The HAMMER Filesystem

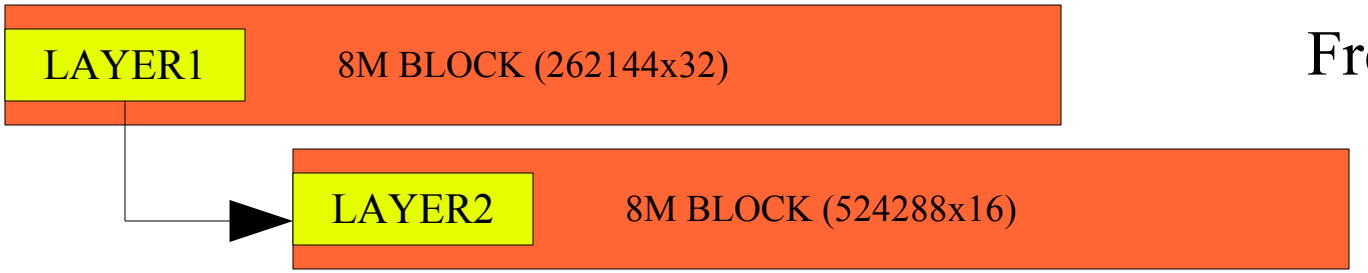
DragonFlyBSD Project
Matthew Dillon
11 October 2008

HAMMER Quick Feature List

- 1 Exabyte capacity (2^60 = 1 million terrabytes).
- Fine-grained, live-view history retention for snapshots and undo.
- Fast crash recovery on mount via UNDO FIFO.
- Non-queued mirroring, master-to-many-slaves, slave-to-many-slaves.
- 64 bit inodes, 2^63 byte file size limit. Stable inode numbers (never reused).
- Data and meta-data CRCs, non-recursive.
- Pseudo-filesytems with independent inode numbering spaces.
  (For export, backup, mirroring, sub-mounts, and snapshot management).

HAMMER Media Layout – Zone & Freemap

- Universal Zone offset, 64 bit, byte granular.
- Used everywhere in HAMMER.
- 16 zones, Zone 0 is reserved.
- Direct mapped, no block number translation, but zone can be validated.

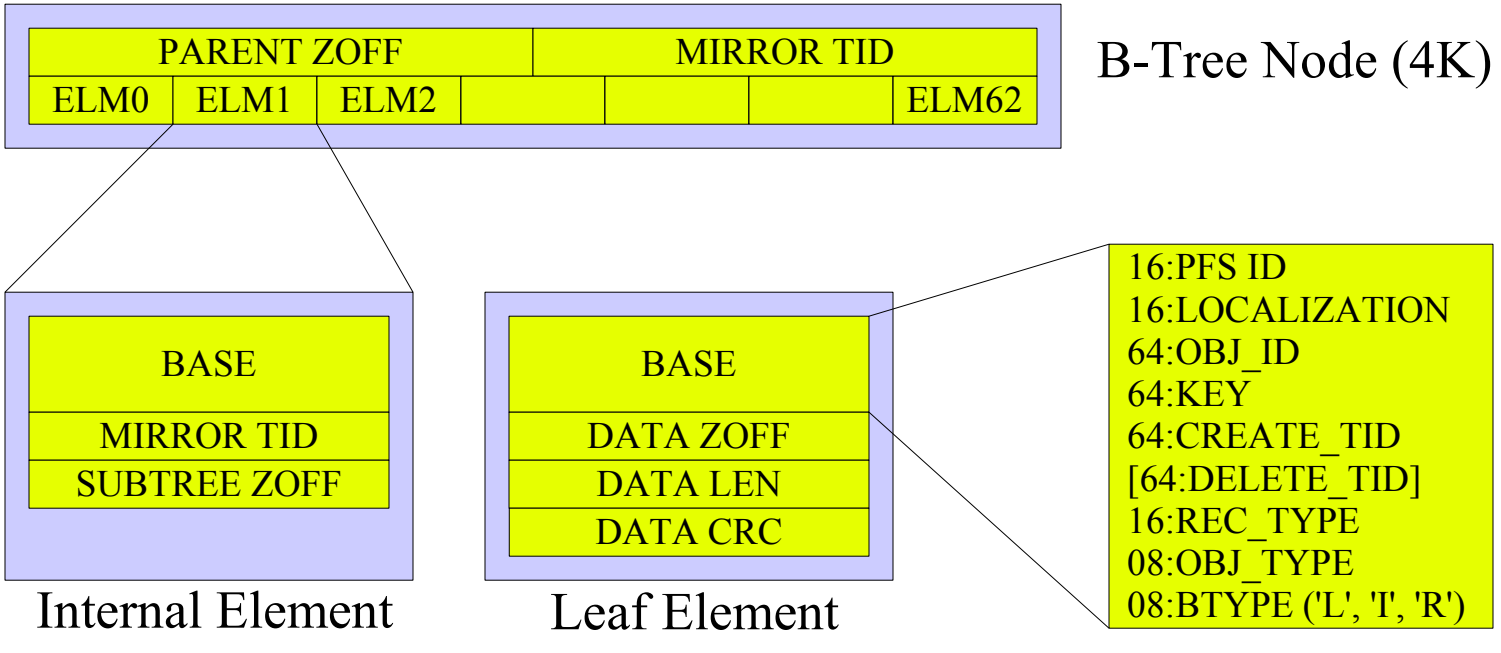


Freemap

- Layer1 – 4TB/entry (1 Exabyte represented in one 8M block).
- Layer2 – 8MB/entry (4 Gigabytes represented in one 8M block).
- Layer1 blockmaps Layer2. Layer2 direct-maps a 64 bit zone offset.
- The freemap can validate but not translate zone offsets.
- Layer2 – Record zone assignment, append point, bytes free in block
- Recent allocation offsets stored in volume header

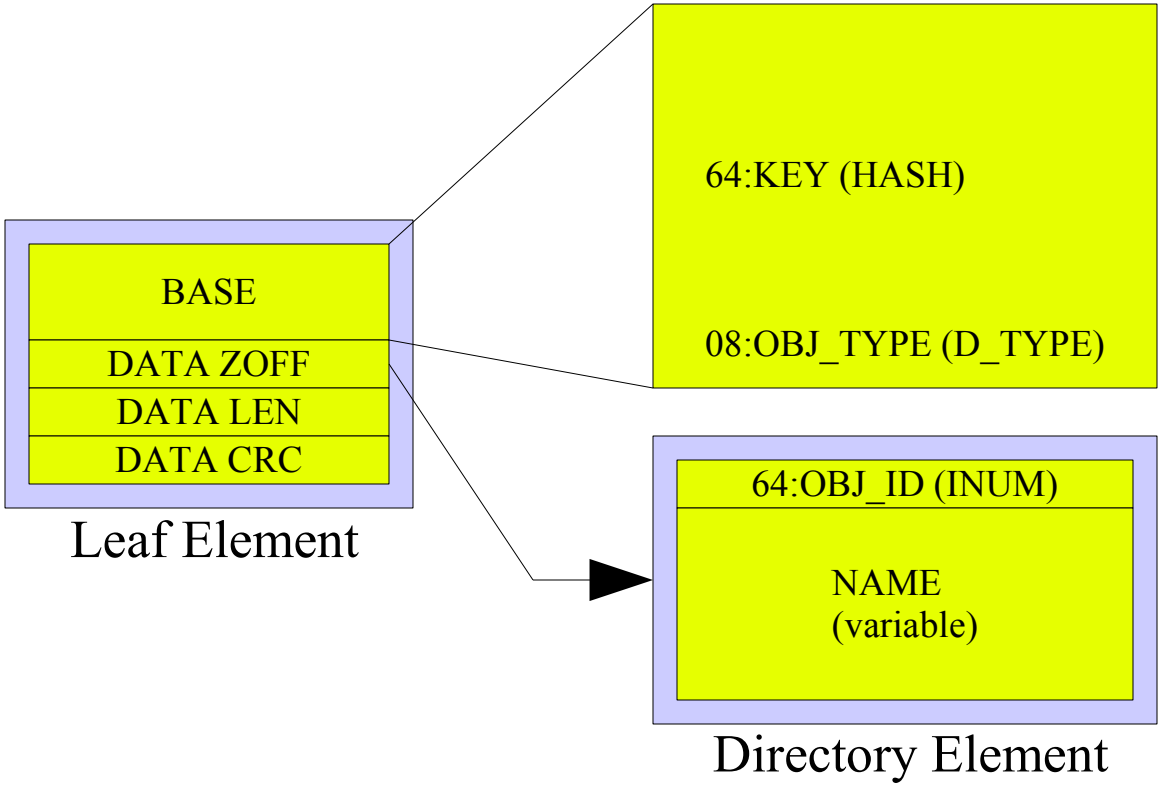
HAMMER Media Layout - B-Tree

- 63-Way B-Tree, fat 64-byte B-Tree elements for now, with large keys.
- Used to index EVERYTHING. Inodes, directory entries, data blocks, etc.
- MIRROR_TID used solely to support incremental mirroring streams.
- Localization groups B-Tree elements and related data, e.g. inodes vs file data.
- Both Left and Right-hand bounds for internal nodes (ELM63 does not recurse)
- Searches complicated by CREATE_TID.
- Insertion, Deletion, Update cases – DELETE_TID and snapshot access.

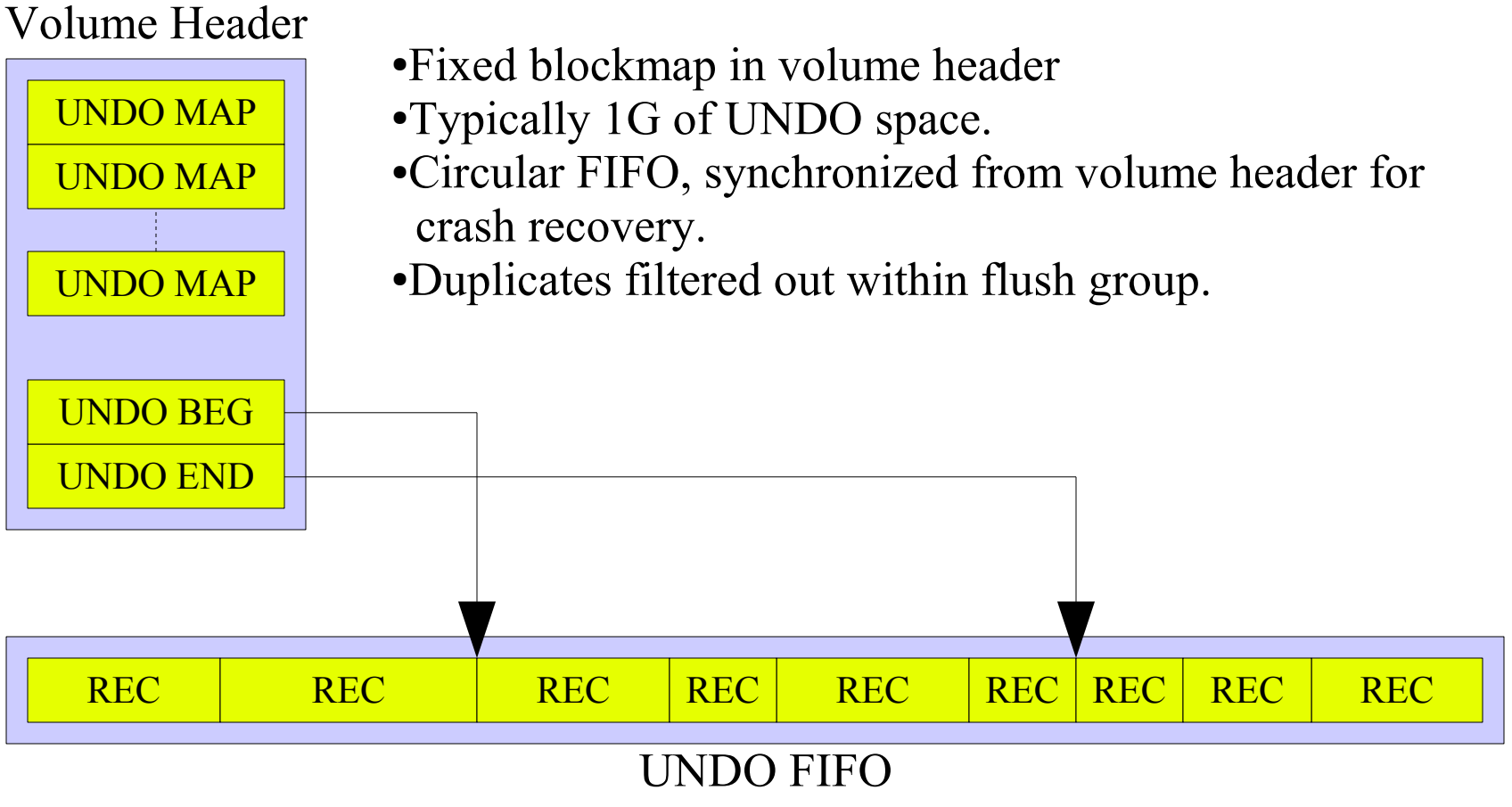


HAMMER Media Layout - Directory

- Directories are directly indexed by the B-Tree
- Directory entries not currently embedded in the B-Tree element
- Directory entries are well packed and localized, however.
- B-Tree uses a name hash for the key, but mistakes were made.



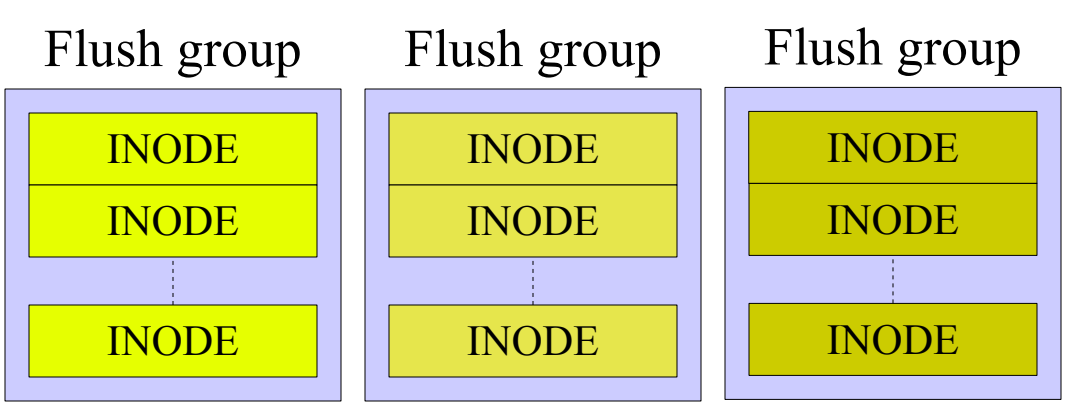
HAMMER Media Layout - UNDO



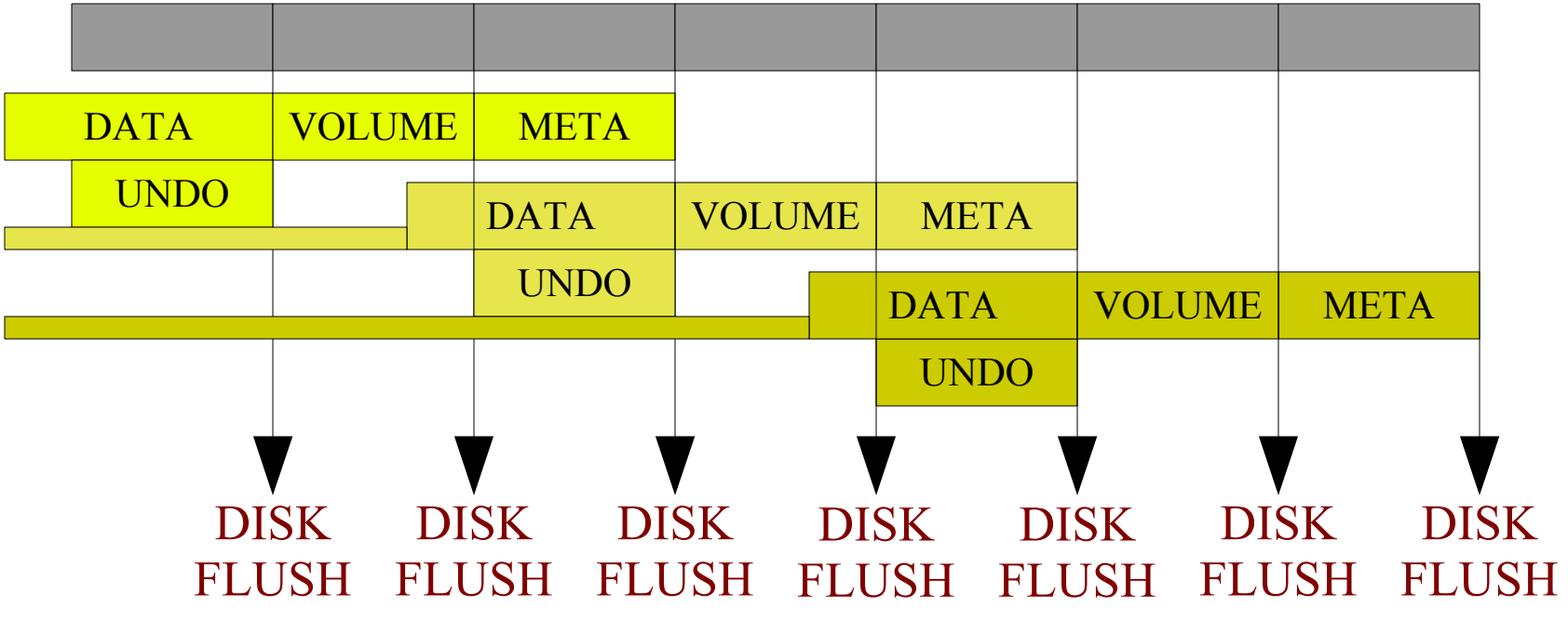
HAMMER Low Level Features

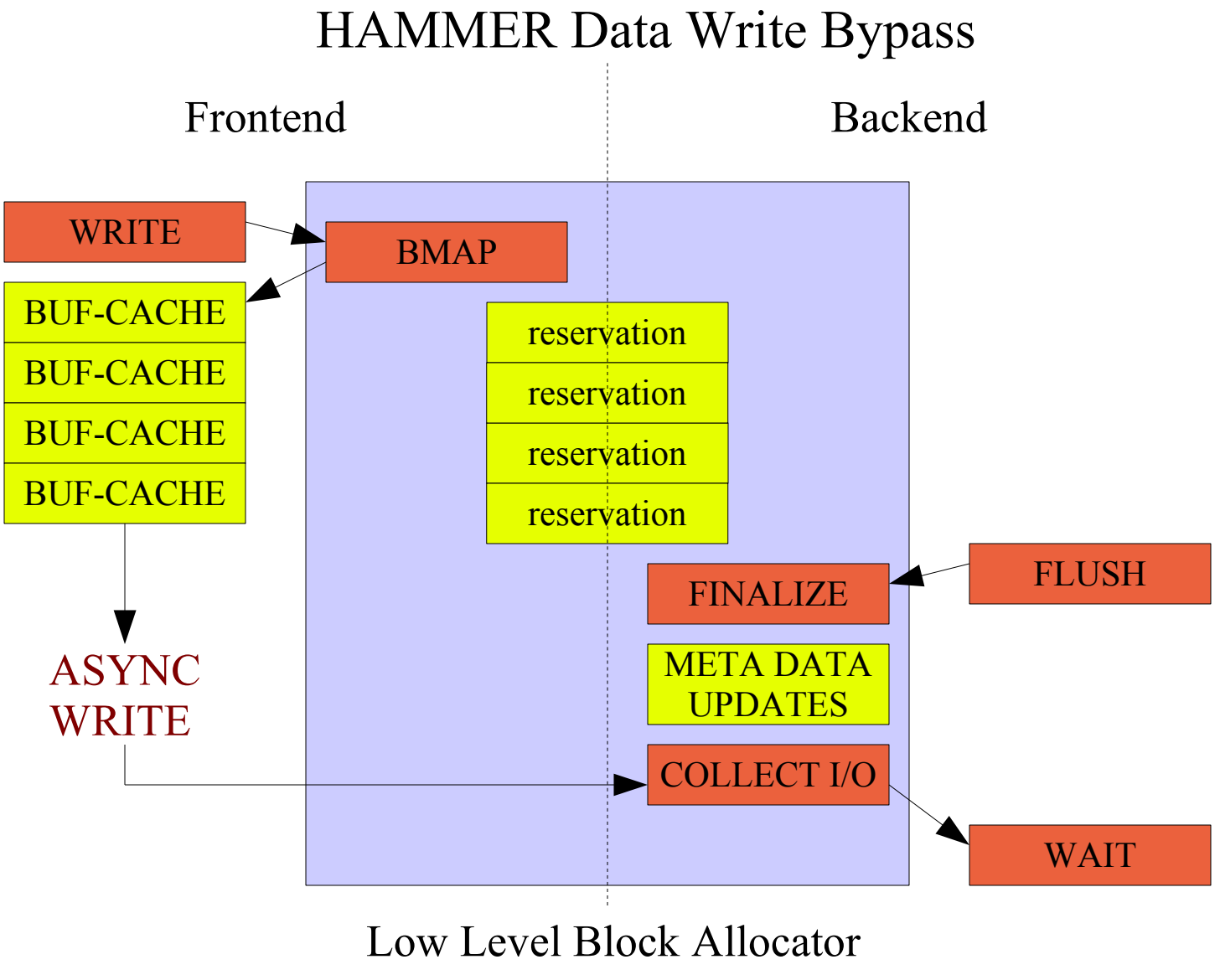
- Frontend VOP operations disconnected from backend Flush.
- Frontend modifying VOP operations 100% logically cached.
- Direct-data bypass for reading and writing – VOP_BMAP works.
- Extent-based data records, but currently limited to 16K/64K.
- Flexible pruning based on transaction id ranges, operates on live filesystem.
- Reblocking of data and meta-data operates on live filesystem.
- Mirroring streams on a per-PFS basis, batch or near real time.

HAMMER Media Flush

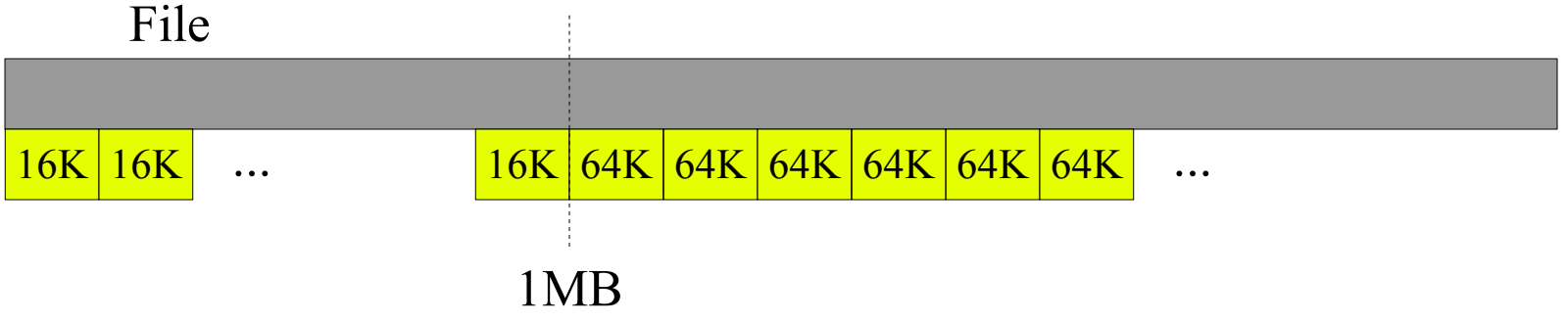


- All operations cached in memory.
- Multiple inodes per flush group.
- Data writes are asynchronous.
- Undo records generated by flush.
- Fsync() is expensive (4 flushes).
- High level of I/O parallelism.
- (See FUTUREs slide)




HAMMER Data Write Bypass
Frontend
Backend
WRITE
BMAP
BUF-CACHE
BUF-CACHE
BUF-CACHE
BUF-CACHE
reservation
reservation
reservation
reservation
FINALIZE
FLUSH
ASYNC
WRITE
META DATA
UPDATES
COLLECT I/O
WAIT
Low Level Block Allocator

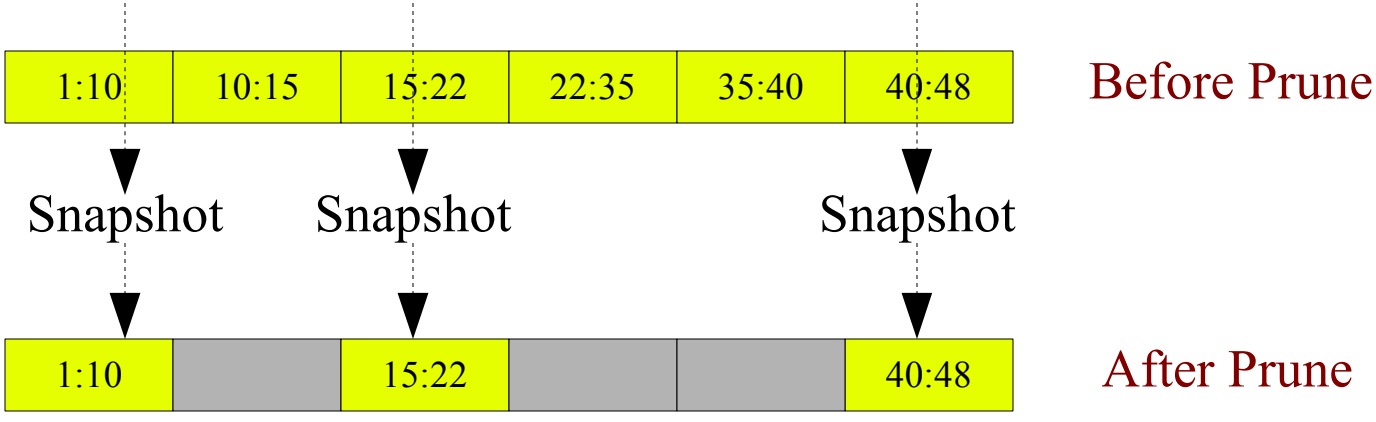
HAMMER Data Extents



- Each Data record is a B-Tree element.
- Switch to 64K records past 1MB.
- Read code is able to stitch extents together.
- Write code cannot yet handle dynamic extents:
  - (1) Breaking up large extents not trivial.
  - (2) Historical access & organization not trivial.
- BSD Clustering supported, but it wasn't fun.
- Serious issues with mixed buffer cache operations.

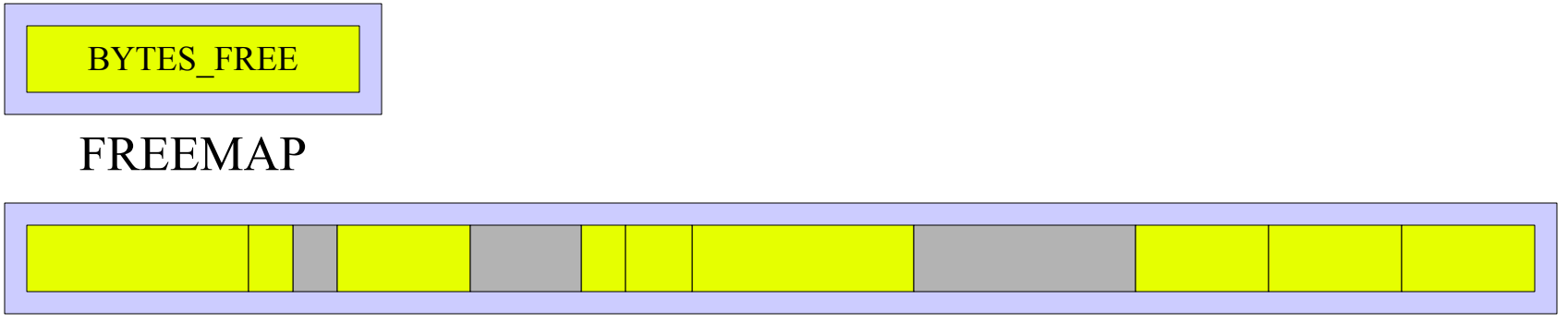
HAMMER Pruning

- Every B-Tree record has a CREATE_TID and a DELETE_TID.
- CREATE_TID is part of the sorting key, DELETE_TID is not.
- Deletions simply set the DELETE_TID in the existing record.
- Updates set DELETE_TID and insert a new record with new CREATE_TID.
- Transaction ids increase monotonically but do not reflect real time.



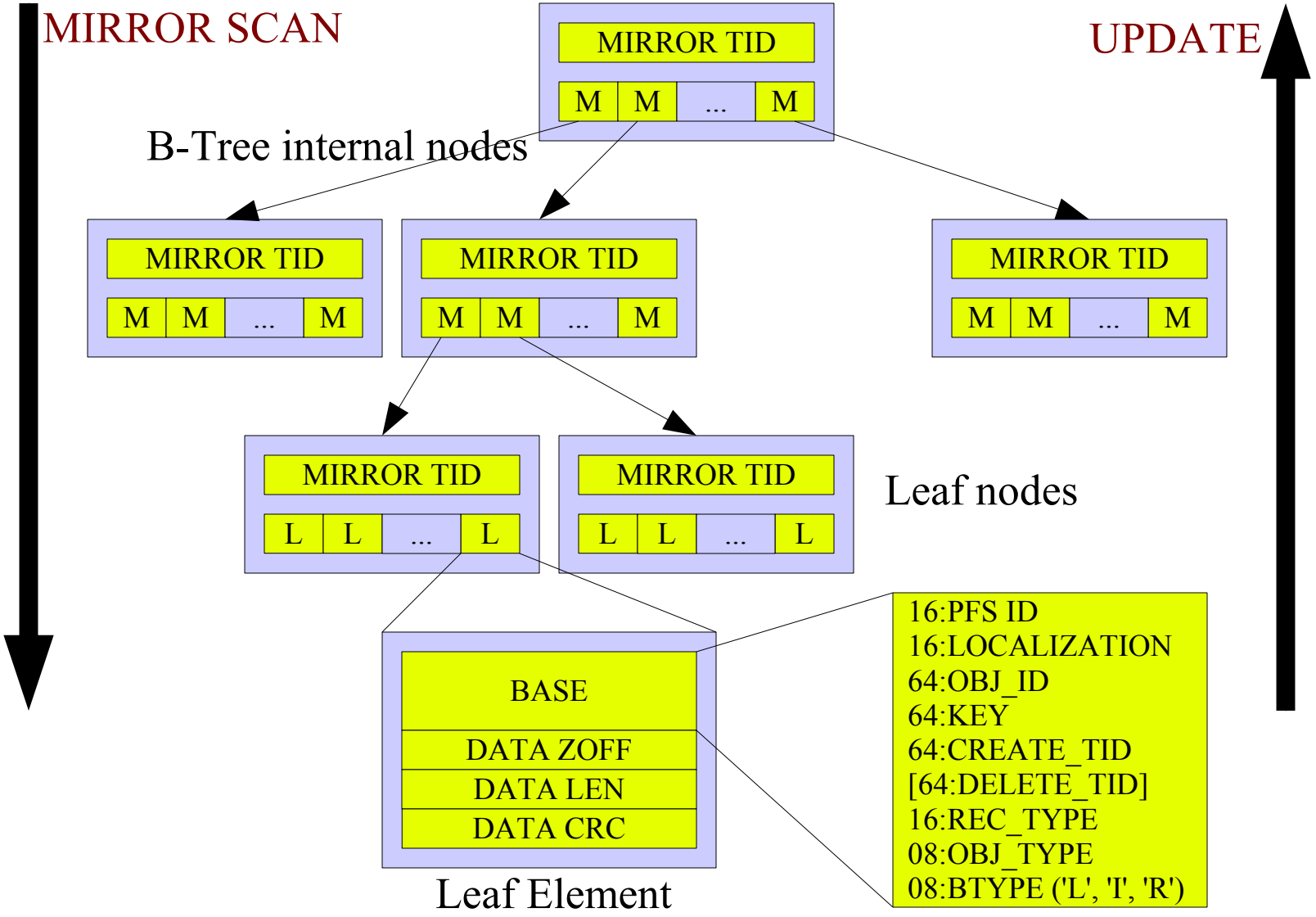
- Remaining elements not expanded to fill voids:
  - Modifying CREATE_TID interferes with mirroring.
  - Modifying CREATE_TID considered too dangerous.
  - Once pruned, only snapshot TIDs prior to most recent snapshot are valid.
  - Expanding DELETE_TID doesn't work well either.

HAMMER Reblocking

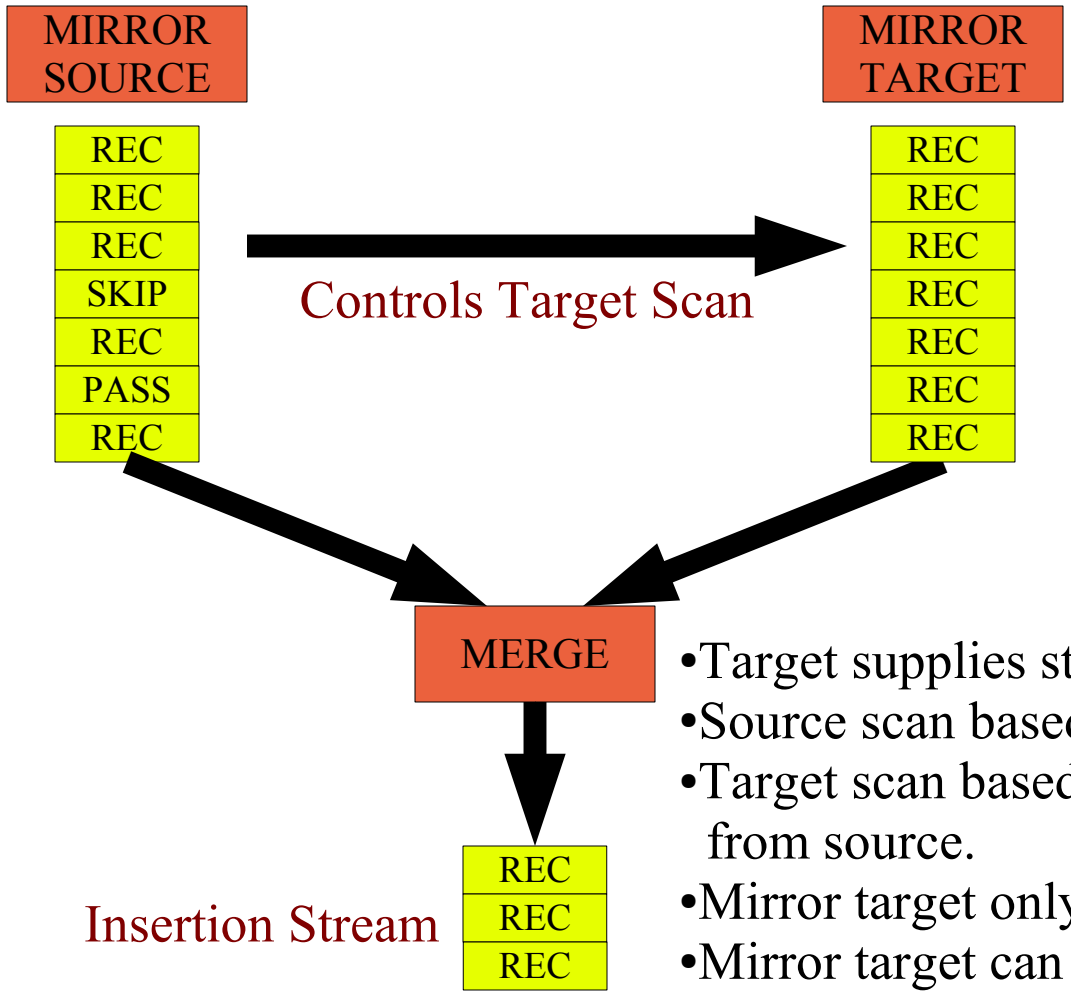


- Allocations are byte granular, but not individually tracked.
- Big-blocks cannot be reused until they are 100% empty.
- Large numbers of deletions can make big-blocks available for re-use.
- Otherwise use reblocking to re-pack data and meta-data into new big-blocks.

HAMMER Mirroring Streams



HAMMER Mirroring Streams

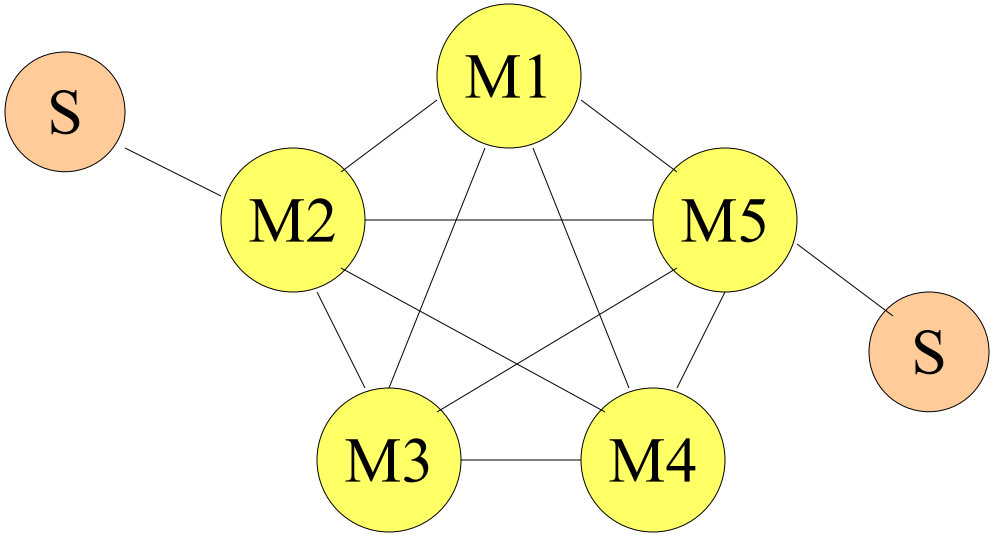


- Target supplies starting TID to source.
- Source scan based on MIRROR_TID.
- Target scan based on mirroring stream from source.
- Mirror target only inserts records, no deletes.
- Mirror target can manage its own snapshots.

HAMMER – Ongoing work

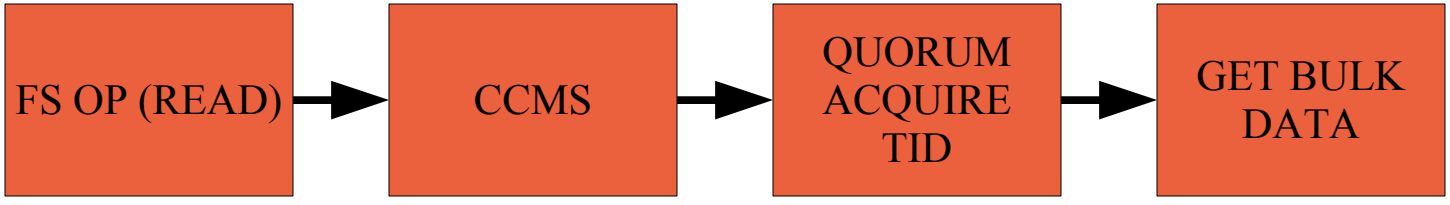
- Add catastrophic recovery scan (2.2).
- Better localization for allocations, and more intelligent reblocking.
- Possible expansion of B-Tree from 63-way to 255-way.
- Adjust directory hash from straight crc to semi-ordered + crc (2.2).
- Implement forward log for short lseek+write+fsync sequences.
- Serialize UNDO buffers to avoid volume header update.
- Give each PFS its own B-Tree to improve integrity.
- Dynamic data extents.
- Support direct data overwrite for things like swap, memory files.
- Ability to add, remove, expand, and contract volumes while live.
- Shared data references (efficient cp, tree duplication, etc).
- Support recursive CRC on B-Tree when filesystem integrity is paramount. Expansion of MIRROR_TID algorithm.
- Ultimate Goal for DragonFly – network-clustered / multi-master replication.

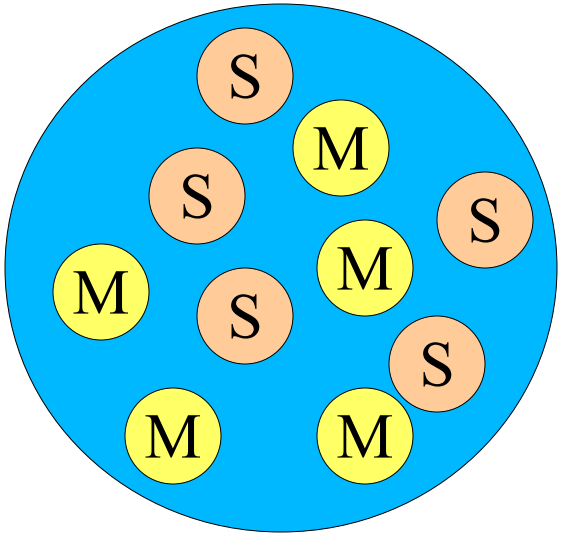
Quorum Protocol Basics



- Quorum is any 3 out of 5 masters in example.
- Once you have a quorum you can ignore any others.
- Robust: Can query all 5 and take first three responders.
- Can be used for transaction and locking protocols.
- Modifications need only be made to 3 masters.
- Multi-master replication to synchronize all nodes.
- Bulk data can be tagged, quorum operations can agree on the tag, then the data can be retrieved from ANY single node containing that tag.
- Robust: Can retrieve from multiple nodes and take first responder.

HAMMER's Final Goal





- Bulk synchronization can be handled by existing mirroring protocols.
- HAMMER transaction id (TID) *is* the data tag.
- All quorum protocol features are supportable.
- Rules can be loosened depending on application.
- Cache Coherency protocols will be very complex.
- VFS Quorum protocols will be very complex.