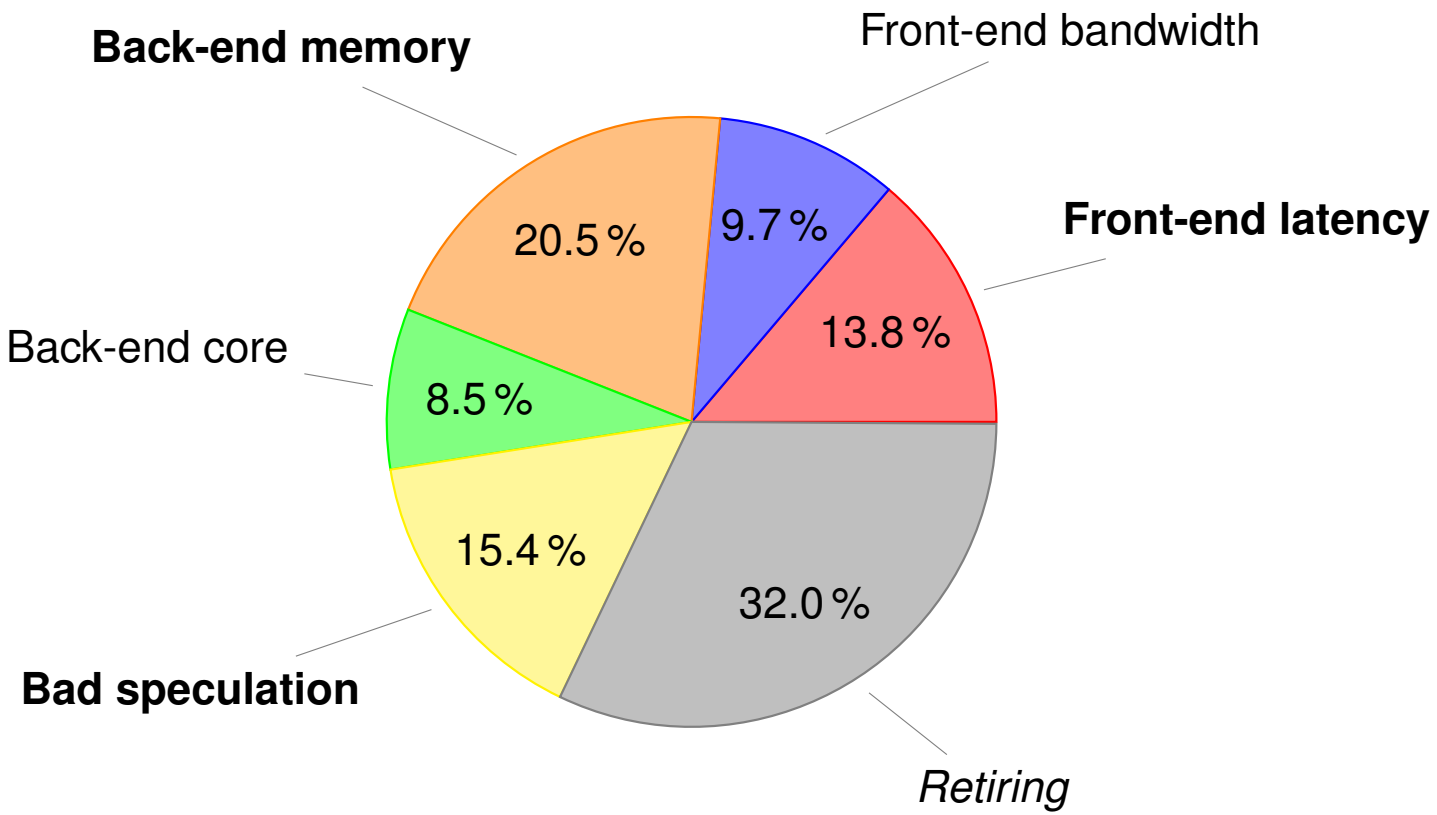
Intelligent Memory Prefetching and Prediction in Modern Processors

Alberto Ros

University of Murcia, Spain

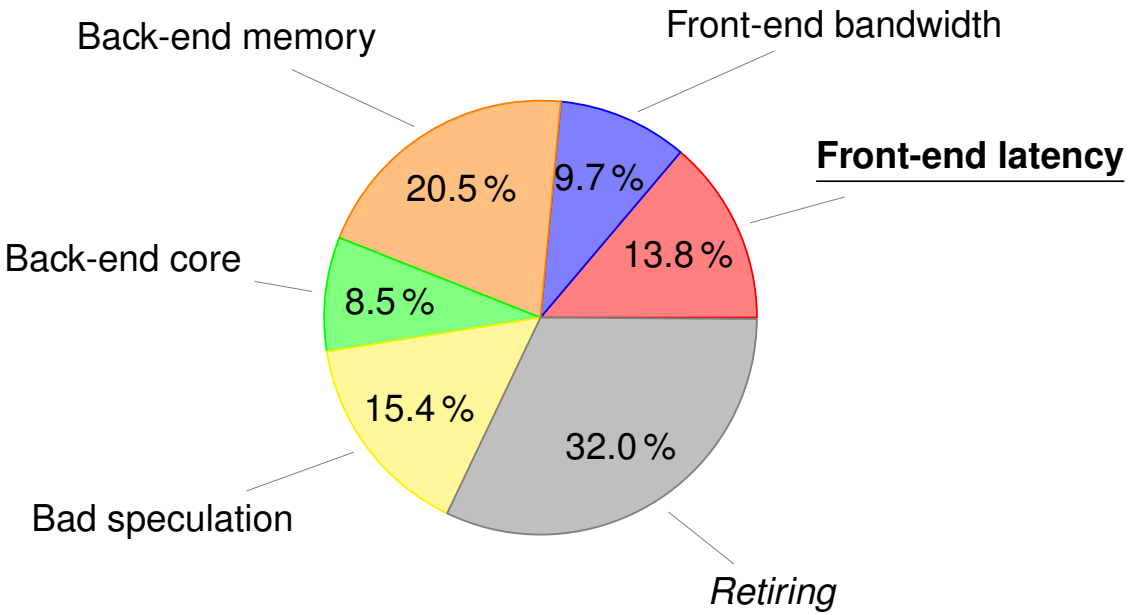
May 20, 2026

FACT: SINGLE-THREAD PERFORMANCE MATTERS



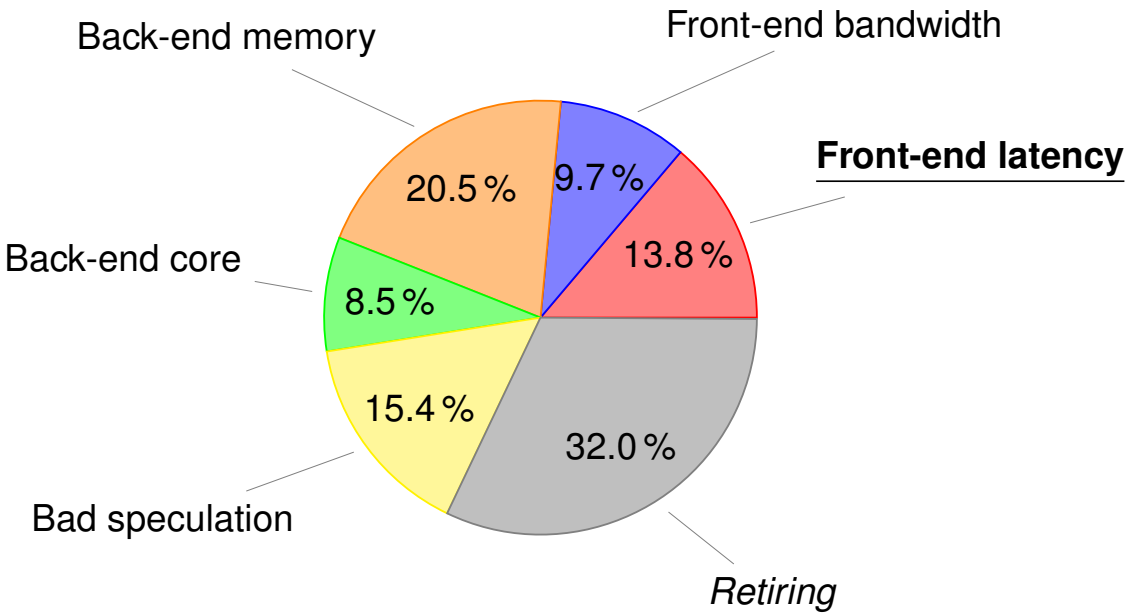
Data source: Ayers et al. *AsmDB: Understanding and Mitigating Front-End Stalls in Warehouse-Scale Computers*, ISCA 2019.

INTELLIGENT PREDICTIONS IS ALL YOU NEED



- Dominated by instruction cache (L1I) misses
  - Server and cloud apps getting larger, far from fitting in L1I
  - Hiting in the L2 or L3
- Critical! Processors need to keep the pipeline full

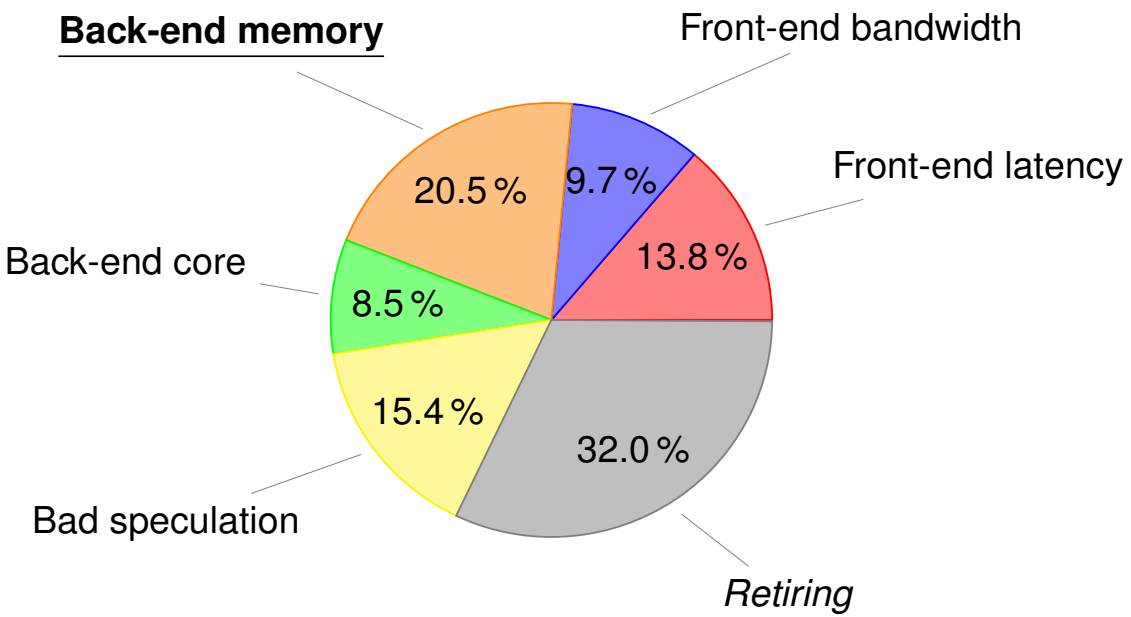
INTELLIGENT PREDICTIONS IS ALL YOU NEED



- Dominated by instruction cache (L1I) misses
  - Server and cloud apps getting larger, far from fitting in L1I
  - Hiting in the L2 or L3
- Critical! Processors need to keep the pipeline full

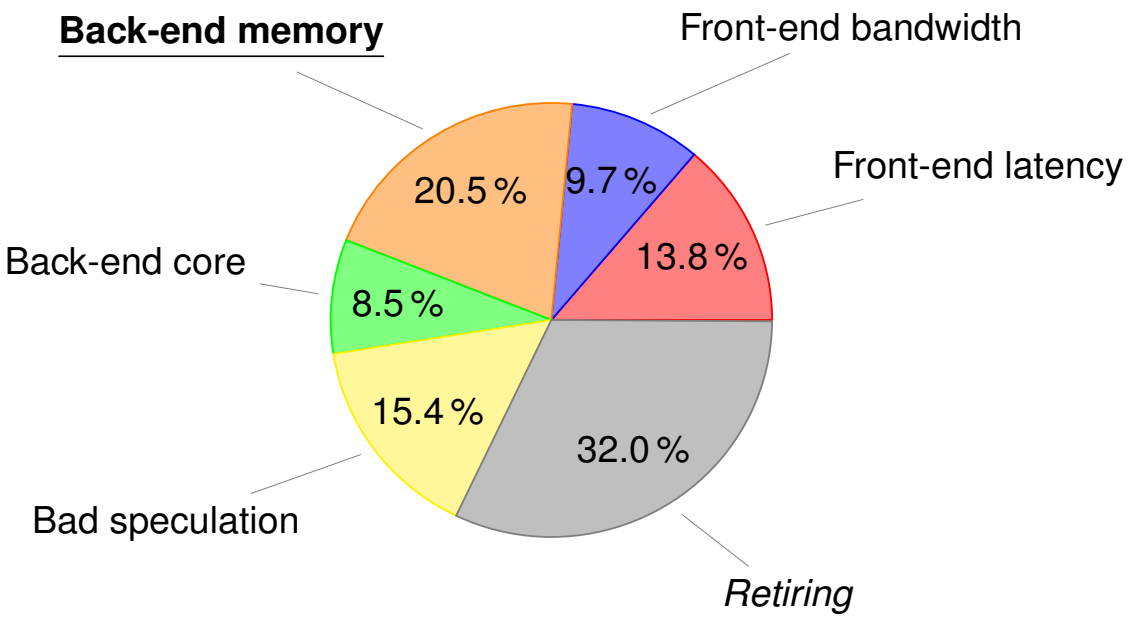
Solution: Instruction prefetching and criticality aware replacement

INTELLIGENT PREDICTIONS IS ALL YOU NEED



- Due to data cache (L1D) misses
  - Many of them reaching main memory
- Cause stalls and late detection of bad speculation

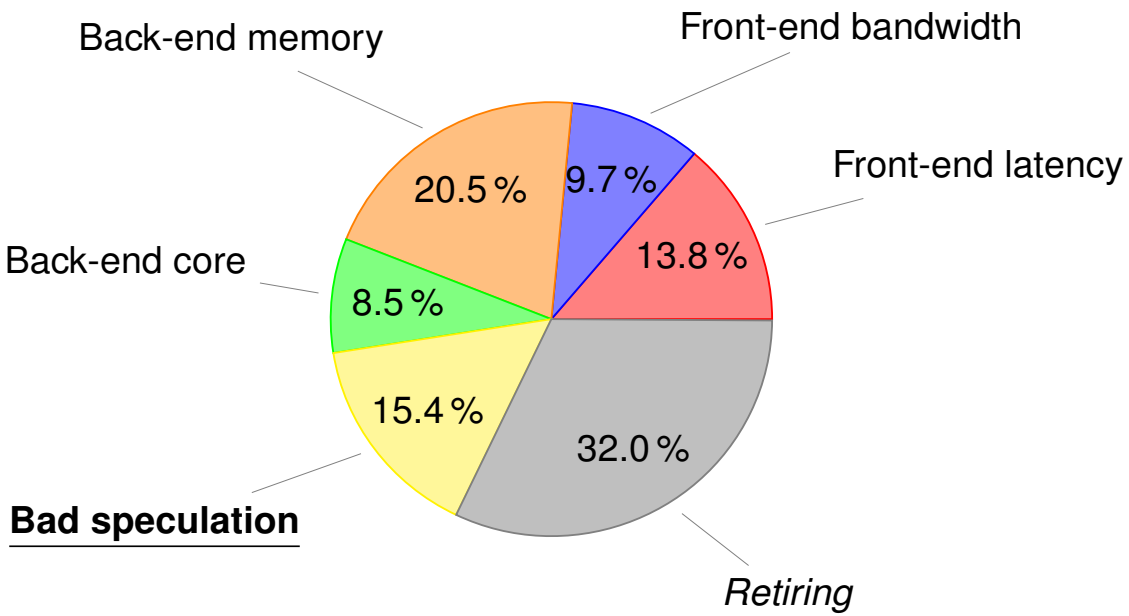
INTELLIGENT PREDICTIONS IS ALL YOU NEED



- Due to data cache (L1D) misses
  - Many of them reaching main memory
- Cause stalls and late detection of bad speculation

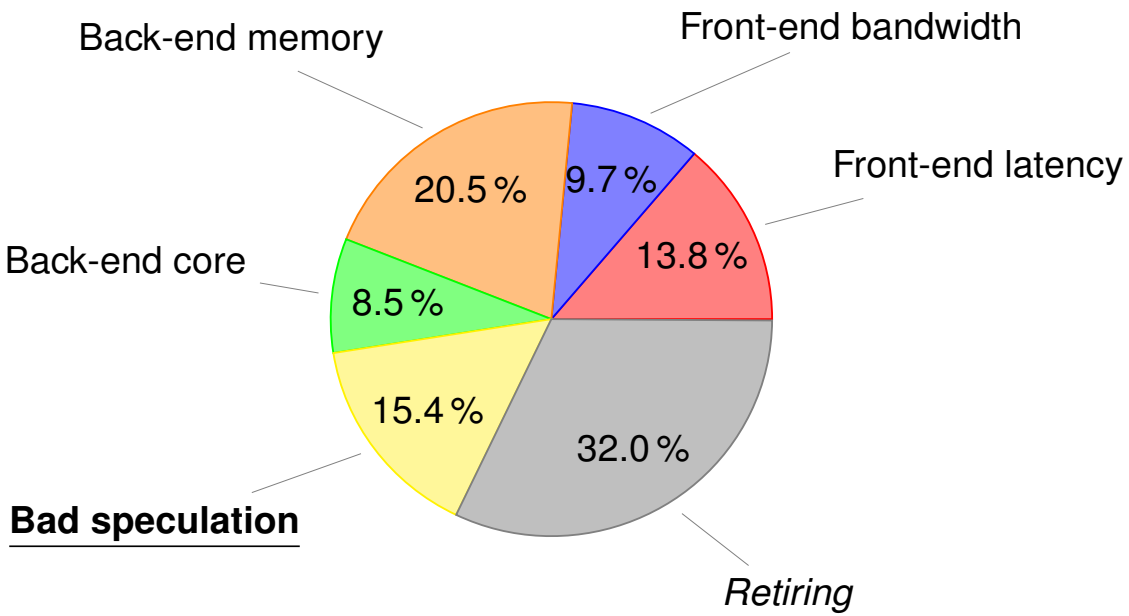
Solution: Timely data prefetching

INTELLIGENT PREDICTIONS IS ALL YOU NEED



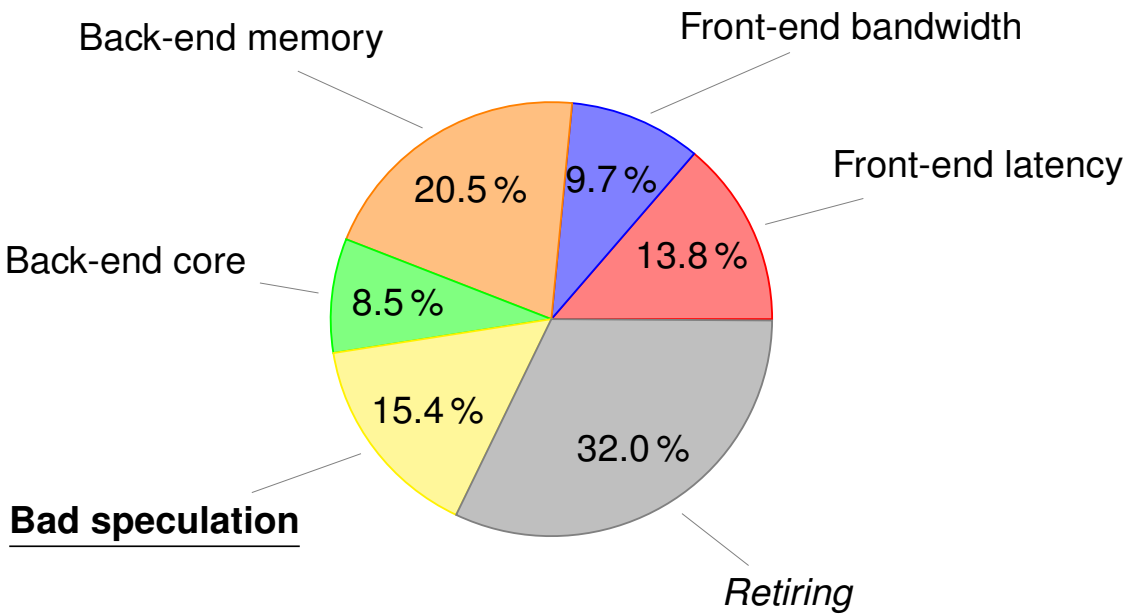
- Memory dependence prediction
- Branch prediction

INTELLIGENT PREDICTIONS IS ALL YOU NEED



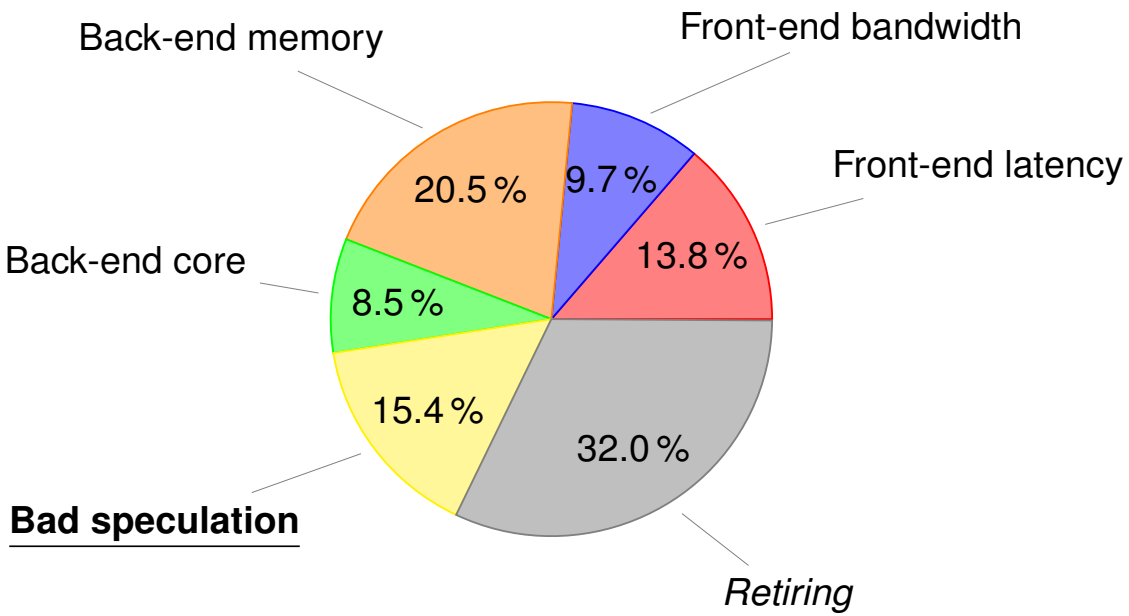
- Memory dependence prediction
  - More critical as the pipeline depth increases
- Branch prediction

INTELLIGENT PREDICTIONS IS ALL YOU NEED



- Memory dependence prediction
  - More critical as the pipeline depth increases
- Branch prediction
  - Hard to optimize! $\Rightarrow$ But still room for improvement

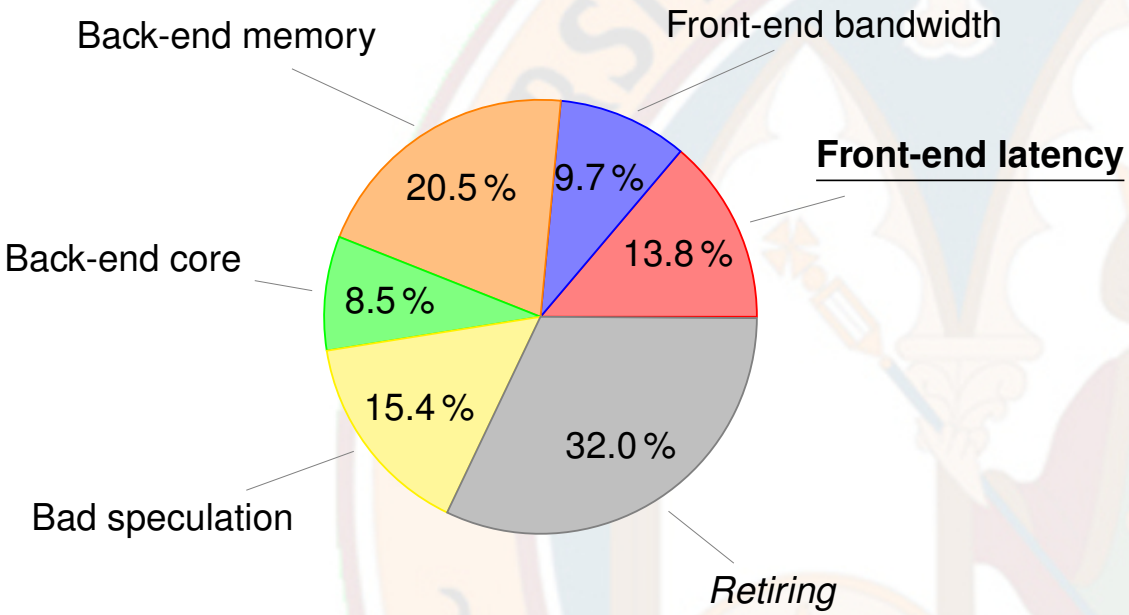
INTELLIGENT PREDICTIONS IS ALL YOU NEED



- Memory dependence prediction
  - More critical as the pipeline depth increases
- Branch prediction
  - Hard to optimize! $\Rightarrow$ But still room for improvement

Solution: Better context information and wrong-path prefetching

Addressing front-end latency (Part 1)



Overview

- Latency more is important than bandwidth at the front end
- Prefetching and keeping instructions near the core is fundamental for performance

Overview

- Latency more is important than bandwidth at the front end
- Prefetching and keeping instructions near the core is fundamental for performance
- **Solutions**
  - Prefetching instructions to L1I
    - $\Rightarrow$ The Entangling instruction prefetcher (code available[1])

[1] https://github.com/alberto-ros/EntanglingInstructionPrefetcher

Overview

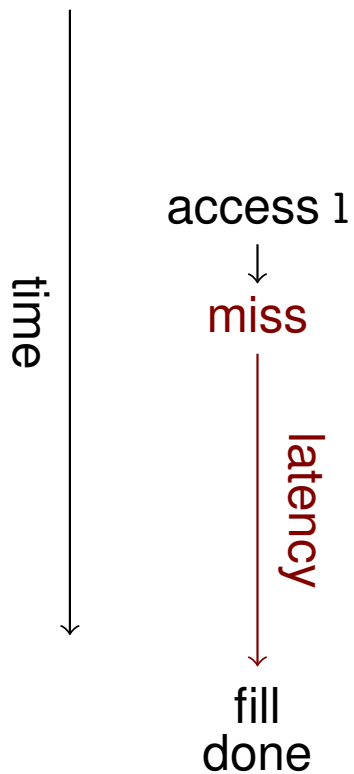
- Latency more is important than bandwidth at the front end
- Prefetching and keeping instructions near the core is fundamental for performance
- **Solutions**
  - Prefetching instructions to L1I
    - ⇒ The Entangling instruction prefetcher (code available[1])
  - Keeping critical instructions at L2
    - ⇒ ICARUS: Criticality and Reuse based Instruction Caching

[1] https://github.com/alberto-ros/EntanglingInstructionPrefetcher

THE ENTANGLING INSTRUCTION PREFETCHER

CONCEPT OF ENTANGLED PAIRS

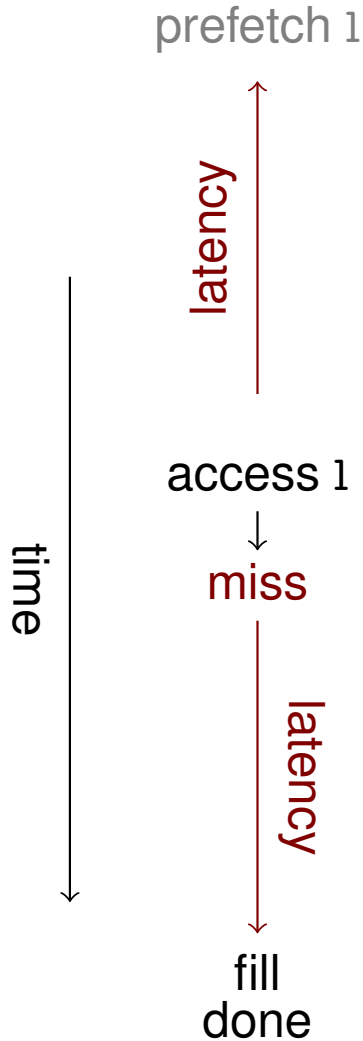
Ros & Jimborean [ISCA'21]



The Entangling Instruction Prefetcher

Concept of entangled pairs

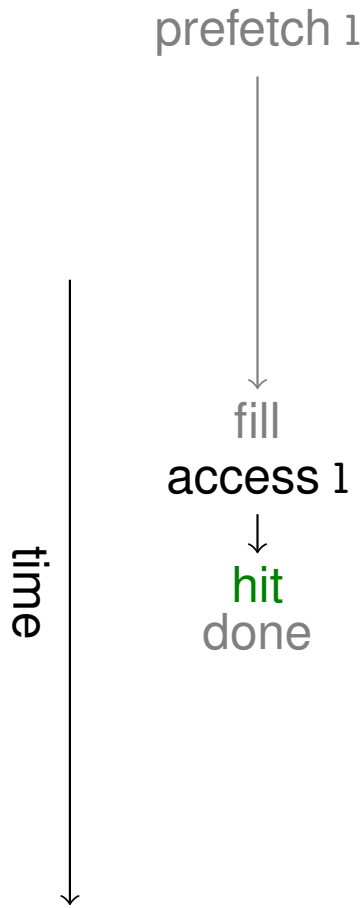
Ros & Jimborean [ISCA'21]



The Entangling Instruction Prefetcher

Concept of entangled pairs

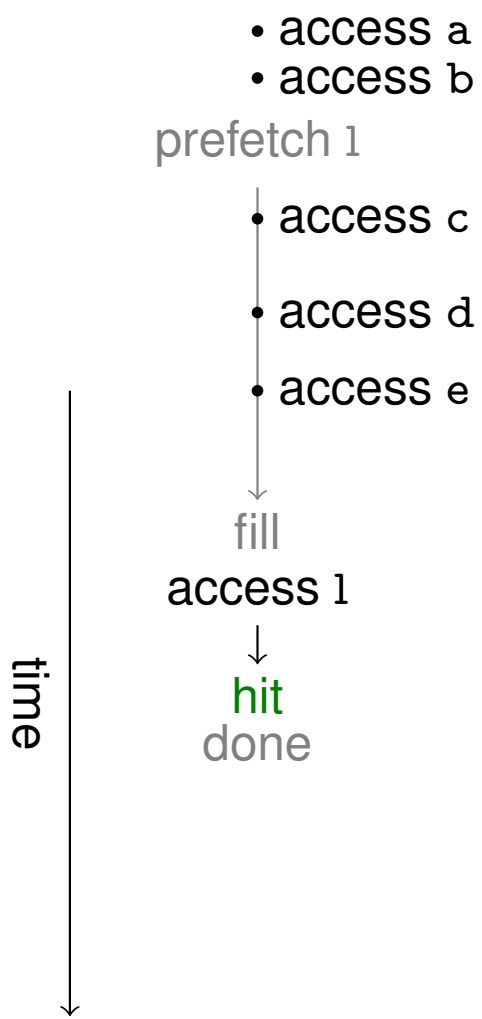
Ros & Jimborean [ISCA'21]



The Entangling Instruction Prefetcher

Concept of entangled pairs

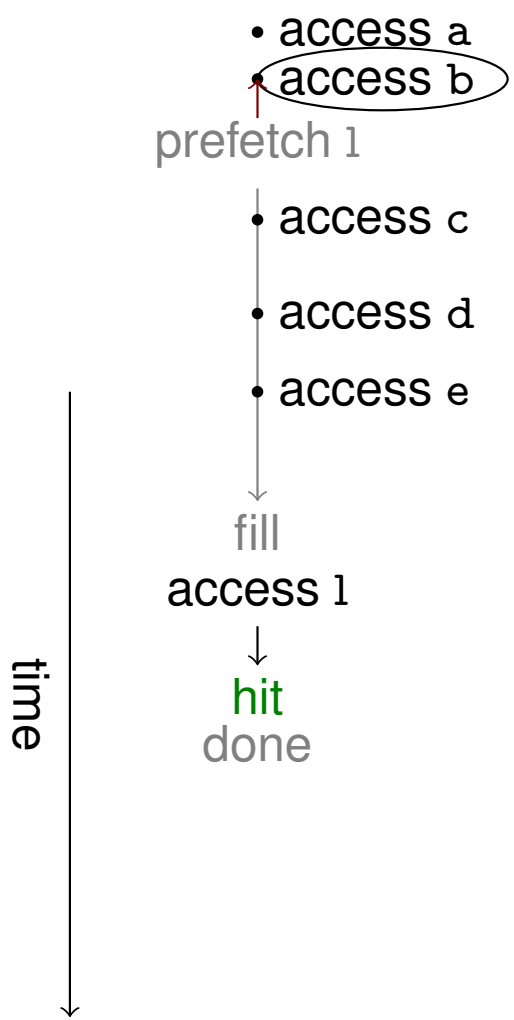
Ros & Jimborean [ISCA'21]



The Entangling Instruction Prefetcher

Concept of entangled pairs

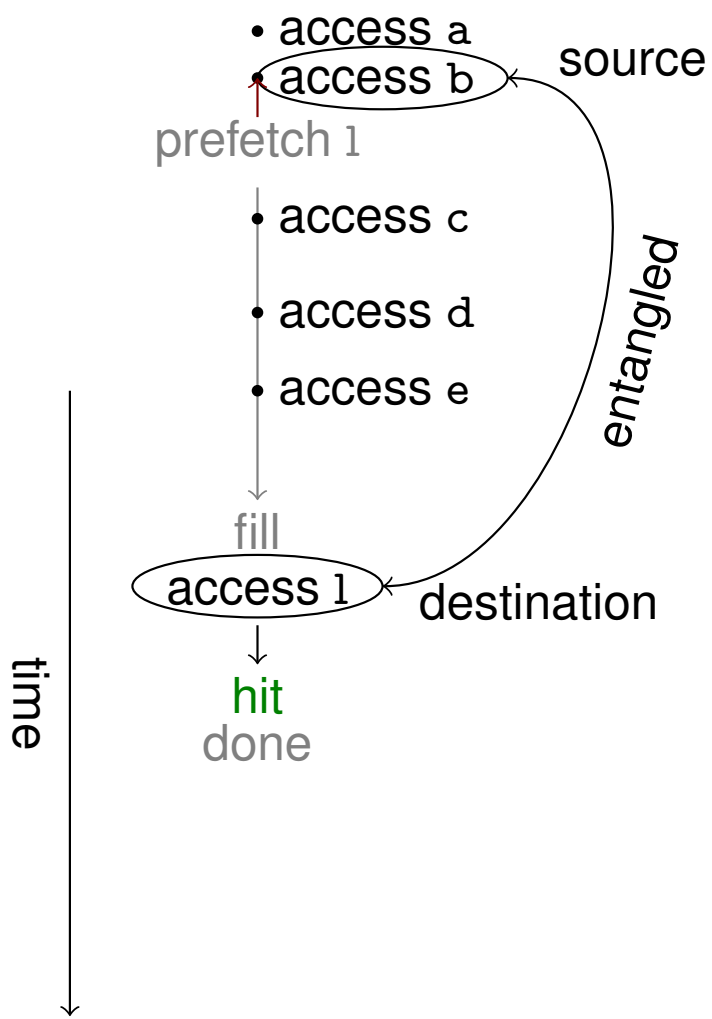
Ros & Jimborean [ISCA'21]



The Entangling Instruction Prefetcher

Concept of entangled pairs

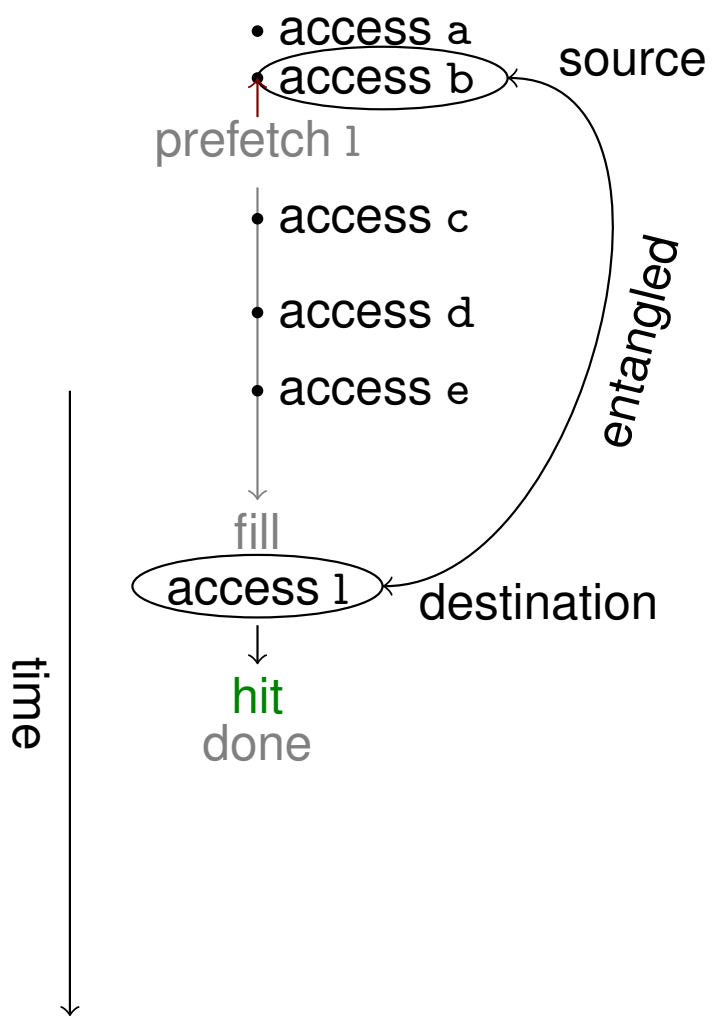
Ros & Jimborean [ISCA'21]

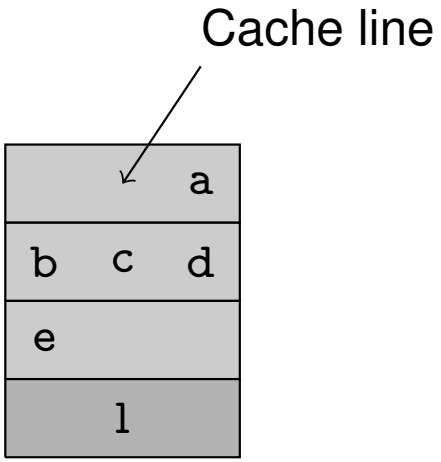


The Entangling Instruction Prefetcher

Concept of entangled pairs

Ros & Jimborean [ISCA'21]

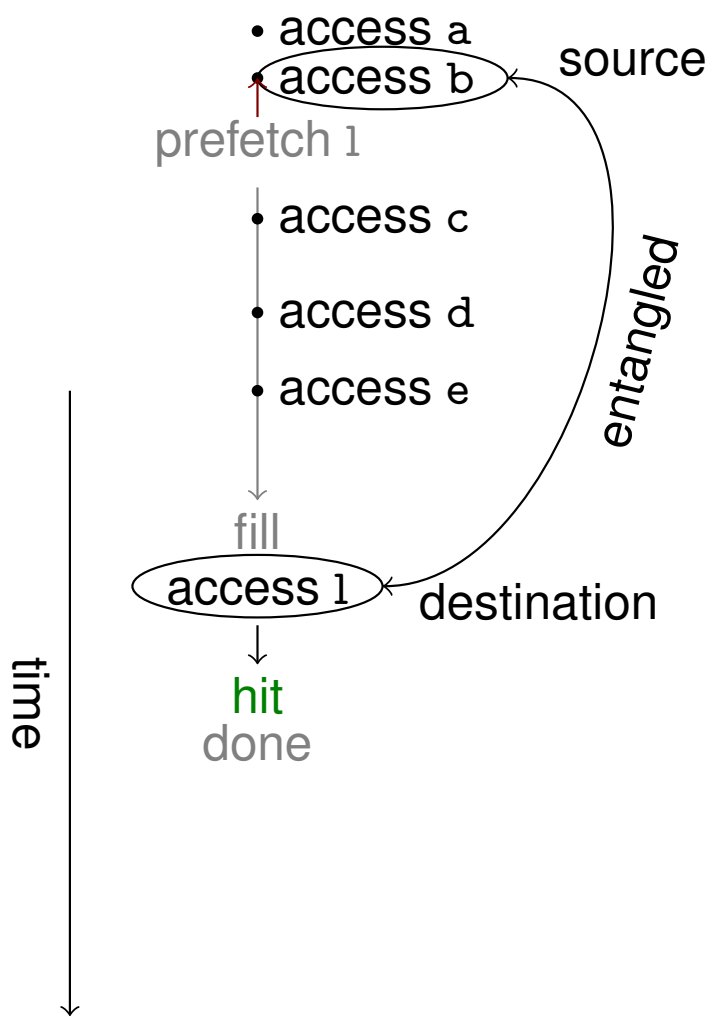


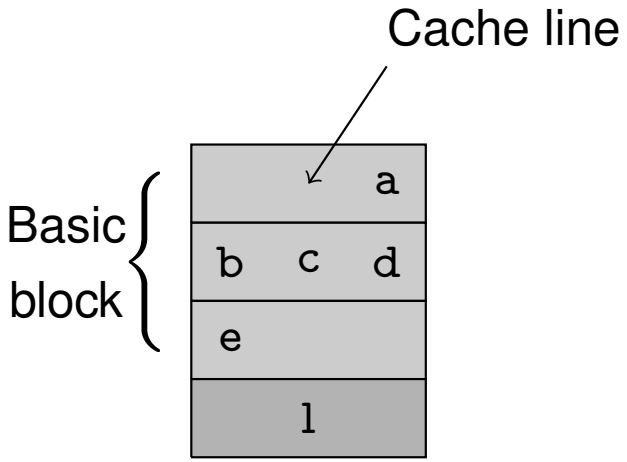


The Entangling Instruction Prefetcher

Concept of entangled pairs

Ros & Jimborean [ISCA'21]

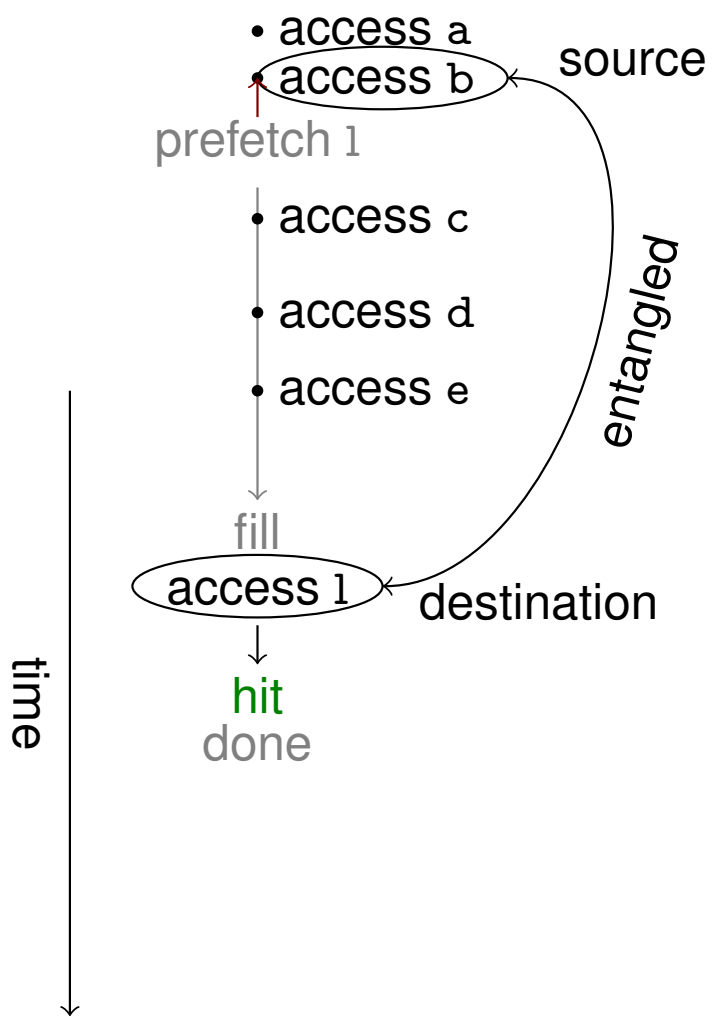


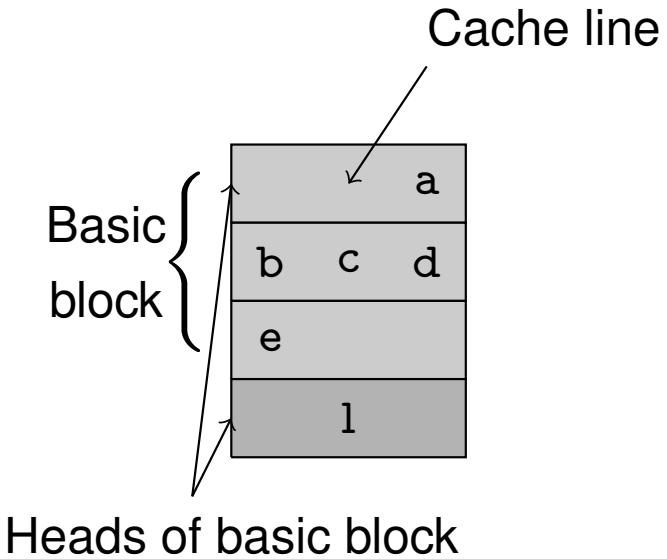


The Entangling Instruction Prefetcher

Concept of entangled pairs

Ros & Jimborean [ISCA'21]

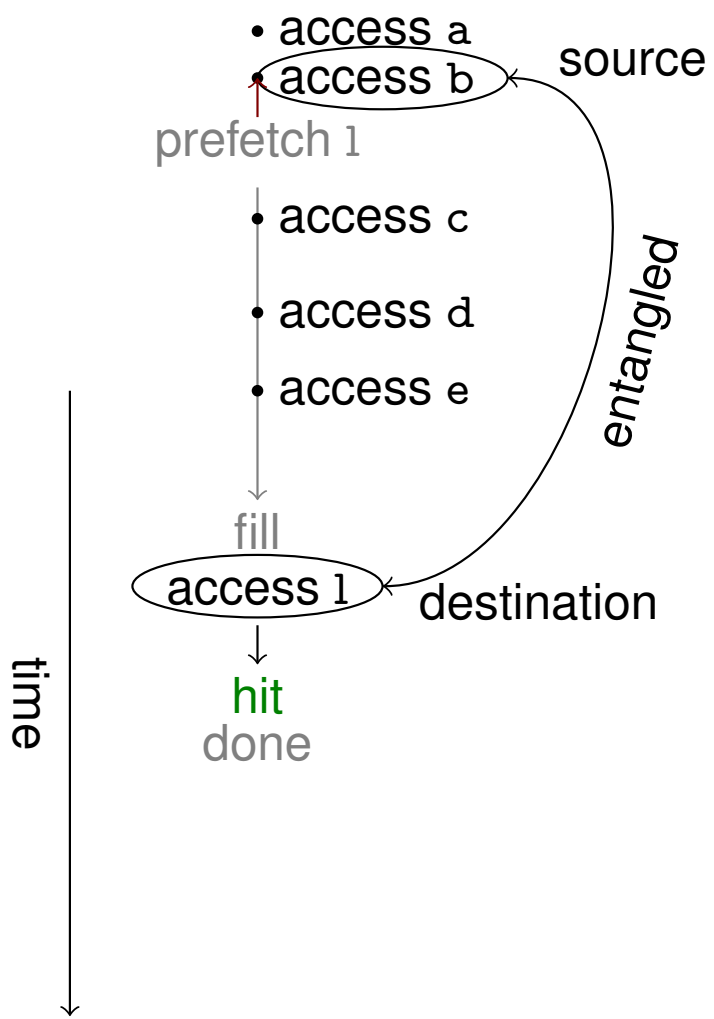


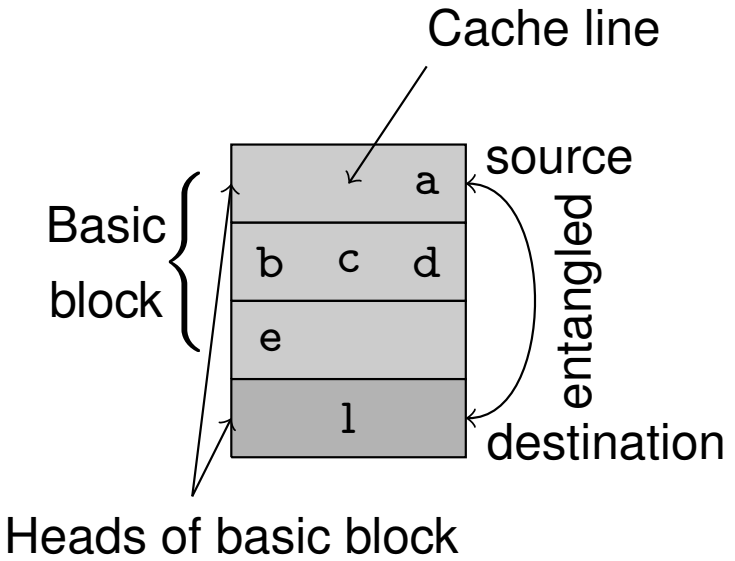


The Entangling Instruction Prefetcher

Concept of entangled pairs

Ros & Jimborean [ISCA'21]

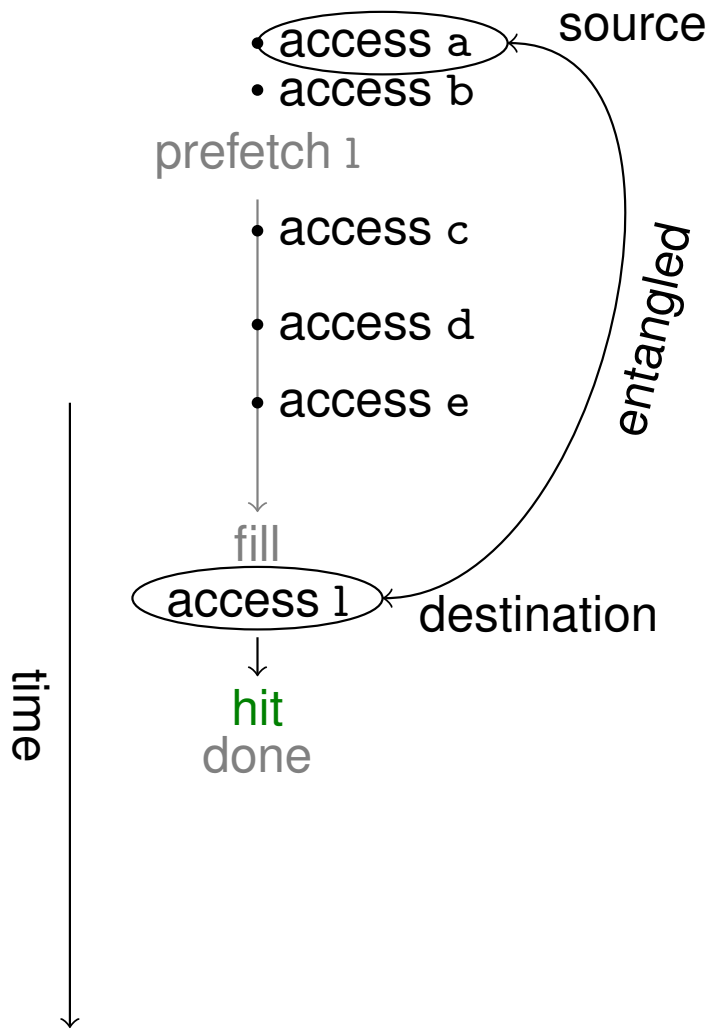


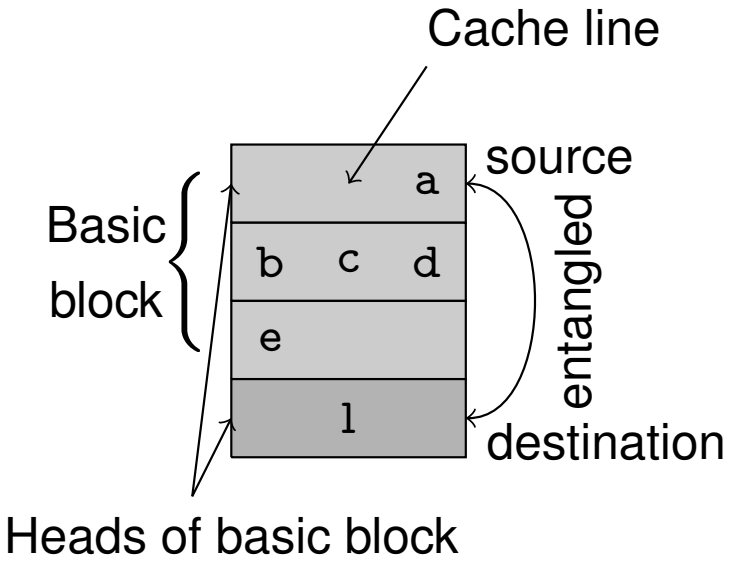


The Entangling Instruction Prefetcher

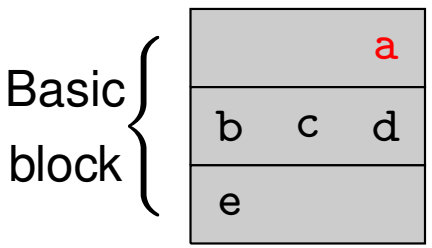
Concept of entangled pairs

Ros & Jimborean [ISCA'21]

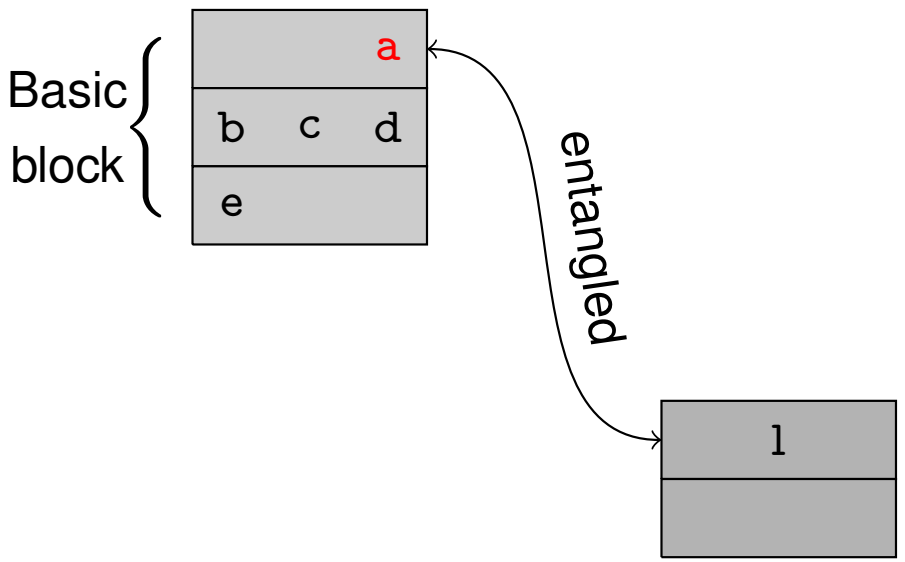




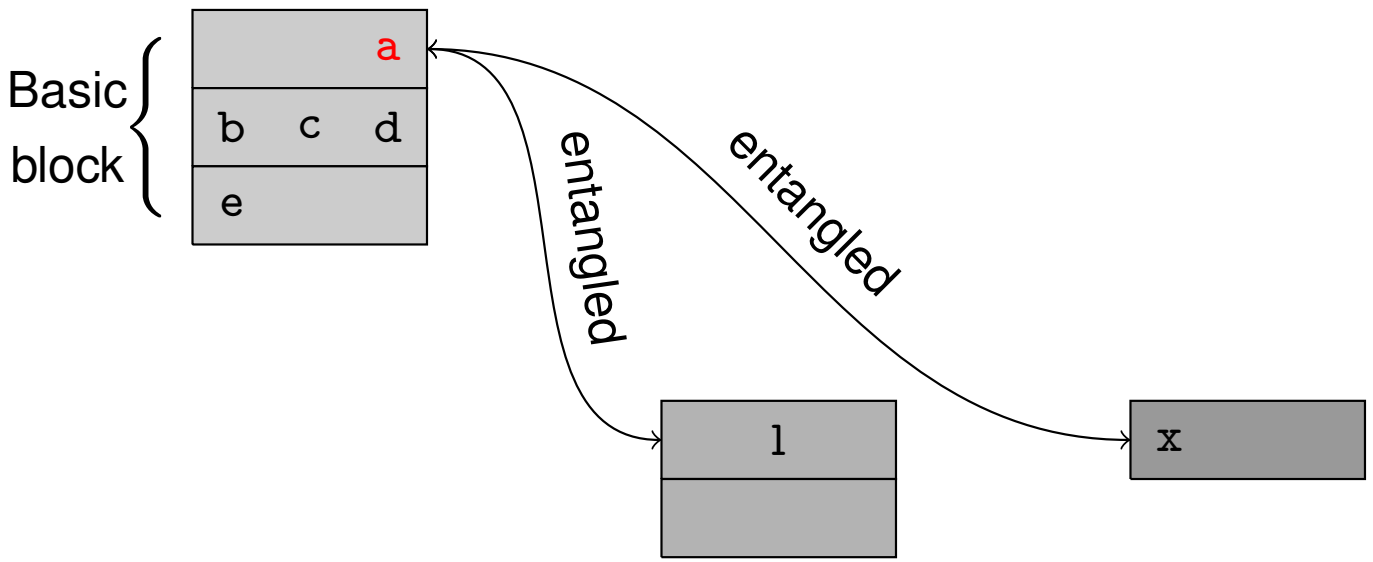
WHAT TO PREFETCH ON AN ACCESS TO a?



WHAT TO PREFETCH ON AN ACCESS TO a?

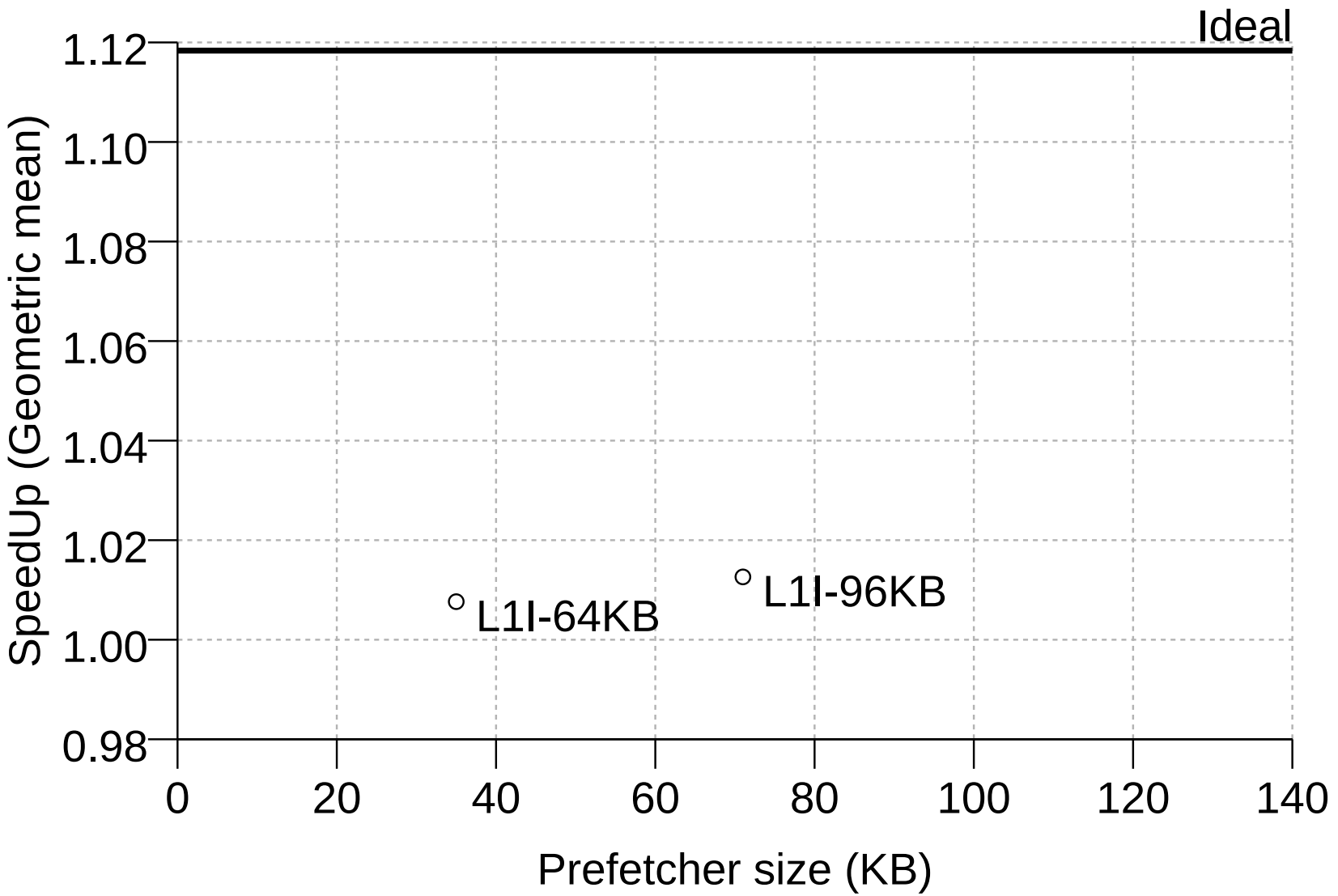


WHAT TO PREFETCH ON AN ACCESS TO a?



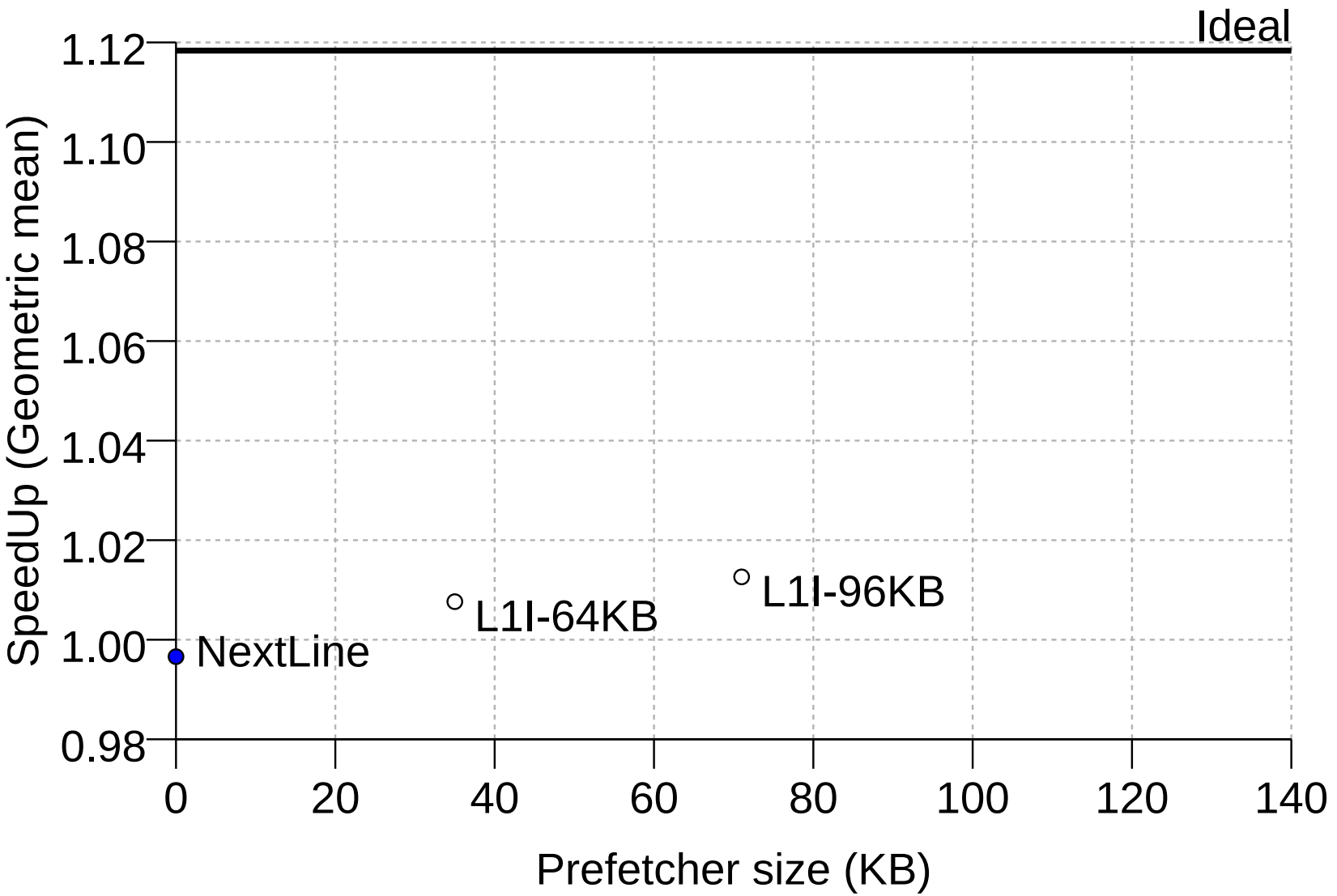
RESULTS: IPC VS STORAGE OVERHEAD

CHAMPSIM SIMULATOR



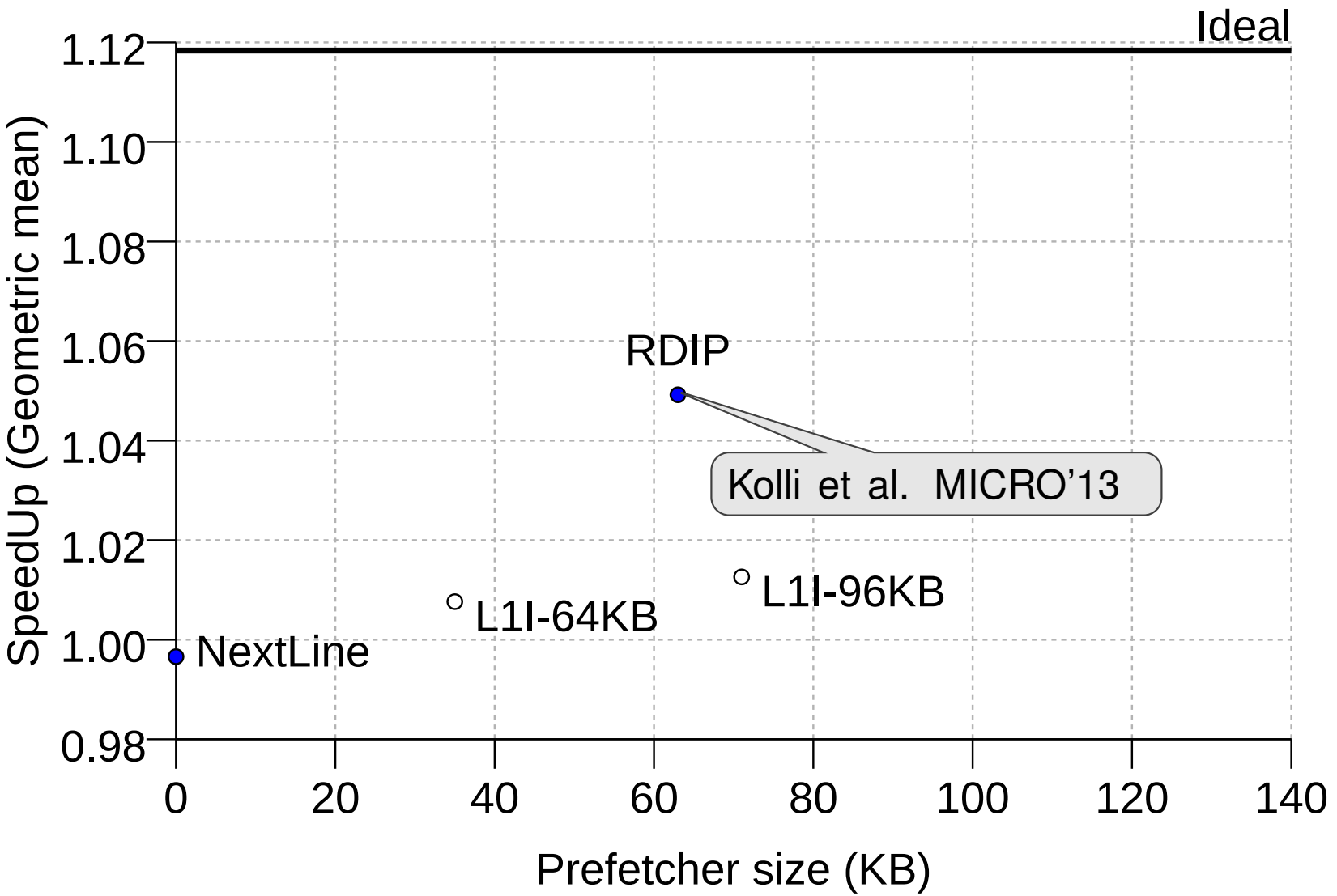
RESULTS: IPC VS STORAGE OVERHEAD

CHAMPSIM SIMULATOR



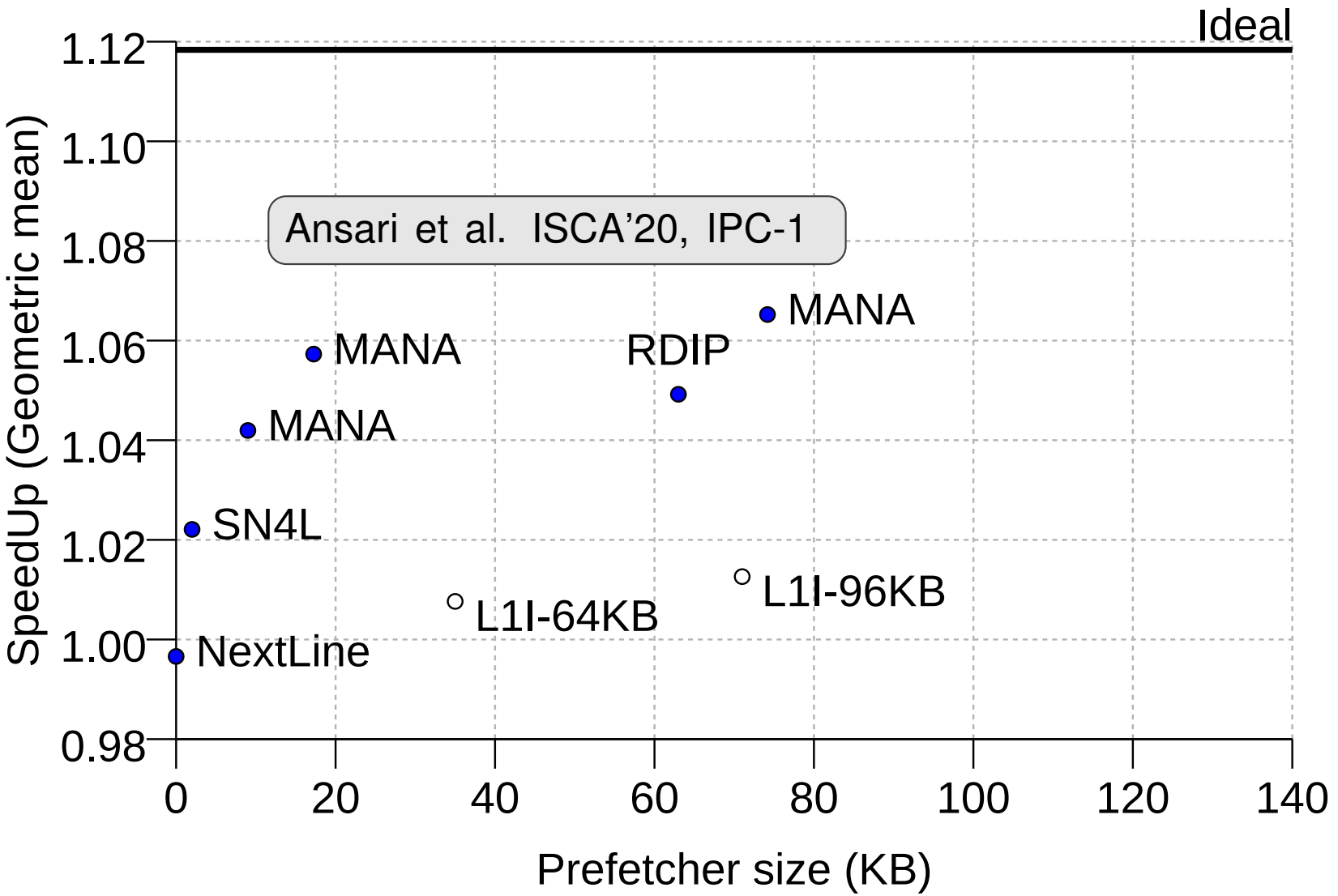
RESULTS: IPC VS STORAGE OVERHEAD

CHAMPSIM SIMULATOR



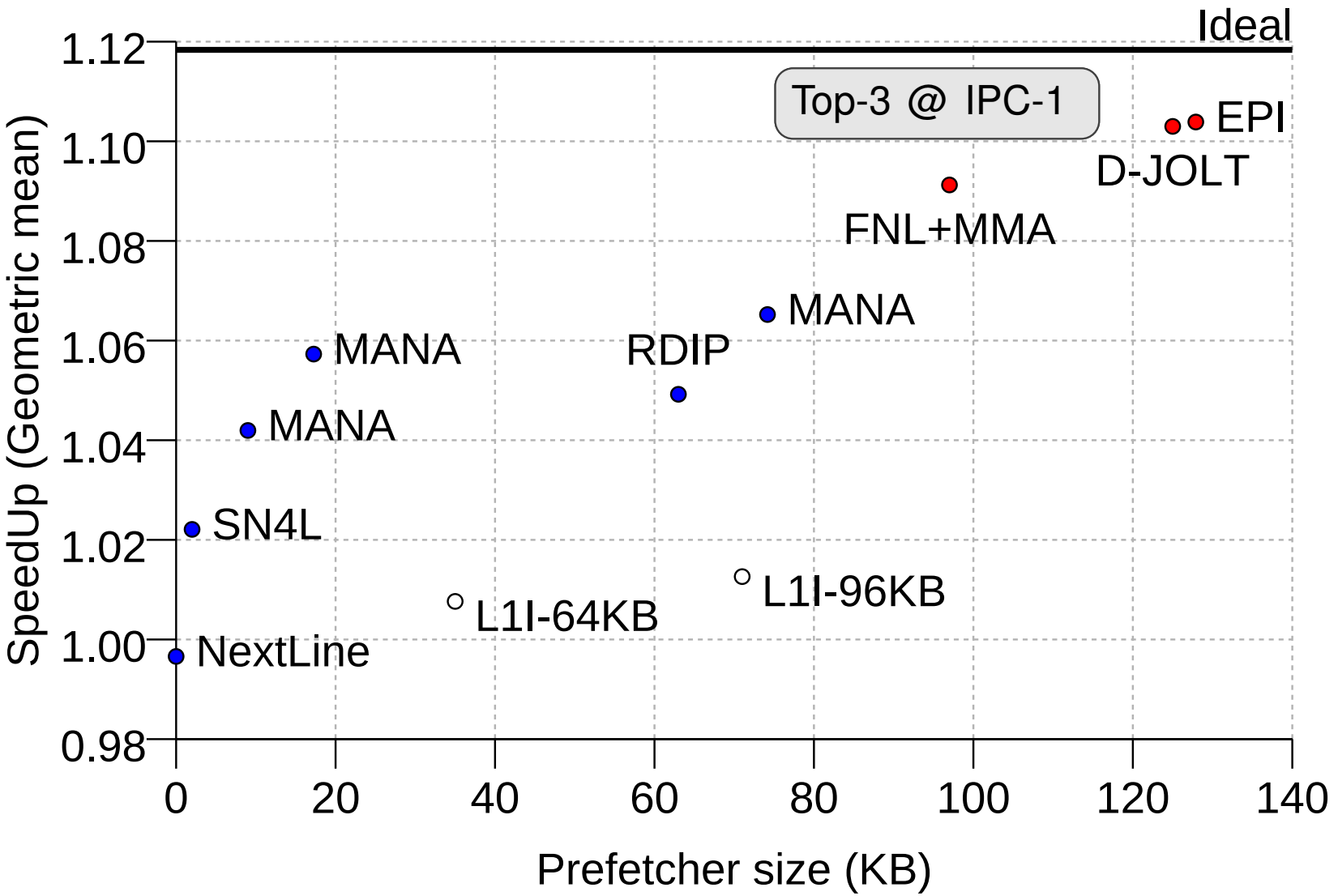
RESULTS: IPC VS STORAGE OVERHEAD

CHAMPSIM SIMULATOR



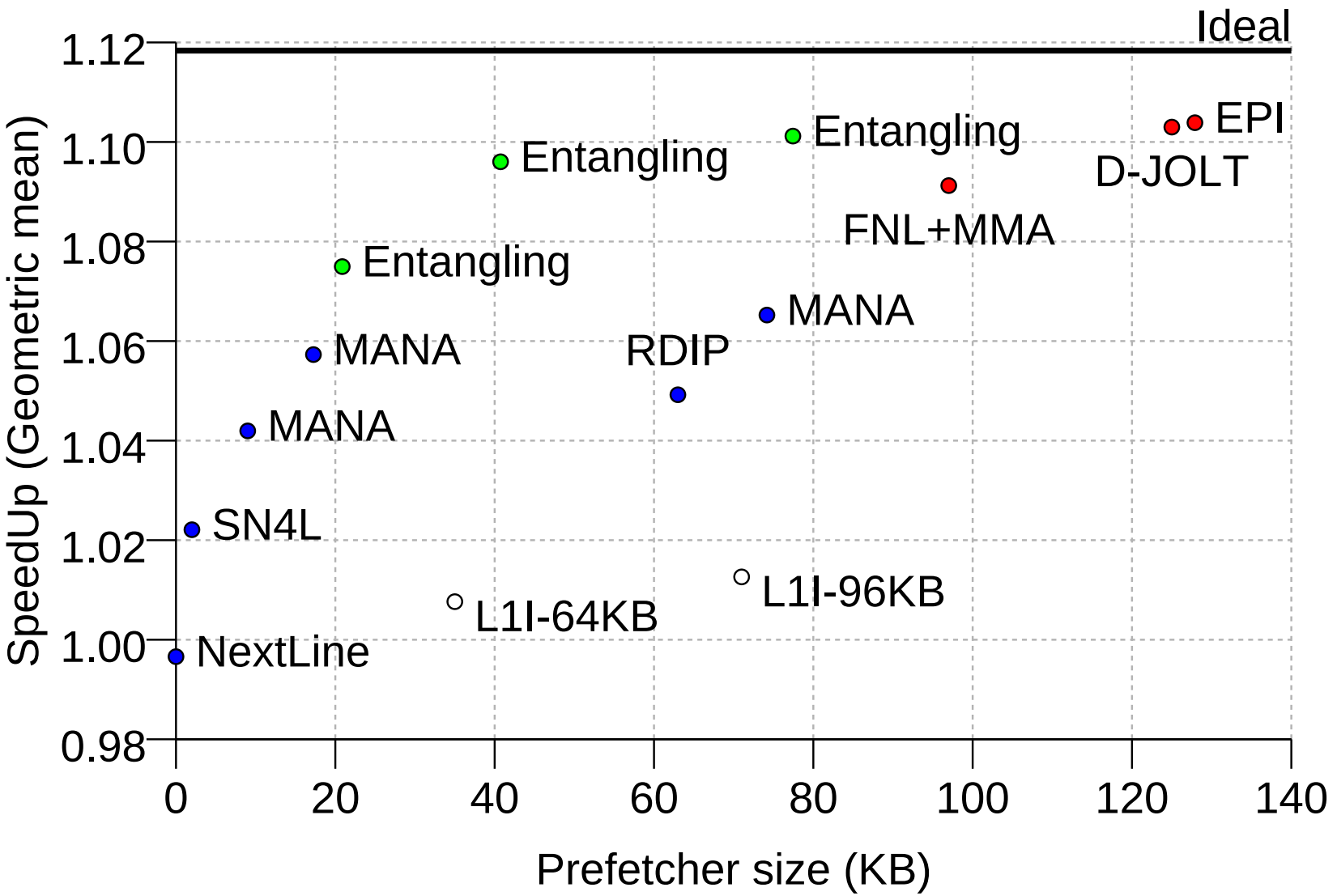
RESULTS: IPC VS STORAGE OVERHEAD

CHAMPSIM SIMULATOR



RESULTS: IPC VS STORAGE OVERHEAD

CHAMPSIM SIMULATOR



BUT WHAT ABOUT THE WRONG PATH?

Ros & Jimborean [IEEE TC'24]

- The Entangling Instruction Prefetcher can easily aviod wrong path pollution

But what about the wrong path?

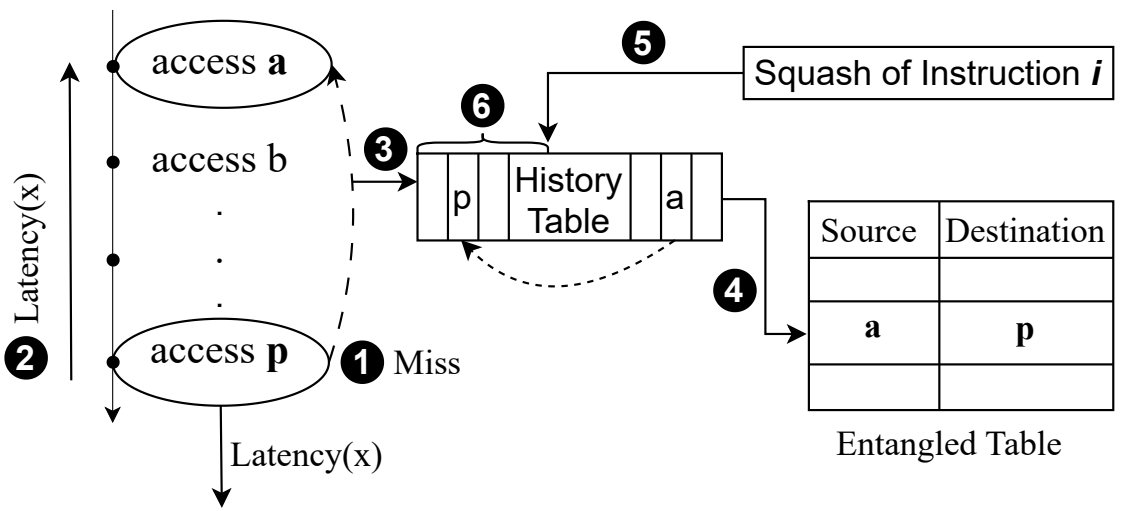
Ros & Jimborean [IEEE TC'24]

- The Entangling Instruction Prefetcher can easily aviod wrong path pollution
- Insights
  - Triggering prefetcher on the wrong path not a problem
  - Training the prefetcher on the wrong path causes pollution

BUT WHAT ABOUT THE WRONG PATH?

Ros & Jimborean [IEEE TC'24]

- The Entangling Instruction Prefetcher can easily aviod wrong path pollution
- INSIGHTS
  - Triggering prefetcher on the wrong path not a problem
  - Training the prefetcher on the wrong path causes pollution
- SOLUTION
  - Buffer learned information and squash on path misprediction



AND DOES IT WORK ONLY FOR CHAMPSIM?

Kalbande, Panda, Jimborean & Ros [CAL'26]

- The Entangling Instruction Prefetcher can easily be adapted to other simulators (e.g. gem5)

AND DOES IT WORK ONLY FOR CHAMPSIM?

Kalbande, Panda, Jimborean & Ros [CAL'26]

- The Entangling Instruction Prefetcher can easily be adapted to other simulators (e.g. gem5)
- BACKGROUND
  - It reports high acurracy (72%) on ChampSim
  - Some works[2,3] have reported low acurracy (30%-45%) when implementing it on gem5

[2] Godala et al., "PDIP: Priority directed instruction prefetching" [ASPLOS'24]
[3] Zhang et al., "Hierarchical prefetching: ..." [ASPLOS'25]

AND DOES IT WORK ONLY FOR CHAMPSIM?

Kalbande, Panda, Jimborean & Ros [CAL'26]

- The Entangling Instruction Prefetcher can easily be adapted to other simulators (e.g. gem5)
- BACKGROUND
  - It reports high acurracy (72%) on ChampSim
  - Some works[2,3] have reported low acurracy (30%-45%) when implementing it on gem5
- PITFALLS
  - Training on retire needs to be done carefully
  - The prefetcher should be called on all accesses

[2] Godala et al., "PDIP: Priority directed instruction prefetching" [ASPLOS'24]
[3] Zhang et al., "Hierarchical prefetching: ..." [ASPLOS'25]

AND DOES IT WORK ONLY FOR CHAMPSIM?

Kalbande, Panda, Jimborean & Ros [CAL'26]

- The Entangling Instruction Prefetcher can easily be adapted to other simulators (e.g. gem5)
- BACKGROUND
  - It reports high acurracy (72%) on ChampSim
  - Some works[2,3] have reported low acurracy (30%-45%) when implementing it on gem5
- PITFALLS
  - Training on retire needs to be done carefully
  - The prefetcher should be called on all accesses
- RESULTS
  - Average accuracy above 65%

[2] Godala et al., "PDIP: Priority directed instruction prefetching" [ASPLOS'24]
[3] Zhang et al., "Hierarchical prefetching: ..." [ASPLOS'25]

ICARUS: KEEP CRITICAL INSTRUCTIONS CLOSE

CRITICALITY AND REUSE BASED INSTRUCTION CACHING

Kalbande, Deshmukh, Ros & Panda [ASPLOS'26]

- Not all instructions are critical
  - ⇒ Most instructions can tolerate large latencies
  - ⇒ Our study with gem5 and data-center workloads show 3.5% average of critical fetches

ICARUS: KEEP CRITICAL INSTRUCTIONS CLOSE

CRITICALITY AND REUSE BASED INSTRUCTION CACHING

Kalbande, Deshmukh, Ros & Panda [ASPLOS'26]

- Not all instructions are critical
  - ⇒ Most instructions can tolerate large latencies
  - ⇒ Our study with gem5 and data-center workloads show 3.5% average of critical fetches
- OBSERVATIONS
  - Criticality is not a property of the instruction
  - Criticality is path sensitive

PC	Branch history	Critical
0xFFFFE8037D00	**0**001010	Always
	1001010	Never

(Rightmost bit denotes the recent branch prediction)

ICARUS: KEEP CRITICAL INSTRUCTIONS CLOSE

CRITICALITY AND REUSE BASED INSTRUCTION CACHING

Kalbande, Deshmukh, Ros & Panda [ASPLOS'26]

- Not all instructions are critical
  - ⇒ Most instructions can tolerate large latencies
  - ⇒ Our study with gem5 and data-center workloads show 3.5% average of critical fetches
- OBSERVATIONS
  - Criticality is not a property of the instruction
  - Criticality is path sensitive

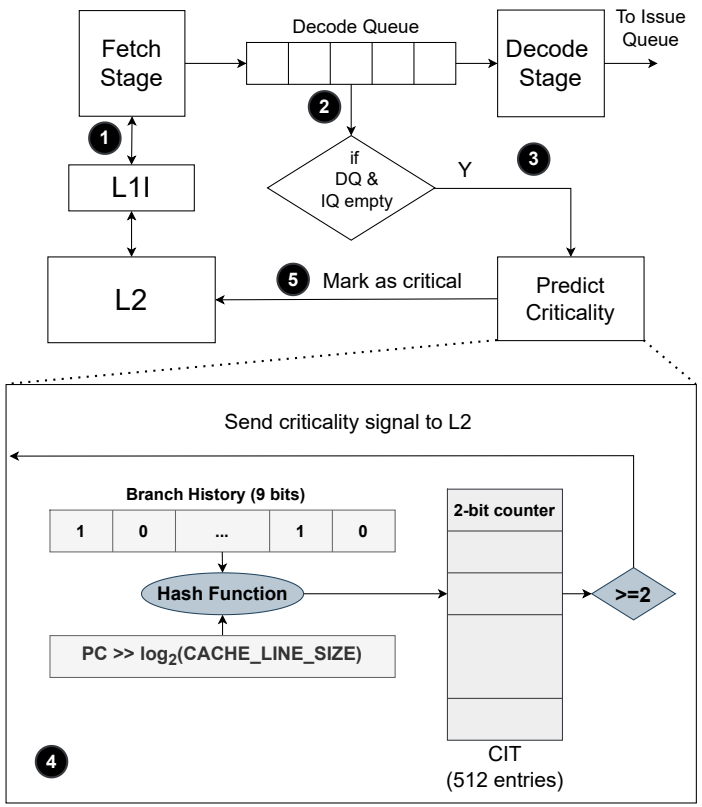
PC	Branch history	Critical
0xFFFFE8037D00	**0**001010	Always
	1001010	Never

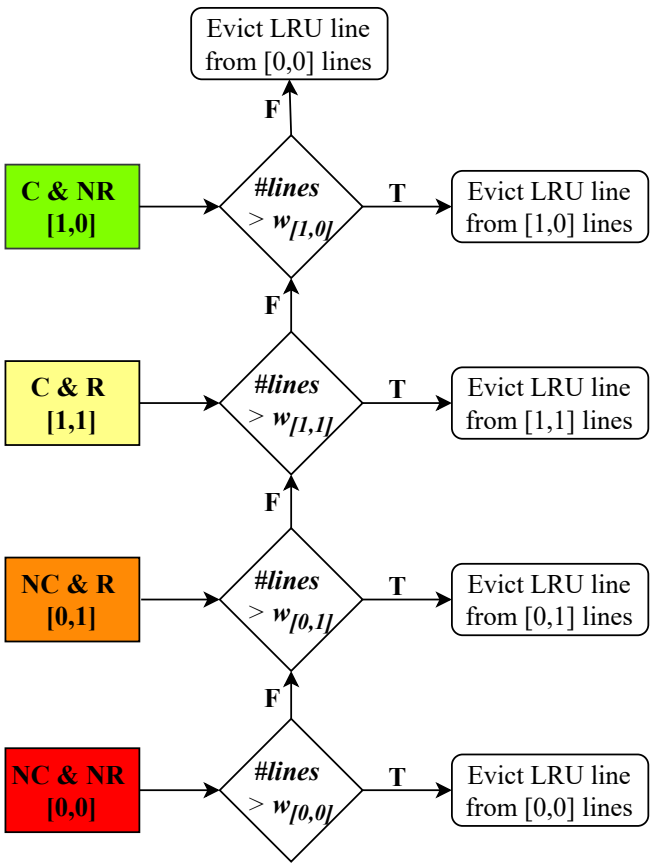
(Rightmost bit denotes the recent branch prediction)

  - But cacheline reuse is also an important factor

ICARUS: IMPLEMENTATION

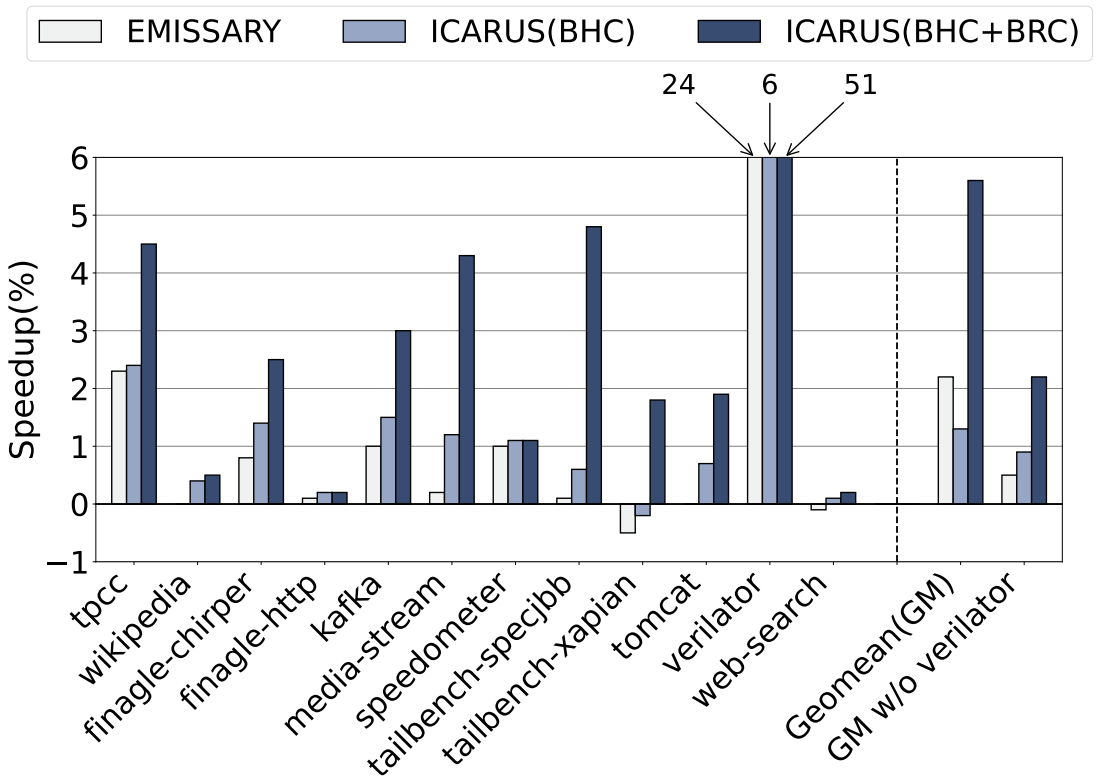
- Criticality and reuse are tracked at L2 *[C,R]*
- Watermarks: $w_{[0,0]} = 2$, $w_{[0,1]} = 4$, $w_{[1,1]} = 6$, $w_{[1,0]} = 4$





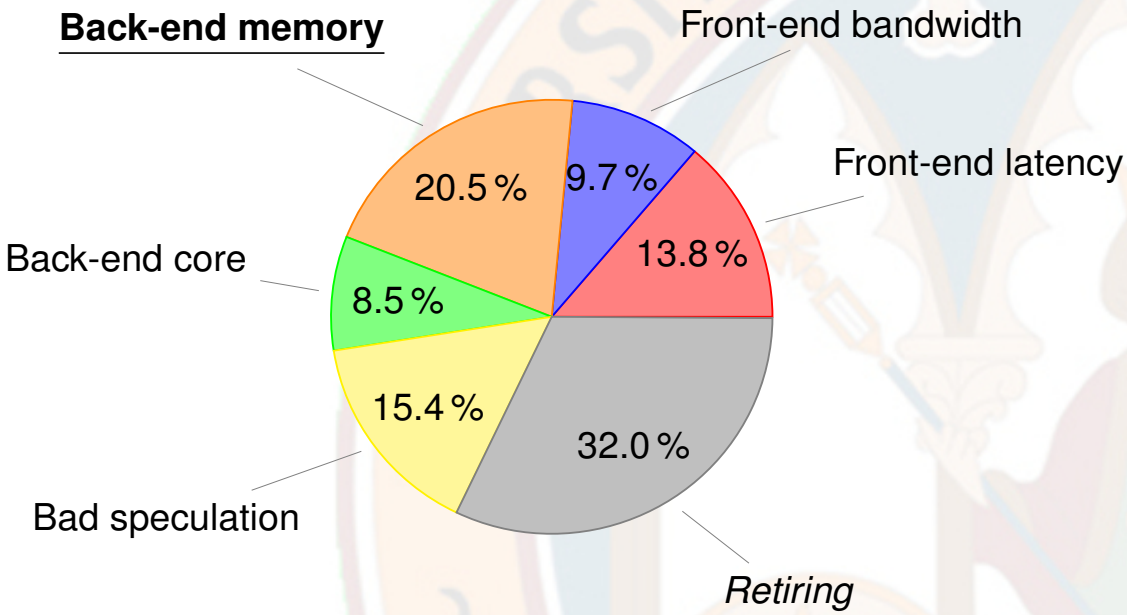
ICARUS: RESULTS
SPEEDUP OVER PLRU

- EMISSARY[4] is the state of the art
- ICARUS BHC: Only history-based criticality; BRC: critiality and reuse



[4] Nagendra et al., "EMISSARY: Enhanced Miss Awareness..." [ISCA'23]

ADDRESSING BACK-END MEMORY (PART 2)



Overview

- Prefetching data timely is key to reduce back-end stalls

Overview

- Prefetching data timely is key to reduce back-end stalls

  $\Rightarrow$ Berti, timely and accurate L1D prefeching

  - Prefetcher code is available[5]
  - L1D baseline for the 2026 Data Prefetching Championship
    - All contestant used a version of Berti for L1D!

[5] https://github.com/agusnt/Berti-Artifact

Overview

- Prefetching data timely is key to reduce back-end stalls

  $\Rightarrow$ Berti, timely and accurate L1D prefeching
  - Prefetcher code is available[5]
  - L1D baseline for the 2026 Data Prefetching Championship
    - All contestant used a version of Berti for L1D!

  $\Rightarrow$ Berti can be secured agaisnt speculative side-channel attacks

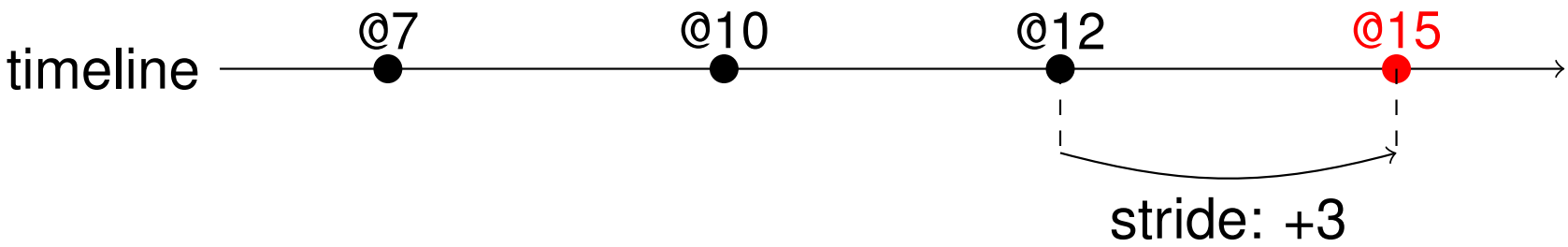
[5] https://github.com/agusnt/Berti-Artifact

BERTI: ACCURATE AND TIMELY LOCAL DELTA L1D PREFETCHER

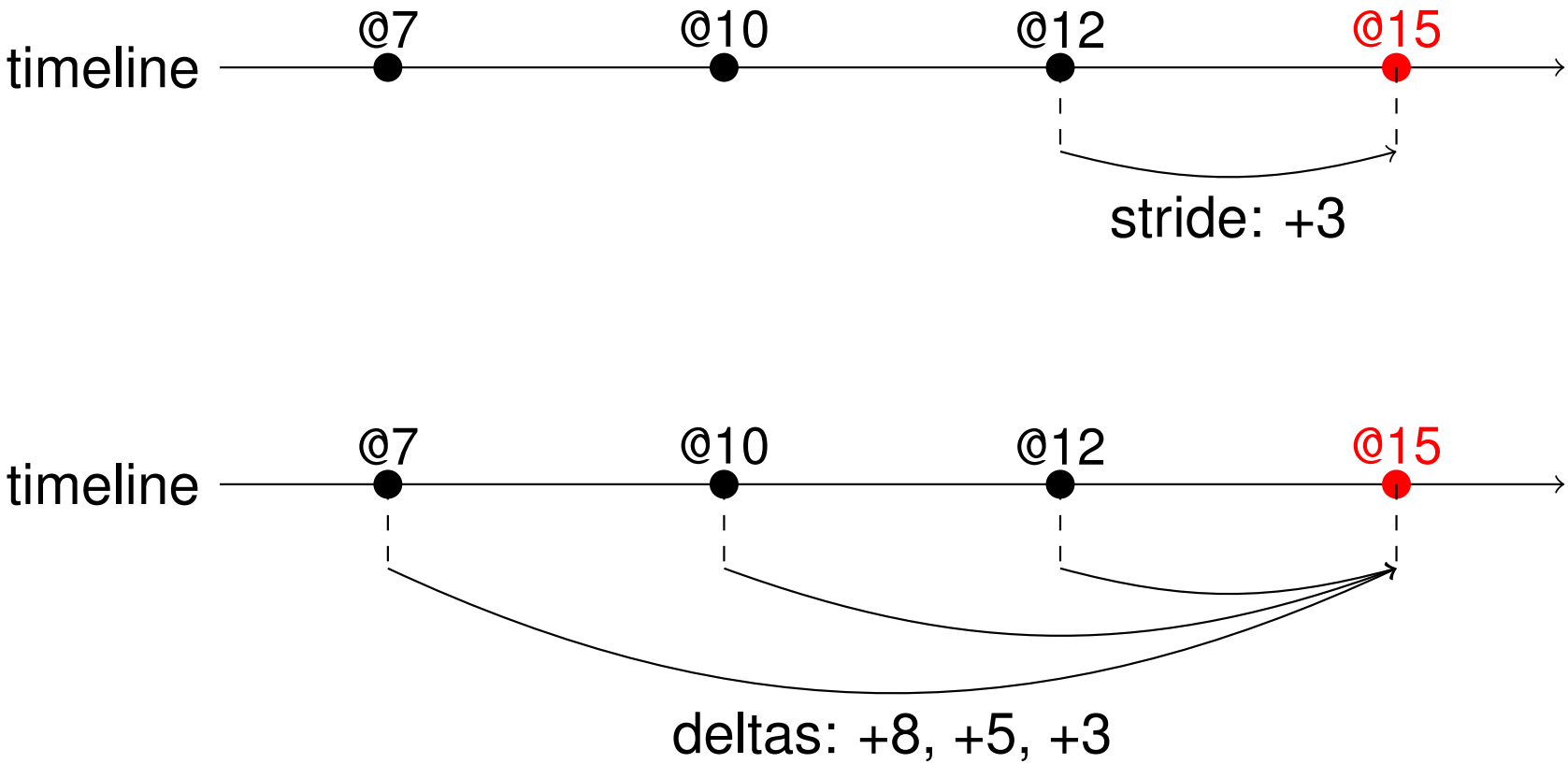
Navarro-Torres, Panda, Alastruey-Benedé, Ibañez, Viñals & Ros [MICRO'22]

- An L1D prefetcher that orchestrates data prefetching from all cache levels
- Advantages of L1D prefetching
  - Training with all processor references
  - Using the Instruction Pointer (IP) information
  - Operating with virtual addresses: cross-page prefetching

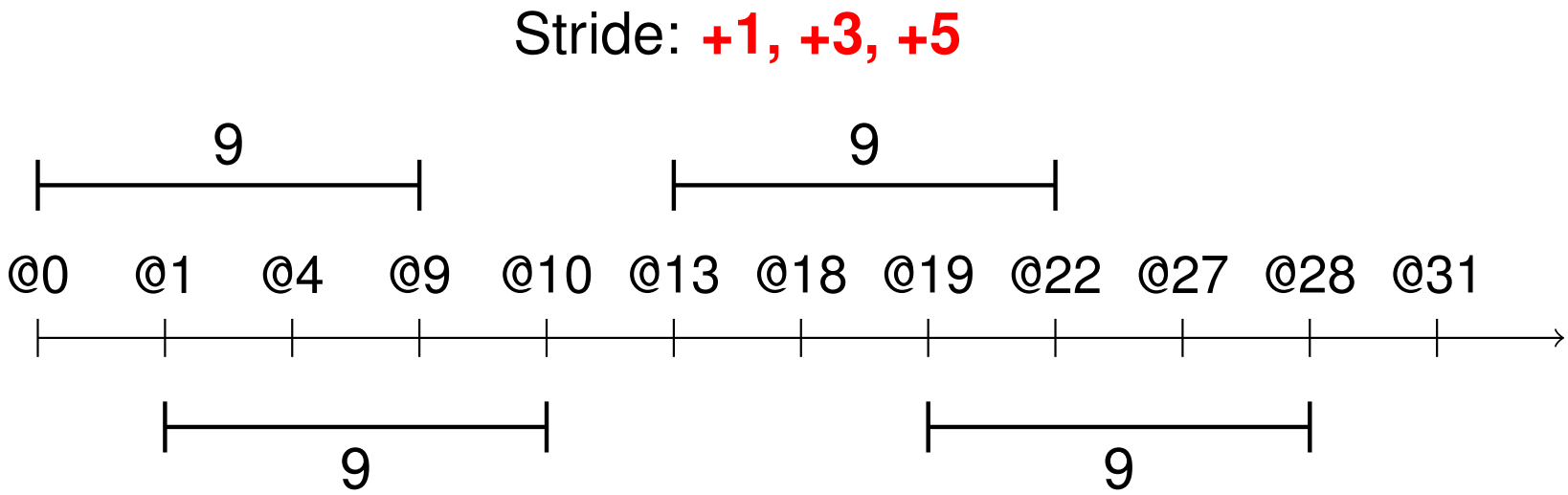
Definition of delta



Definition of delta



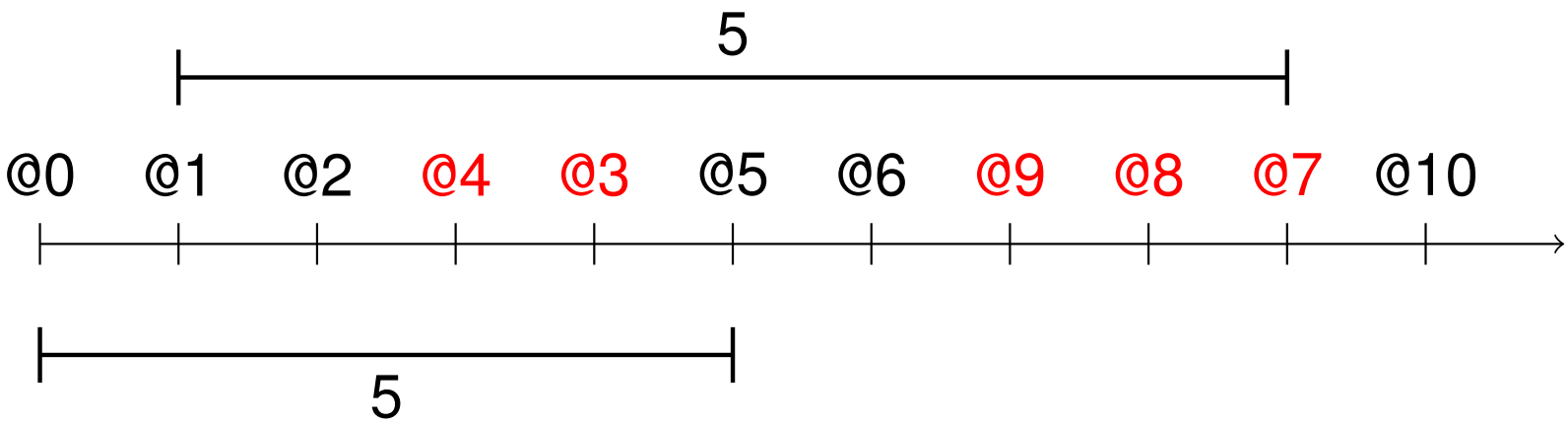
BERTI: ACCURATE AND TIMELY LOCAL DELTA L1D PREFETCHER



With which delta should I prefetch?
delta = 1 + 3 + 5 = 9 → **always hit**

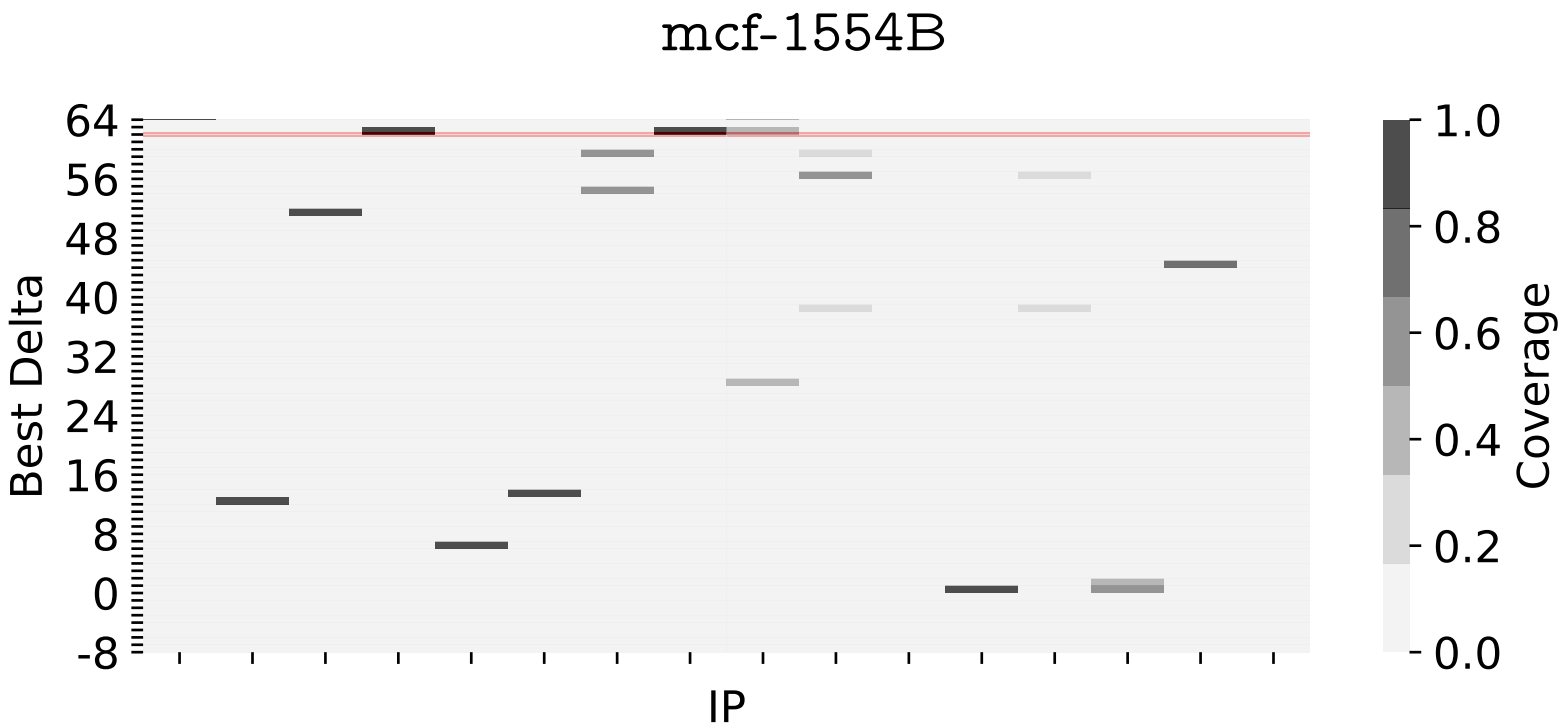
BERTI: ACCURATE AND TIMELY LOCAL DELTA L1D PREFETCHER

Addresses reordered by the out-of-order processor



Stride prefetch requires specific order
Berti can prefetch with delta = 5, for example

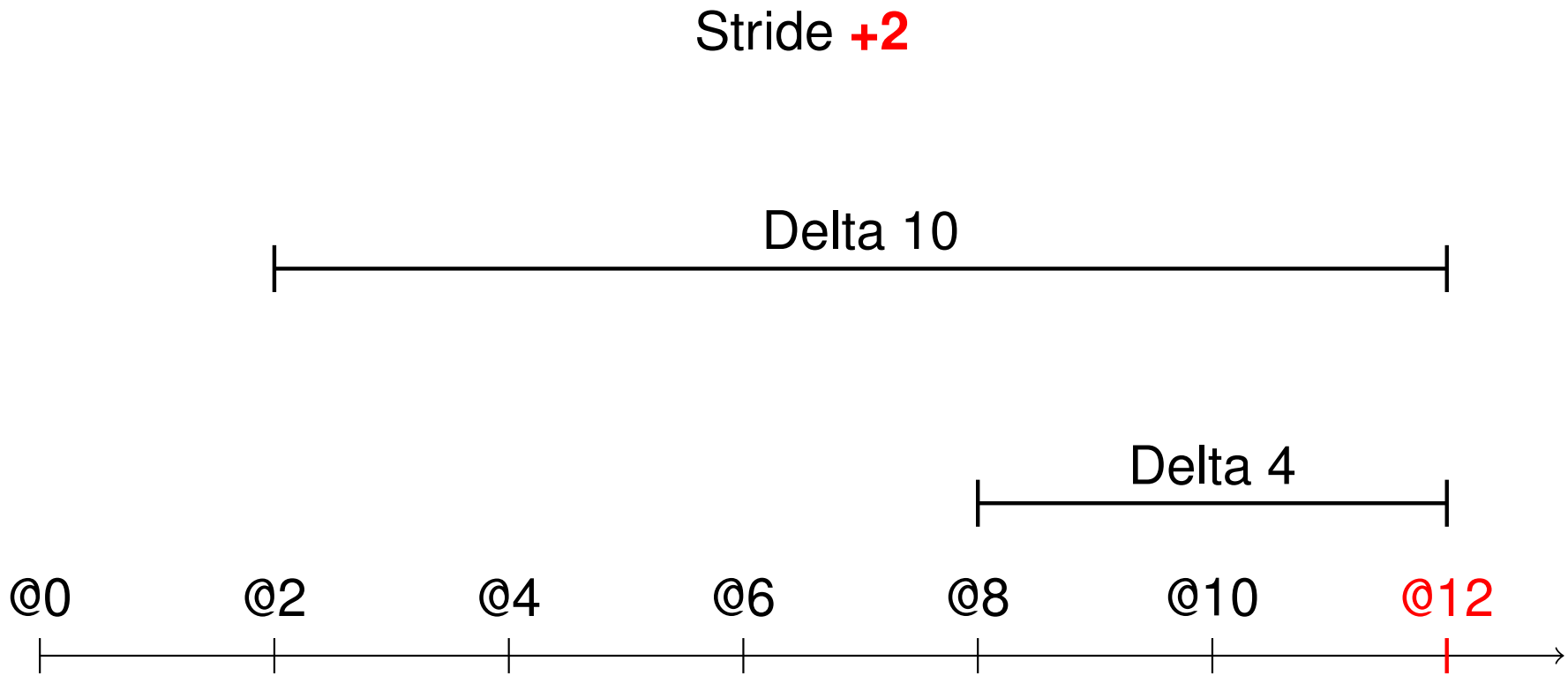
BERTI: ACCURATE AND TIMELY LOCAL DELTA L1D PREFETCHER



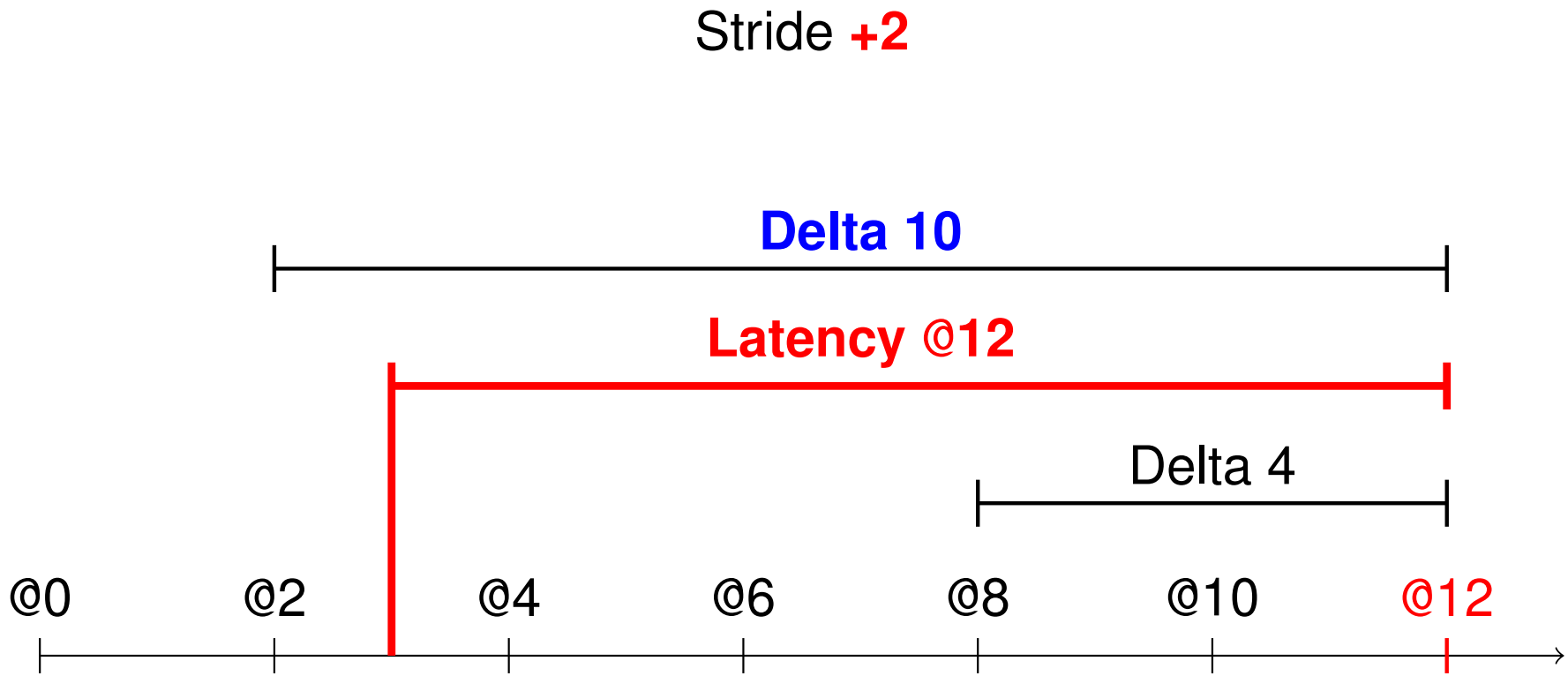
- **Red line**: best delta by BOP[1], coverage: 2%
- **Black lines**: per-IP local deltas, coverage: 10%

[1] Winner of 2nd Data Prefetching Championship (DPC-2)

BERTI: ACCURATE AND TIMELY LOCAL DELTA L1D PREFETCHER



How far in advance should I prefetch address 12?



How far in advance should I prefetch address 12?
Depends on its latency

TRAINING

1. Measure fetch latency
2. Learn timely and accurate deltas
3. Compute coverage of deltas

History table

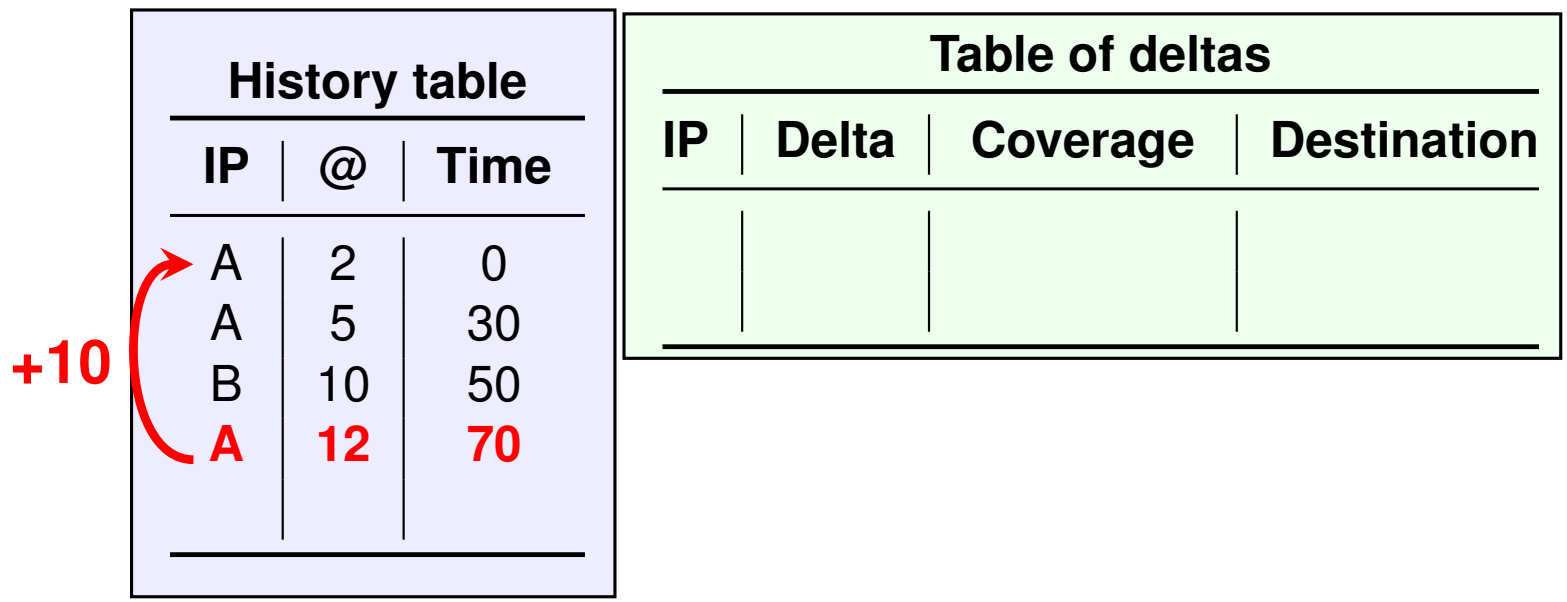
IP	@	Time
A	2	0
A	5	30
B	10	50

Table of deltas

IP	Delta	Coverage	Destination

TRAINING

1. Measure fetch latency
2. Learn timely and accurate deltas
3. Compute coverage of deltas



TRAINING

1. Measure fetch latency
2. Learn timely and accurate deltas
3. Compute coverage of deltas

History table

IP	@	Time
A	2	0
A	5	30
B	10	50
A	**12**	**70**

Table of deltas

IP	Delta	Coverage	Destination
A	**+10**	**1/1 (100%)**	

TRAINING

1. Measure fetch latency
2. Learn timely and accurate deltas
3. Compute coverage of deltas

+10, +13

History table

IP	@	Time
A	2	0
A	5	30
B	10	50
A	12	70
A	**15**	**140**

Table of deltas

IP	Delta	Coverage	Destination
A	**+10**	**2/2 (100%)**	
A	**+13**	**1/2 (50%)**	

ISSUING PREFETCH REQUESTS

1. Select deltas
2. Orchestration

History table

IP	@	Time
A	2	0
A	5	30
B	10	50
A	12	70
A	15	140

Table of deltas

IP	Delta	Coverage	Destination
A	+10	2/2 (100%)	
A	+13	1/2 (50%)	

ISSUING PREFETCH REQUESTS

1. Select deltas
2. Orchestration

History table

IP	@	Time
A	2	0
A	5	30
B	10	50
A	12	70
A	15	140

Table of deltas

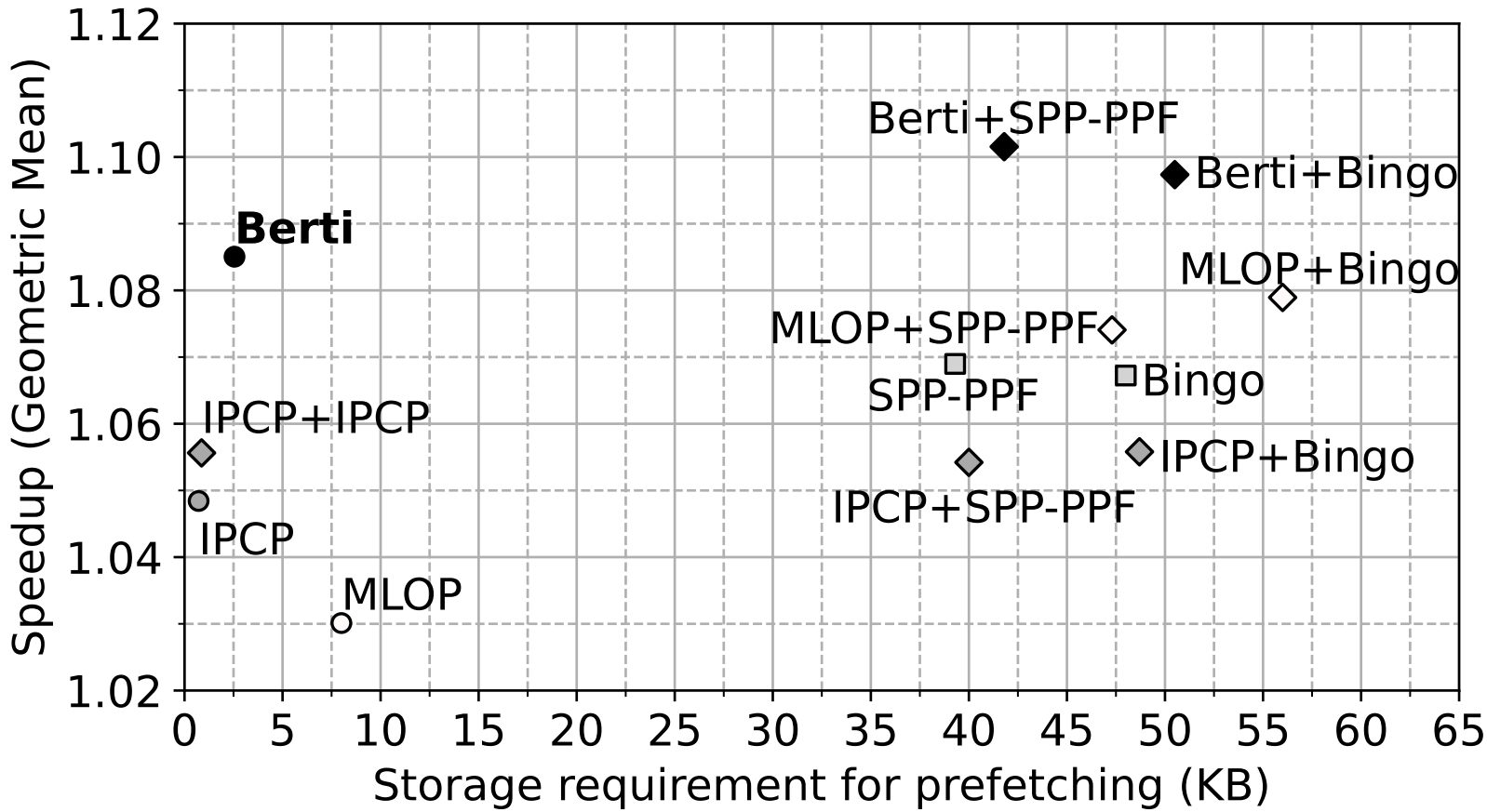
IP	Delta	Coverage	Destination
A	+10	2/2 (100%)	L1D
A	+13	1/2 (50%)	L2

Coverage

> 65% $\rightarrow$ *L1D*

> 35% $\rightarrow$ *L2*

BERTI: OVERALL PERFORMANCE



BERTI CAN BE SECURED

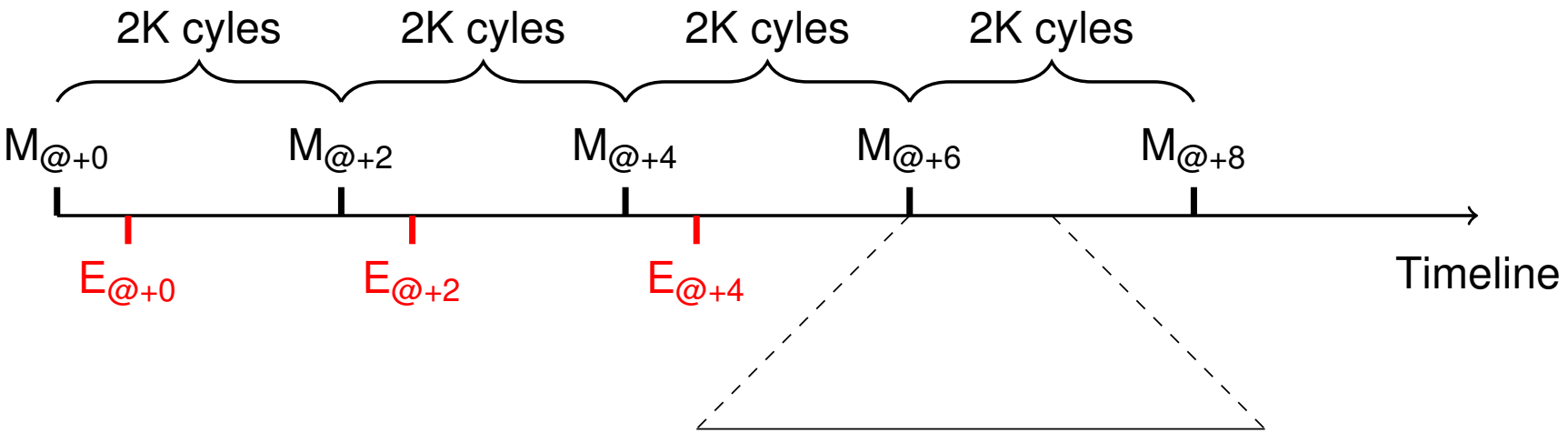
Nath, Navarro-Torres, Ros & Panda [MICRO'24]

- Securing BERTI against speculative side-channel attacks
  - Train at retire: requires carefull latency adjustment
  - Trigger at retire: but bringing the line before its access
- Results show no penalty w.r.t. training and triggering on accesses

WHY NOT ENTANGLING BERTI?

Navarro-Torres, Singh, Panda & Ros [DPC'26]

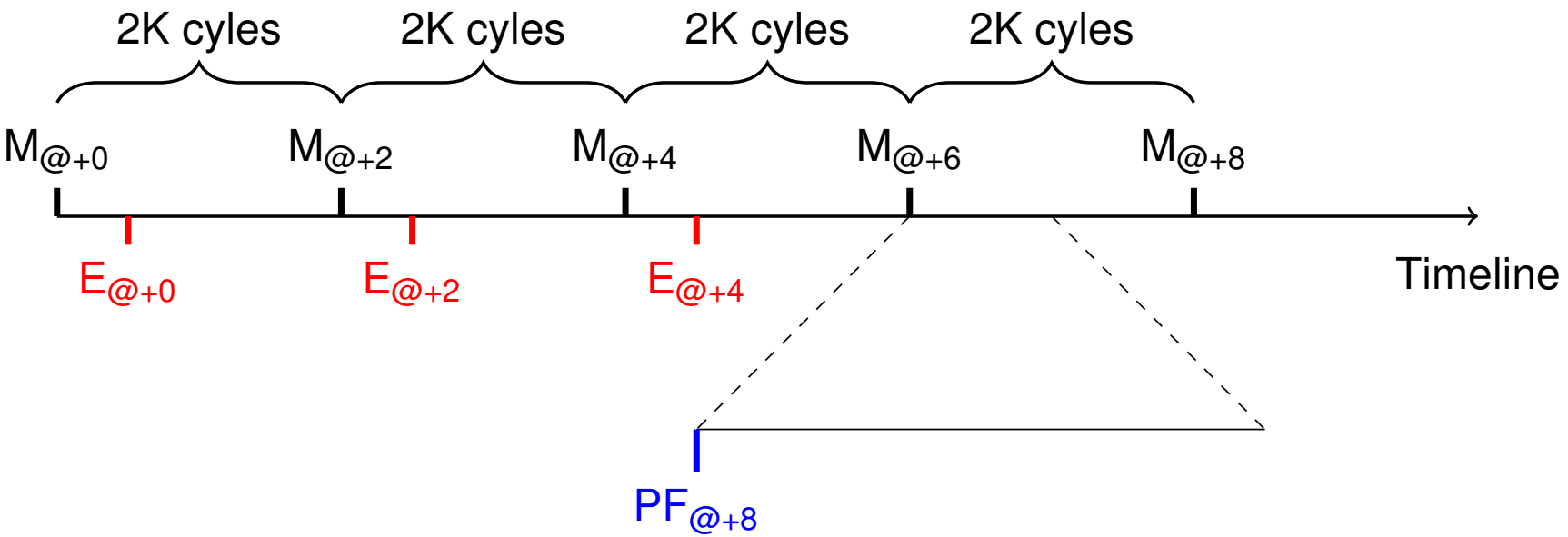
- Which patterns fail Berti to cover?
  - Regular strides with large inter-access intervals (thousands of cycles), e.g., like in `cactuBSSN`



WHY NOT ENTANGLING BERTI?

Navarro-Torres, Singh, Panda & Ros [DPC'26]

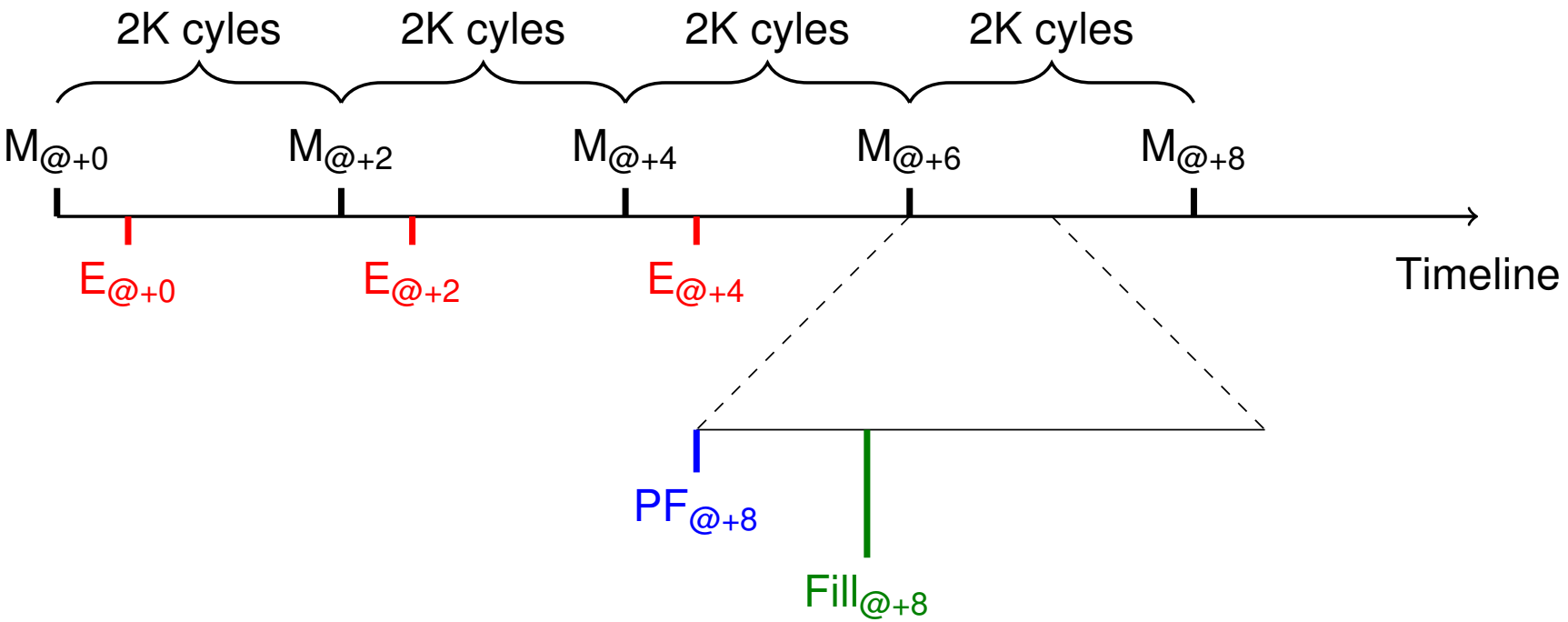
- Which patterns fail Berti to cover?
  - Regular strides with large inter-access intervals (thousands of cycles), e.g., like in `cactuBSSN`



WHY NOT ENTANGLING BERTI?

Navarro-Torres, Singh, Panda & Ros [DPC'26]

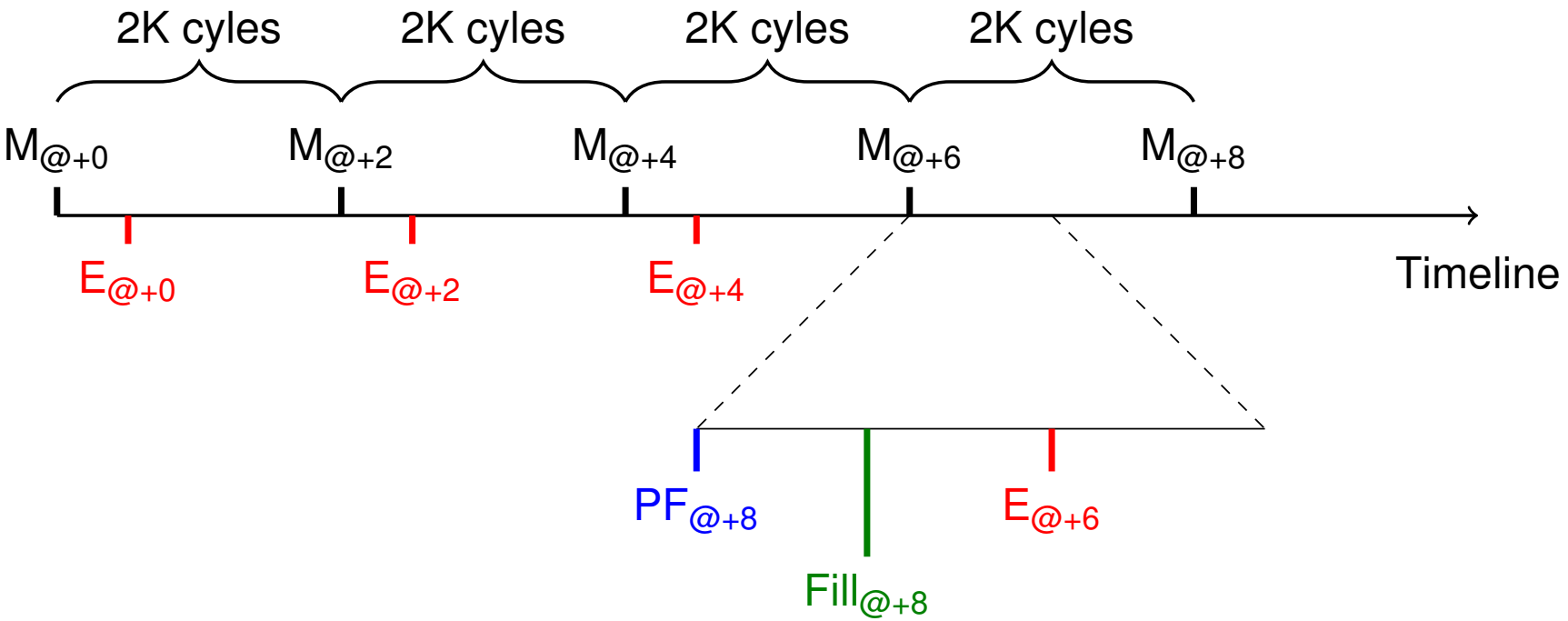
- Which patterns fail Berti to cover?
  - Regular strides with large inter-access intervals (thousands of cycles), e.g., like in `cactuBSSN`



WHY NOT ENTANGLING BERTI?

Navarro-Torres, Singh, Panda & Ros [DPC'26]

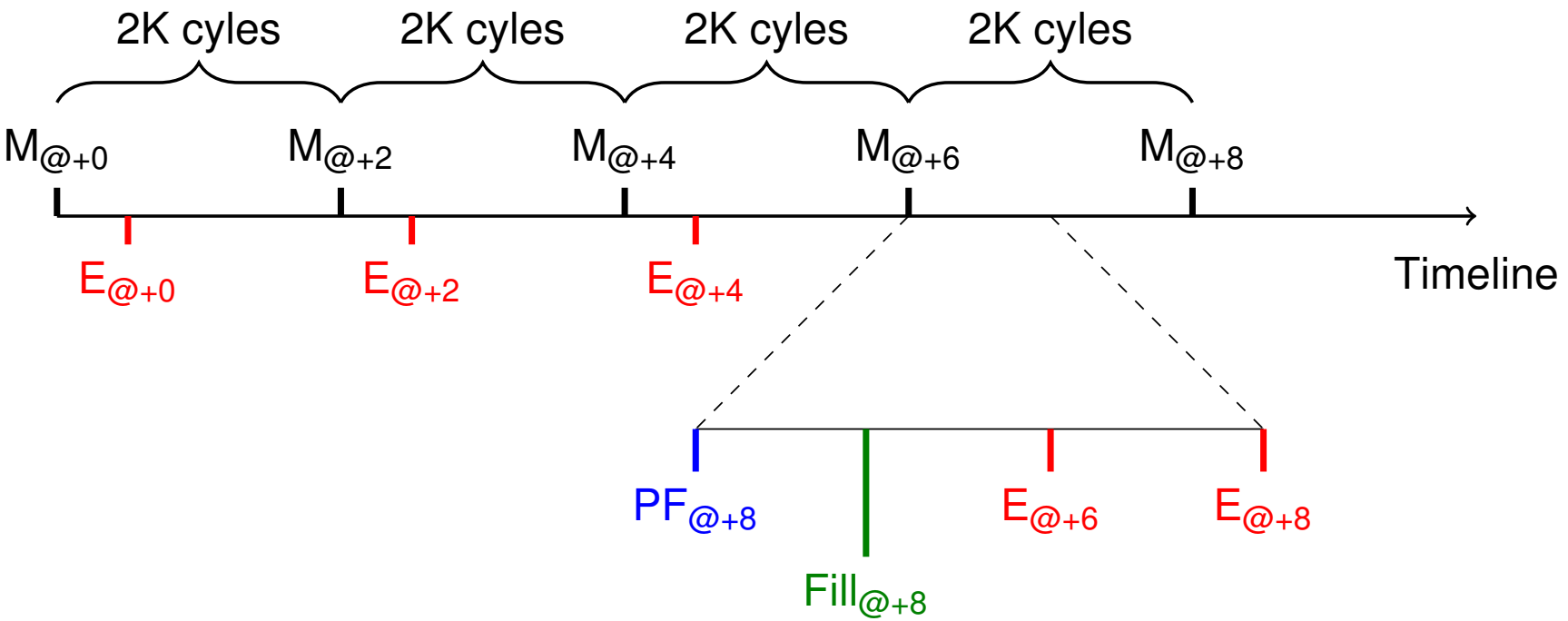
- Which patterns fail Berti to cover?
  - Regular strides with large inter-access intervals (thousands of cycles), e.g., like in `cactuBSSN`



WHY NOT ENTANGLING BERTI?

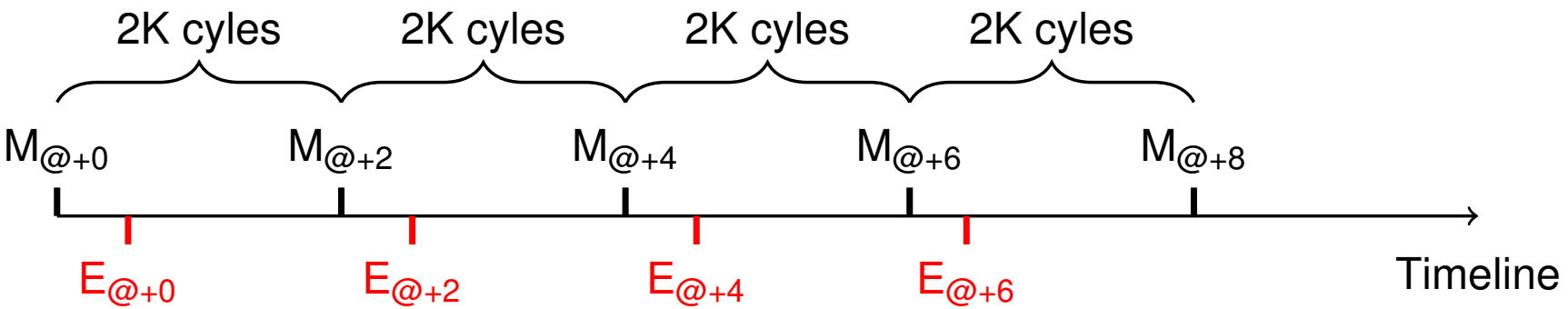
Navarro-Torres, Singh, Panda & Ros [DPC'26]

- Which patterns fail Berti to cover?
  - Regular strides with large inter-access intervals (thousands of cycles), e.g., like in `cactuBSSN`
  - Just because of lack of timeliness



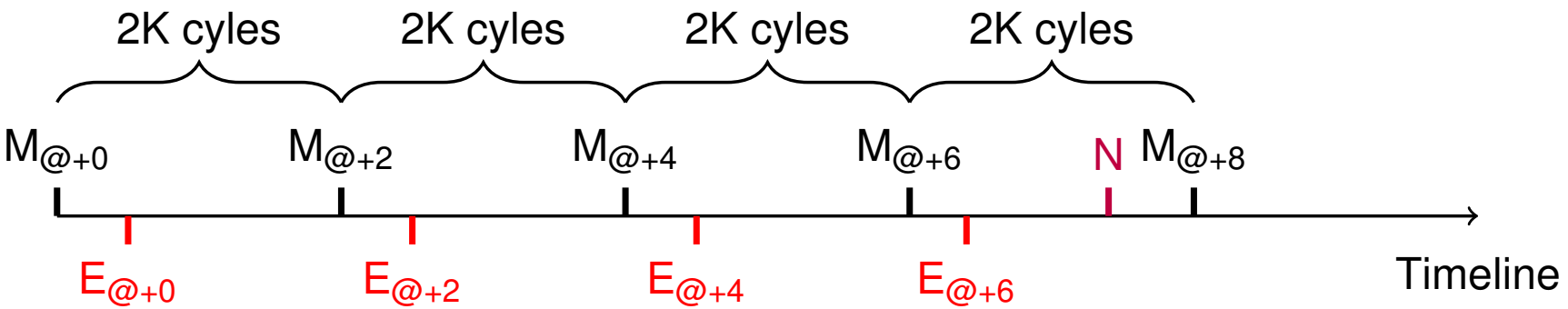
The Entangling Data Prefetcher

- A load triggers the prefetcher for another load
  Rightarrow entangled pairs



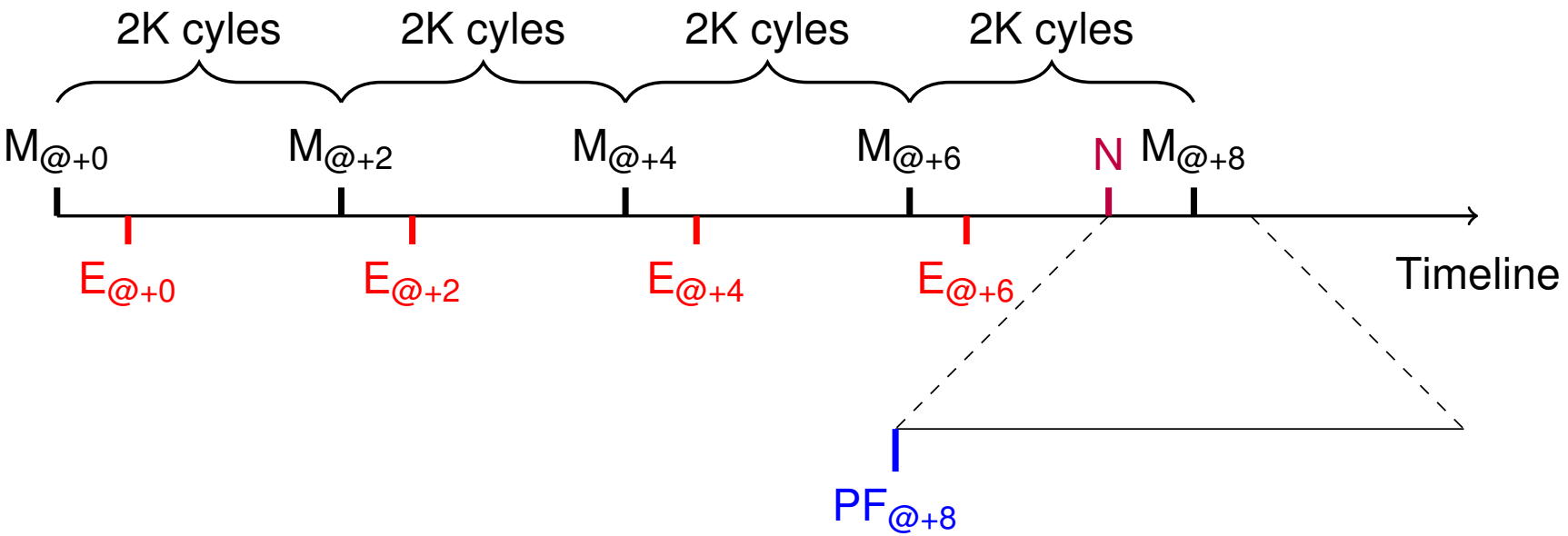
The Entangling Data Prefetcher

- A load triggers the prefetcher for another load
  Rightarrow entangled pairs



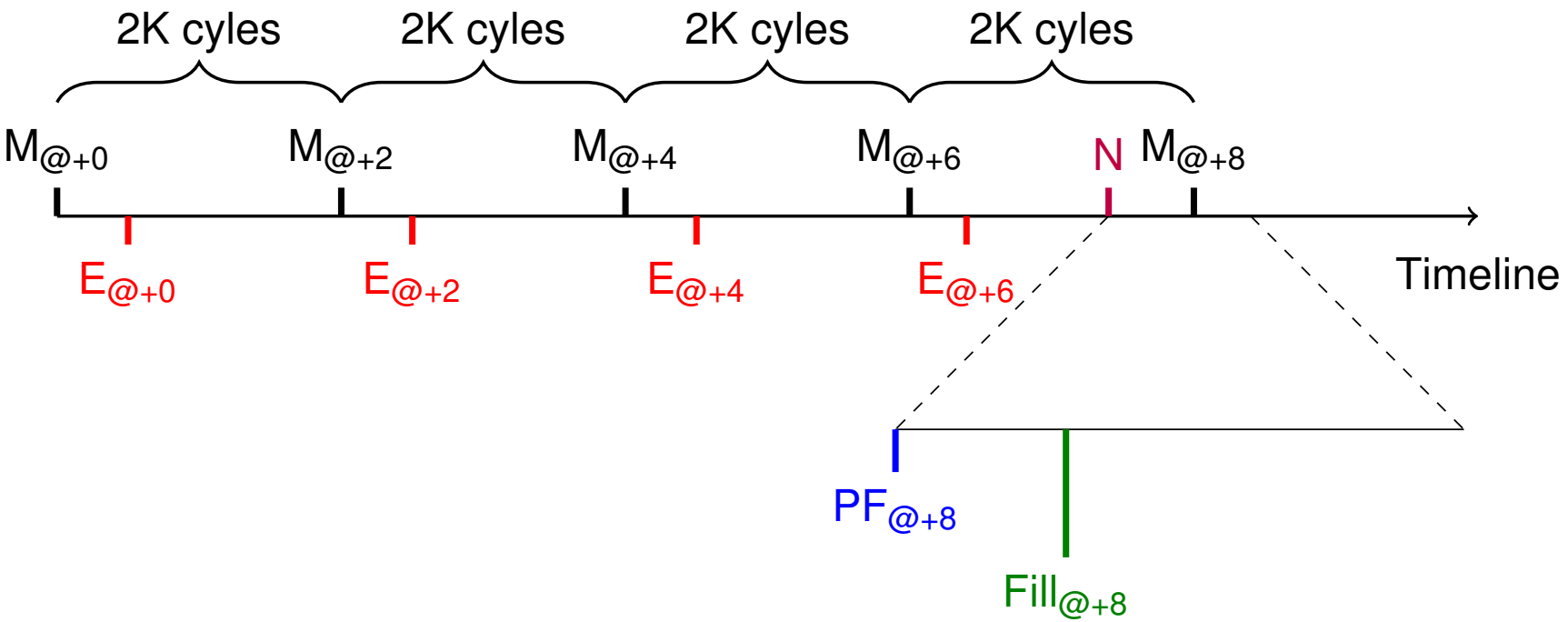
The Entangling Data Prefetcher

- A load triggers the prefetcher for another load
  Rightarrow entangled pairs



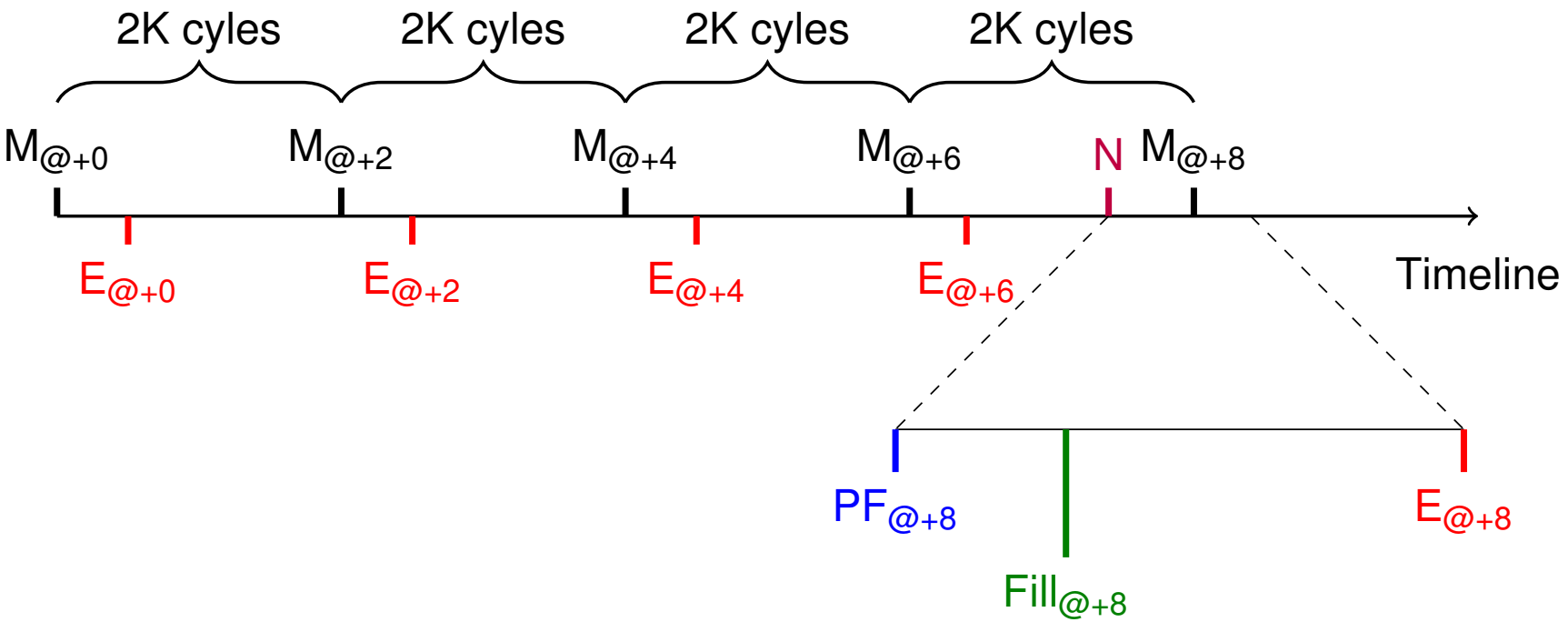
The Entangling Data Prefetcher

- A load triggers the prefetcher for another load
  Rightarrow entangled pairs



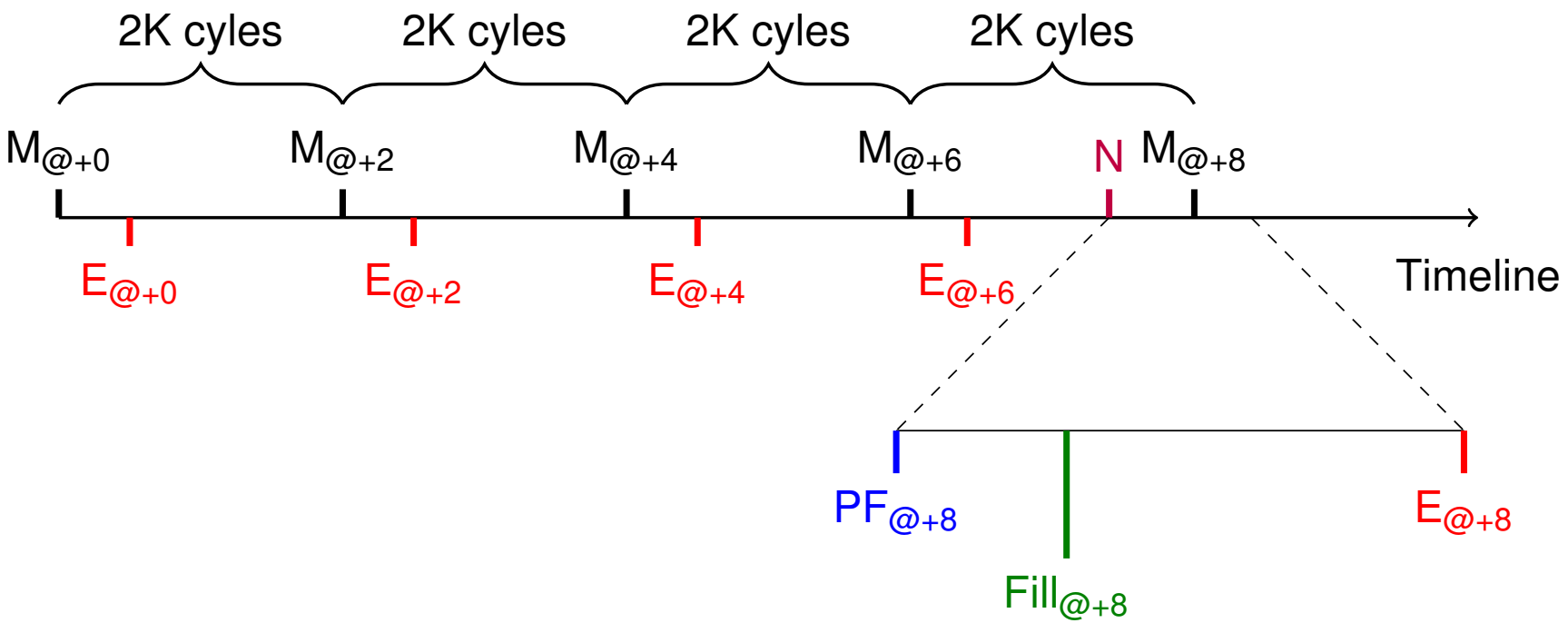
The Entangling Data Prefetcher

- A load triggers the prefetcher for another load
  Rightarrow entangled pairs



The Entangling Data Prefetcher

- A load triggers the prefetcher for another load
  Rightarrow entangled pairs
- It can cover zero-strides[6]



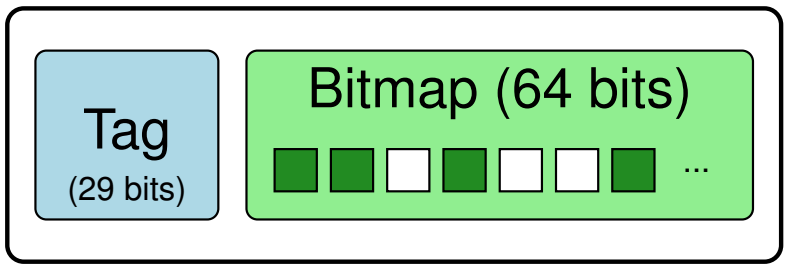
[6] Nakamura et al., "T-SKID: Timing Skid Prefetcher" [DPC-3]

The winner of the 2026 Championship

BertiGo: Region-Based Bitmap Filter

Singh, Navarro-Torres & Ros [DPC'26]

Filter Entry



How it works:

1. On access/prefetch: **Set bit**
2. Before prefetch: **Check bit**
3. Bit set? → **Skip prefetch!**
4. On L1D eviction: **Clear bit**

Learns Useless Prefetches

If L2 prefetch never promoted to L1D:

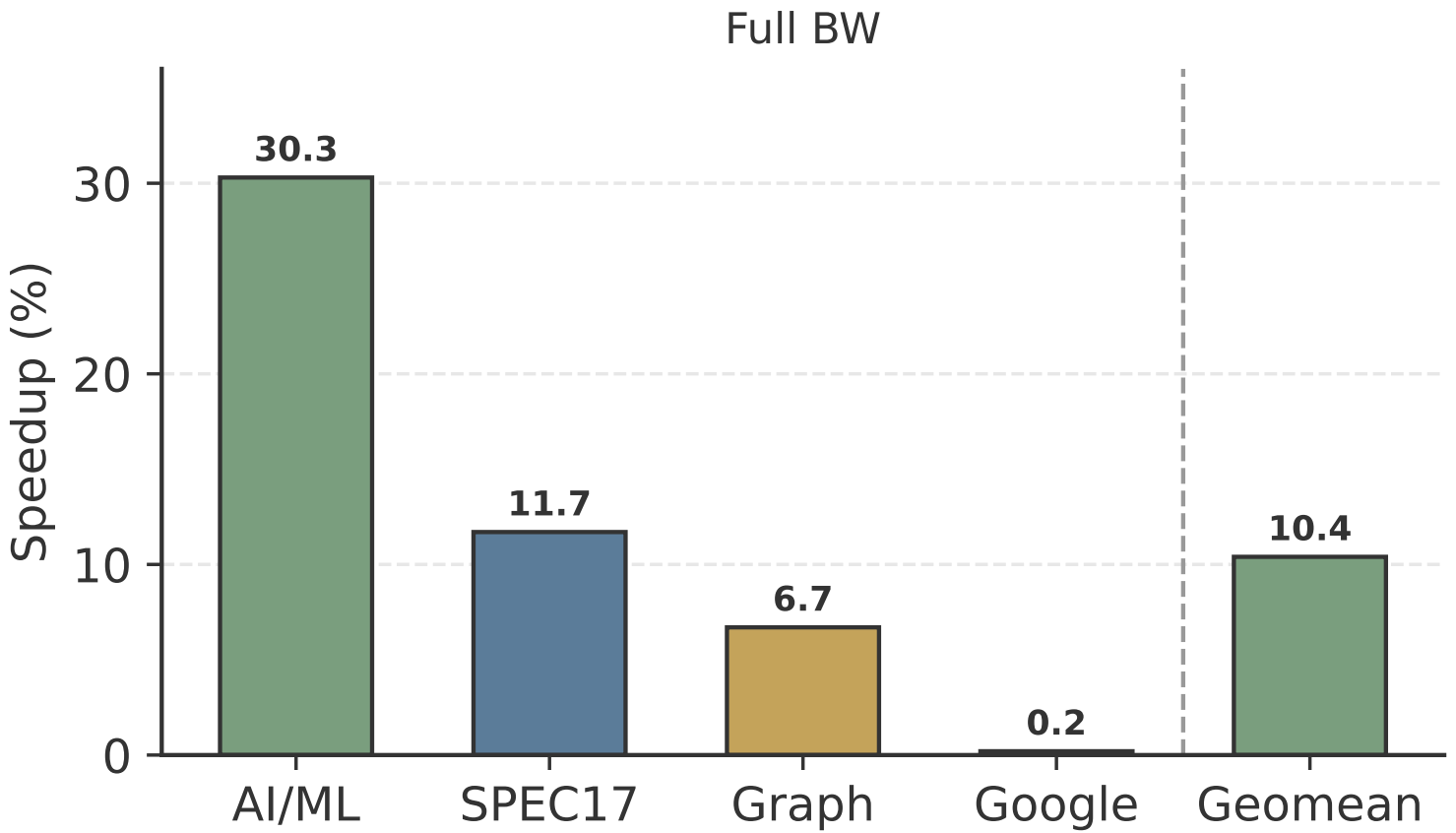
→ No eviction
→ Bit stays set
→ Future requests blocked

Storage: 15.6 KB
1360 entries · NRU
Tracks 87,040 lines

The winner of the 2026 Championship

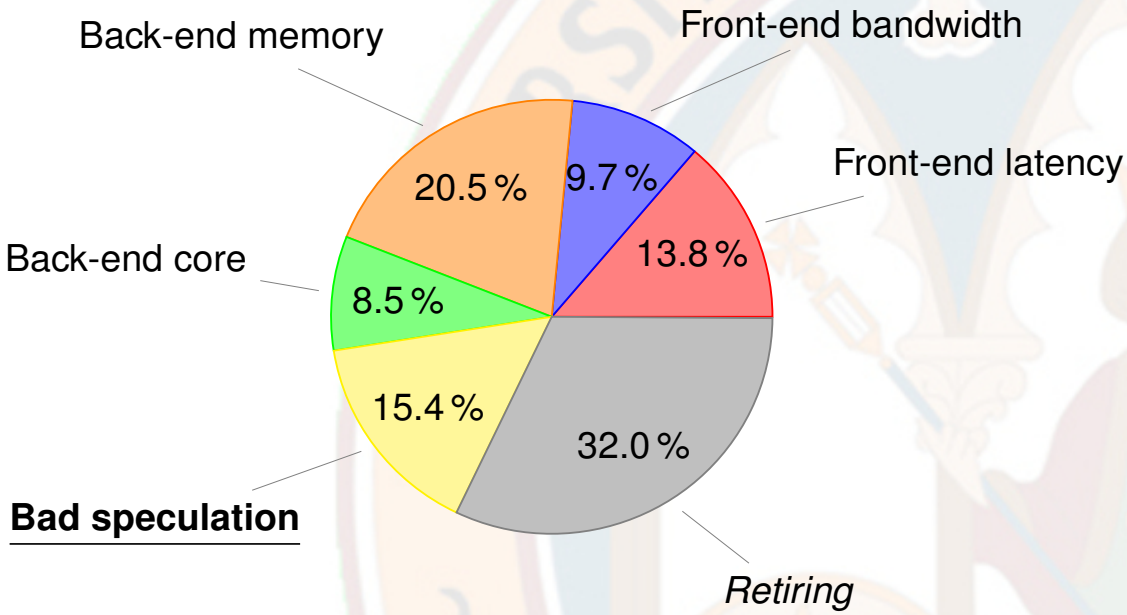
BertiGo: Results

Improvements over championship baseline!



Addressing bad speculation

(Part 3)



Overview

- Hard to improve prediction mechanisms, but still possible
- Misprediction penalties can be reduced too

Overview

- Hard to improve prediction mechanisms, but still possible
- Misprediction penalties can be reduced too

- **Solutions**
  - Improving context information of prediction mechanisms
    - For memory dependence prediction
    - For branch prediction

Overview

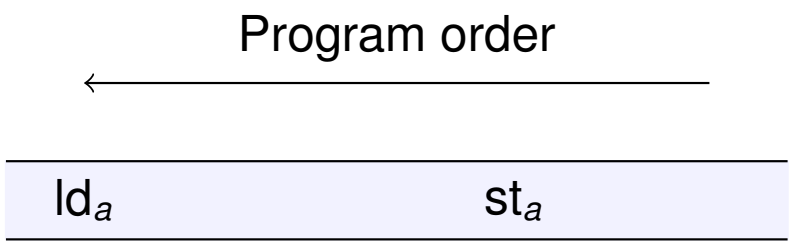
- Hard to improve prediction mechanisms, but still possible
- Misprediction penalties can be reduced too

- **Solutions**
  - Improving context information of prediction mechanisms
    - For memory dependence prediction
    - For branch prediction
  - Reduce branch misprediction penalties
    - Alternate-path uop-cache prefetching

PHAST: MEMORY DEPENDENCE PREDICTION

KNOWING THE EXACT NEEDED CONTEXT

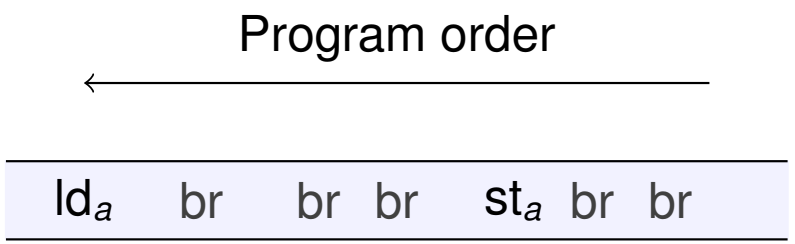
Kim & Ros [HPCA'24]



PHAST: MEMORY DEPENDENCE PREDICTION

KNOWING THE EXACT NEEDED CONTEXT

Kim & Ros [HPCA'24]



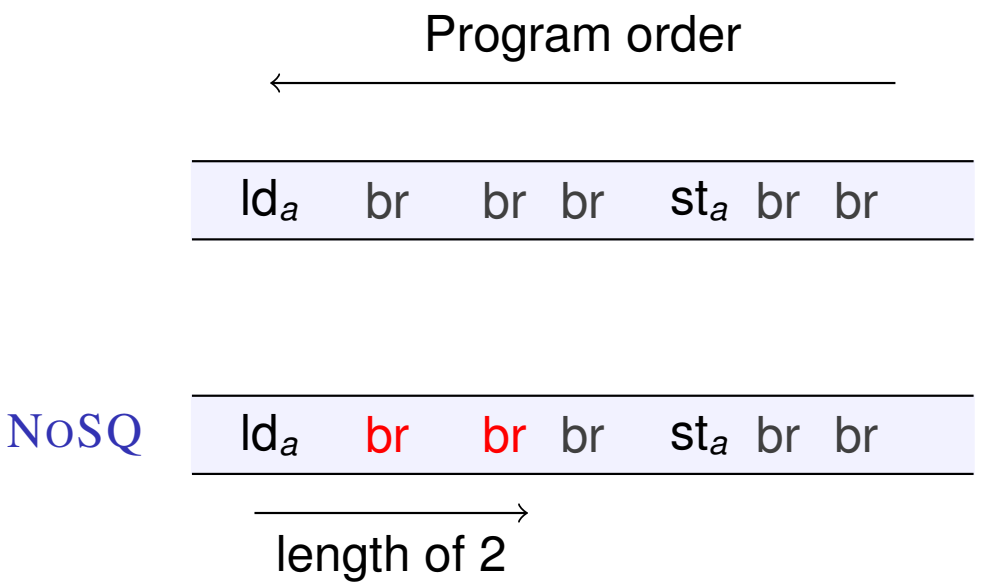
Use load's branch history

But which history length?

PHAST: MEMORY DEPENDENCE PREDICTION

KNOWING THE EXACT NEEDED CONTEXT

Kim & Ros [HPCA'24]



Use load's branch history

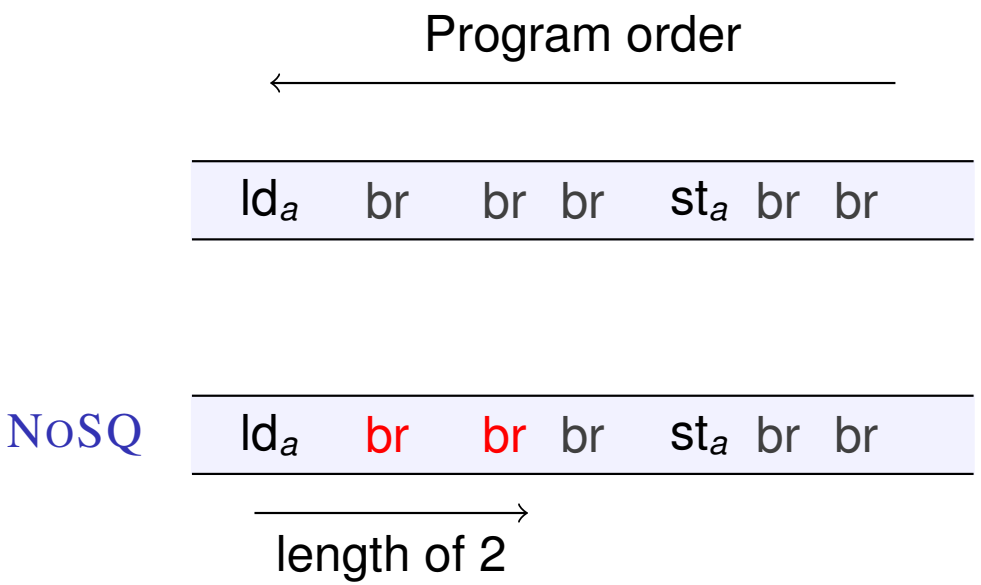
But which history length?

Predetermined history length

PHAST: MEMORY DEPENDENCE PREDICTION

KNOWING THE EXACT NEEDED CONTEXT

Kim & Ros [HPCA'24]



Use load's branch history

But which history length?

Predetermined history length

Too short? Mispredictions

Too long? Unnecessary paths

PHAST: MEMORY DEPENDENCE PREDICTION

KNOWING THE EXACT NEEDED CONTEXT

Kim & Ros [HPCA'24]

Use load's branch history

But which history length?

Predetermined history length

Too short? Mispredictions

Too long? Unnecessary paths

MDP-TAGE

Geometrically increasing history lengths

Brute-force method to find the right lenght

PHAST: MEMORY DEPENDENCE PREDICTION

KNOWING THE EXACT NEEDED CONTEXT

Kim & Ros [HPCA'24]

Use load's branch history

But which history length?

Predetermined history length

Too short? Mispredictions

Too long? Unnecessary paths

MDP-TAGE

Geometrically increasing history lengths

Brute-force method to find the right lenght

- Previous context-sensitive proposals borrow branch-prediction techniques
  - Use predetermined history lengths ⇒ sub-optimal

PHAST: Memory dependence prediction

Knowing the exact needed context

- PHAST uses the proper history length for each conflict
  - $\Rightarrow$ The one that defines the path from the store to the load
    - Intuition: If the exact execution path repeats, it is likely that the dependence repeats, too

PHAST: Memory dependence prediction

Knowing the exact needed context

- PHAST uses the proper history length for each conflict
  - $\Rightarrow$ The one that defines the path from the store to the load
    - Intuition: If the exact execution path repeats, it is likely that the dependence repeats, too

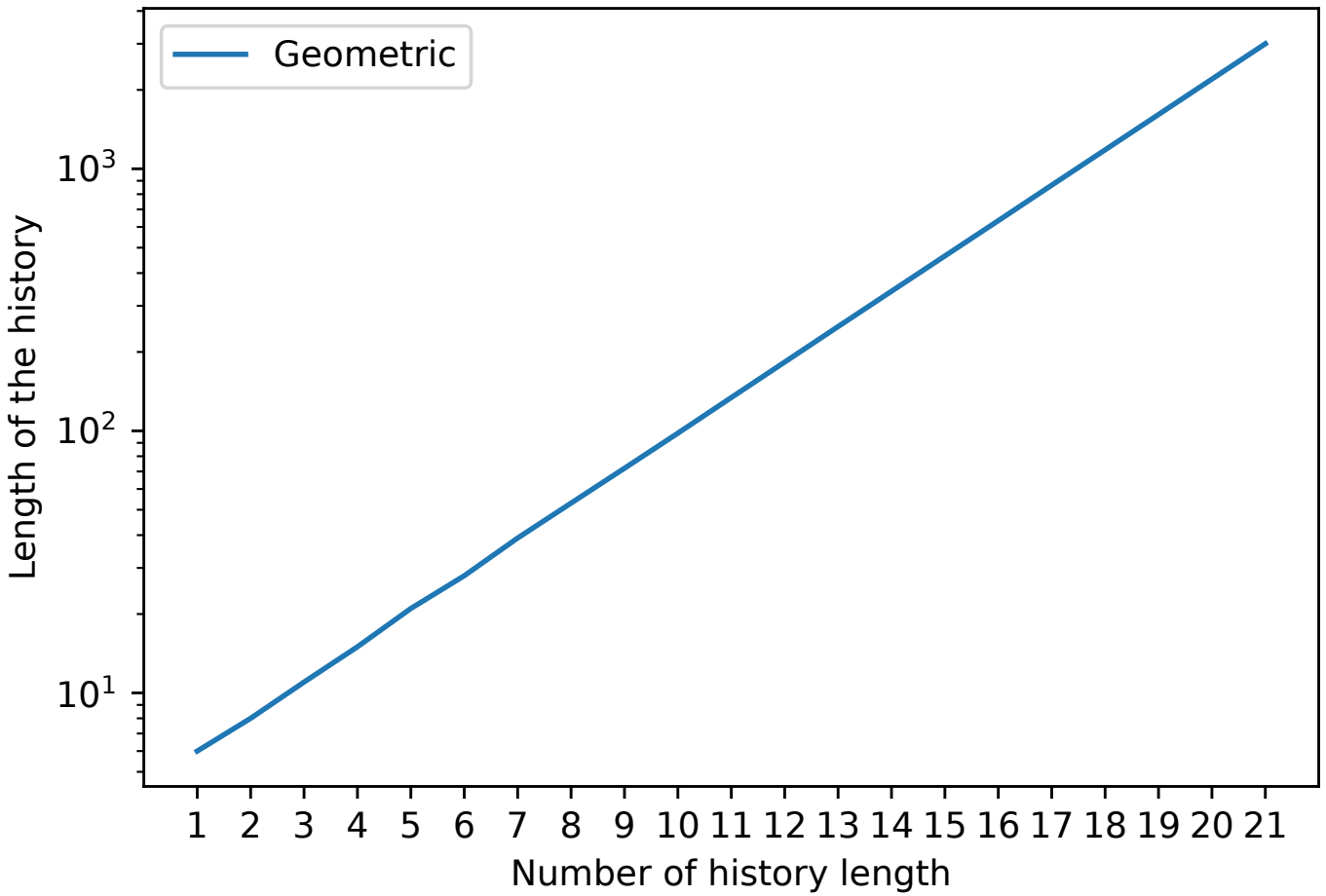
- To identify the path, PHAST only considers divergent branches (conditional and indirect branches):
  - Conditional branches $\rightarrow$ taken/not taken bit
  - Indirect branches $\rightarrow$ target address

IMPROVING BRANCH PREDICTION

HISTORY IS NOT HOW IT WAS TOLD

Ros [CBP'25]

- TAGE (and other predictors) use GEometric history lengths

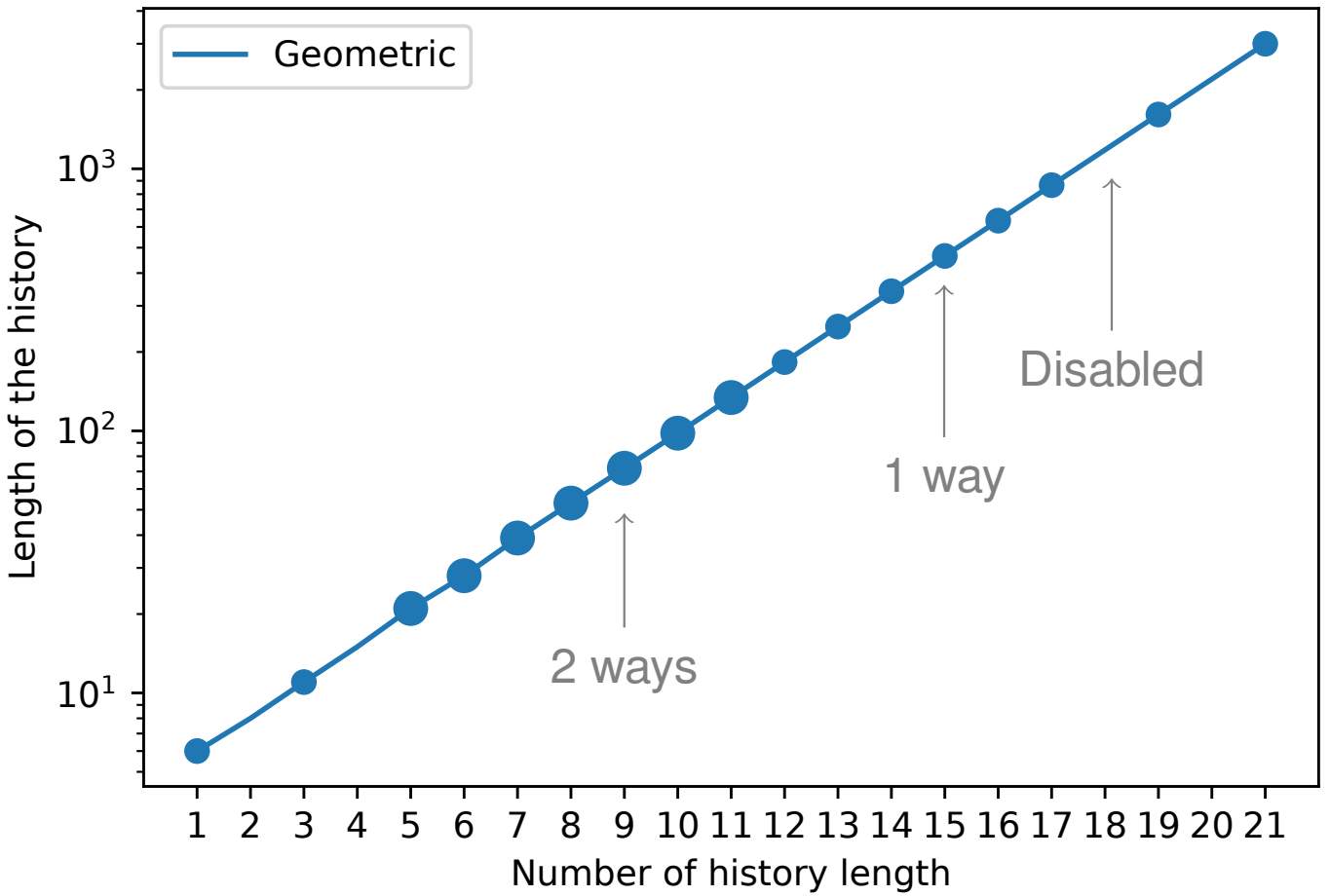


IMPROVING BRANCH PREDICTION

HISTORY IS NOT HOW IT WAS TOLD

Ros [CBP'25]

- TAGE (and other predictors) use GEometric history lengths

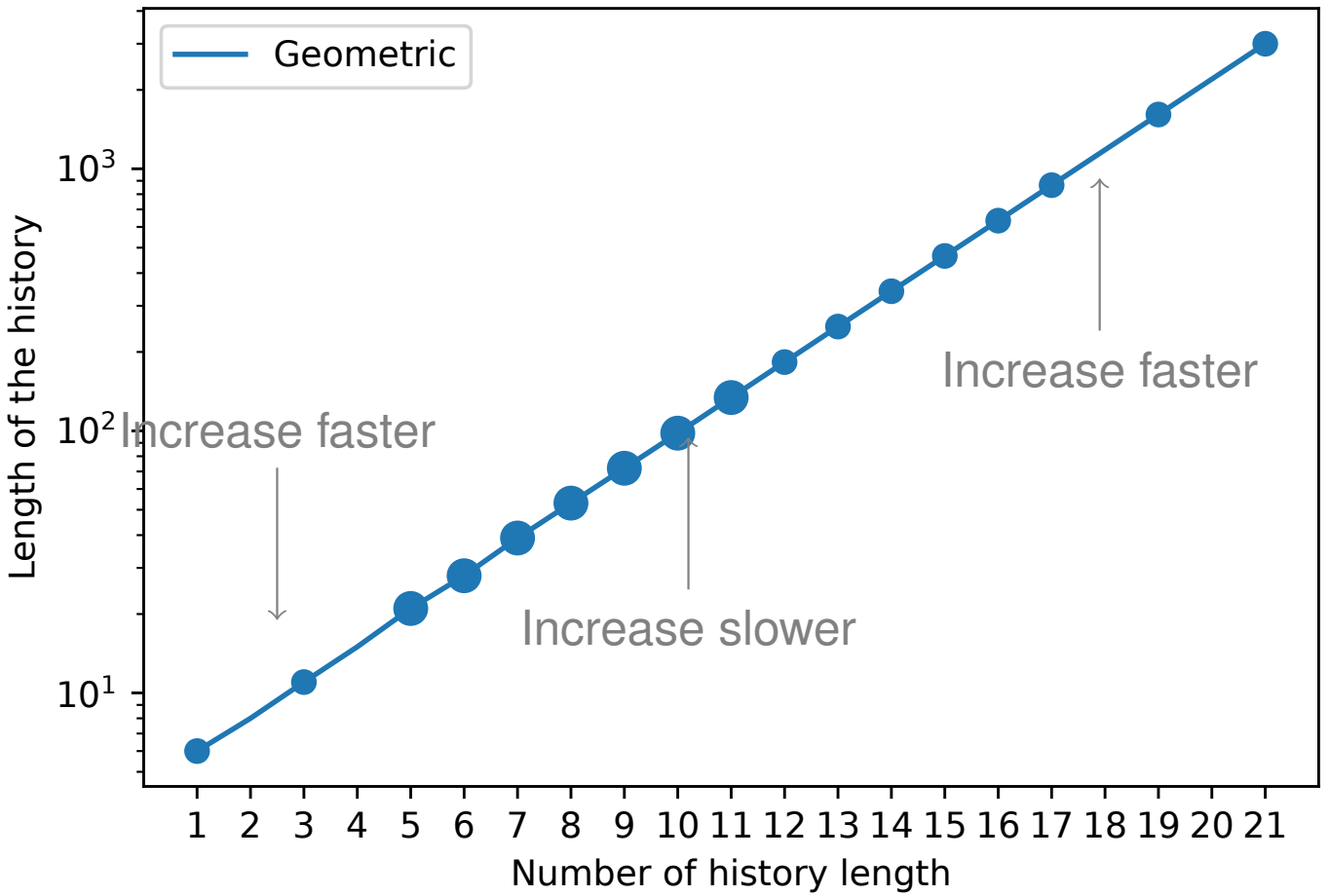


IMPROVING BRANCH PREDICTION

HISTORY IS NOT HOW IT WAS TOLD

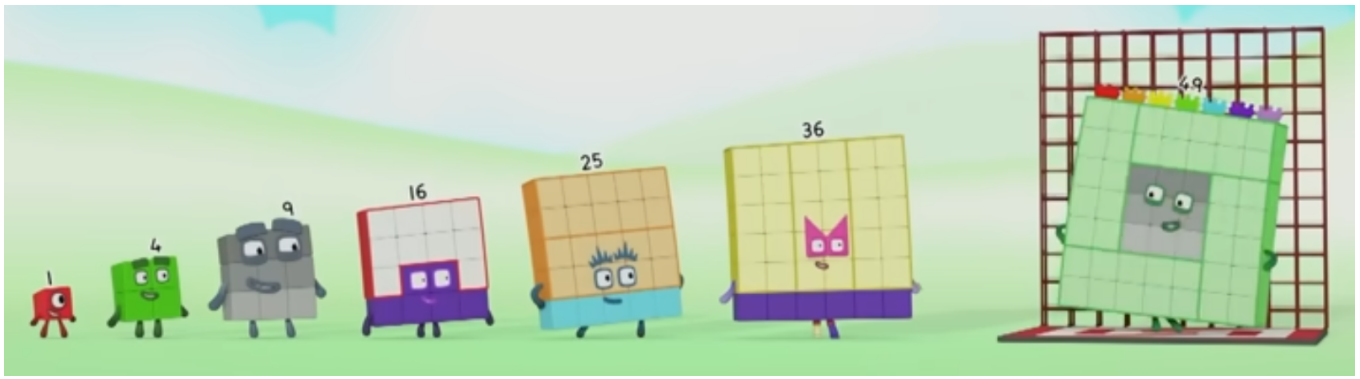
Ros [CBP'25]

- TAGE (and other predictors) use GEometric history lengths



IMPROVING BRANCH PREDICTION
HISTORY IS NOT HOW IT WAS TOLD

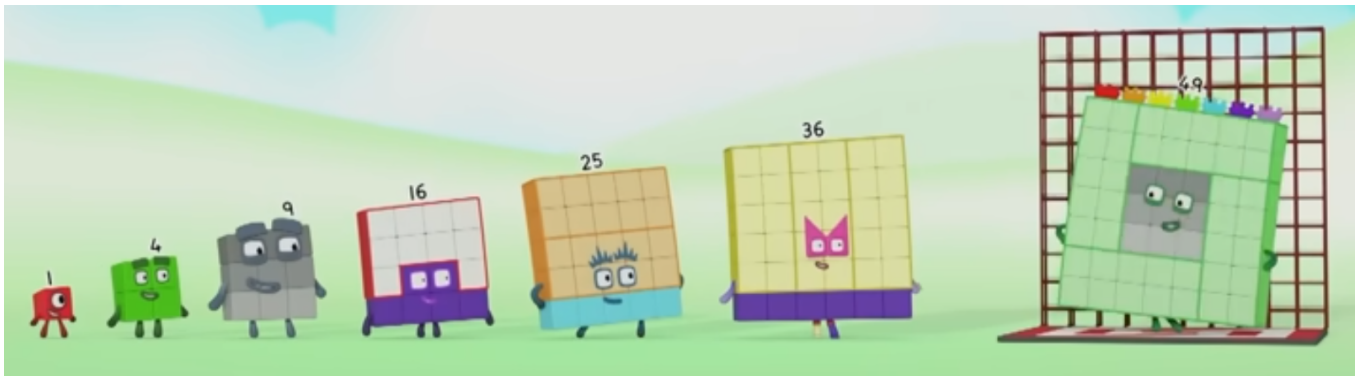
- First approach: Square party!



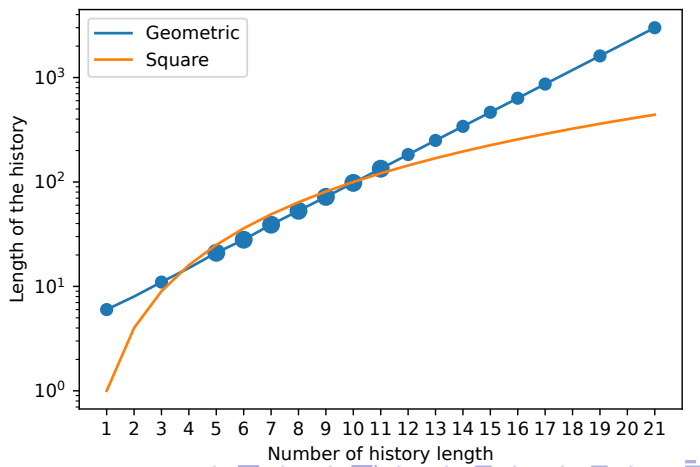
IMPROVING BRANCH PREDICTION

HISTORY IS NOT HOW IT WAS TOLD

- First approach: Square party!



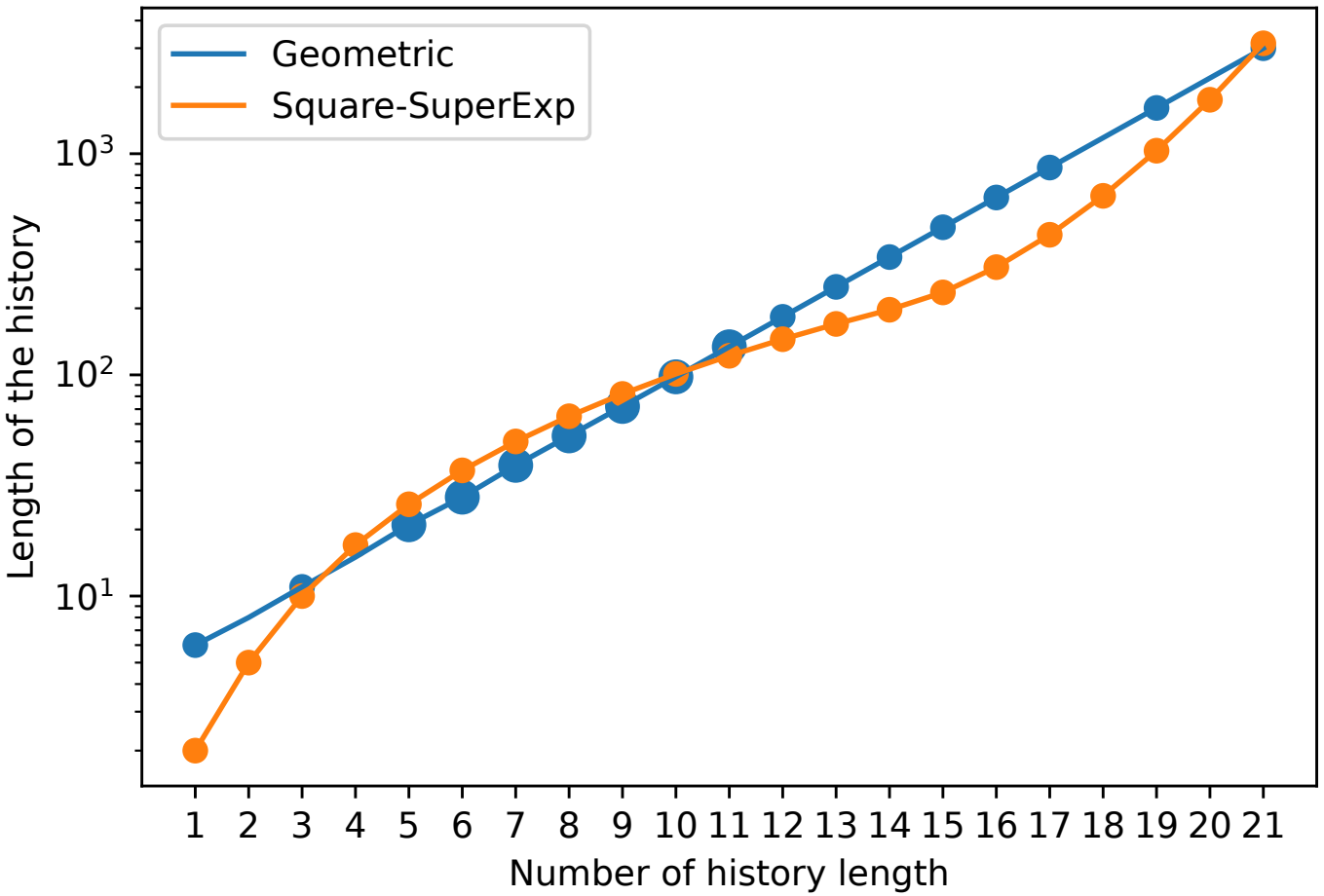
- Starts increasing faster than the geometric series but after some point it goes slower



IMPROVING BRANCH PREDICTION

HISTORY IS NOT HOW IT WAS TOLD

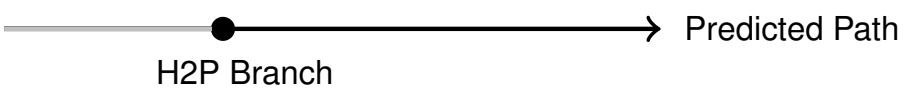
- Solution: At some point in the square sequence switch to a generalized geometric sequence



ALTERNATE PATH μ-OP CACHE PREFETCHING

Singh, Perais, Jimborean & Ros [ISCA'24]

① Identify hard-to-predict branches
$\Rightarrow$ A branch with high chances of being mispredicted[7]



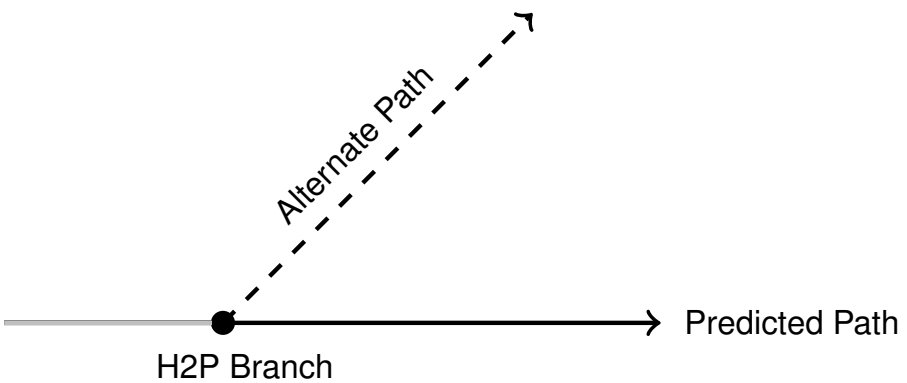
[7] Seznec et al. "Storage free confidence estimation for TAGE..." [HPCA'11]

ALTERNATE PATH μ-OP CACHE PREFETCHING

Singh, Perais, Jimborean & Ros [ISCA'24]

① Identify hard-to-predict branches
⇒ A branch with high chances of being mispredicted[7]

② Generate addresses on alternate path
⇒ Duplicating/banking front-end predictors



[7] Seznec et al. "Storage free confidence estimation for TAGE..." [HPCA'11]

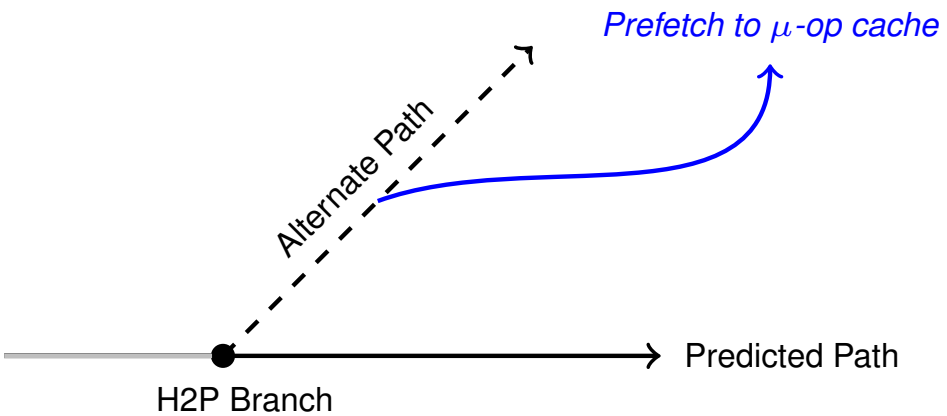
ALTERNATE PATH μ-OP CACHE PREFETCHING

Singh, Perais, Jimborean & Ros [ISCA'24]

① Identify hard-to-predict branches
⇒ A branch with high chances of being mispredicted[7]

② Generate addresses on alternate path
⇒ Duplicating/banking front-end predictors

③ Prefetch μ-ops from alternate path



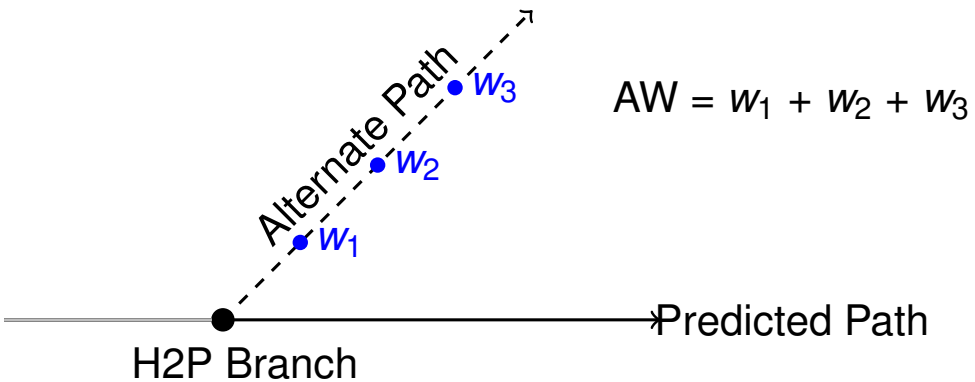
[7] Seznec et al. "Storage free confidence estimation for TAGE..." [HPCA'11]

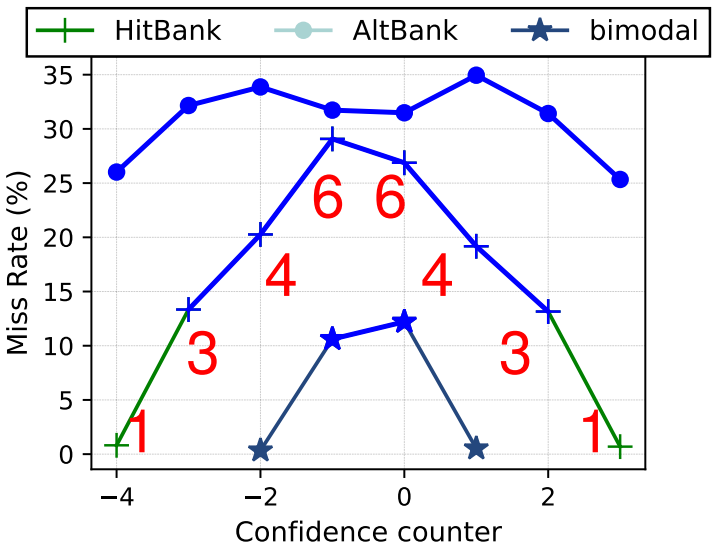
Alternate Path μ-op Cache Prefetching

② Generating Addresses

→ When to stop?

- Weight of each branch on the alternate path is accumulated
  - High confident → lower weight
  - Lower confident → higher weight
- BTB miss on the alternate path
- New H2P branch is detected





FINAL REMARKS

$\Rightarrow$ Prefetching has the potential to remove most stalls

1. INSTRUCTION PREFETCHING (and replacement policy)
   - Reduces the front-end latency problem
   - Bring/keep critical instructions close to the processor
2. DATA PREFETCHING (if secure better)
   - Reduces the back-end memory problem
   - Timeliness is a key factor
3. BRANCH/MEMORY DEPENDENCE PREDICTION
   - Reduces the bad speculation problem
   - Use the appropriate context information
4. μ-OP CACHE PREFETCHING
   - Reduces the bad speculation problem
   - Recover faster by prefetching wrong-path instructions

Intelligent Memory Prefetching and Prediction in Modern Processors

Alberto Ros

University of Murcia, Spain

Thank you!

ECHO, ERC Consolidator Grant (No 819134)

Overview: Instruction prefetcher

- Server and cloud apps getting larger, far from fitting in L1I
  - $\Rightarrow$ stalls processor front-end, performance degradation

Overview: Instruction prefetcher

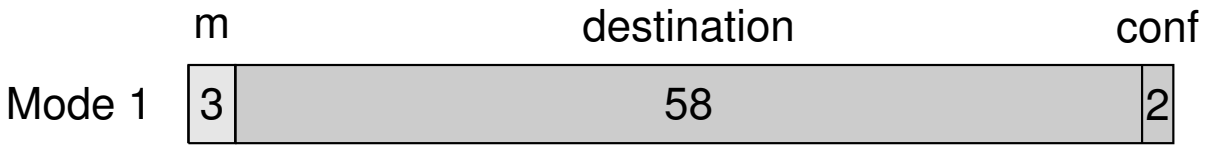
- Server and cloud apps getting larger, far from fitting in L1I
  - $\Rightarrow$ stalls processor front-end, performance degradation
- Prefetching instructions is fundamental for performance
  - Even when a decoupled front-end is implemented

Overview: Instruction prefetcher

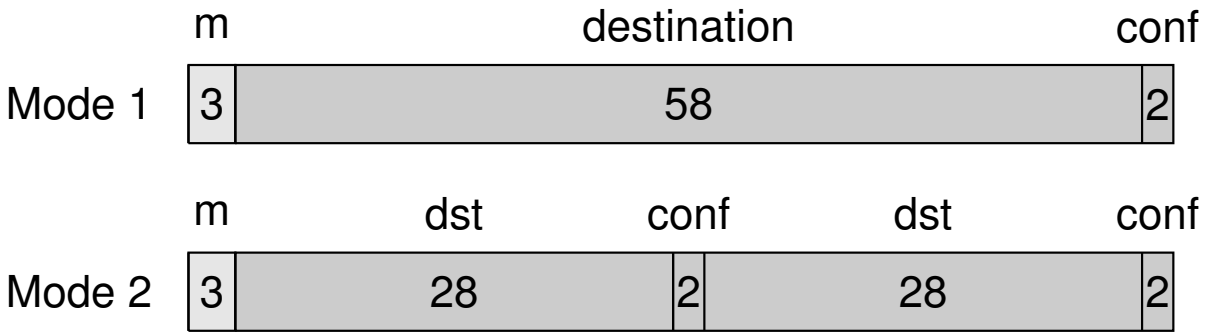
- Server and cloud apps getting larger, far from fitting in L1I
  - $\Rightarrow$ stalls processor front-end, performance degradation
- Prefetching instructions is fundamental for performance
  - Even when a decoupled front-end is implemented
- Our contribution: An Entangling prefetcher
  - Entangling: adaptive correlation based on latency
  - Winner of the 1st Instruction Prefetching Championship
  - A cost-effective prefetcher
  - Prefetcher code is available[1]

[1] https://github.com/alberto-ros/EntanglingInstructionPrefetcher

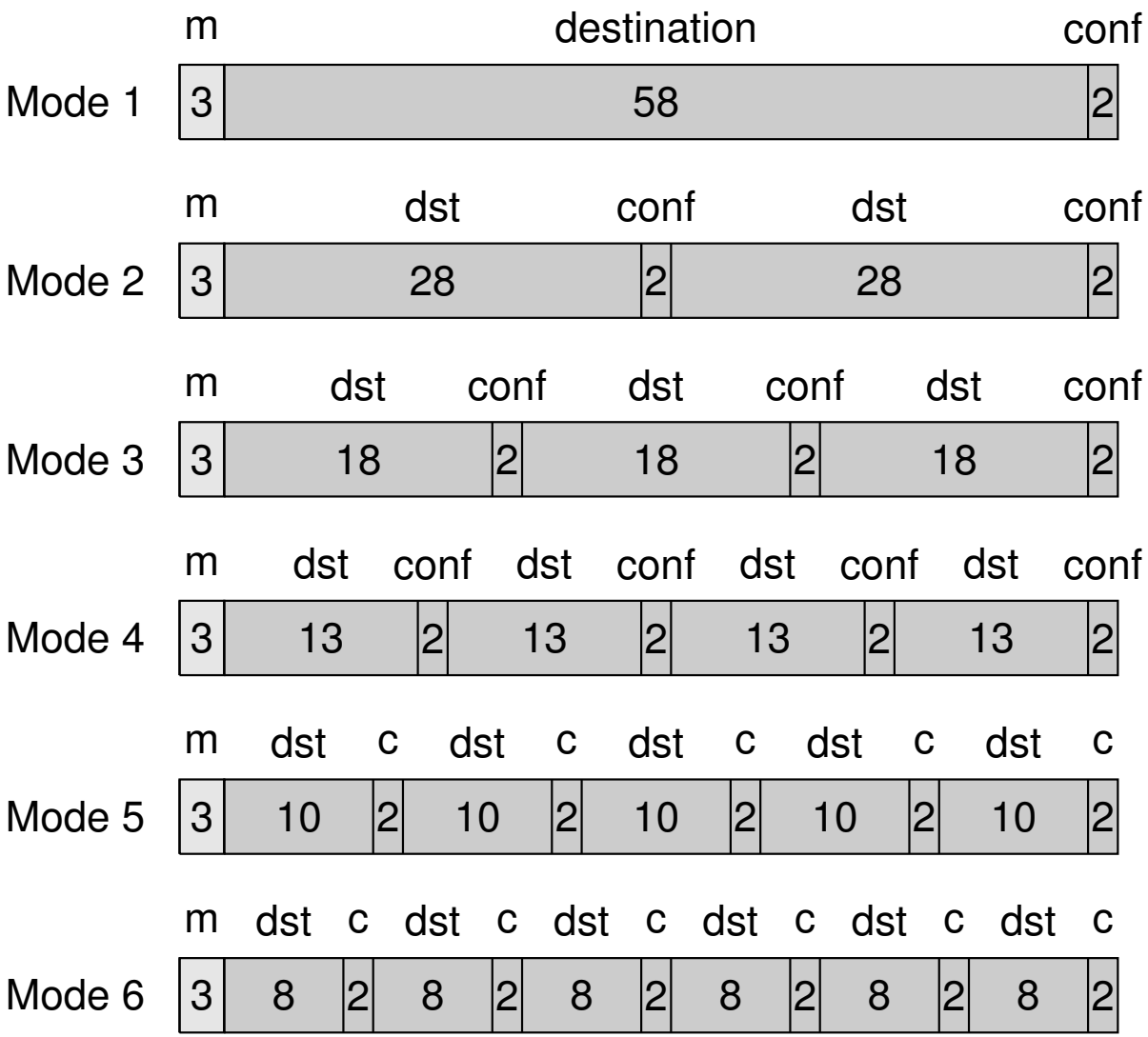
COMPRESSING DESTINATIONS



COMPRESSING DESTINATIONS



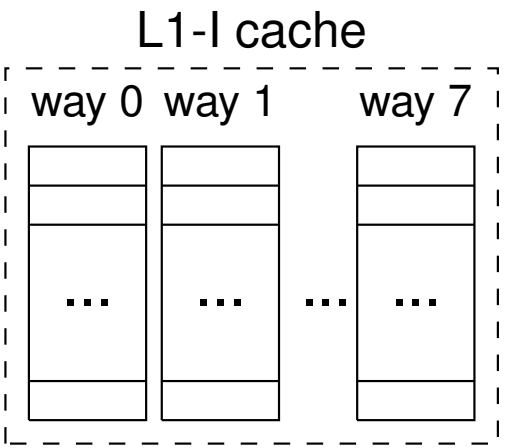
COMPRESSING DESTINATIONS



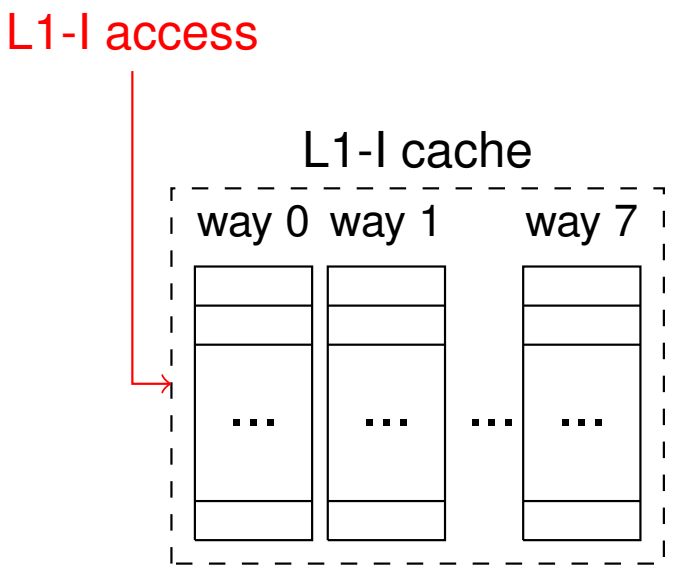
METHODOLOGY

- ChampSim develop branch (nov 2020)
- Baseline:
  - Sunny Cove-like system
  - Decoupled front-end (64-entry fetch queue)
  - 32KB L1I
- ENTANGLED:
  - *History buffer*: 16 entries
  - *Entangled table*: 2K, 4K and 8K entries
- Applications
  - 959 traces from the Championship Value Prediction (provided by Qualcomm)
  - Cloud Suite
- Analysis both for virtual and physical prefetching

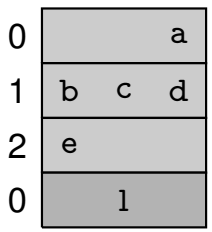
Design of the Entangling Prefetcher

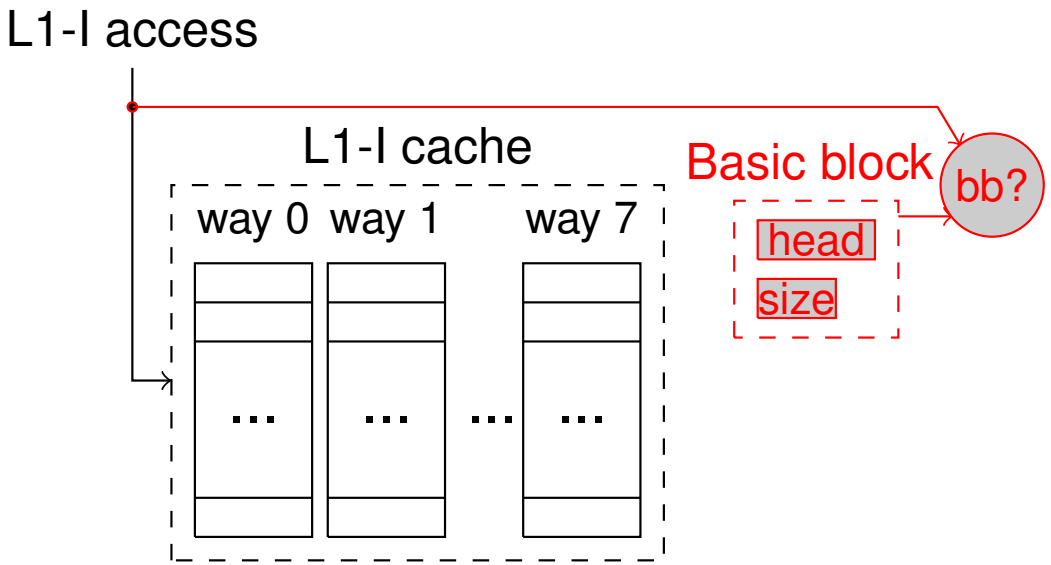


Design of the Entangling Prefetcher

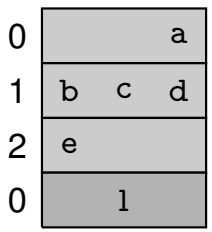


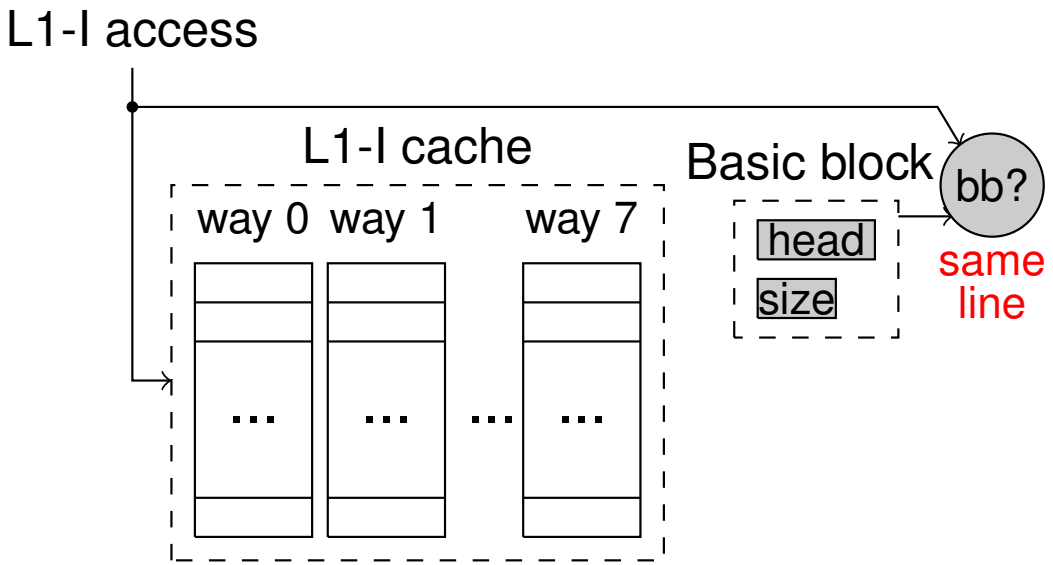
Design of the Entangling Prefetcher - Finding basic blocks



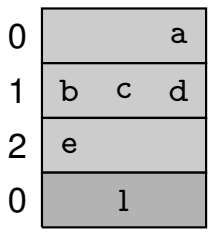


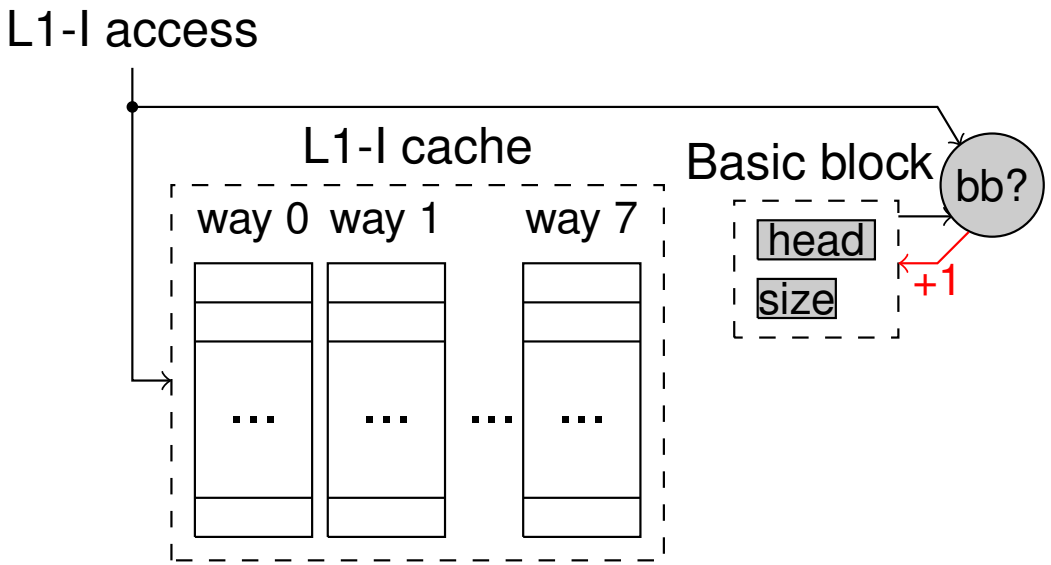
Design of the Entangling Prefetcher - Finding basic blocks



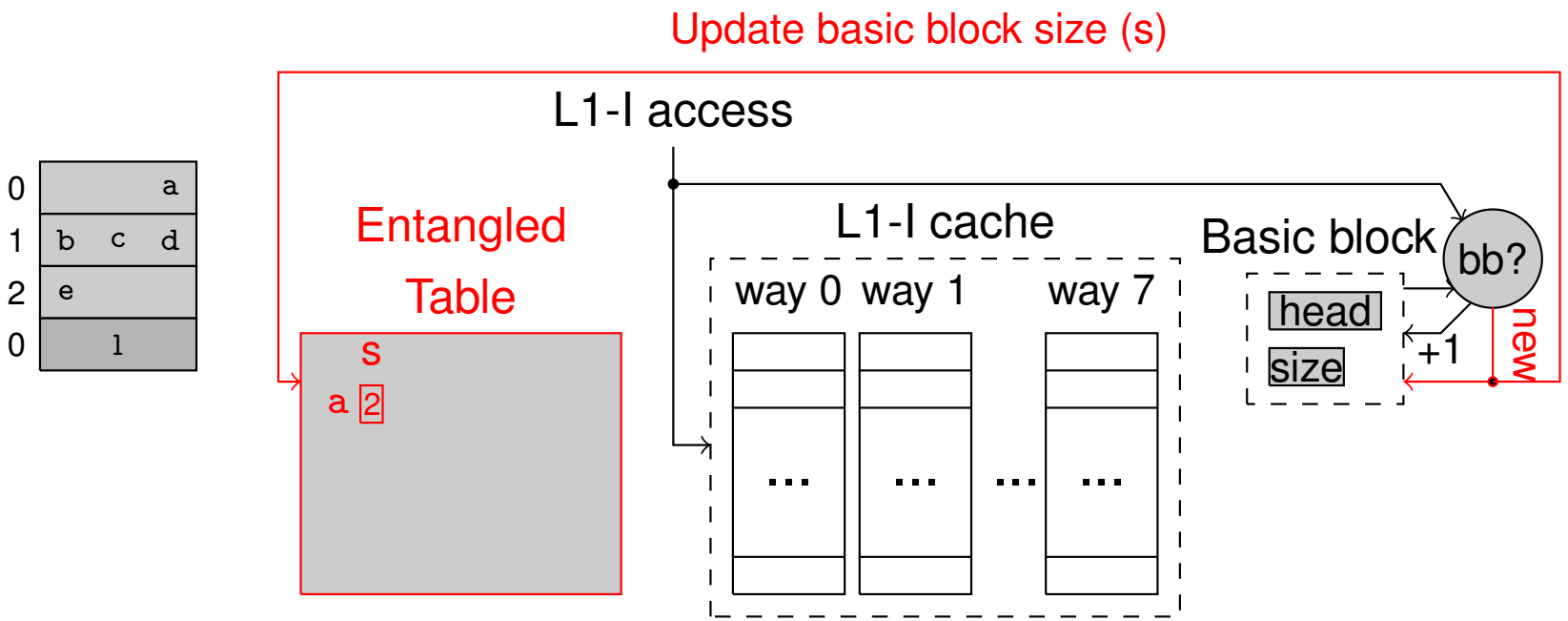


Design of the Entangling Prefetcher - Finding basic blocks

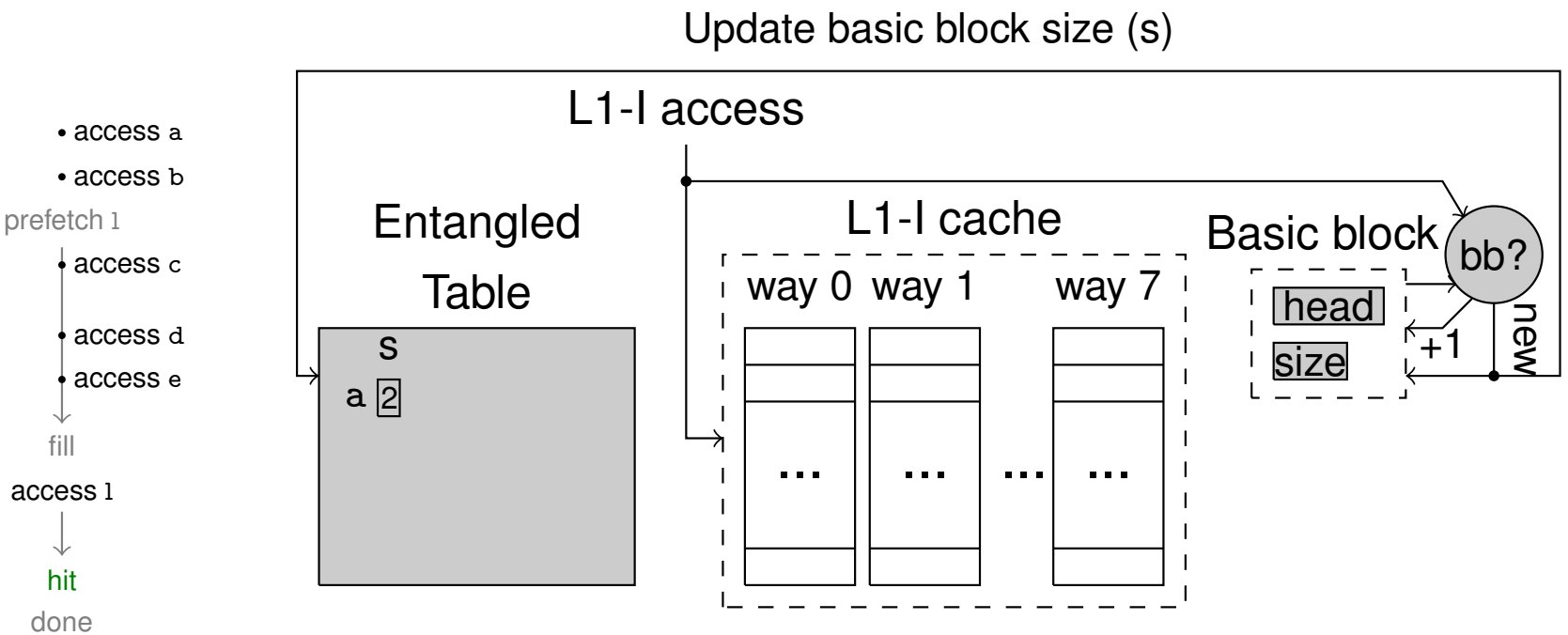




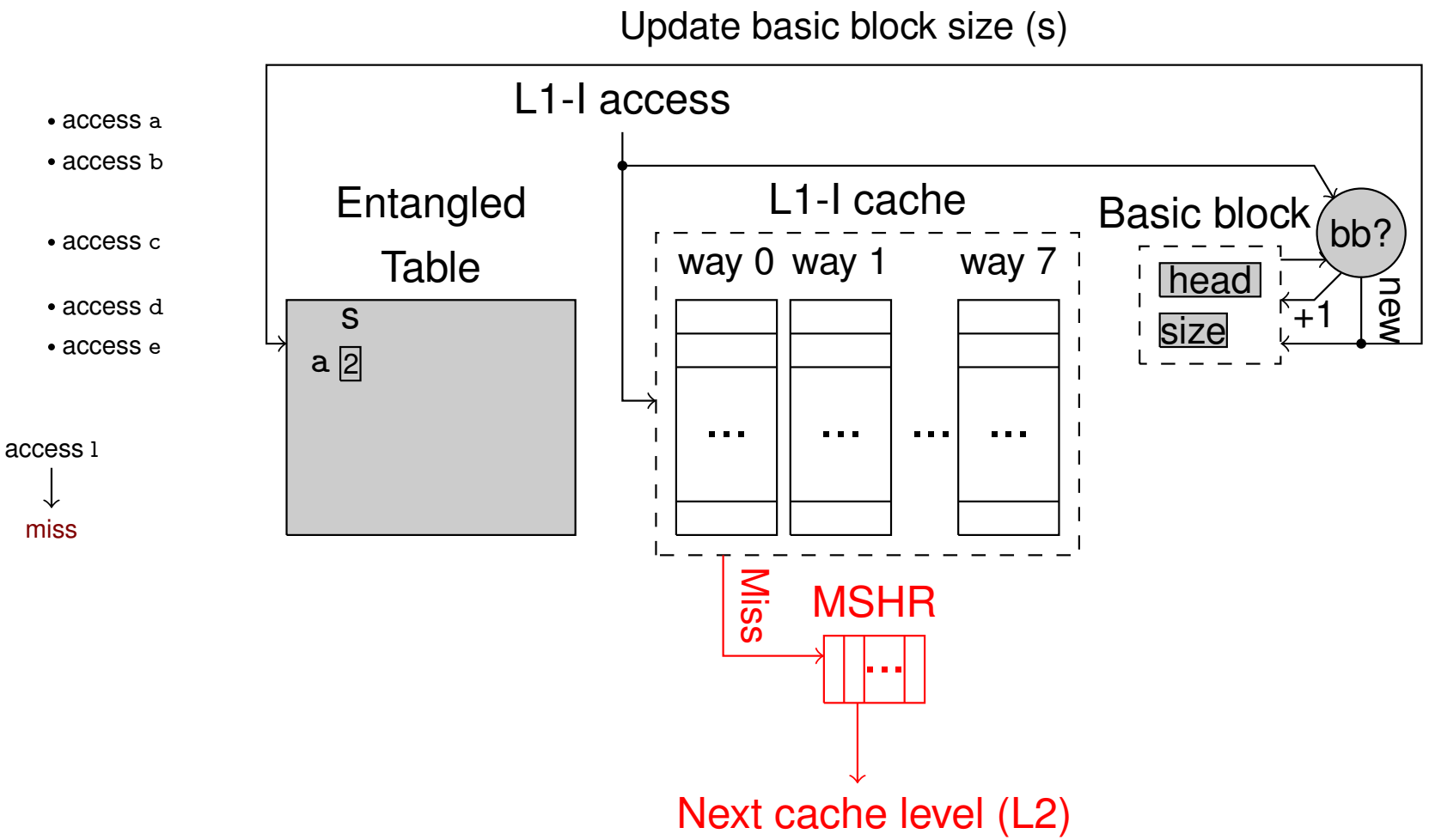
Design of the Entangling Prefetcher - Finding basic blocks



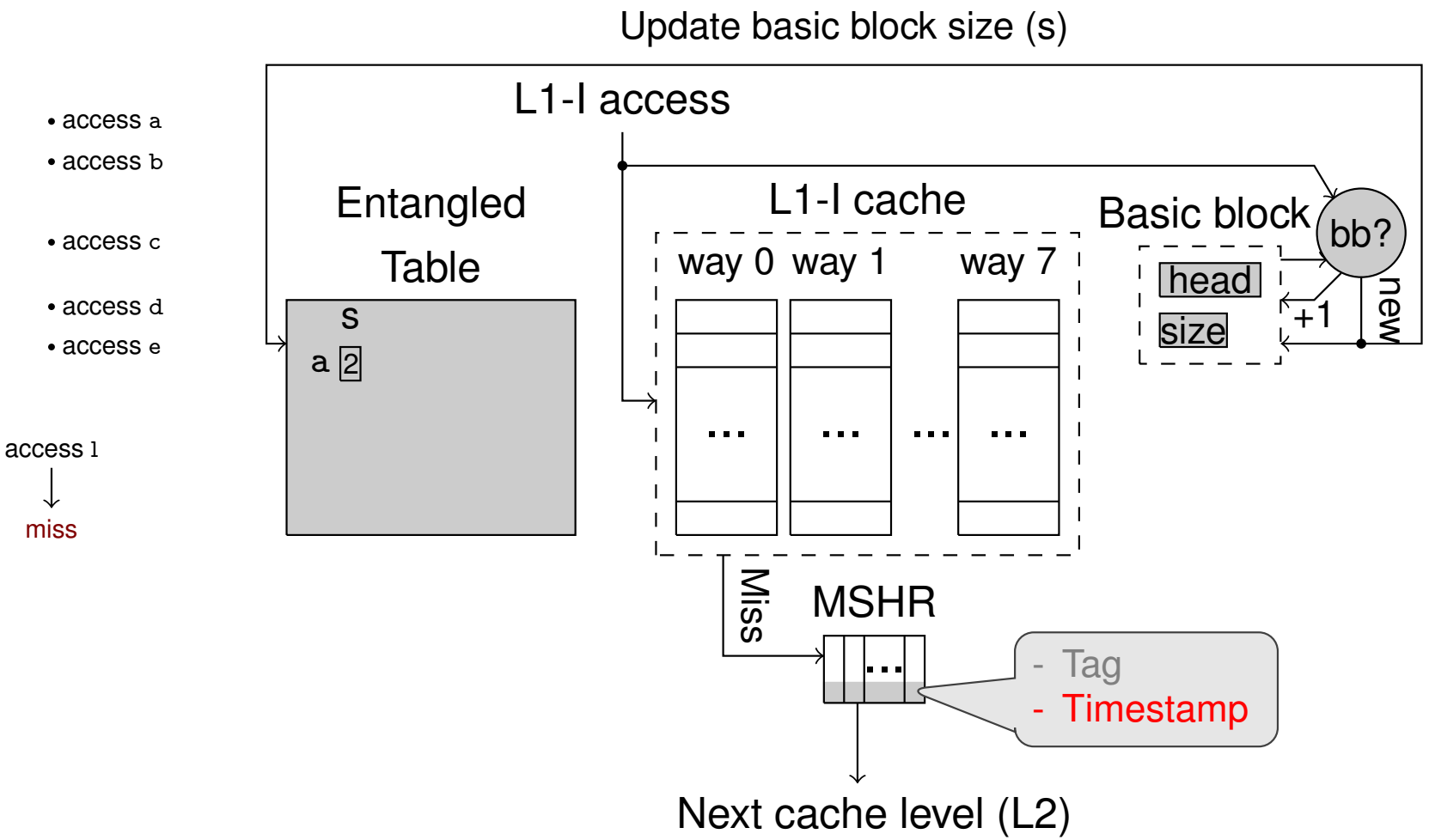
Design of the Entangling Prefetcher - Entangling cache lines



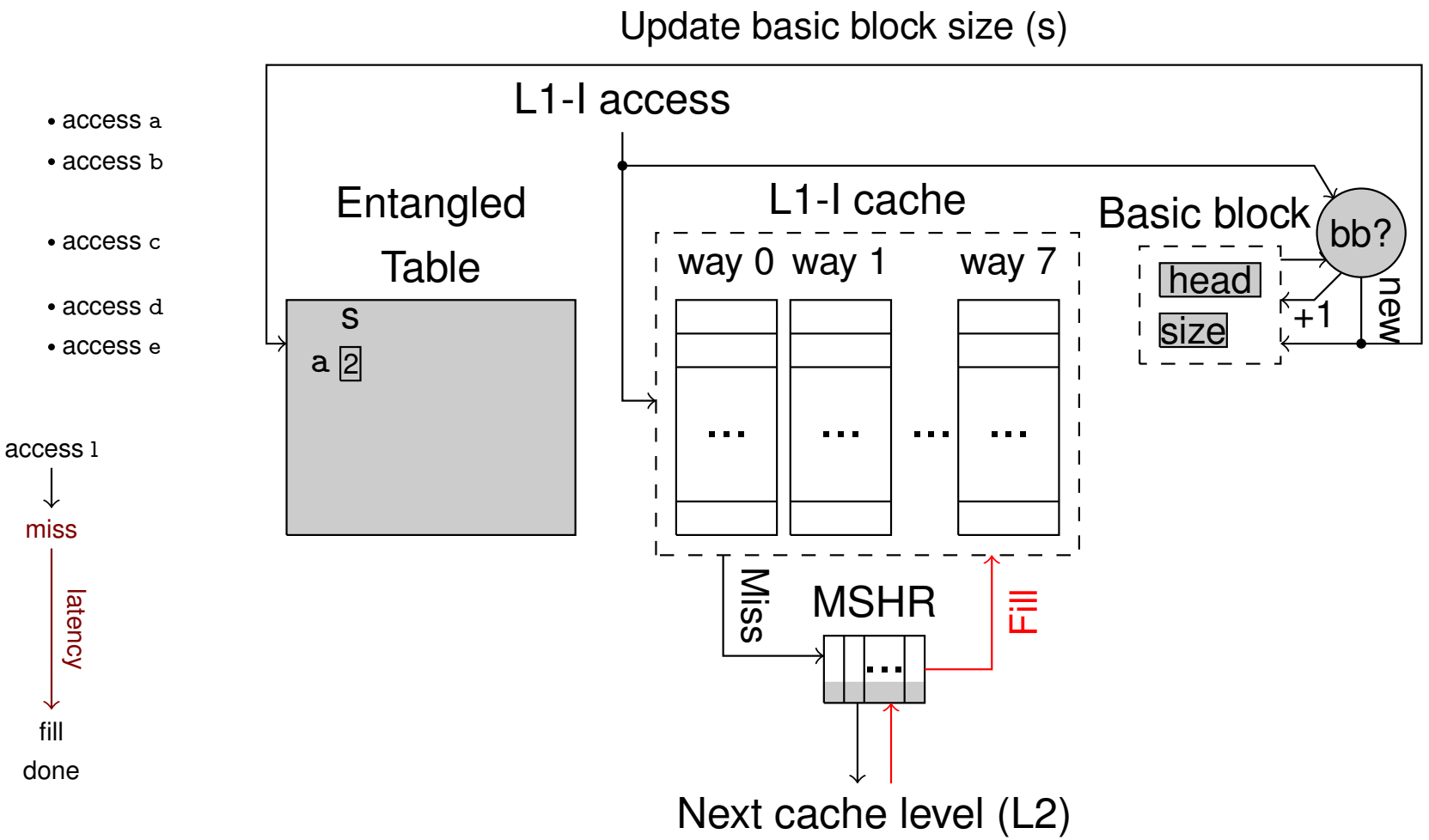
Design of the Entangling Prefetcher - Entangling cache lines



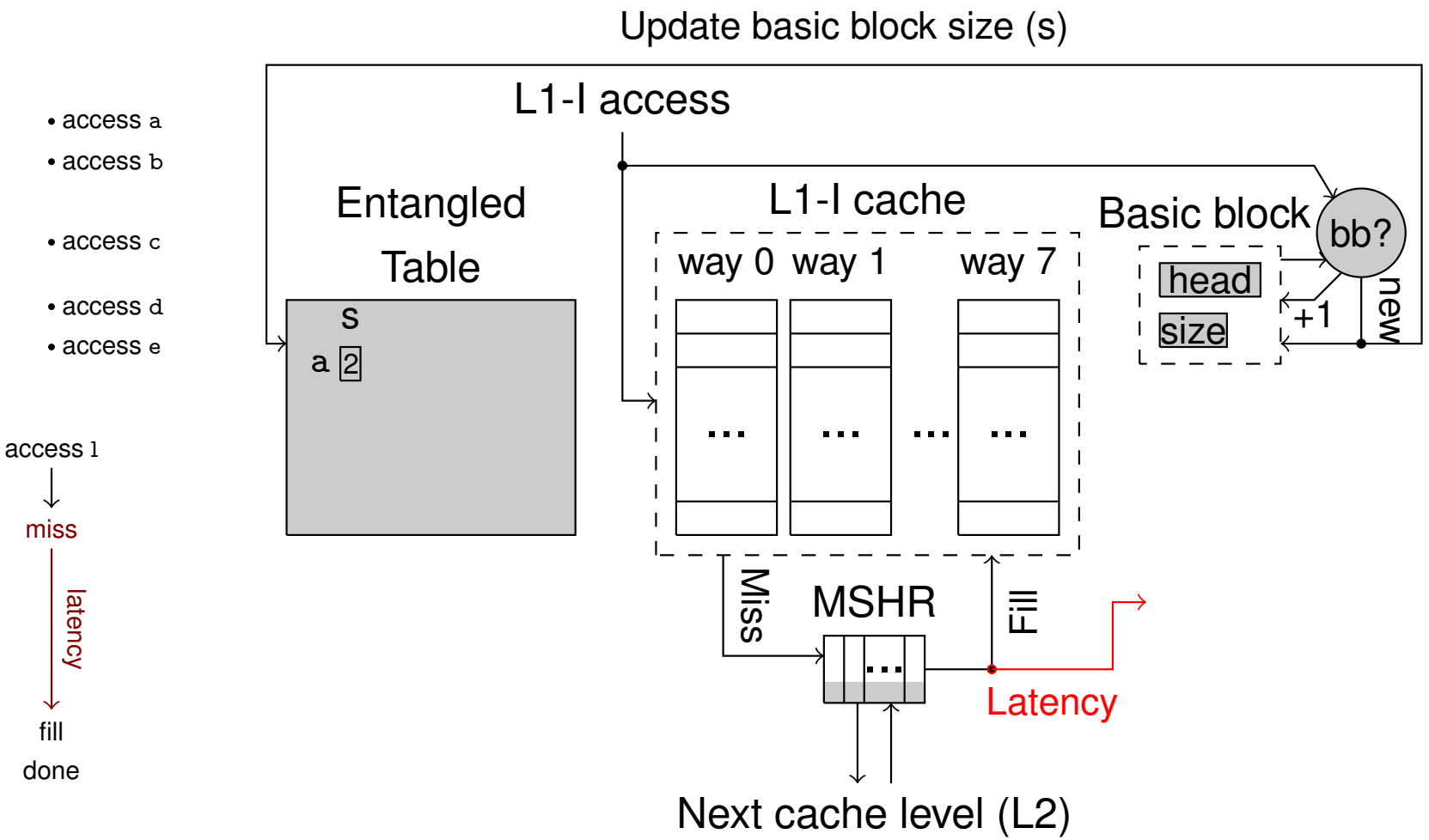
Design of the Entangling Prefetcher - Entangling cache lines



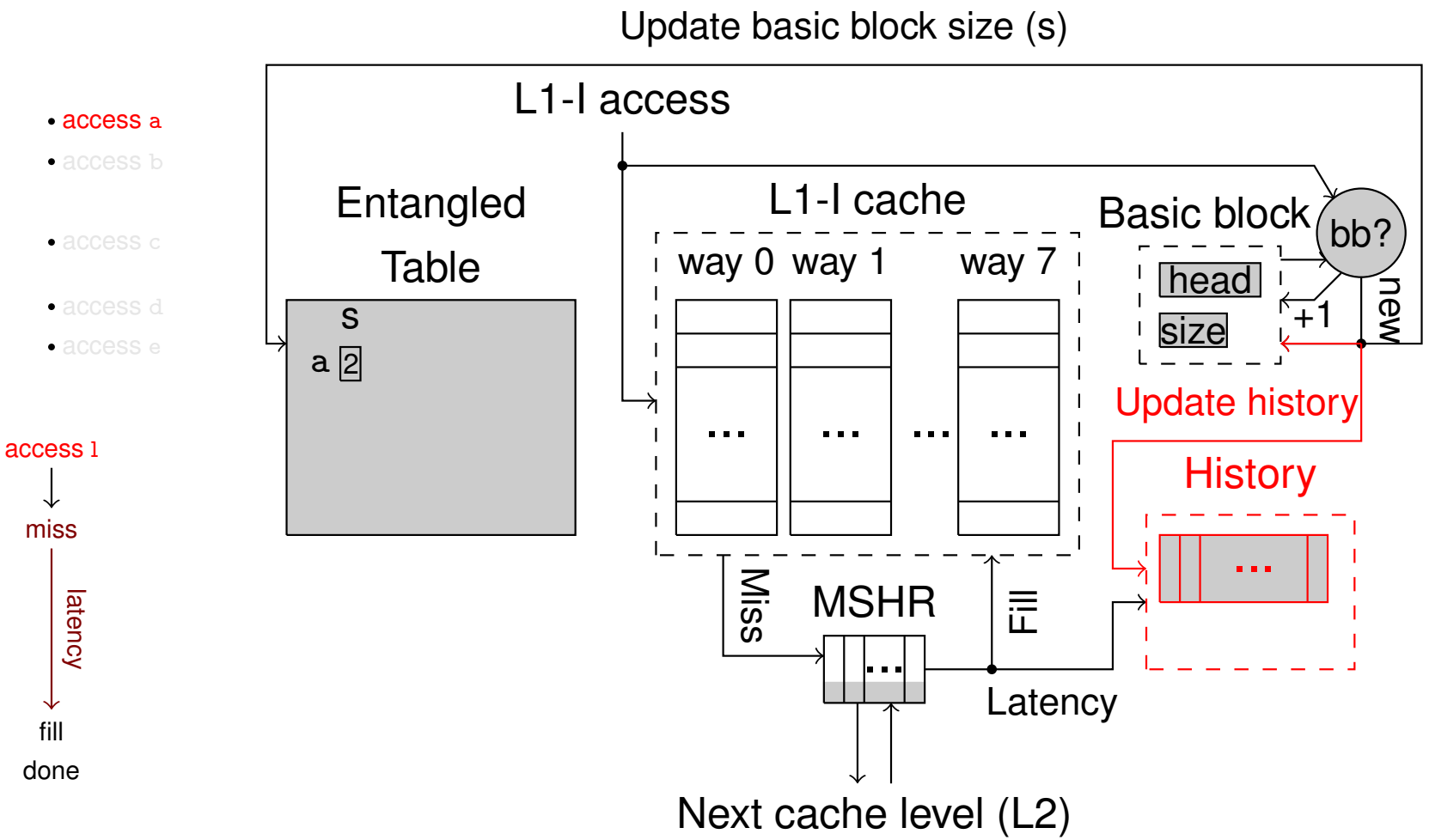
Design of the Entangling Prefetcher - Entangling cache lines



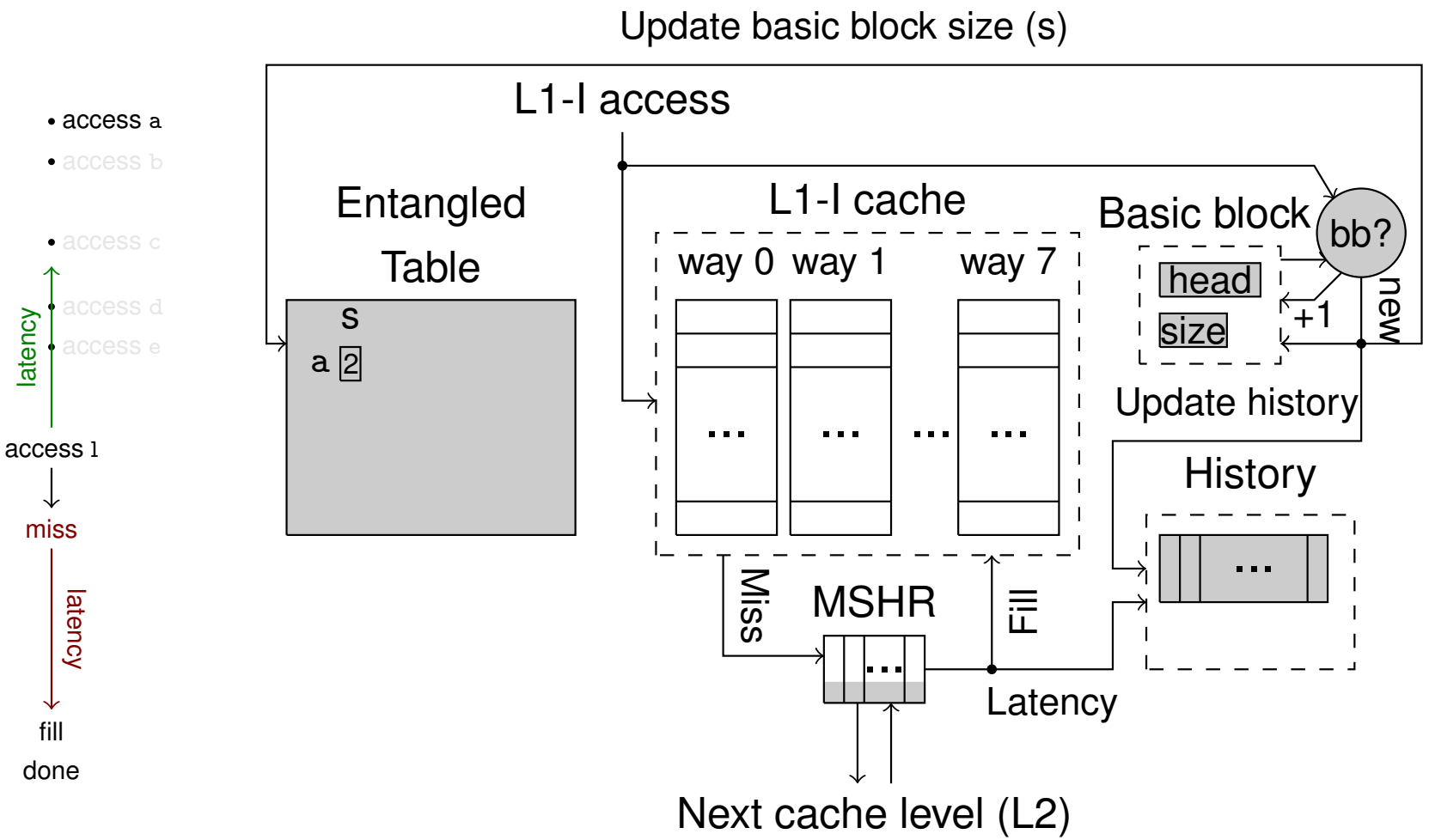
Design of the Entangling Prefetcher - Entangling cache lines



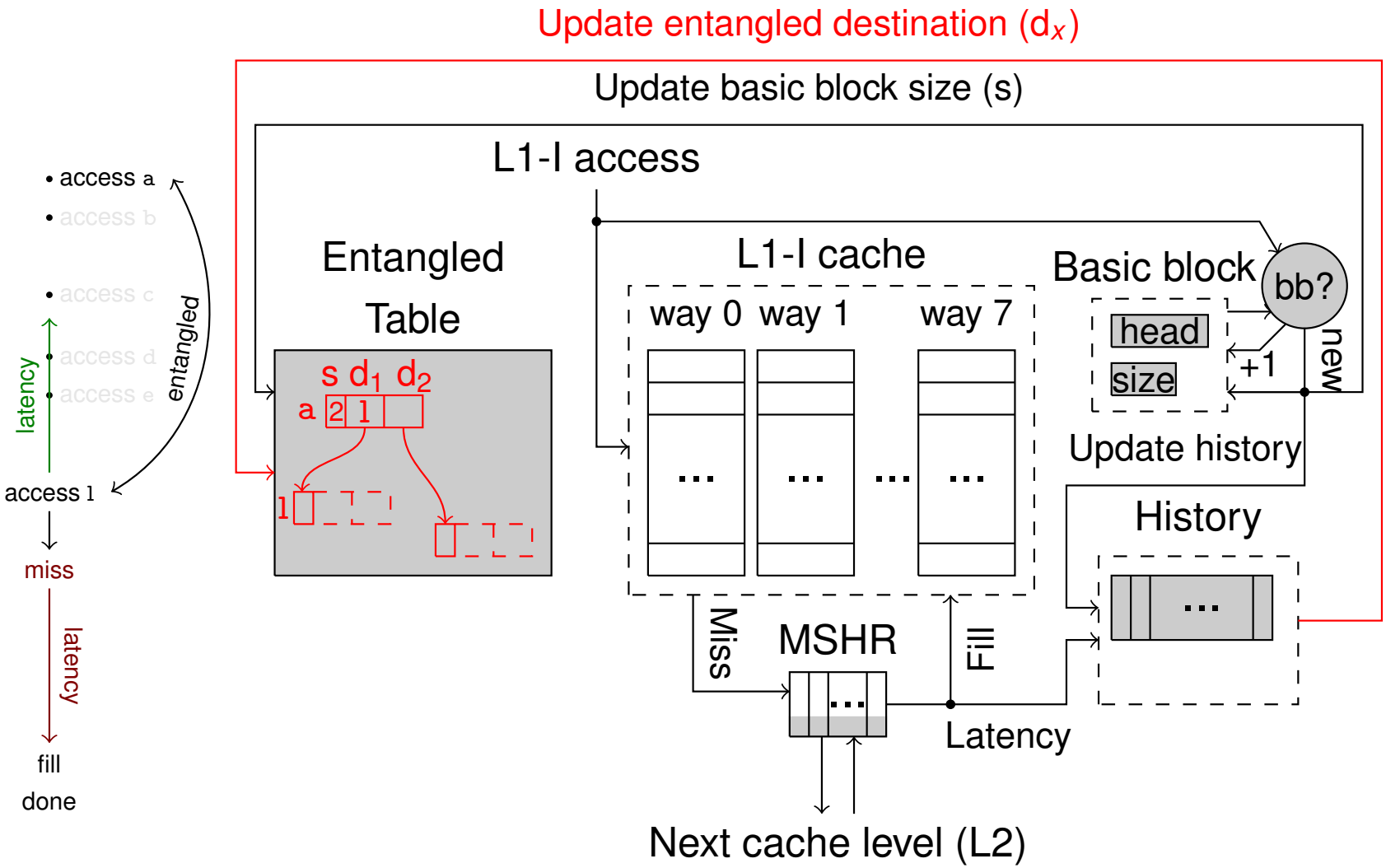
Design of the Entangling Prefetcher - Entangling cache lines



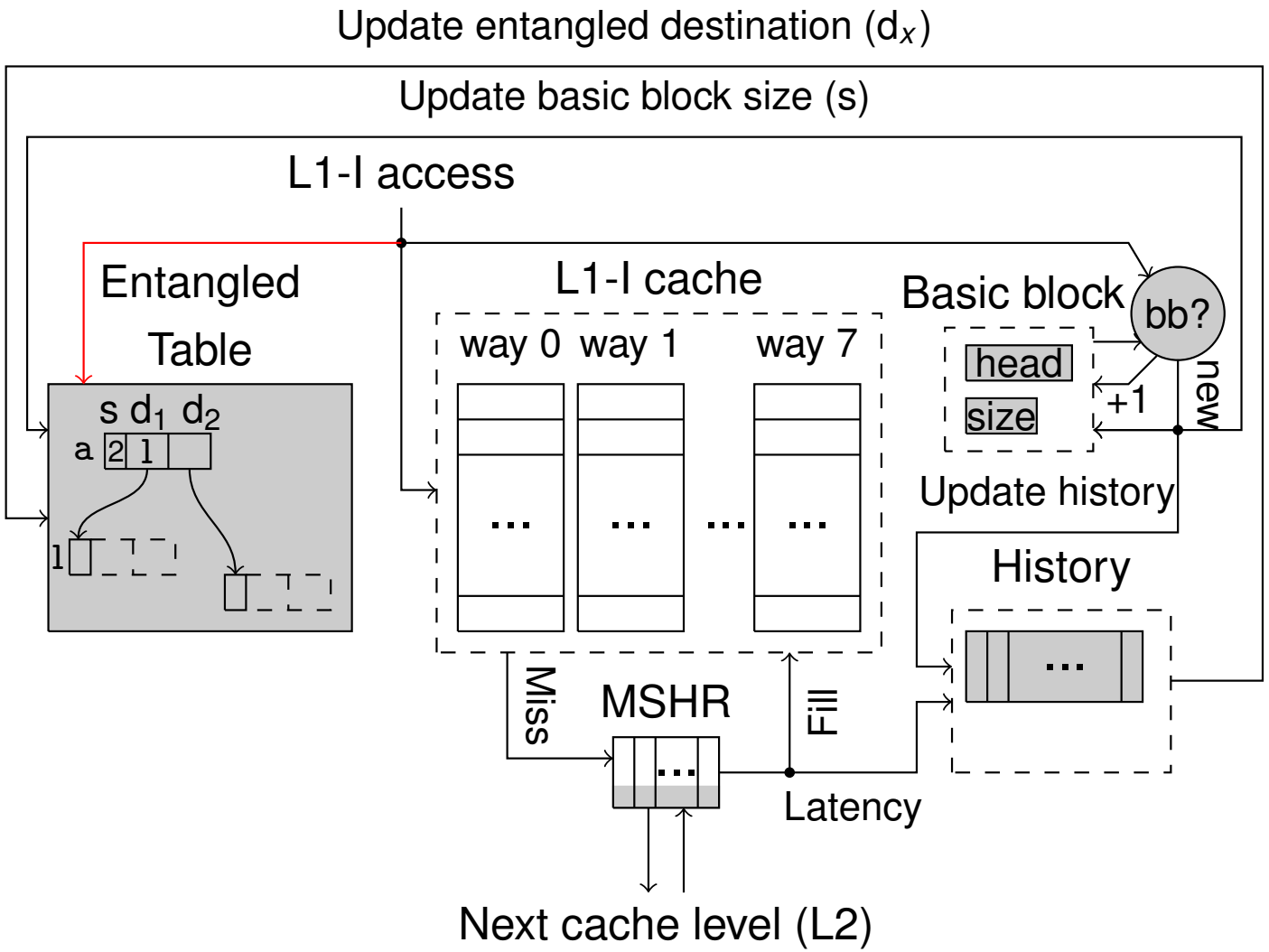
Design of the Entangling Prefetcher - Entangling cache lines



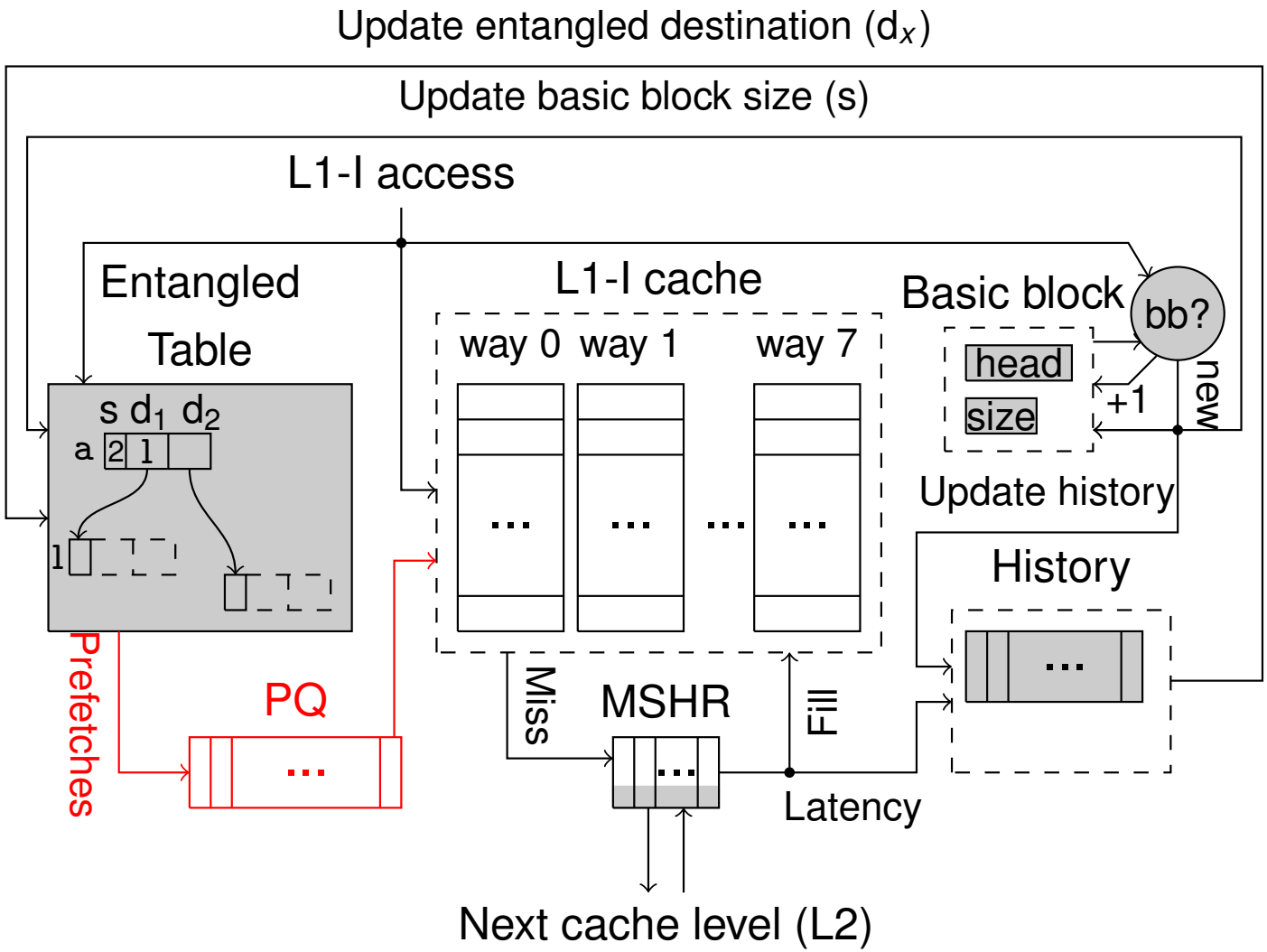
Design of the Entangling Prefetcher - Entangling cache lines



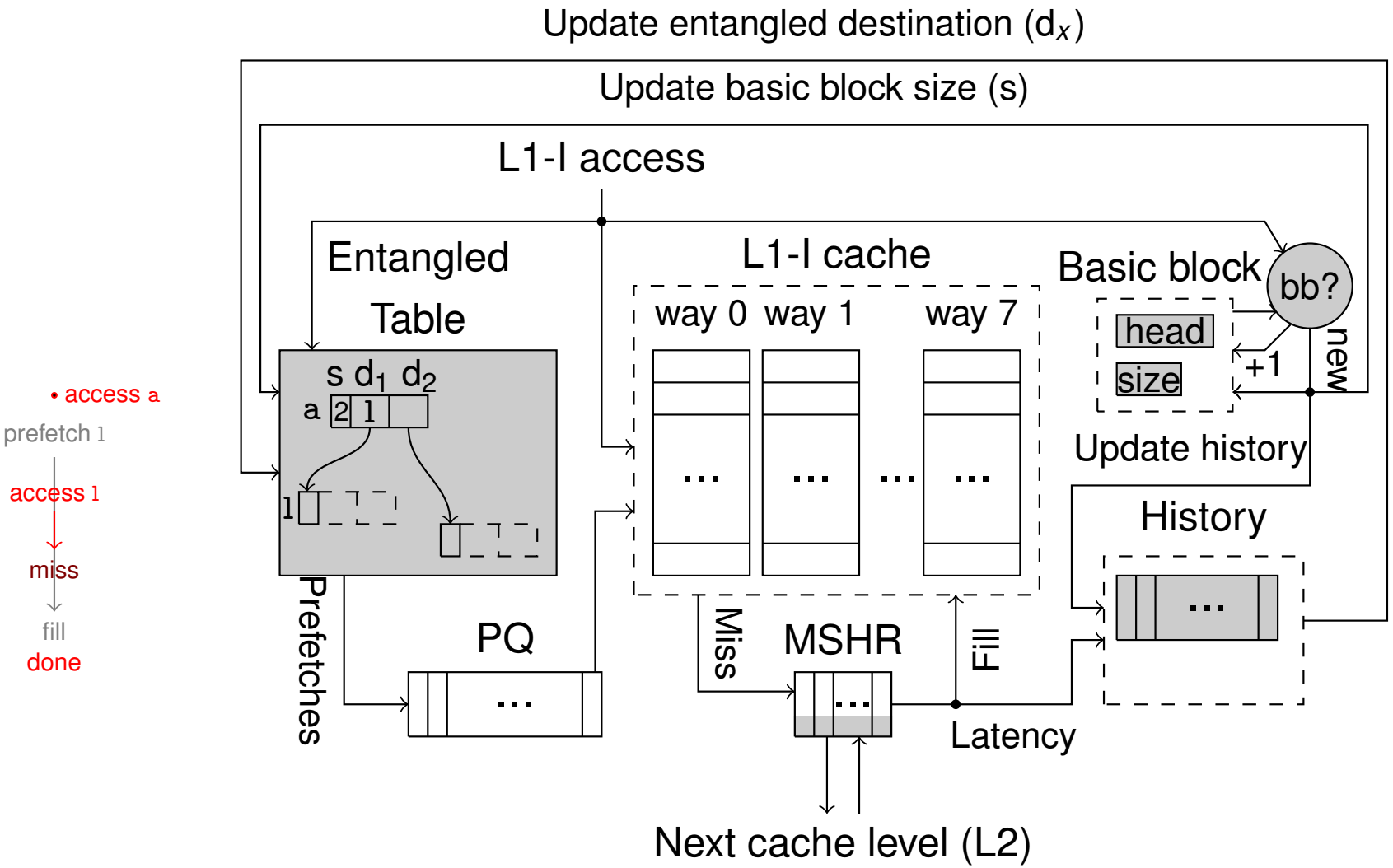
Design of the Entangling Prefetcher - Issuing prefetches



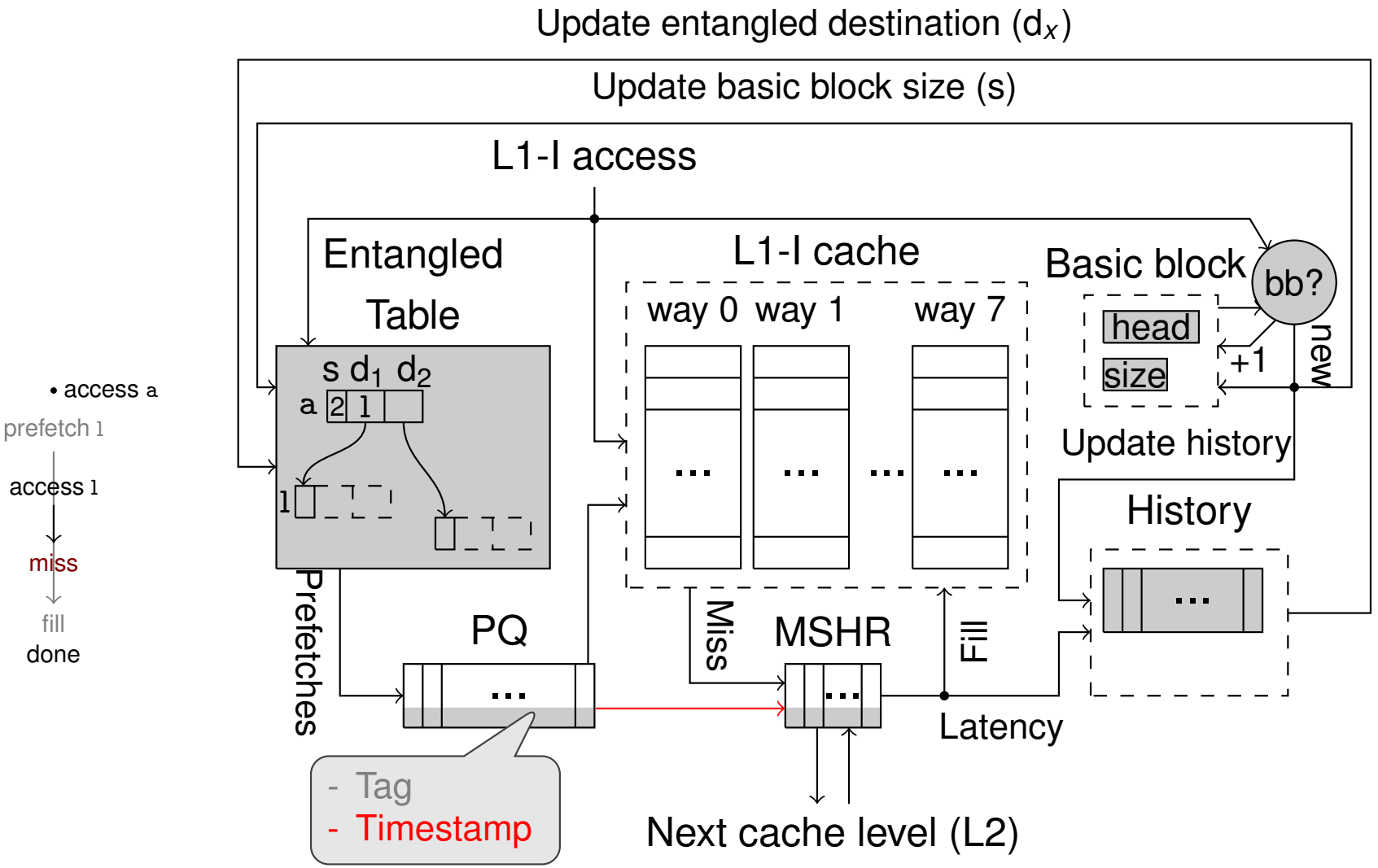
Design of the Entangling Prefetcher - Issuing prefetches



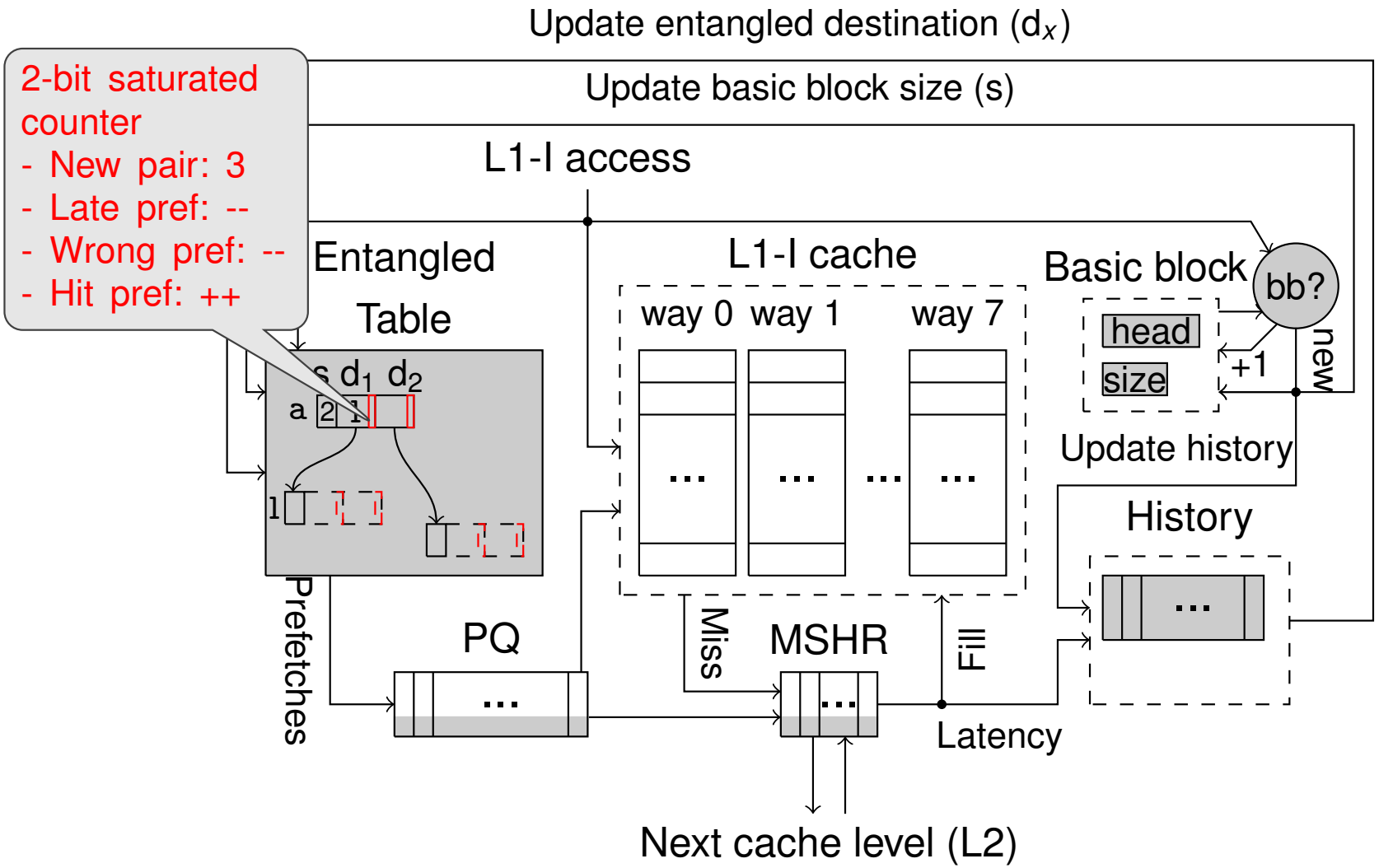
Design of the Entangling Prefetcher - Fixing Late Prefetches



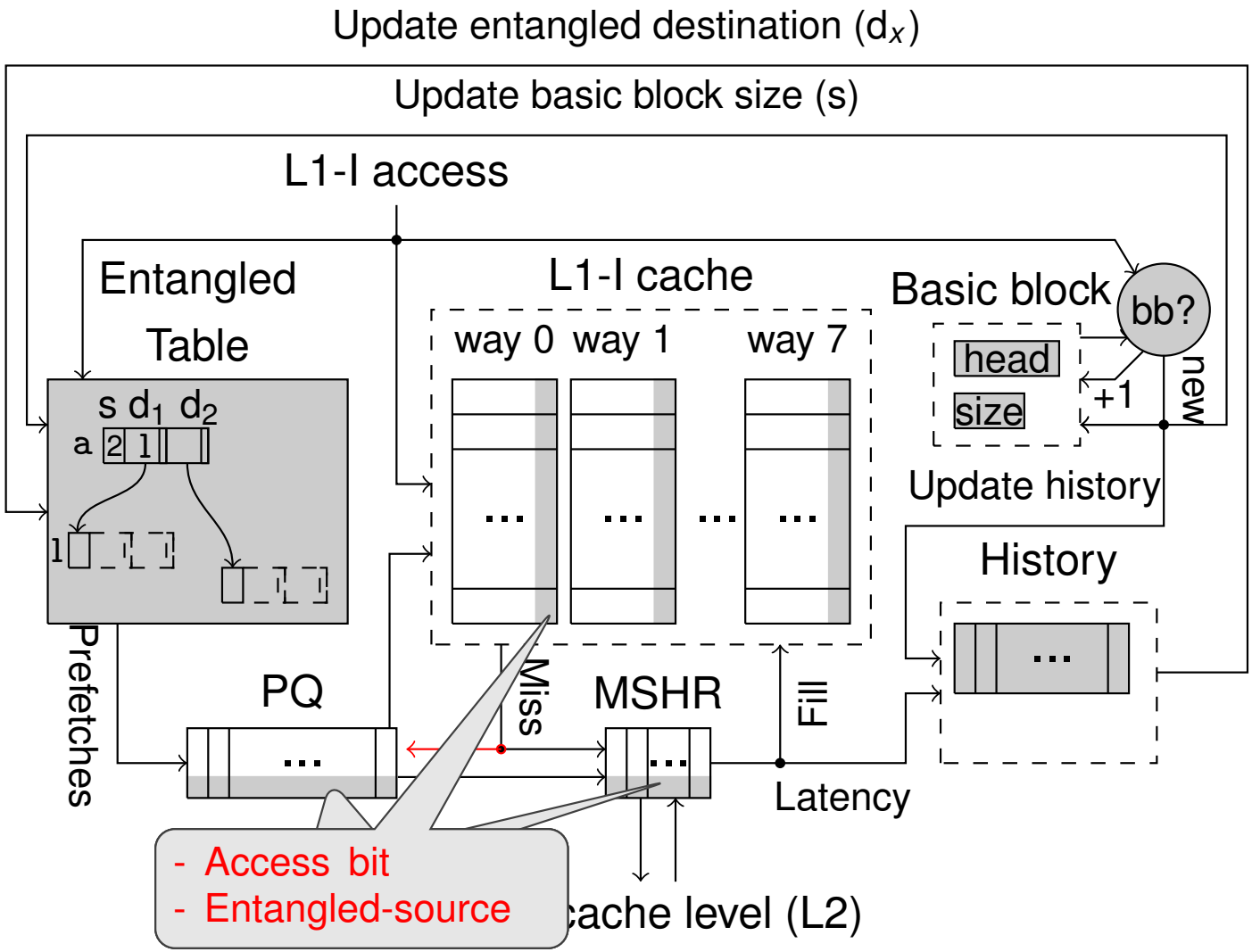
Design of the Entangling Prefetcher - Fixing late prefetches



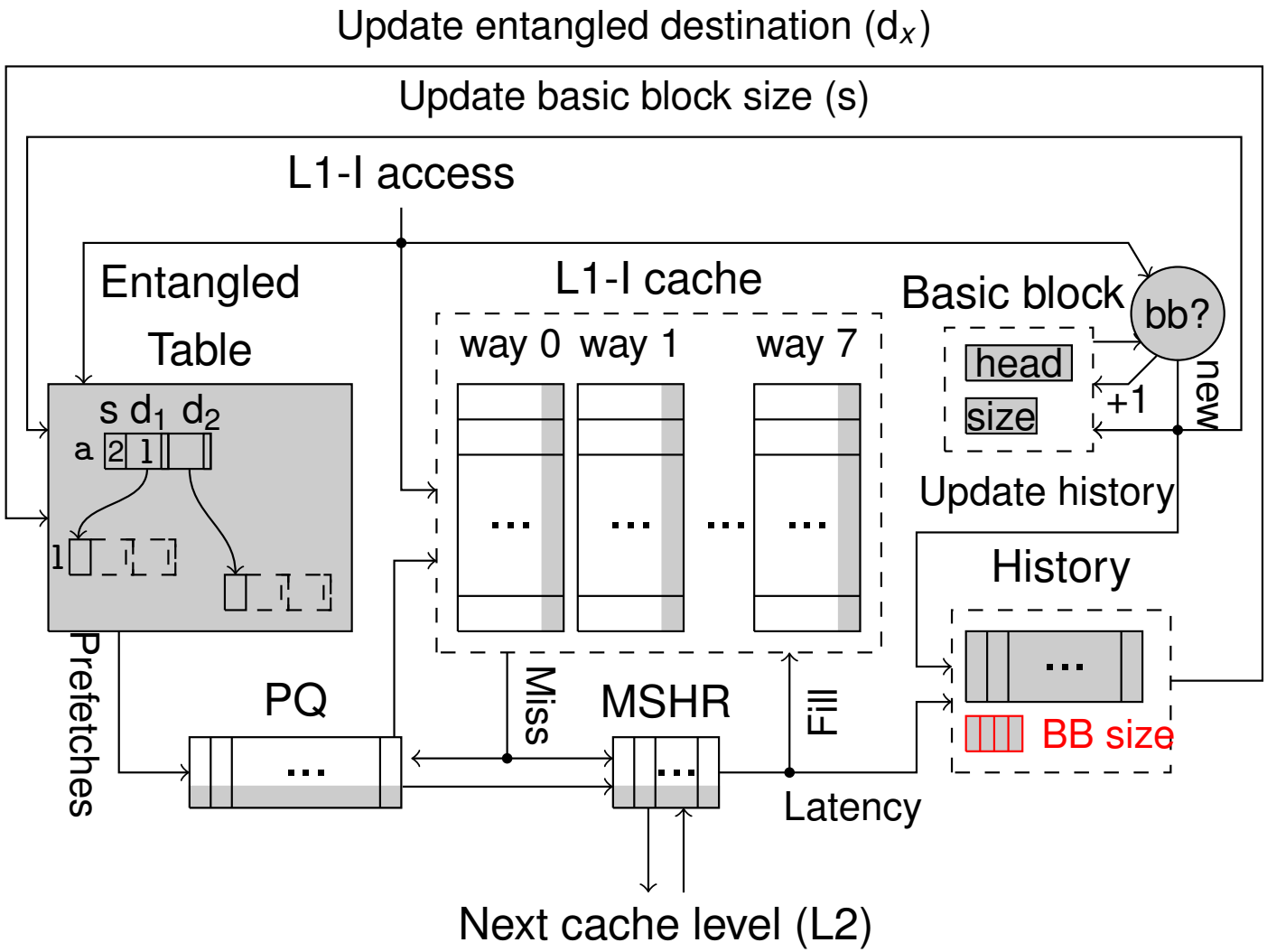
Design of the Entangling Prefetcher - Confidence for entangled pairs



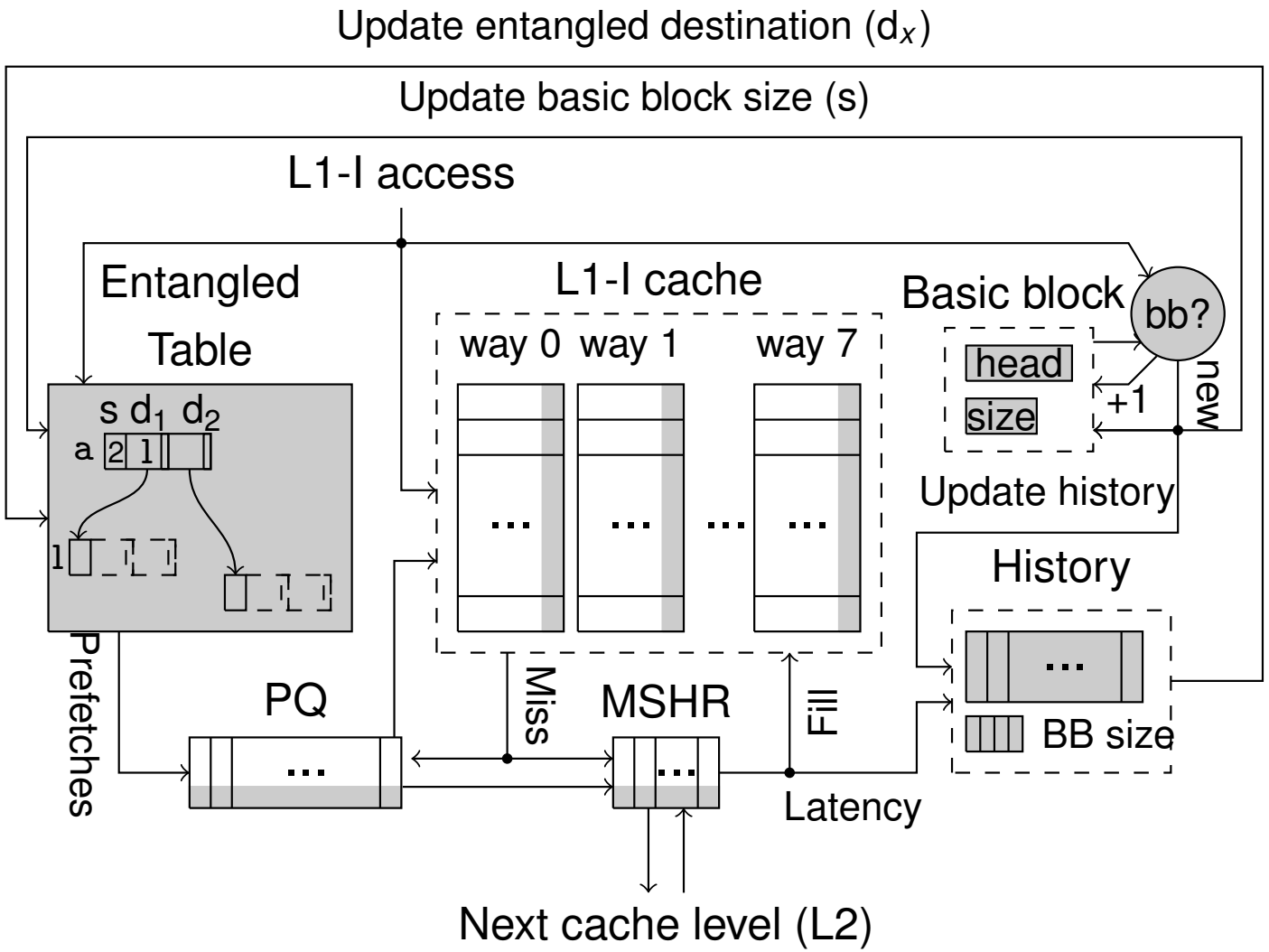
Design of the Entangling Prefetcher - Confidence for entangled pairs



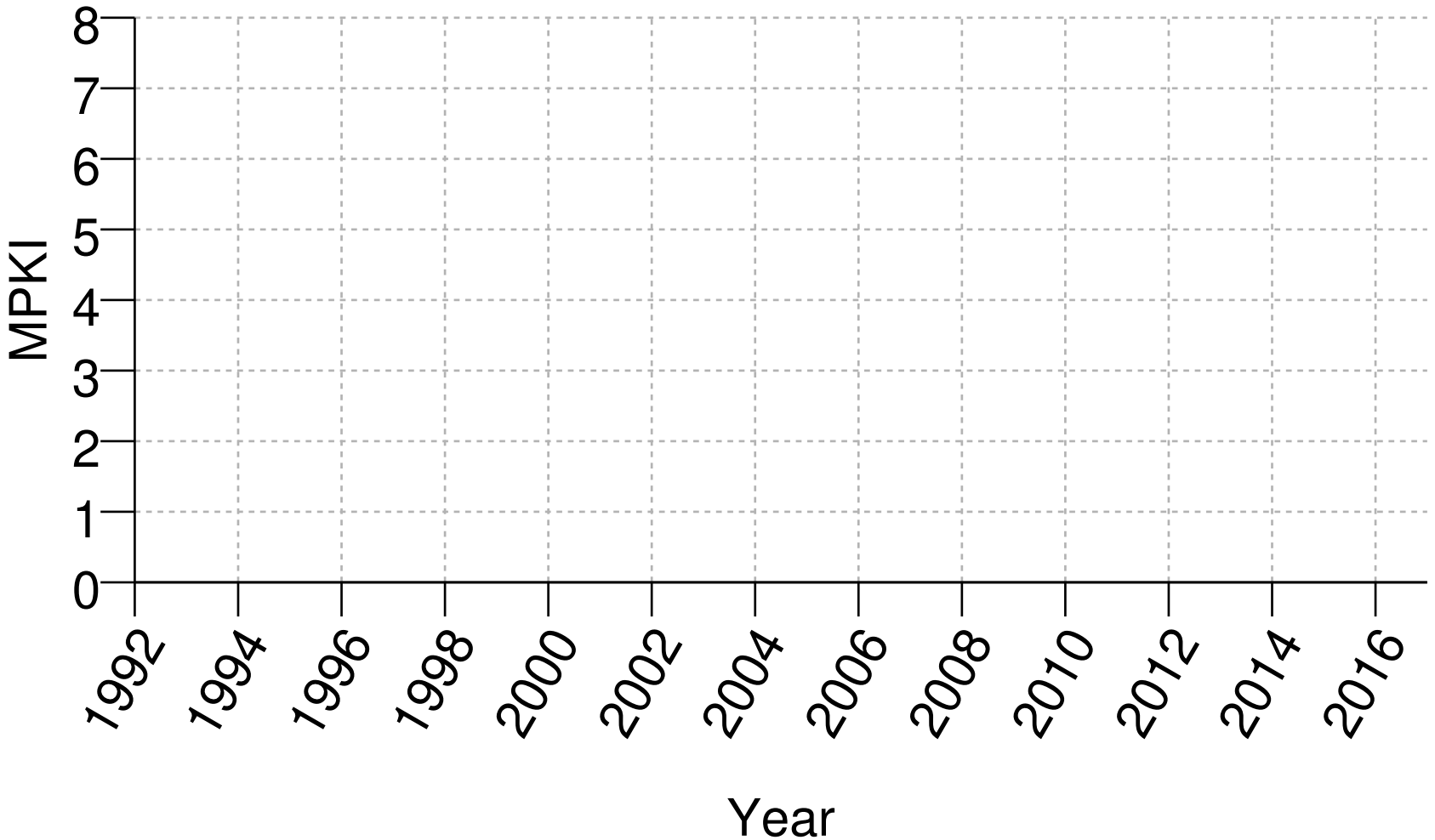
Design of the Entangling Prefetcher - Merging Basic Blocks



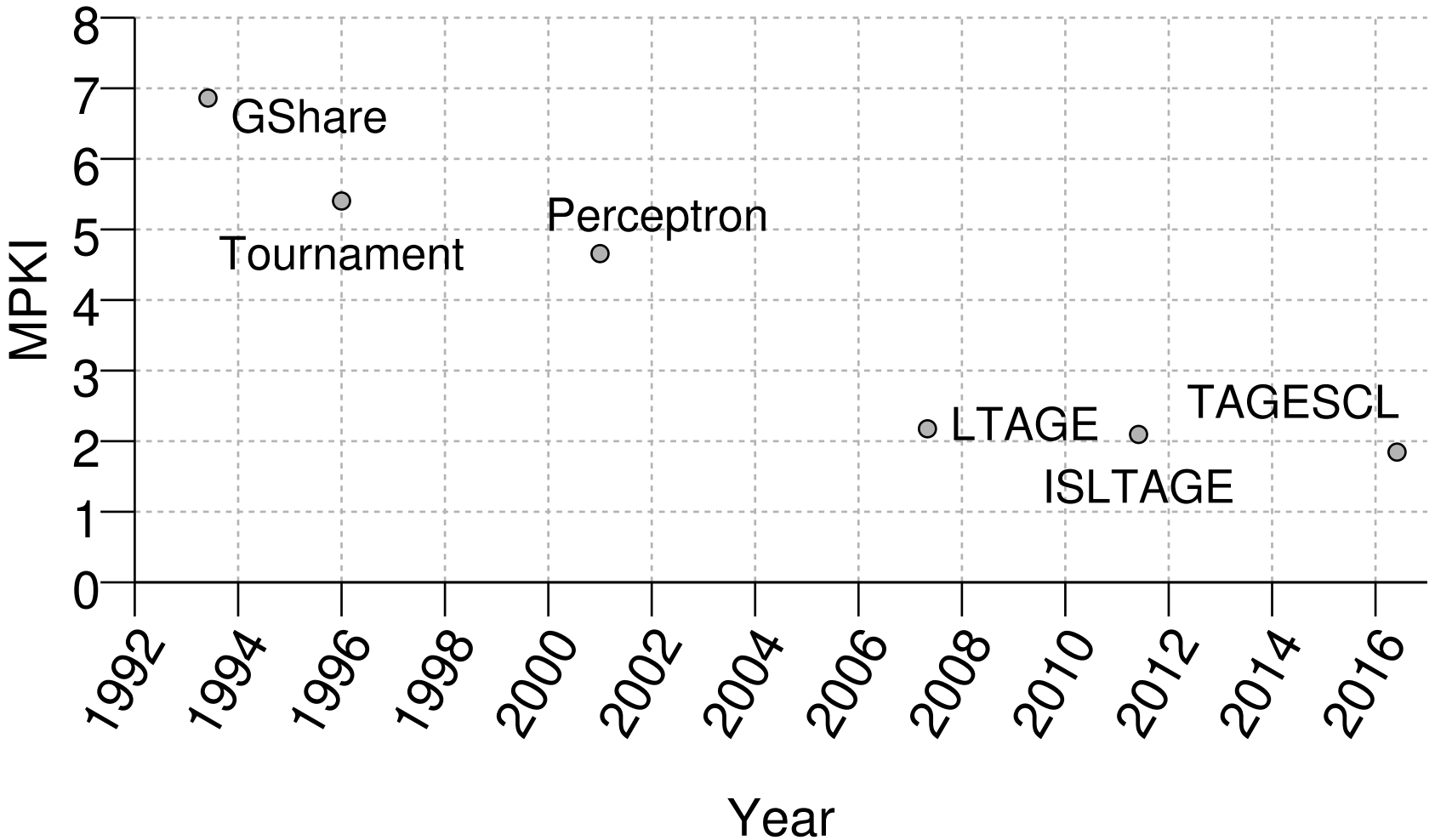
Design of the Entangling Prefetcher



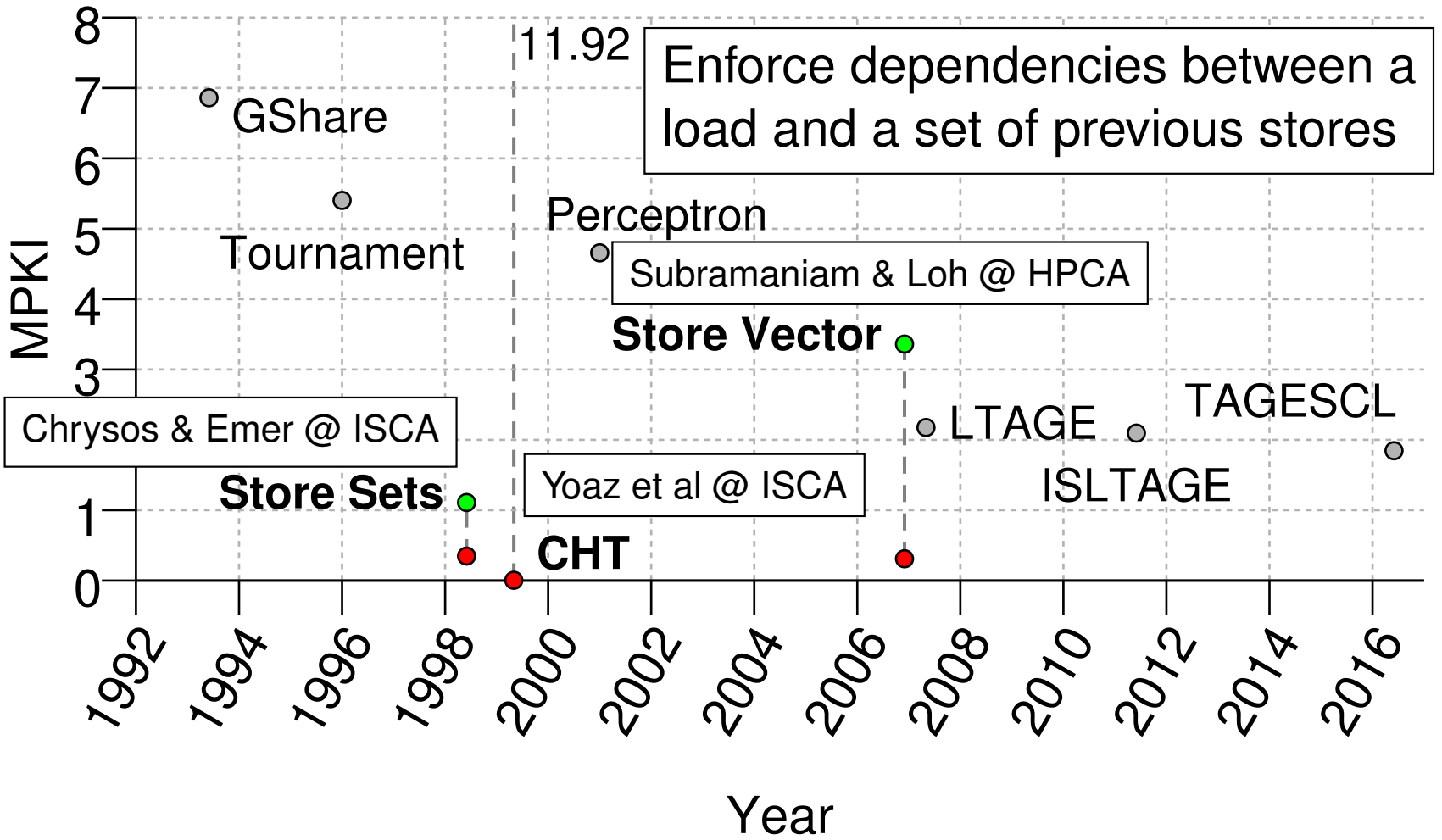
MDP, WAS NOT SOLVED IN THE 90'S?



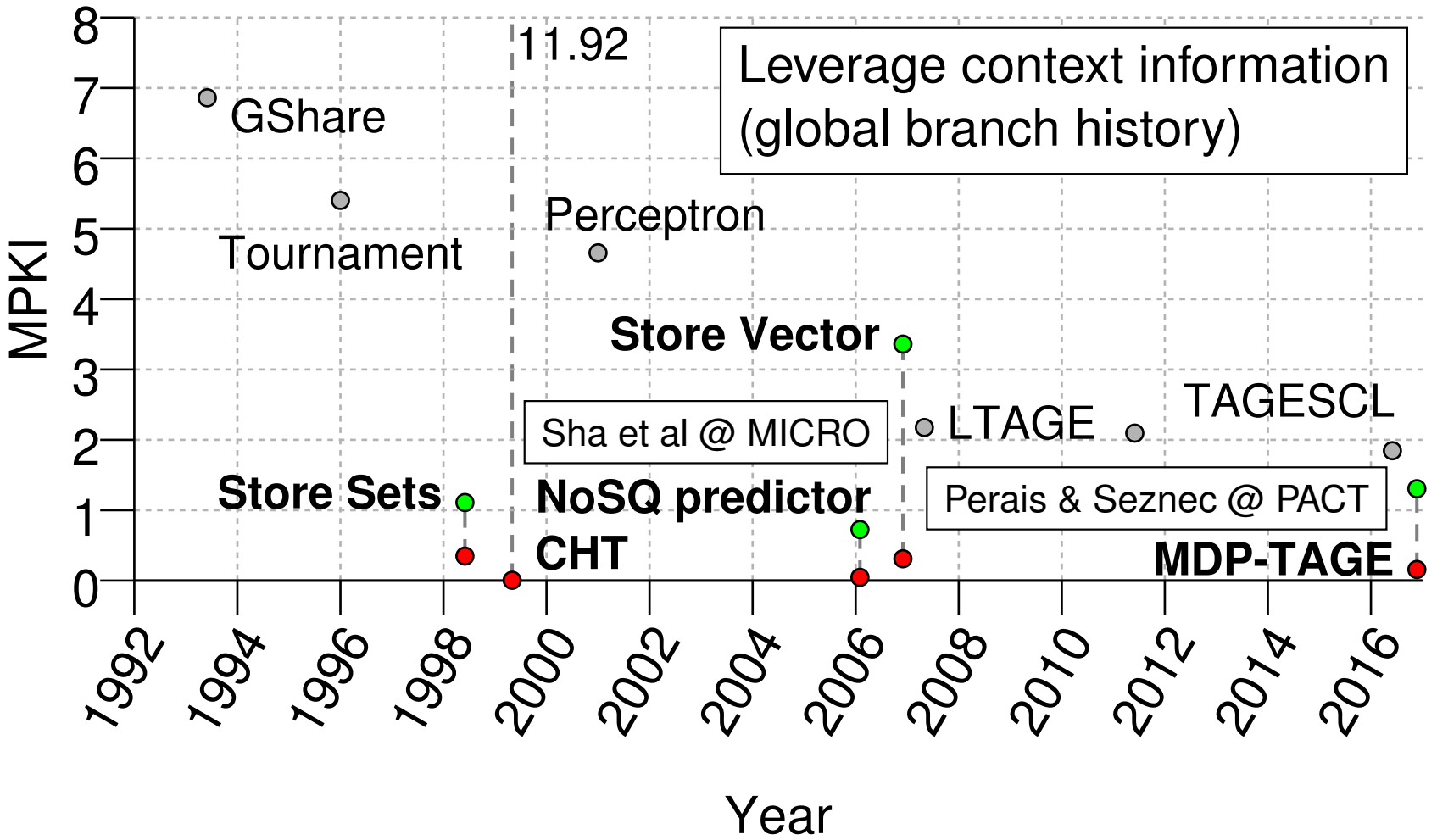
MDP, WAS NOT SOLVED IN THE 90'S?



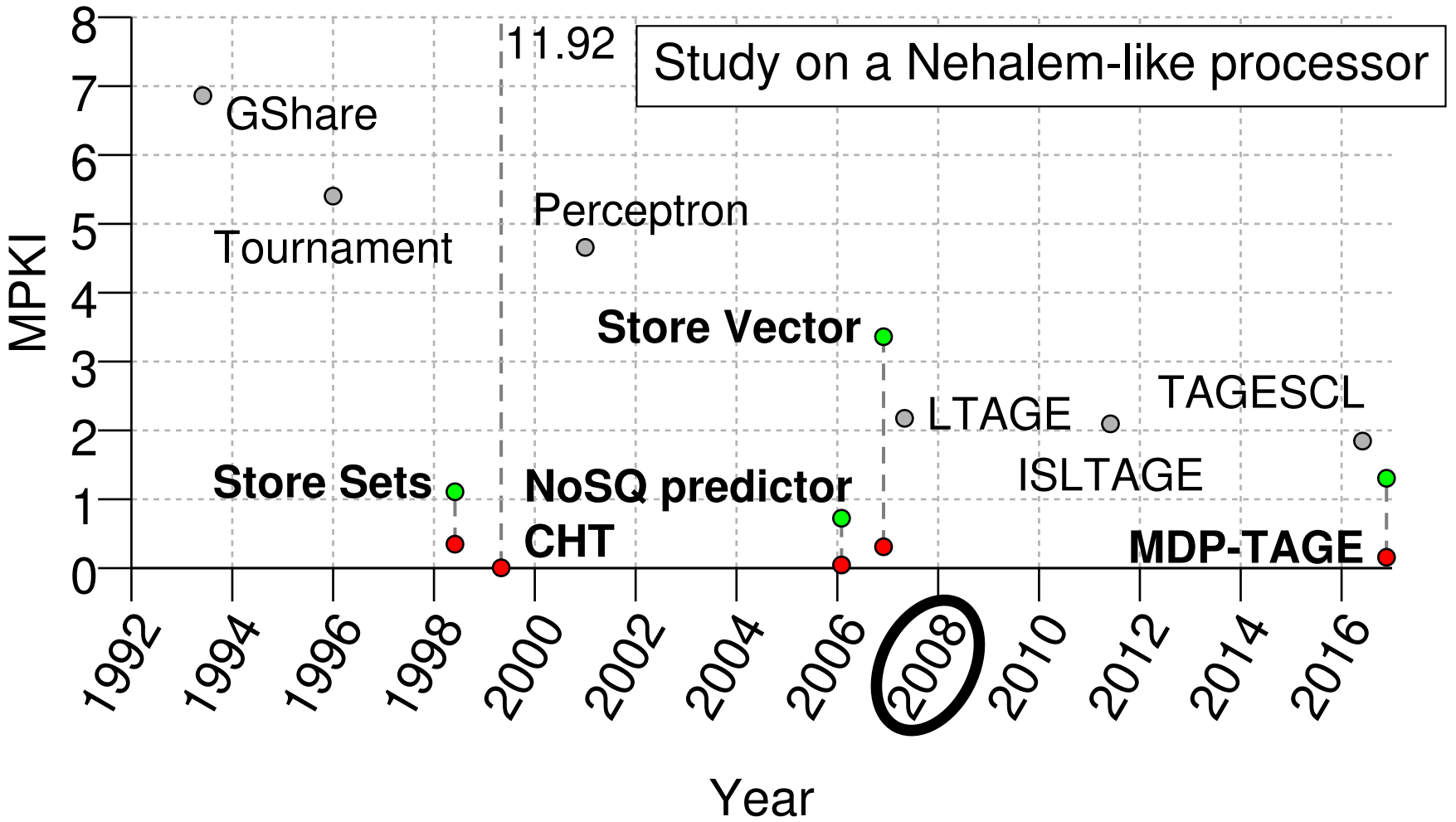
MDP, WAS NOT SOLVED IN THE 90'S?



MDP, WAS NOT SOLVED IN THE 90'S?



MDP, WAS NOT SOLVED IN THE 90'S?



PHAST: IDENTIFYING A SINGLE STORE PRECISELY

- Loads are quite frequently dependent on a single store

PHAST: IDENTIFYING A SINGLE STORE PRECISELY

- Loads are quite frequently dependent on a single store

1. Exceptions (0.04% of total loads):
   - E.g., narrow stores followed by a wide load

   $\Rightarrow$ 70% of those situation stores execute in order, so identifying the younger store is enough

PHAST: IDENTIFYING A SINGLE STORE PRECISELY

- Loads are quite frequently dependent on a single store

1. Exceptions (0.04% of total loads):
   - E.g., narrow stores followed by a wide load

   $\Rightarrow$ 70% of those situation stores execute in order, so identifying the younger store is enough
2. Several stores targeting the same address as the load
   - Only the one closer to the load matters

PHAST: IDENTIFYING A SINGLE STORE PRECISELY

- Loads are quite frequently dependent on a single store

1. Exceptions (0.04% of total loads):
   - E.g., narrow stores followed by a wide load

   $\Rightarrow$ 70% of those situation stores execute in order, so identifying the younger store is enough
2. Several stores targeting the same address as the load
   - Only the one closer to the load matters

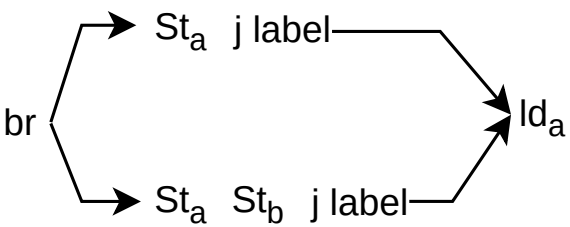
SOLUTION Track precisely a single dependent store distance

PHAST: LEARNING THE PROPER CONTEXT

- PHAST defines the *proper* length as $N + 1$, where N is the number of divergent branches between the store and the load

PHAST: LEARNING THE PROPER CONTEXT

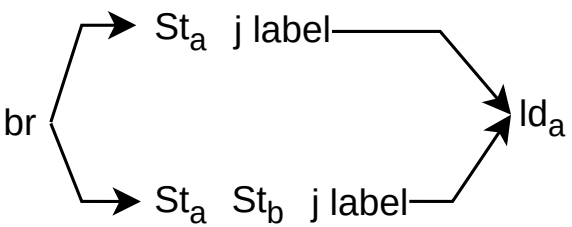
- PHAST defines the *proper* length as $N + 1$, where N is the number of divergent branches between the store and the load
- Why $+1$?



Conflict	Store Distance	History
Up	1	label
Down	2	label

PHAST: LEARNING THE PROPER CONTEXT

- PHAST defines the *proper* length as $N + 1$, where N is the number of divergent branches between the store and the load
- Why $+1$?

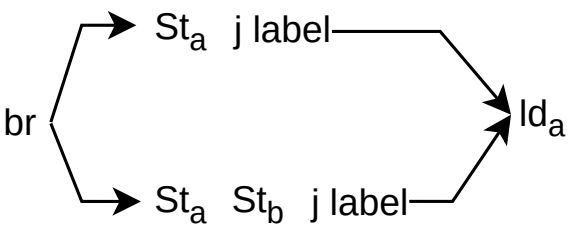


Conflict	Store Distance	History
Up	1	label
Down	2	label

Same history, different distances!

PHAST: LEARNING THE PROPER CONTEXT

- PHAST defines the *proper* length as $N + 1$, where N is the number of divergent branches between the store and the load
- Why $+1$?



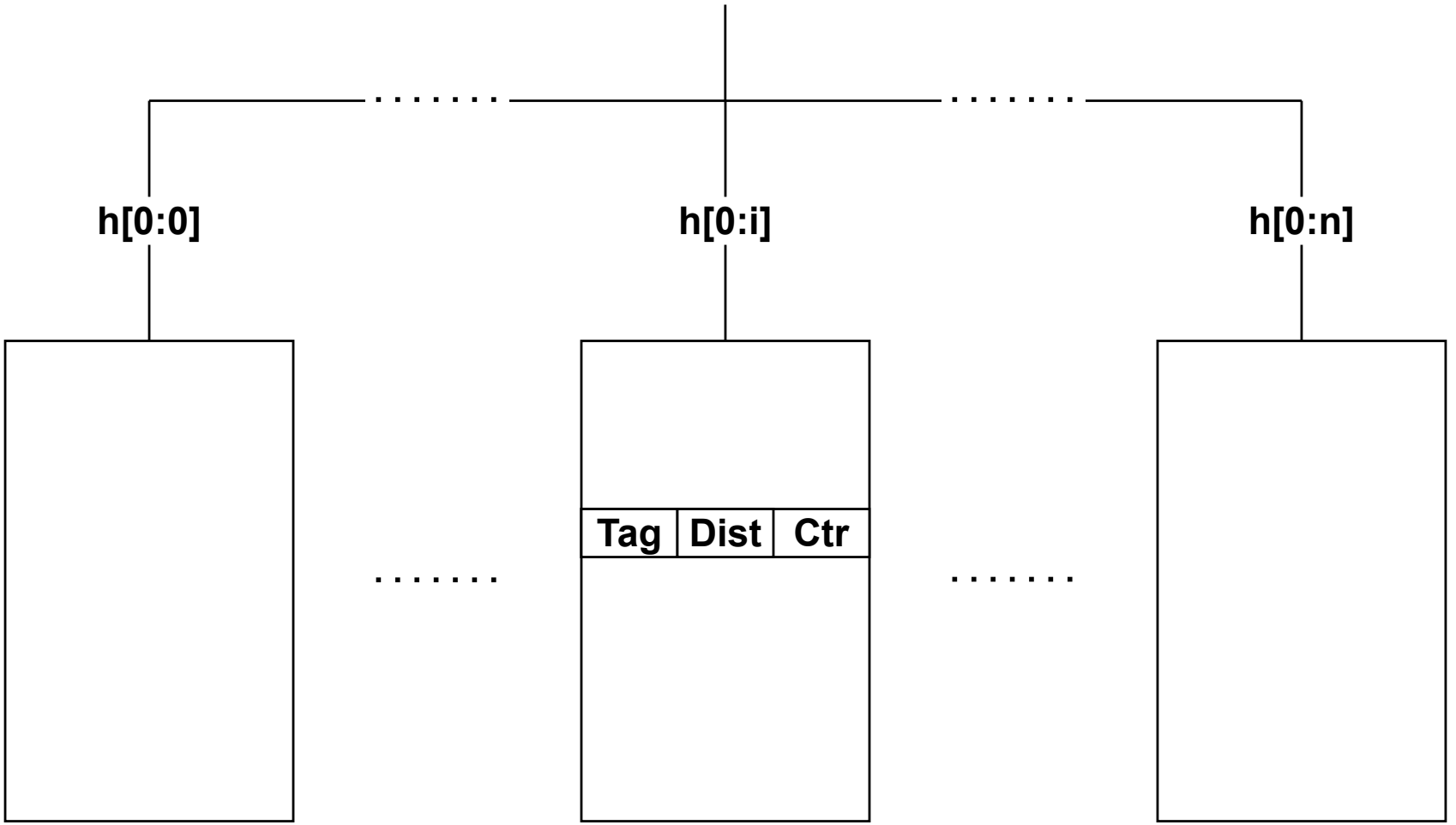
Conflict	Store Distance	History
Up	1	label
Down	2	label

Same history, different distances!

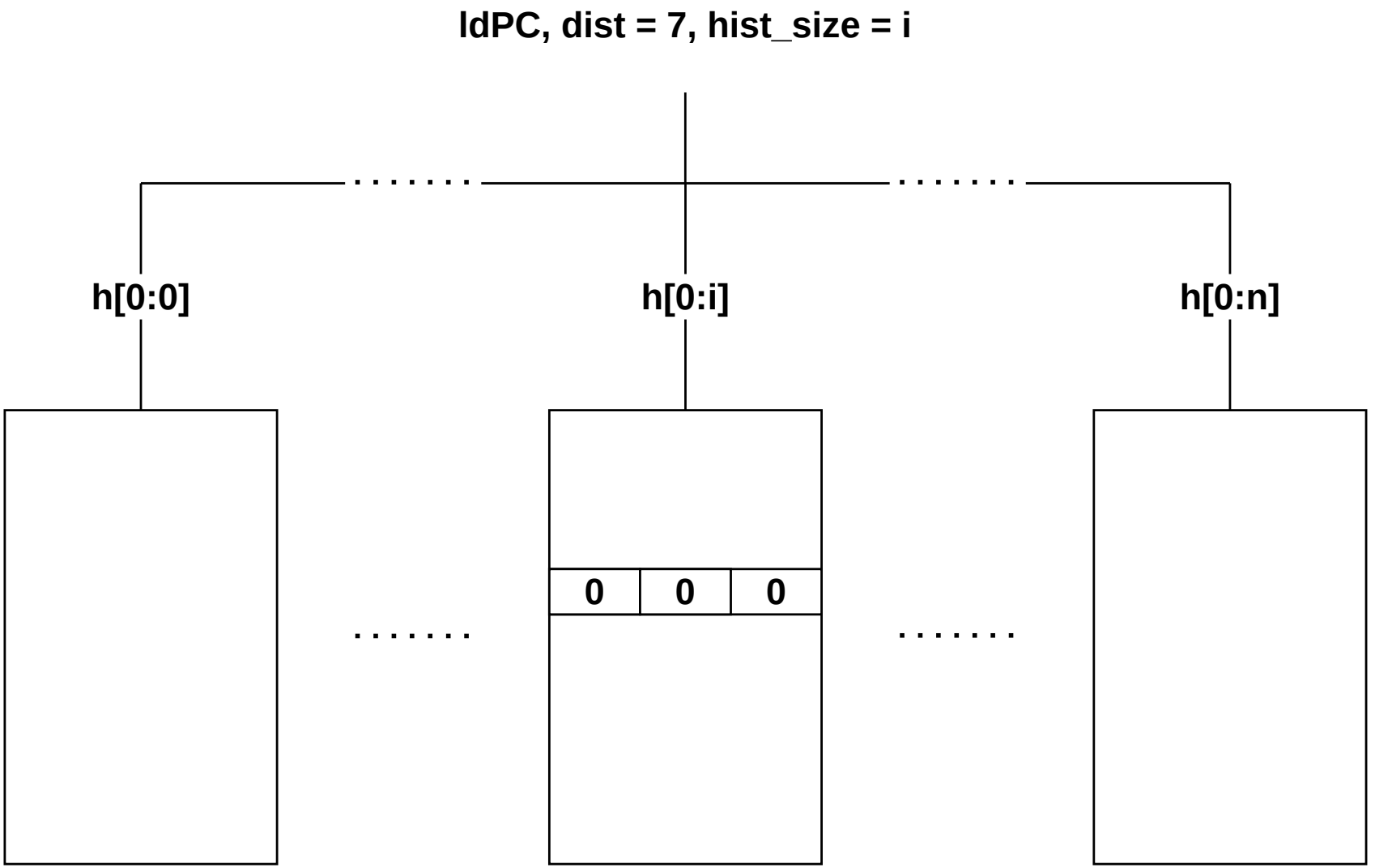
SOLUTION Use target of the branch previous to the store to differentiate the paths

ld$_a$	br	br	br	st$_a$	br	br

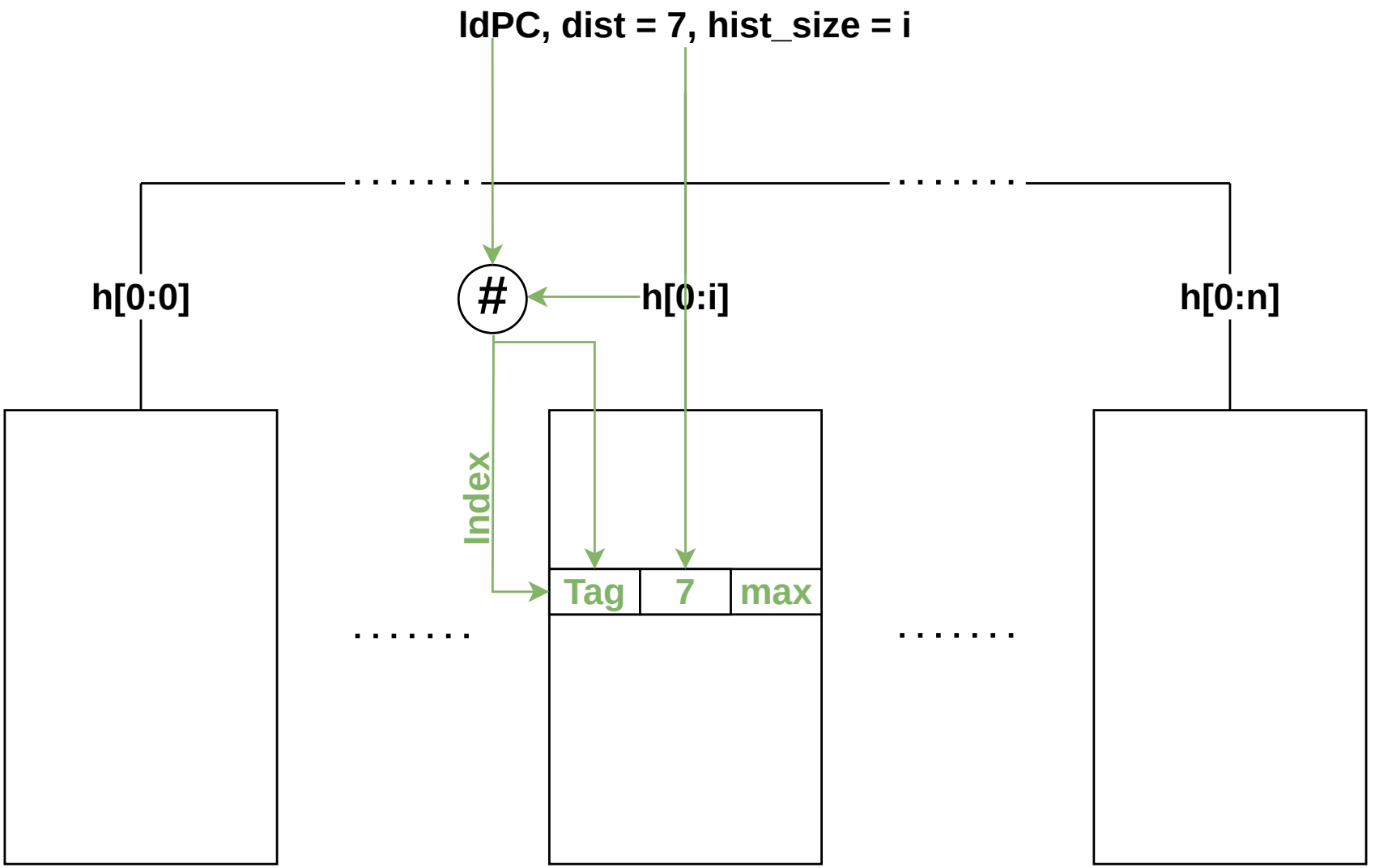
PHAST: STRUCTURE



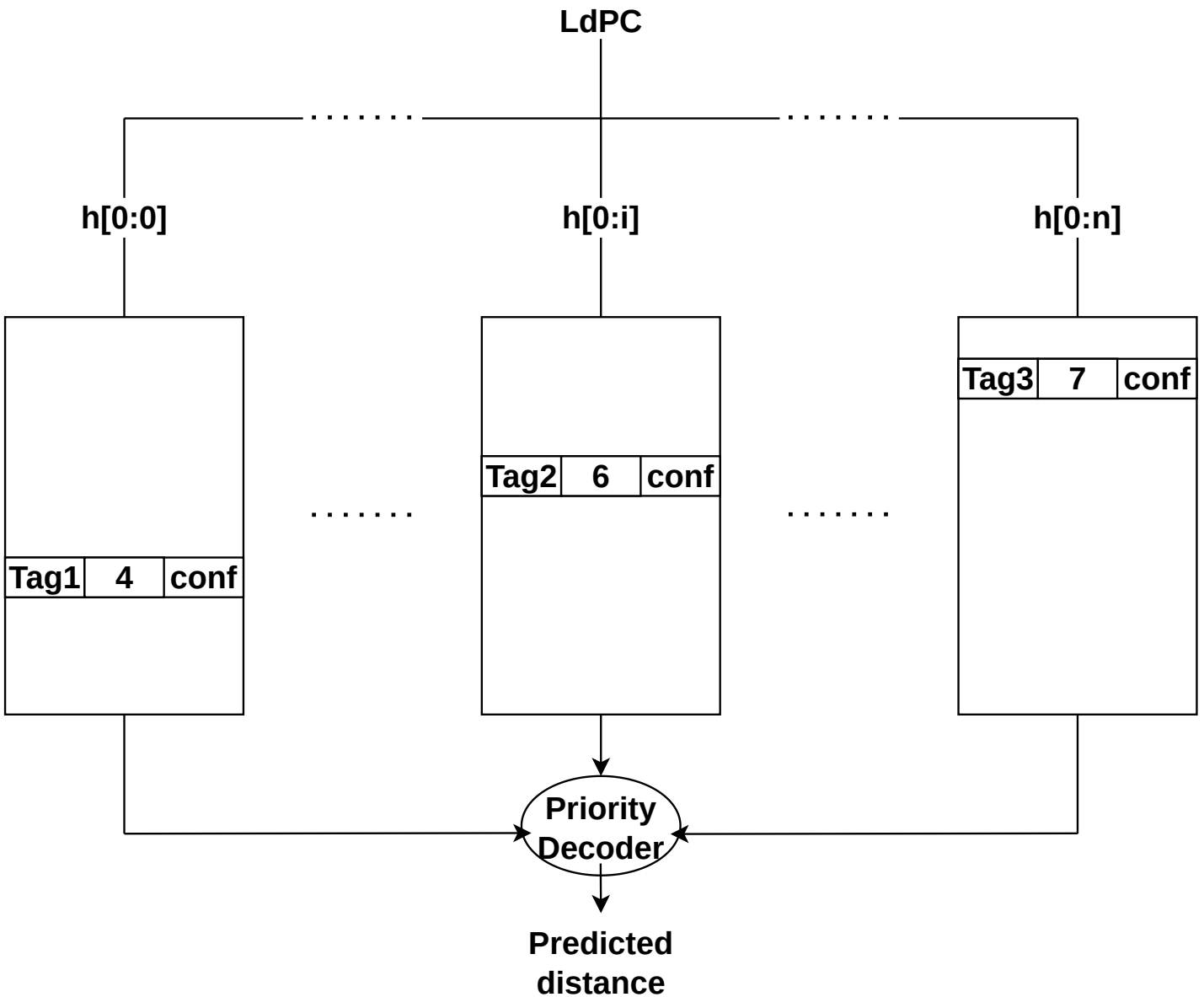
PHAST: UPDATE



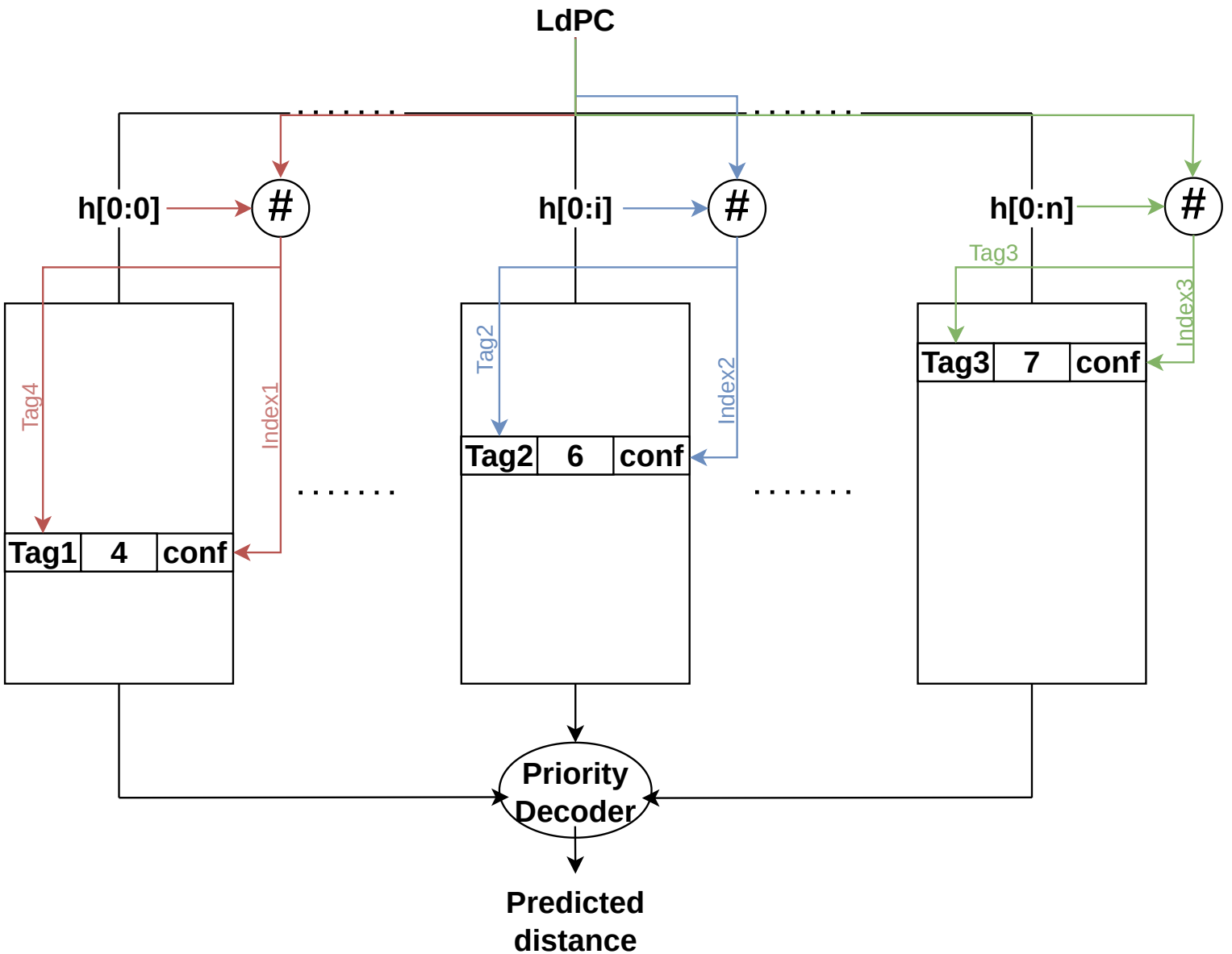
PHAST: UPDATE



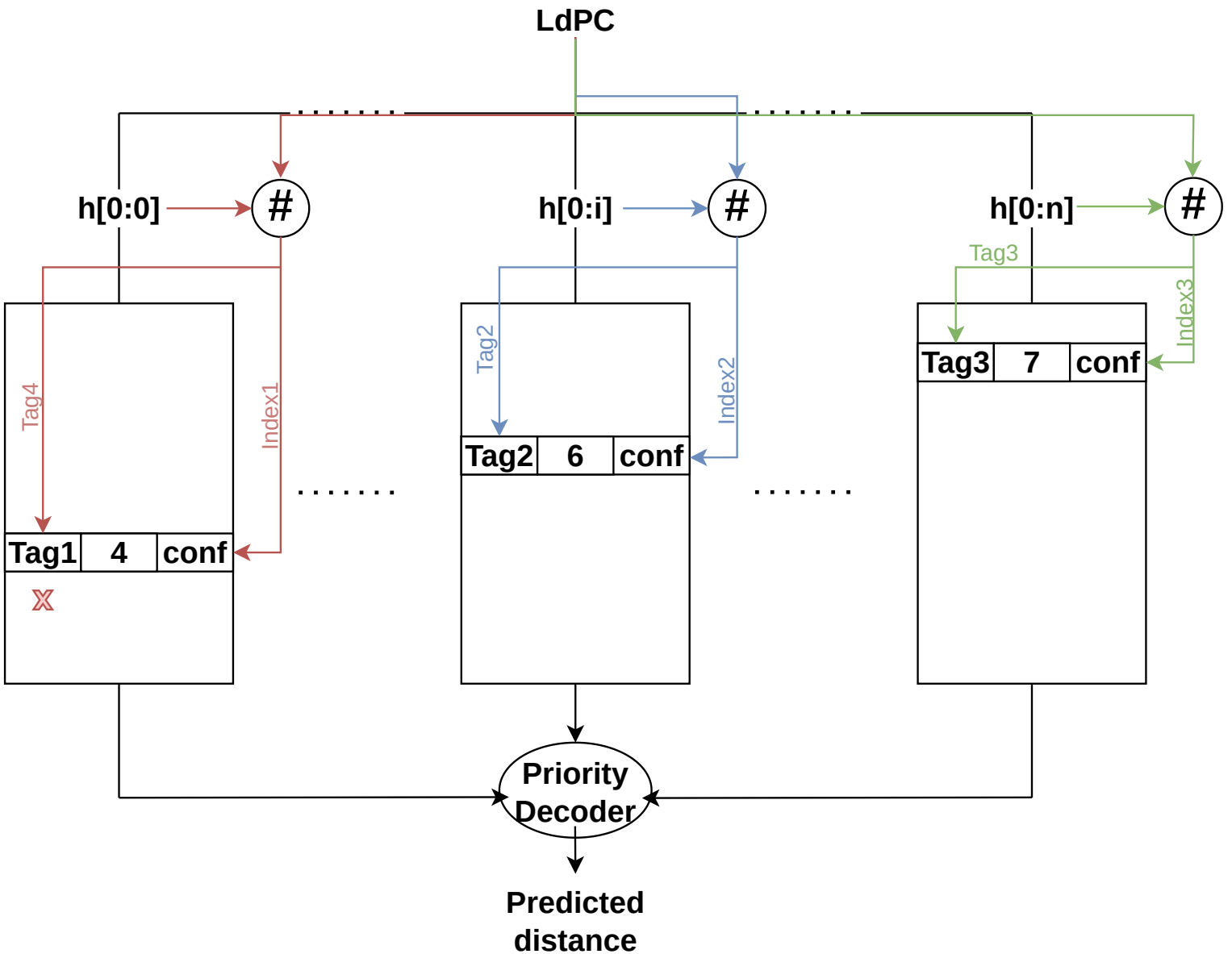
PHAST: PREDICTION



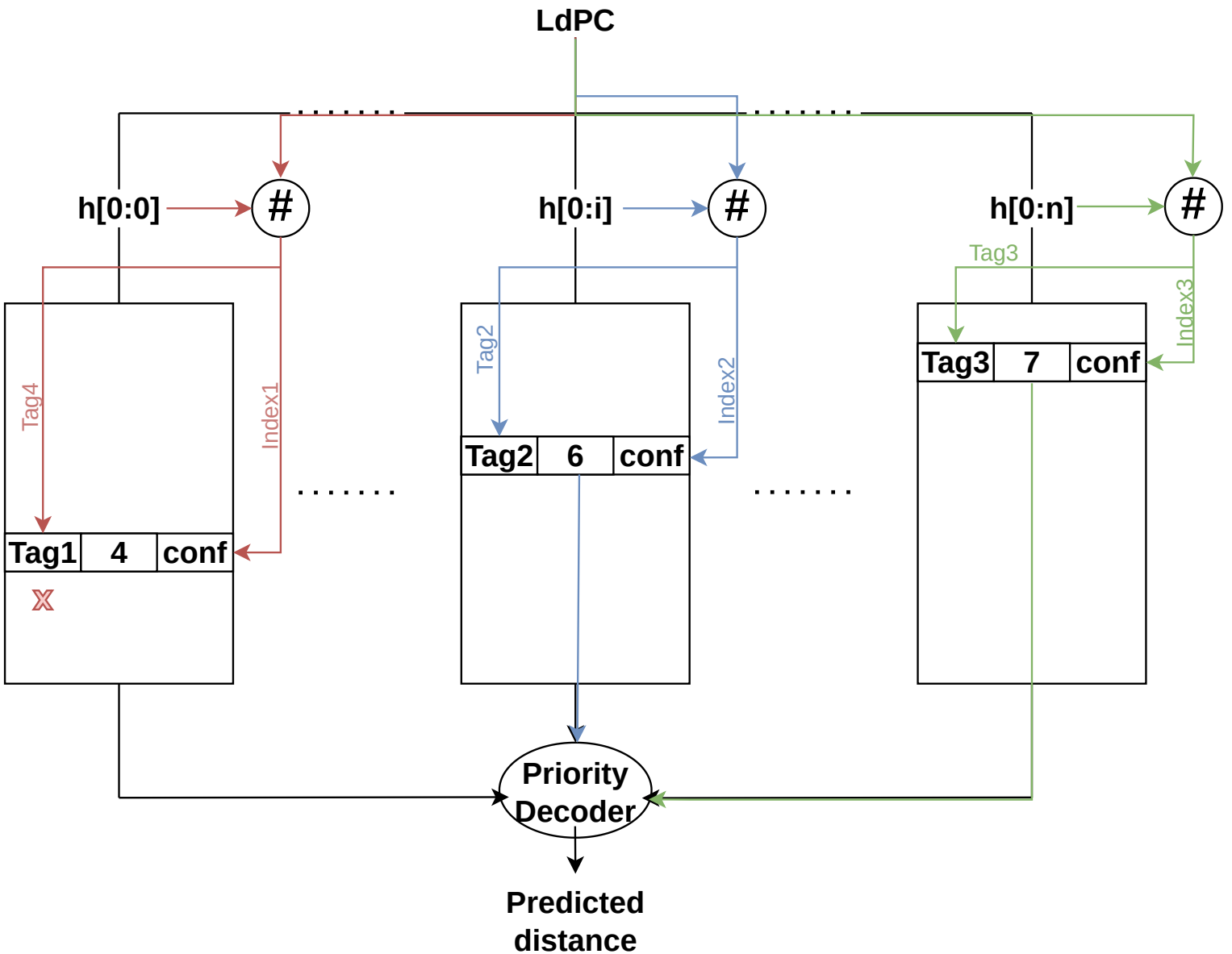
PHAST: PREDICTION



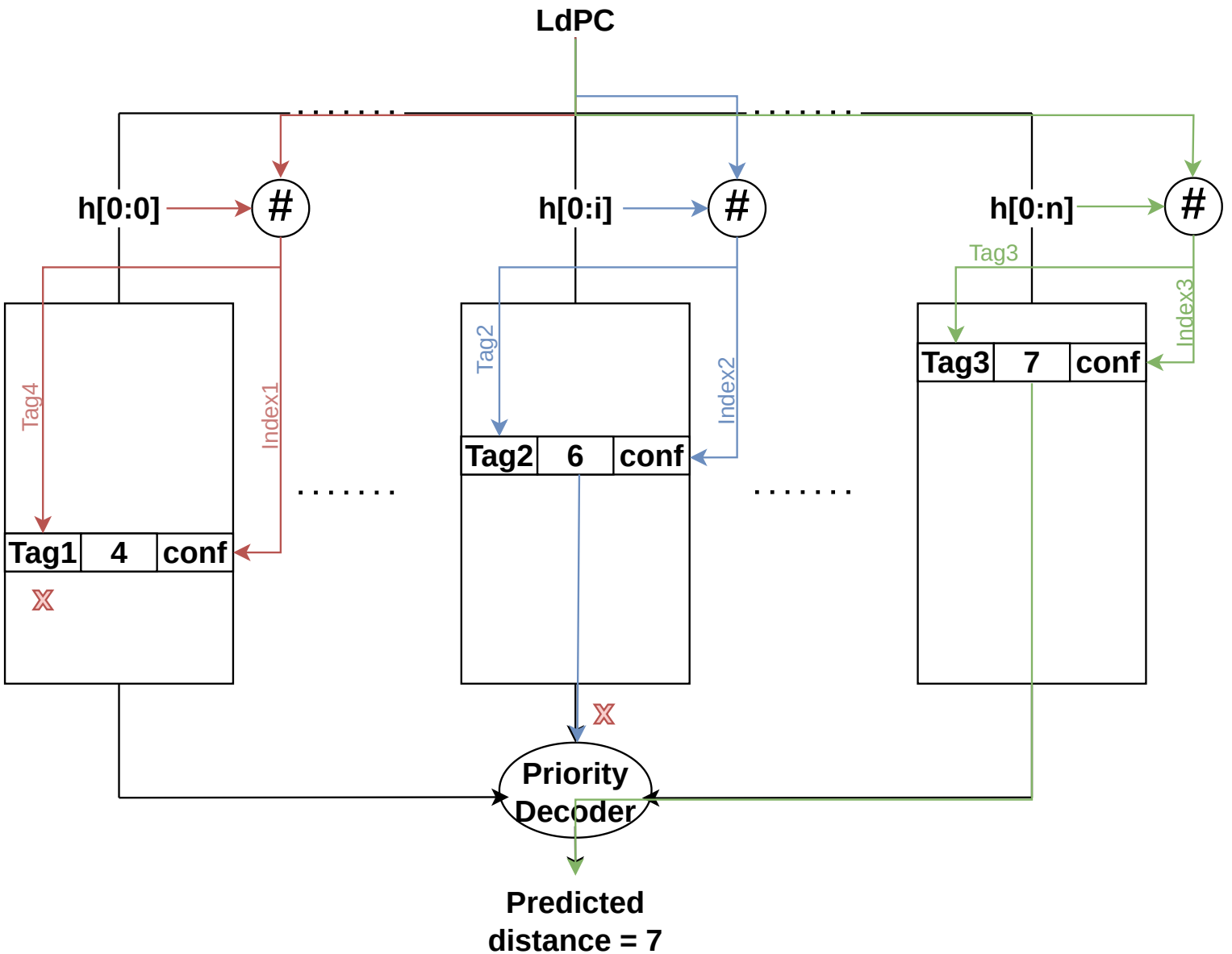
PHAST: PREDICTION



PHAST: PREDICTION



PHAST: PREDICTION



PHAST: IMPLEMENTATION DETAILS

- Confidence counter:
  - 4-bit saturated counter
  - On commit, the prediction is checked
    - Correct $\rightarrow$ Saturate confidence counter
    - False dependence $\rightarrow$ Decrease it by 1

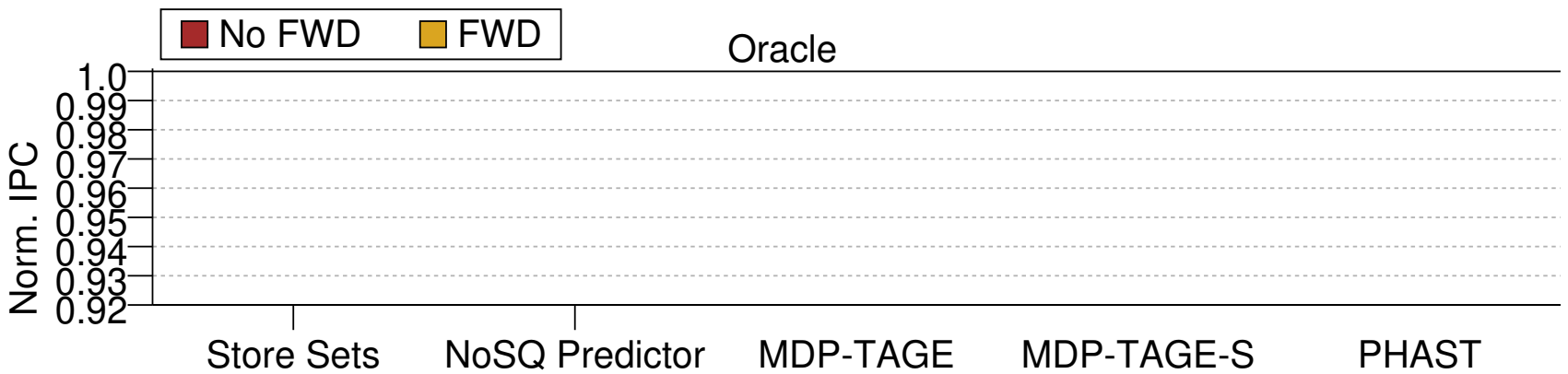
PHAST: IMPLEMENTATION DETAILS

- Confidence counter:
  - 4-bit saturated counter
  - On commit, the prediction is checked
    - Correct $\rightarrow$ Saturate confidence counter
    - False dependence $\rightarrow$ Decrease it by 1

- Number and configuration of tables:
  - 8 tables with geometric-like sequence of history lengths $\Rightarrow$ {0, 2, 4, 6, 8, 12, 16, 32}
  - Histories not covered are truncated
  - Each table is 4-way associative

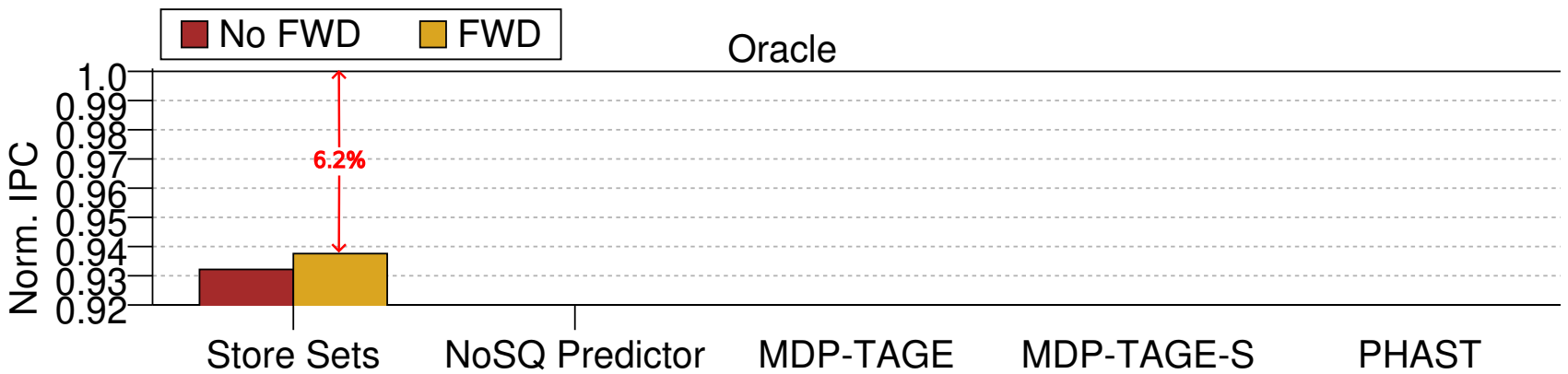
RESULTS: FORWARDING FILTERING

IDENTIFYING PRECISELY THE CONFLICTING STORE



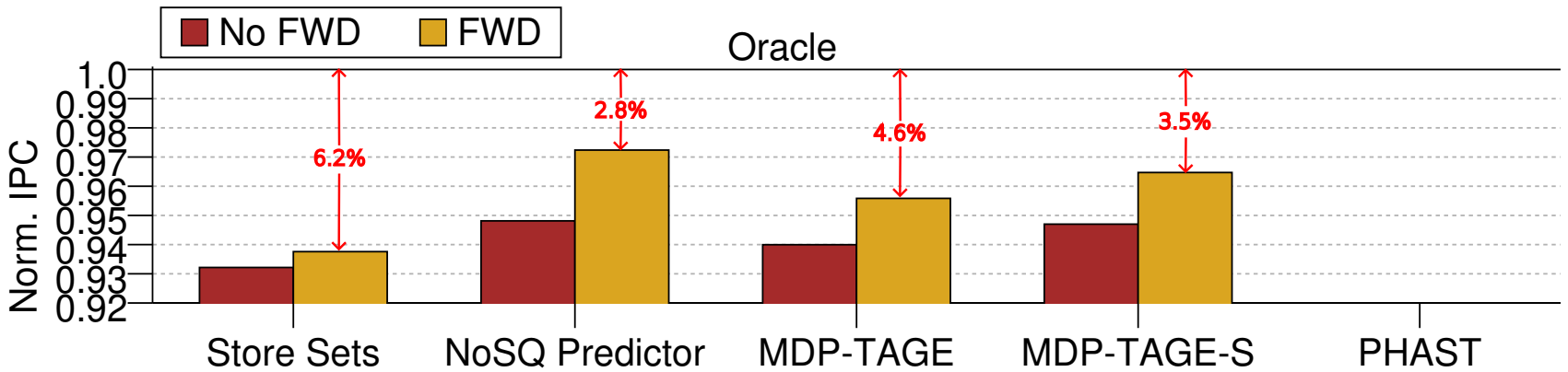
RESULTS: FORWARDING FILTERING

IDENTIFYING PRECISELY THE CONFLICTING STORE



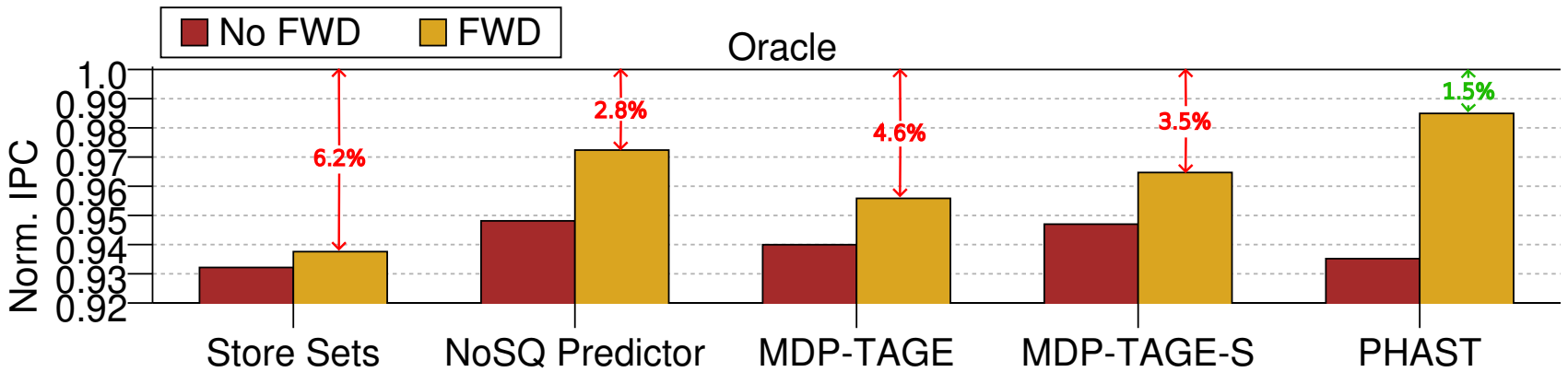
RESULTS: FORWARDING FILTERING

IDENTIFYING PRECISELY THE CONFLICTING STORE

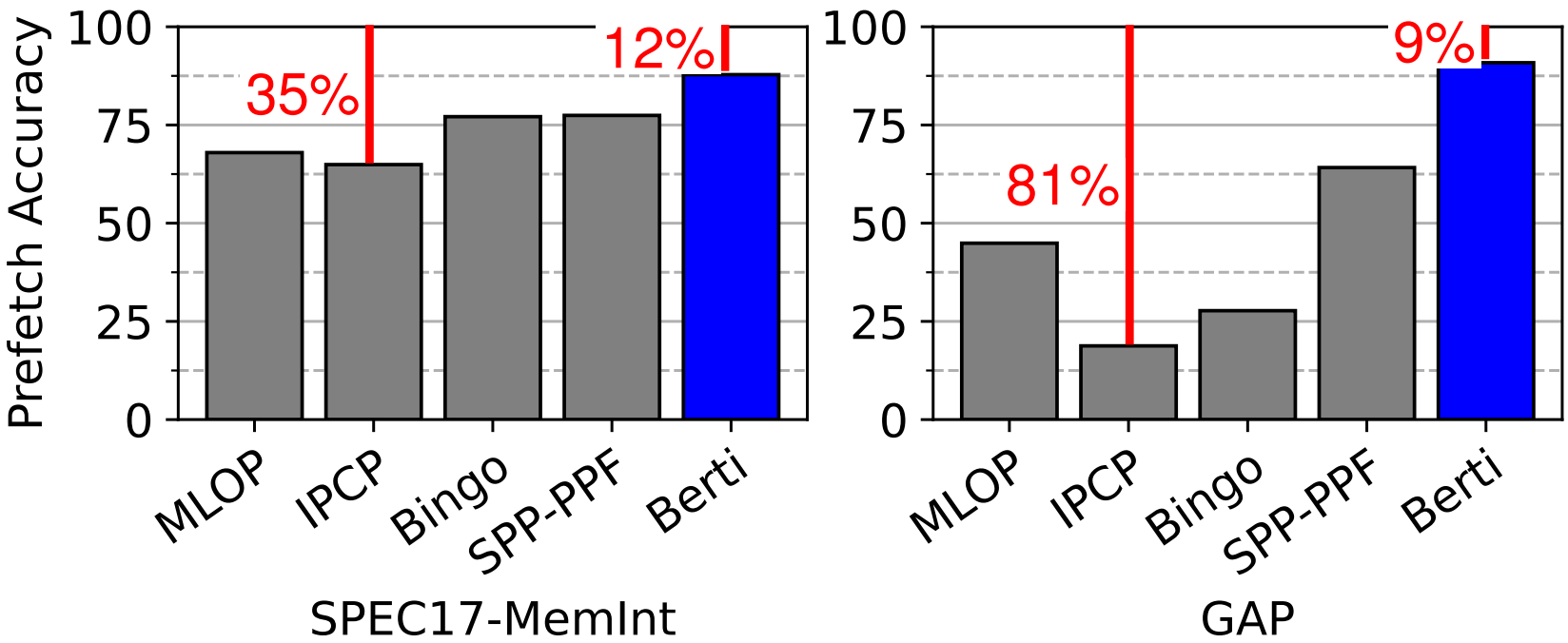


RESULTS: FORWARDING FILTERING

IDENTIFYING PRECISELY THE CONFLICTING STORE



BERTI: ACCURATE AND TIMELY LOCAL DELTA L1D PREFETCHER



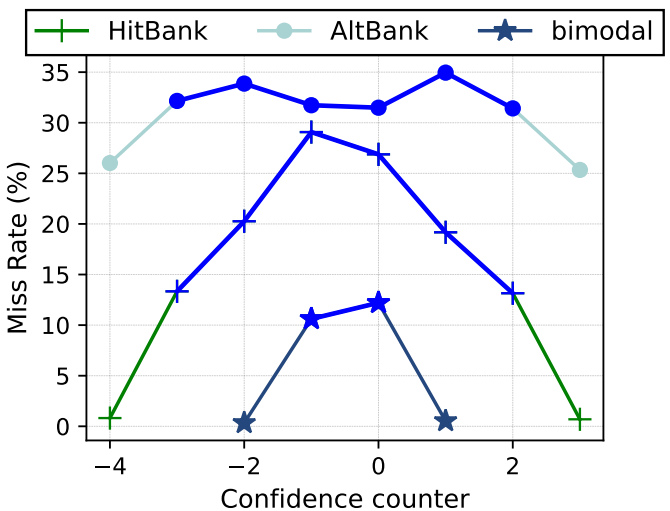
Improved accuracy reduces BW overhead and L1D pollution

UCP
① H2P Branch Detection

→ H2P Branch: a branch which has high chance of being mispredicted

UCP
① H2P Branch Detection

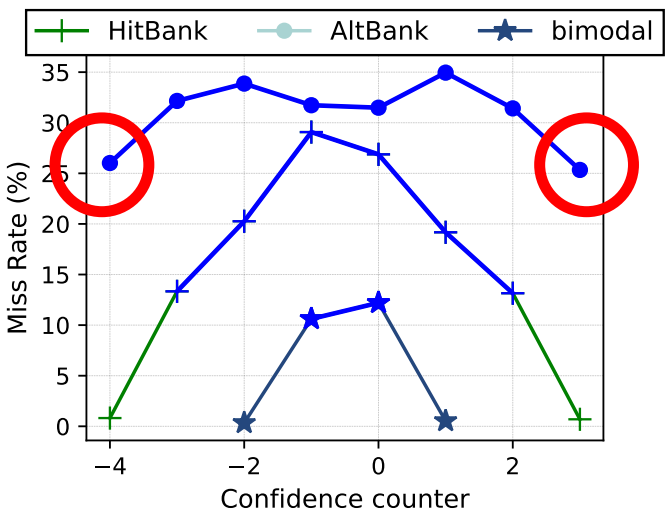
→ H2P Branch: a branch which has high chance of being mispredicted
  → TAGE-Conf[2]
    - Not saturated predictions from bimodal & TAGE banks
    - Does not consider stat corrector (SC) and loop predictor (LP)



[3] Seznec et. al. *Storage free confidence estimation for the TAGE branch predictor*

UCP
① H2P Branch Detection

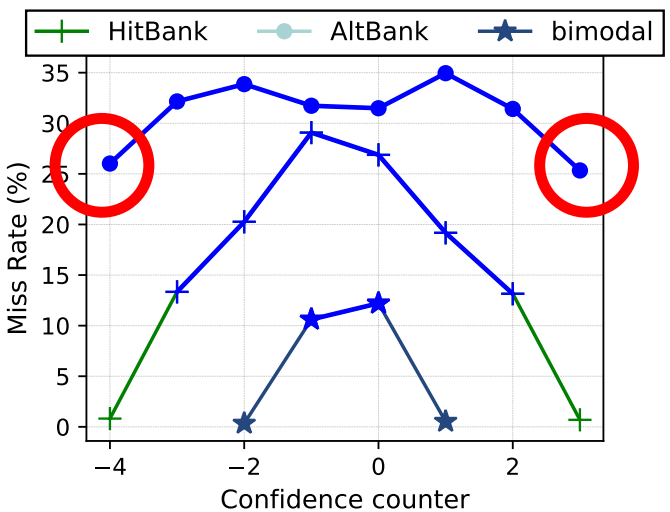
→ H2P Branch: a branch which has high chance of being mispredicted
  → TAGE-Conf[2]
    - Not saturated predictions from bimodal & TAGE banks
    - Does not consider stat corrector (SC) and loop predictor (LP)
  → UCP-Conf
    - All predictions from AltBank shows high miss rate

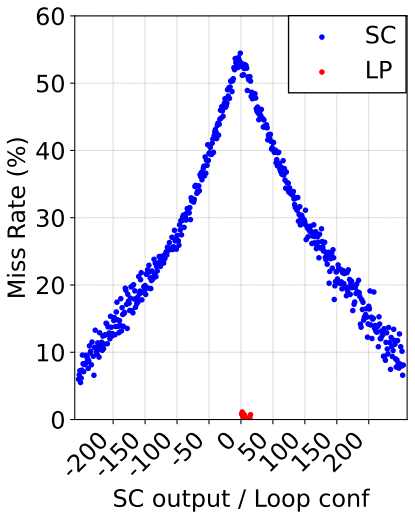


[3] Seznec et. al. *Storage free confidence estimation for the TAGE branch predictor*

UCP
① H2P Branch Detection

→ H2P Branch: a branch which has high chance of being mispredicted
  → TAGE-Conf[2]
    - Not saturated predictions from bimodal & TAGE banks
    - Does not consider stat corrector (SC) and loop predictor (LP)
  → UCP-Conf
    - All predictions from AltBank shows high miss rate
    - SC shows high miss rate

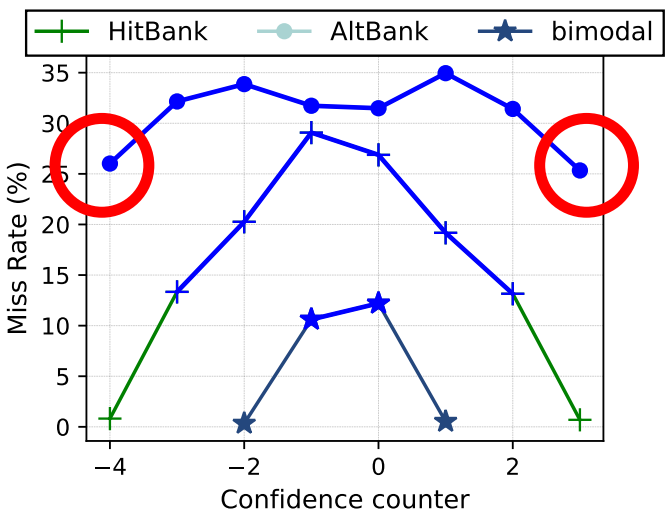


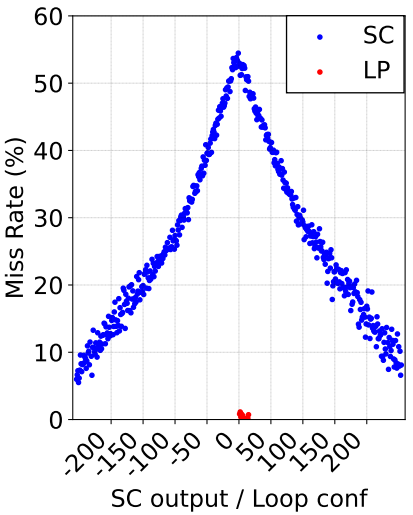


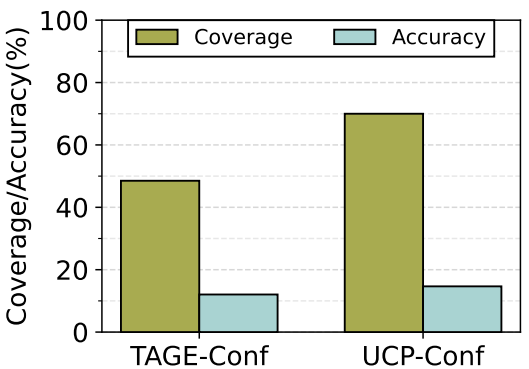
[3] Seznec et. al. *Storage free confidence estimation for the TAGE branch predictor*

UCP
① H2P Branch Detection

- → H2P Branch: a branch which has high chance of being mispredicted
  - → TAGE-Conf[2]
    - Not saturated predictions from bimodal & TAGE banks
    - Does not consider stat corrector (SC) and loop predictor (LP)
  - → UCP-Conf
    - All predictions from AltBank shows high miss rate
    - SC shows high miss rate







[3] Seznec et. al. *Storage free confidence estimation for the TAGE branch predictor*

→ Needs Branch Prediction Units

1. Banking

2. Add new predictors

→ Needs Branch Prediction Units

1. Banking
   - BTB size is critical to generate the alternate path
   - ⇒ Increase the number of banks from 16 to 32
2. Add new predictors
   - Requirements of other predictors are smaller

→ Needs Branch Prediction Units

1. Banking
   - BTB size is critical to generate the alternate path

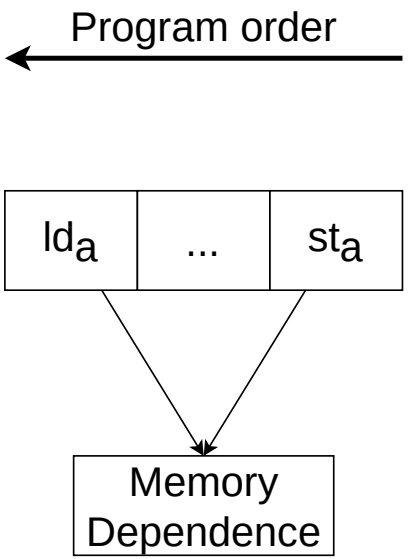
   ⇒ Increase the number of banks from 16 to 32
2. Add new predictors
   - Requirements of other predictors are smaller

   ⇒ Alt BP: 8KB TAGE-SC-L

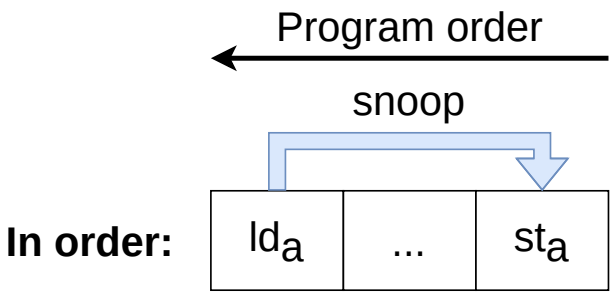
   ⇒ Alt Indirect: 4KB ITTAGE

   ⇒ Alt RAS: 16-entry

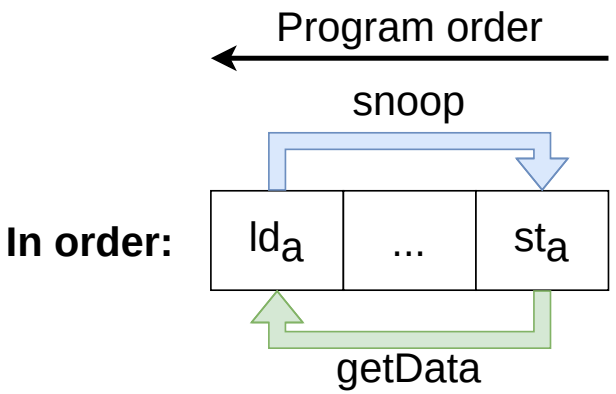
MDP: CRASH COURSE



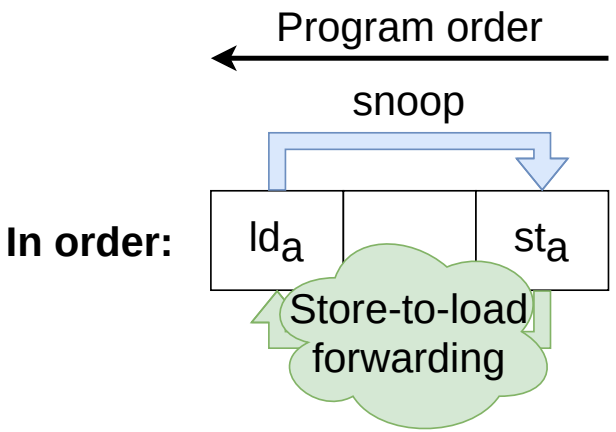
MDP: CRASH COURSE



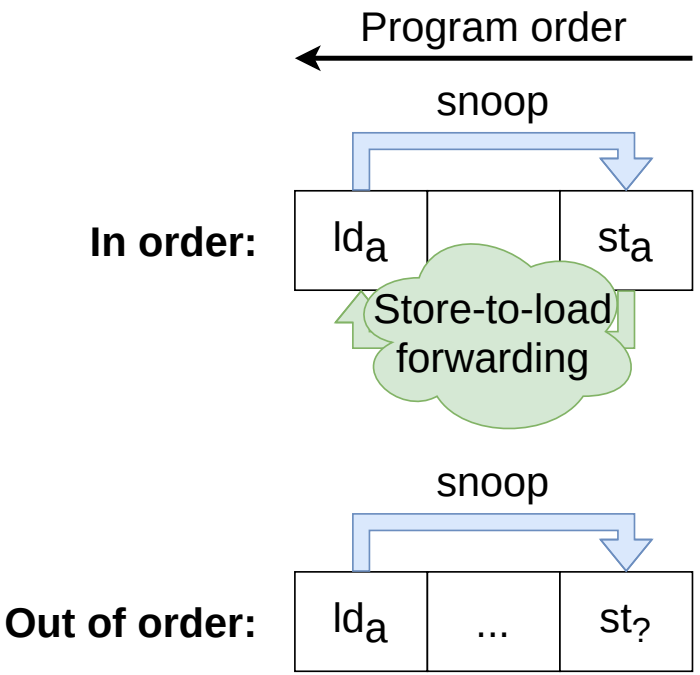
MDP: CRASH COURSE



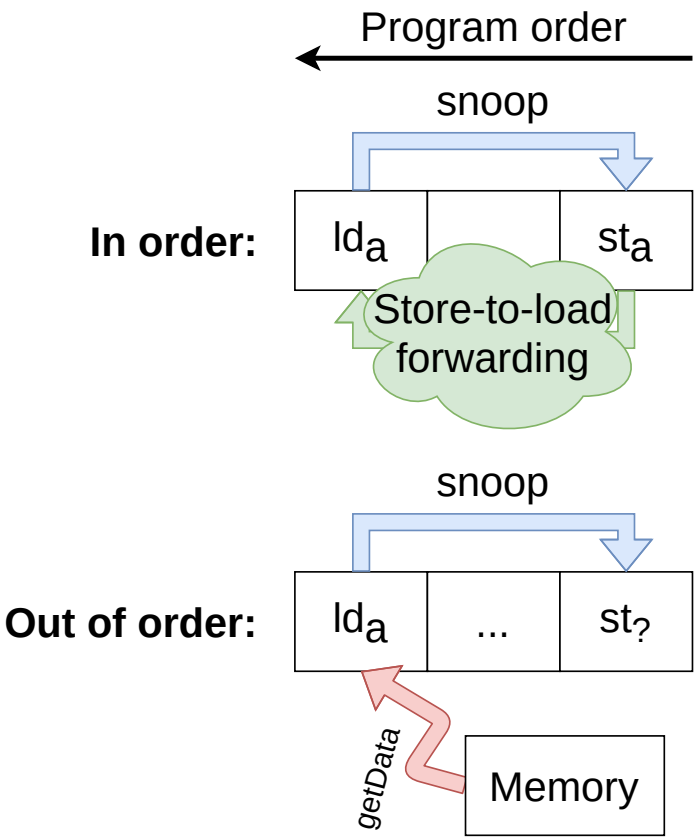
MDP: Crash course



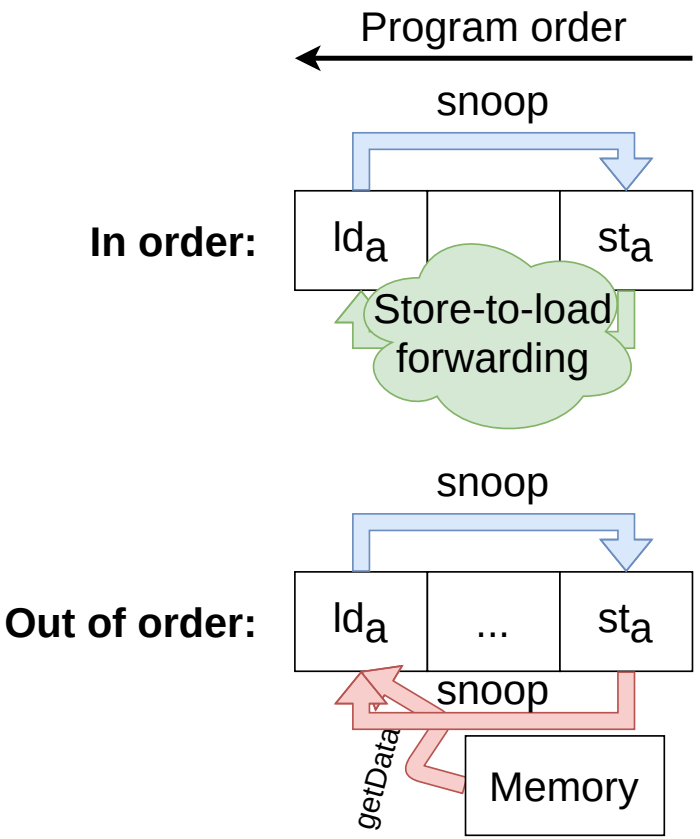
MDP: Crash course



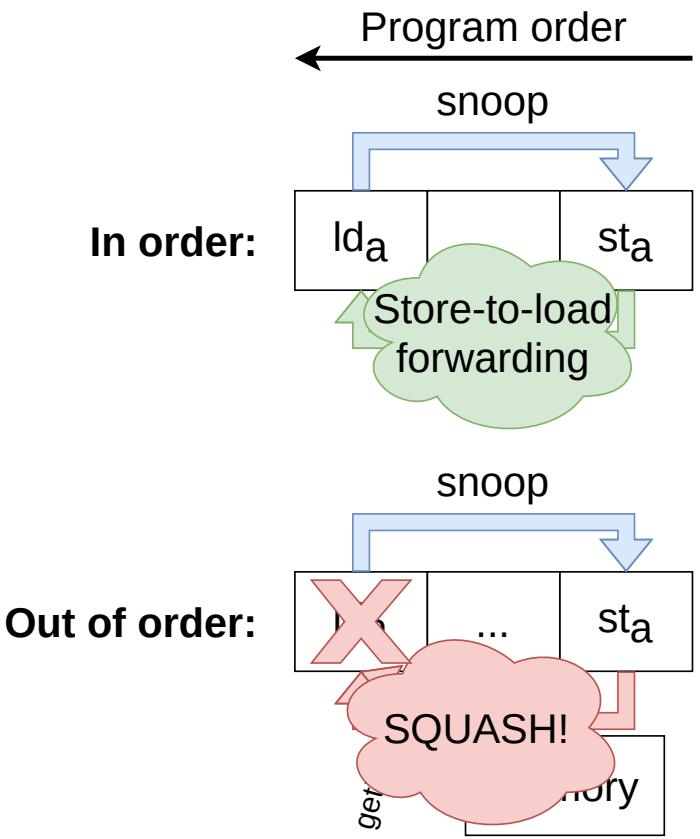
MDP: Crash course



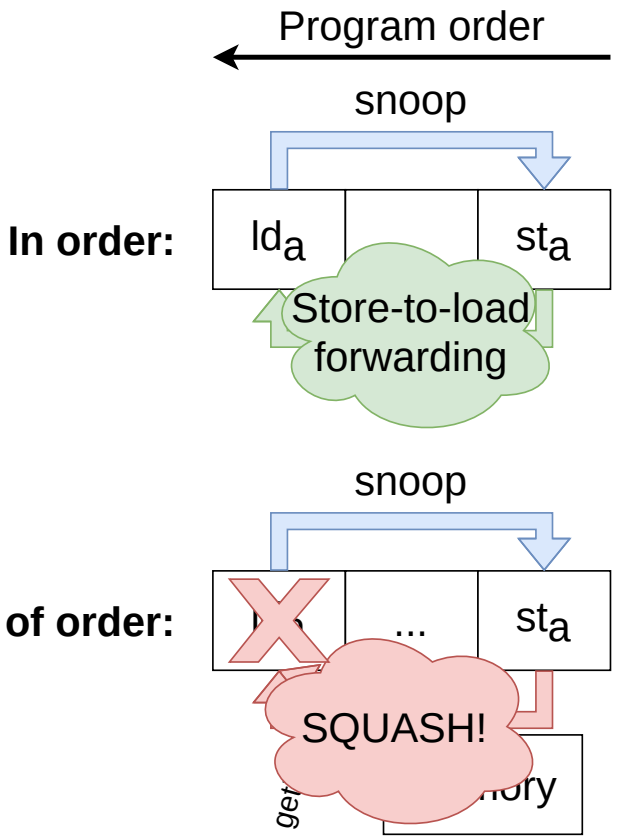
MDP: Crash course



MDP: Crash course



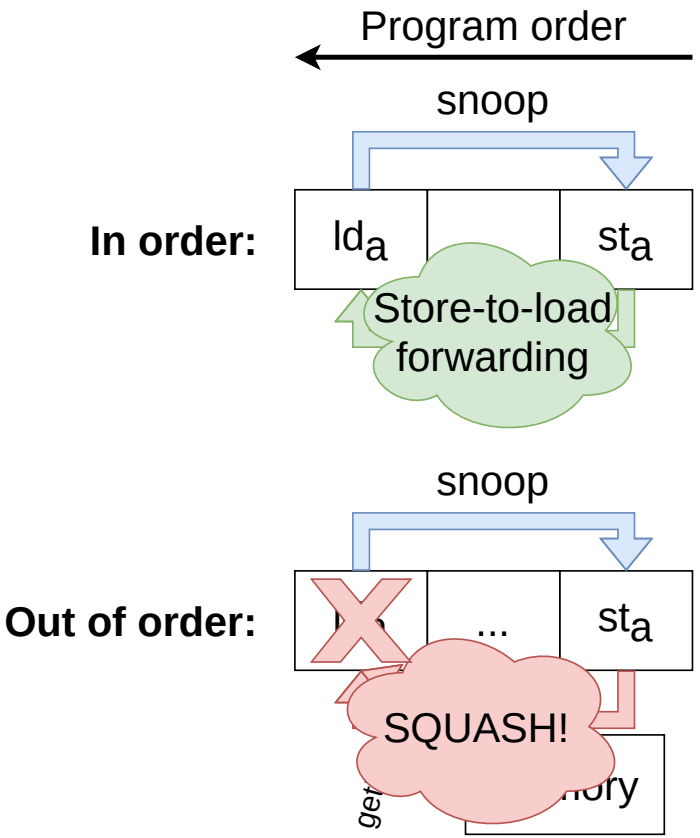
MDP: CRASH COURSE



THE PREDICTION

Is they any previous dependent store?

MDP: CRASH COURSE

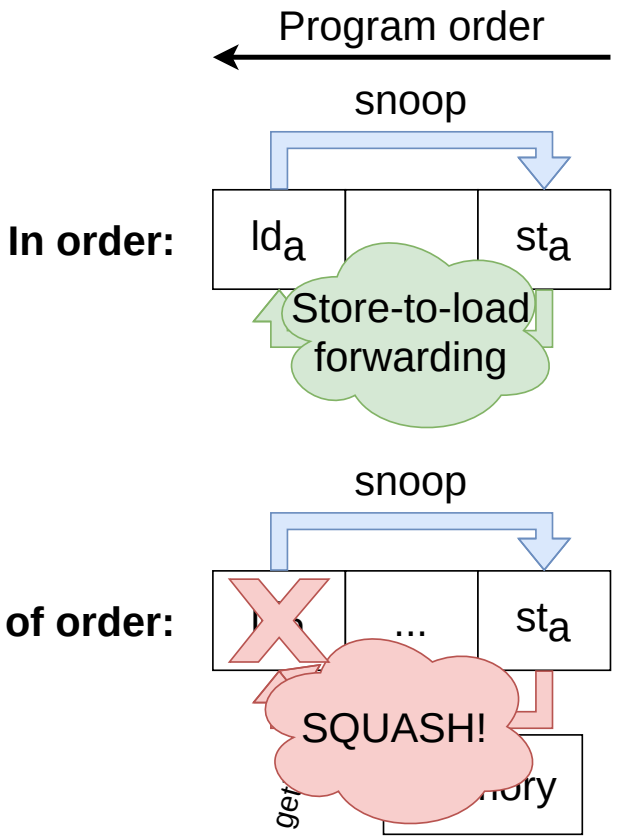


THE PREDICTION

Is they any previous dependent store?

- NO: Execute the load
- YES: Stall (wait) until it executes

MDP: CRASH COURSE



THE PREDICTION

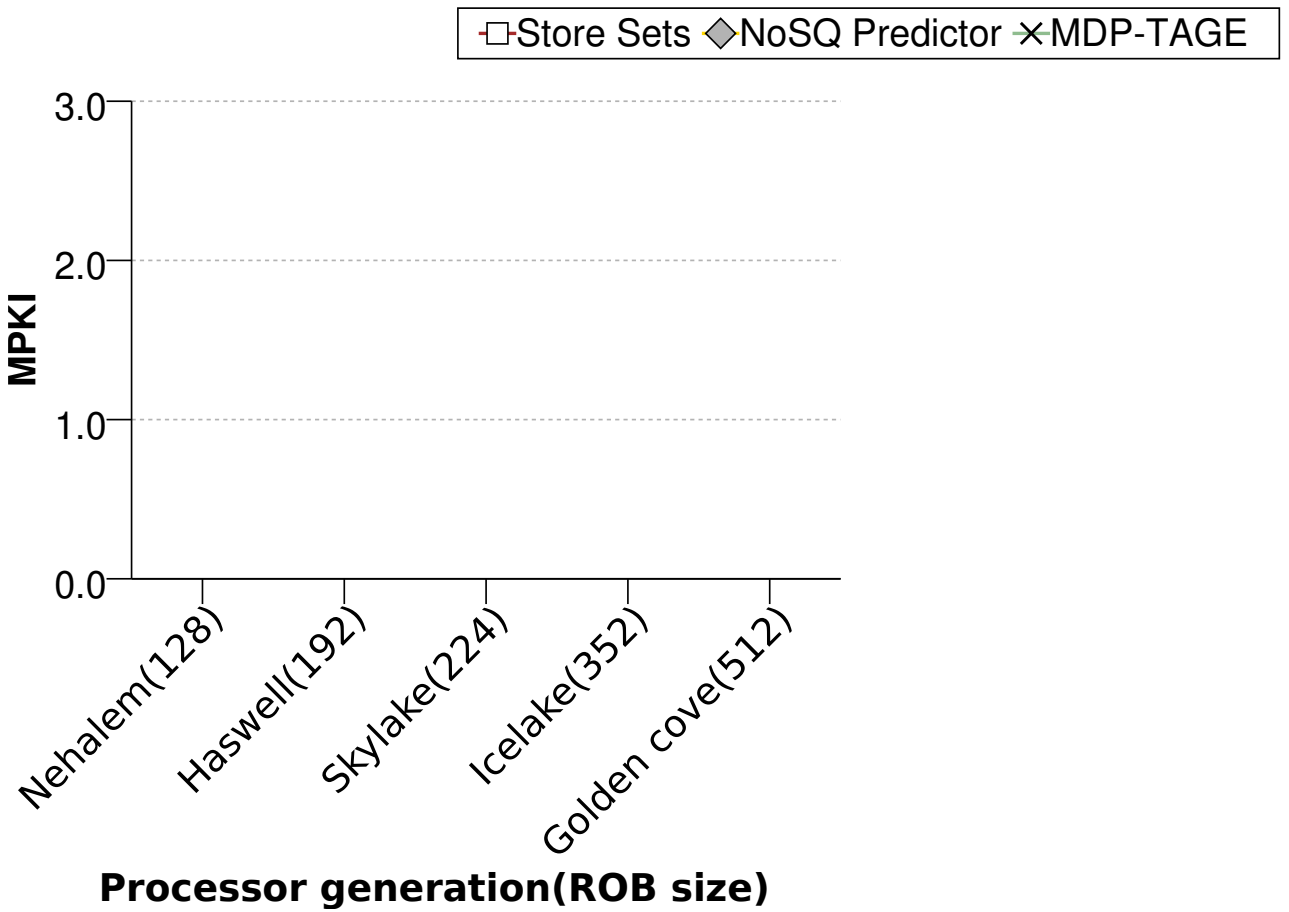
Is they any previous dependent store?

- NO: Execute the load
- YES: Stall (wait) until it executes

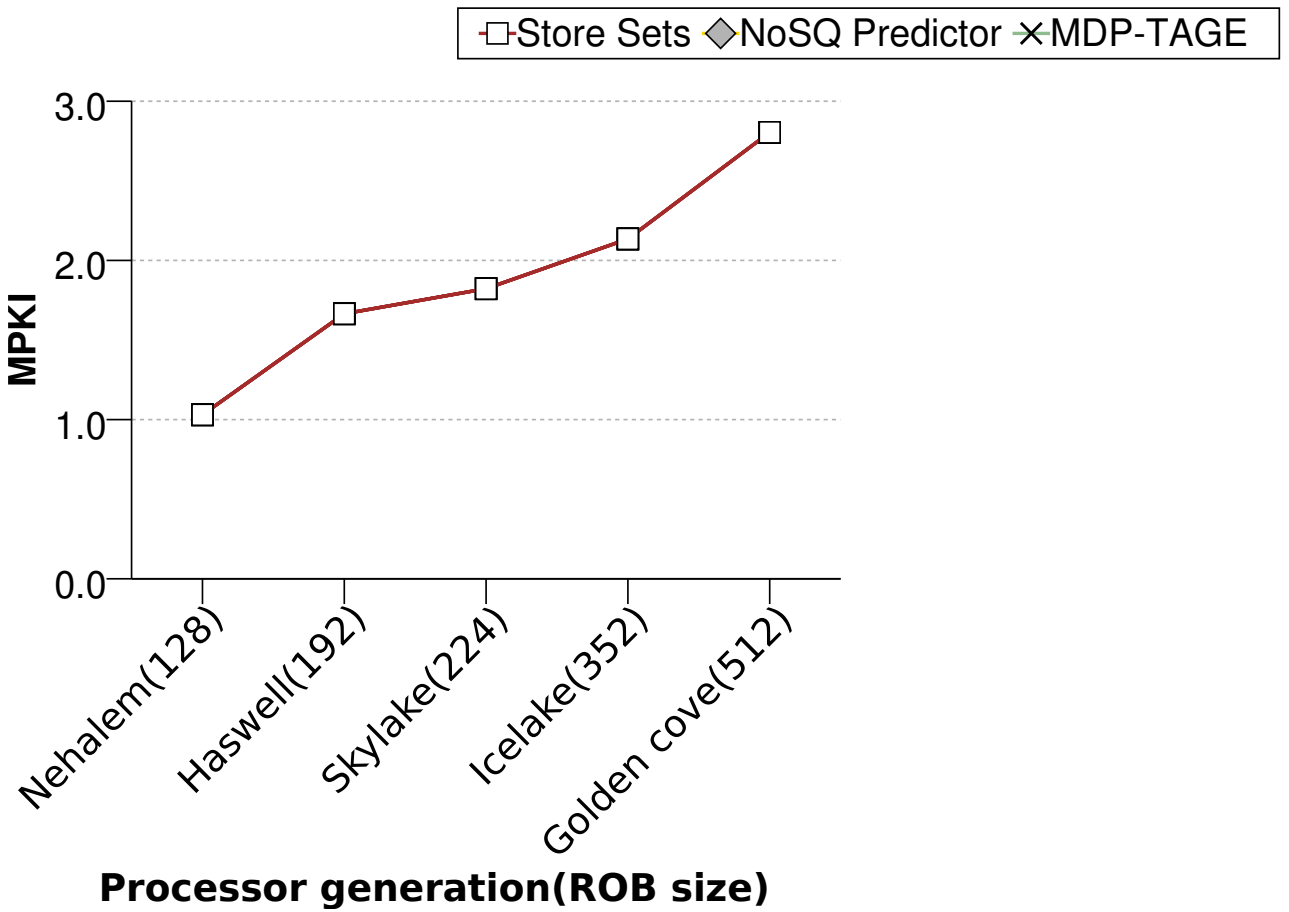
THE MISPREDICTION

- Execute, but conflict ☹
- Stall, but no dependence 😐

MDP, WAS NOT SOLVED IN THE 90'S?

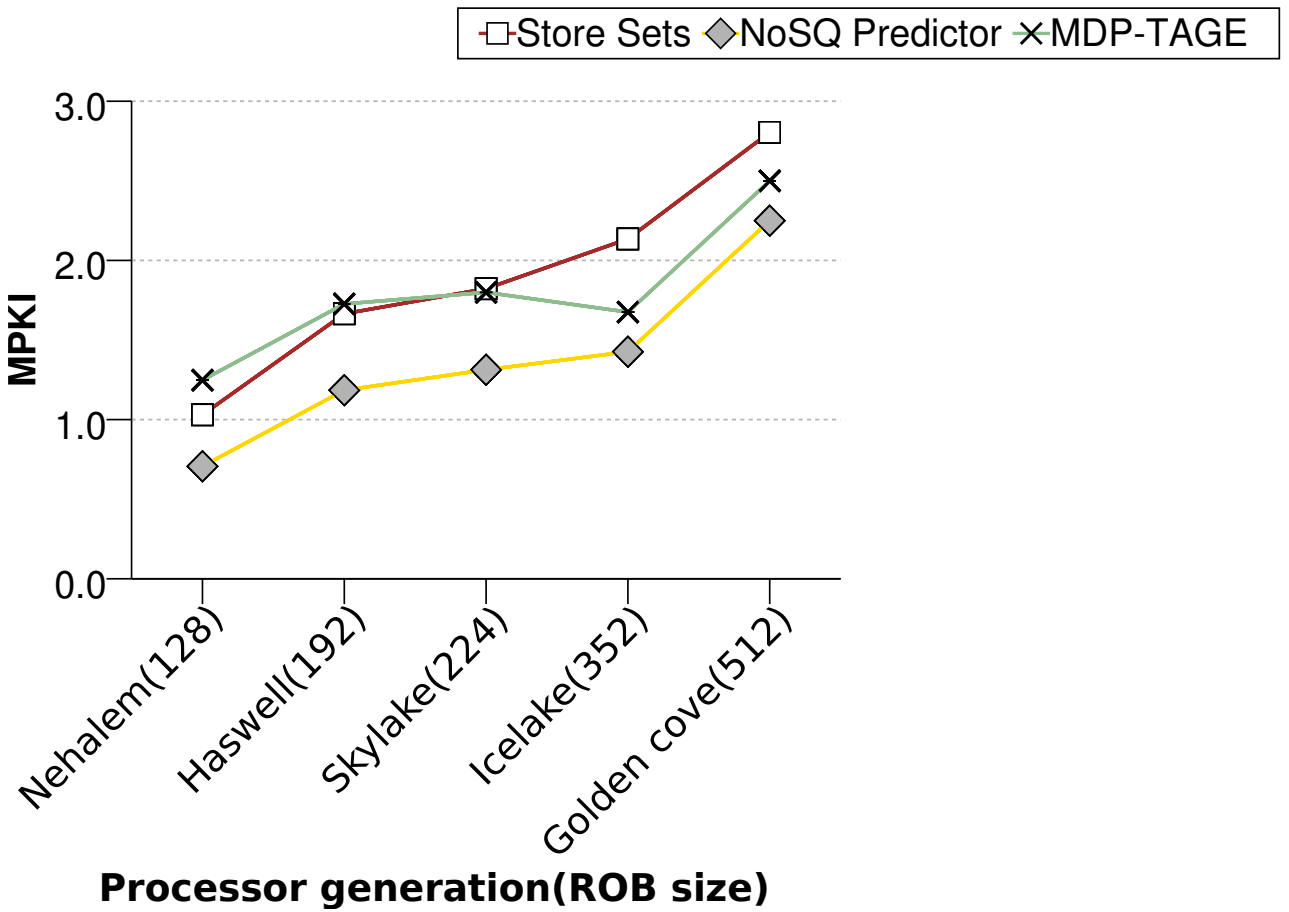


MDP, WAS NOT SOLVED IN THE 90'S?



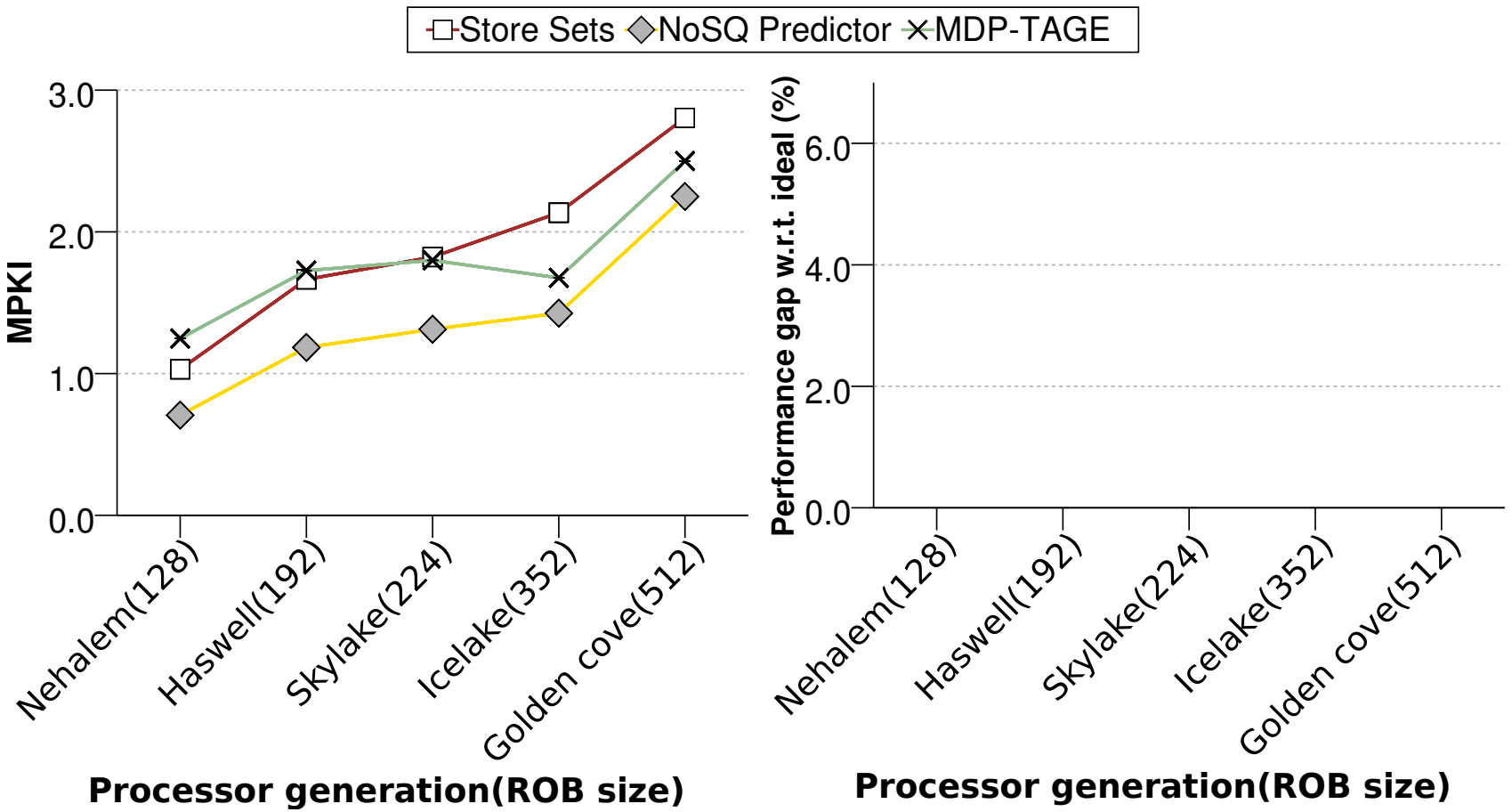
Memory dependence mispredictions increase with processor aggressiveness

MDP, WAS NOT SOLVED IN THE 90'S?



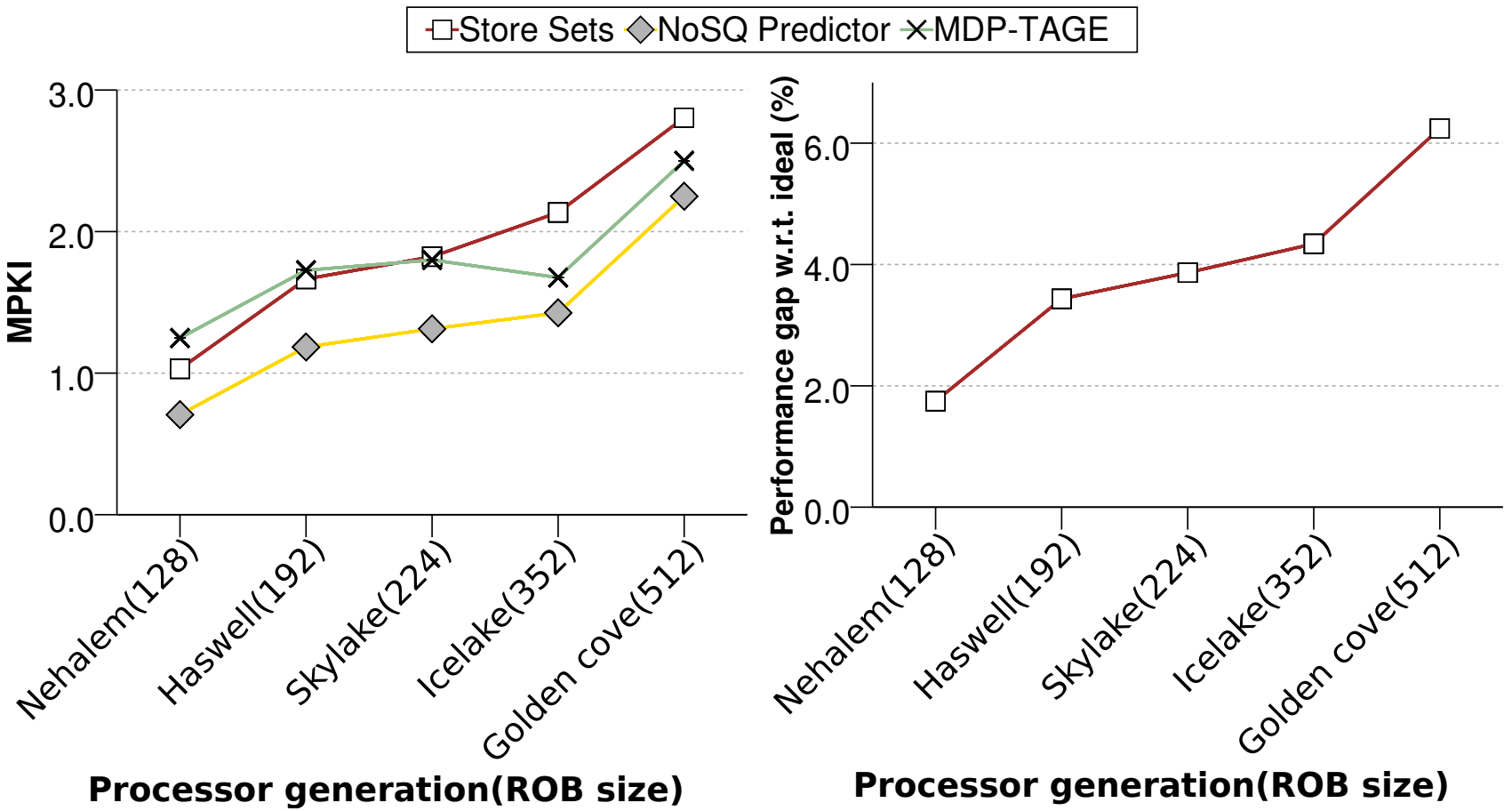
Memory dependence mispredictions increase with processor aggressiveness

MDP, WAS NOT SOLVED IN THE 90'S?



Memory dependence mispredictions increase with processor aggressiveness

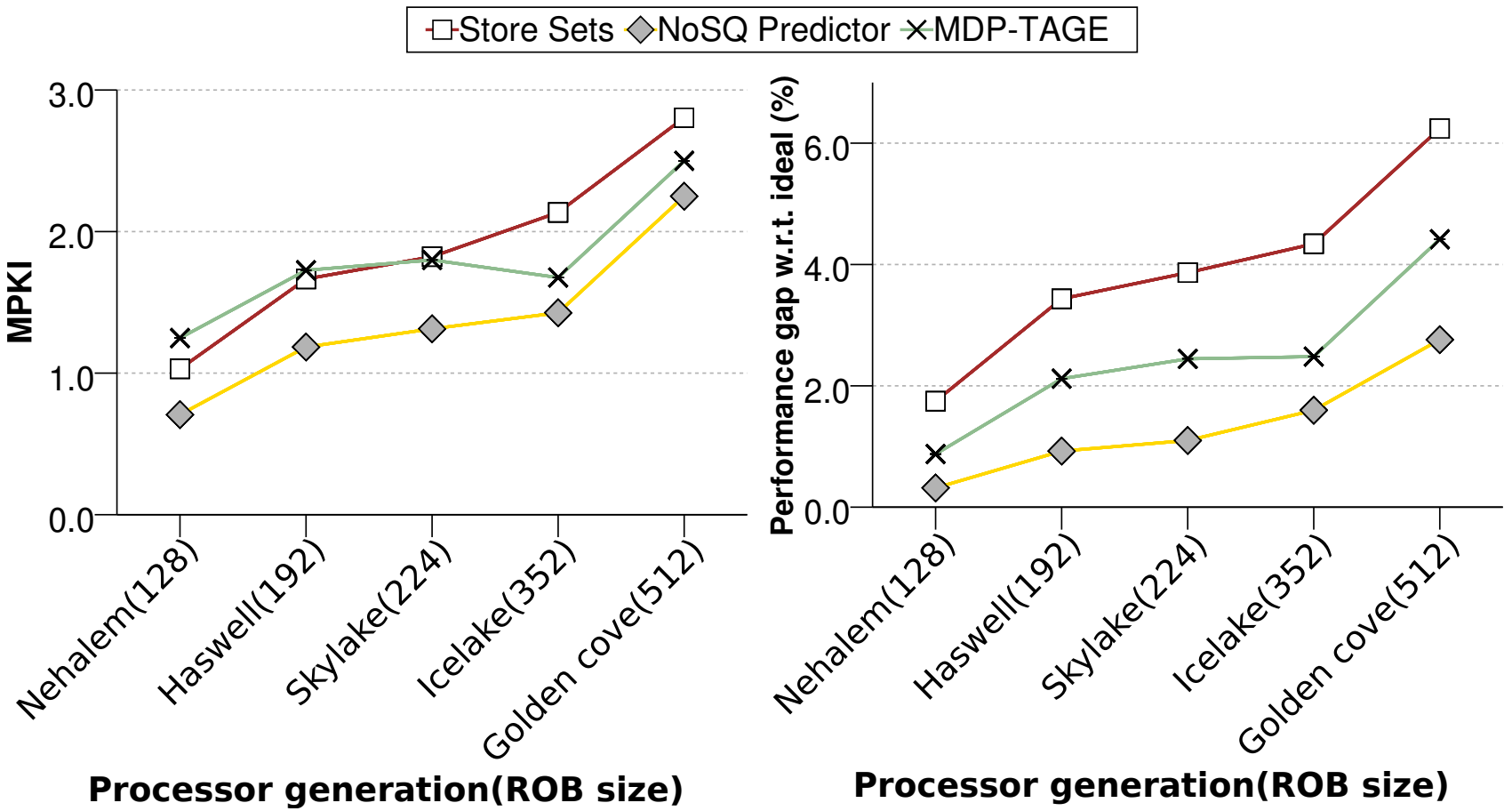
MDP, WAS NOT SOLVED IN THE 90'S?



Memory dependence mispredictions increase with processor aggressiveness

Squashes impact performance more

MDP, WAS NOT SOLVED IN THE 90'S?



Memory dependence mispredictions increase with processor aggressiveness

Squashes impact performance more

PHAST: THE TWO PILLARS

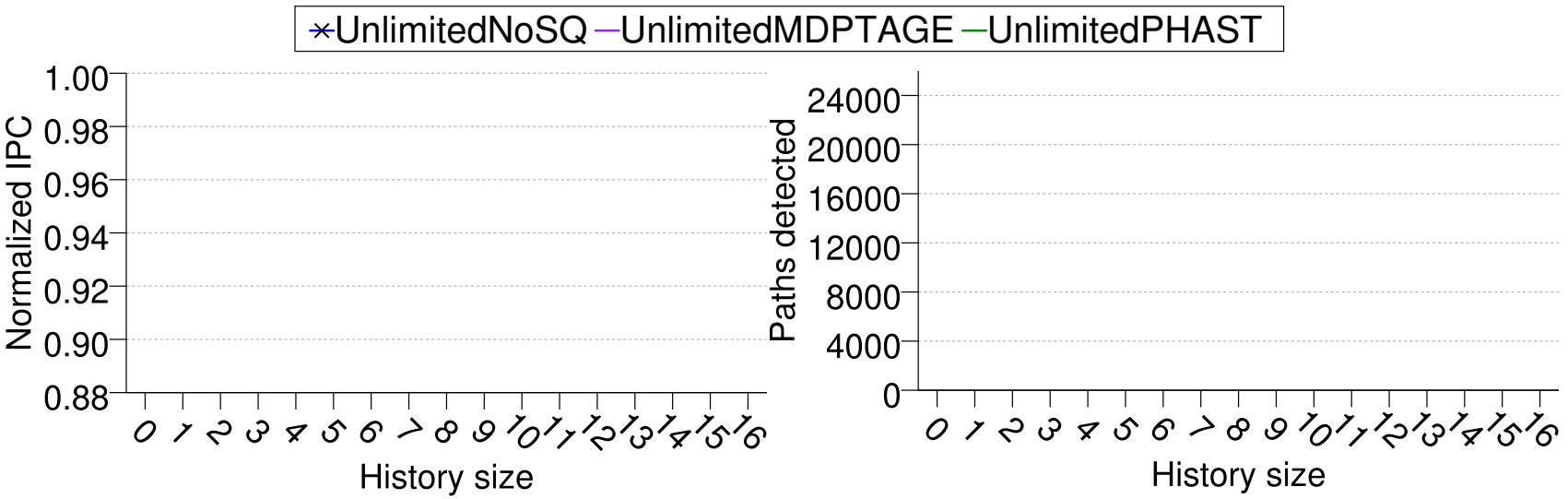
PHAST (**P**at**H**-**A**ware **ST**ore-distance) mem. dep. predictor

1. Precisely identifies a single dependent store
2. Learns the *proper* context information for each conflict

COMPARISON OF IDEAL CONTEXT-SENSITIVE MDPS

[noframenumbering]

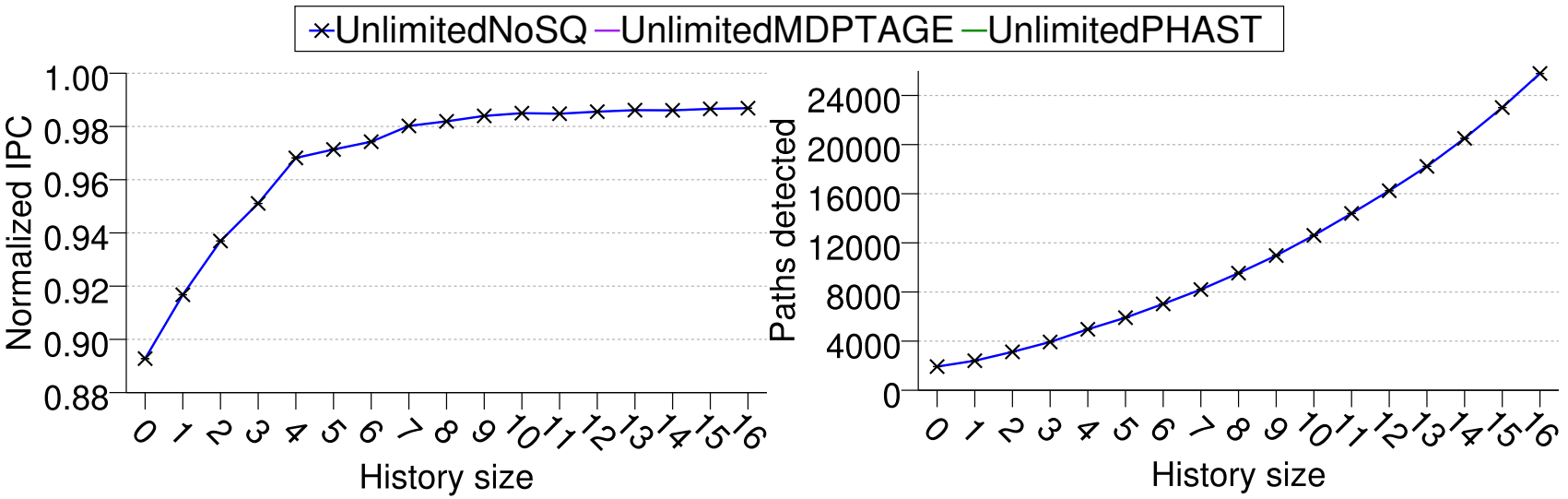
- IPC normalized to an ideal predictor!



COMPARISON OF IDEAL CONTEXT-SENSITIVE MDPS

[noframenumbering]

- IPC normalized to an ideal predictor!

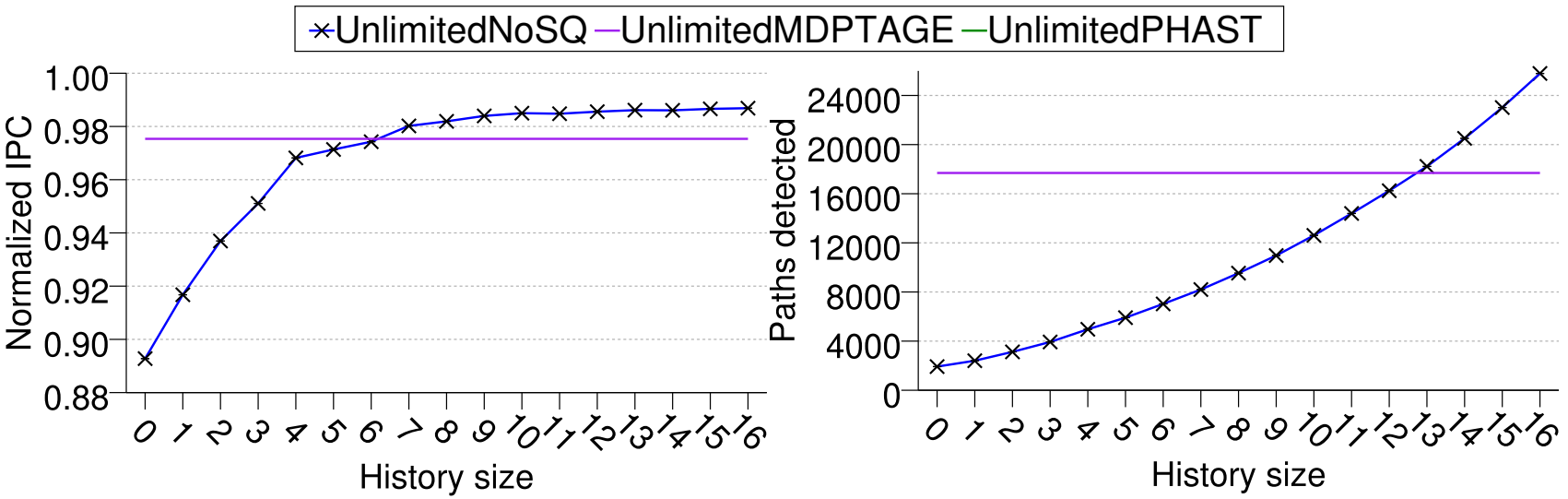


- The larger is the history, the more is the accuracy, but the more are the hardware requirements

COMPARISON OF IDEAL CONTEXT-SENSITIVE MDPS

[noframenumbering]

- IPC normalized to an ideal predictor!

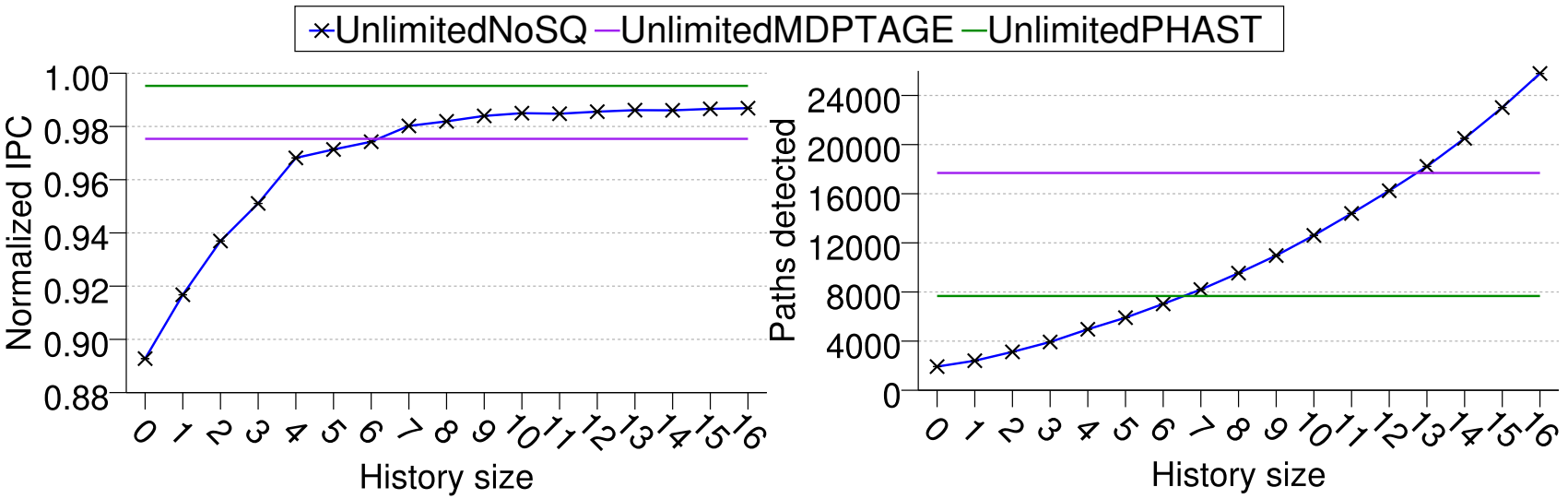


- The larger is the history, the more is the accuracy, but the more are the hardware requirements

COMPARISON OF IDEAL CONTEXT-SENSITIVE MDPS

[noframenumbering]

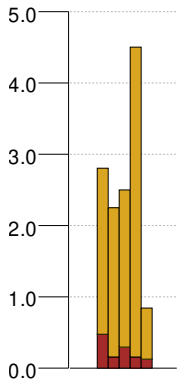
- IPC normalized to an ideal predictor!



- The larger is the history, the more is the accuracy, but the more are the hardware requirements
- Unlimited PHAST peformance falls 0.5% behind an ideal predictor! $\Rightarrow$ Paths are a good proxy for conflicts

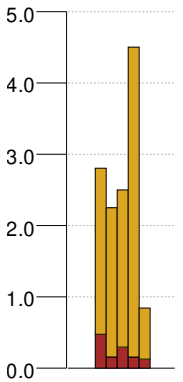
RESULTS

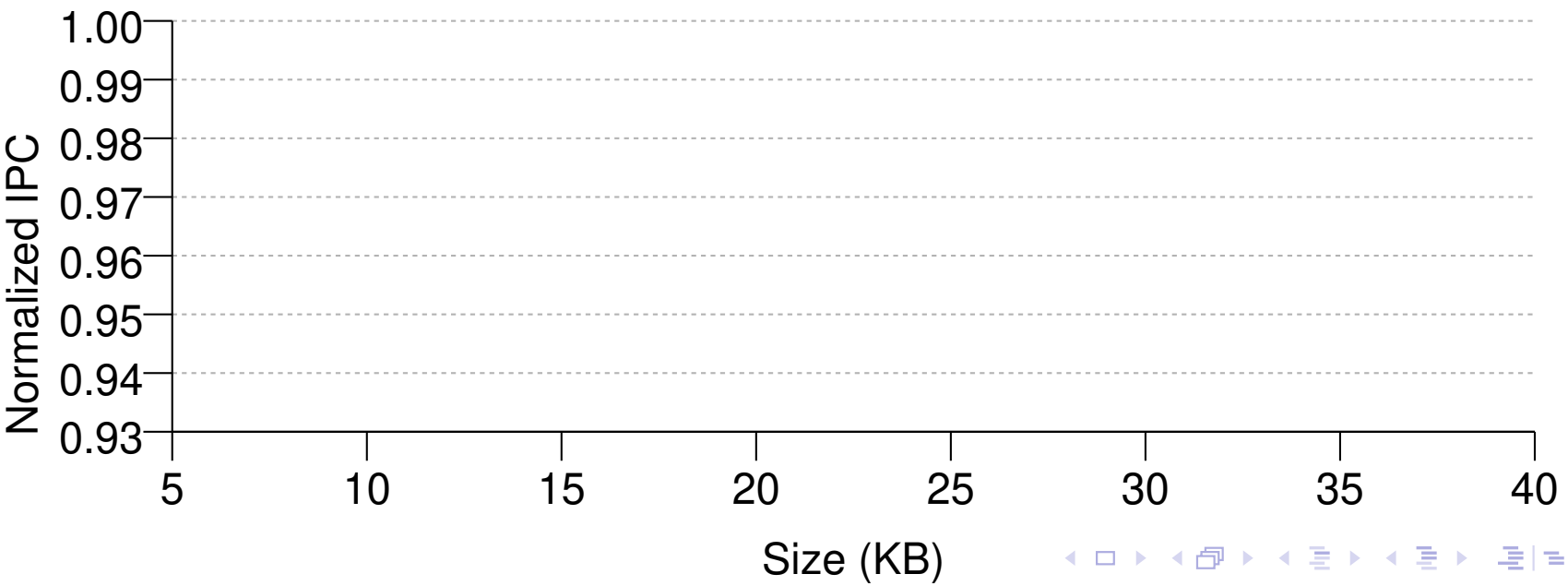
- Overall Misprediction reduction: 62% over NoSQ



RESULTS

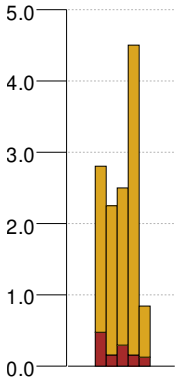
- Overall Misprediction reduction: 62% over NoSQ
- IPC normalized to an ideal predictor!

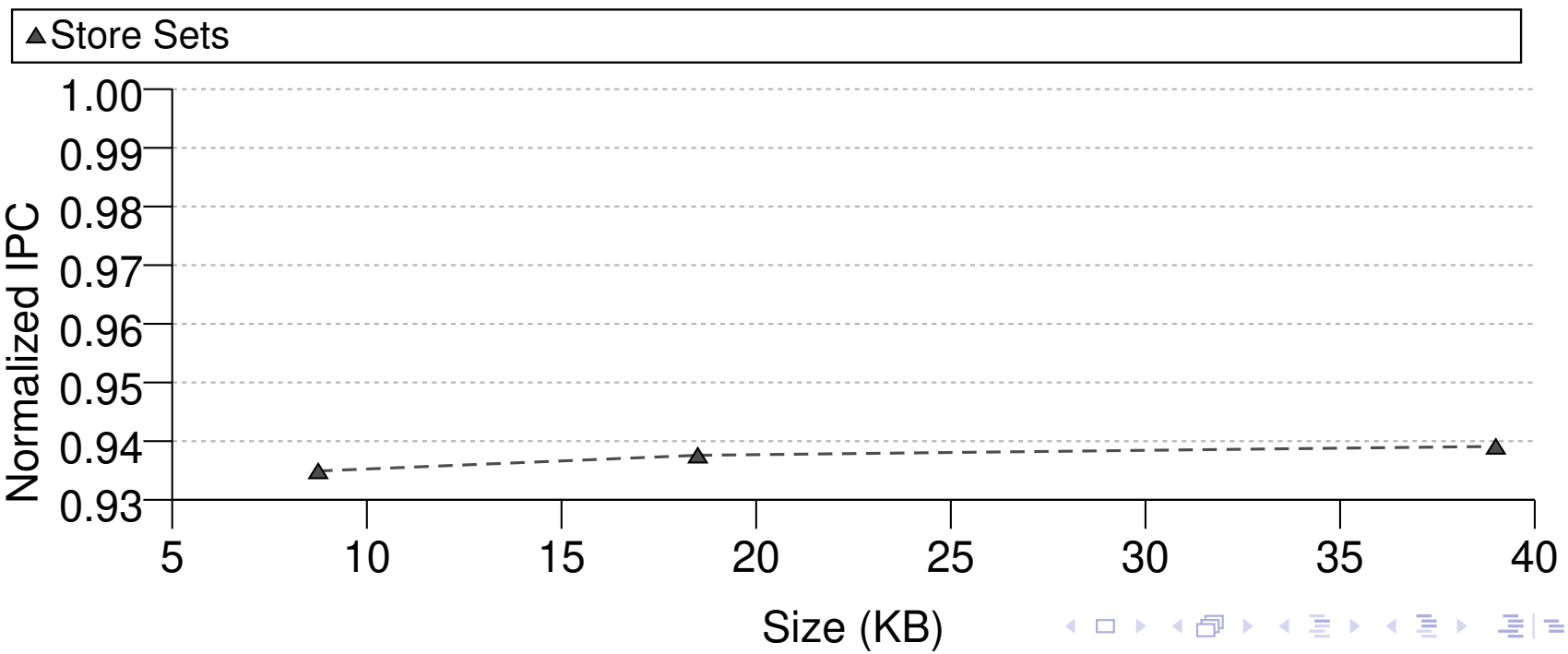




RESULTS

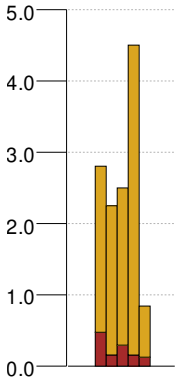
- Overall Misprediction reduction: 62% over NoSQ
- IPC normalized to an ideal predictor!

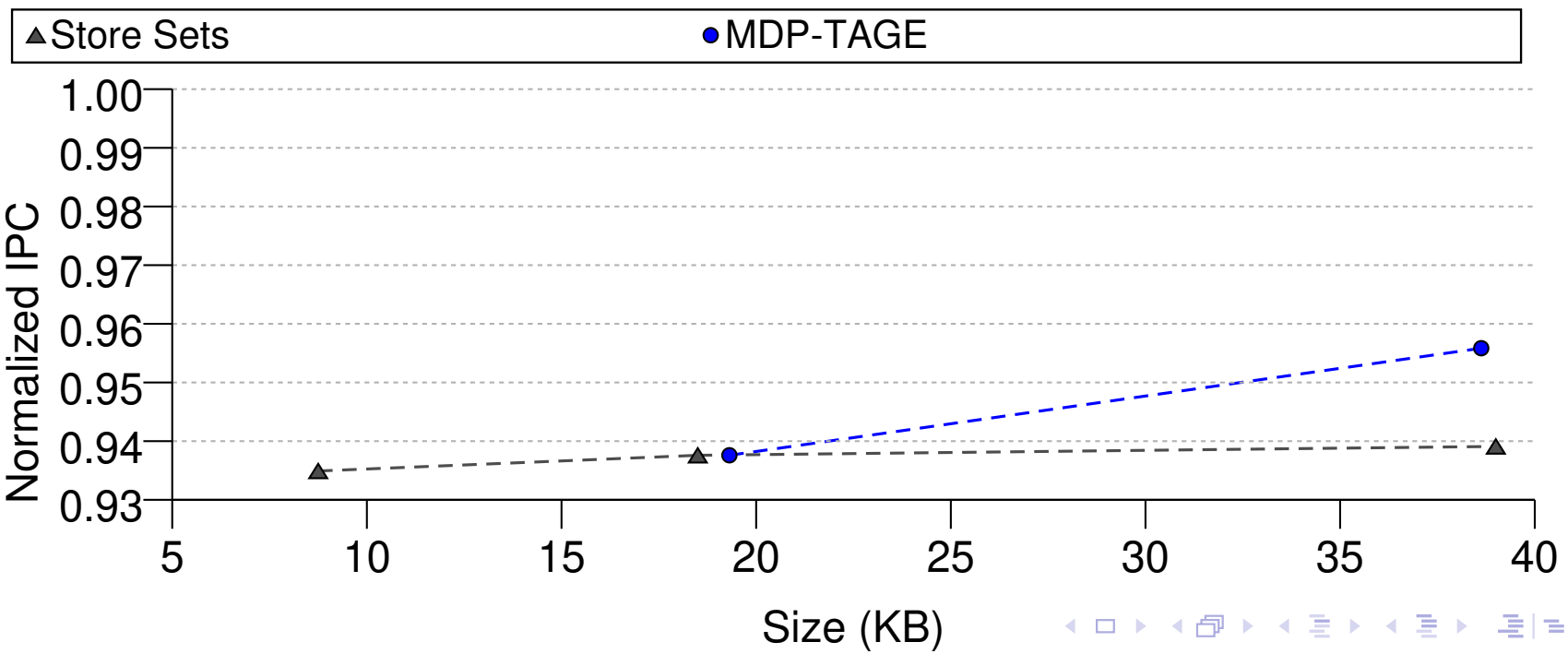




RESULTS

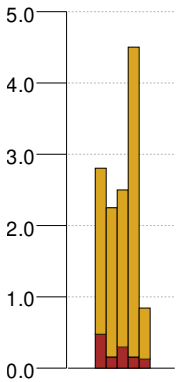
- Overall Misprediction reduction: 62% over NoSQ
- IPC normalized to an ideal predictor!

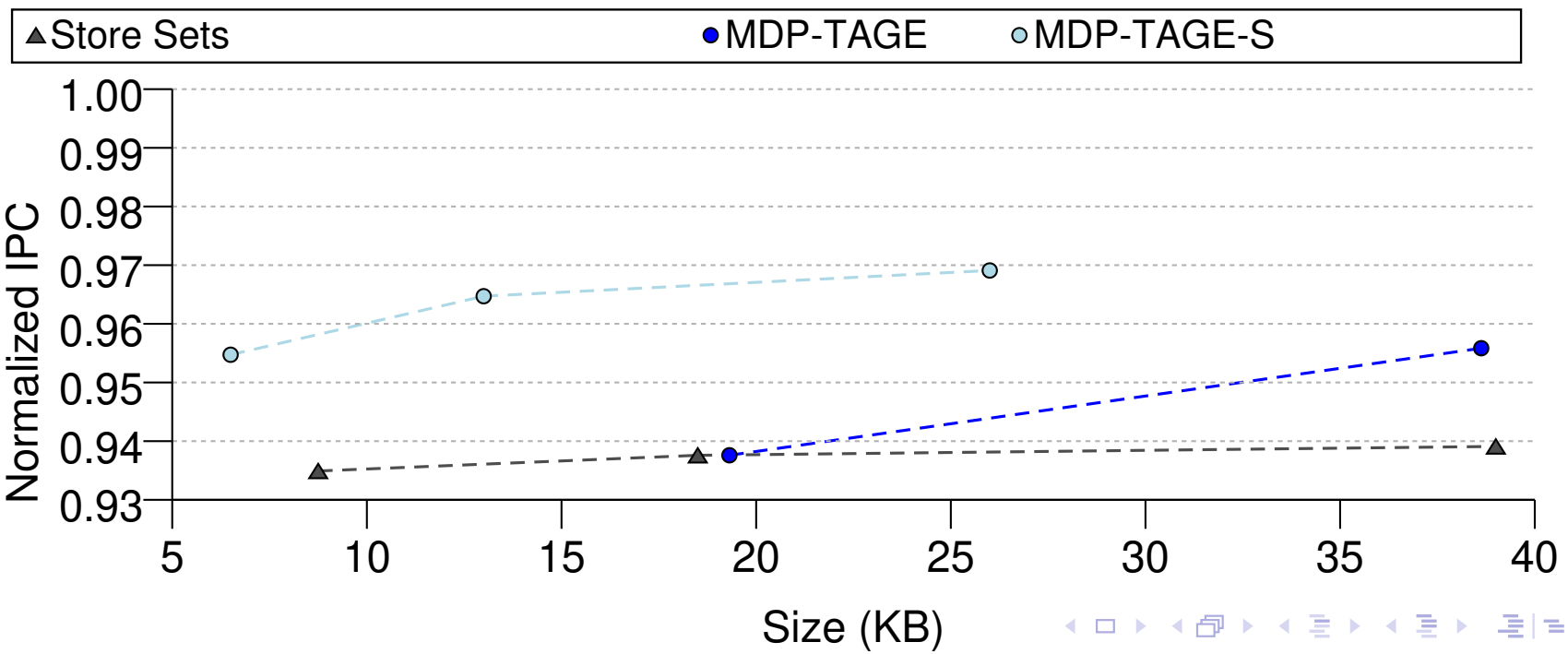




RESULTS

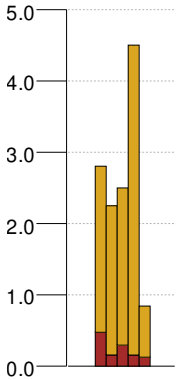
- Overall Misprediction reduction: 62% over NoSQ
- IPC normalized to an ideal predictor!

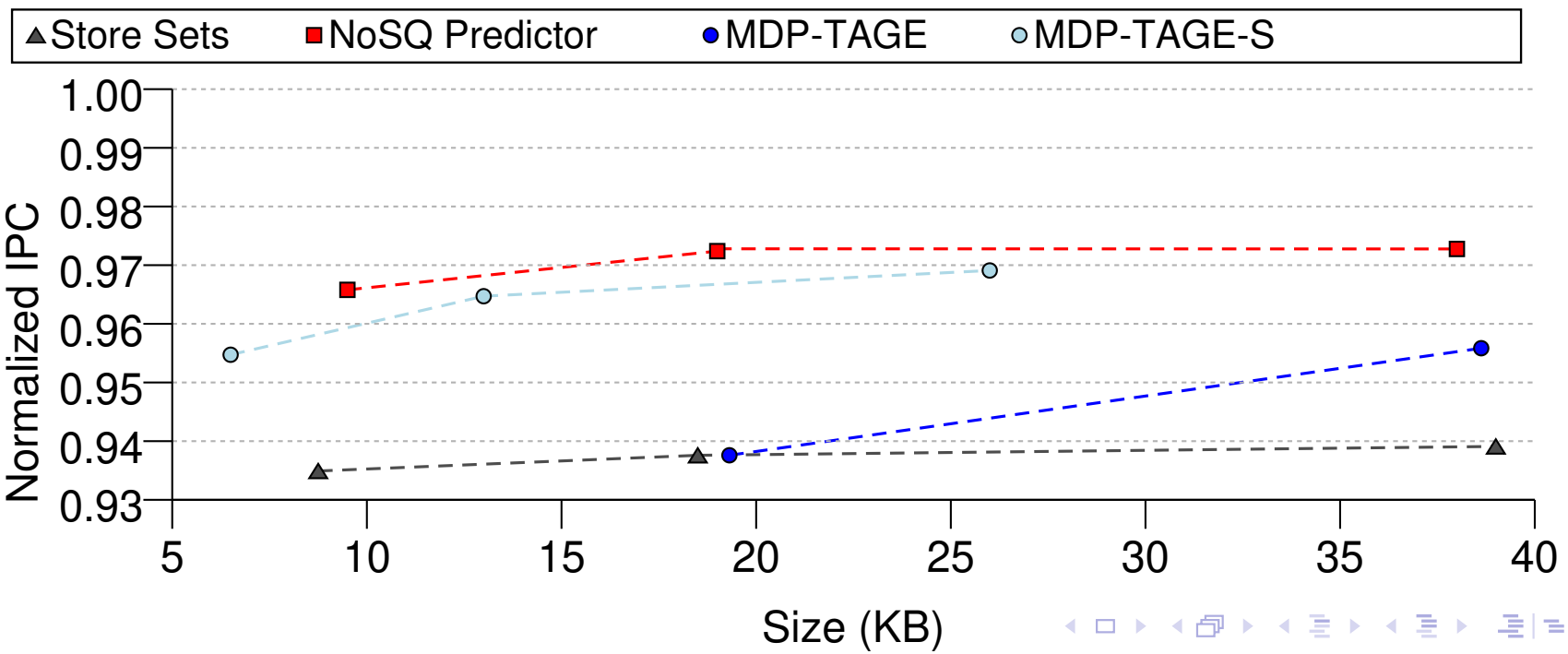




RESULTS

- Overall Misprediction reduction: 62% over NoSQ
- IPC normalized to an ideal predictor!





RESULTS

- Overall Misprediction reduction: 62% over NoSQ
- IPC normalized to an ideal predictor!