ACCORDION: BETTER MEMORY ORGANIZATION FOR LSM KEY-VALUE STORES

Edward Bortnikov, Anastasia Braginsky, Eshcar Hillel, Idit Keidar, Gali Sheffi
Yahoo Research, Oath
NOSQL PERSISTENT KV-STORES ARE POPULAR
THE QUEST

In-Memory Database Performance + Reliable Persistent Storage
LOG-STRUCTURED MERGE (LSM) KEY-VALUE STORES

Random I/O $\Rightarrow$ Sequential I/O

- Frequent compactions
- Space/Write amplification
- Fragmented memory layout
WHAT IS ACCORDION?
WHAT IS ACCORDION?

Novel write-path algorithm for LSM stores

- Flat memory representation
  - Memory footprint ↓, GC overhead ↓, cache friendly ✓, off-heap friendly ✓

- Better performance of write-intensive workloads
  - Disk compaction ↓, write throughput ↑

- Better disk utilization
  - Write amplification ↓, Disk wear ↓
LSM IN **HBASE**

Data updates stored as versions
Compaction eliminates redundancies
Reads access all components

![Diagram](image-url)
ACCORDION: IN-MEMORY LSM

In memory compaction (IMC) flush mutable to immutable segment
FLAT IMMUTABLE INDEX

Lean footprint
The smaller the key/values the better!
SERIALIZED INDEX:
KV OBJECT ELIMINATION

Even leaner footprint

Off-Heap friendly
IMC POLICIES

**Basic** only flattens and merges indices

**Eager** Basic + always eliminates duplications

**Adaptive** Basic + occasionally eliminates duplications
WRITE THROUGHPUT

50GB Dataset, 100% Writes
Zipf Distribution, 4x25B Values
WRITE VOLUME

**HDD**

- none: 151,765
- basic: 136,283
- adaptive 0.5: 119,527
- adaptive 0.35: 107,142
- adaptive 0.2: 98,123

- flush size
- compaction size

**SSD**

- none: 147,127
- basic: 120,315
- adaptive 0.5: 114,265
- adaptive 0.35: 107,168
- adaptive 0.2: 92,303

- flush size
- compaction size

-29%  
-30%
WRITE THROUGHPUT SPEEDUP

- On-heap: +33%
- Off-heap: +44%
READ LATENCY SPEEDUP/SLOWDOWN

HDD

SSD

30GB Dataset Zipf Distribution
Background writes, 4x25B Values

-15-40%
SUMMARY

Accordion = a leaner and faster write path

- Flat memory layout $\Rightarrow$ less GC
- Space-Efficient Index + Redundancy Elimination $\Rightarrow$ less I/O
- Less Frequent Flushes $\Rightarrow$ increased write throughput
- Less On-Disk Compaction $\Rightarrow$ reduced write amplification
- Data stays longer in memory $\Rightarrow$ reduced tail read latency

Available HBase 2.0+
THANK YOU!  Questions ?