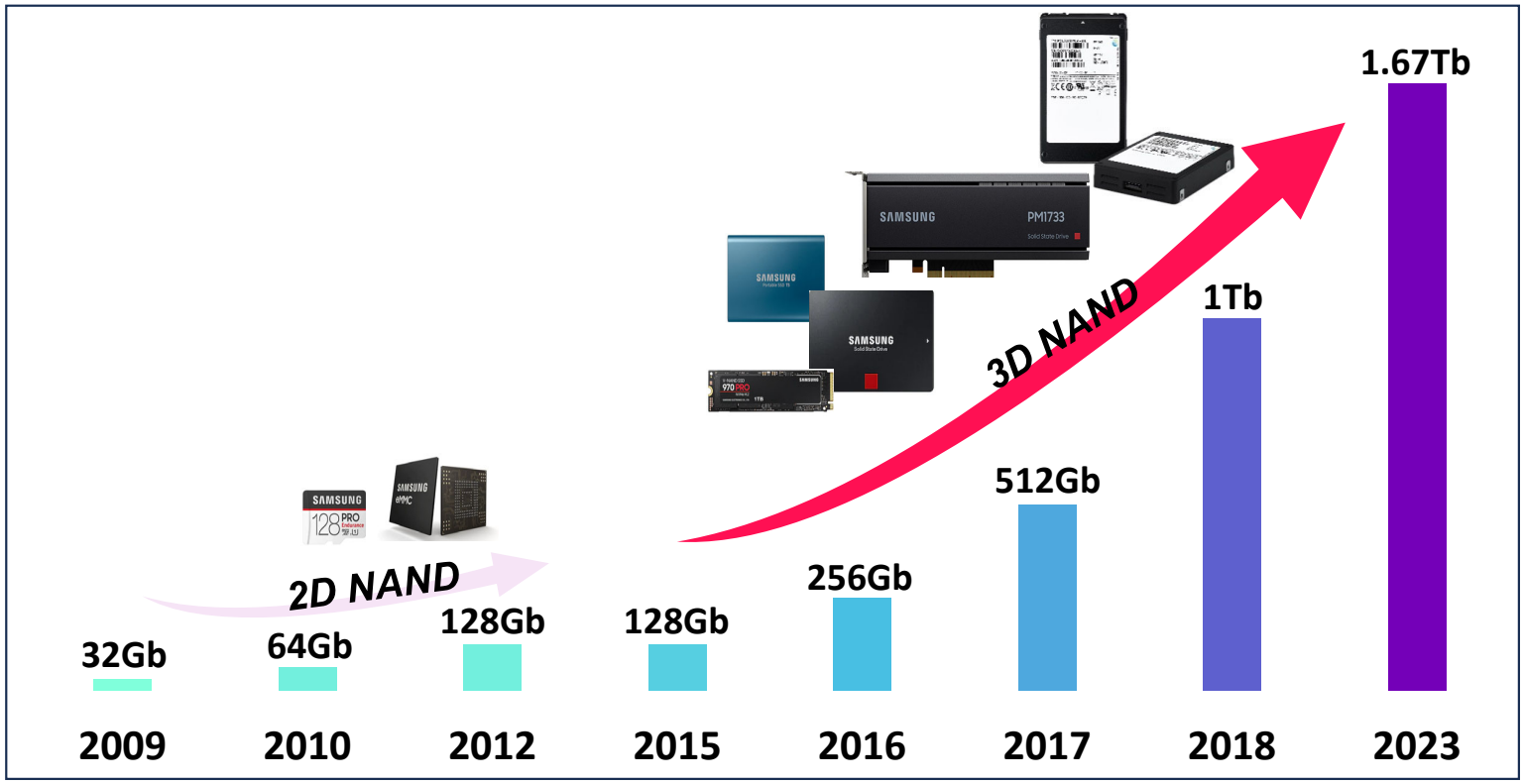
Error-Aware SSD Management Techniques for High-Performance SSDs

NVRAMOS 2025

Myoungjun Chun

Soongsil University

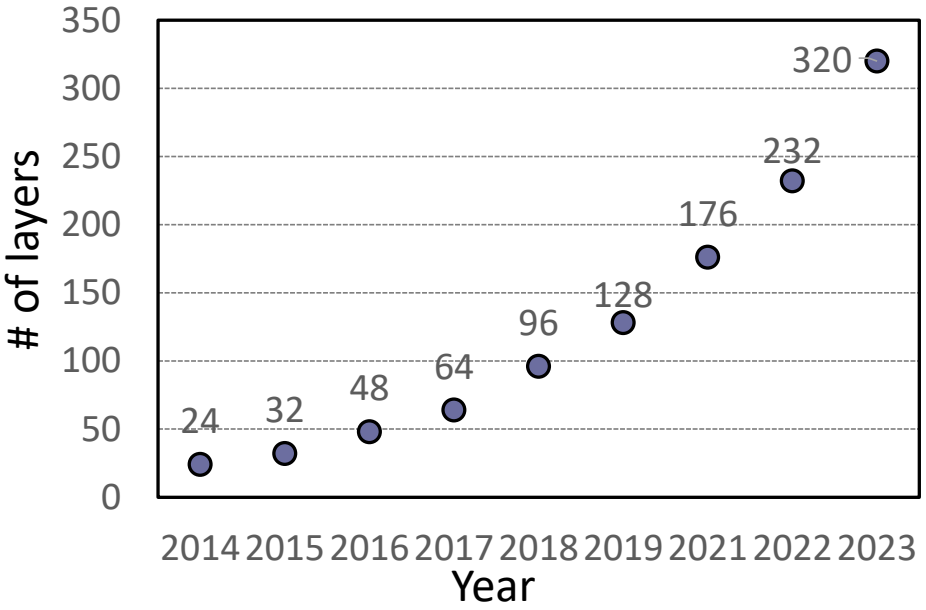
High-Density NAND Flash Storage



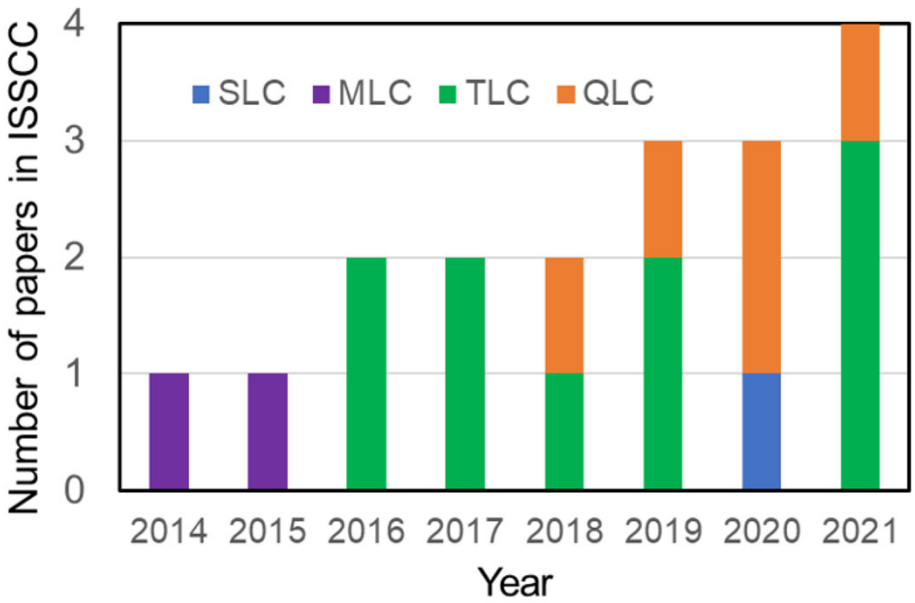
High-density NAND flash storage (SSD) has become ***de facto*** standards in storage systems

High-Density NAND Flash Trend

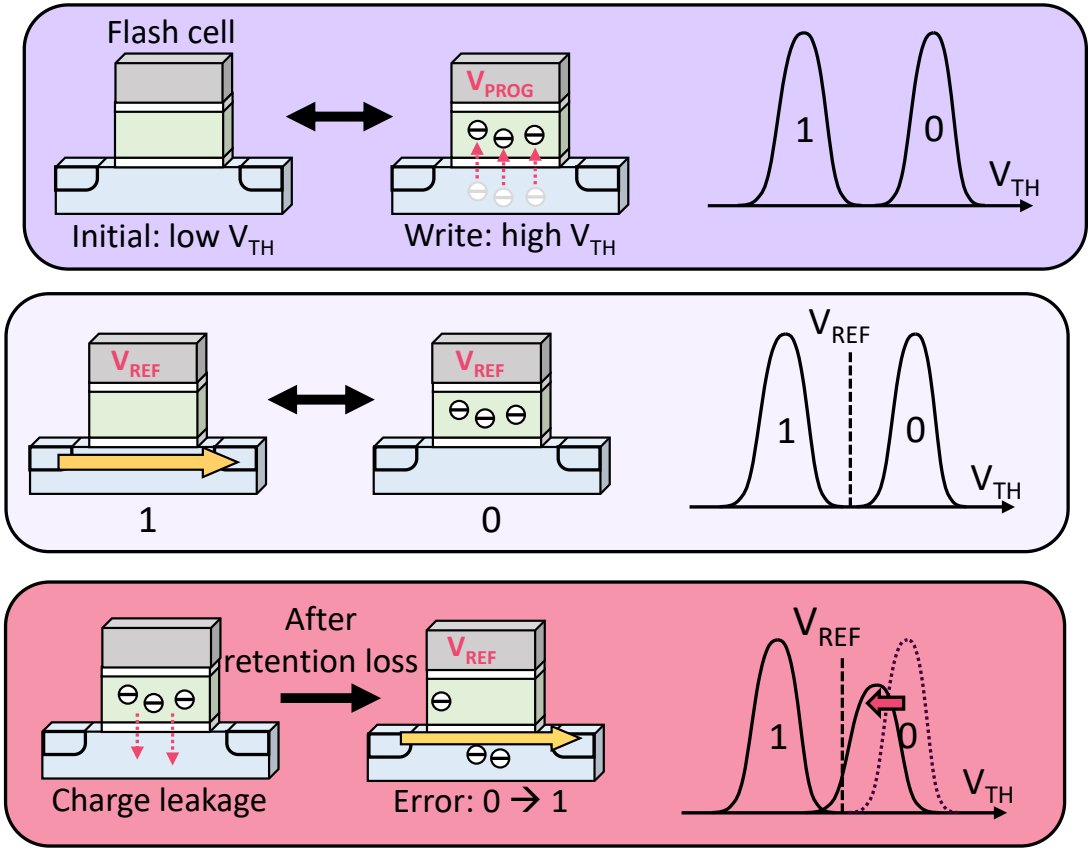
① Layer Stacking



② Increasing bit per cell



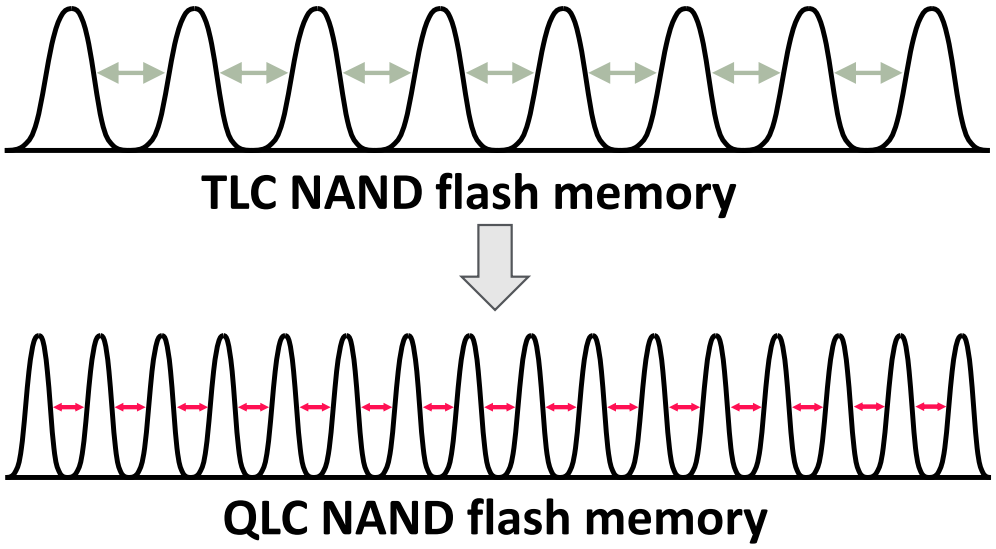
Errors in NAND Flash Memory



Errors in NAND Flash Memory

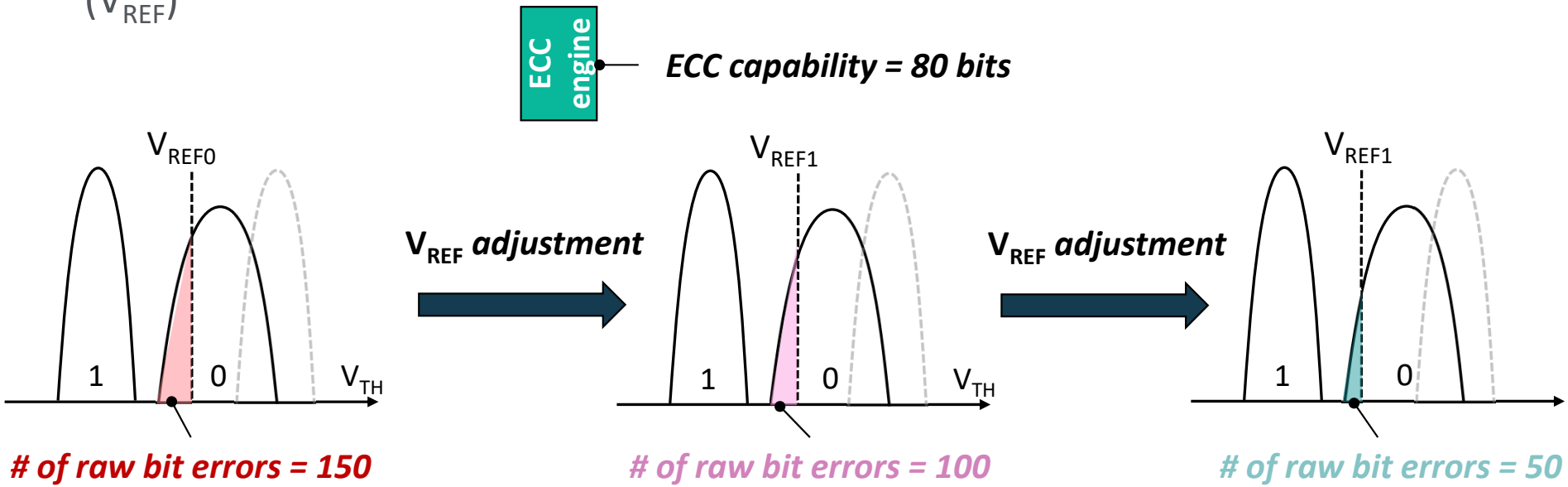
- **Increased bit density** makes flash cells more susceptible to error sources (e.g., retention loss, read/write disturbance)

- For example, a slight shift in V_{TH} can cause numerous bit errors in QLC NAND flash memory due to **the reduced V_{TH} margin**



Reliability Management Mechanism 1: Read-Retry

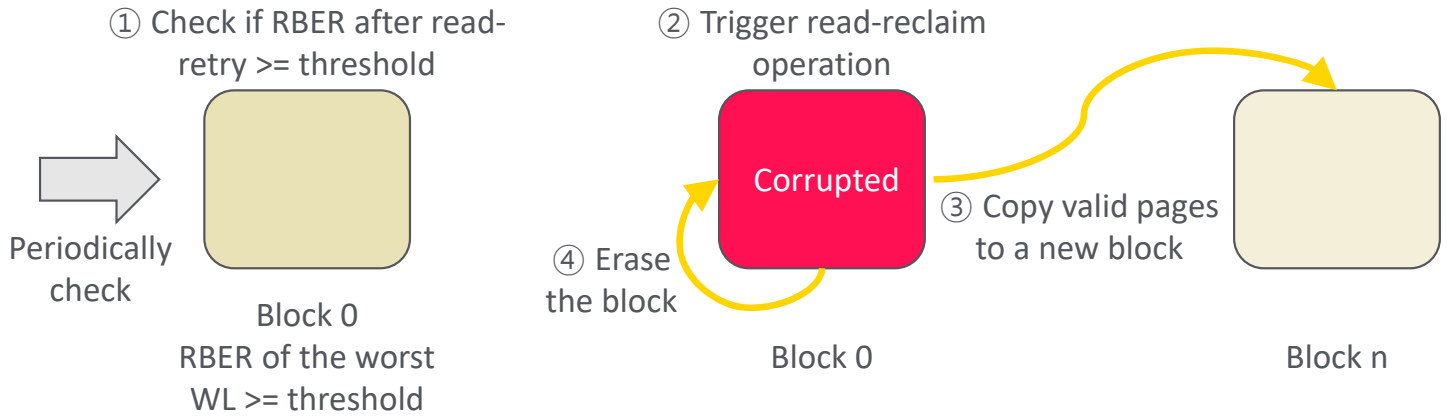
- A modern SSD adopts various **internal reliability management mechanisms**
  - **Read-retry operation**: re-read the target page with adjusted reference voltages (V_{REF})



Read-retry linearly **increase read latency**

Reliability Management Mechanism 2: Read Reclaim

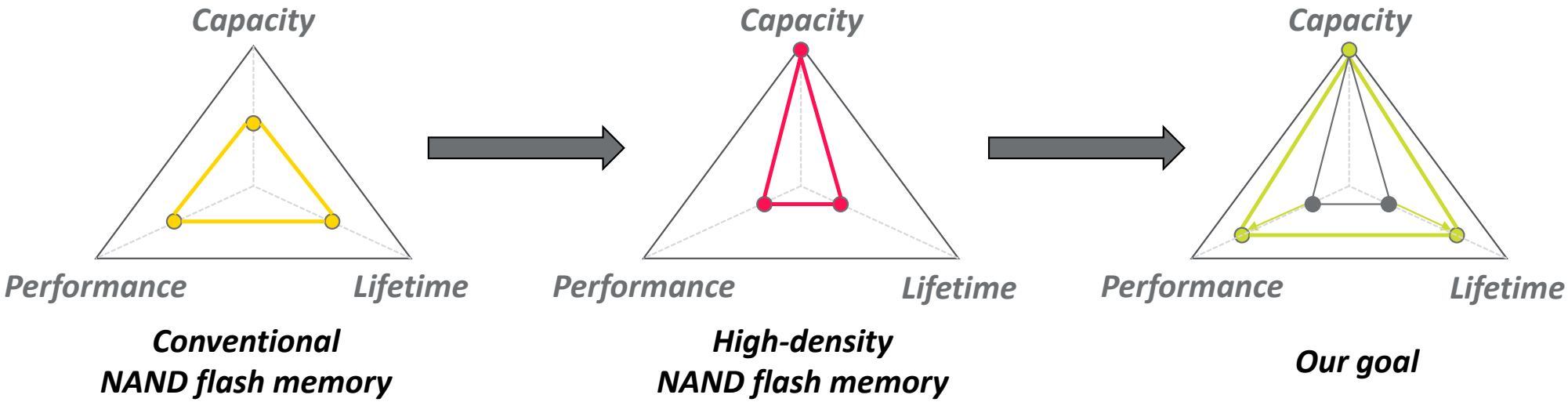
- A modern SSD adopts various **internal reliability management mechanisms**
  - **Read reclaim (RR)**: copies all the valid pages in the target block to free pages



Read-reclaim (RR) introduces **additional page copy** operations

Our Goal

- Essential **internal reliability management mechanisms** inevitably ***sacrifice performance/lifetime***



Conventional NAND flash memory

High-density NAND flash memory

Our goal

Our goal is to improve performance/lifetime
w/ **intelligent reliability management mechanisms**

Today's Talk

- Introduce two reliability management mechanisms that improve the performance and/or lifetime of SSDs
  - ① **Improve read bandwidth by optimizing read-retry**

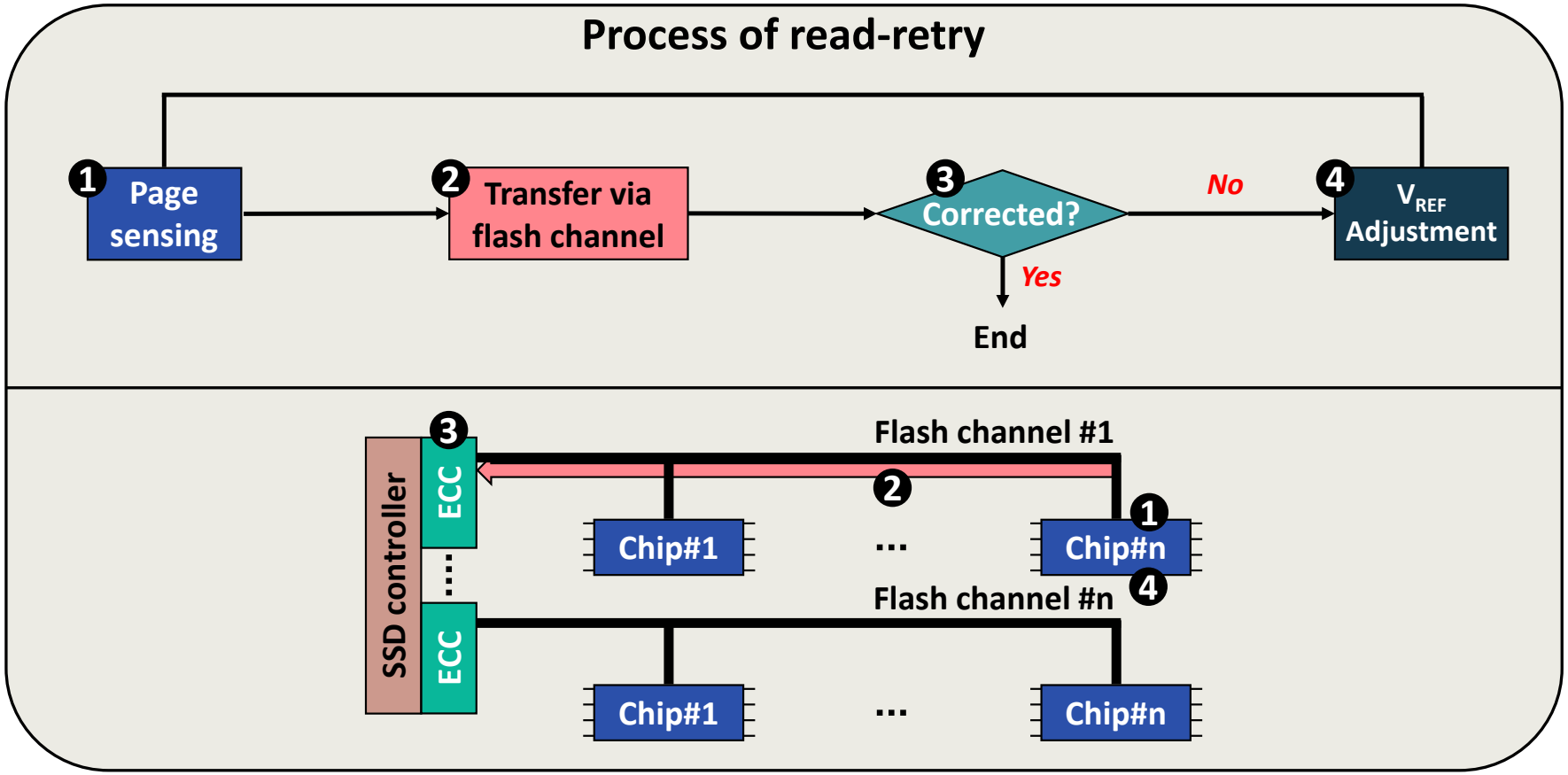
    RiF: Improving Read Performance of Modern SSDs Using an On-Die Early-Retry Engine, ***HPCA'24***

  - ② **Mitigate read reclaim overhead by efficiently managing read disturbance**

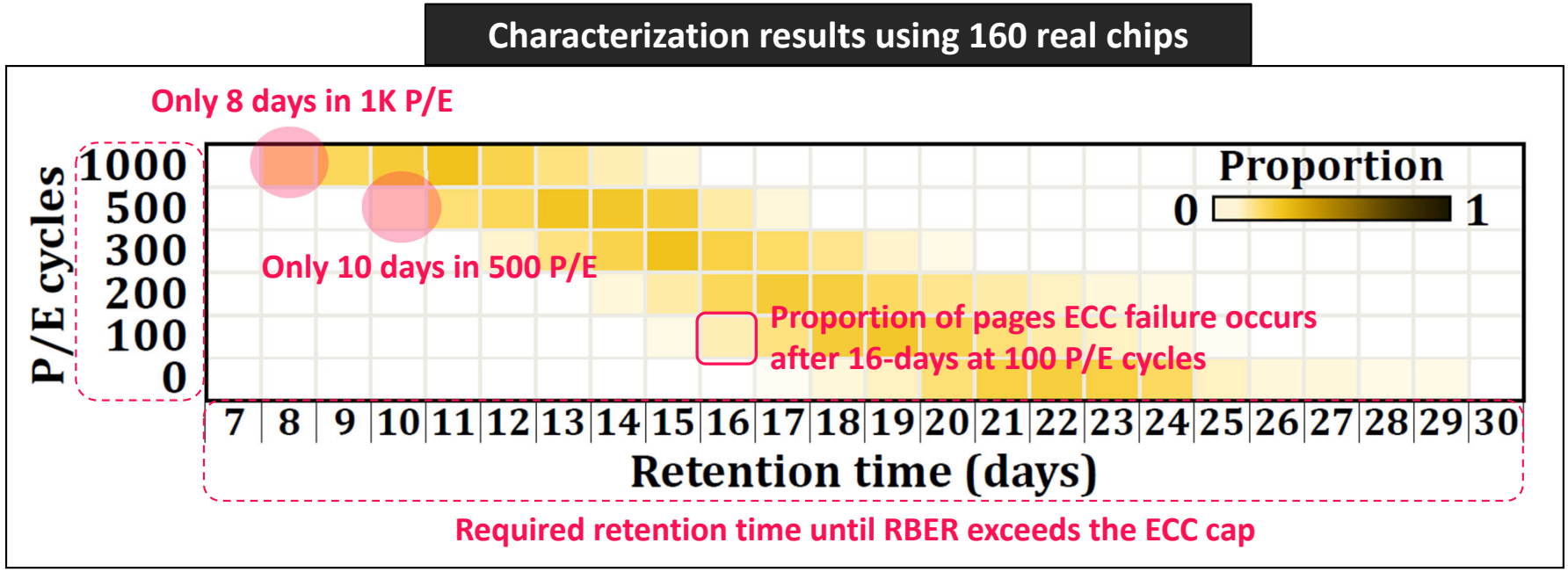
    STRAW: Stress-Aware WL-Based Read Disturbance Management for High-Density NAND Flash Memory, ***IEEE CAL'24, submitted to..***

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

Read-Retry in NAND Flash Memory (cont'd)



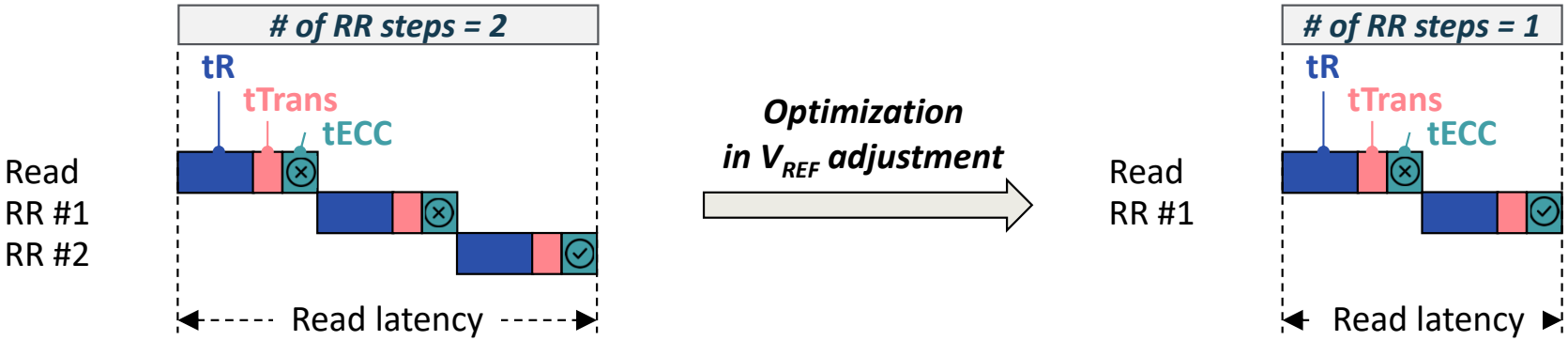
Frequency of Read-Retry in Modern NAND Flash



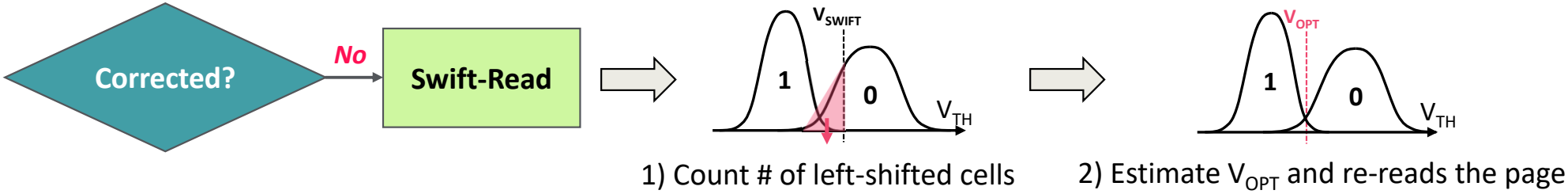
Read-retry (i.e., ECC failure) ***occurs very frequently*** in modern NAND flash

Read-Retry Optimization

- To mitigate the performance overhead of frequent read-retry, many prior works focused on ***reducing # of read-retry steps***



- State-of-the-art: Swift-Read (ISSCC'22) ***reduces # of read-retry steps to almost 1***

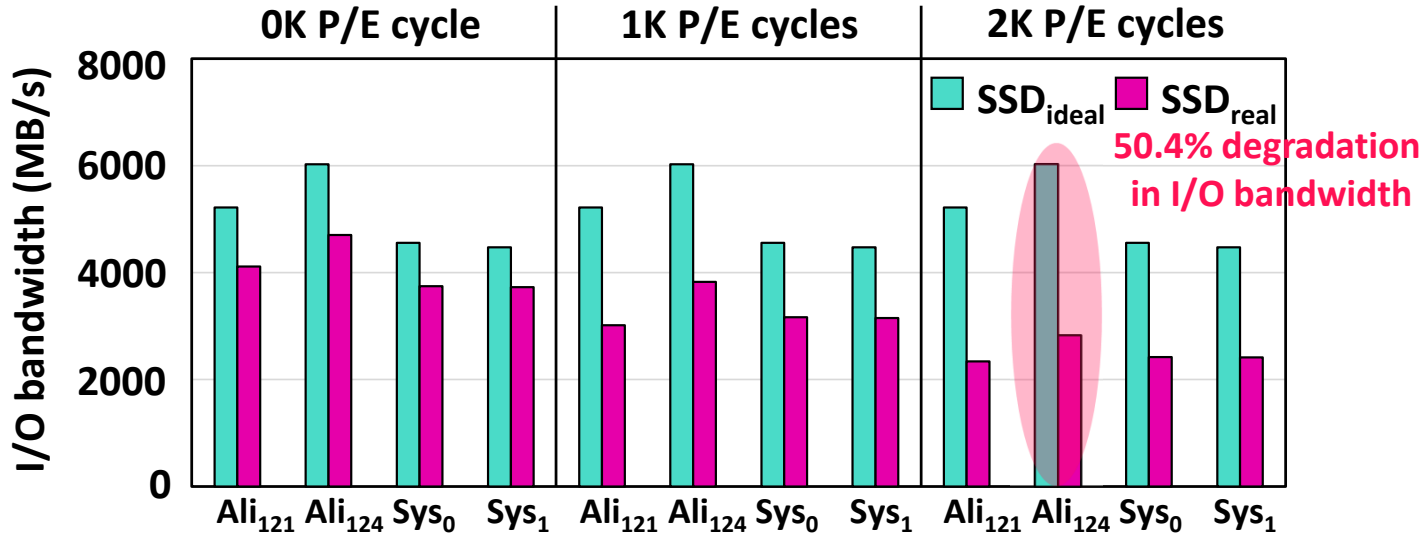


1) Count # of left-shifted cells

2) Estimate V_{OPT} and re-reads the page

Impact of Read-Retry on SSD's Performance

- **Compare I/O bandwidth of two SSDs using a modern SSD simulator (MQSim-E)**
  - ***SSD$_{real}$***: An SSD w/ ***optimal*** read-retry solution ***(# of RR steps = 1)***
  - ***SSD$_{ideal}$***: An ideal SSD w/ no read-retry

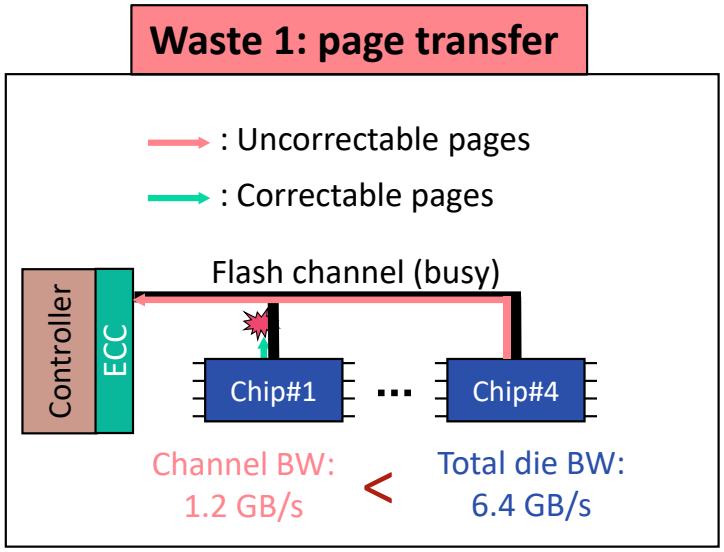


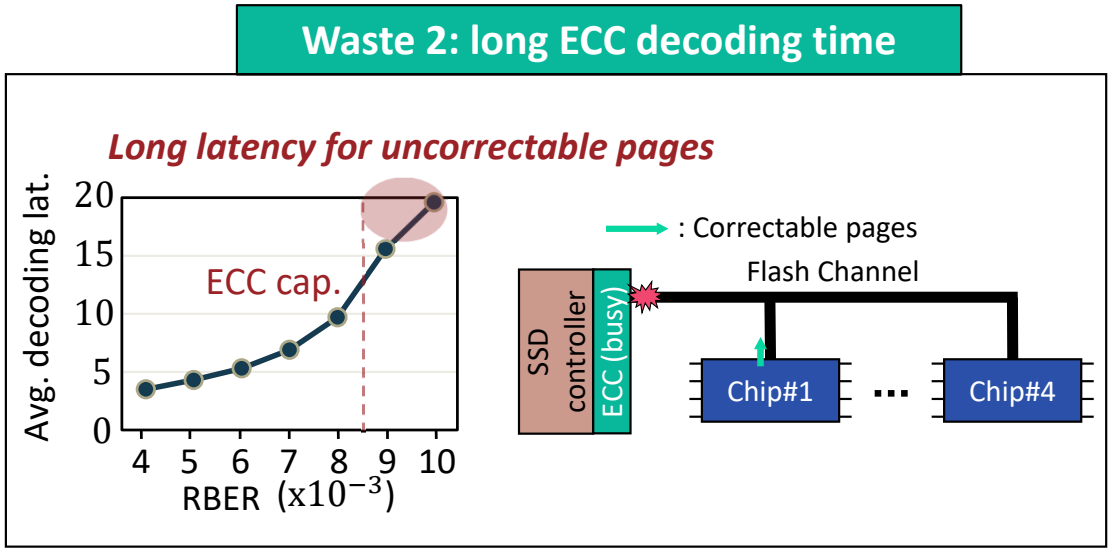
Read-Retry degrades SSD's performance significantly even ***with the optimal solution***

NVRAMOS

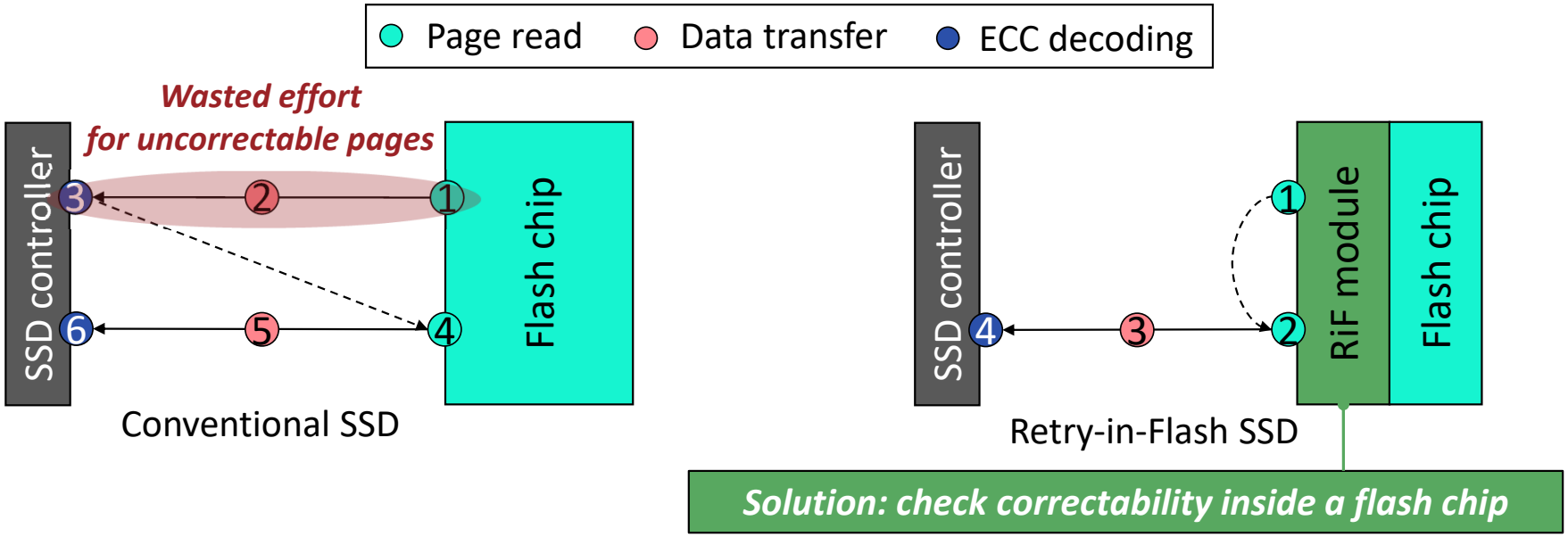
Root Cause Analysis

- ***Root Cause***: wasted effort for ***uncorrectable pages***





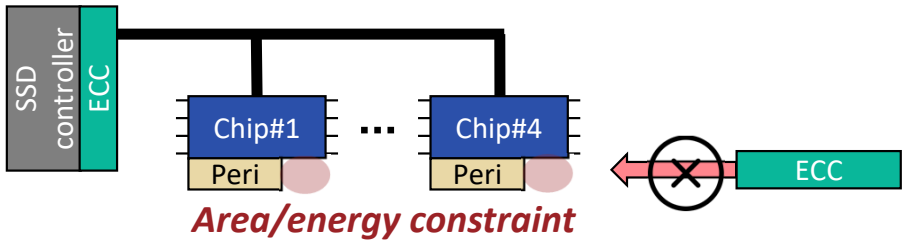
Key Idea: Retry-in-Flash (RiF)



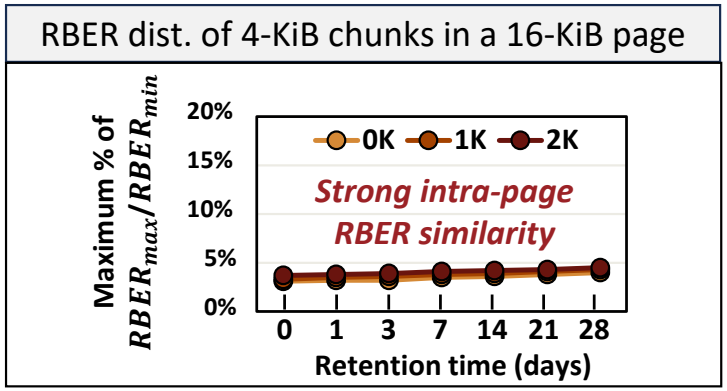
Effect 1: No wasted page transfer (❷)
Effect 2: No wasted decoding time (❸)

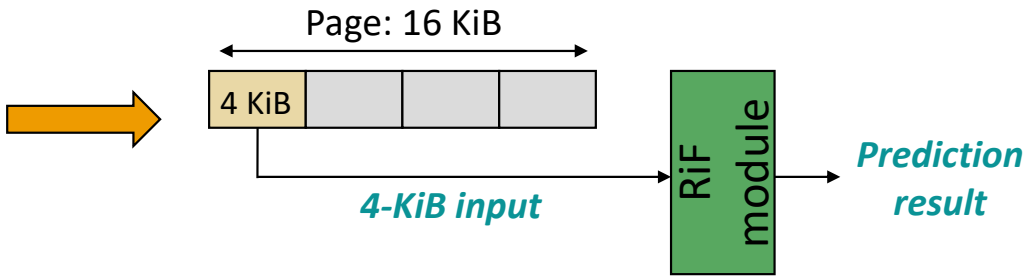
Implementation Challenge

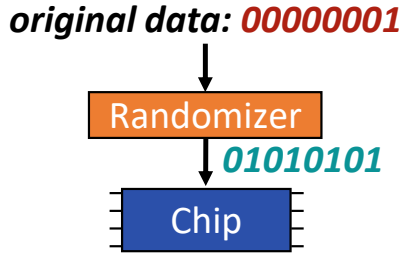
- Hard constraints on ***in-flash RiF module***



- ***Optimization 1:*** Subpage-based prediction







Implementation Challenge (cont'd)

- ***Optimization 2:*** Prediction-centric module

Uncorrectable pages do not require ***entire ECC decoding***
Instead, we simply want to know ***if a page is correctable or not***

Syndrome weight-based prediction

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

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

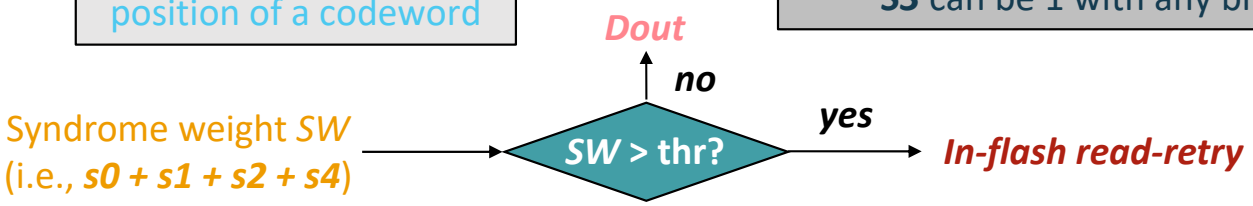
M = 4, N = 8

1's represent the bit position of a codeword

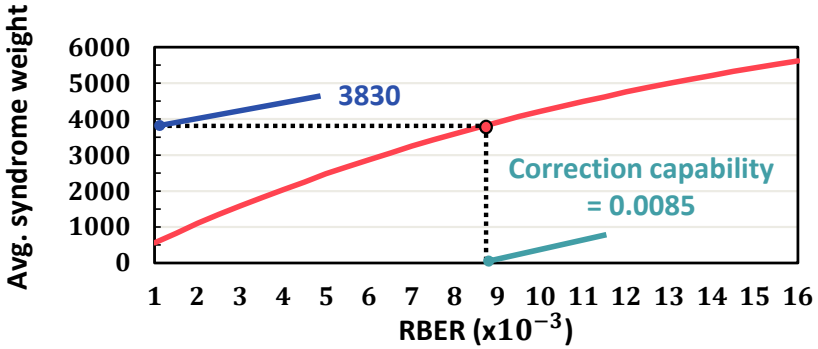
$$S^T = Hc^T = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} c_1 \oplus c_3 \oplus c_4 \oplus c_7 \\ c_0 \oplus c_1 \oplus c_2 \oplus c_5 \\ c_2 \oplus c_5 \oplus c_6 \oplus c_7 \\ c_0 \oplus c_3 \oplus c_4 \oplus c_6 \end{bmatrix}$$

S3 is 0 with no errors in ***c0, c3, c4, c6***

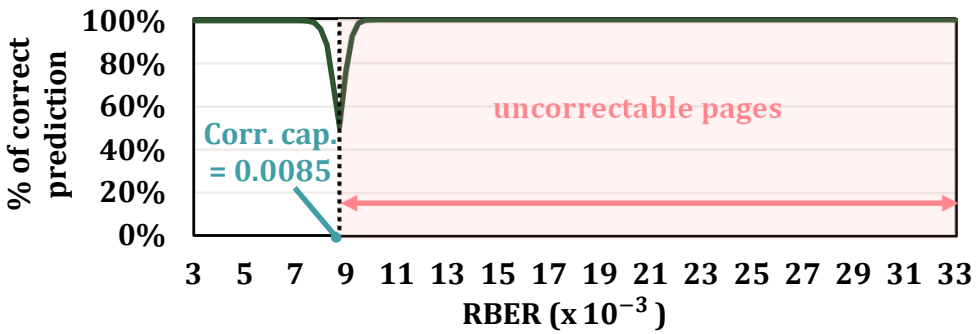
S3 can be 1 with any bit error



Validation Results



✓ ***Strong correlation***



✓ ***High prediction accuracy***

RiF module achieves ***98.7% prediction accuracy*** based on the ***syndrome weight-based prediction***

NVRAMOS

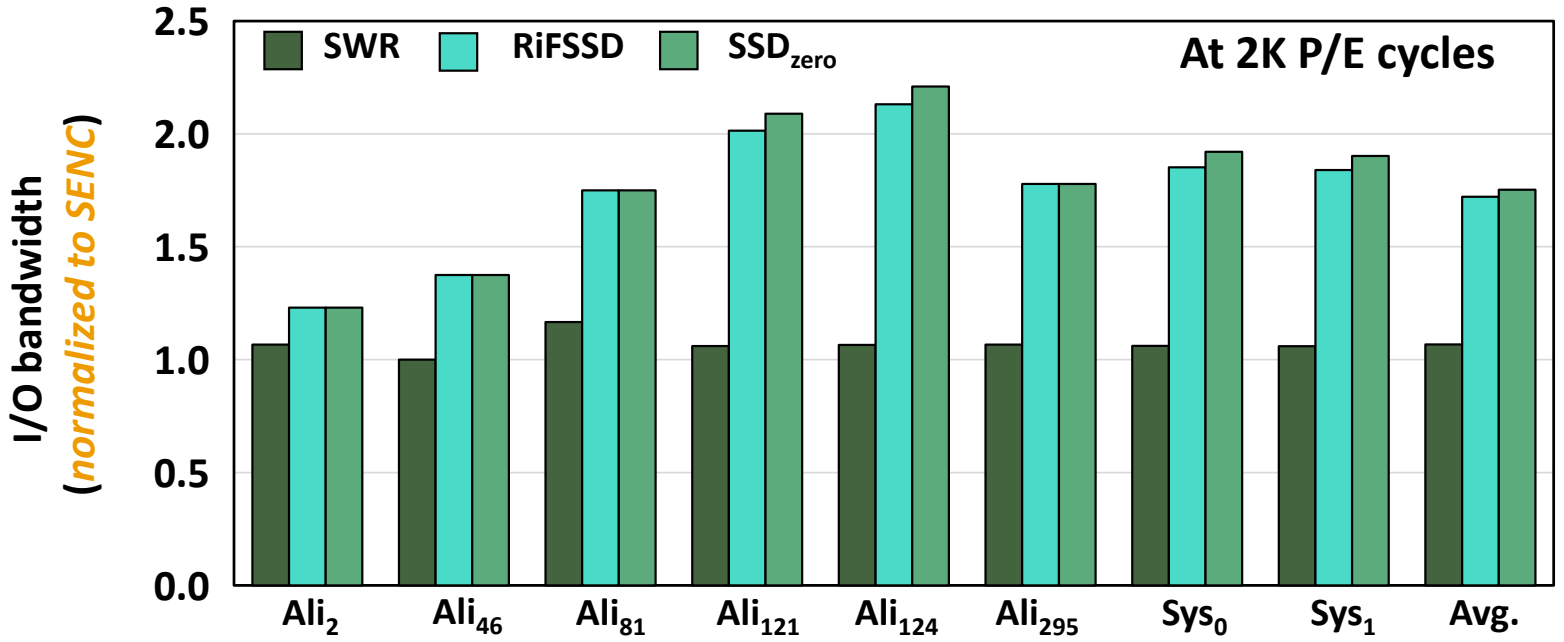
Experiment Settings

- ***Simulator:*** MQSim-E
  - Extend NAND flash models w/ real-device characterization results

- ***Workloads:*** 8 real-world traces
  - 6 from AliCloud traces
  - 2 from Systor traces

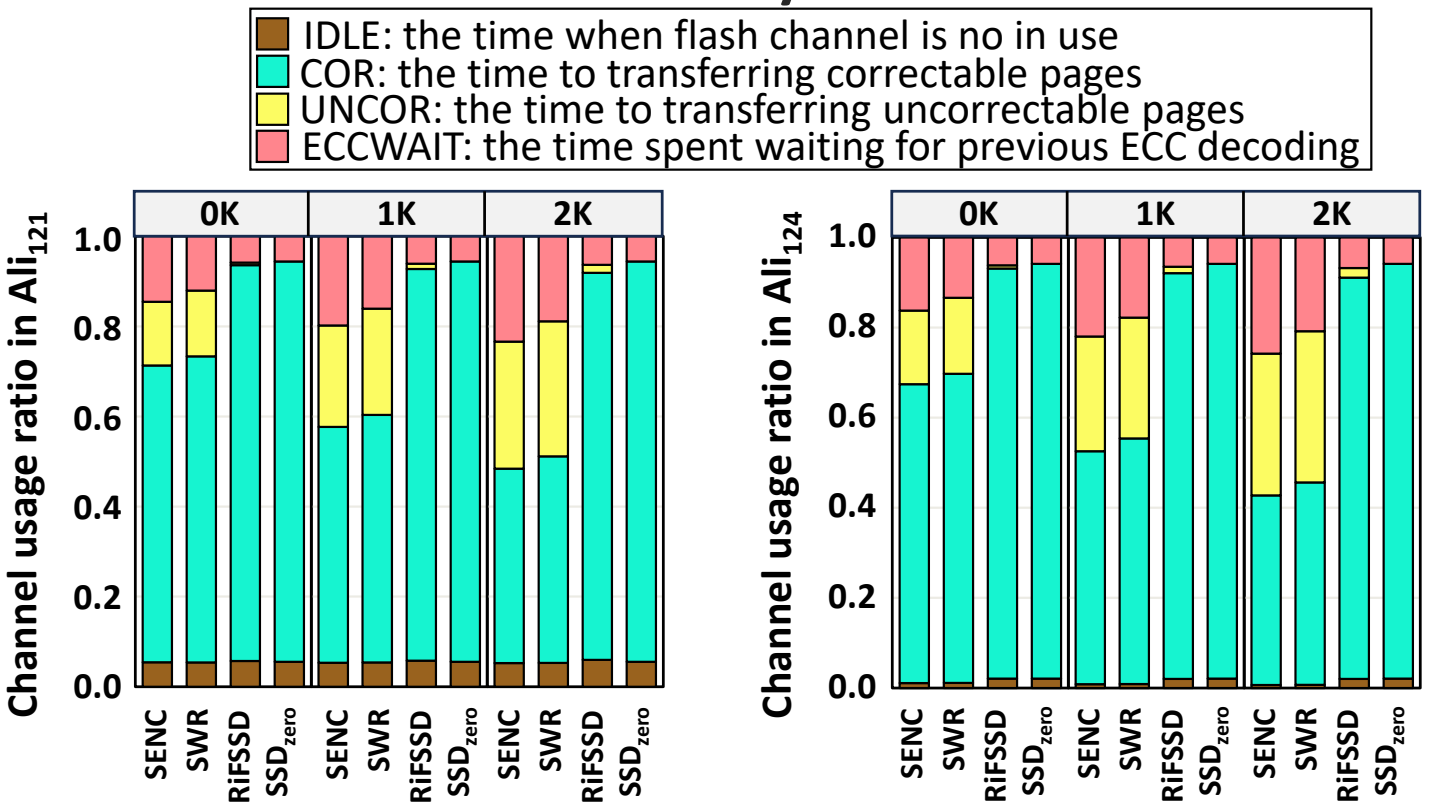
- ***Comparison***
  - **SSD_{zero}**: An ideal SSD where no read-retry occurs
  - **SENC**: A state-of-the-art read-retry mitigation scheme (Sentinel, MICRO21)
  - **SWR**: A state-of-the-art read-retry mitigation scheme (Swift-Read, ISSCC22)

Results: I/O Bandwidth



RiF improves the I/O bandwidth ***71.2%***, ***61.2%*** over SENC, SWR

Results: Channel Efficiency



RiF efficiently reduces ***wasted channel bandwidth*** from ***UNCOR, ECCWAIT***

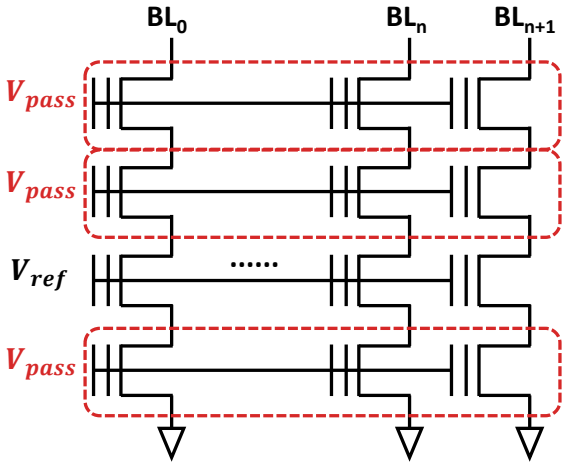
Results: Overhead Analysis

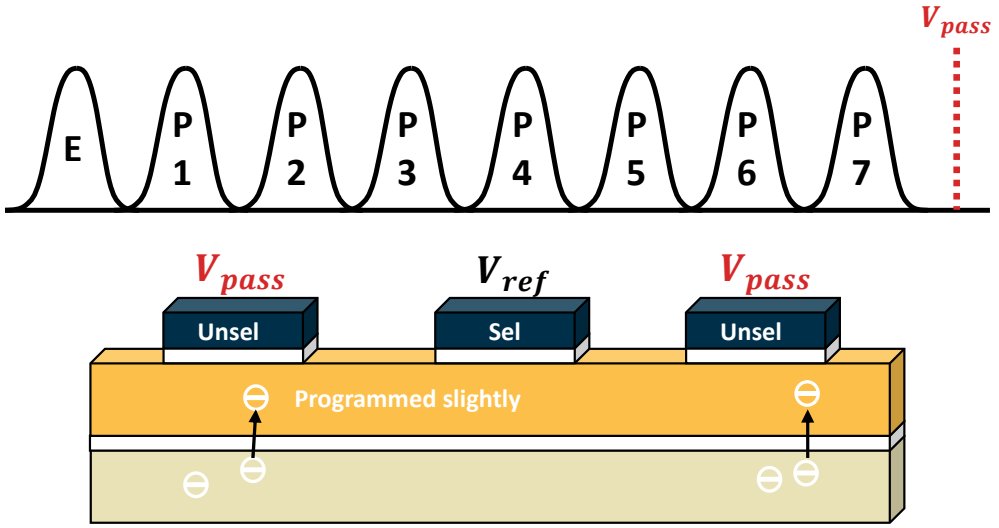
- **Experimental environment**: 130 nm process at a 100 MHz operating frequency
- **Area overhead**: 0.012 mm^2 (the typical area of modern flash dies around 100 mm^2)
- **Power consumption**: 1.28 mW (but when a read retry is needed, the RiF scheme reduces the energy consumption by about 907 nJ)

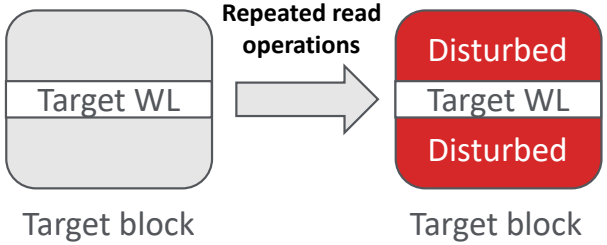
STRAW: Stress-Aware WL-Based Read Disturbance Management for High-Density NAND Flash Memory

Read Disturbance

- When a WL is read, applying a large pass-through voltage (V_{pass}) to ***all non-target WLs causes read disturbance***



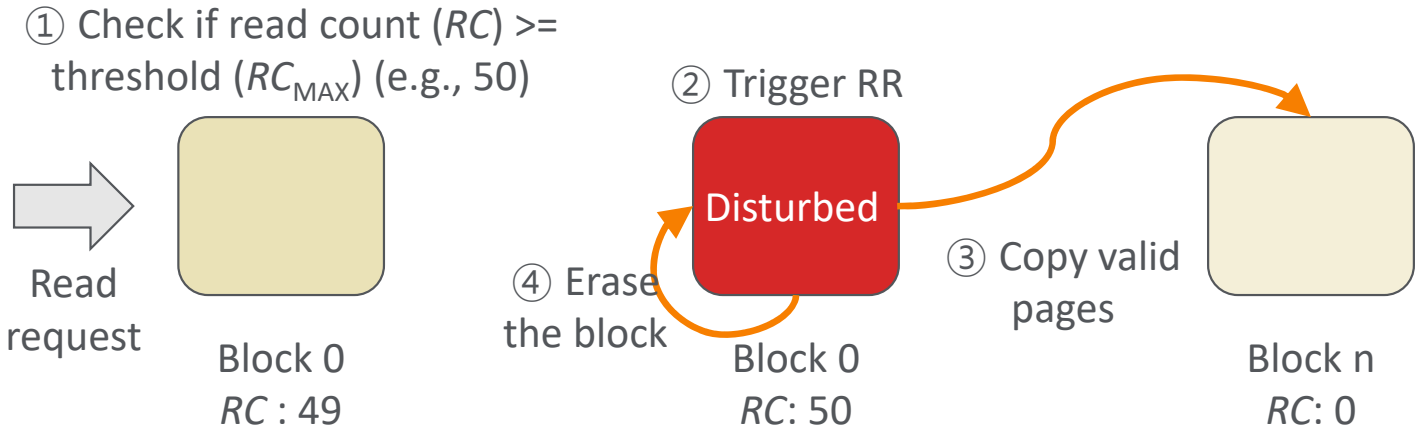




Existing Solution: Read Reclaim

- The amount of data stored in non-target WL rapidly increases; it exceeds a ***hundred MBs*** in 321-layers high-density NAND flash memory

- However, all existing solutions are based on a simple internal management task, ***read reclaim (RR)***

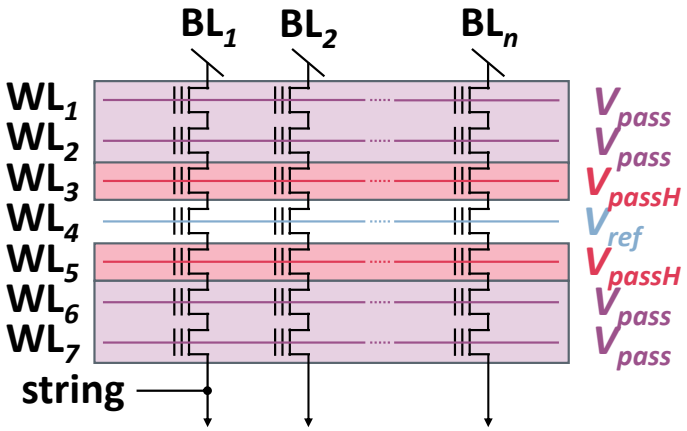


RR can significantly degrade the lifetime and performance of a modern SSD

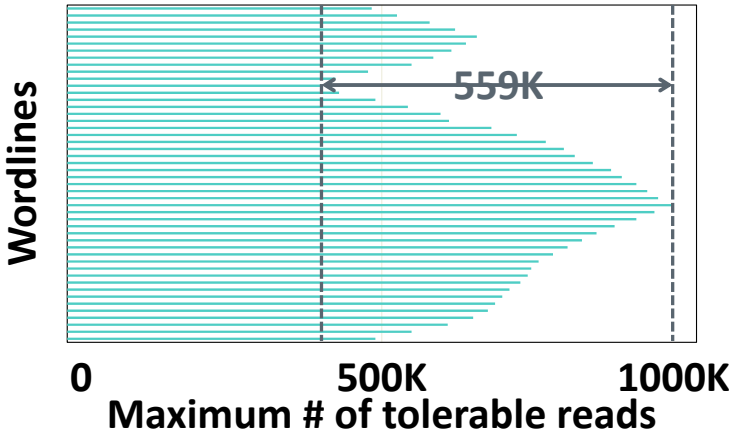
Limitation 1: Block-Level Granularity

- Existing solutions manage read-disturbance at ***a block granularity;*** they assume the effect of read-disturbance is ***symmetry***

- However, the reliability impact of read disturbance ***significantly varies across WLs*** in high-density 3D NAND flash memory due to two key factors



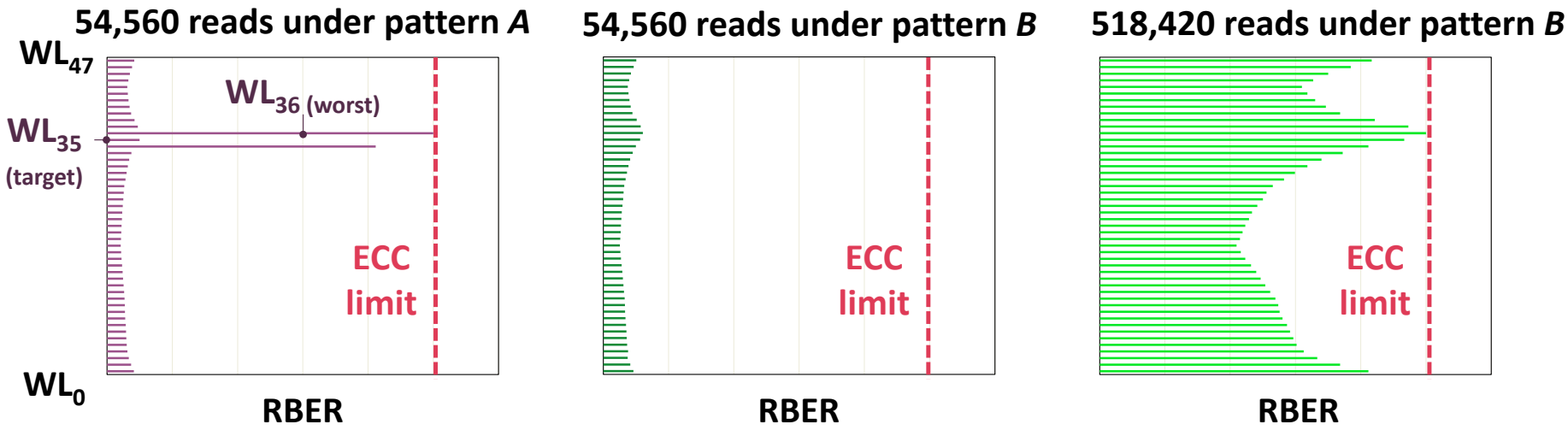
Factor 1: V_{passH} is around 10% higher than default V_{pass}



Factor 2: *Read-disturbance tolerance varies among WLs*

Limitation 1: Block-Level Granularity (cont'd)

- Using real 3D TLC NAND flash chips, we measure **the maximum tolerable read count** with two distinct read patterns
  - **Pattern *A***: Read only the page at WL_{35}, which is adjacent to the worst WL
  - **Pattern *B***: Read all pages in the block sequentially

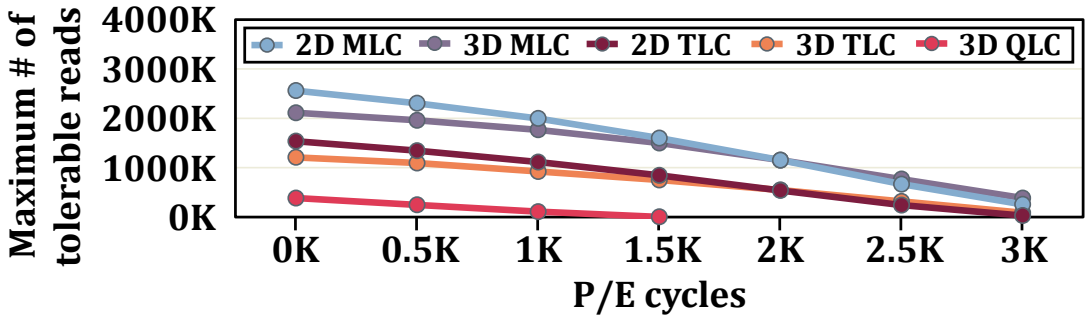


The block-level RR approach must set RC_{MAX} *to 54,560*

Limitation 2: Reactive Manner

- Existing solutions manage read-disturbance ***reactively;*** they unable to ***directly mitigate*** the significant reliability impact of read disturbance

- However, ***the read-disturbance tolerance*** of high-density NAND flash memory is ***significantly reduced***



The tolerable read counts of 3D QLC are ***94.6%*** lower than those of 2D MLC

Primary Goal and Key Mechanisms

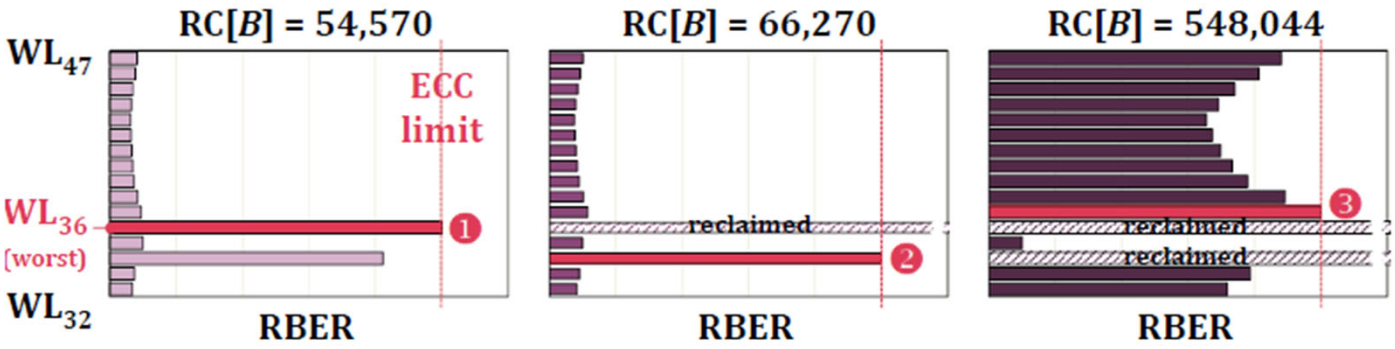
- Our goal in this work is to develop a new read-disturbance management technique that:

  ① **minimizes premature (unnecessary) RR** operations while ensuring data reliability -> propose ***Stress-Aware WL-Based Read Reclaim (WR^2)***

  ② **proactively enhances the read-disturbance tolerance** of high-density NAND flash memory -> propose ***Stress-Reduced Read (SR^2)***

Key Mechanism 1: WR^2

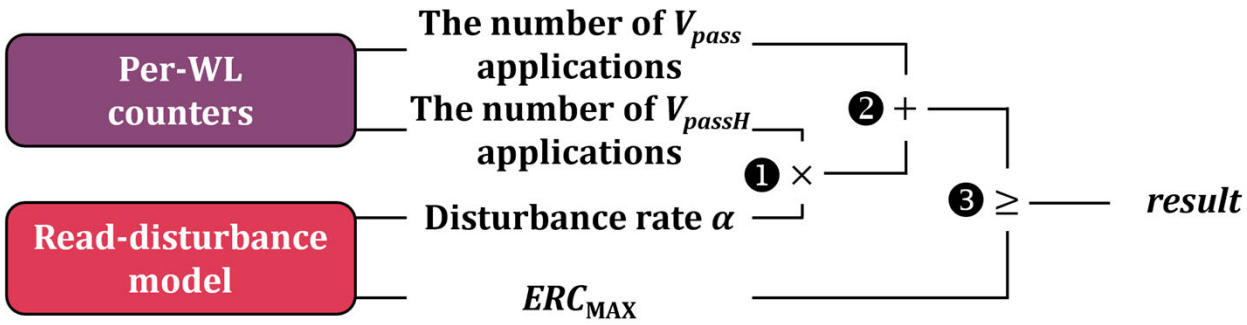
- WR^2 is based on our key insight that the read disturbance should be managed at ***a finer granularity, not per block***
  ① monitors the heterogeneous reliability impacts of read disturbance on **individual WLs**
  ② **reclaims only the heavily-disturbed WLs** in a timely manner



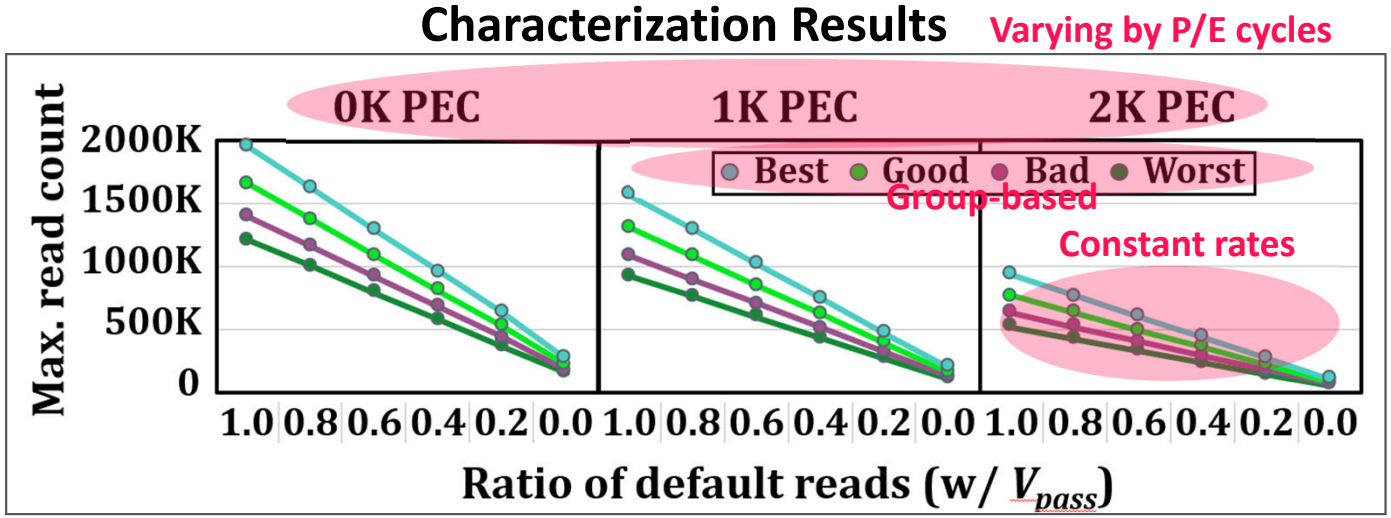
An example how WR^2 manages read disturbance under pattern *A*

Key Mechanism 1: WR^2 (cont'd)

- How to determine whether a WL is heavily disturbed? WR^2 employs:
  - A new read-disturbance model: quantify two key factors that contribute to read-disturbance
    - ERC_{MAX}: the maximum number effective reads of individual WLs (w/ default V_{pass})
    - **Disturbance rates (α)**: the ratio of read-disturbance stress from V_{passH}, to that from V_{pass}
  - Per-WL counters: store the **read count (RC)** for each WL



Characterization Results: WR²



Final Model

PEC	ERC_{MAX}	α	ERC_{MAX}	α	ERC_{MAX}	α	ERC_{MAX}	α
0	1947K	8.0	1657K	7.8	1391K	7.6	1210K	7.4
0.5K	1806K	8.3	1521K	8.0	1270K	7.9	1094K	7.7
1K	1567K	8.6	1309K	8.4	1087K	8.2	922K	7.9
1.5K	1310K	9.0	1086K	8.8	899K	8.6	749K	8.3
2K	933K	9.5	767K	9.2	627K	9.0	518K	8.7
2.5K	539K	10.1	438K	9.7	354K	9.5	288K	9.2
3K	111K	10.7	89K	10.3	71K	10.0	58K	9.7
	Best		Good		Bad		Worst	

Key Mechanism 2: SR^2

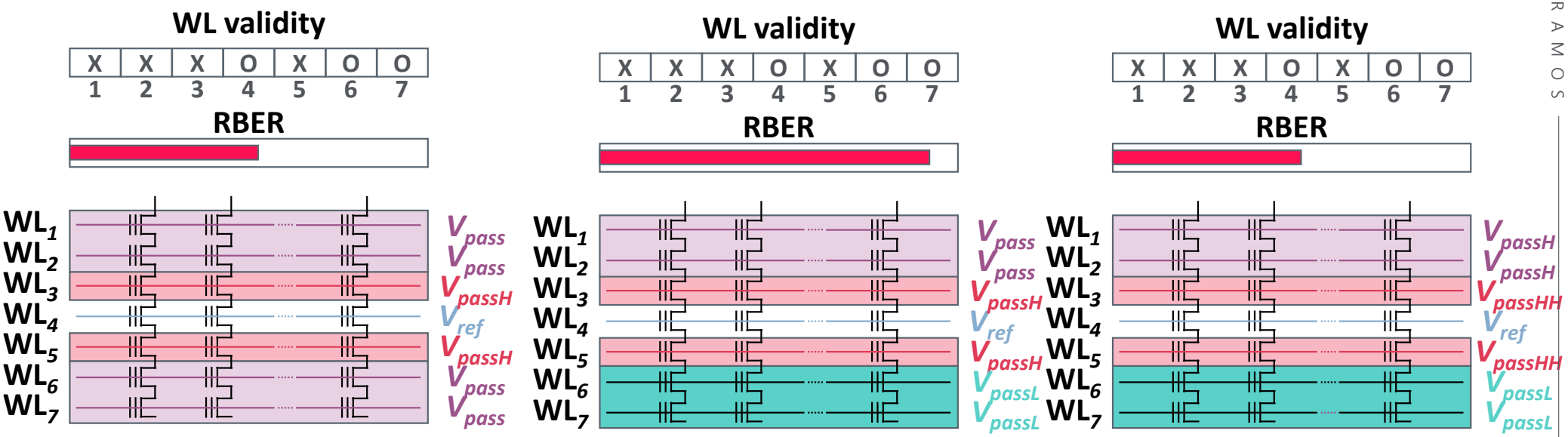
- ***V_{pass} scaling*** is highly effective for mitigating read-disturbance stress; however, it significantly increases # of bit errors
  - ***Side effect***: In modern flash memory, additional bit errors can significantly increase the read latency

- We design an effective V_{pass} scaling technique **without introducing additional bit errors** based on two key intuitions:

  ① applying ***a higher V_{pass}*** to certain non-target WLs can ***reduce the RBER of the target WL***

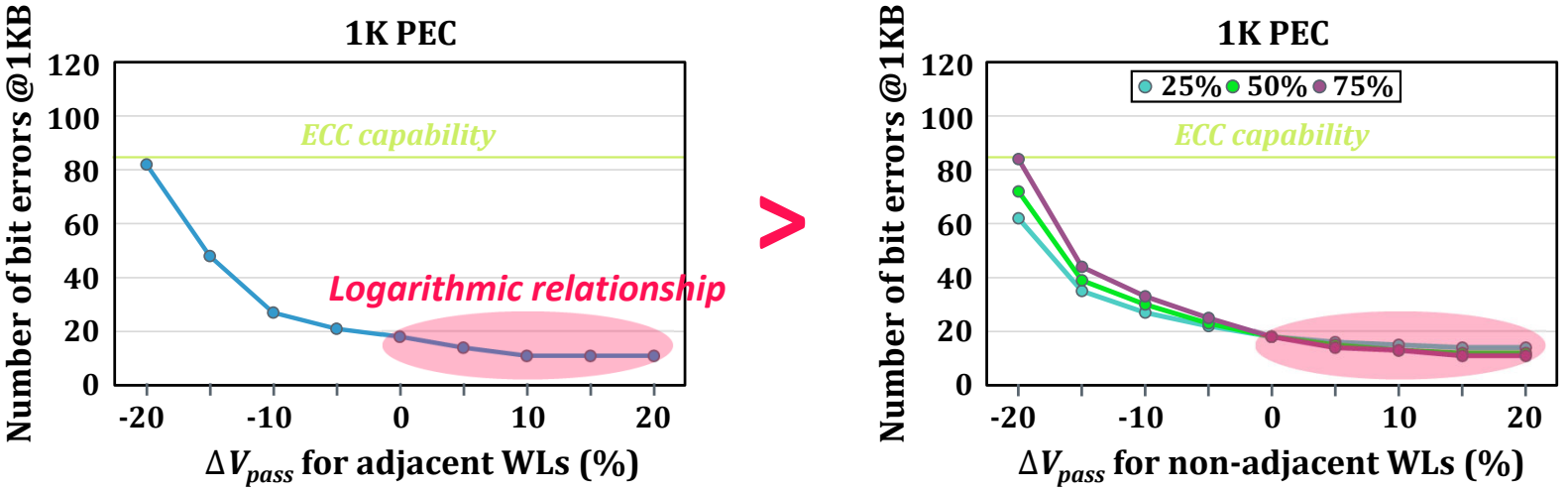
  ② the accumulated read-disturbance impact on ***invalid WLs*** does not affect data reliability

Key Mechanism 2: SR^2 (cont'd)



$$V_{passL} < V_{pass} < V_{passH} < V_{passHH}$$

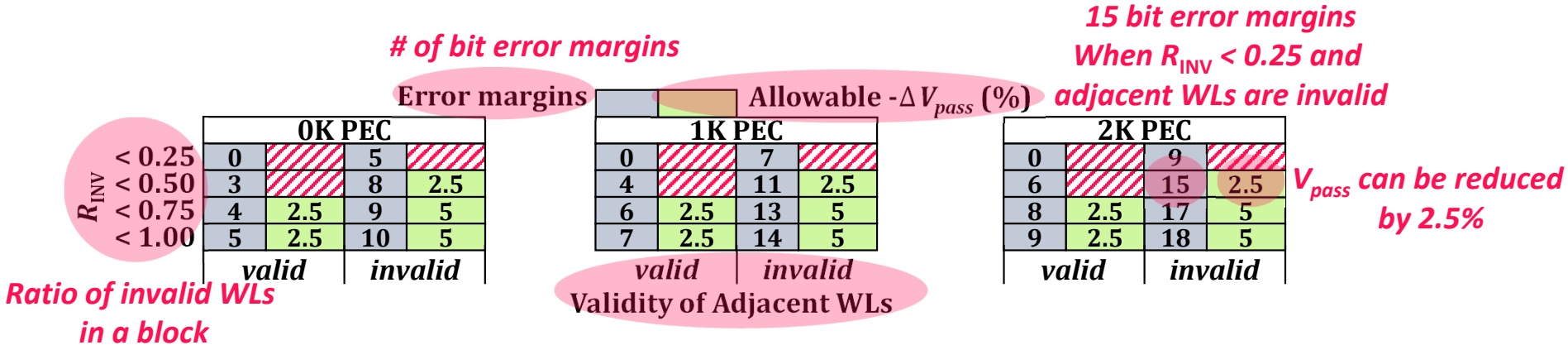
Characterization Results: SR[2]



- Based on the characterization, we validate:
  ① It is possible to reduce errors by ***increasing V_{pass}*** on non-target WLs
  ② ***V_{pass} on the two adjacent WLs*** has a greater impact on errors than V_{pass} on non-adjacent WLs

V_{pass} Reduction Model: SR^2

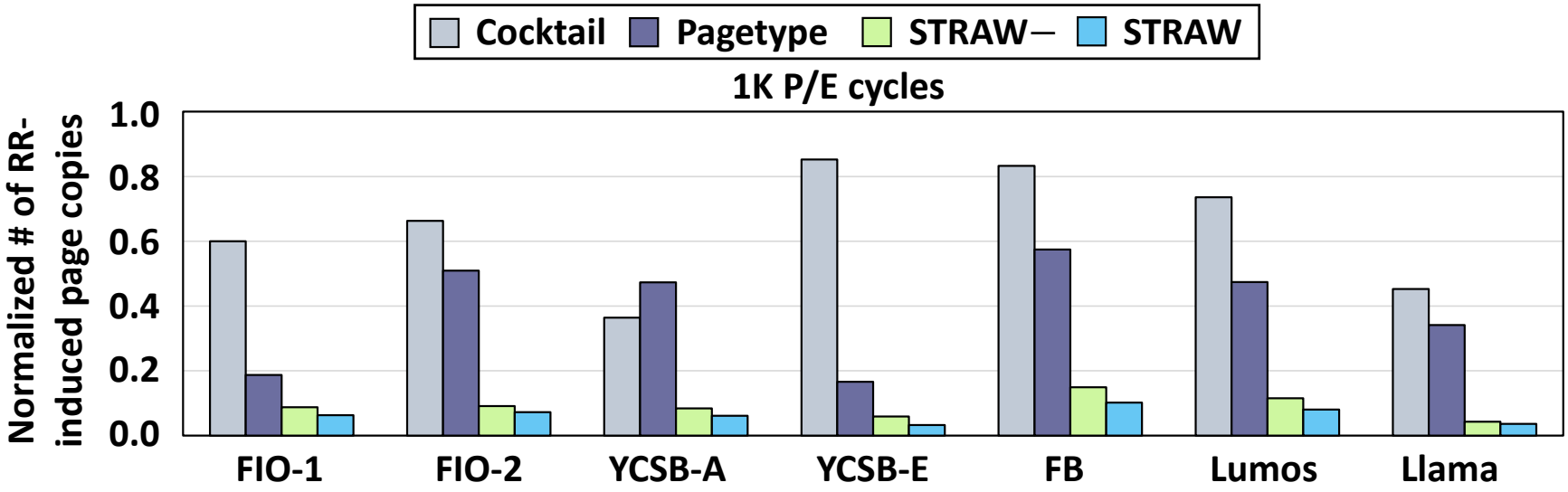
- Based on the characterization, we conclude:
  ① Logarithmic relationship -> We **limit the V_{pass} increase on invalid WLs to 10%**
  ② We **exclude lowering V_{pass} on adjacent WLs**—even when they are valid—because the cost dominates the benefit



Experiment Settings

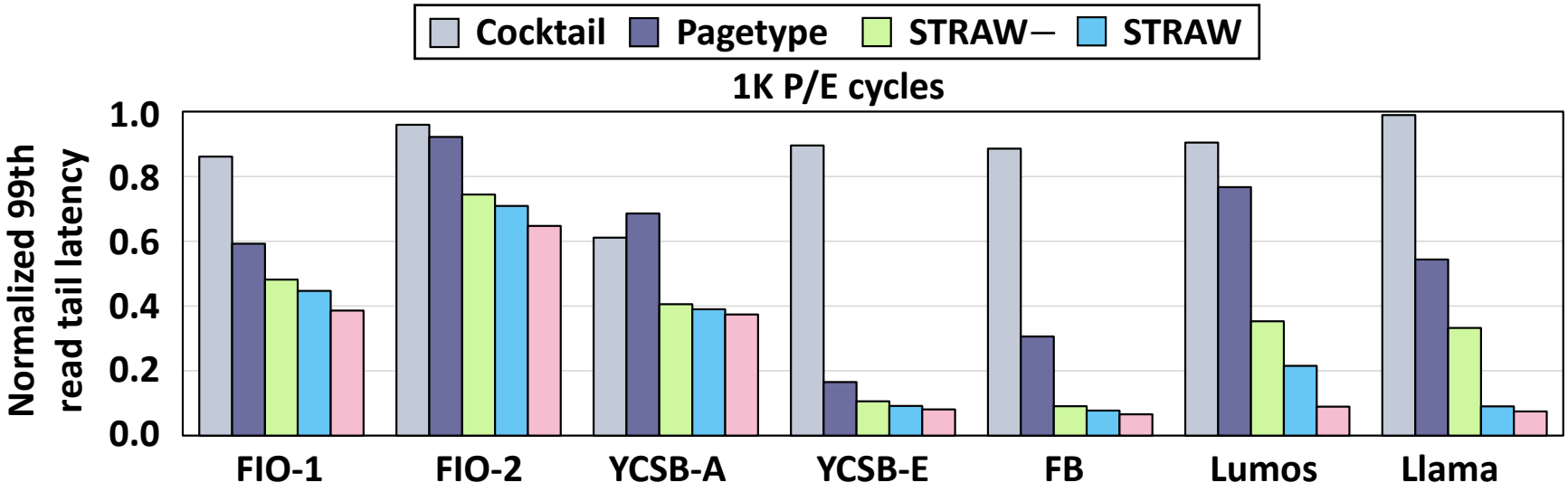
- **Emulator**: NVMeVirt
  - Extend NAND flash models w/ real-device characterization results
- **Seven workloads** from five benchmarks/applications
  - FIO
  - Lumos (graph processing)
  - YCSB
  - Llama.cpp (LLM inference)
  - Filebench
- **Comparison**
  - **BLOCK:** a naïve block-level RR
  - **PAGETYPE [IEEE TVLSI'23]:** page-type-aware RR
  - **Cocktail [IEEE TCAD'22]:** state-of-the-art RR optimization
  - **STRAW–**: STRAW without SR^2
  - **STRAW**

Results: RR-induced page copies



STRAW significantly reduce **# of RR-induced page copies** by **92.2%** at 1K P/E cycles

Results: 99th percentile read tail latencies



STRAW significantly reduce **p99.9 read tail latency** by **71.1%** at 1K P/E cycles

Conclusion

- Increasing NAND density degrades reliability, substantially increasing reliability-management cost

- We propose two techniques for smarter reliability management:
  - **RiF** Integrates near-data processing with read-retry to reduce retry-induced channel bandwidth waste
  - **STRAW** Improves read-reclaim efficiency by revisiting modern read-disturb characteristics

- Future work: extend RiF's error-prediction module to various SSD internal tasks (e.g., read reclaim, wear-leveling)