2021 USENIX Annual Technical Conference
JULY 14–16, 2021
Virtual Event

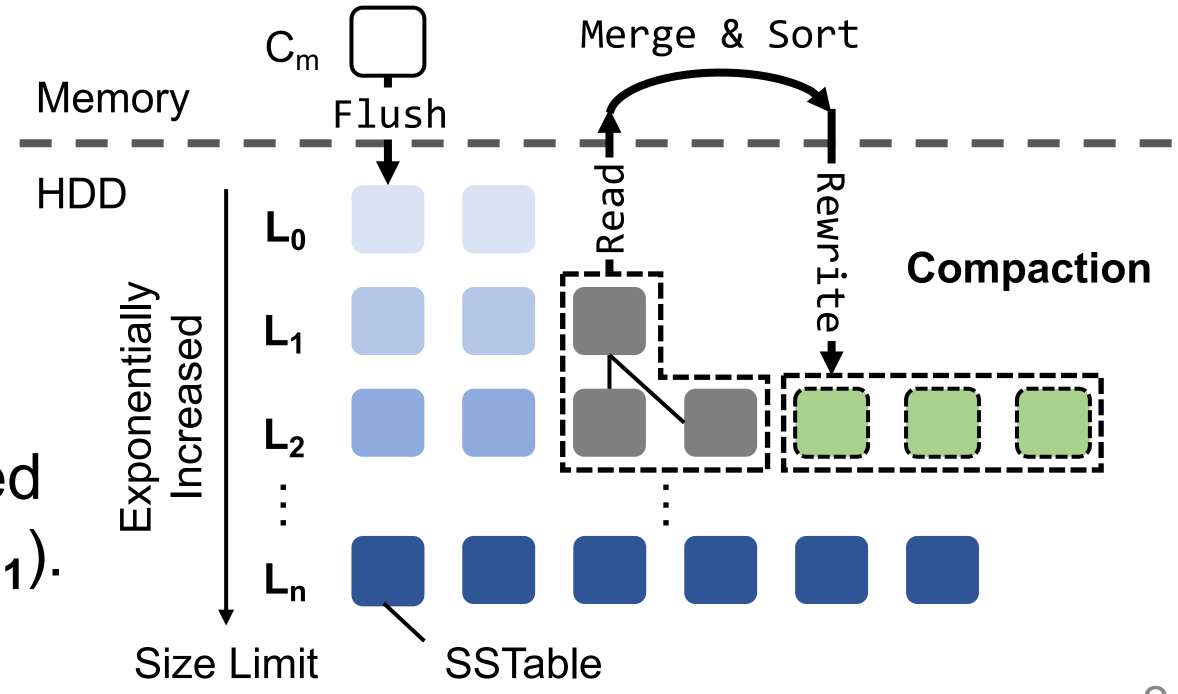
KVIMR: Key-Value Store Aware Data Management Middleware for Interlaced Magnetic Recording Based Hard Disk Drive

Yuhong Liang, Tsun-Yu Yang, and Ming-Chang Yang

Memory And Storage System (MASS) Lab,
Department of Computer Science and Engineering,
The Chinese University of Hong Kong (CUHK)

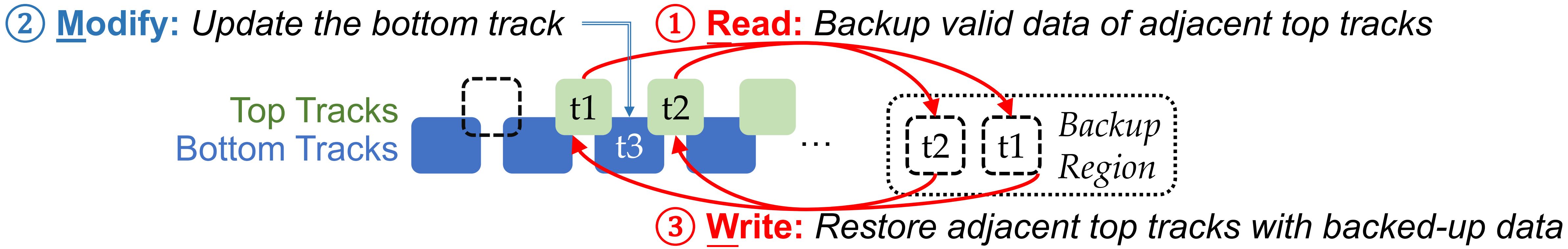
LSM-tree based KV Store

- **Log-Structured Merge-Tree (LSM-tree)** inspires many well-known **key-value (KV) stores**.
  - Examples: RocksDB (Facebook), LevelDB (Google), and HyperLevelDB.
  - It delivers **high write throughput** on the mechanical **hard-disk drive (HDD)**.
- LSM-tree based KV stores share the following two design concepts:
  1) KV pairs are organized as SSTables of **multiple levels** in the disk.
  2) The **compaction thread** merges and sorts an SSTable (**L_i**) with all overlapped SSTables (**L_{i+1}**) into new SSTables (**L_{i+1}**).
     - Whenever **L_i** exceeds its size limit.



Interlaced Magnetic Recording

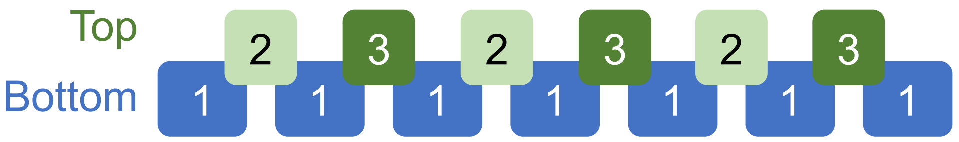
- **IMR** offers an opportunity to construct a **cost-effective KV store**.
  - It organizes top tracks and bottom tracks in an "interlaced" way to deliver higher areal density and lower cost-per-GB for HDD.
  - Every top track can be freely written.
  - Writing a bottom track may **damage** the data of adjacent top tracks.
- **Read-Modify-Write (RMW)** [1] is introduced to rewrite top tracks (if needed) when updating a bottom track:



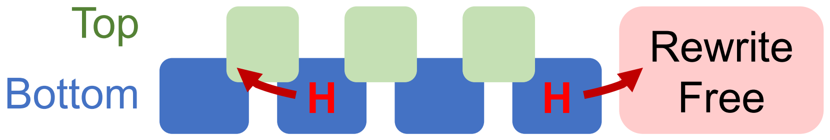
RMW is expensive and time-consuming!

Existing Solutions to Reduce RMWs

- **Direction 1**: **Allocating** tracks for accommodating data based on different space usages.



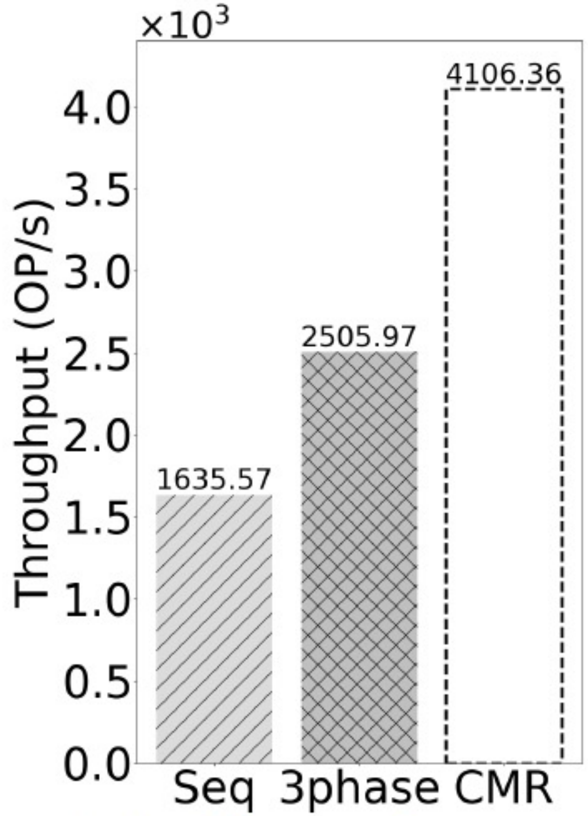
  - Three-Phase Write Management [2], [3]:
    - Phase 1 (0%~50%): Write to bottom tracks only (**no** top track rewrite);
    - Phase 2 (50%~75%): Write to every other top tracks (**at most 1** top track rewrite);
    - Phase 3 (75%~100%): Write to rest of top tracks (**at most 2** top track rewrites).
  - **Noticeable degradation** due to the lack of knowledge about LSM-tree KV store.
- **Direction 2**: **Migrating** hot data into rewrite-free area or top tracks.
  - E.g., Track Flipping [1], Selective Track Caching [1], Top Buffer [4], and Block Swap [4].



  - **Inapplicable** to LSM-tree based KV stores (since SSTables are written once but never updated before being deleted).

KV Store on IMR: A Cost-Effective and High-Throughput Solution?

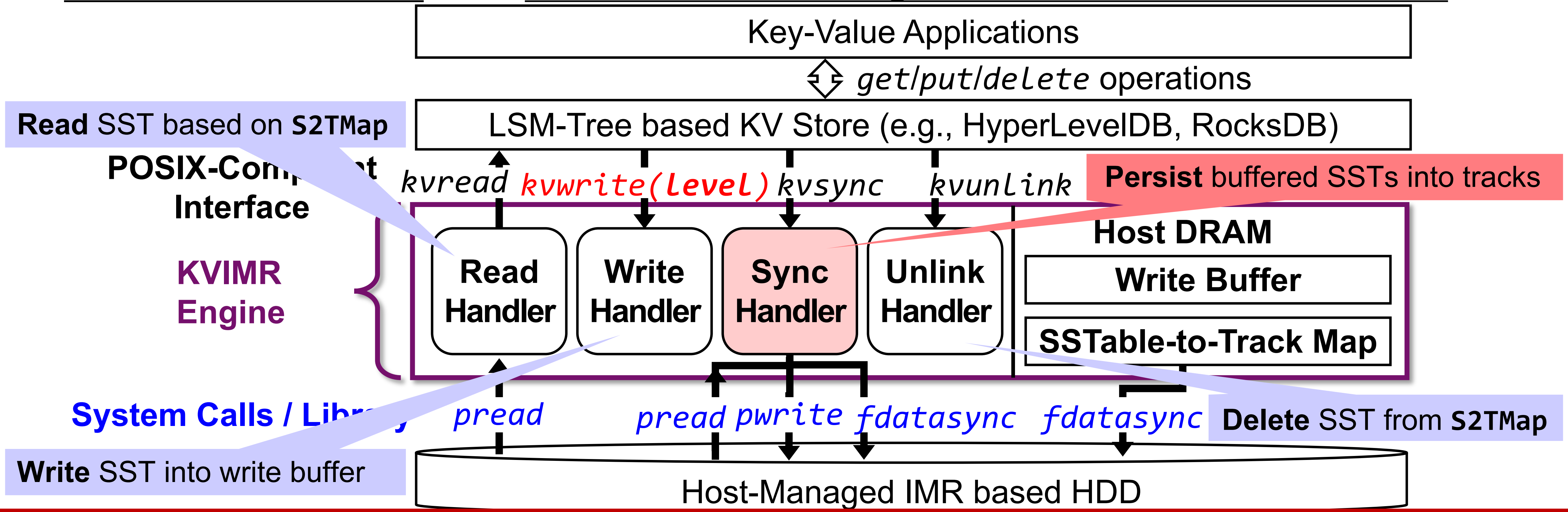
- We deploy RocksDB on an 100 GB **emulated IMR HDD**.
  - **Track Allocation**: The state-of-the-art three-phase (**3Phase**) and the classical sequential (**Seq**) schemes are implemented.
- We also deploy RocksDB on an 100 GB **CMR HDD** with tracks allocated based on 3Phase (**CMR**).
- 75 millions of 1 KB KV pairs, generated by YCSB, are randomly inserted into RocksDB.
  - **3phase** achieves **1.53X** higher throughput than **Seq**.
  - **3phase** still suffers **38.97% noticeable degradation on throughput** when compared to that of **CMR**.
    - RMW accounts for **57.74%** of the total time for persisting SSTables.



The STOA IMR design lacks for the knowledge about KV store!

KVIMR: Key-Value Store Aware Data Management Middleware for IMR

- **KVIMR** is architected as a **middleware** to facilitate the support for various KV stores and the efficient management on IMR based HDD.



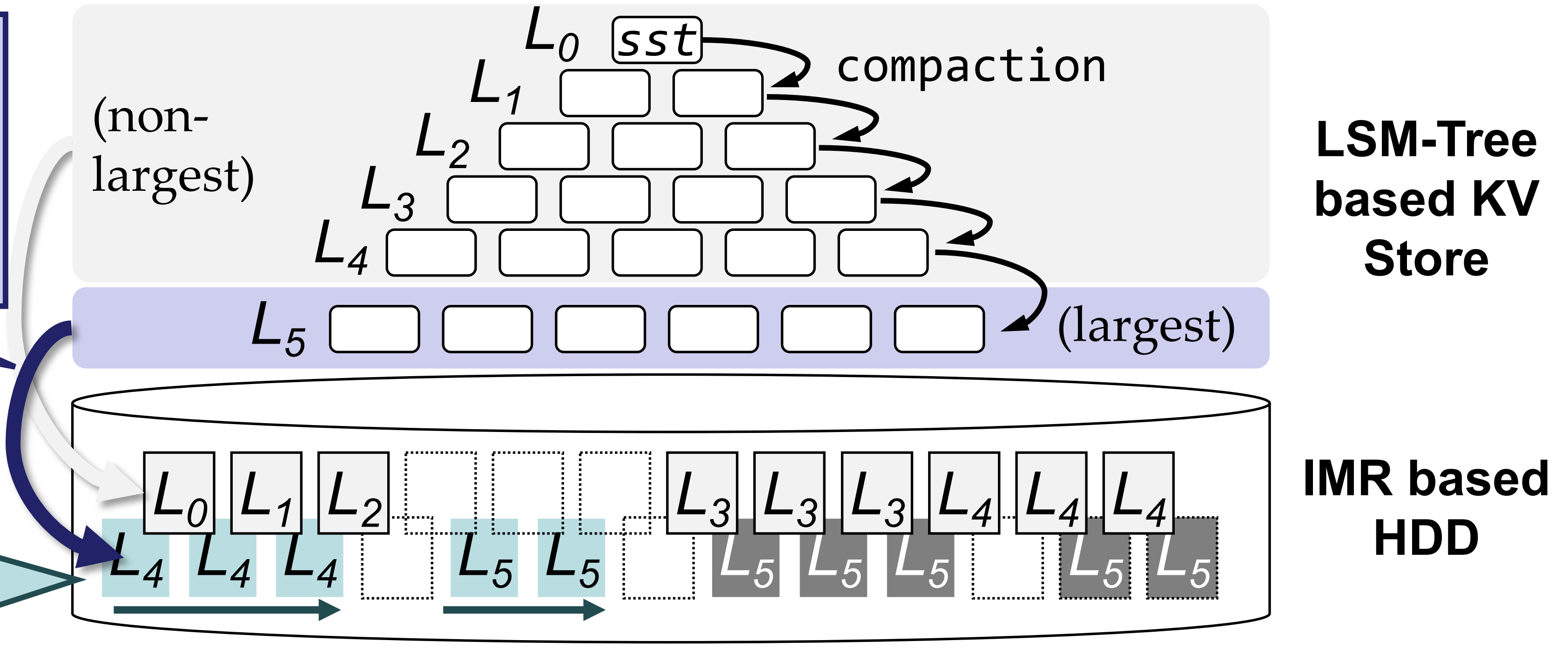
The "level" information of SST is passed down as a key clue!

Compaction-aware Track Allocation

- We leverage the **special properties behind the compaction** on allocating IMR tracks based on the **"level"** information of SSTs.
  - **Compaction Frequency**: *The lifespan of larger-level SSTs may be longer.*
  - **Compaction Locality**: *SSTs are often created and compacted together.*

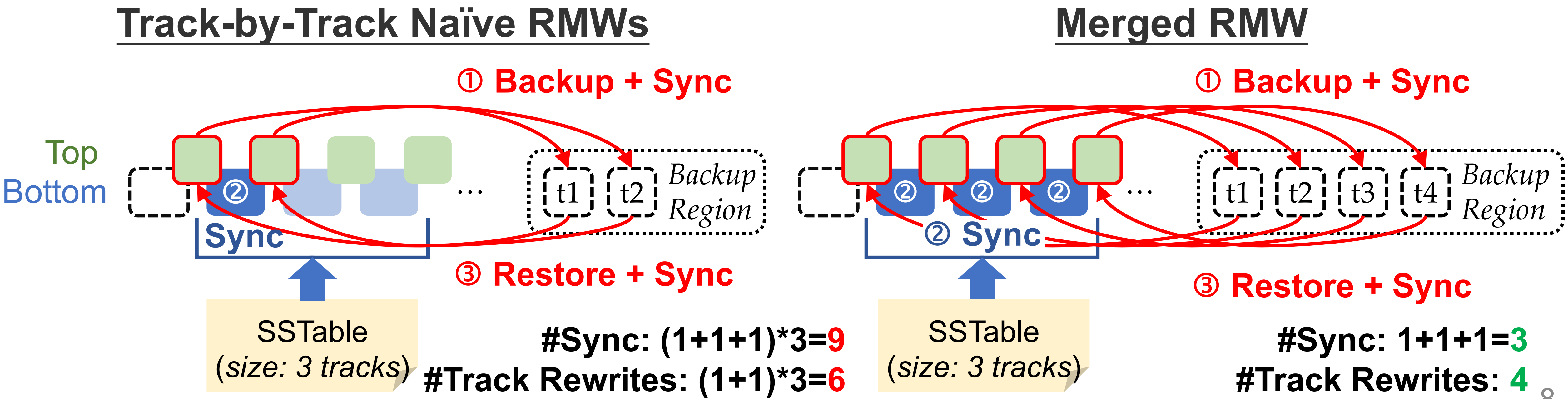
Key Idea #1: Bottom tracks shall be allocated for SSTs of larger levels to avoid RMWs.

Key Idea #2: SSTs shall be written into top tracks or bottom tracks as sequential as possible.



Merged Read-Modify-Write

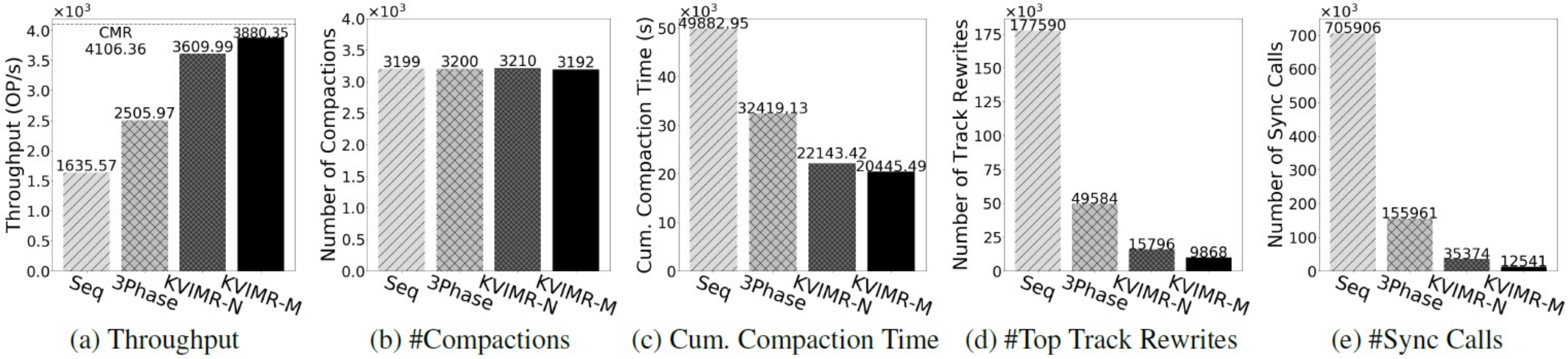
- KVIMR employs a novel **Merged RMW** to efficiently persist an SSTable, which is typically of multiple tracks, into IMR tracks.
  - Its key idea is to **re-order** multiple track-by-track naïve RMWs into a **"merged"** RMW to **reduce the sync functions and track rewrites**.
    - Sync-like function ensures the data are persisted into tracks against crashes.



Evaluation Setup

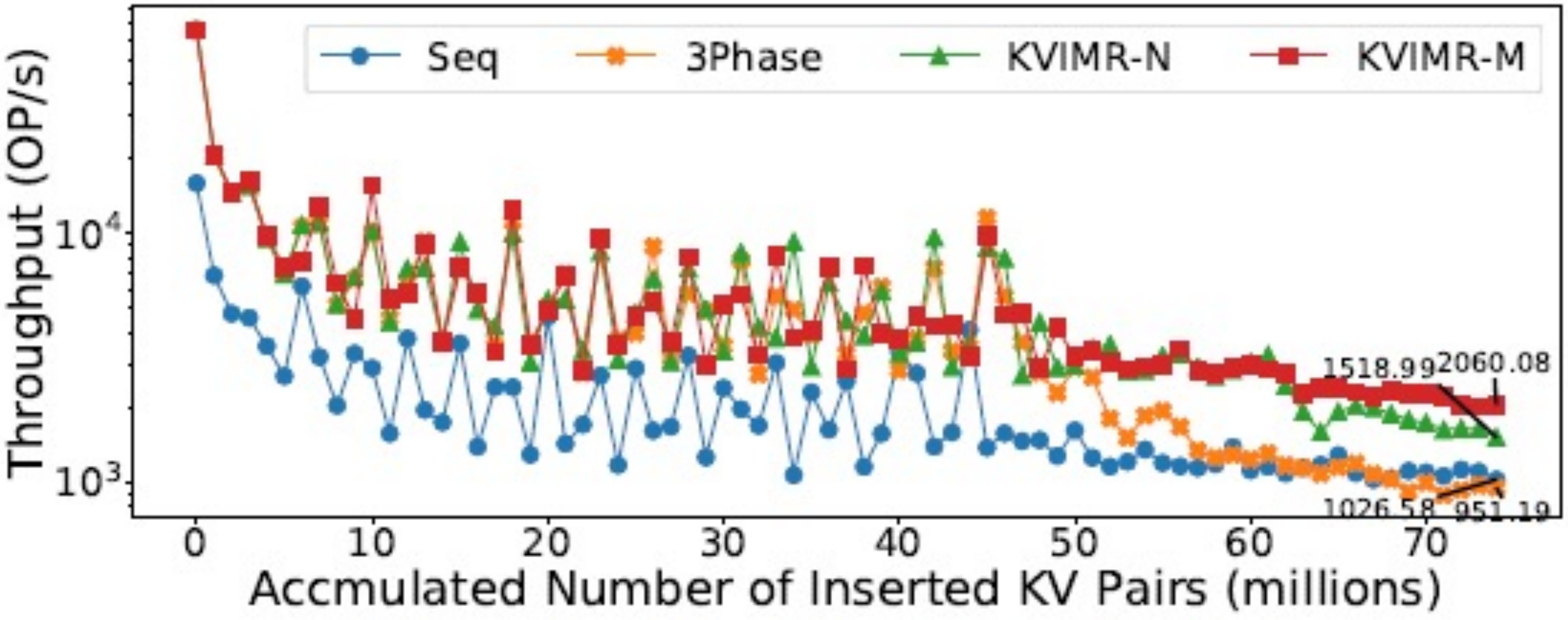
- **Emulating 100 GB IMR based HDD** on a CMR based HDD.
  - The results can reflect actual performance of the disk internal activities.
  - Track size is set to 2 MB.
- **RocksDB**, **LevelDB**, and **HyperLevelDB**, are modified to interface the proposed **KVIMR middleware** (just about 100 LOCs per KV store).
  - The SSTable size is set to 64MB.
- The following schemes are implemented in KVIMR for managing IMR:
  - **`Seq`**: allocates tracks in a sequential order and adopts **`Naïve RMW`**.
  - **`3Phase`**: allocates tracks based on three phases and adopts **`Naïve RMW`**.
  - **`KVIMR-N`**: adopts **`Compaction-Aware Allocation`** and **`Naïve RMW`**.
  - **`KVIMR-M`**: adopts **`Compaction-Aware Allocation`** and **`Merged RMW`**.

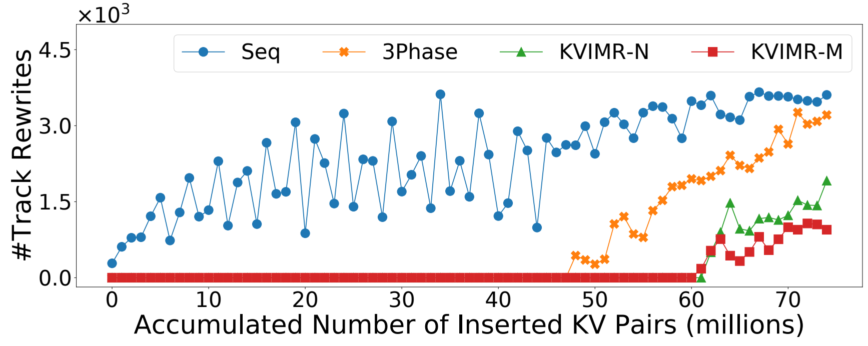
Evaluation Results (1/3)

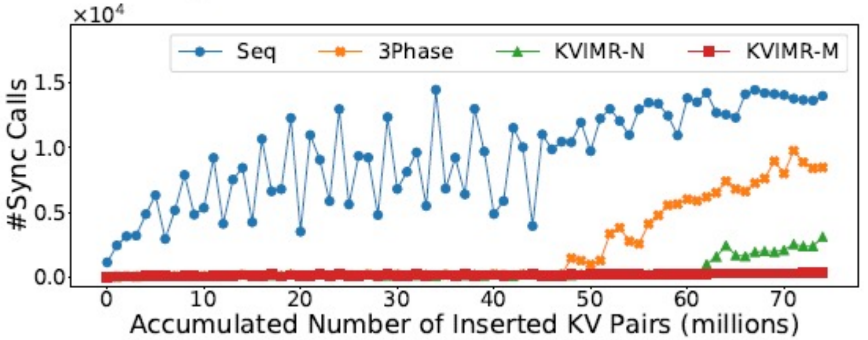


- **KVIMR-M/KVIMR-N** achieves significant throughput improvements.
  - **KVIMR-M** approaches the throughput of **CMR** with only about **5.5% degradation**.
- All schemes share similar number of compactions.
- **KVIMR-M/KVIMR-N** reduces the cumulative compaction time, incurs much less numbers of track rewrites and sync calls.

Evaluation Results (2/3)



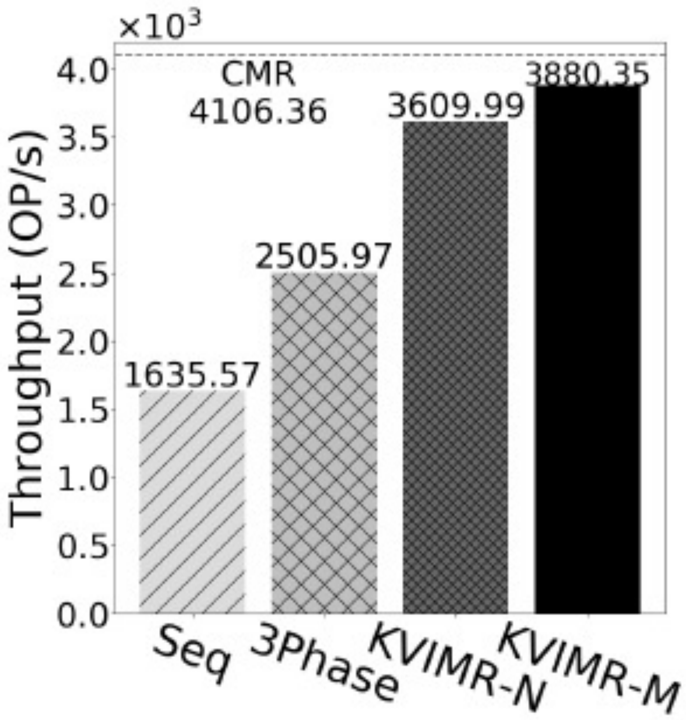




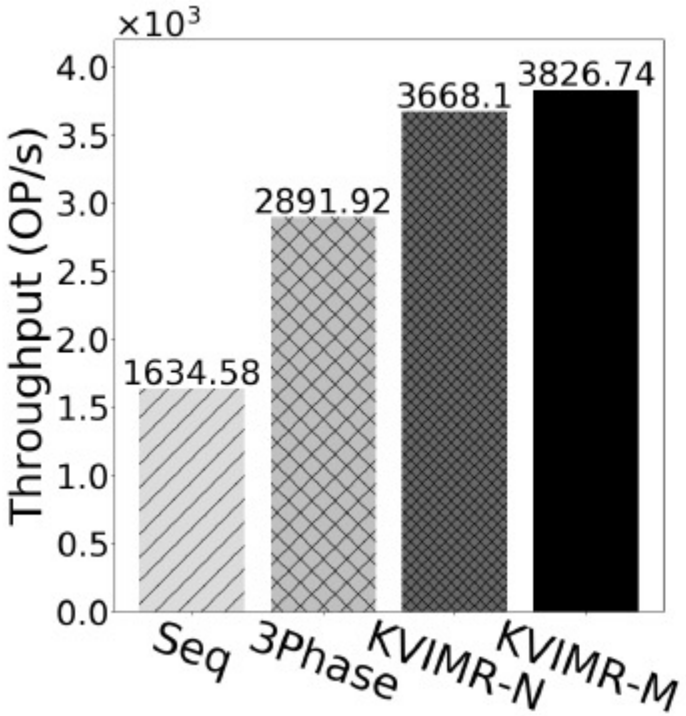
- **KVIMR** leads to much higher throughputs, almost for every 1 million of inserted KV pairs along the whole loading.
- After inserting about 60 millions of KV pairs, **KVIMR-M** starts to achieve the highest throughput than the rest.

Evaluation Results (3/3)

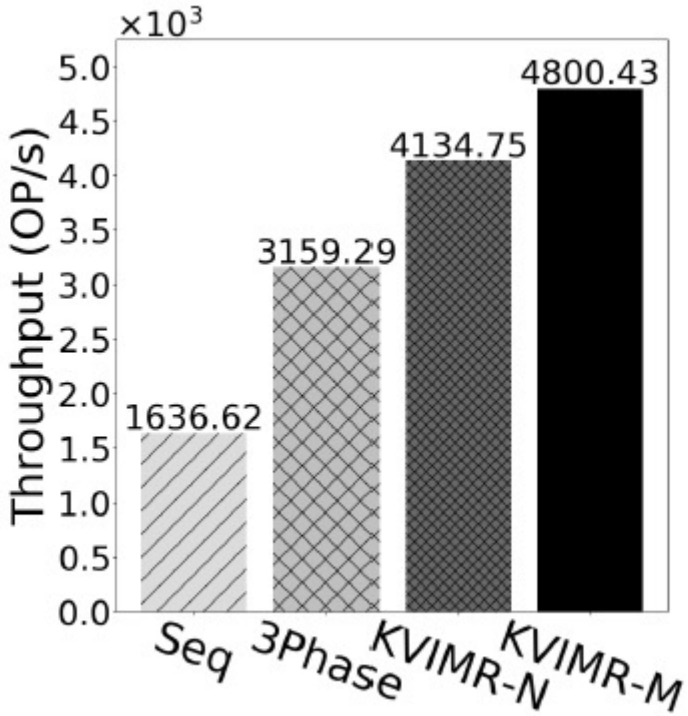
RocksDB



LevelDB



HyperLevelDB



- **`KVIMR`** demonstrates its **good compatibility** for improving the throughputs for various modern LSM-tree based KV stores.

Conclusion

- This paper presents **KVIMR**, a data management middleware, to construct a **cost-effective yet high-throughput** LSM-tree based KV store on IMR based HDD.
  - **Compaction Aware Track Allocation** minimizes the time-consuming RMWs and efficiently access SSTables during the compaction.
  - **Merged RMW** further improves the efficiency of persisting an SSTable when the time-consuming RMWs are inevitable.

- Our evaluations on three well-known LSM-tree based KV stores reveal that KVIMR improves the **overall throughput** by up to **1.55X** and even achieves **2.17X higher throughput** under high space usage.

References

[1] M. H. Hajkazemi, A. N. Kulkarni, P. Desnoyers, and T. R. Feldman, “Track-based translation layers for interlaced magnetic recording,” USENIX ATC 19.

[2] F. Wu, B. Li, B. Zhang, Z. Cao, J. Diehl, H. Wen, and D. H. Du, “Tracklace: Data management for interlaced magnetic recording,” IEEE Transactions on Computers, 2020.

[3] K. Gao, W. Zhu, and E. Gage, “Write management for interlaced magnetic recording devices,” US Patent 9,508,362.

[4] K. Gao, W. Zhu, and E. Gage, “Interlaced magnetic recording,” US Patent 9,728,206.

Thanks for Your Attention!

Yuhong Liang, Tsun-Yu Yang, and Ming-Chang Yang

{**yhliang**, yangty, mcyang}@cse.cuhk.edu.hk

Memory And Storage System (MASS) Lab,
Department of Computer Science and Engineering,
The Chinese University of Hong Kong (CUHK)