Whamcloud

File Level Redundancy and Erasure Coding

Xu Zhenyu
bobijam@whamcloud.com

DDN® STORAGE

Agenda

- Current Lustre high available (HA) approach
- File Level HA for Lustre
- File Level Redundancy (FLR)
- Erasure Coding (EC)
- Further work

High Availability

Whamcloud

- No HA mechanism on Lustre level
- Leverage storage HA mechanism + third party software

File level HA for Lustre

Whamcloud

- FLR: File Level Redundancy
  - Using mirrors to replicate all file data on other storage devices
  - Active-active for read; active-passive for write
- EC: Erasure Coding
  - Using error correction code for recovery of subset of original data
  - K+M configuration

Composite File Layout

Whamcloud

- Simple File Layout
  - Stored in inode's EA (meta data on MDT)
  - File content's stripe information (stripe size, count, offset)
  - Logical Object Volume (LOV) to map logical file offset to offset in specific OST object
- Composite File Layout
  - Extent-based layout component covering range of simple file layout
  - Makes possible for complex file layout

File Level Redundancy (FLR)

Whamcloud

- FLR layout contains components covering [0, EOF) multiple times
- Each component groups covers [0, EOF) is called a mirror
- Redundant read
  - Data can be read from any available mirror
  - Client and layout policies to determine which mirror to read
- Write is partially redundant
  - Lustre needs to pick one available mirror to write
  - Data will be resynchronized among mirror asynchronously
  - Not necessary to resync entire mirror, FLR will synchronize modified components only

FLR write

Whamcloud

Layout flr status: Write Pending

FLR read

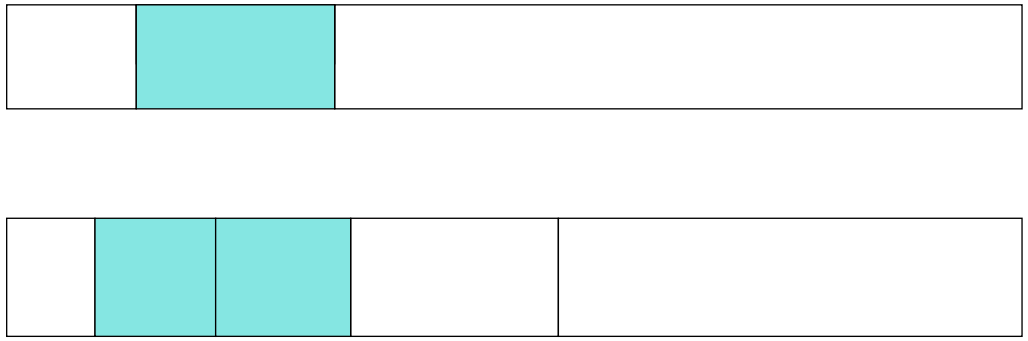
Whamcloud

Layout flr status: Write Pending

FLR resync

Whamcloud

Layout flr status: Read Only



Use cases for FLR delayed write

Whamcloud

- Tiered storage
  - Better HSM
    - For POSIX HSM targets, objects can stay in Lustre as secondary mirrors, which are only accessed when primary is unreachable
- Potential solution for burst buffer
  - HPC workload usually has I/O spikes. Burst buffer is managed as one of mirror; data will be migrated to secondary mirror later on
  - Burst buffer space can be reclaimed by deleting the mirror on the burst buffer

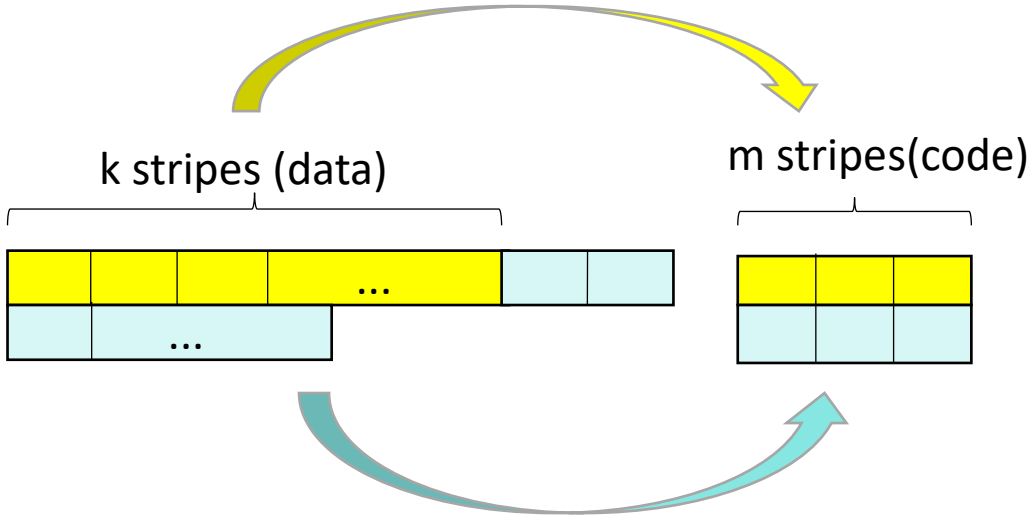
Erasure Coding (EC)

Whamcloud

- EC layout contains parity code components as well as data components
- A recovery group consists of K data stripes + M parity code stripes
- Delayed parity code updating for write
  - Parity will be calculated asynchronously
- Read with unavailable OSTs
  - Can recover data with up to M OSTs are unavailable

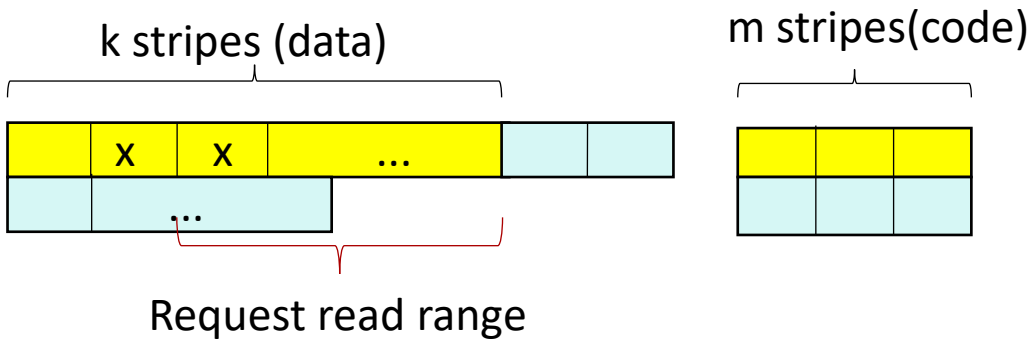
Data and parity mapping

- Data component and corresponding parity component covers the same file extent range
- User can set parity configuration, i.e. <k, m> tuple



EC Read

- Encounters failed OST, check whether parity code component stale
- If parity code component uptodate, read relevant available data stripes and parity code stripes
- Calculate missing stripes and fill the corresponding data page cache
- Could possibly expand the read range to fulfill the recovery



Further work

Whamcloud

- Finish EC phase I work
- Data mover for FLR
  - Resynchronizing replicated files after writing
  - Data movers are dedicated clients

Whamcloud

Thank You!

DDN® STORAGE