UC SANTA BARBARA

CNSI MRL

CSC Center for Scientific Computing

UC SANTA BARBARA Information Technology

HPC Workshop 1

Nov. 04 and 05 , 2025

11:30 – 12:30 pm (followed by pizza)

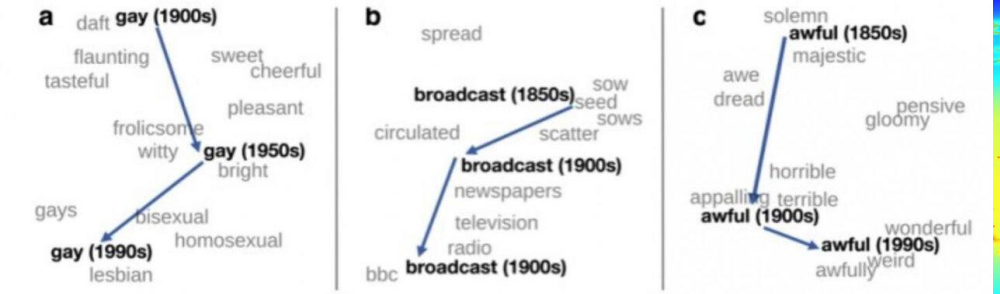
Location: Elings Hall 1605

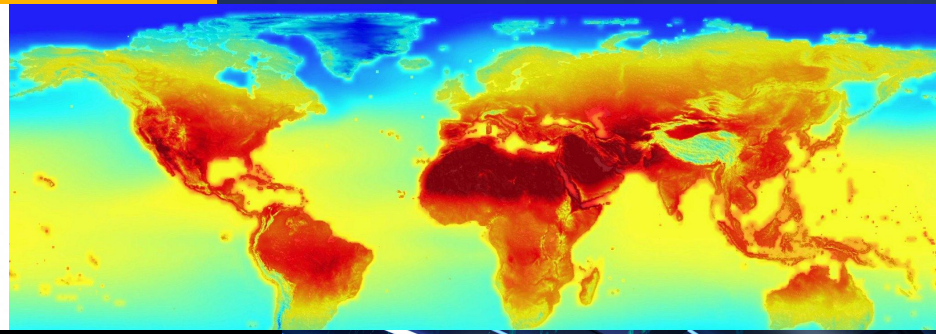
Register @ https://csc.cnsi.ucsb.edu

Quickly start using HPC resource at UCSB

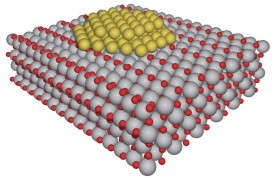
- What is HPC?
- Quickly get Started to Use HPC
- Basic Linux Commands
- Basic Slurm Commands
- Producible and Portable Software Environment

Computational Linguistics





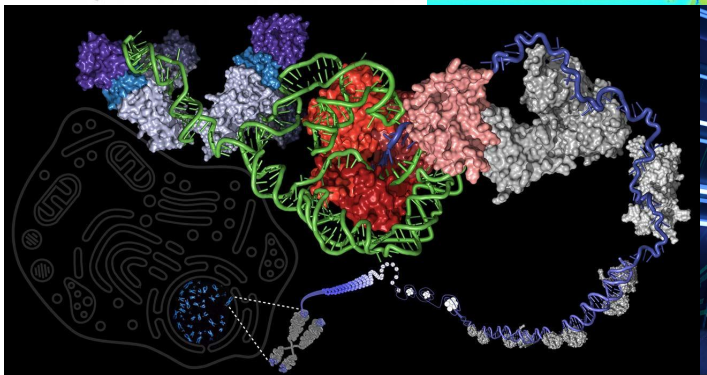
KS-DFT



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

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

Kohn-Sham kinetic energy | External energy | XC energy | Hartree energy



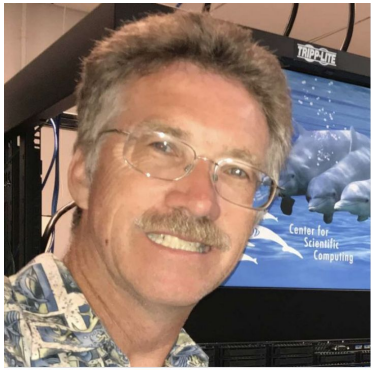
UC **SANTA BARBARA**

Introduction to High-Performance Computing (HPC)

Paul Weakliem, Fuzzy Rogers and Jay Chi

November 04 and 05, 2025

Ye Olde People Introductions



Paul Weakliem, PhD
Co-Director
Center for Scientific Computing &
California Nanosystems Institute
Elings 3231
weakliem@cnsi.ucsb.edu

Fuzzy Rogers
That guy in the MRL
Center for Scientific Computing &
Materials Research Laboratory
MRL 2066B
fuz@ucsb.edu

Yu-Chieh “Jay” Chi, PhD
Research Computing Consultant
Center for Scientific Computing &
Information Technology Services
Elings 3213
jaychi@ucsb.edu

CSC Center for Scientific Computing

CNSI

UC SANTA BARBARA MRL Materials Research Laboratory
NSF Materials Research Science and Engineering Center [DMR 1720256]

UC SANTA BARBARA Information Technology

UC SANTA BARBARA

Introduction *(and disclaimer regarding slides!)*

Many of you are here because your computational project workloads have increased beyond the capacity of your local resources, such as laptops or desktops. Specifically, you may consider utilizing a high-performance computing (HPC) cluster.

- Generally used for very large scale calculations, or many medium scale calculations. ***Primarily accessed via command line (CLI)***
- 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.***
- The actual HPC hardware is probably important
  - Hardware is important but not that interesting

Introduction

- In this workshop, you will learn
  - What is the Center for Scientific Computing (CSC) at UCSB?
  - Introduction to High-Performance Computing (HPC) at UCSB
  - Amdahl's Law
  - Quickly get started to use cluster
  - File Transfer
  - Learn the basic of Linux Commands
  - 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.

Center for Scientific Computing (CSC) Facilities

- High-Performance Computing (HPC) Cluster
  - Interconnected Linux servers & limited storage with high speed interconnect
  - Specialize node - GPUs, Large Memory
  - Resource management done automatically via SLURM
  - ***Long running calculation - upwards of 30+ days***

What is Center for Scientific Computing (CSC)

Support Capabilities

- We provide the HPC computational infrastructure.
- We provide workshops, training, consultation, and knowledge bases to assist researchers.
- We provide user support, assistance with use of resource, and installation of applications.
- We work with your local IT staff to provide ***effective*** 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 offen.

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!!! (see https://rcd.ucsb.edu for other options!!)

- 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 …….

CSC Cluster Resource

- POD (Pod of Dolphins) - pod.cnsi.ucsb.edu
  - FREE … but Campus available, usually pretty busy
  - 64 CPU Compute nodes, 40 cores per node with 192GB RAM
  - 4 Large Memory nodes, 40 cores with 768 GB (up to 1.25TB) RAM
  - 13 GPU nodes, 4 * 32 GB Nvidia V100s & 24 cores per node with 192GB RAM
  - 1 GPU Development node with Nvidia P100 and T4 GPU
  - ***Published papers should acknowledge CSC -*** https://csc.cnsi.ucsb.edu/publications
- Condo Clusters - Braid & Braid2 - braid.cnsi.ucsb.edu & braid2.cnsi.ucsb.edu
  - Individual Faculty (PI) buy-In for access, CSC handles infrastructure
  - Braid: ~80 nodes, mix 12-32 cores, 64GB-192GB RAM, 3 4*Nvidia P100
  - Braid2: ~60 nodes, 24-64 cores, 256GB - 1TB RAM, 8 4*Nvidia A100

  UCSB VPN is required to connect to the campus cluster from off-campus

- ACCESS (when you outgrow CSC, or other local resources)
  - National Supercomputer Center - ACCESS-CI https://access-ci.org
  - PSC, SDSC, TACC, JetStream2, etc.

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?

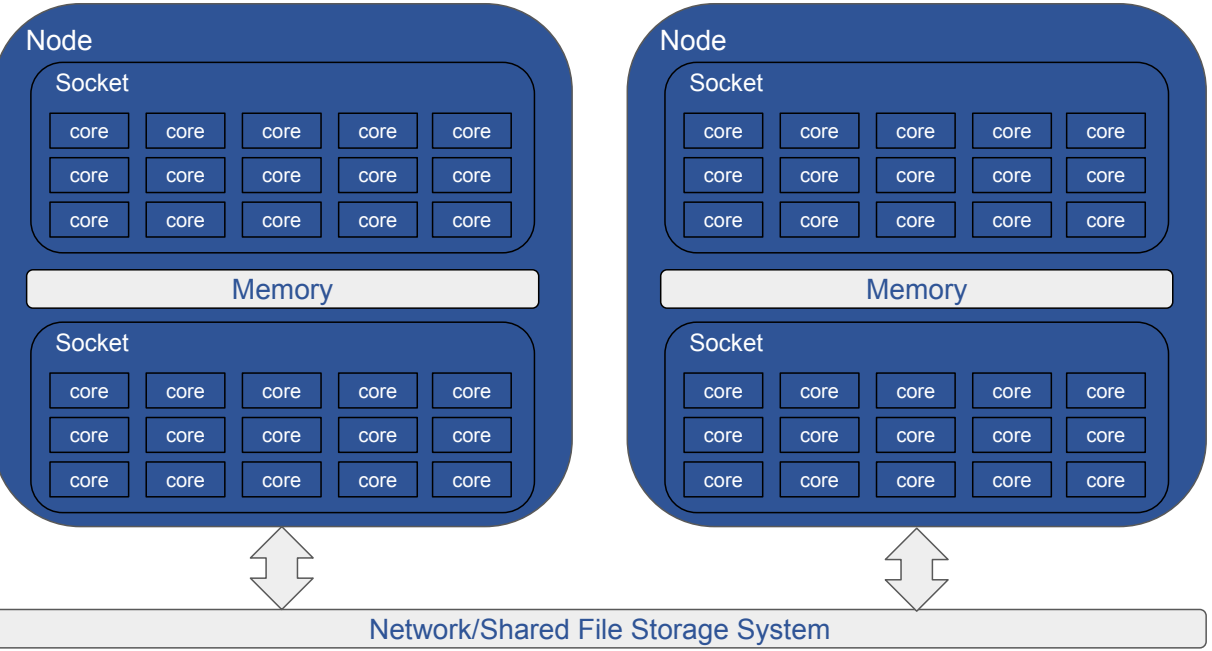
Network mounted scratch file Storage: /scratch

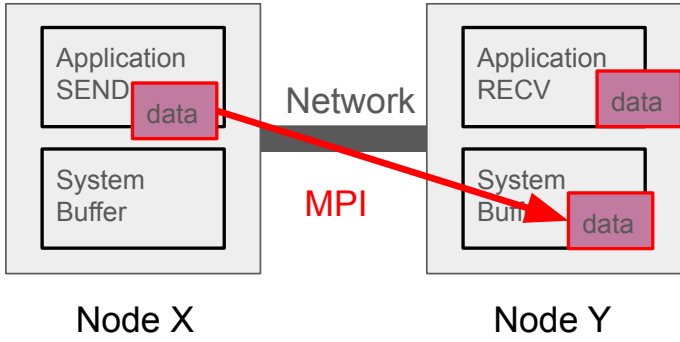
- High speed but temporary files (purged after 90 days)
- 19 TB NFS mounted file system (/scratch)
- 70 TB NFS mounted file system (/bigscratch)

/csc/central - ‘near-line’

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

HPC Infrastructure & Message Passing Interface (MPI





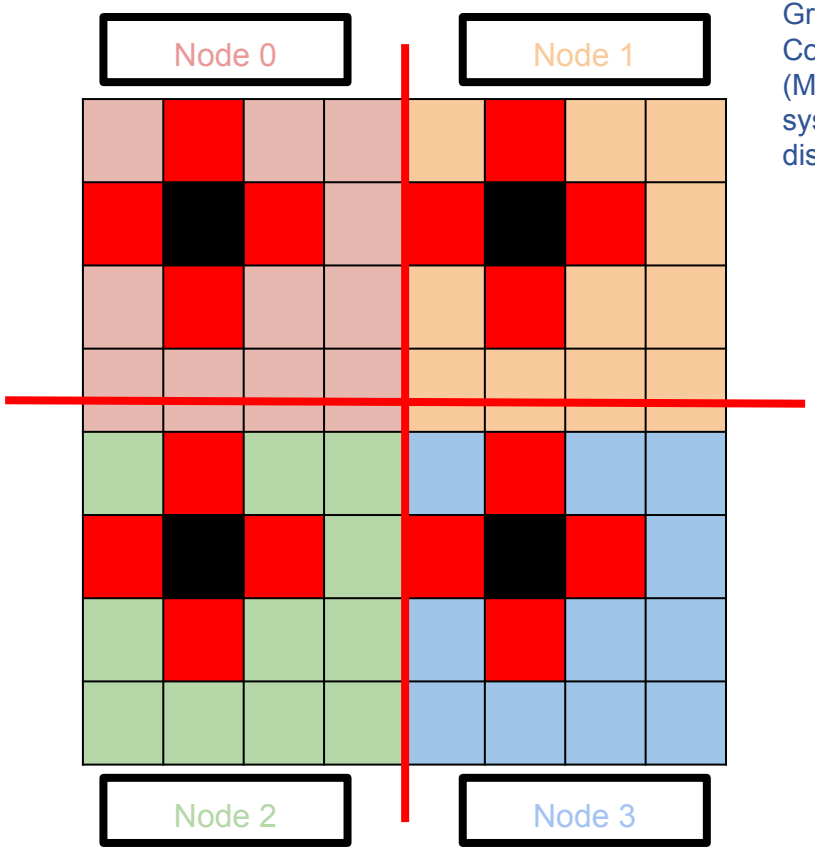
High-Performance Computing Myths

Myth #1: High-Performance computing is for astrophysicists, engineers, climate modelers and others working in traditionally math intensive fields

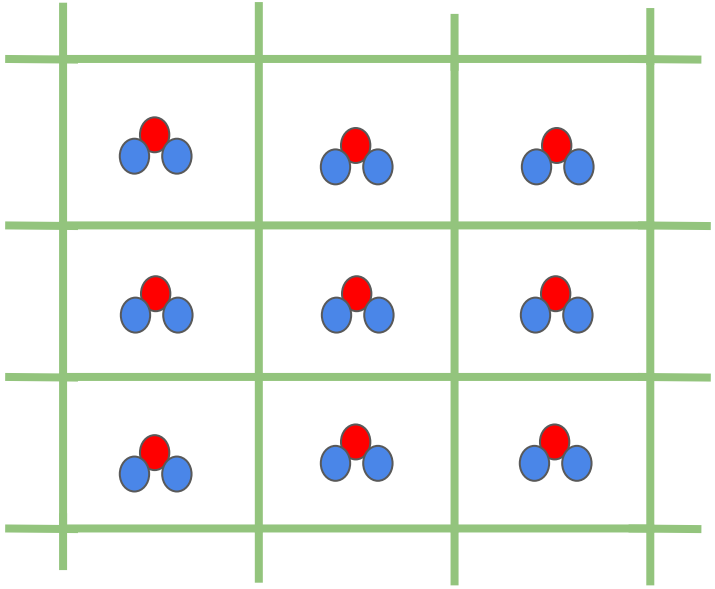
Myth #2: Throwing more hardware at a problem will automatically reduce the time to solution

Myth #3: You need to be a programmer or software developer to make use of high-performance computing

Communications Overhead



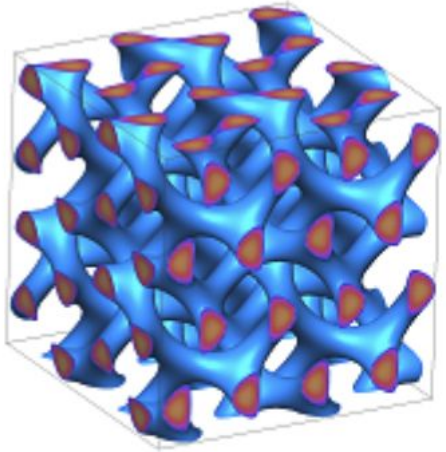
Grid Mesh Partition:
Computational fluid dynamics (CFD), magnetohydrodynamics (MHD), climate, weather and other simulations involve solving system of partial differential equation (PDE) on a grid that is distributed across processors to achieve parallelization.



Periodic Boundary Conditions:
Implementing periodic boundary conditions (PBCs) with MPI requires each process to communicate data with its neighboring processes on the opposite side of the simulated domain.

Examples of some of the science done on CSC HPC clusters

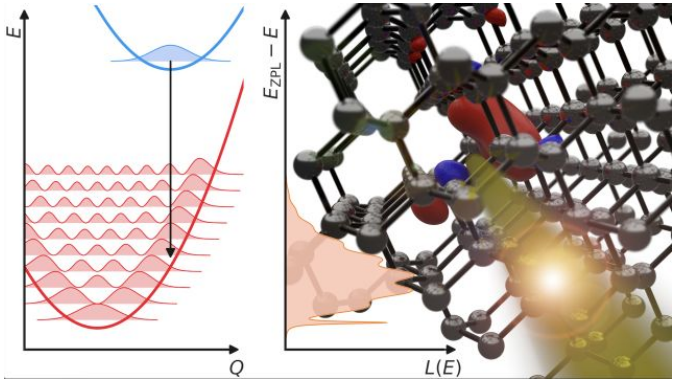
Field-Theoretic Simulation *(soft materials, polymers)



Typical calculation on GPUs, running for days.

https://mrlweb.mrl.ucsb.edu/~fredrickson/research.html

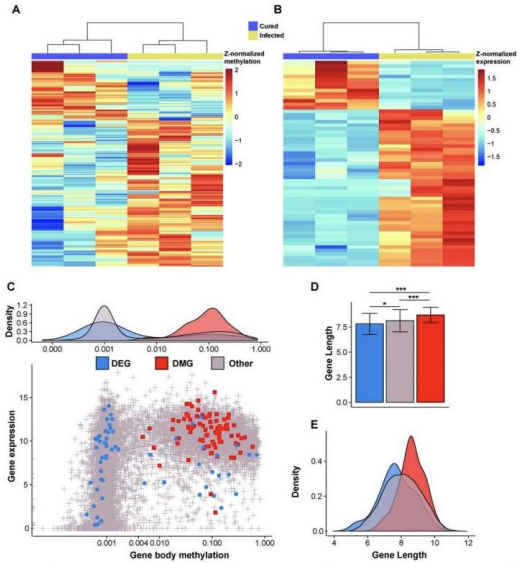
Molecular Dynamics to understand photonics and LEDs.



Typical calculations 120 cores for days

https://vandewalle.materials.ucsb.edu/

Ecological & Evolutionary Epigenomics



Typical requirements for large memory node

https://yilab.eemb.ucsb.edu/research/ecological-evolutionary-epigenomics

Parallel Computing Performance

Let's consider the parallel speedup in relation to our ideal expectation.

- Number of processors = N
- Serial Run-Time = Ts
- Parallel Run-Time = Tp

$$T_p = T_S/N$$

- The speedup of the parallel program is

$$S = \frac{T_s}{T_p}$$

Parallel Computing Performance

However, our expected performance might not be approached due to the computer hardware overhead. The Parallel Run-Time should be considered as

$$T_p = (T_s * P)/N + T_o$$

- P is the fraction of the code that can be parallelized
- To is the Run-Time unparalleled Overhead that can be written as

$$T_o = (1 - P) * T_s$$

Amdahl's Law and Limits on Scalability

Amdahl's law describes the absolute limit on the speedup of a code as function of the proportion of the code that can be parallelized and the number of processors. This is the most fundamental law of parallel computing.

$$S = \frac{1}{(1-P)+\frac{P}{N}}$$

- S is the speedup
- P is the fraction of the code that can be parallelized
- N is the number of processors

Amdahl's Law and Limits on Scalability

In the limit as the number of processors goes to infinity, the theoretical speedup depends only on the proportion of the serial content

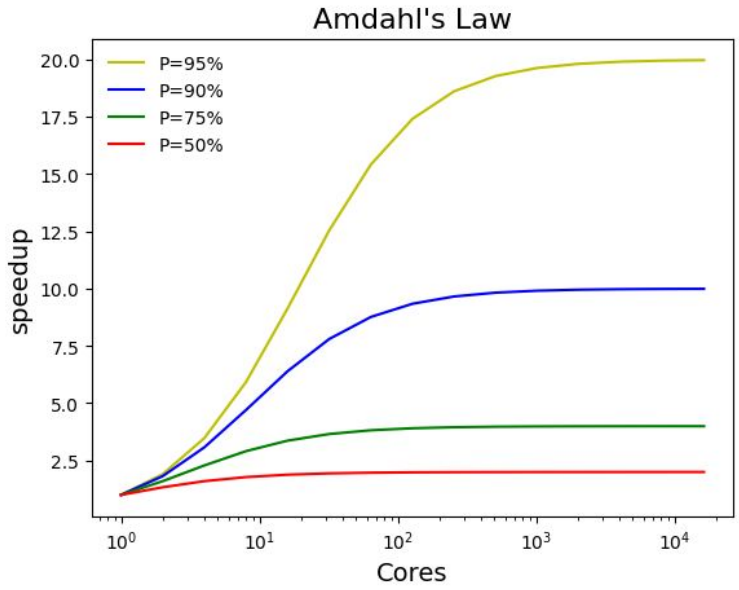
$$\lim_{N\to\infty} = \frac{1}{(1-P)+\frac{P}{N}} = \frac{1}{(1-P)}$$

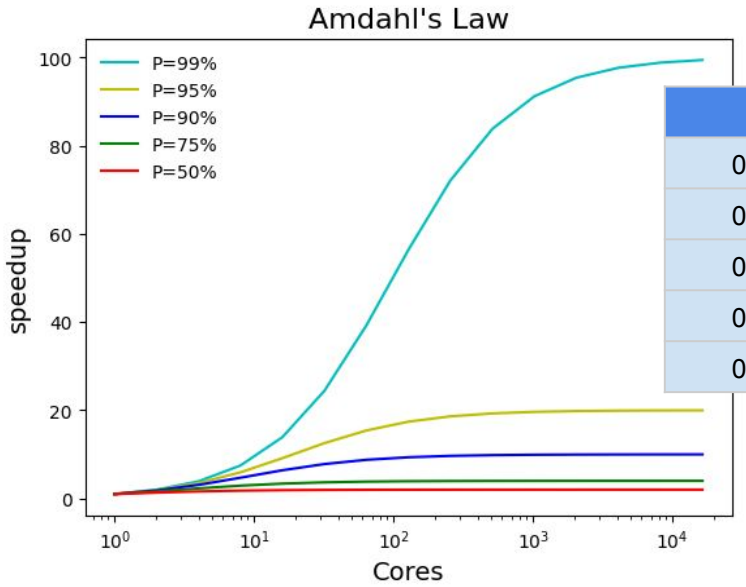
That does not look so bad, but as we will show in the next slide it does not take much serial content to quickly impact the speedup.

Unless virtually all of a serial program is parallelized, the possible speedup will be limited — regardless of the number of cores available.

Amdahl's Law and Limits on Scalability

- Unless all of a serial program is parallelized, the possible speedup will be limited — regardless of the number of cores available.





P	Max Speedup
0.50	2
0.75	4
0.90	10
0.95	20
0.99	100

Other Limits on Scalability

Amdahl's law sets a theoretical upper limit on speedup, but there are other factors that affect scalability:

- Problem Size
- Communication Overhead
- Uneven Load Balancing

In real-life applications that involve communications, synchronization (all threads or processes must complete their work before proceeding) or irregular problems (non-Cartesian grids), the speedup can be much less than predicted by Amdahl's law.

Scalability

- In general, a problem is scalable if it can handle ever increasing problem size.
- If we increase the number of processes and keep the efficiency fixed without increasing problem size, the problem is strongly scalable.
- If we keep the efficiency fixed by increasing the problem size at the same rate as we increase the number of processes, the problem is weakly scalable.

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/
- Transferring Data with Globus
  - https://www.globus.org/
- 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(research computing and data) https://rcd.ucsb.edu

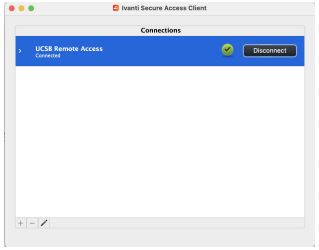
Key points to Use Cluster

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

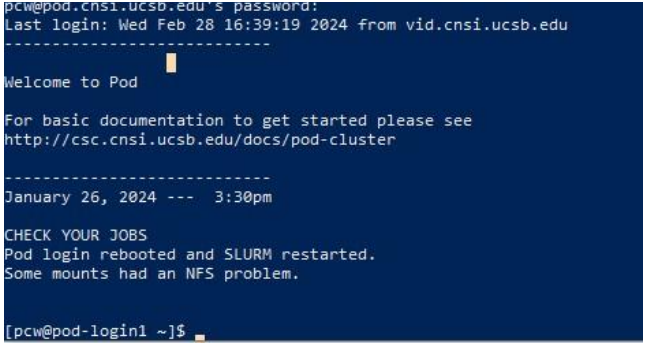
- What does it mean?
- It means you are sharing these two login nodes with many other users when you are login. Running intensive programs on the login node will cause the login nodes to be slow for all users.
  - Login nodes are for editing files, transfering files, changing permissions, submitting 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

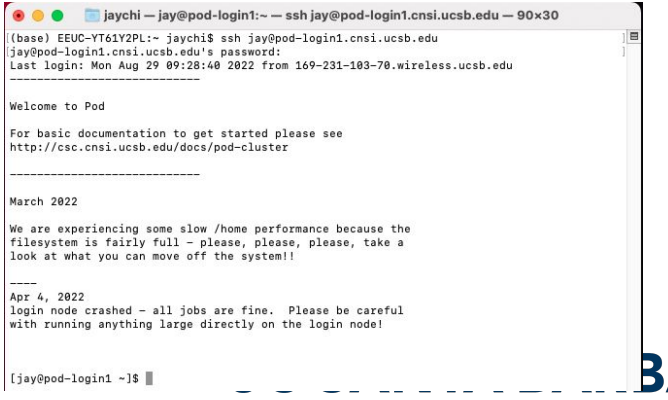
You can’t monopolize cluster - limit jobs/cores

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 either - use the ssh command
  - $ ssh your_user_name@pod.cnsi.ucsb.edu
- ***UCSB VPN is required to connect to the campus cluster from off-campus***
  - https://www.ets.ucsb.edu/pulse-secure-campus-vpn/get-connected-campus-vpn

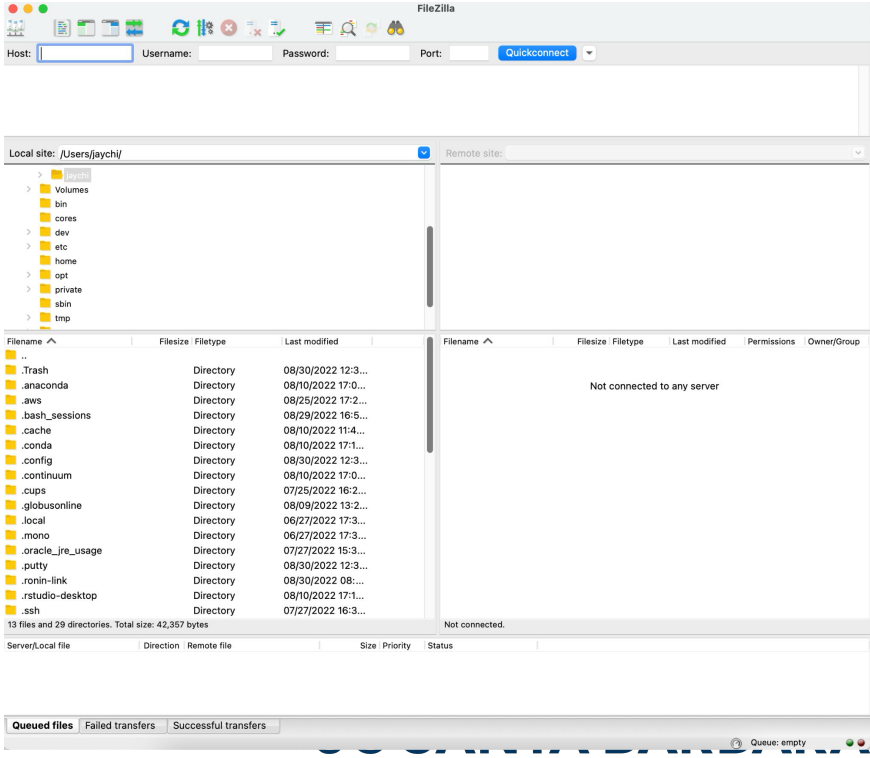




File Transfer

- 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

Filezilla



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.

More Linux Resource Information

- UCSB Software Carpentries
  - Introduction to the Unix Shell

    (https://carpentry.library.ucsb.edu/workshop/2025/10/07/ucsb-shell.html)

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/csc, 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/R2021b MatLab/R2025a

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 (Python)

- 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

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.

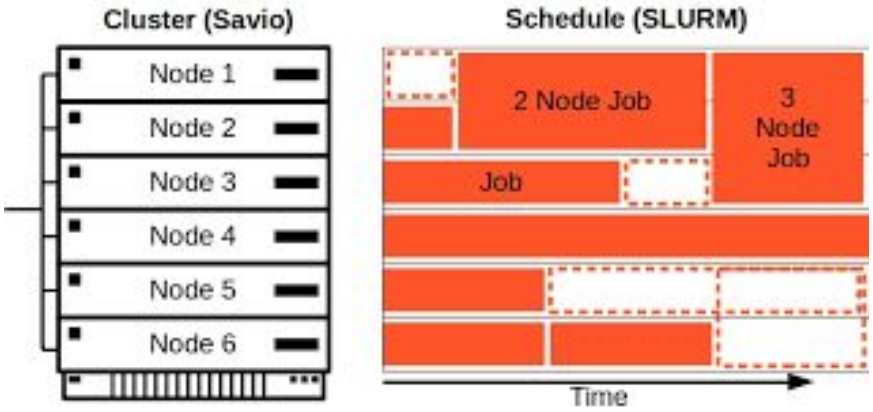
You can't monopolize cluster - limit jobs/cores

Job Submission Script

- 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/R2021b

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                                  ### Set linux shell: Telling the shell to run the script using the batch
#SBATCH -J 'testJob'                         ### Job Name
#SBATCH --nodes=1                            ### No. of Nodes
#SBATCH --ntasks=1                           ### No. of Tasks
#SBATCH -p gpu                               ### Submit the job to Partition
#SBATCH –gres=gpu:1                          ### Request 1 GPU
#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=usernam@ucsb.edu         ### Mail to you (Optional)
#SBATCH --mail-type ALL                      ### Mail send you when the job starts and end (Optional)

module purge all
module load anaconda                         ### Load softwares that the job depends on to execute
source activate my_env_1

cd $SLURM_SUBMIT_DIR/                        ### Absolute path of the current working directory when you submit the job

python my_python.py
```

Example Slurm Job Submission script (VASP)

(note - VASP is licensed per group)

Slurm job script file: vasp.job

```
#!/bin/bash                              ### Set linux shell: Telling the shell to run the script using the batch
#SBATCH --nodes=2                        ### No. of Nodes
#SBATCH --ntasks=10                      ### 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=usernam@ucsb.edu     ### Mail to you (Optional)
#SBATCH --mail-type ALL                  ### Mail send you when the job starts and end (Optional)

module purge all
module load intel/18
cd $SLURM_SUBMIT_DIR/                    ### Absolute path of the current working directory when you submit the job

ulimit -s unlimited                      ###  Needed for many VASP runs (if you see segfault, etc. errors, try this)

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

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=usernam@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/jay/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

Acknowledgement

- 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 / Mathematica / Lumerical on the Cluster, Singularity/Docker Container, etc.

- More Information:

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