Introduction to High Performance Computing (HPC)

QLS 612 - 2022
Brent McPherson, PhD

McGill

HBHL NEUROHUB

The goals of this lecture are to:

- Describe the structure of HPCs and how they differ from other traditional computers and common approaches.
    - Laptops / Desktops
- Introduce the concepts of using High Performance Computers (HPCs).
- How to begin effectively using HPCs in your work.

This will reference concepts covered in the “Introduction to the Terminal and Bash” and “Containerization” lectures.

Modern NeuroData Science Requires Computers

- No part of our research can progress without access to useful computing resources.
    - Data is acquired and stored on computers.
    - Data preparation and curation is performed on computers.
    - Analysis and modeling require computers.
    - Visualization and reporting or results require computers.
    - The distribution and reporting of results require computers.
- If at any point our work cannot continue in a digital space, it has effectively ended.
- In order to maximize our ability to perform innovative work in a field dependent on modern computational advances, a useful understanding of these machines is necessary.

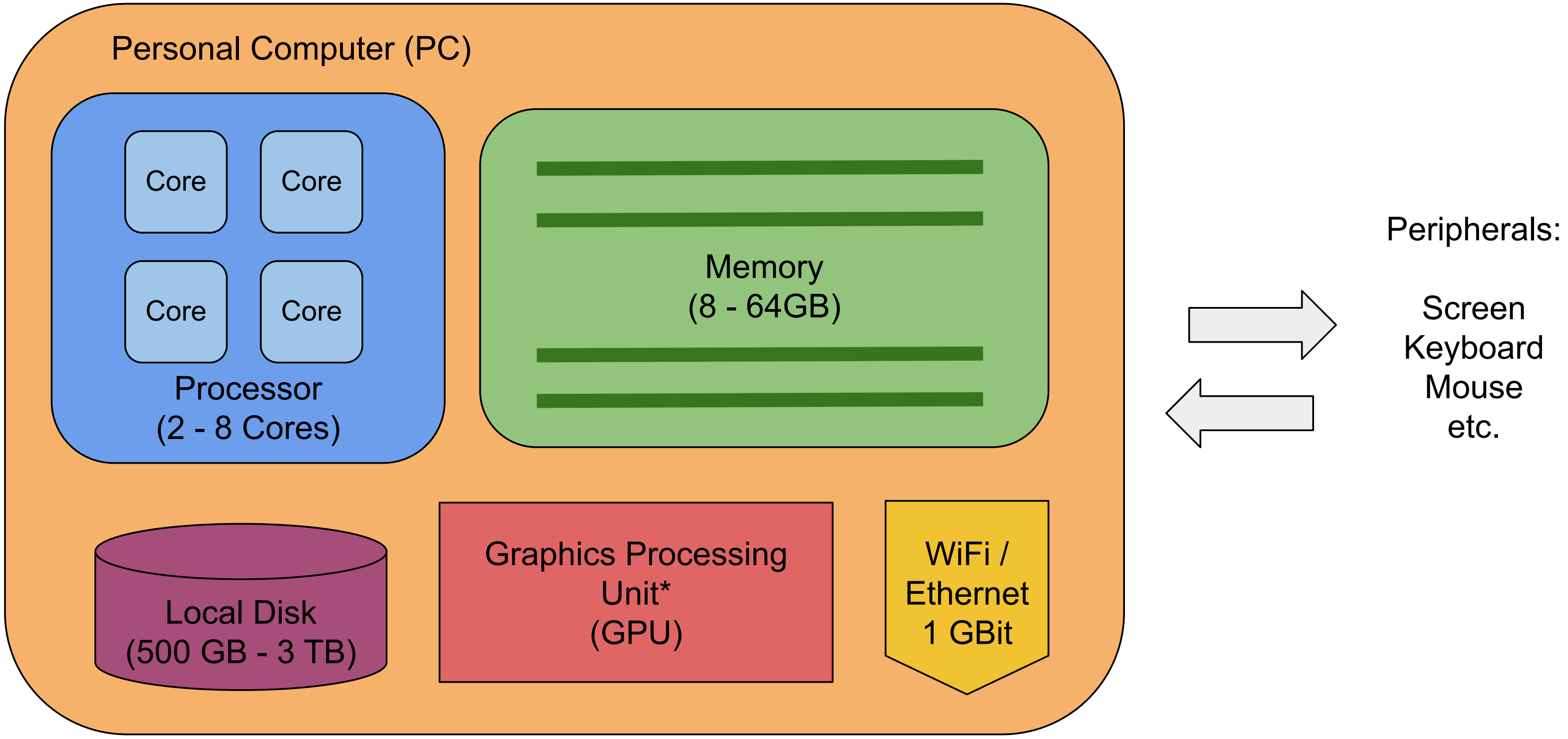
The Concepts of High Performance Computing (HPC)

Working on a personal computer (PC)

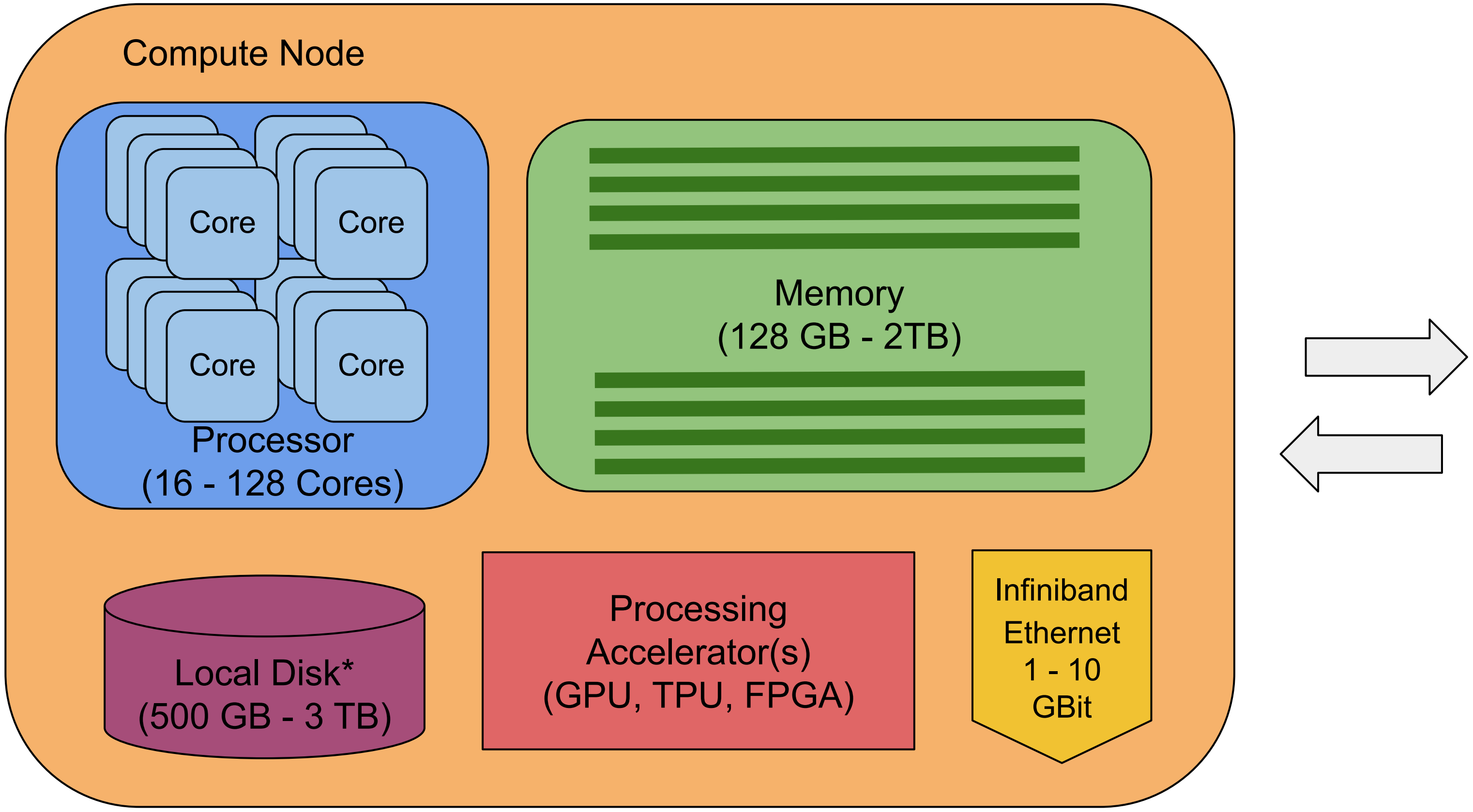
- Everyone is familiar with this.
    - You’re using one now.
- These machines are designed to facilitate the most common needs of the most people.
- However, the limits of these machines can be easily reached when performing modern neuroscience analysis.

Working on an HPC

- These are not “personal” systems, they are multi-user.
- You do not interact with them directly, but over a network connection.
- The interaction with your analysis entails:
    - Transferring data / results.
    - Managing your analysis “job”.
- There is minimal “interactive” computing performed on a HPC.
    - Typically little to no visualization is performed on HPC systems.
- They almost exclusively run Linux.



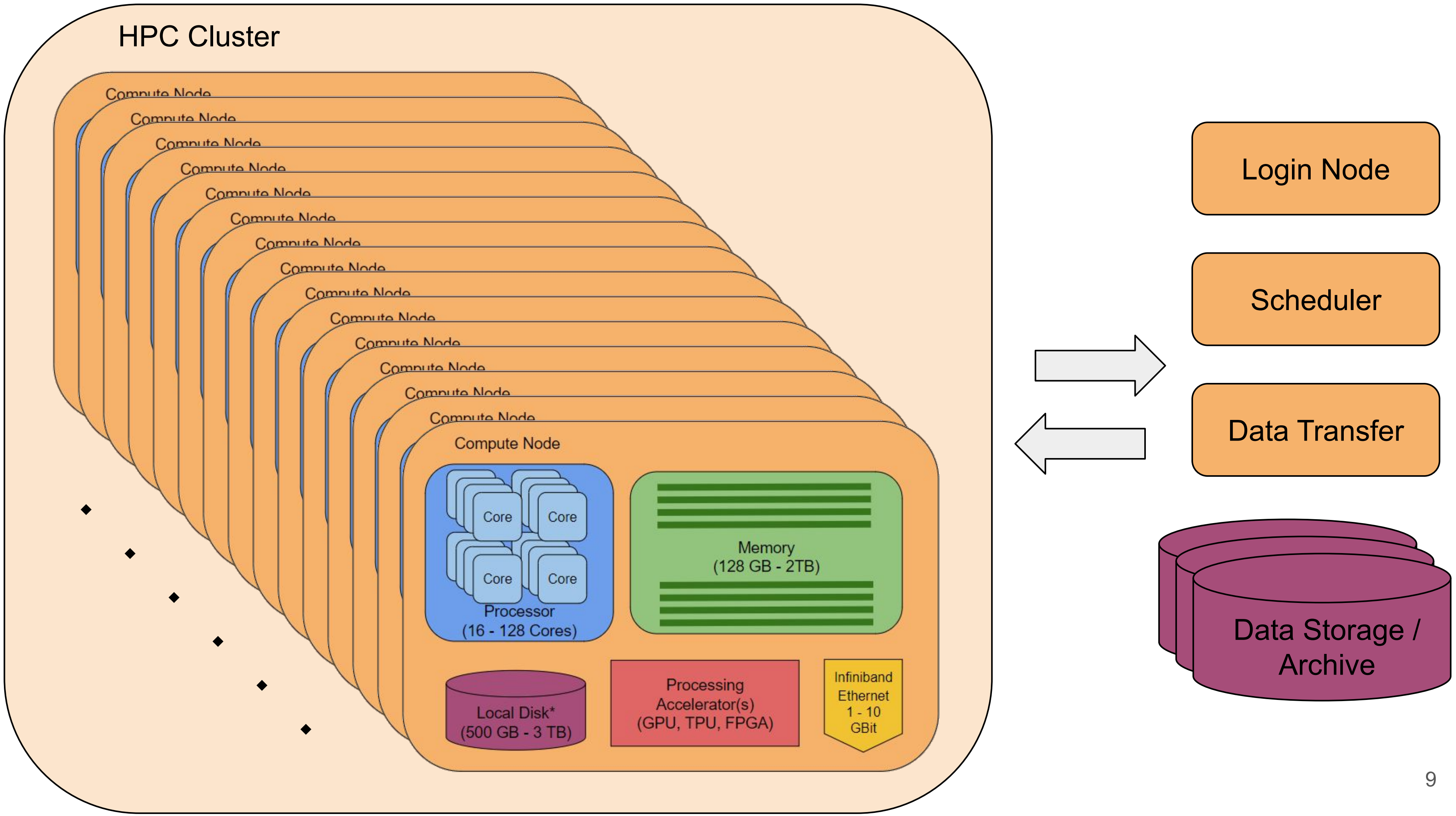
* not necessarily present in all systems (laptops)



Peripherals:
None

Connects to
the rest of the
compute
cluster.

* it may be less and / or have higher access speed


HPC Cluster
Compute Node
Core
Processor
(16 - 128 Cores)
Memory
(128 GB - 2TB)
Local Disk*
(500 GB - 3 TB)
Processing
Accelerator(s)
(GPU, TPU, FPGA)
Infiniband
Ethernet
1 - 10
GBit
Login Node
Scheduler
Data Transfer
Data Storage /
Archive

The Anatomy of a Computer

PCs

- 1 CPU
  - 2 - 8 cores
  - 3.5 - 4.5 GHz
- 8 - 64 GB RAM
- 500 GB - 3 TB Storage
- 1 GPU
  - This may be built into the CPU
- WiFi / 1 GBit Ethernet
- Single-User System
- Good for most general usage
  - This is likely where you will develop your analysis.

HPCs

- Multiple CPUs
  - 16 - 128 cores per CPU
  - 2.4 - 3.5 GHz
- 128 GB - 2 TB RAM
- 500 GB - 3 TB Storage
  - Not for long term storage
- 0 - 4 Processor Accelerators
  - GPUs w/ double precision CUDA
  - TPUs, FPGAs, Coprocessors, etc.
- 1 - 10 GBit Ethernet
  - Infiniband
- Multi-User System
- Good for high throughput / large models.

Working on the Cloud - An expensive middle ground

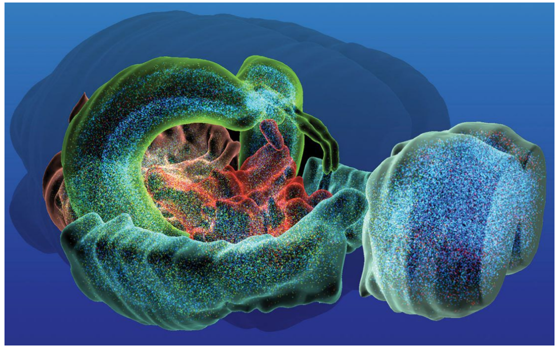
- This is most often an option through dedicated research proposals.
- The cloud system can be tailored to fit your specific analysis and data requirements.
  - A large virtual analysis computer
  - Your own dedicated cluster
- The data may be distributed on the cloud already.
  - Many open datasets already are.
- This will be *very* expensive for long term use.

The advantages of a HPC over a PC

- The ability to handle larger datasets in a more time efficient manner.
- The ability to “**Scale Up**” or “**Scale Out**” an analysis.
  - **Scale Up**: create a larger single instance of computing resources to run a larger model, estimation, or analysis.
  - **Scale Out**: the ability to analyze independent parts of a dataset simultaneously
    - Independent permutations / cross-validations / simulations.
    - Different subjects through the same preprocessing preparations.
      - “Embarrassingly Parallel”
- We’ll be introducing you to “**Scaling Out**” your analysis in the example today.

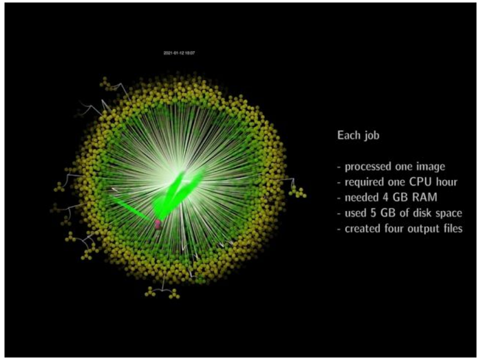
Scaling Up

Blue Brain Project



Scaling Out

biobank uk



Each job

- processed one image
- required one CPU hour
- needed 4 GB RAM
- used 5 GB of disk space
- created four output files

[UKBB high-throughput replication of over 40K subjects]

Beginning to use a Cluster Computer

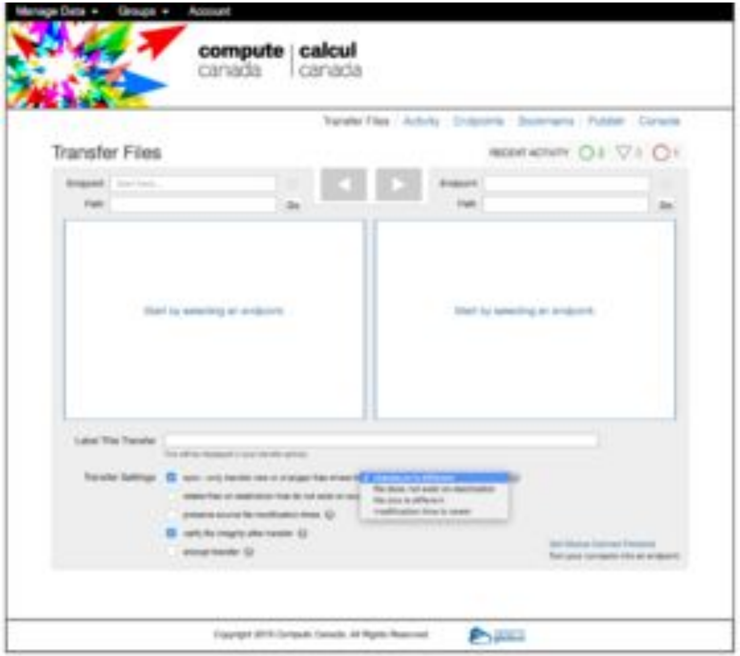
Accessing the Cluster

- This is done exclusively through a remote connection.
  - ssh, a remote desktop client, etc.
- You will connect to the **Login Node** of the cluster.
  - The login in node manages the coordination of user processes on the cluster
  - It also reports the overall status of the system.
- You will submit your jobs and query their status from the Login Node.
  - DO NOT run jobs or do work on the Login Node.

1. Connect to the cluster using SSH

Getting your data on the cluster

- I assume your data is on a computer not accessible by the cluster.
- You will use a file transfer tool to move your data from where it is to the data server.
    - scp, sftp, rsync, FileZilla, mobaXterm, etc.
    - This data server will be accessible to the compute nodes
- This may be done through the Login Node, but there may be a specific connection for moving data.
    - Check with the system administrators.



Getting your data on the cluster - Things to Remember

- Be aware of the policies of this server!
  - Files are often automatically removed after a (potentially short) window of inactivity.
  - These servers are often not backed up.
  - Your home directory (~/) is *probably* backed up. However, it may not be a good location or have enough space to store data that will run on the cluster.
- Once your analysis is complete you need to have a plan to securely store your data long term.

2. Copy over the example analysis scripts

Getting your software on the cluster

- The cluster administrator(s) will likely need to install it.
  - If you are confident you know how the tools works, you may be able to set it up yourself.
- Many tools will be readily available on the cluster through **modules**.
  - These are an efficient way to add and remove tools installed on the cluster to your session.
  - They also allow for easier management of different versions of the same tool.
  - Hopefully, most of the tools you need will already be installed.
- Running your code in **containers** guarantees you have the tools you need to run your analysis.
  - You do have to a have (or be able to build) a working container of the tool.

Getting your software on the cluster - Modules

- How most software is made available to users on a HPC.
- A convenient way to set up basic development environments and change between versions.

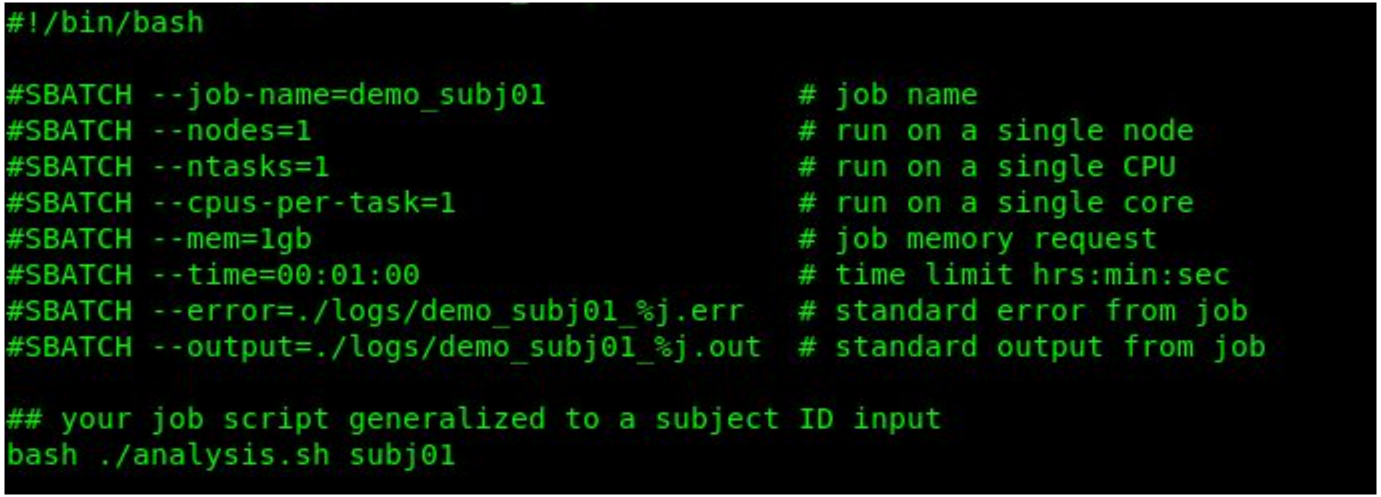
```
> module avail
> module load python
> module load python/3.8.0
> module unload python
> module swap python/3.7.0
```

Running your analysis - Working with a Scheduler

- With your data available you need to run your analysis.
- You will do this by running your analysis as a **Job**.
- **Jobs** are submitted to the Scheduler to be performed.
  - This deploys the analysis onto the compute nodes.
- Your entire analysis will need to be scripted.
  - You cannot interact with or manually input information to a job when it's running.
  - Much of what's been covered in QLS 612 will prepare you for working this way.
- Once your analysis script is complete, you create your job script for the analysis.
  - This script describes the resources needed to successfully run your analysis on the cluster.

Running your analysis - Creating Job Scripts

- Your jobs will be defined with a job script to:
    - Define the system resources to run your job.
    - Launch and run your analysis on the data.
- The job script will be submitted to the scheduler from the login node to be added to the queue.
    - The scheduler uses your resource requests and your priority to determine what runs when.
- Your outputs, along with logs, will be created for each job you submit.
- The example today will assume you are **Scaling Out** your analysis.
    - You want to run a common process across many subjects
    - i.e. fMRIPrep, FreeSurfer, etc.



```
#!/bin/bash

#SBATCH --job-name=demo_subj01                      # job name
#SBATCH --nodes=1                                   # run on a single node
#SBATCH --ntasks=1                                  # run on a single CPU
#SBATCH --cpus-per-task=1                           # run on a single core
#SBATCH --mem=1gb                                   # job memory request
#SBATCH --time=00:01:00                             # time limit hrs:min:sec
#SBATCH --error=./logs/demo_subj01_%j.err           # standard error from job
#SBATCH --output=./logs/demo_subj01_%j.out          # standard output from job

## your job script generalized to a subject ID input
bash ./analysis.sh subj01
```

Running your analysis - be efficient

- For every job you run, you will have:
    - The input data
    - The analysis script
    - The job script and logs
    - The results
- Just like your analysis, you should script as much of your cluster jobs as possible.
    - You should never be individually modifying job files for subjects.

The Example

- The example today reports back a short log about the system status of the node before waiting.
    - This will let you see the different nodes your jobs ran on.
- The analysis input wants a subject ID as an input
    - This is is likely how you would set up a symmetric analysis across your subjects.
- The goal is to provide a basic working template for your job scripts that you can modify.

3. Make and submit your jobs
4. Monitor their status
5. Cancel a job
6. Explore the documentation and modify your resource requests.
7. Run the script in a container?

Commands

```
> ./mk_jobs

> find jobs -type f -name "*.slurm" | xargs -n 1 sbatch

> squeue -u USERNAME

> sacct

> seff JOBID

> scancel JOBID
```

Thanks