GDDRHammer: Greatly Disturbing DRAM Rows — Cross-Component Rowhammer Attacks from Modern GPUs

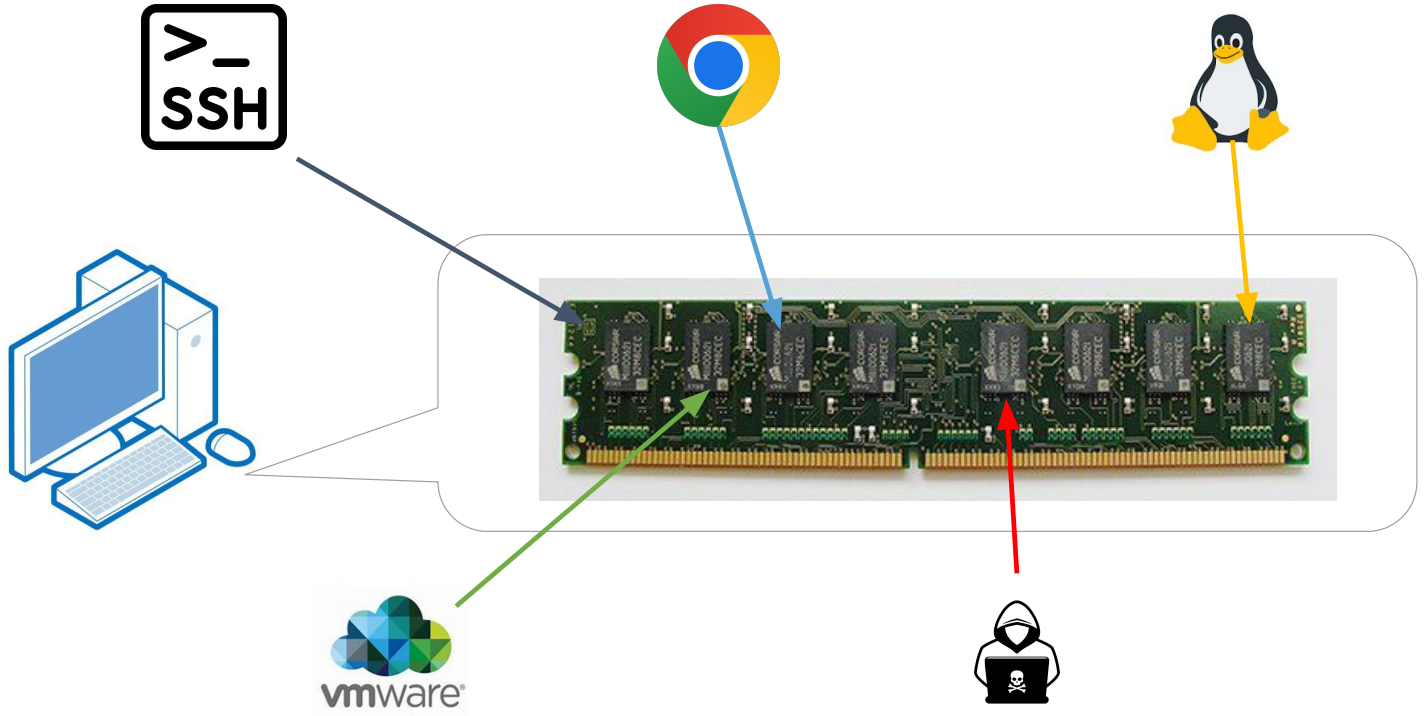
Yichang Hu, Noah Brown, Yuhang Chen, Joshua Bakita, Tianlong Chen, Daniel Genkin, Andrew Kwong

IEEE S&P 2026

Department of Computer Science

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

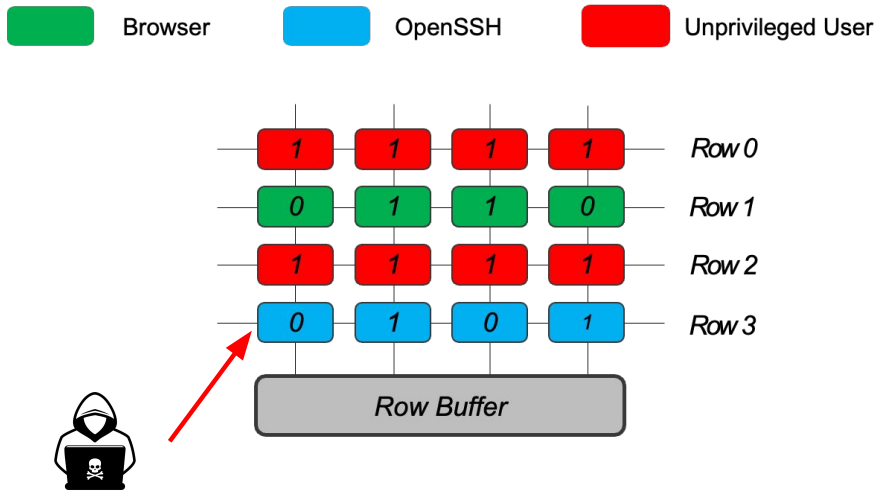
Memory Isolation



Memory Isolation

- OS enforces isolation
- Can physics help us bypass OS and access memory across these boundaries?

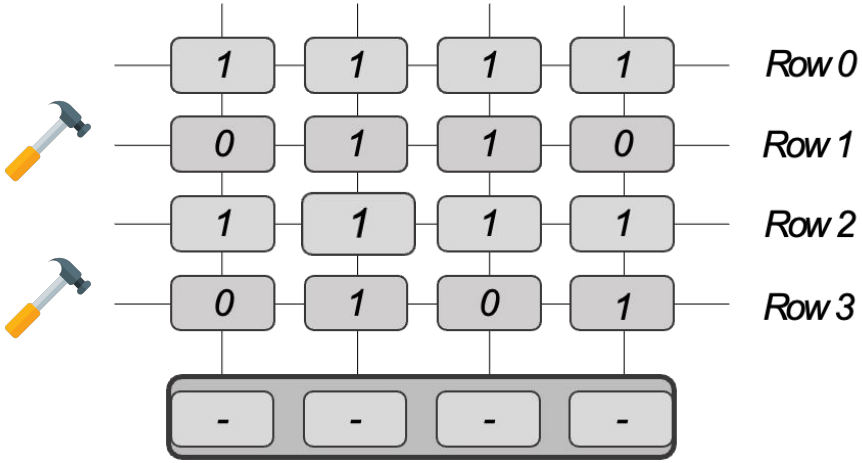
Yes. via Rowhammer!



Rowhammer

- Activating a row drains charge from nearby capacitors
- Repeated activation of rows accelerate the charge leakage in neighboring cells

- Attacker that controls values in rows 1 and 3 writes to victim's memory in row 2



Aggressor Row

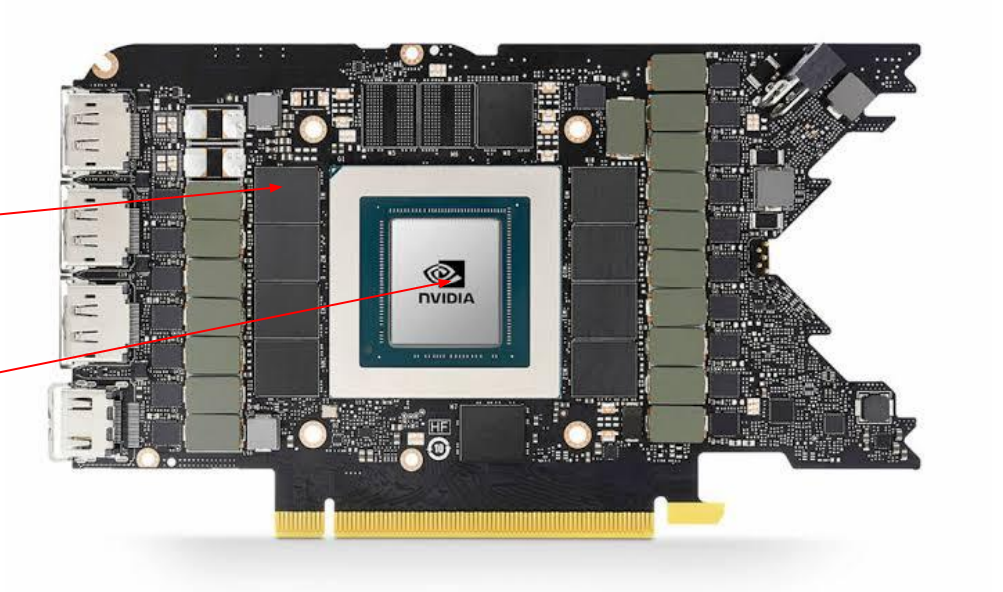
Victim Row

Aggressor Row

Bit flip!

Is GPU Memory Safe from Rowhammer?

Independent PCB and memory hierarchy



GPUHammer

- First Rowhammer on RTX A6000 GPU (GDDR6 memory).
- 8 unique flips in 4 banks (0.5 GB).
- Degrades ML model accuracy by up to 80% with a flip.

Limitations

- *Few* flips on *only* one GPU
- Impact of flips *only* on GPU applications

Lin, Chris S., Joyce Qu, and Gururaj Saileshwar. "{GPUHammer}: Rowhammer attacks on {GPU} memories are practical." *34th USENIX Security Symposium (USENIX Security 25)*. 2025.

Research Questions

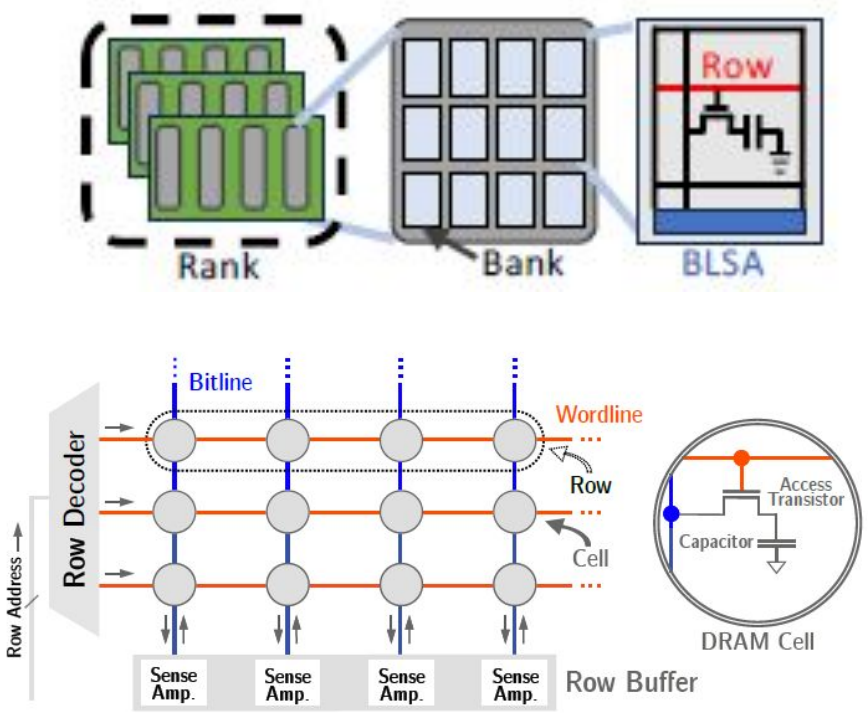
- What is the true prevalence of Rowhammer-induced bit flips on GPU?
- What are the security implications of GPU-based Rowhammer on the security of the host system?
- How can such flips be exploited by attackers?

Overview

- Background
- Novel Rowhammer Techniques
- Comprehensive Study
- GPU-to-CPU Exploit
- Discussion

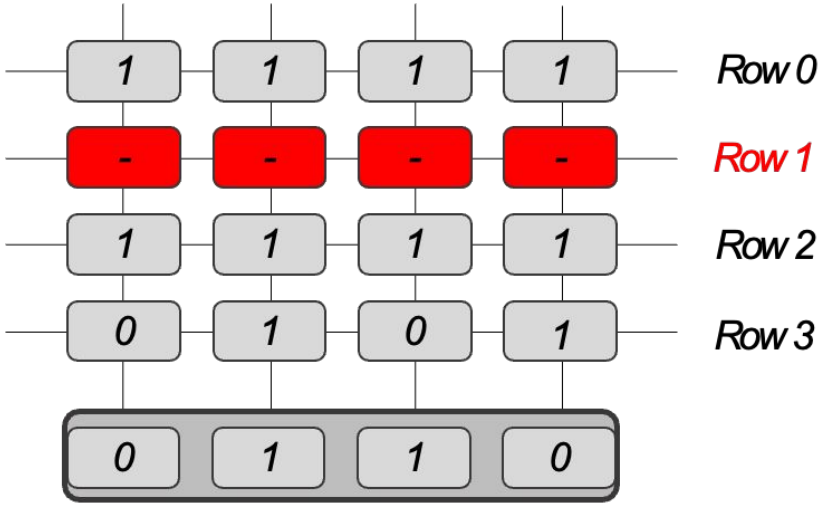
Background

DRAM


Rank
Bank
Row
BLSA
Bitline
Wordline
Row Decoder
Row
Cell
Row Address
Access Transistor
Capacitor
DRAM Cell
Sense Amp.
Sense Amp.
Sense Amp.
Sense Amp.
Row Buffer

How DRAM Works

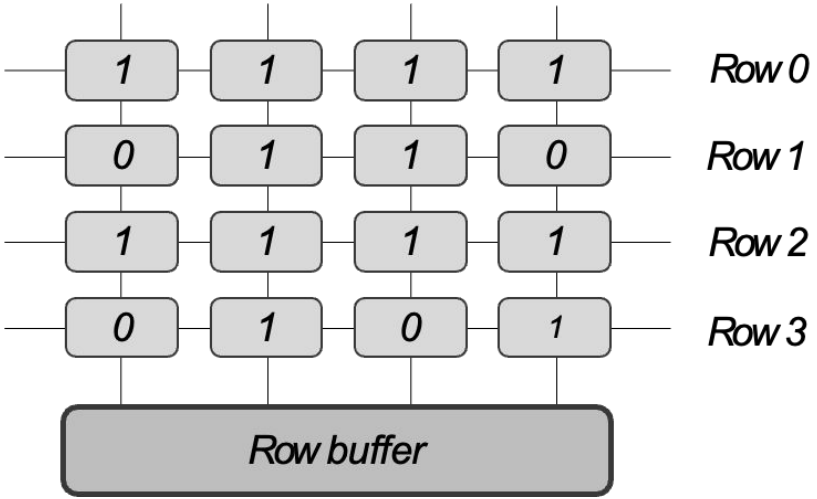
Read operation: Row 1



ACTIVATE Row 1

How DRAM Works

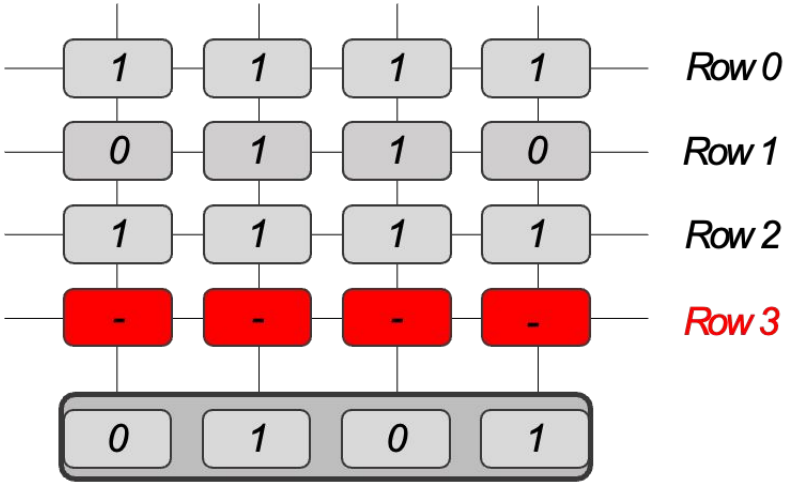
Read operation: Row 3



PRECHARGE Row 1

How DRAM Works

Read operation: Row 3



ACTIVATE Row 3

Refresh Operations

A DRAM cell must be refreshed within ***tREFW***

One refresh command is issued in ***tREFI***

DDR4:

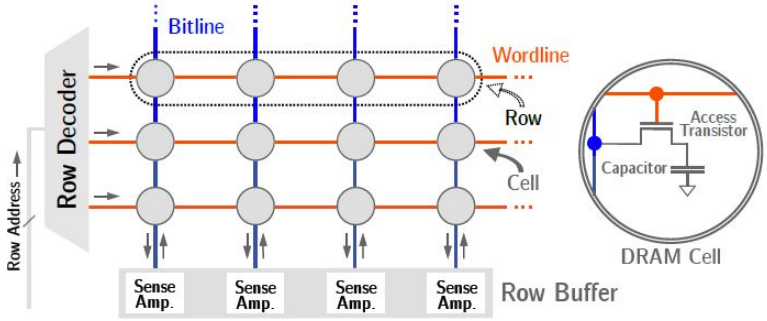
tREFW: 64ms

tREFI: 64ms / 8K = 7.9us

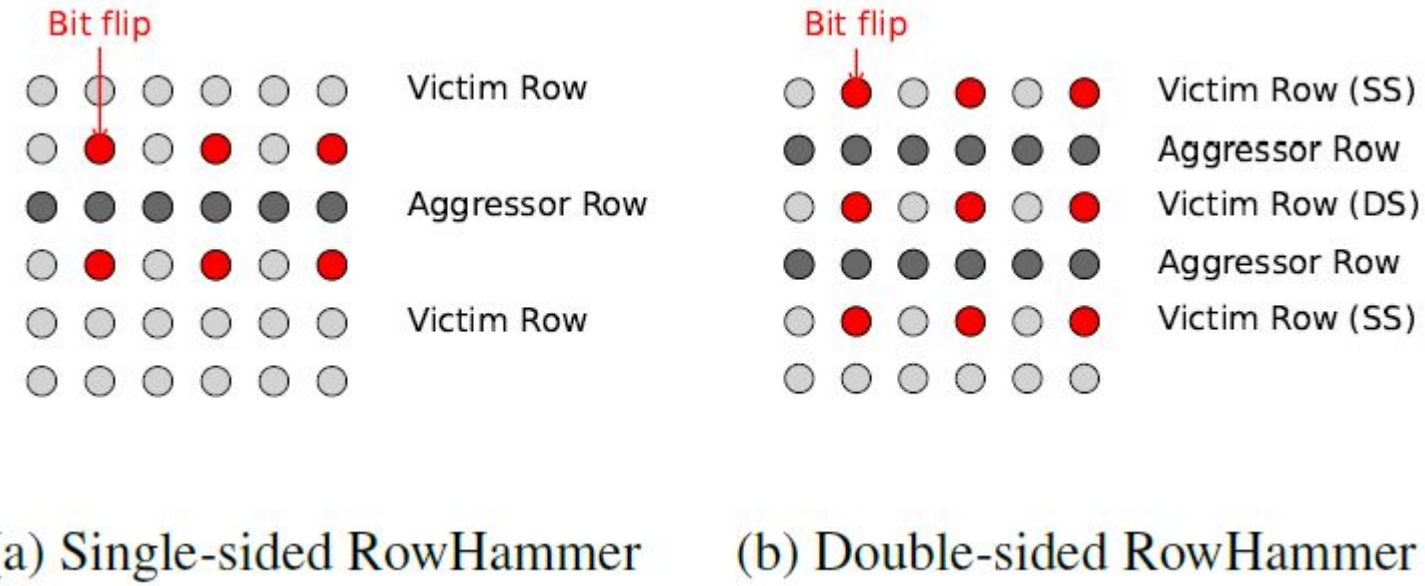
DDR5:

tREFW: 32ms

tREFI: 32ms / 16K = 1.9us



Rowhammer Access Pattern



(a) Single-sided RowHammer (b) Double-sided RowHammer

Target Row Refresh

Sampling-based; Detects hot rows; Refreshes neighboring victim rows

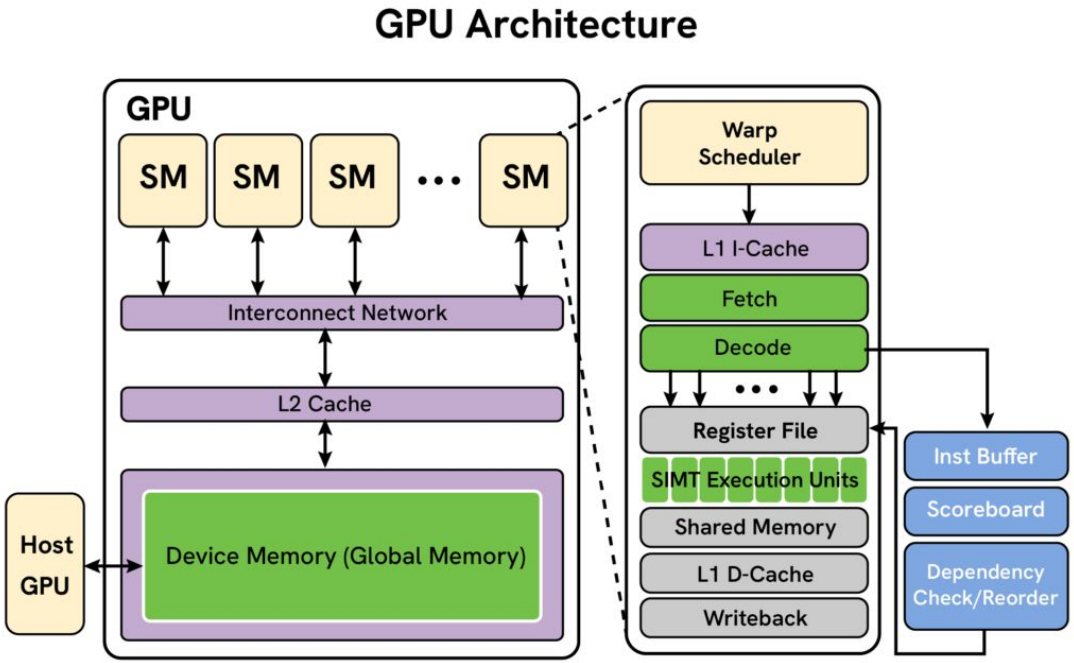
- Issued with tREFI
- Limited tracking capacity
- Samples only part of the tREFI

How to bypass:

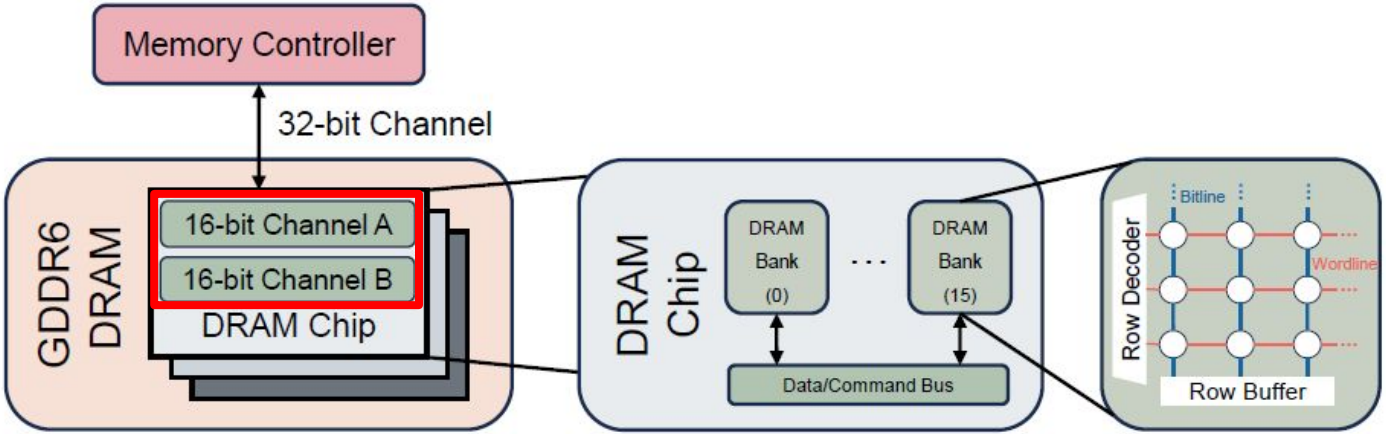
- More dummy ACTs to overwhelm the sampler
- Craft the aggressor at specific time period to evade detection

Synchronized Rowhammer!

GPU Architecture



GDDR6 DRAM



refresh interval (*tREFI*): 1.9 us (32 ms / 16 K)
refresh window (*tREFW*): 32 ms

GPU Execution Model

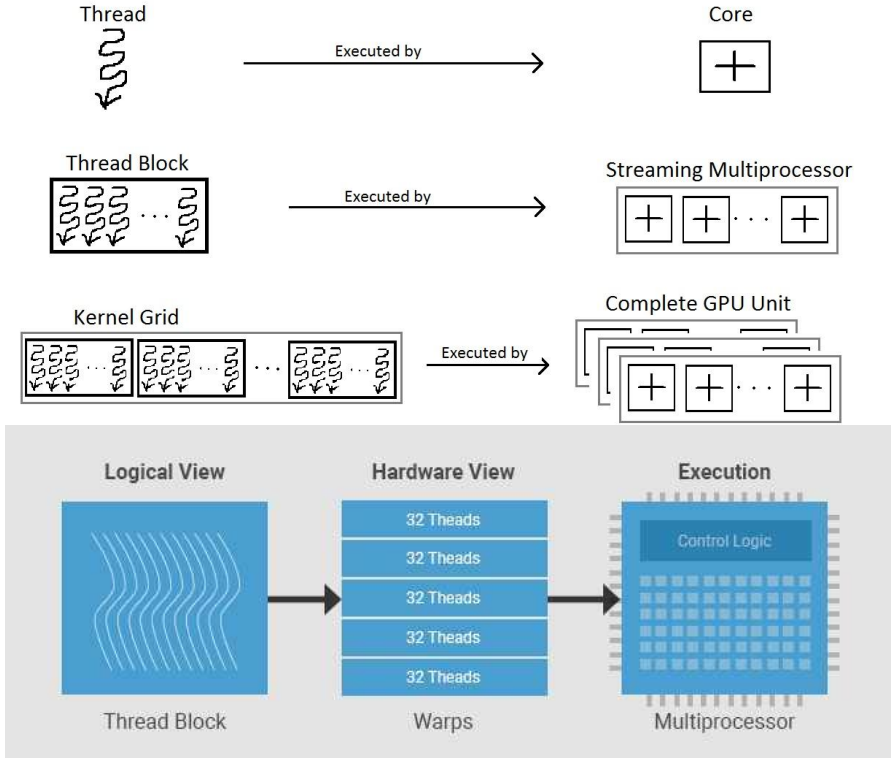
warp size: **32** threads

threads within a warp:

- SIMD
- lock-step under same counter

```
t0 =  clock()
if(thread_id == 0){
        …
}
t[thread_id] = clock() - t0
```

t[0] = 500 cycles!
t[1] = 500 cycles!
…
t[31] = 500 cycles!



Rowhammering GPUs

Reproduce GPUHammer

Number of flips: 11 flips in 4 banks in our A6000 GPU

Hammer pattern: 24-sided; distance - 4 (rows *n, n + 4, … n + 4k*)

Hammer order: naive; multi-warp, multi-thread (8 warps * 3 threads)

Hammer time: ~30 hours for 4 banks

Lin, Chris S., Joyce Qu, and Gururaj Saileshwar. "{GPUHammer}: Rowhammer attacks on {GPU} memories are practical." *34th USENIX Security Symposium (USENIX Security 25)*. 2025.

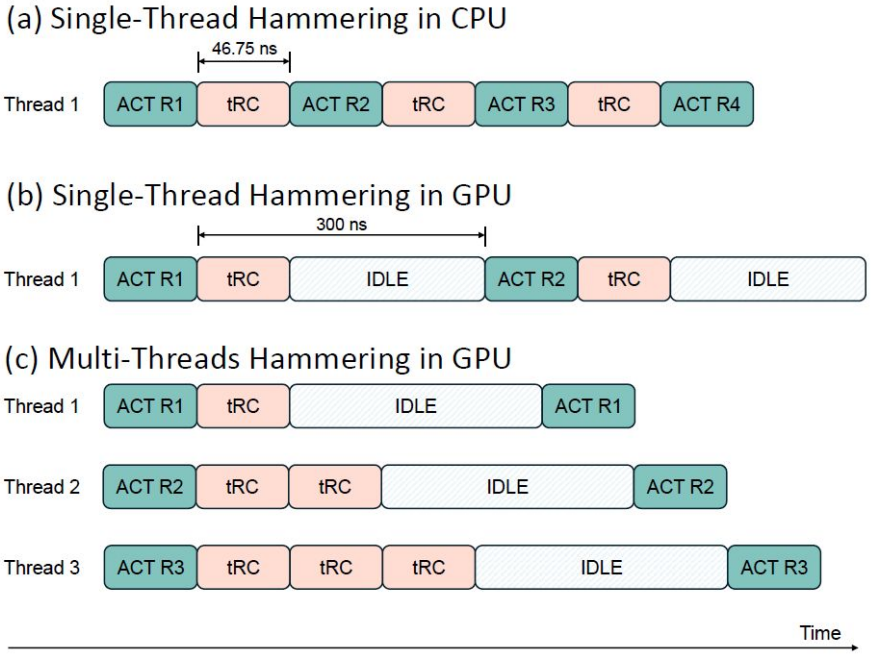
Why Parallel Rowhammering?

single thread:

- 100K ACTS within *tREFW*
- 5K ACTs per row

Multi thread:

- >500 K ACTS within *tREFW*



Challenges

- **Double-Sided Rowhammer.**
  - Which two rows are physically adjacent to the victim?
- **Synchronizing the Hammering Order.**
  - When the TRR occur within one refresh interval?
- **Rapid Profiling.**
  - How to trigger more flips in the limited time?

DRAM Geometry (Bank)

Group addresses with bank conflict timing side-channel

- 48 GB - 384 DRAM banks (RTX A6000)
- non-linear bank mapping - complex to reverse engineer

Solution

1. Profile the entire memory space.
2. Save all the virtual address offsets mapped to the bank.

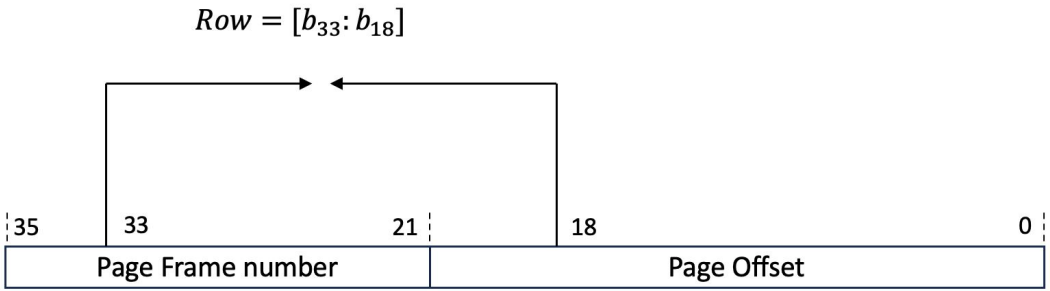
Starting physical address accessible to user-level applications is deterministic.

User-level applications have continuous physical memory space.

DRAM Geometry (Row)

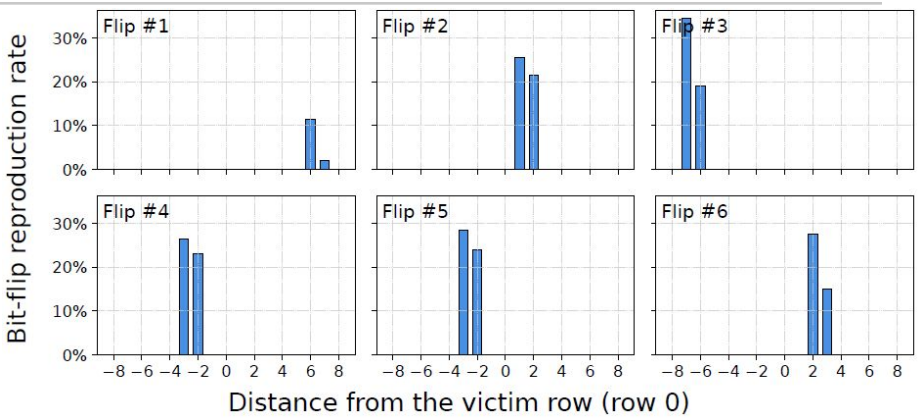
Assumption (linear layout)

- iterate at a cache line granularity
- higher consecutive address bits as row index (bits [33:18])



Locating True Aggressors

- Target a vulnerable cell.
- Test only one candidate aggressor at a time
- Fill all other 23 with far-away dummy rows
- Rows that reproduce the flip = **true aggressors**



Findings

1. Every cell flipped by exactly two aggressors
   -> physically adjacent
2. Non-monotonic row function

- Aggressors are adjacent in logical ID.
- Victim is distant from its aggressors.

Row remapping?

Synchronized Rowhammer

Fact: single-sided is *more* effective than double-sided hammer.

Why? More easy to be detected by TRR!

Naive parallel hammering: execution order of access is unpredictable.

How to evade TRR? Synchronize the hammering order!

Synchronized Rowhammer

Naive parallel hammering:

- execution order is unpredictable
- Warp jitter (~ 50 cycles)

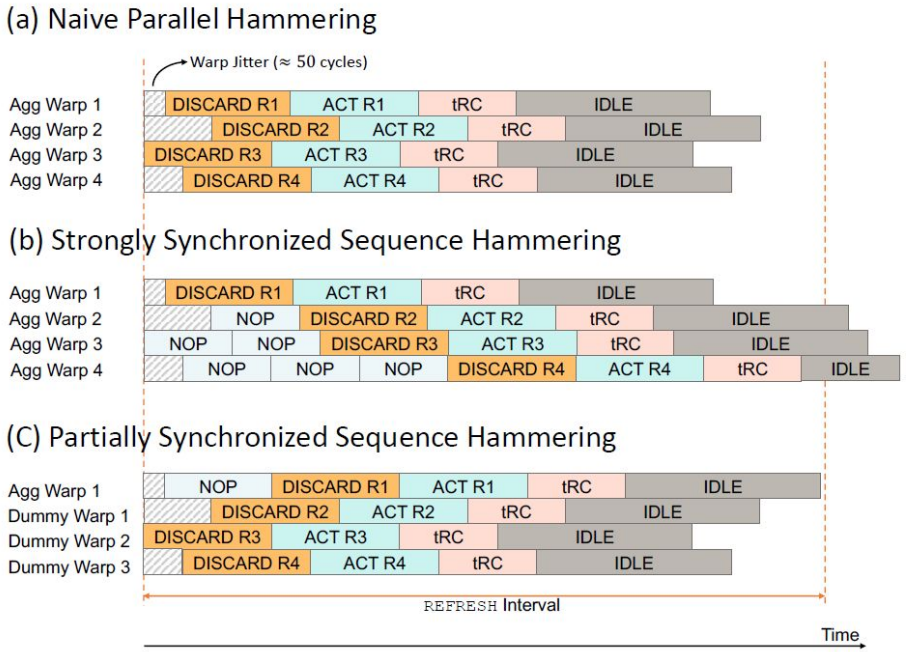
Solution: add *nops* to halt the warp!

Strongly synchronized hammering:

- Exceed one refresh interval
- Less activation throughput

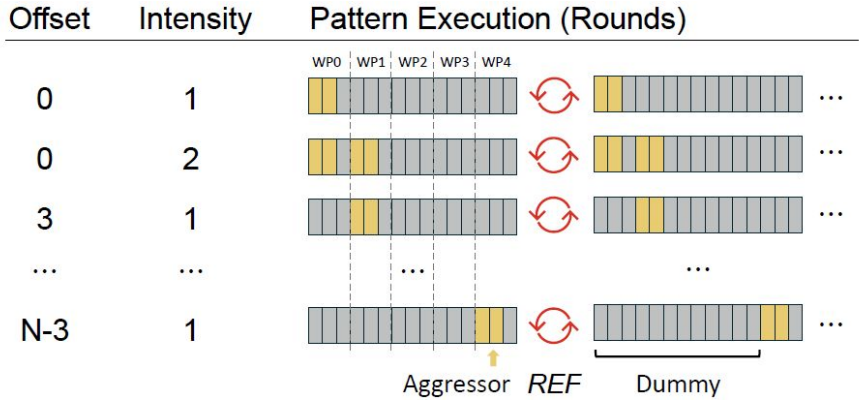
Partially synchronized hammering:

- Only synchronize the warp with *true aggressors*
- Maintain activation throughput

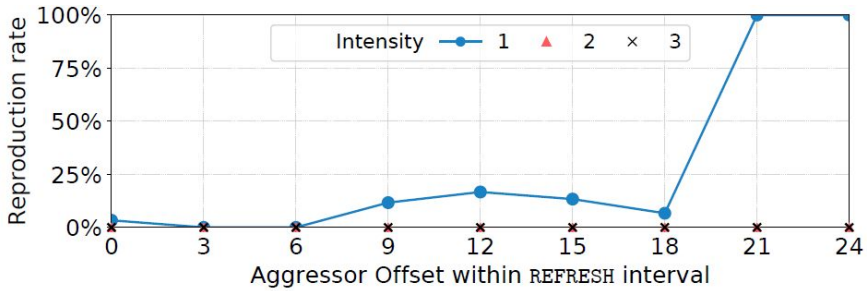


TRR Sampling Characterizations

- Place *true aggressors* in the different offsets within a interval.
- Try different activation intensity.
- Measure reproduction rate.



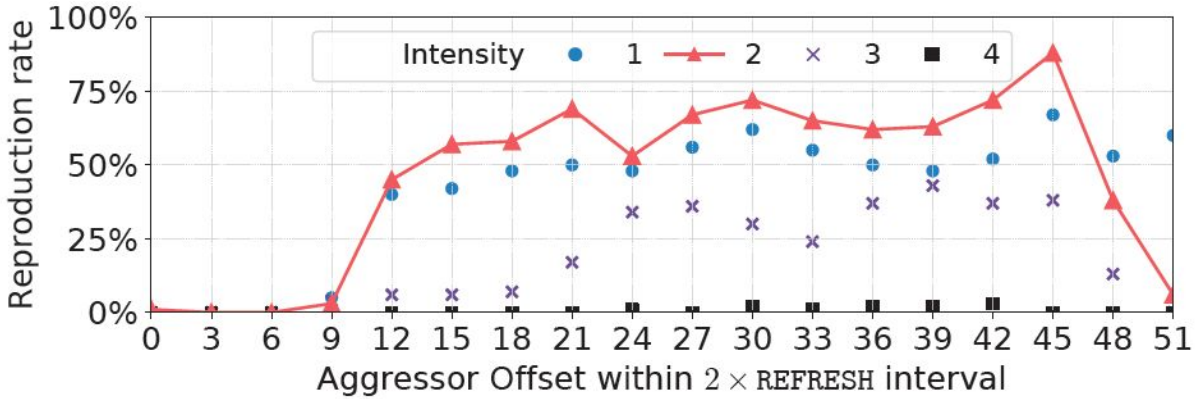
- TRR occurs in early stage within a tREFI.
- Bypass the TRR if the aggressor in the last ~20% within a tREFI



TRR Sampling Characterizations

Expand to multi-interval patterns

Optimal case:
- activate the aggressor only once
- keep the pattern within a single tREFI

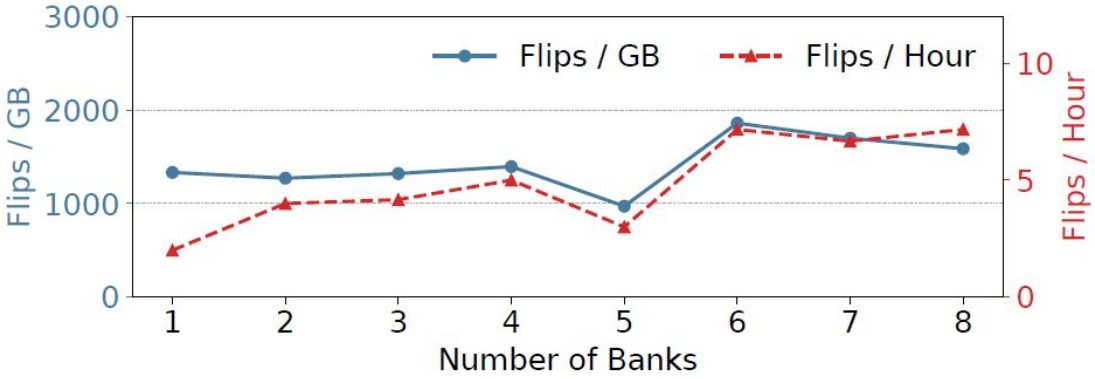


Multibank Rowhammer

Increase the rate of profiling the memory with parallelism.

GPU: one SM hammer one bank

Optimal case: 6-bank Hammering!



Vs GPUHammer in Our A6000

4-bank, double-sided, partially synchronized **vs** naive GPUHammer

	Ours	Lin et al. [3]
Total Flips	379	11
$0 \rightarrow 1$ Flips	347	11
$1 \rightarrow 0$ Flips	32	0
Flips/bank	94.75	2.75
Flips/GB	758	22

Lin, Chris S., Joyce Qu, and Gururaj Saileshwar. "{GPUHammer}: Rowhammer attacks on {GPU} memories are practical." *34th USENIX Security Symposium (USENIX Security 25)*. 2025.

Comprehensive Study

Rowhammer is Getting Worse

- 1580 CPU DRAM chip tested
- Newer chips are more vulnerable to Rowhammer

Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques

Jeremie S. Kim[§†] Minesh Patel[§] A. Giray Yağlıkçı[§]
Hasan Hassan[§] Roknoddin Azizi[§] Lois Orosa[§] Onur Mutlu[§†]
[§]*ETH Zürich* [†]*Carnegie Mellon University*

How About GPU?

Worst pattern:

24-sided, 6-banked, partially synchronized, double-sided hammer pattern for 6 hours

DRAM Vendors:

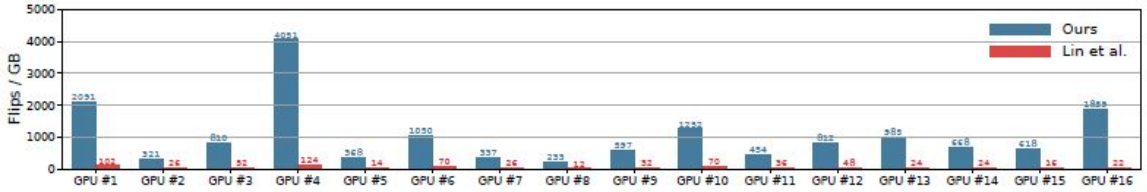
Samsung

GPU	DRAM Type	Vulnerability
Ampere RTX A6000	GDDR6	16/17
Ada RTX A6000	GDDR6x	0/8

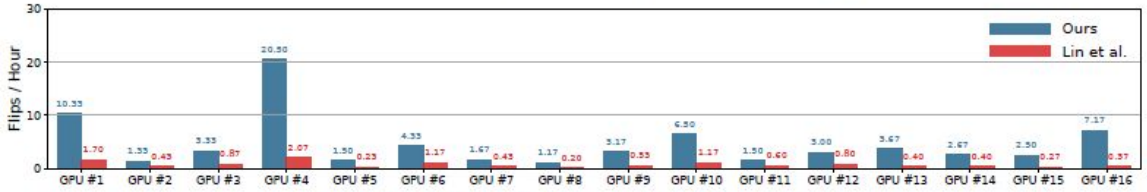
Flips/GB and Flips/Hour

Flips/GB: 1032 vs 44 (GPUHammer)

Flips/Hour: 4.65 vs 0.73 (GPUHammer)



(a) Flips/GB (total number of flips found divided by the hammered memory size).

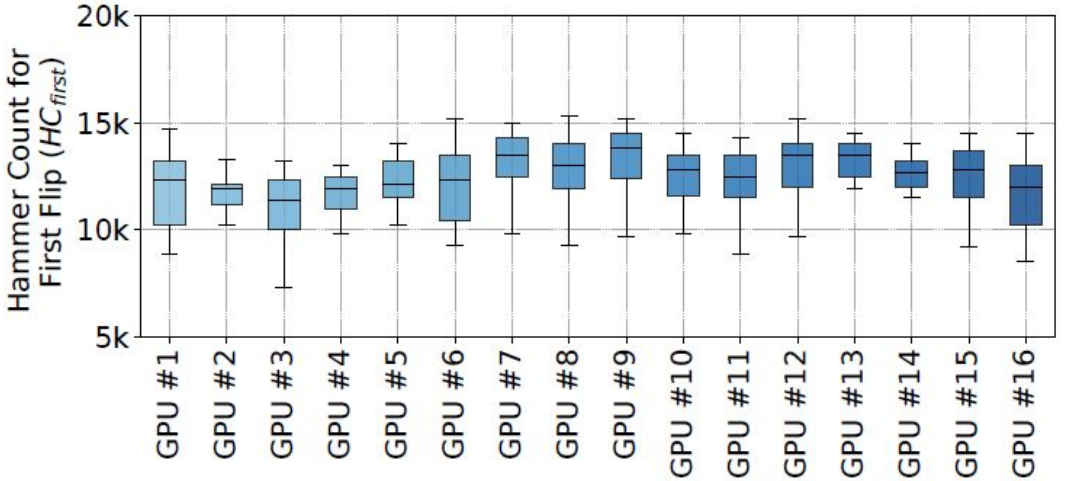


(b) Flips/hour (total number of flips found divided by hammering time).

Hammer Count (*HC*) Threshold

median number: 11*K*

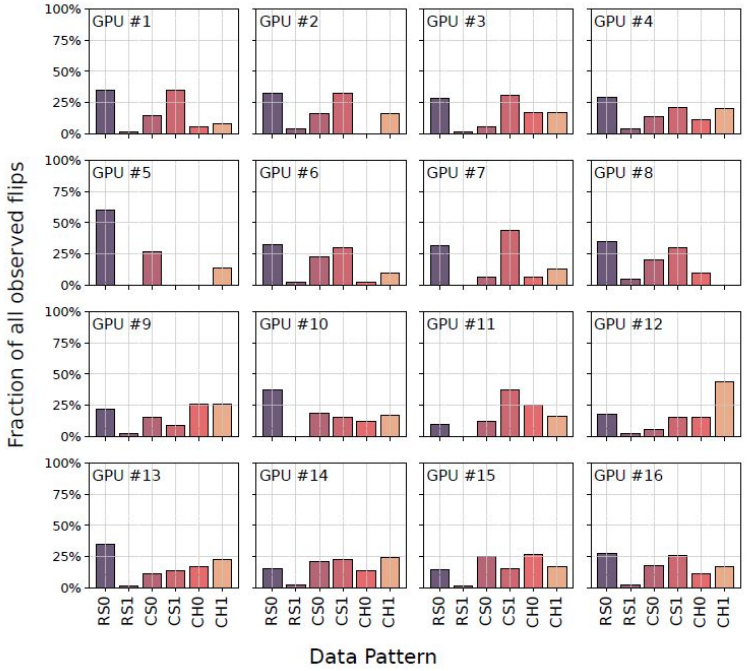
≈ most vulnerable DDR4 DRAM



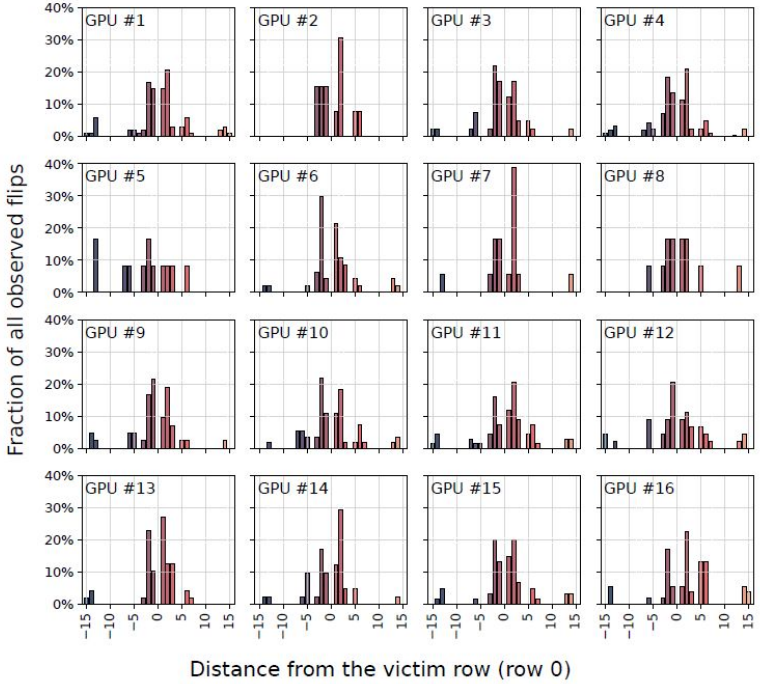
Data Pattern

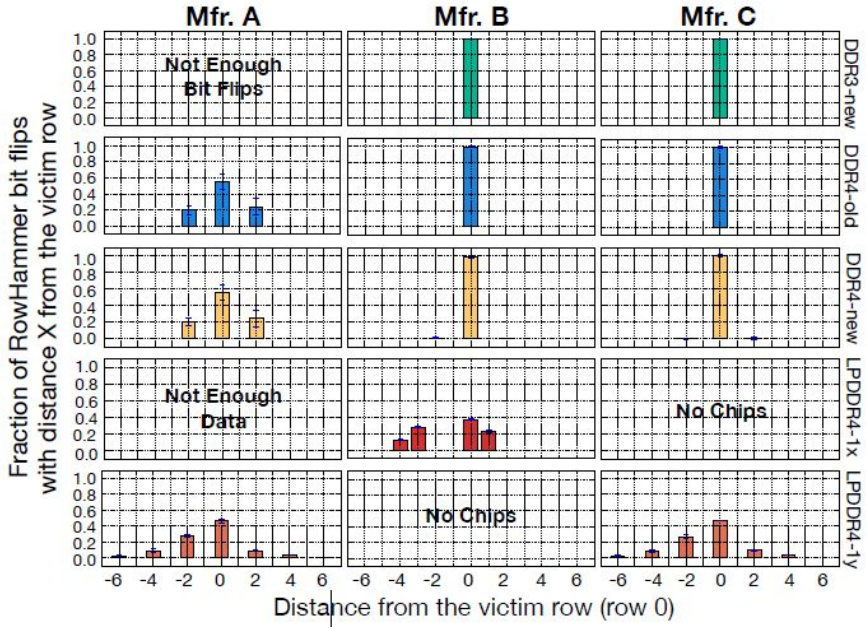
RS0:	0x00; 0xff
RS1:	0xff; 0x00
CS0:	0x55; 0x55
CS1:	0xaa; 0xaa
CH0;	0x55; 0xaa
CH1:	0xaa; 0x55

Best pattern: RS0



Logical Row Distance





Hammer Attempts in Ada RTX A6000 (GDDR6x Memory)

Activations per tREFI:

- 1925 ns
- ~36 memory accesses

1. different aggressor position within tREFI
2. different aggressor distance

No flips

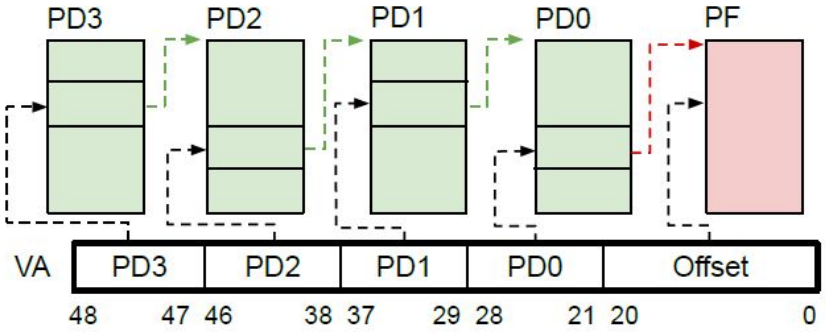
Hijacking CPU Memory via GPU

Walking Page Table Hierarchy

- 2MB page size
- 49-bit virtual address bits

translate virtual to physical address

nvdebug

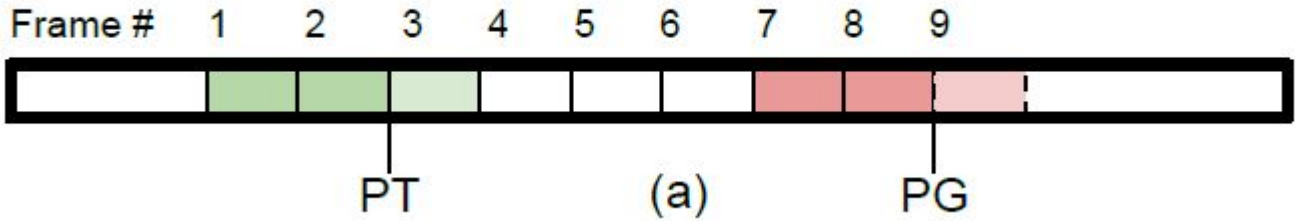


J. Bakita and J. H. Anderson, “Demystifying NVIDIA GPU internals to enable reliable GPU management,” in Proceedings of the 30th IEEE Real-Time and Embedded Technology and Applications Symposium, ser. RTAS, May 2024, pp. 294–305.

Memory Massaging

Goal: attack-controlled aggressor rows are placed next to victim.

Challenge: GPU allocator normally keep page tables (victim) and user-controlled pages (attacker) physically separate.

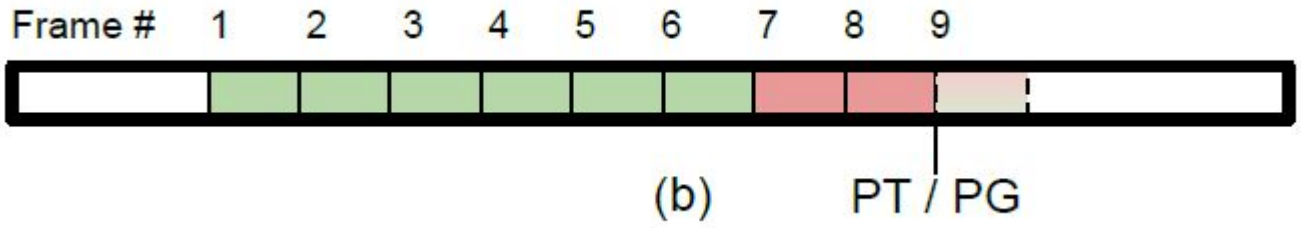


Page Table Overflow

cuMemMap()

Map many virtual addresses to the same physical frame

-> create many page entries; user-controlled space low

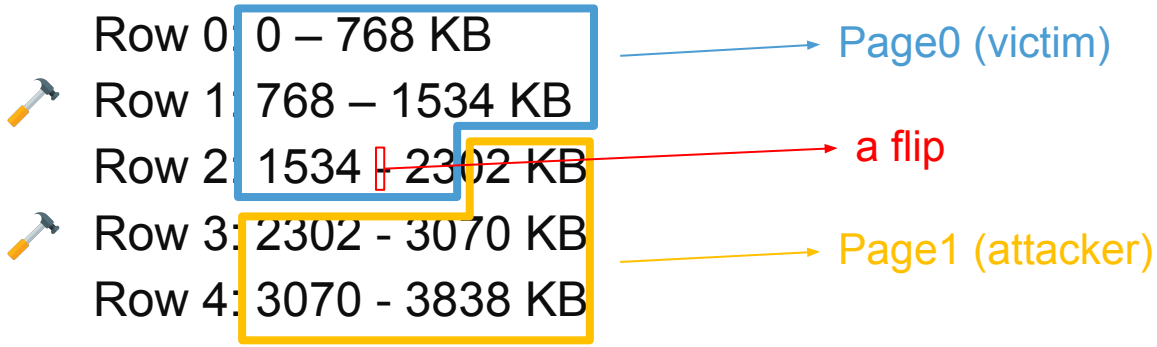


Double-Sided Hammer

page size: 2MB; DRAM row size: 2KB; DRAM bank number: 384

monotonic row mapping:

impossible that victim page reside between 2 aggressor rows

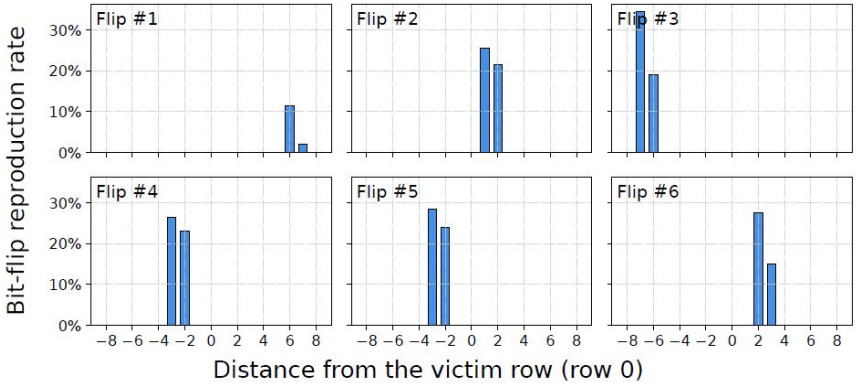


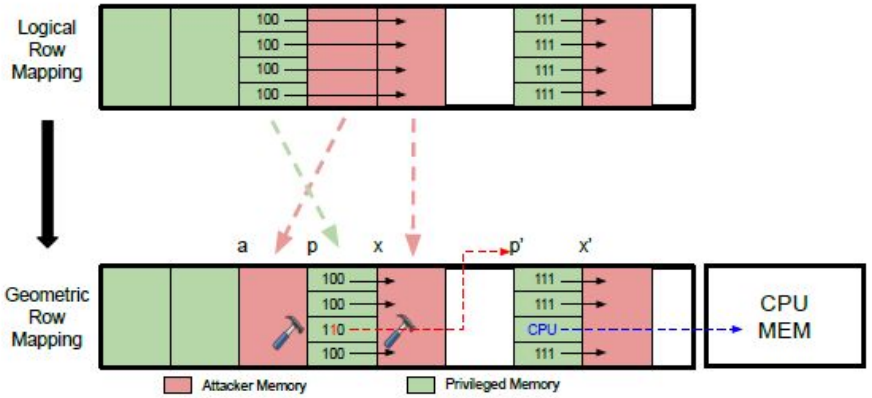
Double-Sided Hammer

non-monotonic

logical: A1 - A2 - V

geometric: A1 - V - A2





Exploitable Flips

Constraints #1: GPU PTE Structure

bit[57:37]: unused for GPU

bit[20:12]: below 2MB boundary

We must flip an orange bit!

16 bits / 16 bytes = 12.5%

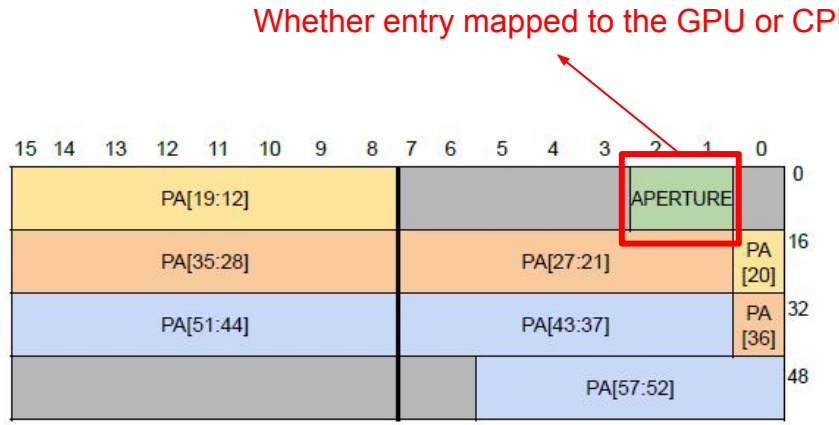


Figure 17: Relevant bits in each 2 MB page table entry for our attack. Yellow bits are below the 2 MB boundary and blue bits are unused for GPU memory, so we must flip an orange bit. The green bits control what device the address maps to (CPU or GPU).

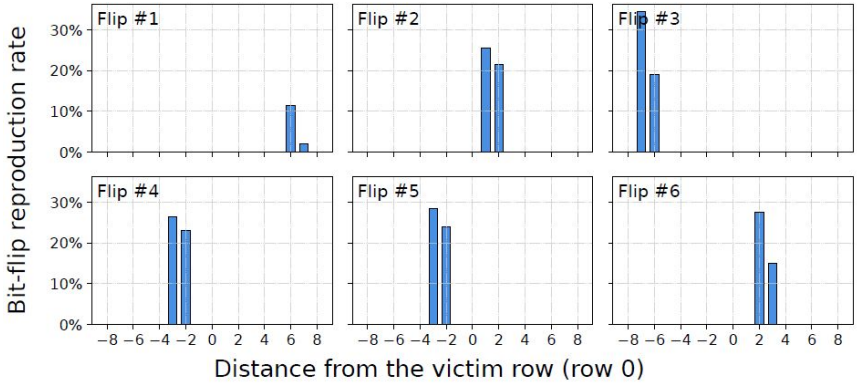
Exploitable Flips

Constraints #2: Page Boundary (2MB) vs Row Stride (768KB)

Worst case: aggressors are *close* to victim

768KB / 2MB = 37.5%

37.5% * 12.5% = 4.69%



Results: **9.2%** of observed flips are useful

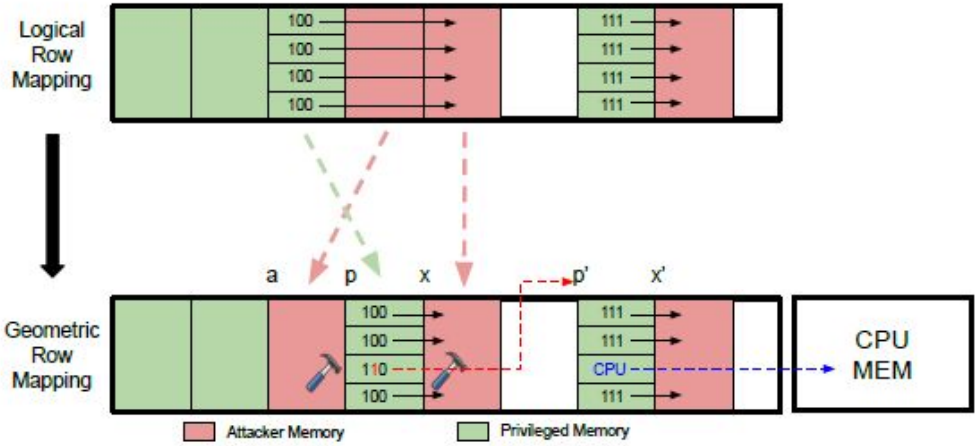
Page Table Overwrite

- Find a exploitable flip
- Page table overflow p -> x
- hammer()
- scan virtual memory p -> p’

read/write APERTURE field

v -> any CPU/GPU memory

average attack time: 63.2s



Discussion

Mitigations

ECC

ECC is disabled by default in GDDR GPU:

- reserved memory (6.25% overhead)
- bandwidth loss up to 12%

On-die ECC in HBM GPU (A100, H100)

Mitigations

Advanced Hardware Mitigations

Refresh Management (**RFM**), defined in

- *Rolling Accumulated ACT* (**RAA**) cou
- RAA reaches the threshold -> block all bank

Evidence shows RFM is **not** implemented by the vendor in DDR5

McSee: Evaluating Advanced Rowhammer Attacks and Defenses via Automated DRAM Traffic Analysis

Patrick Jattke *ETH Zurich* · Michele Marazzi *ETH Zurich* · Flavien Solt *UC Berkeley* · Max Wipfli *ETH Zurich* · Stefan Gloor *ETH Zurich* · Kaveh Razavi *ETH Zurich*

Phoenix: Rowhammer Attacks on DDR5 with Self-Correcting Synchronization

Diego Meyer† *ETH Zurich* · Patrick Jattke† *ETH Zurich* · Michele Marazzi *ETH Zurich* · Salman Qazi *Google* · Daniel Moghimi *Google* · Kaveh Razavi *ETH Zurich*

JEDEC, Graphics Double Data Rate (GDDR6) SGRAM Standard, Document JESD250D

JEDEC, Low Power Double Data Rate 5 (LPDDR5) Standard, Document JESD209-5C.E.

Mitigations

Memory Isolation

Driver and I/O MMU level

- Isolate page tables and user-controlled pages

Conclusion

- Novel Rowhammer techniques to trigger far more flips.
- Comprehensive study of a large number of GPUs.
- First GPU-to-CPU exploit to gain arbitrary read/write.

Concurrent Works

GeForge: Hammering GDDR Memory to Forge GPU Page Tables for Fun and Profit

Junpeng Wan[1,2], Yanan Guo[3], Zhi Zhang[4], Zhuo Li[5], Dave (Jing) Tian[2], Zhenkai Zhang[1]

[1]*Clemson University,* [2]*Purdue University,* [3]*University of Rochester,*
[4]*University of Western Australia,* [5]*HydroX AI*

Concurrent Works

GDDRHammer

- Novel Rowhammer Techniques - *highest* Flips/GB
- GPU/CPU arbitrary read/write

GeForge

- First Rowhammer in RTX 3060 (>1000 flips in 72 banks)
- GPU/CPU arbitrary read/write

GPUBreach

- no novel rowhammer techniques
- GPU/CPU arbitrary read/write with IOMMU enabled

Future Directions

Rowhammer WebGPU applications

- high-level language
- eviction set construction

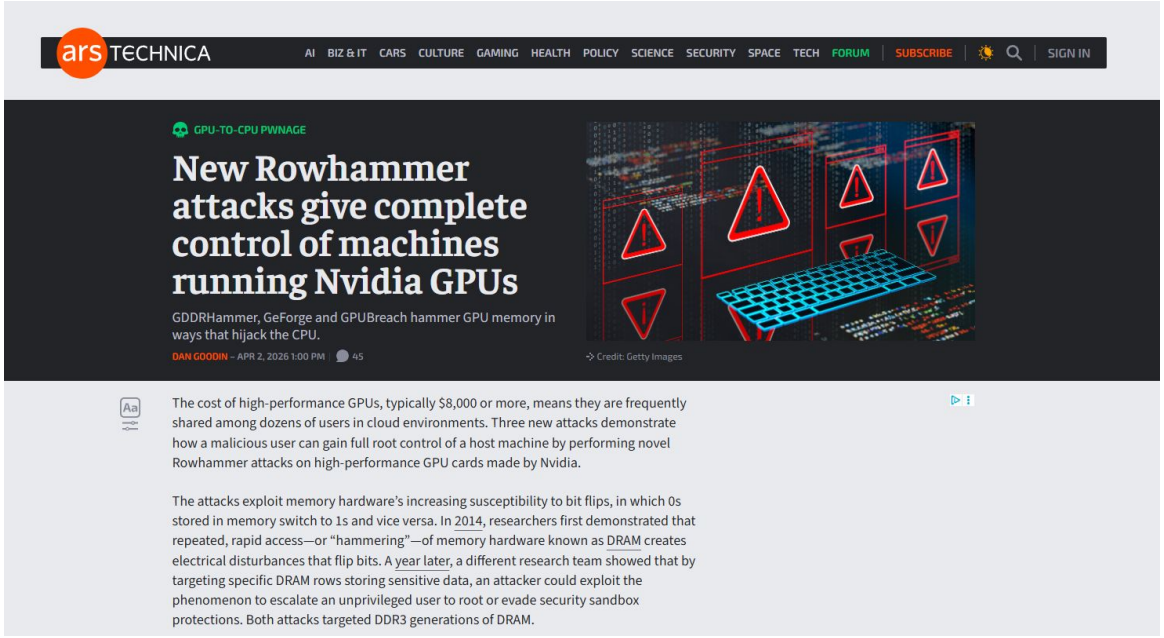
Rowhammer GPU with ECC enabled

- ECC only protects limited error patterns
- Rowhammer can induce crafted multi-bit faults

Reverse engineer the complex non-linear DRAM function

We’re being featured on mainstream tech news!

https://gddr.fail/



THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL