NetApp™

Go further, faster™

Metadata Striping in pNFS
IETF-73
2008-11-21

Mike Eisler
Senior Technical Director

What I am asking for

- Primary request
  - Add metadata striping to the NFSv4 charter
- Secondary request
  - Start with draft-eisler-nfsv4-pnfs-metastripe-01.txt
    - Is based on metadata striping work at NetApp
    - Attempts to generalize to work other metadata striping schemes

Why?

- Metadata matters
  - Benchmarks that people care about are mostly metadata
    - e.g. SPEC SFS 2008
    - e.g. IOZone is adding metadata
- E.g. applications
  - software development (build, revision control)
    - incremental builds are mostly metadata accesses
  - image stores
    - consumer photos
    - social networking
- Yes, file system implementations can stripe metadata without adding metadata striping to pNFS
  - Just like file system implementations can and do stripe data without pNFS
    - Then why are they all supporting pNFS?

The proposal at a glance

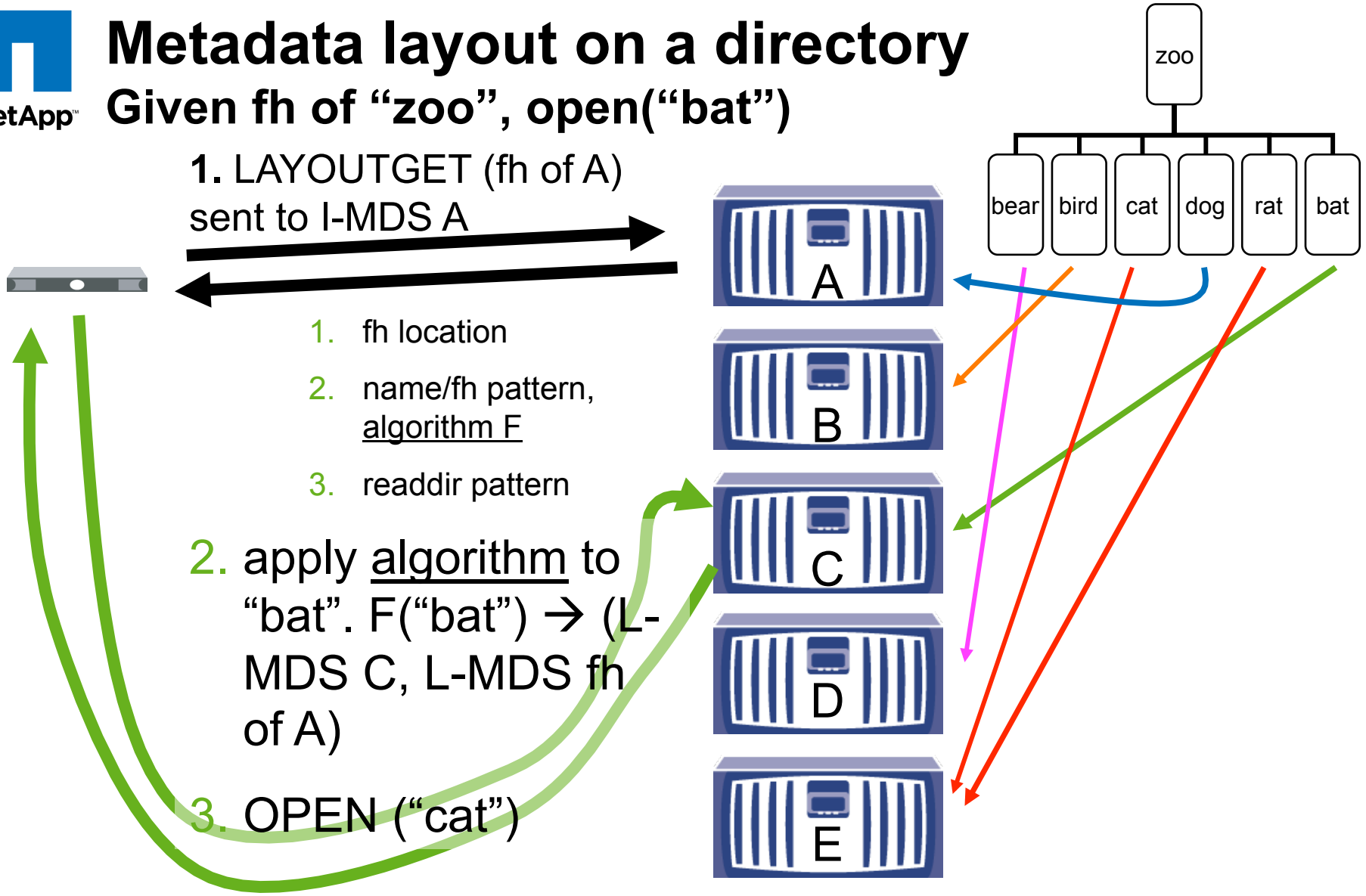
- Does not require a new minor version of NFSv4
- Requires a new layout type
- Provides three types of layouts
  - all three are returned in the same LAYOUTGET
  1. file object location
     - fh-only operations get sent to the object location
       - where attributes live
  2. file name location – directory only
     - fh/name operations get sent to the name location
       - ideally the same place where attributes live
         - links, renames can frustrate this over time
  3. directory contents location – directory only
     - Directories are sort of like regular files when you read them
  - expectation is that most LAYOUTGETs will be for directories
- Borrows from files-based layout NFSv4.1
  - indices array
  - file handles array
  - device addresses

Concepts

- Metadata server: MDS
  - as defined in pNFS specification
- I-MDS – the initial MDS
  - LAYOUTGETs for metadata layouts are sent to the I-MDS
- L-MDS – the layout MDS
  - The client is directed to an L-MDS by a metadata layout

Metadata layout on a directory

Given fh of “zoo”, open(“bat”)



READDIR

- Essentially treat cookies as offsets
- Layout returns a list segments
  - embedded in the metadata layout, not as elements of the logr_layout array
  - each segment has a starting cookie
    - first segment has a starting cookie of zero
- Each segment can have a different striping pattern
  - Each pattern extends up to, but not including the starting cookie of the next segment
    - Last segment extends to the maximum cookie value
- The cookies used with an L-MDS do not have to work on an I-MDS
  - useful if the server's file system directory format is incompatible with striping
    - e.g. cookies might not be returned in ascending order (or any order for that matter)

Layout recall: all or nothing

- Keeps it simple
- Directories are usually read from start to EOD

NetApp™

Go further, faster™

De-Duplication Awareness in pNFS IETF-73 2008-11-21

Mike Eisler
Senior Technical Director

What I am asking for

- Primary request
  - Add de-duplication awareness striping to the NFSv4 charter
    - virtualization is the justification
- Secondary request
  - Start with draft-eisler-nfsv4-pnfs-dedupe-00.txt
    - Seems to fit with known de-duplication schemes
    - Has been normalized to work other metadata striping schemes

Why?

- Magnetic disk is cheap
- And yet customers are driving storage vendors toward eliminating redundancy
  - first it was whole files
  - now it is blocks within files
- NFS clients caches data from storage arrays in DRAM and flash
  - DRAM and flash are expensive
- Ergo, de-duplication in NFS clients matters
- The hypervisors are doing it already
  - So storage arrays should give hypervisors the de-duplication maps

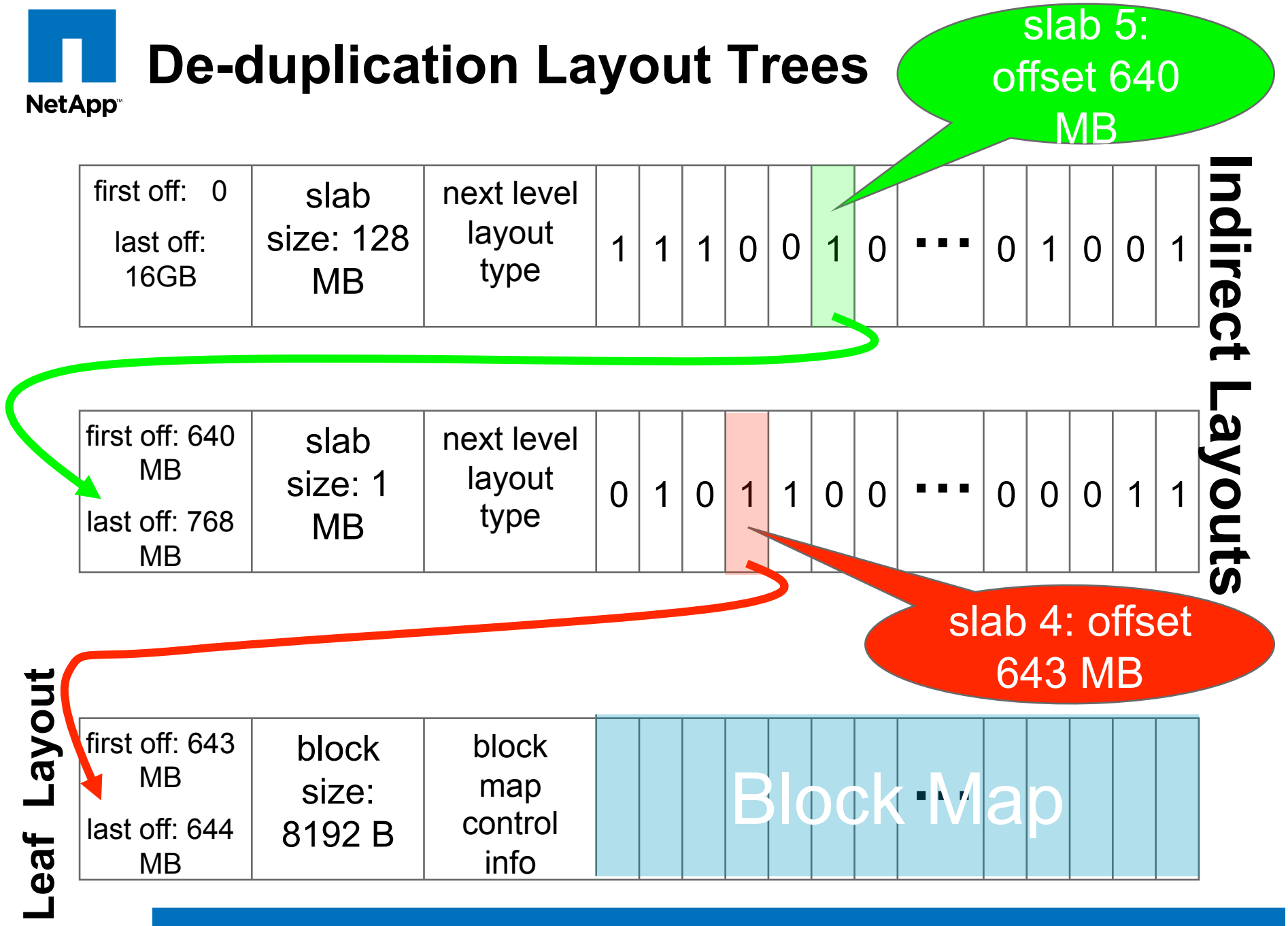
The proposal at a glance

- Does not require a new minor version of NFSv4
- Requires new layout types
- Use bit maps to indicate if a range of data in a file is a duplicate from another file
- Supports hierarchical (e.g., clones, snapshots), in-line, and background de-duplication
- Supports cross-storage-node de-duplication
  - Can integrate with existing files, objects, and blocks layouts
- Limited to regular files
  - De-duplication awareness of directories is reasonable,
    - but perhaps best captured in a separate document

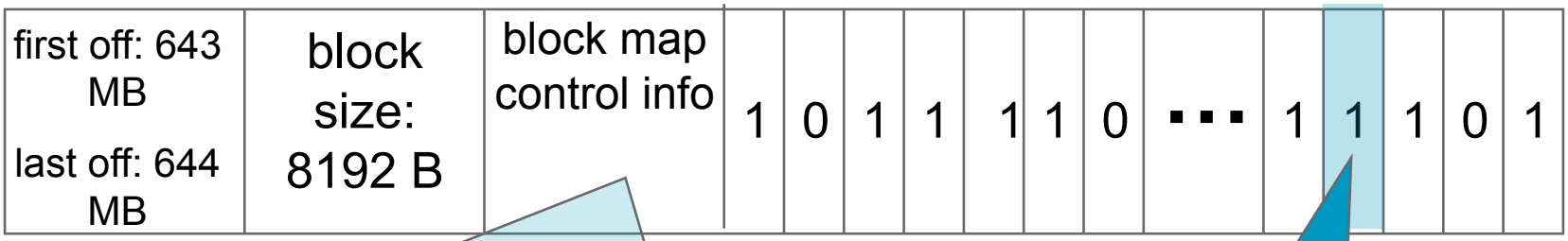
Concepts

- Source file:
  - the file that contains the de-duplicated data.
- Target file:
  - the file the client has opened.
- Block:
  - the smallest unit of de-duplication that the server is willing to support.
- Slab:
  - a byte range that refers to lists of smaller slabs or blocks
- Regular file:
  - An object of file type NF4REG or NF4NAMEDATTR
- Indirect layouts contain slabs
  - Refer to indirect layouts or leaf layouts
- Leaf layouts contain blocks
  - Leaf layouts indicate the source files

De-duplication Layout Trees



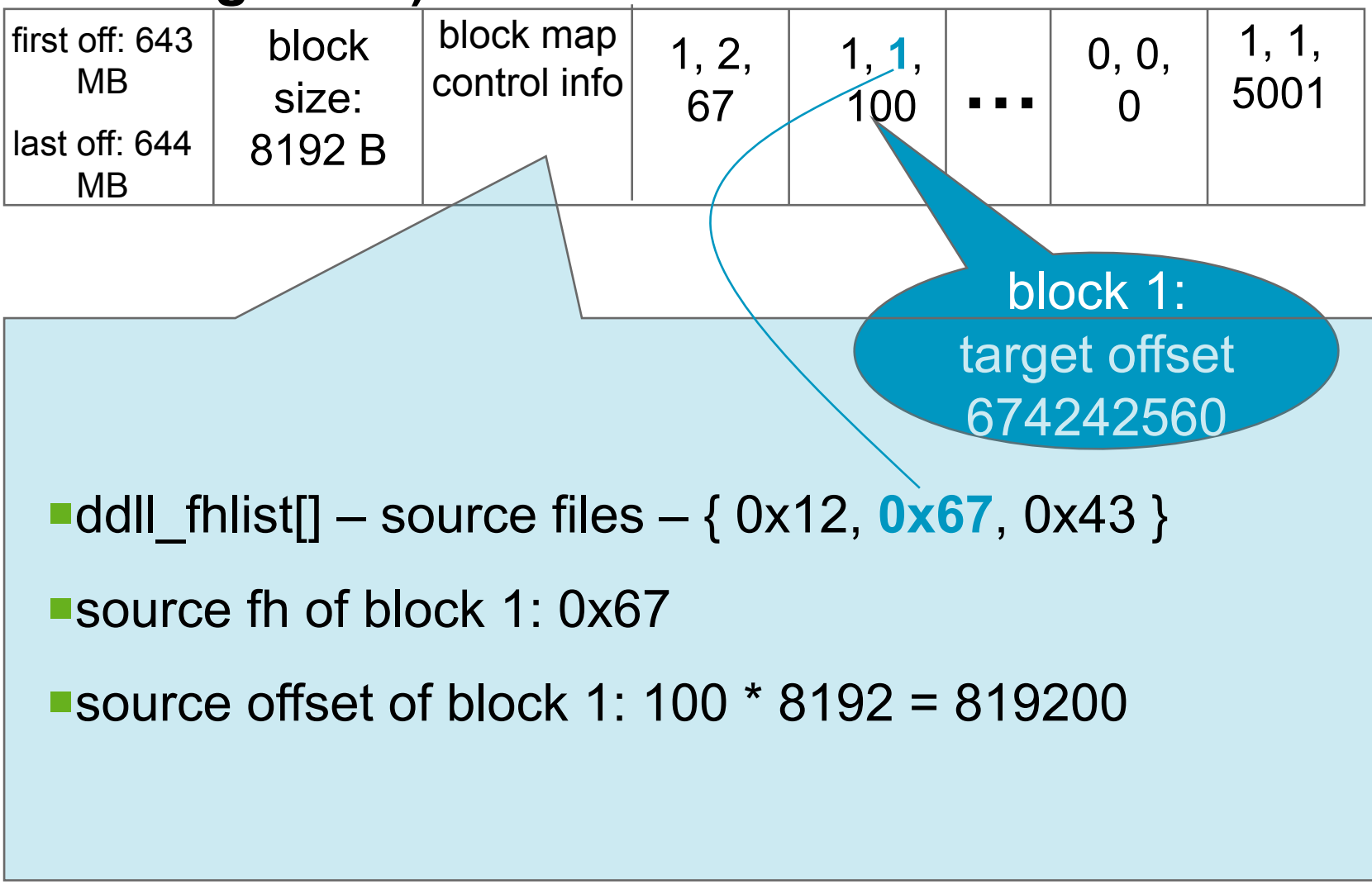
Leaf Layout
Hierarchical De-duplication (snapshot, clone)



block 124: target offset 675250176

- ddll_fhlist[0] – source file
- ddl_change_attr[0]
  - If absent: server will recall leaf layout before changing active blocks of source file.
  - If present: client must compare ddl_change_attr[0] with change attribute of source file before using block from source.
- source offset: also 675250176

Leaf Layout
Non-Hierarchical De-duplication (inline, background)



Leaf Layout
Cross-Node De-duplication

first off: 643 MB last off: 644 MB	block size: 8192 B	block map control info	1, 2, 2, 67	1, **1**, **2**, 100	▪▪▪	0, 0, 0	1, 1, 0, 5001

block 1: target offset 674242560

- ddll_devlist[] – device IDs – { 0x333, **0x111**, 0x222 }
- ddll_fhlist[] – source files – { 0x12, 0x67, **0x43** }
- source file's device: ID 0x111
  - can map to network address of another MDS
  - can map to a non-de-dupe layout type
- source fh of block 1: 0x43
- source offset of block 1: 100 * 8192 = 819200