MatrixKV: Reducing Write Stalls and Write Amplification in LSM-tree Based KV Stores with a Matrix Container in NVM

Ting Yao[1], Yiwen Zhang[1], Jiguang Wan[1], Qiu Cui[2], Liu Tang[2], Hong Jiang[3], Changsheng Xie[1], and Xubin He[4]

[1]Huazhong University of Science and Technology, China

[2]PingCAP, China

[3]University of Texas at Arlington, USA

[4]Temple University, USA

Outline

- **Background and Motivations**
- MatrixKV
- Evaluation
- Conclusion

LSM-tree based Key-value stores

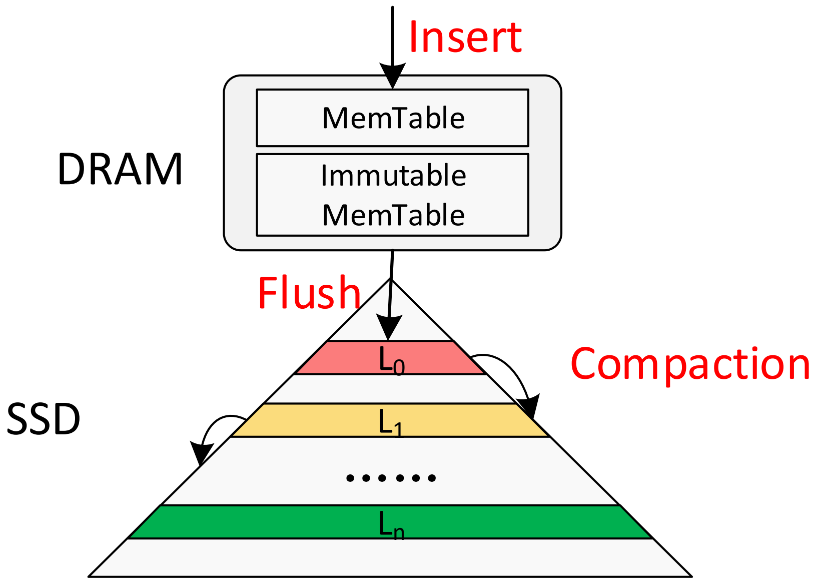
- Log-structured merge tree (LSM-tree)
  - Write intensive scenarios

- Applications :

LEVELDB cassandra APACHE HBASE RocksDB Google BigTable

- Properties:
  - Batched sequential writes: high write throughput
  - Fast read
  - Fast range queries

LSM-tree and RocksDB

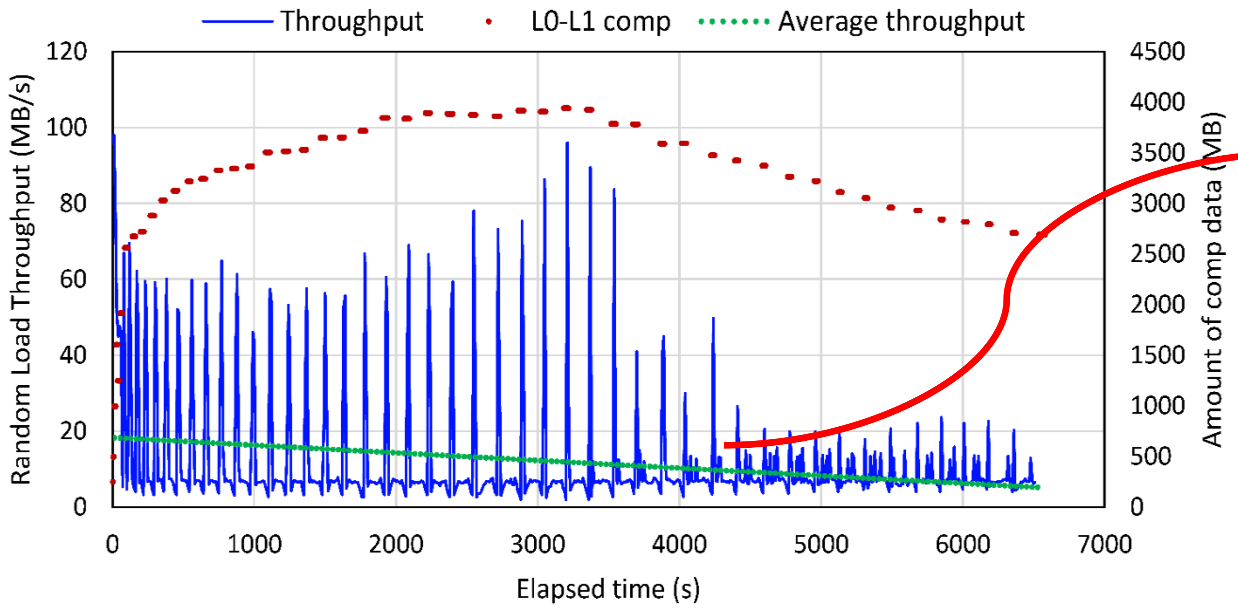


SSD based RocksDB

- Systems with DRAM-SSD storage
- Exponentially increased level sizes (AF)
- Operations
  1. Insert
  2. Flush
  3. Compaction between L_i-L_{i+1}
     - L0-L1 compaction
     - L1-L2 compaction
     - ……

Challenge 1: Write stall

Random write an 80 GB Dataset to an SSD based RocksDB. (20 million KV items, 16byte-4KB)



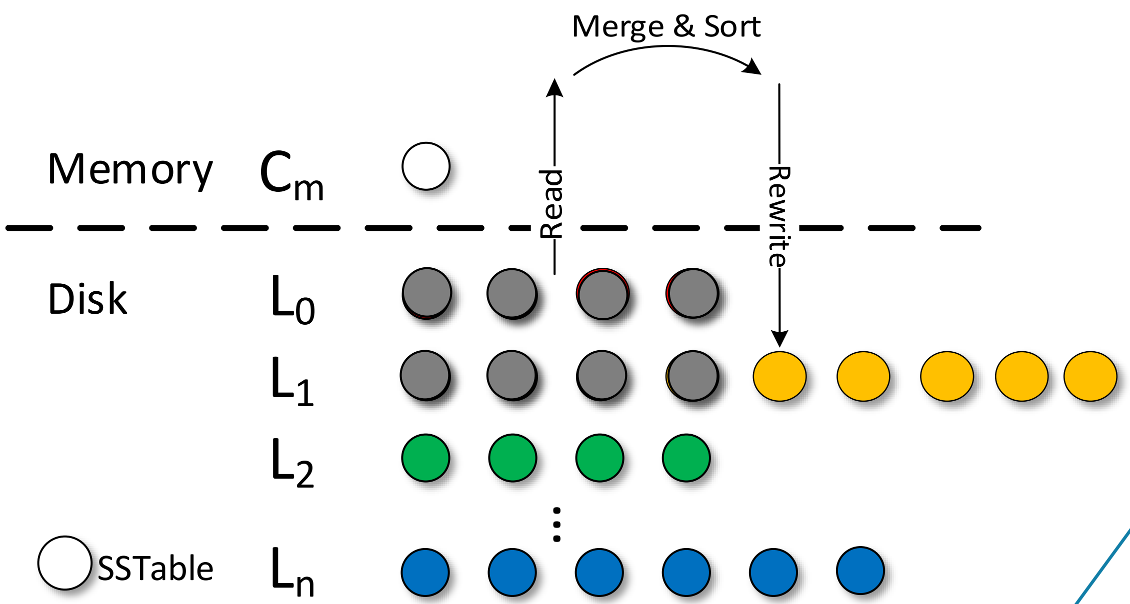
Write stall: Application throughput periodically drop to nearly zero.

- Unpredictable performance.
- Long tail latency.

L0-L1 compaction!

3.1GB compaction data.

Root cause of write stall: L0-L1 compaction



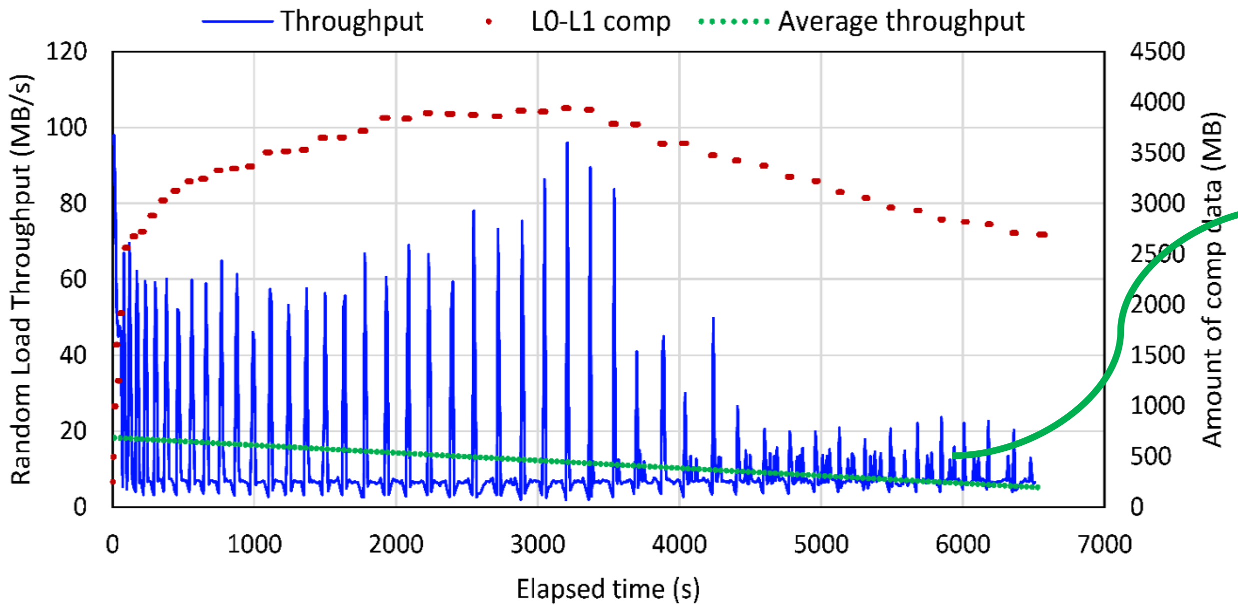
CPU cycle.

SSD bandwidth.

L0-L1 compaction: The all-to-all coarse-grained compaction

Challenge 2: write amplification

Random write an 80 GB Dataset to an SSD based RocksDB. (20 million KV items, 16byte-4KB)



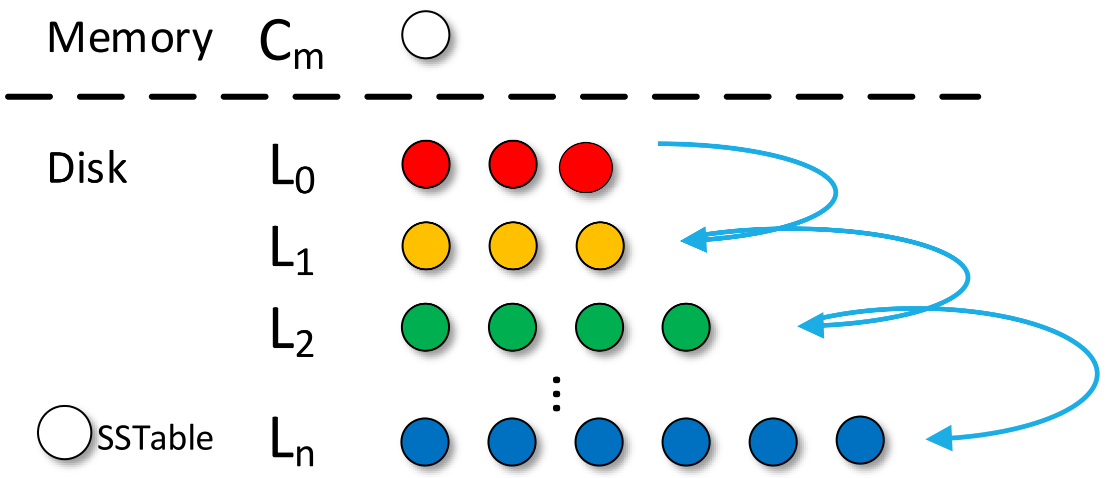
Write amplification: Average throughput decreases gradually.

- **Decreased performance.**

Increased LSM depth!

More compaction and higher WA

Root cause of increased write amplification



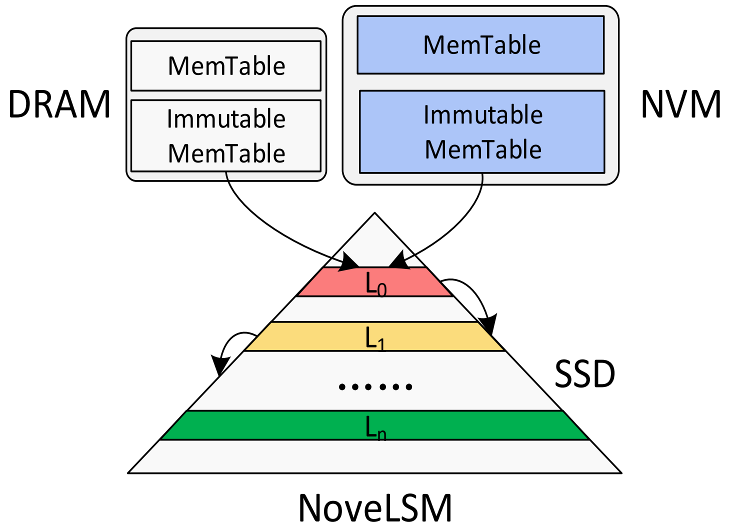
- Level by level compactions: Write amplification increases with the depth of LSM-trees.
- WA=AF * N;

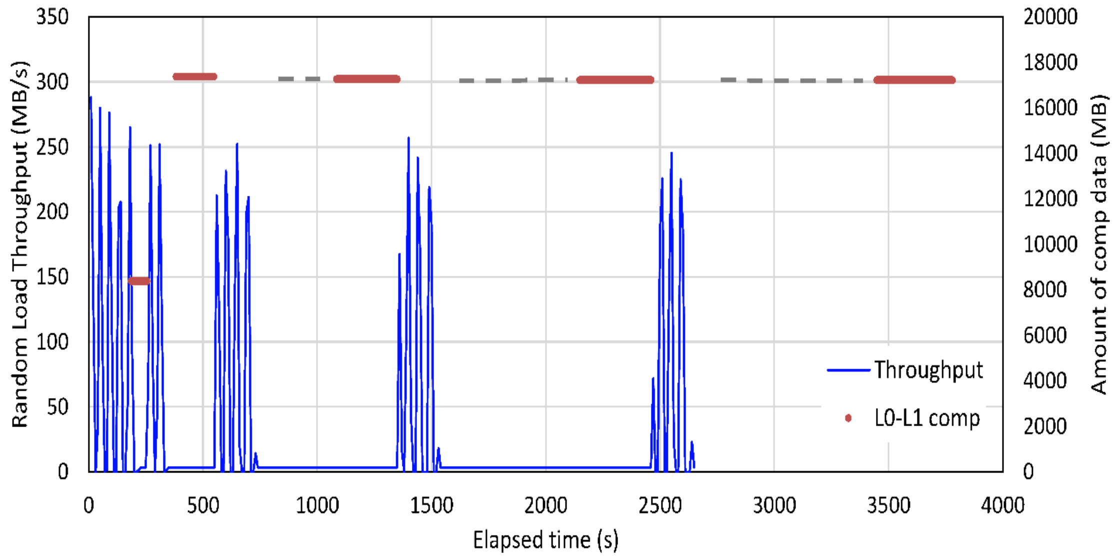
  AF is the amplification factor of adjacent two levels. (AF=10)

  N is the number of levels.

State-of-art solution with NVM

- **NVM is byte-addressable, persistent, and fast!**
- NoveLSM: Adopting NVM to store large mutable MemTable.
- 1.7x higher random write performance but more severe write stalls!





Sudarsun Kannan, Nitish Bhat, Ada Gavrilovska, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. Redesigning lsms for nonvolatile memory with novelsm. In 2018 USENIX Annual Technical Conference (ATC18), 2018.

Motivation

All-to-all L0-L1 compaction

↓

Write stall

↓

Unstable performance

Increased depth

↓

Higher write amplification

↓

Decreased performance

↓

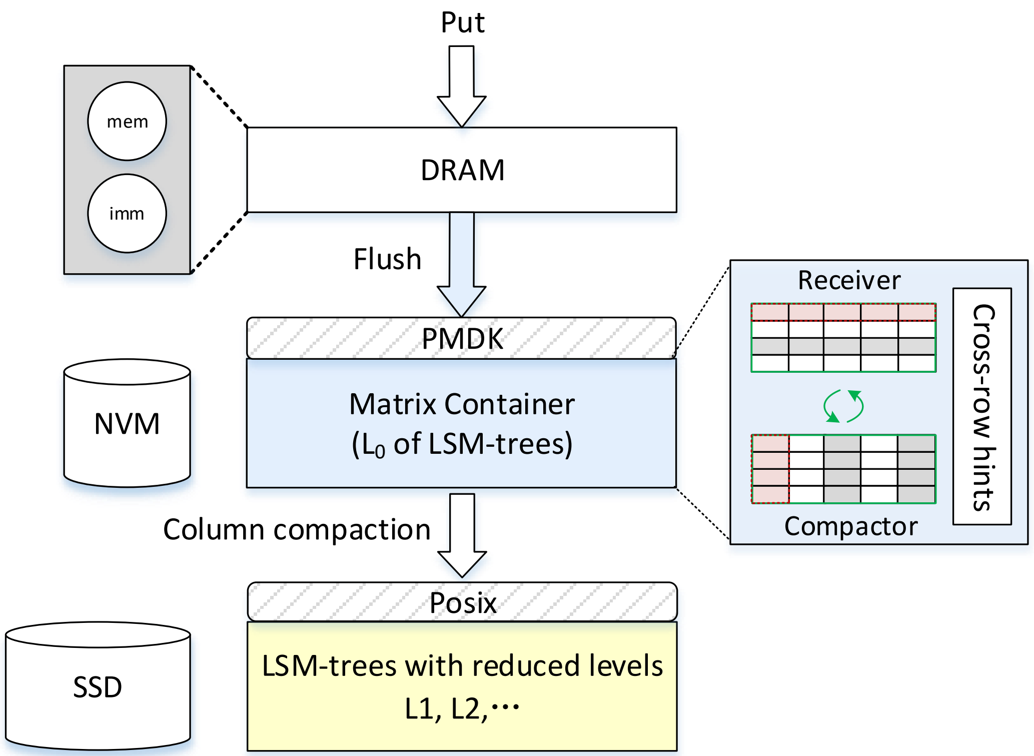
MatrixKV: Reducing Write Stalls and Write Amplification in LSM-tree Based KV Stores by exploiting NVM

Outline

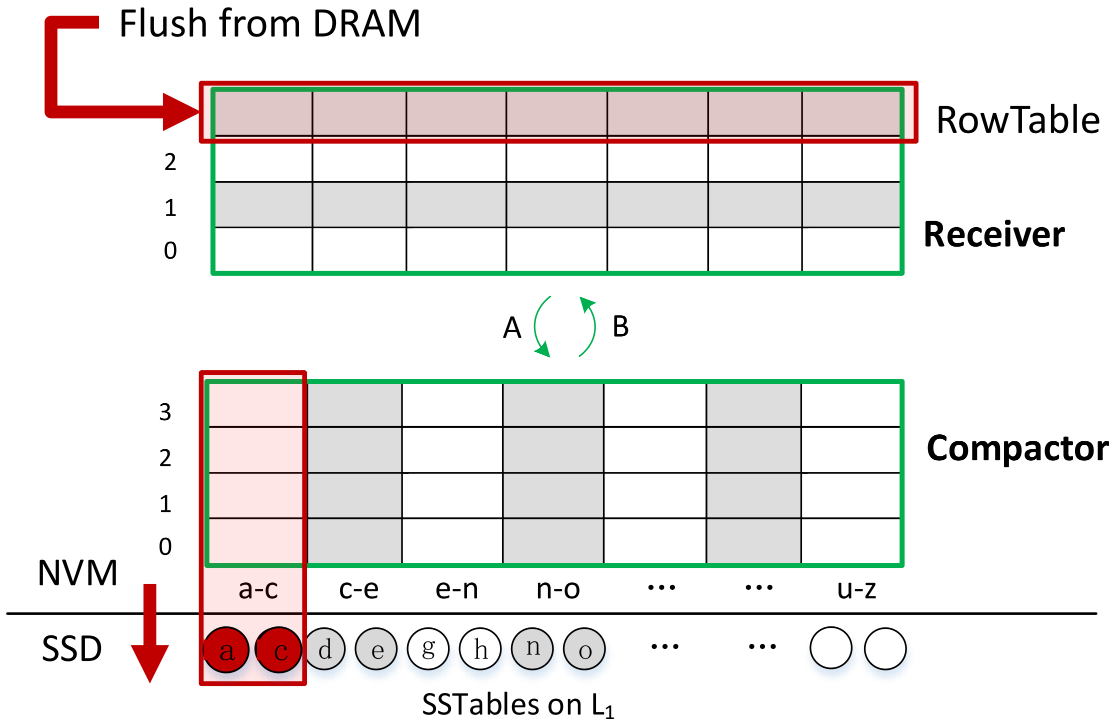
- Background and Motivations
- **MatrixKV**
- Evaluation
- Conclusion

Overall Architecture

1. Matrix container in NVM: Manage L0's data on NVM
2. Column compaction: A fine granularity column compaction to reduce write stalls
3. Reducing levels on SSD: Reduce LSM-tree's level numbers to decrease WA (on SSD)
4. Cross-Row hint search: A hint search algorithm in Matrix container to improve read performance

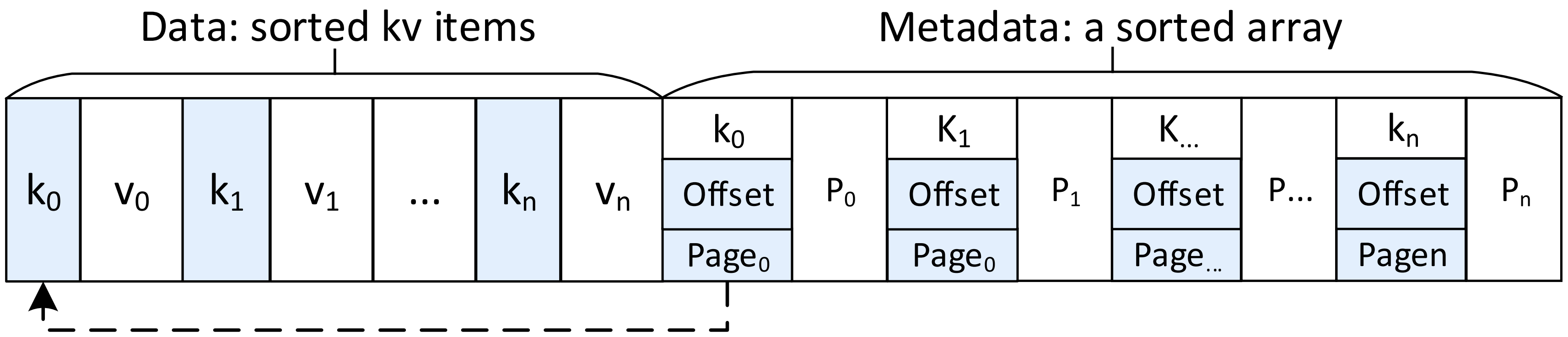


Matrix Container



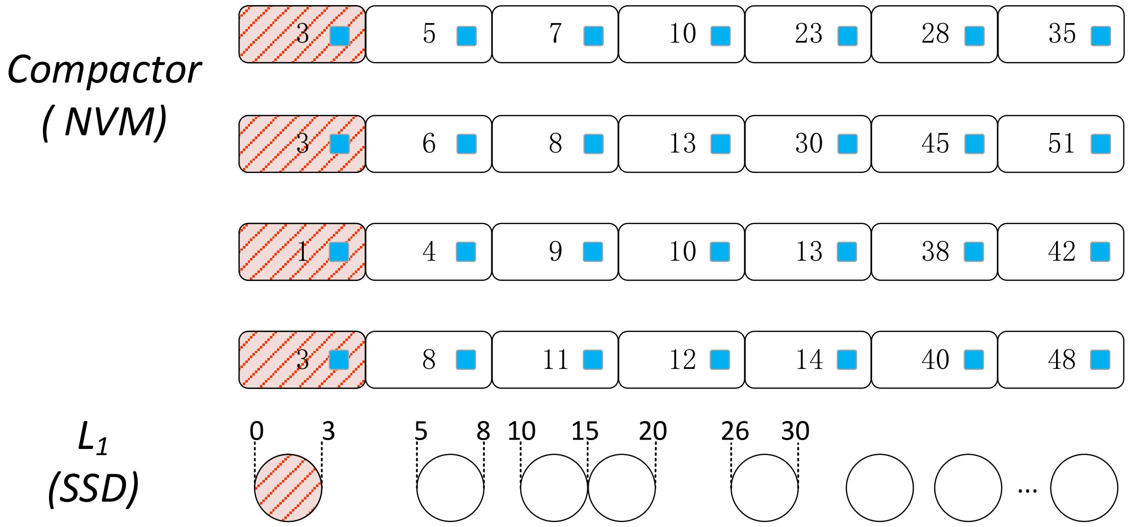
- Matrix container includes a receiver and a compactor.
- Receiver stores flushed data row by row and organized in RowTable.
- A: A receiver turns into a compactor once filled with RowTables
- Compactor compacts data from L_0 to L_1 on SSD column by column.
- B: NVM pages of a column are freed and available for receiver to accept new data after the column compaction.

RowTable



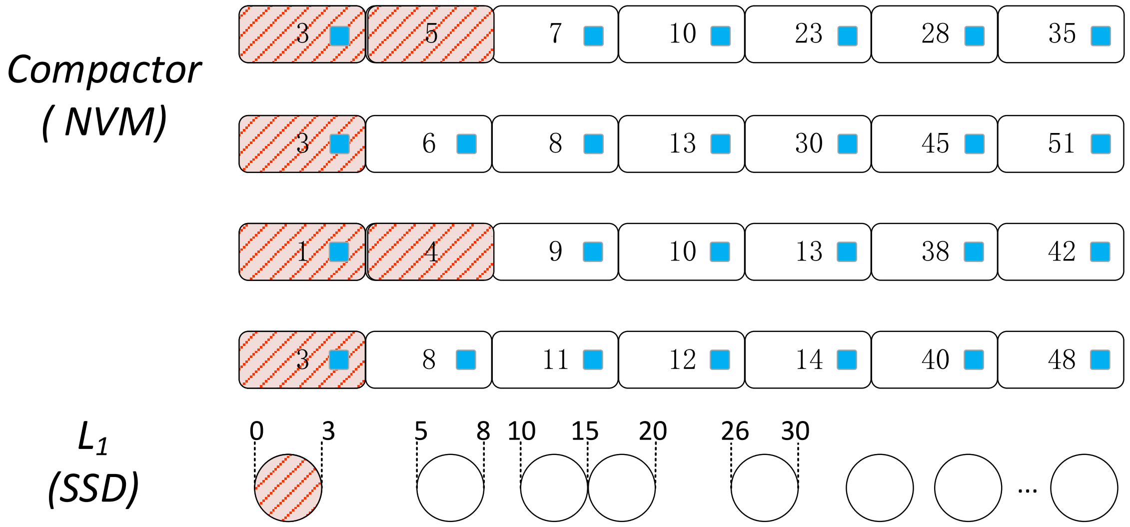
- Consisting of data and metadata.
- Data region: serialized KV items from the immutable MemTable
- Metadata region: a sorted array.
  - Key
  - page number
  - offset in the page
  - forward pointer (i.e., p_n)

Fine grained column compaction



- The non-overlapped L1 is a key space with multiple contiguous key ranges.
- Example:
  1. Range 0-3.
  2. The amount of compaction data VS. the threshold of compaction.
  3. Add the next subrange 3-5 -> Range 0-5.
  4. Add the next subrange 5-8 -> Range 0-8.
  5. Reach the threshold of compaction, Start column compaction

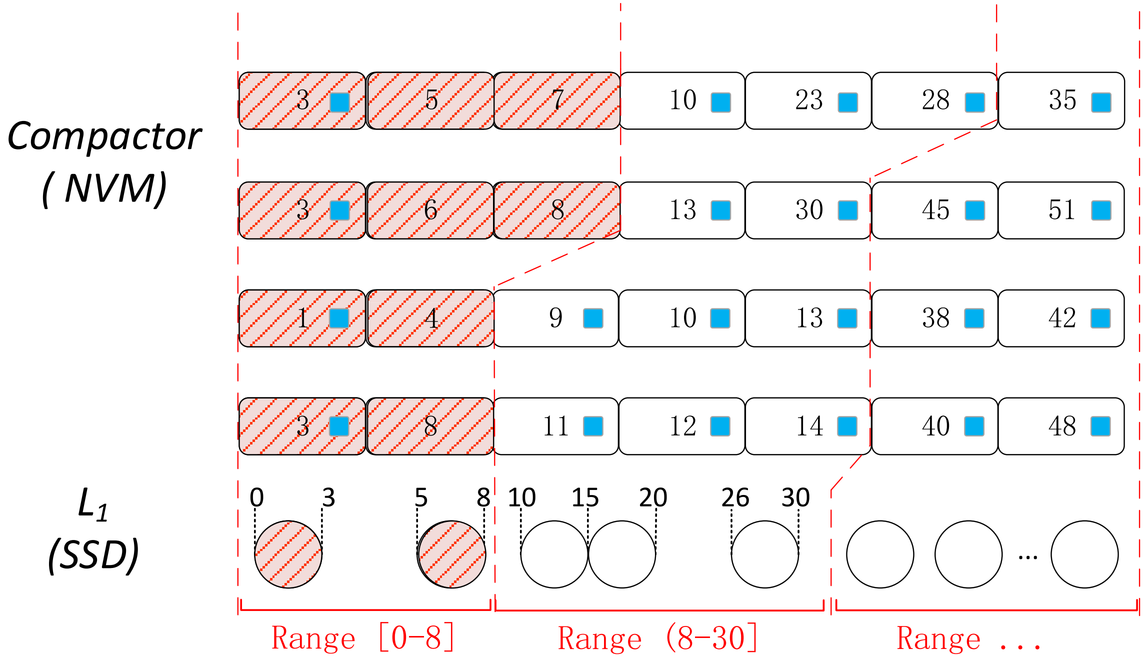
Fine grained column compaction



- The non-overlapped L1 is a key space with multiple contiguous key ranges.

- Example:
  1. Range 0-3.
  2. The amount of compaction data VS. the threshold of compaction.
  3. Add the next subrange 3-5 -> Range 0-5.
  4. Add the next subrange 5-8 -> Range 0-8.
  5. Reach the threshold of compaction, Start column compaction

Fine grained column compaction

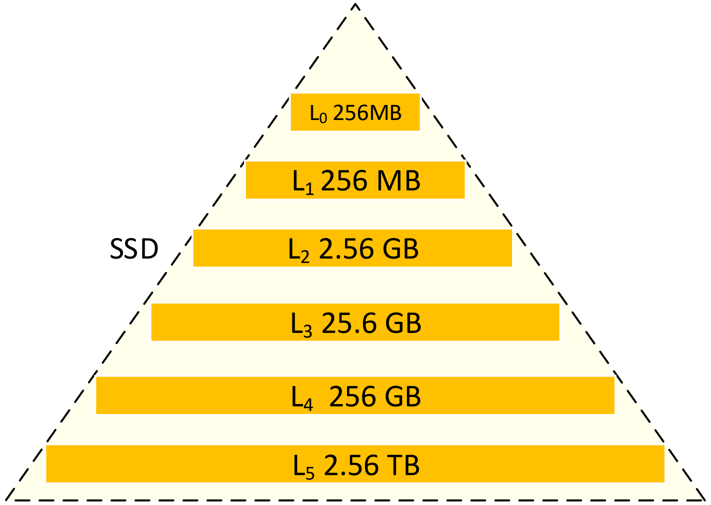


- The non-overlapped L1 is a key space with multiple contiguous key ranges.
- Example:
  1. Range 0-3.
  2. The amount of compaction data VS. the threshold of compaction.
  3. Add the next subrange 3-5 -> Range 0-5.
  4. Add the next subrange 5-8 -> Range 0-8.
  5. Reach the threshold of compaction, Start column compaction

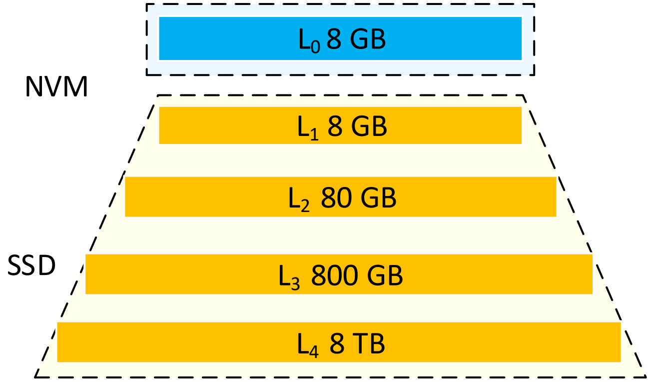
Reducing LSM-tree depth

- WA=AF * N
- Flattening LSM-trees with wider levels
  - Make the AF unchanged
  - Reduce N

- Increased unsorted L0
  - ✓ Column compaction
- Decrease search efficiency in L0
  - ✓ Cross-row hint search

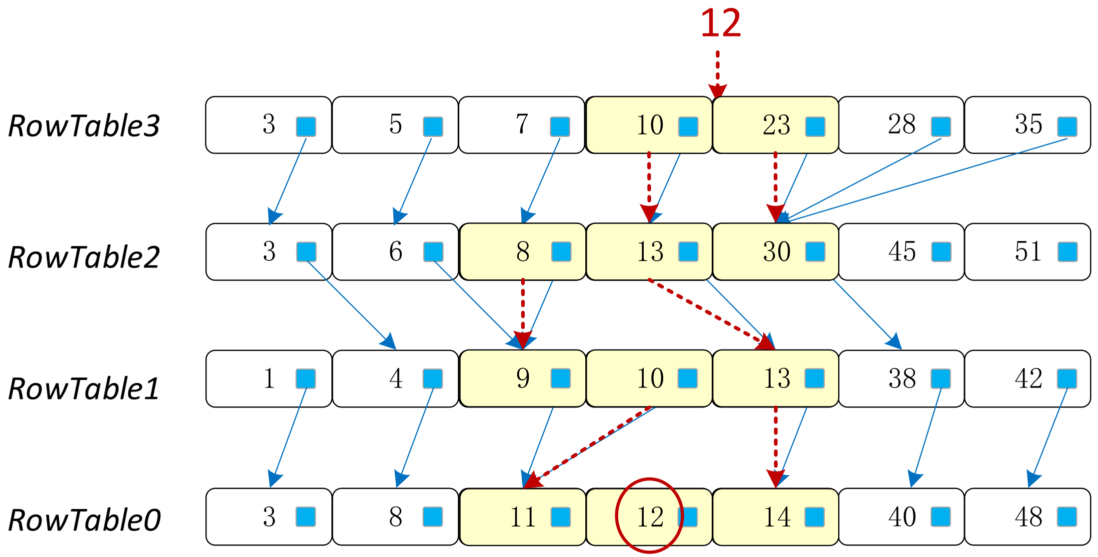


Conventional LSM-tree



Flattened LSM-tree in MatrixKV

Cross-Row hint search



- Constructing with forward pointer
  - RowTable i key x
  - RowTable i-1, key y
  - y ≥ x
- Search process with forward pointer
  - E.g., fetch key=12

Evaluation Setup

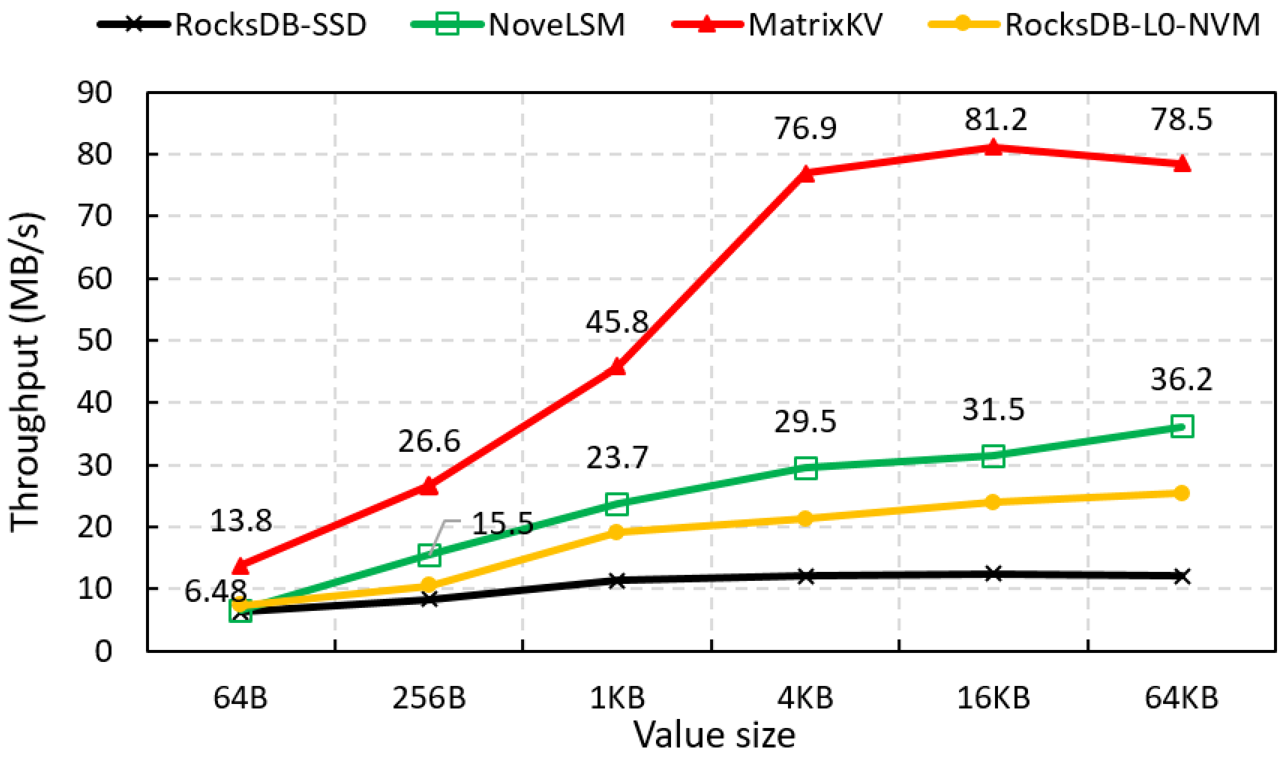
Comparisons

- RocksDB-SSD: SSD based RocksDB
- RocksDB-L0-NVM: placing L0 in NVM, system with DRAM, NVM, and SSD (8GB NVM)
- NoveLSM: a heterogeneous system of DRAM, NVM, and SSD (8GB NVM)
- MatrixKV: a heterogeneous system of DRAM, NVM, and SSD (8GB NVM)

Test environment

Linux	64-bit Linux 4.13.9
CPU	2 * Genuine Intel(R) 2.20GHz processors
Memory	32 GB
NVM	128 GB * 2 Intel Optane DC PMM FIO 4 KB (MB/s) Random: 2346(R), 1363(W) Sequential: 2567(R),1444(W)
SSD	800GB Intel SSDSC2BB800G7 FIO 4 KB (MB/s) Random: 250(R), 68(W) Sequential: 445(R),354(W)

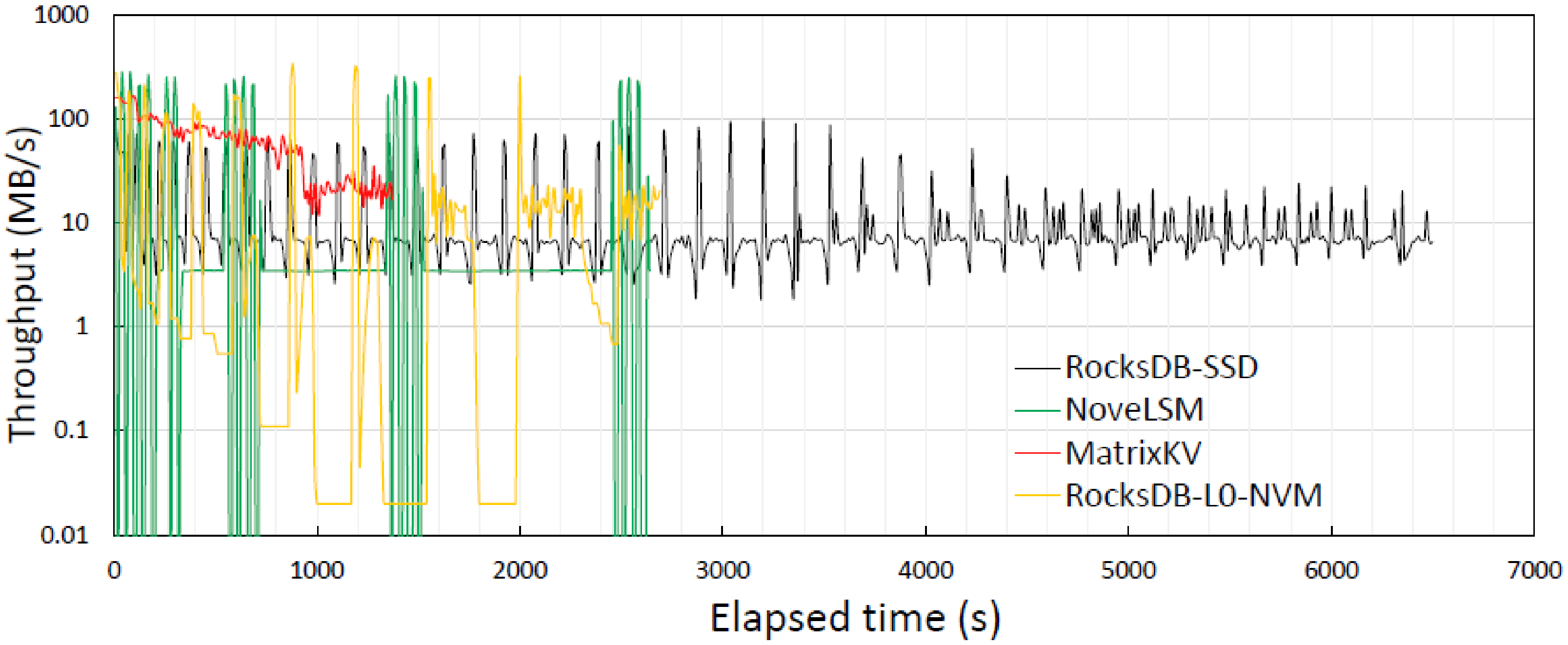
Random Write Throughput



- MatrixKV obtains the best performance in different value sizes

- E.g. 4 KB value size MatrixKV outperforms RocksDB-L0-NVM and NoveLSM by 3.6x and 2.6x.

Write stalls



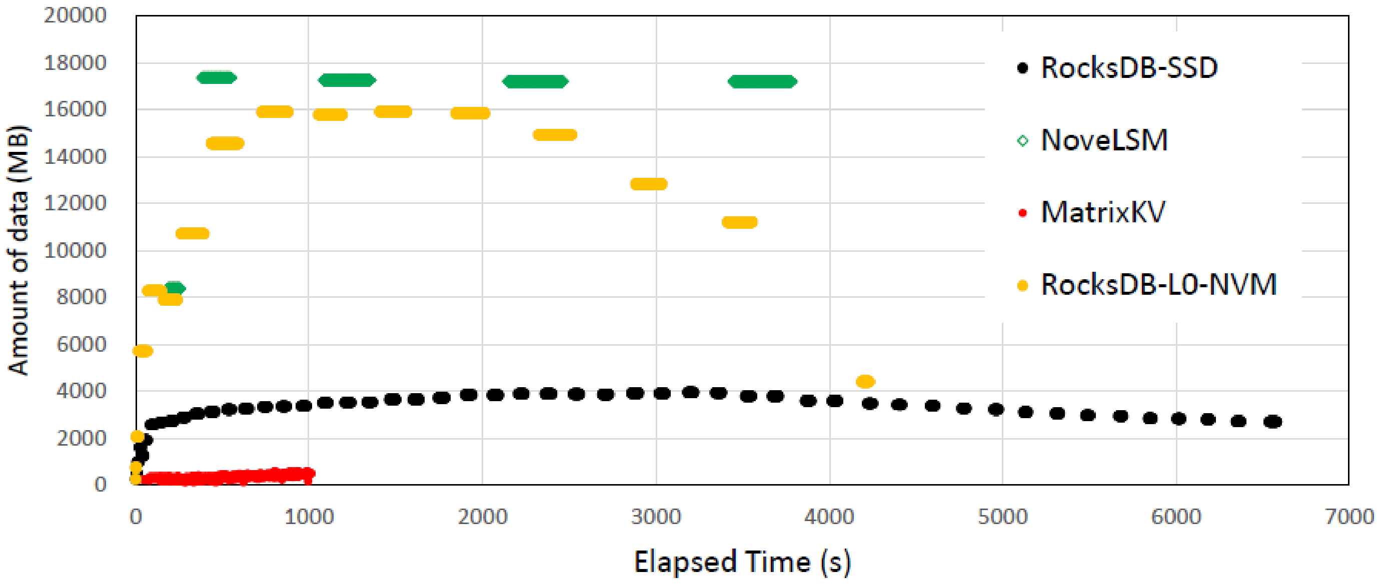
1. Better random write throughout.
2. MatrixKV has more stable throughput. Reduce write stalls!

Tail Latency

Latency (us)	avg.	90%	99%	99.9%
RocksDB-SSD	974	566	11055	17983
NoveLSM	450	317	2080	2169
RocksDB-L0-NVM	477	528	786	1112
MatrixKV	**263**	**247**	**405**	**663**

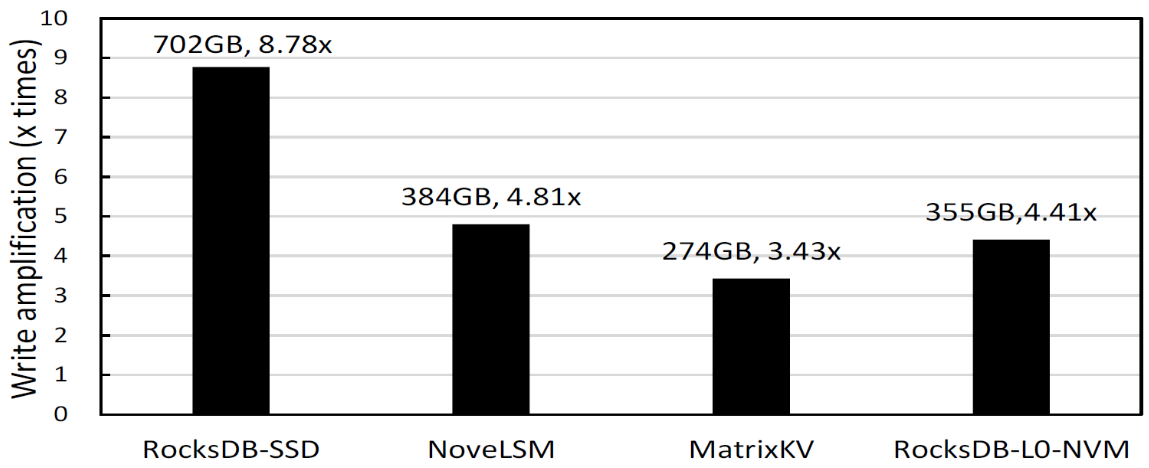
- MatrixKV obtains the shortest latency in all cases.
- E.g. 99% latency of MatrixKV is 27x, 5x, and 1.9x lower than RocksDB-SSD, NoveLSM, and RocksDB-L0-NVM respectively.

Fine granularity column compaction



- Why MatrixKV reduces write stalls？
  - 467 times column compaction
  - 0.33 GB each

Write amplification



- The WA of randomly writing 80 GB dataset.
- WA = Amount of data written to SSDs / Amount of data written by users
- MatrixKV' WA is 3.43x.
- MatrixKV reduces the number of compactions with flattened LSM-trees.

Summary

- Conventional SSD-based KV stores
  - unpredictable performance due to write stalls
  - sacrificed performance due to WA
- MatrixKV: an LSM-tree based KV store on systems with DRAM, NVM, and SSD storages
  - Matrix container in NVM
  - Column compaction
  - Hint search
  - Reducing levels on SSD
- Reduce write stalls and improves write performance.

Thanks!

Open-source code: https://github.com/PDS-Lab/MatrixKV
Email: tingyao@hust.edu.cn