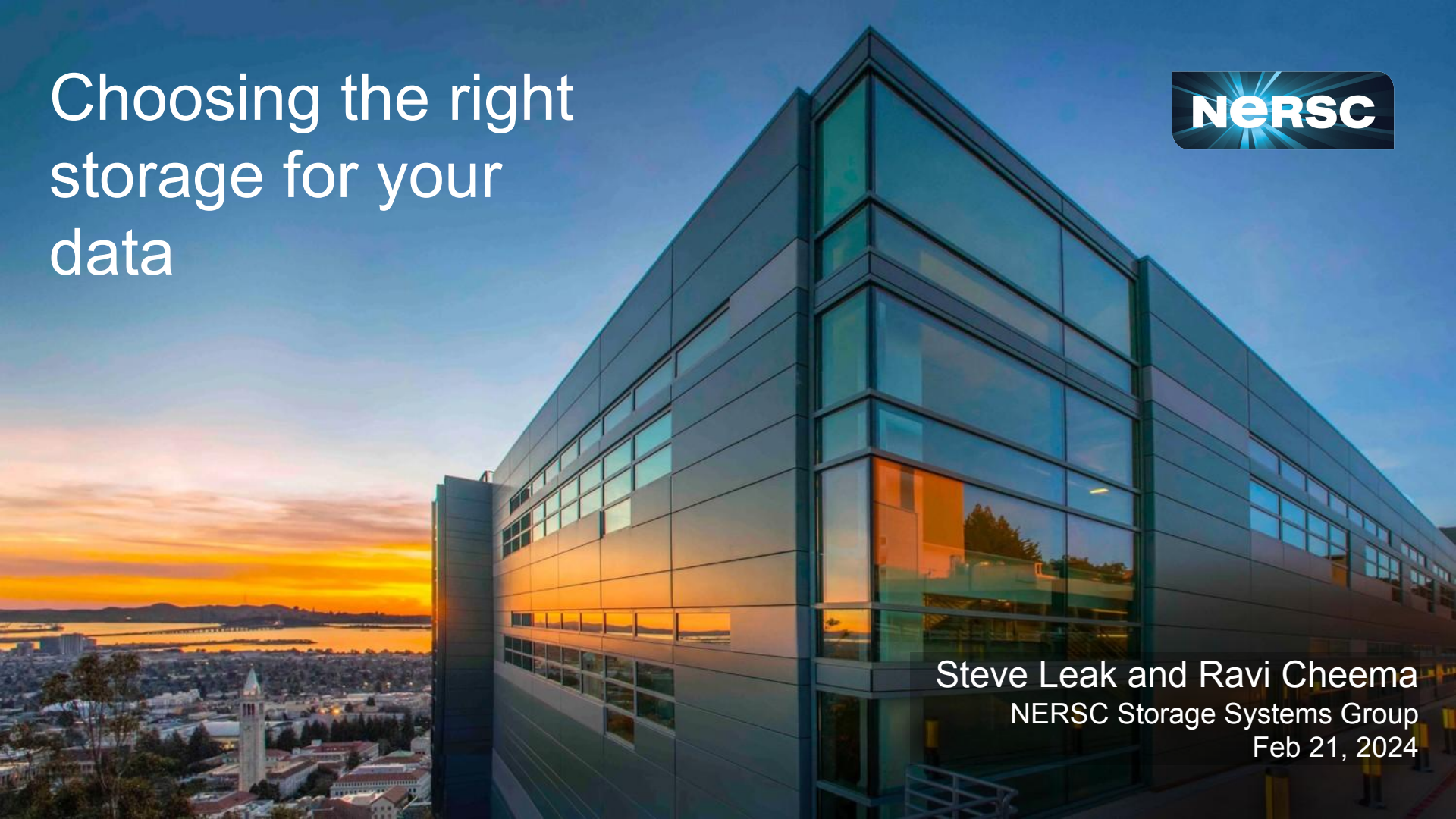
Choosing the right storage for your data



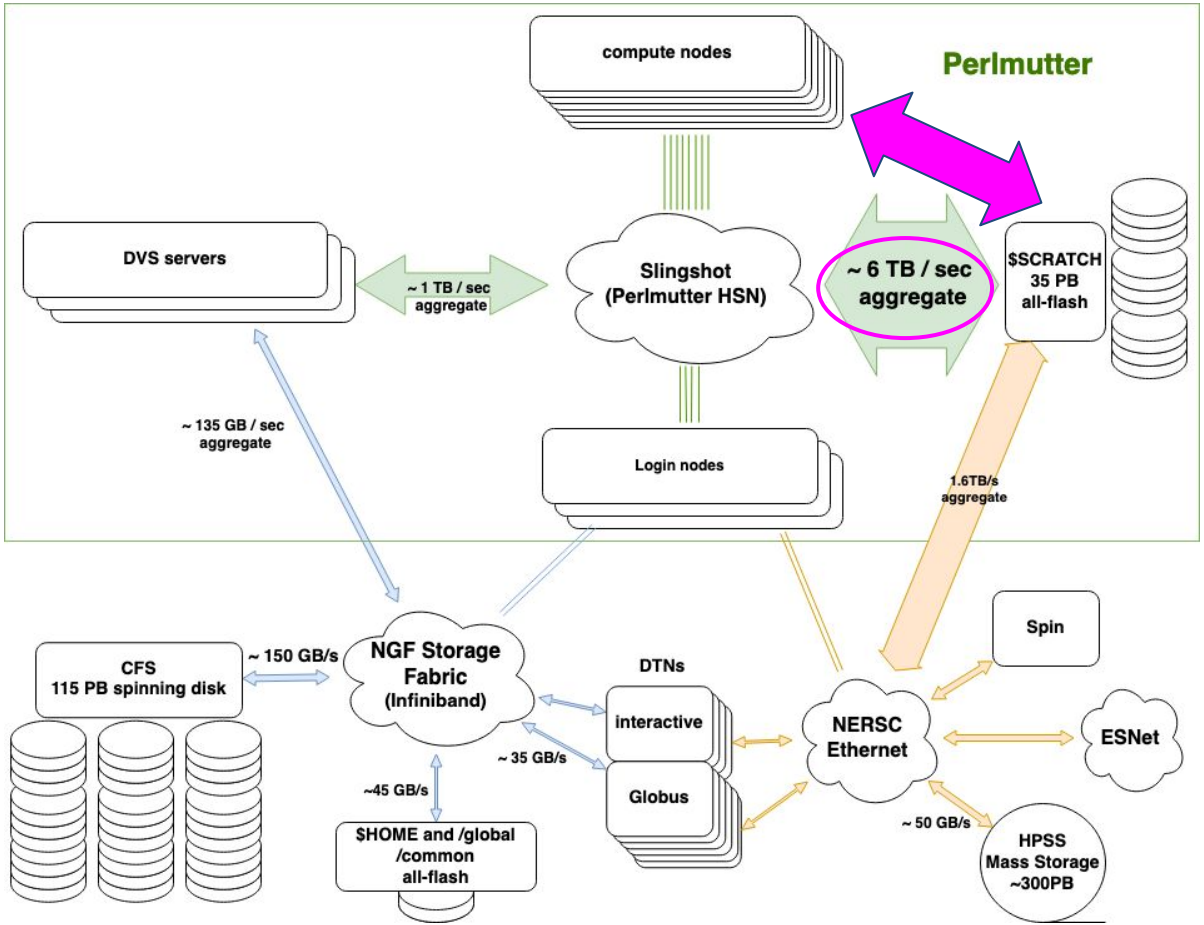
Steve Leak and Ravi Cheema
NERSC Storage Systems Group
Feb 21, 2024

TL; DR

- (Almost) All job I/O should happen on $SCRATCH
- Don't do I/O at scale over DVS
- CFS is best for actively-used data (but not source code)
- Put conda environments in /global/common/software (or better still, a container)
- Not using it for a while? Bundle it into big-ish (100GB->2TB) tar files and store on HPSS
- $HOME is good for scripts and source code
- Use Globus for moving large chunks of data around (even within NERSC)
- Have an off-NERSC copy of everything important!

I/O to $SCRATCH

- Short, fat path between computes and a big, fast, filesystem
- Supports parallel I/O (file locking)
- Short-term storage!



$SCRATCH

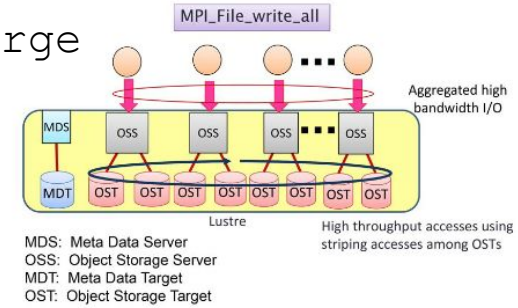
- Big: 20TB soft quota, 30TB hard quota
  - Over soft quota: job won't start
  - Over hard quota: writes fail
- Fast: Highly parallel, all-flash, 6TB/s aggregate bandwidth
- Full POSIX:
  - File locking (for parallel I/O)
  - MPI-IO
  - ACLs
- Handles big and small files and I/O operations well
  - input and output data
  - config files and scripts
  - compilation
- Not huge: full scientific datasets can be hundreds or thousands of TB - $SCRATCH is for I/O, not storage
- No backups:
  - Anything deleted (or purged) is **gone**
  - In event of catastrophic disk crash, data may not be recoverable
- Subject to purging

$SCRATCH tips

- Optimize performance with striping
  - https://docs.nersc.gov/performance/io/lustre/#nersc-file-striping-recommendations
  - Splits the file across multiple OSTs (disks)
  - By default, data on 1 OST, ideal for small files and file-per-process IO
  - Single shared-file I/O should be striped according to its size
  - Helper scripts

    `stripe_small`, `stripe_medium`, `stripe_large`
  - Manually query with

    `lfs getstripe <path>`
  - Set striping on a directory
    - New files will automatically pick it up
    - Copy files in to inherit the striping



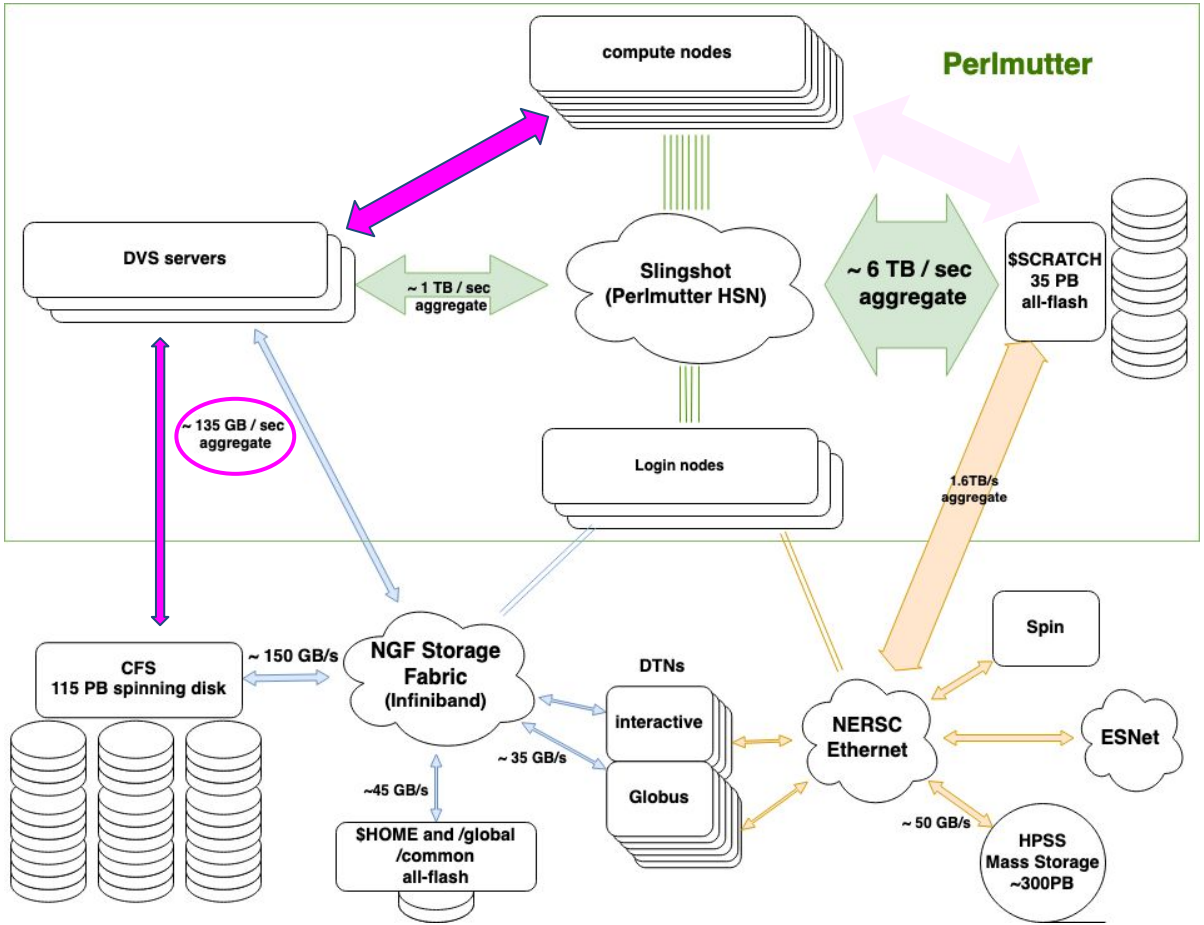
1. MPI-IO on Lustre: https://www.sys.r-ccs.riken.jp/ResearchTopics/fio/mpiio

TL; DR

- (Almost) All job I/O should happen on $SCRATCH
- Don't do I/O at scale over DVS
- CFS is best for actively-used data (but not source code)
- Put conda environments in /global/common/software (or better still, a container)
- Not using it for a while? Bundle it into big-ish (100GB->2TB) tar files and store on HPSS
- $HOME is good for scripts and source code
- Use Globus for moving large chunks of data around (even within NERSC)
- Have an off-NERSC copy of everything important!

Data on CFS

- Capacity-oriented filesystem, huge, robust
- Longer, indirect (**via DVS**) path to compute nodes
  - DVS is not suited for I/O at scale (details shortly)



CFS

- Huge: Currently 114 PB, 33 PB more coming soon
  - Large block size: great for files >>1MB
- Robust:
  - multiple layers of redundancy for reliability
  - daily snapshots retained for 7 days - if the file existed yesterday, you can recover from an accidental deletion
- Never purged, readily accessible
- Projects can split their space allocations between multiple directories and give **separate** working groups **separate** quotas
- Full POSIX when directly mounted
  - ie login nodes, DTNs (but not Perlmutter compute nodes)
- Configured for capacity over performance
  - (Still *pretty* fast, but not $SCRATCH fast)
  - Large block size - inefficient for small files, eg source code
- Not directly mounted on Perlmutter compute nodes
  - Mounted via an I/O forwarding service named DVS (more on that next), which imposes some constraints - not suitable for most job I/O
- Not backed up - make sure you have a copy of data, somewhere else

A bit about DVS

- DVS is an I/O forwarder developed by Cray
  - DVS nodes mount the filesystem, and "project" it to compute nodes
- Designed to deliver file system contents at scale
- Long history of deployment at NERSC, went live on Perlmutter on June 8, 2023
- Used only for compute nodes, logins have a native client mount

DVS

- Can provide filesystem access to thousands of nodes
- Decouples the filesystem from issues on Perlmutter
  - Using DVS on Perlmutter has greatly improved system and filesystem stability

- Not suitable for **I/O** at scale
  - Though using a read-only mount point can alleviate this
- Does not fully support POSIX
  - No file locking (shared-file writing via MPI-IO is not safe, HDF5 will complain and fail)
  - ACLs disable caching
    - `chmod` is fine
    - `setfacl` will cause subsequent accesses to be very slow
  - No mmap

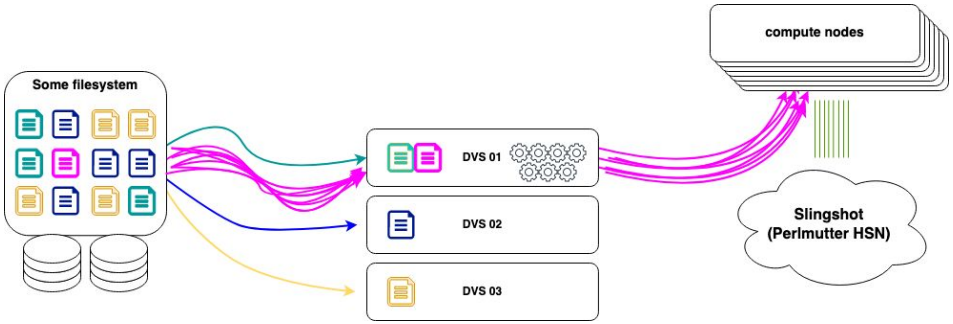
How DVS works

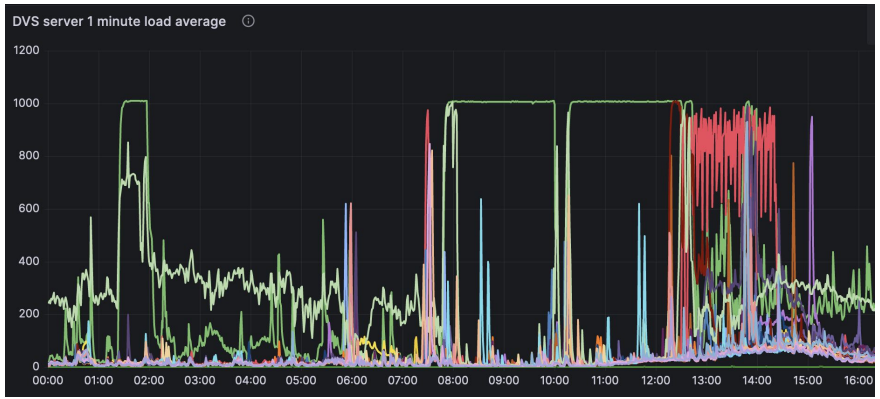
- Perlmutter has 24 gateway nodes that serve as DVS servers
- Each server can work 1000 I/O threads at once
- Can cache data to dramatically improve performance at large scales
- Two service modes:
  - Read / Write (RW): gateway server is determined when file is created (hash of inode), stays constant, zero cache
  - Read Only (RO): file can be served by all gateways, stays in cache for 30 seconds
- How to get the benefits (and avoid the limitations) of DVS:

  https://docs.nersc.gov/performance/io/dvs/#best-practices-for-dvs-performance-at-scale

DVS with read-write mount ($HOME, CFS)

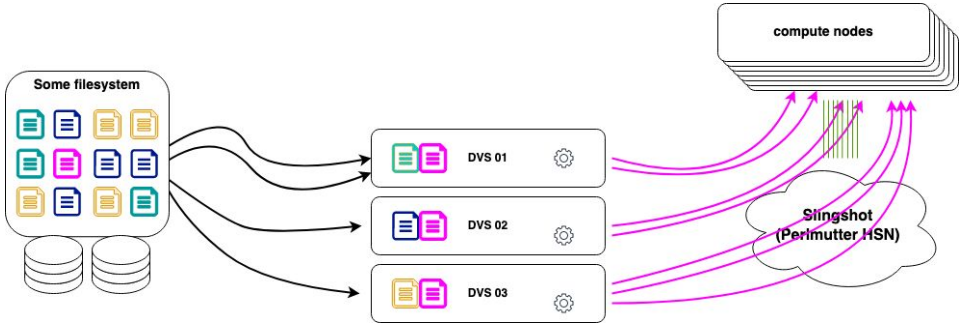
- Eg: a 100-node job using conda environment in $HOME
  - 12,800 processes all try to read `/global/homes/e/elvis/.conda`
  - No cache, so it is fetched from the filesystem 12,800 times
  - The DVS server that "owns" that file drowns under the load, while the processes wait in line
  - The job progresses only very slowly, and may fail (and other jobs using that server might be impacted too)

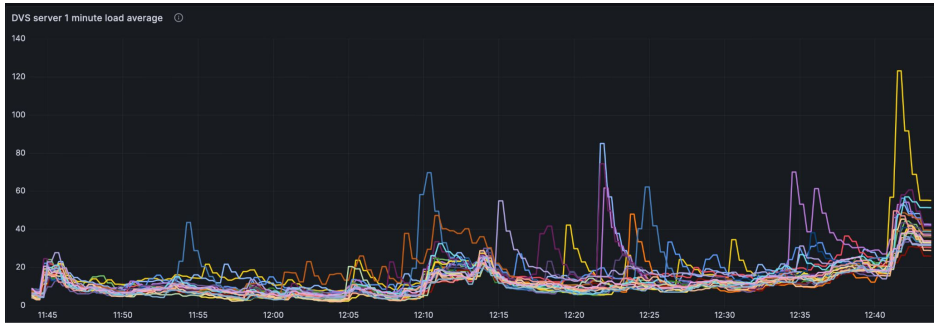




DVS with read-only mount (/global/common)

- Eg: a 100-node job using conda environment in /global/common
  - 12,800 processes are spread across 24 DVS servers
  - `/global/homes/e/elvis/.conda` gets fetched once and cached
  - The load on the DVS servers stays low
  - The load on the filesystem stays low
  - The job continues almost immediately

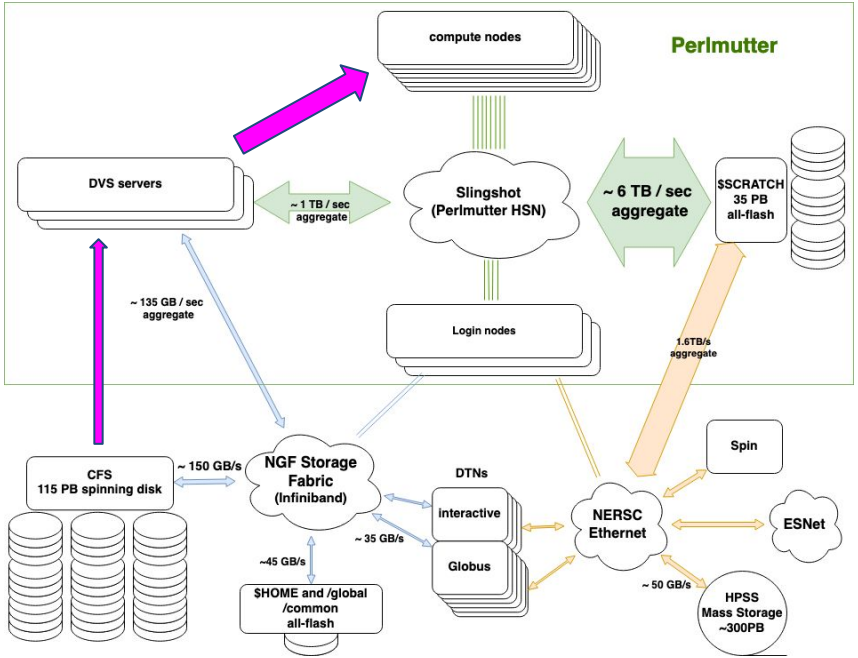




note that this y-axis goes 1/10 as high!

Read-only mount of CFS

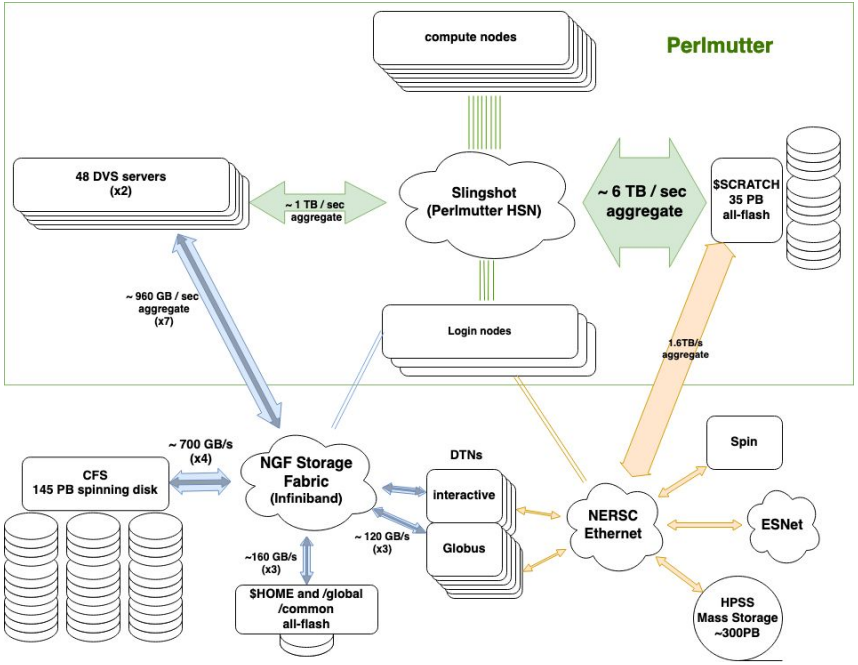
- CFS also has a read-only mount point on Perlmutter: `/dvs_ro/cfs/cdirs/` (the RW one is `/global/cfs/cdirs`)
- $SCRATCH is still faster .. BUT if your input data is:
  - too big for $SCRATCH, and/or:
  - used by multiple people in your project
- .. then you might benefit from reading it directly from `/dvs_ro/cfs/cdirs`



Sneak peek: Upgrades to CFS and NGF Storage Fabric

- We're working on updates to the storage network infrastructure, and to CFS
  - More CFS cabinets = more capacity + more bandwidth
  - More DVS servers = more bandwidth
  - Faster network fabric

Coming soon!

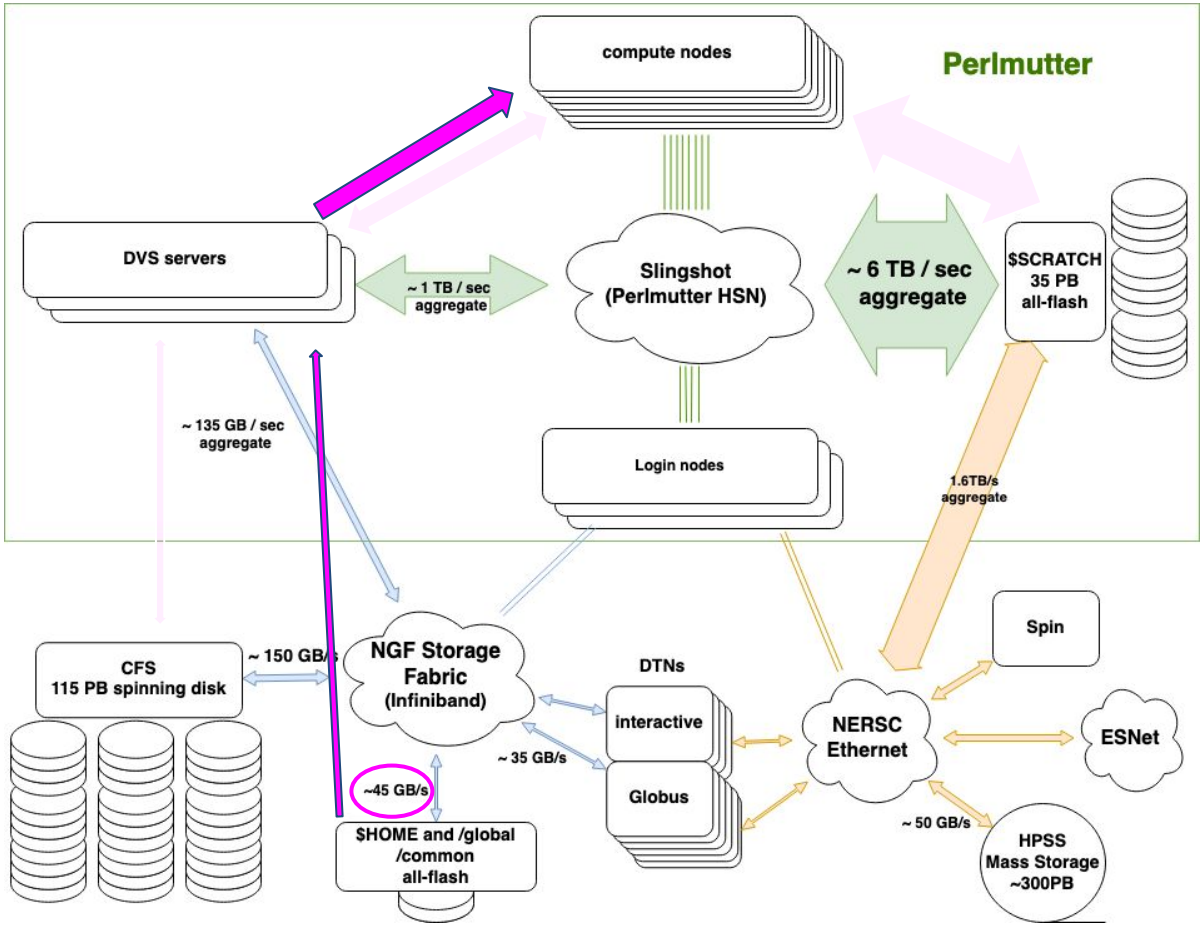


TL; DR

- (Almost) All job I/O should happen on $SCRATCH
- Don't do I/O at scale over DVS
- CFS is best for actively-used data (but not source code)
- Put conda environments in /global/common/software (or better still, a container)
- Not using it for a while? Bundle it into big-ish (100GB->2TB) tar files and store on HPSS
- $HOME is good for scripts and source code
- Use Globus for moving large chunks of data around (even within NERSC)
- Have an off-NERSC copy of everything important!

Software on /global/common

- Small block size, all-flash, **mounted read-only on compute nodes** (read-write on logins)
- Benefits from **DVS caching**, multiple DVS nodes



/global/common/software

- Especially good for python / conda environments!

  ```
  conda create --prefix /global/common/software/myproject/myenv
  ```

  https://docs.nersc.gov/development/languages/python/nersc-python/#moving-your-conda-setup-to-globalcommonsoftware

- Python startup involves loading lots of modules, which involves looking in all of the directories in LD_LIBRARY_PATH - *lots* of disk access
- The read-only DVS mount of /global/common/software mitigates most of this
- Related tip:
  - Don't load a conda environment at login! (via .bashrc/.bash_profile). It will be loaded for every Slurm job too.

Software in containers

- NERSC supports Shifter and Podman (newer, solves some limitations of Shifter). Both provide similar functionality to Docker
  - https://docs.nersc.gov/development/podman-hpc/overview/
  - https://docs.nersc.gov/development/shifter/how-to-use/
- How do they help?
  - Software is *in the container* - vastly reduces load on filesystem
  - Also: consistent environment each run, even if Perlmutter software stack changes -> reproducibility benefits

Podman

Shifter

Number of nodes	`$HOME`	`/global/common/software`	Shifter
1	0m4.256s	0m3.894s	0m3.998s
10	0m10.025s	0m4.891s	0m4.274s
100	0m30.790s	0m17.392s	0m7.098s
500	4m7.673s	0m48.916s	0m14.193s

Python benchmark compared by filesystem or container, over increasing node count

TL; DR

- (Almost) All job I/O should happen on $SCRATCH
- Don't do I/O at scale over DVS
- CFS is best for actively-used data (but not source code)
- Put conda environments in /global/common/software (or better still, a container)
- Not using it for a while? Bundle it into big-ish (100GB->2TB) tar files and store on HPSS
- $HOME is good for scripts and source code
- Use Globus for moving large chunks of data around (even within NERSC)
- Have an off-NERSC copy of everything important!

HPSS - Tape-based Mass Storage

- Tape! Reliable long-term storage
- Really huge - 300PB and growing
- Fast ingest (~ 50GB/s)
  - Data first hits a spinning disk cache and gets migrated to tapes, cache is sized for several weeks of retention

- Tape! Retrieval can take a long time
  - (Robot needs to fetch tape, insert into drive, scroll to where your data starts, then it can start reading)
- Not suitable for small files
  - 100GB -> 2TB per file is best
  - Use tar or htar to bundle files

https://docs.nersc.gov/filesystems/archive/#htar

- Use HPSS for important data you are not actively using
- Retrieval order matters
  - https://docs.nersc.gov/filesystems/archive/#order-large-retrievals

TL; DR

- (Almost) All job I/O should happen on $SCRATCH
- Don't do I/O at scale over DVS
- CFS is best for actively-used data (but not source code)
- Put conda environments in /global/common/software (or better still, a container)
- Not using it for a while? Bundle it into big-ish (100GB->2TB) tar files and store on HPSS
- $HOME is good for scripts and source code
- Use Globus for moving large chunks of data around (even within NERSC)
- Have an off-NERSC copy of everything important!

$HOME

- All-flash filesystem (fast access)
- Small block size (good for small files - source code, scripts, etc)
- Backed up
  - Daily snapshots
  - e.g. my homedir is at `/global/homes/e/elvis/.snapshots/2024-02-19`
  - (note: you can't see `.snapshots` with `ls`, but you can `cd` to it)
  - Also backed up to tape approximately monthly
- Not for large I/O (relatively lower bandwidth)
- Small - not intended for data storage
- Not suitable for running jobs against
- Avoid making your conda environments here, particularly if you will use them in compute jobs!

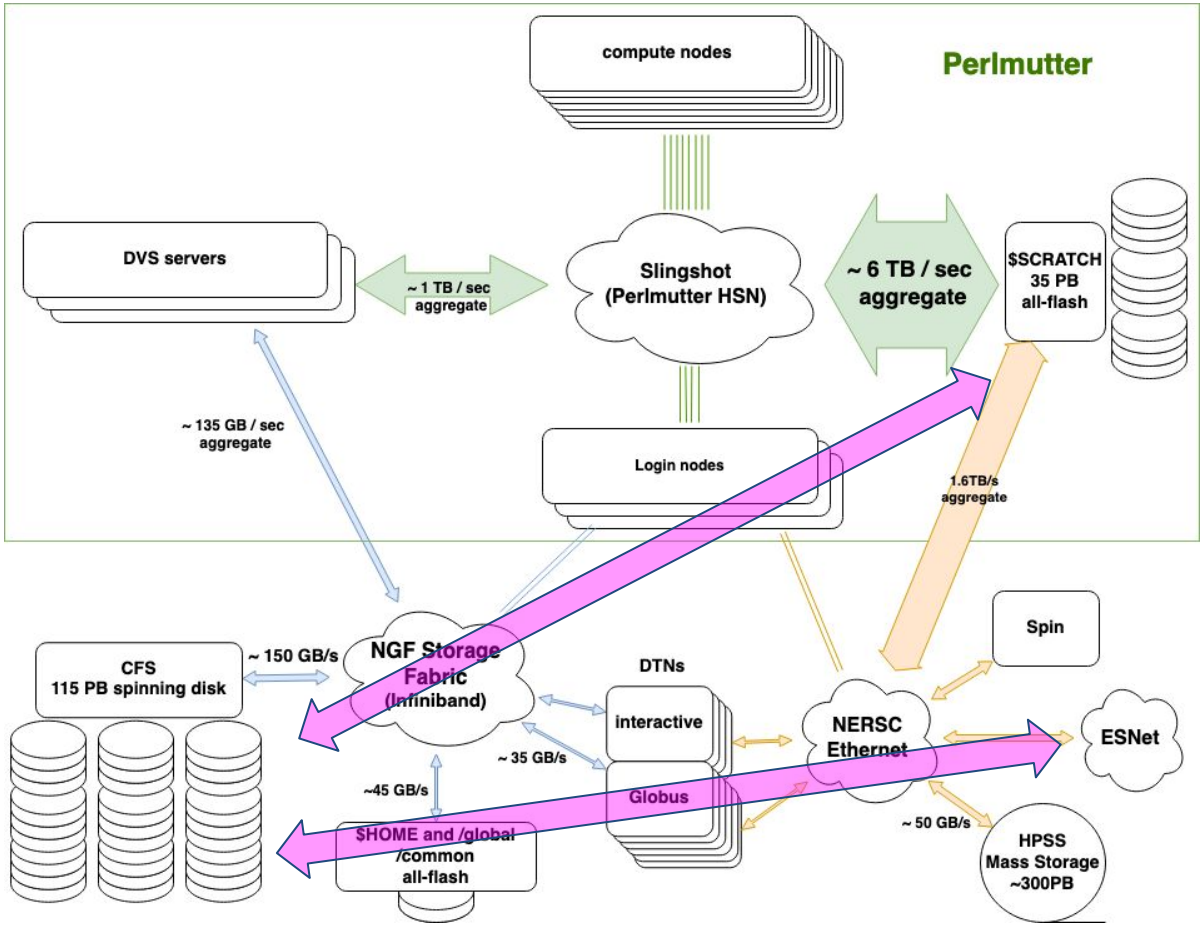
TL; DR

- (Almost) All job I/O should happen on $SCRATCH
- Don't do I/O at scale over DVS
- CFS is best for actively-used data (but not source code)
- Put conda environments in /global/common/software (or better still, a container)
- Not using it for a while? Bundle it into big-ish (100GB->2TB) tar files and store on HPSS
- $HOME is good for scripts and source code
- Use Globus for moving large chunks of data around (even within NERSC)
- Have an off-NERSC copy of everything important!

Globus for data movement

- Managed transfers (uses DTNs and login nodes)
  - survives disconnect
- Multiple streams
  - higher bandwidth

https://docs.nersc.gov/services/globus/



TL; DR

- (Almost) All job I/O should happen on $SCRATCH
- Don't do I/O at scale over DVS
- CFS is best for actively-used data (but not source code)
- Put conda environments in /global/common/software (or better still, a container)
- Not using it for a while? Bundle it into big-ish (100GB->2TB) tar files and store on HPSS
- $HOME is good for scripts and source code
- Use Globus for moving large chunks of data around (even within NERSC)
- Have an off-NERSC copy of everything important!

How to not lose data

- $SCRATCH
  - For I/O, not storage! We actively purge older, not-recently-used data. Copy your results from $SCRATCH to, eg HPSS
- CFS
  - For storage, not I/O. Nightly snapshots, kept 7 days ("accidental deletion protection"). Robust system, but not backed up - keep a second copy somewhere else
  - https://docs.nersc.gov/policies/data-policy/policy/#community-file-system
- $HOME
  - For small-but-important things. Daily snapshots, monthly backups. No off-site backup - keep a copy of critical data at another site
- HPSS
  - Tape - good place to store important data. We only keep a single copy - for critical data, make a second copy, either in HPSS or (better) offsite
  - https://docs.nersc.gov/policies/data-policy/policy/#backup_4

Choosing the right storage for your data

- (Almost) All job I/O should happen on $SCRATCH
- Don't do I/O at scale over DVS
- CFS is best for actively-used data (but not source code)
- Put conda environments in /global/common/software (or better still, a container)
- Not using it for a while? Bundle it into big-ish (100GB->2TB) tar files and store on HPSS
- $HOME is good for scripts and source code
- Use Globus for moving large chunks of data around (even within NERSC)
- Have an off-NERSC copy of everything important!