White-Box Optimization Approaches for Ultra Large-Capacity NAND Flash-Based SSDs

NVRAMOS 2024

Prof. Jisung Park

COMPUTER ARCHITECTURE & OPERATING SYSTEMS LABORATORY

2024.10.24

Outline

- Ultra Large-Capacity NAND Flash-Based SSDs
- RiF: Improving Read Performance of Modern SSDs Using an On-Die Early-Retry Engine
- AERO: Adaptive Erase Operation for Improving Lifetime and Performance of Modern NAND Flash-Based SSDs
- Conclusion

Demands for Unprecedented SSD Capacity

- SSD capacity exceeding several tens of terabytes
  - e.g., Samsung's BM1743 and Solidigm's D5-P5336 (61.44 TB)
  - Effective at reducing TCO of data centers
    - Need to meet explosive storage-capacity requirements from data-intensive applications (e.g., AI)
    - Higher density → Fewer SSDs → Fewer storage servers → Lower power/area → Lower cooling cost
  - Equipped w/ hundreds high-density NAND flash chips
    - ~200 wordline (WL) layers vertically stacked
    - Advanced multi-level cell (MLC) technology (e.g., QLC)

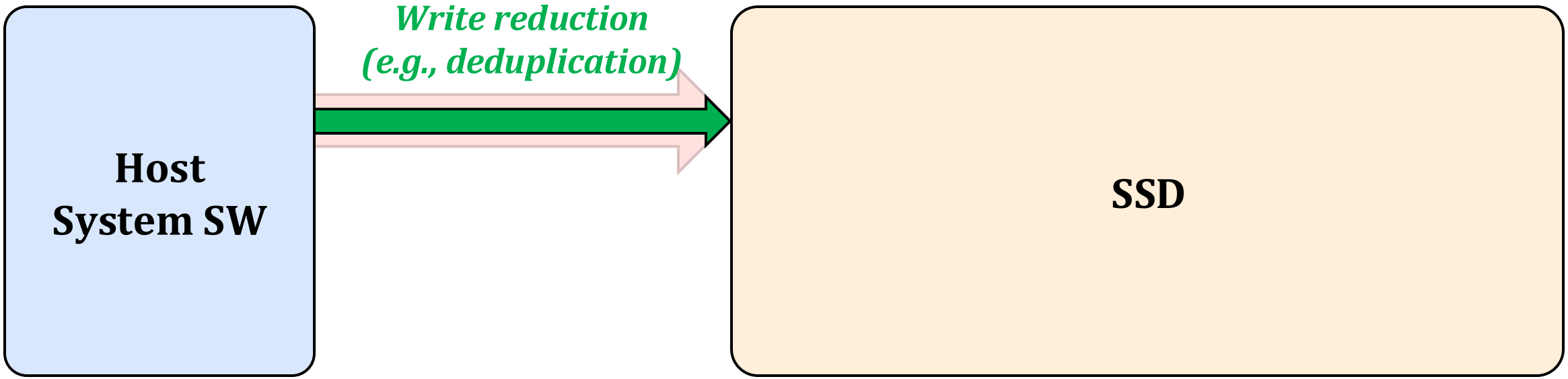
Potential Problems in ULC-SSD Design

- Significant reliability issues in high-density NAND flash memory
  - Lifetime and performance degradation
- Limited SSD-internal I/O bandwidth
  - Due to limited number of channels
- Increasing power consumption
  - Inevitable to put more chips in a device w/ limited power budget
- Metadata overhead
  - Linearly increases with SSD capacity

White-Box Approaches for SSD Optimization

- Enables taking full advantage of the system's potential
  - Opposite to black-box approaches that optimize a single system layer w/o deep understanding of other layers' internal

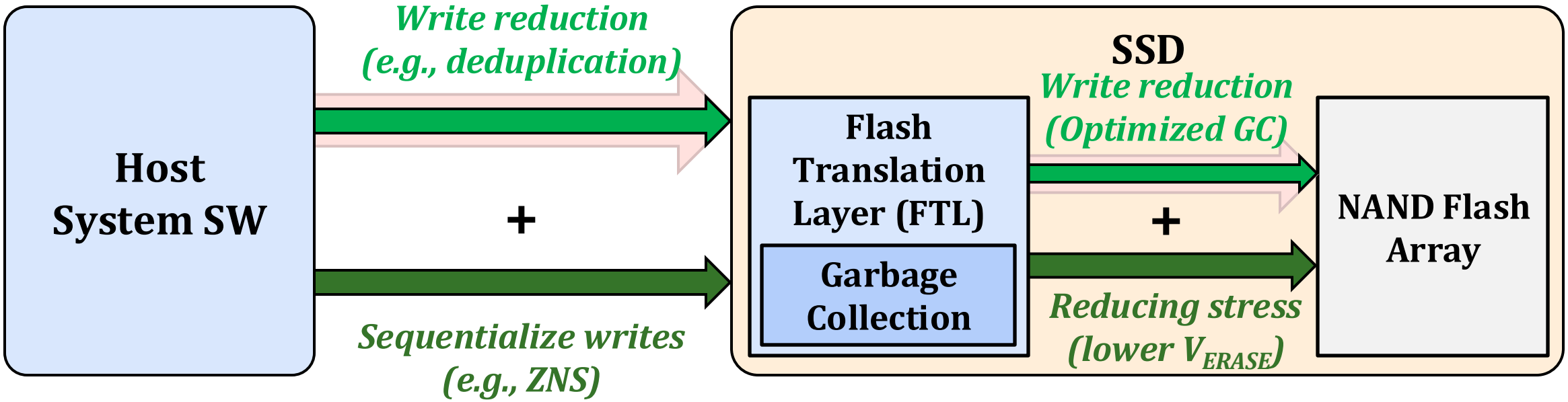
Problem: Limited SSD Lifetime



White-Box Approaches for SSD Optimization

- Enables taking full advantage of the system's potential
  - Opposite to black-box approaches that optimize a single system layer w/o deep understanding of other layers' internal

Problem: Limited SSD Lifetime



White-Box Approaches for SSD Optimization

- Enables taking full advantage of the system's potential
  - Opposite to black-box approaches that optimize a single system layer w/o deep understanding of other layers' internal

White-box optimization approach across system layers is the key to simultaneously meeting various requirements

Outline

- Ultra Large-Capacity NAND Flash-Based SSDs
- RiF: Improving Read Performance of Modern SSDs Using an On-Die Early-Retry Engine
- AERO: Adaptive Erase Operation for Improving Lifetime and Performance of Modern NAND Flash-Based SSDs
- Conclusion

RiF: Retry-in-Flash

RiF: Improving Read Performance of Modern SSDs Using an On-Die Early-Retry Engine

Myoungjun Chun*[1], Jaeyong Lee*[1], Myungsuk Kim[2], Jisung Park[3], Jihong Kim[1]
[1]Seoul National University, [2]Kyungpook National University, [3]POSTECH
[1]{mjchun, jylee, jihong}@davinci.snu.ac.kr, [2]ms.kim@knu.ac.kr, [3]jisung.park@postech.ac.kr

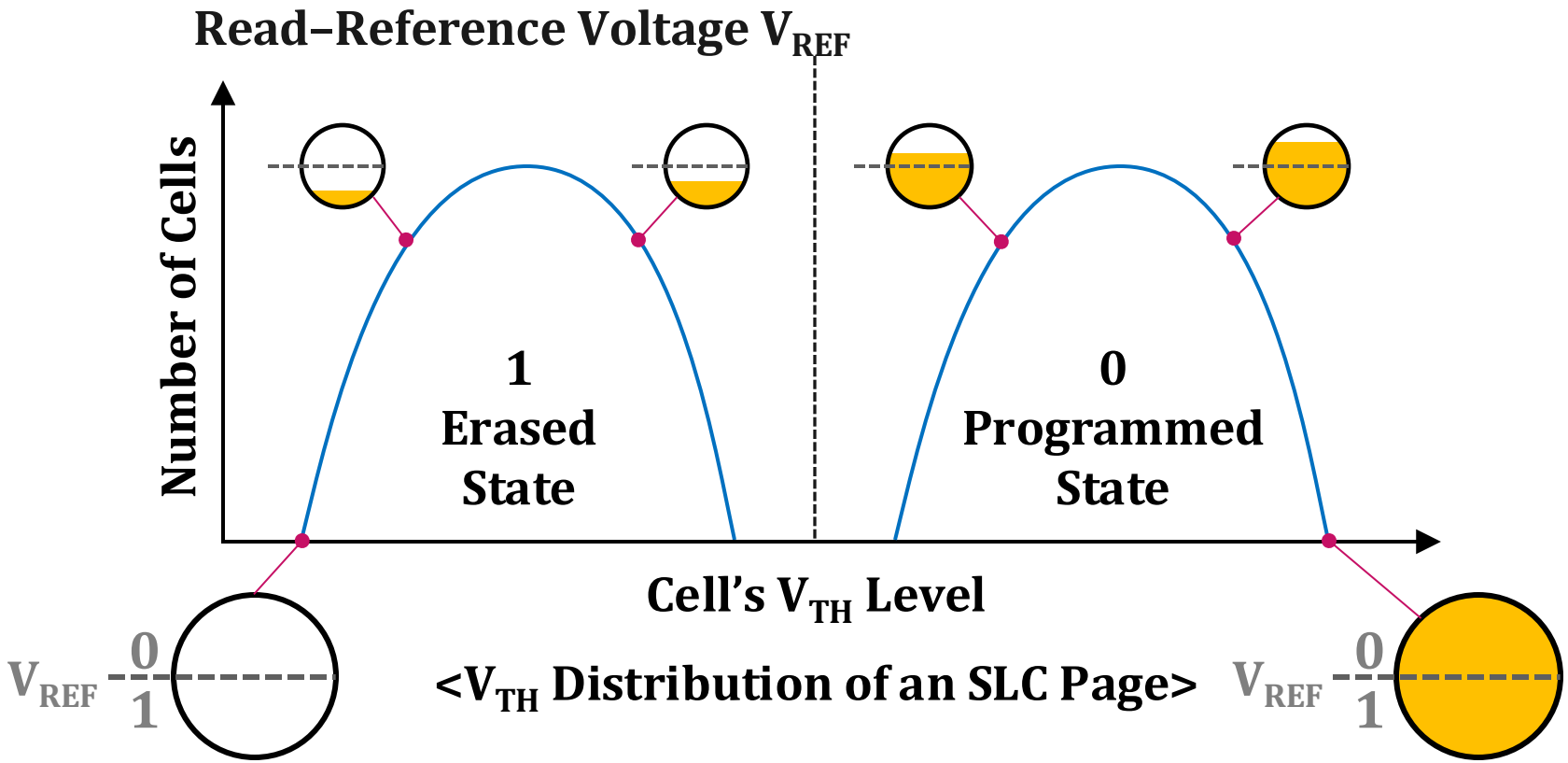
2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)

RiF: Outline

- Read-Retry in Modern SSDs
- Key Idea: Retry-in-Flash (RiF)
- Evaluation Results
- Summary

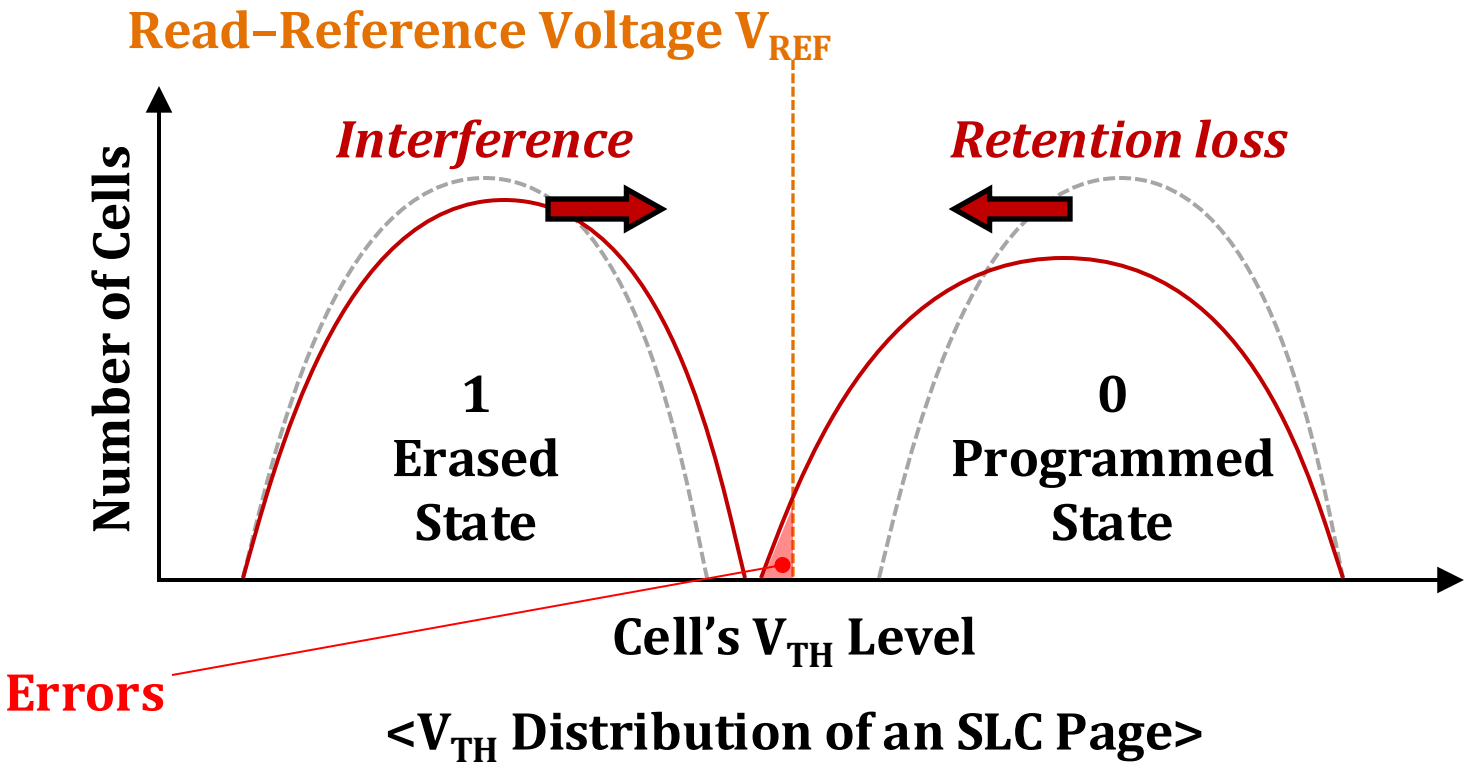
Errors in NAND Flash Memory

- NAND flash memory stores data by using cells' V_{TH} levels



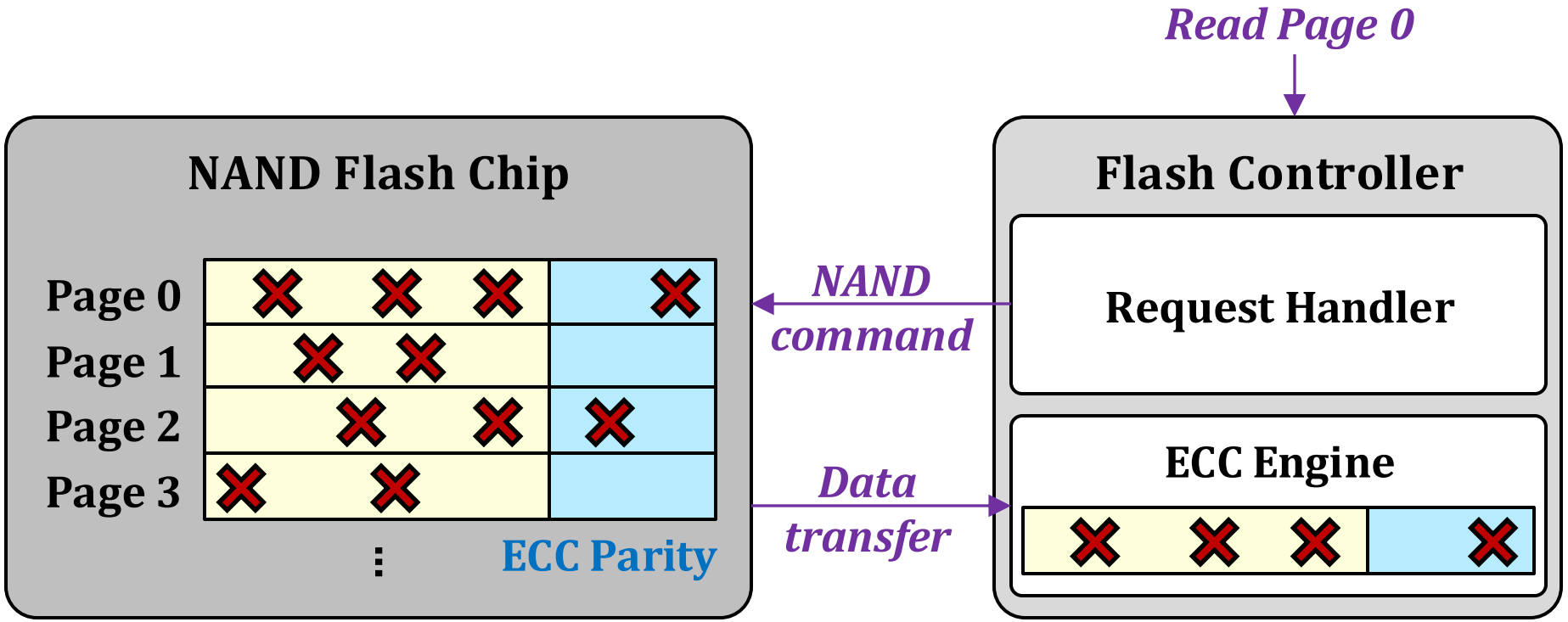
Errors in NAND Flash Memory

- Various sources shift and widen programmed V_{TH} states
  - Retention loss, program interference, read disturbance, etc.



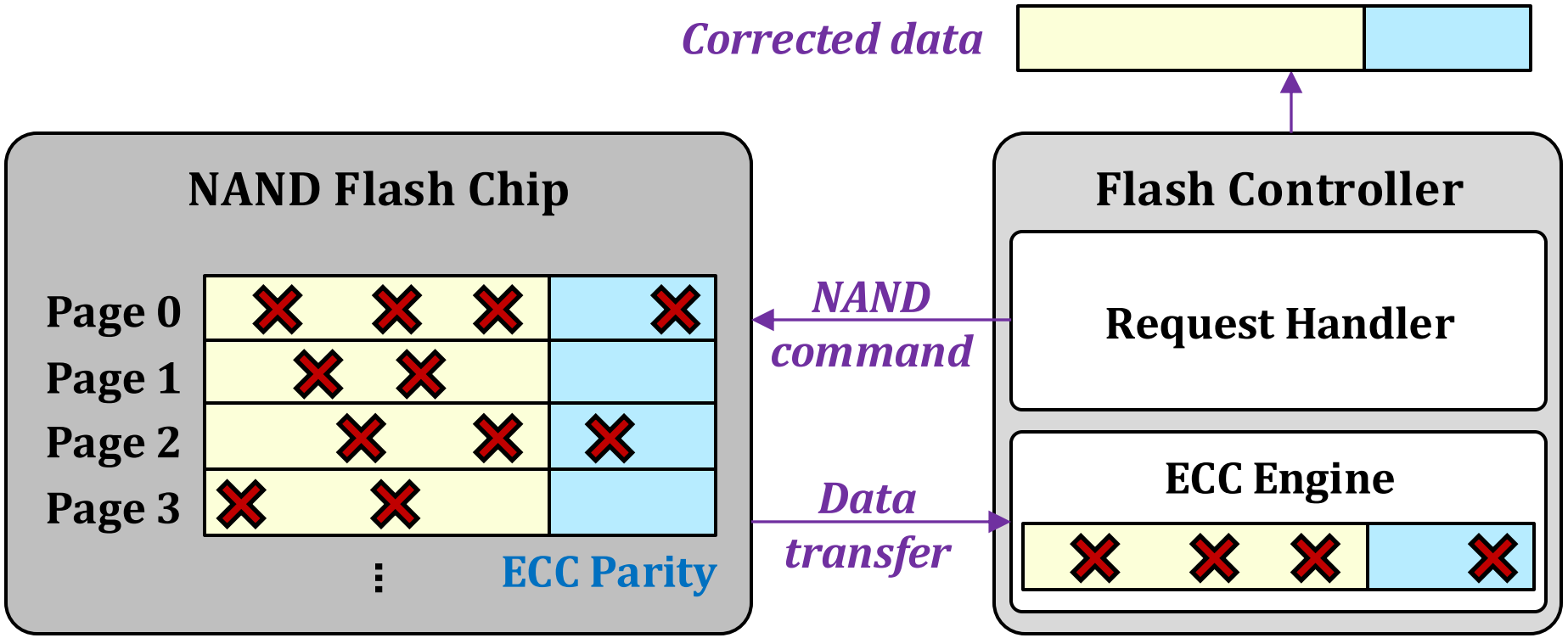
Error-Correcting Codes (ECC)

- Store redundant information (ECC parity)
  - To detect and correct row bit errors



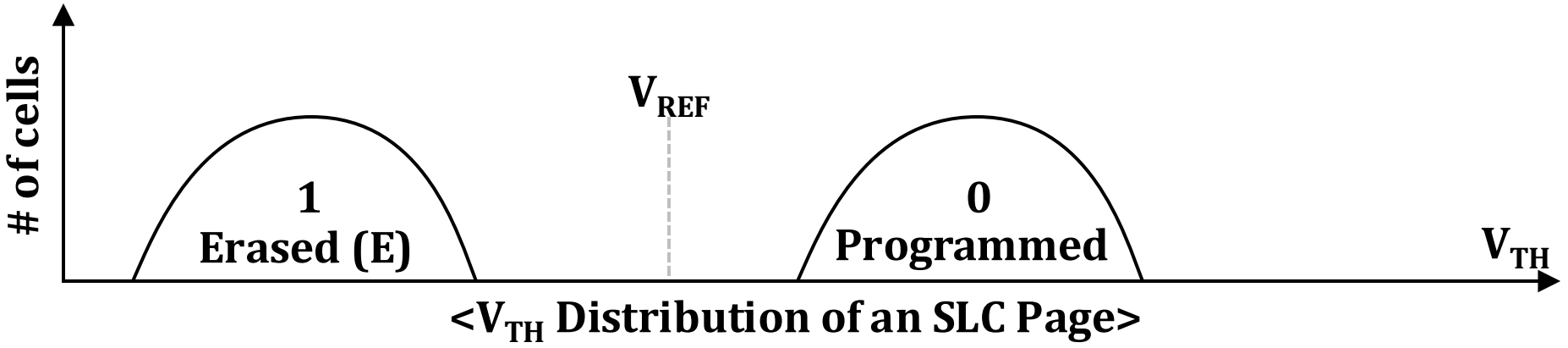
Error-Correcting Codes (ECC)

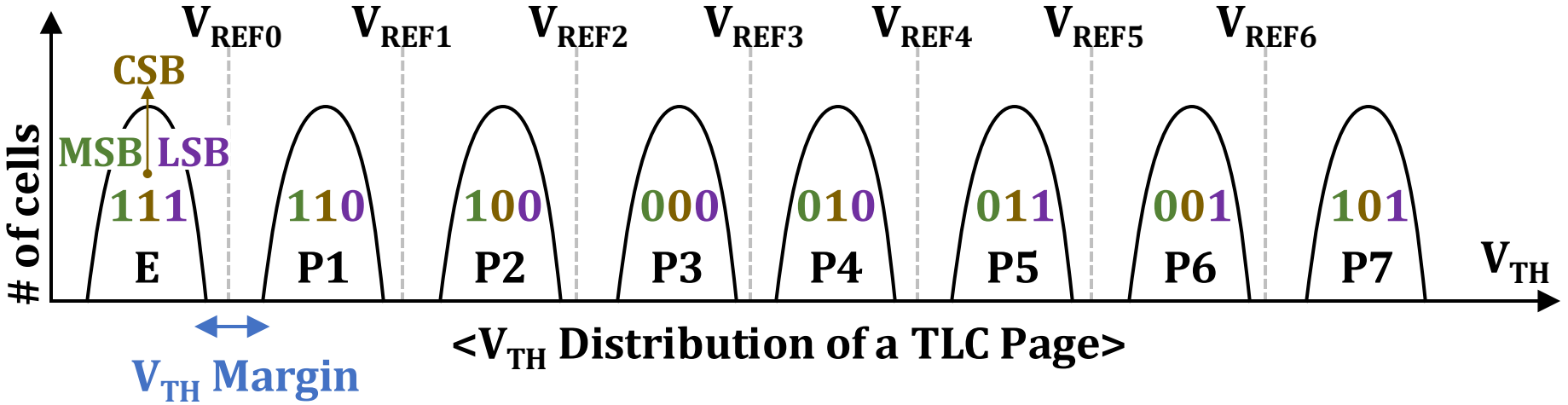
- Store redundant information (ECC parity)
  - To detect and correct row bit errors



Errors in Modern NAND Flash Memory

- High row bit-error rates (RBER) in MLC NAND flash memory
  - Narrow margin b/w adjacent V_{TH} states





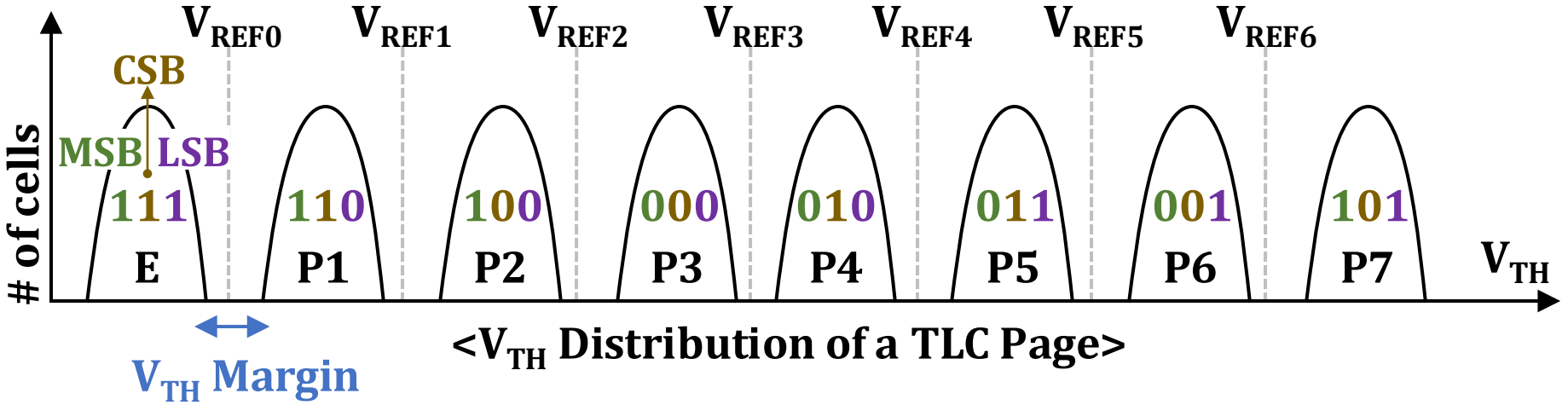
Errors in Modern NAND Flash Memory

- High row bit-error rates (RBER) in MLC NAND flash memory
  - Narrow margin b/w adjacent V_{TH} states

Strong ECC: Corrects ~80 bit errors per 1-KiB data

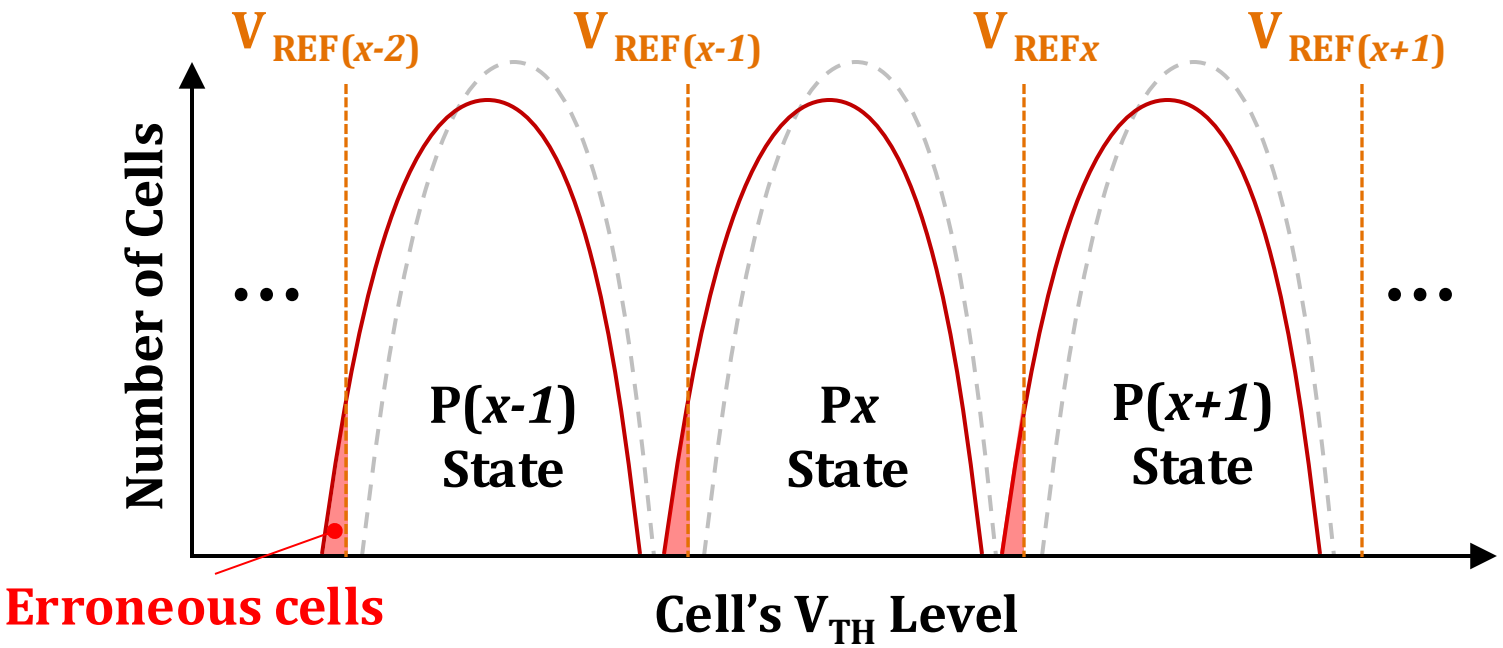
Not Scalable: Area, power, latency, …

What if RBER > ECC Capability?



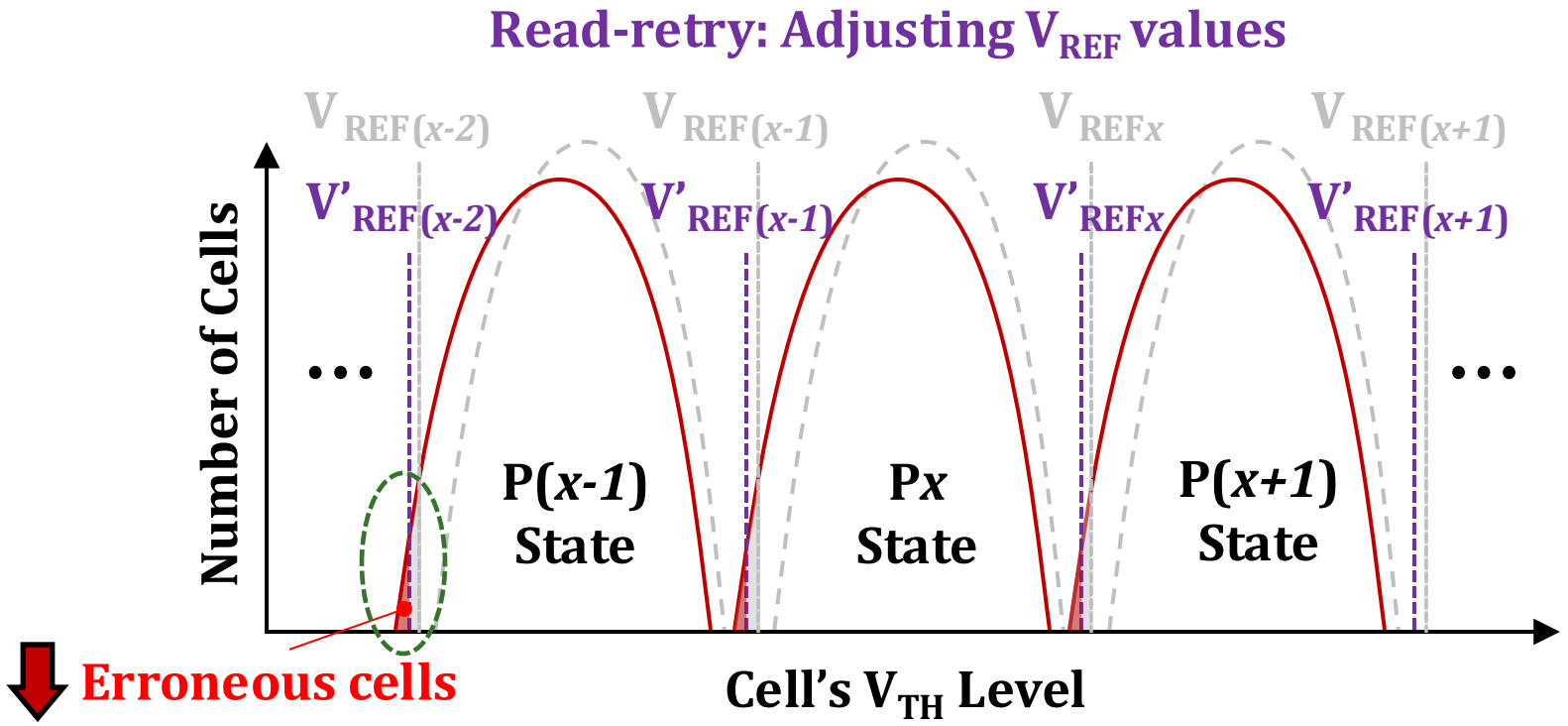
Read-Retry Operation

- Reads the page again with adjusted V_{REF} values



Read-Retry Operation

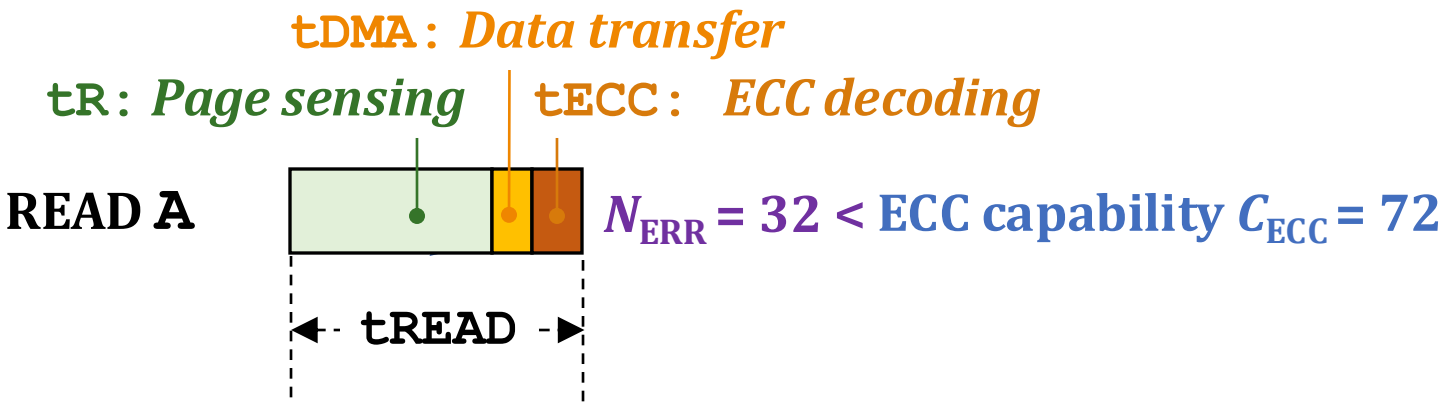
- Reads the page again with adjusted V_{REF} values



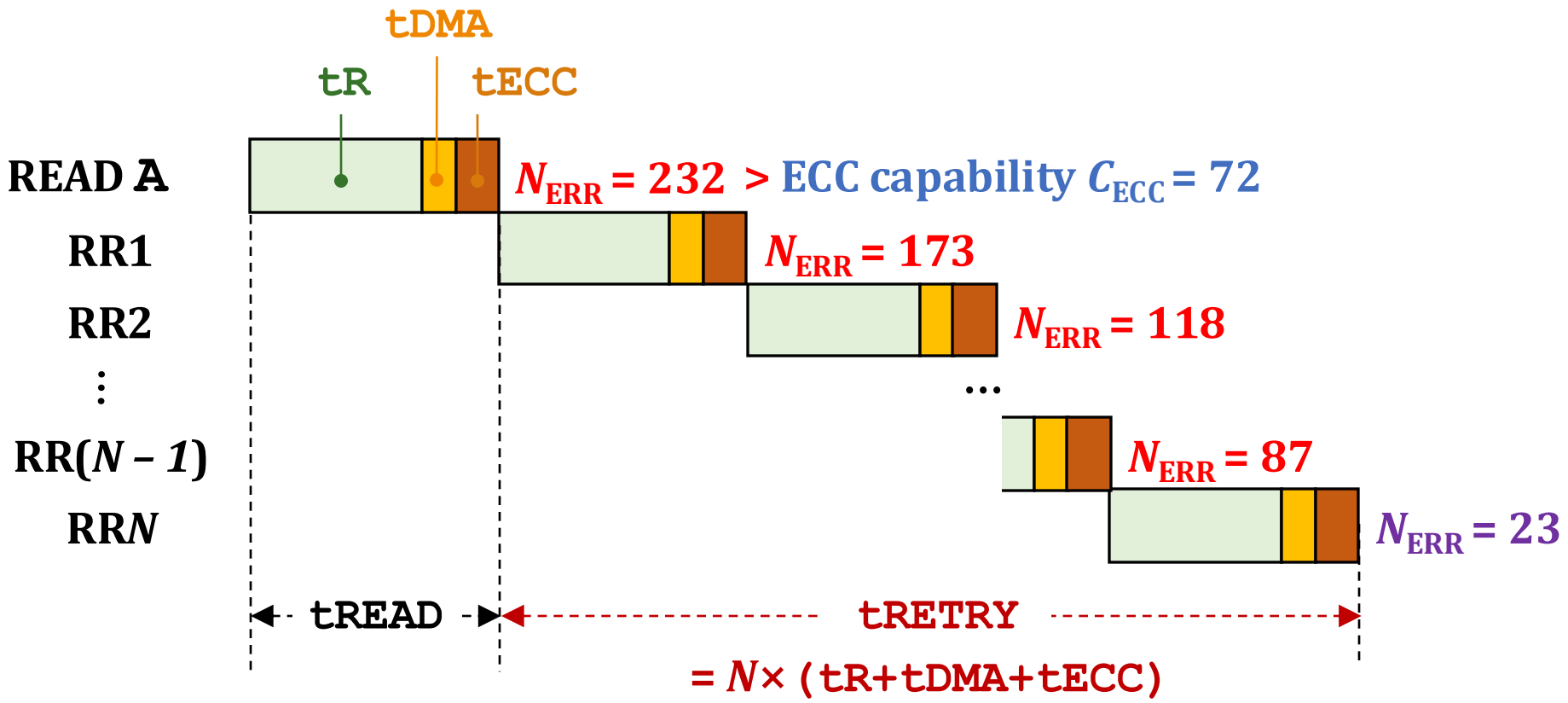
Read using properly-adjusted V_{REF} values

→ Decreases # of raw bit errors to be lower than the ECC capability

Read-Retry: Performance Overhead



Read-Retry: Performance Overhead



Read-retry increases the read latency almost linearly with the number of retry steps

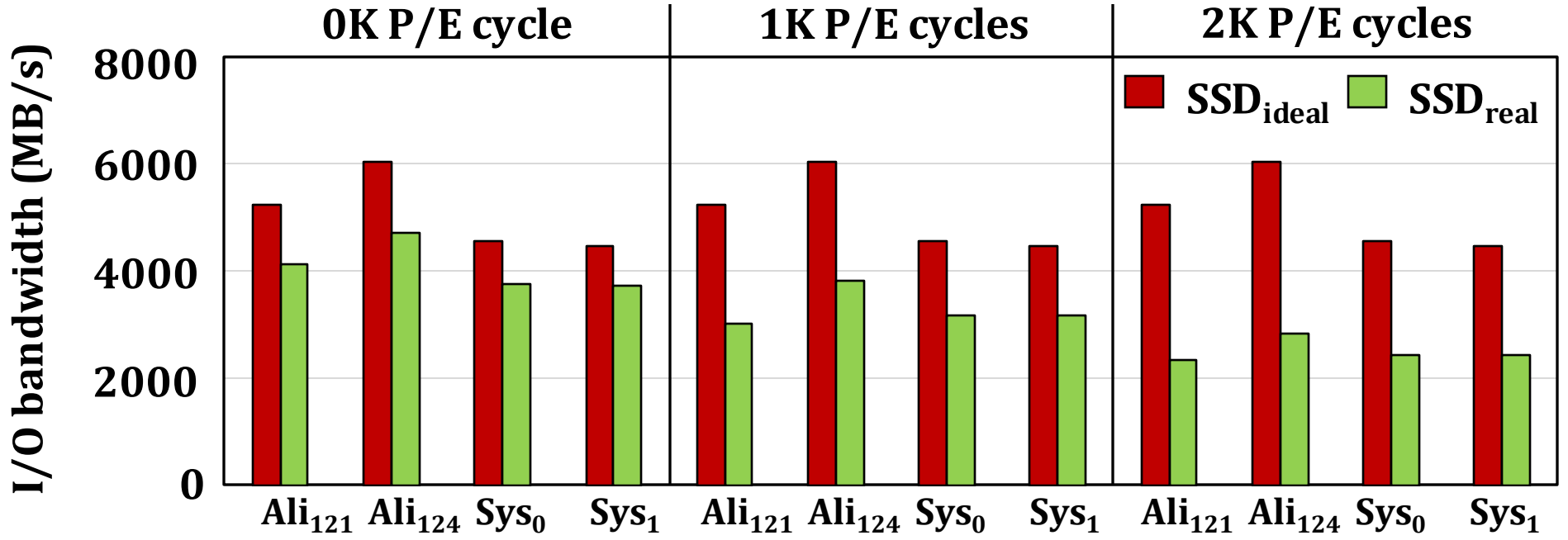
Existing Mitigations

- Reducing # of retry steps
  - e.g., Process-similarity-aware optimization [MICRO'19], Sentinel [MICRO'21], and SwiftRead [ISSCC'22]
  - Quickly identify the near-optimal read-reference voltage levels
    - Based on the error characteristics of NAND flash memory
    - Enables skipping unnecessary retry steps

- Reducing the latency of each retry step
  - Adaptive read-retry [ASPLOS'22]
  - Reduce **tR** to be just enough to reliably read data in the final retry step
    - Leveraging high error-correction capability margin in modern SSDs

Impact of Read Retry on SSD Performance

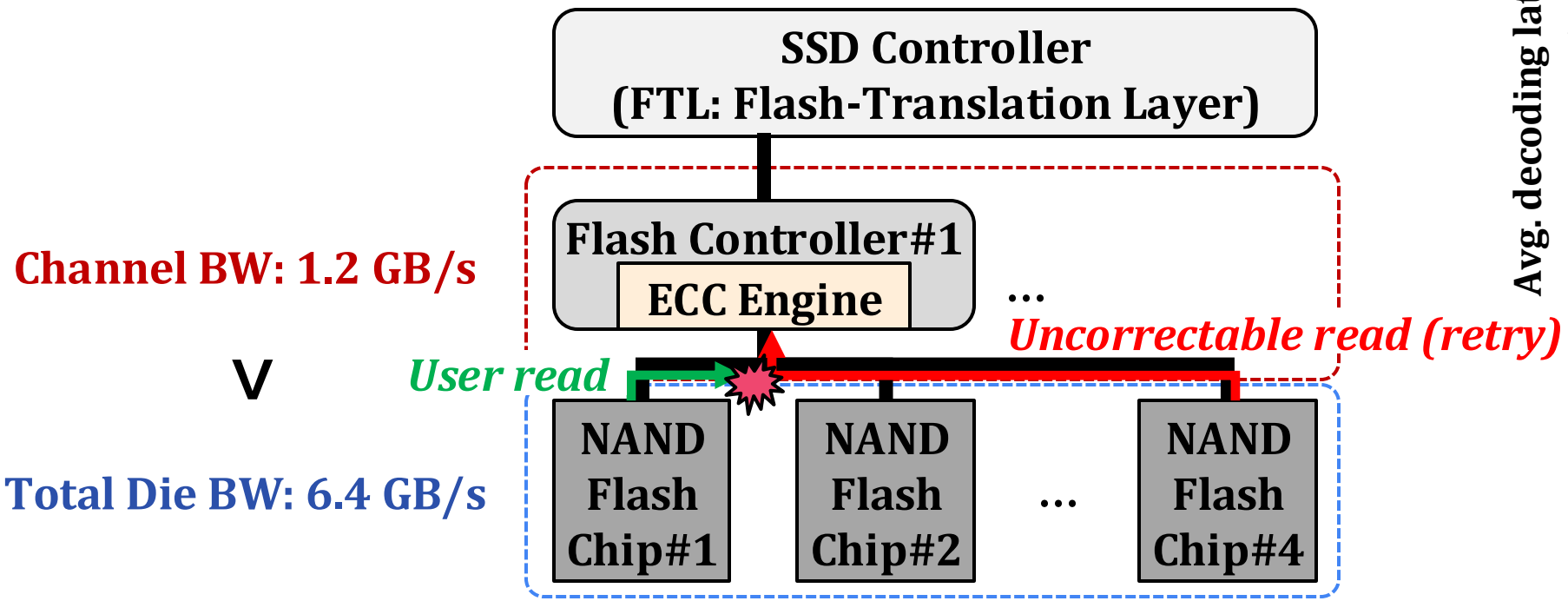
- Motivational experiments using MQSim-E [IEEE CAL 2022]
  - SSD_{real} w/ optimal Swift-Read (up to one retry step)
  - SSD_{ideal} w/ no read-retry

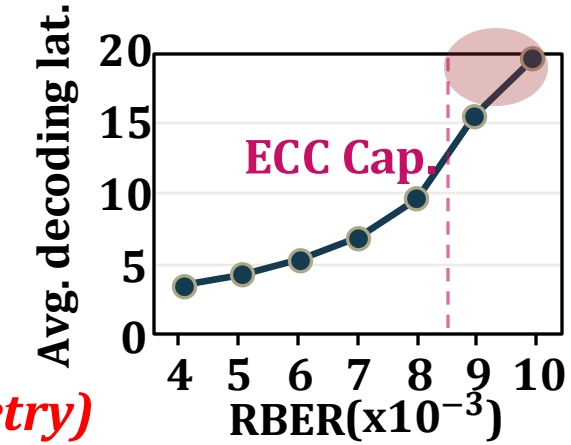


Read-retry still significantly affects SSD performance

Root Cause Analysis

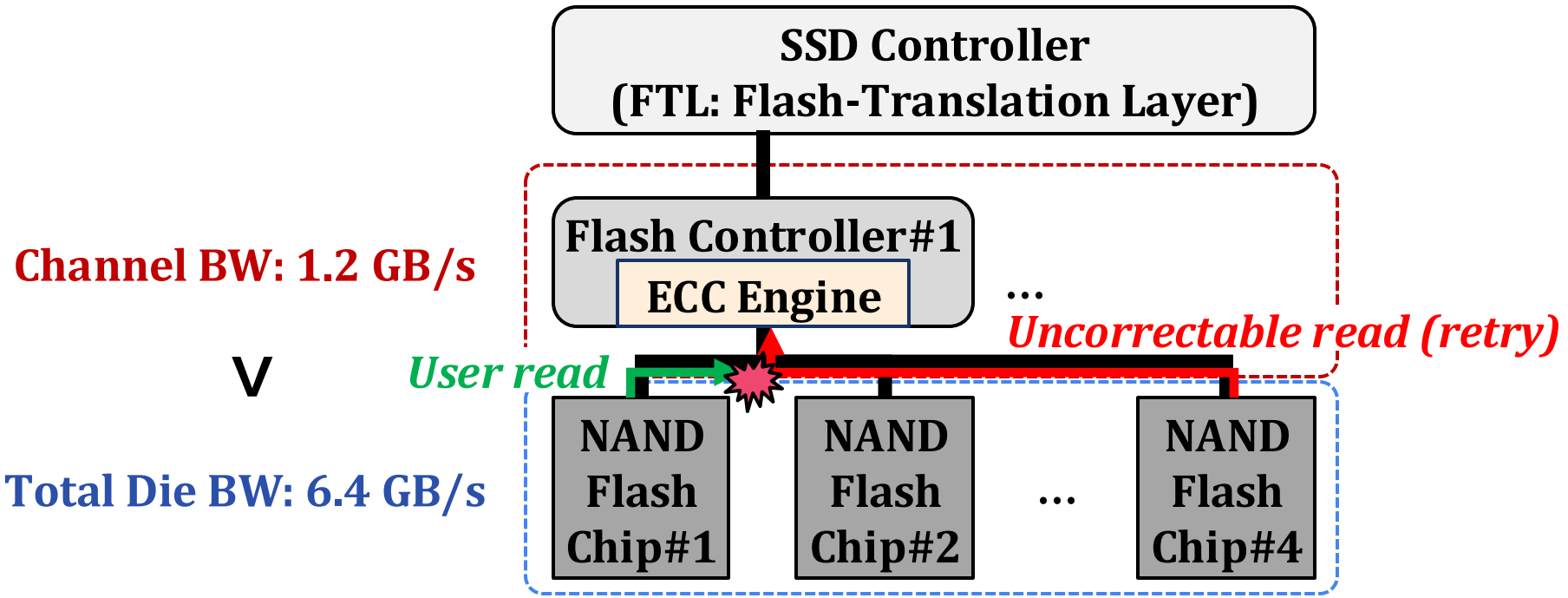
- Significant waste of time for uncorrectable page reads due to contentions at
  - Channel: A limited number of channels shared by a lot more chips
    - Complex ECC (e.g., LDPC) due to high raw bit-error rate (RBER): Hindering on-die ECC
  - ECC engine: The higher the RBER, the longer the decoding latency





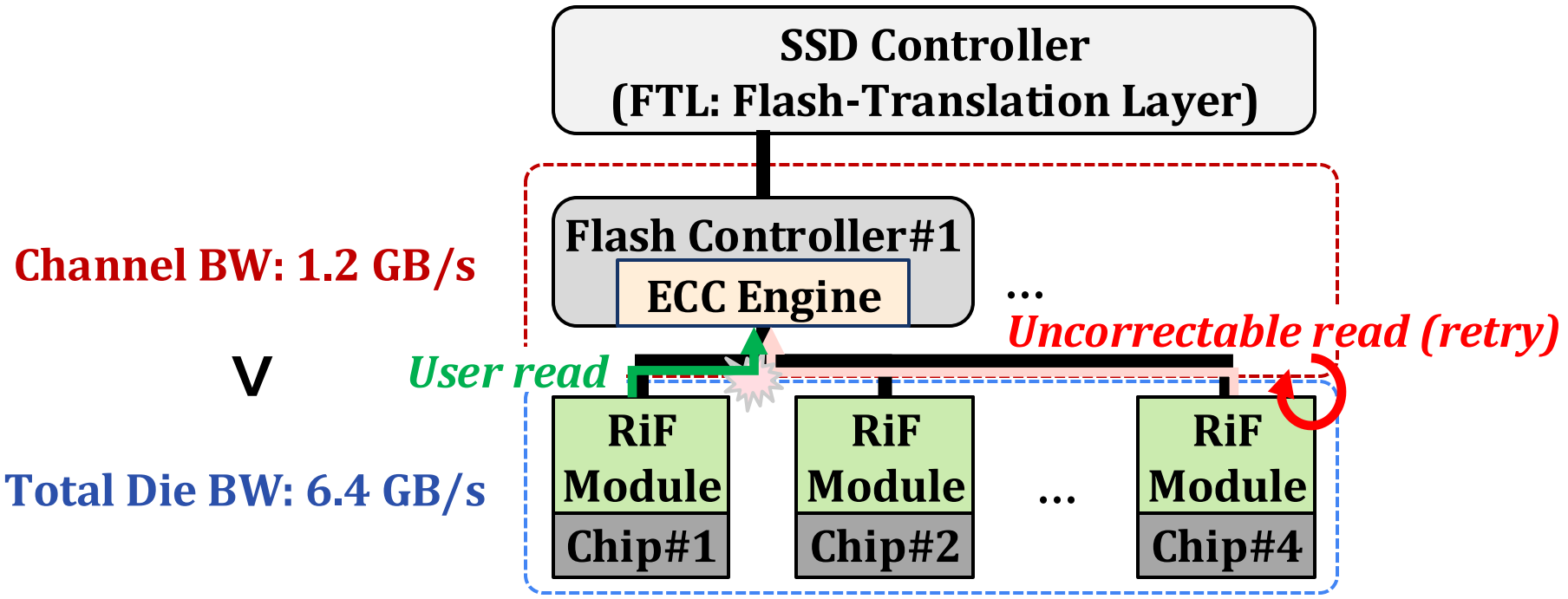
Key Idea: Retry-in-Flash (RiF)

- Performs read-retry inside a NAND flash chip
  - Directly eliminates channel contention
  - Removes ECC decoding for uncorrectable reads off the critical path as well



Key Idea: Retry-in-Flash (RiF)

- Performs read-retry inside a NAND flash chip
  - Directly eliminates channel contention
  - Removes ECC decoding for uncorrectable reads off the critical path as well



Key Challenge and Optimizations

- Challenge: Hard constraints on in-flash RiF module
  - Limited area to put RiF module inside a chip
  - Power can also be another issue
- Optimization 1: Prediction-centric module
  - Insight: Read-retry only requires identification of read-failure
    - But not completion of error correction
  - Proposal: Syndrome weight-based prediction
- Optimization 2: Subpage-based prediction
  - Hypothesis: Raw-bit errors are likely to be evenly distributed within a page
    - All stored data must be randomized in modern NAND flash memory
  - Proposal: Reduce input size for prediction → Further area reduction

Syndrome Weight-Based Prediction

- Key insight: Read-retry does not require entire ECC decoding,
  but only identification of whether the page is correctable

H = a parity check matrix, c = a codeword = $(c_0, c_1 \ldots c_{N-1})$, S = a syndrome vector = $(s_0, s_1, s_2, \ldots s_{M-1})$

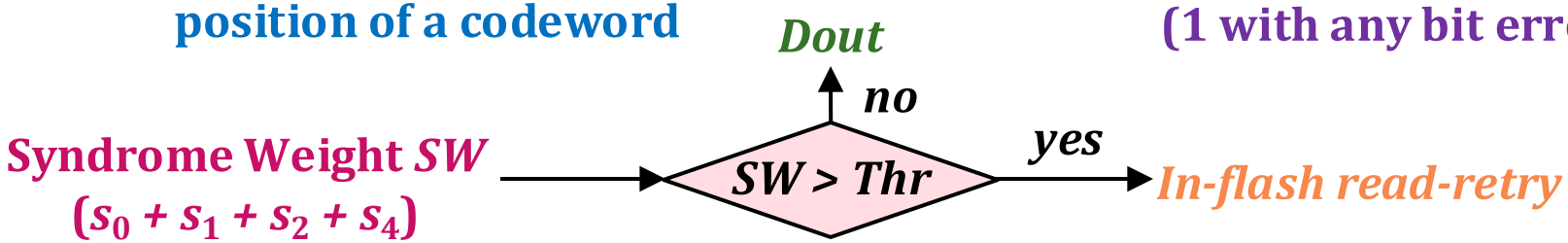
$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

M = 4, N = 8

1's represent the bit position of a codeword

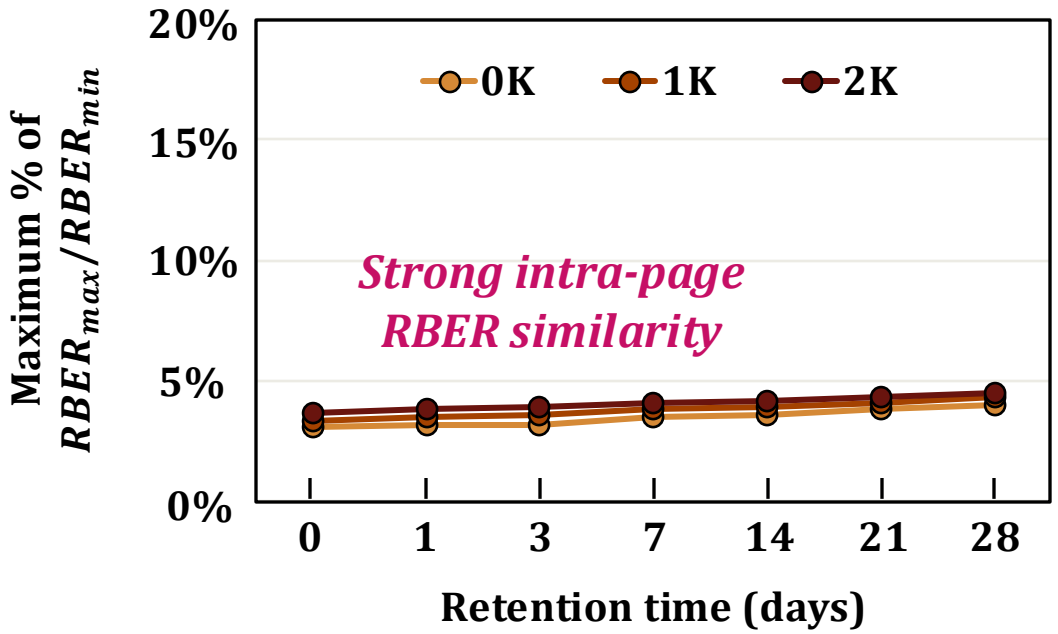
$$S^T = Hc^T = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} c_1 \oplus c_3 \oplus c_4 \oplus c_7 \\ c_0 \oplus c_1 \oplus c_2 \oplus c_5 \\ c_2 \oplus c_5 \oplus c_6 \oplus c_7 \\ c_0 \oplus c_3 \oplus c_4 \oplus c_6 \end{bmatrix}$$

S3 is 0 with no errors in *c0, c3, c4, c6*
(1 with any bit error)

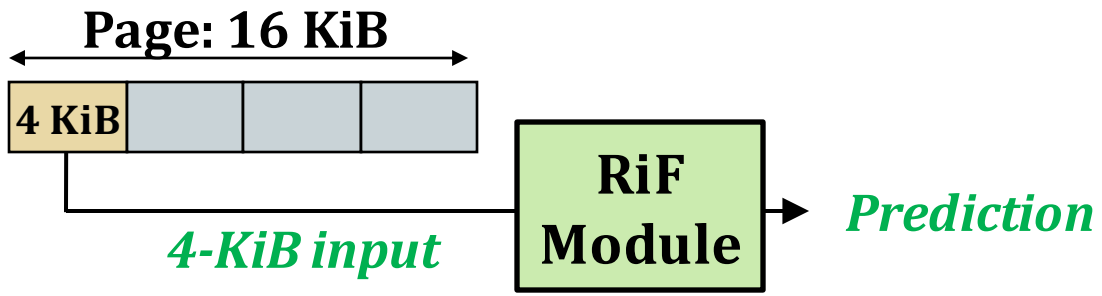


Subpage-Based Prediction

- Hypothesis: Errors are likely to be evenly distributed within a page, which allows SW-based prediction w/ part of the data

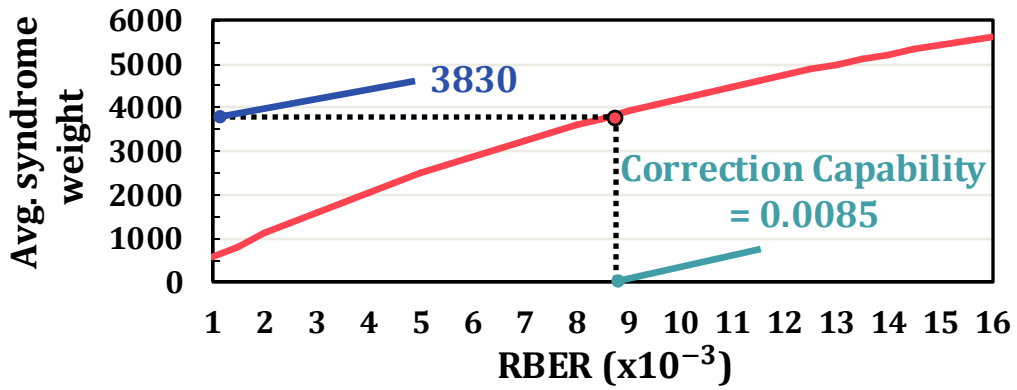


RBER dist. of 4-KiB chunks in a 16-KiB page

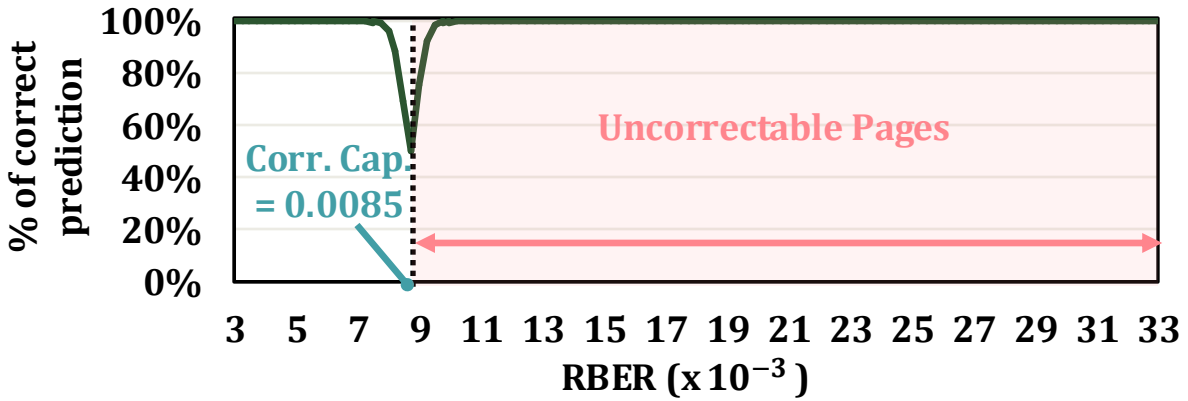


RiF: Validation

- RiF module achieves 98.7% prediction accuracy



✓ *Strong correlation*

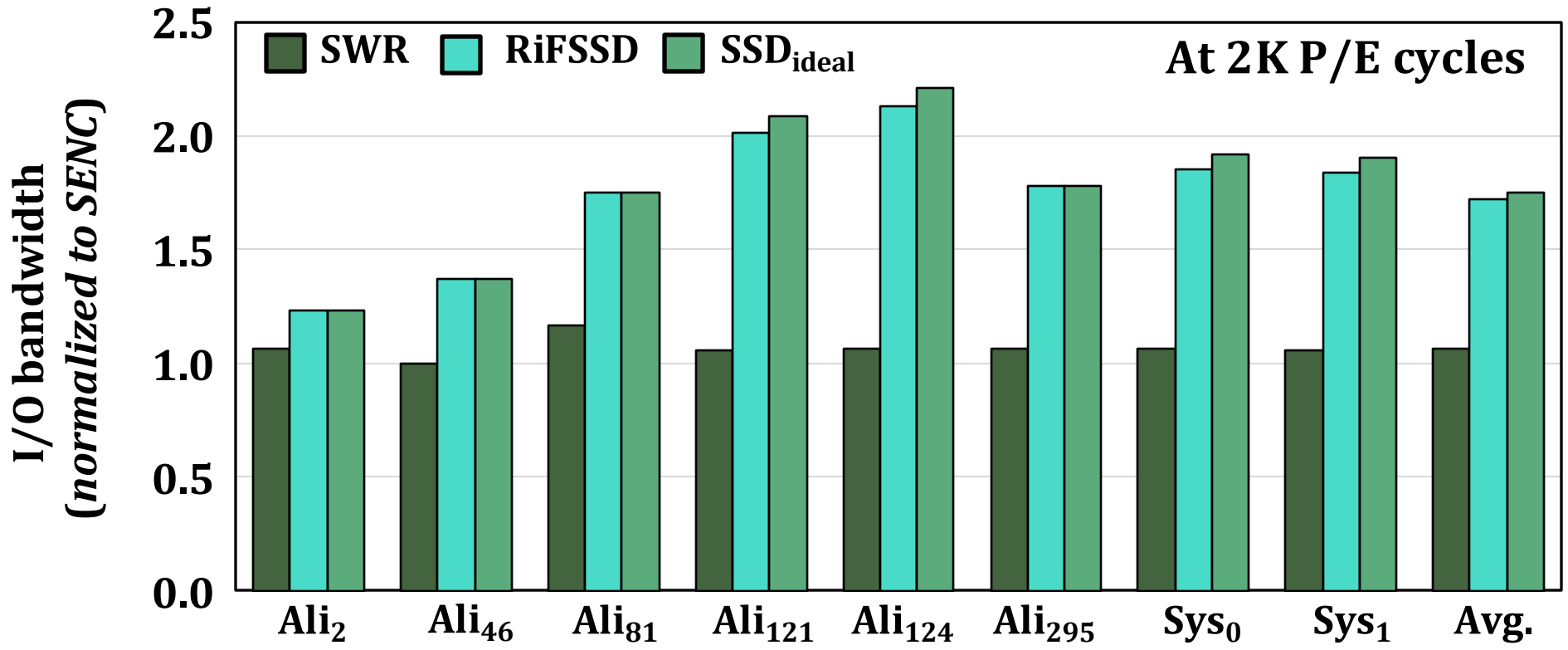


✓ *High prediction accuracy*

Evaluation Methodology

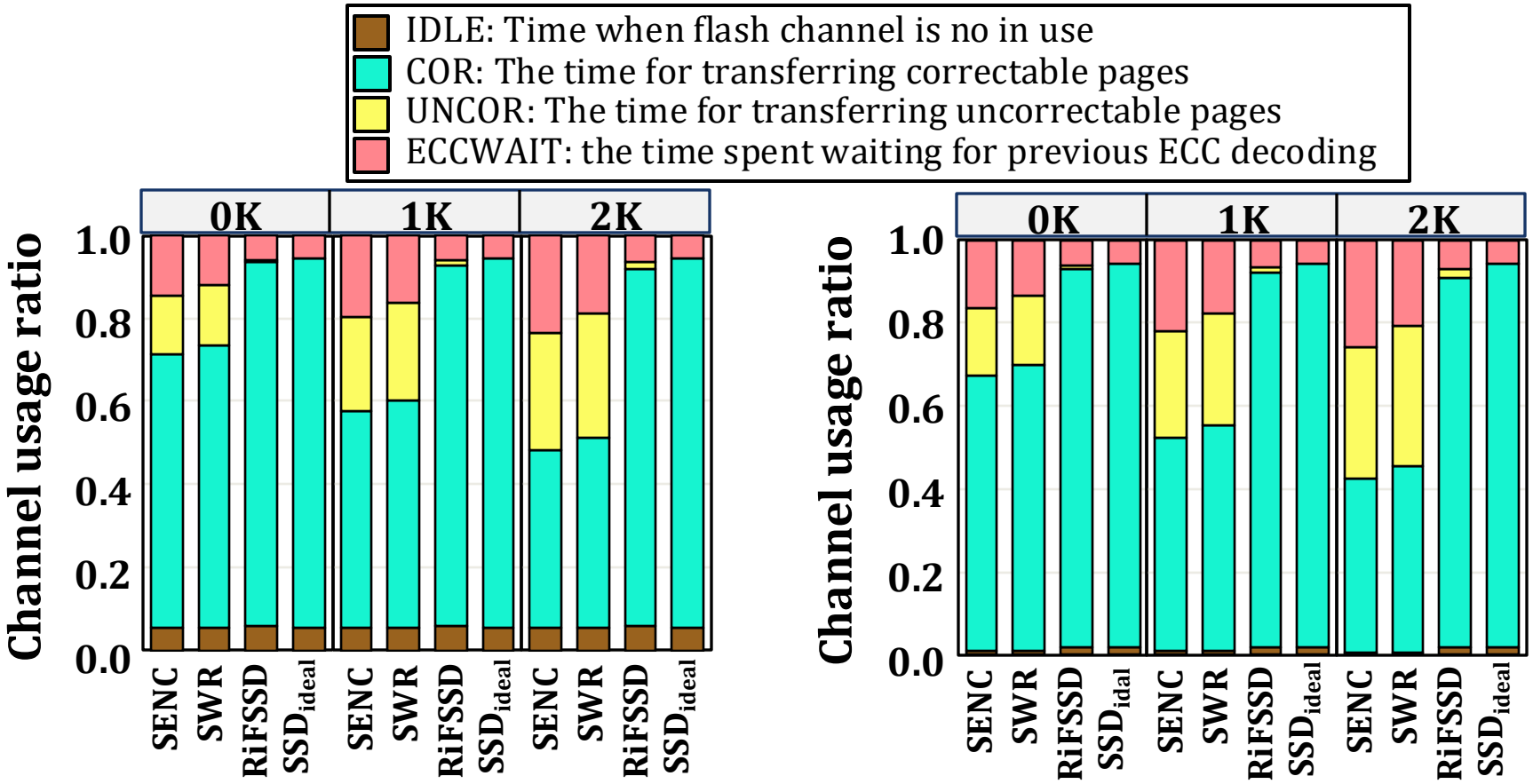
- Simulator: MQSim-E [IEEE CAL'22]
  - Extend NAND flash models w/ real-device characterization results
- Workloads: 8 real-world traces
  - 6 from AliCloud traces
  - 2 from Systor trace
- Comparisons with
  - SSD_{ideal}: No read-retry
  - SENC: w/ Sentinel [MICRO'21]
  - SWR: Swift-Read [ISSCC'22]
  - RiFSSD: Our proposal

Result: I/O Bandwidth



- RiF significantly improves I/O bandwidth by 71.2% and 61.2% on average over SENC and SWR, respectively

Result: Channel Usage Breakdown



- RiF significantly reduces wasted channel bandwidth due to UNCOR and ECCWAIT compared to SENC and SWR

RiF: Summary

- Problem: Significant performance degradation due to read retry
  - Even w/ SotA mitigation techniques
  - Due to wastes for uncorrectable reads

- Key idea: Retry-in-Flash (RiF)
  - Performs read-retry inside a NAND flash chip
  - Syndrome weight-based prediction of read failure

- Results: Significant performance improvement (average 72.1% at 2K P/E cycles) compared to SotA

Outline

- Ultra Large-Capacity NAND Flash-Based SSDs
- RiF: Improving Read Performance of Modern SSDs Using an On-Die Early-Retry Engine
- AERO: Adaptive Erase Operation for Improving Lifetime and Performance of Modern NAND Flash-Based SSDs
- Conclusion

AERO: Adaptive Erase Operation

AERO: Adaptive Erase Operation for Improving Lifetime and Performance of Modern NAND Flash-Based SSDs

Sungjun Cho
allencho1222@postech.ac.kr
POSTECH
Republic of Korea

Beomjun Kim
beomjun0816@knu.ac.kr
Kyungpook National University
Republic of Korea

Hyunuk Cho
gusdnr9779@postech.ac.kr
POSTECH
Republic of Korea

Gyeongseob Seo
syhbong9@knu.ac.kr
Kyungpook National University
Republic of Korea

Onur Mutlu
omutlu@gmail.com
ETH Zürich
Switzerland

Myungsuk Kim
ms.kim@knu.ac.kr
Kyungpook National University
Republic of Korea

Jisung Park
jisung.park@postech.ac.kr
POSTECH
Republic of Korea

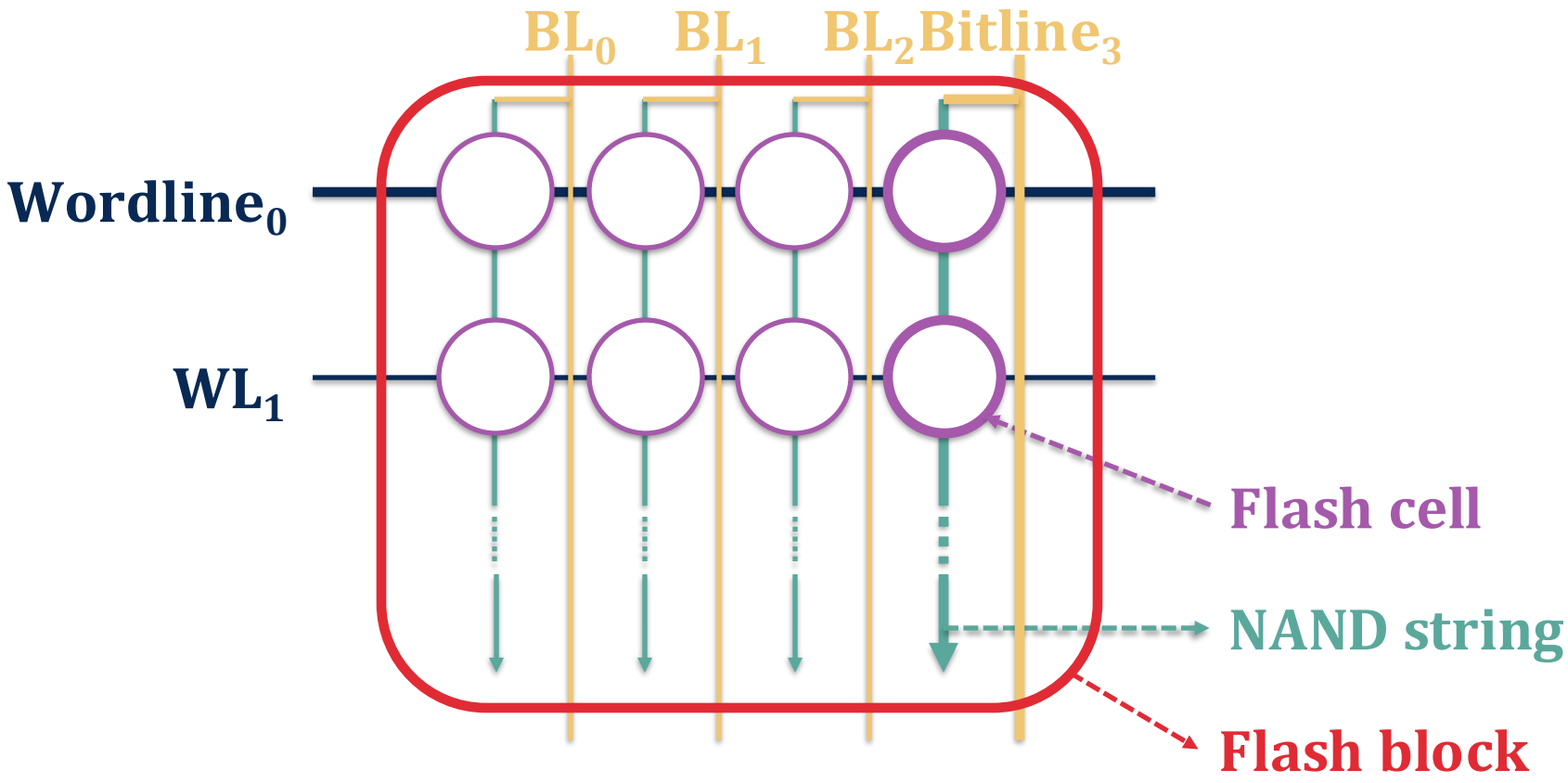
The 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)

AERO: Outline

- Erase Operation in Modern NAND Flash-Based SSDs
- Key Idea: Adaptive Erase Operation (AERO)
- Evaluation Results
- Summary

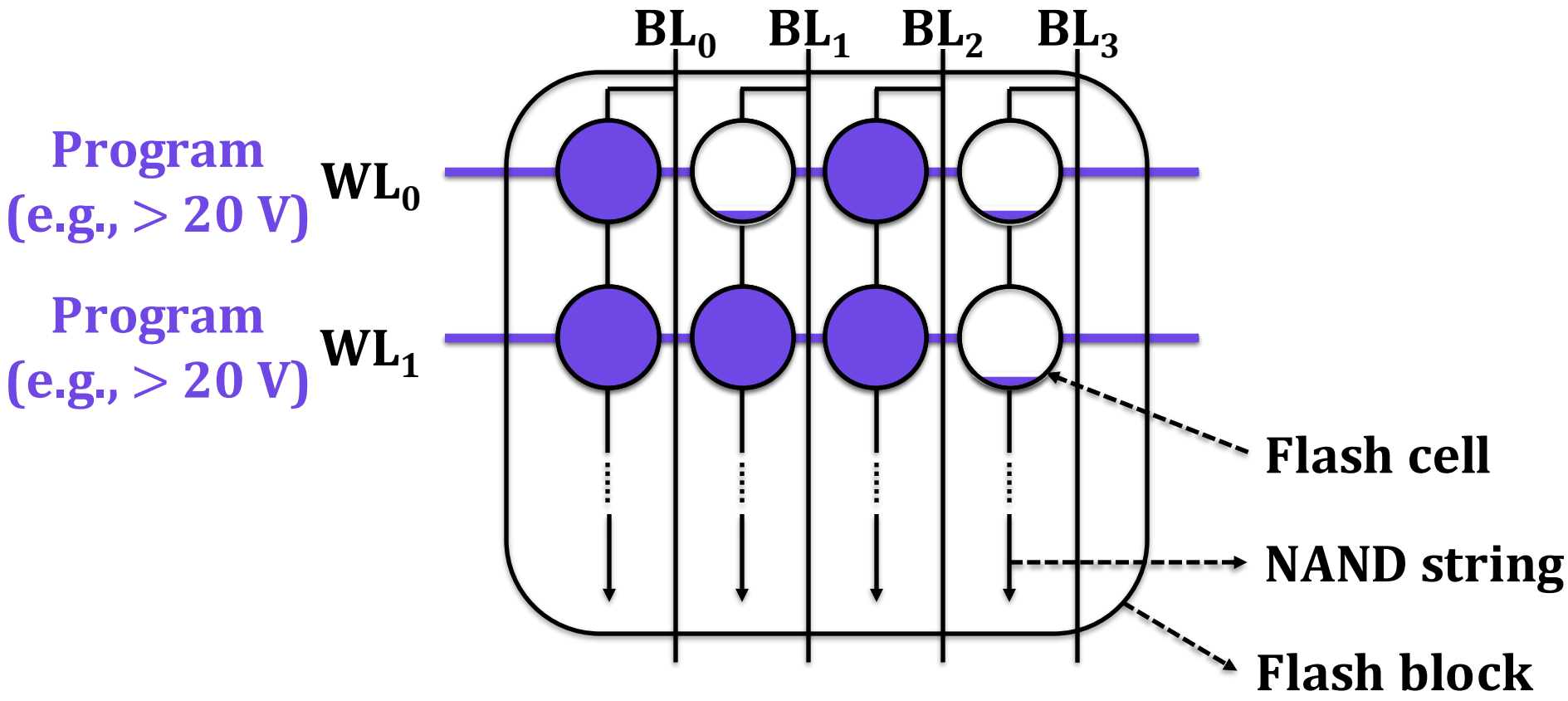
NAND Flash Organization

- NAND flash memory is hierarchically organized



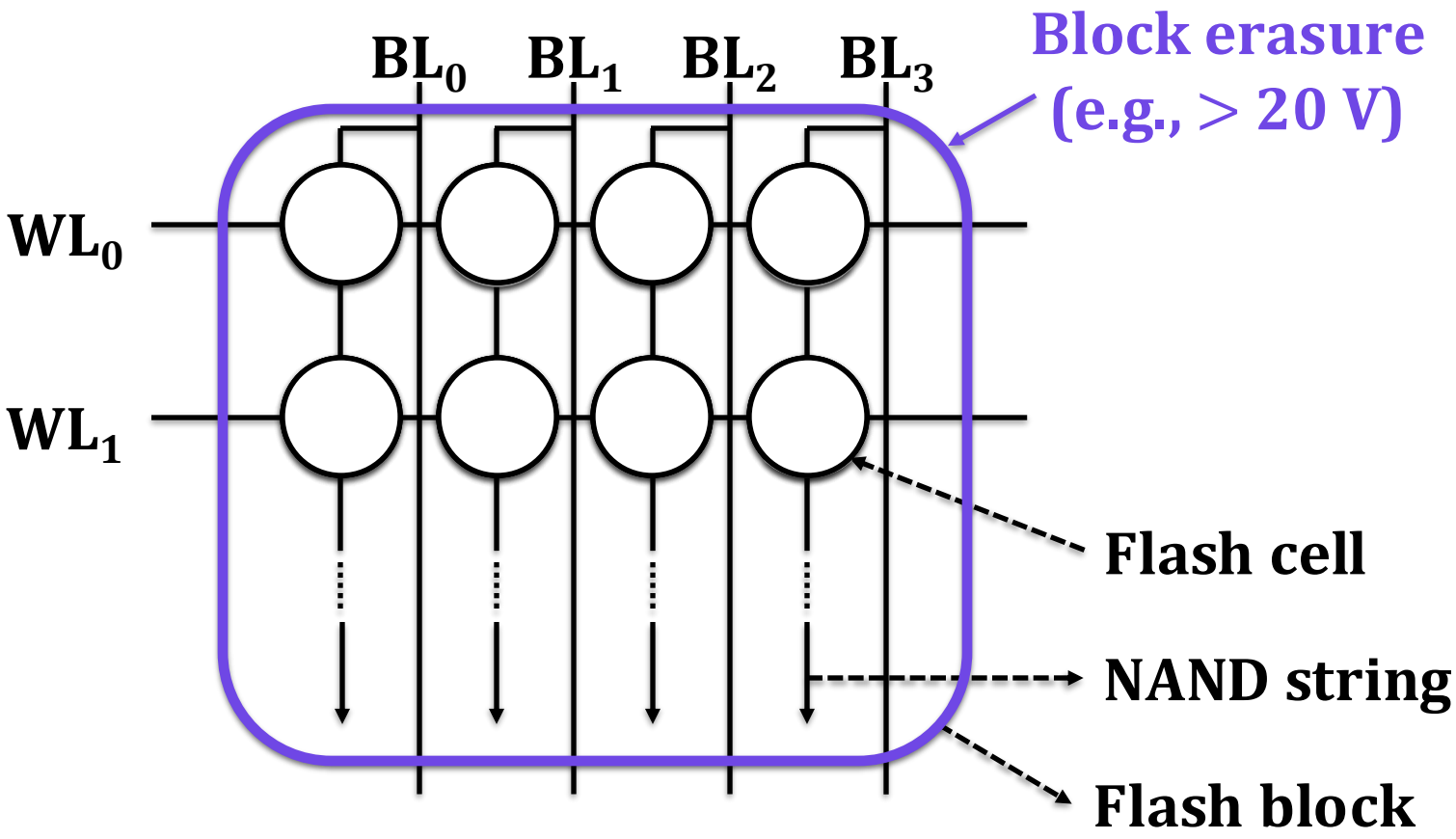
Program and Erase Operations

- Program operation writes data in a WL granularity



Program and Erase Operations

- Erase operation erases data in a **block granularity**



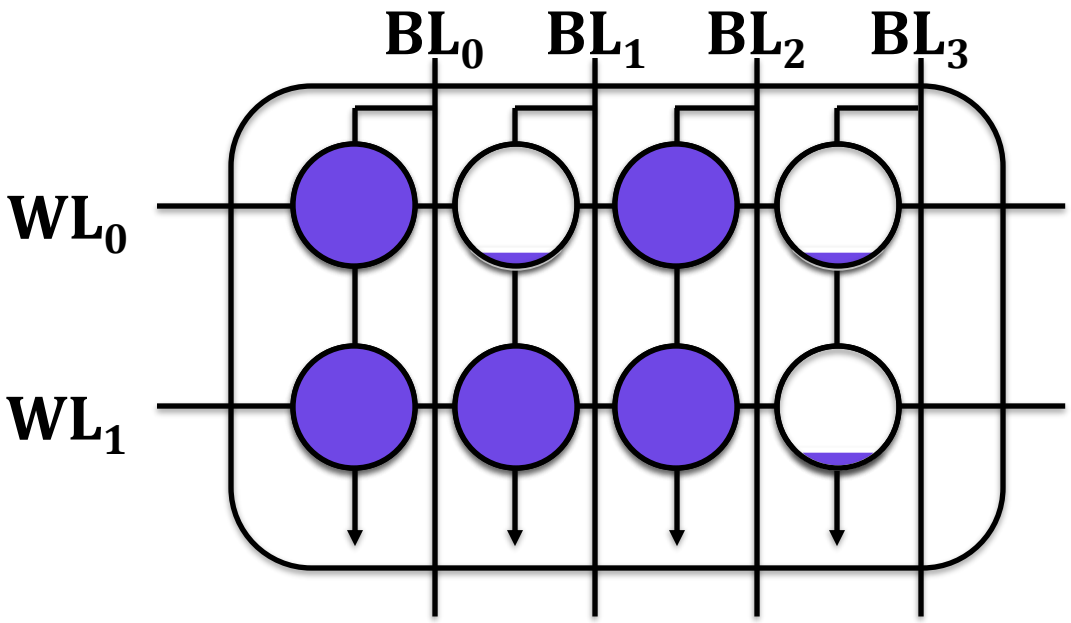
Program and Erase Operations

- Erase operation erases data in a **block granularity**

Erase (program) operation only ejects (injects) electrons:
Erase-before-write property (data cannot be overwritten)

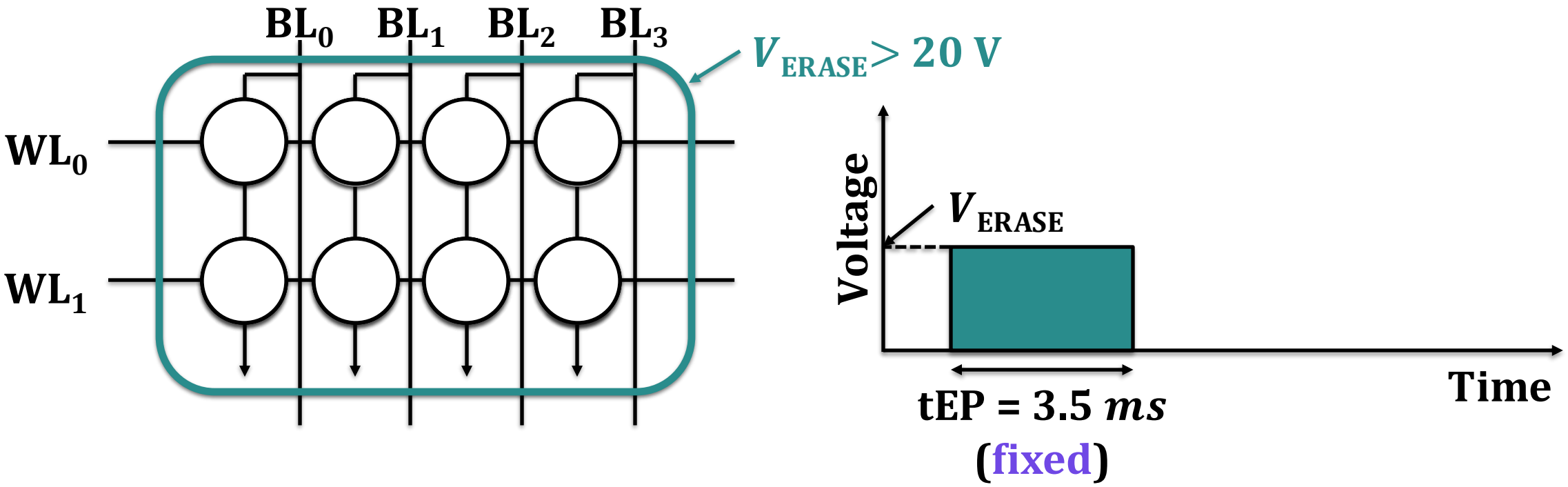
Erase Operation: More Details

- Erase operation consists of two steps
  - Erase pulse (EP)
  - Verify read (VR)



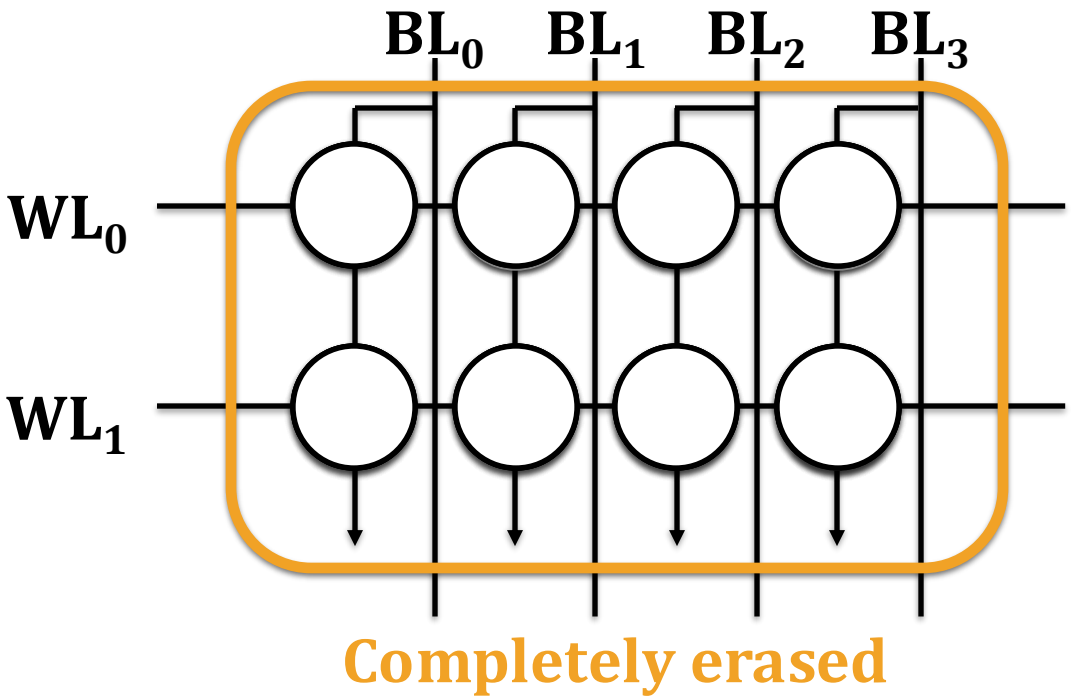
Erase Operation: More Details

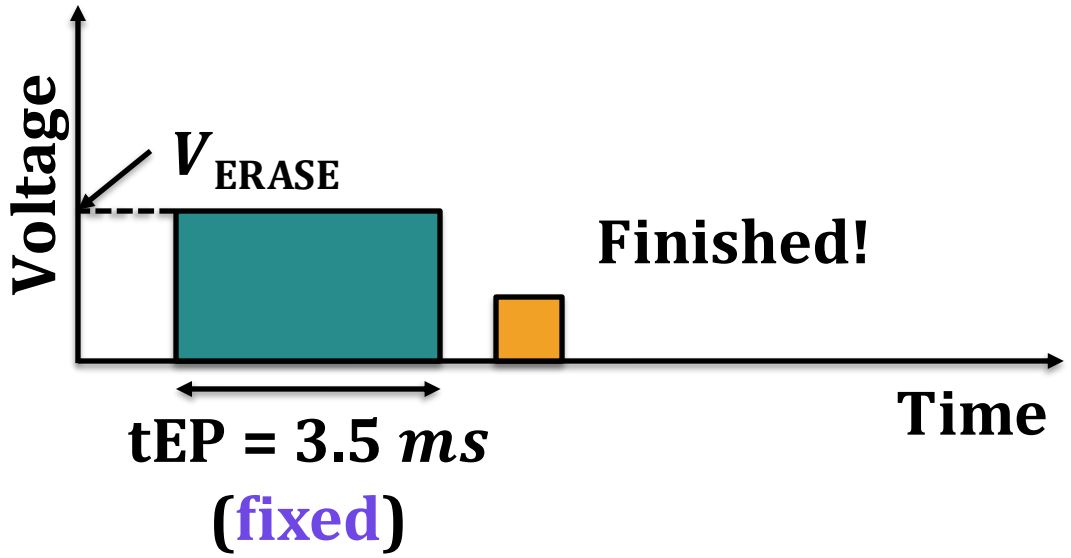
- Erase operation consists of two steps
  - Erase pulse (EP): Applies high voltage for fixed latency (3.5 ms)
  - Verify read (VR)



Erase Operation: More Details

- Erase operation consists of two steps
  - Erase pulse (EP): Applies high voltage for fixed latency (3.5 ms)
  - Verify read (VR): Checks if a block is completely erased





Erase Operation: More Details

- Erase operation consists of two steps

Long latency and high voltage in erase operations

Lifetime: A block becomes unusable after experiencing a certain number of program and erase cycles (P/E cycles)

Erase Operation: More Details

- Erase operation consists of two steps

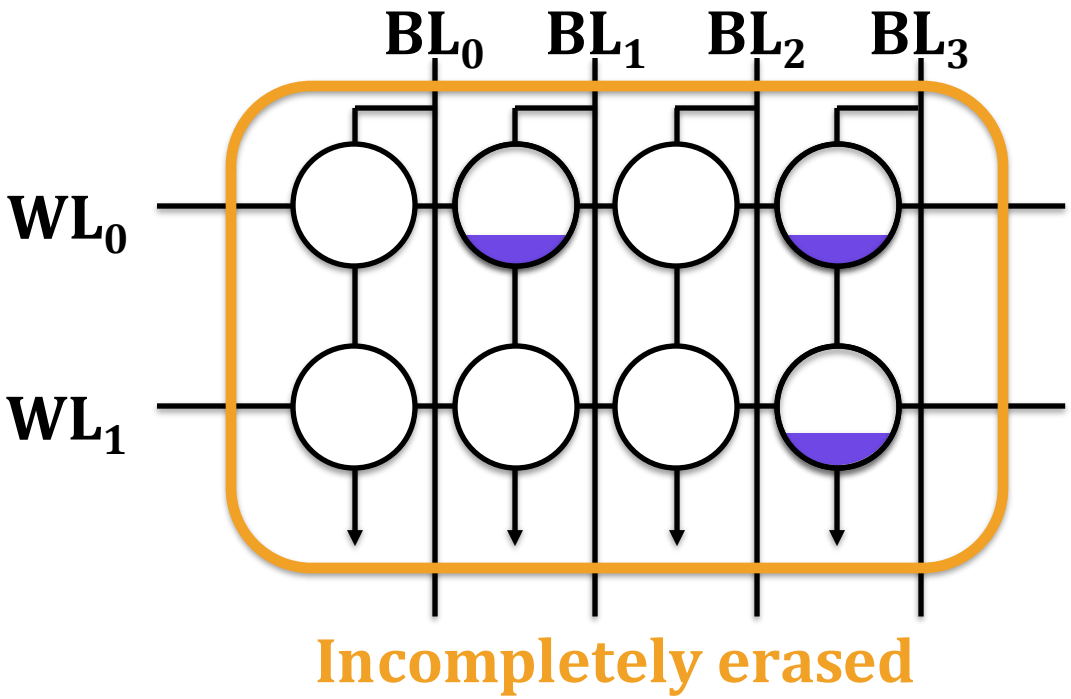
Long latency and high voltage in erase operations

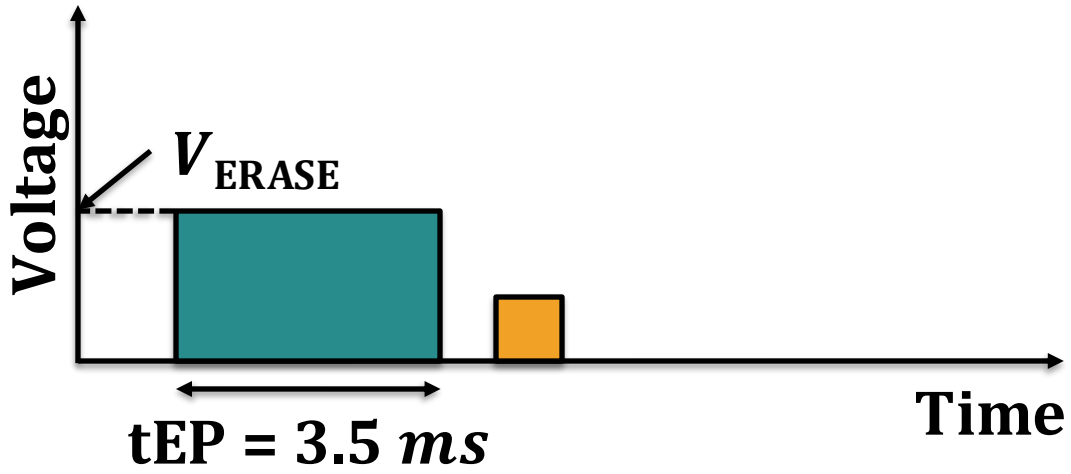
Lifetime: A block becomes unusable after experiencing a certain number of program and erase cycles (P/E cycles)

Performance: An erase operation can delay user requests for a long time, significantly increasing read tail latency

Erase Failure

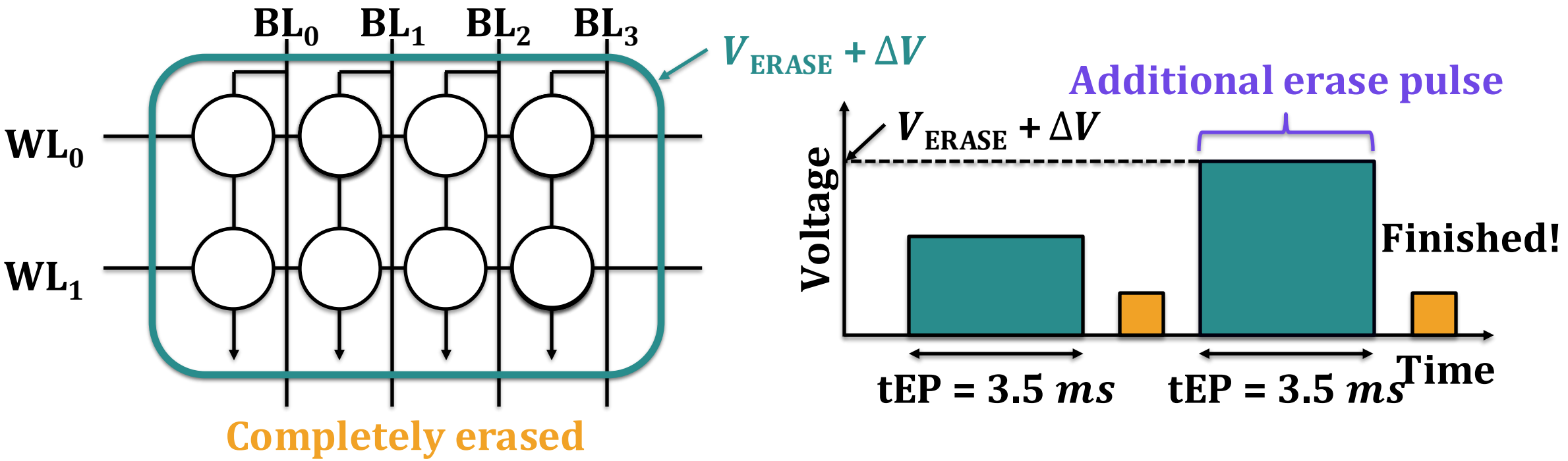
- As P/E cycle increases, erasing a block becomes more difficult





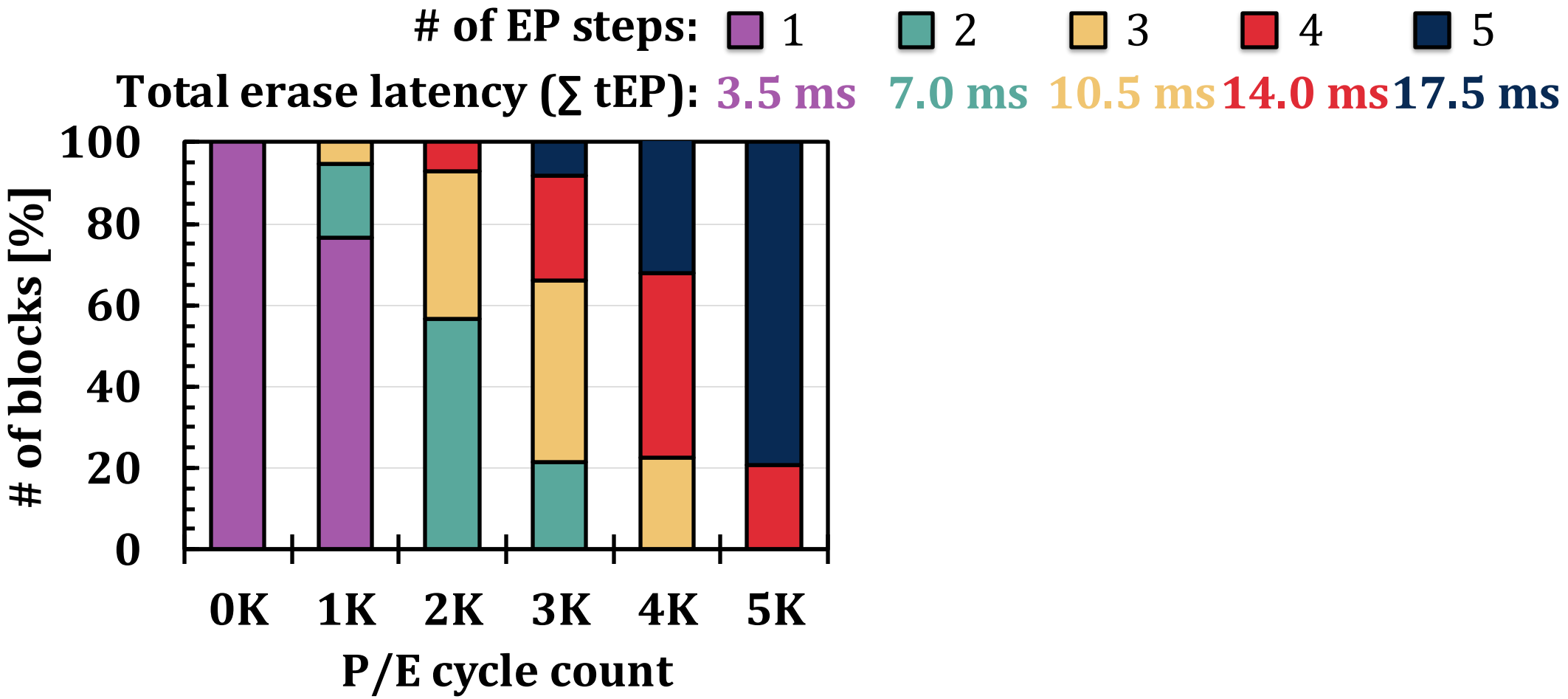
Incremental Step Pulse Erasure (ISPE)

- As P/E cycle increases, erasing a block becomes more difficult
  - ISPE: Performs additional erase pulse with higher voltage



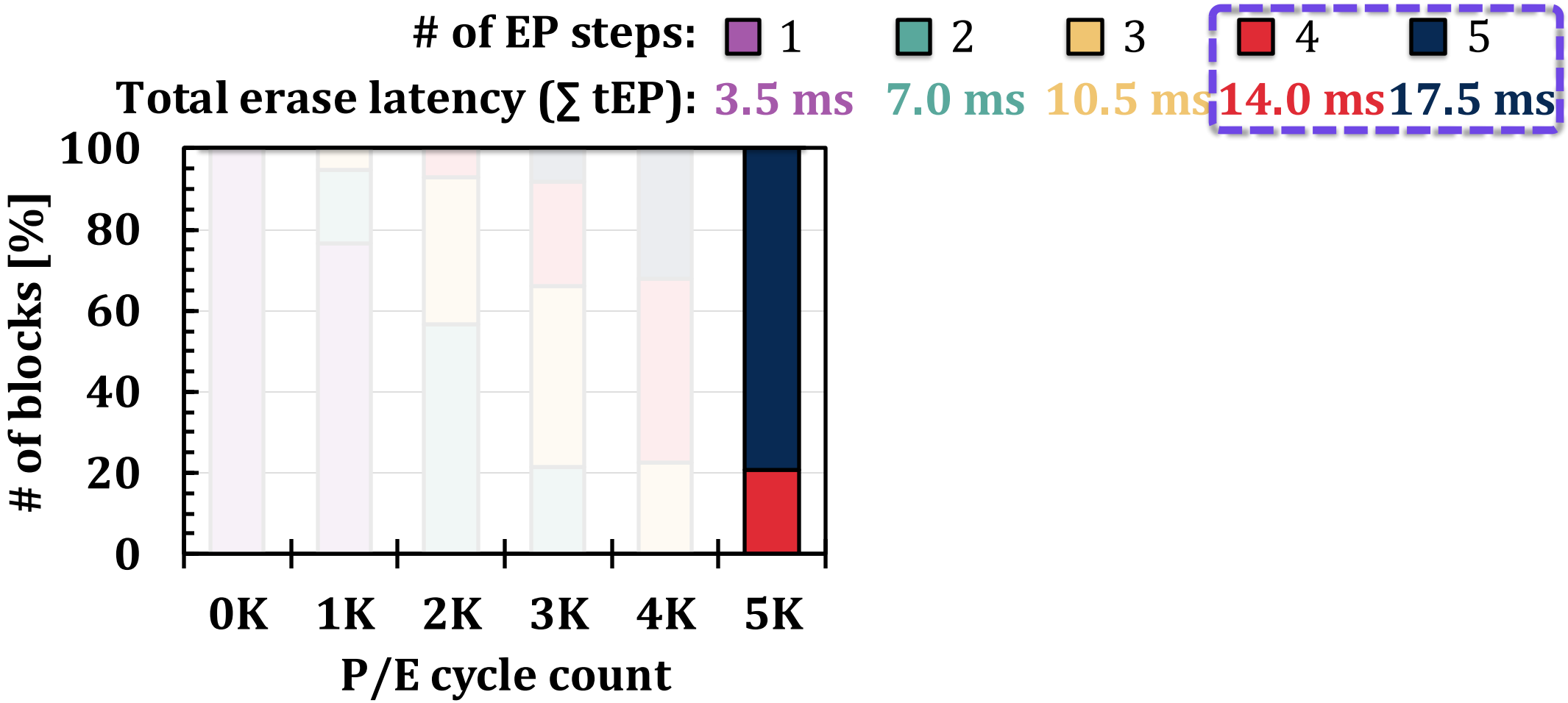
High Latency of Erase Operation

- Erase failure frequently occurs as P/E cycle increases



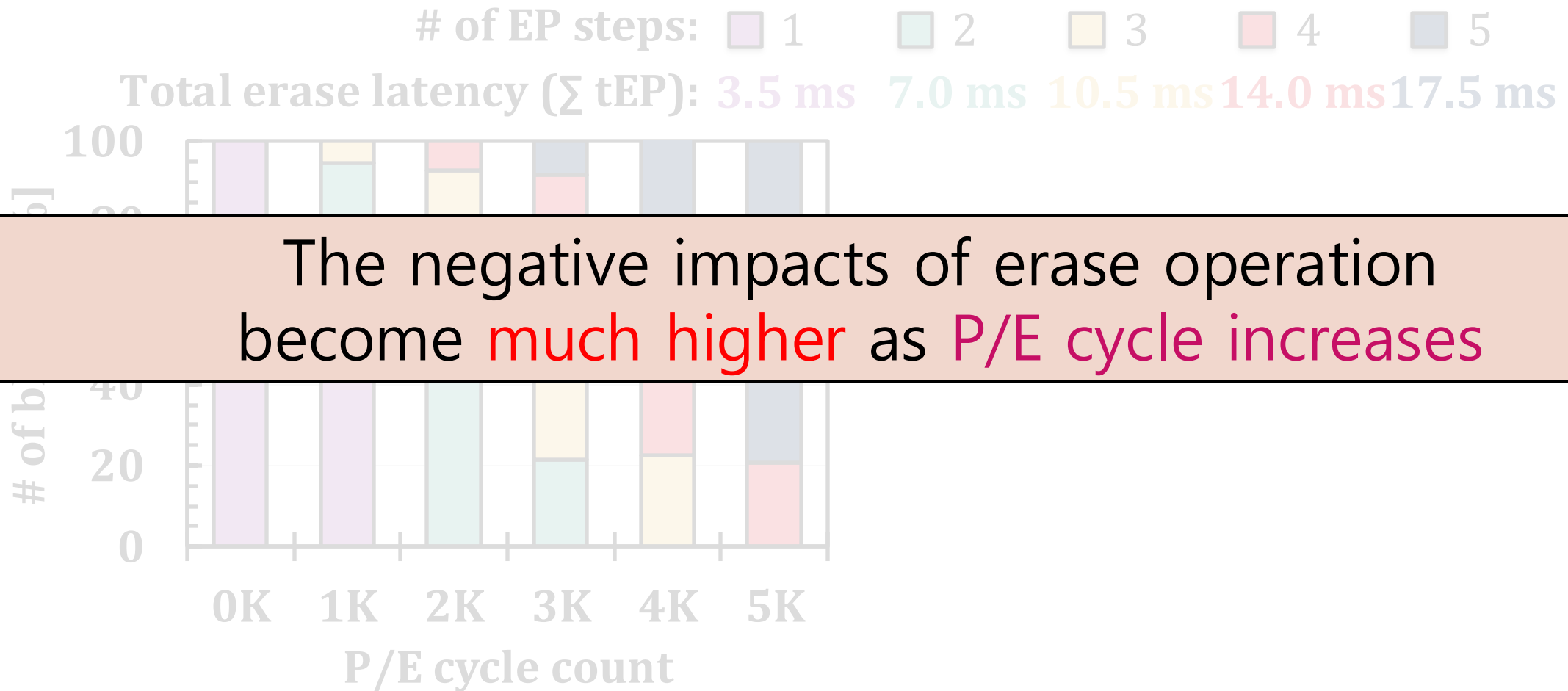
High Latency of Erase Operation

- Erase failure frequently occurs as P/E cycle increases



High Latency of Erase Operation

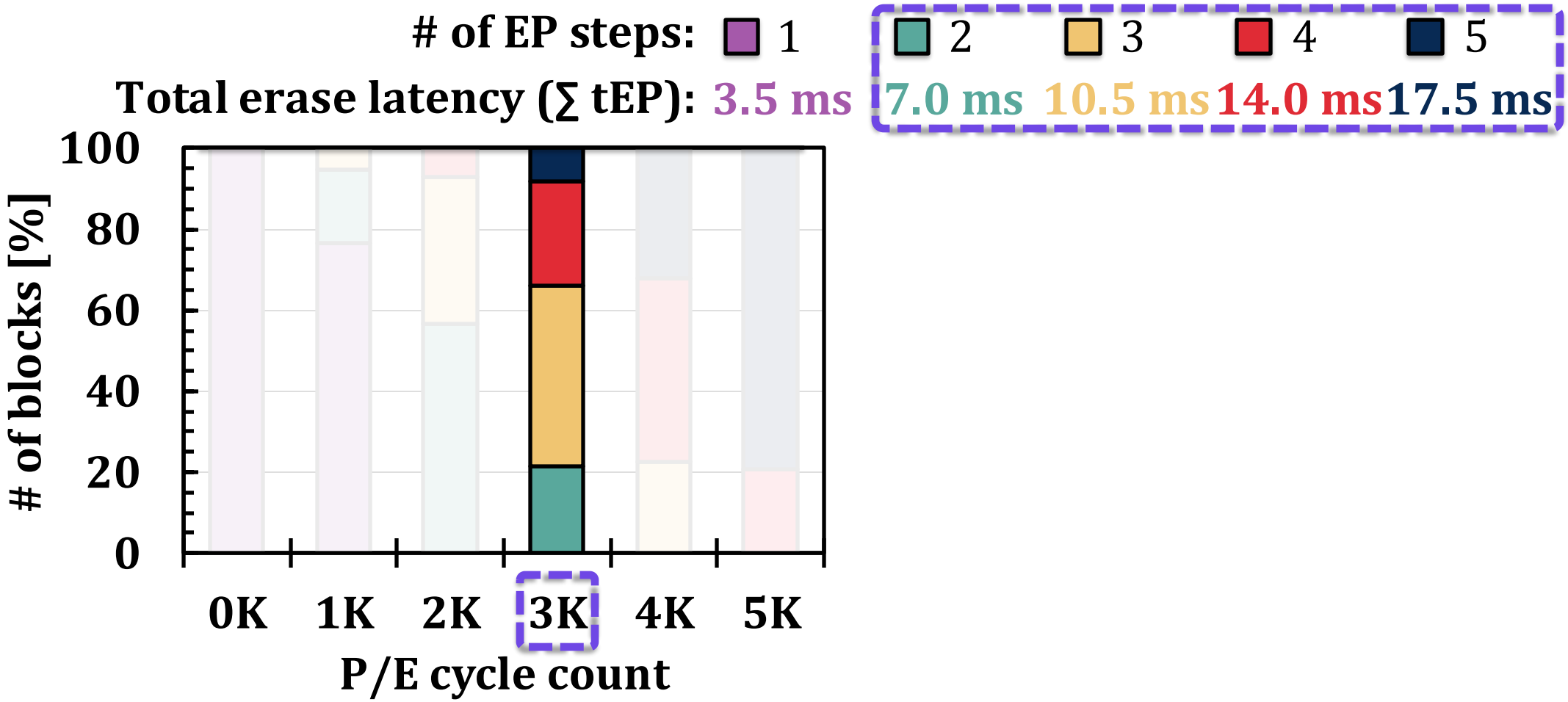
- Erase failure frequently occurs as P/E cycle increases



The negative impacts of erase operation become much higher as P/E cycle increases

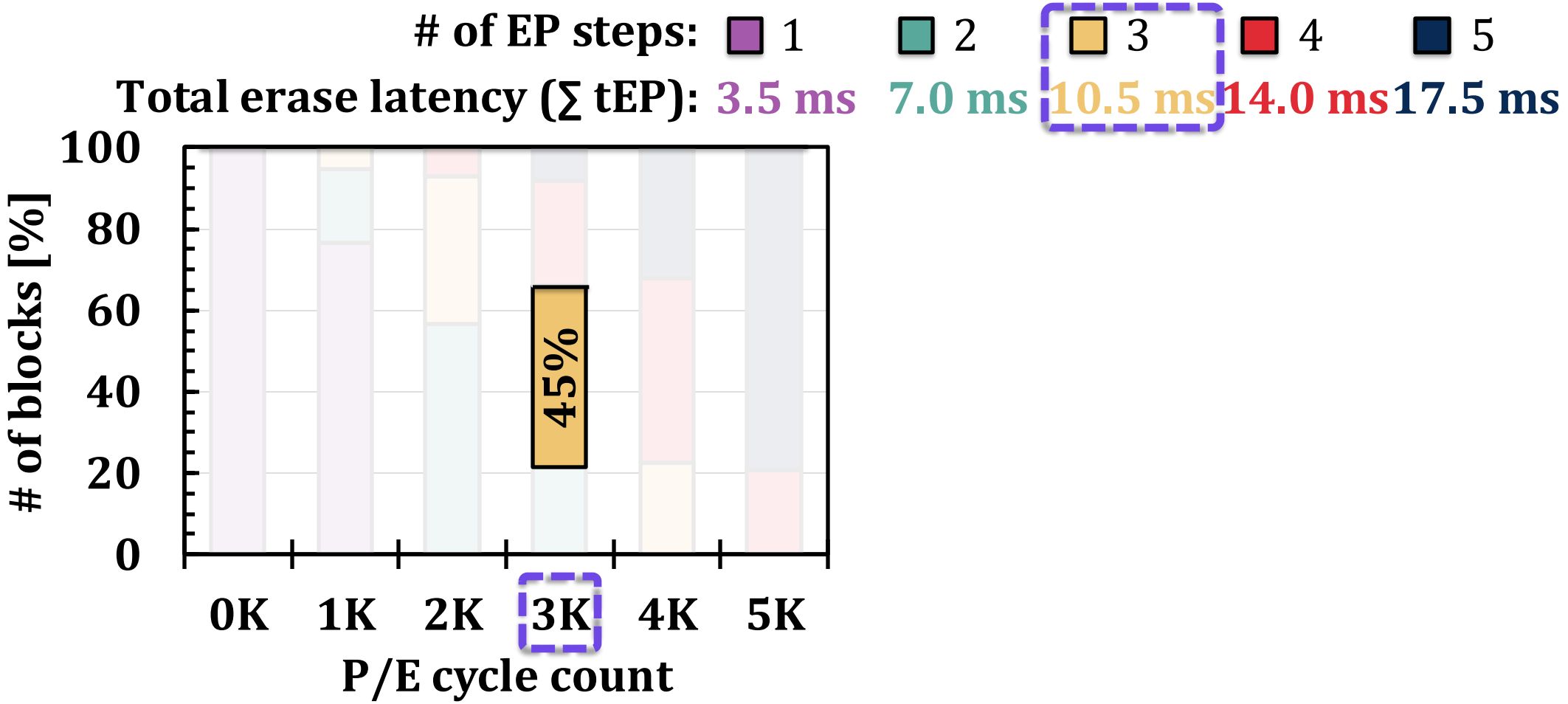
Erase Latency Variation

- Actual erase latency significantly varies across flash blocks



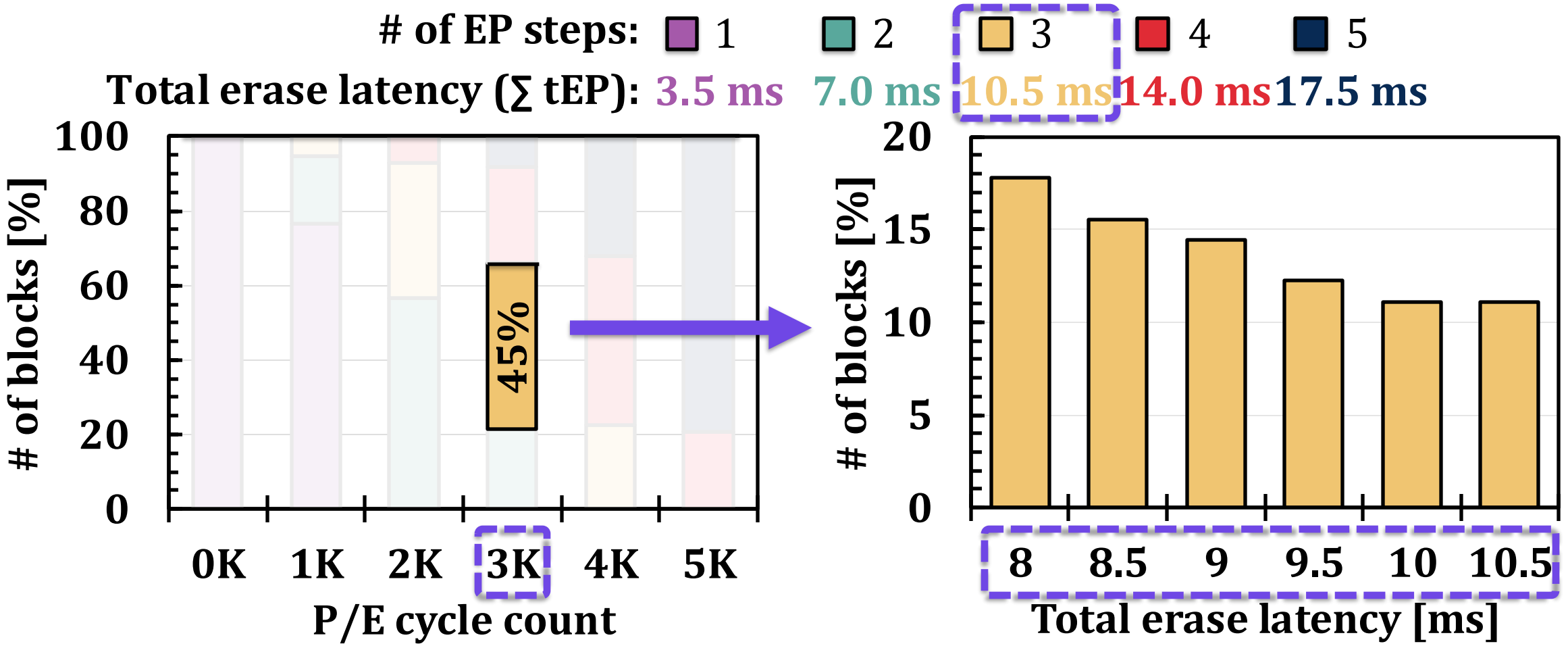
Erase Latency Variation

- Actual erase latency significantly varies across flash blocks



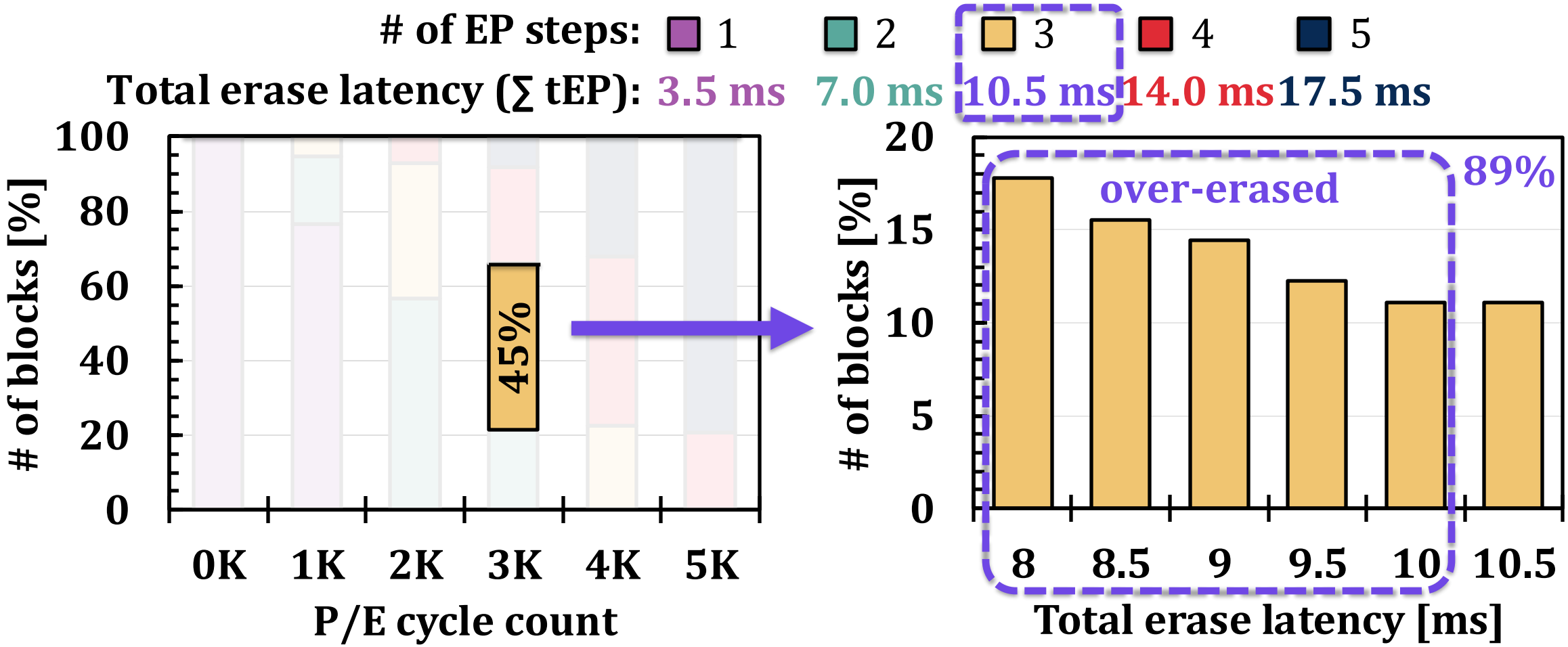
Erase Latency Variation: Key Problem

- Actual erase latency significantly varies across flash blocks



Erase Latency Variation: Key Problem

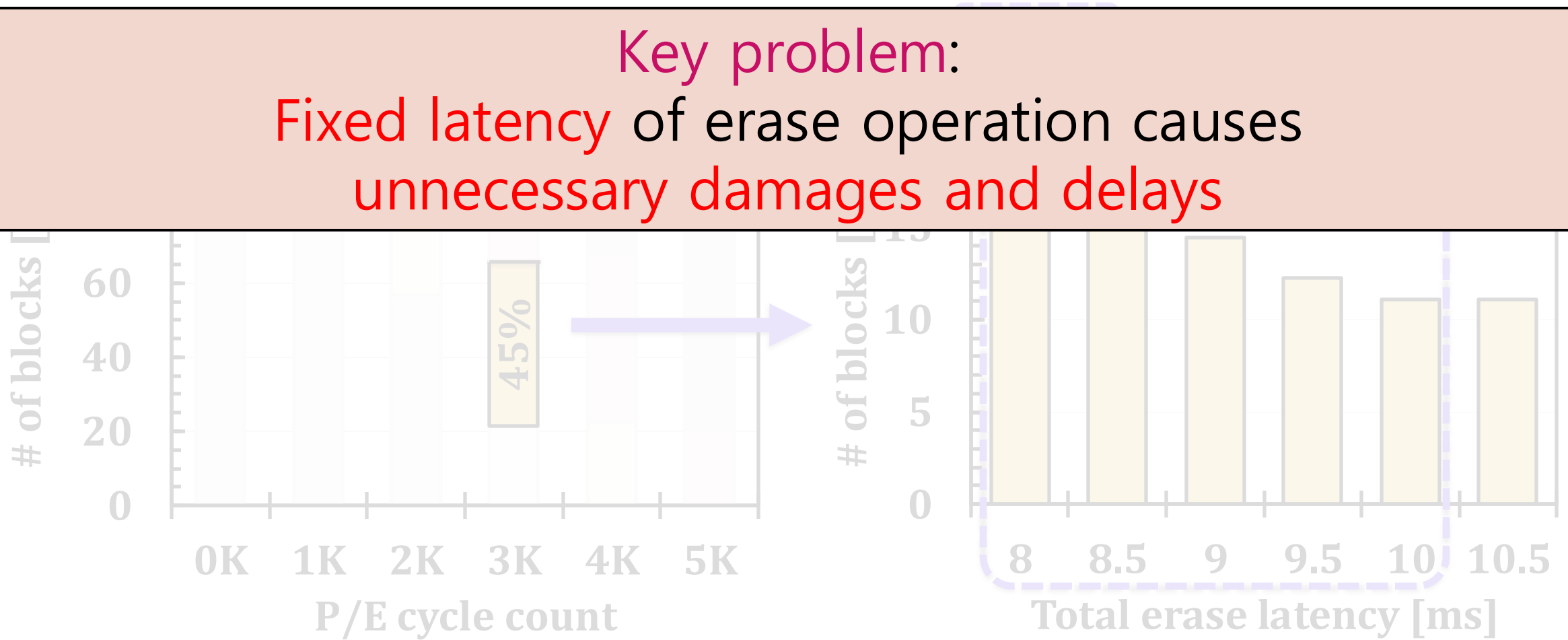
- Fixed latency of erase operation causes over-erase



Erase Latency Variation: Key Problem

- Fixed latency of erase operation causes over-erase

Key problem:
Fixed latency of erase operation causes
unnecessary damages and delays



Erase Latency Variation: Key Problem

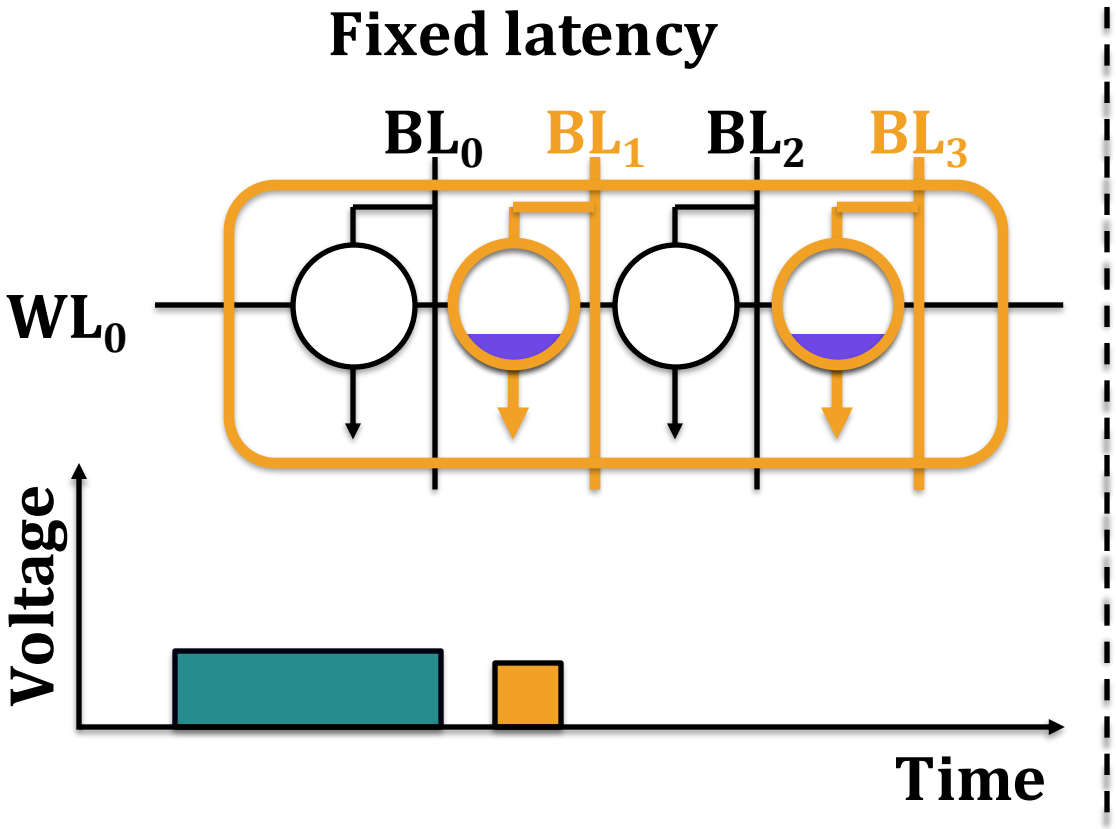
- Fixed latency of erase operation causes over-erase

Key problem:
Fixed latency of erase operation causes
unnecessary damages and delays

Goal:
Minimize erase latency to improve
lifetime and I/O performance

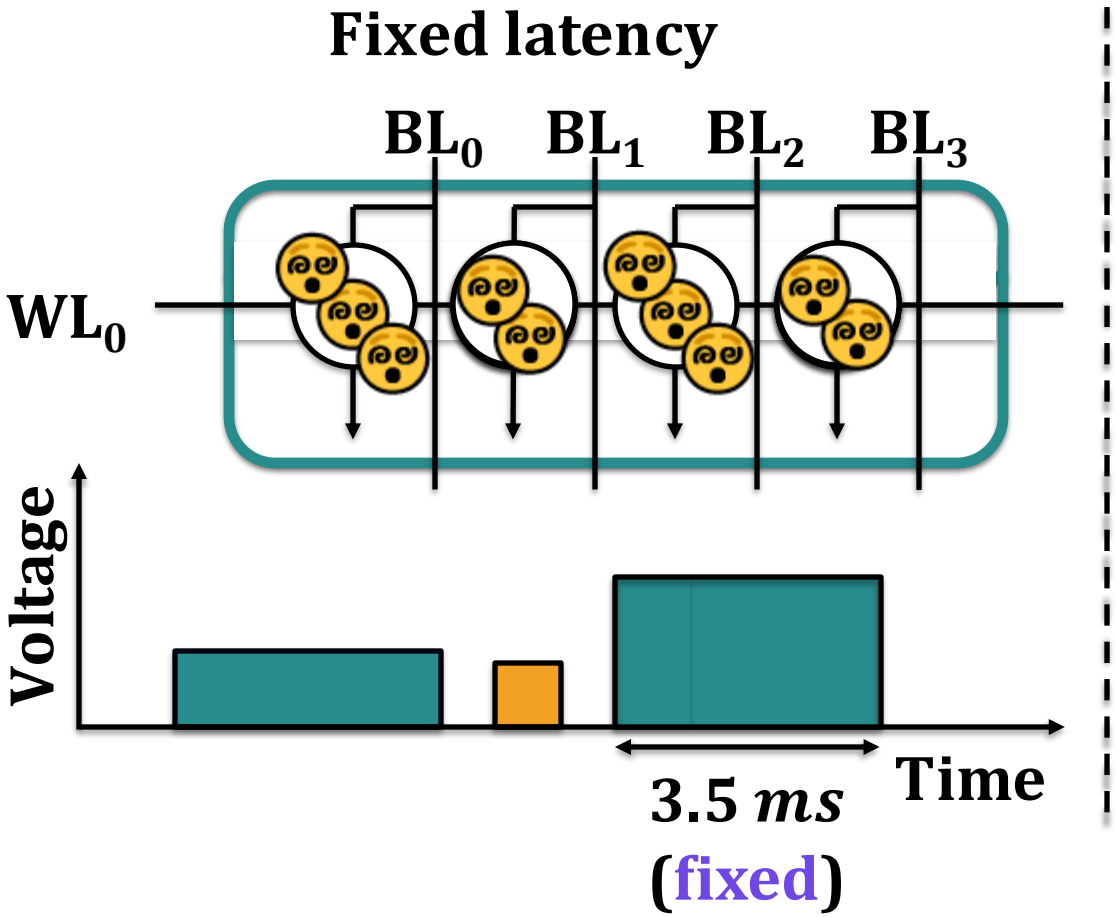
Adaptive Erase Operation (AERO)

- Fixed latency of erase operation causes over-erase



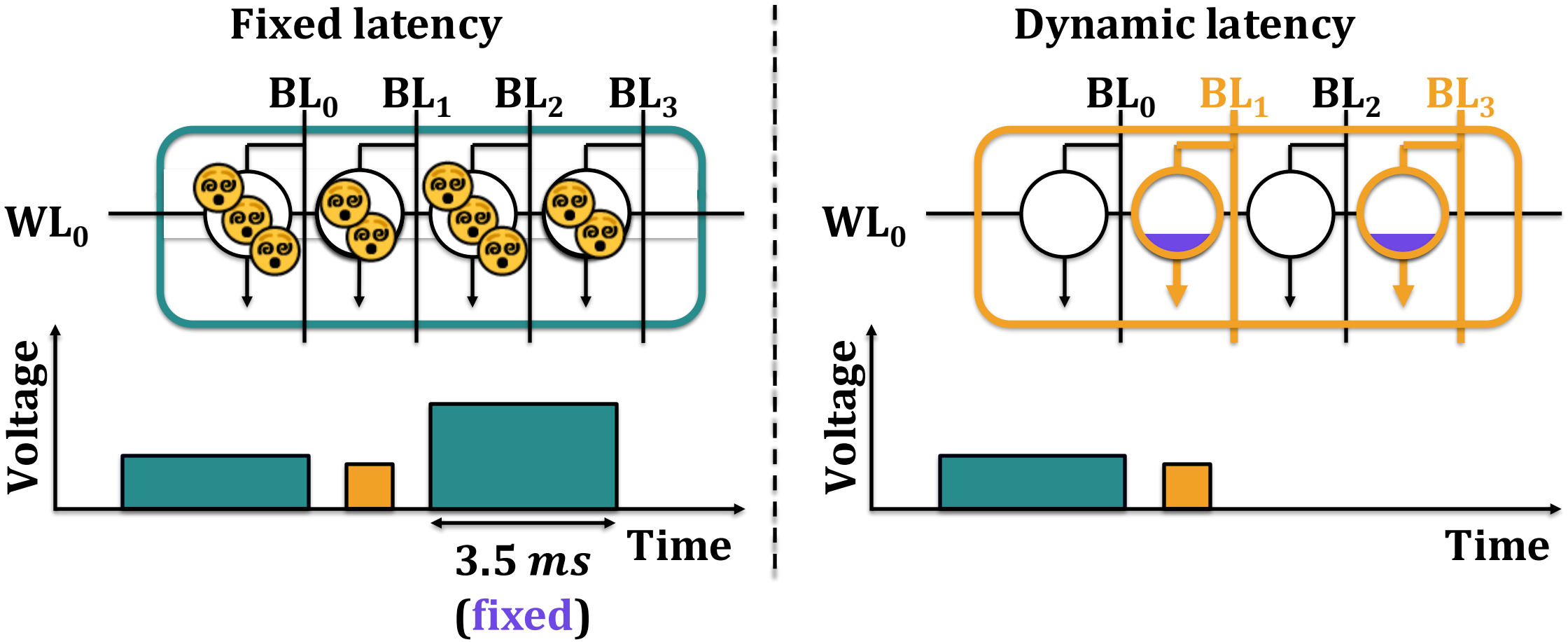
Adaptive Erase Operation (AERO)

- Fixed latency of erase operation causes over-erase



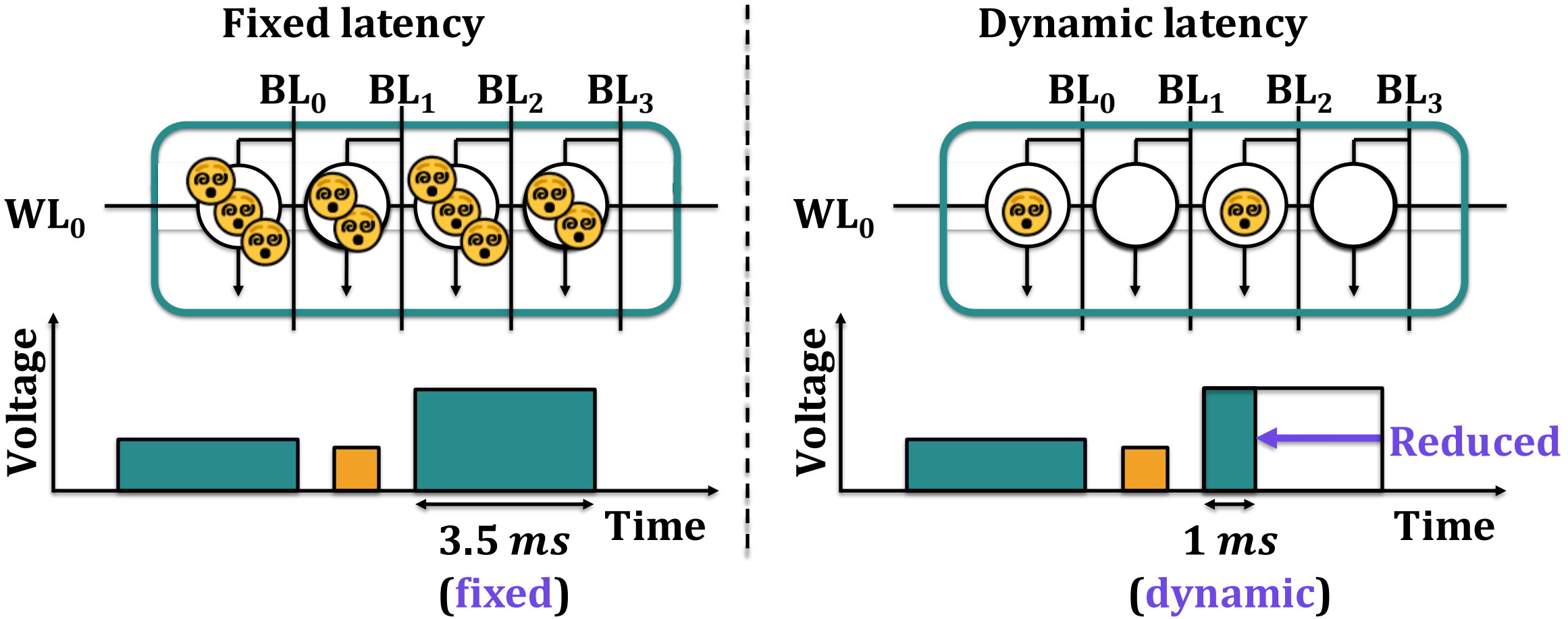
Adaptive Erase Operation (AERO)

- Fixed latency of erase operation causes over-erase



Adaptive Erase Operation (AERO)

- Dynamic latency can minimize damages and delays



Adaptive Erase Operation (AERO)

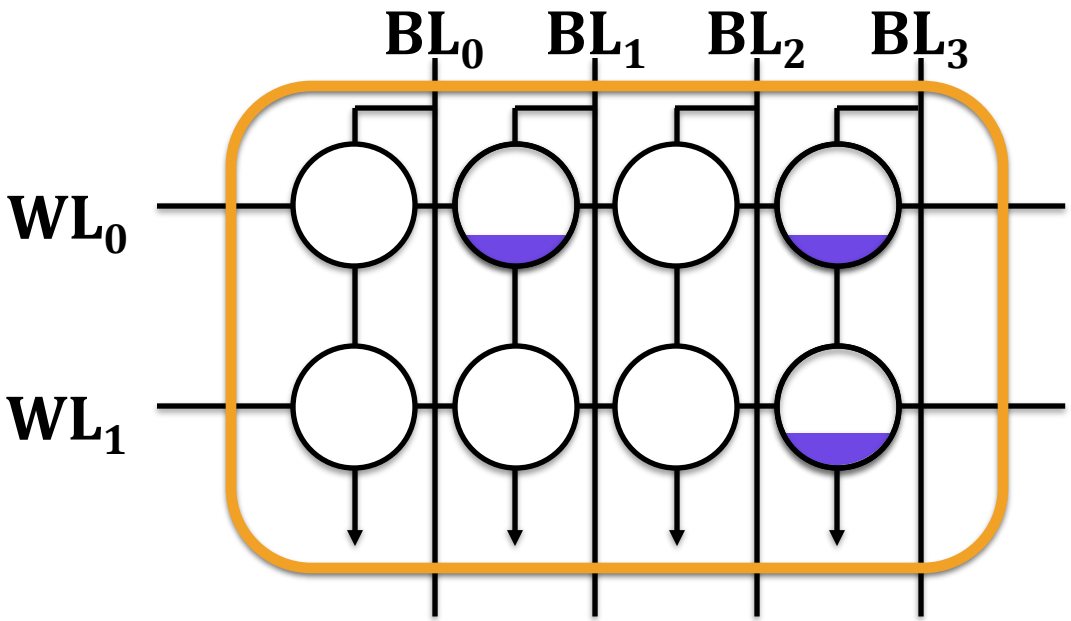
- Dynamic latency can minimize damages and delays

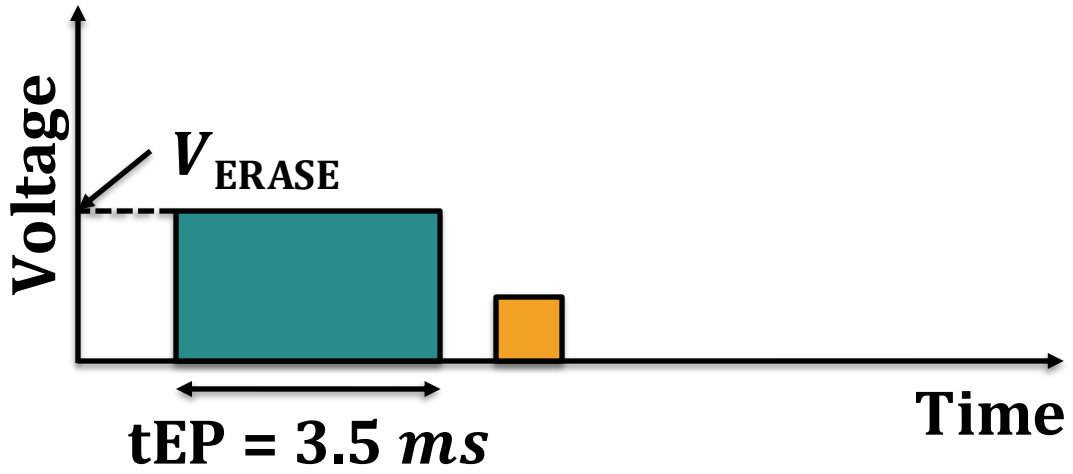
Challenge: Erase latency significantly varies across flash blocks even at the same P/E cycle

→ How to accurately predict erase latency?

Fail-Bit-Count-Based ER. Latency Prediction

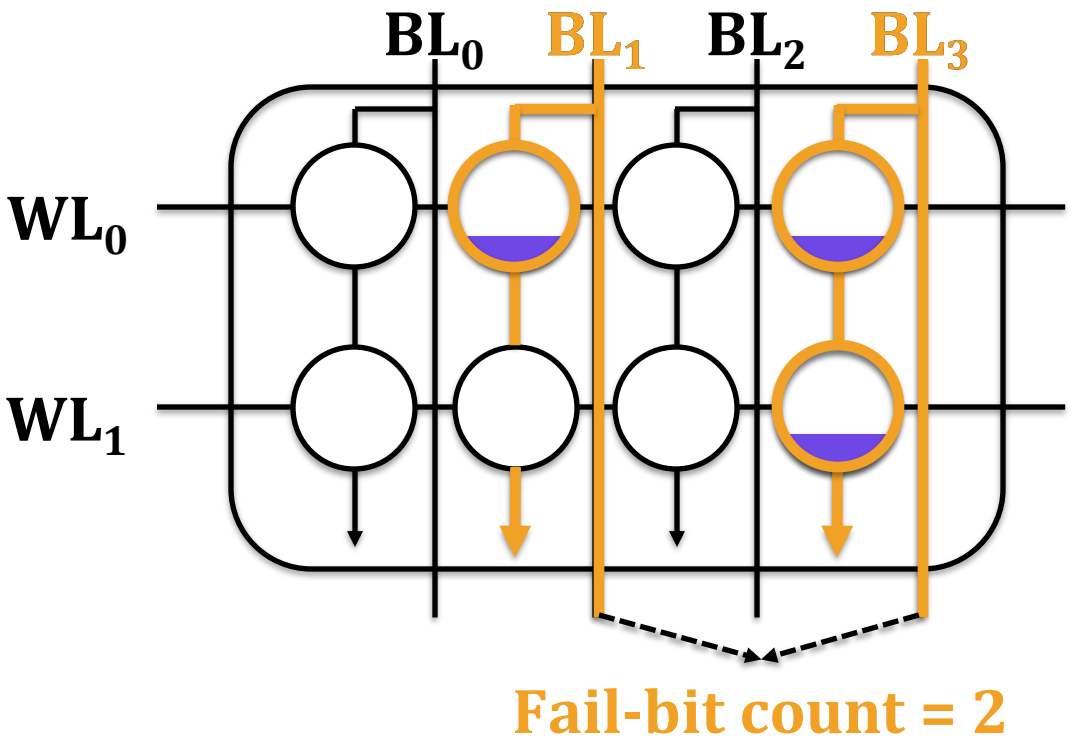
- FELP predicts latency based on the fail-bit count

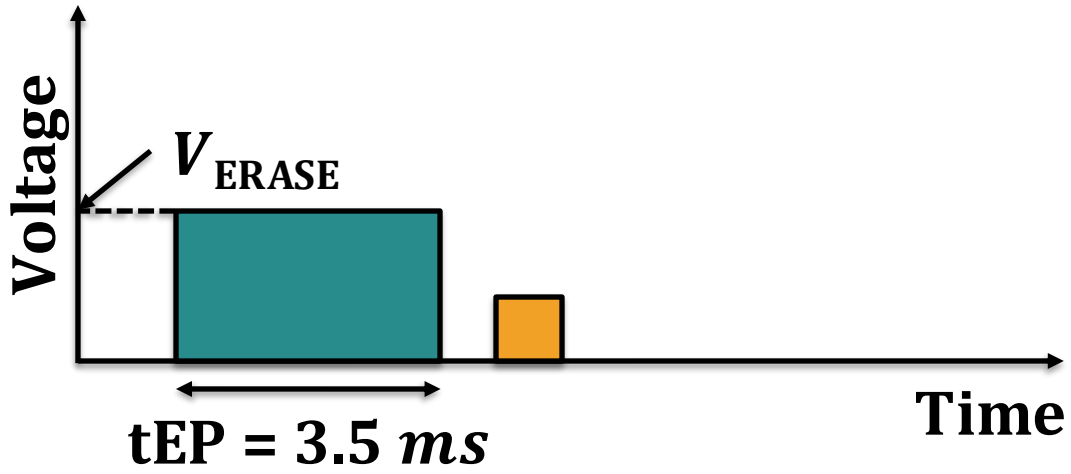




Fail-Bit-Count-Based ER. Latency Prediction

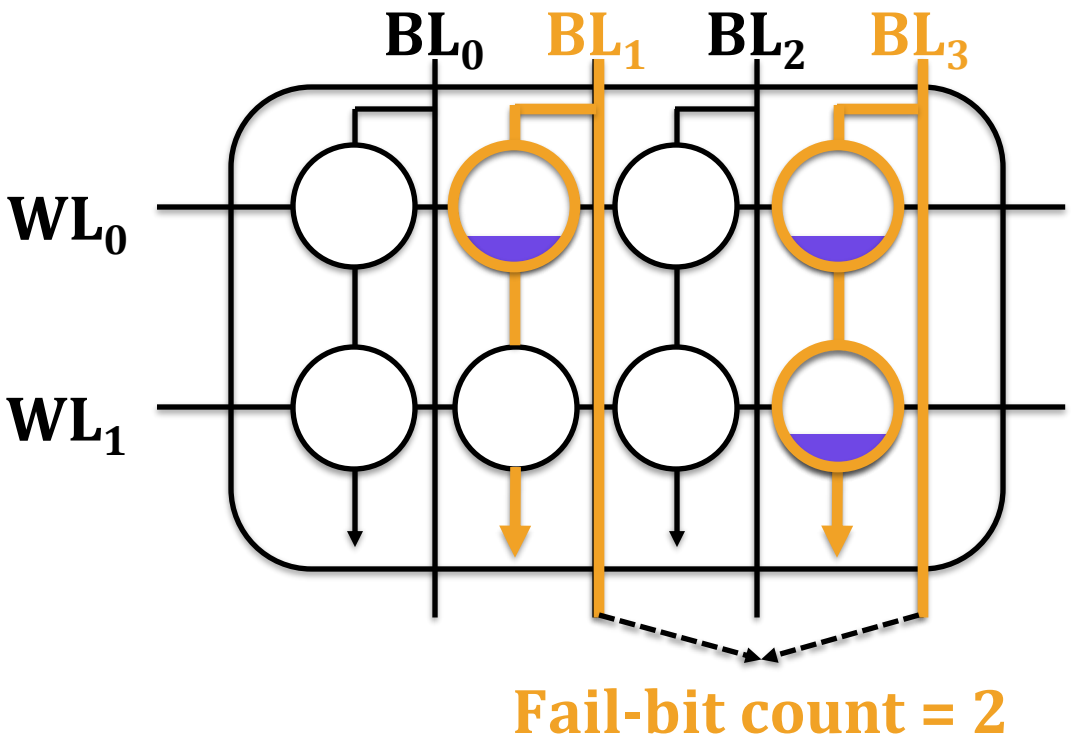
- FELP predicts latency based on the fail-bit count
  - Verify read counts the number of un-erased bitlines (i.e., fail bits)

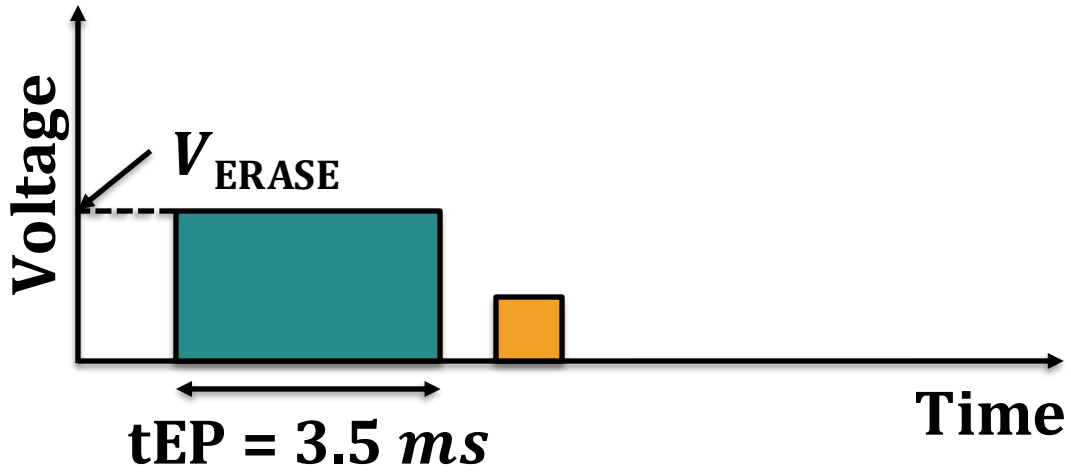




Fail-Bit-Count-Based ER. Latency Prediction

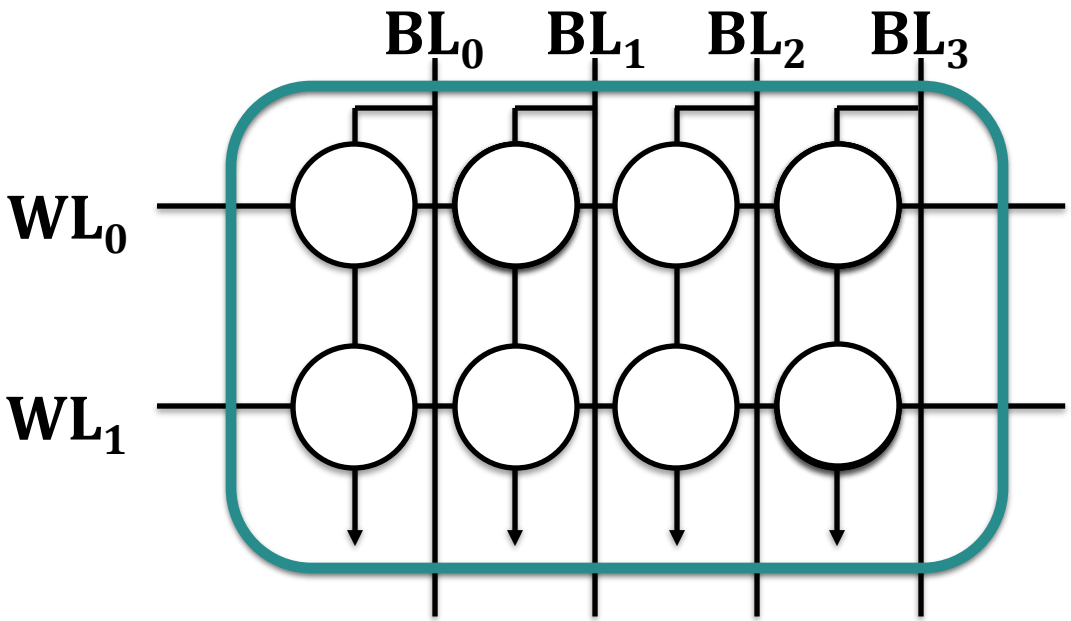
- FELP predicts latency based on the fail-bit count
  - Verify read counts the number of un-erased bitlines (i.e., fail bits)
  - Intuition: The lower the fail-bit count, the lower the latency

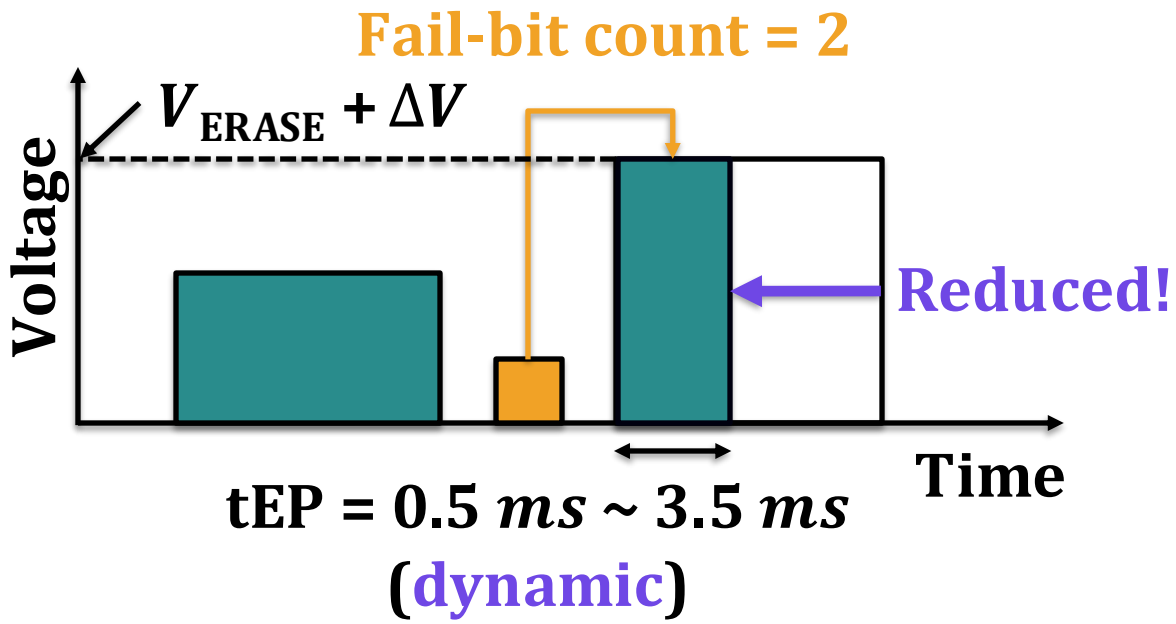




Fail-Bit-Count-Based ER. Latency Prediction

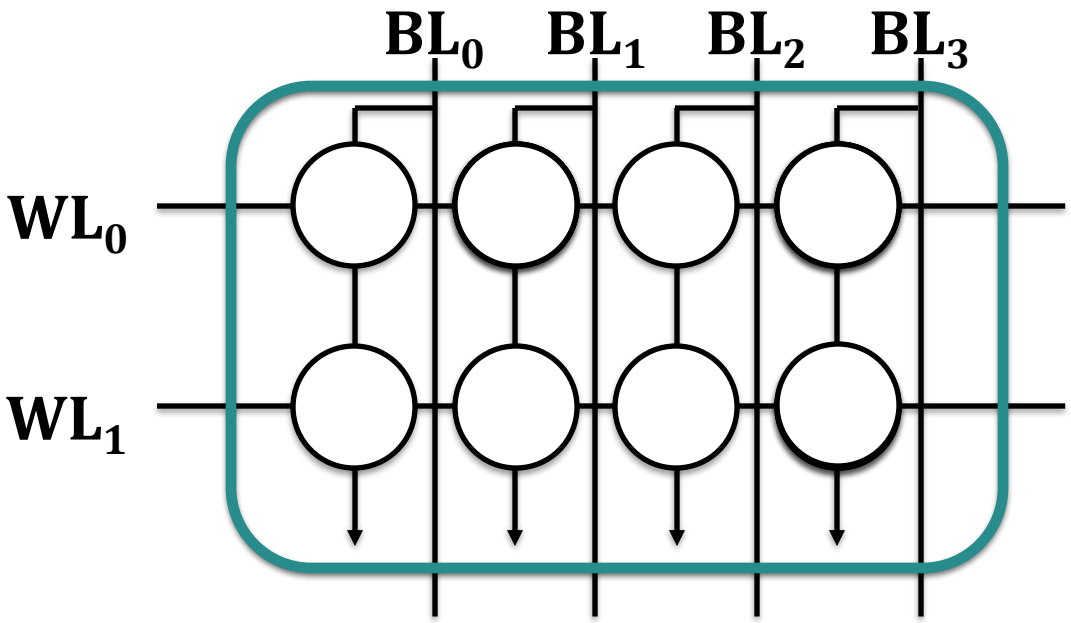
- FELP predicts latency based on the fail-bit count
  - Verify read counts the number of un-erased bitlines (i.e., fail bits)
  - Intuition: The lower the fail-bit count, the lower the latency

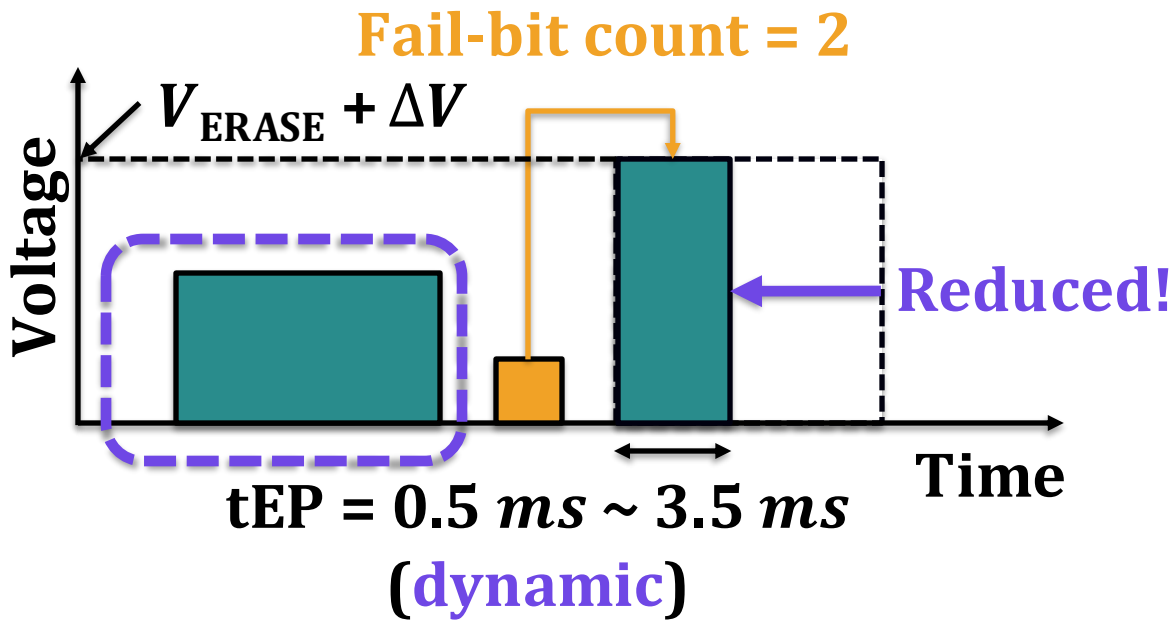




Fail-Bit-Count-Based ER. Latency Prediction

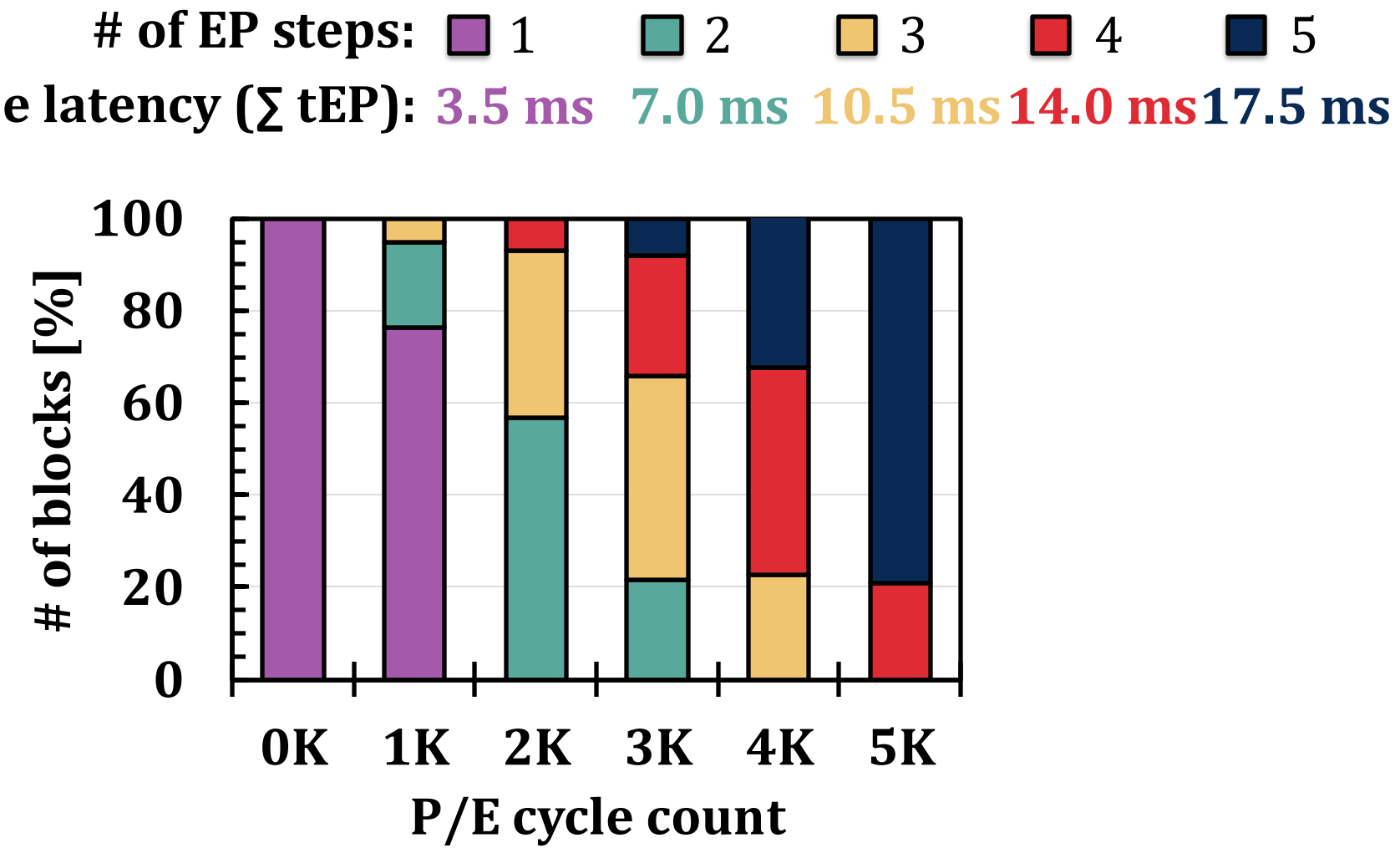
- FELP predicts latency based on the fail-bit count
  - Verify read counts the number of un-erased bitlines (i.e., fail bits)
  - Intuition: The lower the fail-bit count, the lower the latency





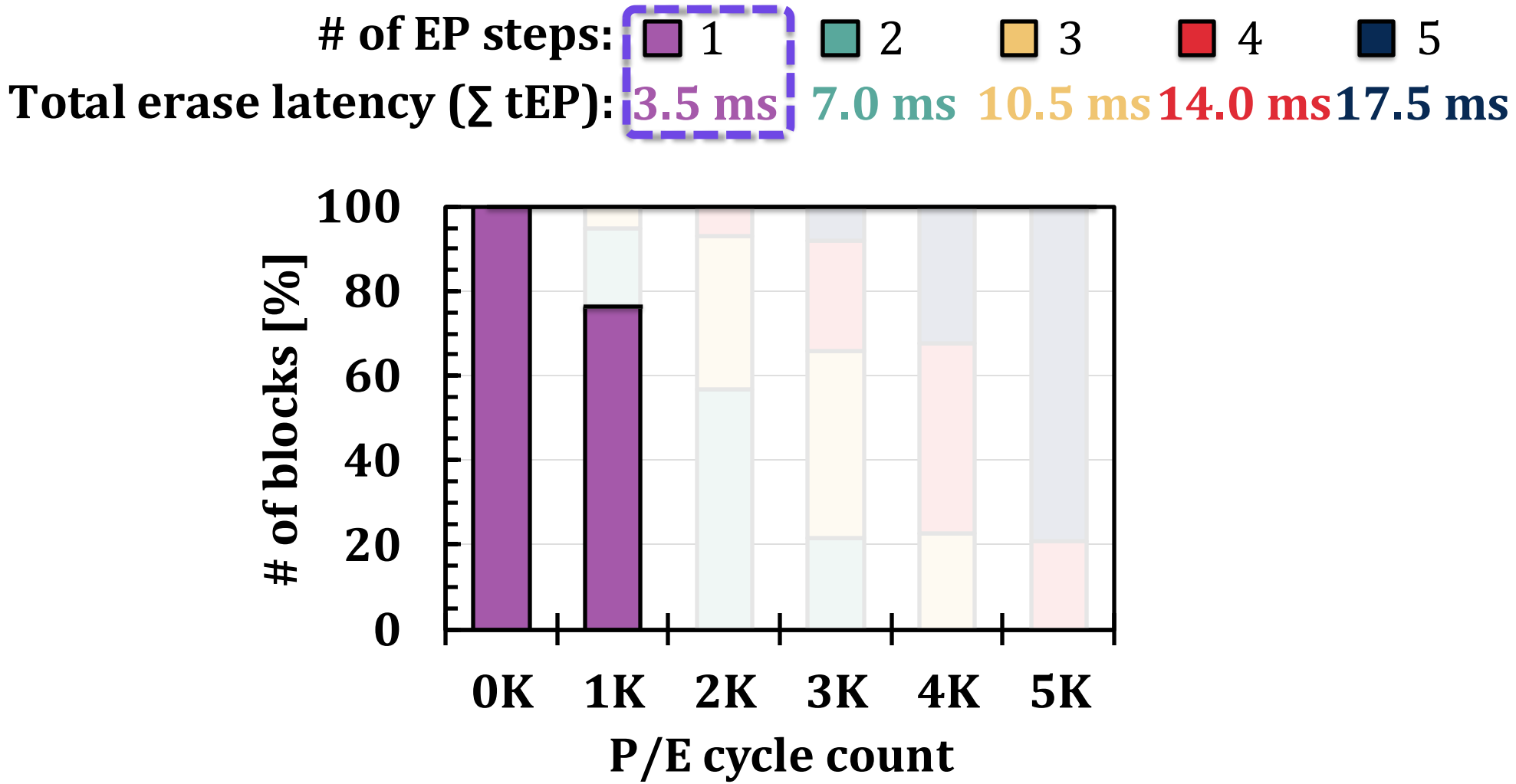
Observations for Optimization

- Many blocks only require single erase pulse in early lifetime



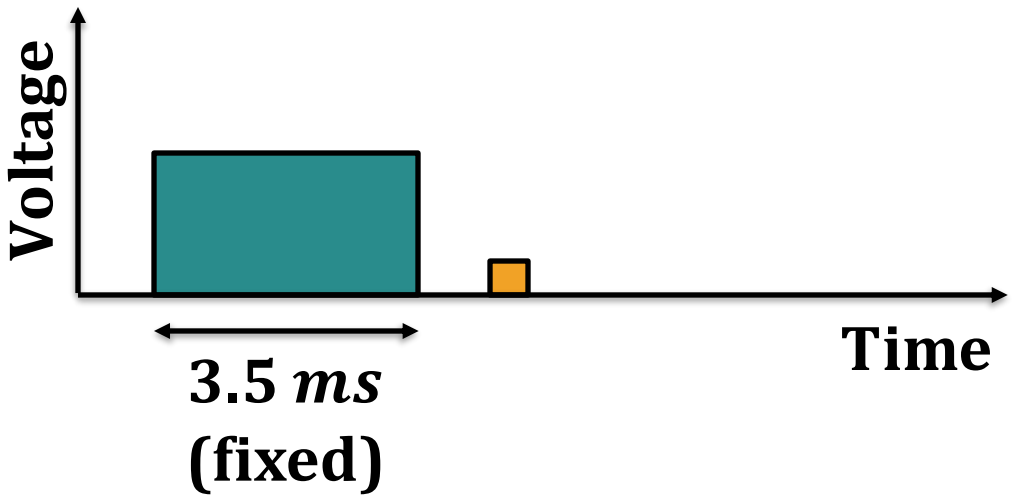
Observations for Optimization

- Many blocks only require single erase pulse in early lifetime



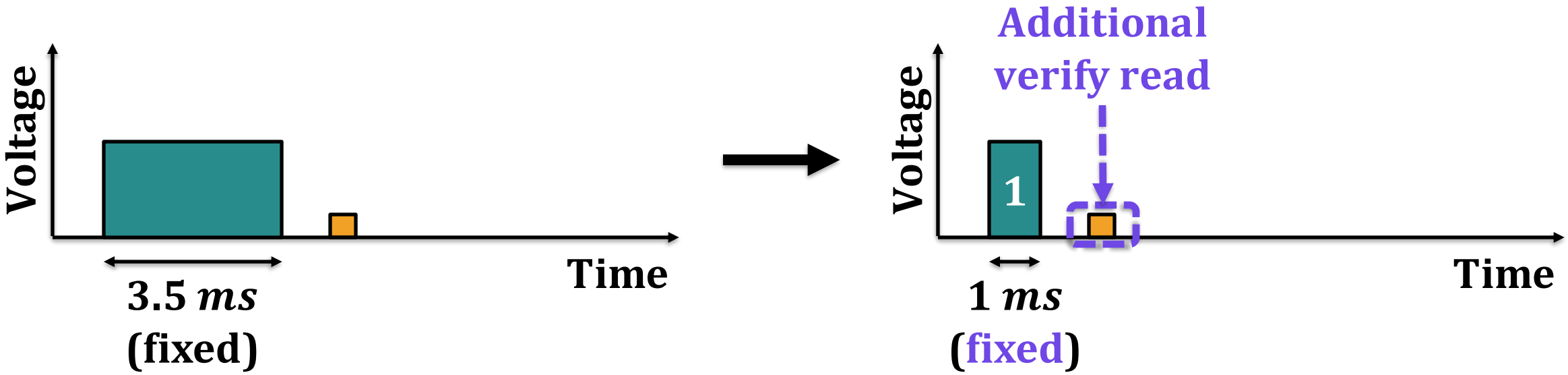
Optimization: Shallow Erasure

- Split the first erase pulse into two parts
  1. Shallow erasure
  2. Remainder erasure



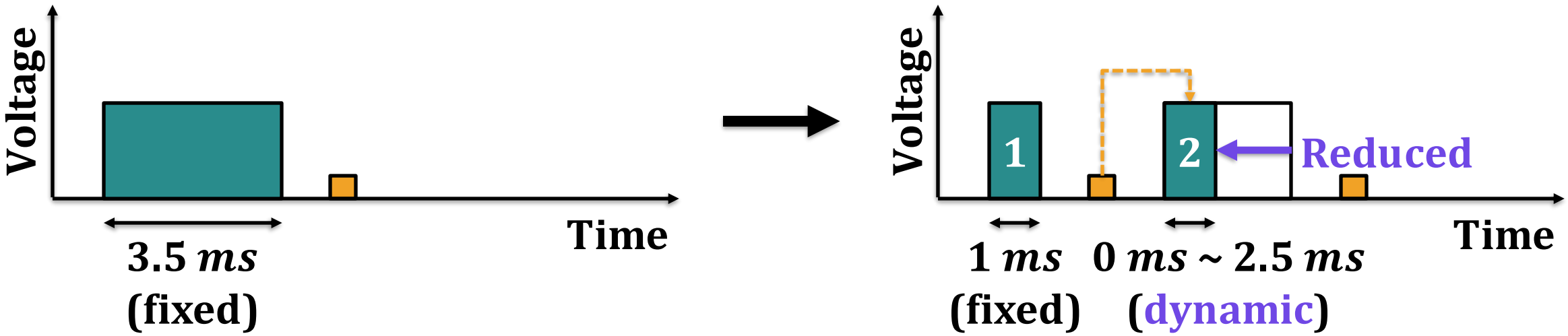
Optimization: Shallow Erasure

- Split the first erase pulse into two parts
  1. Shallow erasure: Short erasure with fixed latency (1 ms)
  2. Remainder erasure



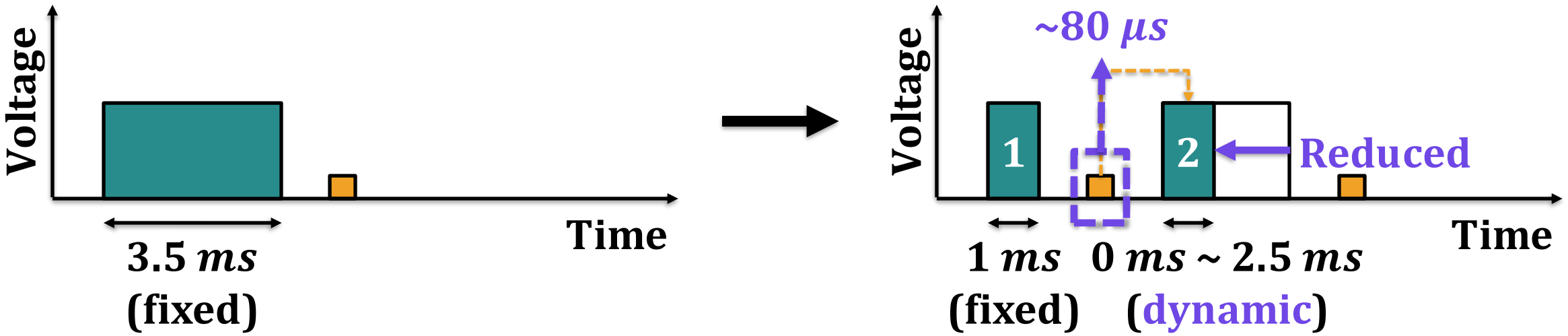
Optimization: Shallow Erasure

- Split the first erase pulse into two parts
  1. Shallow erasure: Short erasure with fixed latency (1 ms)
  2. Remainder erasure: Erasure based on FELP (0 ms ~ 2.5 ms)



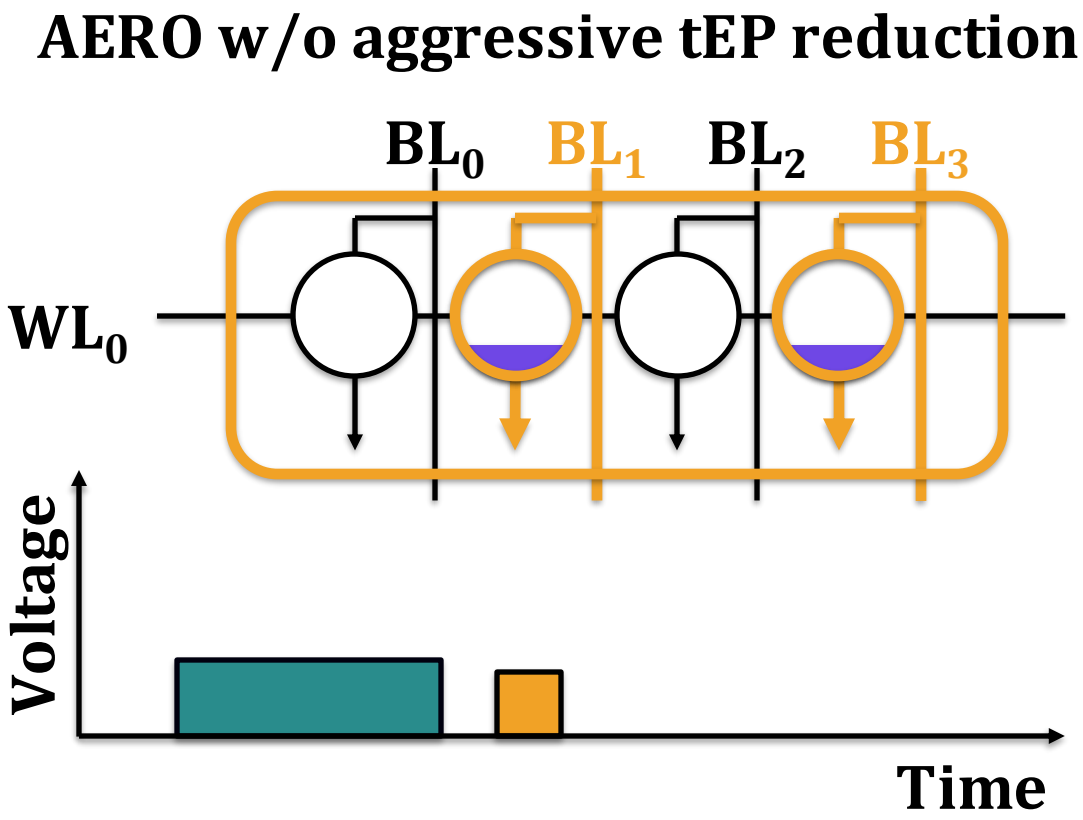
Optimization: Shallow Erasure

- Split the first erase pulse into two parts
  1. Shallow erasure: Short erasure with fixed latency (1 ms)
  2. Remainder erasure: Erasure based on FELP (0 ms ~ 2.5 ms)



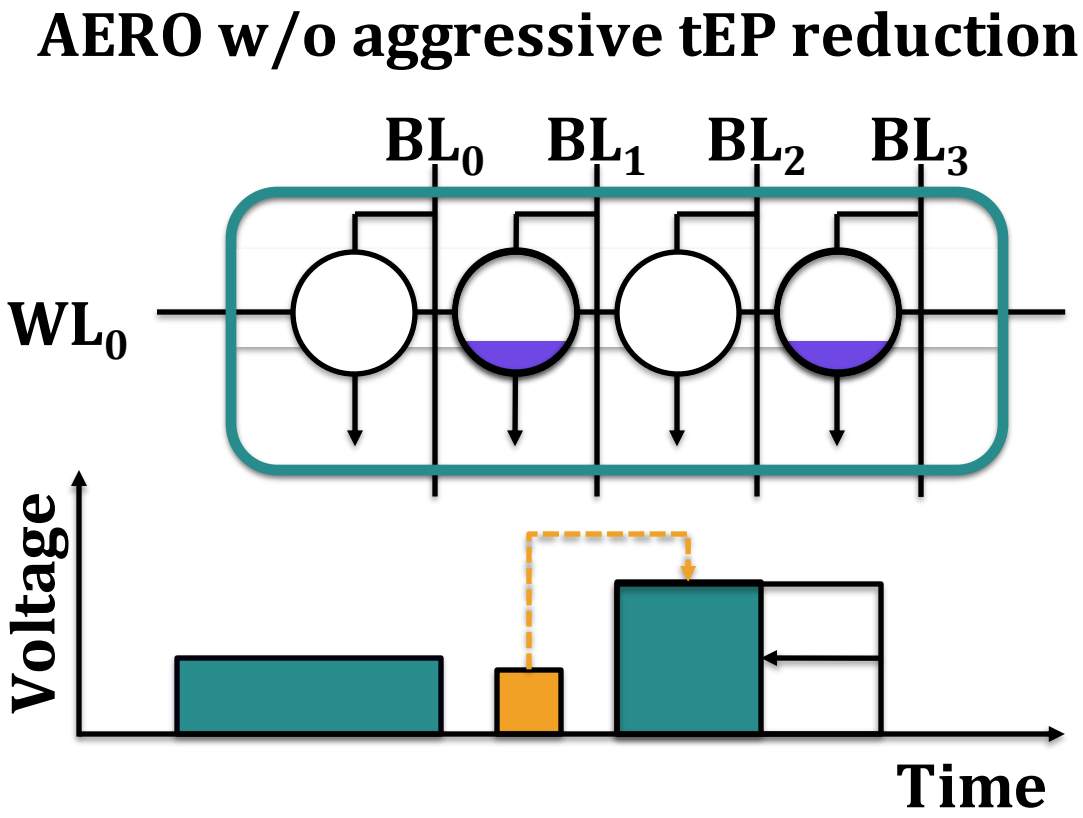
Optimization: Aggressive tEP Reduction

- Allows incomplete erasure to further reduce erase latency



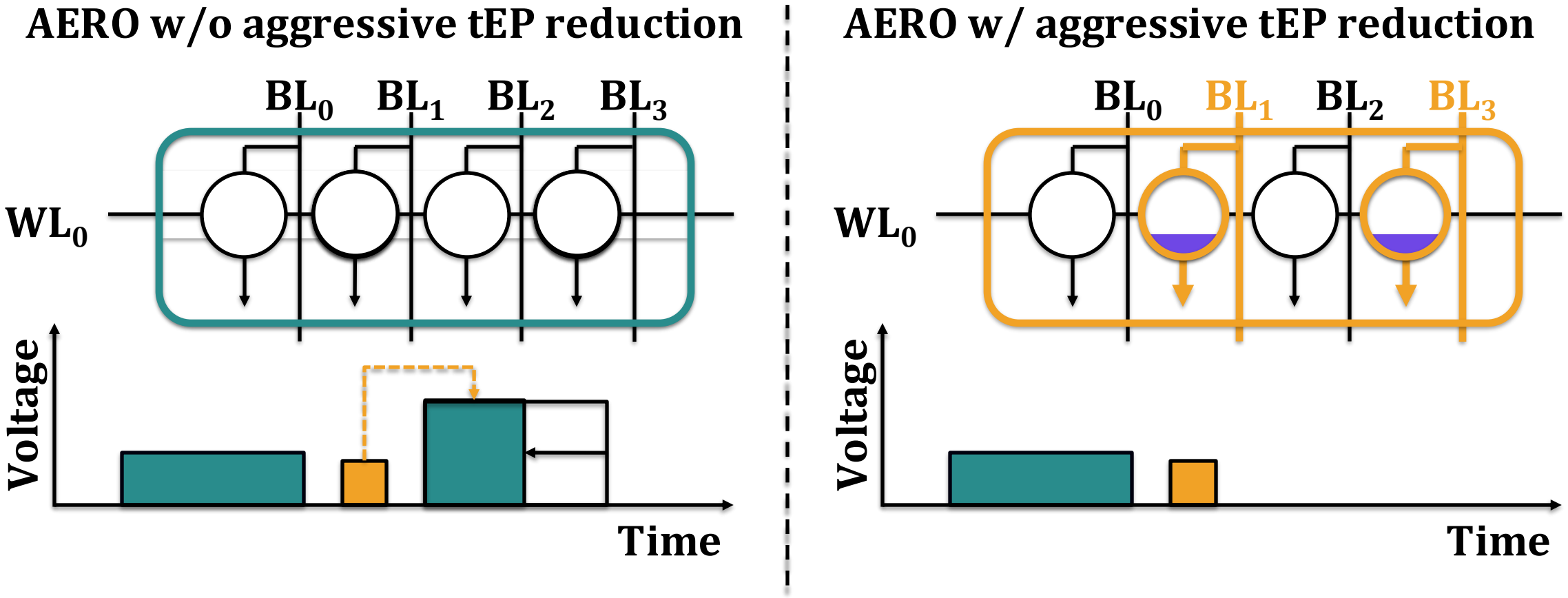
Optimization: Aggressive tEP Reduction

- Allows incomplete erasure to further reduce erase latency



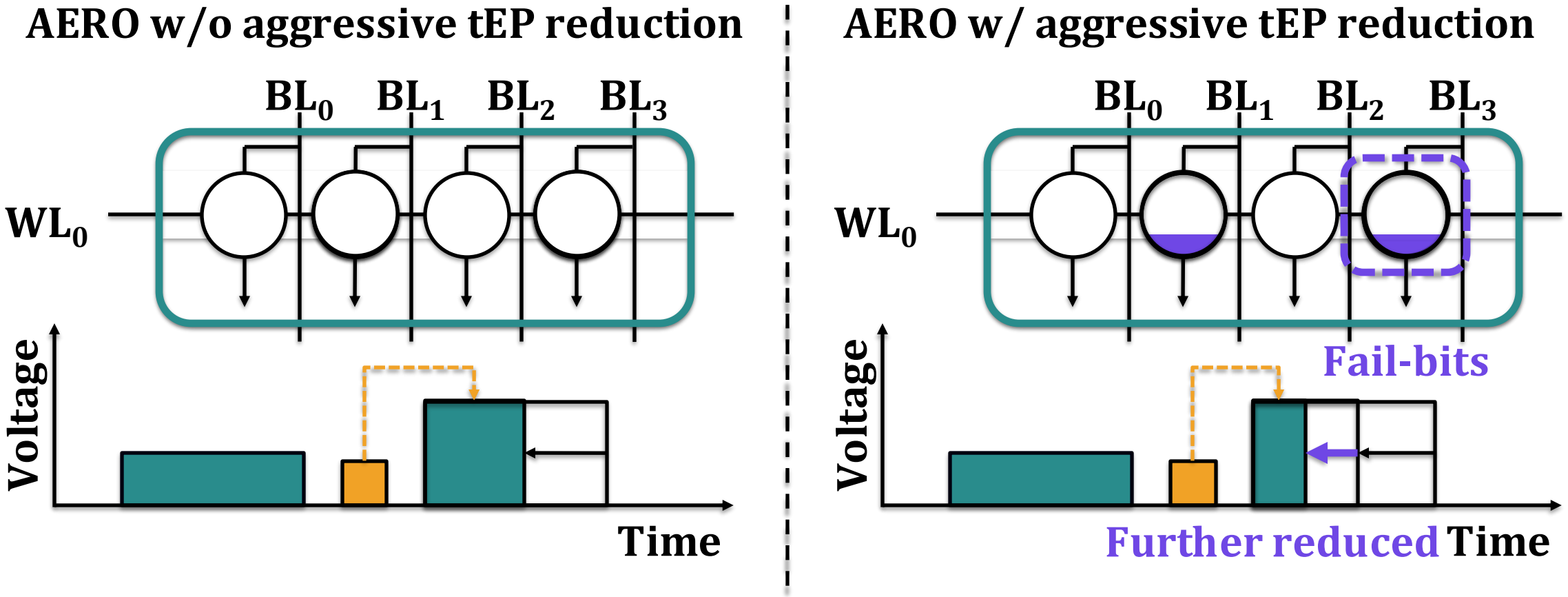
Optimization: Aggressive tEP Reduction

- Allows incomplete erasure to further reduce erase latency



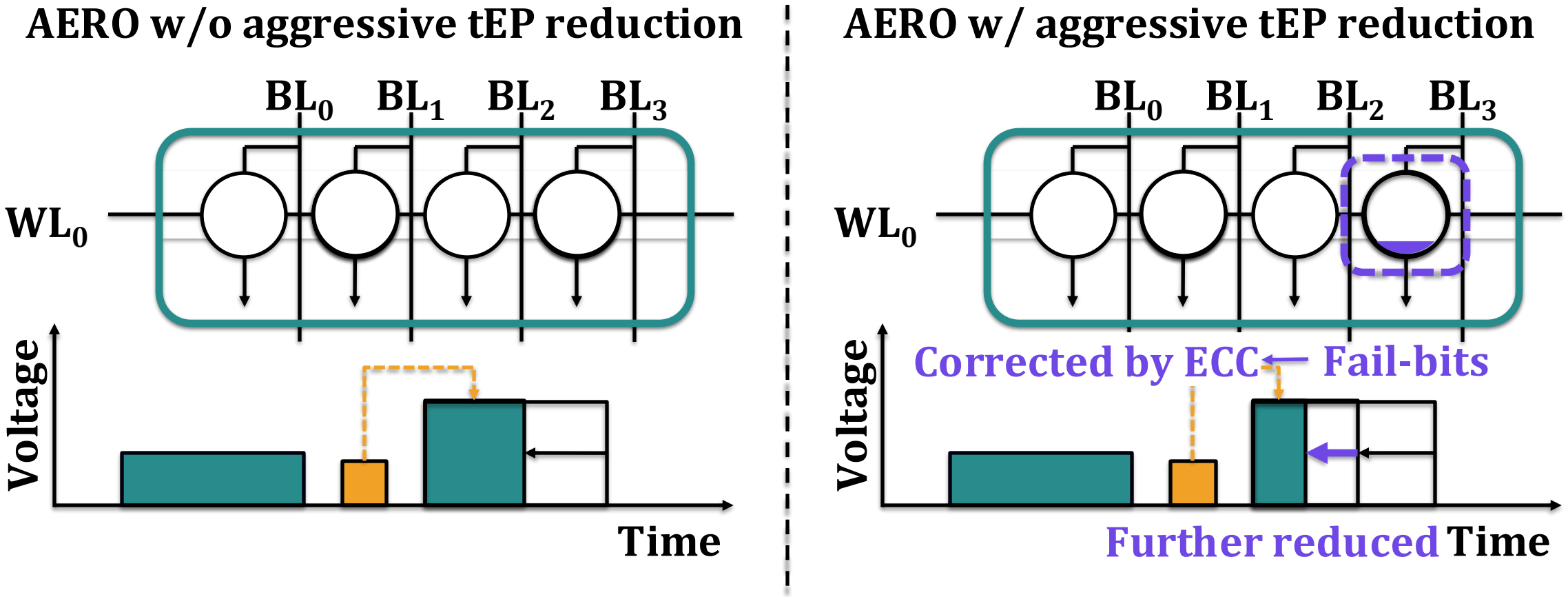
Optimization: Aggressive tEP Reduction

- Allows incomplete erasure to further reduce erase latency
  - Inevitably causes more errors



Optimization: Aggressive tEP Reduction

- Allows incomplete erasure to further reduce erase latency
  - Inevitably causes more errors: Still can be corrected by ECC



Feasibility Validation in the Paper

- Using 160 real 3D TLC NAND flash chips, we study
  - Impact of erase pulse latency on the fail-bit count
  - Feasibility and parameter exploration of shallow erasure
  - Reliability characteristics of incompletely erased block
  - Applicability of AERO for other types of chips
    - 2D TLC and 3D MLC

Please refer to the paper!

Implementation of AERO

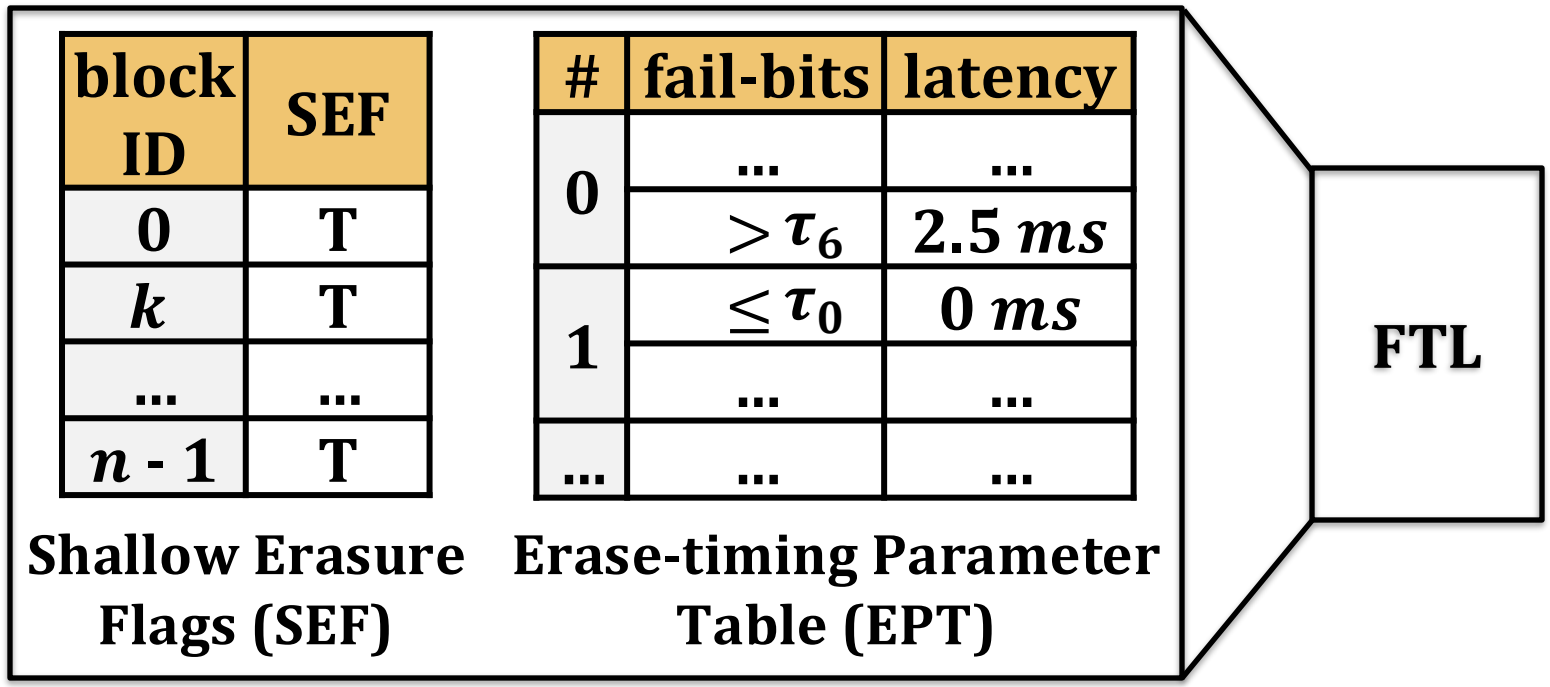
- Extend flash translation layer (FTL) w/ two data structures

Implementation of AERO

- Extend flash translation layer (FTL) w/ two data structures
  - Shallow erasure flags (SEF): Check if shallow erasure is required

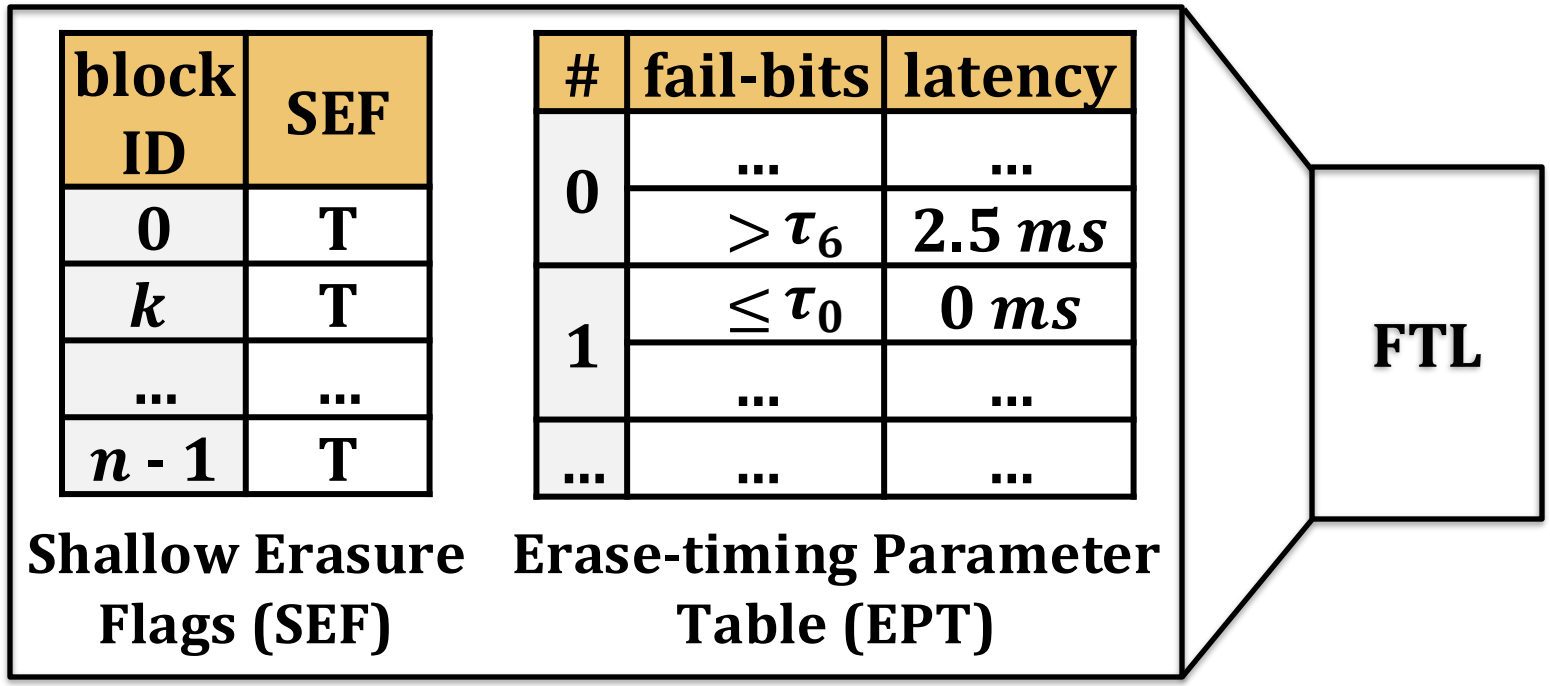
Implementation of AERO

- Extend flash translation layer (FTL) w/ two data structures
  - Shallow erasure flags (SEF): Check if shallow erasure is required
  - Erase-timing parameter table (EPT): Contains erase pulse latency



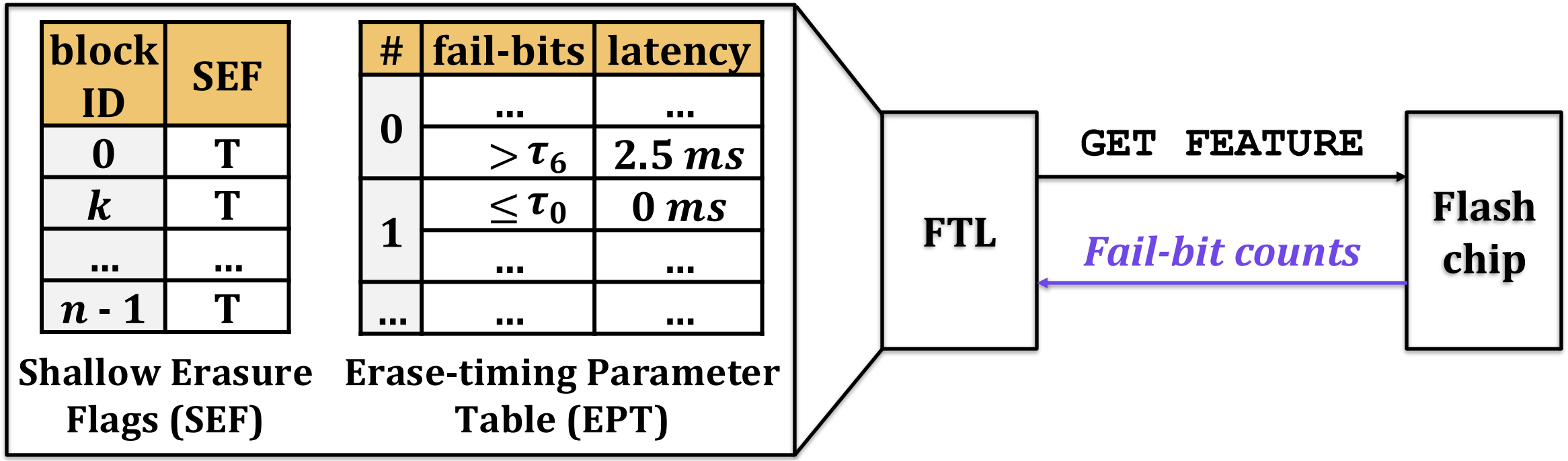
Implementation of AERO

- Uses two existing commands to adjust erase latency



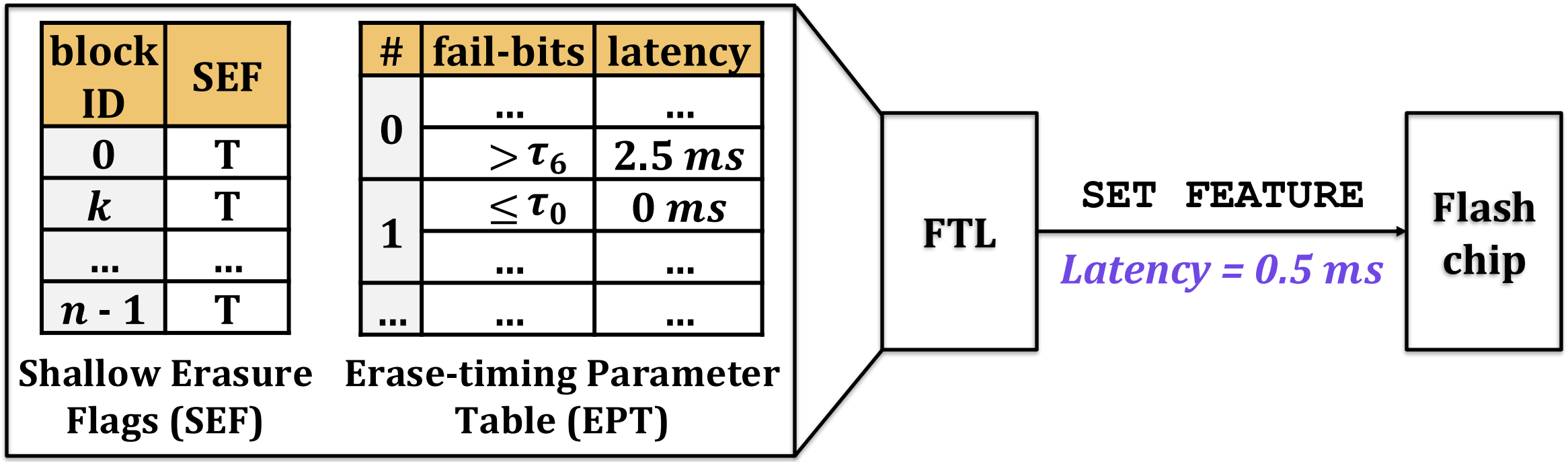
Implementation of AERO

- Uses two existing commands to adjust erase latency
  - **`GET FEATURE`**: Gets fail-bit counts from a flash chip



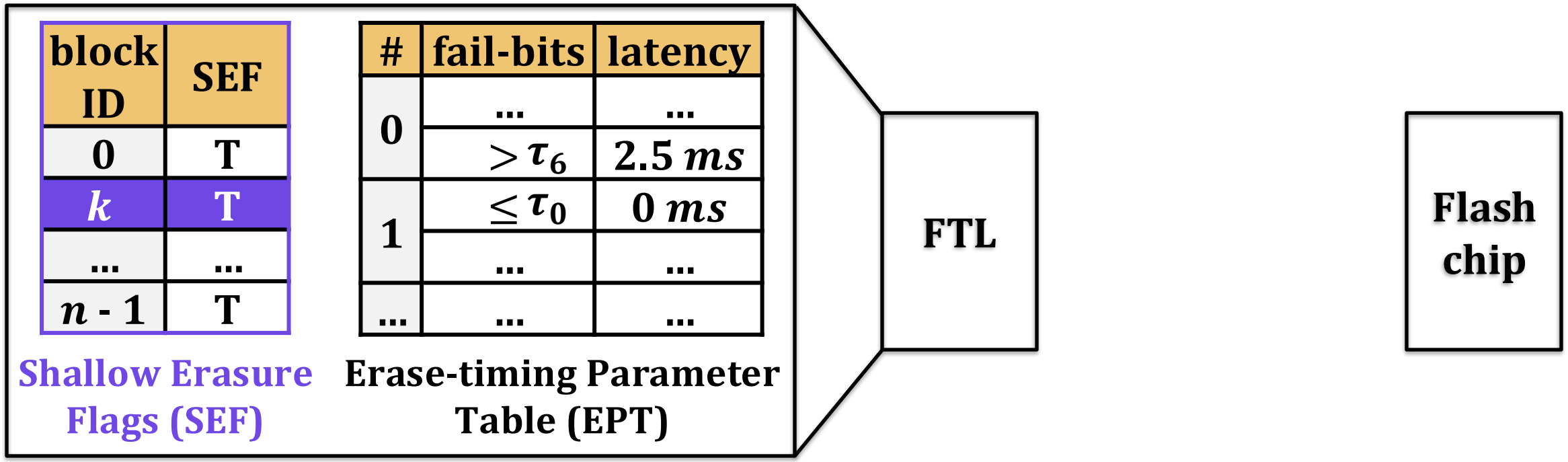
Implementation of AERO

- Uses two existing commands to adjust erase latency
  - **GET FEATURE**: Gets fail-bit counts from a flash chip
  - **SET FEATURE**: Sets latency of erase pulse



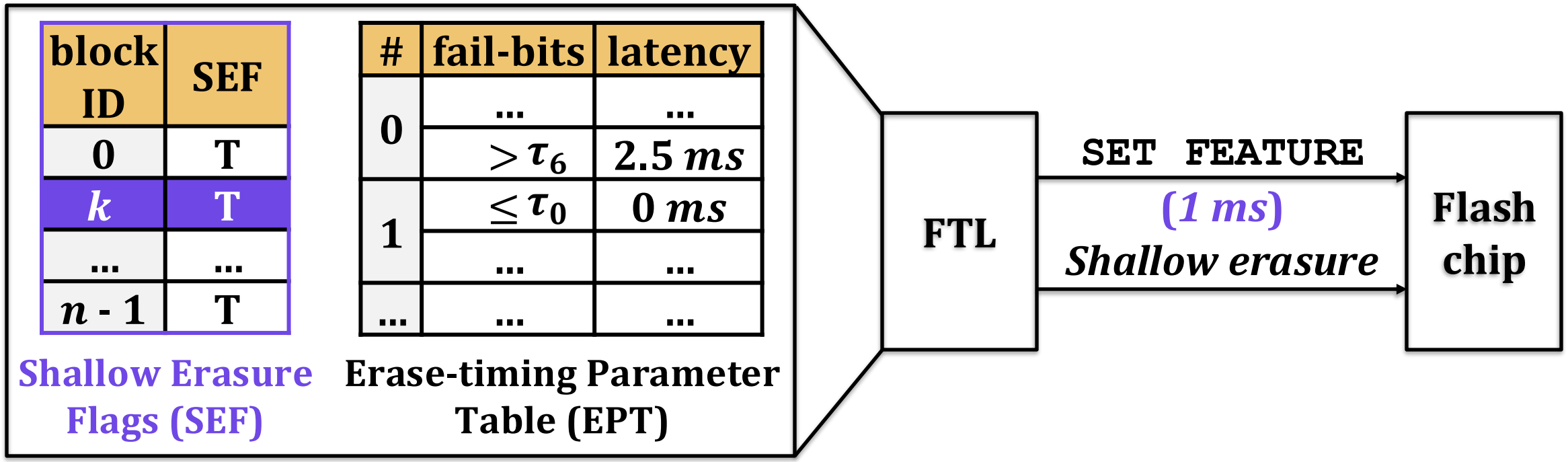
Implementation of AERO

- Step 1: Check the necessity of shallow erasure w/ SEF
  - True: Performs shallow erasure for short latency (1 ms)
  - False: Performs the first erase pulse for fixed latency (3.5 ms)



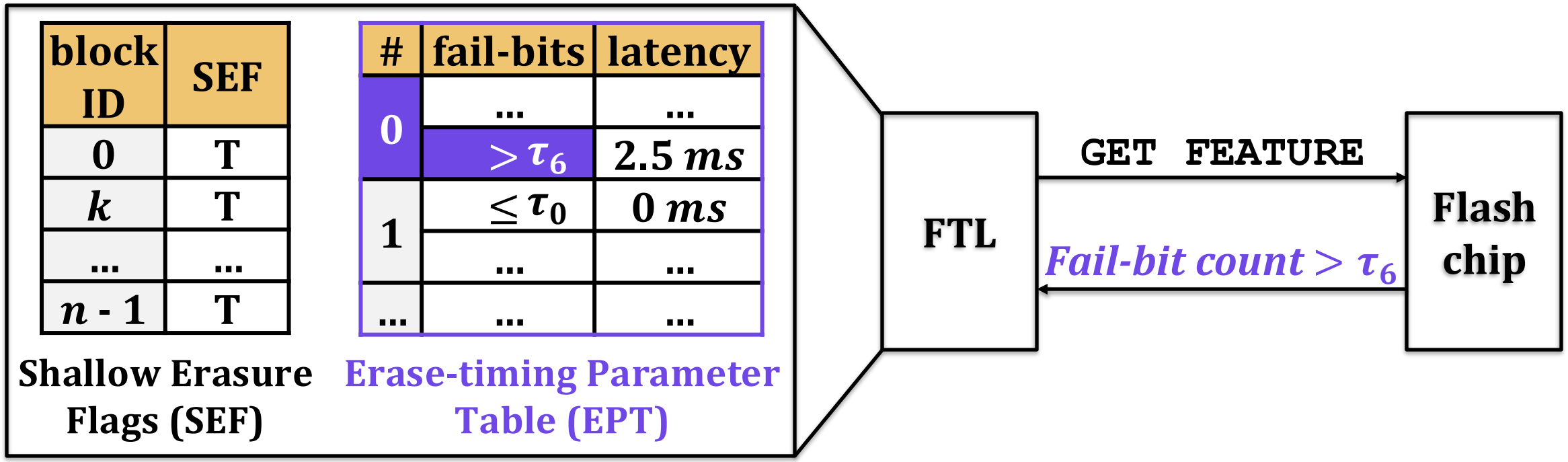
Implementation of AERO

- Step 1: Check the necessity of shallow erasure w/ SEF
  - True: Performs shallow erasure for short latency (1 ms)
  - False: Performs the first erase pulse for fixed latency (3.5 ms)



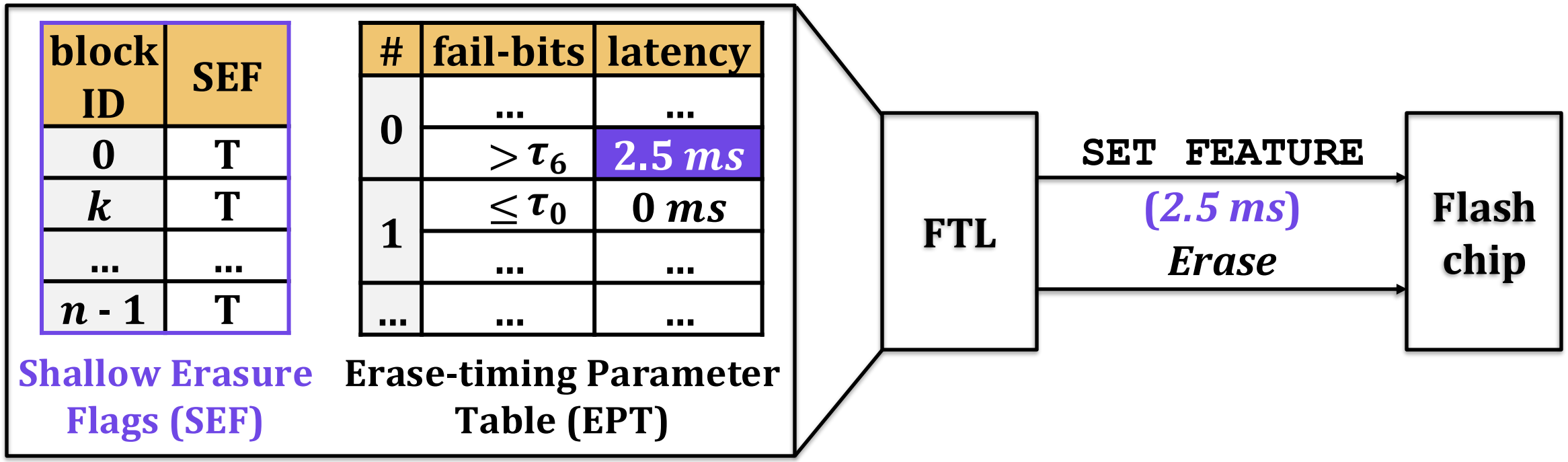
Implementation of AERO

- Step 2: Queries the minimum erase-pulse latency by using
  - The current number of erase pulse and fail-bit count



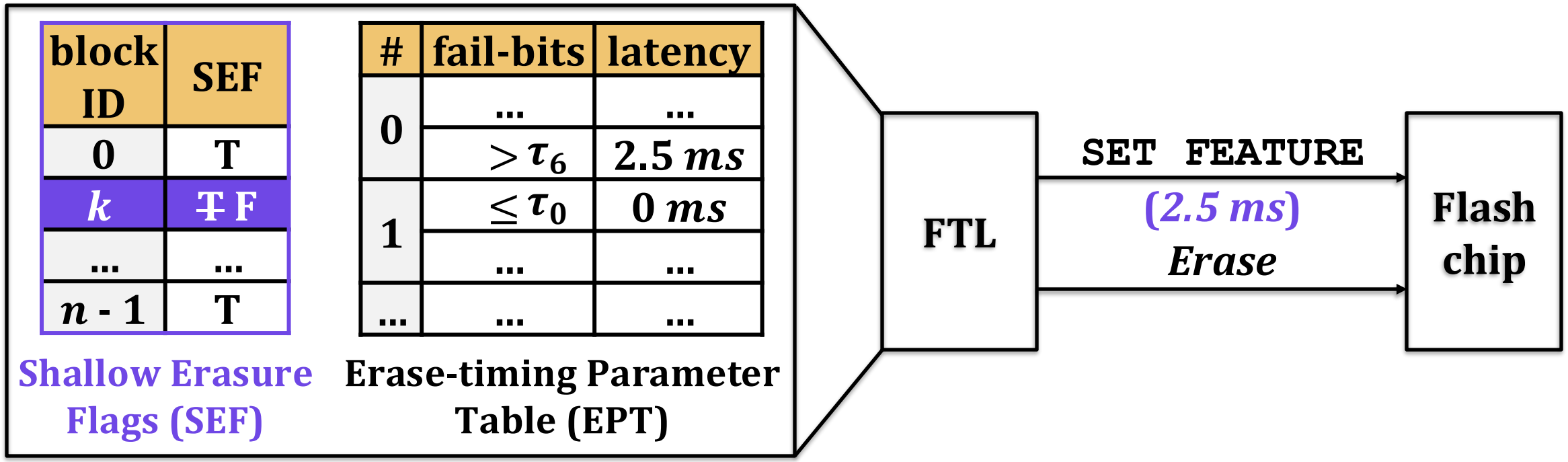
Implementation of AERO

- Step 3: Sets the latency to perform the next erase pulse



Implementation of AERO

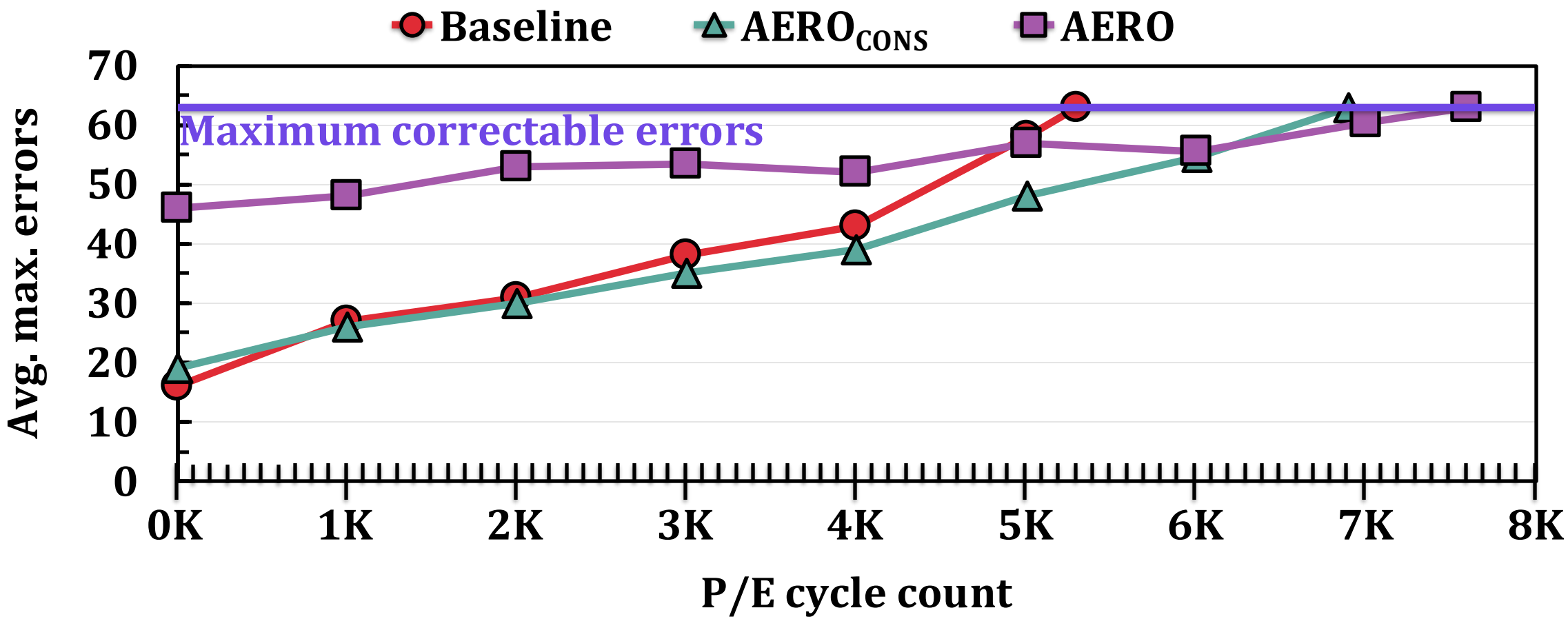
- Step 4: Modifies SEF to false if
  - shallow erasure (1 ms) + remainder erasure (2.5 ms) = 3.5 ms



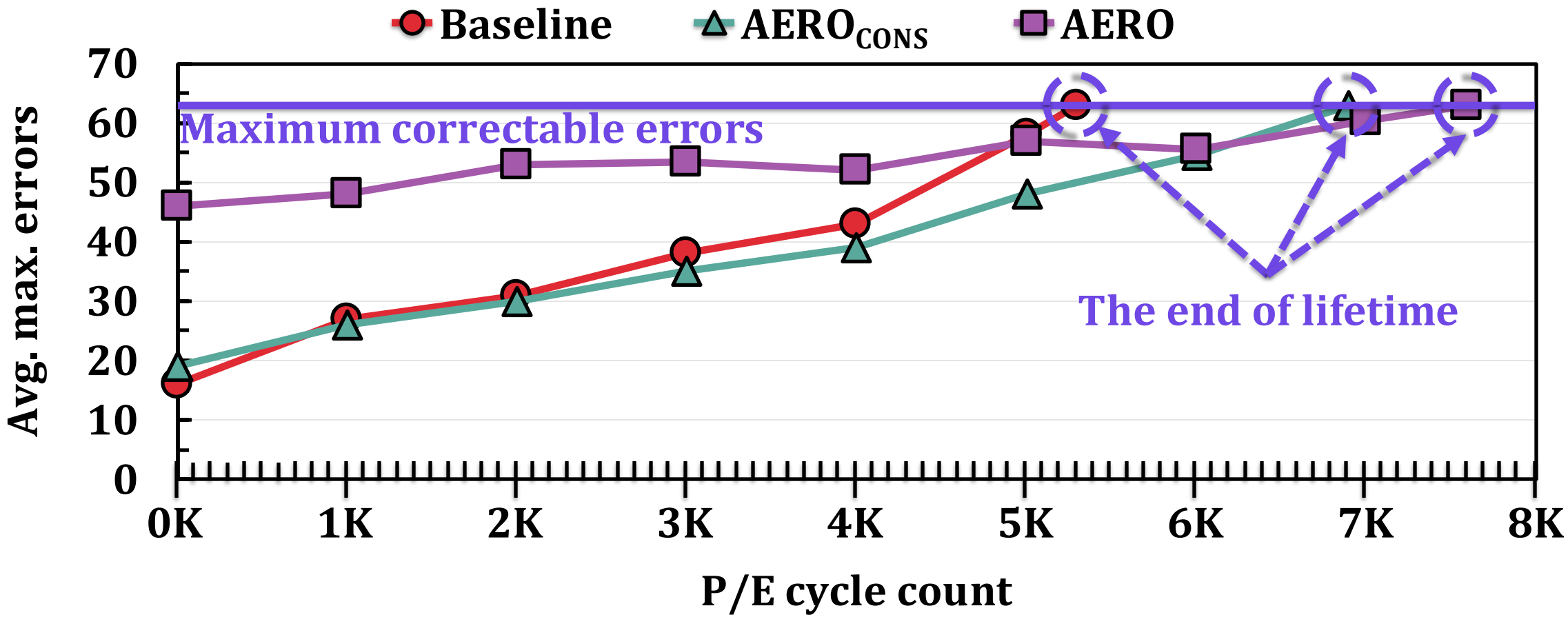
Evaluation Methodology

- SSD lifetime (Max. P/E-cycle count)
  - Real-device characterization of 160 3D TLC NAND flash chips
- I/O Performance (read tail latency)
  - Simulator: MQSim [FAST'19]
  - Workloads: 11 real-world I/O workloads
    - 5 from Alibaba traces
    - 6 from Micro Research Cambridge (MSRC) traces
- Compared erase schemes
  - Baseline: Conventional erase scheme w/ fixed erase-pulse latency (tEP)
  - $AERO_{CONS}$: Dynamic tEP w/o aggressive reduction using ECC margin
  - AERO: Dynamic tEP w/ aggressive reduction using ECC margin

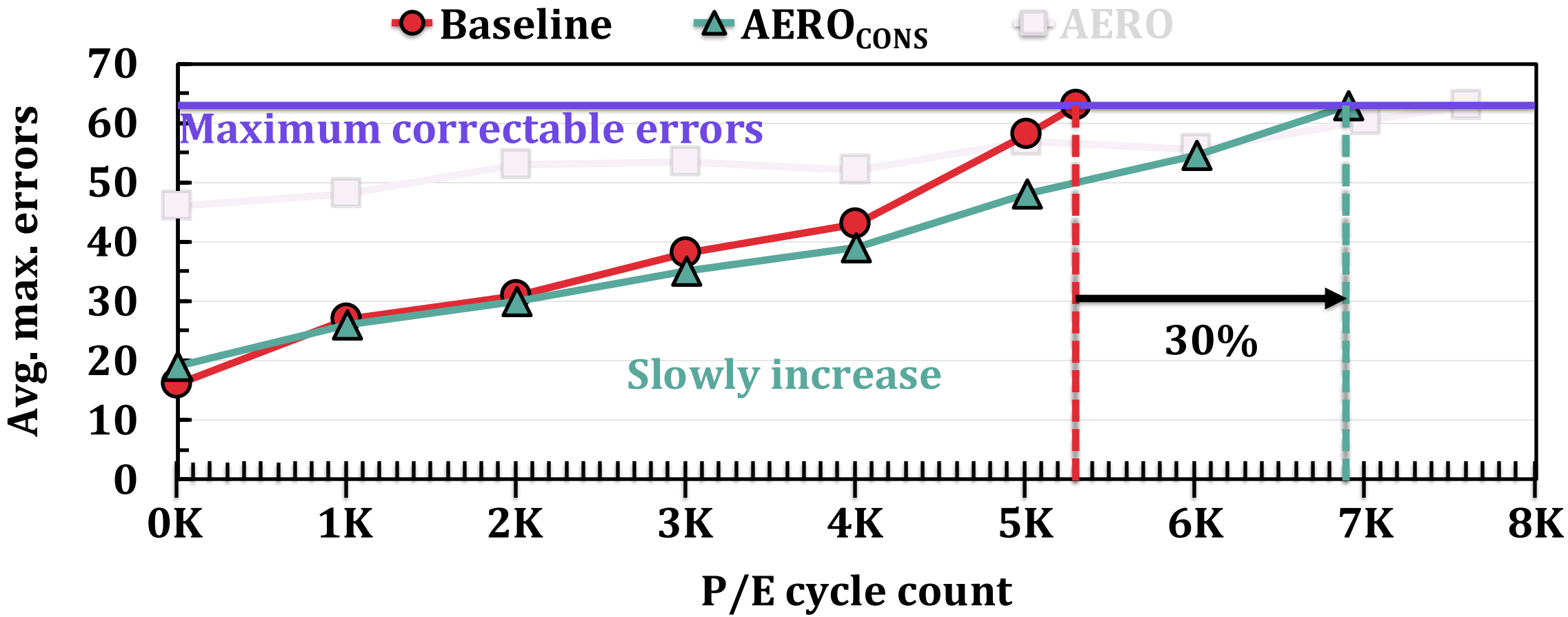
Result: SSD Lifetime



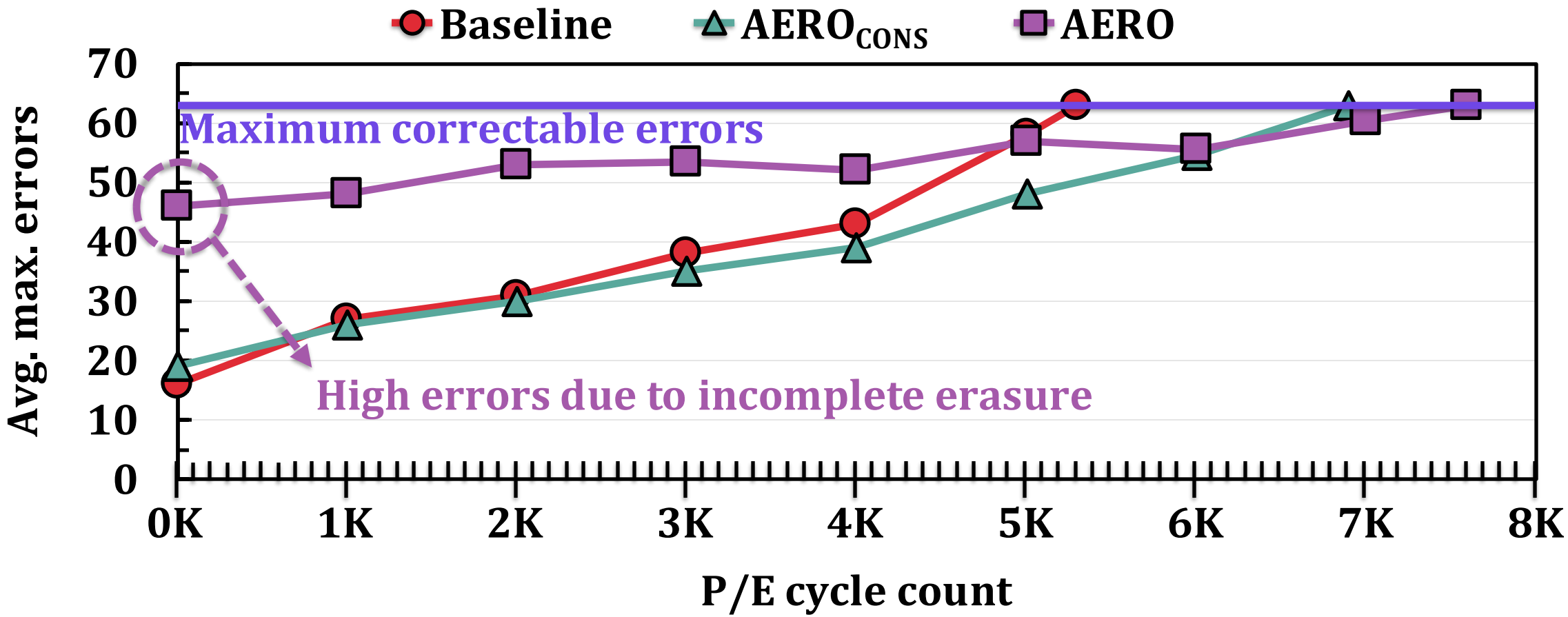
Result: SSD Lifetime



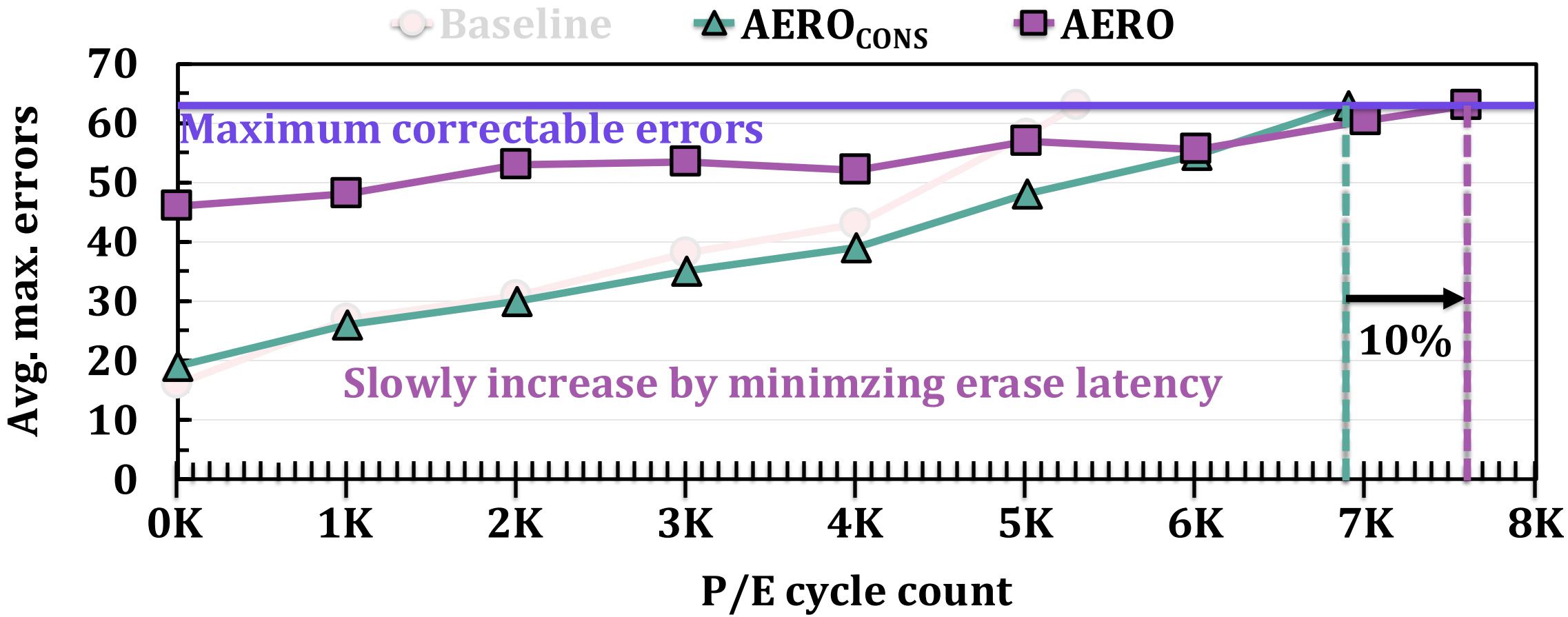
Result: SSD Lifetime



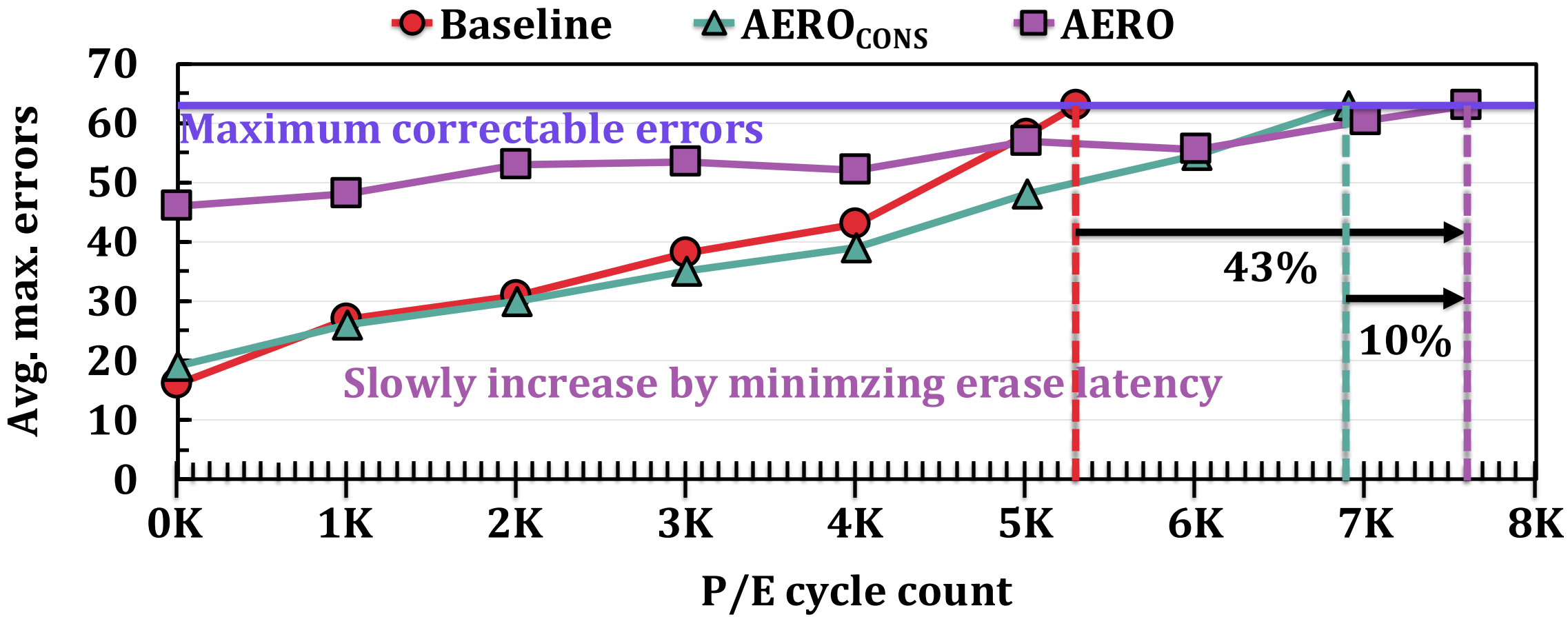
Result: SSD Lifetime



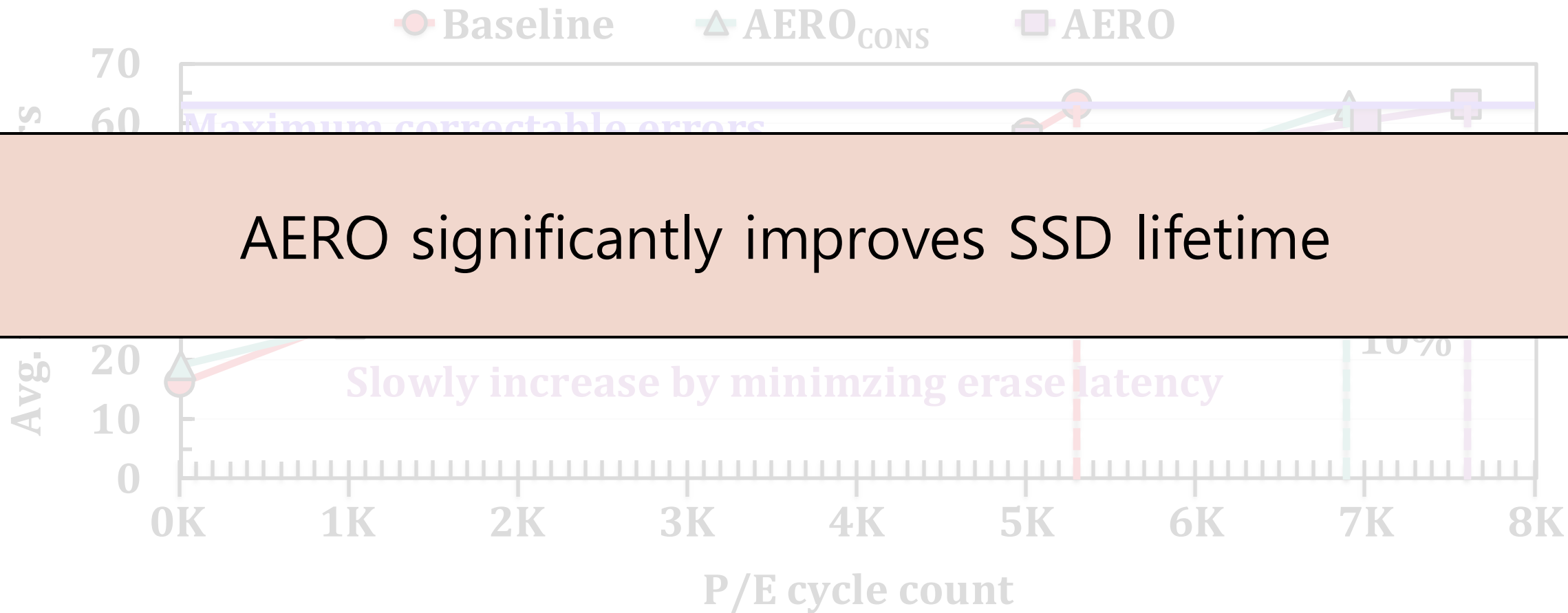
Result: SSD Lifetime



Result: SSD Lifetime

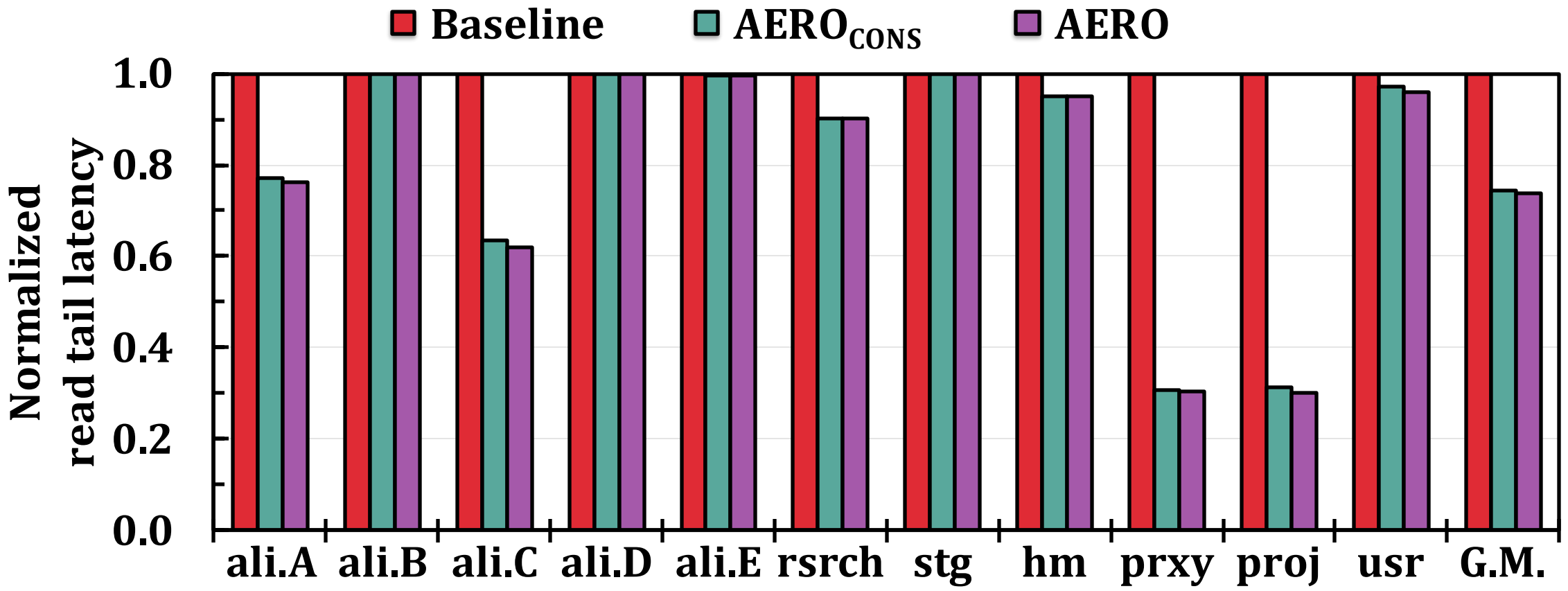


Result: SSD Lifetime

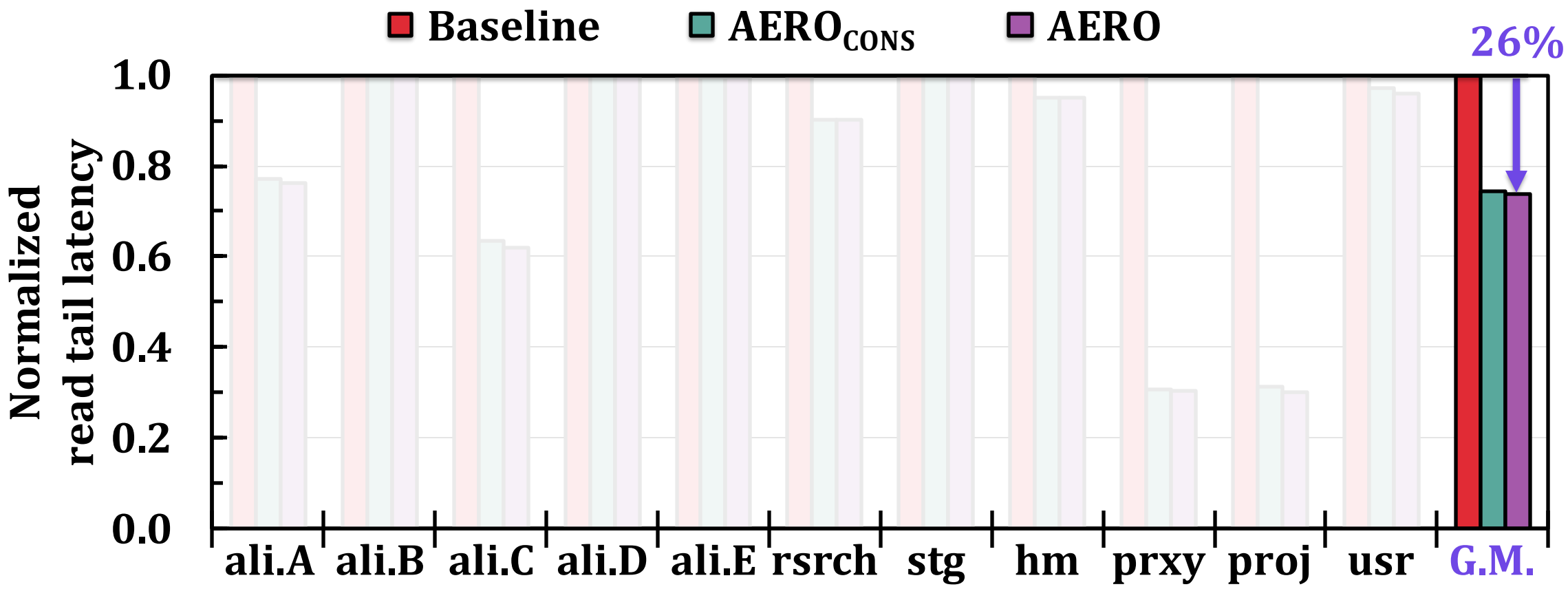


AERO significantly improves SSD lifetime

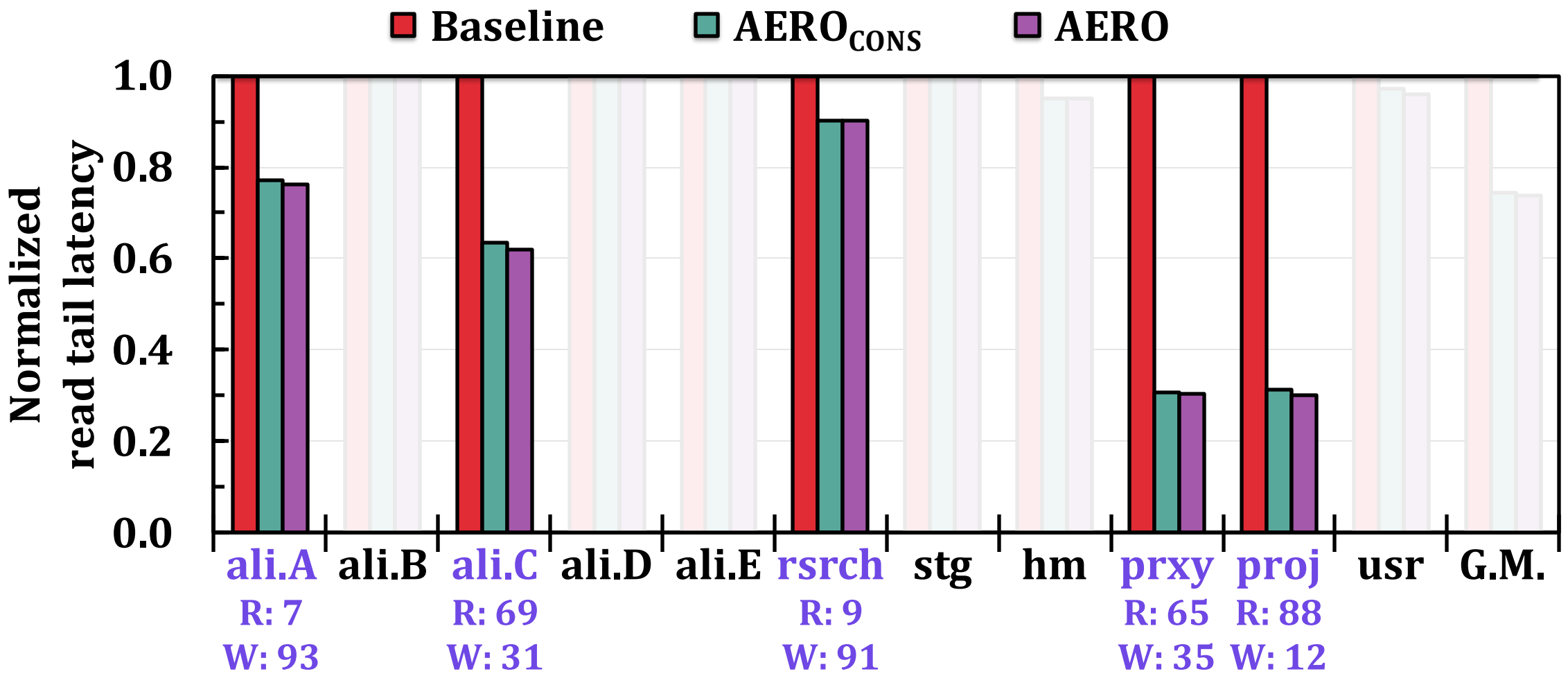
Result: Read Tail Latency (P99.99, 0.5K PEC)



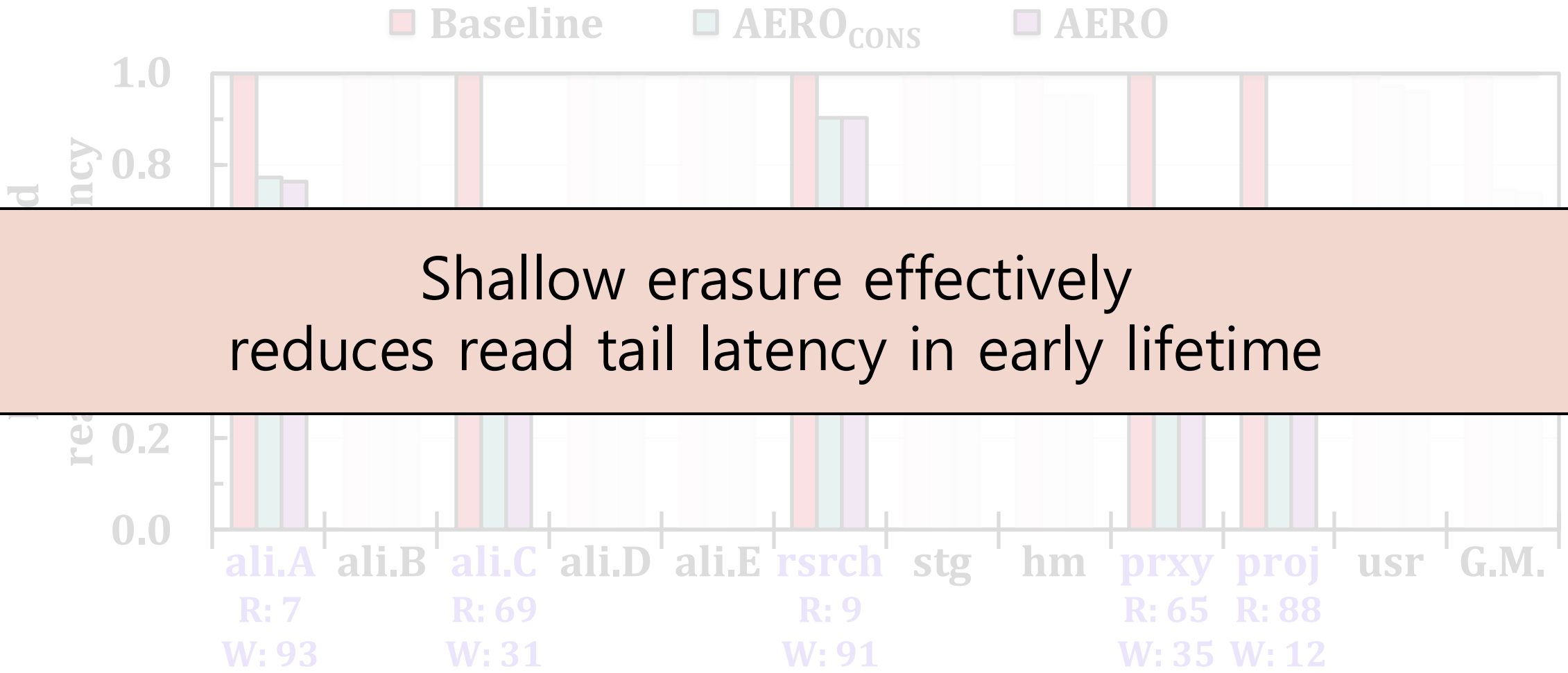
Result: Read Tail Latency (P99.99, 0.5K PEC)



Result: Read Tail Latency (P99.99, 0.5K PEC)

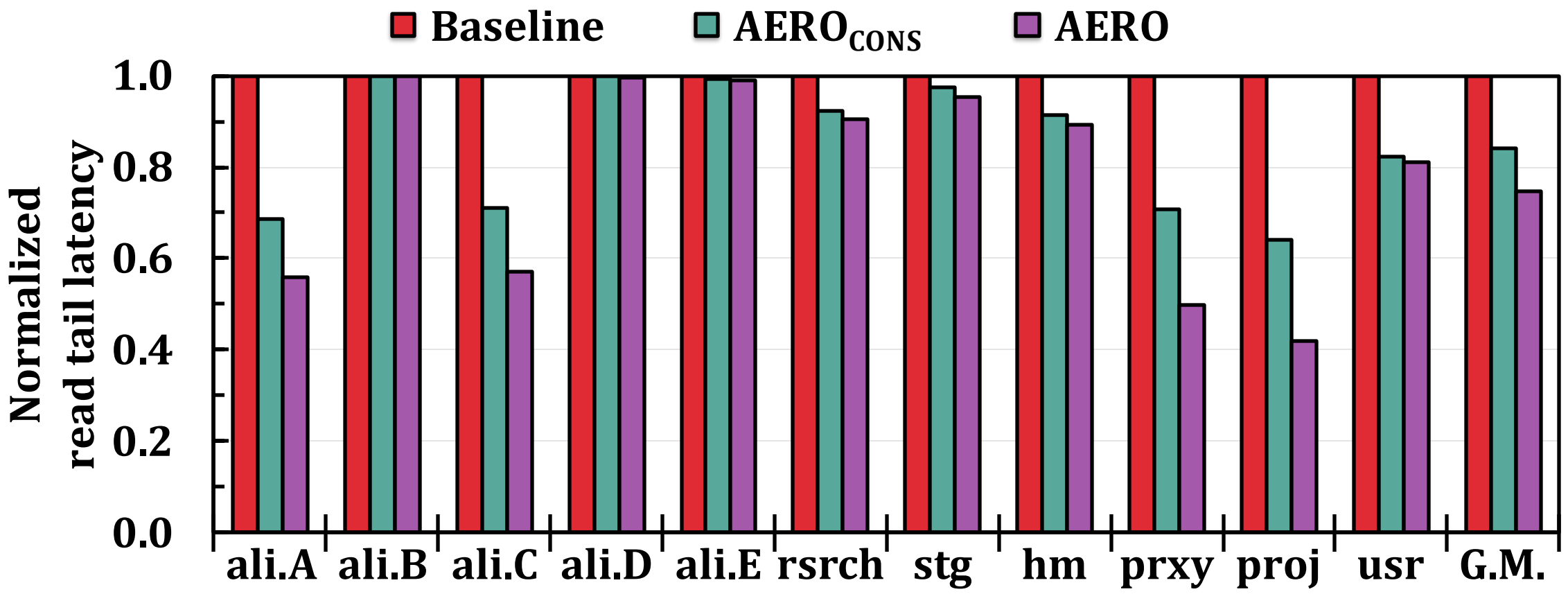


Result: Read Tail Latency (P99.99, 0.5K PEC)

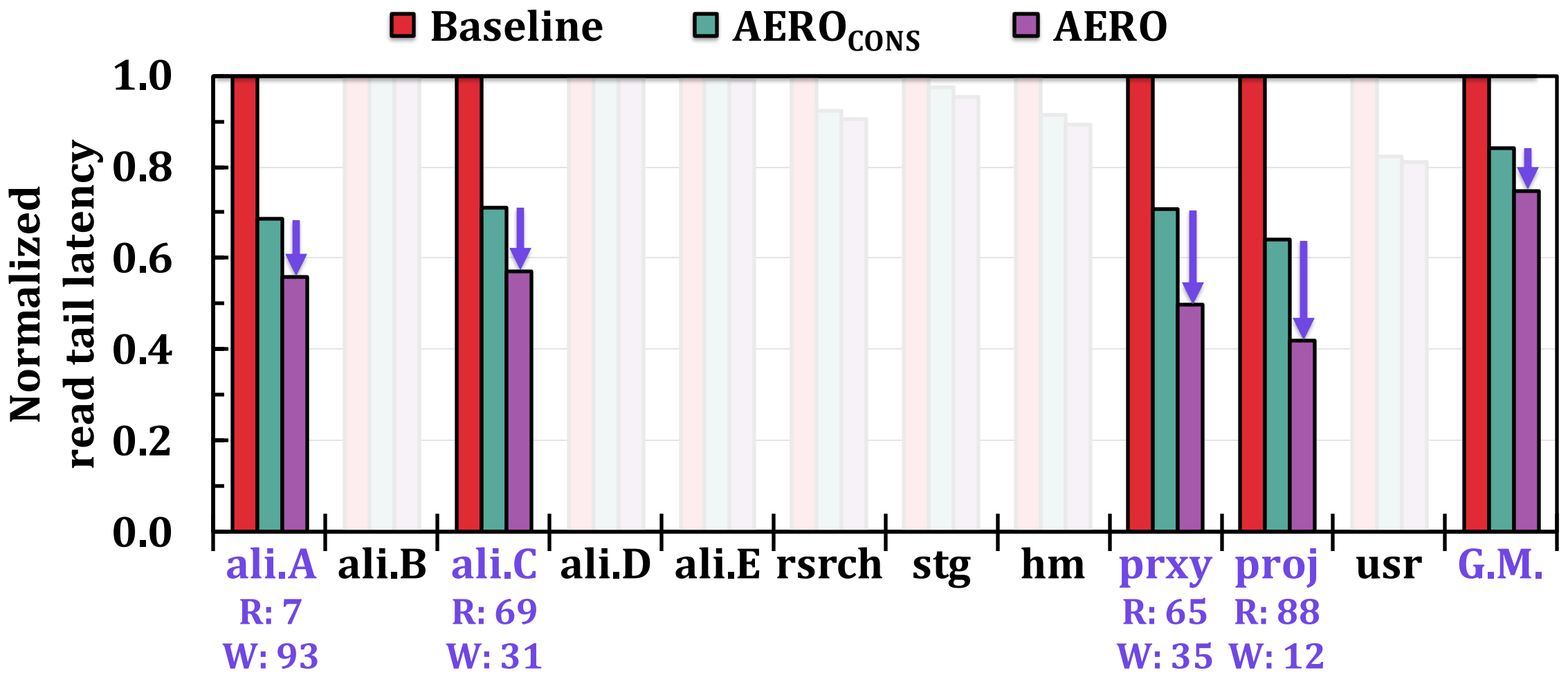


Shallow erasure effectively
reduces read tail latency in early lifetime

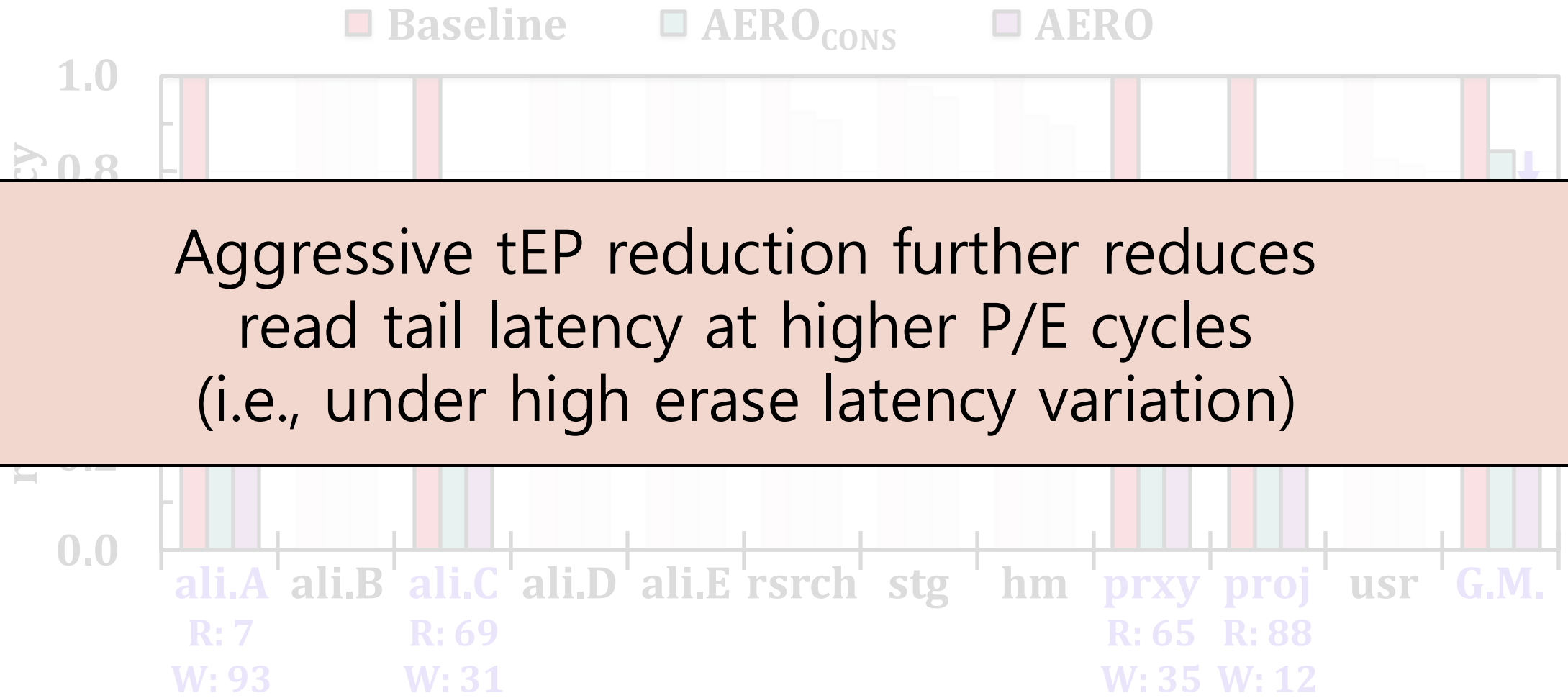
Result: Read Tail Latency (P99.99, 2.5K PEC)



Result: Read Tail Latency (P99.99, 2.5K PEC)



Result: Read Tail Latency (P99.99, 2.5K PEC)



Aggressive tEP reduction further reduces read tail latency at higher P/E cycles (i.e., under high erase latency variation)

Other Analyses in the Paper

- Limitations of exisiting techniques
- Implementation details
  - Handling misprediction
  - Impact of ECC-decoding latency
  - Multi-plane operations
- Evaluation
  - Comparison with other exisiting techniques
  - Impact of erase suspension
  - Sensitivy analysis for misprediction rate and correctable errors

Please refer to the paper!

AERO: Summary

- Problem: Long and fixed latency of erase operation in modern SSDs degrade lifetime and I/O performance

- Key idea: AERO (Adaptive ERase Operation)
  - Fail-bit-count-based Erase-Latency Prediction (FELP): dynamically adjusts erase latency
  - Shallow erasure: enables reducing latency of the first erase pulse
  - Aggressive tEP reduction: enables further reducing erase latency by exploiting the ECC capability

- Results: AERO significantly improves SSD lifetime by 43% and read tail latency by 26% (P99.99)

Conclusion

- Problem: Long and fixed latency of erase operation in modern SSDs degrade lifetime and I/O performance

- Key idea: AERO (Adaptive ERase Operation)
  - Fail-bit-count-based Erase-Latency Prediction (FELP): dynamically adjusts erase latency
  - Shallow erasure: enables reducing latency of the first erase pulse
  - Aggressive tEP reduction: enables further reducing erase latency by exploiting the ECC capability

- Results: AERO significantly improves SSD lifetime by 43% and read tail latency by 26% (P99.99)

White-Box Optimization Approaches for Ultra Large-Capacity NAND Flash-Based SSDs

NVRAMOS 2024

Prof. Jisung Park

COMPUTER ARCHITECTURE & OPERATING SYSTEMS LABORATORY

2024.10.24