Intro to Monsoon and Slurm

2024-03-18

Slides:
https://rcdata.nau.edu/hpcpub/workshops/odintro.pdf

Supplemental video:
https://rcdata.nau.edu/hpcpub/workshops/odintro.mp4

NAU NORTHERN ARIZONA UNIVERSITY

Get logged in!

- Slides here:
  - https://rcdata.nau.edu/hpcpub/workshops/odintro.pdf

- From a Computer:
  - Log into NAU VPN *if off-campus*!
    - Instructions here: https://in.nau.edu/its/remote-services/
    - VPN requires Two Factor Authentication
      - https://nau.service-now.com/kb_view.do?sysparm_article=KB0013321
  - Open a web browser
    - May need to search in start menu for it
  - Browse to http://ondemand.hpc.nau.edu
    - Log in with your louie id
  - Click on Clusters tab, and select Monsoon login-node shell

NAU NORTHERN ARIZONA UNIVERSITY

- HIT RECORD! ☺

Introductions

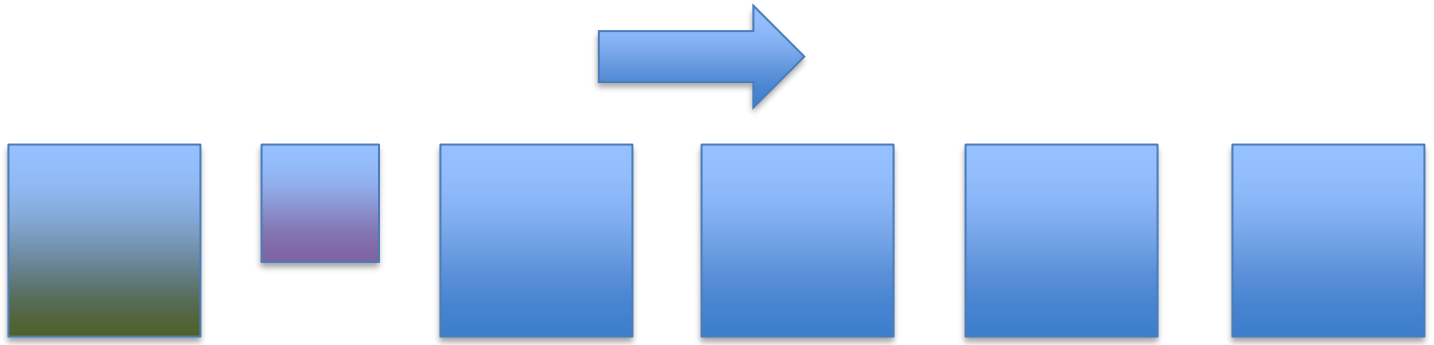
- Introduce yourself
  - Name
  - Department / Group
  - What project(s) do you plan to use monsoon for?
  - Linux or Unix experience
  - Previous cluster experience?

List of Topics

- Cluster education
  - What is a cluster, exactly?
  - Queues, scheduling and resource management
- Cluster Orientation
  - Monsoon cluster specifics
  - How do I use this cluster?
  - Group resource limits
  - Exercises
  - Question and answer

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.

Scheduling

- A scheduler needs to know what resources are available on the cluster in order to make accurate scheduling decisions
- Resource availability changes by the minute
- Assignment of work on a cluster is carried out most efficiently with the scheduler and resource manager working together

Resource Manager

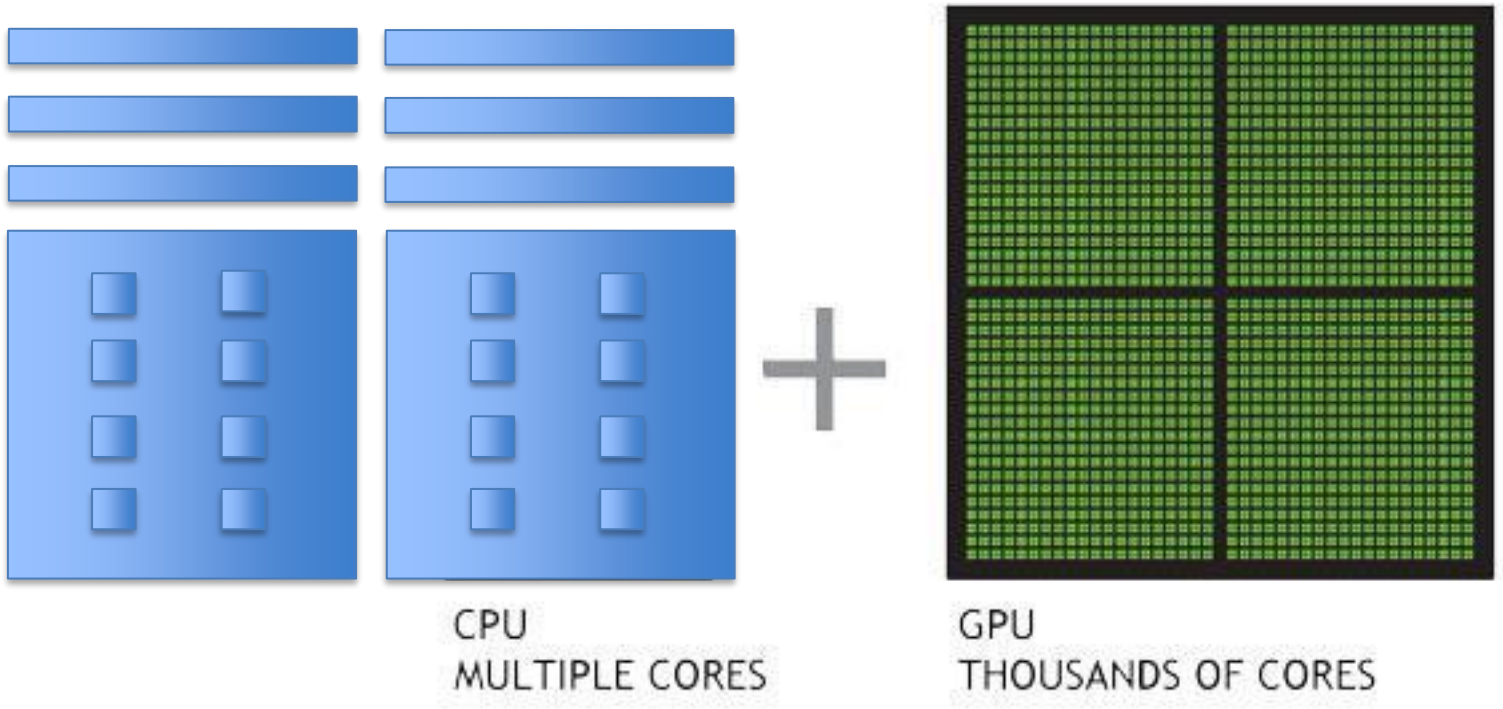
- Monitoring resource availability and health
- Allocation of resources
- Execution of resources
- Accounting of resources

Our Scheduling Goals

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

Cluster Resources

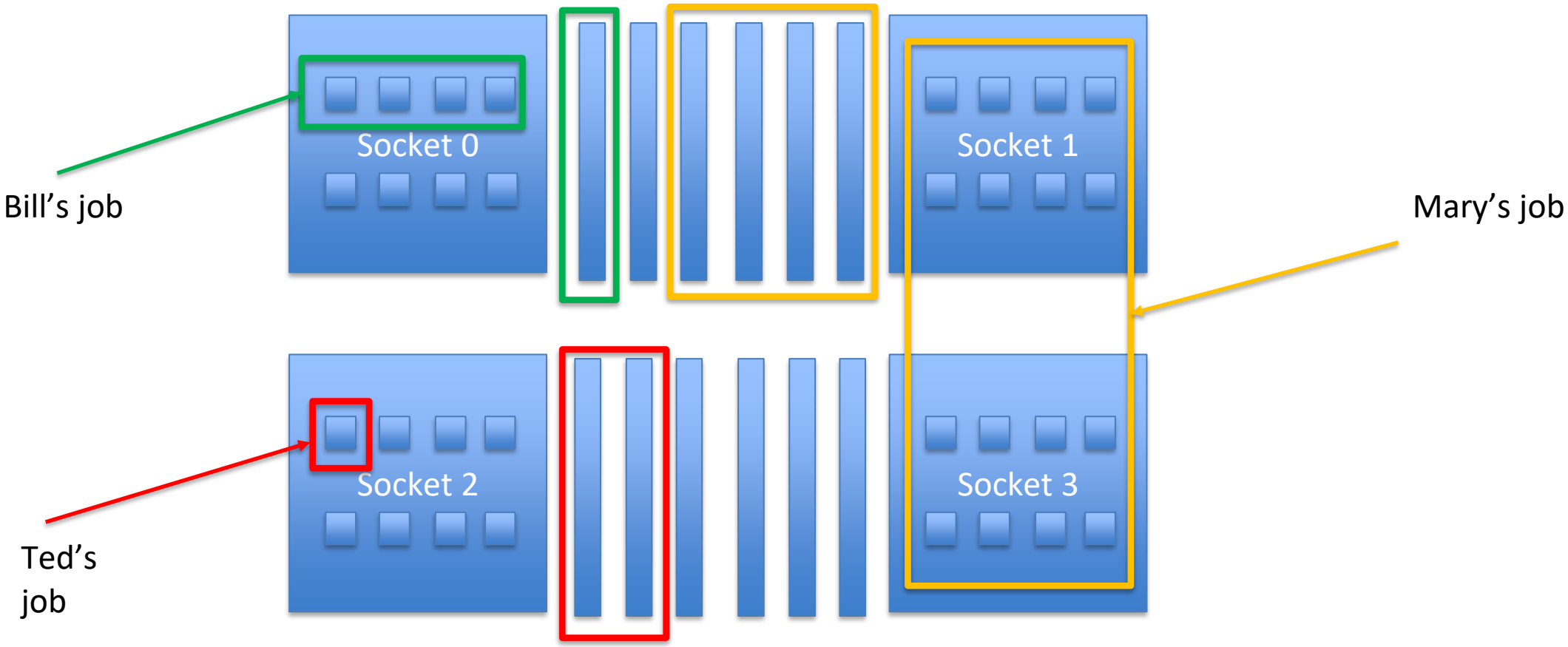
- Node
- Memory
- CPU’s
- GPU’s
- Licenses



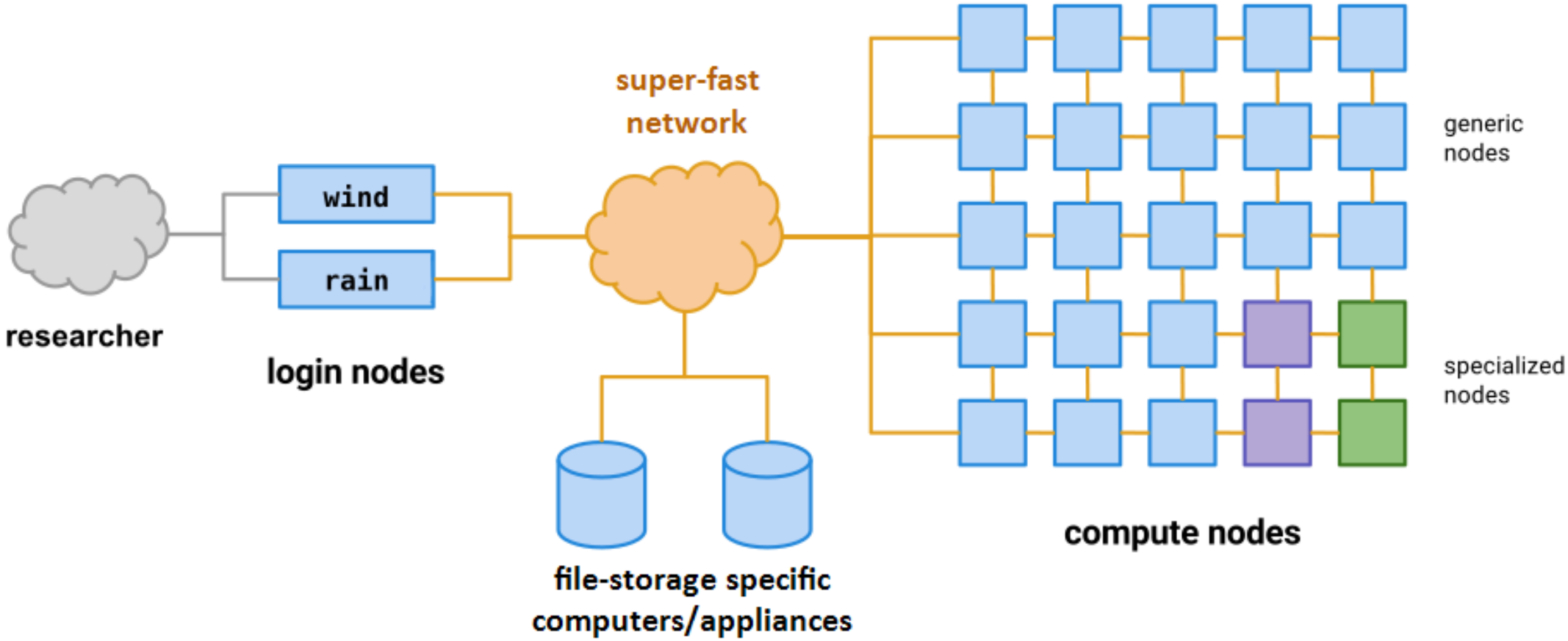
Many scheduling methods

- 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

Inside a (single) Node



Cluster = Login-nodes + Compute-nodes + etc



Monsoon Today

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

- The Monsoon cluster is a resource available to the NAU research enterprise
- 107 systems (nodes) – cn[1-33,35-108]
- 4048 Intel, and AMD cores
- 27 GPUs, NVIDIA Tesla K80, P100, V100, A100
- Red Hat Enterprise Linux 8.9
- 26TB memory - 128GB/node min, 2TB max
- 1PB high-speed scratch storage (Lustre)
- 615TB long-term storage (ZFS)
- High speed interconnect: FDR, and HDR Infiniband

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, 1,194 PF, 21 kW - USA
  - Aurora (#2), 4.7M cores, 585PF, 24 kW - USA

Small Cluster!

Dual core?

Largest Cluster!

8.7M cores

NAU NORTHERN ARIZONA UNIVERSITY

Monsoon scheduling

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

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 – 20GB 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 “getquotas” now
- /scratch – 1PB total space, 30 day retention
  - Very fast storage, capable of 20GB/sec
  - Quota: 15TB, 2M files
  - 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
- /common
  - Cluster support share
  - Contrib: place to put software/libs/confs/db's for others use

Data Flow

1. Keep scripts and executables in /home or in 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
  - Drag and drop files
- scp
  - `scp <files> <nauid>@dtn1.hpc.nau.edu:/scratch/<nauid>`
  - WinSCP (windows)
  - Fetch (mac)
    - Download from: nau.edu/its/software
- samba / cifs
  - 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"

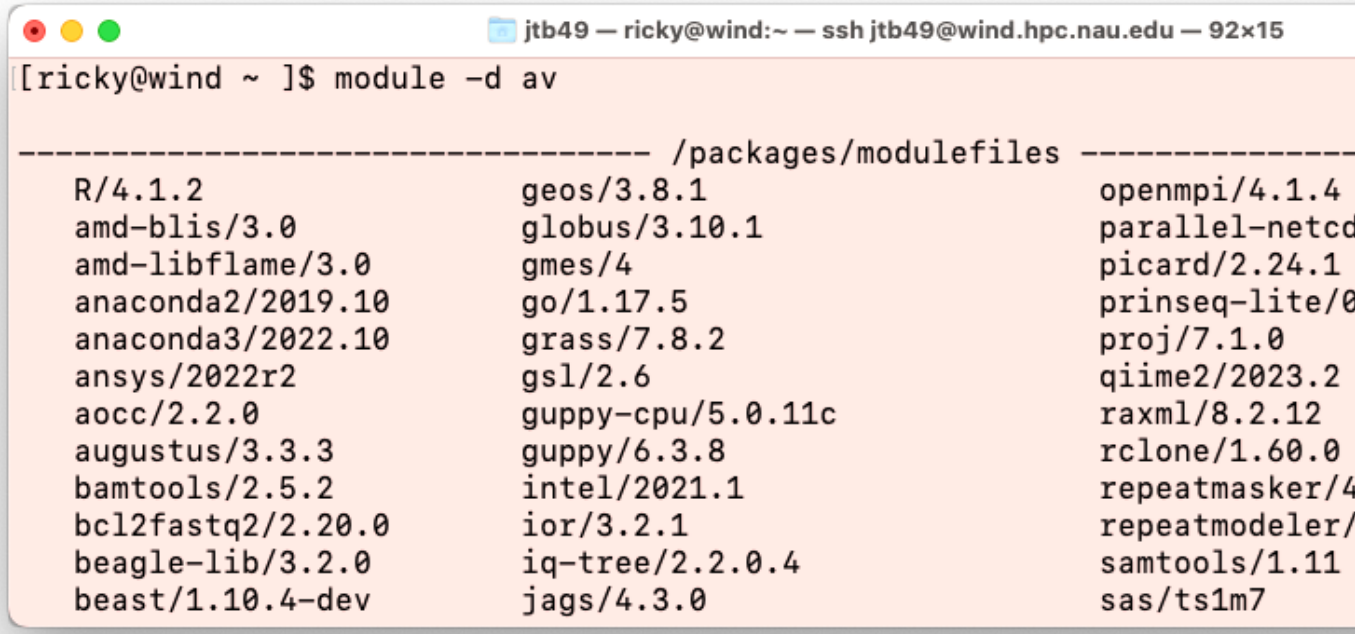
Modules

- 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*
- `module display <module name>` …*detailed information including environment variables effected*

Software

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



NAU NORTHERN ARIZONA UNIVERSITY

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 note on MPI
- 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

- 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 up into smaller pieces? Smaller pieces can make the workload more agile.
- How long should your job run for?
- 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. module loading
3. srun job steps/statements for the actual work

Example Job script

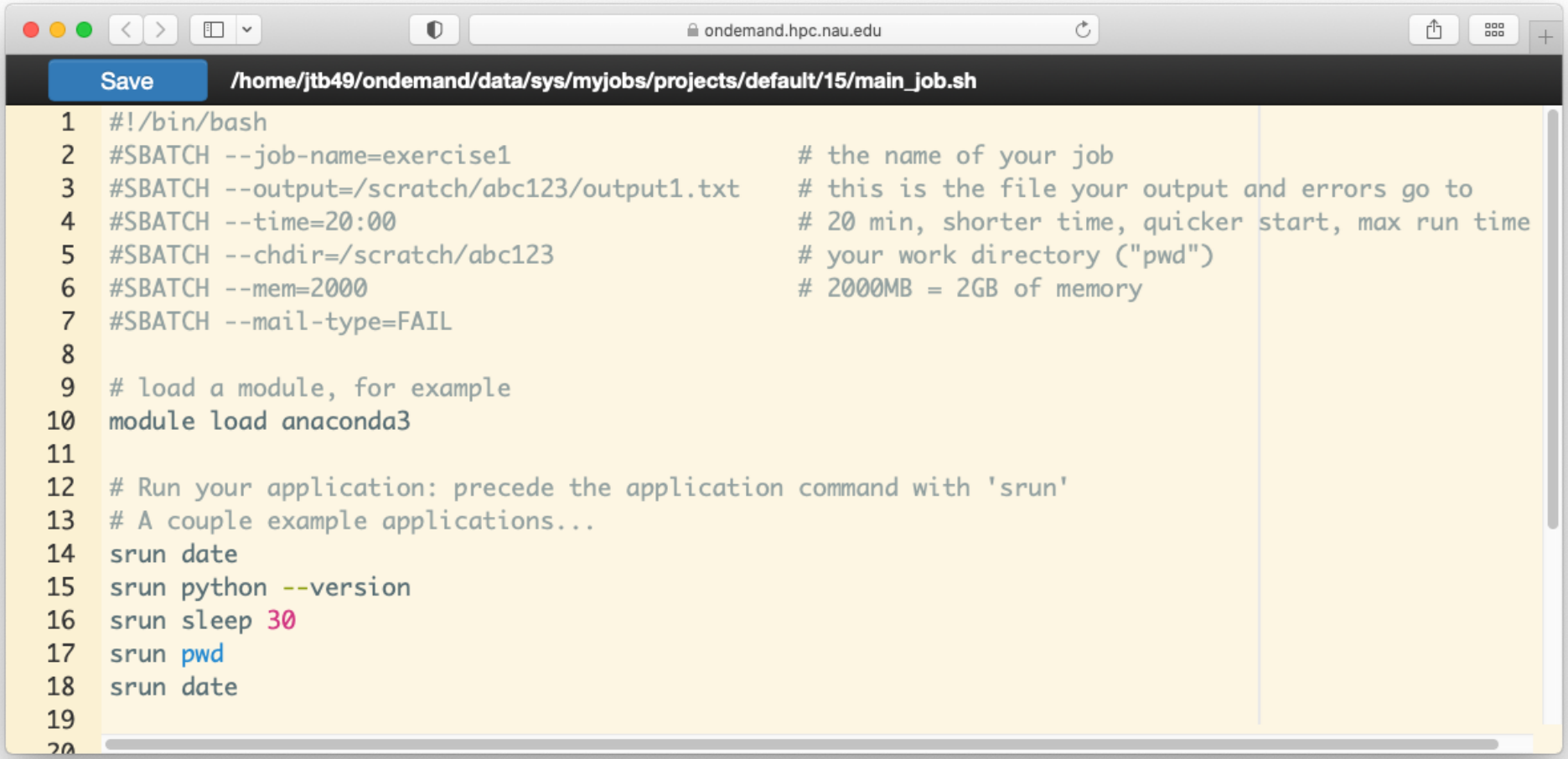
```
#!/bin/bash
#SBATCH --job-name=test
#SBATCH --output=/scratch/NAUID/output.txt  #  the stdout from your job goes here
#SBATCH --time=20:00                                                    #  shorter time = sooner start
#SBATCH --chdir=/scratch/NAUID                                          #  default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

NAU NORTHERN ARIZONA UNIVERSITY

Example Job script (in Ondemand's editor)



NAU NORTHERN ARIZONA UNIVERSITY

Job script Templates

- We have a spot for job script templates here:
  - /common/contrib/examples/job_scripts

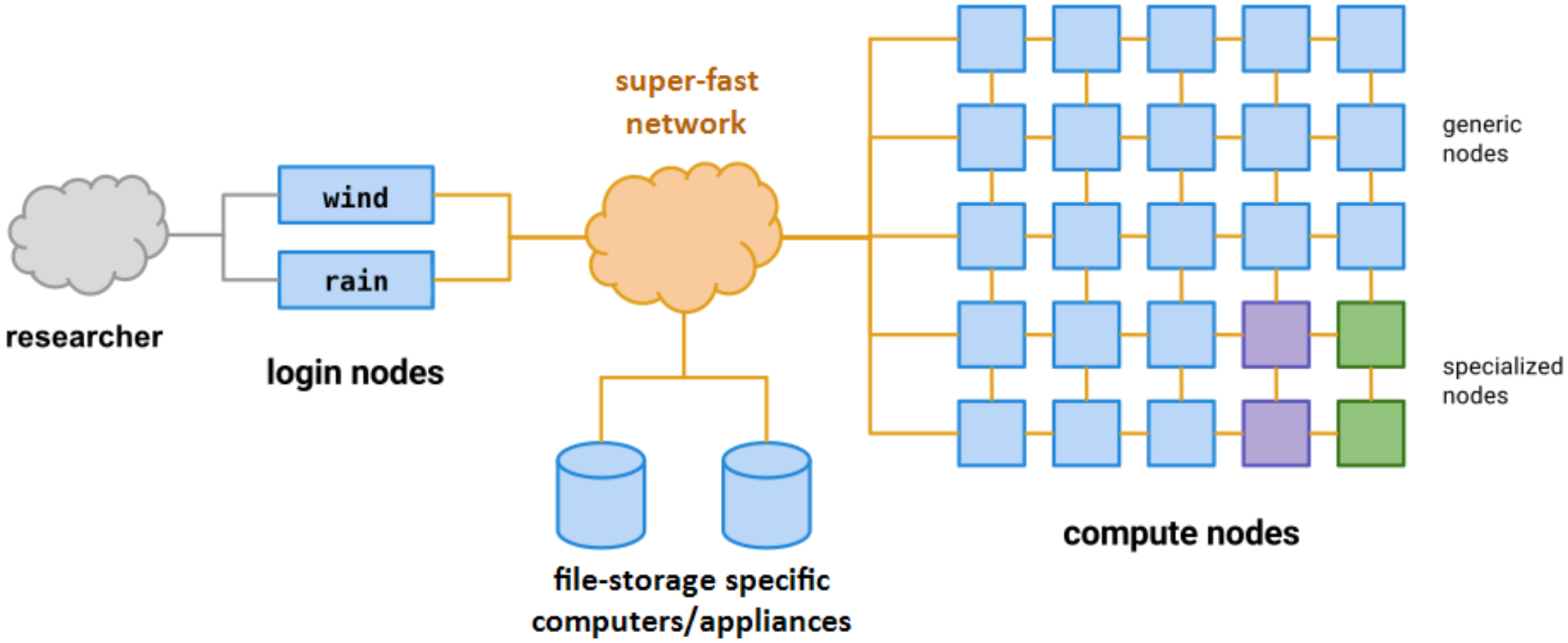
Job Parameters

You want	Switches needed
More than one cpu for the job	--cpus-per-task=2, or -c 2
To specify an ordering of your jobs	--dependency=afterok:job_id, or -d job_id
Split up the output, and errors	--output=result.txt --error=error.txt
To run your job at a particular time/day	--begin=16:00 --begin=now+1hour --begin=2010-01-20T12:34:00
Add MPI tasks/ranks to your job	--ntasks=2, or -n 2
To control job failure options	--norequeue –requeue
To receive status email	--mail-type=ALL

Contraints 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)	-G1, --gres=gpu:tesla:1
To reserve a whole node for yourself	--exclusive
To chose a partition	--partition

Cluster = Login-nodes + Compute-nodes + etc



NAU NORTHERN ARIZONA UNIVERSITY

Accessing Monsoon

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

- Connect to OpenOndemand web interface at: https://ondemand.hpc.nau.edu
- ssh into a login node
  - Windows Users:
    - use Putty (preferred) or Powershell
  - Mac, Linux, or Unix users: use ssh command
  - login nodes:
    - monsoon.hpc.nau.edu aka wind (for research)
    - rain.hpc.nau.edu (for class work)
  - data transfer nodes:
    - dtn1.hpc.nau.edu
      - Special purpose node, use for large data transfers and modifications

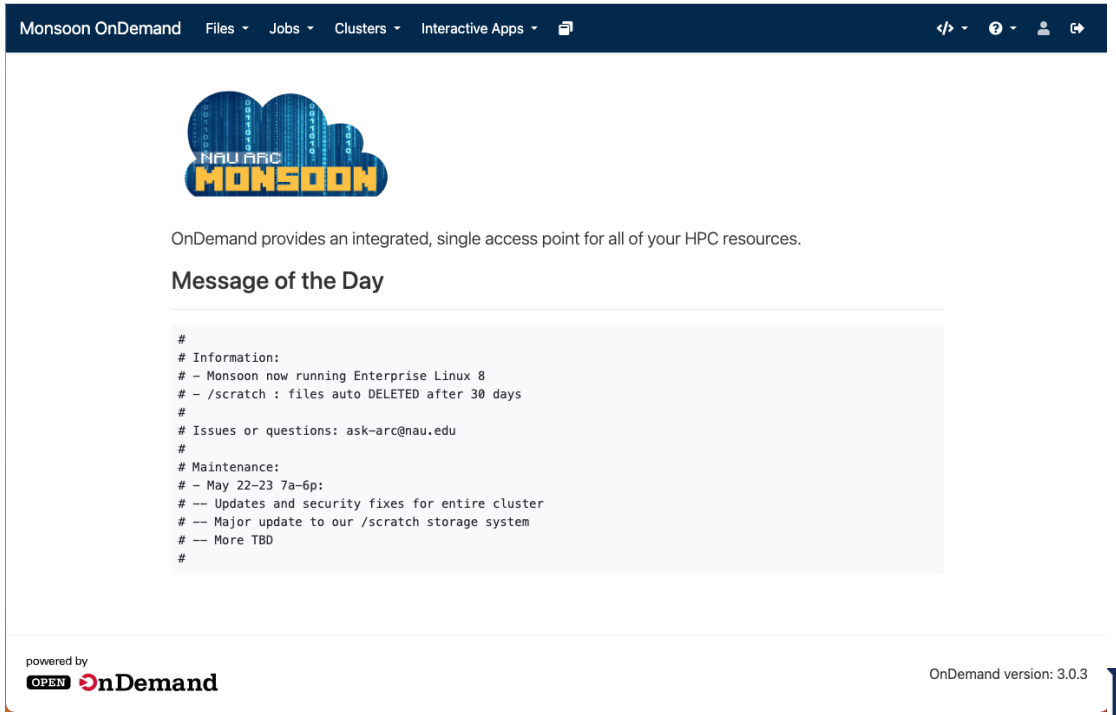
- SMB connection (files only)
  - \\shares.hpc.nau.edu\cirrus
  - see guide here: https://in.nau.edu/arc/overview/file-management/

Login node vs Compute node

- When you log into “monsoon” interactively via SSH 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!

Ondemand

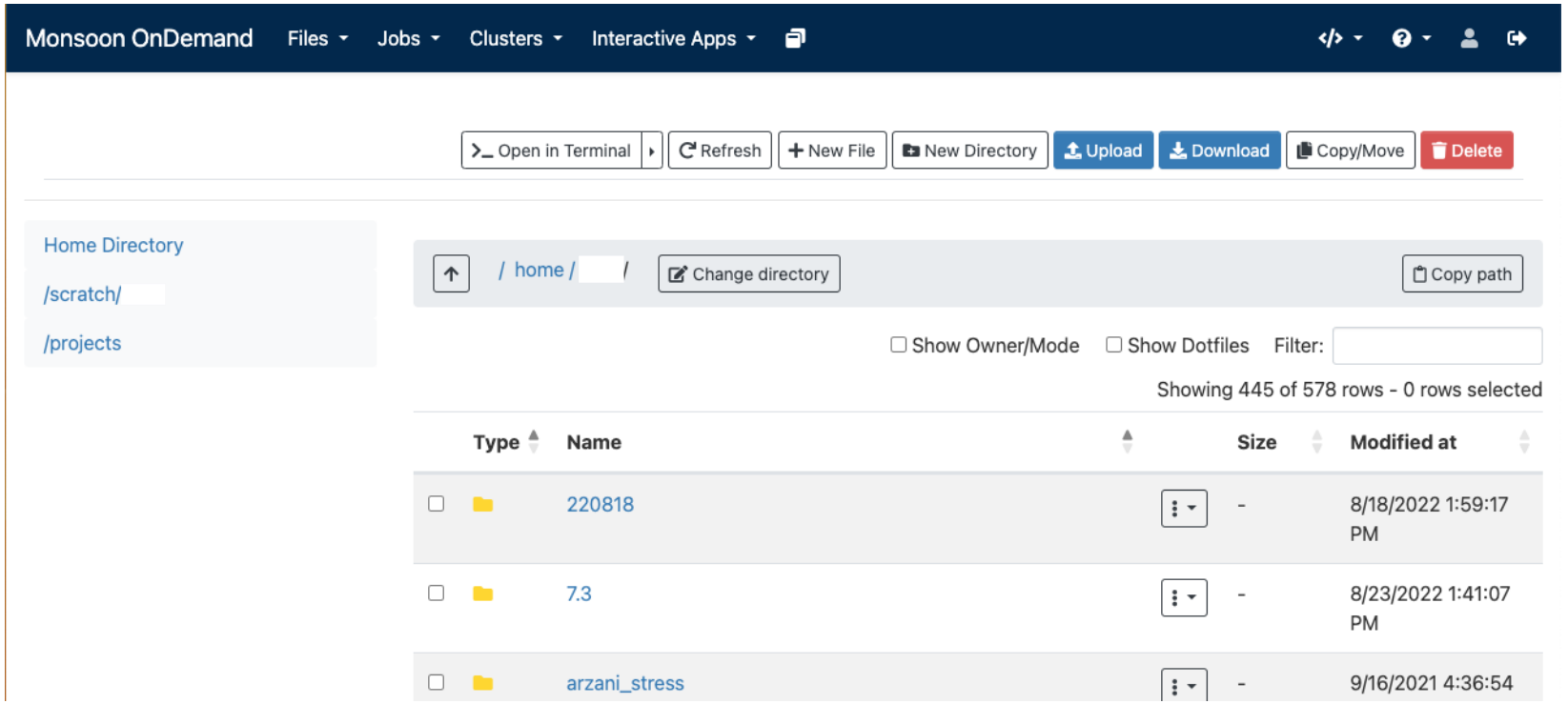
- Open Ondemand (OOD) is an interactive Graphical User Interface (GUI) to the Cluster. You access it from your web browser at https://ondemand.hpc.nau.edu



NAU NORTHERN ARIZONA UNIVERSITY

Ondemand File Explorer

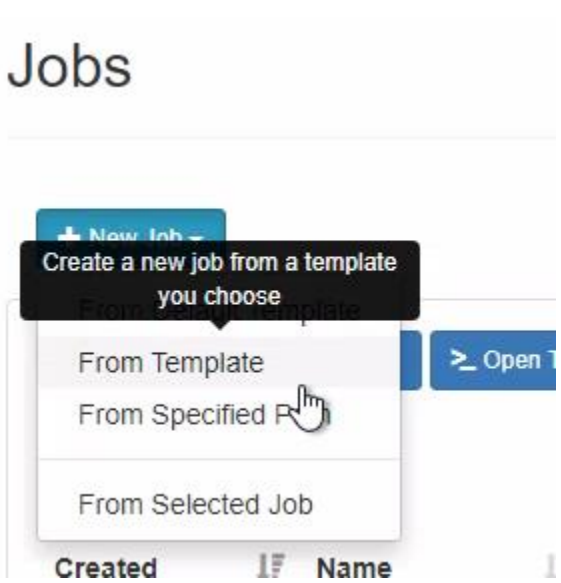
- The file explorer is used to explore, and transfer the files in your home, scratch, or other areas on the cluster.

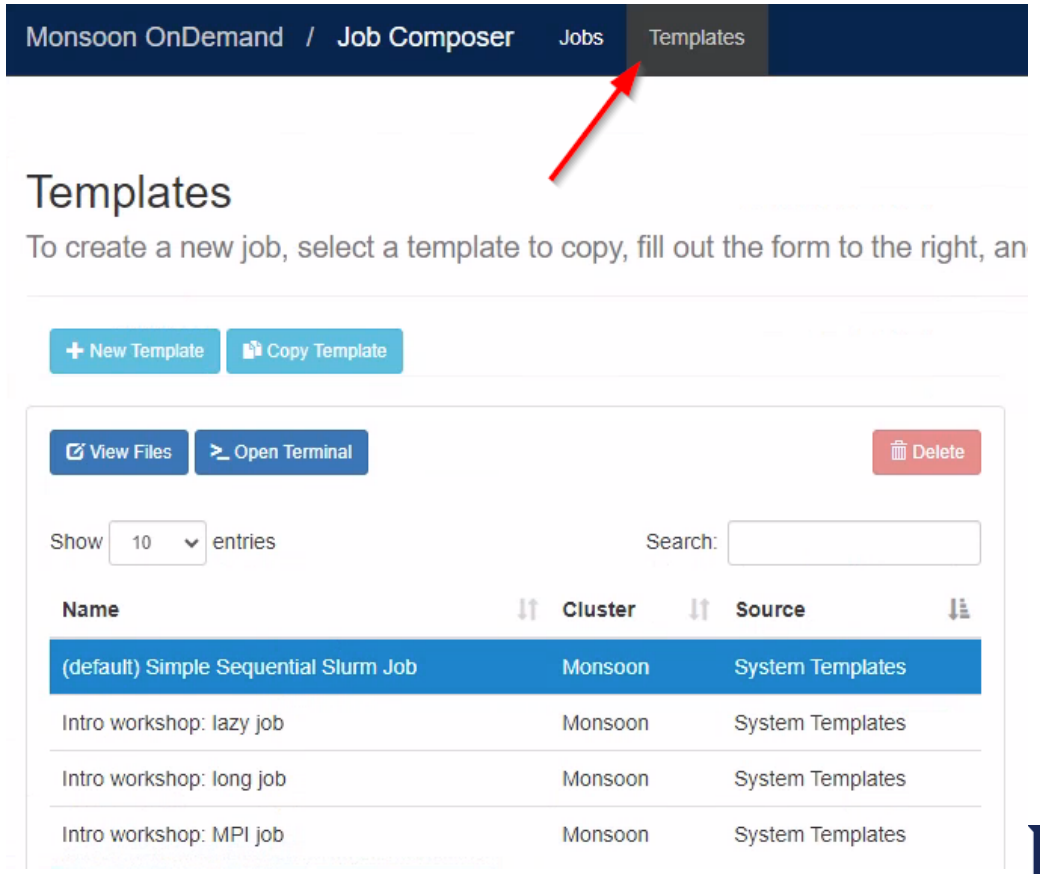


NAU NORTHERN ARIZONA UNIVERSITY

Ondemand Job Composer

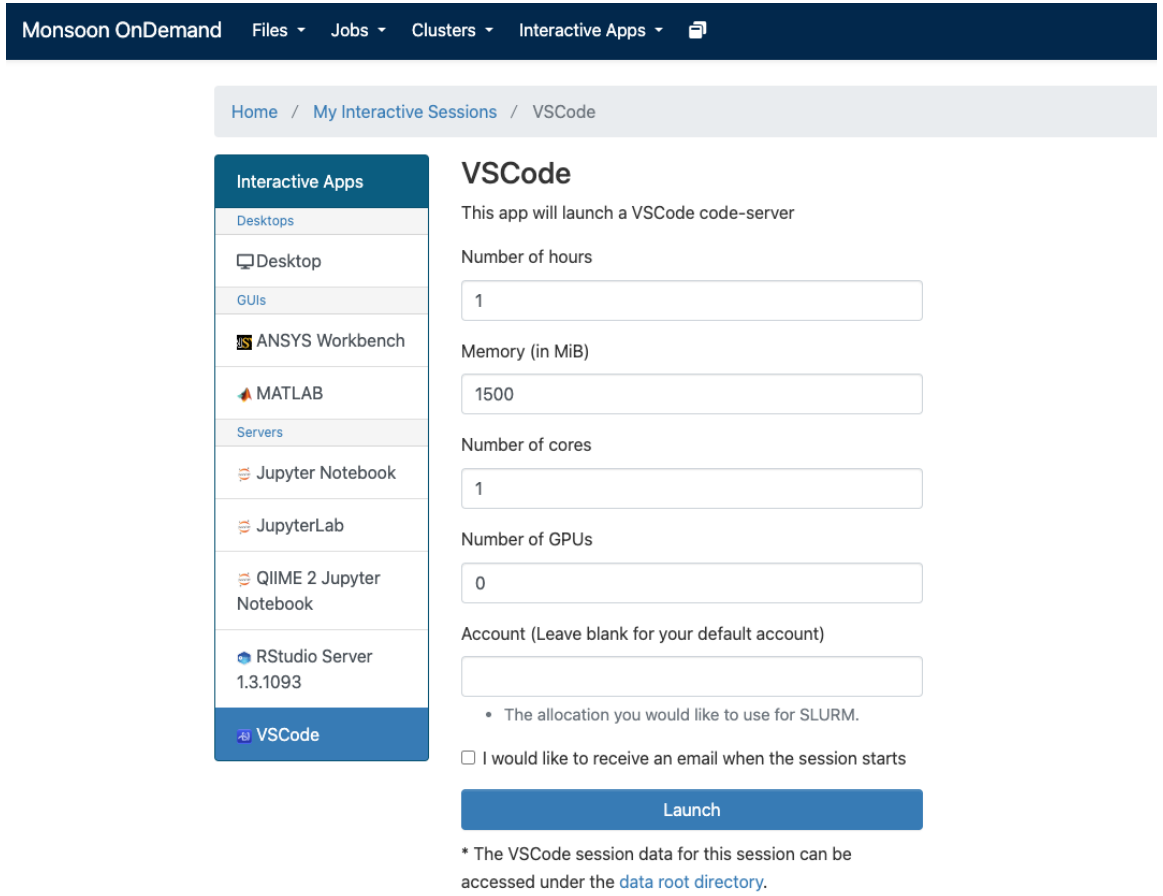
- The Job Composer is used to create and run jobs.





NORTHERN ARIZONA UNIVERSITY

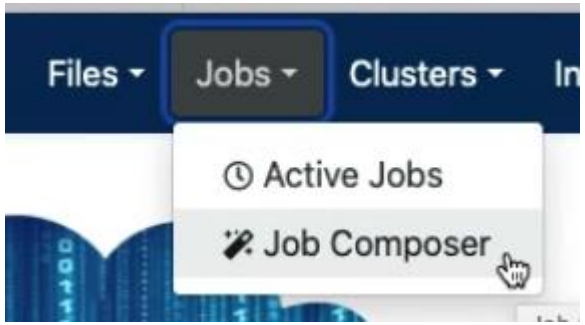
Ondemand Apps

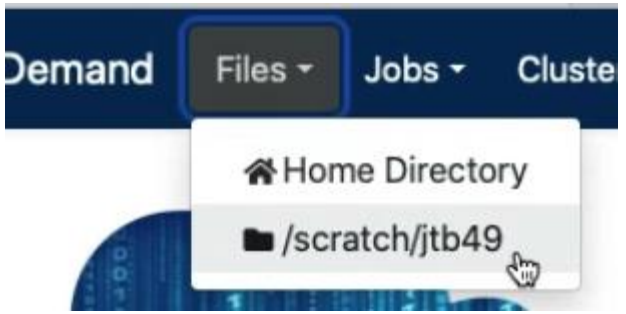


Exercise 1

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`



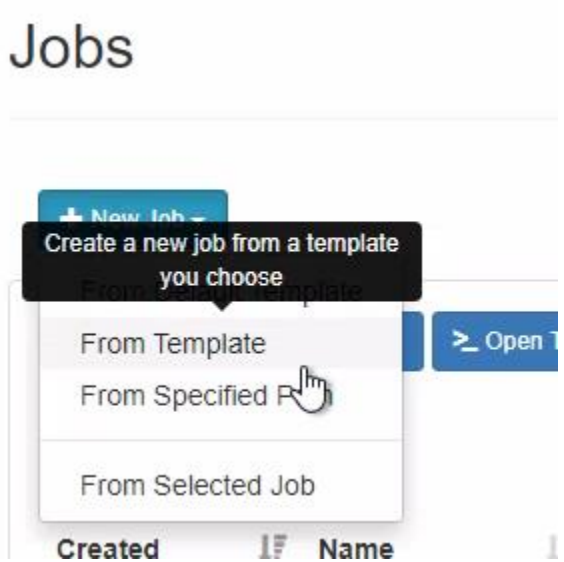


[Exercise 1 (CLI)]

NAU NORTHERN ARIZONA UNIVERSITY

Exercise 2

- Create a new job, this time using **New Job > From Template**
- Choose the “long job” template and click **Create New Job**
- Scroll down and click **Open Editor**
- Change all “NAUID” to be your own NAU user-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 your edits, and Submit this job
- Monitor your job by selecting **Jobs > Active Jobs** from your Dashboard.
- Examine the output in long.txt
- Make a note of the secret code from long.txt



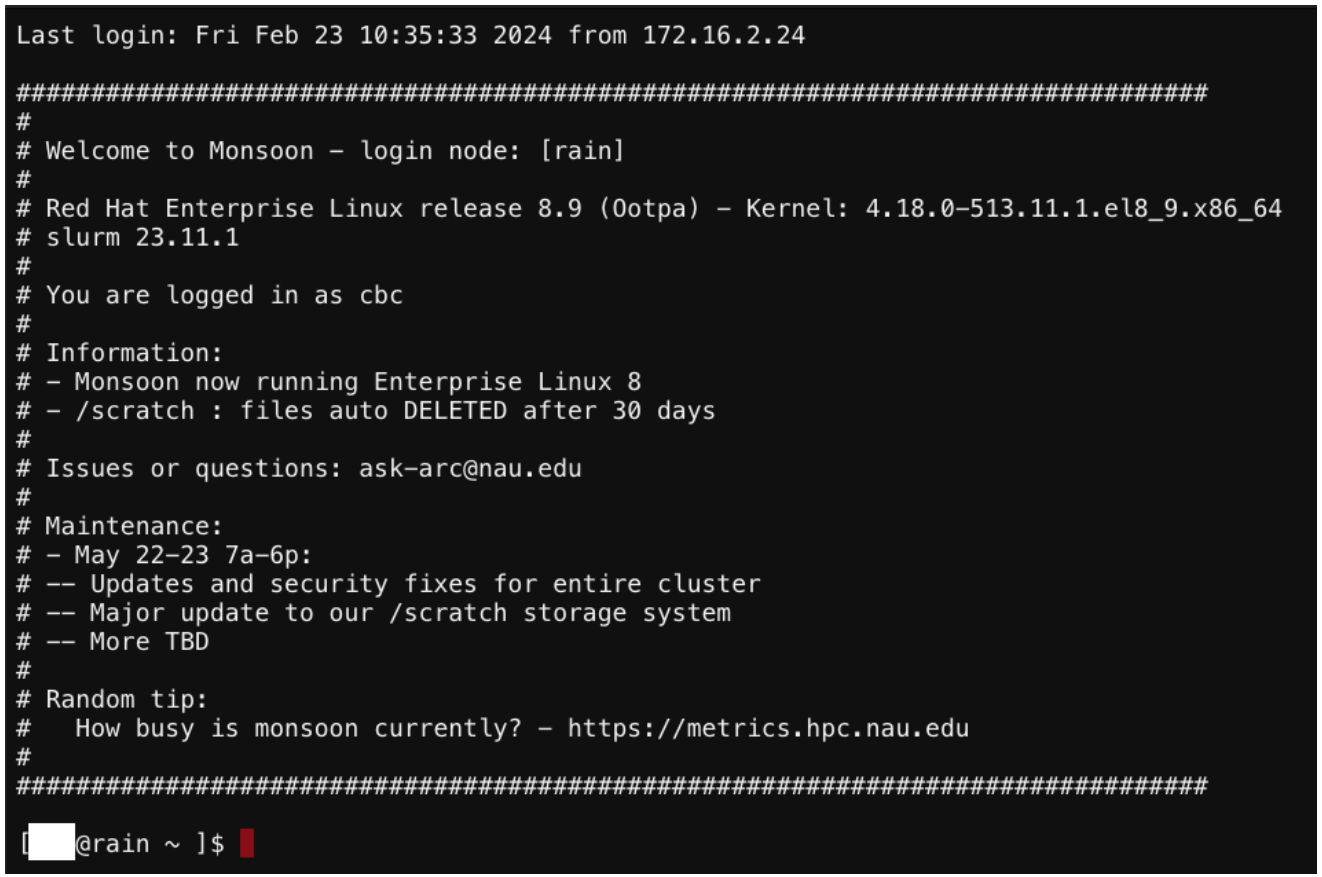
[Exercise 2 (CLI)]

Command-line access

- Once you have the basics down using Ondemand, then the power of the cluster is exposed through the command-line (CLI).
- Access the CLI from the Dashboard, under clusters menu (or via SSH)
- Follow along after opening the CLI
- 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**



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!

NAU NORTHERN ARIZONA UNIVERSITY

Interactive / Debug Work

- Run your compiles and testing on the cluster nodes by:
  - srun -p all 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: Relinquising job allocation 33442
```

(For interactive MPI):

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

Submitting 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

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

Cluster info

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

Monitoring your job

- 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

- sstat
  - Display various statistics and information of a running job
- sstat -j jobid
- sstat -o AveCPU,AveRSS

- Only works with jobs where analysis is executed with “srun”

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)

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 85224
- scontrol update jobid=6880341 timelimit=4:00:00

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>

Account hierarchy

- Your user account belongs to a parent faculty account (group)
- Your user account shares resources that are provided for your group
- Example:
  - account1
    - user1
    - user2
- View the account structure you belong to with: “sshare -a –A <account>”
- Example:
  - sshare -a -A account1

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 squeue to monitor the job
  - sstat -j <jobid>, and squeue –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

  *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, 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 storage areas

```
[ricky@wind ~ ]$ getquotas
Filesystem  #Bytes   Quota     %      |  #Files  Quota  %
/home       13684M  20000M  68%  |  -        -       -
/scratch    67.62G    9.313T    0%    |  419K    2M     20%
```

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=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

analysis 123456_1	analysis 123456_2	analysis 123456_3	analysis 123456_4
analysis 123456_5	analysis 123456_6	analysis 123456_7	analysis 123456_8

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

NAU NORTHERN ARIZONA UNIVERSITY

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 “squeue”, 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 likely utilize the command line interface (CLI)
  - However, the terminal in ondemand is worth using all the time

Question and Answer

- More info here:
  http://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)

- `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 excercise2.out
- Make a note of the secret code from exercise2.out

Next Slide: Command-line Access