Intro to Monsoon and Slurm (classroom)

RECORD ZOOM

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

Christopher Coffey
2/25/2022

NAU NORTHERN ARIZONA UNIVERSITY

Get logged in!

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

- From a Computer:
  - Log into NAU VPN!
    - 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 ondemand.hpc.nau.edu
    - Log in with your_louie_id
  - Click on clusters tab, and select monsoon cluster login shell

NAU NORTHERN ARIZONA UNIVERSITY

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?
  - Exercises
  - Question and answer

What is a cluster?

- A computer cluster is many individual computers systems (nodes) networked together locally to serve as a single resource
- Ability to solve problems on a large scale not feasible alone

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

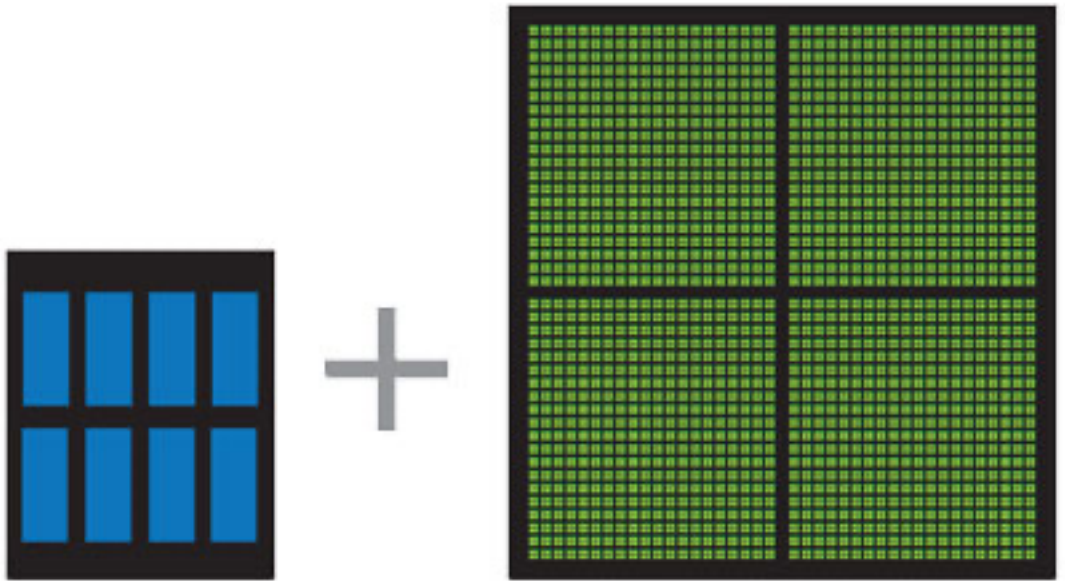
- A scheduler needs to know what resources are available on the cluster
- Assignment of work on a cluster is carried out most efficiently with scheduling and resource management working together

Resource Management

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

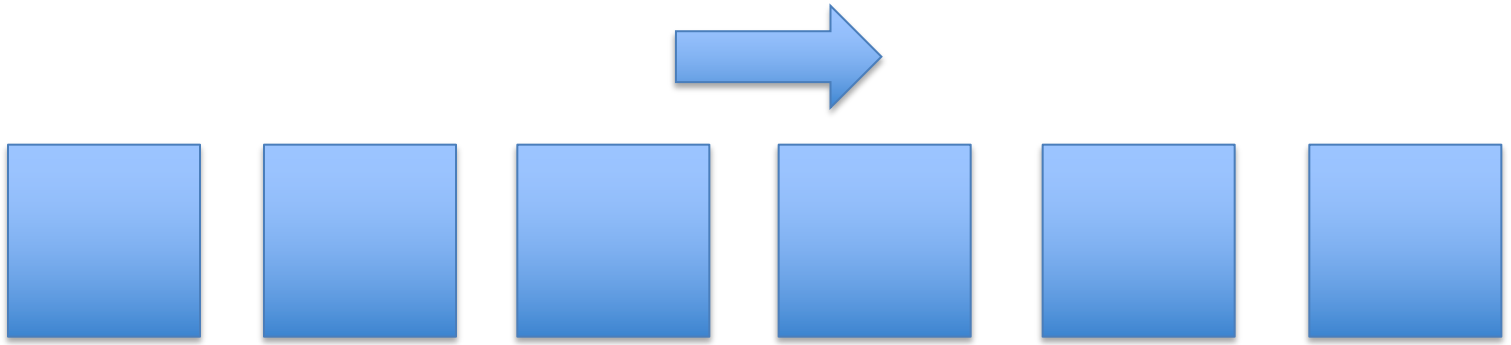
Cluster Resources

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



What is a queue?

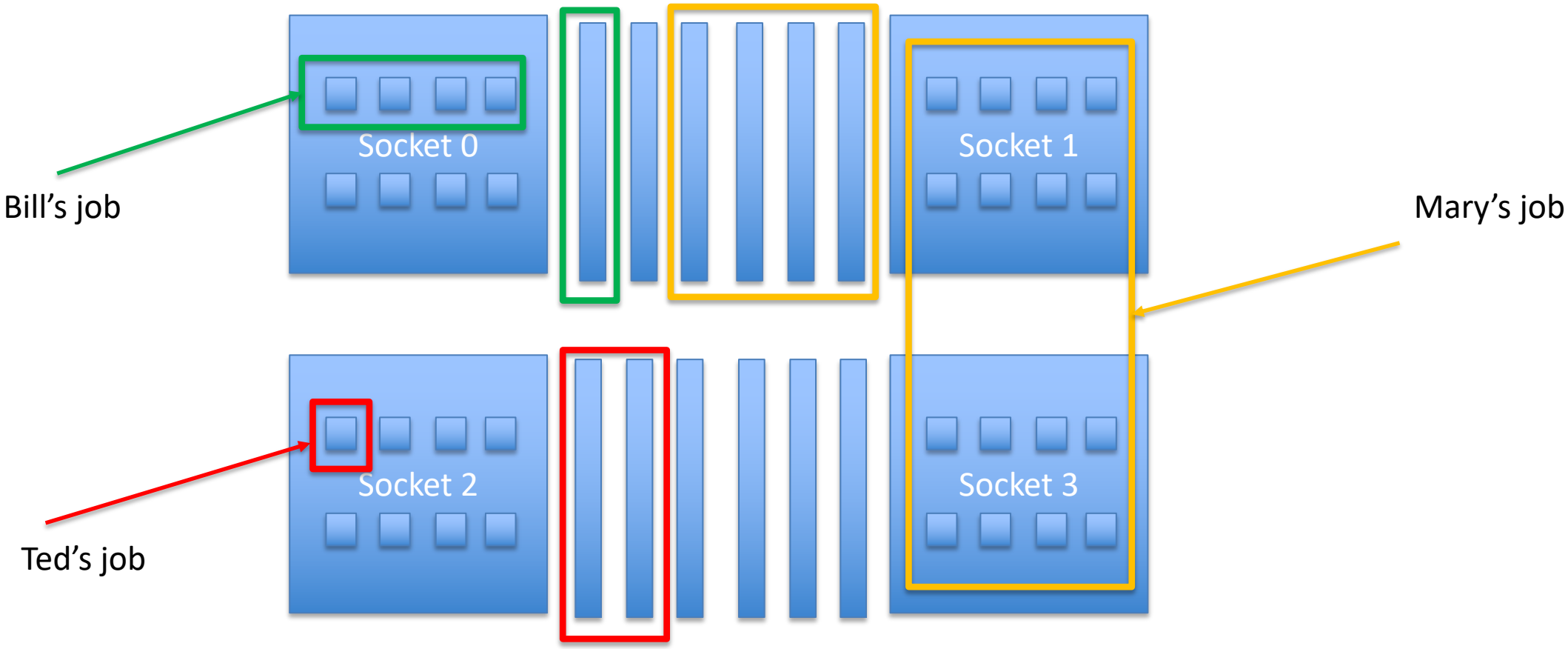
- Normally thought of as a line, FIFO
- Queues on a cluster can be as basic as a FIFO, or far more advanced with dynamic priorities taking into consideration many factors



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 users recent resource consumption

Inside a Node



Monsoon Today

- The Monsoon cluster is a resource available to the NAU research enterprise
- 105 systems (nodes) – cn[1-105]
- 3824 Intel, and AMD cores
- 20 GPUs, NVIDIA Tesla K80, P100, and V100
- Red Hat Enterprise Linux 8.4
- 24TB memory - 128GB/node min, 2TB max
- 1PB high-speed scratch storage (Lustre)
- 615TB long-term storage (ZFS)
- High speed interconnect: FDR, and HDR Infiniband

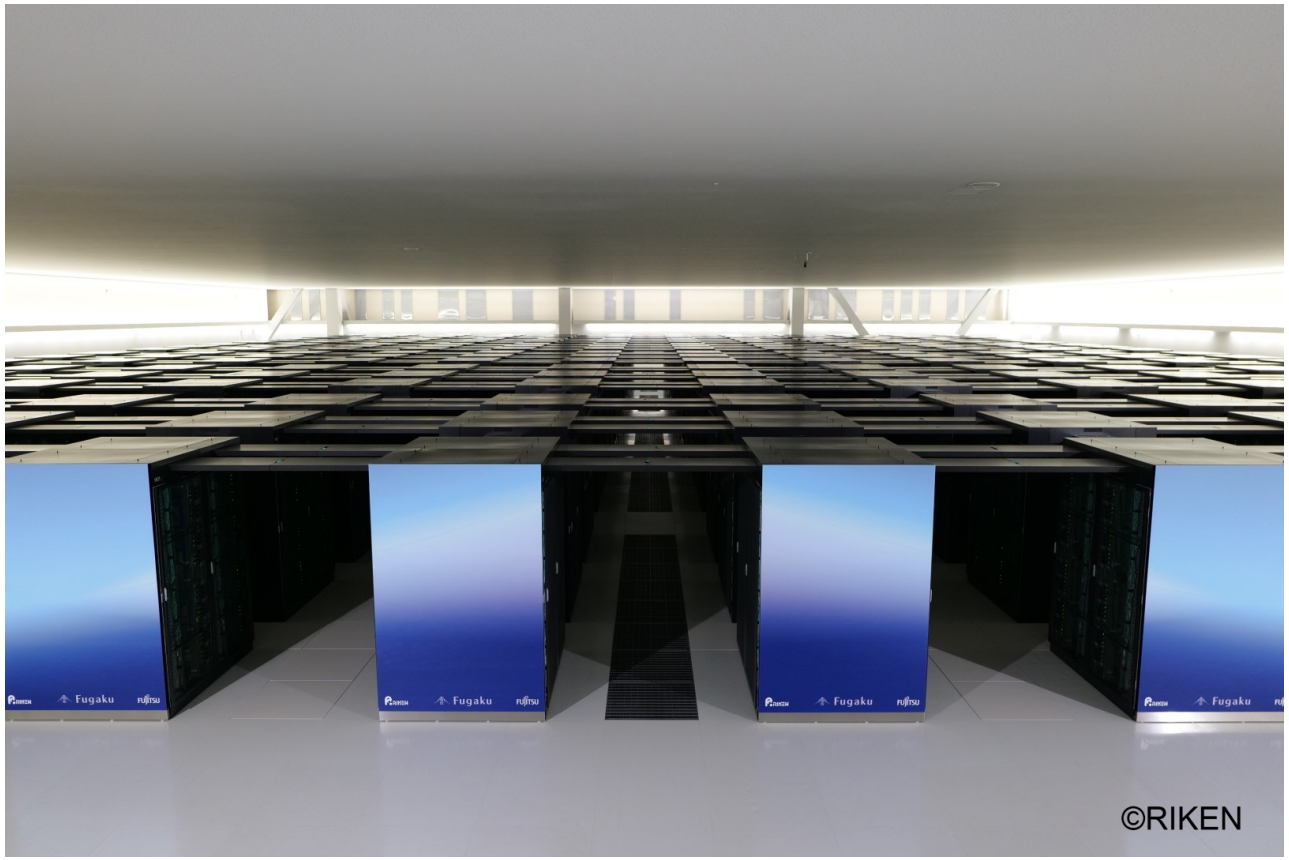
Slurm … yummm

- 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:
  - Fugaku (#1), 7.6M cores, 537 PF, 30 kW - Japan
  - Summit (#2), 2.4M cores, NVIDIA Volta GPUs, 200 PF, 10.1k kW - USA

Small Cluster!

Dual core?

Largest Cluster!



7.6M cores

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 1 days
- 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 – 10GB quota
  - Keep your scripts and executables here
  - Snapshotted twice a day: /home/.snapshot
  - Please do not write job output (logs, results) here!!
- /scratch – 500TB total space, 30 day retention
  - Very fast storage, capable of 11GB/sec
  - Quota: 10TB, 2M files
  - Checkpoints, logs
  - Keep all temp/intermediate data here
  - Should be your default location to perform input/output

Data Flow

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

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

Remote storage access

- scp
  - scp files nauid@wind.hpc.nau.edu:/scratch/nauid
  - WinSCP (windows)
  - Cyberduck (mac)

Modules

- Software environment management handled by the *modules* package management system

- 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!

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, using pthreads, 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 program goes here
- #SBATCH --time=20:00 # shorter time = sooner start
- #SBATCH --chdir=/scratch/nauid # default location slurm searches

- # replace this module with software required in your workload
- 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

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

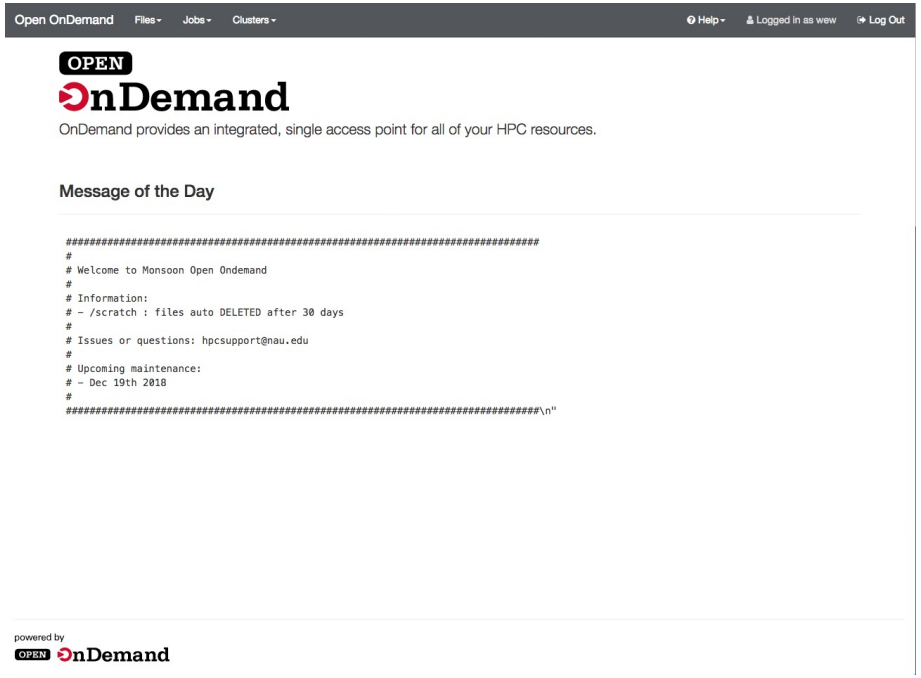
You want	Switches needed
To choose a specific node feature (e.g. avx2)	--constraint=avx2
To use a generic resources (e.g. a gpu)	--gres=gpu:tesla:1
To reserve a whole node for yourself	--exclusive
To chose a partition	--partition

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!
- Don’t attempt to complete your homeworks outside of slurm. If you do, they will be auto-killed, and your professor will be notified!

Ondemand

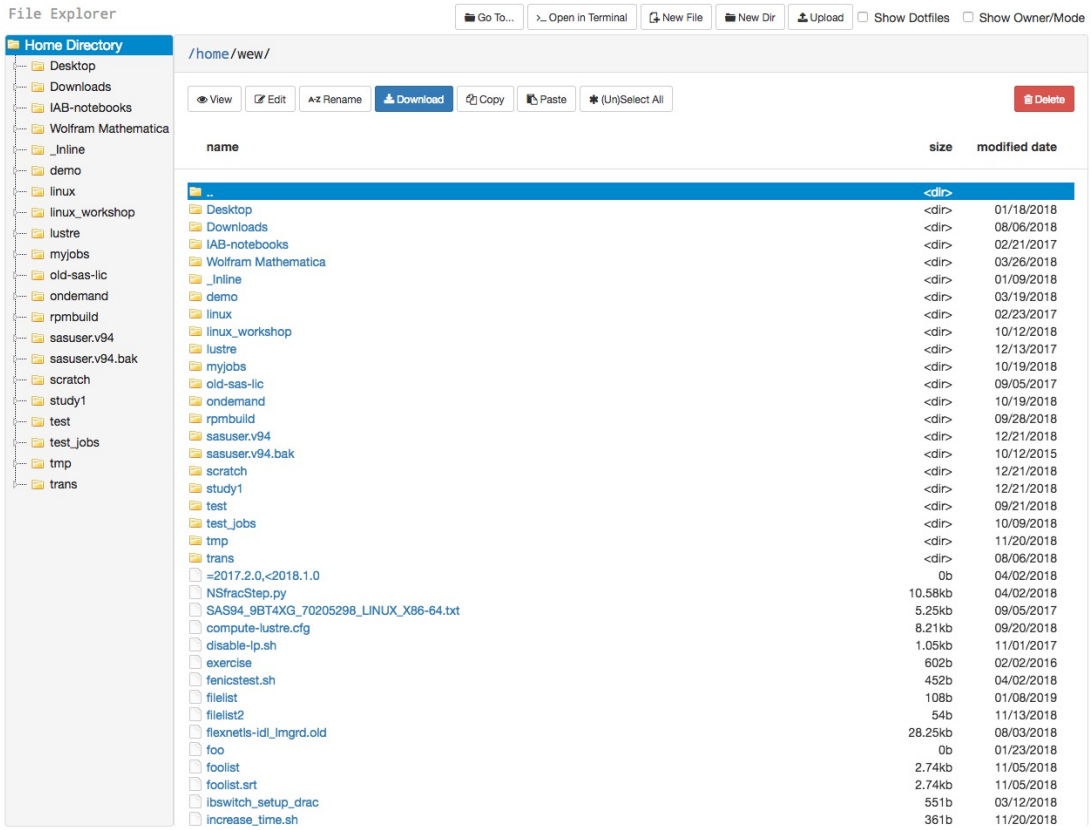
- Open Ondemand (OOD) is an interactive Graphical User Interface (gui) to the Cluster. You access it from your web browser at https://ondemand.hpc.nau.edu



NAU NORTHERN ARIZONA UNIVERSITY

Ondemand File Explorer

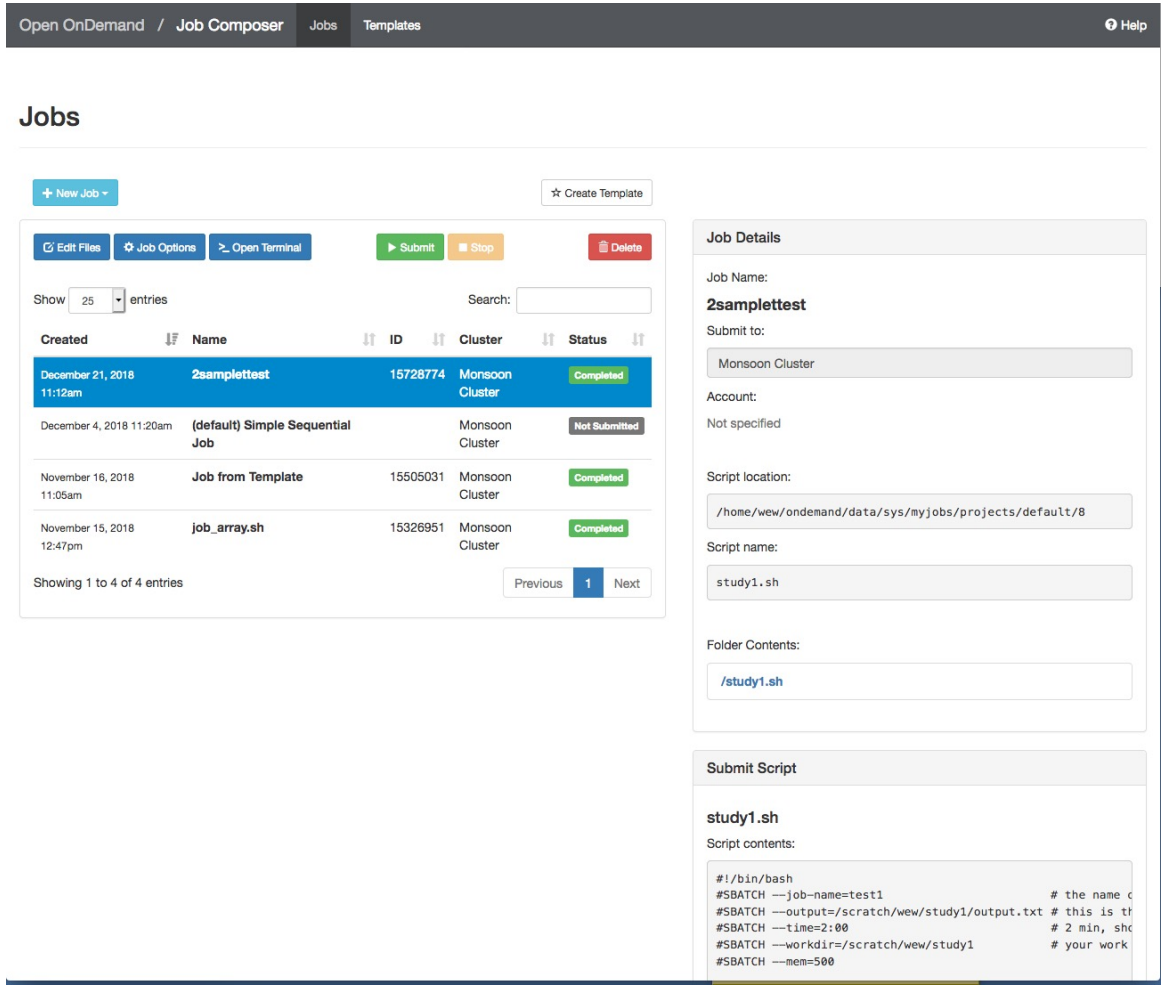
- The file explorer is used to explore, and transfer the files in your home, scratch, or other areas on the cluster.



NORTHERN ARIZONA UNIVERSITY

Ondemand Job Composer

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



NORTHERN ARIZONA UNIVERSITY

Changing default slurm account

- Some of you may already have an research slurm account
- If so, your classroom account won’t be the default
- Specify the correct Slurm account to use with:
- #SBATCH --account=
- For instance …
- #SBATCH --account=inf503-spr22

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.
- Click on New Job and select From Default Template
- Click on “open editor”
- Change NAUID to be your alpha numeric nau id, e.g. abc123!
- Name your job: “exercise1”
- Name your job’s output: “exercise1.out”
- Output should go to /scratch/<user>/exercise1.out
- Load the module called: workshop
- Run the “date” command
  - E.g. “srun date”
- And additionally, the “exercise1” command
- Save your job
- Submit your job via the job composer
- Use the File Explorer to examine your output (Goto -> /scratch/your_id)
- Make a note of the secret code in exercise1.out

Exercise 2

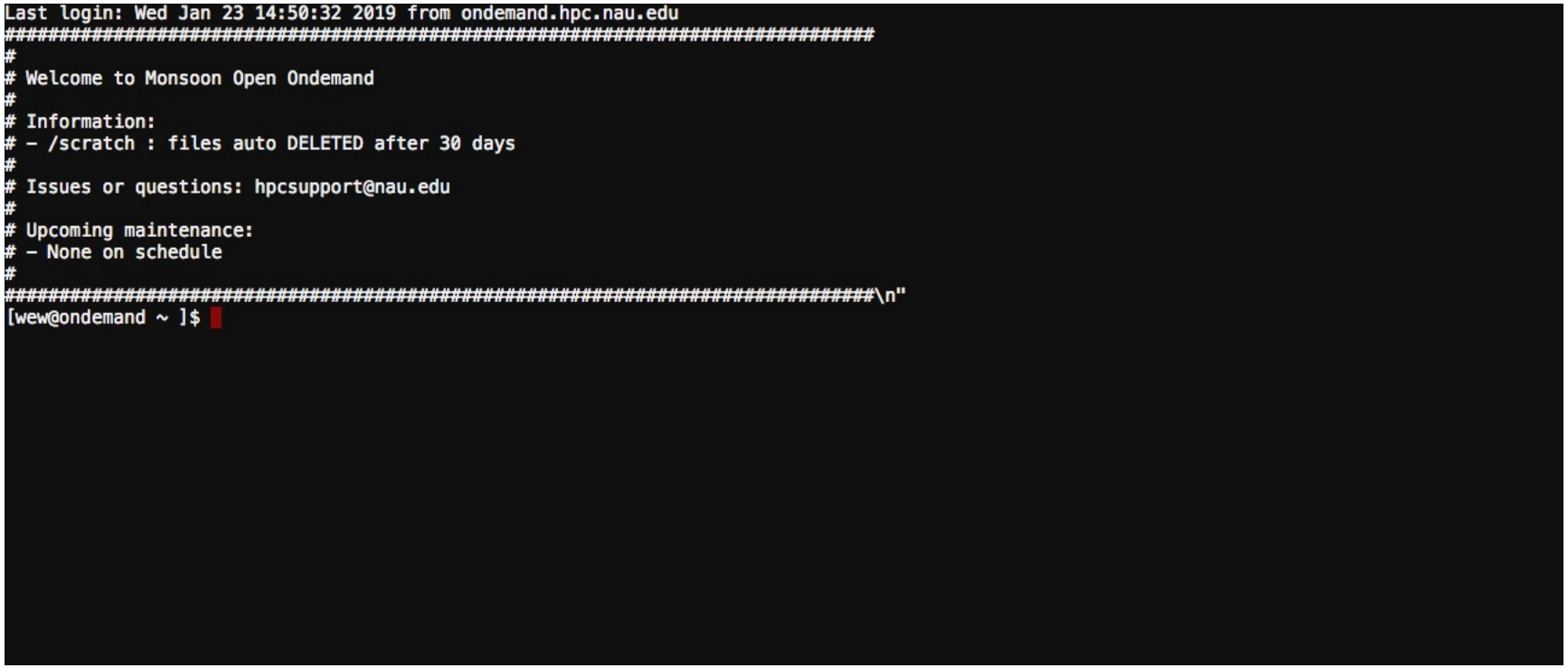
- Create a new job using “New Job” and “From Specified Path”.
- Source path /common/contrib/examples/job_scripts
- Name: longjob
- Script name: longjob.sh
- Cluster and account: leave empty
- Save
- Edit job, bottom left, change NAUID to be your id
- Load the module called: workshop
- Run the “exercise2” 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
- 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 commandline (CLI).
- Access the CLI from the Dashboard, under clusters menu
- 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 and Monsoon Cluster Shell Access



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 via Slurm

- 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 1 --time=2-00:00:00      # allocate 1 cpu for 2 days for your use
salloc: Granted job allocation 33442
[user1@wind ~ ]$ srun python analysis.py
[user1@wind ~ ]$ exit
salloc: Relinquising job allocation 33442

[user1@wind ~ ]$ salloc -c 1 --time=2-00:00:00
salloc: Granted job allocation 33443
[user1@wind ~ ]$ srun gcc -o a.out hw1.cc
[user1@wind ~ ]$ srun ./a.out
```

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)

Controlling your job

- scancel
  - Used to signal jobs or job steps that are under the control of Slurm.
- scancel -j 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 pending state, once the job starts, it can no longer be modified
- scontrol show job 85224
- scontrol update jobid=6880341 timelimit=4:00:00

Job Accounting

- To see job history, and job efficiency use jobstats!

  - jobstats -r # see todays jobs, including running jobs
  - jobstats -j <jobid> # see stats for the individual jobid
  - jobstats -S 9/1/19 # see job stats for all jobs since 9/1/19

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!

NAU NORTHERN ARIZONA UNIVERSITY

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/hpc/obtaining-an-account/

Question and Answer

- More info here:

  http://nau.edu/hpc
- Linux shell help here:
  - http://linuxcommand.org/tlcl.php
  - Free book download
  - https://nau.edu/HPC/Linux-External-Resources/
- And on the nauhpc listserv
  - nauhpc@lists.nau.edu