Intro to O2

HMS Research Computing

Welcome to O2!

- Slides available at github.com/hmsrc/user-training
  - **IntroToO2.pdf**
- Contact us at rchelp@hms.harvard.edu
- Comments/feedback welcome at course survey in the Harvard Training Portal

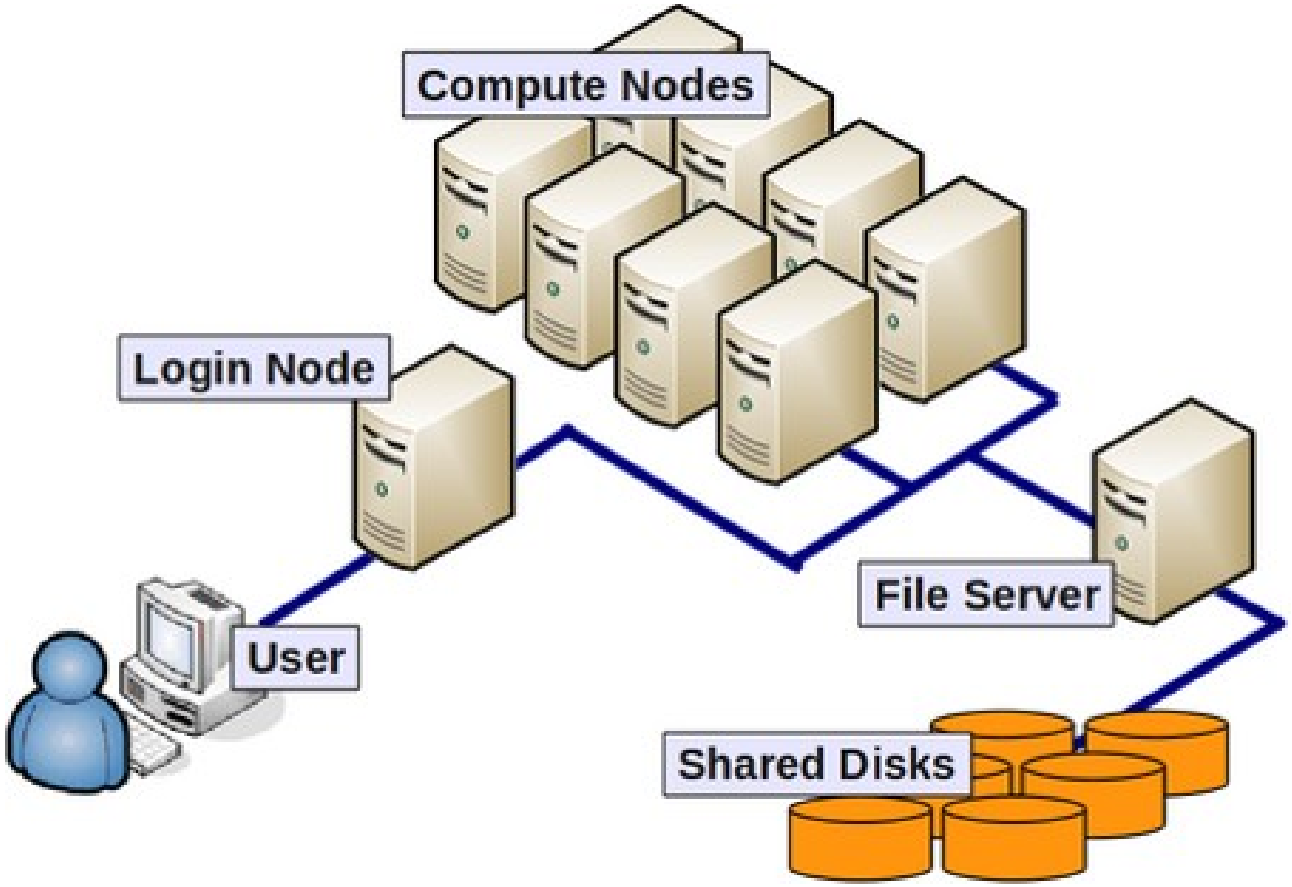
Welcome to O2!

- HMS Research Computing's newest High-Performance Compute cluster to enhance the compute capacity available to HMS Researchers
- Heterogeneous environment of newer, faster cores with high memory allocation to facilitate multi-core and parallelized workflows
- SLURM scheduler to efficiently dispatch jobs

O2 Tech Specs

- 11000+ cores
- 32, 28 or 20 cores per node
- 256-160 GB RAM per node (8-9GB/core)
- 9x756 GB and 1x1TB highmem nodes*
- 32 GPUs (8 M40 / 16 K80 / 8 V100*)
- Login/load balancer 6 VM (8 cores/16GB memory)
- InfiniBand connectivity between nodes available
- CentOS 7

Generic Cluster Architecture



Storage on O2

Important Note about O2 Storage

- O2 can only be used to store data of Harvard Security Level 3 and below.

- None of the standard filesystems are automatically encrypted, and **cannot** be used for HIPAA-protected or other secure data (Harvard's data security above level 3) unless those data have been de-identified.

HMS Storage Offerings

- Renamed HMS storage spaces to:
  - Better align with how researchers think about and use data
  - Better manage data within the research data lifecycle

- Storage offerings:
  - *Active*
  - *Standby*
  - *Cold*

HMS Storage Offerings

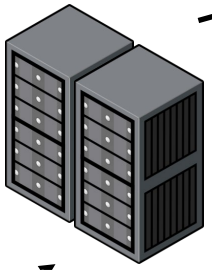
- ***Active*** – available now
  - Formerly Tier 1 (O2 group folders & research.files)
  - Research data that is frequently accessed, modified, or computed against.
- ***Standby*** – coming soon after beta testing
  - New and improved alternative to Tier 2
  - Infrequently accessed data, that is directly available for reference, retrieval, or analysis.
- ***Cold*** – future offering
  - Formerly Tier 3
  - Rarely accessed data requiring long-term retention, for regulatory or historical purposes

O2 Primary Storage

EMC² ISILON

O2 Cluster

- 11000+ cores
- SLURM batch system



Your computer

/home

- /home/user_id
- quota: 100GB per user
- Backup: extra copy and snapshots, daily to 14 days, weekly up to 60 days

/n/data1, /n/data2, /n/groups

- /n/data1/institution/dept/lab/your_dir
- quota: expandable
- Backup: extra copy and snapshots, daily to 14 days, weekly up to 60 days

Temporary “Scratch” storage

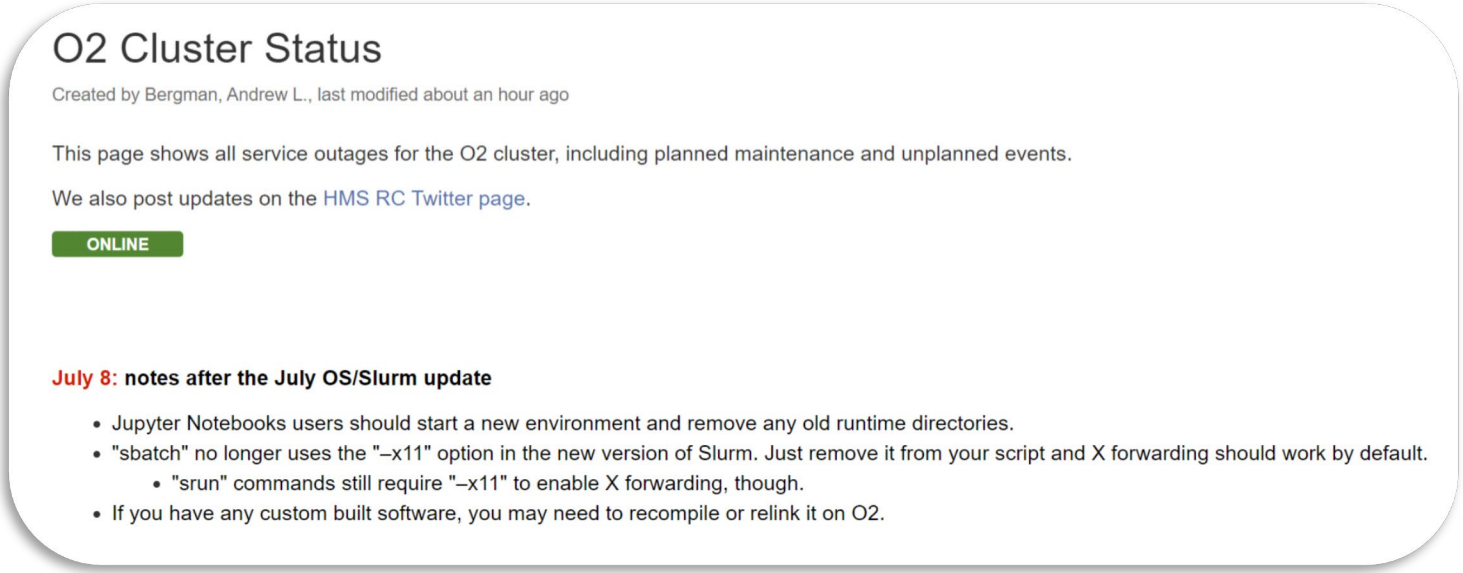
- For data only needed temporarily during analyses
- Each user can use up to 10 TB and 1 million files/directories
- Files not accessed for 30 days are automatically purged
- **No backups!**
- Location: /n/scratch3/users/<first_eCommons_char>/<eCommons>
- Example
  - mfk8@login01:~$ cd /n/scratch3/users/m/mfk8

- For more information: Scratch3 documentation

O2 Cluster Status

- Wiki page:

  https://wiki.rc.hms.harvard.edu/display/O2/O2+Cluster+Status



O2 Cluster Status

Created by Bergman, Andrew L., last modified about an hour ago

This page shows all service outages for the O2 cluster, including planned maintenance and unplanned events.

We also post updates on the HMS RC Twitter page.

ONLINE

July 8: notes after the July OS/Slurm update

- Jupyter Notebooks users should start a new environment and remove any old runtime directories.
- "sbatch" no longer uses the "–x11" option in the new version of Slurm. Just remove it from your script and X forwarding should work by default.
  - "srun" commands still require "–x11" to enable X forwarding, though.
- If you have any custom built software, you may need to recompile or relink it on O2.

Checking Storage Usage

- To check your *Active* storage available:

  mfk8@login01:~$ quota
- /home directory: each user gets 100 GB, total.
- Group directories: space varies, can be increased

  /n/groups/lab

  /n/data1/institution/department/lab

  /n/data2/institution/department/lab

Checking Storage Usage: scratch3

- mfk8@login01:~$ /n/cluster/bin/scratch3_quota.sh
- Must run from login node
- Quota is on user basis, not group basis
- Users are entitled to 10TB and up to 1 million files/directories
- Files not accessed for 30 days have been automatically purged

Storage Policies

- /home: 14 day snapshots + 60 day full backup
- /n/groups, /n/data1, /n/data2: 14 day snapshots + 60 day full backup
- /n/scratch3: 30 day retention, no backups

Snapshots

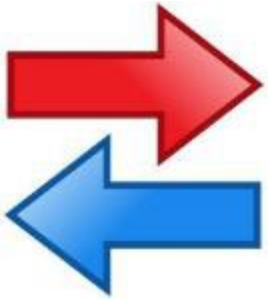
Snapshots (frozen) are retained for up to 60 days: recover data from a hidden `.snapshot` directory

Snapshots

Snapshots (frozen) are retained for up to 60 days:
recover data from a hidden `.snapshot` directory

- mfk8@compute-a:~$ cd .snapshot
- mfk8@compute-a:~$ ls
  O2_home_daily_2015-10-02-02-00
  O2_home_daily_2015-10-01-02-00
- mfk8@compute-a:~$ cd O2_home_daily_2015-10- 02-02-00
- mfk8@compute-a:~$ cp MyRetreivedFile ~

Research.files O2 access



- Research.files Active filesystem is accessible on select compute nodes via a `transfer partition` and `transfer cluster`
- Access to `transfer` allows cp/rsync of files
  - From: Research.files (/n/files)
  - To: O2 storage (/home, /n/groups, /n/data1, /n/data2, /n/scratch3)
  - And reverse direction
- Cannot use O2 to compute against data in Research.files, must be transferred

Data and Script Management

Data Management

As you run more jobs, you'll probably end up creating a whole bunch of files. In the same way it's important to plan bench projects beforehand, it's good to think early about how you should manage and organize all those files you'll be making.

http://datamanagement.hms.harvard.edu/

Top Data Management Best Practices

- **Planning**: Document the activities for the entire lifecycle. Create a Data Management Plan including sponsorship requirements, realistic budget, assigned responsibilities, all the data to be collected, *and each of these topics*! https://datamanagement.hms.harvard.edu/planning-overview

- **Organization**: Define how the data will be organized, including what is your folder hierarchy and how did you get from raw data to the final product? Consider versioning control for changes for both software and data products. https://datamanagement.hms.harvard.edu/versioning-1

- **Documentation**: Explain how the data will be documented such as naming conventions, acronyms, data fields and units. Determine whether there is a community-based metadata standard that can be adopted. Create a README file to record the metadata that will be associated with data. https://datamanagement.hms.harvard.edu/readme-files

Top Data Management Best Practices

- **Storage**: Your storage plan is integral to data management. Consider how the data will be stored and protected over the duration of the project. Identify short-term and long-term storage options. Remember to link accompanying metadata and related code and algorithms. https://datamanagement.hms.harvard.edu/storage-overview

- **Sharing**: Describe what data will be disseminated, to who, when, and where. Identify sharing tools to work with collaborators during the project and publish data in an open repository. Be sure to use standard, nonproprietary approaches and provide accompanying metadata & associated code. https://datamanagement.hms.harvard.edu/data-sharing

- **Retention**: Think about your preservation strategy from the start & adhere to your lab's standard practices. Research records should generally be retained no fewer than seven (7) years after the end of a research project or activity. https://datamanagement.hms.harvard.edu/data-retention

Logging into O2

Create a New O2 Account

- http://rc.hms.harvard.edu/#cluster

Click the red button and fill out the form!

- Your username will be your eCommons ID, with your eCommons password.

2-Factor Authentication

- More secure: thing you know, and thing you have
- Easiest: download Duo app to phone
- Setup details at: https://wiki.rc.hms.harvard.edu/display/O2/Two+Factor+Authentication+on+O2
- If you believe an email to be a phishing scheme, please forward to: phishing@harvard.edu

Logging Into O2: Mac

- Open a terminal (search “terminal”)

  ```
  ssh yourecommons@o2.hms.harvard.edu
  ```

- 2-Factor (when necessary): Choose 1/2/3 (push/phone/sms)

- To display graphics back to your desktop (X11 forwarding)
  Install XQuartz (google it) and have it running

  ```
  ssh -XY yourecommons@o2.hms.harvard.edu
  ```

Logging Into O2: Windows

- Install MobaXterm (google it)

```
ssh yourecommons@o2.hms.harvard.edu
```

- 2-Factor (when necessary): Choose 1/2/3 (push/phone/sms)

- To display graphics back to your desktop (X11 forwarding); MobaXterm already has an X11 client built-in

```
ssh -XY yourecommons@o2.hms.harvard.edu
```

Logging Into O2: Linux

- Open a terminal (search: “terminal”)

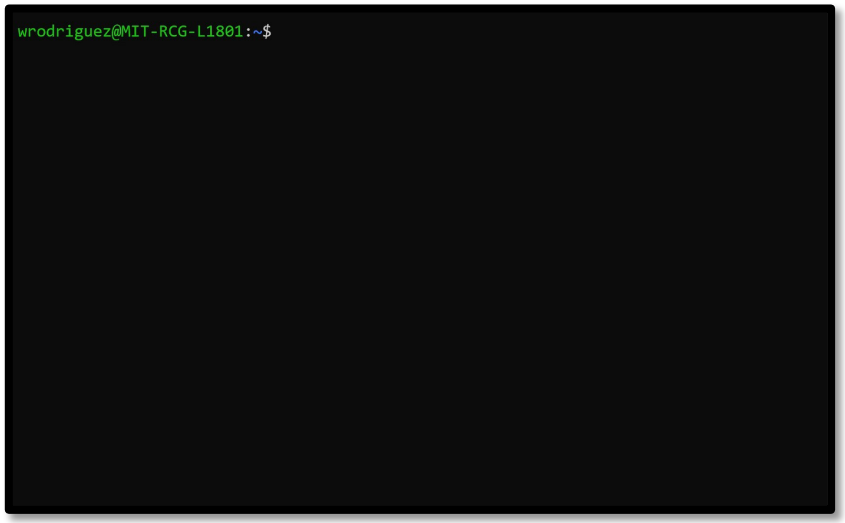
```
ssh yourecommons@o2.hms.harvard.edu
```

- 2-Factor (when necessary): Choose 1/2/3 (push/phone/sms)
- For graphics (X11 Forwarding)

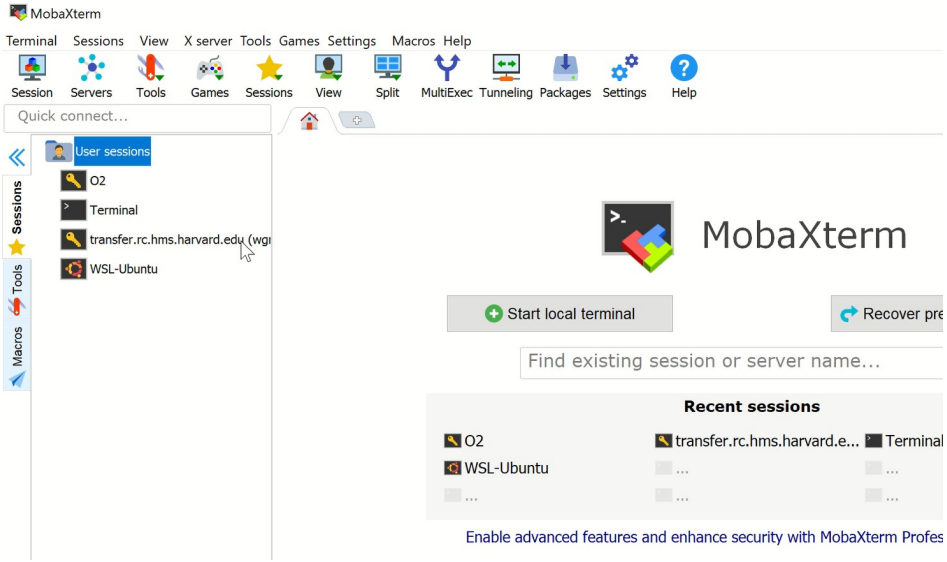
```
ssh -XY yourecommons@o2.hms.harvard.edu
```

Connecting to O2...

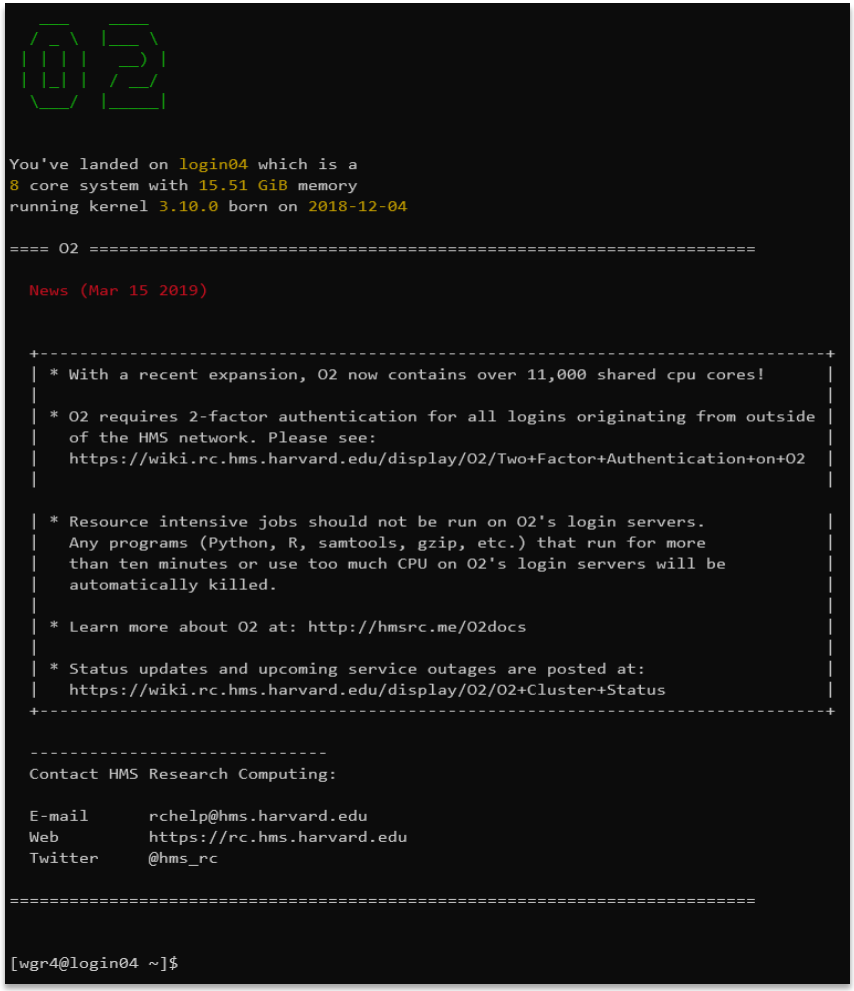
MAC/Linux: Terminal



Windows: MobaXterm



Welcome to O2!



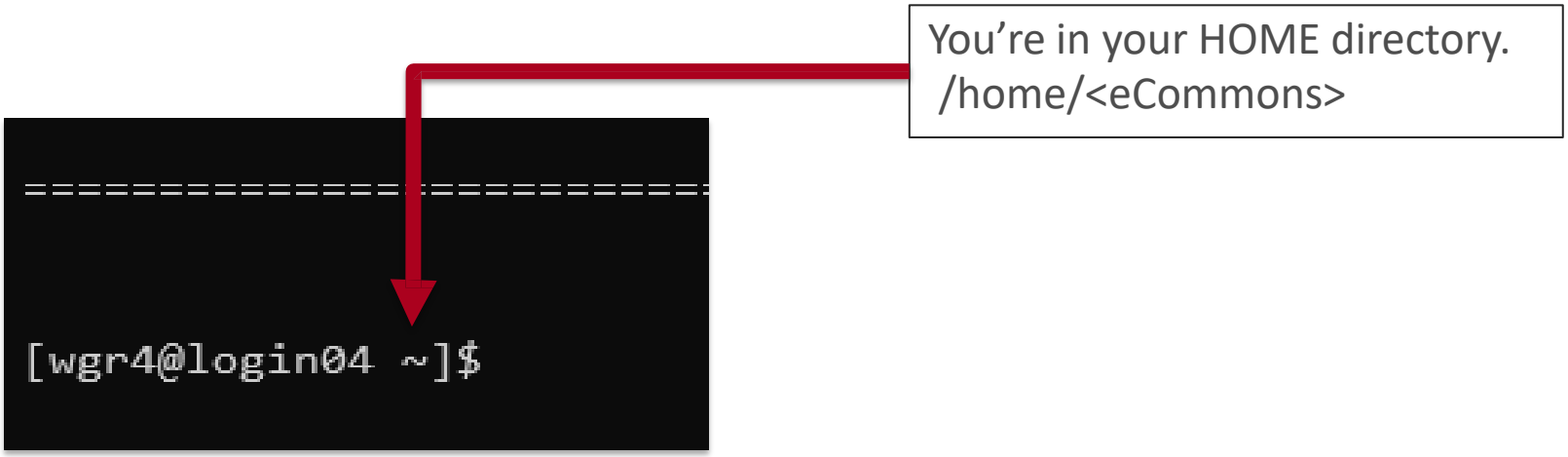
Welcome to O2!

Welcome to O2!

Welcome to O2!

You are logged into a “shell login server”
Do **not** meant for heavy lifting!

Welcome to O2!



Welcome to O2!

Getting Data Onto O2

- Use an FTP client of your choice
- Mac/Windows/Linux: Filezilla (google it)
- Connect to:

  Host: transfer.rc.hms.harvard.edu

  Username: <eCommons> (lowercase username)

  Password: <your_password>

  Port: 22

- Two-factor: will use default option setup in ~/.bashrc as

```
export DUO_PASSCODE=push/phone
```

Getting Data Onto O2

- Use an FTP client of your choice
- Mac/Windows/Linux: Filezilla (google it)
- Connect to:

  Host: transfer.rc.hms.harvard.edu

  Username: <eCommons> (lowercase username)

  Password: <your_password>

  Port: 22

- Two-factor: will use default option setup in ~/.bashrc as

```
export DUO_PASSCODE=push/phone
```

Interactive Sessions

- The login servers are not designed to handle intensive processes, and CPU usage is throttled. Start by entering your first job! This will (usually) log you into a "compute node!"

Interactive Sessions

- The login servers are not designed to handle intensive processes, and CPU usage is throttled. Start by entering your first job! This will (usually) log you into a "compute node!"

```
mfk8@login0~$ srun --pty -p interactive -t 0-2:00 --mem 2G bash
```

"`srun --pty`" is how interactive jobs are started

"`-p interactive`" is the partition

"`-t 0-2:00`" is the time limit (2 hours)

"`--mem 2G`" is the memory requested

```
mfk8@compute-a:~$
```

Interactive Sessions

- The login servers are not designed to handle intensive processes, and CPU usage is throttled. Start by entering your first job! This will (usually) log you into a "compute node!"

```
mfk8@login0~$ srun --pty -p interactive -t 0-2:00 --mem 2G bash
```

"`srun --pty`" is how interactive jobs are started

"`-p interactive`" is the partition

"`-t 0-2:00`" is the time limit (2 hours)

"`--mem 2G`" is the memory requested

```
mfk8@compute-a:~$
```

Class Practical

- Copy the class files and scripts to your /home

```
mfk8@compute-a:~$ cp -r /n/groups/rc-training/o2 ~
```

Class Practical

- Copy the class files and scripts to your /home

```
mfk8@compute-a:~$ cp -r /n/groups/rc-training/o2 ~
```

```
o2
├── date_parallel.sh
├── fastqc_job_array.sh
├── sample_1_R1.fastq
├── sample_2_R1.fastq
├── sample_3_R1.fastq
├── sample_4_R1.fastq
└── submit.slurm
```

Listing a Folder's Contents

- To see the contents of the current folder you are in (~ means "/home/username/"), type **lis**t (ls):

```
mfk8@compute-a:~$ ls
```

- To get the details of a folder's contents, add "-l"

```
mfk8@compute-a:~$ ls -l
```

- You don't have to be in a directory to see its contents

```
mfk8@compute-a:~$ ls /n/groups/rc-training/introtohpc
```

Viewing File Contents

- “less” to view file contents
- Navigate up/down, search
- “q” to quit

```
mfk8@compute-a:~$ less ~/.bashrc
```

Making a Folder (Directory)

- "mkdir" stands for "**m**a**k**e **dir**ectory."
- Create a new directory for this exercise
- Spaces are discouraged. (Underscores are fine!) Case counts in Linux.

```
mfk8@compute-a:~$ mkdir MyTestDir
```

Moving Around: Change Directory

- "cd" stands for "**c**hange **d**irectory"
- 1 period "."means "current directory"
- 2 periods ".." means "the directory above"

```
mfk8@compute-a:~$ cd MyTestDir
```

Notice how the prompt tells you where you are!

```
mfk8@compute-a:~/MyTestDir$ cd ..
mfk8@compute-a:~$
```

Creating a Simple Text File

- "Nano," "vi", "emacs" are simple command-line editors available.
- To create a new file, type the editor you want, then the name of the new file. To edit an existing file, do the same.

```
mfk8@compute-a:~$ nano myfile.txt
    This is my new file text.
    (Control-X to save (yes) and exit.)
mfk8@compute-a:~$
mfk8@compute-a:~$ ls
    myfile.txt
```

Copying Files

- "cp" to **co**p**y** a file from a destination to a new destination. "cp" "from" "to"
- `cp -r` to copy folders (recursively)

```
mfk8@compute-a:~$ cp myfile.txt MyTestDir/
```

- You can copy a file to the current folder or to a new folder with a different name by specifying a different name (rename)

```
mfk8@compute-a:~$ cp myfile.txt mycopy2.txt
#copying and renaming
```

Moving Data

- "**m**ove" "from" "to"

```
mfk8@compute-a~:$ mv MyTestDir/myfile.txt ~
#this rewrites myfile.txt, since it already exists!
mfk8@compute-a~:$ mv MyTestDir/ MyTestDir2/
#in-place move and rename
```

Removing Files/Folders

- "rm" to **rem**ove a file

```
mfk8@compute-a:~$ rm myfile.txt
```

- "rm -r" to remove a folder **r**ecursively

```
mfk8@compute-a:~$ rm -r MyTestDir2
```

Wildcard * Pattern Matching

- Useful for copying/removing/etc all files matching a certain pattern

- Example Case:

To copy “all” files ending in “.fastq”:

```
$ cp *.fastq NewFastqFolder/.
```

Software on O2

LMOD: Software Modules

- LMOD system adds directory paths of software into `$PATH` variable, and resolves software dependencies and conflicts
- Most software compiled against gcc-6.2.0: load first
- `$ module load gcc/6.2.0`
- `$ module avail` #to see software now available
- `$ module spider` #verbose list of software available on O2
- `$ module load software/version` #load software
- `$ module unload software/version` #unload
- `$ module purge` #dump all modules
- `$ module help <software>` #displays run info

Loading/Unloading Modules

- Load modules

```
$ module load gcc/6.2.0 bowtie2/2.2.9
```

- Which module version is loaded (if at all)?

```
$ which bowtie2
```

- See all modules you have loaded

```
$ module list
```

- Unload a specific module

```
$ module unload bowtie2/2.2.9
```

- Dump all modules

```
$ module purge
```

Public Databases on O2

/n/shared_db

- Folder structure: Genome/Software/Version/Database
- Example: /n/shared_db/hg19/uk/bowtie2

/n/groups/shared_databases

- Older databases
- Folder structure: Software/GenomeVersion/Database
- Example: /n/groups/shared_databases/star_reference/grch37

For more information:

https://wiki.rc.hms.harvard.edu/display/O2/Public+Databases

Compiling your own software

- Users can compile software in their /home or group directories, where they have permission

- Binaries just require “unzipping”, for example:
  - `tar -zxvf *.tgz`

Installing Software: Binary Example

- `mfk8@login01:~$ srun --pty -p interactive -t 0-12:00 --mem 8G bash`
- `mfk8@compute-a:~$ wget http://path/to/binary/mysoftware.tar.gz`
- `mfk8@compute-a:~$ tar -zxvf mysoftware.tar.gz`
- `mfk8@compute-a:~$ ls mysoftware/bin`

Installing Software: Binary Example

- `mfk8@login01:~$ srun --pty -p interactive -t 0-12:00 --mem 8G bash`
- `mfk8@compute-a:~$ wget http://path/to/binary/mysoftware.tar.gz`
- `mfk8@compute-a:~$ tar –zxvf mysoftware.tar.gz`
- `mfk8@compute-a:~$ ls mysoftware/bin`

Installing Software: Binary Example

- `mfk8@login01:~$ srun --pty -p interactive -t 0-12:00 --mem 8G bash`
- `mfk8@compute-a:~$ wget http://path/to/binary/mysoftware.tar.gz`
- `mfk8@compute-a:~$ tar -zxvf mysoftware.tar.gz`
- `mfk8@compute-a:~$ ls mysoftware/bin`

Programming Languages

- Python: load module (2.7.12, conda2, 3.6.0, 3.7.4)
  Use virtualenv to maintain packages (pip/easy install)

- R: load module (3.4.1-extra, 3.5.1-extra, 3.6.1, 4.0.1)
  Setup O2-specific personal R library

- Perl: load module (5.24.0, 5.30.0)
  Setup O2-specific local::lib (cpan/cpanm) in .bashrc

- MATLAB: load module(2017a/b, 2018a/b, 2019a)
  Setup cluster profile specific to O2

MPI on O2

- Message Passing Interface
- Distribute work over multiple nodes, allowing for the utilization of more cores
- openMPI-3.1.0 compiled against GCC 6.2.0
- MATLAB, Python, R, Perl, Java, C++, Fortran implementations
- Needs wrapper function “mpirun” to dispatch to compute nodes with SLURM
- Run in “mpi” partition `-p mpi` after obtaining access to partition
- Core cap: **640 processors, 5-day runtime**

Constructing Jobs

Submitting Jobs

- In an “interactive session”, programs can be called directly.

```
mfk8@compute-a:~$ bowtie -c 4 hg19 file1_1.fq file1_2.fq
```

- From the login shell (and also interactive or any compute nodes), a program is submitted to O2 via a job (sbatch)

```
mfk8@compute-a:~$ sbatch mybowtiejob.sh
```

Submitting Jobs

- In an “interactive session”, programs can be called directly.

```
mfk8@compute-a:~$ bowtie -c 4 hg19 file1_1.fq file1_2.fq
```

- From the login shell (and also interactive or any compute nodes), a program is submitted to O2 via a job (sbatch)

```
mfk8@compute-a:~$ sbatch mybowtiejob.sh
```

Jobs: sbatch

- All in one line: `--wrap="command here"`
  - This approach is not recommended!

```
mfk8@login01:~$ sbatch -p partition -t 0-1:00 --wrap="command_here"
```

- Using a complete shell script is recommended!

```
mfk8@login01:~$ sbatch completeSlurmJob.run
```

Jobs: sbatch

- All in one line: `--wrap="command here"`
  - This approach is not recommended!

```
mfk8@login01:~$ sbatch -p partition -t 0-1:00 --wrap="command_here"
```

- Using a complete shell script is recommended!

```
mfk8@login01:~$ sbatch completeSlurmJob.run
```

```
Line 1: #!/bin/bash
Line 2: #SBATCH -p short
Line 3: #SBATCH -t 0-1:00
Line 4: command_here ..
```

Partitions (queues): -p

Partition	Priority	Max Runtime	Max Cores	Limits
short	12	12 hours	20	
medium	6	5 days	20	
long	4	30 days	20	
interactive	14	12 hours	20	2 job limit
priority	14	30 days	20	2 job limit
mpi	12	5 days	640	20 core min
highmem		5 days	20	
gpu		160 GPU hours	34 (total)	420G (total)
transfer		5 days	4	

Wall-Time: -t

- `-t days-hours:minutes`
- `-t hours:minutes:seconds`
- Need to specify how long you estimate your job will run for
- Aim for 125%
- Subject to maximum per partition
- Excessive wall-time (like partition max) takes longer to dispatch, and affect fair-share

CPU: -c

- `-c X` to designate CPU: max 20
- `-N X` to constrain all cores to X nodes
- CPU time: wall time (-t) * (-c) CPUs used
- Unable to use CPU not requested (no overefficient jobs): cgroups constraint
- Adding more cores does not mean jobs will scale linearly with time, and causes longer pend times

Memory: --mem

- Only 1G is allocated by default
- `--mem XG` #total memory over all cores
- `--mem-per-cpu XG` #total memory per CPU requested, use for MPI
- If you don't include a unit request (like G), it defaults to Megabytes

Job Construction

```
#!/bin/bash
#SBATCH -p short #partition
#SBATCH -t 0-01:00 #time days-hr:min
#SBATCH -c X #number of cores
#SBATCH --mem=XG #memory per job (all cores), GB
#SBATCH -o %j.out #out file
#SBATCH -e %j.err #error file
#SBATCH --mail-type=BEGIN,END,FAIL,ALL
#SBATCH --mail-user=mfk8@med.harvard.edu
# put any module load commands here
# put any analysis commands you want to run here
```

Job Construction

```
Line 1:  #!/bin/bash
Line 2:  #SBATCH -p short #partition
Line 3:  #SBATCH -t 0-01:00 #time days-hr:min
Line 4:  #SBATCH -c X #number of cores
Line 5:  #SBATCH --mem=XG #memory per job (all cores), GB
Line 6:  #SBATCH -o %j.out #out file
Line 7:  #SBATCH -e %j.err #error file
Line 8:  #SBATCH --mail-type=BEGIN,END,FAIL,ALL
Line 9:  #SBATCH --mail-user=mfk8@med.harvard.edu
Line 10: # put any module load commands here
Line 11: # put any analysis commands you want to run here
```

Output/Error Files

- Can add jobid to filename with `%j`
- Sample:
  ```
  -e %j.err
  -o %j.out
  ```
- SLURM by default creates this outfile: `slurm-<jobid>.out`
- Additional Flags
- `%a` job array id
- `%A` master array job id
- `%N` node name
- `%u` user id

Mail

- Mail is not auto-generated upon completion/failure
- `#SBATCH --mail-type= NONE, BEGIN, END, FAIL, REQUEUE, ALL`
- `#SBATCH --mail-user=mfk8@med.harvard.edu`
- Not recommended, not a verbose output
- Use `O2sacct` or `sacct` commands instead

Practical: simple sbatch script

- From your ~/o2 directory,

```
$ sbatch submit.slurm
```

```
#!/bin/bash
#SBATCH -p short                    # Partition to submit to
#SBATCH -t 0-00:01                  # Time in minutes
#SBATCH -c 1                        # Number of cores requested
#SBATCH -N 1                        # Ensure that all cores are on one machine
#SBATCH --mem=2G                    # Memory total in GB
#SBATCH -o hostname.%j.out          # Standard out goes to this file
#SBATCH -e hostname.%j.err          # Standard err goes to this file
srun hostname                       #command
```

Command Line Arguments

- slurm scripts can take command line arguments
  Reference as $1, $2 etc
- `sbatch submit.run 25 output.txt`

```
#!/bin/bash
#SBATCH -p short
#SBATCH -t 0-1:00
python myscript.py $1 $2
```

The above runs as

```
python myscript.py 25 output.txt
```

Job Priority

- Dynamically assigned
- Factors contributing:
- Age, Fairshare, Partition, QOS, Nice
- Fairshare: 0-1 scale
- Check your fairshare:
  - `$ sshare -Uu $USER`
- Check job priority values for your pending jobs:
  - `$ sprio -u $USER`

X11 on O2

- To visualize or initiate plot devices, an X11 device must be active
- Mac: XQuartz installed and running
- Windows: MobaXterm download software
- Login: `ssh -XY`
- SSH keys: Must be setup for sbatch jobs!

  https://wiki.rc.hms.harvard.edu/x/ZoArAQ
- To interactives, `srun` add: `--x11`

Job Management

Job Monitoring: Current jobs

- `$ O2squeue`
  - JOBID, PARTITION, STATE, TIME_LIMIT, TIME, NODELIST(REASON), ELIGIBLE_TIME, START_TIME, TRES
  - O2squeue documentation

- *Other options*:
- `$ squeue -u eCommons –t RUNNING/PENDING`
- `$ squeue -u eCommons –p Partition`
- `$ squeue -u eCommons –start`

- *Detailed job info:*

  `$ scontrol show jobid <jobid>`

Job Information: Past Jobs

- `$ O2sacct`
  - JobID, Partition, State , NodeList, Start, Timelimit, Elapsed, CPUTime , TotalCPU, AllocTRES, MaxRSS
  - Can specify job ID, job status, and/or timeframe to report accounting info for
  - O2sacct documentation

- *Other options*:
- `$ sacct -j jobid`
- `$ sacct -r partition`
- `$ squeue -s state`
- `$ sacct --helpformat` #get available fields you can specify

Slurm Job States

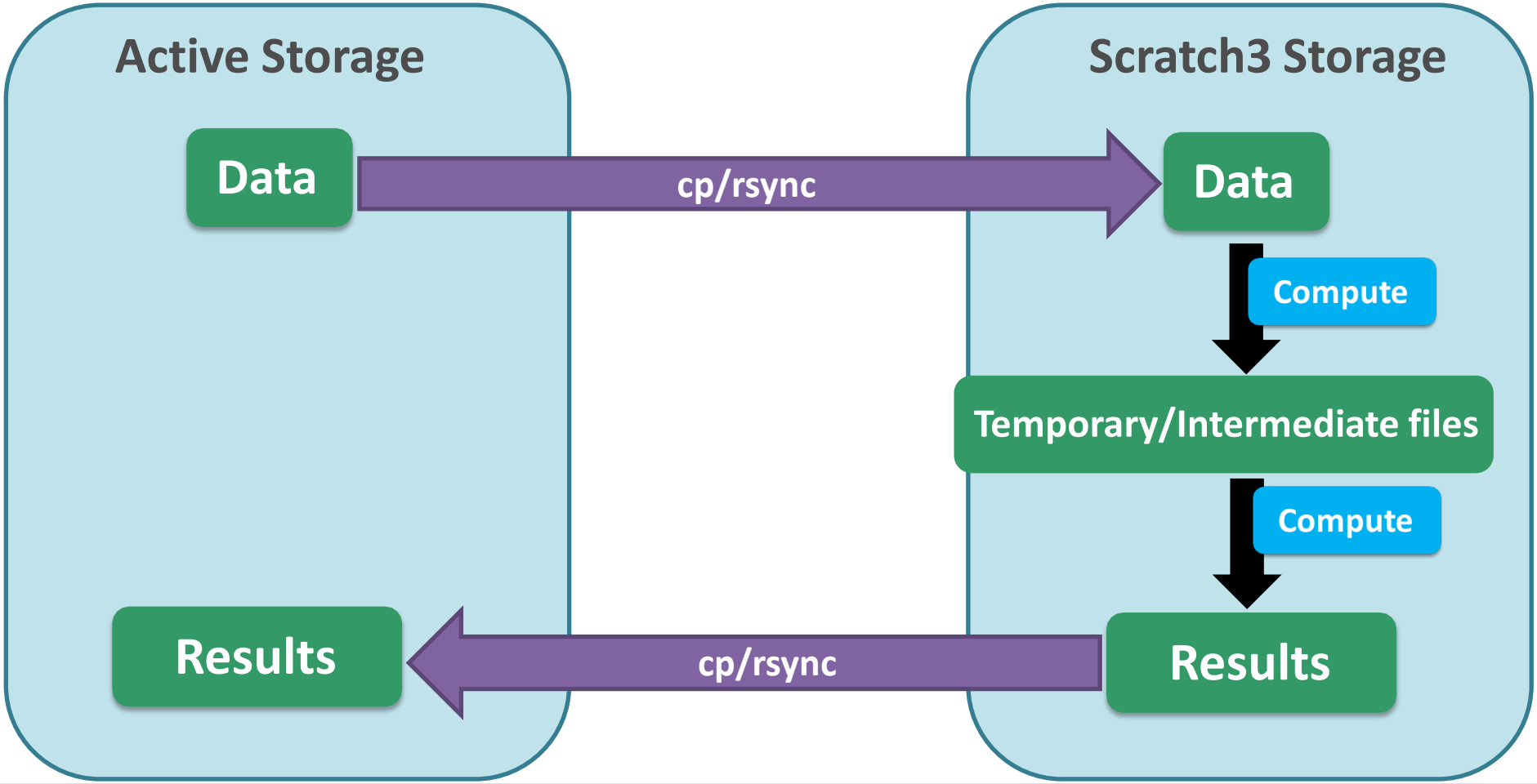
- BF BOOT_FAIL
- CA CANCELLED
- CD COMPLETED
- CF CONFIGURING
- CG COMPLETING
- DL DEADLINE
- F FAILED
- NF NODE_FAIL
- OOM OUT_OF_MEMORY
- PD PENDING
- PR PREEMPTED
- R RUNNING
- RS RESIZING
- S SUSPENDED
- TO TIMEOUT

Cancelling/Pausing Jobs

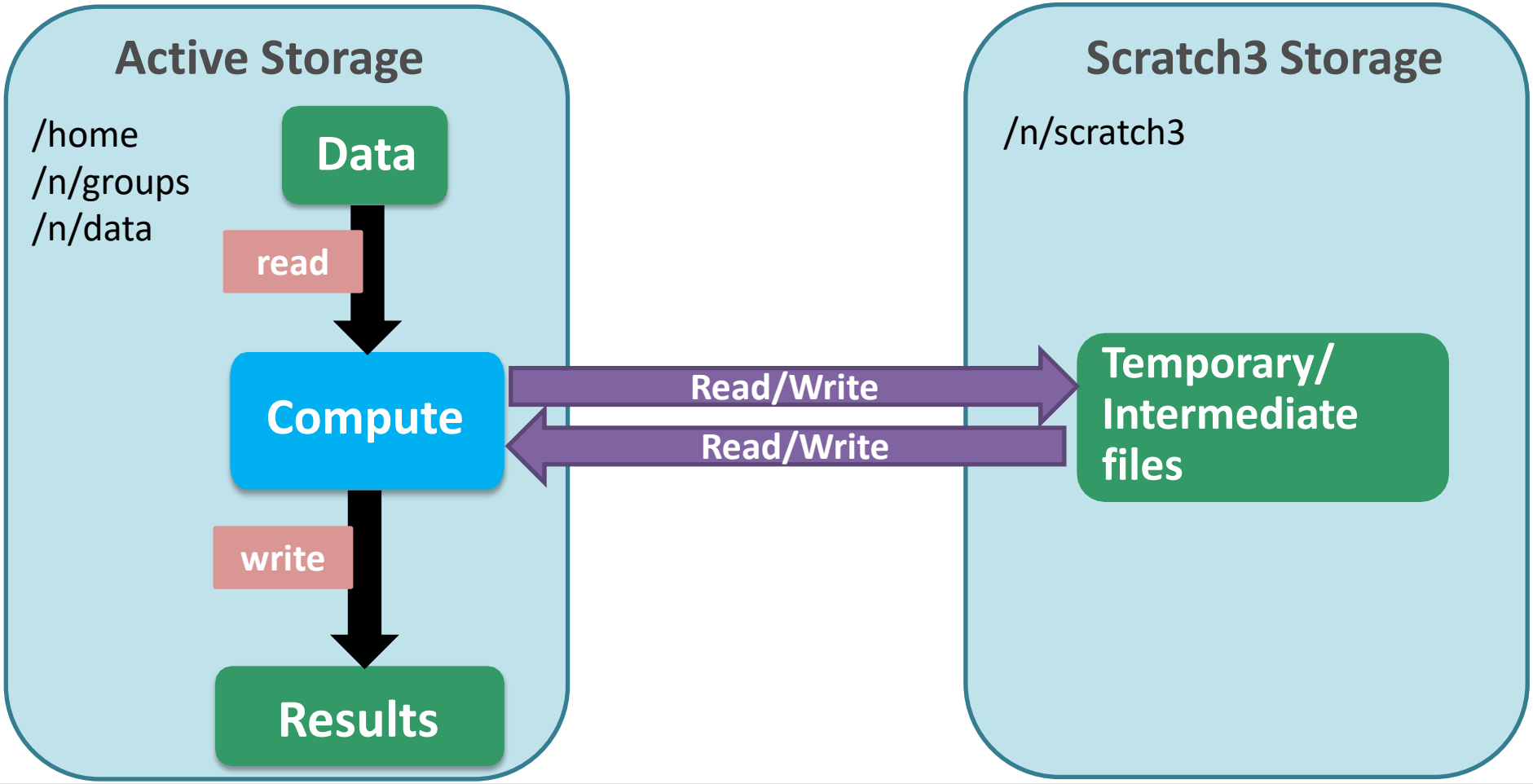
- `$ scancel <jobid>` #Cancels specific job
- `$ scancel -t PENDING` #Cancels pending job
- `$ scancel --name JOBNAME` #Cancels job by name
- `$ scancel jobid_[indices]` #array indices
- `$ scontrol hold <jobid>` #pause pending jobs
- `$ scontrol release <jobid>` #resume

Utilizing /n/scratch3

/n/scratch3 Workflow: Redundancy



/n/scratch3 Workflow: Best Practice



Utilizing /n/scratch3

- Designed for writing a large number of small temporary files
- Use cases:
  - Keep original files in /n/groups (/n/data*) or /home, write intermediate files to /n/scratch3, write final files to /n/groups (/n/data*) or /home
  - Change working directory to /n/scratch3, read files from /n/groups (/n/data*) or /home, write temp files to working directory, write or copy output back to /n/groups (/n/data*) or /home
  - Copy input files to /n/scratch3, compute against, copy output files to /n/groups (/n/data*) or /home

File Properties

- "chmod" to change who can read/write/execute files/directories
- chmod options file/directory
  - Who? **u**ser **g**roup **o**thers **a**ll (u/g/o/a)
  - What? **r**ead **w**rite e**x**ecute (r/w/x)
  - Do? +/-
- For example:
  - To make the file called "myfile" an executable

```
$ chmod u+x myfile owner
```

  - To take away permission from others to read/write/execute for the same file

```
$ chmod o-rwx myfile
```

OMERO

- Microscopy image and metadata management service of the Image Management Core
- Java Application or web interface
- Browse and filter through dimensions, z-sections and timepoints
- Analyze through Java, Python, C++ or MATLAB, Fiji/ImageJ using API/plugins to interface with OMERO server
- O2: CLI environment module, Java desktop client, or web interface
- Upload data from research.files, /home, /n/groups, /n/data1, /n/data2
- imc-support@hms.harvard.edu
- http://imc.hms.harvard.edu

For more direction

Email: rchelp@hms.harvard.edu

Website: http://rc.hms.harvard.edu

Wiki: https://wiki.rc.hms.harvard.edu/display/O2/O2

Phone: 617-432-2000 (HMS IT Service Desk, 8a-5p)

Twitter: @hms_rc

Location: Gordon Hall 500, 5th Floor, 25 Shattuck Street

- https://rc.hms.harvard.edu/office-hours/ for Zoom web conferencing during remote work

Office hours: Wednesdays 1-3p for pressing needs, but appointments encouraged.

Please fill out the survey

- Accessible through the Harvard Training Portal
  - https://trainingportal.harvard.edu/
- Click on “Me” then “Intro to O2”
- Scroll to “Evaluations” and click on the survey

- We appreciate any feedback or comments!