Data Management: Best Practices for Groups

Dr. David Hoover, HPC @ NIH

hooverdm@hpc.nih.gov

24 April 2019

NIH National Institutes of Health
Turning Discovery Into Health

BIOWULF
HIGH PERFORMANCE COMPUTING AT THE NIH

Outline

- Purpose and goals
- Hassles with shared data
- Fundamentals of file access permissions
- Modifying and maintaining file access permissions
- Organizing and tracking data
- Transferring data
- Group and data lifecycle

Purpose of Group and Shared Data

- Shared data processing using similar methods
- Software development and testing (version control?)
- Pooled storage for single lab or core (not archiving!)

Group

- group1 (user1, user2, user3)

Problem 1

- user2 and user3 can't write in new directories

```
[user3]$ cd /data/group1/project1
[user3]$ mkdir 190320_test
```

```
[user2]$ cd /data/group1/project1/190320_test
[user2]$ mkdir subdir
mkdir: cannot create directory 'subdir': Permission denied
[user2]$ ls -ld .
drwxr-x--- 1 user3  group1  1024 Mar 21 12:00 .
```

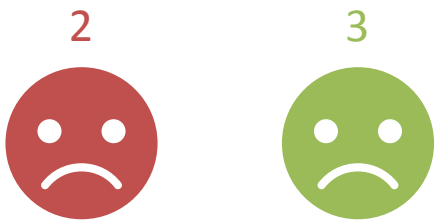
2

Problem 2

- Group members can (or can't) delete each other's files

```
[user2]$ touch /data/group1/project1/datafile.txt
-rw-rw---- 1 user2  group1  1024 Mar 21 12:00 datafile.txt
```

```
[user3]$ rm /data/group1/project1/*
```

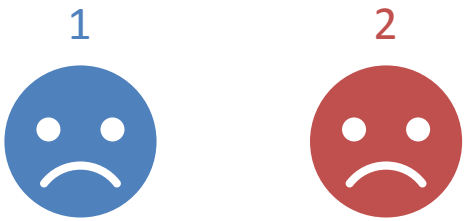


Problem 3

- Slurm job output not accessible by group

```
[user1]$ sbatch /data/group1/bin/shared_script.sh
12345678
```

```
[user2]$ ls -l /data/group1/project2/12345678_output
-rw------- 1 user2  user1  1024 Mar 21 12:00 logfile.out
-rw------- 1 user2  user1  1024 Mar 21 12:00 results.out
```



Problem 4

- Files are difficult to find
- Data becomes lost
- Traversing data is very slow

```
[user1]$ time ls /data/group1/working/output
real    23m5.003s
user    0m7.603s
sys     0m3.022s
[user1]$ ls /data/group1/working/output | wc -l
1500100
```

1

Problem 5

- There is not enough storage to hold all our data

```
[user3]$ ./run_script.sh
run_script.sh: write error: Disk quota exceeded
```

3

FUNDAMENTALS

UNIX Account

- A user account on a UNIX system has two numeric attributes, user id (uid) and group id (gid)

```
$ id user1
uid=1(user1) gid=1(user1) groups=1(user1)
```

- A user always belongs to their own group

UNIX Files

- A file on a UNIX system has three attributes: read, write, execute
- These attributes can be set for: owner, group, world

```
$ ls -l file
-rw-r----- 1 user1  user1  1024 Mar 21 12:00 file
```

- Attributes can be represented as a single number: **mode**

```
$ stat -c %a file
640
```

UNIX mask

- A user account on a UNIX system has a default **mask** value for the mode, as seen with the **umask** command:

```
$ umask
027
```

- This value acts as a bit filter when determining file mode

UNIX File Permissions

- The default permission mode of a file is 666
- The resulting permission on a created file depends on the **current mask** of the user
- The default mask value on HPC is set to 027
- To calculate mode of file, subtract 027 from 666:
- 666 – 027 = 640

UNIX Directory Permissions

- Directories have 777 as the default permission.
- 777 – 027 = 750

```
$ ls -ld dir
-rwxr-x--- 1 user1  user1  4096 Mar 21 12:00 dir
```

```
$ stat -c %a dir
750
```

UNIX Groups

- A UNIX group is a set of user accounts with a unique group name and group identification number (group and gid)
- The purpose of a UNIX group is to allow the members of the group to share files
- File sharing is done on the basis of permissions for the UNIX group of the file

UNIX Groups

- Users belong to their own ***primary*** group

```
$ groups user1
user1 : user1
$ groups user2
user2 : user2
$ groups user3
user3 : user3
```

UNIX Groups

- A new group can be created to hold multiple users (only by sysadmin)
- This is a ***secondary*** group

```
$ getent group group1
group1:x:2001:user1,user2,user3
```

UNIX Groups

- The ***primary*** group is listed before ***secondary*** groups

```
$ groups user1
user1 : user1 group1
$ groups user2
user2 : user2 group1
$ groups user3
user3 : user3 group1
```

- A user can belong to more than one secondary group

Sharing Files

- /home directories are strictly for individuals
- /scratch and /tmp are temporary
- /data directories can be used, but…

Shared Data Directories

- A shared data directory is created for the group

```
$ ls -ld /data/group1/
drwxrws--- 2 user1 group1 4096 Mar 21 12:00 /data/group1/
```

- user1 is the ***group owner***, who has management responsibility
- The PI has ultimate authority over the data

Group Sharing

- By default, members of a group still create files with their ***primary*** group

```
[user1]$ touch file1
-rw-r----- 1 user1  user1  1024 Mar 21 12:00 file1
```

```
[user2]$ touch file2
-rw-r----- 1 user2  user2  1024 Mar 21 12:00 file2
```

```
[user3]$ touch file3
-rw-r----- 1 user3  user3  1024 Mar 21 12:00 file3
```

Changing group

- Users can change the group of their own files and directories

```
[user1]$ ls -l file1
-rw-r----- 1 user1  user1  1024 Mar 21 12:00 file1
[user1]$ chgrp group1 file1
[user1]$ ls -l file1
-rw-r----- 1 user1  group1  1024 Mar 21 12:00 file1
```

- This can be pretty tedious

Shared Data Directories

- The /data directory has ***setgid*** set:

```
$ ls -ld /data/group1/
drwxrws--- 2 user1 group1 4096 Mar 21 12:00 /data/group1/
```

- This causes all new files and directories created to inherit the group

```
[user3]$ touch file
[user3]$ ls -l file
-rwxr----- 1 user3 group1    0 Mar 21 12:00 file
```

Shared Data Directories

- The setgid bit also propagates setgid

```
[user3]$ ls -ld /data/group1
drwxrws--- 2 user1 group1 4096 Mar 21 12:00 /data/group1/
[user3]$ mkdir dir
[user3]$ ls -ld dir
drwxr-s--- 1 user3 group1 4096 Mar 21 12:00 dir
[user3]$ mkdir dir/sub
[user3]$ ls -ld dir/sub
drwxr-s--- 1 user3 group1 4096 Mar 21 12:00 dir/sub
```

MAINTENANCE

Changing Permissions

- Permissions can be changed with ***chmod***:

```
$ ls -l file
-rw-r----- 1 user1  user1  1024 Mar 21 12:00 file
$ chmod 660 file
$ ls -l file
-rw-rw---- 1 user1  user1  1024 Mar 21 12:00 file
```

- Usually done with symbolic representation:

```
$ ls -ld dir
-rwxr-x--- 1 user1  user1  4096 Mar 21 12:00 dir
$ chmod g+w dir
$ ls -ld dir
-rwxrwx--- 1 user1  user1  4096 Mar 21 12:00 dir
```

Generic shared /data

```
$ tree -pugF /data/group1/
/data/group1
├── [drwxr-s--- user2 group1   ]  bin/
├── [drwxrws--- user1 group1   ]  project1/
├── [drwxrws--- user1 group1   ]  project2/
├── [drwxr-s--- user3 group1   ]  reference/
├── [drwxr-s--- user1 group1   ]  user1/
├── [drwxr-s--- user2 group1   ]  user2/
├── [drwxr-s--- user3 group1   ]  user3/
└── [drwxrws--- user1 group1   ]  working/
```

Problem 1: umask

- Custom mask

```
[user3]$ echo umask 007 >> ~/.bashrc
[user3]$ source ~/.bashrc
[user3]$ cd /data/group1/project1
[user3]$ mkdir 190320_test
[user3]$ ls -ld 190320_test
drwxrws--- 1 user3  group1  1024 Mar 21 12:00 190320_test
```

3

```
[user2]$ cd /data/group1/project1/190320_test
[user2]$ mkdir subdir
[user2]$ ls -ld subdir
drwxrws--- 1 user2  group1  1024 Mar 21 12:00 subdir
```

2

Problem 2: stickybit

- Allows group members to create and edit files, but not delete them

```
[user1]$ chmod +t project1
[user1]$ ls -ld project1
drwxrws--T 1 user1  group1  4096 Mar 21 12:00 project1
```

- Shorthand:

```
[user1]$ mkdir -m 3770 project1
```

- Sticky bit is **not** inherited or propagated

Problem 3: sbatch options

- Run job as group explicitly

```
[user1]$ sbatch --gid=group1 --partition=norm --time=10 \
  --gres=lscratch:10 submit.sh
12345678
...
[user1]$ ls -l
-rw-rw-r-- 1  user1  group1  728 Mar 20 12:00 slurm-
12345678.out
```

Problem 3: sbatch options

- Run job with explicit umask

```
[user2]$ SLURM_UMASK=007 sbatch --partition=norm \
  --gid=group1 --time=10 --gres=lscratch:10 submit.sh
87654321
...
[user2]$ ls -l
-rw-rw---- 1  user2  group1  728 Mar 20 12:00 slurm-
87654321.out
```

Set mask to 007

- Edit ~/.bashrc

```
[user2]$ cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

umask=007
```

Shared Data Directories and ACLs

- All new /data directories (GPFS) allow ***Access Control Lists*** (ACLs)
- ACLs allow for more fine-grained access controls

ACLs

- An ACL can be seen with getfacl

```
$ getfacl /data/group1/
# file: /data/group1/
# owner: user1
# group: group1
# flags: -s-
user::rwx
group::rwx
mask::rwx
other::---
```

Setting ACL with setfacl

```
[user1]$ cd /data/group1/
[user1]$ ls -ld project1
drwxr-s--- 2  user1  group1 4096 Mar 22 12:00 project1
[user1]$ setfacl -m u:user2:rwx project1
[user1]$ ls -ld project1
drwxrws---+ 2  user1  group1 4096 Mar 22 12:00 project1
[user1]$ getfacl project1
# file: project1/
# owner: user1
# group: group1
# flags: -s-
user::rwx
user:user2:rwx
group::r-x
mask::rwx
other::---
```

ACLs override group permissions

- Standard UNIX group permissions become the ACL mask
- The mask is the maximum access type for all members of the ACLs

Effect of chmod

```
[user1]$ ls -ld project1
drwxrws---+ 2  user1  group1 4096 Mar 22 12:00 project1
[user1]$ getfacl project1
user:user2:rwx
group::r-x
mask::rwx
[user1]$ chmod g-w project1
drwxr-s---+ 2  user1  group1 4096 Mar 22 12:00 project1
[user1]$ getfacl project1
user:user2:rwx          #effective:r-x
group::r-x
mask::r-x
```

Multiple ACs per file

```
[user1]$ setfacl -m u:user3:r-x project1
[user1]$ getfacl project1
# file: project1
# owner: user1
# group: group1
# flags: -s-
user::rwx
user:user2:rwx          #effective:r-x
user:user3:r-x
group::r-x
mask::r-x
other::---
```

Copying ACLs to new files

- Use --set-file=-

```
[user1]$ setfacl project1 | --set-file=- project2
[user1]$ getfacl project2
# file: project1
# owner: user1
# group: group1
# flags: -s-
user::rwx
user:user2:rwx
user:user3:r-x
group::rwx
mask::rwx
other::---
```

Propagate ACLs By Default

```
[user1]$ setfacl -m d:u:user2:rwx project2
[user1]$ getfacl project2
# file: project1
# owner: user1
# group: group1
# flags: -s-
user::rwx
user:user2:rwx
user:user3:r-x
group::rwx
mask::rwx
other::---
default:user::rwx
default:user:user2:rwx
default:group::rwx
default:mask::rwx
default:other::---
```

Propagate ACLs By Default

```
[user1]$ mkdir project2/190320_test1
[user1]$ getfacl project2/190320_test1
# file: project2/190320_test1
# owner: user1
# group: group1
# flags: -s-
user::rwx
user:user2:rwx
user:user3:r-x
group::rwx
mask::rwx
other::---
default:user::rwx
default:user:user2:rwx
default:group::rwx
default:mask::rwx
default:other::---
```

Default ACLs supersede umask

- Setting umask 077 should disallow all group access, but default ACLs win…

```
[user1]$ umask 077
[user1]$ mkdir project2/190320_test2
[user1]$ ls -l project2
drwxrws---+ 2  user1 group1 4096 Mar 22 15:02 190320_test1
drwxrws---+ 2  user1 group1 4096 Mar 22 15:02 190320_test2
[user1]$ touch user1/project1/190320_test2/file
[user1]$ ls -l user1/project1/190320_test2/file
-rw-rw---- 1  user1 group1 4096 Mar 22 15:02 file
```

Shared /data directory

```
$ tree -pugF /data/group1/
/data/group1
├── [drwxr-s---+ user2 group1   ]  bin/
├── [drwxrws---+ user1 group1   ]  project1/
├── [drwxrws---+ user1 group1   ]  project2/
├── [drwxr-s--- user3 group1    ]  reference/
├── [drwxr-s--- user1 group1    ]  user1/
├── [drwxr-s--- user2 group1    ]  user2/
├── [drwxr-s--- user3 group1    ]  user3/
└── [drwxrws--T user1 group1    ]  working/
```

Ongoing Maintenance

- Permissions can get messed up very easily
- Moving, rather than copying files
- Applications explicitly set permissions
- Slurm runs as primary group

```
[user1]$ ls -l project2
-rwxrwxrwx 1  user2 user2  4321 Mar 22 12:00 file.out
drwx------ 2  user1 user1 4096 Mar 22 15:00 results1
-rwx------ 1  user3 group1 12345 Mar 22 12:00 results1.txt
```

find … -exec chmod

- Bulldoze over permissions

```
[user1]$ find . -exec chmod g+rwx {} +
```

- More precision with conditional execute

```
[user1]$ find . -exec chmod g+rwX {} +
```

find … -exec setfacl

- find can't detect ACLs

```
[user1]$ setfacl -R –m g:group1:rwX,d:g:group1:rwX,o:--- .
```

Carefully find

- Running find on >1M files can take >10 minutes
- Automating is possible, but should be done on a cluster node
- Should be targeted not blind blanket

find … -exec chmod

- find and chmod

```
[user1]$ find . -perm /u+r -user user1 ! -perm /g+r \
  -exec chmod g+r {} +
[user1]$ find . -perm /u+w -user user1 ! -perm /g+w \
  -exec chmod g+w {} +
[user1]$ find . -perm /u+x -user user1 ! -perm /g+x \
  -exec chmod g+x {} +
```

- find and chgrp

```
[user1]$ find . -user user1 ! -group group1 \
  -exec chgrp group1 {} +
```

Targeted setfacl

- Redundant chmod updates ctime

```
[user1]$ stat file -c %z
2019-03-22 12:00:00.000000000 -0400
[user1]$ chmod g+rwx file
[user1]$ stat file -c %z
2019-03-25 09:00:00.000000000 -0400
[user1]$ chmod g+rwx file
[user1]$ stat file -c %z
2019-03-25 09:00:07.747292784 -0400
```

Targeted setfacl

- Redundant setfacl does not change ctime

```
[user1]$ stat file -c %z
2019-03-22 12:00:00.000000000 -0400
[user1]$ setfacl -m g:group1:rwX
[user1]$ stat file -c %z
2019-03-25 09:00:00.000000000 -0400
[user1]$ setfacl -m g:group1:rwX
[user1]$ stat file -c %z
2019-03-25 09:00:00.000000000 -0400
```

Targeted setfacl

- Combine find and setfacl

```
[user1]$ find . –ctime -1 –exec \
  setfacl –m g:group1:rwX,d:g:group1:rwX,o:--- {} +
```

- ctime changes when a file is modified or permissions are changed

Index projects

- Use updatedb and locate to index files

```
[user1]$ cd /data/group1
[user1]$ updatedb \
  --database-root project1 \
  --output project1.db --require-visibility no
```

- Searchable with locate

```
[user1]$ locate --database=project1.db this_file
/gpfs/gsfs11/users/group1/project1/dir/dir/this_file.txt
```

updatedb and locate

- >1M files can take ~2-3 minutes to index
- Searching is much, much faster than find
- Can pass results to grep

```
[user1]$ locate --database=project1.db / | grep this | \
  grep that | grep other
```

- Can only search by name, not by properties (size, mtime, permissions)

ORGANIZATION AND MONITORING

Setting Boundaries

- Many problems can be solved through structural design
- Design subdirectories for different purposes

Organize Files

- No more than 1000 files per directory
- File/directory names should be systematic and logical
- Store date, keywords in name

```
$ ls /data/group1/project2/C5/
190320_r1/      190320_r2/      190320_r3/      190320_r4/
190321_r1/      190321_r2/      190321_r3/      190321_r4/
README.txt      r1.bam          r1.bam.bai      r2.bam
r2.bam.bai      r3.bam          r3.bam.bai      r4.bam
r4.bam.bai      template.sh
```

Metadata

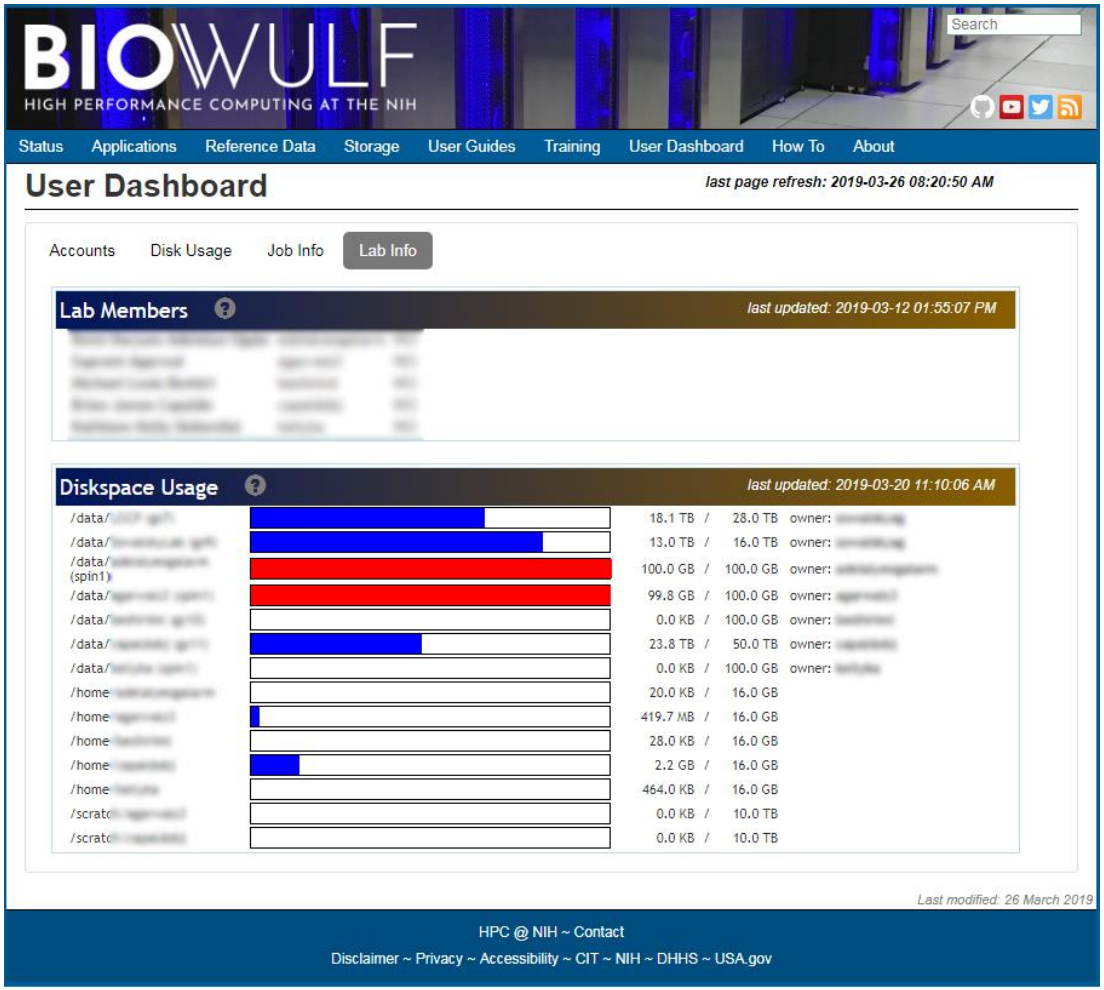
- Document data as it is added

```
$ cat /data/group1/project2/C5/190320_r1/README.txt
================================
2019-03-20, user2
Run 1, trial C5, project1
See submit_r1.sh for methodology and commands run
Used template.sh to generate submit_r1.sh
```

Segregate by expected lifetime

- How long do you expect the file to be needed?
- /lscratch if only for lifetime of job
- /data if for a few months
- /home if for longer

Dashboard


BIOWULF
HIGH PERFORMANCE COMPUTING AT THE NIH
Search
Status
Applications
Reference Data
Storage
User Guides
Training
User Dashboard
How To
About
User Dashboard
last page refresh: 2019-03-26 08:20:50 AM
Accounts
Disk Usage
Job Info
Lab Info
Lab Members
last updated: 2019-03-12 01:55:07 PM
Diskspace Usage
last updated: 2019-03-20 11:10:06 AM
/data/ 18.1 TB / 28.0 TB owner:
/data/ 13.0 TB / 16.0 TB owner:
/data/ (spin1) 100.0 GB / 100.0 GB owner:
/data/ 99.8 GB / 100.0 GB owner:
/data/ 0.0 KB / 100.0 GB owner:
/data/ 23.8 TB / 50.0 TB owner:
/data/ 0.0 KB / 100.0 GB owner:
/home 20.0 KB / 16.0 GB
/home 419.7 MB / 16.0 GB
/home 28.0 KB / 16.0 GB
/home 2.2 GB / 16.0 GB
/home 464.0 KB / 16.0 GB
/scratch 0.0 KB / 10.0 TB
/scratch 0.0 KB / 10.0 TB
Last modified: 26 March 2019
HPC @ NIH ~ Contact
Disclaimer ~ Privacy ~ Accessibility ~ CIT ~ NIH ~ DHHS ~ USA.gov

checkquota

```
$ checkquota
Mount                      Used      Quota  Percent    Files    Limit
/data:                417.8 GB   500.0 GB   83.56%   441553 31129581
/gs10(group1):        100.6 GB     5.0 TB    0.21%    40667 31457280
/home:                 12.1 GB    16.0 GB   75.62%   108094      n/a

ObjectStore Vaults
user1:                  5.7 MB   465.7 GB    0.00%      n/a      n/a
```

FILE TRANSFER AND ARCHIVING

Compression

- Use gzip to compress files

```
$ ls -l
-rw-r--r-- 1 user1  group1 641955988 Sep 22  2018 bogus_R1_001.fastq
-rw-r--r-- 1 user1  group1 643754231 Sep 22  2018 bogus_R2_001.fastq
$ gzip *.fastq
$ ls -l
-rw-r----- 1 user1  group1 134381309 Sep 22  2018 bogus_R1_001.fastq.gz
-rw-r----- 1 user1  group1 134873947 Sep 22  2018 bogus_R1_001.fastq.gz
```

Compression Algorithms

- 247 MB typical file on /spin1

Command	C time	Size	Ratio	D time
gzip	15.8 s	66 M	26.7%	2.2 s
bzip2	61.6 s	47 M	19.0%	10.8 s
xz	199.0 s	32 M	12.9%	3.7 s

- Additional compression settings (1..9), default is 6
- Compression depends on randomness of data

Compress Limitations

- Not all files can be compressed

Ext	File Type	gzip	xz -9
.bam	gzip compressed data, extra field	100%	100%
.tiff	TIFF image data, little-endian	99.2%	99.2%
.jpg	JPEG image data, EXIF standard	97.2%	97.2%
.png	PNG image data, …	97.7%	95.9%
.hdf	Hierarchical Data Format (v5)	92.7%	90.9%
.mrc	CCP4 Electron Density Map	92.8%	82.1%
.pdf	PDF document	100%	56.6%
.tif	TIFF image data, big-endian	7.3%	4.7%
.bed	ASCII text, with very long lines	9.2%	4.3%

Compress Projects

- As projects wrap up, compress files into tarball

```
[user1]$ cd /data/group1
[user1]$ tar czf project1.tgz project1
```

- Also searchable:

```
[user1]$ tar tzf project1.tgz this_file
project1/dir/dir/this_file.txt
```

Archiving Data

- HPC is NOT suitable for archival storage
- Regular backups are NOT done
- Users are encouraged to seek out alternative means of archiving their data

Push to Object Storage

- Short-term archiving

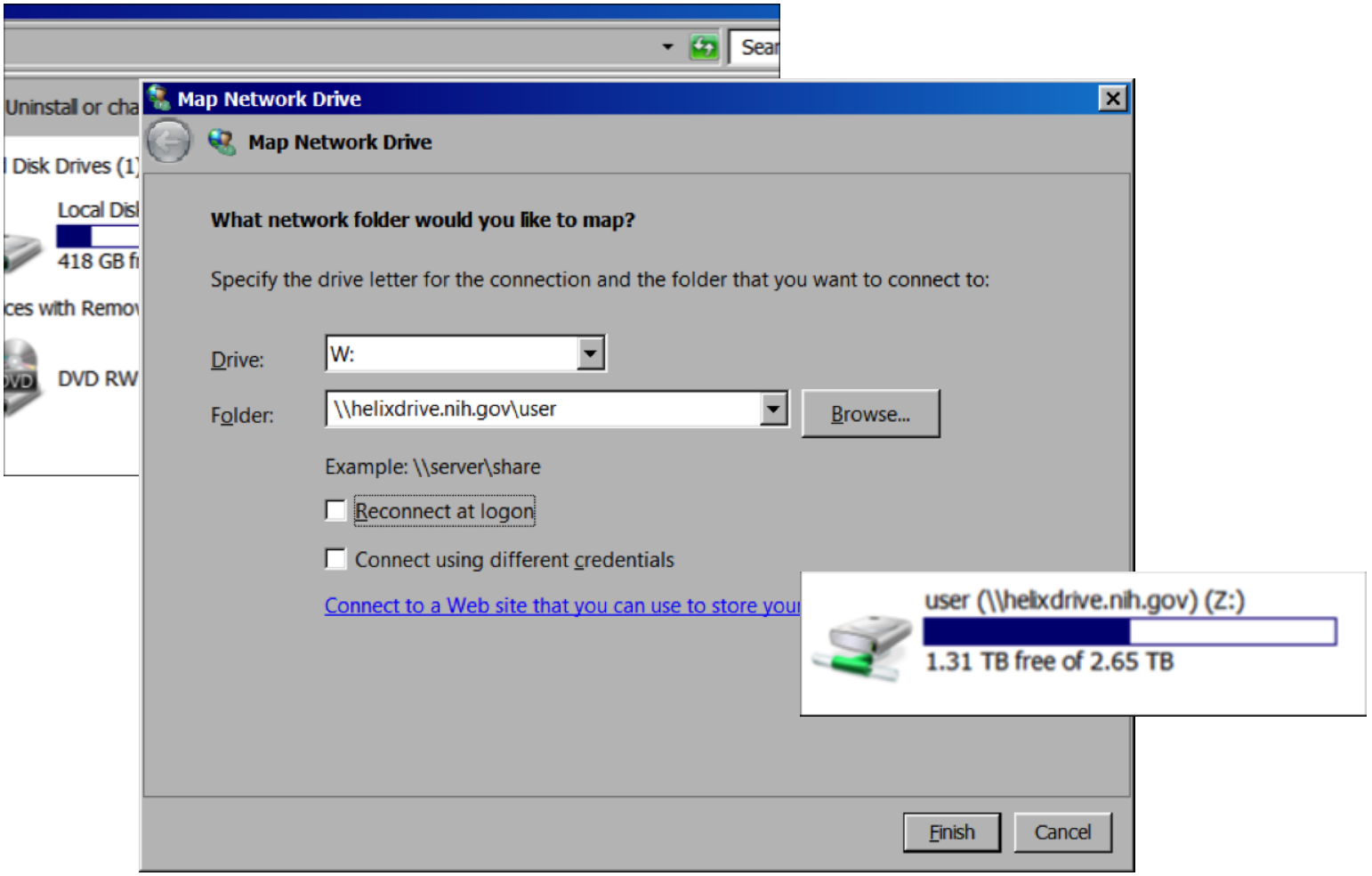
```
[user1]$ obj_put -v -p "group1/" project1.tgz
```

- https://hpc.nih.gov/storage/object.html
- Vaults can hold files for 7 years

Transfer: Helixdrive

- Allows local mount of exported file systems
- /home, /data, /scratch
- Available for Windows, Macs and Linux
- Speed: slow
- Reliability: moderate
- Ease: high

Transfer: Helixdrive



Transfer: Helixdrive

```
$ mount -t cifs -o
rw,vers=2.0,nosetuids,sec=ntlmsspi,user=user1,domain=NIH.g
ov //helixdrive.nih.gov/group1 /mnt/biowulf_group1
```

Transfer: rsync

- Can be run to/from helix.nih.gov
- Available for Mac, Linux
- There are rsync wrapper clients for Windows
- Speed: moderate
- Reliability: high
- Ease: high/low, depends on OS

SSH Keys

- Must have at least 2048 bit encryption
- Must have a secret passphrase
- Length beats complexity
- SSH keys do NOT override AD password expiration and shell locking
- https://hpc.nih.gov/docs/sshkeys.html

rsync Using SSH Keys

```
$ rsync -avz -e "ssh -i foo_rsa“ \
  user1@helix.nih.gov:/data/group1/project1 \
  /path/on/desktop/
```

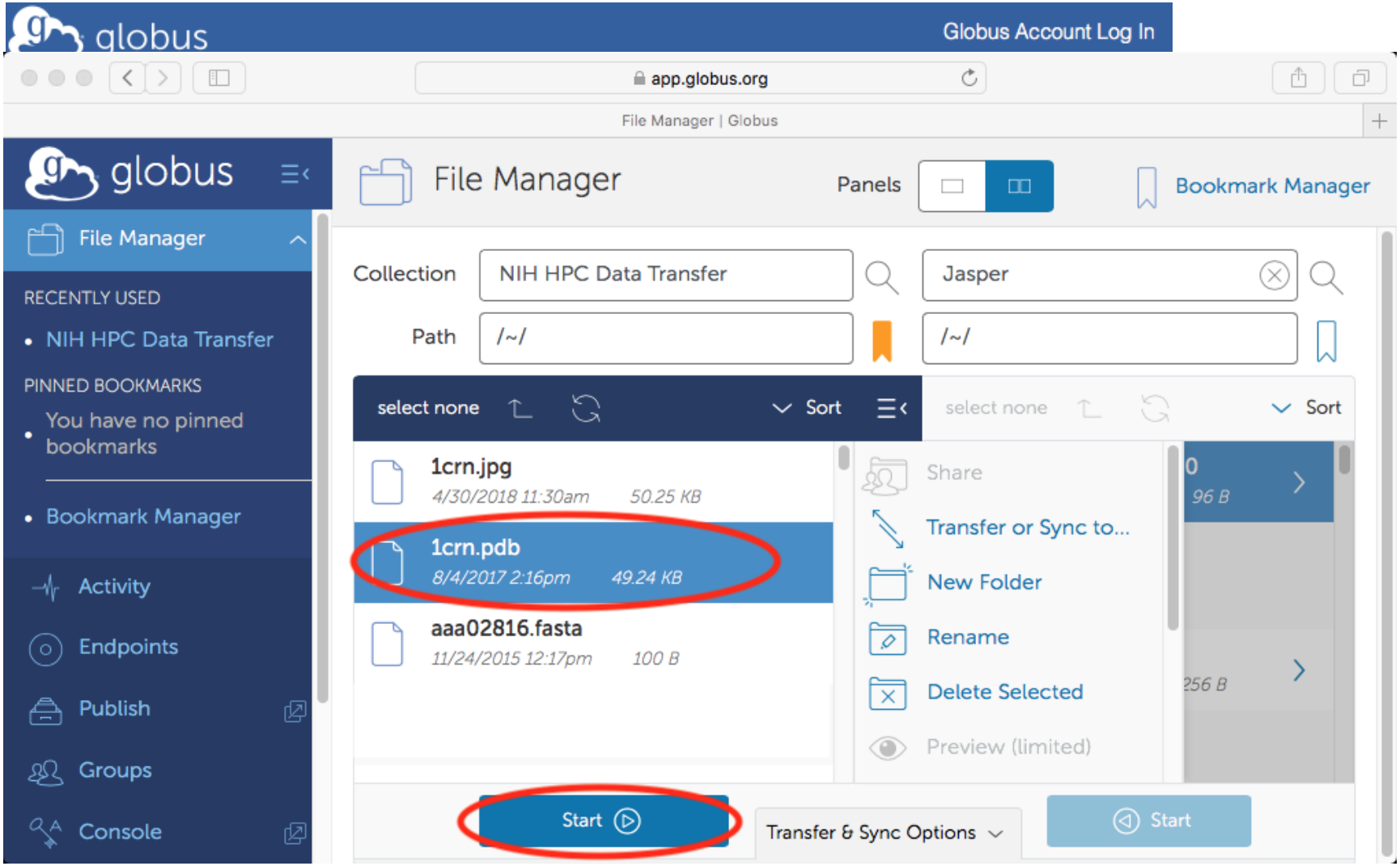
Key Storage Agents

- ssh-agent (Linux, Mac)
- pageant (Windows, with PuTTY)
- Stores private key and passphrase, allows single authentication, multiple connects

Transfer: Globus

- Service provided by University of Chicago
- Requires local client and port availability
- Available for Windows, Mac, Linux
- Speed: fast
- Reliability: high
- Ease: moderate

Transfer: Globus



Transfer: Globus CLI

- Takes some doing, but very worth it

```
[user1]$ globus transfer --recursive --no-verify-checksum \
    d8eb36b6-6d04-11e5-ba46-22000b92c6ec:/path/to/local/dir/ \
    e2620047-6d04-11e5-ba46-22000b92c6ec:/data/group1/project1/
Message: The transfer has been accepted and a task has been created and
queued for execution
Task ID: 6924b21e-f54b-11e6-ba69-22000b9a448b
```

- Can be incorporated into batch scripts

Transfer: Amazon & Google Cloud

- Cloud-based storage obtained outside of HPC
- Various clients available
- Must be run from helix.nih.gov
- Speed: ?
- Reliability: ?
- Ease: low

Transfer: Google Cloud

- https://cloud.google.com/storage/docs/gsutil

```
[user1@helix ~]$ module load google-cloud-sdk
[user1@helix ~]$ gcloud init
```

```
[user1@helix ~]$ gsutil -m cp gs://my_bucket/* /data/group1/project3/.
```

Transfer: Google Drive

- gdrive can automate transfers

```
[user1@helix ~]$ module load gdrive
[+] Loading gdrive 2.1.0 on helix.nih.gov
[user1@helix ~]$ gdrive download 1YQI2m_403yq-TLxJ1QHtkzKE7_c_9Gal
Authentication needed
Go to the following url in your browser:
https://accounts.google.com/o/oauth2/
auth?access_type=offline&client_id=...
Enter verification code: ...
Downloading cloudfile.zip -> cloudfile.zip
Downloaded 1YQI2m_403yq-TLxJ1QHtkzKE7_c_9Gal at 86.0 MB/s, total 23.6 GB
```

Transfer: Amazon Web Services

- https://docs.aws.amazon.com/cli/latest/reference/

```
[user1@helix ~]$ module load aws
[user1@helix ~]$ aws s3 cp /data/group1/project2 \
  s3://mybucket/myfolder –recursive
[user1@helix ~]$ cd /data/group1/project3
[user1@helix project3]$ aws s3 sync s3://mybucket/project3 project3
```

GROUP AND DATA MANAGEMENT

Group Communication

- Who has access to what?
- Is the data being maintained?

Group Owner

- The group owner has the responsibility of managing members and permissions
- Can authorize changes

Group Lifecycle Plan

- What to do when a user leaves?
- Long-term archive?

Data Management Plan

- Create a plan or projection of what data is needed and what will be generated
- Have a rough idea of scale (100GB? 100TB?)
- How long will data be needed?
- Who should have access?
- Who has management responsibility?

HPC Data Lifecyle Policies

- Data is ultimately owned by PI
- When an account is deleted, the data is kept for **six months, then deleted**
- The PI is alerted with options
  - Change permissions (in shared /data directory)
  - Merge into another directory
  - Transfer the contents (size-dependent)
  - Delete the data

Questions? Comments?

staff@hpc.nih.gov