

Introduction to High Performance Computing and Orchestra

HMS Research Computing

Spring 2017



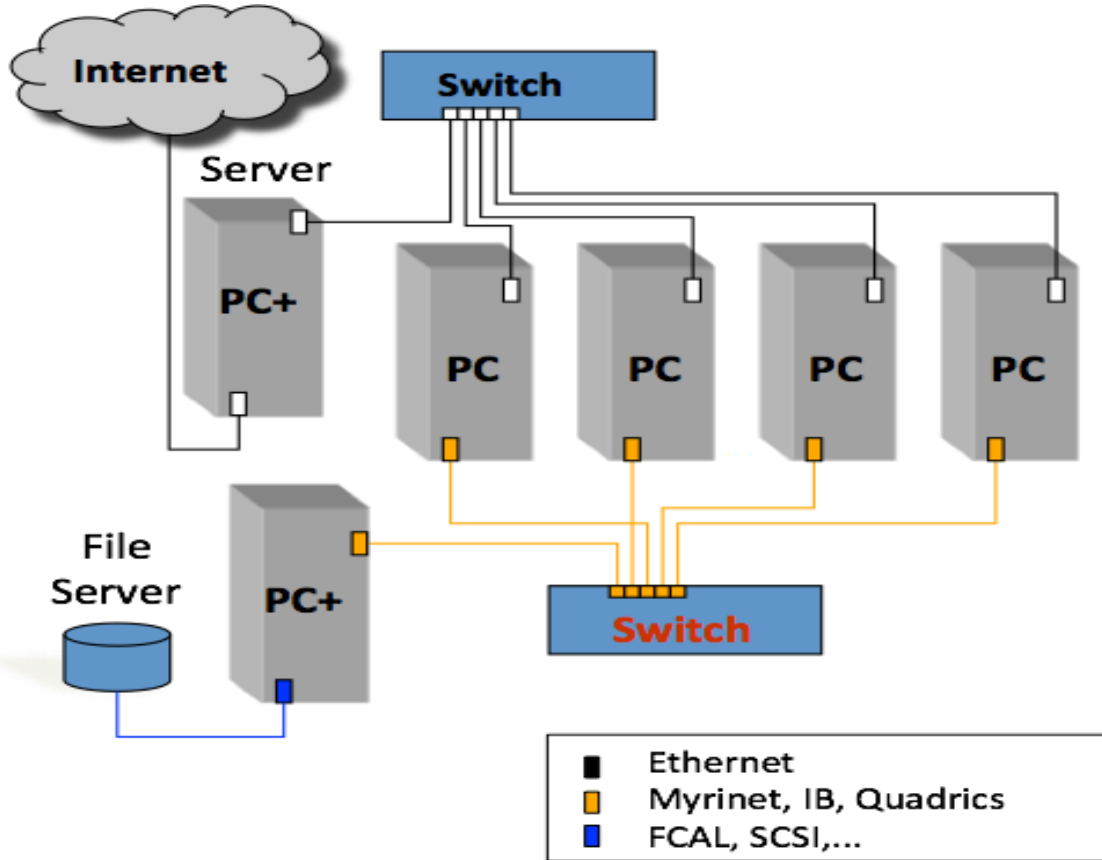
What is Orchestra?



- Wiki page: <https://wiki.med.harvard.edu/Orchestra>
- Tech spec:
 - Over 550 compute nodes
 - Over 8200 cores
 - 10GigE interconnection
 - Over 40TB RAM
- CentOS 6 Linux
- LSF scheduler
- Total 30+PB storage



Generic Cluster Architecture



- Login nodes are for data management, job submission, code development, etc.
- Compute nodes are for production runs
- Your storage space is centralized network storage system
- Hundreds users are sharing the resources, so please be nice each other.
- Job scheduler works based on complex algorithms of fair-share, priority management, load balancing, etc.

Orchestra Compute Nodes



Clarinet
4500 cores
12 x 96GB



Piccolo
800 cores
20 x 120GB



Ocarina
256 cores
32 x 512GB



Flute
400 cores
20 x 160GB



Oboe
2000 cores
28 x 224GB



Saxophone
24 cores
12 X 1TB

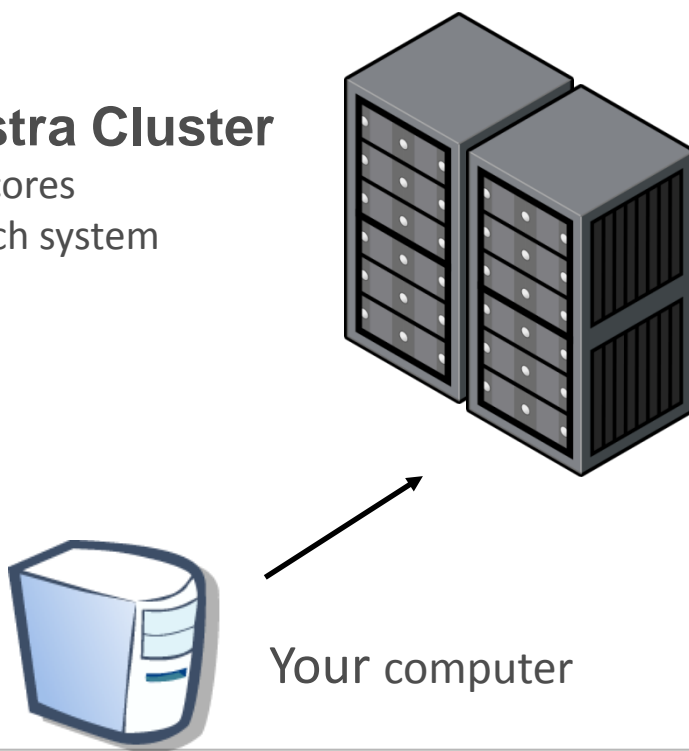
Orchestra Compute Nodes

NODE CLASS	CPU TYPE	Number of Nodes	Cores x Node	MEMORY x Node (~GB)	Total Cores
Clarinet	XeonE5530	21	8	vary	168
Clarinet	XeonL5640	294	12	96 (vary)	3528
Fife	XeonE52680	16	20	190	320
Piccolo	XeonE5266v2	40	20	128	800
Ocarina	Opteron6376	8	32	512	512
Ottavino	XeonE52697	16	28	252	448
Sax	XeonE74807	2	16	1024	32

Orchestra Primary Storage

Orchestra Cluster

- 7500+ cores
- LSF batch system



/home

- [/home/user_id](#)
- quota: 100GB per user
- Backup: extra copy & snapshots:
 - daily to 14 days, weekly up to 60 days

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

- [/n/data1/institution/dept/lab/your_dir](#)
- quota: expandable
- Backup: extra copy & snapshots:
 - daily to 14 days, weekly up to 60 days

Temporary “Scratch” storage



- **/n/scratch2**
 - **For data only needed temporarily during analyses.**
 - **Each account can use up to 10 TB and 1 million files/directories**
-
- **Lustre** --> a high-performance parallel file system running on DDN Storage.
 - More than 1 PB of total shared disk space.
 - No backups! Files are automatically deleted after unaccessed for 30 days, to save space.
 - More info at: <https://wiki.med.harvard.edu/Orchestra/LustreScratch>



“No backup” GPFS storage

- **/n/nobackup**
 - **used for large data sets not requiring backups**
 - **limited to only a few labs with large data requirements**
-
- **GPFS** --> a high-performance clustered file developed by IBM.
 - 2 PB of disk space.
 - No backups, but files are not purged on a schedule like “scratch” storage.
 - More info at: <https://wiki.med.harvard.edu/Orchestra/NoBackupGPFS>

Checking Storage Usage

- To check your storage available:

`mfk8@loge:~$ quota`

Home directory: you get 100 GB, total.

Group directories: space varies, can increase.

`/groups/groupname`

`/n/data1`

`/n/data2`

Checking Storage Usage: scratch2

- `mfk8@loge:~$ 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

Storage Policies

- /home: 14 day snapshots + 60 day full backup
- /groups, /n/data1, /n/data2: 14 day snapshots + 60 day full backup
- /n/scratch2: 30 day retention, no backups
- New tape system for Long Term Storage

Snapshots - Isilon

- Snapshots (static) are retained for up to 60 days: recover data
- `mfk8@clarinet001:~$ cd .snapshot`
- `mfk8@clarinet001:~$ ls`

Orchestra_home_daily_2015-10-02-02-00

Orchestra_home_daily_2015-10-01-02-00

- `mfk8@clarinet001:~$ cd Orchestra_home_daily_2015-10-02-02-00`
- `mfk8@clarinet001:~$ cp MyRetreivedFile ~`

Create an Orchestra Account

- <http://rc.hms.harvard.edu/#orchestra>

Click the **red button** and fill out the form!

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

Account Request

LSF: Fair Sharing

- **Load Sharing Facility:** distributes jobs across Orchestra fairly
- Ensures that no one user or core monopolizes Orchestra
- Users are assigned dynamic priorities
- Queues also have priorities
- Submitting lots of jobs reduces your fairshare priority
- Even if many jobs are pending, your jobs will start quickly provided you have not submitted many jobs



Logging Into Orchestra: Mac

- Open a terminal (search “terminal”)
`ssh username@orchestra.med.harvard.edu`
- To display graphics back to your desktop (X11 forwarding)
Install XQuartz (google it) and have it running
`ssh -X username@orchestra.med.harvard.edu`

Logging Into Orchestra: Windows



- Install PuTTY (google it)
- In box under “Host Name (or IP address)”
orchestra.med.harvard.edu
- To display graphics back to your desktop (X11 Forwarding):
 1. Install Xming (google it) and have it running
 2. In PuTTY, under Connection -> SSH -> X11:
 3. Check “Enable X11 Forwarding” first and THEN log in

Logging Into Orchestra: Linux



- Open a terminal (search: “terminal”)

```
ssh username@orchestra.med.harvard.edu
```

For graphics (X11 Forwarding)

```
ssh -X username@orchestra.med.harvard.edu
```

Welcome to Orchestra!

- Where are you in Orchestra?

See your terminal!

mfk8@mezzanine: ~\$

- You are logged into a “shell login server.” Usually “mezzanine” or “loge.” These are not meant for heavy lifting!
- You are in your home directory. This is symbolized by the “tilde ” (~). This is shorthand for: /home/username
- You are in a bash environment. “\$” means “ready to accept your commands!”

Interactive Sessions

- The login servers are not designed to handle intensive processes. Using 40% of a CPU results in a process being terminated. It is sometimes better to use an “interactive session.” These last up to 12 hours. Start by entering your first job! This will (usually) log you into a “clarinet!”

```
mfk8@mezzanine:~$ bsub -Is -q interactive bash
```

“bsub” is how jobs are submitted to Orchestra LSF

-Is (capital eye, s) tells Orchestra you want an interactive shell

-q is the “queue” to enter (interactive)

“bash” opens the bash session

```
mfk8@clarinet001:~/$
```

Listing a Folder's Contents

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

```
mfk8@clarinet001:~$ ls
```

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

```
mfk8@clarinet001:~$ ls -l
```

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

```
mfk8@clarinet001:~$ ls /groups/rc-training/introthpc
```

Viewing File Contents

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

```
mfk8@clarinet001:~$ less ~/.forward
```

Making a Folder (Directory)

- “mkdir” stands for “**make directory.**”
- Create a new directory for this exercise
- Spaces are discouraged. (Underscores are fine!) Case counts in Linux.

```
mfk8@clarinet001:~$ 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@clarinet001:~$ cd MyTestDir
```

Notice how the prompt tells you where you are!

```
mfk8@clarinet001:~/MyTestDir$ cd ..
```

```
mfk8@clarinet001:~$
```

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@clarinet001:~$ nano myfile.txt
```

```
This is my new file text.
```

(Control-X to save (yes) and exit.)

```
mfk8@clarinet001:~$
```

```
mfk8@clarinet001:~$ 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@clarinet001:~$ 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@clarinet001:~$ cp myfile.txt mycopy2.txt #copying and renaming
```

Moving Data

- “move” “from” “to”

```
mfk8@clarinet001~:$ mv MyTestDir/myfilecopy.txt ~
```

```
mfk8@clarinet001~:$ mv MyTestDir/ MyTestDir2/
```

Removing Files/Folders

- “rm” to remove a file

```
mfk8@clarinet001:~$ rm myfile.txt
```

- “rm -r” to remove a folder recursively

```
mfk8@clarinet001:~$ rm -r MyTest2
```

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 Orchestra



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

transfer.orchestra.med.harvard.edu

your username and password

port 22

Using Software: Environment Modules

- Most “software” on Orchestra is installed as an environment module. Allows for clean, easy loading, including most dependencies, and switching versions.

\$ module avail

\$ module avail seq/

\$ module avail stats/

\$ module avail seq/bowtie/

Loading/Unloading Modules

- Loading modules
`$ module load seq/bowtie/2.0.6`
- Which module version is loaded (if at all)?
`$ which bowtie`
- See all modules loaded
`$ module list`
- Unloading modules
`$ module unload seq/bowtie/2.0.6`
- Dump all modules
`$ module purge`

Compiling your own software

- Users can compile software in their /home or /groups directories, where they have permission
- Binaries just require “unzipping” (ie tar -zxvf .tgz)
- Common compiling libraries are found as modules:

dev/compiler/gcc-4.8.5

dev/boost/1.57.0

dev/openblas/0.2.14

dev/lapack

- Common libraries are found in “utils”

Installing Software: Binary Example

- `mfk8@loge:~$ bsub -Is -q interactive bash`
- `mfk8@clarinet001:~$ wget http://path/to/binary/mysoftware.tar.gz`
- `mfk8@clarinet001:~$ tar -zxvf mysoftware.tar.gz`
- `mfk8@clarinet001:~$ ls mysoftware/bin`

Installing Software: Source

- `mfk8@loge:~$ bsub -ls -q interactive bash`
- `mfk8@clarinet001:~$ module load dev/compiler/gcc-4.8.5`
- `mfk8@clarinet001:~$ wget http://path/to/source/mysoftware.tar.gz`
- `mfk8@clarinet001:~$ tar -zxvf mysoftware.tar.gz`
- `mfk8@clarinet001:~$ cd mysoftware`
- `mfk8@clarinet001:~$ less README`
- `mfk8@clarinet001:~$./configure --prefix=/home/mfk8/software`
- `mfk8@clarinet001:~$ make`
- `mfk8@clarinet001:~$ make install`



Python, R, Perl

- Users manage their own package libraries: get the version you want, when you want it!
- **Python:** virtual environment allows pip/easy installs
<https://wiki.med.harvard.edu/Orchestra/PersonalPythonPackages>
- **R:** set up personal R library for cran, Bioconductor
<https://wiki.med.harvard.edu/Orchestra/PersonalRPackages>
- **Perl:** local::lib allows cpan, cpanm
<https://wiki.med.harvard.edu/Orchestra/PersonalPerlPackages>
- **Shebangs:** `#!/usr/bin/env python/Rscript/perl`

MATLAB on Orchestra



- Free to use, with 2000 licenses
- Graphical interface (GUI) available, but lag over X11
- Compiling code does not improve performance
- Parallel Computing Toolbox: look for future seminars!
- MathWorks specialists will hold Office Hours this semester

Submitting Jobs

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

```
mfk8@clarinet001:~$ bowtie -p 4 hg19 file1_1.fq file1_2.fq
```

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

```
mfk8@clarinet001:~$ bsub < mybowtiejob.sh
```

- Orchestra will notify you when the job is done, or if there is an error.

The “bsub”

```
mfk8@clarinet001:~$ bsub -q queue -W hr:min job
```

- Necessary:
 - q (queue)
 - W (runtime in hr:min)

Shared Queues

- *mpi* queue if you have an MPI parallel job
- *priority* queue if you have just one or two jobs to run
- *mcore* queue if you have multi-core jobs to run.
- *short* queue if your jobs will take less than 12 hours to run.
- *medium* queue if your jobs take between 12 hours and 5 days
- Else: *long* queue.

Shared Queues Breakdown

Queue Name	Priority	Max Cores	Max Runtime
interactive	12	12	12 hours
priority	14	12	1 month
mcore	12	20	1 month
mpi	12	400	1 month
parallel	10	400	1 month
short	8	12	12 hours
medium	7	12	5 days
long	6	12	1 month
mini	5	1	10 minutes

Runtime Limit

- -W in hours:minutes
- Runtimes are subject to the maximum time permitted per queue (see table)
- If your job exceeds your runtime, your job will be killed ☹️
- Running many jobs that finish quickly (less than a few minutes) is suboptimal and may result in job suspension, contact RC to learn how to batch jobs

CPU Limit

- Amount of seconds the cluster works on your job (calculated by LSF)
- Ncores * Runlimit (-n * -W)
- Common error:

```
bsub -q short -W 8:00 tophat -p 8
```

tophat asks for 8 cores but only 1 requested (no -n), job killed in 1 hour

Multithreading

- A single CPU can execute multiple processes (threads) concurrently
- -n indicates how many cores are requested
- Jobs that are overefficient (use more cores than reserved) jeopardize the health of a node
- Reserve the same amount of cores in your job and your bsub!

Reserving Memory

- Most nodes have 90GB memory available over all cores, some have more
- Make a resource request with
 - R “rusage[mem=16000]” (memory requested in MB)
- Memory multiplies by cores requested, so
 - n 4 –R “rusage[mem=16000]” reserves 64GB memory
- Asking for more memory may cause jobs to pend longer
- TERM_MEMLIMIT errors indicate that not enough memory was reserved

MPI

- Efficient way to run parallel jobs
- Dedicated queue with same-type compute nodes for optimal performance (400 cores)
- Matlab, Python, Java, R, C++, Fortran have MPI options
- Orchestra implementation: openMPI-1.8.6
- Talk to us first!

GPU

- Graphical Processing Unit
- Image processing, deep learning, and more!
- Experimental, limited queue to gauge the interest and needs of the research community
- NVIDIA accelerator cards mounted on compute
- 5 x K20 (2496 CUDA cores x 5GB GDDR5)
- 16 x K80 (4992 CUDA cores x 24GB GDDR5)
- 8 x M40 (3072 CUDA cores x 12GB GDDR5)
- Send us feedback on workflows and requirements!



Other bsub options

- n 4 (*number of cores requested*)
- R "rusage[mem=16000]" (*memory requested in MB*)
- J jobname
- a openmpi (type of mpi run)
- ls (interactive shell)
- e errfile (*send errors to file*)
- o outfile (*send screen output to file*)
- N (notify when job completes)

Monitoring Jobs

- List info about jobs/their status:

`mfk8$loge:~$ bjobs`

`-r` (running jobs)

`-p` (pending jobs)

`-l` (command entered, long form)

- List historical job information

`mfk8$loge:~$ bhist jobid`

Job Suspensions

- Jobs in the “long” queue can be suspended by LSF temporarily to allow jobs from the “short” queue (less than 12h) to run.
- If a job is suspended for >50% of the `-W` runlimit cumulatively, it will be automatically moved to the “no_suspend” queue where it can’t be suspended anymore.

Terminating Jobs

- Terminate a job (jobid given at submission)

`mfk8$loge:~$ bkill jobid`

- Terminate multiple jobs

`mfk8$loge:~$ bkill jobid1 jobid2 jobid3`

- Terminate all of your jobs (be careful!)

`mfk8$loge:~$ bkill 0`

Shell Scripts: The Basics

```
#!/bin/sh #always at the top
```

```
#program with options
```

```
module load seq/tophat/2.1.0 seq/bowtie/2.2.4 seq/samtools/1.2
```

```
tophat -p 4 -o ./mytophatdir1 hg19 file1_1.fastq file1_2.fastq
```

```
tophat -p 4 -o ./mytophatdir2 hg19 file2_1.fastq file2_2.fastq
```

```
#save as myshellscript.lsf
```

Calling a Shell Script

```
mfk8@balcony:~$ bsub -q mcore -W 12:00 -n 4 -e %J.err -o  
%J.out -N sh myshellscript.lsf
```

Creating a Complete Shell Script

```
#!/bin/sh
#BSUB -q mcore
#BSUB -W 12:00
#BSUB -o %J.out
#BSUB -e %J.err
#BSUB -N
#BSUB -n 4
#BSUB -R "rusage[mem=12000]"
module load seq/tophat/2.1.0 seq/bowtie/2.2.4 seq/samtools/1.2
tophat -p 4 -o ./mytophatdir1 hg19 file1_1.fastq file1_2.fastq
#save as myshellscript2.lsf
```

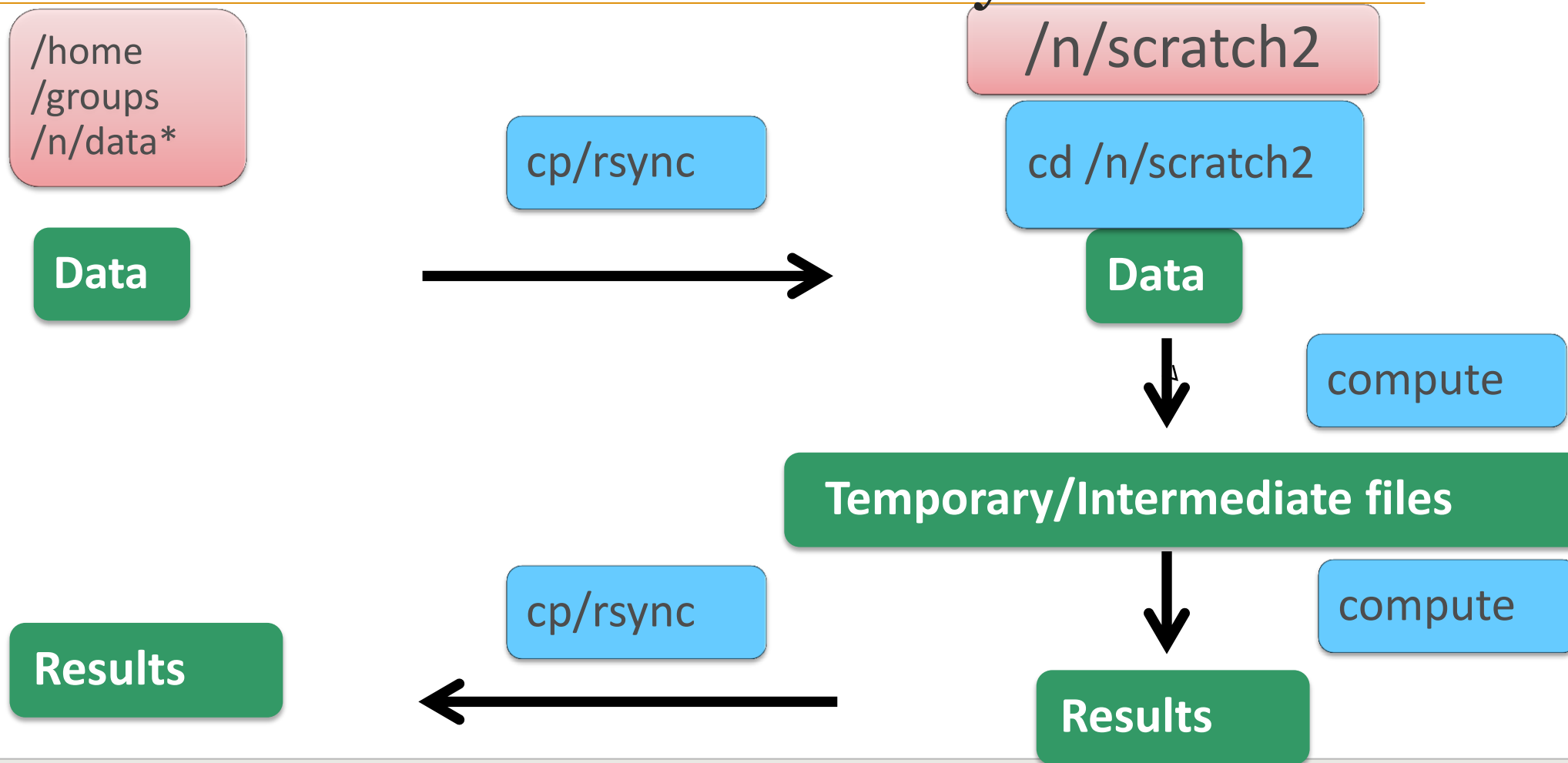
Calling a Complete Shell Script

```
mfk8@balcony:~$ bsub < myshellscript2.lsf
```

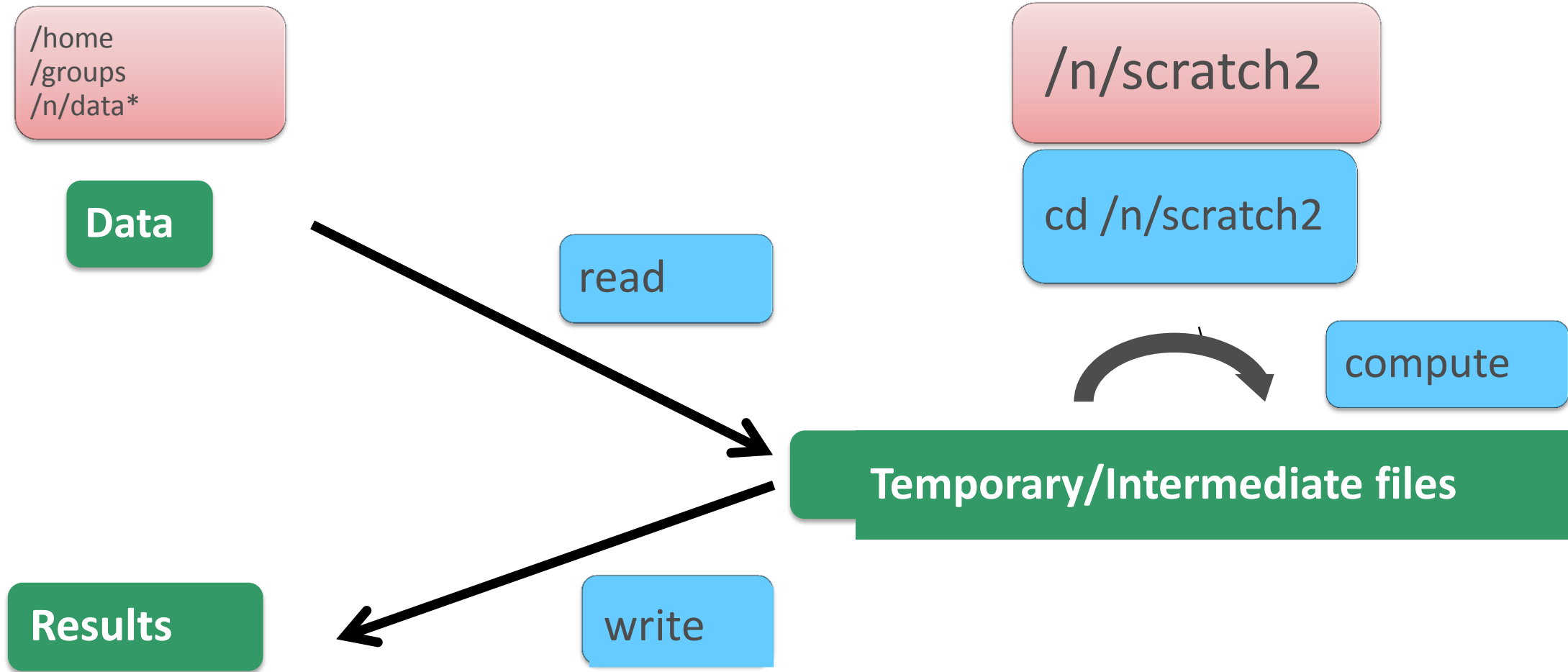
Using /n/scratch2

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

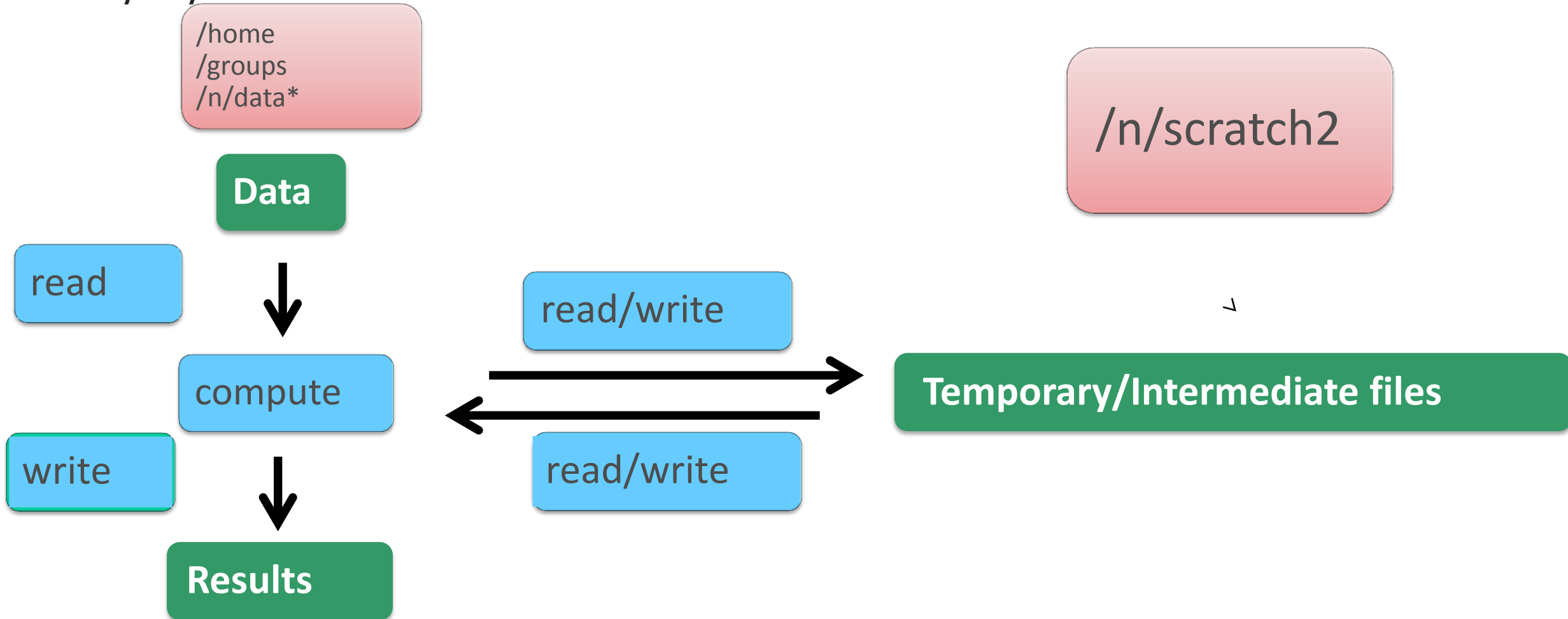
/n/scratch2 Workflow: Redundancy



/n/scratch2 Workflow: Medium Flexibility



/n/scratch2 Workflow: Best Practice



File Properties



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

chmod options file/directory

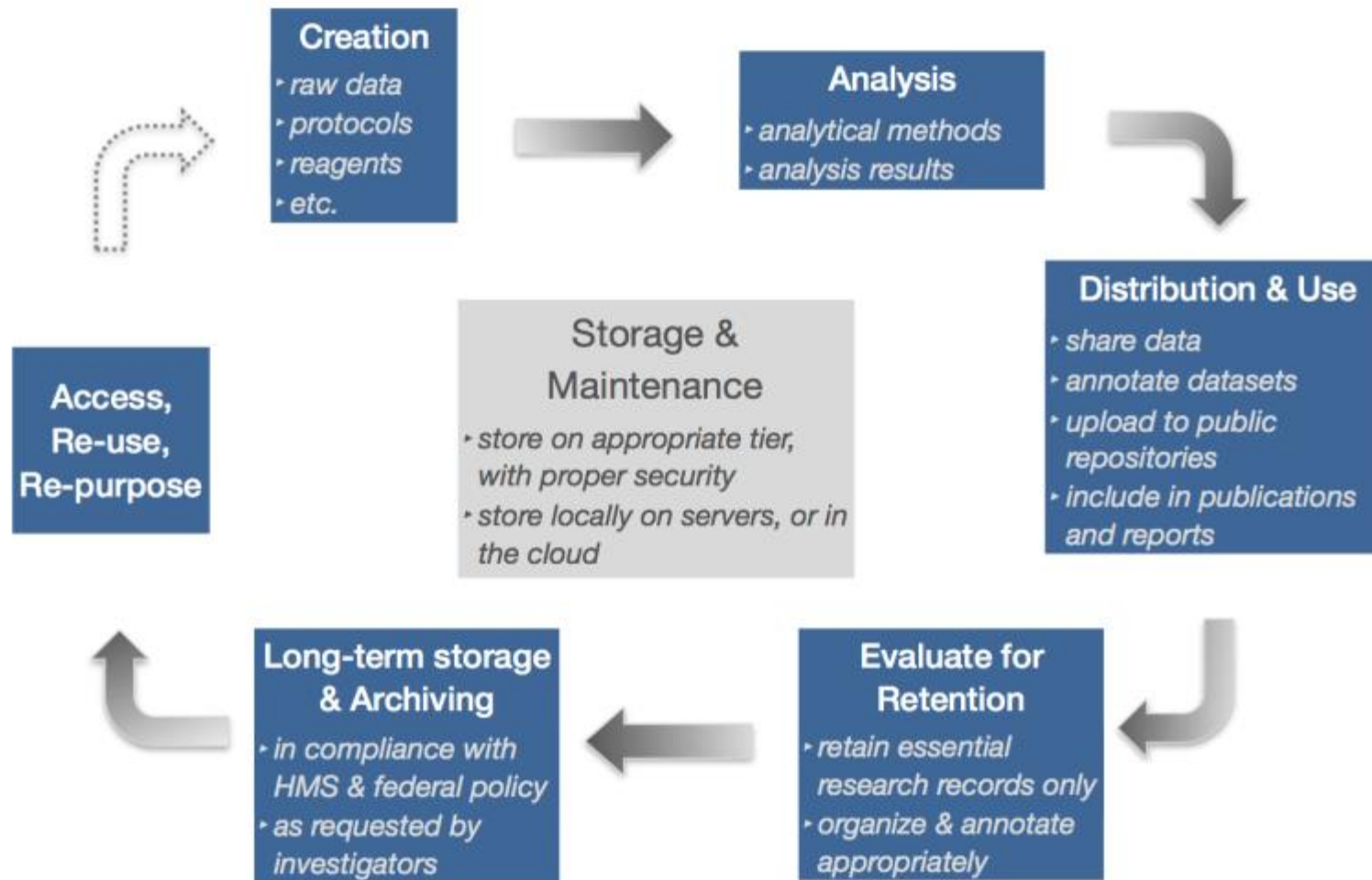
Who? **u**ser **g**roup **o**thers **a**ll (u/g/o/a)

What? **r**ead **w**rite **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

Data lifecycle for biomedical research



(HMS Data Management Working Group)

Data Management Planning

- When starting new projects, spend time thinking about how you will organize your data and metadata.
- What should the directory structure look like?
- Consider data type and file sizes, and total amount of data collected.
- Raw data:
 - Raw data should not be modified after data are collected.
 - Determine where raw data should be stored.
 - As data are annotated and analyzed, the resulting derived data files should be saved separately.
 - Consider identifying and using a read-only repository for raw data.

Thinking About Metadata

- Ask yourself:
 - What does metadata for this project look like?
 - What information should I keep track of?
 - Would a new project member be able to follow how data are created, stored, and documented?
- Create a readme.txt and store with each distinct dataset
 - Explain file naming conventions, abbreviations, codes, etc
 - Save as a plain text file
 - Avoid proprietary formats (e.g., Microsoft Word)

Example Readme File

Dataset title: Raw Images for Experiment A, Smith Lab

Principal Investigator: John Smith, PI, 555-555-5555, jsmith@hms.harvard.edu

Filename structure:

Structure:

ExperimentName_InstrumentID_CaptureDateTime_ImageID.tif

The base filename is composed of the name of the experiment, the ID number of the

instrument used, the date and time that the image was captured, and the unique identifier of the image.

Attributes:

ExperimentName = Name of the experiment.

Instrument ID = Five-digit code assigned to the lab instrument. See the Codes section for a list of instruments and their ID numbers.

CaptureDateTime = Date and time at which the image was captured, in YYYYMMDDThhmm format.

Image ID = Three-digit unique identifier for image, such as 001, 002, 003.

Codes:

[List of instruments and IDs]

Examples:

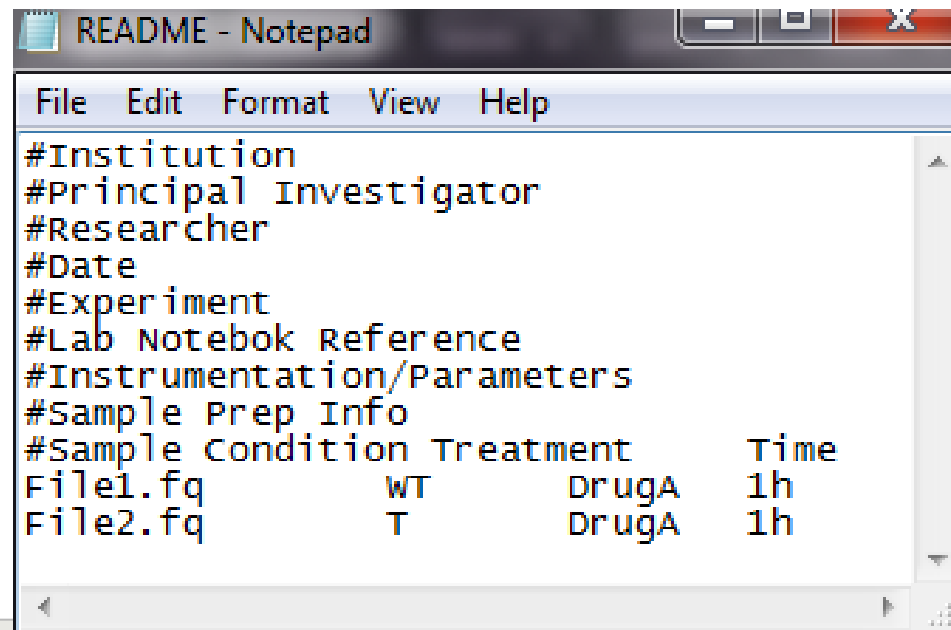
daf2-age1_14052_20150412T0515_005.tif

File formats: tif

Versioning: All changes to this dataset will be documented in a changelog in this readme document.

Example Readme File

- Create in text editor (Notepad, TextEdit, vi, emacs, nano)
- Start metadata information with comment (hashtag)
- Save as tab-delimited .txt file (usually README.txt)



```
File Edit Format View Help
#Institution
#Principal Investigator
#Researcher
#Date
#Experiment
#Lab Notebook Reference
#Instrumentation/Parameters
#Sample Prep Info
#Sample Condition Treatment Time
File1.fq WT DrugA 1h
File2.fq T DrugA 1h
```


Naming Convention Best Practices Files (1 of 2)

- Use naming conventions consistently.
- Should be descriptive and provide contextual information.
- File Name Length:
 - Don't make the name too long
 - Aim for 40-50 characters
 - Operating systems have different limits to the number of characters

Naming Convention Best Practices Files (2 of 2)

- Consider including a combination of the following:
 - Project or experiment name or acronym
 - Lab name/location
 - Researcher name/initials
 - Date or date range of experiment
 - Reference to lab notebook record
 - Type of data
 - Experiment conditions
 - Version number of file

Naming Convention Best Practices DOs

- DOs
 - Dates:
 - YYYYMMDD (e.g., 20160907)
 - Times:
 - use 24-hour military time to avoid confusion over a.m./p.m. (e.g., 1623 for 4:23 pm)
 - Sequential numbering:
 - use leading zeros (e.g., 001, 002, ... 010, 011, ... 100, 101, etc.)
 - Names:
 - surname then given (e.g., Smith_Bob)
 - Versioning:
 - use numbers to indicate updated versions (v1, v2)

Naming Convention Practice DON'Ts

- DON'Ts
 - Avoid special characters, such as
- ~ ! @ # \$ % ^ & * () ` ; : < > ? . , [] { } ' " |
 - Do not use carriage returns.
- Future-proof files: may need to be ported to a Linux/HPC environment later
- Special characters are often used for specific tasks in different operation systems.
- Commas are problematic when using comma separated values (csv) file format.

Naming Convention Practice DON'Ts Part II

- DON'Ts
 - Do not use spaces. Instead, try:
 - Underscores (e.g., file_name.xxx)
 - Dashes (e.g., file-name.xxx)
 - No separation (e.g., filename.xxx)
 - Camel case* (e.g., FileName.xxx)
- Some operating systems are case sensitive
- Some software will not recognize file names with spaces.
- File names with spaces must be enclosed in quotes when using the command line.

Naming Convention Best Practices Examples

-

Example files with no naming conventions:

- Test data 2016.xlsx
- Final FINAL! last version.docx

-

Example files with naming conventions:

- 20160104_ProjectA_Ex1Test1_SmithE_v1.xlsx
- 20160104_ProjectA_MeetingNotes_SmithE_v.1.docx

General best practices

- Be careful with Excel!
 - It introduces characters that are not recognized by UNIX
 - It modifies gene names into dates, e.g. SEPT2 ~ September 2
 - Be careful when sorting, make sure all columns are included
 - Be careful with numeric data, and any unexpected increments from one row to the next
- - Don't modify raw data
 - Raw data is sacrosanct!
 - Raw data can be anything you start your analysis with, e.g. fastq files, files from collaborators, data
 - Keep it as it is, if you want to change the names, make a symbolic link with the new name.

Long Term Storage

- HMS IT offers a Long-Term Storage service
- For storing large quantities of **infrequently** accessed data that do not need to be retrieved immediately
- Ideal for:
 - Completed or published projects
- For more information about the service, please visit:

<http://rits.hms.harvard.edu/dm/lts>

- Send questions to the Research Data Manager:

rdm@hms.harvard.edu

Publishing and Data Retention (1 of 2)

- Adhere to your lab's standard practices for data management and organization.
 - If you do not have standards, make them, write them down and follow them.
- Keep your data for **at least** seven (7) years.
 - IP (Intellectual Property), human subjects information and other factors can influence (extend) retention timelines, so before you delete data, check with your lab and sponsor guidelines first. If you have IP, talk to the Office of Technology Department (OTD).
- Store your data on University premises and/or systems.
 - If you are not sure what options are available to you, contact HUIT.

Publishing and Data Retention (2 of 2)

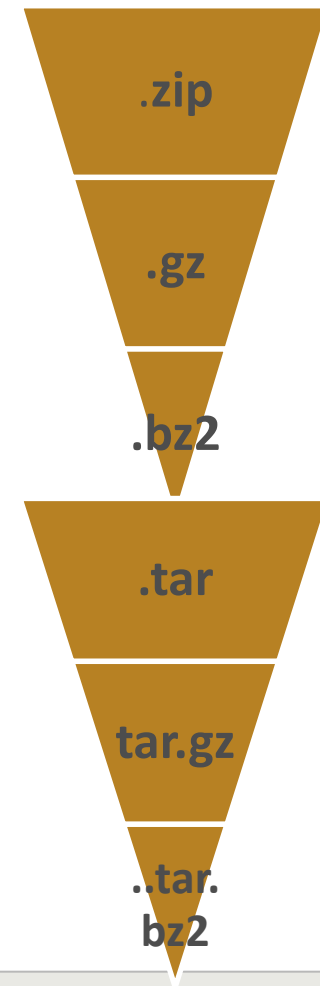
- If you leave HMS, and would like a copy of your data, please discuss this with your PI. HMS must retain the original data in order to meet its obligations to sponsors and the federal government.
- Remember: Harvard administration is here to help you organize and manage your materials. Stay in compliance and reduce administrative burden by calling HUIT or talking to your department administrator.
- HU's Retention and Maintenance of Research Records and Data: Principles and FAQs: http://osp.finance.harvard.edu/files/osp/files/research_records_and_data_retention_and_maintenance_guidance_2015.pdf

Data Deposition

- Certain NIH-funded research requires deposition of data into public repositories
- NIMH: dbGAP, GEO, NDAR
- Harvard DataVerse: Harvard Libraries, HUIT, IQSS
- Structural Biology Data Grid

Data Compression Methods

- zip: DEFLATE coding
- gzip: Lempel-Ziv coding (LZ77)
- bzip2: Burrows-Wheeler block sorting text and Huffman coding
- tar: archival utility preserving hierarchy and permissions, often used with gzip and bzip2



Scripting: Version Tracking

- Record changes (additions/deletions/replacements) to scripts
- Collaboration: many people can work on a file at once
- Helps with reverting to previous (working) versions
- Public or private repository options
- GitHub
- Bitbucket
- SVN
- Open Science Framework: integrate: Github/DropBox/Google Drive/AWS and more



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
- Orchestra: CLI environment module, Java desktop client, or web interface
- Upload data from research.files, /home, /groups, /n/data1, /n/data2
- imc-support@hms.harvard.edu
- <http://imc.hms.harvard.edu>

For more direction:

- <https://wiki.med.harvard.edu/Orchestra/NewUserGuide>
- <http://rc.hms.harvard.edu>
- rchelp@hms.harvard.edu
- Office Hours: Wednesdays 1-3p Gordon Hall 500
- Class survey:
<http://hmsrc.me/introhpc2017-survey1>