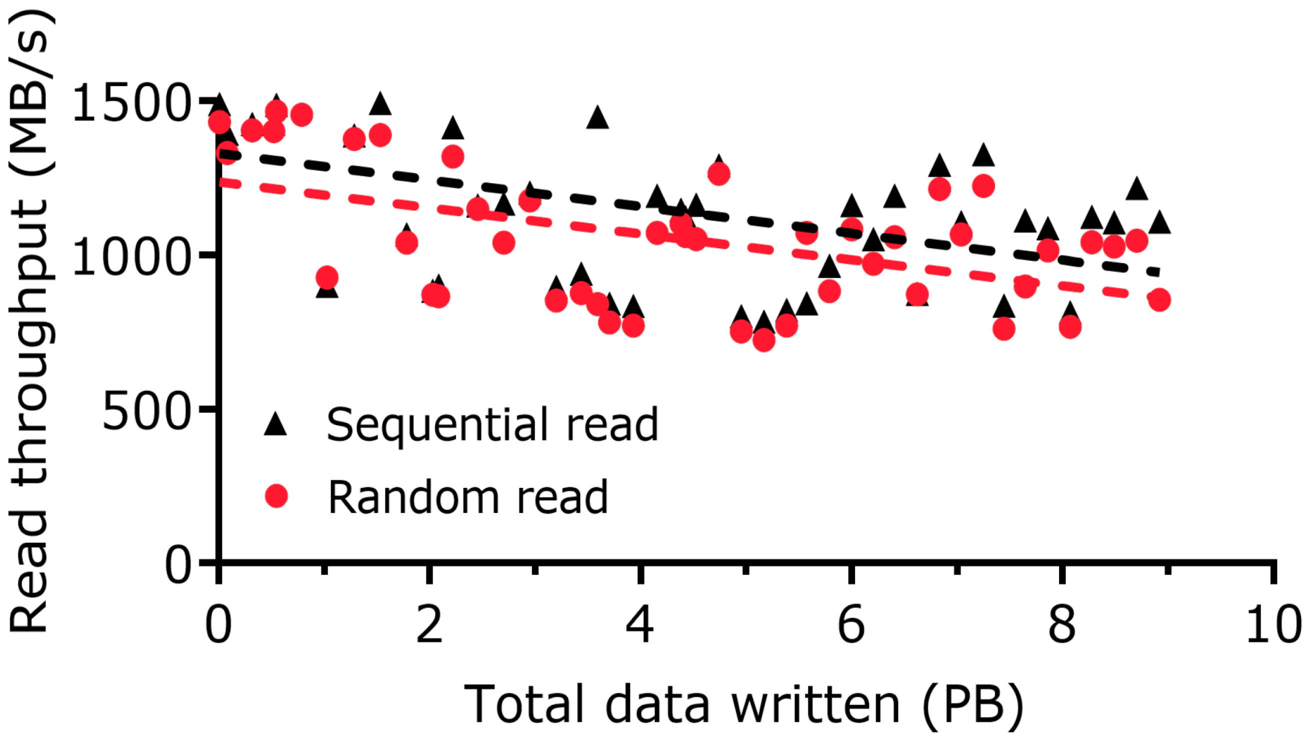
The Design and Implementation of a Capacity-Variant Storage System

Ziyang Jiao[1], Xiangqun Zhang[1], Hojin Shin[2], Jongmoo Choi[2], Bryan S. Kim[1]
[1]Syracuse University, [2]Dankook University

2024 USENIX Conference on File and Storage Technologies

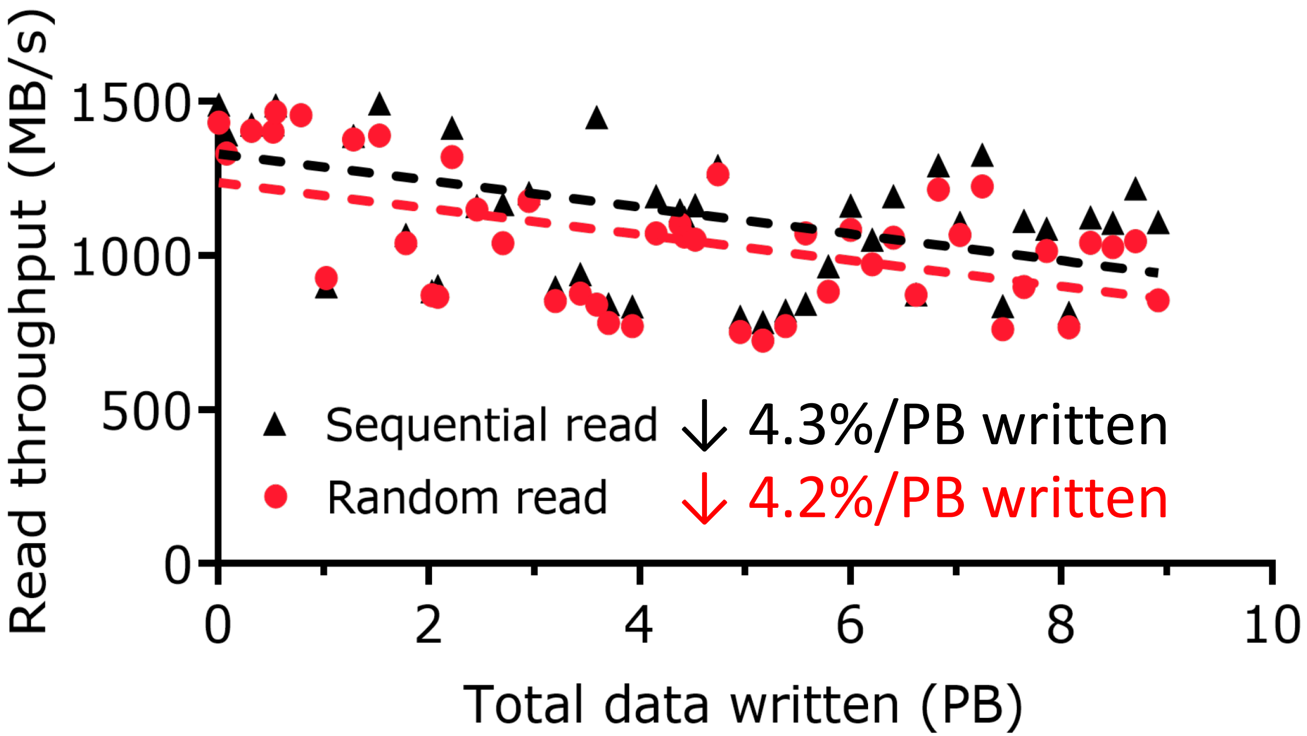
Aging on modern SSDs

- Use an enterprise-grade NVMe drive
- Age through random writes (~100 TB/day)
- Measure read-only I/O



Aging on modern SSDs

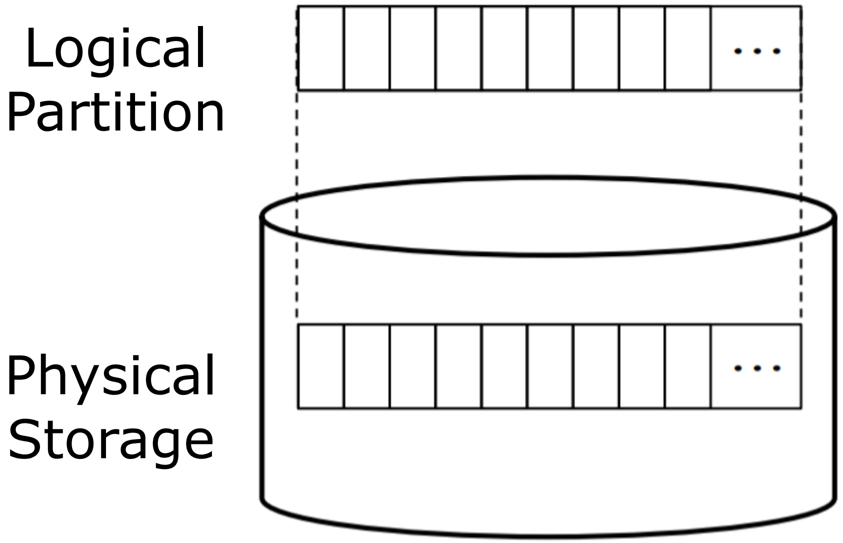
- Use an enterprise-grade NVMe drive
- Age through random writes (~100 TB/day)
- Measure read-only I/O



The current storage abstraction

- Logical capacity is fixed:
  - Assume physical capacity does not change
  - Expect a fail-stop behavior
  - Built around traditional HDDs

- Not accurate for SSDs:
  - Physical capacity naturally reduces
  - Bad blocks accumulate
  - Flash memory blocks fail partially



- Juwon Kim et al, "IPLFS: Log-Structured File System without Garbage Collection", ATC 2022

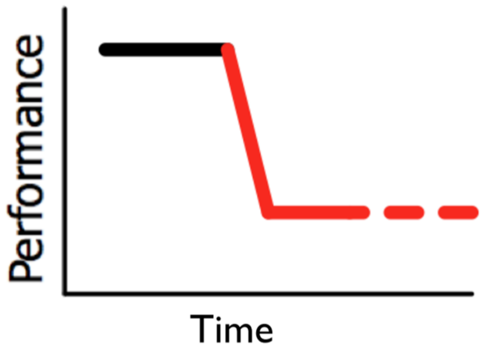
Tax from the fixed-capacity abstraction

The fixed logical capacity

+

The decreased physical capacity

=



Wear leveling & OP are required

- Maintain an illusion of a fixed-capacity device

- Haryadi S. Gunawi et al, "Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems", FAST 2018

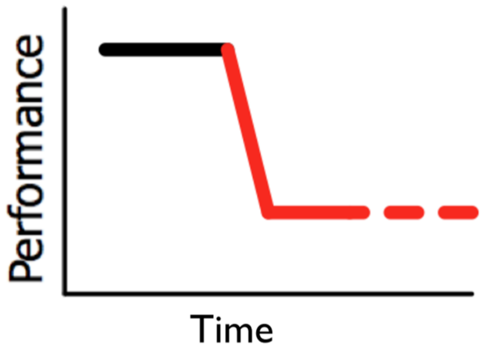
Tax from the fixed-capacity abstraction

The fixed logical capacity

+

The decreased physical capacity

=



Wear leveling & OP are required	• Maintain an illusion of a fixed-capacity device
Complicated error-handling (ECC, data re-read, redundancy...)	• Manifest the fail-slow symptom

- Haryadi S. Gunawi et al, "Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems", FAST 2018

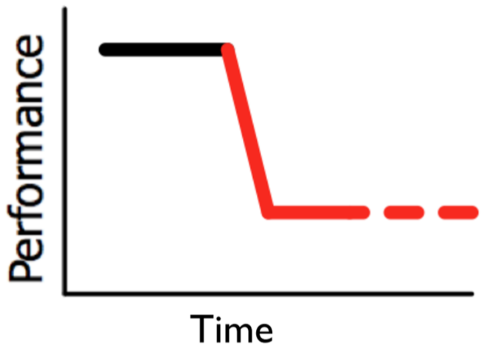
Tax from the fixed-capacity abstraction

The fixed logical capacity

+

The decreased physical capacity

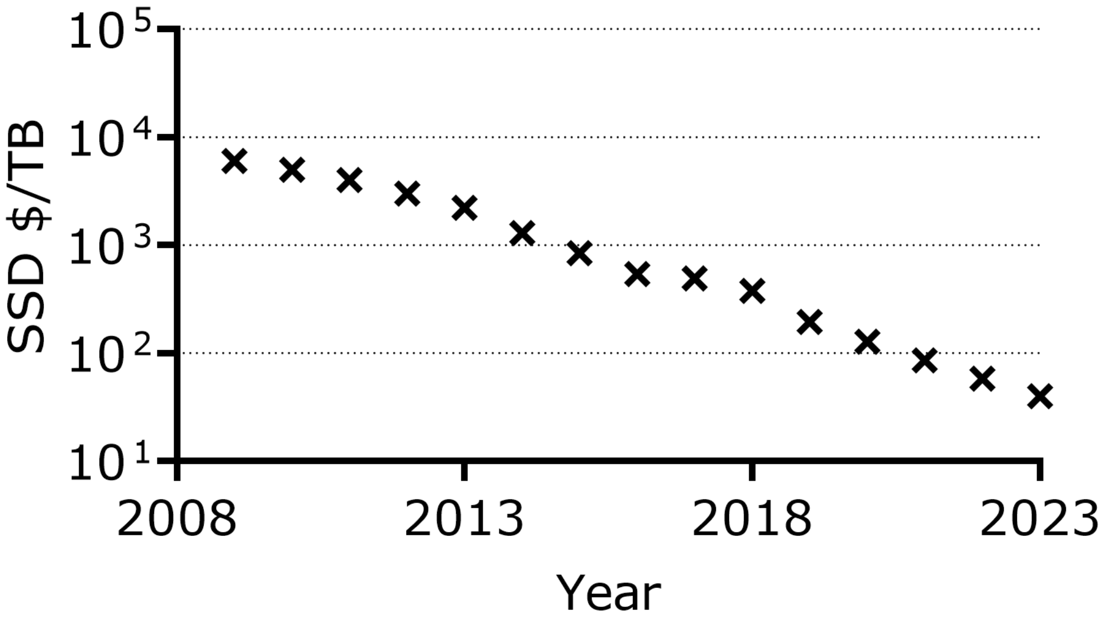
=

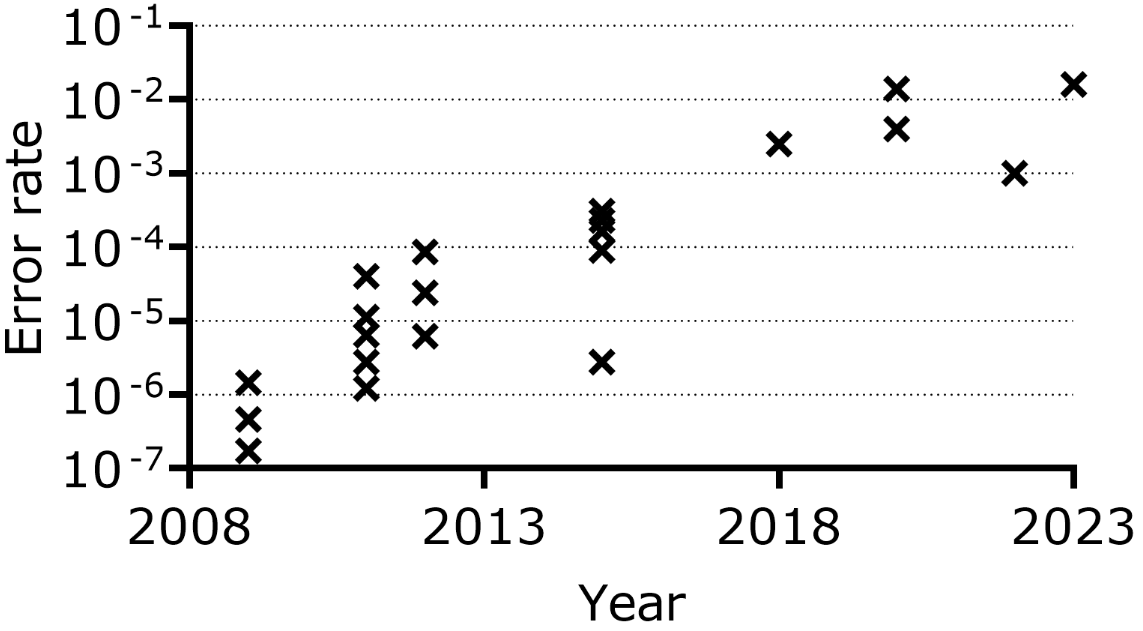


Wear leveling & OP are required	• Maintain an illusion of a fixed-capacity device
Complicated error-handling (ECC, data re-read, redundancy...)	• Manifest the fail-slow symptom
Lifetime ends early	• When exported capacity can't be maintained

- Haryadi S. Gunawi et al, "Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems", FAST 2018

The trends in SSD reliability





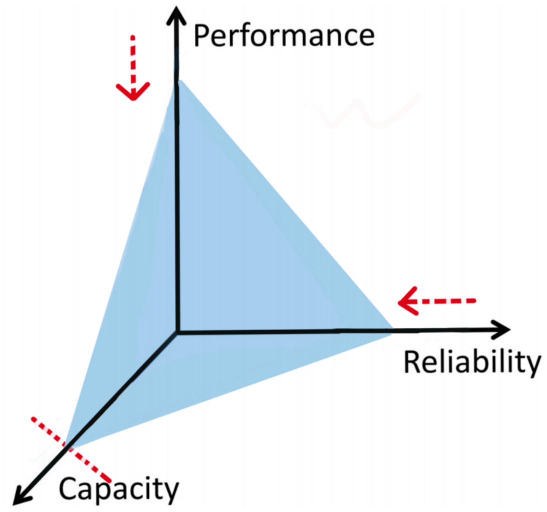
- Yajuan Du et al, "Towards LDPC Read Performance of 3D Flash Memories with Layer-induced Error Characteristics", TODAES 2023
- Seungwoo Son et al, "Differentiated Protection and Hot/Cold-aware Data Placement Policies through K-means Clustering Analysis for 3D-NAND SSDs", Electronics 2022
- Kong-Kiat Yong et al, "Error Diluting: Exploiting 3-D NAND Flash Process Variation for Efficient Read on LDPC-based SSDs", TCAD 2020
- B. Kim et al, "Design Tradeoffs for SSD Reliability", FAST 2019
- Yixin Luo et al, "HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-recovery and Temperature Awareness", HPCA 2018
- Xin Shi et al, "Program Error Rate-based Wear Leveling for NAND Flash Memory", DATE 2018
- Yu cai et al, "Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling", Proceedings of the IEEE 2017
- Yu cai et al, "Error Characterization, Mitigation, and Recovery in Flash-memory-based Solid-state Drives", DATE 2013

Outline

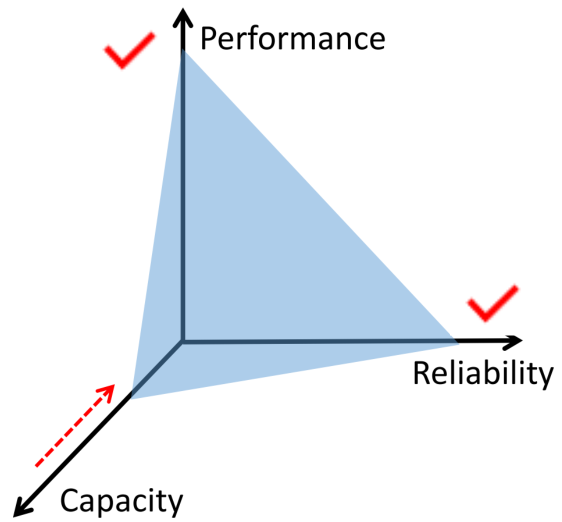
- Background & motivation
- Design principles
- Capacity-variant storage system
- Evaluation
- Summary

Design principles

- The fixed-capacity storage system
  - Trade performance & reliability for capacity

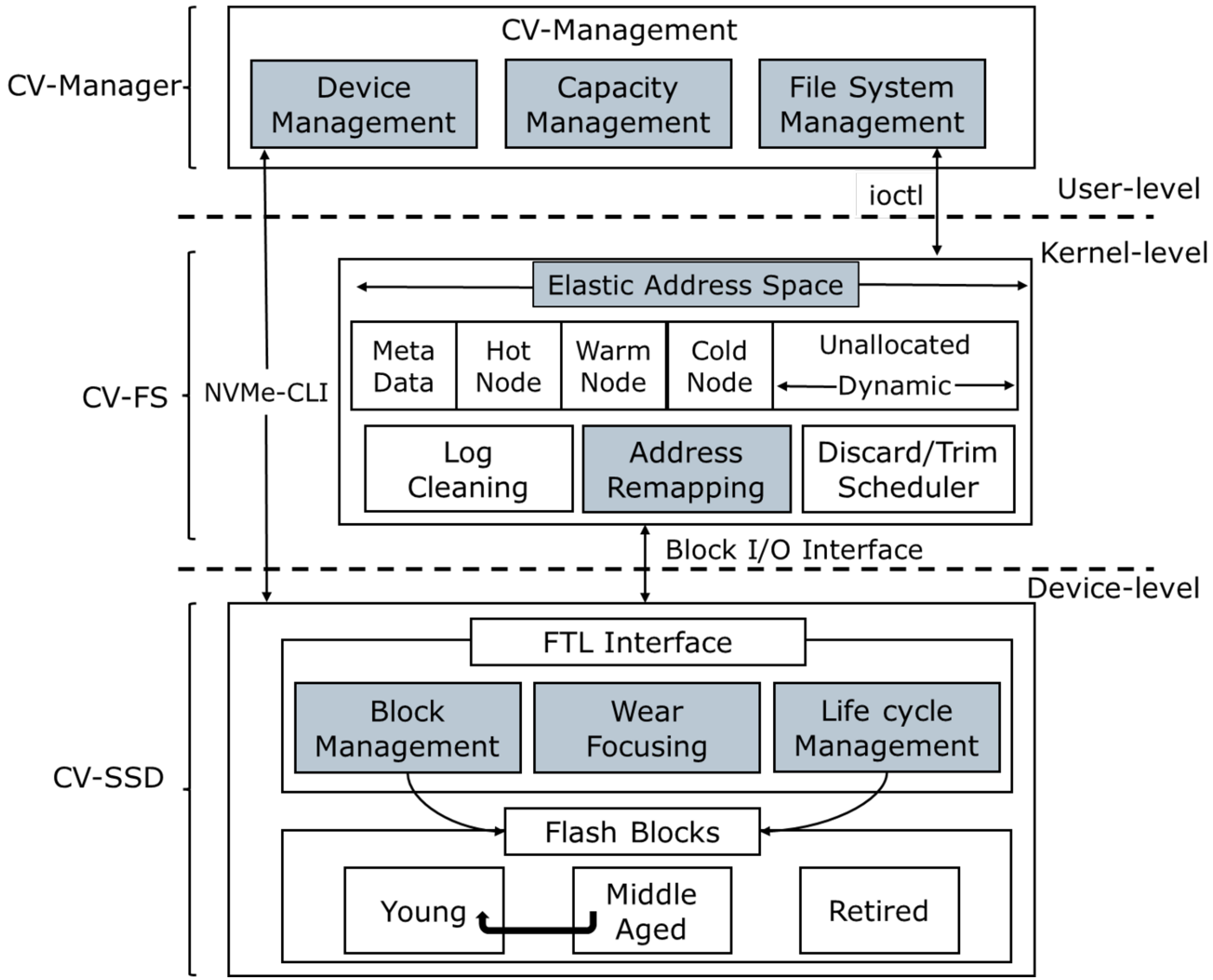


- The capacity-variant storage system
  - Trade capacity for performance & reliability

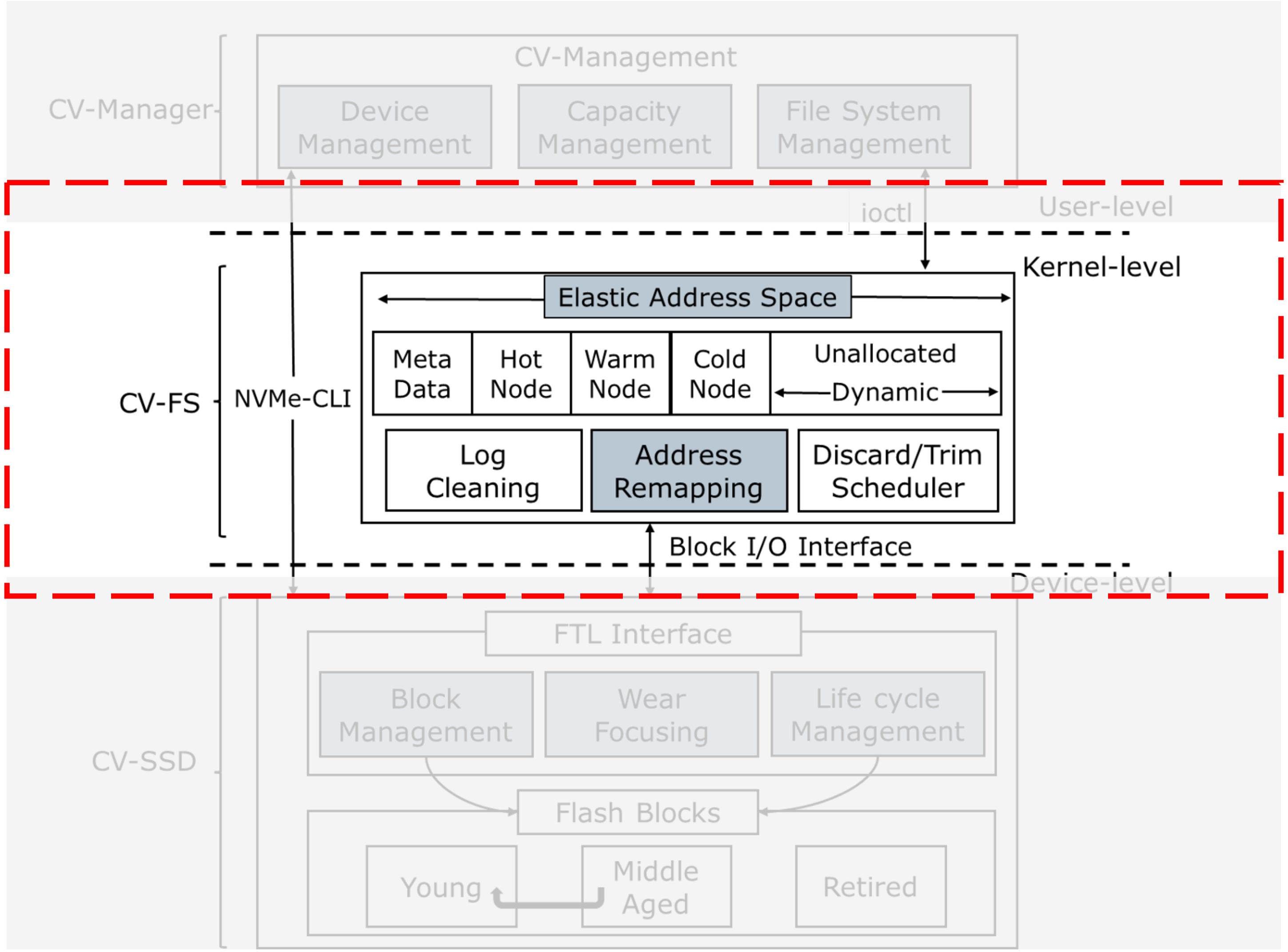


- Haryadi S. Gunawi et al, "Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems", FAST 2018
- B. Kim et al, "Design Tradeoffs for SSD Reliability", FAST 2019

CVSS overview

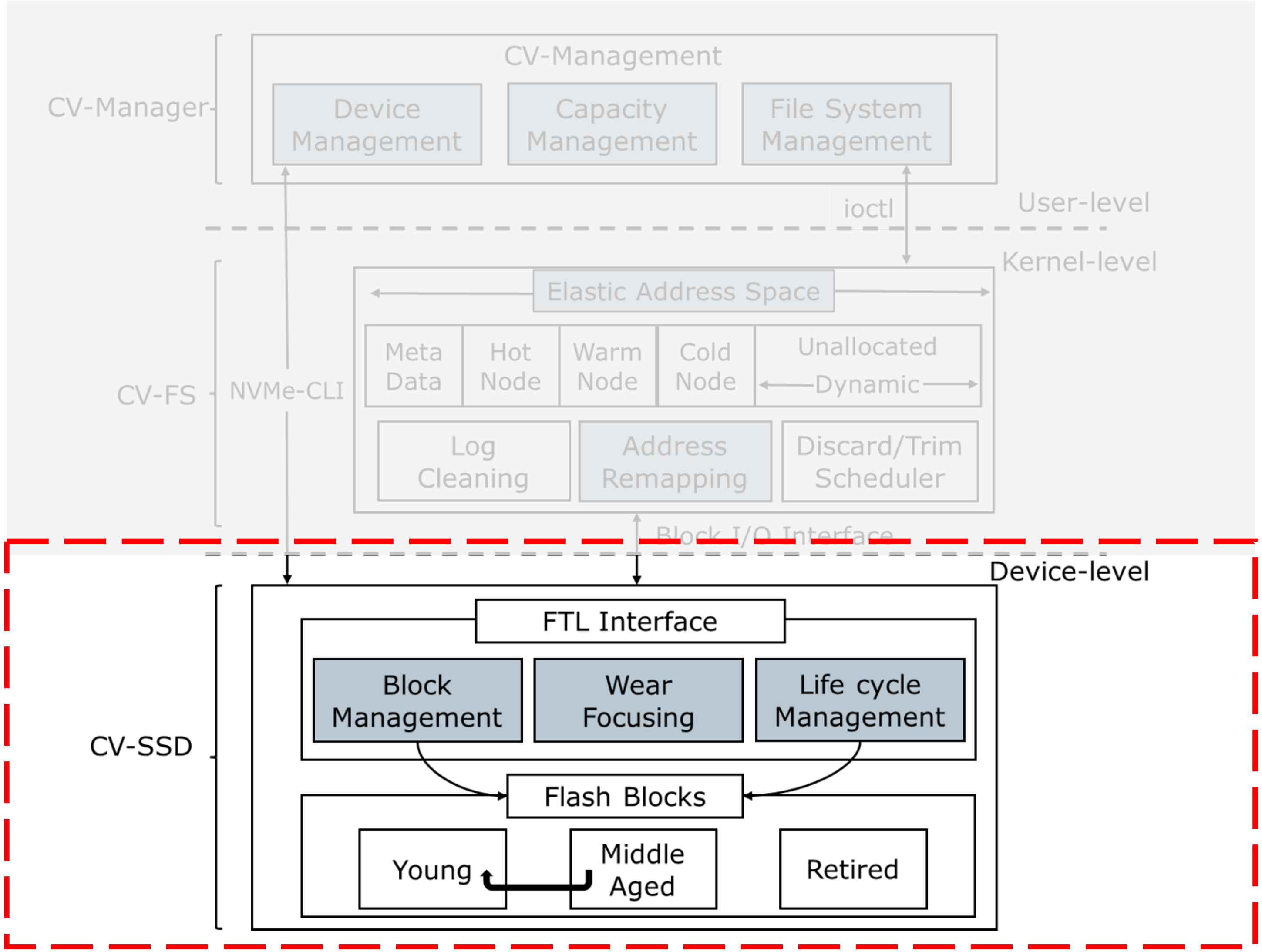


CVSS overview



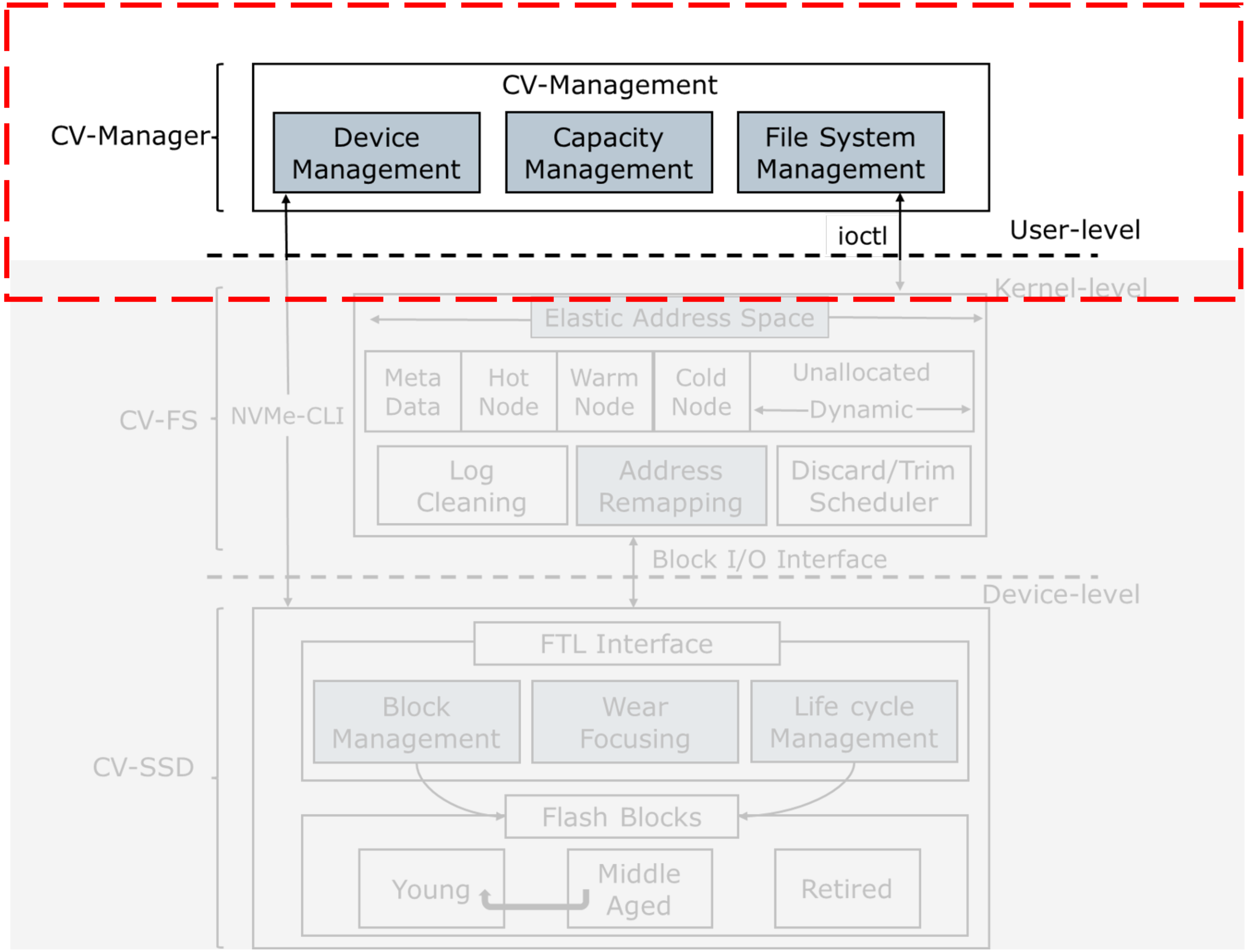
- ✓ Tune logical capacity dynamically
- ✓ Manage user data to avoid loss

CVSS overview



✓ Exclude aged blocks earlier
✓ Mitigate fail-slow symptoms

CVSS overview



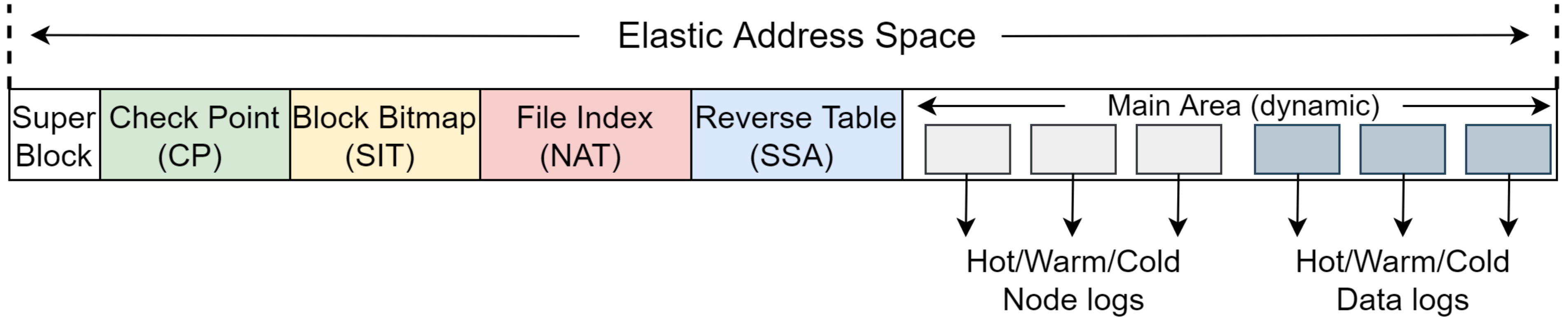
✓ Provide host interfaces
✓ Orchestrate CV-FS and CV-SSD

Outline

- Background & motivation
- Design principles
- Capacity-variant storage system
- Evaluation
- Summary

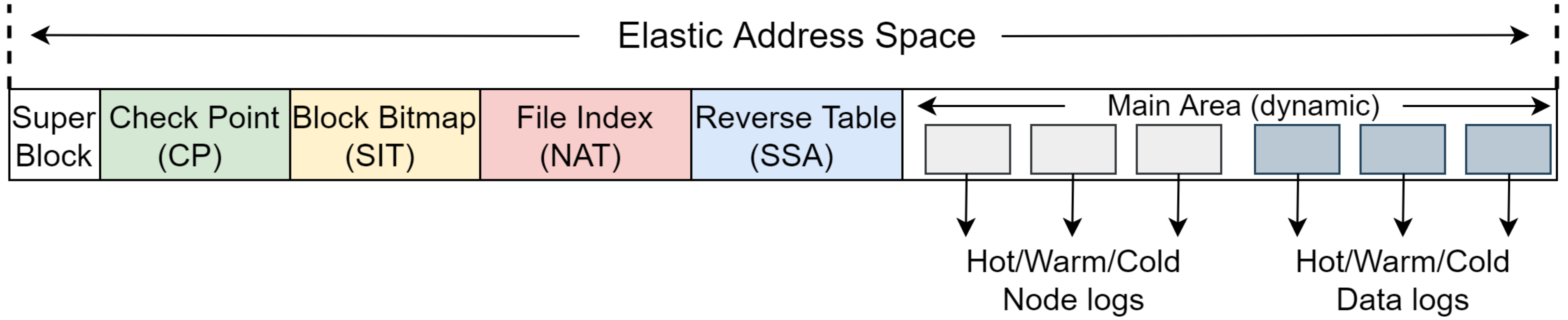
Capacity-variant FS

- Log-structured file system (e.g., f2fs)
  - Perform well on modern flash storage devices
  - Elastic address space



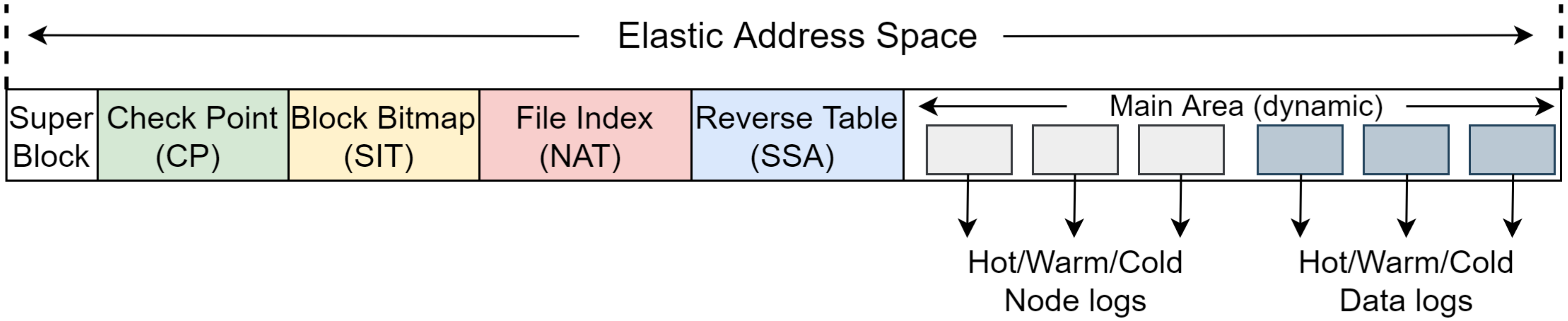
Capacity-variant FS

- Requirements for logical capacity adjustment
  1. Avoid data loss and maintain consistency



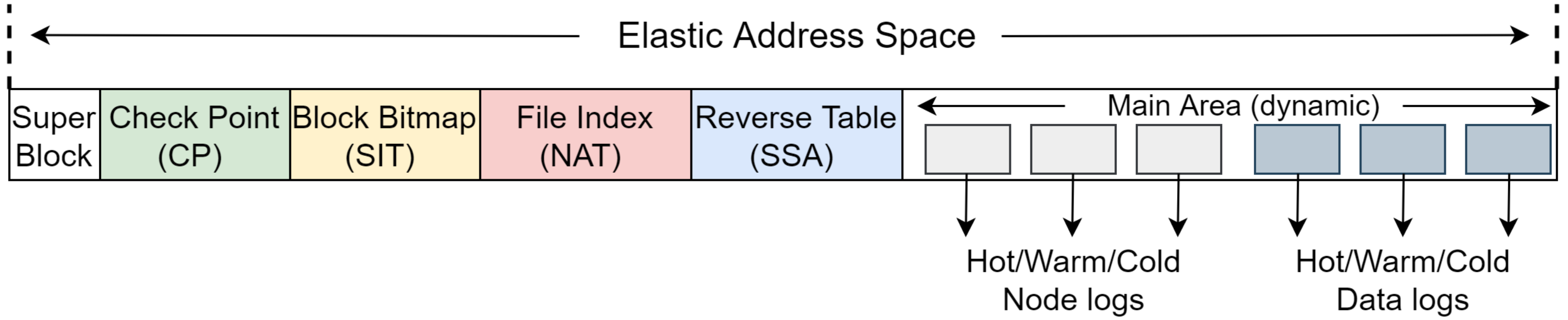
Capacity-variant FS

- Requirements for logical capacity adjustment
  1. Avoid data loss and maintain consistency
  2. Online, fine-grained adjustment



Capacity-variant FS

- Requirements for logical capacity adjustment
    1. Avoid data loss and maintain consistency
    2. Online, fine-grained adjustment
    3. Overall low overhead

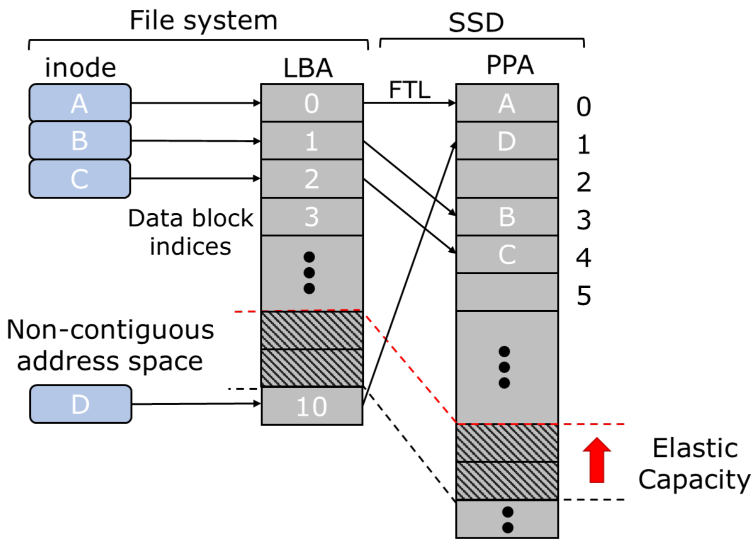


Elastic logical capacity

What are some potential approaches and tradeoffs?

- Image from Google searches

File system designs for capacity variance



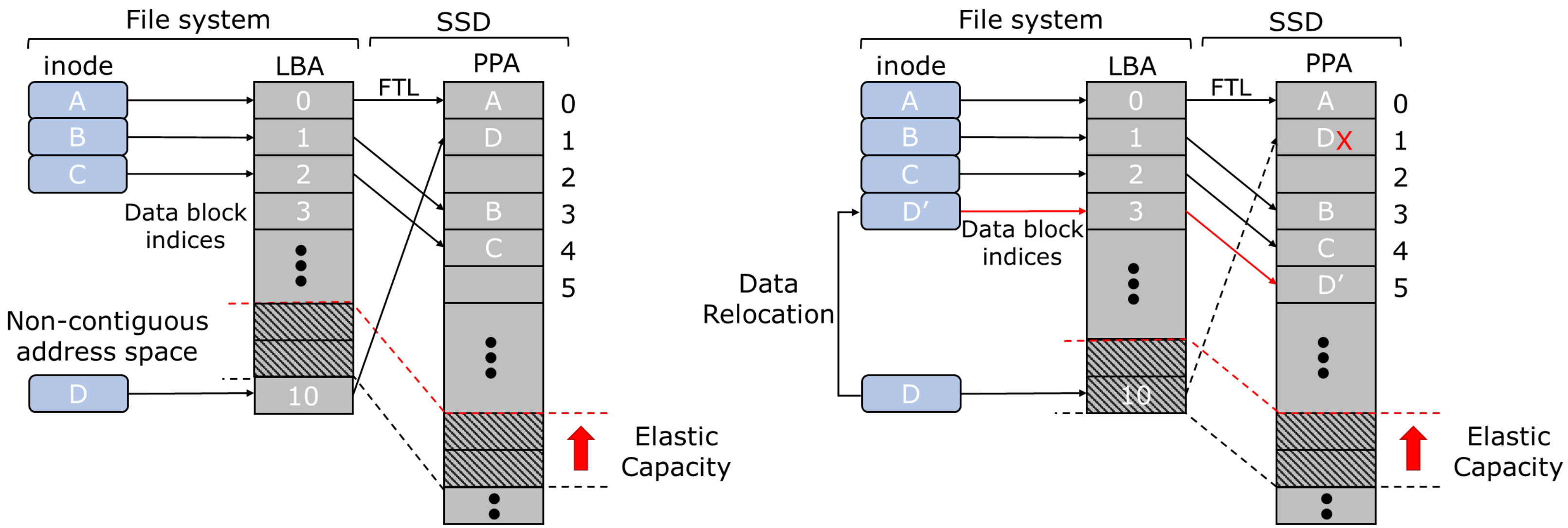
(a) Non-contiguous address space

✓ Incur lowest upfront cost

✗ Fragment address space

✗ Increase LFSs cleaning overhead

File system designs for capacity variance



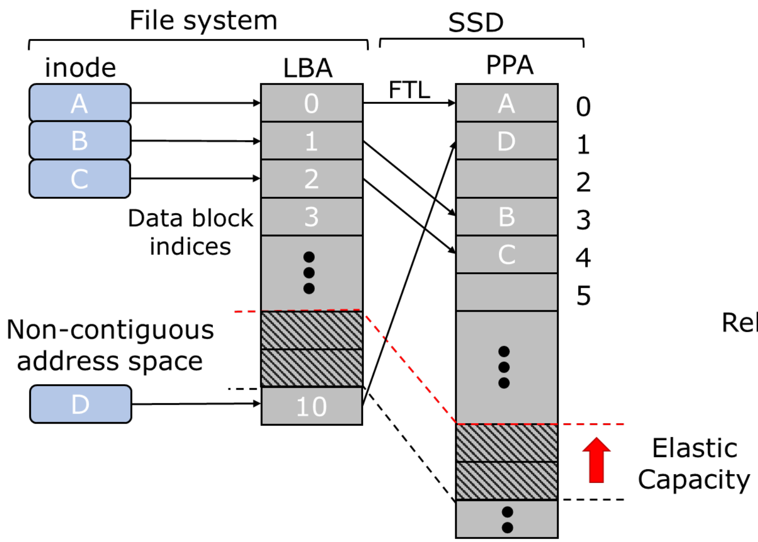
(a) Non-contiguous address space

✓ Incur lowest upfront cost
✗ Fragment address space
✗ Increase LFSs cleaning overhead

(b) Data relocation

✓ Maintain address space contiguity
✗ Exert additional write on the SSD
✗ Stall user requests

File system designs for capacity variance

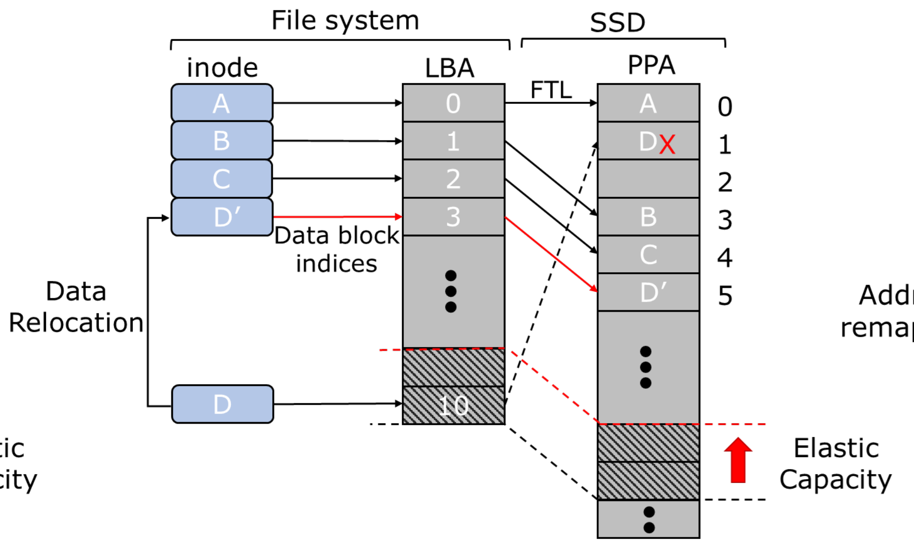


(a) Non-contiguous address space

✓ Incur lowest upfront cost

✗ Fragment address space

✗ Increase LFSs cleaning overhead

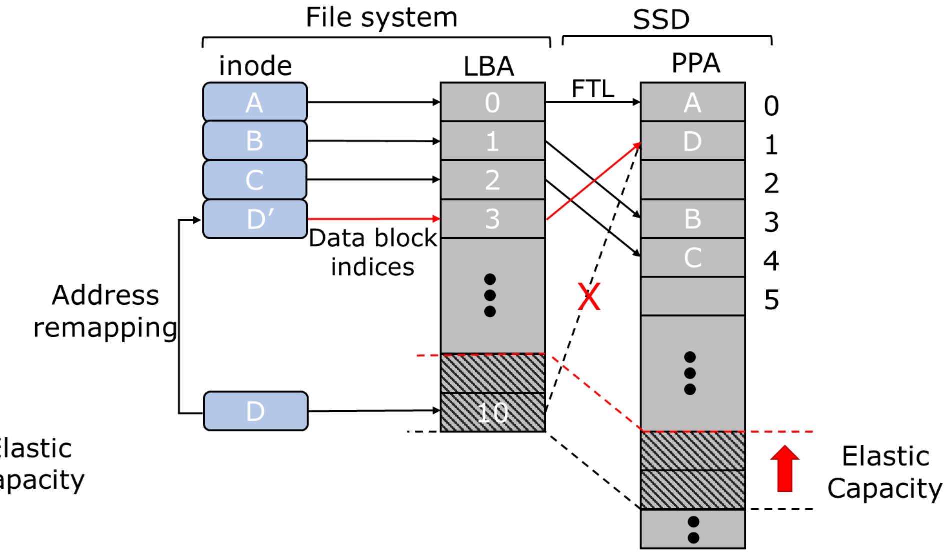


(b) Data relocation

✓ Maintain address space contiguity

✗ Exert additional write on the SSD

✗ Stall user requests



(c) Address remapping

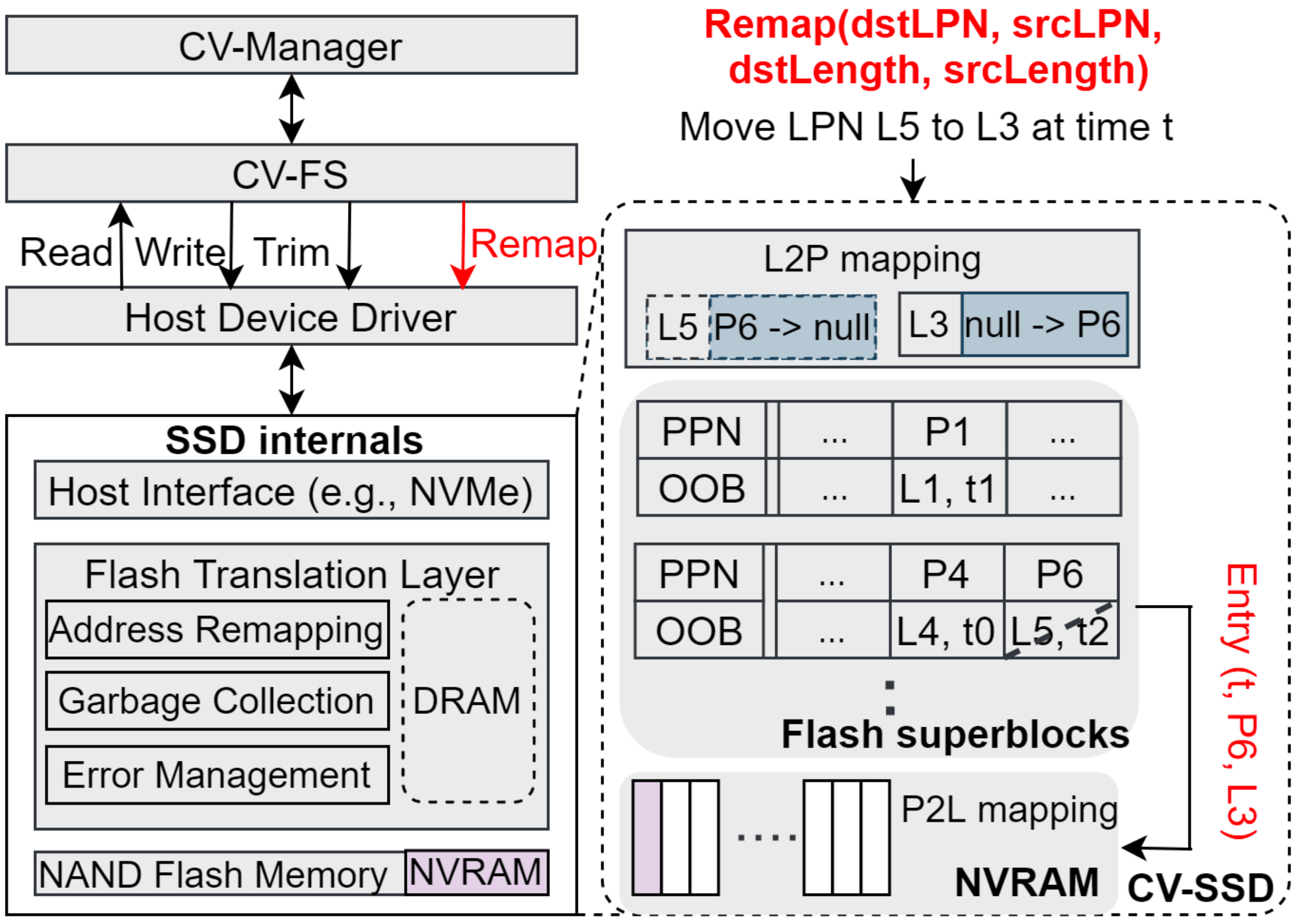
✓ Maintain address space contiguity

✓ Negligible system overhead

? Require a special SSD command

Interface changes for capacity variance

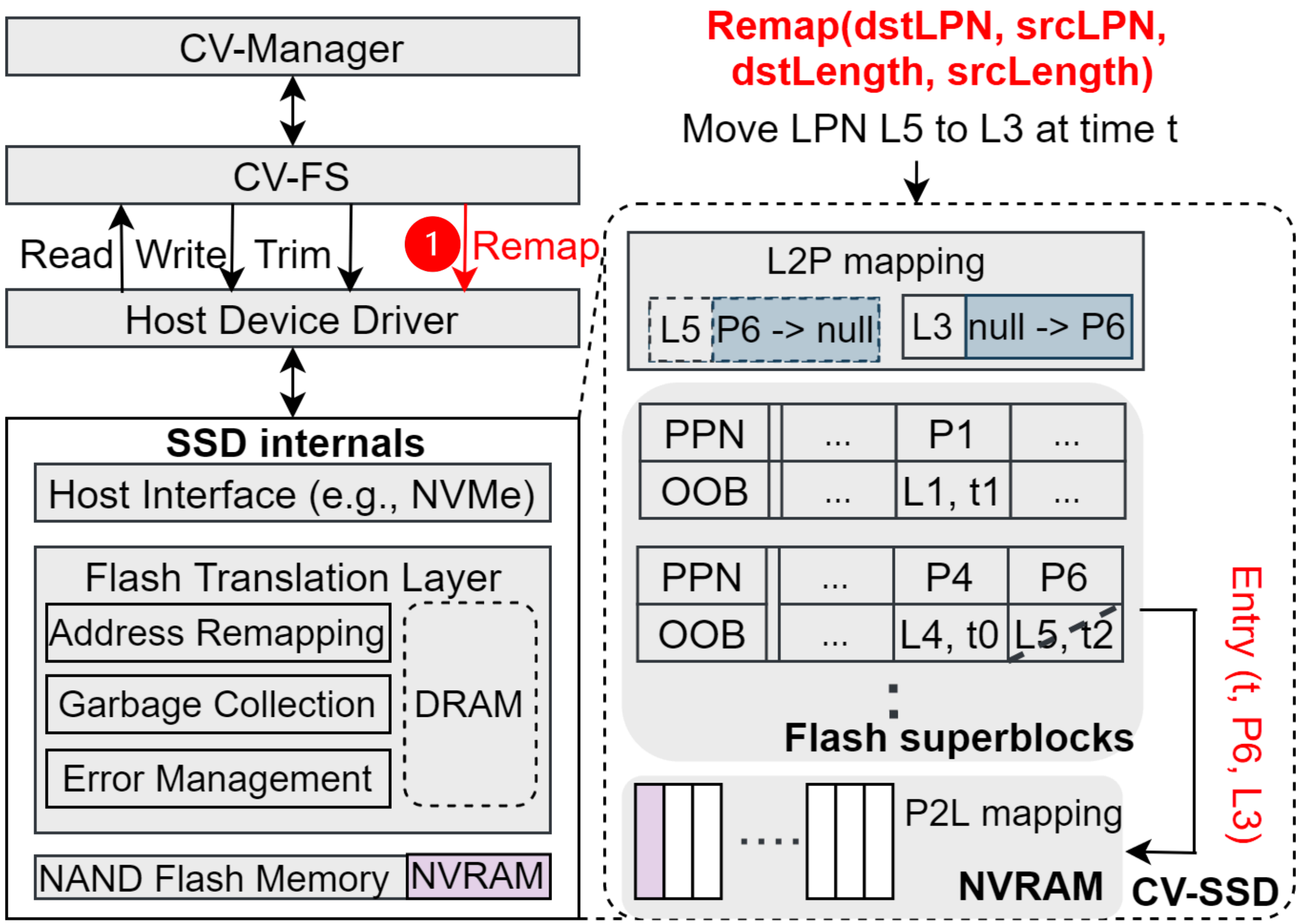
- Remap (*dstLPN, srcLPN, dstLength, srcLength*)
- Associate data from *srcLPN + srcLength - 1* to *dstLPN*
- *dstLength* is optionally used to ensure I/O alignment.



- You Zhou et al, “Remap-SSD: Safely and Efficiently Exploiting SSD Address Remapping to Eliminate Duplicate Writes”, FAST 2021

Interface changes for capacity variance

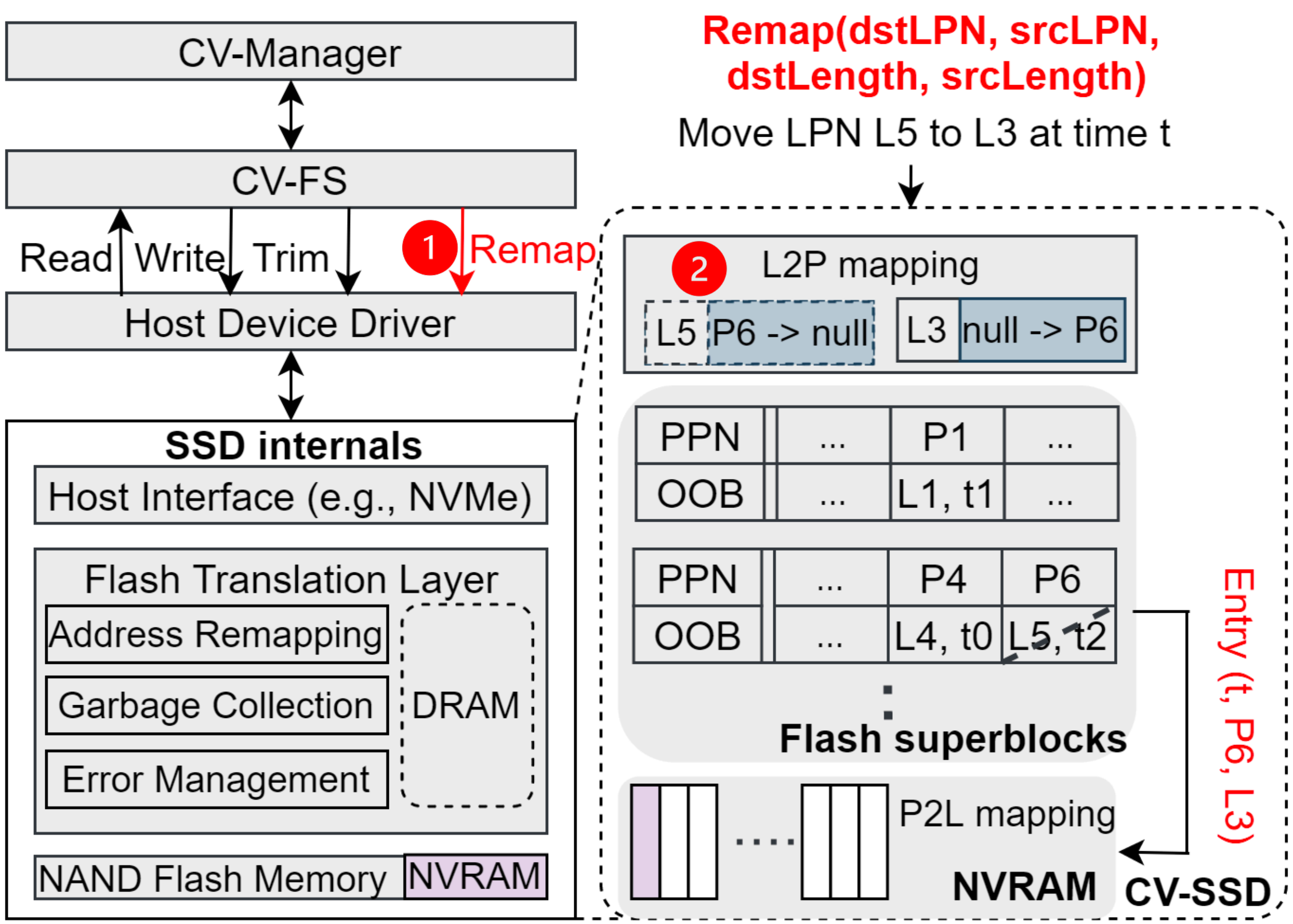
- Remap (L3, L5, 1, 1):
  1. Issued by CV-FS



- You Zhou et al, "Remap-SSD: Safely and Efficiently Exploiting SSD Address Remapping to Eliminate Duplicate Writes", FAST 2021

Interface changes for capacity variance

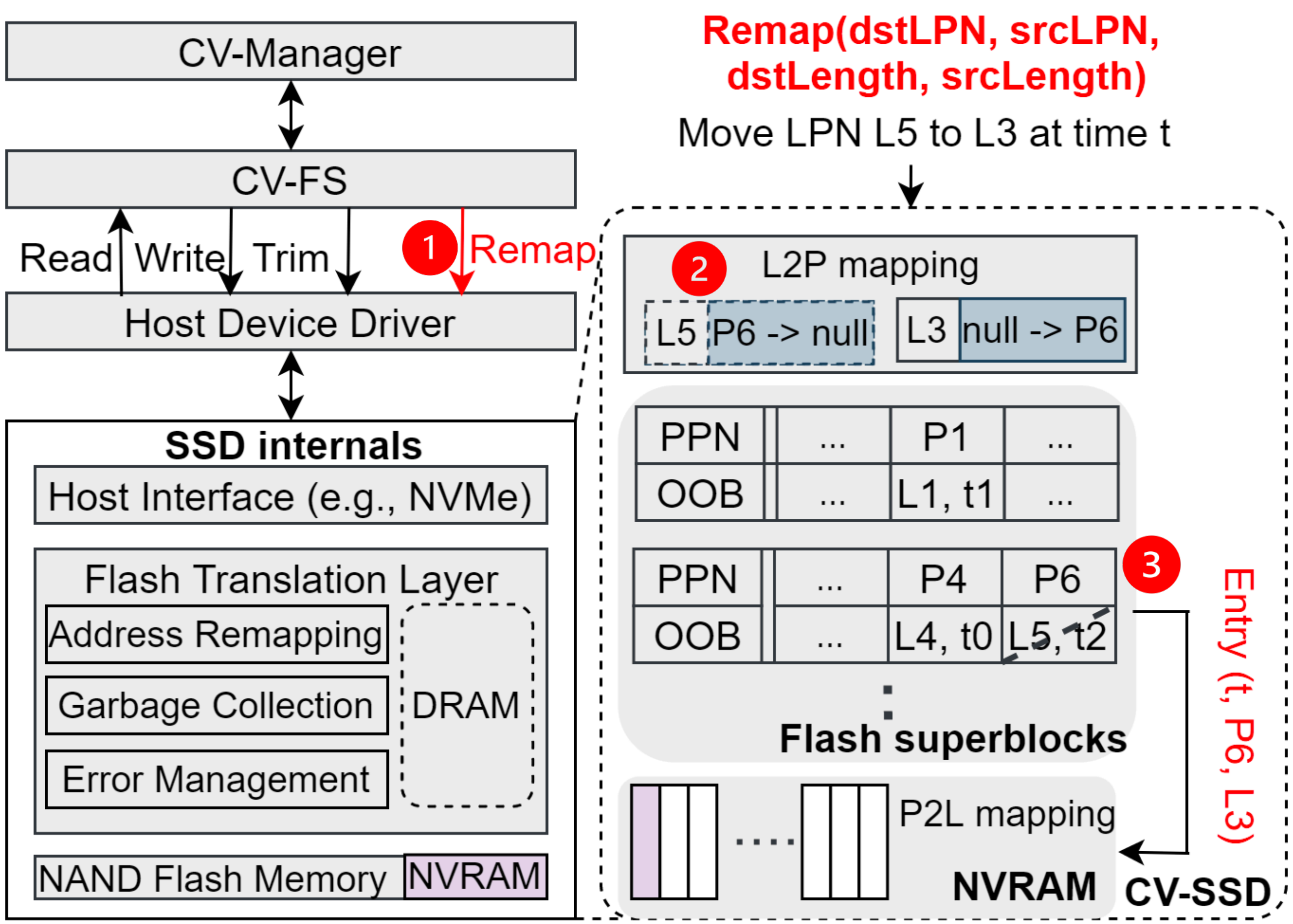
- Remap (L3, L5, 1, 1):
  1. Issued by CV-FS
  2. Access L2P mapping: L5 → P6



- You Zhou et al, "Remap-SSD: Safely and Efficiently Exploiting SSD Address Remapping to Eliminate Duplicate Writes", FAST 2021

Interface changes for capacity variance

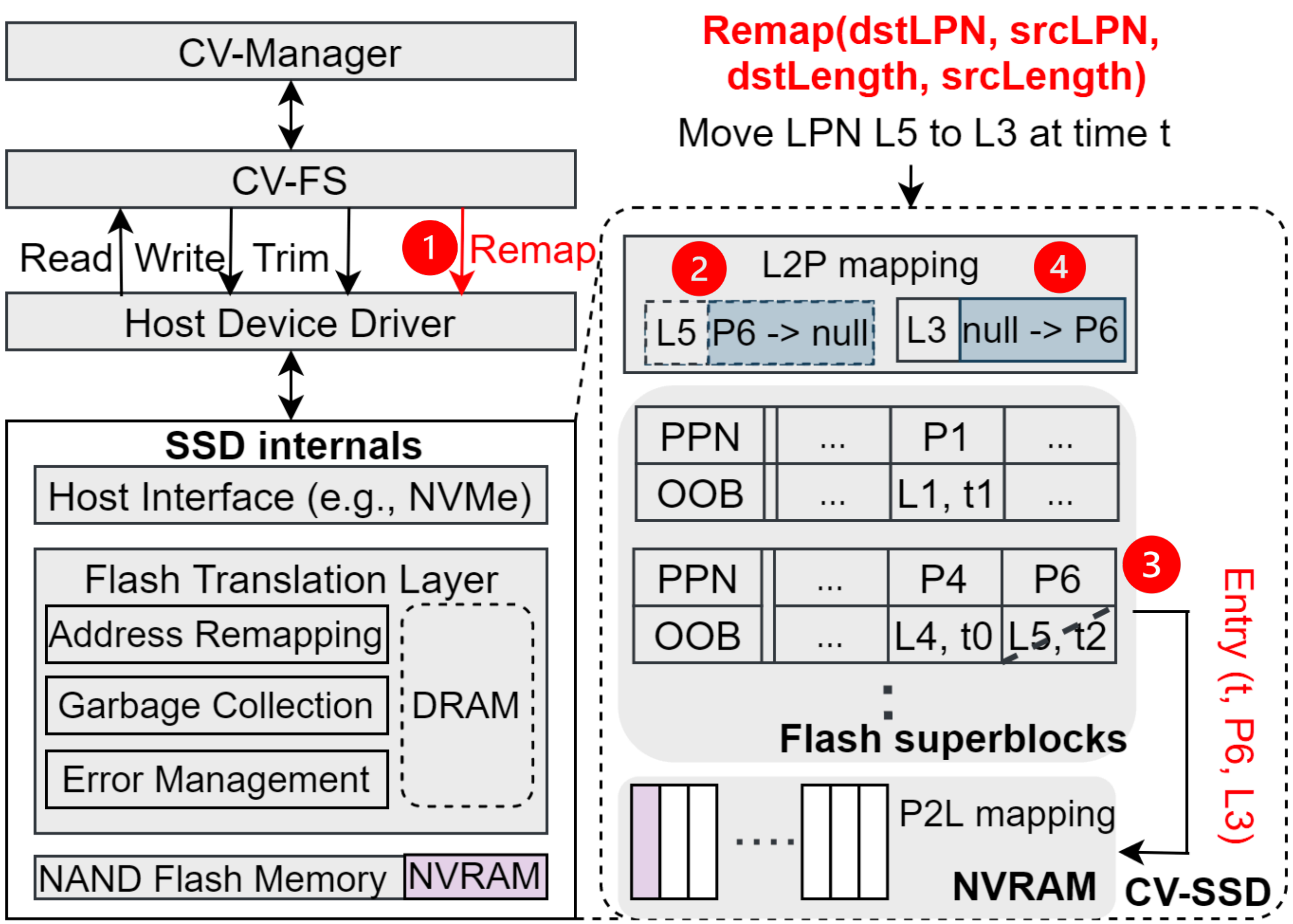
- Remap (L3, L5, 1, 1):
  1. Issued by CV-FS
  2. Access L2P mapping: L5 → P6
  3. Check OOB of P6: validation



- You Zhou et al, "Remap-SSD: Safely and Efficiently Exploiting SSD Address Remapping to Eliminate Duplicate Writes", FAST 2021

Interface changes for capacity variance

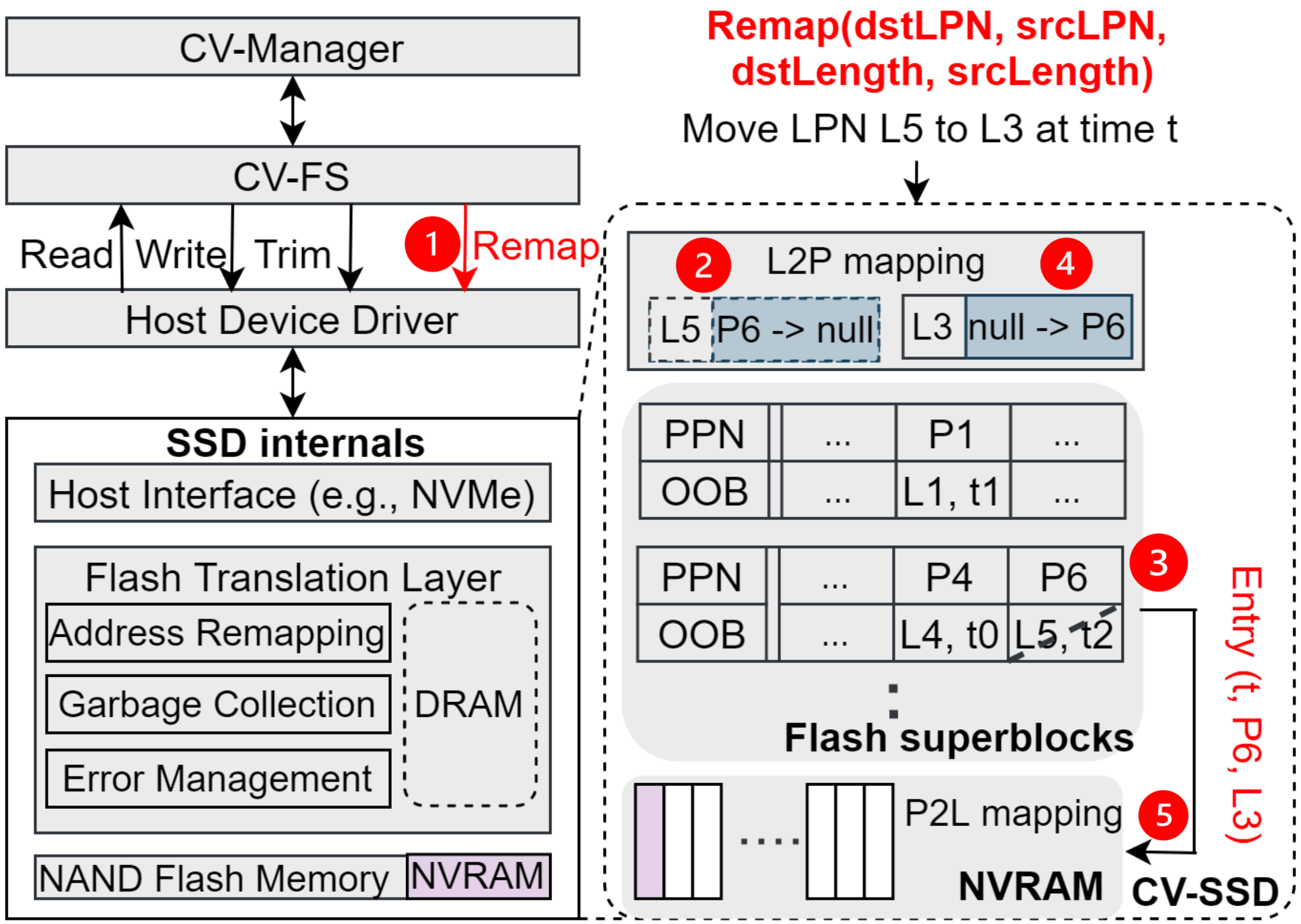
- Remap (L3, L5, 1, 1):
  1. Issued by CV-FS
  2. Access L2P mapping: L5 → P6
  3. Check OOB of P6: validation
  4. Update L2P mapping: L3 → P6



- You Zhou et al, "Remap-SSD: Safely and Efficiently Exploiting SSD Address Remapping to Eliminate Duplicate Writes", FAST 2021

Interface changes for capacity variance

- Remap (L3, L5, 1, 1):
  1. Issued by CV-FS
  2. Access L2P mapping: L5 → P6
  3. Check OOB of P6: validation
  4. Update L2P mapping: L3 → P6
  5. Update P2L mapping: P6 → L3

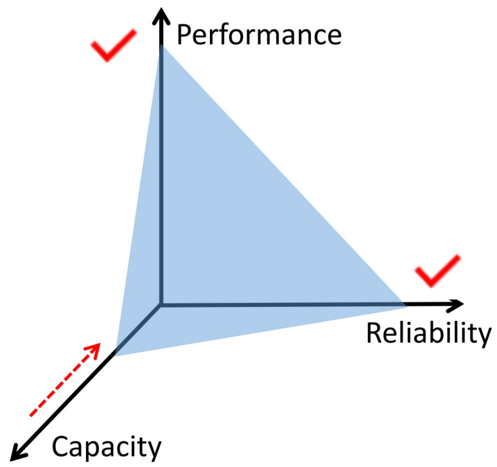


- You Zhou et al, "Remap-SSD: Safely and Efficiently Exploiting SSD Address Remapping to Eliminate Duplicate Writes", FAST 2021

Capacity-variant SSD

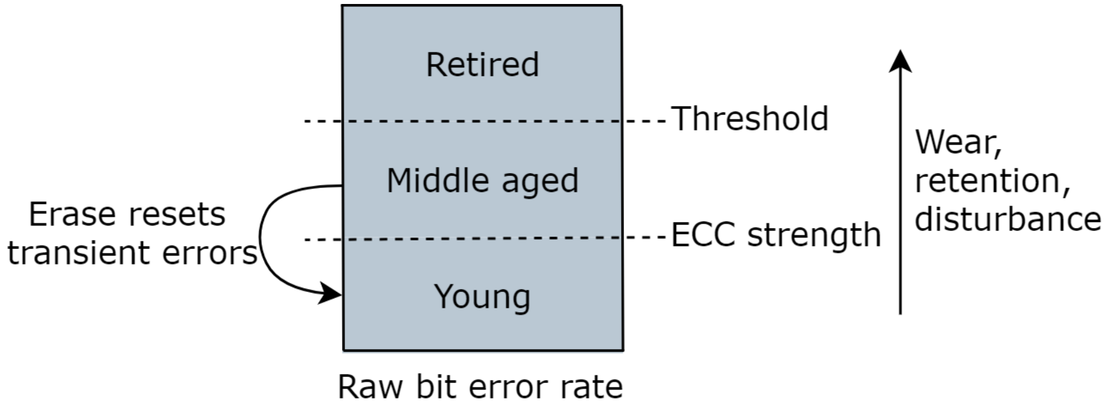
- Goal:
  - Maintain performance even when aged
  - Allow user-defined performance
  - Achieve a better capacity-performance-reliability (CPR) tradeoff

- Approaches:
  - Block management
  - Wear focusing
  - Life cycle management



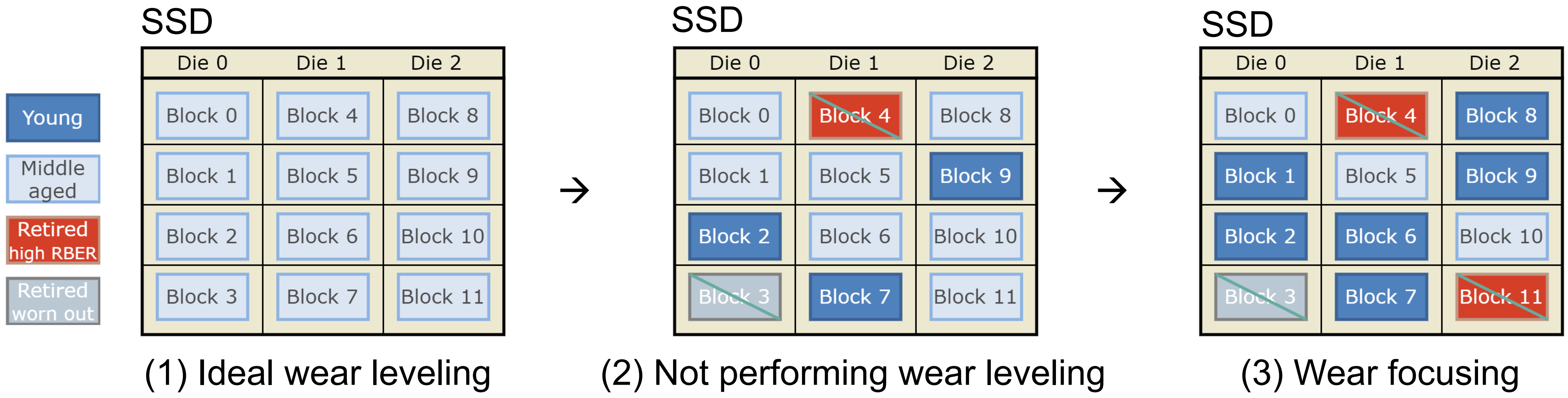
Block management

- Define blocks based on the aging states:
  - Young blocks: RBER <= ECC strength → Performant
  - Middle-aged blocks: ECC strength < RBER < Threshold → Meet expectation
  - Retired blocks: RBER >= Threshold and Erase count > Endurance → Fall below expectation



Wear focusing

- Focus the wear on a small amount of blocks
  - Keep most in-used blocks at peak performance
- Exclude underperforming and aged blocks



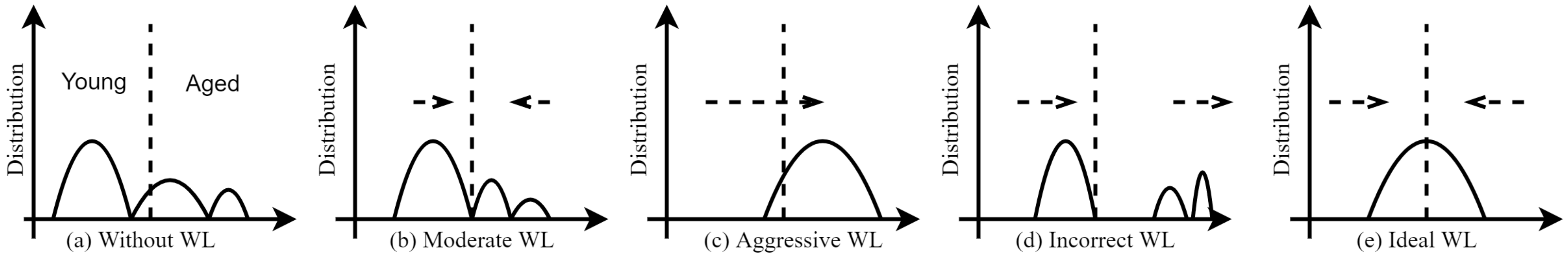
Wear focusing

- Keep most in-used blocks at peak performance and exclude underperforming and aged blocks.
- Avoid wear leveling overhead:
  - Static/Dynamic: affect WAF
  - Effective under limited scenarios
    - *"Wear leveling is not perfect"*

- Stathis Maneas et al, "Operational Characteristics of SSDs in Enterprise Storage Systems: A Large-Scale Field Study", FAST 2022
- Ziyang Jiao et al, "Wear Leveling in SSDs Considered Harmful", HotStorage 2022

Wear focusing

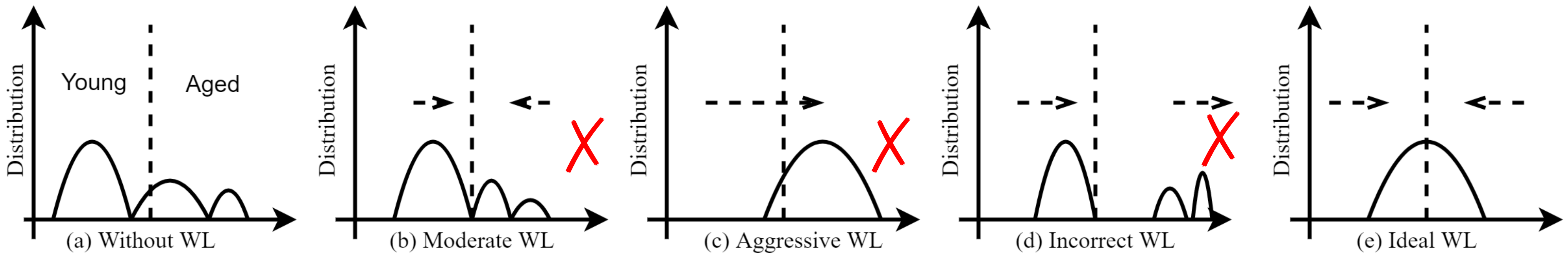
- Keep most in-used blocks at peak performance and exclude underperforming and aged blocks.
- Avoid wear leveling overhead:
    - Static/Dynamic: affect WAF
    - Effective under limited scenarios
        – *"Wear leveling is not perfect"*



- Stathis Maneas et al, "Operational Characteristics of SSDs in Enterprise Storage Systems: A Large-Scale Field Study", FAST 2022
- Ziyang Jiao et al, "Wear Leveling in SSDs Considered Harmful", HotStorage 2022

Wear focusing

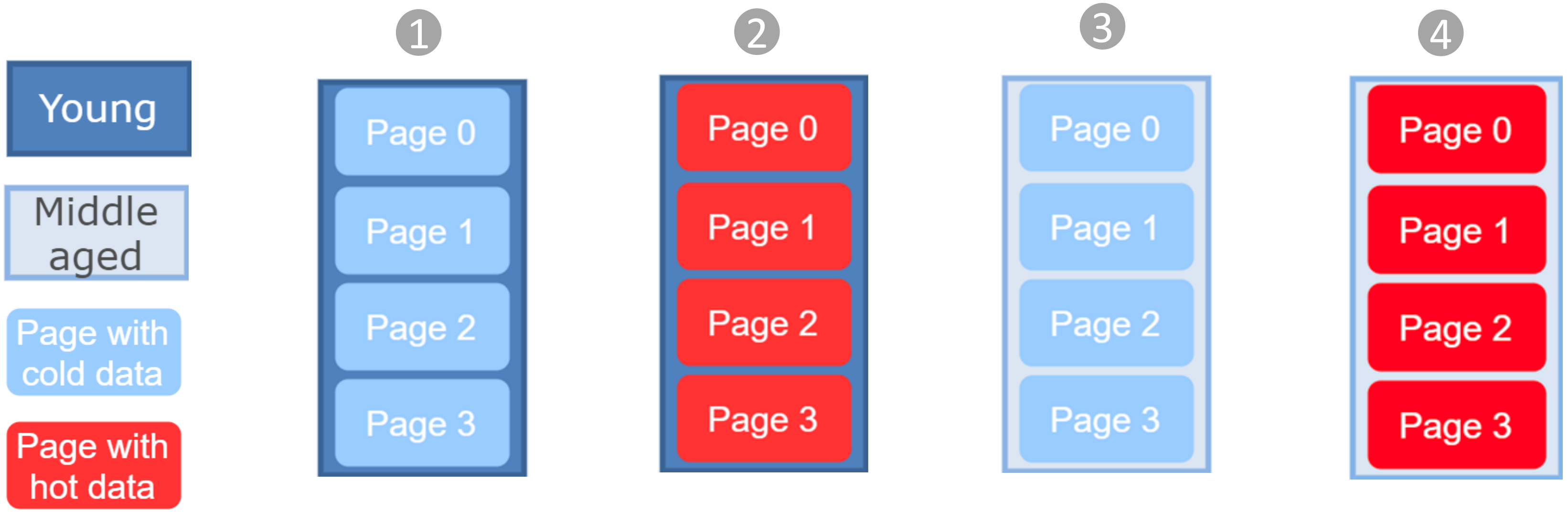
- Keep most in-used blocks at peak performance and exclude underperforming and aged blocks.
- Avoid wear leveling overhead:
  - Static/Dynamic: affect WAF
  - Effective under limited scenarios
    - *"Wear leveling is not perfect"*



- Stathis Maneas et al, "Operational Characteristics of SSDs in Enterprise Storage Systems: A Large-Scale Field Study", FAST 2022
- Ziyang Jiao et al, "Wear Leveling in SSDs Considered Harmful", HotStorage 2022

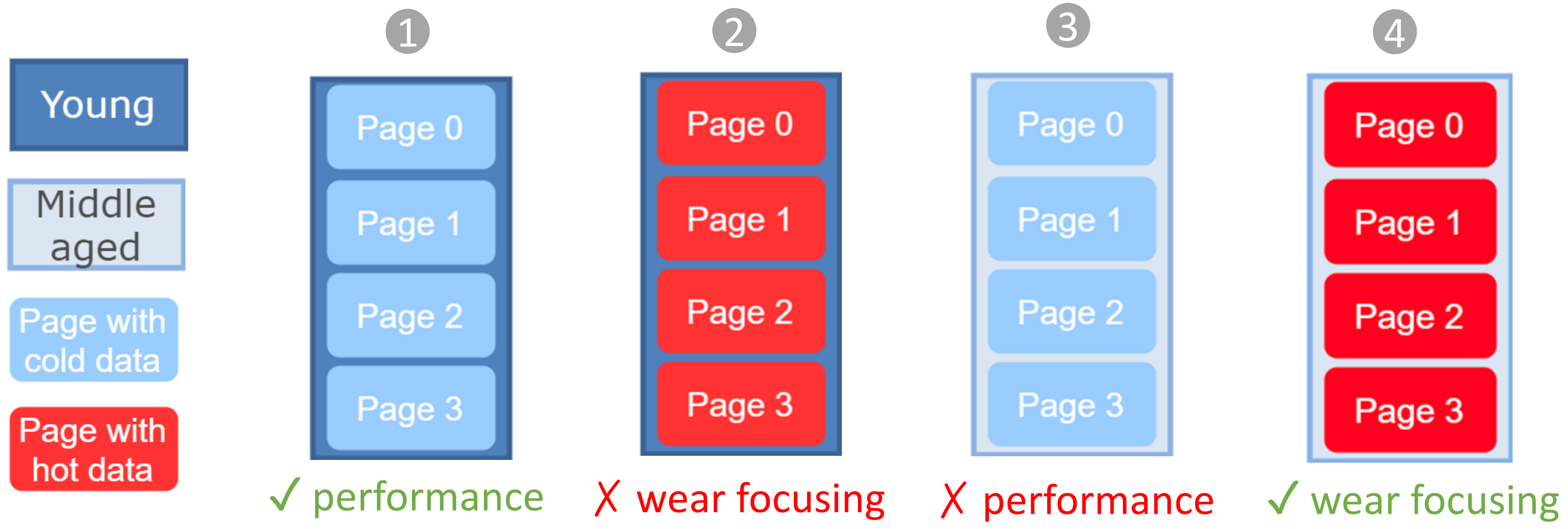
Life cycle management

- Four scenarios when considering data characteristics:
  1. Read-intensive data + young blocks
  2. Write-intensive data + young blocks
  3. Read-intensive data + middle-aged blocks
  4. Write-intensive data + middle-aged blocks



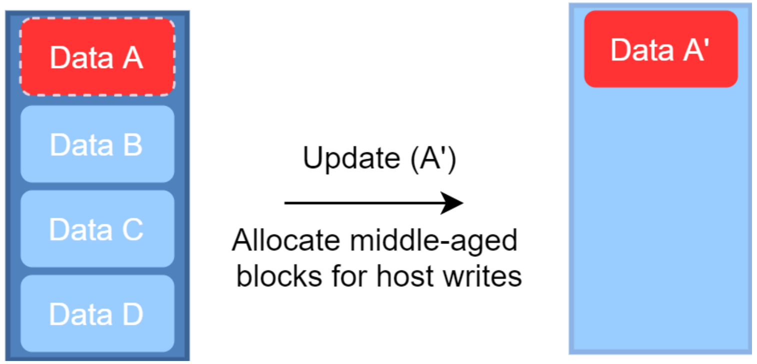
Life cycle management

- Four scenarios when considering data characteristics:
  1. Read-intensive data + young blocks
  2. Write-intensive data + young blocks → ✗ leveling wear
  3. Read-intensive data + middle-aged blocks → ✗ error correction
  4. Write-intensive data + middle-aged blocks



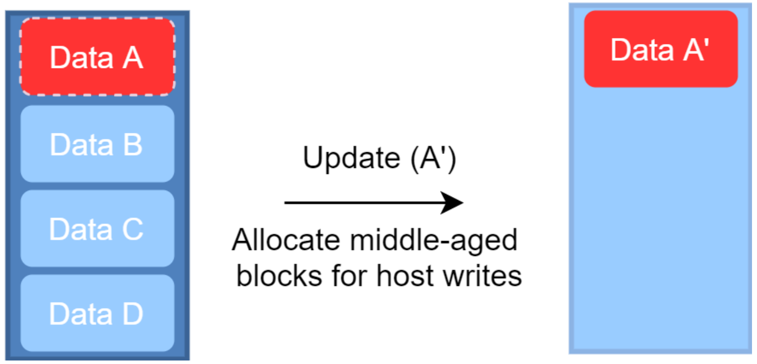
Life cycle management

- Write-intensive data + young blocks → ✗ leveling wear
  - Allocation policy:
    - Young blocks for GC
    - Middle-aged blocks for the host



Life cycle management

- Write-intensive data + young blocks → ✗ leveling wear
  - Allocation policy:
    - Young blocks for GC
    - Middle-aged blocks for the host



- Read-intensive data + middle-aged blocks → ✗ error correction
  - Garbage collection policy:
    - $$\begin{aligned} Victim\ score = & W_{invalidity} \cdot invalid\ ratio \\ & + W_{aging} \cdot aging\ ratio \\ & + W_{read} \cdot read\ ratio \end{aligned}$$

$$invalid\ ratio = \frac{\#\ of\ invalid\ pages}{\#\ of\ toal\ pages}, \qquad aging\ ratio = \frac{erase\ count}{endurance}$$

$$read\ ratio = \frac{\#\ of\ host\ read}{maximum\ host\ read\ among\ unretired\ blocks}$$

Outline

- Background & motivation
- Design principles
- Capacity-variant storage system
- Evaluation
- Summary

Evaluation setup

- Host environment
  - CPU: Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz * 32
  - Memory: Samsung 64GB DDR4 RAM * 16
  - SSD: Intel DC P4510 1.6TiB
  - OS: Ubuntu 20.04.5 LTS (Focal Fossa)
- Target configurations
  - TrSS: F2FS + traditional SSD
  - AutoStream: place data based on access pattern
  - ttFlash: reduce latency with data reconstruction
  - CVSS: our solution
- Workloads
  - FIO, Filebench, Twitter traces, and YCSB

FEMU configurations (Tr/CV-SSD)			
Channels	8	Physical capacity	128 GiB
Luns per channel	8	Logical capacity	120 GiB
Planes per lun	1	Program latency	500 μs
Blocks per plane	512	Read latency	50 μs
Pages per block	1024	Erase latency	5 ms
Page size	4 KiB	Wear leveling	PWL
Endurance	300	ECC strength	50 bits

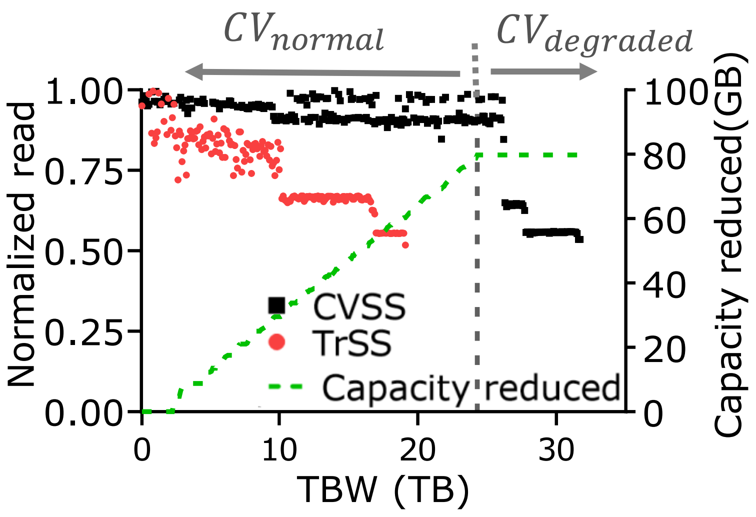
- Juncheng Yang et al, “A Large Scale Analysis of Hundreds of In-memory Cache Clusters at Twitter”, OSDI 2020
- Jingpei Yang et al, “AutoStream: Automatic Stream Management for Multi-streamed SSDs”, SYSTOR 2017
- Shiqin Yan et al, “Tiny-tail flash: Near-perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs”, FAST 2017
- Fu-Hsin Chen et al, “PWL: A Progressive Wear Leveling to Minimize Data Migration Overheads for NAND Flash Devices”, DATE 2015

Evaluation overview

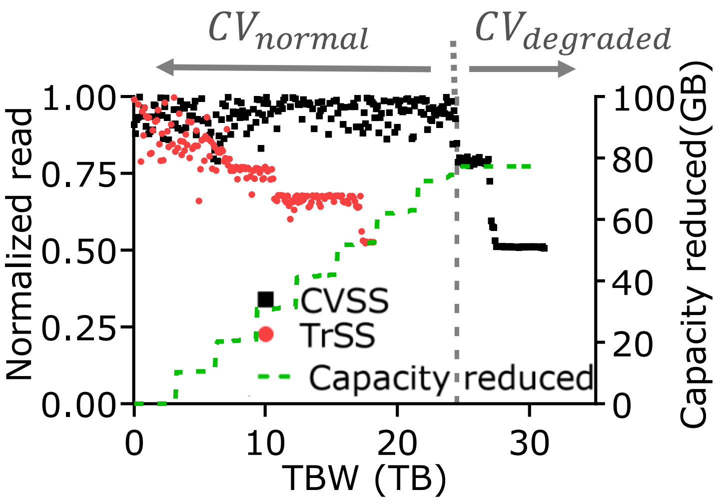
1. Can CVSS maintain performance while the underlying device ages?
2. How does CVSS perform compared to other techniques under real workloads?
3. Can CVSS extend the device lifetime given different performance requirements?

Synthetic workloads (FIO)

- Device utilization: 30%
- FIO read/write ratio: 0.5/0.5
- Measure until the performance drops below 50%



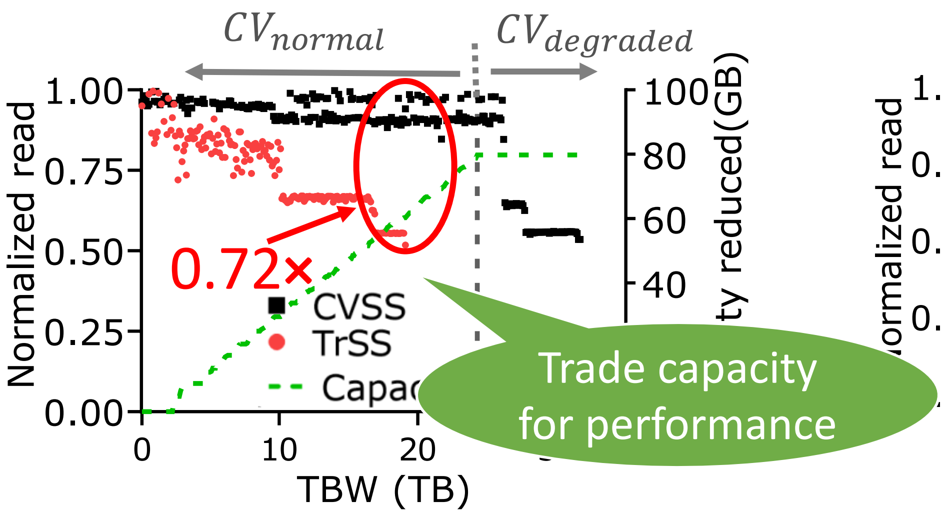
Zipfian workloads



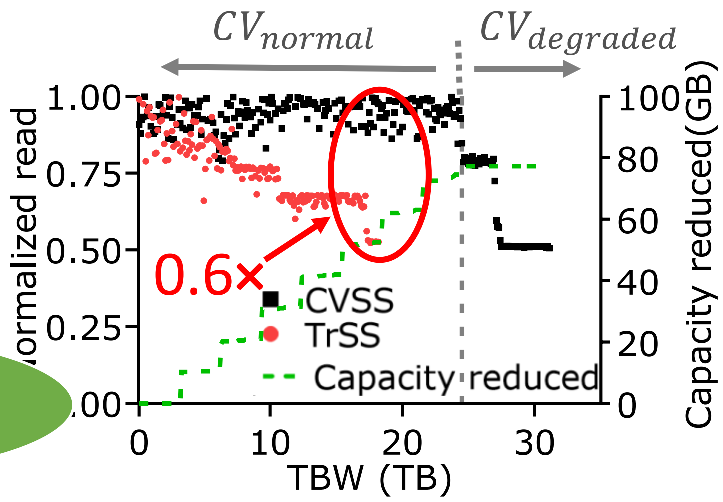
Random workloads

Synthetic workloads (FIO)

- Device utilization: 30%
- FIO read/write ratio: 0.5/0.5
- Measure until the performance drops below 50%



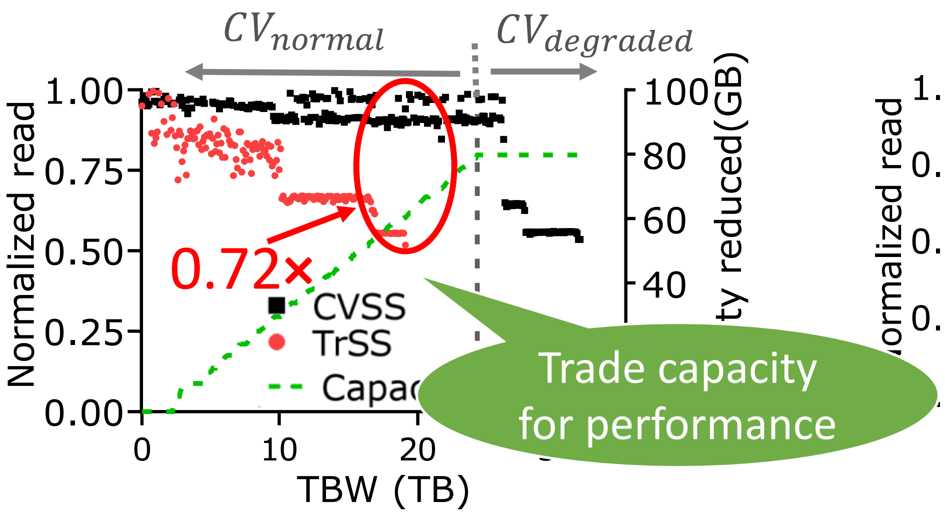
Zipfian workloads



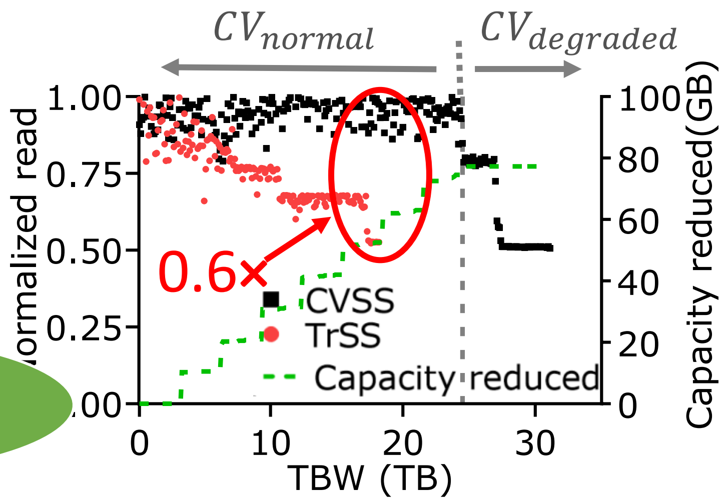
Random workloads

Synthetic workloads (FIO)

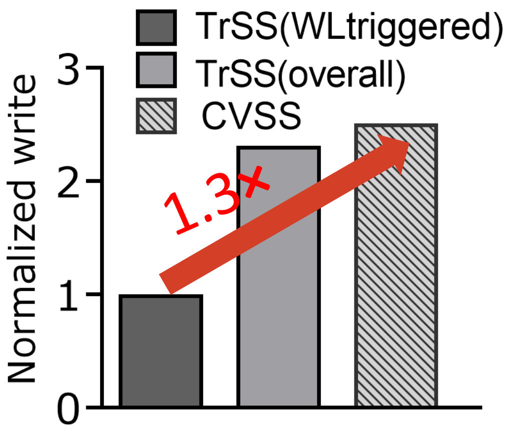
- Device utilization: 30%
- FIO read/write ratio: 0.5/0.5
- Measure until the performance drops below 50%



Zipfian workloads



Random workloads



Average write throughput

Twitter traces

- Key-value traces from Twitter production
- 36.7 GB key-value pairs + RocksDB
  - Up to 65 GB during running due to RocksDB’s space amplification

- Juncheng Yang et al, “A Large Scale Analysis of Hundreds of In-memory Cache Clusters at Twitter”, OSDI 2020

Twitter traces

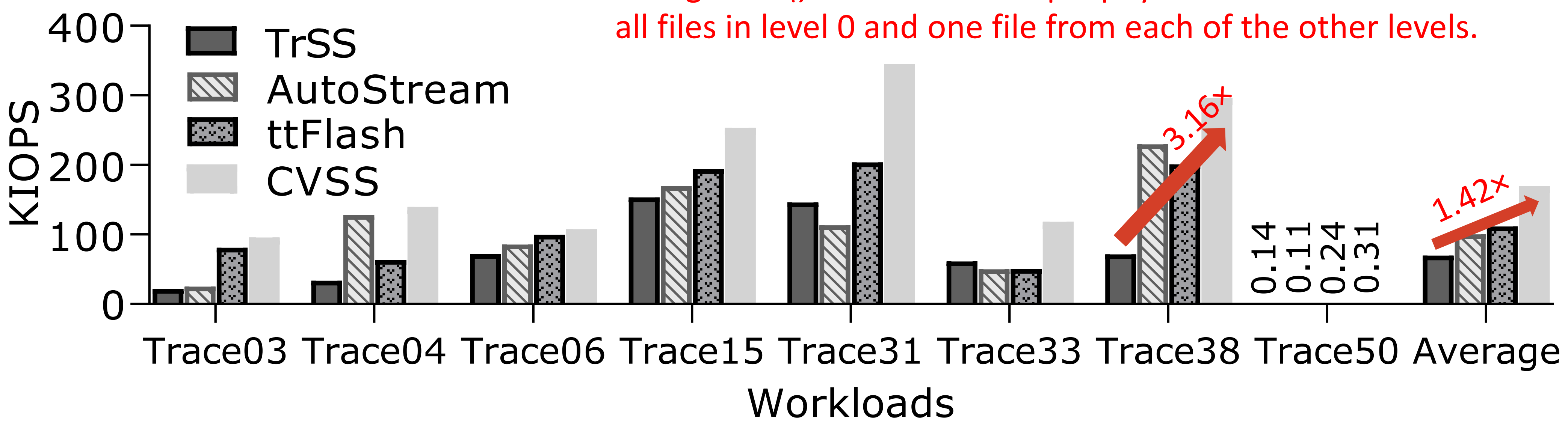
- Key-value traces from Twitter production
- 36.7 GB key-value pairs + RocksDB
  - Up to 65 GB during running due to RocksDB's space amplification

- Juncheng Yang et al, "A Large Scale Analysis of Hundreds of In-memory Cache Clusters at Twitter", OSDI 2020

Twitter traces

- Key-value traces from Twitter production
- 36.7 GB key-value pairs + RocksDB
  - Up to 65 GB during running due to RocksDB's space amplification

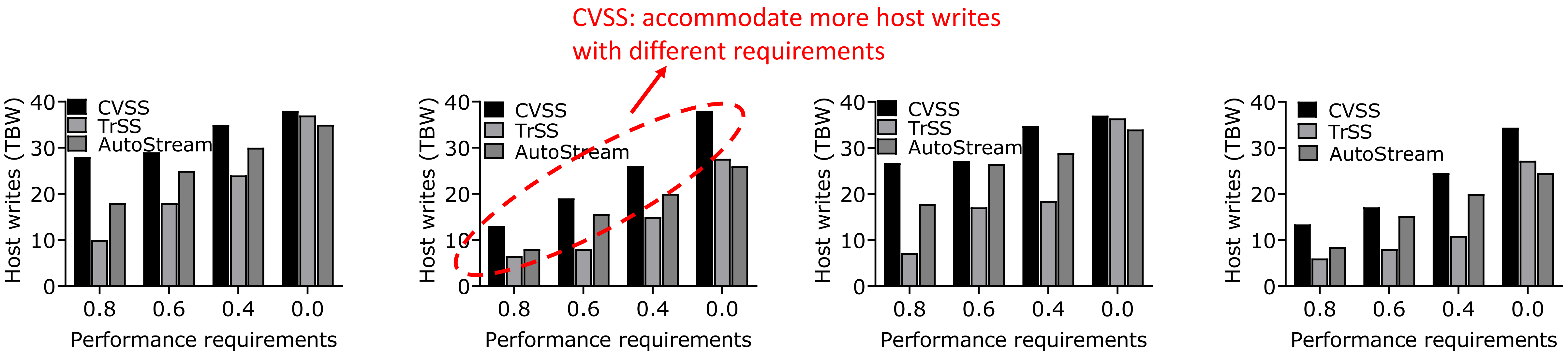
A single *Get()* can cause multiple physical reads:
all files in level 0 and one file from each of the other levels.



- Juncheng Yang et al, "A Large Scale Analysis of Hundreds of In-memory Cache Clusters at Twitter", OSDI 2020

Lifetime extension

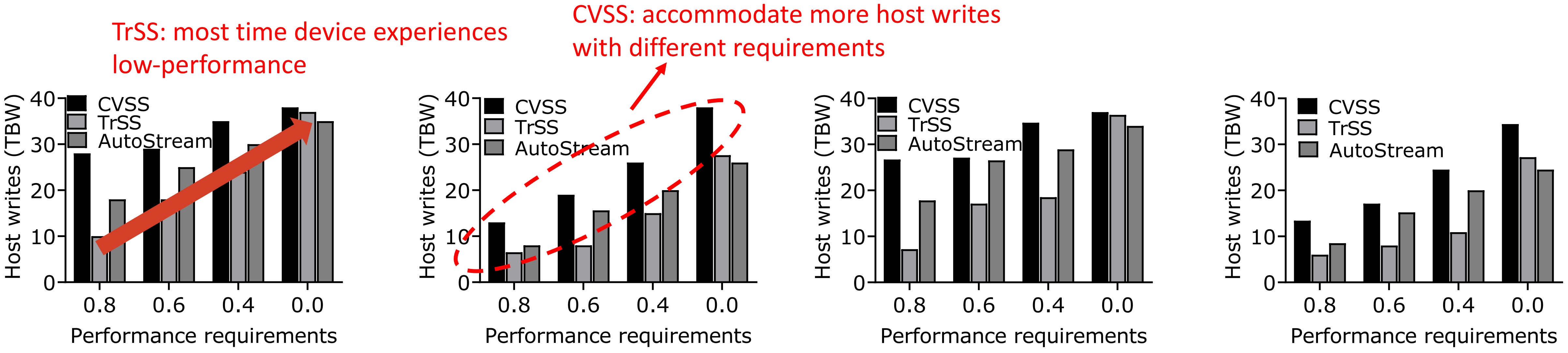
- TBW before the device performance drops below 0.8, 0.6, 0.4, and 0 of the initial state
- ttFlash introduces additional write overhead coming from RAIN



(a) Zipfian (30% utilization) (b) Zipfian (70% utilization) (c) Random (30% utilization) (d) Random (70% utilization)

Lifetime extension

- TBW before the device performance drops below 0.8, 0.6, 0.4, and 0 of the initial state
- ttFlash introduces additional write overhead coming from RAIN



(a) Zipfian (30% utilization) (b) Zipfian (70% utilization) (c) Random (30% utilization) (d) Random (70% utilization)

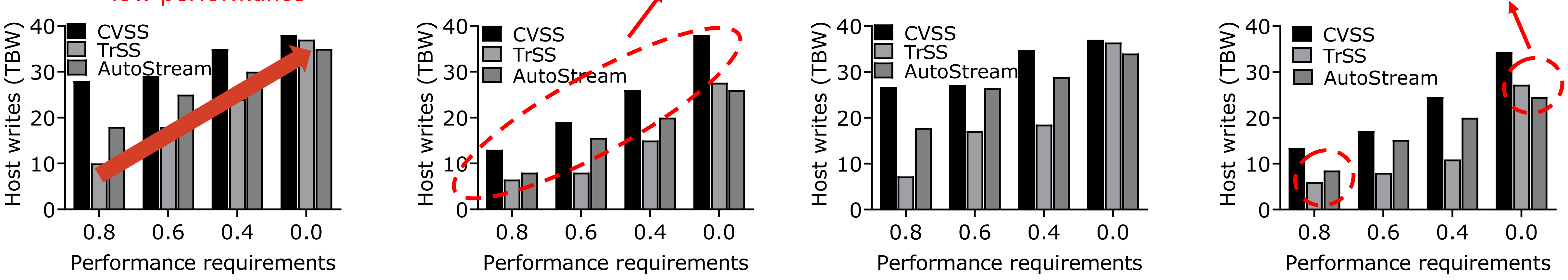
Lifetime extension

- TBW before the device performance drops below 0.8, 0.6, 0.4, and 0 of the initial state
- ttFlash introduces additional write overhead coming from RAIN

TrSS: most time device experiences low-performance

CVSS: accommodate more host writes with different requirements

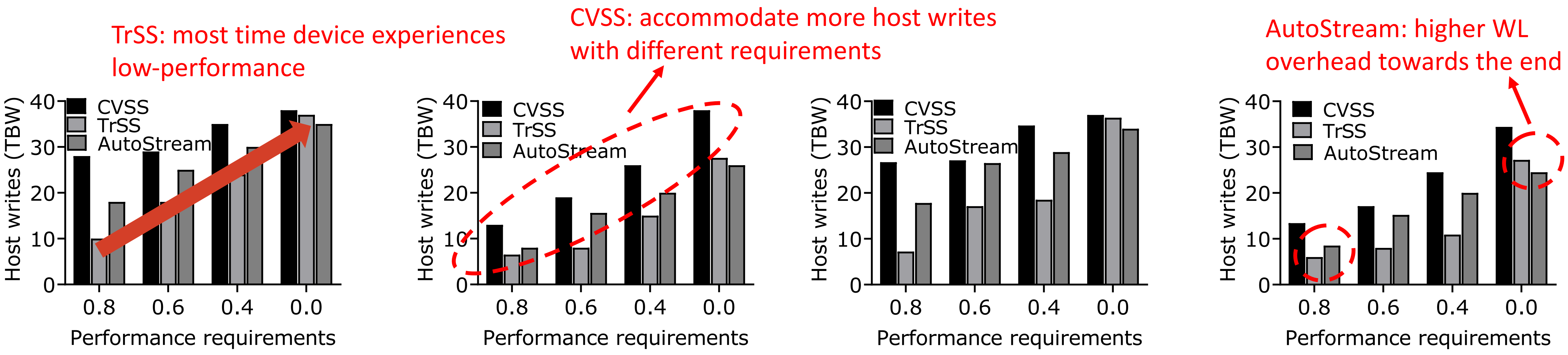
AutoStream: higher WL overhead towards the end



(a) Zipfian (30% utilization) (b) Zipfian (70% utilization) (c) Random (30% utilization) (d) Random (70% utilization)

Lifetime extension

- TBW before the device performance drops below 0.8, 0.6, 0.4, and 0 of the initial state
- ttFlash introduces additional write overhead coming from RAIN



(a) Zipfian (30% utilization) (b) Zipfian (70% utilization) (c) Random (30% utilization) (d) Random (70% utilization)

More results are included in the paper!

Summary

- The current storage system abstraction of fixed capacity worsens aging-related performance degradation for modern SSDs.
- The capacity-variant storage systems
  - Relax the fixed-capacity abstraction of the underlying storage device
  - Components
    - CV-FS, CV-SSD, and CV-manager
  - Benefits
    - Performant SSD even when aged
    - Extended lifetime for SSD-based storage
    - Streamlined SSD design

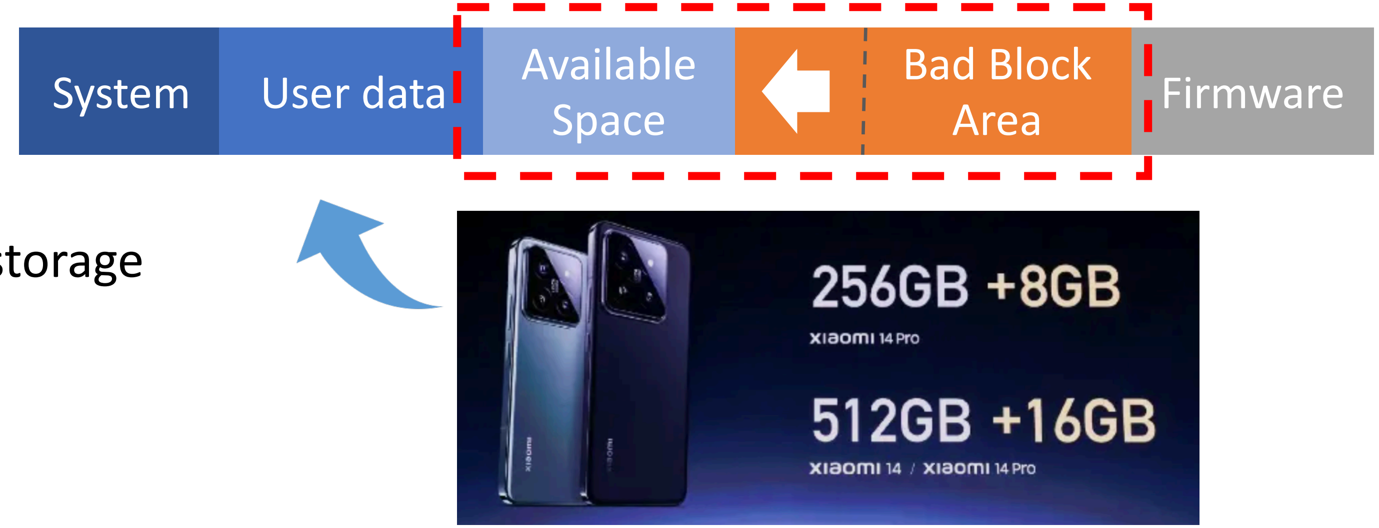
Summary

- The current storage system abstraction of fixed capacity worsens aging-related performance degradation for modern SSDs.
- The capacity-variant storage systems
  - Relax the fixed-capacity abstraction of the underlying storage device
  - Components
    - CV-FS, CV-SSD, and CV-manager
  - Benefits
    - Performant SSD even when aged
    - Extended lifetime for SSD-based storage
    - Streamlined SSD design

- Image from Google searches

Summary

- The current storage system abstraction of fixed capacity worsens aging-related performance degradation for modern SSDs.
- The capacity-variant storage systems
  - Relax the fixed-capacity abstraction of the underlying storage device
  - Components
    - CV-FS, CV-SSD, and CV-manager
  - Benefits
    - Performant SSD even when aged
    - Extended lifetime for SSD-based storage
    - Streamlined SSD design
- Future work
  - CV-RAID and new features



- Image from Google searches

Thank you
Any questions?

Contact: zjiao04@syr.edu

Source Code: https://github.com/ZiyangJiao/FAST24_CVSS_FEMU