Balanced File System Migration

LUG 2023 / San Diego

Cameron Harr
Lustre Ops

May 3, 2023

LLNL-PRES-847387
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

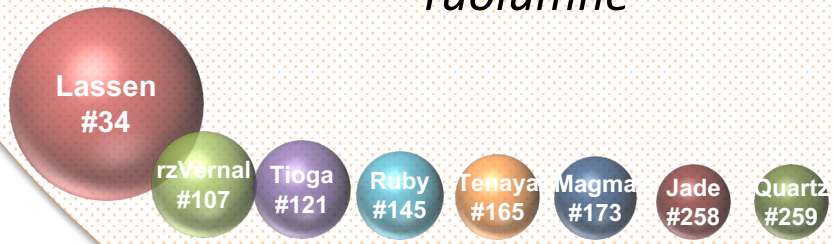
Lawrence Livermore National Laboratory

Agenda

- Livermore Computing (LC)
- Why Migrate?
- Migration Strategies
- My Solution

Livermore Computing (LC)

- 3+ Data centers
  - ~240+ “clusters” *
  - TSF: 85MW
- 5+ Network *Centers*
  - *GZ, CZ, RZ, SCF, SNSI*
- *Upcoming*
  - *El Capitan (2+ EF)*
  - *Tuolumne*



* 2+ node Compute | Infrastructure | T&D

Lustre @ LC (2023)

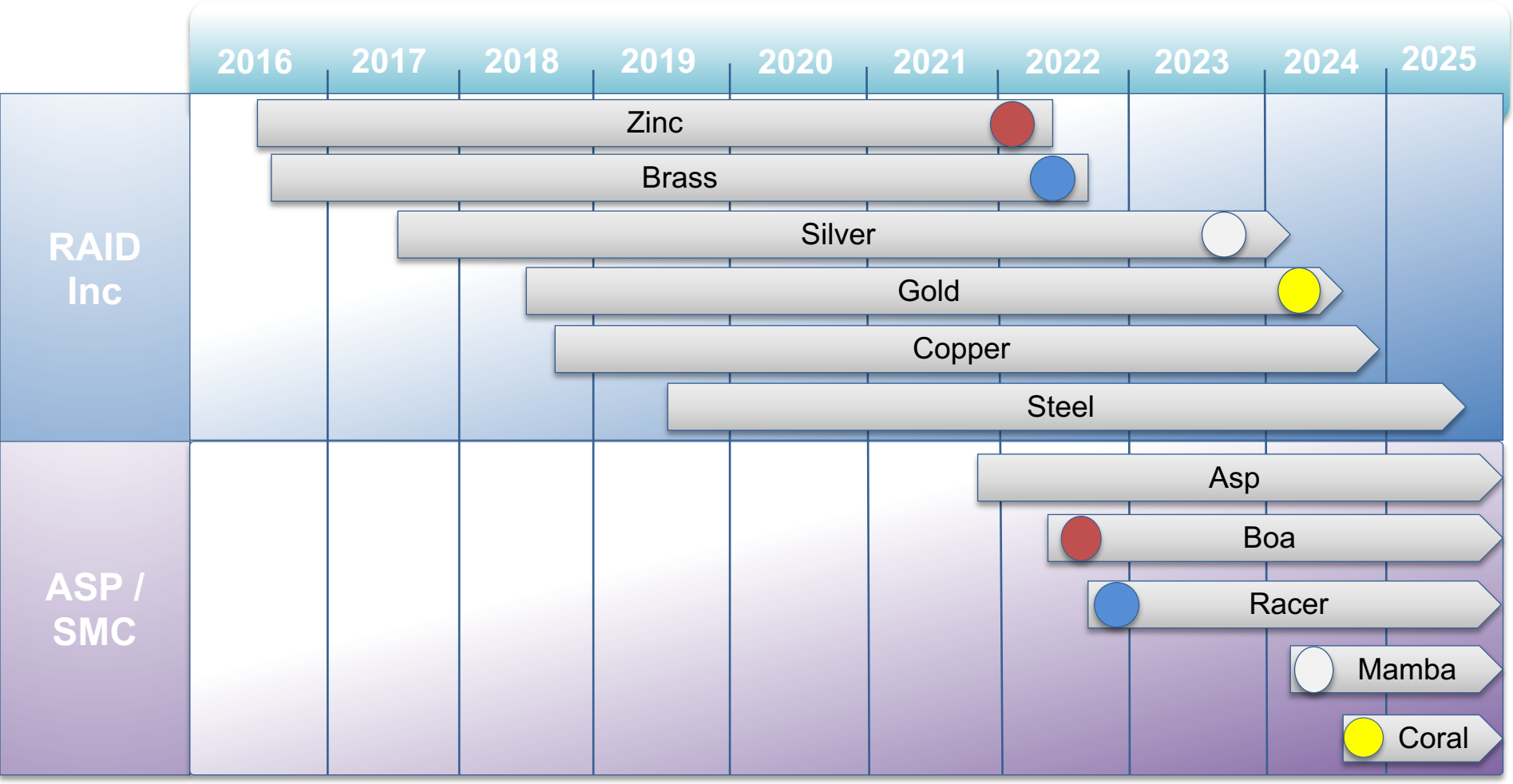
- Production Lustre
  - ~100 PiB (useable)
  - 7 production file systems
    - 4x Lustre 2.12 (RAID Inc.)
      - Due for replacement over next 2 years
    - 3x Lustre 2.14 (Supermicro)
    - Across 4 “Centers”
    - Production == 24/7 support
  - Plan to coalesce on Lustre 2.15
    - LNet router issues (LU-16244, …)
  - User quotas
    - 3 Tiers
      - Tier 1: 20TB / 1M
      - Tier 2: 75TB / 25M
      - Tier 3: Per user justification (time-limited)

filesystem	Used Space in TB	Percent Full	Millions of files	Average File Size in KB
/p/czlustre1	7466	45%	944	8489
/p/czlustre2	8740	33%	988	9502
/p/czlustre3	1446	17%	588	2639
/p/lustre1	6169	28%	1169	5664

No More Scratch

- LC *un-scratched* Lustre in 2018
  - No purges
  - User data is persistent
- Data self-managed via quotas
  - Initial fear → Happiness
  - Much lower capacity/inode utilization
- But…
  - LC Staff on the hook to migrate *ALL* user data
    - Data doesn't "age out"
    - Telling users to migrate data violates the class of service

LC Lustre Lifecycle



- Procured through multi-year vendor-specific contract

ASP (Adaptable Storage Platform)

- 5-yr contract w/ SMC in 2020
- Provides standard “Modules” with which to construct storage systems
  - *FBB* (Flash Building Block)
  - *DBB* (Disk Building Block)
  - Mgmt
  - Infrastructure
    - Switch
    - Rack/PDU
  - Options
- Customer now selects quantity and type of modules to achieve desired capacity and performance
  - Ex: Lustre®
    - FBBs for MDS
    - DBBs for OSS
- Allows purchase vehicle for many storage configurations
  - Lustre
  - Globus
  - NAS
  - HPSS control, cache
  - Log aggregation / ELK
- ASP concept unifies:
  - Parts cache
  - Staff expertise
  - Vendor support
  - Procurement overhead

Migration Strategies

- Our needs
  - Changing numbers of targets
  - Balanced Metadata
  - Timely completion
  - Data Integrity Guarantee
- Lots of potential solutions…
  1. ZFS send-receive
  2. Pools + LFS migrate
  3. FLR
  4. Misc
     1. HDD swaps
     2. Open source solutions
  5. Commercial solutions

Migration Solutions Explored – ZFS Send/Recv

- Benefits
  - Good performance
    - Sent ~ 1.2PB in 38 hr (32 TB/hr)
      - Rate can depend on # objects
      - Over QDR IB
      - Thanks to Olaf Faaland for data
  - Incremental updates
  - Doesn't need colocation
- Drawbacks
  - # of dest targets == # src targets
    - We wanted 80 OSTs -> 8 OSTs
  - Requires (slow) post-sync migration to consolidate OSTs or balance MDTs
  - Requires surgery post-transfer
    - Change NIDs
    - Change F/s name
- `continue;`

1. Create a zpool to receive data:

```
[root@zwicky17:~]# zpool create zwicky17 /dev/mapper/zwicky17_1
/dev/mapper/zwicky17_2 /dev/mapper/zwicky17_3
[root@zwicky17:~]# zpool list
NAME    SIZE ALLOC FREE CKPOINT EXPANDSZ FRAG CAP DEDUP HEALTH ALTROOT
zwicky17 65.5T 130K 65.5T   -        -      0%   0% 1.00x ONLINE –
```

2. Create Snap on source and send

```
[root@zwicky3:~]# zfs snapshot zwicky3/ost1@12192019
[root@zwicky3:~]# zfs list -t all -r zwicky3
NAME                                USED   AVAIL  REFER  MOUNTPOINT
zwicky3                             13.8G  63.4T  24K    /zwicky3
zwicky3/ost1                        13.7G  63.4T  13.7G
/zwicky3/ost1 zwicky3/ost1@12192019                  0B      -
13.7G  -
[root@zwicky3:~]# zfs send –R zwicky3/ost1@12192019 \
       | rsh zwicky17 "zfs recv zwicky17/z3ost1"
```

3. Monitor progress

```
[root@zwicky17:~]# zfs list
NAME            USED  AVAIL  REFER  MOUNTPOINT
zwicky17        412M  63.4T    24K  /zwicky17
zwicky17/z3ost1  412M  63.4T  412M  /zwicky17/z3ost1
...
[root@zwicky17:~]# zfs list –H
zwicky17        13.7G  63.4T    24K  /zwicky17
zwicky17/z3ost1  13.7G  63.4T 13.7G  /zwicky17/z3ost1
```

Migration Solutions Explored – lfs_migrate

- Benefits
  - Built into Lustre
  - Restripes files
- Drawbacks
  - Slow
    - ~20 items/s
    - ~40 items/s (`lfs find | lfs_migrate`)
    - Max ~400 0B files/s @ 32 procs
  - Switch to `lfs migrate` for open files
  - Couldn't migrate MDT0
    - Potentially could use ZFS Send/Recv
  - Pool migrate Undependable
    - Some files silently did not migrate
    - Would try to migrate to *old* pools
      - Will prompt *sometimes* to set `max_create_count=0`
  - Removal of targets disruptive
    - New `lctl del_ost` an improvement!
- `continue;`

lfs_migrate with pools

- 2 MDS/MDT (iron[1,11]) ## Single digits represent old MDTs/OSTs
  6 OSS/OST (iron[2-4,12-14]
  2 Clients (iron[5,15])
- Create Pools and add OSTs

```
# lctl pool_new charrfs.old
# lctl pool_new charrfs.new
# lctl pool_add charrfs.old OST[2-4]
# lctl pool_add charrfs.new OST[c-e]
```

- Migrate directory to new pool

```
# time lfs_migrate -p charrfs.new ./
lfs_migrate is currently NOT SAFE for moving in-use files.
Use it only when you are sure migrated files are unused.
If emptying an OST that is active on the MDS, new files may
use it.  To stop allocating any new objects on OSTNNNN run:
  lctl set_param osp.<fsname>-OSTNNNN*.max_create_count=0'
on each MDS using the OST(s) being emptied.
Continue? (y/n)
.//migrate.out: done
.//ost32: done
.//ost31: done
.//ost34: done
.//ost33: done
.//mdt0001_test/file4897: done
.//mdt0001_test/file8378: done
…
.//mdt0001_test/file1924: done

real    9m46.764s
```

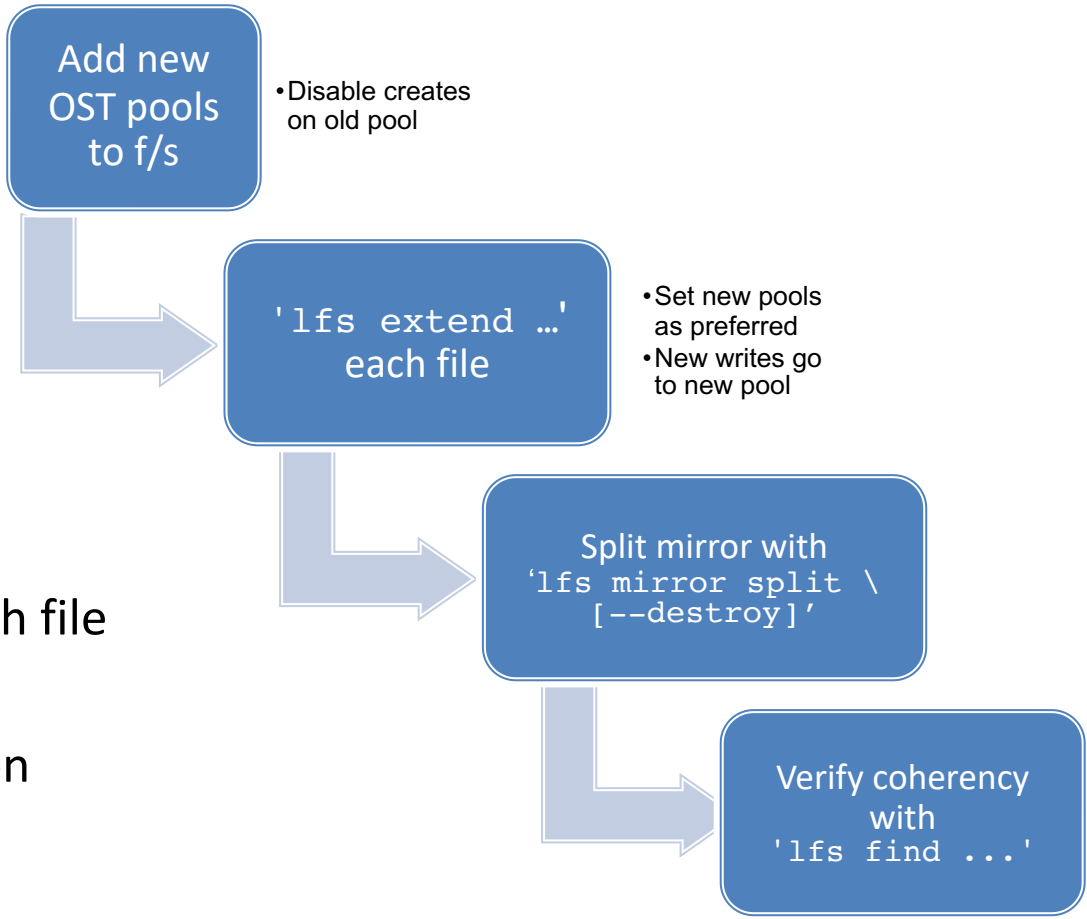
- Use lfs find to send files to lfs_migrate (after restoring all targets from snapshot)

```
# time lfs find . -O 2,3,4 | xargs lfs migrate -p charrfs.new
…
real    4m7.579s
```

- After running 'lfs getstripe * | grep stripe_offset' to verify, some files didn't migrate
  - Perhaps due to xargs?
- Without setting `max_create_count=0` to zero, some files were migrated to old OSTs
  - Should be smart enough to not use OSTs you're migrating off of

Migration Solutions Explored – File Level Redundancy

- Benefits
  - Built into Lustre
  - Keep copy of original data
  - Works with active f/s
  - Restripes files
  - Doesn't re-migrate old data
- Drawbacks
  - Needs same # of targets
  - Need to manually extend each file
  - Labor-intensive
  - Doesn't cover MDT0 migration
- `continue;`



Migration Solutions Explored – Misc

- Open source solutions
  - Migratefs with overlayfs (Thiell, LUG ‘19)
    - Data doesn’t “age-out” so all data must be copied
  - ldsync/ldmigrate
    - Lustre Replication and Migration Tool (Ihara, DDN, LAD ’16)
    - Couldn’t find code base
- HDD swaps
  - Feasible for MDTs (NVMe)
  - Slow
  - Human Labor-intensive
    - Error-prone
- `continue;`

Migration Solutions Explored - Commercial

- Atempo Miria
  - Changelog allows incremental syncs ✚
  - No support for DNE ▬
  - Serial f/s scan ▬
  - } Supposedly fixed
  - Required Source and Dest to look similar. ▬
- Starfish (sf-sync-and-verify)
  - Already have Starfish implemented
    - Great support team ✚
  - Allows incremental syncs ✚
  - No MDT balancing ▬
  - Limited b/w (our implementation) ▬
  - Already had my 'fssync' going ▬
- `continue;`

End Solution: lustre_fssync

Requirement	End Result
Allow differing numbers of nodes/targets	Doesn't care about #s of targets
Balanced metadata / DNE2	Autodetects and balances inode load on MDTs
Timely completion / Parallel transfers	Uses MPI File Utils with configurable # of jobs and CPUs
Data integrity verification	Allows for content comparison
Work with PFL, DoM	Sets up PFL/DoM if desired
Not labor-intensive	Runs full sync from one command

lustre_fssync

```
# /admin/scripts/lustre/lustre_fssync [-h] -s <source f/s> -d <dest f/s> [-b <#files>] [-c <#TB>] [-C] \
        [-D] [-e <excludes>] [-i <#inodes>] [-m <#jobs>] [-M] [-n <#CPUs>] [-N] [-p <part>] [-v] [-V]

        -h: This menu.
        -s: The mount name of the source file system, e.g. lustre1.
        -d: The mount name of the destination file system, e.g. lustre4.
        -b: # of files in sync batch. Default=50000
        -c: Threshold of capacity in TB defining a heavy user. Default=20
        -C: Compare file contents between Src & Dest. This will take longer.
        -D: Delete extraneous files from destination.
        -e: File name with list of users to exclude
        -i: Threshold of number of inodes defining a heavy user. Default=1000000.
        -m: Maximum number of sync jobs. Default=<# of MDTs>.
        -M: Only report the calculated MDT balancing. No syncing is done.
        -n: Number of CPUs per sync job. Default=64.
        -N: Do not enable PFL and DoM. Default enables them on root of destination f/s.
        -p: Slurm Partition to submit to. Default is 'putil'.
        -v: Verbose -- display more messages
        -V: Debug -- display even more messages
```

- Dependencies
  - MPI File Utils 0.11.1+
  - Lustre user quotas
  - SLURM Resource Manager, with partition that accepts jobs from `root`
- Will be part of lustre-tools-llnl package when fully approved: https://github.com/LLNL/lustre-tools-llnl

lustre_fssync - Process

- First pass
  1. Set PFL/DoM layout
  2. Find heaviest users (inode or capacity) and balance across MDTs
  3. Create and balance remaining user & group directories
  4. Submit one batch job per directory (user)
  5. Script spins until all jobs complete
- Repeat as desired until final sync
- Optional: Add -C for Byte-level comparison (vs. mtime, size)
  - Takes a long time, but ensures integrity
- Final sync
  - Put src f/s in read-only for weekend and run final sync, then re-mount

lustre_fssync - Example

- Source f/s: /p/lustre1
  - Lustre 2.12
  - ZFS 0.7 w/ RAIDZ2
  - 4x MDS nodes
    - 1 MDT/MDS
  - 18x OSS nodes
    - 1 OST/OSS
  - 5.4 / 7.5 PiB
    - (compressed @ ~1.7x)
  - ~1.02B files
  - >2100 user directories
- Dest. f/s: /p/asplustre1
  - Lustre 2.14
  - ZFS 2.1 w/ dRAID2
  - 8x MDS nodes
    - 1 MDT/MDS
  - 20x OSS nodes
    - 1 OST/OSS
  - 20.2 PiB

lustre_fssync – Early Sync Example (cont.)

```
CMD: /admin/scripts/lustre_fssync -v -s /p/lustre1/ -d /p/asplustre1/ -m 32 -n 72 -p pfstest
Preparing to sync lustre1 to asplustre1 via the pfstest partition...
  * Collecting Project/Group directories...
  * Finding heavy users...
  * Setting DoM && PFL on /p/asplustre1/
  * Creating user directories on 8 MDTs...
    - Balancing heavy users across MDTs...
      - MDT 0: 69678425 inodes
      - MDT 1: 69635523 inodes
      - MDT 2: 69761822 inodes
      - MDT 3: 69678081 inodes
      - MDT 4: 69675530 inodes
      - MDT 5: 69673621 inodes
      - MDT 6: 69666277 inodes
      - MDT 7: 69640146 inodes
    - Spreading project dirs across MDTs...
    - Spreading remaining users across MDTs...
      ... userA
      ... userB
      ... userC
           . . .
 * Submitting user sync jobs. This will spin until all jobs are complete...
    - Submitted job for userA: 8215651
       |-- sbatch -p pfstest -n 144  fssync_21050/userA.dsync.sbatch
    - Submitted job for userB: 8215652
       |-- sbatch -p pfstest -n 144  fssync_21050/userB.dsync.sbatch
    - Submitted job for userC: 8215653
       |-- sbatch -p pfstest -n 144  fssync_21050/userC.dsync.sbatch
           . . .
    - Submitted job for userNNNN: 8227240
       |-- sbatch -p pfstest -n 72  fssync_21050/userN.dsync.sbatch
Sync complete!
Waiting for sync jobs to finish...
Total runtime: 6-18:27:14
```

lustre_fssync – Final Sync Example (cont.)

```
CMD: /admin/scripts/lustre_fssync -e excludes -D -v -s /p/lustre1/ -d /p/asplustre1/ -m 32 -n 72 -p pfstest
Preparing to sync lustre1 to asplustre1 via the pfstest partition...
  * Collecting Project/Group directories...
  * Finding heavy users...
  * Setting DoM && PFL on /p/asplustre1/
  * Creating user directories on 8 MDTs...
    - Balancing heavy users across MDTs...
      - MDT 0: 70672987 inodes
      - MDT 1: 70918099 inodes
      - MDT 2: 70625344 inodes
      - MDT 3: 71098256 inodes
      - MDT 4: 70647964 inodes
      - MDT 5: 70669558 inodes
      - MDT 6: 70955552 inodes
      - MDT 7: 70671313 inodes
    - Spreading project dirs across MDTs...
    - Spreading remaining users across MDTs...
    The following user directories already existed: /p/asplustre1/userA /p/asplustre1/userB …
            . . .
 * Submitting user sync jobs. This will spin until all jobs are complete...
    - Submitted job for userA: 8409637
       |-- sbatch -p pfstest -n 144  fssync_23685/userA.dsync.sbatch
    - Submitted job for userB: 8409638
       |-- sbatch -p pfstest -n 144  fssync_23685/userB.dsync.sbatch
    - Submitted job for userC: 8409639
       |-- sbatch -p pfstest -n 144  fssync_23685/userC.dsync.sbatch
            . . .
    - Submitted job for <userNNN>: 8411693
       |-- sbatch -p pfstest -n 72  fssync_23685/userN.dsync.sbatch
Sync complete!
Waiting for sync jobs to finish...
Total runtime: 1-17:3:56
```

Lessons Learned

- Throughput on B/W between Src/Dest
  - # Routers matter
  - We're were constrained by inter-bldg. routers
- Large users can throw off time and usage balance
  - Stuck with small number of still syncing users at switch-over time
- Human supervision strongly recommended
  - Spot stalled or failed jobs and restart
- Didn't need byte-level final sync
  - Perform earlier on and trust `dsync`
- Patience still required

https://giphy.com/gifs/kim-novak-tXL4FHPSnVJ0A

Thank you!

Lawrence Livermore
National Laboratory