New User Seminar

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

Be Boulder.

Research Computing New User Seminar

Instructor:

- Website: www.rc.colorado.edu
- Helpdesk: rc-help@colorado.edu
- Slides: https://github.com/ResearchComputing/New_User_Seminar

- Survey: http://tinyurl.com/curc-survey18

Account Check

Does anyone ***not*** have a **CU Research Computing account** who would like to use a temporary account*?

**only available during seminar*

Learning Goals

1. What is CU Research Computing (CURC)?
2. Understand Basic Resources
3. Getting an account & logging in
4. Navigate the RC system
5. Run a job
6. Help!

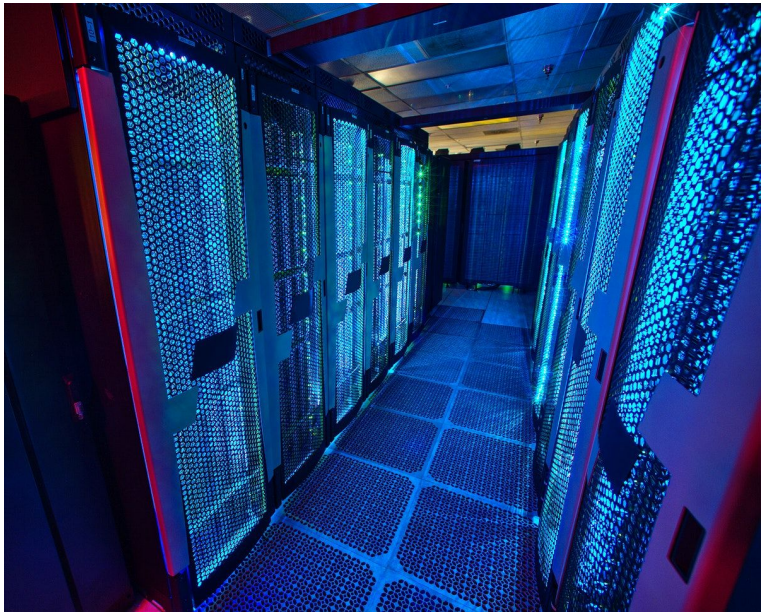
Things to Take Note Of

- Confusing, ambiguous, highly nuanced concepts
- Goal is to avoid common mistakes, pitfalls, and frustrations

Ask Questions!

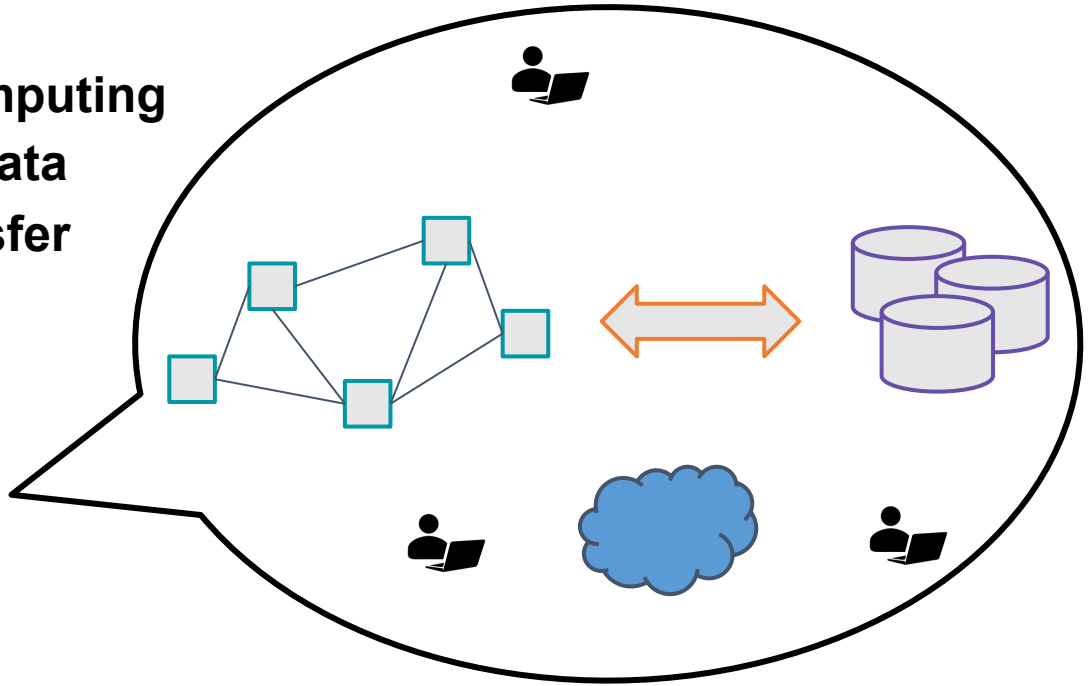
CU Research Computing provides

Large-scale, beyond-the-desktop computing resources for the University of Colorado and more!

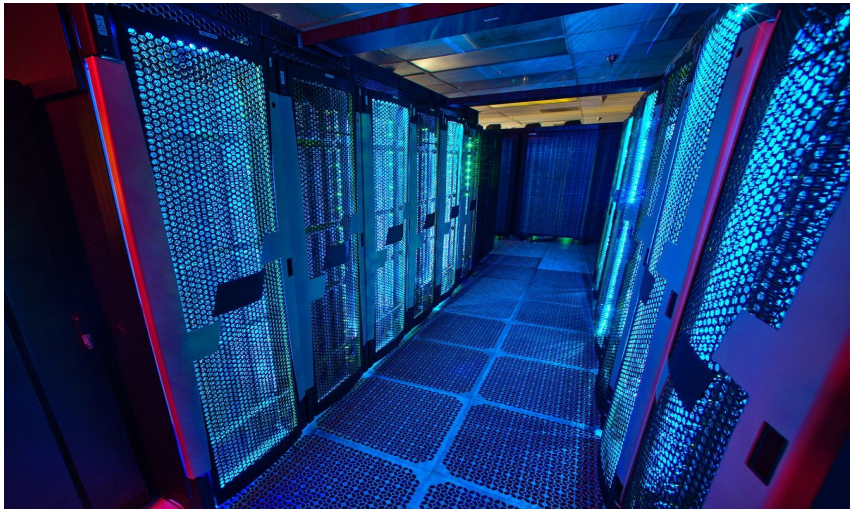


Resources include:

- High Performance Computing
- Storage of Research Data
- High-Speed Data Transfer
- Data Sharing
- Cloud Computing
- Training & Education



Primarily known for:
High Performance Computing (HPC)

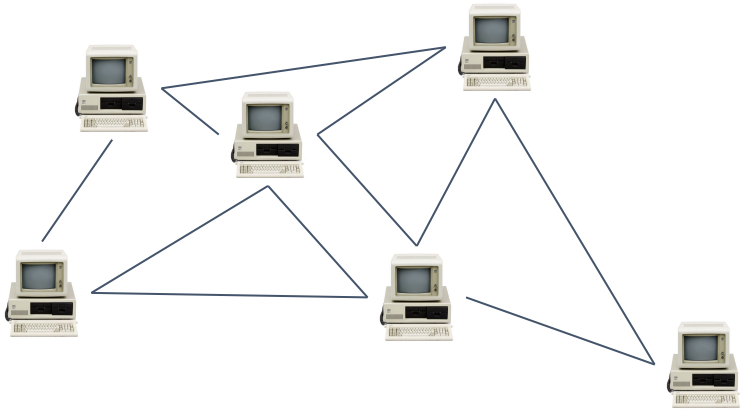


High Performance Computing (HPC) vs. Traditional Computing

- Traditional computing *generally* has access to a single processor (perhaps multiple cores)

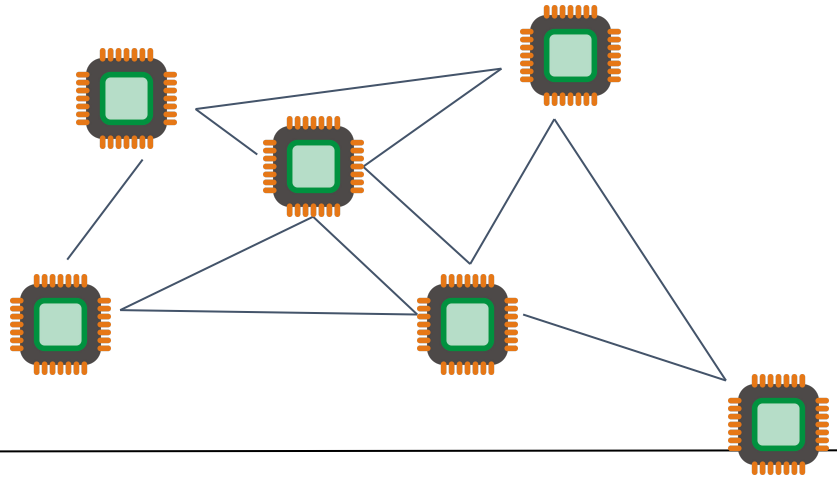
High Performance Computing (HPC) vs. Traditional Computing

- HPC is a network (or cluster) of hardware we call “nodes” linked together with high-speed interconnects



High Performance Computing (HPC)

- HPC is a multiprocessor environment that allows users to run jobs on several processors at once (also called parallel processing)



What can / use HPC for?

- Solving large problems that require more:
  - Memory than you have on your PC
  - cores/nodes/power thank you have on your PC
- Jobs that require hardware you may not have:
  - High Performance GPU computing
  - Specific Operating System
- Visualization rendering

Puzzle Analogy

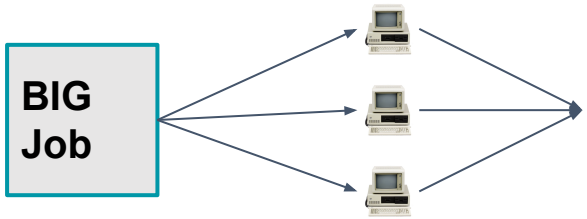
- Working on a puzzle by yourself (although enjoyable!) will take time, even if you're fast!

- You can speed it up by inviting friends over and enticing them with food

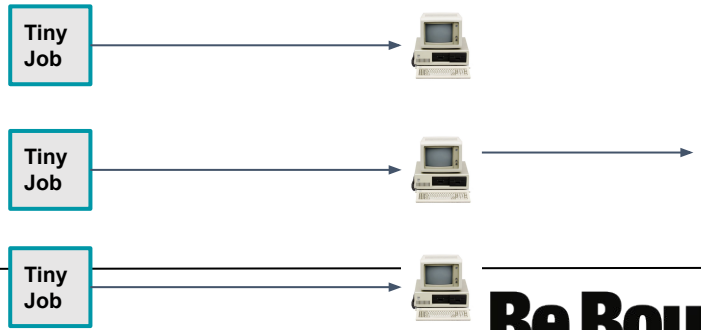
- What happens if there are too many people?

What can / use HPC for?

- Jobs that would take a long time on local machines can instead be distributed over hardware:
  - Parallelized to split up then joined (if software enabled)



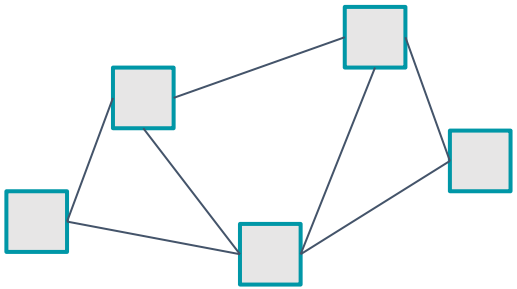
  - Broken up into many serial jobs



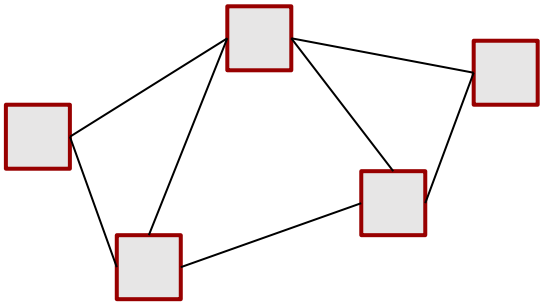
Research Computing Resources

HPC Clusters at CURC

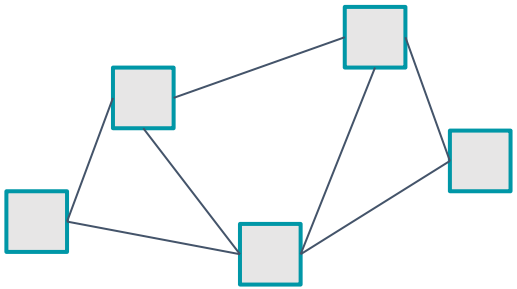
Summit



Blanca

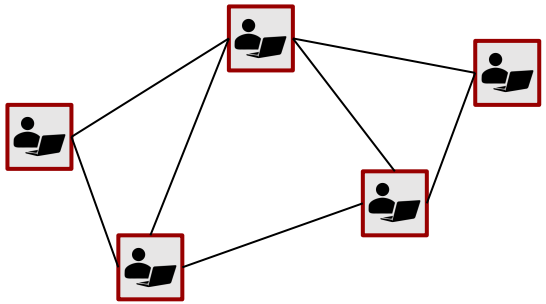


Summit



- NSF-Funded
- Shared
- 450+ Nodes (mostly Intel Xeon Haswell)
  - 24 cores per “shas” (general compute) node
  - Additional Node types:
    - 10 GPU Nodes (NVIDIA Tesla K80)
    - 5 High Memory Nodes (2TB ram)
    - 20 Intel Xeon Phi nodes
  - **11,400 total cores**

Blanca



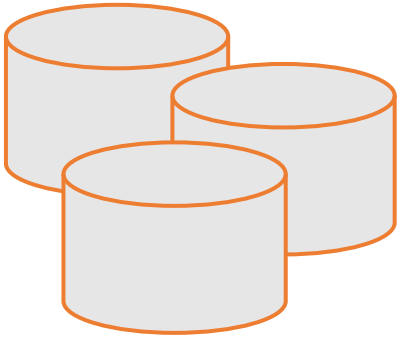
- Buy-in Cluster
  - Per user specs and vendor constraints
- High priority use on your hardware
  - Preemptable use to all other nodes
- Heterogenous
  - ~223 Compute Nodes
  - ~10 GPU nodes

Storage at CURC

Core

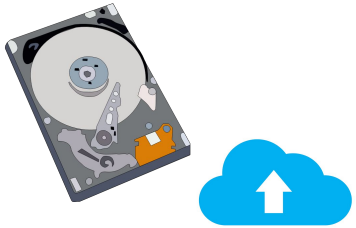
- Included with RC account
  - /home
  - /projects
  - scratch space

PetaLibrary



- Paid Service for:
  - Storage
  - Archive
  - Sharing of research data

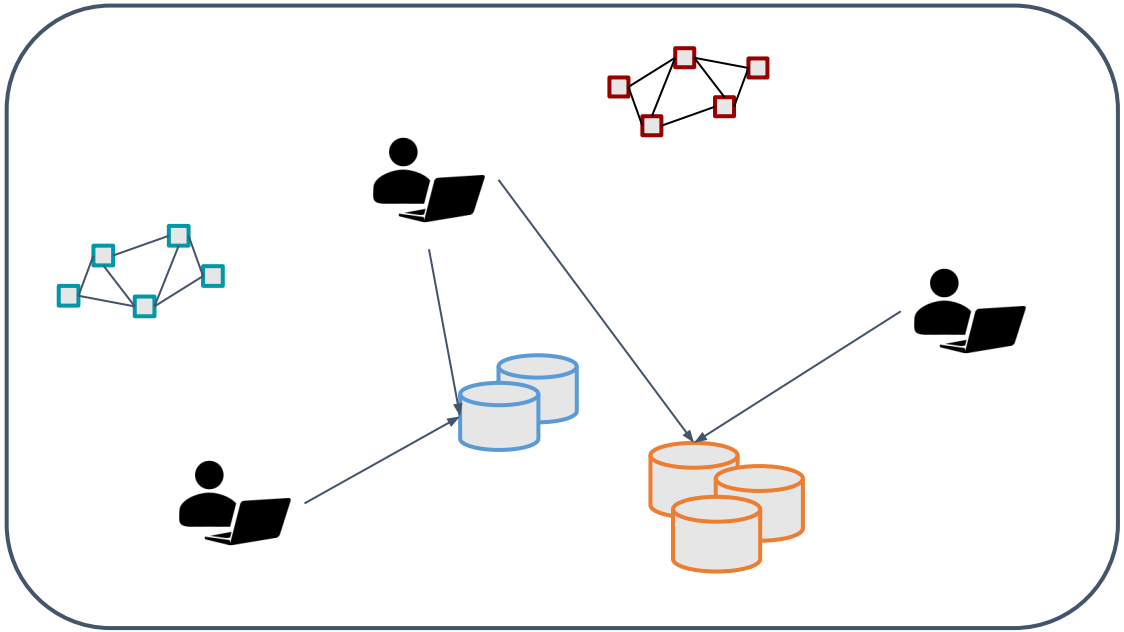
Local or Cloud



- You can download your data locally or to a variety of other cloud resources

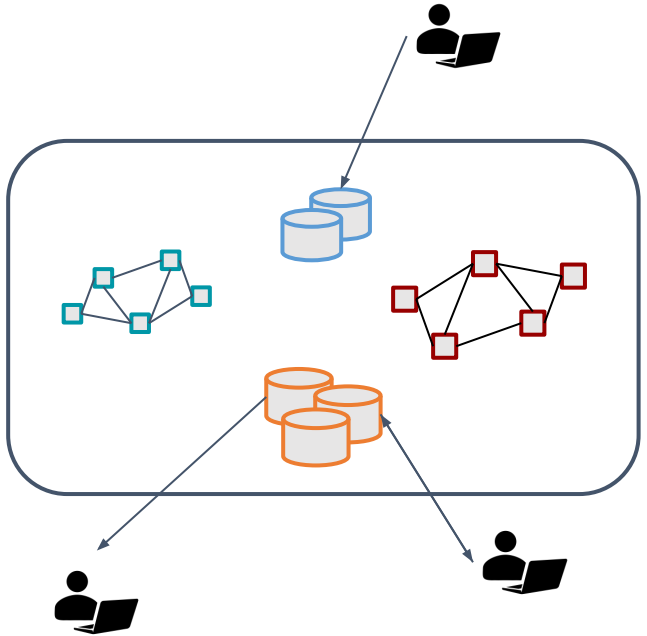
Data Sharing: Within RC

- Sharing workspaces
  - Project space
  - Scratch space
  - PetaLibrary space



Data Sharing: Outside RC

- Globus (recommended):
  - GUI Web Application
  - Automates large transfers
  - Resumes failed transfers
  - Distributes large transfers across DTNs
  - Endpoints that can shared
- Data Transfer Nodes (DTN)
  - Internal CU network needed
- Command line tools
  - scp, sftp, rsync, rclone



Accessing Research Computing

How to Access RC Resources?

1. Get an account
2. Set up two factor authentication
3. (Inform us of any specific needs)
4. Log in
5. Create greatness! (responsibly)

Getting an account

- CU Boulder users and affiliates:
  - Request an account through the RC Account request portal
  - https://rcamp.rc.colorado.edu/accounts/account-request/create/organization

- CSU Users:
  - Request an CSU eID if you don't have one
  - Fill out account application form
  - Duo authentication
  - Then get an RC user account
  - https://www.acns.colostate.edu/hpc/summit-get-started/

- RMACC Users:
  - Contact your local representative, if known. Email rc-help@colorado.edu
  - We'll guide you through the process

Demo: Getting an Account

- CU Boulder users and affiliates:
  - Request an account through the RC Account request portal
  - https://rcamp.rc.colorado.edu/accounts/account-request/create/organization

Your RC Account

Access to:

1. Summit Cluster (soon Alpine)
2. “Core” storage
3. PetaLibary Storage (if applicable)
4. Gateways (CU affiliates)

Two Factor Authentication

- Provides an extra level of authentication
  - We are outside the firewall!
  - Valuable resources
  - Inviting, high-profile target
  - Lost time investigating/fixing

- Duo
  - You will receive a Duo invitation when your RC account is created.

Duo Authentication

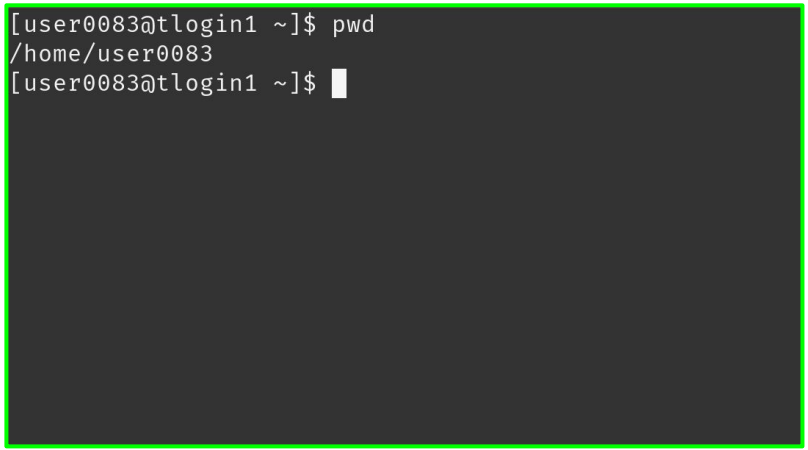
1. **Most users use the Duo smartphone app**
2. “Phone Call” is an alternatives
3. Physical code generator “token” available for $20

Linux comfort level check!

- On a scale from (1-10) **how familiar/comfortable are you with Linux**?
  - The command line
  - Basic commands
  - Linux filesystem
  - Navigating the filesystem

Terminal

- Mac or Linux
  - Terminal application
- Windows
  - PuTTY (recommended)
  - Powershell



- Open OnDemand (*alternative for CU affiliates*)
  - For those less familiar with Linux (ondemand.rc.colorado.edu/)

Demo: Logging in via Terminal

- To login to an RC login node:

```
ssh <username>@login.rc.colorado.edu
```

Supply your IdentiKey password and your Duo app will alert you to confirm the login

---

If you're using a tutorial account:

```
ssh <username>@tlogin1.rc.colorado.edu
```

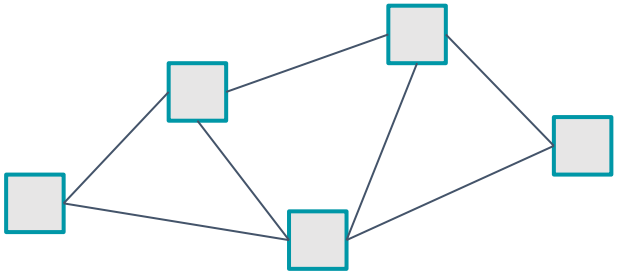
Demo: Logging in with OnDemand

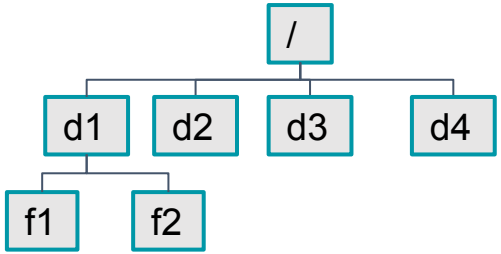
Logging In

- It’s important to note that you are NOT logging into any specific resource, Summit, etc.
- When you log in, you land on our login nodes
- From there, you can access our other resources

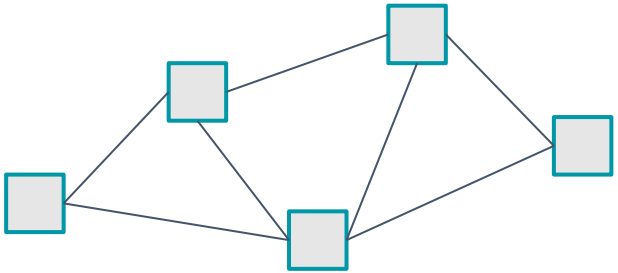
Navigating Research Computing

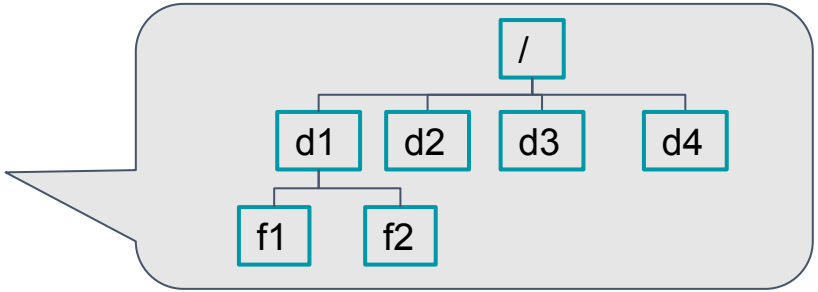
Node	File System
• One computing server • Physical hardware • Work together in parallel	• The basic tree-like layout • From most nodes* you have access to the entire file system





Node	File System
• One computing server • Physical hardware • Work together in parallel	• The basic tree-like layout • From most nodes* you have access to the entire file system





Node Types

Login	Compile	Compute
• Where you start • For editing code, job submission • No heavy computation	• Where you compile code, install packages • Explore the Summit software environment • Edit code, submit jobs • No heavy computation	• Where scheduled jobs run • Intended for heavy computation
Ex. edit job script	Ex. Install python libs	Ex. Running Matlab

Demo: Exploring nodes

- Once logged in:

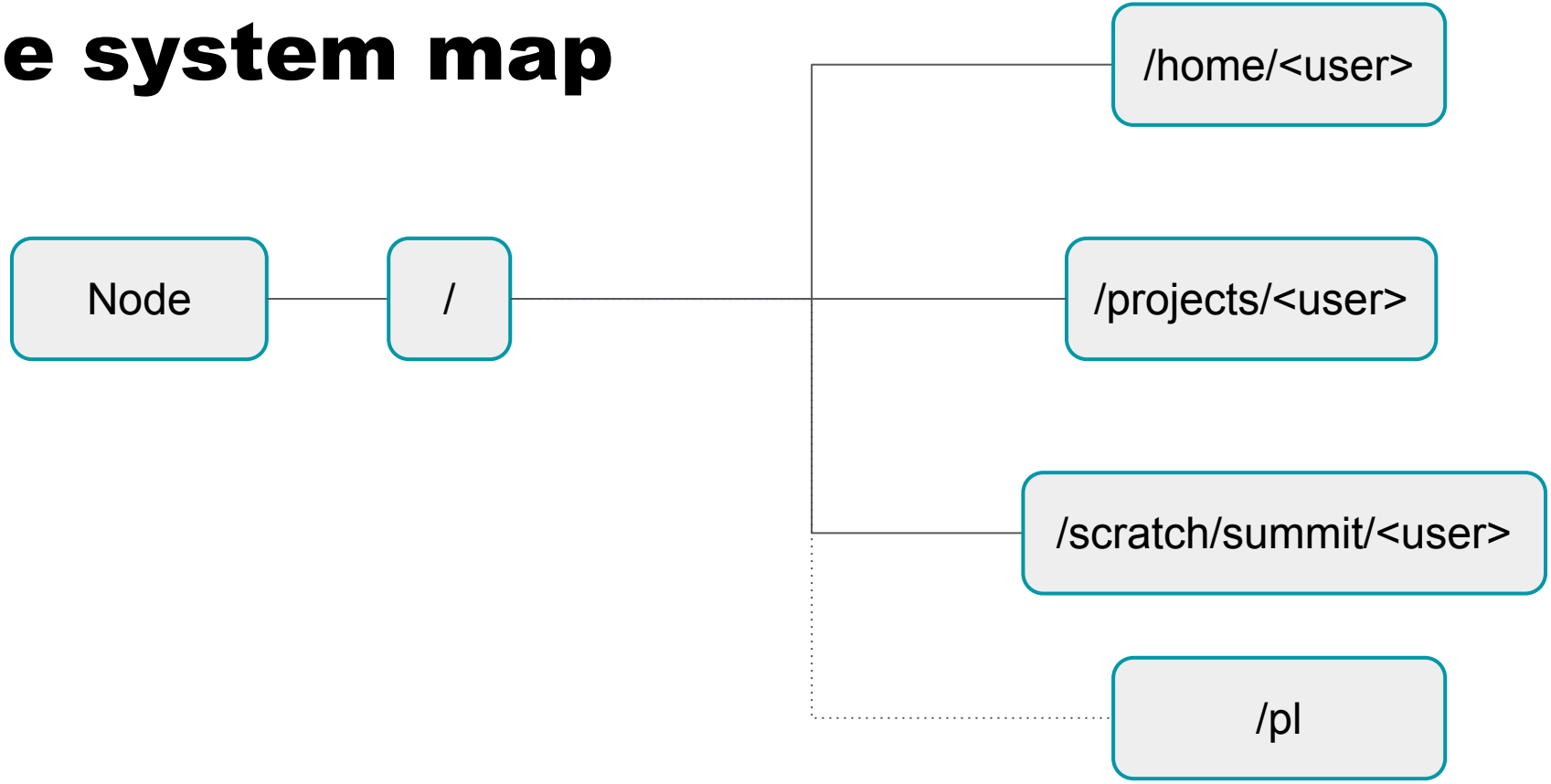
```
$ ssh scompile
```

To log in to a compile (or head) node.

```
$ module avail
```

To check currently available software

File system map



File Structures

/home (2GB)	/projects (250GB)	Scratch (10TB)
• Scripts, Code, Small, important files/directories • Not for sharing files or job output	• Code/files/libraries • Software you are installing • Sharing files • Not for job output	• Output from running jobs • Large files/datasets • Sharing files • Not for long term storage
Ex .bashrc	Ex. Shared job scripts	Ex. Data

Demo: Exploring the Filesystem

- Once logged in:

```
cd  /home/<user>
cd  /projects/<user>
cd  /scratch/summit/<user>
```

To navigate to your different directories

Using Research Computing

- We have logged on
- We have explored nodes and filesystem
- But how do we actually *use* the computing resources?

Running a Job

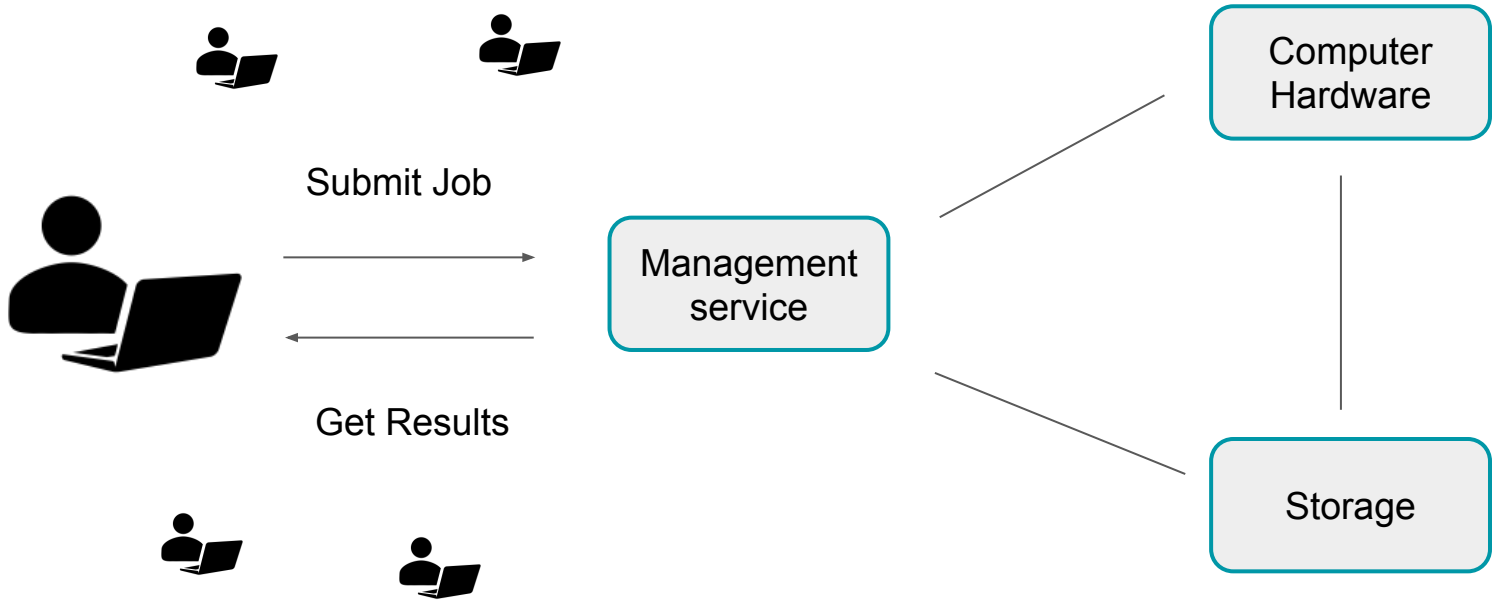
The fundamental “job”

What is a “***job***”?

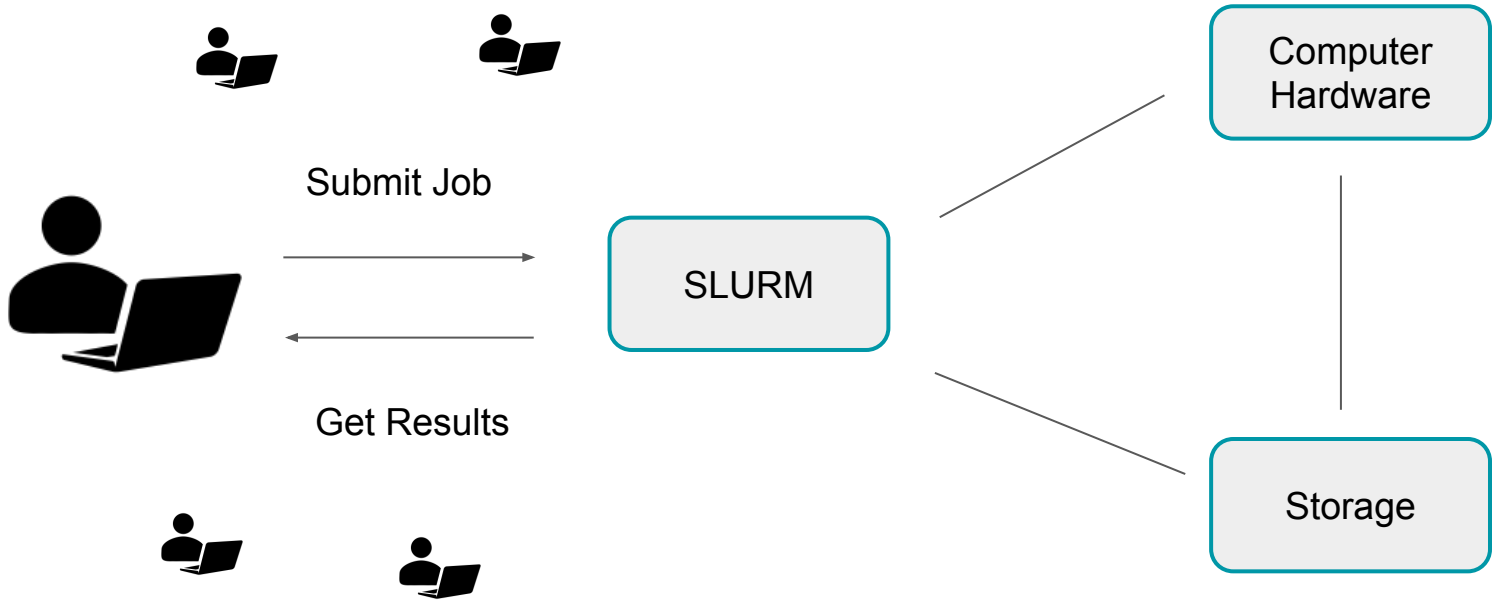
- Work for the cluster to perform on

1. **Batch jobs**
   - Submit job script which will be executed when resources are available
     - Create script containing information about the job
     - Submit the job file to a queue
2. **Interactive jobs**
   - Work interactively at the command line of a compute node

Job Scheduling

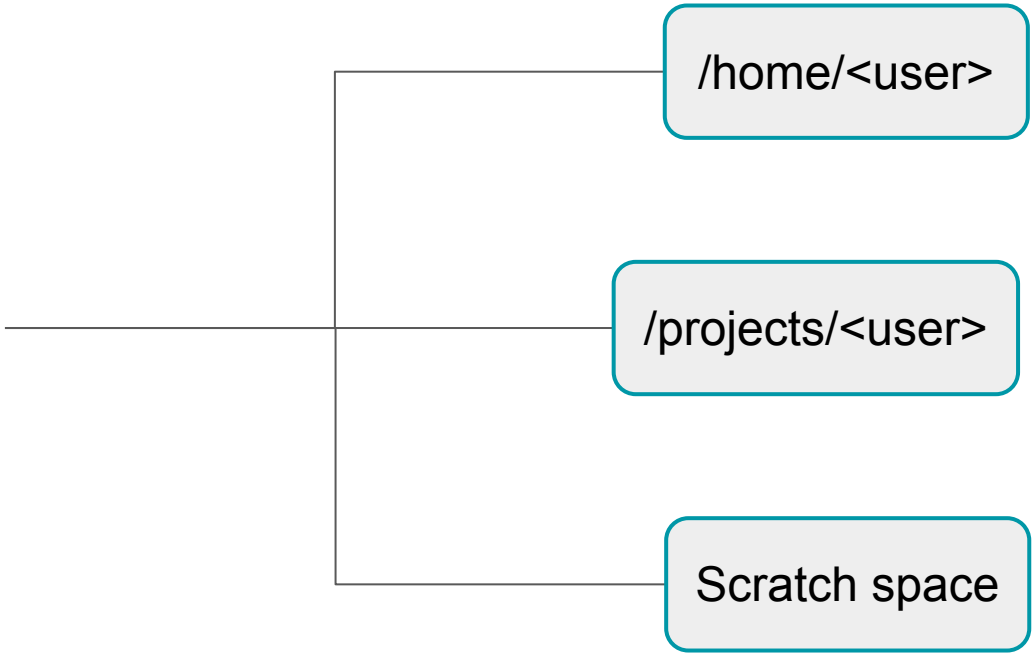


Job Scheduling



Your first job

- Where to write it?
- How to write it?
- How to run it?



Job Script: 3 main parts

1. Directives
   - Specify resource requirements

2. Software
   - Because jobs run on a different node than from where you submitted…
   - …software that is needed must be loaded via the job script

3. User scripting
   - the actual user scripting that will execute when the job runs

```
#!/bin/bash

## Directives
#SBATCH --<resource>=<amount>

## Software
module purge

## User Scripting
```

Directive Options http://slurm.schedmd.com/sbatch.html

```
#SBATCH <options>     sbatch <options>
```

- Allocation: `--account=<account_no>`
- Partition: `--partition=<partition_name>`
- Sending emails: `--mail-type=<type>`
- Output file: `--output=<file name> (%j gives you job id)`
- Number of nodes: `--nodes=<nodes>`
- Number of tasks: `--ntasks=<processes>`
- Quality of service: `--qos=<qos>`
- Reservation: `--reservation=<name>`
- Wall time: `--time=<wall time>`
- Job Name: `--job-name=<jobname>` `...etc...`

- FYI: You do NOT actually type <> above – this designates something specific you as a user must enter about your job

Writing a Job: `hostname.sh`

- Submit a slurm job with the following instructions:

1. The job will be submitted from a bash script **named hostname_summit.sh**
2. The job will run **on 1 node**
3. Request **1 minute wall time**
4. Run on the **shas-testing partition**
5. Output should **contain job id**
6. The job should run **the Unix "`hostname`" command**

Writing a Job: Hostname

- Set up job boilerplate (nano or vim):

```
#!/bin/bash

## Directives
#SBATCH --<resource>=<amount>

## Software
module purge

## User Scripting
```

Writing a Job: Hostname

- Directives:
  - 1 node
  - 1 minute wall time
  - “shas-testing” partition
  - Output should contain job id

- Number of nodes:
  `--nodes=<nodes>`
- Wall time:
  `--time=<wall time>`
- Partition:
  `--partition=<partition_name>`
- Output file:
  `--output=<file name>`
  `(%j gives you job id)`

Writing a Job: Hostname

- Software:
  - Do we need any software? (modules)
- User script:
  - What command do we want to run?

Demo: Writing Hostname

Demo: Writing Hostname

```
#!/bin/bash
#SBATCH --nodes=1                          # Number of requested nodes
#SBATCH --time=0:01:00                     # Max wall time
#SBATCH --partition=shas-testing           # Specify Summit haswell
nodes
#SBATCH --output=hostname_%j.out           # Rename standard output file

# purge all existing modules
module purge

hostname
```

Submitting a Job

1. Load up the slurm module (default)

```
module load slurm/summit
```

2. Submit the job:

```
sbatch <script name>.sh
```

3. Check output

Review: Learning Goals

1. Why High Performance Computing (HPC)?
2. Understand Basic Resources
3. Get an account & log in
4. Navigate the RC system
5. Run a job
6. Help!

Help! I'm stuck, where do I go?

- **Documentation**: curc.readthedocs.io/
- **Trainings with Center for Research Data and Digital Scholarship (CRDDS)**: https://www.colorado.edu/crdds/
- **Helpdesk**: rc-help@colorado.edu

Questions

Survey and feedback

http://tinyurl.com/curc-survey18