Computer Architecture

Lecture 6c: The Reach Profiler (REAPER)

Minesh Patel
ETH Zürich

Fall 2018

4 October 2018

The Reach Profiler (REAPER):

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

Minesh Patel **Jeremie S. Kim**

Onur Mutlu

SAFARI

ETH Zürich

Carnegie Mellon

Executive Summary

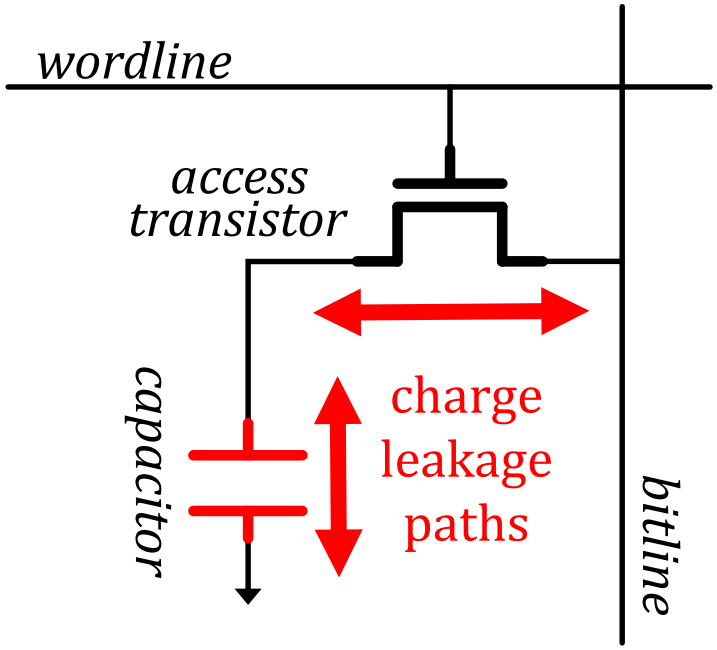
- **Motivation:** DRAM refresh energy/performance overhead is high
- **Problem**: DRAM retention failure profiling is hard
  - Complicated by cells *changing* retention times *dynamically*
  - Current profiling methods are **unreliable** or **too slow**
- **Goals**:
  1. Thoroughly analyze tradeoffs 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

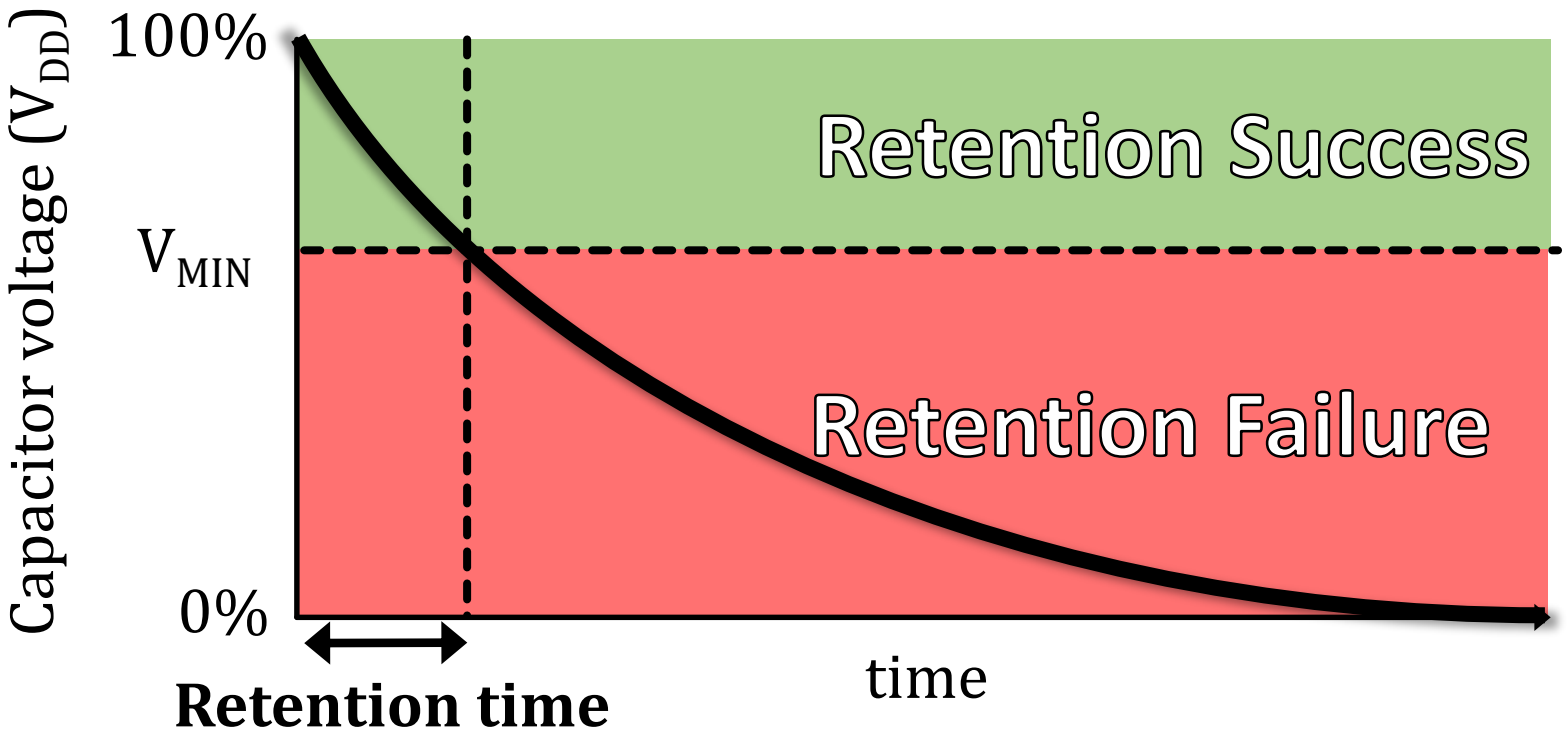
DRAM Cell Leakage

DRAM encodes information in leaky capacitors



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

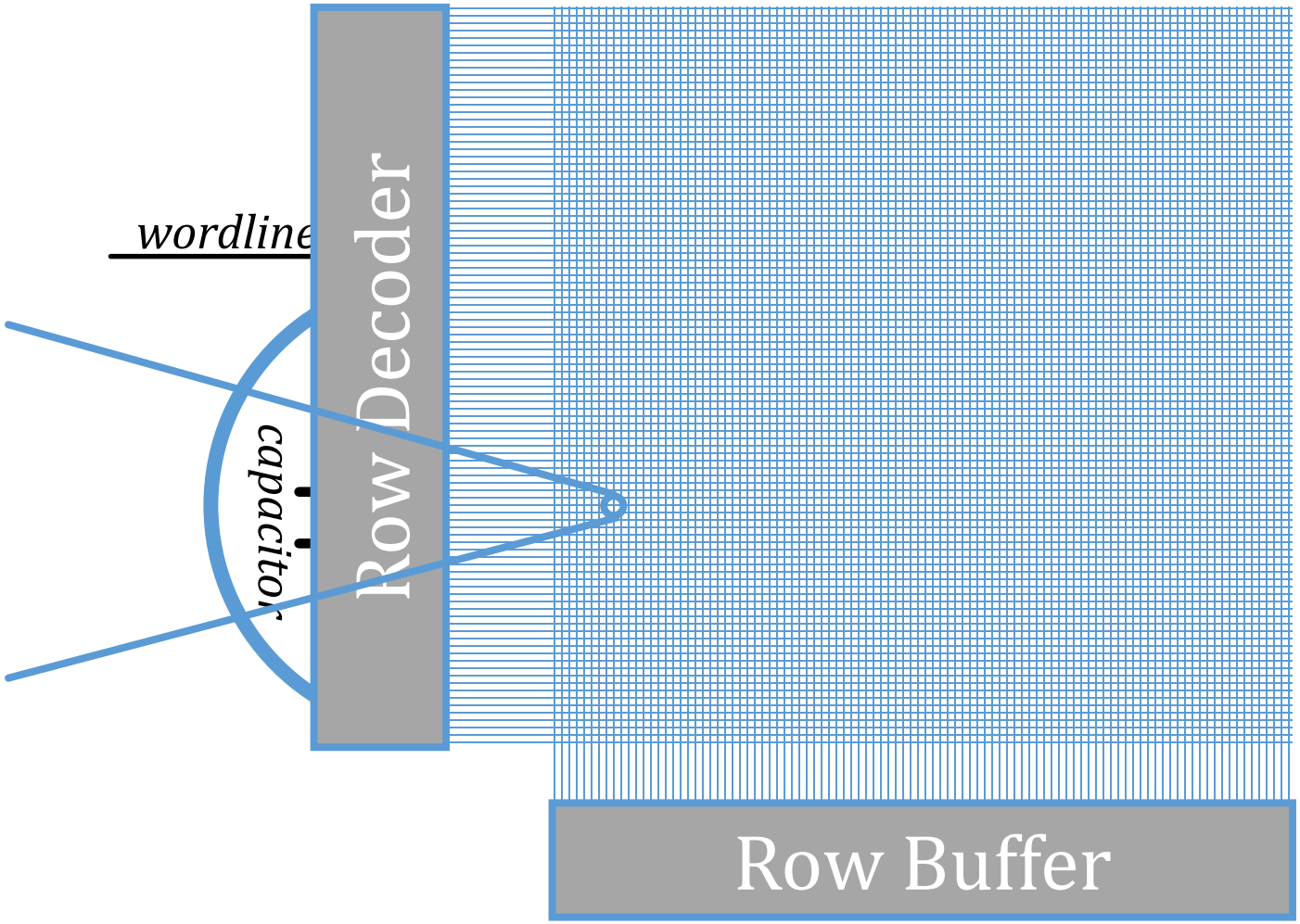
DRAM Cell Retention



Retention failure – when leakage corrupts stored data

Retention time – how long a cell holds its value

DRAM is Much More Than Just One Cell!

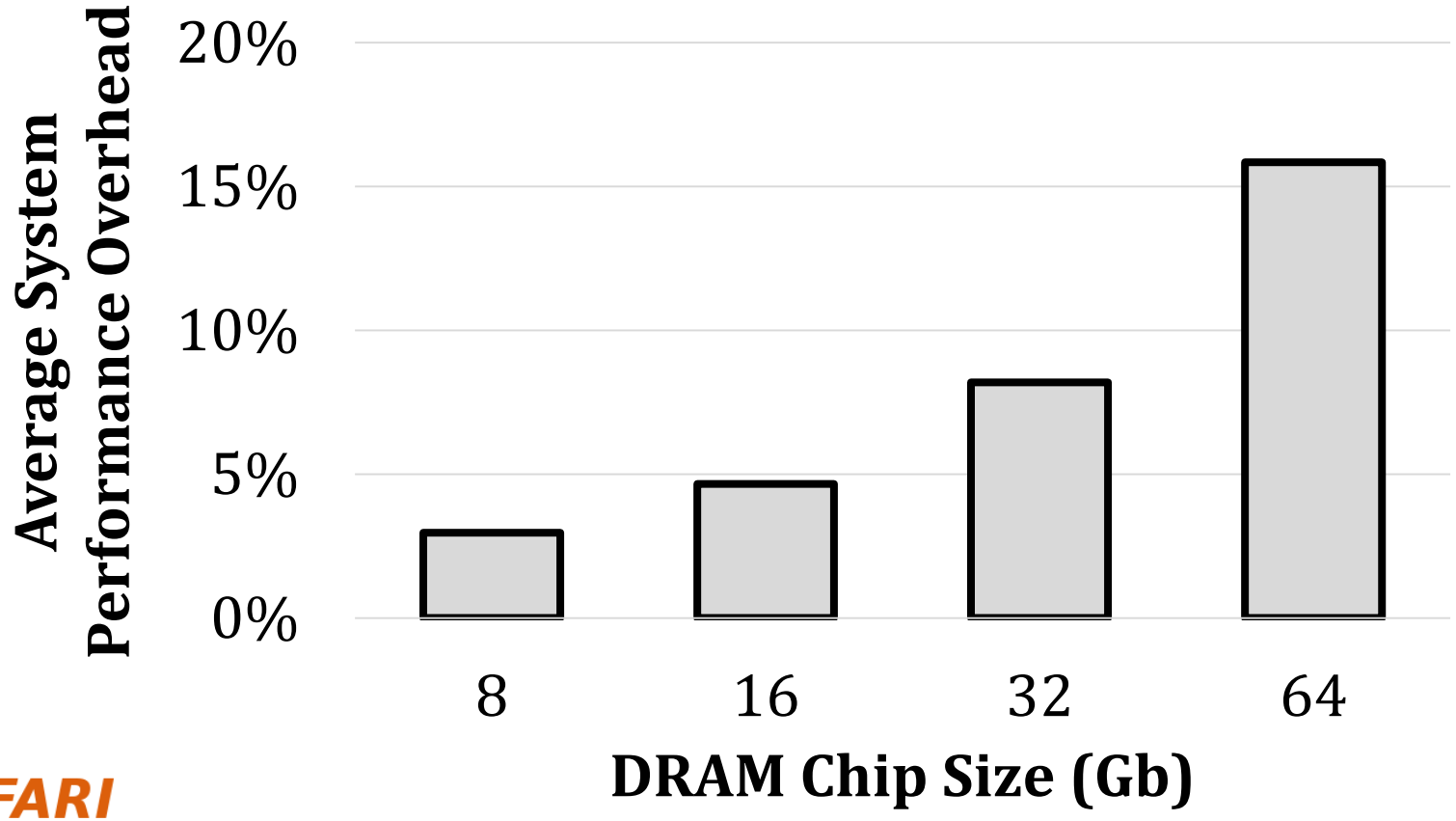


8GB DRAM = 6.4e10 cells

DRAM Refresh

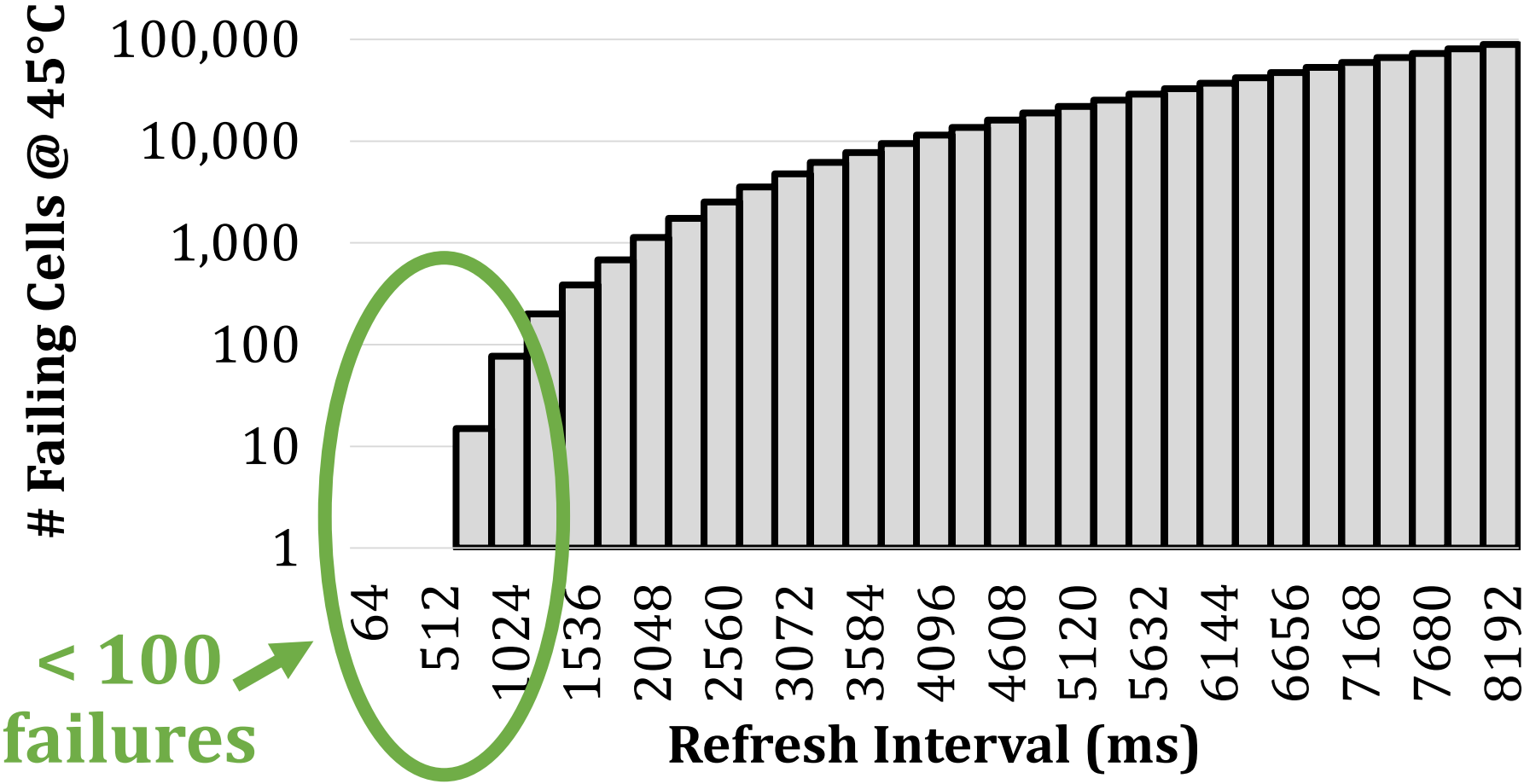
DRAM refresh periodically restores leaked charge

- *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]

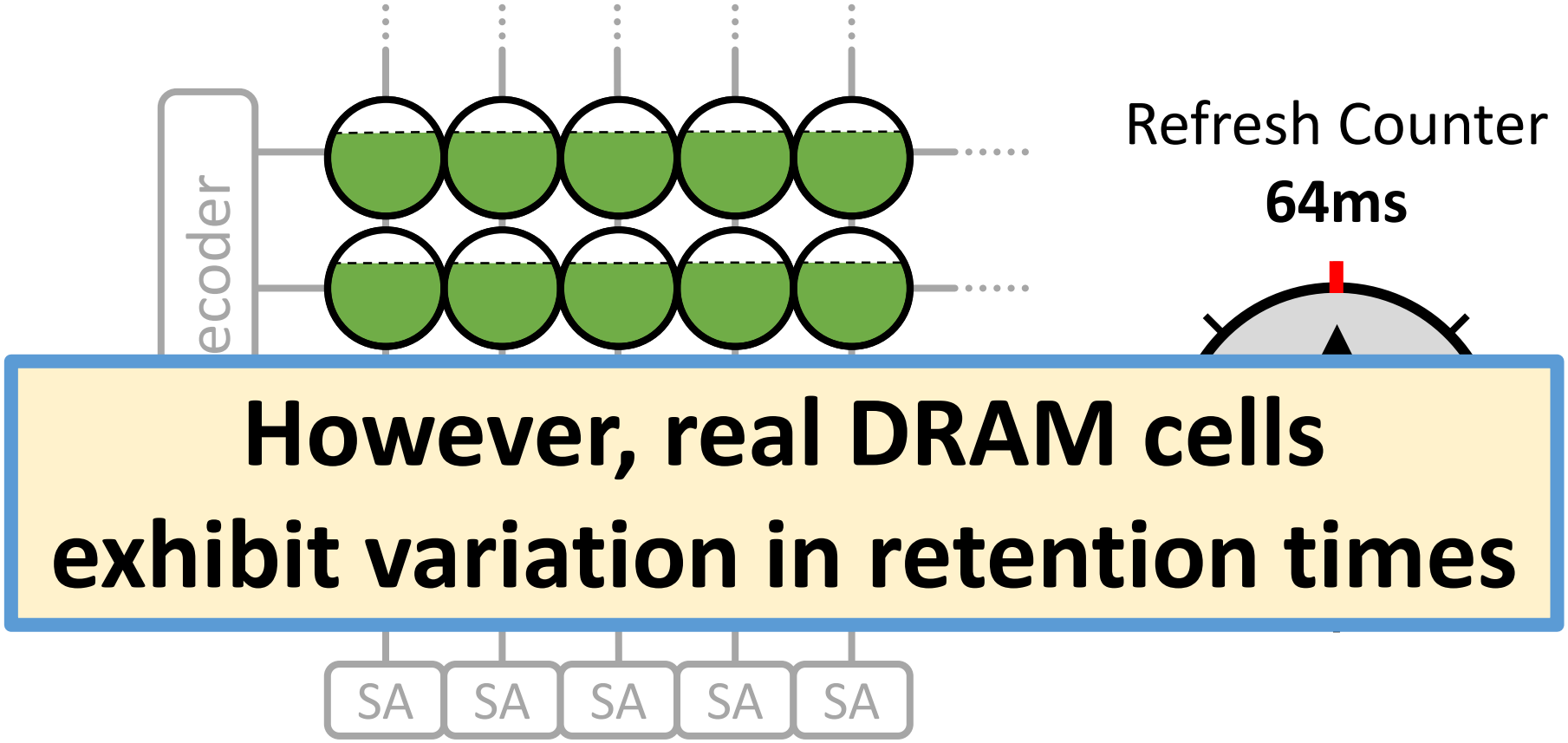
Need a fast and reliable *profiling mechanism* to find the set of retention failures!

- 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



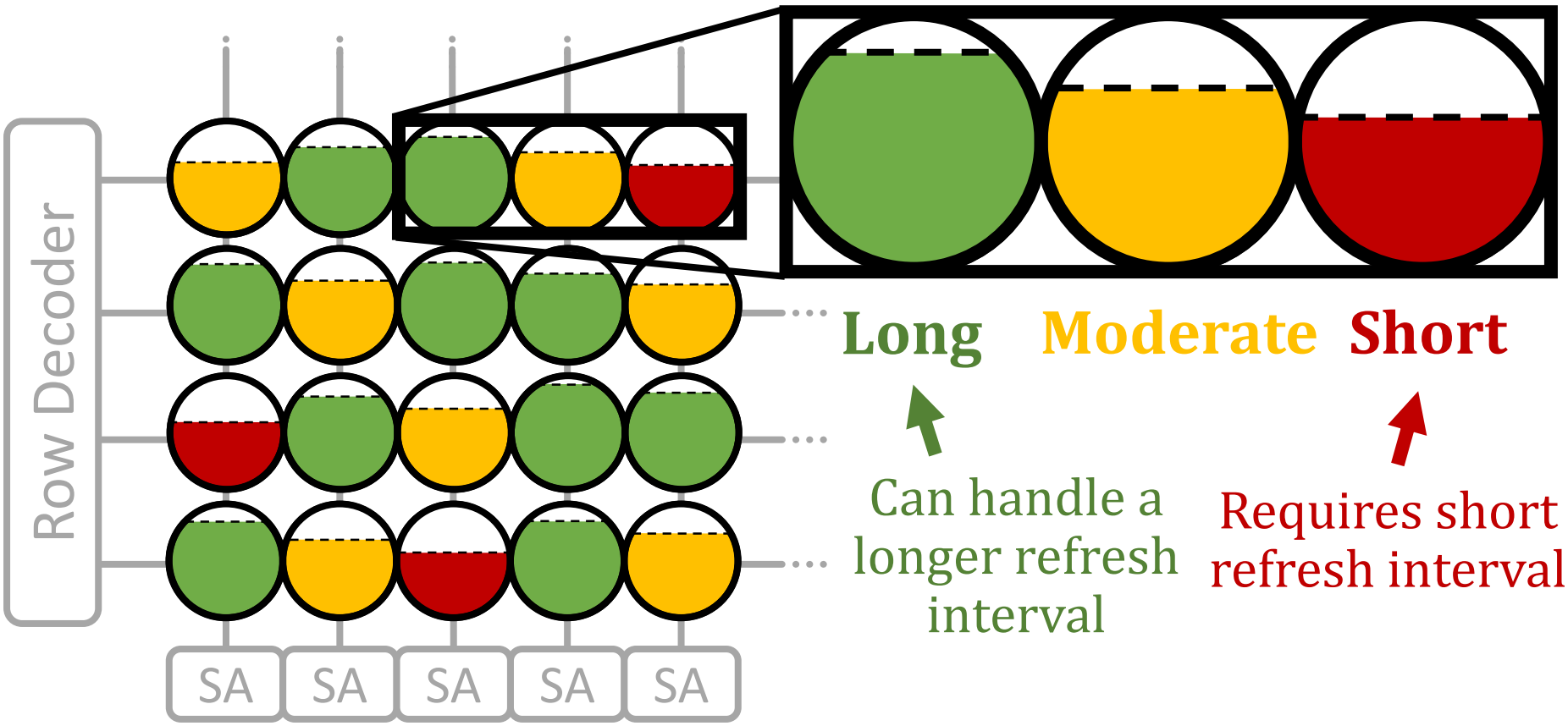
However, real DRAM cells exhibit variation in retention times

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

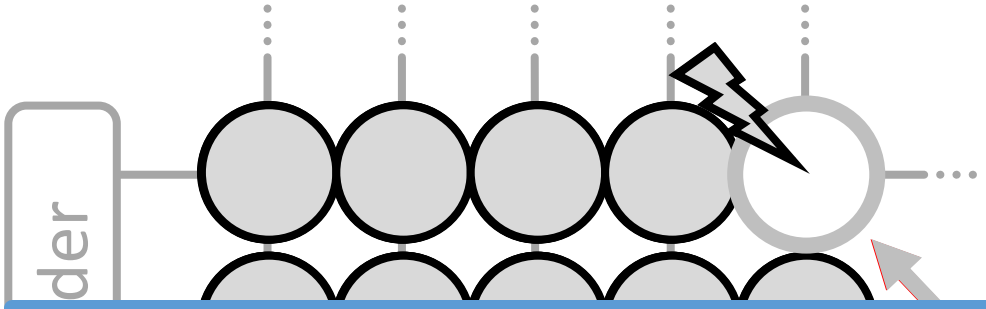
Sources of Retention Time Variation

- **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***?

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

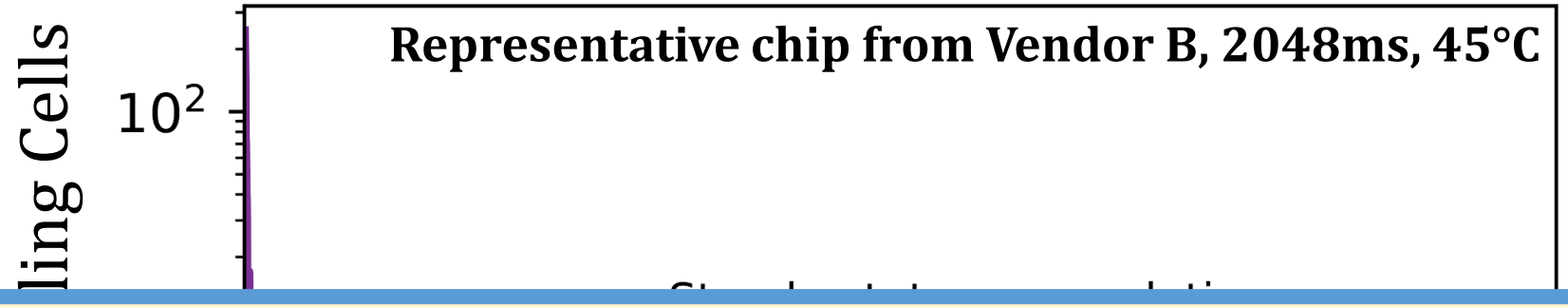
- **368 2y-nm LPDDR4 DRAM chips**
  - 4Gb chip size
  - From 3 major DRAM manufacturers

- **Thermally controlled testing chamber**
  - Ambient temperature range: {40°C – 55°C} ± 0.25°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

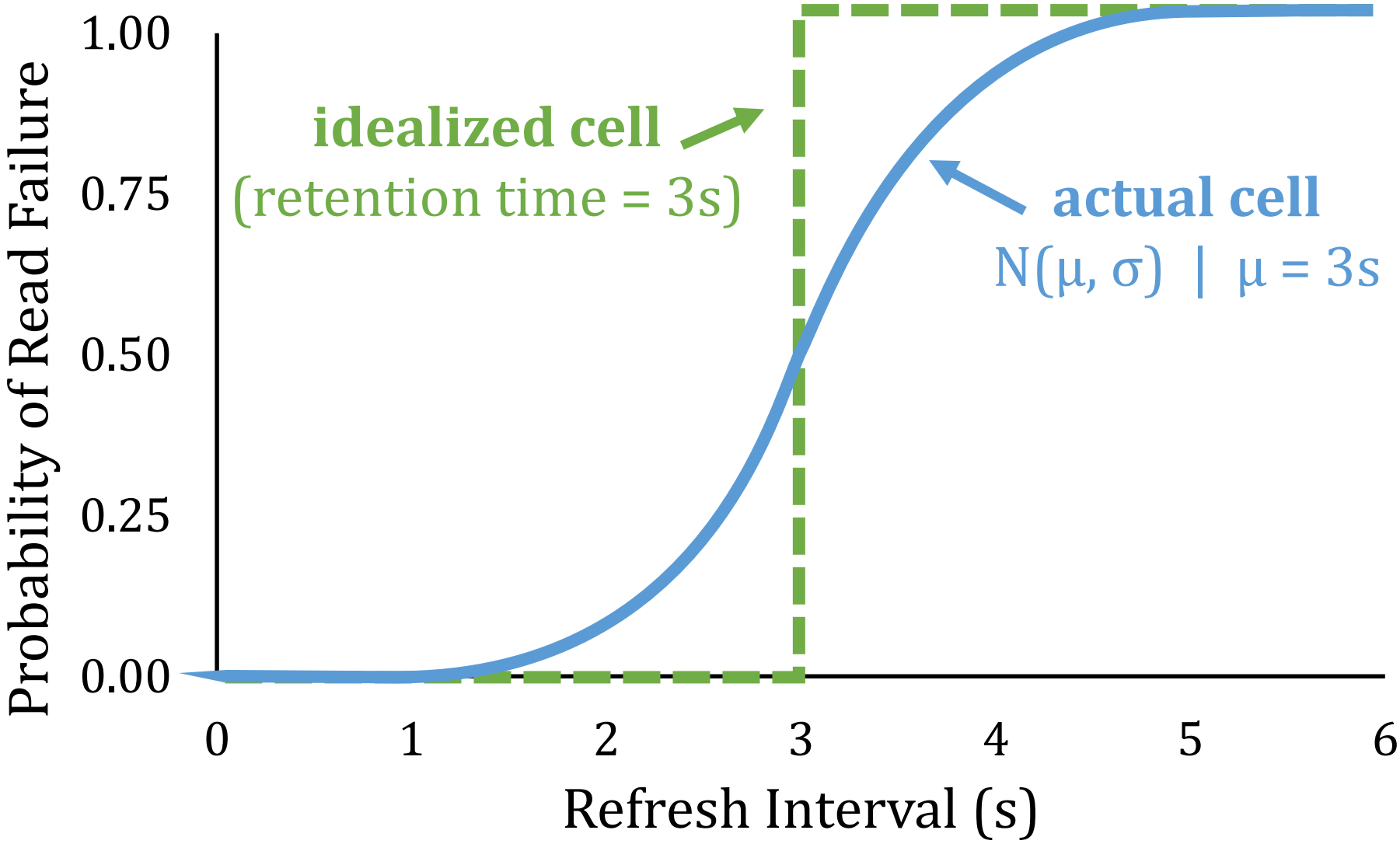
Long-term Continuous Profiling



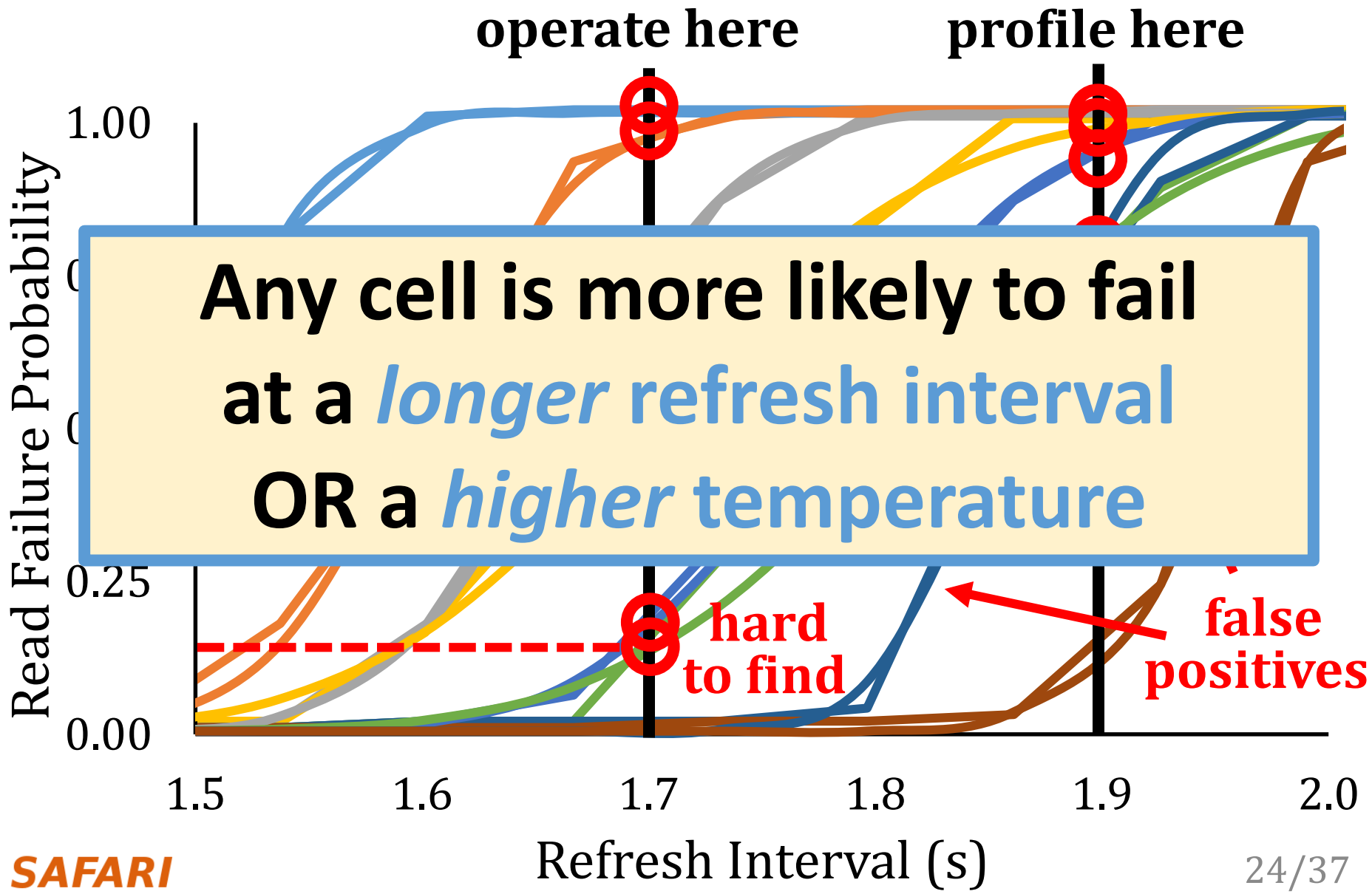
Error correction codes (ECC) and online profiling are *necessary* to manage new failing cells

- 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)



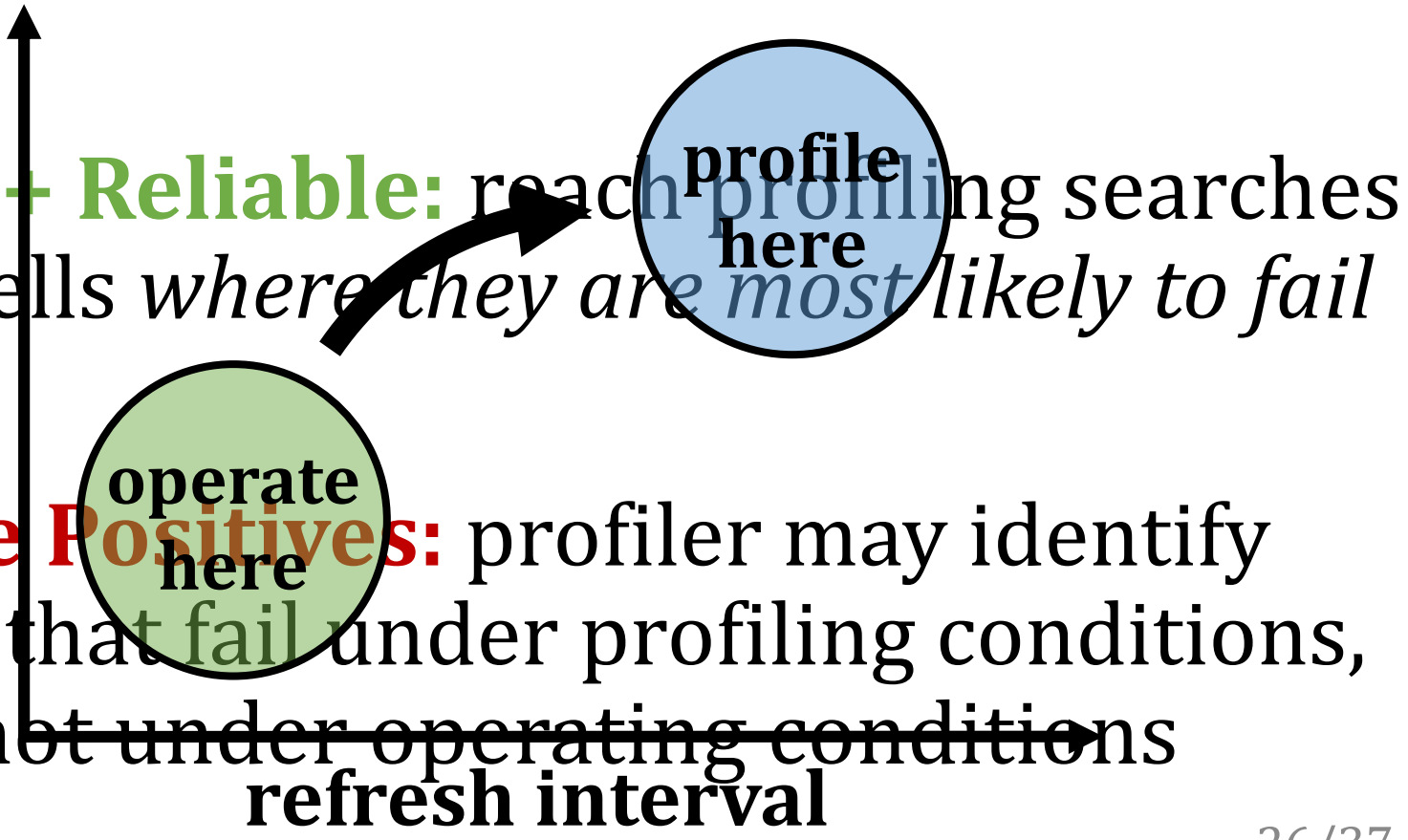
REAPER Outline

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



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

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

Reprofile rarely

Reprofile 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

The Reach Profiler (REAPER): Mitigating the DRAM Refresh Bottleneck

Minesh Patel[1,2] Jeremie S. Kim[1,2] Onur Mutlu[1,2]

[1] ETH zürich [2] Carnegie Mellon

1: Summary

Motivation: High DRAM refresh energy/performance overhead

Problem: DRAM retention failure profiling is hard
- Complicated by cells *changing retention times dynamically*
- Current profiling methods are **unreliable** or **too slow**

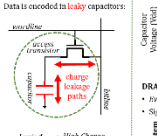
Goals:
1. Thoroughly analyze tradeoffs in retention failure profiling
2. Develop a **fast** and **reliable** profiling mechanism

REAPER:
1. **First** detailed characterization of 368 LPDDR4 DRAM chips
2. **Reach profiling**: Profile at a **longer refresh interval** and/or a **higher temperature**, where cells are more likely to fail
3. **2.5x** faster profiling with **99%** coverage and **50%** false positives
4. **Enables** longer refresh intervals that were previously unreasonable

2: DRAM Refresh

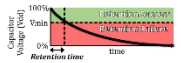
DRAM Cell

Data is encoded in leaky capacitors:



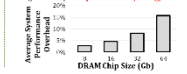
Stored data is corrupted if too much charge leaked

DRAM Cell Retention



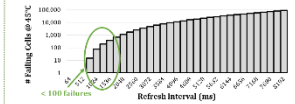
DRAM Refresh

DRAM refresh periodically restores leaked charge
- *Every* cell *every* **refresh interval** (default = 64ms)
- Significant system **performance/energy overhead**



Decreasing DRAM Refresh Overhead

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



Prior works handle these few failures to allow **reliable** operation at a longer refresh interval (*RAIDR [Liu+, ISCA'12], SECRET [Lin+, ICCD'12], ArchShield [Nair+, ISCA'13], DTail [Cui+, SC'14], AVATAR [Qureshi+, DSN'15], ...*)

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

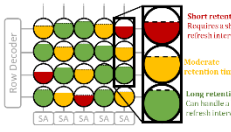
Need a **fast** and **reliable** *profiling mechanism* to find the set of retention failures!

3: Retention Failure Profiling

Profiling Challenges

Retention times vary non uniformly cell to cell

Static and **dynamic** variations due to a combination of:
- **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



Approach #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

Approach #2: Brute-Force

Key idea: for {N data patterns} * {M test rounds}:
1) Write data pattern to DRAM
2) Wait for the refresh interval
3) Check for errors

(e.g., RAMPage [Venkatesan+, HPCA'06], RAIDR [Liu+, ISCA'12])

Pros
- **More reliable:** finds a higher percentage of all possible failures using many different data patterns

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

4: Experimental Studies

Experimental Infrastructure

368 2y nm LPDDR4 DRAM chips tested
- 4Gb chip size (512 Mbit)
- Chips from 3 major DRAM manufacturers, labeled A, B, and C

Thermally controlled testing chamber
- Ambient temperature range: (40°C – 55°C) ± 0.25°C
- DRAM temperature is held at 15°C above ambient.

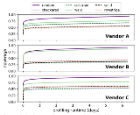
Study 1: Temperature Effects

Well-fitting **exponential** relationship of failure rate vs. temperature for refresh intervals of interest (> 909ms):

$R_A = e^{0.22T}$ $R_B = e^{0.20T}$ $R_C = e^{0.26T}$

Takeaway: Failure rates can be predictably manipulated by changing temperature

Study 2: Data Pattern Dependence

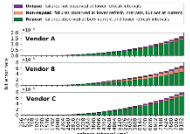


Ratio of total failures discovered by different data patterns over time using brute-force profiling at 45°C + 2048ms

Random data appears to discover failures most easily, but no one data pattern discovers all failures alone

Takeaway: An effective profiling mechanism should use multiple data patterns to attain high coverage of failures

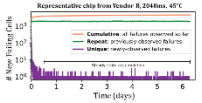
Study 3: Retention Time Distributions



Failure rates compared between different refresh intervals

Takeaway: Failures at a given refresh interval provide a large proportion of failures found at *longer* refresh intervals

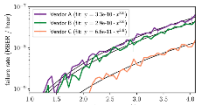
Study 4: Variable Retention Time



Failures discovered over time using brute-force profiling.

We observe that after approximately 10 hours of testing, the number of failures discovered per unit time assumes a constant **nonzero** value.

We attribute these newly discovered failures to the VRT effect, which is consistent with prior work on DDR3 DRAM [Liu+ ISCA'13, Qureshi+ DSN'15]

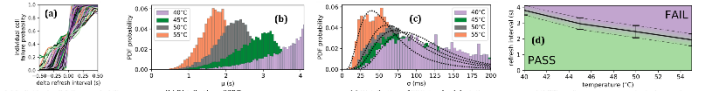


VRT steady-state failure rates for different refresh intervals

Longer refresh intervals result in higher steady-state failure rates, and the resulting curves fit polynomial curves well

Takeaway: Profiling inevitably must be repeated after some period of time (i.e., online profiling is required)

Study 5: Individual Cell Failures



(a) Individual cell failure probability across refresh intervals for many different cells (normalized by mean).

Cells fail with a **normally-distributed CDF** with respect to refresh interval.

(b) Distribution of CDF means across different temperatures.

The means follow an exponential-like distribution, which is consistent with the failure rate vs. refresh interval curves.

(c) Distribution of CDF standard deviations across different temperatures.

We see that they follow a log normal distribution, not to be confused with the log-normal distribution of retention times.

(d) Effect of temperature manipulation on the combined distributions of all failing cells from a representative chip of vendor B (solid line for the combined mean, dashed line for the combined standard deviation).

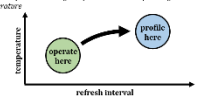
The typical cell is more likely to fail at a *longer refresh interval* or at a *higher temperature*.

Takeaway: In order to find failures more quickly and consistently, we should profile at a **larger refresh interval** and/or a **higher temperature**.

5: Reach Profiling

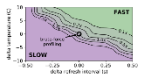
Reach Profiling: Key Idea

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



Example profiling runtime for different choices of profiling conditions relative to the operating conditions.

Implementation

Reach profiling is a **general methodology**

Must answer 3 key questions for an implementation:
1. What are **desirable profiling conditions**?
2. **How often** should the system profile?
3. **What information** does the profiler need?

Three Key Metrics

We identify three key profiling metrics any effective profiling mechanism must achieve:
1. **Runtime:** how long profiling takes
2. **Coverage:** portion of all possible failures discovered by profiling
3. **False positives:** number of cells observed to fail during profiling but never during actual operation

Q1: Desirable Conditions

We find 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 are in the paper

Q2: How often to profile

Estimation using a **probabilistic model**
- Uses ECC to temporarily manage VRT effects
- 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 [Patel+ ISCA'17]

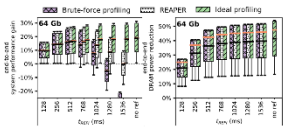
Our Mechanism: REAPER

Simple implementation of reach profiling

Pessimistic assumption:
- Whole system pauses while firmware executes profiling
- Only controls refresh interval, not temperature

Performance and Energy Evaluation

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



End-to-end system performance (left) and DRAM power consumption (right) for 64 Gb DRAM chips with a reduced refresh interval across 20 heterogeneous 4-core workloads.

On average, REAPER:
- Enables high performance improvement and DRAM power reduction by reliably increasing the refresh interval, achieving **16.3% (max 27.0%) system performance** improvement **36.4% (max 47.4%) DRAM power** reduction
- Significantly **outperforms brute-force profiling** at long refresh intervals
- **Enables long refresh intervals** (e.g., 1280ms) that were previously unreasonable
- Has **negligible impact** on power consumption

The Reach Profiler (REAPER):

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

Minesh Patel **Jeremie S. Kim**

Onur Mutlu

SAFARI

ETH Zürich

Carnegie Mellon

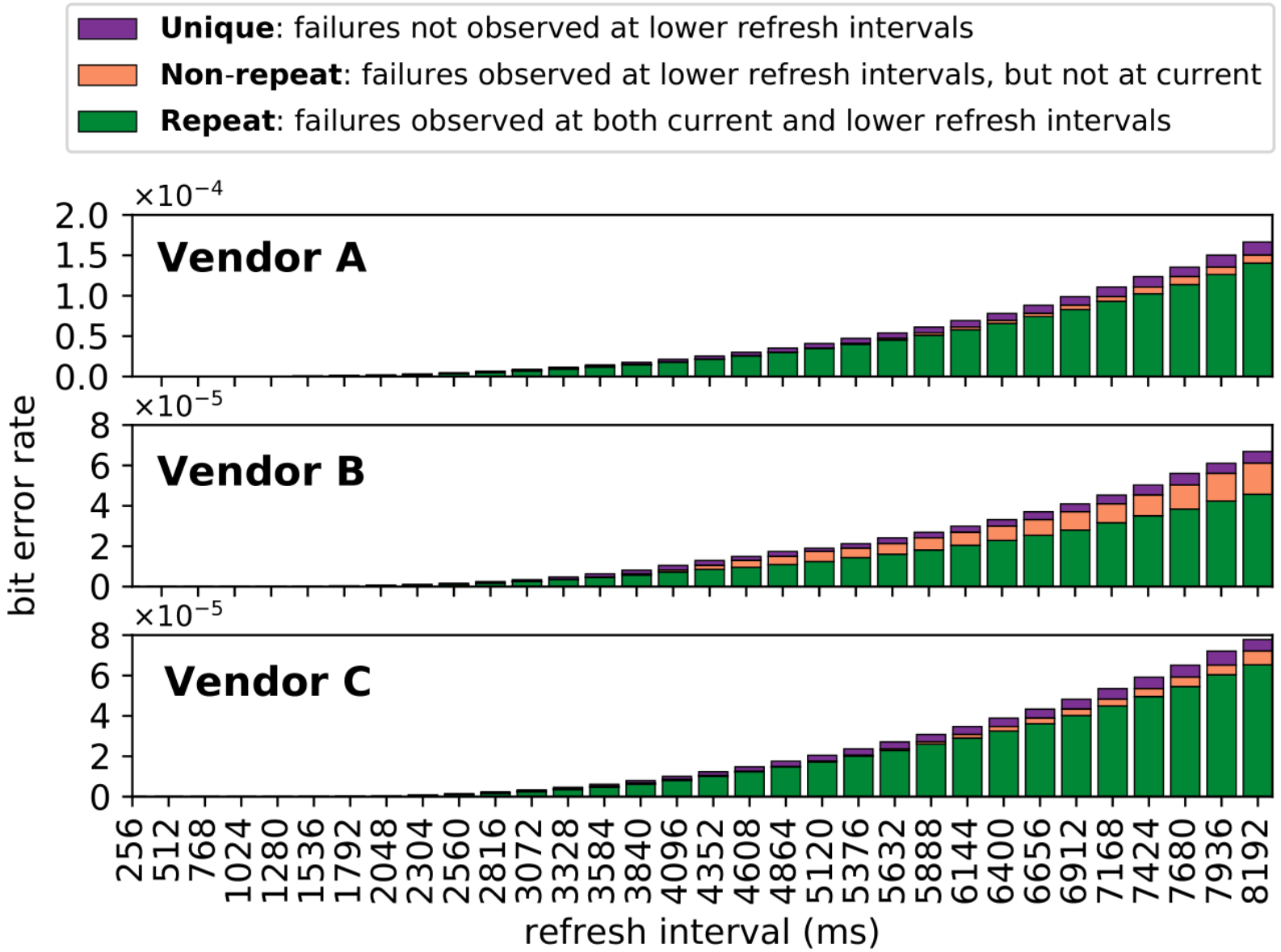
Temperature Relationship

- Well-fitting exponential relationship:

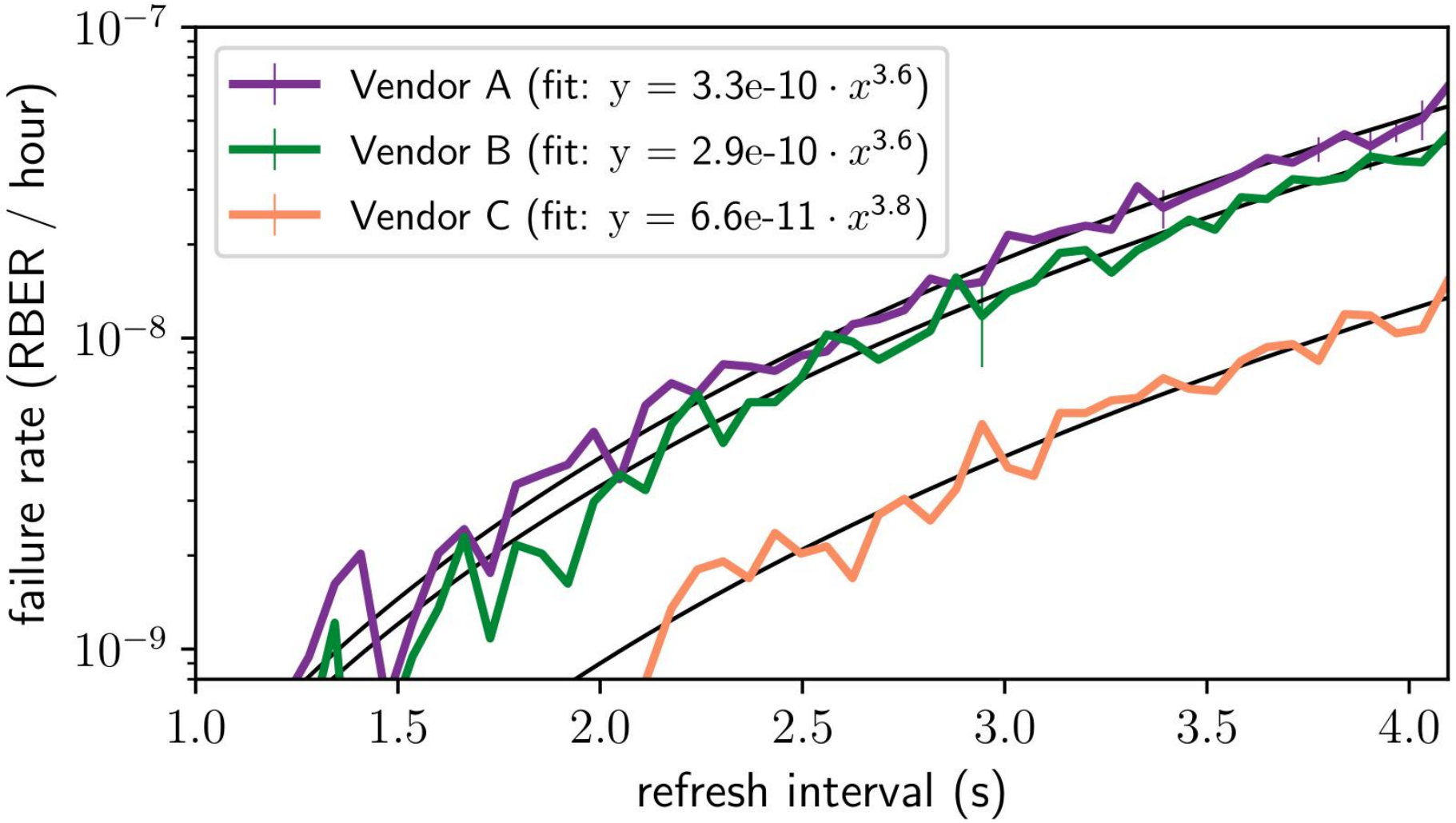
$$R_A \propto e^{0.22\Delta T} \qquad R_B \propto e^{0.20\Delta T} \qquad R_C \propto e^{0.26\Delta T}$$

- E.g., 10°C ~ 10x more failures

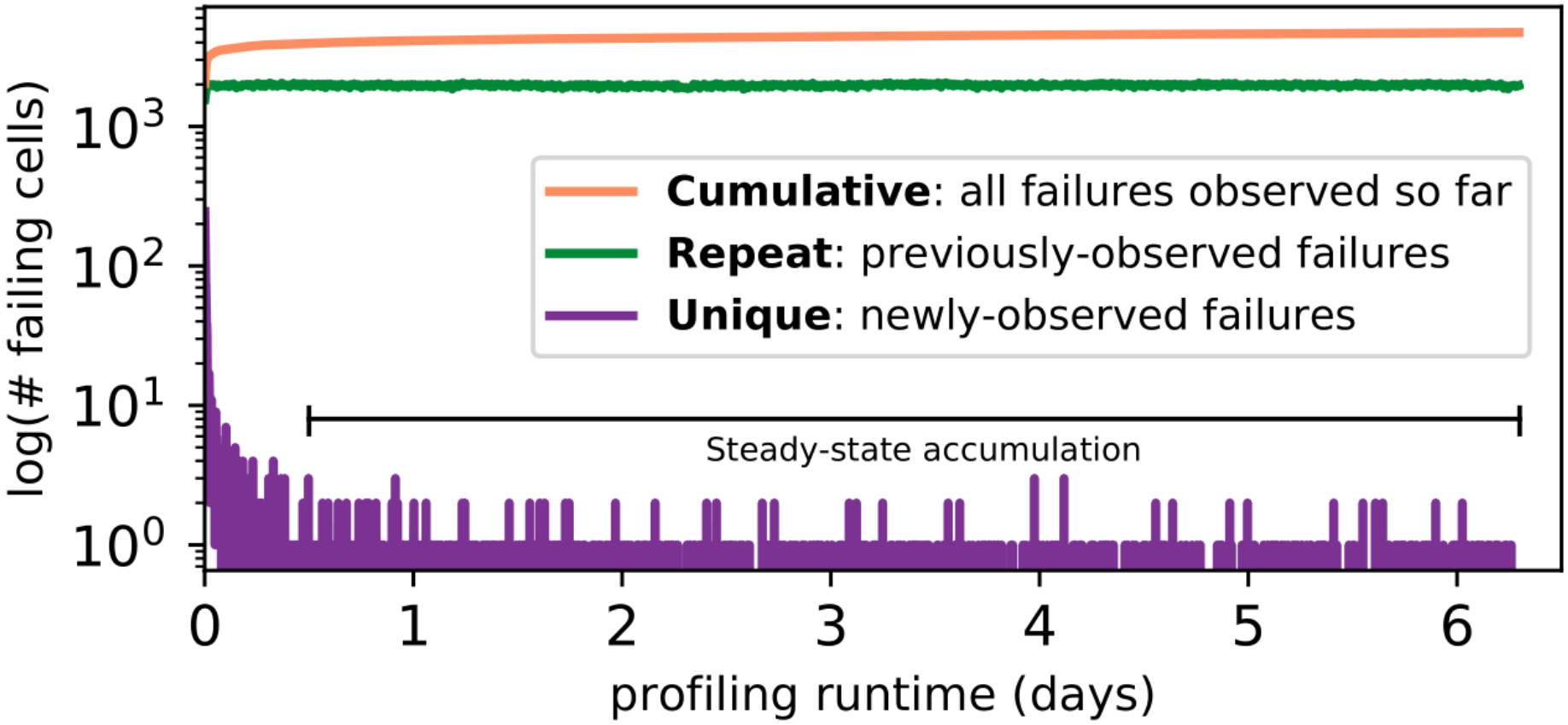
Retention Failures @ 45°C



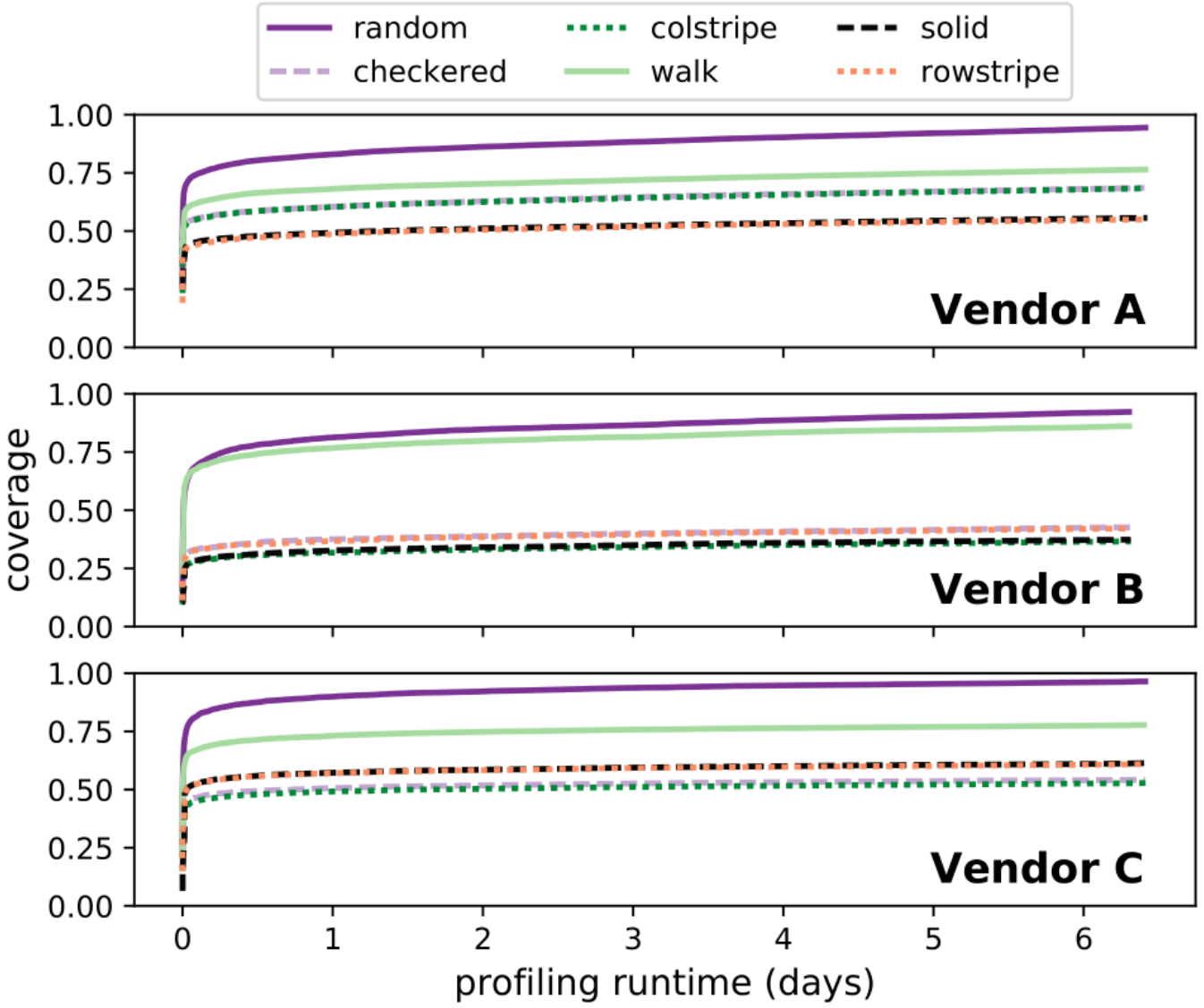
VRT Failure Accumulation Rate



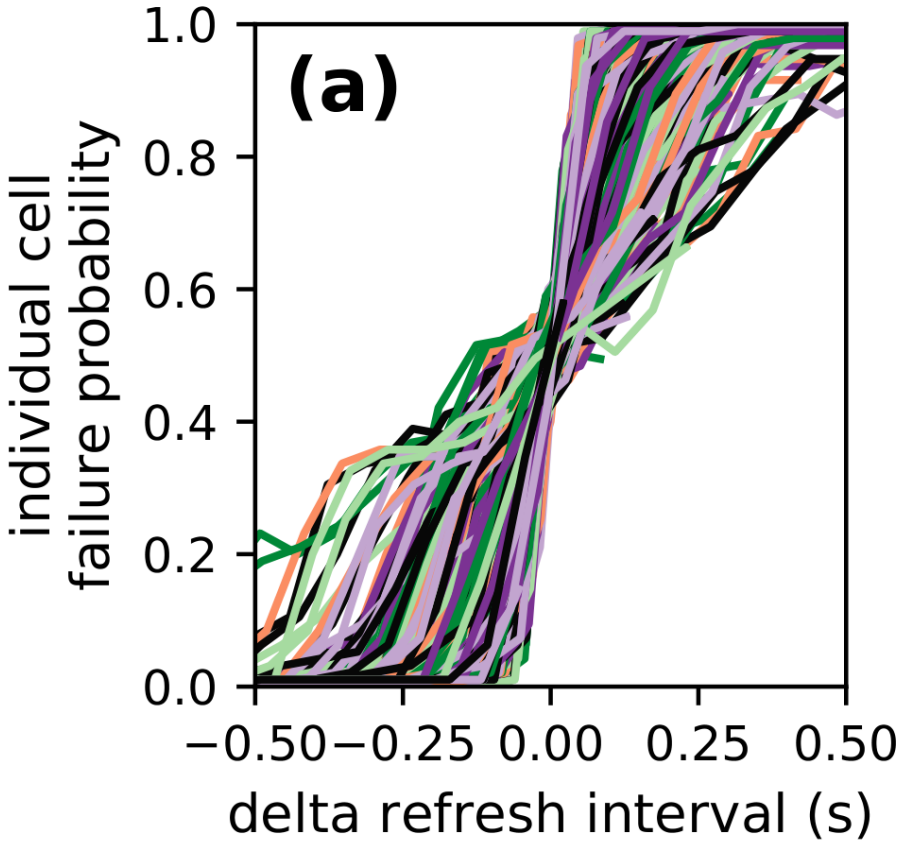
800 Rounds of Profiling @ 2048ms, 45°C

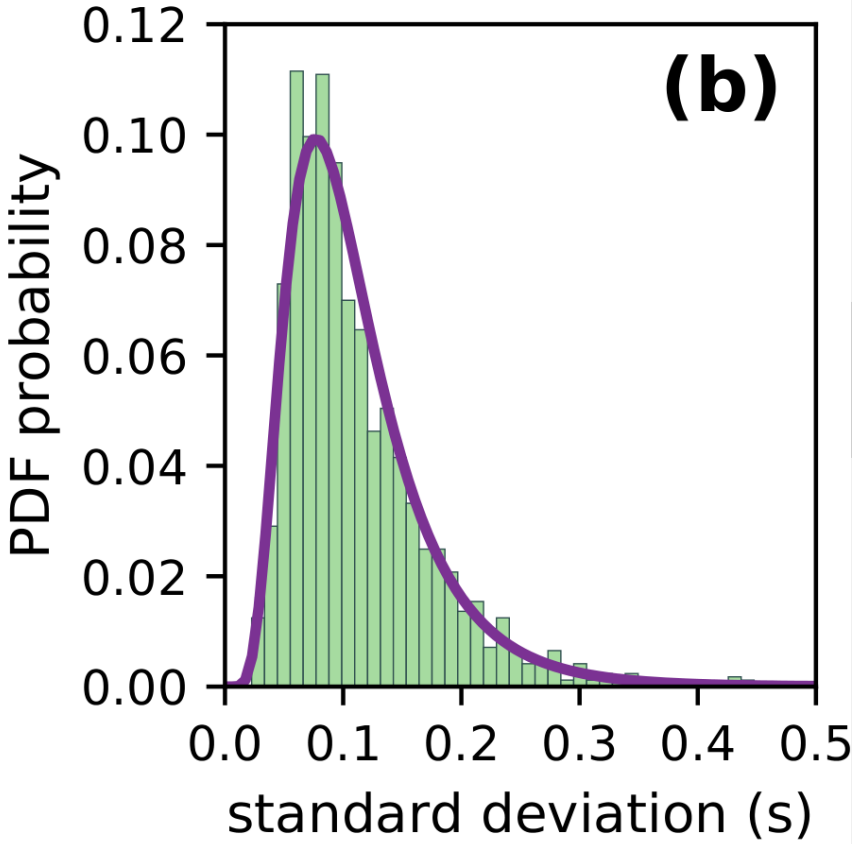


800 Rounds of Profiling @ 2048ms, 45°C



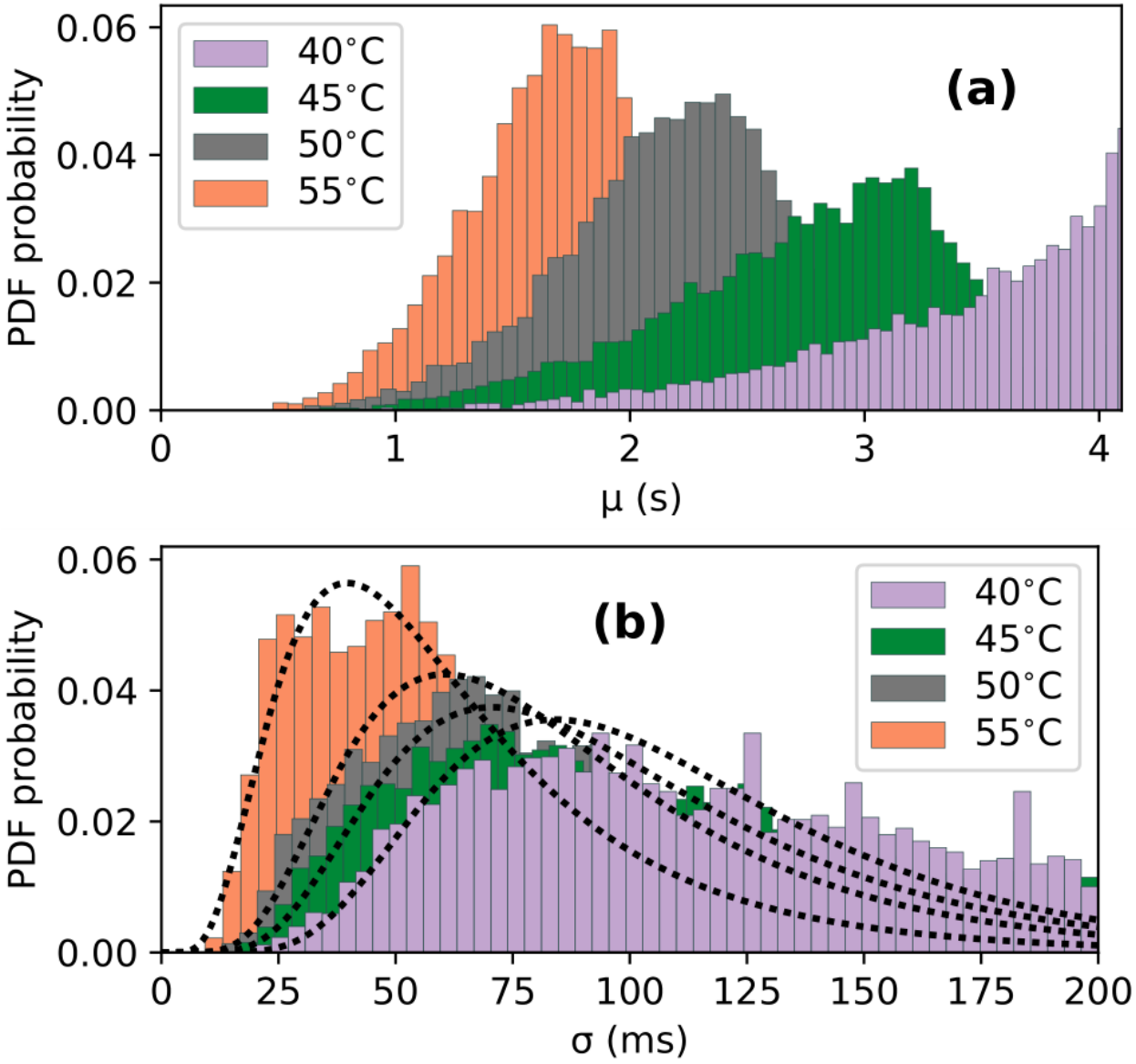
Individual Cell Failure Probabilities



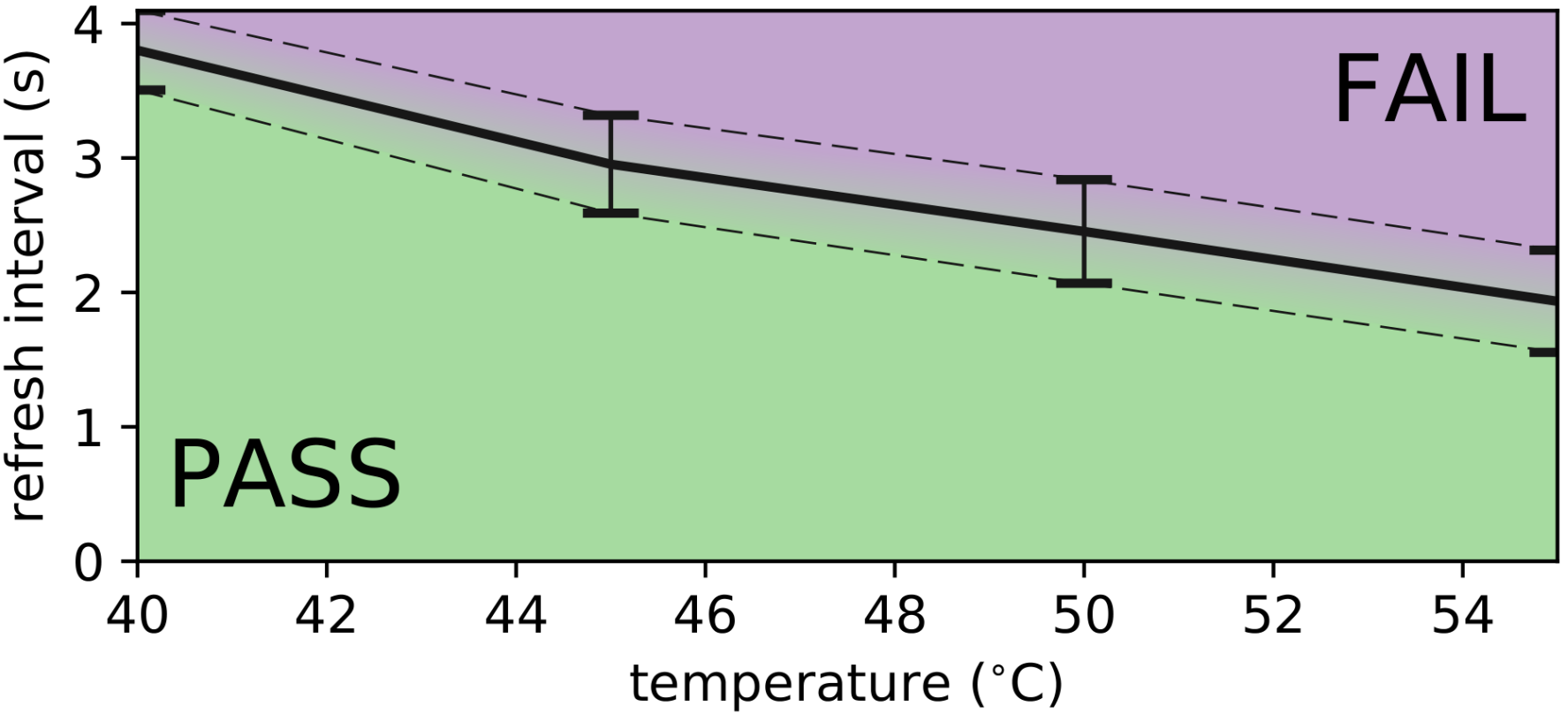


- Single representative chip of Vendor B at 40° C
- Refresh intervals ranging from 64ms to 4096ms

Individual Cell Failure Distributions



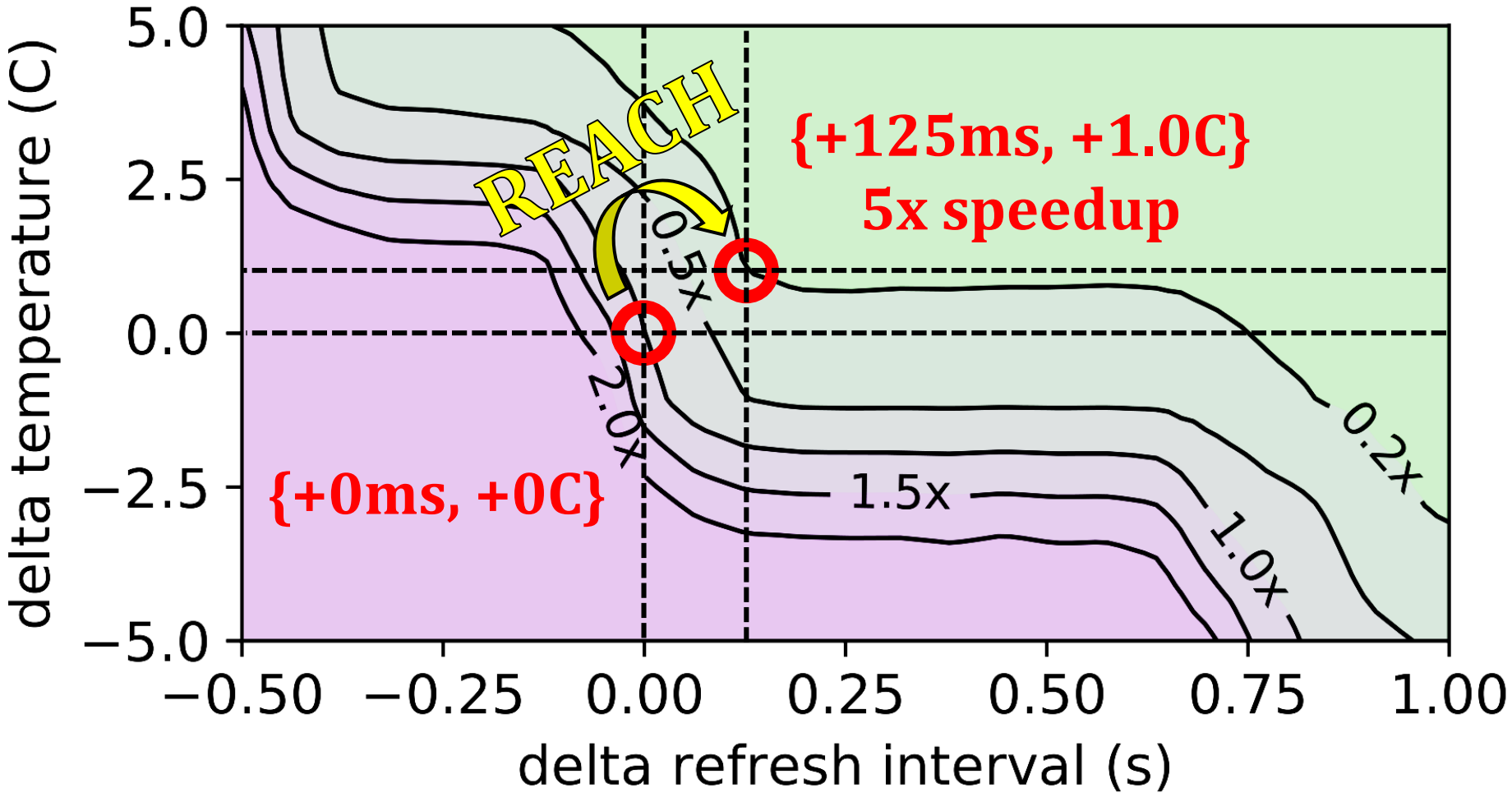
Single-cell Failures With Temperature



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

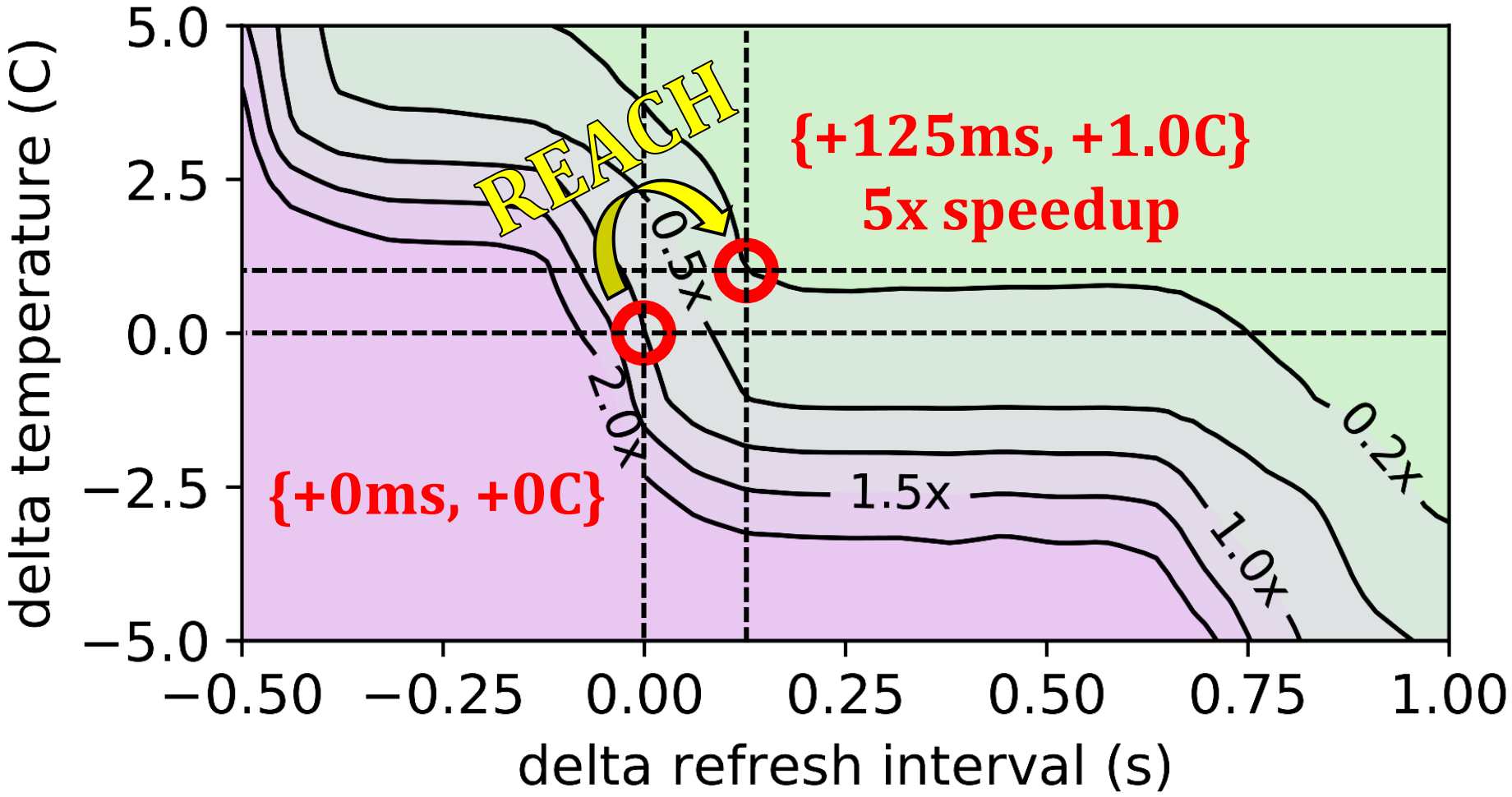
Example experimental analysis

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:

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

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

Allowable Errors

- Tolerable **RBER** 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 RBER		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

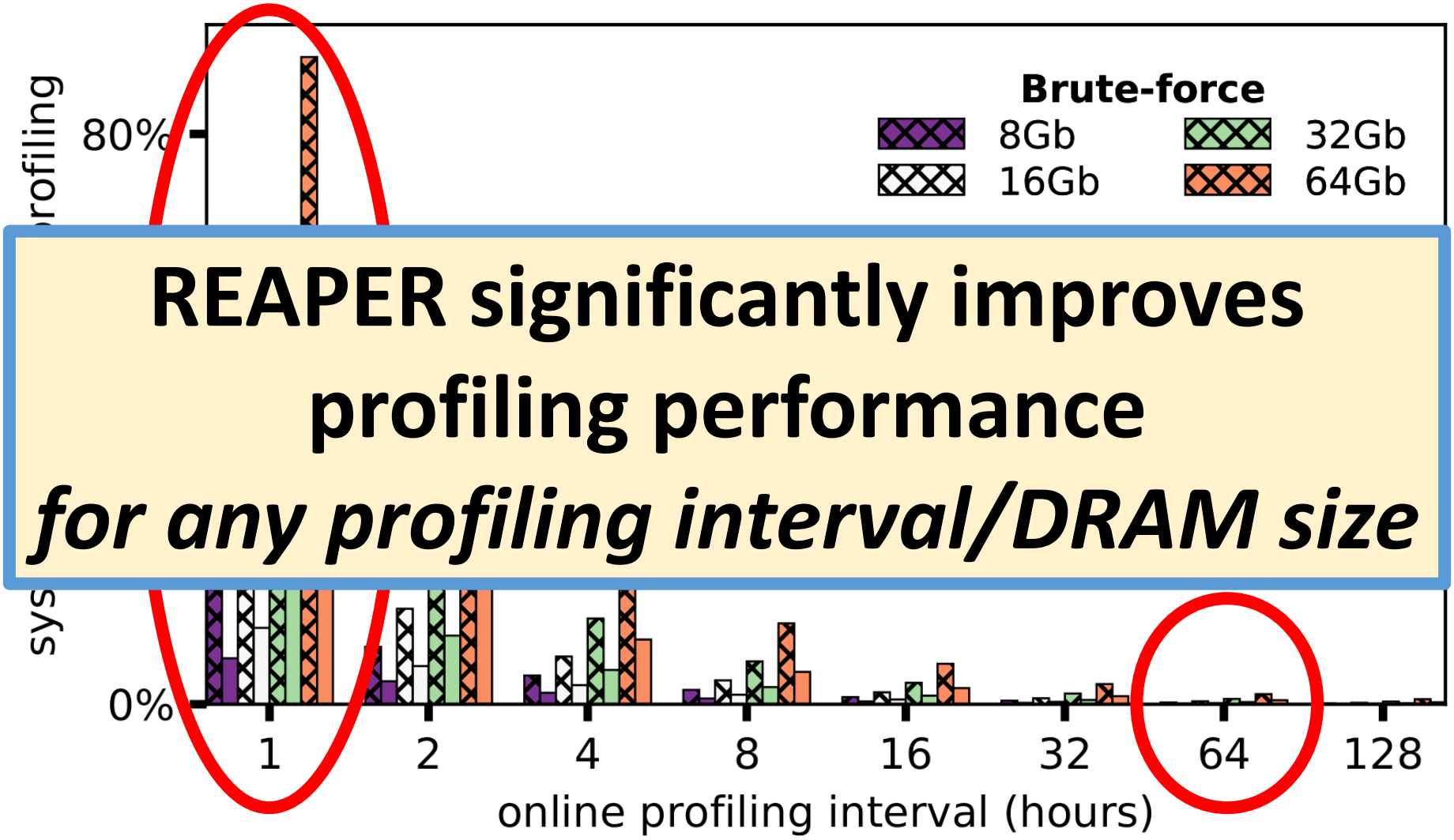
Tradeoff Space Exploration

- 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

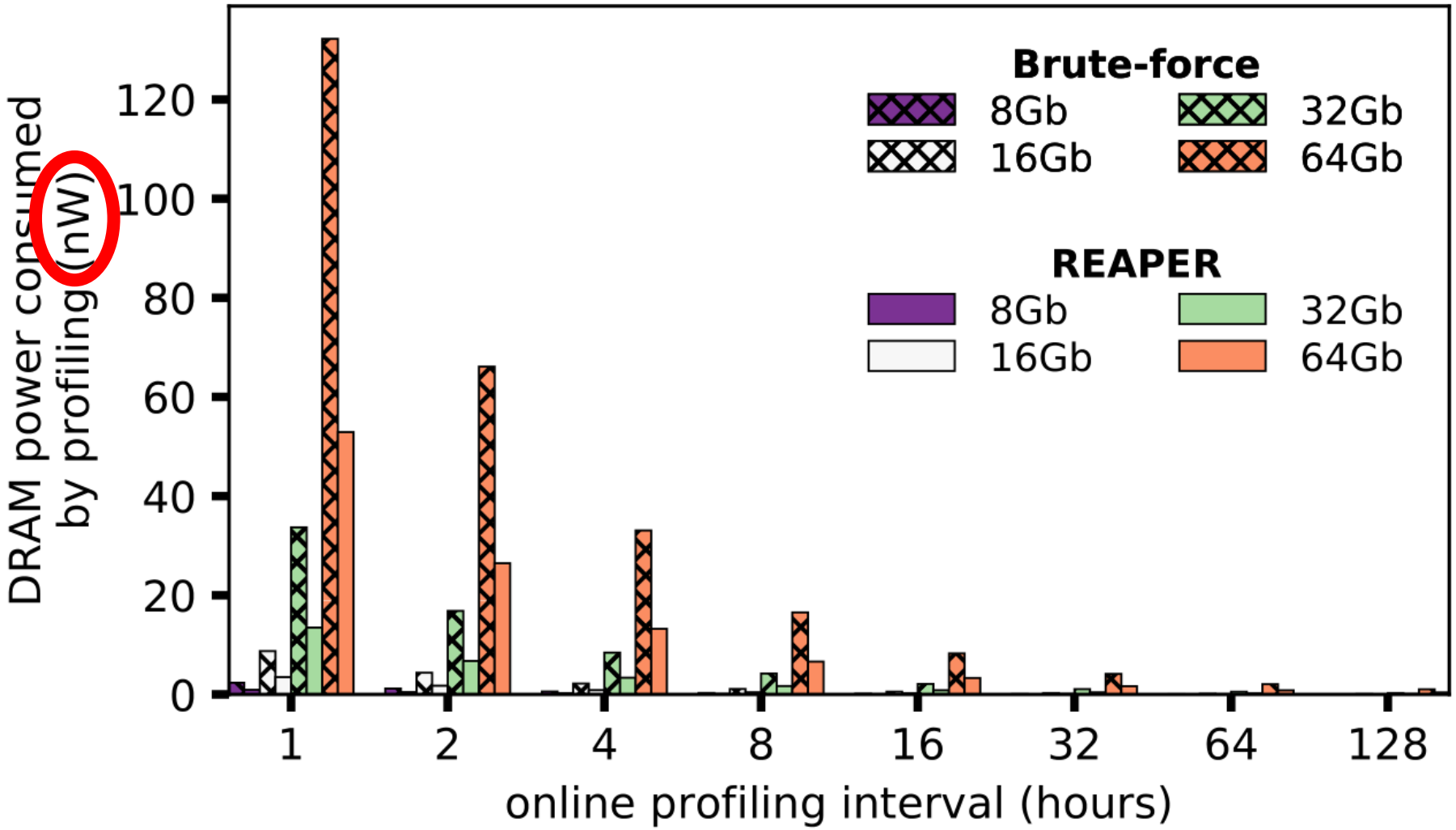
Profiling Performance Overhead



REAPER significantly improves profiling performance

for any profiling interval/DRAM size

Profiling Energy Overhead



End-to-end Performance/Energy

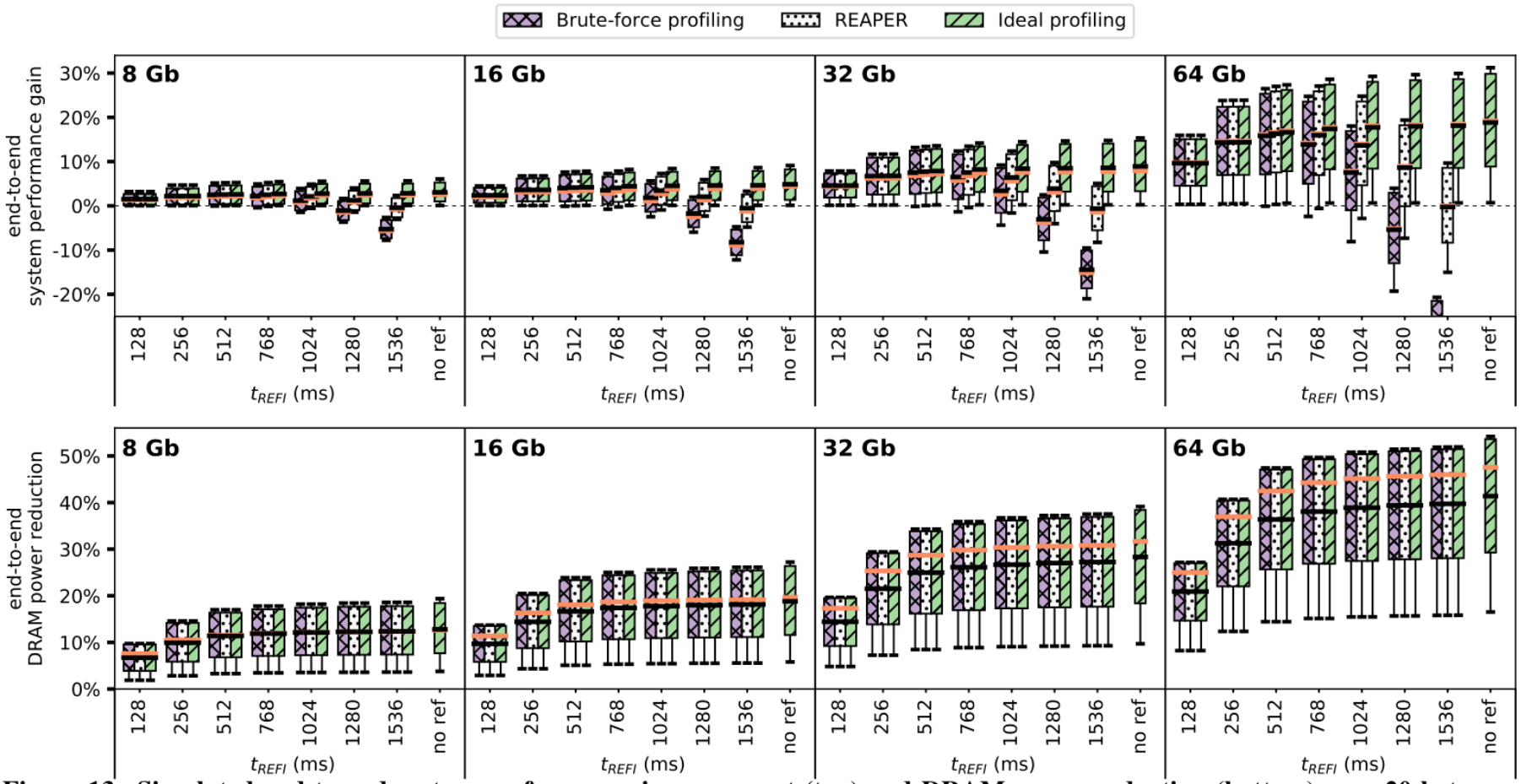


Figure 13: Simulated end-to-end system performance improvement (top) and DRAM power reduction (bottom) over 20 heterogeneous 4-core workloads for different refresh intervals at 45°C, taking into account online profiling frequency and profiling overhead.

Algorithm 1: Brute-Force Profiling Algorithm

```
1  PROFILE(target_t_REF, num_iterations):
2      failed_cells = []
3      for it ← {1 to num_iterations}:
4          for dp ∈ data_patterns:
5              write_DRAM(dp)
6              disable_refresh()
7              wait(target_t_REF)
8              enable_refresh()
9              this_iteration_failures ← 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**

A Complex Tradeoff Space