SAMSUNG

FDP Integration in CacheLib

Arun George

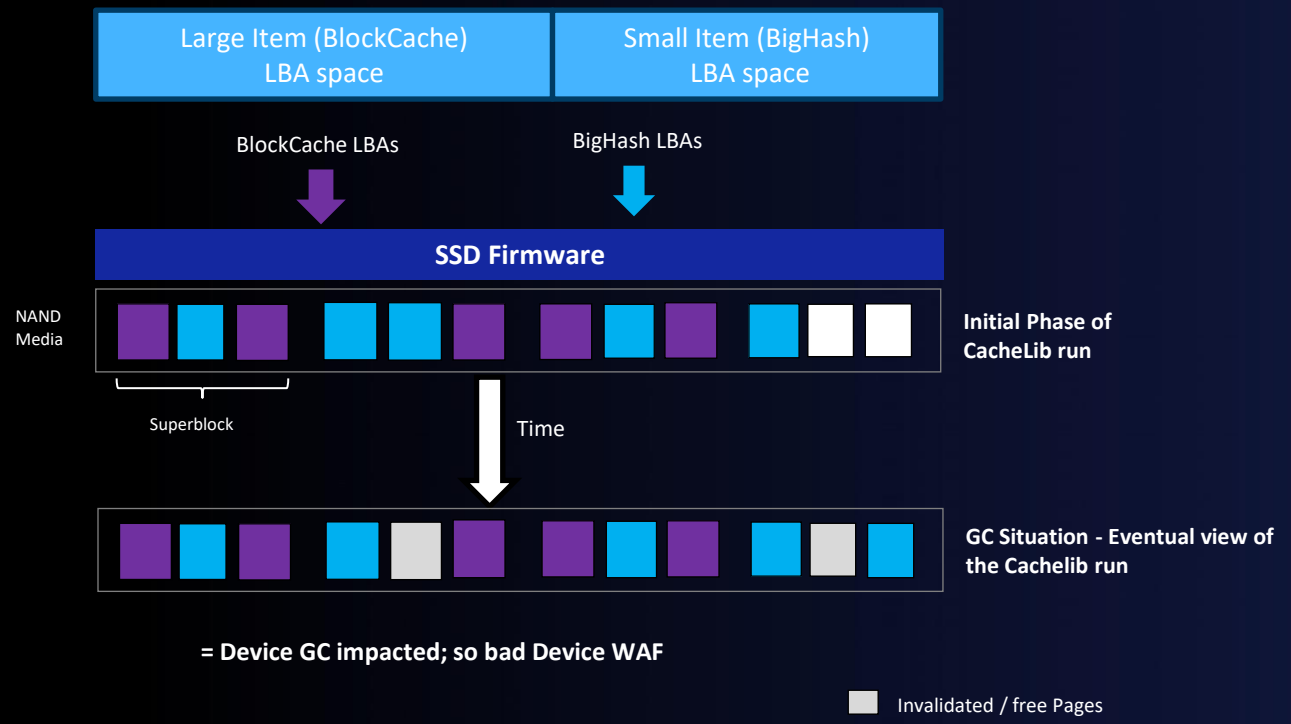
SAMSUNG

Problem Statement – WAF and Overprovisioning

- SSD Device WAF is a challenge in CacheLib deployments
  - SSD WAF can be 3.5 and above
  - Impacts device endurance, latency, power consumption etc.
- Up-to 50% host overprovision(OP) to limit the device WAF
  - This means limiting the SSD utilization to 50%
- This results in inefficient capacity utilization of SSD

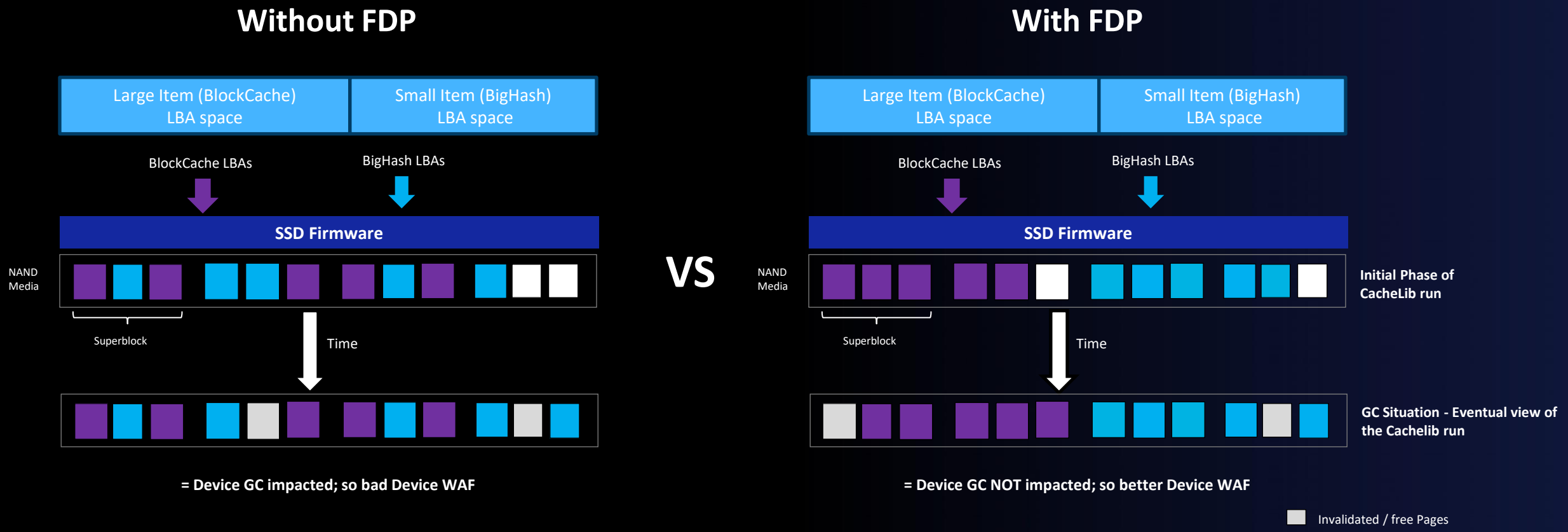
Root Cause – Intermixing of Hot and Cold IO streams

- Two IO engines produces different IO Patterns
- BlockCache(Large Objects) produces relatively cold data pattern
  - Sequential and non-frequent 16 MB sized updates
  - SSD friendly, low WAF pattern
- BigHash(Small Objects) produces hot data pattern
  - Random and frequent 4KB sized updates
  - SSD unfriendly, high WAF pattern
- Intermixing of IO patterns at NAND blocks
  - Inefficient device GC, and hence high WAF



FDP in CacheLib - The Opportunity

- FDP can help to segregate the two data IO patterns at NAND level
- WAF due to intermixing of BlockCache and BigHash data is avoided



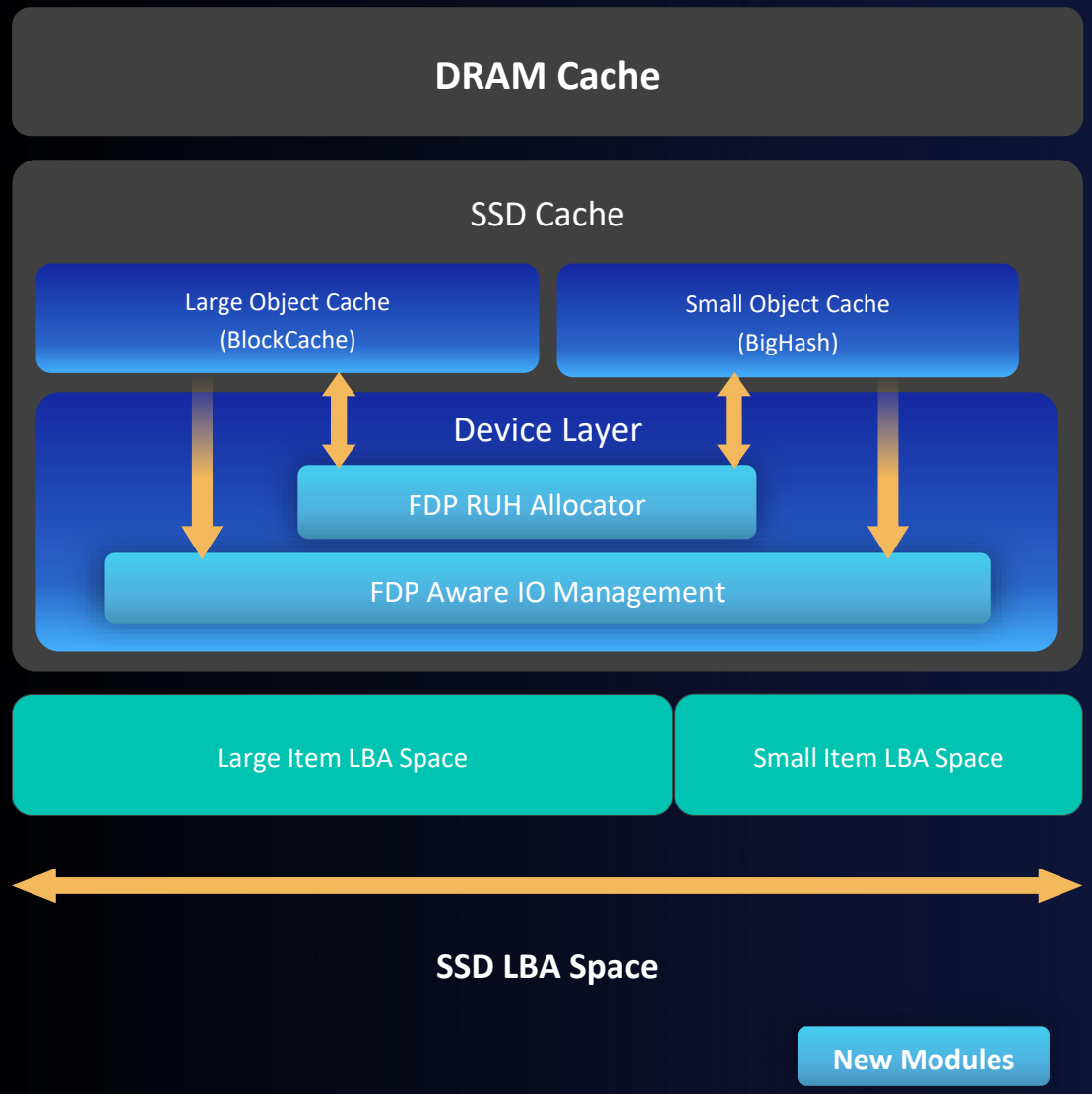
FDP Integration - Approach

CacheLib

- FDP-aware IO management
- FDP RUH allocator
- BlockCache and BigHash allocates different FDP RUHs
- FDP directives are passed to the Linux kernel by I/O Uring Passthrough interface
- This used NVMe character device (Ex: /dev/ng0n1)

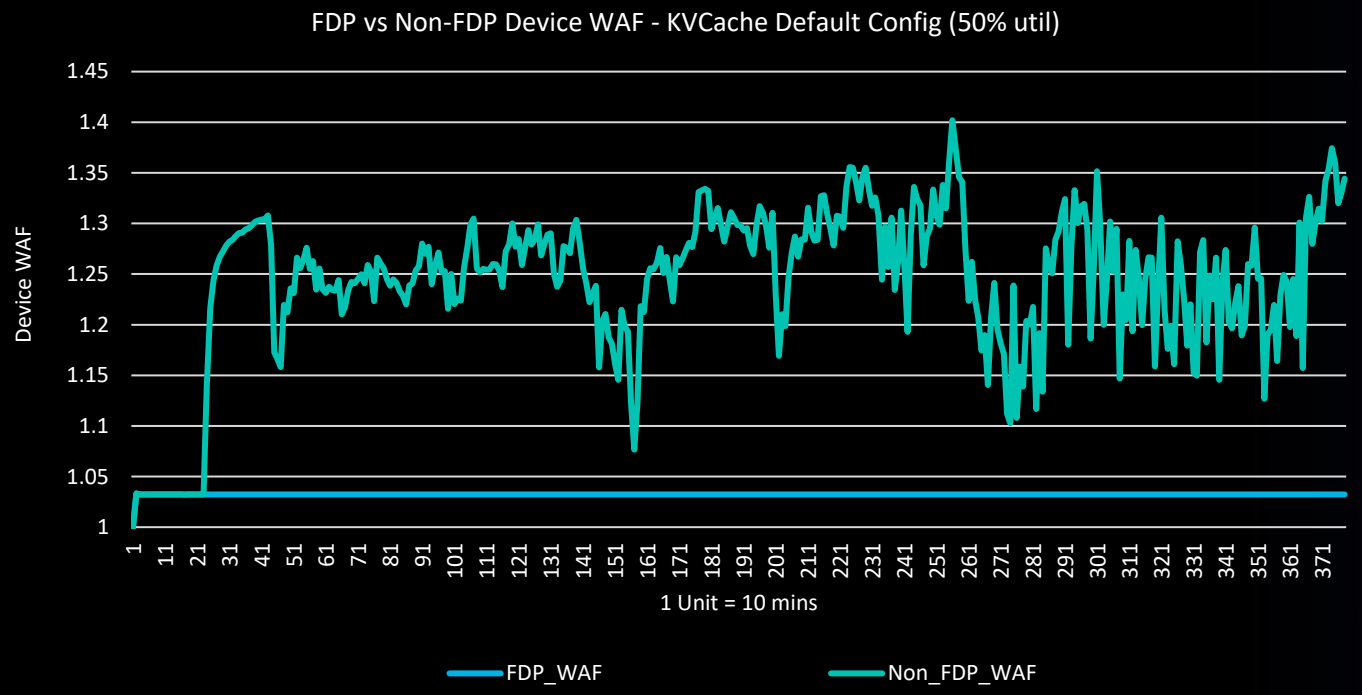
Evaluation

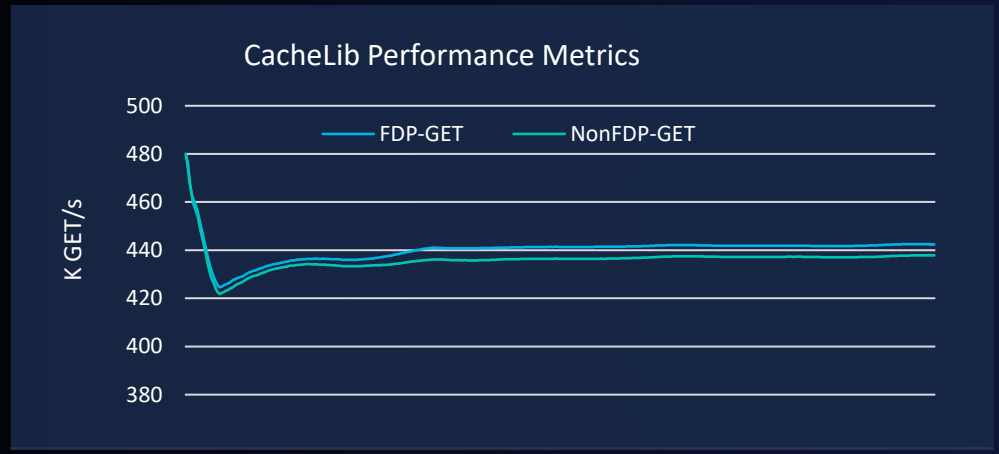
- Samsung PM9D3 is used for evaluation
- Supports up-to 8 FDP RUHs
- Dell PowerEdge R750 server

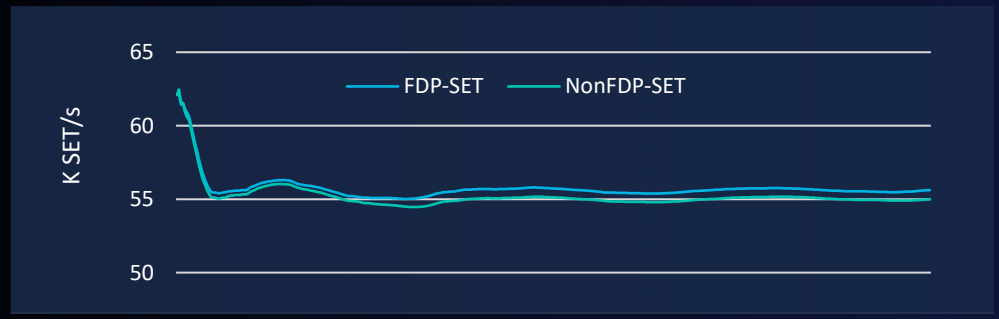


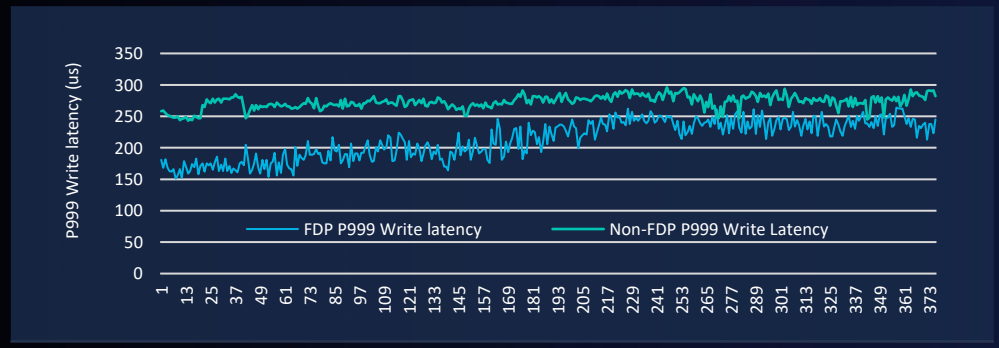
Default KVCache Workload – 50% Utilization

- **FDP reduces the Device WAF from 1.3 to ~1 in default configuration**
- CacheLib metrics like Throughput, Hit rate, App-WAF are unaffected
- FDP improves the write latency
- Note: Default config uses only 50% of the SSD (50% Host OP to control WAF)



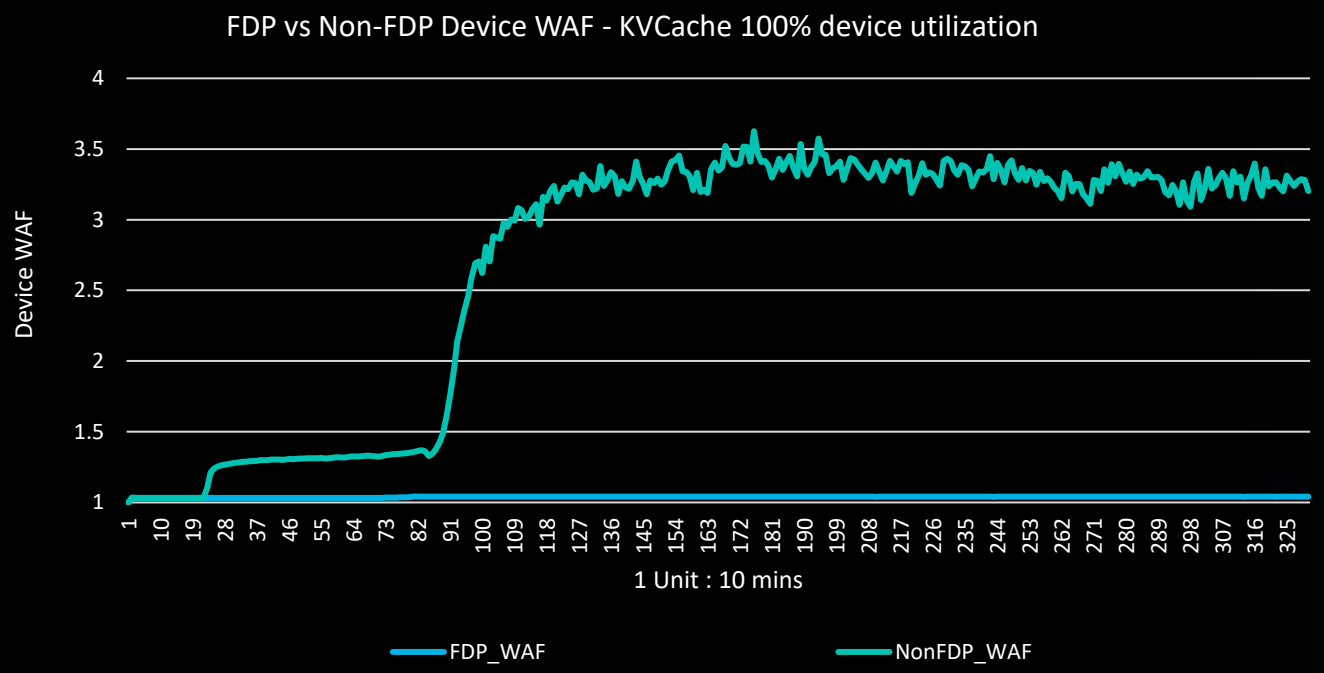


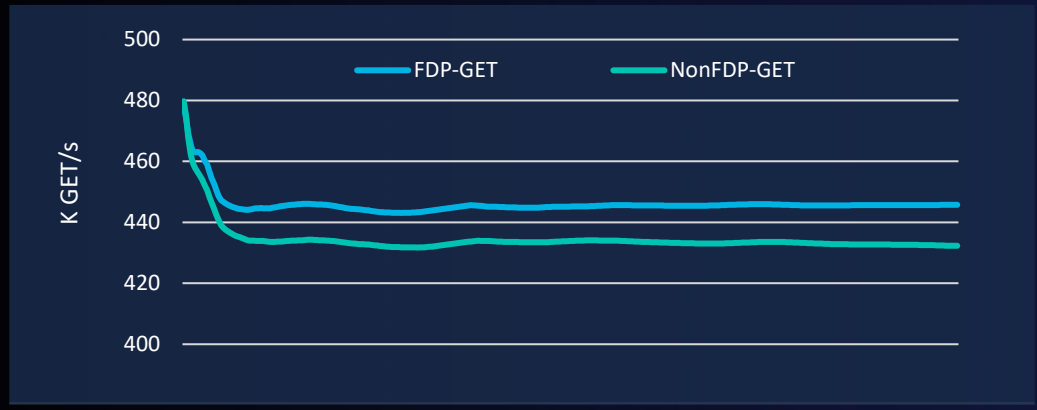


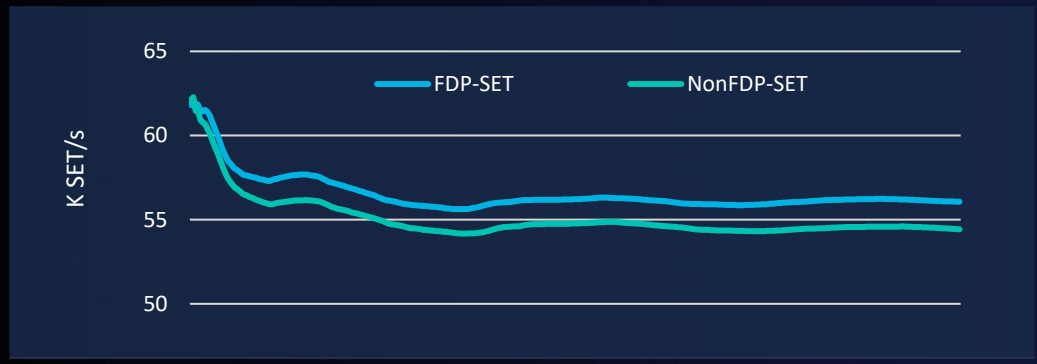


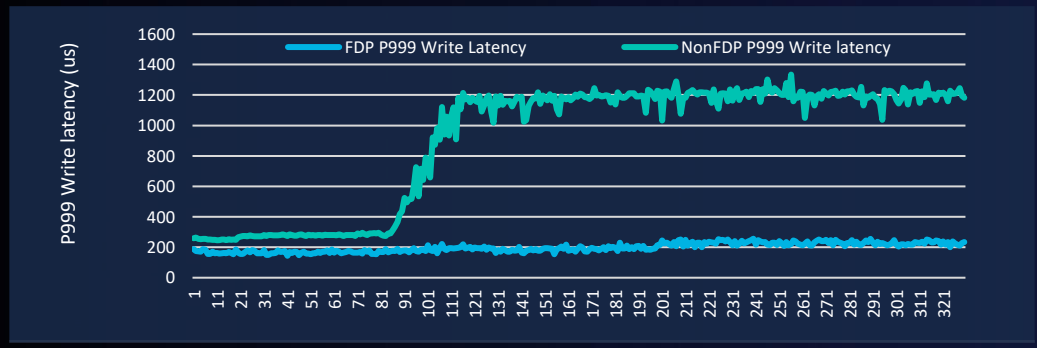
KVCache 100% Device utilization

- **The same KVCache instance uses 100% of SSD now**
- **FDP reduces the Device WAF from 3.5 to ~1**
  - CacheLib metrics like Throughput, Hit rate, App-WAF are unaffected
  - FDP improves the write latency
- So FDP enables efficient Capacity Utilization without WAF impact

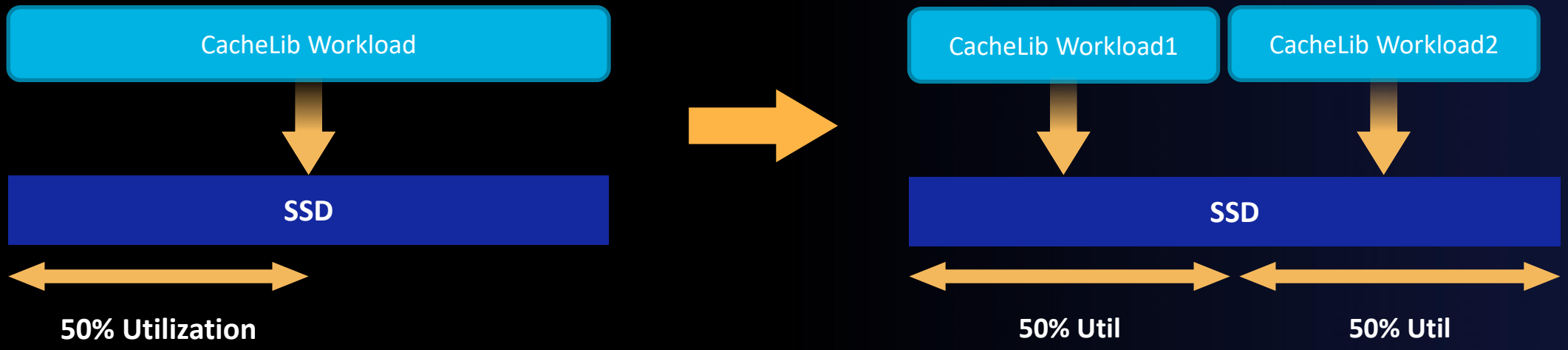








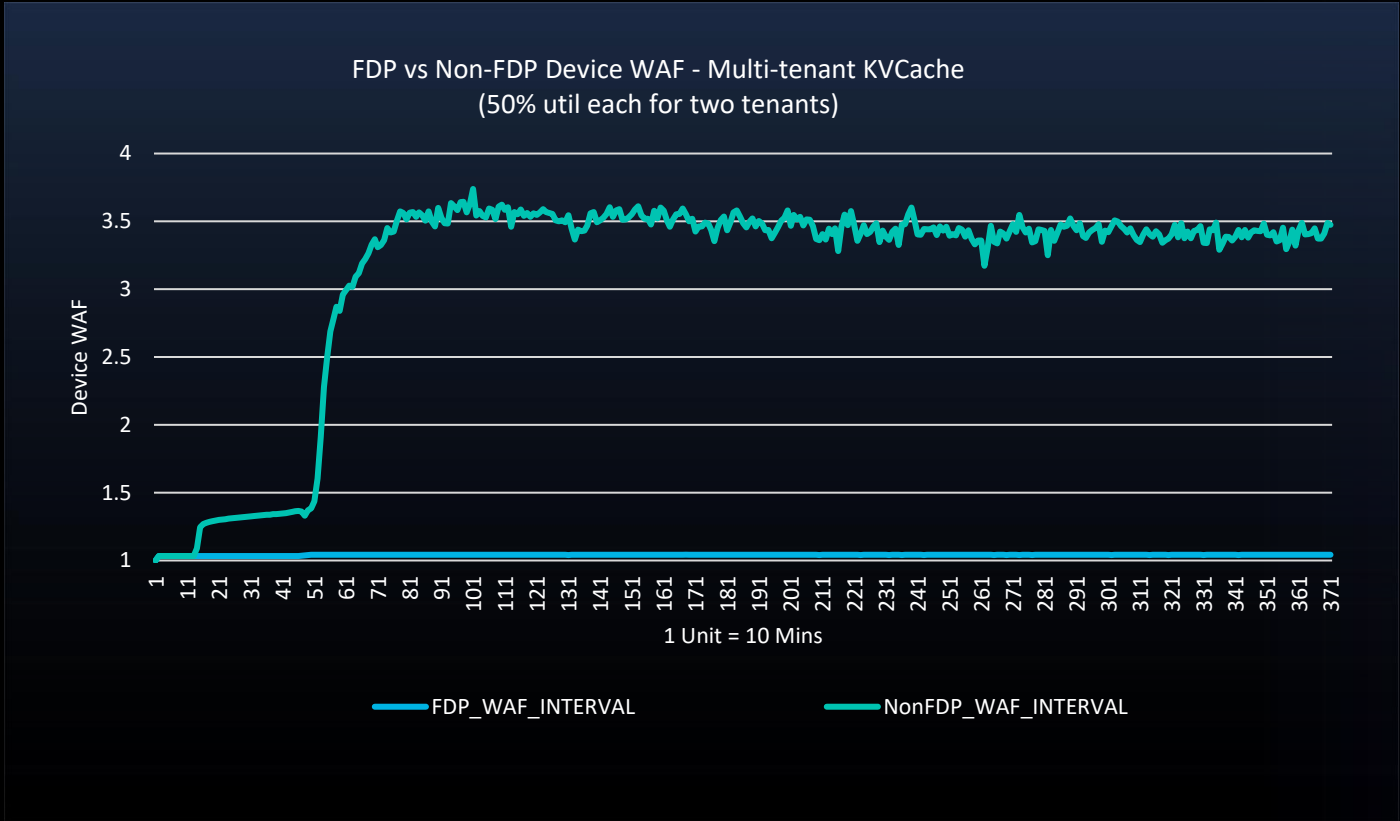
Scaling to More tenants



- **Goal**: Analyse FDP behaviour on increased workload and varied IO pattern
- Workload Throughput increases as more tenants are added
  - Ex: Throughput of N tenants = Single Tenant Throughput X N
- Total Usage of SSD is kept at 100%

Scaling to two identical tenants - Multi-tenant KVCache

- **Two KVCache instances using 50% SSD each** (workload is doubled)
- **FDP reduces the device WAF from 3.5 to ~1**
- CacheLib metrics like Throughput, Hit rate, App-WAF are unaffected
- FDP provides tenant isolation in terms of read and write latencies

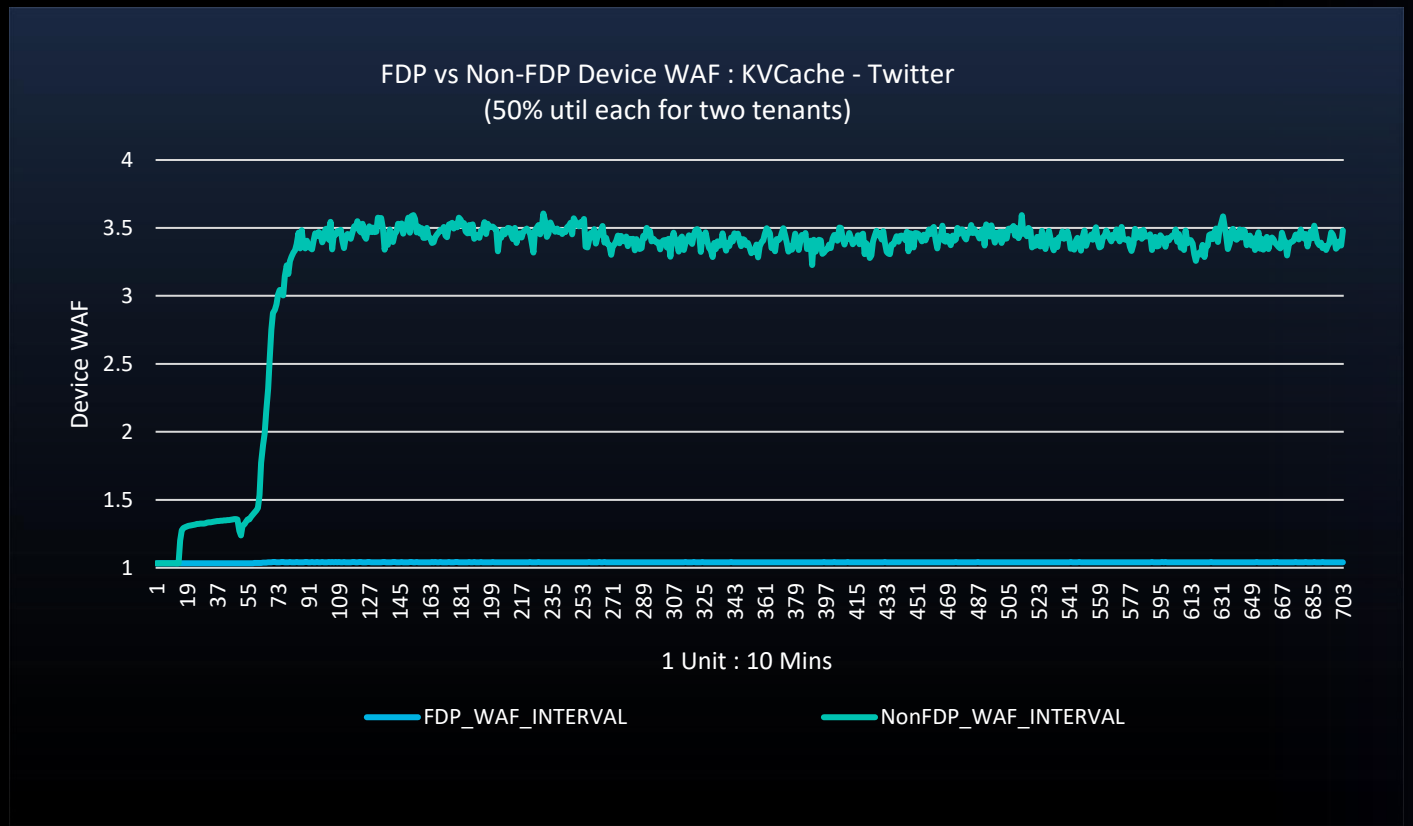


Metrics	Non-FDP		FDP	
	Tenant1	Tenant2	Tenant1	Tenant2
Write Throughput, SET/s	48 K	48.8 K	49.3 K	49.4 K
Read Throughput, GET/s	382.6 K	388 K	392 K	392.5 K
P999 Write latency, usec	606us	639us	146us	153us
P999 Read latency, usec	1422us	1670us	453us	413us

Scaling to two heterogeneous tenants – KVCache +Twitter

SAMSUNG

- **KVCache instance using 50% SSD, Twitter workload uses the other 50%**
- Both workloads achieve WAF of ~1 in FDP mode as single tenants
- **FDP reduces the Device WAF from 3.5 to ~1 in this experiment**
- CacheLib metrics like Throughput, Hit rate, App-WAF are unaffected
- FDP provides tenant isolation for varied workloads also.



Metrics	Non-FDP		FDP	
	KVCache	Twitter	KVCache	Twitter
Write Throughput, SET/s	43 K	20 K	46 K	20 K
Read Throughput, GET/s	344 K	5 K	372 K	5 K
P999 Write latency, usec	1294 us	1464 us	140 us	92 us
P999 Read latency, usec	542 us	653 us	367 us	397 us

SAMSUNG

Summary

Key Points

- Achieves device WAF of ~1 in production workloads
- Enables efficient Capacity utilization and better SSD latencies
- CacheLib design and Application-WAF remain unchanged
- CacheLib KPIs like Performance or Hit-rate remain unchanged
- WAF behavior remains the same while scaling to more tenants
- FDP provides an efficient NAND media level isolation for tenants

Upstream status:

- Merged upstream on Jan 26, 2024

Thank You