



# Introduction to HPC on Blanca

# Introduction to HPC on Blanca

- Andrew Monaghan
- *Email:* [Andrew.Monaghan@Colorado.edu](mailto:Andrew.Monaghan@Colorado.edu)
- *RC Homepage:* <https://www.colorado.edu/rc>
  
- Slides available for download at:  
[https://github.com/ResearchComputing/APPM\\_HPC](https://github.com/ResearchComputing/APPM_HPC)

# Outline for this presentation

- Session 1 (today)
  - Overview of RC, Blanca
  - Logging in
  - Basic Linux commands
  - File editing
  - Linux filesystem
  - Environment variables
- Session 2: (next week)
  - Software modules on Blanca
  - Bash scripts and Job Scheduling

# Session 1

# What is Research Computing?

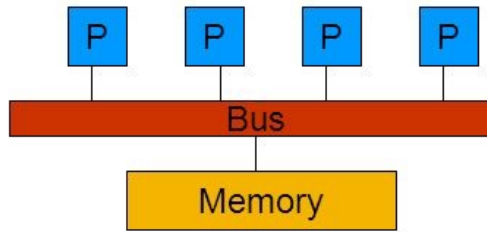
- Provide services for researchers that include:
  - Large scale computing
  - Data storage
  - High speed data transfer
  - Data management support
  - Consulting
  - Training
- We are likely best known for:
  - Summit and Alpine Supercomputers (~12,000 cores each)
  - **Blanca "condo" cluster** (~4,000 cores)
  - PetaLibrary storage



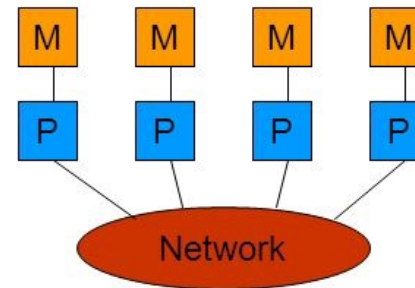
# What Would I Use Blanca For?

Solving large problems that require more:

- Memory than you have on your personal computer
- Cores/nodes/power than you have on your personal computer
- Blanca is set up for both shared memory (single node) and distributed memory (multi-node) parallelization.
  - Can also use Summit/Alpine for “big” distributed memory parallelization



- **Shared memory**



- **Distributed memory**

Source: [https://images.slideplayer.com/25/7599921/slides/slide\\_4.jpg](https://images.slideplayer.com/25/7599921/slides/slide_4.jpg)

# Blanca

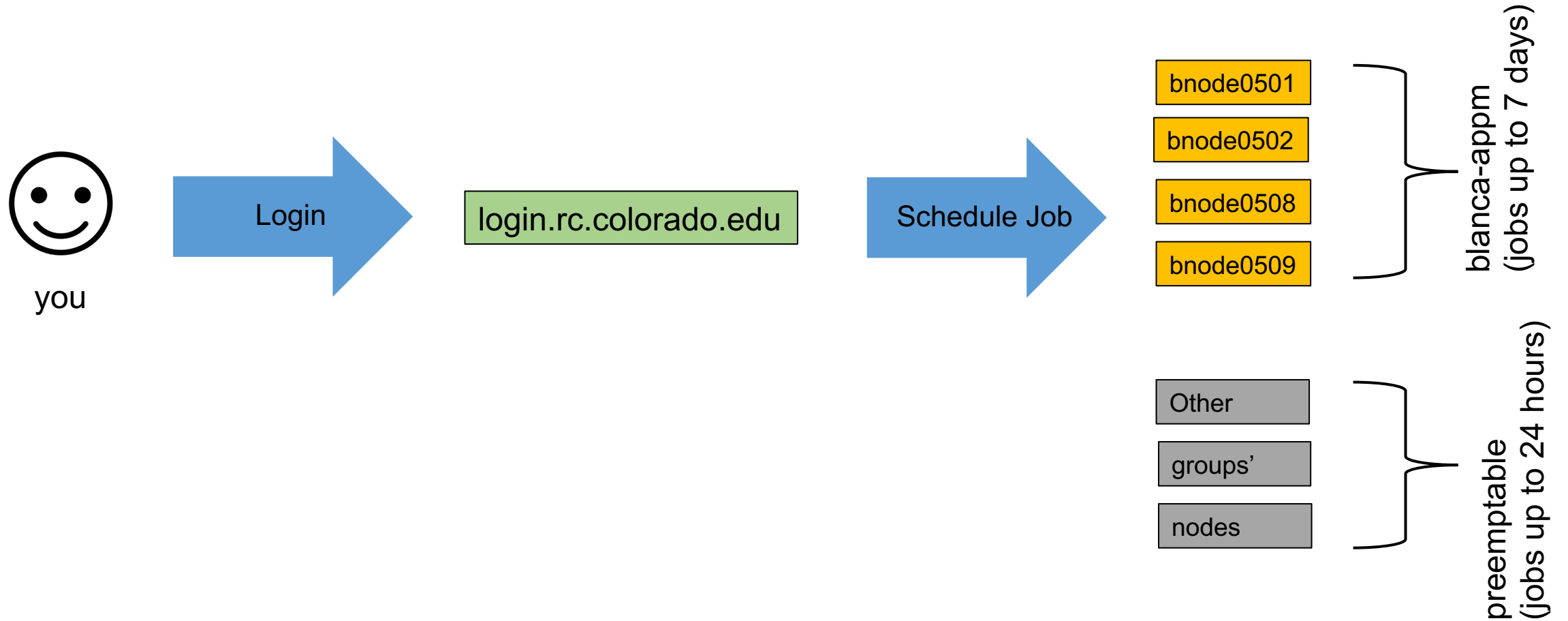
- A “condo” cluster whereby individual research groups own nodes
- List of nodes and groups can be found [here](#)
- Users have dedicated access to their group’s nodes (e.g., blanca-appm)
  - Jobs up to 7 days long.
  - Can also run ‘preemptable’ jobs on other groups nodes (jobs up to 24 hours long)
- More documentation on Blanca:  
<https://curc.readthedocs.io/en/latest/access/blanca.html>

# Blanca APPM nodes

- bnode0501, bnode0502, (2 nodes)
  - 32 (effectively 64) cores, avx2, Cascade, 2.3 GB RAM/core)
- bnode0508, bnode0509 (2 nodes)
  - 40 (effectively 80) cores, avx2, Cascade, 2.3 GB RAM/core)



# Blanca Workflow



# Logging In

- `ssh <identkey>@login.rc.colorado.edu`
- Enter your `identkey_password`
- Authenticate by accepting the Duo push to your smartphone
  - Can also authenticate by text message, phone call, or token
- More info here:  
<https://curc.readthedocs.io/en/latest/access/logging-in.html>

# Linux

- Part of the Unix-like family of operating systems.
- Started in early '90s
- Several distributions are available – from enterprise-grade, like RedHat Linux (RHEL), to more consumer-focused, like Ubuntu.
  - Blanca nodes presently run RHEL7
- Runs on everything from embedded systems to supercomputers.
- Linux is simple, flexible, fast, many potent tools

# Anatomy of a Linux command

- command [flags] [flag arguments] [target(s)]
  - `ls -l myworkdir/`
- Case is important!
- Help on commands is available through the "man" command (short for manual). E.g.,
  - `man ls`

# File and directory related commands

- **pwd** – prints full path to current directory
- **cd** – changes directory; can use full or relative path as target
- **mkdir** – creates a subdirectory in the current directory
- **rm** – removes a file (**rm -r** removes a directory and all of its contents)
- **cp** – copies a file
- **mv** – moves (or renames) a file or directory
- **ls** – lists the contents of a directory (**ls -l** gives detailed listing)

# File-viewing commands

- **more** – displays a file one screen at a time
- **cat** – prints entire file to the screen
- **head** – prints the first few lines of a file
- **tail** – prints the last few lines of a file
- **diff** – shows differences between two files
- **grep** – prints lines containing a string or other regular expression (ps -ef | grep XX)
- **sort** – sorts lines in a file
- **find** – searches for files that meet specified criteria
- **wc** – count words, lines, or characters in a file



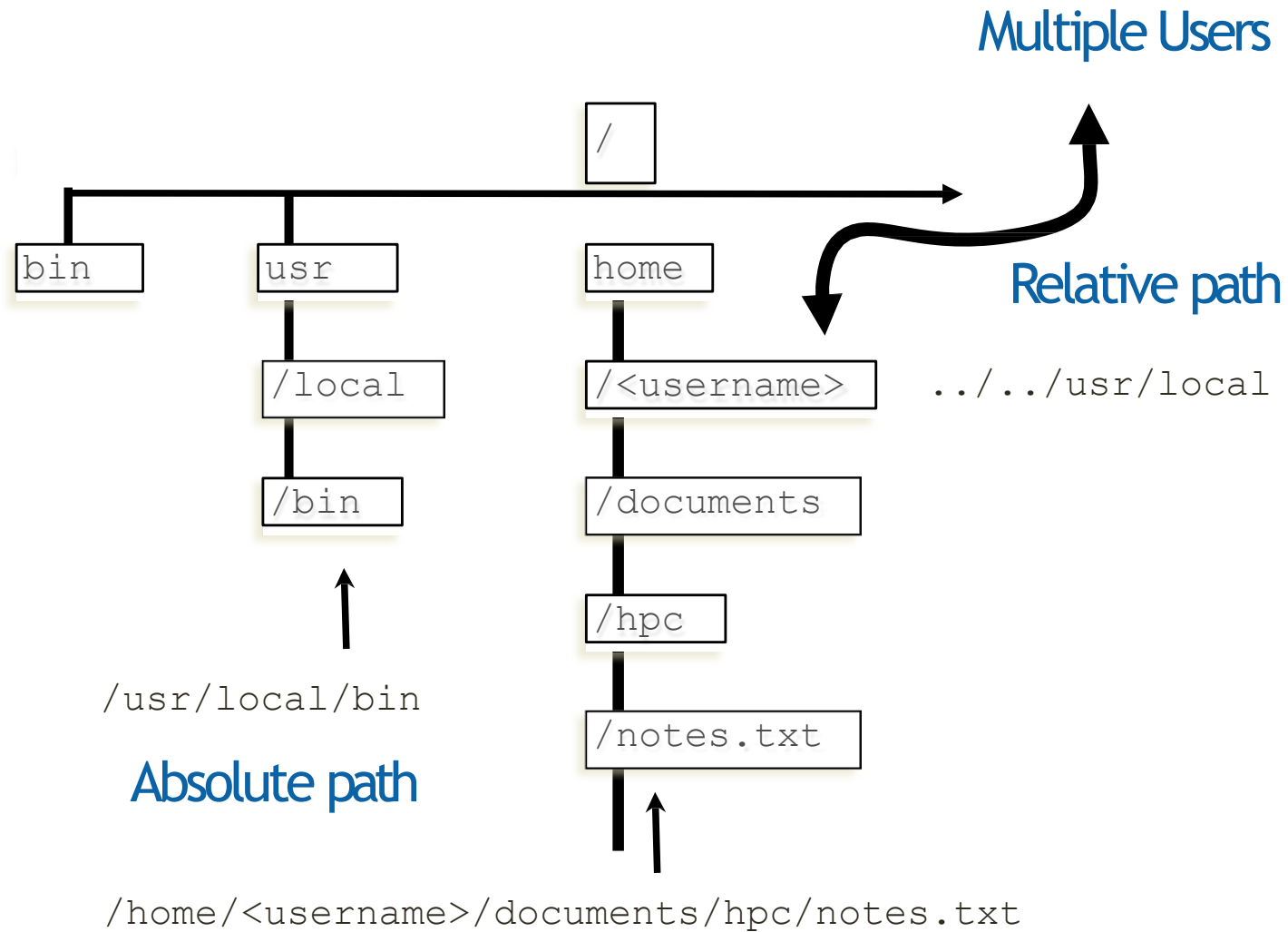
# File editing with **nano**

- To edit a file:
  - `nano myfile.txt`
- From within Nano:
  - `Ctrl+o` save (need to confirm filename)
  - `Ctrl+x` exit
  - `Ctrl+k` cut
  - `Ctrl+u` paste
- Other popular Linux editors: vi, emacs

# The Linux Filesystem

- System of arranging files on disk
- Consists of directories (folders) that can contain files or other directories
- Levels in full paths separated by *forward* slashes, e.g.
- `/home/user/scripts/analyze_data.sh`
- Case-sensitive; spaces in names discouraged
- Some shorthand:
  - . (the current directory)
  - .. (the directory one level above)
  - ~ (home directory)

# Filesystem



# Your personal directories on Blanca

- **/home/<username>**
  - Very small: 2GB.
  - Backed up daily.
  - Good for 'can't lose' files
- **/projects/<username>**
  - 250 GB
  - Backed up regularly
  - Good for storing scripts, self-installed software, some data
- **/rc\_scratch/<username>**
  - Group-shared 130 TB partition.
  - Good for jobs with lots of I/O
  - Not backed up
  - Temporary: data deleted 90 days from creation.

# Session 1

# Software

- Common software is available to everyone on the systems
- Research Computing uses modules to manage software
  - You load modules to prepare your environment for using software
    - Modules set any environment variables, paths, etc.
    - Set environment so application can find appropriate libraries, etc.
- You can also install your own software
  - It is best if you are responsible for support
  - We are happy to assist



# Using Modules

- Must be on a **compute** node to browse the modules (e.g, bnode0501)
- To set up your environment to use a software package, type **module load <package>/<version>**
- Some modules might require a specific hierarchy to load
  - For some modules, you may need to specify a specific version
    - For example, **module load R/3.3.0**
  - For other modules, you may be able to be more generic
    - For example, **module load matlab**
- Some modules may require you to first load other modules that they depend on
- To find dependencies for a module, type **module spider <package>**
- To find out what software is available, you can type **module avail**

# Using Anaconda for Python, R

- Many users prefer to use conda to manage python (and R, etc) packages and environments.
- We have Anaconda python installed on our system!
  - (but it isn't a module...)
- Documentation on use:  
<https://curc.readthedocs.io/en/latest/software/python.html>

# Next topic: Job scheduling

- **Job** – a system allotment of resources that run a particular application.
- **Slurm** -- resource manager
- Because Summit and Blanca are shared resources among a variety of groups on campus, users must run their applications through jobs.
  - Ensures everyone can utilize the system
  - No one person taking up too much of the system!

# Two types of jobs

- **Interactive Jobs**

- Interactive allotments or resources where user run their applications manually
- Useful for:
  - Debugging Applications
  - Running GUI Applications

- **Batch Jobs**

- Non-interactive allotment of resources that run applications in the background
- Think “baking a batch of cookies”
- Useful for:
  - Applications that take a substantial amount of time
  - Non-interactive Applications

# Scheduling an interactive job

- To work with R interactively, we request time from Summit
- When the resources become available the job starts
- Commands to run:

```
sinteractive --reservation=appm --time=00:10:00
```

\*note “--reservation=appm” is only for this workshop

- Once we receive a prompt, then:

```
module load R  
R
```

- Once we finish we must exit! (job will time out eventually)

# Scheduling Batch Jobs

- `sbatch` command: schedule a batch job with Slurm
- The `sbatch` command usually takes in 1 parameter: A job script.
  - Job scripts provide all information on what is needed for the job.
  - Parameters can be overwritten or added externally by specifying the parameter as a flag.
- Example:

```
sbatch test.sh
```

or

```
sbatch test.sh --time=02:00:00
```



# Anatomy of a job script

```
#!/bin/bash
```

Job parameters

```
#SBATCH --ntasks=1           # Number of requested tasks
#SBATCH --time=0:01:00       # Max wall time
#SBATCH --partition=blanca-appm # Specify APPM nodes
#SBATCH --qos=blanca-appm-student # Specify you are student
#SBATCH --output=test_%j.out  # Rename standard output file
```

Your job commands

```
# Now run job (commands below here)

# Purge and load needed modules
module purge
module load R

# Run commands
Rscript myscript.R
```

# Job Parameters

- Allocation: `--account=<account-name>`
- Partition: `--partition=<partition-name>`
- Number of nodes: `--nodes=<nodes>`
- Number of Tasks: `--ntasks=<number-of-tasks>`
- Quality of service: `--qos=<qos>`
- Reservation: `--reservation=<name>`
- Wall time: `--time=<wall-time>`
- Job Name: `--job-name=<jobname>`
- Output File: `--output=<outputname>`

*More on slurm commands:*  
<https://slurm.schedmd.com/quickstart.html>

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

# APPM QoS, Account, and Partition

- When using the APPM Blanca nodes, job scripts should specify the following information:

```
--account=blanca-appm  
--partition=blanca-appm  
--qos=blanca-appm-student #for some in APPM qos may just be blanca-appm
```

- Users of Blanca also have access to a low-priority preemptible QoS that will attempt to run your job on any available node on Blanca
  - Preemptable jobs will be booted off the node and restarted at a later time if a high priority user runs a job on the node.

```
--qos=preemptable
```

# schedule\_hostname.sh

```
#!/bin/bash
#SBATCH --nodes=1                # Number of requested nodes
#SBATCH --ntasks=1              # Number of requested cores
#SBATCH --time=00:01:00         # Max wall time
#SBATCH --qos=blanca-appm-student # Specify QOS; may be just "blanca-appm" for some
#SBATCH --partition=blanca-appm  # Specify APPM nodes
#SBATCH --account=blanca-appm    # Specify account
#SBATCH --output=hostname_%j.out # Rename standard output file
#SBATCH --job-name=hostname      # Job name

# purge all existing modules
module purge

hostname
```

# Running the job script

Schedule the job:

```
$ sbatch --reservation=appm schedule_hostname.sh  
(note “--reservation=appm” for this class only)
```

Check the status of the job:

```
$ squeue / $ squeue -u <user> /  
$ squeue -q <qos>  
...or  
$ sacct / $ sacct --format=<options>  
...or  
$ scontrol show job <job number>
```

Look at the job output:

```
$ cat hostname_<job-id>.out
```

*(\*note that **<job-id>** is your job number)*

*More on slurm commands: <https://slurm.schedmd.com/quickstart.html>*

# Topics we didn't cover today

- Job arrays (when you need to run lots of similar tasks)
- Running preemptable jobs
- CURC JupyterHub (for running Jupyter Notebooks on Blanca)
- Use of conda on CURC
- ..and lots of other cool stuff
- See: <https://curc.readthedocs.io> ; or email [rc-help@colorado.edu](mailto:rc-help@colorado.edu) to schedule a consultation



# Thanks!

- Please fill out the survey: <http://tinyurl.com/curc-survey18>
- Contact: [rc-help@Colorado.edu](mailto:rc-help@Colorado.edu), [Andrew.Monaghan@colorado.edu](mailto:Andrew.Monaghan@colorado.edu)
- Course materials for today:
  - [https://github.com/ResearchComputing/APPM\\_HPC](https://github.com/ResearchComputing/APPM_HPC)
- Blanca (and other) documentation:  
<https://curc.readthedocs.io/en/latest/access/blanca.html>
- Slurm Commands: <https://slurm.schedmd.com/quickstart.html>

# Supplemental Slides

# Environment variables

- Environment variables store important information needed by Linux users, programs, etc.
- Type '`env`' to see your currently set environment variables
- Useful Environment variables:
  - `PATH`: directories to search for commands
  - `HOME`: home directory
  - `PWD`: current working directory
  - `USER`: username
  - `LD_LIBRARY_PATH`: directories to search for shared objects (dynamically-loaded libs)

# Shell Wildcards and Special Characters

- \* - matches zero or more characters
- ? - matches a single character
- # - comment; rest of the line is ignored
- \ - escape; don't interpret the next character

# Modes (aka permissions)

- Three classes of users:
  - User (u) aka “owner”
  - Group (g)
  - Other (o)
- Three types of permissions
  - Read (r)
  - Write (w)
  - Execute (x)

.. own grp oth  
-|---|---|---

drwxr-xr--

# Modes (continued)

- `chmod` changes modes:
- To add write and execute permission for your group:  
`chmod g+wx filename`
- To remove execute permission for others:  
`chmod o-x filename`
- To set only read and execute for your group and others:  
`chmod go=rx filename`