

HYDRA: Enabling Low-Overhead Mitigation of Rowhammer at Ultra-Low Thresholds via Hybrid Tracking

Moinuddin Qureshi

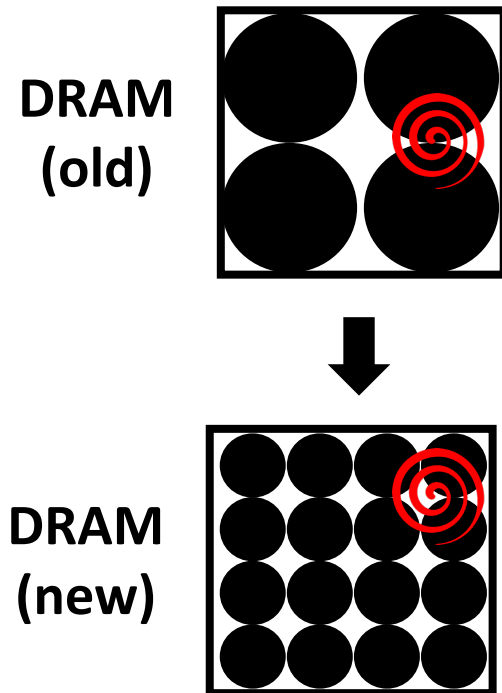
Aditya Rohan, Gururaj Saileshwar, Prashant Nair



Rowhammer Attacks

DRAM Scaling for Increased Capacity

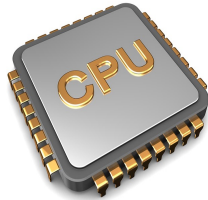
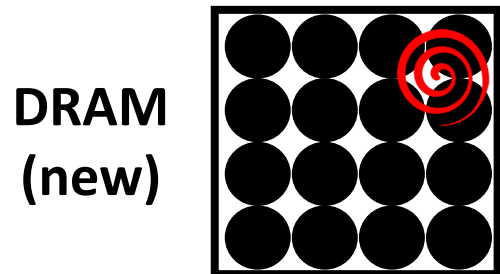
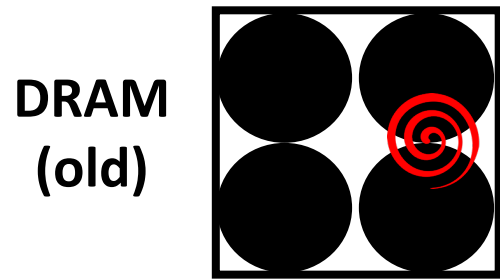
More Inter-Cell Interference



Rowhammer Attacks

DRAM Scaling for Increased Capacity

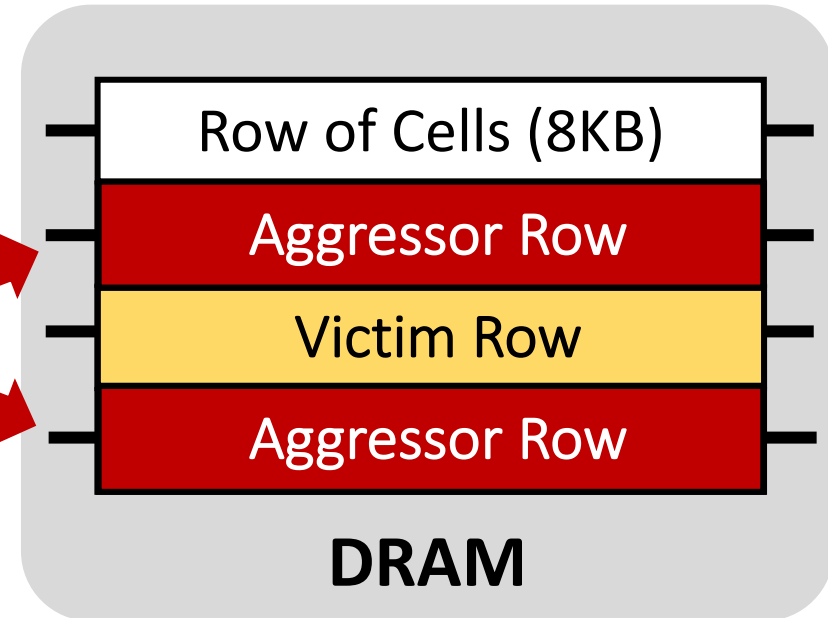
More Inter-Cell Interference



Rapid
Accesses



Rowhammer Attack



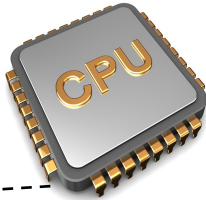
Bit-Flips in Neighboring Rows

Rowhammer Attacks

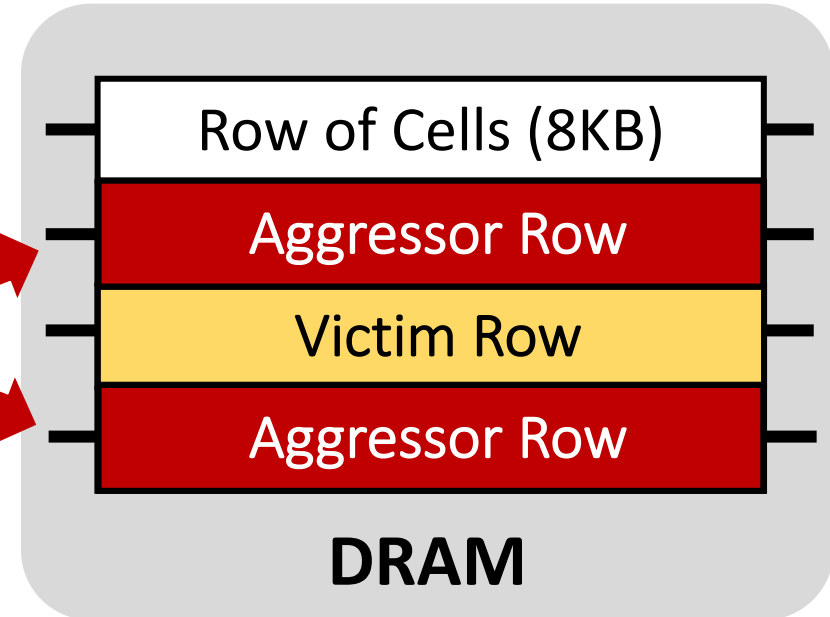


Rowhammer Attack

```
loop:  
  mov (X), %eax  
  mov (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp loop
```



Rapid
Accesses



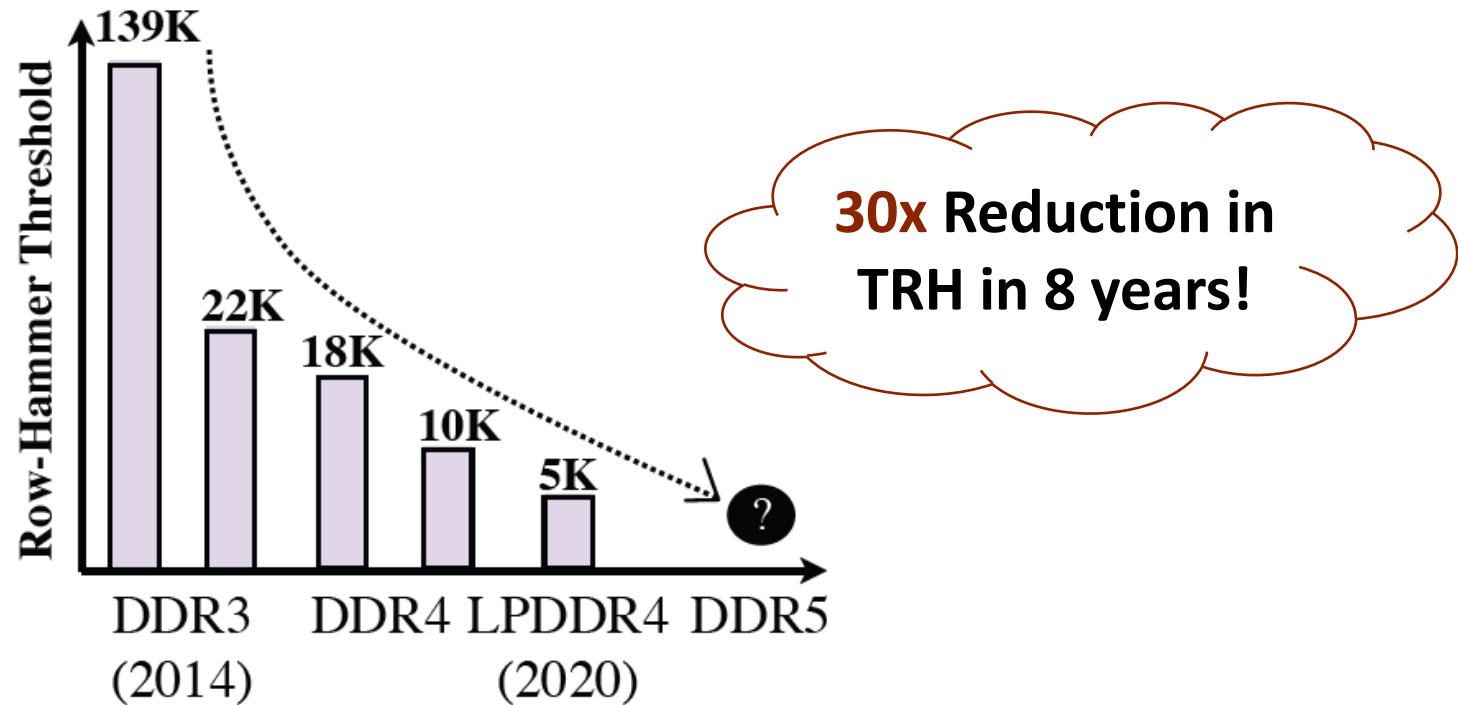
Bit-Flips in Neighboring Rows

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

[Seaborn+, Blackhat'15]

Rowhammer is Getting Worse!

Rowhammer Threshold: Bitflips require activating at least one row TRH times

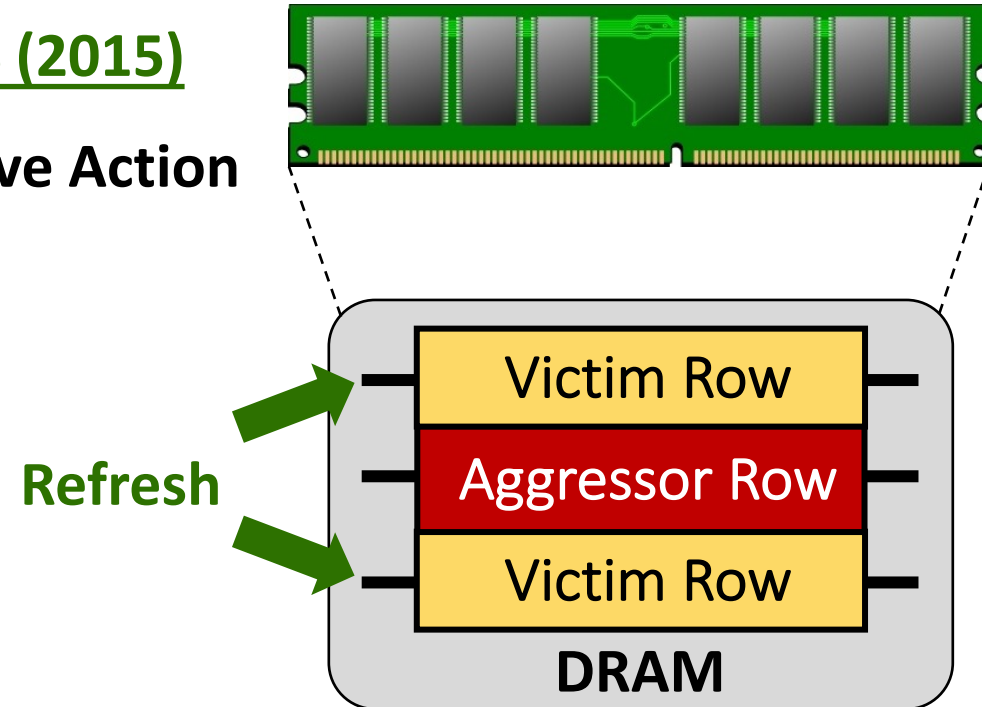


Solutions must tolerate not only current TRH but future TRH (Our goal: TRH \leq 500)

Typical Mitigation for Rowhammer

Targeted Row Refresh (TRR) in DDR4 (2015)

- 1 Track Aggressor Rows
- 2 Mitigative Action



Typical Mitigation for Rowhammer

Targeted Row Refresh (TRR) in DDR4 (2015)

- ~~1 Track Aggressor Rows~~
- 2 Mitigative Action

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

**Poor Rowhammer Fixes On DDR4 DRAM
Chips Re-Enable Bit Flipping Attacks**

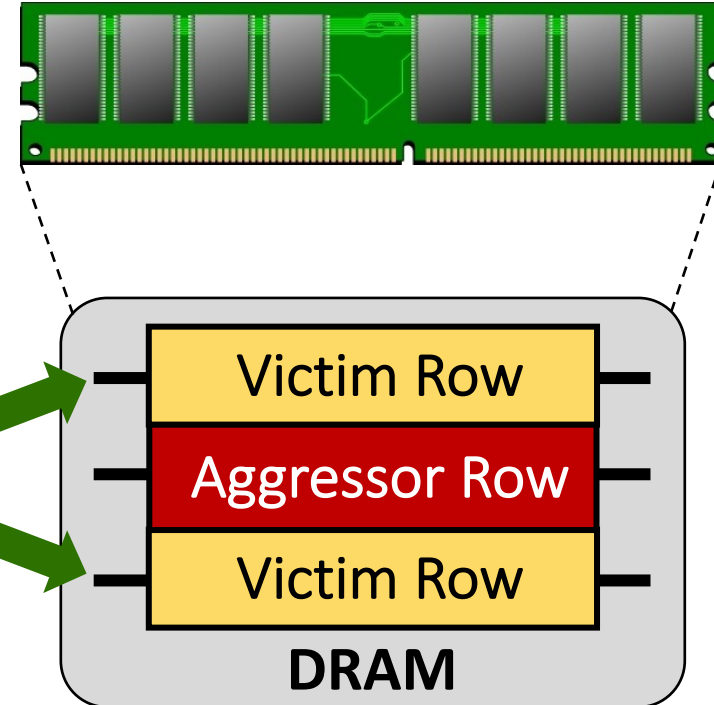
Source: The Hacker News

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

Refresh



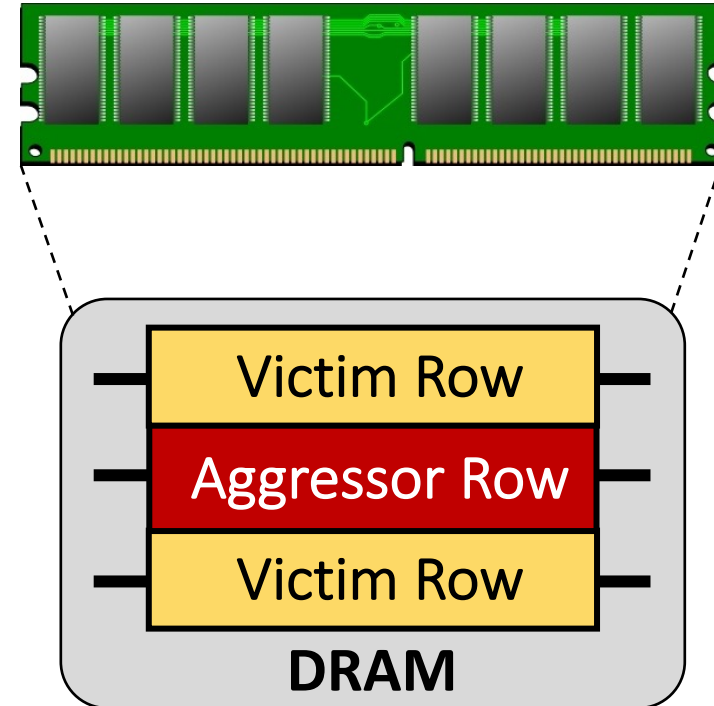
Source: The Register

Why Trackers Break?

Refresh Period: **64ms**, Row-Cycle Time: **45ns**

Activation Budget Per Bank: **~1.4 Million**

	Max. Attack Rows	
RH Threshold	1-Bank	32-banks
100,000	14	448



The number attack rows increases inversely to Threshold, so does tracker state

Pitfall of SRAM-Based Trackers

SRAM-tables can be in **memory controller** (CAT) or **inside DRAM** chip (TRR)
Overheads for 16GB Rank (16 Banks, 8KB Rows)

RH Threshold	Graphene (100% CAM)	TWiCE	CAT	D-CBF (Blacklist only)	Naive
32K	5KB	37KB	25KB	53KB	3.8MB

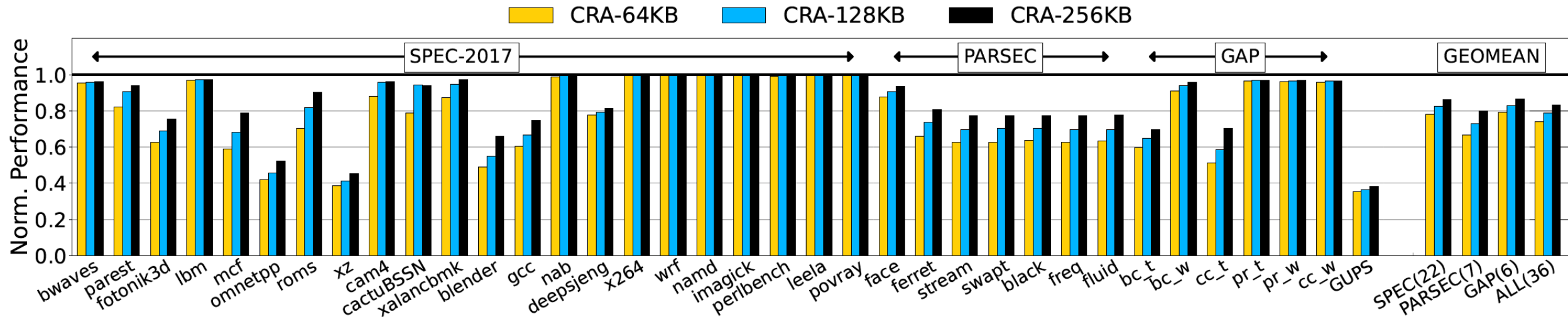
Our goal is < 64KB SRAM per Rank

SRAM-Based trackers incur impractical SRAM overheads at ultra-low thresholds

Pitfall of DRAM-Based Trackers

Counter-Based Row Activation [Kim, Nair, Qureshi - CAL'14]:

Keep counters for ALL rows in a dedicated region in DRAM, cache counter-lines on-chip



DRAM-based trackers incur low SRAM storage but high performance overhead (25%)

Goal

Develop secure and low-cost Rowhammer mitigation:



- ✓ **Effective at Ultra-Low thresholds (500 or below)**
- ✓ **Low SRAM overhead (<64KB per rank)**
- ✓ **Low performance overhead (< 1%)**
- ✓ **No changes to DRAM arrays or memory interfaces**

Observation and Insight

Rowhammer: Race against time (64 ms)

- Access many rows few times ✓
- Access few rows many times ✓
- Access many rows many times ✗

Applications (within 64ms):

- Access 100k+ rows
- On average, few 10s of ACT/row
- Few thousand rows have 250+ ACT

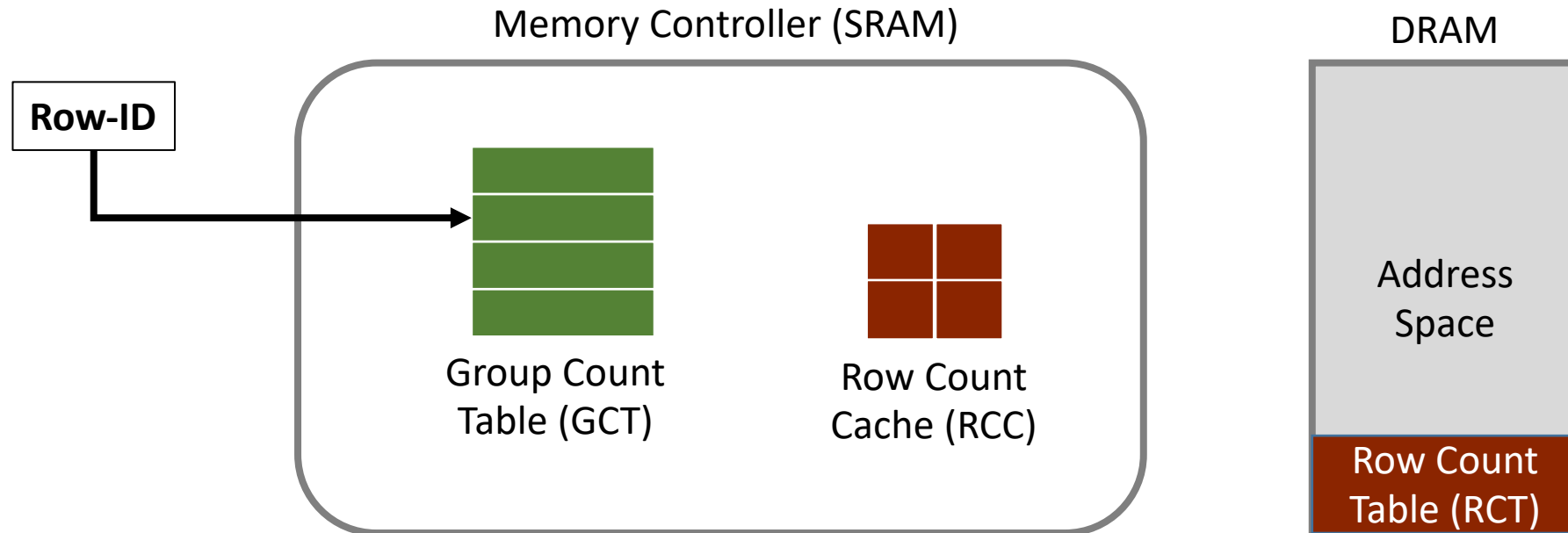
Insight:

- Have **ability** to track all rows (DRAM)
- Use SRAM to **filter** out common-case

Workload	MPKI LLC	Unique Rows*	ACT-250+ Rows*	ACTs Per Row*
bwaves	39.6	77.9K	0	38.6
parest	27.6	13.8K	5,882	237
fotonik3d	25.9	212K	0	17.5
lbm	25.6	41.8K	0	82.1
mcf	20.8	112K	0	28.8
omnetpp	9.75	312K	195	10.7
roms	9.15	115K	1,169	22.9
xz	5.87	102K	1,755	26.4
cam4	3.23	45.5K	5	54.1
cactuBSSN	3.20	24.6K	4,609	107
xalancbmk	1.61	60.8K	0	49.8
blender	1.52	52.4K	2,288	58.7
gcc	0.65	144K	159	18.0
nab	0.61	61.9K	0	31.9
deepsjeng	0.29	802K	0	1.78
x264	0.28	25.0K	0	34.0
wrf	0.27	19.3K	18	20.9
namd	0.26	24.7K	0	34.9
imagick	0.16	10.7K	0	19.1
perlbench	0.09	25.6K	0	5.88
leela	0.03	0.72K	0	2.68
povray	0.03	0.50K	0	2.28

HYDRA: Hybrid Tracker

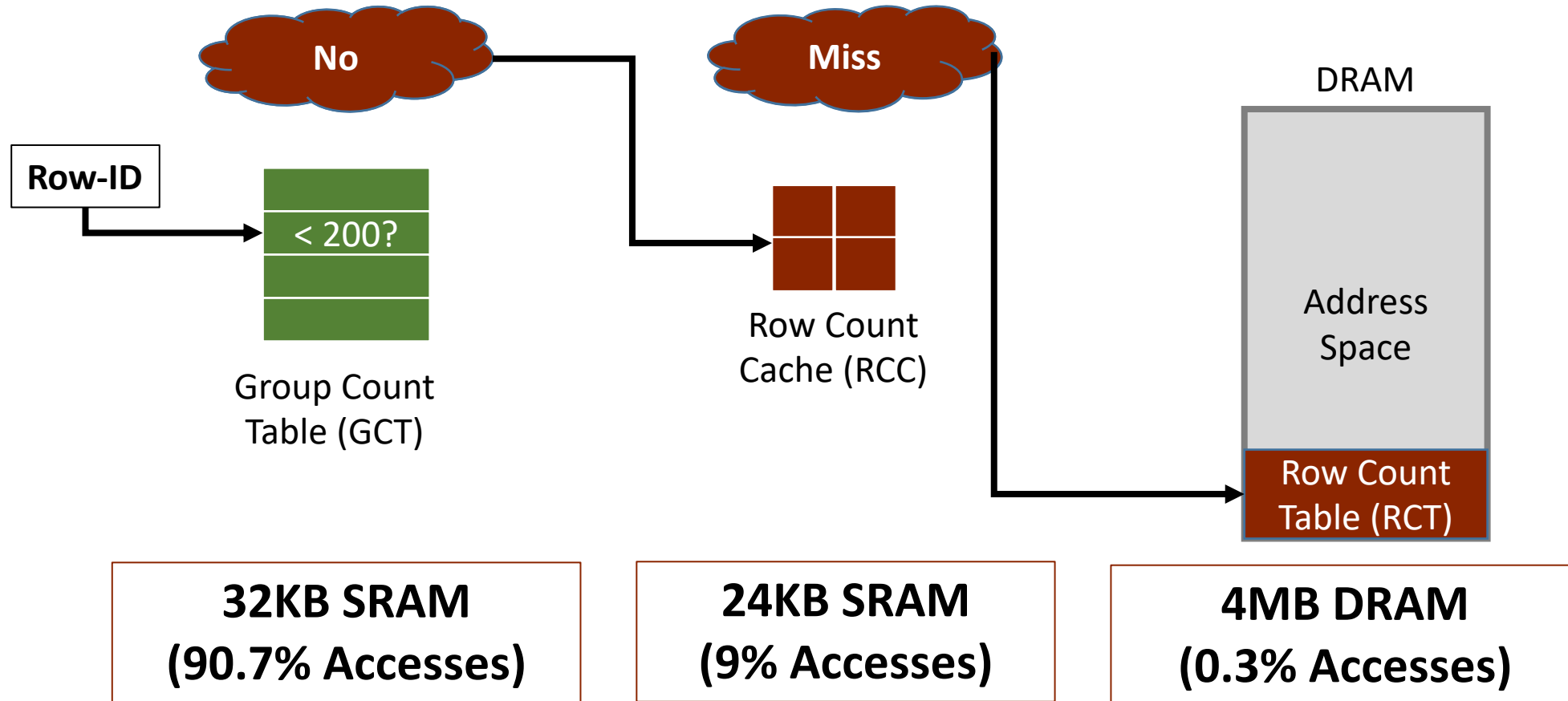
HYDRA: SRAM for group-tracking, and DRAM for per-row tracking (cached)



HYDRA uses DRAM to get scalable tracking, and SRAM to avoid performance overheads

HYDRA: Operation

Assume: Hydra Threshold = 250, Group Threshold = 200

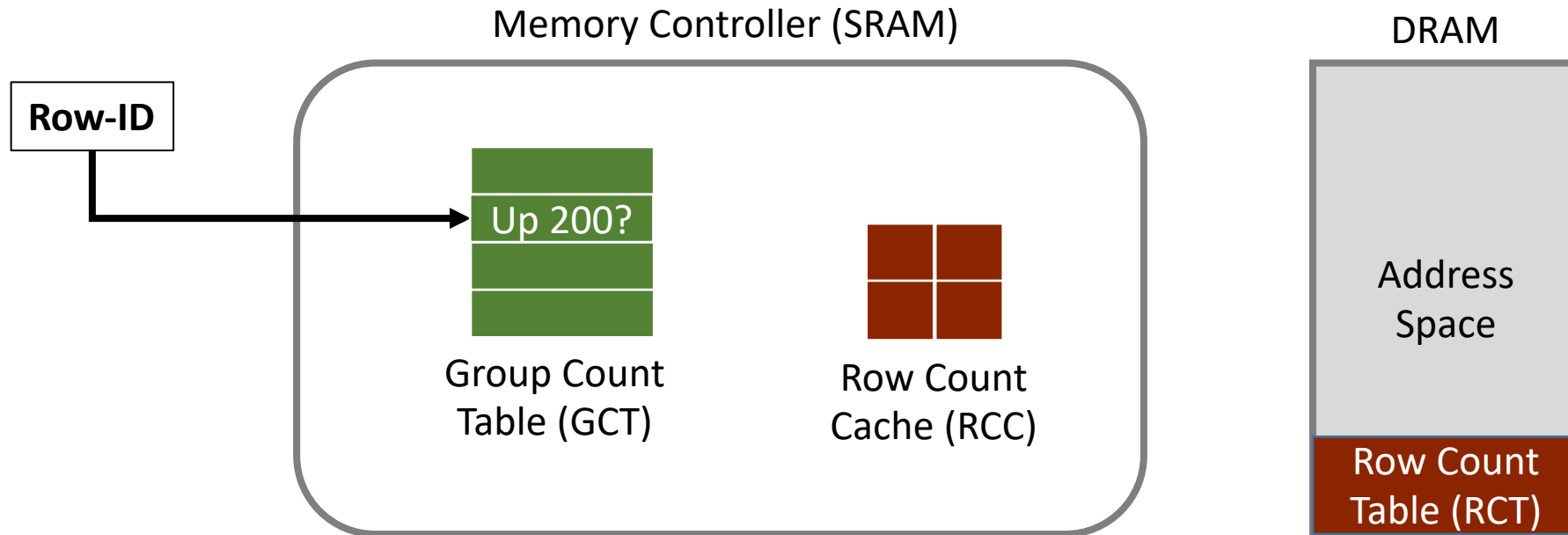


The GCT filters most of the counter checks, less pressure on RCC, minimal RCT access

HYDRA: Reset and Set

Every 64ms, **Reset** SRAM state

When GCT-entry reaches 200
Set relevant RCT-entries to 200




If a Row is accessed continuously, first mitigation may be at 50, then 250 each

HYDRA uses intelligent indexing to reduce RCT set overhead (128 counters in two lines)

Security Analysis

- HYDRA Threshold is set to $TRH/2$ due to periodic reset (**TH=250, TRH=500**)
- Proof that HYDRA issues a mitigation at-least once per TRH
- Protection for RCT rows (their counters kept in SRAM, 500 bytes)
- Protection against Half-Double (mitigations increment count for victim)
- HYDRA works with any mitigating action (BH/RRS): **Refresh 2 rows on each side**

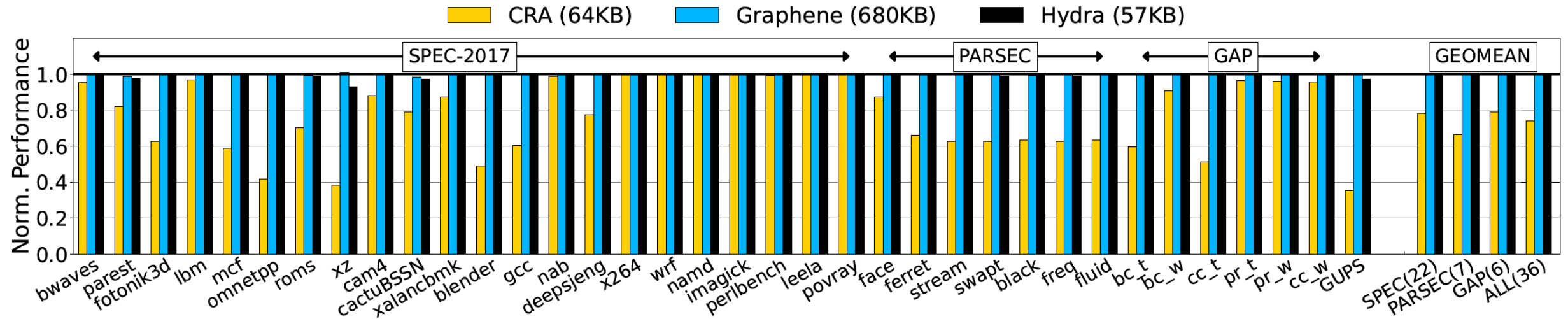


Security NOT
dependent
on pattern

For detailed security analysis, please see the paper

Results

Config: 8-core OOO, 32GB DRAM (2 Ranks, 8KB Row-Buffer)



HYDRA has negligible slowdown (0.7%) and low SRAM overheads (57KB for two ranks)

Say Hello! to DDR5

SRAM overheads for 32GB (2 Ranks) for DDR4 (current) and DDR5 (soon)

Scheme	DDR-4 (16 banks/rank)	DDR-5 (32 banks/rank)
Graphene	640 KB (CAM)	1.3 MB (CAM)
TWiCE	4.6 MB	9 MB
CAT	3 MB	6 MB
D-CBF	1.5 MB	1.5 MB
HYDRA	56.5 KB	56.5 KB

DRAM overhead of HYDRA is negligible (4MB out of 32GB, 0.01%)

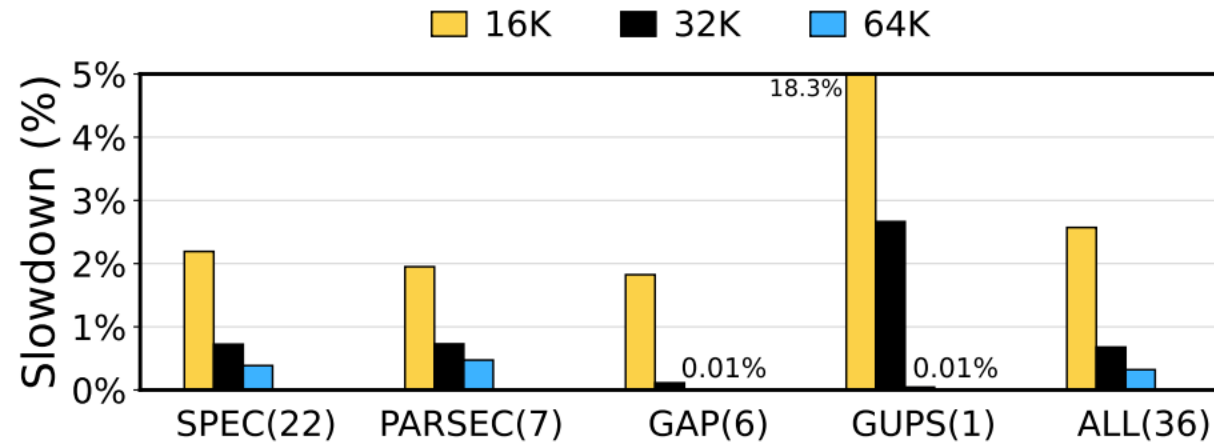
HYDRA provides scalable Rowhammer mitigation even for DDR5

Conclusion

- Rowhammer is a moving target: Threshold keeps reducing (30x in 8 years)
- Existing SRAM trackers needs unacceptable storage (at $TRH \leq 500$)
- Existing DRAM-based tracker incurs unacceptable slowdown (25%)
- We develop **HYDRA**, a hybrid tracker, that combines the best of both
 - Ability to track arbitrary number of rows (good for security)
 - Filter per-row updates using SRAM for group tracking (avoids slowdown)
- HYDRA incurs **0.7% slowdown** and needs **28KB/rank (at $TRH=500$)**

Sensitivity to GCT Capacity

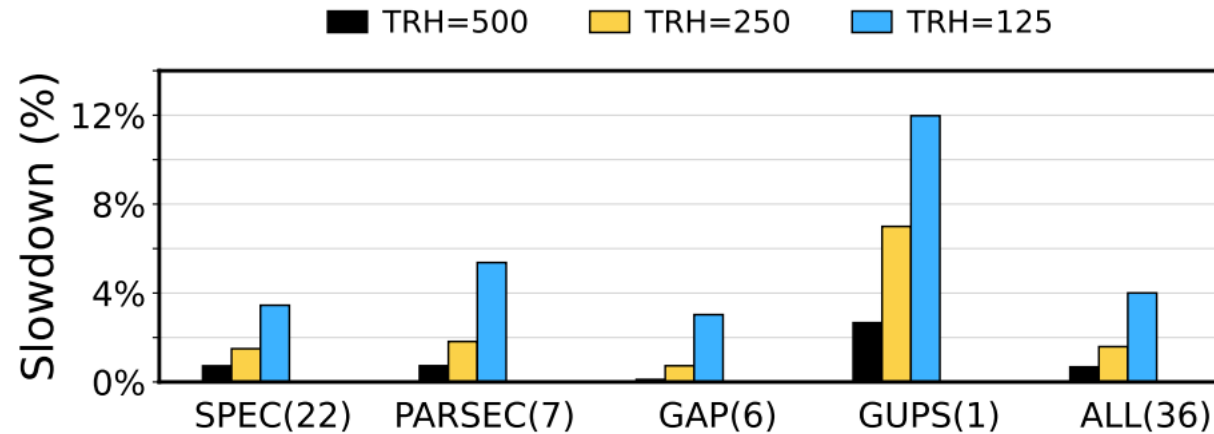
We use a default GCT of 32K



A larger GCT virtually eliminates all slowdown (except for mitigating actions)

Sensitivity to RH Threshold

Structures scaled 2x for TRH=250, and 4x for TRH=125



HYDRA has can provide scalable Rowhammer mitigation even at low threshold

GCT vs RCC: Which One?

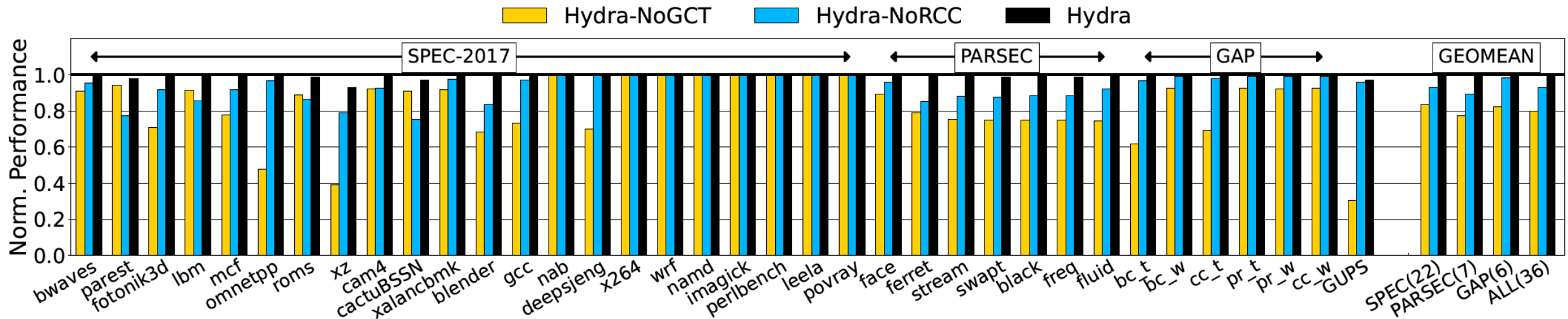


Figure 8: Slowdown of Hydra without GCT or RCC. The average slowdown of Hydra-NoRCC is 4.5% and Hydra-NoGCT is 20%.

GCT is the primary source of effectiveness (filtering 90% of the accesses)

D-CBF

Comparison with D-CBF: Both D-CBF and Hydra (GCT) use *filters* to identify (possibly) *hot-rows*, however, these two proposals are at radically different design points. As D-CBF is a single-line of defense, D-CBF must be over-provisioned to support extremely low false-positive rates. Whereas, Hydra uses three lines of defense, GCT for identifying (possibly) hot-rows, then the RCC for caching per-row count, and RCT for providing unconstrained storage (if both the GCT and RCC fail). Thus, Hydra can easily use a small filter and handle overflows. Furthermore, D-CBF can support mitigation via only delay and not victim refresh (once the entry in the filter saturates, it stays in that state until reset). Unfortunately, inserting a delay is not viable at ultra-low thresholds.⁶ Hydra can support victim refresh as it can reset the per-row state.

⁶For example, at $T_{RH}=500$, about 250 activations would go in identifying the hot-row, and the remaining 250 activations must be spread over almost 64ms, which means the access rate to the hot-row would get limited to once every 0.25 millisecond, which is almost 2000 \times lower than the access rate possible in the baseline. We note that such Denial-of-Service would occur even in regular workloads as we observe that several workloads have a few thousand rows receiving 250+ activations (Table 3).