

Seminar in Computer Architecture

Meeting 3b: Example Review III

Minesh Patel

ETH Zürich

Fall 2019

3 October 2019

We will review this paper

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
"The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"
Proceedings of the 44th International Symposium on Computer Architecture (ISCA), Toronto, Canada, June 2017.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]

The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel^{§‡} Jeremie S. Kim^{‡§} Onur Mutlu^{§‡}
§ETH Zürich ‡Carnegie Mellon University

The Reach Profiler (REAPER):

Enabling the Mitigation of DRAM Retention Failures
via Profiling at Aggressive Conditions

Presented at ISCA on June '17 (Toronto, CA)

Minesh Patel Jeremie S. Kim

Onur Mutlu



SAFARI

ETH zürich

Carnegie Mellon

Summary and Technical Content

Executive Summary

- **Motivation:** DRAM refresh energy/performance overhead is high
- **Problem:** DRAM retention failure profiling is hard
 - Complicated by cells *changing* retention times *dynamically*
 - Current profiling methods are **unreliable** or **too slow**
- **Goals:**
 1. Thoroughly analyze tradeoffs in retention failure profiling
 2. Develop a **fast** and **reliable** profiling mechanism
- **Key Contributions:**
 1. **First** detailed characterization of 368 LPDDR4 DRAM chips
 2. **Reach profiling:** Profile at an **longer refresh interval** and/or **higher temperature**, where cells are more likely to fail
- **Evaluation:**
 - **2.5x** faster profiling with **99%** coverage and **50%** false positives
 - Enables longer refresh intervals that were previously unreasonable

REAPER Outline

1. DRAM Refresh Background

2. Failure Profiling Challenges

3. Current Approaches

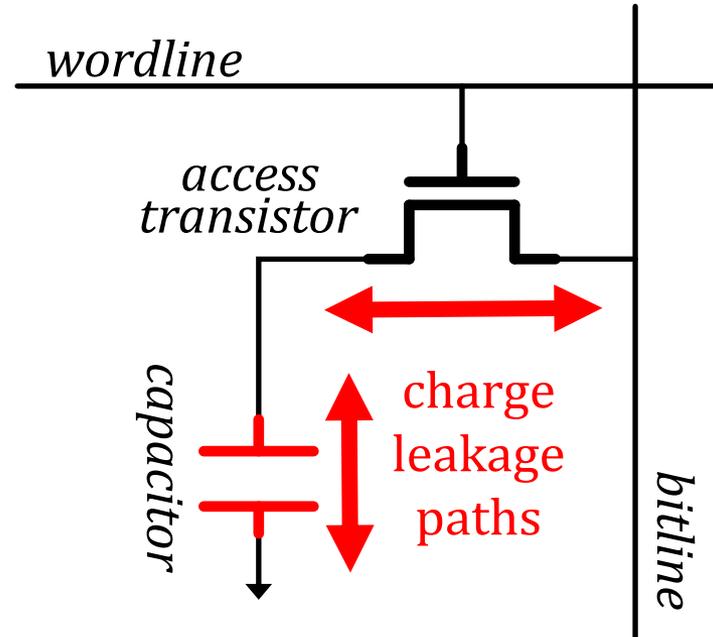
4. LPDDR4 Characterization

5. Reach Profiling

6. End-to-end Evaluation

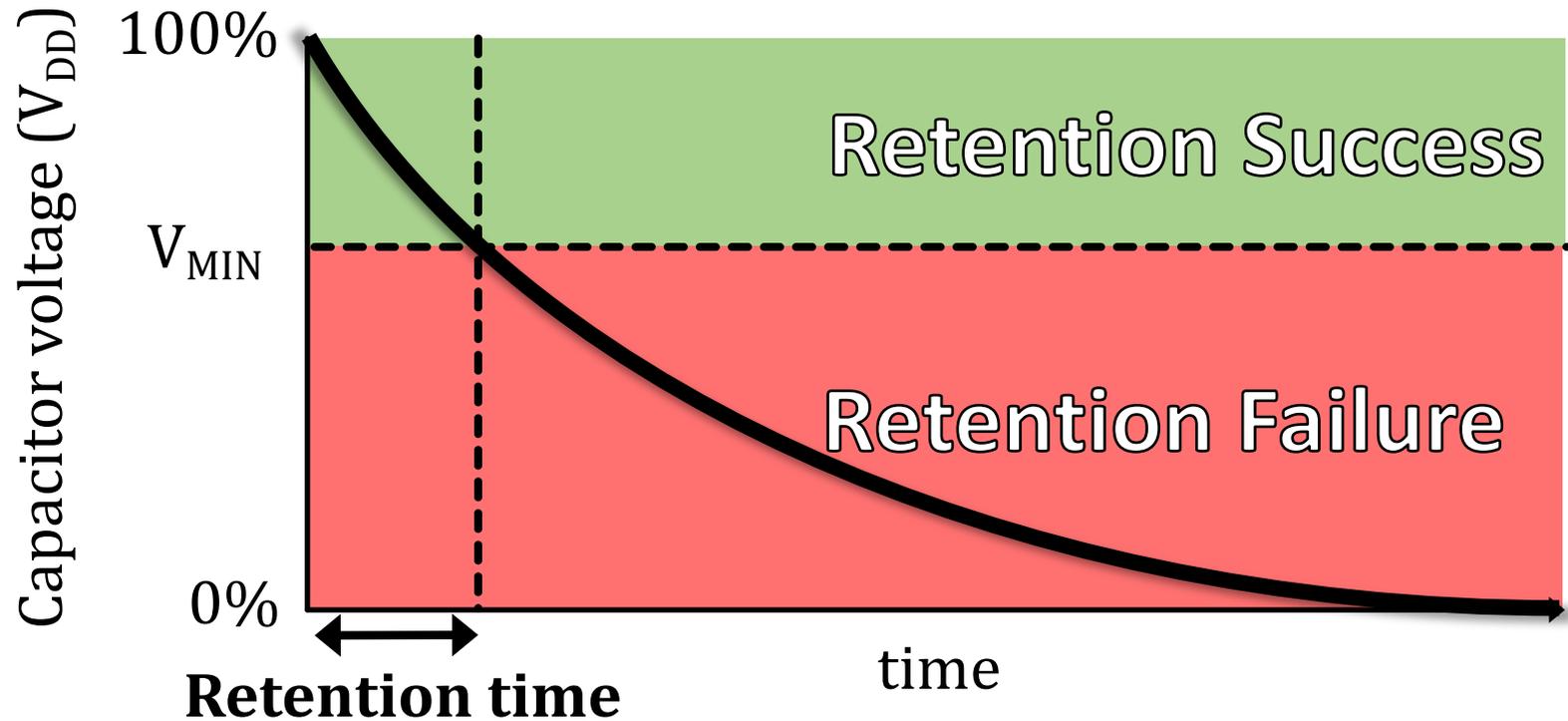
DRAM Cell Leakage

DRAM encodes information in **leaky** capacitors



Stored data is **corrupted** if too much charge leaks (i.e., the capacitor voltage degrades too far)

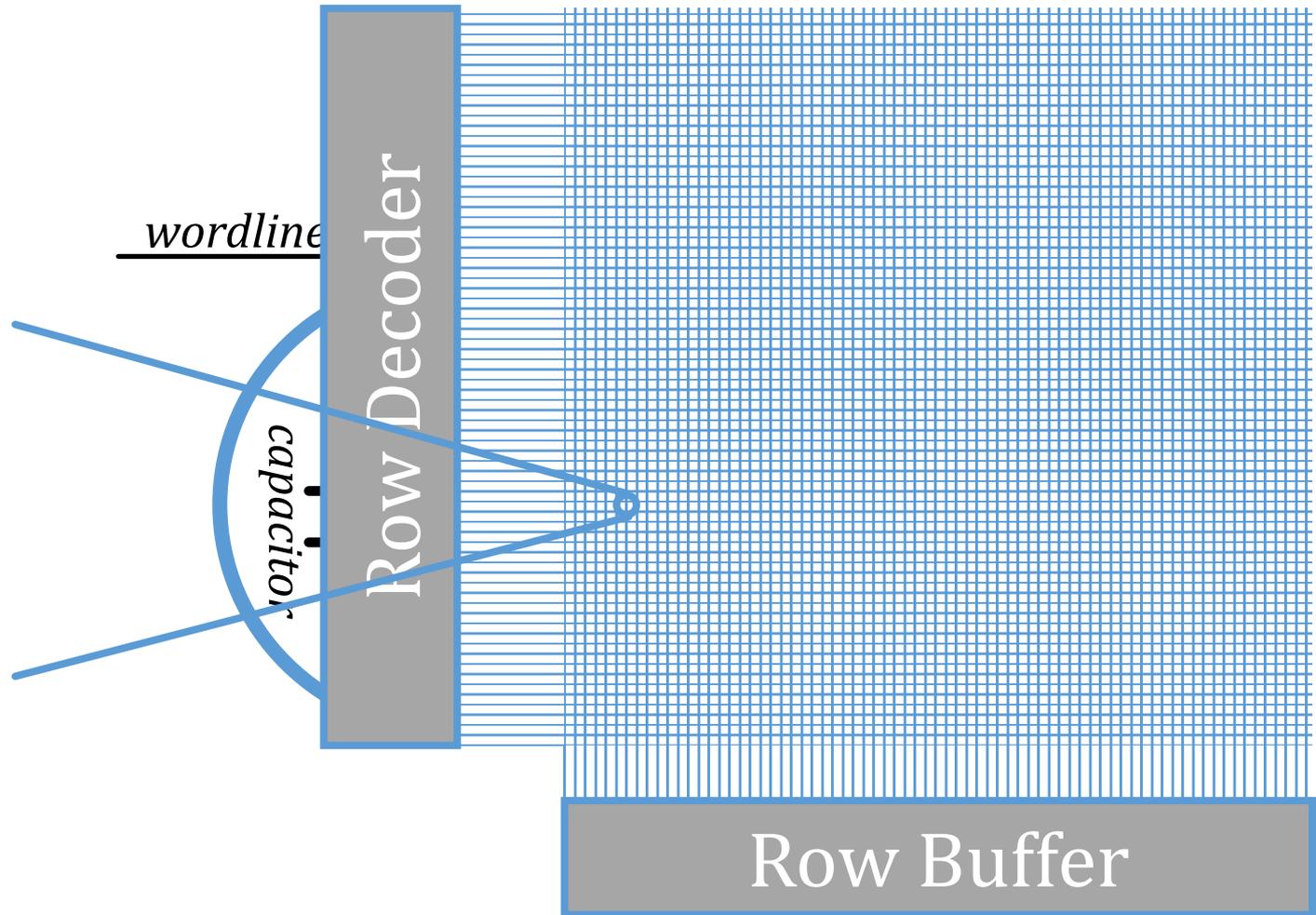
DRAM Cell Retention



Retention failure – when leakage corrupts stored data

Retention time – how long a cell holds its value

DRAM is Much More Than Just One Cell!

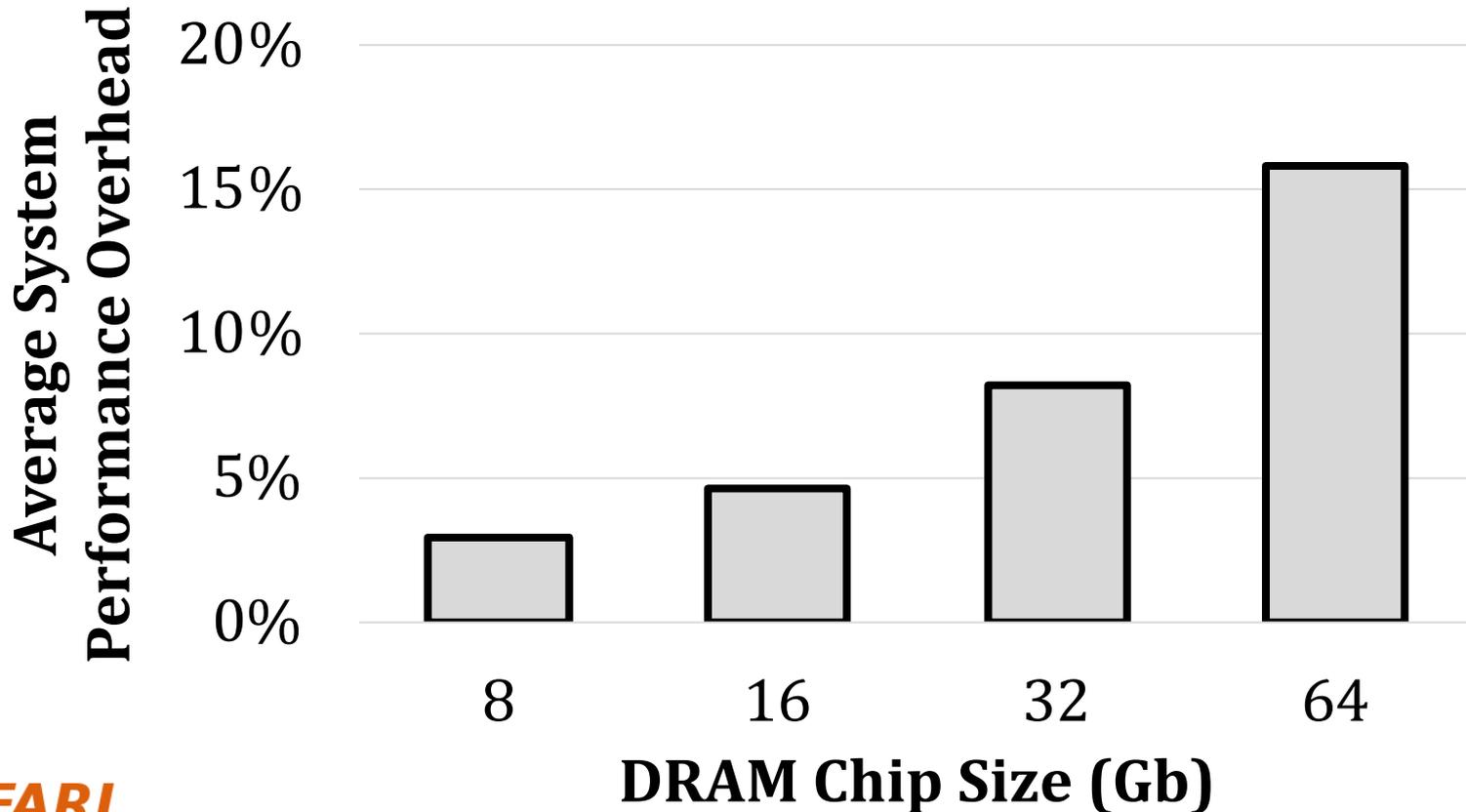


8GB DRAM = 6.4e10 cells

DRAM Refresh

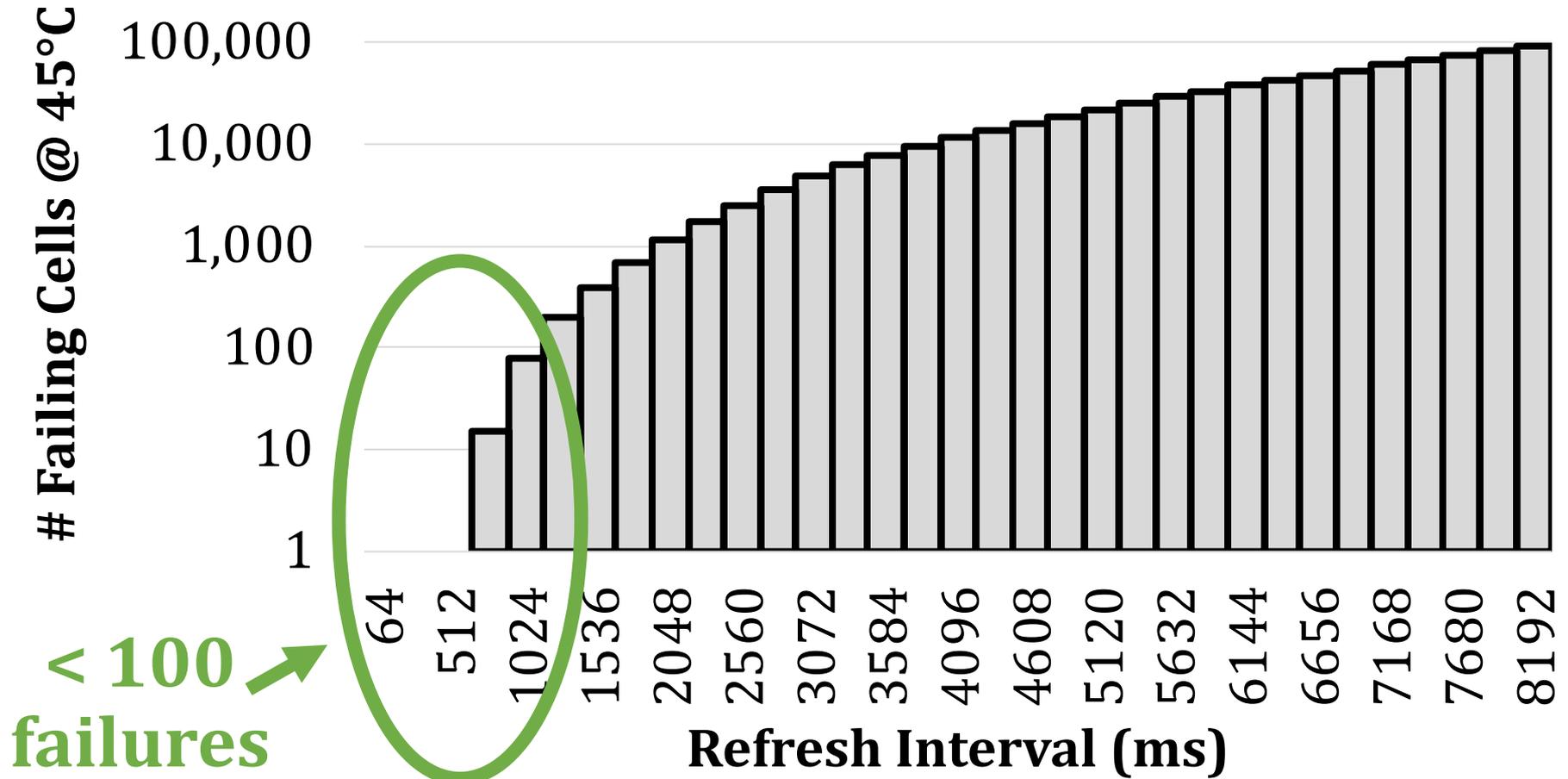
DRAM refresh periodically restores leaked charge

- *Every* cell every **refresh interval** (default = 64ms)
- Significant system **performance/energy overhead**



Decreasing Refresh Overhead

Most cells **do not fail** at a longer refresh interval



Retention Failure Mitigation

- Prior works handle these few failures to allow **reliable** operation at a longer refresh interval
 - RAIDR [Liu+, ISCA'12]

Need a **fast and reliable
profiling mechanism
to find the set of retention failures!**

- However, they **assume** they can **perfectly** identify the set of failing cells to handle

REAPER Outline

1. DRAM Refresh Background

2. Failure Profiling Challenges

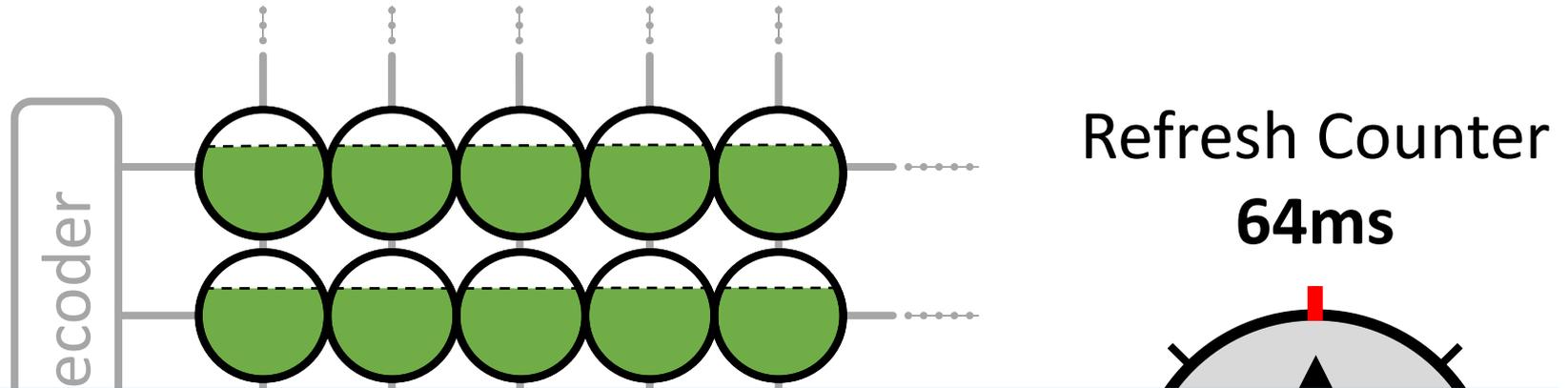
3. Current Approaches

4. LPDDR4 Characterization

5. Reach Profiling

6. End-to-end Evaluation

Idealized DRAM Refresh Operation



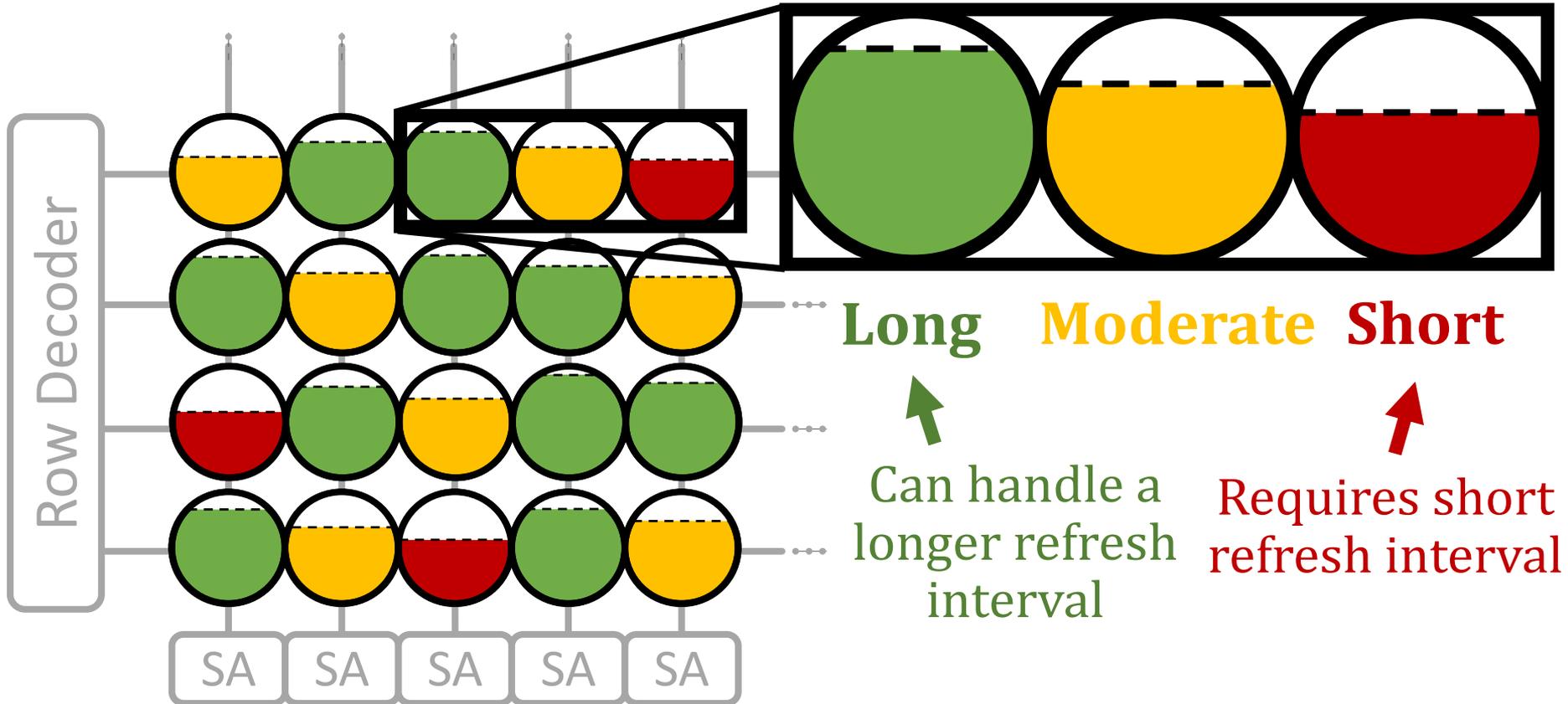
However, real DRAM cells exhibit variation in retention times

- Here, all cells have identical retention times
- All cells require the same **short** refresh interval

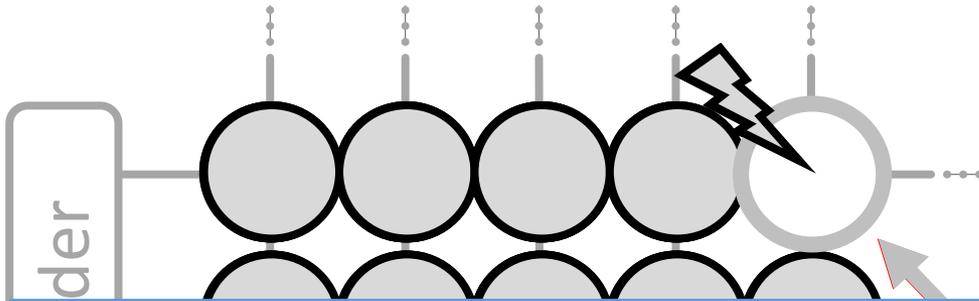
Sources of Retention Time Variation

- **Process/voltage/temperature**
- **Data pattern dependence (DPD)**
 - Retention times **change with data** in cells/neighbors
 - e.g., all 1's vs. all 0's
- **Variable retention time (VRT)**
 - Retention time changes **randomly (unpredictably)**
 - Due to a combination of various circuit effects

Heterogeneous Retention Times



Extended Refresh Interval (128ms)



How can we **quickly** and **reliably** determine the failing cells at an increased refresh interval T ?

SA SA SA SA SA

Long Moderate Short

REAPER Outline

1. DRAM Refresh Background

2. Failure Profiling Challenges

3. Current Approaches

4. Individual Bit Failures

5. Reach Profiling

6. End-to-end Evaluation

Solution #1: ECC-Scrubbing

Key idea: leverage error-correcting codes (ECC) by periodically accessing all ECC words to continuously detect new failures
(e.g., AVATAR [Qureshi+, DSN'15])

• Pros

- **Simple:** read accesses to all DRAM locations
- **Low overhead:** DRAM is available during scrubs

• Cons

- **Unreliable:** does not account for changes in data pattern, which changes cell retention times
 - Can potentially miss failures between scrubs

Solution #2: Brute-force Profiling

Key idea: for {N data patterns} * {M test rounds}:

- 1) Write data pattern to DRAM
- 2) Wait for the refresh interval

Our goals:

- 1) study profiling tradeoffs
- 2) develop a **fast** and **reliable** profiling mechanism

- **Slow:** many test rounds required for reliability
- **High overhead:** DRAM is unavailable for a long time

REAPER Outline

1. DRAM Refresh Background

2. Failure Profiling Challenges

3. Current Approaches

4. LPDDR4 Characterization

5. Reach Profiling

6. End-to-end Evaluation

Experimental Infrastructure

- **368 2y-nm LPDDR4 DRAM chips**
 - 4Gb chip size
 - From 3 major DRAM manufacturers

- **Thermally controlled testing chamber**
 - Ambient temperature range: $\{40^{\circ}\text{C} - 55^{\circ}\text{C}\} \pm 0.25^{\circ}\text{C}$
 - DRAM temperature is held at 15°C above ambient

LPDDR4 Studies

1. Temperature
2. Data Pattern Dependence
3. Retention Time Distributions
- 4. Variable Retention Time**
- 5. Individual Cell Characterization**

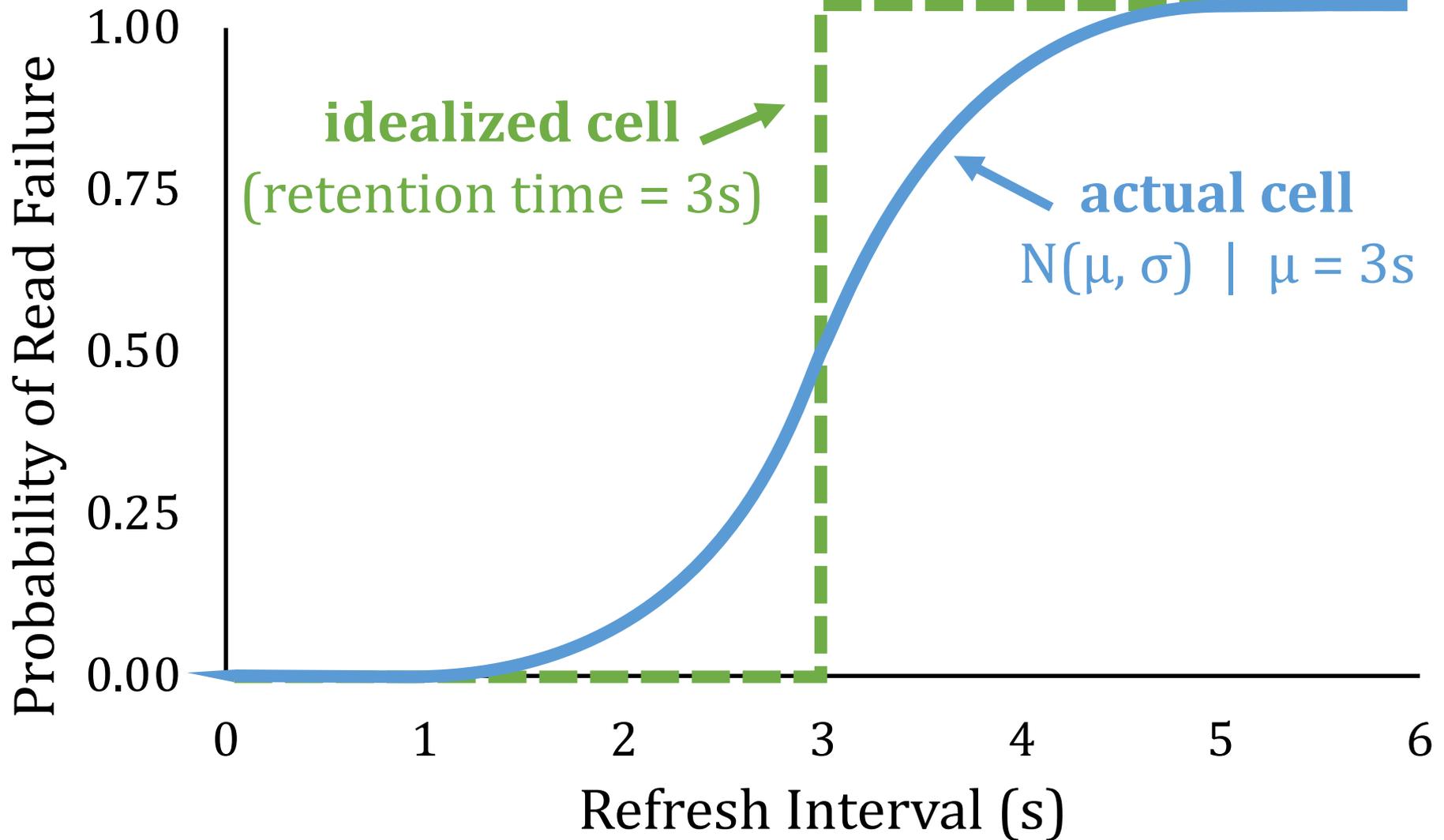
Long-term Continuous Profiling



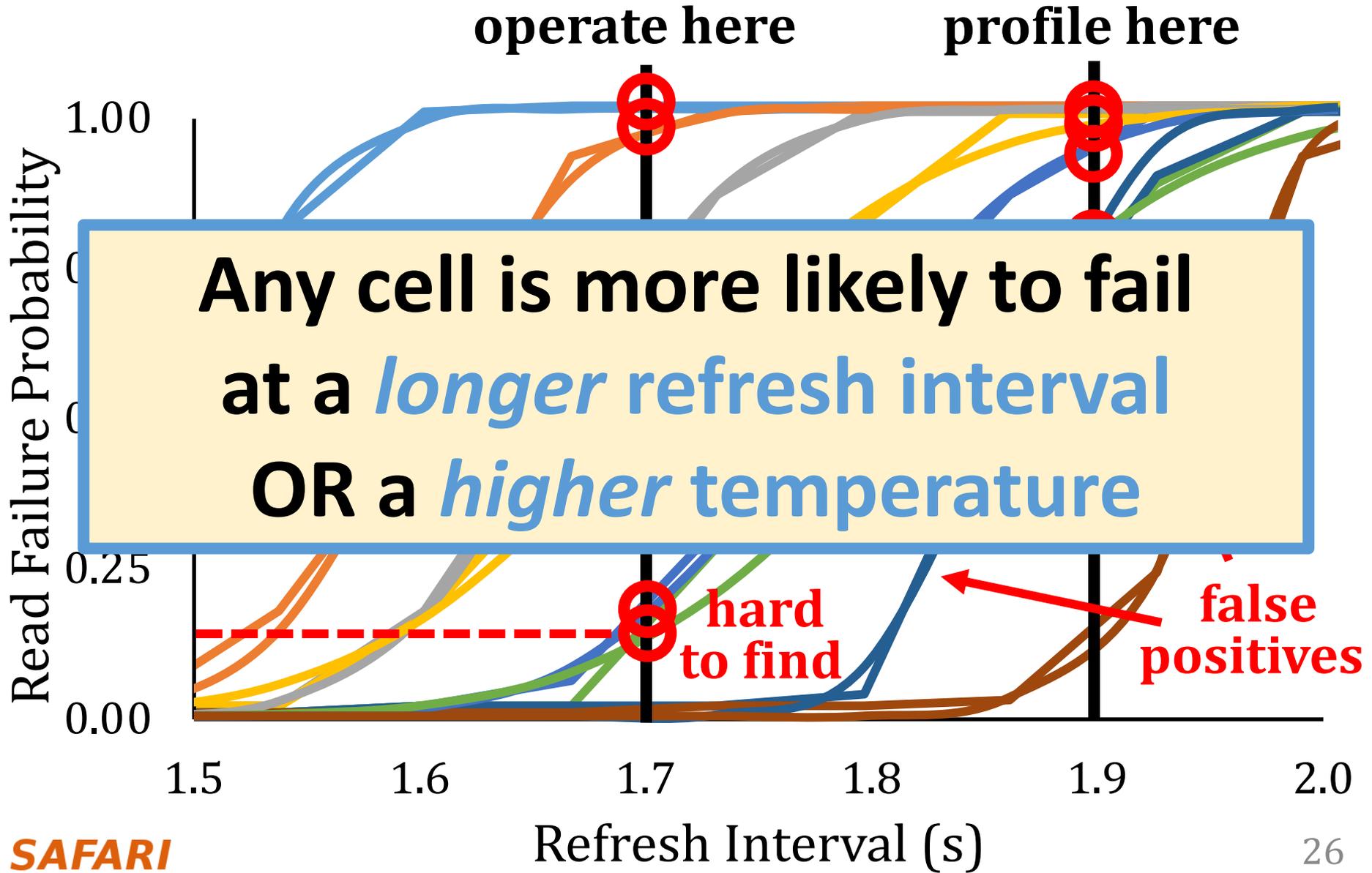
**Error correction codes (ECC)
and online profiling are *necessary*
to manage new failing cells**

- New failing cells continue to appear over time
 - Attributed to **variable retention time (VRT)**
- The set of failing cells changes over time

Single-cell Failure Probability (Cartoon)



Single-cell Failure Probability (Real)



REAPER Outline

1. DRAM Refresh Background

2. Failure Profiling Challenges

3. Current Approaches

4. LPDDR4 Characterization

5. Reach Profiling

6. End-to-end Evaluation

Reach Profiling

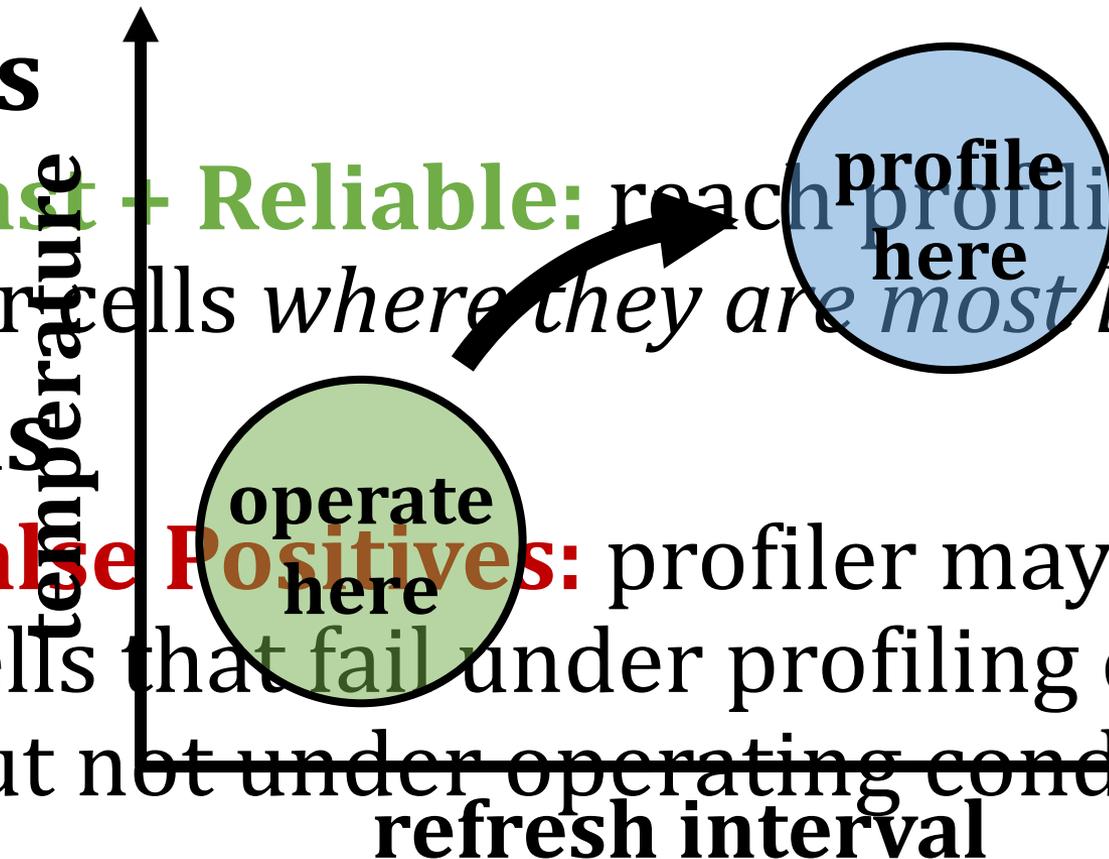
Key idea: profile at a *longer refresh interval* and/or a *higher temperature*

- **Pros**

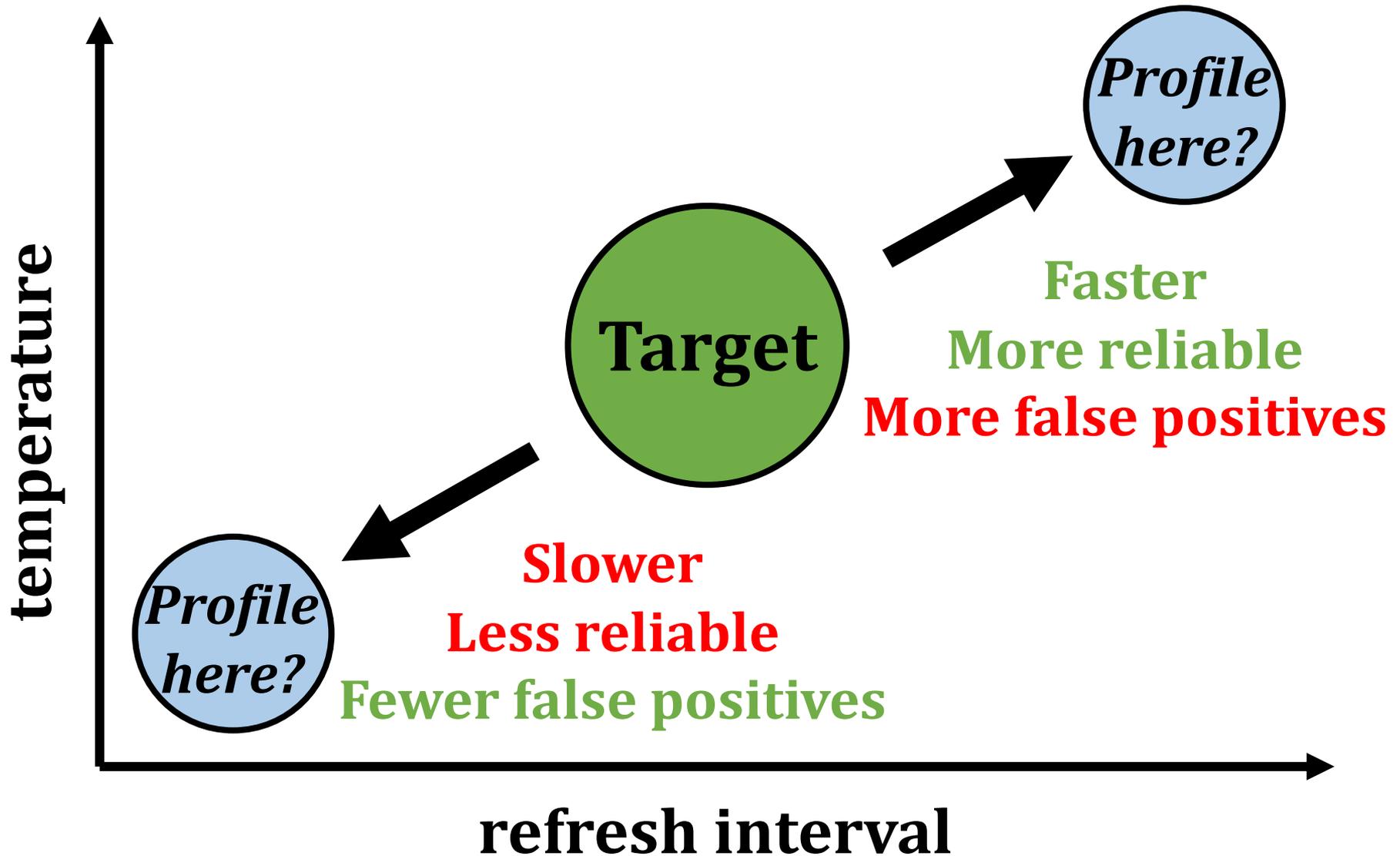
- **Fast + Reliable:** reach profiling searches for cells *where they are most likely to fail*

- **Cons**

- **False Positives:** profiler may identify cells that fail under profiling conditions, but not under operating conditions



A Complex Tradeoff Space



Towards an Implementation

Reach profiling is a **general methodology**

3 key questions for an implementation:

What are desirable profiling conditions?

How often should the system profile?

What information does the profiler need?

Three Key Profiling Metrics

1. **Runtime:** how long profiling takes
2. **Coverage:** portion of all possible failures discovered by profiling

We explore how these three metrics change under **many** different profiling conditions

Q1: Desirable Profiling Conditions

- Similar trends across chips *and* vendors!
- For 99% coverage, we find on average:
 - **2.5x speedup** by profiling at **+250ms** at a cost of a **50% false positive rate**
 - **>3.5x speedup** by profiling at **+ >500ms** at a cost of a **>75% false positive rate**
- More examples and detail in the paper

Q2: How Often to Profile

- Estimation using a **probabilistic model**
 - Can use our empirical data for estimates
 - Details are in the paper

- e.g., Need to reprofile every **2.3 days** for a:
 - 2GB ECC DRAM
 - 1024ms refresh interval at 45°C
 - Profiling with 99% coverage

Q3: Necessary Information

- **The cost of handling identified failures**
 - Determines how many errors we can mitigate
 - e.g., error-correction codes (ECC)
- **Empirical per-chip characterization data**
 - Used to reliably estimate profiling parameters
 - Details are in the paper

REAPER Outline

1. DRAM Refresh Background

2. Failure Profiling Challenges

3. Current Approaches

4. LPDDR4 Characterization

5. Reach Profiling

6. End-to-end Evaluation

Our Mechanism: REAPER

- Simple implementation of reach profiling
- Pessimistic assumptions
 - Whole system pauses during profiling
 - Firmware executes profiling routine
 - Exclusive DRAM access
 - Only manipulates refresh interval, not temperature

Evaluation Methodology

- Simulators
 - **Performance:** Ramulator [Kim+, CAL'15]
 - **Energy:** DRAMPower [Chandrasekar+, DSD'11]
- Configuration
 - 4-core (4GHz), 8MB LLC
 - LPDDR4-3200, 4 channels, 1 rank/channel
- Workloads
 - 20 random 4-core benchmark mixes
 - SPEC CPU2006 benchmark suite

Simulated End-to-end Performance

 Brute-force profiling  REAPER  Ideal profiling

On average, REAPER enables:
16.3% system performance improvement
36.4% DRAM power reduction



REAPER enables longer refresh intervals,
which are unreasonable
using brute-force profiling

refresh
rarely

refresh
often

Other Analyses in the Paper

- **Detailed LPDDR4 characterization data**
 - Temperature dependence effects
 - Retention time distributions
 - Data pattern dependence
 - Variable retention time
 - Individual cell failure distributions
- **Profiling tradeoff space characterization**
 - Runtime, coverage, and false positive rate
 - Temperature and refresh interval
- **Probabilistic model for tolerable failure rates**
- **Detailed results for end-to-end evaluations**

Summary

Motivation: DRAM refresh performance/energy overhead is high

Problem: Current retention failure profiling is unreliable or slow

Goals:

1. Thoroughly analyze profiling tradeoffs
2. Develop a **fast** and **reliable** profiling mechanism

Key Contributions:

1. **First** detailed characterization of 368 LPDDR4 DRAM chips
2. **Reach profiler:** Profile at a **longer refresh interval** and/or **higher temperature**, where cells are more likely to fail

Evaluation:

- **2.5x** faster profiling with **99%** coverage and **50%** false positives
- REAPER enables **16.3% system performance improvement** and **36.4% DRAM power reduction**
- Enables longer refresh intervals that were previously unreasonable

Analysis: Strengths

Strengths

- Tackles a difficult and important problem
- Intuitive key idea
- Enables many *error mitigation mechanisms* that rely on an efficient retention failure profiling mechanism
- Reach profiling is an effective solution
 - Improves both performance and coverage over brute-force profiling
- First paper to present system-level characterization data using LPDDR4 devices
 - Aggregate error rates/distributions from 368 devices
 - Tradeoff space surrounding retention error profiling

Analysis: Weaknesses

Weaknesses and Limitations

- Needs hardware modifications for profiling
 - Firmware in the memory controller
- Requires ability to modify refresh timings/temperature
- Profiling is still expensive
 - REAPER *infrequently* runs *expensive* profiling jobs
 - Requires data migration while profiling
- Probabilistic – not guaranteed to identify every weak bit
- Evaluation could be more thorough
 - Does not account for DRAM idle time
 - Generic SPEC benchmarks
 - Does not incorporate overheads of real error-mitigation mechanisms
 - Reliability is analyzed analytically rather than demonstrated in experiment

Thoughts and Ideas

Continuing Work

- Maybe we can identify ‘weak’ cells faster by:
 - Manipulating other DRAM access characteristics (e.g., data, timings)
 - Combining with a different profiling approach (e.g., ECC scrubbing)
- We can improve REAPER performance costs by:
 - Intelligently migrating data during/before profiling
 - Implementing fine-grained refresh commands
- Demonstrate REAPER in practice on a real system

Takeaways

Key Takeaways

- Retention failures are more likely at increased *temperature* and *refresh intervals*
- Reach profiling: profile where errors are *likely to occur*
 - Improves *speed* and *failure coverage*
- Online profiling is necessary to deal with VRT errors
- Uses insights from real devices to propose reach profiling
- Enables many different error mitigation mechanisms

Questions and General Discussion

Open Discussion Questions (1/2)

- How else can we improve profiling speed/coverage?
- If not, can we better tolerate the overheads of profiling?
- For what types of systems is periodic profiling okay?
- How might this affect the workloads we study?
 - Standardized benchmark suite (e.g., SPEC 2006)?
 - More representative workloads?

Open Discussion Questions (2/2)

- Is retention error profiling now a solved problem?
- How important is retention error profiling going forward?
- Assume that profiling is a solved problem – can we devise a better error mitigation mechanism?
- Can we find a way to profile for other types of errors?
 - RowHammer
 - Variable retention time (VRT) errors

Seminar in Computer Architecture

Meeting 3b: Example Review III

Minesh Patel

ETH Zürich

Fall 2019

3 October 2019

Backup Slides

Temperature Relationship

- Well-fitting exponential relationship:

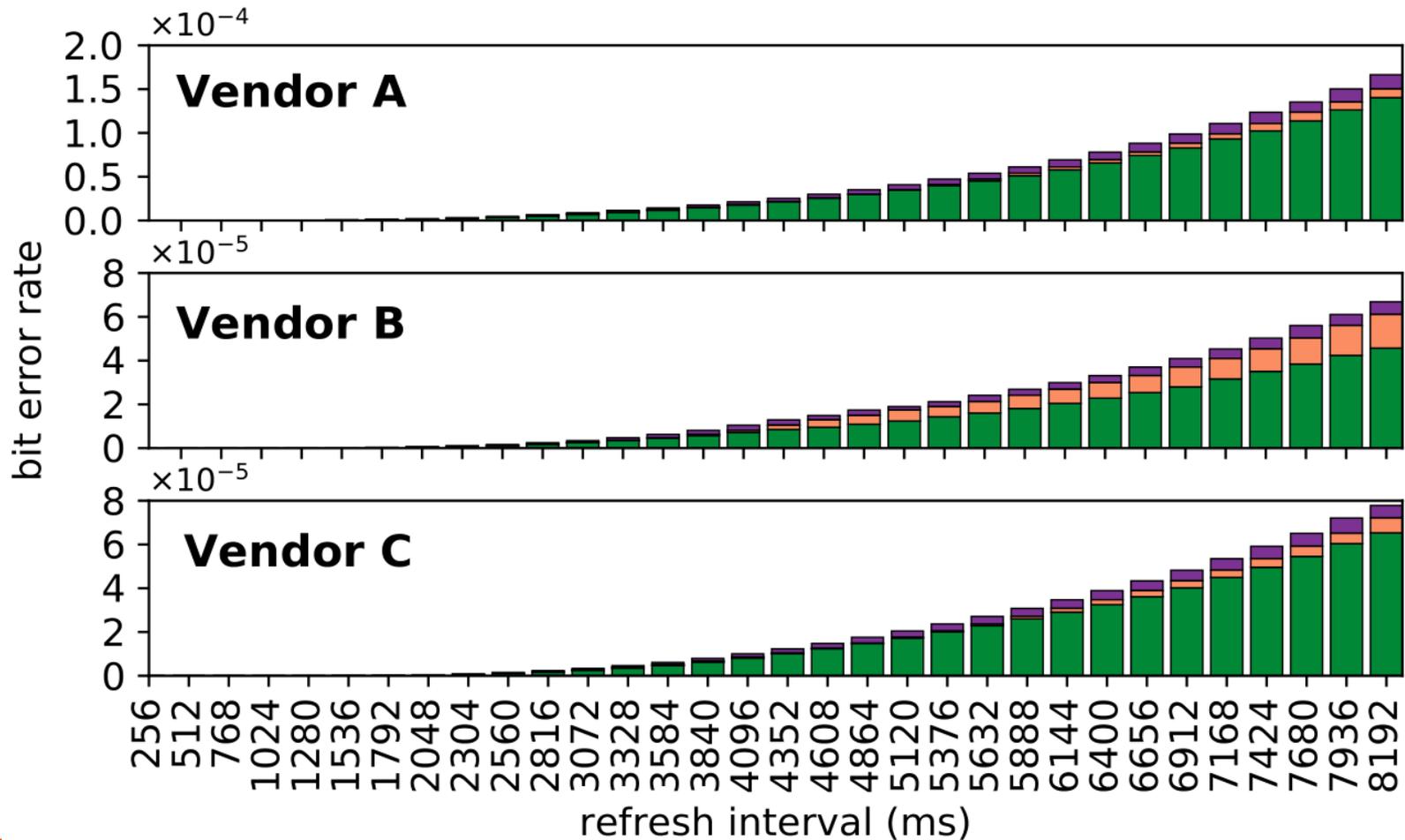
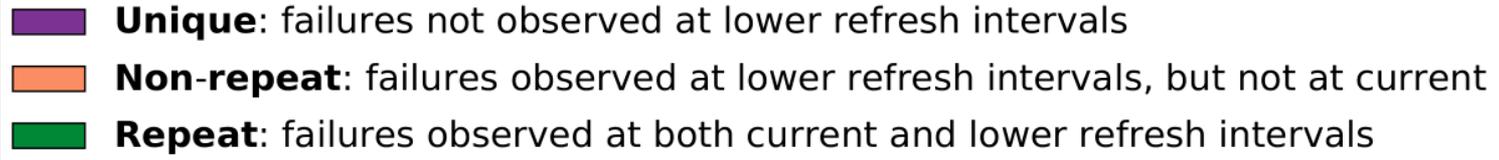
$$R_A \propto e^{0.22\Delta T}$$

$$R_B \propto e^{0.20\Delta T}$$

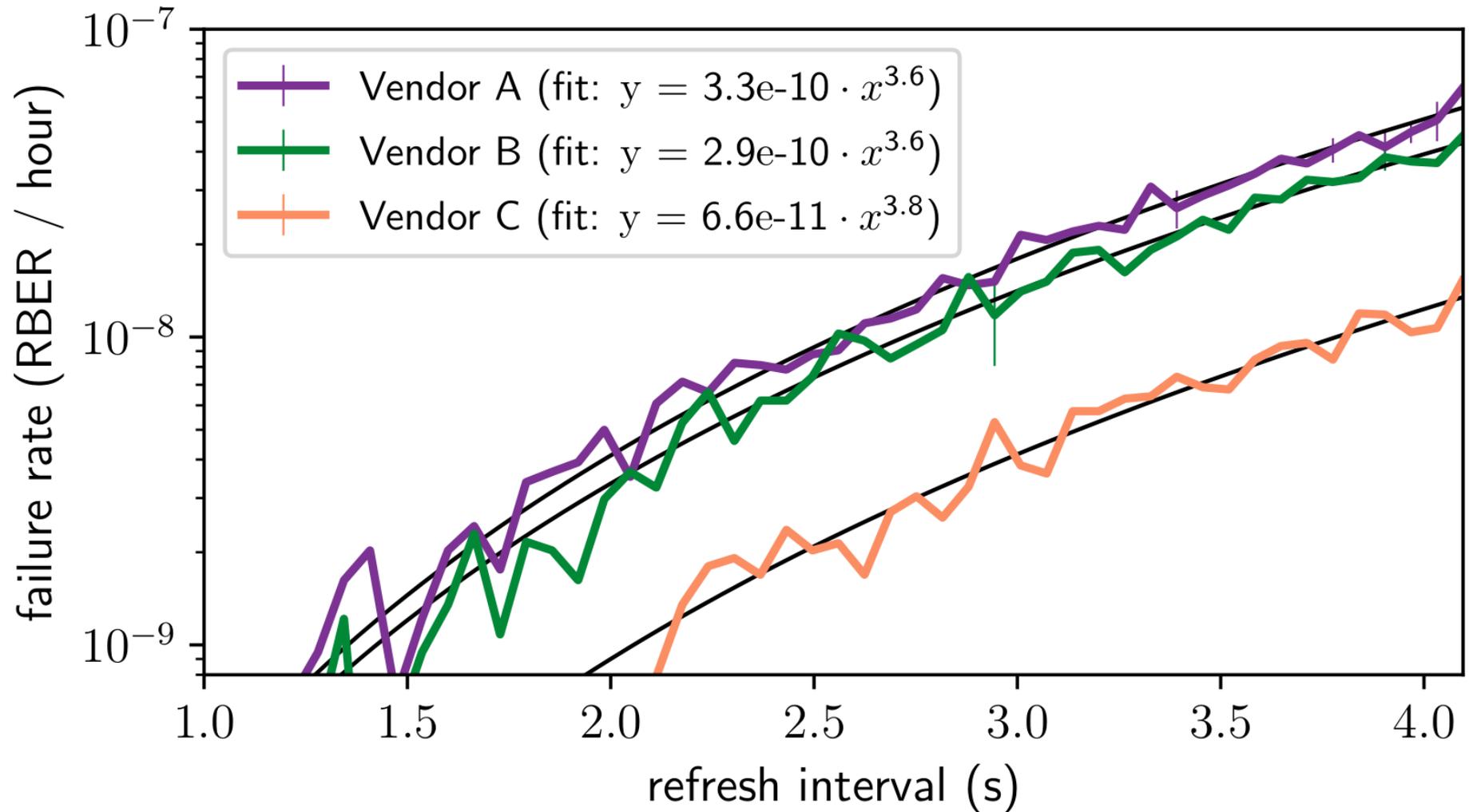
$$R_C \propto e^{0.26\Delta T}$$

- E.g., 10°C ~ 10x more failures

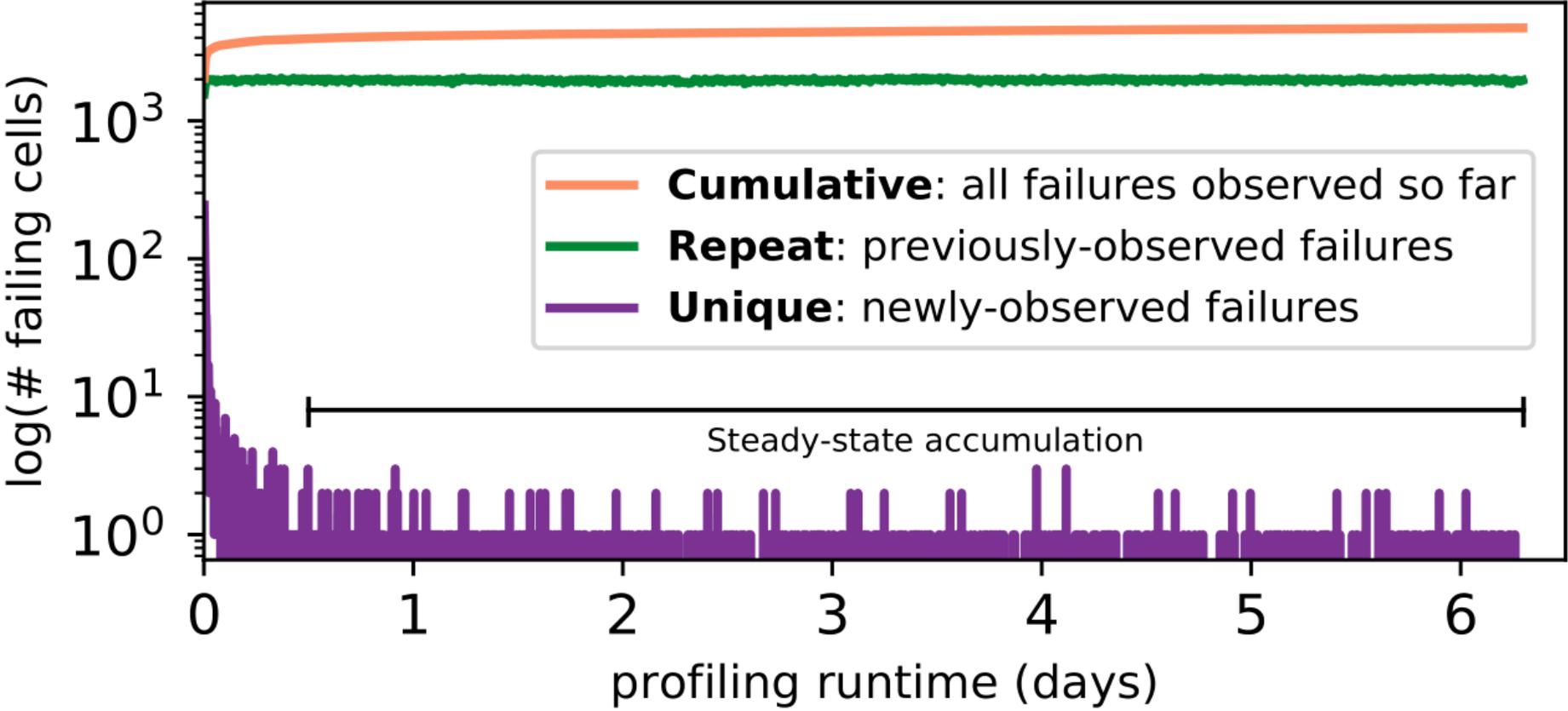
Retention Failures @ 45°C



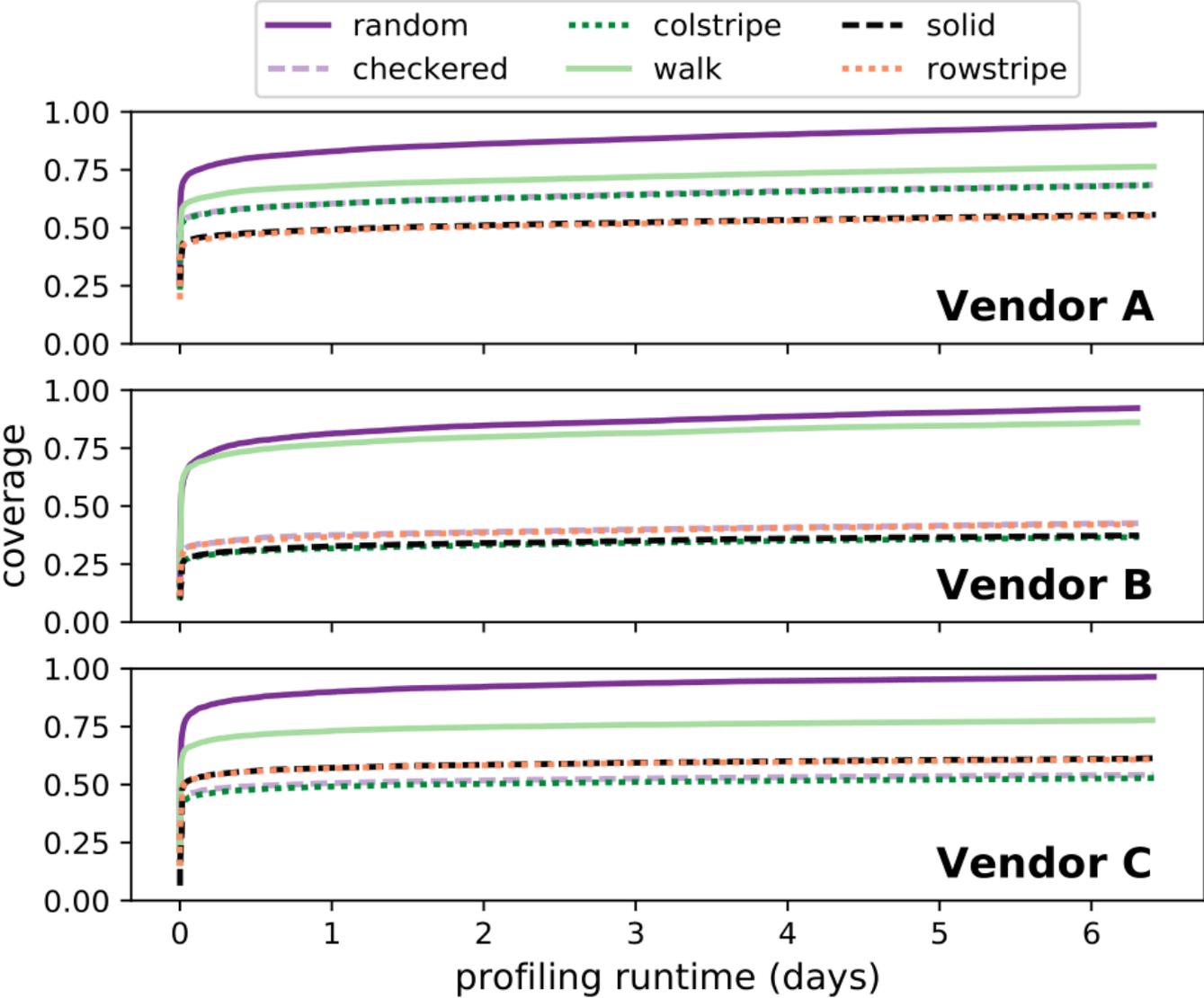
VRT Failure Accumulation Rate



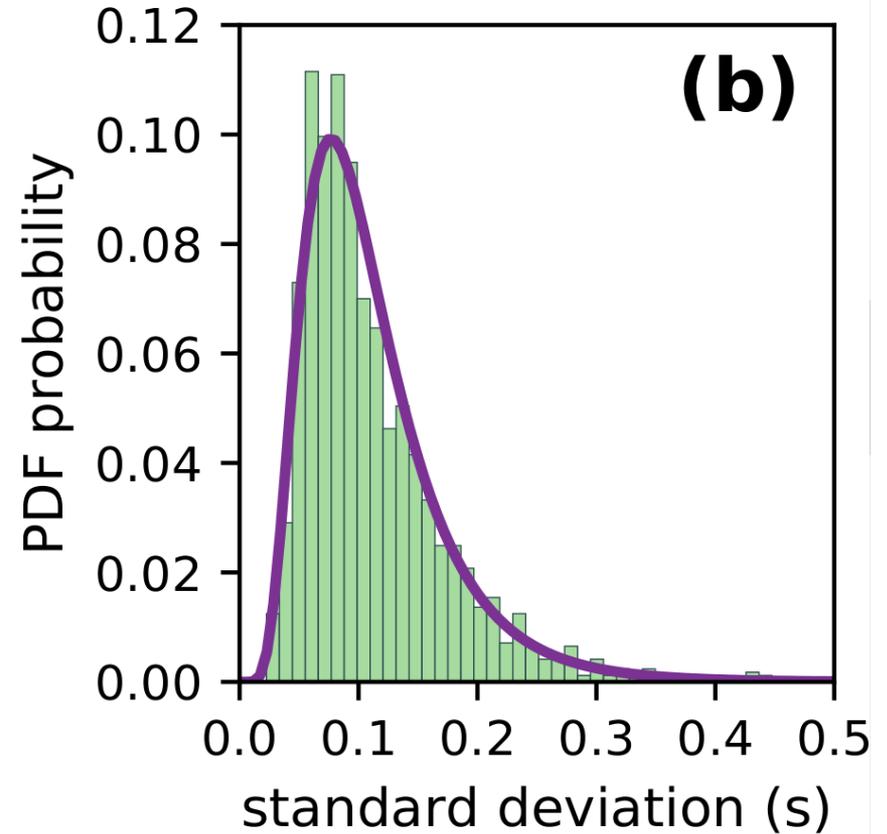
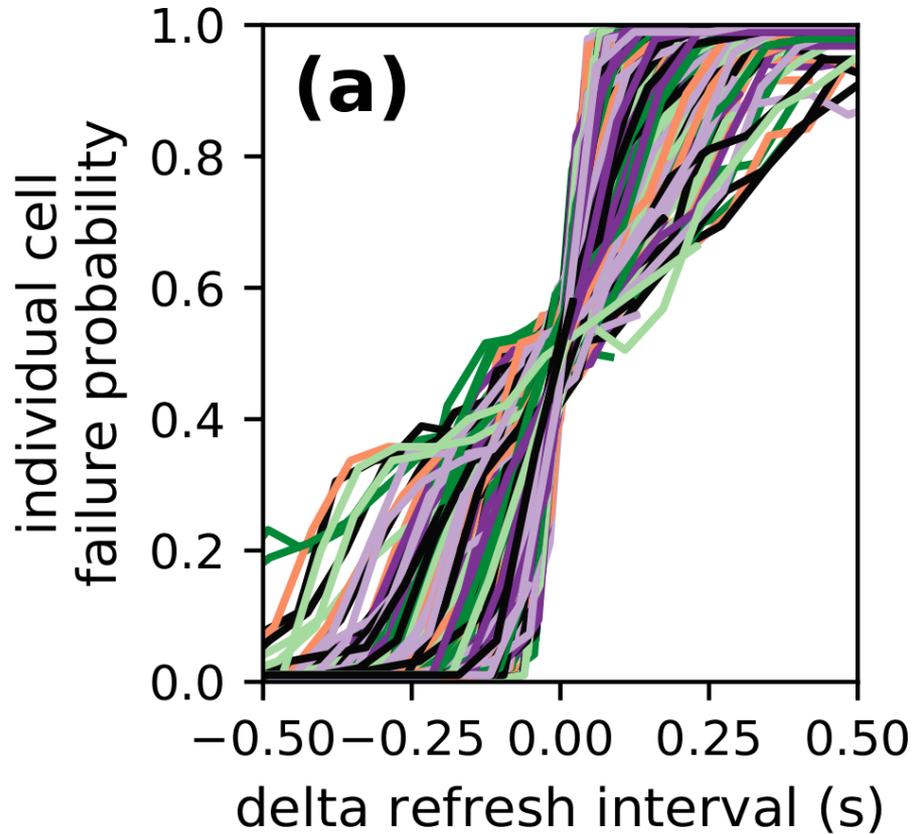
800 Rounds of Profiling @ 2048ms, 45°C



800 Rounds of Profiling @ 2048ms, 45°C

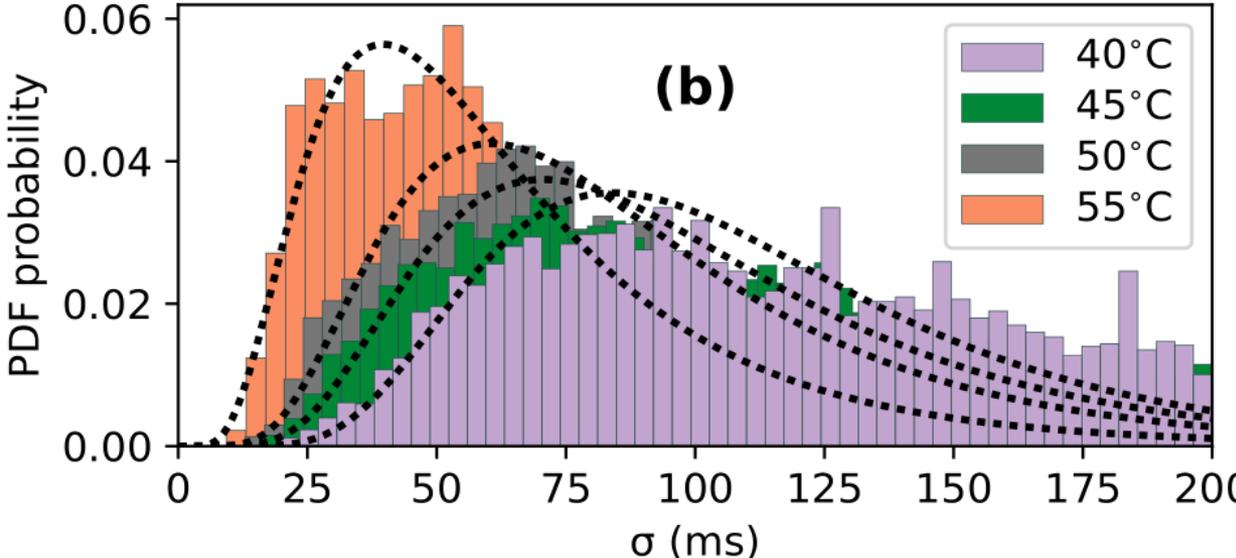
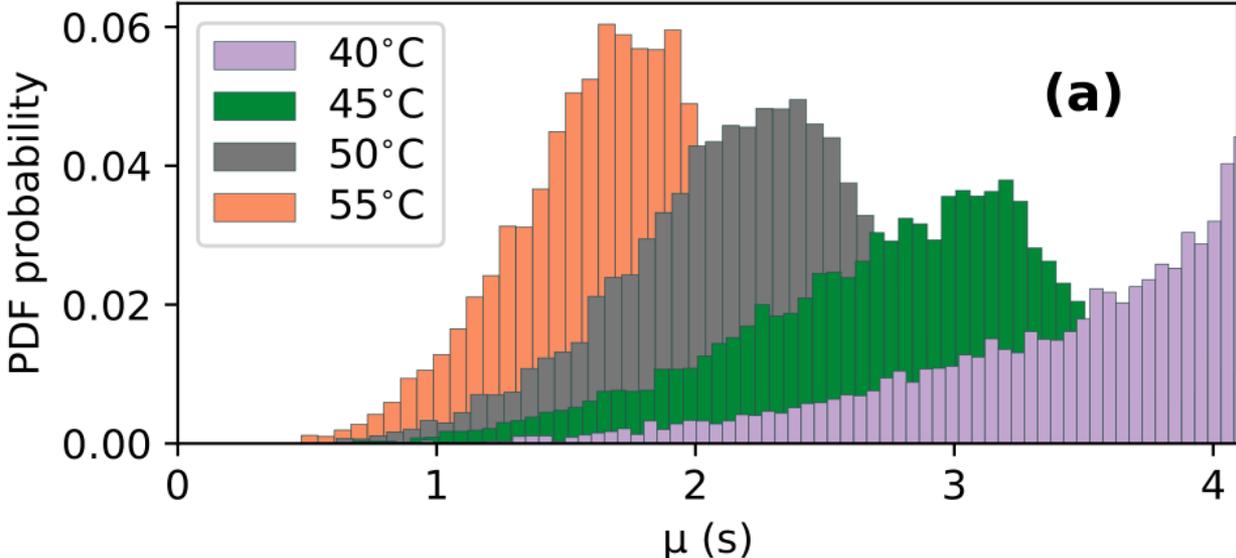


Individual Cell Failure Probabilities

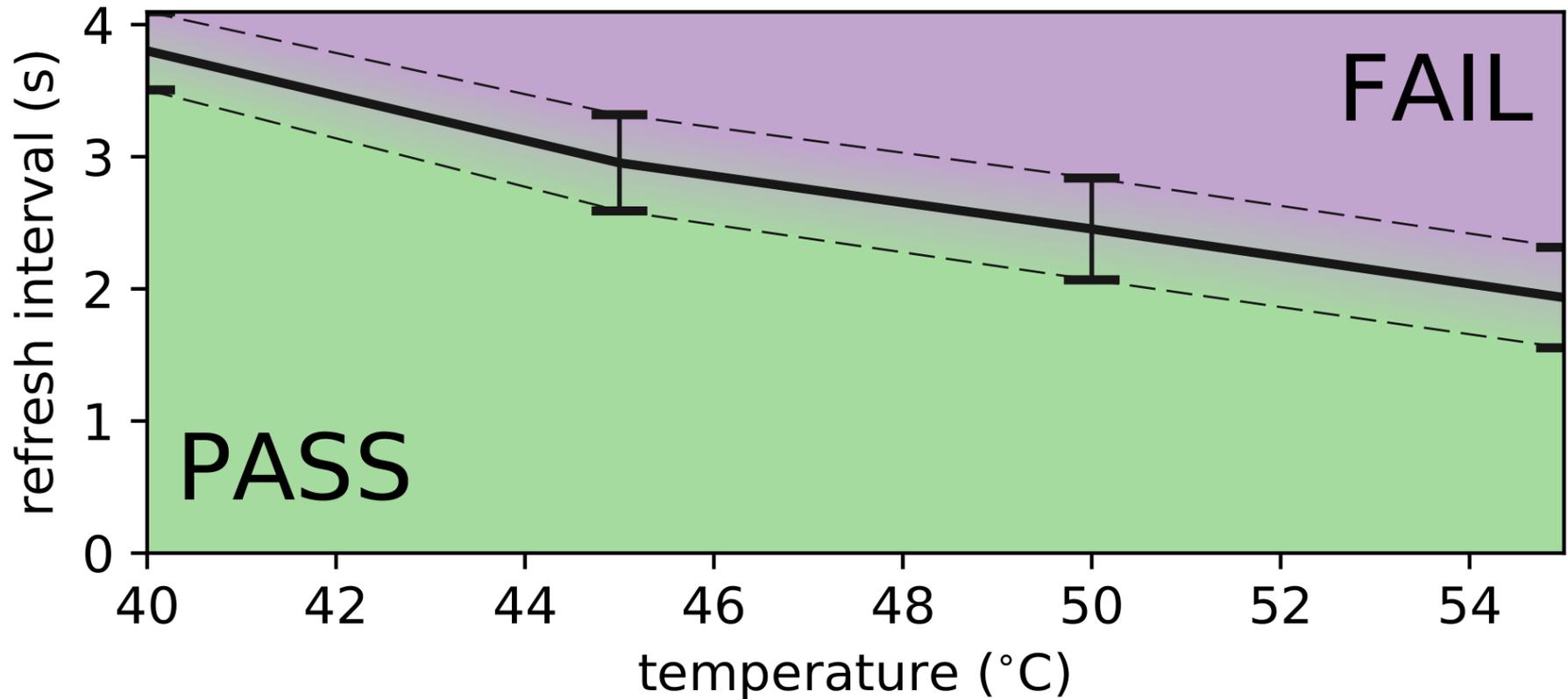


- Single representative chip of Vendor B at 40° C
- Refresh intervals ranging from 64ms to 4096ms

Individual Cell Failure Distributions



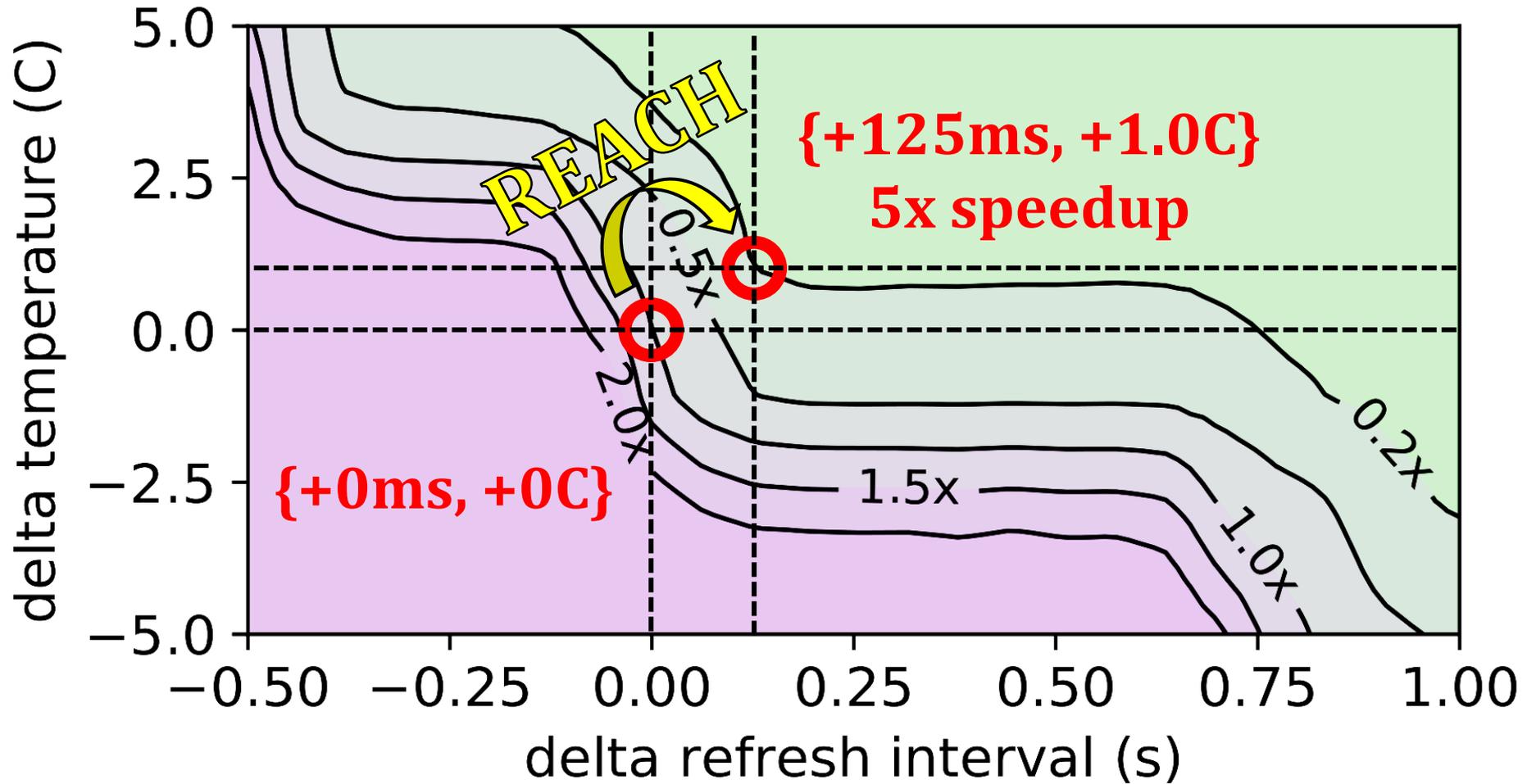
Single-cell Failures With Temperature



- Single representative chip of Vendor B
- {mean, std} for cells between 64ms and 4096ms

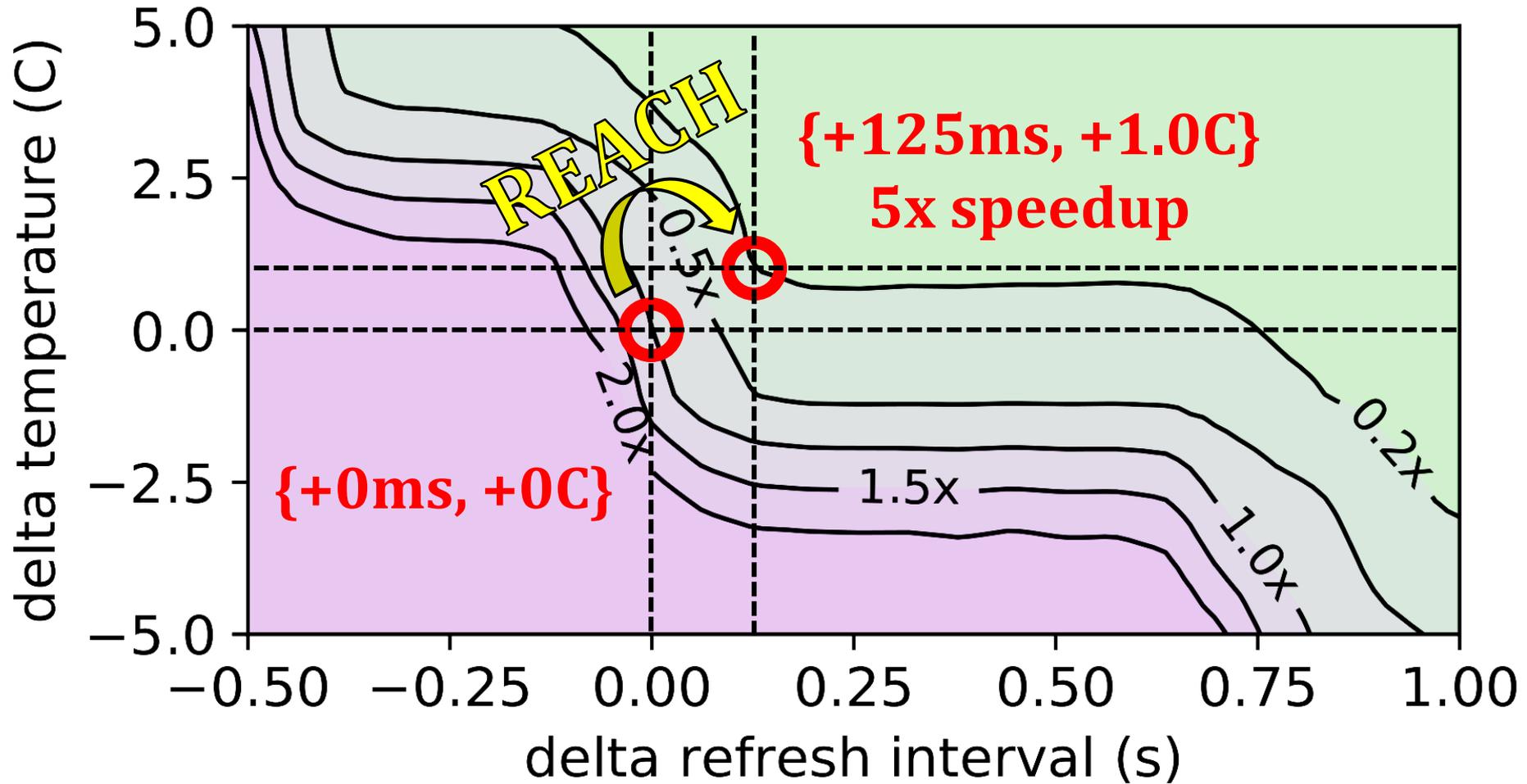
Example experimental analysis

Runtime for 95% coverage of {2048ms, 50C}



Example experimental analysis

Runtime for 95% coverage of {2048ms, 50C}



Q2: How often must we re-profile?

Raw Bit Error Rate (RBER) – ratio of actual failing DRAM cells

Uncorrectable Bit Error Rate (UBER) – error rate observed by the system

We can compute the *maximum tolerable RBER* for a given UBER and ECC strength

	No ECC	SECDED	ECC-2
Max RBER for UBER = 10^{-15}	1e-15	3.8e-9	6.9e-7
Equivalent # bits in 2GB DRAM	< 1	65	12,000

Without ECC, we can't tolerate even one failure!

Probabilistic Failure Model

k = ECC strength (e.g., SECDED = 1)

w = ECC word size (e.g., SECDED 64/72 word = 72 bits)

$$\text{UBER} = \frac{1}{w} \text{P}[\text{uncorrectable error in a } w\text{-bit ECC word}]$$

$$\text{UBER} = \frac{1}{w} \sum_{n=k+1}^w \text{P}[n\text{-bit failure in a } w\text{-bit ECC word}]$$

$$\text{UBER}(k = 0) = \frac{1}{64} \sum_{n=1}^{64} \text{P}[n\text{-bit failure in a 64-bit ECC word}]$$

$$\text{UBER}(k = 1) = \frac{1}{72} \sum_{n=2}^{72} \text{P}[n\text{-bit failure in a 72-bit ECC word}]$$

Probabilistic Failure Model

k = ECC strength (e.g., SECDED = 1)

w = ECC word size (e.g., SECDED 64/72 word = 72 bits)

$$\text{UBER} = \frac{1}{w} \sum_{n=k+1}^w \text{P}[n\text{-bit failure in a } w\text{-bit ECC word}]$$

Binomial distribution of errors in an n -bit word:

$$\text{P}[n\text{-bit failure in a } w\text{-bit ECC word}] = \binom{w}{n} R^n (1 - R)^{w-n}$$

$$\text{UBER} = \frac{1}{w} \sum_{n=k+1}^w \binom{w}{n} R^n (1 - R)^{w-n}$$

Allowable Errors

- Tolerable **RBBER** and tolerable **number of bit errors** for $UBER = 10^{-15}$ across different ECC strengths for selected DRAM sizes

		ECC Strength		
		No ECC	SECDED	ECC-2
Tolerable RBBER		1.0e-15	3.8e-9	6.9e-7
# Tolerable Bit Errors	512MB	4.3e-6	16.3	3.0e3
	1GB	8.6e-6	32.6	5.9e+3
	2GB	1.7e-5	65.3	1.2e+4
	4GB	3.4e-5	130.6	2.4e+4
	8GB	6.9e-5	261.1	4.7e+4

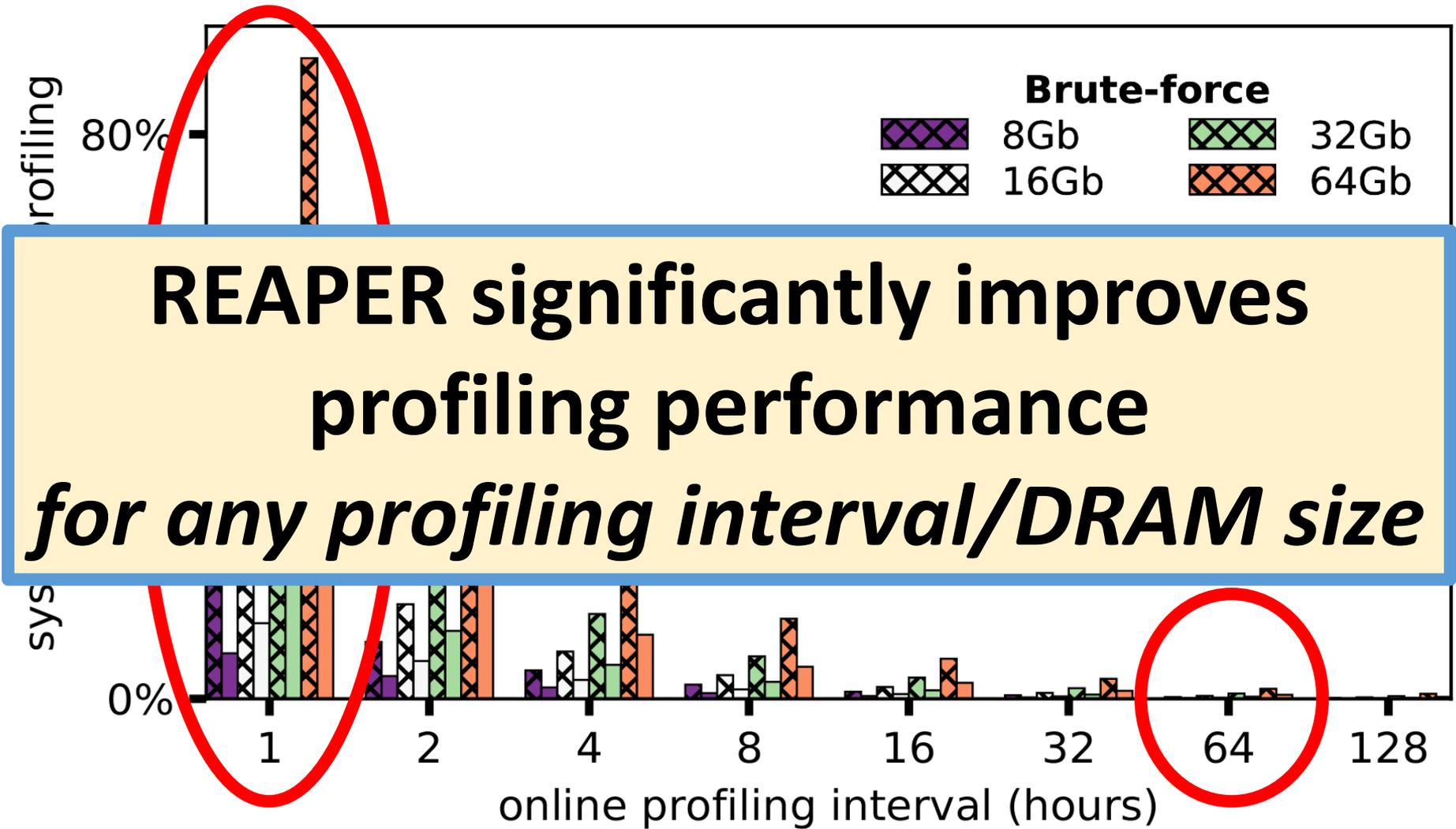
Tradeoff Space Exploration

- We explore:
 - **368** LPDDR4 chips
 - Refresh intervals from **64ms - 4096ms**
 - Temperatures from **40C - 55C**

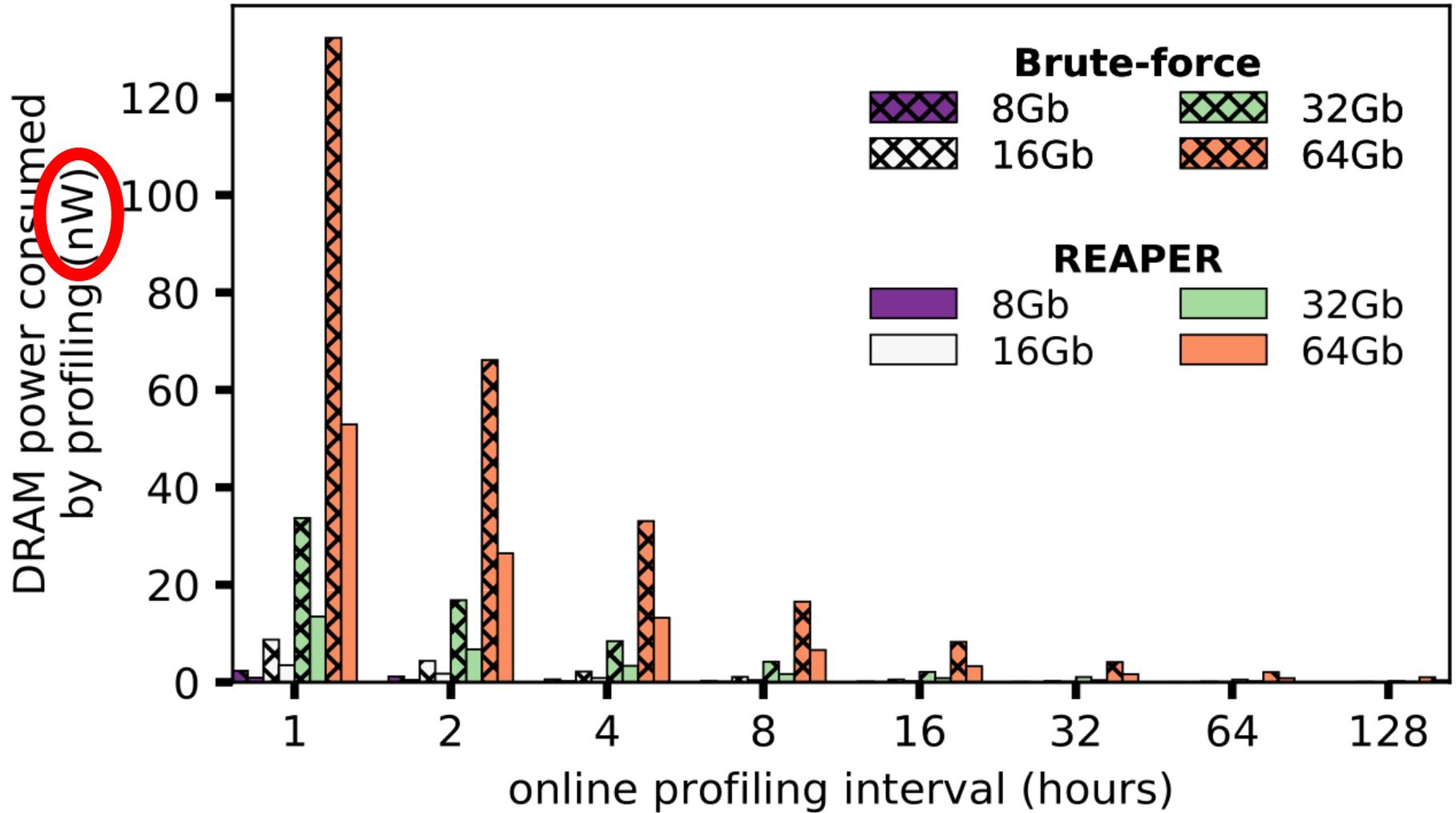
Evaluation Configuration Details

Processor	4 cores, 4GHz clock frequency, 3-wide issue, 8 MSHRs/core, 128-entry instruction window
Last-level Cache	64B cache line, 16-way, 8MB cache size
Memory Controller	64-entry read/write request queues, FR-FCFS scheduling policy [83, 102], open/closed row policy [50, 51] for single/multi-core
DRAM	LPDDR4-3200 [37], 4 channels, 1 rank, 8 banks/rank, 32K-256k rows/bank, 2KB row buffer

Profiling Performance Overhead



Profiling Energy Overhead



End-to-end Performance/Energy

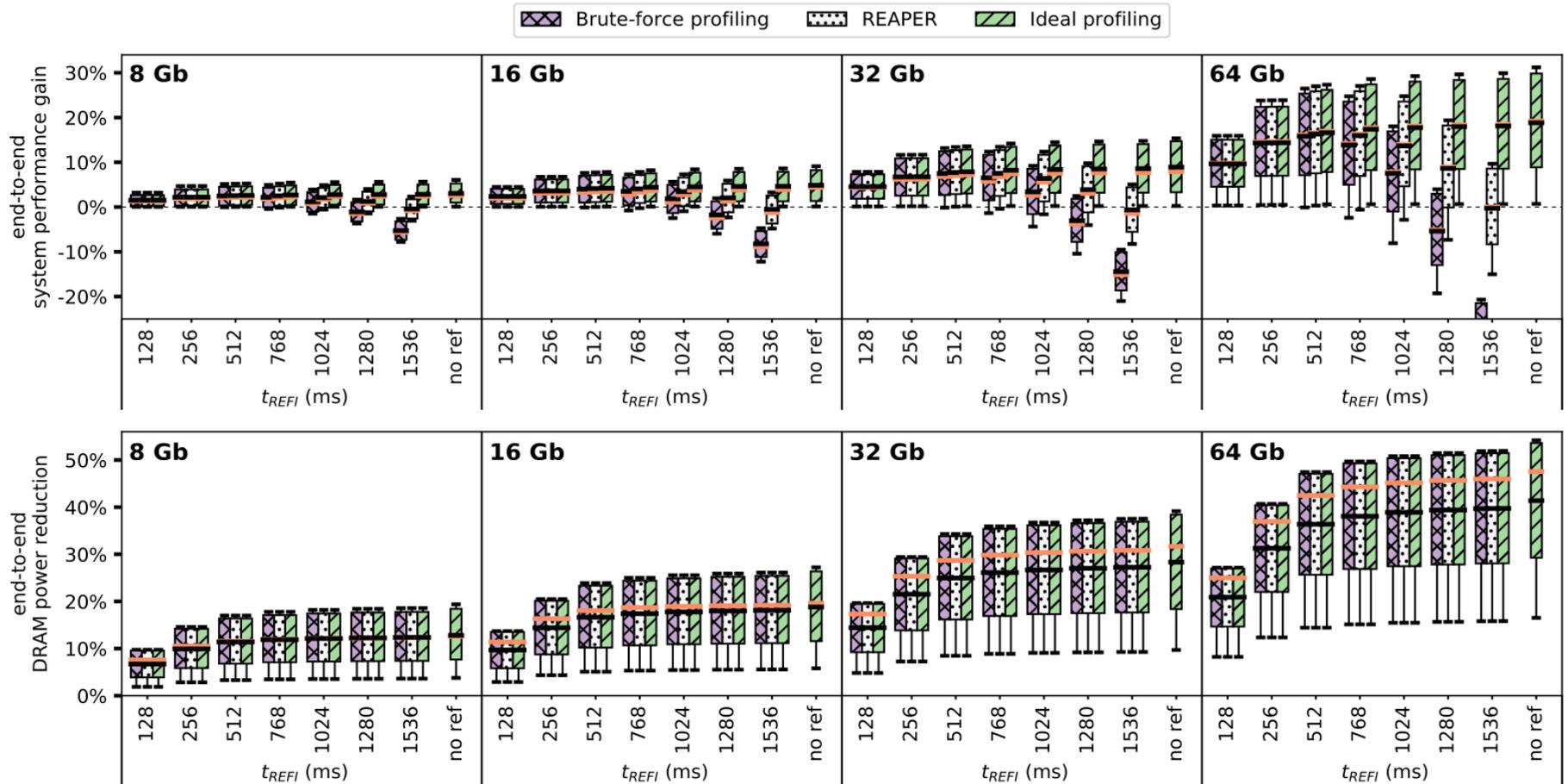


Figure 13: Simulated end-to-end system performance improvement (top) and DRAM power reduction (bottom) over 20 heterogeneous 4-core workloads for different refresh intervals at 45°C, taking into account online profiling frequency and profiling overhead.

Algorithm 1: Brute-Force Profiling Algorithm

```
1 PROFILE(target  $t_{REF}$ , num_iterations):
2     failed_cells = []
3     for it  $\leftarrow$  {1 to num_iterations}:
4         for dp  $\in$  data_patterns:
5             write_DRAM(dp)
6             disable_refresh()
7             wait(target  $t_{REF}$ )
8             enable_refresh()
9             this_iteration_failures  $\leftarrow$  get_DRAM_errors()
10            failed_cells.add(this_iteration_failures)
11    return failed_cells
```

Evaluation Caveat

- Profiling tradeoff space is **enormous**
 - Temperature
 - Refresh interval
 - Desired coverage
 - etc.

- Results depend on specific choices
 - We're making **worst-case assumptions**
 - Other choices could be even better

With a Mitigation Mechanism

- REAPER can be combined with most mitigation mechanisms
 - **RAIDR** [Liu+, ISCA'12]
 - **SECRET** [Lin+, ICCD'12]
 - **ArchShield** [Nair+, ISCA'13]
 - **DTail** [Cui+, SC'14]
 - **AVATAR** [Qureshi+, DSN'15]
 - ...
- REAPER periodically profiles, and mitigation takes care of discovered failures

Exploring the Tradeoff Space



We explore *in detail* the effect of **different reach conditions** on

- 1) Runtime
- 2) Coverage
- 3) False positives

for **different target conditions**