GearDB: A GC-free Key-Value Store on HM-SMR Drives with Gear Compaction

Ting Yao[1,2], Jiguang Wan[1], Ping Huang[2], Yiwen Zhang[1], Zhiwen Liu[1]
Changsheng Xie[1], and Xubin He[2]

[1]Huazhong University of Science and Technology, China
[2]Temple University, USA

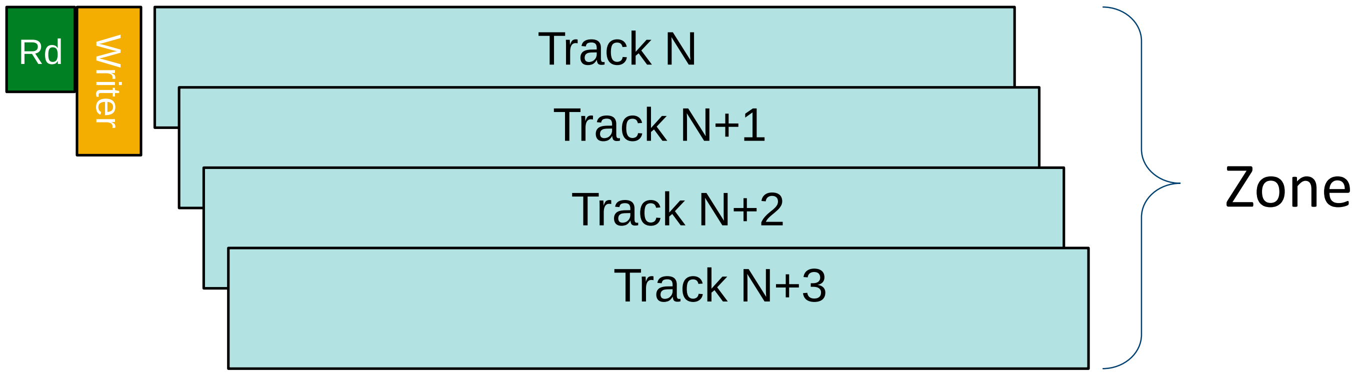
Outline

- Background and Motivation
  - Why do we run KV stores on SMR drives?
  - Challenges
- GearDB
- Evaluation
- Conclusion

SMR Drives

- Shingled Magnetic Recording (SMR)
  - ➢Increasing disk areal density

- Properties:
  - ➢ Overlapping tracks
  - ➢ Zones
  - ➢ Free read
  - ➢ Random write complexity
  - ➢ Sequential write is preferred : Log-structured



- Types: Drive-managed (DM-SMR), Host-aware (HA-SMR), and Host-managed (HM-SMR)

Host-managed SMR Drives (HM-SMR)

- Advantages:
  - Large capacity
  - Low total cost of ownership (TCO)
  - Predictable performance

- Seagate: 13TB Seagate ST13125NM007 (Test Drive)

  Exos X14 14TB 512E SATA HM-SMR
- West Digital:15 TB Ultrastar DC HC620 SMR Hard Drive

HM-SMR Drives

- Best For Applications
  - **Write data sequentially**
  - **Read data randomly**
  - Require predictable performance
  - Control of how data is handled

- Application domains:
  - Social media, cloud storage, life sciences…

LSM-tree based Key-value stores

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

- NoSQL: concerns predictable performance
- Trend: increasing demand on KV store's capacity

KV stores on HM-SMR

LSM-tree based KV stores

- Batched sequential write
- Good for hard disk drives
- Demand large capacity
- Concern predictable performance

HM-SMR drives

- Require sequential writes
- Provide large capacity
- Predictable performance
- Low total cost of ownership (TCO)

KV stores on HM-SMR

LSM-tree based KV stores

- Batched sequential write
- Good for hard disk drives
- Demand large ca
- Concern pred
  performance

HM-SMR drives

- Require sequential writes
- rovide large capacity
- e performance
- of ownership

SMORE form NetApp [MSST ’17];

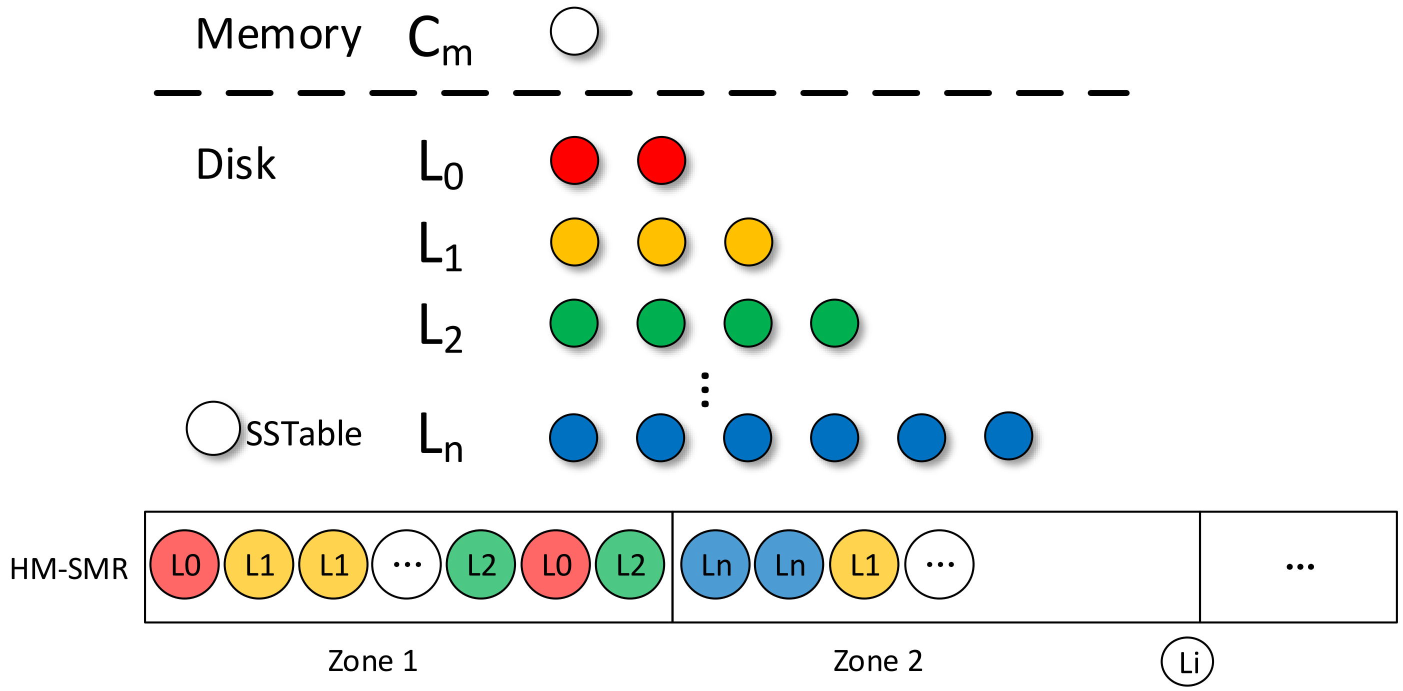
SMR based key-value store from

Huawei [SDC’15];

Outline

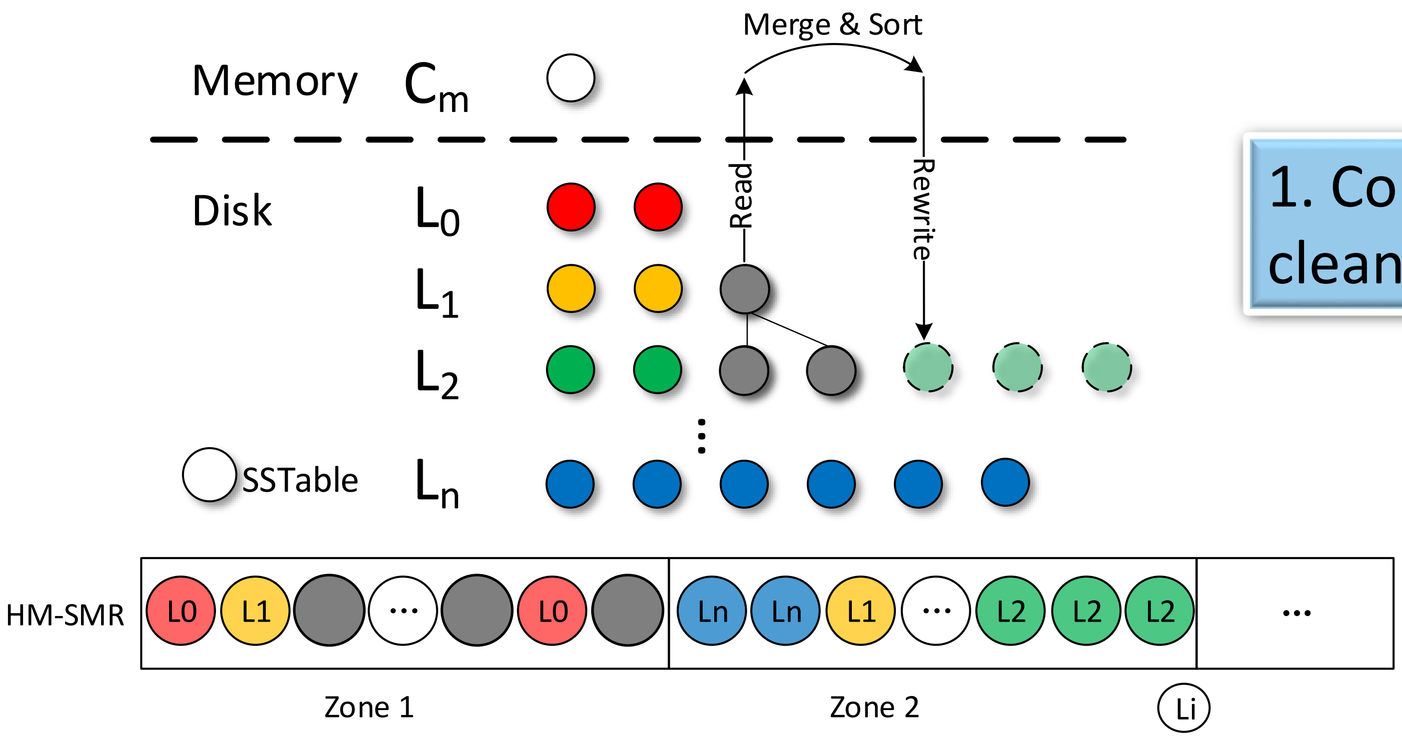
- Background and Motivation
  - Why do we run KV stores on SMR drives?
  - Challenge
- GearDB
- Evaluation
- Conclusion

Redundant Cleaning Processes



- Log structured write on HM-SMR drives:
  SSTables form different levels with different compaction frequencies are mixed in a same zone.

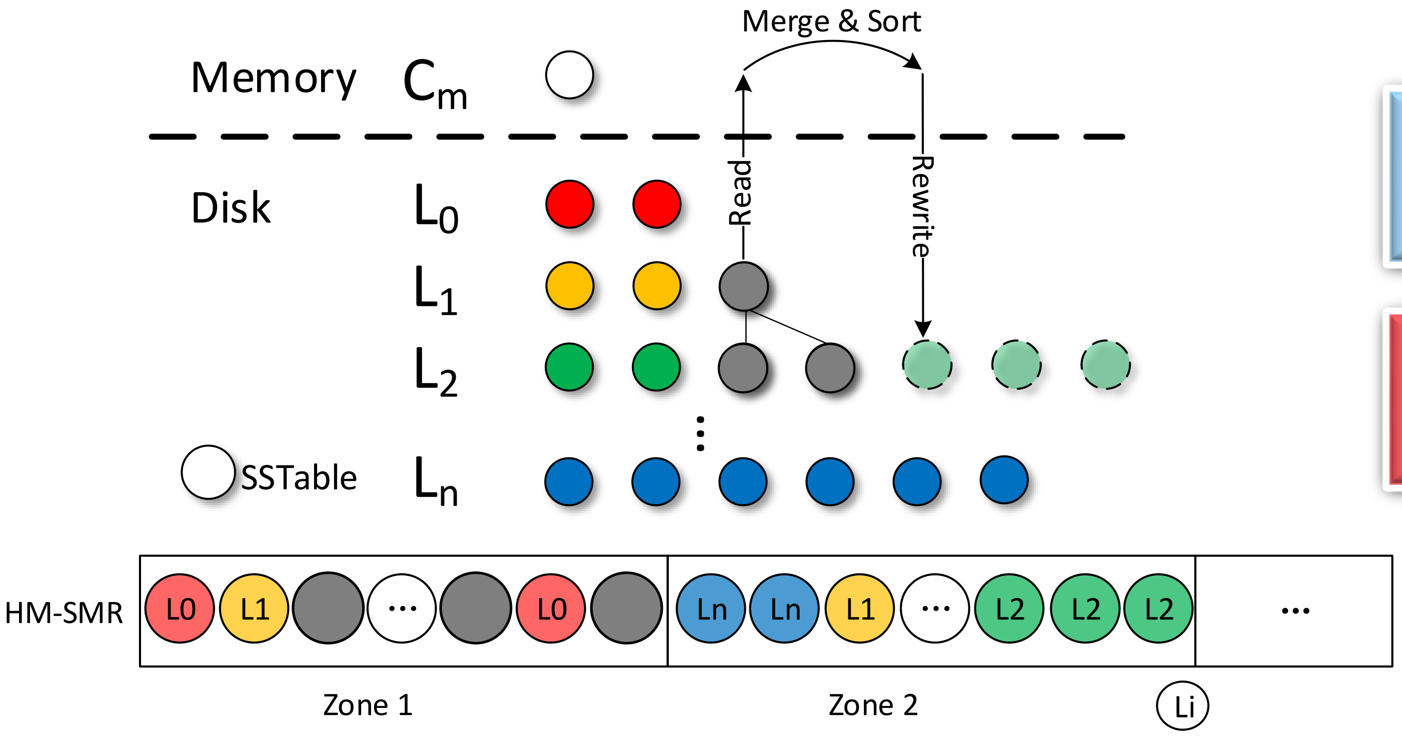
Redundant Cleaning Processes



1. Compaction in LSM-trees cleans invalid KV items

2. Garbage collections on HM-SMR drives clean invalid SSTables on Disks

Goals of GearDB



1. Compaction in LSM-trees cleans invalid KV items

Improve compaction efficiency

2. Garbage collections on HM-SMR drives clean invalid SSTables on Disks

Motivational tests

- LevelDB on an HM-SMR drive with two GCs
  - ➢Ldb-Greedy: Zones with the most invalid data
  - ➢Ldb-Cost Benefits: Zones with the oldest age and the lowest space utilization
- Trigger GC: free space under 20%
- Migrating valid data from one zone to another.

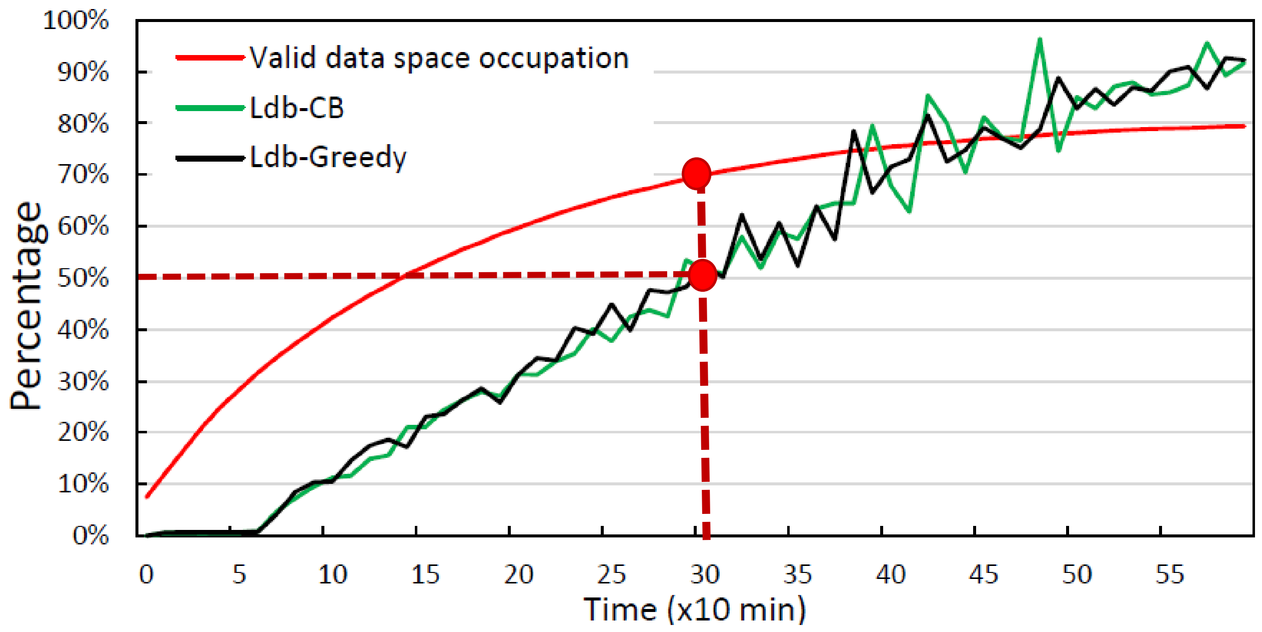
➢Randomly loading an 80 GB dataset to restricted disk space

(Making valid data takes 80% of the disk space)

Overhead of on-disk GC

- Record the valid data volume and time consumption of GCs in every ten minutes.
- 50% of the execution time is spent on GCs when valid data volume is 70% of disk space.

- Garbage collections take a substantial proportion of the system execution time.
- Degrade system performance.

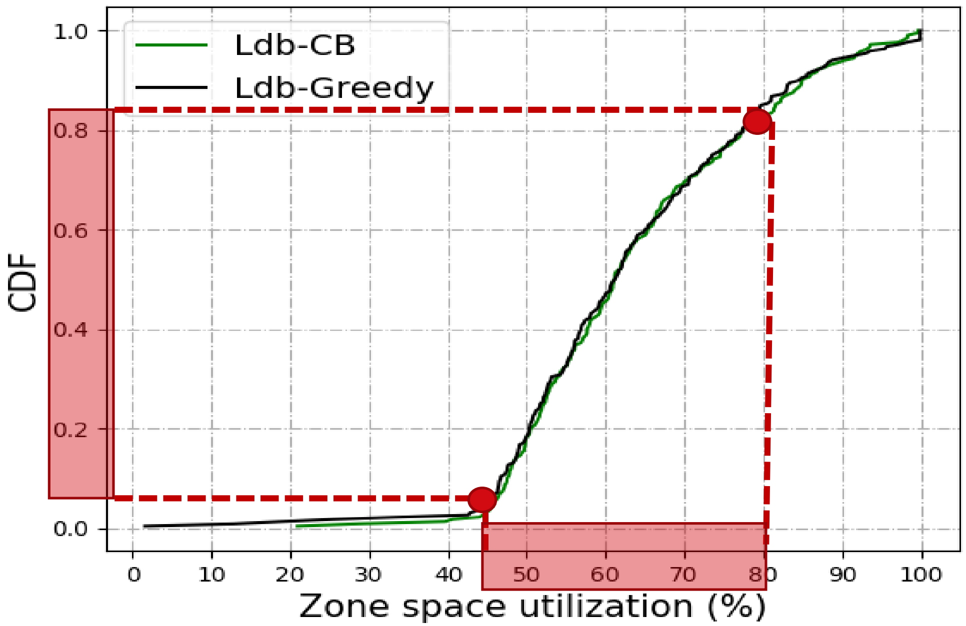


Poor Space Utilization

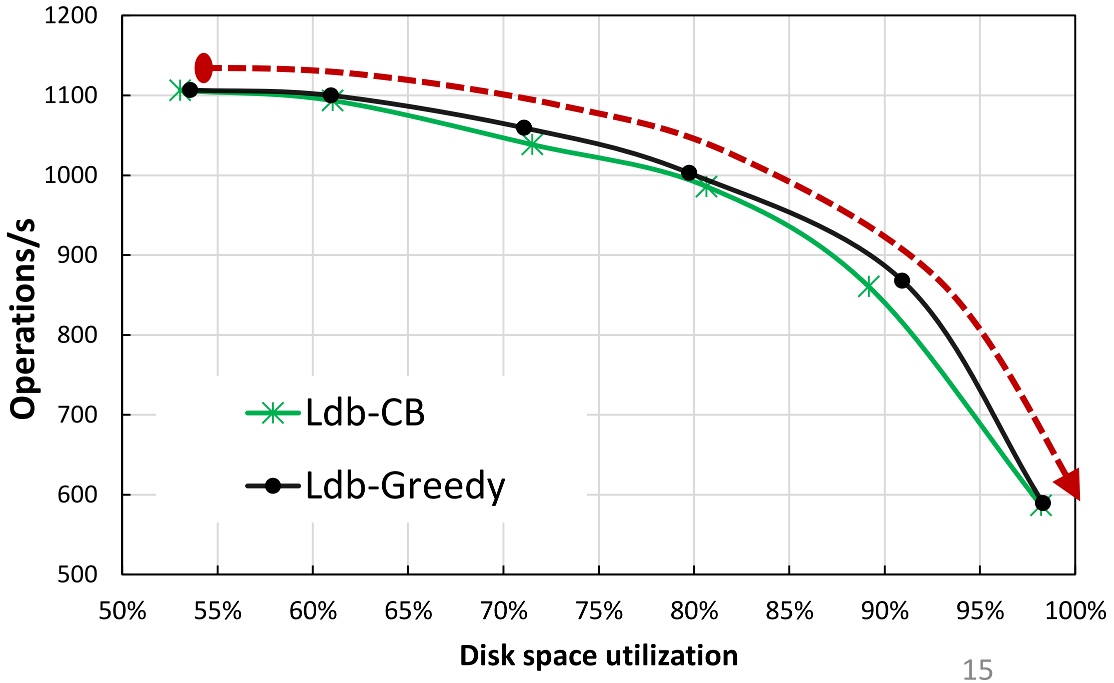
- 85% of zones have a zone space utilization ranges from 45% to 80%.

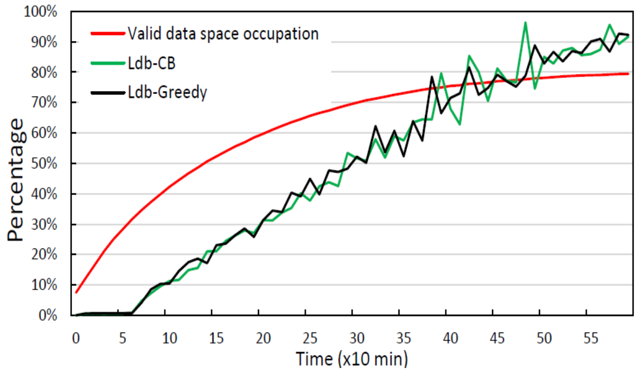
  (Zone Space Utilization $= \frac{\text{Valid } data\ volume}{zone\ size}$)
- **Overall disk space utilization: 60%**

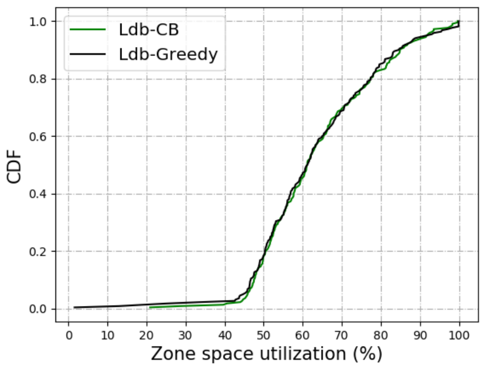
  (Space Utilization $= \frac{Valid\ data\ volume}{occupied\ disk\ space}$)

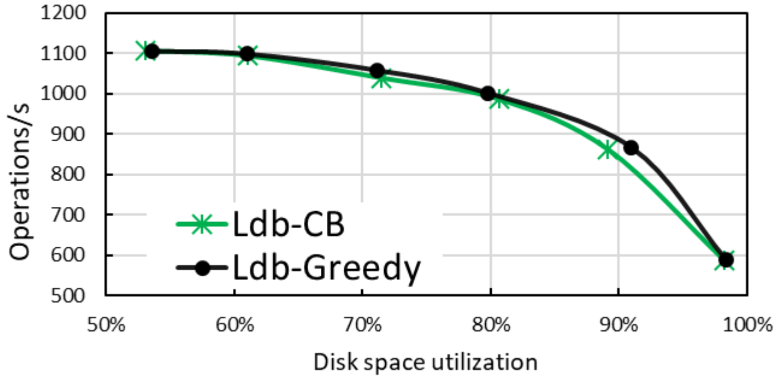


- Changing the threshold of GCs, we will get different space utilization.
- **System performance decreases with disk space utilization.**









GCs bring large overhead

Poor space utilization

System performance degrades with the increase of disk space utilization

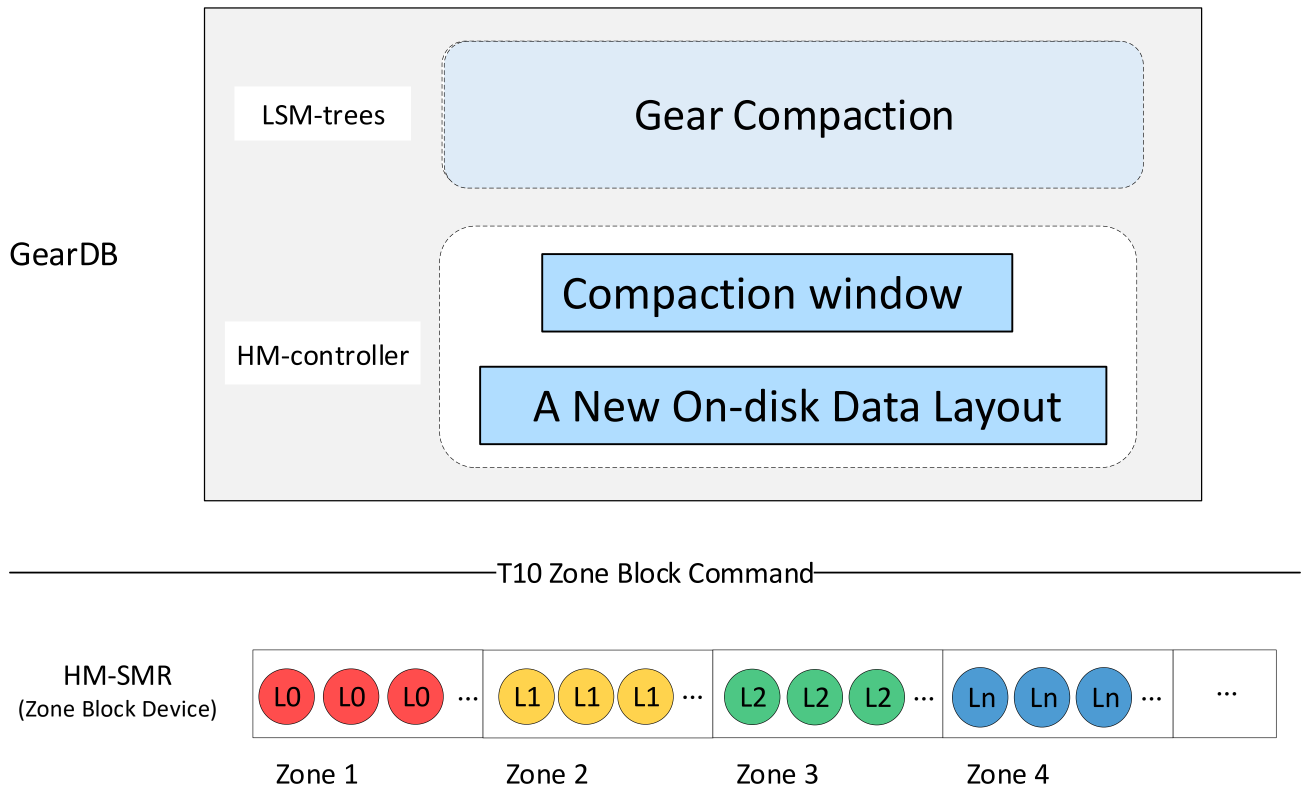
GearDB： an LSM-tree based KV store on HM-SMR drives aims to achieve both high performance and space efficiency.

Outline

- Background and Motivation
  - Why do we run KV stores on SMR drives?
  - Challenges
- GearDB
- Evaluation
- Conclusion

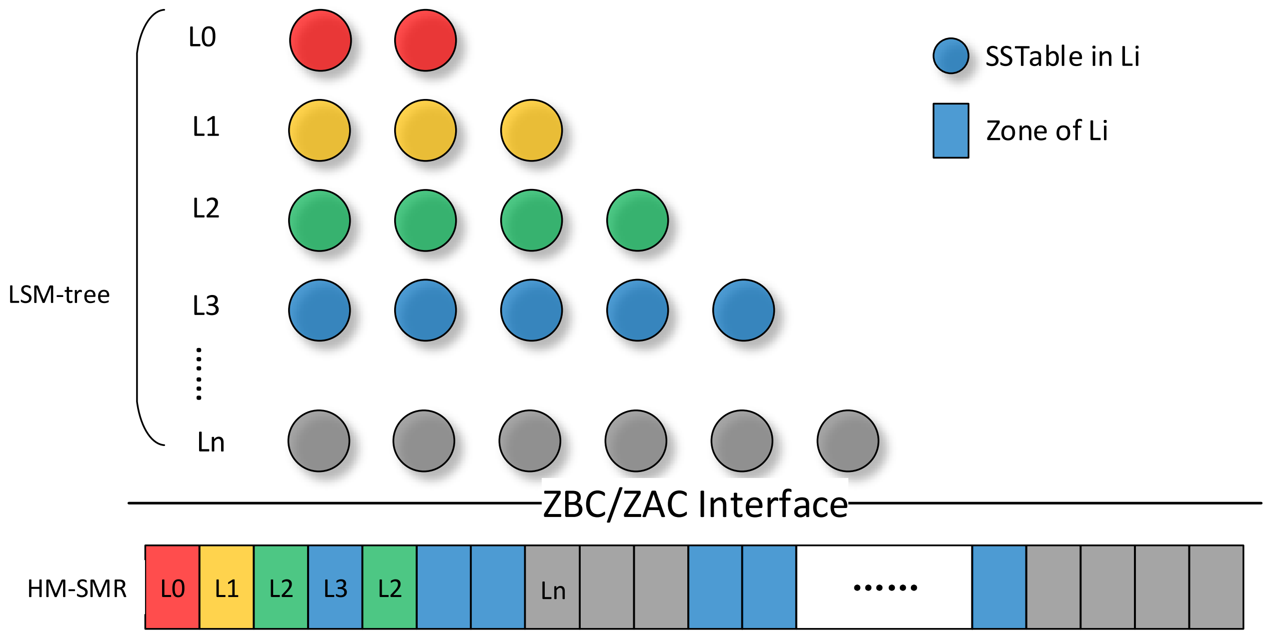
Overall Architecture

- New disk layout
  - ➢Mitigate fragments
- Compaction Window
  - ➢Restrict compactions and fragments in CWs
- Gear compaction
  - ➢Clean zones automatically



New disk layout

- Key idea: Each zone only serves SSTables from one level.
- Each level has multiple zones.
- SSTables in a zone share similar age and same compaction frequency
- Less fragmented disk space

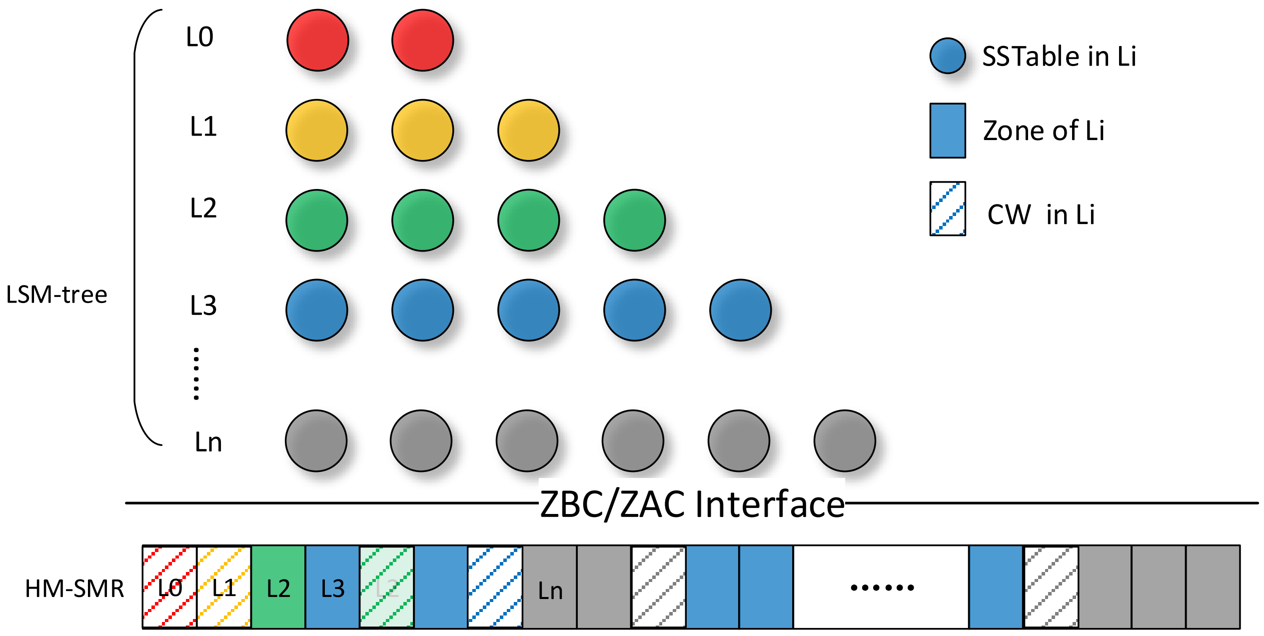


Compaction window (CW)

- For each level, a group of zones are selected rotationally to construct a compaction window.
- Each level has a CW.
- A CW contains a group of zones of one level. (e.g., k=4)

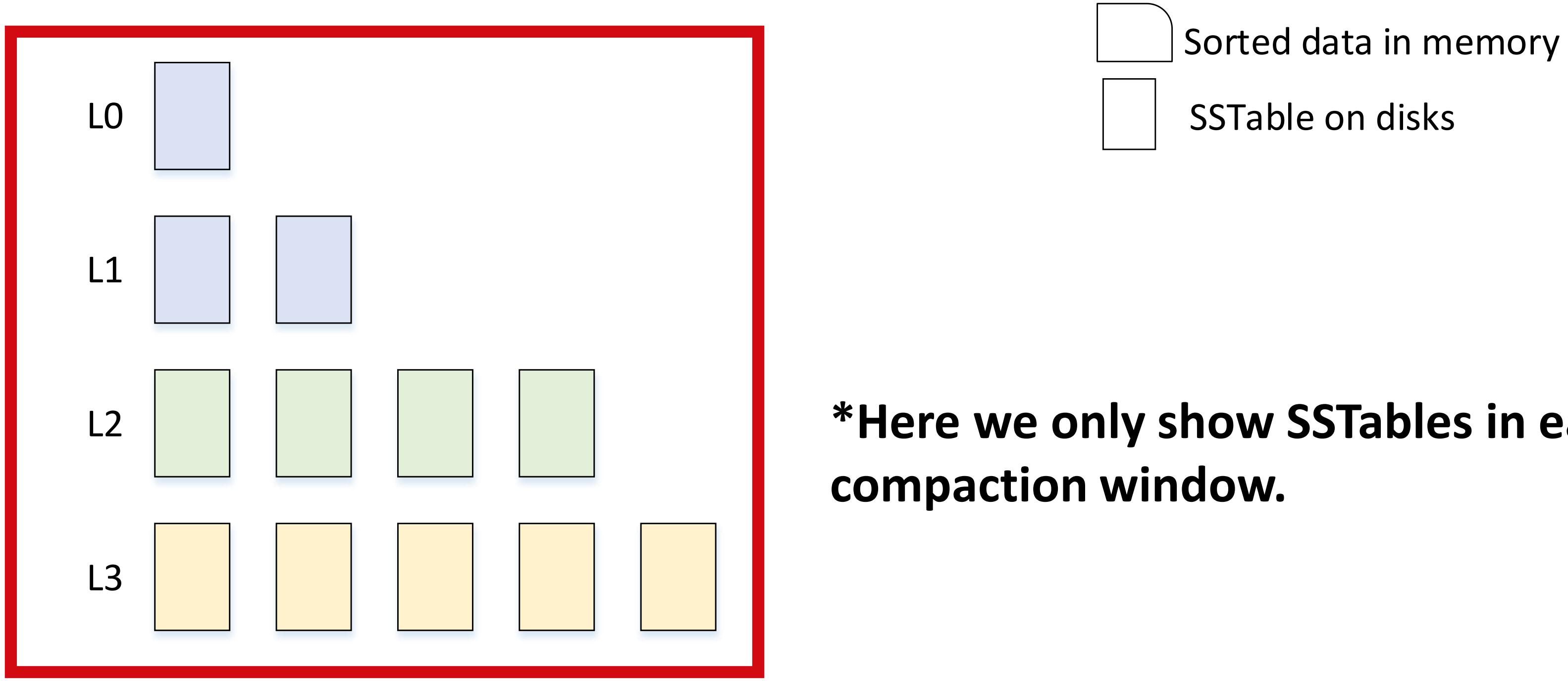
$$S_{cwi} = \frac{1}{k} \times L_{Li} \quad (1 \leq k \leq AF)$$

- CW is used to restrict compactions and fragments.



Gear compaction

- Gear compaction aims to automatically clean compaction windows by conducting compaction only within CWs.

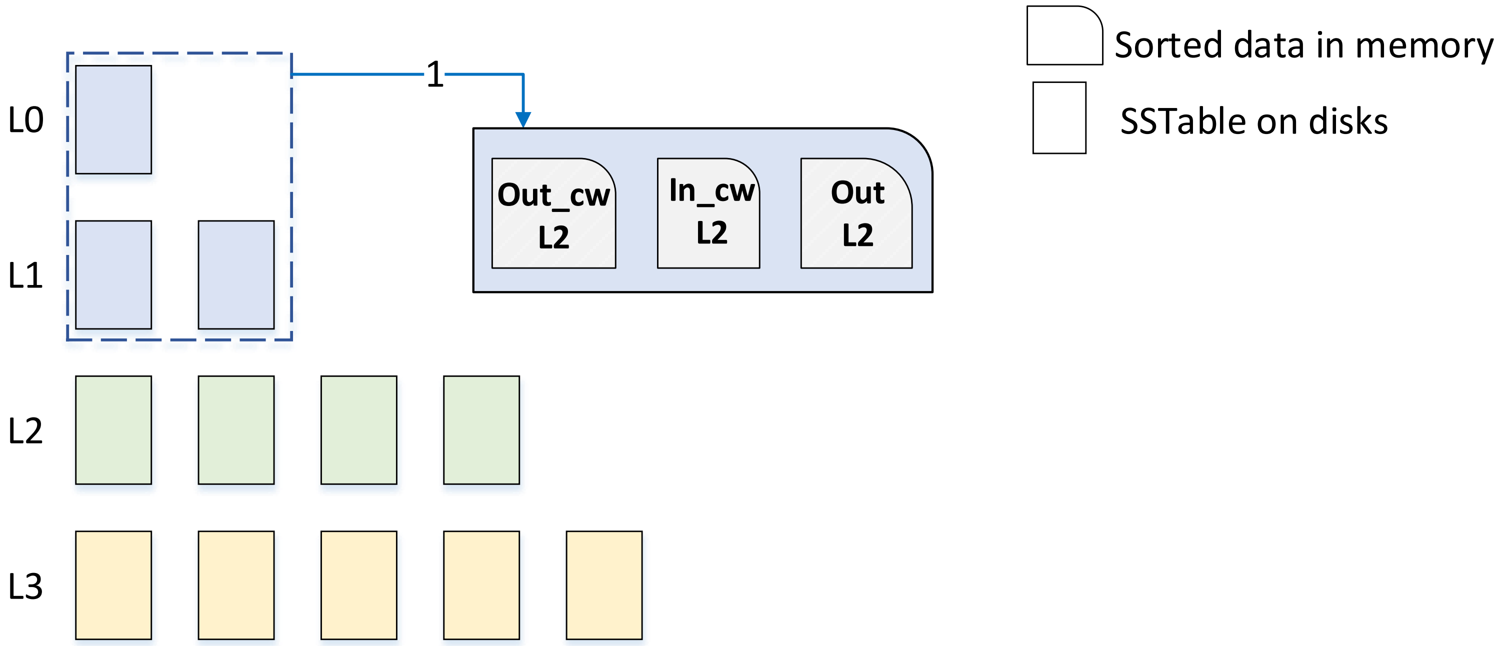


***Here we only show SSTables in each level's compaction window.**

Gear compaction

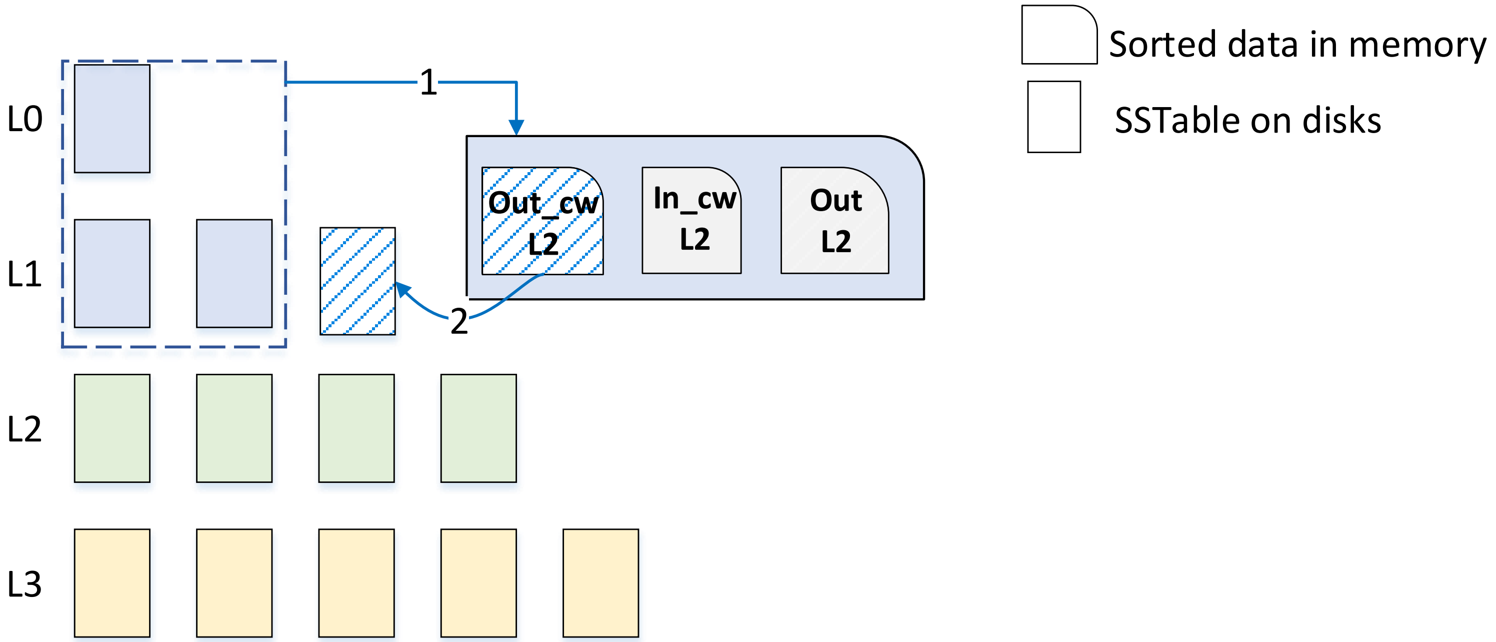
- Step 1:
  - ➢ Fetch compaction data into memory
  - ➢ Merge and sort
  - ➢ Divide the resultant data into three parts

    Out_cw L_i, In_cw L_i, and Out L_i



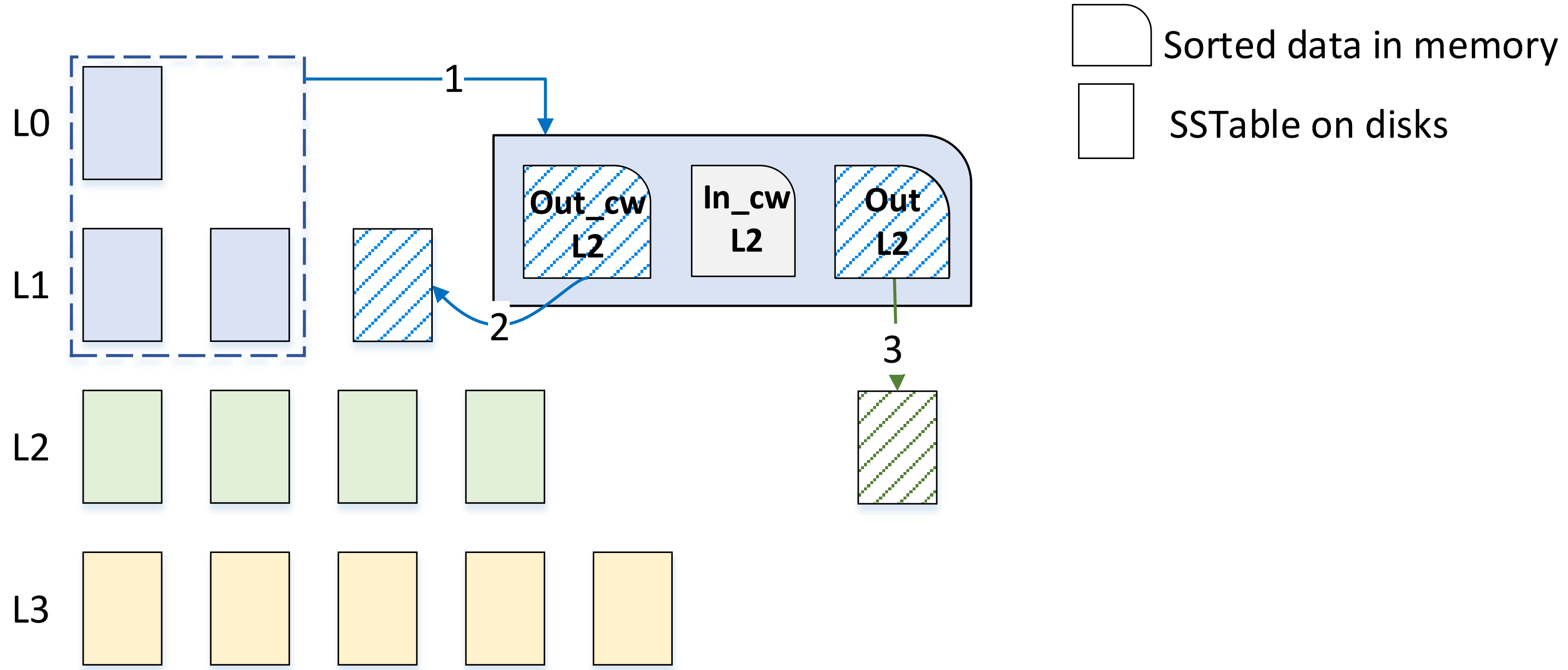
Gear compaction

- *Out_cw L_i :* data overlapped with some SSTables that are out of L_i's compaction window
- Step 2: write data *Out_cw L_2* back to L_1



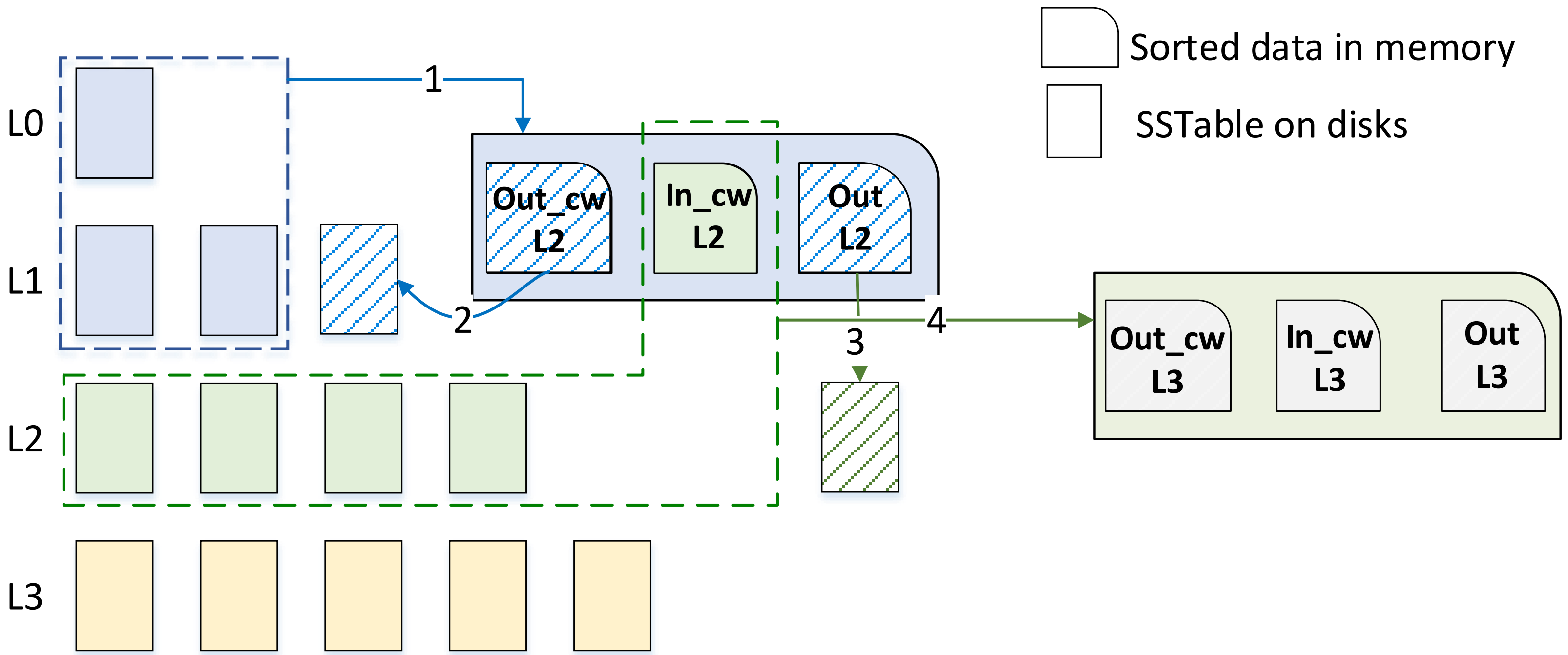
Gear compaction

- *Out L_i :* data does not overlap any SSTables in L_i
- Step 3: dump data *Out L_2* to L_2 to reduce further compactions



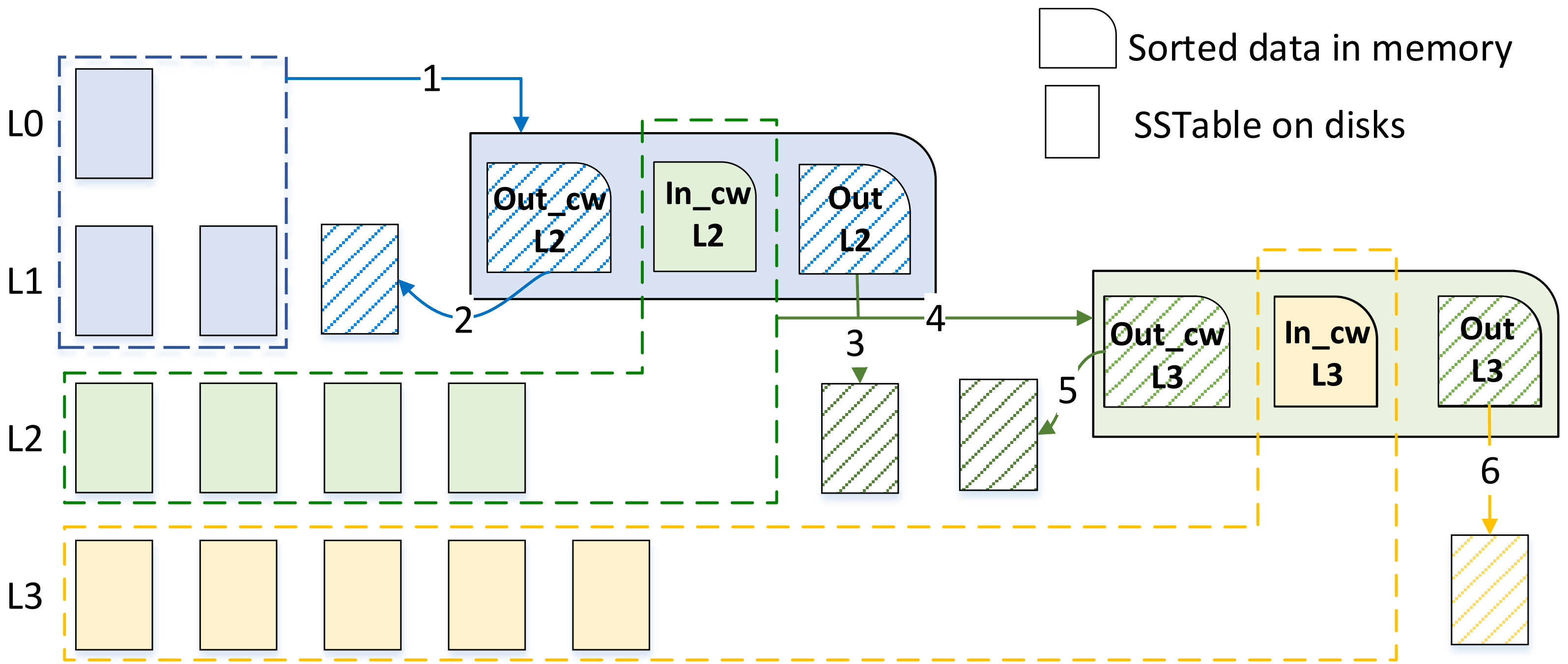
Gear compaction

- *In_cw L_i:* data overlapped with some SSTables in L_i's CW
- Step 4: Compact the data *In_cw L_2* with the overlapped SSTables in L_2's CW



Gear compaction

- Proceed recursively in compaction windows, level by level.
- Stop when compactions reach the highest level or resultant data does not overlap the CW in the next level.



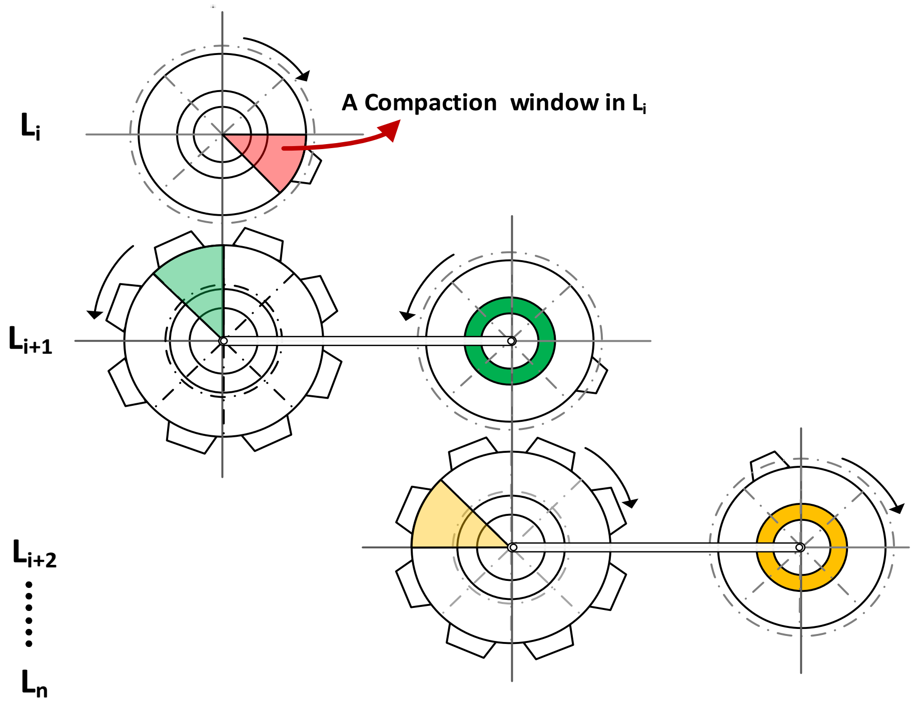
Automatically reclaim CWs

- Gear compactions only proceed within CWs
- Invalid data is restricted within CWs
- Zones filled with invalid data can be reused as empty zones
- GearDB reclaims CWs automatically with gear compactions

Reclaim CWs in a Gear fashion

- A gear represents a level (Li)
- A sector is a compaction window
- A single move of a gear: reclaiming zones in a CW by compaction
- A full round move of a gear: reclaiming all zones in L_i by compaction

- Reclaim all CWs in L_i -> clean one CW in L_{i+1}
- A full round moving of a gear -> one move in the driven gear



Evaluation Setup

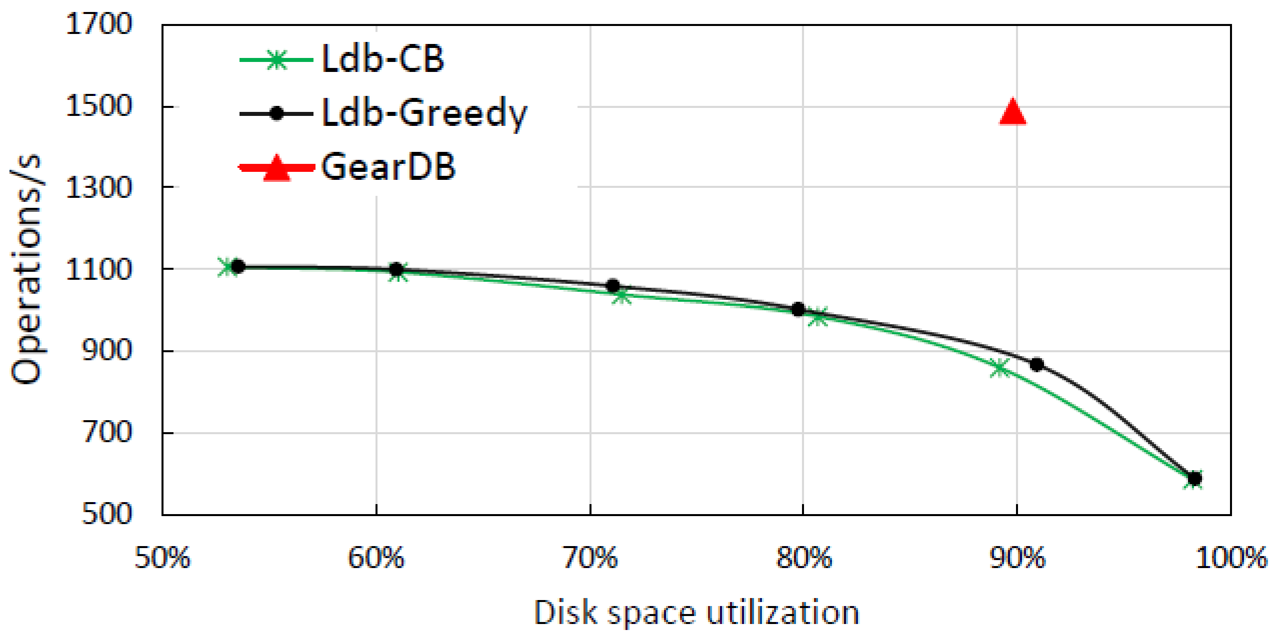
- Comparisons
  - GearDB
  - Ldb-Greedy: LevelDB with greedy GCs
  - Ldb-CB: LevelDB with cost-benefit GCs
- Test environment

Linux	64-bit Linux 4.15.0-34-generic
CPU	8 * Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
Memory	32 GB
HM-SMR	13TB Seagate ST13125NM007 Random 4 KB request (IOPS): 163(R) Sequential (MB/s): 180(R), 178(W)
Defaults	Key size=16 bytes, Value size = 4 KB, SSTable size = 4 MB

What has GearDB achieved?

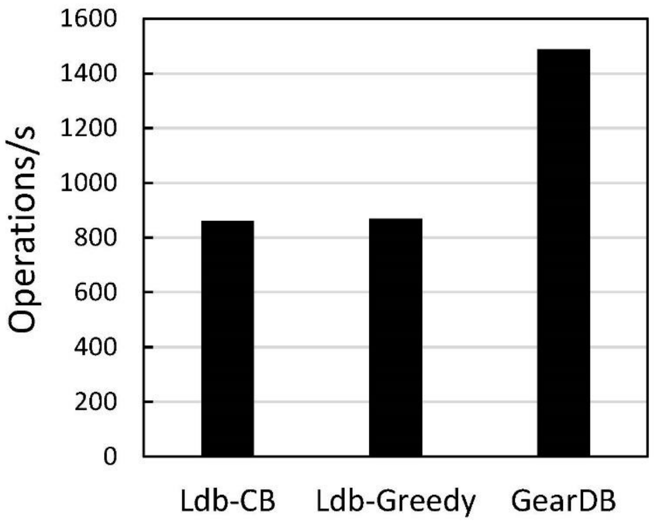
- Random load an 80 GB dataset
- Random Load performance: 1.7× higher than LevelDB
- Space Utilization: 90%

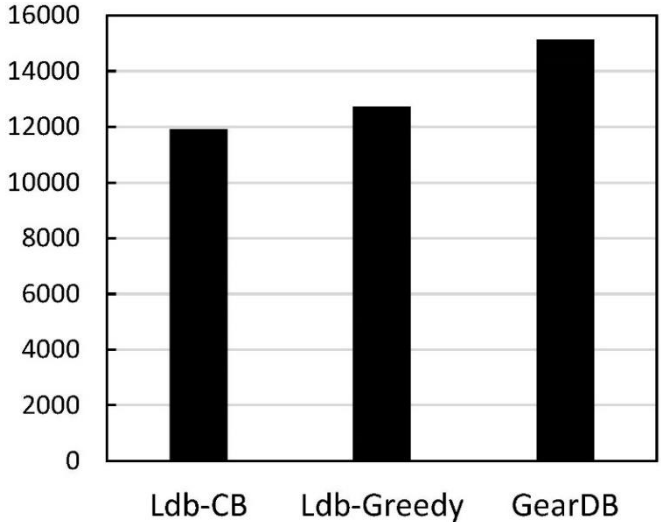
➢ High random load performance and space efficiency



Write and Read Performance

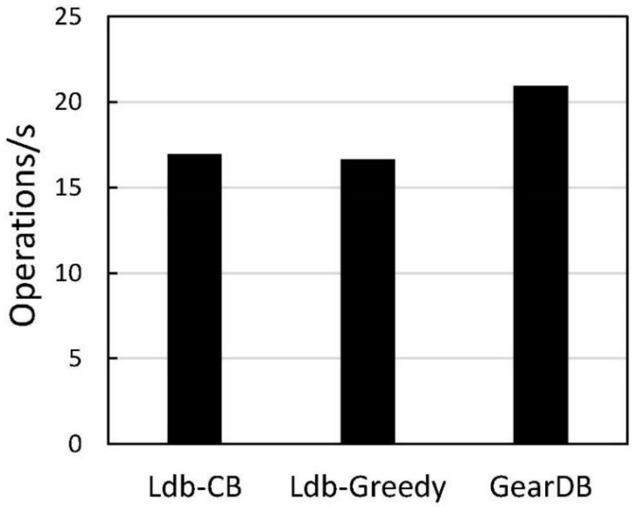
Random Write:
1.71× faster than Ldb-Greedy
1.73× faster than Ldb-CB

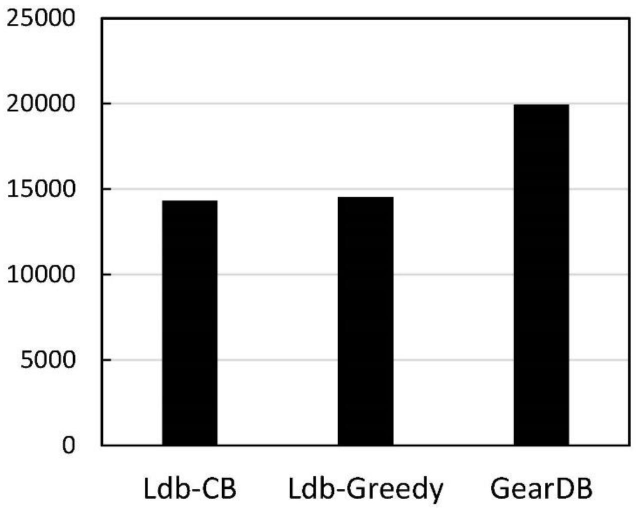




Sequential Write:
1.37× faster than Ldb-Greedy
1.39× faster than Ldb-CB

Random Read



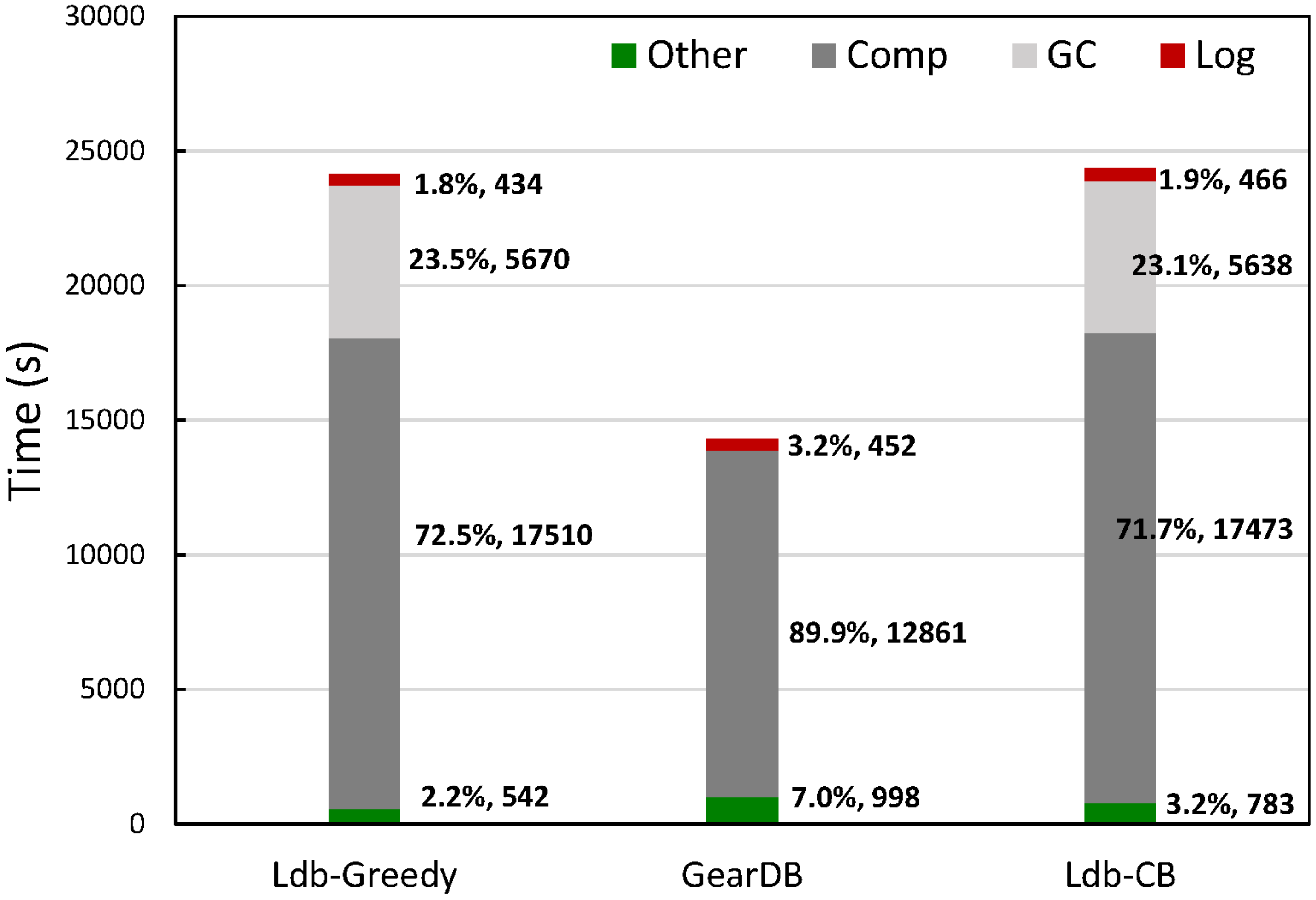


Sequential Read

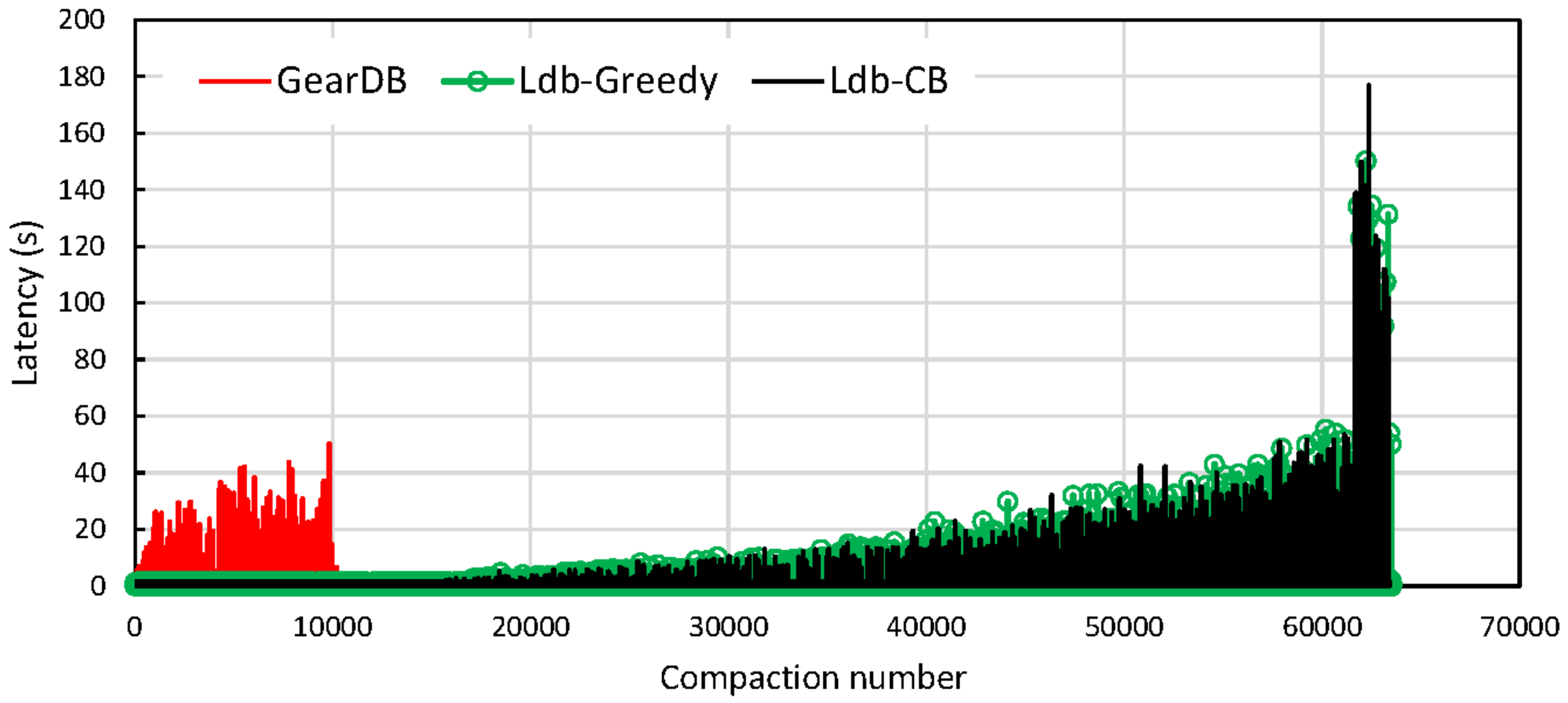
Why does GearDB perform better?

- Break down the random load time into different operations

➢ Eliminate device level GCs
➢ More efficient compaction

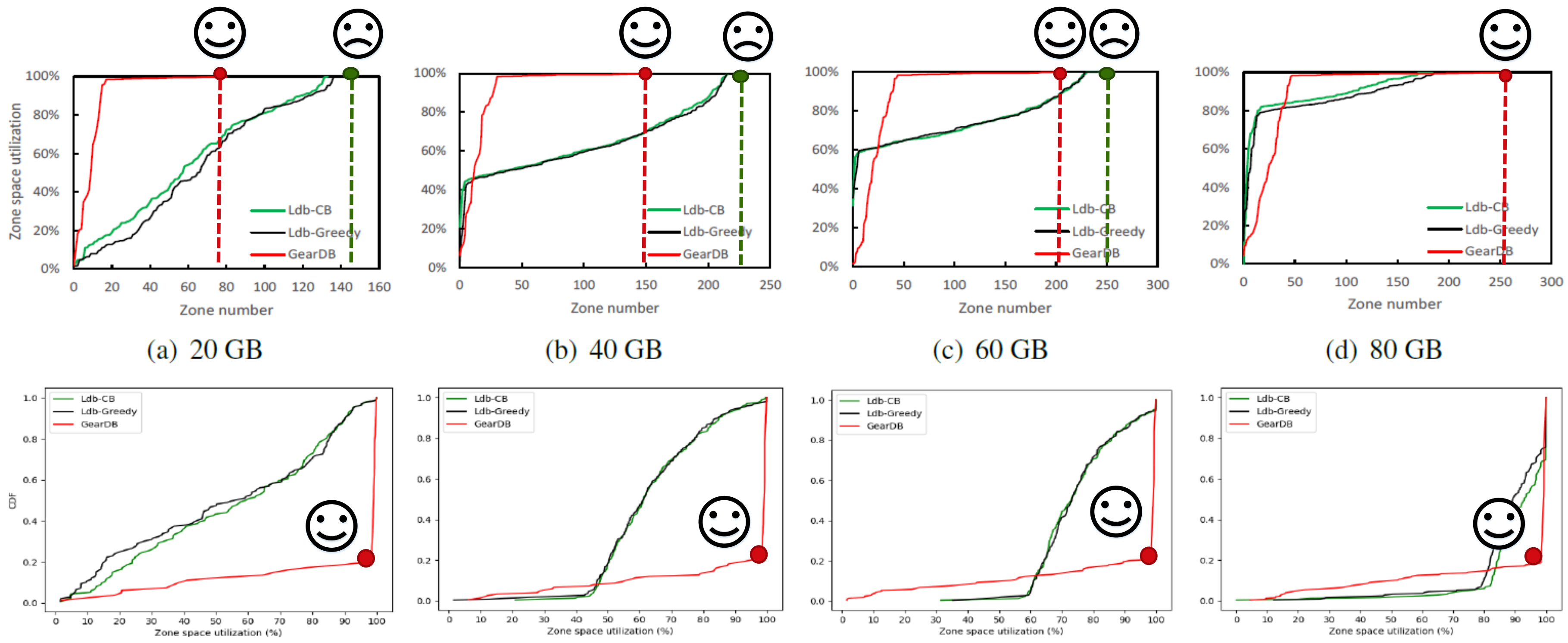


Compaction Efficiency



- The latency of each compaction during the random loading.
  - Reduce over 5000 compactions.
  - The overall compaction latency of GearDB is 55% of LevelDB.

Space Efficiency



(a) 20 GB (b) 40 GB (c) 60 GB (d) 80 GB

(e) CDF 20 GB (f) CDF 40 GB (g) CDF 60 GB (h) CDF 80 GB

- Zone space utilization after randomly loading 20, 40, 60, and 80 GB databases.

➢ GearDB occupies fewer zones

➢ GearDB shows a bimodal zone space utilization: most zones are nearly full, and a few zones are nearly empty restricts fragments in a CWs

➢ GearDB maintains a high space utilization consistently (i.e., nearly 90%)

Conclusion

- Conventional Key-value stores on HM-SMR drives
  - ➢ Redundant cleaning processes in application levels and storage levels
  - ➢ Poor performance and inefficient space utilization
- We propose GearDB to eliminate on-disk GCs and improve compaction efficiency
  - ➢ New data layout
  - ➢ Compaction windows
  - ➢ Gear compaction algorithm
- 1.7× speedup for random writes with a zone space utilization of 90%

Thanks! Q&A

Open-source code: https://github.com/PDS-Lab/GearDB

Email: tingyao@hust.edu.cn