VLDB2023
vancouver

NVRAMOS 2023
Workshop on Operating System Support for Next Generation Large Scale NVRAM
Oct 19-21, 2023, Jeju, Korea

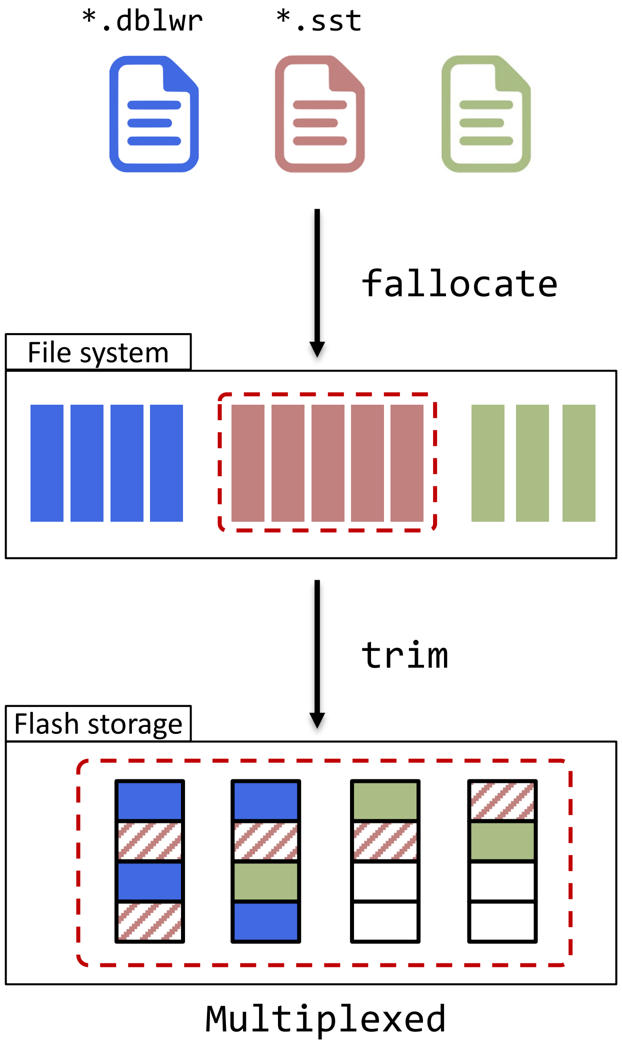
FlashAlloc: Dedicating Flash Blocks By Objects

Jonghyeok Park[*], Soyee Choi[**], Gihwan Oh[+], Soojun Im[**],
Moon-Wook Oh[**], Sang-Won Lee[+]

Hankuk University of Foreign Studies[*], Samsung Electronics[**]
Sungkyunkwan University[+]

Background

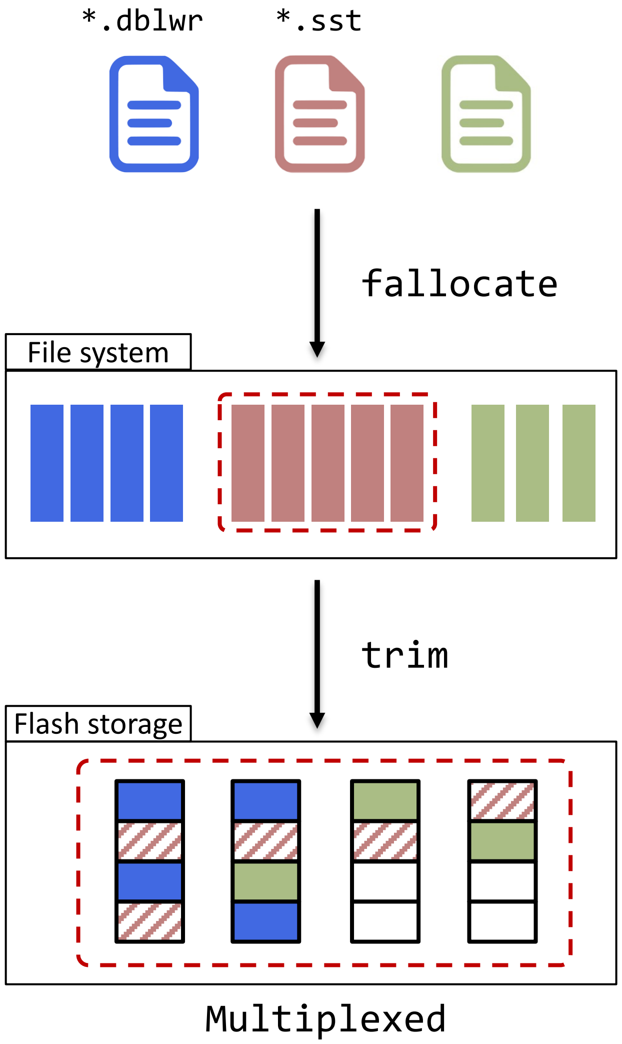
- Most data stores manages data by **logical** objects
  - ***SSTable*** in RocksDB
  - ***Double write buffer (DWB)*** in MySQL
  - ***Segment*** in F2FS



Background

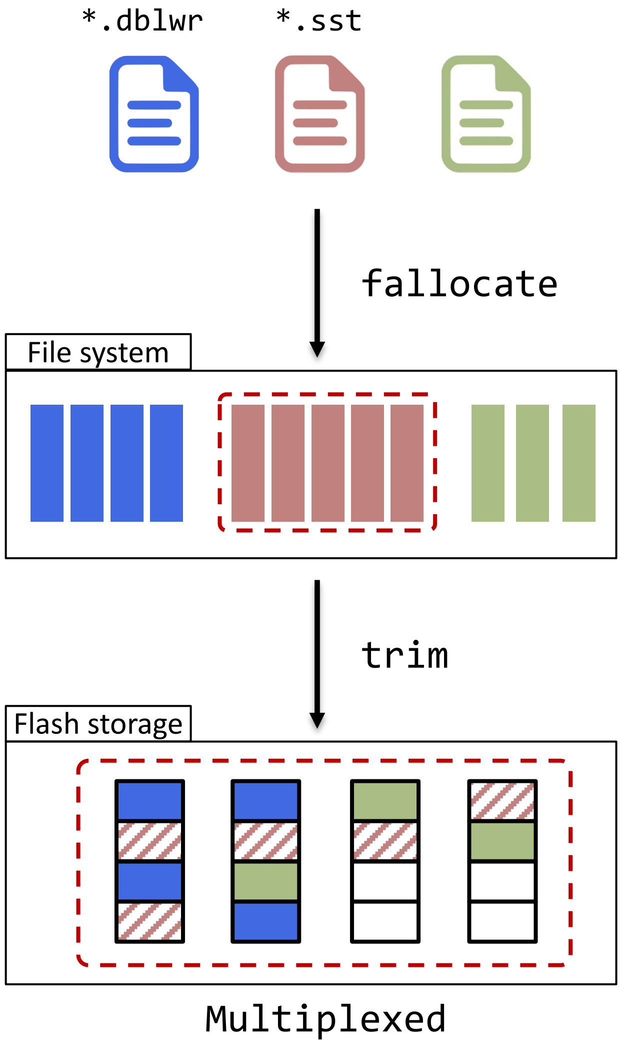
- Most data stores manages data by **logical** objects
  - ***SSTable*** in RocksDB
  - ***Double write buffer (DWB)*** in MySQL
  - ***Segment*** in F2FS

- Each objects is the **unit** of logical space allocation
  - Host: `fallocate()`
  - Flash Device: **stream-write-by-time**



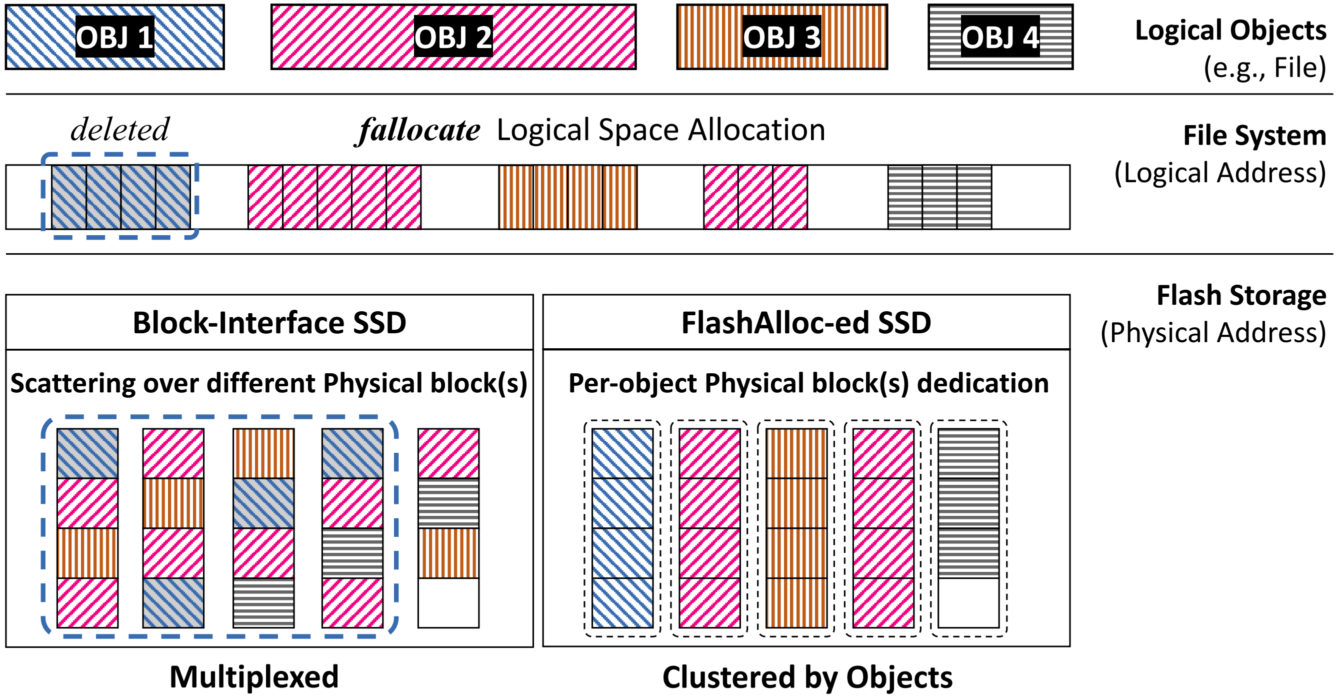
Background

- Most data stores manages data by **logical** objects
  - ***SSTable*** in RocksDB
  - ***Double write buffer (DWB)*** in MySQL
  - ***Segment*** in F2FS
- Each objects is the **unit** of logical space allocation
  - Host: `fallocate()`
  - Flash Device: **stream-write-by-time**
- Page deathtime
  - Host: TRIM command
  - Flash Device: **multiplexed with different deathtimes**



Multiplexing

- Flash blocks are multiplexed with pages with **different deathtimes**
  - **Stream-write-by-time policy**
  - **Copyback overhead in GC ⇒ write amplification ↑**

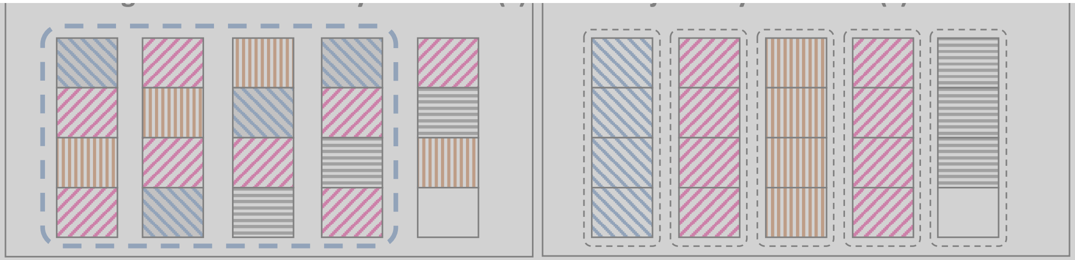


Multiplexing

- Flash blocks are multiplexed with pages with **different deathtimes**
  - **Stream-write-by-time policy**
  - **Copyback overhead in GC ⇒ write amplification ↑**

Flash devices are **object-oblivious**:

Host semantic about the object’s logical address range
CAN NOT cross the storage Interface WALL



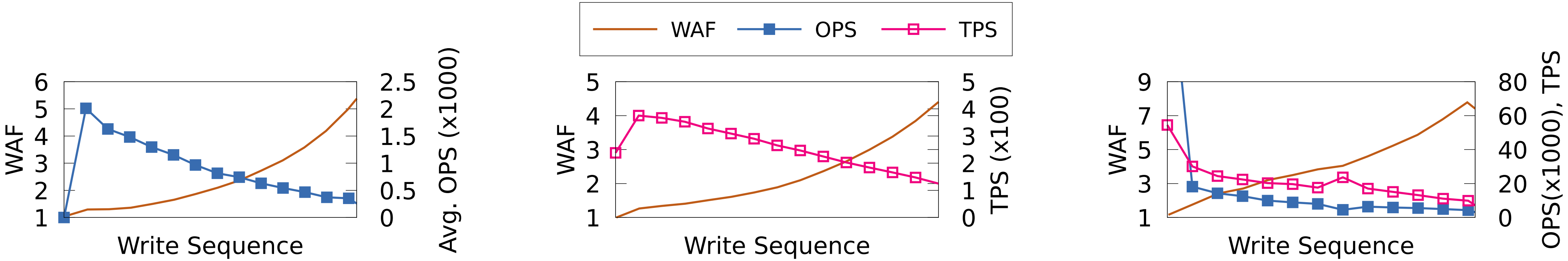
Multiplexed

Clustered by Objects

The Myth of Flash-Friendly Writes

- Flash-friendly sequential writes are **no less harmful** than random writes in terms of write amplification
  - Split into smaller write requests due to **file system fragmentation** and **kernel IO scheduling**
  - Pages from multiple SSTables **with distinct deathtime** tend to be stored together in the same flash blocks.

- Object-oblivious and stream-writes-by-time policy

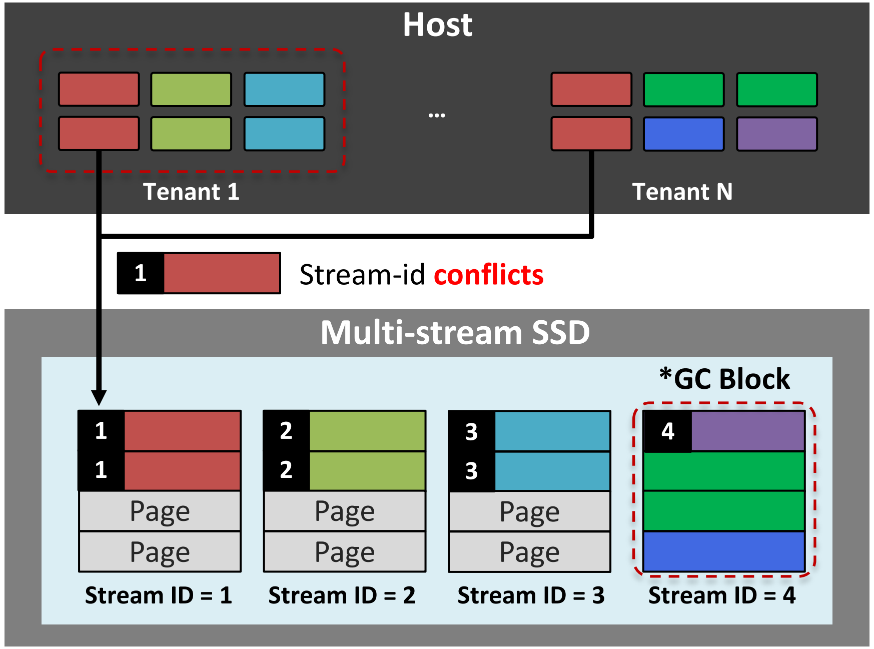


(a) 4 db_bench Tenants (RocksDB)

(b) TPC-C (MySQL)

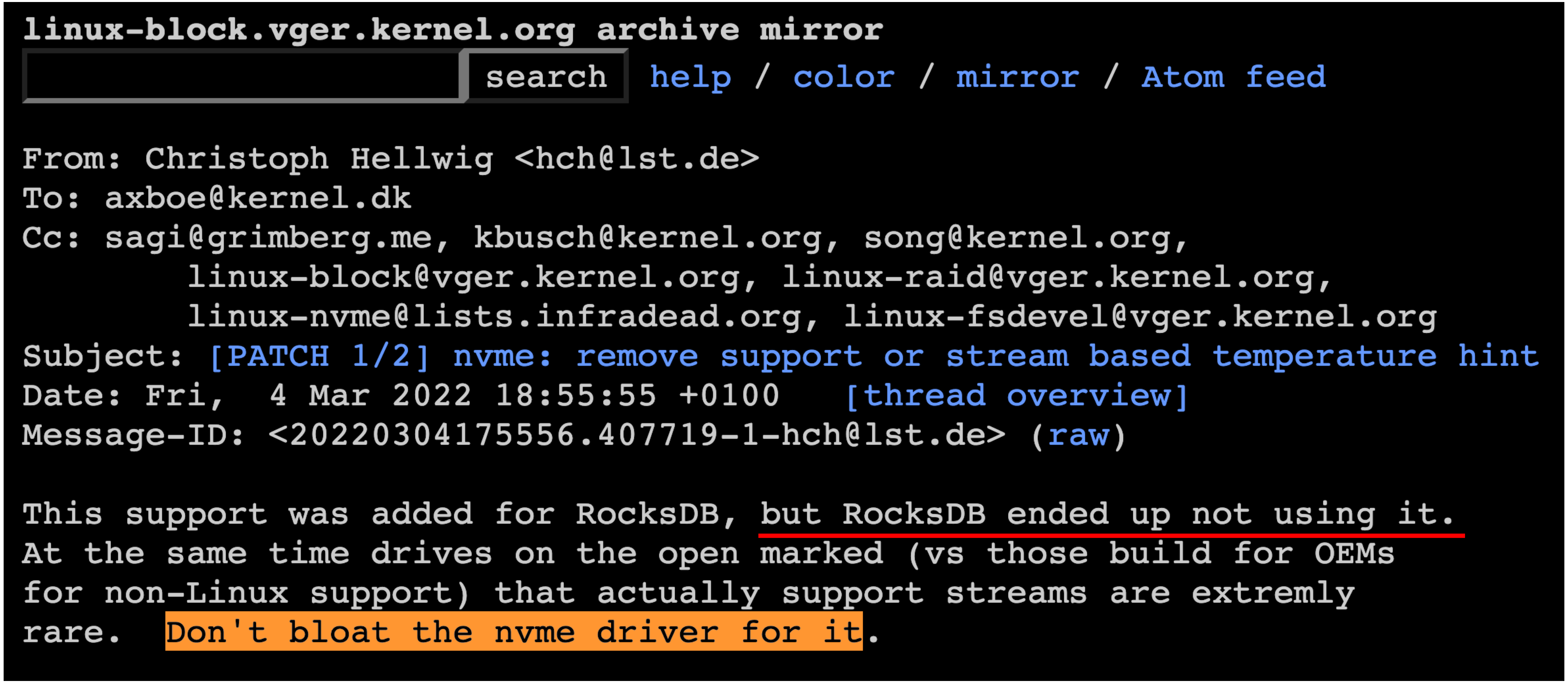
(c) Multi-tenants DB (db_bench+TPC-C)

How about the Object-aware SSD? – MS-SSD and ZNS



- Static binding of limited stream-id
- Stream-id conflicts in multi-tenant
- No stream-aware GC
- Suffer from write amplification

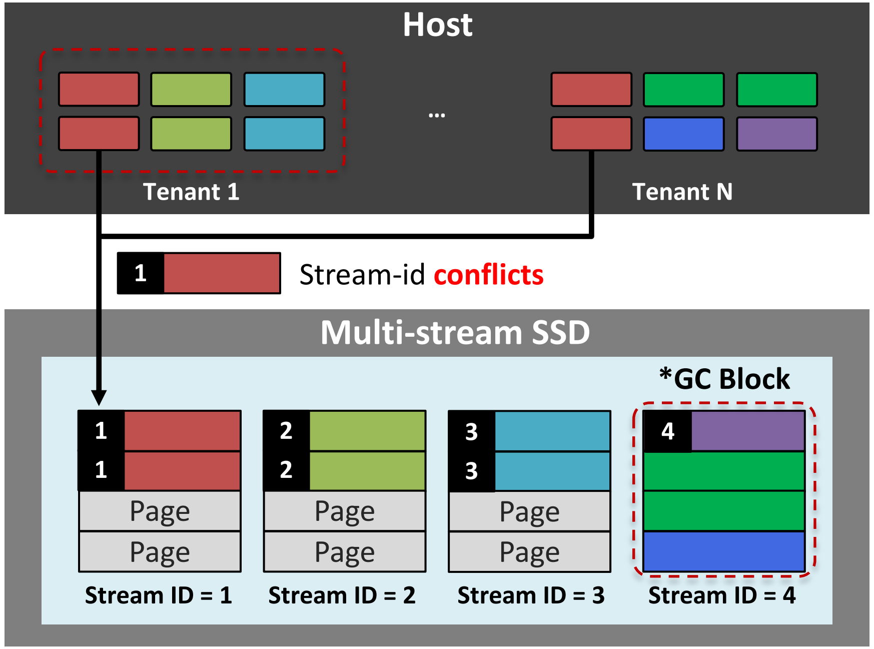
Farewell, Multi-Stream SSD

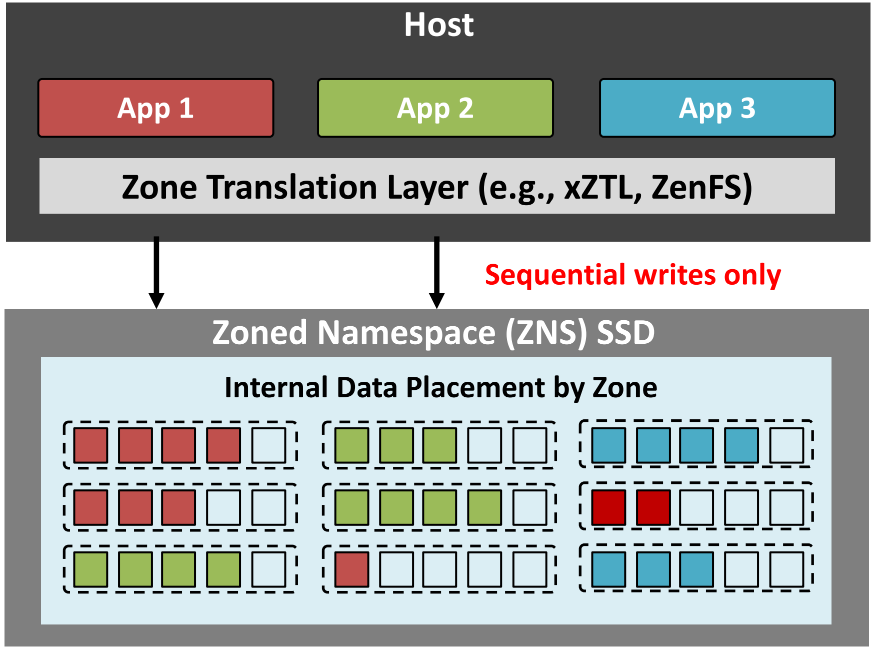


```
linux-block.vger.kernel.org archive mirror
                         search  help / color / mirror / Atom feed

From: Christoph Hellwig <hch@lst.de>
To: axboe@kernel.dk
Cc: sagi@grimberg.me, kbusch@kernel.org, song@kernel.org,
        linux-block@vger.kernel.org, linux-raid@vger.kernel.org,
        linux-nvme@lists.infradead.org, linux-fsdevel@vger.kernel.org
Subject: [PATCH 1/2] nvme: remove support or stream based temperature hint
Date: Fri,  4 Mar 2022 18:55:55 +0100   [thread overview]
Message-ID: <20220304175556.407719-1-hch@lst.de> (raw)

This support was added for RocksDB, but RocksDB ended up not using it.
At the same time drives on the open marked (vs those build for OEMs
for non-Linux support) that actually support streams are extremly
rare.  Don't bloat the nvme driver for it.
```

https://lore.kernel.org/linux-block/20220304175556.407719-1-hch@lst.de/

How about the Object-aware SSD? – MS-SSD and ZNS




Host
App 1
App 2
App 3
Zone Translation Layer (e.g., xZTL, ZenFS)
Sequential writes only
Zoned Namespace (ZNS) SSD
Internal Data Placement by Zone

- Static binding of limited stream-id
- Stream-id conflicts in multi-tenant
- No stream-aware GC
- Suffer from write amplification

- Strict write-ordering rule
- Non-transparent write streaming
- Yet-more expensive tax for log-structured writes

FlashAlloc

- Enlighten flash device to stream writes by objects
  - Offload the host semantic about object's **LBA ranges** to the storage
  - De-multiplex concurrent writes from multiple objects with **distinct deathtimes** into **per-object dedicated blocks**

- Clusters data from the same object into same flash blocks
  - Logically fragmented → Physically de-fragmented into same flash block

- Enables **per-object fine-grained** write streaming

- Minimal changes on applications and FTL
  - No need for additional translation layer or mapping information in **host-side**

FlashAlloc: Interface

🙂

- Logical objects with distinct deathtime
- Sequentially append and cyclically reused

🙁

- Small and random overwrites
- Tiny object
- Append-only writes of unknown size

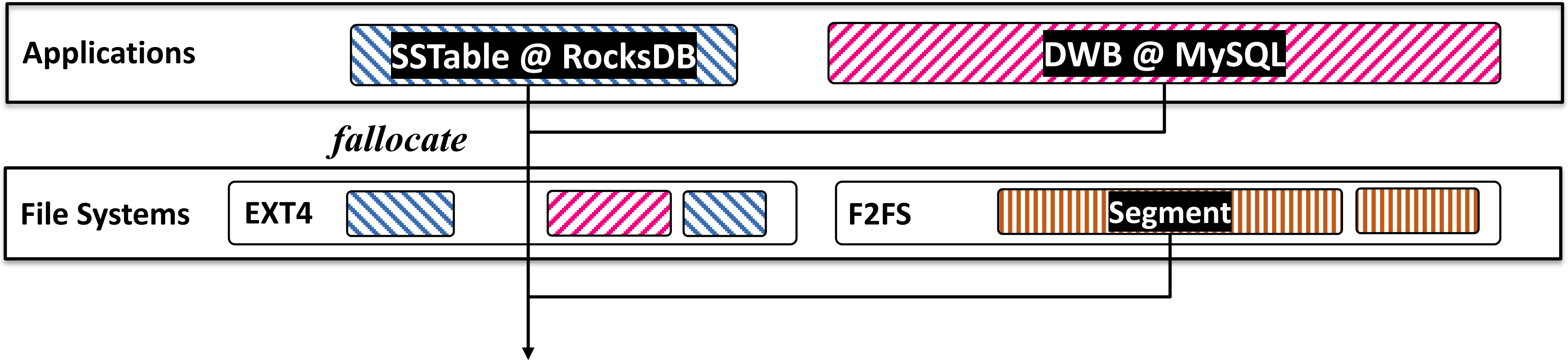
FlashAlloc: Interface

🙂

- Logical objects with distinct deathtime
- Sequentially append and cyclically reused

☹

- Small and random overwrites
- Tiny object
- Append-only writes of unknown size



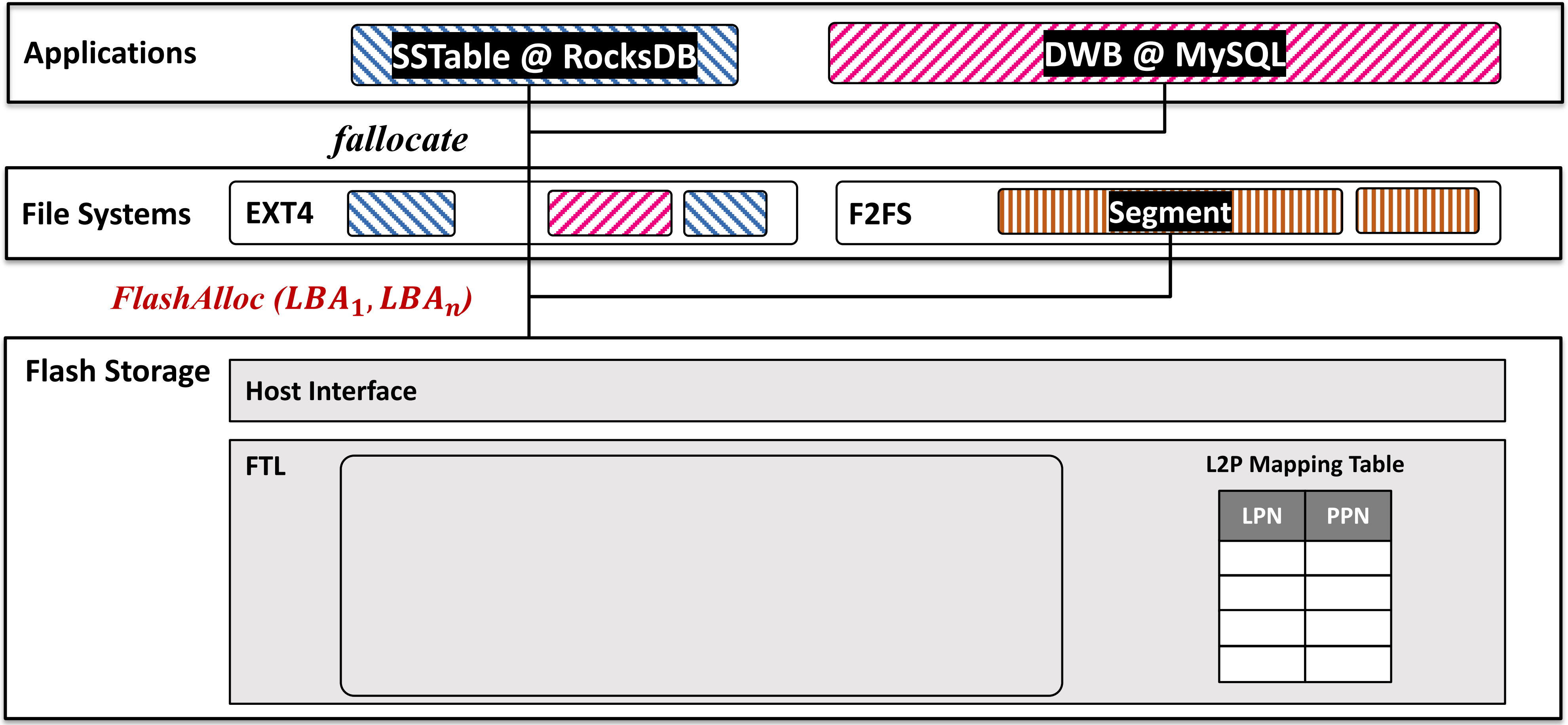
FlashAlloc: Interface

🙂

- Logical objects with distinct deathtime
- Sequentially append and cyclically reused

☹

- Small and random overwrites
- Tiny object
- Append-only writes of unknown size



LPN	PPN

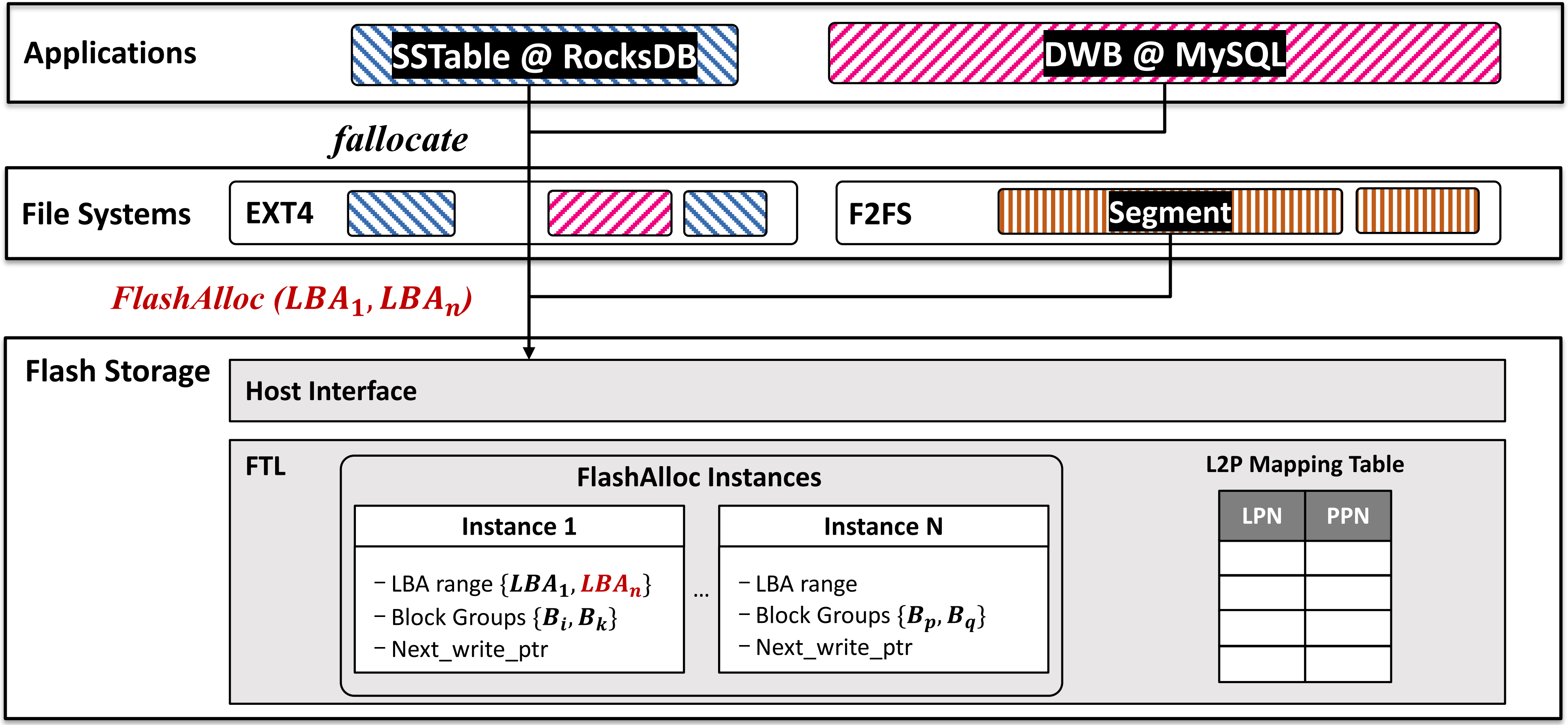
FlashAlloc: Interface

☺

- Logical objects with distinct deathtime
- Sequentially append and cyclically reused

☹

- Small and random overwrites
- Tiny object
- Append-only writes of unknown size



Applications

SSTable @ RocksDB

DWB @ MySQL

fallocate

File Systems

EXT4

F2FS

Segment

$FlashAlloc\ (LBA_1, LBA_n)$

Flash Storage

Host Interface

FTL

FlashAlloc Instances

Instance 1

- LBA range $\{LBA_1, LBA_n\}$
- Block Groups $\{B_i, B_k\}$
- Next_write_ptr

...

Instance N

- LBA range
- Block Groups $\{B_p, B_q\}$
- Next_write_ptr

L2P Mapping Table

LPN	PPN

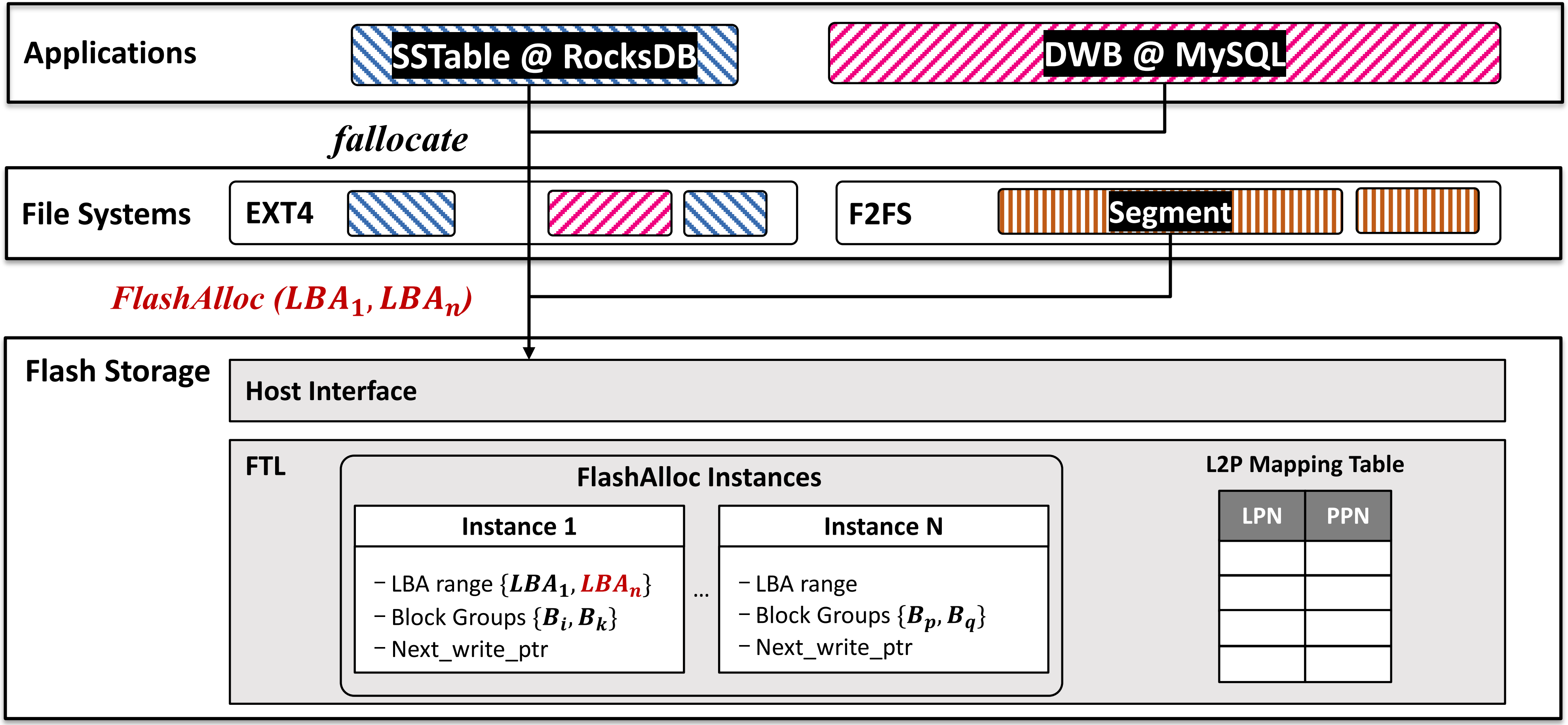
FlashAlloc: Interface

☺

- Logical objects with distinct deathtime
- Sequentially append and cyclically reused

☹

- Small and random overwrites
- Tiny object
- Append-only writes of unknown size



Applications

SSTable @ RocksDB

DWB @ MySQL

fallocate

File Systems

EXT4

F2FS

Segment

FlashAlloc (LBA_1, LBA_n)

Flash Storage

Host Interface

FTL

FlashAlloc Instances

Instance 1

- LBA range $\{LBA_1, LBA_n\}$
- Block Groups $\{B_i, B_k\}$
- Next_write_ptr

…

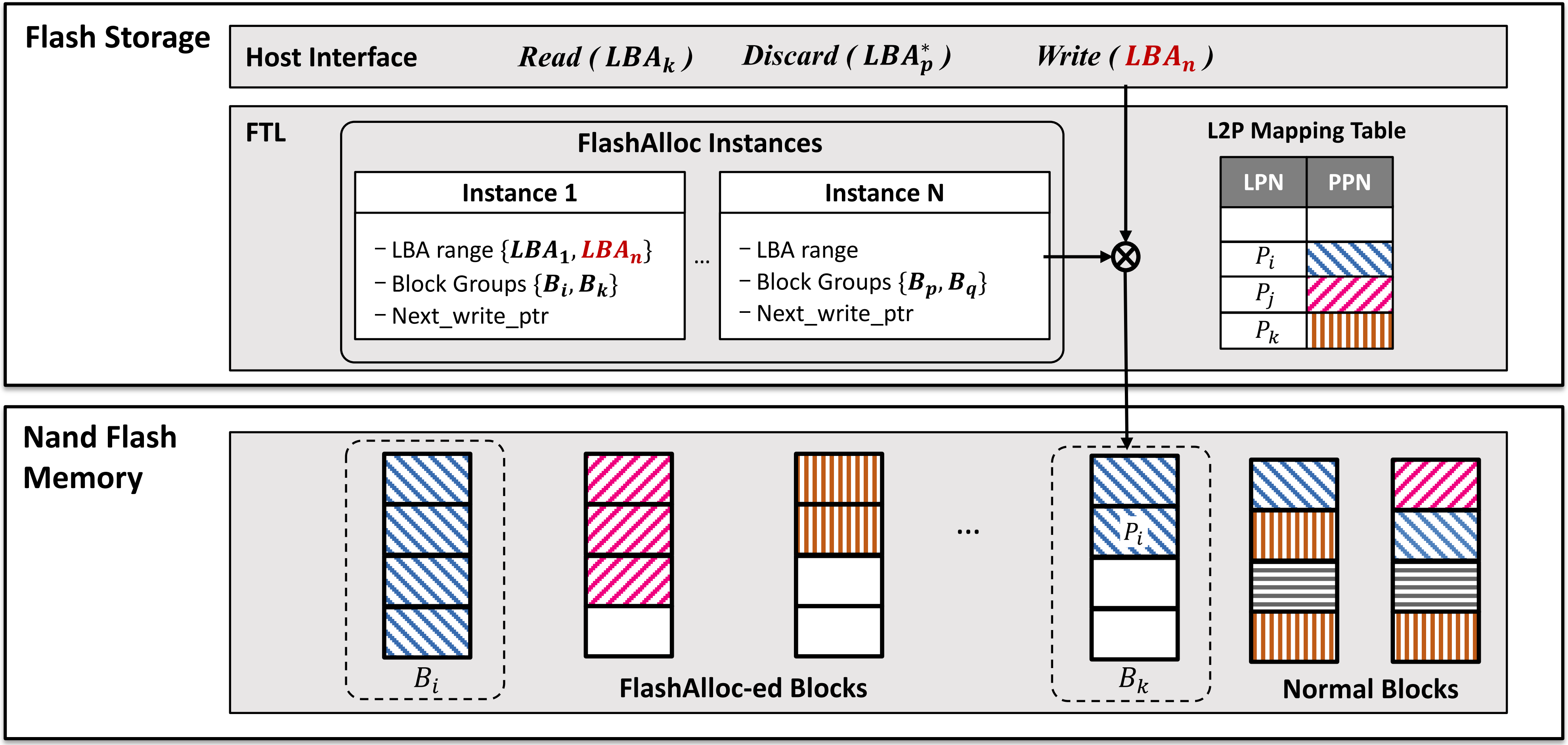
Instance N

- LBA range
- Block Groups $\{B_p, B_q\}$
- Next_write_ptr

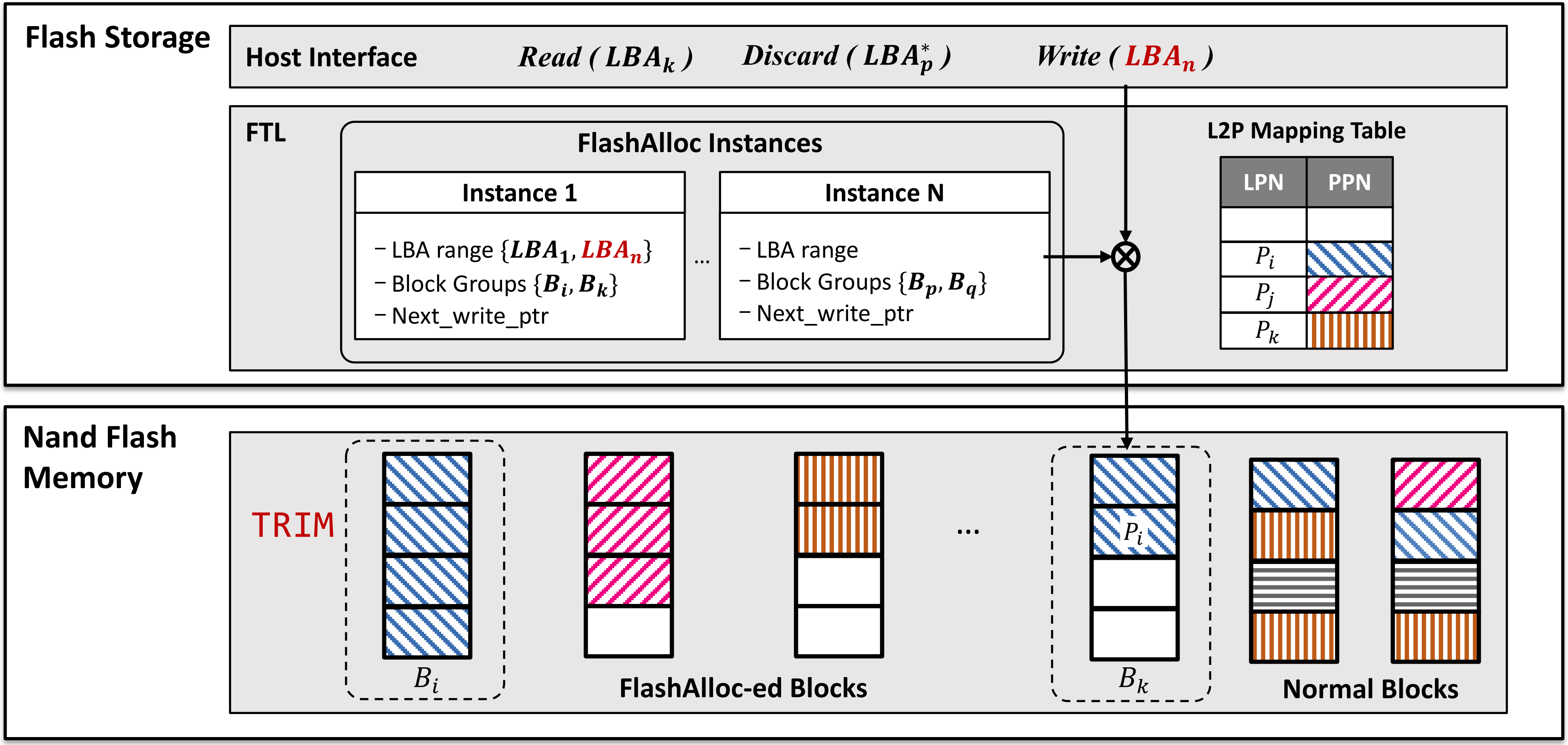
L2P Mapping Table

LPN	PPN

FlashAlloc: Core operations

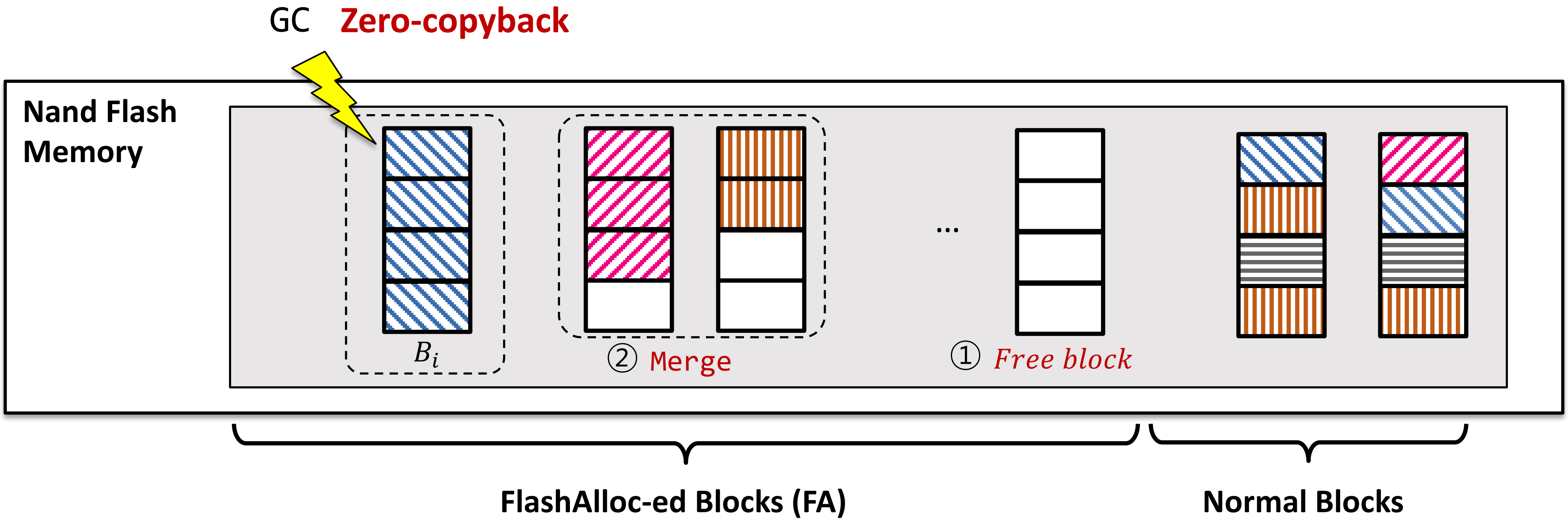


FlashAlloc: Core operations



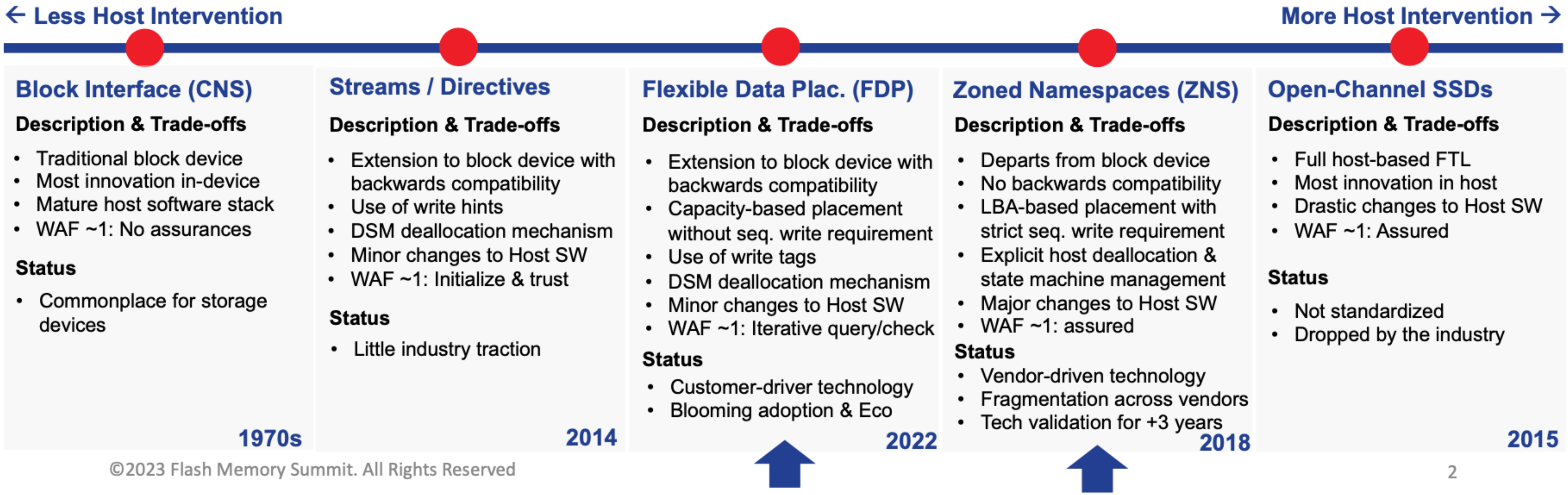
FlashAlloc: Garbage Collection

- Pages from normal blocks are **not mixed** with those from FA blocks
- GC-by-Block-types
  - New FlashAlloc-ed block must be secured to total-clean block (①, ②)
  - Adaptive space allocation – Depending on victim block type (FA vs. Normal blocks)



FDP : Flexible Data Placement

- Data placement is a prevalent problem across NAND consumer & industries
  - Impact: **WAF**, TCO, Predictability (latencies), and overall performance
- Several approaches in the past few years account for innovation in this area
  - Well explored design space a good understanding of the trade-offs



← Less Host Intervention

More Host Intervention →

Block Interface (CNS)

Description & Trade-offs

- Traditional block device
- Most innovation in in-device
- Mature host software stack
- WAF ~1: No assurances

Status

- Commonplace for storage devices

1970s

Streams / Directives

Description & Trade-offs

- Extension to block device with backwards compatibility
- Use of write hints
- DSM deallocation mechanism
- Minor changes to Host SW
- WAF ~1: Initialize & trust

Status

- Little industry traction

2014

Flexible Data Plac. (FDP)

Description & Trade-offs

- Extension to block device with backwards compatibility
- Capacity-based placement without seq. write requirement
- Use of write tags
- DSM deallocation mechanism
- Minor changes to Host SW
- WAF ~1: Iterative query/check

Status

- Customer-driver technology
- Blooming adoption & Eco

2022

Zoned Namespaces (ZNS)

Description & Trade-offs

- Departs from block device
- No backwards compatibility
- LBA-based placement with strict seq. write requirement
- Explicit host deallocation & state machine management
- Major changes to Host SW
- WAF ~1: assured

Status

- Vendor-driven technology
- Fragmentation across vendors
- Tech validation for +3 years

2018

Open-Channel SSDs

Description & Trade-offs

- Full host-based FTL
- Most innovation in host
- Drastic changes to Host SW
- WAF ~1: Assured

Status

- Not standardized
- Dropped by the industry

2015

2

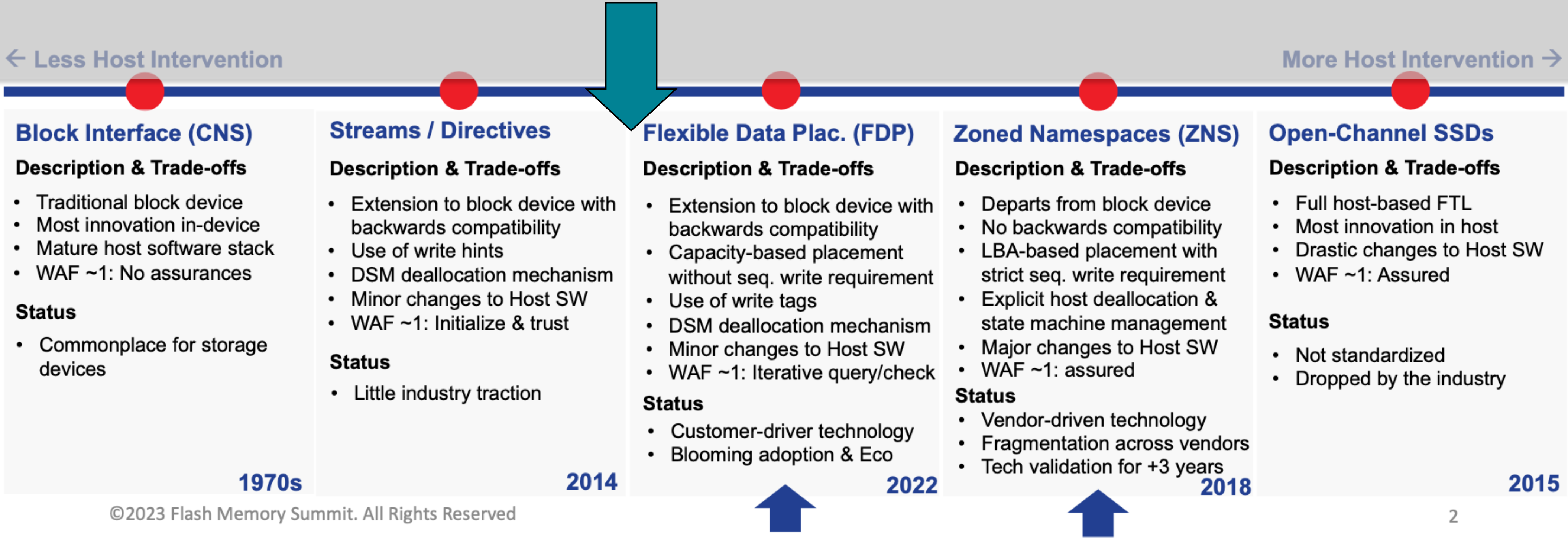
NAND Data Placement Landscape, Trade-Offs, and Direction, Javier González , FMS 2023

FDP : Flexible Data Placement

- Data placement is a prevalent problem across NAND consumer & industries

FlashAlloc

- **Stream writes by object**
- **No additional translation layer in host**



← Less Host Intervention

More Host Intervention →

Block Interface (CNS)

Description & Trade-offs

- Traditional block device
- Most innovation in-device
- Mature host software stack
- WAF ~1: No assurances

Status

- Commonplace for storage devices

1970s

Streams / Directives

Description & Trade-offs

- Extension to block device with backwards compatibility
- Use of write hints
- DSM deallocation mechanism
- Minor changes to Host SW
- WAF ~1: Initialize & trust

Status

- Little industry traction

2014

Flexible Data Plac. (FDP)

Description & Trade-offs

- Extension to block device with backwards compatibility
- Capacity-based placement without seq. write requirement
- Use of write tags
- DSM deallocation mechanism
- Minor changes to Host SW
- WAF ~1: Iterative query/check

Status

- Customer-driver technology
- Blooming adoption & Eco

2022

Zoned Namespaces (ZNS)

Description & Trade-offs

- Departs from block device
- No backwards compatibility
- LBA-based placement with strict seq. write requirement
- Explicit host deallocation & state machine management
- Major changes to Host SW
- WAF ~1: assured

Status

- Vendor-driven technology
- Fragmentation across vendors
- Tech validation for +3 years

2018

Open-Channel SSDs

Description & Trade-offs

- Full host-based FTL
- Most innovation in host
- Drastic changes to Host SW
- WAF ~1: Assured

Status

- Not standardized
- Dropped by the industry

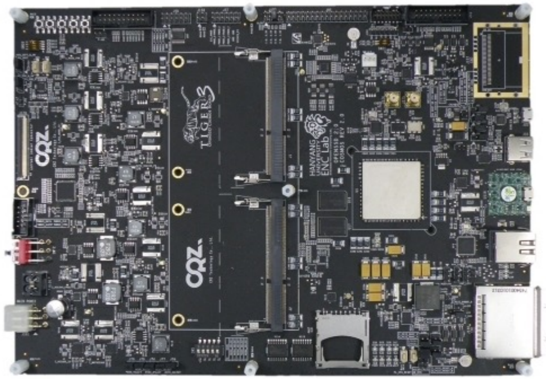
2015

NAND Data Placement Landscape, Trade-Offs, and Direction, Javier González , FMS 2023

Experimental Setup

- System Setup
  - Intel Core i7-6700 CPU 3.40GHz 8 cores
  - 50GB DRAM
  - 256GB Samsung 850 Pro SSD

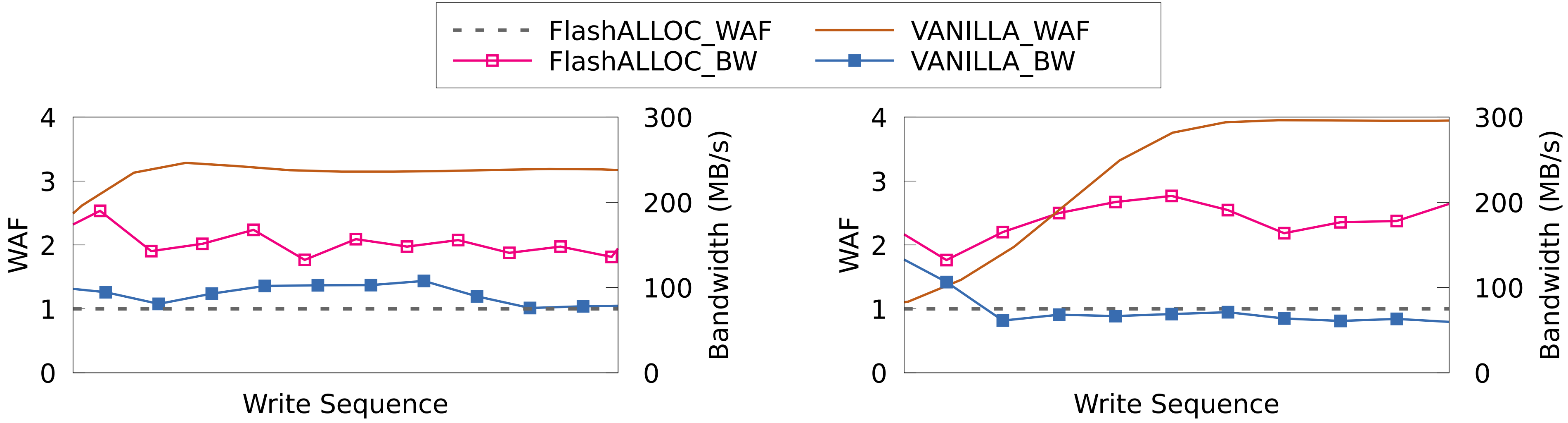
- Cosmos OpenSSD
  - Xilinx Zynq-7000 with dual Core ARM Cortex-A9
  - 256KB SRAM, 1GB DDR3DRAM
  - 16GB MLC Nand flash memory (Over-Provision: 10%)



- Database Setup
  - MySQL : 32 threads + **TPC-C** 80 warehouse (8GB)
  - RocksDB : 4 clients, 64MB SSTables + **db_bench** fillrandom

Evaluation #1. Synthetic FIO workloads

- The considerable gain of the FlashAlloc version is direct reflection of reductions in the **garbage collection overhead**.
- Under more concurrent write threads, a flash block in the Cosmos board will be multiplexed by more files with more deviating lifetimes.

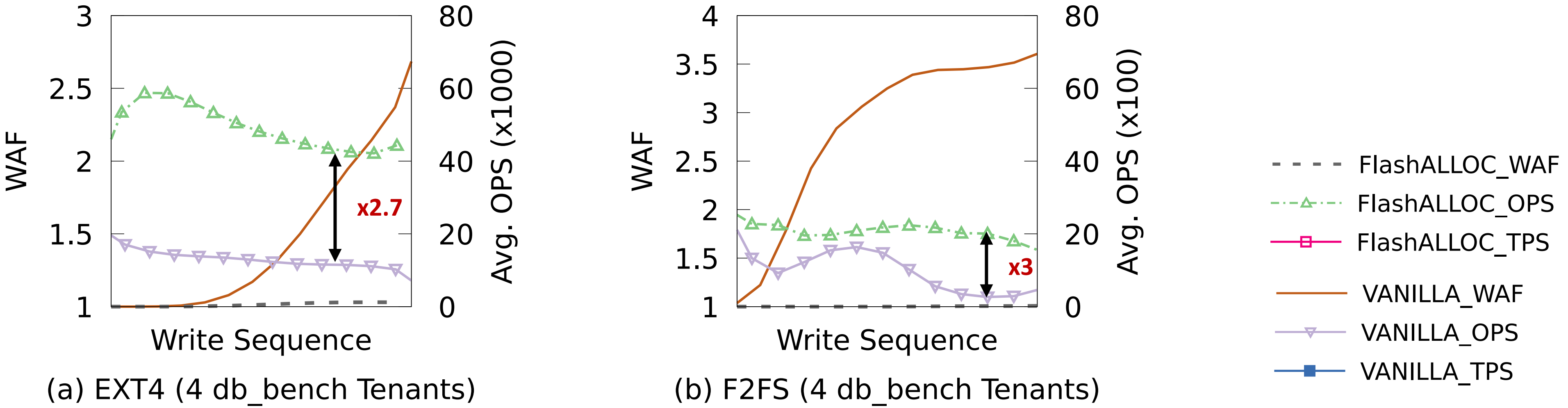


(a) 8 Threads

(b) 32 Threads

Evaluation #2. RocksDB (SSTables) & F2FS (Segments)

- De-multiplexing SSTables/Segments into different flash blocks
  - Enables RocksDB/F2FS to achieve near ideal WAF (i.e., 1)
- FlashAlloc can be fundamental solution for *log-on-log* problem

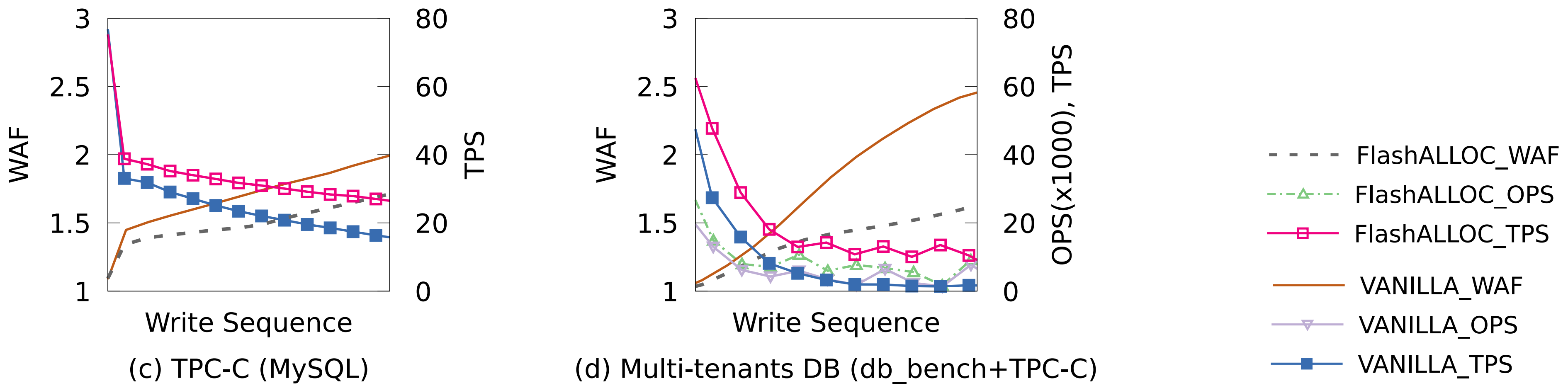


(a) EXT4 (4 db_bench Tenants)

(b) F2FS (4 db_bench Tenants)

Evaluation #3. MySQL (DWB) & Multi-tenancy

- Always to **beneficial** to apply FlashAlloc to appropriate objects and isolate them to dedicate blocks
  - Separating DWB object with cyclic and sequential writes from main databases (i.e., FA writes 50% and non-FA writes 50% case)
- For multi-tenancy (RocksDB+MySQL), FlashAlloc makes tenants **altruistic** to neighbor tenants

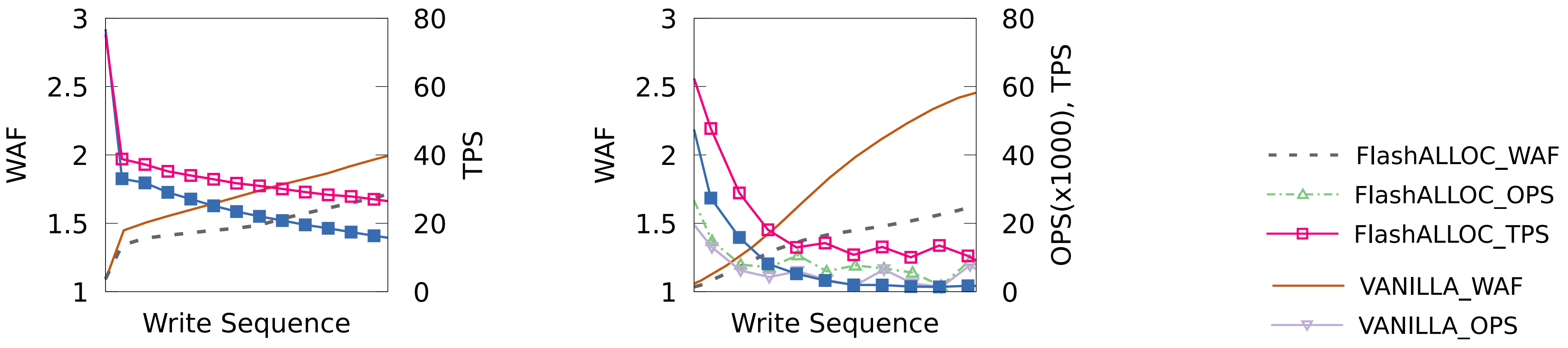


(c) TPC-C (MySQL)

(d) Multi-tenants DB (db_bench+TPC-C)

Evaluation #3. MySQL (DWB) & Multi-tenancy

- Always to **beneficial** to apply FlashAlloc to appropriate objects and isolate them to dedicate blocks
  - Separating DWB object with cyclic and sequential writes from main databases (i.e., FA writes 50% and non-FA writes 50% case)
- For multi-tenancy (RocksDB+MySQL), FlashAlloc makes tenants **altruistic** to neighbor tenants

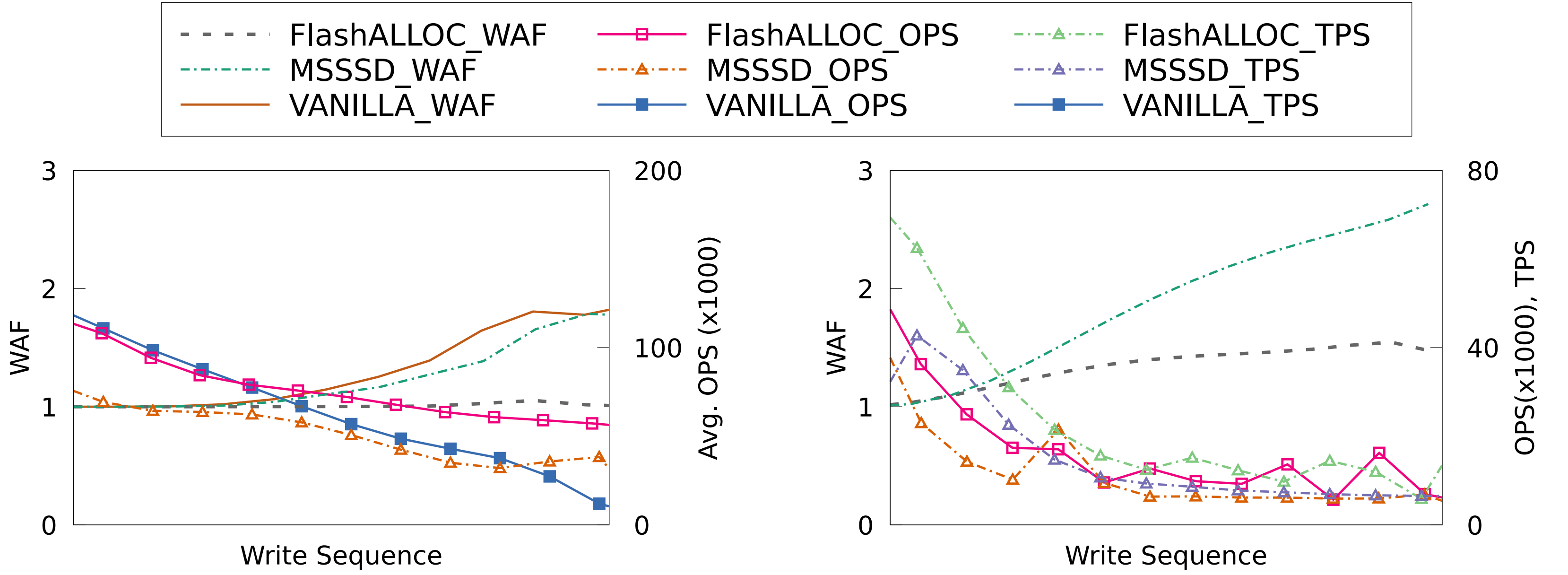


(c) TPC-C (MySQL)

(d) Multi-tenants DB (db_bench+TPC-C)

Evaluation #4. Quantitative Comparison with MS-SSD

- MS-SSD still suffers from write amplification
  - # of physical streams **<<** # of SSTable files
  - **No** stream-aware GC —pages with different lifetime streams mixing in the same flash block

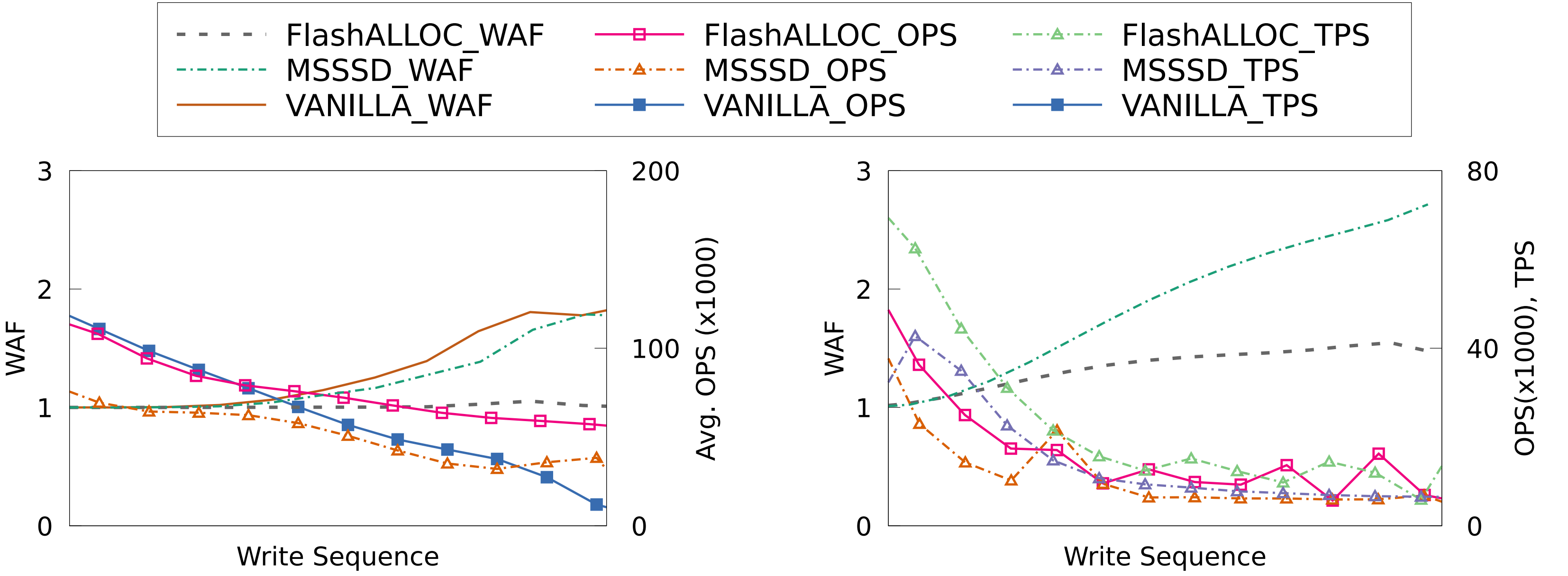


(a) RocksDB (1 db_bench Tenant)

(b) Multi-tenants DB (db_bench+TPC-C)

Evaluation #4. Quantitative Comparison with MS-SSD

- MS-SSD still suffers from write amplification
  - # of physical streams **<<** # of SSTable files
  - **No** stream-aware GC —pages with different lifetime streams mixing in the same flash block

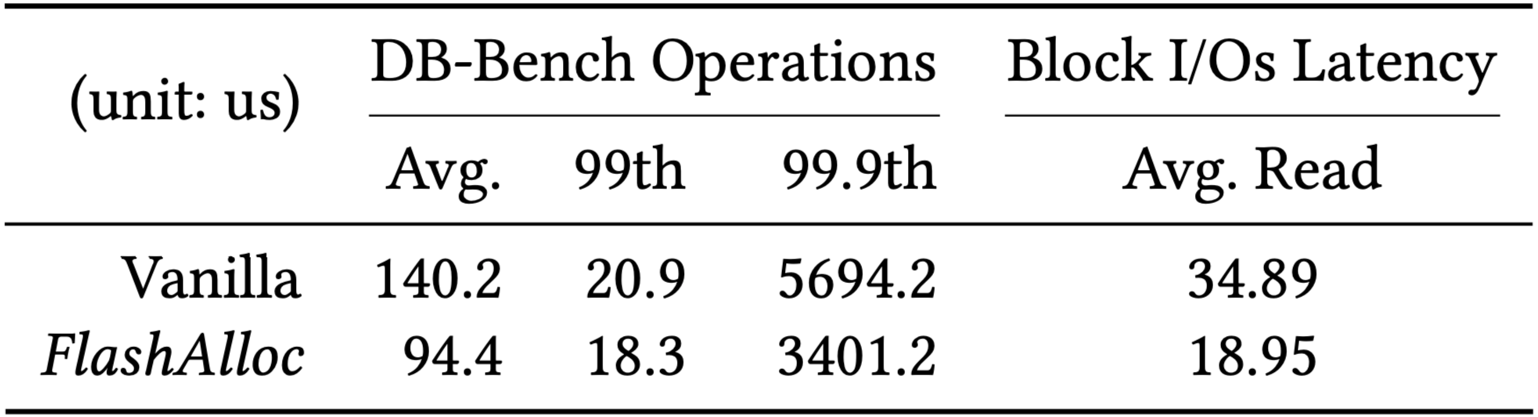


(a) RocksDB (1 db_bench Tenant)

(b) Multi-tenants DB (db_bench+TPC-C)

Evaluation #5. Latency

- Vanilla suffers from high latency spike during GC – victim blocks have pages with different lifetime, resulting in **copyback overhead**
- FlashAlloc reduces latency and narrows latency distribution
  - Eliminates copyback overhead



(unit: us)	DB-Bench Operations			Block I/Os Latency
	Avg.	99th	99.9th	Avg. Read
Vanilla	140.2	20.9	5694.2	34.89
FlashAlloc	94.4	18.3	3401.2	18.95

Summary

- We present **FlashAlloc**, a novel interface, which enables flash devices to ***stream writes by logical objects*** into different physical flash blocks.

- **FlashAlloc** supports ***per-object fine-grained*** write streaming and be the great alternative to existing solution – MS-SSD and ZNS.

- Benefits of **FlashAlloc**
  - Zero-copyback overhead
  - Reduce write amplification overhead
  - Mitigate WAF interference among multiple tenants

Thank you

FlashAlloc: Dedicating Flash Blocks By Objects

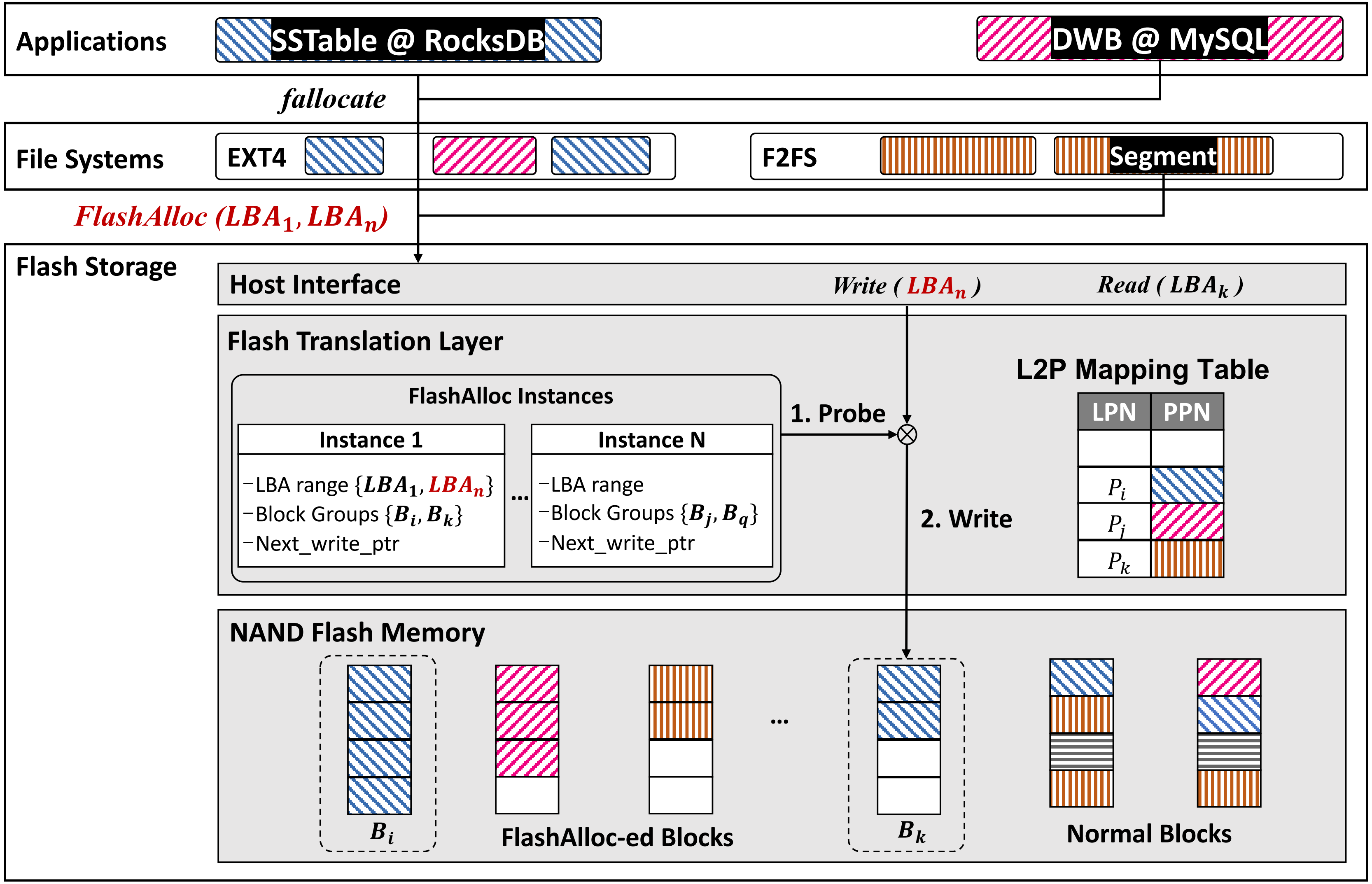
Jonghyeok Park[*], Soyee Choi[**], Gihwan Oh[+], Soojun Im[**],
Moon-Wook Oh[**], Sang-Won Lee[+]

Hankuk University of Foreign Studies[*], Samsung Electronics[**]
Sungkyunkwan University[+]

E-mail: jonghyeok.park@hufs.ac.kr
Github: https://github.com/JonghyeokPark/Flashalloc-Cosmos

Check out more details
in our paper!

FlashAlloc Architecture


Applications
SSTable @ RocksDB
DWB @ MySQL
fallocate
File Systems
EXT4
F2FS
Segment
FlashAlloc (LBA_1, LBA_n)
Flash Storage
Host Interface
Write (LBA_n)
Read (LBA_k)
Flash Translation Layer
FlashAlloc Instances
Instance 1
−LBA range {LBA_1, LBA_n}
−Block Groups {B_i, B_k}
−Next_write_ptr
...
Instance N
−LBA range
−Block Groups {B_j, B_q}
−Next_write_ptr
1. Probe
2. Write
L2P Mapping Table
LPN
PPN
P_i
P_j
P_k
NAND Flash Memory
B_i
FlashAlloc-ed Blocks
...
B_k
Normal Blocks