HPC Tutorial

CWRU SIAM Student Chapter
Date: October 5th



Case Department of Mathematics,
Applied Mathematics, and Statistics
CASE WESTERN RESERVE
UNIVERSITY

SIAM Student Chapter

… in case you came for free food, you can also get a free membership

In case you are not part of SIAM, you can join here!

→ discounted textbooks

→ reduced prices to attend SIAM conferences

Members Benefits

Members get discounts on SIAM textbooks. Members also have access to SIAM Early Career Awards as well as discounts on conference registrations.



Events & Activities

Be the first to know about what we have planned and add our group calendar to your schedule.



Exclusive Resources

Get our newsletter and stay in the loop.



Connnections

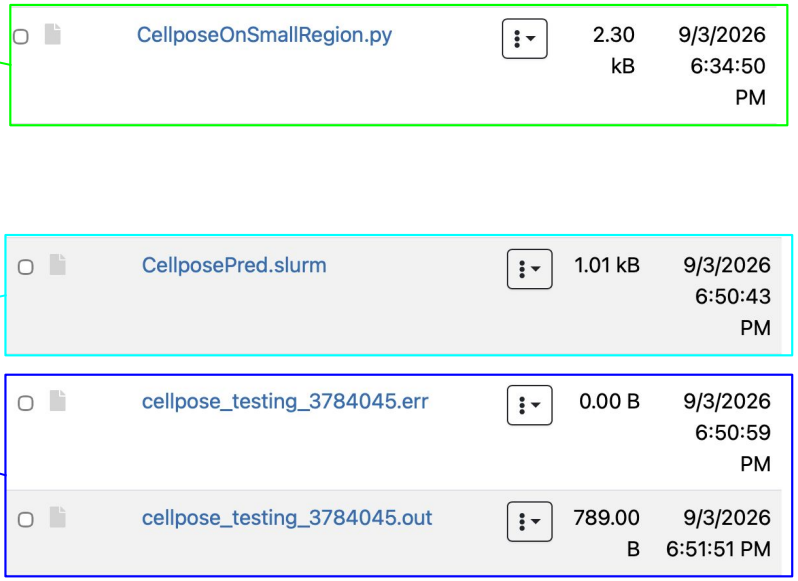
Meeting new students with shared interest

HPC Overview

1. Big picture
2. General Resources
3. How to access HPC
4. Adding scripts to HPC
5. Login vs. compute nodes
6. Submitting jobs (interactive / login node, slurm / compute node)
7. Virtual environments
8. Other helpful commands
9. Summary

Big Picture: Steps to Running Code on HPC
(it's ok if none of this makes sense right now)

1. upload script to HPC
2. test small problem (*) to ensure code runs
3. create slurm script
4. submit batch job to HPC and wait for output



* short run time, low memory tests can be run on login node

General Resources

CWRU HPC home
- all HPC details in one place

HPC good conduct
- guidelines for shared resources

Quickstart
- can access OnDemand, online access to cluster

Batch jobs
- slurm codes / using compute nodes

The CWRU HPC cluster is designed to **support the active work of dozens, even hundreds, of researchers at a time.** Please go through Acceptable User Policy (AUP) for High Performance Computing. For campus-wide AI policy and other policies, please refer to Campus Website.

Key operations that promote or inhibit efficient resource usage can occur

- **within the compute nodes** (cpu/gpu/mem operations);
- when **data input/ouput** occurs (file system ops); and,
- in highly-shared resources such as the **login nodes**, or the **Slurm scheduler node.**

The approach that we encourage is two-fold.

1. **Have a good understanding of your own workflow**, including how the software uses memory, cpu and gpus, and file transfers.
2. **Develop operational practices** that work with the HPC architecture.

Login nodes caution

Running jobs

Please DO NOT use the login node for running your jobs. The login nodes serve dozens of people at a time, and computational work would take too many resources for one person's use. See Job Scheduling for more information. **AI agents are no longer appropriate to use on login nodes, as they often consume compute resources, that may degrade performance and stability of the login nodes.**

Compiling on a head node (limit '-j <>')

Compiling packages on the login nodes is sometimes necessary, as these nodes tend to have a more complete development environment. Lengthy builds should be done on compute nodes. In the case of those packages that support parallel builds, limit the number of processes to 4: e.g. make --jibs=4

Use caution polling the system

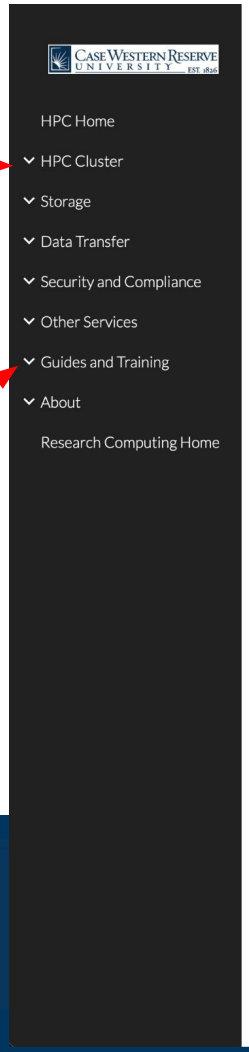
Understanding the state of jobs and the available cluster resources is an important aspect of managing your work. Sometimes, you will set up scripts to run to monitor conditions.

HPC Resources

CWRU HPC website

User guides, good conduct and OnDemand

Extra resources and training guides



Case Western Reserve HPC



Welcome to the HPC Clusters at CWRU

- **Research clusters: pioneer.case.edu**
- **Coursework cluster: markov.case.edu**
- **HPC Good Conduct & Acceptable Use Policy for High Performance Computing**
- **Besides the sidebar menu, you can also do keyword search via the search icon on the top right corner.**
- **You can also ask AI/Search Engine about the HPC by adding keyword "CWRU HPC" in the prompt.**

Introduction to Case Western Reserve HPC

Computationally intensive research is supported through the operation of High Performance Computing (HPC) resources at CWRU. Services include provisioning user access to our computational resources, assistance in optimizing use through training and direct support, and providing answers to questions regarding daily HPC activities. HPC services are offered by the [U]Tech's Research Computing and Cyberinfrastructure (RCCI) group. Please let us know if you need assistance in pre-award activities related to budgeting and how we can help further your research goals.

Access to the cluster is restricted, and available from the campus network or through VPN (vpnsetup.case.edu), if connecting from outside campus.

HPC Acknowledgement

If you have utilized the HPC resources for your publications, please **acknowledge HPC** in your HPC-Assisted research works as showed:

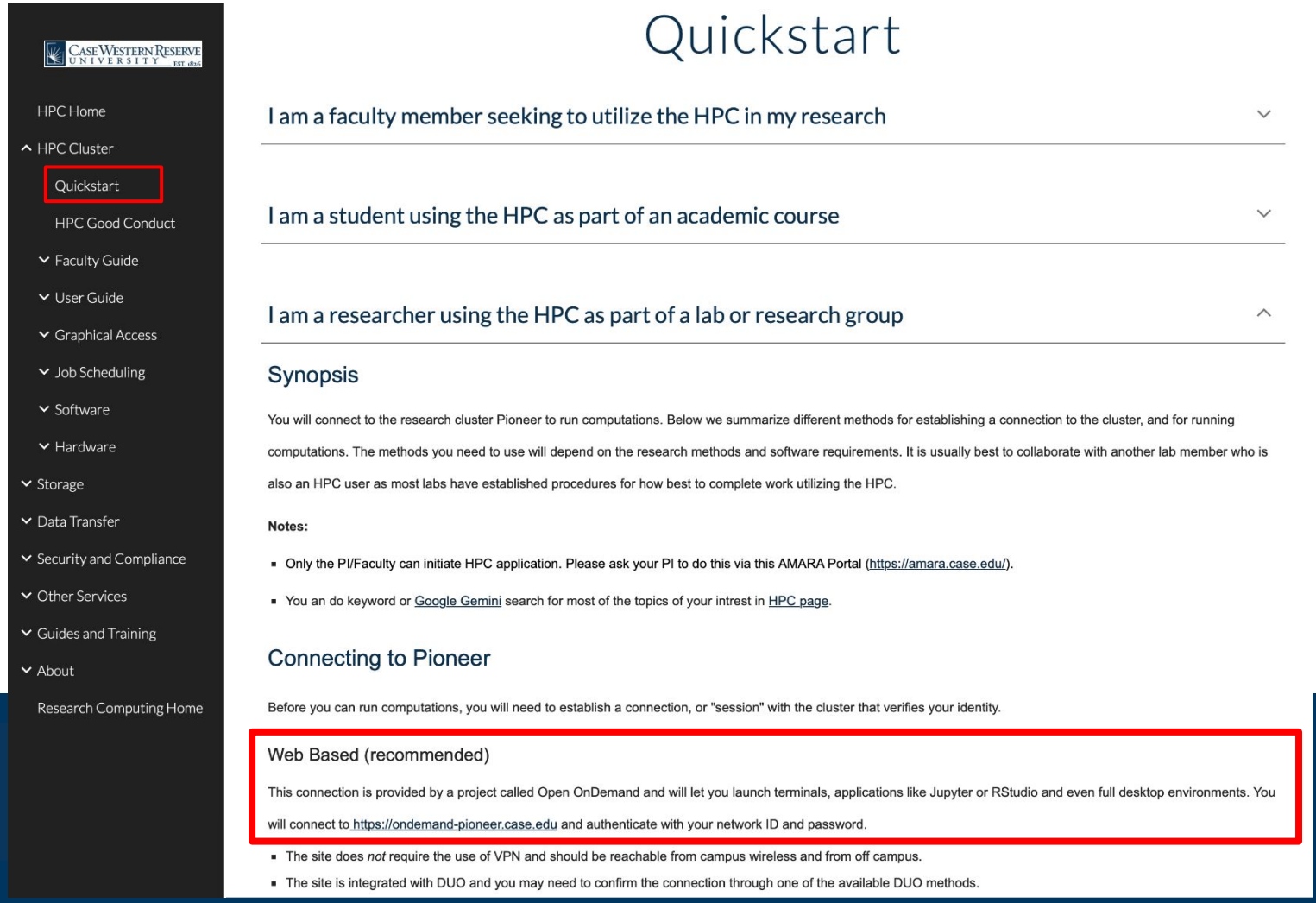
"This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University."

New to HPC?

- The best place to start is our general Quickstart.
- For faculty members, you can find additional information about setting up your account and membership at the Faculty Guide page.
- For staff or students, you can find additional information under the User Guide page.

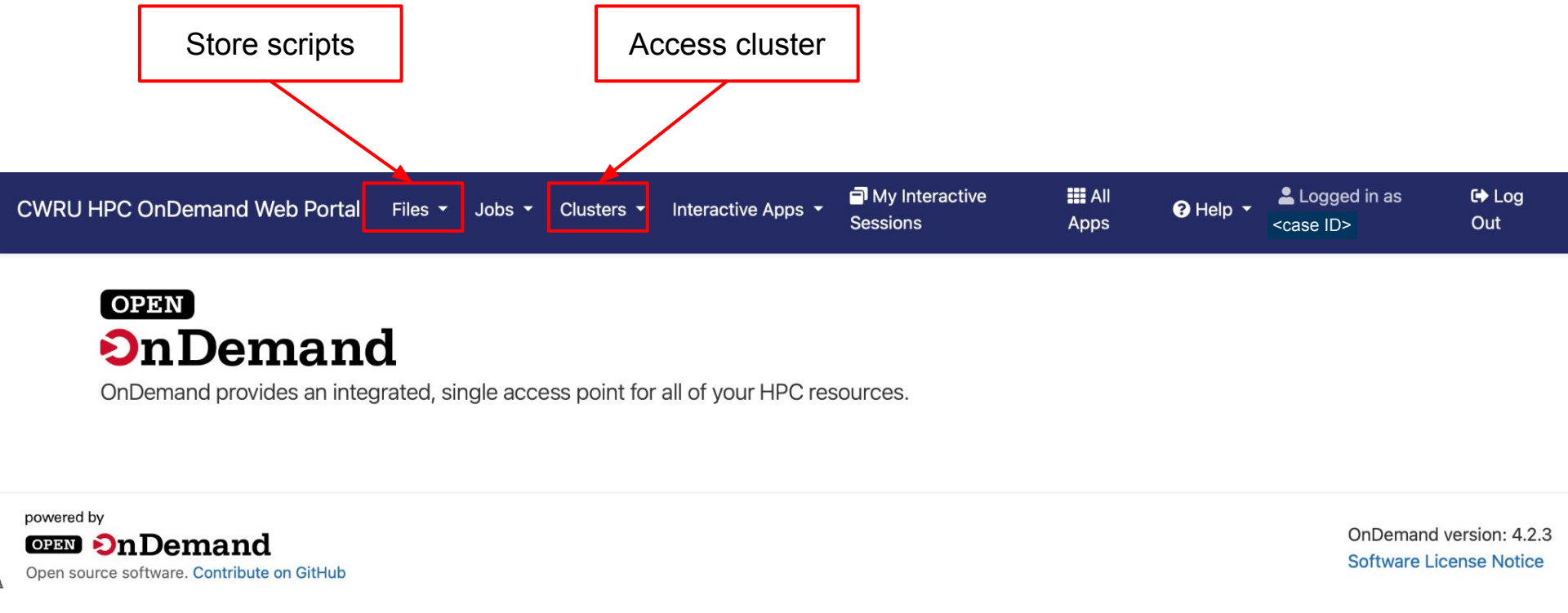
Accessing the HPC

Recommended: OnDemand (HPC web access through Quickstart)



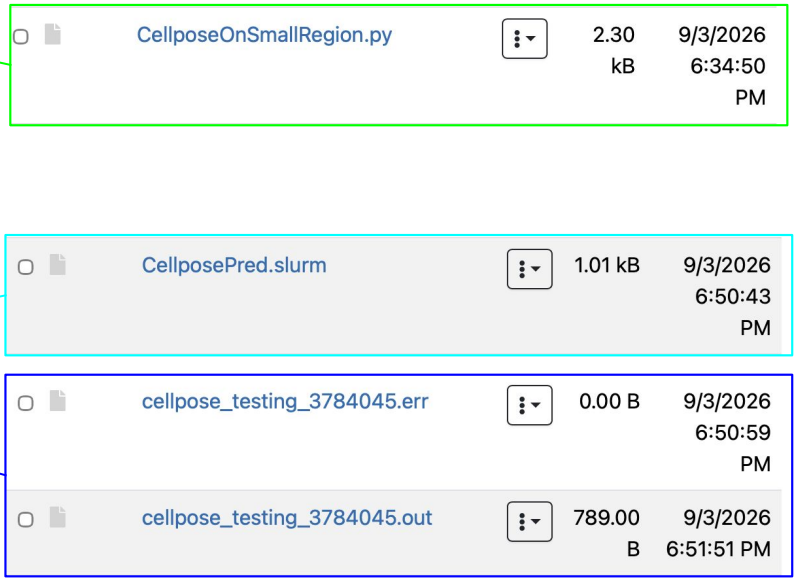
Accessing the HPC: OnDemand

OnDemand home screen



Steps to Running Code on HPC: Adding Scripts to HPC

1. upload script to HPC
2. test small problem (*) to ensure code runs
3. create slurm script
4. submit batch job to HPC and wait for output



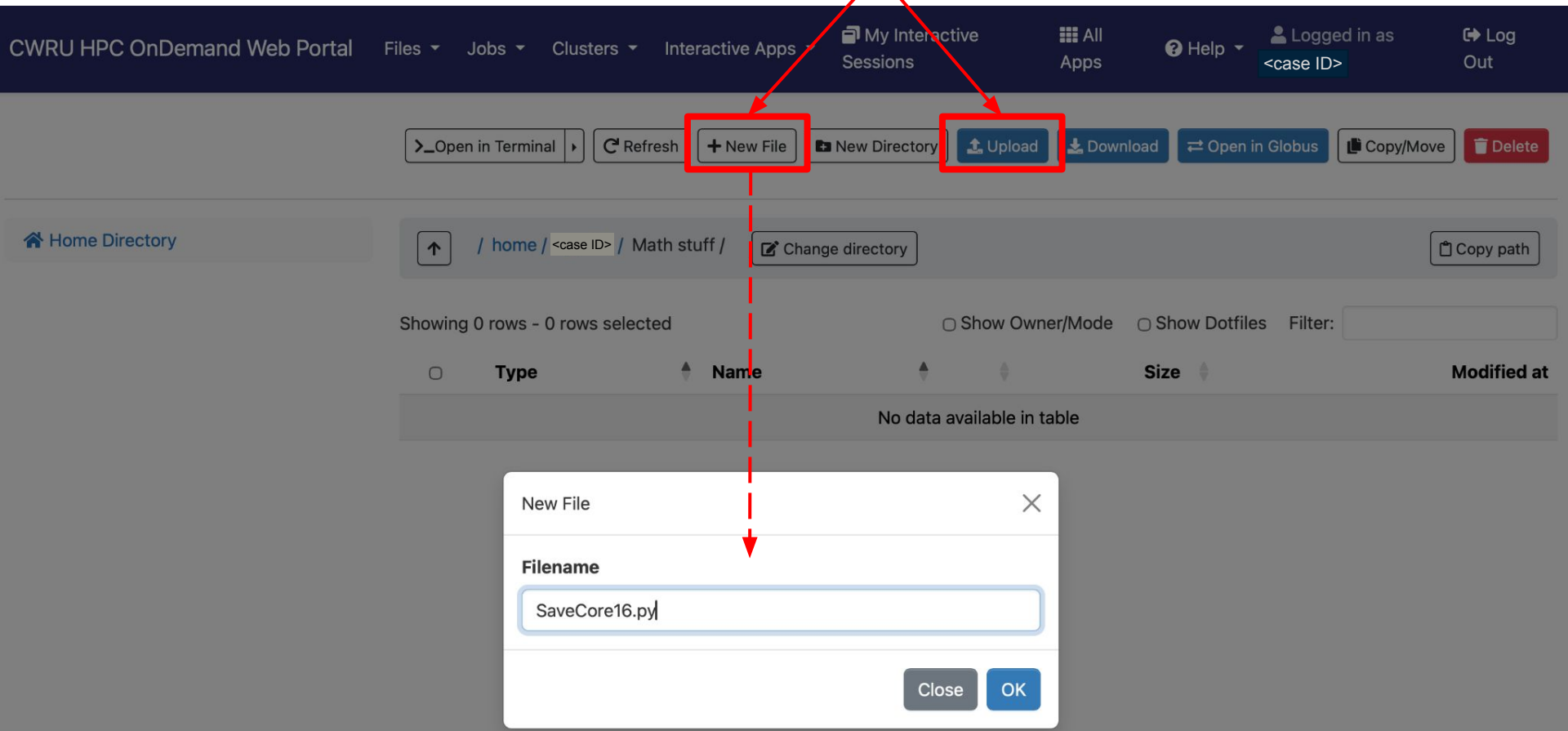
* short run time, low memory tests can be run on login node

Adding Scripts to HPC

Add scripts to HPC by uploading files or creating file directly

Add scripts to home directory

Create or upload a new script file



Login vs. Compute Nodes

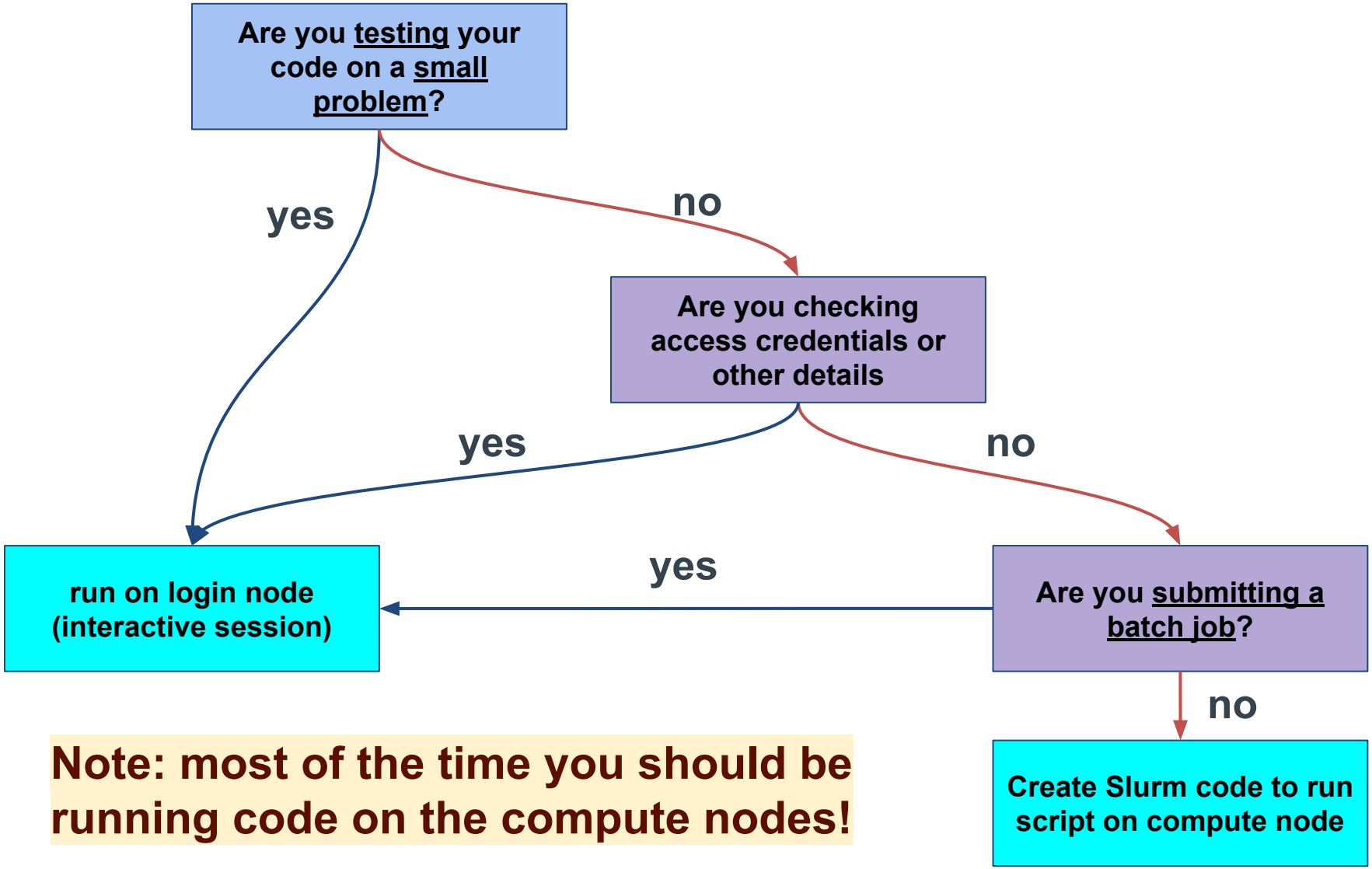
Login nodes

- “interactive session”
- can be used to do small tests to make sure code runs
- submitting slurm codes
  - sends code to be run on compute node

Compute nodes

- use for running scripts that need more compute resources
- managed resources
  - wait time depends on available resources, time needed for code to run and memory needed

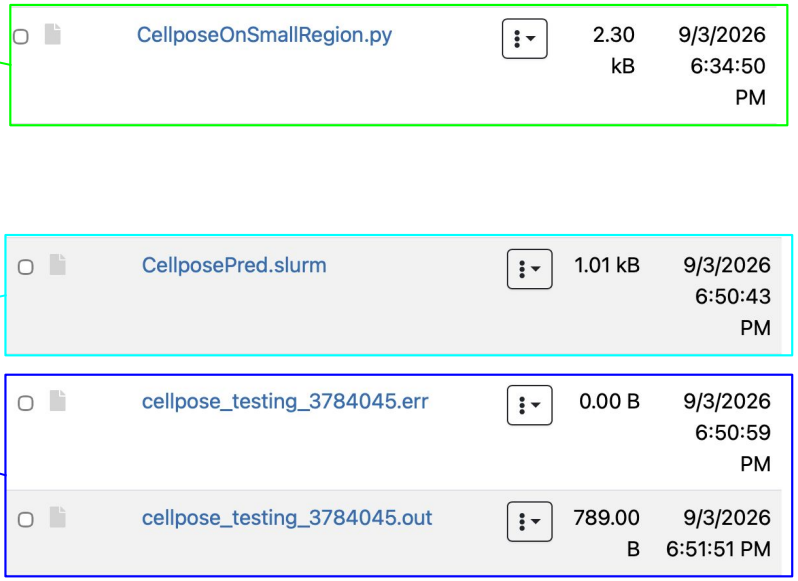
Login vs. Compute Nodes



Note: most of the time you should be running code on the compute nodes!

Steps to Running Code on HPC: Creating Slurm Codes

1. upload script to HPC
2. test small problem (*) to ensure code runs
3. create slurm script
4. submit batch job to HPC and wait for output



* short run time, low memory tests can be run on login node

Slurm Code: General Structure

A slum code allows you to move work to the compute nodes

/home/<case ID>/MathStuff/CellposePred.slurm

Save

Key Bindings: Default

Font Size: 12px

Mode: Text

Theme: Solarized Lig

Wrap

Soft Tabs

```
1  #!/bin/bash
2  #SBATCH --account= <HPC account>
3  #SBATCH --job-name=cellpose_pred_test
4  #SBATCH --nodes=1
5  #SBATCH --partition=gpu
6  #SBATCH --gres=gpu:1
7  #SBATCH --mem=10G
8  #SBATCH --time=0:15:00
9  #SBATCH --output=cellpose_testing_%j.out
10 #SBATCH --error=cellpose_testing_%j.err
11
12 echo "Starting job on $(hostname)"
13 echo "Time: $(date)"
14
15 # Load module
16 module load Python/3.12
17
18 # Navigate to project directory
19 cd /home/<case ID>/FullSlideTransforms/CellposePredictions/
20
21 # Create virtual environment if it doesn't exist
22 if [ ! -d "reg_venv3" ]; then
23     python -m venv reg_venv3
24 fi
25
26 # Activate environment
27 source reg_venv3/bin/activate
28
29 cd /home/<case ID>/MathStuff/
30
31 # Run script
32 python CellposeOnSmallRegion.py
33
34 echo "Job finished at $(date)"
35
36
37
```

Parameters to run on compute node

Load libraries and packages

Run script

Slurm Code

A slum code allows you to move work to the compute nodes

/home/<case ID>/MathStuff/CellposePred.slurm

Save

Key Bindings: Default

Font Size: 12px

Mode: Text

Theme: Solarized Lig

Wrap

Soft Tabs

```
1  #!/bin/bash
2  #SBATCH --account= <HPC account>
3  #SBATCH --job-name=cellpose_pred_test
4  #SBATCH --nodes=1
5  #SBATCH --partition=gpu
6  #SBATCH --gres=gpu:1
7  #SBATCH --mem=10G
8  #SBATCH --time=0:15:00
9  #SBATCH --output=cellpose_testing_%j.out
10 #SBATCH --error=cellpose_testing_%j.err
11
12 echo "Starting job on $(hostname)"
13 echo "Time: $(date)"
14
15 # Load module
16 module load Python/3.12
17
18 # Navigate to project directory
19 cd /home/<case ID>/FullSlideTransforms/CellposePredictions/
20
21 # Create virtual environment if it doesn't exist
22 if [ ! -d "reg_venv3" ]; then
23     python -m venv reg_venv3
24 fi
25
26 # Activate environment
27 source reg_venv3/bin/activate
28
29 cd /home/<case ID>/MathStuff/
30
31 # Run script
32 python CellposeOnSmallRegion.py
33
34 echo "Job finished at $(date)"
35
36
37
```

change to MAMS account

max memory and time before your job is canceled

Change current directory to where your code is

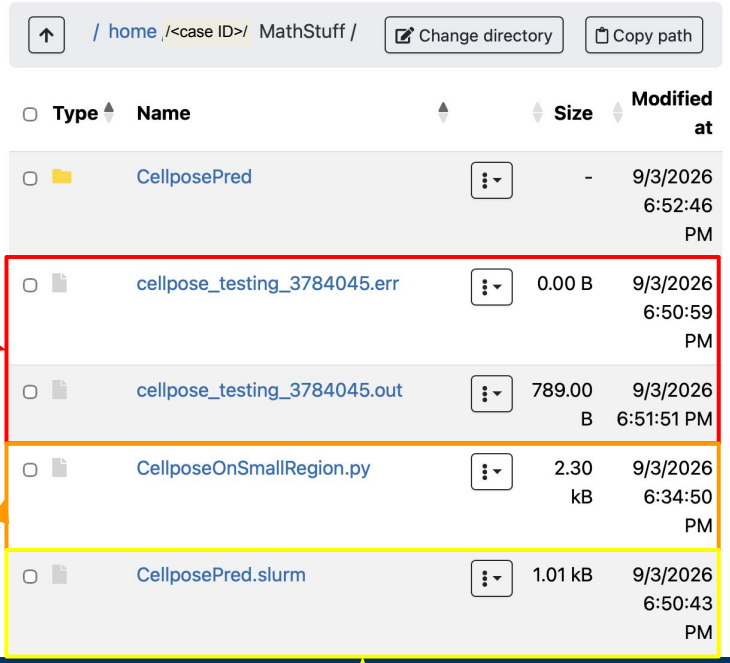
Script to run on compute nodes

Slurm Code: Output and Error

Paths to files containing output and error messages

When running code in HPC, you must direct the output and error messages to their own files

```
1  #!/bin/bash
2  #SBATCH --account= <HPC account>
3  #SBATCH --job-name=cellpose_pred_test
4  #SBATCH --nodes=1
5  #SBATCH --partition=gpu
6  #SBATCH --gres=gpu:1
7  #SBATCH --mem=10G
8  #SBATCH --time=0:15:00
9  #SBATCH --output=cellpose_testing_%j.out
10 #SBATCH --error=cellpose_testing_%j.err
11
12 echo "Starting job on $(hostname)"
13 echo "Time: $(date)"
14
15 # Load module
16 module load Python/3.12
17
18 # Navigate to project directory
19 cd /home/<case ID>/FullSlideTransforms/CellposePredictions/
20
21 # Create virtual environment if it doesn't exist
22 if [ ! -d "reg_venv3" ]; then
23     python -m venv reg_venv3
24 fi
25
26 # Activate environment
27 source reg_venv3/bin/activate
28
29 cd /home/<case ID>/MathStuff/
30
31 # Run script
32 python CellposeOnSmallRegion.py
33
34 echo "Job finished at $(date)"
35
```



Python script

Slurm code

Virtual Environments

/home/<case ID>/MathStuff/CellposePred.slurm

```
1  #!/bin/bash
2  #SBATCH --account= <HPC account>
3  #SBATCH --job-name=cellpose_pred_test
4  #SBATCH --nodes=1
5  #SBATCH --partition=gpu
6  #SBATCH --gres=gpu:1
7  #SBATCH --mem=10G
8  #SBATCH --time=0:15:00
9  #SBATCH --output=cellpose_testing_%j.out
10 #SBATCH --error=cellpose_testing_%j.err
11
12 echo "Starting job on $(hostname)"
13 echo "Time: $(date)"
14
15 # Load module
16 module load Python/3.12
17
18 # Navigate to project directory
19 cd /home/<case ID>/FullSlideTransforms/CellposePredictions/
20
21 # Create virtual environment if it doesn't exist
22 if [ ! -d "reg_venv3" ]; then
23     python -m venv reg_venv3
24 fi
25
26 # Activate environment
27 source reg_venv3/bin/activate
28
29 cd /home/<case ID>/MathStuff/
30
31 # Run script
32 python CellposeOnSmallRegion.py
33
34 echo "Job finished at $(date)"
35
36
37
```

Virtual environments are isolated directories with an interpreter and sets of packages + libraries

- ensures different project libraries are kept separate
  - version updates don't break code
- simplifies package imports
- easy to share package lists / create dependency files

Deals with loading all libraries and packages

Virtual Environments

Virtual environments help mitigate dependency issues and simplify package installation

Create requirements txt file in pip

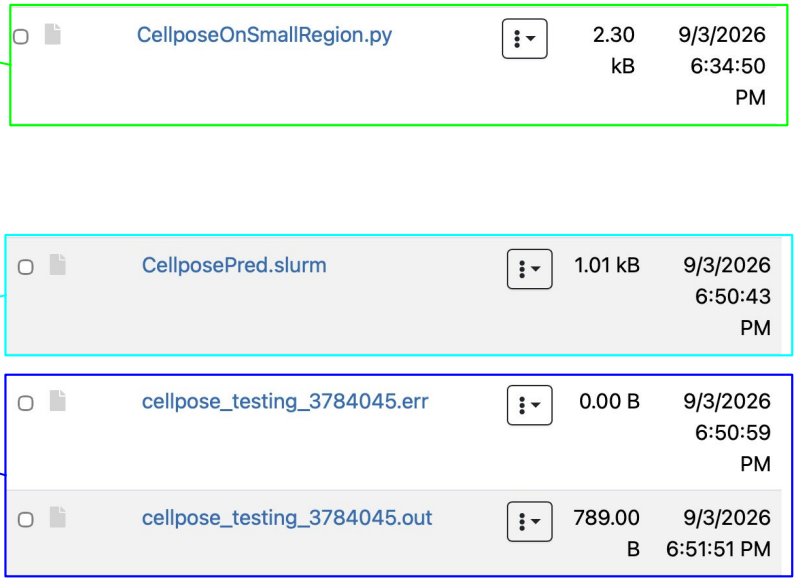
```
pip freeze > requirements.txt
```

Create virtual environment

```
python -m venv reg_venv                                  Make virtual environment
source reg_venv/bin/activate                             Activate virtual environment
pip install -r "<file path>/requirements.txt"            Install packages into environment
```

Steps to Running Code on HPC: Submitting Batch Jobs (Compute Node)

1. upload script to HPC
2. test small problem (*) to ensure code runs
3. create slurm script
4. submit batch job to HPC and wait for output



* short run time, low memory tests can be run on login node

Submitting Code

Login node

(Interactive session)

Request resources
salloc -N 1 -c 1 --mem=1G

Load Python (or other language, R may have different steps)
module load Python/3.12

Import libraries and packages
python -m venv reg_venv
source reg_venv/bin/activate
pip install -r "<file path>/requirements.txt"

Run code
cd <path for script directory>
python <script>

Login node example:

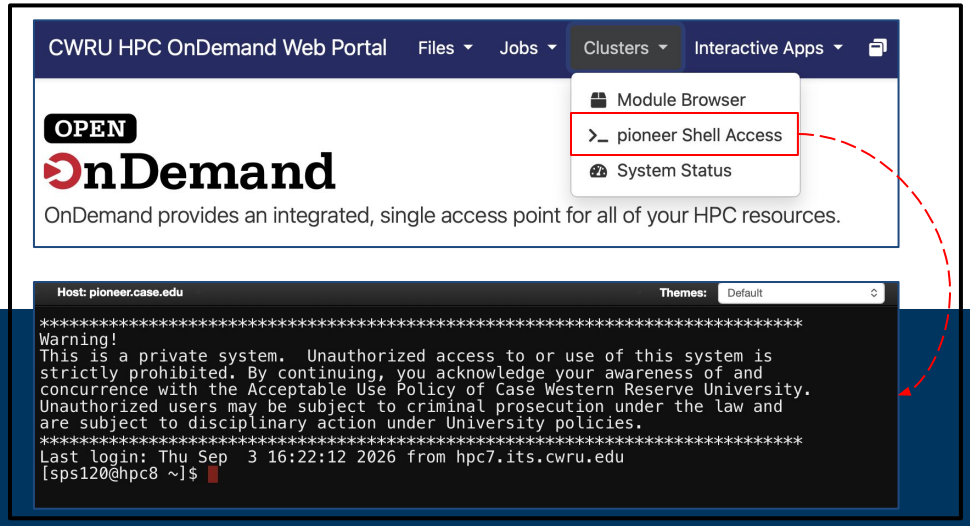
```
salloc -N 1 --mem=10M -p smp
module load Python/3.12
python -m venv reg_venv2
source reg_venv2/bin/activate
cd /home/<case ID>/FullSlideTransforms/
python TestROIFinder.py
```

Compute node

```
cd <path for slurm code directory>
sbatch <script name>.slurm
```

Compute node example:

```
cd /home/<case ID>/MathStuff/
sbatch CellposePred.slurm
```



Important Notes

1. **Ensure you are using the right nodes**
   a. most of the time you should use compute nodes by running slurm codes
   b. input nodes are good for testing and submitting batch jobs
2. **Keep an eye on the memory needed and time**
   a. don't request lots of memory if unnecessary
   b. if you don't allocate enough time / memory, your job will be canceled

Fair Warning

If using the login node for computational jobs, the following may occur:

- HPC staff may kill your job
- you can receive a strongly worded email from HPC staff

In extreme cases, misuse of login nodes can cause the node to crash due to the computational load!

For more tech-savvy researchers, note that you are **NOT to use agent-based assistants on the login node** as they may create large computational loads without your knowledge.

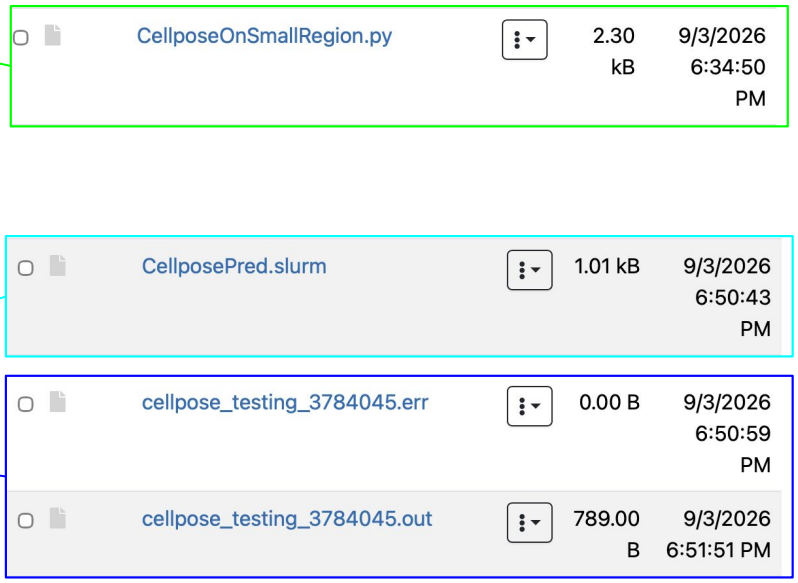
Helpful Commands

- Check files in path:
  ls “<file path name>”
- Change current directory:
  cd <file path>
- To get number of files in current directory:
  cd <folder path>
  ls -A | wc -l
- To see your batch jobs in queue:
  squeue -u $USER
- Check summary of jobs (start times of all jobs):
  squeue --start
- Cancel job:
  scancel -u <caseID>

Summary of Running Code on HPC

Steps to run code on HPC:

1. upload script to HPC
2. test small problem (*) to ensure code runs
3. create slurm script
4. submit batch job to HPC and wait for output



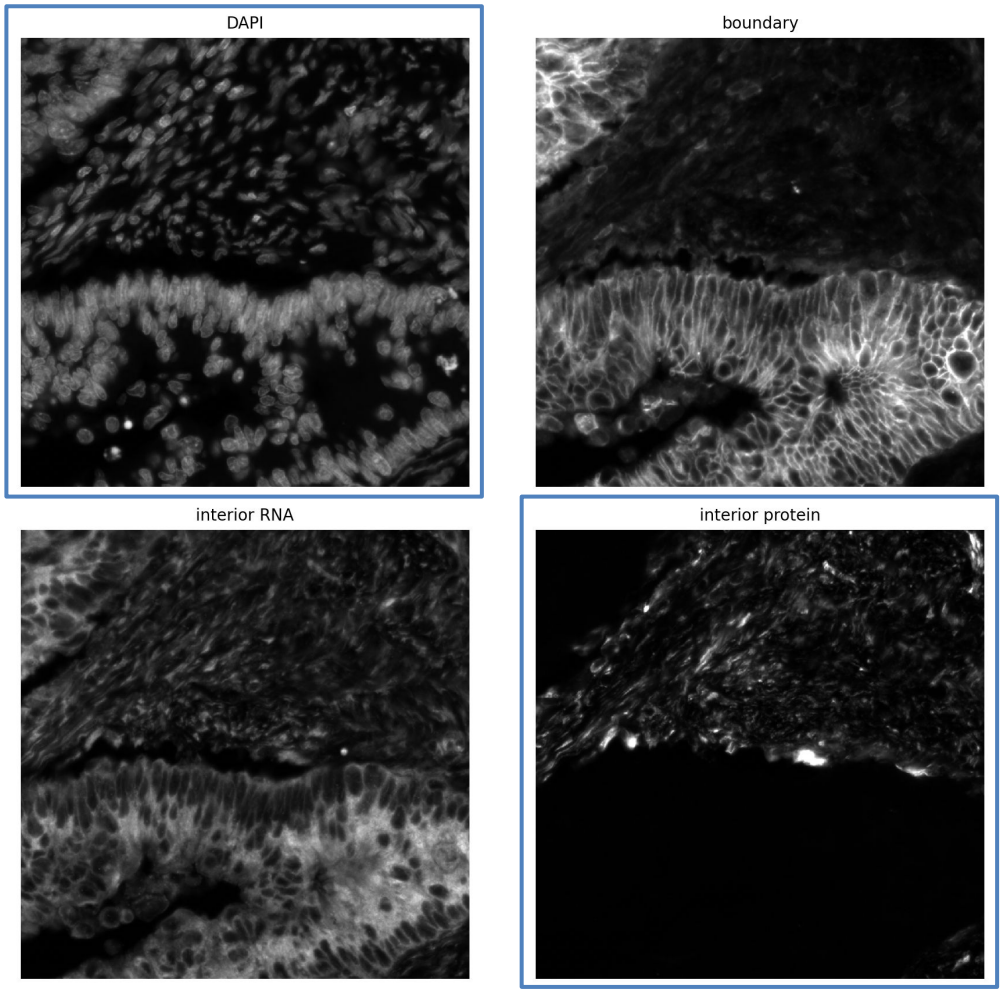
* short run time, low memory tests can be run on login node

Which node am I on?

- submitted a slurm code → job is run on compute node
- directly run file in interactive session → job is run on login node
  - uses: small tests, submitting slurm codes (goes to compute node)

Demo: Overview

Cell segmentation with Cellpose model



Input stains:

- DAPI
- interior protein

Output: instance cell segmentation

Demo

HPC demonstration… from HPC access

Show:

1. Cellpose prediction python script
2. Corresponding Slurm script
3. Running code

Working in R

If you work in R, check this resource to ensure correct module loading

Installing R Packages

Synopsis

Loading R via module load R/version configures the read-only location $R_HOME/library where system wide packages are installed. R will fallback to installing in the user's home directory if it detects that the default library location is not writable. Custom install locations are also supported by environment variables and options to package and library functions.

Important Note: The R versions below R/4.4.0 have serious vulnerabilities. Please use the newer version module.

Default User Install Location

The default user install location for R is ~/R/x86_64-pc-linux-gnu-library/version/ and R creates the directory if it does not exist:

```
> install.packages("yaml")
Warning in install.packages("yaml") :
  'lib = "/usr/local/easybuild_allnodes/software/R…" is not writable
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library
‘~/R/x86_64-pc-linux-gnu-library/4.1’
to install packages into? (yes/No/cancel) yes
```

R module and packages are a bit trickier to navigate on HPC:

- ensure using version 4.0.0 or newer
- make sure downloaded to correct directory and has correct functionality

Example in R

Example code for R in HPC can also be found [here]

Interactive Job Submission

Interactive Serial

To use R interactively for a serial job, first request interactive use of a processor in a compute node

```
srun --x11 -N 1 -c 1 --time=1:00:00 --pty /bin/bash
```

and wait until you are connected to a shell in a compute node. (The default length of an interactive session is ten hours.) Then load the R module, and potentially identify the path to any custom modules that you may have installed for this version of R:

```
$ module load R/4.2.1-foss-2022a
$ export R_LIBS_USER=/home/mrd20/.local/pioneer/R/4.2.1
```

and then invoke R using the shell command "R":

```
[<caseID>@<computeNode> $ R

R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
 Type 'q()' to quit R.
>x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
> x
[1] 10.4  5.6  3.1  6.4 21.7
> 1/x
[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295
> q()
Save workspace image? [y/n/c]: n
```

Batch Job Submission

Serial Job

To run an R batch job, prepare a SLURM script that contains the command to load the R module. A simple single-processor example is as follows:

```
#!/bin/bash
#SBATCH -t 1:00:00
#SBATCH -c 1
#SBATCH -o serial-R.out%j # capture jobid in output file name

# load the R module
module load R/4.2.1-foss-2022a

# copy files to scratch space for job execution
# (not strictly necessary unless data size is large)
cp R-example/* $PFSDIR
cd $PFSDIR

# Run R script
R CMD BATCH <scriptfile.r>

# copy all work from scratch-space back to submit directory
cp * $SLURM_SUBMIT_DIR
```

In the above sample script, note that 'input.file' should be replaced by the name of your input file containing R commands to be carried out.

Other Ways to Access Cluster

Quinn's Demo

HPC Command Line Demo

CWRU SIAM Student Chapter
Date: October 5th



Case Department of Mathematics,
Applied Mathematics, and Statistics
CASE WESTERN RESERVE
UNIVERSITY

Note: Demo on Windows

- Demo intended to show an example workflow using the HPC
- It includes programs which maybe specific to windows
- Mac has analogous programs that can be used

- Also note this demo is done using my username, etc. All of this will need changed before use

Text Editor in Windows

- Notepad that comes with Windows should not be used when editing files for use on HPC
- Notepad injects different formatting into the file, making it unusable
- This can be fixed with command dos2unix [filename]
- Notepad++ is a free text editor available online to avoid this



Command Line Access in Windows

- While command prompt can be used to run bash commands, it is not idea for running Linux
- Windows Subsystem Linux (WSL) allows the use of a Linux environment without the difficulty of dual booting



File Access on the Cluster

- Files stored on the cluster can be accessed through the command line or through GUI based file explorers
- Cyberduck is a free opensource SFTP file transfer tool that allows access to HPC file system though a GUI

Accessing the Cluster

- If not on Case Wi-Fi, a VPN (available from the software center) must be used to access the cluster
- Once in WSL the command ssh qpa@pioneer.case.edu to login

```
qpa@Quinn:~$ ssh qpa@pioneer.case.edu
**************************************************************************
Warning!
This is a private system.  Unauthorized access to or use of this system is
strictly prohibited. By continuing, you acknowledge your awareness of and
concurrence with the Acceptable Use Policy of Case Western Reserve University.
Unauthorized users may be subject to criminal prosecution under the law and
are subject to disciplinary action under University policies.
**************************************************************************
qpa@pioneer.case.edu's password:
Last login: Tue Oct 13 09:54:09 2026 from 172.20.17.18
[qpa@hpc8 ~]$
```

- [username@hpc# ~] shows the login was successful

Navigating to Example Directory

- Basic commands for file navigation
  - ls - lists files in current directory
  - cd [dir name] - change directory (use tab for autocomplete)
  - cd .. - go up one directory level
  - cd - go to home directory

- Keep long commands in separate text file for quick/repetitive use

Running on Compute Nodes

- Large installs / jobs must be ran on compute nodes, not login nodes
- Methods to run on compute node:
  - srun not ./ to execute a specific file
  - srun -c 4 --mem=16G --pty /bin/bash to move command prompt to compute node rather than login node
  - sbatch script.sh to execute a slurm file (.sh)
- Slurm files are scripts that are batch to a compute node
  - For specific details/format please refer to Sara's presentation
- Running jobs on compute nodes is very important and something to be constantly mindful of

My Workflow

- We have already logged in, so I will navigate to my desired directory and list its contents

```
[qpa@hpc8 ~]$ cd autoibamr/tmp/build/v0.19.0/examples/IB/explicit/ex3
[qpa@hpc8 ex3]$ ls
CMakeFiles            fila_1024.vertex  input2d
CTestTestfile.cmake   fila_256.beam     main2d
IB2D.out              fila_256.spring   script.sh
IB2d.log              fila_256.target   update_target_point_positions.cpp
Makefile              fila_256.vertex   update_target_point_positions.h
cmake_install.cmake   fila_512.beam     vertwallmesh.asv
example.cpp           fila_512.spring   vertwallmesh.m
fila_1024.beam        fila_512.target   viz_IB2d
fila_1024.spring      fila_512.vertex
fila_1024.target      hier_data_IB2d
```

Loading Modules

- I will now load all necessary modules
- The code is now ready to run, which is executed by a slum file

```
[qpa@hpc8 ex3]$ module load libtirpc/1.3.4-GCCcore-13.2.0
[qpa@hpc8 ex3]$ module load foss/2023b
[qpa@hpc8 ex3]$ module load CMake/3.27.6-GCCcore-13.2.0
[qpa@hpc8 ex3]$ module list

Currently Loaded Modules:
  1) GCCcore/13.2.0                    16) UCC/1.2.0-GCCcore-13.2.0
  2) libtirpc/1.3.4-GCCcore-13.2.0     17) OpenMPI/4.1.6-GCC-13.2.0
  3) zlib/1.2.13-GCCcore-13.2.0        18) OpenBLAS/0.3.24-GCC-13.2.0
  4) binutils/2.40-GCCcore-13.2.0      19) FlexiBLAS/3.3.1-GCC-13.2.0
  5) GCC/13.2.0                        20) FFTW/3.3.10-GCC-13.2.0
  6) numactl/2.0.16-GCCcore-13.2.0     21) gompi/2023b
  7) XZ/5.4.4-GCCcore-13.2.0           22) FFTW.MPI/3.3.10-gompi-2023b
  8) libxml2/2.11.5-GCCcore-13.2.0     23) ScaLAPACK/2.2.0-gompi-2023b-fb
  9) libpciaccess/0.17-GCCcore-13.2.0  24) foss/2023b
 10) hwloc/2.9.2-GCCcore-13.2.0        25) ncurses/6.4-GCCcore-13.2.0
 11) OpenSSL/1.1                       26) bzip2/1.0.8-GCCcore-13.2.0
 12) libevent/2.1.12-GCCcore-13.2.0    27) cURL/8.3.0-GCCcore-13.2.0
 13) UCX/1.15.0-GCCcore-13.2.0         28) libarchive/3.7.2-GCCcore-13.2.0
 14) libfabric/1.19.0-GCCcore-13.2.0   29) CMake/3.27.6-GCCcore-13.2.0
 15) PMIx/4.2.6-GCCcore-13.2.0



[qpa@hpc8 ex3]$ sbatch script.sh
Submitted batch job 4151261
```

Job Status and Canceling

- Job status can be checked using command squeue –u [username]

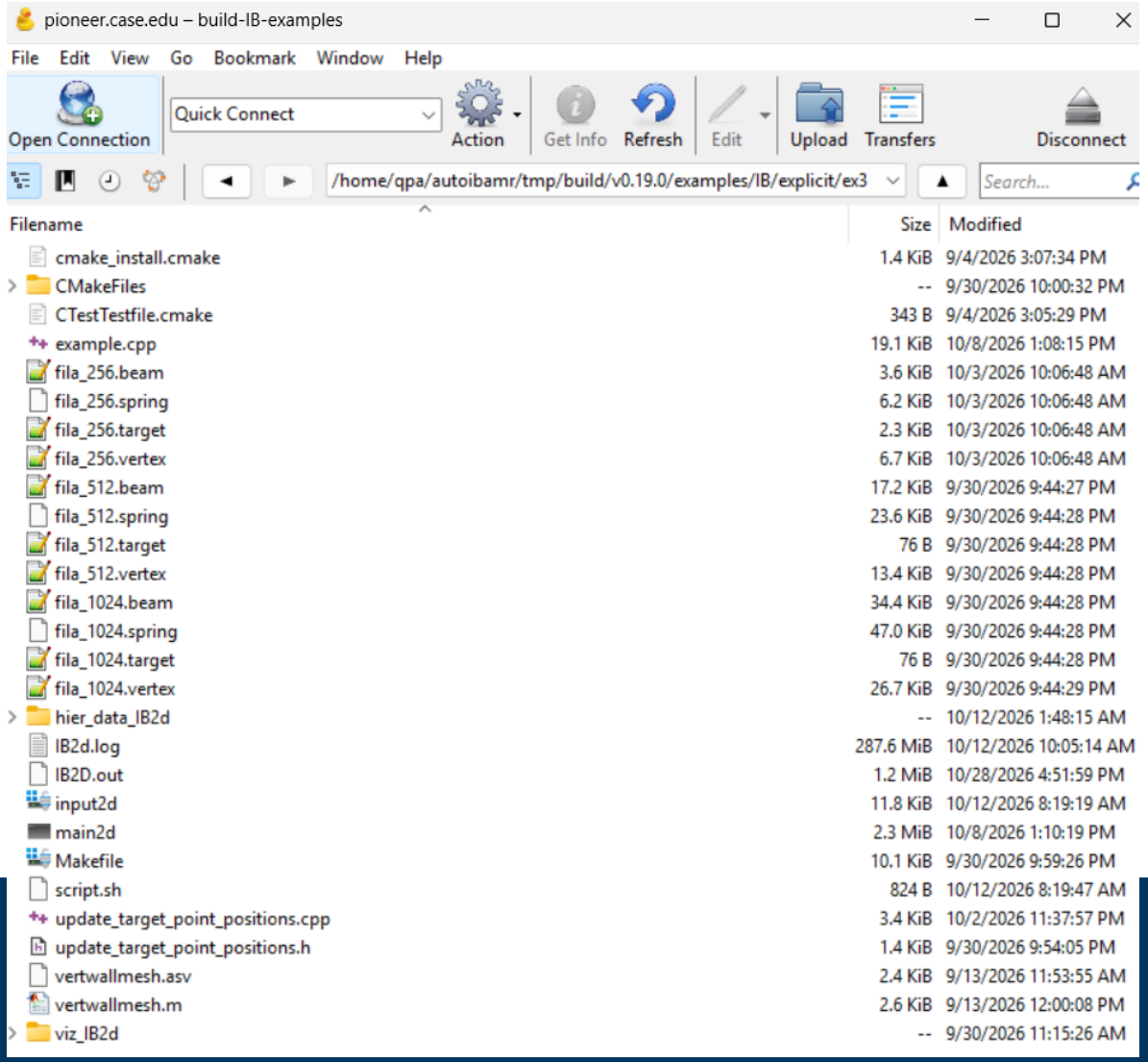
```
[qpa@hpc8 ex3]$ squeue -u qpa
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST
(REASON)
           4151261     batch ex3CFNSt      qpa  R       1:09      1 compt298
```

- If we wish to cancel a job, that can be done using scancel [JOBID]

```
[qpa@hpc8 ex3]$ scancel 4151261
[qpa@hpc8 ex3]$ squeue -u qpa
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST
(REASON)
```

Accessing Results

- Use Cyberduck to navigate to the desired directory
- Output files can be downloaded, input files uploaded or files opened in a separate editor



Conclusion

- This is an example of the logistics of running an example workload on the cluster
- This was not meant to be a comprehensive guide on how the cluster must be used, rather just how it can be used
- More details on slurm files etc can be found in Sara's HPC presentation
- If you have any questions feel free to reach out