io.stream: cgroup-Granular FDP for Unmodified Workloads

Suho Son, Jinheon Kim, Kiyoung Lee

Memory Division,

Samsung Electronics Co., Ltd.

HotStorage 2026

The 18th ACM Workshop on Hot Topics in Storage and File Systems

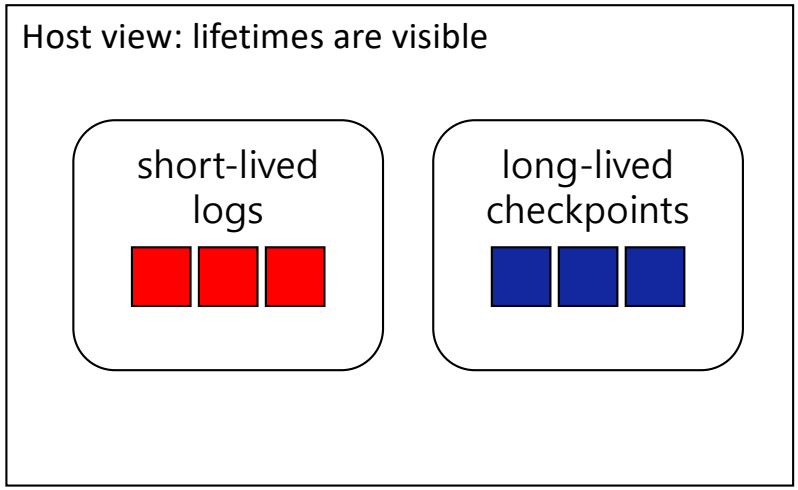
Sept. 28-29 Prague, Czechia

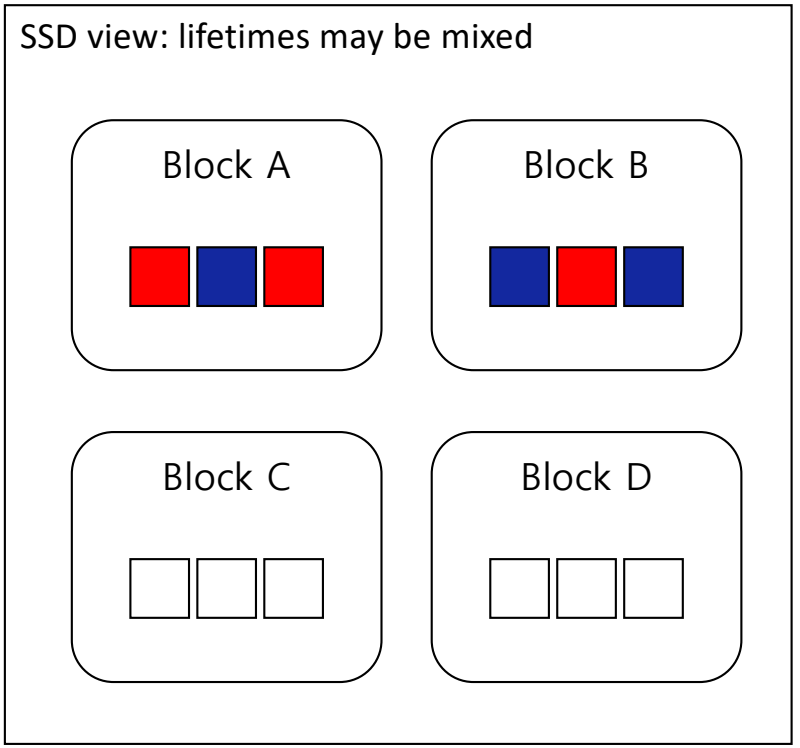
Co-located with SOSP 2026

SAMSUNG

Lifetime mixing creates write amplification

Host knows lifetime · Device decides placement

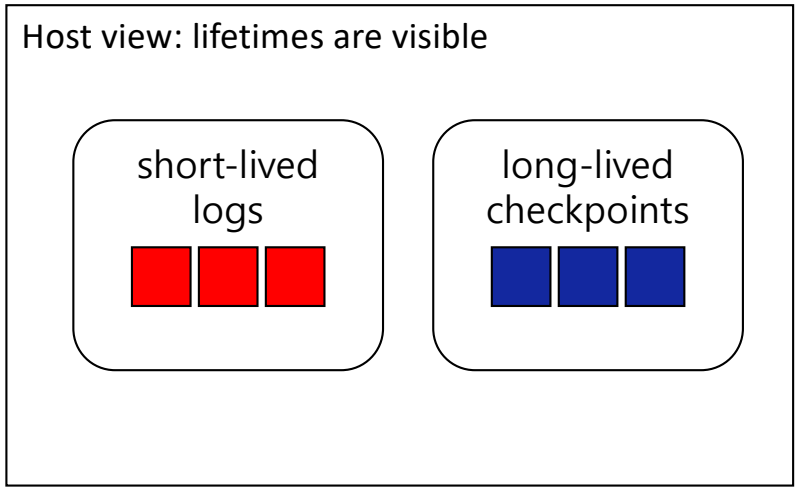


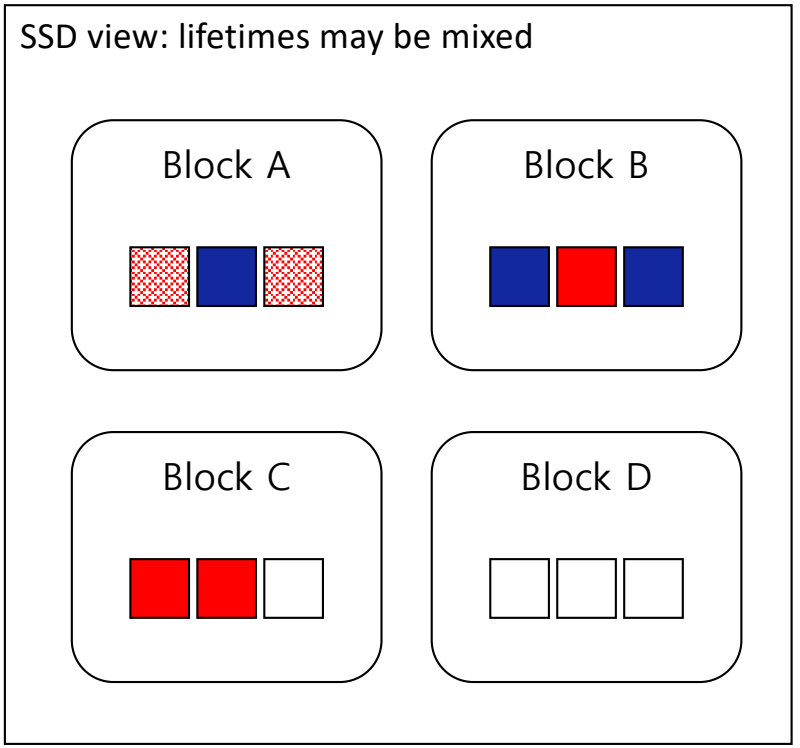


SAMSUNG

Lifetime mixing creates write amplification

Host knows lifetime · Device decides placement

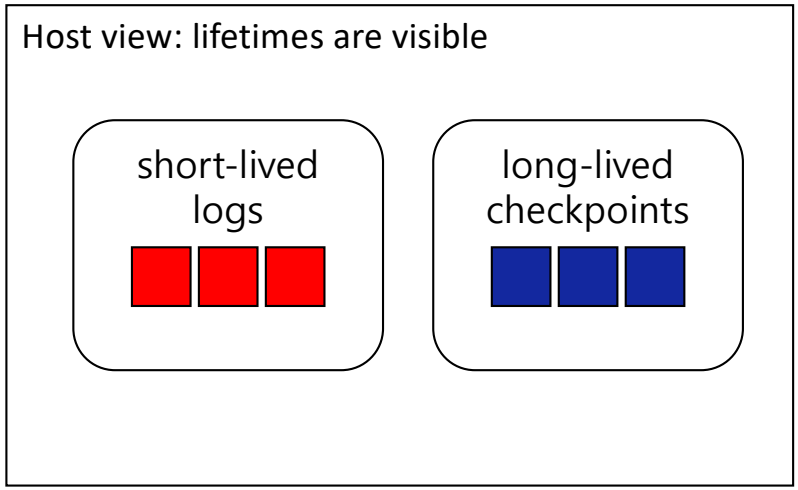


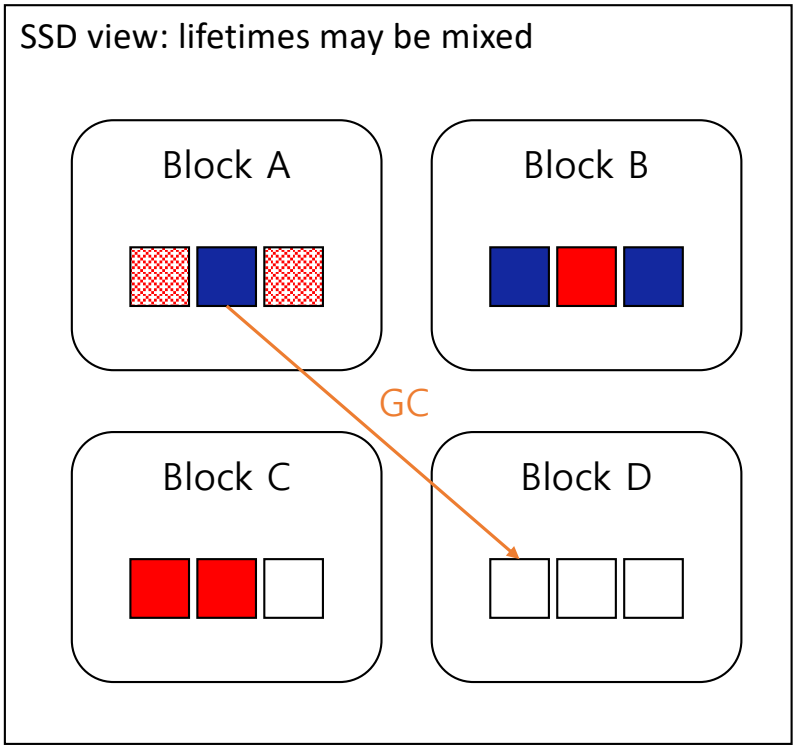


SAMSUNG

Lifetime mixing creates write amplification

Host knows lifetime · Device decides placement

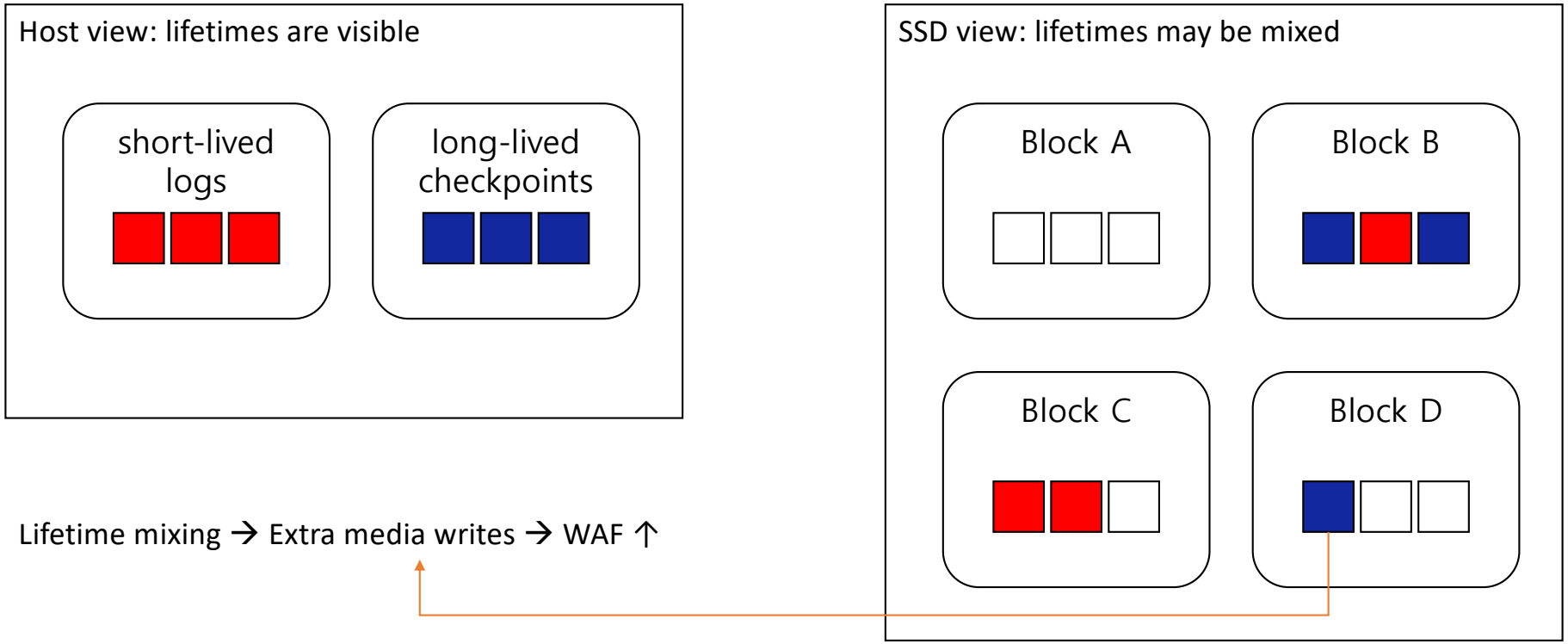




SAMSUNG

Lifetime mixing creates write amplification

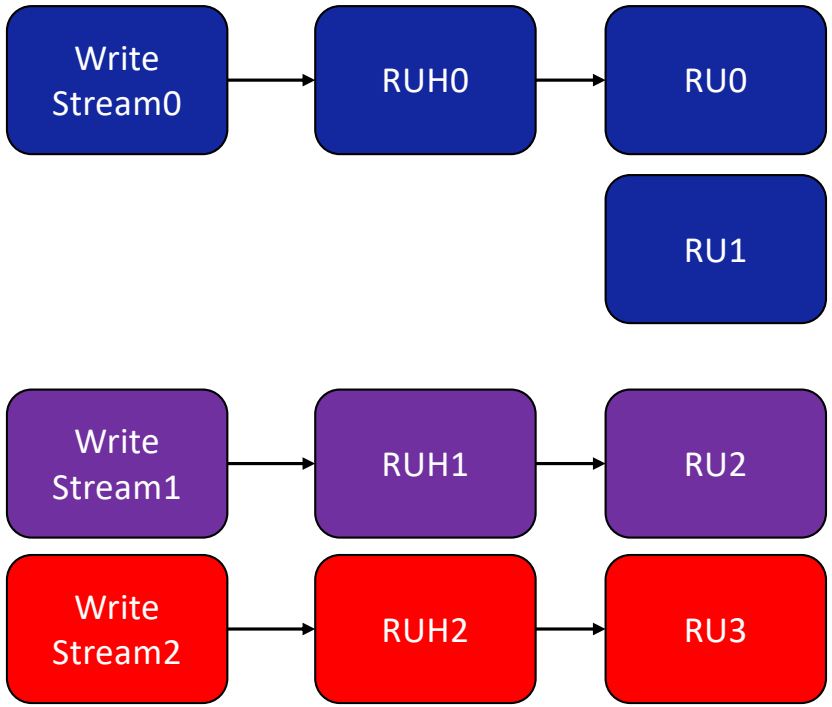
Host knows lifetime · Device decides placement



SAMSUNG

FDP gives the host a placement channel

Same lifetime in one RU → WAF → 1



RU

Reclaim granularity

RUH

A scarce hardware handle

Write stream

The block-layer tag Linux already carries

How to place. Not who decides.

Who should choose the write stream?

Our answer: the operator, through the cgroup

	Accuracy	Controllability	Deployability	Transparency
 **Application** CacheLib · io_uring passthru	**Very High** App semantics	**High** Explicit hints	**Very Low** App changes	**Very Low** App changes
 **File system** FStream · F2FS*	**Medium** Coarse classification	**Medium** FS policy	**Low** FS integration	**Medium** FS dependent
 **Kernel inference** AutoStream · PCStream	**Medium** Prediction errors	**Low** Automatic decisions	**Very High** Automatic	**Very High** Fully transparent
 **Operator · cgroup** io.stream (this work)	**Medium** Tenant-level	**Very High** Explicit policy	**High** Simple config	**Very High** Unmodified workloads

* Inho Song, Shoaib Asif Qazi, Javier Gonzalez, Matias Bjørling, Sam H. Noh, and Huaicheng Li. 2026. Characterizing and Emulating FDP SSDs with WARP. In 24th USENIX Conference on File and Storage Technologies (FAST 26). USENIX Association, Santa Clara, CA, 347–362. https://www.usenix.org/conference/fast26/presentation/song

SAMSUNG

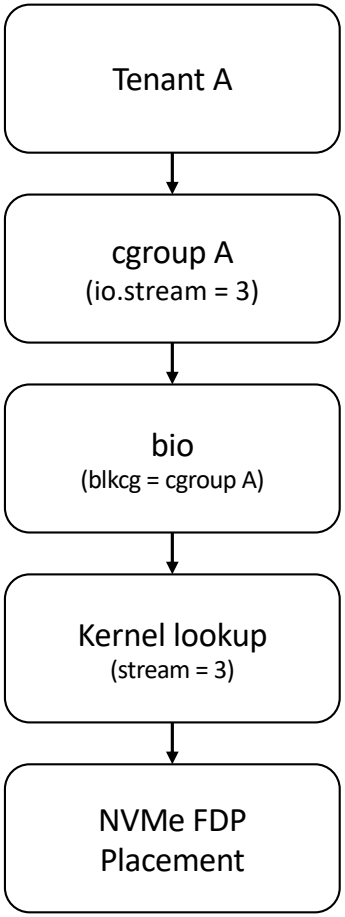
The producer is already on the write path

cgroup = the deployment unit

blkcg rides every data-write bio

— including buffered writeback

io.stream stamps writes from that cgroup



No inference needed.

Interface and precedence

```
echo "259:0 1" >
/sys/fs/cgroup/tenant_A/io.stream
```

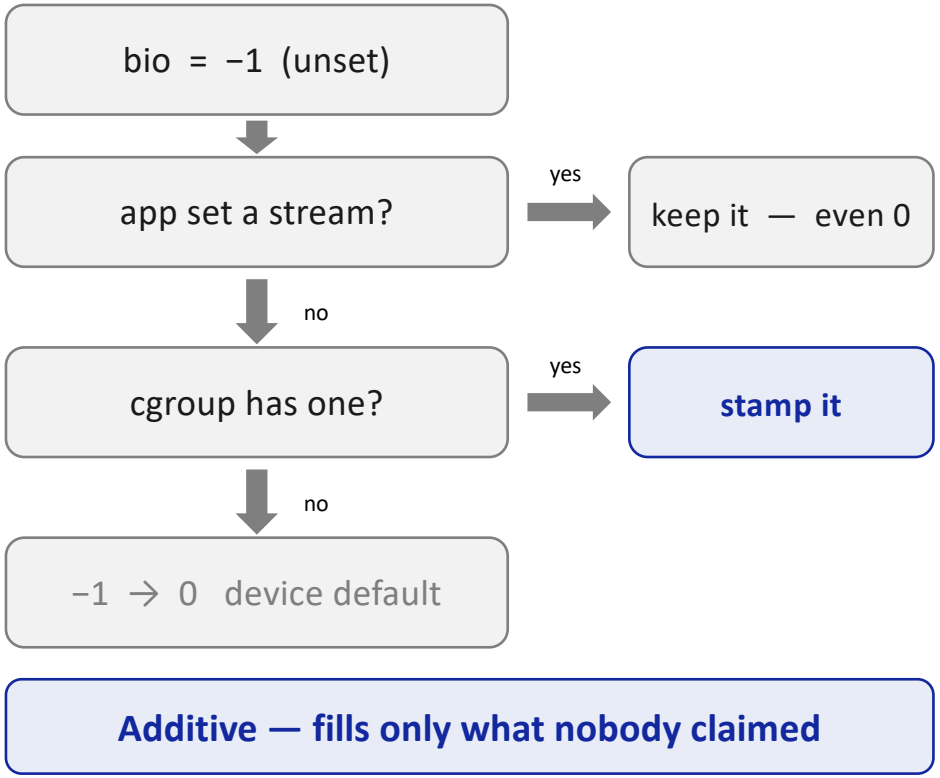
per device · maj:min stream

inherit by default

0 ≠ inherit

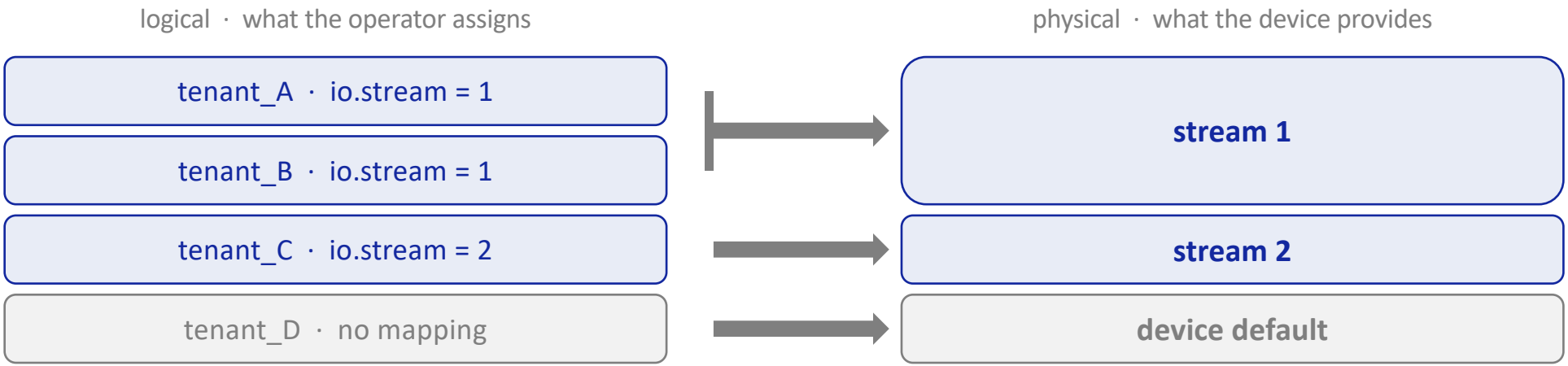
explicit 0 forces the device default

Precedence at bio submission



When cgroups outnumber streams

e.g. 3 RUHs — one is the device default, 2 assignable to tenants · a node runs far more cgroups



pin one · collapse several by lifetime · reserve one for a sensitive tenant

Overflow: fallback, not hashing

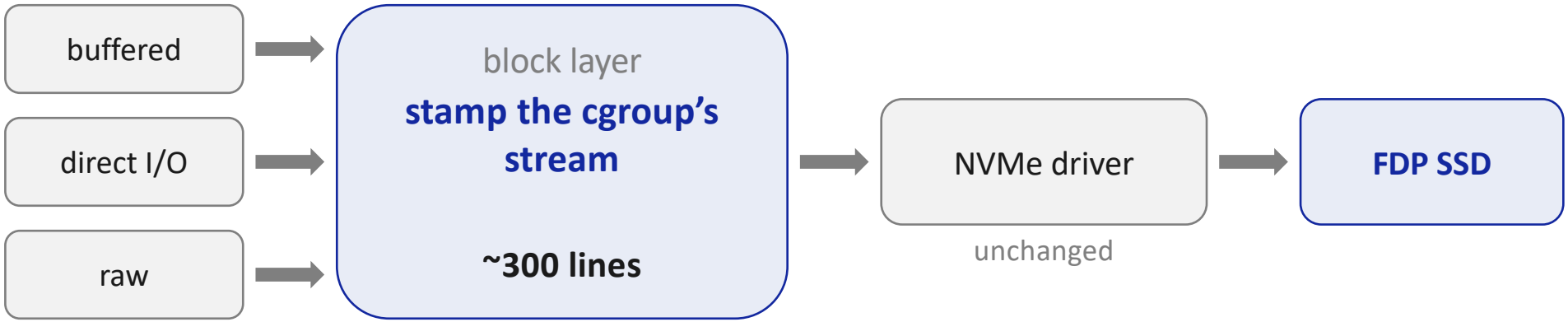
no mapping, or no room left → the device default, never another tenant's stream

Hashing would spread evenly — and seat hot and cold in one reclaim unit. Stage 2 measures that.

SAMSUNG

One stamping point

Every path becomes a bio — we stamp there



< 1%

IOPS and p99

writeback for free

the flusher's bio already carries blkcg

SAMSUNG

Evaluation setup

One device, one firmware, one OP setting — for every arm

PM9D3a · 4 TB
7% OP · 8 RUHs (7 for data)

WAF from the device
SMART + OCP

pre-filled · nodiscard
× 3 runs



Two baselines

No-FDP
FDP off

FDP-default
FDP on, no stream set

= today's unmodified stack

SAMSUNG

Sanity check: the stamp survives every path

Hot 5% and Cold 95%, on distinct streams

raw	ext4	xfs
1.65 → 1.00	**1.53 → 1.00**	**1.61 → 1.00**

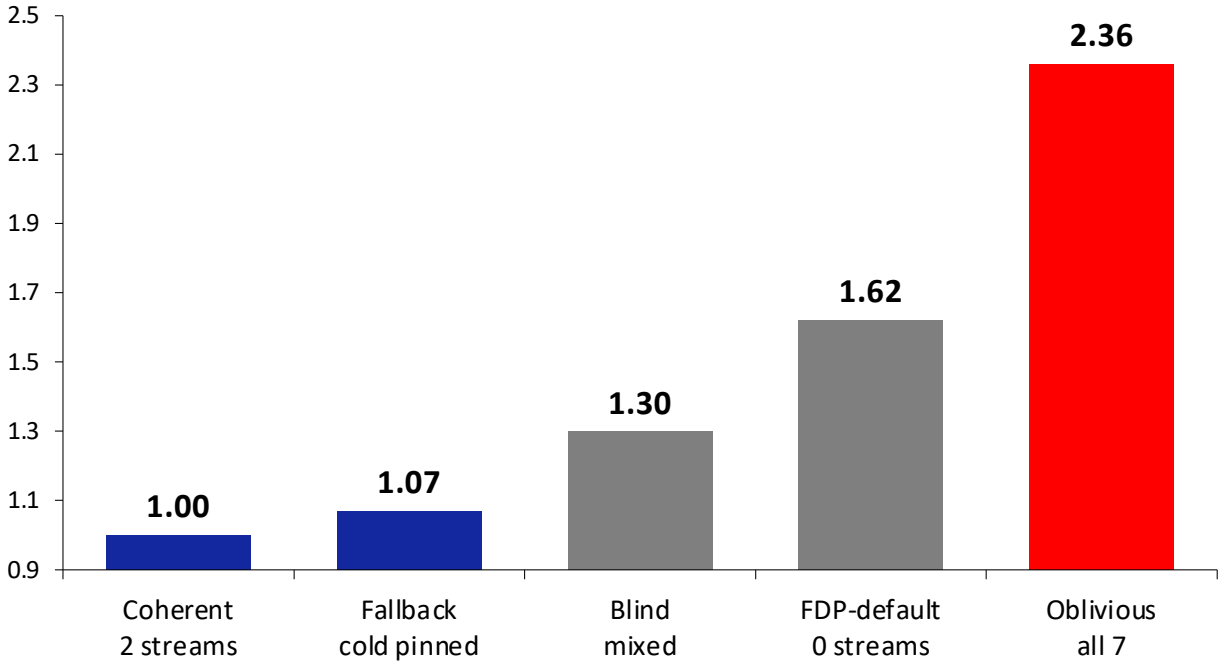
FDP-default → io.stream · controls: No-FDP ≈ FDP-default · Metadata-only = FDP-default

the gain is tenant-data separation — not FDP itself

39% less media · 1.8× throughput on raw · ext4 p99 8.4 → 0.6 ms

Coherence beats stream count

14 cgroups over 7 streams — every policy collides



2 streams

1.00

all 7 streams

2.36

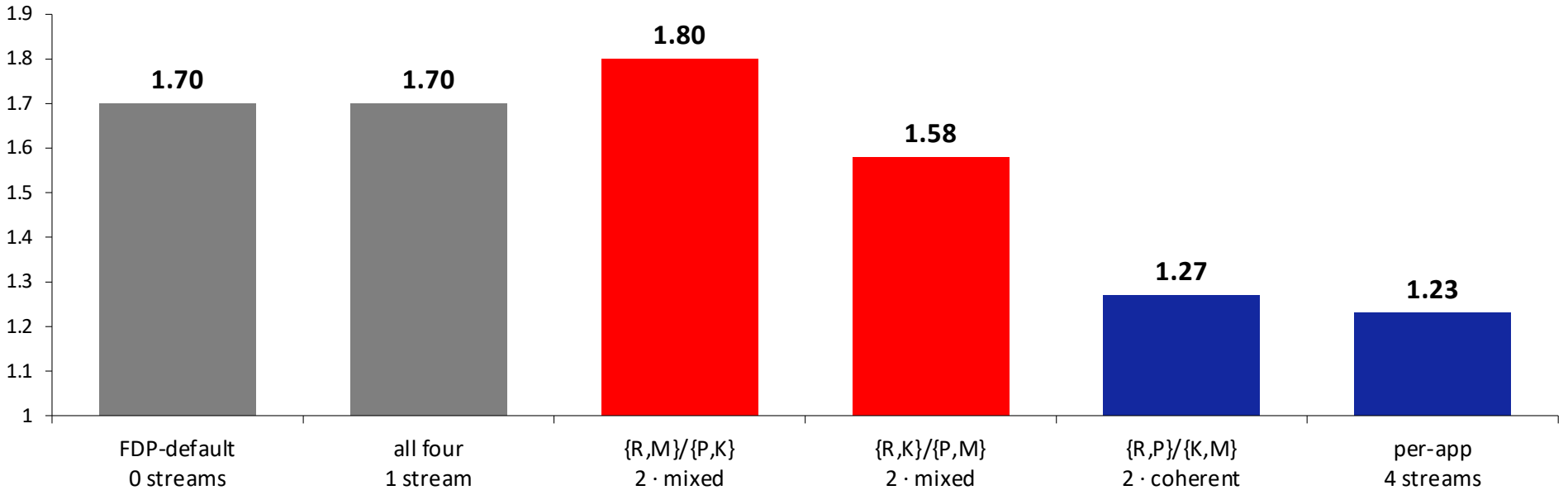
sharing a stream is fine — careless collisions are not

SAMSUNG

Half the streams, the same result

Four unmodified containers, one ext4 volume, 12 h

Hot R RocksDB · P PostgreSQL Cold K Kafka · M MongoDB



same count — only coherence differs

the three 2-stream arms: 1.80 · 1.58 · 1.27

2 coherent ≈ 4 per-app

1.27 on two streams · 1.23 on four

SAMSUNG

Discussion

Macro × Micro



Additive — via the unset sentinel

One namespace — must be coordinated

Provisioning: a second knob



8 units provisioned · Coherent uses 2

Fewer streams, more OP?

but all Stage 2 arms provision 8, so the spread is mapping

Who supplies coherence?

?

Today: by hand

Writes already carry their cgroup

Left: estimate lifetime

Refuse to guess

Conclusion

Not the number of streams —

The lifetime coherence of the mapping

Built

One cgroup v2 file
~300 lines
no app change

Works

The cgroup label is
already on every
data-write bio

Open

Adaptive
cgroup-based
stream mapping

Thank you!

SAMSUNG

Backup — configuration

Stages 1–2 · fio

4 KB sequential · libaio · QD 256

2 h per run · 128 KB preconditioning

direct on raw · buffered on FS

lifetime = region size

hot rewrites small · cold sweeps large

Stage 3 · stock images

RocksDB · db_bench fillrandom

PostgreSQL · pgbench

Kafka · producer-perf-test

MongoDB · bulk insertMany

jbd2 kept continuously active

Backup — metadata (jbd2)

Journal · jbd2

runs in the root cgroup

no tenant → stream 0

tenants on non-zero streams

separated for free

no file system change

Effect is modest

Metadata-only = FDP-default

ext4 1.53 vs 1.53

xfs 1.61 vs 1.61

gains come from tenant data

FS-specific · XFS CIL differs