Using the NIH HPC Storage Systems Effectively

Tim Miller

btmiller@hpc.nih.gov

https://hpc.nih.gov

Motivation

- Often get e-mail complaining about slow access to directories
- File storage systems are often over-saturated, leading to system problems that take staff time to resolve
- Users’ work is often negatively impacted (sometimes without their knowledge).

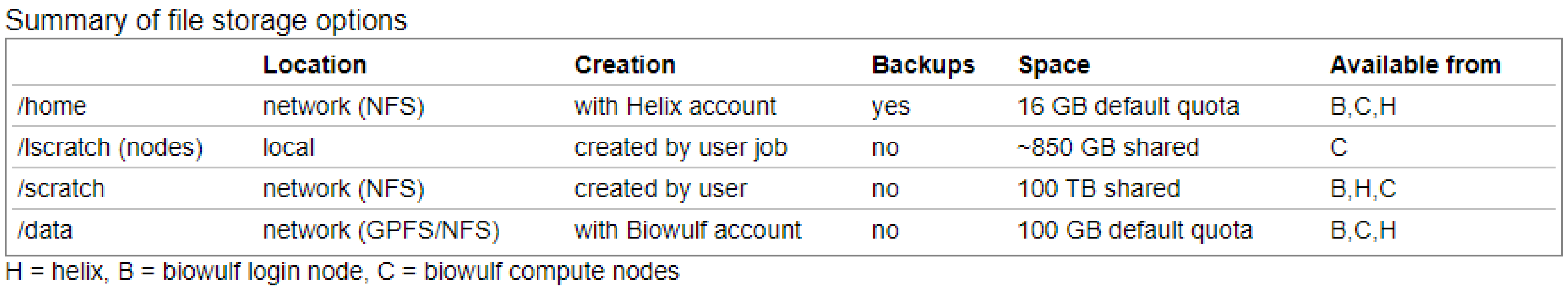
Presentation overview/outline

- Overview of HPC systems storage
    - Different areas (/home, /data, /scratch, object store)
    - Quotas and quota increases
    - Snapshots
- Understanding input and output (I/O)
    - I/O patterns
    - Introduction to data and metadata
    - A brief look at HPC storage systems
- Putting it all together – using storage effectively
    - Profiling and benchmarking your application’s I/O usage
    - Understanding when you’re generating too much I/O on the system

Presentation overview/outline

- Overview of HPC systems storage
  - Different areas (/home, /data, /scratch, object store)
  - Quotas and quota increases
  - Snapshots
- Understanding input and output (I/O)
  - I/O patterns
  - Introduction to data and metadata
  - A brief look at HPC storage systems
- Putting it all together – using storage effectively
  - Profiling and benchmarking your application's I/O usage
  - Understanding when you're generating too much I/O on the system

The NIH HPC filesystems



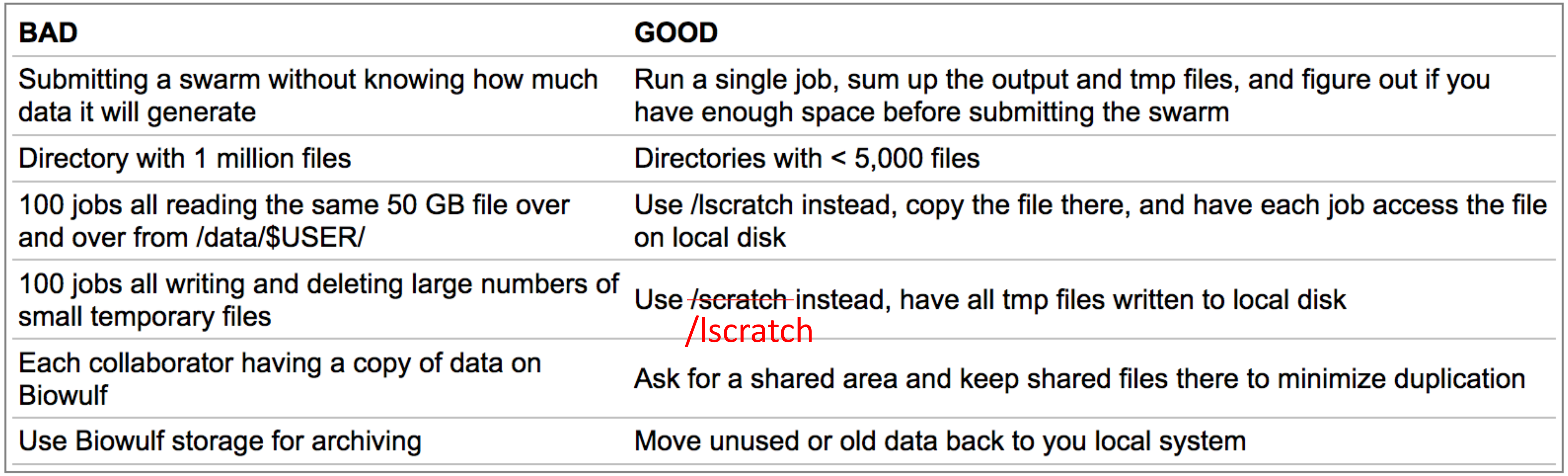
Summary of file storage options

	Location	Creation	Backups	Space	Available from
/home	network (NFS)	with Helix account	yes	16 GB default quota	B,C,H
/lscratch (nodes)	local	created by user job	no	~850 GB shared	C
/scratch	network (NFS)	created by user	no	100 TB shared	B,H,C
/data	network (GPFS/NFS)	with Biowulf account	no	100 GB default quota	B,C,H

H = helix, B = biowulf login node, C = biowulf compute nodes

- /home is small – only filesystem backed up to tape – No shared areas or quota increases!
  - Snapshots and off-site tape back-ups
- /data - in practice you will keep most of your working files here
  - Quota increases available with justified need
  - Shared data directories available on request
  - Snapshots available (less frequent than /home)
  - ACLs
- /scratch – shared area for temporary data
- /lscratch – compute node local scratch; allocated with batch jobs

General best practices



BAD	GOOD
Submitting a swarm without knowing how much data it will generate	Run a single job, sum up the output and tmp files, and figure out if you have enough space before submitting the swarm
Directory with 1 million files	Directories with < 5,000 files
100 jobs all reading the same 50 GB file over and over from /data/$USER/	Use /lscratch instead, copy the file there, and have each job access the file on local disk
100 jobs all writing and deleting large numbers of small temporary files	Use ~~/scratch~~ /lscratch instead, have all tmp files written to local disk
Each collaborator having a copy of data on Biowulf	Ask for a shared area and keep shared files there to minimize duplication
Use Biowulf storage for archiving	Move unused or old data back to you local system

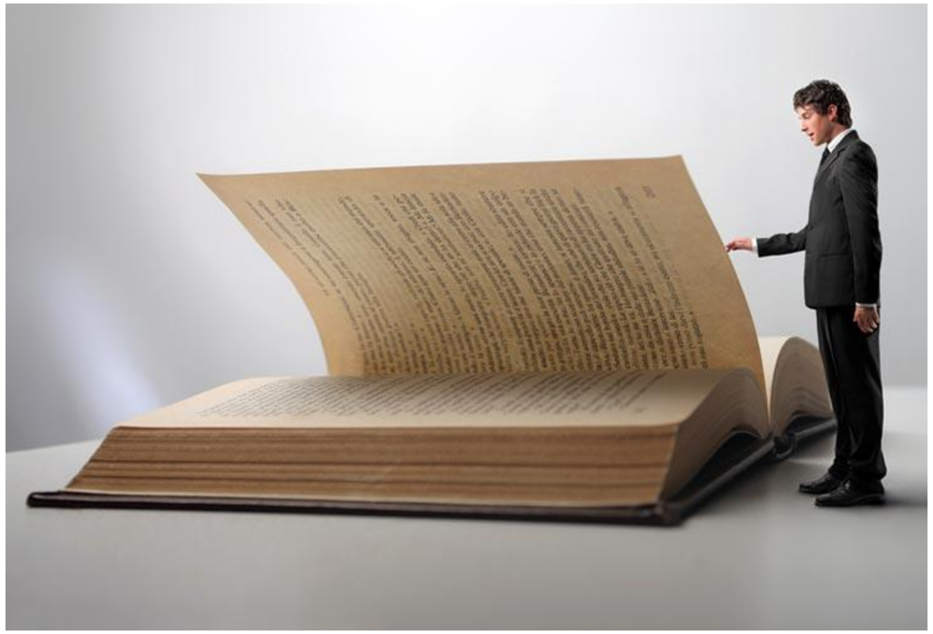
We'll be talking a lot about the "whys" behind some of these rules!

The HPC Object Store

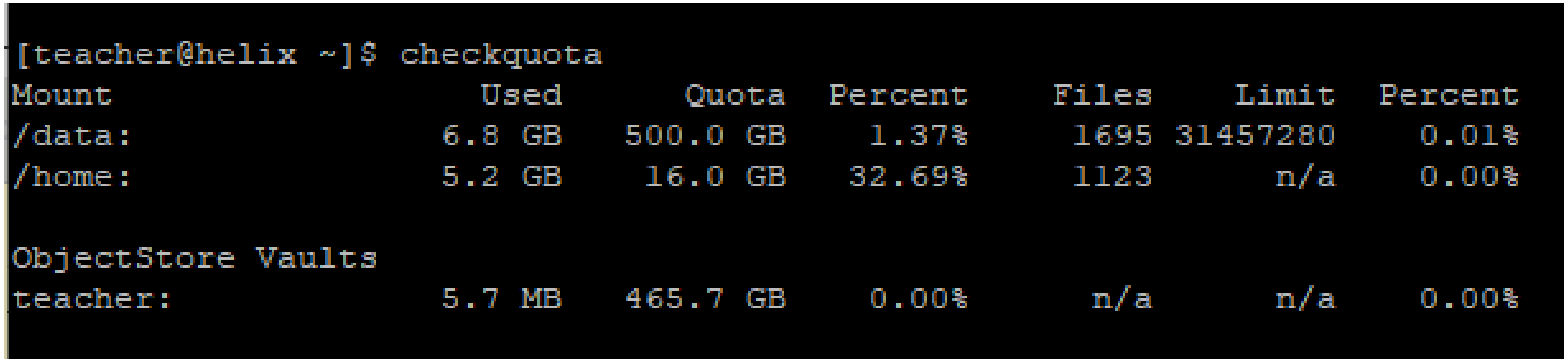
- "Web Scale" storage
  - Highly reliable (dispersed over multiple sites)
  - Easy to expand (just add more disks)
  - Accessed via simple list, put, get, delete semantics (examples forthcoming)
- Different from file based storage systems
  - Objects are accessed by **NAME**, not **PATH**
  - Completely flat name space
  - No concept of directories, but "/" is a valid character in object names
  - Data and metadata are stored together with the object (sometimes true in file storage systems as well)
- More info: https://hpc.nih.gov/storage/object.html
- Unless otherwise noted, the rest of this class deals with **file** (not object) storage. Separate object storage class!

Use cases for object storage

- Read-intensive workloads
  - Object storage is much more efficient at reading than writing.
  - An **entire** object has to be re-written for each change
    - Computationally expensive to process and disperse the data
    - Lots of over-writing
- Static data
  - Related to the above
  - Data that doesn’t change often, but still used
  - E.g. reference genomics files



Understanding your disk quota

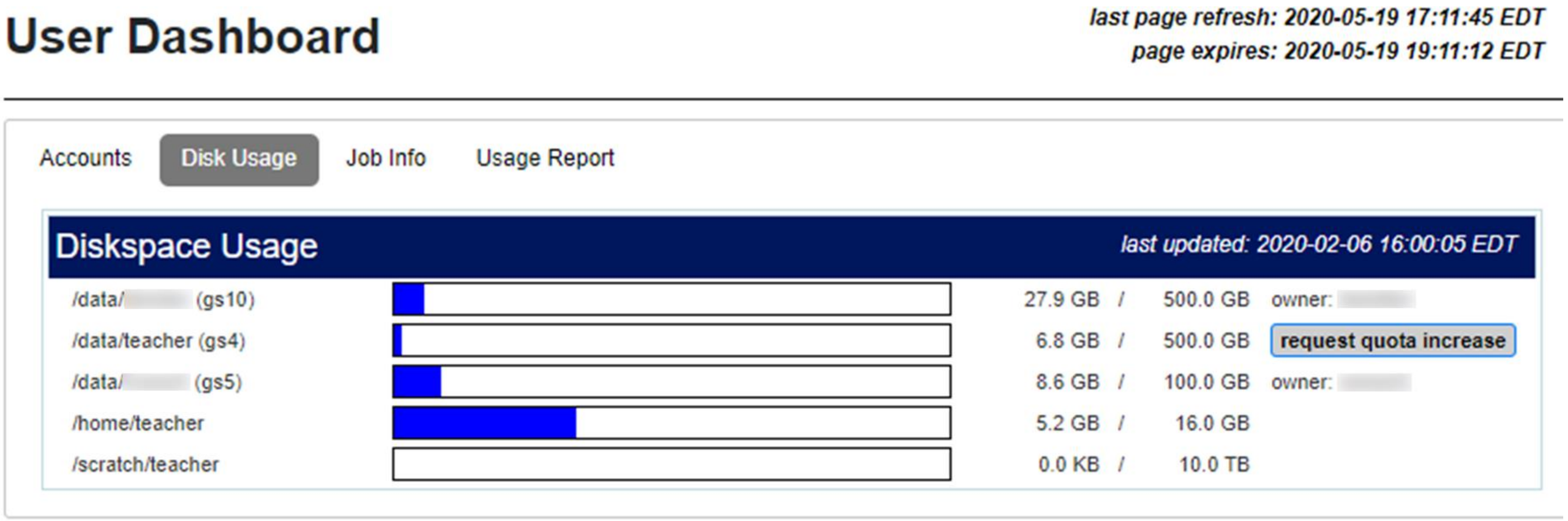


```
[teacher@helix ~]$ checkquota
Mount                   Used      Quota   Percent    Files     Limit   Percent
/data:                6.8 GB   500.0 GB     1.37%     1695  31457280     0.01%
/home:                5.2 GB    16.0 GB    32.69%     1123       n/a     0.00%

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

- Use the checkquota command
  - Shows all directories <u>you</u> have access to.
- You can also get this information from your user dashboard
- Some storage systems have limits on the number of files – please keep these in mind (more on this later).

Quota increases: on the user dashboard!



User Dashboard

last page refresh: 2020-05-19 17:11:45 EDT
page expires: 2020-05-19 19:11:12 EDT

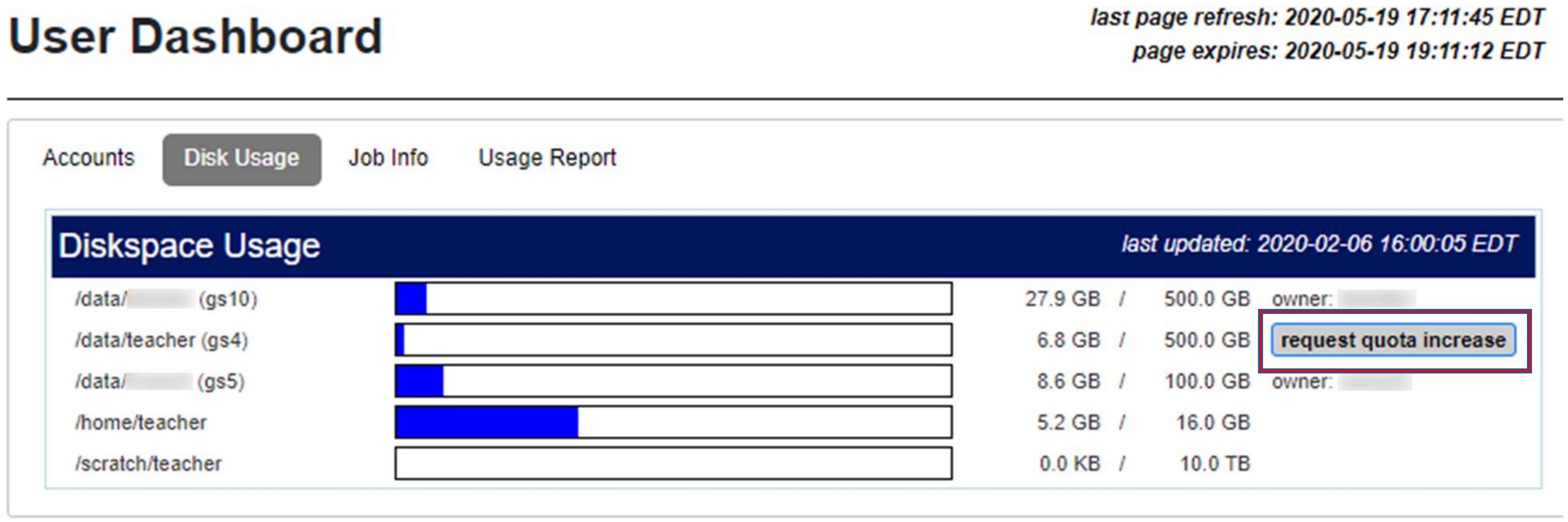
Accounts | **Disk Usage** | Job Info | Usage Report

Diskspace Usage

last updated: 2020-02-06 16:00:05 EDT

/data/[redacted] (gs10)	27.9 GB /	500.0 GB	owner: [redacted]
/data/teacher (gs4)	6.8 GB /	500.0 GB	request quota increase
/data/[redacted] (gs5)	8.6 GB /	100.0 GB	owner: [redacted]
/home/teacher	5.2 GB /	16.0 GB	
/scratch/teacher	0.0 KB /	10.0 TB	

Quota increases: on the user dashboard!



User Dashboard

last page refresh: 2020-05-19 17:11:45 EDT
page expires: 2020-05-19 19:11:12 EDT

Diskspace Usage

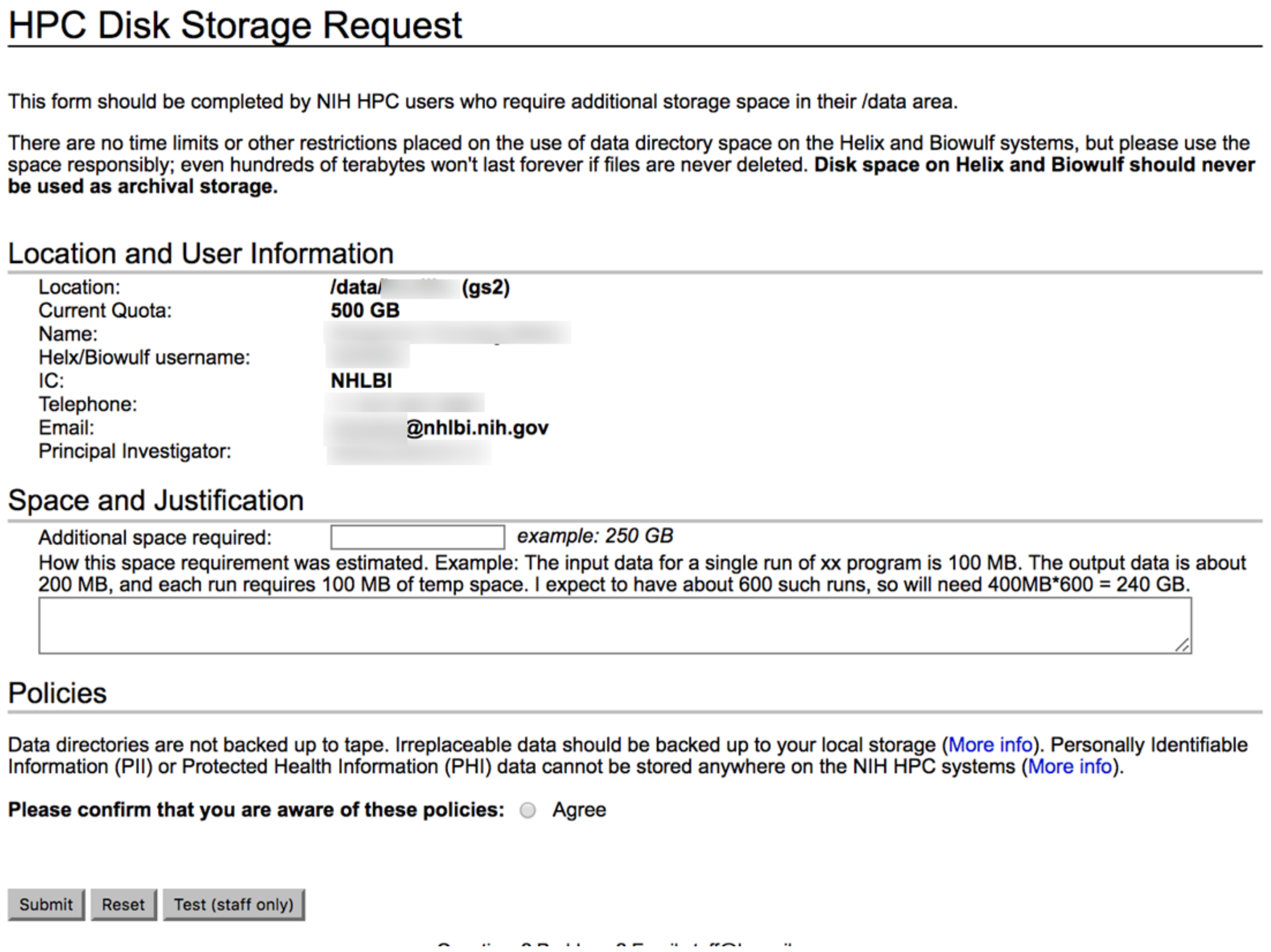
last updated: 2020-02-06 16:00:05 EDT

/data/[redacted] (gs10)	27.9 GB	/	500.0 GB	owner: [redacted]
/data/teacher (gs4)	6.8 GB	/	500.0 GB	request quota increase
/data/[redacted] (gs5)	8.6 GB	/	100.0 GB	owner: [redacted]
/home/teacher	5.2 GB	/	16.0 GB	
/scratch/teacher	0.0 KB	/	10.0 TB	

Can only request quota increases on data directories you own!

Quota increases: on the user dashboard!

- Only two things to fill in now!
  - Amount of space
  - Justification
- Also need to accept storage policies
  - NO PHI/PII
  - NO BACKUPS!



HPC Disk Storage Request

This form should be completed by NIH HPC users who require additional storage space in their /data area.

There are no time limits or other restrictions placed on the use of data directory space on the Helix and Biowulf systems, but please use the space responsibly; even hundreds of terabytes won't last forever if files are never deleted. **Disk space on Helix and Biowulf should never be used as archival storage.**

Location and User Information

Location:	**/data/**[redacted] **(gs2)**
Current Quota:	**500 GB**
Name:	[redacted]
Helx/Biowulf username:	[redacted]
IC:	**NHLBI**
Telephone:	[redacted]
Email:	[redacted]**@nhlbi.nih.gov**
Principal Investigator:	[redacted]

Space and Justification

Additional space required: ____ *example: 250 GB*

How this space requirement was estimated. Example: The input data for a single run of xx program is 100 MB. The output data is about 200 MB, and each run requires 100 MB of temp space. I expect to have about 600 such runs, so will need 400MB*600 = 240 GB.

Policies

Data directories are not backed up to tape. Irreplaceable data should be backed up to your local storage (More info). Personally Identifiable Information (PII) or Protected Health Information (PHI) data cannot be stored anywhere on the NIH HPC systems (More info).

Please confirm that you are aware of these policies: ○ Agree

Submit | Reset | Test (staff only)

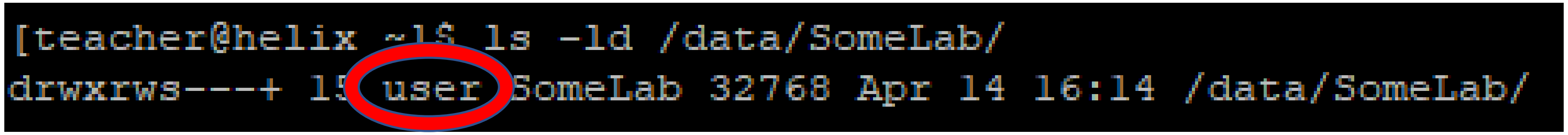
Using space efficiently

- Instead of running to the quota request form, think about whether you can be using space more efficiently.
  - Move files back to your local computing infrastructure when you're done processing
  - Delete any raw data or intermediate files that you're sure you won't need again (or that are backed up elsewhere)
  - Compress (gzip, bzip2, etc.) files when not in active use (note: not all files compress well).
    - FASTQ files should always be compressed
    - Molecular dynamics binary trajectories generally don't compress much
- Storage space on the NIH HPC file systems is NOT to be used for archiving. However, we provide a time-limited archive for large data sets using the object store.

Shared data directories

- Requested from https://hpc.nih.gov/dashboard/shared_data_request.html
  - Redirects to the user dashboard.
- A new group will be created (group name must begin with a capital letter).
  - Group members must be specified
- Justification for the storage request must be given as with a personal data directory quota increase.
- Don’t open your personal data directory to world access!
  - Shared group directories should only be accessible by the group that owns them.
- Requestor becomes “group owner”
  - The only one who can request a quota increase
  - The only one who can request users be added to or removed from the group

Shared data directories and permissions

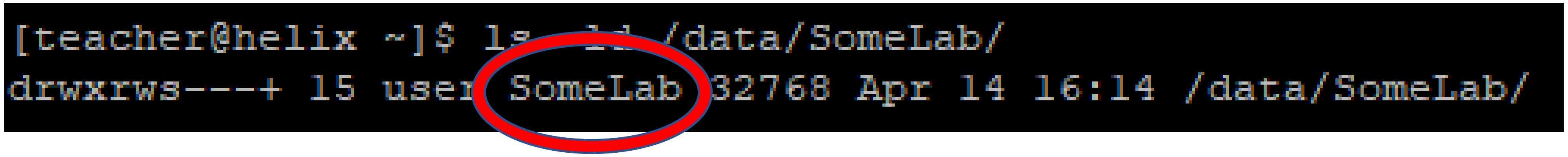


```
[teacher@helix ~]$ ls -ld /data/SomeLab/
drwxrws---+ 15 user SomeLab 32768 Apr 14 16:14 /data/SomeLab/
```

Group owner – the ONLY user allowed to request quota increases for the group.

- New directories created under the top level may **NOT** have the SGID bit set.
- Users can override group ownership and permissions of directories that they create within the shared area.
  - Some applications do this without informing the user.
- Coordination and care among group members is needed.
- Users in groups are responsible for maintaining correct permissions!

Shared data directories and permissions

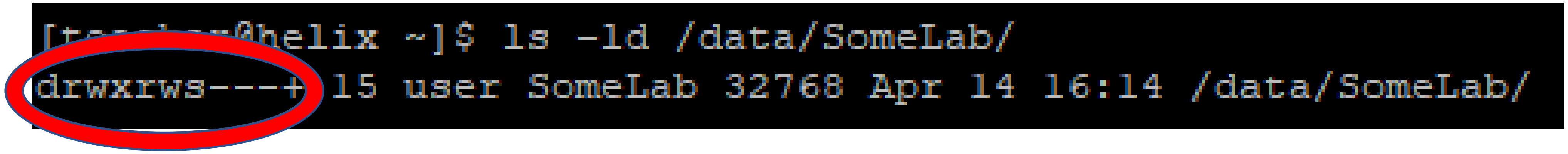


```
[teacher@helix ~]$ ls -ld /data/SomeLab/
drwxrws---+ 15 user SomeLab 32768 Apr 14 16:14 /data/SomeLab/
```

Group name – same as the shared data directory name.

- New directories created under the top level may **NOT** have the SGID bit set.
- Users can override group ownership and permissions of directories that they create within the shared area.
  - Some applications do this without informing the user.
- Coordination and care among group members is needed.
- Users in groups are responsible for maintaining correct permissions!

Shared data directories and permissions



```
[t[illegible]r@helix ~]$ ls -ld /data/SomeLab/
drwxrws---+ 15 user SomeLab 32768 Apr 14 16:14 /data/SomeLab/
```

Permissions – all group members can write at the top level. The SGID bit (“s” in the group execute permissions) indicates all files created in this directory will have a group ownership of SomeLab.

- New directories created under the top level may **NOT** have the SGID bit set.
- Users can override group ownership and permissions of directories that they create within the shared area.
  - Some applications do this without informing the user.
- Coordination and care among group members is needed.
  - Set umask to 007 so that all files/directories created become group writeable
  - Some users request that their primary group be changed to the shared group
- Bonus: what does the “+” sign after the permissions mean?

A few notes about /scratch

- /scratch is a **network accessed, global** /scratch directory
  - The same scratch directory is available on helix, biowulf, and all of the compute nodes.
  - Files deleted if not accessed for 7 days or when /scratch gets full!
- Good use-cases for /scratch
  - A temporary means of sharing data with a colleague who has a HPC account
  - Storing lightly accessed temporary files that must be accessed from multiple nodes.
- Bad use-cases for /scratch
  - Storing application temporary files that are only accessed from a single host (use /lscratch instead).
  - Storing anything that must be read and written frequently (use /data or /lscratch instead).
  - Storing valuable, hard to reproduce data (no back-ups or snapshots; use /home or /data as appropriate).

Using /lscratch

- /lscratch = local scratch space on a compute node
- Allocated as a generic resource:
  - sbatch … --gres=lscratch:100 ← allocates 100 GB
  - swarm -f swarmfile --gres=lscratch:100 ...
  - Ibid for sinteractive
  - At /lscratch/$SLURM_JOBID
- Benefits
  - Local to node, no network traffic
  - Fewer users sharing it
  - Nodes have fast solid-state disks

When to use /lscratch (and when not)

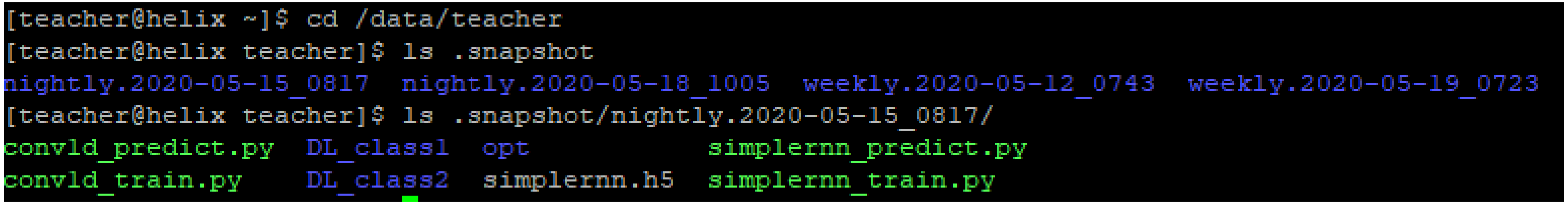
- Use /lscratch when…
  - Many jobs will be independently reading in the same data file.
  - Many jobs will be independently writing out output files.
  - Many jobs will be writing out a lot of independent temporary files
  - Jobs will be doing large amounts of random I/O (more on this later)
  - Swarms that read/write a lot of I/O should almost always use lscratch!!!
  - MATLAB batch jobs & swarms should copy the MCR to local scratch!!
- Don’t use /lscratch (or why bother) when...
  - Your job needs to write out a file visible to multiple nodes (this is rare?)
  - Your job is only reading/writing a few files and not very much data.
- If in doubt, e-mail staff@hpc.nih.gov

Snapshots

- No back-ups of HPC data directories, but there are snapshots.
- A snapshot is a copy of the directory as it existed at a given point in time.
- Current snapshot retention policies:
  - /home – 7 hourly, 6 daily, 8 weekly
  - /data – 2 daily, 2 weekly
  - Weekly snapshots are taken on Sundays for NFS home directories and on Tuesdays for GPFS data directories.

Accessing snapshots

- Navigate to .snapshot/ in your /data directory
  - Will NOT tab complete; need to type out the full directory name.
  - Users on the VF storage system will need to go to /vf/users/.snapshot.
- Will see several subdirectories that contain read-only copies of your directory at a point in time.
- Can copy data from snapshots into your "main" home directory if, for example, you accidentally delete a file!



```
[teacher@helix ~]$ cd /data/teacher
[teacher@helix teacher]$ ls .snapshot
nightly.2020-05-15_0817  nightly.2020-05-18_1005  weekly.2020-05-12_0743  weekly.2020-05-19_0723
[teacher@helix teacher]$ ls .snapshot/nightly.2020-05-15_0817/
convld_predict.py  DL_class1  opt           simplernn_predict.py
convld_train.py    DL_class2  simplernn.h5  simplernn_train.py
```

Snapshots are NOT back-ups

Snapshots are stored on the same physical hardware as home/data directories. Therefore, **in the event of a hardware or facilities failure, the snapshots will also be lost!**

You **must** back up irretrievable data to a local system (or put it in your /home directory if it will fit).

A guide to choosing a storage system

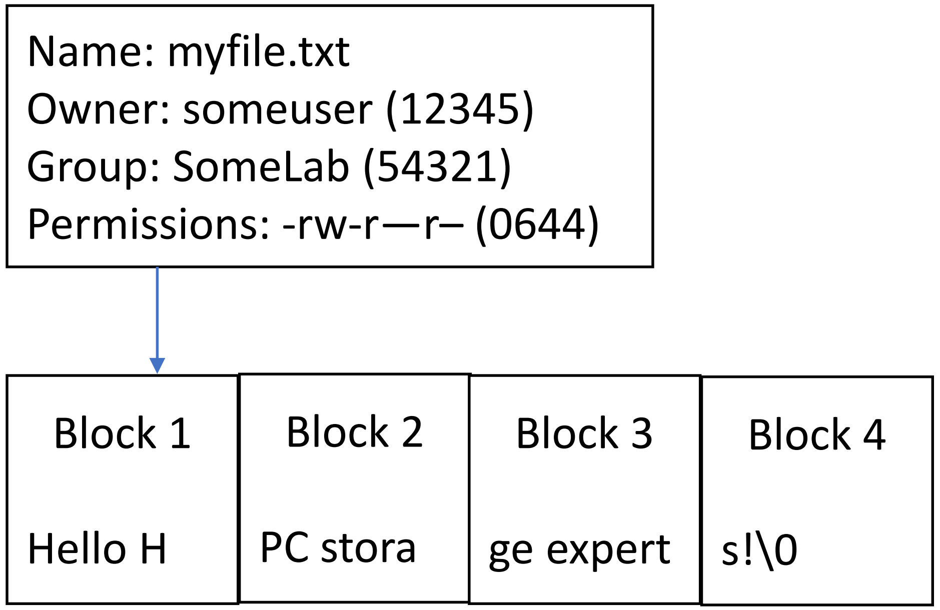
- /home
  - Small files that are very important, low I/O intensity
  - Not for files that need to be shared with other users
- /data
  - Bulk data files and scripts
  - Consider using shared data areas for sharing
  - Shared, parallel filesystems – intensive, single node I/O -> /lscratch.
- /scratch
  - **Low I/O intensity** job data that needs to be accessed from multiple nodes.
  - Good for short-term sharing.
- /lscratch
  - **High I/O intensity** job data that only needs to be accessed from a single node.
- Object
  - Primarily read-only data; low to medium intensity, but large capacity.

Presentation overview/outline

- Overview of HPC systems storage
  - Different areas (/home, /data, /scratch, object store)
  - Quotas and quota increases
  - Snapshots
- Understanding input and output (I/O)
  - I/O patterns
  - Introduction to data and metadata
  - A brief look at HPC storage systems
- Putting it all together – using storage effectively
  - Profiling and benchmarking your application's I/O usage
  - Understanding when you're generating too much I/O on the system

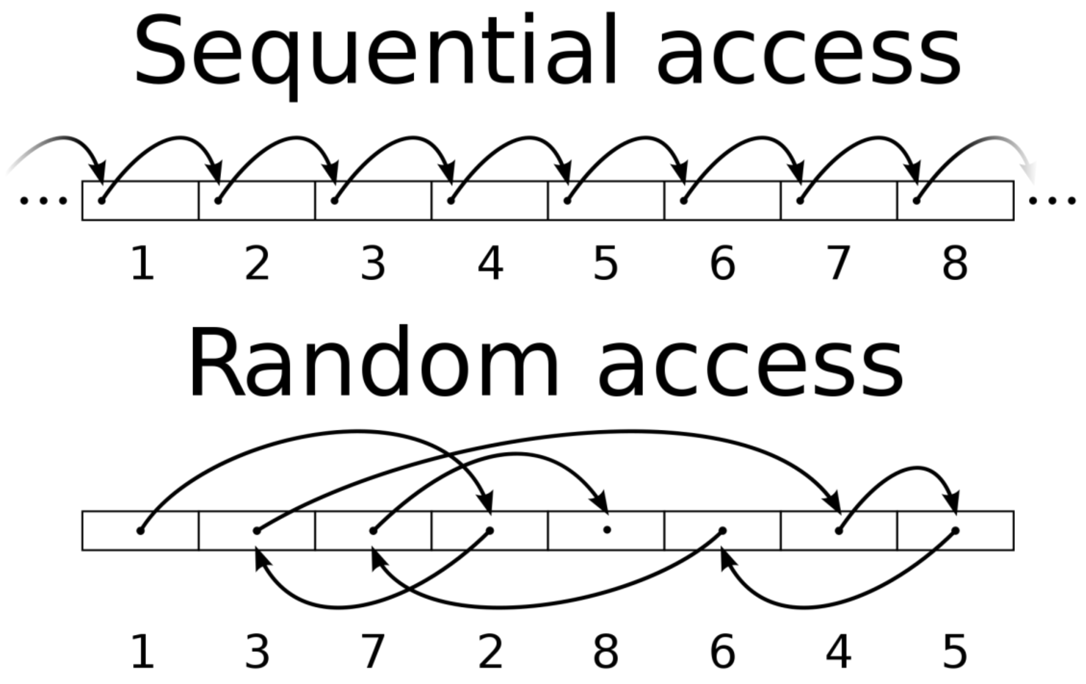
A block about blocks

- A filesystem organizes files and directories into **blocks** of data
  - This is because hard disks can only read and write information in discrete-sized chunks.
- Each file you store that's more than ~ 3 KB takes up at least one full block on the filesystem.
  - The exact size of a block differs from filesystem to filesystem and may be chosen by the administrator who creates the filesystem.



Sequential vs. random access

- Sequential access: blocks in a file are read one after the other
  - Example: reading in a series of sequences from a file, one after another.
  - Sometimes referred to as “streaming”.
  - E.g. reading a FASTQ file into memory
- Random access: blocks in the file are read in a random order.
  - Common in database applications
  - Much harder for I/O systems to optimize
  - Generally MUCH slower!
  - E.g. Pulling non-adjacent sequences from a BAM file.



I/O comes in all sizes

- In addition to the pattern of I/O, we also have to be concerned with the size of I/O.
- Which do you think will be faster?
  - An application that reads 100 MB sequentially by issuing 50 read requests of 2 MB each
  - An application that reads 100 MB sequentially by issuing 5000 read requests of 20 KB each
- The first case is (usually) a lot faster
  - There’s a certain amount of overhead in performing requests.
  - Some systems will try to coalesce multiple small requests into bigger ones
  - With random I/O, this can be difficult to impossible.
  - IMPORTANT NOTE: SSDs and all flash filesystems are changing this picture somewhat. What’s true today may not be true five years from now!

I/O comes in all sizes

- In addition to the pattern of I/O, we also hav be concerned with the size of I/O.
- Which do you th ill be f
    - An application tha ad requests of 2 MB each
    - An applicat ad requests of 20 KB each
- The first ca
    - There's a certain amo n p g req
    - Some systems will try to coales ltiple s equests into gger ones
    - With random I/O, this can be di lt to imposs le.
    - IMPORTANT NOTE: SSDs and all ash filesystems are changing this picture somewhat. What's true today may not be true five years from now!

Unless you are writing your own application, you don't have any direct control over this. However, it's something to be aware of!

Data vs. metadata

- When we think of a file, we usually think in terms of the **data** it contains.
  - A BAM file contains sequence alignments
  - A molecular dynamics trajectory contains atomic coordinates/velocities
  - An MRI output contains an image of someone's brain.
  - An EM image contains a picture of a cell in the body.
- **Metadata** literally means "data about data"

WHAT STARTED IT?
ANOTHER AMBIGUOUS STUDY RESULT.
SALOON
Statistically-Funny.blogspot.com

THINGS GOT TENSE FOR THE TOWNSFOLK WHEN THE THIRD META-ANALYST GANG RODE INTO TOWN.

Data vs. metadata

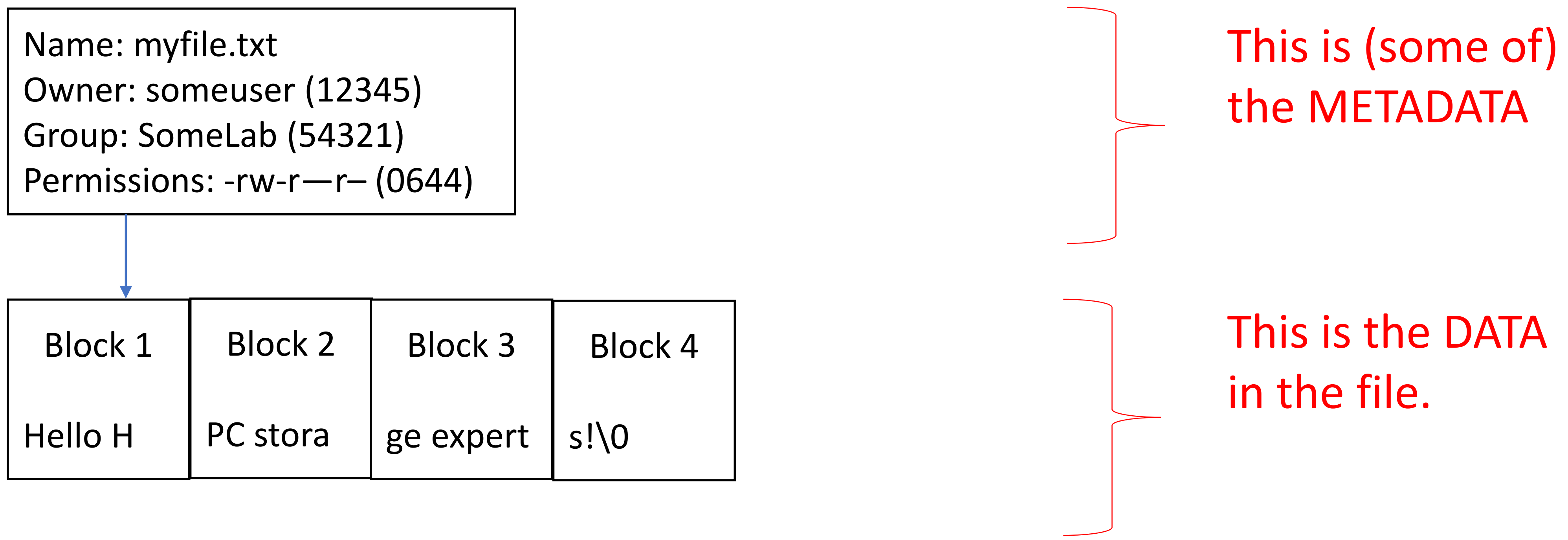
- Examples of metadata
  - When the file was created/accessed
  - The sample ID from which the file was generated
  - The access permissions of the file
  - Where the data is located physically on the underlying disk
  - Can you think of more examples?

WHAT STARTED IT?
ANOTHER AMBIGUOUS STUDY RESULT.
SALOON
Statistically-Funny.blogspot.com

THINGS GOT TENSE FOR THE TOWNSFOLK WHEN THE THIRD META-ANALYST GANG RODE INTO TOWN.

A conceptual diagram

- Remember our simple schematic…

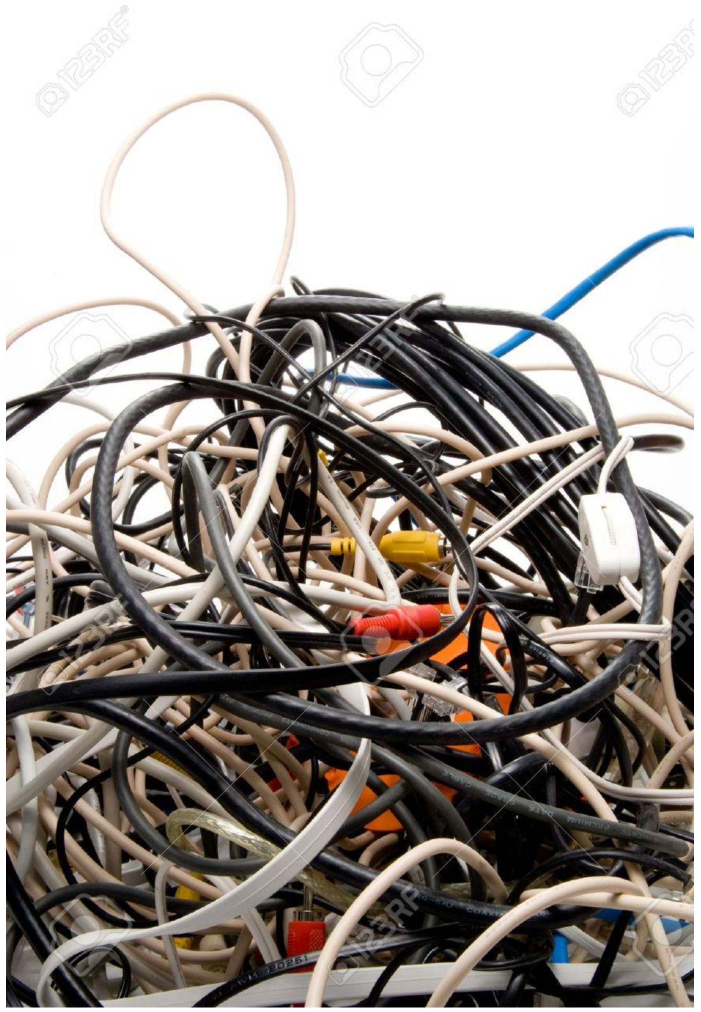


Some notes on directories

- Directories are special files that hold pointers (links) to other files.
- The more files there are in a directory, the larger amount of space the directory blocks will take up on disk
- Listing directories, resolving path names, moving files, etc. all require operations on the directory blocks.
    - The file system has to iterate through files in the directory individually
    - The bigger the directory is, the longer these operations will take.
    - This is why the HPC staff recommends having < 5000 files per directory
    - Especially true with directories that will have lots of operations happening simultaneously!

File and directory parallel access

- Reading and writing the same file from multiple parallel jobs can cause contention.
  - Especially with writing – due to the need for locks to avoid corruption
- Likewise, lots of different processes listing, creating files, etc. in the same directory is a bottleneck.
- Constant creation and deletion of files can create performance issues, particularly when multiple processes are doing it in the same directory.



Exercise: Good practice or bad practice?

- Which of the following are not good practice? Why?
  - Having 1,000,000 data files in a single directory.
  - Having separate runs of a program write output to separate files.
  - Reading a data file in once at program initiation, and then keeping the data cached in memory.
  - Using the name of a file to encode program results.
  - Having a swarm of 1,000 jobs each use the same temporary directory in /scratch.

What you can do to avoid bottlenecks!

- Use /lscratch whenever possible!!!
  - Since lscratch is local to the node – avoid all network operations
  - Also, lscratch on newer nodes is provisioned with SSDs!
- Avoid many parallel I/O operations.
  - They tend to oversaturate the disks
- Try to do I/O on large chunks
  - i.e. read and write large amounts of data
  - Less network overhead, and easier for the disk systems to optimize
  - If you're using someone else's code, this is difficult/impossible
- Avoid excessive metadata operations
  - Tend to be filtered to a small amount of disks/controllers.
  - This includes directory operations!

Exercise – where is the bottleneck?

- For the ”bad practices” we identified earlier (marked in red), what bottlenecks are likely to be relevant?
  - Having 1,000,000 data files in a single directory.
  - Having separate runs of a program write output to separate files.
  - Reading a data file in once at program initiation, and then keeping the data cached in memory.
  - Using the name of a file to encode program results.
  - Having a swarm of 1,000 jobs each use the same temporary directory in /scratch.

Presentation overview/outline

- Overview of HPC systems storage
  - Different areas (/home, /data, /scratch, object store)
  - Quotas and quota increases
  - Snapshots
- Understanding input and output (I/O)
  - I/O patterns
  - Introduction to data and metadata
  - A brief look at HPC storage systems
- Putting it all together – using storage effectively
  - Understanding when you’re generating too much I/O on the system

How do you know if you're abusing the storage systems?

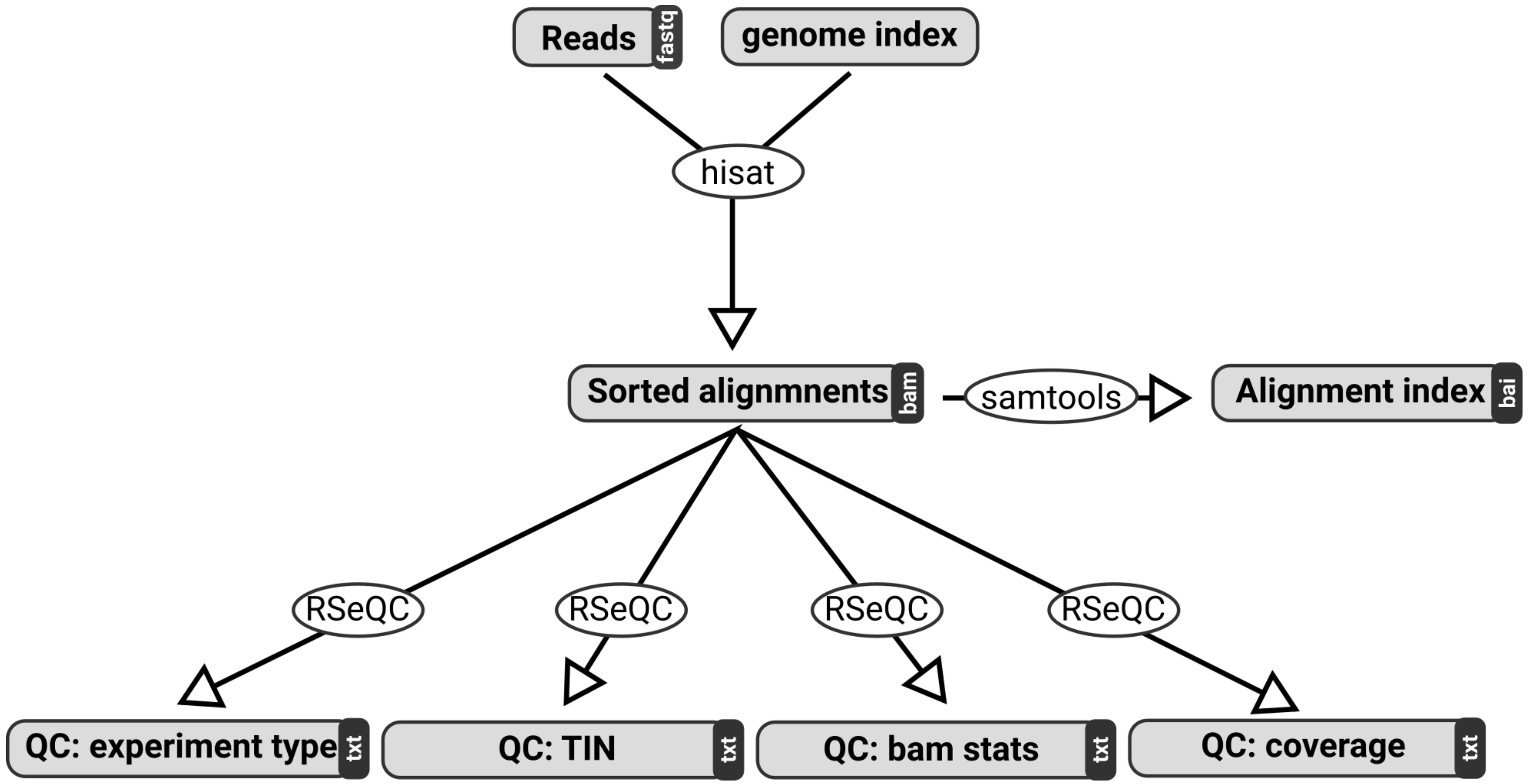
- This can be difficult to know, but there are a few clues
  - Very slow access to working directories involved with the job. (ls etc. take a long time to return)
  - For swarms, job completion speed was acceptable for small numbers of jobs, but gets dramatically slower as the size or number of jobs increases.
    - You did test with small-scale jobs first, right?!
  - The problem may be with another user's jobs, but if you started seeing problems right after you started a bunch of jobs, you are the prime suspect.
- HPC staff will notify you if we notice your jobs having an impact
  - However, please be proactive and don't wait for us to notice the problem
  - If we send you mail, it means you're having a significant negative impact on system performance.

Refactoring a workload – general principles

- Look for places where lots of parallel processes are doing I/O
  - Think about if only one process could do I/O and communicate with other processes (probably not possible with swarm).
  - Can some or all of that I/O use /lscratch instead of /scratch or /data?
- Think about bottlenecks in the workflow
  - E.g. the whole workload has to wait until one file is updated
  - Does the usage on a shared filesystem cause delays in this process?
- Does the workflow behavior change over time?
  - Do jobs have different I/O patterns in the beginning, middle, or end of their runs.
  - Would staggering this I/O be possible?

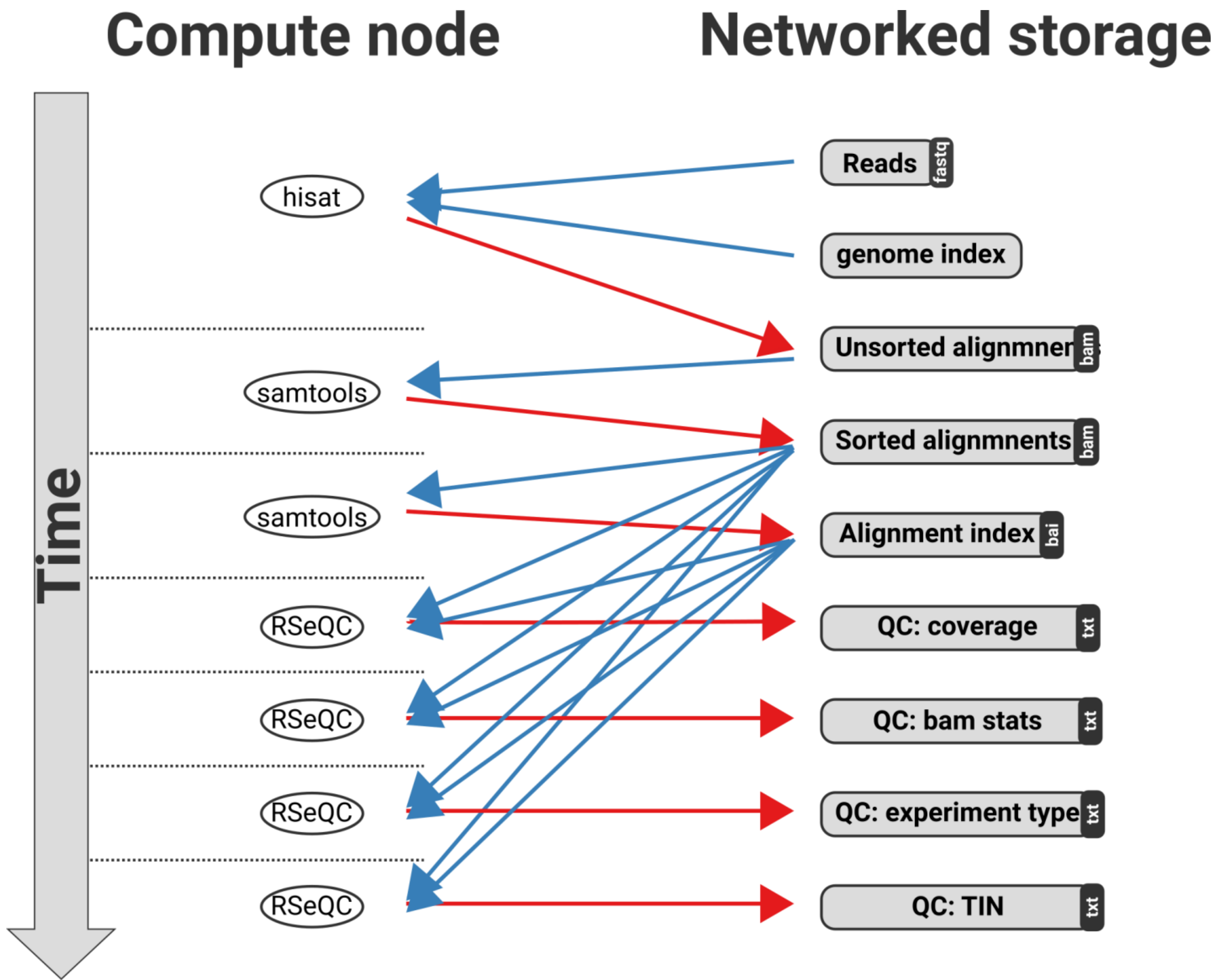
Workload refactoring: an example

- Start of a real-world genomics pipeline
  - Aligns sequences, creates index
  - RSeQC performs QC steps before continuing the pipeline
- Pipeline ran much more quickly after workload was refactored to use I/O more efficiently!



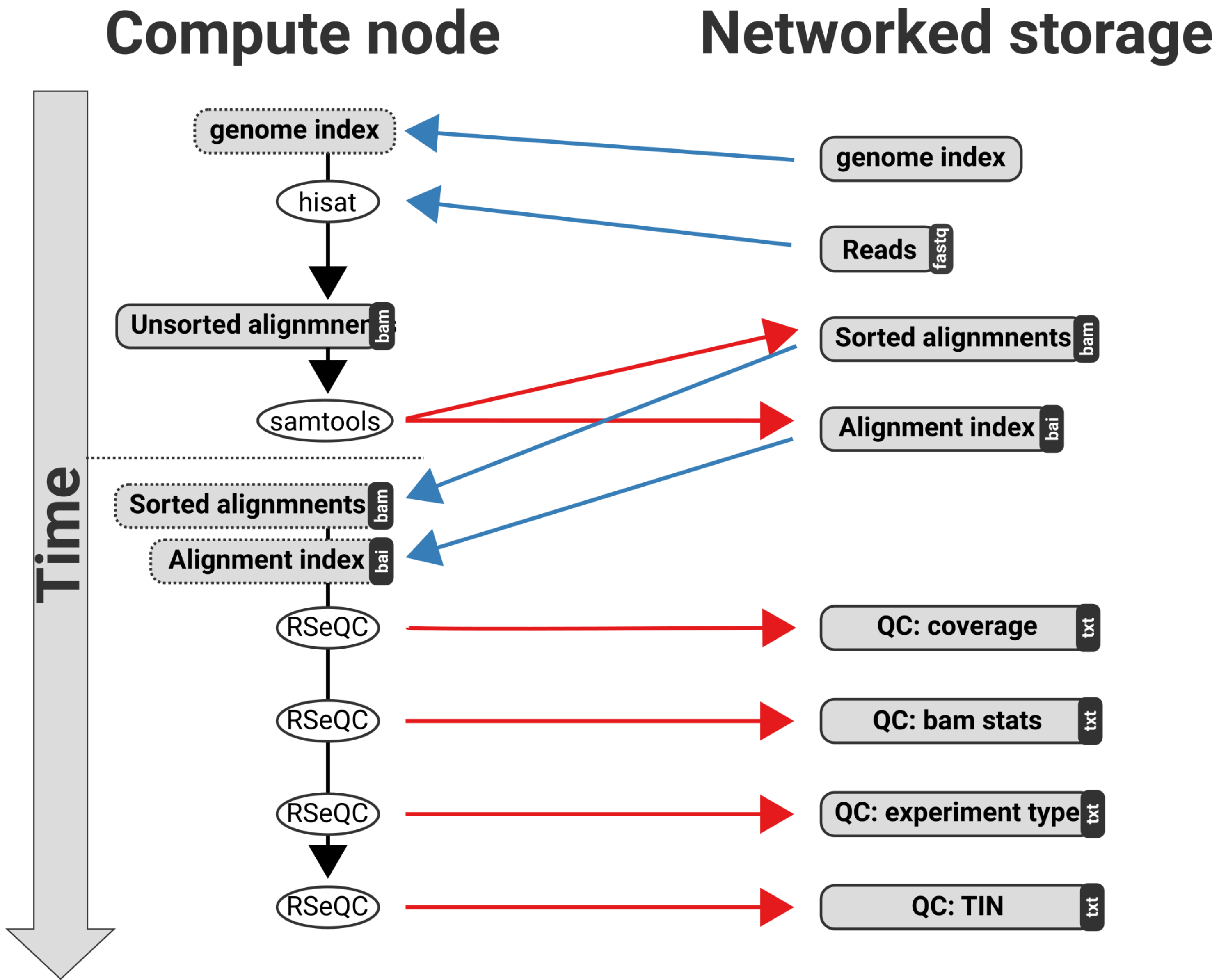
Original Pipeline

- Lots of read and writes back to network storage.
- Many parallel processes reading the same data from the storage system.
- Depending on the exact analysis done by RSeQC, random I/O is heavy.
    - Remember, lots of RSeQC processes in parallel.
- How would you fix this?



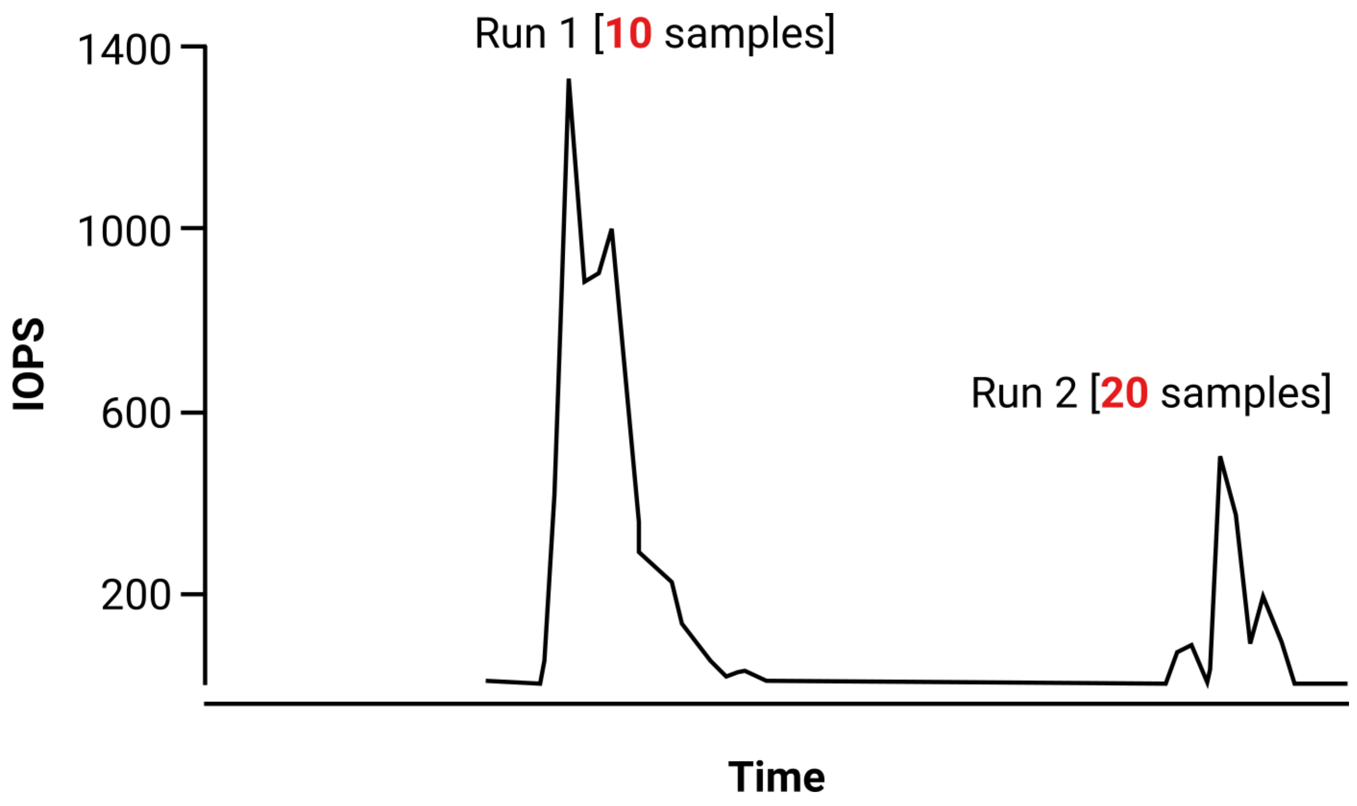
Refactored pipeline

- Only writes to networked storage when needed.
  - E.g. not after the initial read, only when index is built.
- Instead of each RSeQC reading the data from network storage, use local scratch
- Only write out final result files.

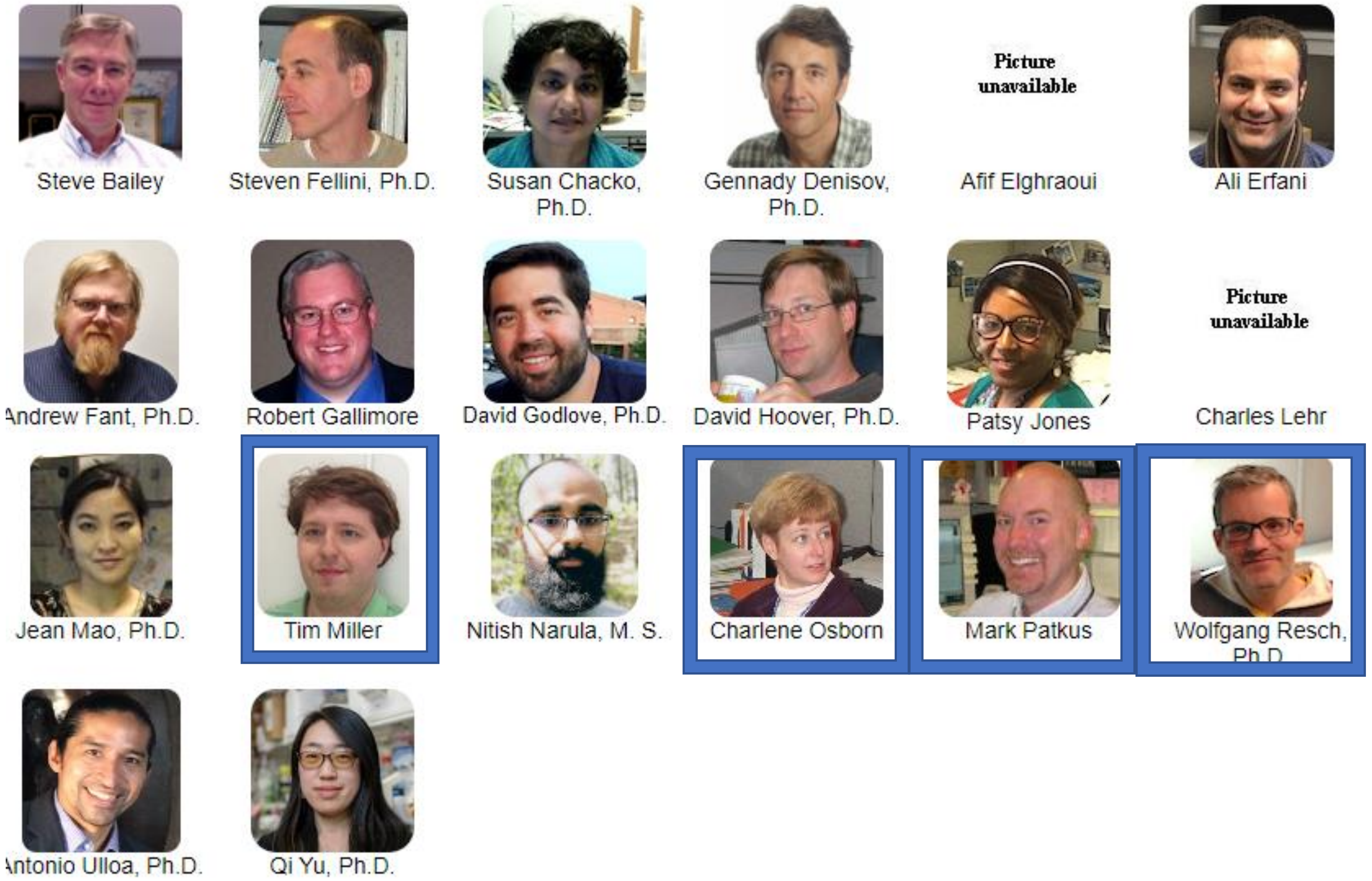


IOPS comparison

- Bottom line 1: User was able to do more science in less time.
- Bottom line 2: HPC storage admins did not have to troubleshoot performance problems.



staff@hpc.nih.gov



Wrap-up; Q&A

- Thank you for coming!
  - We hope you are able to apply the lessons learned to your own particular storage issues.
  - PLEASE reach out to staff@hpc.nih.gov for assistance; we'd love to work with you **proactively** instead of **reactively**.
- Please provide feedback on this presentation!
  - E-mail Tim btmiller@hpc.nih.gov
  - General questions staff@hpc.nih.gov

Basic storage system architecture

- /home and /scratch directories are on a large storage system that uses NFS.
- A few /data directories are on a newer flash based storage system (VF) that also uses NFS.

Basic storage system architecture

- Other data directories are on systems running IBM’s General Parallel File System (GPFS)
- Use “checkquota --gpfs” to determine if any of your data directories are on GPFS.

NFS and GPFS

- NFS and GPFS have different back-end implementations, but from a user's perspective, they work the same way.
- The systems perform similarly, though file system performance is variable dependent on how many users are accessing a given filesystem at any one time.
    - There is **no** significant performance advantage to using one system vs. the other.
- Main difference from a feature perspective: GPFS and the VF NFS system allow access control lists (ACLs) whereas the NFS implementation used for /home and /scratch does not.
    - ACLs are an advanced way of setting granular file/directory permissions – see https://hpc.nih.gov/storage/acls.html for more details
    - We will not discuss ACLs further (unless someone really wants to).

Figuring out where your data is stored

- You can use the “-a” flag on checkquota to see what filesystems the directories you have access to are on.
- spin1 = NFS, vf = NFS (newer, with ACLs), /gs[4-12] = GPFS
- Can also use “checkquota --gpfs"
- **NEVER** refer to any data directory by its absolute path (i.e. use /data/username **NOT** /gs11/users/username
  - The storage admins move directories for a variety of reasons, so the absolute paths can and do change.