Reducing Solid-State Drive Read Latency by Optimizing Read-Retry

Jisung Park[1], Myungsuk Kim[2], Myoungjun Chun[2],
Lois Orosa[1], Jihong Kim[2], and Onur Mutlu[1]

[1] SAFARI ETH zürich

[2] VERI LUX MEA TAS

ASPLOS 2021 (Session 17: Solid State Drives)

Executive Summary

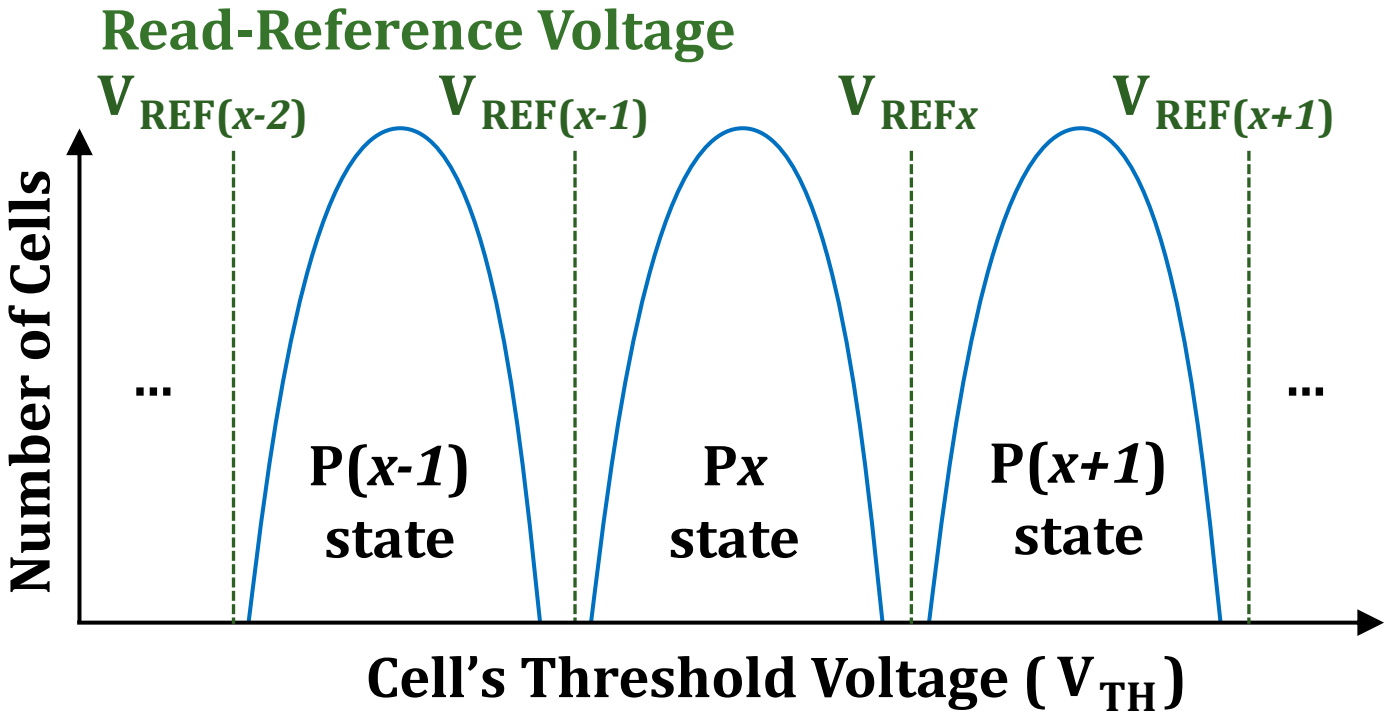
- **Problem:** Long read latency in modern SSDs due to read-retry
  - Frequently requires multiple retry steps to read an erroneous page
- **Goal:** Reduce the latency of each read-retry operation
- **Key Ideas:**
  - **Pipelined Read-Retry (PR^2):** Concurrently perform consecutive retry steps using the CACHE READ command
  - **Adaptive Read-Retry (AR^2):** Reduce read-timing parameters for every retry step by exploiting the reliability margin provided by strong ECC
  - Small implementation overhead and no changes to NAND flash chips
- **Evaluation Results:** Our proposal improves SSD response time by
  - Up to 51% (35% on average) compared to a high-end SSD
  - Up to 32% (17% on average) compared to a state-of-the-art baseline

Talk Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR^2: Pipelined Read-Retry
- AR^2: Adaptive Read-Retry
- Evaluation Results

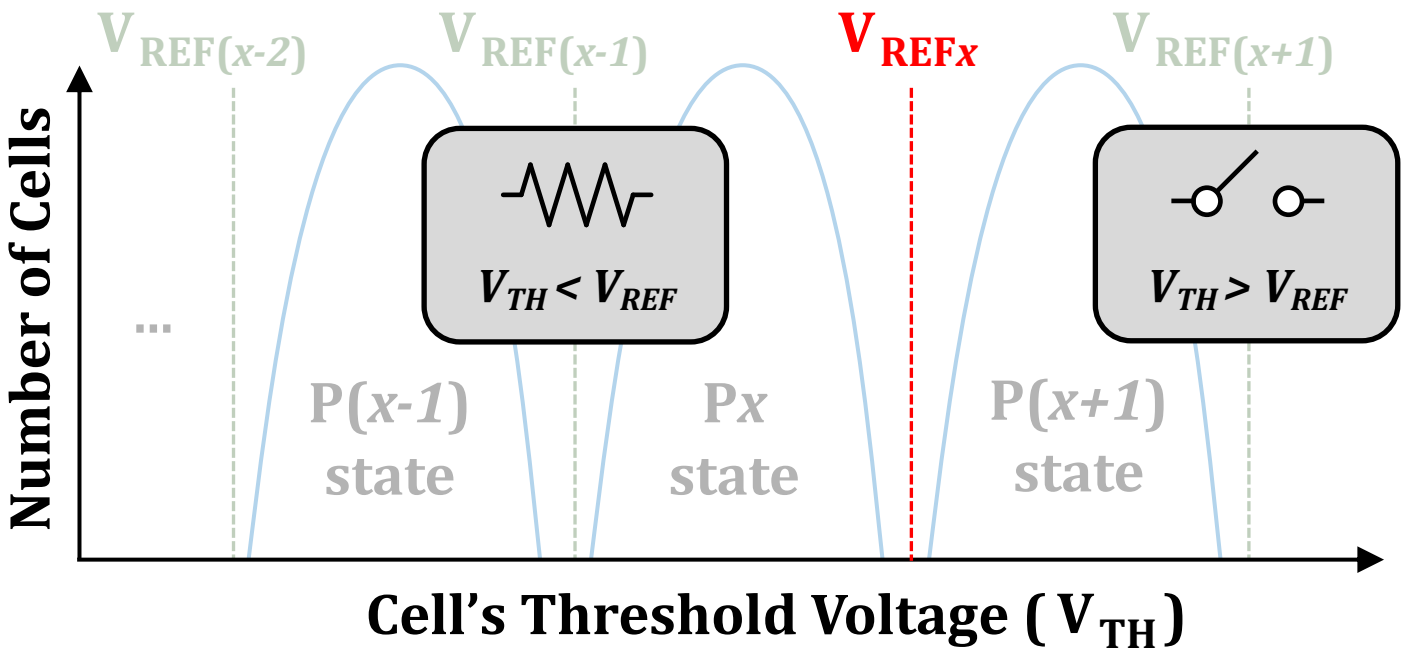
NAND Flash Basics

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



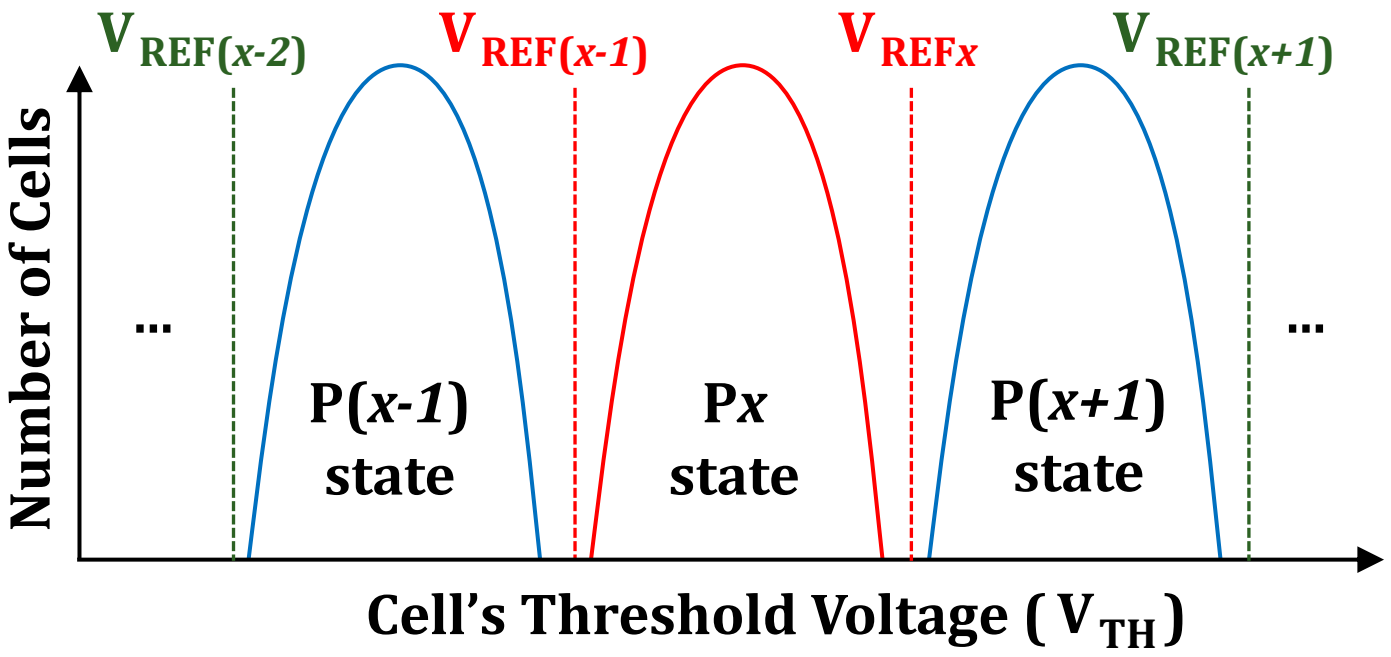
NAND Flash Basics

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



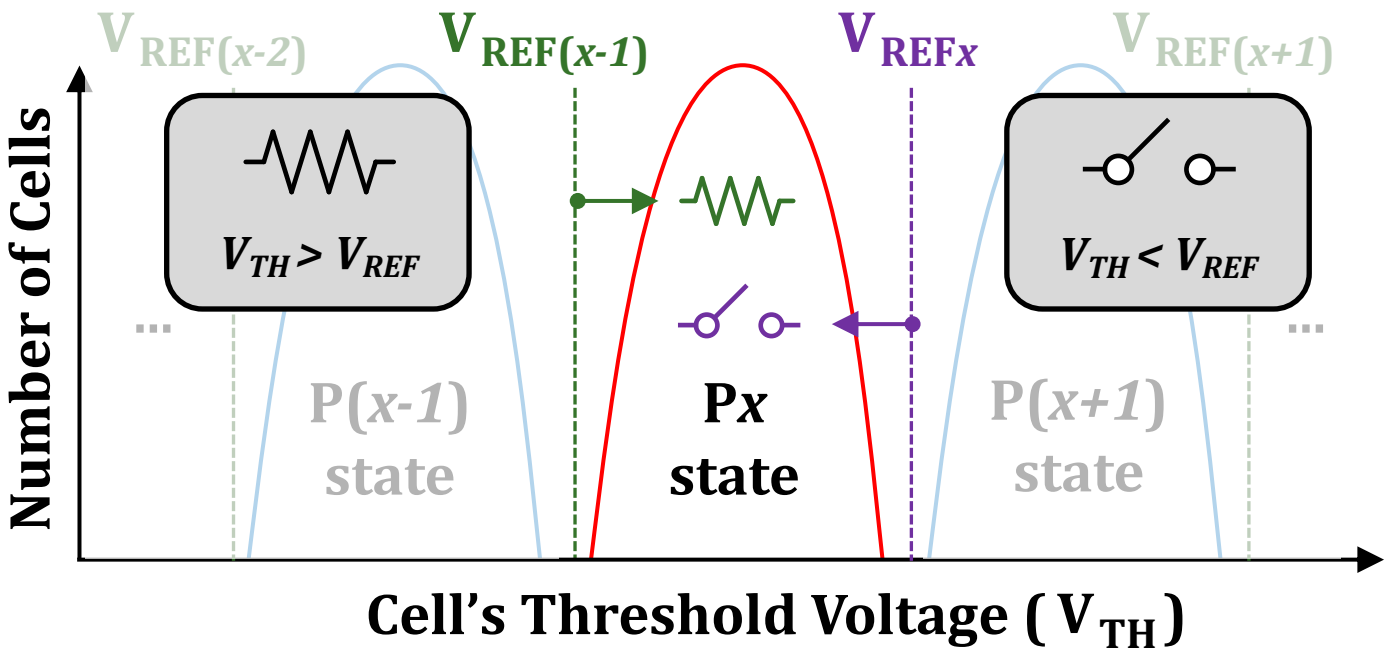
NAND Flash Basics

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



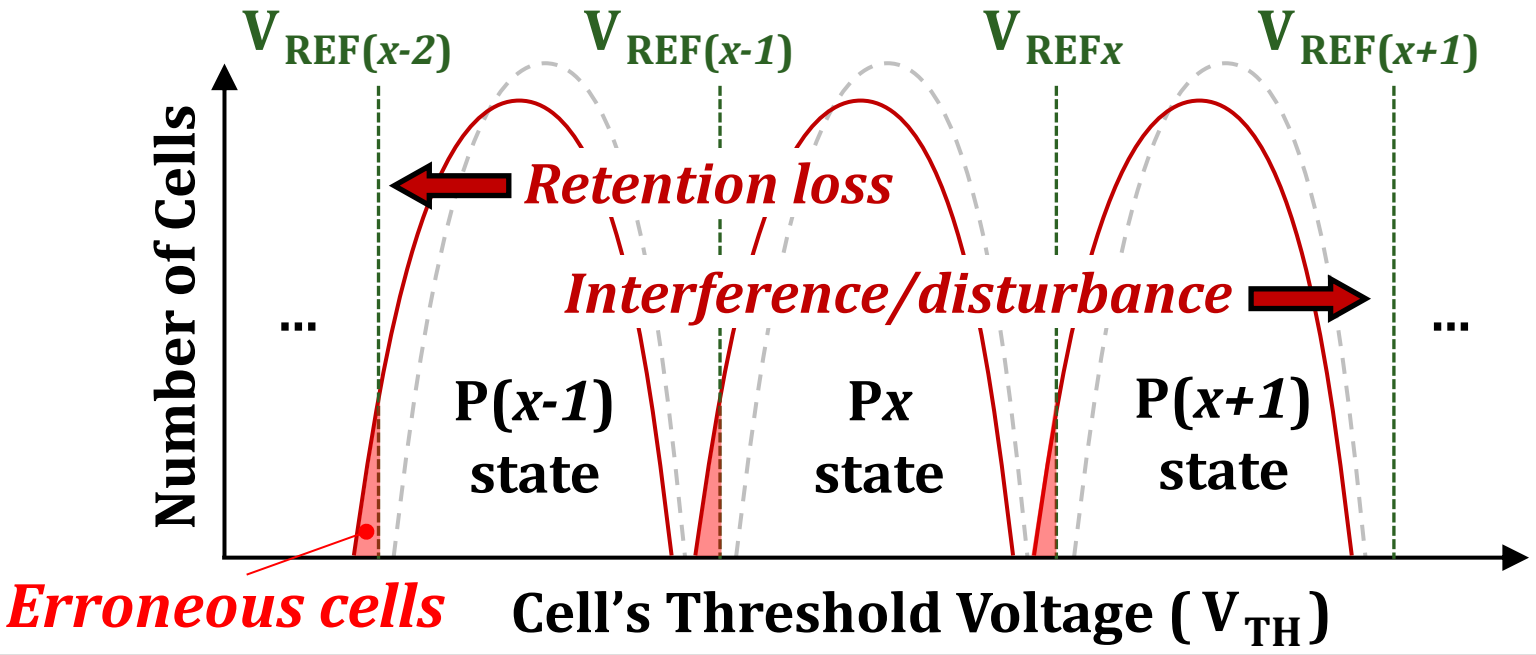
NAND Flash Basics

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



Errors in NAND Flash Memory

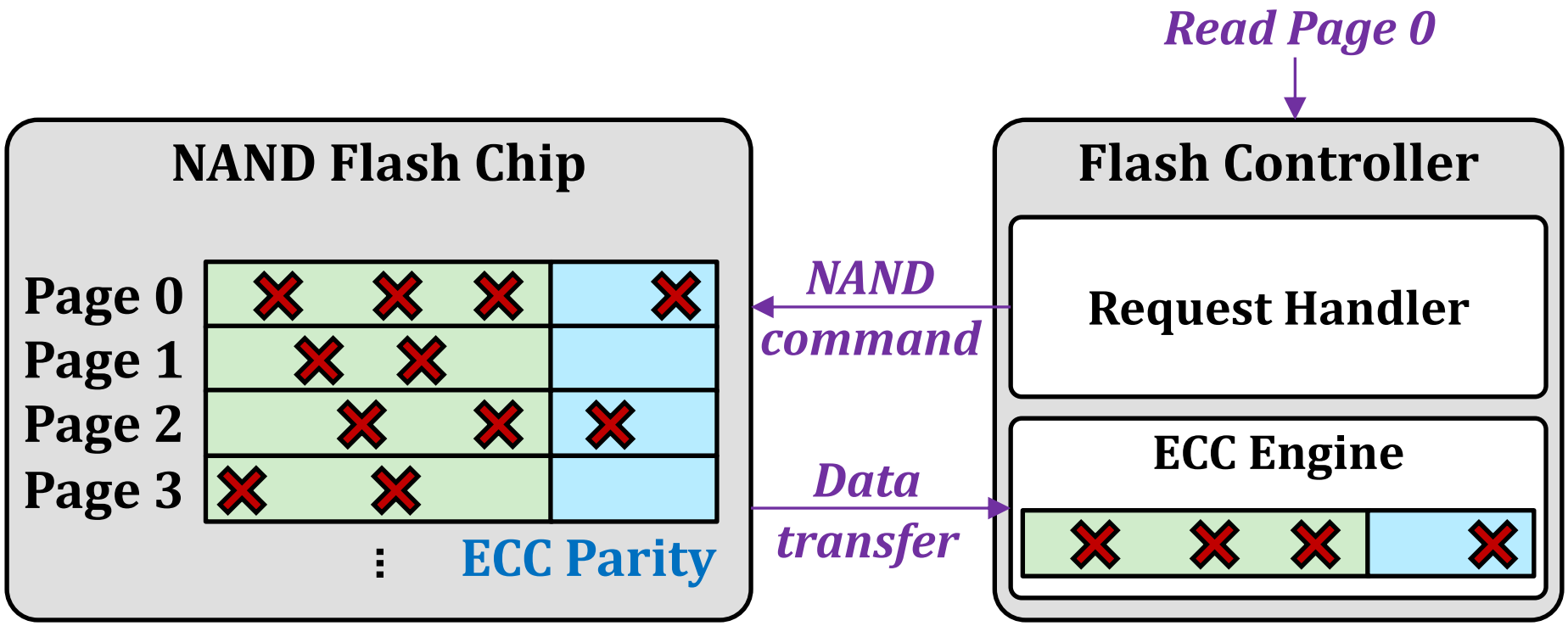
- Various sources shift and widen programmed V_{TH} states
  - Retention loss, program interference, read disturbance, etc.



Modern NAND flash memory is highly error-prone due to narrow V_{TH} margins

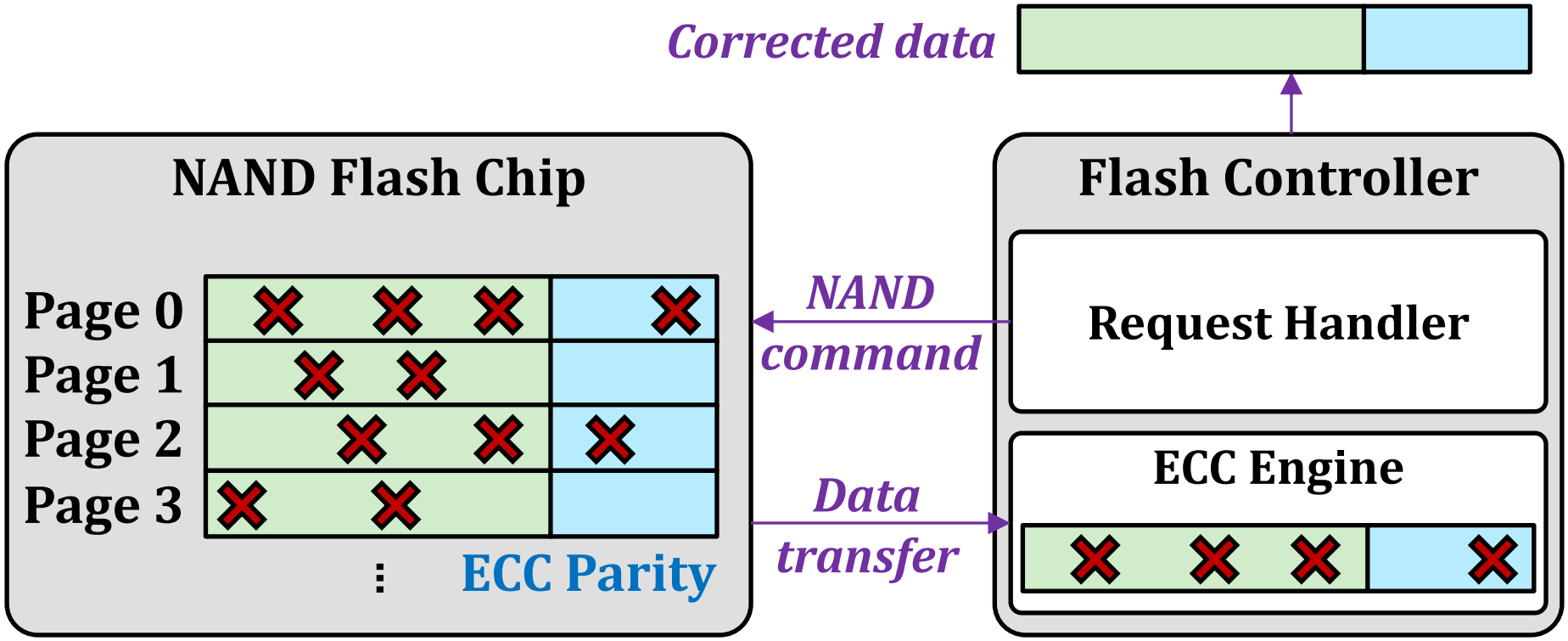
Error-Correcting Codes (ECC)

- Store redundant information (ECC parity) for error correction



Error-Correcting Codes (ECC)

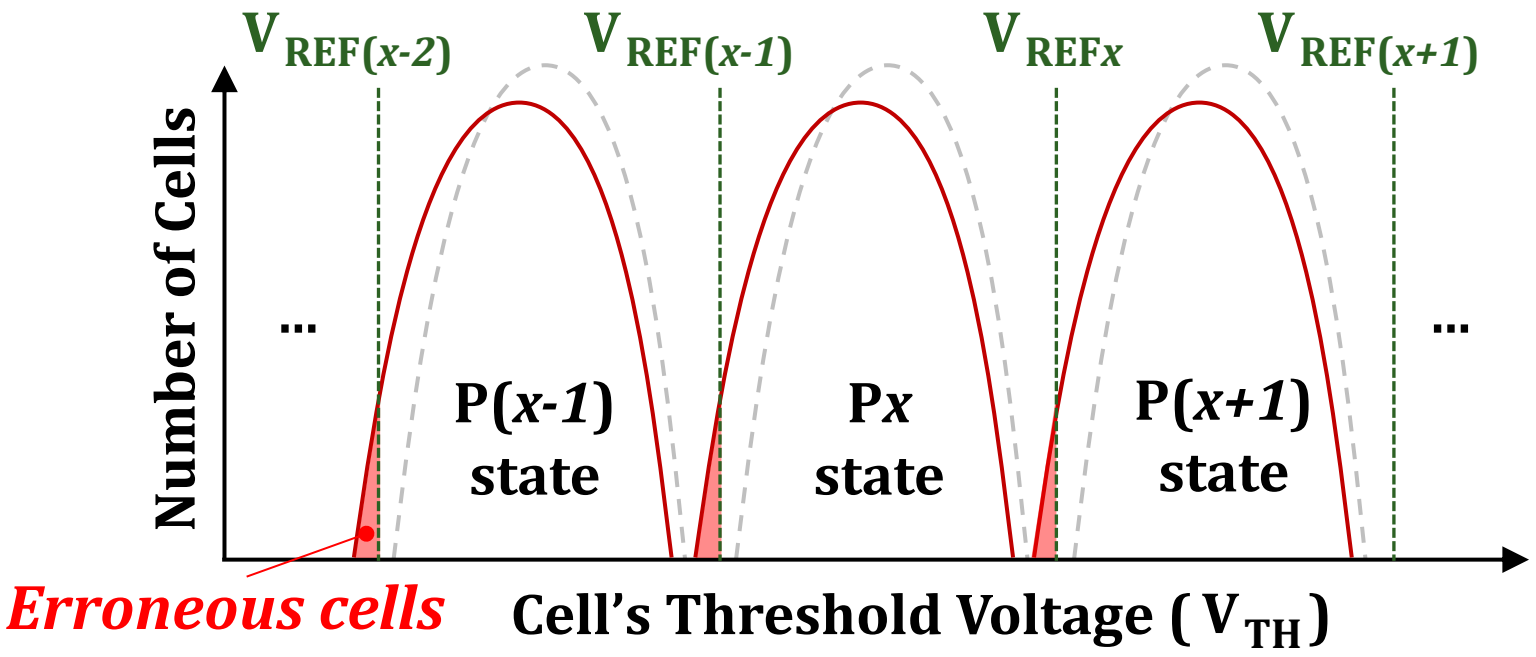
- Store redundant information (ECC parity) for error correction



# of raw bit errors > ECC correction capability
→ Uncorrectable errors in stored data

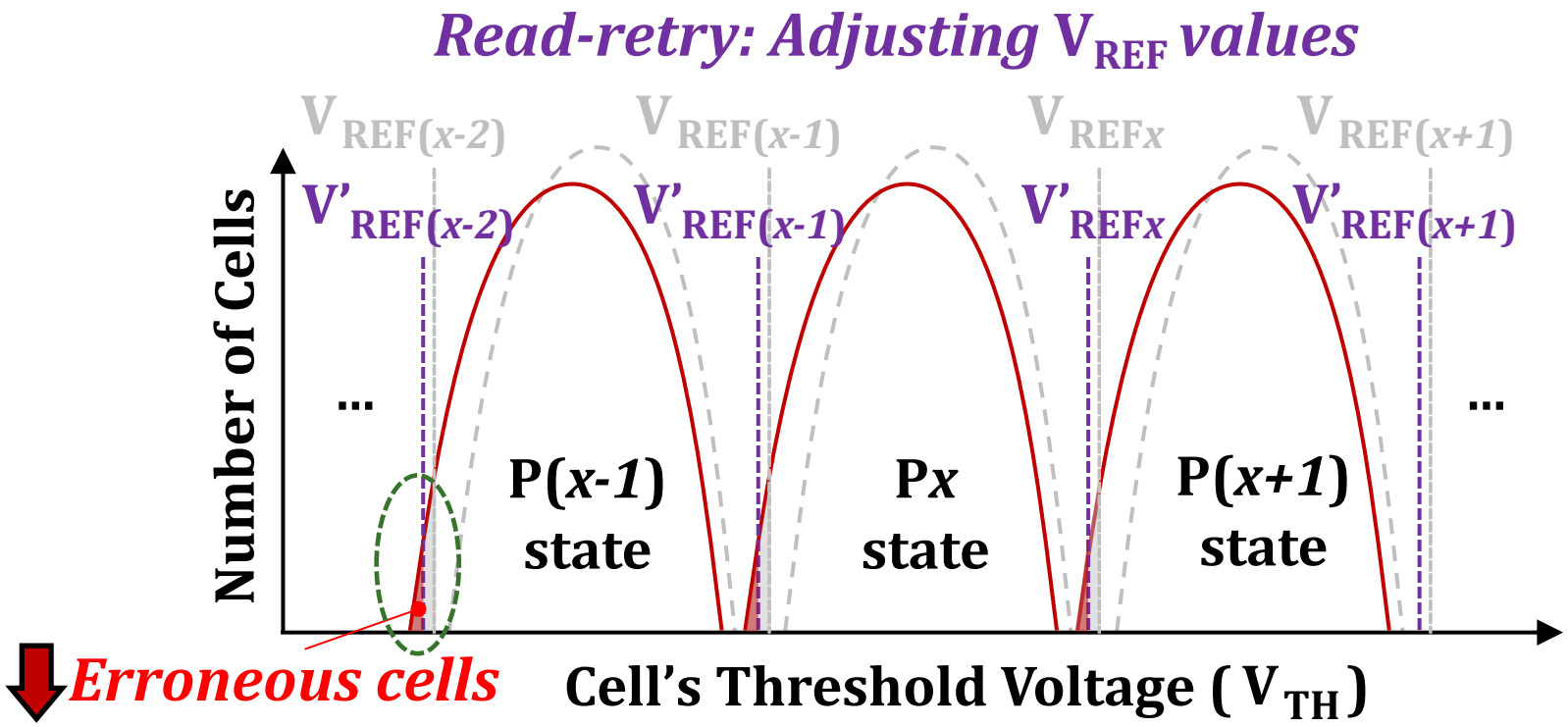
Read-Retry Operation

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



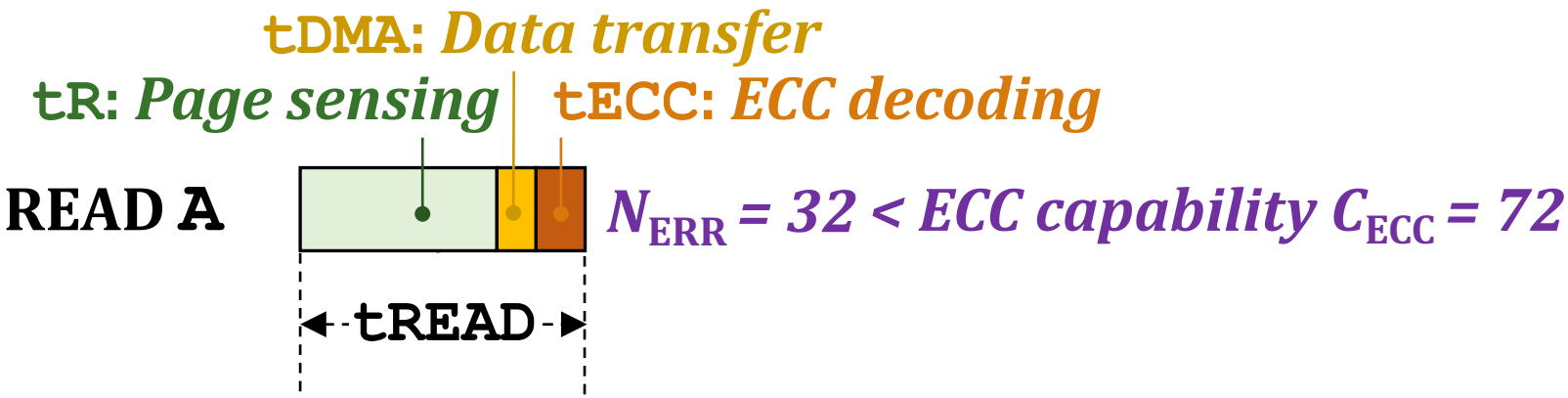
Read-Retry Operation

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

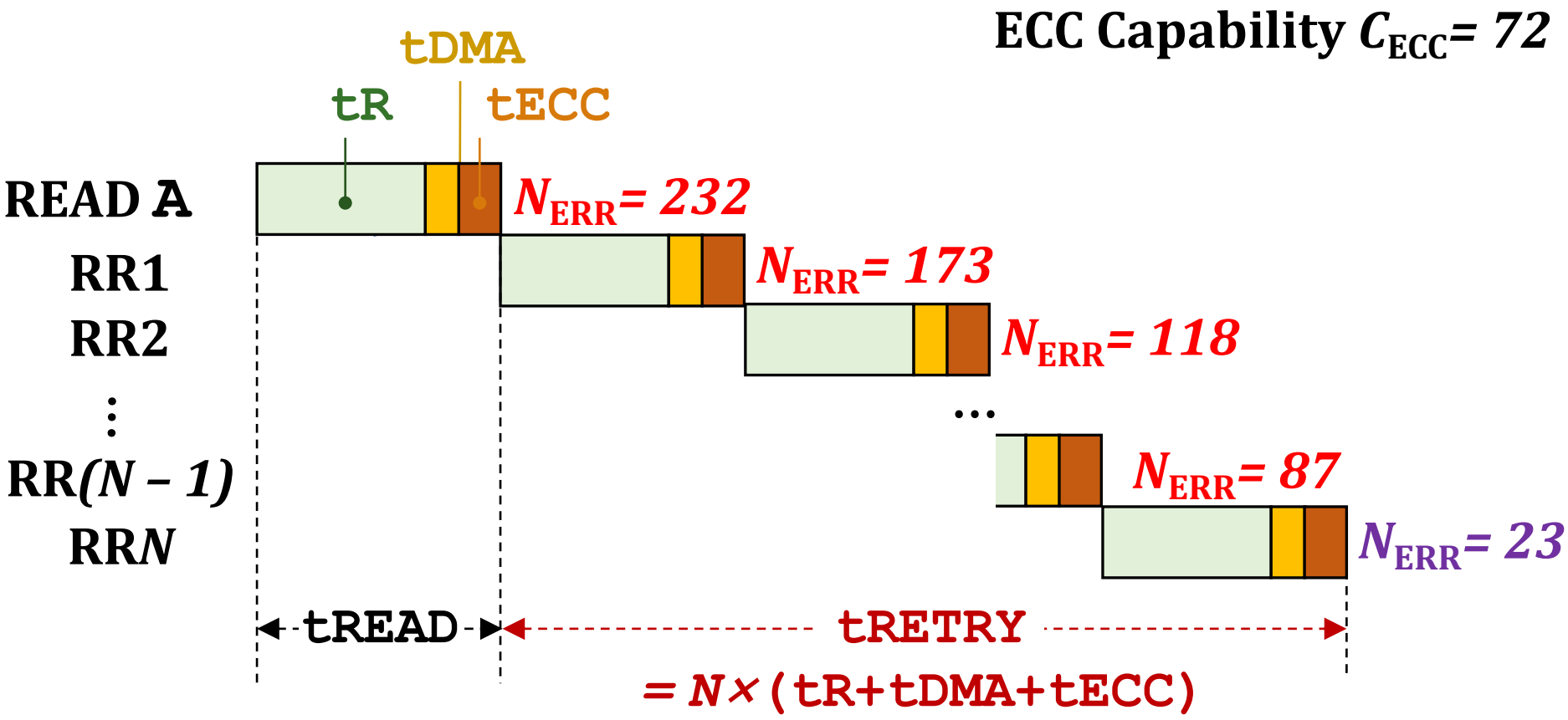


Read using properly-adjusted V_{REF} values
→ Decreases # of raw bit errors

Read-Retry: Performance Overhead



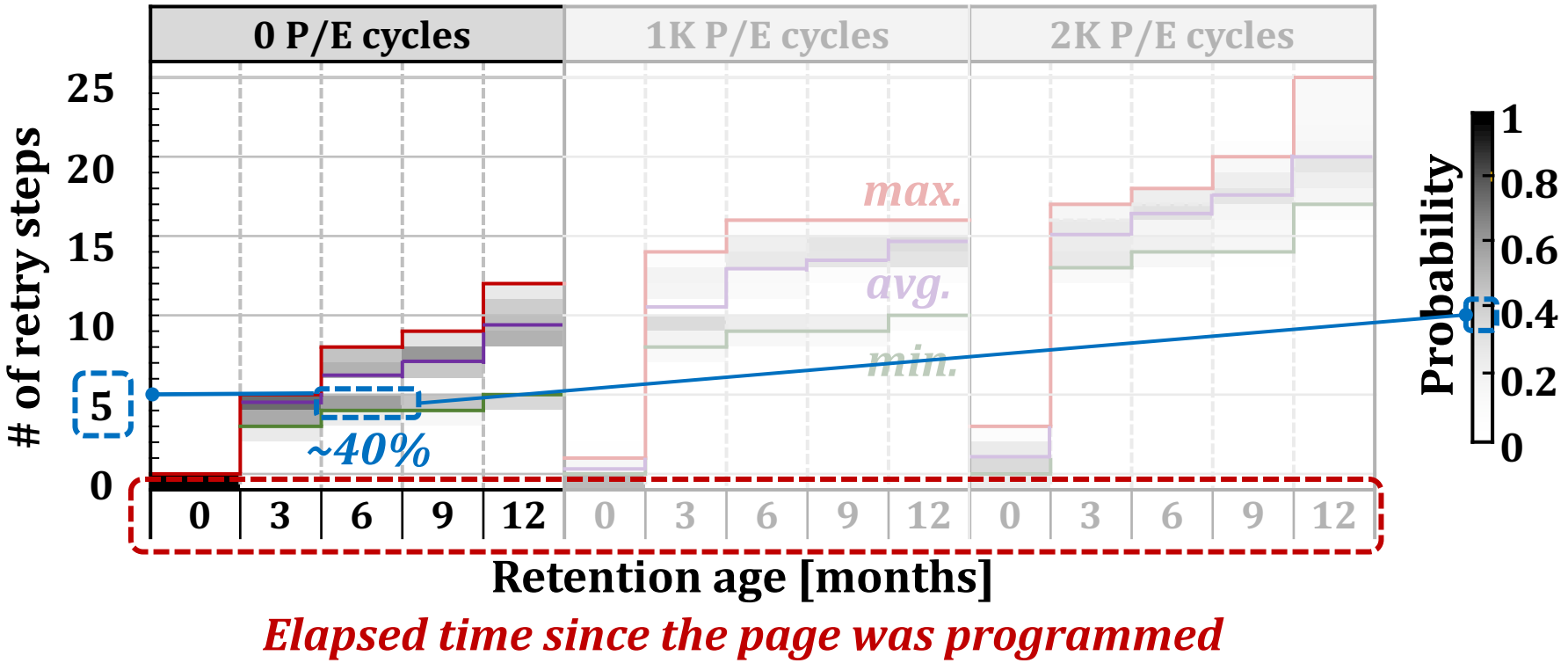
Read-Retry: Performance Overhead



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

Read-Retry in Modern SSDs: Experimental Data

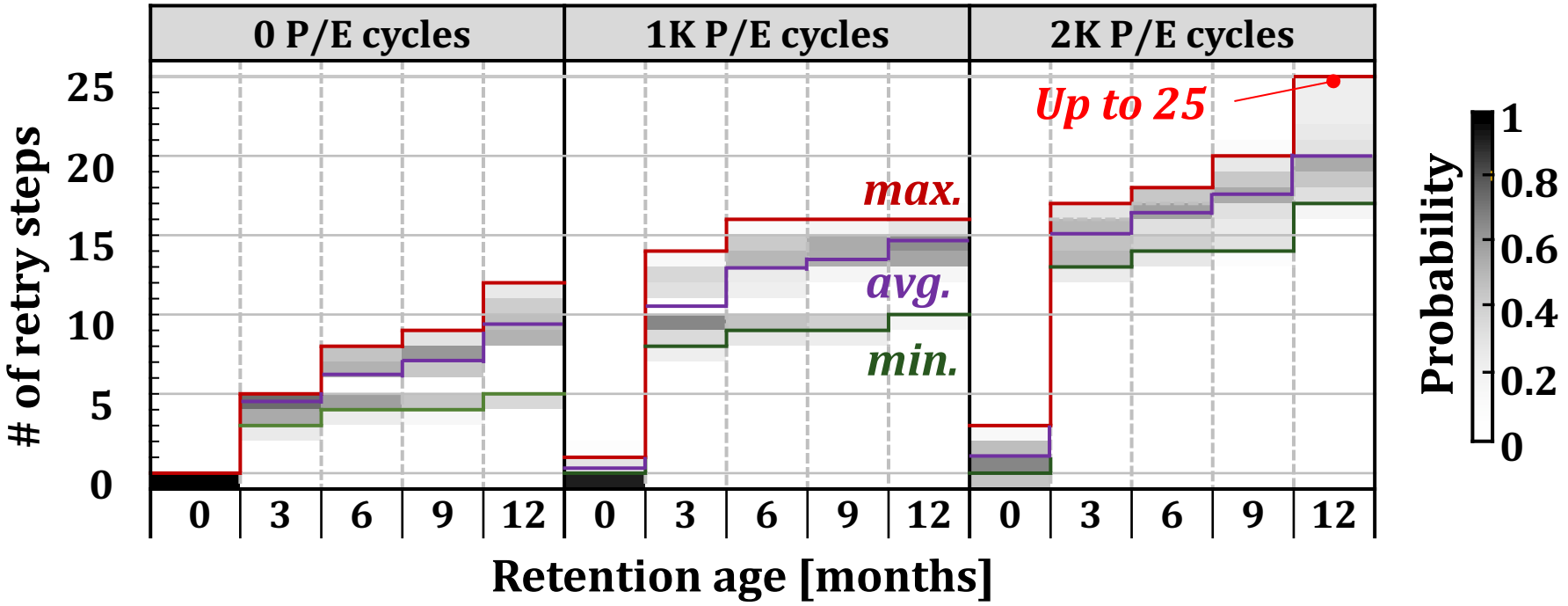
- Characterization of 160 real 3D TLC NAND flash chips
  - ECC correction capability: 72 bits per 1-KiB data



Elapsed time since the page was programmed

Read-Retry in Modern SSDs: Experimental Data

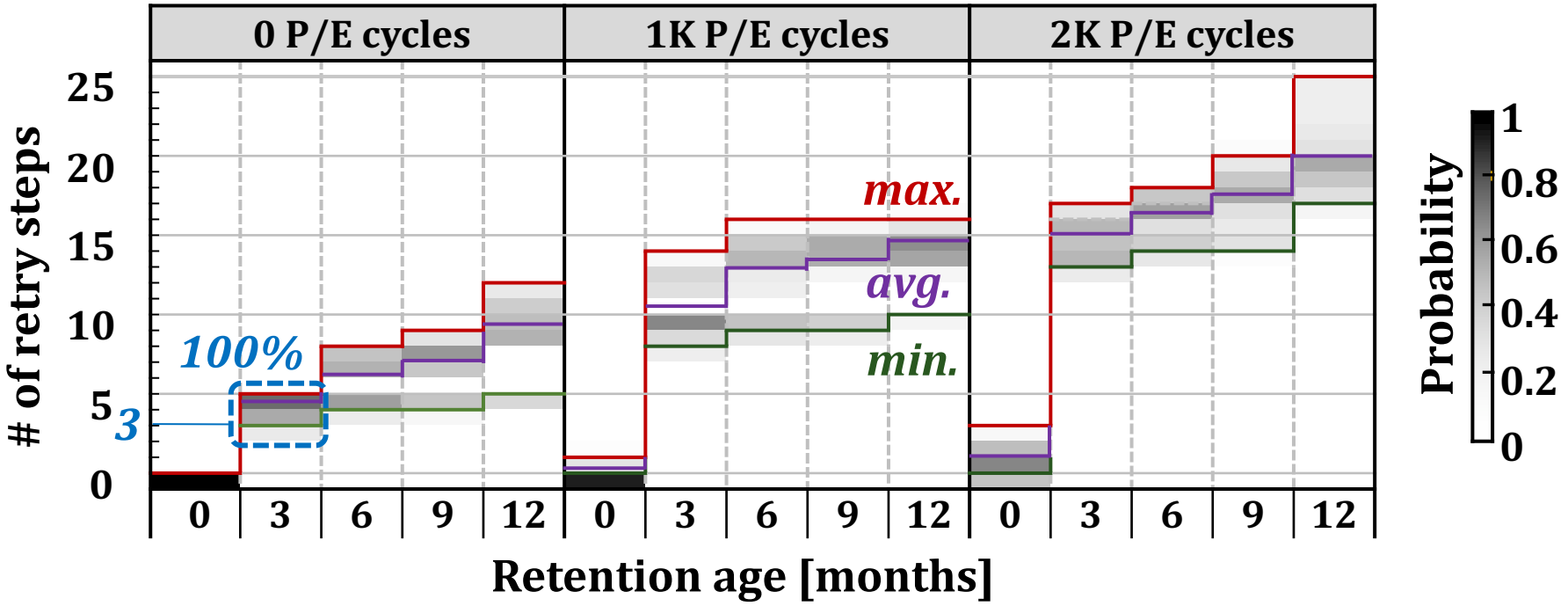
- Characterization of 160 real 3D TLC NAND flash chips
  - ECC correction capability: 72 bits per 1-KiB data



High P/E cycles and long retention age
→ More retry steps per read

Read-Retry in Modern SSDs: Experimental Data

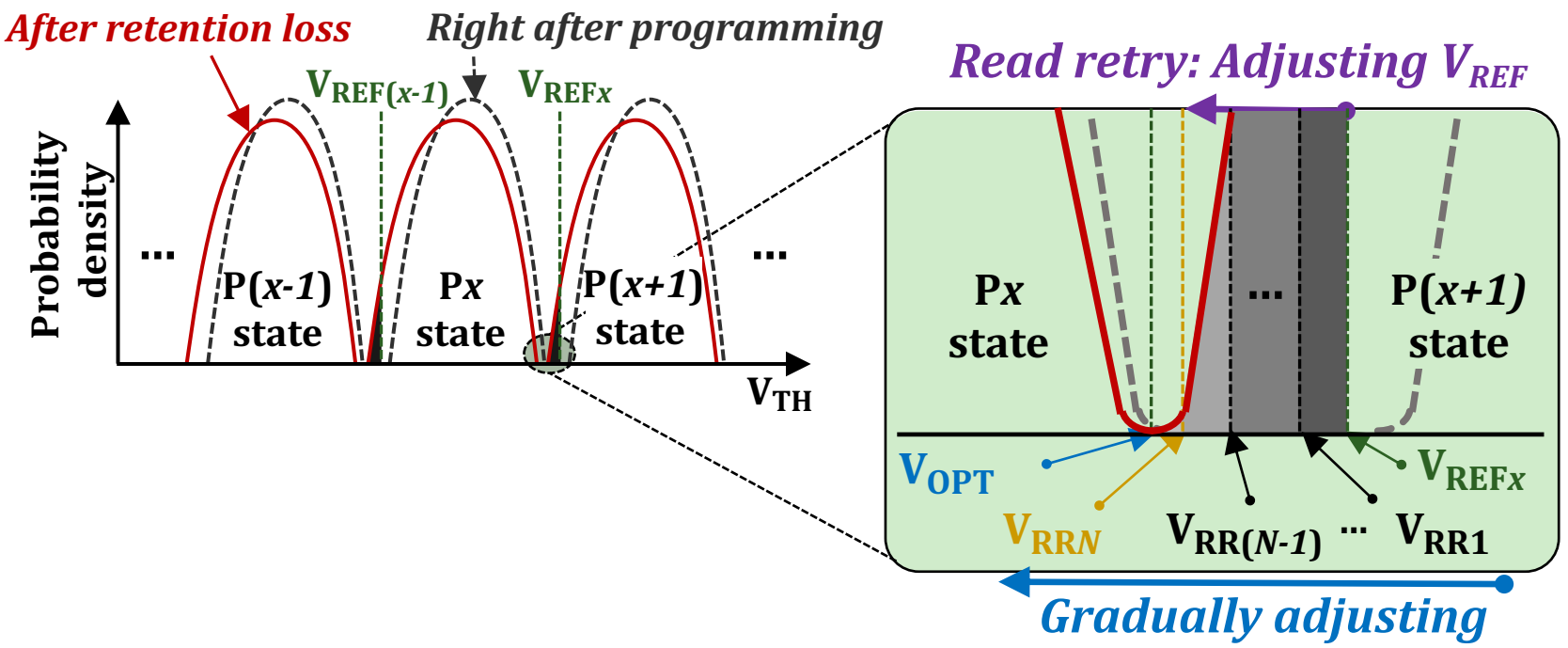
- Characterization of 160 real 3D TLC NAND flash chips
  - ECC correction capability: 72 bits per 1-KiB data



Many reads require multiple retry steps even under modest operating conditions

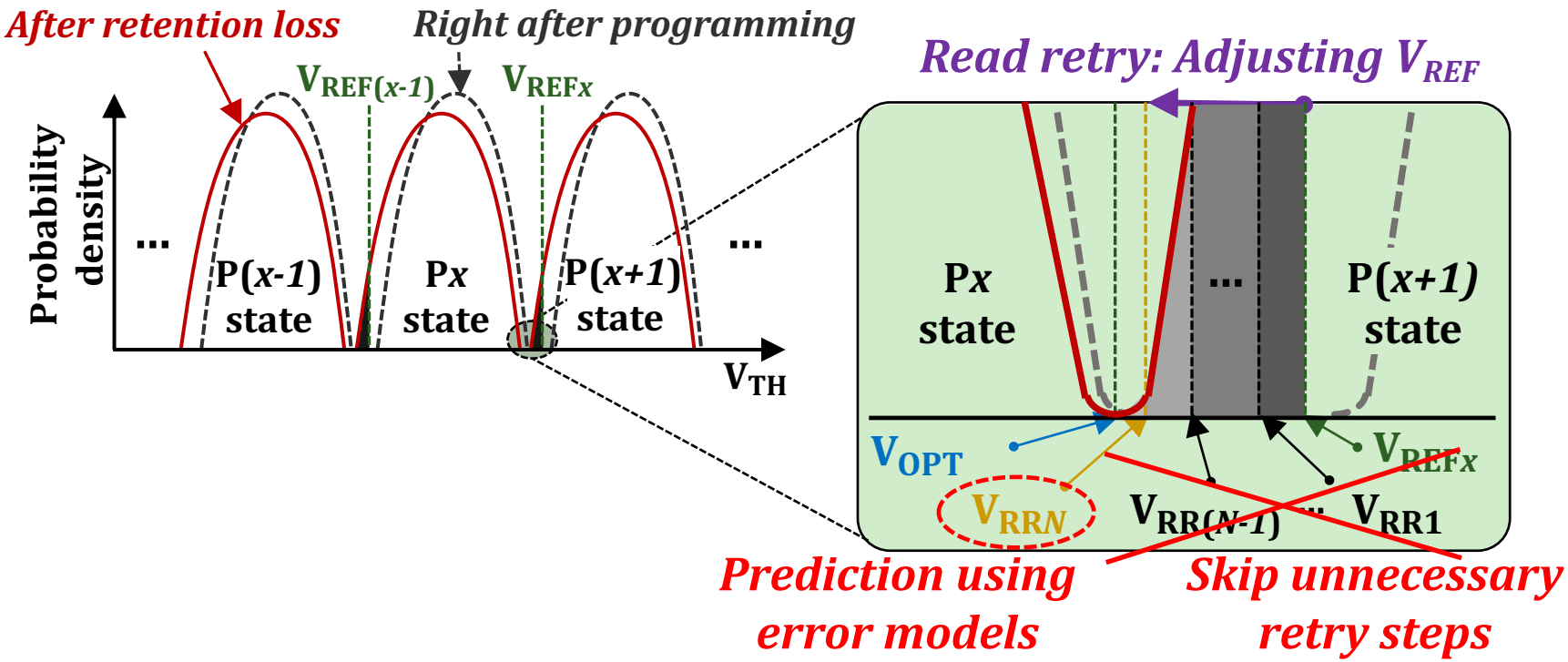
Existing Read-Retry Mitigation Schemes

- Try to reduce N_{RR} by predicting near-optimal V_{REF} values



Existing Read-Retry Mitigation Schemes

- Try to reduce N_{RR} by predicting near-optimal V_{REF} values



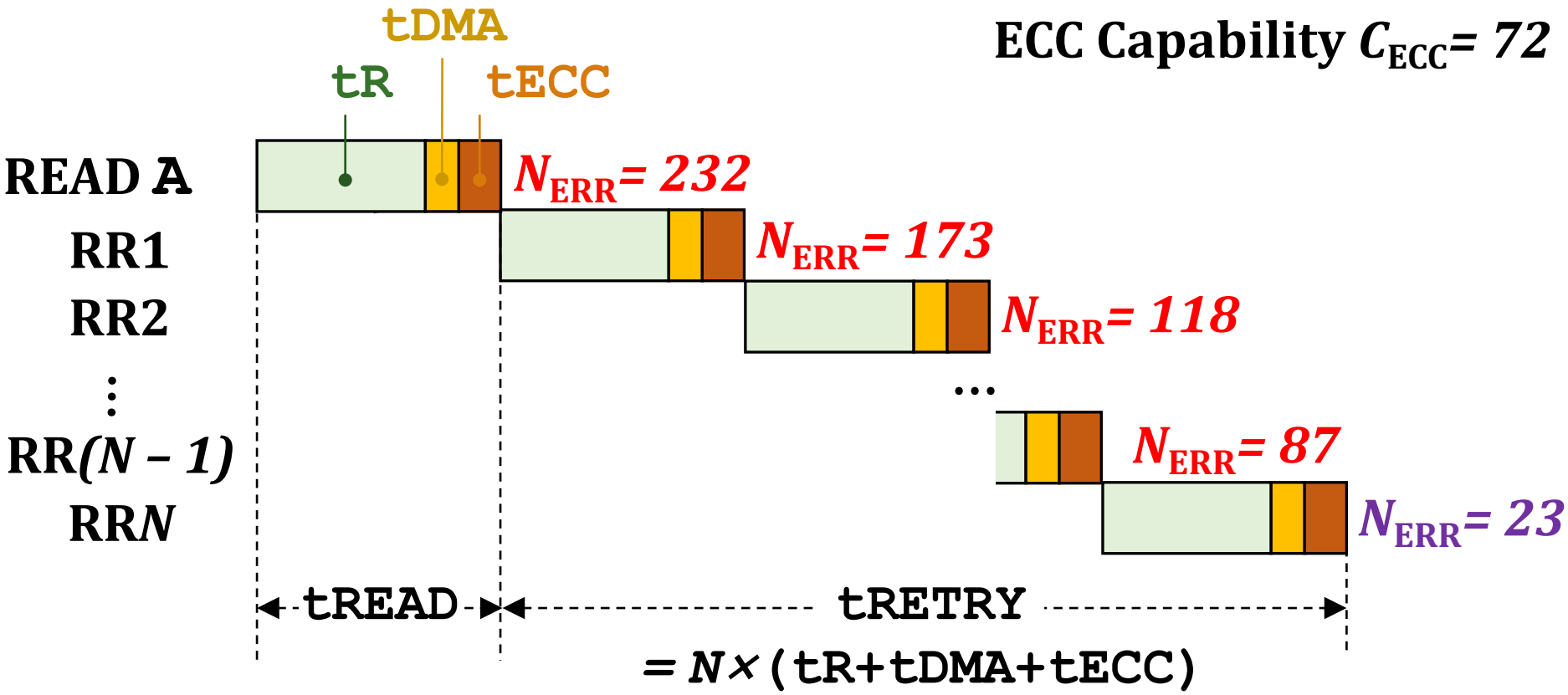
V_{TH} changes are fast and large in modern SSDs
→ Hard to eliminate read-retry

Talk Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR^2: Pipelined Read-Retry
- AR^2: Adaptive Read-Retry
- Evaluation Results

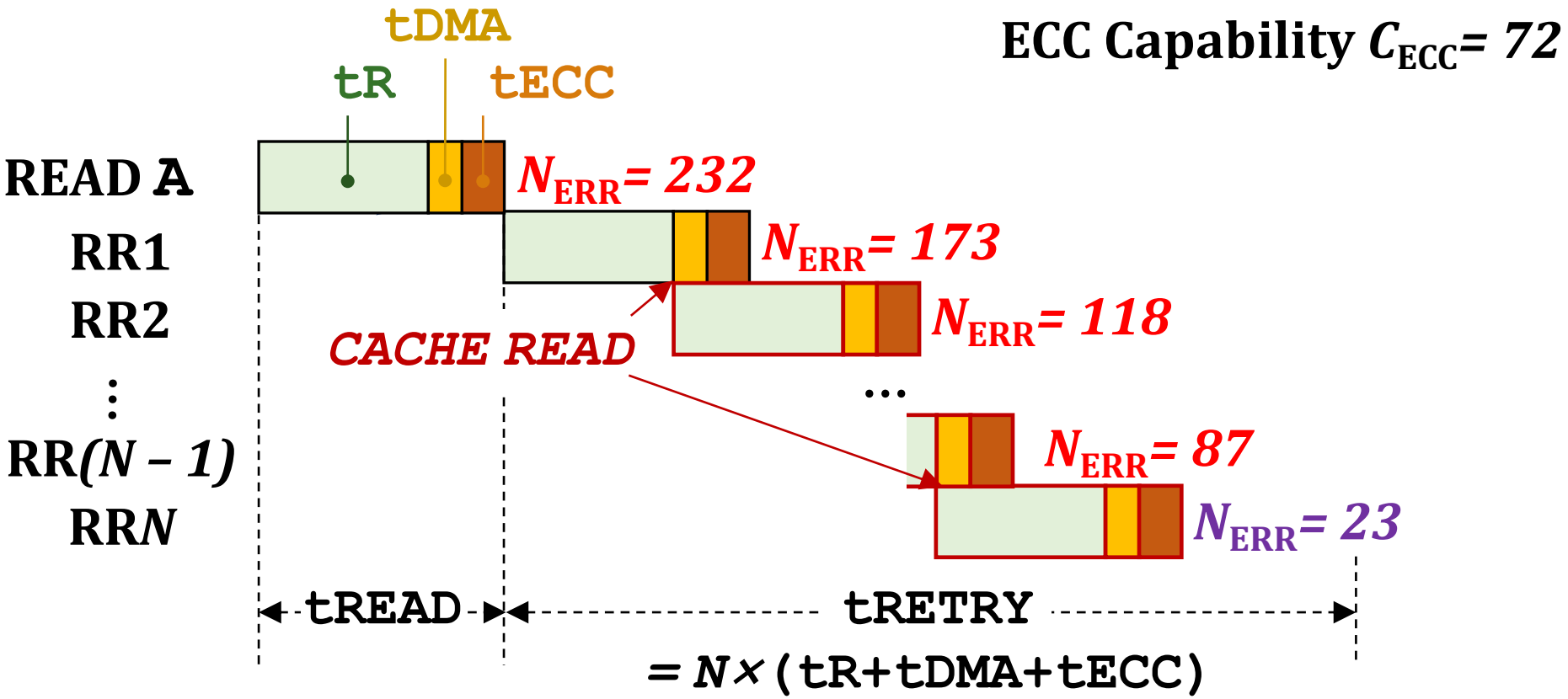
PR²: Pipelined Read-Retry

- **Key idea:** Concurrently perform consecutive retry steps



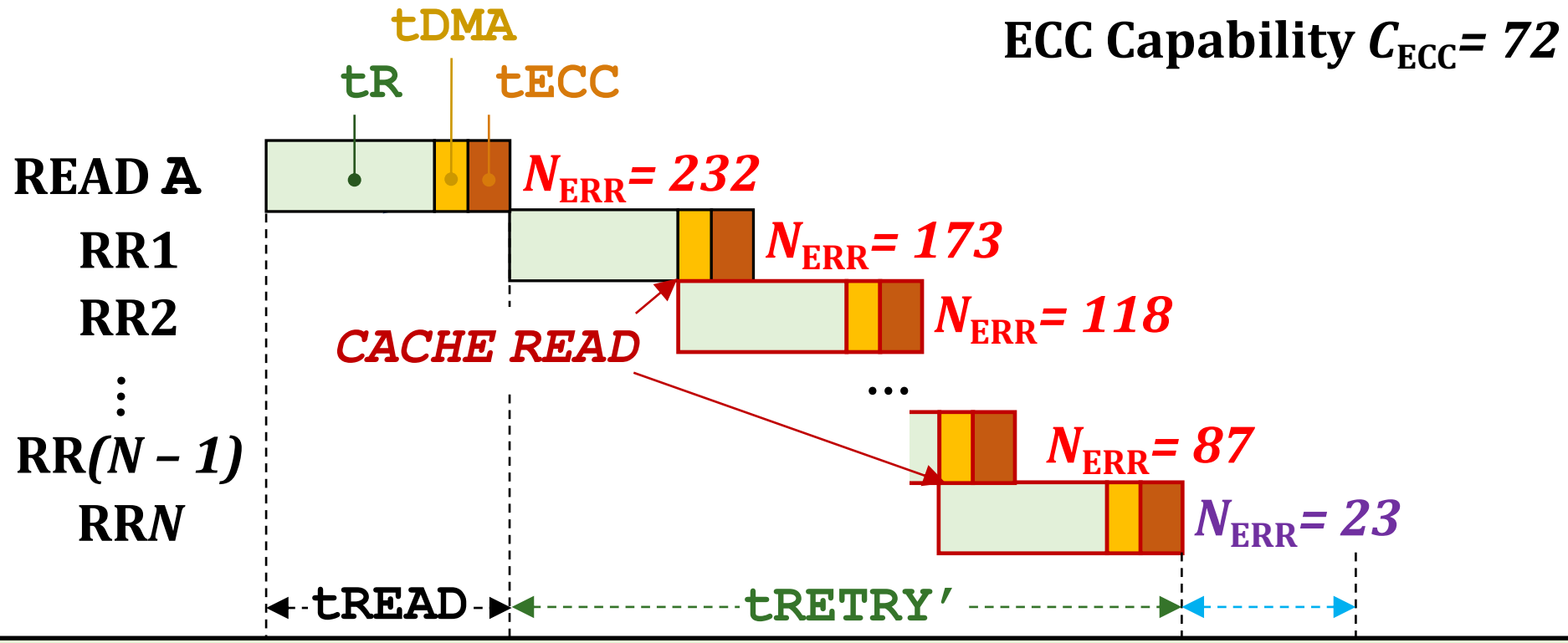
PR²: Pipelined Read-Retry

- **Key idea:** Concurrently perform consecutive retry steps



PR²: Pipelined Read-Retry

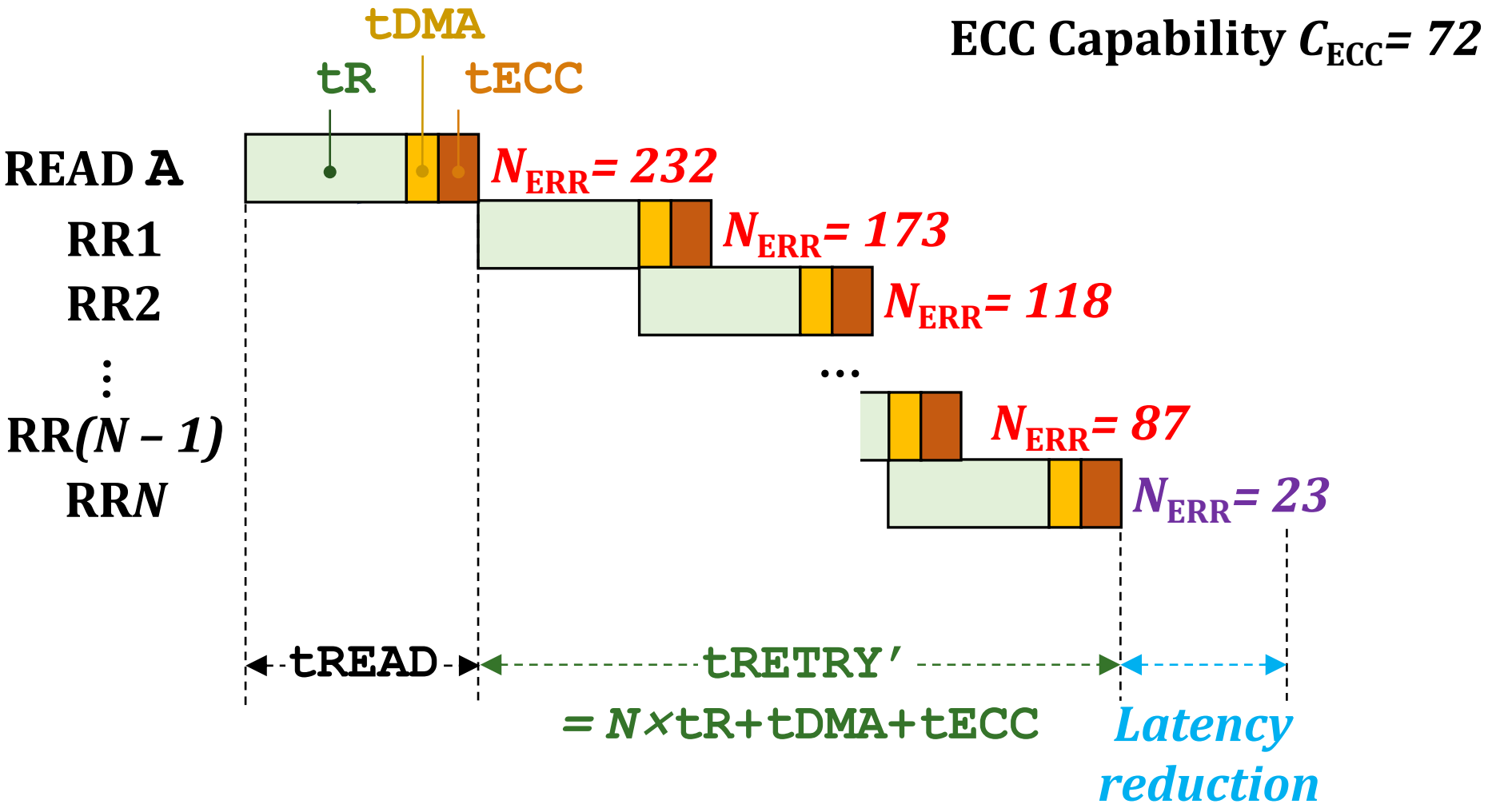
- **Key idea:** Concurrently perform consecutive retry steps



PR²: Removes tDMA & tECC (~30% of each retry step) from the critical path

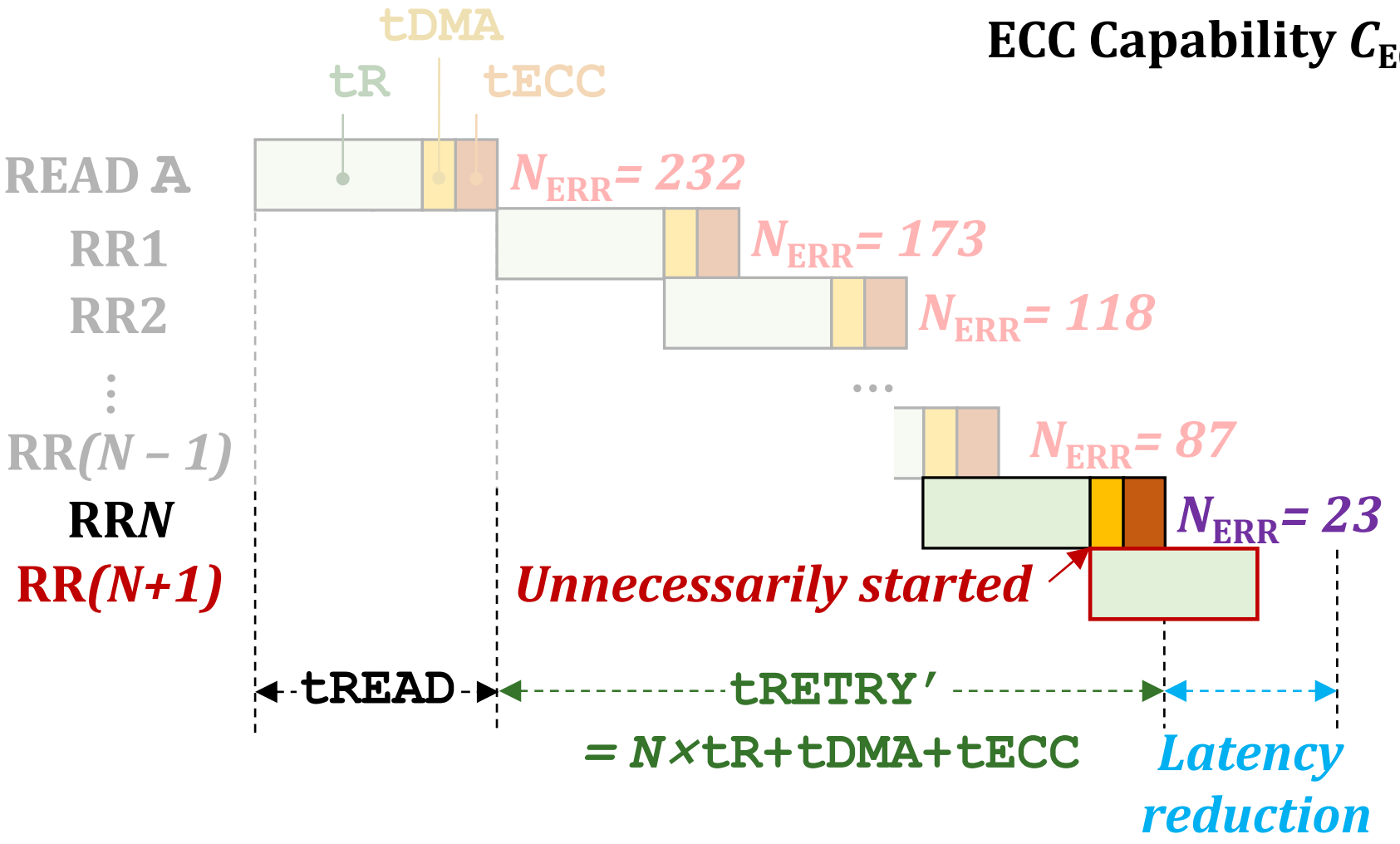
PR²: Pipelined Read-Retry

- **Key idea:** Concurrently perform consecutive retry steps



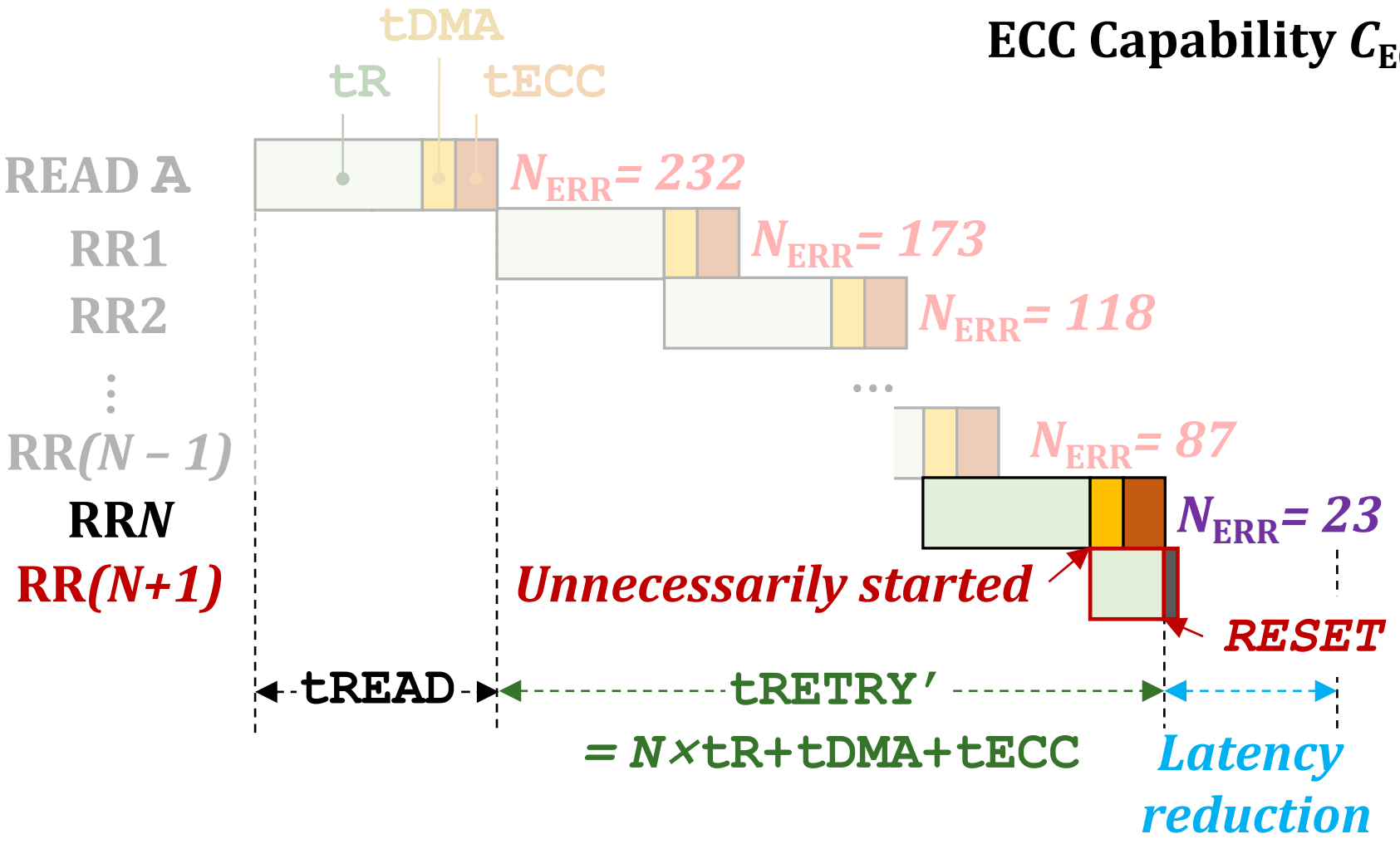
PR²: Pipelined Read-Retry

- **Key idea:** Concurrently perform consecutive retry steps



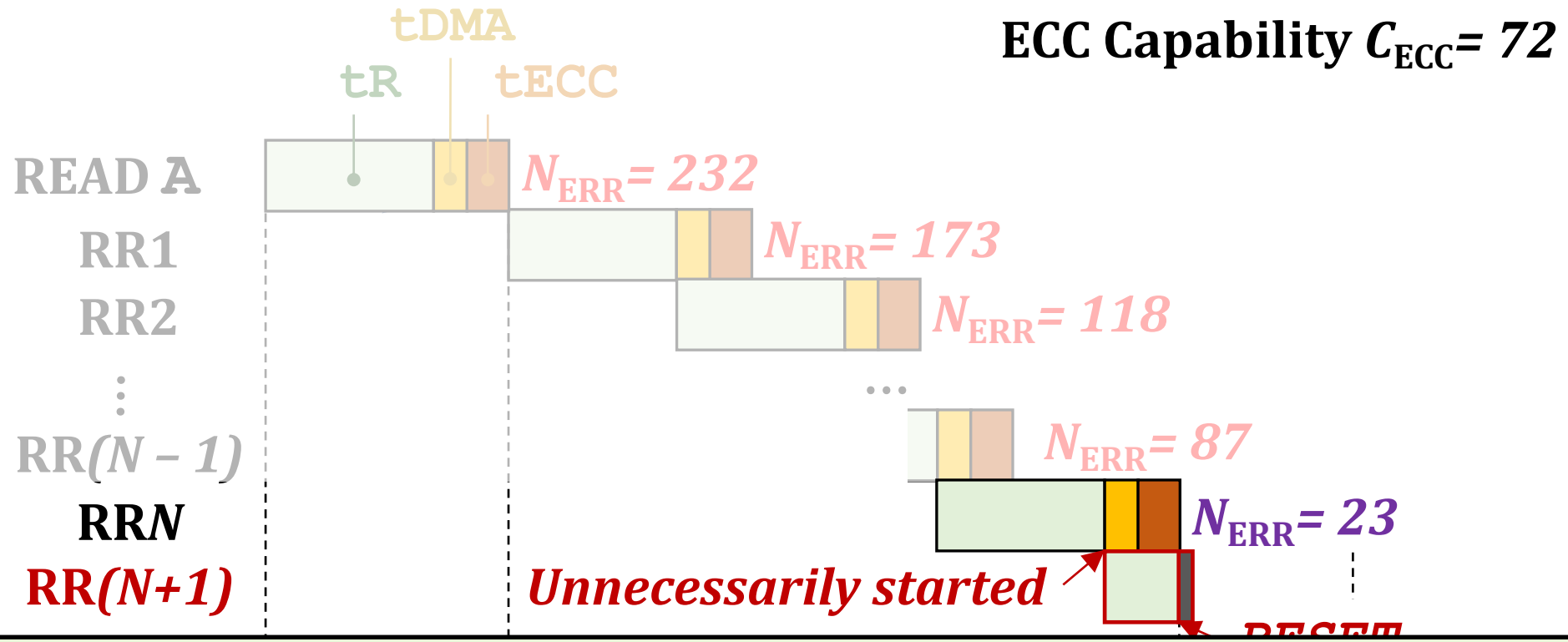
PR²: Pipelined Read-Retry

- **Key idea:** Concurrently perform consecutive retry steps



PR²: Pipelined Read-Retry

- **Key idea:** Concurrently perform consecutive retry steps

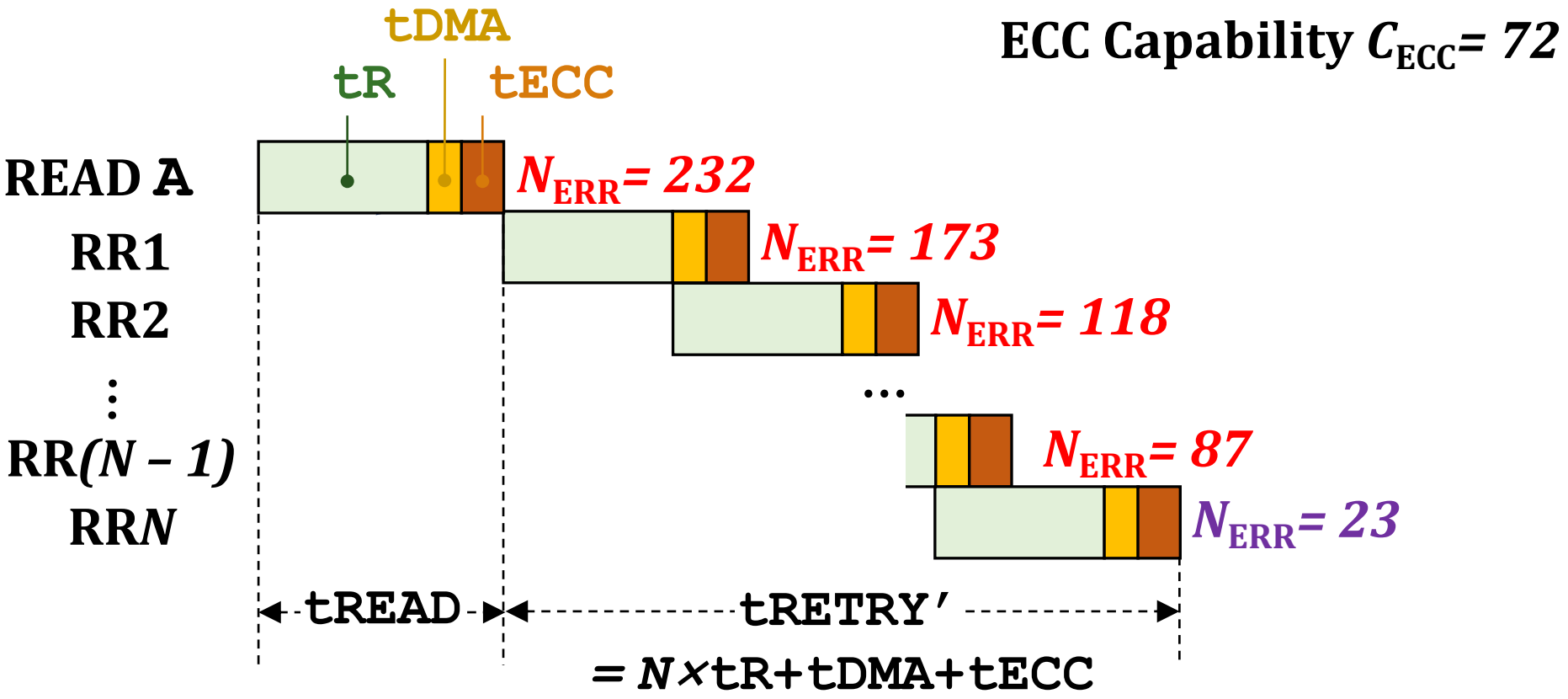


PR²: Large latency reduction (~30%)
w/ negligible performance penalty

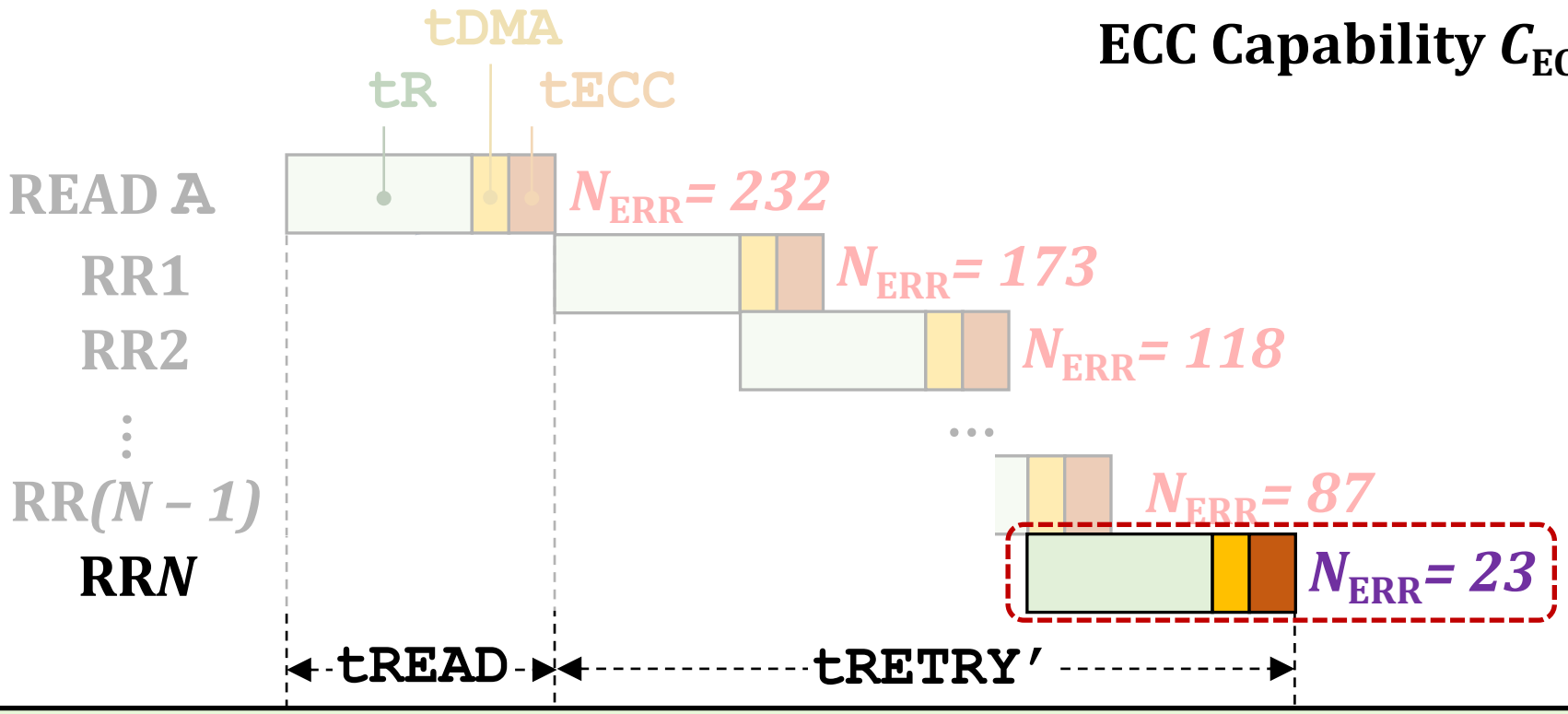
Talk Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR^2: Pipelined Read-Retry
- AR^2: Adaptive Read-Retry
- Evaluation Results

AR²: Adaptive Read-Retry



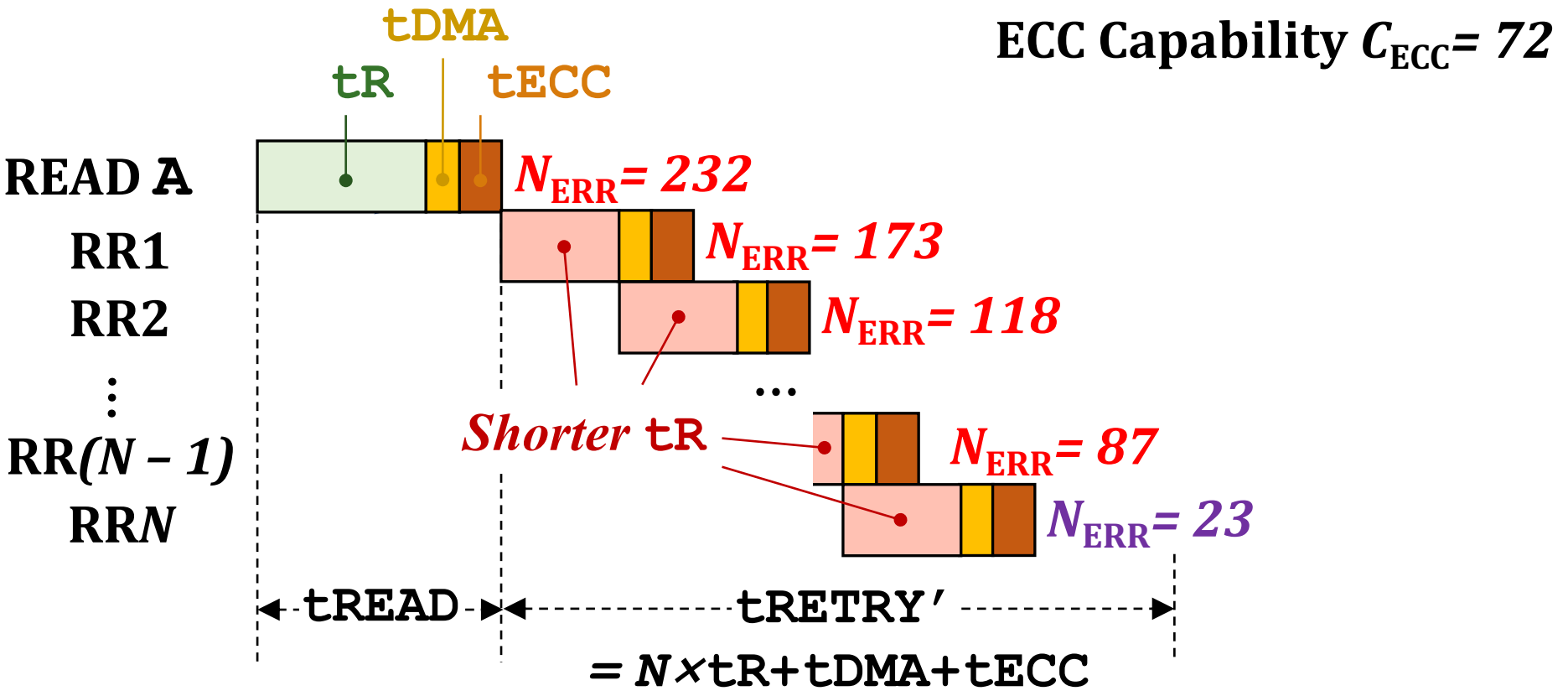
AR²: Adaptive Read-Retry



Observation: A positive ECC margin
in the final retry step when read-retry succeeds

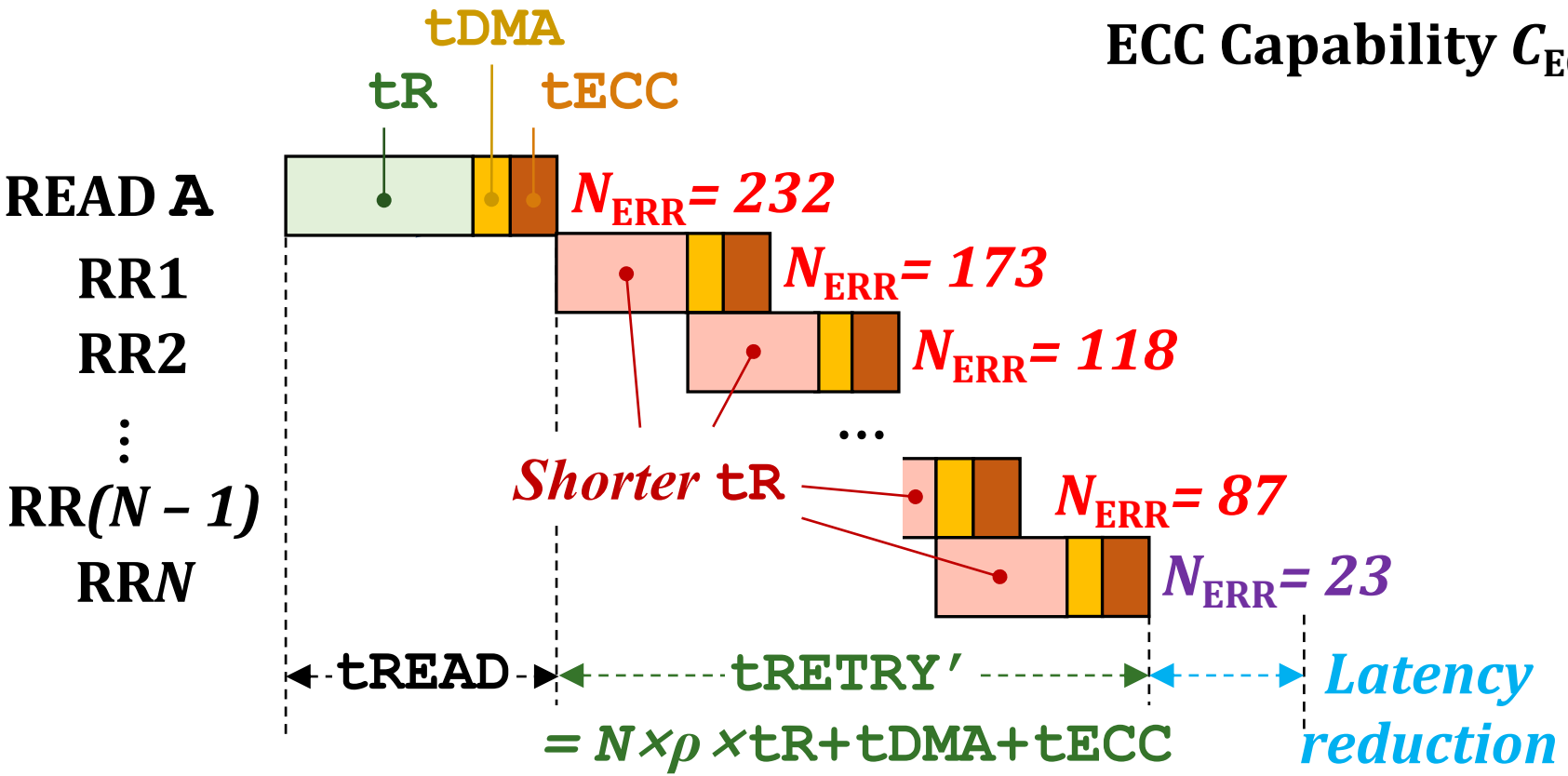
AR²: Adaptive Read-Retry

- **Key idea:** Reduce read-timing parameters for every retry step



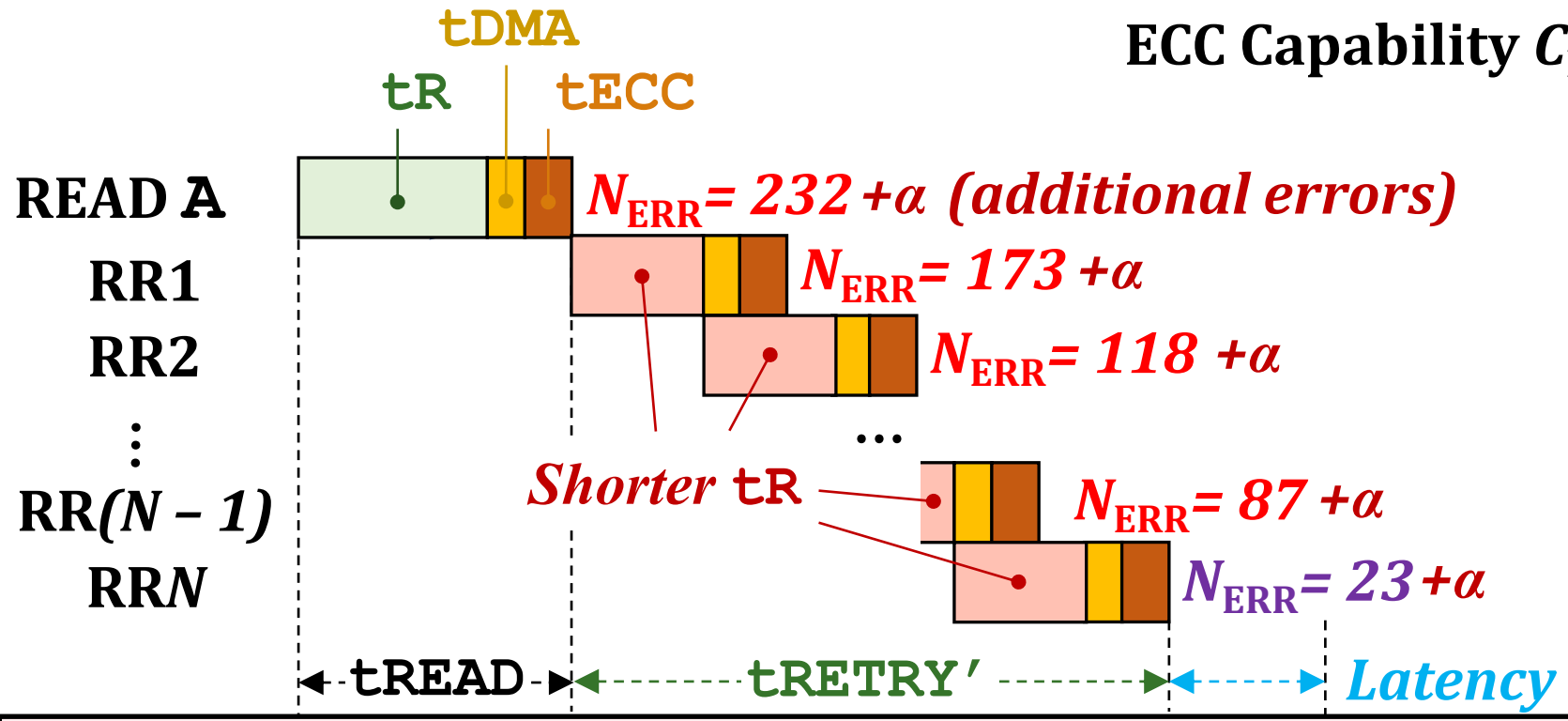
AR²: Adaptive Read-Retry

- **Key idea:** Reduce read-timing parameters for every retry step



AR²: Adaptive Read-Retry

- **Key idea:** Reduce read-timing parameters for every retry step



Needs to ensure that
# of additional errors < ECC margin

AR²: Necessary Conditions

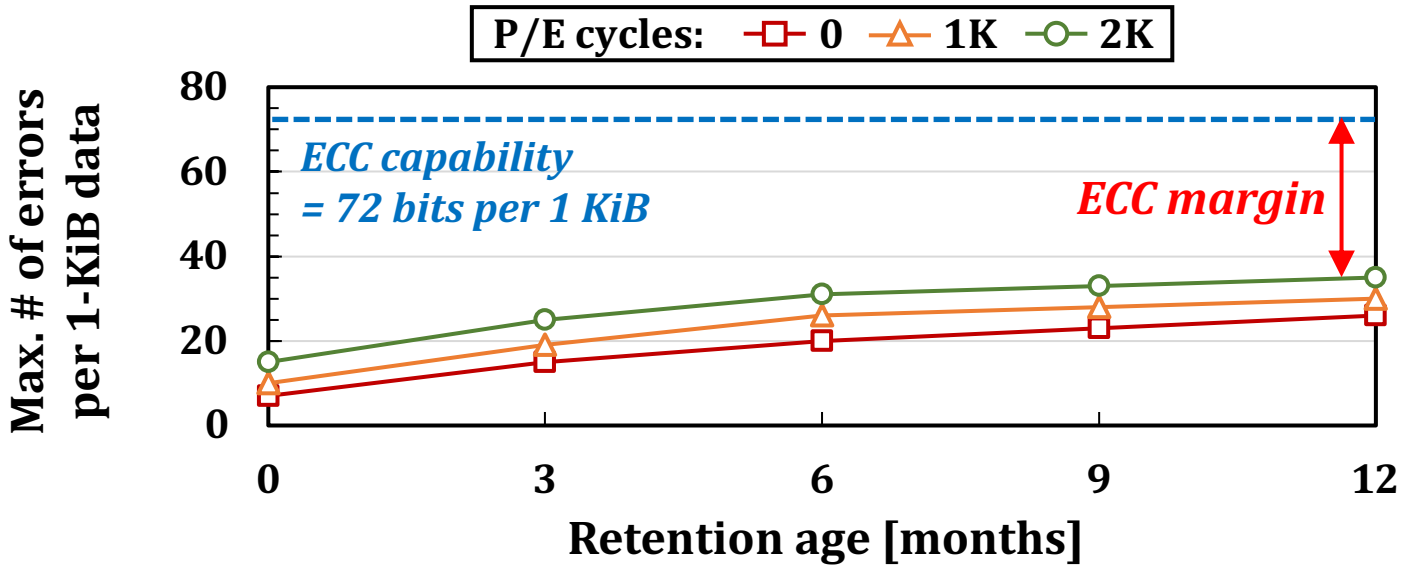
- Condition 1: Large ECC margin in the final retry step
  - Strong ECC: 72 bits correctable per 1-KiB data
  - Use of near-optimal V_{REF} in the final retry step
    - # of raw bit errors drastically increases if $V_{REF} >> V_{OPT}$
    - ∴ In the final retry step, $V_{RRN} \sim V_{OPT}$
- Condition 2: Sufficient reliability margin in read-timing parameters
  - Manufacturers pessimistically set read-timing parameters
  - To cover for worst-case process variation and operating conditions
- We experimentally analyze if these conditions hold

AR²: Real-Device Characterization

- Goals: Rigorously characterize
  - The ECC margin in the final retry step
  - Reliability impact of reducing read-timing parameters
  - Under different operating conditions

- Methodology
  - 160 real chips (48WL-layer 3D TLC NAND memory)
  - Randomly selected 11,059,200 pages
  - FPGA-based custom flash controller
    - Basic commands + test-mode commands (e.g., changing V_{REF} values and read-timing parameters)

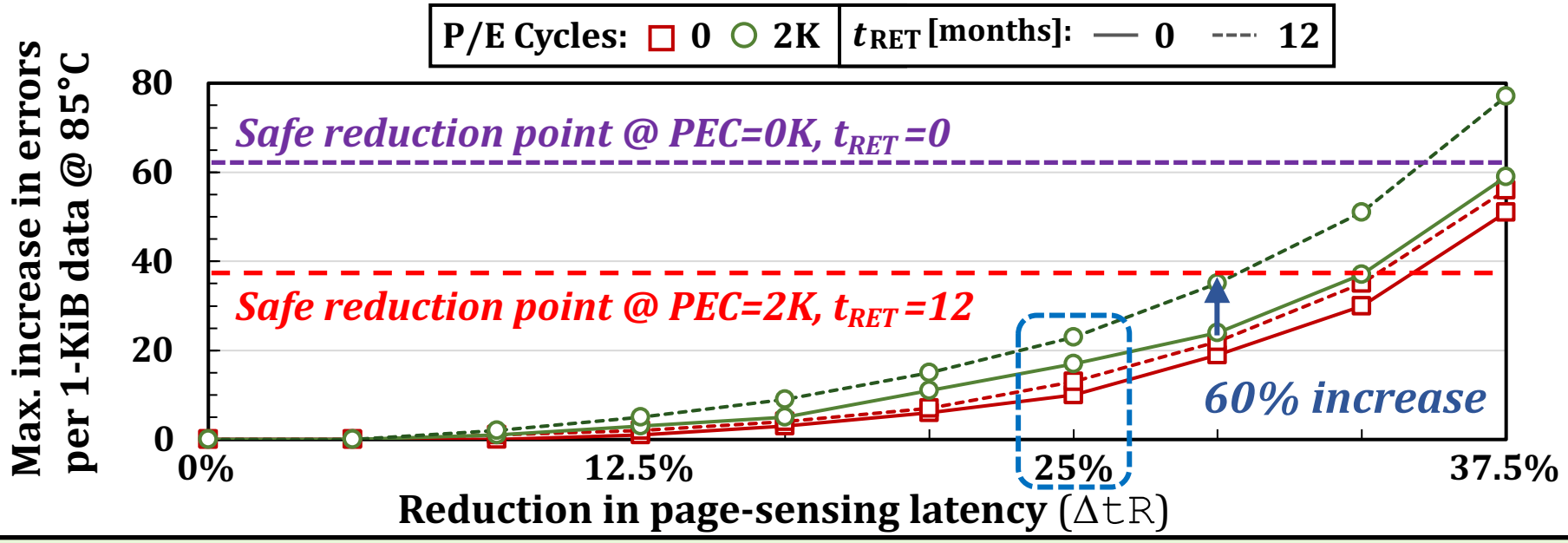
AR²: ECC Margin in the Final Retry Step



Large ECC margin in the final retry step
even under worst-case operating conditions

Smaller ECC margin
at higher P/E cycles and longer retention age

AR²: Effect of Reducing Timing Parameters



Large reliability margin in read-timing parameters
→ 25% tR reduction under worst-case conditions

Considerable variation depending on operating conditions

AR²: Effect of Reducing Timing Parameters

AR² Device-Characterization Takeaways

1. AR² can easily work in state-of-the-art NAND flash chips

2. Must properly reduce `tR` depending on the current operating conditions

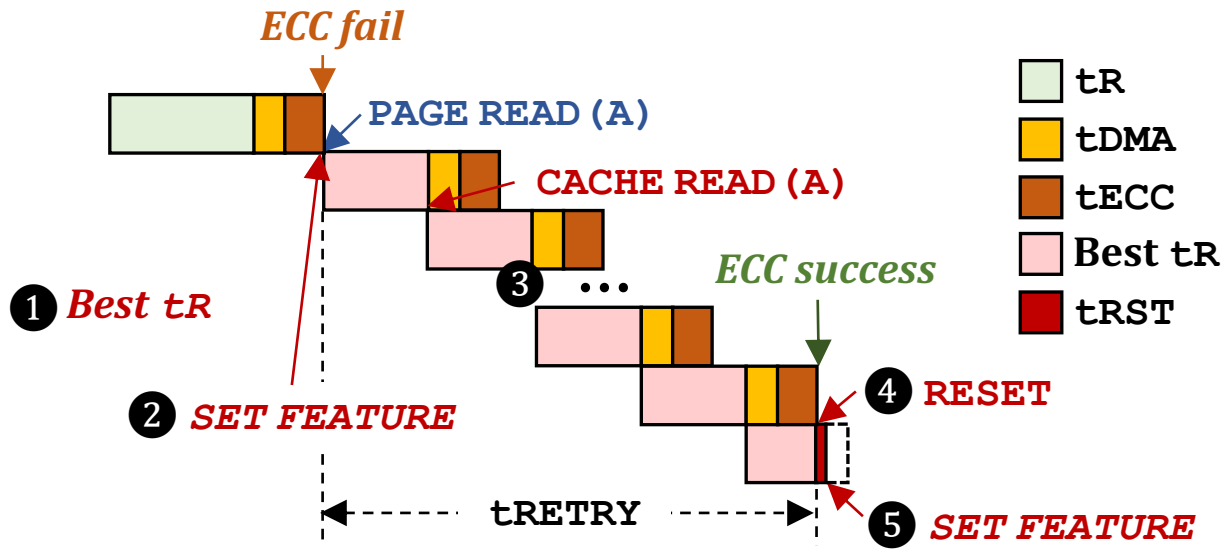
Considerable variation depending on operating conditions

Pipelined & Adaptive Read-Retry

PEC	t_{RET} [days]	tR [µs]
< 250	< 60	65
	⋮	⋮
	< 360	70
	⋮	⋮
< 1.5K	< 60	70
	⋮	⋮
	< 360	75

Current conditions

Read-timing Parameter Table



No change to chips and no impact on V_{TH} states
→ Easy to combine with other techniques

Talk Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR^2: Pipelined Read-Retry
- AR^2: Adaptive Read-Retry
- Evaluation Results

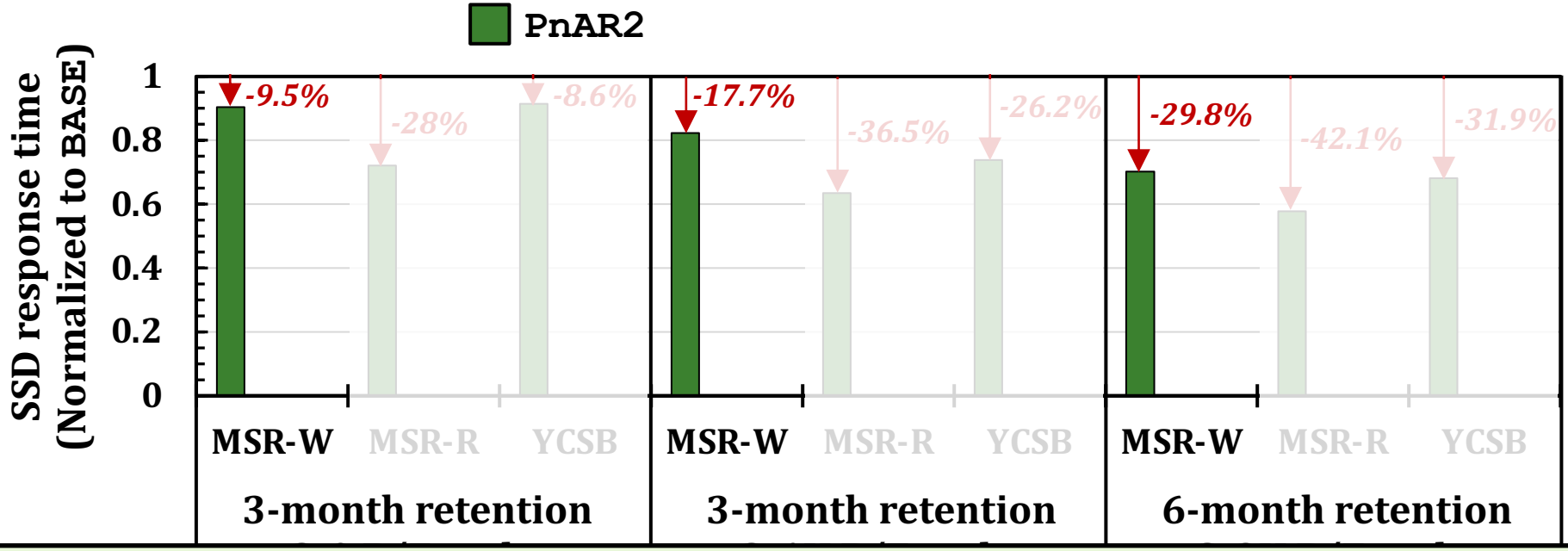
Evaluation Methodology

- **Simulator:** MQSim [Tavakkol, FAST18]
  - Extend NAND flash models w/ real-device characterization results

- **Workload:** 12 real-world I/O workloads
  - 6 from Microsoft Research Cambridge (MSRC) traces
    - 2 write-dominant: stg-0, hm-0
    - 4 read-dominant: prn-1, proj-1, mds-1, usr-1
  - 6 from Yahoo! Cloud Service Benchmark (YCSB)

- **Baselines**
  - **IDEAL:** An ideal SSD where no read-retry occurs
  - **BASE:** A high-end SSD w/o read-retry mitigation
  - **SOTA:** A state-of-the-art read-retry mitigation scheme [Shim, MICRO19]
    - Reduces the average number of retry steps by 70%
    - By predicting V_{REF} values close to the optimal values

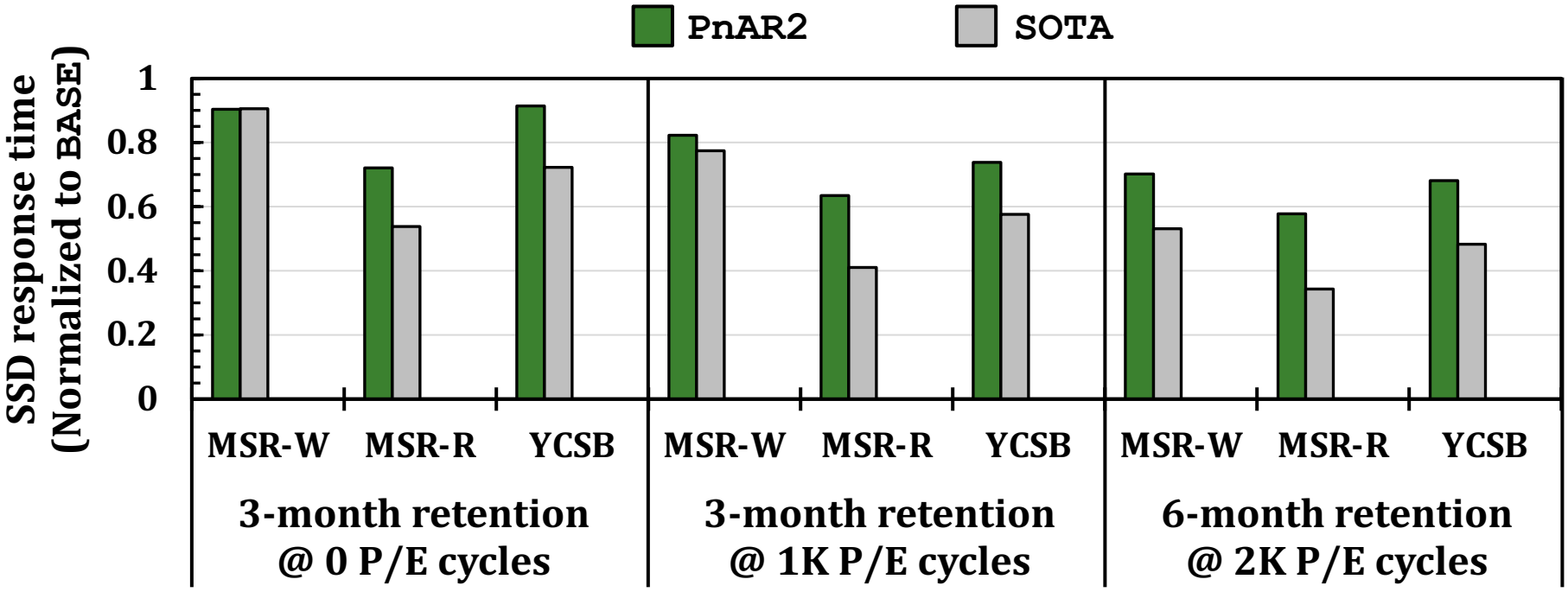
Results: PR²+AR² Performance



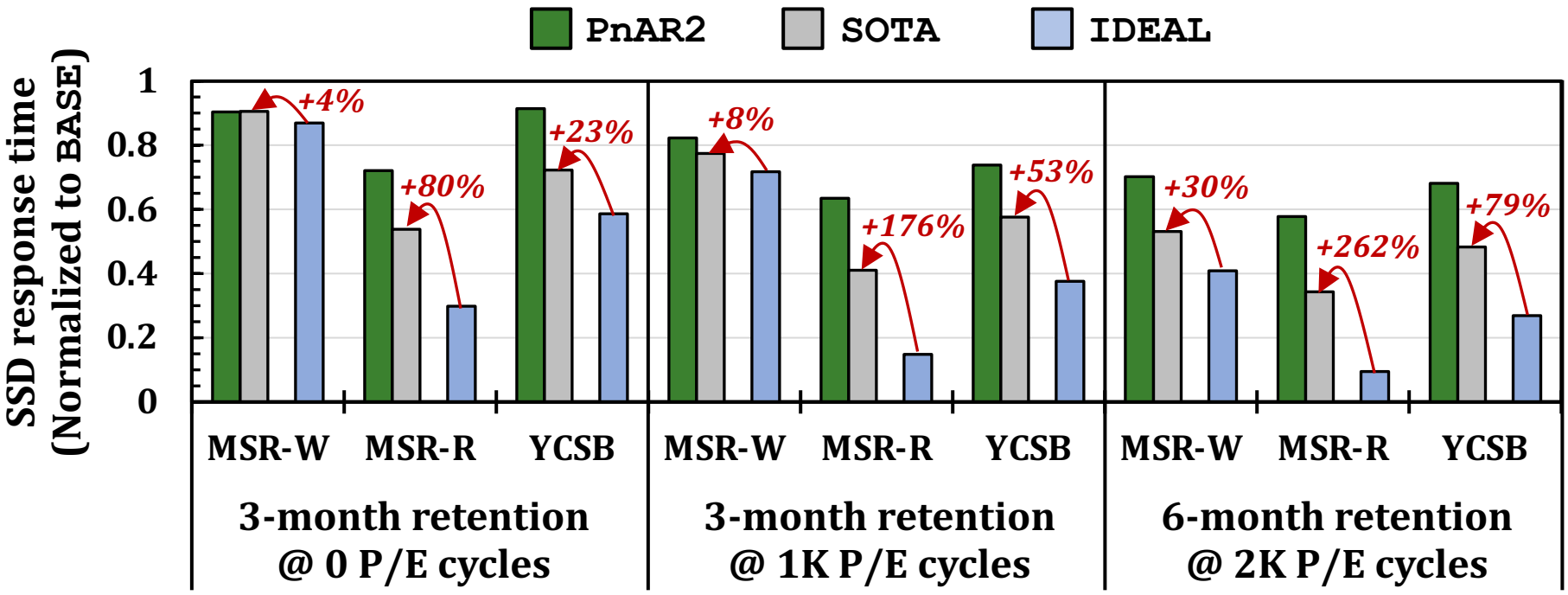
Large response time improvement:
Up to 42% (26% on average)

Considerable improvement in write-dominant workloads due to garbage-collection reads

Results: SOTA & Optimal Performance

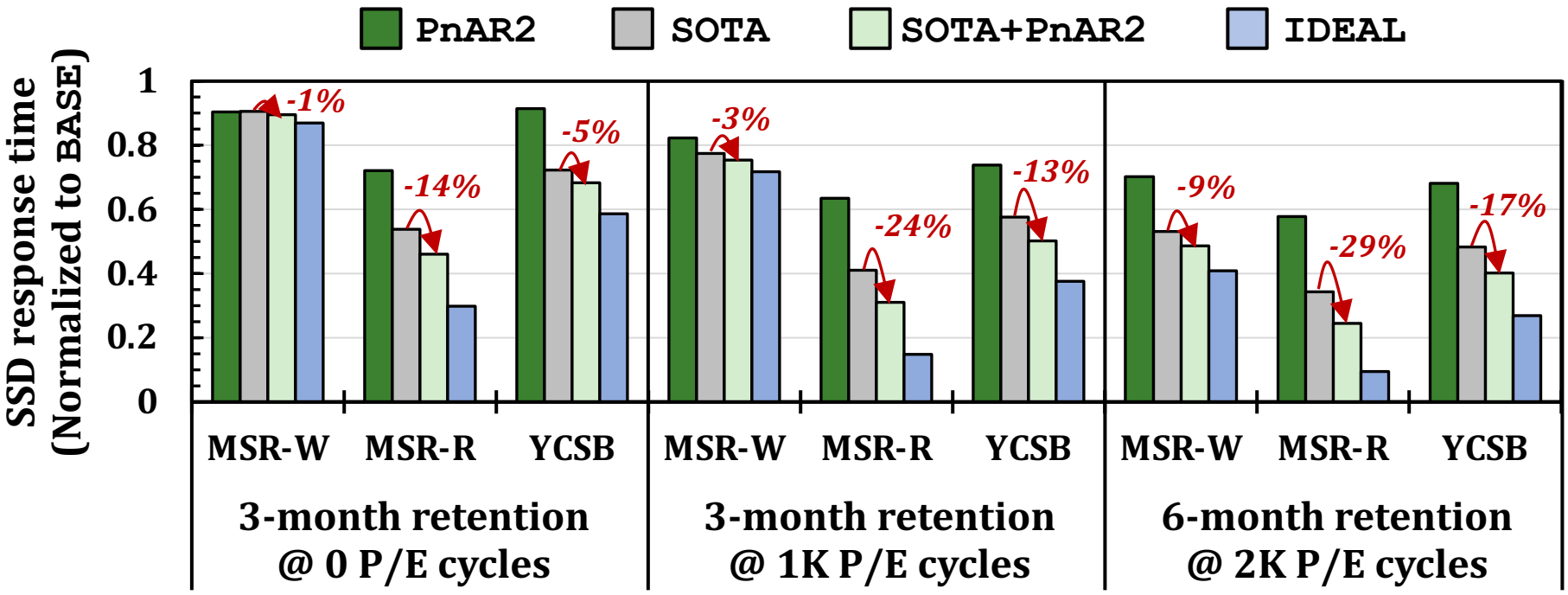


Results: SOTA & Optimal Performance



SOTA has large gap from ideal SSD

Evaluation Results: SSD Response Time



Up to 29% performance improvement when combined with SOTA

Other Analyses in the Paper

- Thorough analysis of read mechanism in modern SSDs
- More detailed results from real-device characterization
  - Effect of reducing individual read-timing parameters
  - Effect of reducing multiple read-timing parameters
  - Effect of operating temperature
  - How to choose the best read-timing paratmers
- Detailed evaluation of PR^2 and AR^2 when applied individually
- Discussion of future directions to reduce SSD read latency

Executive Summary

- **Problem:** Long read latency in modern SSDs due to read-retry
  - Frequently requires multiple retry steps to read an erroneous page
- **Key Ideas:**
  - **Pipelined Read-Retry (PR^2):** Concurrently perform consecutive retry steps using the CACHE READ command
  - **Adaptive Read-Retry (AR^2):** Reduce read-timing parameters for every retry step by exploiting the reliability margin provided by strong ECC
- **Evaluation Results:** Our proposal improves SSD response time by
  - Up to 51% (35% on average) compared to a high-end SSD
  - Up to 32% (17% on average) compared to a state-of-the-art baseline

We hope that our key idea and characterization results inspire many valuable studies going forward

Reducing Solid-State Drive Read Latency by Optimizing Read-Retry

Jisung Park[1], Myungsuk Kim[2], Myoungjun Chun[2],
Lois Orosa[1], Jihong Kim[2], and Onur Mutlu[1]

[1] SAFARI ETH zürich

[2] VERI LUX MEA TAS

ASPLOS 2021 (Session 17: Solid State Drives)