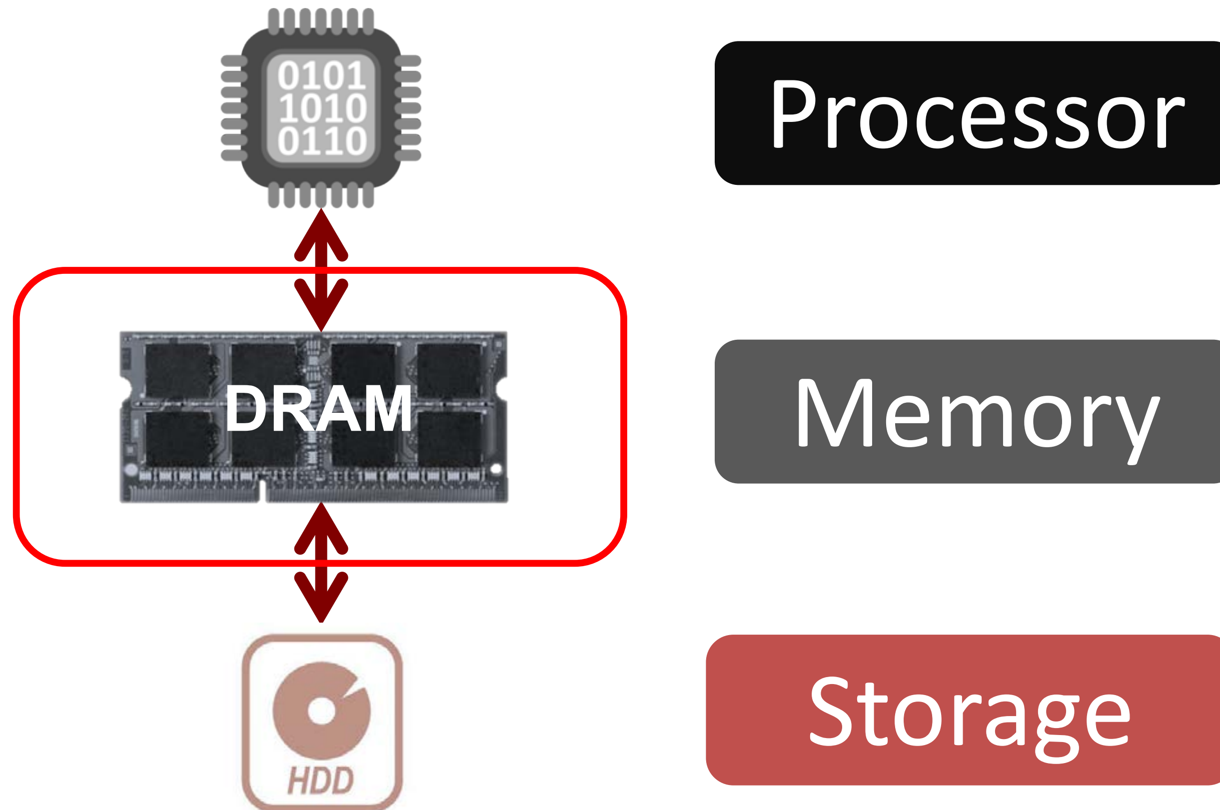
SOLVING THE DRAM SCALING CHALLENGE

Samira Khan

MEMORY IN TODAY'S SYSTEM

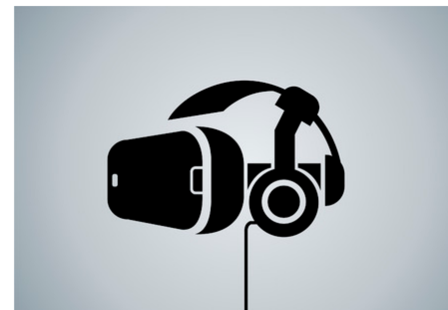


DRAM is critical for performance

TREND: DATA-INTENSIVE APPLICATIONS

DNA/PROTEIN SYNTHESIS

IMAGE ANALYSIS



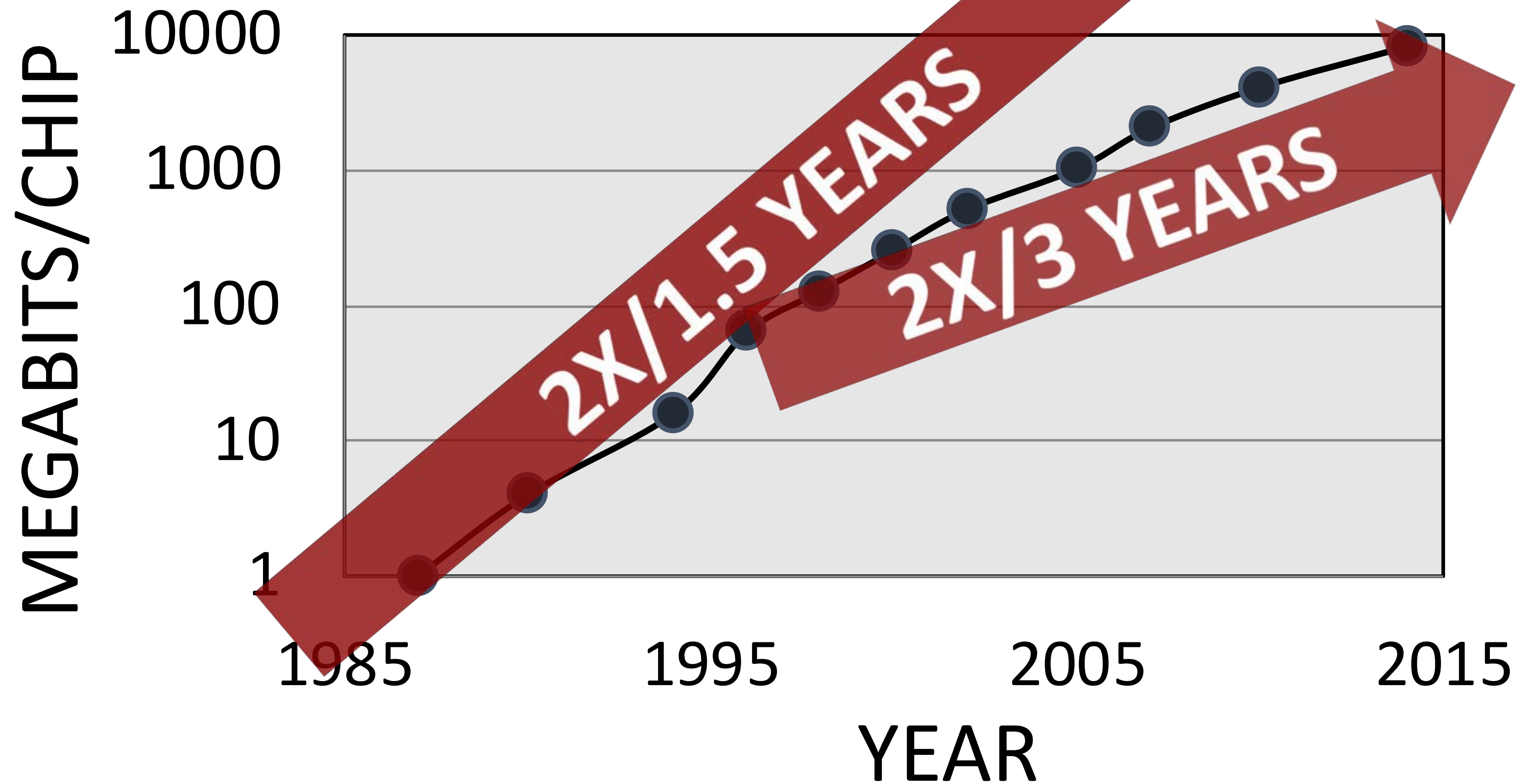
VIRTUAL REALITY

Spark™ amazon DynamoDB SAP HANA

IN-MEMORY FRAMEWORKS

Increasing demand for high-capacity, high-performance, energy-efficient main memory

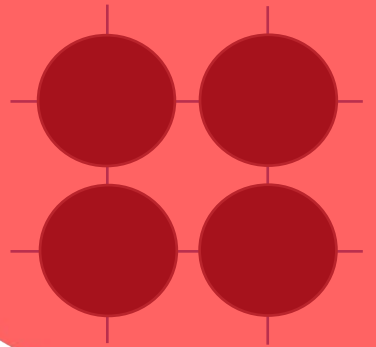
DRAM SCALING TREND



DRAM scaling is getting difficult

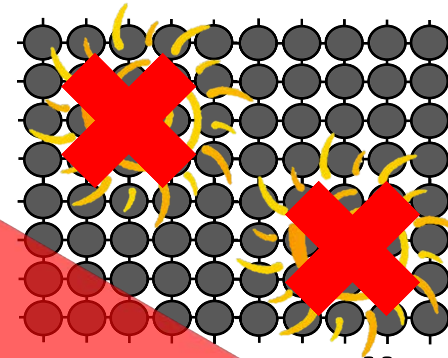
Source: Flash Memory Summit 2013, Memcon 2014

DRAM SCALING CHALLENGE



DRAM Cells

Technology Scaling

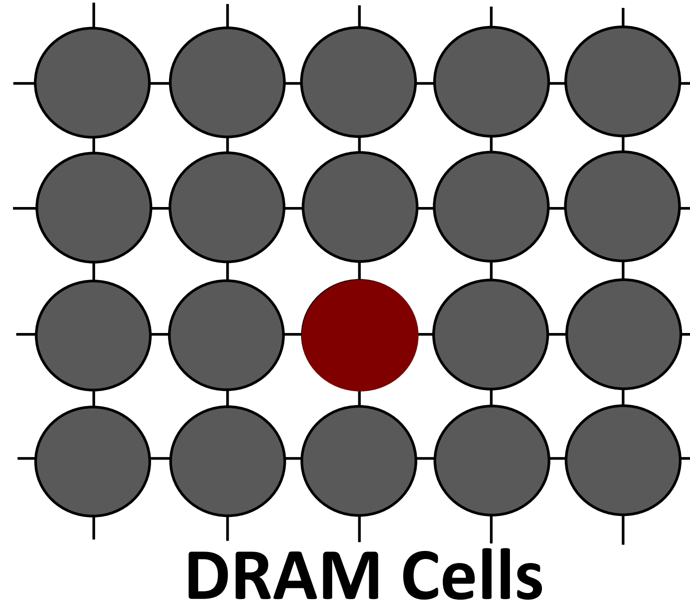


DRAM Cells

WHY?

Manufacturing reliable cells at low cost is getting difficult

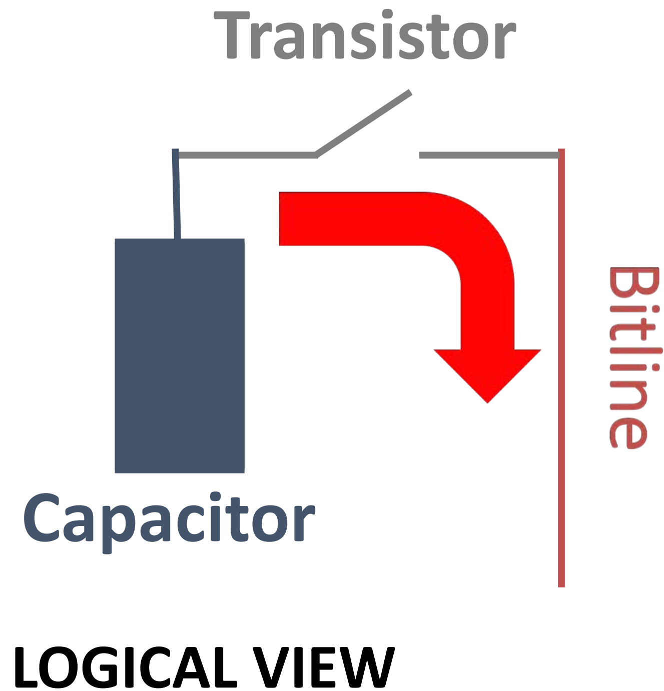
WHY IS IT DIFFICULT TO SCALE?



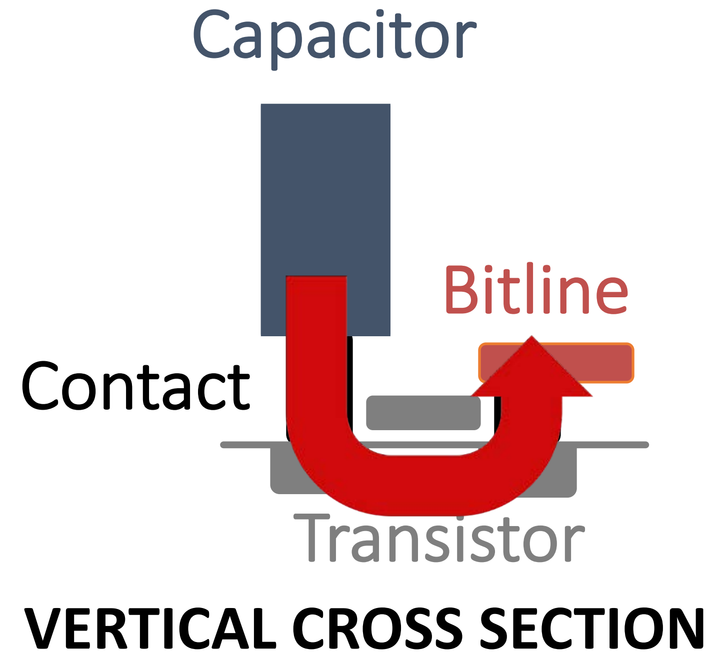
In order to answer this we need to take a closer look to a DRAM cell

DRAM CELL OPERATION

1. A DRAM cell stores data as *charge*
2. A DRAM cell is refreshed every *64 ms*



LOGICAL VIEW



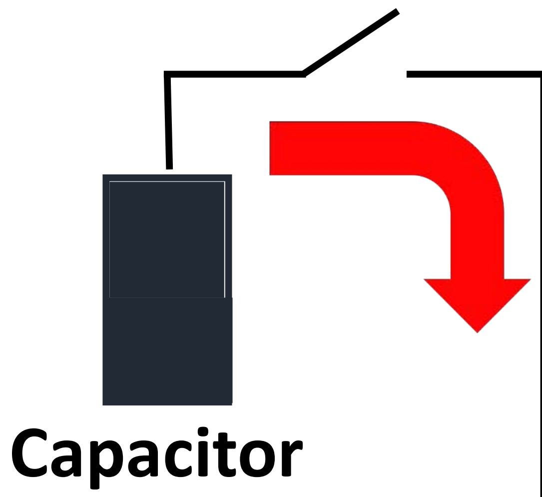
VERTICAL CROSS SECTION

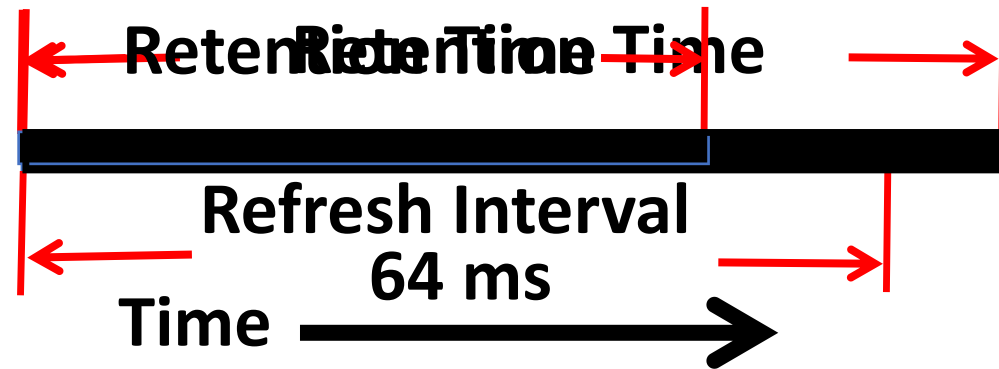
A DRAM cell

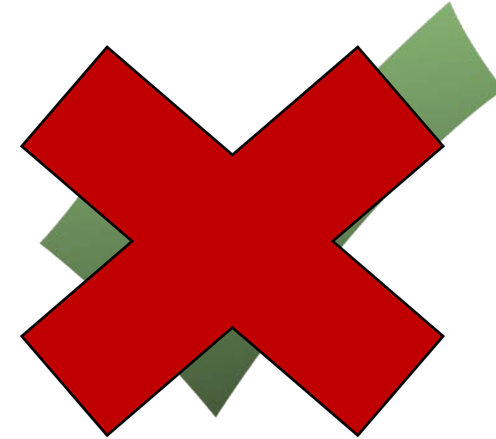
DRAM RETENTION FAILURE

Retention time: The time when we can still access a cell reliably

Cells need to be refreshed before that to avoid failure







Retention time is

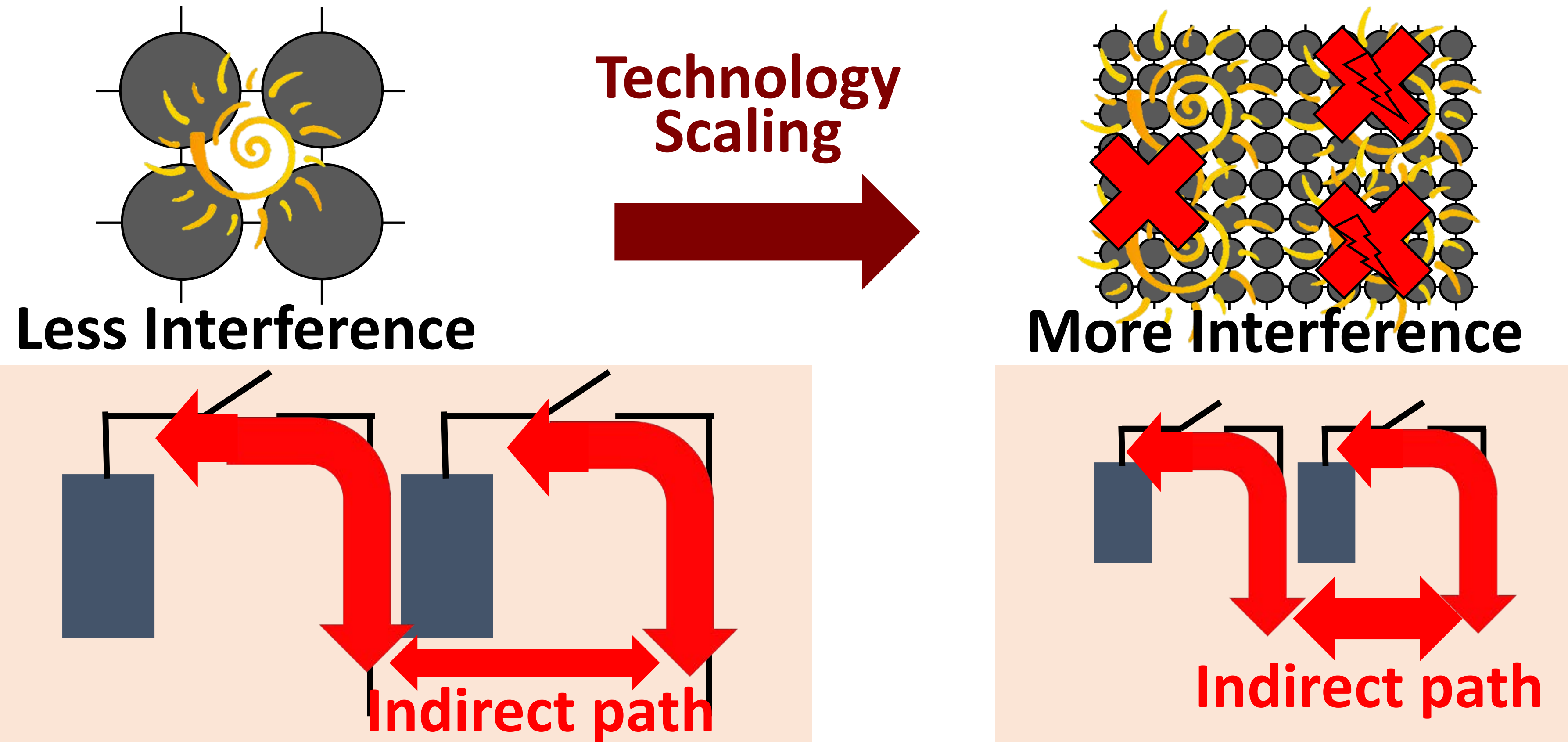
greater than refresh interval

less than refresh interval

Failure depends on the amount of charge

SCALING CHALLENGE: CELL-TO-CELL INTERFERENCE

Cell-to-cell interference affects the charge in neighboring cells



More interference results in more failures

IMPLICATION: DRAM ERRORS IN THE FIELD

DRAM Errors in the Wild: A Large-Scale Field Study

1.52% of DRAM modules failed in Google Servers

A Study of DRAM Failures in the Field

1.6% of DRAM modules failed in LANL

Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field

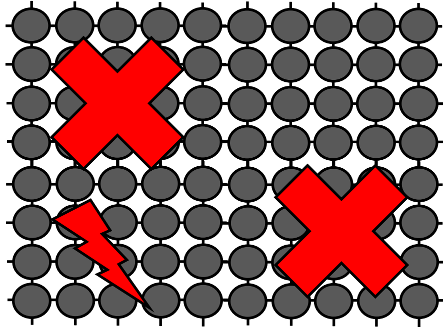
1.8X more failures in new generation DRAMs in Facebook

SIGMETRICS'09, SC'12, DSN'15

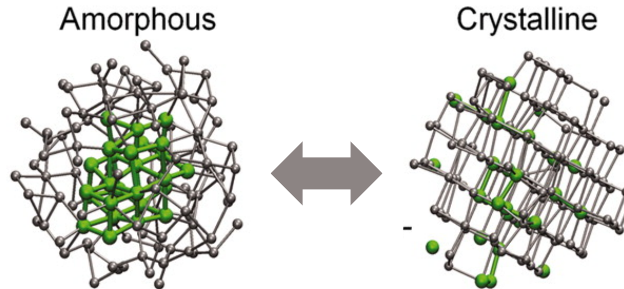
GOAL

Enable
high-capacity, low-latency memory
without
sacrificing reliability

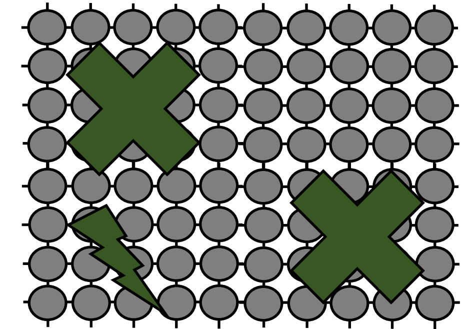
Traditional DRAM Scaling is Ending



MEMORY
Difficult to scale



NON-VOLATILE MEMORY
Highly scalable



MEMORY
Difficult to scale

MAKE DRAM SCALABLE

LEVERAGE NEW TECHNOLOGIES

TOLERATE FAILURES

SIGMETRICS'14, DSN'15, HPCA'15, SIGMETRICS'16, DSN'16, CAL'16, SIGMETRICS'17, HPCA'17, MICRO'17

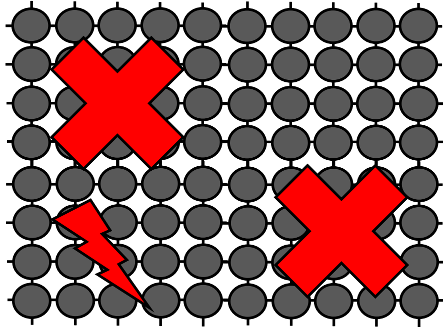
WEED'13, MICRO'15, HPCA'18, ONGOING

WAX'18,ONGOING

Solution Space

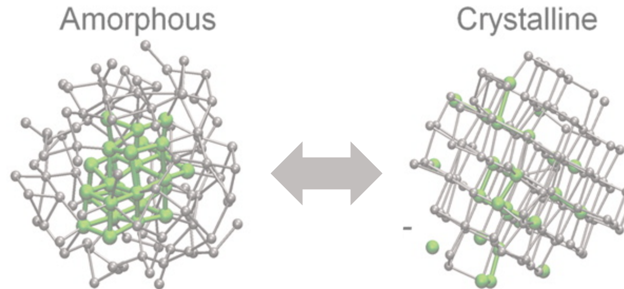
Image: Loke et al., Science 2012

Traditional DRAM Scaling is Ending



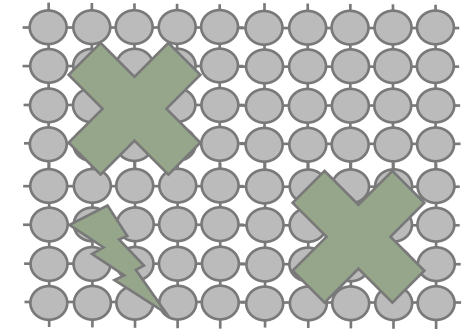
MEMORY

MAKE DRAM SCALABLE



NON-VOLATILE MEMORY

LEVERAGE NEW TECHNOLOGIES



MEMORY

TOLERATE FAILURES

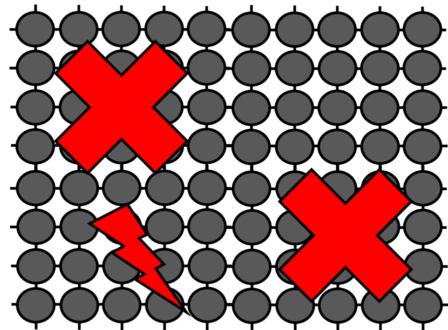
System-Level Detection and Mitigation of Failures

Unifying Memory and Storage with NVM

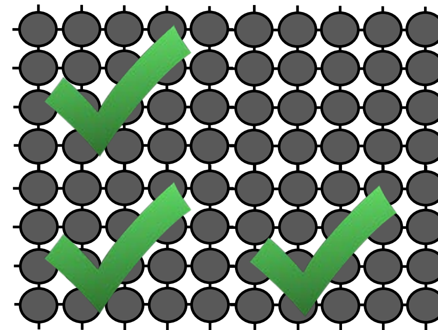
Restricted Approximation

Solution Space

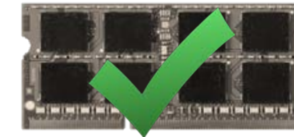
TRADITIONAL APPROACH TO ENABLE DRAM SCALING



Unreliable DRAM Cells



Reliable DRAM Cells



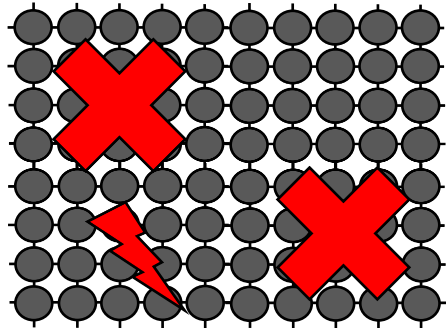
Reliable System

Manufacturing Time

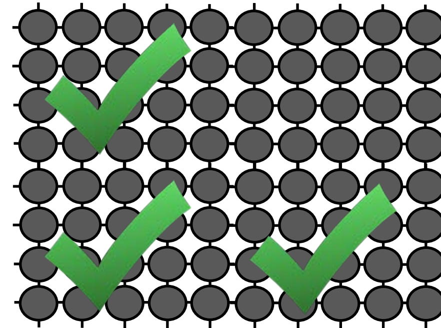
System in the Field

DRAM has strict reliability guarantee

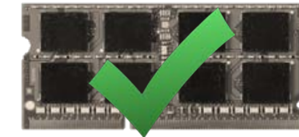
MY APPROACH



Unreliable
DRAM Cells



Reliable
DRAM Cells



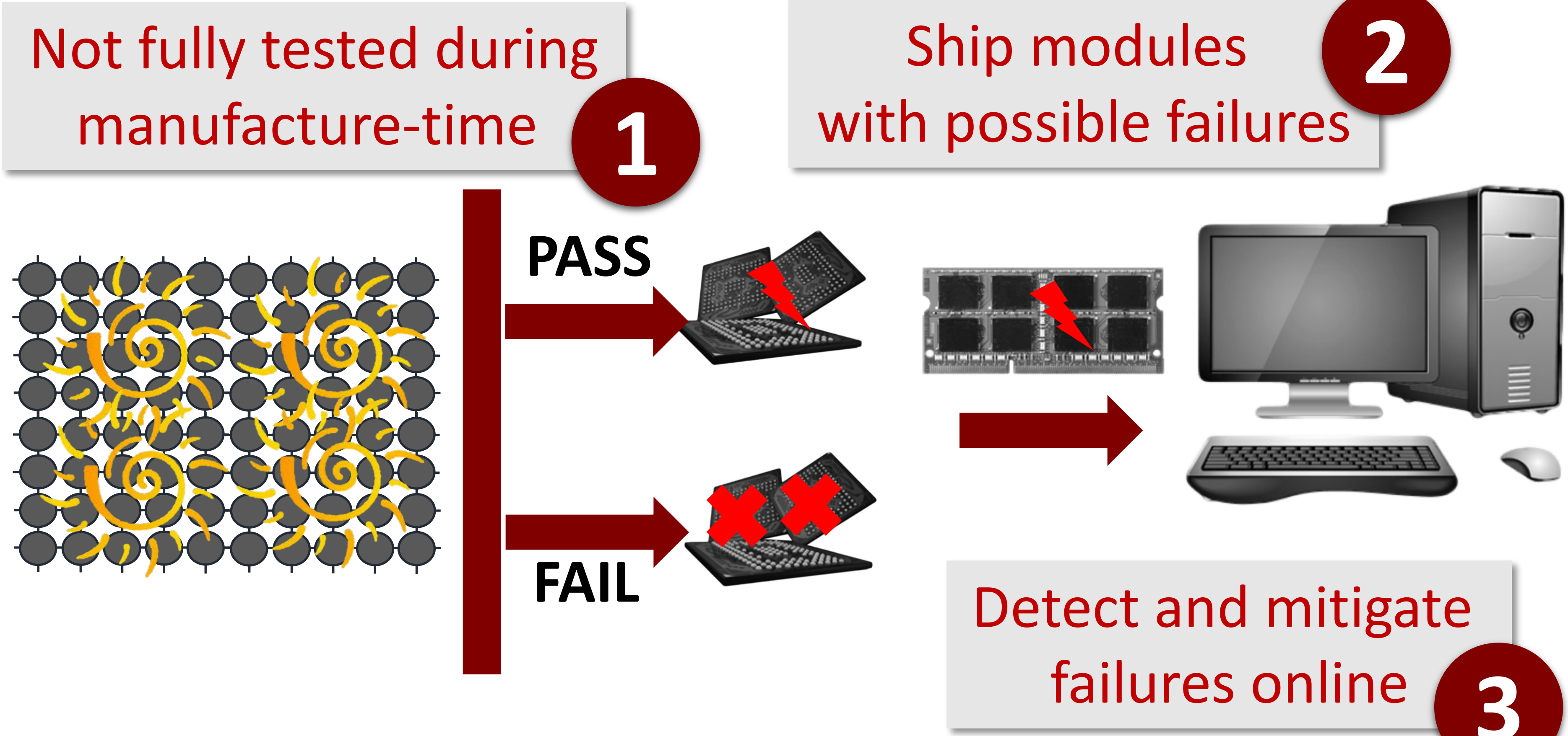
Reliable System

Manufacturing
Time

System
in the Field

Shift the responsibility to systems

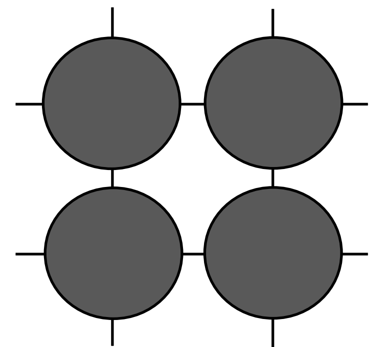
VISION: SYSTEM-LEVEL DETECTION AND MITIGATION



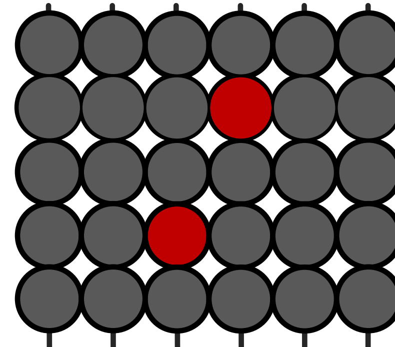
Detect and mitigate errors after the system has become operational

ONLINE PROFILING

BENEFITS OF ONLINE PROFILING



Reliable
DRAM Cells



Unreliable
DRAM Cells

1. Improves yield, reduces cost, enables scaling

Vendors can make cells smaller without a strong reliability guarantee

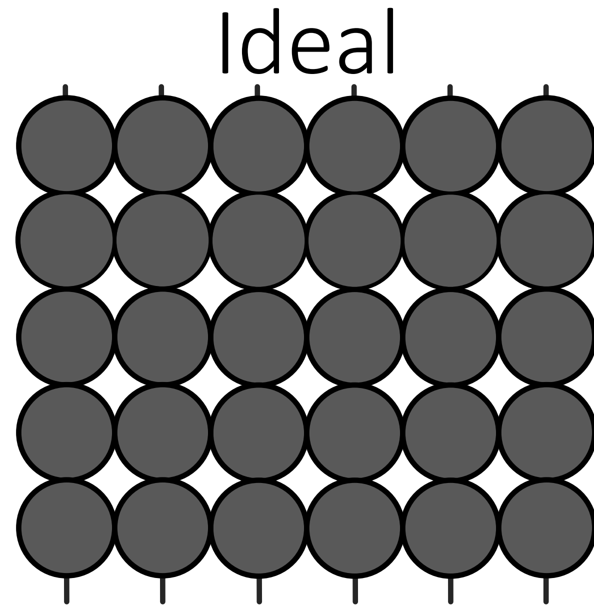
BENEFITS OF ONLINE PROFILING

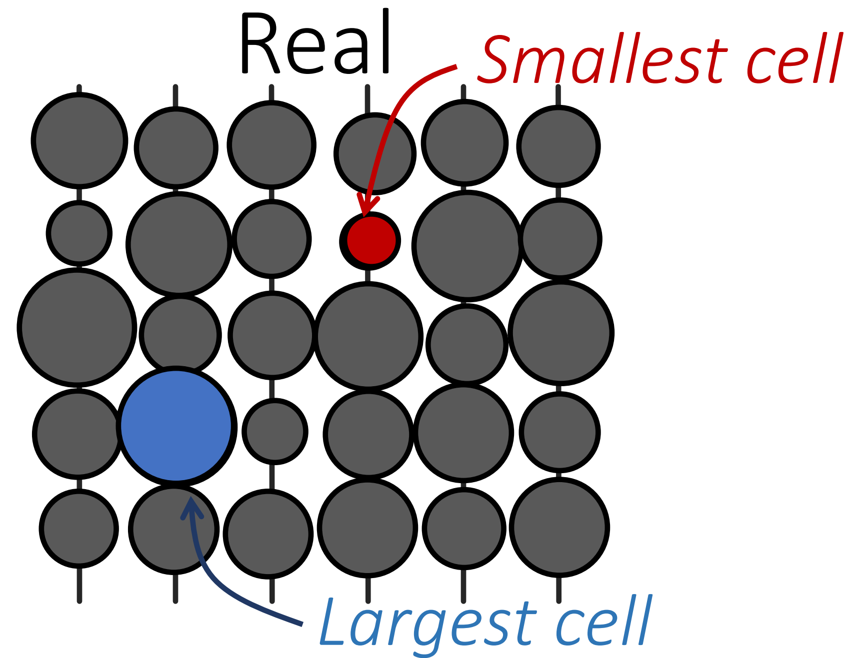
1. Improves yield, reduces cost, enables scaling

Vendors can make cells smaller without a strong reliability guarantee

2. Improves performance and energy efficiency

DRAM CELLS ARE NOT EQUAL



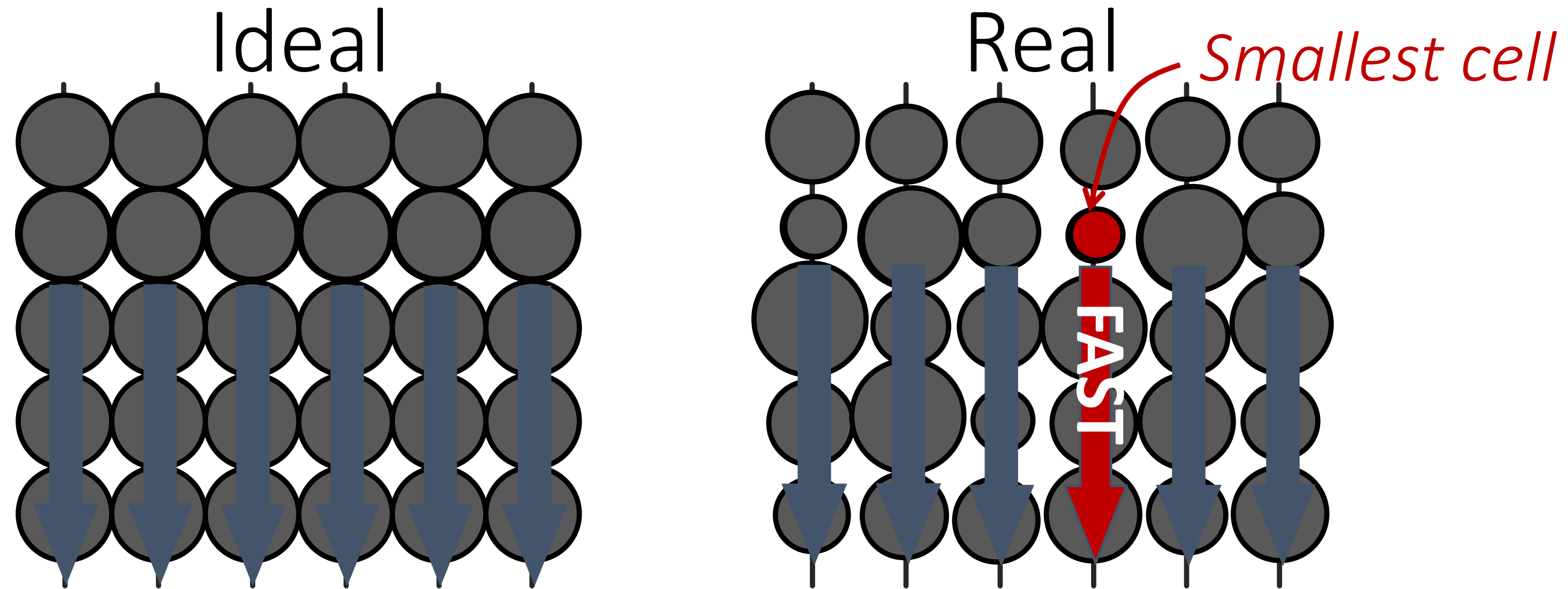


Same size →
Same charge →

Different size →
Different charge →

Large variation in DRAM cells

DRAM CELLS ARE NOT EQUAL

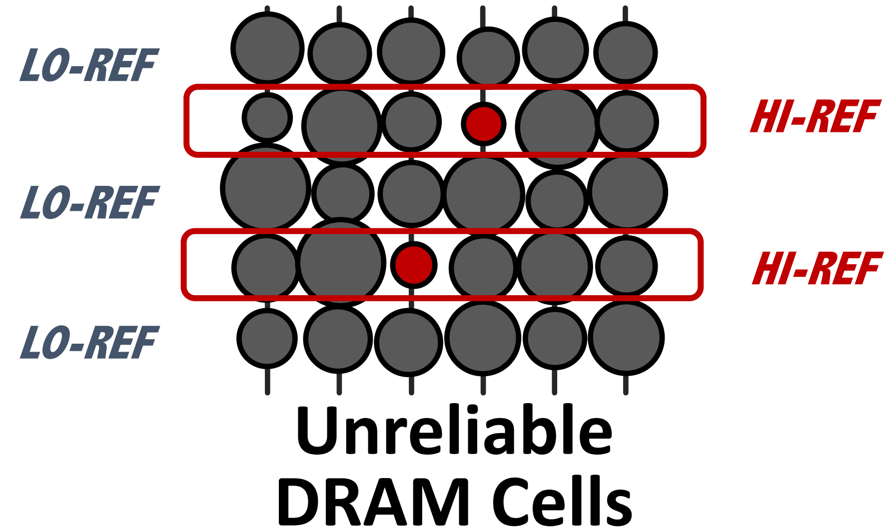


Large variation in retention time

Most cells have high retention time →
can be refreshed at a lower rate without any failure

Smaller cells will fail to retain data at a lower refresh rate

BENEFITS OF ONLINE PROFILING



Reduce refresh count by using a lower refresh rate,
but use higher refresh rate for faulty cells

1. Improves yield, reduces cost, enables scaling

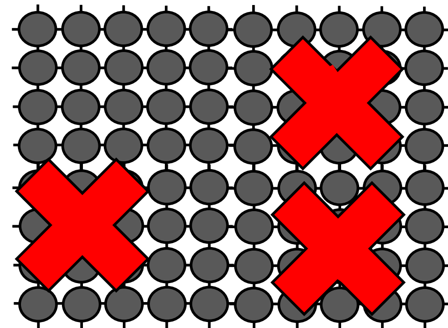
Vendors can make cells smaller without a strong reliability guarantee

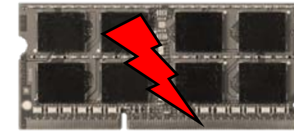
2. Improves performance and energy efficiency

Reduce refresh rate, refresh faulty rows more frequently

In order to enable these benefits,
we need to ***detect*** *the failures*
at the system level

CHALLENGE: INTERMITTENT FAILURES



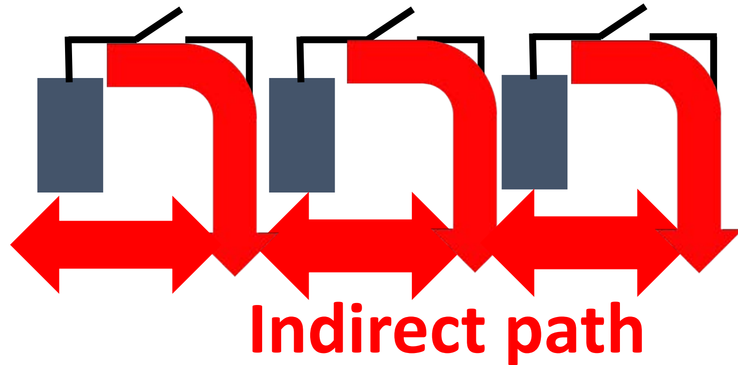


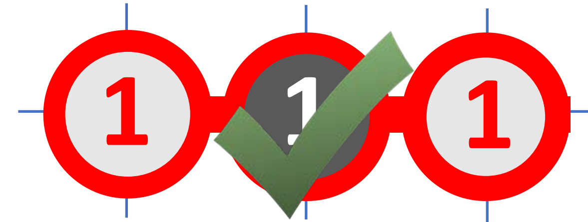
Depends on accurately *detecting* DRAM failures

If failures were permanent,
a simple boot up test would have worked,
but there are intermittent failures

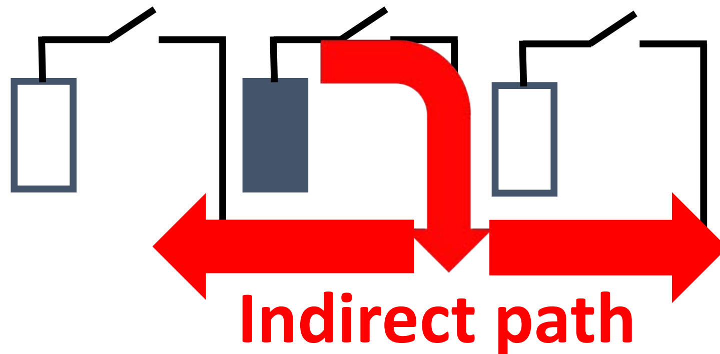
What are the these intermittent failures?

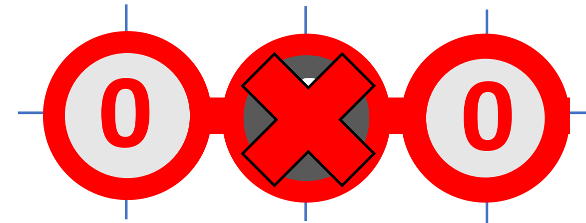
CELL-TO-CELL INTERFERENCE: DATA-DEPENDENT FAILURES





NO FAILURE

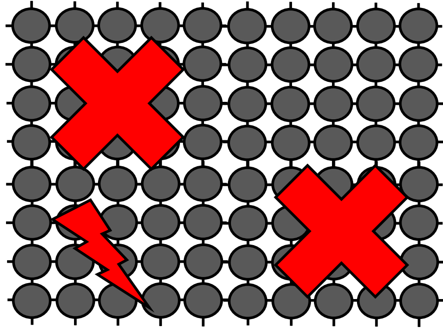




FAILURE

Some cells can fail depending on the data stored in neighboring cells

How to detect these failures at the system?



DRAMMAKE DRAM SCALABLE

System-Level Detection and Mitigation of Failures

CHALLENGE:
Data-Dependent Failures

Efficacy of Testing
Data-Dependent Failures

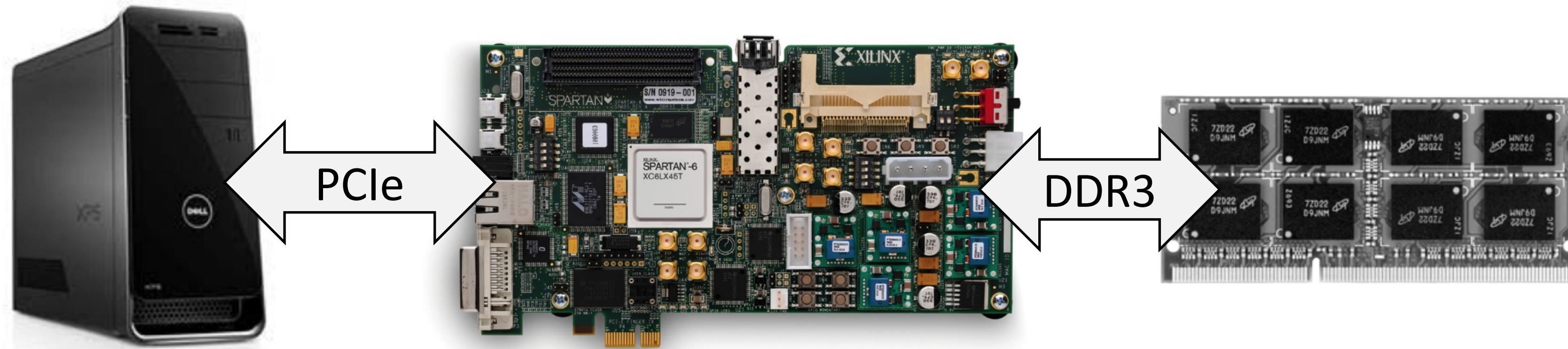
SIGMETRICS’14

MEMCON: DRAM-Internal
Independent Detection

CAL’16, MICRO’17

Experimental Methodology

Custom FPGA-based infrastructure



PC

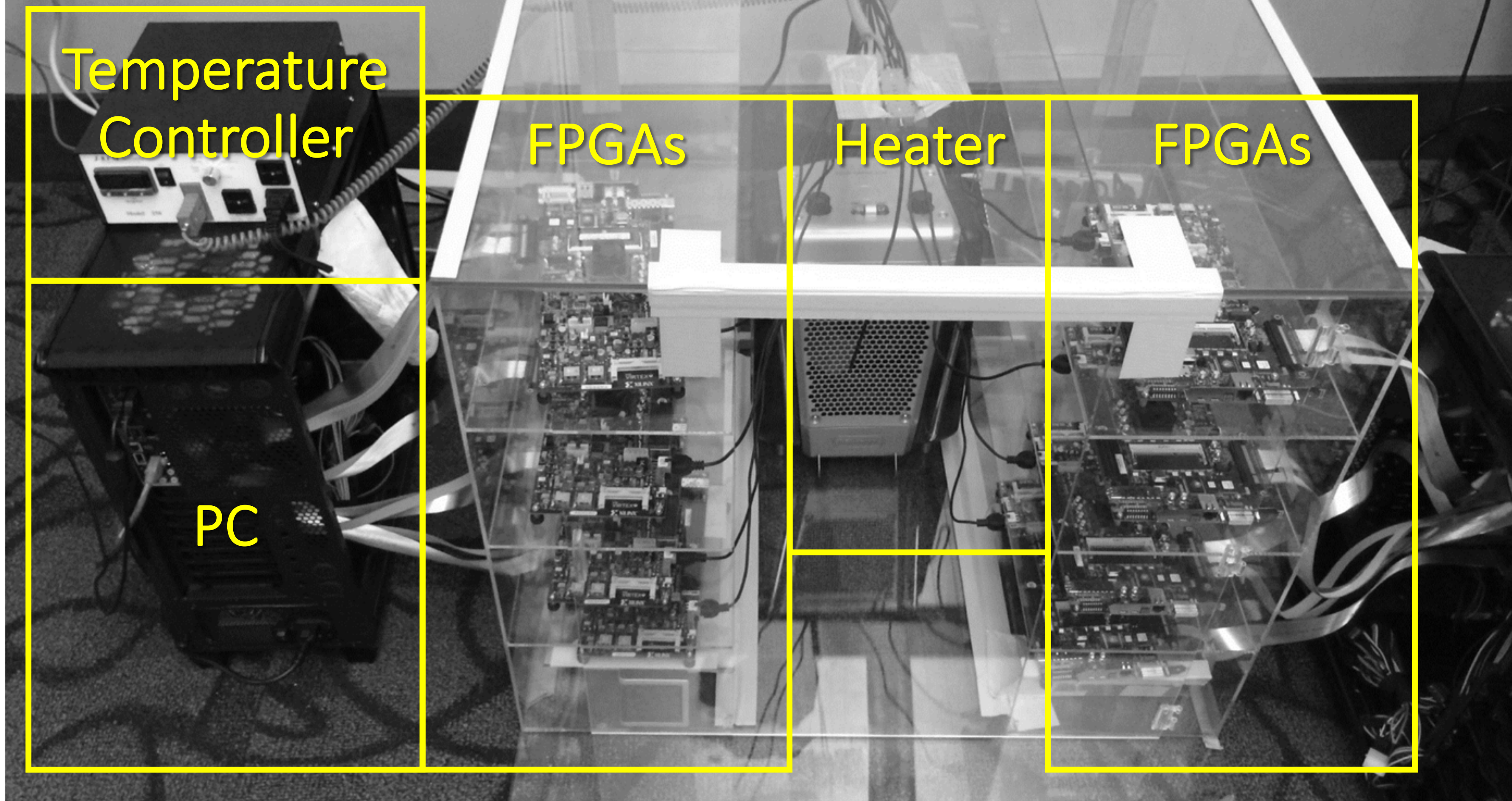
C++ programs to specify commands

FPGA

Generate command sequence

DIMM

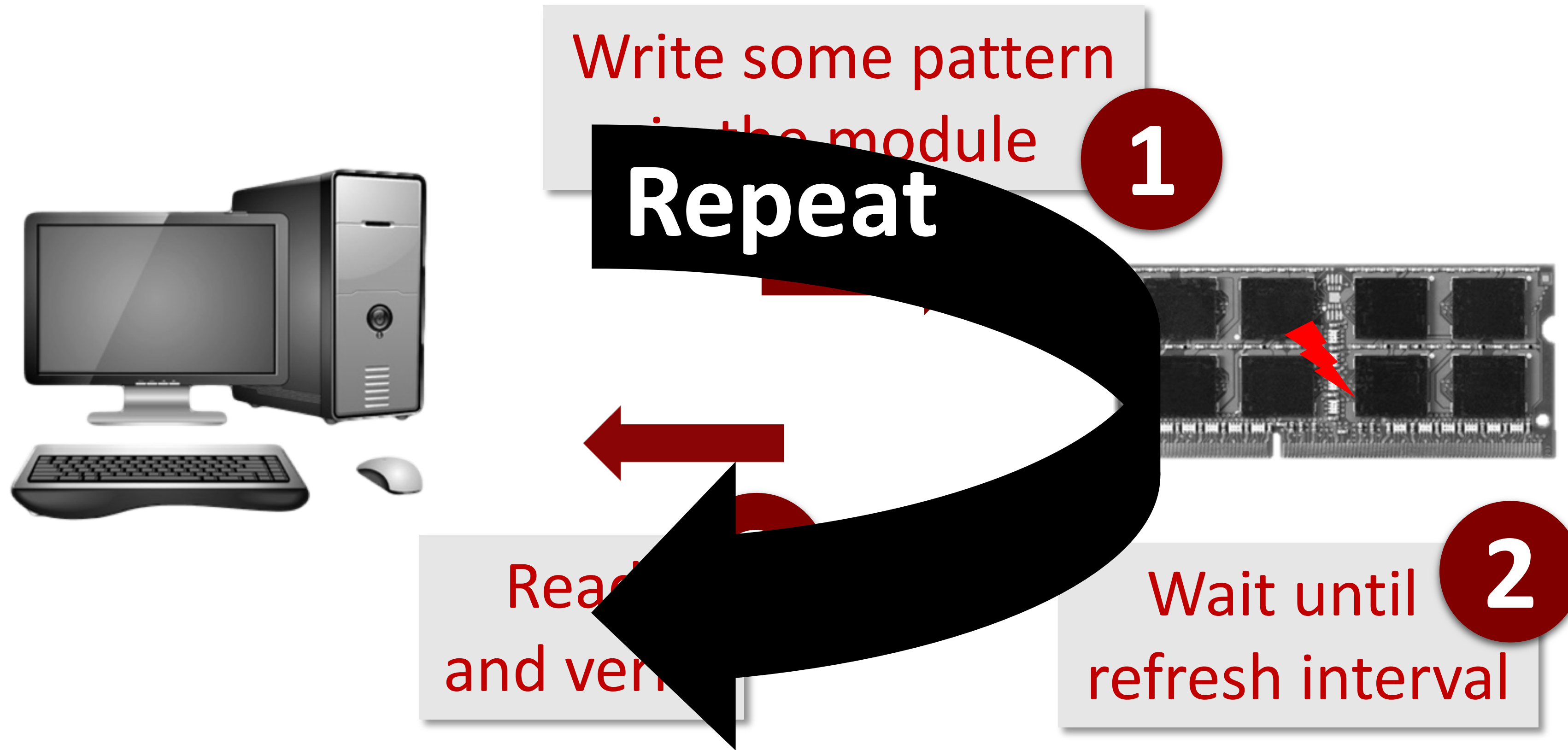
Tested more than hundred chips from three different manufacturers



Open-source infrastructure to test real DRAM chips
Characterization data publicly available

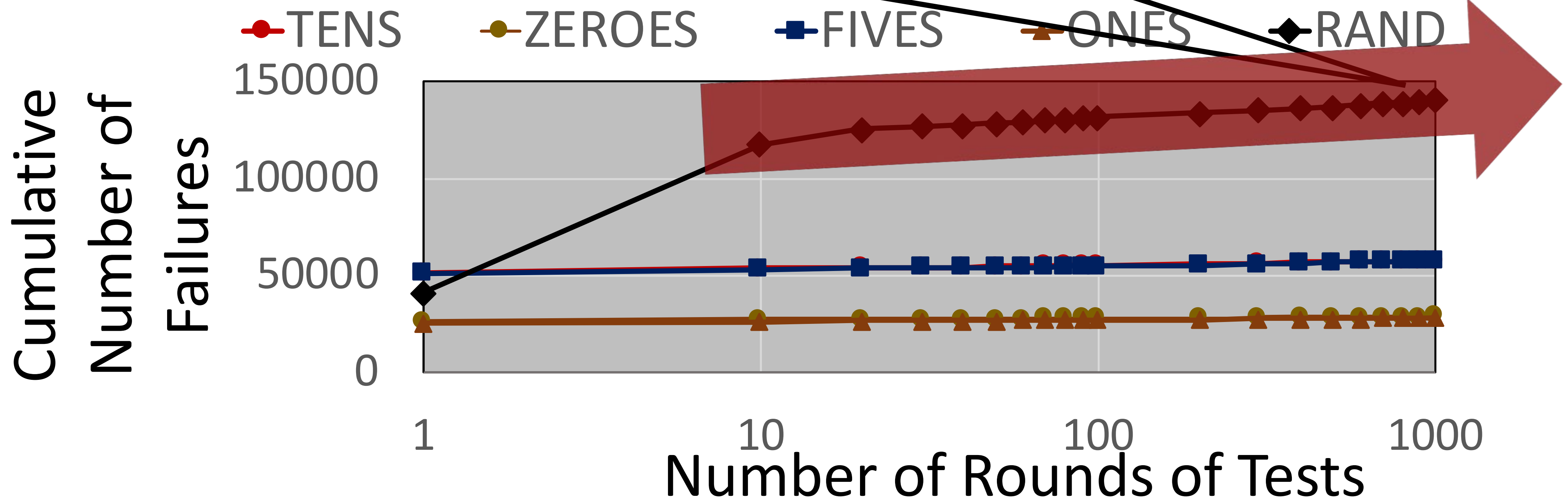
HPCA'17, SIGMETRICS'14, DSN'15, HPCA'15, SIGMETRICS'16, DSN'16, CAL'16, SIGMETRICS'17, MICRO'17

DETECT FAILURES WITH TESTING



Test with different data patterns

DETECTING DATA-DEPENDENT FAILURES

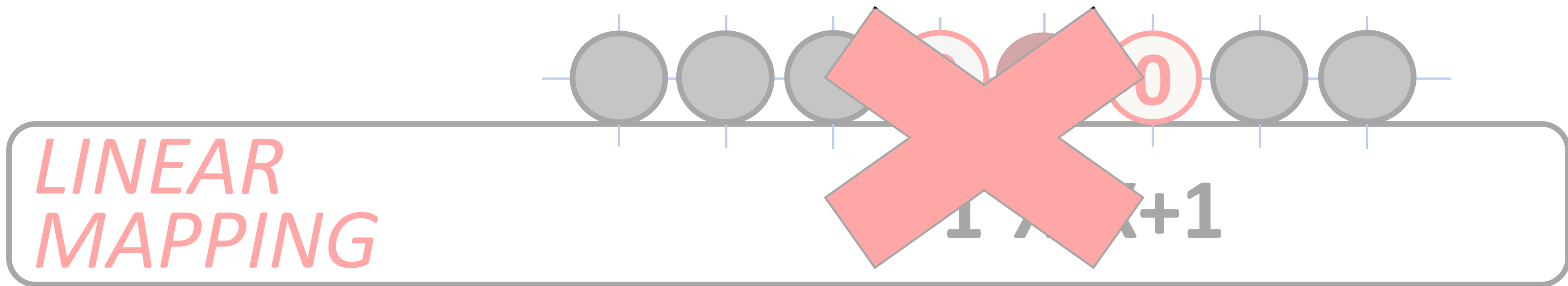


Conclusion: Tests with many rounds of random patterns cannot *detect* all failures

WHY SO MANY ROUNDS OF TESTS?

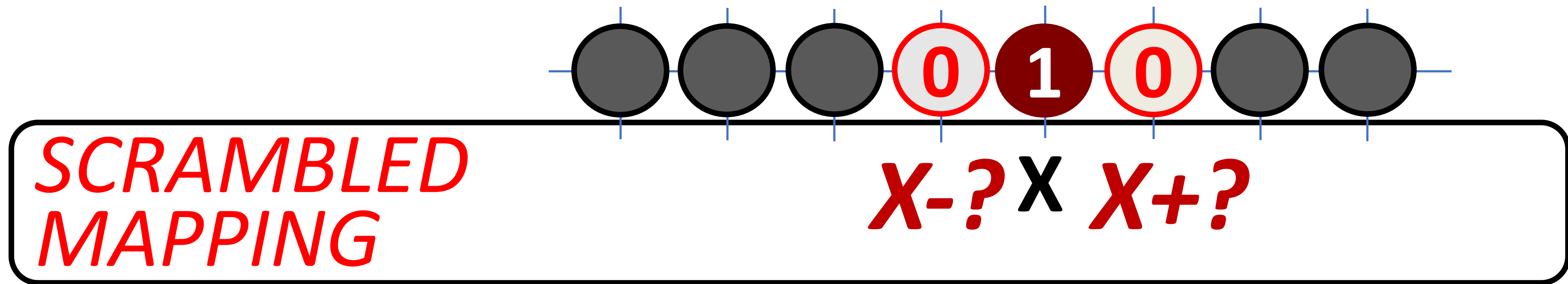
DATA-DEPENDENT FAILURE

Fails when specific pattern in the neighboring cell

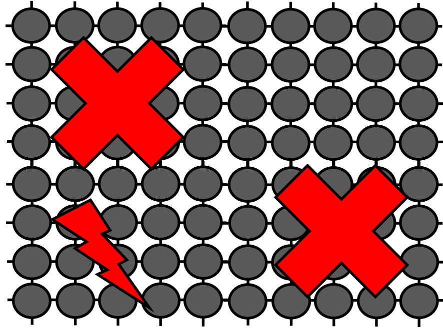


Even many rounds of random patterns cannot detect all failures

CHALLENGE IN DETECTION



How to detect *data-dependent failures* when we even do not know which *cells are neighbors*?



DRAM

MAKE DRAM SCALABLE

System-Level Detection and Mitigation of Failures

CHALLENGE:
Data-Dependent Failures

Efficacy of Testing Data-Dependent Failure

SIGMETRICS'14

MEMCON: DRAM-Internal Independent Detection

CAL'16, MICRO'17

CURRENT DETECTION MECHANISM

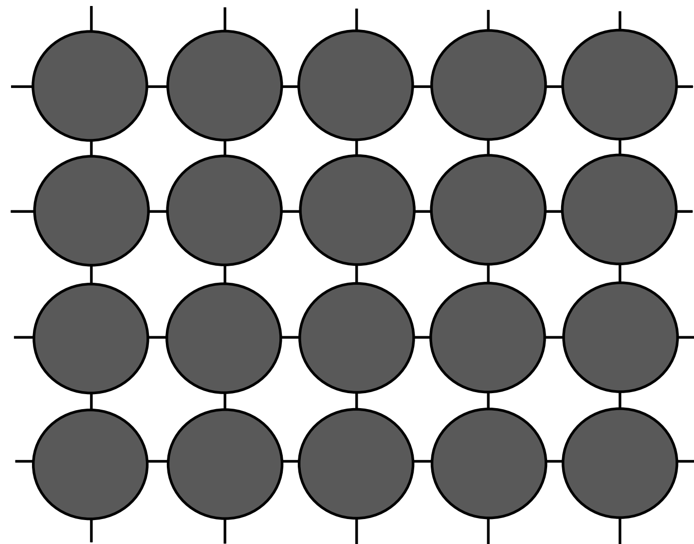
Initial Failure Detection and Mitigation	Execution of Applications

Detection is done with some initial testing isolated from system execution

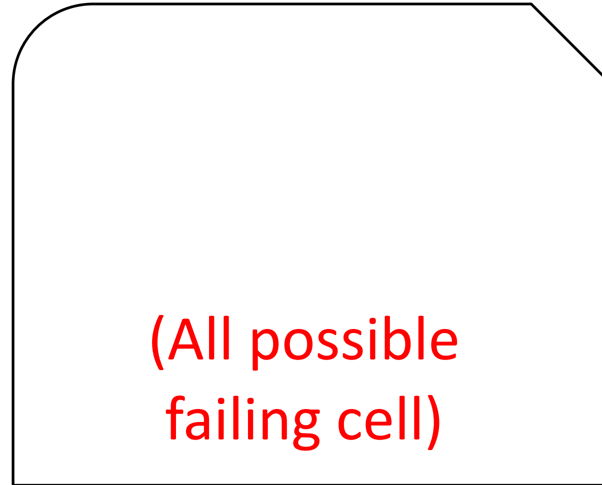
1. Detect and mitigate all failures with every possible content
2. Only after that start program execution

CURRENT DETECTION MECHANISM

Detect every possible failure with all content before execution



Unreliable DRAM Cells

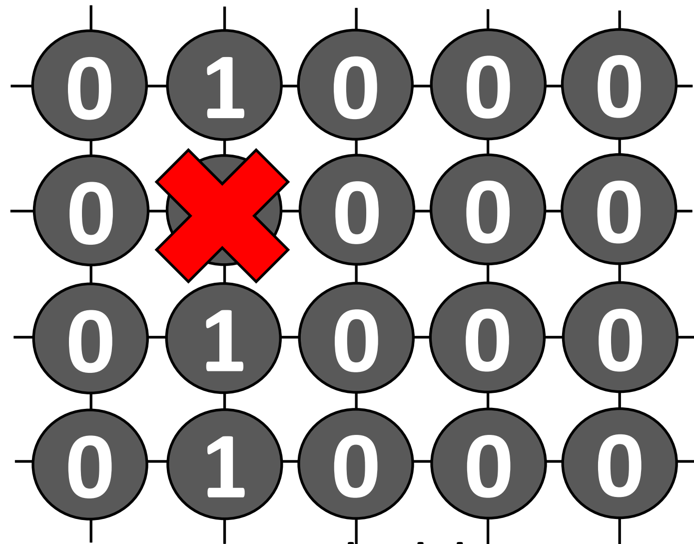


List of Failures

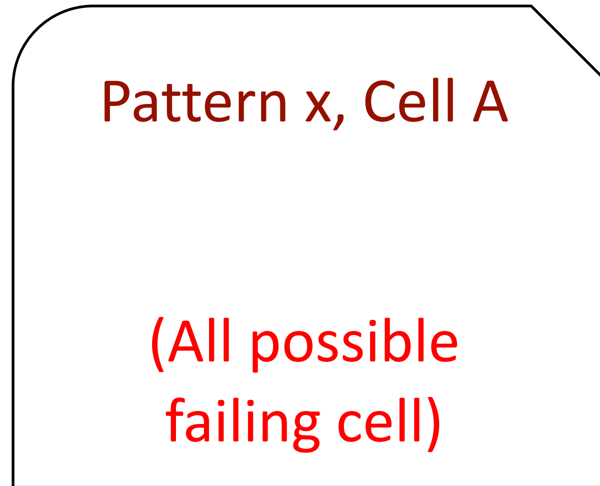
Initial Failure Detection and Mitigation

CURRENT DETECTION MECHANISM

Detect every possible failure with all content before execution



Unreliable
DRAM Cells

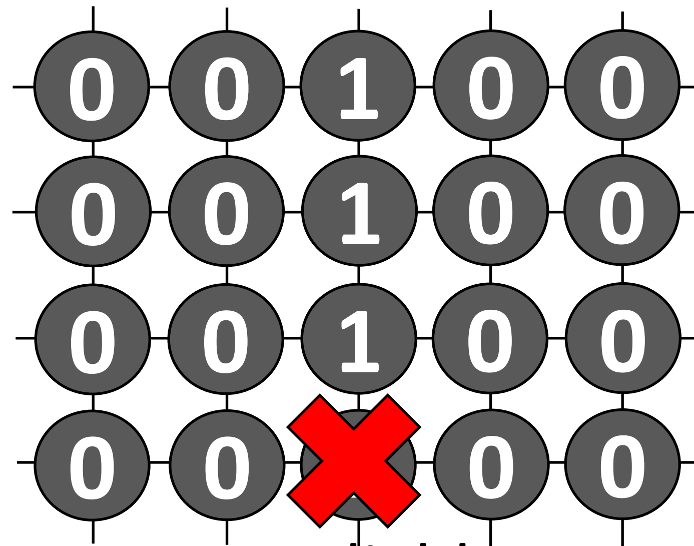


List of Failures

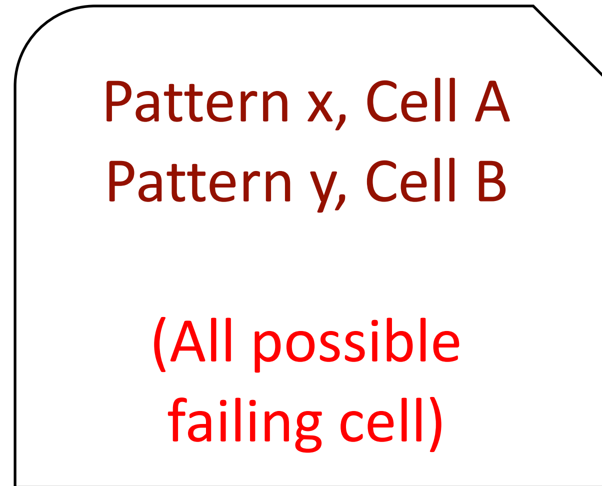
Initial Failure Detection and Mitigation

CURRENT DETECTION MECHANISM

Detect every possible failure with all content before execution



Unreliable
DRAM Cells

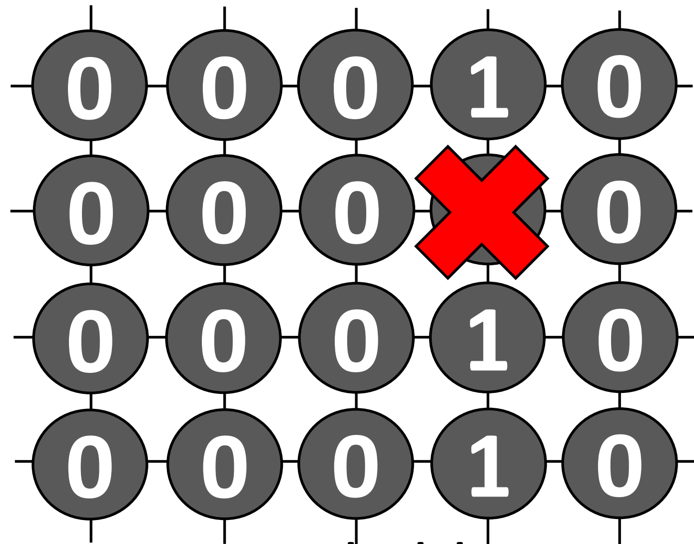


List of Failures

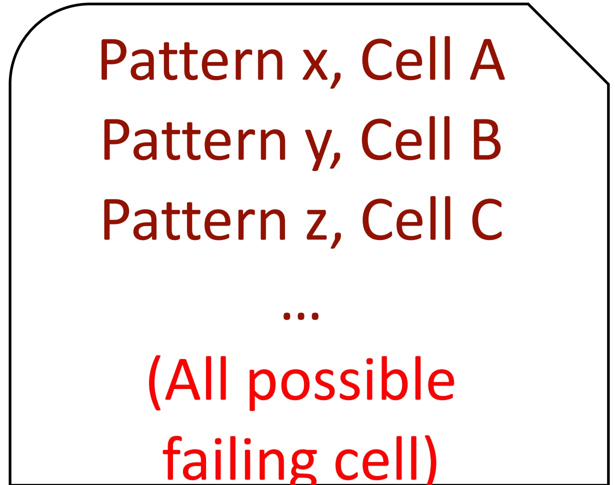
Initial Failure Detection and Mitigation

CURRENT DETECTION MECHANISM

Detect every possible failure with all content before execution



Unreliable DRAM Cells



List of Failures

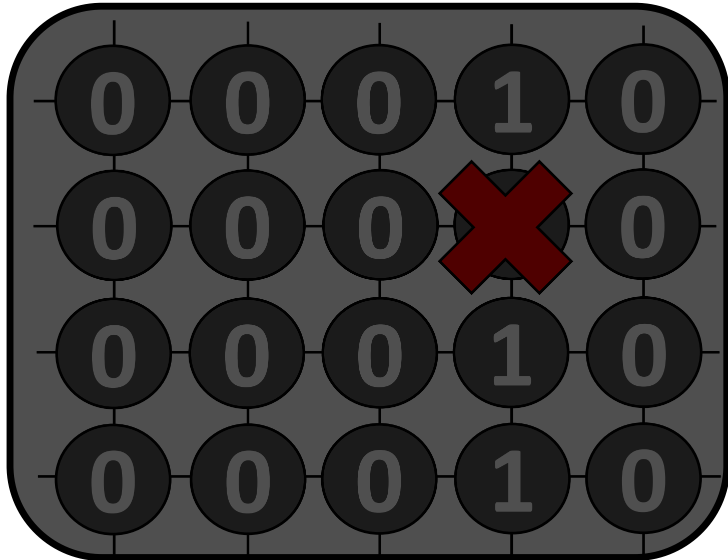
Applications

Initial Failure Detection and Mitigation

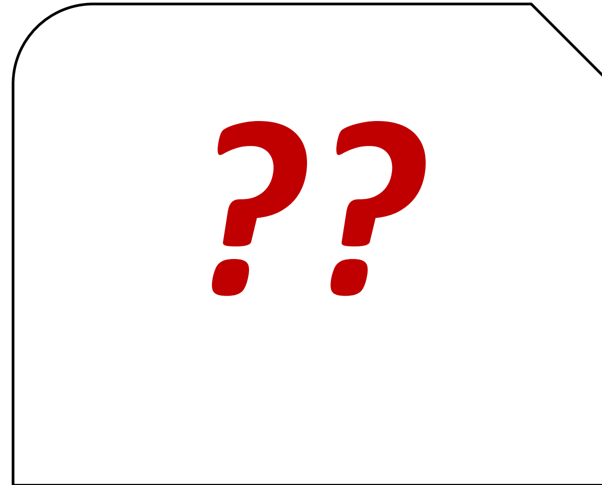
Execution of Applications

CURRENT DETECTION MECHANISM

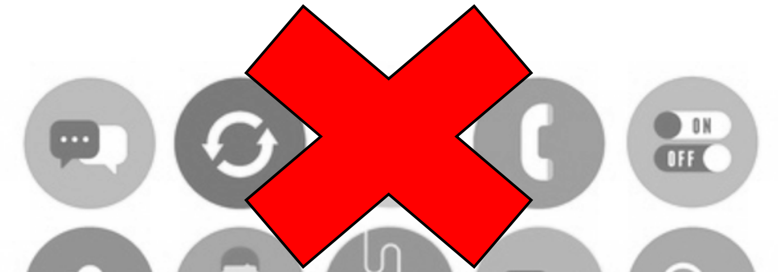
Detect every possible failure with all content before execution



Unreliable
DRAM Cells



List of Failures



No Reliability Guarantee

Applications

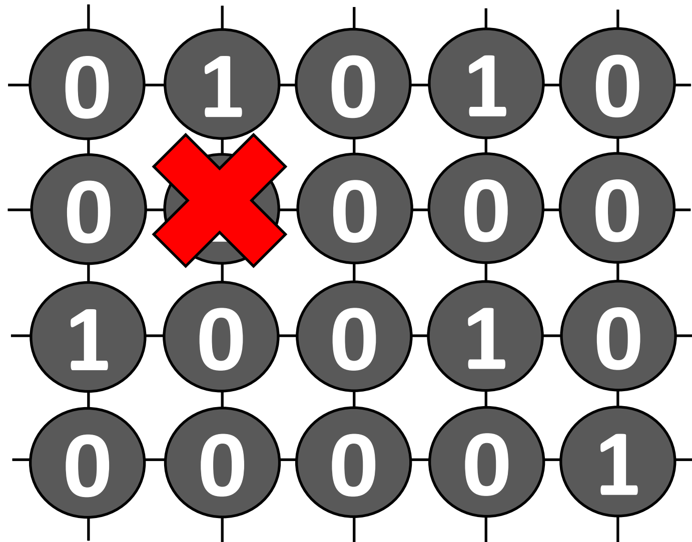
Initial Failure Detection and Mitigation

Execution of Applications

Online profiling cannot detect ***all*** failures as the address mapping is not visible to the system

MEMCON: MEMORY CONTENT-BASED DETECTION AND MITIGATION

NO NEED TO DETECT EVERY POSSIBLE FAILURE



Unreliable DRAM Cells with Program Content

Current content, Cell A

List of Failures

Application

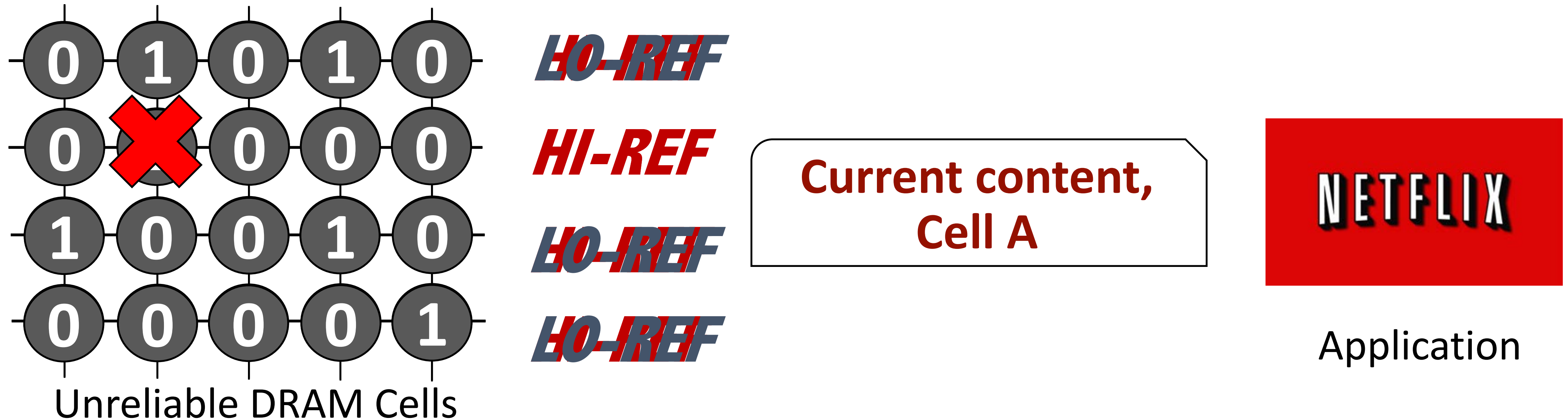
Simultaneous Detection and Execution

Based on current memory content of running applications

Need to detect and mitigate only with the current content

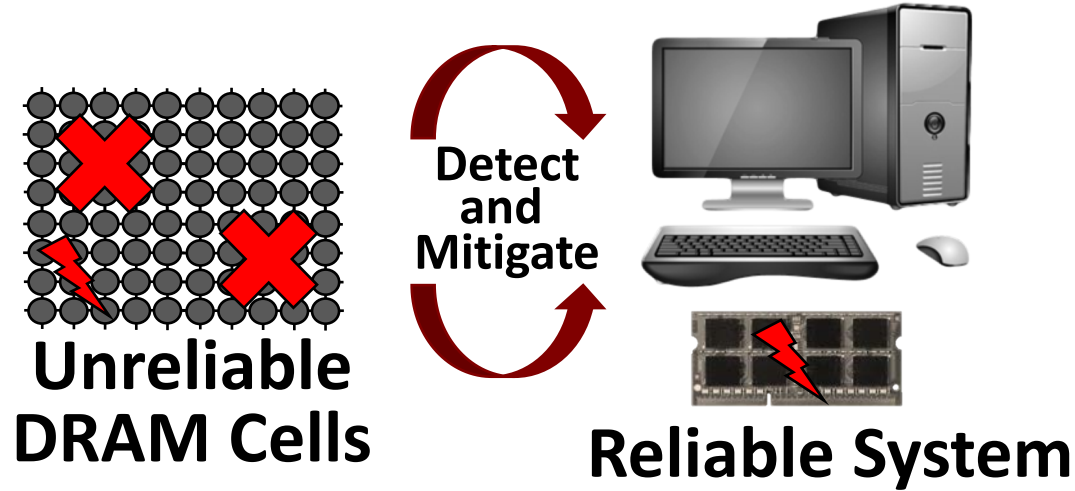
MEMCON: HIGH-LEVEL DESIGN

Simultaneous Detection and Execution



1. No initial detection and mitigation
2. Start running the application with a high refresh rate
3. Detect failures with the current memory content
   - If no failure found, use a low refresh rate

SUMMARY: ONLINE PROFILING



Detection at the system-level is challenging due to data-dependent failures

It is possible to ***detect and mitigate*** data-dependent failures ***simultaneously with program execution***

65%-74%
Reduction in refresh count

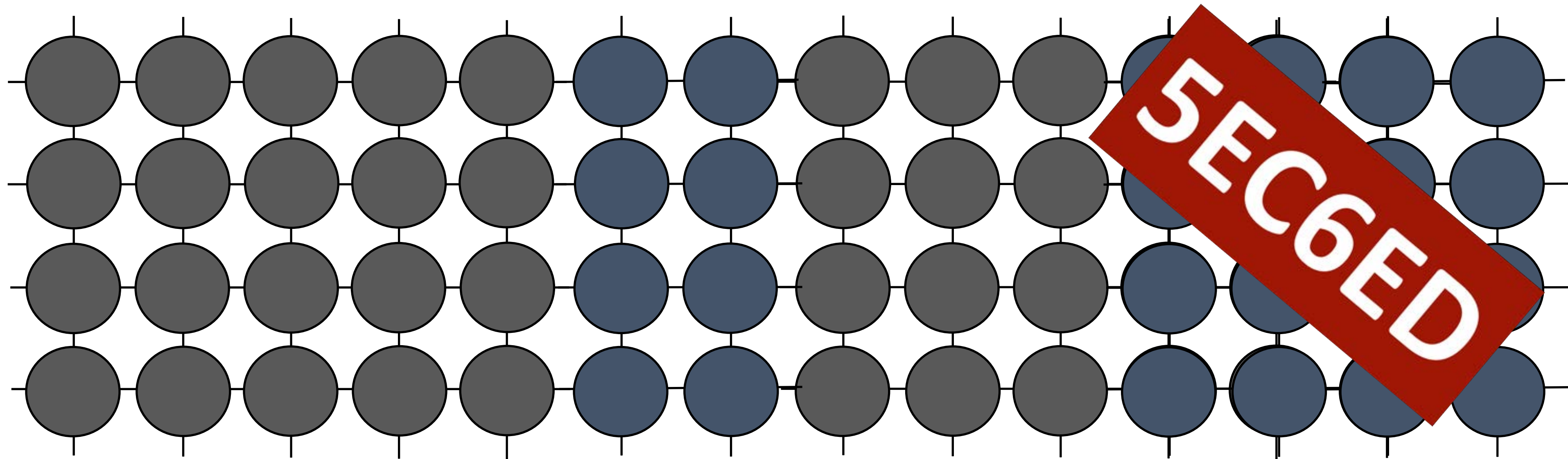
40%-50%
Performance improvement

SOLVING THE DRAM SCALING CHALLENGE

Samira Khan

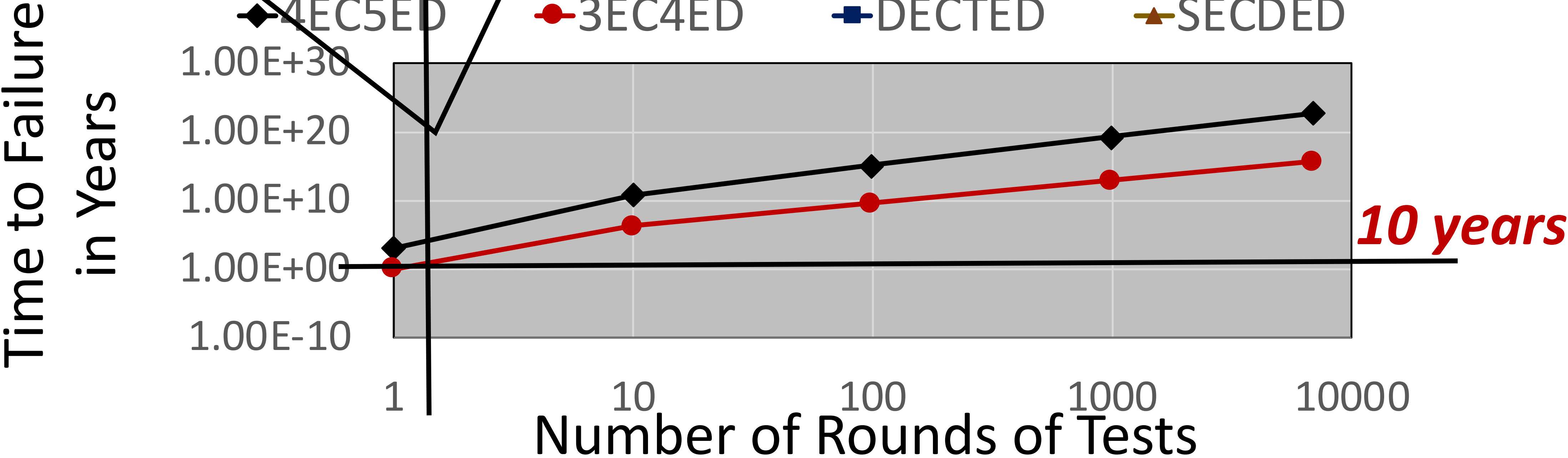
WHY NOT USE ECC?

No testing, use strong ECC
But amortize cost of ECC over larger data chunk

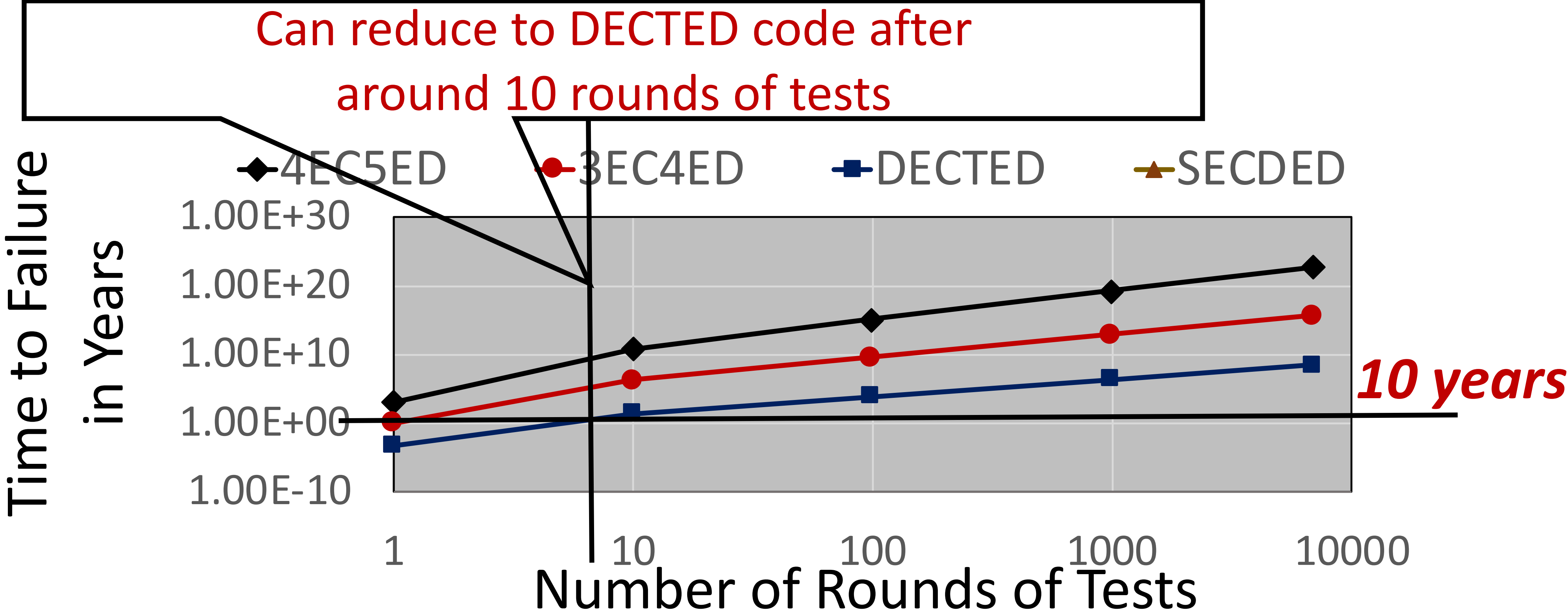


Can potentially tolerate errors
at the cost of higher strength ECC

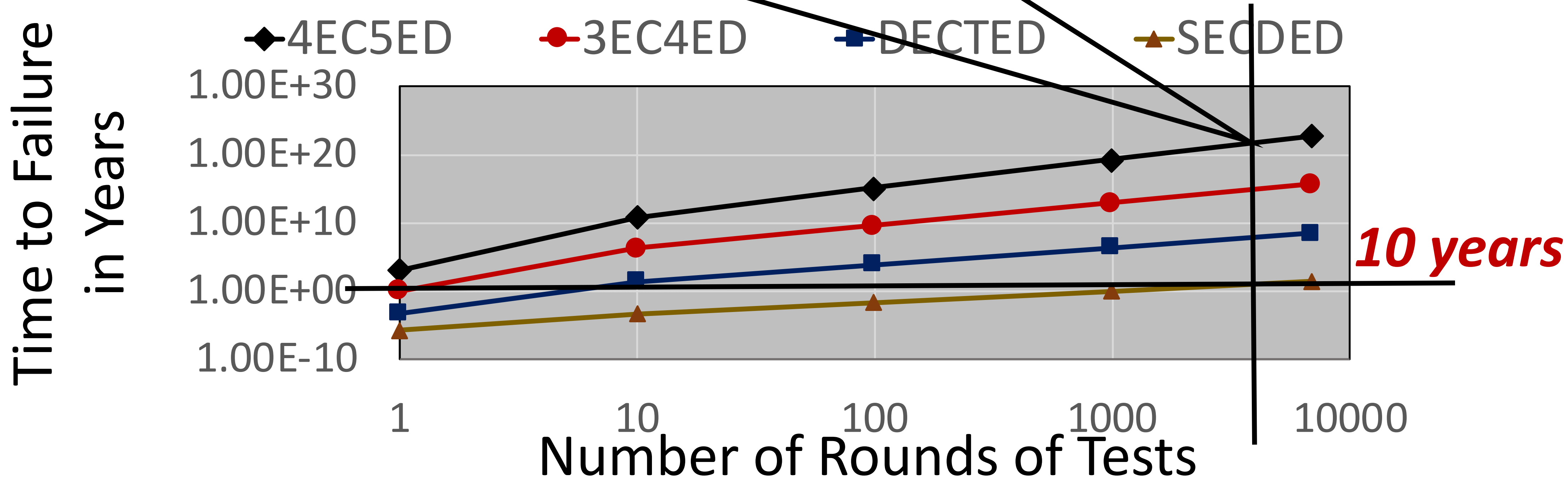
DETECTING DATA-DEPENDENT FAILURES



DETECTING DATA-DEPENDENT FAILURES

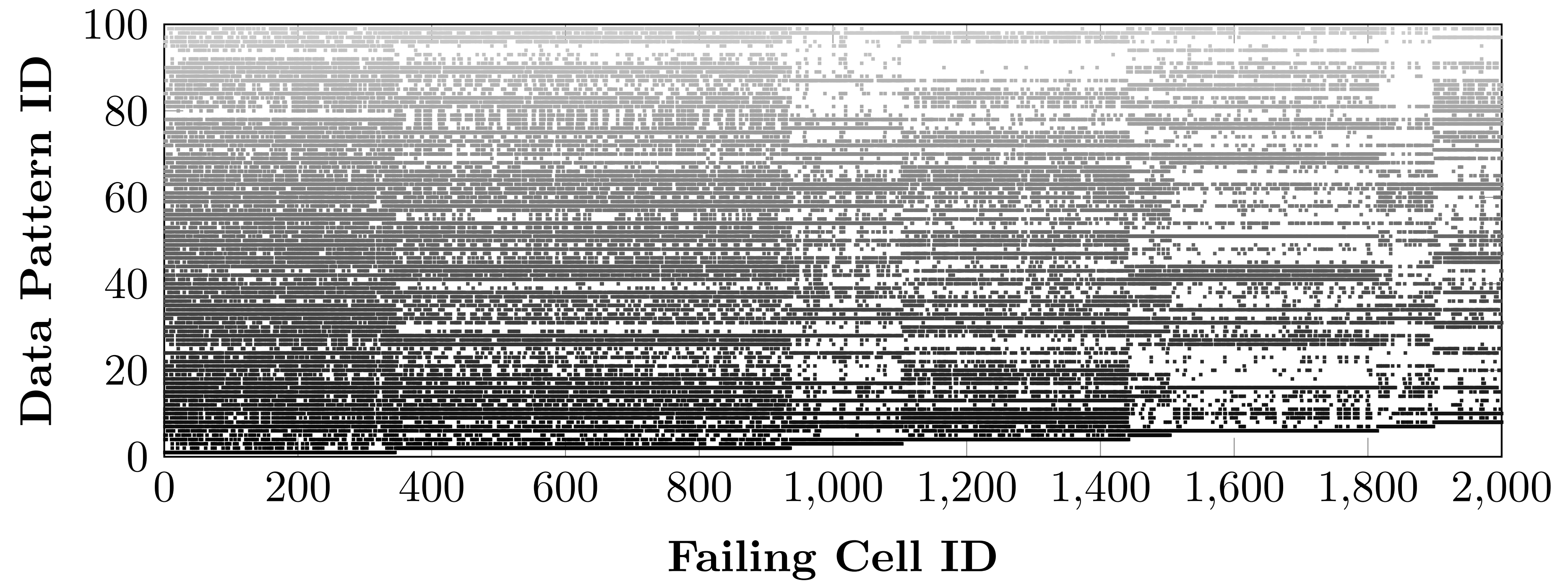


DETECTING DATA-DEPENDENT FAILURES



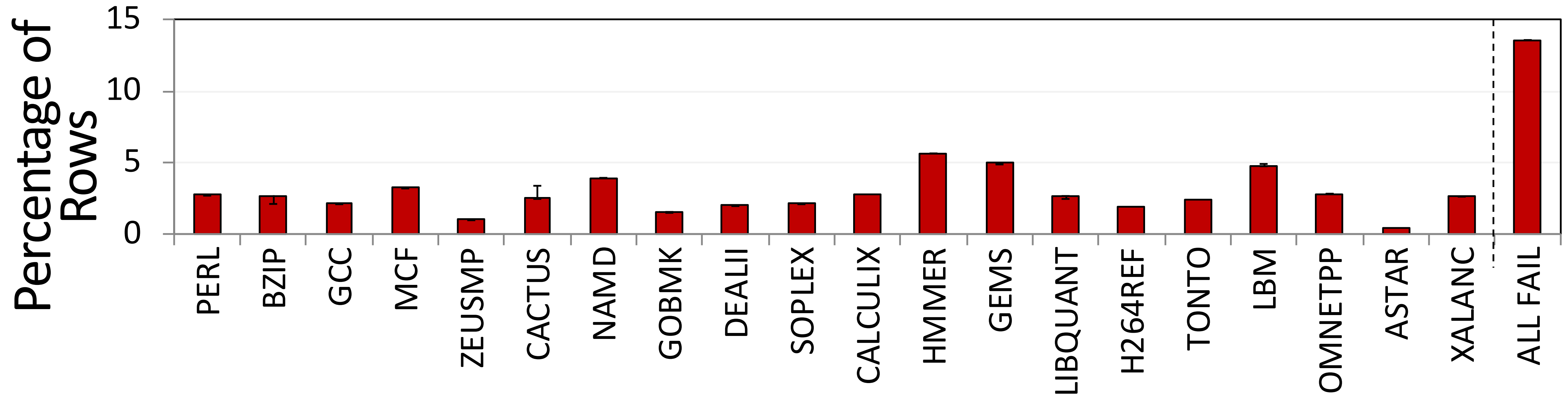
Conclusion: Testing can help to reduce the ECC strength, but *blocks memory for hours*

DATA-DEPENDENT FAILURE



NUMBER OF FAILING ROWS WITH PROGRAM CONTENT

Tested with program content in real DRAM chips

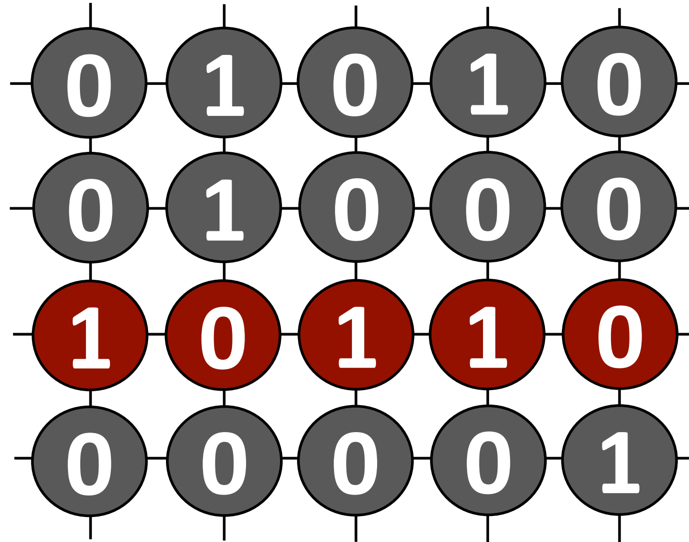


Program content exhibits significantly less failures

CHALLENGE WITH MEMCON

NEED TO DETECT FAILURE AT A WRITE

Write access to row A



Current content, Cell ??

NETFLIX

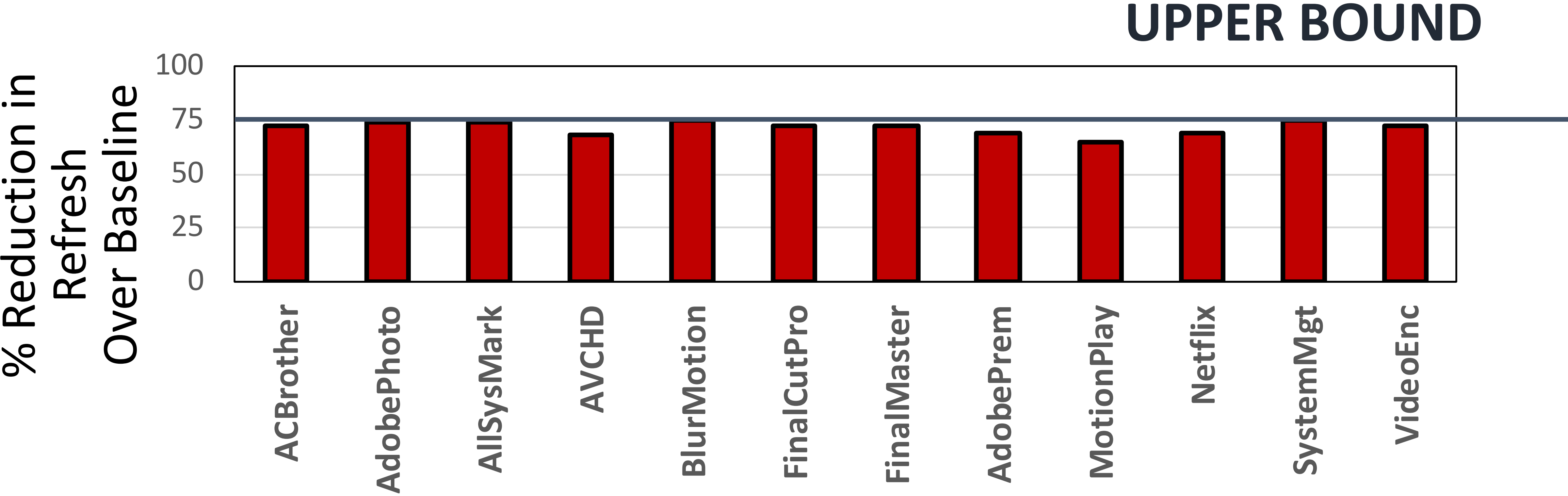
Testing at every write is expensive!!!

MEMCON: MECHANISM

Does not test at every write

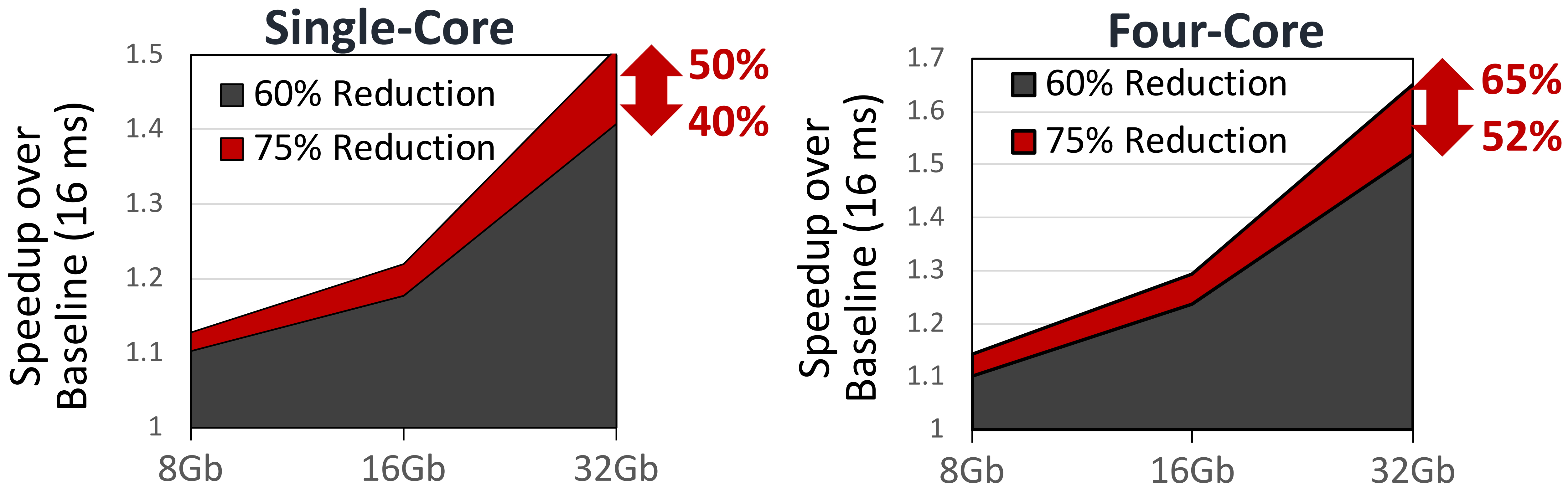
MEMCON *selectively* initiates testing when the write interval is *long enough to amortize* the cost of testing

MEMCON: REDUCTION IN REFRESH COUNT



On average 71% reduction in refresh count, very close to the upper bound of 75%

MEMCON: PERFORMANCE IMPROVEMENT DUE TO REDUCTION IN REFRESH COUNT

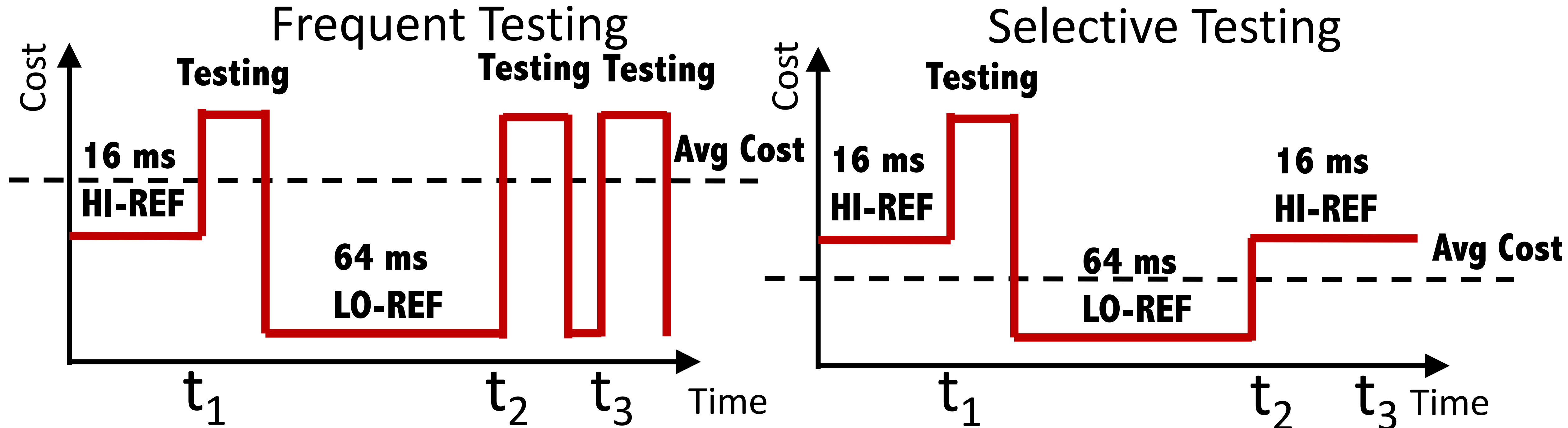


Refresh reduction leads to significant performance improvement

MEMCON: COST-BENEFIT ANALYSIS

Cost: Extra memory accesses to read and write rows

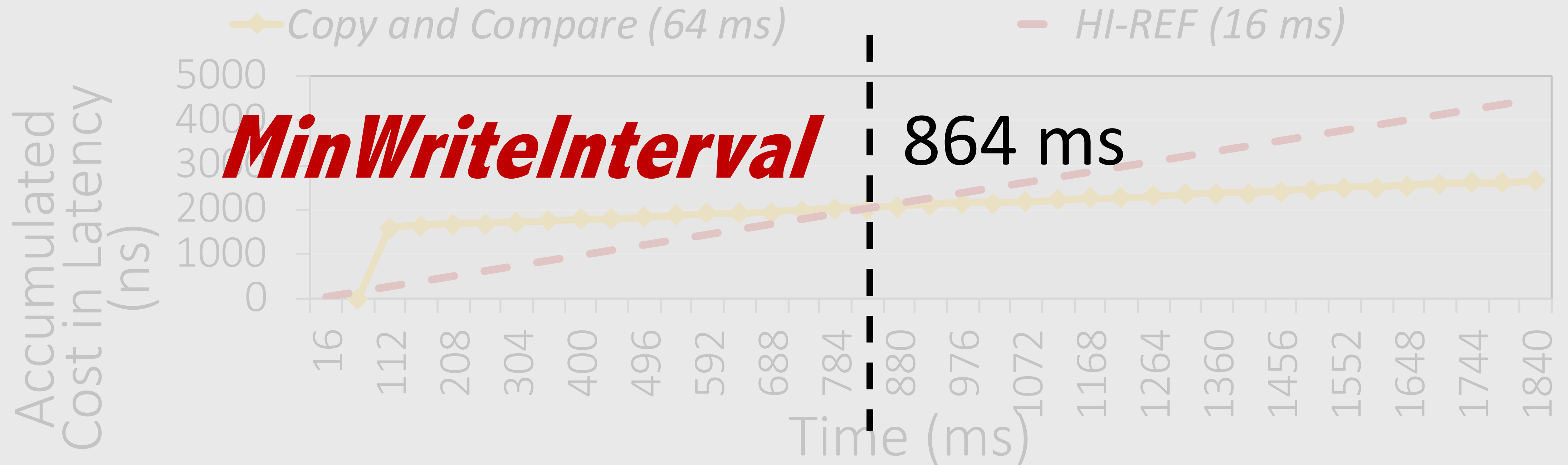
Benefit: If no failure found, can reduce refresh rate



Initiate a test only when the cost can amortized

MEMCON: COST BENEFIT ANALYSIS

What is the write interval that can amortize the cost?



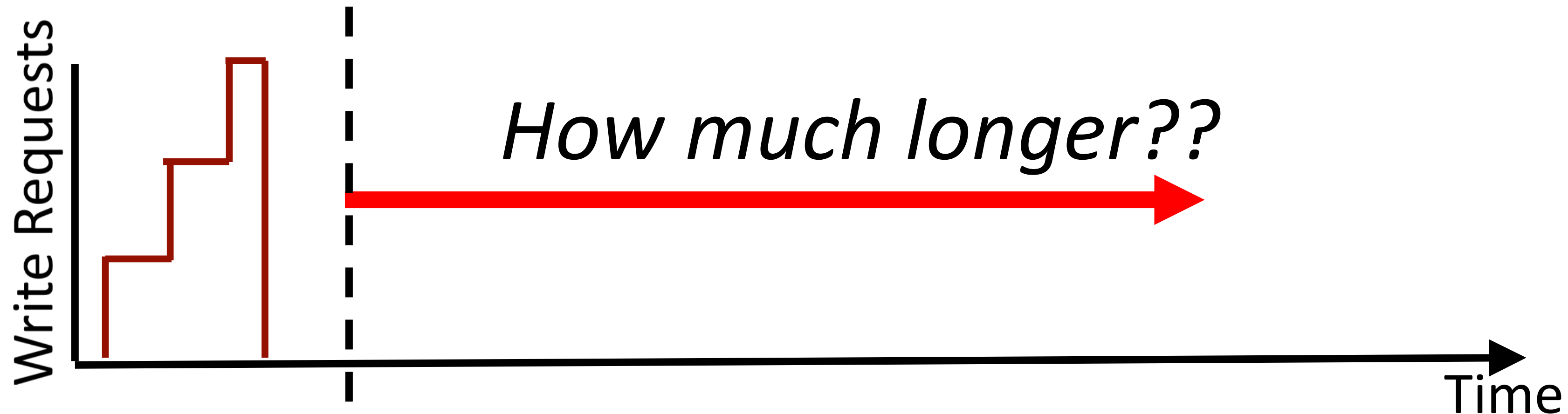
MEMCON: MECHANISM

Does not test at every write

MEMCON *selectively* initiates testing when the write interval is *long enough to amortize* the cost of testing

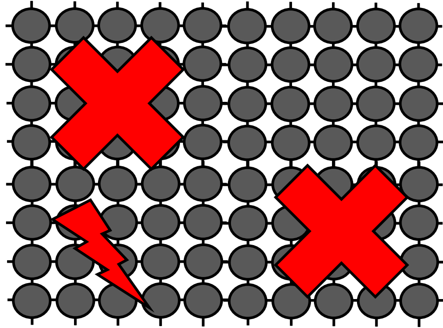
MEMCON: MECHANISM

MinWriteInterval is quite long



MEMCON *selectively* initiates testing when the write interval is *long enough to amortize* the cost of testing

How do we predict the interval on a write access?



DRAM

MAKE DRAM SCALABLE

System-Level Detection and Mitigation of Failures

PROBLEM: SCRAMBLED ADDRESS MAPPING

MEMCON: DRAM-Internal Independent Detection

CAL'16, MICRO'17

GOAL

KEY IDEA

CHALLENGE

MECHANISM

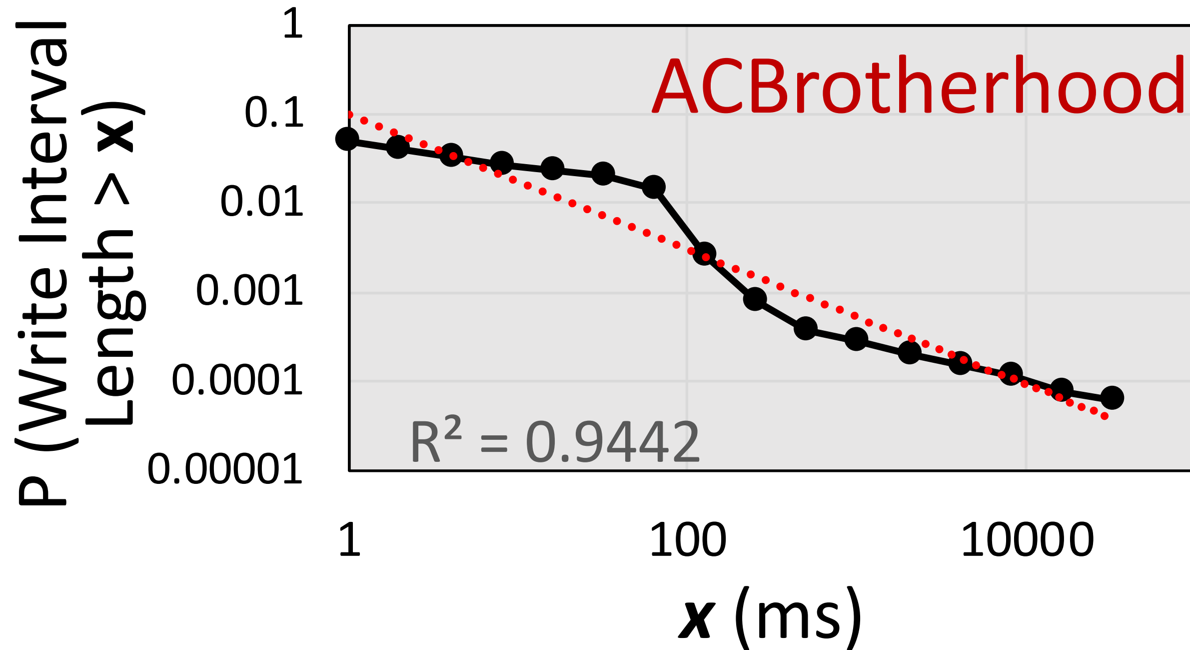
RESULTS

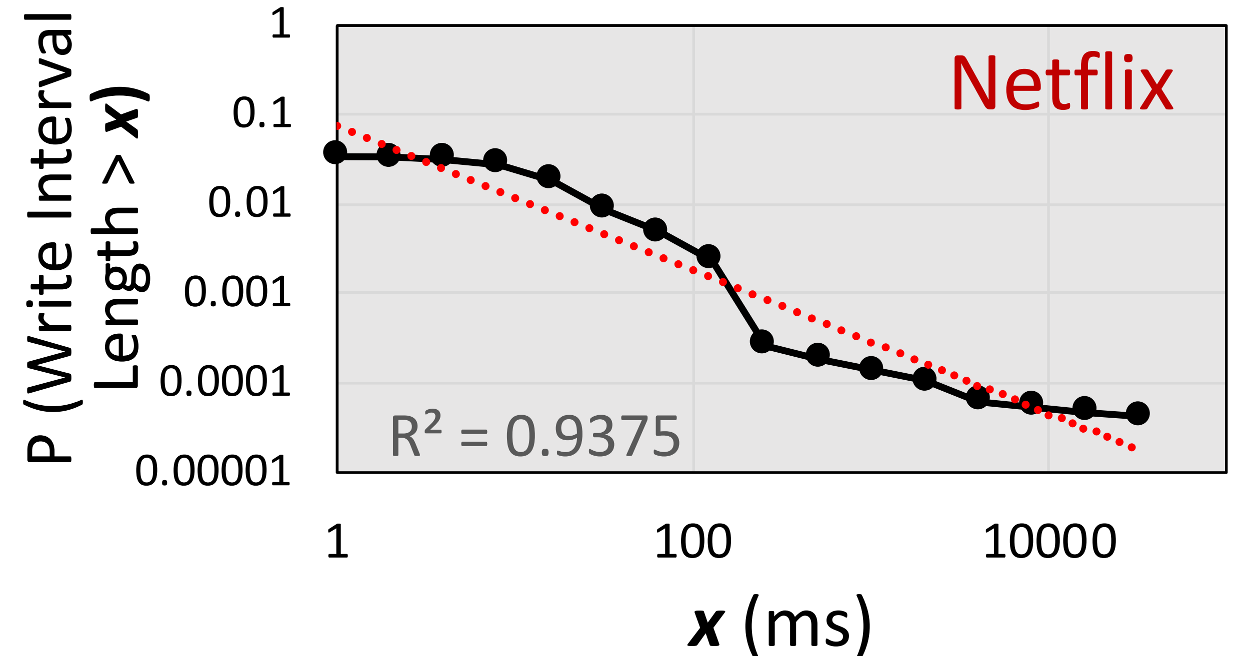
SYSMARK 2014

32 seconds of execution on a real machine

Profiled with a custom FPGA-based infrastructure

WRITE INTERVAL CHARACTERISTICS

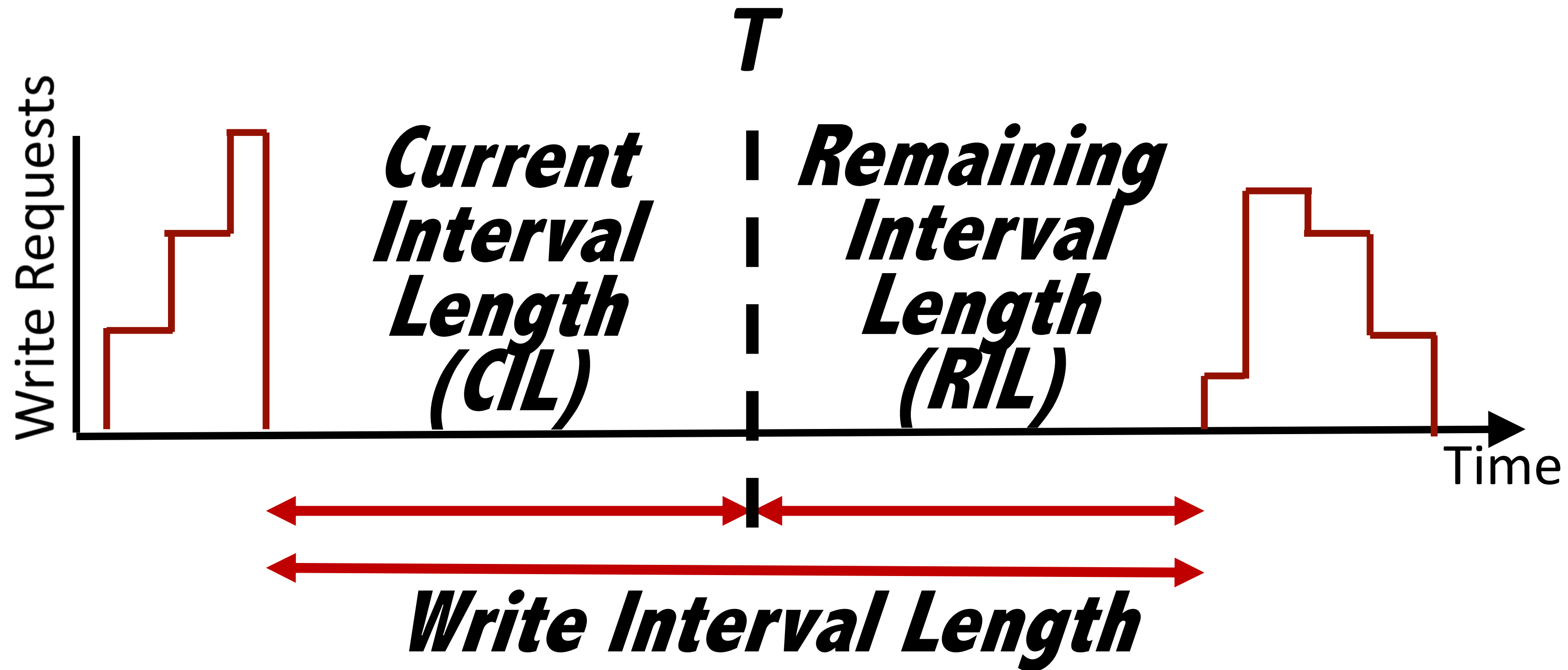




Write intervals follow a Pareto distribution

The longer the elapsed time after a write
→ The longer the write interval

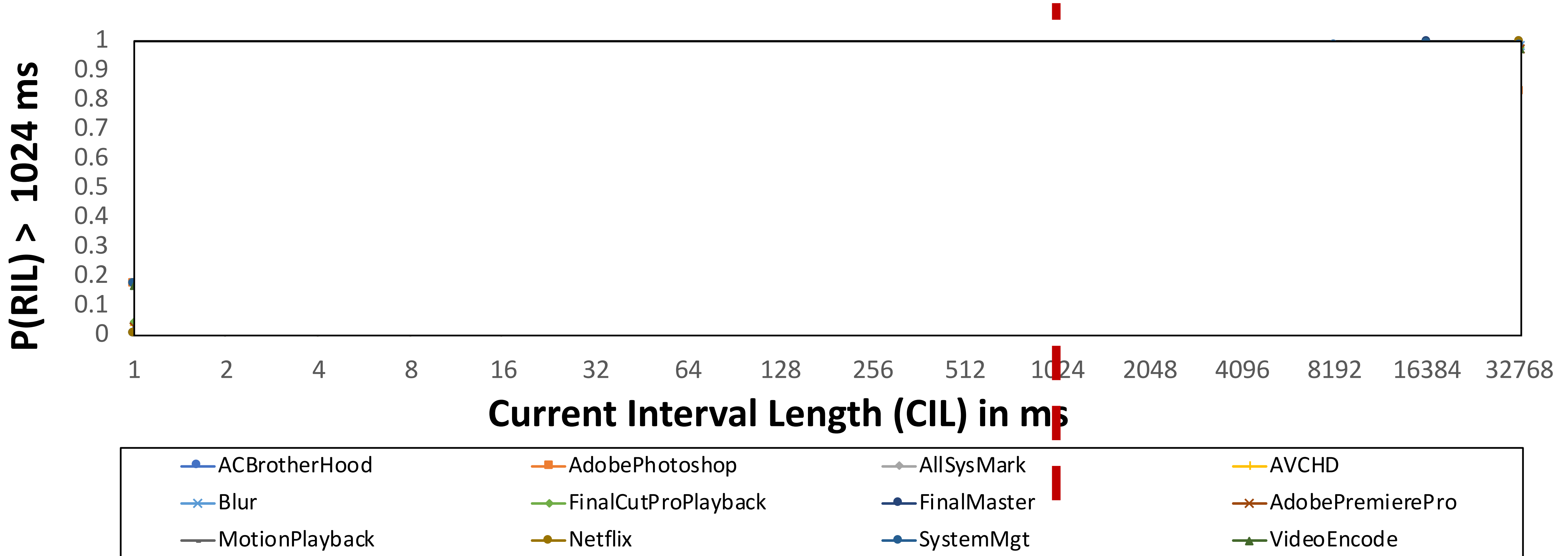
WRITE INTERVAL CHARACTERISTICS



The longer the ***CIL***

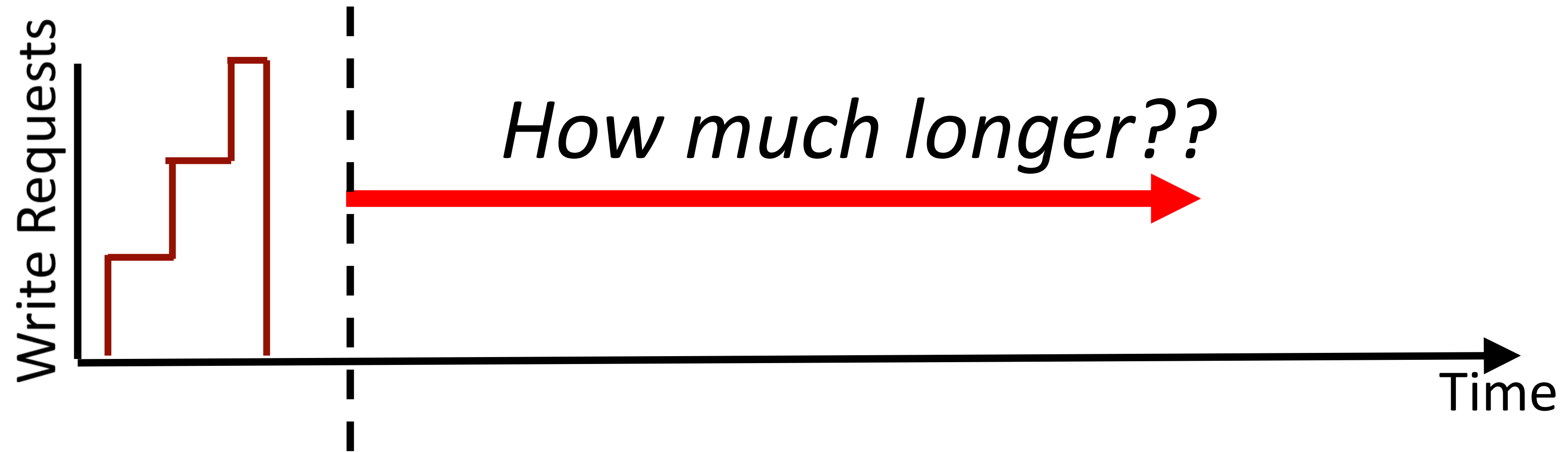
→ It is expected that the longer the ***RIL***

WRITE INTERVAL CHARACTERISTICS

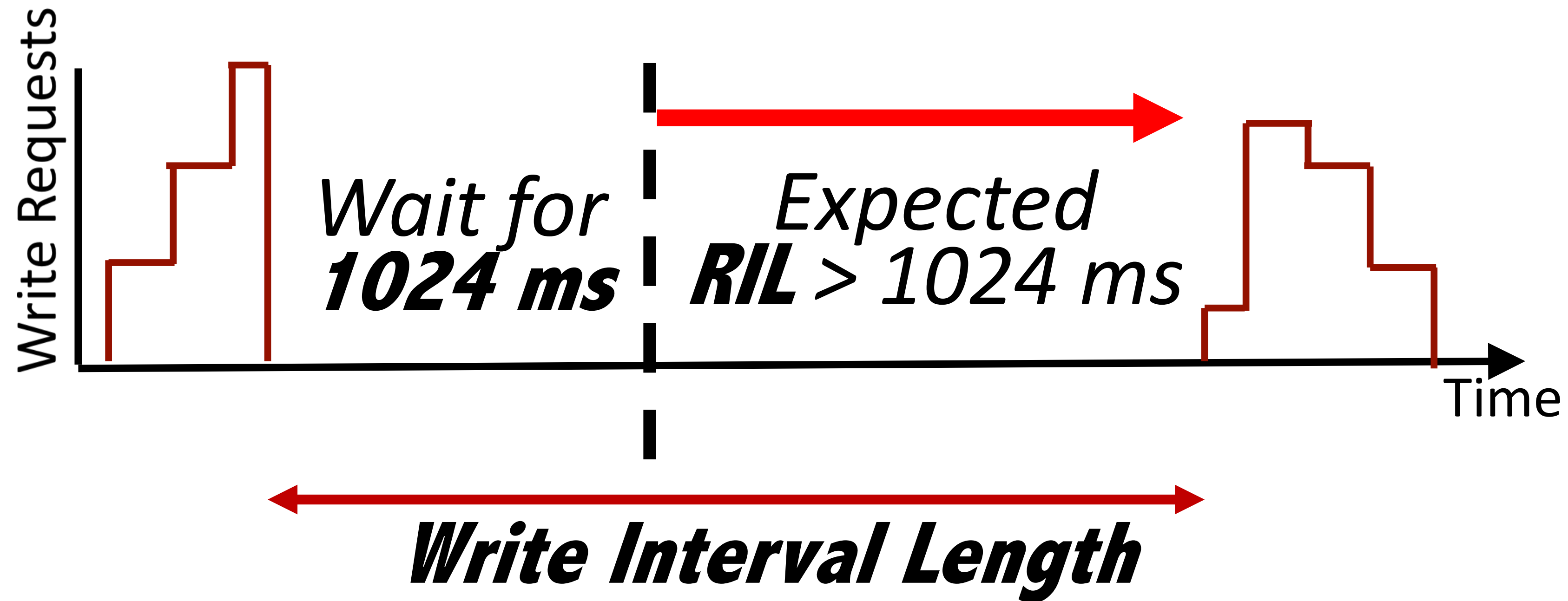


If the interval is already 1024 ms long,
the probability that the remaining interval is greater than 1024 ms is on average 76%

How do we predict the interval on a write access?

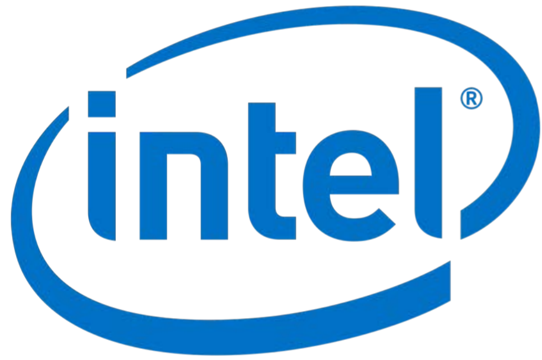


WRITE INTERVAL PREDICTION



After a write, wait for a CIL, where P(RIL) > 1024 is high

If idle, predict the interval will last more than 1024 ms



Microsoft®

Stony Brook
University

ETH Zürich

PennState