

Cutting DRAM Refresh Energy with Bank-Granular Rowhammer Mitigation

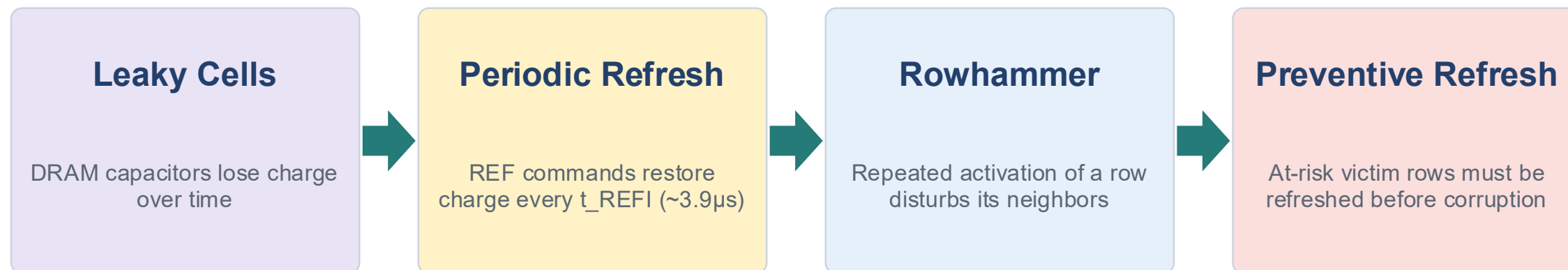
Jeeho Ryoo · Patrick Stockton · Cory Davis · Marco Ho · Michael Hsiao · Eugene B. John

Fairleigh Dickinson University · University of Texas at San Antonio · British Columbia Institute of Technology · Virginia Tech

ASAP 2026

DRAM refresh keeps data alive — but creates a security gap

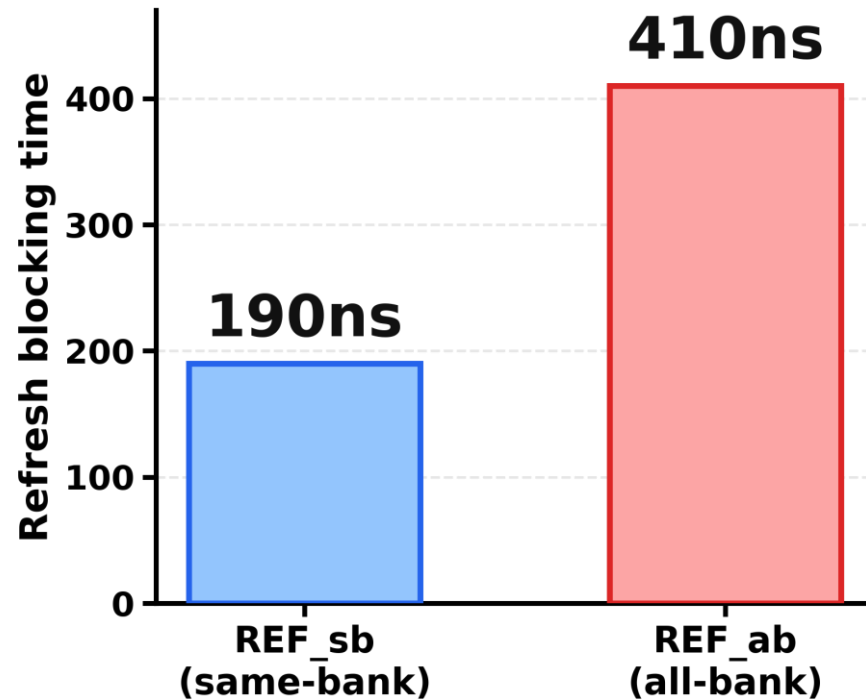
Leaky cells need periodic refresh; repeated row activation creates Rowhammer risk.



Rowhammer mitigation adds preventive refresh on top of periodic refresh — compounding the energy cost as DRAM density scales.

DDR5 already knows how to refresh efficiently

Same-bank refresh cuts blocking time in half — but only if you know which bank.



PRAC's Per-Row Activation Counting

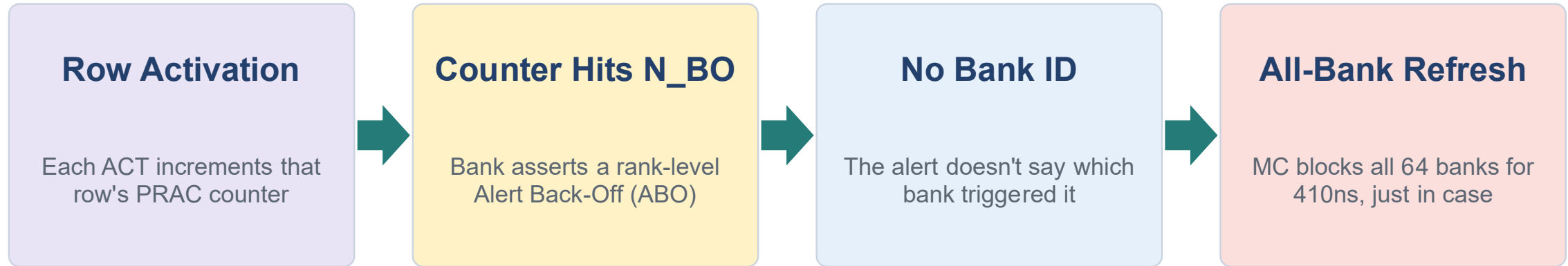
DDR5 tracks every row's activation count. When a counter reaches N_BO, the DRAM chip asserts a rank-level Alert Back-Off (ABO) to the memory controller.

But the ABO signal never says which bank triggered it. So preventive refresh can't use the fast same-bank command — it falls back to slow, rank-wide refresh.

The fix isn't a new DRAM command — it's a missing piece of information.

Rowhammer mitigation can't tell which bank needs help

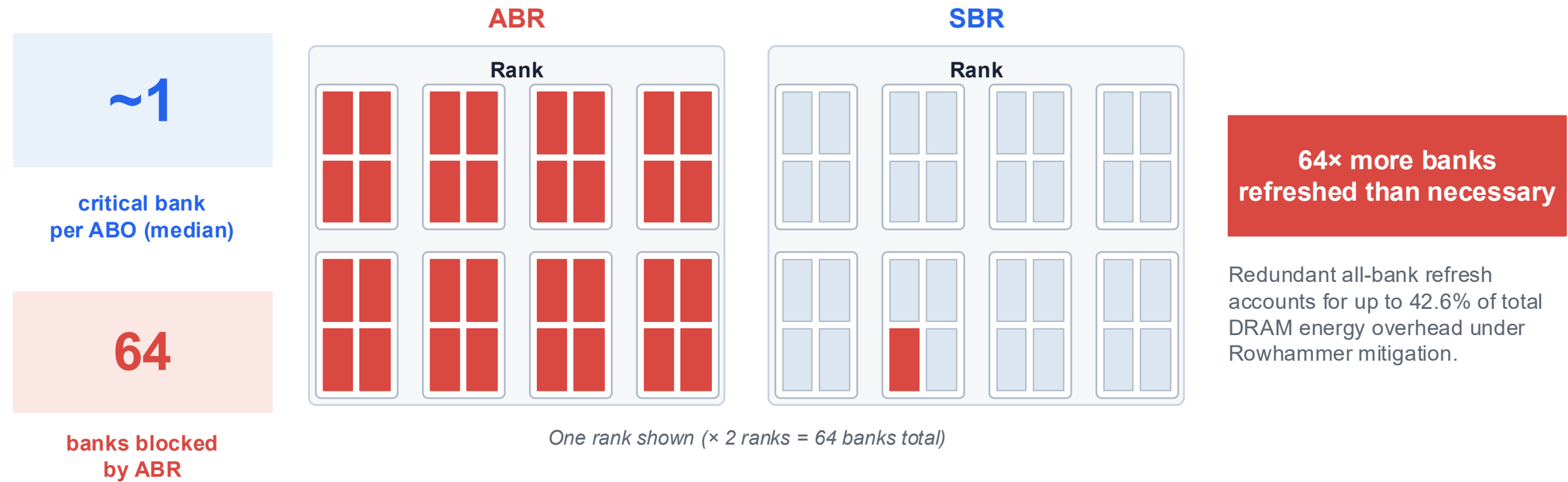
DDR5's rank-level alert forces the controller to refresh every bank, even when only one is at risk.



**The cost: 63 banks get refreshed for nothing —
redundant preventive refresh wastes both energy and time.**

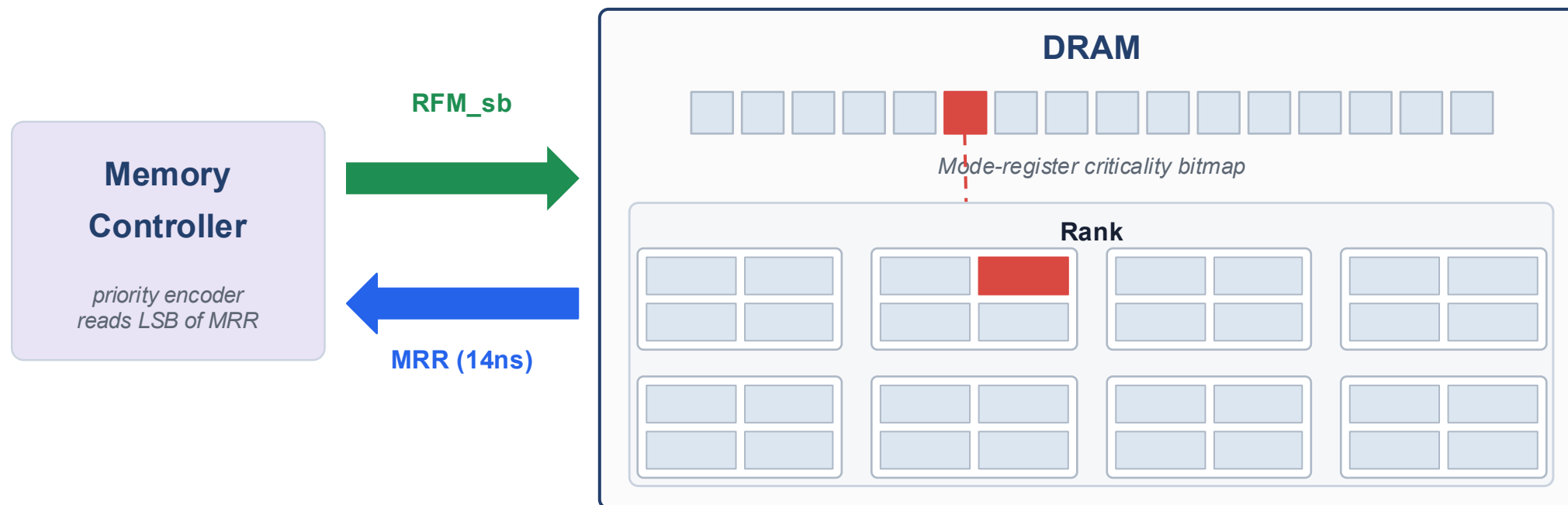
ABR blocks 64 banks to fix the one that's critical

Most ABO events involve only a single critical bank — the other 63 are refreshed anyway.



Inside the SBR architecture

A critical-bank bitmap gives the memory controller the missing piece: which bank to refresh.



The LSB position in the bitmap directly encodes the critical bank's ID — no lookup table needed.

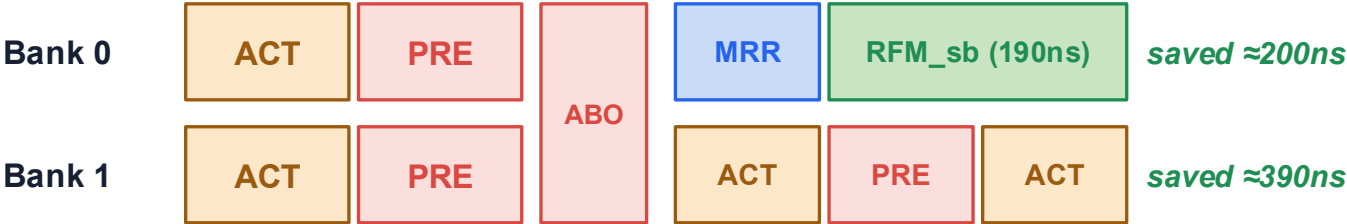
ABR blocks both banks; SBR blocks only one

A two-bank example where only Bank 0 is critical.

A All-Bank Refresh



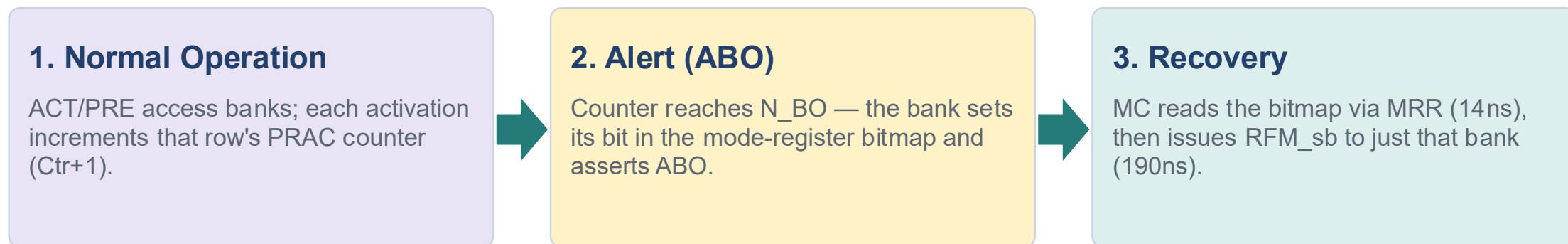
B Same-Bank Refresh (SBR)



Bank 1 resumes normal operation right after the 14ns MRR — it was never actually at risk.

SBR exposes per-bank criticality in a lightweight bitmap

The MC reads which bank is critical and refreshes only that one.



Per-bank criticality bitmap — mapped to reserved DDR5 mode-register bits



1 bit set → 1 bank flagged critical

The bitmap lives in existing DDR5 mode-register bits — no new pins, no new commands.

The hardware cost is a rounding error

SBR reuses existing DDR5 structures on both the DRAM die and the memory controller.

On the DRAM die

One AND gate + one write driver per bank, wiring the PRAC comparator to an RFU mode-register bit.

32

single-bit write paths on a 32-bank die —
under 1% of the PRAC counter area

On the memory controller

One priority encoder extracts the least-significant set bit from the 64-bit MRR value to select the target bank.

1

small combinational circuit — no new
queues or per-bank tracking structures

No new pins · no new DRAM commands · no changes to DDR5 timing constraints

SBR preserves PRAC's mitigation guarantee by construction

Only the refresh scope changes — detection and thresholds stay identical.

1. Detection Unchanged

PRAC counters, N_BO, and ABO assertion are internal to the DRAM die — identical under ABR and SBR.

2. Critical Bank Always Served

Each bank sets its bit before asserting ABO, so the MC's MRR always identifies the triggering bank.

3. Recovery Within Budget

MRR (14ns) + RFM_sb (190ns) complete well within the ABO_ACT / ABO_DELAY window.

Multi-bank criticality is handled too: the MC services critical banks sequentially, one per recovery window, until every flagged bank has been refreshed.

0 missed refreshes — benign & adversarial, every N_BO

56 workloads, adversarial traces, cycle-accurate simulation

Ramulator2 + DRAMPower, DDR5 JESD79-5C timing, N_BO swept 32 – 1,024.

System Configuration

Processor	4-core, 4GHz, 4-wide, 128-entry ROB
DRAM	32GB DDR5-6400, 1 channel, 2 ranks, 8 bank groups × 4 banks
Policy	Closed-page, interleaved mapping, FRFCFS
Controller	64-entry R/W queues, 8MB LLC

Workload Suite

56

multi-programmed 4-core workloads,
MPKI-binned Low / Medium / High

Drawn from SPEC CPU2006/2017, TPC, Hadoop,
MediaBench, and YCSB — plus adversarial attack traces that
repeatedly hammer aggressor rows across banks.

Representative results reported at N_BO = 64, a typical PRAC-style threshold.



SBR cuts DRAM energy while improving performance

Across 56 workloads and adversarial attack traces, at $N_{BO} = 64$.

27.8%

avg · up to 42.6% peak

DRAM energy reduction

3.2%

avg · up to 12.6% peak

performance speedup

0

at every N_{BO} threshold

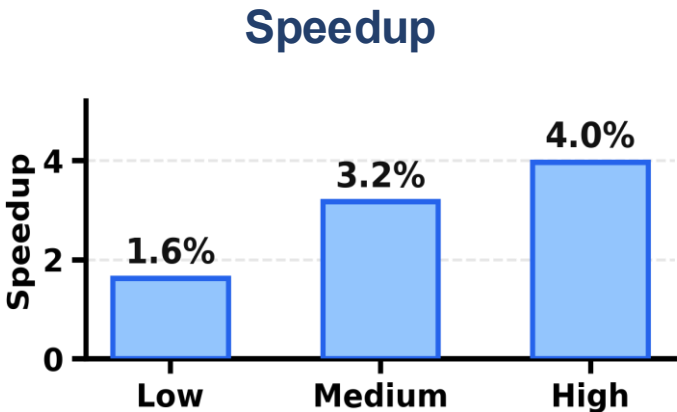
missed refreshes

Gains compound: a shorter refresh window, a narrower scope, and fewer alerts overall all pull the same way.

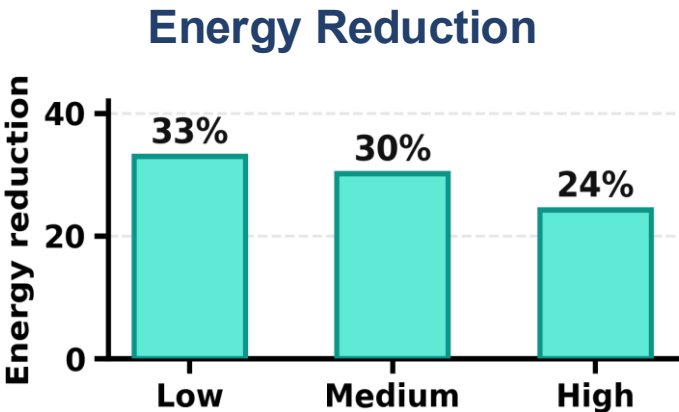


Gains hold across the full MPKI spectrum

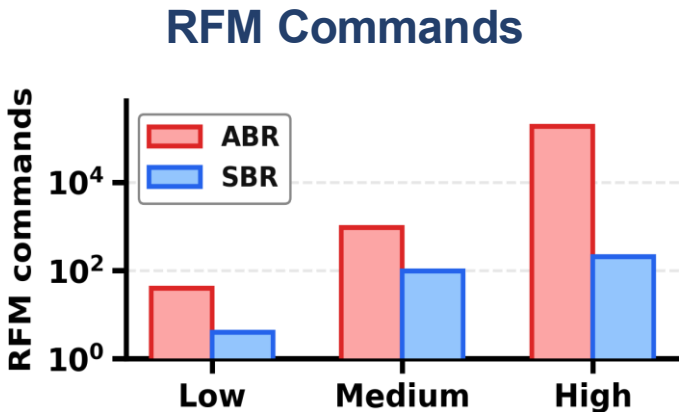
Low-, medium-, and high-MPKI workloads all benefit — for different reasons.



1.6 – 4.0%



24 – 33%

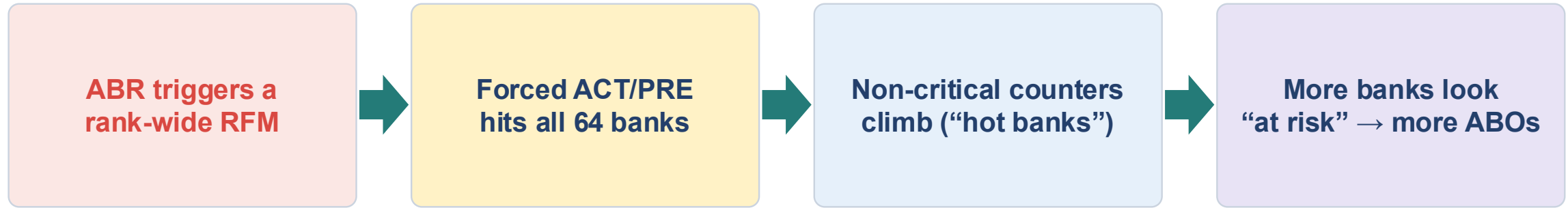


up to 912× fewer

Low-MPKI workloads save the most energy relatively; high-MPKI workloads cut the most commands — SBR wins across the board.

Rank-wide refresh doesn't proactively help — it inflates its own workload

ABR's own stalls push non-critical banks toward the threshold, creating a self-reinforcing cycle.



∪ the cycle repeats as N_BO tightens

These rows aren't actually at risk — periodic refresh resets their counters well before N_BO. SBR breaks the cycle by confining forced activations to the one bank that's truly critical.

In the worst-case workloads, this inflation makes ABR issue up to 63.1% more preventive refreshes than SBR.

SBR is a drop-in replacement underneath any PRAC scheme

Prior work optimizes when and how alerts are scheduled; SBR optimizes where refresh happens.

Prior work: tracking & scheduling

QPRAC multi-entry priority queues for mitigation ordering

MOAT dual-threshold, coordinated activation tracking

Chronus dynamic back-off timing under varying conditions

SBR: execution scope

Same RFM commands these schemes already issue — SBR just targets them to the critical bank instead of the whole rank. No changes to detection or prioritization logic required.

Validated with MOAT: MOAT-SBR and PRAC-SBR both outperform their ABR counterparts in every MPKI category — the two optimizations compound rather than overlap.

Bank-granular refresh closes the gap PRAC left open

SBR turns rank-wide guesswork into precise, per-bank recovery.

27.8% avg
(up to 42.6%)

energy reduction

3.2% avg
(up to 12.6%)

speedup

0
missed refreshes

correctness maintained

DDR5's rank-level alert hides which bank is critical — so ABR refreshes all 64 to fix the one that matters.

SBR closes that gap with a lightweight bitmap in existing mode-register bits — no new pins, no new commands.

Thank you