# Intro to O2

HMS Research Computing

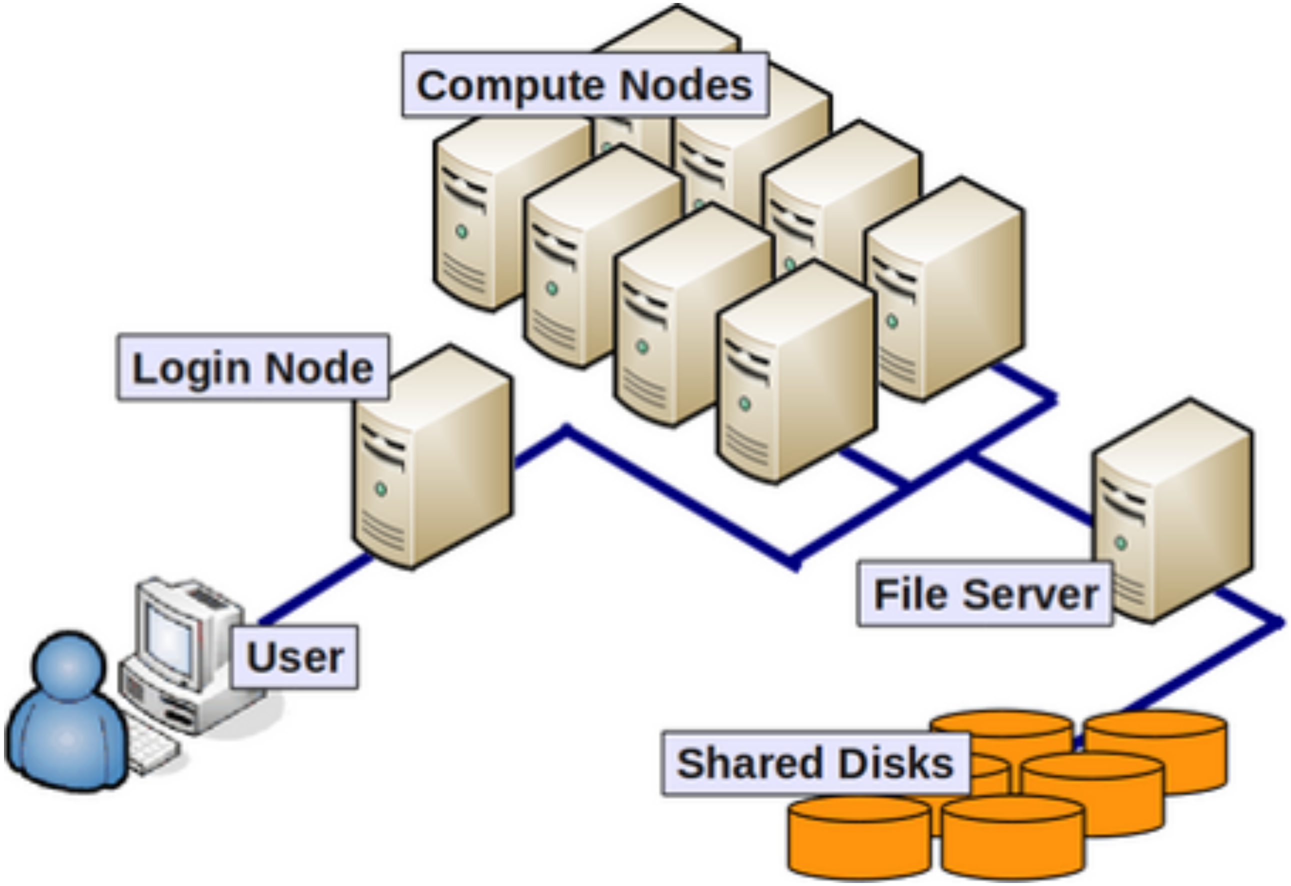
# Welcome to O2!

- 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

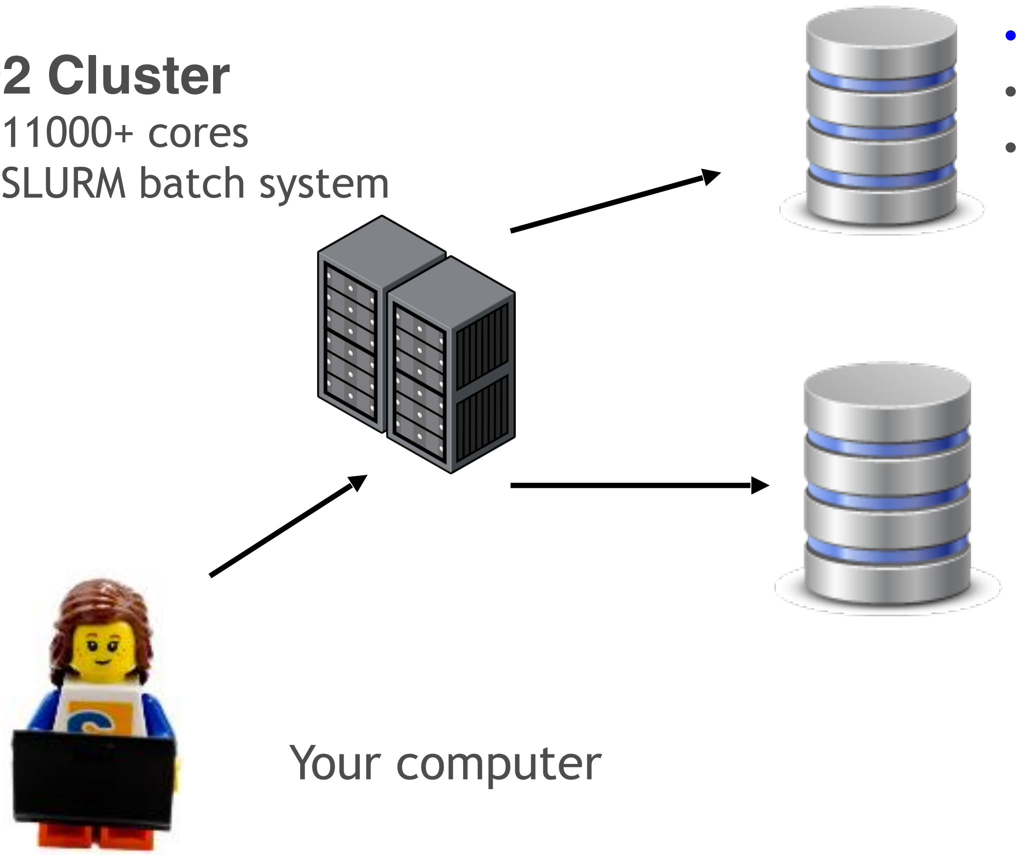
- 11000+ cores
- 32, 28 or 20 cores per node
- 256-160 GB RAM per node (8-9GB/core)
- 9x756 GB and 1x1TB highmem nodes*
- 32 GPUs (8 M40 / 16 K80 / 8 V100*)
- Login/load balancer 6 VM (8 cores/16GB memory)
- InfiniBand connectivity between nodes available
- CentOS 7

# Generic Cluster Architecture



# Storage on O2

# O2 Primary Storage



**O2 Cluster**

- 11000+ cores
- SLURM batch system

**/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

Your computer

# Temporary "Scratch" storage

l·u·s·t·r·e

Scalable Storage

- /n/scratch2/user_id
- For data only needed temporarily during analyses
- Fastest connection to O2 compute nodes
- Each account can use up to 10 TB and 1 million files/directories
- Files not accessed (atime) after 30 days are automatically purged
- No backups!
- No Tier 2/3 options
- Lustre --> a high-performance parallel file system
- More than 1 PB of total shared disk space

# Tier 2 Storage

- Cloud storage offering
- Data not frequently accessed moved to a cheaper platform
- Still visible in Isilon (/n/data1-/n/data2-/n/groups)
- Access to Tier 2 data is subject to initial delays ranging from seconds to hours based on the size of data.
- http://it.hms.harvard.edu/services/storage-data-management/tiered-storage

# Checking Storage Usage

- To check your storage available:

```
mfk8@login01:~$ quota
  mfk8@login01:~$ du -sh --apparent-size  ~
```

- /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
- Only shows Tier 1 usage, does not include Tier 2/3

# Checking Storage Usage: scratch2

-
  ```
  mfk8@login01:~$ lfs quota -h /n/scratch2
  ```
- Quota is on user basis, not group basis
- Users are entitled to 10TB and up to 1 million files/ directories
- Files not accessed for 29 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/scratch2: 30 day retention, no backups
- Tier 2 and Tier 3 storage options available

# Snapshots

- Snapshots (static) 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

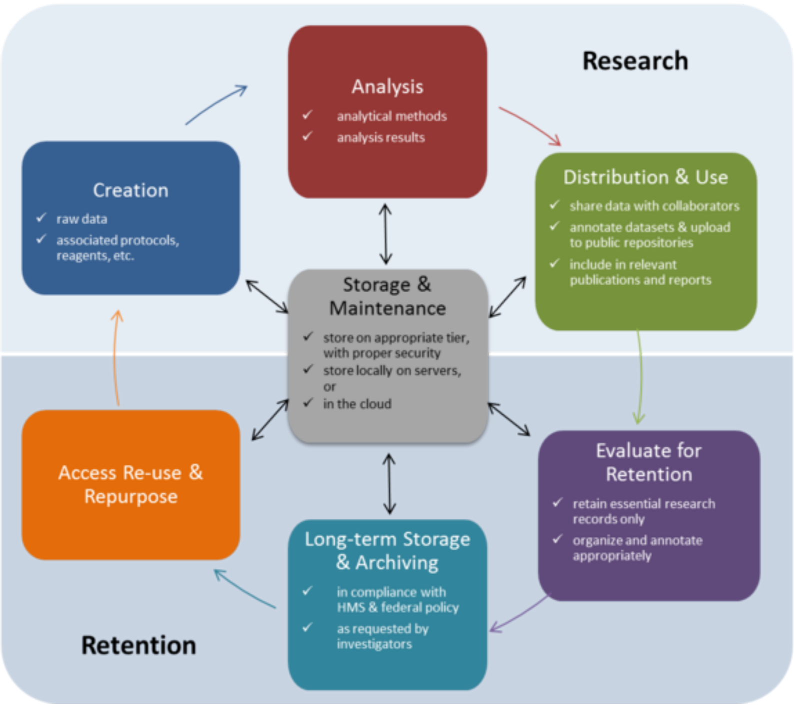
- Research.files Tier 1 filesystem is accessible on select compute nodes via a `transfer partition` and `transfer cluster`
- Access to `transfer` allows cp/rsync of files
  - From: Research.files (/n/files)
  - To: O2 storage (/home, /n/groups, /n/data1, /n/data2, /n/scratch2)
  - And reverse direction
- 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.

## Data lifecycle for biomedical research



Research

Analysis
- ✓ analytical methods
- ✓ analysis results

Distribution & Use
- ✓ share data with collaborators
- ✓ annotate datasets & upload to public repositories
- ✓ include in relevant publications and reports

Creation
- ✓ raw data
- ✓ associated protocols, reagents, etc.

Storage & Maintenance
- ✓ store on appropriate tier, with proper security
- ✓ store locally on servers, or
- ✓ in the cloud

Evaluate for Retention
- ✓ retain essential research records only
- ✓ organize and annotate appropriately

Access Re-use & Repurpose

Long-term Storage & Archiving
- ✓ in compliance with HMS & federal policy
- ✓ as requested by investigators

Retention

# 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.

Account Request

# New feature: 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
```

# Welcome to O2!

- Where are you in O2?

  See your terminal!

```
mfk8@login01:~$
```

- You are logged into a “shell login server”, login01-06.These are not meant for heavy lifting!

- You are in your home directory.  This is symbolized by the “tilde ” (~).  This is shorthand for:  /home/eCommons

- You are in a bash environment.  “$” means “ready to accept your commands!”

# 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:~$
```

# Linux Basics

# Class Practical

- Login to O2 via

```
$ ssh ecommons@o2.hms.harvard.edu
```

- Copy the class files and scripts to your /home

```
$ cp -r /n/groups/rc-training/o2 ~
```

# Listing a Folder’s Contents

- To see the contents of the current folder you are in (~ means “/home/username/”), type **list** (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 “**make 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 **copy** 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

- “**move**” “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/.
```

# Getting Data Onto O2

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

transfer.rc.hms.harvard.edu

your username and password (lowercase username)

port 22

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

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

# 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 software currently available
- $ `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

- Loading modules

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

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

```
$ which bowtie2
```

- See all modules loaded

```
$ module list
```

- Unloading modules

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

- Dump all modules

```
$ module purge
```

# Compiling your own software

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

- Binaries just require “unzipping” (ie 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
```

# Programming Languages

- Python: load module (2.7.12, conda2, 3.6.0)

  use virtualenv to maintain packages (pip/easy install)

- R: load module (3.2.5, 3.3.3, 3.4.1-extra, 3.5.1-extra)

  Setup O2-specific personal R library, .Renviron
  (install.packages/biocLite/install_github)

- Perl: load module (5.24.0)

  Setup O2-specific local::lib (cpan/cpanm) in .bashrc

- MATLAB: load module(2016b, 2017a/b, 2018a/b)

  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 being added 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
```

# Jobs: sbatch

- All in one line: --wrap=“command here”
- #not recommended

```
sbatch -p partition -t 0-1:00 -wrap=“command_here”
```

- Complete shell script #recommended

```
sbatch completeSlurmJob.run
```

```
#!/bin/bash
#SBATCH –p short
#SBATCH –t 0-1:00
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 | 12 | 5 days | 20 | 750G |
| 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% over
- 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
- No unit request (G) 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
```

# 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 like LSF
- Use sacct 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
```

# Job Master Allocations/Job Steps

- “sbatch” or “salloc” will create a master allocation of shared resources
- To run job steps or multiple commands in parallel with resources from the master allocation, use srun

```
#!/bin/bash
#SBATCH -c 8
#SBATCH --mem 32000
srun -c 2 --mem=8000 COMMAND1 &
srun -c 4 --mem=8000 COMMAND2 &
srun -c 1 --mem=4000 COMMAND3 &
srun -c 1 --mem 12000 COMMAND4 &
wait #necessary

# starts 4 independent multicore tasks with memory constraints
```

# Practical: Job Parallelization

- From ~/o2:

```
$ sbatch date_parallel.sh
```

Contents:

```
#SBATCH -c 3
srun -c 1 date &
srun -c 1 sleep 2m &
srun  -c 2 date &
srun  -c 3 date &
wait
```

- Output file can be read as

```
$ less parallel.*.out
```

# Job Arrays

- `sbatch --array=1-30 submit.sh`
- `#SBATCH --array=1-30`
- slurm creates:
- `$SLURM_JOB_ID` #jobid of each job in array %j
- `$SLURM_ARRAY_JOB_ID` #jobid of entire array %A
- `${SLURM_ARRAY_TASK_ID}` #index of job %a

Example:

Files named: File1.txt File2.txt File3.txt ...File30.txt

Execution is:

```
myscript.sh File${SLURM_ARRAY_TASK_ID}.txt
```

# Practical: Job Array

- From ~/o2

```
$ sbatch --array=1-4 fastqc_job_array.sh
```

- Relevant file pieces:

```
#SBATCH –o fastqc_%A_%a.out
module load fastqc/0.11.5
fastqc sample_"${SLURM_ARRAY_TASK_ID}"_R1.fastq
```

- Creates fastqc report for reach fastq file
- Creates output progress file named `fastqc_ArrayId_ArrayIndex.out`

# Job Dependencies

- `sbatch --dependency=`
- `after:jobid` #(asynchronous)
- `afterany:jobid` #after exit or done
- `afterok:jobid` #success, exit code 0
- `afternotok:jobid` #failure
- `singleton` #after jobs with same name have terminated
- `--kill-on-invalid-dep=<yes|no>` #kill on unmet dependency (on by default)

# 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

#runs as
python myscript.py 25 output.txt
```

# Job Priority

- Dynamically assigned
- Factors contributing:
- Age, Fairshare, Partition, QOS, Nice
- Fairshare: 0-1 scale
- $ sprio -u $USER

# X11 on O2

- To visualize or initiate plot devices, an X11 device must be active
- Mac: Xquartz installed and running
- Windows: Xming installed and running
- Login: ssh –XY
- SSH keys: Must be setup for sbatch jobs!

https://wiki.rc.hms.harvard.edu/display/O2/How+to+Generate+SSH+Keys

- To sbatch jobs add: --x11=batch, with ssh keys set up
- To interactives, srun add: --x11

# Job Management

# Job Monitoring

- `$ O2squeue`
- Other options:
- `$ squeue -u eCommons -t RUNNING/PENDING`
- `$ squeue -u eCommons -p partition`
- `$ squeue -u eCommons --start`
- Detailed job info:

  ```
  $ scontrol show jobid <jobid>
  ```
- Completed job statistics:

  ```
  $ sacct -j <jobid> --format=JobID,JobName,MaxRSS,Elapsed
  ```

# Job information: sacct

- Advanced job accounting options
- `$ sacct -j jobid`

Options:

```
--name
-r/--partition
-s/--state
-o/--format
```

- `$ sacct –u eCommons`
- `$ sacct --helpformat` #get available accounting features

# sacct: basic information

- `sacct --helpformat` displays fields
- RC Recommends:

```
mfk8@login02~:$ O2sacct
```

- JobID, Partition, State, NodeList, Start, Timelimit Elapsed CPUTime, TotalCPU , AllocTRES MaxRSS

# sacct: state

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

# Cancelling/Pausing Jobs

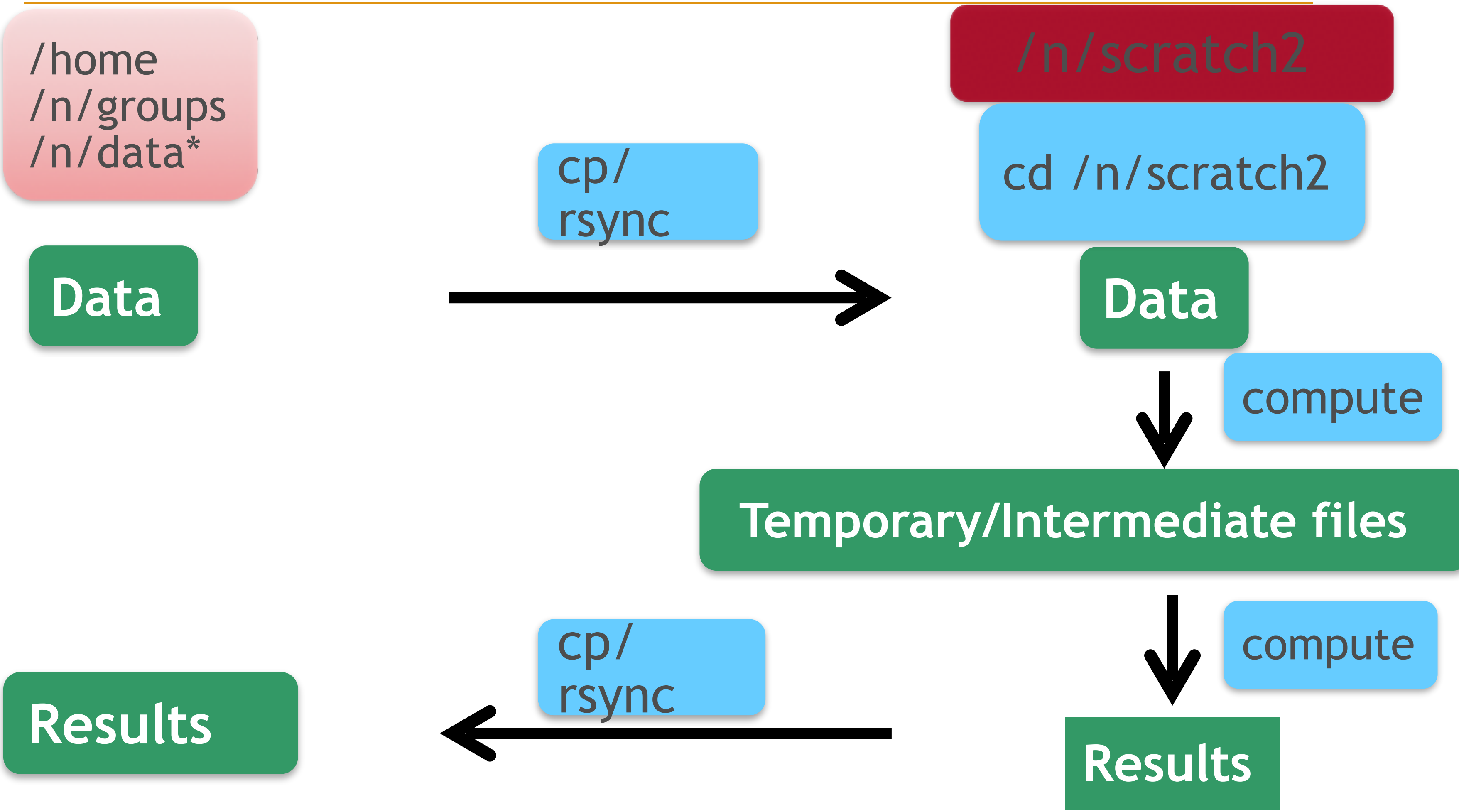
- `$ scancel <jobid>`
- `$ scancel –t PENDING`
- `$ scancel --name JOBNAME`
- `$ scancel jobid_[indices]` #array indices
- `$ scontrol hold <jobid>` #pause pending jobs
- `$ scontrol release <jobid>` #resume

# Utilizing /n/scratch2

# Utilizing /n/scratch2

- Designed for writing large, temporary files
- Use cases:
- Keep original files in /n/groups (/n/data*) or /home, write intermediate files to /n/scratch2, write final files to /n/groups (/n/data*) or /home
- Change working directory to /n/scratch2, 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/scratch2, compute against, copy  output files to /n/groups (/n/data*) or /home

# /n/scratch2 Workflow: Redundancy



/home
/n/groups
/n/data*

Data

cp/
rsync

/n/scratch2

cd /n/scratch2

Data

compute

Temporary/Intermediate files

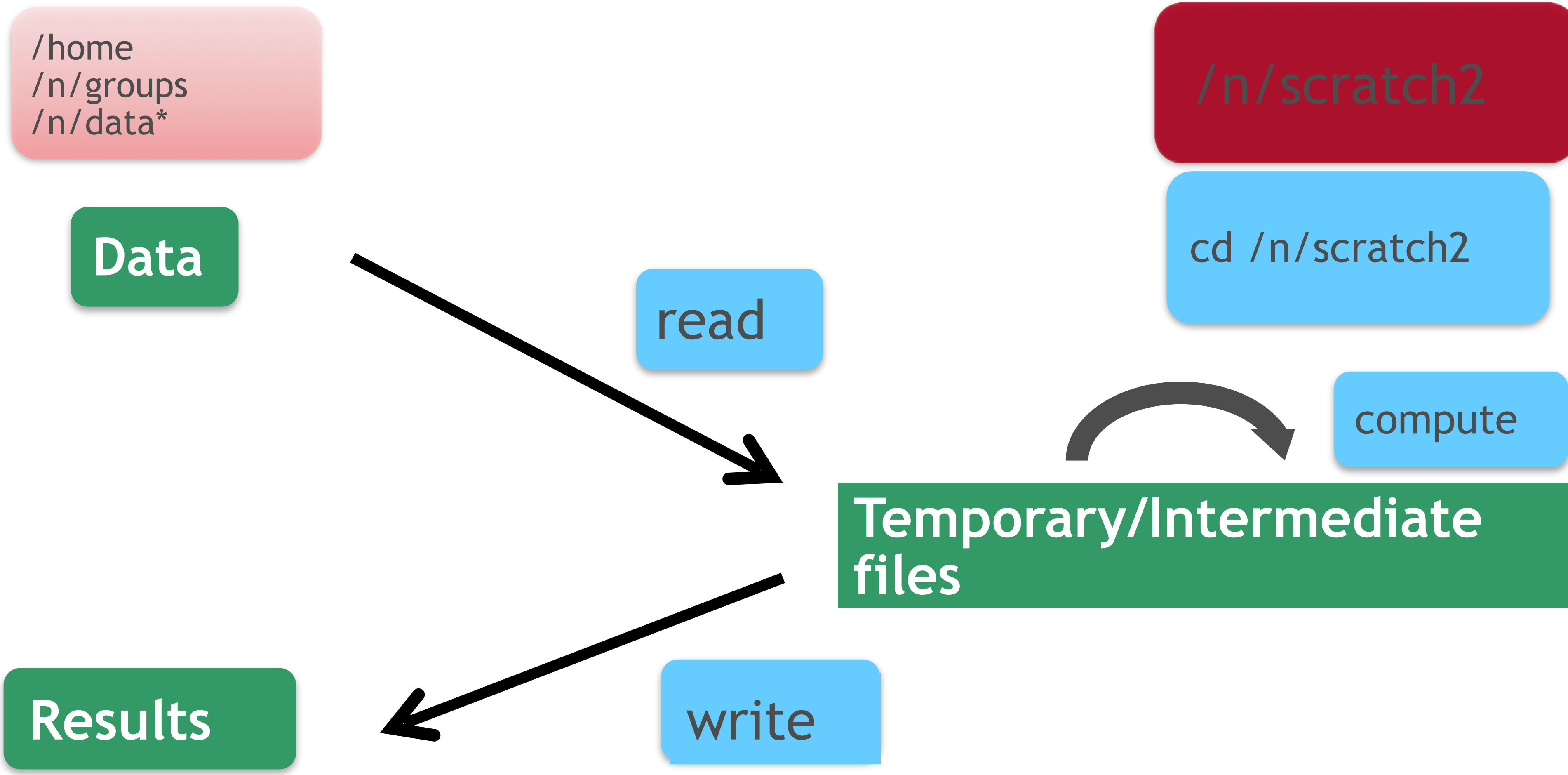
compute

Results

cp/
rsync

Results

# /n/scratch2 Workflow: Medium Flexibility



/home
/n/groups
/n/data*

Data

read

/n/scratch2

cd /n/scratch2

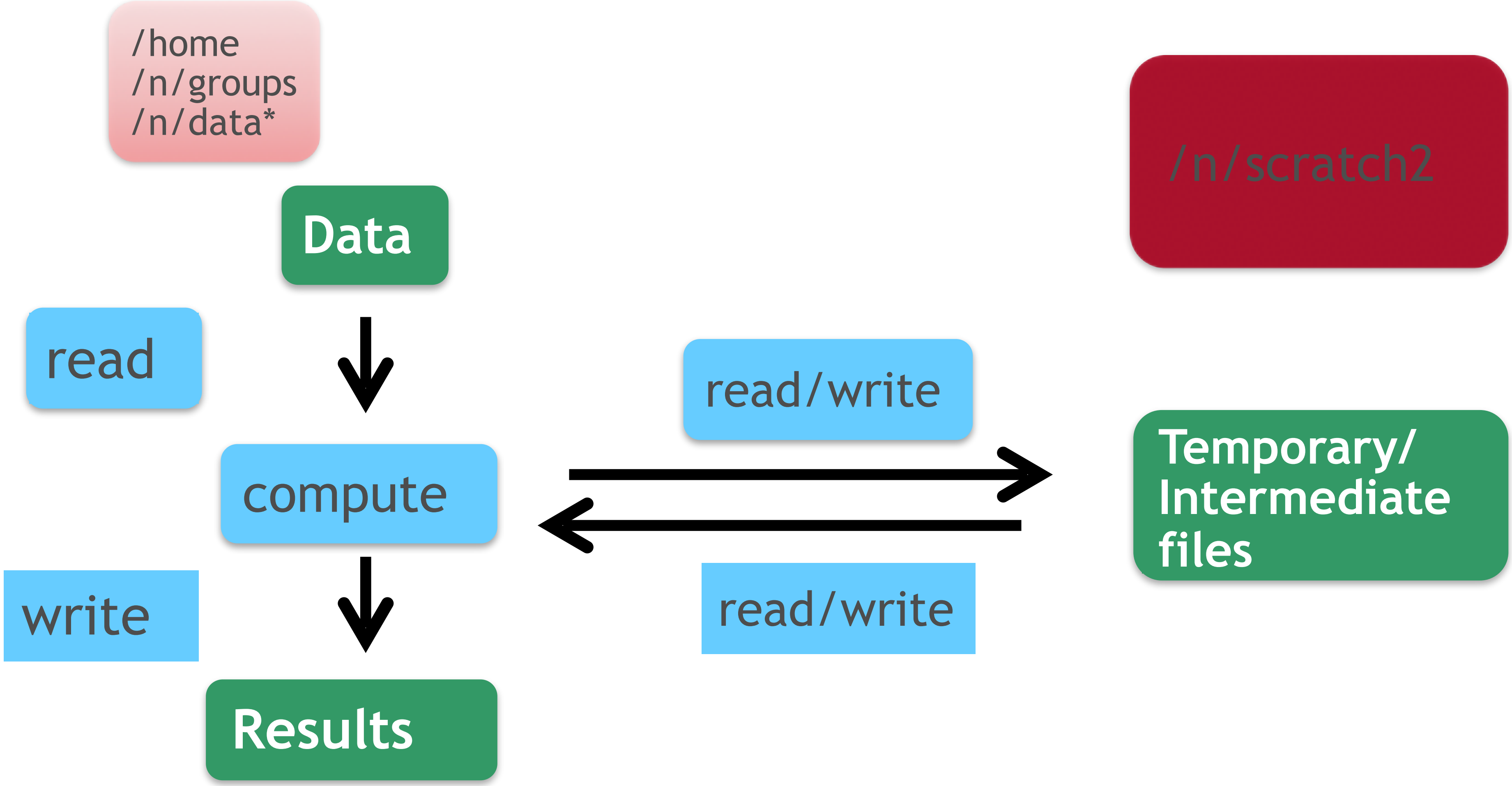
compute

Temporary/Intermediate files

write

Results

# /n/scratch2 Workflow: Best Practice



# File Properties

- “chmod” to change who can read/write/execute files/directories

chmod options file/directory

Who? **u**ser **g**roup **o**thers **all** (u/g/o/a)

What? **r**ead **w**rite e**x**ecute (r/w/x)

Do? +/-

- chmod u+x myfile #makes a file executable to owner
- chmod o-rwx myfile #takes away permission from others to read/write execute

# 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

- http://hmsrc.me/O2docs
- http://rc.hms.harvard.edu
- RC Office Hours: Wed 1-3p Gordon Hall 500
- rchelp@hms.harvard.edu