Whamcloud

Lustre Reflink

Li Xi

Sept 2019

DDN® STORAGE

What is reflink?

Whamcloud

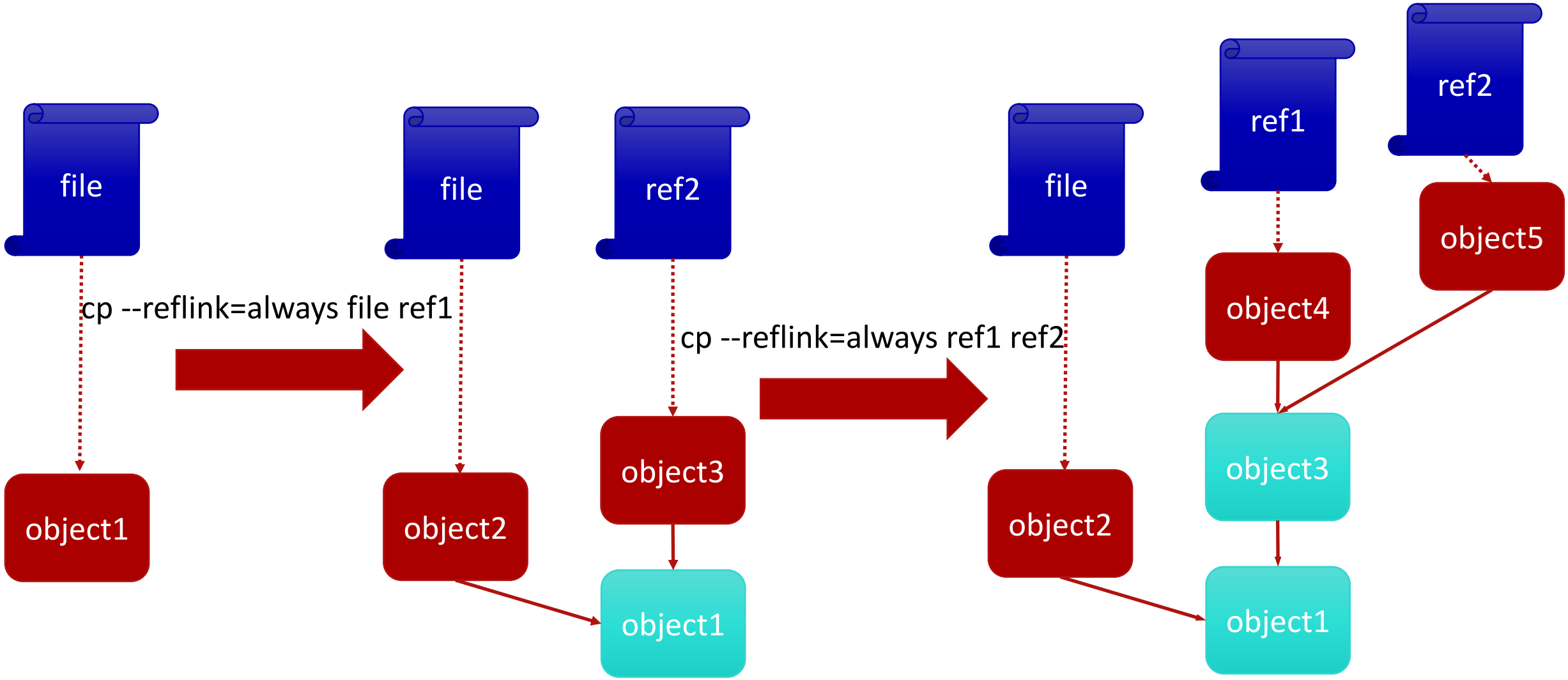
- Supported by
  - XFS
  - Btrfs
  - Not on Ext4/ZFS
- copy-on-write
  - quick copy
  - shared data blocks to save space
- Interface
  - cp --reflink=always
  - ioctl(FICLONERANGE)
  - ioctl(FICLONE)
- High level design
  - http://wiki.lustre.org/Lreflink_High_Level_Design

General Ideas for Lustre Reflink

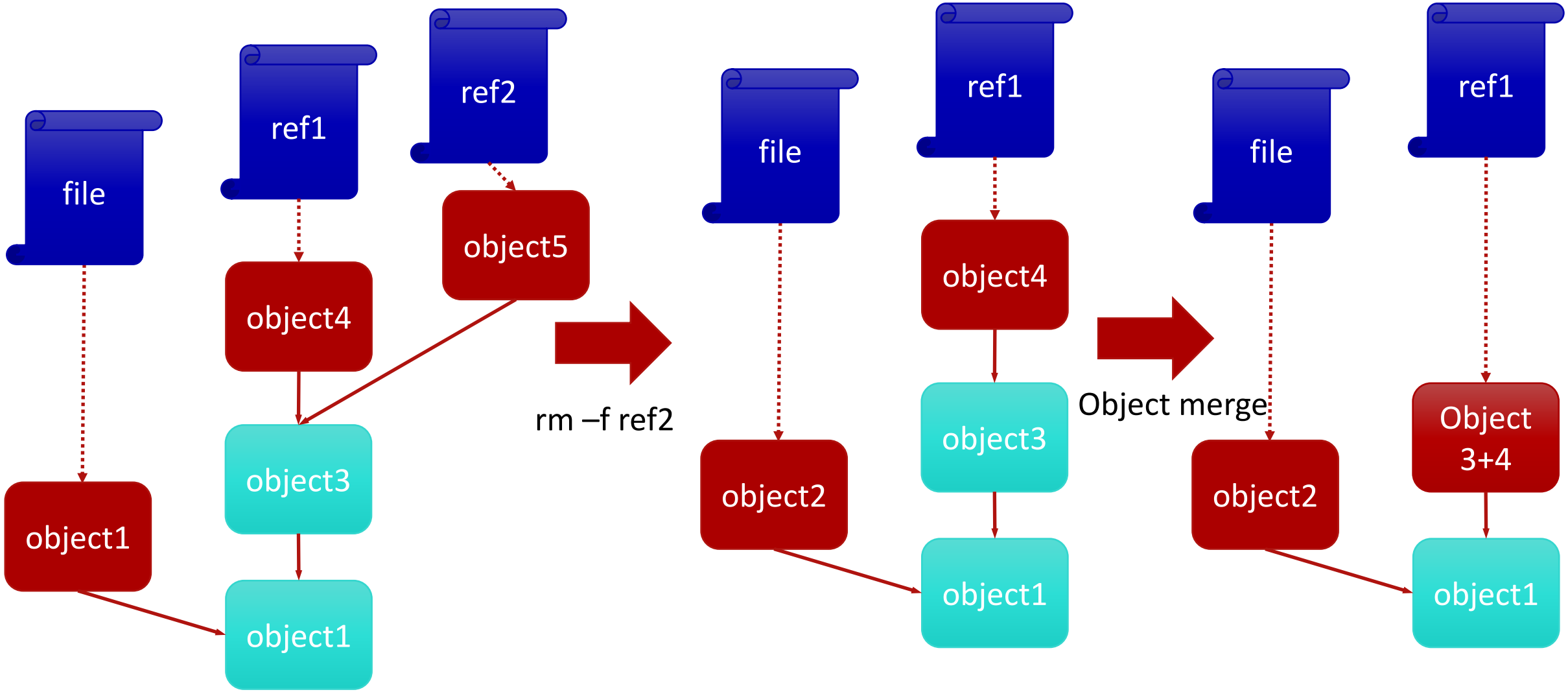
Whamcloud

- OSD level implementation
  - Not on Ext4/ZFS level
  - OSD object will point to its parent object for original version of the data
- Lreflink doesn't clone data across OSTs
  - Data copy will only limited to OST internally
  - Lookup of parent object is quicker
- Binary tree of reflinks
  - The reflink objects will form a binary tree
- Bitmaps of Lreflink
  - Lreflink uses bitmap to indicate whether the data locates on this object
- Xattr for bitmap
  - Bitmap will be saved as EA of object

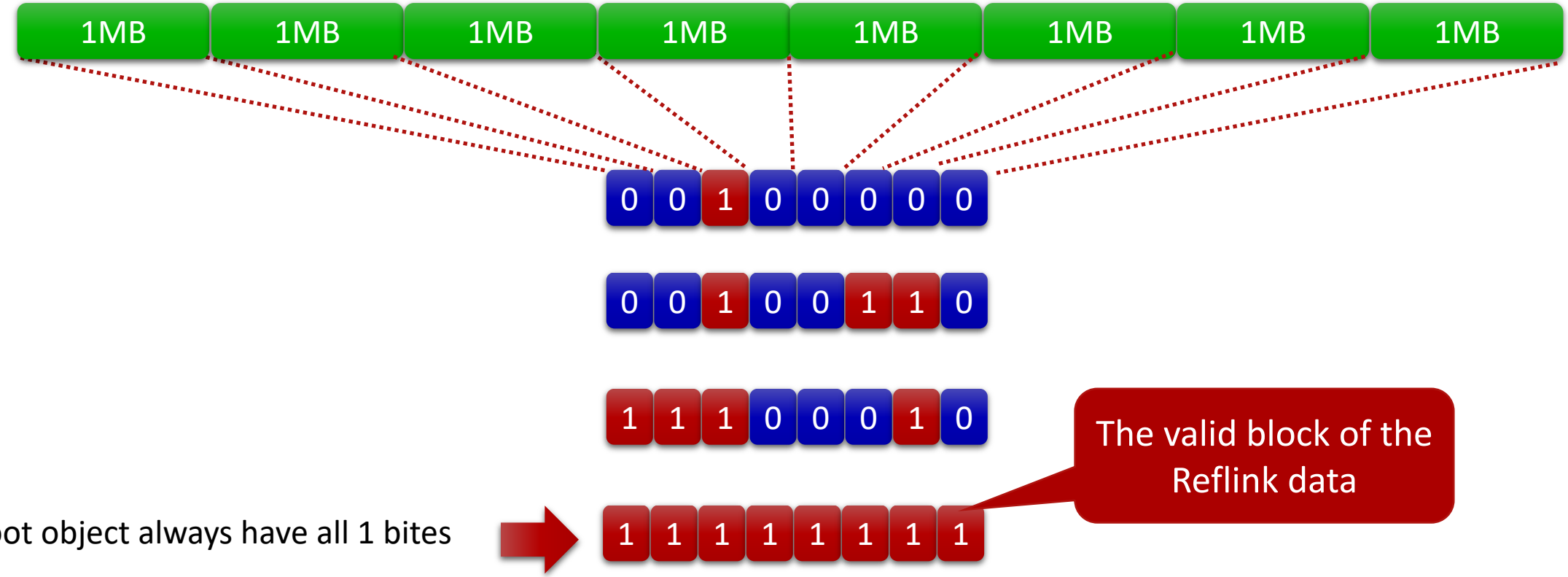
Creating Lreflinks of Lustre Files



Removal of reflink file



Bitmaps of Lreflink



The root object always have all 1 bites

Interfaces to Read/Write EA Efficiently

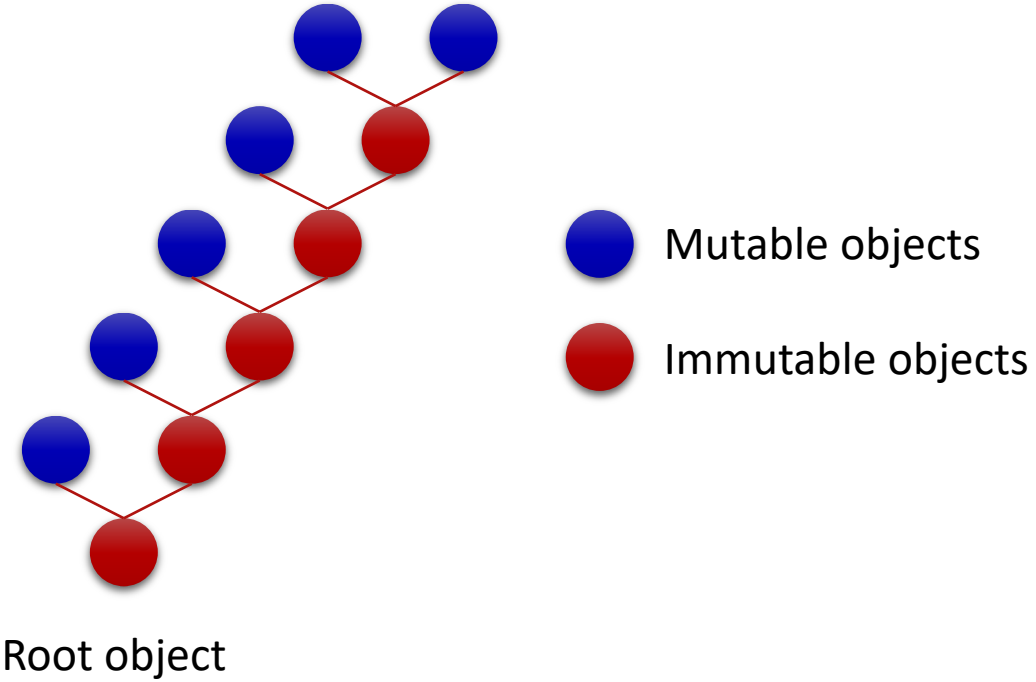
Whamcloud

- Ext4/ZFS need to provide interfaces to read/write EA quickly
- EA size could be huge
  - For 1PB file, EA size for reflink is 128MB
  - For 1TB file, EA size is 128 KB
- [g|s]etxattr_range(struct inode *, void *buf, size_t count, off_t offset)

Time Efficiency

Whamcloud

- Zero overhead for non-reflink data
- The bitmap reading/writing through EA should be as efficient as file reading/writing
- M level relink tree needs to read bitmaps for M - 1 times in maximum
- Big aligned write should have similar performance with non-reflink file
- Small write might trigger COW in worst case
  - M – 1 read of bitmaps
  - 1MB read from root object
  - 1MB write to leaf object
  - Then write a small range



Space Efficiency

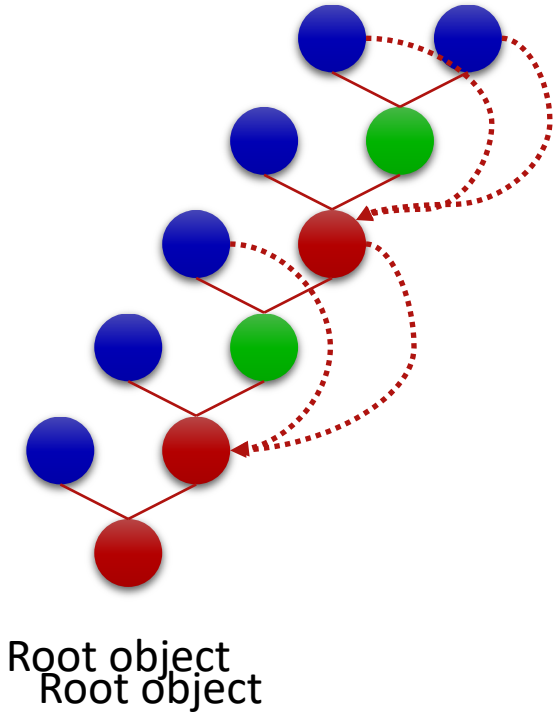
Whamcloud

- Reflink tree with L leaves have 2 * L - 1 objects
- If all objects are modified completely, the space is wasted in the worst way
  - Comparing to non-reflink, space usage is doubled
- If the old file is kept untouched, only the new reflink is completely modified
  - The space usage is the same with non-reflink copies
- Bitmap costs file size / 8M space
- 1MB space will be occupied even only a small range is modified

Optimizations – Skip Unchanged Objects

Whamcloud

- Maintain a point to the first ancestor that has real data



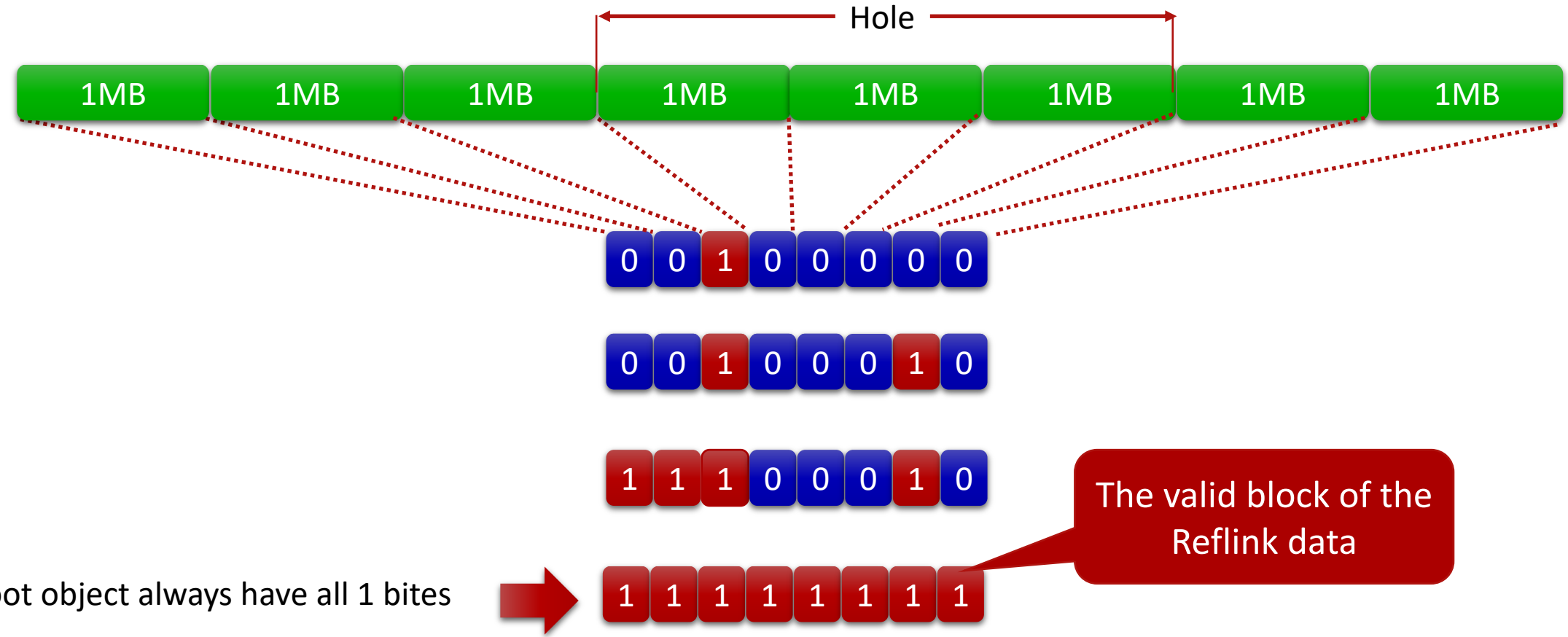
Mutable objects

Immutable objects

Immutable objects that has never been modified

Optimizations - Files with hole

- How to avoid unnecessary reading of bitmaps?



The root object always have all 1 bites

New ideads

- Use fiemap rather than bitmap
- Hole is still a problem even using fiemap

Whamcloud

DDN®
STORAGE