Intro to Monsoon and Slurm (compressed)

Christopher Coffey

3/12/2019

NORTHERN ARIZONA UNIVERSITY

Get logged in!

- From a Windows pc:
- Download putty:
  - Google for “putty”
    - Download, and install
  - Open the putty application
    - May need to search in start menu for it
  - In the hostname field:
    - your_louie_id@rain.hpc.nau.edu
  - Click open button
  - And accept the security key by typing ”y”

- From a Mac:
  - Open the terminal application
  - ssh your_louie_id@monsoon.hpc.nau.edu

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

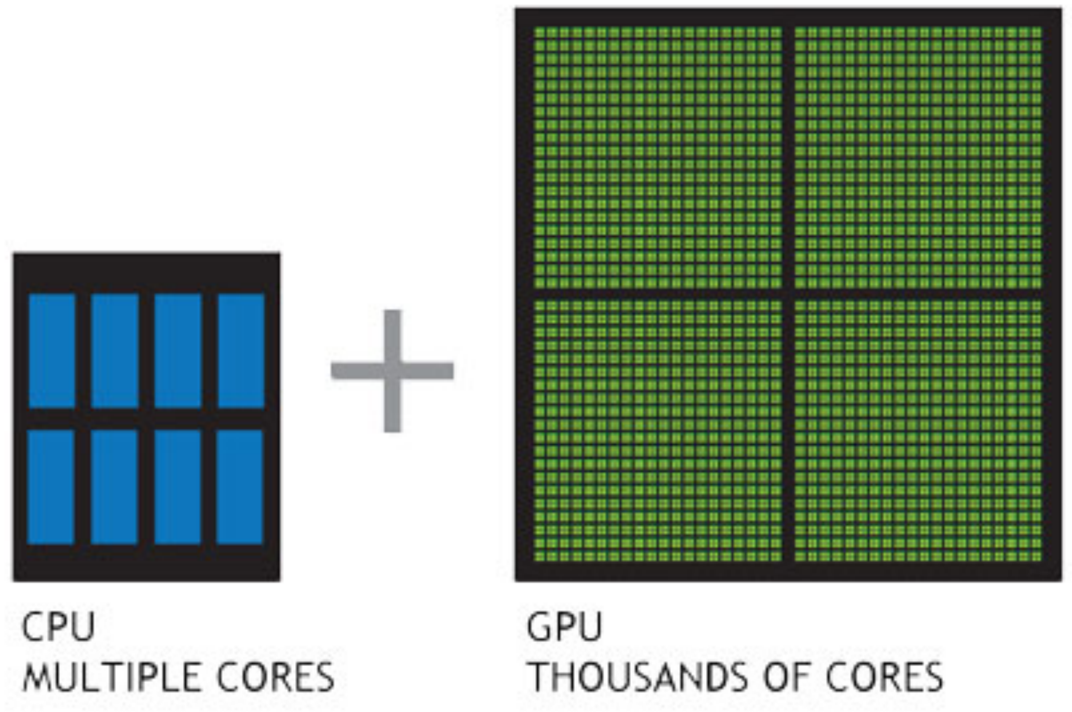
- A scheduler needs to know what resources are available on the cluster
- Assignment of work on a cluster is carried out most efficiently with scheduling and resource management working together

Resource Management

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

Cluster Resources

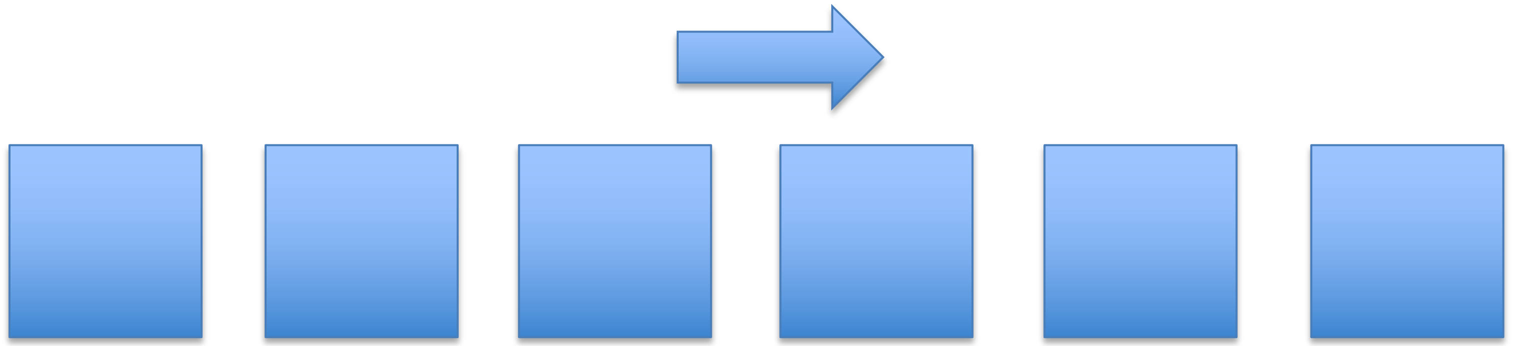
- Node
- Memory
- CPU’s
- GPU’s
- Licenses



NORTHERN ARIZONA UNIVERSITY

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



NORTHERN ARIZONA UNIVERSITY

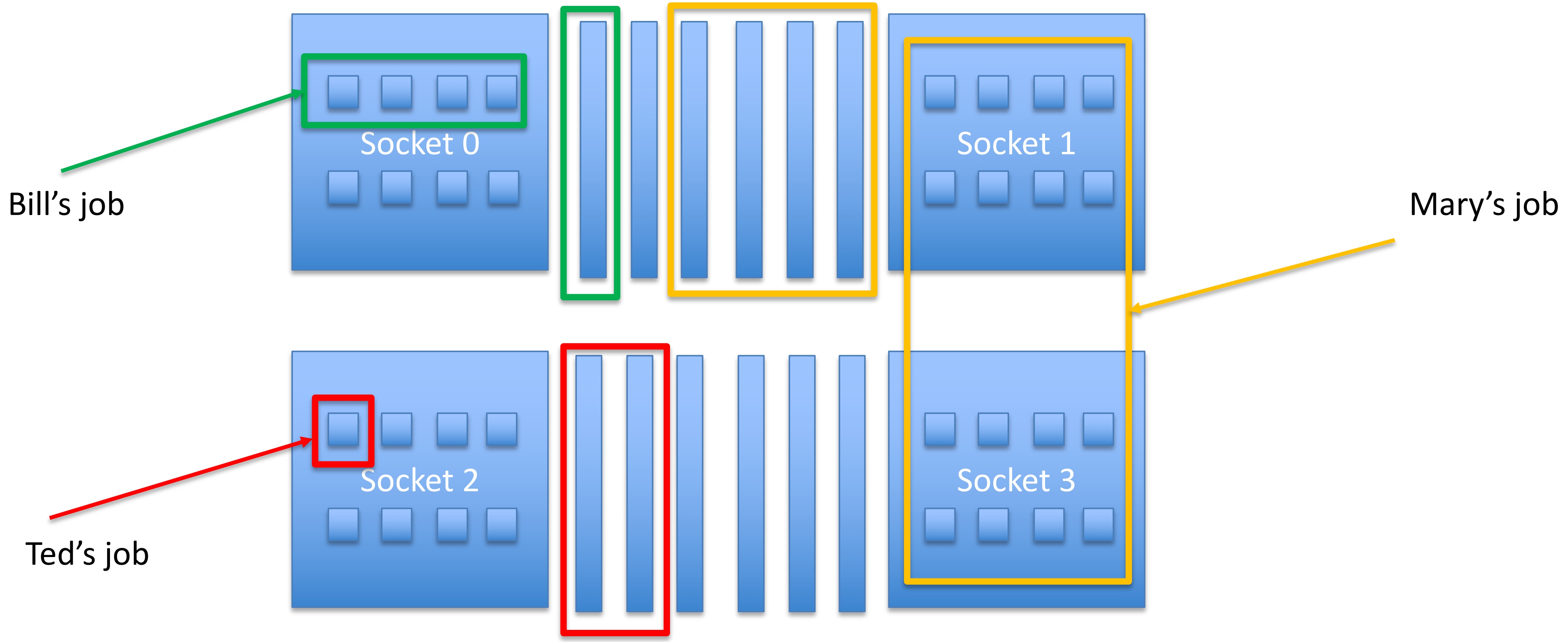
Cluster scheduling goals

- 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 users recent resource consumption

Inside a Node



Monsoon

- The Monsoon cluster is a resource available to the NAU research enterprise
- 103 systems (nodes) – cn[3-105]
- 2860 Intel Xeon cores
- 20 GPUs, NVIDIA Tesla K80, P100, and V100
- Red Hat Enterprise Linux 6.8
- 24TB memory - 128GB/node min, 1.5TB max
- 500TB high-speed scratch storage (Lustre)
- 615TB long-term storage (ZFS)
- High speed interconnect: FDR 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:
  - Summit (#1), 2.4M cores, NVIDIA Volta GPUs, 200 PF, 9.7k kW - USA
  - Sierra (#2), 1.5M cores, 125 PF, 7.4k kW - USA

Small Cluster!

Dual core?

Largest Cluster!

Monsoon scheduling

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

NORTHERN ARIZONA UNIVERSITY

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, 30 day retention
  - Very fast storage, capable of 11GB/sec
  - Checkpoints, logs
  - Keep all temp/intermediate data here
  - Should be your default location to perform input/output
  - Scratch will be upgraded to 500TB in Fall 2017

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@monsoon.hpc.nau.edu:/scratch/nauid
  - WinSCP (windows)
  - Cyberduck (mac)
- samba / cifs
  - \\nau.froot.nau.edu\cirrus (windows)
  - smb://nau.froot.nau.edu/cirrus (mac)

Software

- ENVI / IDL
- Matlab
- Mathematica
- Intel Compilers, and MKL
- R
- SAS
- Qiime, and Qiime2
- Anaconda Python
- Lots of bioinformatics programs
- Request additional software to be installed!

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

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?

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 --workdir=/scratch/nauid   # default location slurm searches

- # replace this module with software required in your workload
- module load python/3.3.4

- # 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
Add MPI tasks/ranks to your job	--ntasks=2, or -n 2
MPI ranks per node	--ntasks-per-node
MPI ranks per socket	--ntasks-per-socket
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

Interactive / Debug Work

- You are logged into a ”login node”
  - The login node is meant for preparing job scripts, and debug work
  - Ok to run commands for less than 30 minutes
- Debug, and interactive use
  - srun --qos=debug Rscript analysis.r
  - srun --qos=debug --mem=6000 a.out
  - srun -c2 --mem=2000 analysis.py
- The debug qos provides high priority to get jobs started quickly
  - Limited to 30 minutes, and 1 job per user

Long Interactive work

- salloc
  - Obtain a SLURM job allocation that you can work with for an extended amount of time. With an allocation, srun commands are run instantly against this allocation.

```
 [user1@wind ~ ]$ salloc –c 2 --time=5:00:00
 salloc: Granted job allocation 33442
[user1@wind ~ ]$ srun python analysis.py
[user1@wind ~ ]$ exit
salloc: Relinquising job allocation 33442
[user1@wind ~ ]$ salloc -N 2
salloc: Granted job allocation 33443
[user1@wind ~ ]$ srun hostname
cn3.nauhpc
cn2.nauhpc
[user1@wind ~ ]$ exit
salloc: Relinquising job allocation 33443
```

Submit the script

```
[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)

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

Controling 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.
- scontrol show job 85224

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 alias “jobstats” instead
  - jobstats -r
  - jobstats –j <jobid>

Exercise 1

Get to know monsoon and Slurm, on your own.

1. How many nodes make up monsoon?
   – Hint: use “sinfo”
2. How many nodes are in the all 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 2

- Create a simple job in your home directory
- Example here: /common/contrib/examples/job_scripts/simplejob.sh (copy it if you like ☺)
- Name your job: “exercise”
- Name your jobs output: “exercise.out”
- Output should go to /scratch/<user>/exercise.out
- Load the module “workshop”
- Run the “date” command
- And additionally, the “secret” command
- Submit your job with sbatch, i.e. “sbatch simplejob.sh”

Exercise 3

- Edit the jobscript from previous exercise 2
- Make your job sleep for 5 minutes (sleep 300)
  - Sleep is a command that creates a lazy process that … sleeps and does nothing
- Monitor your job
  - squeue -u your_nauid
  - squeue -t R
  - scontrol show job jobnum
  - sacct -j jobnum
    - Inspect the steps

- Cancel your job
  - scancel jobnum

Exercise 4

- Copy job script and edit:
  - /common/contrib/examples/job_scripts/lazyjob.sh
- Edit the job with your user id
- Submit the job, it will take 65 sec to complete
- Use sstat and monitor the job
  - sstat -j <jobid>
- Review the resources that the job used
  - jobstats -j <jobid>
- We are looking for "MaxRSS", *MaxRSS is the max amount of memory used*
- Edit the job script, 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 -j <jobid>
- 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 -j <jobid>, 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!

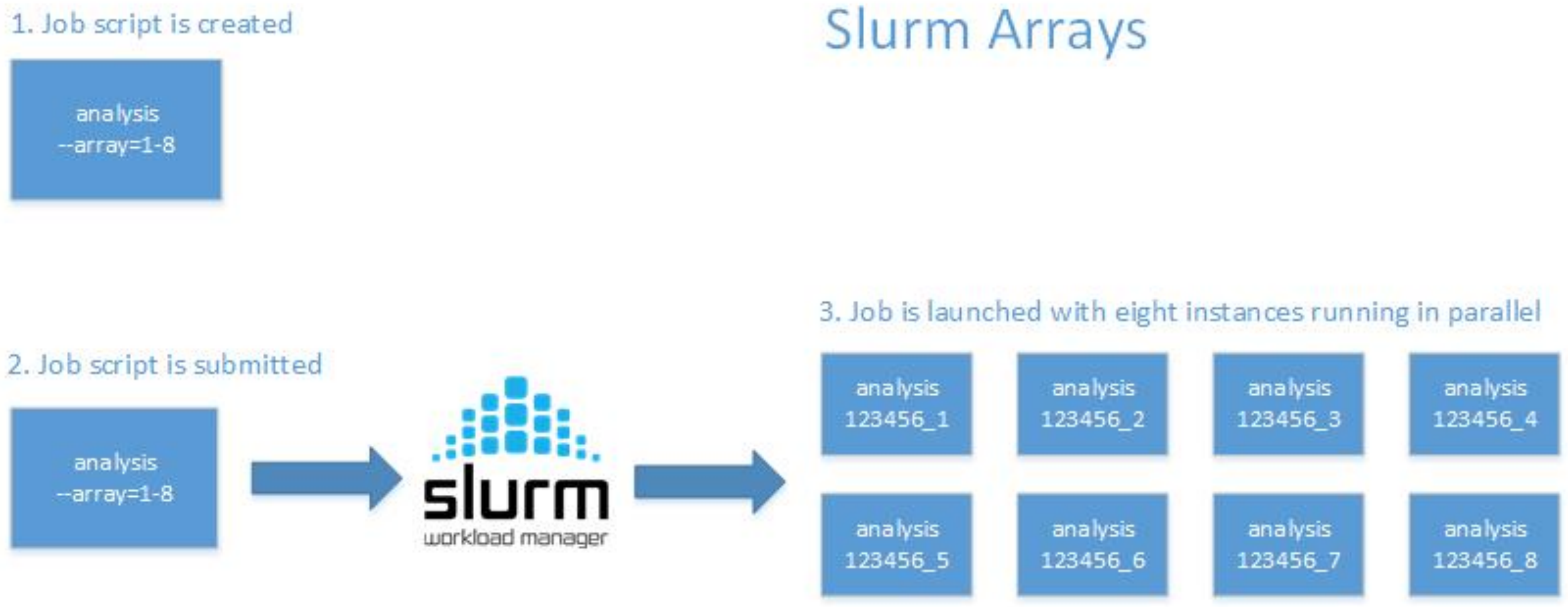
Keep these tips in mind

- 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 effect you and your group negatively

Question and Answer

- More info here:
  http://nau.edu/hpc
- Linux shell help here:
  - http://linuxcommand.org/tlcl.php
  - Free book download
- Intro to Linux command line video:
  - https://www.lynda.com/Linux-tutorials/Welcome/435539/482226-4.html
  - Info here: https://nau.edu/HPC/Linux-External-Resources/
- And on the nauhpc listserv
  - nauhpc@lists.nau.edu

Slurm Arrays



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 “squeue”, notice there are 5 jobs running, how did that happen!

NORTHERN ARIZONA UNIVERSITY