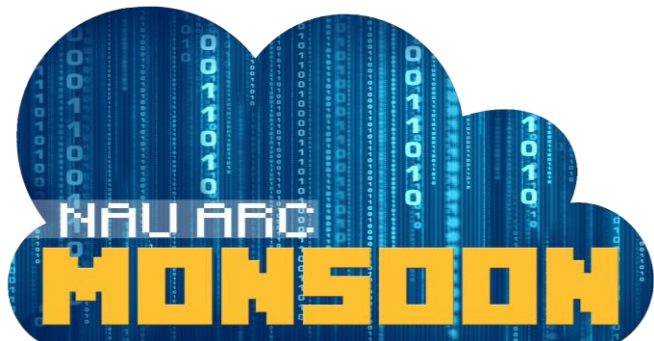


# Intro to Monsoon and Slurm

2024-10-28 presentation with Jason Buechler

These slides:

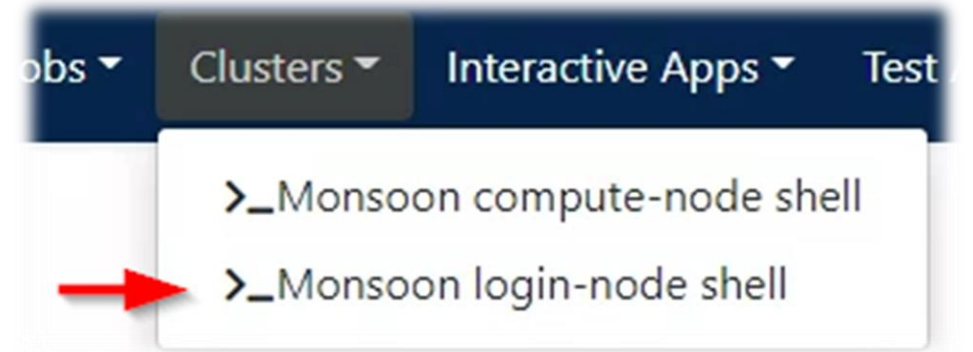
<https://rcdata.nau.edu/hpcpub/workshops/odintro.pdf>



# 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
- Click on Clusters tab
- Select 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 cluster experience?

Jason Buechler

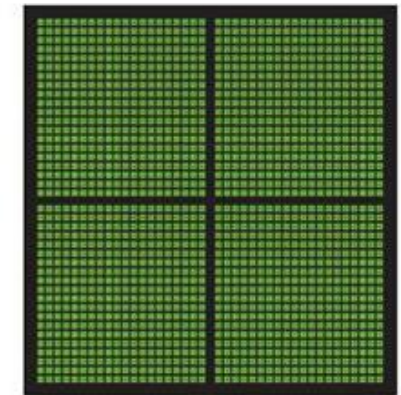
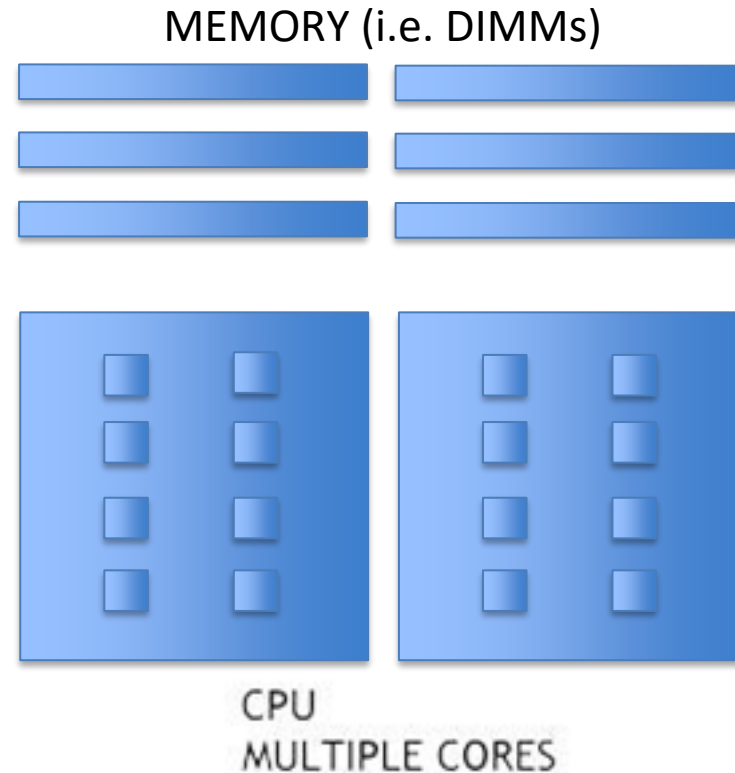
- Masters in Mech Engineering (fluidthermodynamics & renewable energy/grid engineering)
- Been with Monsoon for 86 years?
- Linux/supercomputing since 2000

# 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

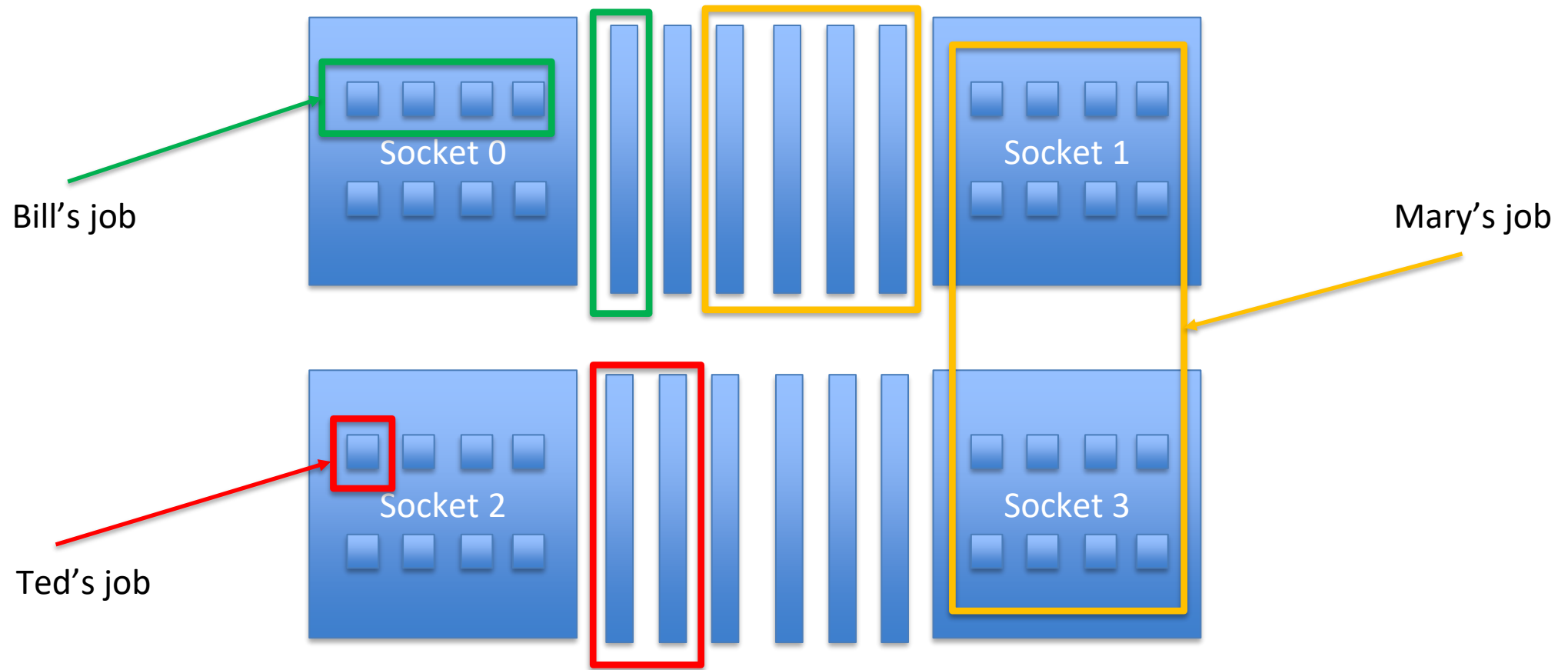
# Cluster (of) Resources

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

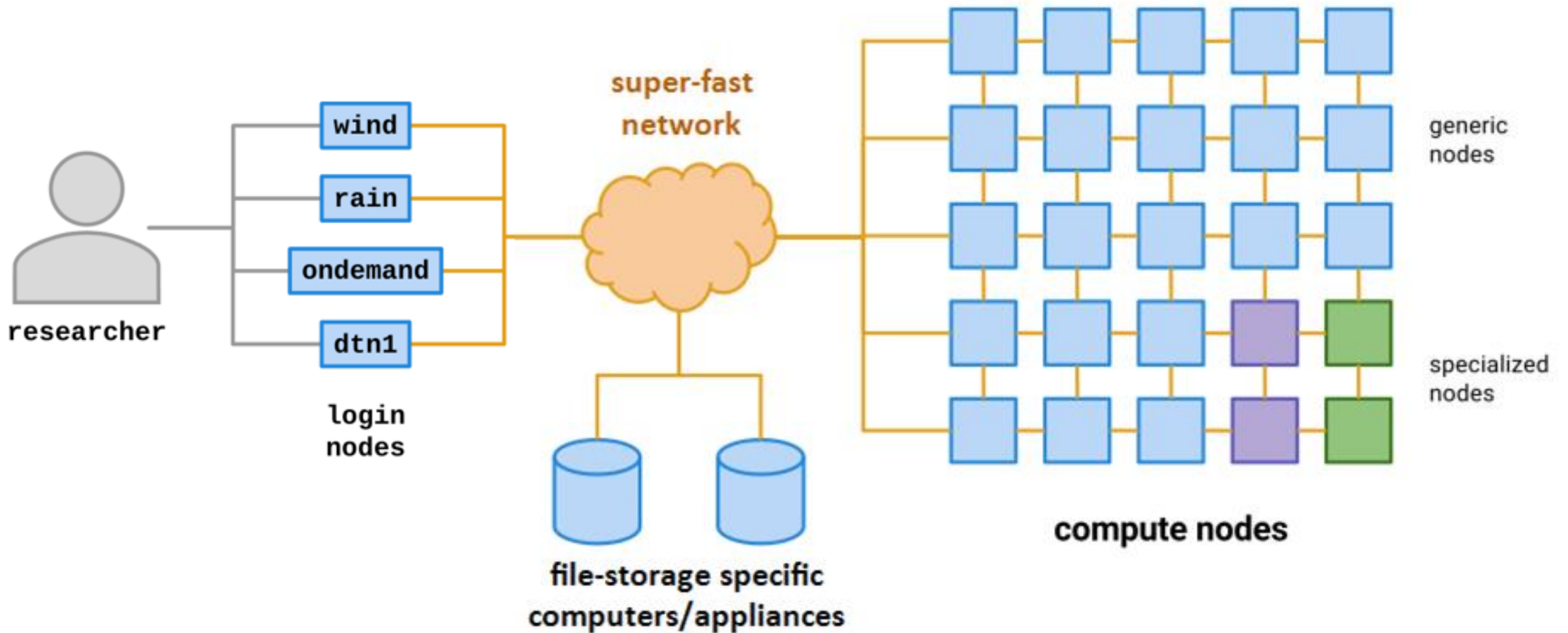


GPU  
THOUSANDS OF CORES

# Inside a (single) compute-node

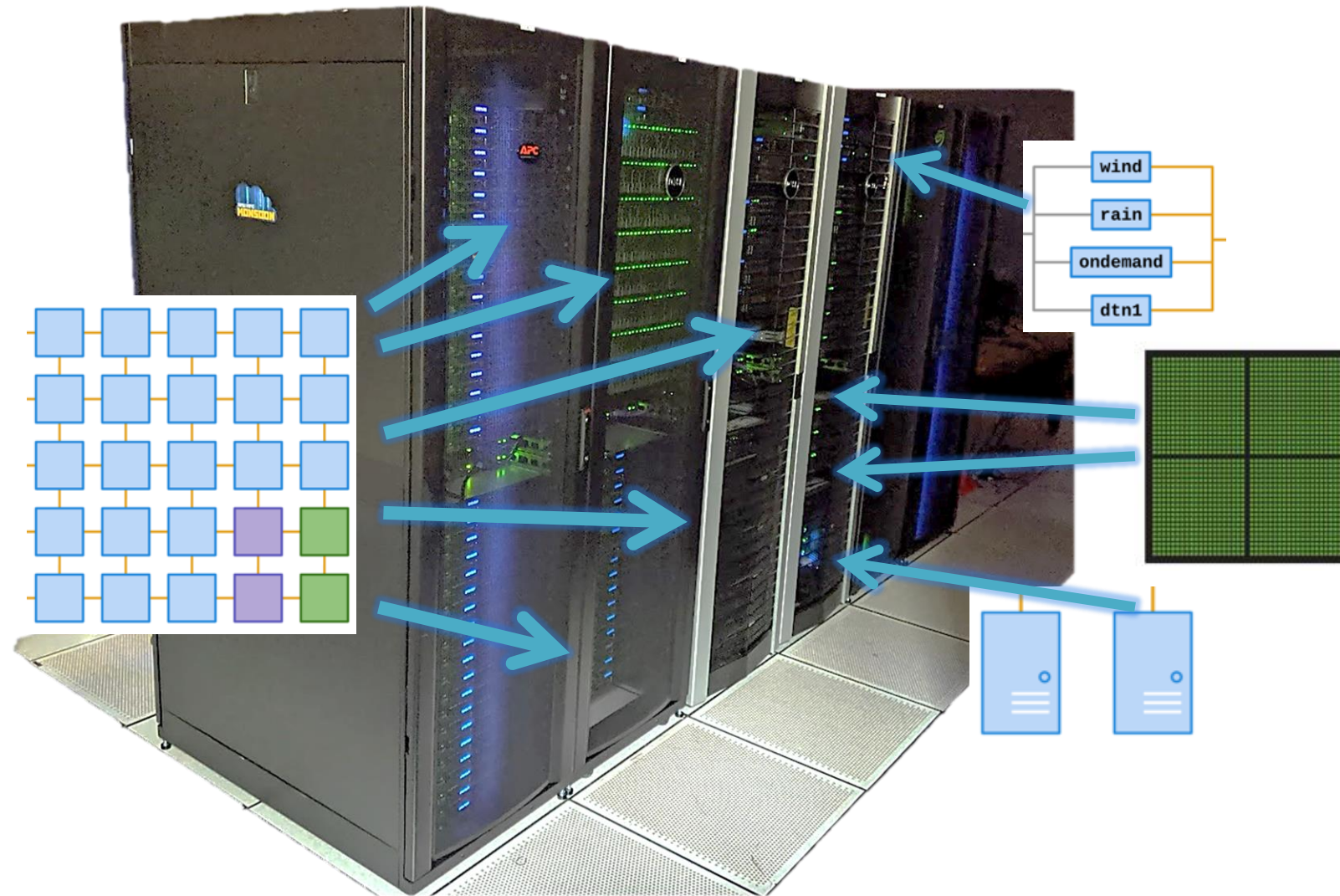


# Cluster = Login-nodes + Compute-nodes + etc





# Monsoon ( $\approx 4000$ cores)



# Largest Cluster!



8.7M cores

# Small Cluster!



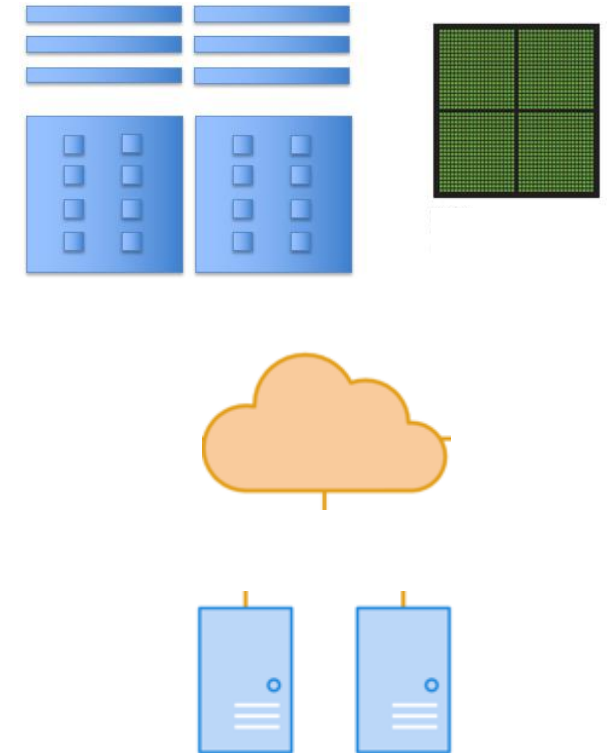
Dual core?



# Monsoon Today

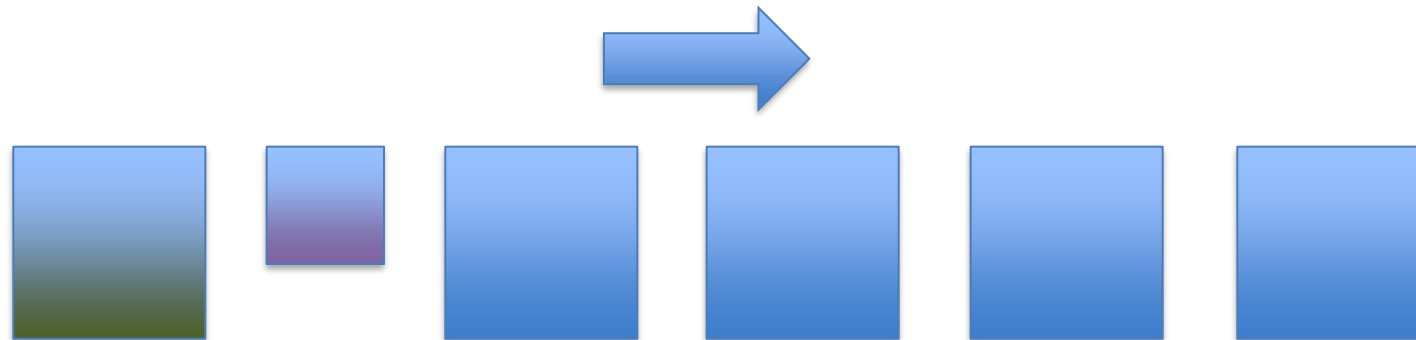
( summarized from <https://in.nau.edu/arc/details> )

- 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 8.9
- High speed interconnect: FDR, and HDR Infiniband
- Storage
  - 1PB *scratch* high-speed storage (Lustre)
  - 615TB *long-term* storage (ZFS)



# 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 far more advanced with dynamic priorities taking into consideration many factors

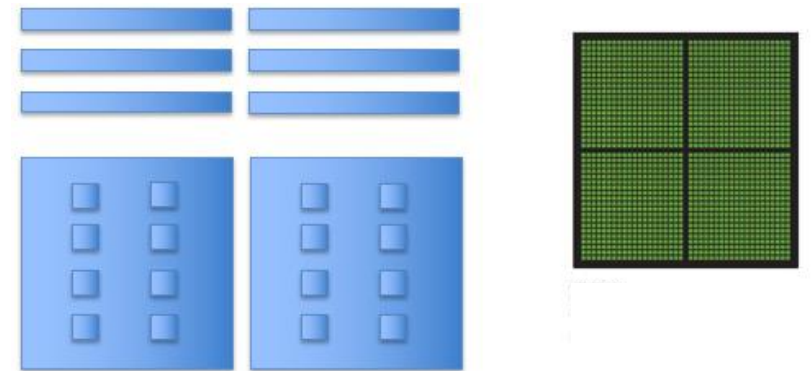


# What is scheduling?

- *“A plan or procedure with a goal of completing some objective within some time frame”*
  - Scheduling for a cluster at the basic level is much the same. Assigning work to computers to complete objectives within some time availability
  - Not exactly that easy though. Many factors come into play scheduling work on a cluster.
- A scheduler needs to know **what resources are available** on the cluster in order to make accurate scheduling decisions

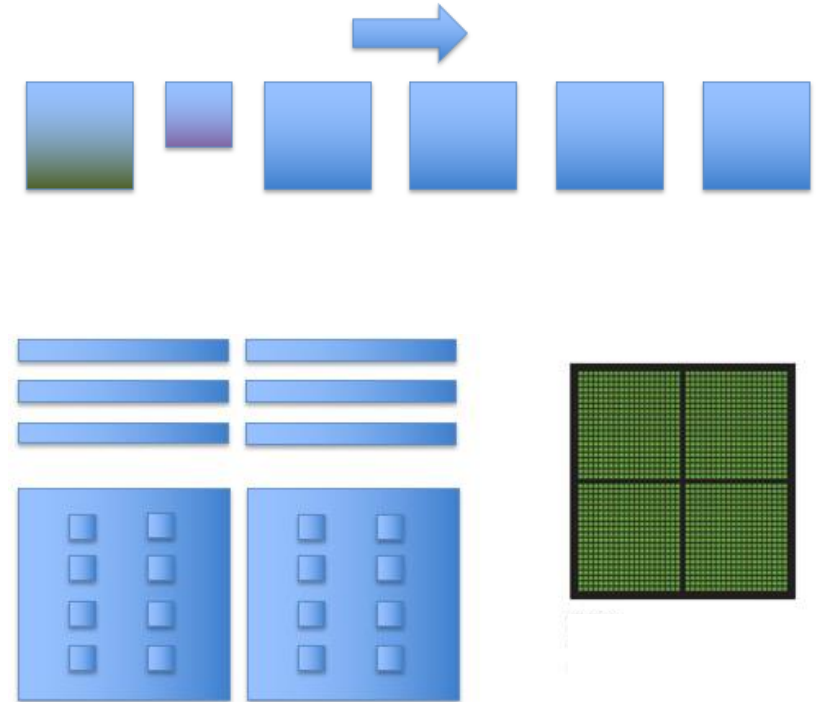
# Resource Manager

- 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



# Our Scheduling Goals

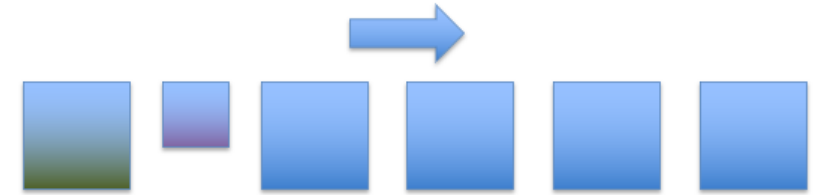
- Optimize quantity of work
- Optimize usage of resources
- Service all users and projects justly
- Make scheduling decisions transparent





# Many scheduling methods

- 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 a users recent resource consumption



# Monsoon scheduling

- Combination of scheduling methods
- Currently configured to utilize backfill along with a multifactor priority system to prioritize jobs

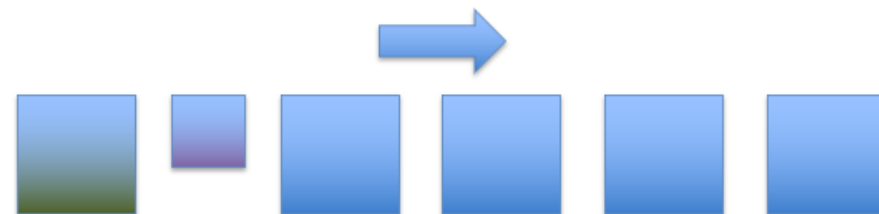


# Slurm ... yummm

- 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:
  - Frontier (#1), 8.7M cores, 1,102 PF, 21 kW - USA
  - Fugaku (#2), 7.6M cores, 537 PF, 30 kW - Japan

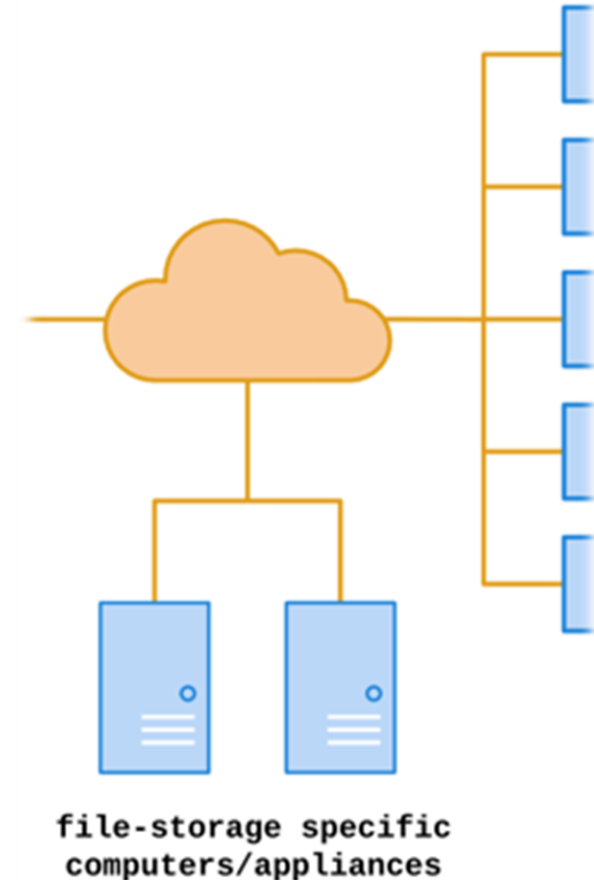
# Factors attributing to priority

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



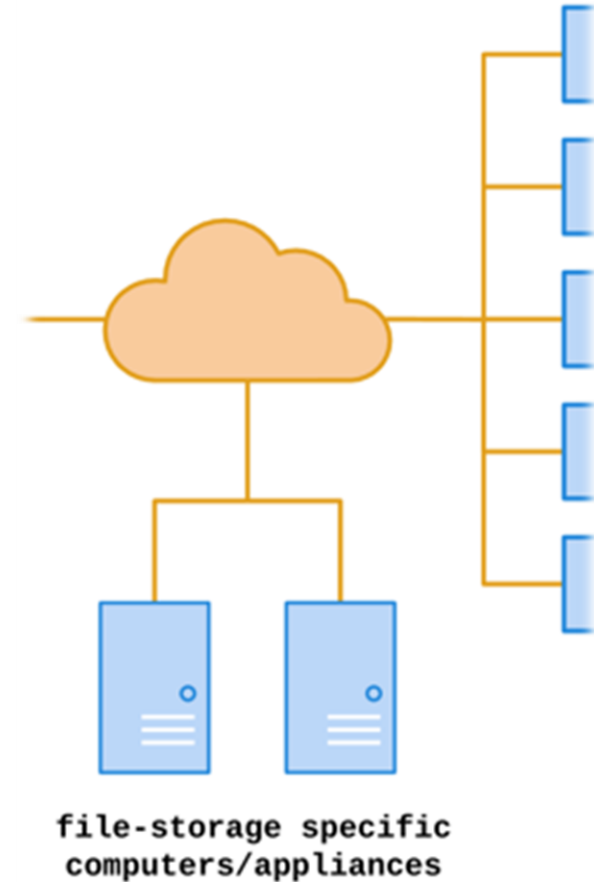
# Storage

- /home – **10 GB** quota
  - Keep your scripts and executables here
  - **Snapshotted twice a day**: /home/.snapshot
  - Please do not write job output (logs, results) here!!
  - Run the command “quota” now
- /scratch – **15 TB** 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 – 615TB
  - Long-term storage project shares
  - 5TB is assigned to faculty member for group to share
  - \$24/TB/year above 5TB
  - Snapshots available
  - Backups available - \$.10/GB/month
- /common
  - Cluster support share
  - Contrib: place to put software/libs/conf/db's for others use



# Data Workflow

1. Keep scripts and executables in /home (or on Ondemand)
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

- Via Ondemand in a web browser
  - Drag and drop files
- <https://in.nau.edu/arc/overview/file-management/>
  - scp command/protocol
    - `scp <files> <nauid>@dtn1.hpc.nau.edu:/scratch/<nauid>/`
    - GUI examples: WinSCP on windows, Fetch for mac
  - samba /smb / cifs / “windows file sharing” / “shared drive”
    - Windows: `\\shares.hpc.nau.edu\cirrus`
    - Mac: `smb://shares.hpc.nau.edu/cirrus`
- **Globus**
  - <https://nau.edu/high-performance-computing/globus/>



# Data transfer node

- We have a dedicated (login-node) system for transferring data
- This host's name is dtn1.hpc.nau.edu
- Use dtn1 for moving large datasets around on monsoon, and to/from the internet

# 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 you on the cluster if you'd like to manage access to your data
- <https://my-old.nau.edu>
  - “Open directory services”
  - “Enterprise groups”
  - Take a look at our FAQ :: <https://nau.edu/high-performance-computing/faqs/>
  - If they are not working for you, contact ITS Solution Center
- What groups are you in? Run the command “groups”, or “id”

# 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 list` ...*modules currently loaded*
- `module load <module name>` ...*load a package module*

# Software

- Matlab
- Mathematica
- R
- SAS
- Qiime2
- Anaconda Python
- Lots of bioinformatics programs
- Request additional software to be installed!

```
jtb49 — ricky@wind:~ — ssh jtb49@wind.hpc.nau.edu — 92x15
[ricky@wind ~ ]$ module -d av
----- /packages/modulefiles -----
R/4.1.2                geos/3.8.1            openmpi/4.1.4
amd-blis/3.0          globus/3.10.1        parallel-netcd
amd-libflame/3.0      gmes/4                picard/2.24.1
anaconda2/2019.10    go/1.17.5            prinseq-lite/0
anaconda3/2022.10    grass/7.8.2          proj/7.1.0
ansys/2022r2         gsl/2.6              qiime2/2023.2
aocc/2.2.0           guppy-cpu/5.0.11c   raxml/8.2.12
augustus/3.3.3       guppy/6.3.8         rclone/1.60.0
bamtools/2.5.2       intel/2021.1        repeatmasker/4
bcl2fastq2/2.20.0    ior/3.2.1            repeatmodeller/
beagle-lib/3.2.0     iq-tree/2.2.0.4     samtools/1.11
beast/1.10.4-dev     jags/4.3.0          sas/ts1m7
```

# Requesting Software

- You can install quite a bit of R, and python software yourself!
- For R
  - `module load R`
  - `R`
  - `install.package(c(package))`
- For python
  - `module load anaconda3/<ver>`
  - `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 here:
  - <https://in.nau.edu/high-performance-computing/request-software/>

# 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`

# 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. module loading
  3. srun job steps/statements for the actual work

# Interacting with Slurm

- What resources are needed?
  - 2 **cpus**, 12GB **memory**, for 2 **hours**?
- What steps are required?
  - Run prog1, then prog2 ... etc
  - Are the steps dependent on one another?
- Can your work, or project be broken into pieces?
  - Smaller pieces can make the workload more agile.
- How long should your job run for? (i.e.: *educated guess*)
- Is your software multithreaded, uses OpenMP or MPI?



# Example Job script

- `#!/bin/bash`
- `#SBATCH --job-name=test`
- `#SBATCH --output=/scratch/NAUID/output.txt` # the stdout from your job goes here
- `#SBATCH --time=20:00` # shorter time = sooner start
- `#SBATCH --chdir=/scratch/NAUID` # default location slurm searches

- *# replace this module with software-*
- *# modules required by your jobscript*
- `module load anaconda3/2021.11` # 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's editor)



```
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
```

# 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 --begin=now+1hour --begin=2010-01-20T12:34:00</code>
Add MPI tasks/ranks to your job	<code>--ntasks=2</code> , or <code>-n 2</code>
To control job failure options	<code>--norequeue --requeue</code>
To receive status email	<code>--mail-type=ALL</code>

# Constraints and Resources

You want	Switches needed
To choose a specific node feature (e.g. avx2)	--constraint=avx2
To use a generic resources (e.g. a gpu)	--gres=gpu:tesla:1, -G1
To reserve a whole node for yourself	--exclusive
To chose a partition	--partition

# *Interacting with Monsoon*

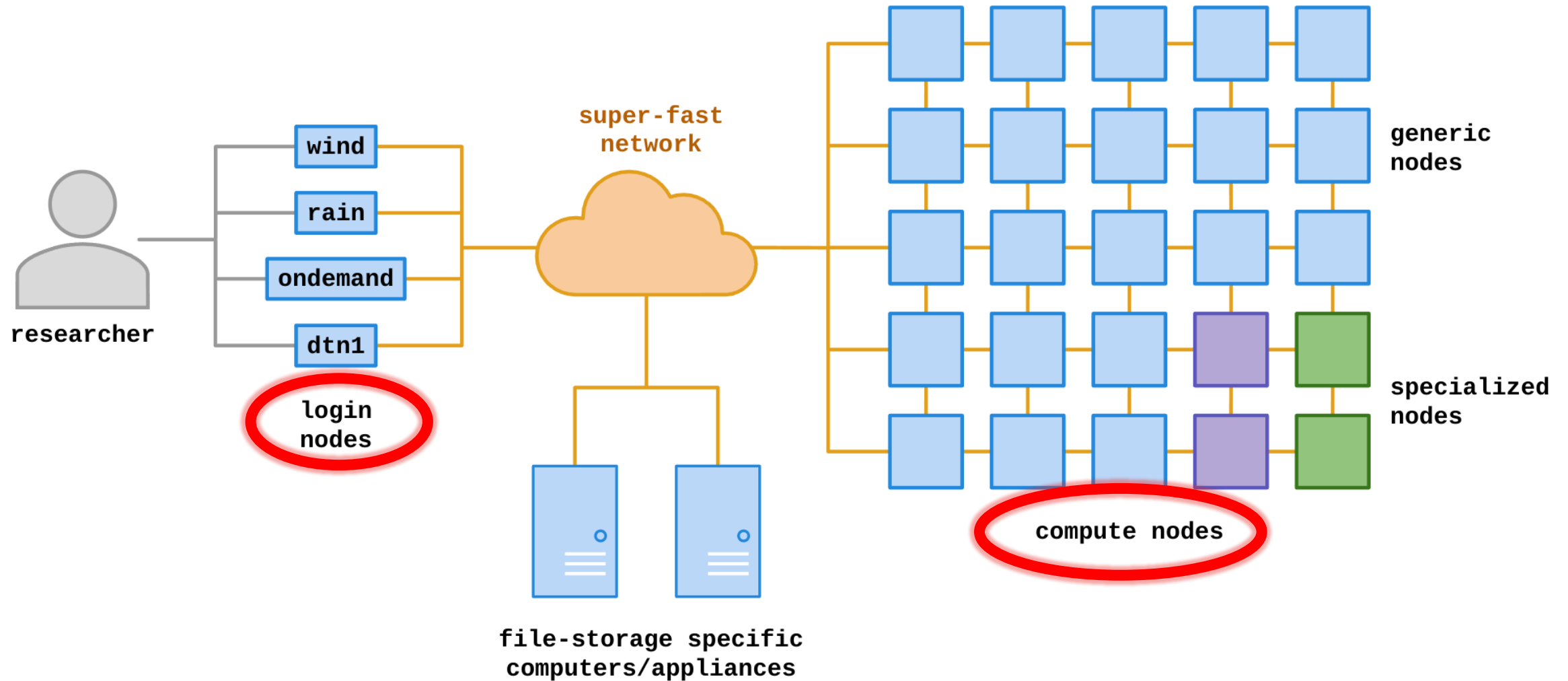
Three Methods (**must be on NAU Internet or NAUVPN**):

- Connect to OpenOndemand web interface at: <https://ondemand.hpc.nau.edu>
- Via SSH protocol in a command-line shell
  - Type “`ssh <nau-id>@<login-node>`”
    - ...within **Powershell** (Windows), or **Terminal** (Mac,\*nix)
  - login nodes:
    - monsoon.hpc.nau.edu (for research)
    - wind.hpc.nau.edu
    - ondemand.hpc.nau.edu
    - rain.hpc.nau.edu (for class work)
- SMB connection (files only -- no linux tools/commands)
  - <\\shares.hpc.nau.edu\cirrus> ...win/mac... <shares.hpc.nau.edu/cirrus>
  - see guide here: <https://in.nau.edu/arc/overview/file-management/>

# Login node vs Compute node

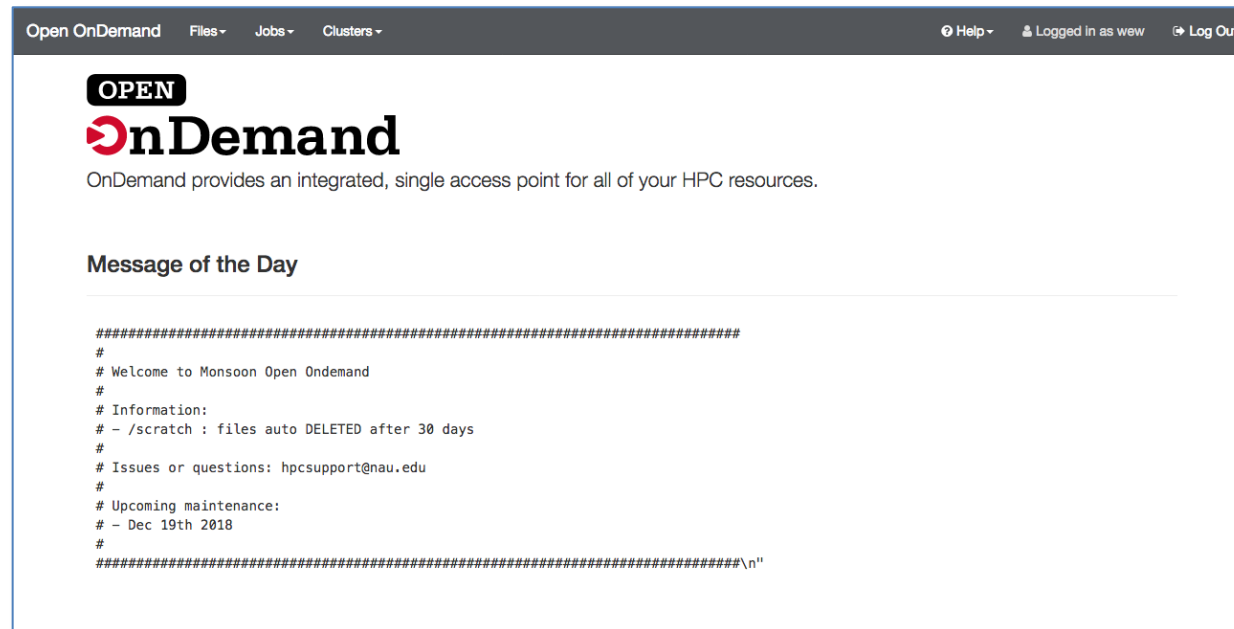
- When you log into “monsoon” interactively or via Ondemand you are “placed” on a login node.
- The login node is a shared system used solely for:
  - Developing scripts
  - Transferring small data
  - Submitting work to the scheduler
  - Analyzing results
  - Debug work less than 30 minutes in length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

# Cluster = Login-nodes + Compute-nodes + etc



# Ondemand

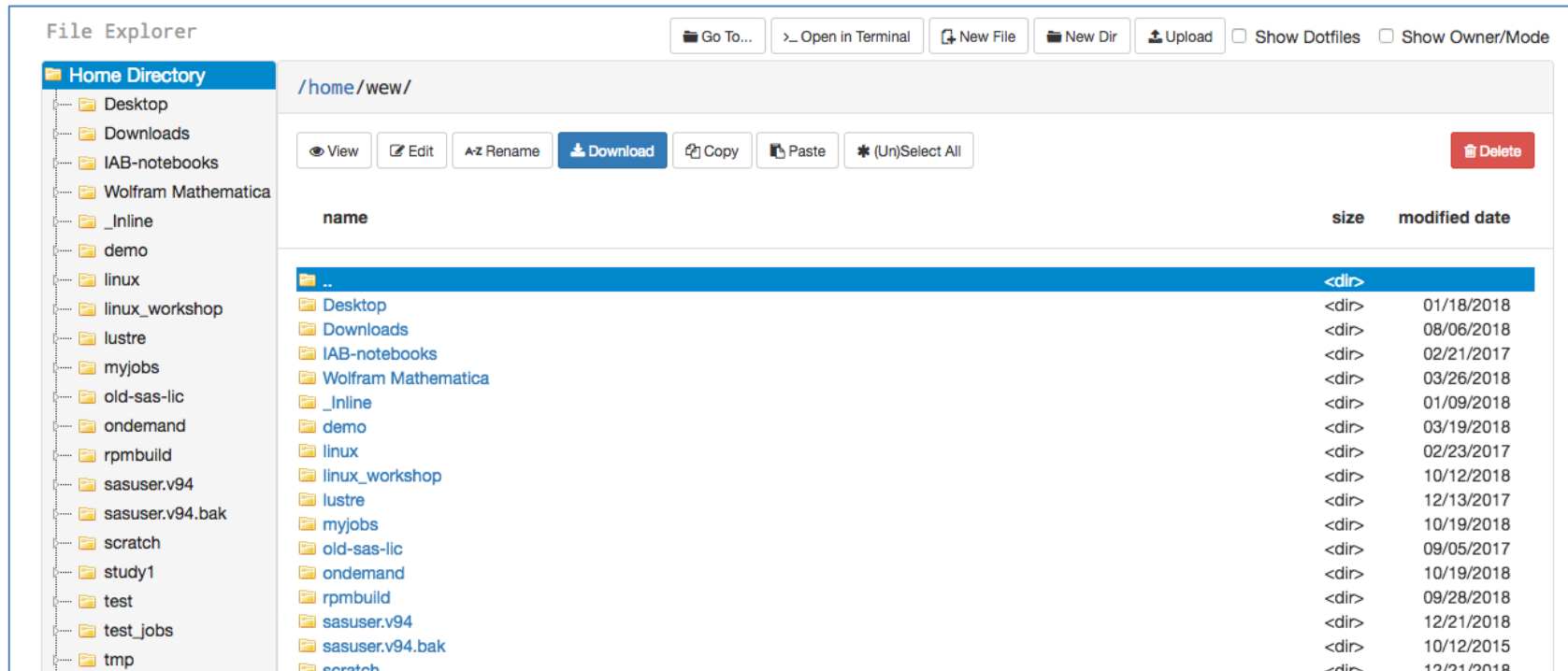
- Open Ondemand (OOD) is an interactive Graphical User Interface (gui) to the Cluster. You access it from your web browser at <https://ondemand.hpc.nau.edu>





# Ondemand File Explorer

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



# OnDemand Job Composer

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

The screenshot displays the OpenOnDemand Job Composer interface. The top navigation bar includes 'Open OnDemand / Job Composer', 'Jobs', 'Templates', and a 'Help' icon. The main section is titled 'Jobs' and features a '+ New Job' button and a 'Create Template' button. Below these are buttons for 'Edit Files', 'Job Options', 'Open Terminal', 'Submit', 'Stop', and 'Delete'. A search bar and a 'Show 25 entries' dropdown are also present. The main content is a table of jobs:

Created	Name	ID	Cluster	Status
December 21, 2018 11:12am	2sampletest	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' with 'Previous', '1', and 'Next' navigation buttons.

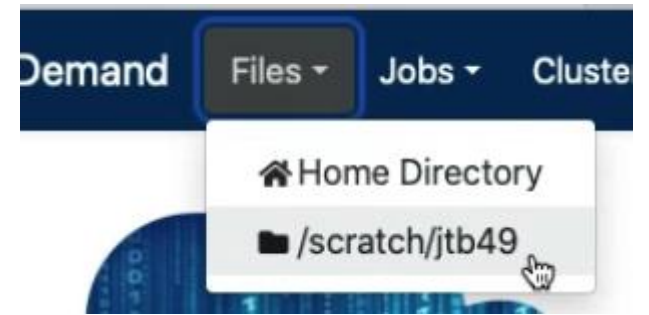
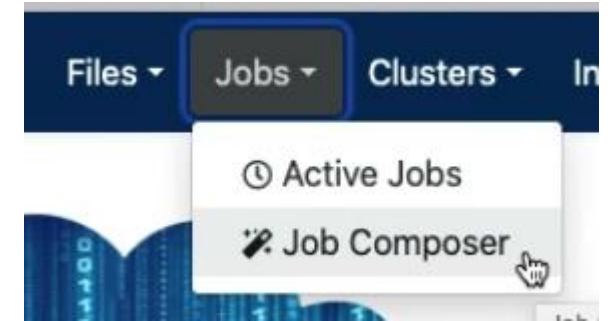
The right sidebar shows 'Job Details' for the selected job '2sampletest'. It includes fields for 'Job Name', 'Submit to:' (Monsoon Cluster), 'Account:' (Not specified), 'Script location:' (/home/wew/ondemand/data/sys/myjobs/projects/default/8), 'Script name:' (study1.sh), and 'Folder Contents:' (/study1.sh). Below this is the 'Submit Script' section, which shows the script contents for 'study1.sh':

```
#!/bin/bash
#SBATCH --job-name=test1 # the name c
#SBATCH --output=/scratch/wew/study1/output.txt # this is th
#SBATCH --time=2:00 # 2 min, sh
#SBATCH --workdir=/scratch/wew/study1 # your work
#SBATCH --mem=500
```

# Exercise 1

Create a simple job in the job composer from the template that you will then submit to the scheduler to run on the compute nodes.

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



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

# Exercise 2

- Create a new job using **New Job > From Specified Path**
- Source path: `/common/contrib/examples/job_scripts`
- Name: “longjob” (this is *your* name for your job)
- Script name: “`longjob.sh`” (this is *our* existing filename)
- For Cluster and Account: leave empty
- Save, select “longjob” from the Jobs list, and click Open Editor button as before
- Change all “NAUID” to be your nau ID
- Make your jobscript load the module named “`workshop`”
- Make your jobscript run the “`exercise2`” command
  - e.g. “`srun exercise2`”
- Make your job sleep for 5 minutes (sleep 300)
  - Sleep is a command that creates a lazy process that ... sleeps and does nothing
- Save, and then Submit
- Monitor your job by selecting Jobs and Active Jobs from your Dashboard.
- Examine the output in long.txt
- Make a note of the secret code from long.txt

# 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
- Follow along after opening a CLI (from “Clusters” menu)
  - Feel free to tryout the commands that we will be discussing
  - Tip: The Monsoon CLI may also be accessed outside of ondemand via an “ssh client” such as Putty on Windows or Terminal on the Mac.

# The Ondemand CLI

- You may access the CLI from the dashboard and selecting **Clusters > Monsoon login node shell**

```
#####  
#  
# Welcome to Monsoon Open Ondemand  
#  
# Information:  
# - /scratch : files auto DELETED after 30 days  
#  
# Issues or questions: hpcsupport@nau.edu  
#  
# Upcoming maintenance:  
# - None on schedule  
#  
#####\n"  
[wew@ondemand ~ ]$ █
```

**Note: When logging in, ssh does NOT give interactive feedback while you enter your password, but it will evaluate your password attempt upon hitting enter!**

# Cluster info

- `sinfo`
  - view information about SLURM nodes and partitions.
- `sinfo -N -l`
- `sinfo -R`
  - List reasons for downed nodes and partitions

# 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`



# Long Interactive work

- **salloc**
  - Obtain a SLURM job allocation that you can work with for an extended amount of time interactively.
  - This is useful for testing/debugging for an extended amount of time.

```
[user1@wind ~ ]$ salloc -c 8 --time=2-00:00:00
salloc: Granted job allocation 33442
[user1@wind ~ ]$ srun python analysis.py
[user1@wind ~ ]$ exit
salloc: Relinquishing job allocation 33442
```

```
[user1@wind ~ ]$ salloc -N 2
salloc: Granted job allocation 33443
[user1@wind ~ ]$ srun hostname
cn3
cn2
[user1@wind ~ ]$ exit
salloc: Relinquishing job allocation 33443
```

# 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.

```
[user1@wind ~ ]$ sbatch jobscript.sh
```

Submitted batch job 85223

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

# Monitoring your job

- **jobstats: your main tool**
  - `jobstats -r`
  - `jobstats -j <jobid>`
- `sprio`: view the factors that comprise a job's scheduling priority
  - `sprio -l`
    - list priority of users jobs in pending state
  - `sprio -o "%j %u ... "`
  - `sprio -w`

# Monitoring your job

- `queue`
  - view information about jobs located in the SLURM scheduling queue.
- `queue --start`
- `queue -u login`
- `queue -o "%j %u ... "`
- `queue -p partitionname`
- `queue -S sortfield`
- `queue -t <state> (PD or R)`

# 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 mylogin
- scancel -t pending (only yours)

# 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 mylogin`
  
- Try our `sacct` wrapper “`jobstats`”
  - `jobstats -r`
  - `jobstats -j <jobid>`

# 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>

# 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 cpu minutes in use at one time: 5M, and gpu minutes: 64K
- This resource limit mechanism is referred to as: “TRESRunMins”.
- This limiting mechanism has nothing to do with priority!



# Helpful Linux Commands

List Files	ls options -l – to show more information
Change Directory	cd <directory path> cd by itself will return you to your home directory
Show/print current working directory	pwd
Copy Files	cp <source> <destination> use a period for the destination to copy a file 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>
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.

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

# Exercise 4 via CLI

- Copy job script and edit:
  - `/common/contrib/examples/job_scripts/lazyjob.sh`
- Edit the job, change NAUID to be your id
- Save the job
- Submit the job (`sbatch lazyjob.sh`), it will take 65 sec to complete
- Use `sstat` and `queue` to monitor the job
  - `sstat -j <jobid>`, and `queue -u <userid>`
- 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, reduce the memory being requested in MB and resubmit, edit “`--mem=`”, e.g. `--mem=600`
- Review the resources that the optimized job utilized once again
  - `jobstats -r`
- Ok, memory looks good, but notice that the `usercpu` is the same as the elapsed time
  - $$\text{Usercpu} = \text{num utilized cpus} * \text{elapsed time}$$
- 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, and uncomment the second `srun` command.
- Resubmit
- Rerun `jobstats -r`, notice now `usercpu` is a 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/<user>/<year>/<month>
  - <job id>.sh – job script
  - <job id>.env – job scripts 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

# Checking your quotas

- From time to time you may need 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

1. Job script is created

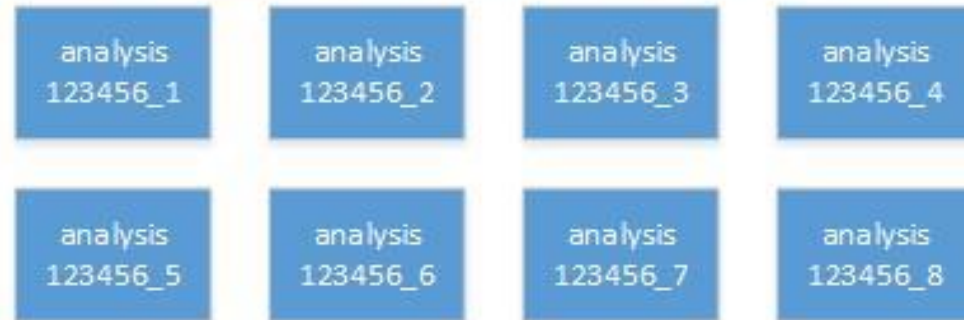
```
analysis
--array=1-8
```

2. Job script is submitted

```
analysis
--array=1-8
```



3. Job is launched with eight instances running in parallel



## Useful environment variables

- SLURM\_ARRAY\_JOB\_ID: the job array's ID (parent)
- SLURM\_ARRAY\_TASK\_ID: the id of the job array member n (child)

%A

%a

# Slurm Arrays Exercise

- From your scratch directory: “/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 yours
- Submit the script “`sbatch job_array.sh`”
- Run “`queue`”, notice there are 5 jobs running, how did that happen!

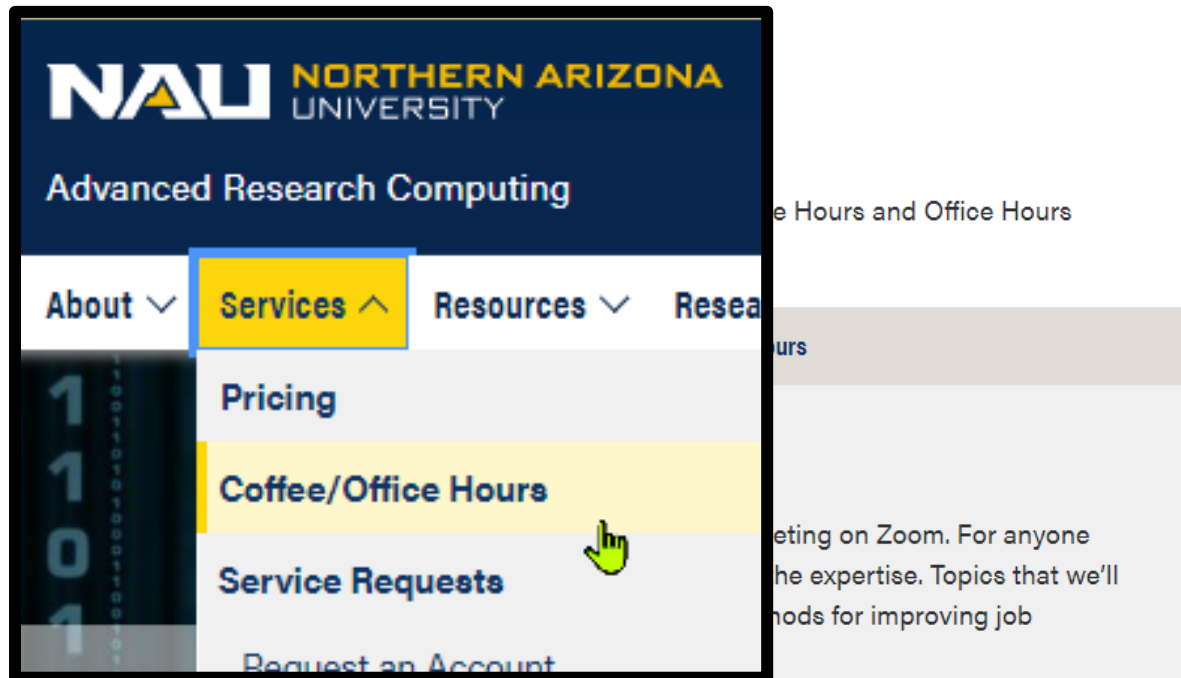
# Keep these tips in mind

- 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

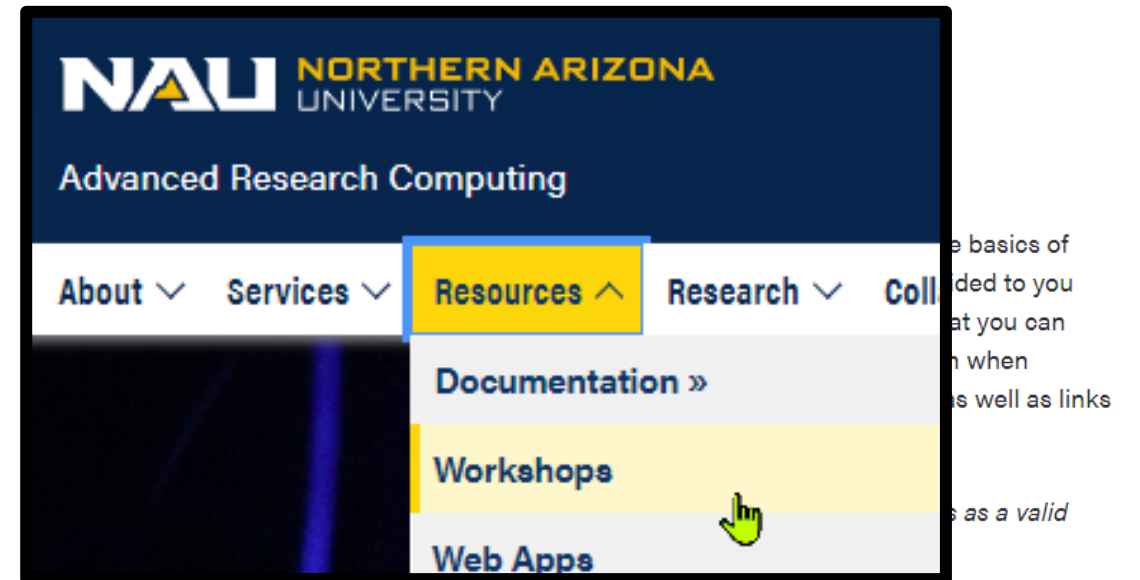
# Workshops, Office-, and Coffee-hours!



The screenshot shows the top navigation bar of the Northern Arizona University website. The 'Services' menu is expanded, and 'Coffee/Office Hours' is highlighted with a yellow background. A green hand cursor is pointing at the 'Coffee/Office Hours' link. Other menu items include 'About', 'Resources', and 'Research'. The page title is 'Advanced Research Computing'.

## Schedule

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



The screenshot shows the top navigation bar of the Northern Arizona University website. The 'Resources' menu is expanded, and 'Workshops' is highlighted with a yellow background. A green hand cursor is pointing at the 'Workshops' link. Other menu items include 'About', 'Services', 'Research', and 'Collaboration'. The page title is 'Advanced Research Computing'.

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 HPC	Thursday, October 3, 2024	2:00 PM - 4:00 PM	ITS (Building 54), Room 106

# 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](mailto:nauhpc@lists.nau.edu)



# Exercise 1 (CLI)

- `cp /common/contrib/examples/job_scripts/exercise1.sh ~/`
- `nano exercise1.sh` (or another editor)
- Replace all occurrences of “NAUID” to be your nau user-ID, e.g.: abc123!
- Name your job (`--job-name`): “exercise1”
- Set `--output` to be `/scratch/<NAUID>/exercise1.out`
  - Example: `--output=/scratch/abc123/exercise1.out`
- Make your jobscript load the module named “workshop”
  - Example: `module load workshop`
- Make your jobscript run the “date” command
  - Example: `srun date`
- Additionally, run the “exercise1” command, as well
  - Example: `srun example1`
- Save the file
  - For nano: `cntrl x`, and “yes”
- Submit the batch script to slurm
  - `sbatch exercise1.sh`
- Make a note of the secret code in `exercise1.out`

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

# Exercise 2 (CLI)

- `cp /common/contrib/examples/job_scripts/exercise2.sh ~/`
- `nano exercise2.sh`
- Replace all occurrences of NAUID with your nau ID, e.g. abc123
- Name your job output “exercise2.out”
  - Example: `--output=/scratch/abc123/exercise2.out`
- Make your jobscript load the module named “workshop”
  - Example: `module load workshop`
- Make your jobscript run the “exercise2” command
  - Example: `srun exercise2`
- Make your job sleep for 5 minutes (sleep 300)
  - Example: `srun sleep 300`
  - Sleep is a command that creates a lazy process that ... sleeps and does nothing
- Save the file
  - For nano: `cntrl x`, and “yes”
- Submit the batch script to Slurm
  - Example: `sbatch exercise2.sh`
- Monitor your job utilizing “`squeue -u <NAUID>`”
- Examine the output in `exercise2.out`
- Make a note of the secret code from `exercise2.out`

[Next Slide: Command-line Access](#)