

HPC Workshop 1

Jan. 23 and 24 , 2024

11:30 – 12:30 pm (followed by pizza)

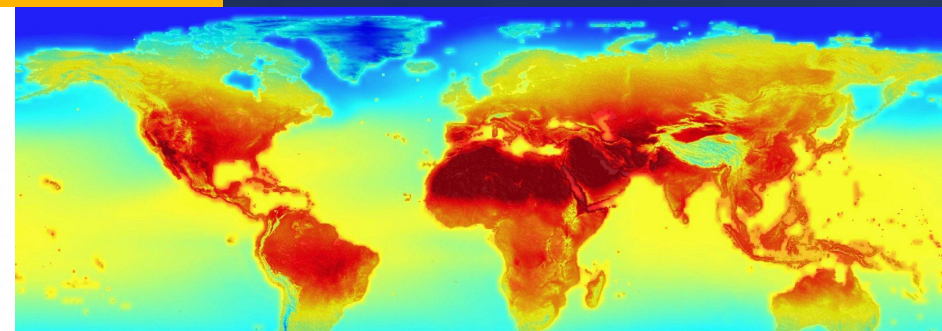
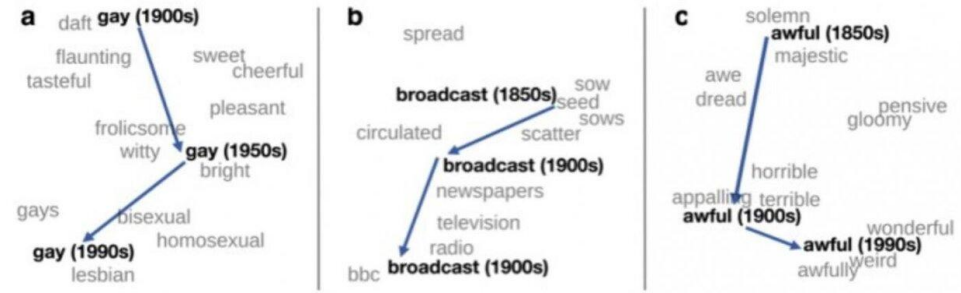
Location: Elings Hall 1601

Register @ <https://csc.cnsi.ucsb.edu>

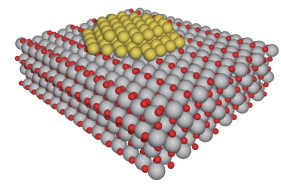
Quickly start using HPC resource at UCSB

- What is HPC?
- Quickly get Started to Use HPC
- Basic Linux Commands
- Basic Slurm Commands
- Producible and Portable Software Environment

Computational Linguistics

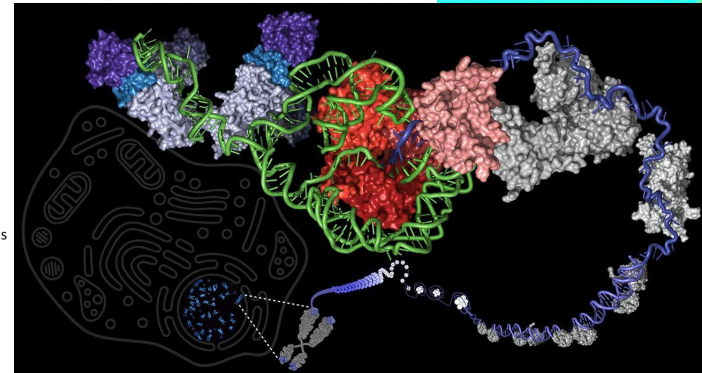


KS-DFT



The total energy in Kohn-Sham Density Functional Theory (KS-DFT) is expressed as

$$E_{total} = T_s + \int dr V_{ext}(r)\rho(r) + E_{xc}[\rho] + \frac{1}{2} \int \int dr dr' \frac{\rho(r)\rho(r')}{|r-r'|}$$



Introduction to High-Performance Computing (HPC)

Paul Weakliem and Jay Chi

January 23 and 24, 2025

Speaker Introductions



Paul Weakliem, PhD

Co-Director

Center for Scientific Computing &
California Nanosystems Institute

Eling 3231

weakliem@cnsi.ucsb.edu



Fuzzy Rogers

Research Computing Administrator

Center for Scientific Computing &
Materials Research Laboratory

MRL 2066B

fuz@mrl.ucsb.edu



Yu-Chieh "Jay" Chi, PhD

Research Computing Consultant

Center for Scientific Computing &
Enterprise Technology Services

Elings 3213

jaychi@ucsb.edu



UC SANTA BARBARA
Information Technology

Our Research IT Partners



Justin Maness
Interim Director of Engineering
Computing Infrastructure (ECI)
3152a Harold Frank Hall
jmaness@engineering.ucsb.edu



Michael Colee
Director of Research IT
6703 Ellison Hall
mtc@eri.ucsb.edu



Ted Cabeen
Director of Life Science
Computing Group (LSCG)
2306 Life Science
ted.cabeen@lscg.ucsb.edu

Letters & Science
INFORMATION TECHNOLOGY

Andreas Boschke
Director of Infor. Infrastructure
at Letter & Science IT (LSIT)
2306 Life Science
andreas@lsit.ucsb.edu

Introduction

- Most of you are here because you are doing computational research.
 - Have a specific scientific computing project you are interested in
 - Need large-scale computing or data memory resources
- The actual HPC hardware is probably important
 - Hardware is important but not that interesting
- How many linux command do you need to know for working in the **Pure Linux environment?**
 - **You don't have to be a Linux expert.**
- In this workshop, you will learn
 - What is the Center for Scientific Computing (CSC) at UCSB?
 - Introduction to High-Performance Computing (HPC) at UCSB
 - Quickly get started to use cluster
 - Learn the basic of Linux Commands
 - File Transfer
 - Reproducible and Portable Software Environment
 - Learn the basic of Slurm (Simple Linux Utility for Resource Management) commands to submit jobs to the cluster



What is Center for Scientific Computing (CSC)

What we are:

- A home for HPC and expertise with national supercomputing centers leveraging CNSI, MRL, and ITS resources to enable researchers to focus on the research project/education and not the infrastructure.

Support Capabilities

- We provide the computational infrastructure.
- We provide a large amount of storage for data to be processed.
- We provide user support, assistance with use of resource, installation of applications, and training.
- **We work with your local IT staff to provide help.**
- Regular working hours, realistically, 8:30 am - 5 pm Monday through Friday. But we try to make sure the clusters are running near 24/7 (I'd say 365, but it's UCSB and we're a small group)

We **don't upgrade** our cluster system often.

Overview

- Most research now involves some form of computing
 - Often you're solving equations, or analyzing data/doing statistics ('data science'). Engineers often will model a device.
 - Some specific examples:
 - Protein Folding
 - Structure of crystal
 - Search for patterns in DNA
 - Predicting the spread of wildfire
 - Weather prediction
 - Natural Language Processing
- Like many parts of research, you often start a small, with a simple idea, but it grows beyond what you (or your computer) can do yourself!
- Solution:
 - Better Computer
 - High-Performance Computing (HPC)
 - Cloud (Can be both of above, with arbitrary size) - somebody else's computer!

When to run on HPC

Laptop/Desktop

- Coding
- Exploratory phase
- Small Datasets
- Run on few cores
- Few computational jobs



HPC

- Scaling up to
 - Large datasets
 - Long runtimes
- Run on many cores
- Multiple computational jobs



What is High-Performance Computing (HPC)?

- High-Performance Computing (HPC) allows scientists and engineers to solve complex science, engineering, and business problem using applications that require high bandwidth, enhanced networking, and very high compute capabilities. Ref: <https://aws.amazon.com/hpc/>
- Multiple computer nodes connected by a very fast interconnect.
- Each node contains many CPU cores (around 12-40 cores) and 4-6G RAM.
- Allows many users to run calculations simultaneously on nodes.
- Allows a single user to use many CPU caress incorporating multiple nodes.
- Often has high end (64 bit/high memory) GPUs

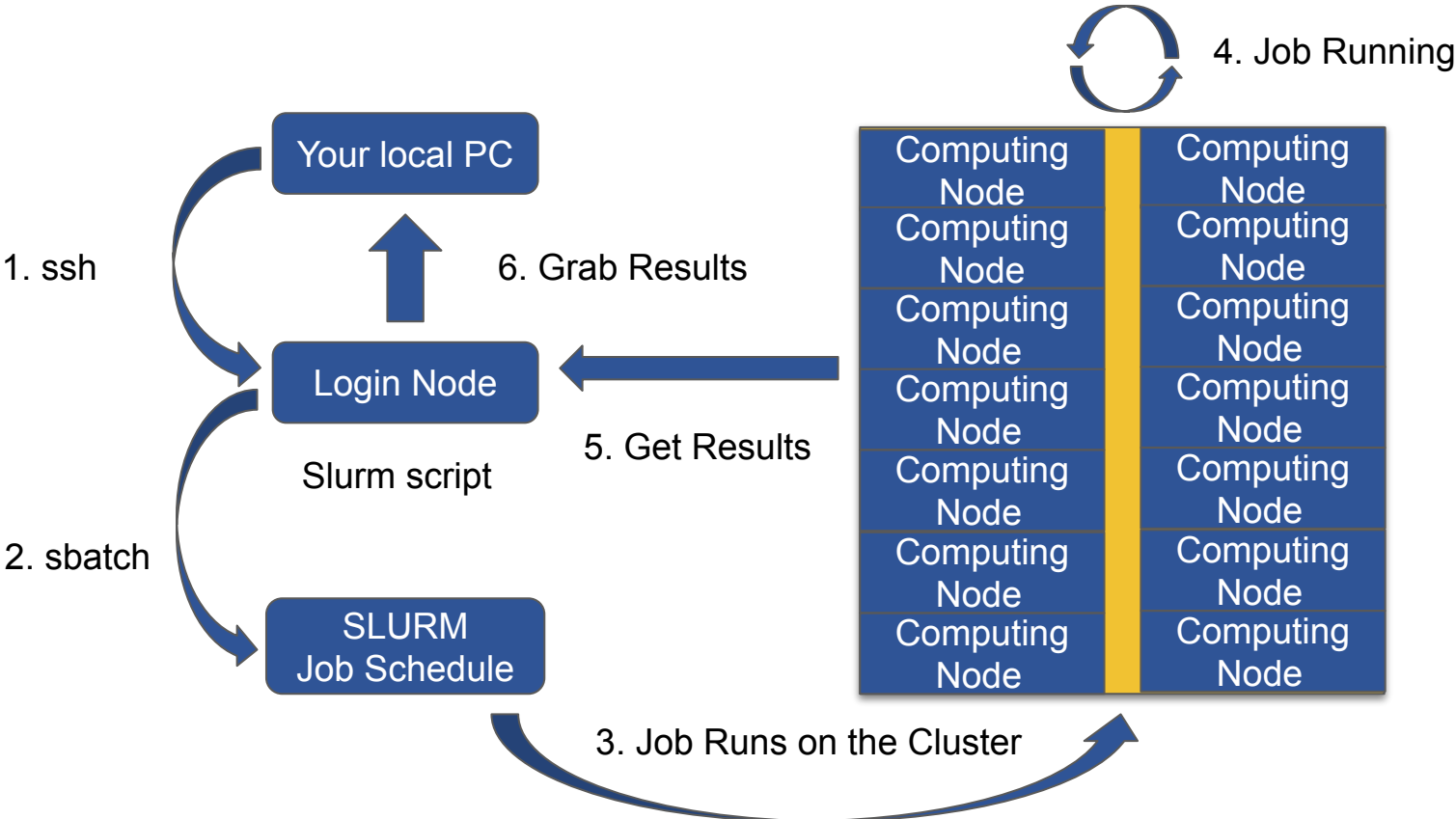


UCSB provides access and support for multiple HPC resources and educational/training/research support.

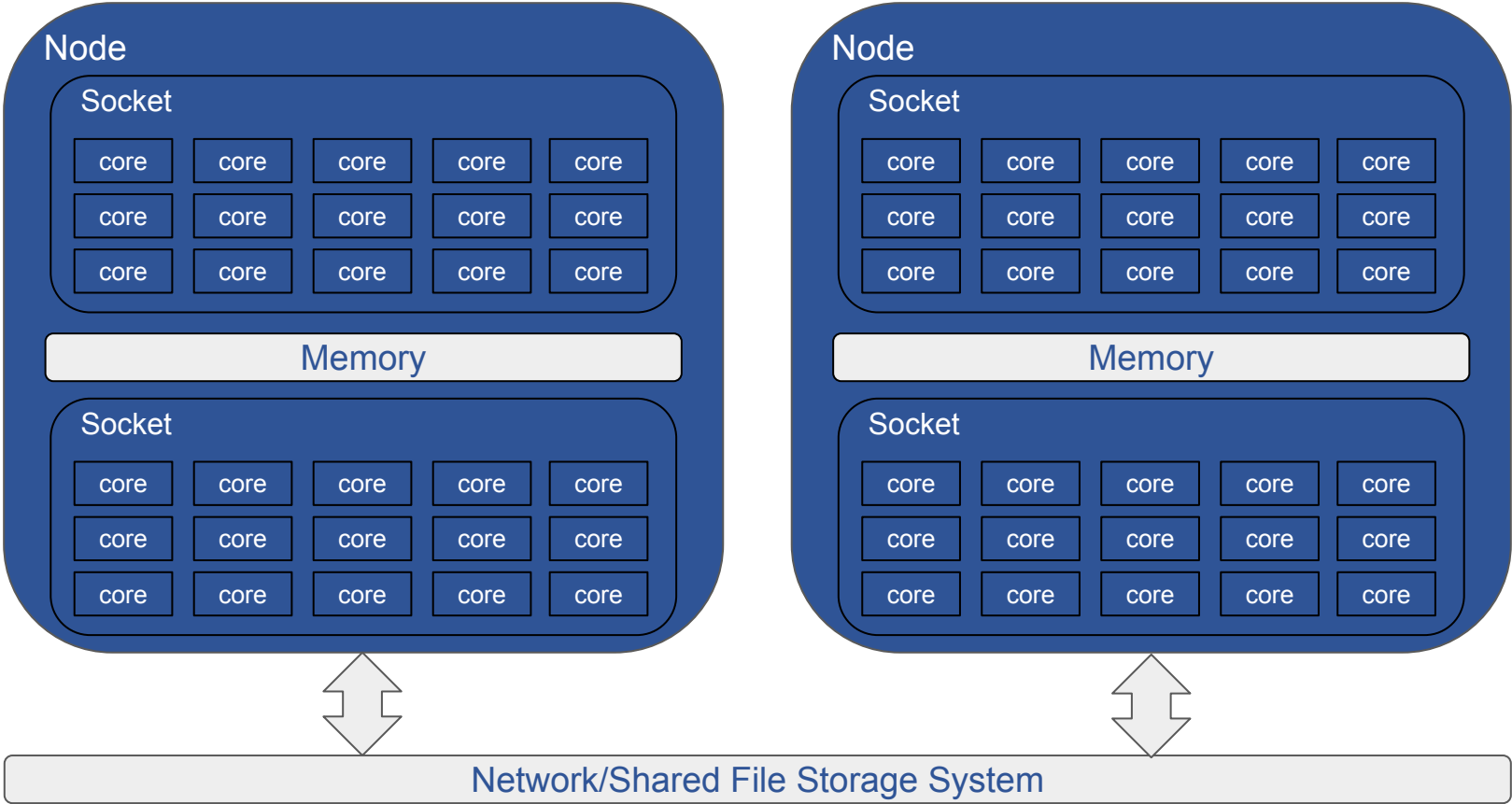
HPC is not always the only one solution!!!

- Sometimes you need a faster desktop workstation
- Sometimes 'Cloud' is the right solution (need 1000 nodes, but only once every 3 months)
- Sometimes you might even need your own cluster

General HPC Workflow



HPC Infrastructure





Terminologies Definitions

- **Core:** The smallest compute unit that can process logic and floating point (run a program).
- **CPU:** The chip that processes the basic instructions that drive a computer. The term *processor* is used interchangeably with the term **central processing unit (CPU)**. CPUs have many cores.
- **Socket:** A physical processor which includes multiple cores with sharing memory. Most of our stuff has 2 sockets, for 2 CPUs.
- **Node:** An individual computer that includes one or more sockets, memory, storages, etc. The fast network connects other nodes.

HPC system at CSC

- Campus available cluster Knot (aka 'knot7') (CentOS/RH 7):
 - 110 nodes with ~ 1400 cores system
 - 4 "fat" nodes with 1TB memory RAM
 - GPU nodes (12 M2050's) (too old now)
- Campus available cluster Pod (CentOS/RH7)
 - 70 nodes with ~ 2600 cores system
 - 4 "fat" nodes with 1TB memory RAM
 - 15 GPU nodes (Quad NVIDIA V100/32GB with NVLINK)
 - GPU Development node (P100, T4)
- Published papers should acknowledge CSC - <https://csc.cnsi.ucsb.edu/publications>

Request access: <https://csc.cnsi.ucsb.edu/forms/user-account>

- Condo Clusters
 - Guild (70 nodes) EOL
 - Braid (120 nodes, also has GPUs) fairly old now
 - Braid2 (20 nodes with some GPUs)

Pls buy nodes in the clusters, CSC handles infrastructure.

Pod Data Storage System

Home directory Space: /home/user_name

- Not unlimited quotas
 - Each dollar spent on storage is one not spent on compute (funded by either NSF (Pod), or CNSI and MRL)
 - Keep it to a TB or two?

Network mounted scratch file Storage: /scratch

- High speed but temp files
- 19 TB NFS mounted file system (/scratch)
- 70 TB NFS mounted file system (/bigscratch)

/csc/central - 'near-line'

What do you do with your data - your local desktops, NAS, etc. Google drive?? AWS - S3 (cost, but small)

What Other Computational Resources are available

- UCSB Center for Scientific Computing (CSC) HPC clusters
 - Access to all UCSB staff, Pod (free) and condo (PI) clusters.
- Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS)
 - National HPC resources funded by NSF. Free*
- NRP Nautilus Cluster (Consumer GPUs)
 - National cloud computing resource for accelerating machine learning on the GPUs. Free*
- Aristotle Cloud (LSIT)
 - UCSB local cloud resource, e.g. [Jupyter hub](#)
- Secure Compute Research Environment (SCRE)
- Other discipline specific UCSB resources
 - NCEAS, ERI, ECI, your local department
- Commercial Cloud Computing Resources:
 - AWS, Microsoft Azure, google Cloud Platform

HPC Resources of Useful Information

- CSC Software Documentation
 - <https://csc.cnsi.ucsb.edu/docs>
- National HPC resources
 - ACCESS: <https://access-ci.org/>
 - San Diego Supercomputer Center: <https://www.sdsc.edu/>
 - NRP Nautilus: <https://portal.nrp-nautilus.io/>
- Transitioning from XSEDE to ACCESS by using Globus
 - <https://www.globus.org/advance-to-access>
- UCSB Aristotle Cloud (LSIT):
 - <https://www.aristotle.ucsb.edu/> and <https://help.lsit.ucsb.edu/hc/en-us/categories/360005255312-Jupyter>
- UCSB Campus Cloud Information:
 - <https://www.it.ucsb.edu/explore-services/ucsb-campus-cloud>
 - <https://docs.cloud.ucsb.edu/>
- More information, go to <https://csc.cnsi.ucsb.edu/resources>

Connecting to POD

- For the Windows system, open a 'powershell' window (type powershell in the search bar)
- For the Mac or Linux system, you can open the terminal (in Applications->Utilities)
- From Windows, Mac, or Linux, connect to POD via the Visual Studio Code (VS Code)
- From either - use the ssh command
 - `$ ssh your_user_name@pod.cnsi.ucsb.edu`

```
PS C:\Users\weakliem> ssh pcw@pod.cnsi.ucsb.edu
pcw@pod.cnsi.ucsb.edu's password:
Last login: Wed Feb 28 16:39:19 2024 from vid.cnsi.ucsb.edu
-----
Welcome to Pod

For basic documentation to get started please see
http://csc.cnsi.ucsb.edu/docs/pod-cluster
-----
January 26, 2024 --- 3:30pm

CHECK YOUR JOBS
Pod login rebooted and SLURM restarted.
Some mounts had an NFS problem.

[pcw@pod-login1 ~]$
```

```
jaychi — jay@pod-login1:~ — ssh jay@pod-login1.cnsi.ucsb.edu — 90x30
(base) EEUC-YT61Y2PL:~ jaychi$ ssh jay@pod-login1.cnsi.ucsb.edu
jay@pod-login1.cnsi.ucsb.edu's password:
Last login: Mon Aug 29 09:28:40 2022 from 169-231-103-70.wireless.ucsb.edu
-----
Welcome to Pod

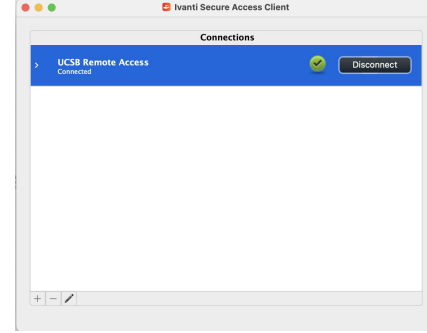
For basic documentation to get started please see
http://csc.cnsi.ucsb.edu/docs/pod-cluster
-----
March 2022

We are experiencing some slow /home performance because the
filesystem is fairly full - please, please, please, take a
look at what you can move off the system!!
-----
Apr 4, 2022
login node crashed - all jobs are fine. Please be careful
with running anything large directly on the login node!

[jay@pod-login1 ~]$
```

Connecting to POD

- ***Important:*** Remote (non UCSB) login via the VPN client:
 - <https://www.ets.ucsb.edu/pulse-secure-campus-vpn/get-connected-campus-vpn>
- For the Linux user, the steps are as same as for Mac users.
- If the VPN GUI does not work, please use VPN command line Interface (CLI).
 - `$ sudo apt install openconnect network-manager-openconnect network-manager-openconnect-gnome`
 - `$ sudo openconnect --protocol=nc https://ps.vpn.ucsb.edu/ra`
 - After prompting for your sudo password, you will be prompted for your username - this is your UCSB NetID.
 - Next, you will be prompted for your UCSB NetID password.
 - When prompted for "password#2", refer to the Step 4 instructions above (or the instructions in your terminal that should appear after you type your sudo password) to complete your Duo authentication.
 - Open one new Terminal window and ssh login to the POD/Braid2 cluster.



Key points to Use Cluster

There are 70+ computed nodes on the POD cluster, but there are only 2 login nodes.

- What does it mean?
- It means you are sharing these two login nodes with many other users when you are login. Running intensive programs on the login node will cause the login nodes to be slow for all users.
 - Login nodes are for editing files, transferring files, changing permissions, submitting jobs, and other “small-intensive” tasks.
 - We recommend to use interactive for interactive runs (e.g. Compiling, installation, testing)
 - For long running jobs: submit jobs to the queue
 - `$ srun -N 1 -n 1 -p batch --time=1:00:00 --pty bash -i`

You can't monopolize cluster - limit jobs/cores

Modules: Finding and Using Software on the POD

- Module system provides for the dynamics modification of a user's environment.
- Module commands allow the user to add applications and libraries to your environment.
- This allows us to simultaneously and safely provides several versions of the same softwares.
- All clusters have a default programming environment loaded for you when you login.
- There are some functional software are not modularized in /sw directory.
Please take a look if you need.
 - E.g. /sw/cnsi, or /sw/chem
 - Add to path in .bashrc or .bash_profile

Modules: Finding and Using Software on the POD

1. List available modules

```
$ module avail
```

```
...
```

2. Search available modules for MatLab

```
$ module avail MatLab
```

```
----- /sw/modulefiles -----
```

```
MatLab/R2016b MatLab/R2018a MatLab/R2018b MatLab/R2019a MatLab/R2019b MatLab/R2021b
```

3. Load the MatLab module

```
$ module load MatLab/R2021b
```

Modules: Finding and Using Software on the POD

4. Unload the MatLab module

```
$ module unload MatLab/R2021b
```

5. Purge all modules

```
$ module purge
```

6. List currently loaded modules

```
$ module list
```

Currently Loaded Modulefiles:

```
1) autotools 2) prun/1.2 3) gnu/5.4.0 4) ohpc
```

Reproducible and Portable Software Environment

Conda

- Beginner
- Experience with Conda
- Frequently changing dependencies
- Support on Linux, Mac, and Windows
- Run on native OS



HPC Containers

- Advanced User
- Experience with containers
- Often setup for a single tool
- Support on Linux, Mac and Windows require a VM
- Run on packaged OS, e.g. Ubuntu



- For more details about the Container will offer in the future quarter.

Conda Environment

- Package management system
 - Conda install and update open source packages (e.g. numpy, scipy, pytorch), and their dependencies
- Environment management system
 - You can use conda to create, load, and switch between multiple different environments
 - Multiple versions of software packages can co-exist without interference
- Multi-platform (Linux, MacOS, and Windows)
 - Conda environment are portable and can be installed on multiple platforms
- Multi-language (Python, R, etc.)

Environment_1:

```
pytorch==2.0.0  
pytorch-cuda==11.7
```

Environment_2:

```
pytorch==1.13.1  
cpuonly
```

Environment_3:

```
pytorch==1.13.1  
pytorch-cuda==11.6
```


Conda Environment

- **Activate your conda environment**

- \$ module load anaconda
- \$ source activate base

```
[jay@pod-login1 ~]$
```



```
(base) [jay@pod-login1 ~]$
```

- **Create a Conda environment**

- \$ conda create --name my_env_1

- **Activate your environment**

- \$ conda activate my_env_1

- **Install packages**

- \$ conda install pytorch==2.0.0 torchvision==0.15.0 torchaudio==2.0.0 cpuonly -c pytorch

- **List your Conda environment**

- \$ conda env list

- **Deactivate conda environment**

- \$ conda deactivate

Basic Linux Commands

- Listing files (ls)
- Print Working Directory (pwd)
- Change Directory (cd)
- Make Directory (mkdir)
- Copy (cp)
- Moving Files (mv)
- Remove Files (rm)
- Secure Copy (scp)
- Display beginning/end of file (head/tail)
- View file (cat)
- Display manual for a command (man)
- nano, vim, or emacs to edit your file.

Basic Linux Commands (ls & pwd)

- The ls (list) command files and directories in a directory.

- General syntax:

ls [OPTIONS] [FILENAME]

- OPTIONS include:

-l long listing, includes file date and size

-a displays all files

- Example: `$ ls -al /home/jay`

- ls: command

- -al: flag

- /home/jay: argument

- pwd stands for print working directory.

`$ pwd`

Basic Linux Commands (cd)

- The cd (change directory) command is used to change one directory to another.
 - General syntax:
`cd [DIRECTORY]`
 - Change your present directory to the parent directory:
`$ cd ..`
 - Change your present directory to the home directory:
`$ cd ~`
 - `..`: your current directory

Basic Linux Commands (mkdir & cp)

- The mkdir (make directory) command creates a new directory.

- General syntax:

```
mkdir [OPTIONS] Folder_name
```

- The touch command creates a new file.

- General syntax:

```
touch file_name
```

- The cp (copy) command is used to copy a file or directory.

- General syntax:

```
cp [OPTIONS] Source Destination
```

- OPTIONS include:

- r recursively copy a directory, all files and subdirectories inside it.

Basic Linux Commands (mv & rm)

- The mv (move) command is used to move or rename a file or directory.

- General syntax:

```
mv Source Destination
```

- The rm (remove) command is used to delete a file or directory.

- General syntax:

```
rm [OPTIONS] file_name
```

- OPTIONS include:

- -r recursively delete a directory, all files and subdirectories inside it.

- **Important:** After **rm** or **rm -r** command is executed, all files are gone and can't be found in recycle bin or the like!

Basic Linux Command (head, tail, and cat)

- The head/tail command is used to display the starting/ending lines of the file.
 - General syntax:
`head [options] file_name`
 - Print the first n line
`$ head -n file_name`
 - Print the last n line
`$ tail -n 5 file_name`
- The cat (concatenate) command is used to display the entire file on the screen.
 - General syntax:
`cat file_name`

There's also the 'more' and 'less' commands to display a page at a time!

File Transfer

- The scp (secure copy) command is used to transfer files between two locations.

- General syntax:

```
scp [OPTIONS] LOCAL REMOTE
```

```
scp [OPTIONS] REMOTE LOCAL
```

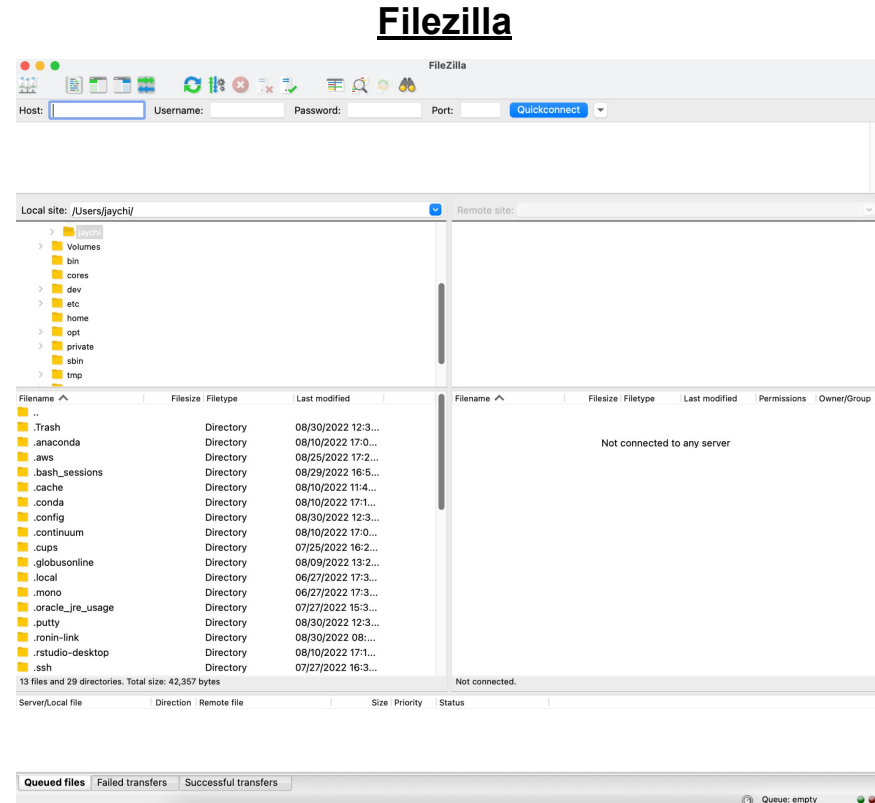
```
scp [OPTIONS] REMOTE REMOTE
```

- OPTIONS include:

-r recursively copy a directory, all files and subdirectories inside it.

File Transfer

- How do I uploaded data & download my files?
 - Graphical User Interface (GUI)
 - Filezilla: <https://filezilla-project.org/>
 - Cyberduck: <https://cyberduck.io/>
 - Command-Line Interface (CLI)
 - “scp” command
- FileZilla
 - Host: pod.cnsi.ucsb.edu
 - Username: your_user_name
 - Password: your_password
 - Port: 22
- Globus (for larger files transfers)
 - <https://csc.cnsi.ucsb.edu/docs/globus-v5-new>



More Linux Resource Information

- Access to [GitHub Copilot](#)
 - [GitHub CLI Installation](#) and [Copilot in the CLI extension Installed](#)
 - UCSB Software Carpentries
 - Introduction to the Unix Shell and Version Control with Git
- (<https://ucsbcarpentry.github.io/workshop/2024/10/22/ucsb-git.html>)

Batch Computing

- As computational and data requirements grow, researchers may find that they need to make a transition from local resources (e.g., laptop, desktop) to computer cluster or national HPC system.
- Jobs on these shared resources are typically executed under the control of a batch submission system such as SLURM, PBS, etc.
- Jobs need to be configured so that the application(s) can be run non-interactively and at a time determined by the scheduler.
- The user needs to specify the job duration, hardware requirements, and partition. This is done with a batch script.

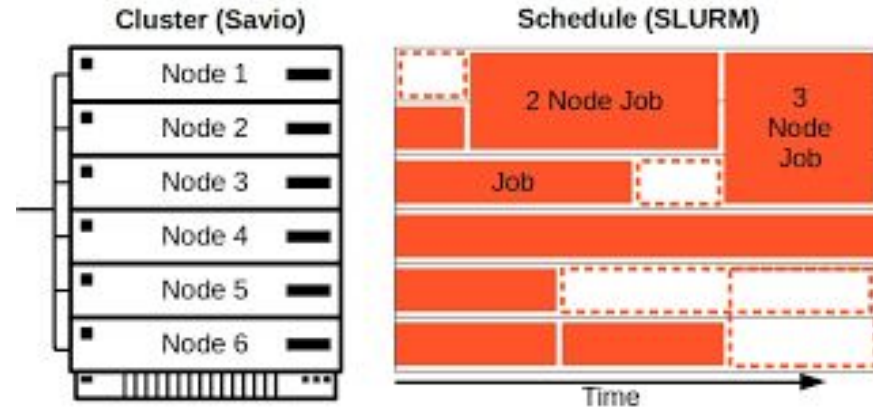
You can't monopolize cluster - limit jobs/cores

Job Submission Script

- When you login to the Cluster, you are on the login node. This node is **NOT** for running BIG calculations!
- All jobs must be submitted to the queue - it just allocate nodes.
- Submission to the queue requires a job script to be written.
- Job script need to specify the resource that you need. There are three basic units:
 - Number of Nodes
 - Number of Cores
 - Time (Optional)
- Other resource you might need to add such as: job name, memory, reminder email, etc.

Simple Scheduling Algorithms

- Backfilling
 - The scheduler maintains the "First Come, First Serve" concept without preventing long-running jobs from executing.
 - The scheduler checks whether the first job in the queue can be executed:
 - If true, the job is executed without further delay.
 - If false, the scheduler looks for the next job that can be executed without extending the waiting time of the first job in the queue and runs it.
 - Jobs that only need a few computing resources are easily "backfillable."
 - Small jobs will usually encounter shorter queue times.



Ref:

<https://docs-research-it.berkeley.edu/services/high-performance-computing/user-guide/running-your-jobs/why-job-not-run/>

Simple Slurm Job Submission script

- There are three simple Slurm script files in your directory: **slurm-mpi2.job**, **slurm-mpi.job**, and **slurm-serial.job**.

Slurm job script file: **slurm-serial.job**

```
#!/bin/bash -l
#SBATCH --nodes=1 --ntasks-per-node 1
module load MatLab/R2018a
cd $SLURM_SUBMIT_DIR
/bin/hostname
matlab -nodisplay -nodesktop -nosplash <
my-inputfile.m
```

Slurm job script file: **slurm-mpi.job**

```
#!/bin/bash -l
#SBATCH --nodes=1 --ntasks-per-node 12
module load intel/18
cd $SLURM_SUBMIT_DIR
/bin/hostname
mpirun -np $SLURM_NPROCS ./a.out >& logfile
```

\$ sbatch slurm-serial.job 'or' \$ sbatch -p short slurm-serial.job

Example Slurm Job Submission script

Slurm job script file: job.s

```
#!/bin/bash
#SBATCH -J 'testJob'
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH -p gpu
#SBATCH --gres=gpu:1
#SBATCH -o outLog
#SBATCH -e errLog
#SBATCH -t 00:10:00
#SBATCH --mail-user=username@ucsb.edu
#SBATCH --mail-type ALL

module purge all
module load anaconda
source activate my_env_1

cd $SLURM_SUBMIT_DIR/

python my_python.py

### Set linux shell: Telling the shell to run the script using the batch
### Job Name
### No. of Nodes
### No. of Tasks
### Submit the job to Partition
### Request 1 GPU
### Output Log File (Optional)
### Error Log File (Optional but suggest to have it)
### Job Execution Time
### Mail to you (Optional)
### Mail send you when the job starts and end (Optional)

### Load softwares that the job depends on to execute

### Absolute path of the current working directory when you submit the job
```

Example Slurm Job Submission script (VASP)

(note - VASP is licensed per group)

Slurm job script file: vasp.job

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks=10
#SBATCH -o outLog
#SBATCH -e errLog
#SBATCH -t 00:10:00
#SBATCH --mail-user=usernam@ucsb.edu
#SBATCH --mail-type ALL

module purge all
module load intel/18
cd $SLURM_SUBMIT_DIR/

ulimit -s unlimited

(time mpirun -np $SLURM_NTASKS ~/bin/vasp/vasp.5.4.4.std ) >& logfile
```

Set linux shell: Telling the shell to run the script using the batch
No. of Nodes
No. of Tasks
Output Log File (Optional)
Error Log File (Optional but suggest to have it)
Job Execution Time
Mail to you (Optional)
Mail send you when the job starts and end (Optional)

Absolute path of the current working directory when you submit the job

Needed for many VASP runs (if you see segfault, etc. errors, try this)

Example Slurm Job Submission script

Slurm job script file: job.s

```
#!/bin/bash                                     ### Set linux shell: Telling the shell to run the script using the batch
#SBATCH -J 'downloadFile'                       ### Job Name
#SBATCH --nodes=1                               ### No. of Nodes
#SBATCH --ntasks=1                             ### No. of Tasks
#SBATCH -o outLog                               ### Output Log File (Optional)
#SBATCH -e errLog                               ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                             ### Job Execution Time
#SBATCH --mail-user=userid@ucsb.edu             ### Mail to you (Optional)
#SBATCH --mail-type ALL                         ### Mail send you when the job starts and end (Optional)

declare -xr SCRATCH_DIR="/scratch/scratch_jay"
cd "${SCRATCH_DIR}"
git clone https://github.com/YoongiKim/CIFAR-10-images.git
tar -czf CIFAR-10-images.tar.gz CIFAR-10-images/

cp CIFAR-10-images.tar.gz $SLURM_SUBMIT_DIR/
cd $SLURM_SUBMIT_DIR/                           ### Absolute path of the current working directory when you submit the job
tar -xvf CIFAR-10-images.tar.gz
```

How to Submit and Monitor Your Job

- Once you have a job script, you may submit this script to SLURM using the sbatch command. SLURM will find an available compute node or set of compute nodes and run your job there, or leave your job in a queue until some resources become available.

```
$ sbatch job.s
```

```
Submitted batch job 1234567
```

- List all current jobs from the user.

```
$ squeue -u your_user_name
```

```
$ squeue -u $USER
```

```
$ showq your_user_name
```

- Stop and delete the Job

```
$ scancel 1234567
```

How to Submit and Monitor Your Job

- List all partitions on the cluster

```
$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
batch*	up	32-00:00:0	1	down*	node1
batch*	up	32-00:00:0	1	drng	node20
batch*	up	32-00:00:0	1	drain	node4
batch*	up	32-00:00:0	30	mix	node[6,8-9,11,13-16,19,22-23,26-28,32,36-39,42-44,49,52-56,58,60]
batch*	up	32-00:00:0	29	alloc	node[3,5,7,10,12,17-18,21,24-25,29-31,33-35,40-41,45-48,50-51,57,59,61-63]
batch*	up	32-00:00:0	1	idle	node2
short	up	2:00:00	1	mix	node64
largemem	up	37-12:00:0	4	mix	node[101-104]
gpu	up	7-00:00:00	1	down*	node117
gpu	up	7-00:00:00	13	mix	node[111-113,115-116,118-125]

- List the partition who are using

```
$ squeue -p short
```

- Report the job expected start time

```
$ squeue --start -j job_ID
```

Running Jobs on Pod (Slurm)

- Start/submit a job: `$ sbatch job.s`
- Check status of the running jobs: `$ squeue -u user_name`
`$ showq user_name`
- Delete a running job: `$ scancel job_id`
- Available partition:
 - Short partition: running under 2 hrs
 - `#SBATCH -p short`
 - Large memory partition: running the longest 37 days
 - `#SBATCH -p largemem`
 - GPU partition: running the longest 10 days
 - `#SBATCH -p gpu`

Wrap up

- What is the Center for Scientific Computing (CSC) at UCSB?
- Introduction to High-Performance Computing (HPC) at UCSB
- Connect to the HPC Cluster
- Frequently Linux Commands
- File Transfer
- Conda Environment
- Modules on the Cluster
- SLURM Commands

Ack!



- Acknowledgements - <https://csc.cnsi.ucsb.edu/publications>

Please acknowledge the CSC in publications and presentations if you are using our facility's computational resources (including staff involvement) in your research.

“We acknowledge support from the Center for Scientific Computing from the CNSI, MRL: an NSF MRSEC (DMR-2308708) and NSF CNS-1725797.”

For users of GPU nodes, please add the grant number NSF OAC-1925717

Questions and Thought

- What else content should we cover?
- Other ideas for a workshop?
 - Running Parallel Python / Matlab / R on the Cluster, Mathematica, Lumerical, Singularity/Docker Container, etc.
- More Information:

<https://csc.cnsi.ucsb.edu/>