CS-773 Project Final Checkpoint

Hybrid Cache Architecture for Comprehensive Security

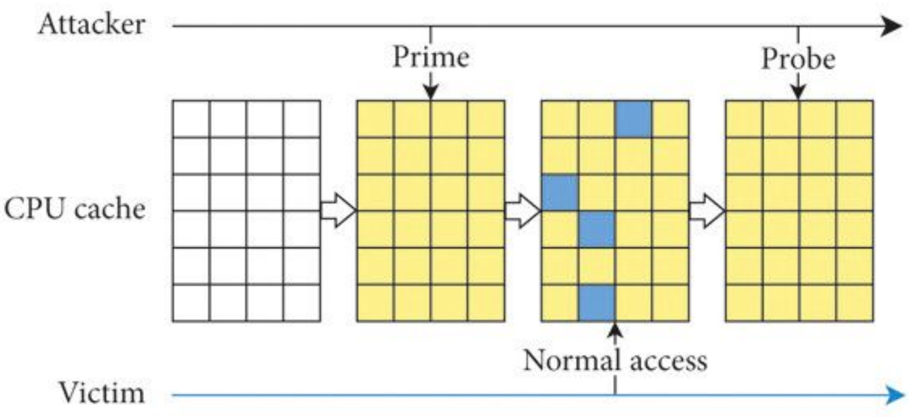
Soumik Dutta, Arnab Bhakta, SM Arif Ali

Team Gandiva

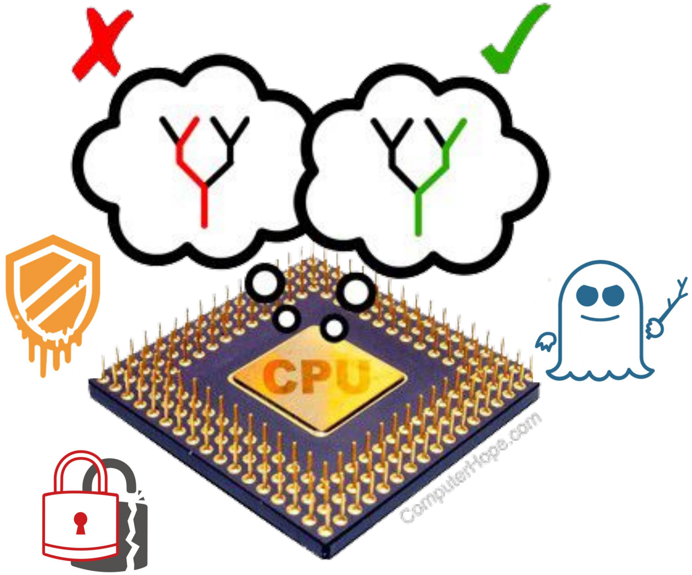
23m0826@iitb.ac.in,23m0835@iitb.ac.in,23m0822@iitb.ac.in

Problem statement

Modern processors are vulnerable to **two** major classes of **attacks**



Conflict based attacks



Transient execution attacks

How can we create an **unified** solution to defend against both attack types keeping performance-security tradeoff in mind?

Prior Works

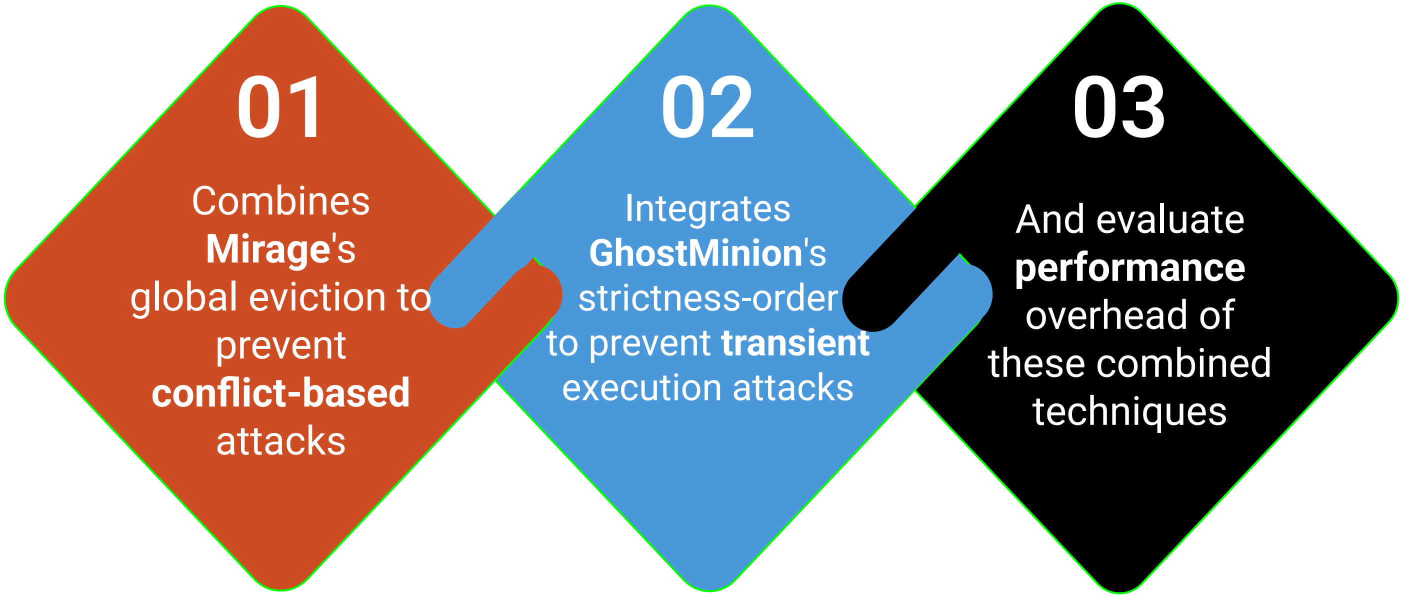
MIRAGE: Mitigates Conflict-Based Cache Attacks with a Practical Fully-Associative Design
USENIX Sec '21

GhostMinion: A Strictness-Ordered Cache System for Spectre Mitigation
MICRO '21

Combining them could provide comprehensive **security** but requires careful **integration**

Goal of the Project

Design a **hybrid cache architecture** that:



01

Combines **Mirage**'s global eviction to prevent **conflict-based** attacks

02

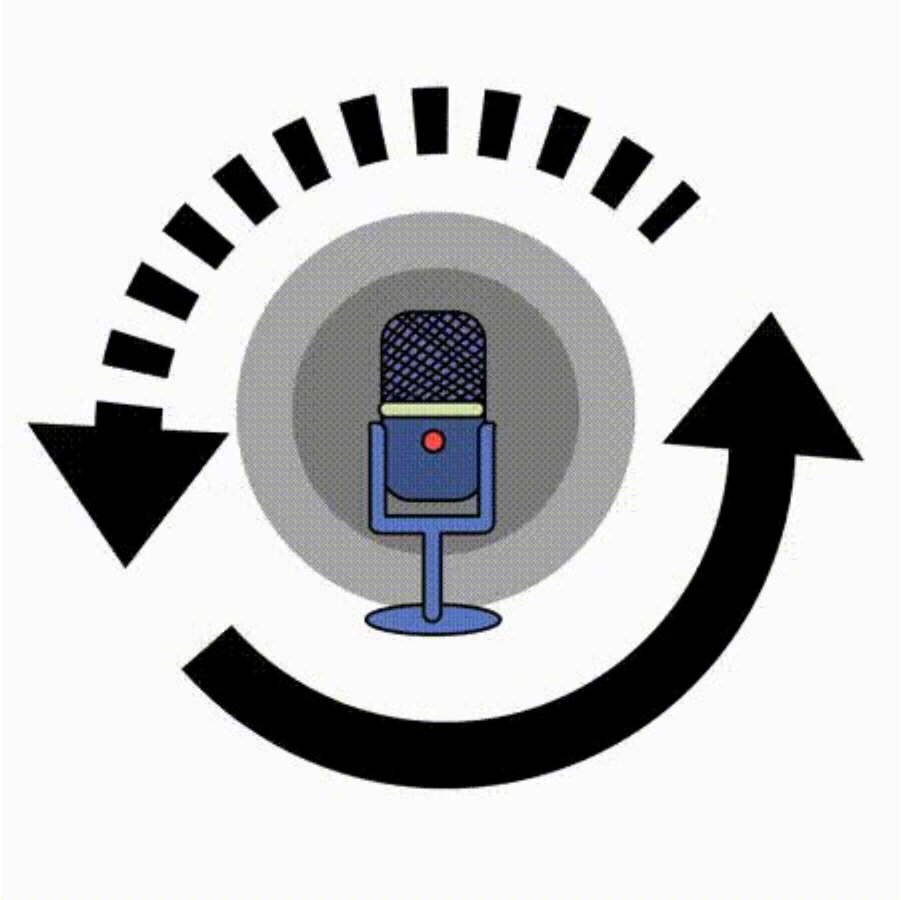
Integrates **GhostMinion**'s strictness-order to prevent **transient** execution attacks

03

And evaluate **performance** overhead of these combined techniques

Metrics of interest: **IPC, MPKI**

Recap of Checkpoint 1



Checkpoint 1: Challenges faced

- Version mismatch in gem5 version
    - Original Mirage in gem5 **v19** is not replicable
        - Ported to latest gem5 **v24**

    - Original GhostMinion => gem5 **v20**
        - Ported to gem5 **v24** but not compatible.
        - Sol: fallback to gem5 **v20**

    - Architecture mismatch
        - Original Mirage => **X86**; Original GhostMinion => **ARM**;
        - Ported GhostMinion to **X86** but not supported;
        - Finally, Ported Mirage to **ARM**.

- Running benchmark suite
    - Checkpointing is needed to reproduce results same as given in paper.

Checkpoint 1: Work done

- Setup **MIRAGE** artifact in gem5**v24**.
- Setup **GhostMinion** artifact in gem5**v20.1**.
- Evaluation of the techniques with 7 SPECspeed®2017Integer & 5 SPECspeed®2017Floating Point benchmarks.

Checkpoint 1: Simulation Configuration

Architecture: **ARM-64**

Core: **Single-Core**, 8-Wide, **Out-of-order**, 2.0GHz

L1D: 32KiB, 2-cycle-latency, 8 way, 4 MSHRs

L1I: 32KiB, 2-cycle-latency, 8 way, 4 MSHRs

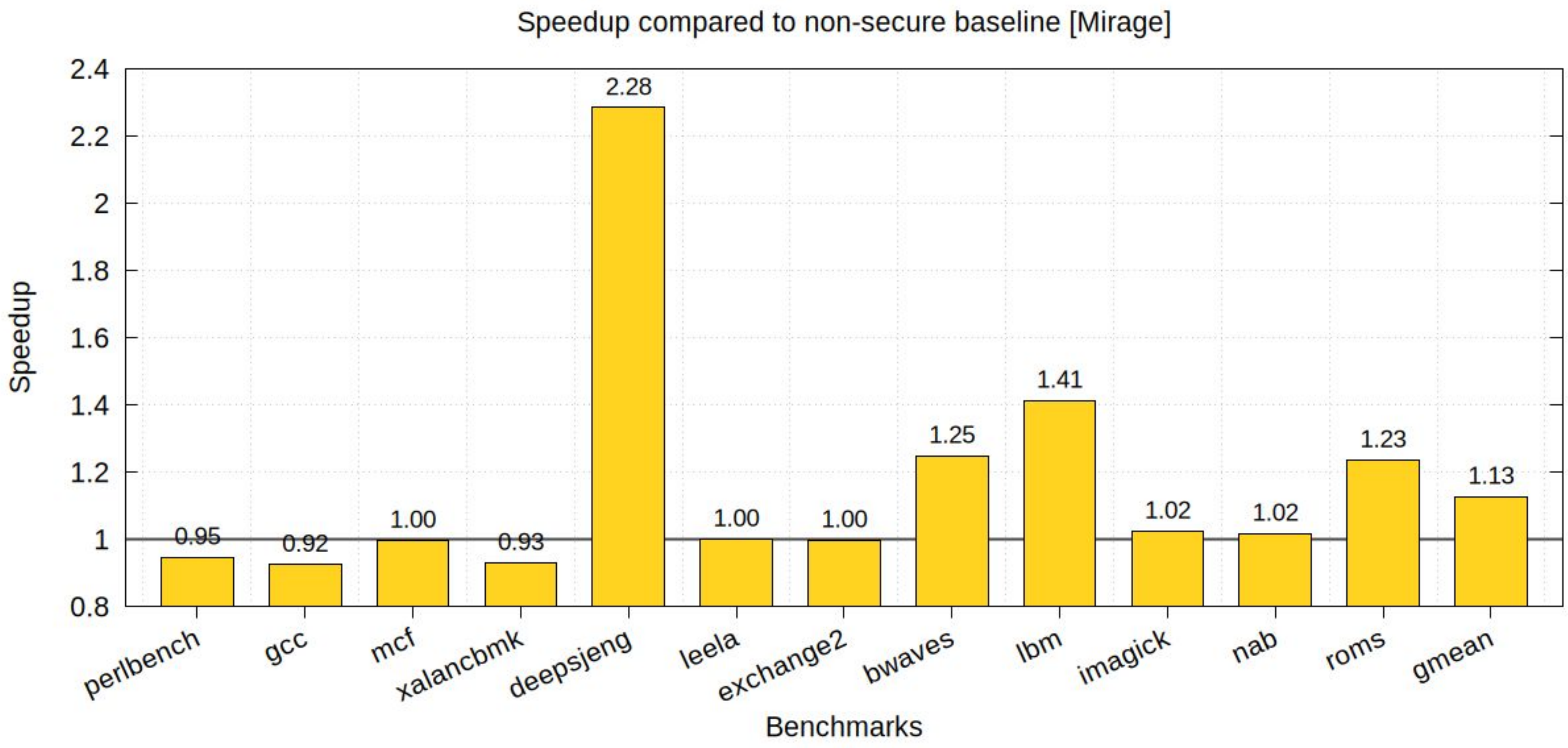
L2: 8MiB, 20-cycle-latency, 16 way, 20 MSHRs

DRAM: 8GiB DDR4 2400MT/s

Warmup-Instruction: **500M**

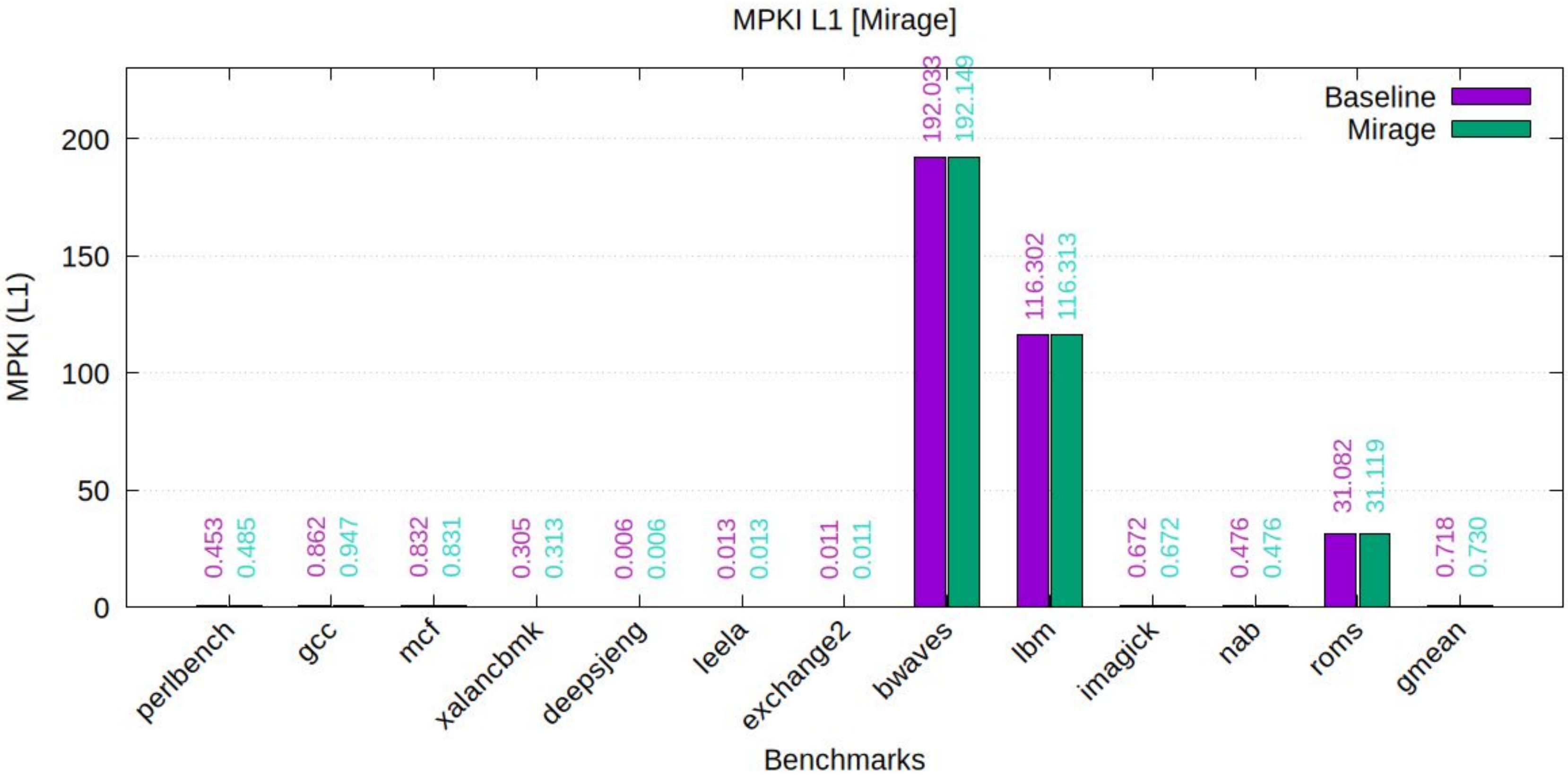
Simulation-Instruction: **500M**

MIRAGE Results - Speedup

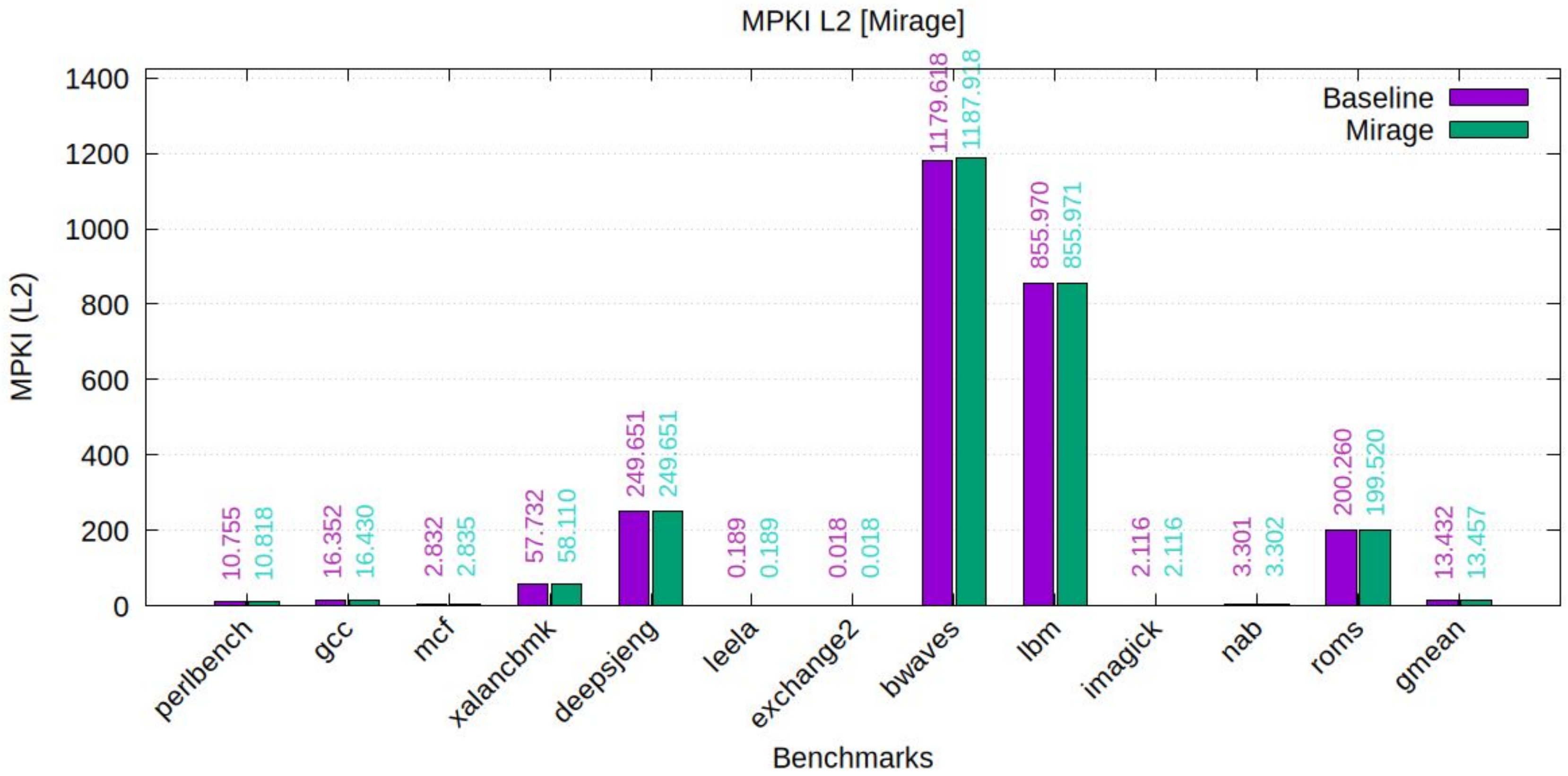


Why do we get a speedup in some cases?

MIRAGE Results - L1 MPKI



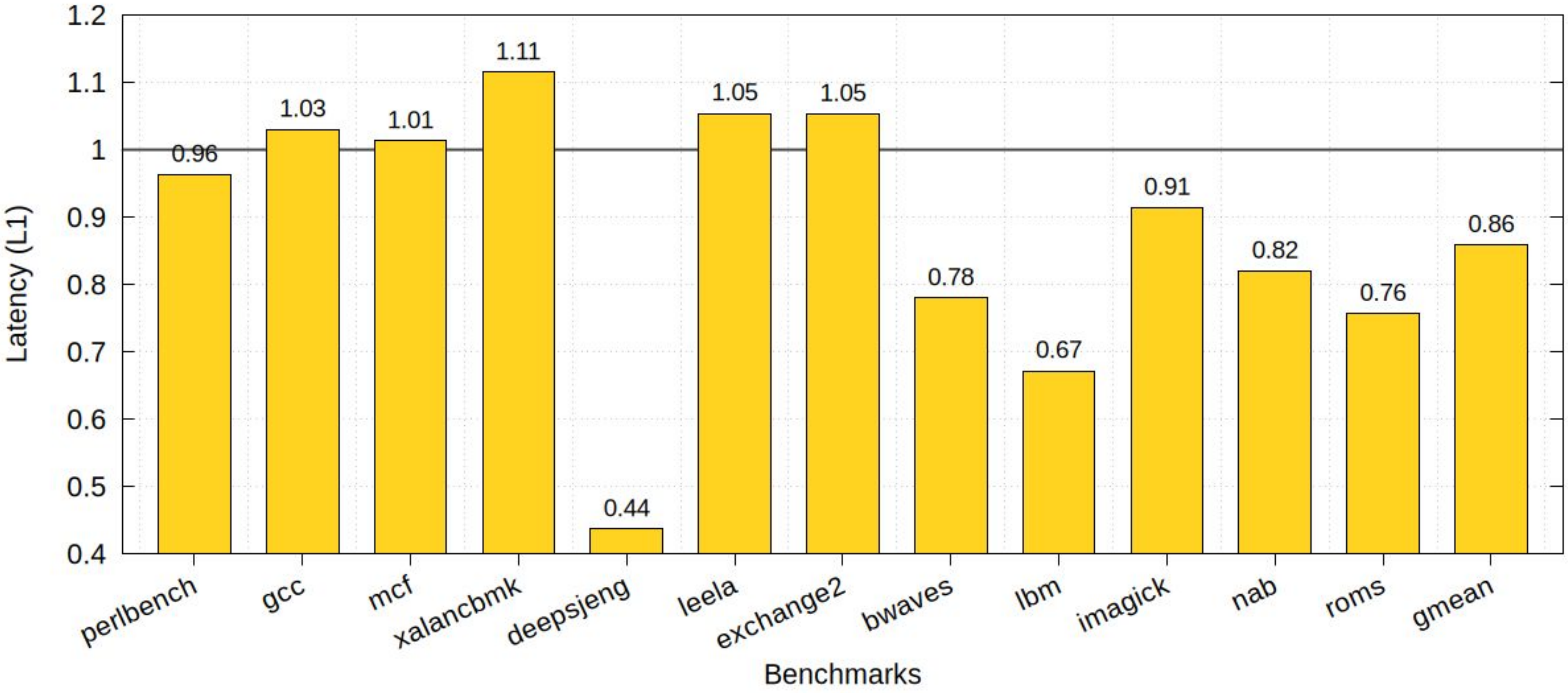
MIRAGE Results - L2 MPKI



MPKI doesn’t explain the performance difference

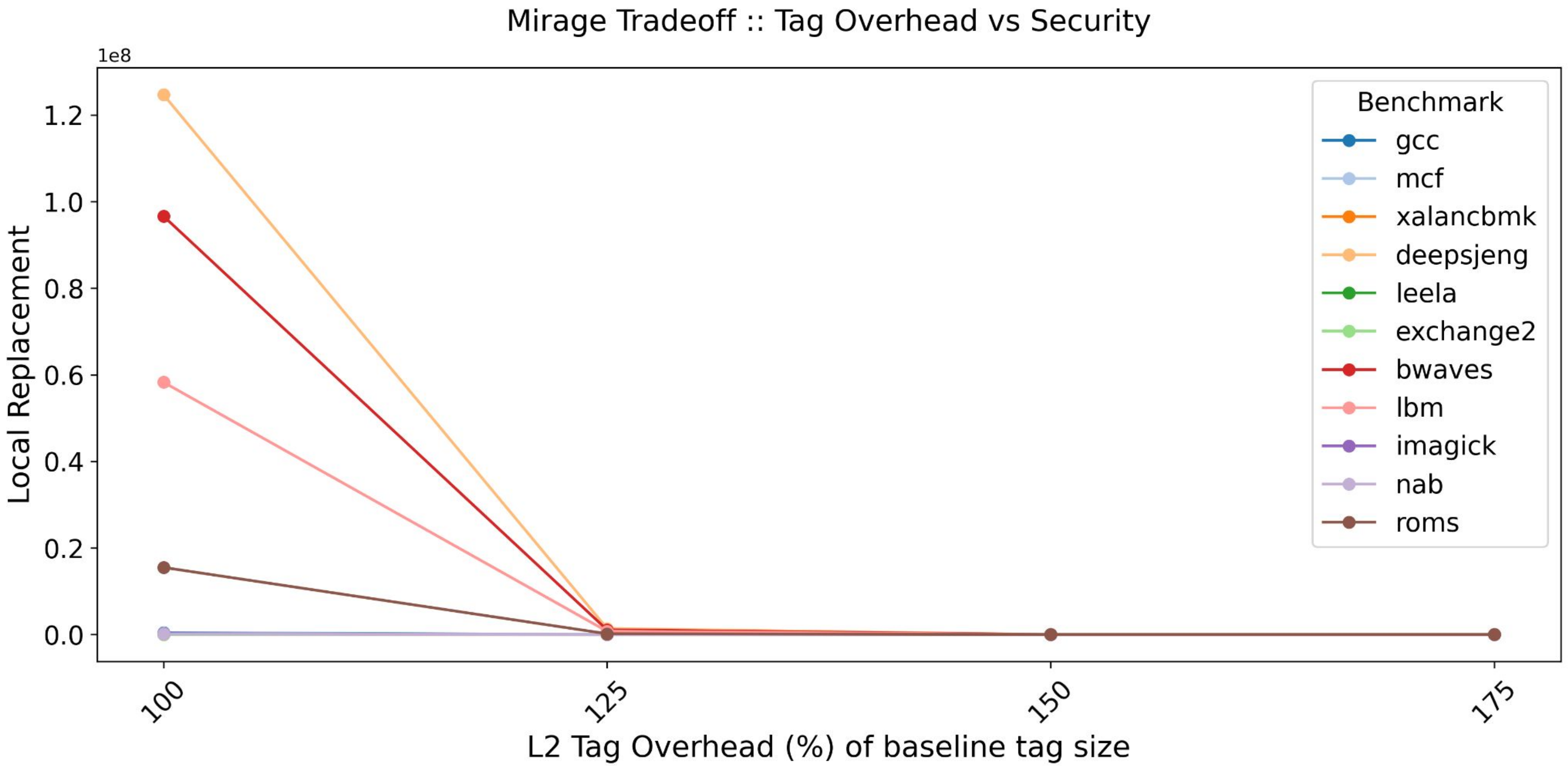
MIRAGE Results - L1 Miss Latency

L1 Miss Latency normalised to baseline



Significant differences in miss latency between baseline and MIRAGE

MIRAGE Results - Tradeoff



Set-Associative Eviction decreases as extra tags are used

GhostMinion Configuration

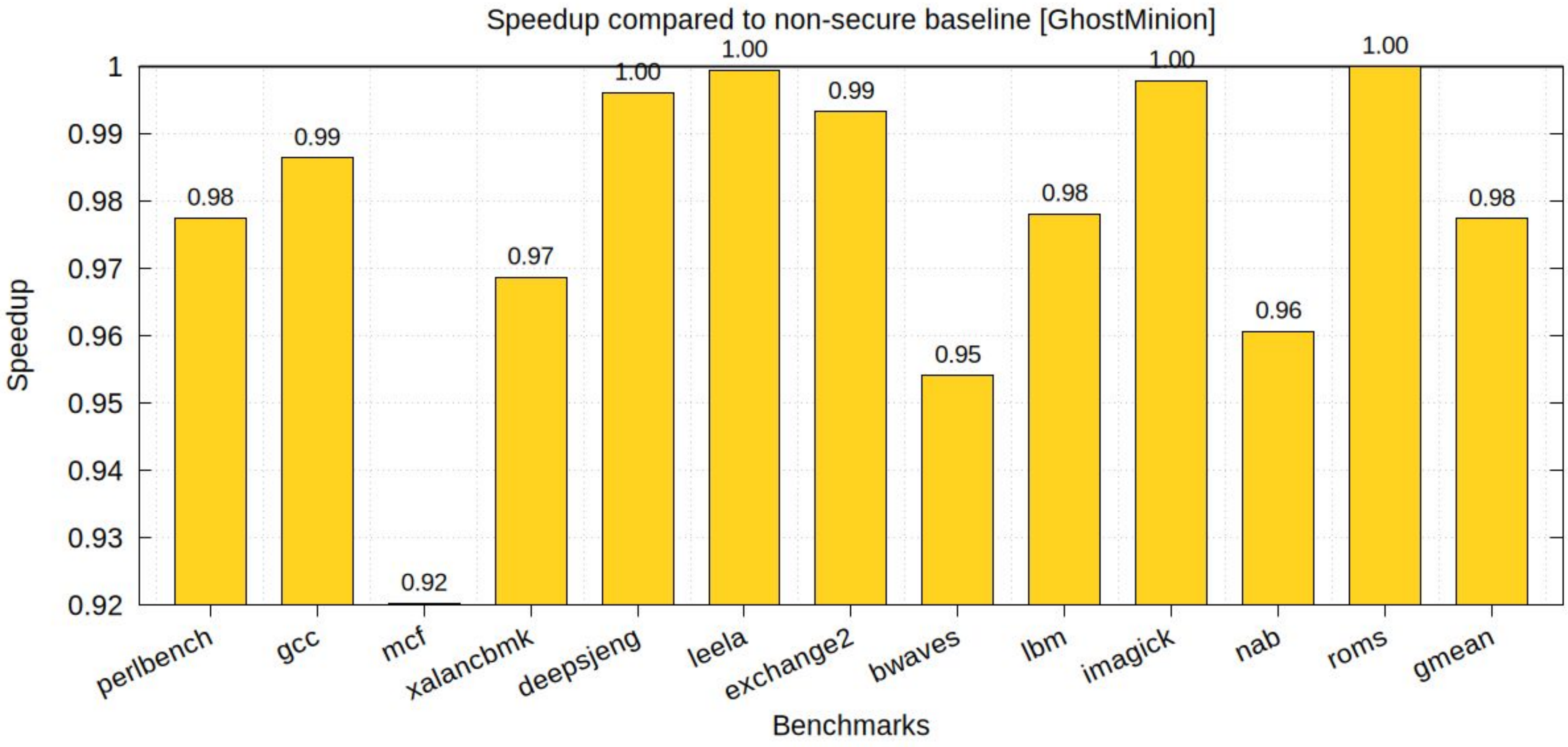
Level: L1-D Cache, L1-I Cache

L2 clusivity: Exclusive

D/I GhostMinions: 2KiB, 2-way, accessed with D/I cache

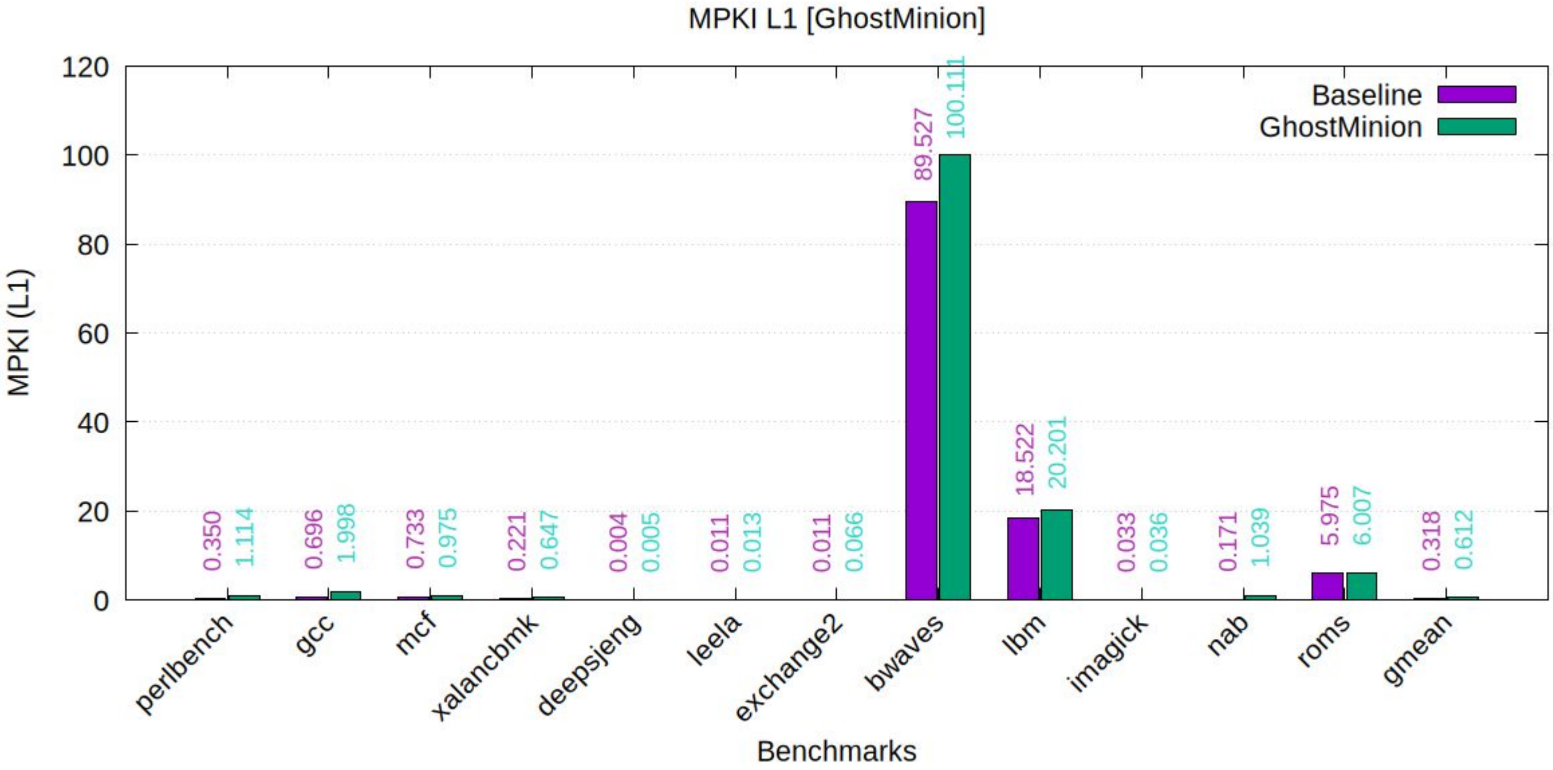
L2 prefetcher: 8 degree, Stride Prefetcher (64-entry RPT)

GhostMinion Results - Speedup



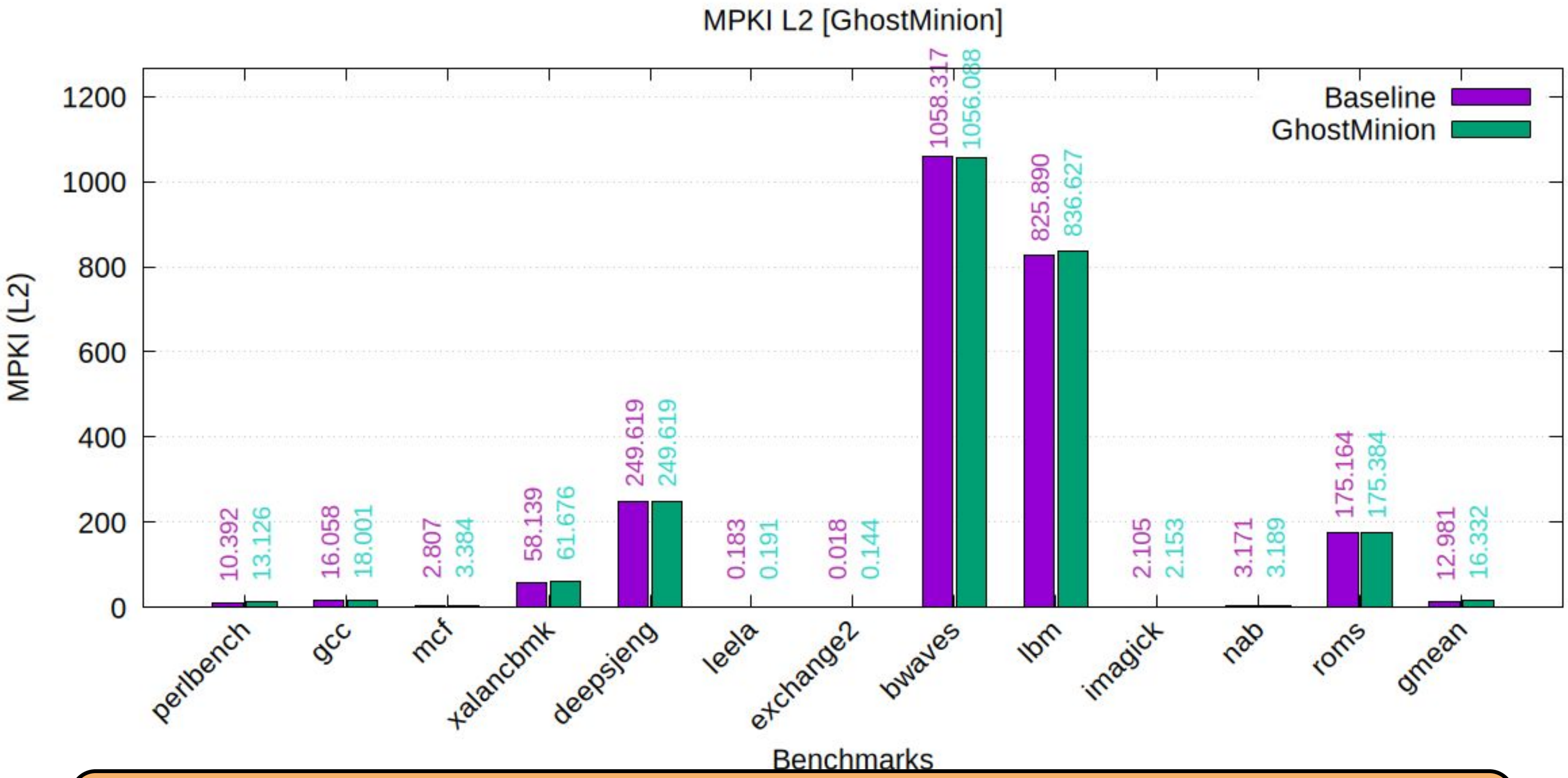
Speculative data hiding and strict-ordering causes commit stall

GhostMinion Results - L1 MPKI



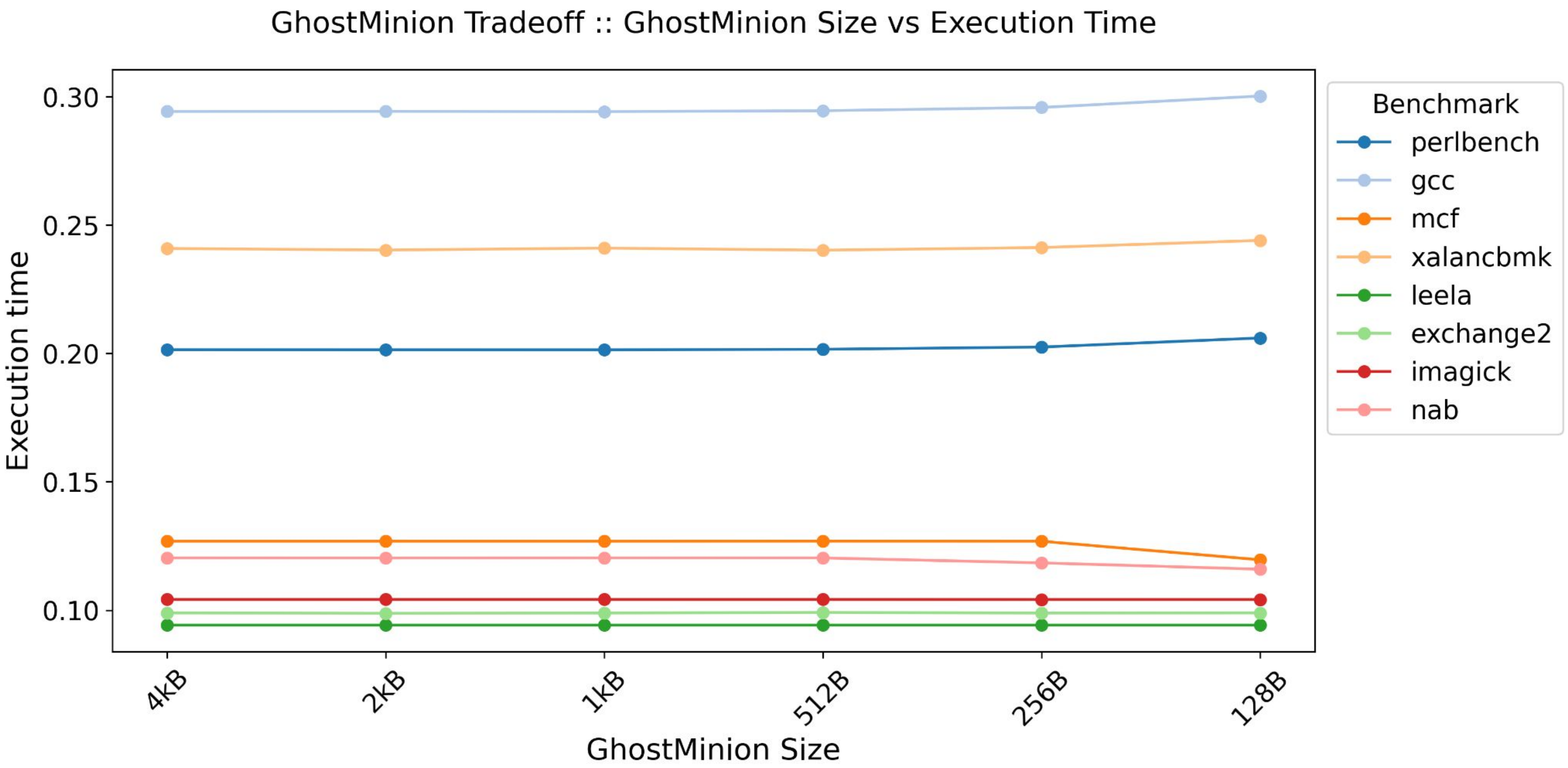
Increased MPKI in non-speculative L1 cache

GhostMinion Results - L2 MPKI



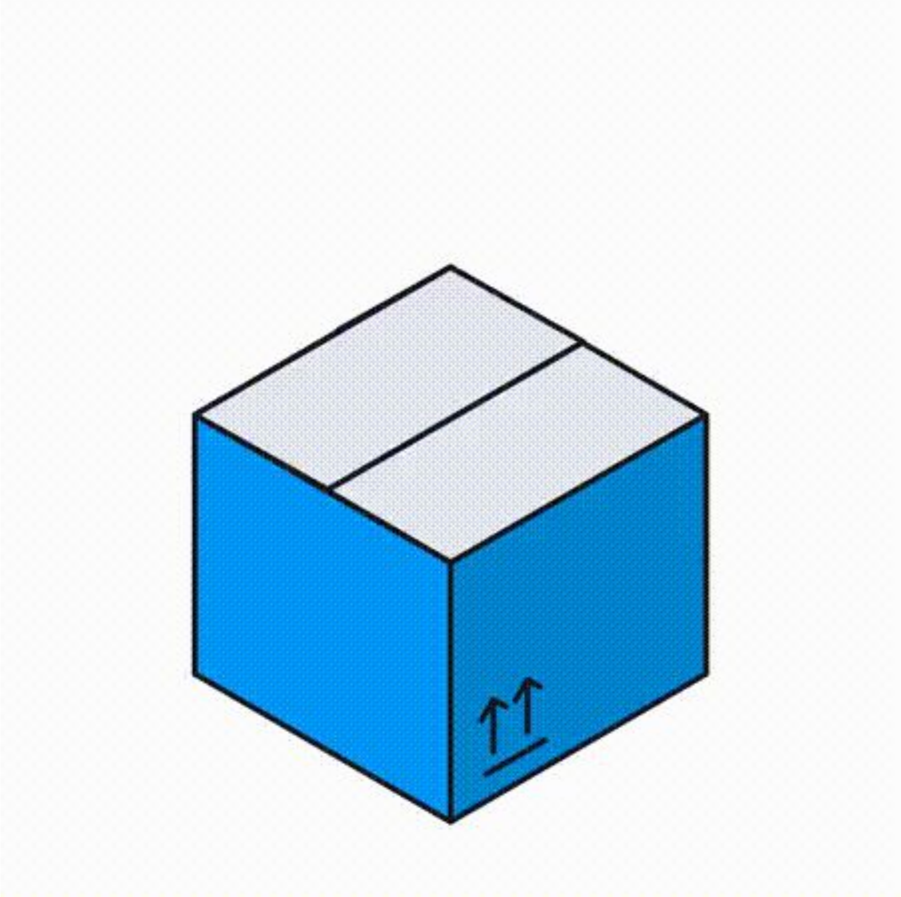
No noticeable differences in L2 MPKI for baseline and GhostMinion

GhostMinion Results: Tradeoff



Insignificant effect on performance on changing GhostMinion size

Entering Checkpoint 2



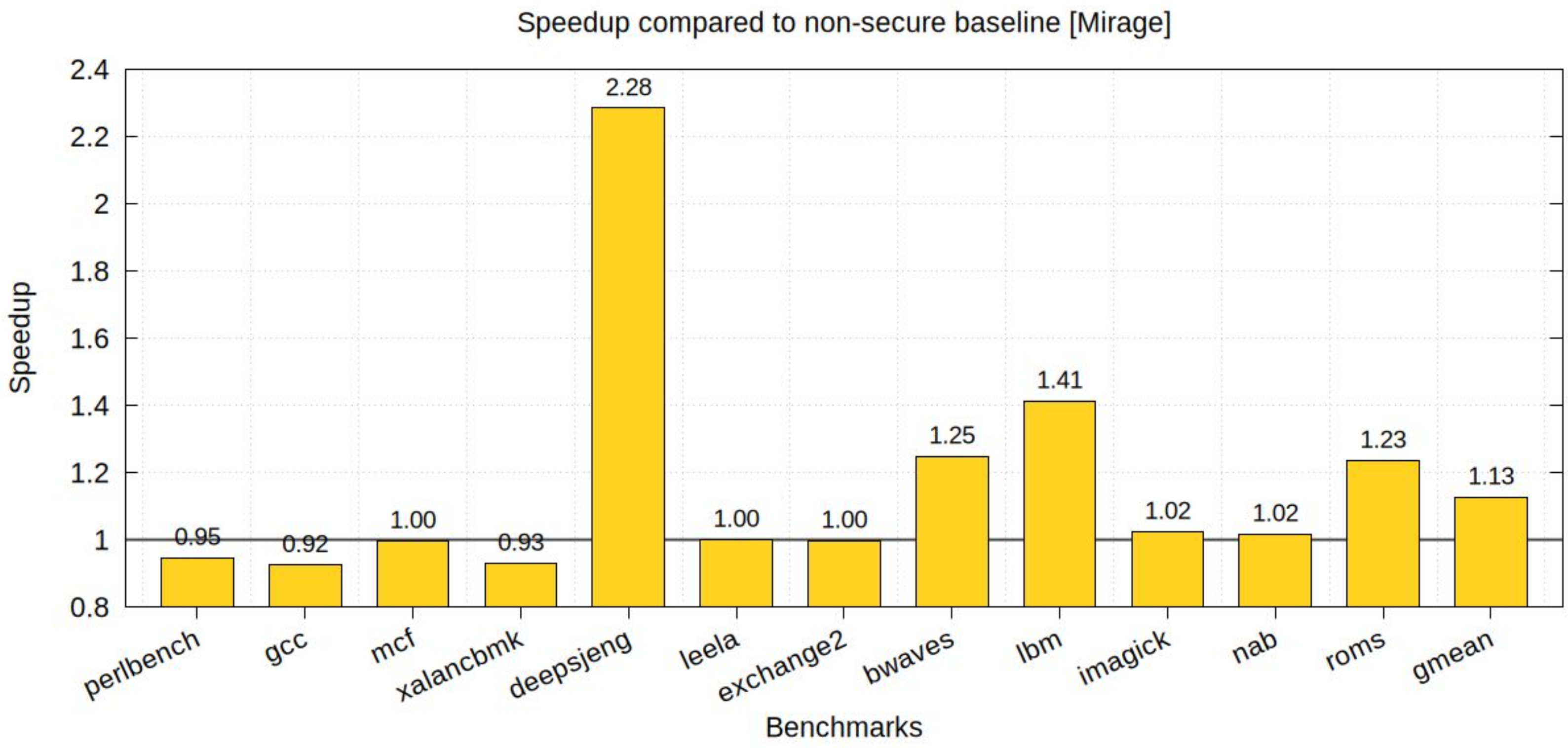
Action on checkpoint 1 feedback

Feedback Take care of some unexpected results

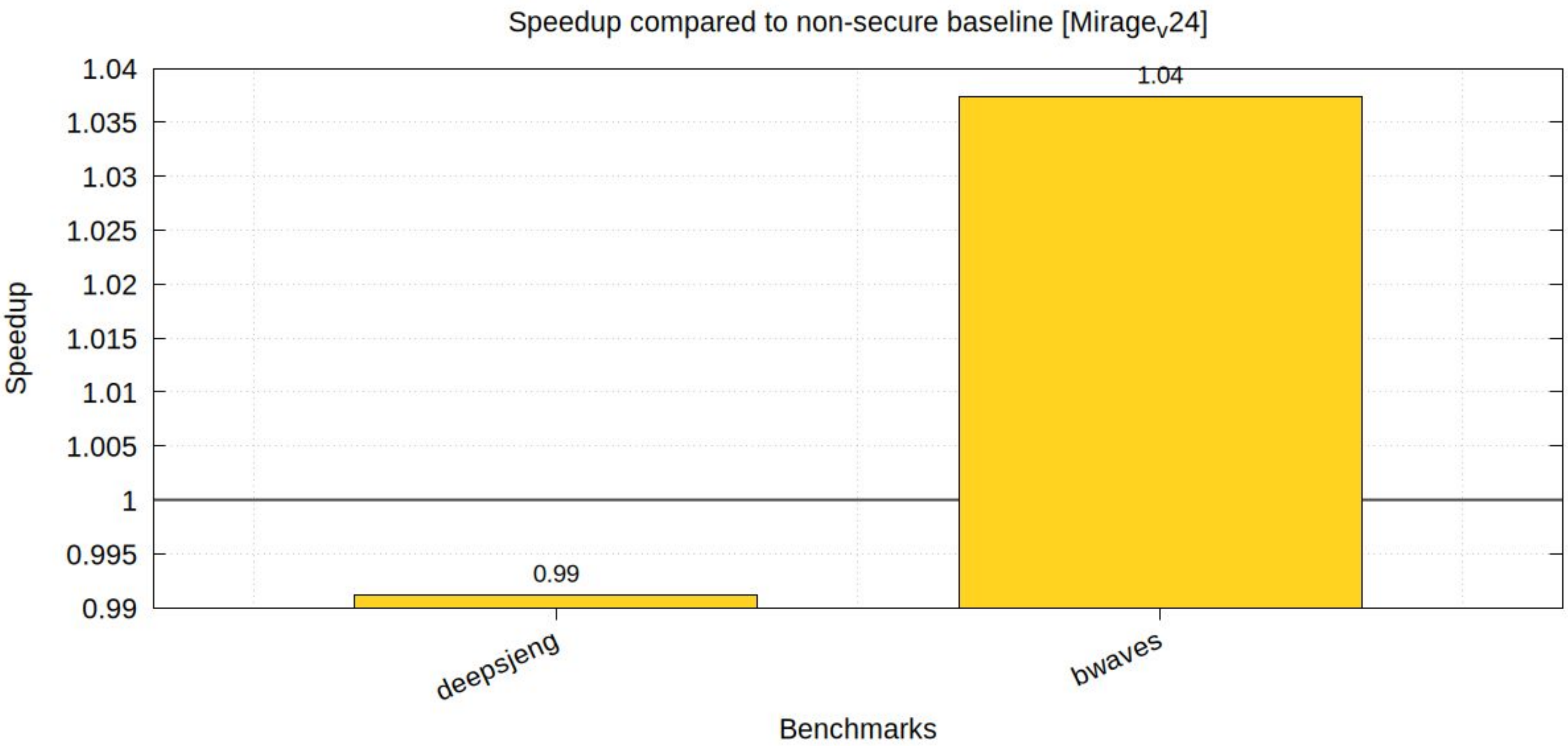
Action

- Took checkpoint at 10B instruction
- Re-evaluated benchmarks for larger instruction count from checkpoint

MIRAGE Results Previous - Speedup



Revised result Revised - Speedup



Work done in checkpoint-II

- Integrated MIRAGE & GhostMinion in gem5 v20.1
- Implemented checkpointing system to skip first 10B instruction and simulation thereafter.

Configuration in merged technique

- DRAM page management policy
  - GhostMinion: Open adaptive
  - Mirage: Closed
  - Combined: Open adaptive

- L2 clusivity
  - GhostMinion: exclusive
  - Mirage: Inclusive
  - Combined: Inclusive (exclusive w.r.t ghostminion cache)

Revised Simulation Configuration

Architecture: **ARM-64**

Core: **Single-Core**, 8-Wide, **Out-of-order**, 2.0GHz

L1D: 32KiB, 2-cycle-latency, 8 way, 4 MSHRs

L1I: 32KiB, 2-cycle-latency, 8 way, 4 MSHRs

L2: 8MiB, 20-cycle-latency, 16 way, 20 MSHRs

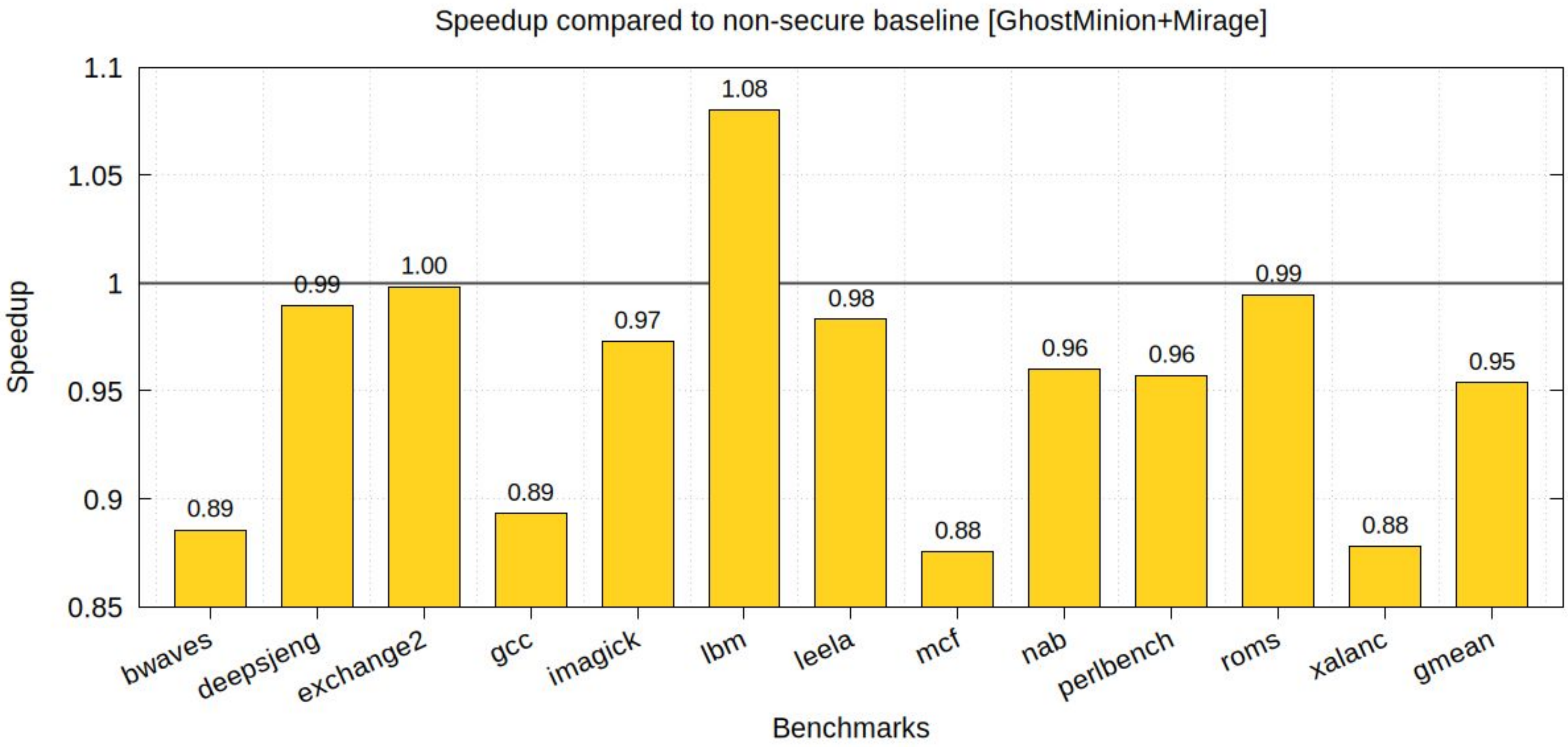
DRAM: 16GiB DDR4 2400MT/s

Warmup-Instruction: **500M**

Simulation-Instruction: **2B***

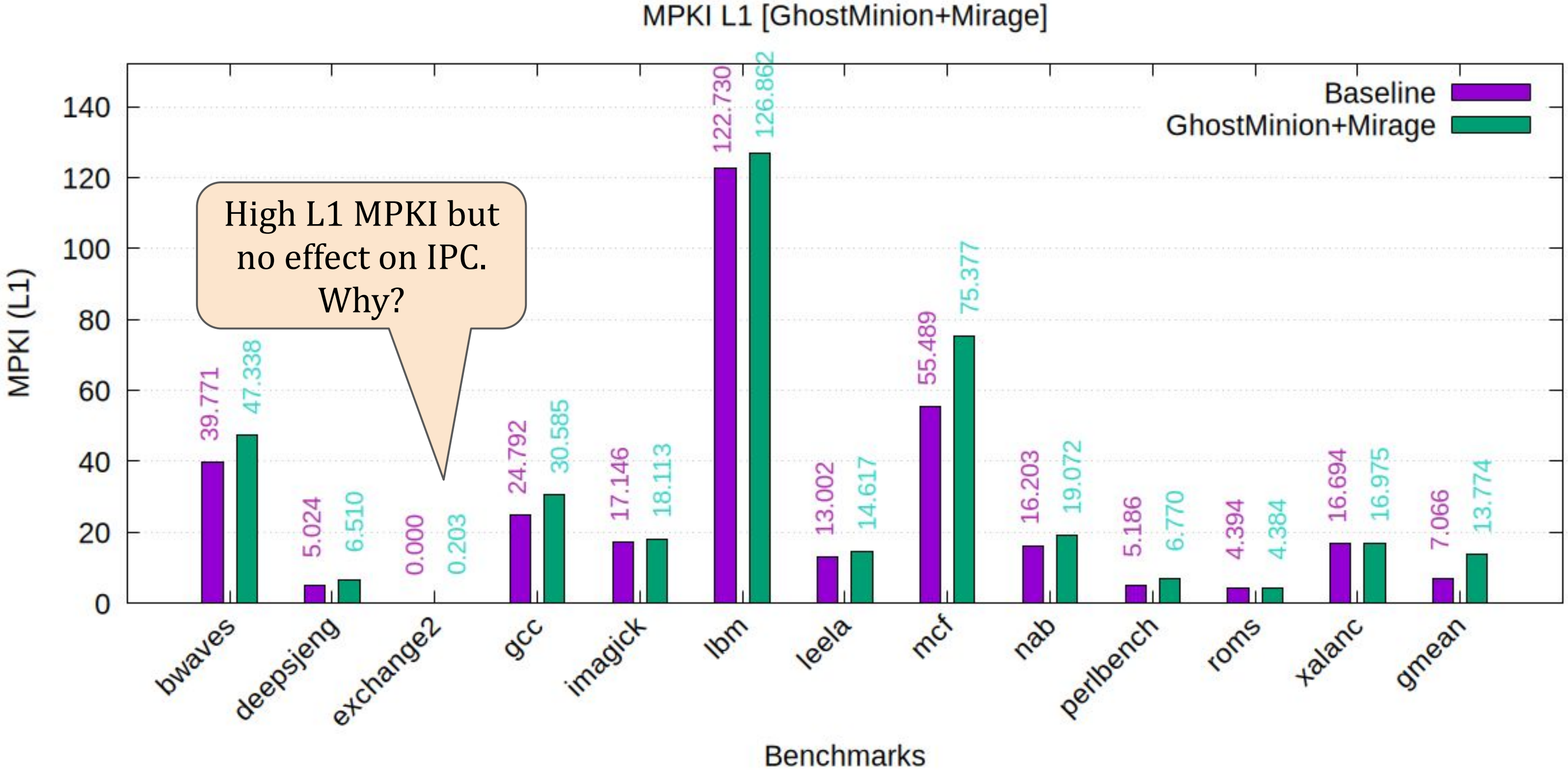
*initially tried with 10B instructions but each benchmark took about 45hrs

Results: Speedup



The combined technique exhibit a 5% slowdown on average

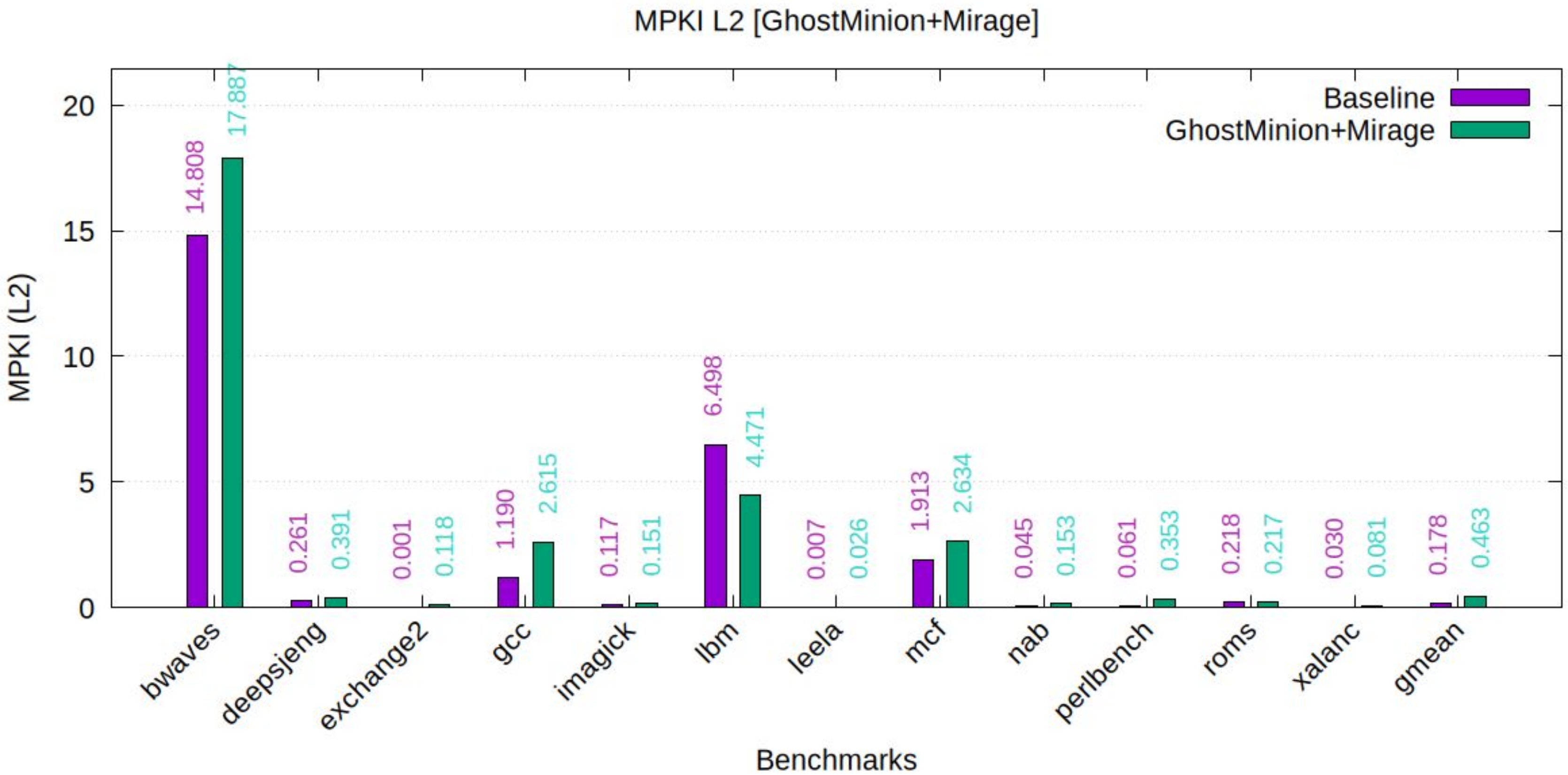
Results - L1 MPKI



High L1 MPKI but no effect on IPC. Why?

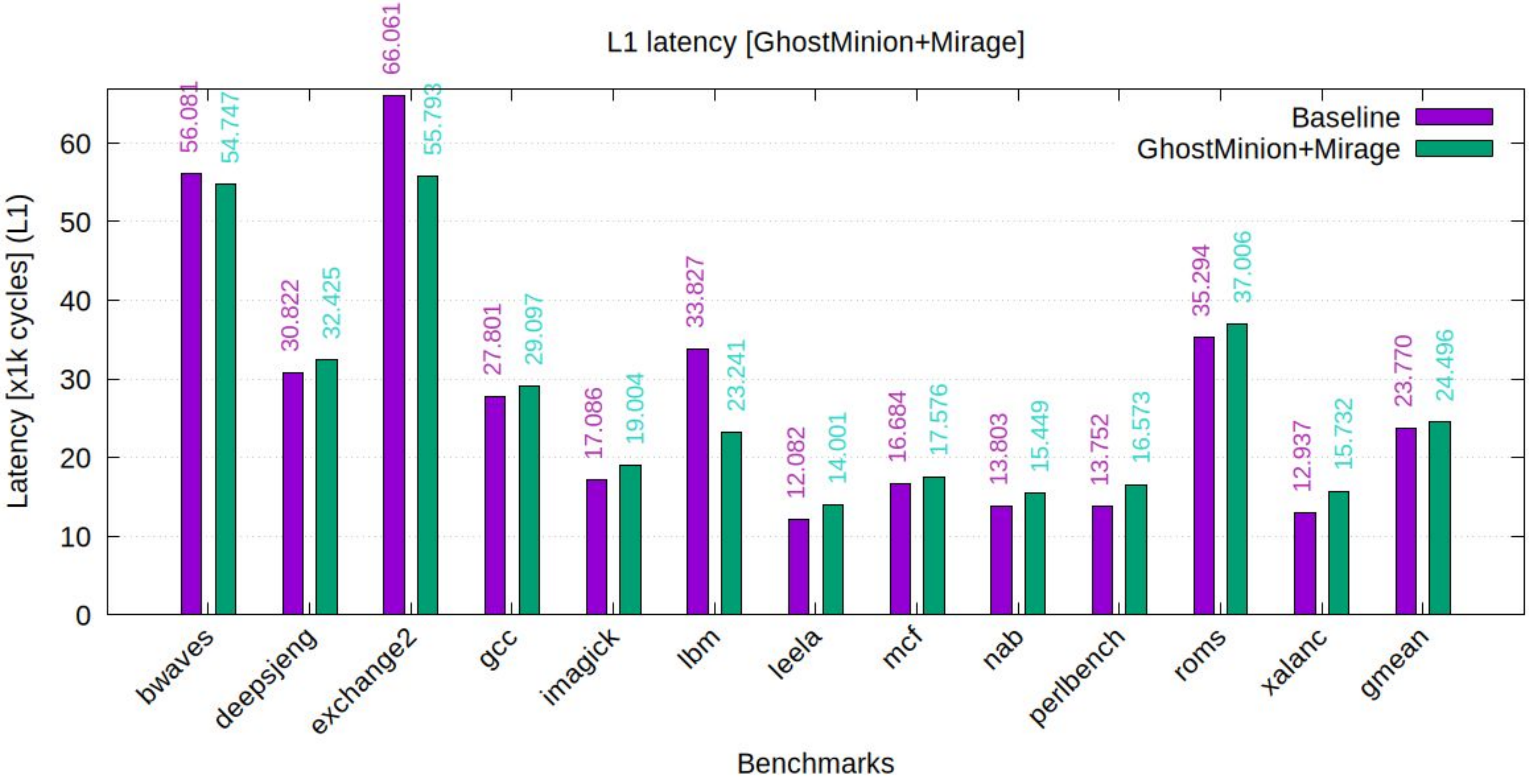
bwaves, gcc, mcf, xalanc suffers high MPKI at both L1 and L2

Results - L2 MPKI



Slight speedup in lbm due to L2 MPKI lower than baseline

Results - L1 Miss Latency



Both lbm and exchange2 encounter significantly lower miss latency

Github link

- https://github.com/sammagnet7/cs773_CompArch-Perf-Security

Final Checkpoint Video

- https://www.youtube.com/watch?v=IM6voFCyAXU

MIRAGE Configuration

Level: L2

L2 clusivity: Inclusive

L2 Skews: 2

Tag to Data Ratio: 1.75 (75% extra tags)

Encryption Latency: 2 cycles