Advanced Research Computing (ARC) Training

Matthew Brown/Ayat Mohammed/Chris Kuhlman/Sarah Ghazanfari/Sofia Lima

Computational Scientists

Advanced Research Computing (ARC), Division of Information Technology

Virginia Tech

in collaboration with **other ARC staff members** and **GRAs**

03-05 June 2024
Summer 2024

Running Code/Software on ARC Systems in Different Ways

Chris Kuhlman

ckuhlman@vt.edu

Computational Scientist

Advanced Research Computing (ARC), Division of Information Technology

Virginia Tech

in collaboration with **other ARC staff members** and **GRAs**

Tuesday, 04 Jun 2024
Summer 2024 Series

Sign Up Sheet: Please Sign Up

1. Google sign up sheet is here:
    - A. https://docs.google.com/document/d/1uVrupbvN6-2ZsxOFzokp_gLWAaeGocP2/edit
2. Please sign in to ensure:
    - A. You get credit for the course
    - B. Our roster is complete
3. Also, this google sheet has
    - A. Commands that we are going to execute together.
    - B. Link for these slides
    - C. Space for feedback

Advanced Research Computing (ARC) Trainings, Summer 2024

via Zoom video conferencing

Monday

- 06/03*: Introduction to Advanced Research Computing (SG)

  Basics of HPC, computer clusters, HPC resources, access to ARC systems

- 06/03*: Connect to ARC Systems and Run your first jobs (AM)

  Connect via Open OnDemand, connect via SSH, cluster and scheduler orientation, run demo jobs

Tuesday

- 06/04*: Running code/software on ARC systems in different ways (CK)

  Job environments (modules and Conda), running interactive and batch jobs

- 06/04*: Launching Jobs in Parallel on ARC Clusters (AM)

  MPIRUN vs. SRUN, GNU parallel for load balancing, SRUN for resource detection and binding, "Built-in" or library-based parallelism

Wednesday

- 06/05*: Monitoring Resource Utilization and Job Efficiency (CK)

  Acquiring resources, characteristics of compute nodes, overall activity, current loads, job status

Get These Slides

- Slides are available from within here
    A. https://docs.google.com/document/d/1uVrupbvN6-2ZsxOFzokp_gLWAaeGocP2/edit

Resources

- ARC documentation
  - https://www.docs.arc.vt.edu/
  - READ THIS (No joke; there is vocabulary, computing resources, etc. Can save a lot of time.)
- Get an account on ARC
  - https://arc.vt.edu/account
- Get a project on ARC (lot more storage)
  - https://coldfront.arc.vt.edu
- Help Desk
  - https://arc.vt.edu/help
- Office hours for with GRAs
  - https://arc.vt.edu/office-hours

Context, Goals, Feedback

- Context
  - This is an informal workshop
  - Mostly informational about ARC and research computing at VT
  - For new students, faculty, staff, researchers. And anyone else.
- Goals
  - Run jobs in real environments
  - Learn job submission types
  - You will have the code and commands; you can repeat and/or modify at your leisure.
- We want to hear your questions
  - Just interrupt the talk
  - Welcome to use chat to ask questions + some time at the end
- Feedback needed to help improve future workshops. ***PLEASE***
  - One up / one down at the end
  - More detailed feedback

First Thing’s First (Thanks Ayat)

VPN needed for connections from off-campus

- https://www.nis.vt.edu/ServicePortfolio/Network/RemoteAccess-VPN.html
- Nearly all ARC services require being on the campus network or VPN
- Use “VT Traffic over SSL VPN” connection
- ColdFront (accounting system) available with or without VPN

Get an ARC account:

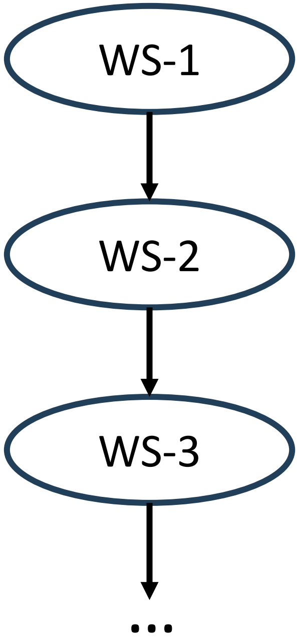
- https://coldfront.arc.vt.edu/account/create
- Acceptable Use Policy

Get Your Code/Software to Run on ARC Systems

- ARC systems run software that spans the full spectrum of modern research computing.
- This workshop addresses some of the most common software delivery models and how they can be accessed and used on ARC systems.
- The demonstrations will be predominantly via the Linux shell command line interface and will cover our "software modules" system and python environments via Anaconda.
- Participants who already have ARC accounts are invited to follow along with the demonstrations if desired.

There is a Flow to these Workshops

- We are moving from the general to the specific.
- Workshops (WS) 1 and 2, i.e., WS-1 and WS-2, were given the day before this one, on 03 Jun 2024.
- This in-progress workshop is WS-3.



WS-1

Overview of ARC, human resources, computing resources, what we do.

WS-2

Resources, how to create accounts, documentation, interactive systems, OOD, ssh, command line, a first job.

WS-3

Weeds of running jobs, serial jobs, from terminal and command line.

…

Outline

- Two operational models.
- Use of head nodes (or login nodes).
- Five exercises using
  - Environments
    - Modules
    - Conda envs
  - Codes
    - Python commands
    - Python codes
  - Job types
    - Interactive
    - Batch

HPC Resources at ARC/VT

We use
TC today

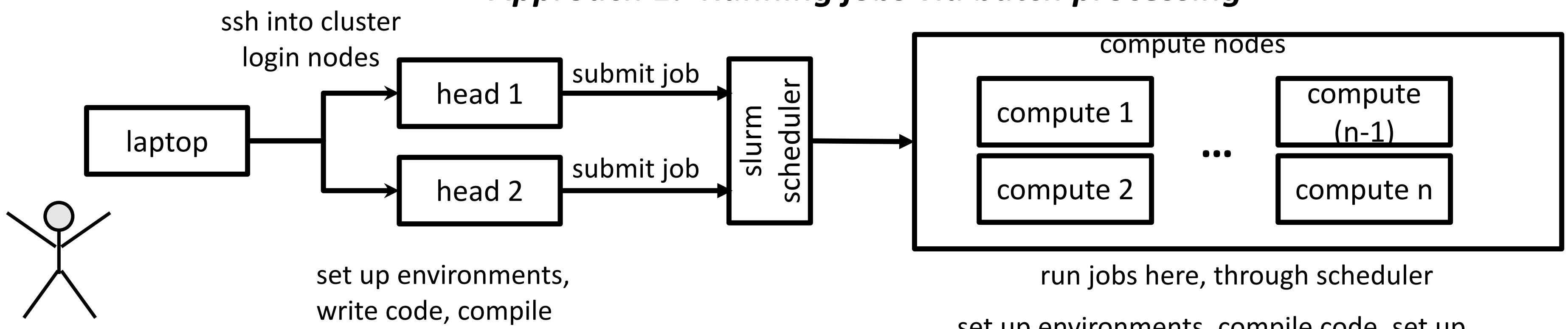
Cluster	Description	Since
CUI	Dense GPU + some CPU for projects with controlled data/software	c. 2021
Tinkercliffs	HPC/HTC Flagship CPU HPE Dense GPU nodes (A100) DGX Dense GPU nodes (A100)	c. 2020 c. 2021 c. 2022
Infer (nearing end of life)	Accelerating inference and ML workloads (T4 GPU) Added P100 GPUs from Newriver Added V100 GPUs from Cascades	c. 2021 c. 2016 (EOL) c. 2018 (EOL)
OWL (coming soon)	Water-cooled latest generation AMD CPU high mem-per-core DDR5	c. 2024
Falcon (later in 2024)	GPU node expansion L40S GPUs (20 nodes x4 GPUs) A30 GPUs (32 nodes x4 GPUs)	c. 2024

Operational Models

- We are going to review briefly the overall system hardware and your laptop or tower.
  - This will help us to use codes and system (shell, directives) software to set up jobs and run jobs.
  - Different pieces of the software run on different pieces of the hardware.
- If you get these concepts down, your life will be much easier, going forward, in all sorts of ways.
- Real examples and real code require these ideas.

High Level Operating Environment

Approach 1: Running jobs via batch processing



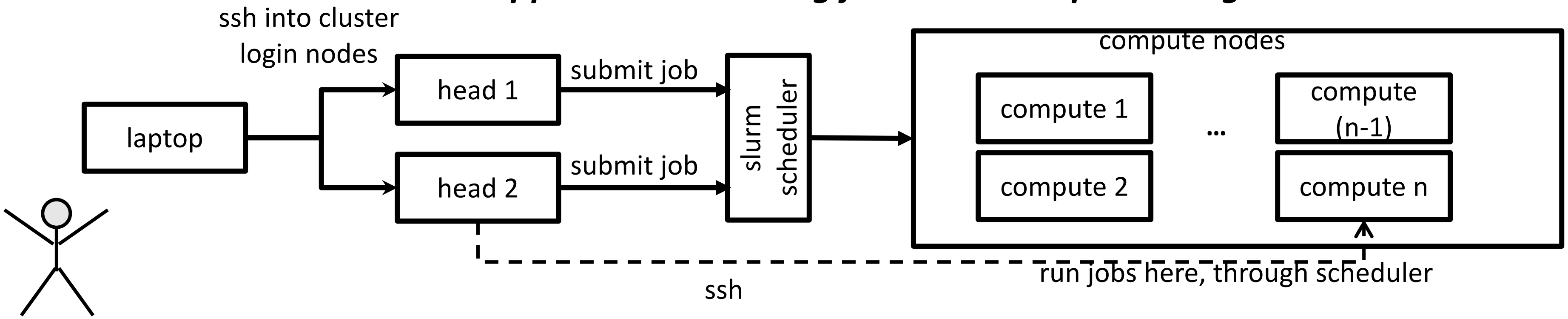
set up environments, write code, compile code, set up directories, write configuration files, create scripts, move files, submit jobs.

Do not run jobs on the head nodes, puhleeeze.

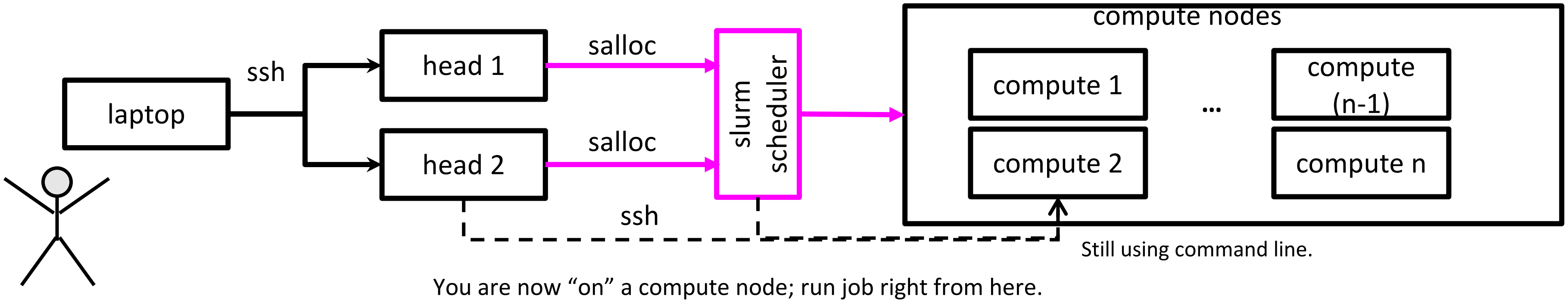
set up environments, compile code, set up directories, write configuration files, create scripts, move files, submit jobs.

High Level Operating Environment

Approach 1: Running jobs via batch processing



Approach 3: Running interactive jobs



Still using command line.

You are now “on” a compute node; run job right from here.

Acceptable Uses of Login (Head) Nodes

shared resources

1. Navigating around the hard drive.
2. Understanding files (e.g., their existence, size, permissions, whether data or code, file formats).
3. Understanding directory structures.
4. File/directory management (e.g., create a new directory, change permissions on a file, moving files, copying files, deleting directories).
5. Creating, editing, delete files.
6. Moving files onto, and off of, a cluster.
7. Creating build and execution environments in which you construct your software and run it.
8. Compiling source code.
9. …

Login nodes *NOT* for running jobs. Please don't do it; causes lots of problems for other users. We want to be friendly.

Log In To Tinkercliffs

- Start vpn as you did yesterday, or have done, if you are off-site.
- Open a terminal window (preferably 2 or 3) on your tower or laptop.
- Use ssh, in that terminal window, to connect to tinkercliffs, typing:
- ssh <user-name>@tinkercliffs1.arc.vt.edu
- Enter password; PID password
- You are now on the tinkercliffs head node 1.

- Repeat this in another (different) terminal window.
- You will have two terminal connections to tinkercliffs.

Exercises Matrix

		Environment		Code Types		Job Types	
Exercise No.	**Relative Directory**	**Modules**	**Conda Envs**	**Python Commands**	**Python Code**	**Interactive Job**	**Batch Job**
1	exercise01	🟩	🟩			🟩	
2	exercise02	🟩	🟩	🟩		🟩	
3	exercise03	🟩	🟩		🟩	🟩	
4	exercise04	🟩			🟩		🟩
5	exercise05	🟩	🟩		🟩		🟩

There is a "commands-to-execute" file in each directory.

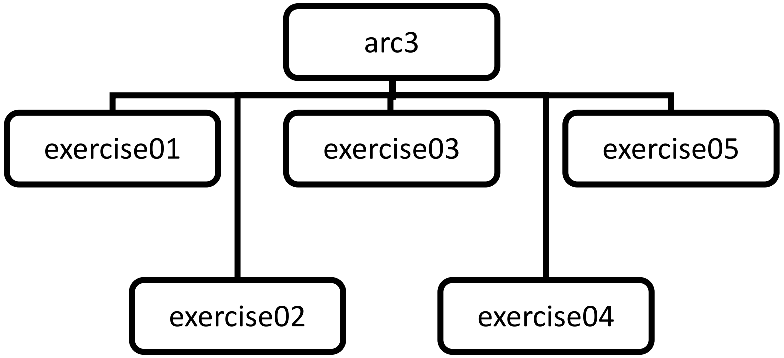
*Do "**cat commands-to-execute**" and then you can copy and paste commands (text) onto the command line to execute commands.*

The commands are also in the google doc signup sheet.

Notes on Exercises

- There are some dependencies among exercises. This is good.
- Each exercise has this approach
    - We start from the beginning (from scratch). We go in.
    - We do our work.
    - We come out.
- Some of these exercises could be combined and done all at one time.
- But we want you to see how you start “clean” each time and exit “clean” each time.
- Our starting point for each exercise is as if we just logged onto the cluster.

Locations of Codes on TC---Once We Get the Files Per This Slide



- ## After ssh'ing into Tinkercliffs …
- ## go to your home directory on TC
- cd
- ## create a new directory
- mkdir arc3-ws
- ## change directory to this new directory
- cd arc3-ws
- ## copy tarball from /globalscratch on TC (copy through the ".")
- cp /globalscratch/ckuhlman/arc-workshops-mar-2024/arc3.pres.exercises.final.tar.gz   .
- ## expand the contents of the tarball.
- ## this will create new directories and put files into them.
- tar xvzf arc3.pres.exercises.final.tar.gz
- ## the directory names are the exercise numbers.
- ## files have commands to execute (you can copy and paste them) and code that we will execute.
- cd arc3
- ls –lrt *

Use of Interactive (Shell) Jobs

1. Test a code to see that it runs on very small-sized inputs before submitting [many, large] jobs.
2. Run some small scripts (to be kind to other users on head node).
3. Set up Python virtual environments.
4. Run a series of very small jobs, for first time.
5. Other

Anaconda (Conda) Environments for Python

this is for reference

Standard steps:

Get interactive job on a node of the correct type; here, a GPU node.	interact --partition=dgx_dev_q –gres=gpu:1 --account=personal --time=0:10:00
Get interactive job on a node of the correct type; here, a multicore node.	interact --partition=dev_q --account=personal --time=2:00:00
Load an Anaconda module and other needed modules	module reset module load Anaconda3 (optional) module load foss
Create an environment at a path	conda create -p ~/env/tcdgx/tf
Activate the environment	source activate ~/env/tcdgx/tf
Install packages into the environment	conda install python=3.9 tensorflow

One Way to Organize Conda Envs

- There are multiple machines with the ARC clusters family
  - Tinkercliffs (TC)
  - Infer
  - Owl
  - Falcon (soon-ish)
- Compute nodes on each machine varies, too.

- But you have only one “home” space where all of your files reside.
- So it is useful to create a hierarchical structure into which conda envs will be written.
- Compute nodes on Tinkercliffs (TC)
  - Multicore nodes
  - GPU nodes
    - A100
    - DGX

Format:
/home/uname/env/machine/nodeType/computeNode/envName

Examples:
/home/ckuhlman/env/tc/gpu/a100/py311_tensorFlow
/home/ckuhlman/env/tc/gpu/dgx/py311_tensorFlow
/home/ckuhlman/env/tc/cpu/py312_kiran01
/home/ckuhlman/env/tc/cpu/py39_snap
/home/ckuhlman/env/owl/hugemem_q_milan/cpu/py312_kiran01
/home/ckuhlman/env/owl/cpu/py312_kiran01

Exercise 1: Create a Conda Python Env

- Preliminaries
  - env is generated for TinkerCliffs cluster, cpu nodes, basic env
  - must create the env on the nodes that you want to run on.
  - make directory structure reflect these details (to maintain control).
- Operations for the audience to do
  - cd exercise01
  - cat commands-to-execute
  - interact --partition=dev_q --account=personal --time=2:00:00
  - ## You must wait on the above command to complete, so you have an interactive session.
  - module reset
  - module load Anaconda3/2020.11
  - conda create -p ~/env/tc/cpu/py39_base
  - source activate ~/env/tc/cpu/py39_base
  - conda install python=3.9
  - conda deactivate
  - exit
- Confirm:
  - cd ~/env/tc/cpu/py39_base
  - ls

Exercise 2: Run Python Interactively on a Compute Node

- cd ../exercise02
- cat commands-to-execute
- interact --partition=dev_q --account=personal --time=2:00:00
- module reset
- module load Anaconda3/2020.11
- source activate ~/env/tc/cpu/py39_base
- python --version
- python
- print("Hi. The audience is great")
    - Output: Hi. The audience is great
- print("Hello world. Using interactive nodes, anaconda module, a python env")
    - Output: Hello world. Using interactive nodes, anaconda module, a python env
- exit()
- conda deactivate
- exit

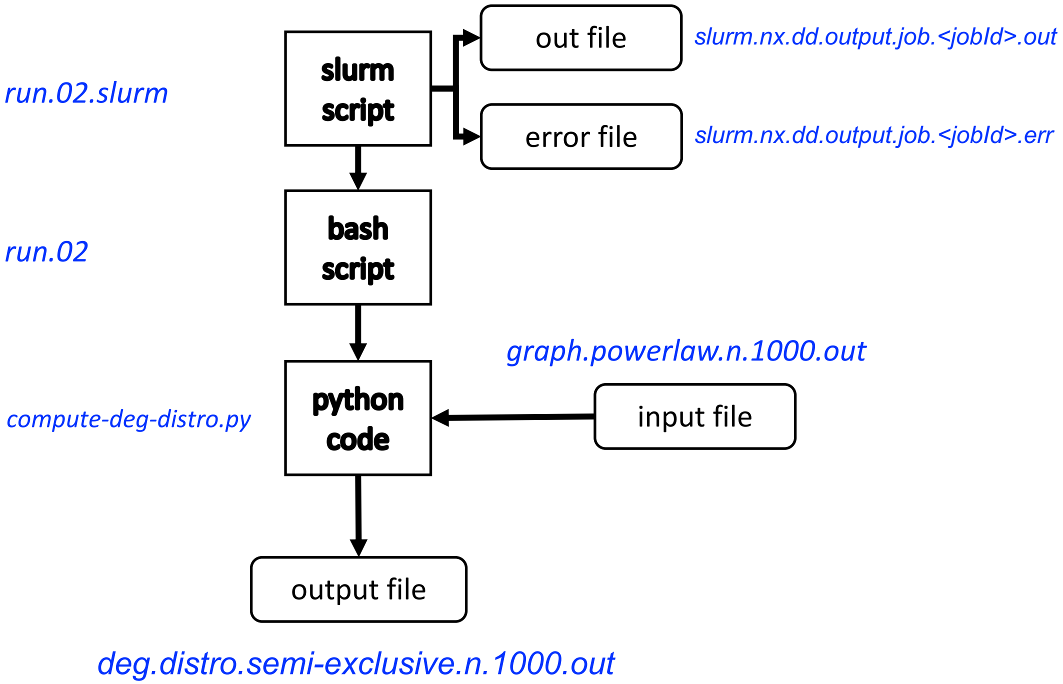
Exercise 3: Run Python Code Interactively on a Compute Node

- cd ../exercise03
- cat commands-to-execute
- interact --partition=dev_q --account=personal --time=2:00:00
- module reset
- module load Anaconda3/2020.11
- source activate ~/env/tc/cpu/py39_base
- python --version
- python simple-py-01.py
- conda deactivate
- exit

Exercise 4: Run a Python Code In Batch Mode Using Slurm

- Commands to run job:
  - cd ../exercise04
  - sbatch run.02.slurm
  - Note the unique ID that slurm returns to you.
- That’s it; slam dunk.
- How to check status of job
  - squeue –u <your-user-name>
  - Issue this again and again (up-arrow)
- Output file: deg.distro.semi-exclusive.n.1000.out

What is going on here?
At command prompt, type “ls”
Now, match the filenames with diagram below.



run.02.slurm

slurm script

out file

slurm.nx.dd.output.job.<jobId>.out

error file

slurm.nx.dd.output.job.<jobId>.err

run.02

bash script

graph.powerlaw.n.1000.out

compute-deg-distro.py

python code

input file

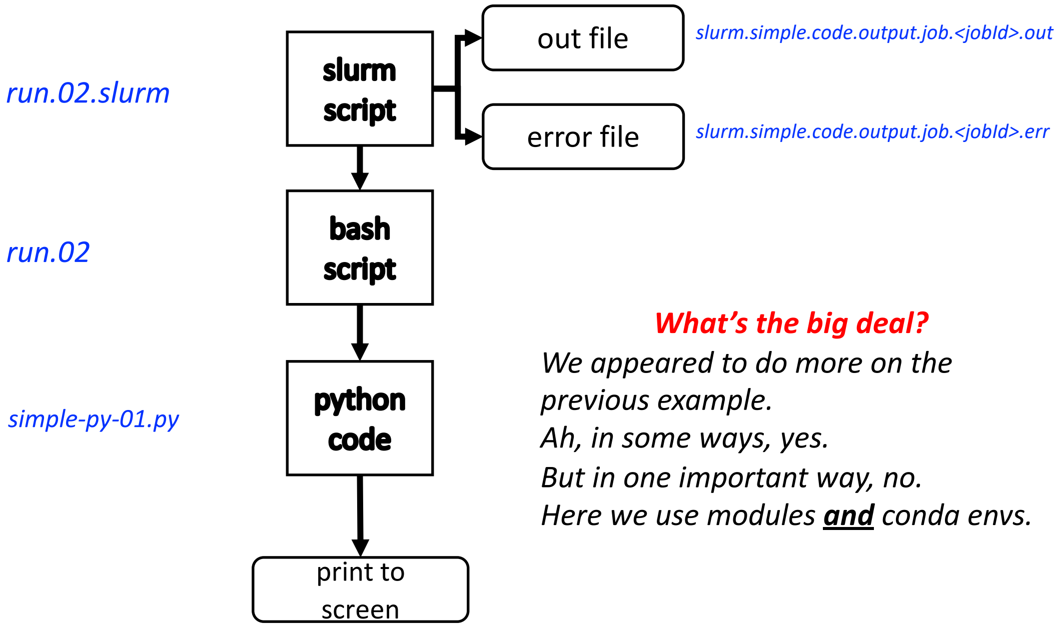
output file

deg.distro.semi-exclusive.n.1000.out

Exercise 5: Run a Python Code In Batch Mode Using Slurm

- Commands to run job:
  - cd ../exercise05
  - sbatch run.02.slurm
  - Note the unique ID that slurm returns to you.
- That's it; slam dunk.
- How to check status of job
  - squeue –u <your-user-name>
  - Issue this again and again (up-arrow)

What is going on here?
At command prompt, type "ls"
Now, match the filenames with diagram below.



What's the big deal?
We appeared to do more on the previous example.
Ah, in some ways, yes.
But in one important way, no.
Here we use modules ***and*** *conda envs.*

What is Slurm Doing For Us?

- Slurm enables us to encapsulate all sorts of directives about the job, how to set up the job environment, how to run it, in one file.
- Example at right.
- Match the colored text between the two exercises.

Exercise 03

- interact --partition=dev_q --account=personal --time=2:00:00
- module reset
- module load Anaconda3/2020.11
- source activate ~/env/tc/cpu/py39_base
- python --version
- python simple-py-01.py
- conda deactivate
- exit

Exercise 05

File: run.02.slurm

```
#!/bin/bash
#SBATCH -J sc
## Wall time.
#SBATCH --time=01:00:00 # 1 hour
#SBATCH --account=personal
### Other queues are:  a100_normal_q,  dgx_normal_q
#SBATCH --partition=normal_q
## Use the compute node only for this job, and use all memory on this node.
## #SBATCH --exclusive
## #SBATCH --mem=0
## Slurm output and error files.
#SBATCH -o slurm.simple.code.output.job.%j.out
#SBATCH -e slurm.simple.code.output.job.%j.err
# Load modules.
module load Anaconda3/2020.11
# Activate a python env.
source activate ~/env/tc/cpu/py39_base
# Code to execute.
./run.02
```

Anaconda Environments for Python

Other Info to Know

- Miniconda is smaller, uses same package manager (conda)
- Create environments once, not repeatedly in batch jobs
- For intensive program/library I/O, consider /globalscratch or $TMPFS
- Environments can be loaded in OnDemand Jupyter notebooks
- Specify only the most essential packages/versions when installing and let conda do the rest.

Thanks for Participating

- Google sign up sheet is here:
  - https://docs.google.com/document/d/1uVrupbvN6-2ZsxOFzokp_gLWAaeGocP2/edit
- Please sign in to ensure:
  - You get credit for the course
  - Our roster is complete

Acknowledgments

- Matt Brown developed an earlier version of this presentation, which heavily informs this work.
- Thanks to Ayat Mohammed and Sarah Ghazenfari for some overview slides, which I stole.

END