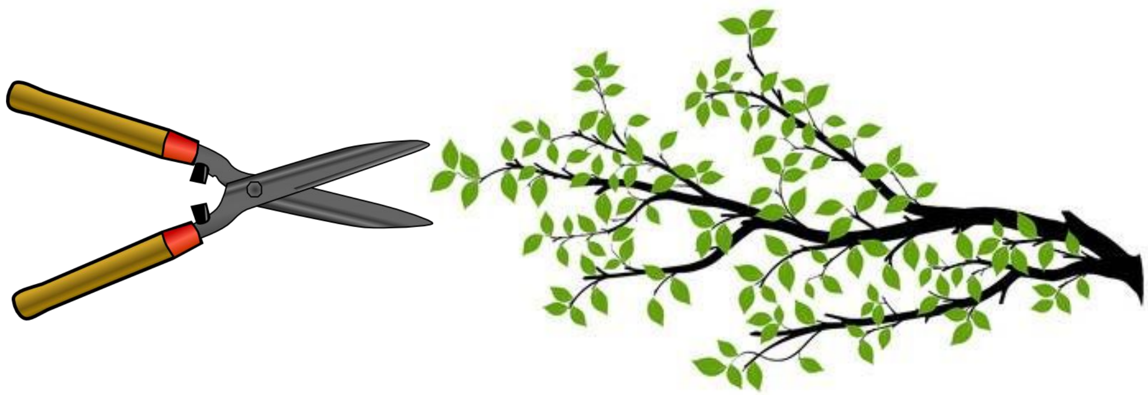
SHEARS: Persisting Deletes in LSM Trees

Megan Fantes, Ketill Guðmundsson, Nikhilesh Murugavel, Allison Weaver



Introduction & Background

Introduction

- Unoptimized deletes in LSM trees
- Deleted value is logically hidden
- **Problem?**
    - Increased tree size
    - Compromise in security
- **Goal?**
    - Faster persistent deletes

Background

Mem-table: Data structure (skip-list) in memory

SSTs: Sorted Sequence Table in disk

Out-of-place updates: When data is updated, it is added as a new key-value pair in the mem-table

Tombstones: Used to mark a key as deleted

Persistent Deletes: When a deleted key-value pair is removed from the tree

Compaction/Partial Merging: Some of level-L SSTs merged with level-(L+1) SSTs

Our Solution

SHEARS Design

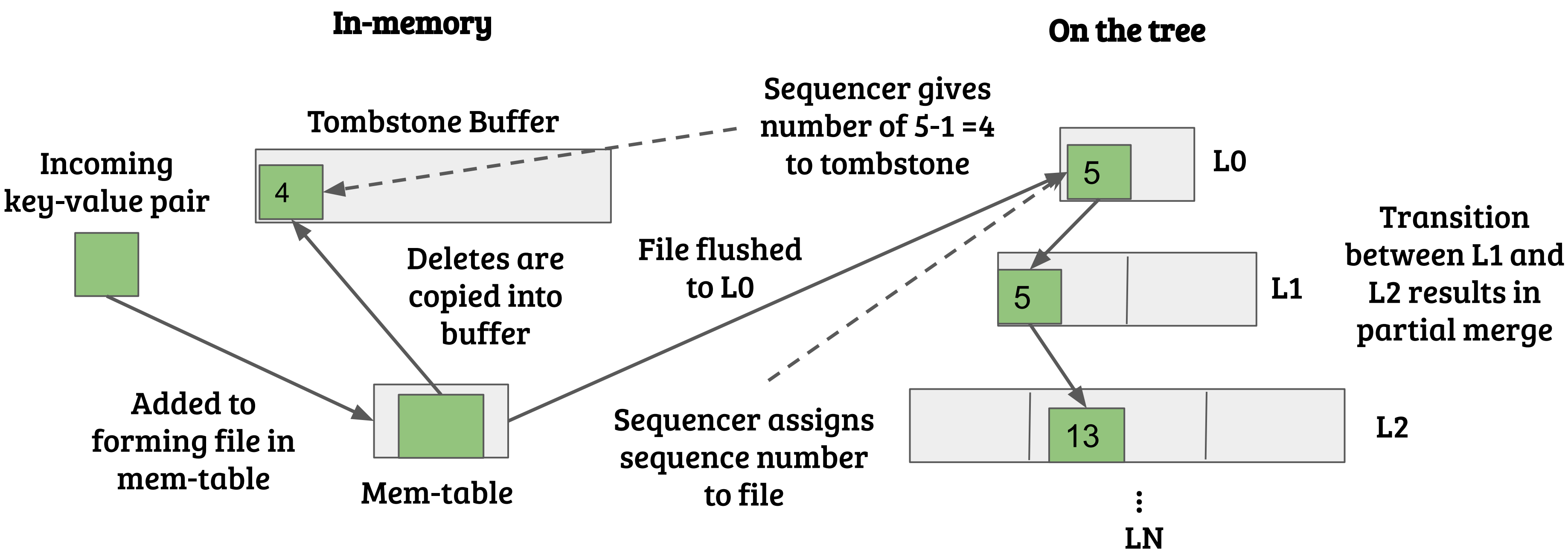
Our additions:

- Sequencer
- Sequence Number
- Tombstone buffer
- Tombstone Group

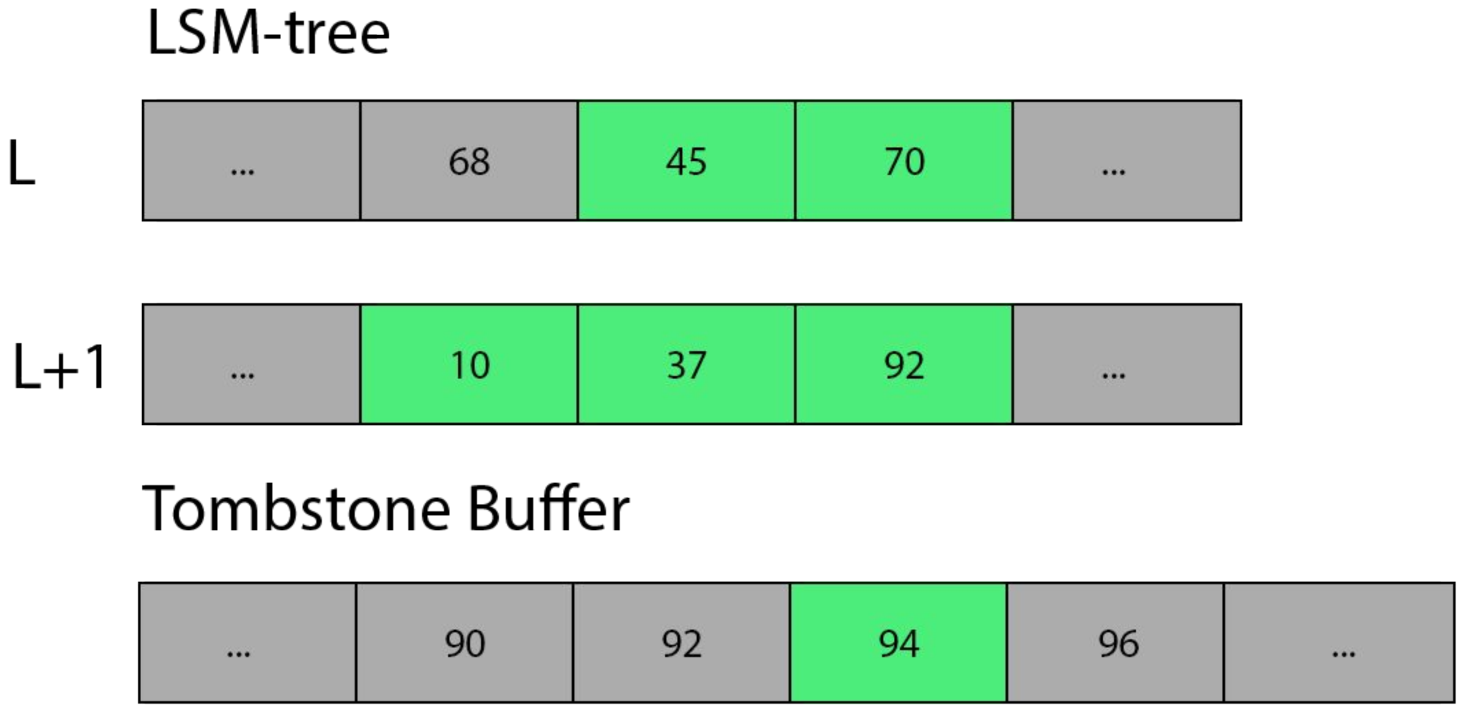
What it does:

- Three Way merge
- Inserts deletes
- Does not guarantee persistent deletes
- Better read performance and increases storage space

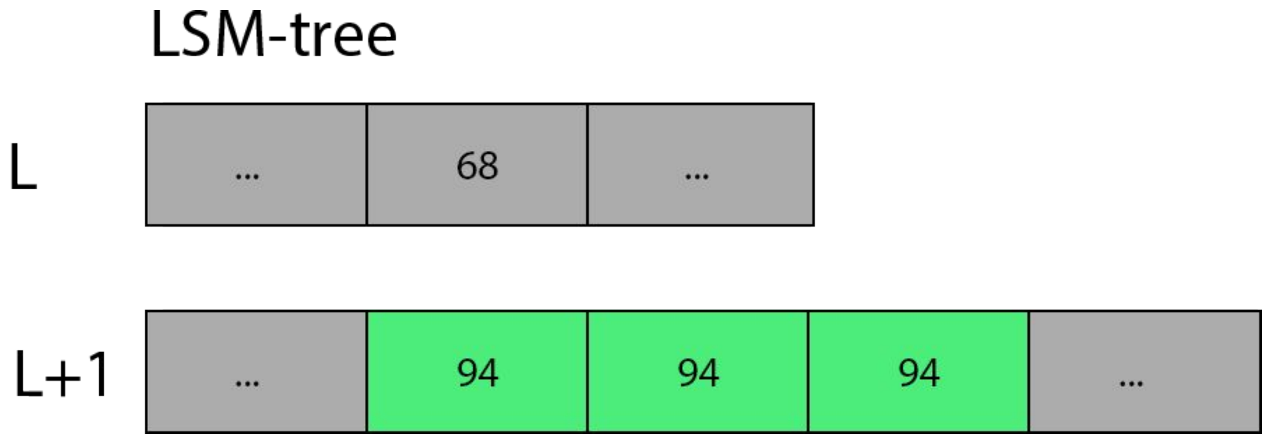
How it works:


In-memory
On the tree
Tombstone Buffer
Incoming
key-value pair
4
Sequencer gives
number of 5-1 =4
to tombstone
5
L0
Transition
between L1 and
L2 results in
partial merge
Deletes are
copied into
buffer
File flushed
to L0
5
L1
Added to
forming file in
mem-table
Mem-table
Sequencer assigns
sequence number
to file
13
L2
⋮
LN

Merge Policy



Merge Policy



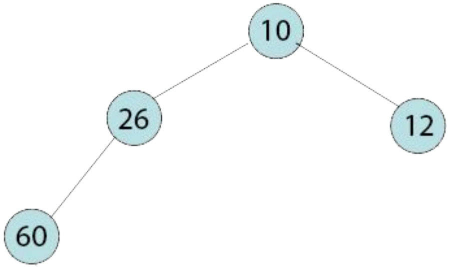
K-way Merge

- Merge many tombstone groups in a compaction
- Makes system more fluid
- Downside is higher compaction cost
- Possible with the use of priority cue
- Additional cost is now O(n log k)

K-way Merge

Tombstone Buffers

X	10	12	54	72	...
X	X	26	30	40	...
12	18	58	82	94	...
X	X	X	60	66	...



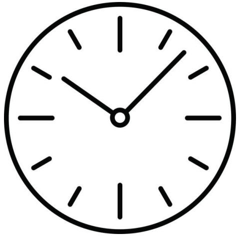
Deleting Tombstone Groups

- Keep track of lowest sequence numbers in LSM tree
  - If any tombstone group has lower number then delete
  - Costly operation
  - Runs in background periodically
- Backup systems in place of overflow
  - Deletions happening too slow
  - Discard oldest tombstone groups

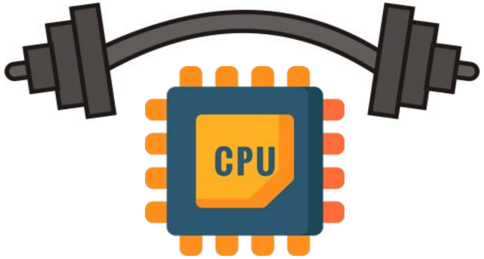
Experiments

Experiments: Storage, Latency, CPU Load

- Hypothesis: SHEARS uses more memory, less disk space

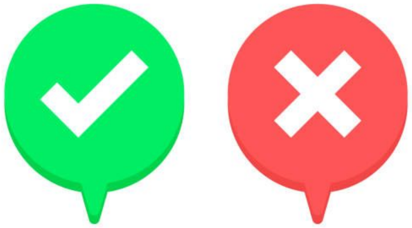


- Hypothesis: SHEARS increases write latency, decreases read latency



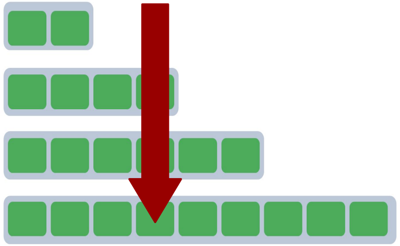
- Hypothesis: sorting, merging increases CPU load

Experiments: Bloom filters, SeqNum Distribution



- Hypothesis: SHEARS decreases false positives by pruning LSM tree
- Insight: tracking the min. SeqNum to define delete policy

Experiments: Force deletes, delete persistence



- Trade-offs:
  increased CPU/IO cost, memory used, delete persistence

- Hypothesis:
  SHEARS persists deletes faster (that is the point)

Thank You

Questions?