CASPER

Do not Forget Hardware Prefetching While Designing Secure Cache System

MS Bi-annual Progress Seminar

Sumon Nath | *sumon@cse.iitb.ac.in*
Advised by **Prof. Biswabandan Panda**

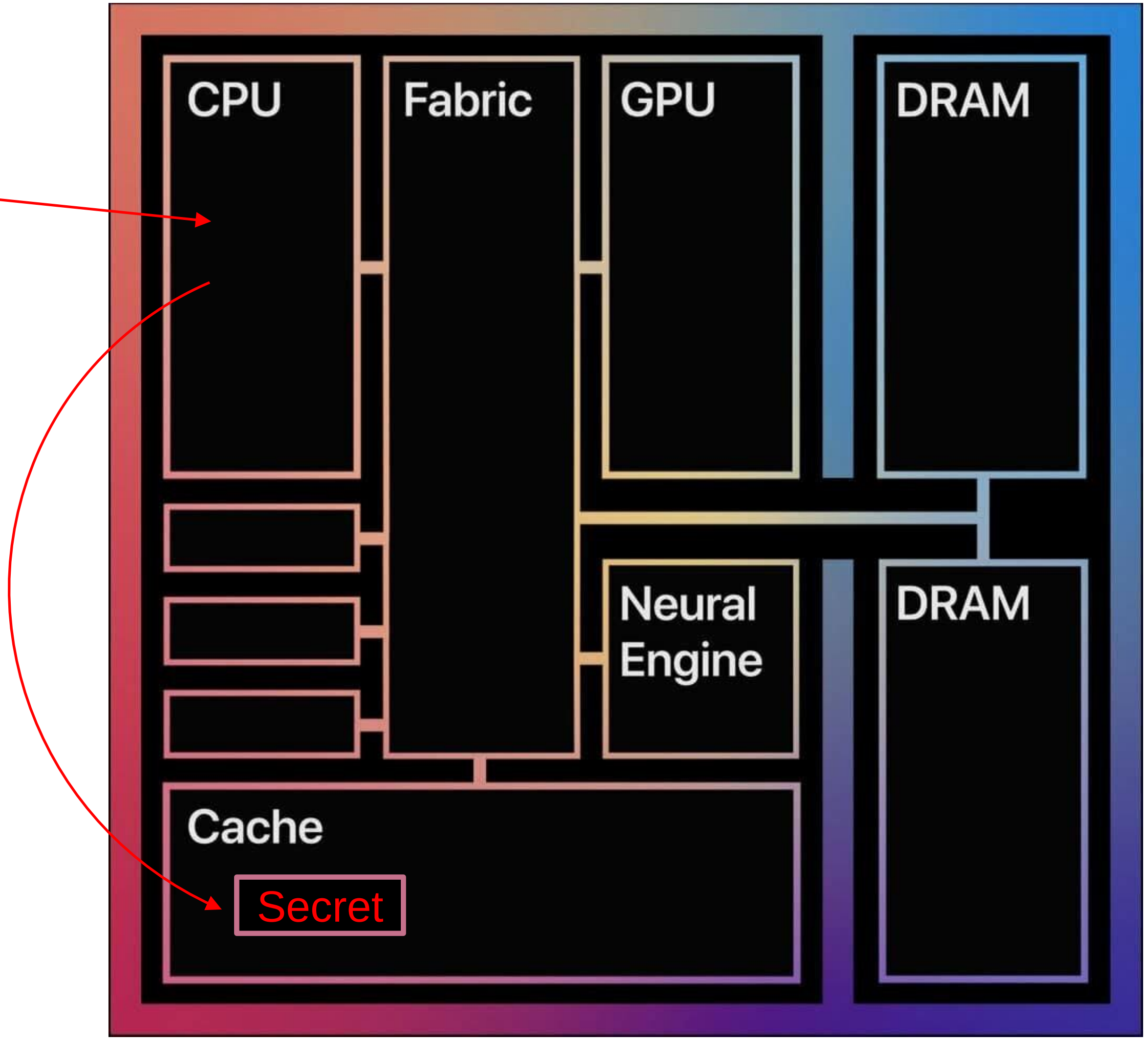
Recap: Speculative execution attacks

- Out-of-order execution
- Branch prediction

- Wrong path execution
- Secret data into cache

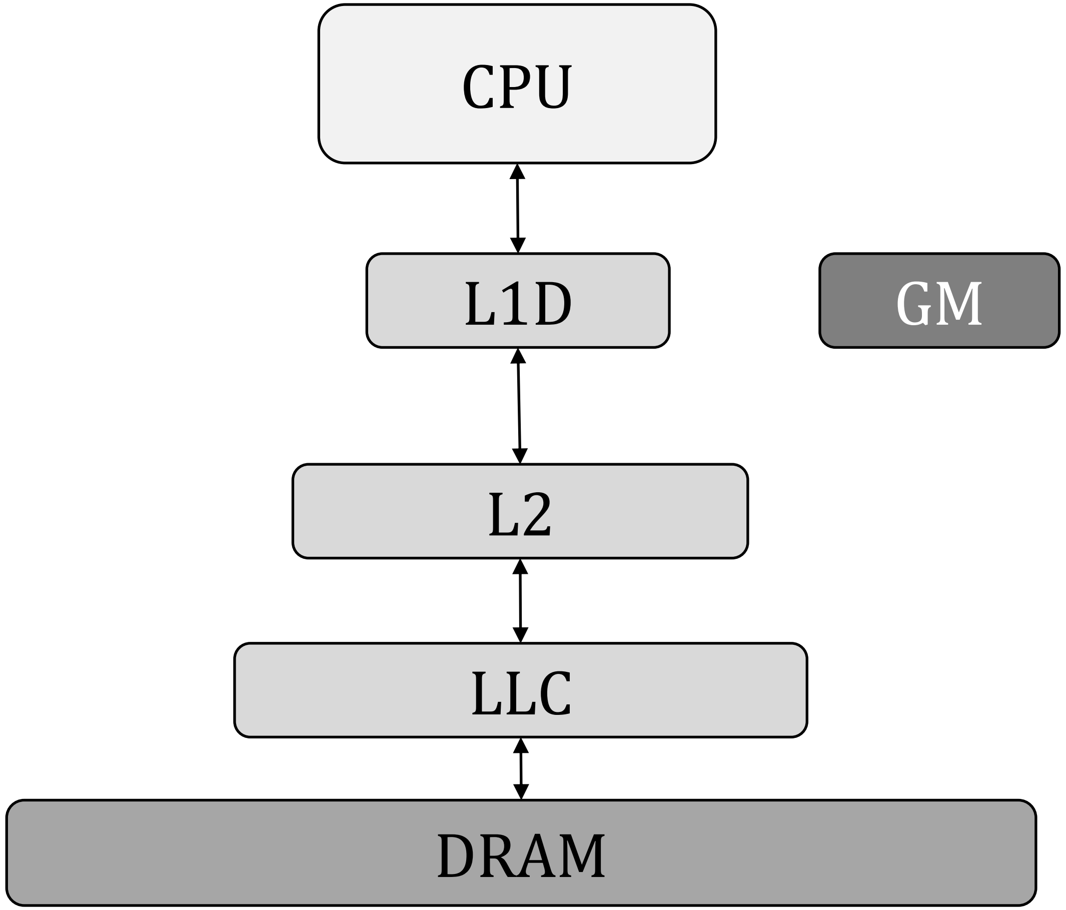
- Attacker access secret data across context switches

Apple M1



https://www.apple.com/in/newsroom/2020/11/apple-unleashes-m1/

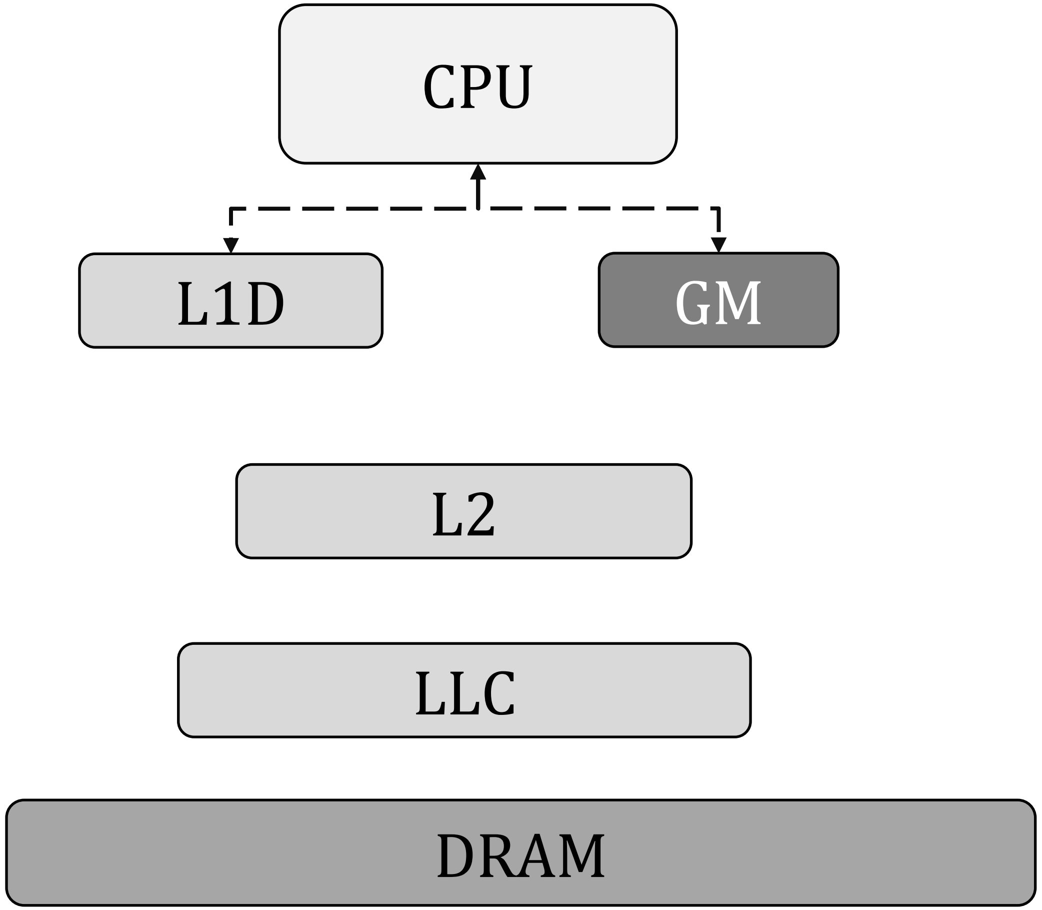
Secure Cache system: GhostMinion[MICRO '21]



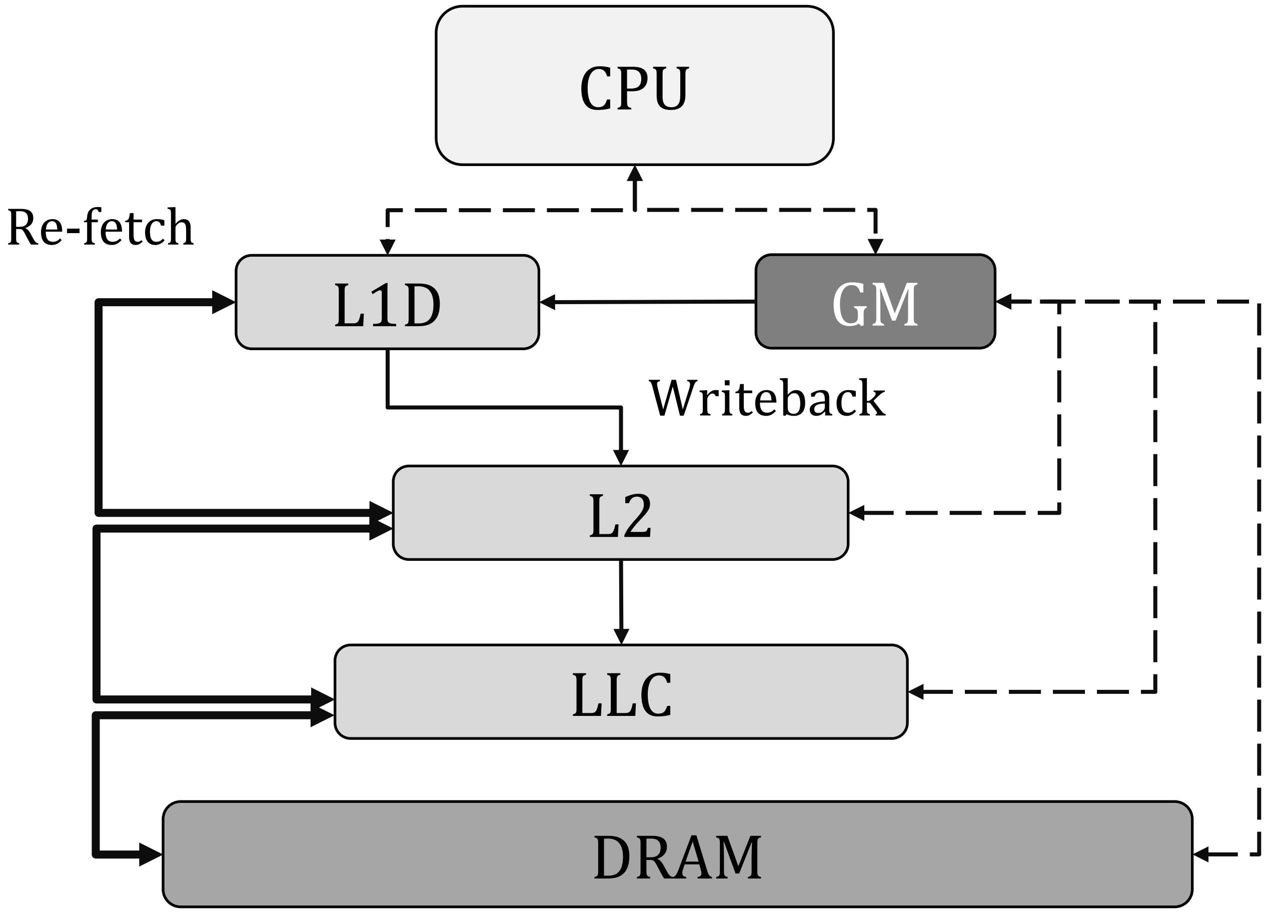
Filter cache

- Stores all speculative data
- Wiped on context switch

Secure Cache system: GhostMinion[MICRO '21]



Secure Cache system: GhostMinion[MICRO '21]



- - - - Speculative path
—— Commit Write – hit in GM
—— Commit Load – miss in GM

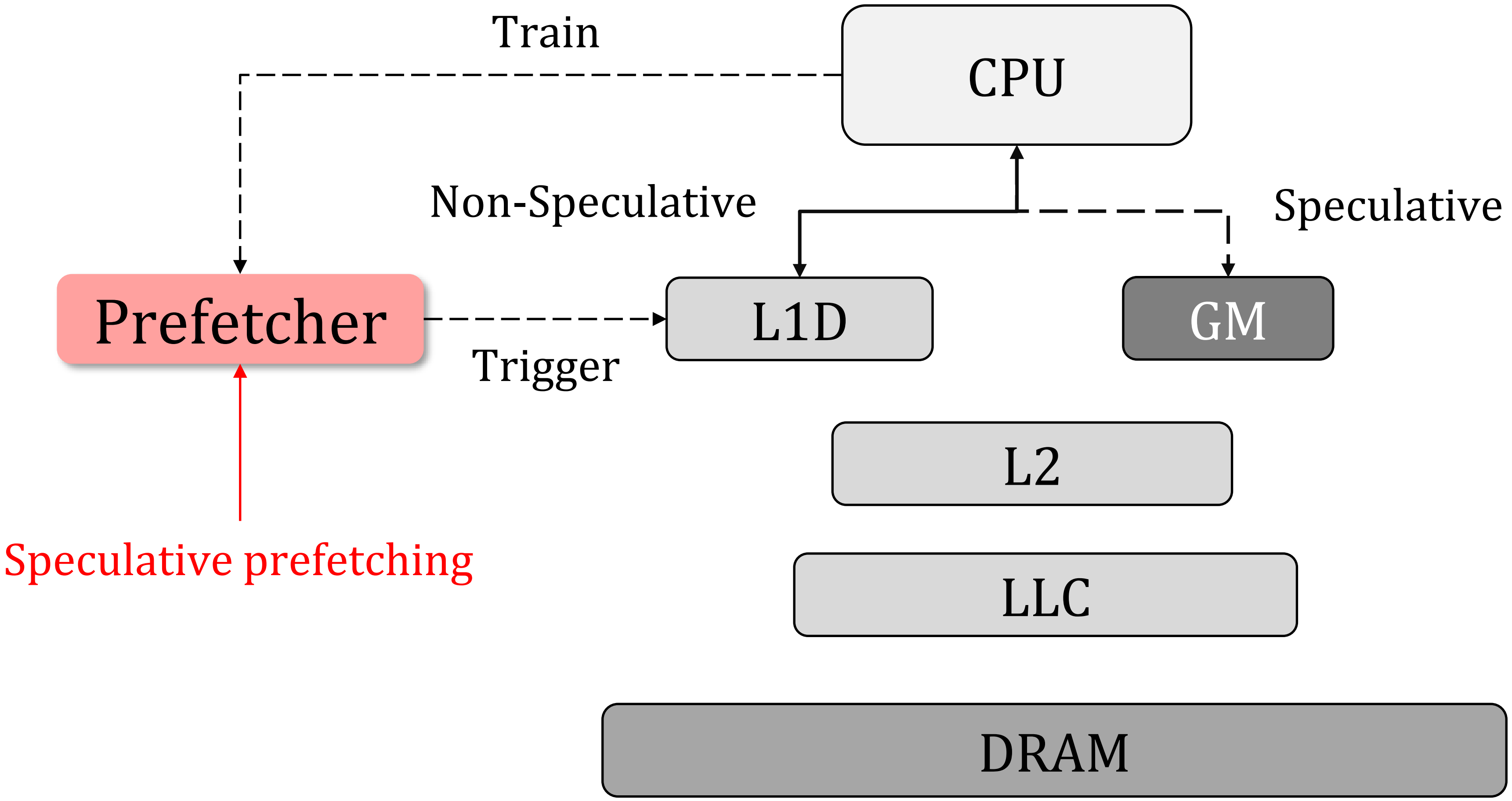
Cache state untouched

- LRU bits **not updated** on hit
- **Bypass fills** to L1/L2/LLC

Caches updated On-commit

- **Writeback** to L1-> L2 -> LLC
- **Re-fetch** to L1

What about prefetching?



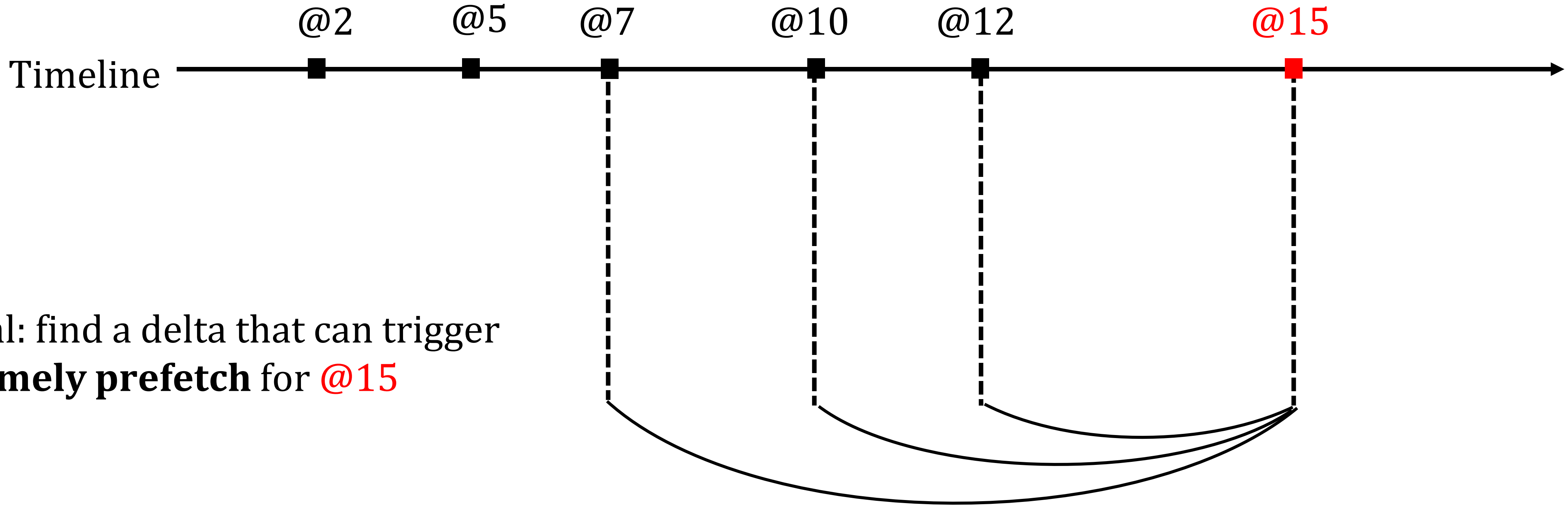
Speculative prefetching

Train & Trigger
~~**On-access**~~
On-commit

Prefetcher of interest: Berti[MICRO '22]

Accurate and **timely local delta** L1D prefetcher

- **Delta**: Diff. b/w two cache lines
- **Local**: Per Instruction Pointer
- **Timely**: Prefetched before demand access



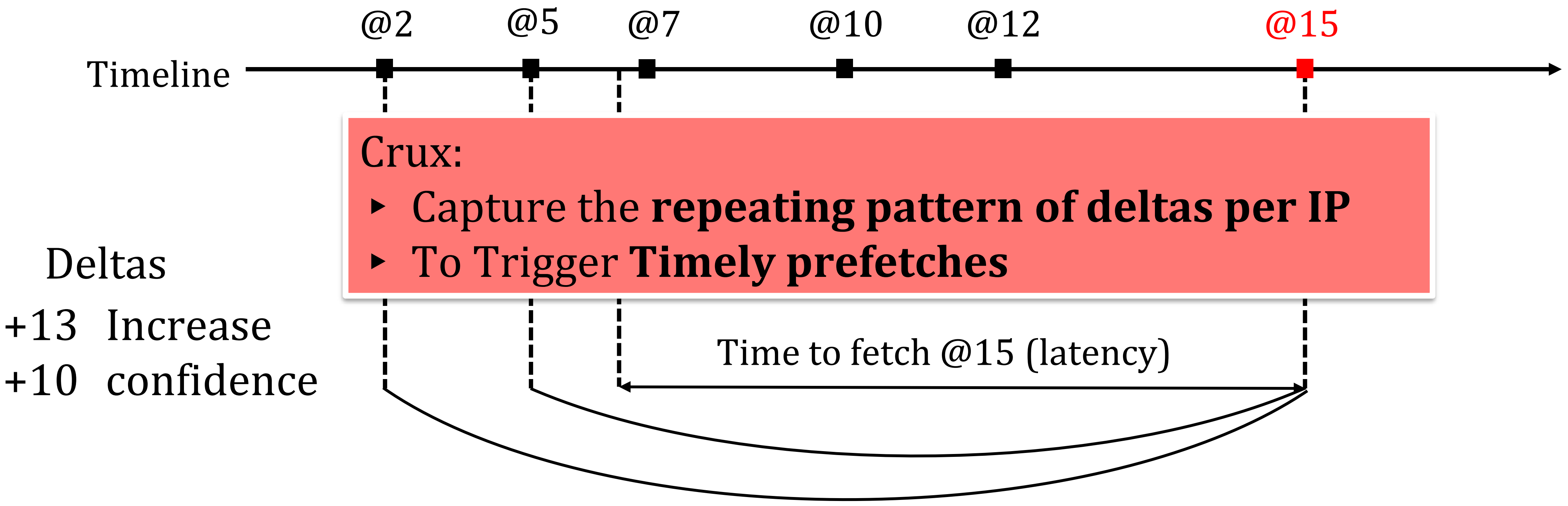
Goal: find a delta that can trigger
a **timely prefetch** for @15

deltas: +8, +5, +3

Berti[MICRO ‘22] prefetcher

Accurate and **timely local delta** L1D prefetcher

- **Delta**: Diff. b/w two cache lines
- **Local**: Per Instruction Pointer
- **Timely**: Prefetched before demand access

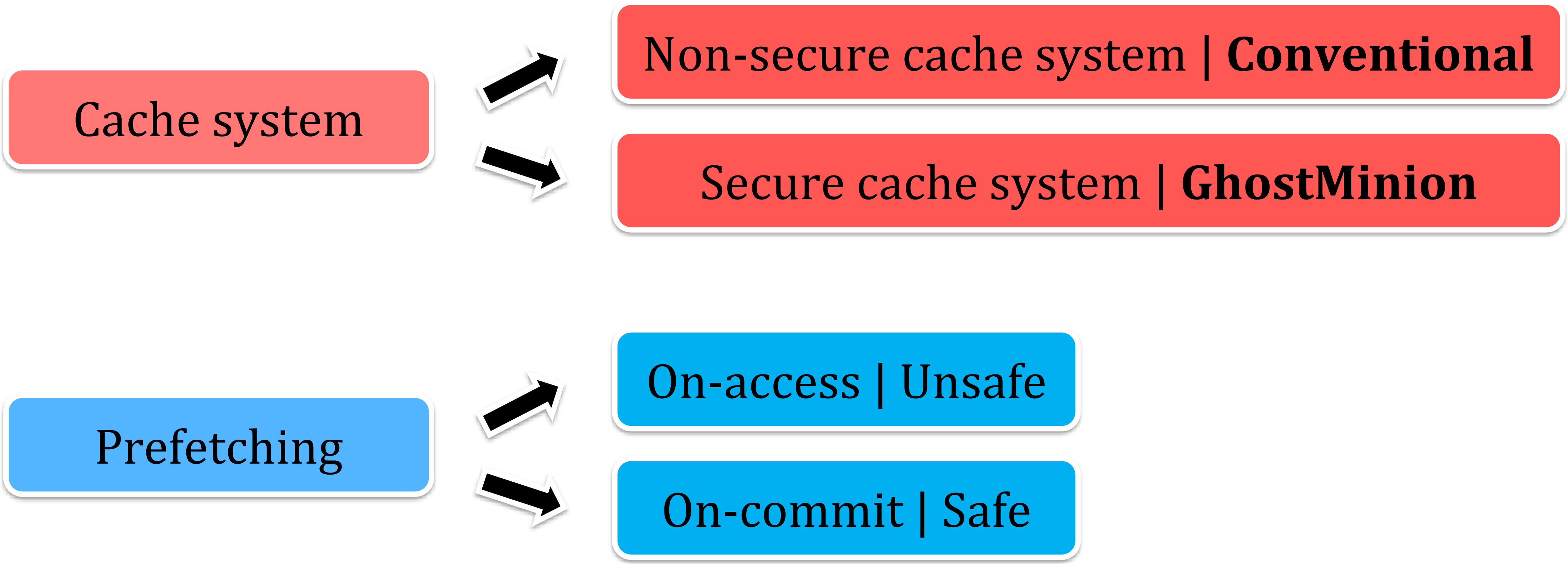


Crux:

- Capture the **repeating pattern of deltas per IP**
- To Trigger **Timely prefetches**

Timely deltas: +13, +10

Terminology



Cache system

Non-secure cache system | **Conventional**

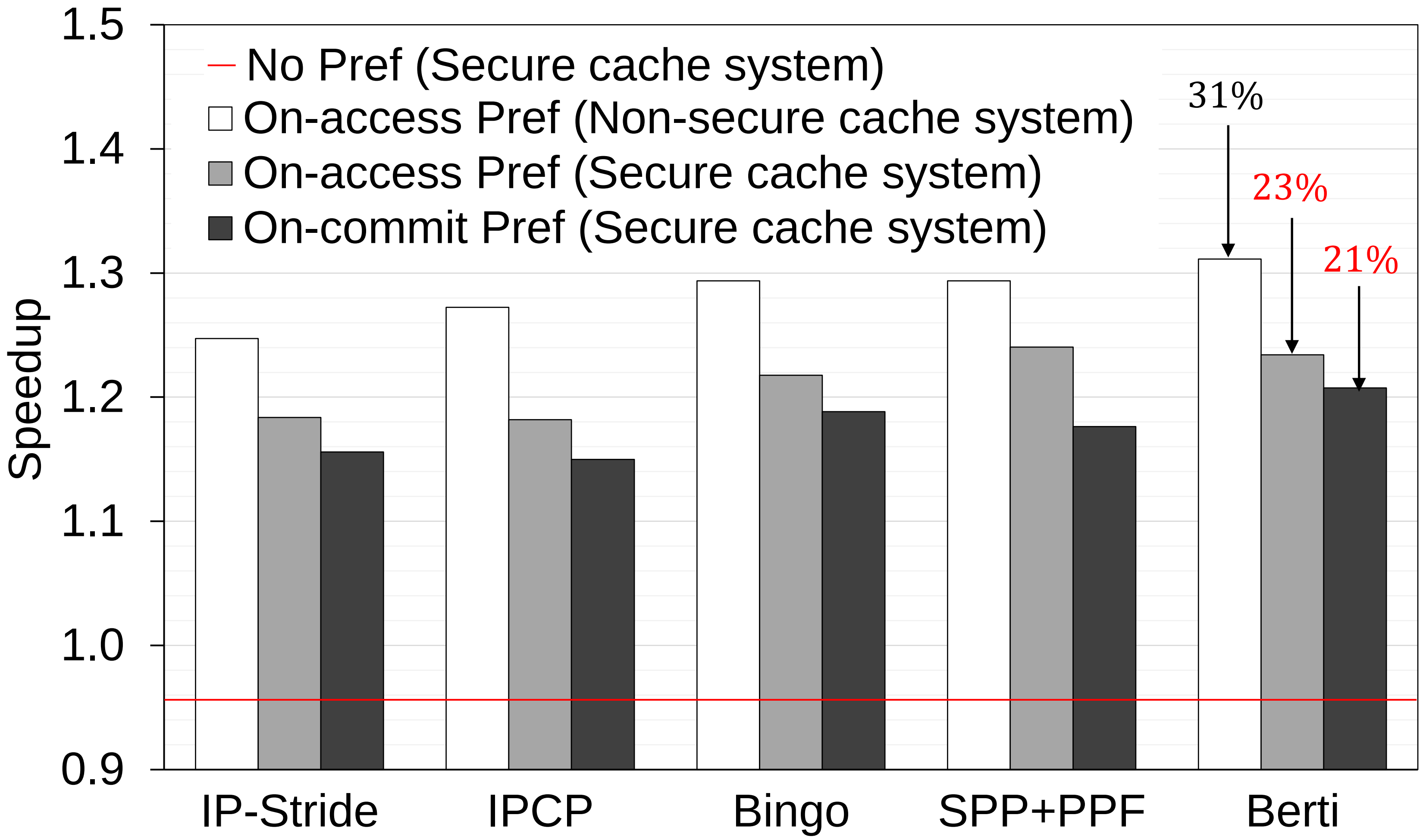
Secure cache system | **GhostMinion**

Prefetching

On-access | Unsafe

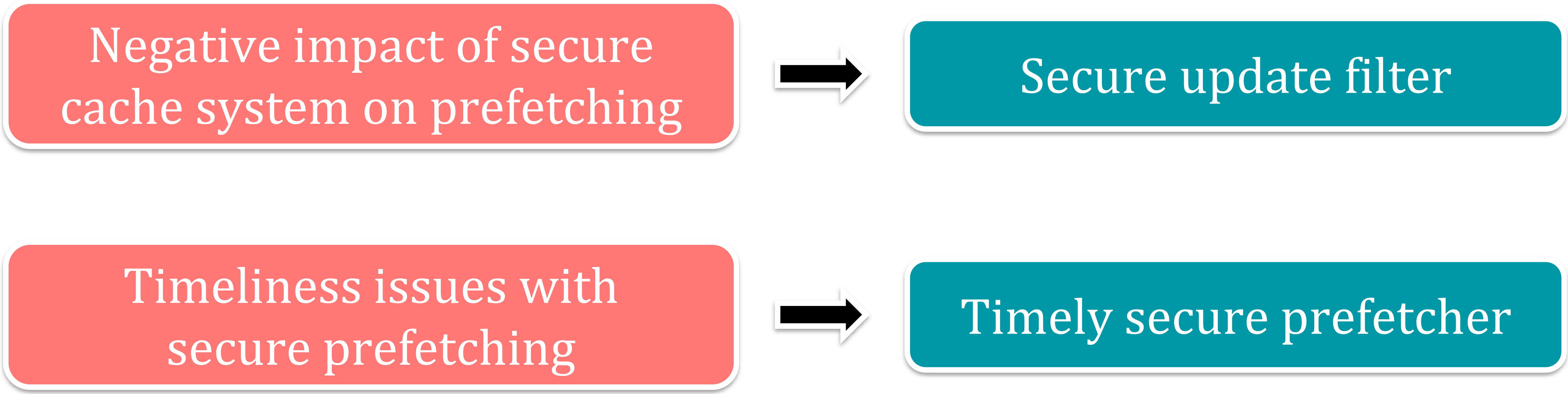
On-commit | Safe

Impact of secure cache system & on-commit prefetching



Overall **~11 % degradation** from on-access(non-secure) to on-commit (secure)

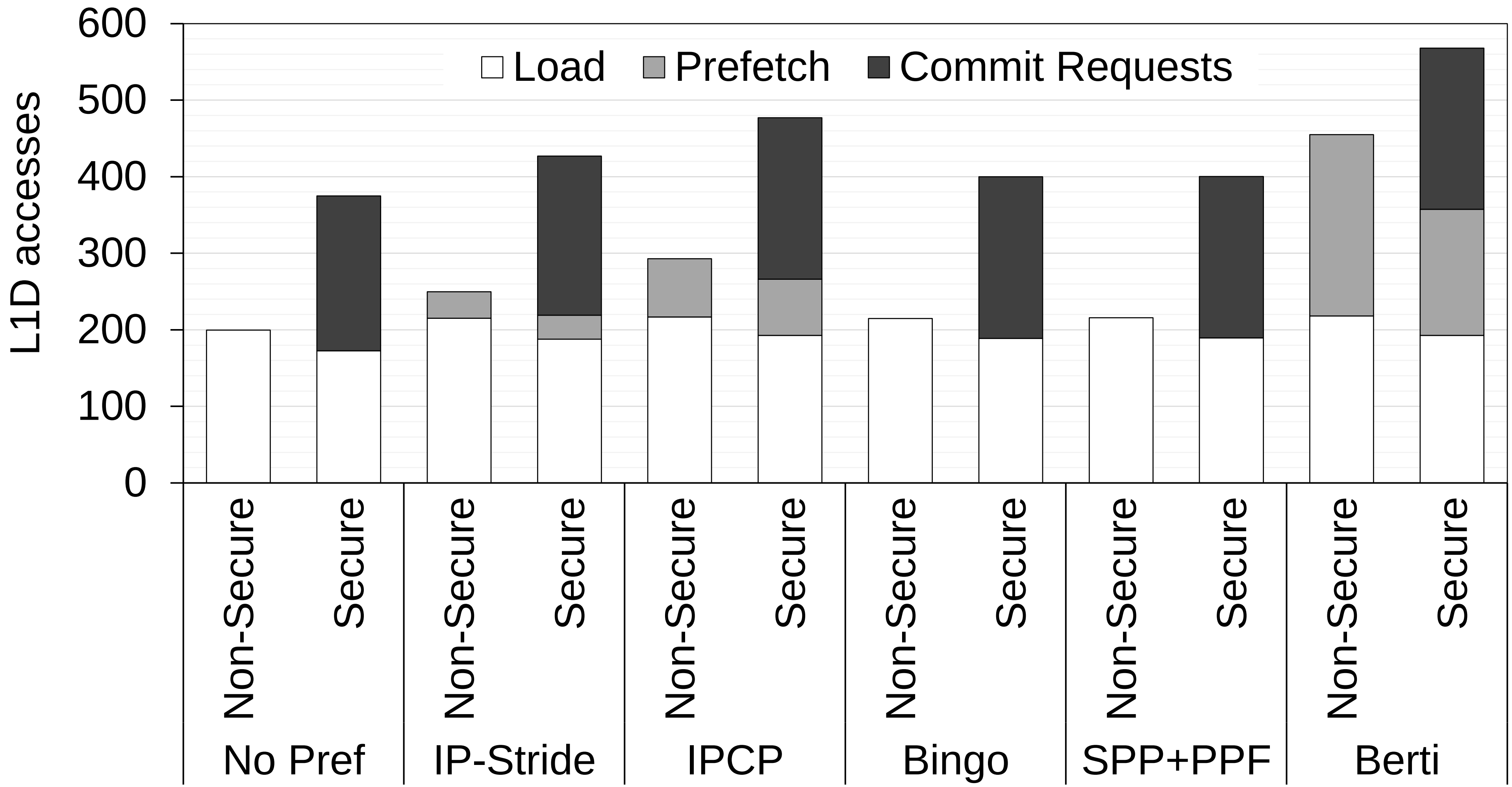
Problems & contributions



Negative impact of secure cache system on prefetching → Secure update filter

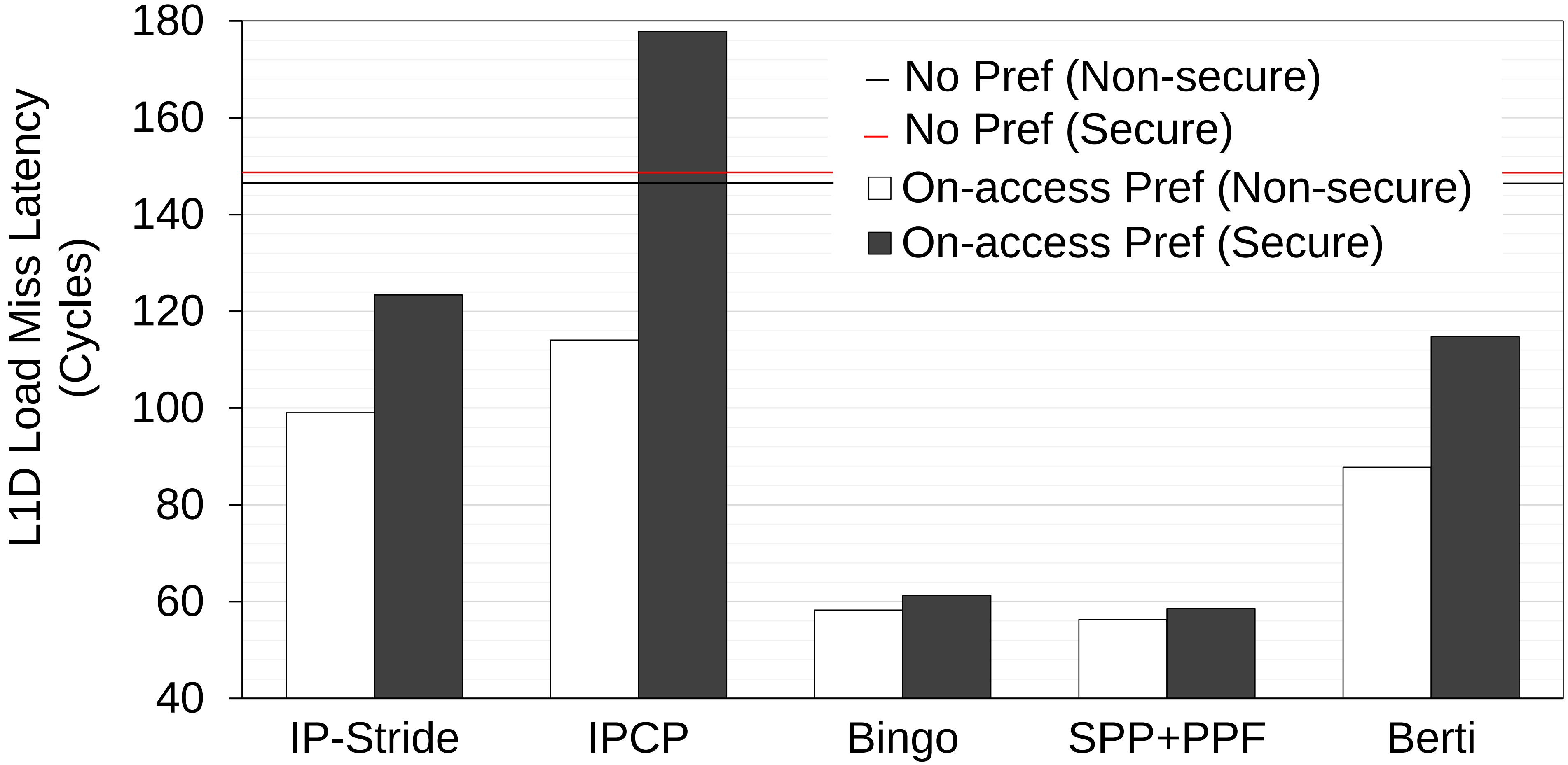
Timeliness issues with secure prefetching → Timely secure prefetcher

Problem 1: Impact of secure cache system



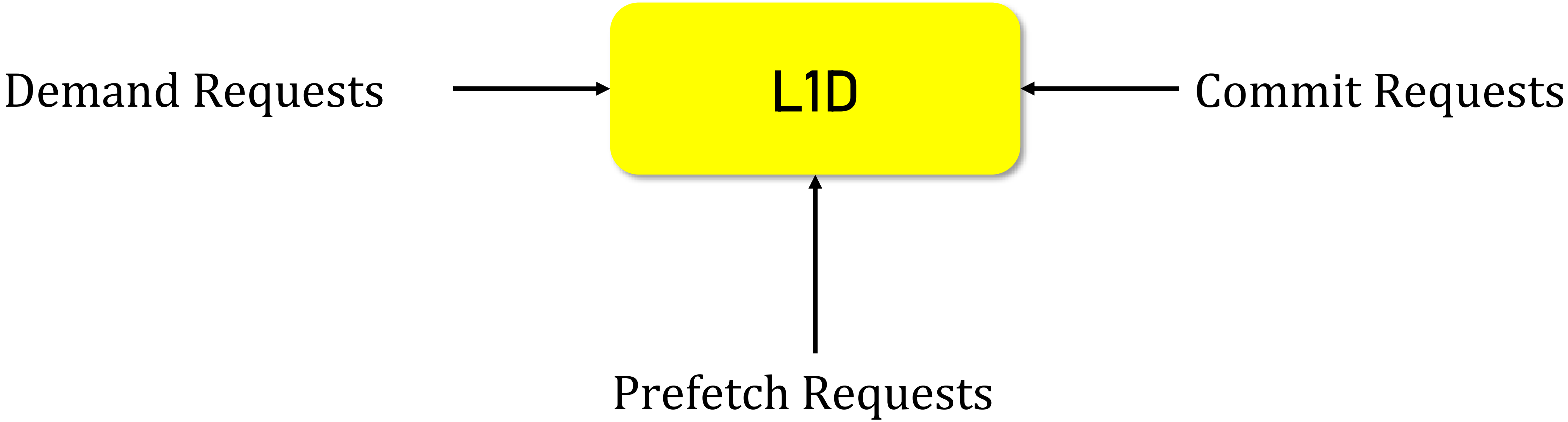
Average **Traffic** increases from **199 to 375** accesses per kilo instruction (APKI)

Increase in miss latency



L1D load miss latency increases from **83 to 117** cycles

Increased Pressure on L1D



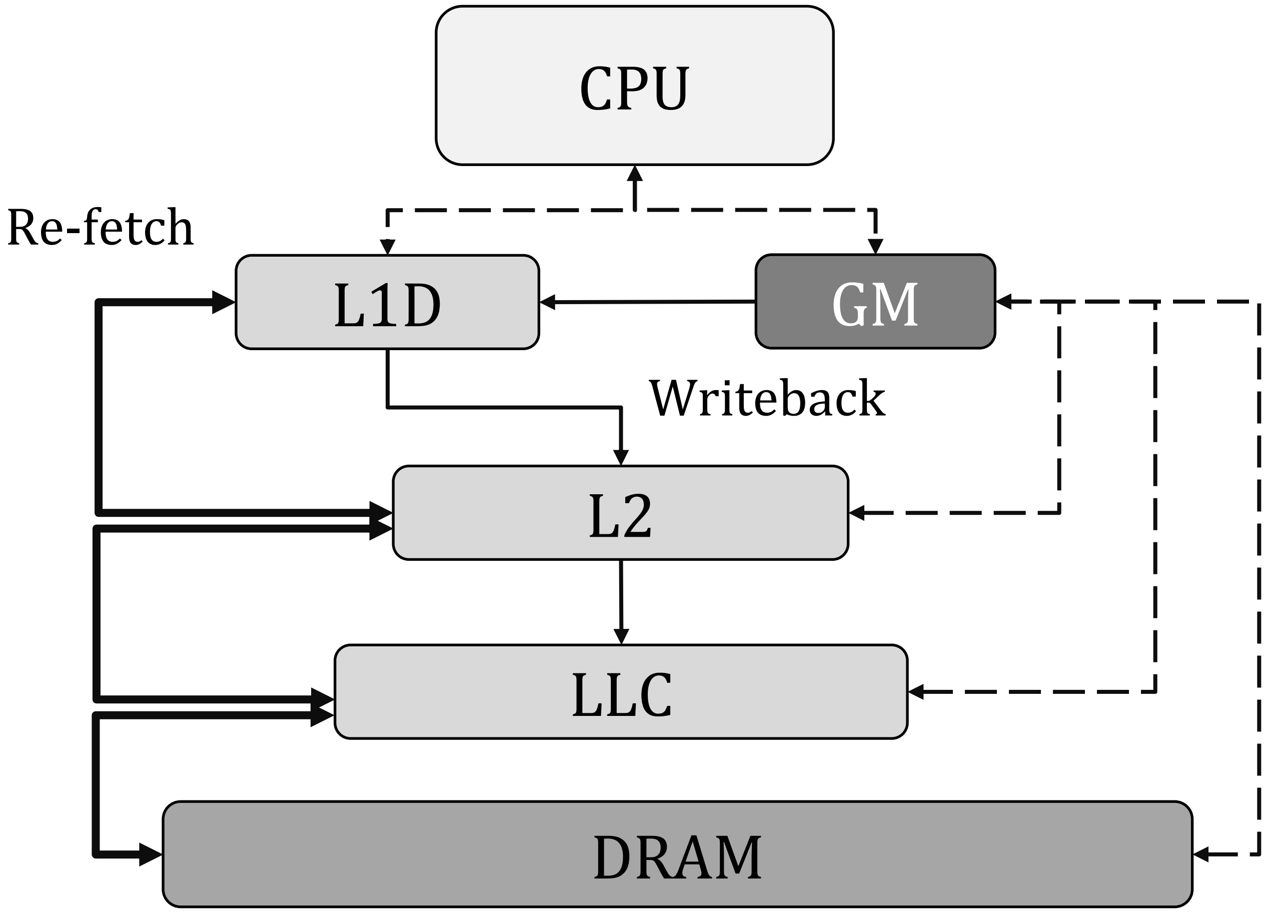
Demand Requests

L1D

Commit Requests

Prefetch Requests

Redundant requests to cache system

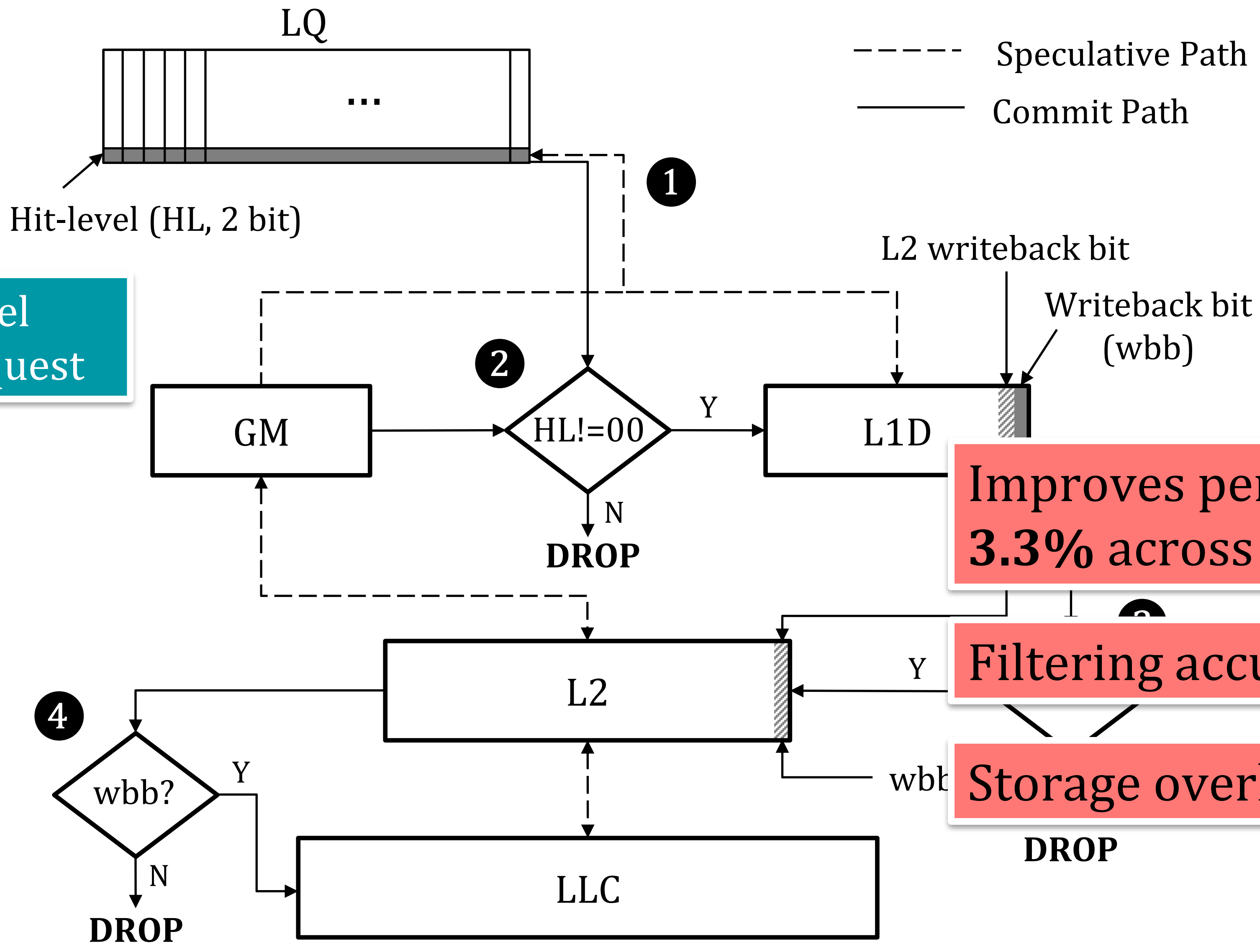


- - - - Speculative path
──── Commit Write
━━━━ Commit Load

More than **95%** redundant commit requests to L1D

Enhancement 1: Secure update filter(SUF)

Hit level: cache level that served the request



Improves performance by **3.3%** across prefetchers

Filtering accuracy > **98%**

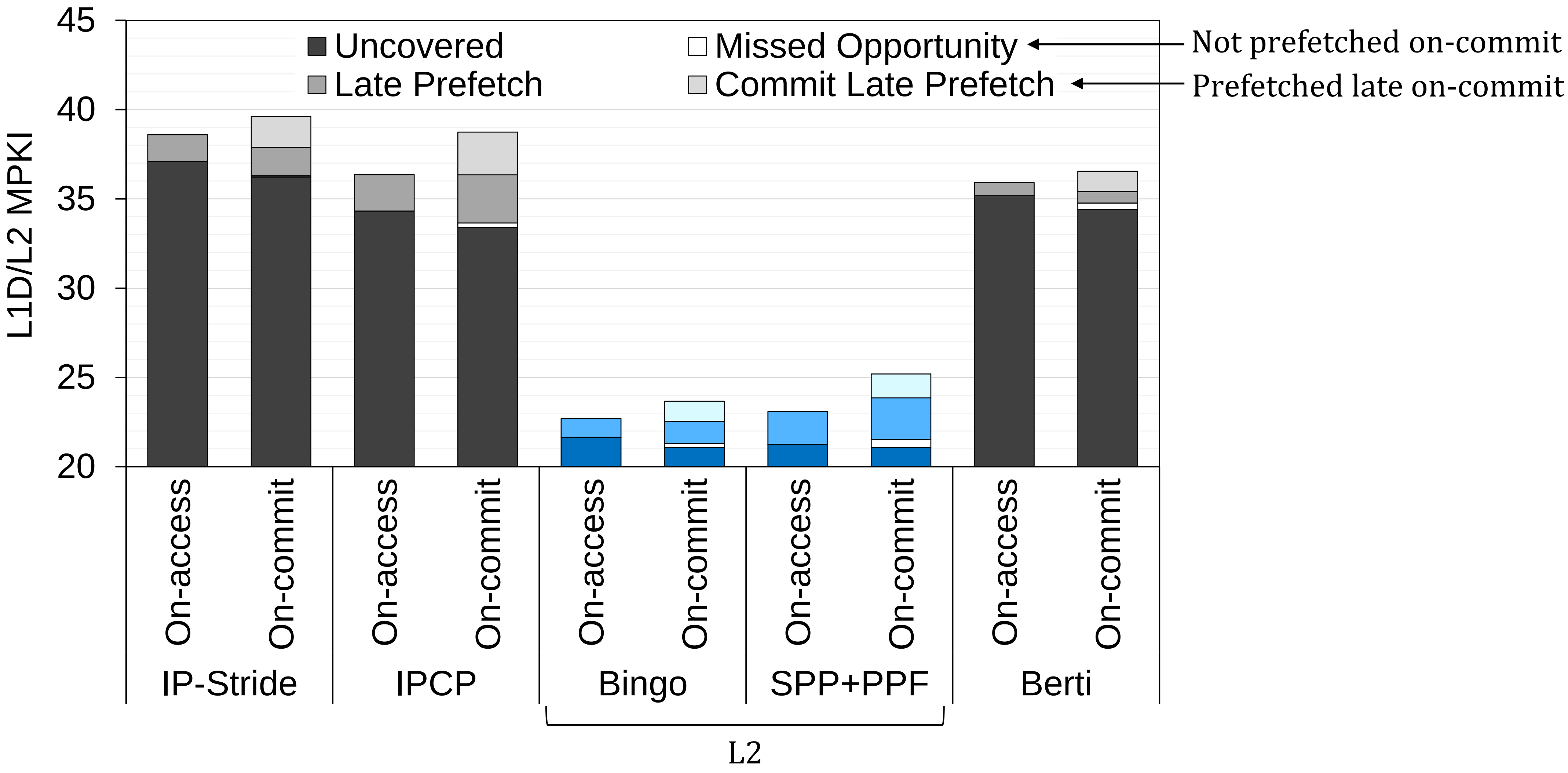
Storage overhead **0.12 KB**

Problem 2: Issues with Secure Prefetcher

- Berti relies on fetch latency
- Latency seen by prefetcher is mis-leading (Timeliness)
- Trigger events should be relative to on-commit events

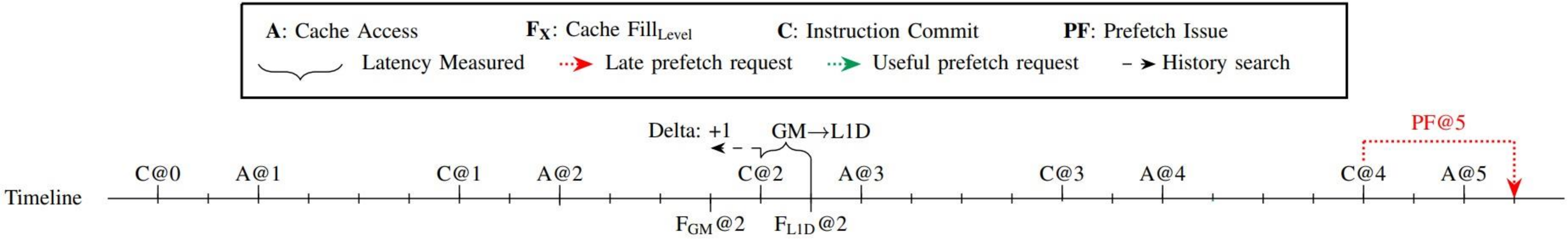
On-commit prefetching Timeliness

Timely prefetch if prefetcher trained on-access

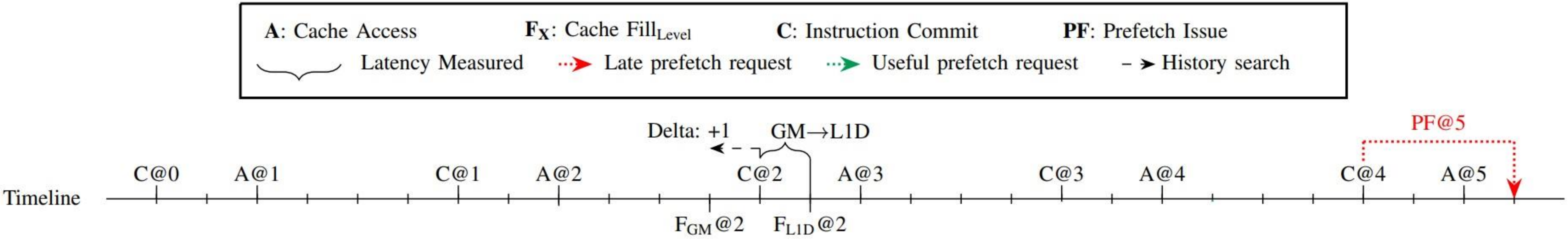


Commit Late prefetches increases overall MPKI

On-commit Berti: Measured latency problem



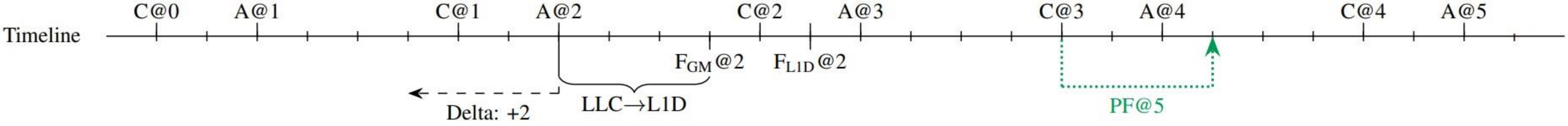
On-commit Berti: Measured latency problem



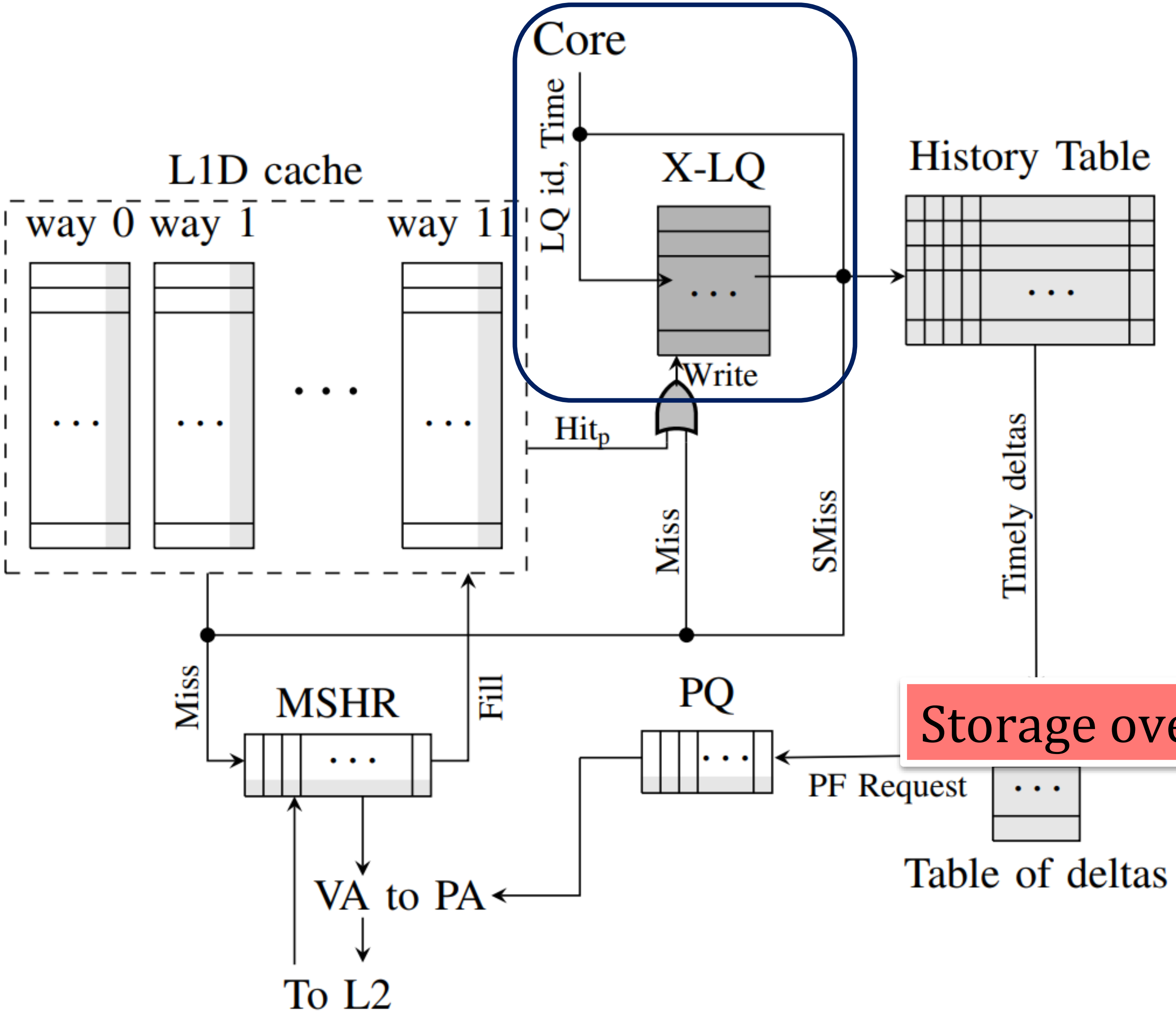
A: Cache Access **F$_X$**: Cache Fill$_{\text{Level}}$ **C**: Instruction Commit **PF**: Prefetch Issue

Latency Measured ⇢ Late prefetch request ⇢ Useful prefetch request – ➤ History search

Enhancement 2: Timely Secure Berti (TSB)



TSB: Hardware implementation

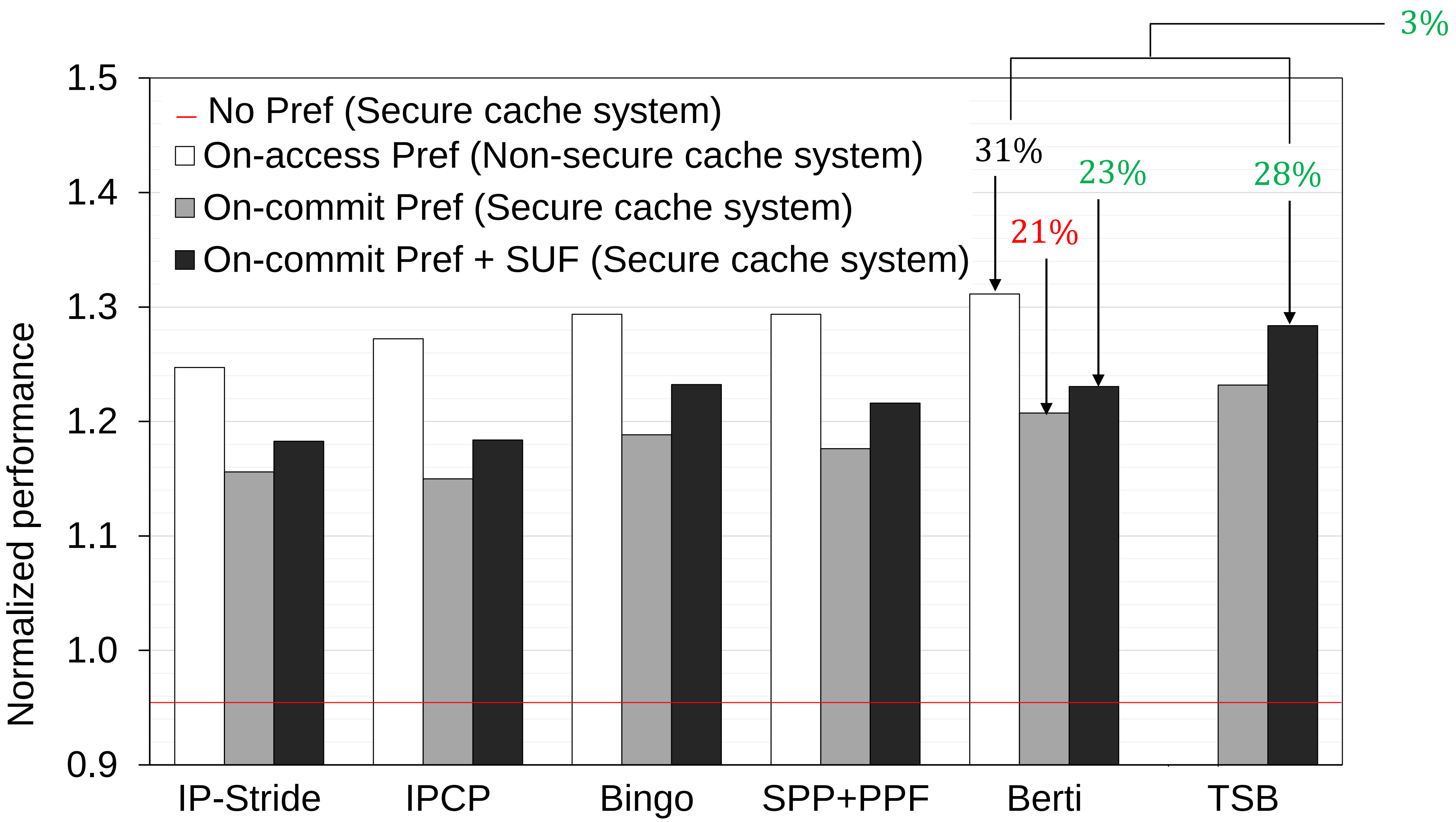


Storage overhead **0.47 KB**

Evaluation setup

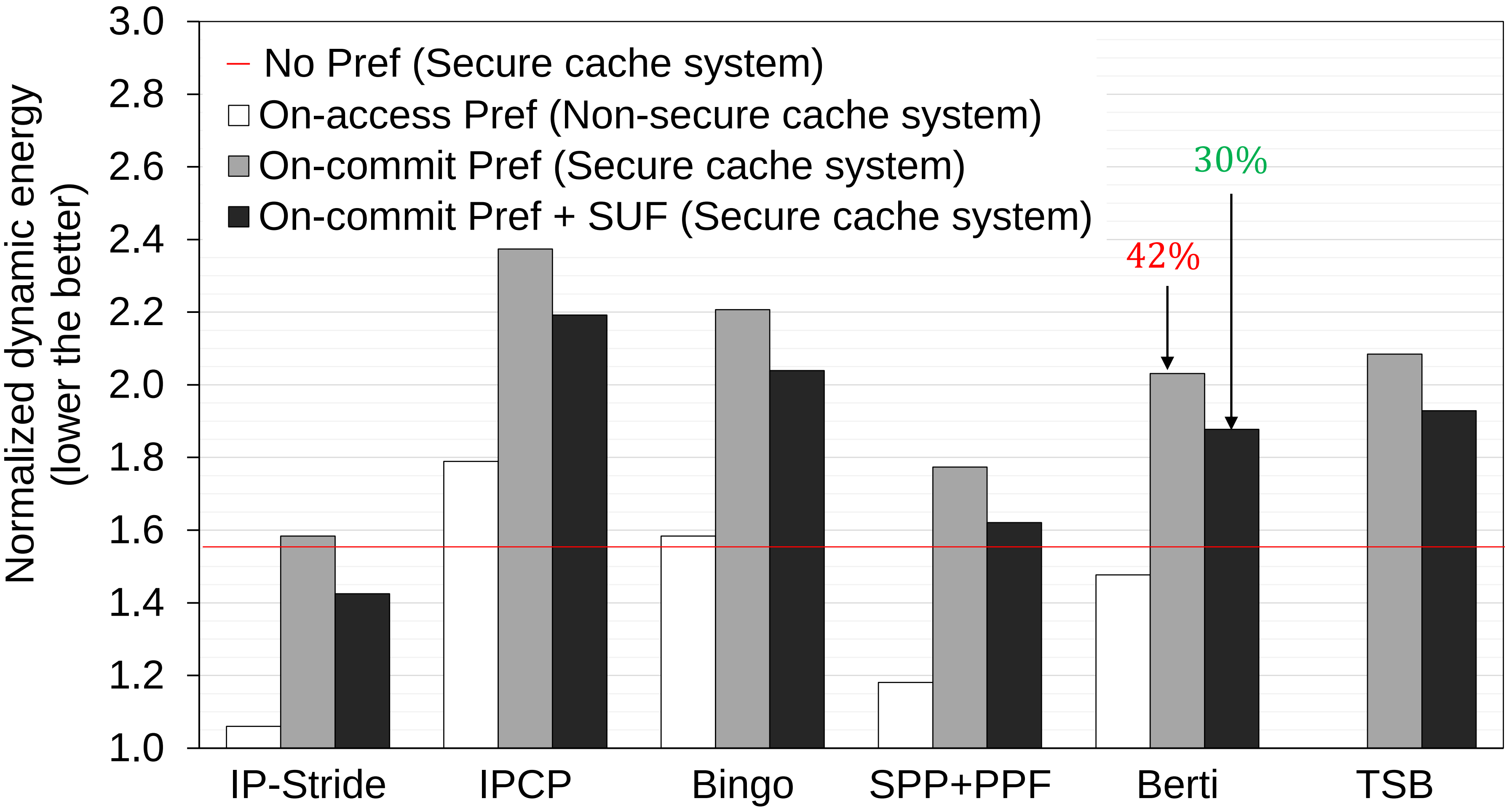
- Champsim(Intel Sunny Cove)
- Normalized to non-secure cache system without prefetching
- SPEC CPU 2017 & GAP benchmarks

Performance improvement



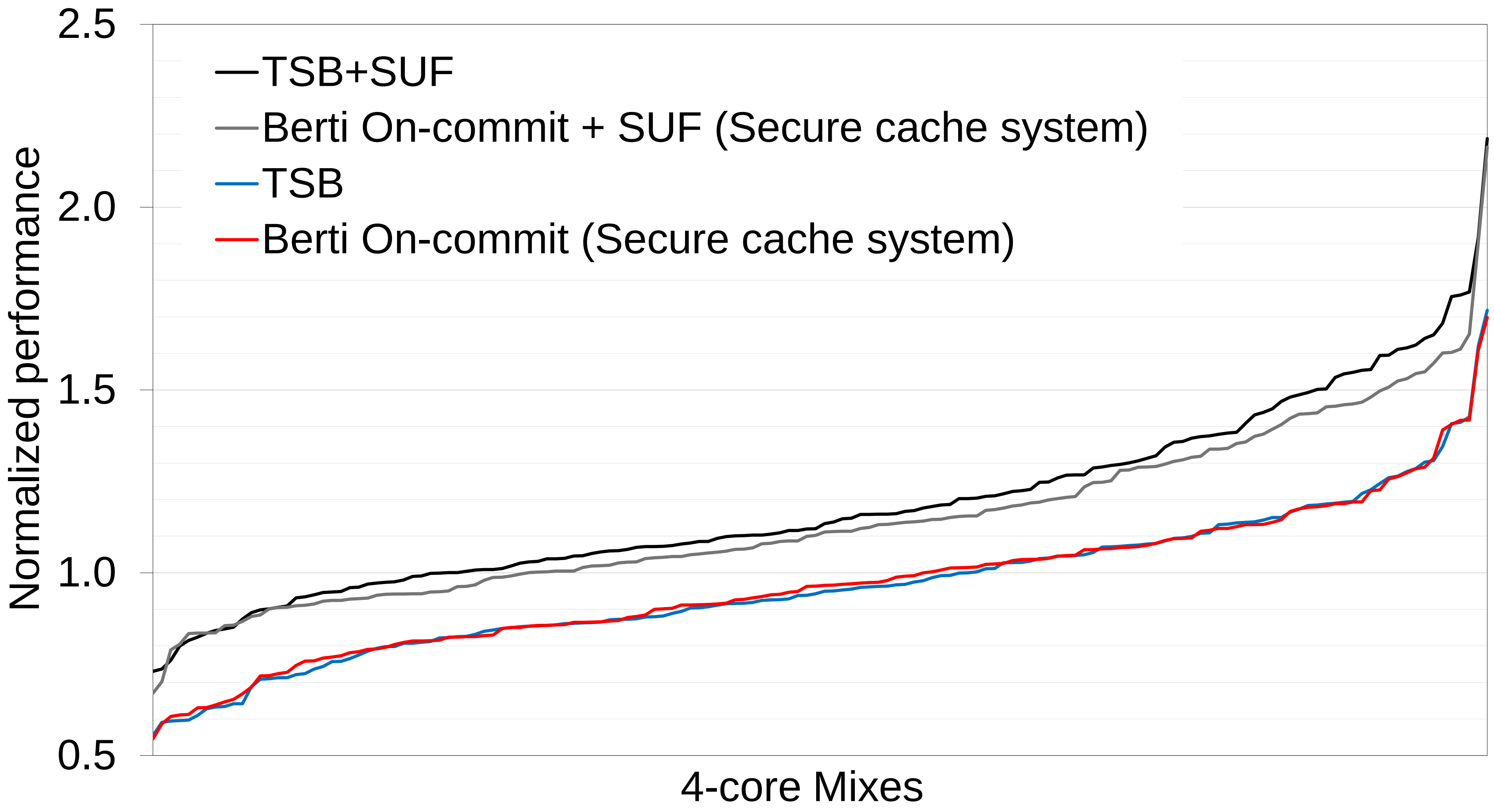
TSB improves performance by **6.3 %** compared to On-commit Berti

Power reduction



SUF reduces dynamic energy consumption by **9.1%**

Multi-core performance



TSB+SUF improves performance by **16.1%** over baseline

Summary

- **Secure cache systems** hinders prefetcher performance
- Prefetchers are not built for **on-commit training/trigger**
- **SUF , TSB** improves prefetcher performance with security guarantees

Thanks and fingers crossed!

- Collaboration with **Prof. Alberto Ros and Agustin**, University of Murcia
- Work under submission to **ISCA 2024**, flagship forum in the field of computer architecture

Thank You

Background: Spectre[S&P '19](variant 1)

cached

uncached

```
if ( x < array1_size )
        y = array2[ array1[ x ] * 512 ]
```

11 E2 45 CD

- **x** , adversary controlled
- Adversary wants to read some **out of bounds** value of **array1**
- **array1_size** and **array2[ ]** not in cache
- Mistrains branch predictor

array2 [0 * 512]
array2 [1 * 512]
array2 [2 * 512]
array2 [3 * 512]
array2 [4 * 512]
array2 [5 * 512]
array2 [6 * 512]
array2 [7 * 512]
array2 [8 * 512]
array2 [9 * 512]
array2 [10 * 512]
array2 [11 * 512]
array2 [12 * 512]

Background: Spectre[S&P ‘19] (variant 1)

```
if ( x < array1_size )
        y = array2[ array1[ x ] * 512 ]
```

cached

uncached

Secret

11	E2	45	CD

array2 [0 * 512]
array2 [1 * 512]
array2 [2 * 512]
array2 [3 * 512]
array2 [4 * 512]
array2 [5 * 512]
array2 [6 * 512]
array2 [7 * 512]
array2 [8 * 512]
array2 [9 * 512]
array2 [10 * 512]
array2 [11 * 512]
array2 [12 * 512]

- Attacker calls victim with out-of-bounds x
  - Wrongly predicts true and starts Speculative exec
  - Reads addr (array1 base + x) with out-of-bounds x
  - Read returns Secret byte = **11**
  - Brings array2 [**11** * 512] into cache
  - Branch resolves: instructions squashed
- Attacker times reads from array2[i * 512]
  - Read for i = **11** is fast, revealing secret data