FlashKV: Accelerating KV Performance with Open-Channel SSDs

JIACHENG ZHANG, YOUYOU LU, JIWU SHU, and XIONGJUN QIN

2023.08.11

Presentation by Kim Boseung, Shin Suhwan

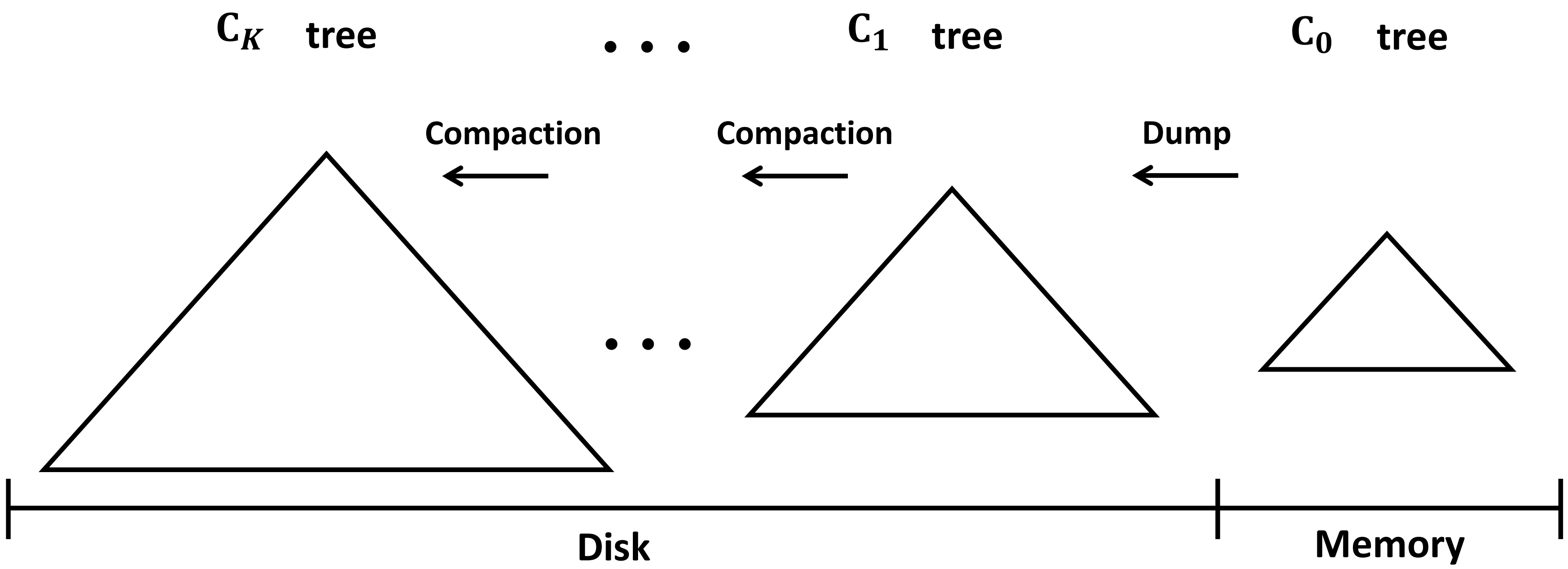
kbskbs1102@dankook.ac.kr
shshin@dankook.ac.kr

단국대학교 DANKOOK UNIVERSITY
Dankook University System Software Laboratory

Contents

1. Introduction

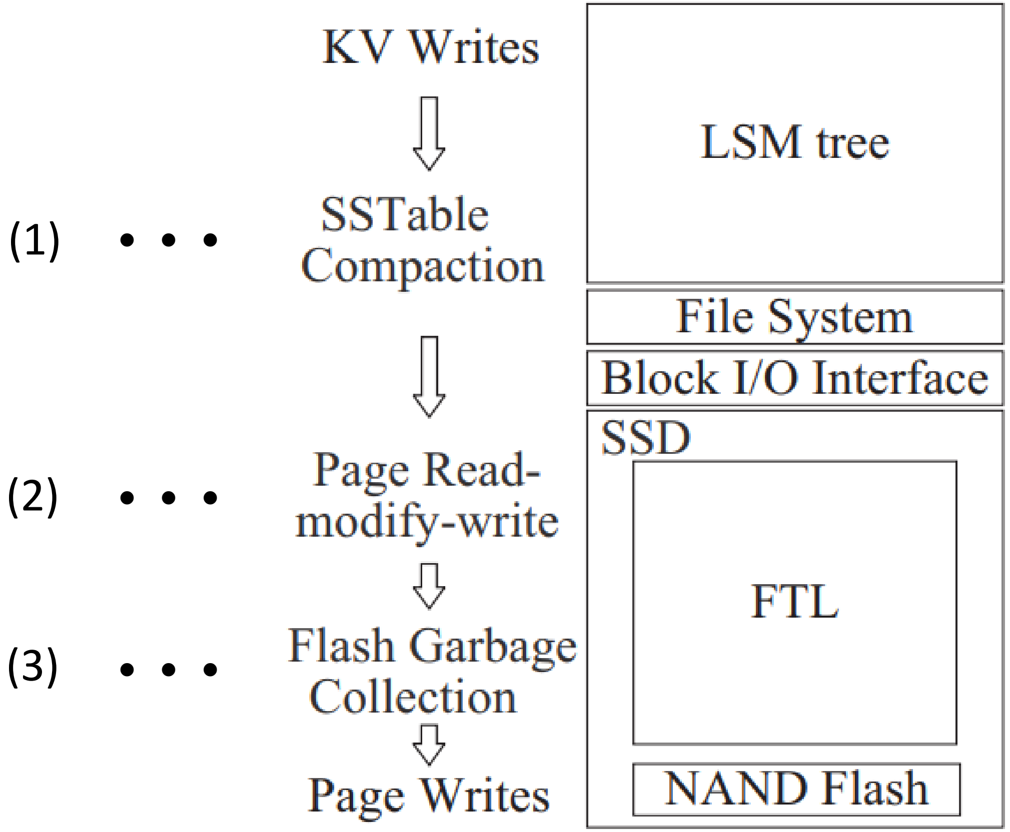
- **LSM-Tree**



C_K tree
. . .
C_1 tree
C_0 tree
Compaction
Compaction
Dump
. . .
Disk
Memory

1. Introduction

- **LSM-Tree with SSD**



출처: Wu, Sung-Ming et al. "KVSSD: Close integration of LSM trees and flash translation layer for write-efficient KV store."

1. Introduction

- **Unawareness between KV stores & SSDs**

1. Triple redundant and unnecessary functions - mapping table, GC, space management

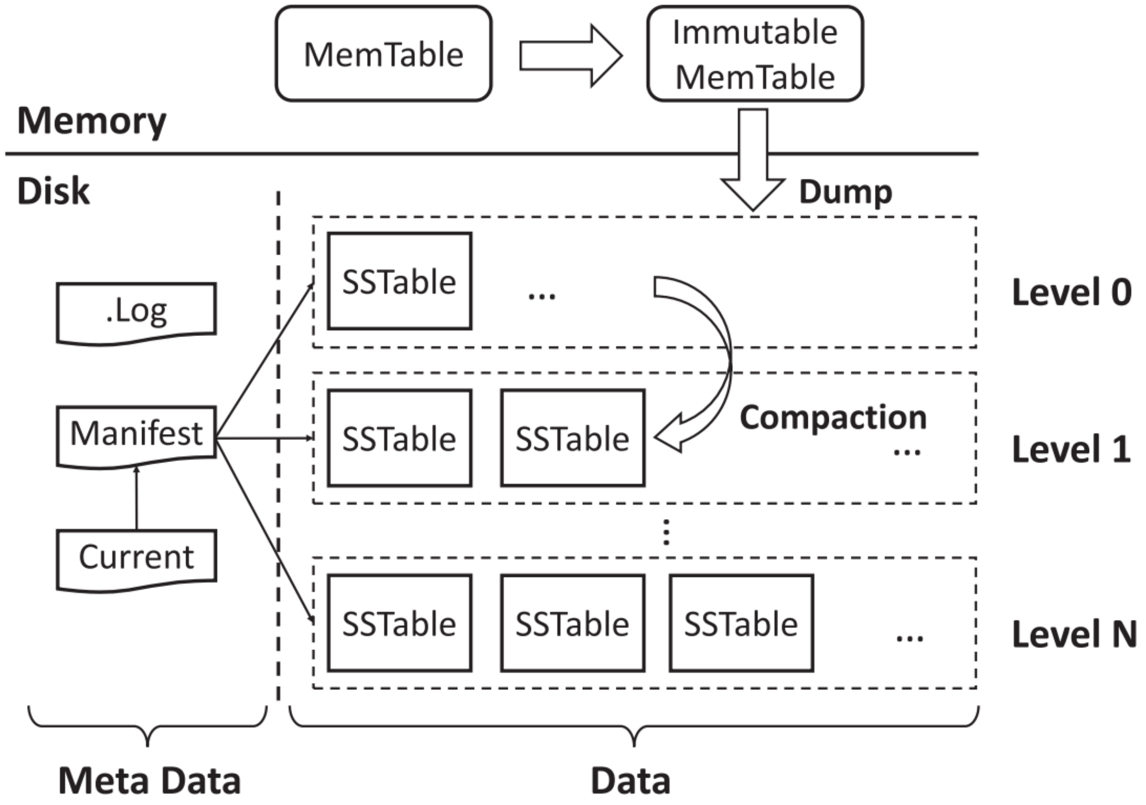
2. Internal parallelism of SSDs has not been well-leveraged

3. The domain knowledge of LSM-tree has not been exploited

2. Background

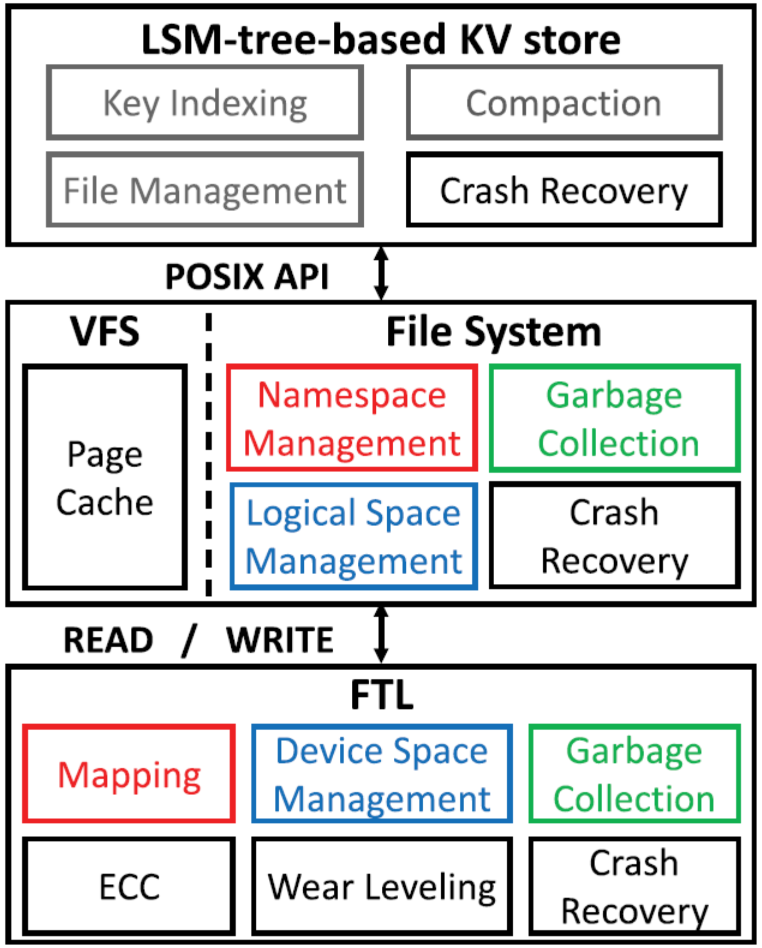
2. Background

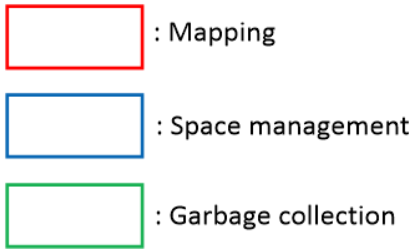
- **Level DB**



2. Background

- **KV Store over Commercial SSD**





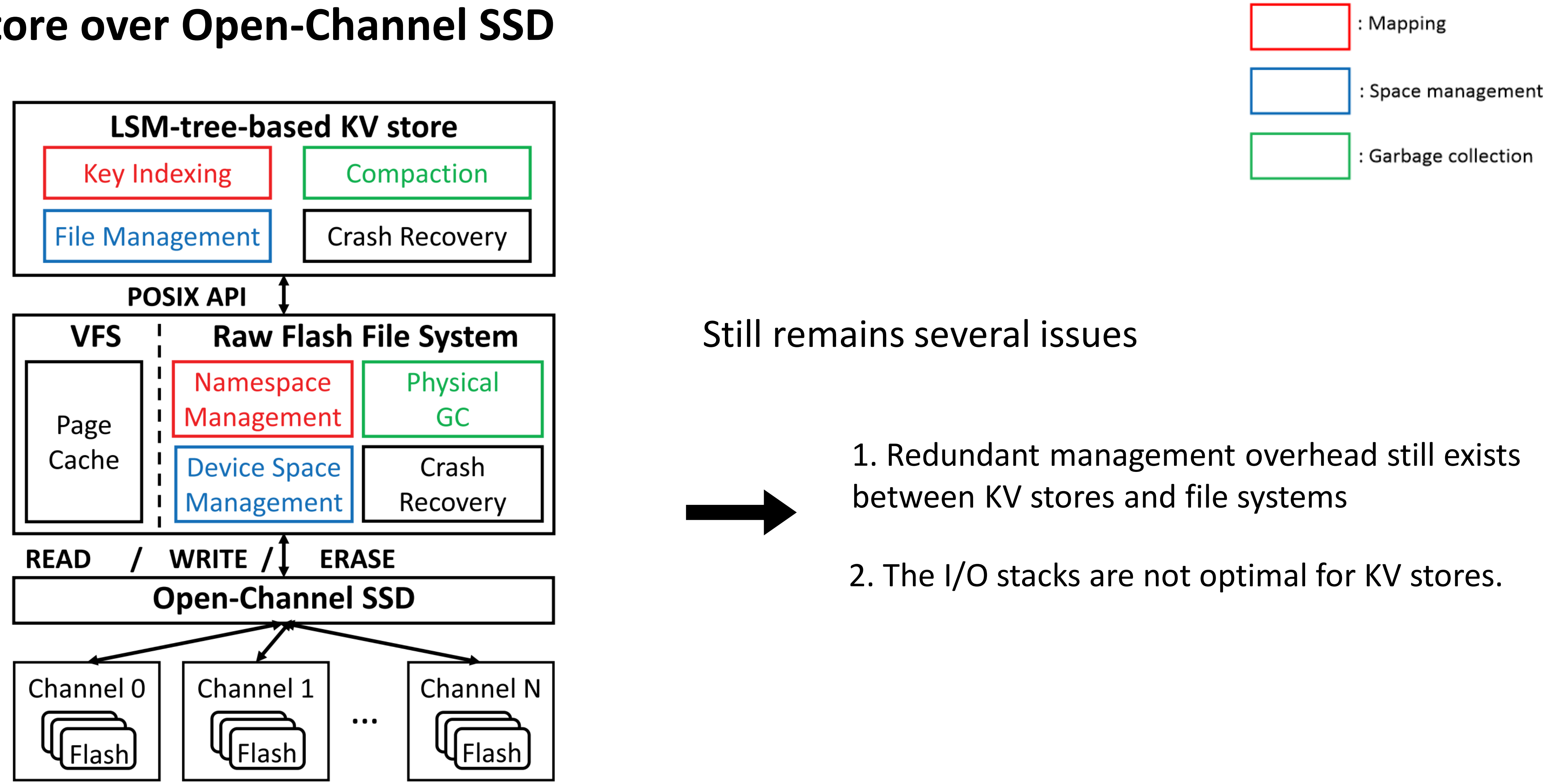
Redundant functionalities in FTL and filesystems

→

1. Inefficiency in I/O processing
2. Significant Write Amplification

2. Background

- **KV Store over Open-Channel SSD**



Still remains several issues

1. Redundant management overhead still exists between KV stores and file systems

2. The I/O stacks are not optimal for KV stores.

3. Design

- Parallel Data Layout
- Adaptive Parallelism Compaction
- Compaction-Aware Cache
- Priority-Based Scheduler

Design Summary

- **Parallel Data Layout**
  - Distribute blocks to multiple channels: SSTable -> SuperBlock (unique ID & offset)

- **Adaptive Parallelism Compaction**
  - Adjust parallelism by workload (Intensive for Read or Write)

- **Compaction-Aware Cache**
  - Separate the two types of read operations (Client or Compaction)

- **Priority-Based Scheduler**
  - Consider priority (Foreground requests or Background requests)

Parallel Data Layout

- SuperBlock

  - Unique ID

  - A group of blocks at an offset within a channel

  - Mapping SST Files

  Metadata

  - First page: file name

  - Last page: SSTable version & SuperBlock state

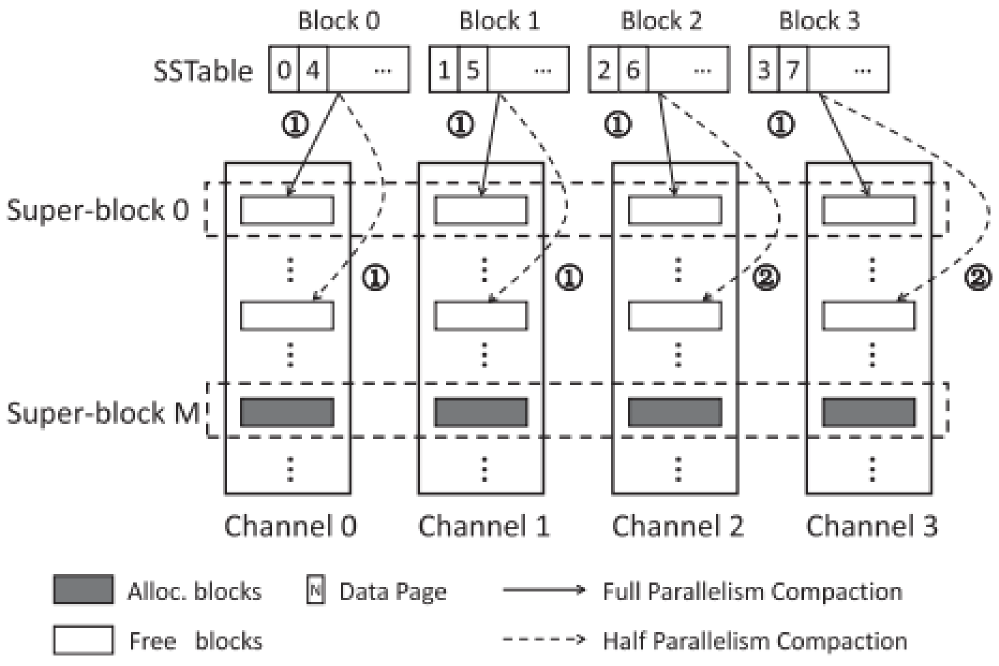


Fig. 4. The Parallel Data Layout of FlashKV.

Adaptive Parallelism Compaction

- NAND Flash operation
  - Overhead: Write operation >> Read operation
    (Erase before write)

- Problem: [Write -> Read] [Read -> Write]

- Solution: Case for parallelism
  Case 1) Write intensive workload : Full parallelism
  Case 2) Read intensive workload : Half parallelism

※ Write intensive threshold: write requests 30% ~

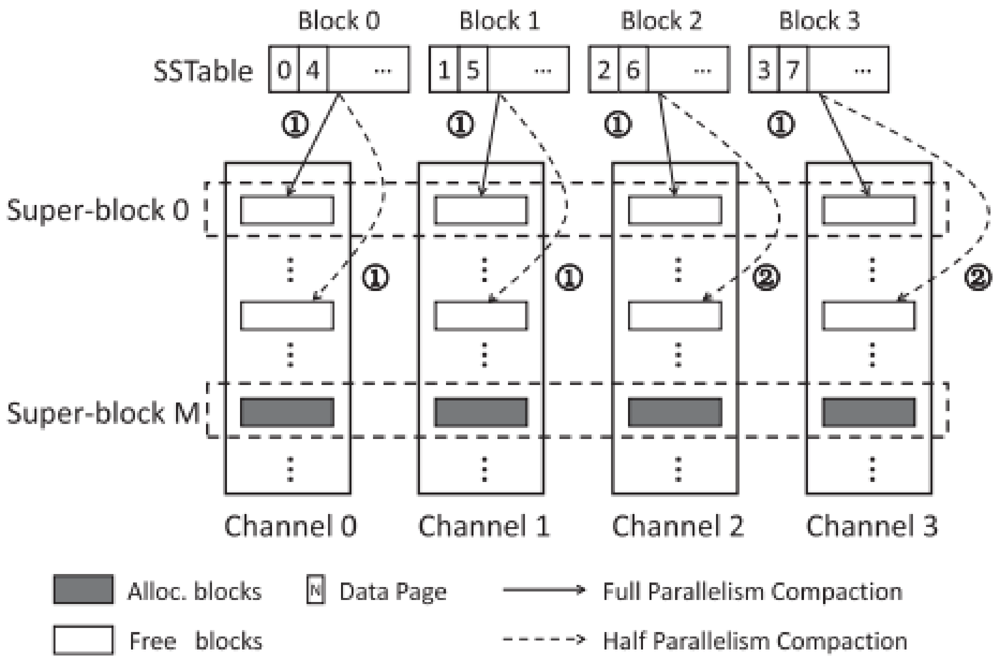


Fig. 4. The Parallel Data Layout of FlashKV.

Compaction-Aware Cache

- Type of read request
  - Clients' reads (get)
    - Small request size
    - Unnecessary prefetch
    - Unit: Page
  - Compaction reads (By compaction thread)
    - Large request size
    - Prefetching
    - Unit: Batch (MS_Batch move LRU list's head)

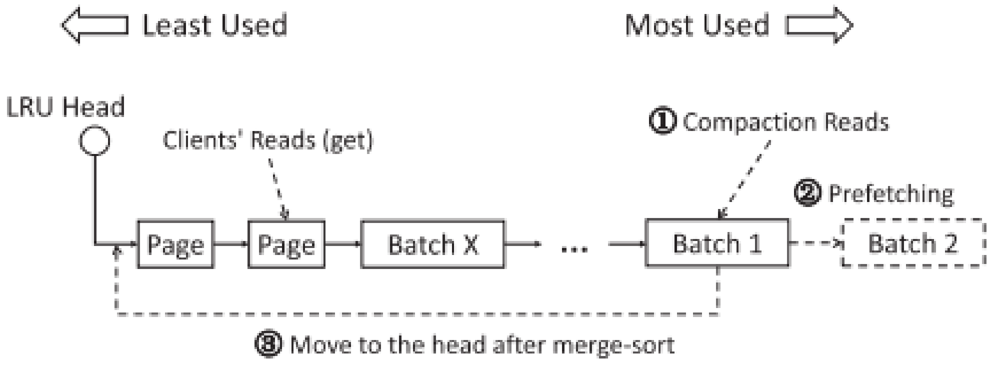


Fig. 5. Compaction-aware Cache.

Priority-Based Scheduler

- I/O Requests
  - Foreground request: requests for Client (get, put ..)
  - Background request: request for Merge-sort

- Priority
  - Foreground > Background
  - Read > write
  - Erase (less than free space 20%) : the highest priority
  - Erase (greater than free space 20%) : the lowest priority

- Priority: H_erase > [F_read > F_write] > [B_read > B_write] > L_erase

4. Evaluation

4. Evaluation

- **Experimental Setup**

Host Interface	PCIe 2.0 * 8
Number of Flash Cahnnel	34
Page Size	8KB
Block Size	2MB
Pages per Block	256
Read Bandwidth per Channel	94.62MB/s
Write Bandwidth per Channel	13.4MB/s

Parameter of SSD

<table>
<tr><th>KV store</th><th>File System</th><th>FTL</th></tr>
<tr><td>FlashKV</td><td>X</td><td>X</td></tr>
<tr><td rowspan="3">LevelDB</td><td>ParaFS</td><td>X</td></tr>
<tr><td>Ext4</td><td rowspan="2">PBlk</td></tr>
<tr><td>F2FS</td></tr>
</table>

System setup

4. Evaluation

- **Experimental Setup**

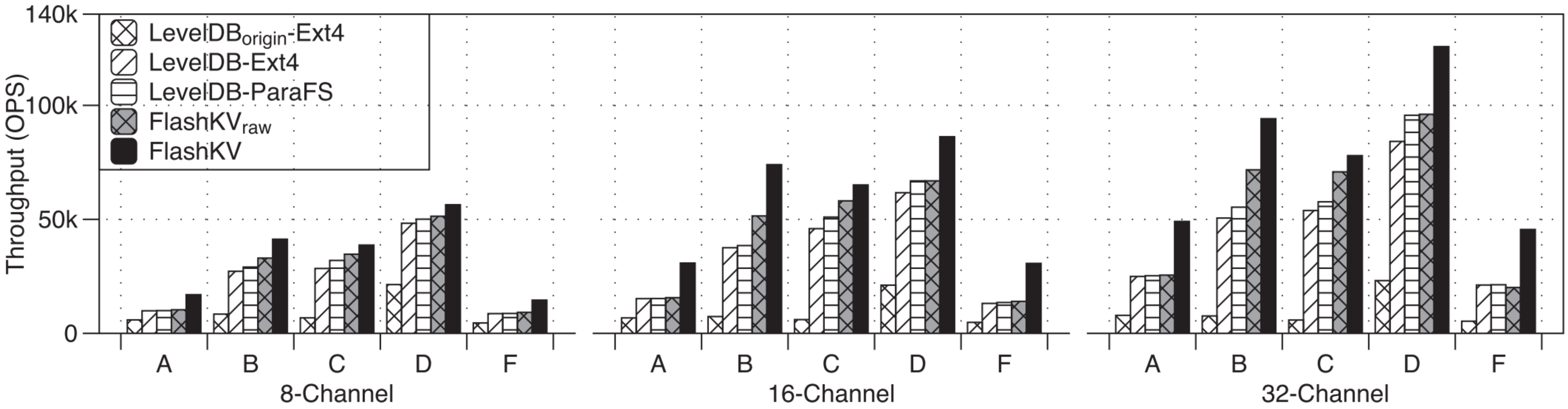
Name	Description	R/W	Dist.
Workload A	Update heavy workload	50/50	zipfian
Workload B	Read mostly workload	95/5	zipfian
Workload C	Read only workload	100/0	zipfian
Workload D	Read latest workload	95/5	latest
Workload F	Read-modify-write workload	50/50	zipfian

Workload Characteristics

4. Evaluation

- **with Light Write Traffic**

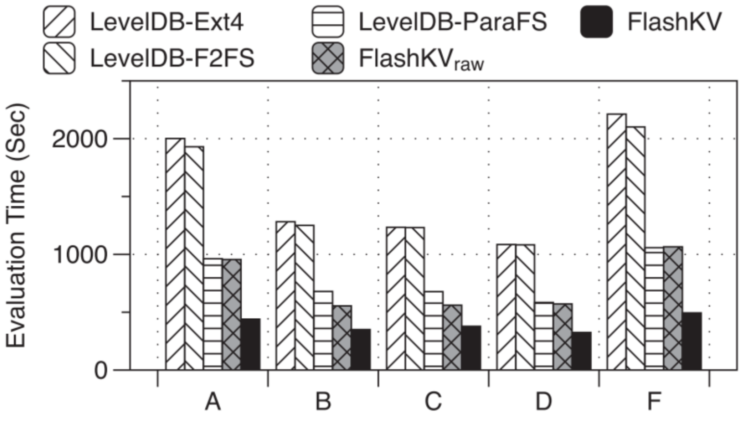
Name	Description	R/W	Dist.
Workload A	Update heavy workload	50/50	zipfian
Workload B	Read mostly workload	95/5	zipfian
Workload C	Read only workload	100/0	zipfian
Workload D	Read latest workload	95/5	latest
Workload F	Read-modify-write workload	50/50	zipfian



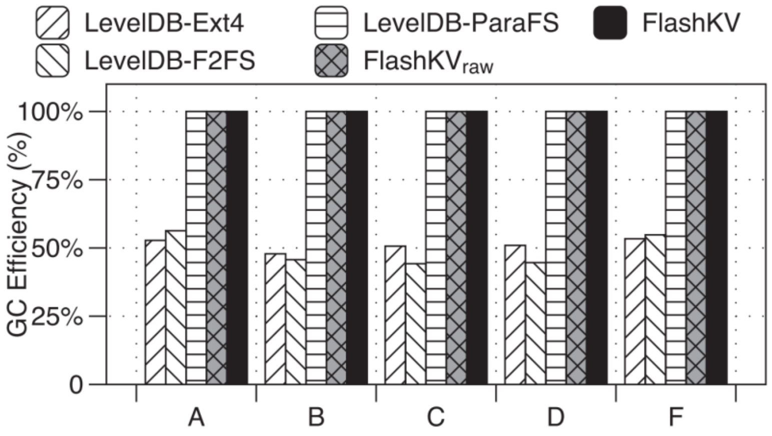
4. Evaluation

- **with Heavy Write Traffic**

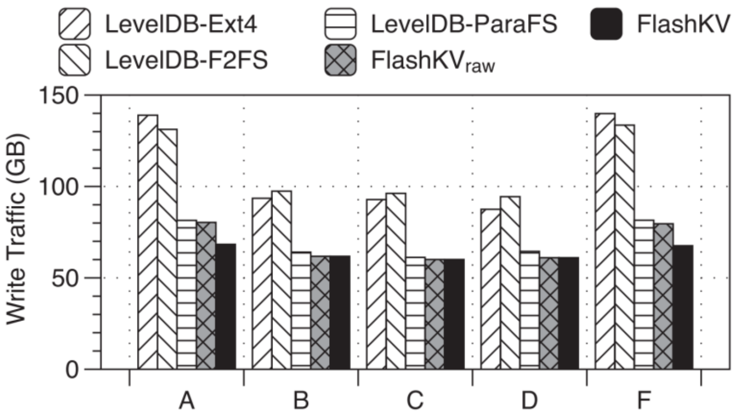
Name	Description	R/W	Dist.
Workload A	Update heavy workload	50/50	zipfian
Workload B	Read mostly workload	95/5	zipfian
Workload C	Read only workload	100/0	zipfian
Workload D	Read latest workload	95/5	latest
Workload F	Read-modify-write workload	50/50	zipfian



(a) Evaluation Time



(b) GC Efficiency

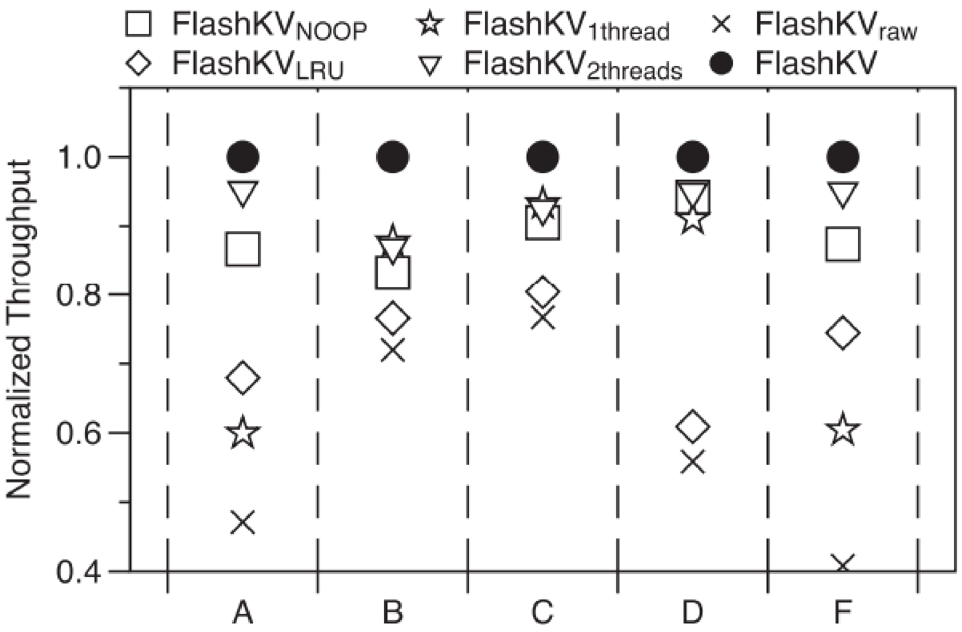


(c) Write Traffic to Flash

4. Evaluation

- **Optimization Breakdown**

Name	Description	R/W	Dist.
Workload A	Update heavy workload	50/50	zipfian
Workload B	Read mostly workload	95/5	zipfian
Workload C	Read only workload	100/0	zipfian
Workload D	Read latest workload	95/5	latest
Workload F	Read-modify-write workload	50/50	zipfian



5. Conclusion

5. Conclusion

- FlashKV is a user-space system that directly manages open-channel flash devices, eliminating overhead and compatibility issues.
- It leverages flash device details for optimized parallel data layout, compaction, cache, and I/O scheduling.
- Outperforming LevelDB, it reduces write traffic by 30-50% with up to 4.5× performance improvement.

FlashKV: Accelerating KV Performance with Open-Channel SSDs

JIACHENG ZHANG, YOUYOU LU, JIWU SHU, and XIONGJUN QIN

Thank you!
Q & A ?

2023.08.11

Presentation by Kim Boseung, Shin suhwan

kbskbs1102@dankook.ac.kr
shshin@dankook.ac.kr