ISCA 2024

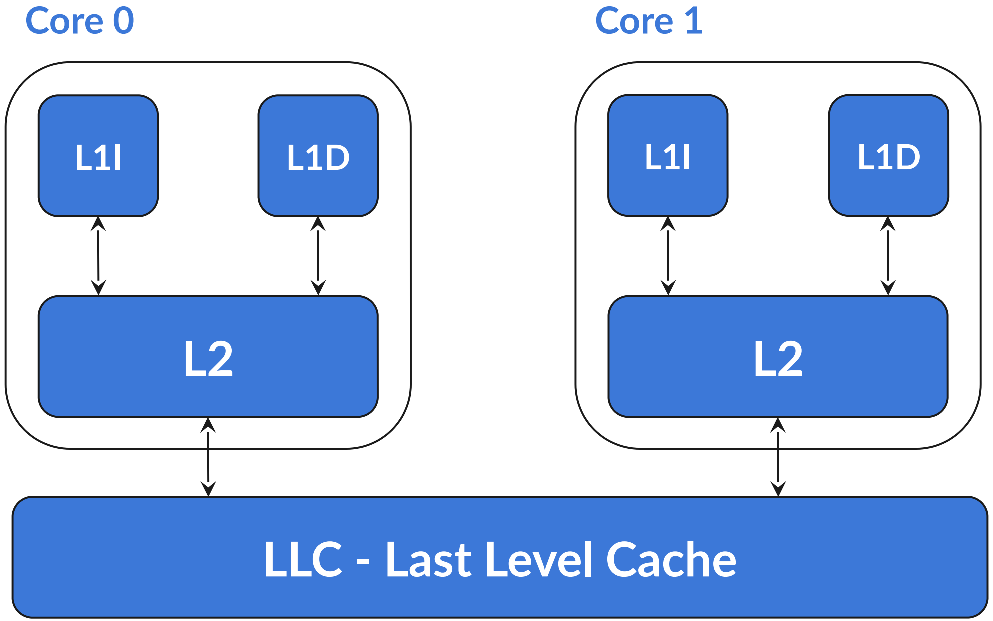
The Maya Cache

A Storage-efficient and Secure Fully-associative Last-level Cache

Anubhav Bhatla, Navneet, Biswabandan Panda

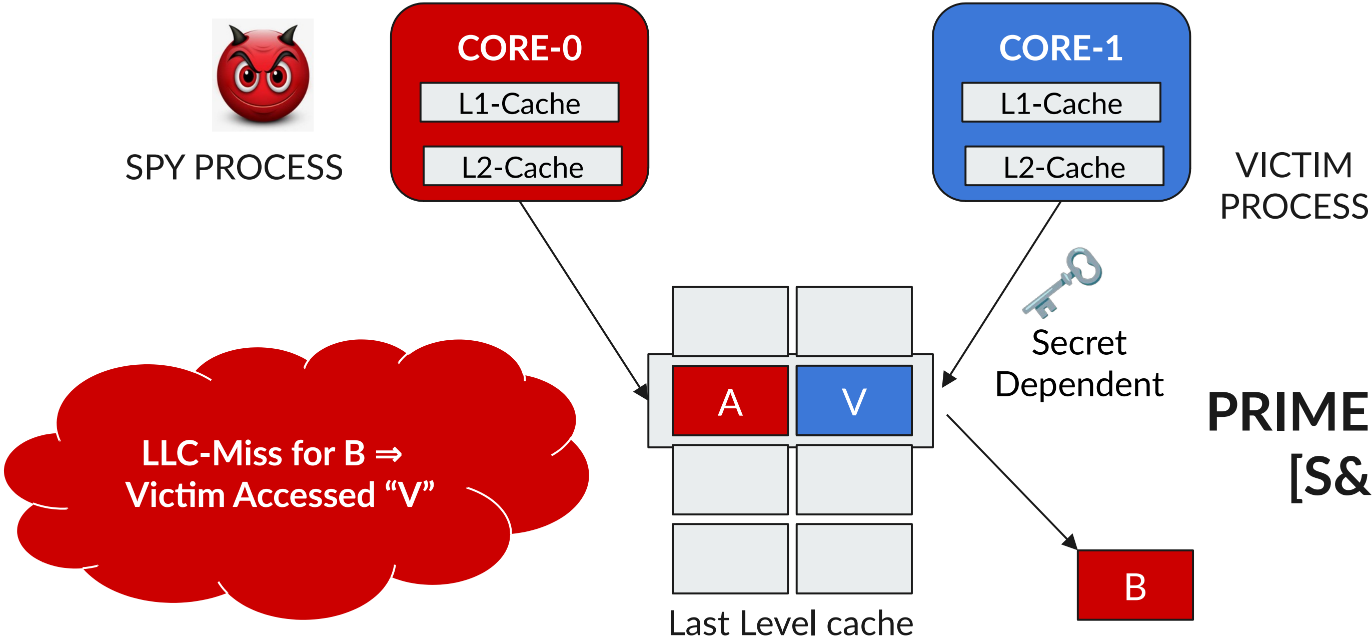
CASPER@Indian Institute of Technology Bombay

Background



Cache hierarchy in modern processors

Conflict-based Attacks

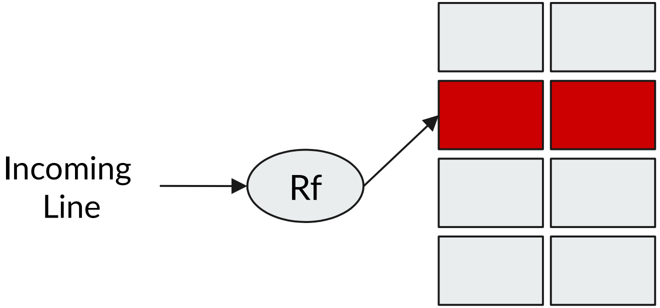


PRIME+PROBE [S&P 15]

A cache miss results a timing difference in access due to high DRAM latency

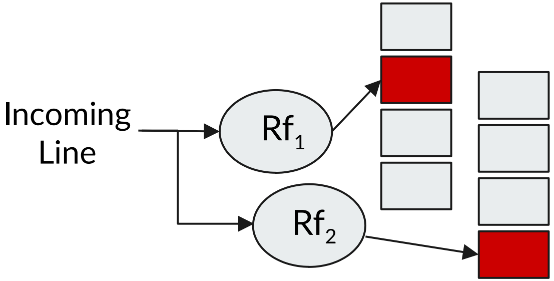
How to defend?

Randomized mapping



CEASER [MICRO '18]

Skews



CEASER-S [ISCA '19]
SCATTER CACHE [S&P '19]

2018

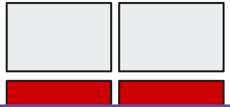
2019
Eviction set discovery
in O(N) accesses
[ISCA'19], [S&P'19]

2019

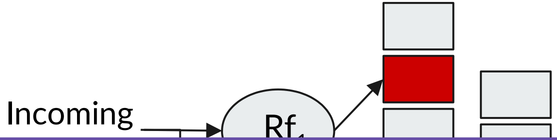
2021
Broken using
probabilistic evictions
[S&P'21]

How to defend?

Randomized mapping



Skews



SAEs make eviction set discovery possible

CEASER [MICRO '18]

CEASER-S [ISCA '19]
SCATTER CACHE [S&P '19]

2018

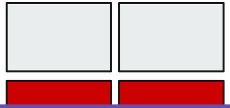
2019
Eviction set discovery
in O(N) accesses
[ISCA'19], [S&P'19]

2019

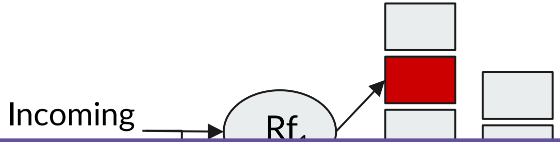
2021
Broken using
probabilistic evictions
[S&P'21]

How to defend?

Randomized mapping



Skews



SAE covery possible

So, what’s the FIX ??

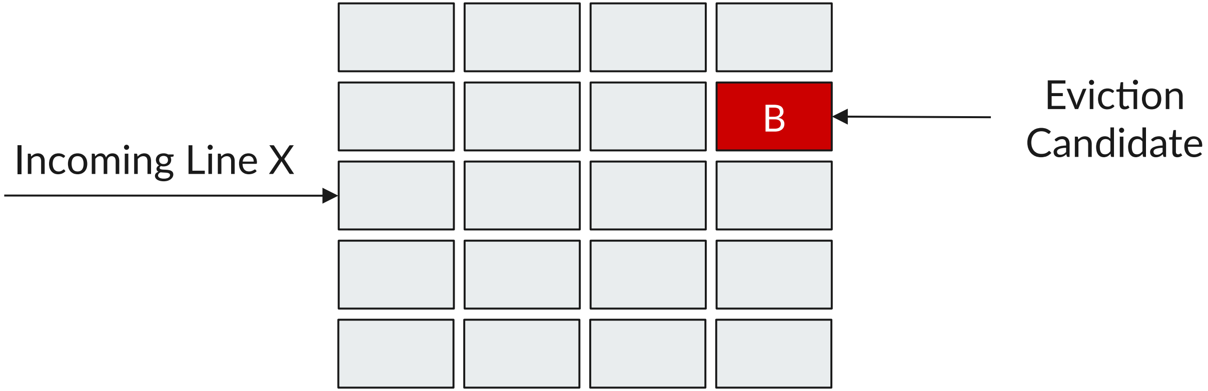
ASER-S [ISCA ‘19]

ER CACHE [S&P ‘19]

discovery
accesses
[ISCA’19], [S&P’19]

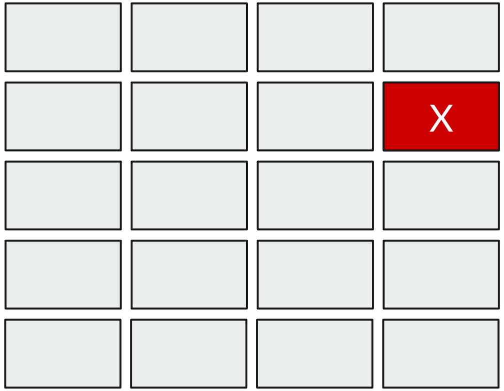
Broken using probabilistic evictions [S&P’21]

Fully Associative Randomized Cache



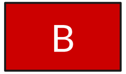
Fully associative randomized LLC with random global eviction

Fully Associative Randomized Cache



Fully associative randomized LLC with random global eviction

Cannot infer anything about X from random eviction of B

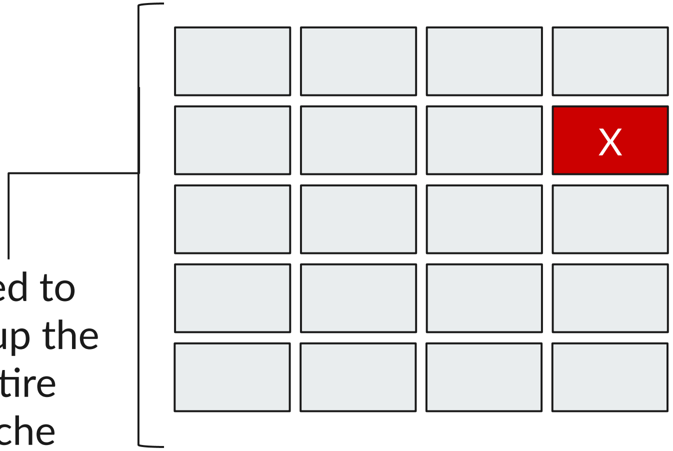


Evicted Block

No set-associative evictions; makes it harder for conflict-based attacks

Fully Associative Randomized Cache

Large lookup increases power consumption and latency

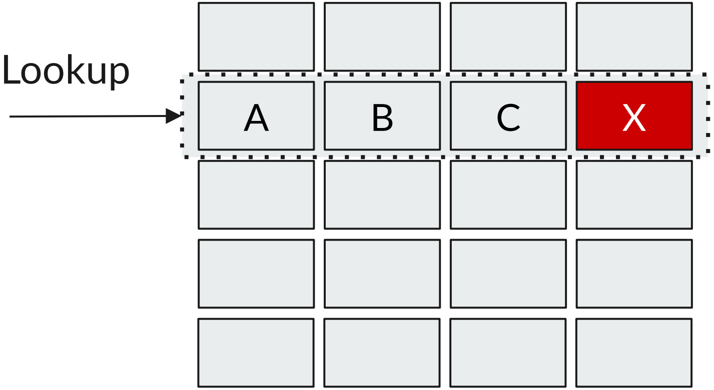


Need to lookup the entire cache

Fully associative Randomized LLC

Smaller lookup improves latency

Smaller Lookup



Set Associative LLC

Fully associative caches improve security but at the cost of power and latency

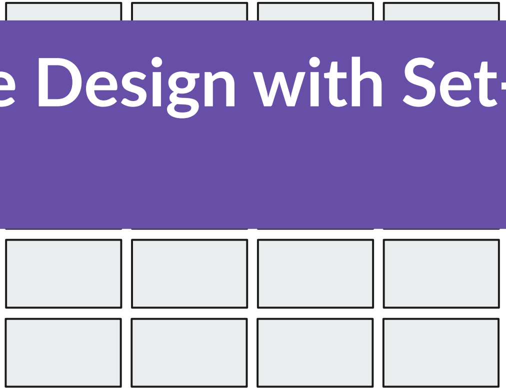
Fully Associative Randomized Cache

Large lookup increases power consumption and latency

Need to lookup the entire cache

Fully associative Randomized LLC

Smaller lookup improves latency



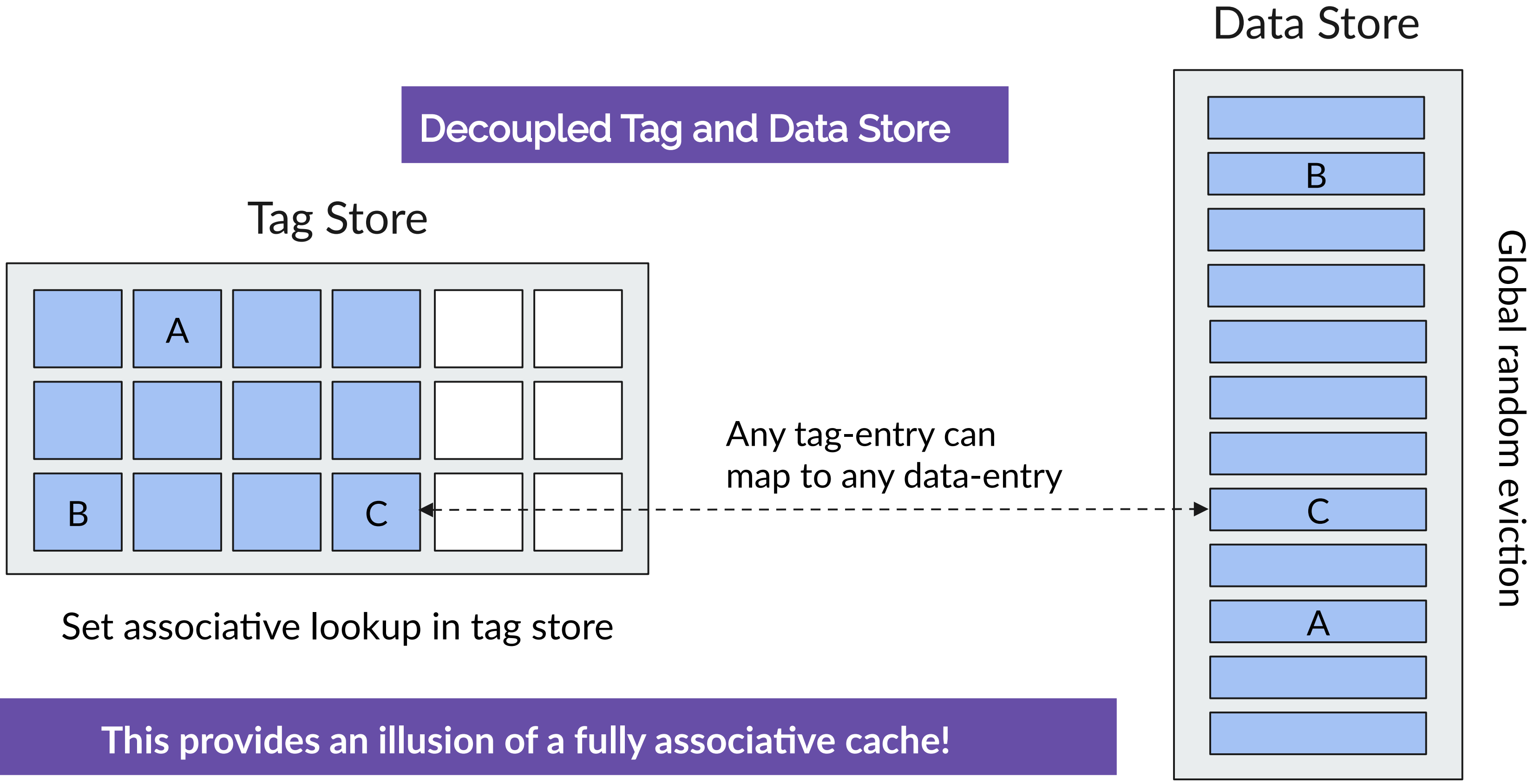
Set Associative LLC

How to get Security of Fully-Associative Design with Set-Associative Lookups?

Fully associative caches improve security but at the cost of power and latency

MIRAGE [USENIX SECURITY '20]

Decoupled Tag and Data Store

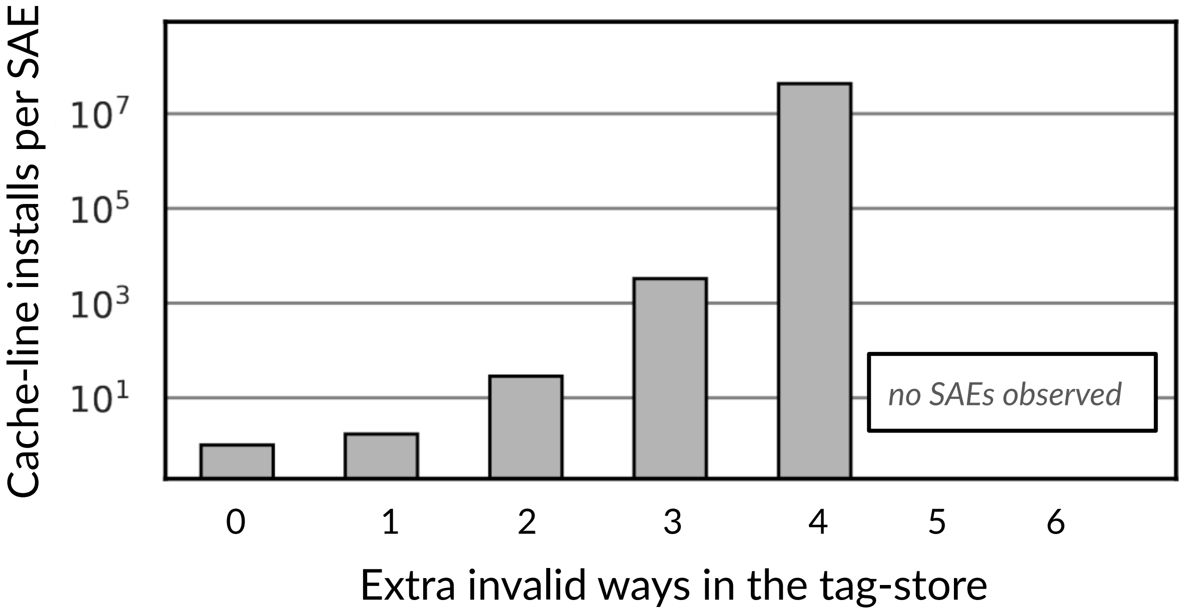


Any tag-entry can map to any data-entry

Set associative lookup in tag store

This provides an illusion of a fully associative cache!

MIRAGE Security



Security can be achieved by using extra invalid tag-ways

MIRAGE Tradeoffs

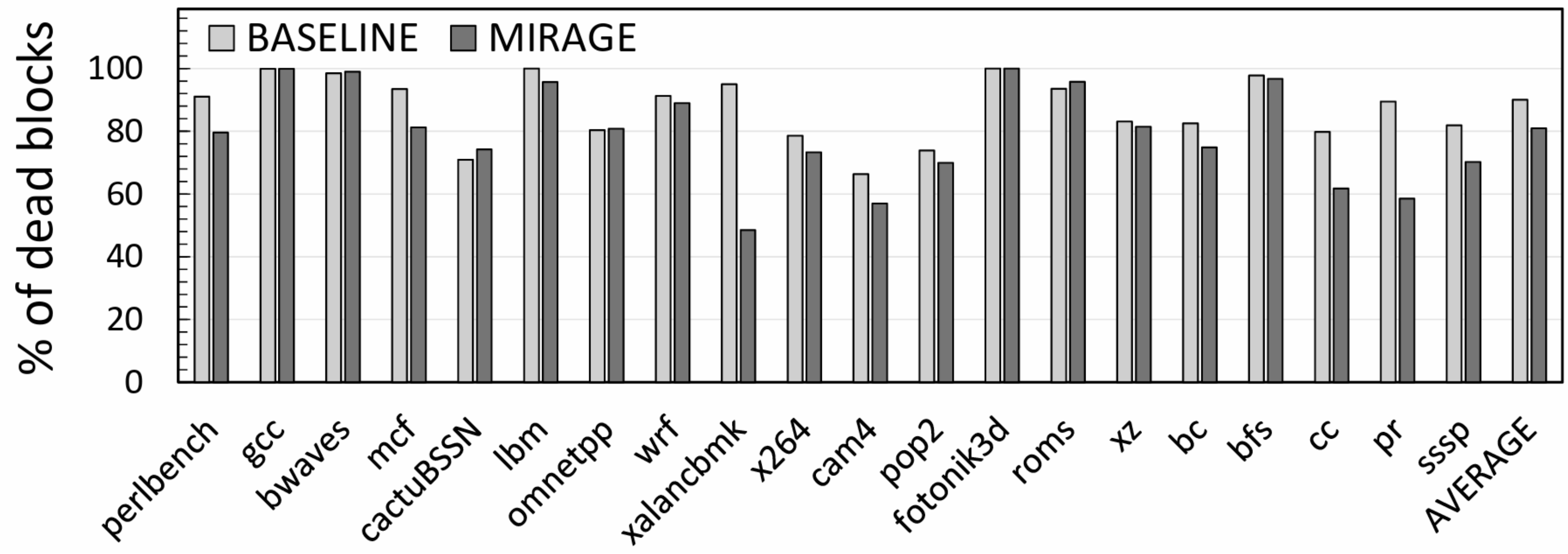
(+) Provides complete security against conflict-based attacks

(+) Performance comparable to a non-secure baseline

(-) High storage and area overheads (>20%)

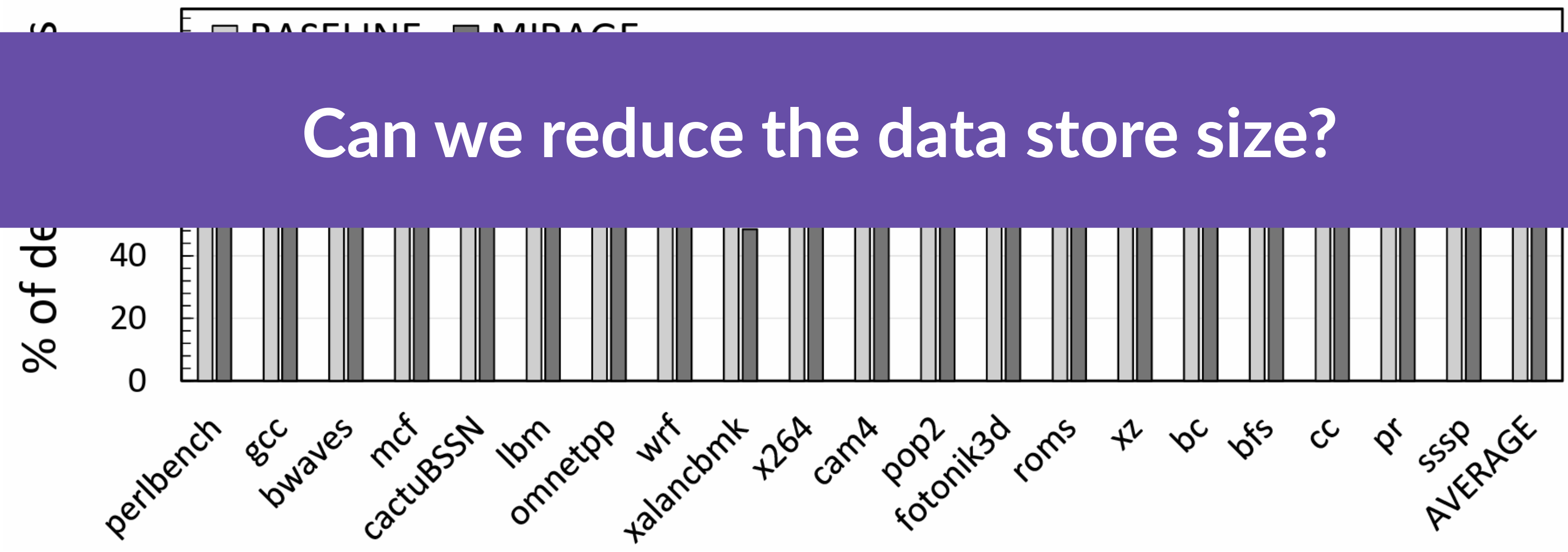
(-) High power overheads (19%)

Motivation



>80% dead blocks in the LLC

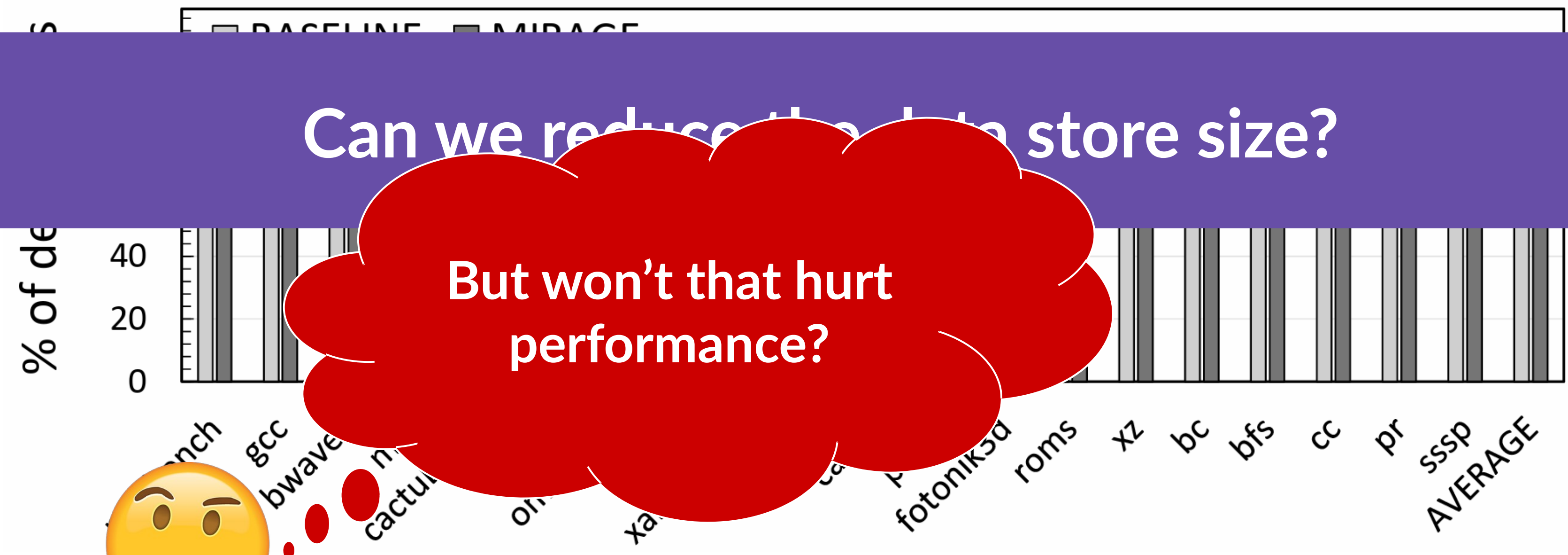
Motivation



Can we reduce the data store size?

>80% dead blocks in the LLC

Motivation



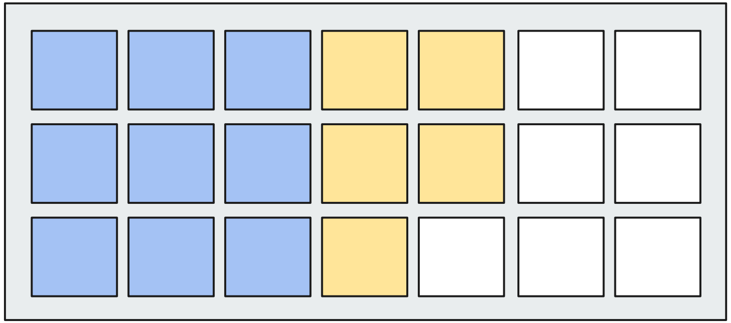
Can we reduce the data store size?

But won’t that hurt performance?

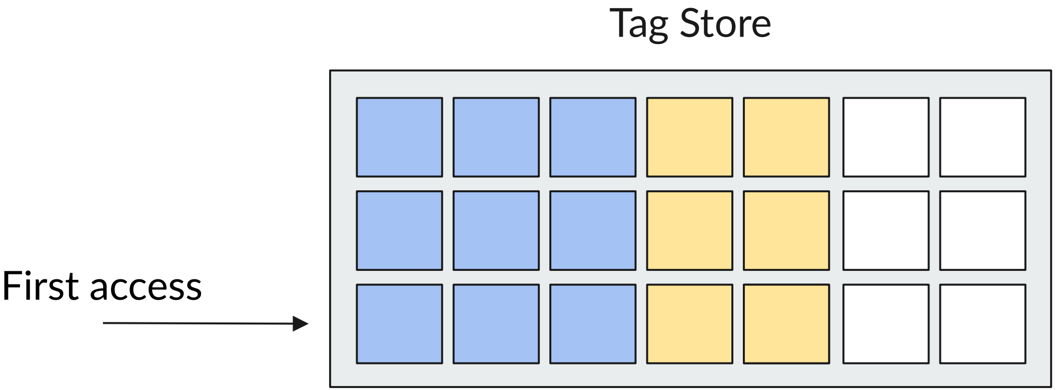
>80% dead blocks in the LLC

Tracking Reuse [MICRO ‘13]

Tag Store

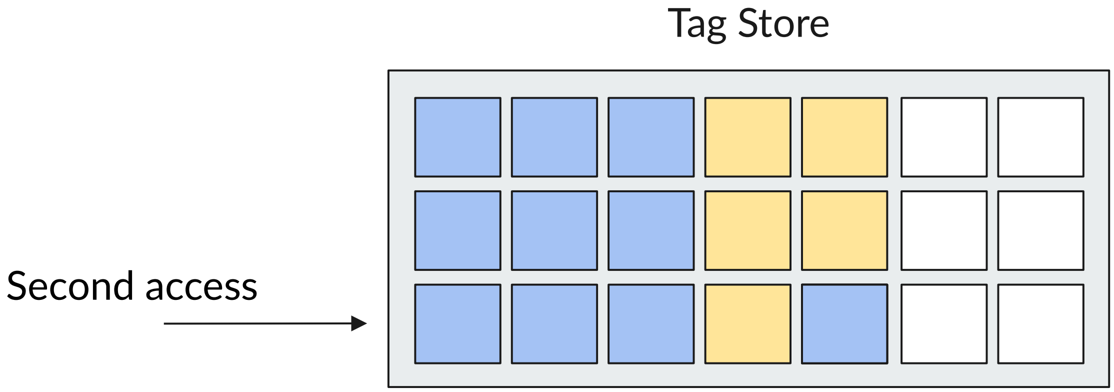


Tracking Reuse [MICRO '13]



Tag Miss ⇒ Only tag is inserted

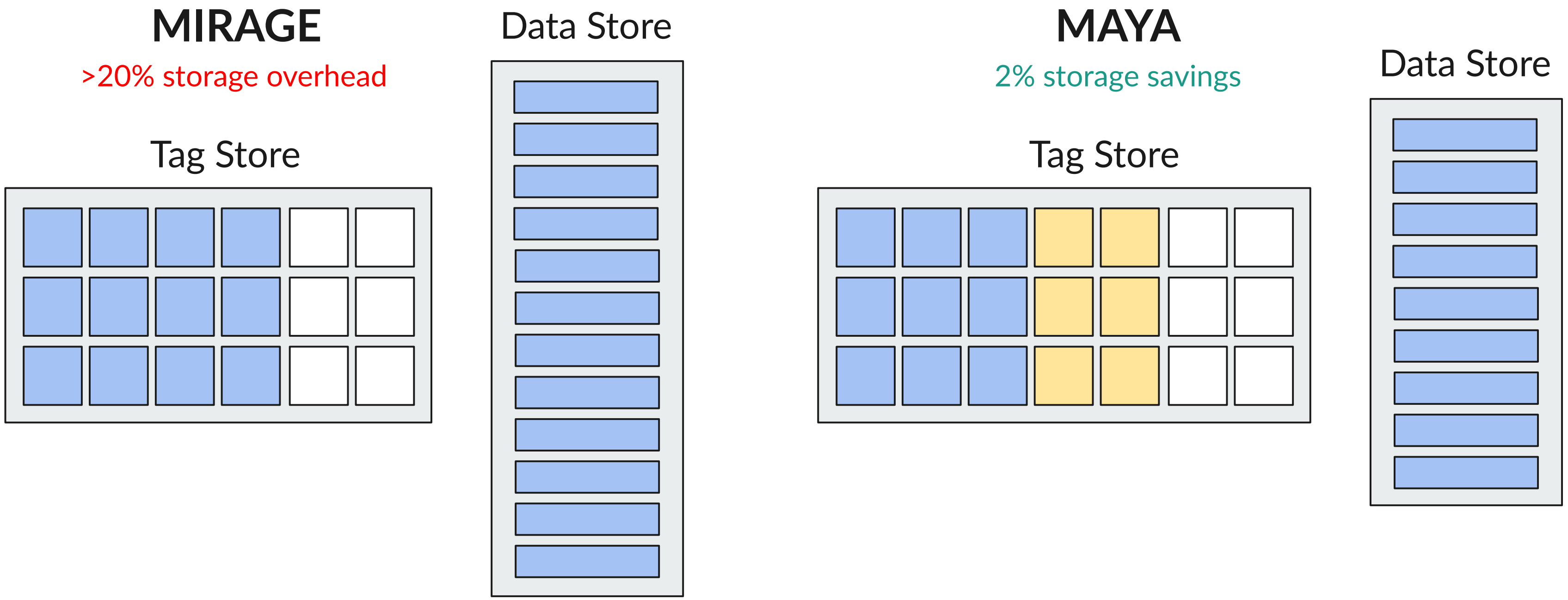
Tracking Reuse [MICRO '13]



Tag Hit, Data Miss ⇒ Data is brought in

Data is stored only when reuse is detected

Smaller data store and Reuse



MIRAGE

>20% storage overhead

Tag Store

Data Store

MAYA

2% storage savings

Tag Store

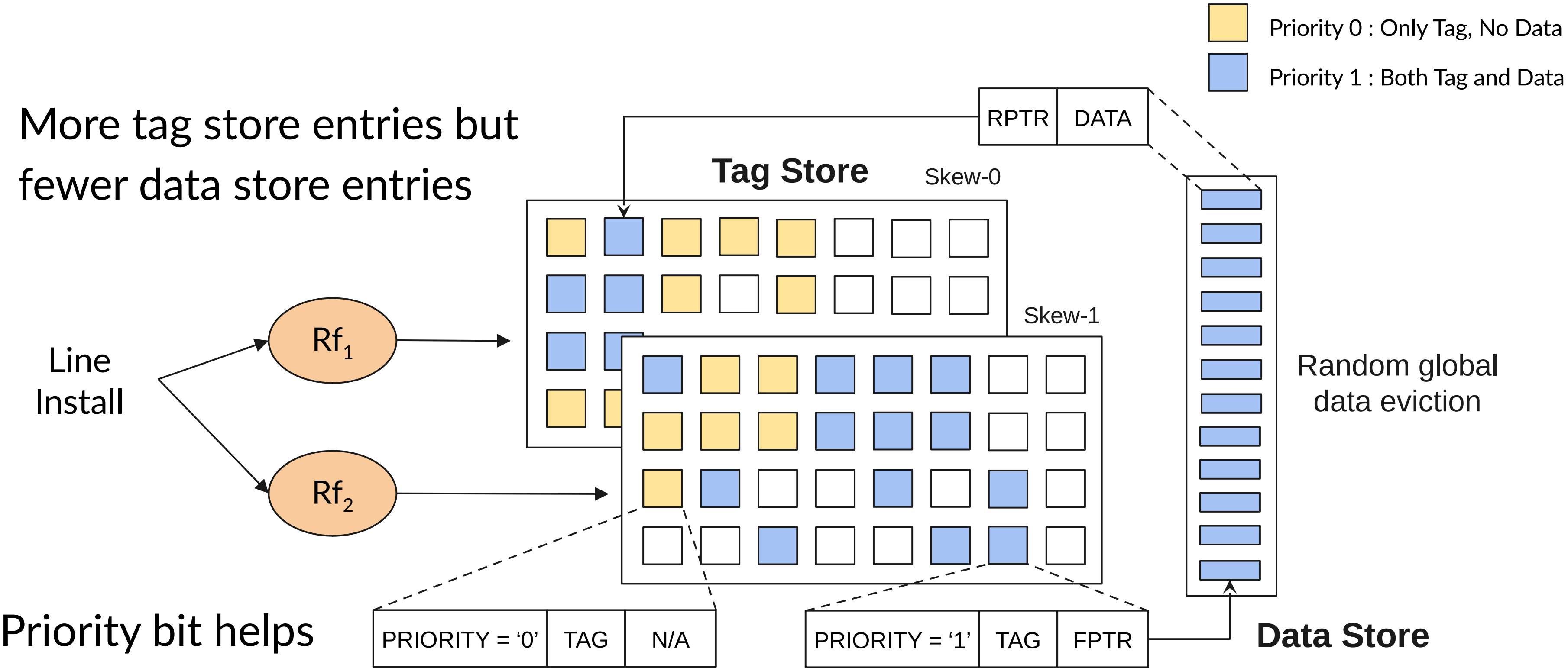
Data Store

Data entry requires 8X the number of bits compared to a tag entry

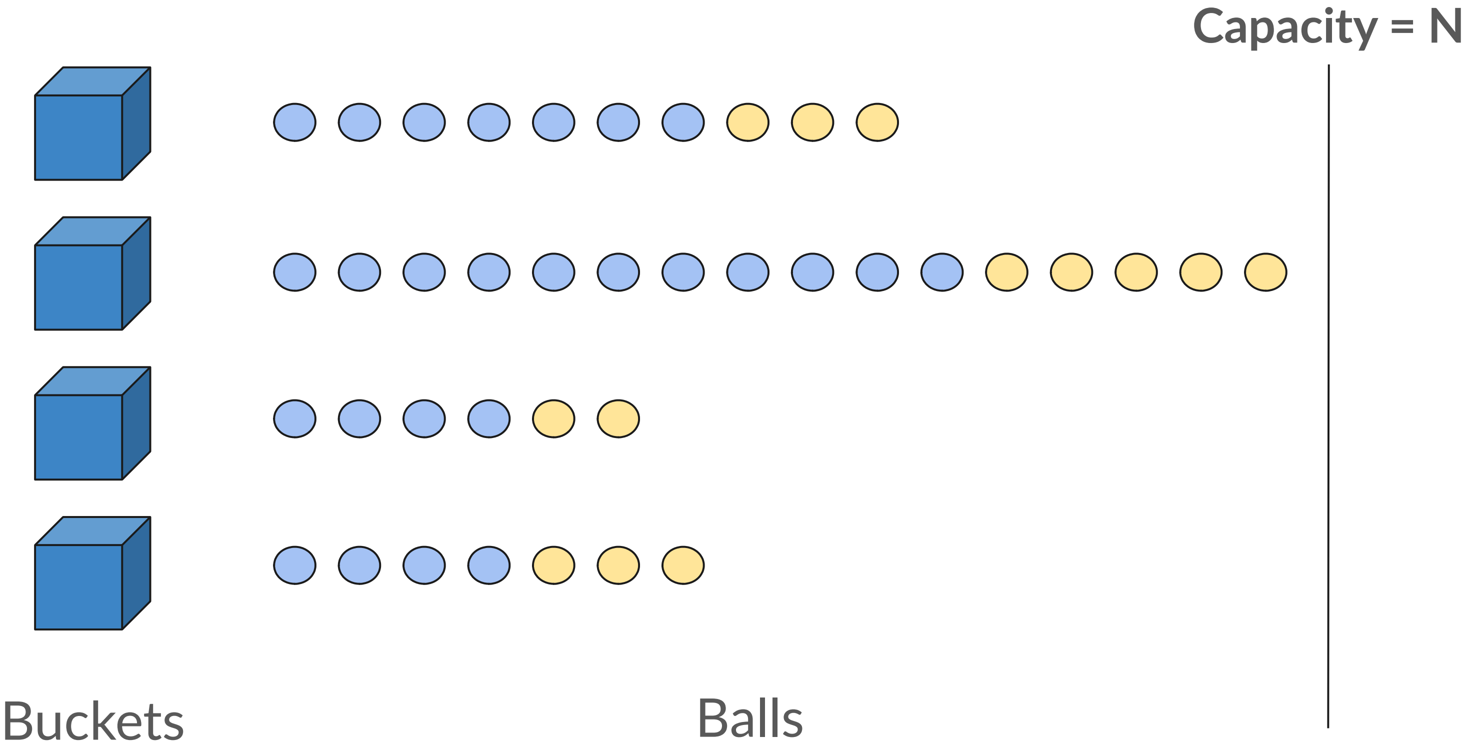
Maya Cache Design

More tag store entries but fewer data store entries

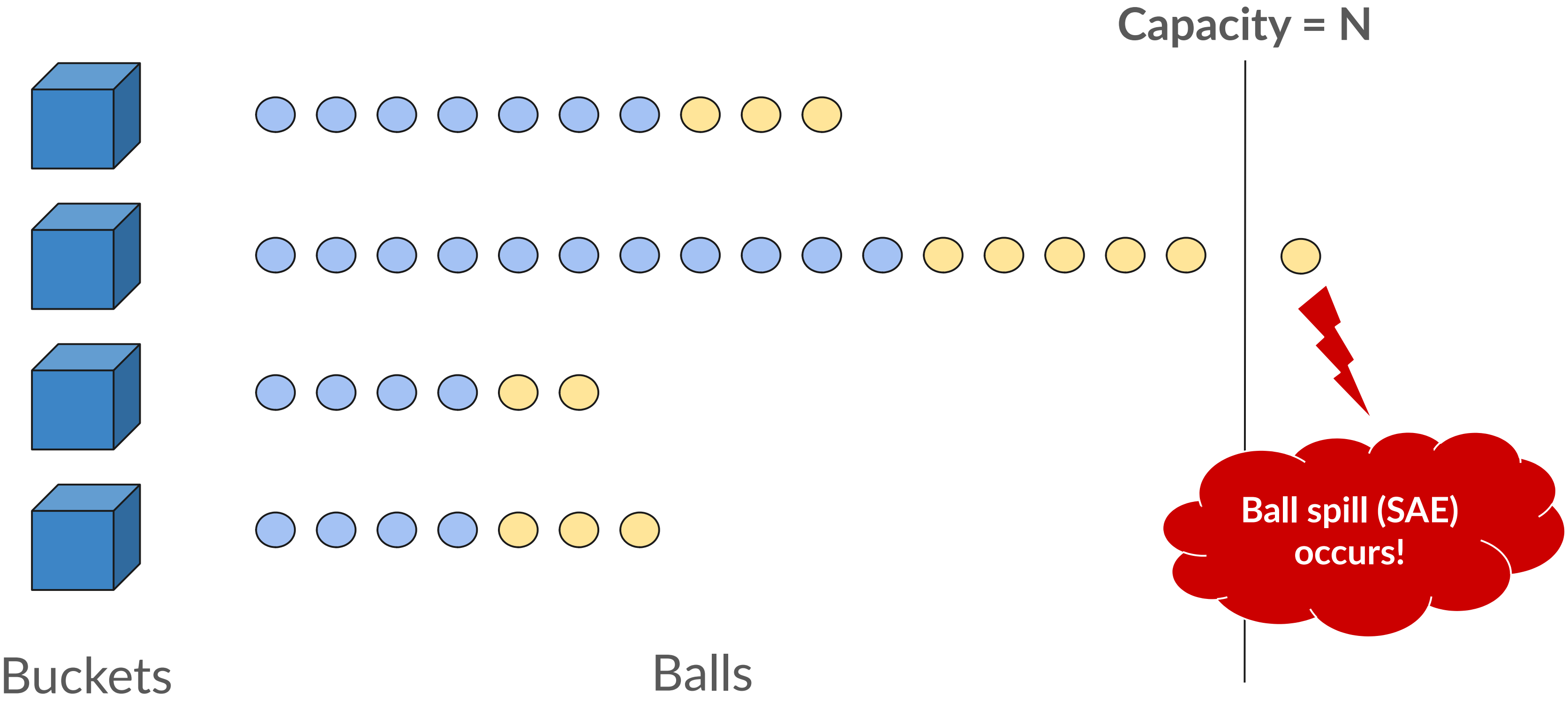
Priority bit helps track reuse



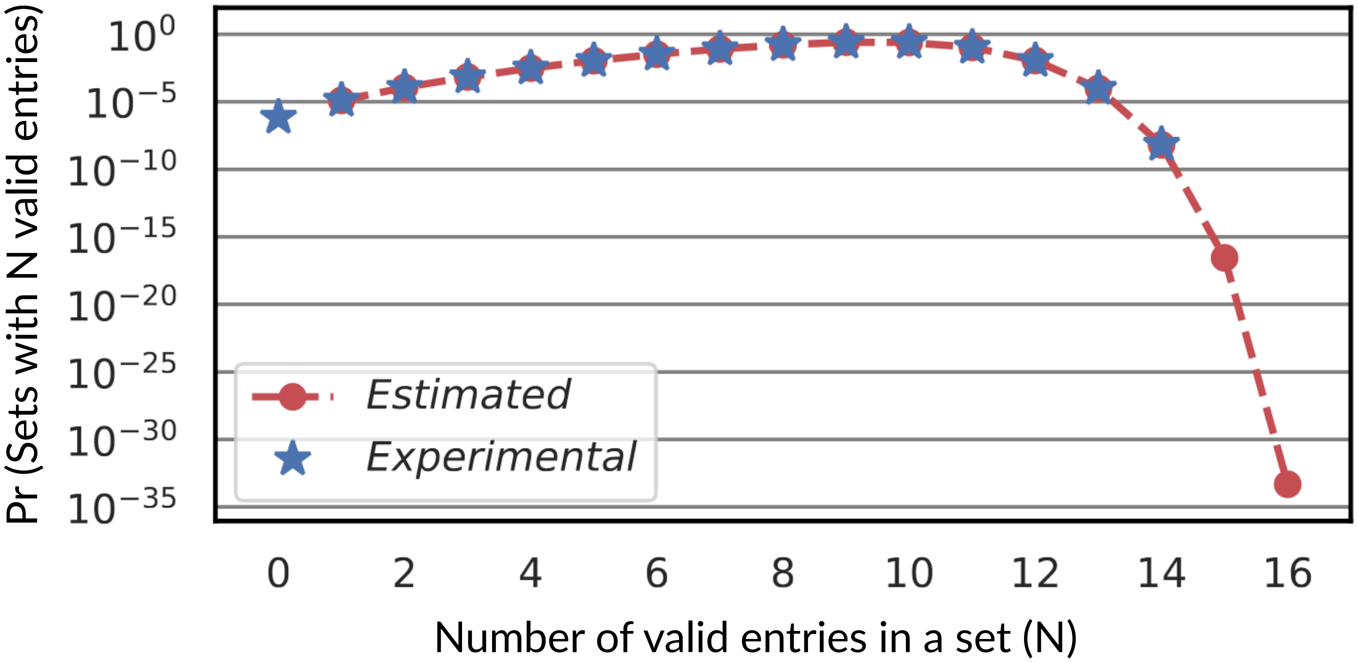
Security Model



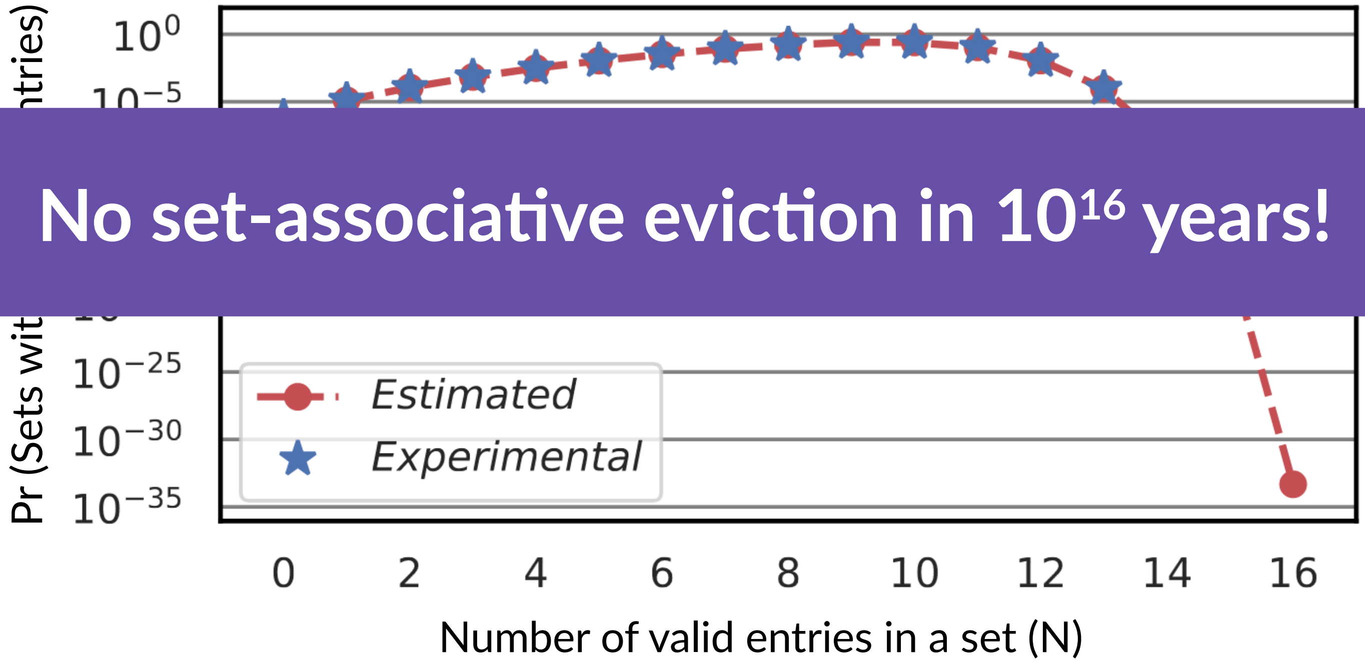
Security Model



Security of Maya

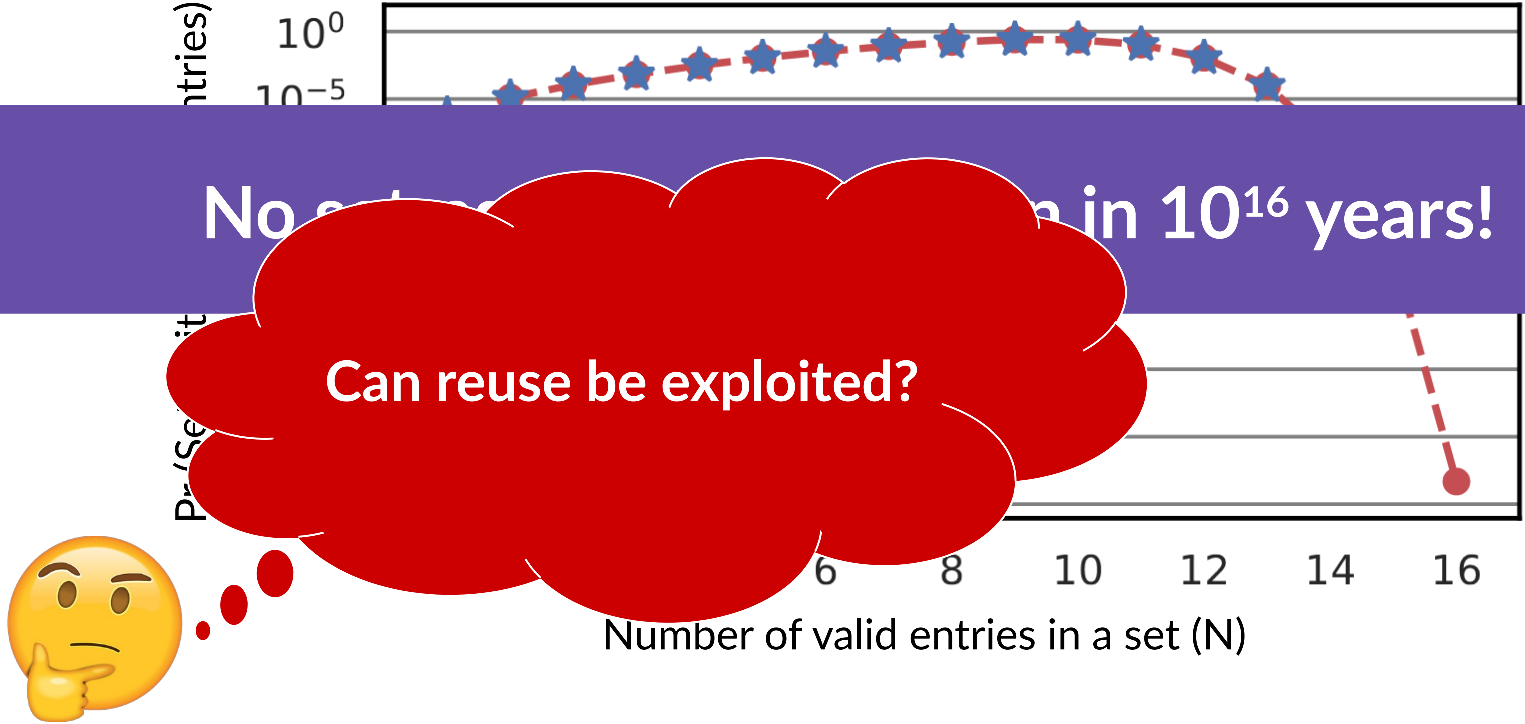


Security of Maya



No set-associative eviction in 10^{16} years!

Security of Maya



No se n in 10^{16} years!

Can reuse be exploited?

Exploiting Reuse

Maya uses Domain IDs for each tag entry

This isolates the reuse pattern of each domain

Therefore, Maya prevents any reuse-dependent fill-based attack

Exploiting Reuse

Maya uses Domain IDs for each tag entry

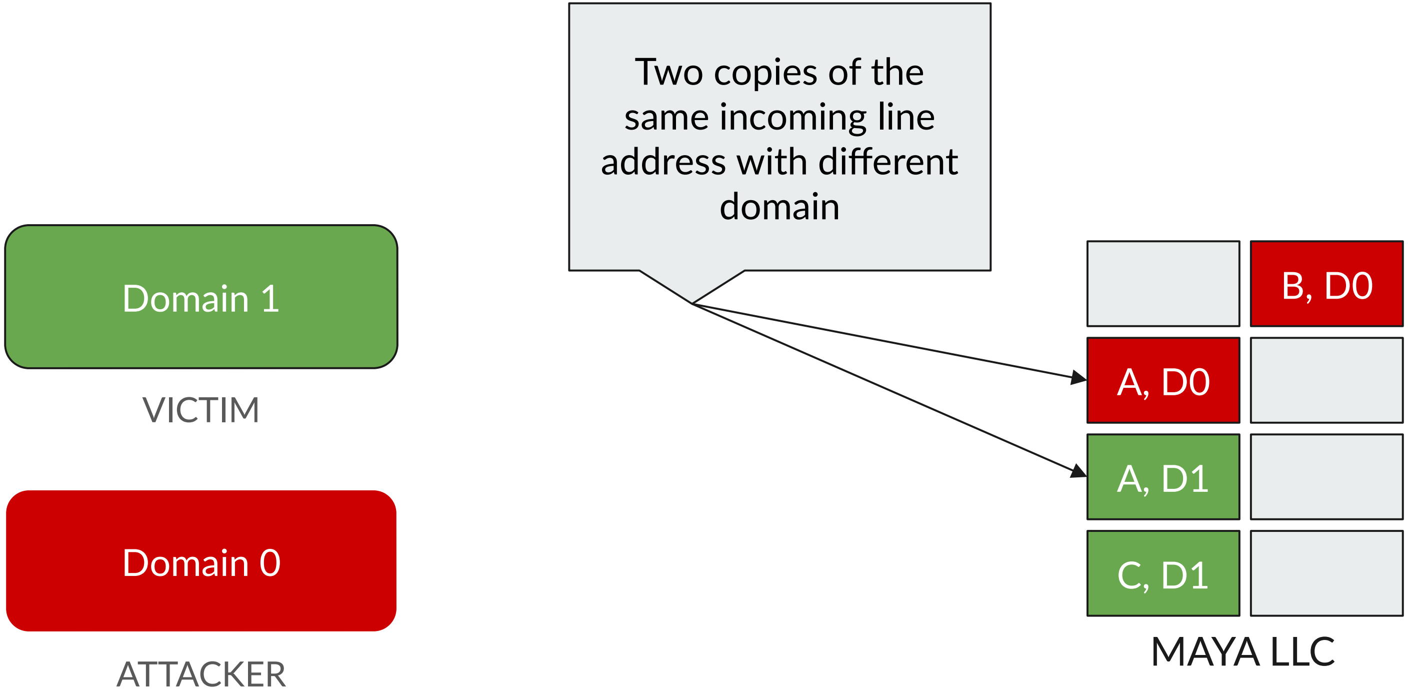
T each domain

What about shared memory attacks?

T e-dependent fill-

k

Shared memory attacks



Two copies of the same incoming line address with different domain

Domain 1

VICTIM

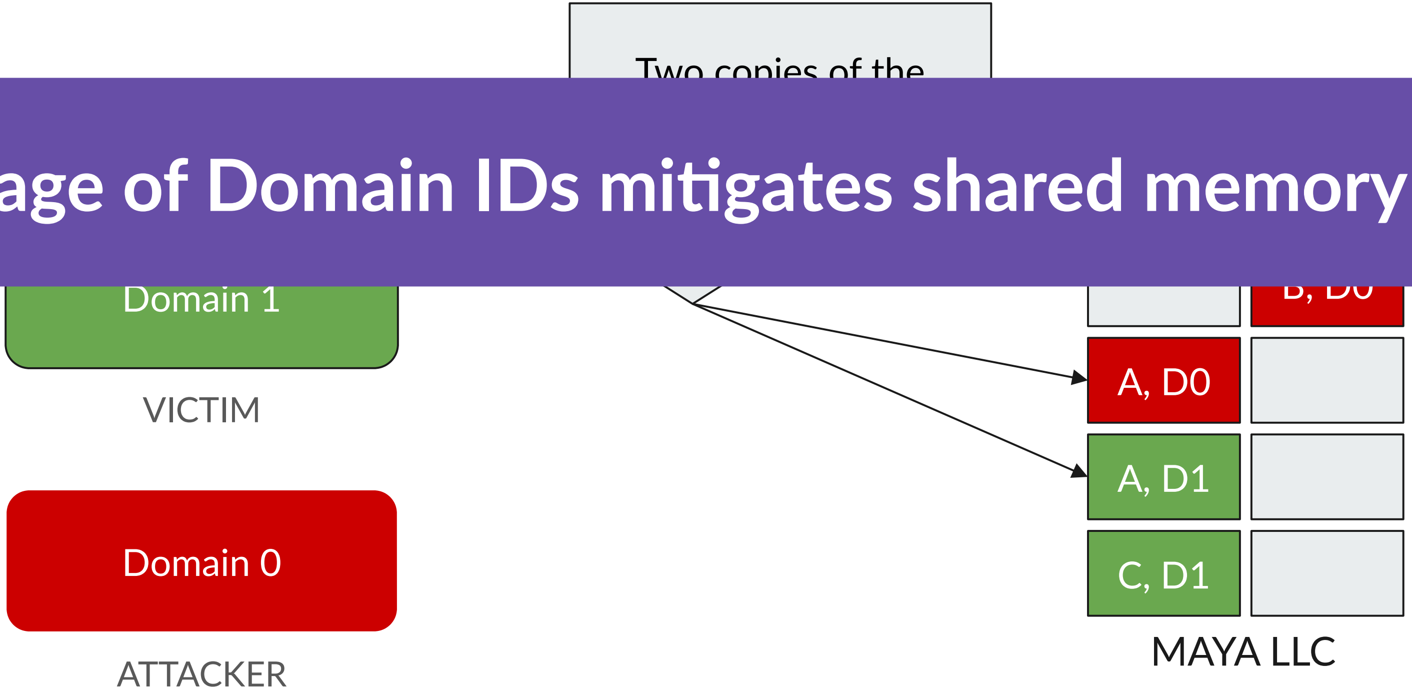
Domain 0

ATTACKER

	B, D0
A, D0	
A, D1	
C, D1	

MAYA LLC

Shared memory attacks

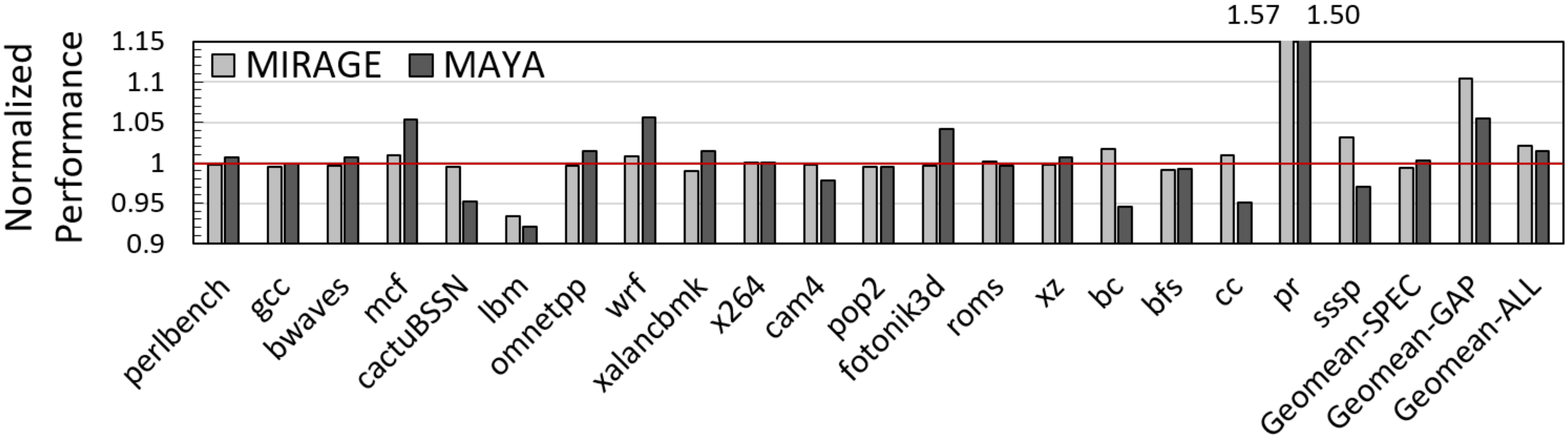


Usage of Domain IDs mitigates shared memory attacks

Evaluation

Simulator	ChampSim Multicore Simulator
CPU	8-core OoO
L1/L2C	Private L1/L2 Caches
Baseline LLC	Shared, 16MB data store, 24 cycles
Maya LLC	Shared, 12MB data store, 28 cycles
Benchmarks	42 SPEC2017 traces, 20 GAP traces
Instructions	200M warmup, 200M simulation

Performance Results



Marginal performance improvement over a non-secure baseline

The game of tradeoffs (win-win)

Performance	~2% improvement
Storage	2% savings
Area	28% savings
Read Energy	15% savings
Write Energy	11% savings
Leakage Power	5% savings
Security?	Yes :)

Summary

Maya is a randomized fully associative last-level cache that uses additional tag entries and fewer data entries

Uses a reuse-based insertion policy

It guarantees no set-associative evictions in 10^{16} years

Maya provides a sweet spot in terms of security, performance, area and energy requirements

Thank You!

Artifact