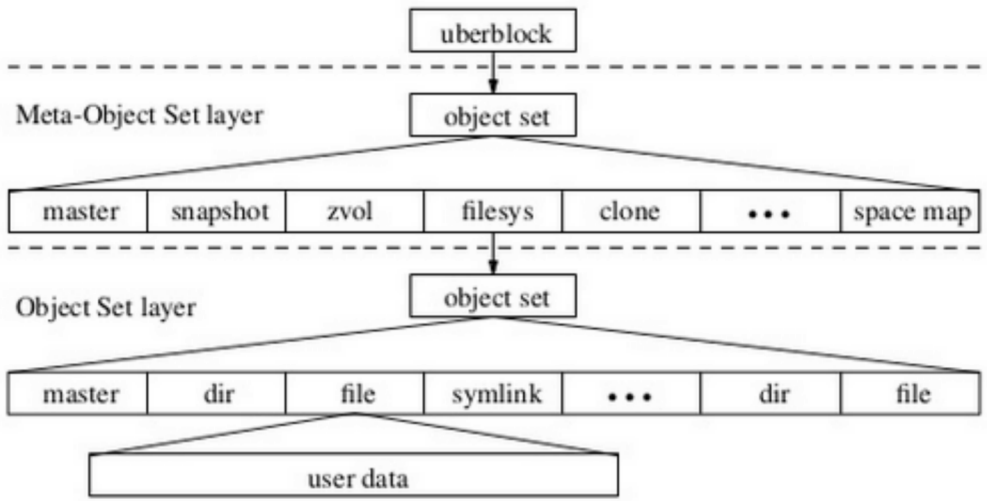
ZFS & TRIM

Agenda

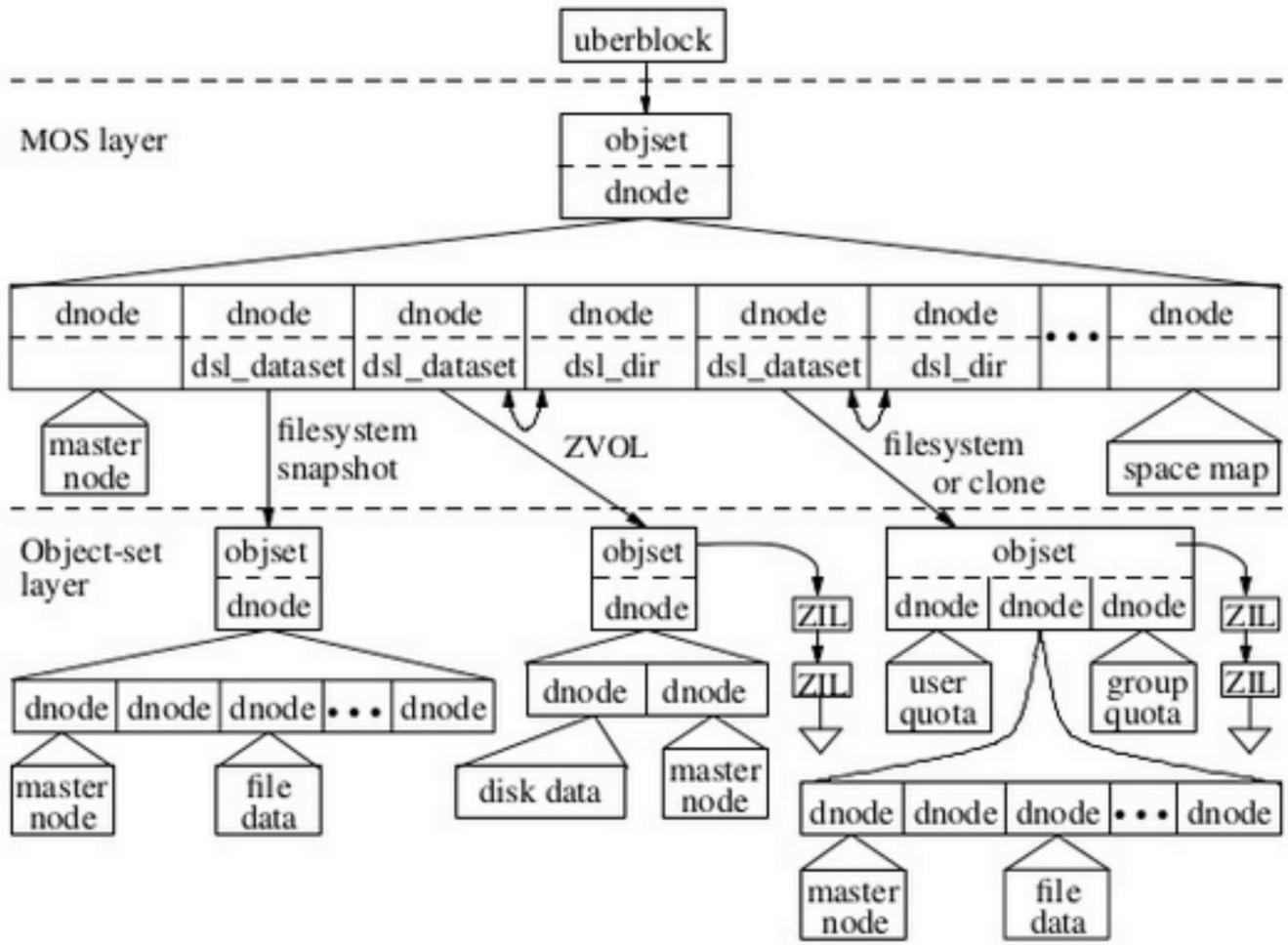
1. ZFS Structure and Organisation
    1. Overview
    2. MOS Layer
    3. Object-Set Layer
    4. Dnode
    5. Block Pointer
2. ZFS Operations
    1. Writing new data to disk
    2. **Freeing blocks**
3. TRIM

ZFS Structural Overview



- uberblock points to a data structure that describes an array of meta-objects
- meta-objects include filesystems, snapsots, clones, ZVOLs and the space map of free/allocated blocks in the pool
- MOS object references an object-set that describes its array of objects
- the objects include things like directories, files, symbolic links, etc
- Finally, these objects reference an array of blocks that contain the objects' data

ZFS Structure



Meta-Object Set(MOS) Layer

- Dataset and Snapshot Layer(DSL) and Storage Pool Allocator(SPA) modules implement the MOS layer
- It manages the pool of space and makes it available to filesystem modules of object-set layer
- DSL tracks datasets, which includes snapshots, clones, active filesystems, and ZFS Volumes(ZVOLs), and deadlists
- SPA tracks allocated vs free blocks in the current pool and is also responsible for handling compression and deduplication

Object-Set Layer

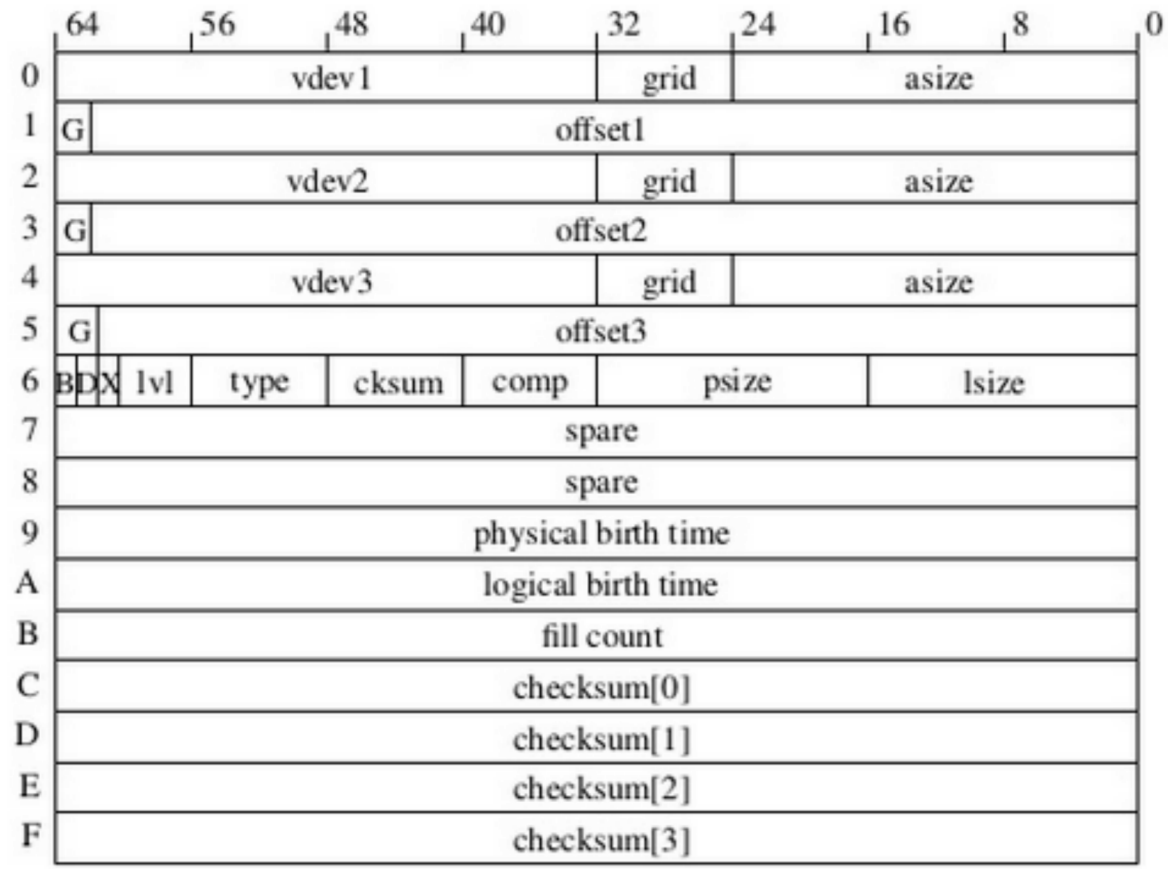
- ZVOLs - single dnode which references two dnodes
  - disk data - dnode references an array of block pointers
  - master node - records ZVOl-specific information
- Filesystems - three dnodes
  - 2 dnodes record user and group space usage for a filesystem
  - 3rd dnode references an array of files and directories
- Clones of filesystem/ZVOL have same organization as the filesystem/ZVOL

DNODE

- Analogous to INODE but also describes objects in MOS layer.
- Managed by DMU.
- Describe files, directories, filesystems, snapshots, clones, space maps etc.
- Size < 128 Kb -> Direct pointer to appropriate size block
- else -> 1 level of indirection: points to 16Kb block -> each entry points to 128 Kb blocks.
- Can increase level of indirection if required.
- Reference ZAP objects

Block Pointer

- Checksum for every block (up to 3 copies of data).
- All meta-data blocks have double redundancy by default.
- Birth time - counted in terms of number of checkpoints since the ZFS pool was created.
- Dedup flag - quick shortcut



Freeing blocks

- **ds_deadlist_obj** in **dsl_dataset_phys_t**
- **Deadlist ->** I don't want this block, but a previous snapshot might.
- Only free a block if:
  - No references to this block
  - birth_time of the block is more than the birth_time of the latest snapshot
- While deleting snapshot, free those blocks that
  - Are in the next snapshot's deadlist **AND**
  - have birth_time greater than previous snapshot.

TRIM - Motivation

- Attempt to make sure certain writes do not take very long (as compared to other writes).
- SSD Blocks only have a limited number of erases.
- Using TRIM, a FS can tell the underlying SSD that certain blocks are no longer relevant.
- TRIM reduces, on average, garbage collection cost and also increases the lifetime of SSDs.
- TRIM does have overhead - so use judiciously !!