HotStorage 2024

Advocating for Key-Value Stores with Workload Pattern Aware Dynamic Compaction

Heejin Yoon, Jin Yang, Juyoung Bang,
Sam H. Noh*, Young-ri Choi

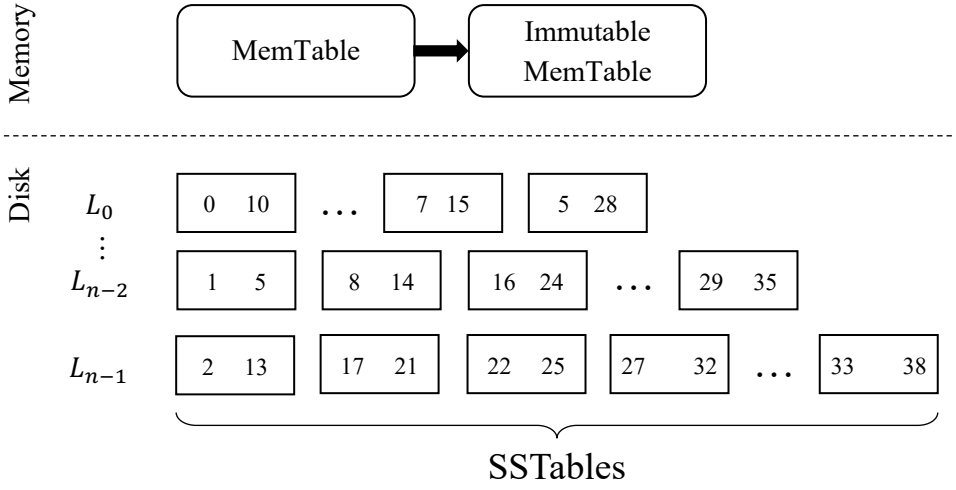
UNIST, *Virginia Tech

Contents

- Background and Motivation
- DOPA-DB
- Evaluation
- Discussion & Future works
- Conclusion

Background: LSM-based KVS

- Log-Structured Merge-tree based KV store
  - High write performance due to sequential write
  - Used for various data-intensive applications

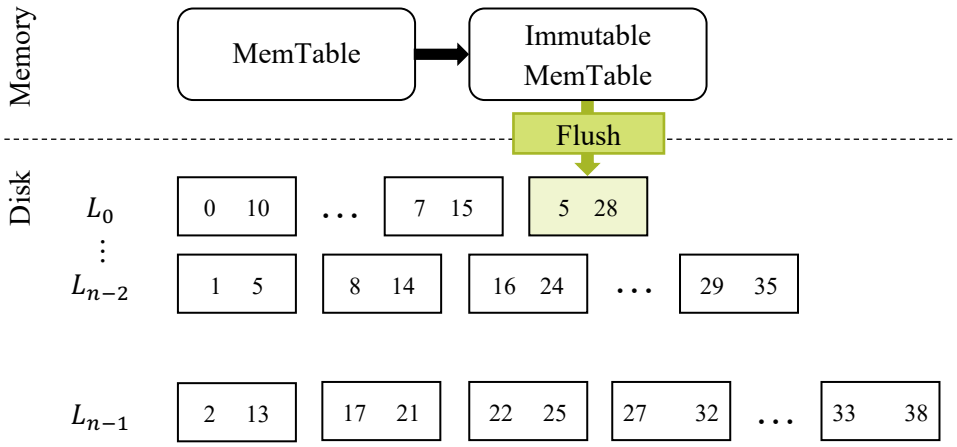


Lower (Recent)

Upper (Old)

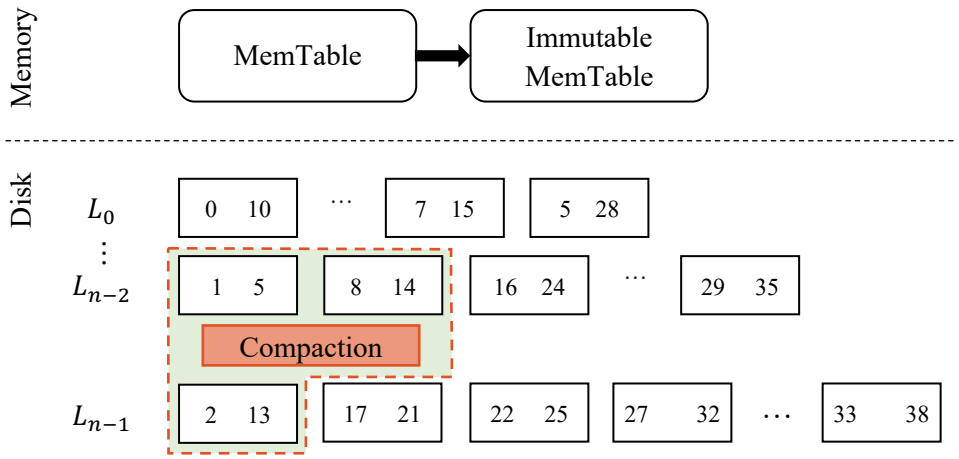
Flush of LSM-based KVS

- Data is saved to disk in the form of file through Flush



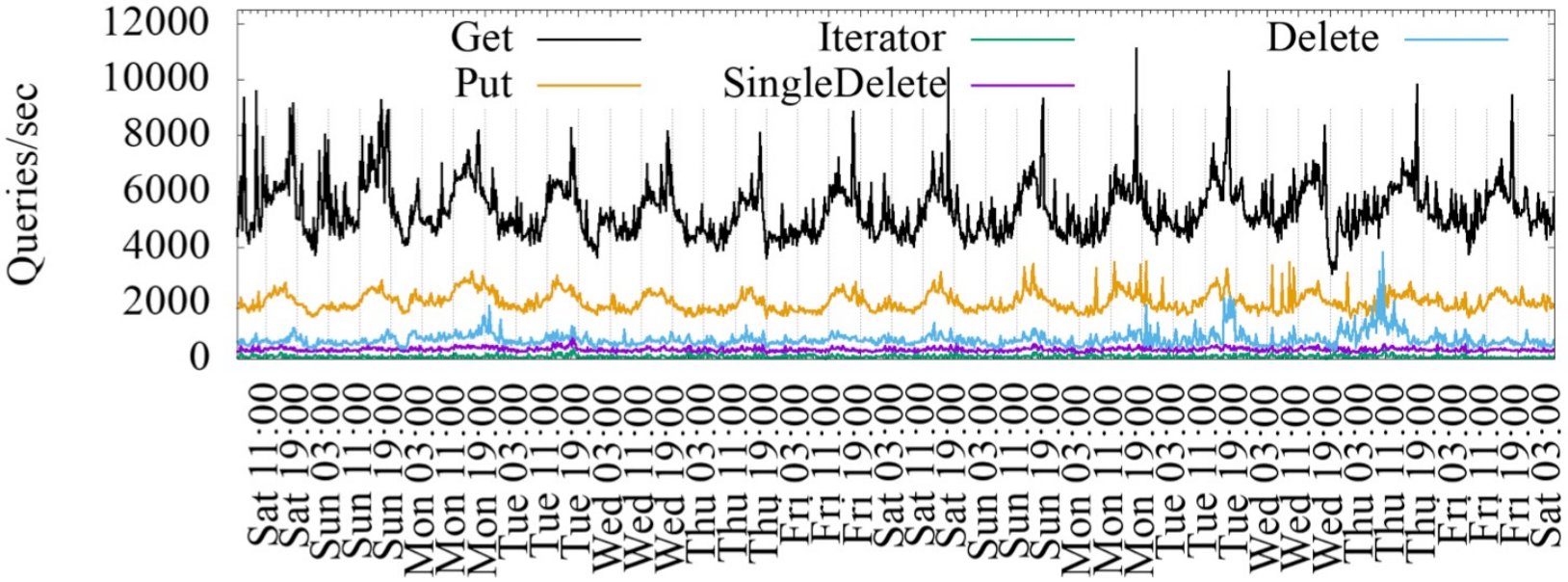
Compaction of LSM-based KVS

- Disk-level data is sorted by Compaction



Dynamic workload in real world

- In real life, workload pattern regarding ratio of write and read operations changes over time

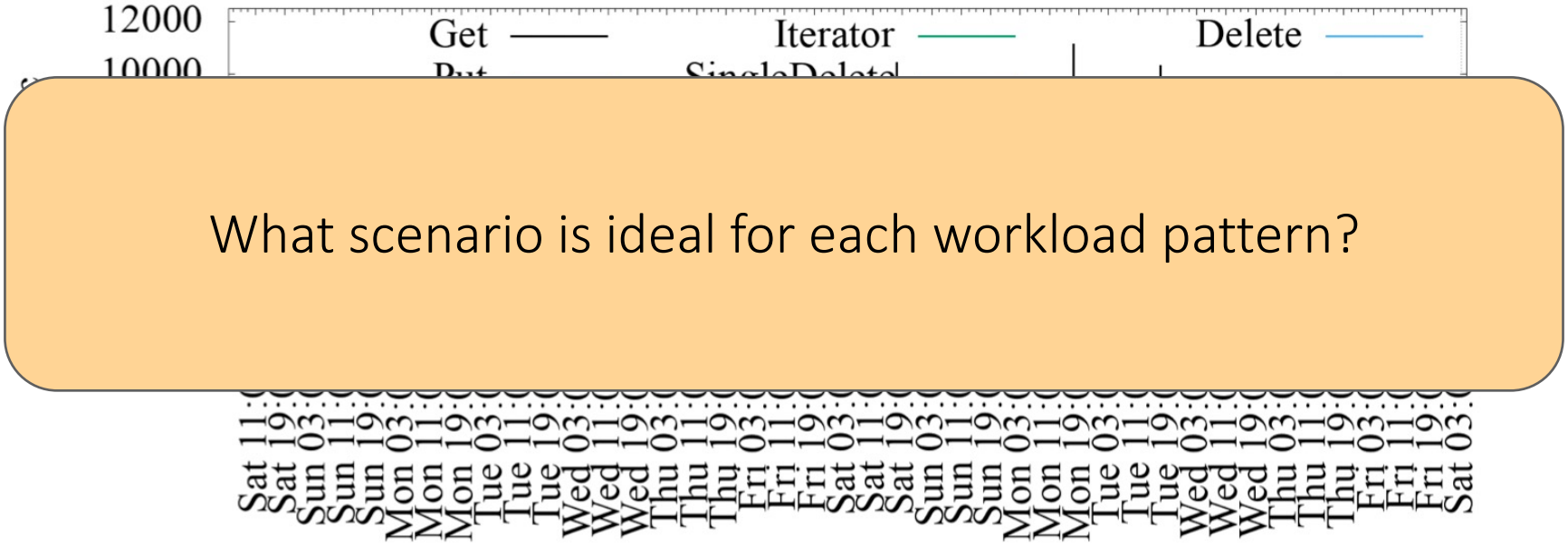


The QPS variation in a 14-day time span of UDB [1]

[1] Characterizing, modeling, and benchmarking RocksDB Key-Value workloads at facebook (FAST 2020)

Dynamic workload in real world

- In real life, workload pattern regarding ratio of write and read operations changes over time

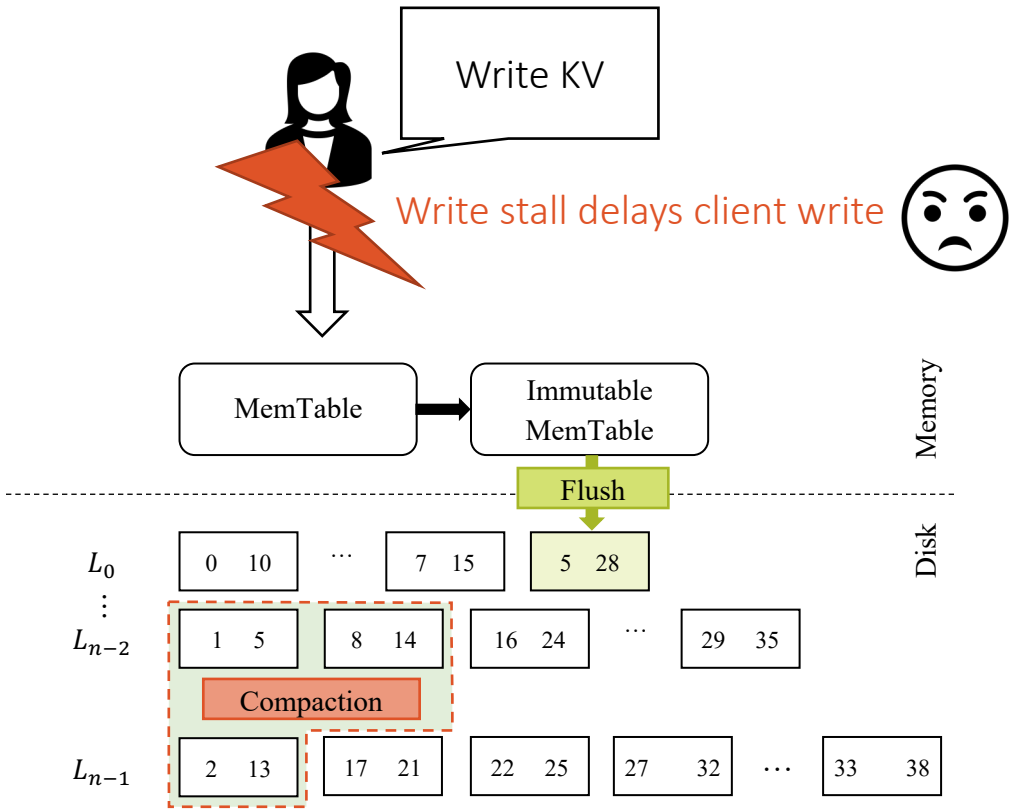


The QPS variation in a 14-day time span of UDB [1]

[1] Characterizing, modeling, and benchmarking RocksDB Key-Value workloads at facebook (FAST 2020)

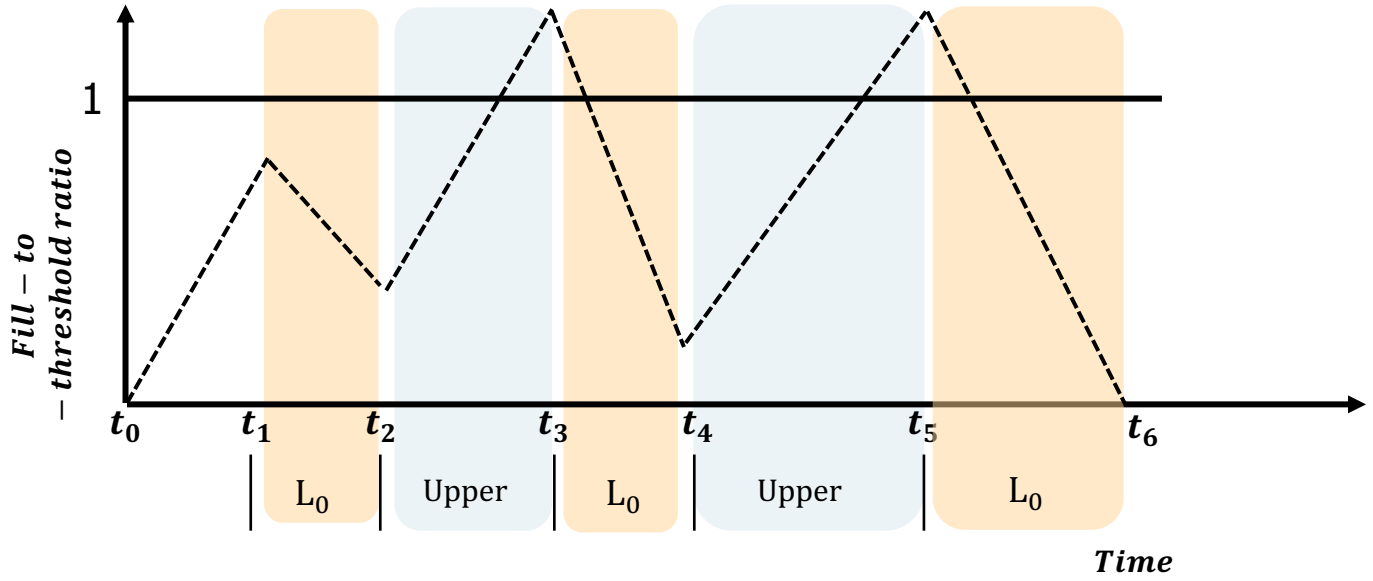
Write stall on write intensive workload

- In write intensive workloads, write stalls may occur
  - L_0 stall ➔ most crucial!
  - MemTable Stall
  - Pending stall



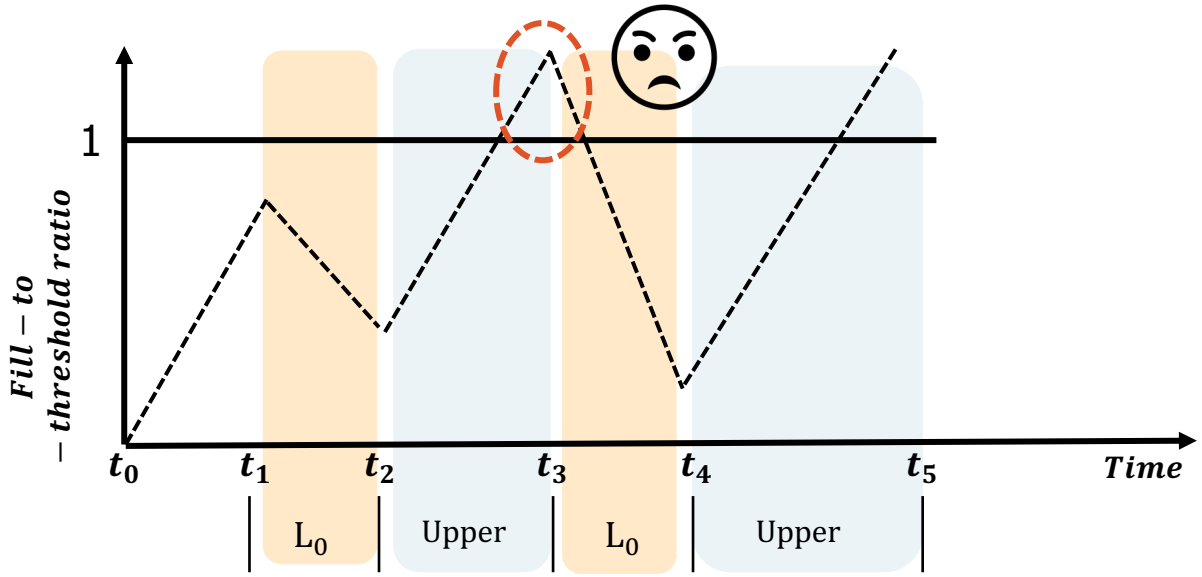
Visualize L_0 status with fill-to-threshold ratio

- Fill-to-threshold ratio = $\frac{L_0\ size}{L_0\ stall\ threshold}$
  - $L_0\ stall\ threshold$: Size that triggers L_0 stall
  - Fill-to-threshold ratio > 1 ➔ L_0 stall triggered



L_0 stall on write intensive workload

- Large upper-level compaction causes L_0 stall
  - L_0 compaction cannot be triggered on time



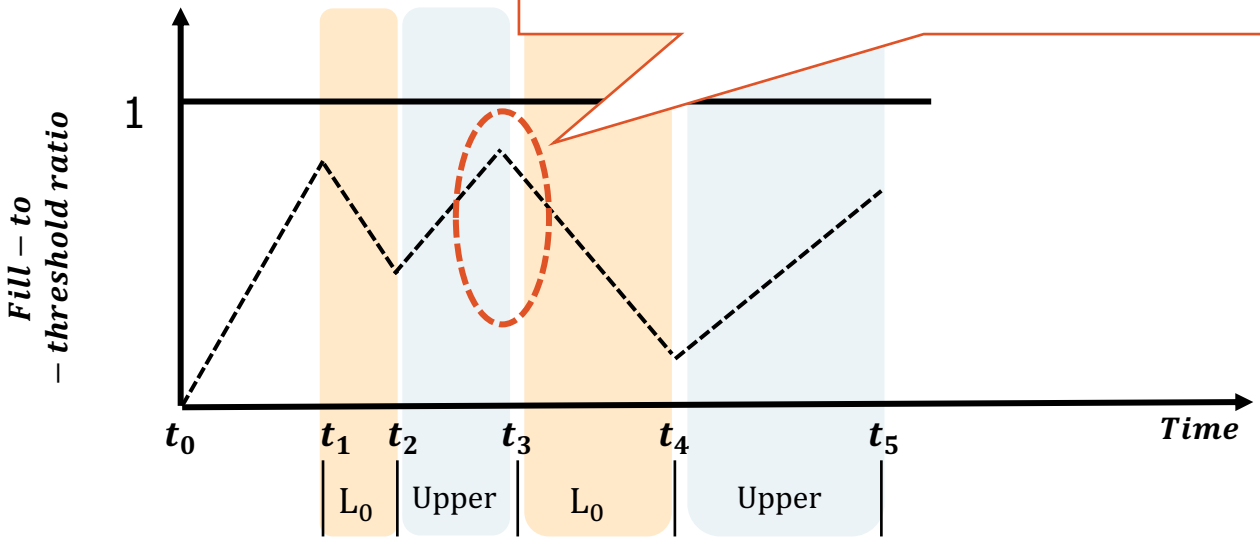
< L_0 status on write intensive workload>

Compaction strategy for write intensive workload

- Adjusting upper-level compaction(ULC) size could avoid L_0 stall

  - With ULC latency, next L_0 compaction latency must be considered

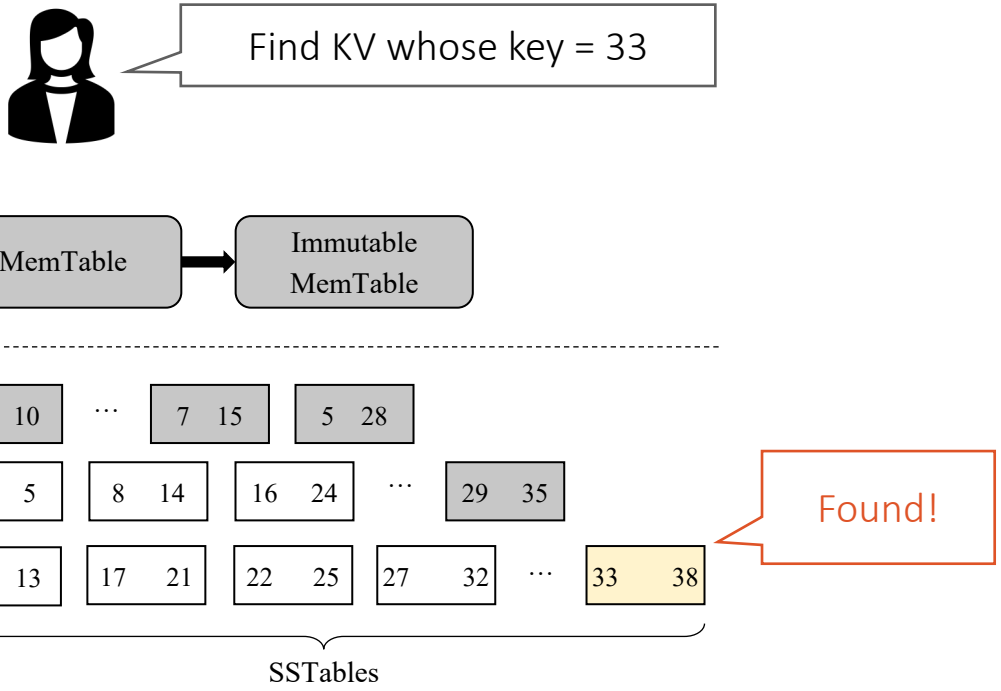
Largest size such that L_0 stall does not occur
- Not too frequent
- Not too long



< L_0 status with adjusted ULC size >

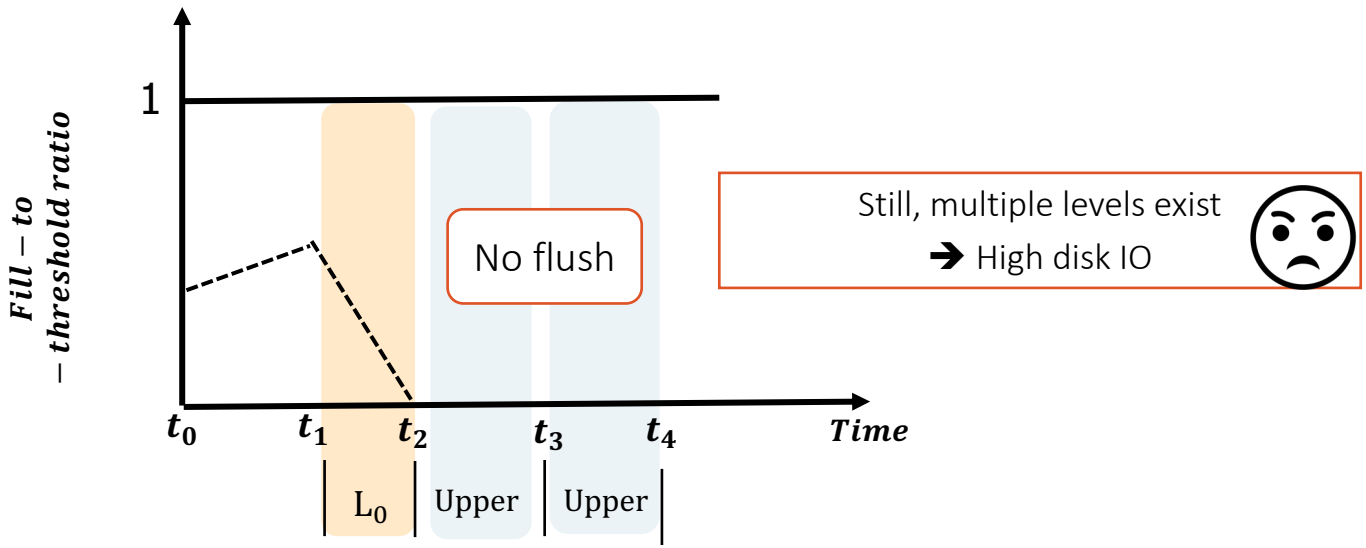
Multiple disk access on read intensive workload

- For read, LSM KVS has to check multiple components
  - Multiple disk accesses 



No L_0 stall on read intensive workload

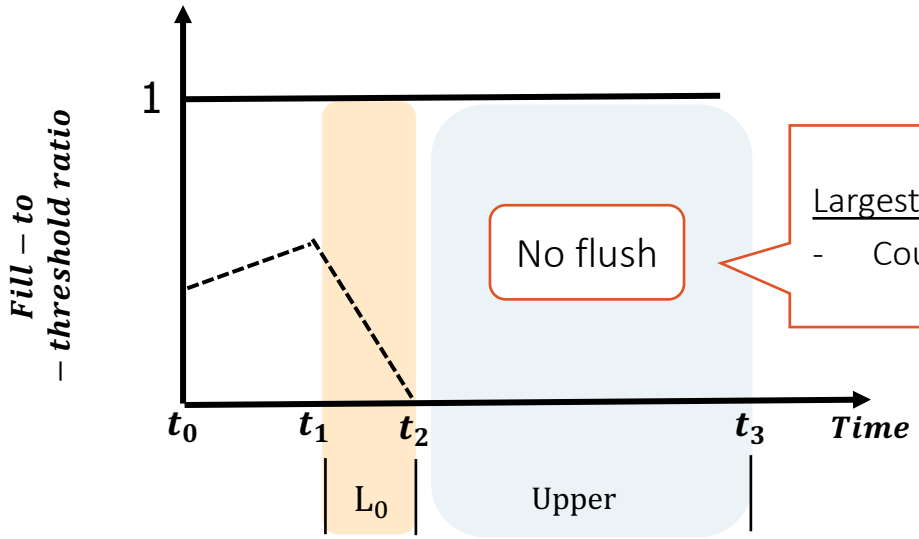
- On read intensive workload, L_0 stall is rare
  - No flush on read-only workload



< L_0 status on read intensive workload>

Compaction strategy for read intensive workload

- Upper-level compaction size can be very large
  - Helpful in reducing the number of levels to access

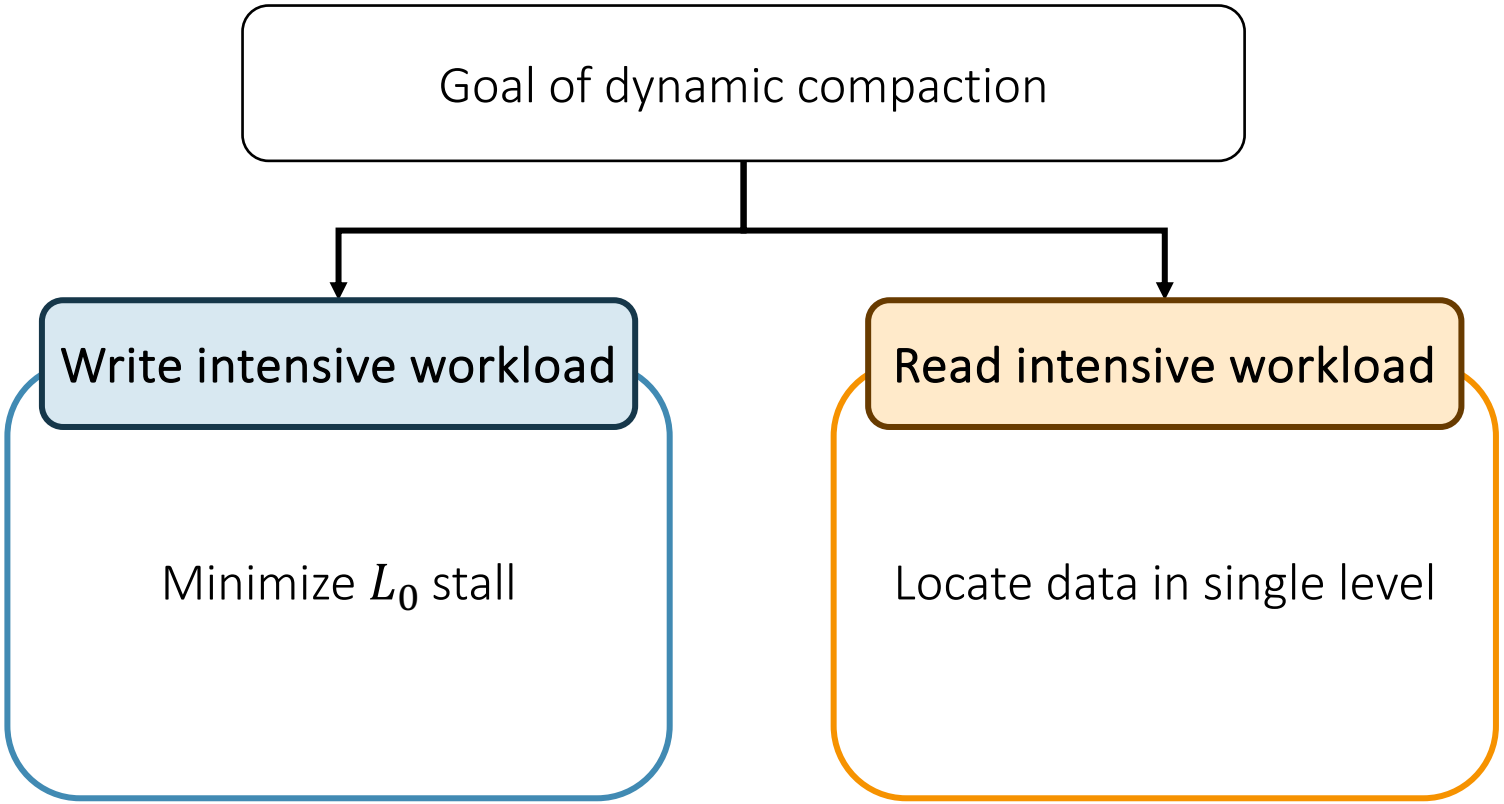


Largest size such that L_0 stall does not occur

- Could be very large as long as no L_0 stall occurs

< L_0 status with adjusted ULC>

Goal of dynamic compaction

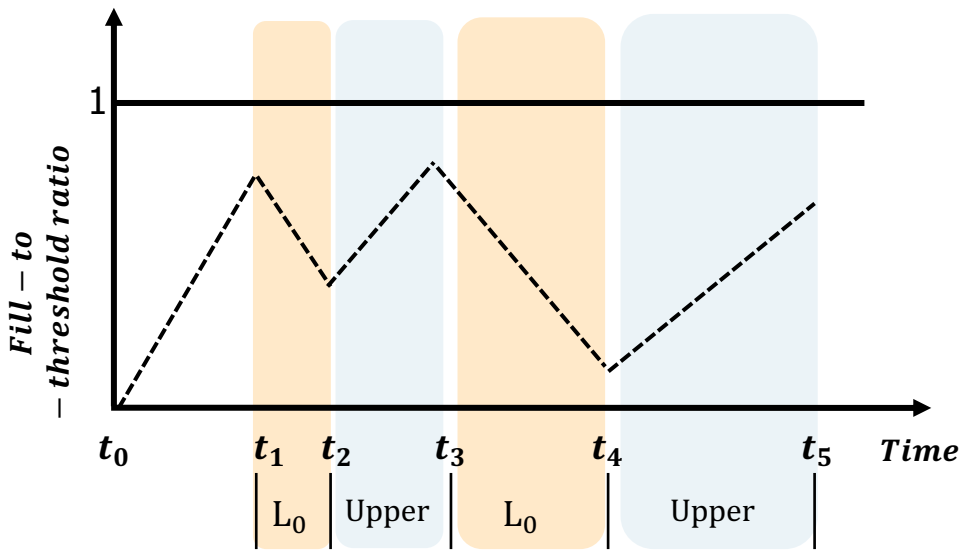


Compaction size depending on workload pattern

Depending on workload pattern, different compaction size is preferred

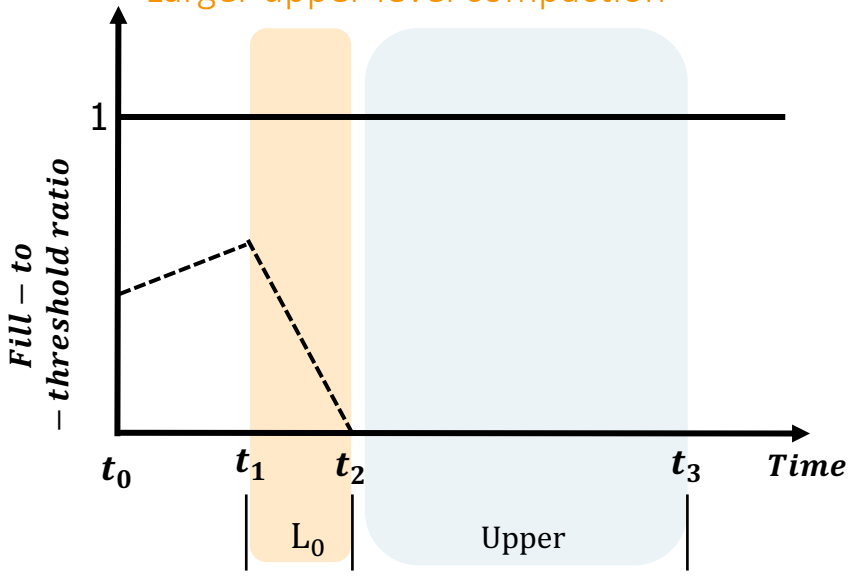
Write intensive workload

Smaller upper-level compaction



Read intensive workload

Larger upper-level compaction



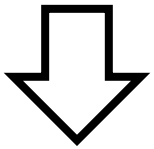
DOPA-DB

Compaction Size Recommender :

Dynamic compaction size recommendation for dynamic workload

+

Tiered LSM-tree based KV store with multiple key ranges

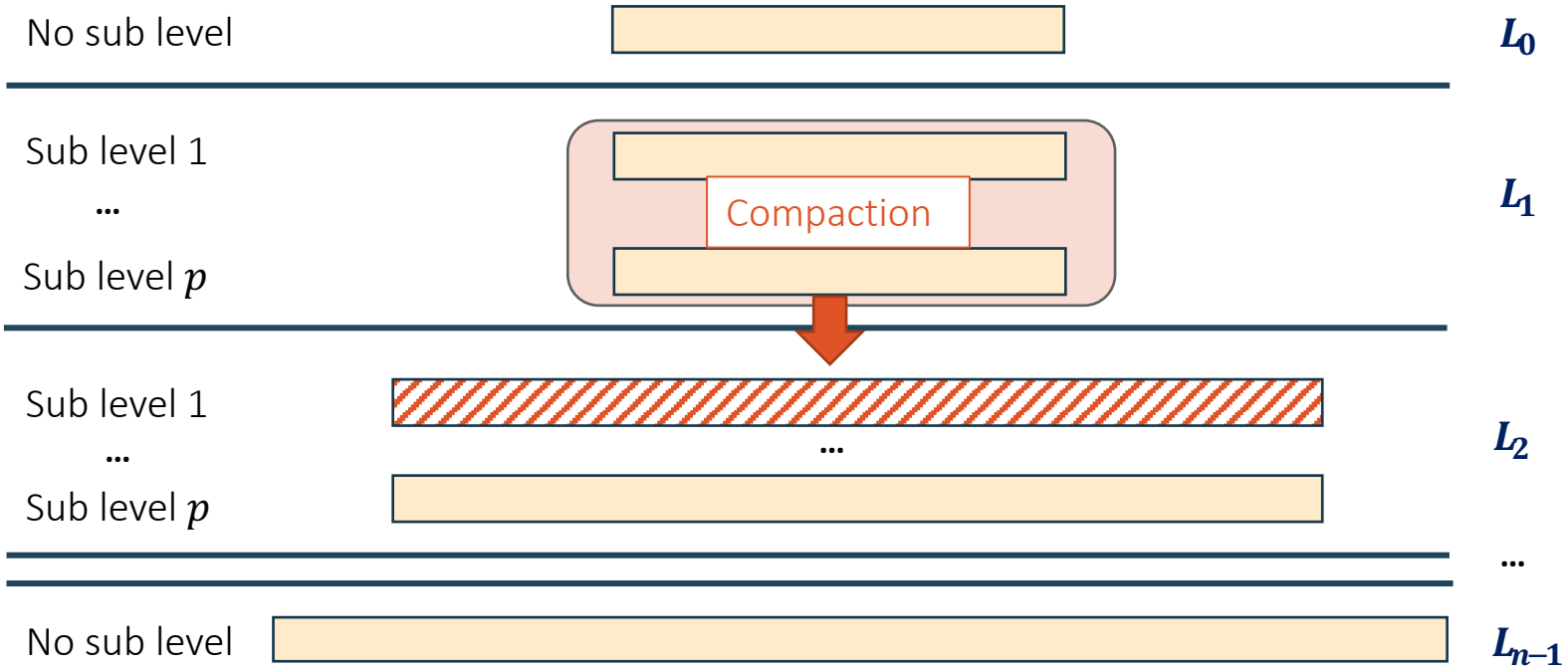


DOPA-DB

Dynamic wOrkload Pattern Aware LSM-Based KV store

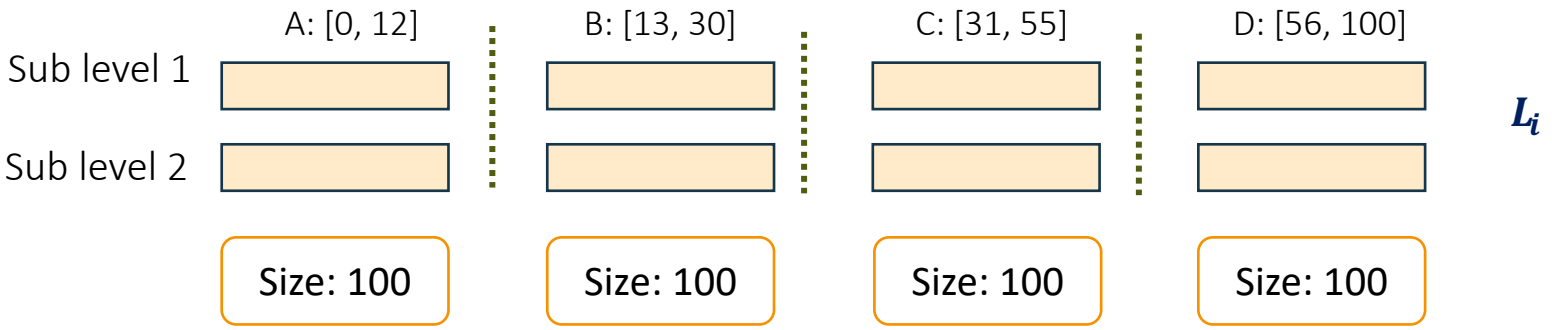
DOPA-DB based on tiered LSM KVS

- Each level, except for L_0 and last level, consists of p sub-levels
- Tiered LSM KVS avoids compaction at neighboring levels except for the last level



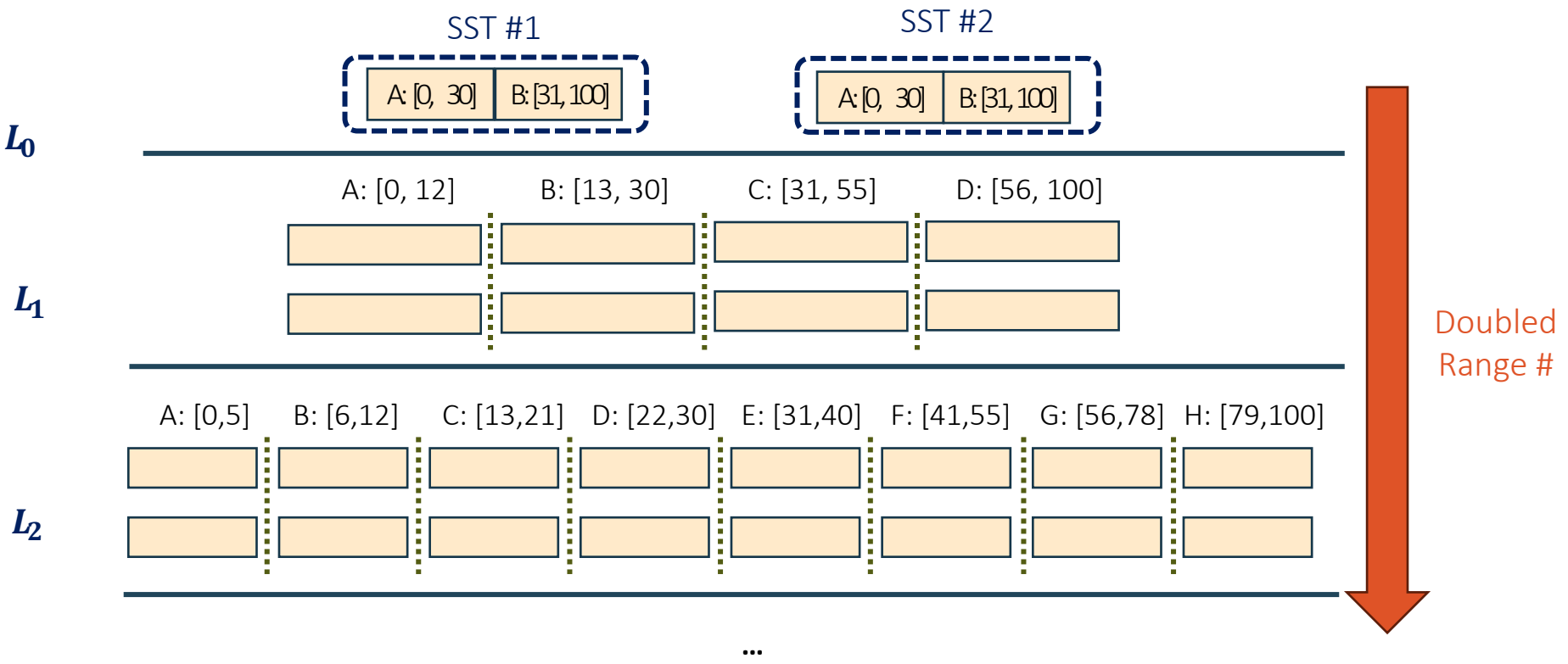
DOPA-DB with multiple key ranges

- To support dynamic compaction strategy, range is used for compaction unit
  - Not overlapped
  - Uniformly sized



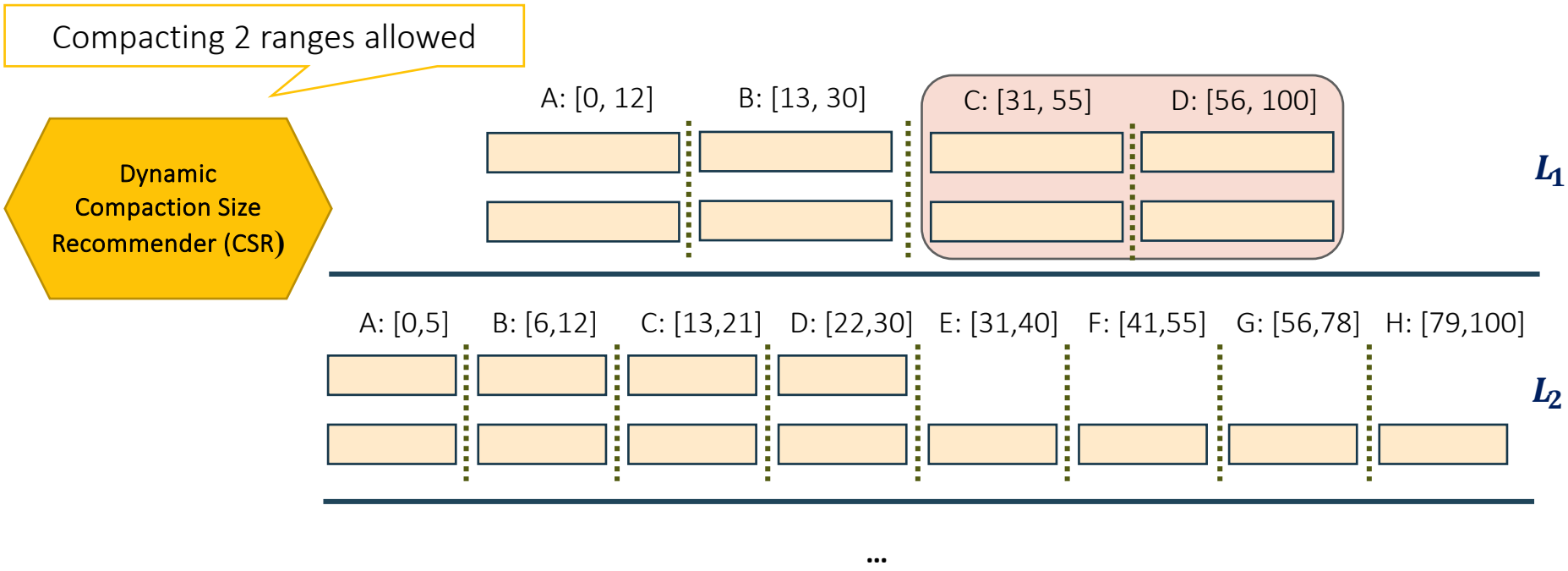
Tiered LSM KVS with multiple key ranges for DOPA-DB

- Every level has multiple key ranges
  - Number of range grows based on range amplification factor



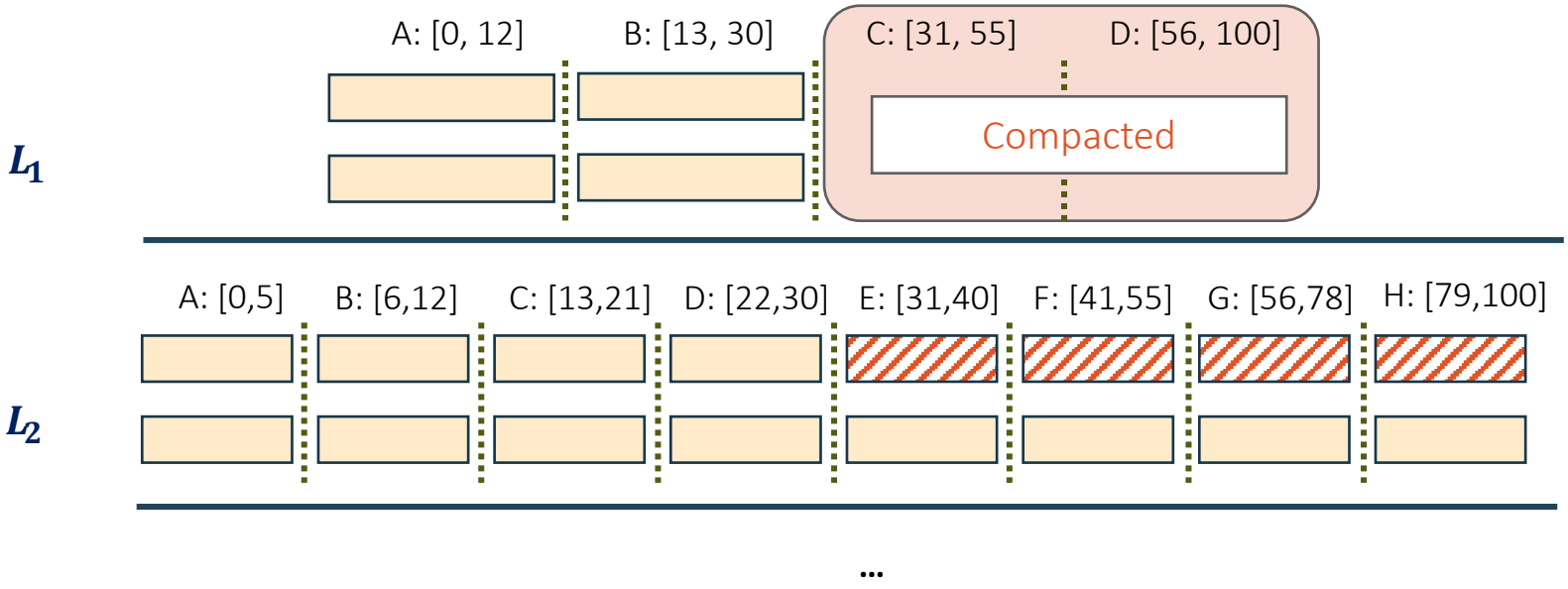
Tiered compaction with multiple key range

- Tiered compaction with multiple key ranges is performed based on sub-levels
  - Number of compaction units is decided by CSR



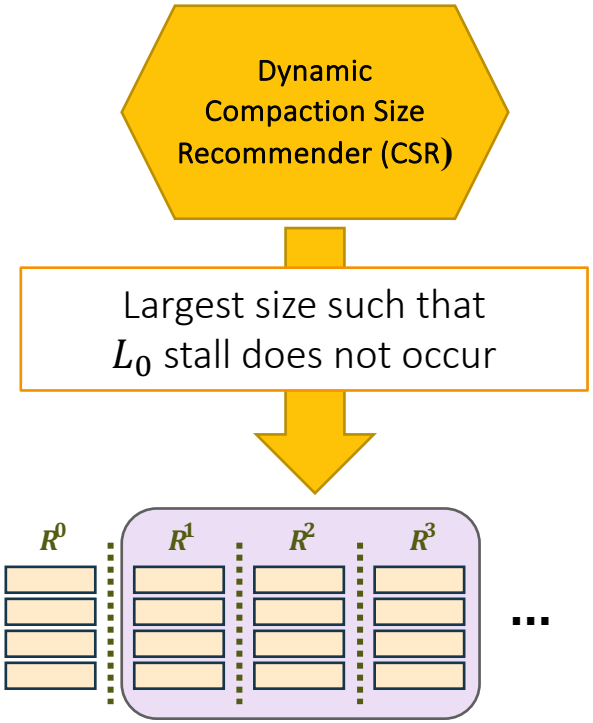
Placement after tiered compaction

- Placement of compaction output maintains non overlapped status of each sub level



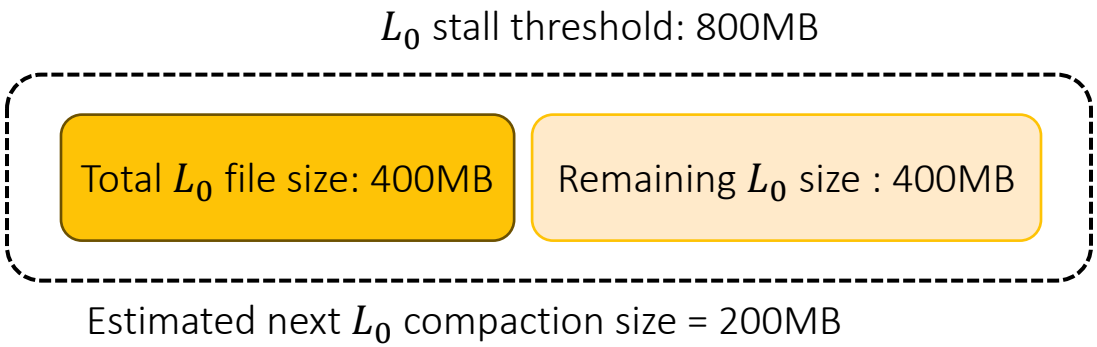
DOPA-DB with CSR

- Compaction Size Recommender(CSR) suggests size of upper-level compaction



How CSR calculates ULC size

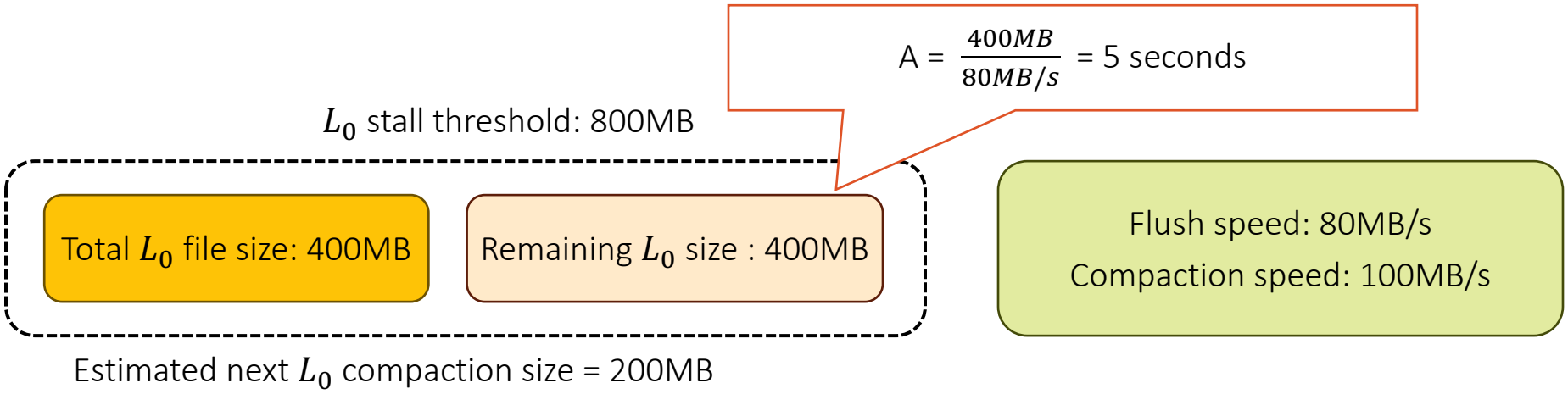
- To calculate ULC (upper-level compaction) size, values of parameters are estimated
  - Monitor flush & compaction speed during fixed time window (5 sec)
  - Remaining L_0 size
  - Estimated next L_0 compaction size



Flush speed: 80MB/s
Compaction speed: 100MB/s

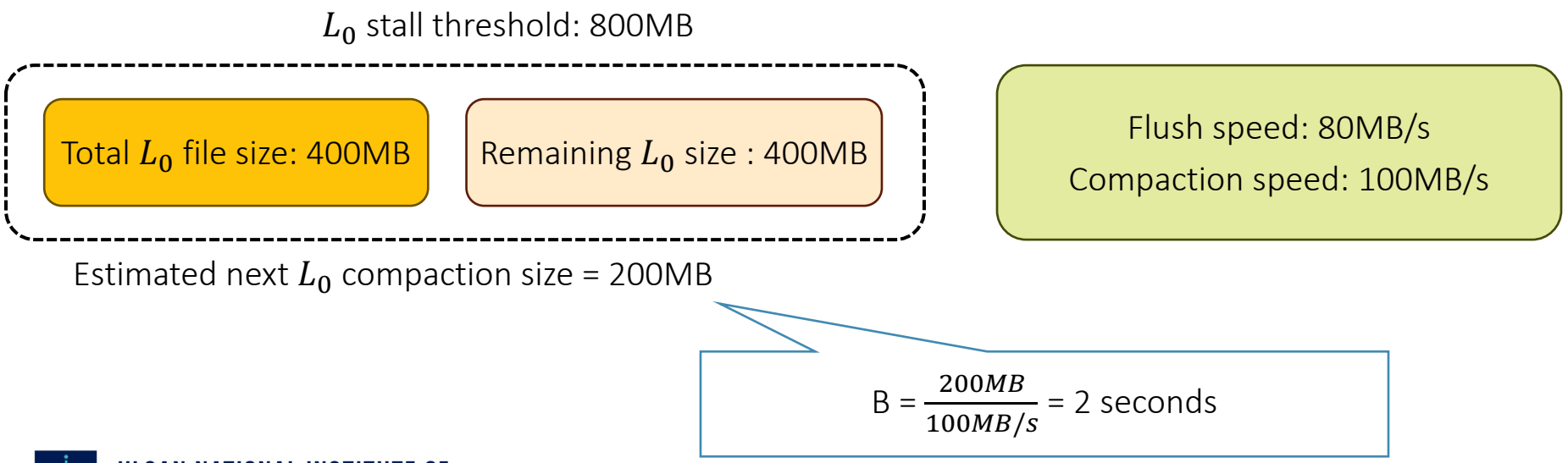
How CSR calculates ULC size: Step 1

- Step 1. Calculate latency for ULC (upper-level compaction)
  - A = Latency until L_0 stall threshold reached = $\frac{Remaining\ L_0\ size}{Flush\ speed}$



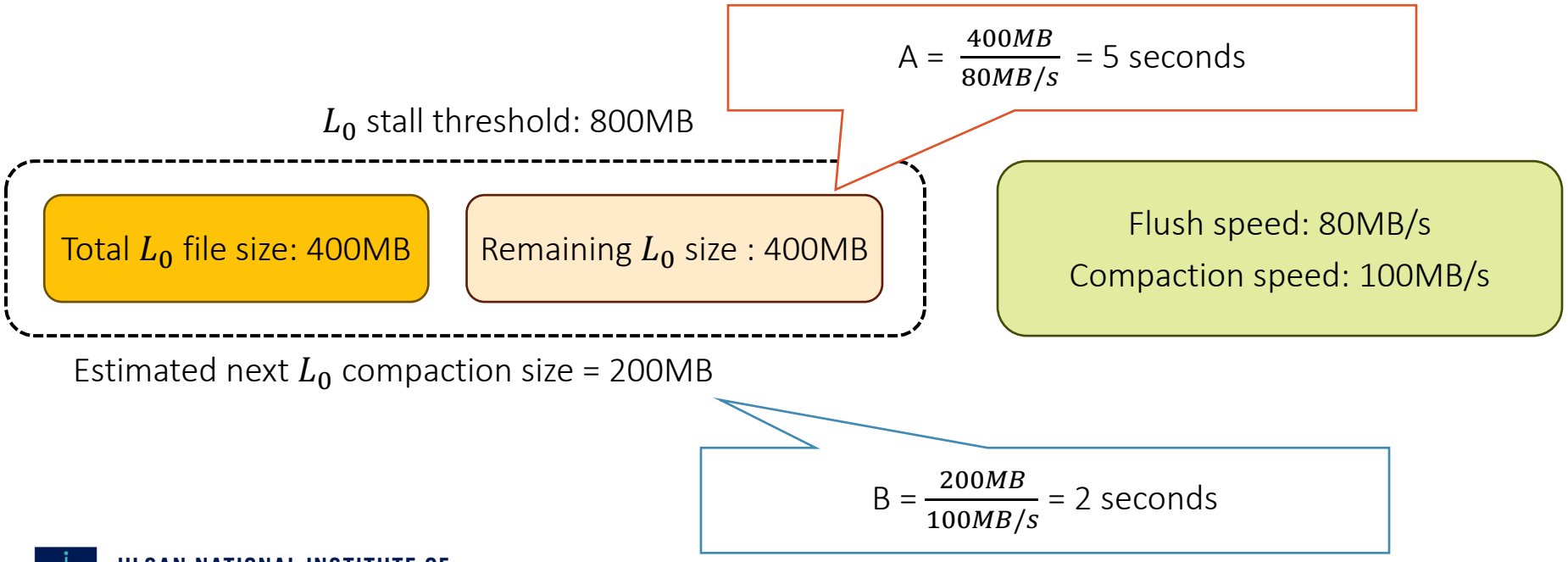
How CSR calculates ULC size: Step 1

- Step 1. Calculate latency for ULC (upper-level compaction)
  - B = Next L_0 compaction latency = $\frac{Estimated\ next\ L_0\ compaction\ size}{Compaction\ speed}$



How CSR calculates ULC size: Step 1

- Step 1. Calculate latency for ULC (upper-level compaction)
  - Latency until L_0 stall threshold reached(A) - Next L_0 compaction latency(B)
  - A - B = 3 seconds allowed for upper-level compaction



How CSR calculates ULC size: Step 2

- Step 2. Calculate compaction size within latency for ULC
  - Based on result of Step 1, we can obtain compaction size
    - Compaction size = latency of compaction × compaction speed

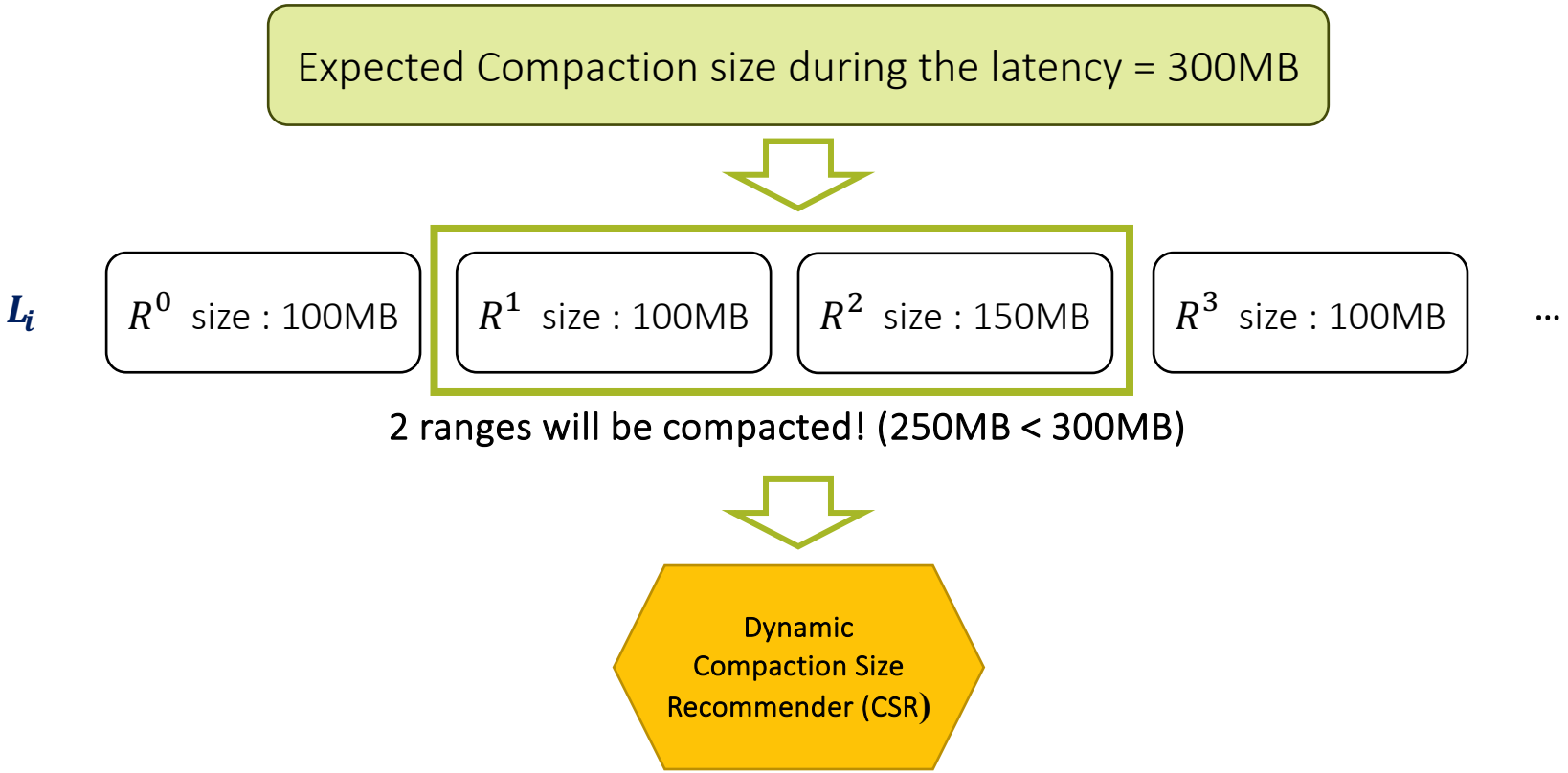
Latency for upper-level compaction = 3 seconds

Flush speed: 80MB/s
Compaction speed: 100MB/s

➔ Compaction size = 3 seconds × 100MB/s = 300MB

How CSR calculates ULC size: Step 3

- Step 3. Find number of ranges corresponding to the size



Evaluation setup

Testbed

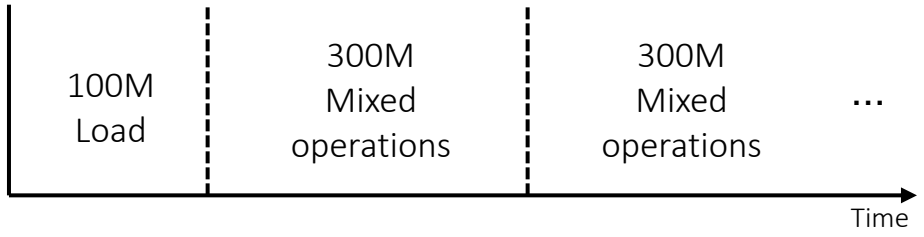
- 32-core Intel Xeon CPU E5-2683 v4
- 32GB DRAM
- Samsung 870 EVO SSD 1TB
- Ubuntu 20.04.4

Parameter

- Enable Direct I/O
- Number of sub levels: 4
- Range amplification factor: 4
- Number of L0 ranges : 4

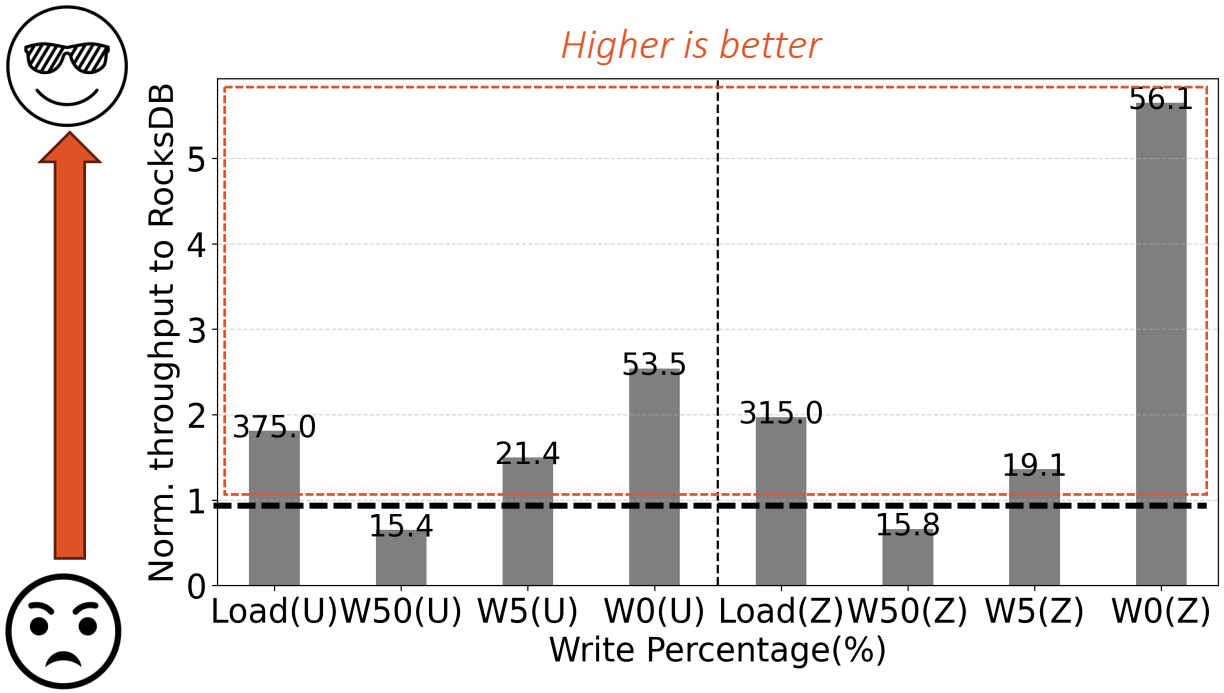
Workload

- 16B key + 100B value
- Run with 16 threads



Evaluation on mixed workload

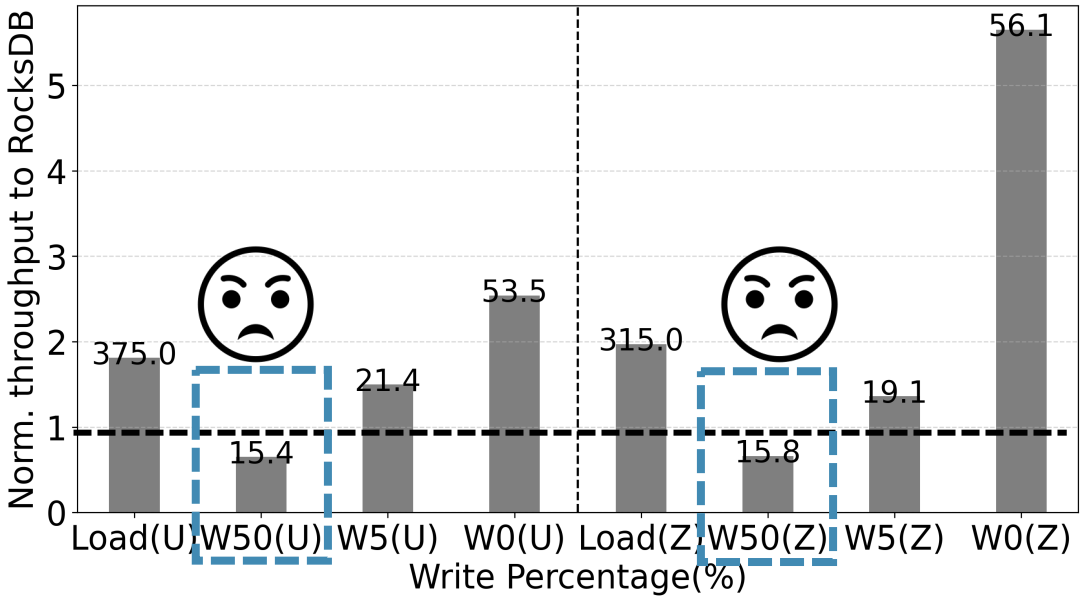
- For write/read intensive workloads, throughput of DOPA-DB is higher
  - Dynamic compaction size recommendation helps to optimize compaction for each operation



	Workload description
Load	Insert 100%
W50	Insert 50% + Read 50%
W5	Insert 5% + Read 95%
W0	Read 100%
U/Z	Uniform / Zipfian

Evaluation on mixed workload

- Multiple sub levels increases the number of disk access when workload pattern changes
- Large amount of upper-level compaction causes disk bandwidth competition



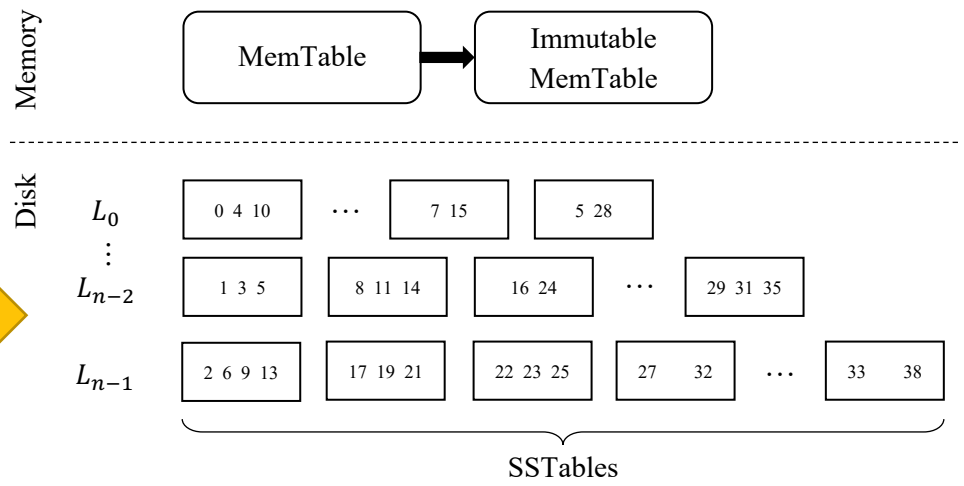
	Workload description
Load	Insert 100%
W50	Insert 50% + Read 50%
W5	Insert 5% + Read 95%
W0	Read 100%
U/Z	Uniform / Zipfian

Discussion & Future work

Nimble and accurate adaptation

- L_0 size
- Flush speed
- Compaction speed

Dynamic Compaction Size Recommender (CSR)

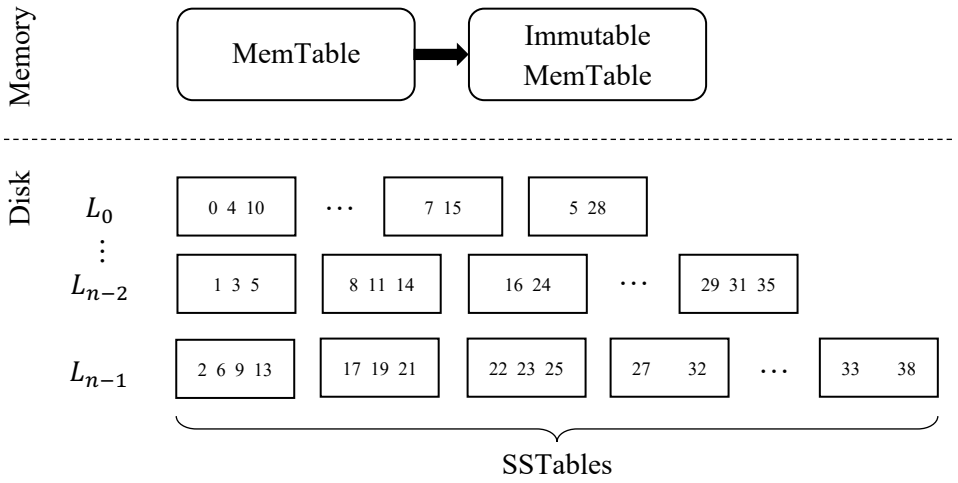


Discussion & Future work

Nimble and accurate adaptation

- L_0 size
- Flush speed
- Compaction speed
- Operation ratio?
- Bandwidth competition?

Dynamic Compaction Size Recommender (CSR)



Discussion & Future work

Using multiple compaction threads

- Current DOPA-DB **only** supports single-threaded compaction
- Controlling compaction with multiple threads has to be considered

Parameters of DOPA-DB

- Parameters of current DOPA-DB are chosen **empirically** with 2 considerations
  - Range must not make size of SST too small
  - Size of compaction unit size must not be too large

Skewed datasets

- Key order distribution of real-world dataset is **dynamic**
- Dynamic range readjustment is necessary for uniformly sized ranges

Conclusion

- DOPA-DB : Dynamic wOrkload Pattern Aware LSM-based KV store
  - Advocate dynamic compaction size for dynamic workload
  - Tiered based LSM KVS can control fine-grained compaction size

- Evaluation on mixed workload
  - Potential benefits of dynamic compaction
  - Insight into research directions for dynamic compaction strategies

- Envisioned dynamic approach can further drive performance improvements

Thank you

heejin5178@unist.ac.kr