

Optimizing NAND Flash-Based SSDs via *Retention Relaxation*

Ren-Shuo Liu^{*}, Chia-Lin Yang^{*}, Wei Wu[†]

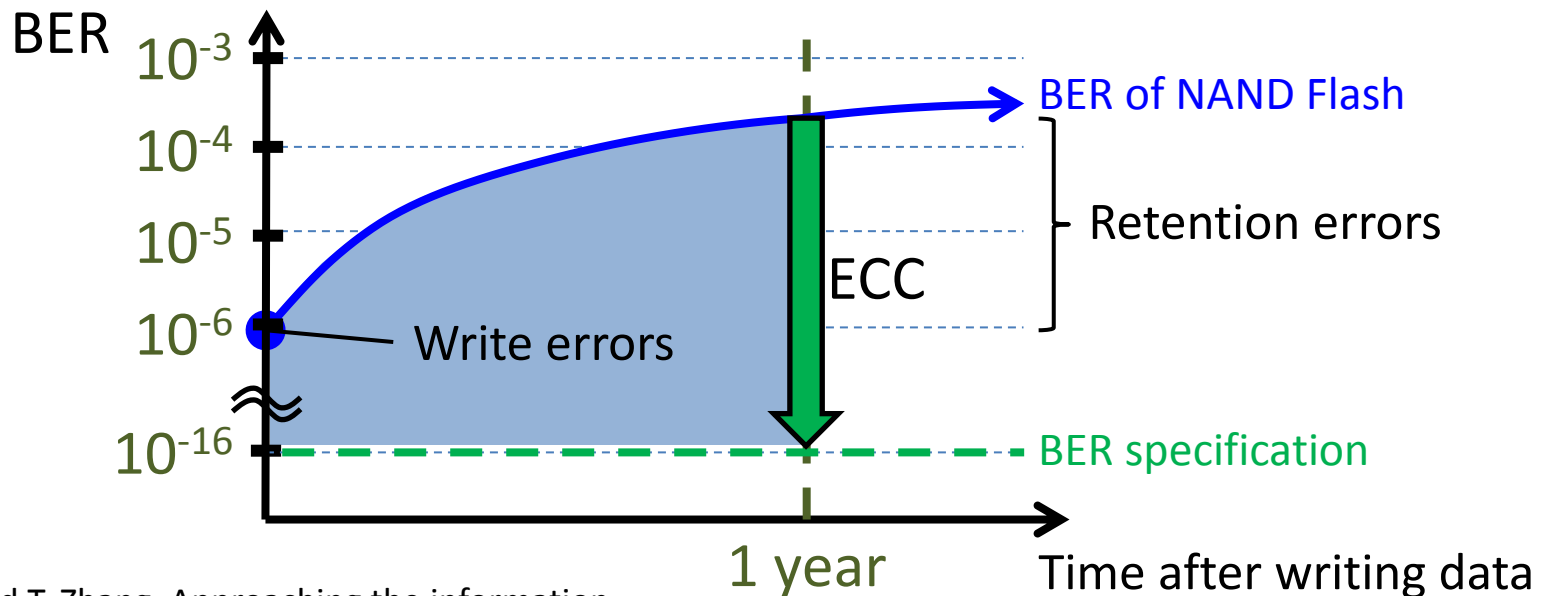
^{*}National Taiwan University and [†]Intel Corporation



2/15/2012 USENIX FAST '12

NAND Flash in Reliable Storage

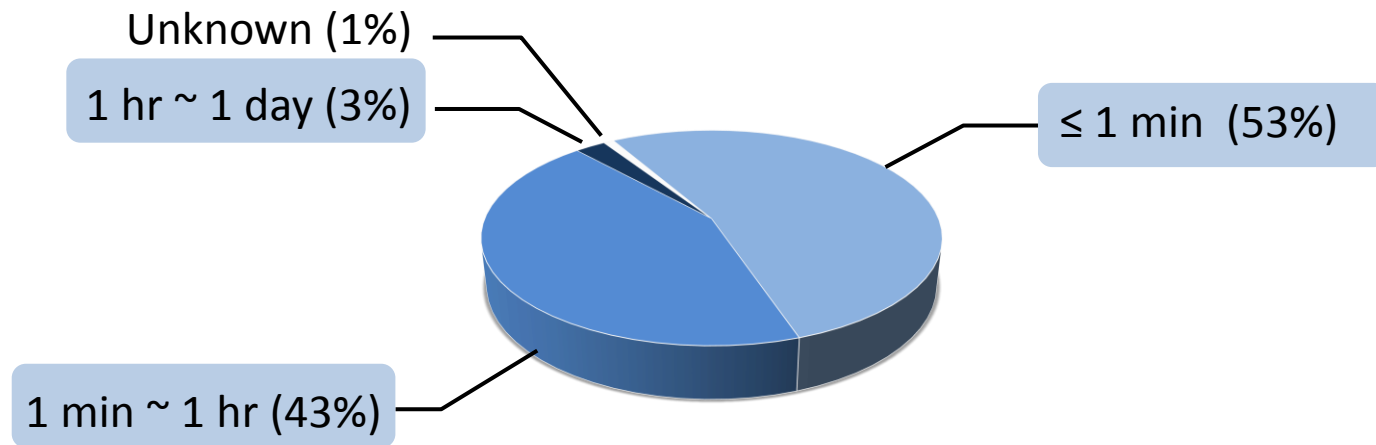
- Two main reliability specifications
 - Bit error rate (BER): $10^{-13} \sim 10^{-16}$
 - Data retention: 10 years (cycled to 10% of the max. endurance)
1 year (cycled to 100% of the max. endurance)
- As NAND Flash's density increases, its raw reliability degrades
 - Need to slow down writes to mitigate the worsening BER
 - Need stronger ECCs
 - When the BER $\geq 10^{-3}$, advanced ECCs such as LDPC (low-density parity-check) are required*



* S. Li and T. Zhang. Approaching the information theoretical bound of multi-level NAND Flash memory storage efficiency. IMW '09

Actual Retention Requirements

- Retention requirements in real-world applications are **usually much shorter than a year**



Retention breakdown of a TPC-C workload

Our Contribution

- Retention Relaxation
 - Exploit the gap between retention specification vs. actual retention requirements to improve **write speed** or **ECC cost/performance** in Flash-based SSDs

Industrial standards:
1 to 10 years

vs.

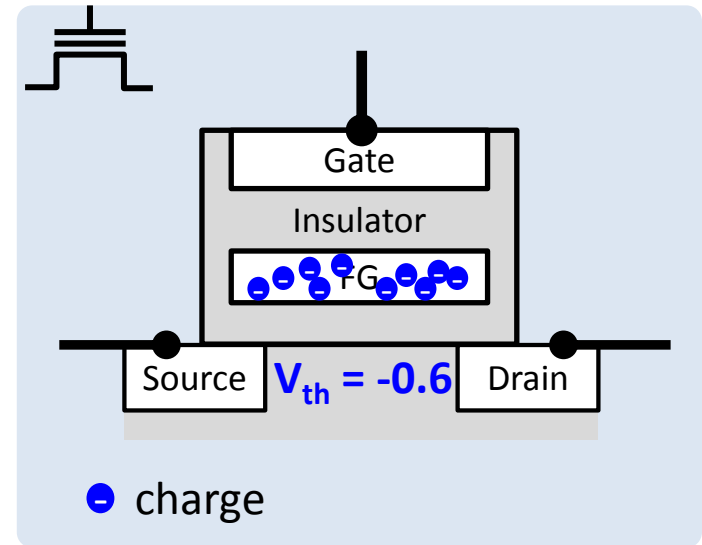
Actual requirements:
days or shorter

Outline

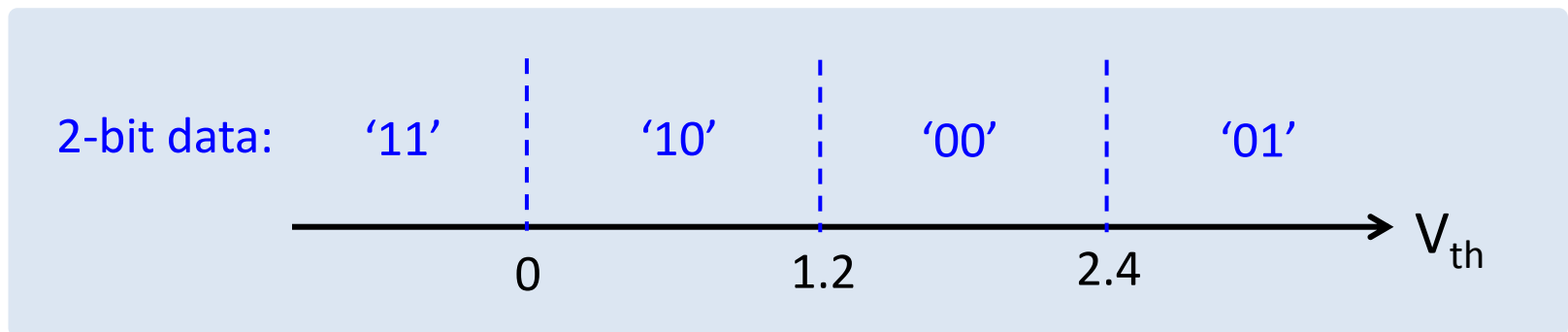
- Motivation
- **NAND Flash background**
- Main idea
- Methodology
- Evaluation
- Conclusions

NAND Flash Background

- NAND Flash memories
 - Composed of floating gate (FG) transistors
 - Injecting charge on the FG can adjust a transistor's threshold voltage (V_{th})
 - Different V_{th} levels are used to represent different data



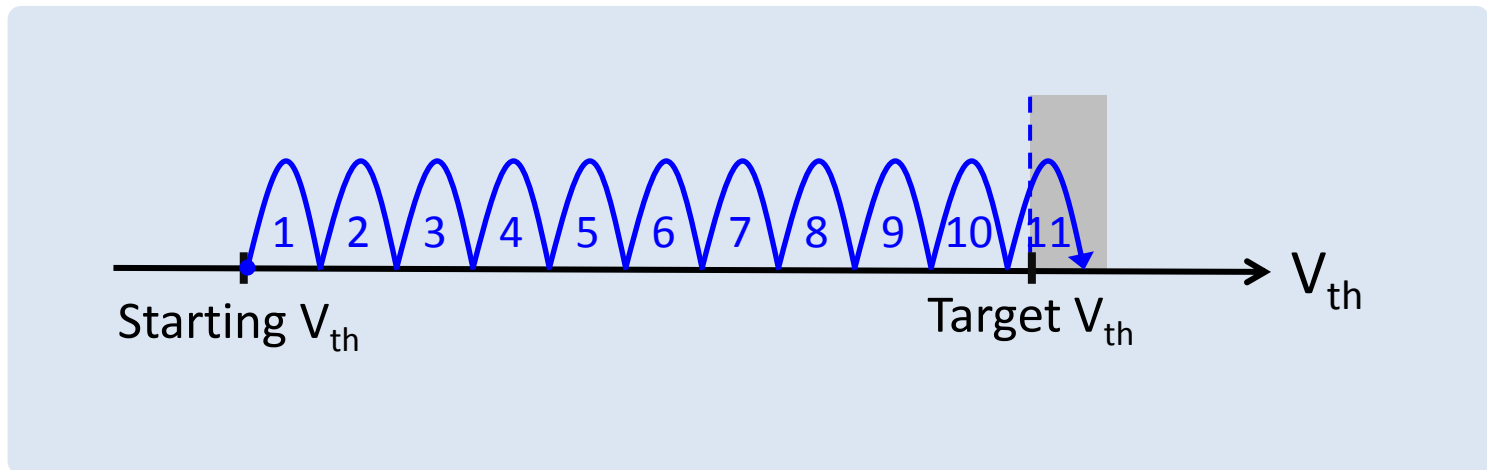
Flash cell structure



Example V_{th} levels for 2-bit cells

Programming NAND Flash

- Incremental step pulse programming (ISPP)*
 - Increase V_{th} step-by-step with step increment = ΔV_p



- Tradeoffs in ISPP
 - $\Delta V_p \uparrow \rightarrow$ fewer steps (**faster**)
 - $\Delta V_p \downarrow \rightarrow$ more precise control on V_{th} (**fewer write errors**)

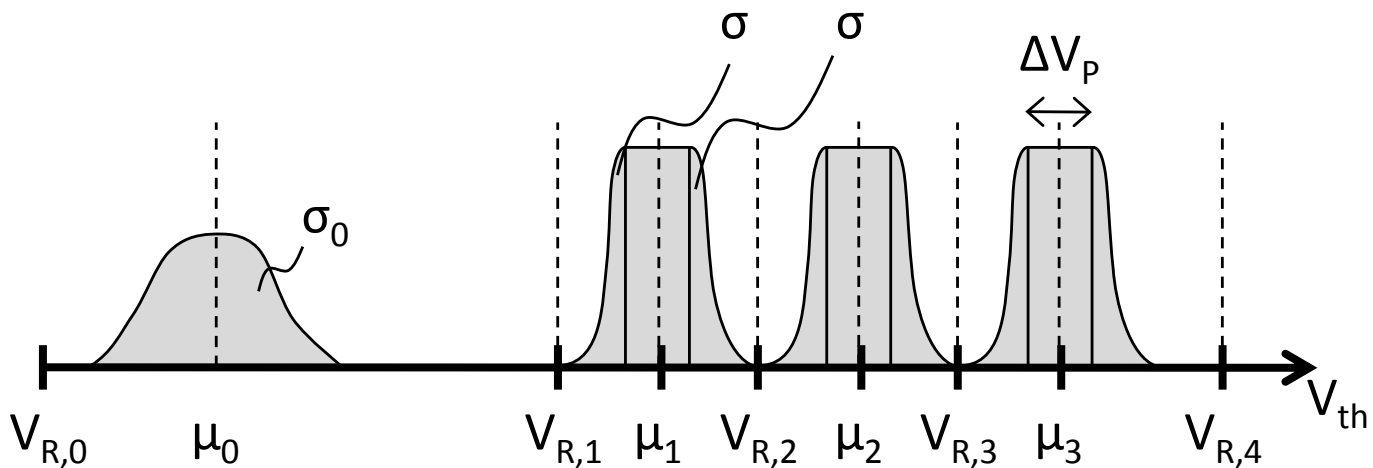
* Suh *et al.*. A 3.3 V 32 Mb NAND Flash memory with incremental step pulse programming scheme. JSSC '95

V_{th} Distribution of NAND Flash

- Probability density function of cells' V_{th}
 - Modeled using bell-shaped functions in the previous work*

$$P_k(v) = \alpha_0 \cdot e^{-\frac{(v-\mu_0)^2}{2\sigma_0^2}}, k = 0$$

$$P_k(v) = \begin{cases} \alpha & , \mu_k - \frac{\Delta V_P}{2} \leq v \leq \mu_k + \frac{\Delta V_P}{2} \\ \alpha \cdot e^{-\frac{(v-\mu_k+0.5\Delta V_P)^2}{2\sigma^2}} & , v < \mu_k - \frac{\Delta V_P}{2} \\ \alpha \cdot e^{-\frac{(v-\mu_k-0.5\Delta V_P)^2}{2\sigma^2}} & , v > \mu_k + \frac{\Delta V_P}{2} \end{cases}$$

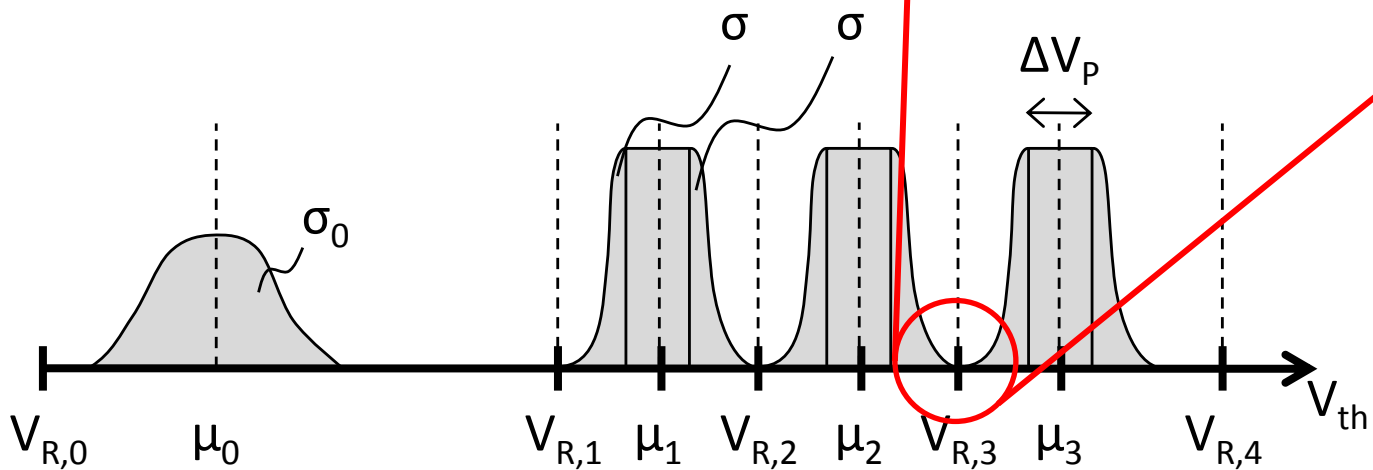
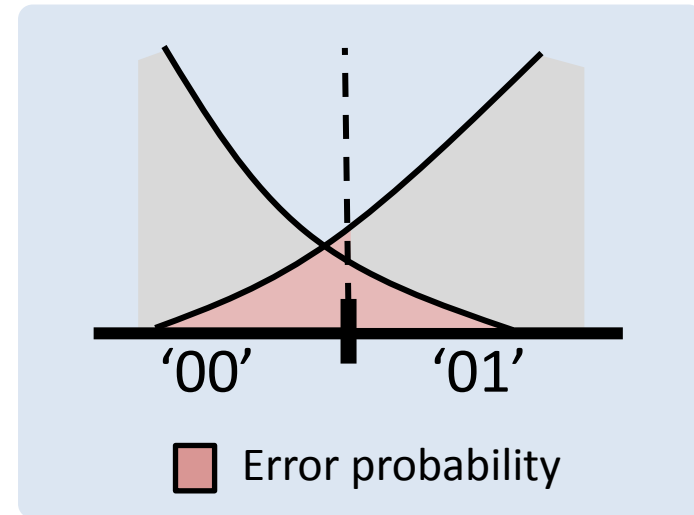


* Xie *et al.*. Using lossless data compression in data storage systems: Not for saving space. TC '11

Bit Error Rate vs. V_{th} Distribution

- RBER is the integral of the V_{th} distributions that fall into wrong states

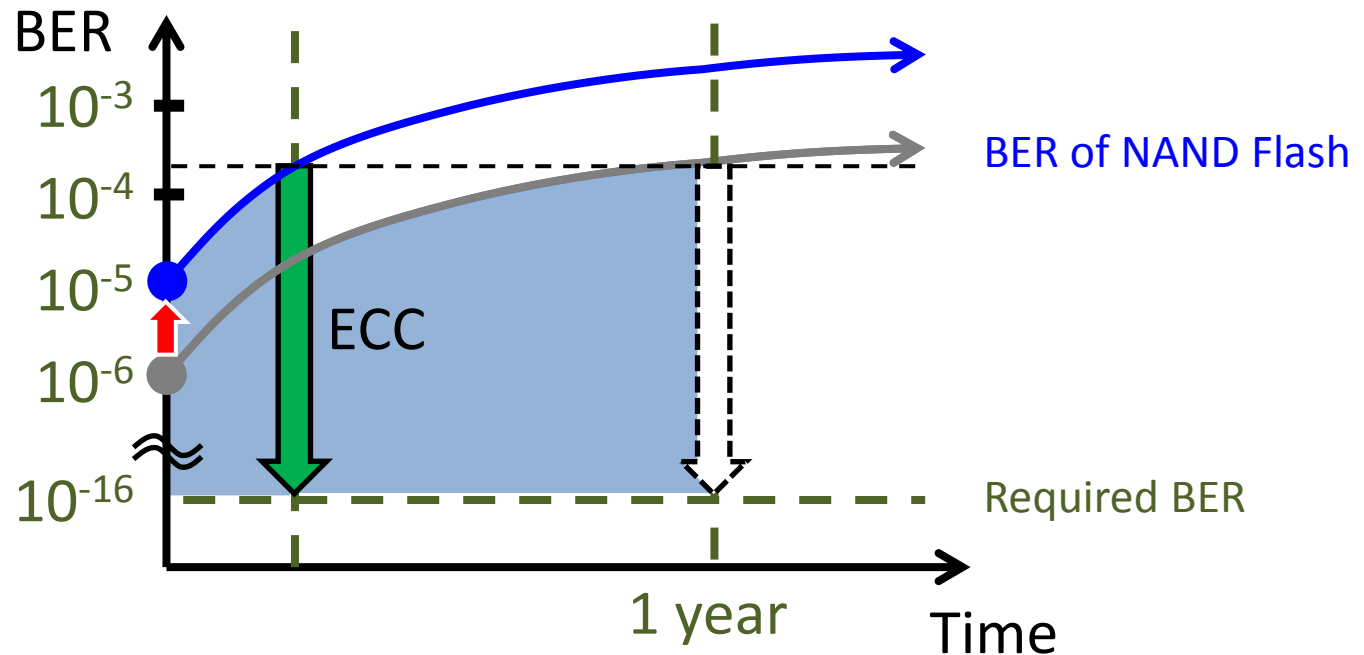
$$BER = \sum_{k=0}^3 \left(\underbrace{\int_{-\infty}^{V_{R,k}} P_k(v) dv}_{V_{th} \text{ lower than intended}} + \underbrace{\int_{V_{R,(k+1)}}^{\infty} P_k(v) dv}_{V_{th} \text{ higher than intended}} \right)$$



Outline

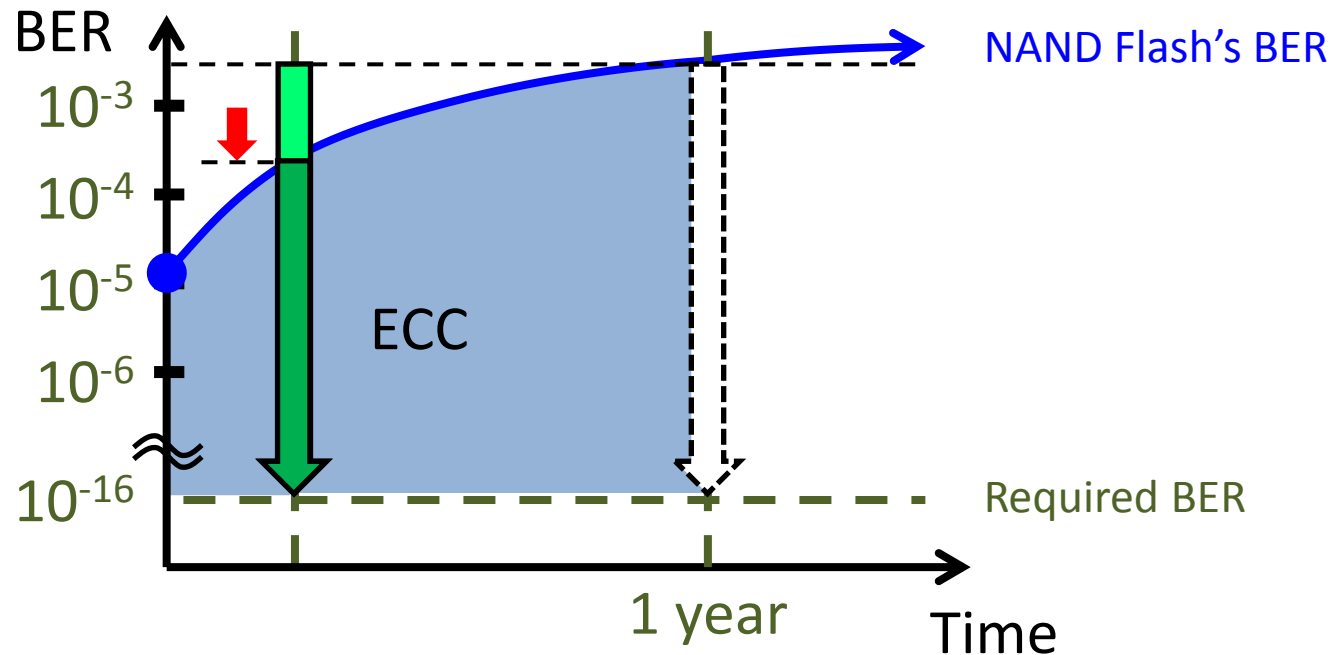
- Motivation
- NAND Flash background
- **Main idea**
 - **Retention time vs. Write speed**
 - **Retention time vs. ECC**
- Methodology
- Evaluation
- Conclusions

Retention Relaxation vs. Write Speed



- Shorter retention guarantee
 - Can tolerate more write errors
 - Allow larger ΔV_p in the ISPP
 - **Faster write**

Retention Relaxation vs. ECC



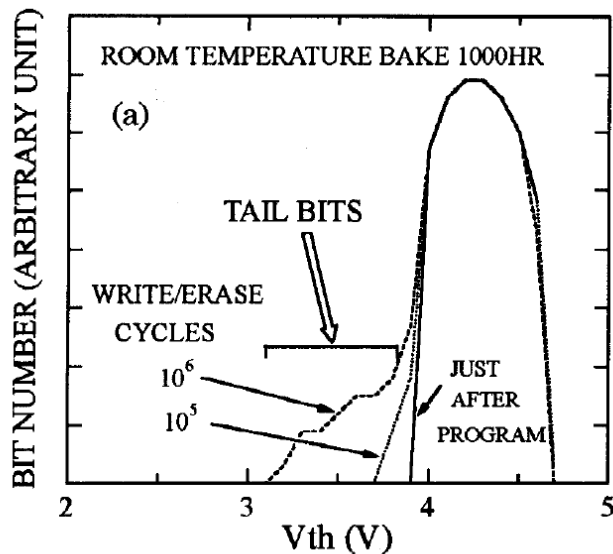
- Shorter retention guarantee
 - Need to tolerate fewer retention errors
 - Allow less capability of the ECC
 - **Simpler ECC design**

Outline

- Motivation
- NAND Flash background
- Main idea
- **Methodology**
 - **NAND Flash model**
 - Retention analysis on real-world applications
 - Retention-aware system architecture
- Evaluation
- Conclusions

Model Extension

- Base NAND Flash model is not able to capture the charge-loss effect which causes the low- V_{th} tail to widen over time*



Measurement of charge-loss effect

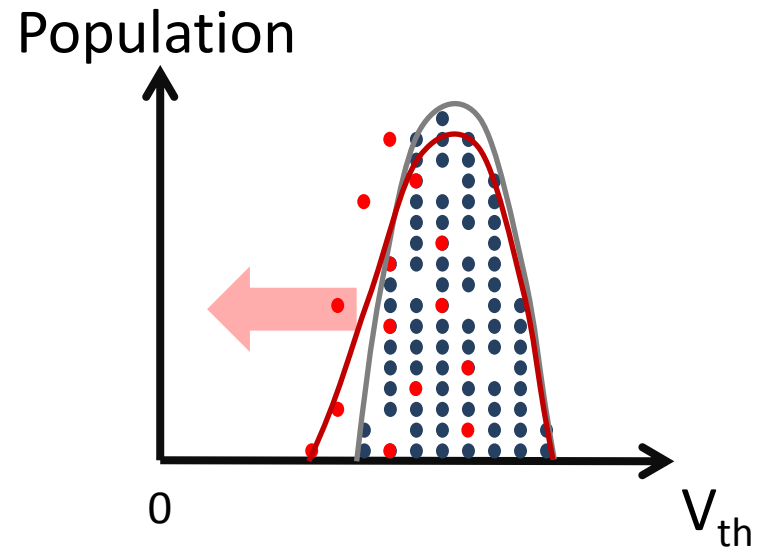


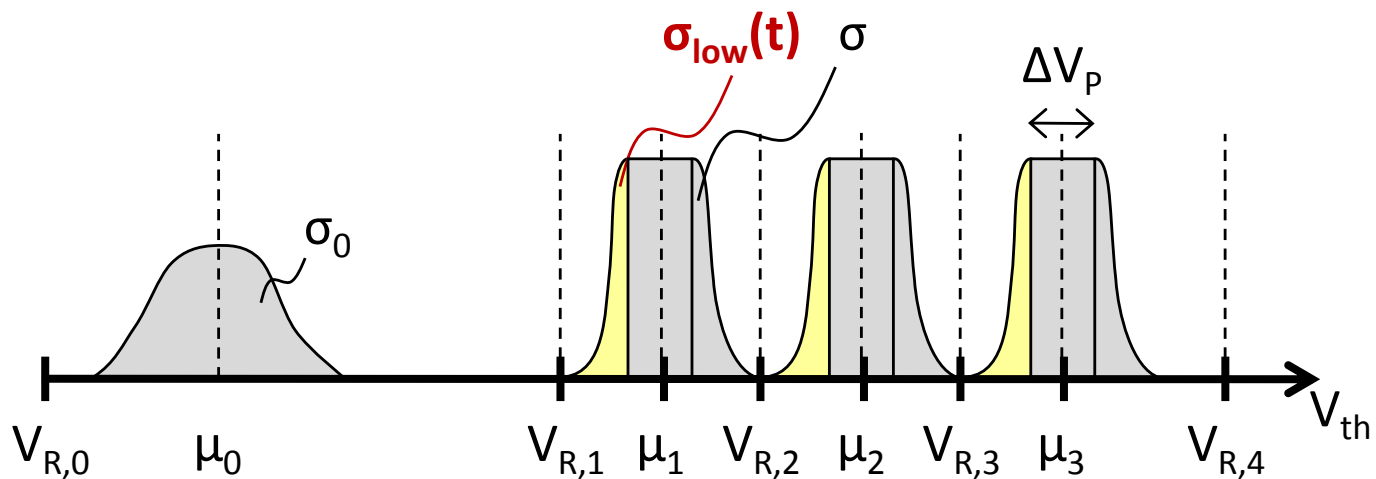
Illustration of charge-loss effect

* Arai *et al.*. Extended data retention process technology for highly reliable Flash EEPROMs of 10^6 to 10^7 W/E cycles, 14 IPRS '98

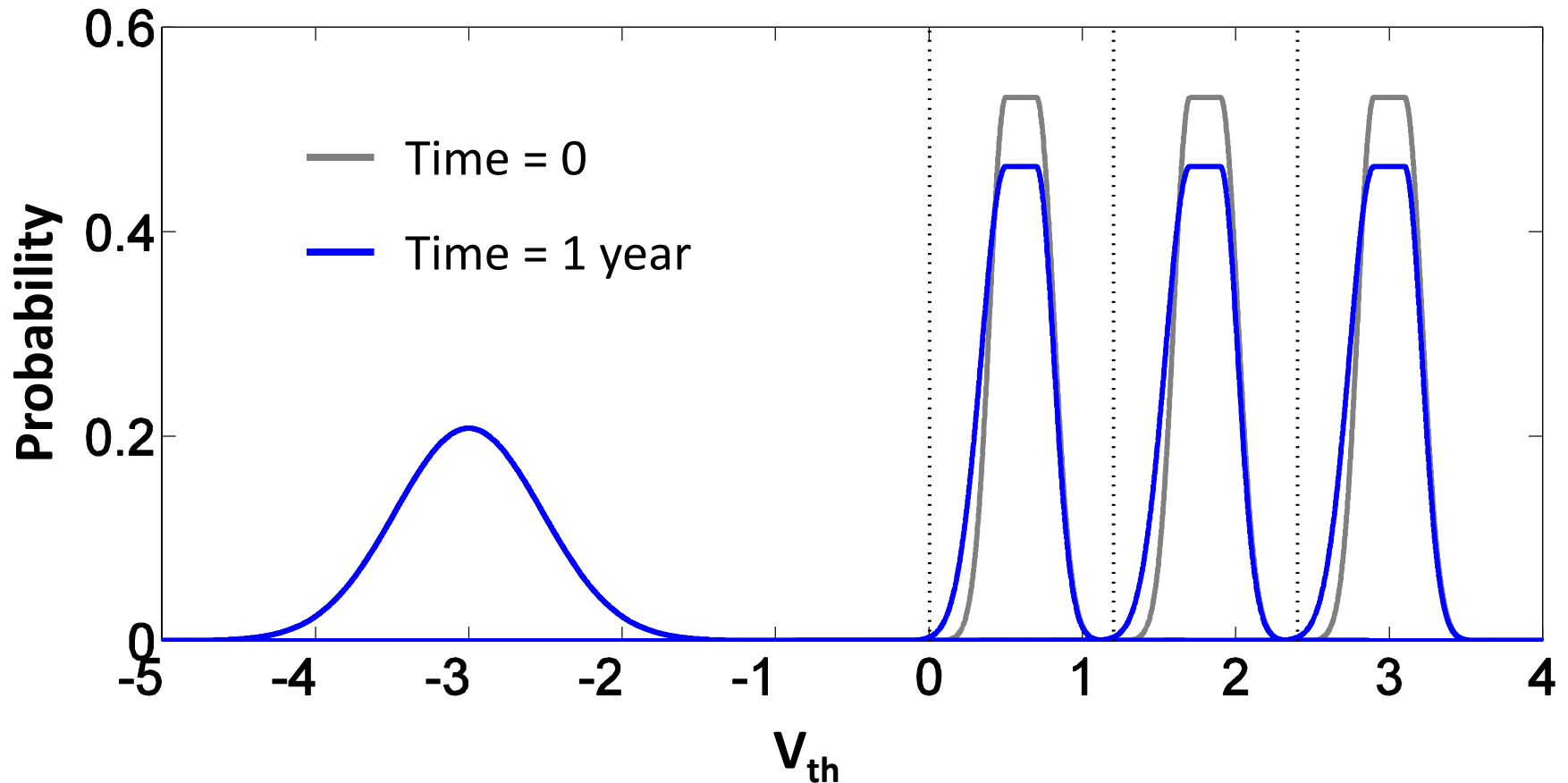
Model Extension

- Model the standard deviation of the low- V_{th} tail as a time-increasing function, $\sigma_{low}(t)$

	Low- V_{th} tail
Base model	$P_k(v) = \alpha \cdot e^{-\frac{(v-\mu_k+0.5\Delta V_P)^2}{2\sigma^2}}$
Extended model	$P_k(v, t) = \alpha(t) \cdot e^{-\frac{(v-\mu_k+0.5\Delta V_P)^2}{2\sigma_{low}(t)^2}}$

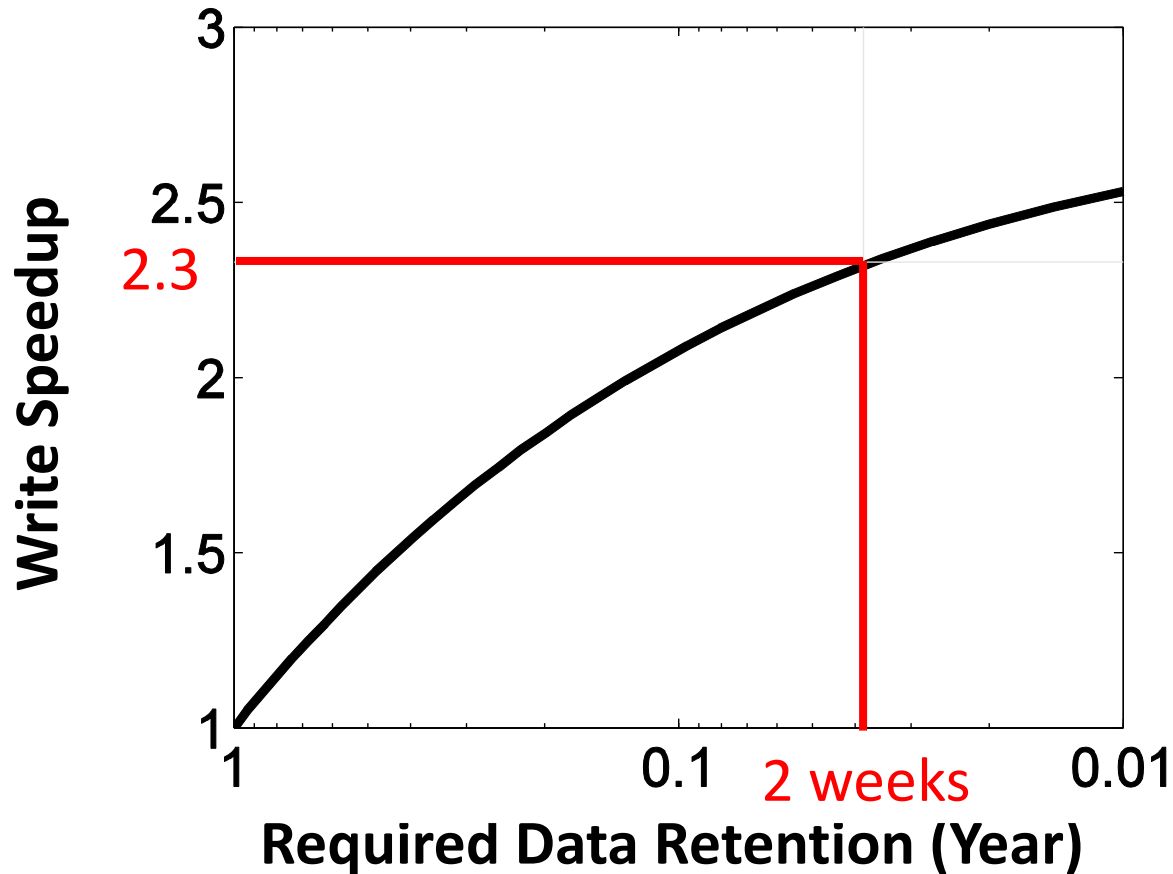


Modeling Results



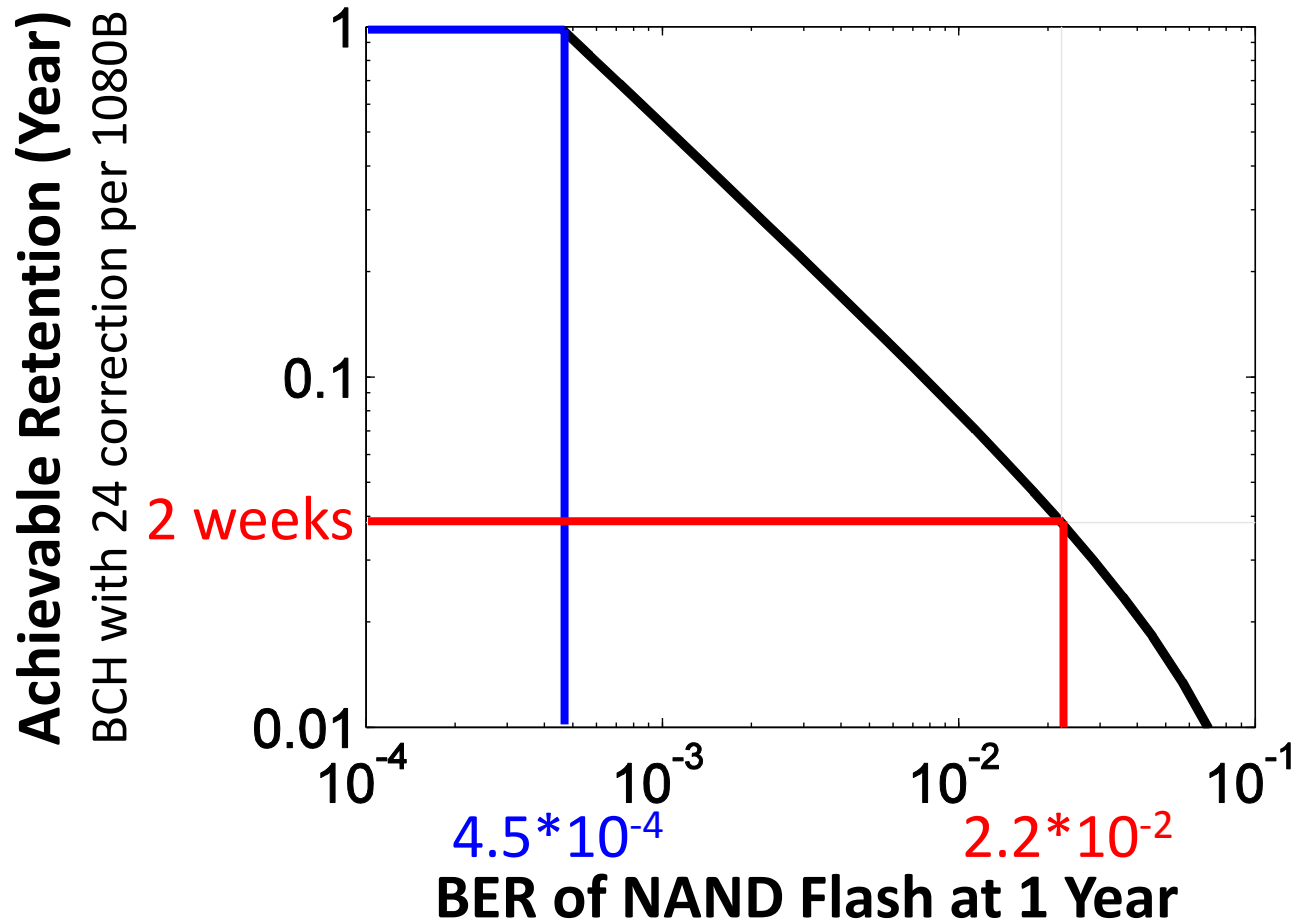
V_{th} Distribution vs. Time

Retention vs. Write Speed



- Relax retention from 1 year to 2 weeks
 - 2.3x write speedup is achievable

Retention vs. ECCs



- Relax retention from 1 year to 2 weeks
 - We can use the BCH (24 corrections per 1080B) to replace an LDPC whose strength is 2.2×10^{-2}

Outline

- Motivation
- NAND Flash background
- Main idea
- Methodology
 - NAND Flash model
 - **Retention analysis on real-world applications**
 - Retention-aware system architecture
- Evaluation
- Conclusions

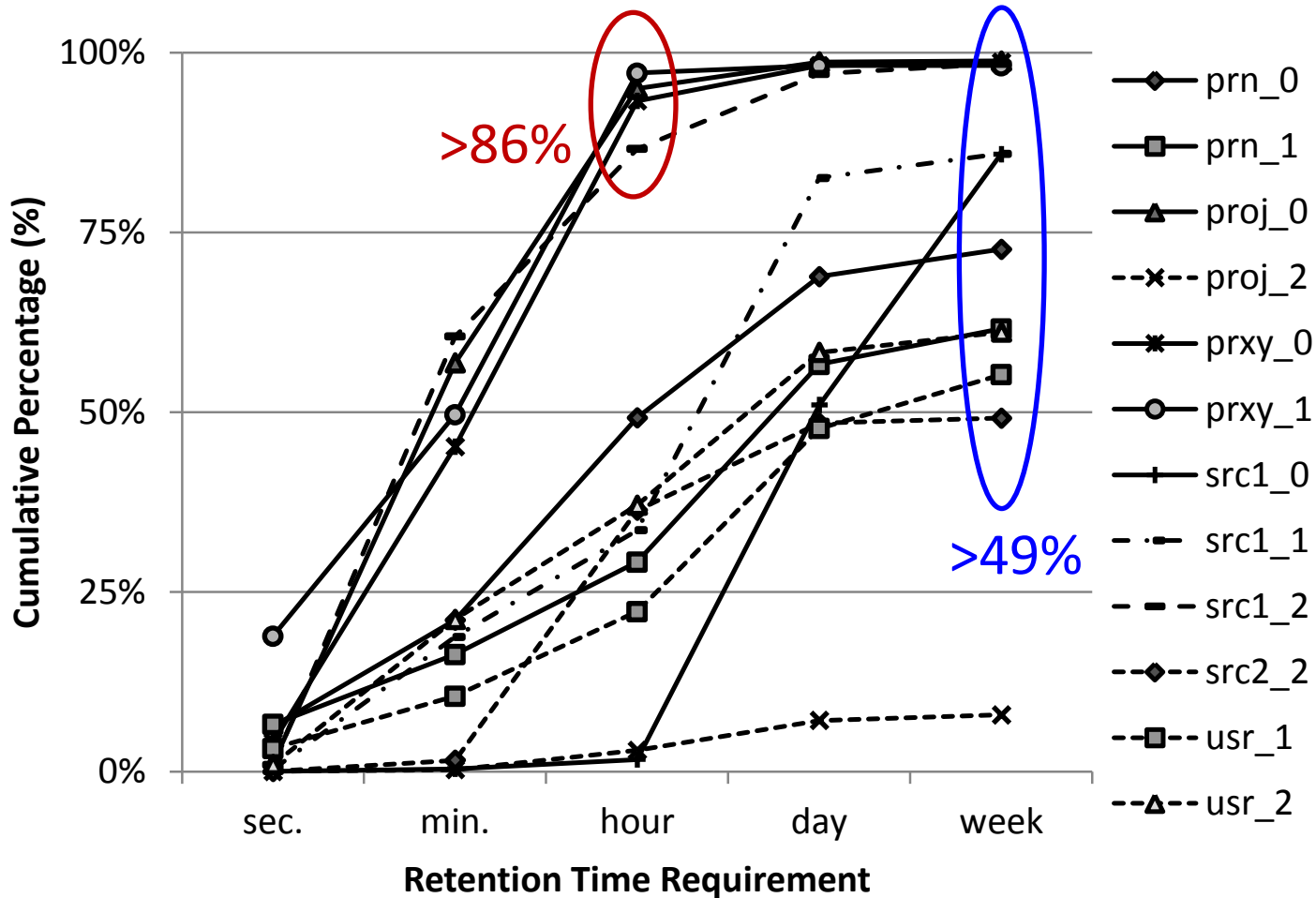
Retention Analysis

- Retention requirement is defined as
 - The time period from writing the sector until the sector is **overwritten**
- We analyze 3 sets of real-world applications

Category	Name	Description	Span
MSRC	prn_0 proj_0, proj_2 prxy_0, prxy_1 src1_0, src1_2 src2_2 usr_1, usr_2	Print server Project directories Web proxy Source control Source control User home directories	1 week
Hadoop	hd1 hd2	WordCount benchmark	1 day
TPC-C	tpcc1 tpcc2	OLTP benchmark	1 day

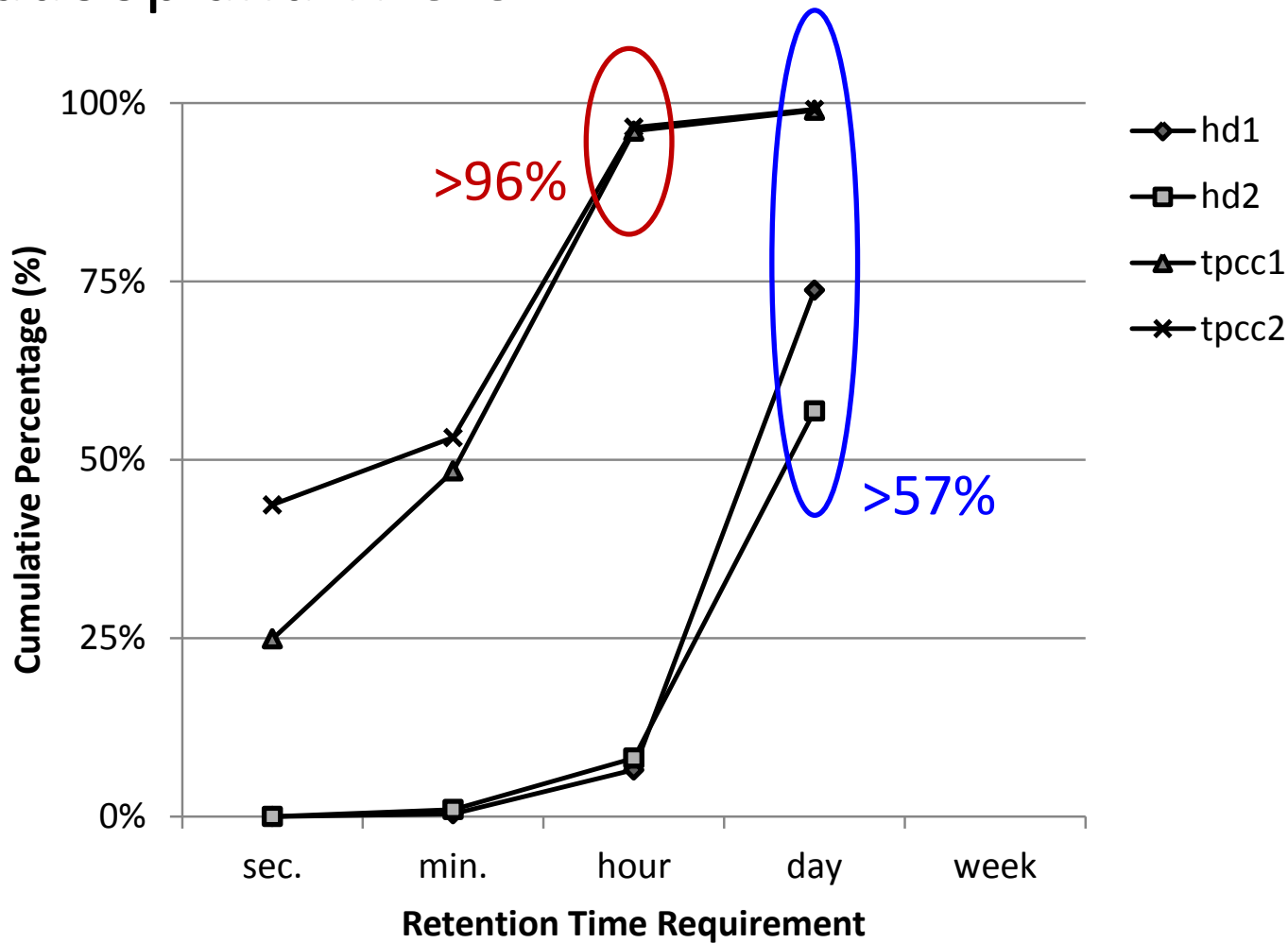
Retention Analysis

- MSRC



Retention Analysis


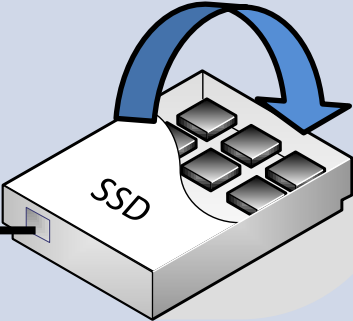
- Hadoop and TPC-C



Outline

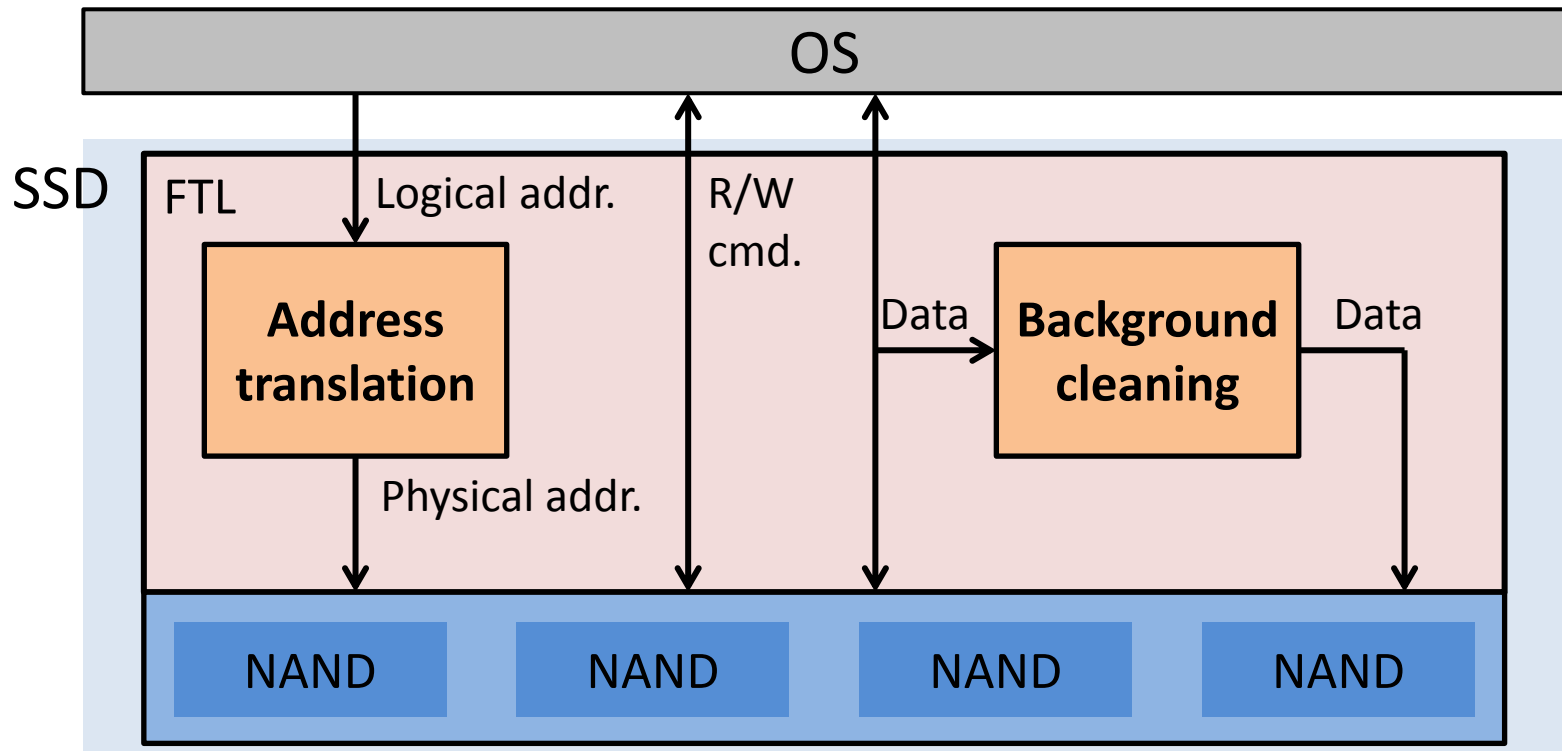
- Motivation
- NAND Flash background
- Main idea
- Methodology
 - NAND Flash model
 - Retention analysis on real-world applications
 - **Retention-aware system architecture**
- Evaluation
- Conclusions

Two-Level Retention Guarantee

	Host writes		Background writes
	 <p>The diagram shows a grey rounded square labeled 'OS' on the left. A black arrow points from the OS to a stack of colorful rectangular blocks representing data. A larger red arrow points from this stack to a white SSD device on the right. The SSD has a blue curved arrow looping over its top surface, indicating background operations.</p>		 <p>The diagram shows a white SSD device with several black chips on its surface. A blue curved arrow loops over the top of the device, representing background write operations.</p>
Description	Writes from the host		E.g., cleaning, wear-leveling
Importance of performance	High		Low
Retention requirements	Low		High (cold data)
Write operation	Short retention guarantee		Normal retention guarantee
	Fast write	Less-strong ECCs	

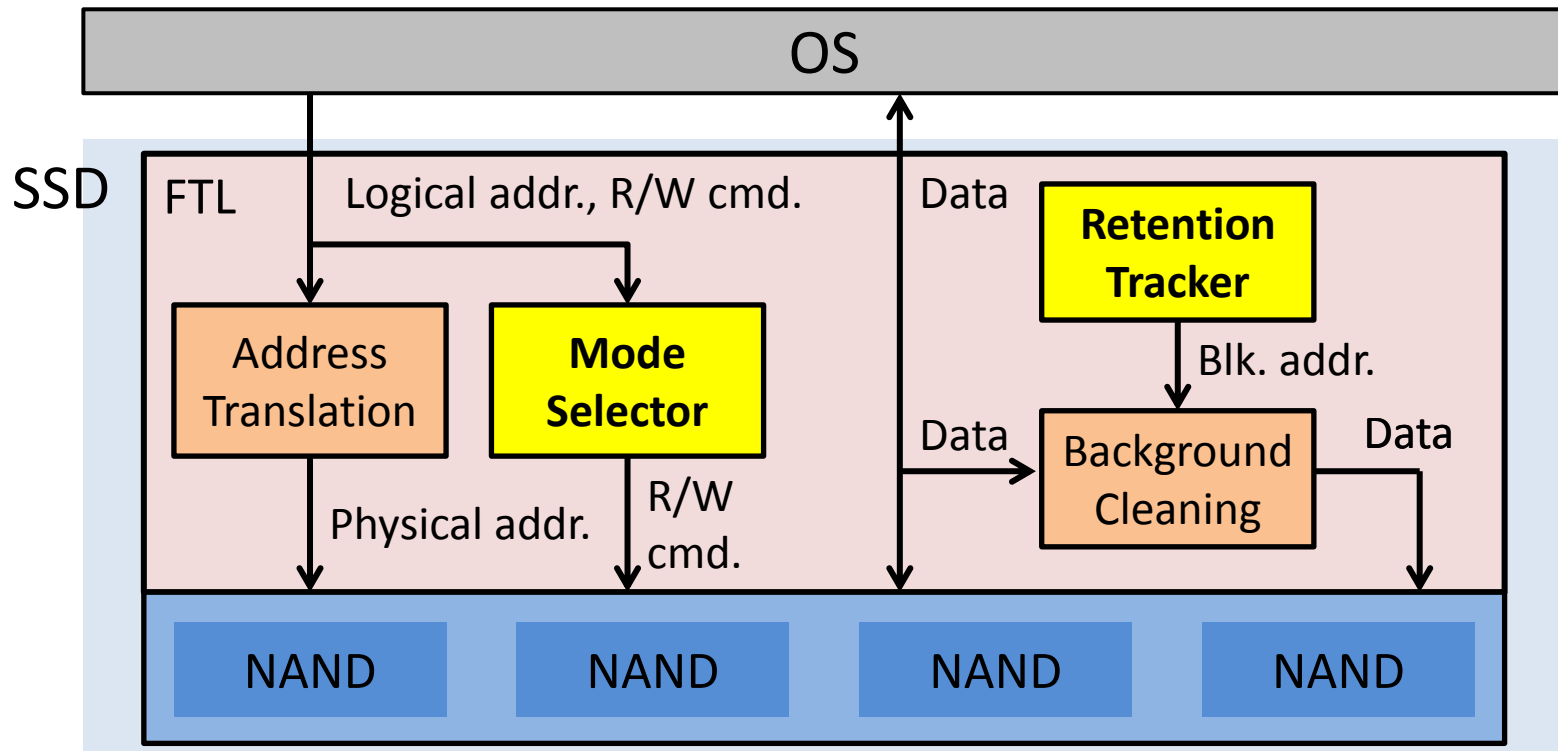
Retention-Aware FTL Design

- FTL (Flash Translation Layer)
 - Software layer emulating a block device over NAND Flash memories in SSDs



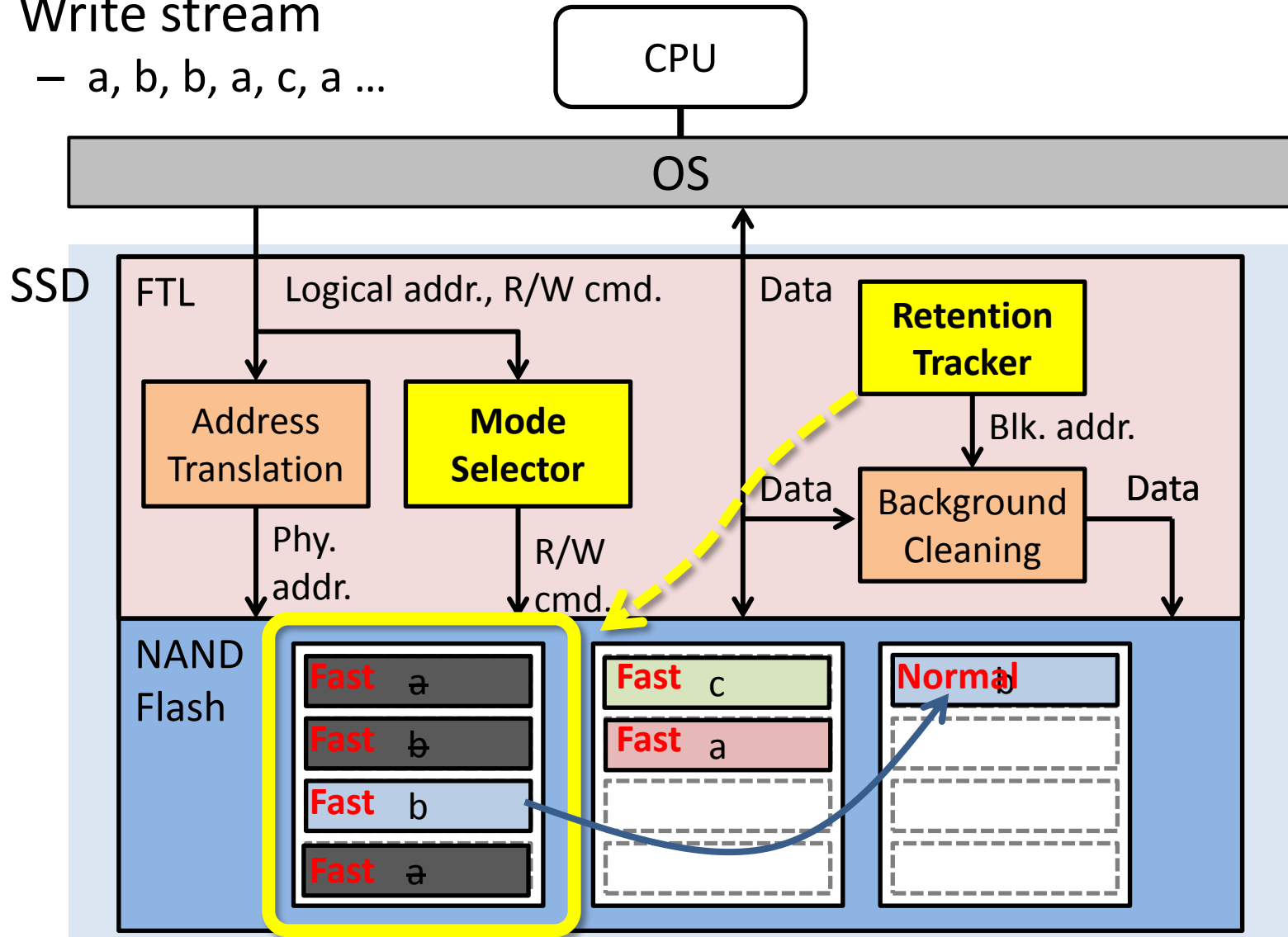
Retention-Aware FTL

- Two new components employed in the FTL
 - Mode Selector
 - Invoke different NAND Flash write commands or different ECC engines for blocks with different retention guarantees
 - Retention Tracker
 - Monitor the remaining retention time of blocks
 - Reprogramming a block when the block is about to run out of retention



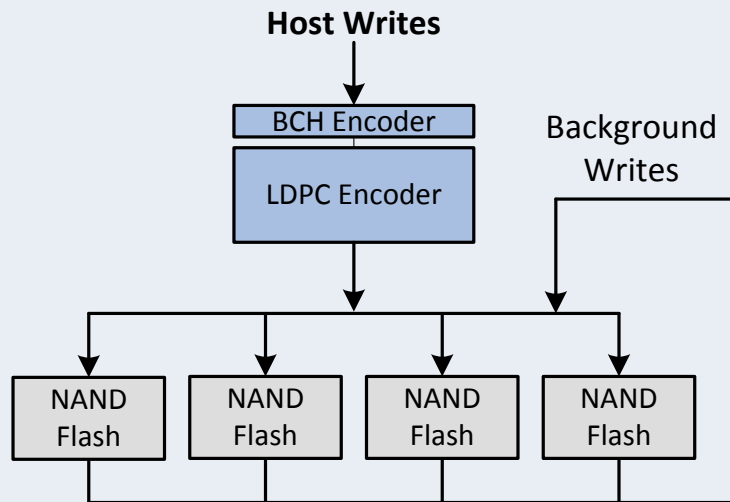
Retention Relaxation for Write Speedup

- Write stream
 - a, b, b, a, c, a ...



Retention Relaxation for ECC Optimization

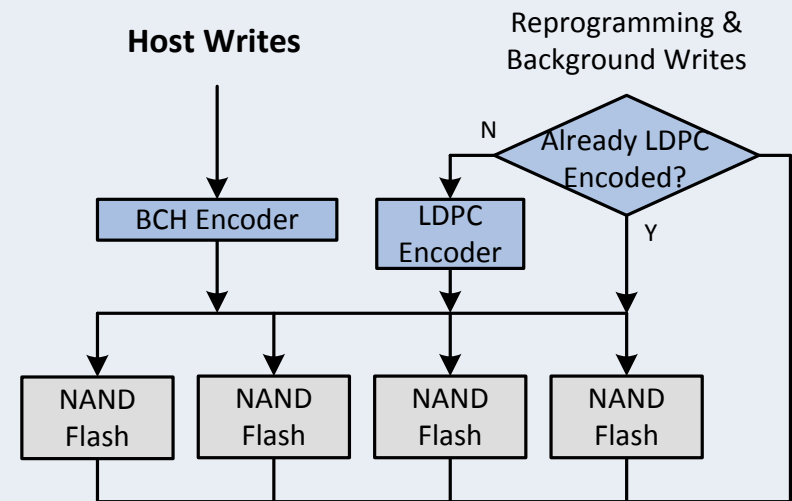
Concatenated BCH-LDPC



Issue:

Since all host writes go through the LDPC encoder, a high-throughput LDPC encoder is required, otherwise it will become the bottleneck

Retention-Aware Architecture



Advantages:

- Time-consuming LDPC is kept out of the critical performance path
- LDPC encodes only data with retention longer than what the BCH guarantees
- LDPC encoding can be scheduled over a period of time in the background

→ Reduce the throughput requirements of the LDPC

Outline

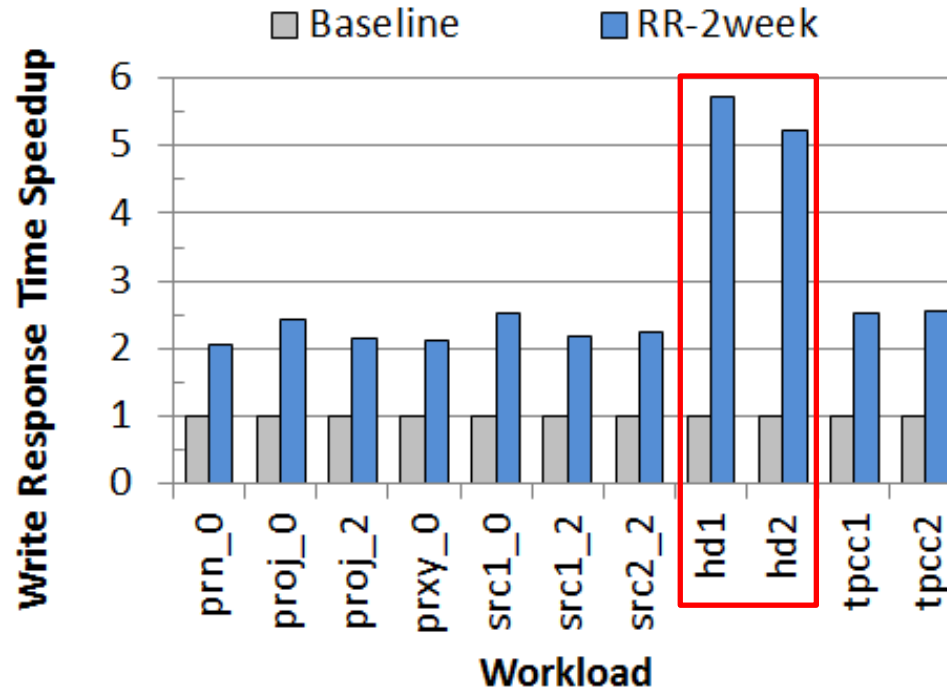
- Motivation
- NAND Flash background
- Main idea
- Methodology
- **Evaluation**
- Conclusions

Experimental Setup

- Simulations using Disksim 4.0 & SSDsim
- Workloads
 - 11 traces from MSRC, Hadoop, and TPC-C
- Two configurations are evaluated
 - Baseline: SSDs with 1-year retention for all writes
 - Proposed retention-relaxation design: RR-2week
 - 2-week retention guarantee for host writes
 - Blocks not overwritten in one week are reprogrammed with full retention guarantees

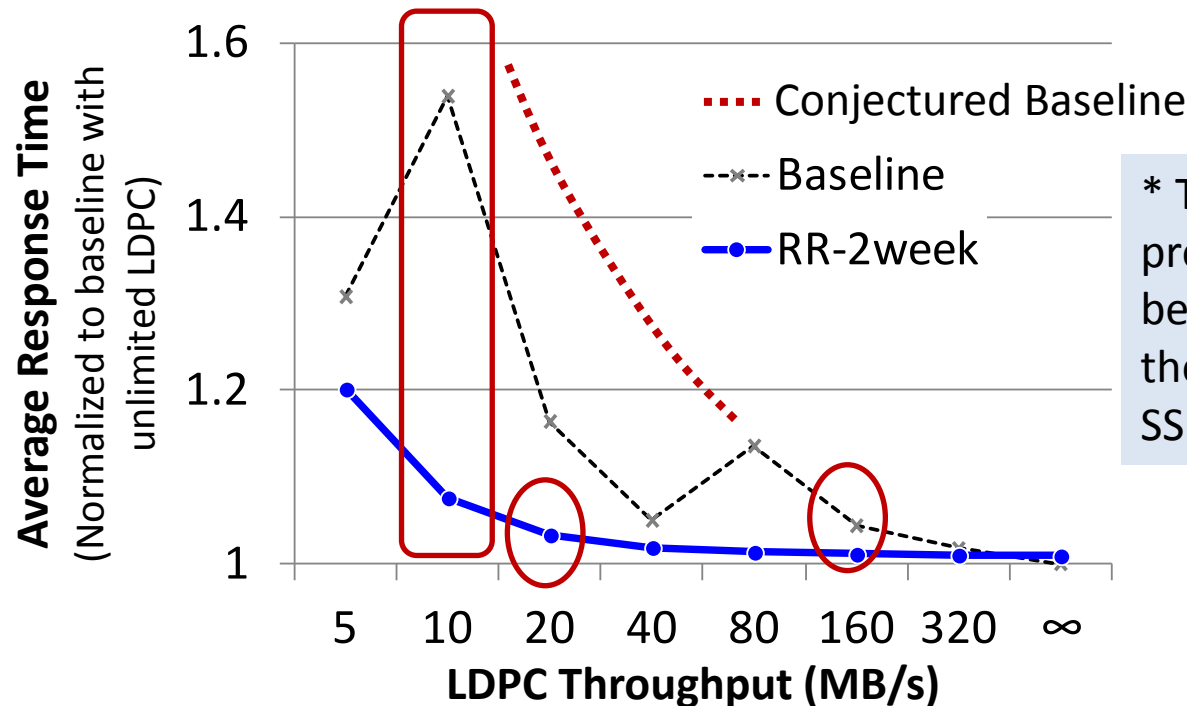
Trace Name	Dies per Disk	Exported Capacity (GB)
prn_0, proj_0, prxy_0, src1_2	16	106
src2_2	32	212
src1_0	64	423
proj_2, hd1, hd2, tpcc1, tpcc2, tpcc1_n, tpcc2_n	128	847

Improving Write Speed



- Typical 2x to 2.5x speedup
- 3.9x to 5.7x for hd1 and hd2
 - Due to long queuing time
 - Retention relaxation reduces queuing time by 5.4 to 6.1x

Improving Cost & Performance of ECCs



* The curve of the baseline presents a zigzag appearance because several traces cause the I/O queue saturation in the SSD simulator.

- SSD performance vs. various LDPC throughput
 - Under the same LDPC throughput (HW cost)
 - RR-2week outperforms the baseline
 - RR-2week approaches the ideal performance with 20MB/s LDPC

Conclusions

- First work to exploit retention relaxation for optimizing NAND Flash-based SSDs
 - Improving write speed
 - Improving the cost & performance of ECCs
- Quantitative analysis on the retention requirements of datacenter workloads
 - In most cases, 49% to 99% of writes have retention less than a week.
- Retention-aware FTL design
 - Enabling retention relaxation without hampering reliability
 - Achieving 2x to 5.7x write speedup or 8x less LDPC throughput requirements for SSDs

Thank You

