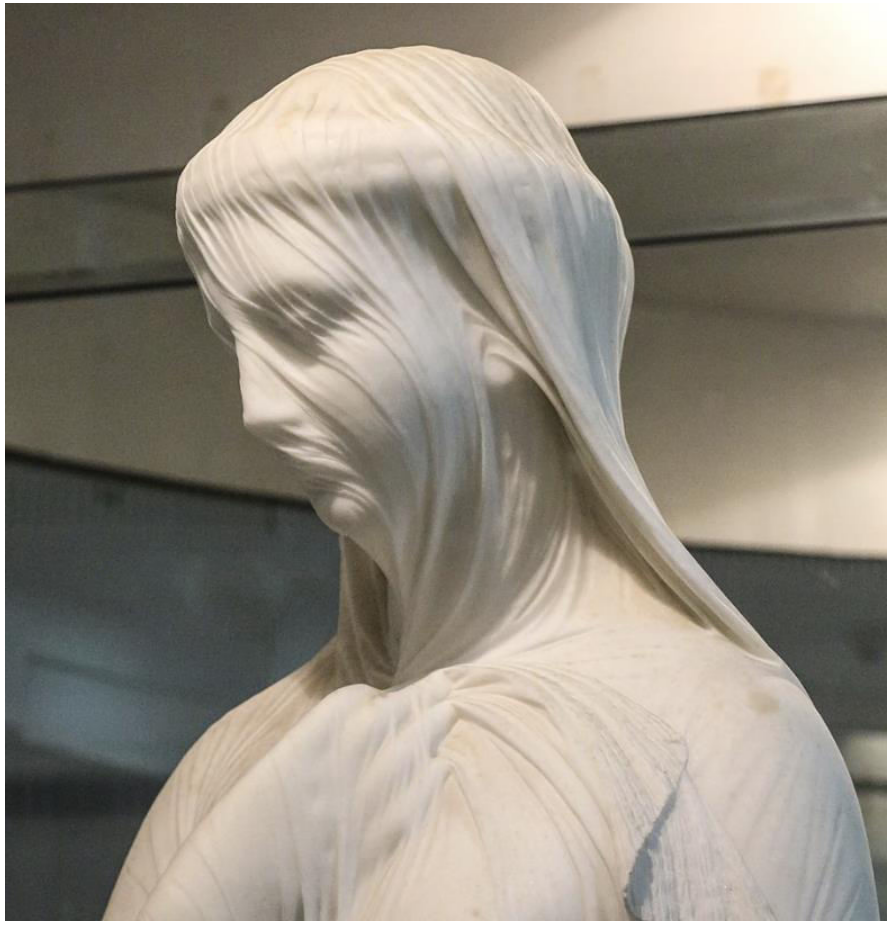
Moving the Needle in Rowhammer Defense with a Minimalist Approach

Moinuddin Qureshi
(moin@gatech.edu)

Keynote, DRAMSec-2025

Painting vs. Sculpture?



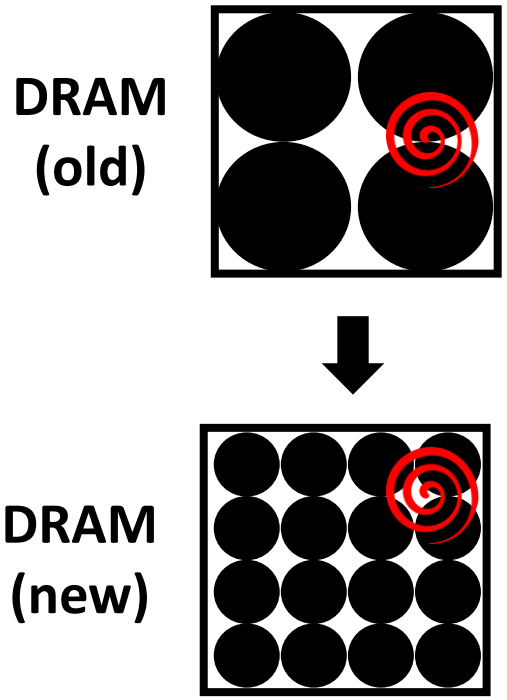
A painting is complete when nothing more needs to be added

A sculpture is complete when nothing more can be removed

The Rowhammer Saga

DRAM Scaling for Increased Capacity

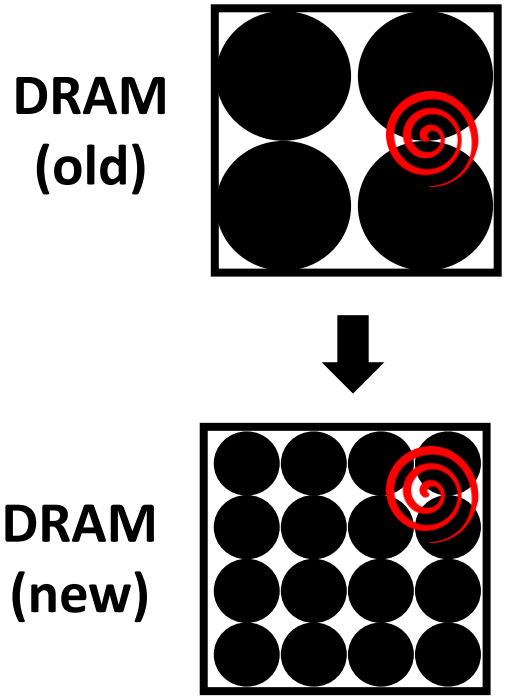
More Inter-Cell Interference

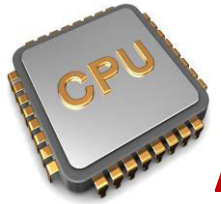


Rowhammer Attacks

DRAM Scaling for Increased Capacity

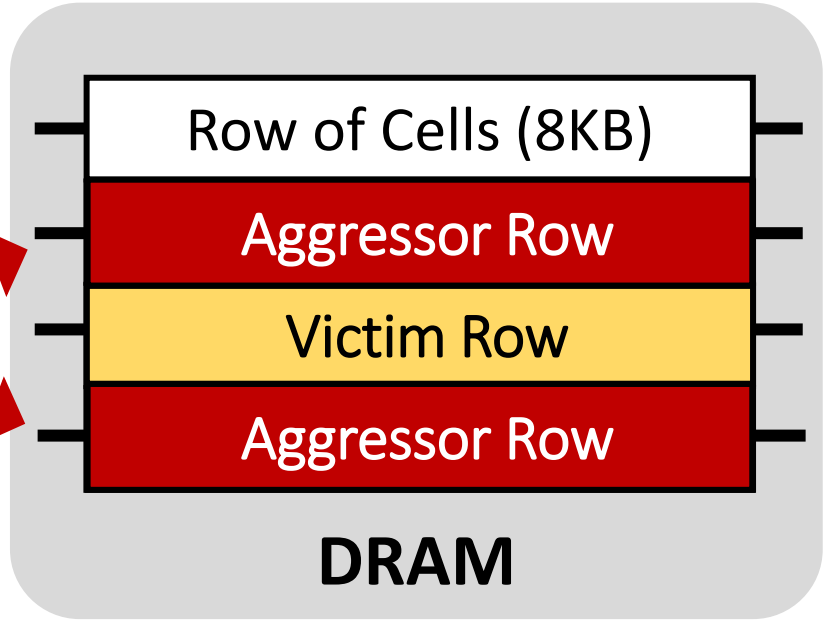
More Inter-Cell Interference





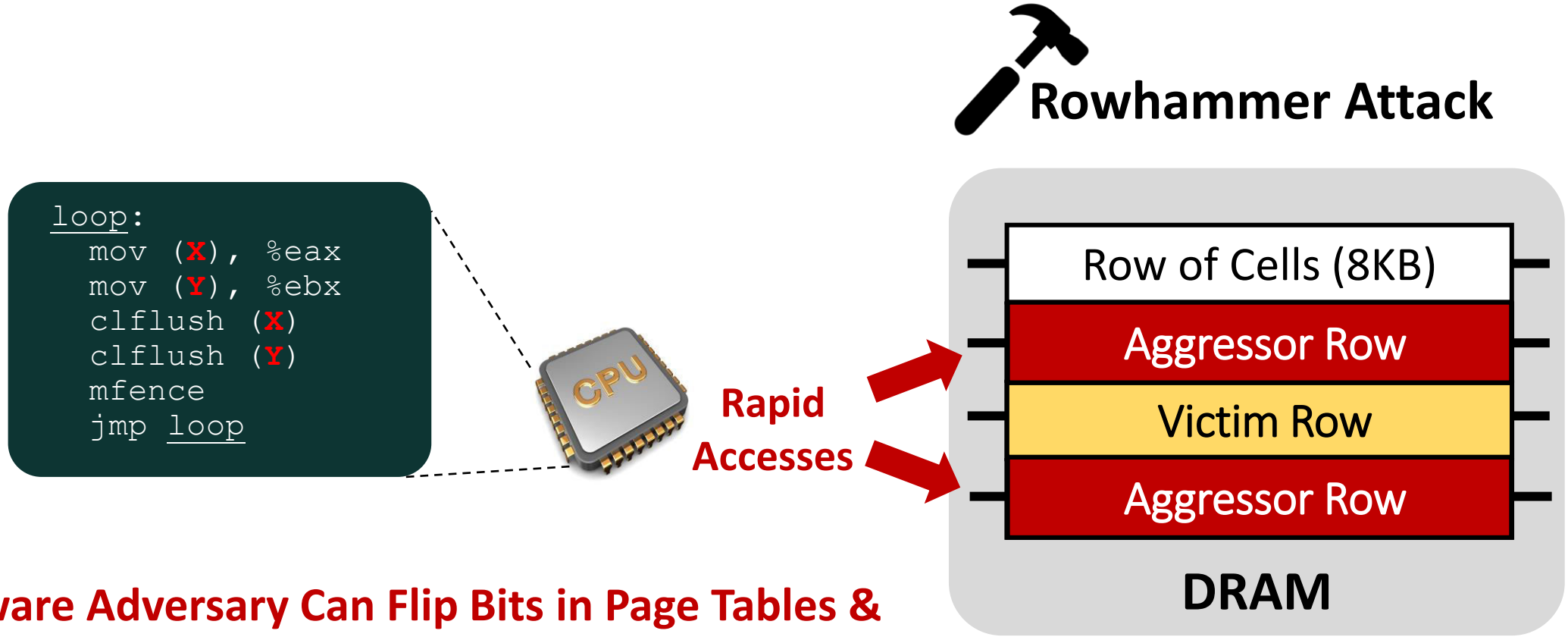
Rapid Accesses

Rowhammer Attack



Bit-Flips in Neighboring Rows

Rowhammer Attacks

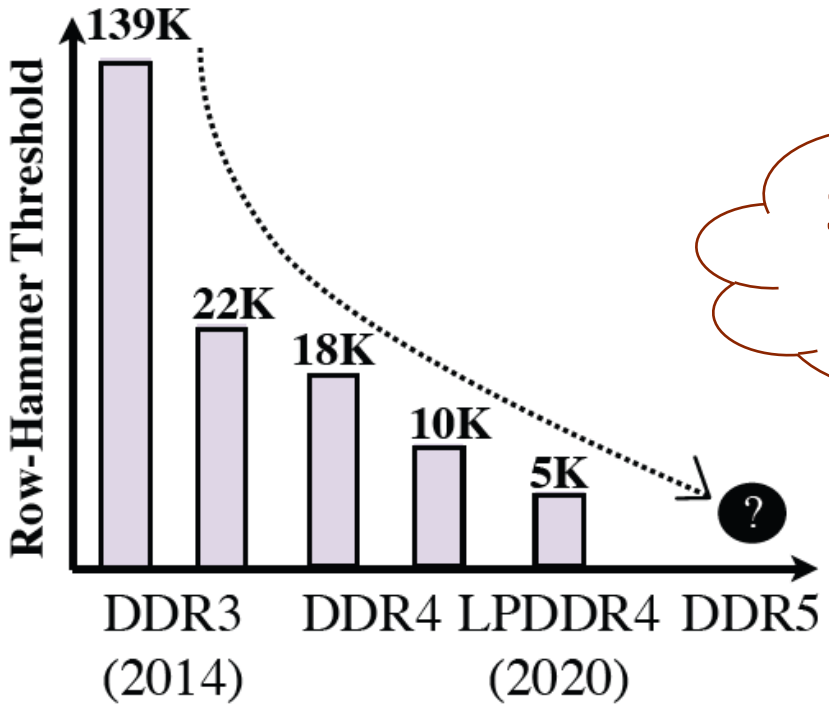


Software Adversary Can Flip Bits in Page Tables & Gain Kernel Privileges (Take Over a System)

[Seaborn+, Blackhat'15]

Rowhammer is Getting Worse!

Rowhammer Threshold: TRH-S (Single Sided) and TRH-D (Double-Sided)

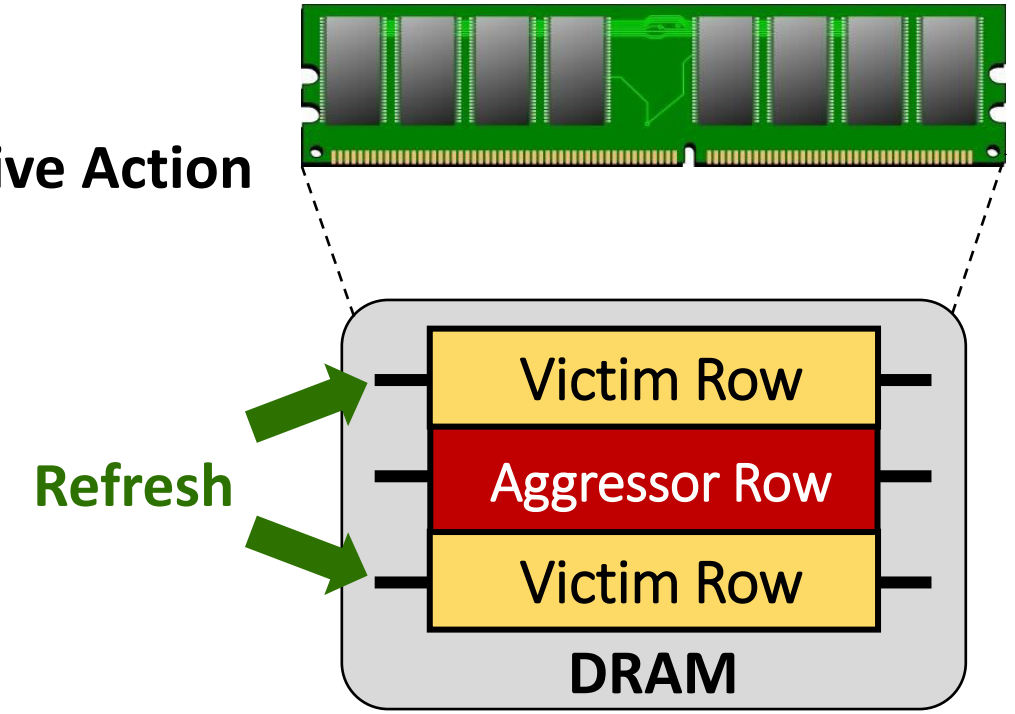


30x Reduction in TRH in 8 years!

Solutions must tolerate not only current TRH but future TRH

Typical Mitigation for Rowhammer

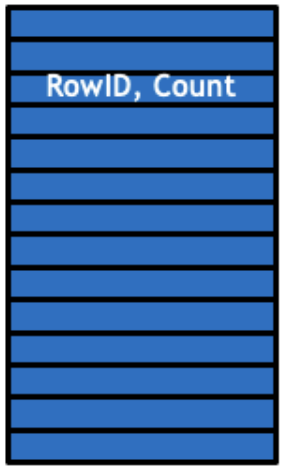
❶ Track Aggressor Rows ❷ Mitigative Action



Tracking can be done at Memory Controller (MC) or In-DRAM
(In-DRAM mitigation can solve DRAM problem within DRAM)

What is In-DRAM Mitigation?

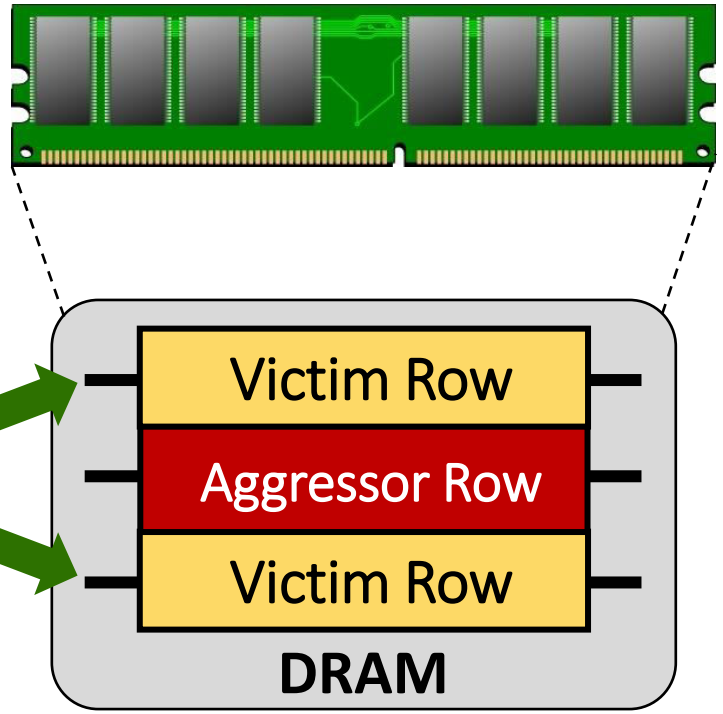
❶ Track Aggressor Rows



DDR4: 4-16 Entries

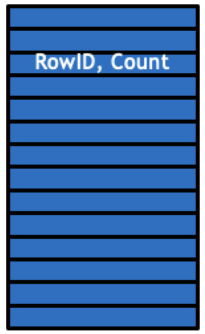
❷ Mitigative Action

Borrow Time from REF

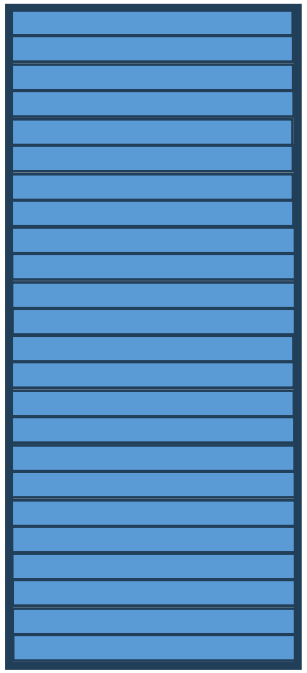


In-DRAM Mitigation needs Space (Tracking) and Time (Victim Refresh)

What About the In-DRAM Tracker?



Optimal Trackers



ProTRR, Mithril
(100-1000s entries)

Low-Cost Tracker

4-20 entries

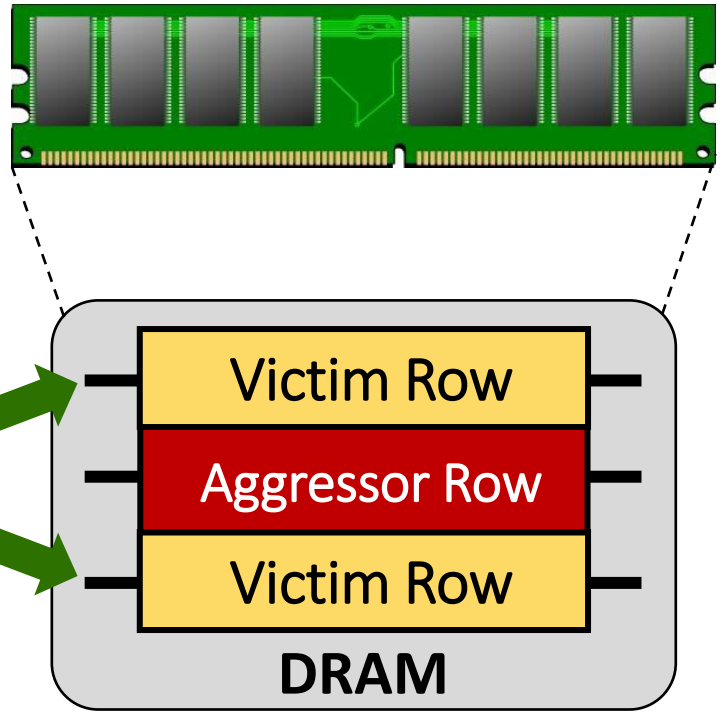
(TRR, DSAC, PAT)

Optimal trackers incur high SRAM overheads. Low-Cost Trackers Secure?

Low-Cost In-DRAM Trackers Broken!

Targeted Row Refresh (TRR) in DDR4 (2015)

❶ ~~Track Aggressor Rows~~ ❷ Mitigative Action



TRResspass Breaks TRR Tracker [Frigo+, SP'20]

> **Poor Rowhammer Fixes On DDR4 DRAM Chips Re-Enable Bit Flipping Attacks**
>
> Source: The Hacker News

Blacksmith Attack: All DDR4 DRAM Vulnerable [Jattke+, SP'22]

> **When the world ends, all that will be left are cockroaches and new Rowhammer attacks: RAM defenses broken again**
>
> Blacksmith is latest hammer horror
>
> Source: The Register

DRAM Industry: Our TRR was broken, so ALL low-cost TRR will be broken

DRAM Industry: Throw the Baby Out with the Bathwater

Right question: How do we design a low-cost TRR that is provably secure?

Outline

- The Rowhammer Saga
- **MINT: Minimalist Randomized Tracker**
- **MOAT: Secure Mitigation with PRAC**
- **MoPAC: Reducing the Slowdown of PRAC**
- **Parting Thoughts**

MICRO-2024

MINT: Securely Mitigating Rowhammer with a Minimalist In-DRAM Tracker

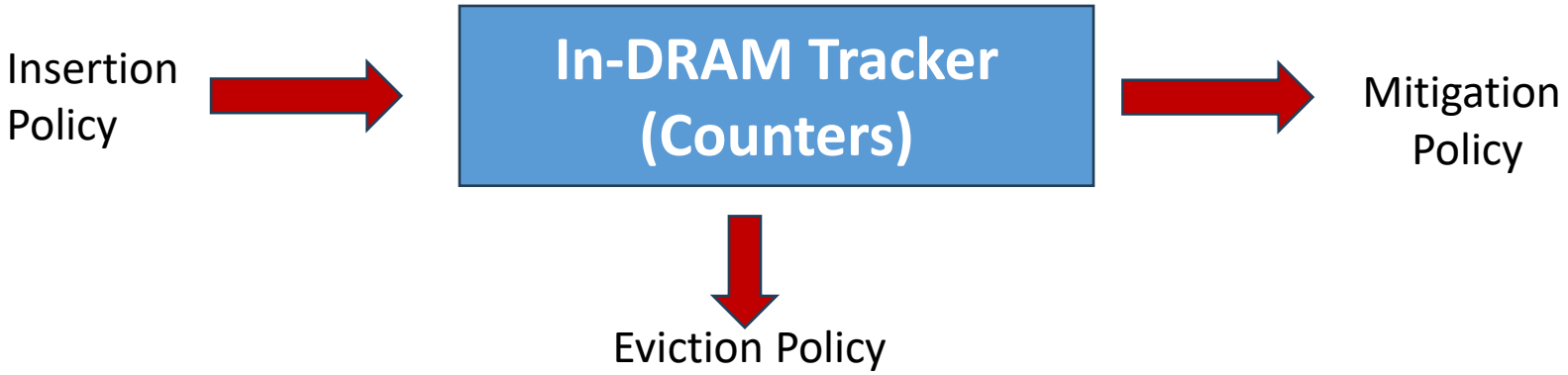
Moinuddin Qureshi
Georgia Tech
moin@gatech.edu

Salman Qazi
Google
sqazi@google.com

Aamer Jaleel
NVIDIA
ajaleel@nvidia.com

Why In-DRAM Trackers Fail?

In-DRAM Trackers characterized as having N entries and managed by 3 policies



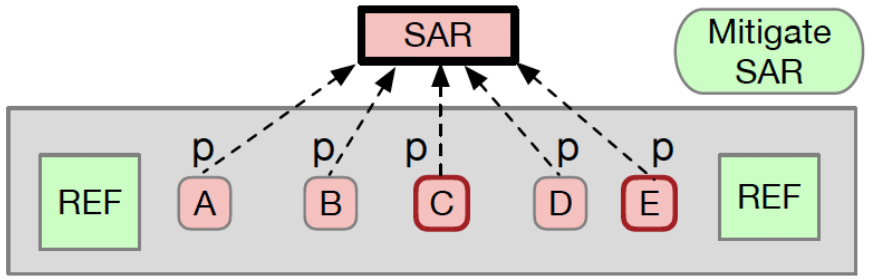
Three sources of failures:

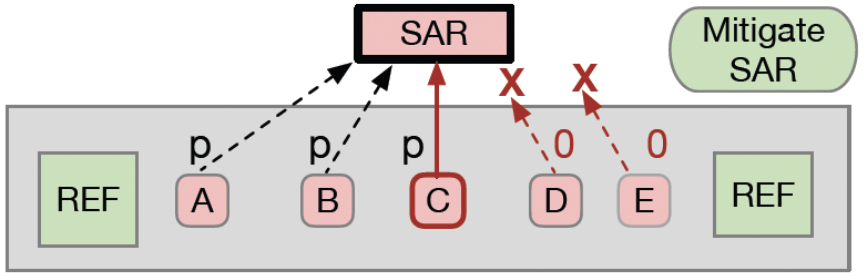
1. **Insertion Failure (target address not get inserted in tracker within TRH)**
2. **Retention Failure (the address was inserted but got evicted without mitigation)**
3. **Tardiness (more activations between insertion and mitigation breaching TRH)**

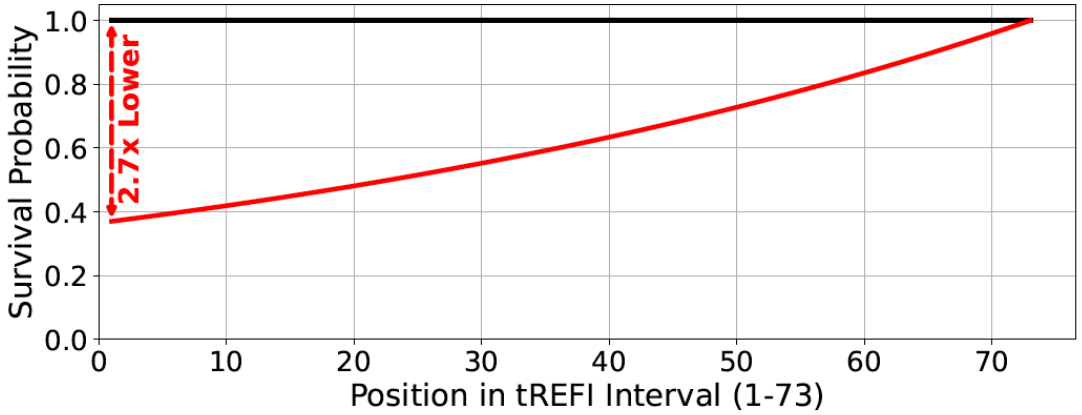
Build trackers such that it is easy to reason about the worst-case patterns and the resulting tolerated threshold

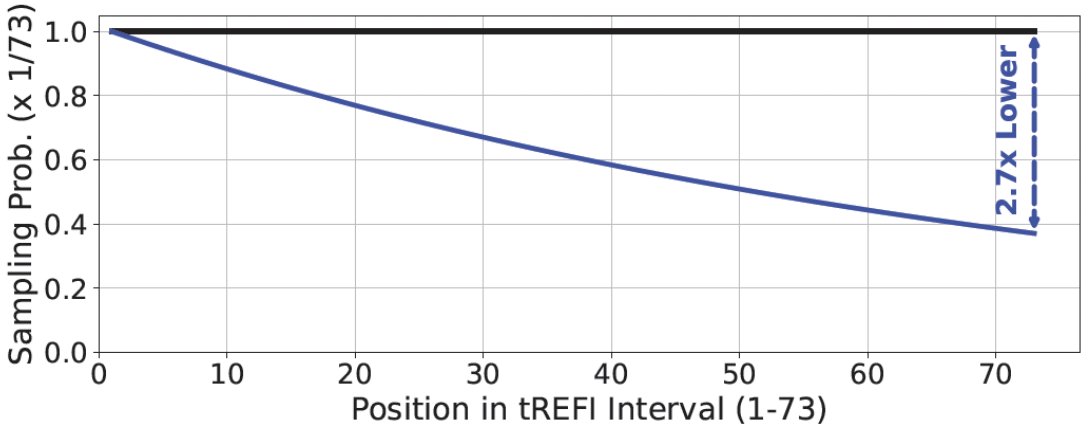
The Problem of Non-Uniform Mitigation

Observation: In-DRAM PARA (IID Tracking) has non-uniform mitigation



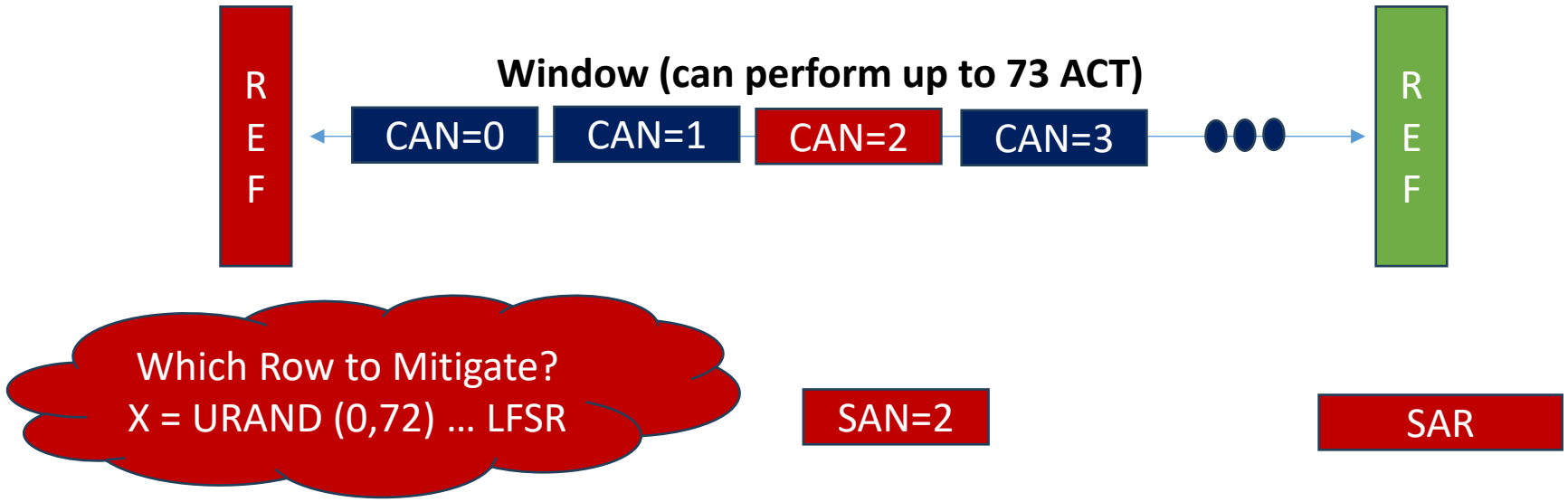






The non-uniform mitigation means attacker can focus on most vulnerable position

MINT: Minimalist In-DRAM Tracker

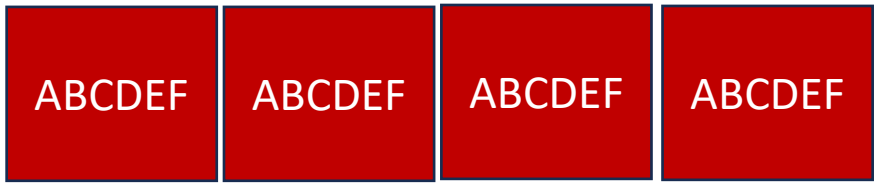


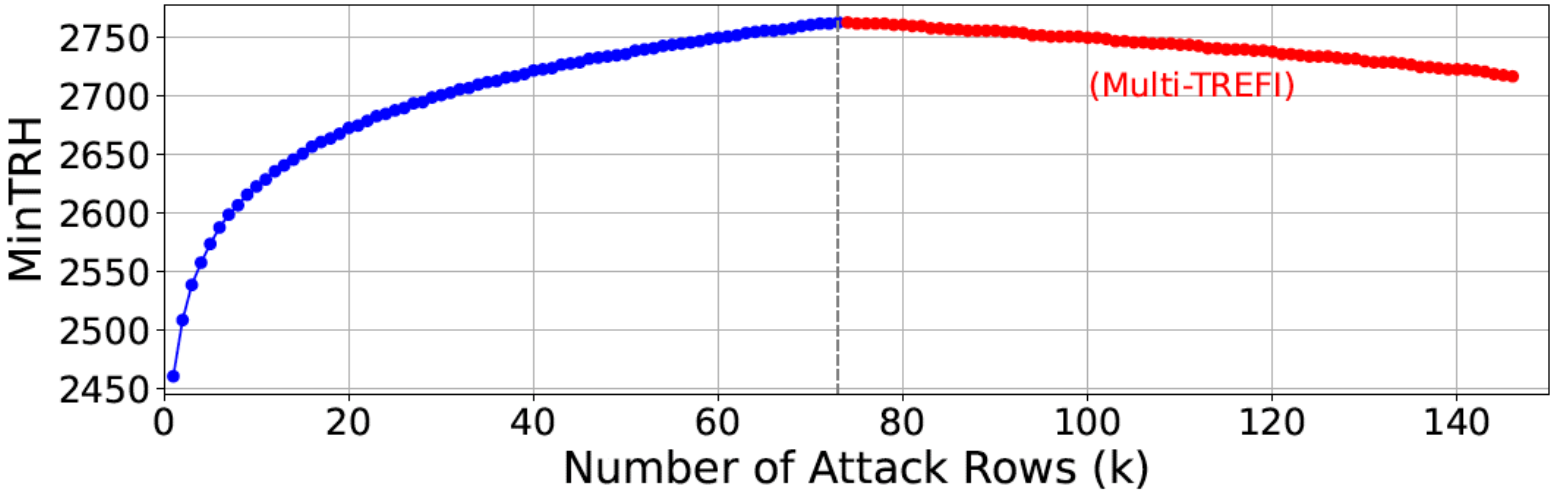
SAN: Selected Activation Num
CAN: Current Activation Num
SAR: Sampled Address Reg

MINT requires low hardware overhead (about 4 bytes)

Deriving Worst-Case Pattern

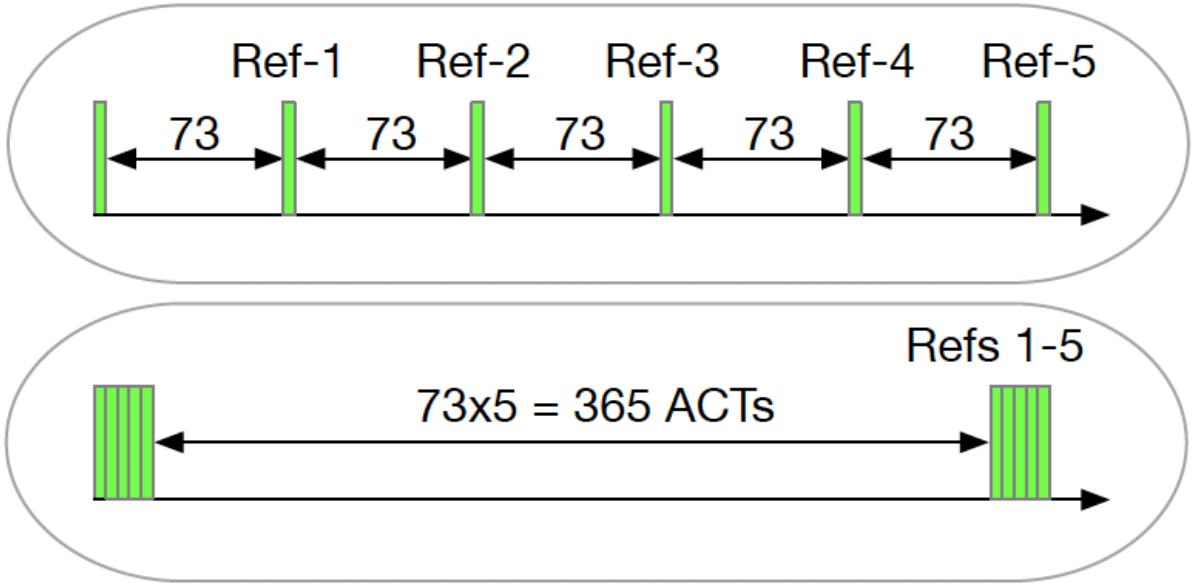
What if we keep have 73 attack lines in the tREFI window? Failure probability becomes 73x





MINT has a TRH of 2800, TRH-D of 1400 (25% lower than PrIDE)
Outperforming 700-entry tracker with a single-entry!

Refresh Postponement Attacks



Severity: Attacker can cause 478K unmitigated ACTs on a row!
(easy to tolerate with counter-based optimal trackers)

Refresh postponement breaks low-cost trackers

Delayed Mitigation Queue (DMQ)

Simple Solution to Refresh Postponement Attacks

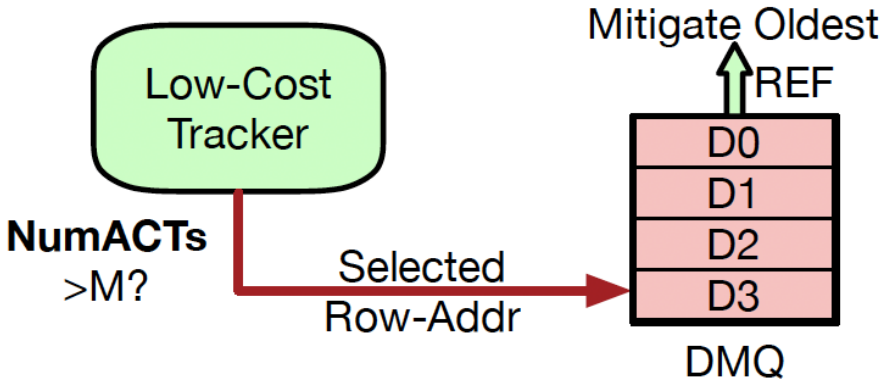


TABLE IV
IMPACT OF REFRESH POSTPONEMENT AND DMQ ON TRACKERS

Design	Entries (Bank)	MinTRH-D (NoPostpone)	MinTRH-D (No DMQ)	MinTRH-D (with DMQ)
PRCT	128K	623	769	769
Mithril	677	1400	1546	1546
PARFM	73	4096	478K	4242
InDRAM-PARA	1	3732	21.3K	3650
MINT	1	1400	478K	1404/1482*

MINT+DMQ: 1 Mitig per tREFI
TRH-D: 1480, Slowdown=0%

MINT+DMQ+RFM16
TRH-D: 356, Slowdown=1.6%

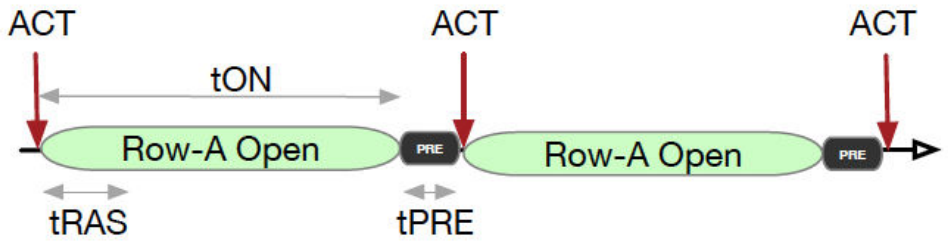
MINT+DMQ can tolerate a TRHD of 1482 (under Adaptive Attacks) (within 2x of an idealized per-row tracker)

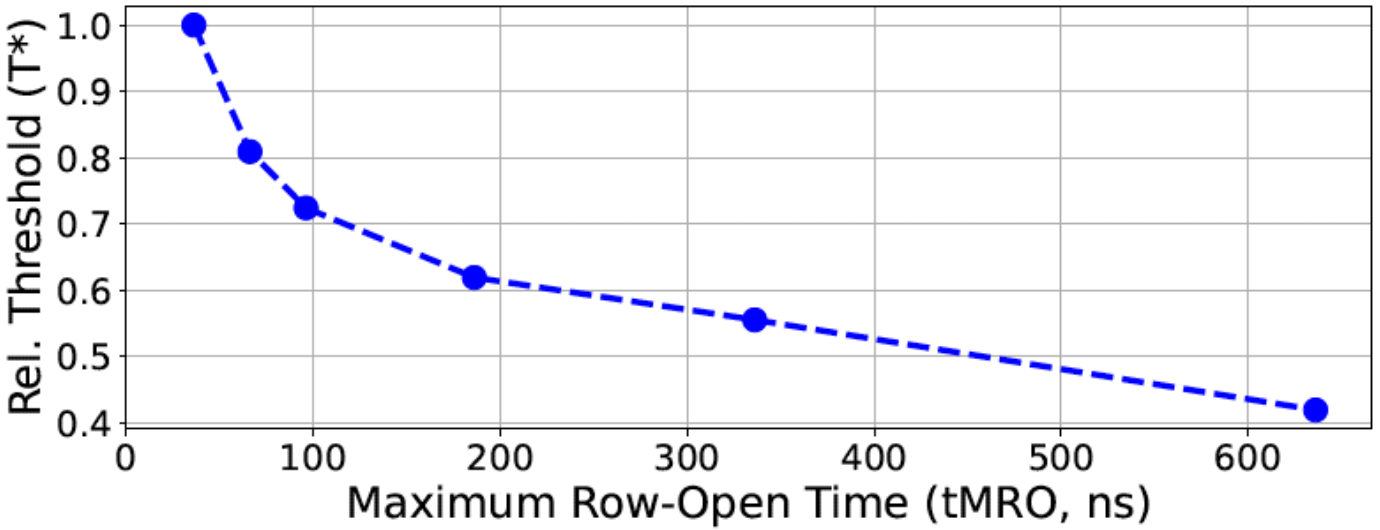
Row-Press: New Data-Disturbance Error

After ACT, neighbor row continue to leak charge on the bit line

Row-Press: Keep the row open for a long time!

[Luo, Mutlu, et al. ISCA-2023]





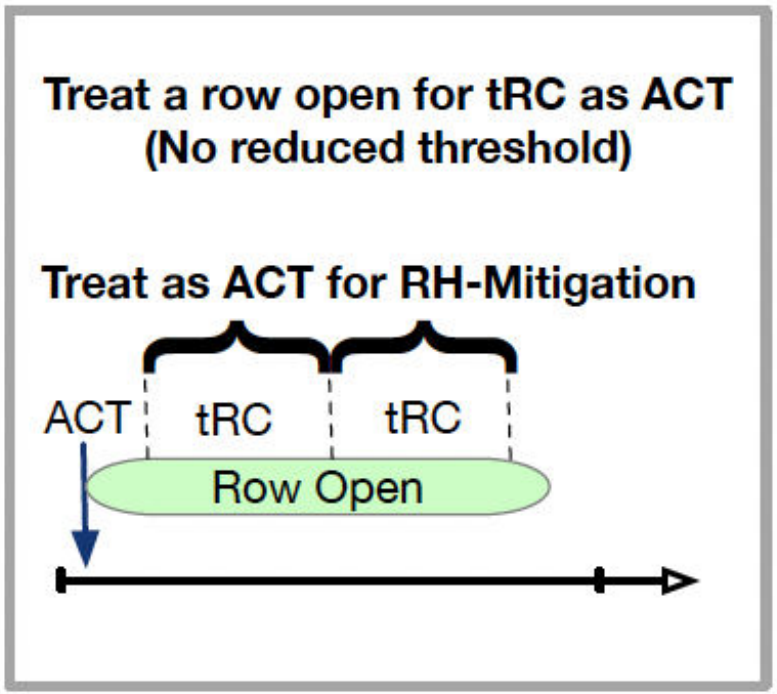
Row-Press cause bitflips with fewer ACTs, breaks all RH trackers

Insight

Tolerate Row-Press without affecting tolerated threshold

No artificial limit on tON (applicable to in-DRAM trackers)

Only minor impact on storage and performance overheads



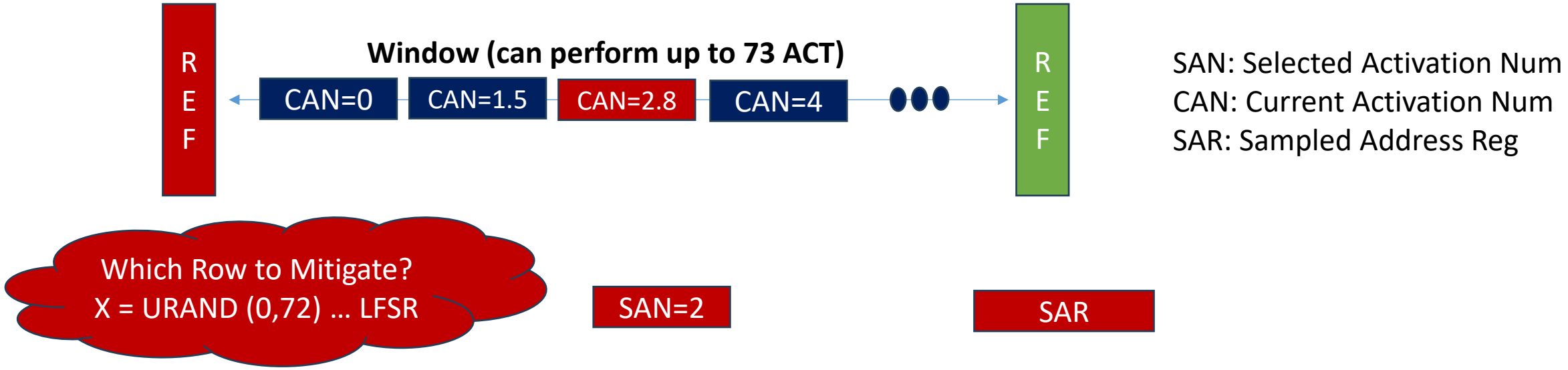
MICRO-2024

ImPress: Securing DRAM Against Data-Disturbance Errors via Implicit Row-Press Mitigation

Anish Saxena Aamer Jaleel Moinuddin Qureshi

Tolerating Row-Press with MINT?

Instead of incrementing CAN by 1 on ACT, increment by EACT
Select Row if CAN crosses SAN



MINT+ImPress defends both RH and RP at ultra-low cost!

Outline

- The Rowhammer Saga
- MINT: Minimalist Randomized Tracker
- **MOAT: Secure Mitigation with PRAC**
- **MoPAC: Reducing the Slowdown of PRAC**
- **Parting Thoughts**

ASPLOS-25

MOAT: Securely Mitigating Rowhammer with Per-Row Activation Counters

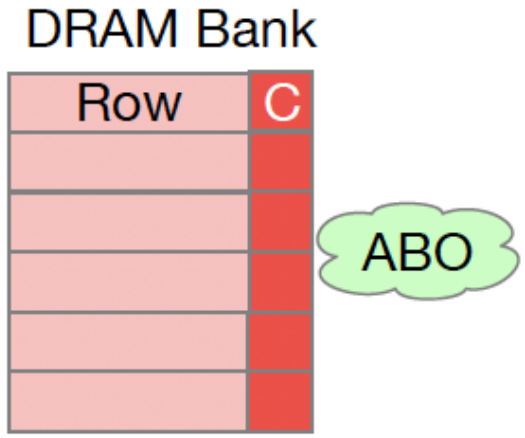
Moinuddin Qureshi
moin@gatech.edu

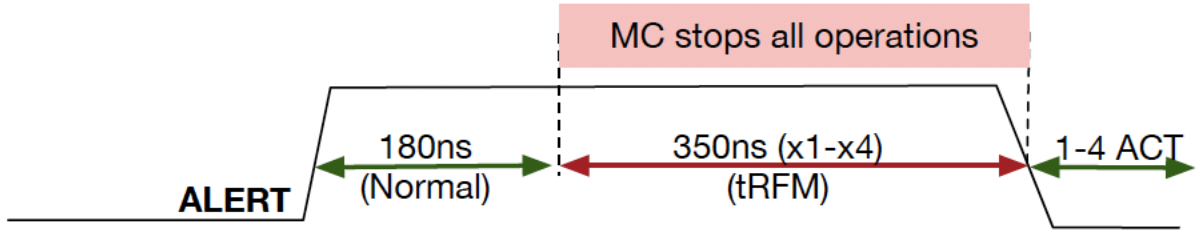
Salman Qazi
sqazi@google.com

JEDEC Introduces PRAC+ABO

JEDEC: low-cost trackers broken, so let's go for per-row tracking (space) and ABO (time)

PRAC: Per-Row-Activation-Counting, solves the space issue
ABO: Alert Back-off, solves the time issue



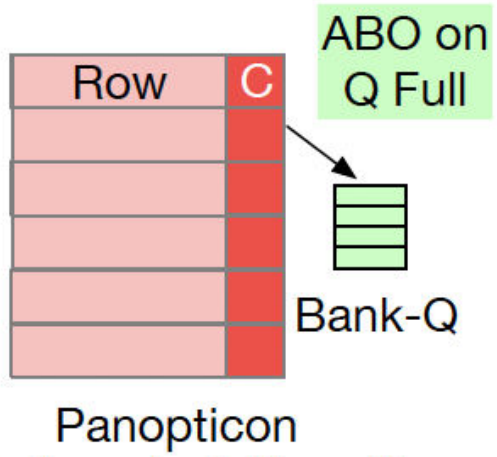


PRAC+ABO is a principled defense against RH (big changes to DRAM)

Panopticon Design

PRAC is a framework, does not specify implementation details, security depends on design

Typically, some SRAM needed to track which row(s) will get mitigated



Insert row in queue
if threshold bit flips

One aggressor-row
mitigated every Nth REF

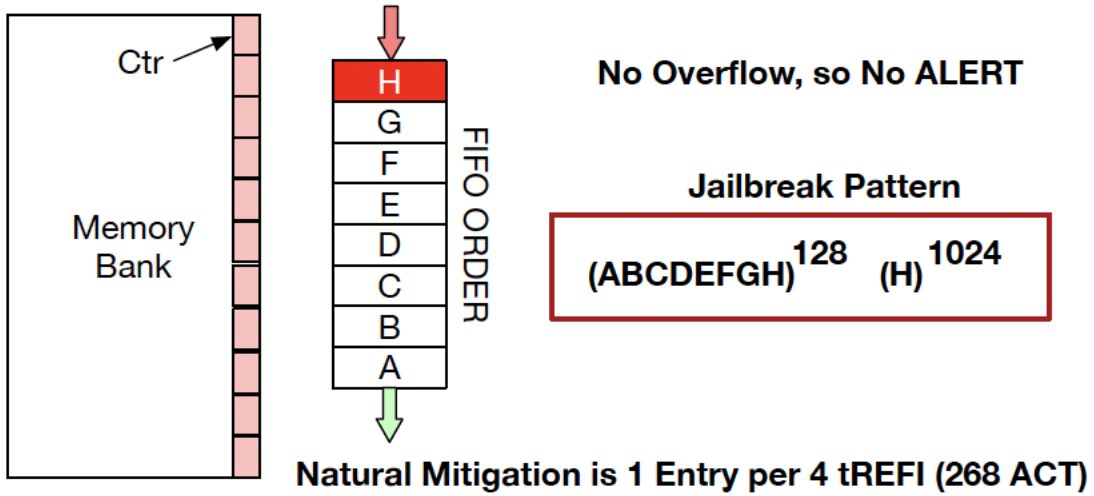
Trigger ALERT
if queue is full

Panopticon: seminal work & inspiration behind PRAC and ABO

Exploiting Tardiness for Attack

E.g. Design does not track activations while the row is buffered in the queue

We can attack this design by filling the queue, and hammering the youngest row

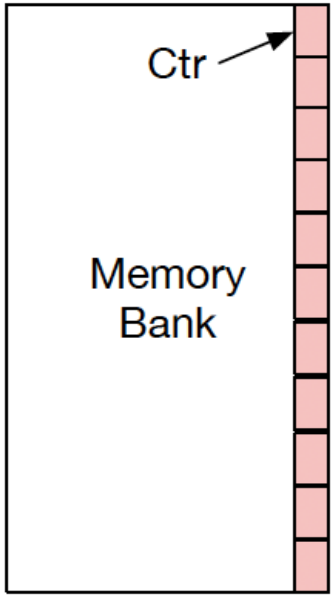


Jailbreak inflicts 1150 ACTs on a row for design with threshold of 128

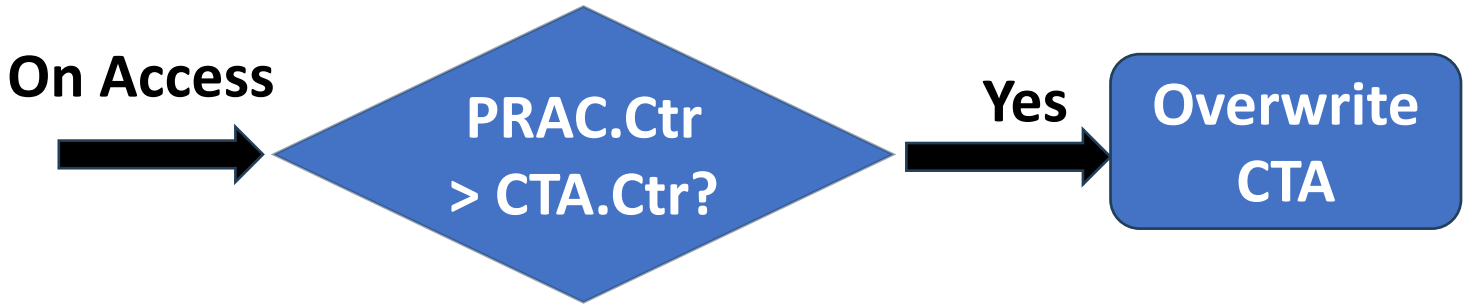
MOAT: Minimalist Provably Secure Design

Insight: Greedily track a single entry per bank (minimalist design, low SRAM cost)

ATH: ALERT Threshold determines TRH



CTA: Currently Tracked Addr



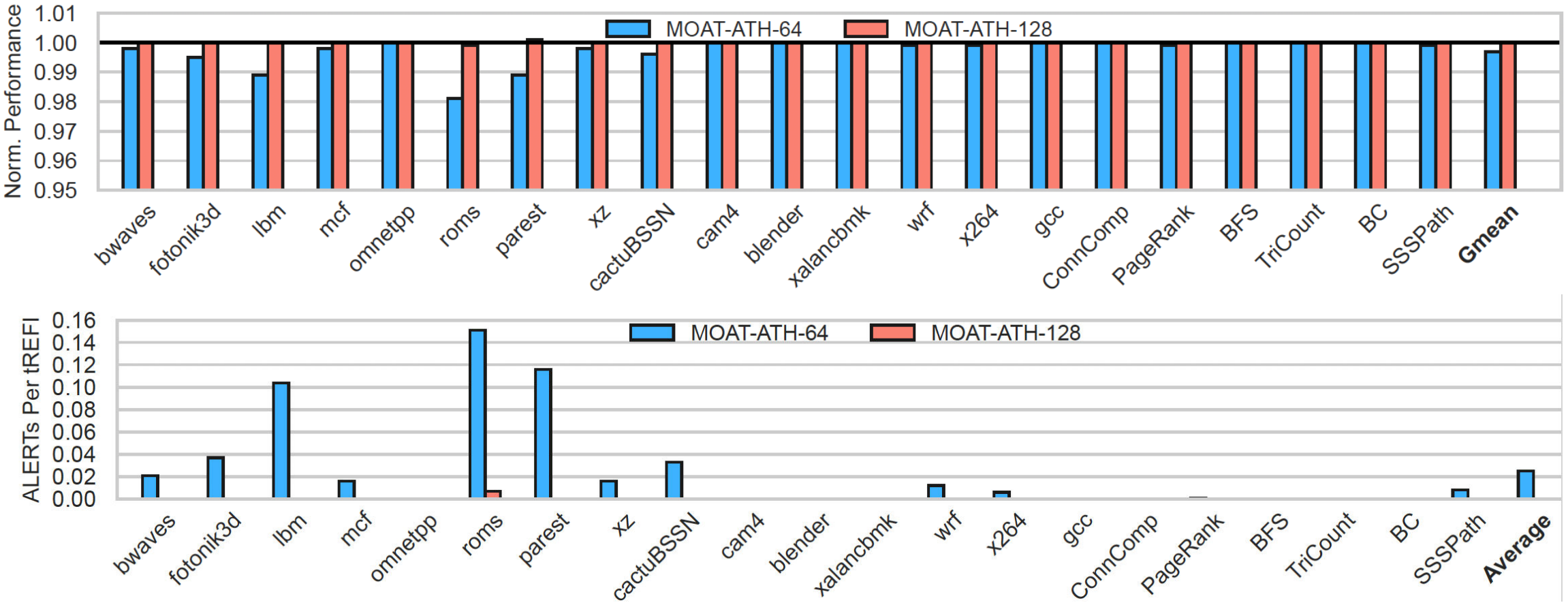
On Access: if CTA.Ctr > ATH, trigger ALERT

On ALERT/Mitigative REF: mitigate CTA

MOAT is the first provably secure implementation for PRAC+ABO

MOAT Overheads (norm. to PRAC)

The mitigation (ALERTs) from MOAT cause negligible performance overheads



MOAT with ATH = 128+ incurs virtually zero ALERTs

Outline

- The Rowhammer Saga
- MINT: Minimalist Randomized Tracker
- MOAT: Secure Mitigation with PRAC
- **MoPAC: Reducing the Slowdown of PRAC**
- **Parting Thoughts**

ISCA-2025

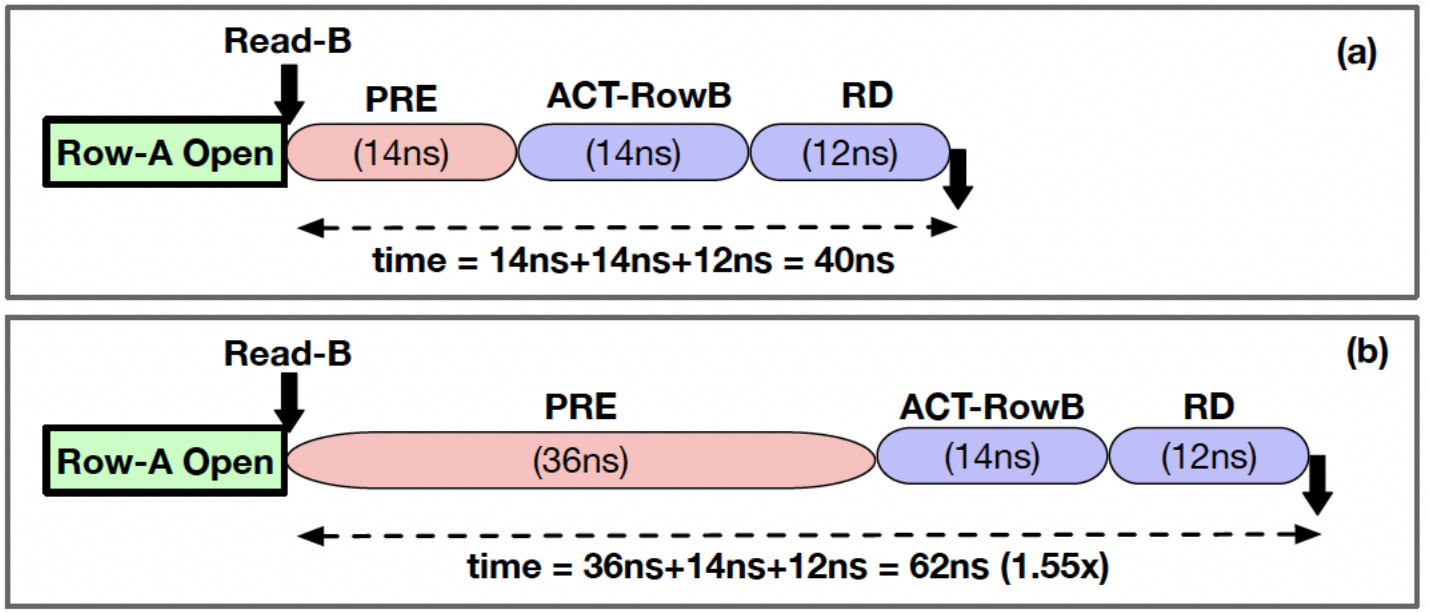
MoPAC: Efficiently Mitigating Rowhammer with Probabilistic Activation Counting

Suhas Vittal (Georgia Tech), Salman Qazi (Google), Poulami Das (UT Austin), Moinuddin Qureshi (Georgia Tech)

The Problem with PRAC

Two sources of slowdown: Intrinsic (inflated timing) + extrinsic (ABO related stalls)

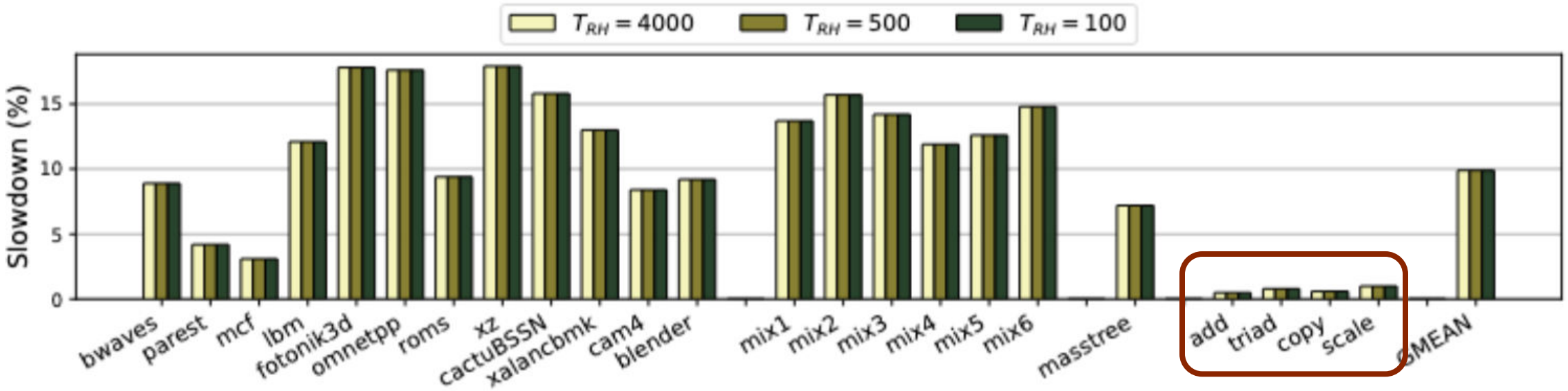
PRAC suffers from high slowdown because of increased tRP/tRC (average 10% slowdown)



For row conflict, service time increases from 40ns to 62ns (1.5x)
For closed-page systems, tRC increase still significant (13%)

The Problem with PRAC

The high tRP and tRC causes significant slowdown (consistent with other studies)



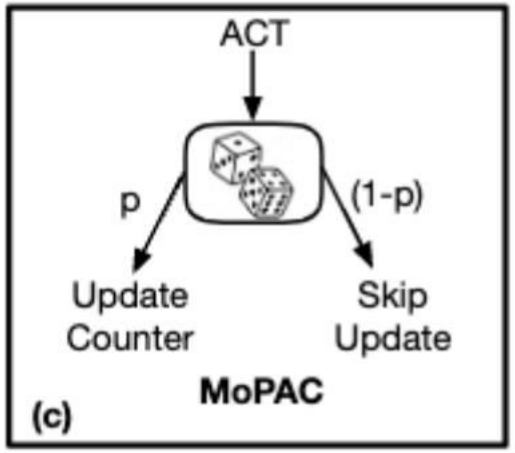
Note: some prior studies showed 1% bus BW impact (BW != IPC)

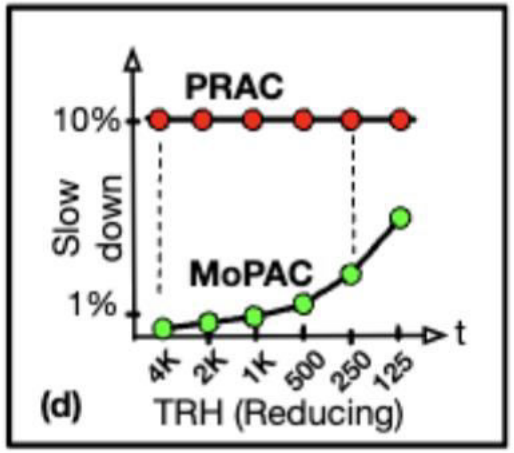
If PRAC related slowdown is high, will the industry adopt PRAC?

Insight: Are all counter updates necessary?

Instead of incrementing PRAC counter on each activation, update probabilistically

Revise ATH to account for sampling rate and sampling inefficiency





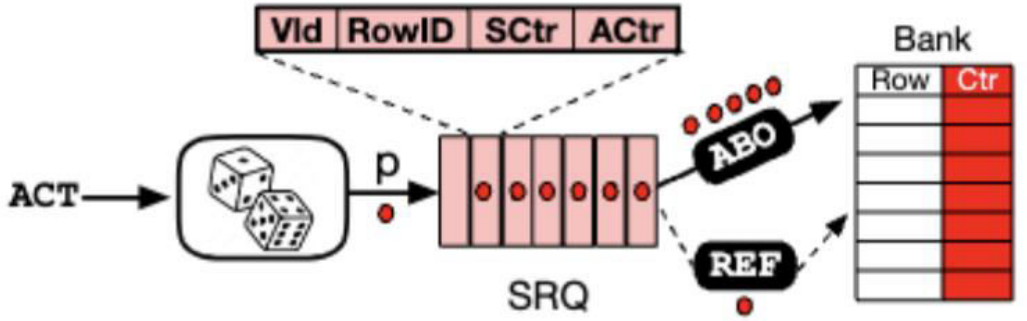


Figure 10: Overview of MoPAC-D. MoPAC-D selects activations with prob "p" and buffers them in the SRQ. The update to PRAC counters occur by triggering an ABO (or under REF).

MOPAC uses REF for ctr updates (triggers ABO if buffer fills up)

Performance Overhead with MoPAC

Unlike PRAC, MoPAC performance overhead depends on threshold

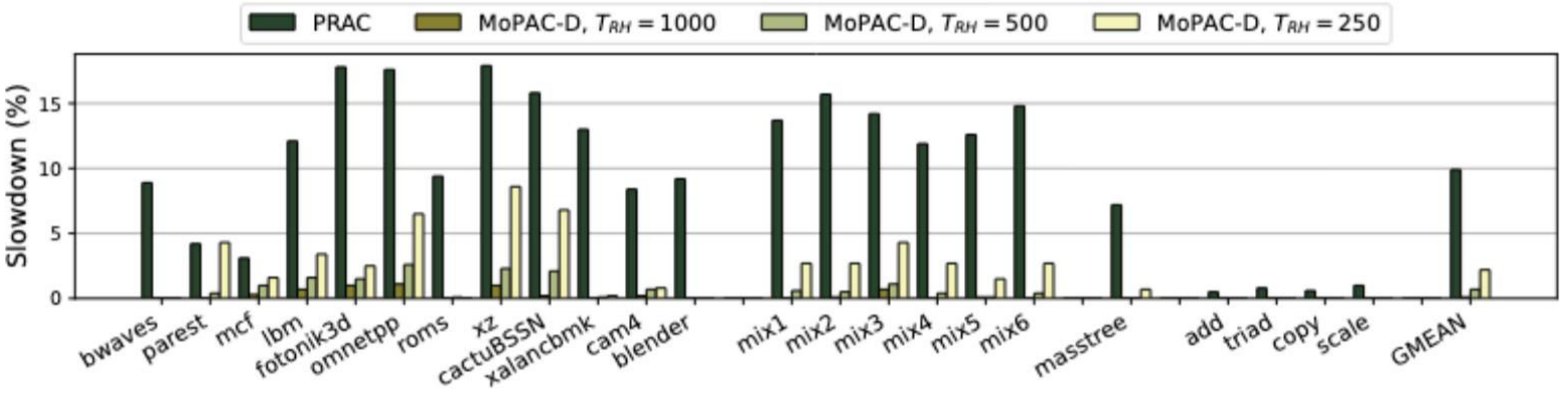


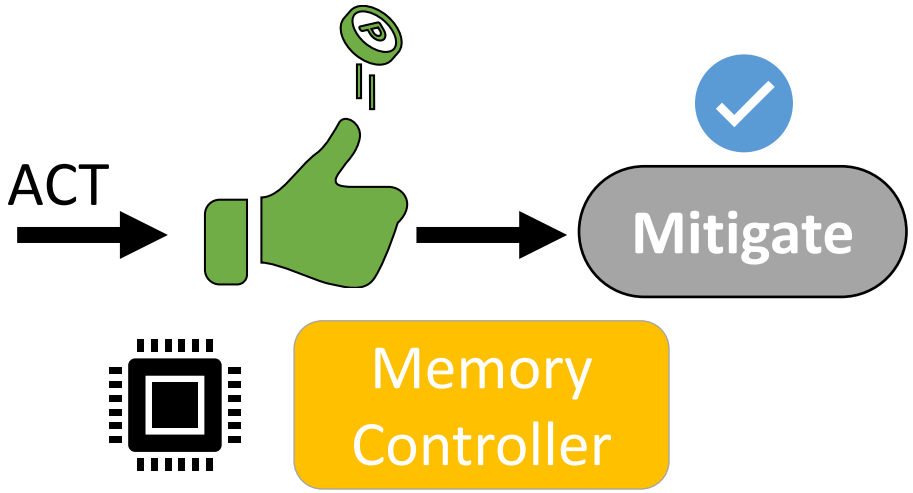
Figure 11: Performance of PRAC and MoPAC-D for different T_{RH}. MoPAC-D results in an average slowdown of 0.1%, 0.7%, 2.2% at T_{RH} of 1000, 500, and 250, respectively, much lower than the 10% with PRAC.

MoPAC reduces the PRAC slowdown from 10% to 0.1%-2.2%

Outline

- The Rowhammer Saga
- MINT: Minimalist Randomized Tracker
- MOAT: Secure Mitigation with PRAC
- MoPAC: Reducing the Slowdown of PRAC
- **DREAM: MC-Side Mitigations with DRFM**
- **Parting Thoughts**

MC-Side Rowhammer Mitigation



TRHD	p
500	1/25
1000	1/50
2000	1/100

Why?

Avoid obscure/proprietary DRAM TRR

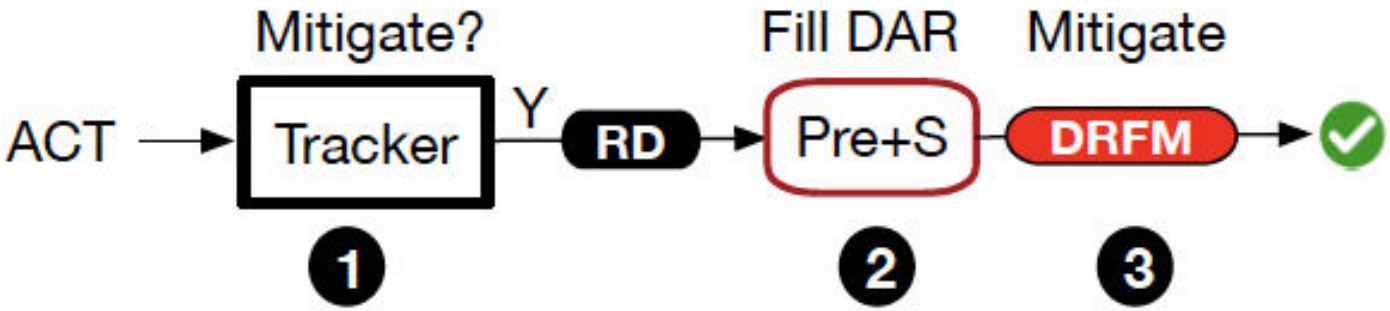
Avoid slowdown of PRAC

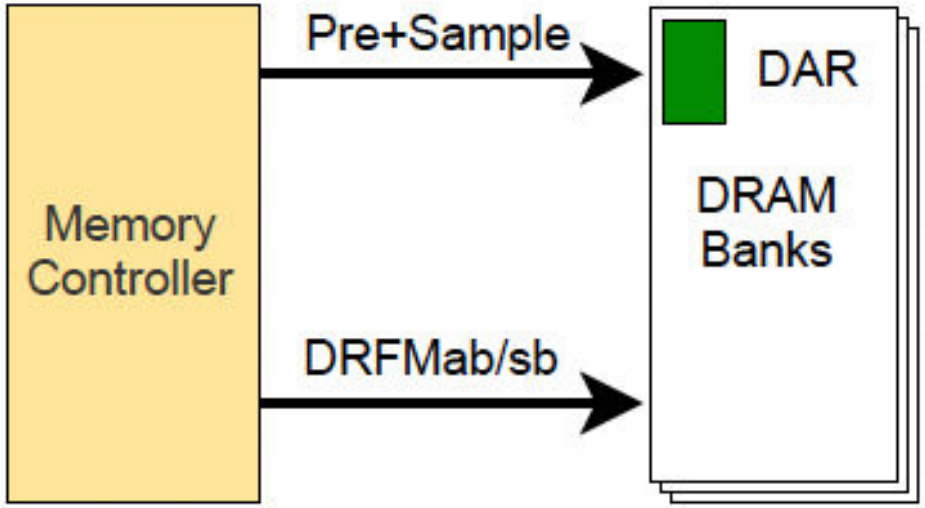
Avoid reliance on REF cannibalization

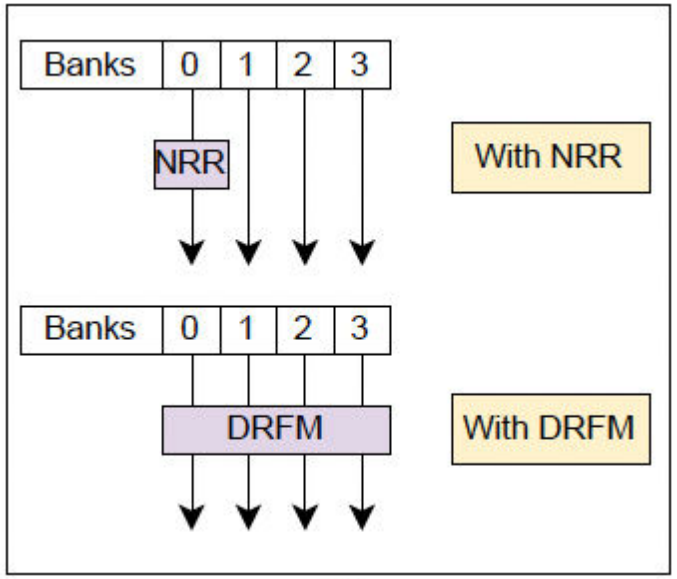
Ensure security at low-cost in SOC

MC-based RH mitigation enables SOC vendors to ensure security

DRFM: DDR5 Support for Mitigation

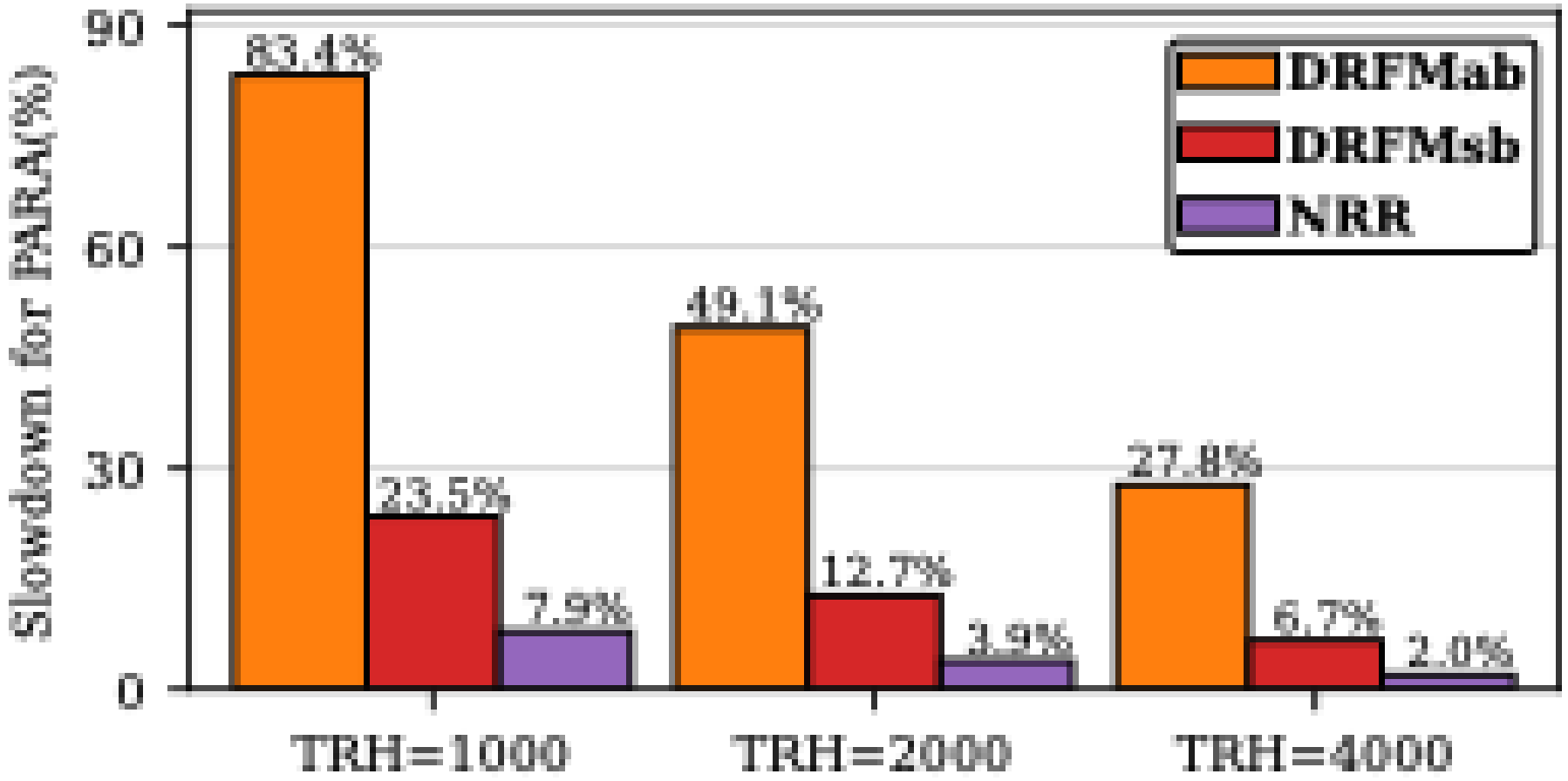






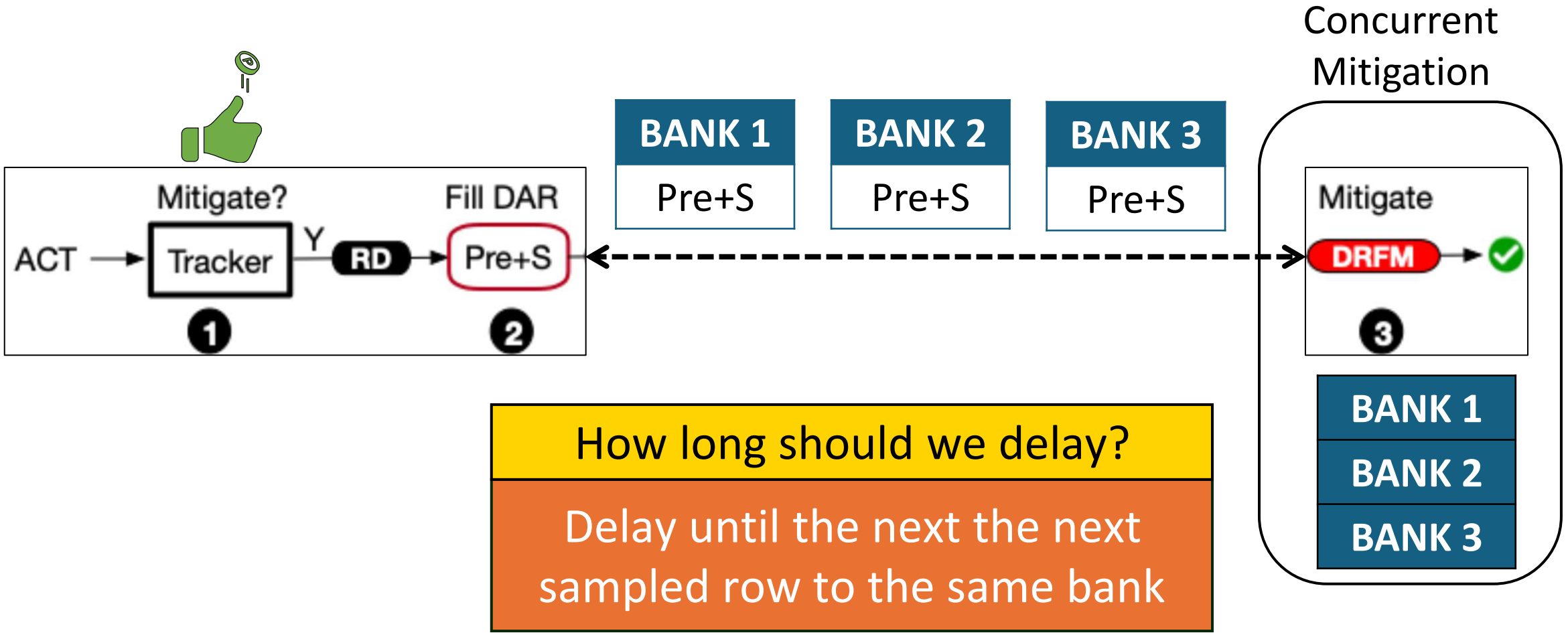
DRFM stalls 8-32 banks for each mitigation, performance impact?

Slowdown from PARA with DRFM



Replacing NRR with DRFM causes significant slowdowns!

Insight: Delay Issuing DRFM

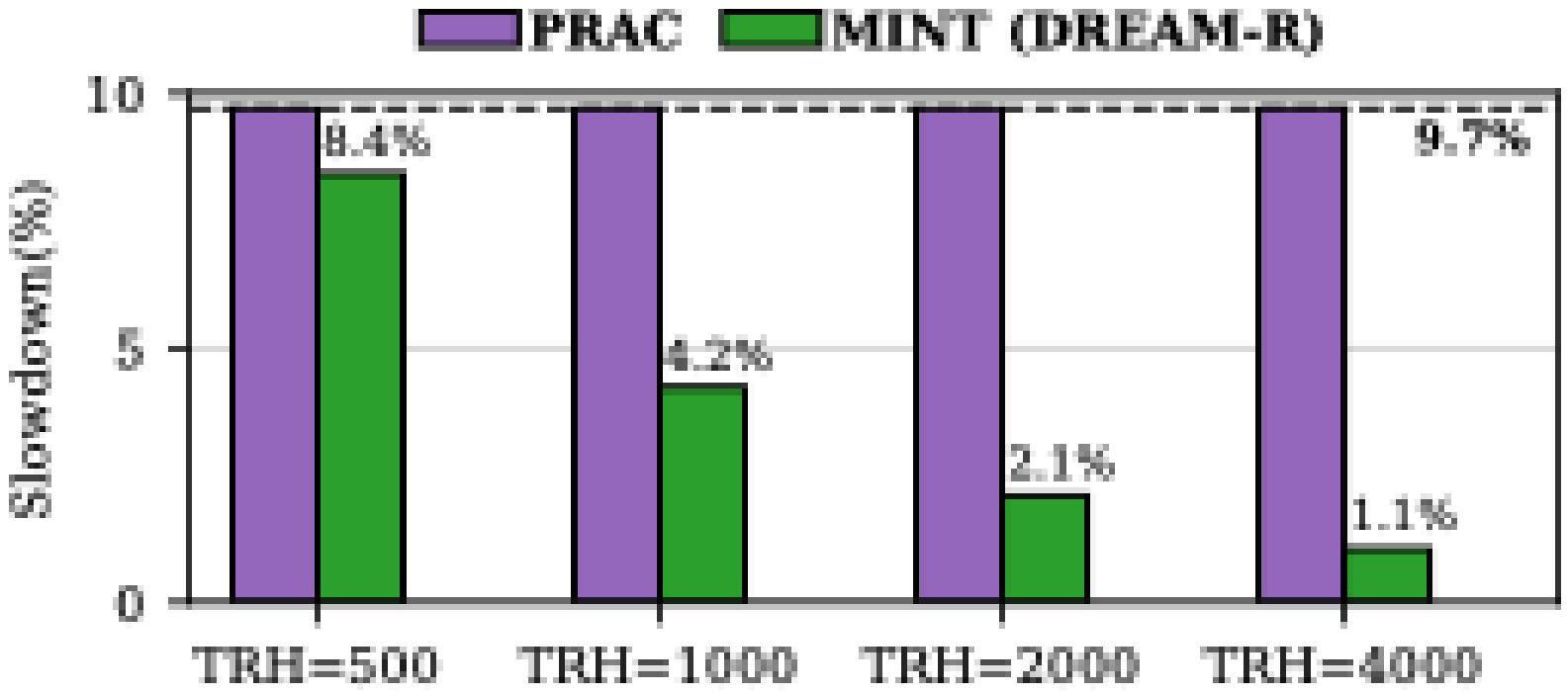


How long should we delay?

Delay until the next the next sampled row to the same bank

Delaying gives other banks chance to sample into DAR

DRFM Aware Mitigation (DREAM)



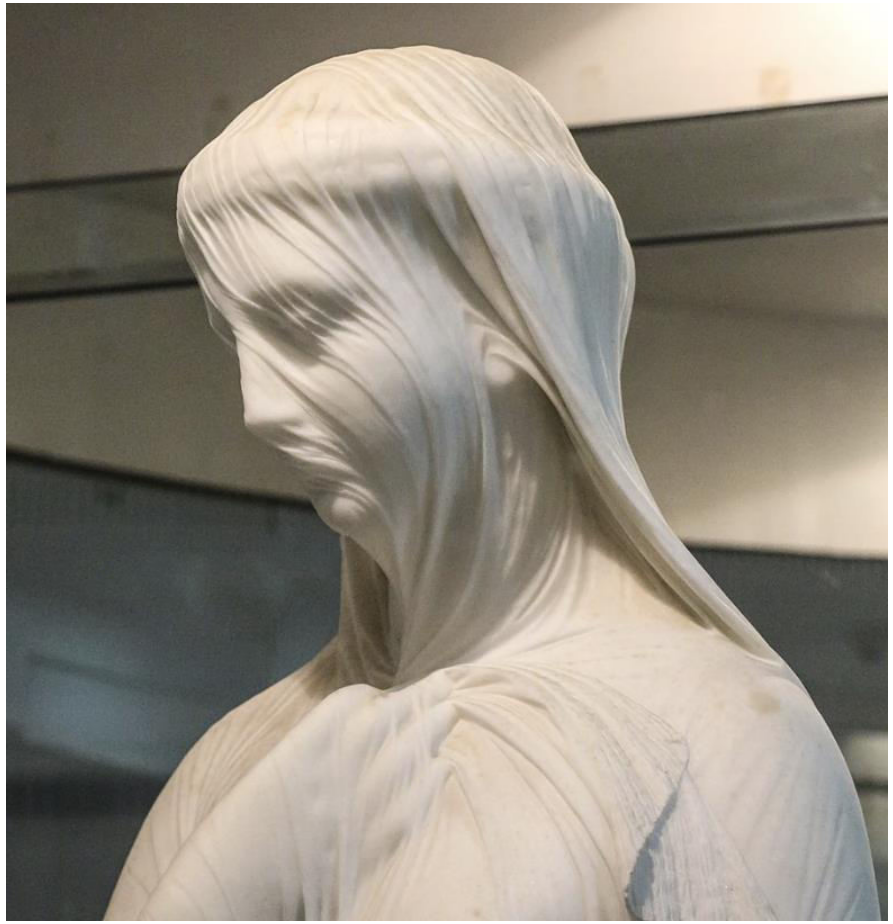
DREAM has much lower slowdown than PRAC at current TRH

MC-Side mitigation are quite useful (vs PRAC) for SOC vendors

Outline

- The Rowhammer Saga
- MINT: Minimalist Randomized Tracker
- MOAT: Secure Mitigation with PRAC
- MoPAC: Reducing the Slowdown of PRAC
- DREAM: MC-Side Mitigations with DRFM
- Parting Thoughts

Painting vs. Sculpture



Conclusion

Even after a decade, DRAM industry is unable to solve Rowhammer

- **Don’t throw TRR away – it is the lowest cost solution. Make it secure.**
- **PRAC still needs to be carefully architected to ensure security**
- **PRAC has high slowdown, need to reduce it for practical adoption**
- **SOC vendors can take matters in their hand with DRFM, still useful for DDR6**

Minimalism is key to ensuring security!

Performance matters: Everyone wants security, no one wants to pay