

Exploring Lustre's Erasure code feature

Current implementation, Sept 2025

James Simmons

Storage Systems Engineer

Oak Ridge National Laboratory

ORNL is managed by UT-Battelle LLC for the US Department of Energy

What's happen in the last year

- Four rewrites during the 2.17 development cycle
 - Heavy testing of Adam Disney / Bobi Jam version
 - Component having both parity and data OST objects
 - N data components to 1 parity mapping model
 - Current version discussed here
 - Share for feedback with the community
- Why is ORNL interested
 - Systems are always 85% full so no wide scale mirroring
 - When things break it tends to be many things
 - Team of code developers that work with users

Fundamentals for PFL parity implementation

- Extent expectation changed

- Extent align is not strict



- 2nd eof and 0 is considered a misalignment
- Code is such that parity can be placed *almost* anywhere in layout

- Data I/O and DoM must be first component in any mirror

- Parity settings are different then I/O components.

- Parity components are setup with comp IDs instead of extents
 - Allows parity for non I/O components (foreign and parity)
 - N to 1 parity component for non-IO components
 - Dependency between parity and components being parity checked
 - Foreign and data I/O together for parity component forbidden
 - Comp IDs can be mirror specific
- Why foreign component support?
 - LU-10606 and LU-14319

liblustreapi parity API additions

- New pattern LLAPI_LAYOUT_PARITY
 - llapi_layout_pattern_set(layout, LLAPI_LAYOUT_PARITY)
 - llapi_layout_pattern_get(layout, pattern)
 - Only returns LLAPI_LAYOUT_PARITY
- New lcme_flags – LCME_FL_PARITY
 - llapi_layout_comp_flag_get(layout, flags)
 - Can't set LCME_FL_PARITY
- Parity setup
 - llapi_layout_parity_count_get(layout, data_count, parity_count, comp_ids, id_count)
 - llapi_layout_parity_count_set(layout, data_count, parity_count, comp_ids, id_count)
- llapi_layout_non_io_comp_add() *
 - Same as llapi_layout_add_first_comp()

Existing liblustreapi API impacted by parity

- Parity sanity setup checking and final setup
 - `llapi_layout_(v2)_sanity(layout, incomplete, flr. 'pool')`
 - This completes the parity component
 - No automatic updating of parity
- Adding layout with parity to an already existing layout
 - `llapi_layout_merge(dst_layout, src_layout)`
 - Imm sent to kernel is only based on `src_layout` (ost_id fid set) for comp add
 - For component addition and mirror setup
- Removing parity components
 - `llapi_layout_comp_del()` - any parity can be removed, data I/O only last component.
 - Need to use `llapi_layout_comp_iterate()` to delete all parity components
- `llapi_layout_file_comp_set()`
 - Don't allow changing `LCME_FL_PARITY` setting

Purposed user land tools interface

- `lfs setstripe -E 3M -S 1M -c 3 -l 1 -L ec:d3+p1 -E eof -S 1M -c 3 -l 3 ec:d3+p1`
 - Instead of `-E` we use `-l` since its comp IDs
 - Order shouldn't matter
- `lfs setstripe < yaml.conf`
 - Yes 1 to 1 parity to data component makes this painful. Sorry
- `lfs mirror resync`: update EC parity values.
 - Layouts can get stale
- Concerns for future
 - Doubling of layout entries might hit 64K xattr limit
 - Limit of 256 parity components per layout is really 64 if you use all mirrors. 1:1 mapping limits maximum data components to between 64 and 256 that is protected.

Internal kernel changes for parity work

- Challenges introduced by parity (LU-19298)
 - lod: ldo_comp_entries, lov: lsm_entries. Struct lov_comp_md_v1 used by both
 - For lod we modify both ldo_comp_entries and lov_comp_md_v1
 - ldo_comp_entries and lsm_entries are static arrays.
 - Adding or subtracting to static arrays is very limiting
 - Parity requires scans to figure out comp ↔ ec mappings
 - Replace static arrays with Xarray
 - We can have gaps.
 - Xarray can be multi-tier



Conclusion

- Erasure coding work is not dead. It is progressing and will be completed.
- Present this information to help foster more I/O library development
 - Allow development of new tools on top of the library
- Presenting this work in a bazaar fashion
 - Welcome feedback on the design from everyone

Acknowledgments

This work was performed under the auspices of the U.S. DOE by
Oak Ridge Leadership Computing Facility at ORNL under
contract DE-AC05-00OR22725.