migratefs: overlay filesystem for transparent, distributed migration of active data

Stéphane Thiell
Stanford Research Computing Center (SRCC)
sthiell@stanford.edu
Contents

Our Lustre ecosystem
HPC filesystem lifecycle problem
Introduction to migratefs
Our Lustre ecosystem
Lustre ecosystem at the SRCC (May 2019)
Sherlock at Stanford

**Condo** cluster constantly evolving to support research

**Numbers**
- ~4,000 users in ~670 different research groups
- Compute nodes have a 4-year lifecycle
- Currently **1,371 nodes**
- 26,040 CPU cores; 716 GPUs; **1.876 PFlops**
- Two separate Infiniband fabrics: **FDR, EDR**

**Lustre 2.12** with **DNE+DoM+PFL** since February 2019

More info @ [https://www.sherlock.stanford.edu/](https://www.sherlock.stanford.edu/)
HPC filesystem lifecycle problem
Objectives

Replace Sherlock’s scratch filesystem (Regal with Fir)

▸ Data-integrity risk: Regal’s hardware is now obsolete
▸ Improve performance with small files
▸ Better network bandwidth for large streaming I/O

Migration should be transparent for the users

▸ as in no change in workflow and scripts
▸ reasonable performance tradeoff is acceptable
▸ no prolonged downtime: use regular cluster maintenance windows
Usual methods of data migration

Data copy: rsync, fpsync, lustre-data-mover, etc.
- Requires multiple passes and a long downtime
- Why copy data that are going to be purged anyway?

User-led data migration
- Provide a new mount point to users and let them handle the data migration with a deadline
- Occasional users will miss the deadline
- Too disruptive for our users
Usual methods of data migration (cont’d)

In-place expansion (Lustre-specific method)

▸ Possible scenario:
  ▶ upgrade old system
  ▶ add new MDS, OSS and storage hardware
  ▶ backup/restore MDT(s) to new hardware
  ▶ use lfs migrate to move file objects to new OSTs
  ▶ decommission the empty, old OSTs...

▸ Error prone, especially for network upgrade

None of the usual methods is satisfactory :-(

migratefs
migratefs - principle

Node-local overlay filesystem in user space

Merge multiple directories/filesystems (layers) and seamlessly migrate data to upper layer when needed

Dispatch I/O syscalls to the right underlying layer

Originally forked from fuse-overlayfs

- https://github.com/containers/fuse-overlayfs
migratefs - easy to deploy and use

Launch daemon to merge /regal and /fir into /scratch

```
# migratefs -o lowerdir=/regal,upperdir=/fir /scratch
```

Track open files with:

```
# ls -l /proc/$(pidof migratefs)/fd
```

Systemd unit file available for automatic start
migratefs - copyup!

open() with write intent triggers a copyup operation, migrating the file to the upper layer

data is copied to a temporary file which is renamed when the copy is done

copyup is performed as root to copy all file attributes and parent directory path to the upper layer
migratefs - lifecycle example (scratch)

Short maintenance #1

Short maintenance #2

data migration/purge time

full filesystem migration time
migratefs - features

Uses FUSE 3.2 low-level API with no caching
  ▶ with added multithreading support

Metadata operations
  ▶ avoid copyup, lower layer writable for mkdir, rename

Permission handling
  ▶ runs as root and then drops privilege

Inode numbers
  ▶ migratefs encodes the layer ID into inode numbers

Monitoring
  ▶ distributed logging can be aggregated by Splunk or similar
Daily volume of data copied up during the first weeks in production < 3 TiB/day and decreasing

Only modified files are copied up!
migratefs & purge in action
migratefs on Sherlock – changelog

Started production on February 5, 2019
  ▶ Singlethreaded

migratefs 0.3 released on February 27, 2019
  ▶ Multithreading support
  ▶ Travis-CI with fstest, stress-ng and custom tests

migratefs 0.4 released on March 11, 2019
  ▶ Bug fixes

migratefs 0.5 released on April 21, 2019
  ▶ performance improvements
migratefs - latest release 0.5.4

Skip copying data up on open(0_TRUNC)

Added a boolean flag (multilayer) to cached inode information
  ▶ Lookup speedup for directories that only exist in the upper

Added st_nlink caching for multilayer directories
  ▶ Improved performance with a directory having more than 2 million aggregated entries...
migratefs & Lustre, room for improvement

Linux kernel 4.20+
- maximum I/O size for FUSE increased from 128k to 1M
- come on, Red Hat!

Support for renameat2() in Lustre
- Potential race/retry on parallel copyup would be avoided with renameat2(RENAME_NOREPLACE)
  - LU-12272: Support renameat2() with RENAME_NOREPLACE flag
migratefs - conclusion

All the cluster nodes contribute to the data migration

Only *active* data are copied to the new filesystem

No change in the user environment
  ▶ same paths, no LD_PRELOAD, etc.

Performance tradeoff during data migration
  ▶ upper layer can still be accessed directly if needed
migratefs - download & contribute

https://github.com/stanford-rc/fuse-migratefs
Extra slides
**migratefs - metadata operations**

Most metadata ops don’t trigger a *copyup by default*
- *copyup rules should match filesystem purge policies*

Unlike *fuse-overlayfs*, with *migratefs*, the lower layer(s) must be writable
- `rename()` may create missing directories in lower layer
- Lustre disk quota can be set to 0 to avoid direct writes from users to the lower layer
migratefs - FUSE (Filesystem in Userspace)

migratefs requires FUSE 3.2

- Low level FUSE 3 API (like fuse-overlayfs)
- Not easy to find a package for CentOS 7 providing libfuse3
  - https://github.com/stanford-rc/fuse3-centos7

migratefs does not use FUSE name lookup caching

migratefs has multithreading support

- Strong requirement for decent performance over Lustre
- Supported by FUSE but not by fuse-overlayfs
- Can be disabled with -o mt=0
migratefs - permission handling

migratefs daemon runs as root

- drops privilege to run as the effective ID of the calling user
  - similar to LANL’s MarFS
- root is needed for copyup to copy permissions
- Secondary groups are supported

Do not use FUSE’s default_permissions

- POSIX ACLs are not supported by FUSE when default_permissions is enabled!
- rely on the underlying filesystem for permission and ACL checking, always under the context of the user
**migratefs** - inode numbers

*fuse-overlayfs* assumes that `/lower` and `/upper` are part of the same filesystem (common for containers)

- thus inode numbers from the upper and lower layers are just exposed as is, easy!

*migratefs* encodes the layer ID in the high 4 bits of the inode #

- works with all filesystems
- inode numbers in Lustre are flattened FID (seq, oid)

\[
\text{ino} = (\text{seq} \ll 24) + ((\text{seq} \gg 24) \& 0xffffffff0000ULL) + \text{fid_oid}(	ext{fid});
\]

(FID's seq is 64-bit)  
(FID's oid is 32-bit)

Any better idea?
migratefs - monitoring

Each migratefs daemon prints interesting logs

- *copyup* operation results
- other unexpected errors

These distributed logs are sent to Stanford’s Splunk

Logs use clear key-value pairs for use with Splunk’s automatic field extraction:

```
May 09 14:43:23 sh-ln05.stanford.edu migratefs[47585]: version=0.5.4
copyup=succes uid=315672 st_uid=315672 written=1130471060 truncate=false
path=users/user1/WACC/M/WACC/CTL_122.cam.h1.0100-01-01-00000.nc
```
migratefs on Sherlock – changelog 1/3

Started production on February 5, 2019

▸ Only singlethreaded versions at first (0.1.x-0.2.x)
▸ Improved error handling for cluster-awareness
  ▶ For example: handle ENOENT on mkdirat()
▸ Fixed a deadlock due to recursion because of calling
  \{get/set/remove\}xattr instead of l\{get/set/remove\}xattr
▸ Disabled FUSE’s default_permissions to support Lustre’s
  POSIX ACLs
▸ Adjusted OOM score in systemd unit file to avoid killing of the
  migratefs daemon (because of the user context switching)
migratefs on Sherlock – changelog 2/3

Version 0.3 released on February 27, 2019
▶ First multithreaded version
▶ Use direct syscall for per-thread setresuid()
▶ Got rid of umask() (not thread safe) and honor umask at open() instead from fuse_ctx
▶ Set up Travis-CI with fstest, stress-ng and custom tests

Version 0.4 released on March 11, 2019
▶ improve FUSE inode lookup count handling
▶ also fixed defects and race conditions in 0.4.x
migratefs on Sherlock – changelog 3/3

Version 0.5 released on April 21, 2019

▶ performance improvements (st_nlink, multilayer flag)
▶ now encoding layer ID in inode numbers
▶ fixed an issue reported by a user with “du” when inodes were not refreshed correctly (now added as custom test)

    du: WARNING: Circular directory structure.
    This almost certainly means that you have a corrupted file system.
    NOTIFY YOUR SYSTEM MANAGER.
    The following directory is part of the cycle:
    ‘./scripts’