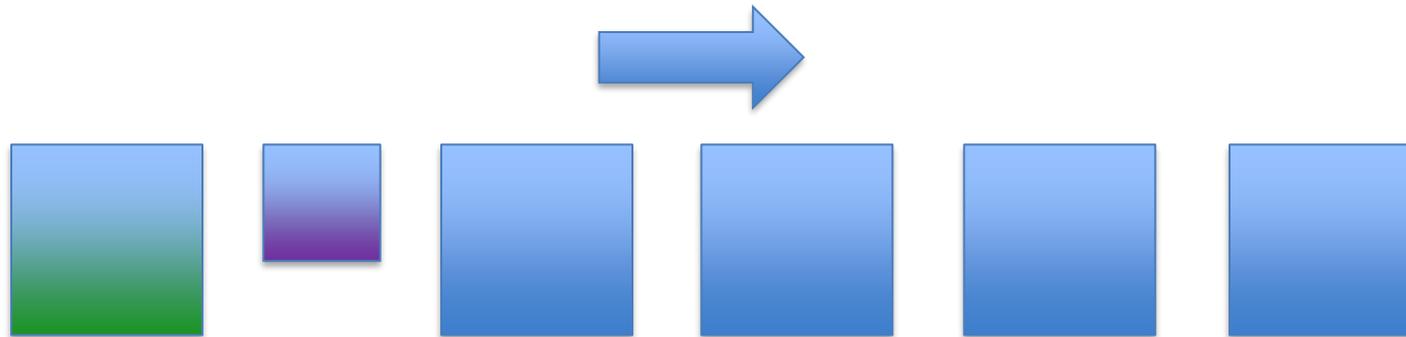
- The Monsoon cluster is a resource available to the NAU research enterprise
- 107* compute-nodes (cn1 ~ cn108)
 - 26TB memory - 128GB/node min, 2TB max
 - 27 GPUs, NVIDIA Tesla K80, P100, V100, A100
 - 4048 cores (Intel + AMD)
- Red Hat Enterprise Linux 8.10
- High speed interconnect: FDR, and HDR Infiniband
- Storage
 - 1PB *scratch* high-speed storage (Lustre)
 - 1.6PB *long-term* storage (proprietary object-store)



What is a queue?



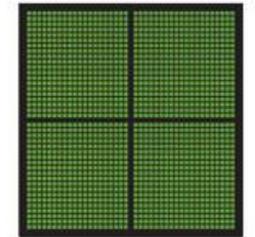
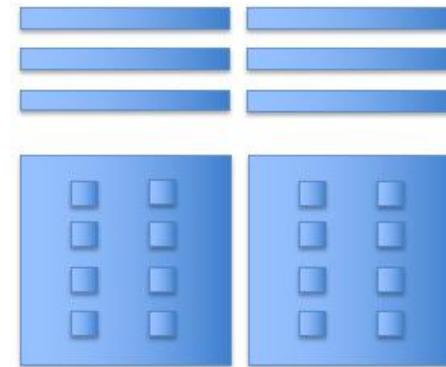
- Normally thought of as a line, FIFO (Line at Starbucks)
- Queues on a cluster can be as basic as a FIFO, or more advanced with dynamic priorities taking into consideration **many factors**
- A ***scheduler*** needs to know what resources are available on the cluster in order to make accurate scheduling decisions



Resource Manager



- Resource availability changes by the second!
- Assignment of work on a cluster is carried out most efficiently with the **scheduler** and **resource manager** working together
- Monitoring resource availability and health
 - Accounting of resources
 - Allocation of resources
 - Execution of resources



Slurm ... yummm



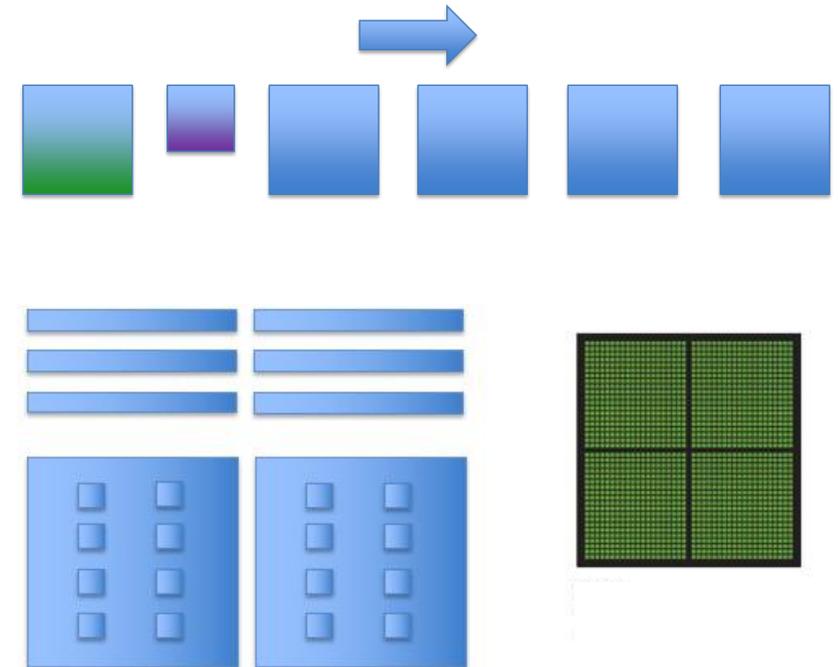
- Slurm (*Simple* Linux Utility for Resource Management)
- Excellent resource manager and scheduler
- Precise control over resource requests
- Developed at LLNL, continued by SchedMD
- Used everywhere from small clusters to the largest clusters:
 - El Capitan (#1), 11.3M cores, 1742 PF, 29.6 MW - USA
 - Frontier (#2), 9.1M cores, 1353 PF, 24.6 MW - USA



Our Scheduling Goals



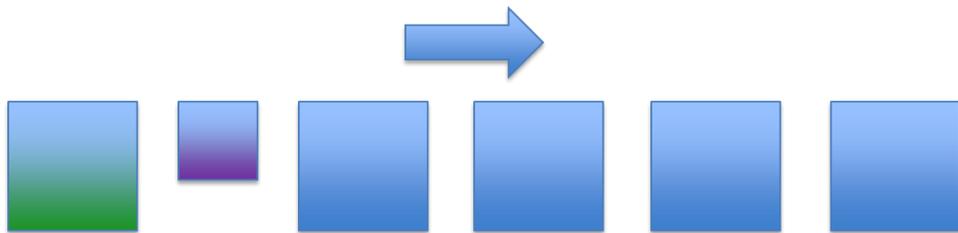
- Optimize quantity of work
- Optimize usage of resources
- Service all users and projects justly
- Make scheduling decisions transparent



Monsoon scheduling



Monsoon utilizes a combination of scheduling methods to optimize cluster productivity and resource usage:



- FIFO
 - Simply first in first out
- Backfill
 - Runs smaller jobs with lower resource requirements while larger jobs wait for higher resource requirements to be available
- Fairshare
 - Prioritizes jobs based on multiple factors such as users' recent resource consumption
- Additional factors
 - Including job-specific and user-specific

Factors attributing to priority

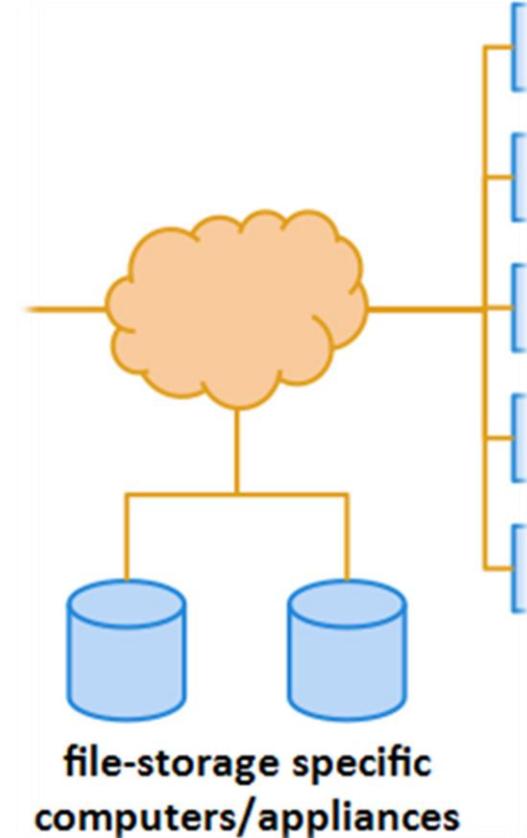


- Fairshare (predominant factor)
 - Priority points determined on users' recent resource usage
 - Decay half life over 12 hours
- Age – how long has the job sat pending
- Job size - the number of nodes/cpus a job is requesting
- QOS (Quality of Service)
 - Some QOS have higher priority than others, for instance: debug

Storage



- /home – 20 GB quota
 - Keep your scripts and executables here
 - Backed up to tape
 - Snapshotted twice a day: /home/.snapshot
 - Please do not write job output (logs, results) here!!
 - Run the command “getquotas” now
- /scratch – 15000 GB quota (also 2M files quota)
 - 1PB total space, 30 day retention
 - Very fast storage, capable of 20GB/sec
 - Checkpoints, logs
 - Keep all temp/intermediate data here
 - Should be your default location to perform input/output

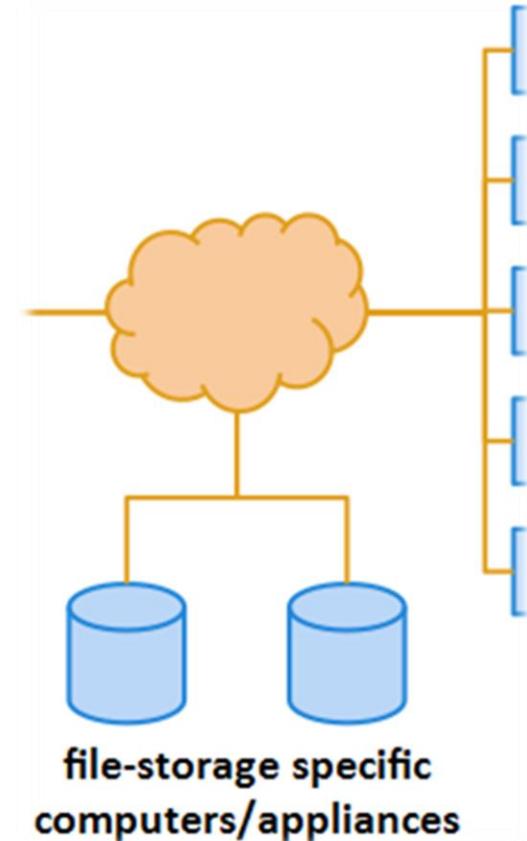


Storage



- /projects – 1.6 PB
 - Long-term storage project shares
 - 5TB is assigned to faculty member for group to share
 - \$48/TB/year above 5TB
 - Snapshots available
 - Option to make datasets available to the web
- /common
 - Cluster support share

NOTE: the only storage area on monsoon backed-up* is your /home area (due to size limitations)



Data Workflow



1. Keep scripts and executables in /home
2. Write logs/temp/intermediate data to /scratch/<uid>
3. Copy data to /projects/<group_project>, for group storage and reference in other projects
4. Cleanup /scratch files

** Remember, /scratch is a scratch filesystem, used for high-speed temporary and intermediate data

Remote storage access



Quick transfers via web
<10 GB, <100 files

Ondemand's file-browser, or Globus*

Very large/numerous files

Globus: <https://in.nau.edu/arc/globus>

General purpose

scp: 100's of GBs, 1000's of files
smb: 10's of GBs, 100's of files

SCP/SMB: <https://in.nau.edu/arc/file-management>

scp protocol / shell-command

- GUI apps: **WinSCP** on windows, **Fetch** for mac
- `scp <files> <nauid>@<dtm>:/scratch/<nauid>/`

smb / samba / "shared drive"

- Windows: `\\shares.hpc.nau.edu\cirrus`
- Mac: `smb://shares.hpc.nau.edu/cirrus`

Data transfer nodes



We have dedicated, specialized login-nodes for transferring data



These hosts' names are

`dttn1.hpc.nau.edu` & `dttn2.hpc.nau.edu`



Use dttn1/dttn2 for moving large datasets around on monsoon, and to/from the internet

Enterprise Groups



- NAU has a resource called Enterprise groups. Enterprise Groups are utilized to manage who has access to specific folders and files on the cluster
- They are available to your group on the cluster if you'd like to manage access to your data
- What groups are you in? Run the command “groups”, or “id”
- More info in our FAQs <https://in.nau.edu/arc/faqs>
 - “Creating and Managing an Enterprise Group”

(Software) Modules



- Software environment management handled by the *modules* package management system. This is available through the Command Line Interface (cli)
- `module avail` ...*what modules are available*
- `module load <module name>` ...*load a package module*
- `module list` ...*modules currently loaded*

Modules on Monsoon



What's already available?

`module -d avail`

- Anaconda Python
- R
- Matlab
- Qiime2
- Many bioinformatics programs

```
ricky@wind:~
[ricky@wind ~]$ module --default avail
----- Monsoon Modules (MPI) -----
amber/22
cdo/2.1.0
cp2k/2024.1
elpa/2021.11.001
exabayes/1.5.1-v3
fftw/3.3.10
hdf5/1.12.2-v2
lammps/20240829.1-gpu-mc
madingley/2020-04-01
mrbayes/3.2.7a-v2
ncl/6.6.2
nco/5.3.3
netcdf-c/4.9.0-v3
netcdf-cxx4/4.3.1
netcdf-fortran/4.5.4
netlib-scalapack/2.2.0
nwchem/7.0.2
osu-micro-benchmarks/7
----- Monsoon Modules (Core) -----
R/4.5.1
alsa-lib/1.2.3.2
amd-blis/3.0
amd-libflame/3.0
anaconda2/2019.10
googletest/1.12.1-yjsk
gperf/3.1
grass/7.8.2
gsl/2.6
guppy-cpu/5.0.11c
ninja/1.11.1-7zaz
node/v22.18.0
numactl/2.0.14
oc2/2.0
ollama/0.9.5
```

Requesting Software



- You can install quite a bit of R, and python software yourself!
- For R
 - `module load R`
 - `R`
 - `install.packages(c(package))`
- For python
 - `module load anaconda3`
 - `conda create -n myenv`
 - `conda activate myenv`
 - `conda install package`
- You are also welcome to compile your own programs
- If you'd like our help installing a piece of software, please have your research sponsor request it

More info:

<https://in.nau.edu/arc/installing-software-packages>

MPI



Quick PSA for advanced users

- Message Passing Interface for parallel computing
- Open MPI set as default MPI
- Example MPI job script:
 - `/common/contrib/examples/job_scripts/mpijob.sh`

Interacting with Slurm



1. What resources are needed?
 - 2 **cpus**, 12GB **memory**, for 2 **hours**?
 - You can (must!) start with an educated guess
2. What software environment is needed?
3. What steps are required? (*must prog1 run before prog2?*)
4. Can your work, or project be broken into pieces?
 - Smaller pieces can make the workload more agile
 - Is each step dependent on the last?
 - Is your software multithreaded, uses OpenMP or MPI?

Job Scripts and sbatch



- Except for limited testing and debugging, all jobs on the cluster should be run via a shell script which is typically denoted by the extension `.sh` on the filename
- sbatch shell scripts are composed of **three sections**:
 1. Slurm job parameters (“`#SBATCH ...`”)
 2. Software environment/module loading (“`module load ...`”)
 3. srun job-steps/cmd-statements for actual work (“`srun <command>`”)

Example Job script



- `#!/bin/bash`
 - `#SBATCH --job-name=test`
 - `#SBATCH --time=20:00` # shorter jobs usually start sooner
 - `#SBATCH --chdir=/scratch/NAUID` # default location of files
 - `#SBATCH --output=/scratch/NAUID/output.txt` # the job's output/errors go here
- # replace this module with software-*
- # modules required by your jobscript*
- `module load anaconda3/2025.06` # loads a specific anaconda-python
- # example job commands: each srun command is*
- # a job step, so this job will have 2 steps*
- `srun sleep 300`
 - `srun python -V`

Example Job script (in Ondemand)

A screenshot of a web-based code editor interface. The browser address bar shows "ondemand.hpc.nau.edu". The file path is "/home/jtb49/ondemand/data/sys/myjobs/projects/default/15/main_job.sh". A red arrow points to a "Save" button. The code is as follows:

```
1 #!/bin/bash
2 #SBATCH --job-name=exercise1           # the name of your job
3 #SBATCH --output=/scratch/abc123/output1.txt # this is the file your output and errors go to
4 #SBATCH --time=20:00                   # 20 min, shorter time, quicker start, max run time
5 #SBATCH --chdir=/scratch/abc123       # your work directory ("pwd")
6 #SBATCH --mem=2000                     # 2000MB = 2GB of memory
7 #SBATCH --mail-type=FAIL
8
9 # load a module, for example
10 module load anaconda3
11
12 # Run your application: precede the application command with 'srun'
13 # A couple example applications...
14 srun date
15 srun python --version
16 srun sleep 30
17 srun pwd
18 srun date
19
20
```

- 1. Job parameters (for slurm)
- 2. Software environment setup
- 3. Job commands

Job Parameters



You want	Switches needed
More than one cpu for the job	<code>--cpus-per-task=2</code> (or: <code>-c 2</code>)
To specify an ordering of your jobs	<code>--dependency=afterok:job_id</code> (or: <code>-d job_id</code>)
Split up the output, and errors	<code>--output=result.txt --error=error.txt</code>
To run your job at a particular time/day	<code>--begin=16:00</code> <code>--begin=2026-02-14T12:34:00</code>
Add MPI tasks/ranks to your job	<code>--ntasks=2</code> , or <code>-n 2</code>
To control job failure options	<code>--norequeue</code> <code>--requeue</code>
To receive status email	<code>--mail-type=ALL</code>

More info:

<https://slurm.schedmd.com/sbatch.html>

Constraints and Resources

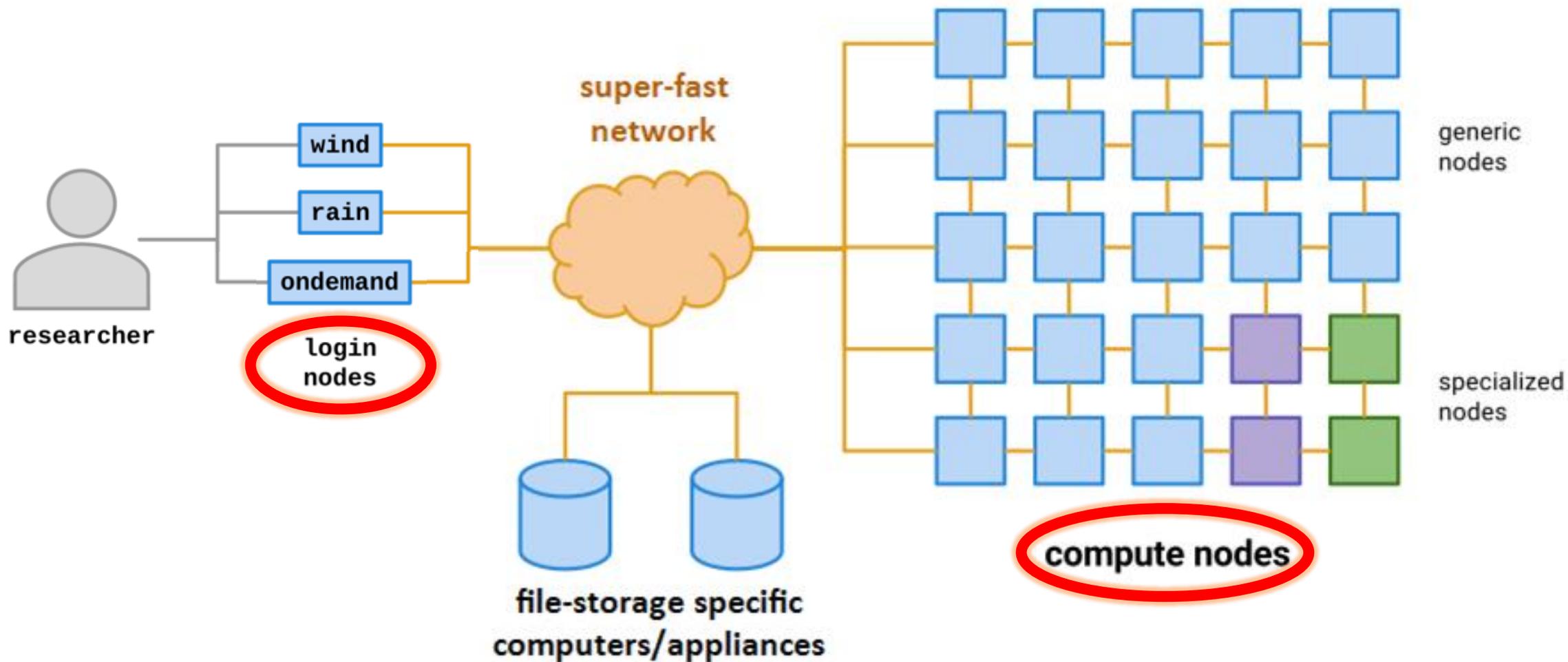


You want	Switches needed
To choose a specific node feature (e.g. avx2)	<code>--constraint=avx2</code>
To use a generic resources (e.g. a gpu)	<code>--gpus=a100, -G1</code>
To reserve a whole node for yourself	<code>--exclusive</code>
To chose a partition	<code>--partition</code>

More info:

<https://slurm.schedmd.com/sbatch.html>

Cluster Review



Login node vs Compute node



- When you log into “monsoon” interactively or via Ondemand you are “placed” on a (shared!) login node.
- The login node is a shared system used solely for:
 - Developing scripts
 - Transferring small data
 - Submitting work to the scheduler
 - Analyzing results
 - Debug work* less than 30 minutes in length
- The compute nodes are what make the cluster powerful!

Interacting with Monsoon



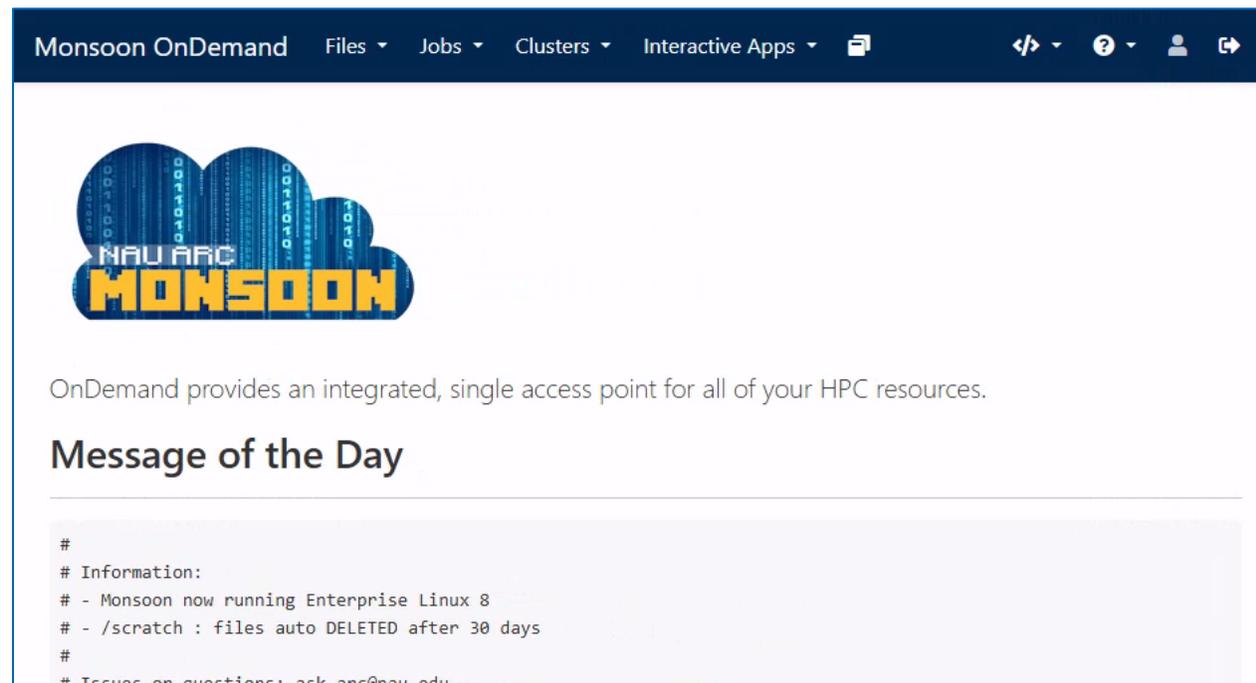
Three Methods (**must be on NAU network or NAUVPN**):

- Connect to OpenOndemand web interface at: <https://ondemand.hpc.nau.edu>
- Via SSH protocol in a command-line shell
 - Enter “`ssh <nau-id>@<login-node>`”
 - ...within **Powershell** (Windows), or **Terminal** (Mac, Win, Linux)
 - ...using Monsoon’s login nodes:
 - monsoon.hpc.nau.edu (for research)
 - wind.hpc.nau.edu
 - ondemand.hpc.nau.edu
 - rain.hpc.nau.edu (for class work)
- Remote storage access_{_*} (files only -- no command interface)

Ondemand



- Open Ondemand (OOD) is an interactive Graphical User Interface (gui) to the Cluster. You access it from your web browser at <https://ondemand.hpc.nau.edu>



Ondemand File Explorer



- The file explorer is used to explore, and transfer the files in your home, **scratch**, or other areas on the cluster.

The screenshot shows the Ondemand File Explorer interface. The left sidebar displays the Home Directory structure, including folders like Desktop, Downloads, IAB-notebooks, Wolfram Mathematica, _Inline, demo, linux, linux_workshop, lustre, myjobs, old-sas-lic, ondemand, rpmbuild, sasuser.v94, sasuser.v94.bak, scratch, study1, test, test_jobs, and tmp. The main pane shows the current directory /home/wew/ with a table of files and folders. The table has columns for name, size, and modified date. The files listed are:

name	size	modified date
..	<dir>	
Desktop	<dir>	01/18/2018
Downloads	<dir>	08/06/2018
IAB-notebooks	<dir>	02/21/2017
Wolfram Mathematica	<dir>	03/26/2018
_Inline	<dir>	01/09/2018
demo	<dir>	03/19/2018
linux	<dir>	02/23/2017
linux_workshop	<dir>	10/12/2018
lustre	<dir>	12/13/2017
myjobs	<dir>	10/19/2018
old-sas-lic	<dir>	09/05/2017
ondemand	<dir>	10/19/2018
rpmbuild	<dir>	09/28/2018
sasuser.v94	<dir>	12/21/2018
sasuser.v94.bak	<dir>	10/12/2015
scratch	<dir>	12/21/2018

OnDemand Job Composer



- The Job Composer **can be** used to create and run jobs.

The screenshot displays the OnDemand Job Composer interface. The top navigation bar includes "Open OnDemand / Job Composer", "Jobs", "Templates", and a "Help" icon. The main heading is "Jobs".

Below the heading, there are several action buttons: "+ New Job", "Create Template", "Edit Files", "Job Options", "Open Terminal", "Submit", "Stop", and "Delete".

A search bar and a "Show 25 entries" dropdown are present. The main content is a table of jobs:

Created	Name	ID	Cluster	Status
December 21, 2018 11:12am	2samplertest	15728774	Monsoon Cluster	Completed
December 4, 2018 11:20am	(default) Simple Sequential Job		Monsoon Cluster	Not Submitted
November 16, 2018 11:05am	Job from Template	15505031	Monsoon Cluster	Completed
November 15, 2018 12:47pm	job_array.sh	15326951	Monsoon Cluster	Completed

At the bottom of the table, it says "Showing 1 to 4 of 4 entries" and "Previous 1 Next".

On the right side, the "Job Details" panel is visible for the selected job "2samplertest". It shows:

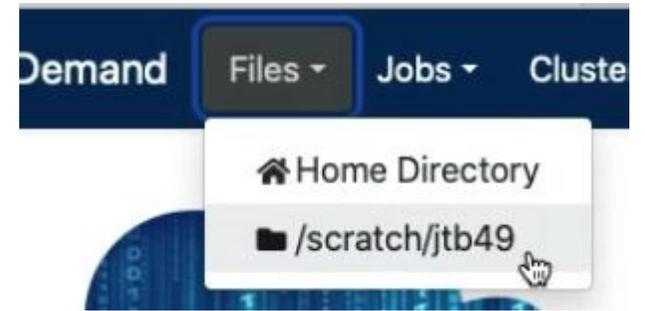
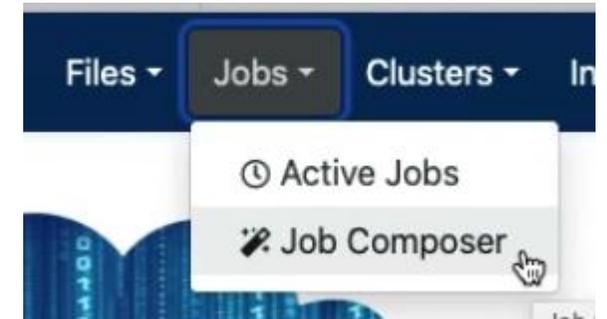
- Job Name: 2samplertest
- Submit to: Monsoon Cluster
- Account: Not specified
- Script location: /home/wew/ondemand/data/sys/myjobs/projects/default/8
- Script name: study1.sh

Exercise 1



Create a simple job and run it on the compute nodes

- From Ondemand, click the **Jobs > Job Composer** menu
- Click on **New Job** and select **From Default Template**
- From the *bottom of the right-column*, click the blue **Open Editor** button
- Change all “**NAUID**” to be *your* nau user-ID, e.g.: **abc123**!
- Set your job-name to: “*simple*”
- Set your output location to `/scratch/<NAUID>/exercise1.txt`
- Make your jobscript load the module named “*workshop*”
- Make sure it runs the “*date*” command (i.e. “`srun date`”)
- Finally, have it run the “*exercise1*” command, as well
- Save (in this tab), and then submit your job via the composer (previous tab)
- Use the File Explorer to examine your output (**Files > /scratch/NAUID**)
- Make a note of the secret code written to `exercise1.txt`



[Exercise 1 \(CLI\)](#)

Exercise 2



- Create a new job using **New Job > From Template**
- Select the 'Long Job' template, optionally rename it, and click **Create New Job**
- From the Jobs list, select the new draft job, and click **Open Editor** as before
- Change all "NAUID" to be your nau user-ID, again
- Make your jobscript load the module named "workshop"
- Make your jobscript run the "exercise2" command
 - i.e. "srun exercise2"
- Make your job sleep for 5 minutes
 - i.e. "srun sleep 300" (Sleep is a command that does nothing for N seconds)
- Save your jobscript, and submit this job
- Monitor your job by selecting **Jobs > Active Jobs** from your Dashboard
- When it has completed, examine the output in long.txt
- Make a note of the secret code from long.txt

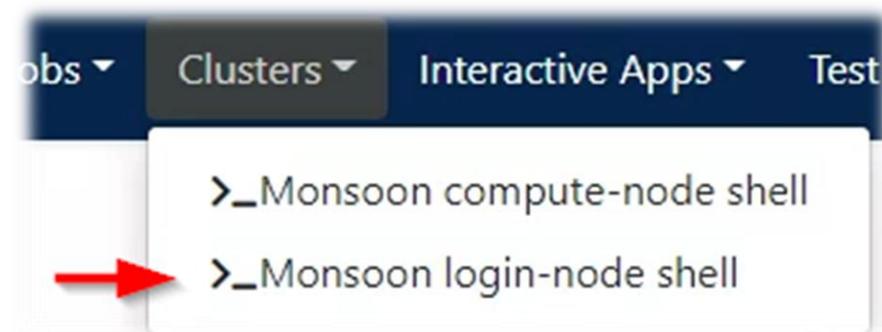
[Exercise 2 \(CLI\)](#)

Command-line access



Once you have the basics down using Ondemand, then the power of the cluster is exposed through the command-line interface (CLI)

- We will be utilizing a CLI built-in to Ondemand
- Feel free to tryout the commands we discuss
- Tip: The Monsoon CLI may also be accessed outside of Ondemand using any other “ssh client” (ex: Putty on Windows, or via the ssh command in your PowerShell or Terminal app)
- Now open a **login-node shell** (from the “Clusters” menu) and follow along!



The Ondemand CLI



You may access the Ondemand's CLI from the dashboard by selecting **Clusters > Monsoon login-node shell**

Note: Linux does NOT give interactive feedback when you enter passwords!* But it will evaluate your password attempt upon hitting enter!

```
ricky@wind:~
Last login: Tue Sep 16 09:31:24 MST 2025 on pts/24
#####
#
# Welcome to Monsoon - login node: [wind]
#
# Information:
# - Monsoon now running Enterprise Linux 8
# - /scratch : files auto DELETED after 30 days
#
# Issues or questions: ask-arc@nau.edu
#
# Next Maintenance:
# - Winter break
#
#####
[ricky@wind ~ ]$
```

Cluster info



- `sinfo`
 - view state information about SLURM nodes and partitions.
- `sinfo -N -l`
 - view state and specs on all individual nodes
- `sinfo -R`
 - List reasons for downed nodes and partitions

```
[ricky@wind ~ ]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
core*      up 14-00:00:0    4  drain cn[35,40,52-53]
core*      up 14-00:00:0   79   mix  cn[1,4-19,28-30,38-39]
core*      up 14-00:00:0   10  alloc cn[20-26,36-37,107]
core*      up 14-00:00:0    3   idle cn[3,32-33]
core*      up 14-00:00:0    4   down cn[31,42,51,65]
gpu        up 14-00:00:0    1   mix  cn1
gpu        up 14-00:00:0    3   idle cn[3,32-33]
gpu        up 14-00:00:0    1   down cn31
```

More info:

<https://slurm.schedmd.com/sinfo.html>

Interactive / Debug Work



- Run your compiles and testing on the cluster nodes by:
 - srun gcc hello.c -o a.out
 - srun **--qos=debug -c12** make -j12
 - srun Rscript analysis.r
 - srun python analysis.py
 - Try this now:
 - srun hostname
 - hostname

```
[ricky@wind ~ ]$ srun -c 4 --mem=9000 hostname
cn37
[ricky@wind ~ ]$ jobstats
JobID           JobName      ReqMem      MaxRSS      ReqCPUS
=====
22695207        hostname     8.79G       4.25M       4
=====

Memory       : 00.05%
CPU          : 01.18%
Time Limit  : 00.01%
=====
Efficiency Score: 0.41
=====
```

More info:

<https://slurm.schedmd.com/srun.html>

Long Interactive work



- `salloc`
 - Obtain a SLURM job allocation that you can work interactively with for an extended amount of time
 - This is useful for testing/debugging for an extended amount of time
 - For when you need more than a single `srun` but aren't ready for `sbatch`

```
[ricky@wind ~ ]$ salloc -c 8 --time=2-00:00:00
salloc: Granted job allocation 33442
[ricky@wind ~ ]$ srun python analysis.py
[ricky@wind ~ ]$ exit
salloc: Relinquishing job allocation 33442
```

```
[ricky@wind ~ ]$ salloc -N 2
salloc: Granted job allocation 33443
[ricky@wind ~ ]$ srun hostname
cn3
cn2
[ricky@wind ~ ]$ exit
salloc: Relinquishing job allocation 33443
```

More info:

<https://slurm.schedmd.com/salloc.html>

Submitting non-interactive jobs



The sbatch command is used to submit batch jobs to the slurm workload manager. Jobs submitted with sbatch are placed in a queue where they wait for resources to become available.

```
[ricky@wind ~ ]$ sbatch jobscript.sh
```

Submitted batch job 6880341

- slurm returns a job id for your job that you can use to monitor or modify constraints

More info:

<https://slurm.schedmd.com/sbatch.html>

Monitoring your job



- **jobstats**
 - Your main tool
 - Show usage stats for slurm jobs
 - combines features of scontrol and sacct tools
- `jobstats -h`
 - Usage instructions
- `jobstats -r`
 - Also show **running jobs**
- `jobstats -j <jobid>`
- `queue`
 - View information about jobs located in the SLURM scheduling queue.
- `queue --start`
- `queue -u <nauid>`
- `queue -o "%j %u ..."`
- `queue -p partitionname`
- `queue -S sortfield`
- `queue -t <state> (PD or R)`

More info:

<https://slurm.schedmd.com/queue.html>

Monitoring your job



- `sstat`
 - Display various statistics and information of a running job
 - Only works with jobs where analysis is executed with “`srun`”
- `sstat -j <jobid>`
- `sstat -o "AveCPU,AveRSS"`
- `sprio`
 - View the factors that comprise jobs’ scheduling priority
 - Shows only pending by default
- `sprio -l`
- `sprio -o "%j %u ..."`
- `sprio -w`

More info:

<https://slurm.schedmd.com/sstat.html>

<https://slurm.schedmd.com/sprio.html>

Controlling your job



- scancel
 - Used to signal jobs or job steps that are under the control of Slurm.
- scancel <jobid>
- scancel -n <jobname>
- scancel -u <nauid>
- scancel -t pending (only yours)

More info:

<https://slurm.schedmd.com/scancel.html>

Controlling your job



- `scontrol`
 - Used to view and modify Slurm configuration and state
 - Can change job constraints while it's in the pending state, but once the job starts, it can no longer be modified
- `scontrol show job 6880341`
 - When scheduled to start
 - Path to jobscript
 - Requested resources
 - Target node
- `scontrol update jobid=6880341 timelimit=4:00:00`

More info:

<https://slurm.schedmd.com/scontrol.html>

Job Accounting



- `sacct`
 - displays accounting data for of your jobs and job steps in the SLURM job accounting log or SLURM database
- `sacct -j <jobid> -o "jobid,elapsed,maxrss"`
- `sacct -N <nodelist>`
- `sacct -u <nauid>`
- Try our `sacct` wrapper "**jobstats**"
 - `jobstats -r`
 - `jobstats -j <jobid>`

More info:

<https://slurm.schedmd.com/sacct.html>

Job Accounting



- `sshare`
 - Tool for listing the shares of associations to a cluster.
- `sshare -l` : view and compare your groups cpu minutes usage
- `sshare -a` : view all users fairshare
- `sshare -a -A <account>` : view all members in your account (group)
- `group_efficiency <account>`

More info:

<https://slurm.schedmd.com/sshare.html>

Limits on the account (group)



- Limits are in place to prevent intentional or unintentional misuse of resources to ensure quick and fair turn around times on jobs for everyone.
- Groups are limited to a total number of *resource-minutes* in use at one time: 5M CPU-minutes, 64K GPU-minutes*
- This resource limit mechanism is referred to as: “TRESRunMins”.
- This limiting mechanism has nothing to do with priority!

Helpful Linux Commands



List Files	ls With "-l" option ("ls -l") will show more information
Change Directory	cd <directory path> (e.g.: "cd /scratch/NAUID/") cd by itself will return you to your home directory
Show/print current working directory	pwd
Copy Files	cp <source> <destination> use a period (".") as the destination to copy to your current directory
Move or rename a file	mv <source> <destination>
Delete a file	rm <filename>
Create a directory	mkdir <directory name>
View contents of a file	more <filename> less <filename> cat <filename>
Edit a file	nano <filename> (save=ctrl-o & exit=ctrl-x)
Exit your terminal session (log off)	exit

Exercise 3 via CLI



Get to know monsoon and Slurm, on your own. Start by opening a shell to Monsoon.

- How many nodes make up monsoon?
 - Hint: use “sinfo”
- How many nodes are in the gpu partition?
- How many jobs are currently in the running state ?
 - Hint: use “squeue -t R”
- How many jobs are currently in the pending state? Why?
 - Hint: use “squeue -t PD”

Exercise 4 via CLI



- Copy this job script to your scratch area:
 - `/common/contrib/examples/job_scripts/lazyjob.sh`
- Edit that copy: change any “NAUID” to be your id
- **Save the job script**
- Submit the job (`sbatch lazyjob.sh`), it will take 65 sec to complete
- Use `sstat` and `squeue` to monitor the job
 - `sstat -j <jobid>`, and `squeue -u <nauid>`
- Review the resources that the job used
 - `jobstats -r`
- We are looking for “MaxRSS”: *MaxRSS is the max amount of memory used*
- Edit the job scripts memory request (“`--mem`”): reduce the number of MB of memory being requested, eg: `--mem=600`; and resubmit
- Use `jobstats` to review the resources that the optimized job utilized, once again
- Ok, memory looks good, but notice that the `usercpu` is the same as the elapsed time
 - $$Usercpu = num\ utilized\ cpus * elapsed\ time$$
- This is because the application we were running only used 1 of the 4 cpus that we requested
- Edit the lazy job script: comment out first `srun` command, uncomment the second `srun` command; and resubmit
- Rerun `jobstats -r`, notice now `usercpu` is multiple times the elapsed time, in this case (4). Because we were allocated 4 cpus, and **used** 4 cpus.
- Now address the egregious time estimate!
- Make a note of the secret code from `lazy.txt`!

Archived Job scripts



Every job script that is submitted to slurm on monsoon is archived for three reasons:

1. Convenience – if you forget what script was used for what job, you can find out!
2. Support assistance – we can find the job script that was used in your job to help troubleshoot with you.
3. Security / stability – in case of any security or stability issues, we can connect issues and outages to associated jobs

Retrieval of a job script



- Archived job scripts, and their environment are stored here:
 - /common/jobscript_archive/<NAUID>/<year>/<month>
 - <job id>.sh – job script
 - <job id>.env – job script’s environment
 - Only the individual researcher and our support group can access their job scripts
- Example:
 - User abc123, accessing job id 2600 from March, 2021
 - `cat /common/jobscript_archive/abc123/2021/03/2600*.sh`
 - `cp /common/jobscript_archive/abc123/2021/03/2600*.sh ~/`
- Use “showscript” to make it easy!!!!

Showscript Demo



```
[ricky@wind ~ ]$ jobstats
JobID          JobName      ReqMem      MaxRSS      ReqCPUS      UserCPU      Timelimit    Elapsed      State        JobEff
=====
22696318       simple       1.95G       17.0M       1            00:00.372    00:20:00     00:00:03     COMPLETED   0.55
22696443       long         9.77G       17.6M       1            00:00.374    03:00:00     00:05:02     COMPLETED   1.49
=====
Memory        : 00.29%
CPU           : 100.00%
Time Limit    : 02.54%
=====
Efficiency Score: 34.28
=====
[ricky@wind ~ ]$ showscript 22696318
/common/jobscript_archive/ricky/2025/09/22696318.sh
=====
#!/bin/bash
#SBATCH --job-name=simple           # the name of your job
#SBATCH --output=/scratch/ricky/exercise1.txt # this is the file your output and errors go to
#SBATCH --chdir=/scratch/ricky      # your work directory
#SBATCH --time=20:00                # (max time) 20 min (shorter time=quicker start)
#SBATCH --mem=2000                   # (total mem) 2GB of memory
#SBATCH --mail-type=FAIL             # email notification if it fails

module load workshop
srun date
srun exercise1
=====
Job Efficiency - OVERALL: 33.7%, CPU: 100.00%, Mem: 00.85%, TIME: 00.25%
```

Checking your disk usage



- You can use the “**getquotas**” command to examine how much space you are using in the various monsoon storage areas

```
$ getquotas
Filesystem      #Bytes  Quota  %   | #Files  Quota  %
/home           23592M  30000M 78% | -        -      -
/scratch       70.5G   36.4T  0%  | 31K      2.9M   1%
```

Changing Your Default Account



- All researchers have a default slurm account to track usage
- **See it now** by: “`sacctmgr show user name=<NAUID>`”
- Some researchers belong to multiple slurm accounts
- Example to override the default:
 - `#SBATCH --account=inf503-fall24`
 - `#SBATCH --account=prof_lastname`

Confirming Your Account



- This is a required step for your account to be fully enabled!
- After completing the exercises: one, two, and four, you will have three, 32 character alpha-numeric codes
- With the codes in hand, confirm your monsoon account with the commands:
 - module load workshop
 - confirm_user
- More information here:
 - <https://in.nau.edu/arc/obtaining-an-account/>

Optimizing Your Cluster Use



- To get the most out of the cluster for yourself and your team, it is important to optimize the settings for your jobs.
- Optimization includes memory requested, time for the job to run, number of cpus

Slurm Arrays!



`$SLURM_ARRAY_JOB_ID (%A)` : the (parent) job-ID
`$SLURM_ARRAY_TASK_ID (%a)` : the ID of (child) job array member n

Slurm Arrays Exercise



From your scratch directory: (“cd /scratch/NAUID”)

- `tar xvf /common/contrib/examples/bigdata_example.tar`
- `cd bigdata`
- edit the file “job_array.sh” so that it works with your nau id, replacing all “NAUID” with your own
- Submit the script “`sbatch job_array.sh`”
- Run `squeue` and notice you have 5 jobs running, how did that happen!

Keep these tips in mind



- Know the software you are running, is it multi-threaded?
- Request resources accurately
- Supply an accurate time limit for your job
- Don't be lazy, it will affect you and your group negatively

Common Questions



- Should I use OnDemand or the command line?
 - Power users will tend to use command line
 - However, the terminal in ondemand is worth using all the time

Workshops, Office-, and Coffee-hours!



NAU NORTHERN ARIZONA UNIVERSITY
Advanced Research Computing

About ▾ Services ▲ Resources ▾ Research ▾

- Pricing
- Coffee/Office Hours**
- Service Requests
- Request an Account

Office Hours and Office Hours

Hours

eting on Zoom. For anyone
he expertise. Topics that we'll
hods for improving job

Schedule

Date	Time	Location
Wednesday 09/04/2024	2:00 PM - 4:00 PM	B54 ITS Room of Requirement (Room 106) AND Online via Zoom
Tuesday 10/01/2024	2:00 PM - 4:00 PM	B54 ITS Room of Requirement (Room 106) AND Online via Zoom

NAU NORTHERN ARIZONA UNIVERSITY
Advanced Research Computing

About ▾ Services ▾ Resources ▲ Research ▾ Coll ▾

- Documentation »
- Workshops**
- Web Apps

the basics of
ded to you
at you can
n when
s well as links
s as a valid

Helpful Tip: [Linux on the command-line](#) is a great beginner friendly resource for those who are just starting with Monsoon!

Workshop Dates

Workshop	Date	Time	Location
Intro to Monsoon	Thursday, September 5, 2024	2:00 PM - 4:00 PM	ITS (Building 54), Room 106
Intro to Monsoon (In Depth)	Thursday, September 26, 2024	2:00 PM - 4:00 PM	ITS (Building 54), Room 106
Linux in LDC	Thursday, October 3, 2024	2:00 PM - 4:00 PM	ITS (Building 54)

Question and Answer



- More info here:
 - <http://in.nau.edu/arc>
 - ask-arc@nau.edu
- Job efficiency
 - <http://metrics.hpc.nau.edu>
- FREE – Linux command line book:
 - <http://linuxcommand.org/tlcl.php>
 - Info here: <https://in.nau.edu/arc/external-resources/linux-resources/>
- And on the nauhpc listserv
 - nauhpc@lists.nau.edu

Exercise 1 (CLI)



Create a simple job and run it on the compute nodes

- `cp /common/contrib/examples/job_scripts/exercise1.sh ~/`
- `nano exercise1.sh` (or another editor)
- Change all “**NAUID**” to be *your* nau user-ID, e.g.: **abc123**!
- Set the name of your job (`--job-name`) to “simple”
- Set your `--output` path to be `/scratch/<NAUID>/exercise1.txt`
- Make your jobscript load the module named “workshop”
- Make your jobscript run the “date” command (i.e. “`srun date`”)
- Finally, have it run the “`exercise1`” command, as well
- Save the file (for nano: `ctrl-x`, and `y(es)`)
- Submit the batch script to slurm: `sbatch exercise1.sh`
- Make a note of the secret code written to `exercise1.txt`

[Next Slide: Exercise 2 \(CLI\)](#)

Exercise 2 (CLI)



- `cp /common/contrib/examples/job_scripts/exercise2.sh ~/`
- `nano exercise2.sh` (or another editor)
- Change all “**NAUID**” to be *your* nau user-ID, e.g.: **abc123!**
- Set your `--output` path to be `/scratch/<NAUID>/long.txt`
- Make your jobscript load the module named “workshop”
- Make, have it run the “`exercise2`” command (i.e. “`srun exercise2`”)
- Finally, make your job sleep for 5 minutes
 - i.e. “`srun sleep 300`” (Sleep is a command that does nothing for N seconds)
- Save the file (for nano: ctrl-x, and y(es))
- Submit the batch script to slurm: `sbatch exercise2.sh`
- When it has completed, examine the output in `long.txt`
- Make a note of the secret code from `long.txt`

[Next Slide: Command Line Access](#)