FairyWREN: A Sustainable Cache for Emerging Write-Read-Erase Flash Interfaces

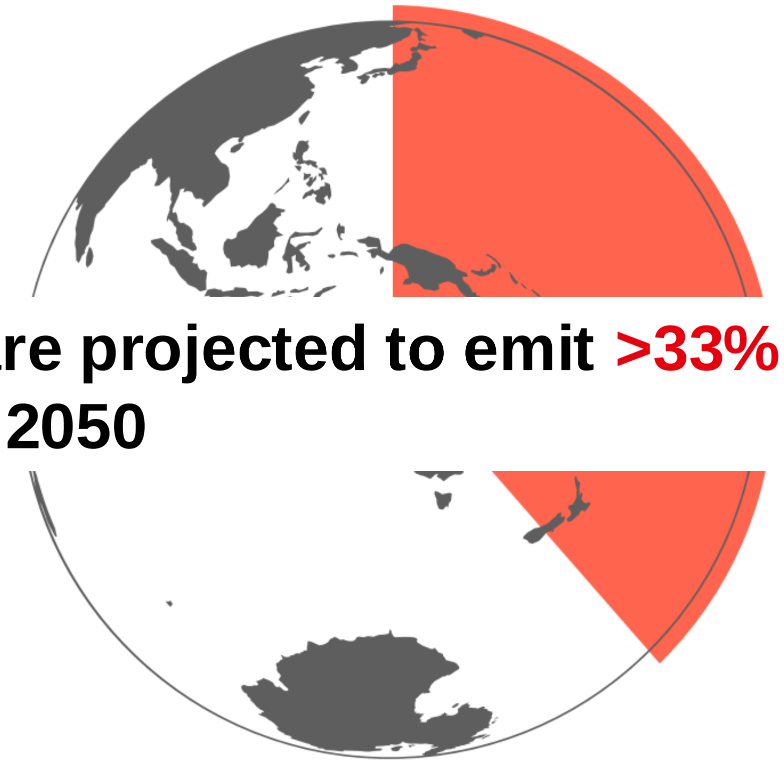
Sara McAllister
Sherry (Yucong) Wang, Benjamin Berg#, Daniel S. Berger*,
George Amvrosiadis, Nathan Beckmann, Gregory R. Ganger

Presented by Qingyuan Chen

Contents

Contents

Flash cache emissions



Datacenters are projected to emit >33% global emissions by 2050

ACM TechBrief - Computing and Climate Change ‘21

40% of server emissions are storage

Lyu HotCarbon ‘23

Flash cache emissions

- Embodied emissions are projected to be 82% of emissions
    - Chasing Carbon - Gupta HPCA 2021
- 61% of datacenter embodied emissions are storage
    - GreenSKU - Wang ISCA ‘24

Flash cache emissions

- Embodied emissions are projected to be 82% of emissions
  - Chasing Carbon - Gupta HPCA 2021
- 61% of datacenter embodied emissions are storage
  - GreenSKU - Wang ISCA ‘24

- Flash is an increasingly attractive option for caching

How to reduce flash emissions

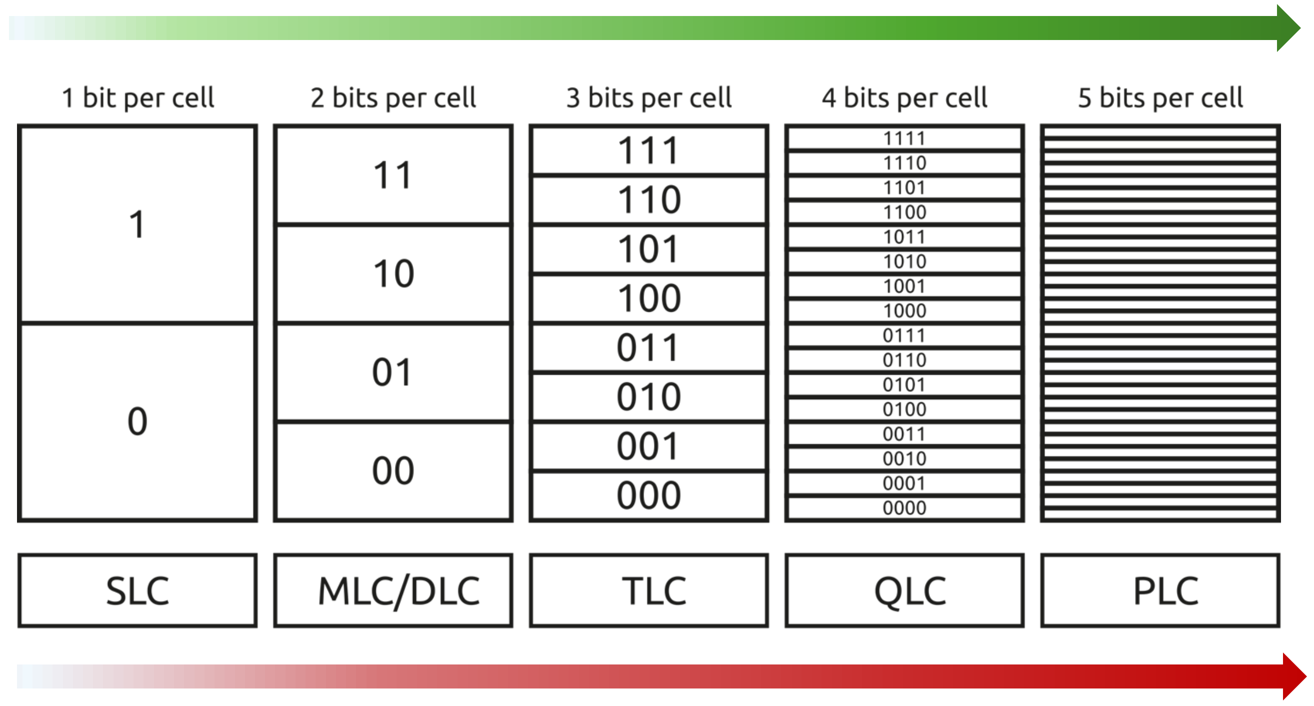
- Low DRAM overhead
- Using denser flash is possible to reduce emissions
- Lengthen device lifetime to improve datacenter sustainability

Low DRAM overhead

- **12x** less embodied emissions per bit
- DRAM has larger operational emissions than flash
- **30 bits / object metadata overhead**
  - Flashield (Eisenman NSDI ’19)
  - **2 TB flash cache 75 GB memory overhead**

Using denser flash

- Denser flash can reduce emissions by reduce *NAND cell* needed



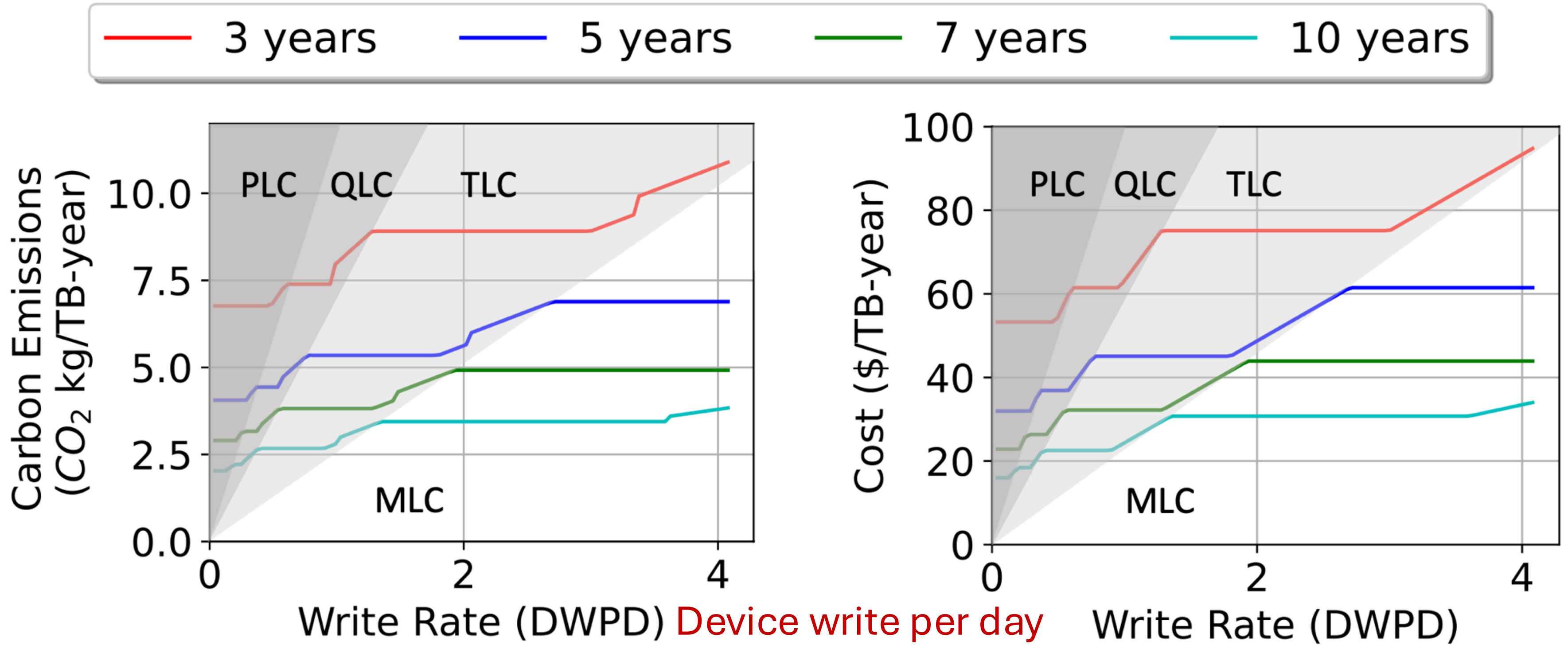
Less HW embodied emissions

Fewer write endurance

https://www.tomshardware.com/news/solidigm-plc-nand-ssd, 2022

Lengthen device lifetime

- Shaded regions show the best flash density for a given write rate

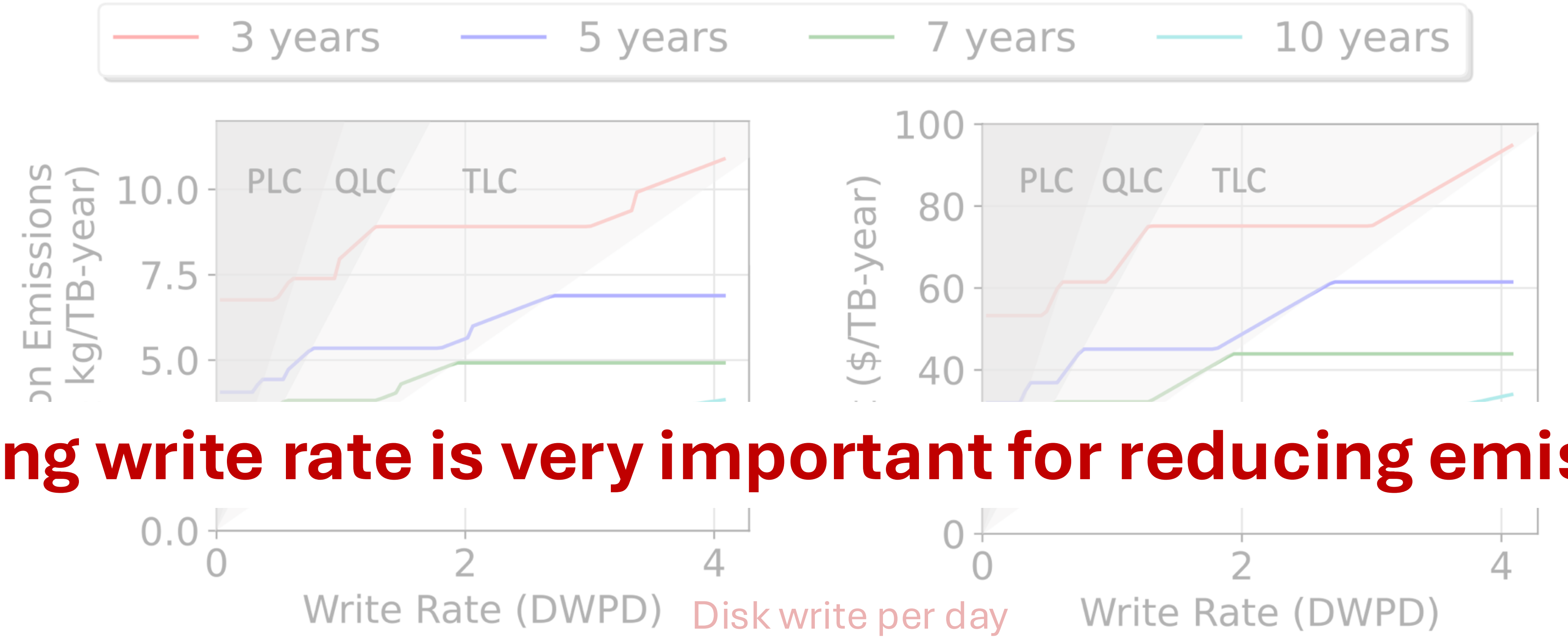


Device write per day

Carbon emissions **Cost**

Lengthen device lifetime

- Shaded regions show the best flash density for a given write rate



Reducing write rate is very important for reducing emissions

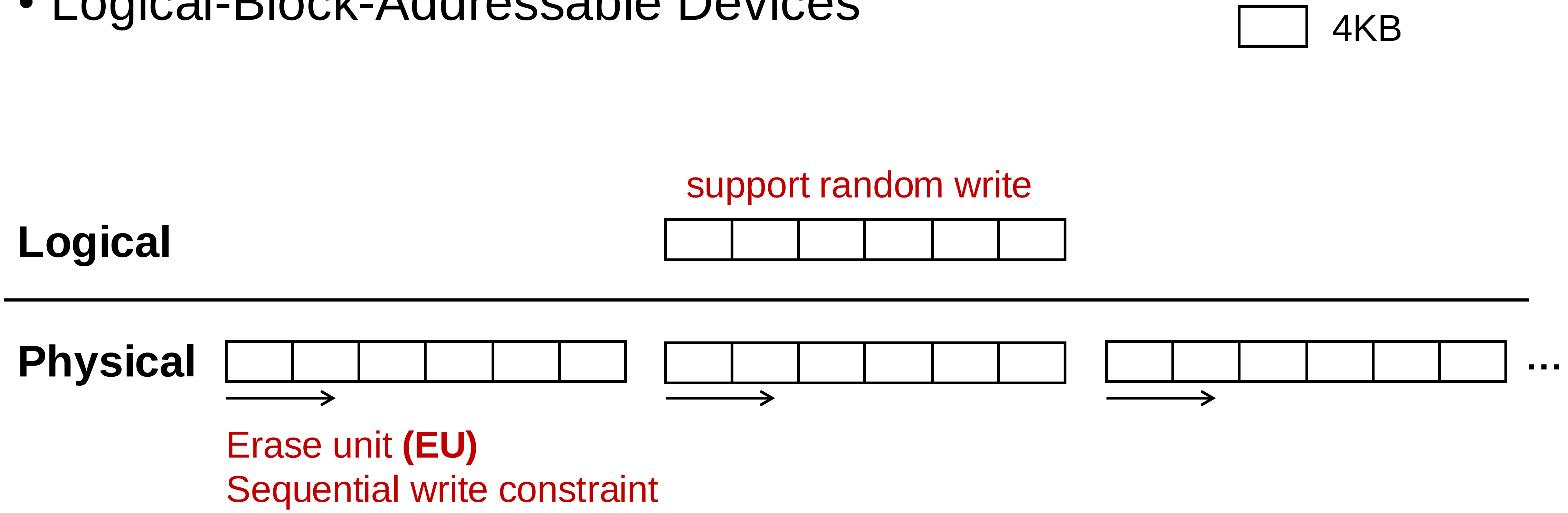
Disk write per day

Carbon emissions

Cost

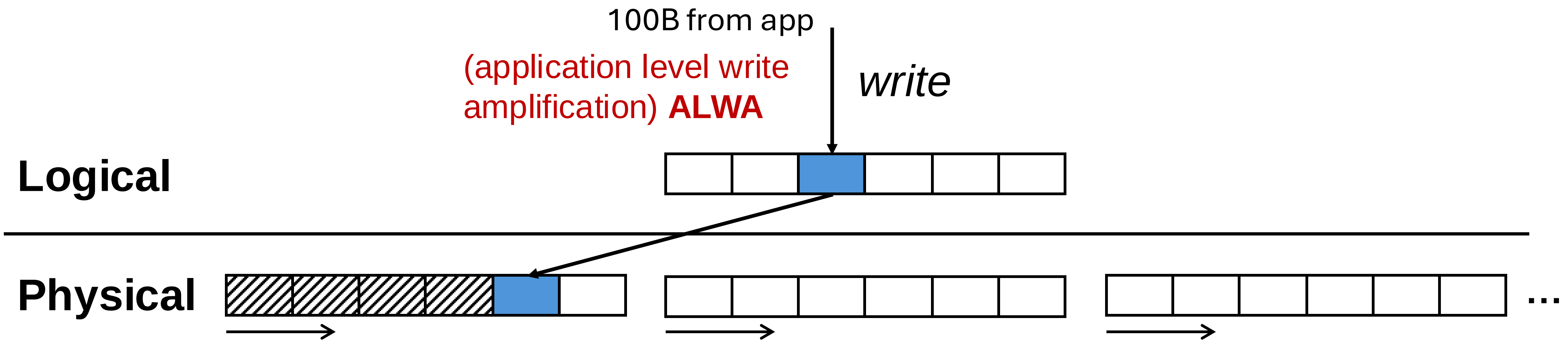
LBAD interface

- Logical-Block-Addressable Devices



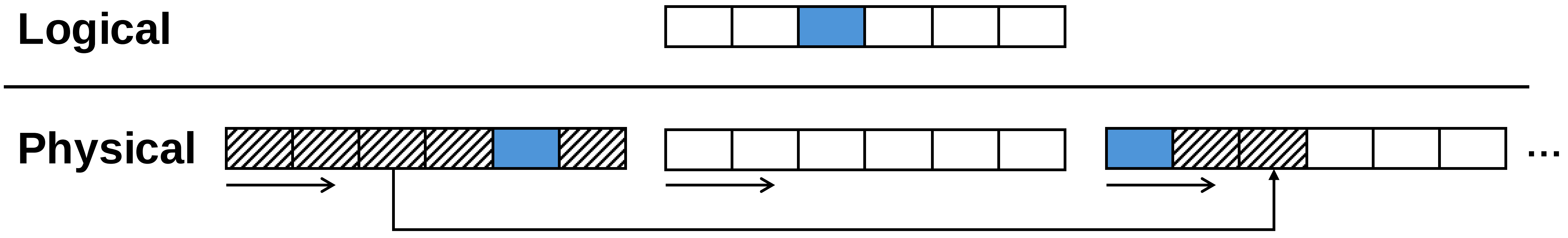
LBAD interface

- Logical-Block-Addressable Devices



LBAD interface

- Logical-Block-Addressable Devices

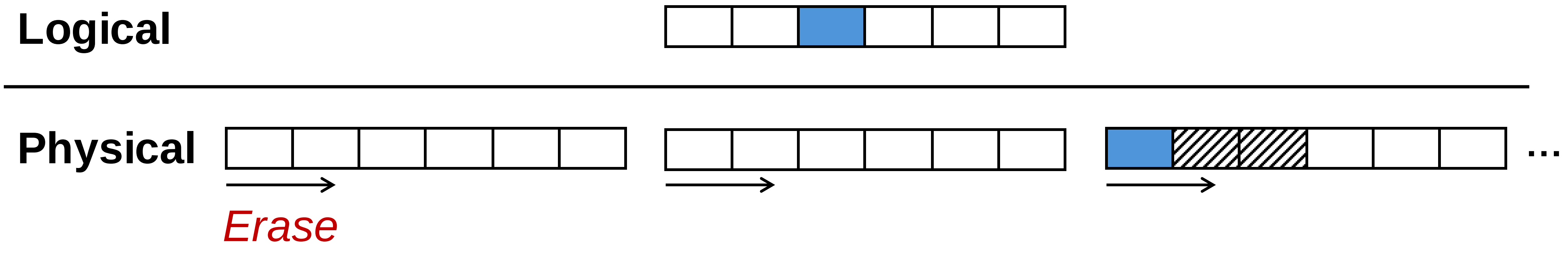


Garbage Collection

(data level write amplification) **DLWA**

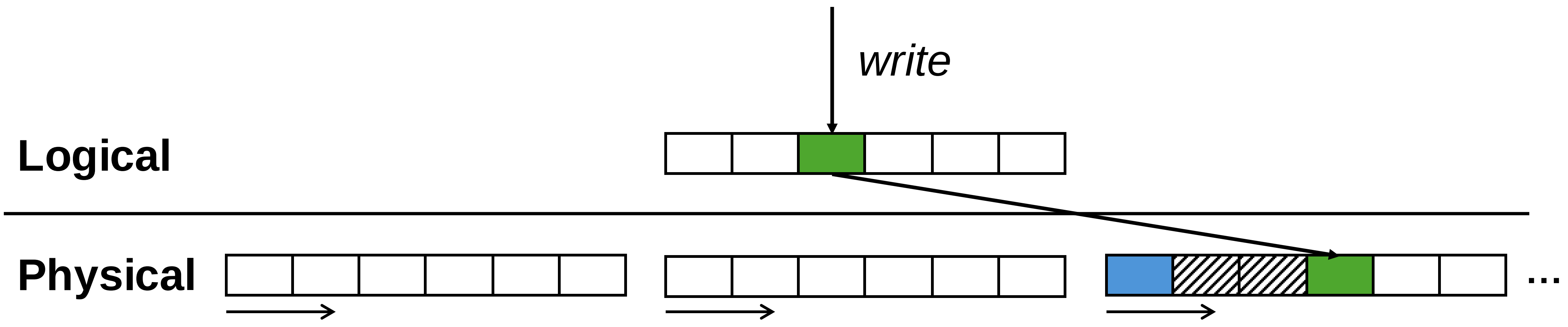
LBAD interface

- Logical-Block-Addressable Devices



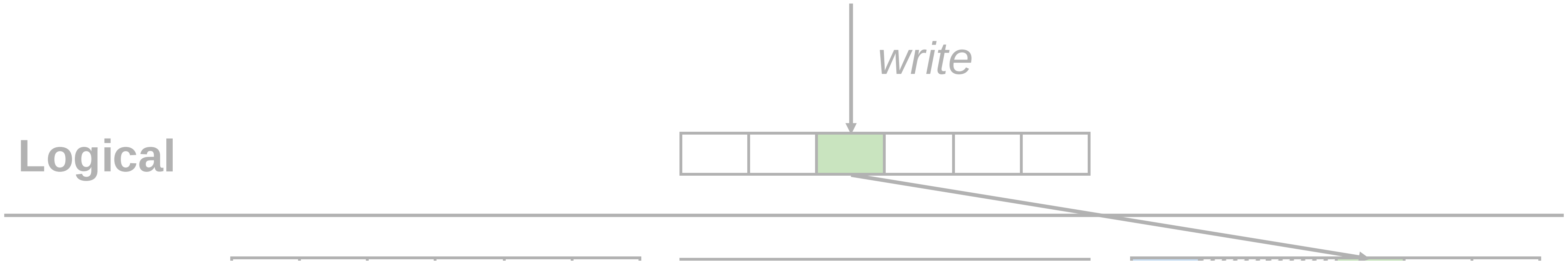
LBAD interface

- Logical-Block-Addressable Devices



LBAD interface

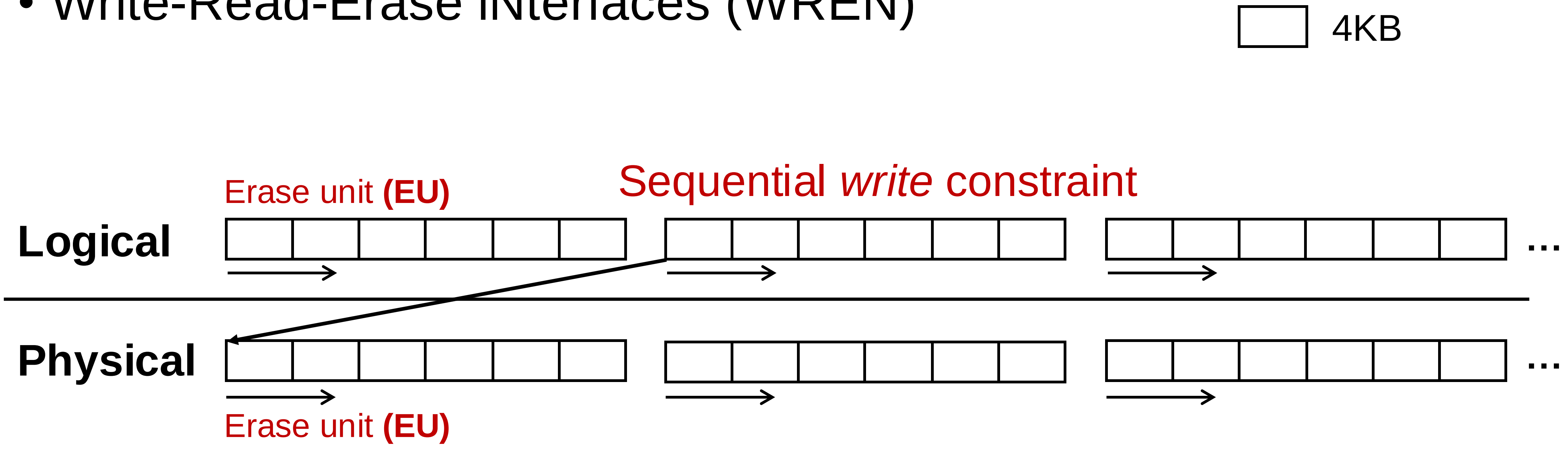
- Logical-Block-Addressable Devices



LBAD interface doesn’t allow cache to control all writes

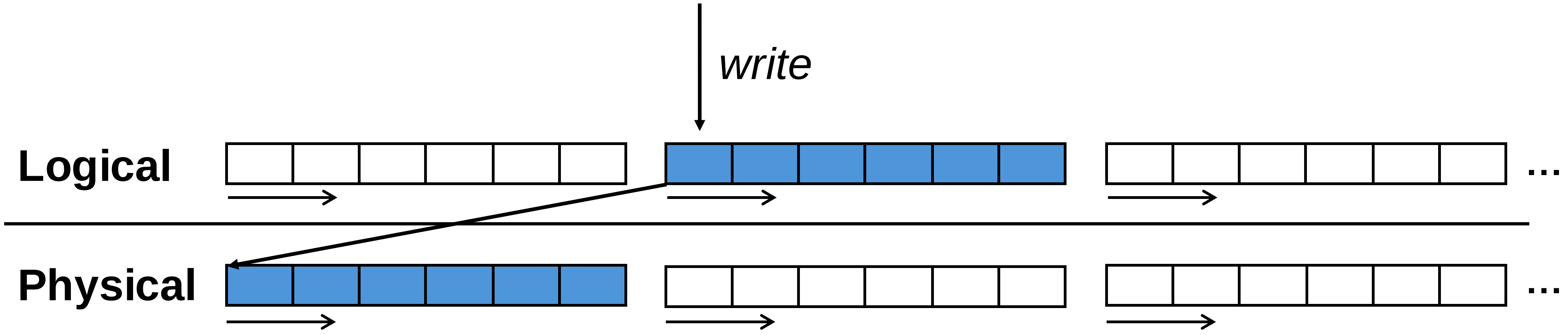
Write-Read-Erase iNterfaces

- Write-Read-Erase iNterfaces (WREN)



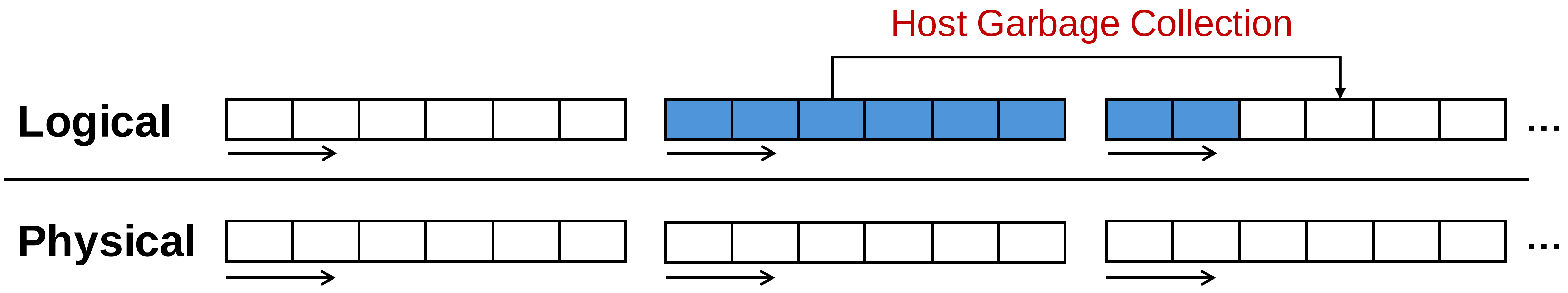
Write-Read-Erase iNterfaces

- Write-Read-Erase iNterfaces (WREN)



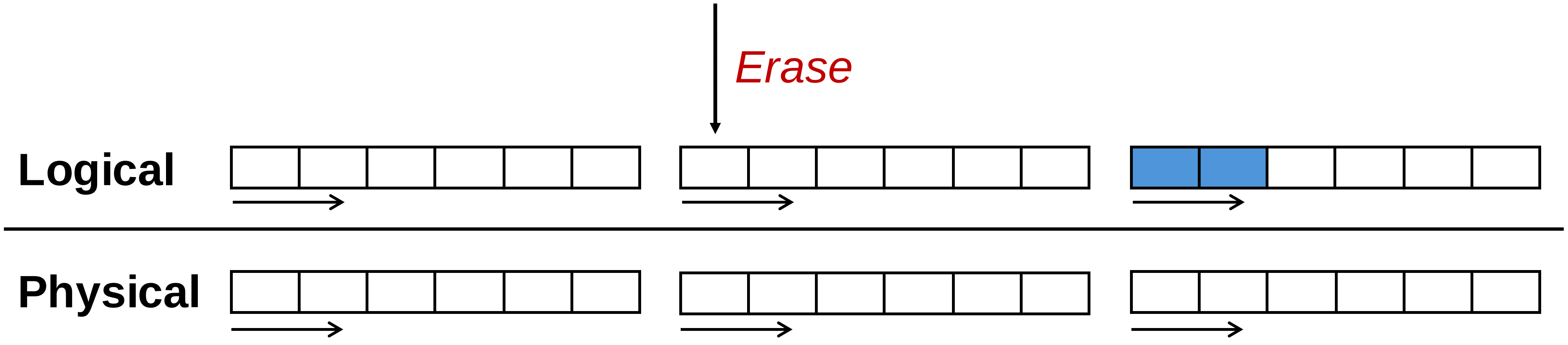
Write-Read-Erase iNterfaces

- Write-Read-Erase iNterfaces (WREN)



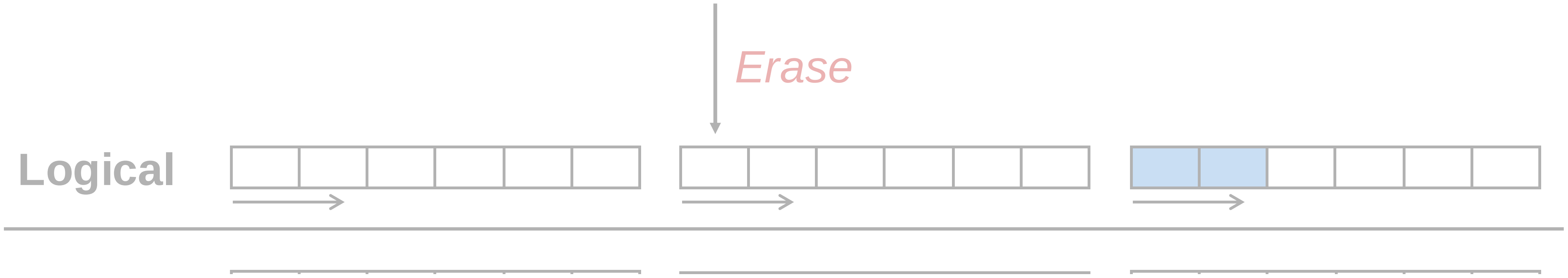
Write-Read-Erase iNterfaces

- Write-Read-Erase iNterfaces (WREN)



Write-Read-Erase iNterfaces

- Write-Read-Erase iNterfaces (WREN)

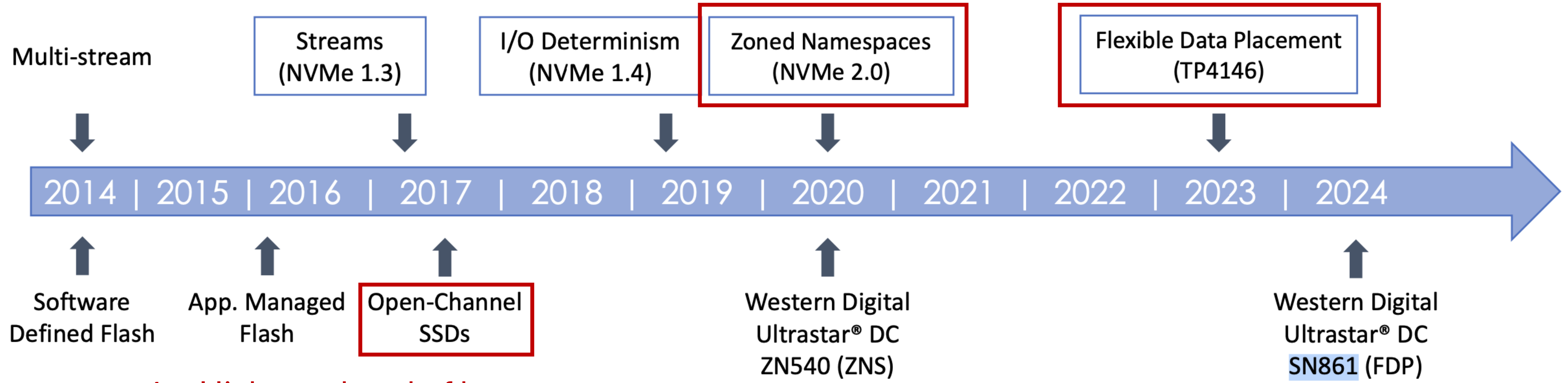


WREN interface allow cache to control all writes

History of New interface SSDs

- **From FMS-2024**

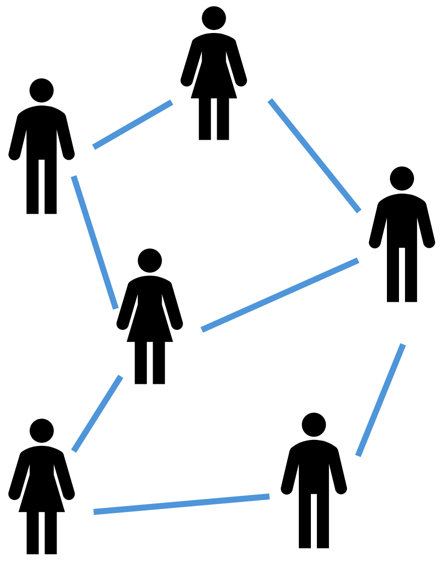
FairWERN targeted on ZNS/FDP SSD and evaluated on ZNS SSD


Multi-stream
Streams (NVMe 1.3)
I/O Determinism (NVMe 1.4)
Zoned Namespaces (NVMe 2.0)
Flexible Data Placement (TP4146)
2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024
Software Defined Flash
App. Managed Flash
Open-Channel SSDs
Western Digital Ultrastar® DC ZN540 (ZNS)
Western Digital Ultrastar® DC SN861 (FDP)

1. High overhead of host management
2. High overhead of development

Contents

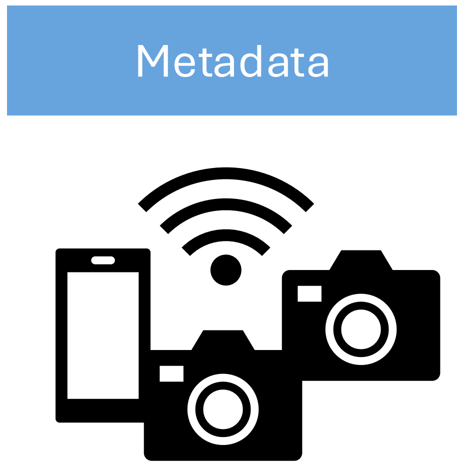
Tiny objects are prevalent



Social Graph

Meta *(Facebook)* social graph edges

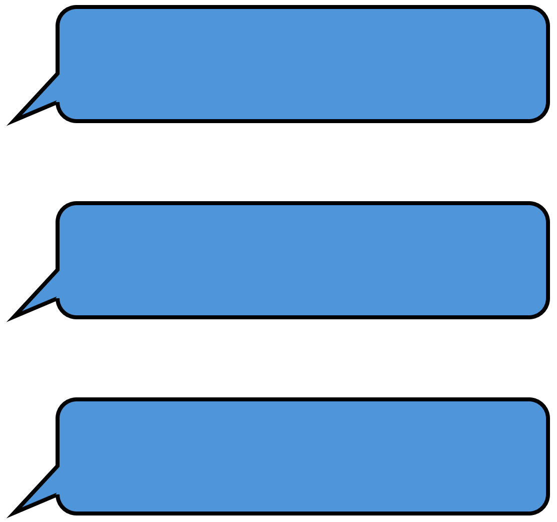
~100 Bytes



IoT Metadata

Microsoft Azure sensor metadata

~300 Bytes



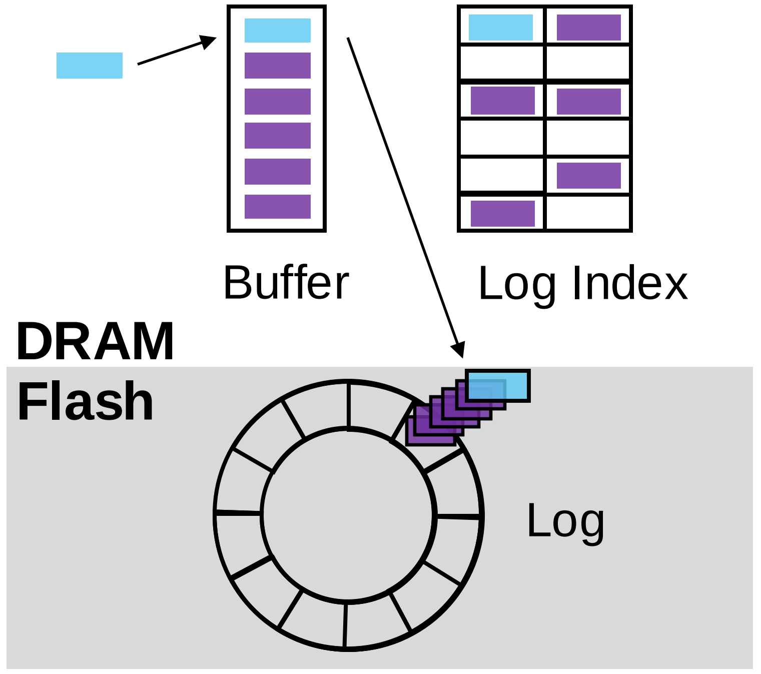
Tweets

X *(Twitter)* tweets average

<33 characters

Log-structure & Set-associative cache

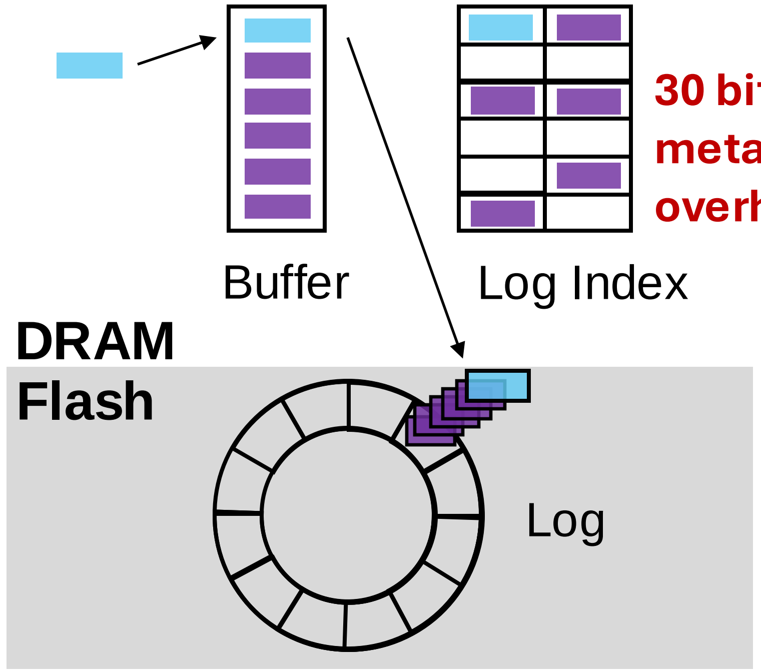
Flashield (Eisenman NSDI'19)



Log-structure & Set-associative cache

Flashield (Eisenman NSDI'19)

High memory overhead



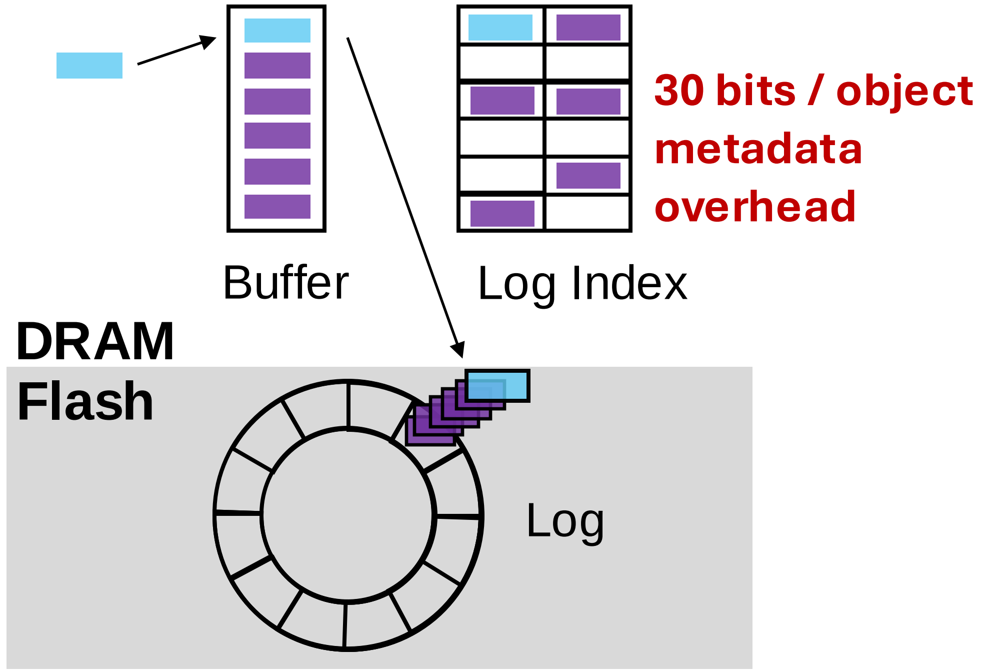
30 bits / object metadata overhead

e.g. 2TB 100B object consume 75GB memory

Log-structure & Set-associative cache

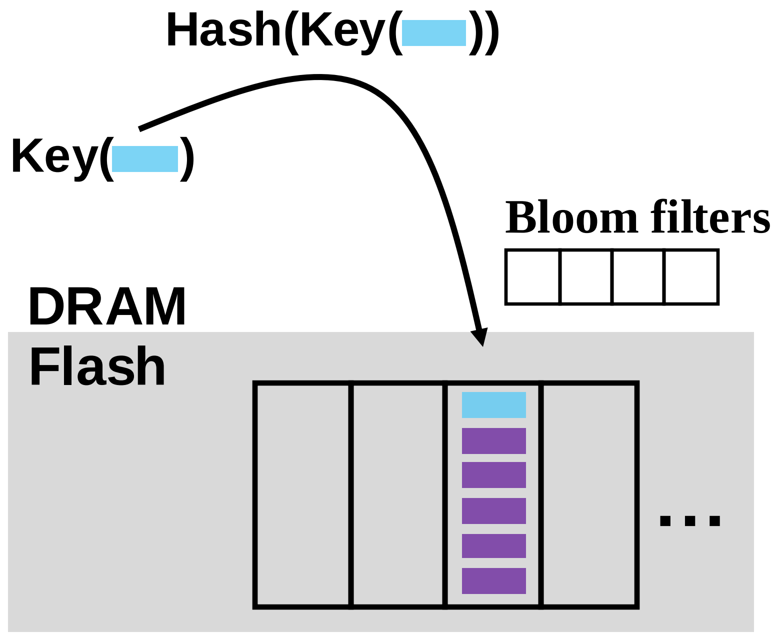
Flashield (Eisenman NSDI'19)

High memory overhead



e.g. 2TB 100B object consume 75GB memory

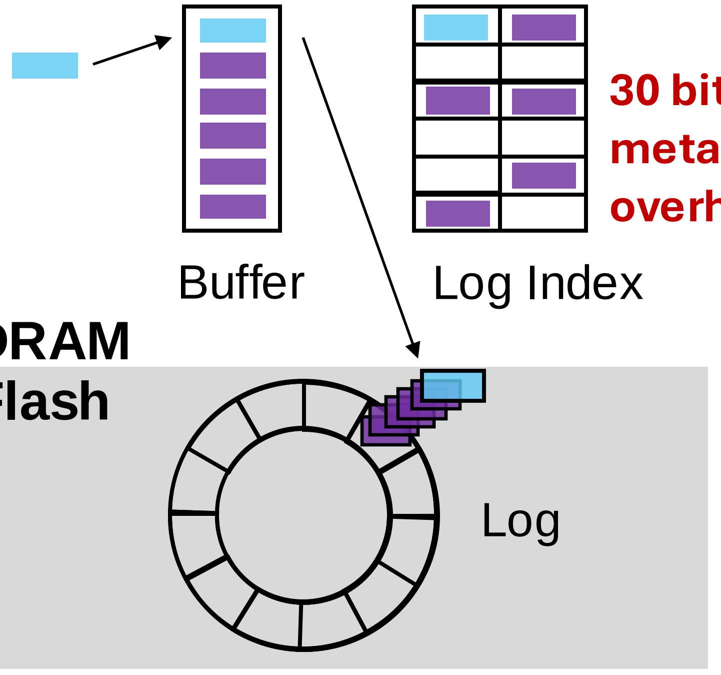
CacheLib (Berg OSDI'20)



Log-structure & Set-associative cache

Flashield (Eisenman NSDI’19)

High memory overhead

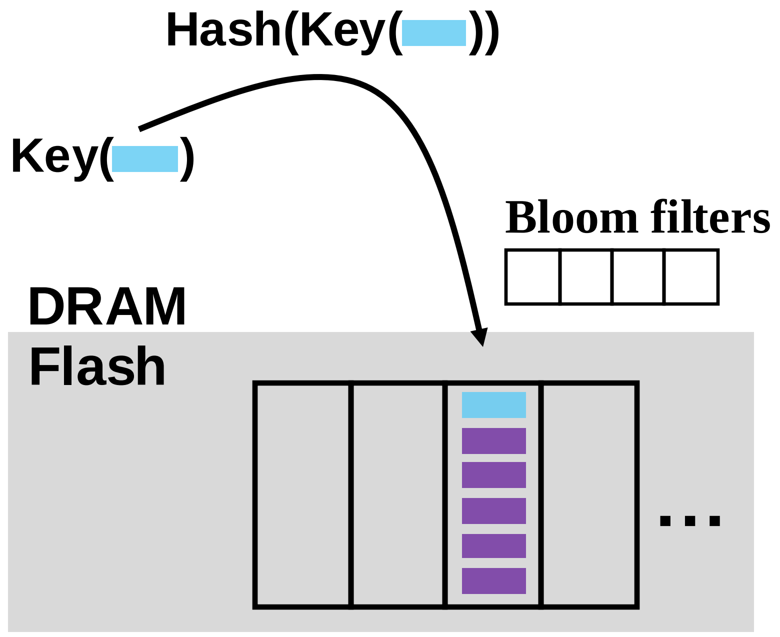


30 bits / object metadata overhead

e.g. 2TB 100B object consume 75GB memory

CacheLib (Berg OSDI’20)

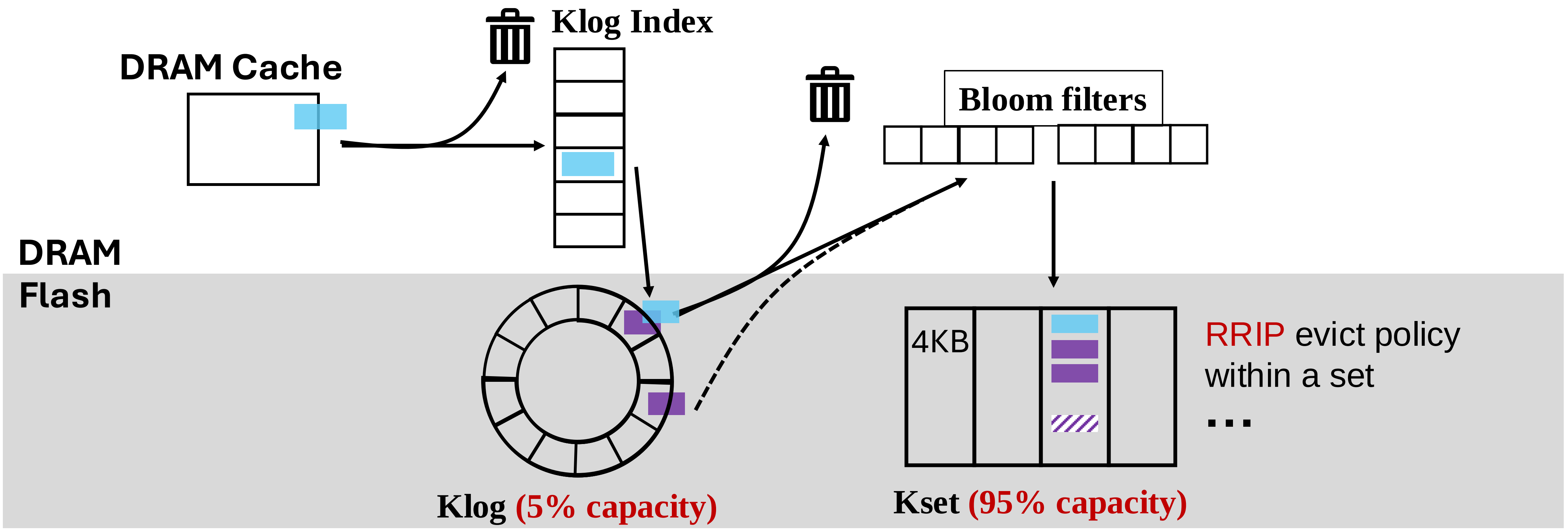
High ALWA



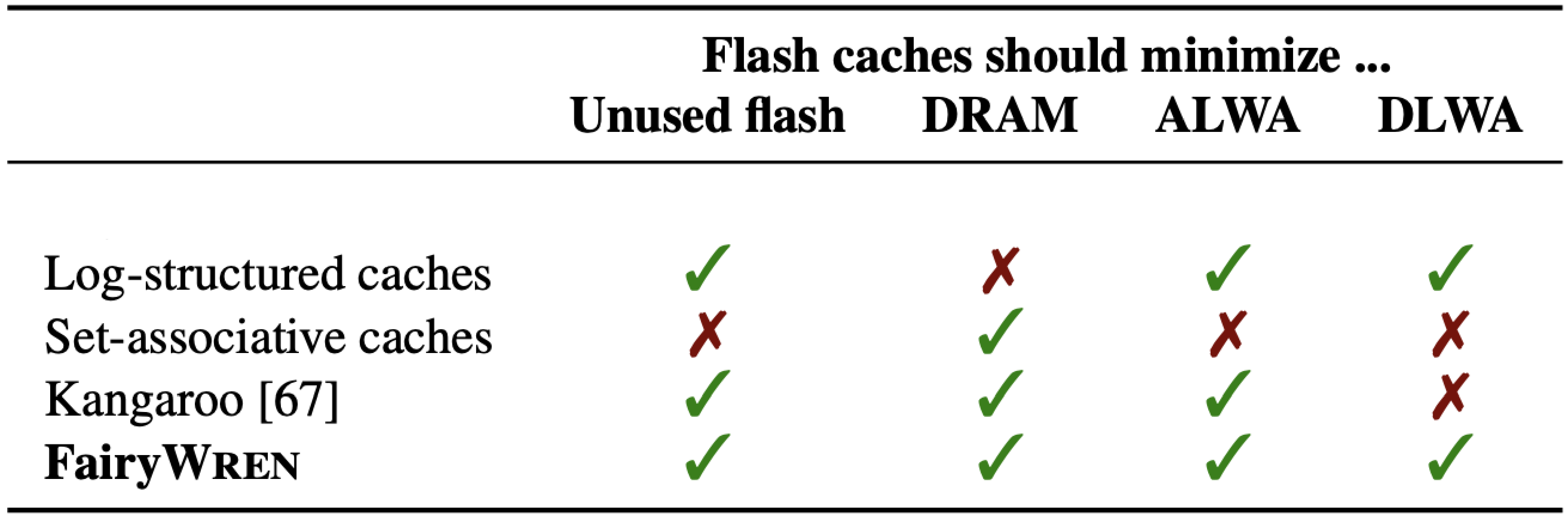
e.g. 100B object can cause 40x ALWA

Kangaroo Design

Kangaroo(SOSP’21)



Summary of existing solutions



	Flash caches should minimize ...			
	Unused flash	DRAM	ALWA	DLWA
Log-structured caches	✓	✗	✓	✓
Set-associative caches	✗	✓	✗	✗
Kangaroo [67]	✓	✓	✓	✗
FairyWren	✓	✓	✓	✓

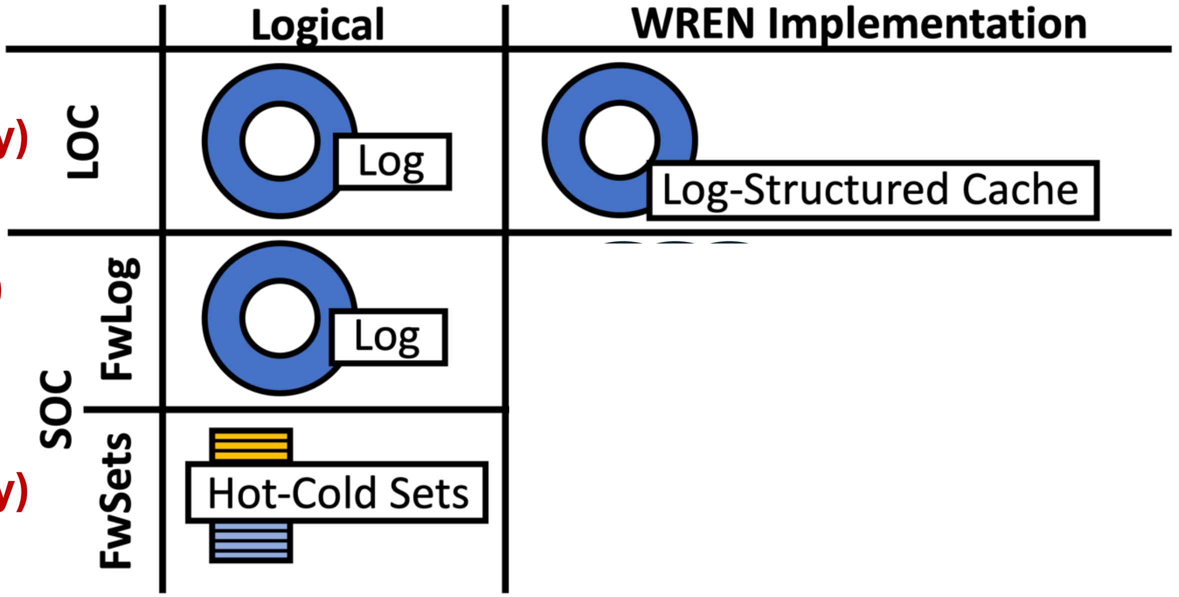
Contents

FairyWREN architecture

(10% capacity)

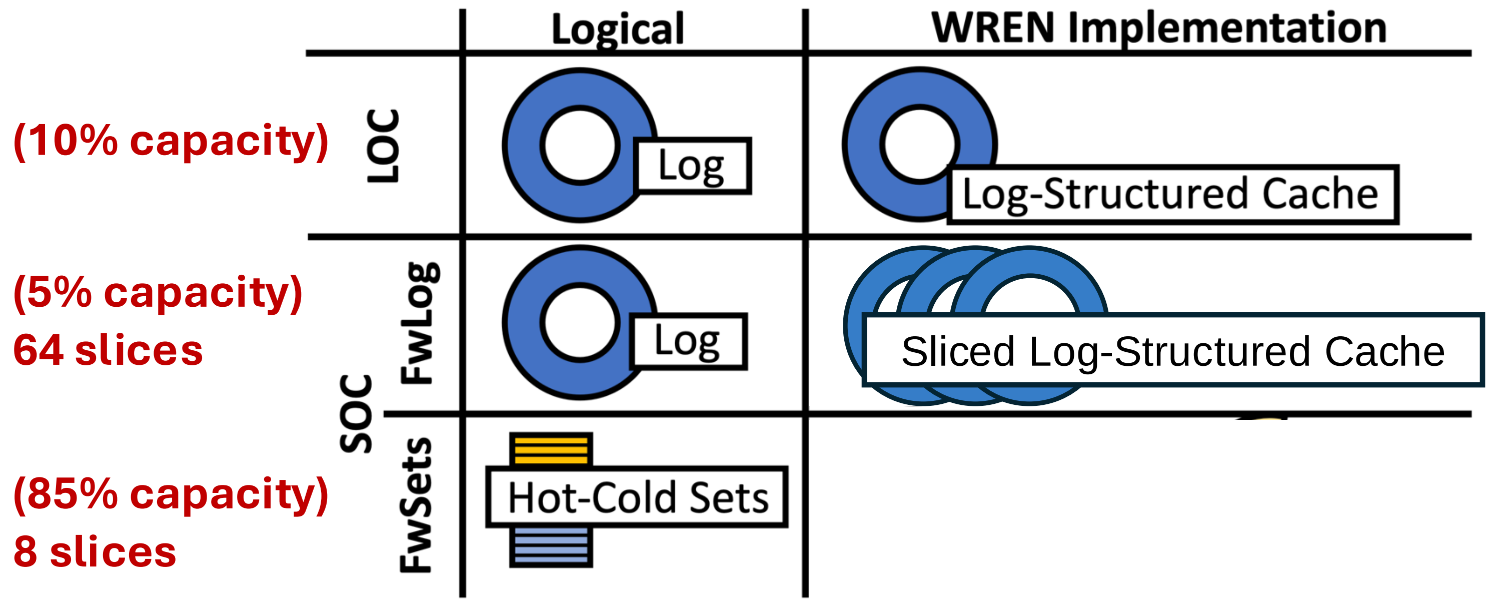
(5% capacity)
64 slices

(85% capacity)
8 slices



- LOC (Similar with KLog)
  - Item size larger than **2KB**

FairyWREN architecture



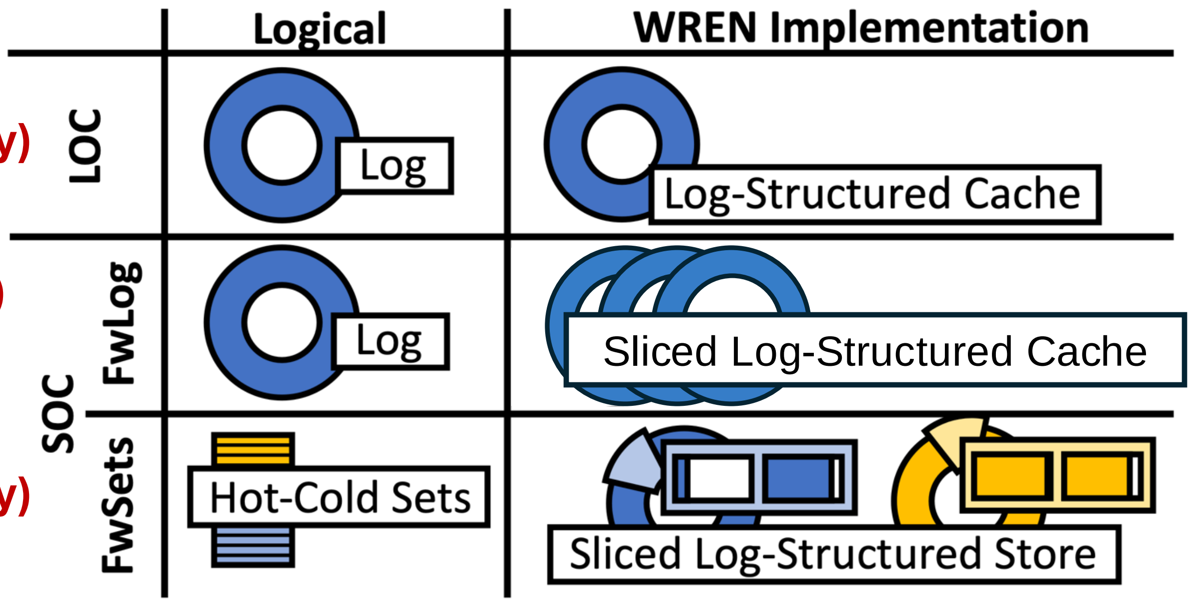
- LOC (Similar with KLog)
  - Item size larger than **2KB**
- SOC
  - FwLog(Similar with KLog)

FairyWREN architecture

(10% capacity)

(5% capacity)
64 slices

(85% capacity)
8 slices



- LOC (Similar with KLog)
  - Item size larger than **2KB**
- SOC
  - FwLog(Similar with KLog)
  - FwSets(Similar with KSet)
    - **8KB/4KB** set size
    - FWSets stores the sets themselves as objects in a log-structured store

DRAM usage

Component	Kangaroo	Naïve SOC	SOC
Log total	*48 bits/obj*	*48 bits/obj*	*48 bits/obj*
Set index	–	**$\approx$ *3.1 b***	**$\approx$ *1.4 b***
Sets (other)	4 b	4 b	4 b
Sets total	*4 bits/obj*	*7.1 bits/obj*	*5.4 bits/obj*
Log metadata	$\approx$ 0.8 b	$\approx$ 0.8 b	$\approx$ 0.8 b
Log size	5% = 2.4 b	5% = 2.4 b	5% = 2.4 b
Set size	95% = 3.8 b	95% = 6.7 b	95% = 5.1 b
Total	**7.0 bits/obj**	**9.9 bits/obj**	**8.3 bits/obj**

FWSet memory index overhead
4KB set size
200B object size

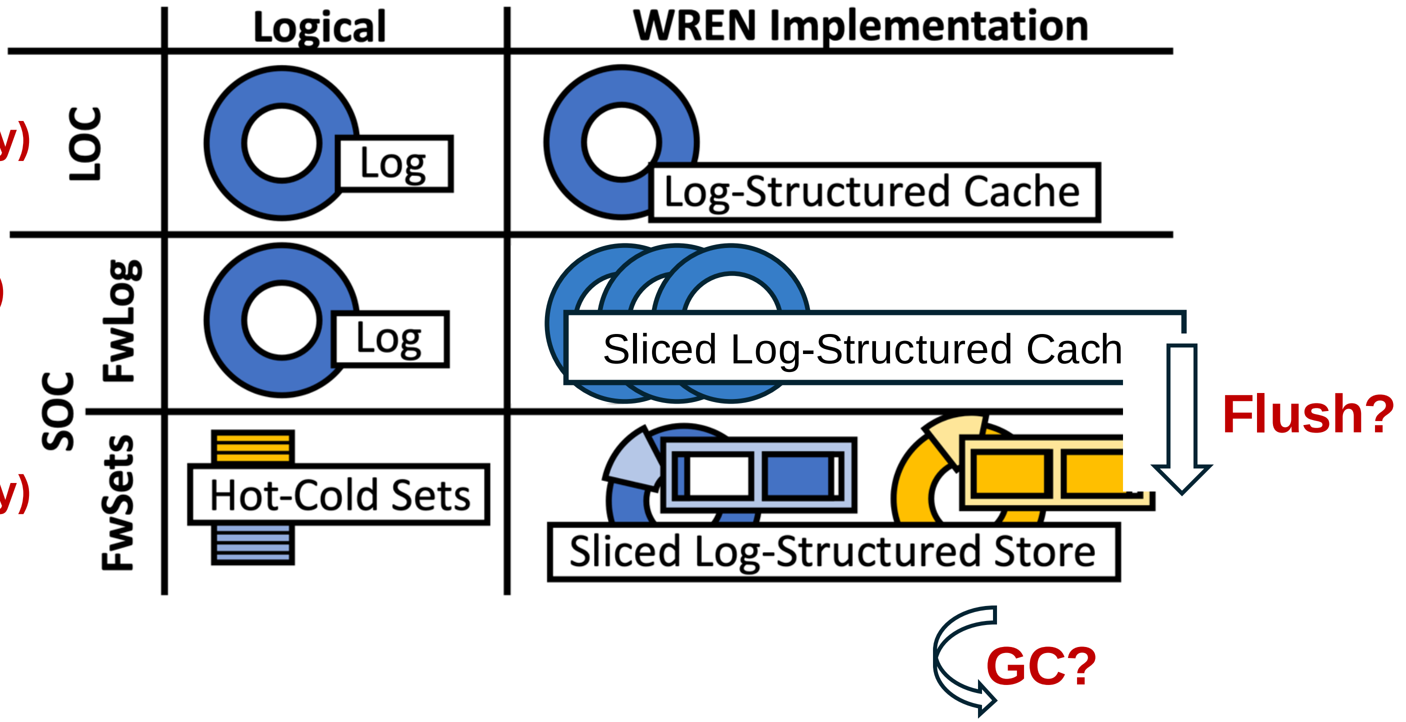
FWLog index overhead
64 Slices can reduce 3bit/obj

FairyWREN architecture

(10% capacity)

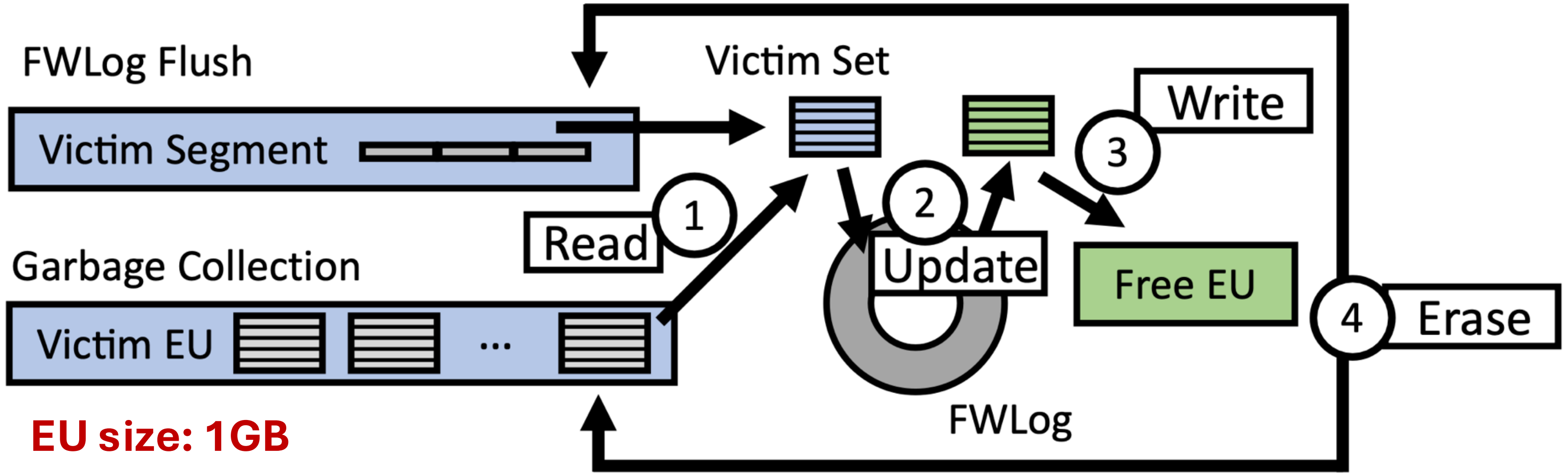
(5% capacity)
64 slices

(85% capacity)
8 slices



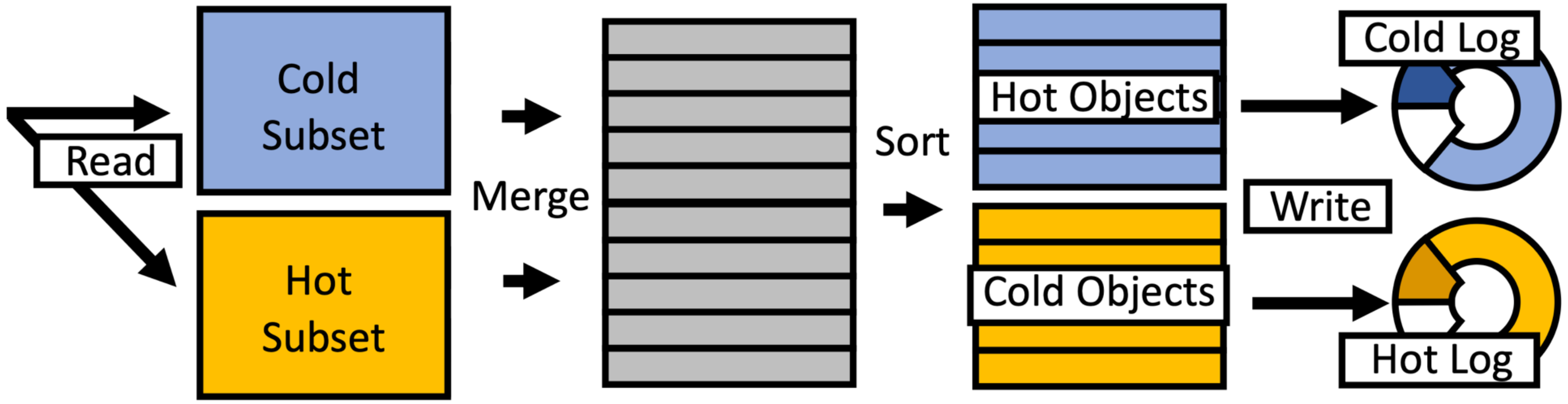
Nest packing (GC & Eviction)

- 1. Victim Set
  - Each objects from Victim FWLog hashes to an Victim Set
  - Each set in victim EU is Victim Set
- 2. Finding all objects in FWLog that map to the Victim Set



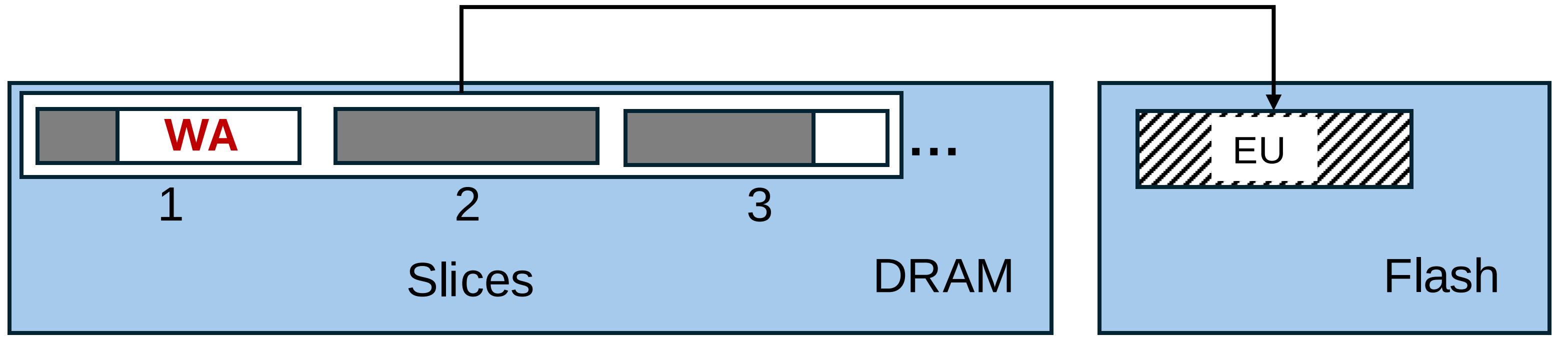
Hot-cold separation within sets

- Split 1 set (8KB) into 2 subset (4KB)
  - Every 1 *nest packing* rewrite Hot subset
  - Every n (5) *nest packing* both rewrite Hot/Cold subset



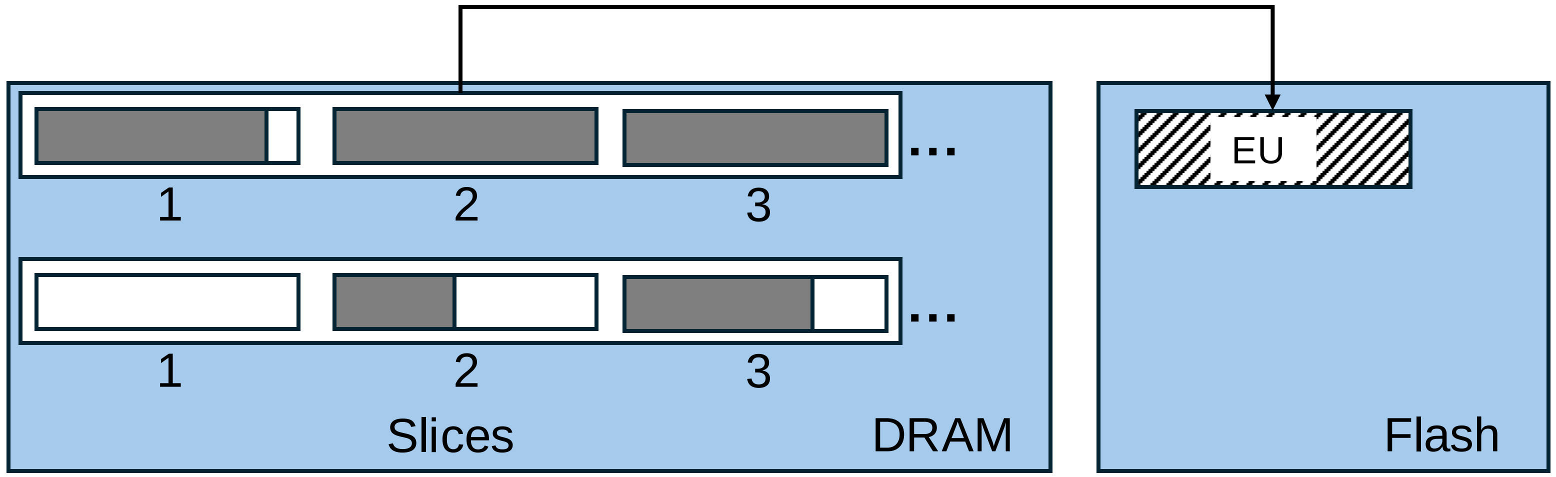
Double buffering

- 1 EU support 64 slices can cause fragmentation



Double buffering

- 1 EU support 64 slices can cause fragmentation
- FWLog reduces fragmentation via double buffering



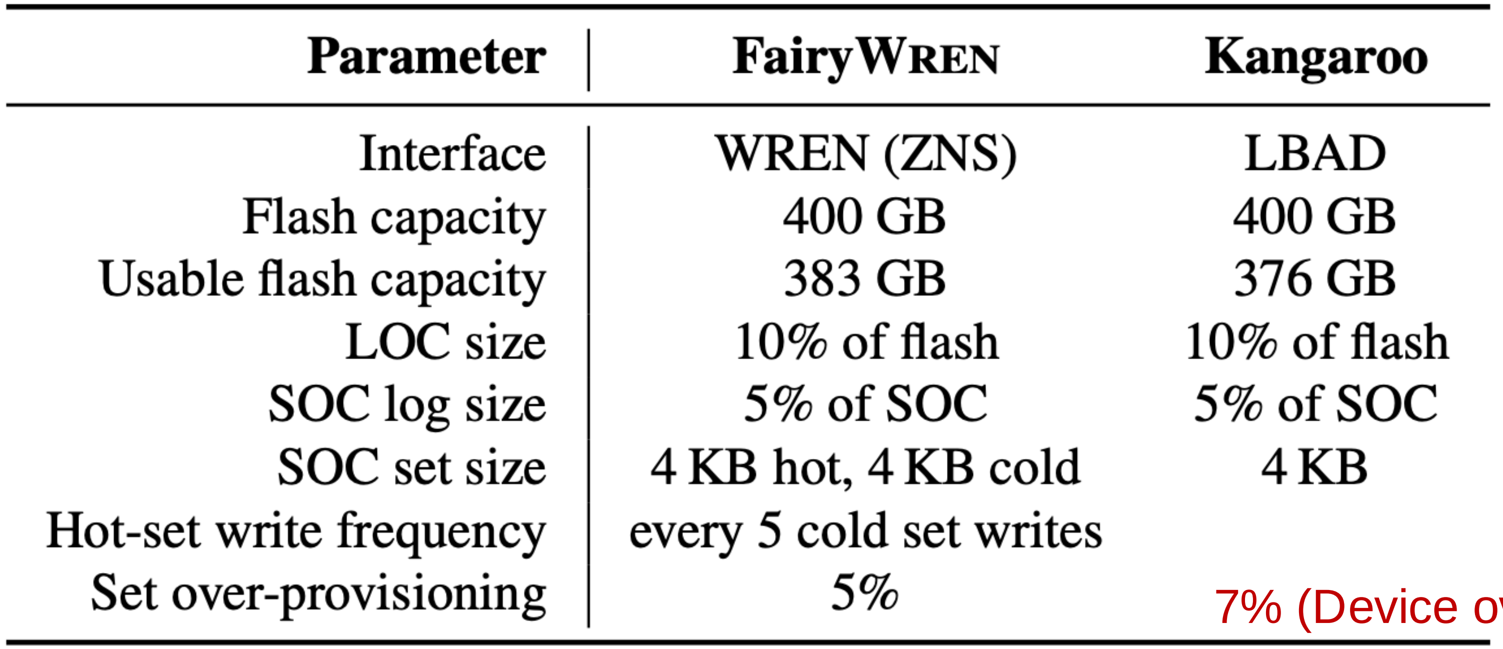
Contents

Evaluation

- Setup:
  - 2 16-core Intel Xeon CPU E5-2698
  - 64GB DRAM
  - Western Digital Ultrastar DC ZN540 1 TB ZNS SSD
    - 1077MB EU size
    - 3.5 DWPD for 5 years
- Baseline:
  - Kangaroo deployed on LBAD with similar parameters
- Workloads:
  - 21-day trace from **Meta**: 95.2% of requests < 2KB
  - 7-day trace from **Twitter**: >99% of requests < 2KB

FairyWREN & Kangaroo config

- Both caches use 400 GB of flash capacity and achieve similar miss ratios as Kangaroo's production experiments



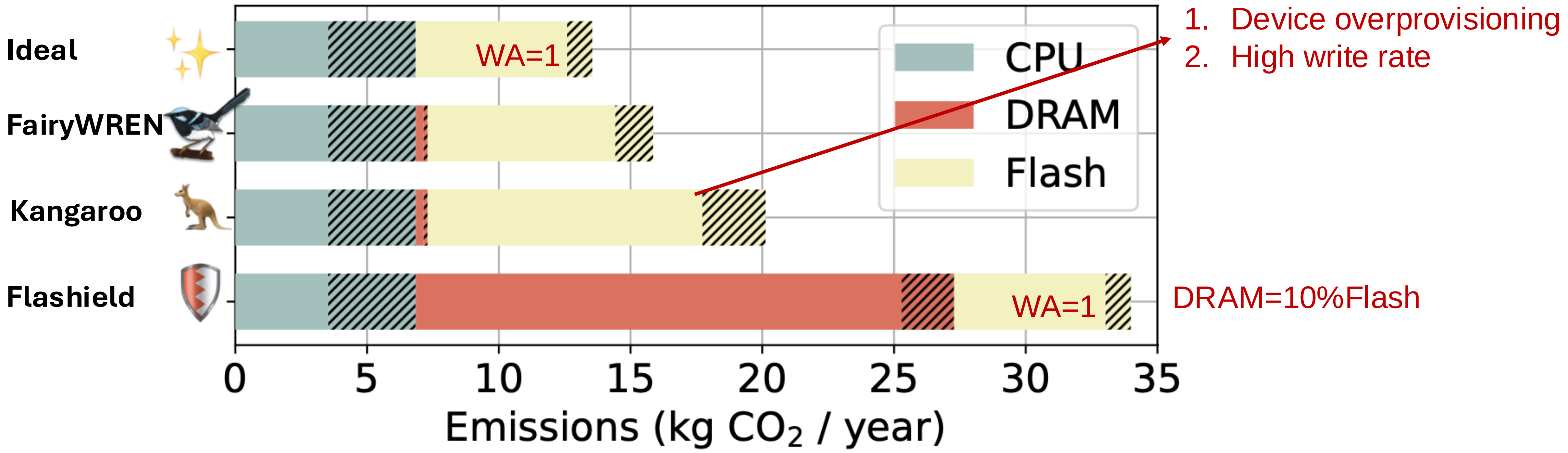
Parameter	FairyWREN	Kangaroo
Interface	WREN (ZNS)	LBAD
Flash capacity	400 GB	400 GB
Usable flash capacity	383 GB	376 GB
LOC size	10% of flash	10% of flash
SOC log size	5% of SOC	5% of SOC
SOC set size	4 KB hot, 4 KB cold	4 KB
Hot-set write frequency	every 5 cold set writes	
Set over-provisioning	5%	7% (Device overprovisioning)

Carbon emissions and cost model

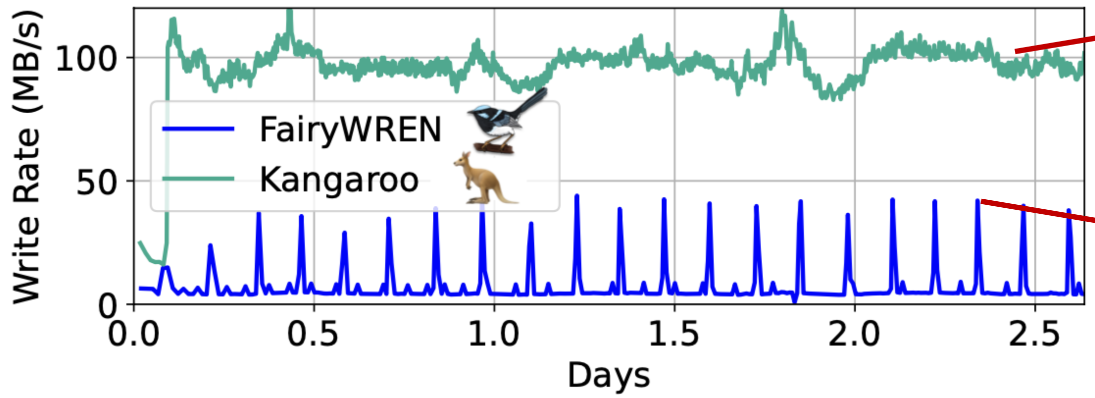
- Flash have the same cost and emissions per cell
- ACT model (ISCA'22) for operational and embodied emissions from CPUs, DDR4 DRAM, and flash

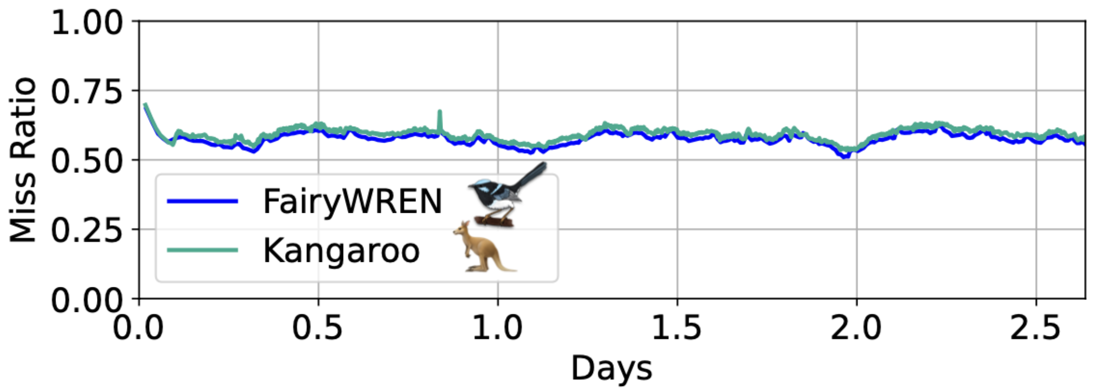
Carbon emissions of flash caches

- Carbon emissions of 6-year deployment 30% miss ratio target on a ***Twitter*** trace



On-flash experiments

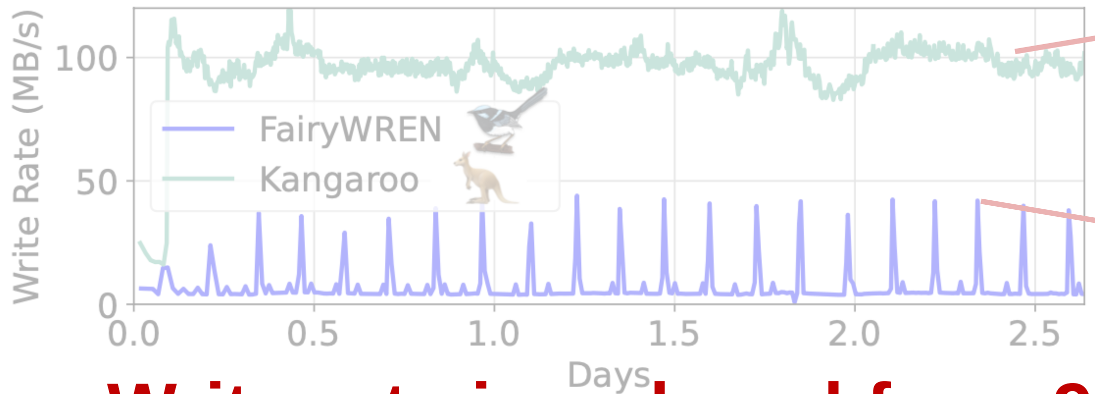




write rate appears smooth as it flushes more frequently, at **256 KB** granularity

FairyWREN performs nest packing at the granularity of an EU, **~1 GB**

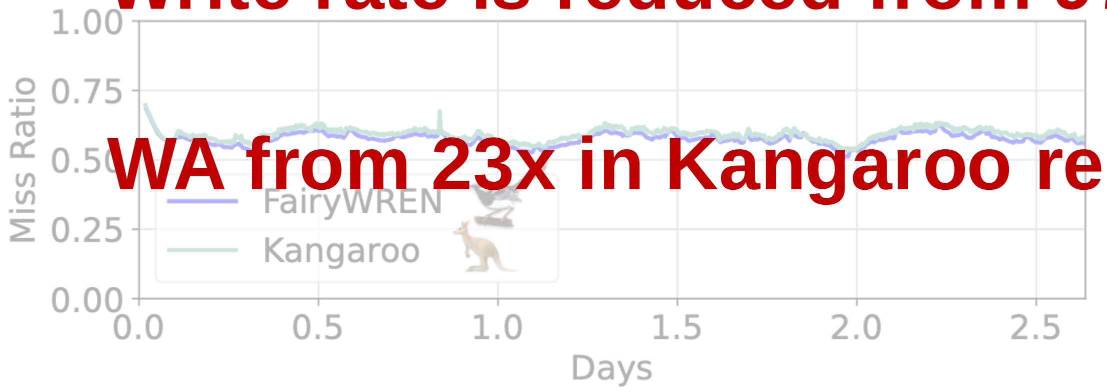
On-flash experiments



write rate appears smooth as it flushes more frequently, at 256 KB granularity

FairyWREN performs nest packing at the granularity of an EU, ~1 GB

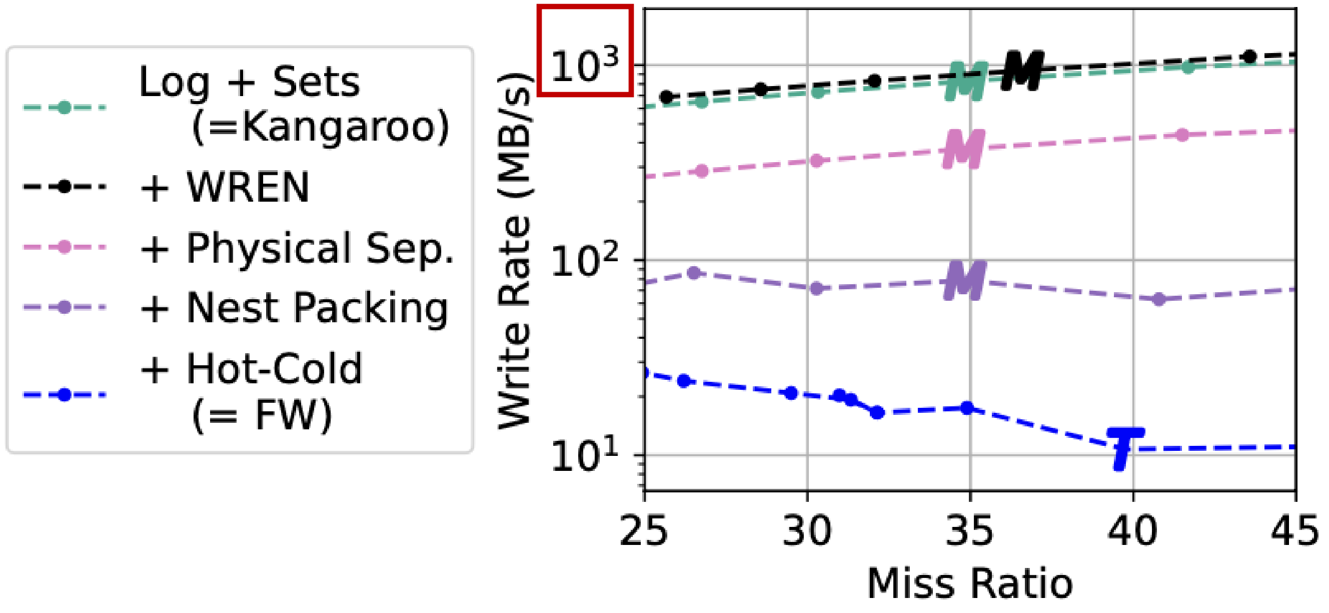
Write rate is reduced from 97 MB/s to 7.8 MB/s.

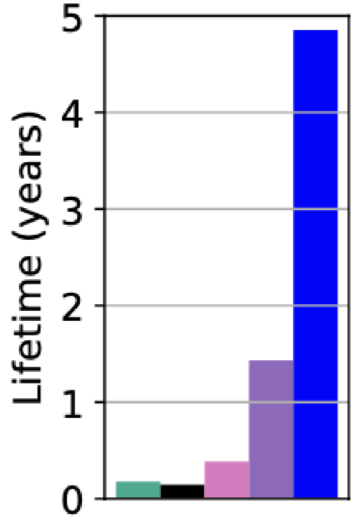


WA from 23x in Kangaroo reduce to 1.89x in FairyWREN

FairyWREN breaking down

- Log + Sets
- +WREN: Kangaroo naively to WREN
- +Physical Sep. (separate LOC and SOC)
- +Nest Packing
- +Hot-Cold (Hot-Cold Sets)



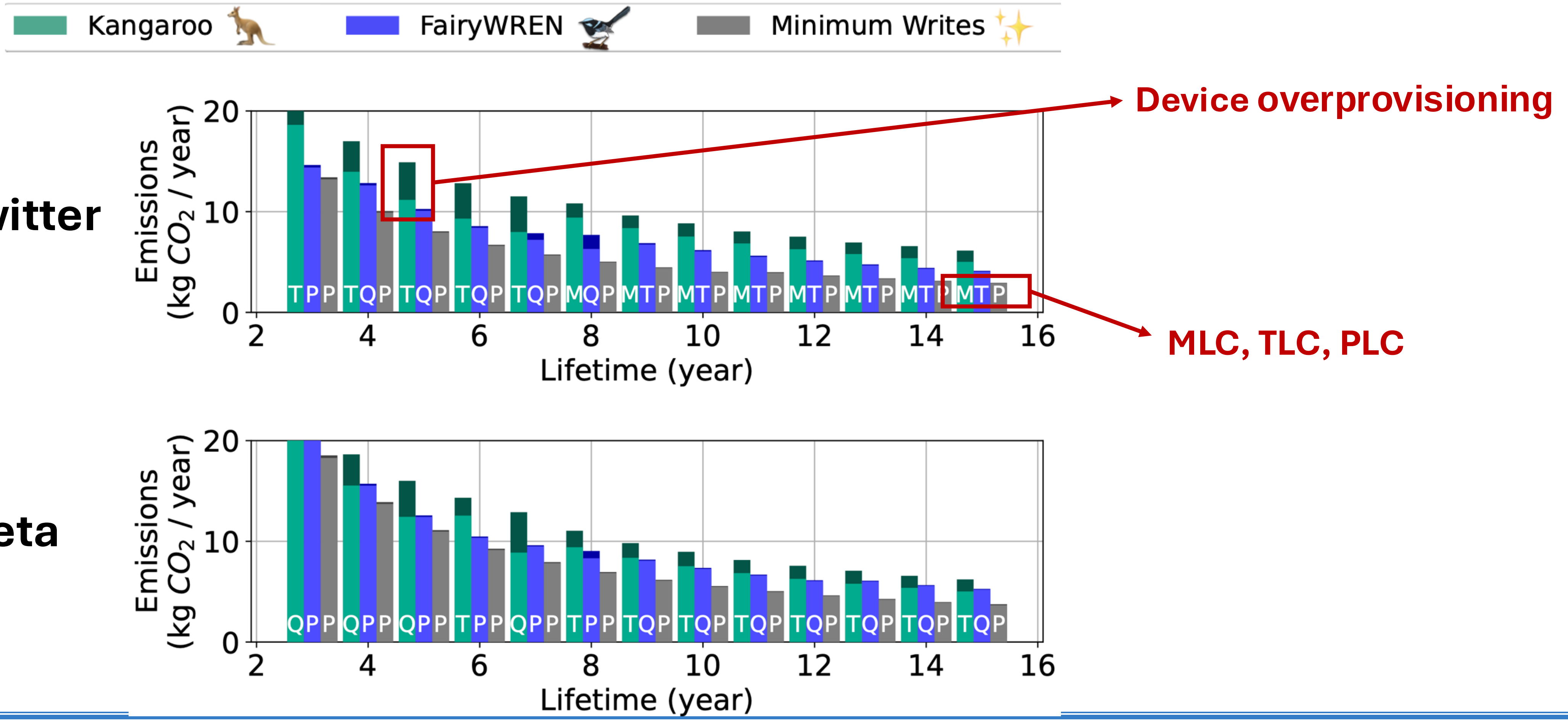


Lifetime deployed on QLC drive

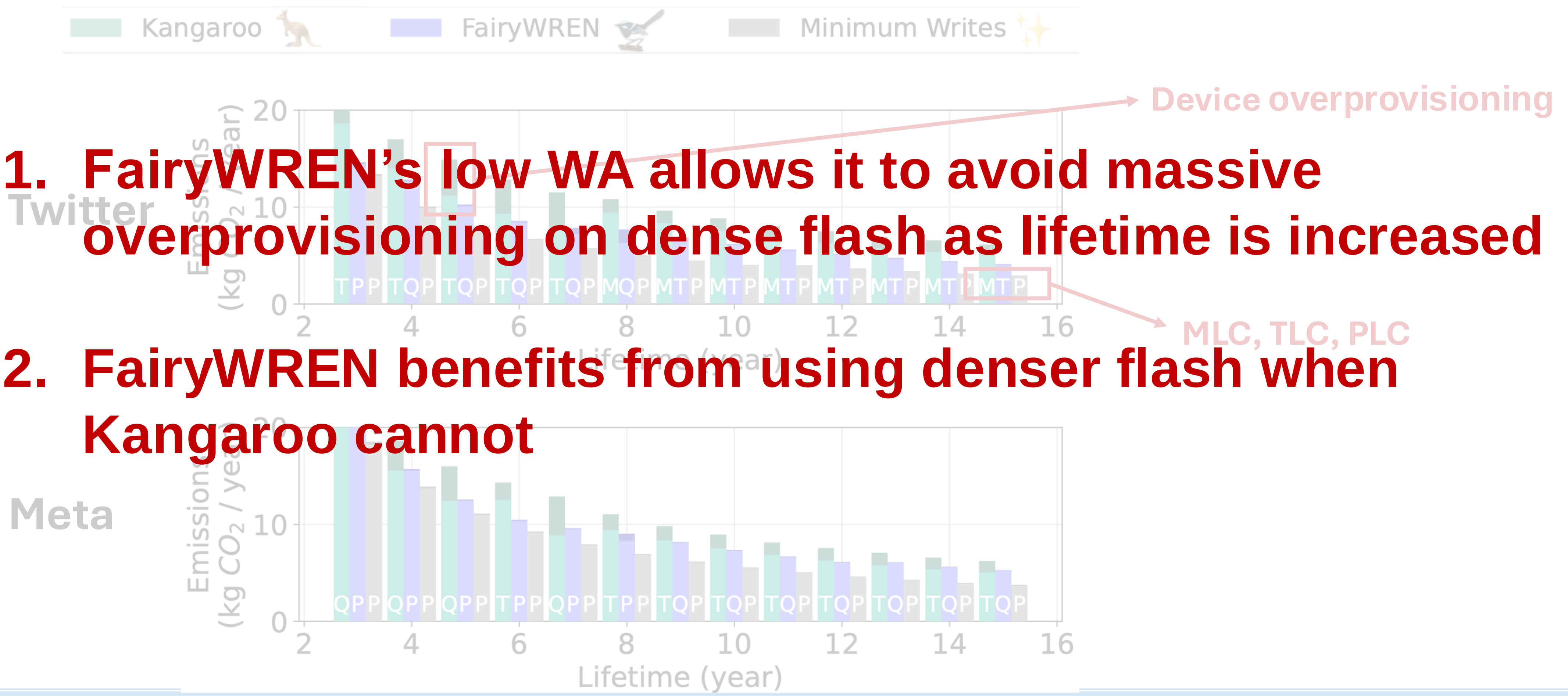
Carbon emissions simulation

Twitter

Meta



Carbon emissions simulation



1. **FairyWREN's low WA allows it to avoid massive overprovisioning on dense flash as lifetime is increased**

2. **FairyWREN benefits from using denser flash when Kangaroo cannot**

Thanks