Computation via Hummingbird

Why and how to use the UCSC Hummingbird Computational Cluster

April 8, 2019 — s/lab

Jake Vincent

What is Hummingbird?

What is Hummingbird?

- **Open access high-performance computational cluster** located on campus at UCSC

≈ lots of processors (cores) that can work together* to perform computations rapidly and efficiently

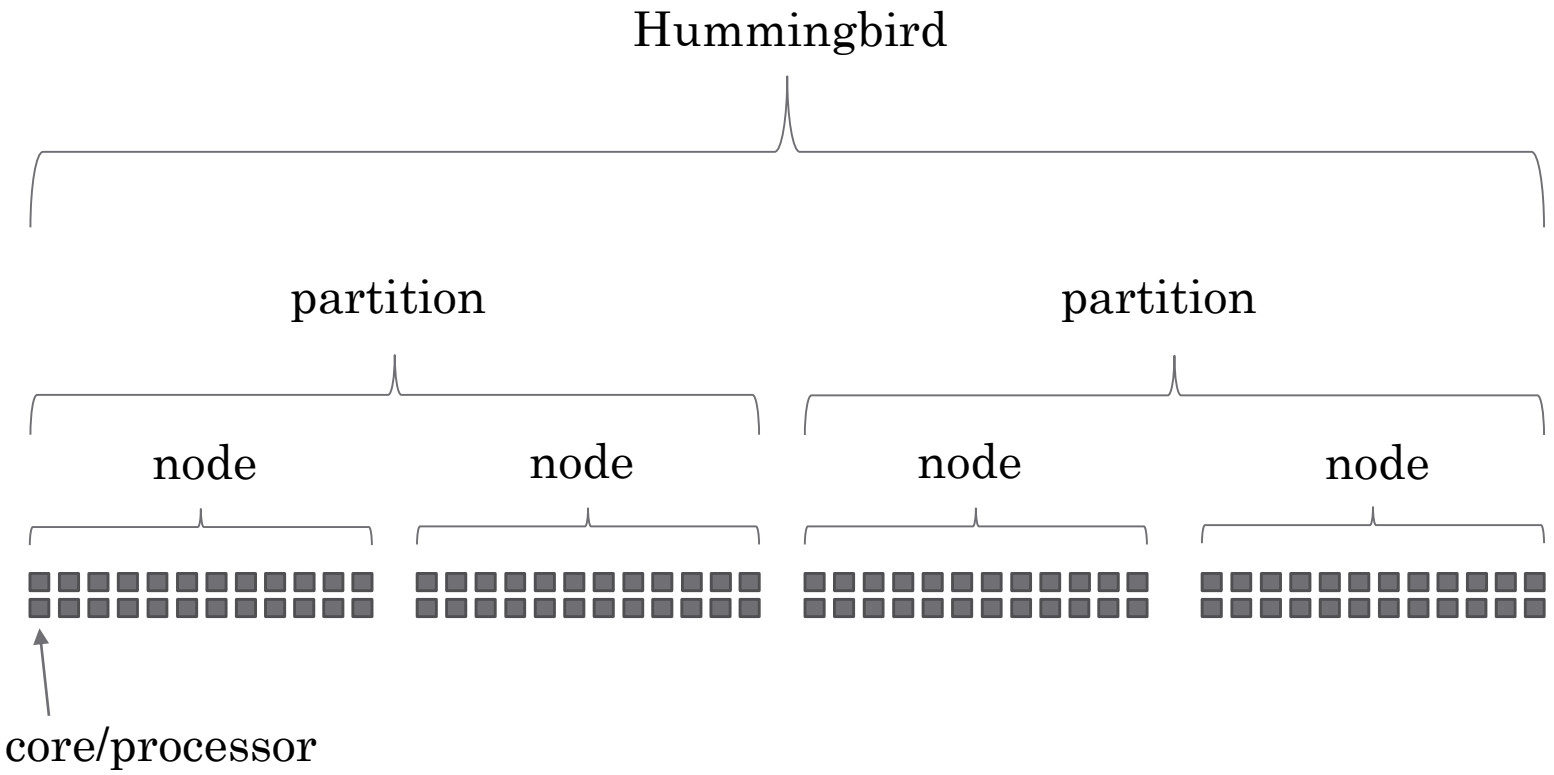
- Cores are grouped into nodes; nodes are grouped into partitions
- Each node has its own memory resources, storage, network connection, etc. (like your laptop)
- Most nodes have 24 cores; some have more
- Nodes run on a version of Linux

*depending on the software you're using

What is Hummingbird?

a single node

Organization of resources



What is Hummingbird?

- Accessible from your own computer, even if you are off campus (via campus VPN)
- Each user has their own home directory (1TB quota) and scratch directory (no quota)

Why to use Hummingbird

Why to use Hummingbird

- More computational power than your laptop or lab computer
- **Computations can't be interrupted** by other lab users, updating computers, accidental shutdowns, keyboard-loving cats, etc.
- Makes you 2.7x cooler

How to use Hummingbird

How to use Hummingbird

- **Before proceeding:**

1. Go to cruzid.ucsc.edu and log in using your Gold password
2. Click “Advanced” tab
3. Click “Change Shell or SSH Key”
4. Under “Select Shell”, select “bash” and click “Change”.

other option is C-shell, which you could use in principle—I just don’t know how

How to use Hummingbird

- **Bird's-eye view (heh heh)**

1. Log in to Hummingbird
2. Make sure the software you want to use is installed on Hummingbird (including packages)
3. Upload files needed to run your computation (e.g. an R script) to your home directory
4. Tell Hummingbird how to run your script and with what resources (how many cores to use, etc.).
5. Run it.
6. View/collect the output of your computation.

Logging in to Hummingbird

Logging in to Hummingbird

- Done using an SSH (secure shell) client, which allows data to be sent to and from remote computers securely
  - SSH client: "a software program which uses the secure shell protocol to connect to a remote computer."
- macOS, Linux-based OSs → included and accessible through the command line
- Windows 10 → available through command prompt in version 10.0.16299 and up—type "System Information" in search bar to find your version
  - also available through the Windows Subsystem for Linux (WSL)
- Older versions of Windows → download and install an SSH client
  - Bitvise SSH client
  - PuTTY SSH and telnet client

Logging in to Hummingbird

- Open your SSH client
  - macOS, Linux-based OS, Windows 10 → open command line
  - Older versions of Windows → open Bitvise/PuTTY/whatever SSH client
- Enter the following in your SSH client:

```
> ssh [ucsc_username]@hb.ucsc.edu
```

- Enter your UCSC Blue password when prompted
- You should see a welcome screen. If you do, you have logged in to the head node!

Checking for available software

Checking for available software

- Software is available as “modules” that can be loaded into your environment
- Some module-related commands:

```
$ module list ← lists loaded modules

$ module spider ← lists all available software applications

$ module spider <string*> ← lists all available software starting
w/ a particular string (e.g. module spider R*)
```

Logging out

- While logged in to the cluster, logout by simply typing in “logout”.

```
$ logout
```

Feeding Hummingbird

Feeding Hummingbird

- To run a script, you need to direct Hummingbird to a script in your home directory on the cluster
- Assuming your script uses external data, you will also have to upload your data in some format.

Two methods for uploading files

1. Command line SFTP utility (available on all OSs)
2. FTP client (e.g. FileZilla)

Feeding Hummingbird

- Method 1: Log in to your directory using SFTP in command line

```
> sftp <ucsc_username>@hbfeeder.ucsc.edu
```

- Answer ‘yes’ to prompt
- Enter your UCSC Blue password. Once you’ve logged in successfully, you’ll see the following, and you’re ready to move files around.

```
sftp>
```

None of this should be done while logged in on the cluster. If you are logged in and want to move files to or from your directory on the cluster, logout and then use the sftp utility as shown above

Feeding Hummingbird

- While logged in via SFTP client, navigate to the local directory where the file to transfer is located and to the remote directory where you want to transfer the file.
- Directory navigation commands have both local and remote variants. The local variants are prepended with an 'l'.
  - To see what directory you're currently in: **`pwd/lpwd`**
  - To change directories: **`cd/lcd <path>`**
- Once you've navigated to the appropriate local and remote directories:
  - Upload a file with **`put <filename>`**
  - Download a file with **`get <filename>`**

Feeding Hummingbird

- Say I want to {up/down}load a file called “script.R”:

```
sftp> put script.R ← moves script.R from local to remote directory
sftp> get script.R ← moves script.R from remote to local directory
```

- To leave SFTP client, type “exit” and hit enter

```
sftp> exit
```

Feeding Hummingbird

- Method 2: Log in to your directory using an FTP client like FileZilla.

1. Open FileZilla
    1. Host: hbfeeder.ucsc.edu
    2. Username: your UCSC username
    3. Password: your Blue password
    4. Port: 22
    5. Click “Quickconnect”
2. Accept host key
3. {Up/down}load files by dragging and dropping

DON'T FORGET TO SAVE

- If there is anything you want to hold onto from the software session that the scheduler runs for you other than the output, make sure to save it!
- If you're running an R session and want to save something from your R environment that you can then download to your computer, make sure to use the relevant commands in your R script to save the data.
  - saveRDS() will save anything in your environment as a data script that you can then download and load into your environment on your personal computer using readRDS().
    - For an object called "clmm", for example, I would save it using:

      saveRDS(clmm, file = "clmm.RDS")

Telling Hummingbird how to compute

Telling Hummingbird how to compute

- Since Hummingbird has many resources that multiple people may want to use at a given time, **there needs to be a way to distribute these resources fairly and according to each user's needs**.
- Solution: no user interacts with the compute nodes directly.

Telling Hummingbird how to compute

- Users interact with a **scheduler** that keeps users' jobs in a queue and assigns computational resources to your job when available.
  - SLURM scheduler (Simple Linux Utility for Resource Management)
- Interact with the scheduler by submitting your job as a SLURM script
  - How many nodes you want to use
  - What partition to use
  - How to update you on job status (e.g. when job fails or finishes)
  - How many cores to use
  - Where to put/what to call output log and error log
  - Modules to load
  - Commands for running your script through a Linux command line

Sample SLURM script

```
#!/bin/bash
#SBATCH --time=1:00:00 #(leave line out if you don’t want a limit)
#SBATCH --nodes=1 #(number of nodes to use)
#SBATCH --partition=128x24 #(128GBx24cores)
#SBATCH --job-name=my_job
#SBATCH --mail-type=ALL #(mail events(NONE,BEGIN,END,FAIL,ALL))
#SBATCH --mail-user=jwvincen@ucsc.edu
#SBATCH --ntasks=1 #(number of cores, this case 1)
#SBATCH --output=%j.out #(standard output log)
#SBATCH --error=%j.err #(standard error log)

# select the software you need
module load R/R-3.5.0

# command that the scheduler sends to the command line
R --no-save < script.R
```

To submit your job

- Once you've put both your SLURM script and the script you want to run into your hummingbird home directory (or composed them directly in your home directory using vim), submit your job using commands specific to the SLURM scheduler:

```
$ sbatch job.SLURM ← submits your job to the scheduler, which will run it when the requested resources are available
```

To monitor your job

- Your job can be monitored from the same place you submitted it.

```
$ squeue ← returns the status of all currently running jobs
$ squeue –u jwvincen ← returns the status of all currently
running jobs associated with my username
```

To cancel your job

```
$ scancel [job id] ← cancels the specified job
```

Viewing, downloading computational output

Viewing output/error logs

- If you're logged in to the head node, output and error logs can be viewed using the *more* command. If you prefer, you can save these logs to your computer and open them in a text editor. If my output file is called "job.out," I can view it by entering the following:

```
$ more job.out
```

- It would be useful to familiarize yourself with linux commands for navigating folders, etc. here. Some basic ones:
  - "ls" lists the contents of the directory you're in (useful if you don't remember what you told the schedule to name your output/error files, for example).
  - "cd" changes the directory. If I'm in /hb/home/user, for example, and want to move to a folder in jwvincen called "foo", I'd type in "cd foo". To go back/up one level, enter "cd .." and to go back to the home directory, enter "cd ~".

Downloading files

- Downloading files can be done the same way you uploaded them—using an sftp utility via the command line or a program such as FileZilla.
- If you'll use the sftp utility, log in to hbfeeder.ucsc.edu using the sftp utility (as shown earlier) and use the "get" command to download files to a local directory. Entering the following will download the data.rds file in the remote directory to your local directory.

```
sftp> get data.rds
```

- If using FileZilla, log in to hbfeeder.ucsc.edu as shown earlier and drag and drop the remote file to a local directory.