alexandre.jamet@bsc.es

A Two Level Neural Approach Combining Off-Chip Prediction with Adaptive Prefetch Filtering

Alexandre Valentin Jamet[1], Georgios Vavouliotis[2], Lluc Alvarez[1], Daniel A. Jiménez[3], Marc Casas[1]

[1]Barcelona Supercomputing Center
[2]Huawei Zurich Research Center
[3]Texas A&M University

Executive Summary

Executive Summary

- Emerging workloads (e.g., graph processing) have massive data footprints
  - Significant pressure on the caches

Executive Summary

- Emerging workloads (e.g., graph processing) have massive data footprints
  - Significant pressure on the caches
- Latency Tolerance Techniques:
  - Off-Chip Prediction (*e.g.*, Hermes (MICRO'22))
  - Data Prefetching (*e.g.*, PPF (ISCA'19))
  - Cache bypassing (*e.g.*, Multiperspective Reuse Prediction (MICRO'17))
  - Disruptive cache designs (*e.g.*, Distill Cache (HPCA'07))

Executive Summary

- Emerging workloads (e.g., graph processing) have massive data footprints
  - Significant pressure on the caches
- Latency Tolerance Techniques:
  - **Off-Chip Prediction** (*e.g.*, Hermes (MICRO'22))
  - **Data Prefetching** (*e.g.*, PPF (ISCA'19))
  - Cache bypassing (*e.g.*, Multiperspective Reuse Prediction (MICRO'17))
  - Disruptive cache designs (*e.g.*, Distill Cache (HPCA'07))

Executive Summary

- Emerging workloads (e.g., graph processing) have massive data footprints
  - Significant pressure on the caches
- Latency Tolerance Techniques:
  - **Off-Chip Prediction** (*e.g.*, Hermes (MICRO'22))
  - **Data Prefetching** (*e.g.*, PPF (ISCA'19))
  - Cache bypassing (*e.g.*, Multiperspective Reuse Prediction (MICRO'17))
  - Disruptive cache designs (*e.g.*, Distill Cache (HPCA'07))

- **Limitations** in **Off-Chip Prediction** mechanisms:
  - Balancing speed-up with increased DRAM transactions
  - Adaptability issues in prefetch filtering approaches

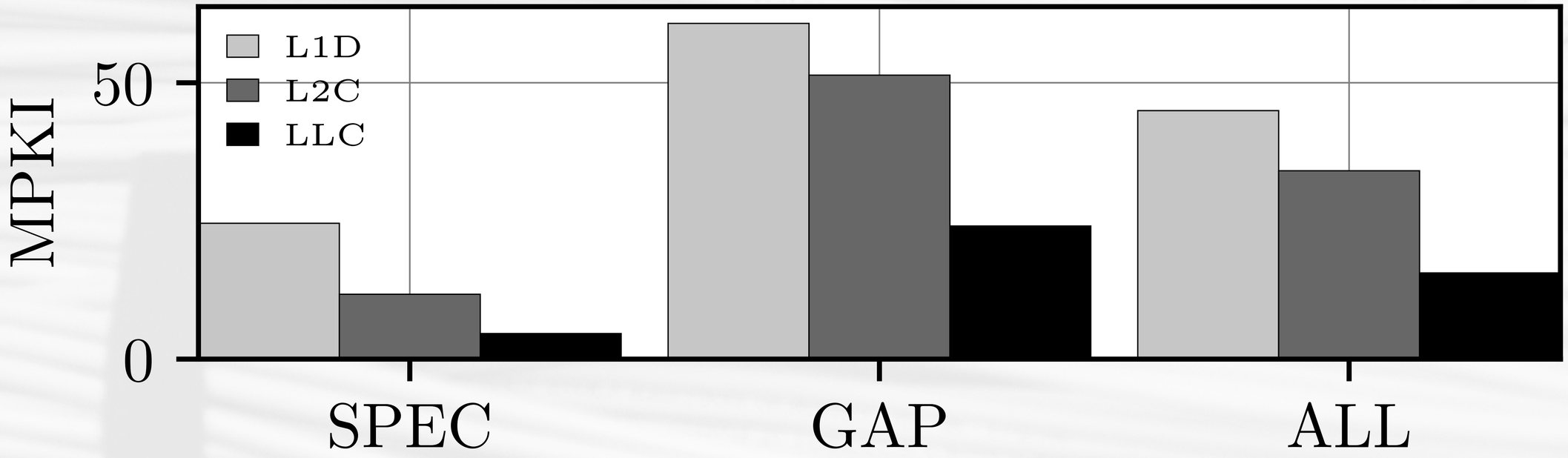
Executive Summary

- Emerging workloads (e.g., graph processing) have massive data footprints
  - Significant pressure on the caches
- Latency Tolerance Techniques:
  - **Off-Chip Prediction** (*e.g.*, Hermes (MICRO'22))
  - **Data Prefetching** (*e.g.*, PPF (ISCA'19))
  - Cache bypassing (*e.g.*, Multiperspective Reuse Prediction (MICRO'17))
  - Disruptive cache designs (*e.g.*, Distill Cache (HPCA'07))

- **Limitations** in **Off-Chip Prediction** mechanisms:
  - Balancing speed-up with increased DRAM transactions
  - Adaptability issues in prefetch filtering approaches
- **Our approach**
  - Make **Off-Chip prediction highly accurate**
  - We leverage Off-Chip prediction properties to drive **Adaptive Prefetch Filtering**

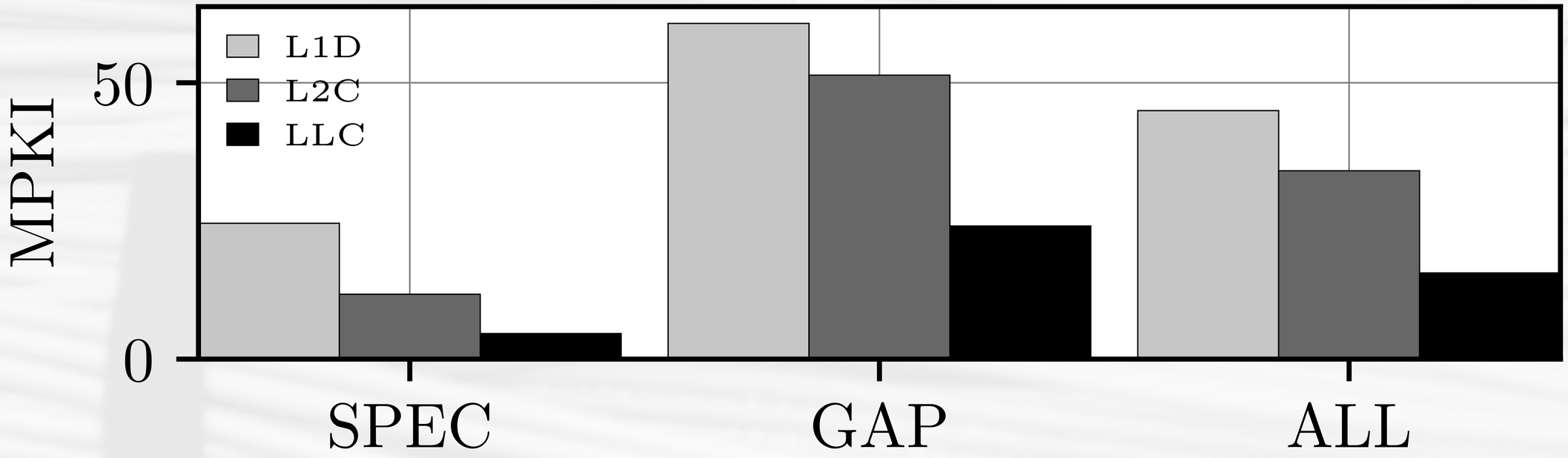
Cache Behaviour of Modern Workloads

Cache Behaviour of Modern Workloads



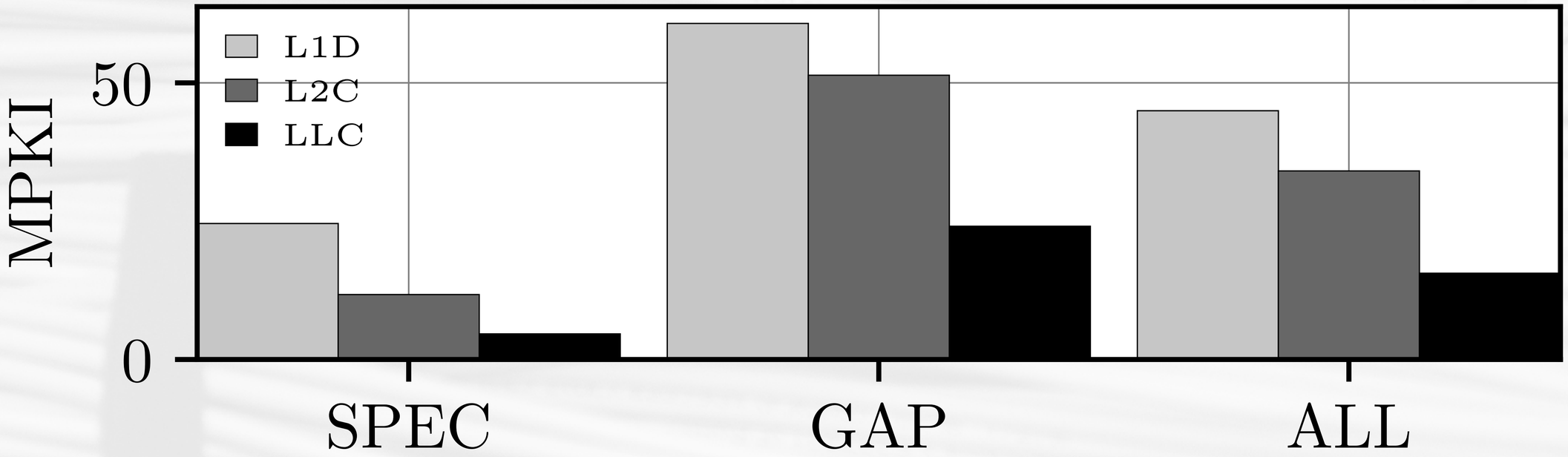
- For SPEC and GAP workloads, we analyse the load in the cache hierarchy

Cache Behaviour of Modern Workloads



- For SPEC and GAP workloads, we analyse the load in the cache hierarchy
- On average, 34.7% of L1D misses eventually require a DRAM access
  - Graph workloads exhibit **irregular access pattern** that caches can hardly capture

Cache Behaviour of Modern Workloads

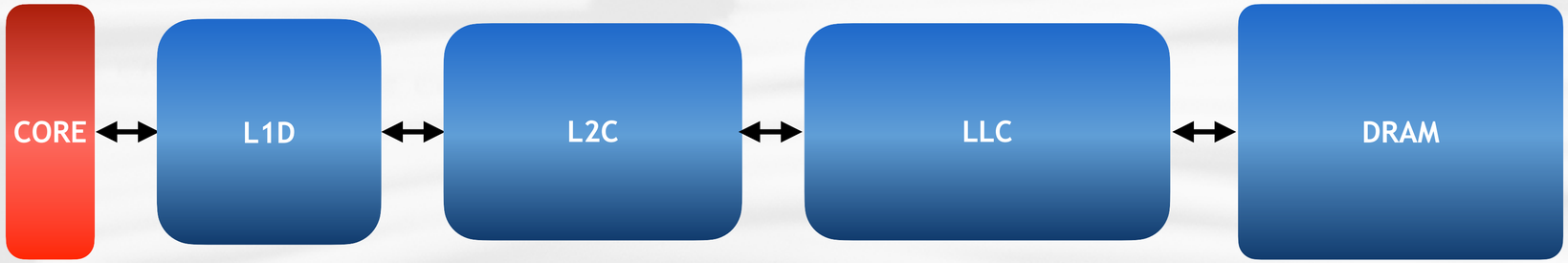


- For SPEC and GAP workloads, we analyse the load in the cache hierarchy
- On average, 34.7% of L1D misses eventually require a DRAM access
  - Graph workloads exhibit **irregular access pattern** that caches can hardly capture

Most demand load requests triggered by applications with large working set sizes miss in all cache levels.

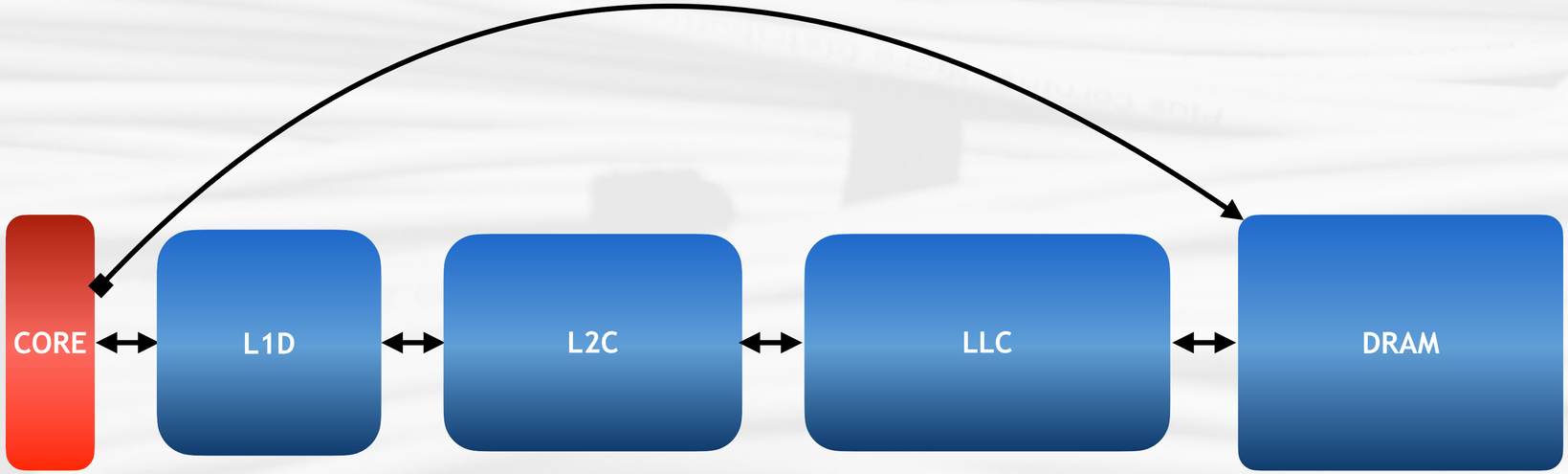
Off-Chip Prediction

Off-Chip Prediction



Off-Chip Prediction

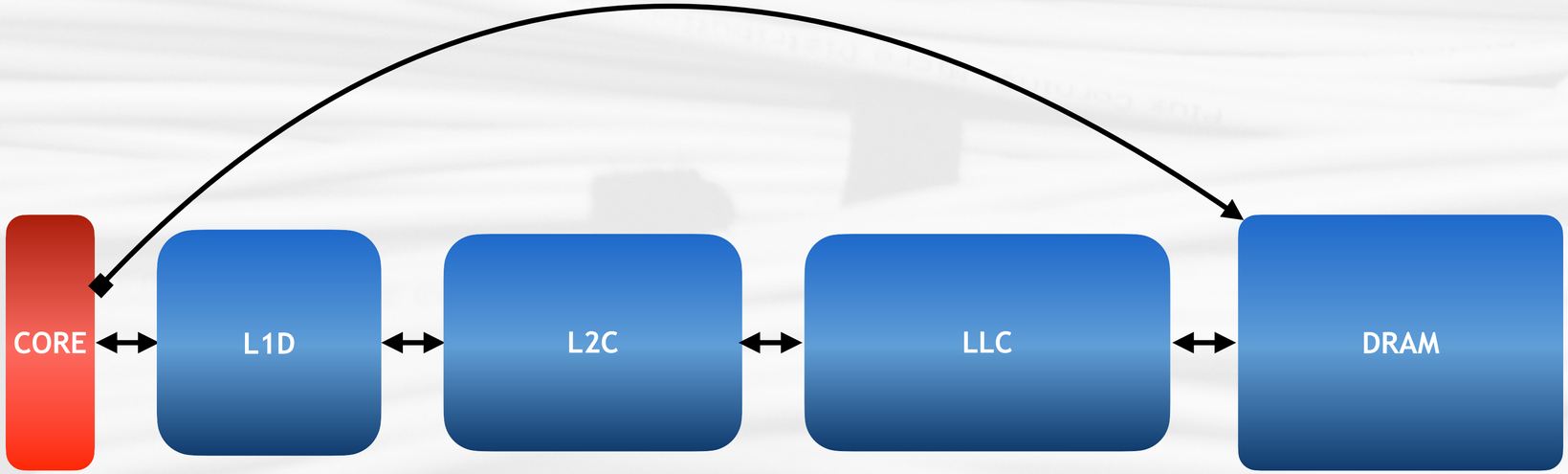
Throw a speculative request to DRAM?



- Assess whether a load demand access will be served from the caches or the DRAM

Off-Chip Prediction

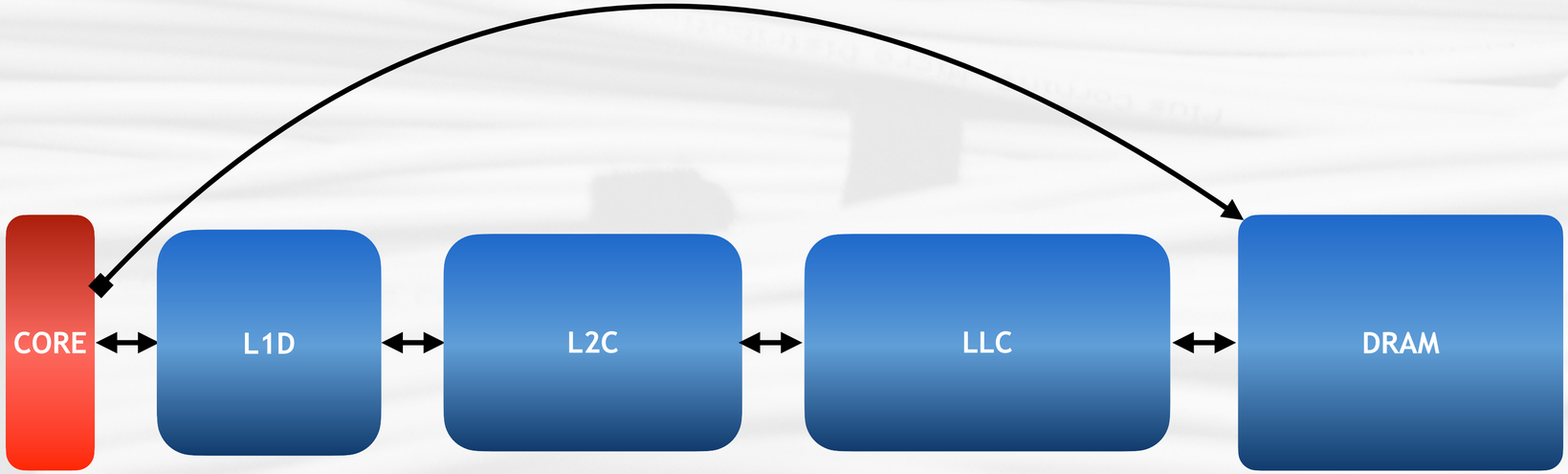
Throw a speculative request to DRAM?



- Assess whether a load demand access will be served from the caches or the DRAM
- Promising technique. No yet implemented by leading vendors

Off-Chip Prediction

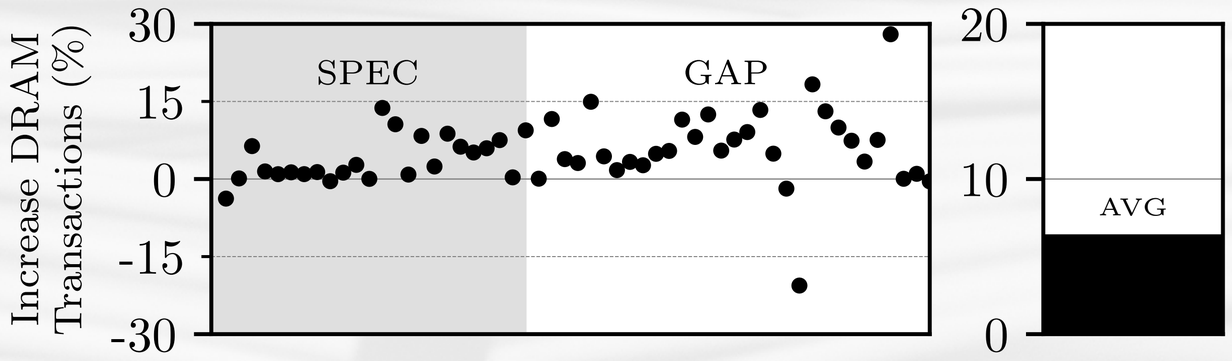
Throw a speculative request to DRAM?

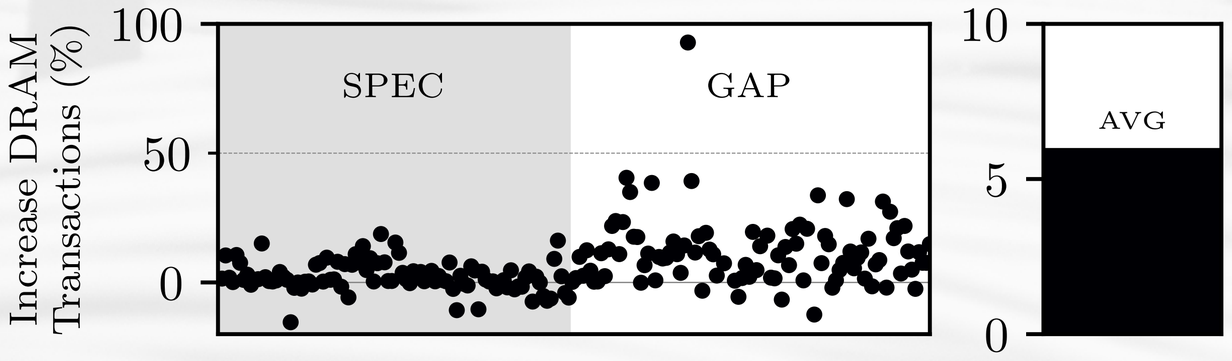


- Assess whether a load demand access will be served from the caches or the DRAM
- Promising technique. No yet implemented by leading vendors
- A state-of-the-art design is Hermes (MICRO'22)

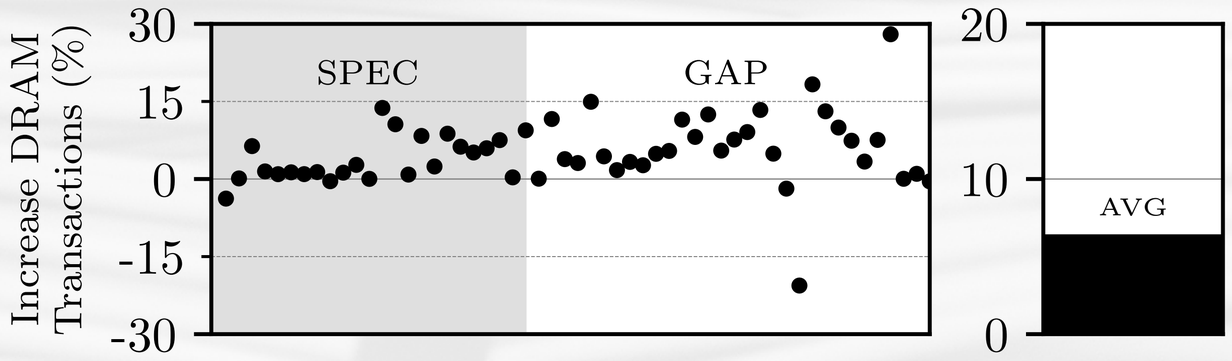
Limitations of Off-Chip Prediction

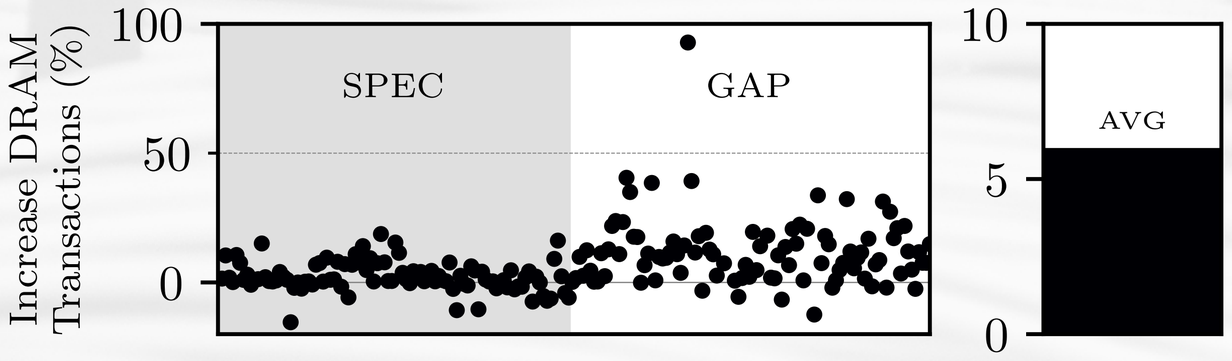
Limitations of Off-Chip Prediction





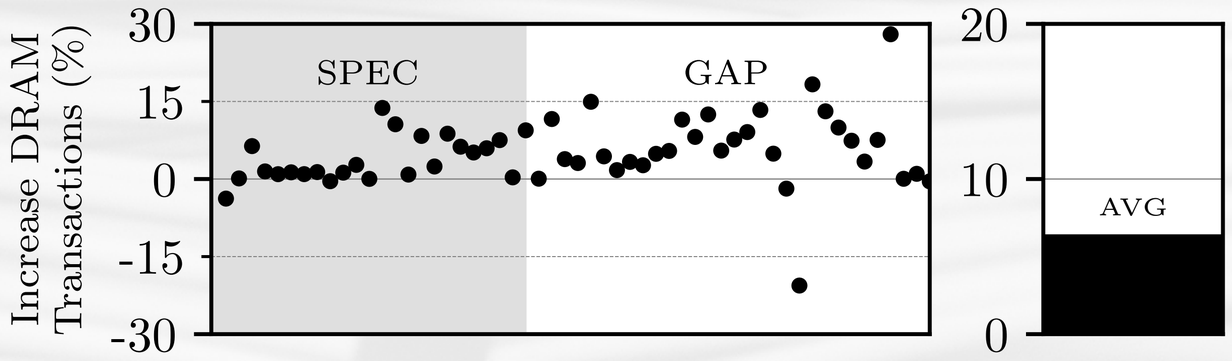
Limitations of Off-Chip Prediction

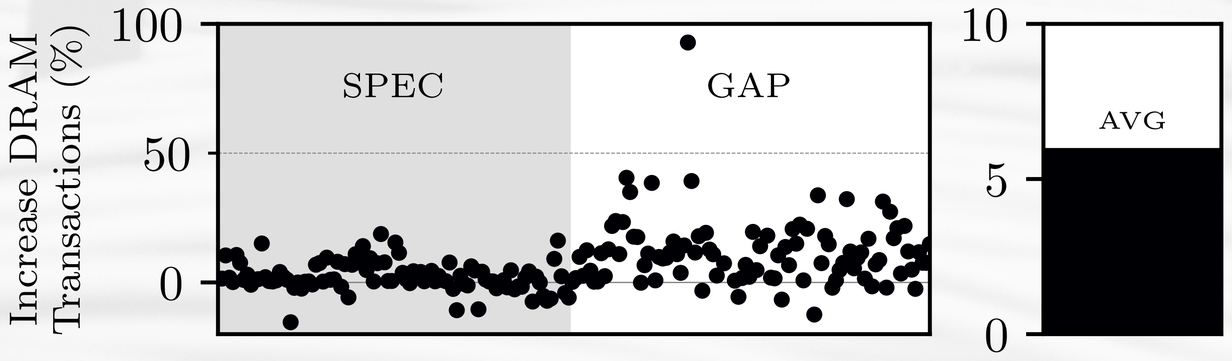




- Hermes provides performance benefits
  - Single-core —> 5.2% geomean speed-up
  - Multi-core —> 11.5% geomean speed-up

Limitations of Off-Chip Prediction

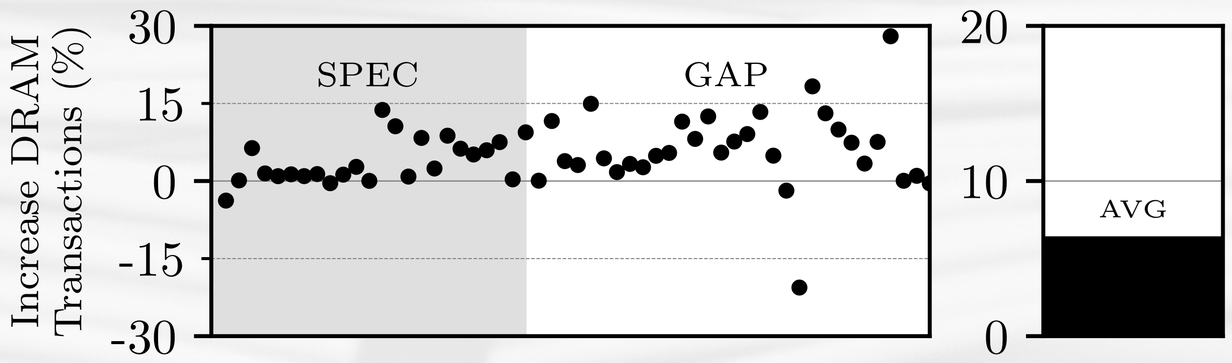


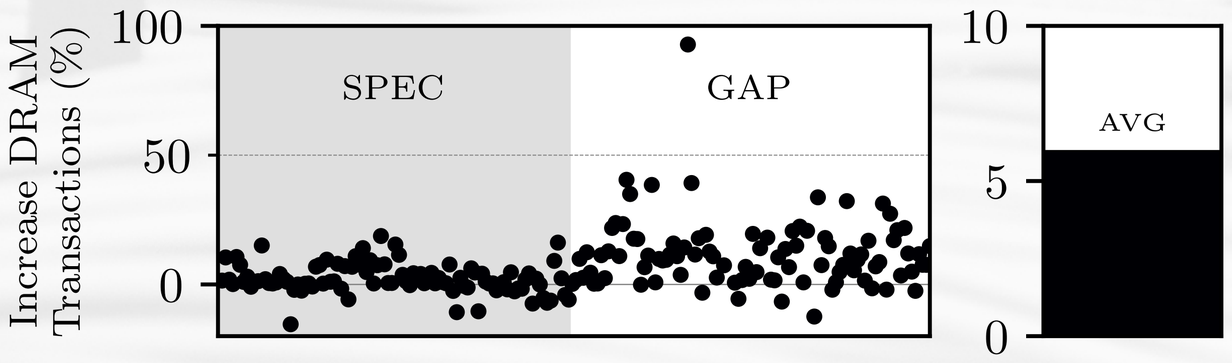


- Hermes provides performance benefits
  - Single-core —> 5.2% geomean speed-up
  - Multi-core —> 11.5% geomean speed-up

- Hermes significantly increases the # of DRAM transactions
  - Single-core —> 6.4% on average
  - Multi-core —> 6.0% on average

Limitations of Off-Chip Prediction





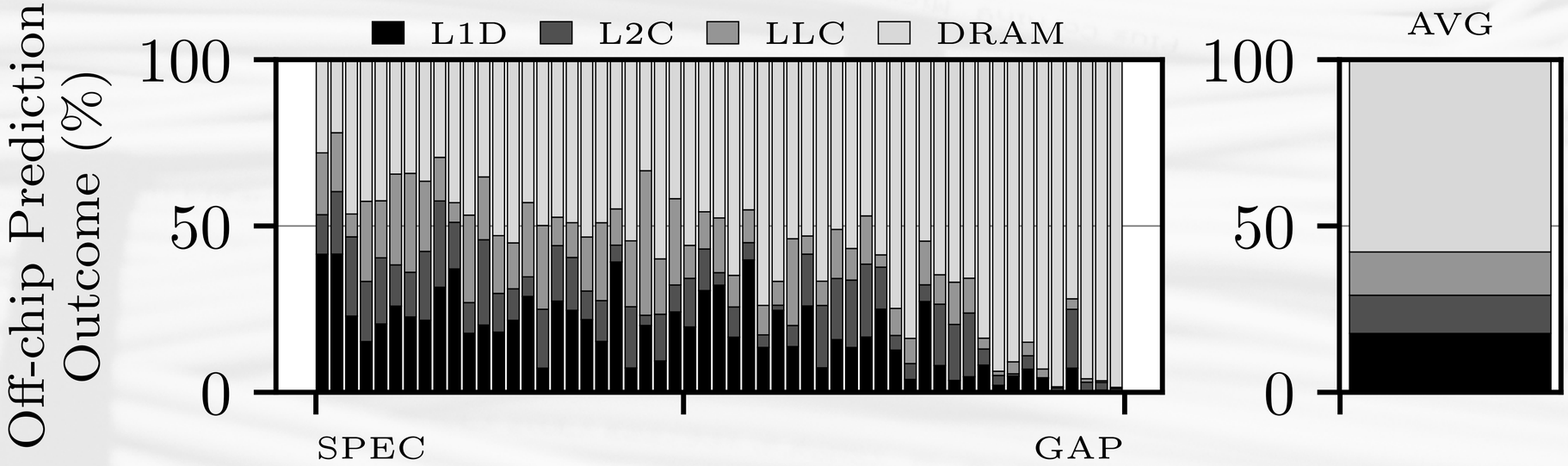
- Hermes provides performance benefits
  - Single-core —> 5.2% geomean speed-up
  - Multi-core —> 11.5% geomean speed-up

- Hermes significantly increases the # of DRAM transactions
  - Single-core —> 6.4% on average
  - Multi-core —> 6.0% on average

Hermes significantly increases DRAM accesses (both single-core and multi-core contexts).

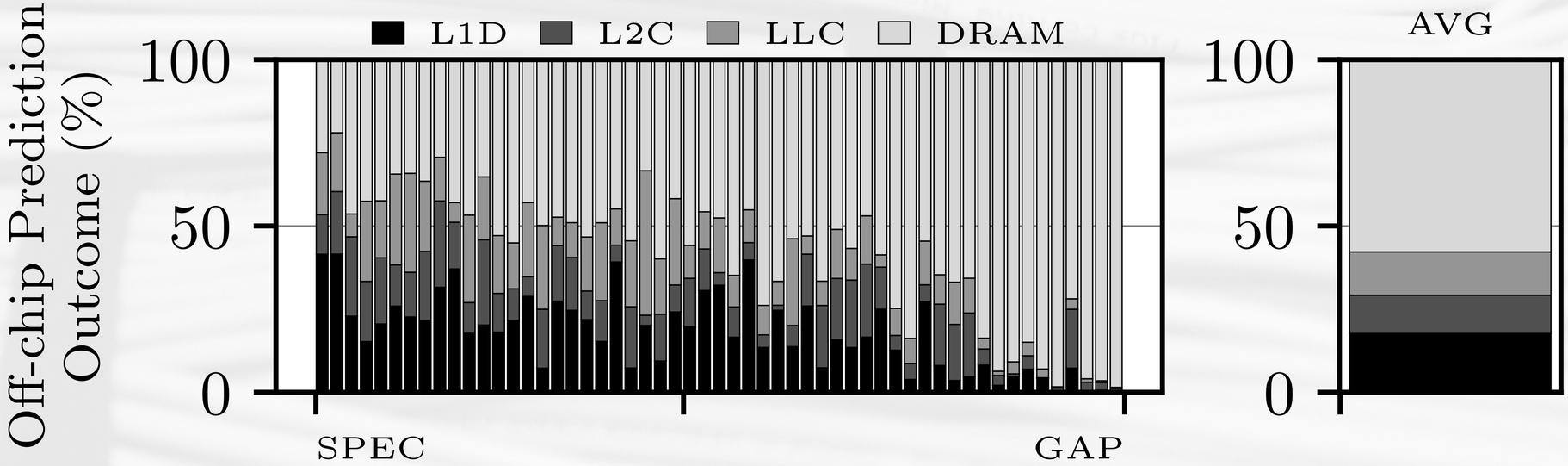
Limitations of Off-Chip Prediction

Limitations of Off-Chip Prediction



- Measuring Hermes' **inaccurate off-chip predictions**:
  - 42.2% of **off-chip** predictions are inaccurate —> hit in cache hierarchy
  - 17.7% of total off-chip predictions are found in the **L1D**.

Limitations of Off-Chip Prediction

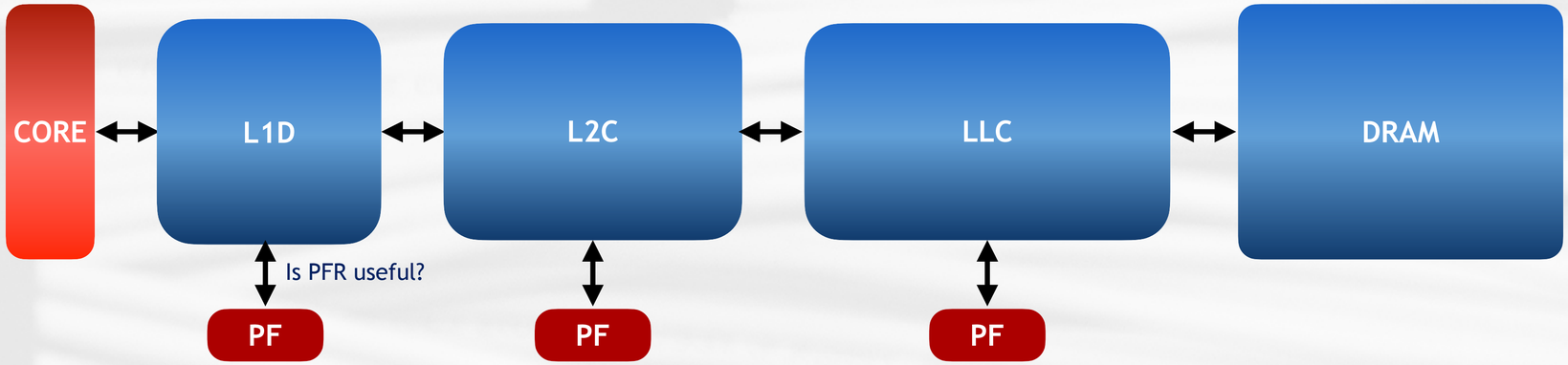


- Measuring Hermes' **inaccurate off-chip predictions**:
  - 42.2% of **off-chip** predictions are inaccurate —> hit in cache hierarchy
  - 17.7% of total off-chip predictions are found in the **L1D**.

Selectively delaying Hermes off-chip predictions until the L1D lookup is resolved has the potential to significantly reduce the number of useless DRAM transactions and deliver performance.

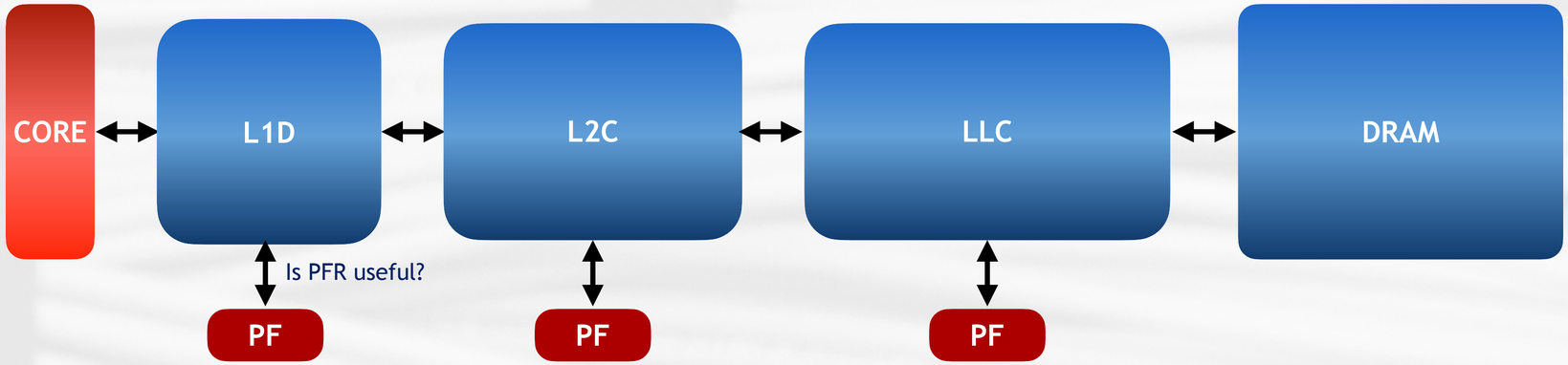
Hardware Prefetching & Prefetch Filtering

Hardware Prefetching & Prefetch Filtering



- Prefetchers proactively fetch blocks into caches before they are demanded by the core

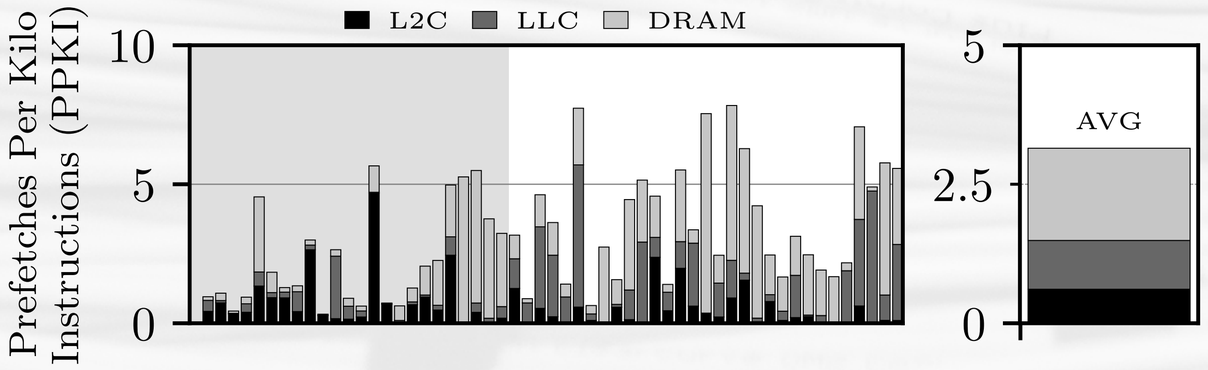
Hardware Prefetching & Prefetch Filtering

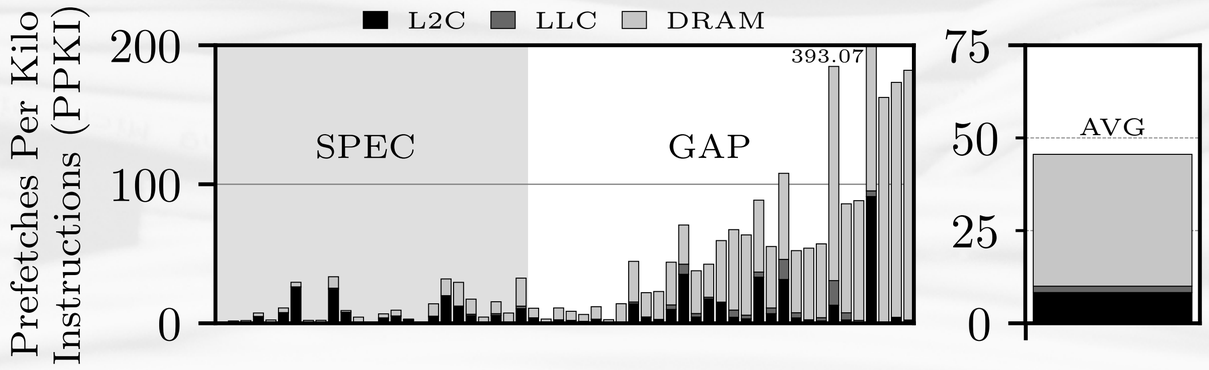


- Prefetchers proactively fetch blocks into caches before they are demanded by the core
- Not all prefetch requests are useful —> Need to filter useless prefetch requests

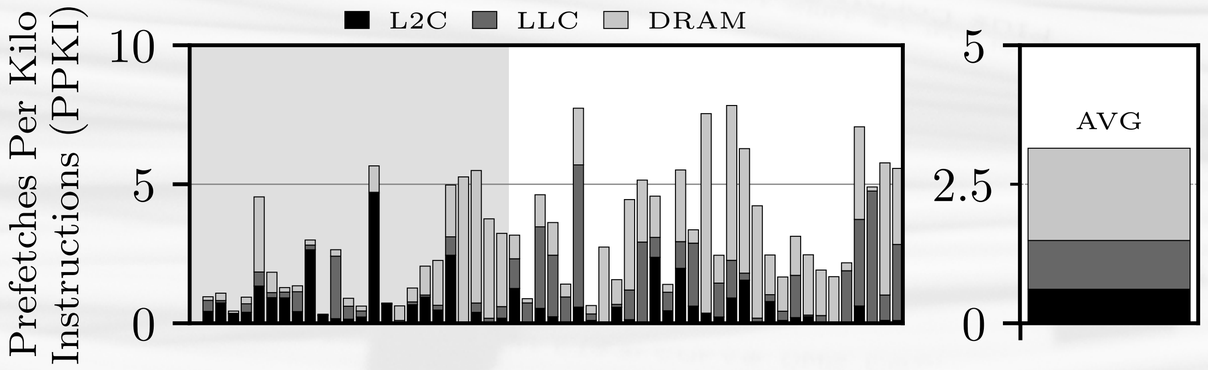
Hardware Prefetching & Prefetch Filtering

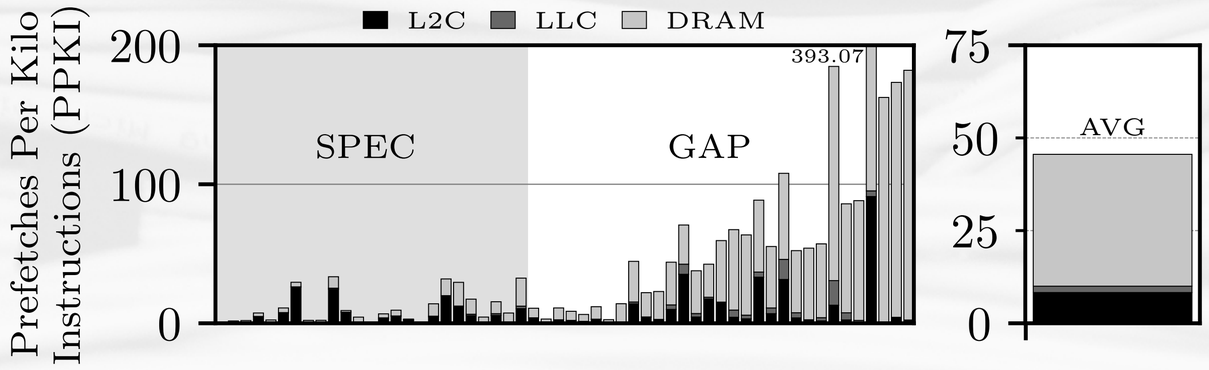
Hardware Prefetching & Prefetch Filtering





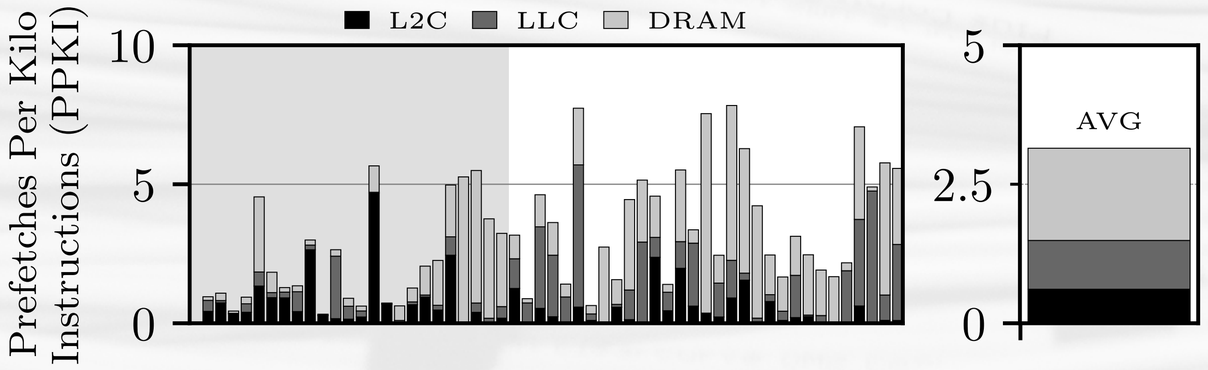
Hardware Prefetching & Prefetch Filtering

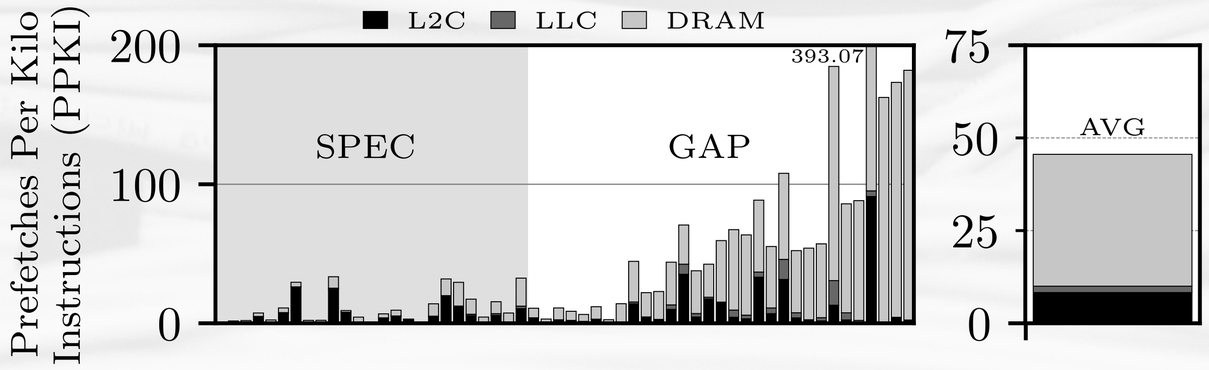




- Where are useless (not touched during their lifetime) prefetches fetched from?
  - L2C —> 18.2%
  - LLC —> 3.8%
  - DRAM —> 78%

Hardware Prefetching & Prefetch Filtering

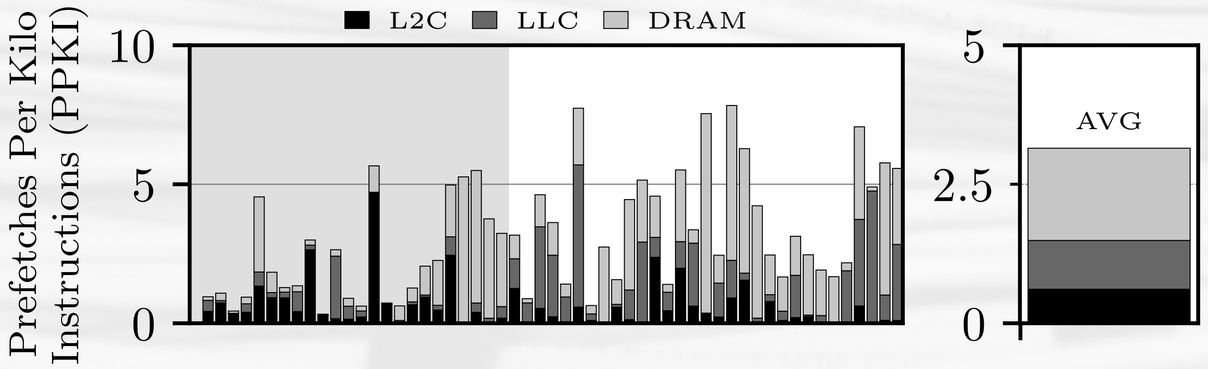


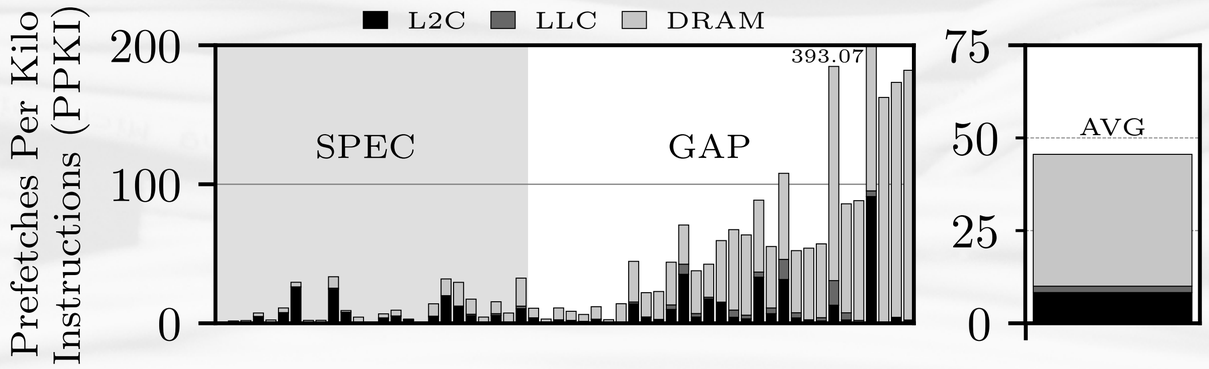


- Where are useless (not touched during their lifetime) prefetches fetched from?
  - L2C —> 18.2%
  - LLC —> 3.8%
  - DRAM —> 78%

- On average 95.2% of prefetch requests served by the DRAM are inaccurate

Hardware Prefetching & Prefetch Filtering





- Where are useless (not touched during their lifetime) prefetches fetched from?
  - L2C —> 18.2%
  - LLC —> 3.8%
  - DRAM —> 78%
- On average 95.2% of prefetch requests served by the DRAM are inaccurate

Off-chip prediction can be leveraged to design an effective prefetch filtering scheme for L1D.

TLP: A Solution to address useless DRAM Transactions

TLP: A Solution to address useless DRAM Transactions

- Selectively delaying Off-Chip predictions can reduce the # of DRAM transactions
  - Improve Hermes to selectively delay speculative DRAM accesses after L1D misses
  - We propose the *First Level Perceptron (FLP)*

TLP: A Solution to address useless DRAM Transactions

- Selectively delaying Off-Chip predictions can reduce the # of DRAM transactions
  - Improve Hermes to selectively delay speculative DRAM accesses after L1D misses
  - We propose the *First Level Perceptron (FLP)*

- Useless L1D prefetch requests are mostly fetched from DRAM
  - Off-Chip prediction can drive Prefetch Filtering
  - We propose the *Second Level Perceptron (SLP)*

TLP: A Solution to address useless DRAM Transactions

- Selectively delaying Off-Chip predictions can reduce the # of DRAM transactions
  - Improve Hermes to selectively delay speculative DRAM accesses after L1D misses
  - We propose the *First Level Perceptron (FLP)*

- Useless L1D prefetch requests are mostly fetched from DRAM
  - Off-Chip prediction can drive Prefetch Filtering
  - We propose the *Second Level Perceptron (SLP)*
- Finally, we assemble FLP and SLP into the *Two Level Perceptron (TLP)*

TLP: A Solution to address useless DRAM Transactions

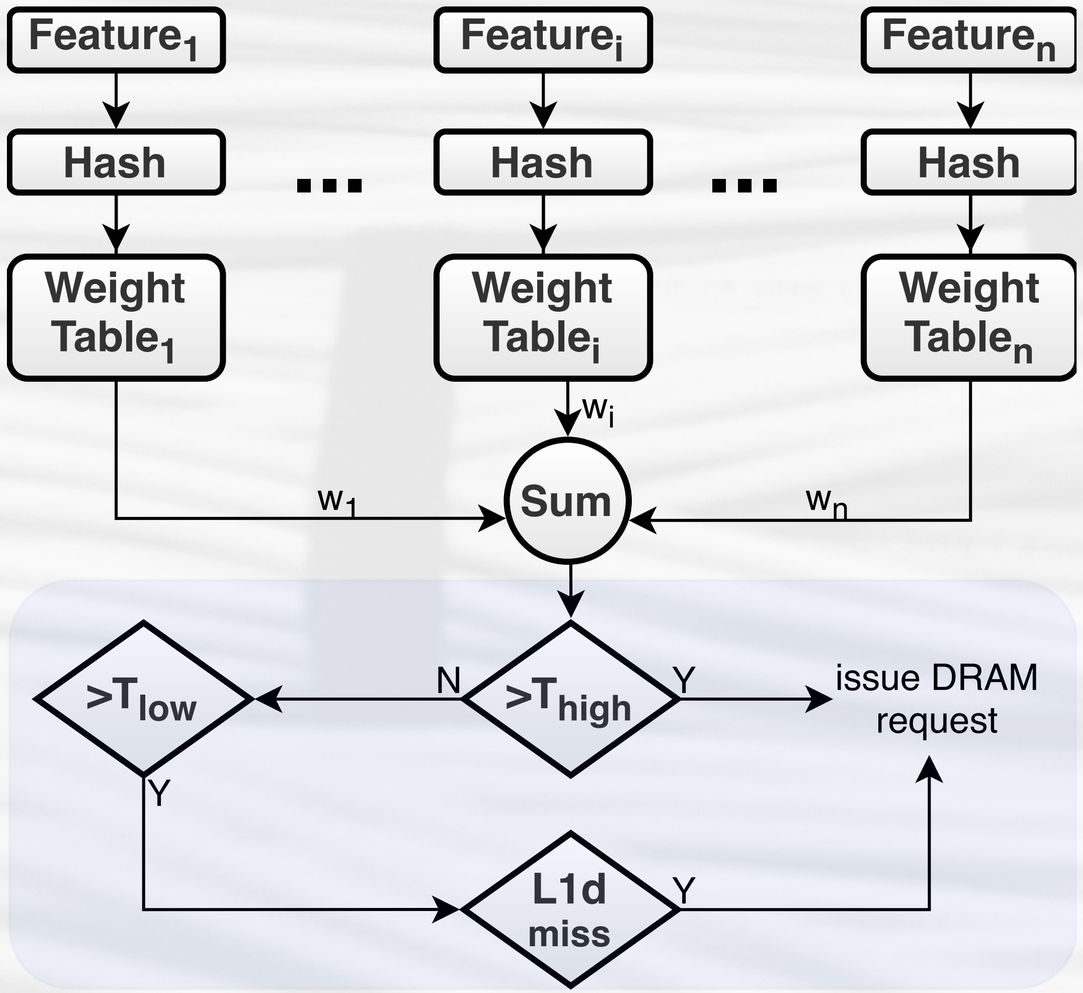
- Selectively delaying Off-Chip predictions can reduce the # of DRAM transactions
  - Improve Hermes to selectively delay speculative DRAM accesses after L1D misses
  - We propose the *First Level Perceptron (FLP)*

- Useless L1D prefetch requests are mostly fetched from DRAM
  - Off-Chip prediction can drive Prefetch Filtering
  - We propose the *Second Level Perceptron (SLP)*
- Finally, we assemble FLP and SLP into the *Two Level Perceptron (TLP)*

We assemble FLP and SLP into the *Two Level Perceptron (TLP)*.

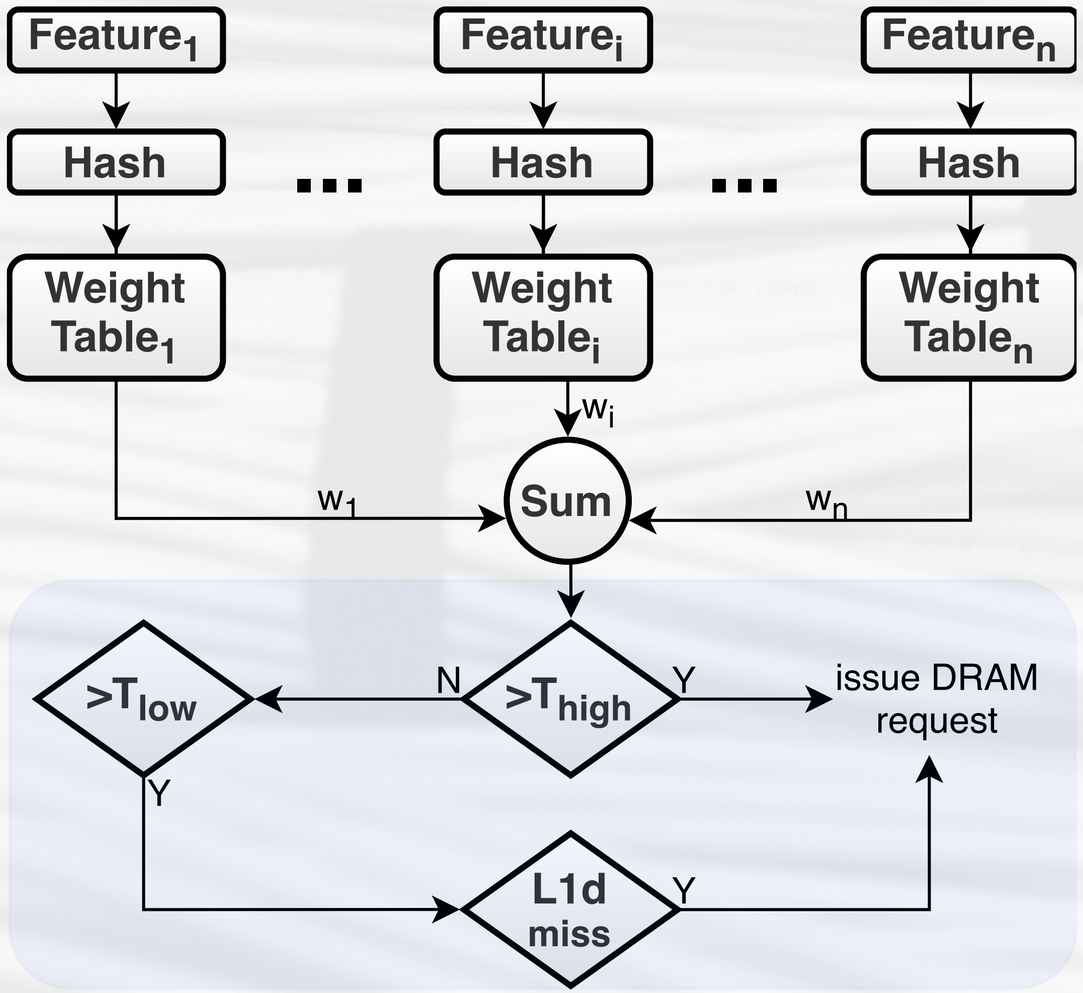
FLP: Leveraging Off-Chip Prediction for Prefetch Filtering

FLP: Leveraging Off-Chip Prediction for Prefetch Filtering



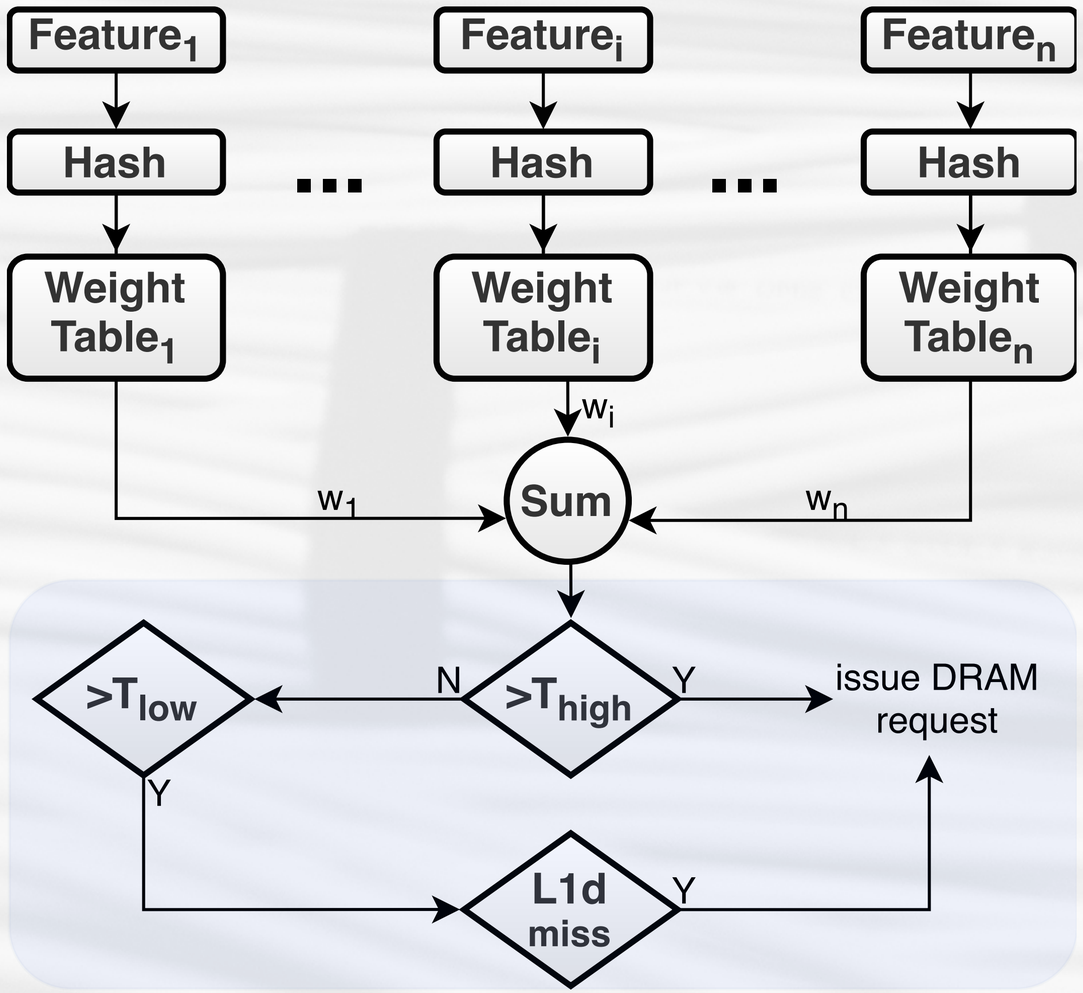
- FLP is a hashed perceptron predictor

FLP: Leveraging Off-Chip Prediction for Prefetch Filtering



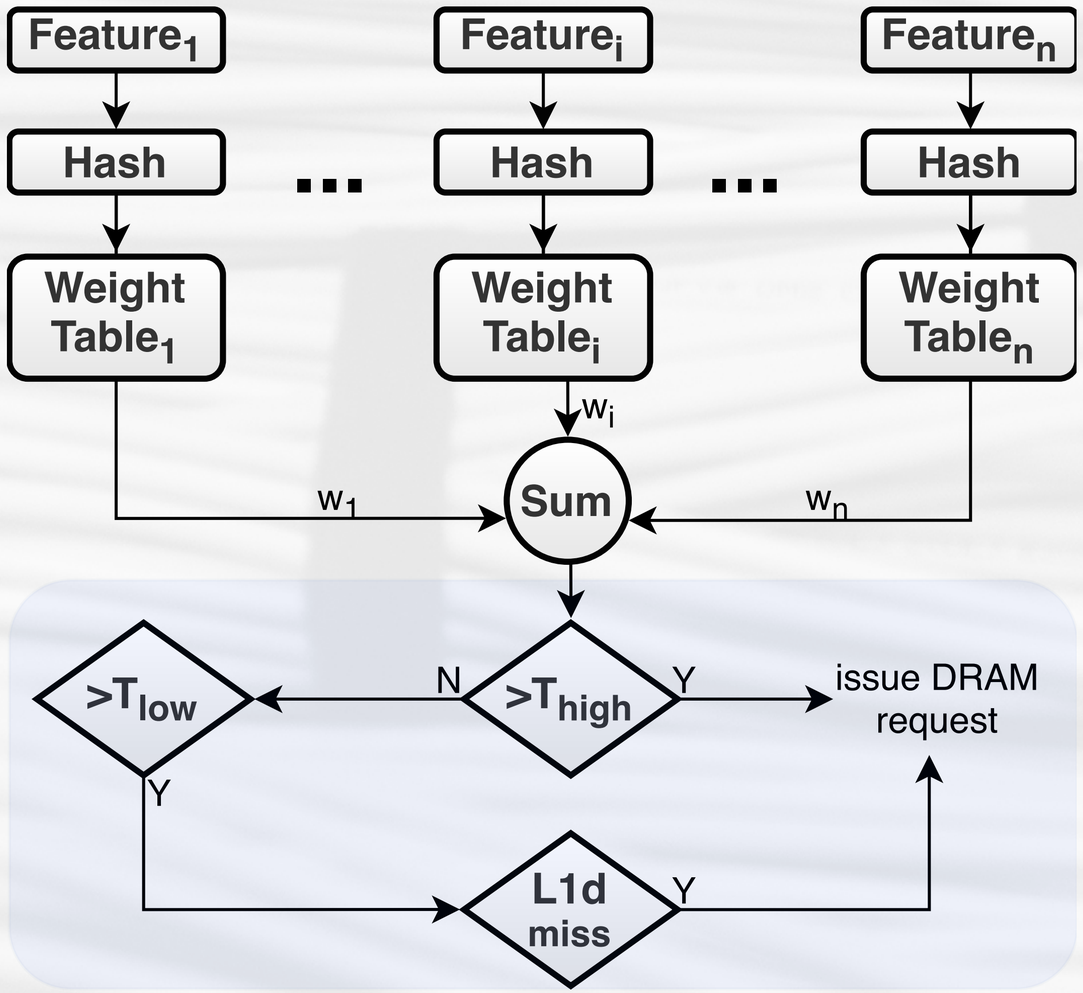
- FLP is a hashed perceptron predictor
- Utilizes several μ-arch features (history of load PCs, etc.)
  - Components (5 prediction tables, 128-1K entries, history of the last n-load PCs, virtual address bits, etc.)

FLP: Leveraging Off-Chip Prediction for Prefetch Filtering



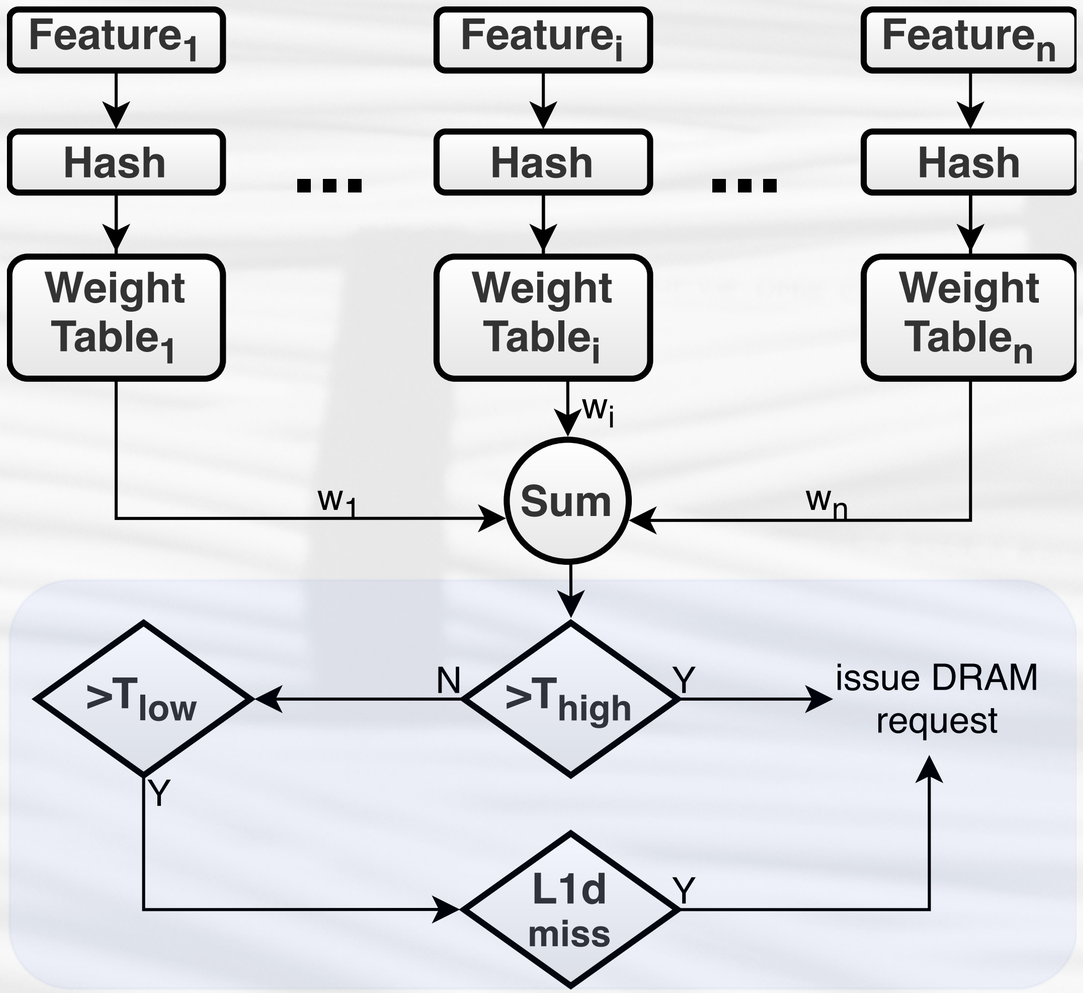
- FLP is a hashed perceptron predictor
- Utilizes several μ-arch features (history of load PCs, etc.)
  - Components (5 prediction tables, 128-1K entries, history of the last n-load PCs, virtual address bits, etc.)
- FLP is consulted upon a load demand request

FLP: Leveraging Off-Chip Prediction for Prefetch Filtering



- FLP is a hashed perceptron predictor
- Utilizes several μ-arch features (history of load PCs, etc.)
  - Components (5 prediction tables, 128-1K entries, history of the last n-load PCs, virtual address bits, etc.)
- FLP is consulted upon a load demand request
- Prediction outcomes:
  1. On-Chip —> No DRAM prefetch
  2. Off-Chip —> DRAM prefetch from the core
  3. **Delayed Prediction** —> Delays DRAM prefetch after L1D lookup

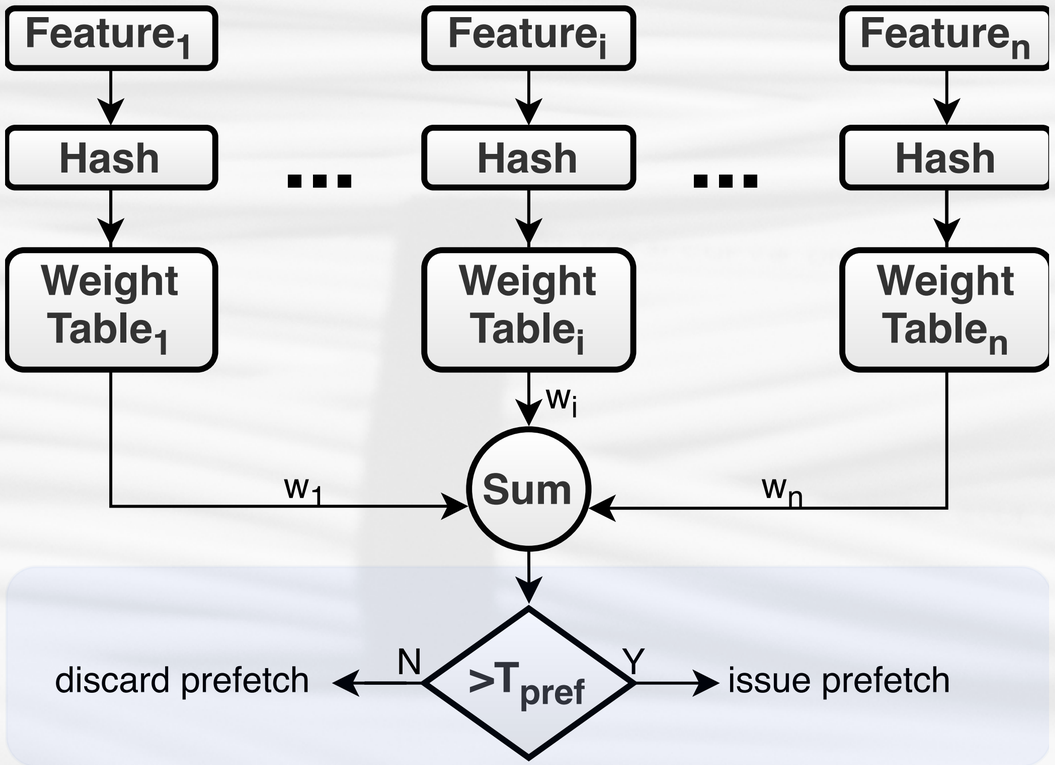
FLP: Leveraging Off-Chip Prediction for Prefetch Filtering



- FLP is a hashed perceptron predictor
- Utilizes several μ-arch features (history of load PCs, etc.)
  - Components (5 prediction tables, 128-1K entries, history of the last n-load PCs, virtual address bits, etc.)
- FLP is consulted upon a load demand request
- Prediction outcomes:
  1. On-Chip —> No DRAM prefetch
  2. Off-Chip —> DRAM prefetch from the core
  3. **Delayed Prediction** —> Delays DRAM prefetch after L1D lookup
- FLP is trained when a request **returns** to the core

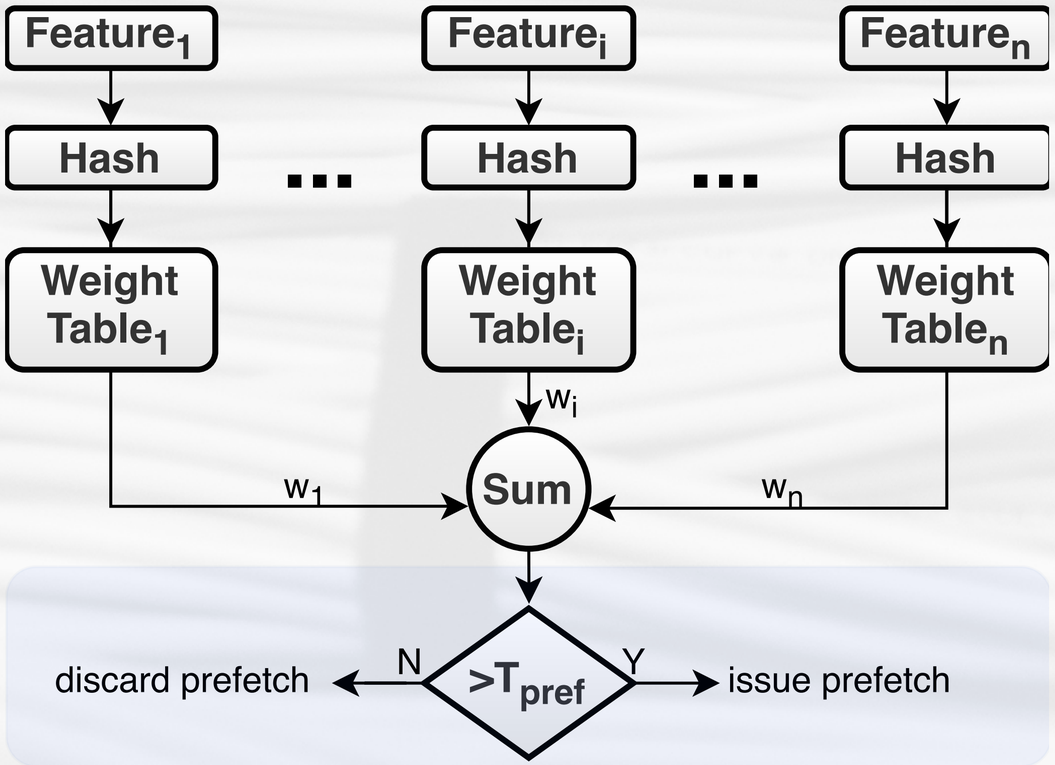
SLP: An Approach to Off-Chip Prediction-based Prefetch Filtering

SLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



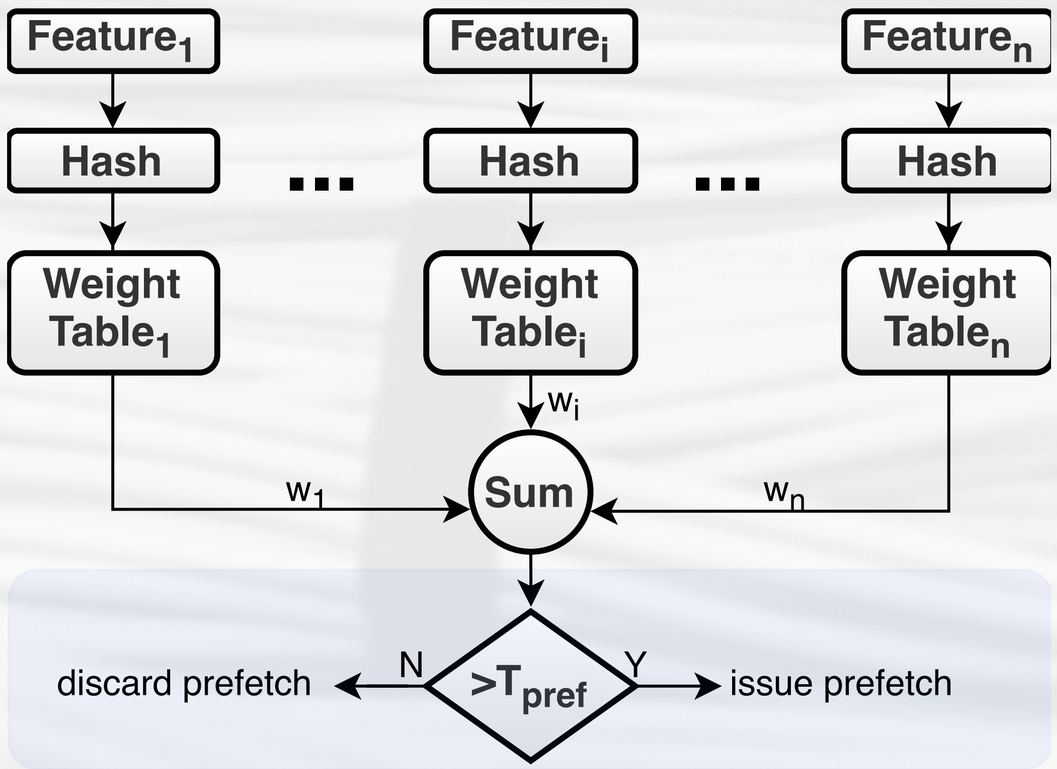
- SLP is a hashed perceptron predictor

SLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



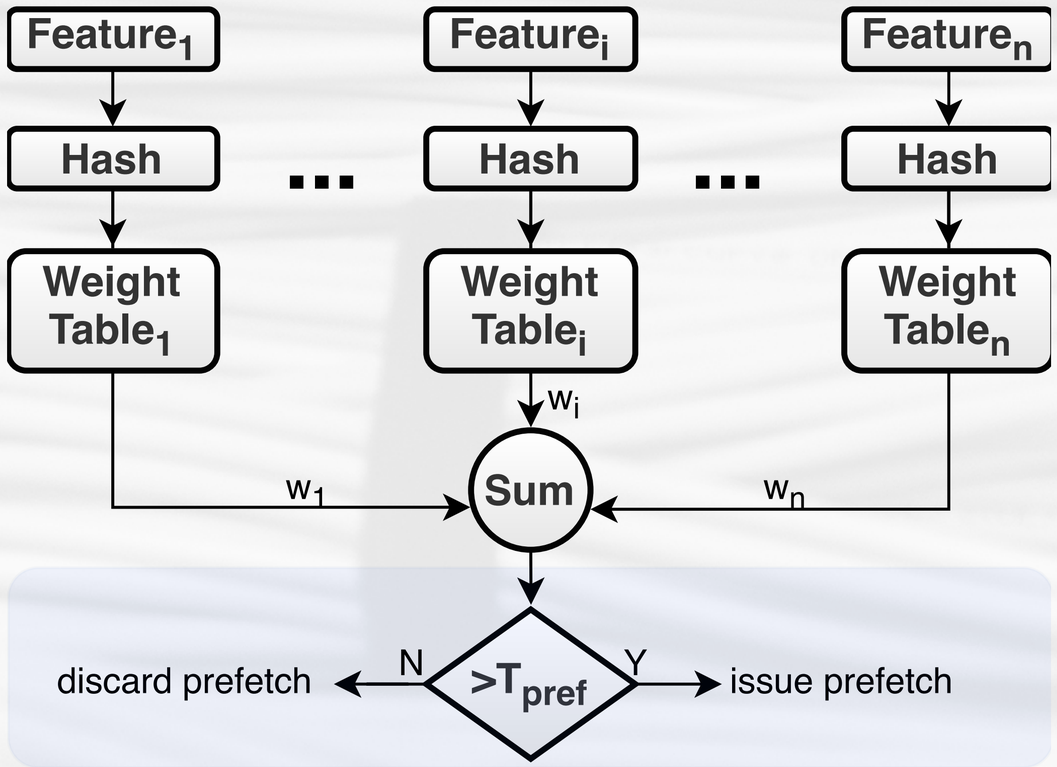
- SLP is a hashed perceptron predictor
- Utilizes several μ-arch features (history of load PCs, etc.)
  - Components (6 prediction tables, 128-1K entries, history of the last n-load PCs, physical address bits, output of FLP, etc.)

SLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



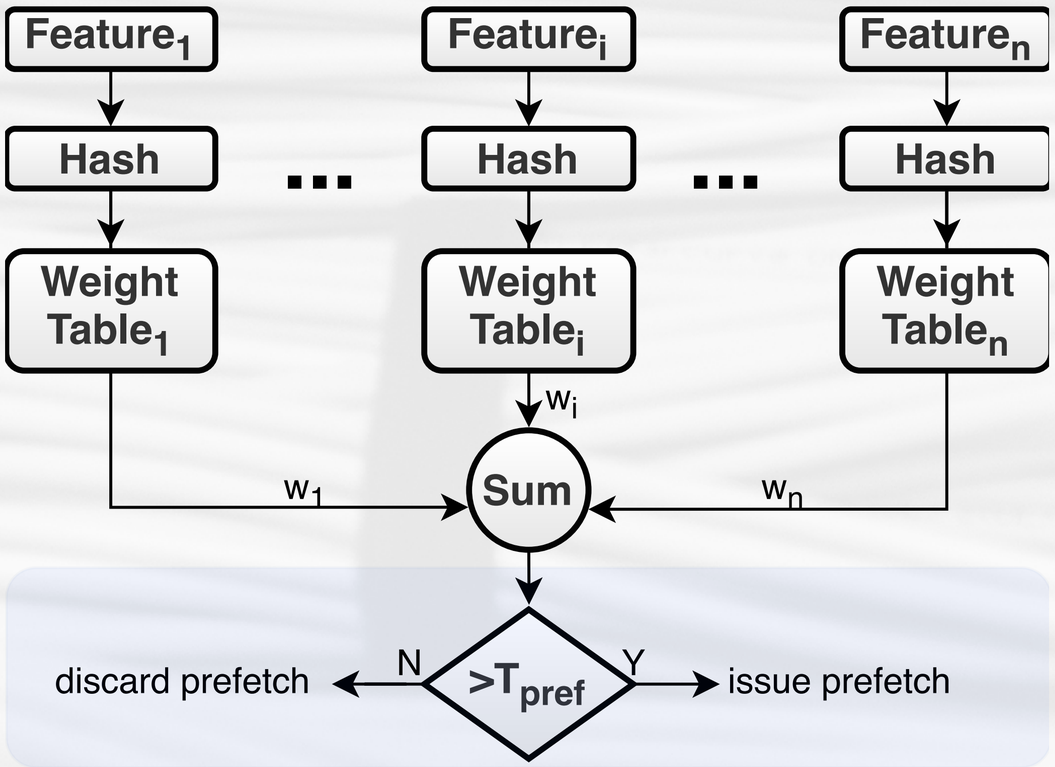
- SLP is a hashed perceptron predictor
- Utilizes several μ-arch features (history of load PCs, etc.)
  - Components (6 prediction tables, 128-1K entries, history of the last n-load PCs, physical address bits, output of FLP, etc.)
- SLP is consulted upon a L1D prefetch request emission

SLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



- SLP is a hashed perceptron predictor
- Utilizes several μ-arch features (history of load PCs, etc.)
  - Components (6 prediction tables, 128-1K entries, history of the last n-load PCs, physical address bits, output of FLP, etc.)
- SLP is consulted upon a L1D prefetch request emission
- Prediction outcomes:
  1. On-Chip —> the prefetch request is issued
  2. Off-Chip —> the prefetch request is discarded

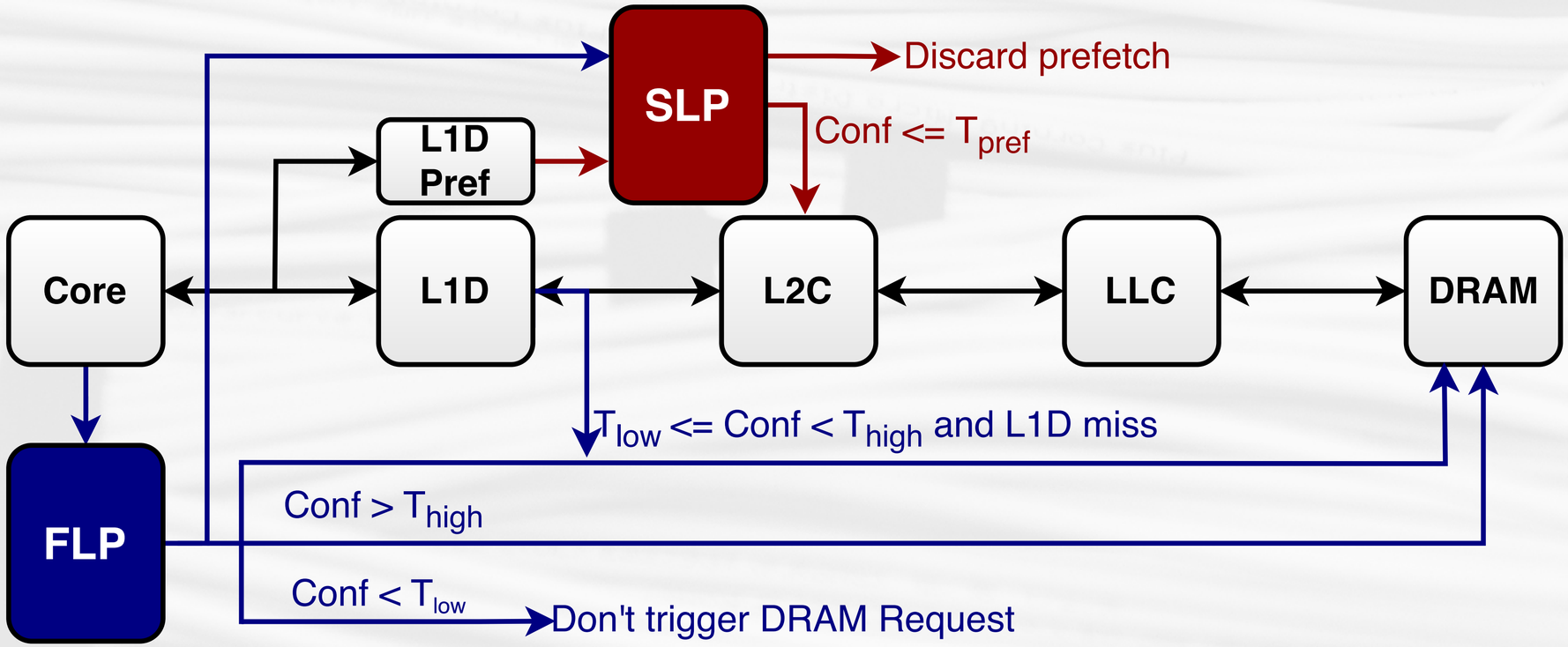
SLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



- SLP is a hashed perceptron predictor
- Utilizes several μ-arch features (history of load PCs, etc.)
  - Components (6 prediction tables, 128-1K entries, history of the last n-load PCs, physical address bits, output of FLP, etc.)
- SLP is consulted upon a L1D prefetch request emission
- Prediction outcomes:
  1. On-Chip —> the prefetch request is issued
  2. Off-Chip —> the prefetch request is discarded
- Similarly to FLP, SLP is trained upon the **completion** of an L1D prefetch request.

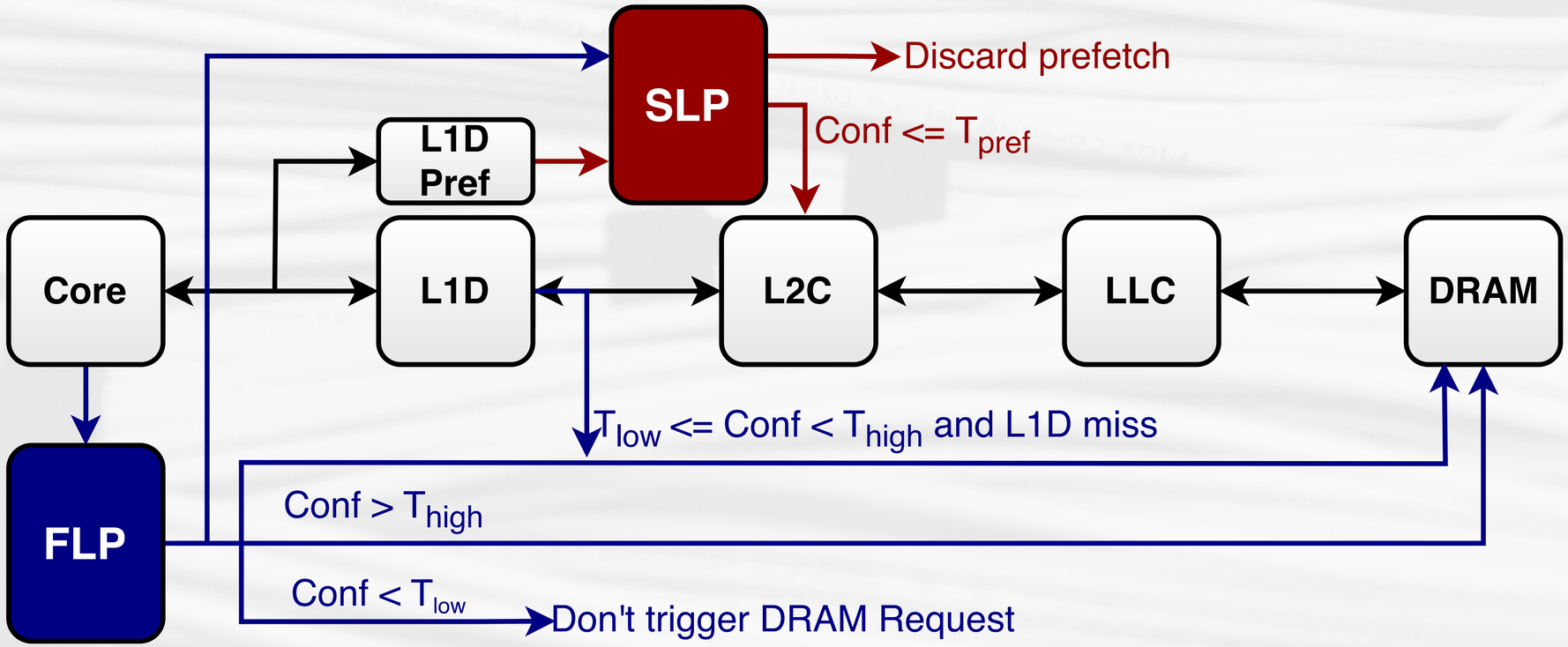
TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering

TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



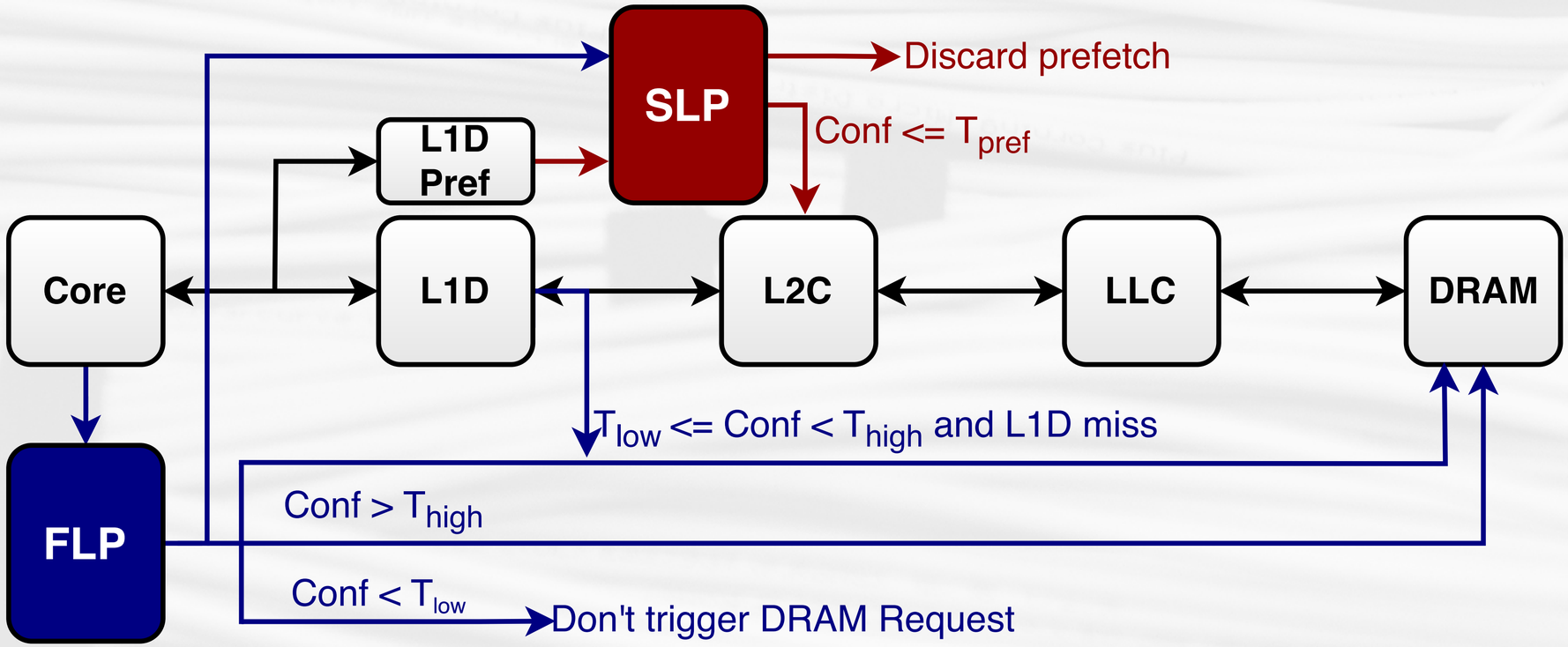
- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*

TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

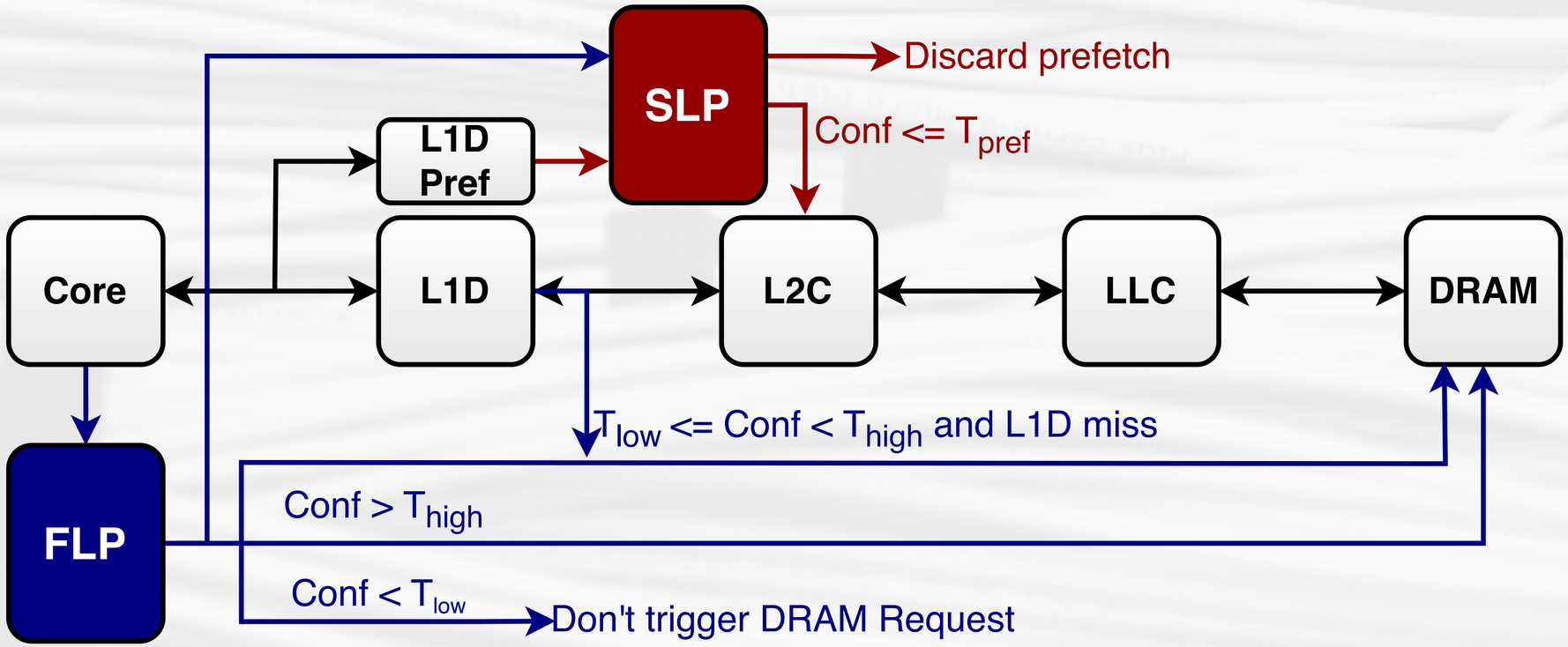
TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



❶ FLP provides confidence on a load going off-chip

- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering

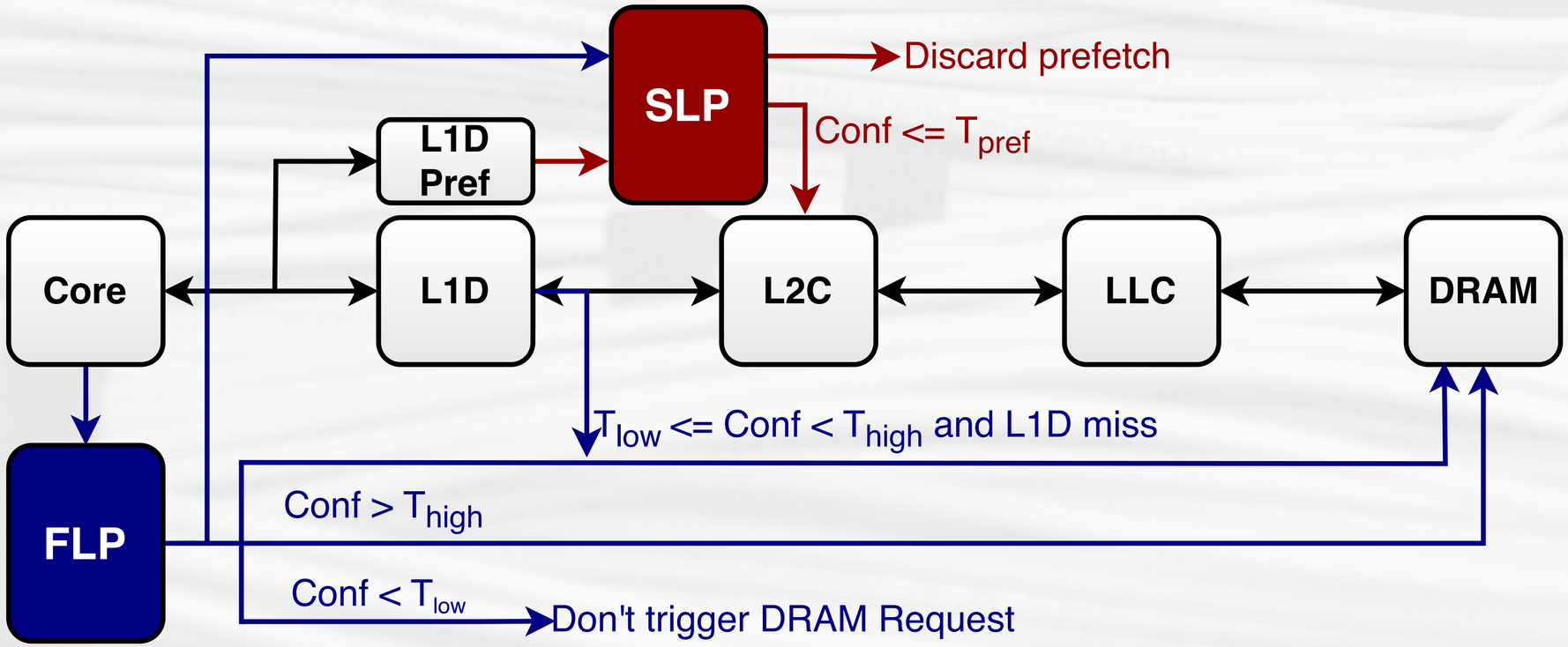


- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

❶ FLP provides confidence on a load going off-chip

- High confidence triggers speculative DRAM request ❷

TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering

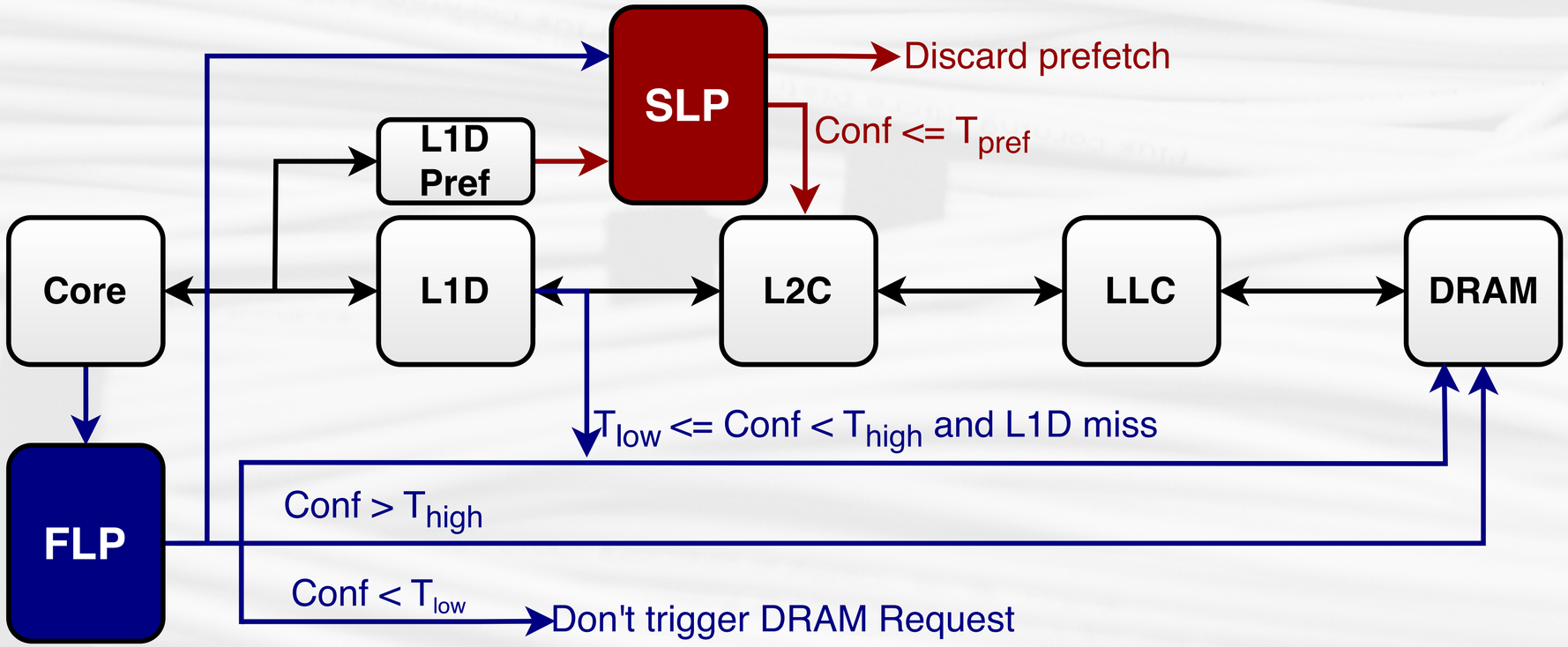


- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

❶ FLP provides confidence on a load going off-chip

- High confidence triggers speculative DRAM request ❷
- Low confidence delays DRAM request until L1D miss ❸

TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering

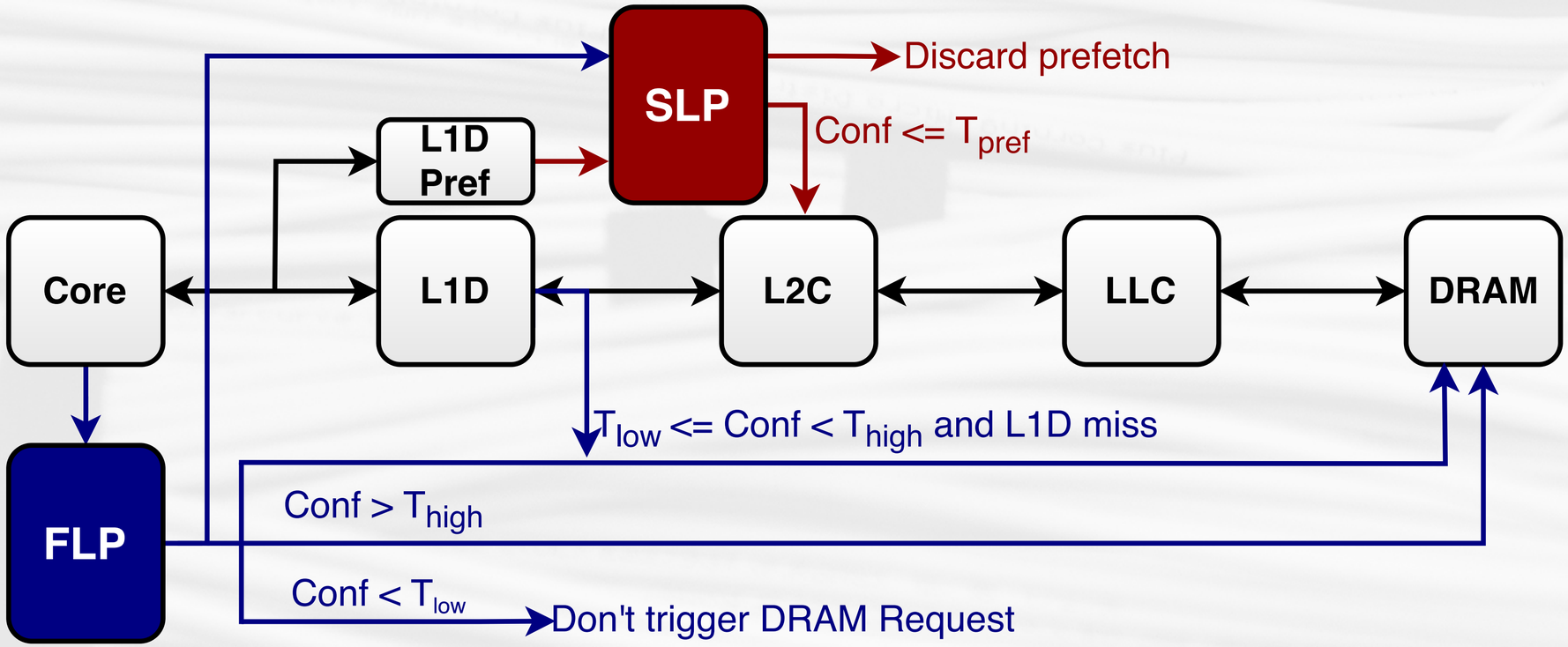


- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

❶ FLP provides confidence on a load going off-chip

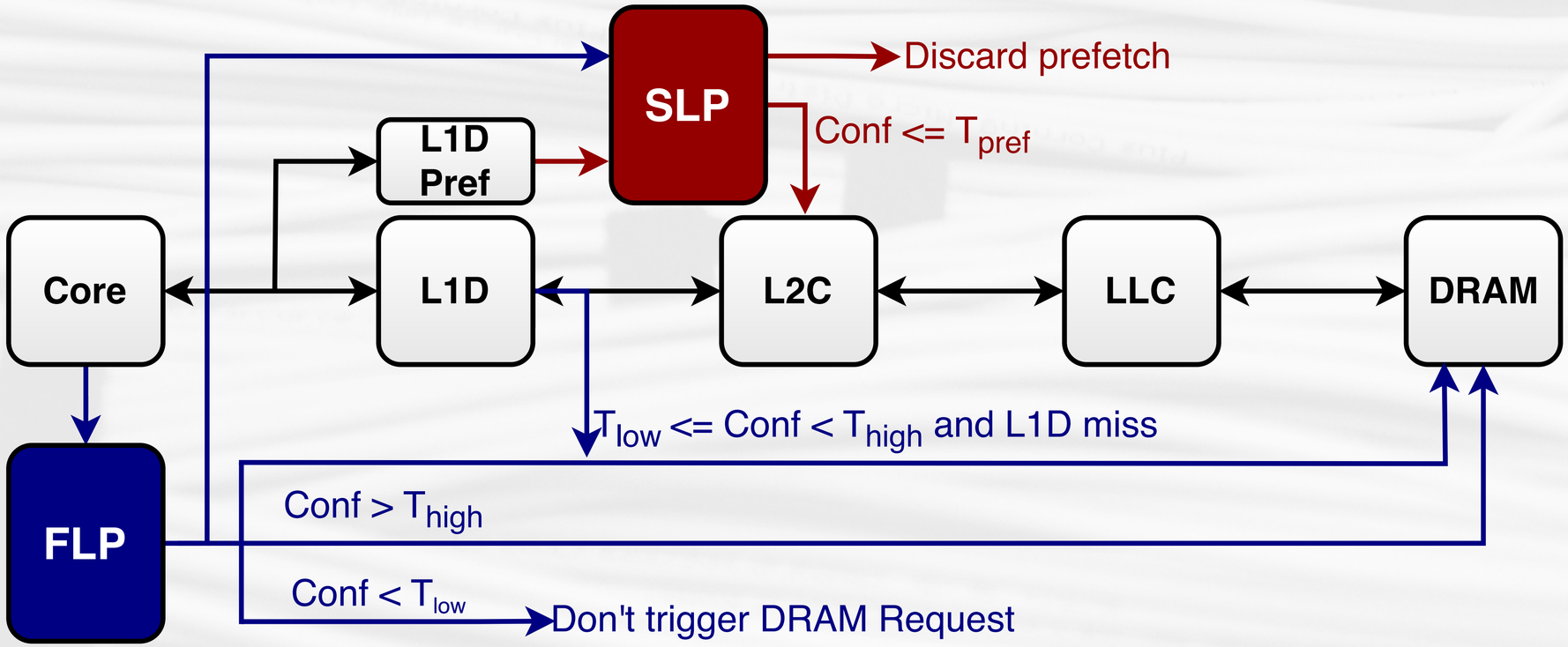
- High confidence triggers speculative DRAM request ❷
- Low confidence delays DRAM request until L1D miss ❸
- If predicted to be on-chip, no action taken ❹

TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

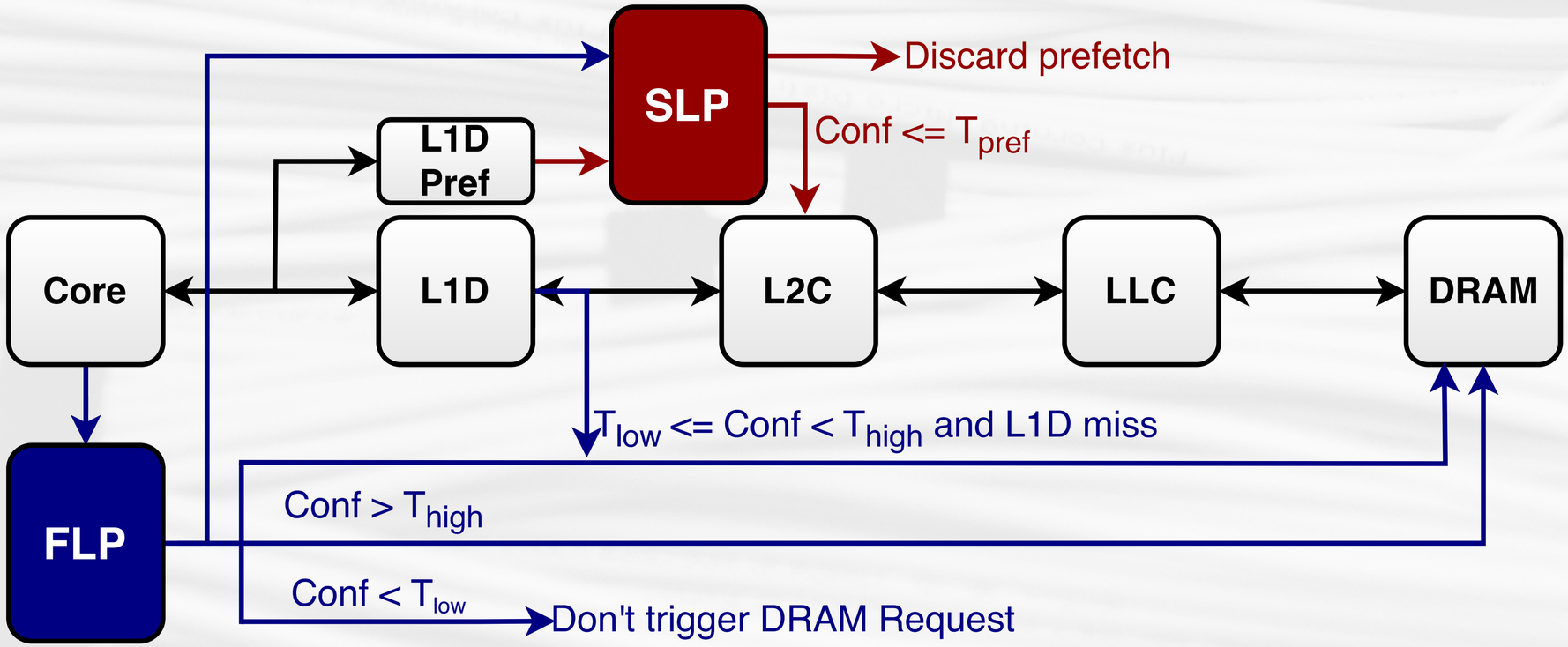
TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



SLP is consulted upon L1D prefetch requests ❺

- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering

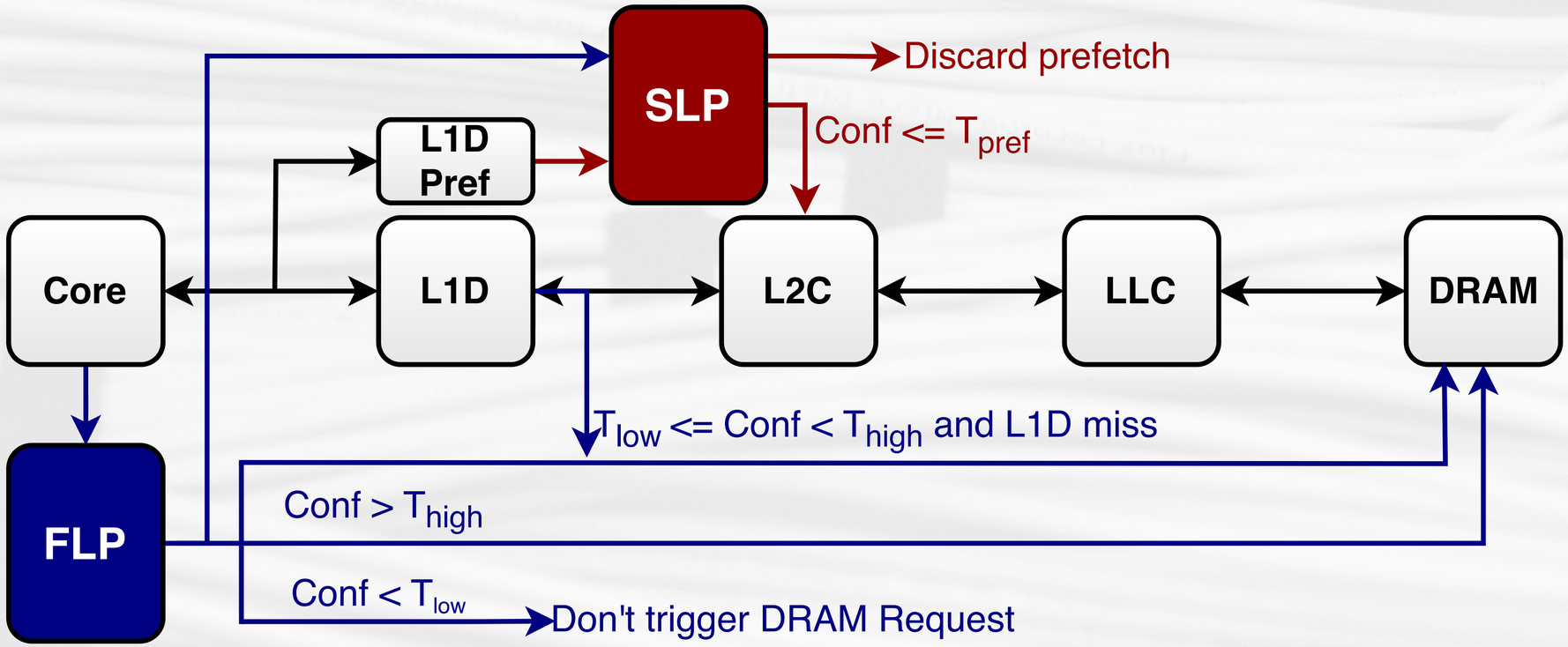


- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

SLP is consulted upon L1D prefetch requests ❺

- If the request is predicted to be off-chip ❻, the request is discarded

TLP: An Approach to Off-Chip Prediction-based Prefetch Filtering



- The *Two Level Perceptron (TLP)* Predictor combines the *FLP* and the *SLP*
- It allows a layered perceptron design
  - SLP uses the output of the FLP as an input

SLP is consulted upon L1D prefetch requests ❺

- If the request is predicted to be off-chip ❻, the request is discarded
- If the request is predicted to be on-chip ❼, the request continues as usual

Experimental Setup

- ChampSim simulator (GitHub)
- **Cascade Lake** micro-architecture (specs)
- Workloads
  - SPEC CPU 2006 & 2017 (24 workloads)
  - GAP Benchmark Suite (31 workloads)
  - SimPoints methodology
- **Single**-core evaluation
- **Multi**-core evaluation

Component	Parameters	Latency
ITLB	64-entry, 4-way	1cc
DTLB	64-entry, 4-way	1cc
STLB	1536-entry, 12-way	8cc
L1 I-Cache	32KB, 8-way	4cc
L1 D-Cache	32KB, 8-way, IPCP/Berti Prefetcher	4cc
L2 Cache	1MB, 16-way, SPP Prefetcher	10cc
LLC	1.375MB/core, 11-way	20cc
DRAM	16GB, tRP=tRCD=tCAS=24cc	variable

Alternative Techniques

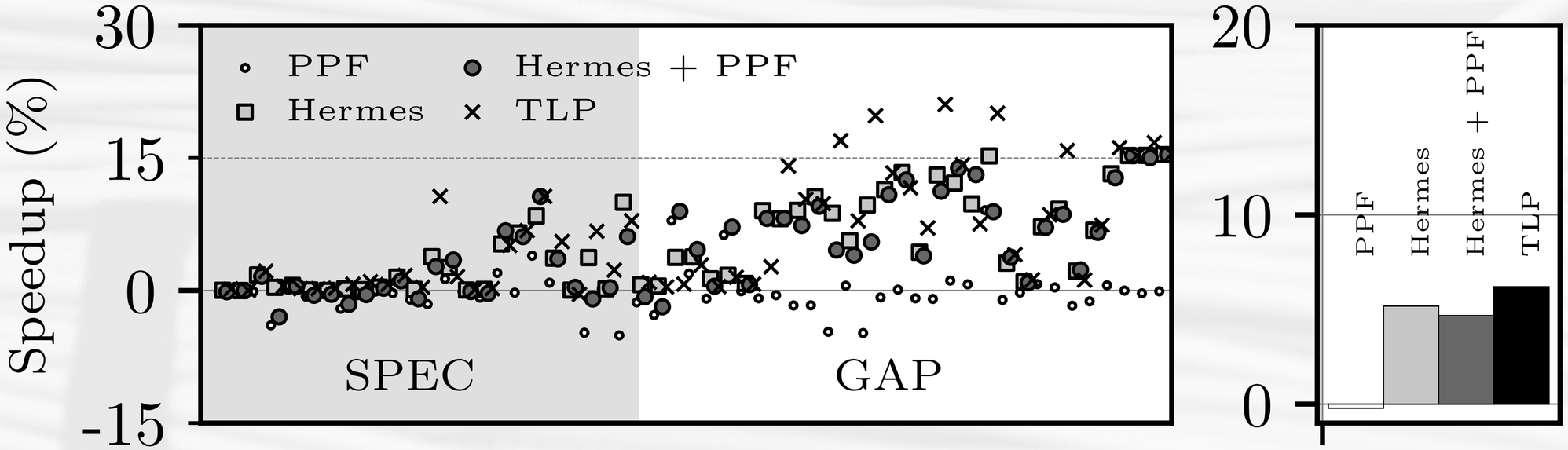
- Designs evaluated:
  - PPF (ISCA'19)
  - Hermes (MICRO'22)
  - Hermes+PPF
  - TLP (our design)
- Considered L1D Prefetchers:
  - IPCP (ISCA'20)
  - Berti (MICRO'22)

Alternative Techniques

- Designs evaluated:
  - PPF (ISCA'19)
  - Hermes (MICRO'22)
  - Hermes+PPF
  - TLP (our design)
- Considered L1D Prefetchers:
  - IPCP (ISCA'20)
  - Berti (MICRO'22)

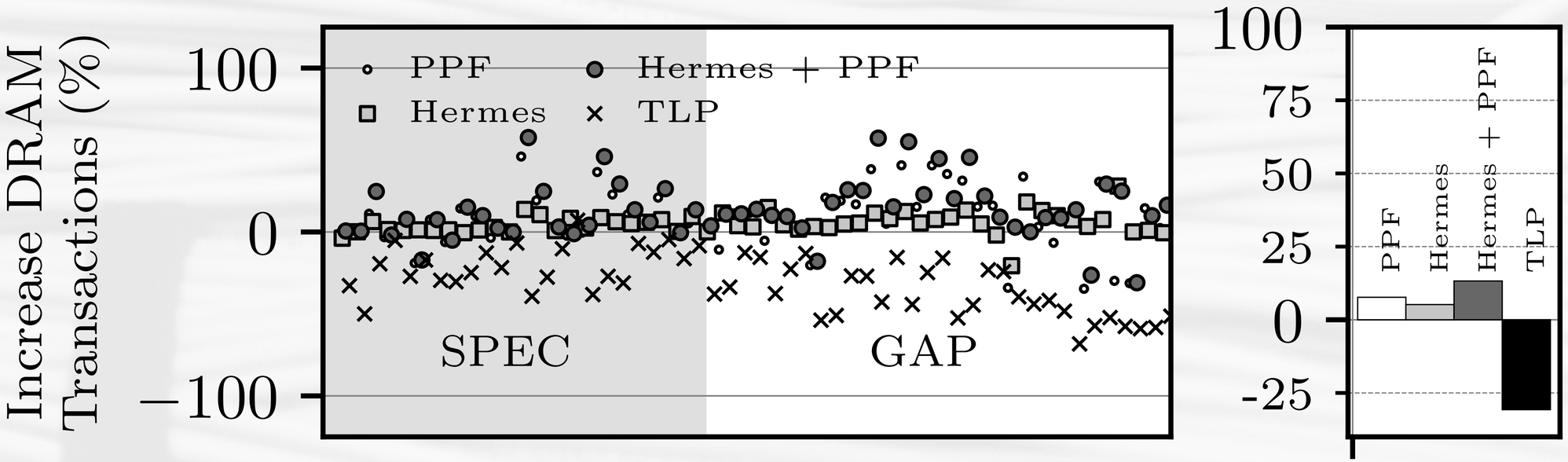
Hermes+PPF, like **TLP** provides Off-Chip Prediction and Adaptive Prefetch Filtering.

Single-Core Evaluation | Performance



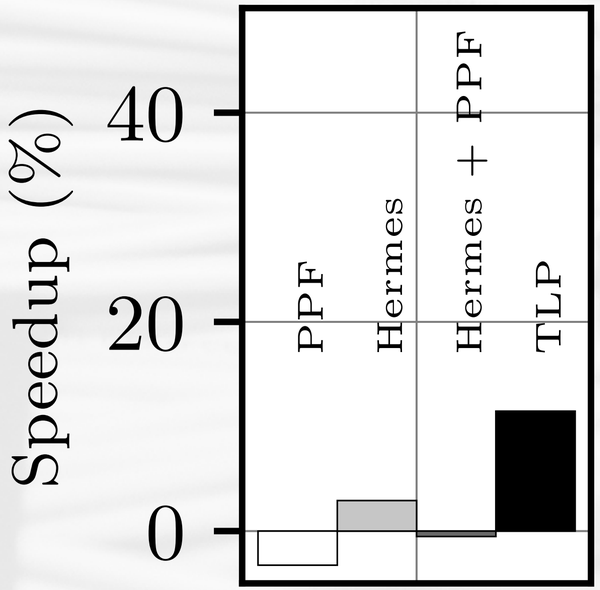
- Considering a single-core scenario, TLP outperforms all considered designs
  - **n.b.**, The single-core scenario assumes 12.8 GB/s per core of DRAM bandwidth
  - **n.b.**, We evaluated TLP over a range of DRAM bandwidth allocation. We observe that as bandwidth decreases, TLP's potential grows higher
  - **TLP** leverages **6.2%** geomean speed-up as compared to **-0.2%**, **5.2%**, and **4.7%** for **PPF**, **Hermes**, and **Hermes+PPF**, respectively

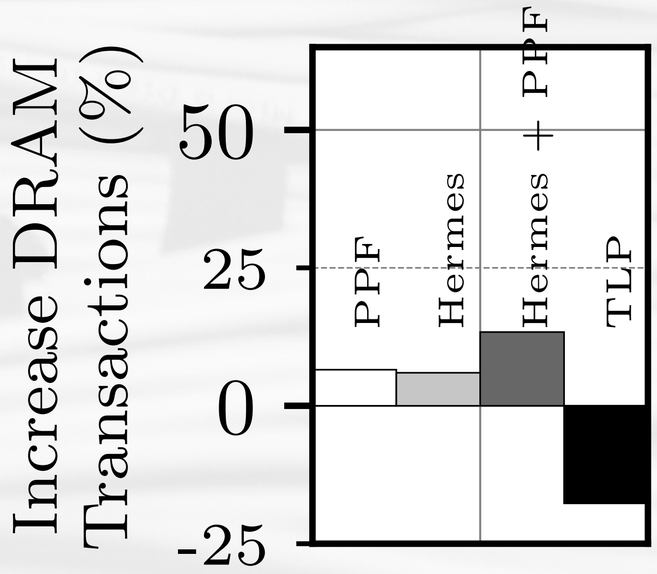
Single-Core Evaluation | DRAM Transactions



- Considering a single-core scenario, TLP significantly reduces DRAM transactions as compared all considered designs
  - **TLP** reduces DRAM transactions by **30.7%** on average, while DRAM transactions increase by **7.7%**, **5.2%**, and **13.3%** for **PPF**, **Hermes**, and **Hermes+PPF**, respectively

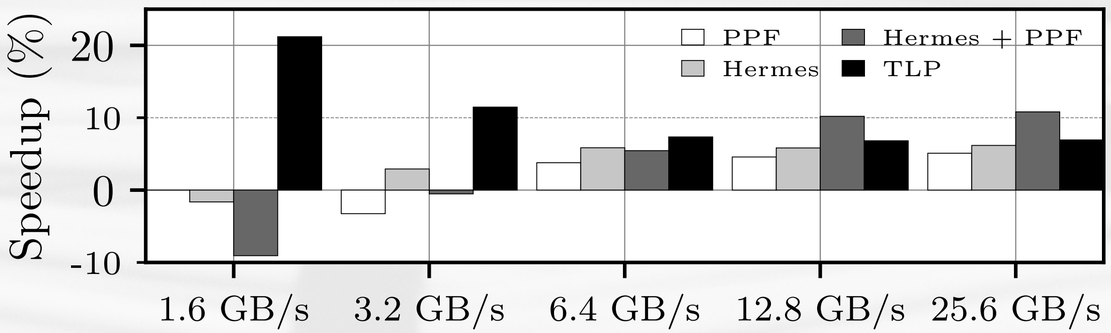
Multi-Core Evaluation | Performance & DRAM Transactions

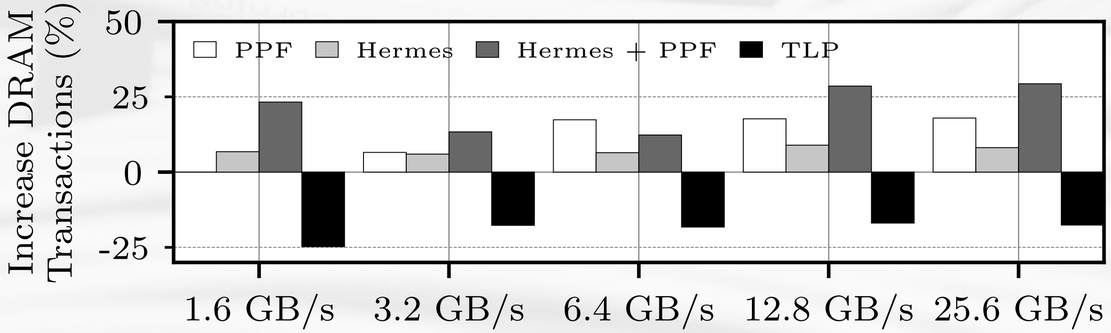




- Considering a multi-core scenario, TLP outperforms all considered designs
  - **n.b.**, The multi-core scenario assumes 3.2 GB/s per core of DRAM bandwidth
  - **TLP** leverages **11.5%** geomean speed-up as compared to **-3.3%**, **3.0%**, and **-0.5%** for **PPF**, **Hermes**, and **Hermes+PPF**, respectively
  - **TLP** reduces DRAM transactions by **17.7%** on average, while DRAM transactions increase by **6.5%**, **6.0%**, and **13.4%** for **PPF**, **Hermes**, and **Hermes+PPF**, respectively

Multi-Core Evaluation | Impact of Bandwidth





- Considering a multi-core scenario, TLP outperforms all considered designs
  - **n.b.**, The x-axes measures bandwidth in **GB/s per core**.
  - **n.b.**, Here, we vary DRAM bandwidth from **1.6** to **25.6** GB/s **per core**.
  - **TLP** outperforms all approaches in scenarios where bandwidth ranges between **1.6** to **6.4** GB/s **per core**
  - Even considering unrealistic scenarios (**12.8** and **25.6** GB/s per core), **TLP** outperforms **Hermes** and **PPF**

Conclusions

Conclusions

We show that off-chip prediction can be used to drive prefetch filtering, reducing the # of DRAM transactions and ultimately improving performance.

Conclusions

We show that off-chip prediction can be used to drive prefetch filtering, reducing the # of DRAM transactions and ultimately improving performance.

This work generalises to any L1D prefetcher and/or workloads.

Conclusions

We show that off-chip prediction can be used to drive prefetch filtering, reducing the # of DRAM transactions and ultimately improving performance.

This work generalises to any L1D prefetcher and/or workloads.

Off-chip prediction could be correlated to other μ-architectural problems (*Dead-On-Arrival* STLB entries, etc.).

Reproducing results — Public Artifact

- An artifact is publicly available for you to reproduce the results and play around.
  - Main artifact on Zenodo.
- We provide the traces used for the paper.
  - Volume 1 on Zenodo.
  - Volume 2 on Zenodo.
  - Volume 3 on Zenodo.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES

HUAWEI

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Generalitat de Catalunya

Thank you for your attention!

BACKUP SLIDES

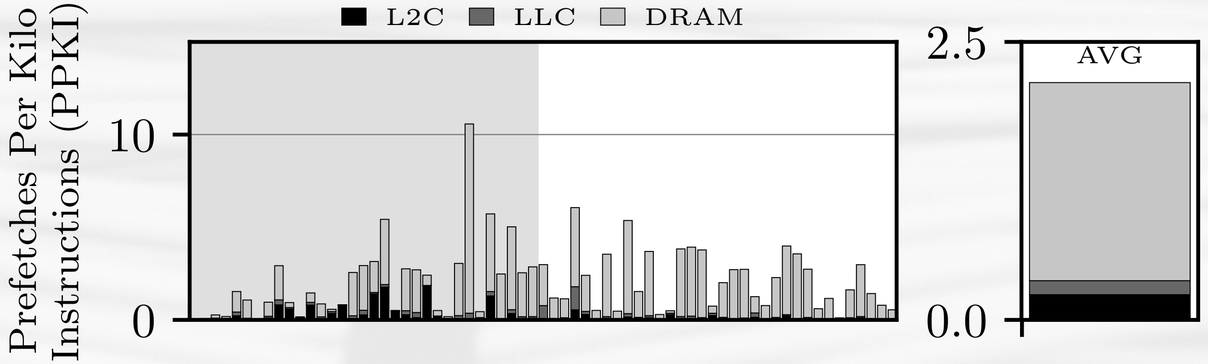
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

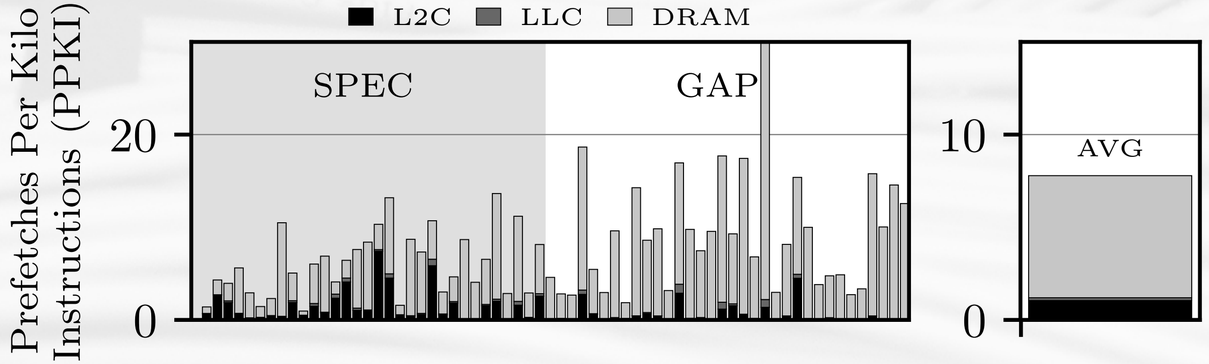
Details on Graph Workloads

	BC	BFS	CC	PR	TC	SSP
irregData ElemSz	8B + 4B	4B	4B	4B	4B	4B
Execution style	Push-Mostly	Push & Pull	Push-Mostly	Pull-Only	Push-Only	Push-Only
Use Frontier	Yes	Yes	No	No	No	Yes

	Web	Road	Twitter	Kron	Urand	Friendster
# Vertices (in M)	50.6	23.9	61.6	134.2	134.2	65.2
# Edges (in M)	1,949.4	58.3	1,468.4	2,111.6	2,147.4	3,612.1

Hardware Prefetching & Prefetch Filtering (Berti)

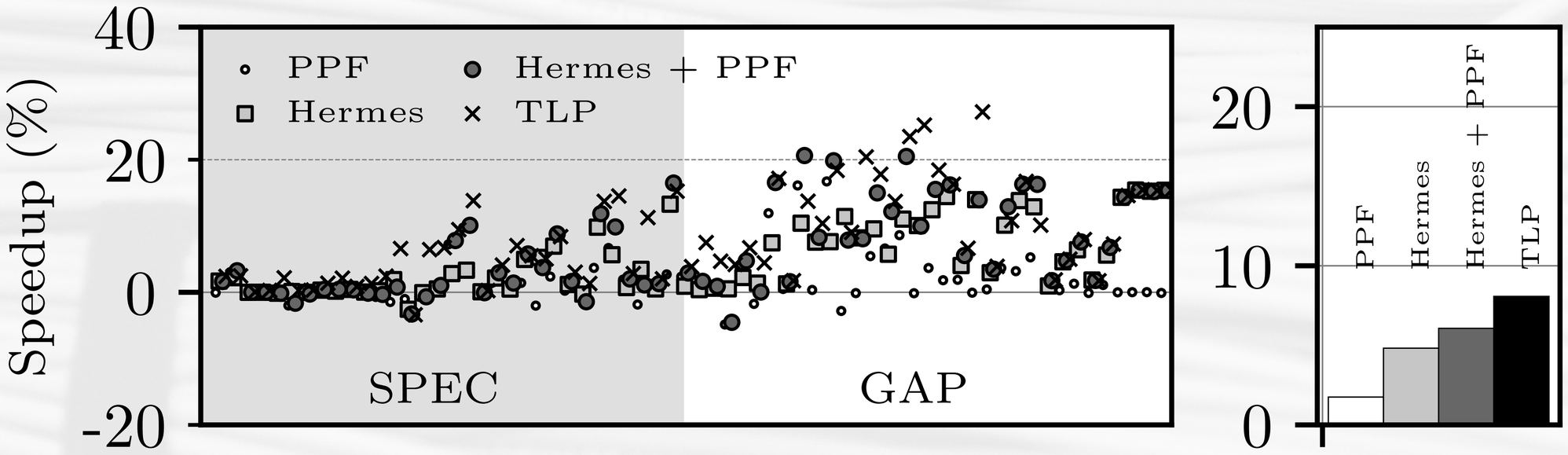




- Where are useless (not touched during their lifetime) prefetches fetched from?
  - L2C —> 18.2%
  - LLC —> 3.8%
  - DRAM —> 78%

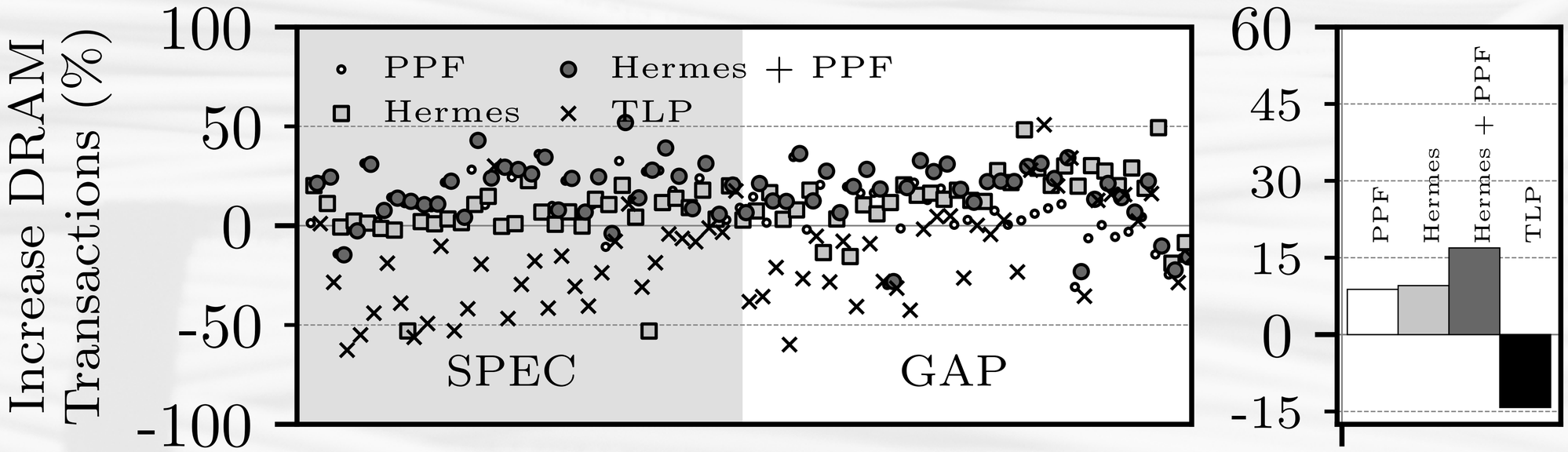
- On average 95.2% of prefetch requests served by the DRAM are inaccurate
  - Worse on GAP workloads (96.7%) as compared to SPEC (82%).

Single-Core Evaluation / Performance (Berti)



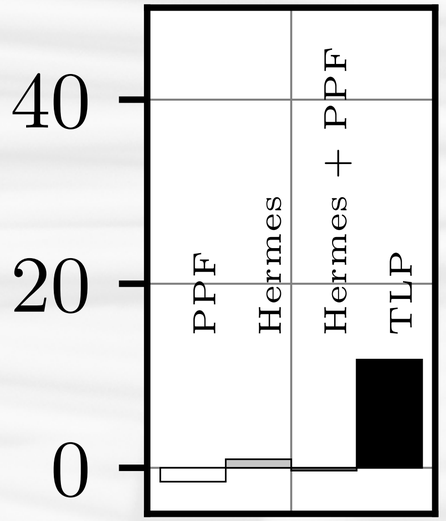
- Considering a single-core scenario, TLP outperforms all considered designs.
  - **n.b.**, The single-core scenario assumes 12.8 GB/s per core of DRAM bandwidth. This favours aggressive prefetching techniques.
  - TLP leverages 8.1% geomean speed-up as compared to 1.7%, 4.8%, and 6.1% for PPF, Hermes, and Hermes+PPF, respectively.

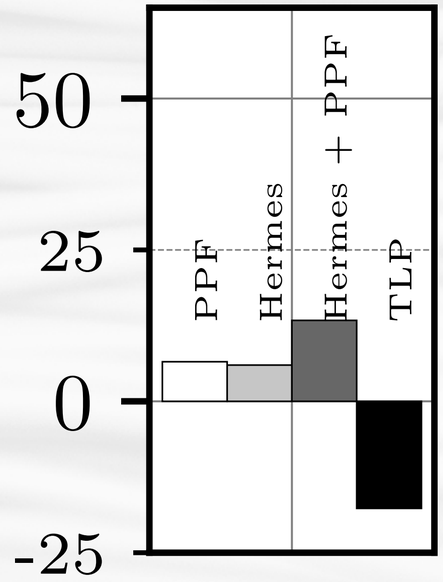
Single-Core Evaluation / DRAM Transactions (Berti)



- Considering a single-core scenario, TLP significantly reduces DRAM transactions as compared all considered designs.
  - TLP reduces DRAM transactions by 14.2% on average, while DRAM transactions increase by 8.8%, 9.6%, and 16.9% for PPF, Hermes, and Hermes+PPF, respectively.

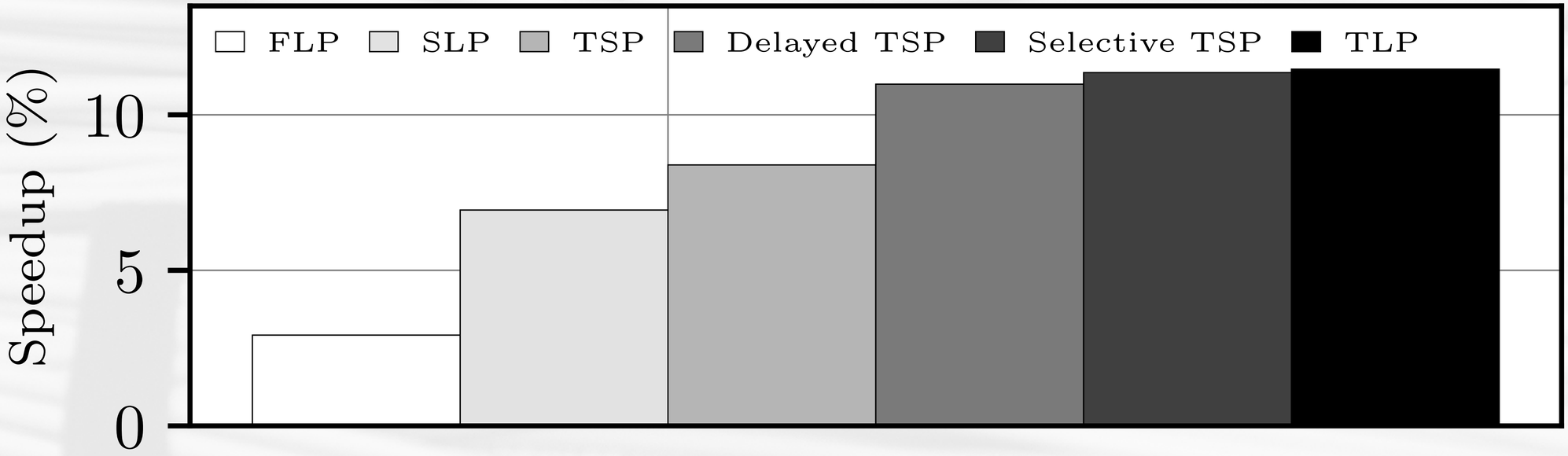
Multi-Core Evaluation / Performance & DRAM Transactions (Berti)





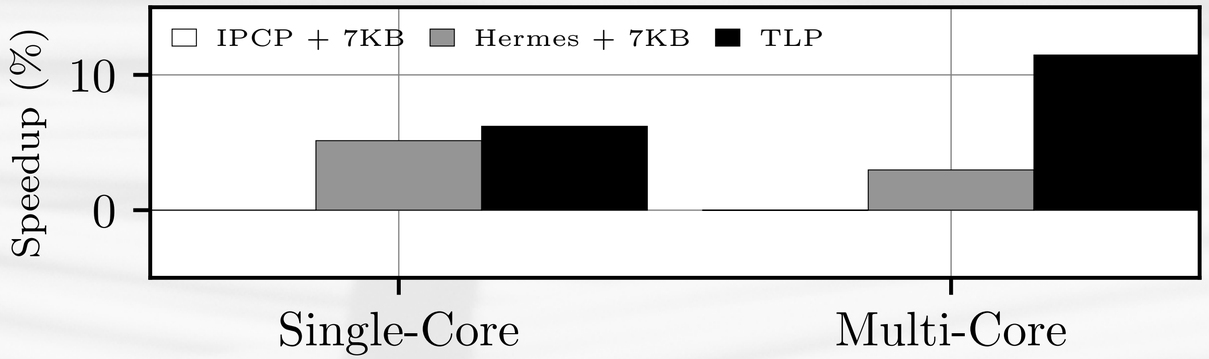
- Considering a multi-core scenario, TLP outperforms all considered designs.
  - **n.b.**, The multi-core scenario assumes 3.2 GB/s per core of DRAM bandwidth. The favours less aggressive prefetching techniques.
  - TLP leverages 11.8% geomean speed-up as compared to -1.5%, 1.0%, and 0.3% for PPF, Hermes, and Hermes+PPF, respectively.
  - TLP reduces DRAM transactions by 17.7% on average, while DRAM transactions increase by 6.5%, 6.0%, and 13.4% for PPF, Hermes, and Hermes+PPF, respectively.

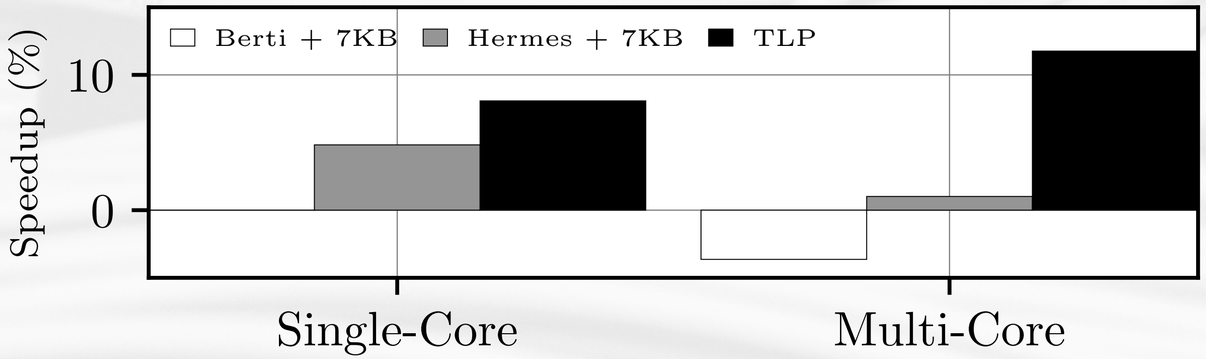
Performance contribution of each TLP component



- We breakdown the the performance contribution of each building blocks of the final TLP proposal:
  - FLP —> 2.9%
  - SLP —> 6.9%
  - TSP —> 8.4%
  - Delayed TSP —> 10.2%
  - Selective TSP —> 11.4%
  - TLP —> 11.5%

Enhanced designs with TLP's budget





- Comparing TLP do designs enhanced with TLP's hardware budget (7KB)
  - IPCP + 7KB —> Enhancing IPCP does not provide any improvements.
  - Berti + 7KB —> Single-core: 0.0% | Multi-core: -3.6%
  - Hermes + 7KB —> Single-core: 5.2% | Multi-core: 4.8%