ALTERNATE PATH μ-OP CACHE PREFETCHING

Sawan Singh[1] Arthur Perais[2] Alexandra Jimborean[1]
Alberto Ros[1]

[1]Computer Engineering Department
University of Murcia, Spain

[2]TIMA, Univ. Grenoble Alpes, CNRS,
Grenoble INP, Grenoble, France

ISCA'51, Session 10A, July 3, 2024

Overview

→ μ-op Cache

Overview

→ μ-op Cache

- Holds recently decoded μ-ops

Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder

[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder
- Current μ-op caches are severely overwhelmed by server workloads

[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder
- Current μ-op caches are severely overwhelmed by server workloads
  - Only provide 0.87% improvement over no μ-op cache

[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder
- Current μ-op caches are severely overwhelmed by server workloads
  - Only provide 0.87% improvement over no μ-op cache
  - Ideal μ-op cache can provide 10.82% improvement

[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder
- Current μ-op caches are severely overwhelmed by server workloads
  - Only provide 0.87% improvement over no μ-op cache
  - Ideal μ-op cache can provide 10.82% improvement

→ We propose UCP (*Alternate Path μ-op Cache Prefetching*)

[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder
- Current μ-op caches are severely overwhelmed by server workloads
  - Only provide 0.87% improvement over no μ-op cache
  - Ideal μ-op cache can provide 10.82% improvement

→ We propose UCP (*Alternate Path μ-op Cache Prefetching*)

- Identify hard-to-predict branches

H2P Branch

[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

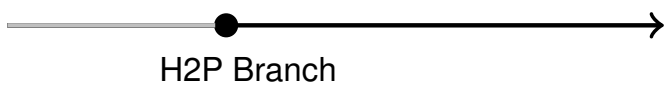
Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder
- Current μ-op caches are severely overwhelmed by server workloads
  - Only provide 0.87% improvement over no μ-op cache
  - Ideal μ-op cache can provide 10.82% improvement

→ We propose UCP (*Alternate Path μ-op Cache Prefetching*)

- Identify hard-to-predict branches
- Prefetch μ-ops from alternate path of the hard-to-predict branch



[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

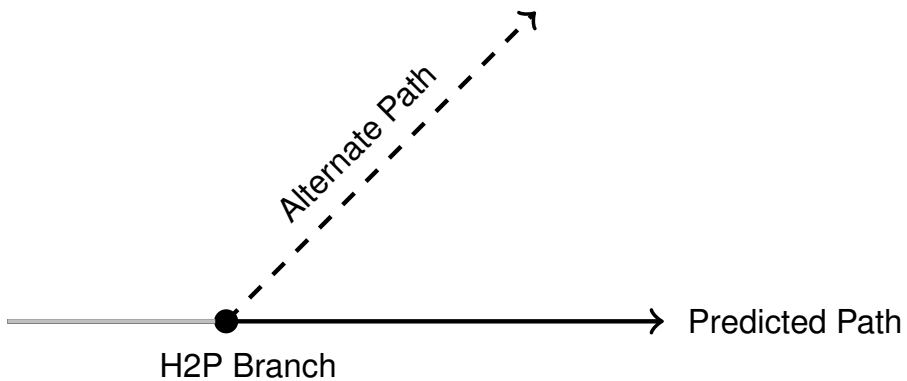
Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder
- Current μ-op caches are severely overwhelmed by server workloads
  - Only provide 0.87% improvement over no μ-op cache
  - Ideal μ-op cache can provide 10.82% improvement

→ We propose UCP (*Alternate Path μ-op Cache Prefetching*)

- Identify hard-to-predict branches
- Prefetch μ-ops from alternate path of the hard-to-predict branch



[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

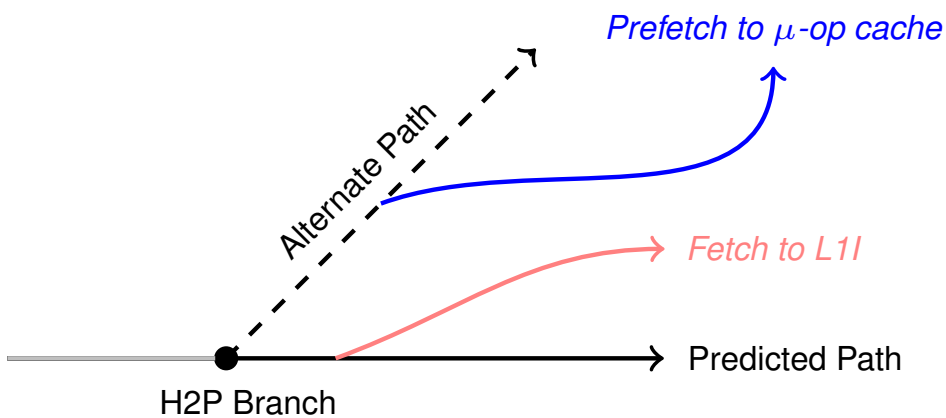
Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoder
- Current μ-op caches are severely overwhelmed by server workloads
  - Only provide 0.87% improvement over no μ-op cache
  - Ideal μ-op cache can provide 10.82% improvement

→ We propose UCP (*Alternate Path μ-op Cache Prefetching*)

- Identify hard-to-predict branches
- Prefetch μ-ops from alternate path of the hard-to-predict branch

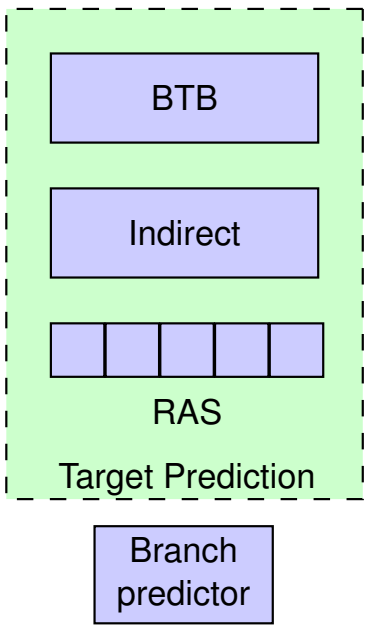


[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

Outline

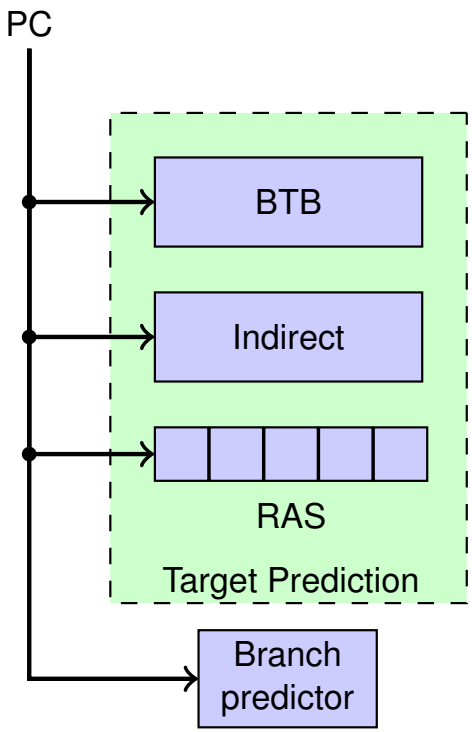
Background & Motivation

Processor Front-End



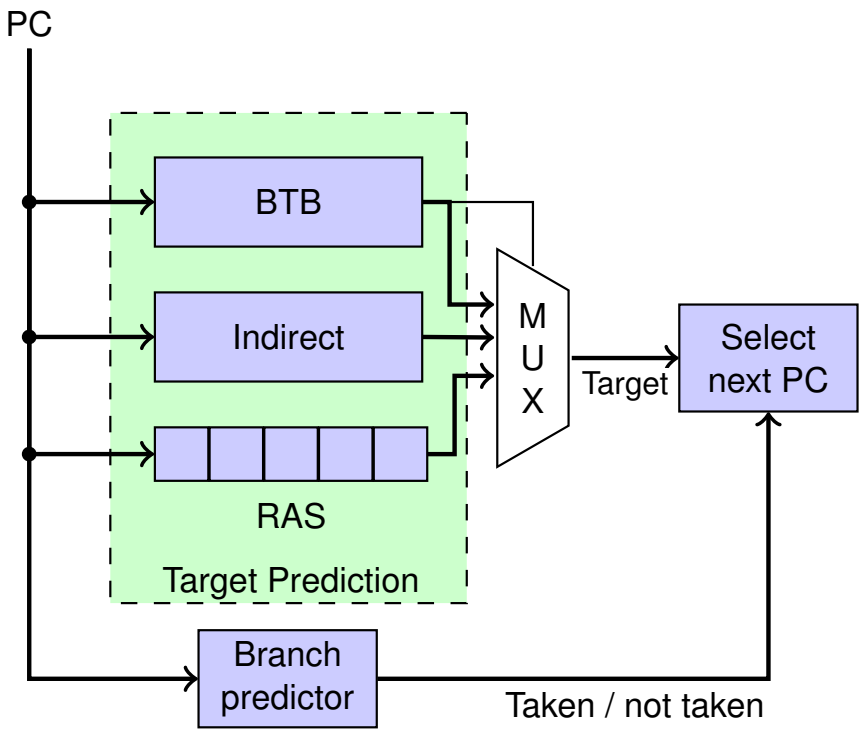
Background & Motivation

Processor Front-End



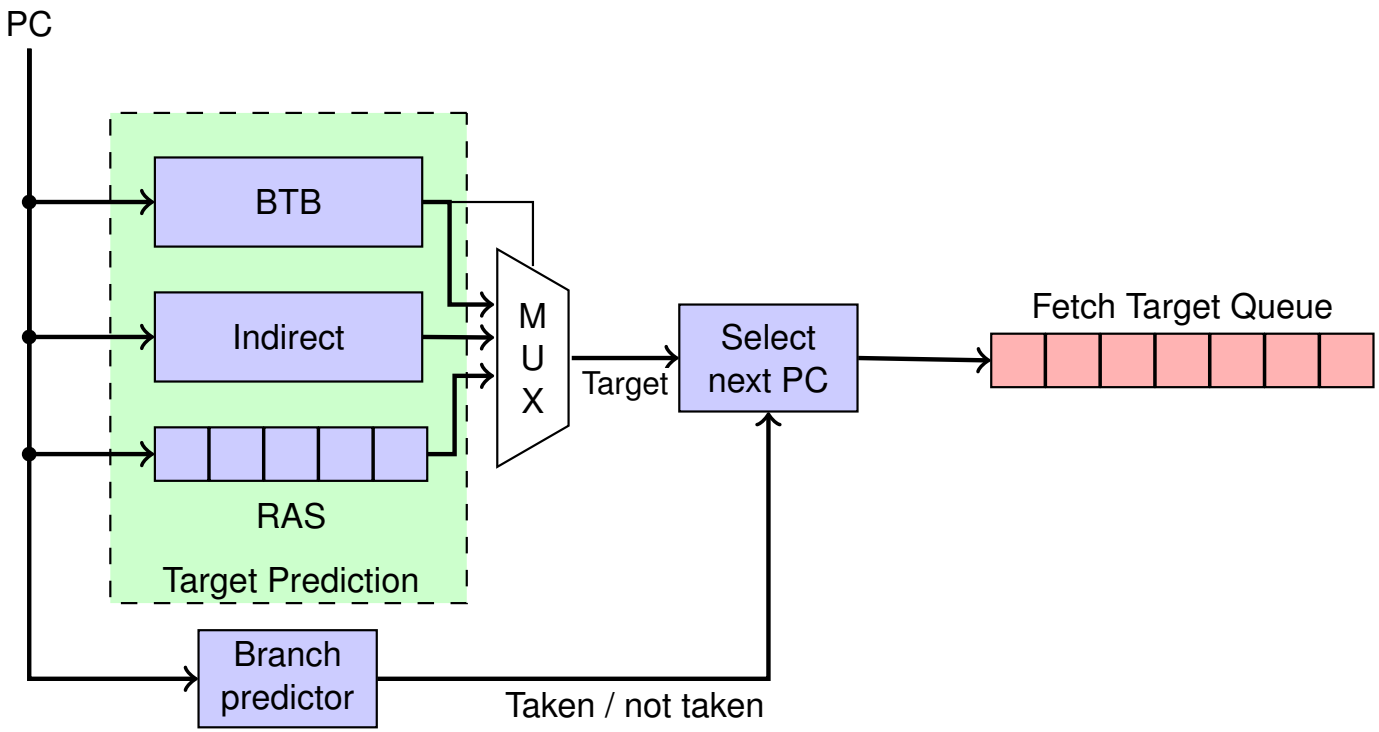
Background & Motivation

Processor Front-End



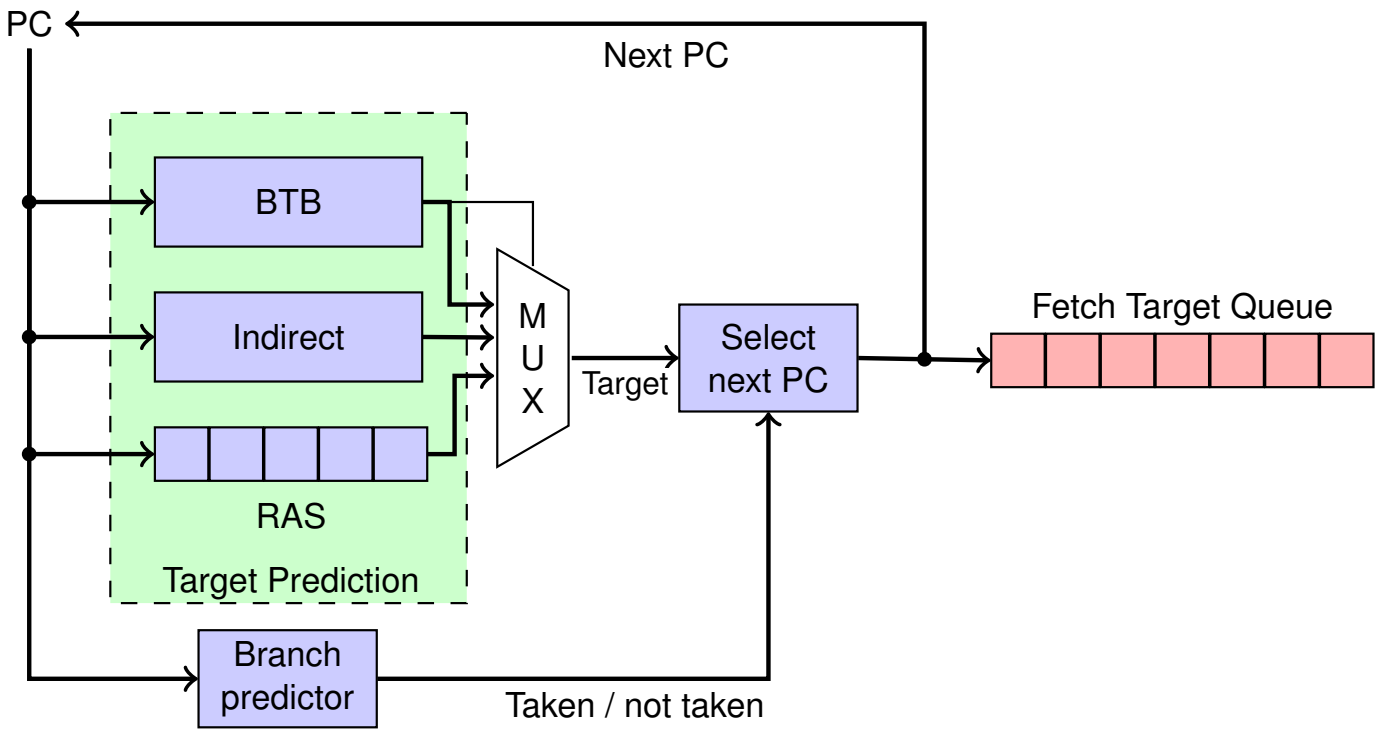
Background & Motivation

Processor Front-End



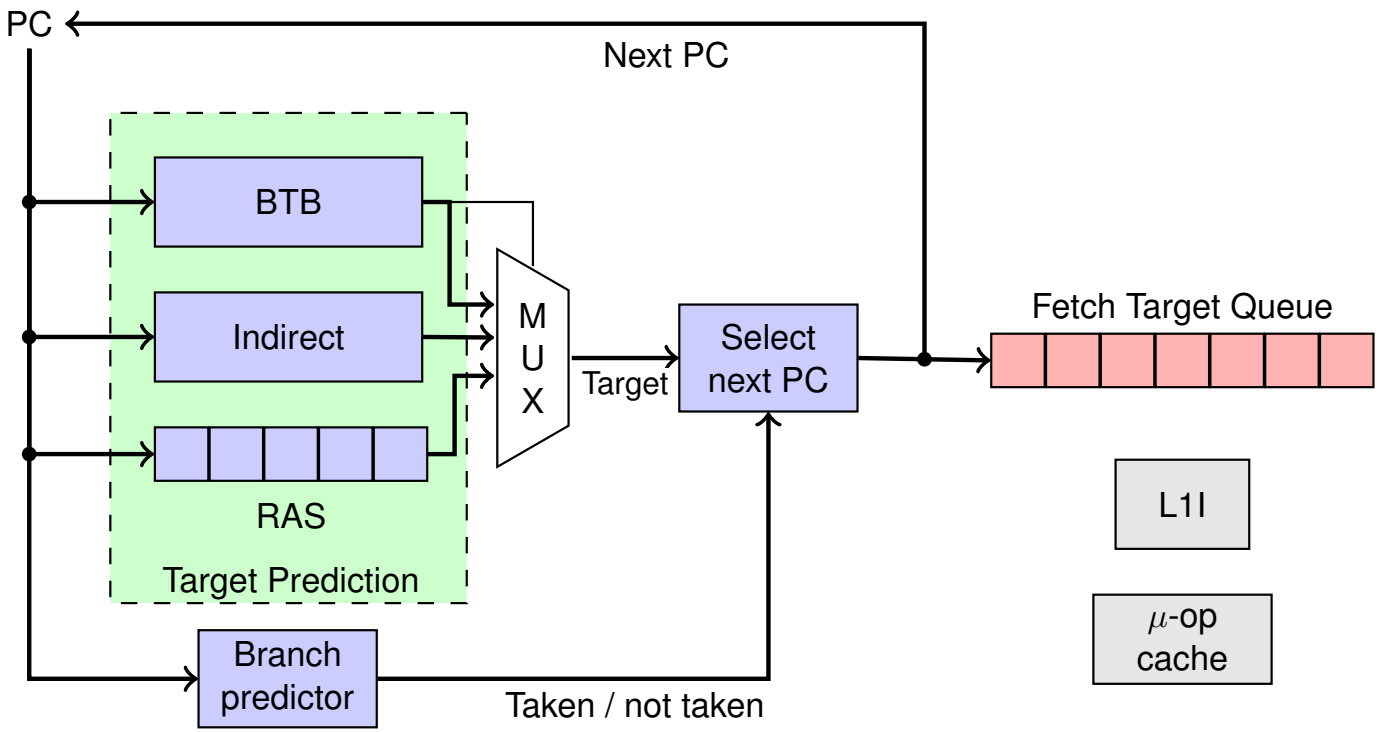
Background & Motivation

Processor Front-End



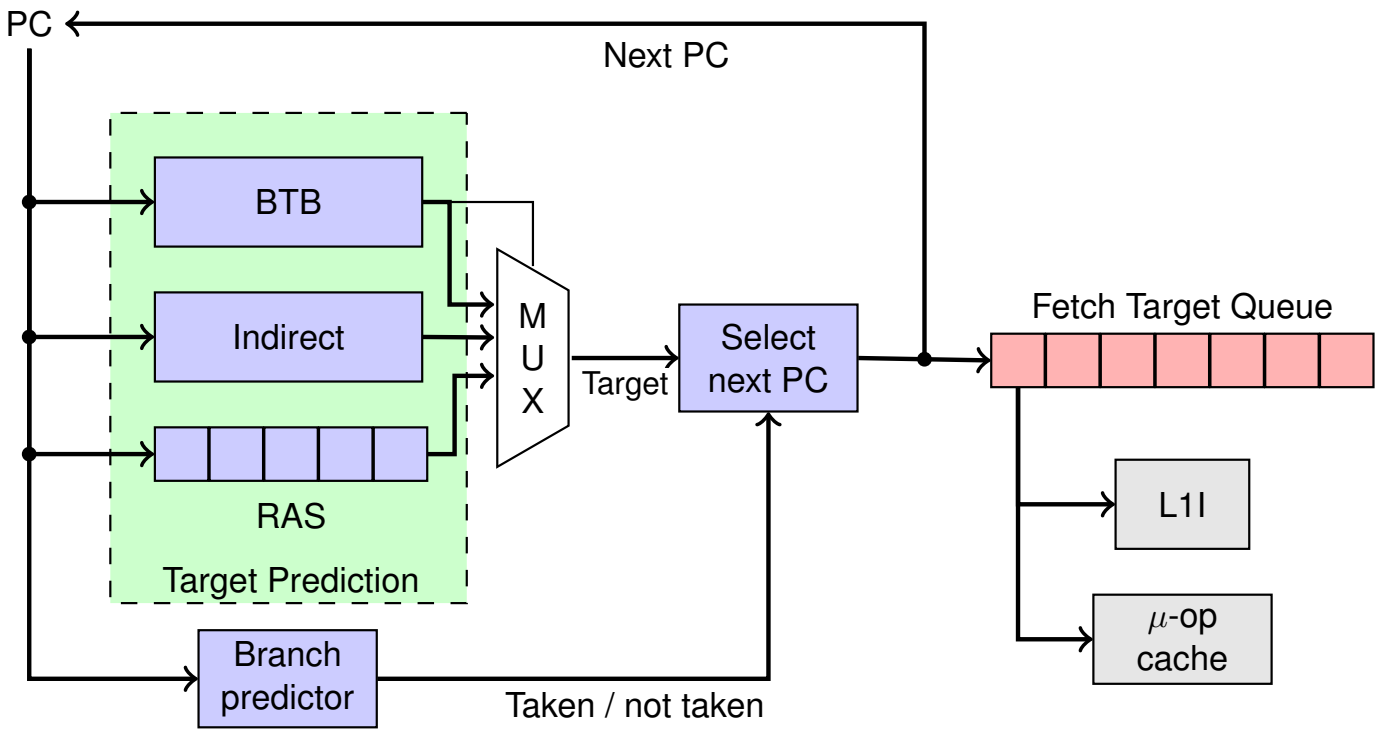
Background & Motivation

Processor Front-End



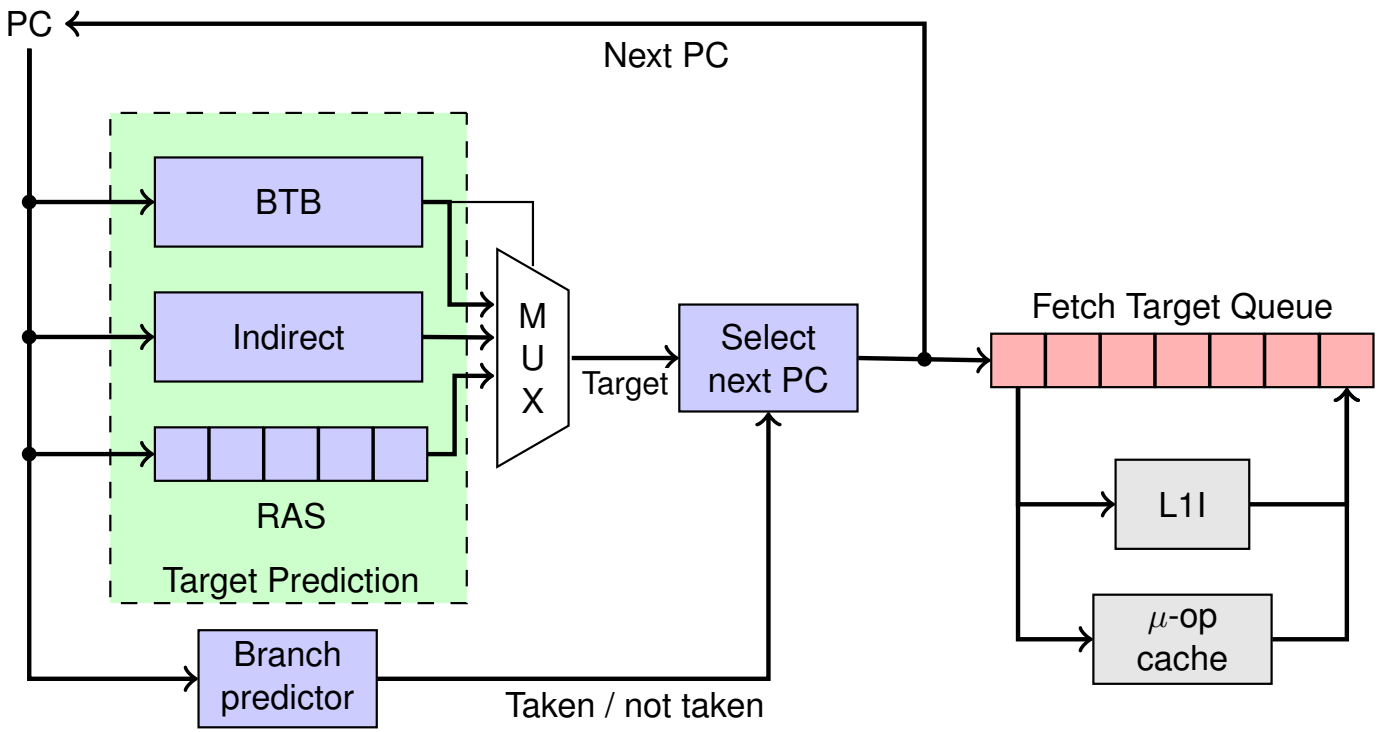
Background & Motivation

Processor Front-End



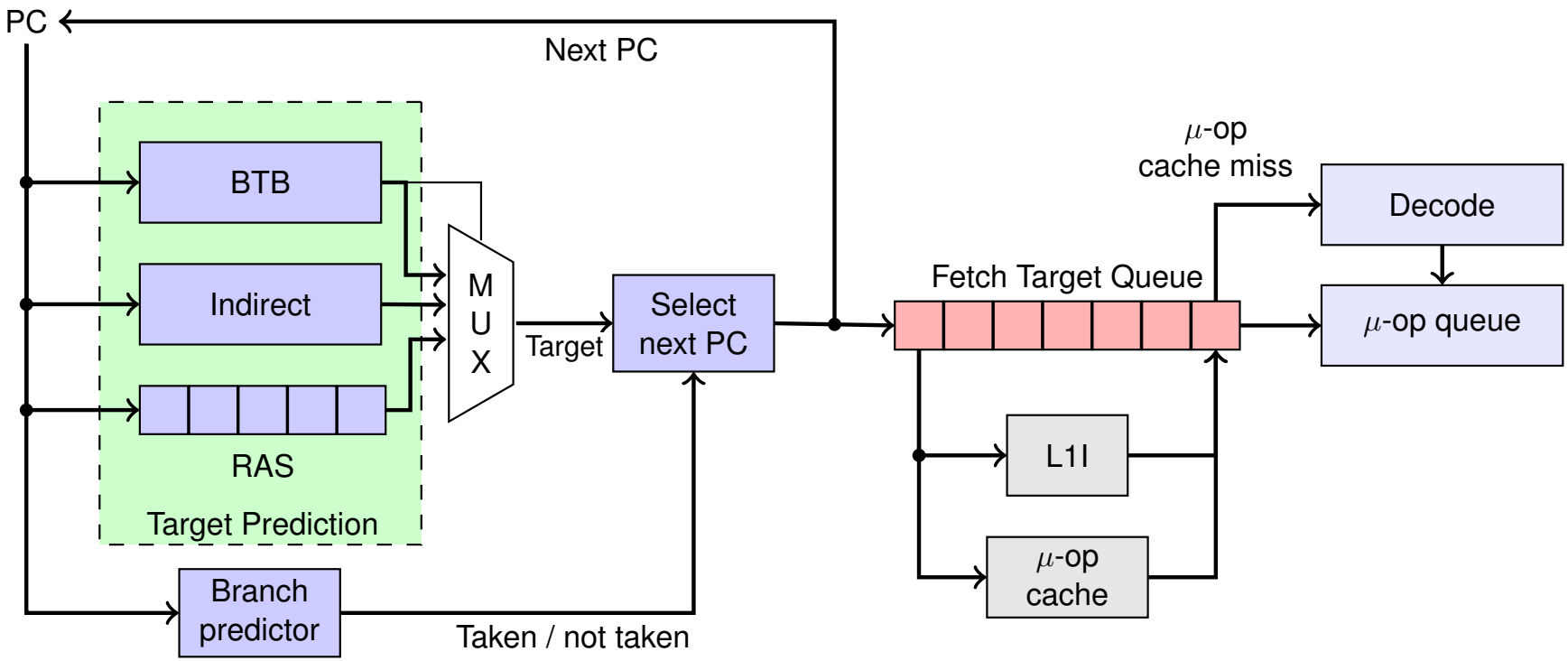
Background & Motivation

Processor Front-End



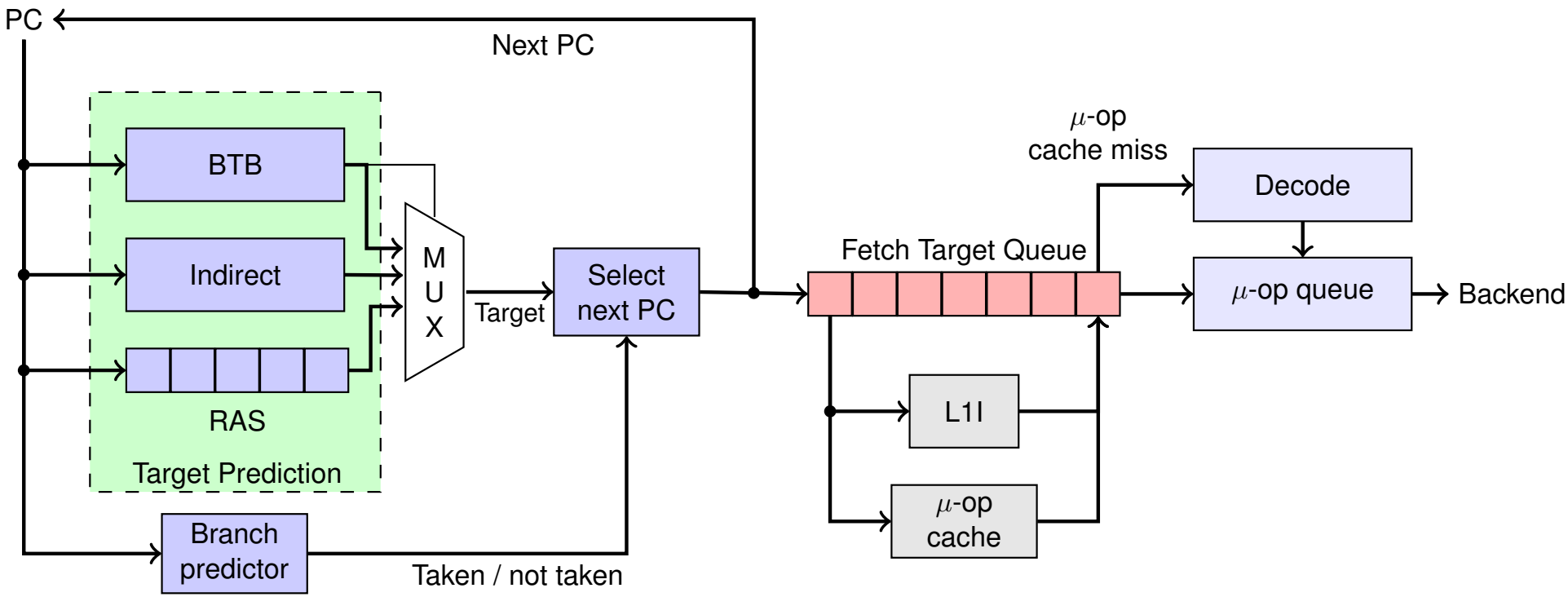
Background & Motivation

Processor Front-End



Background & Motivation

Processor Front-End



Background & Motivation

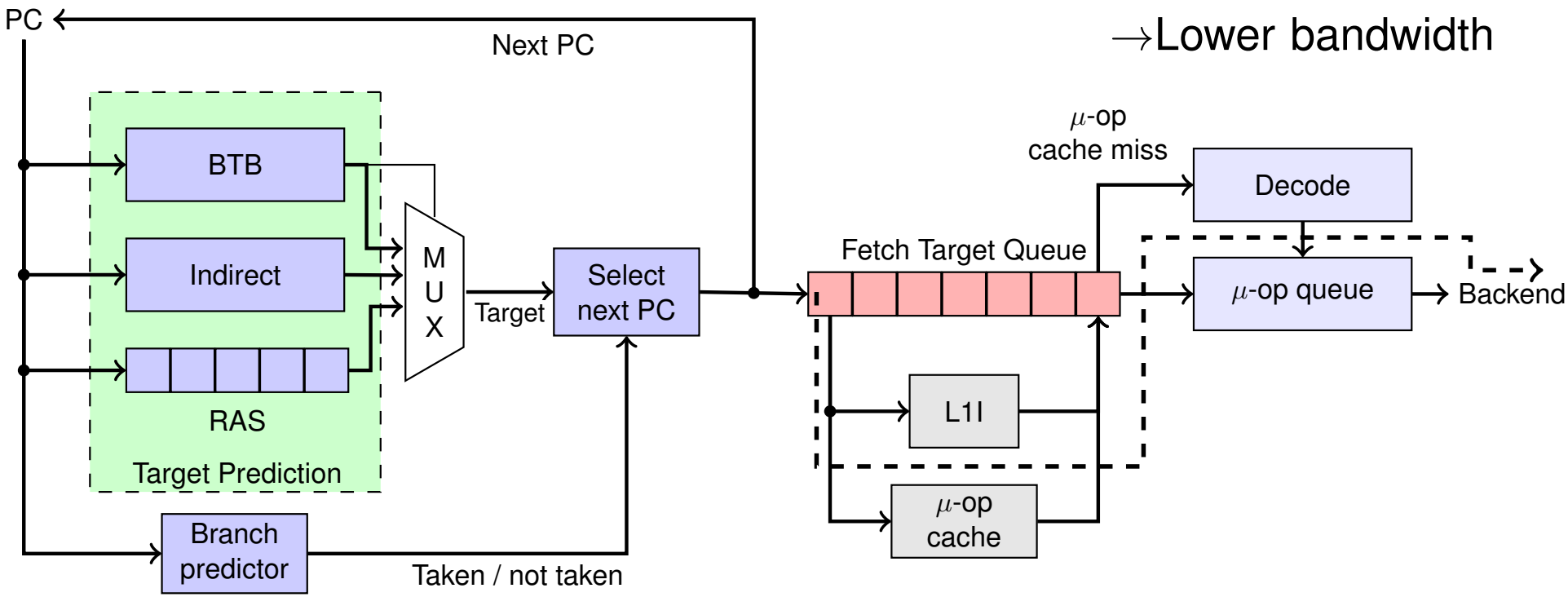
Processor Front-End

From L1I:

→Decode latency

→Decode energy

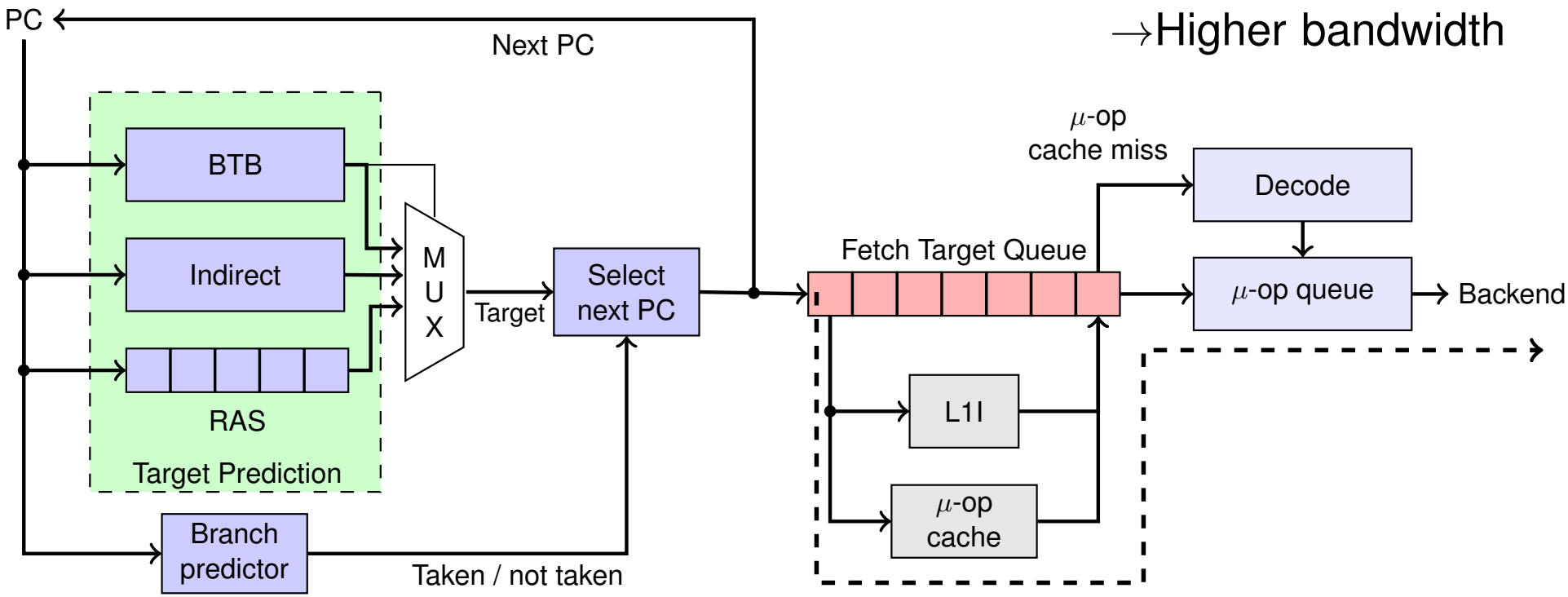
→Lower bandwidth



Background & Motivation

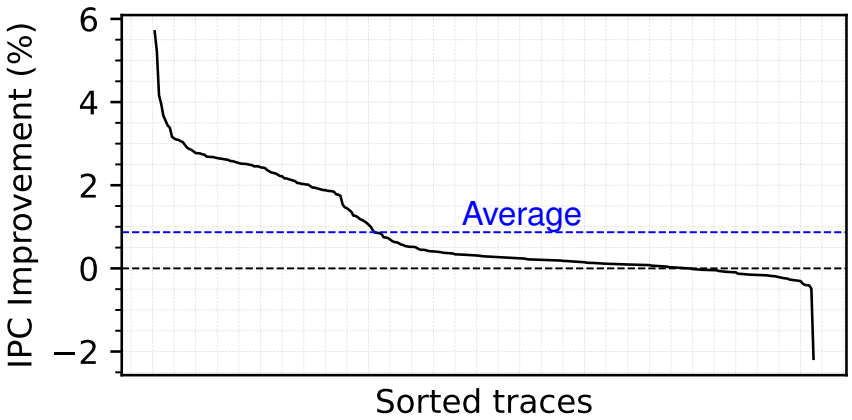
Processor Front-End

From μ-op cache:
- →~~Decode latency~~
- →~~Decode energy~~
- →Higher bandwidth



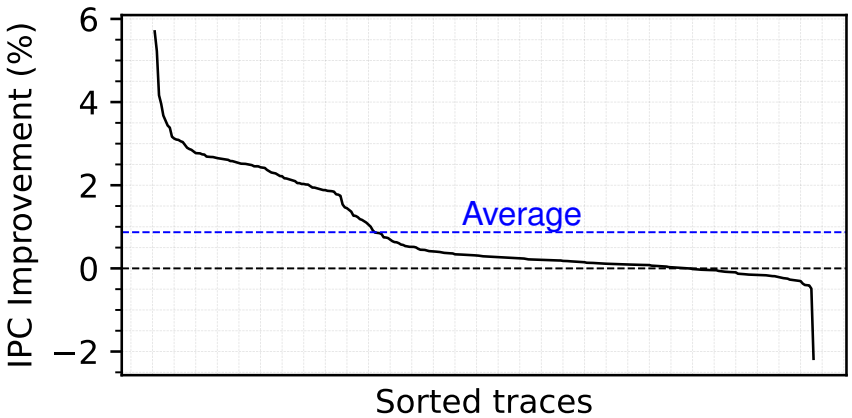
Background & Motivation

Performance of μ-ops cache with server workloads



Background & Motivation

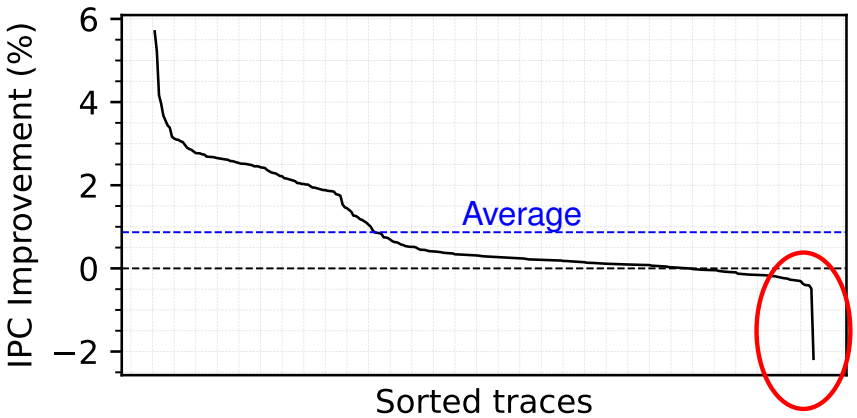
Performance of μ-ops cache with server workloads



$\rightarrow$ 4Kops μ-op provides only 0.87% improvement over no μ-op cache

Background & Motivation

Performance of μ-ops cache with server workloads

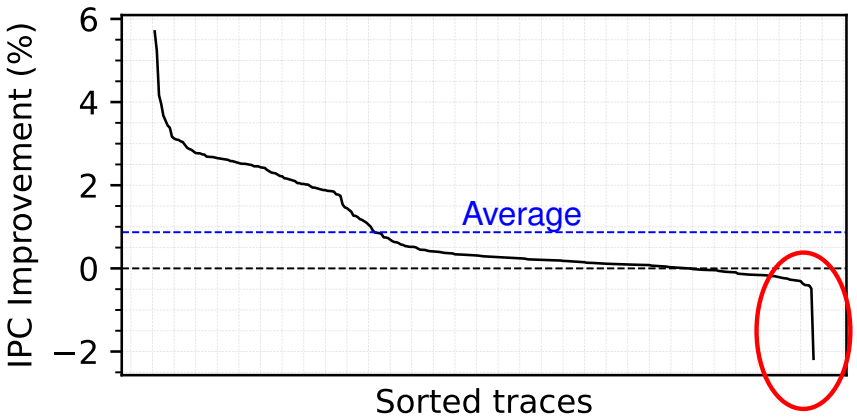


→ 4Kops μ-op provides only 0.87% improvement over no μ-op cache

→ 19.3% of traces show a slowdown

Background & Motivation

Performance of μ-ops cache with server workloads

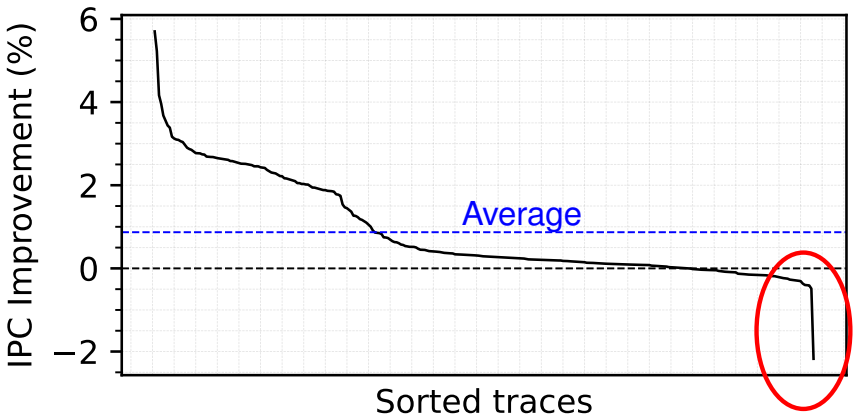


- → 4Kops μ-op provides only 0.87% improvement over no μ-op cache
- → 19.3% of traces show a slowdown
- → Increasing size of μ-op cache does not help

IPC Improvement (w.r.t 4Kops μ-op cache) →

Background & Motivation

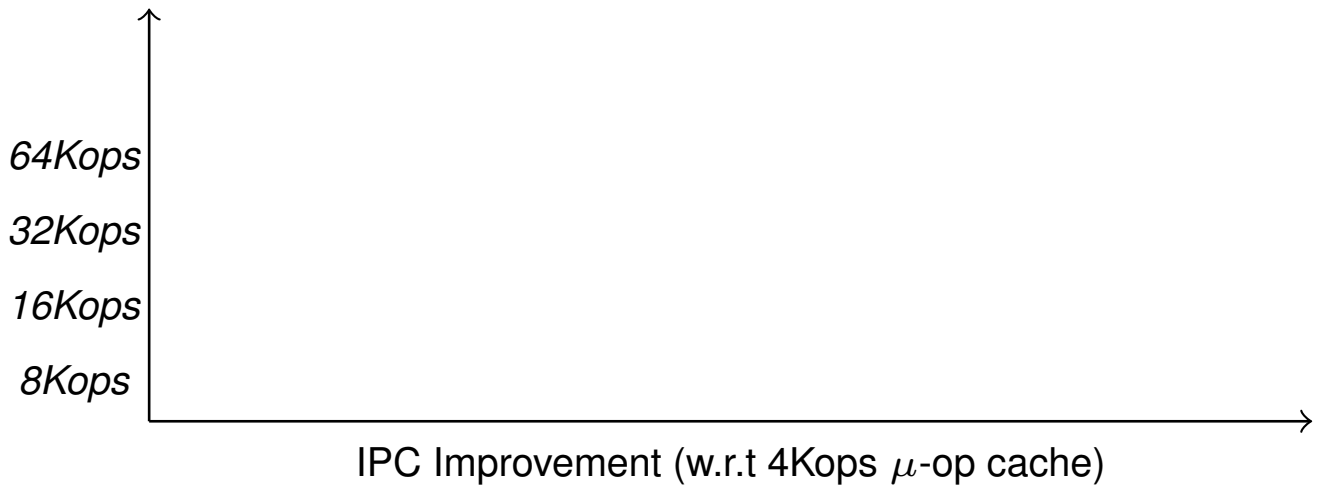
Performance of μ-ops cache with server workloads



→ 4Kops μ-op provides only 0.87% improvement over no μ-op cache

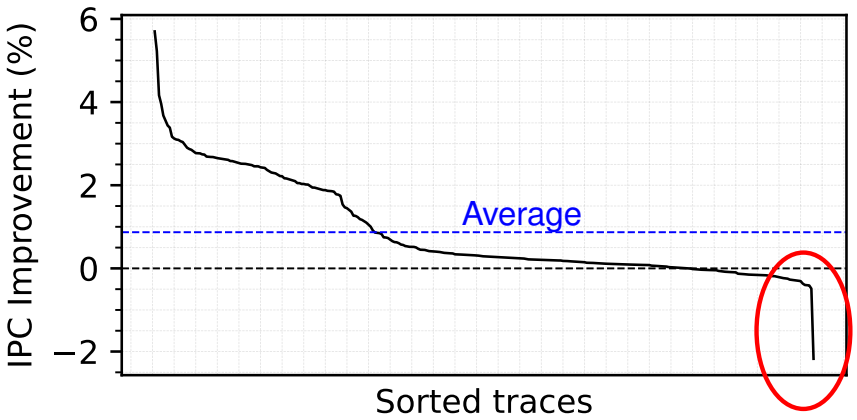
→ 19.3% of traces show a slowdown

→ Increasing size of μ-op cache does not help



Background & Motivation

Performance of μ-ops cache with server workloads



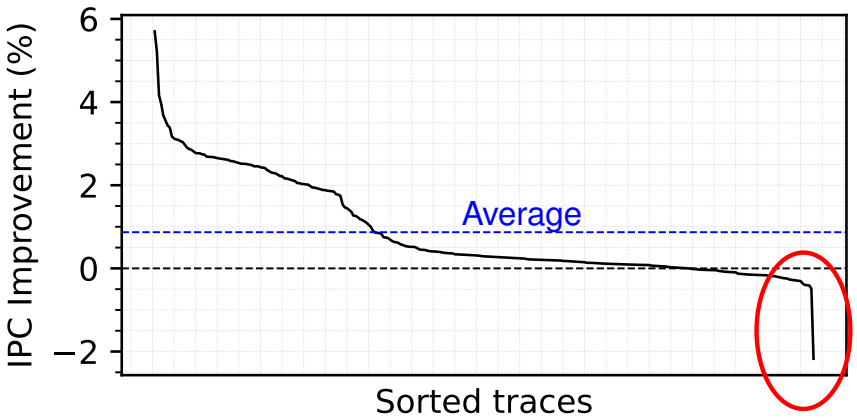
→ 4Kops μ-op provides only 0.87% improvement over no μ-op cache

→ 19.3% of traces show a slowdown

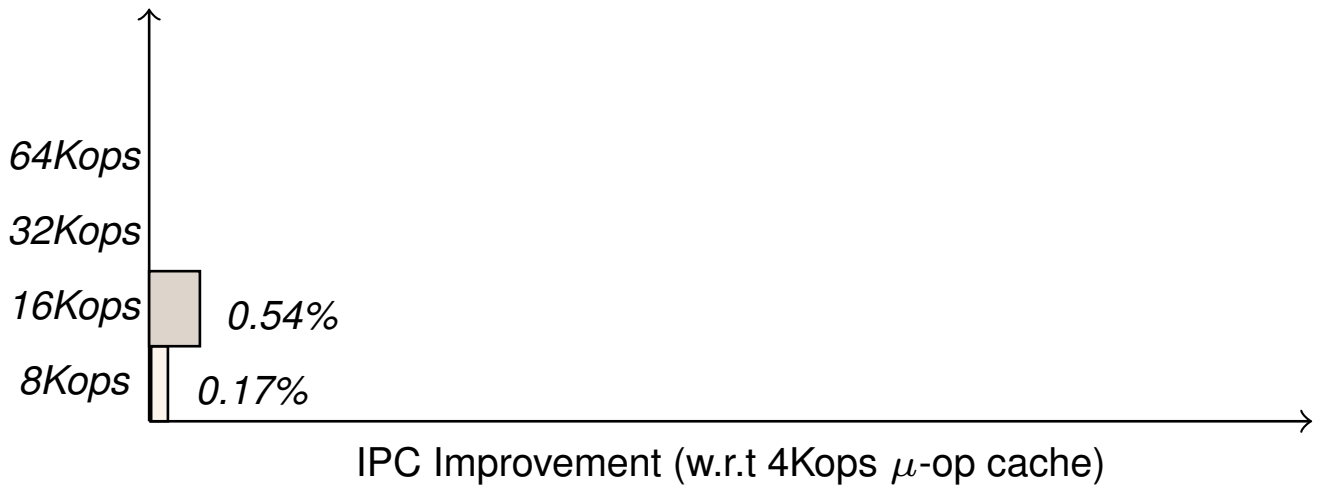
→ Increasing size of μ-op cache does not help

Background & Motivation

Performance of μ-ops cache with server workloads

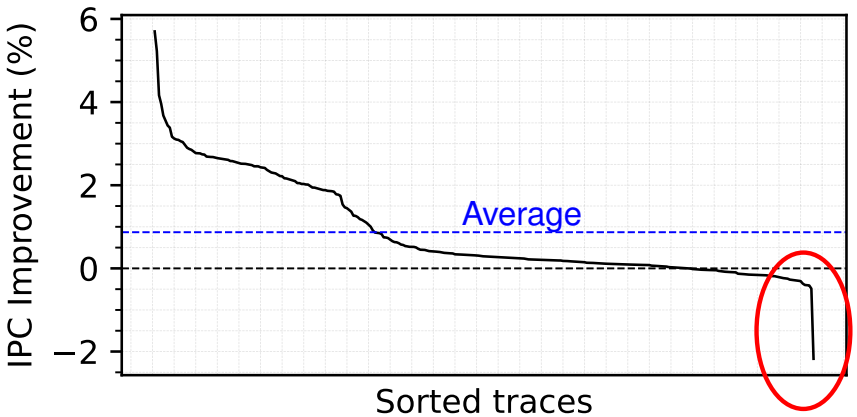


- → 4Kops μ-op provides only 0.87% improvement over no μ-op cache
- → 19.3% of traces show a slowdown
- → Increasing size of μ-op cache does not help

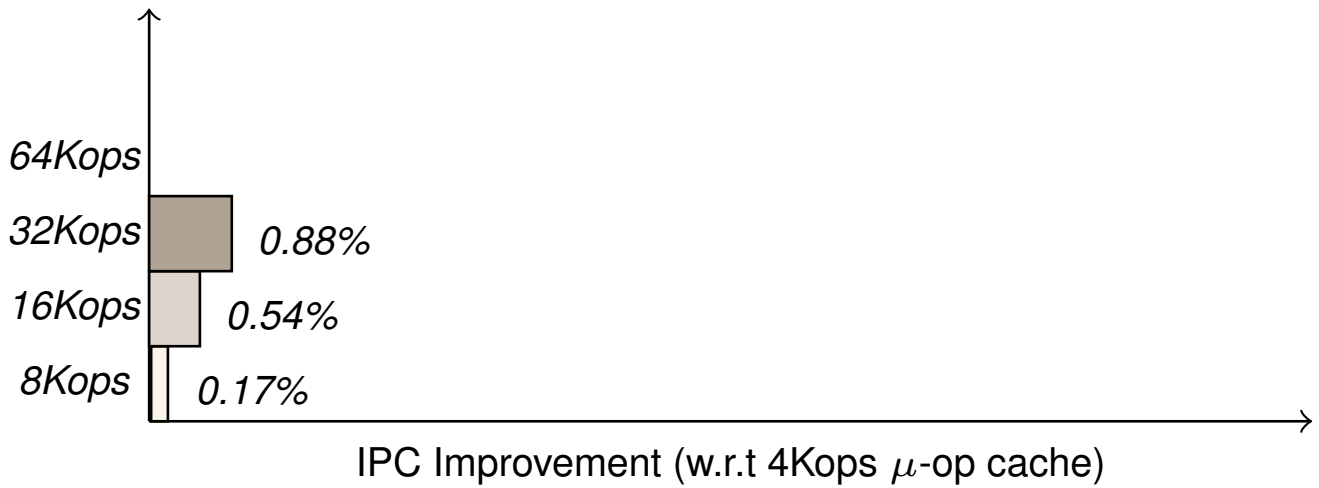


Background & Motivation

Performance of μ-ops cache with server workloads

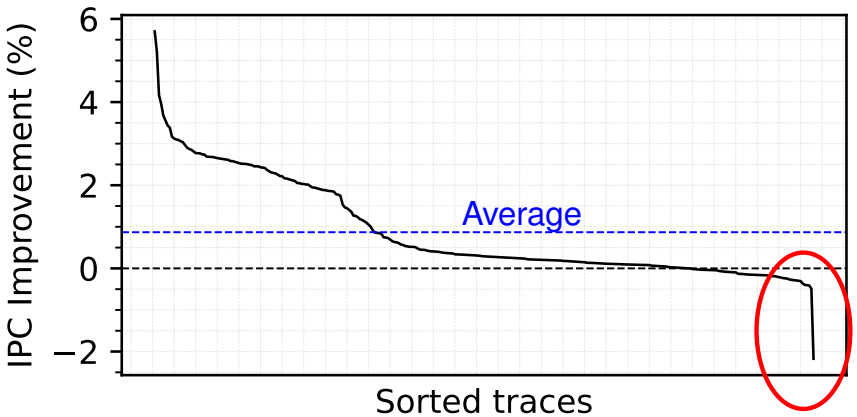


- → 4Kops μ-op provides only 0.87% improvement over no μ-op cache
- → 19.3% of traces show a slowdown
- → Increasing size of μ-op cache does not help

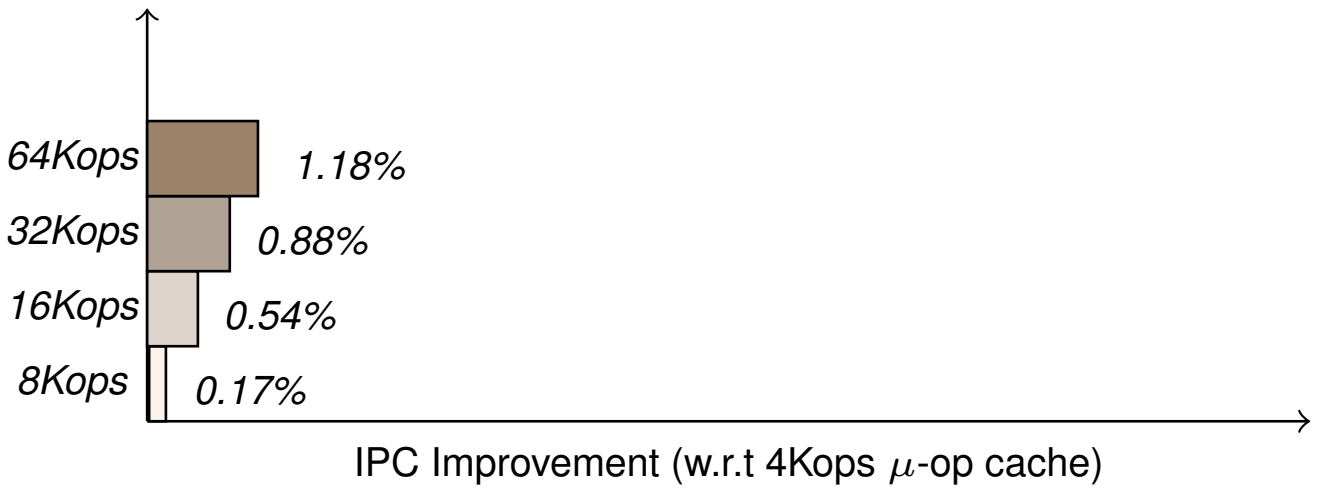


Background & Motivation

Performance of μ-ops cache with server workloads

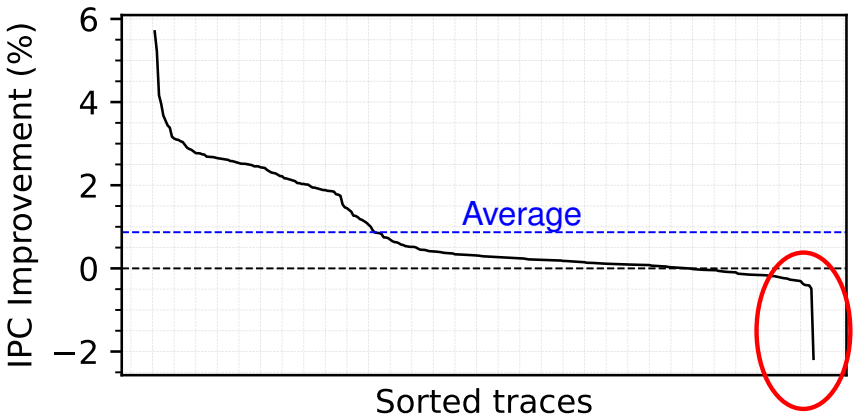


- → 4Kops μ-op provides only 0.87% improvement over no μ-op cache
- → 19.3% of traces show a slowdown
- → Increasing size of μ-op cache does not help



Background & Motivation

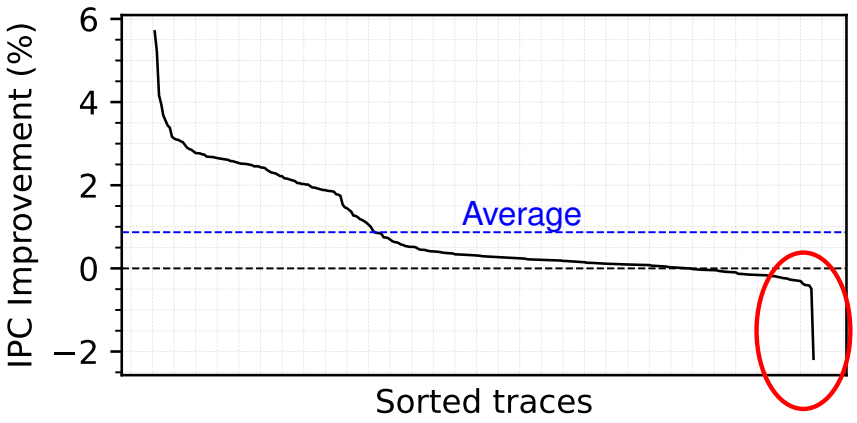
Performance of μ-ops cache with server workloads



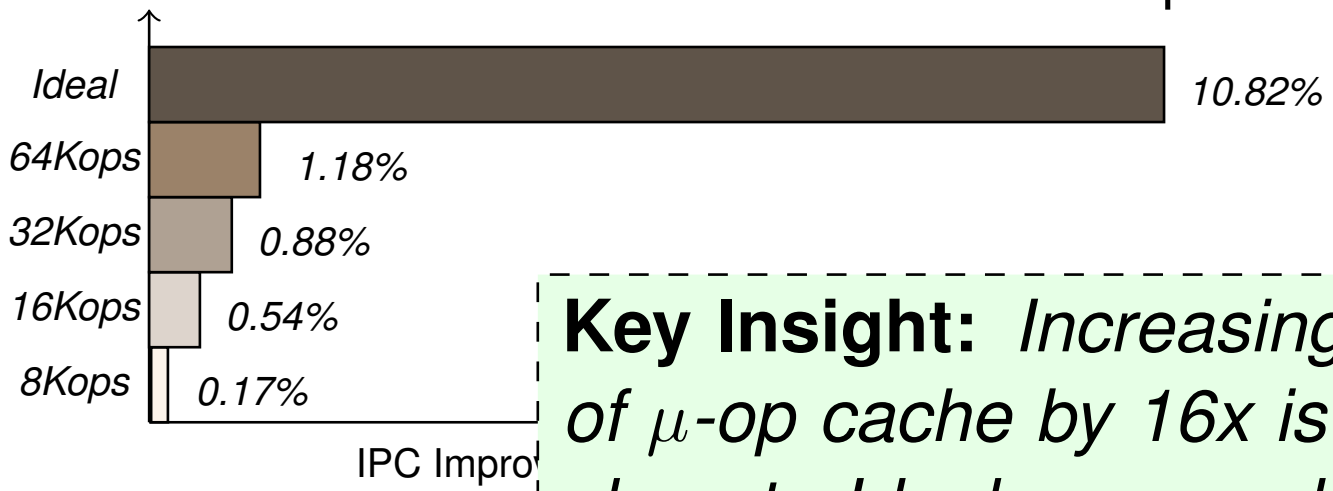
- → 4Kops μ-op provides only 0.87% improvement over no μ-op cache
- → 19.3% of traces show a slowdown
- → Increasing size of μ-op cache does not help
- → Ideal μ-op cache can provide 10.82% improvement

Background & Motivation

Performance of μ-ops cache with server workloads



- → 4Kops μ-op provides only 0.87% improvement over no μ-op cache
- → 19.3% of traces show a slowdown
- → Increasing size of μ-op cache does not help
- → Ideal μ-op cache can provide 10.82% improvement



Key Insight: *Increasing the size of μ-op cache by 16x is still not close to Ideal μ-op cache*

Background & Motivation

FTQ behavior on branch miss

→ FTQ is unable to hide the L1I miss latency on branch misprediction

Background & Motivation

FTQ behavior on branch miss

→ FTQ is unable to hide the L1I miss latency on branch misprediction

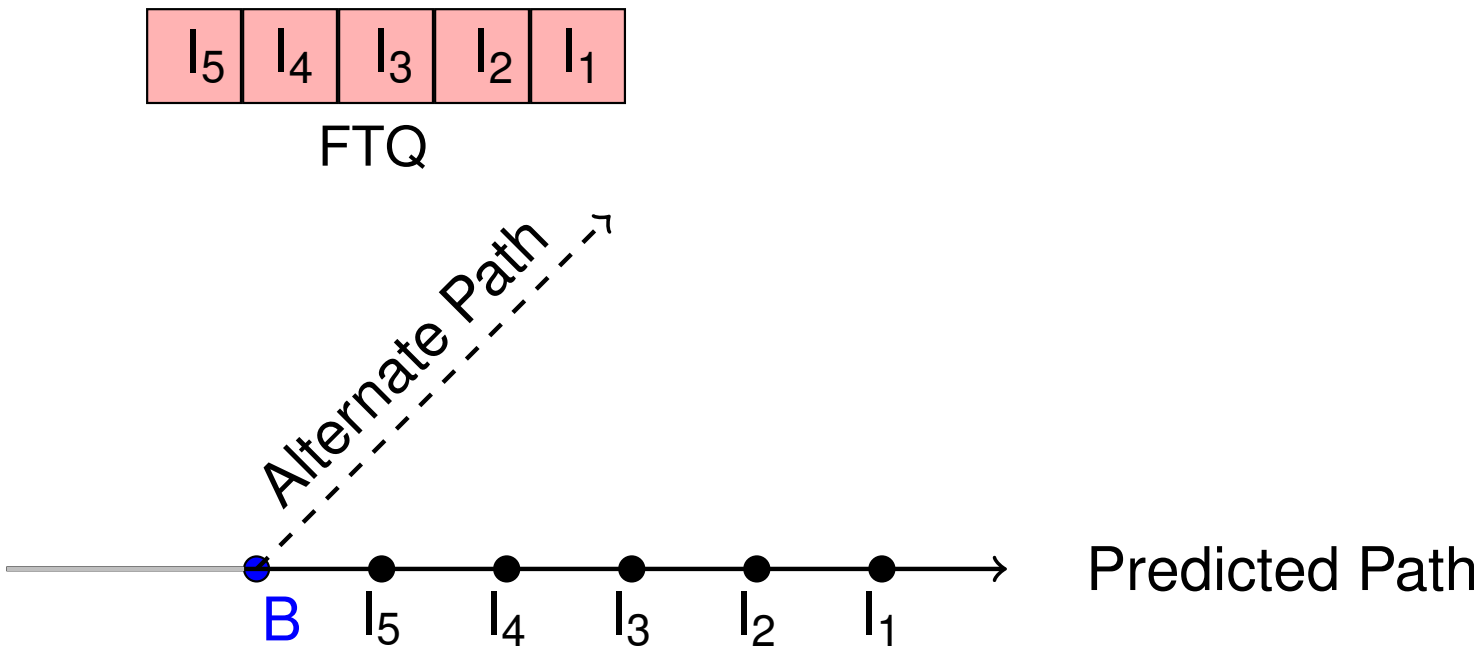
- FTQ is cleared on a branch misprediction

BACKGROUND & MOTIVATION

FTQ BEHAVIOR ON BRANCH MISS

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction

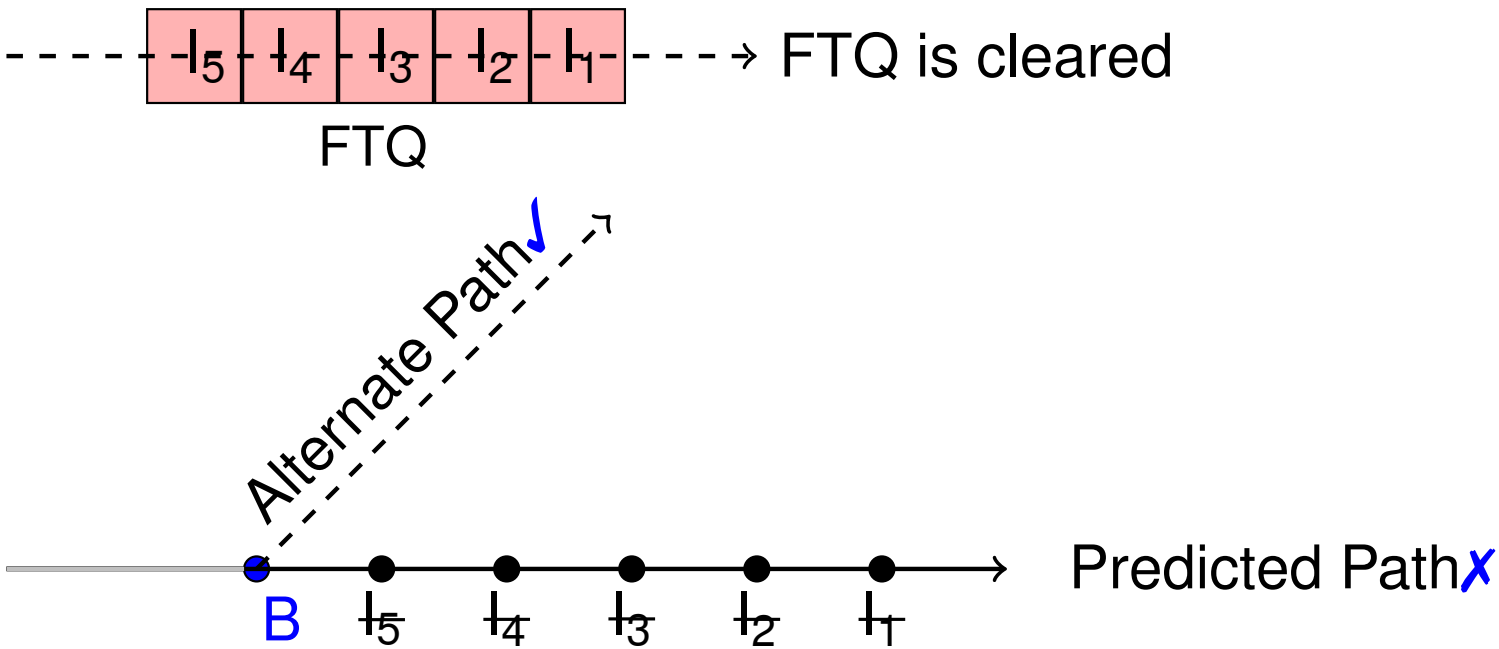


BACKGROUND & MOTIVATION

FTQ BEHAVIOR ON BRANCH MISS

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction



Background & Motivation

FTQ behavior on branch miss

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction

→ What if the correct path was always in the μ-op cache after a pipeline flush due to branch misprediction?

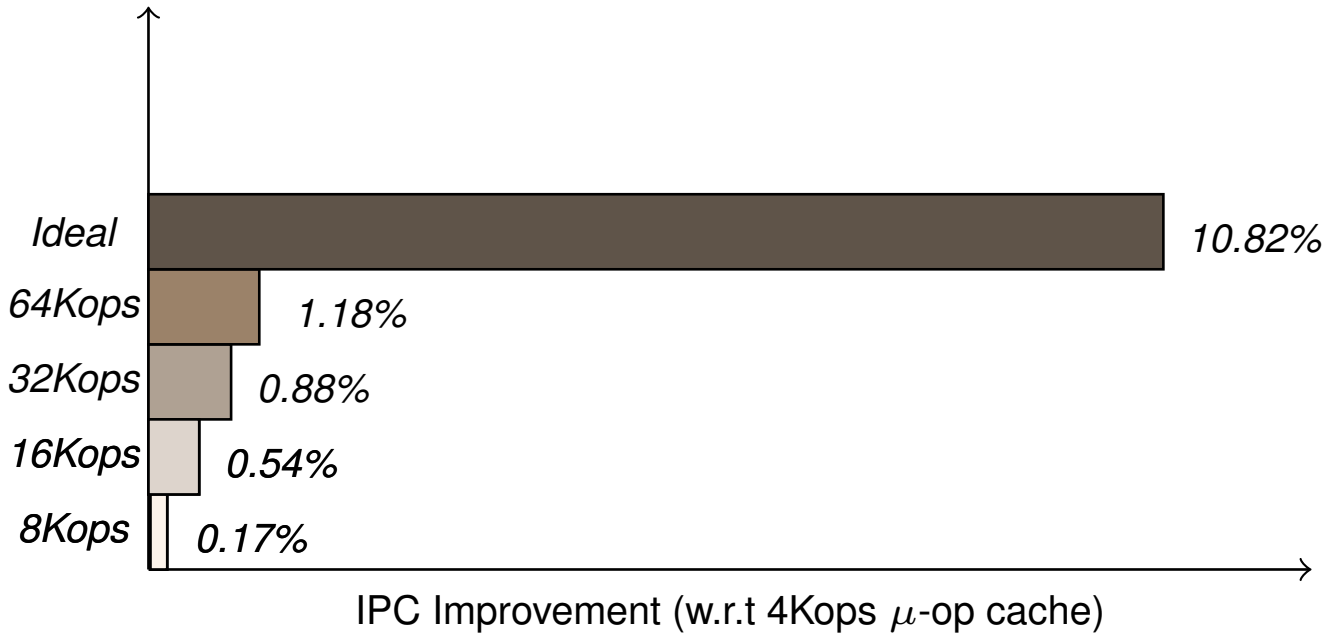
Background & Motivation

FTQ behavior on branch miss

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction

→ What if the correct path was always in the μ-op cache after a pipeline flush due to branch misprediction?



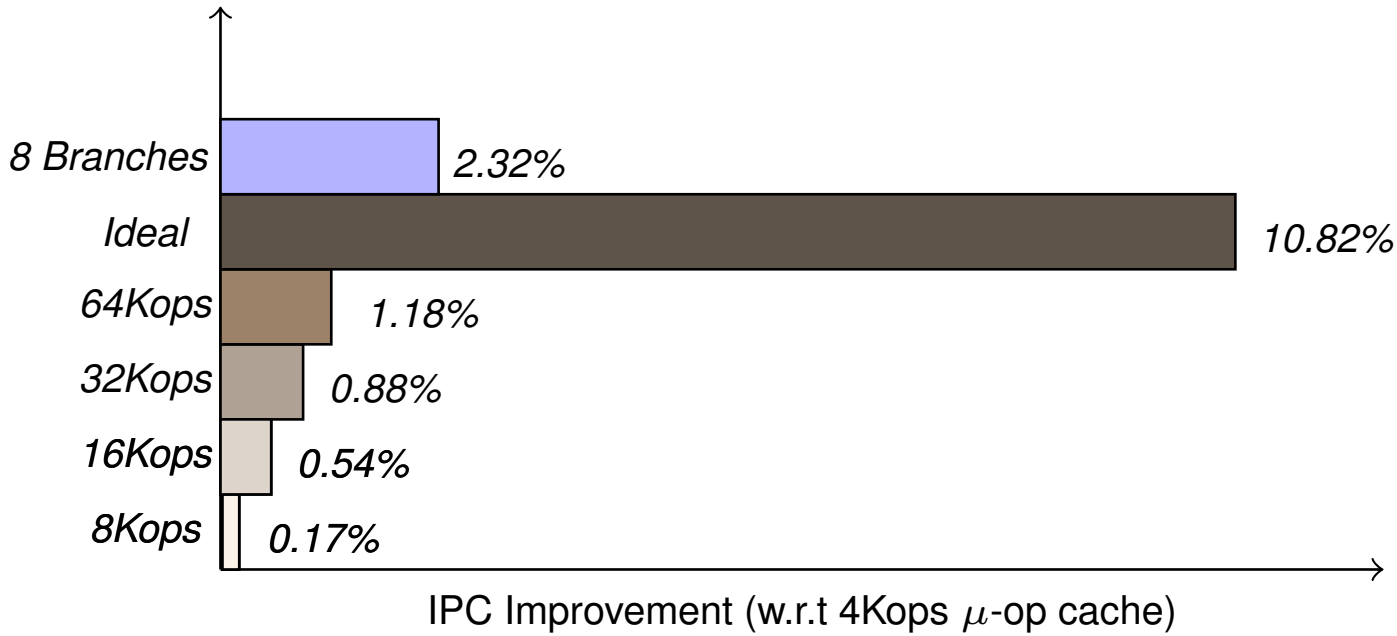
Background & Motivation

FTQ behavior on branch miss

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction

→ What if the correct path was always in the μ-op cache after a pipeline flush due to branch misprediction?



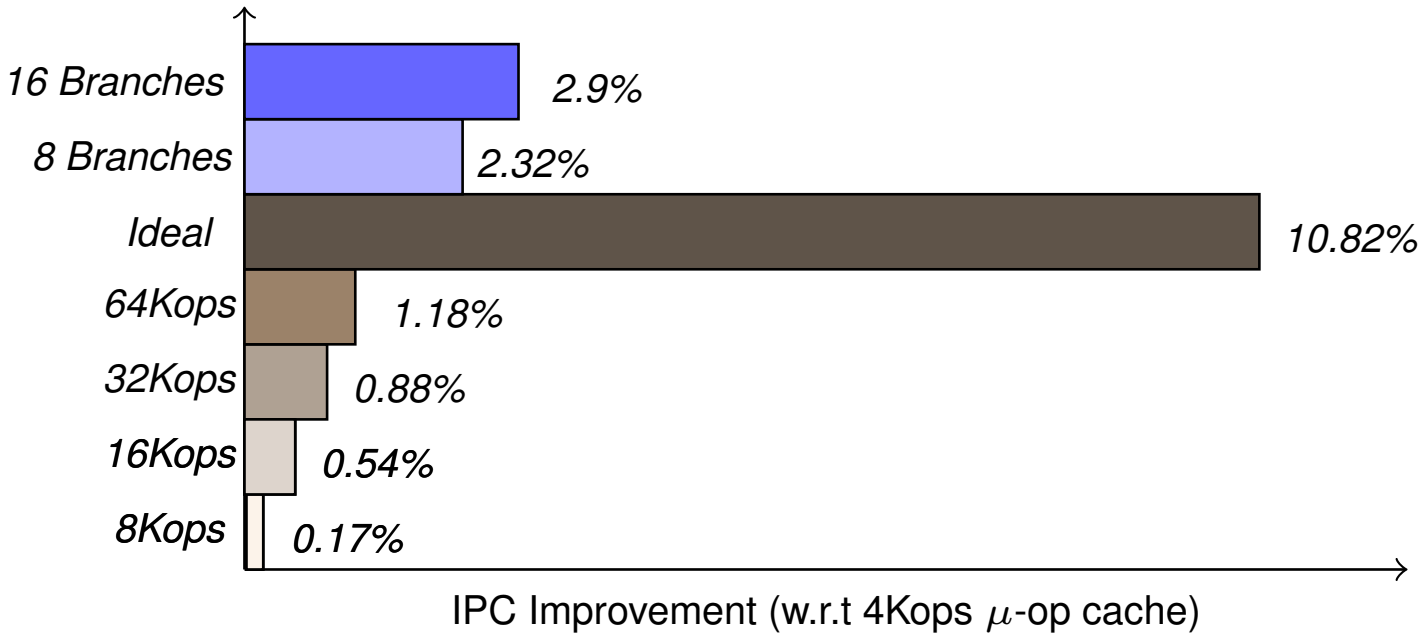
Background & Motivation

FTQ behavior on branch miss

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction

→ What if the correct path was always in the μ-op cache after a pipeline flush due to branch misprediction?



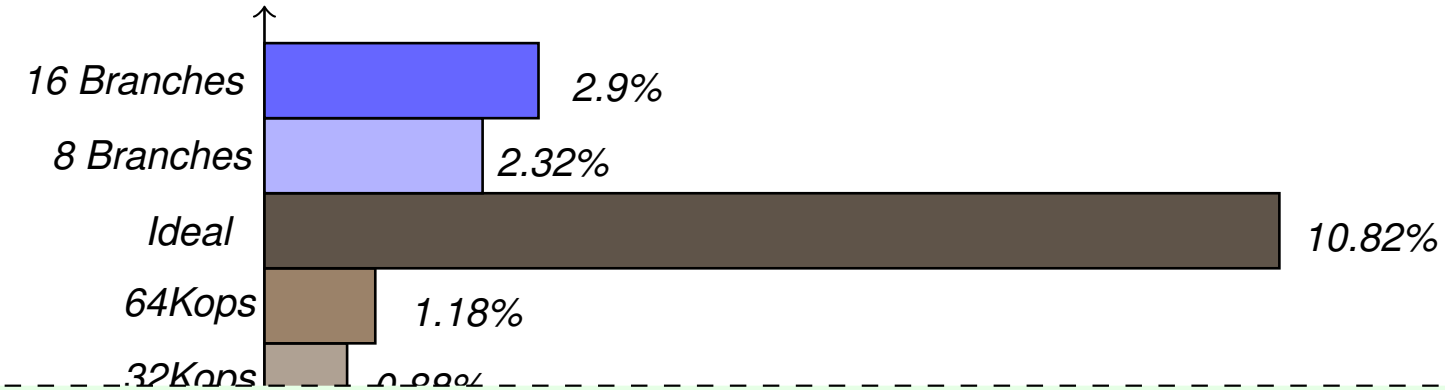
Background & Motivation

FTQ behavior on branch miss

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction

→ What if the correct path was always in the μ-op cache after a pipeline flush due to branch misprediction?



Key Insight:

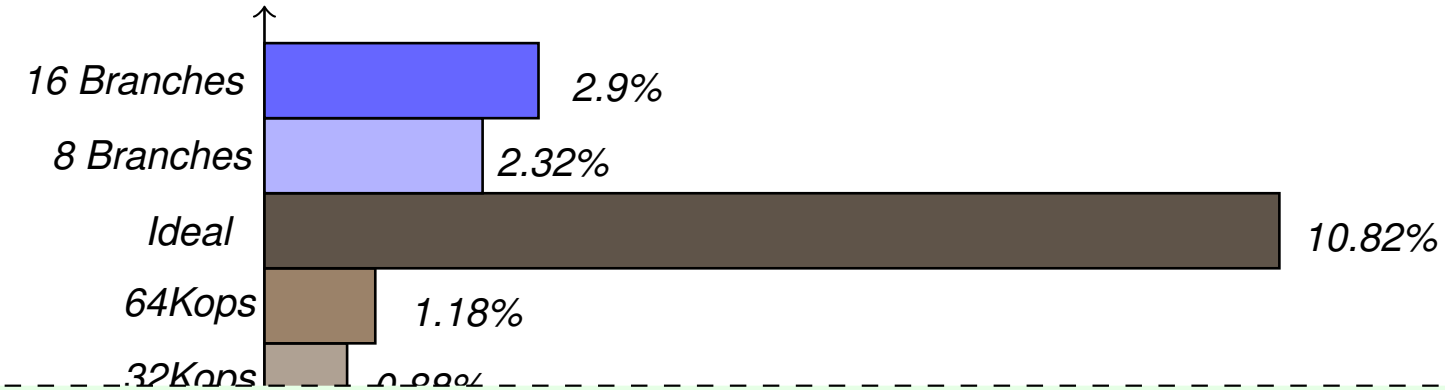
Background & Motivation

FTQ behavior on branch miss

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction

→ What if the correct path was always in the μ-op cache after a pipeline flush due to branch misprediction?



Key Insight:

1. *FTQ is unable to hide the fetch latency on branch misprediction*

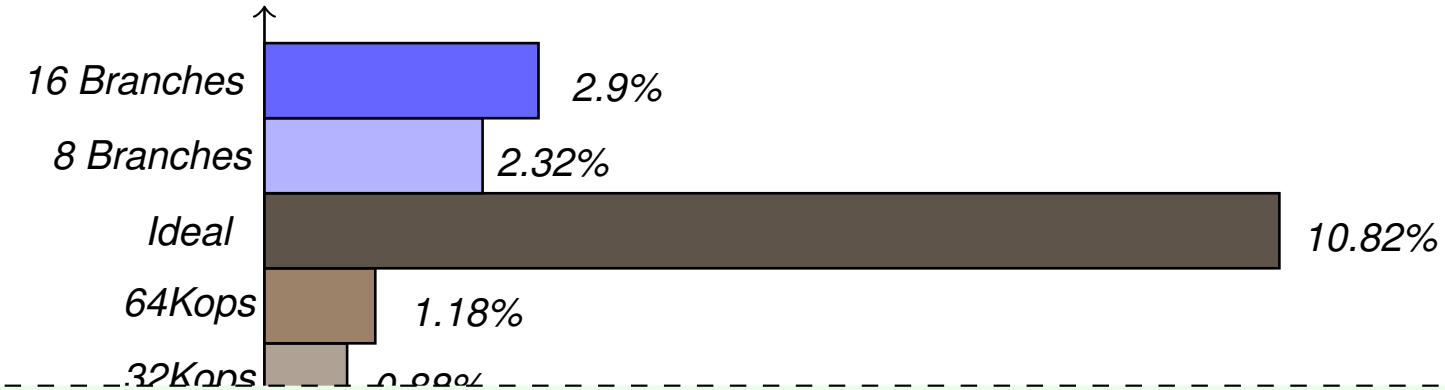
BACKGROUND & MOTIVATION

FTQ BEHAVIOR ON BRANCH MISS

→ FTQ is unable to hide the L1I miss latency on branch misprediction

- FTQ is cleared on a branch misprediction

→ What if the correct path was always in the μ-op cache after a pipeline flush due to branch misprediction?



Key Insight:

1. *FTQ is unable to hide the fetch latency on branch misprediction*

2. *Focusing on few but critical instructions can provide significant performance benefits*

Outline

1. Overview
2. Background & Motivation
3. UCP
4. Methodology & Results
5. Conclusions

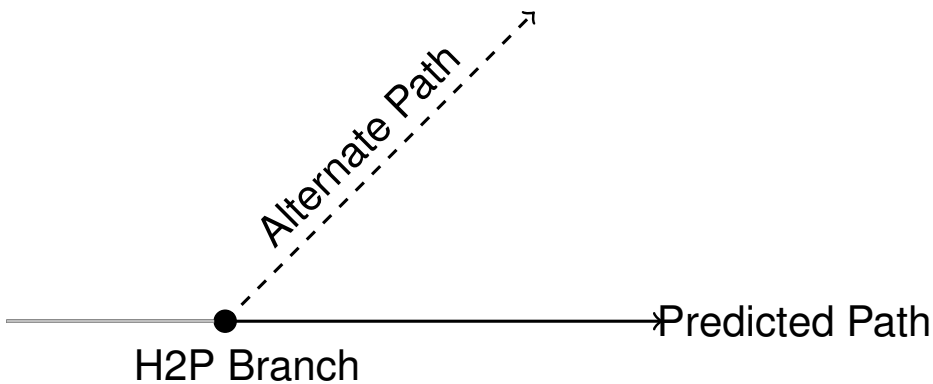
UCP: Overview

① Identifies a hard-to-predict conditional branch (H2P)

UCP

UCP: Overview

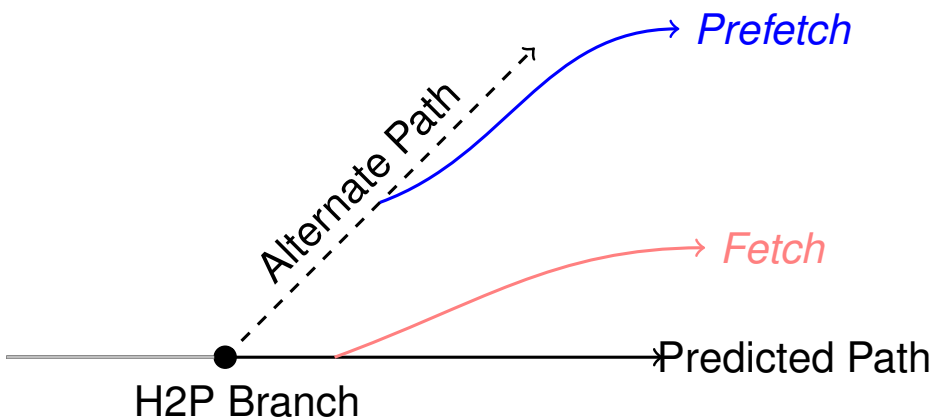
1. Identifies a hard-to-predict conditional branch (H2P)
2. On a H2P begin generating addresses on alternate path (alternate path)



UCP

UCP: Overview

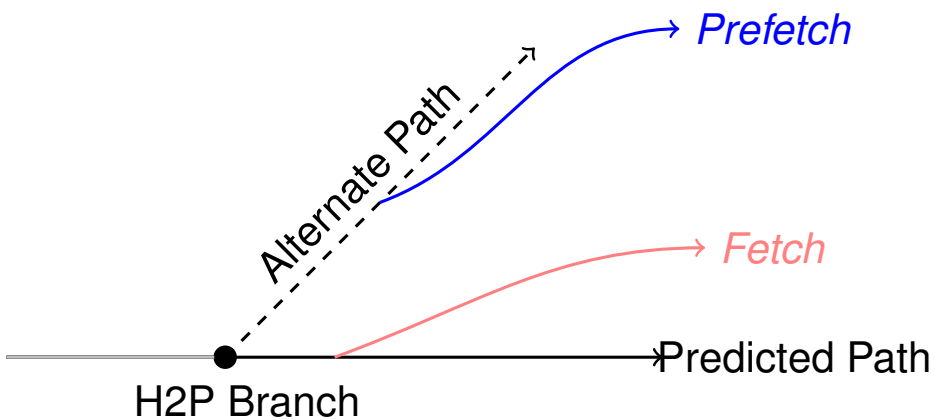
1. Identifies a hard-to-predict conditional branch (H2P)
2. On a H2P begin generating addresses on alternate path (alternate path)
3. Generated addresses on alternate path are prefetched in parallel to predicted path fetch



UCP

UCP: Overview

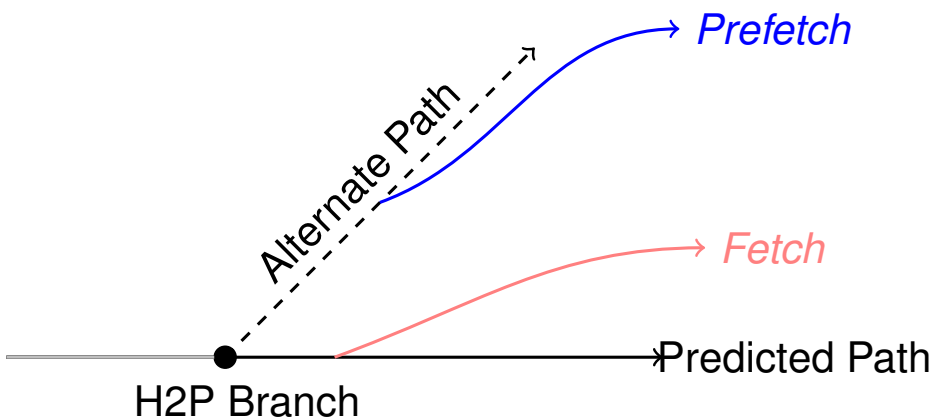
1. Identifies a hard-to-predict conditional branch (H2P)
2. On a H2P begin generating addresses on alternate path (alternate path)
3. Generated addresses on alternate path are prefetched in parallel to predicted path fetch
4. Prefetched instructions are decoded and inserted in the μ-op cache



UCP

UCP: Overview

1. Identifies a hard-to-predict conditional branch (H2P)
2. On a H2P begin generating addresses on alternate path (alternate path)
3. Generated addresses on alternate path are prefetched in parallel to predicted path fetch
4. Prefetched instructions are decoded and inserted in the μ-op cache



Key Idea: *Keep the alternate path in the μ-op cache for H2P branches*

UCP

UCP: H2P Branch Detection

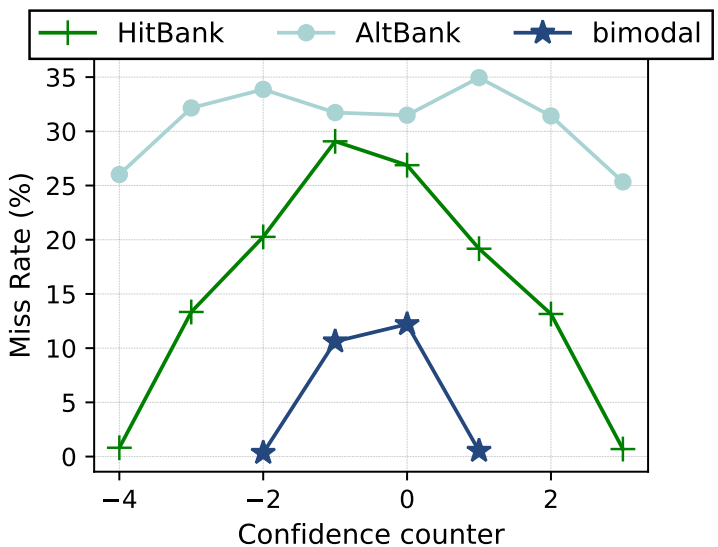
→ H2P Branch: a branch which has high chance of being mispredicted

UCP

UCP: H2P Branch Detection

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

→ TAGE-Conf[2]



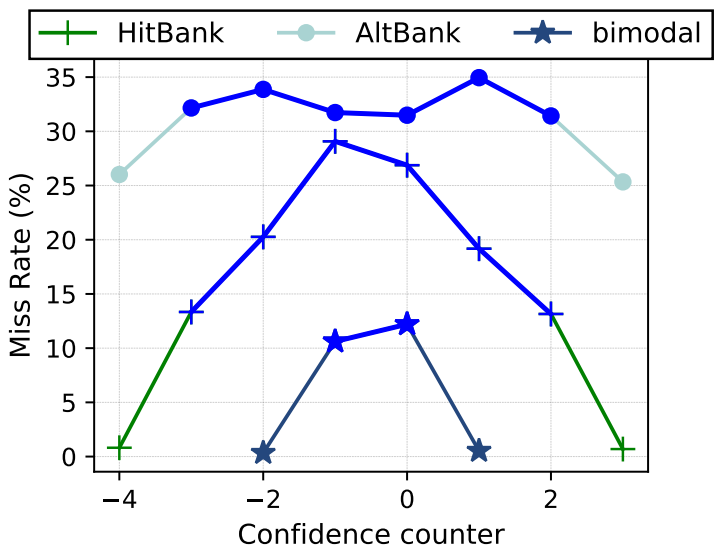
UCP

UCP: H2P Branch Detection

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

→ TAGE-Conf^2

- Not saturated predictions from AltBank, HitBank & BiModal



UCP

UCP: H2P Branch Detection

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

→ TAGE-Conf^2

- Not saturated predictions from AltBank, HitBank & BiModal
- Does not consider SC and LP

UCP

UCP: H2P Branch Detection

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

- → TAGE-Conf^2
  - Not saturated predictions from AltBank, HitBank & BiModal
  - Does not consider SC and LP
- → UCP-Conf

UCP

UCP: H2P Branch Detection

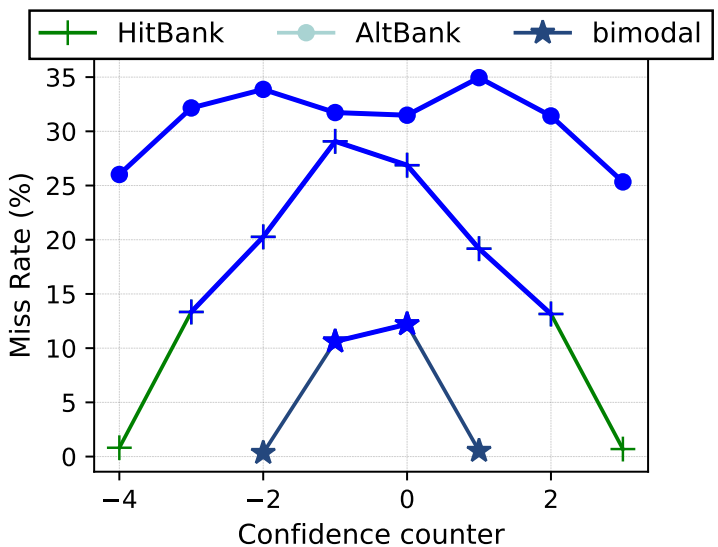
→ H2P Branch: a branch which has high chance of being mispredicted

→ TAGE-Conf^2

- Not saturated predictions from AltBank, HitBank & BiModal
- Does not consider SC and LP

→ UCP-Conf

- All predictions from AltBanks shows high miss rate



UCP

UCP: H2P Branch Detection

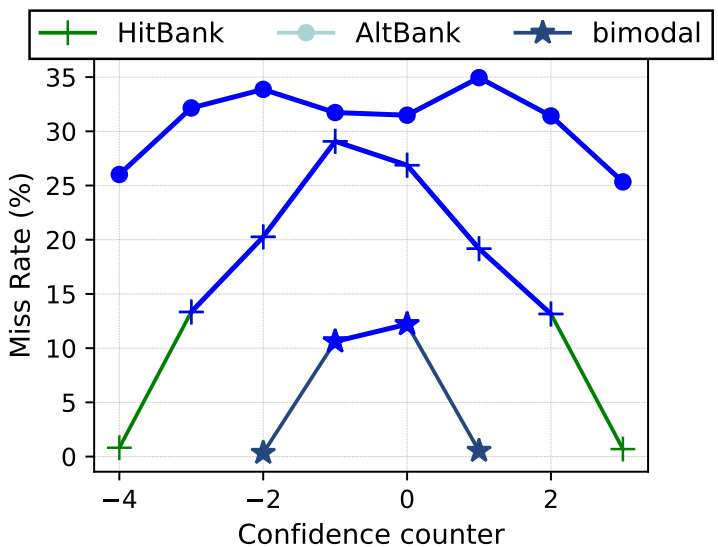
→ H2P Branch: a branch which has high chance of being mispredicted

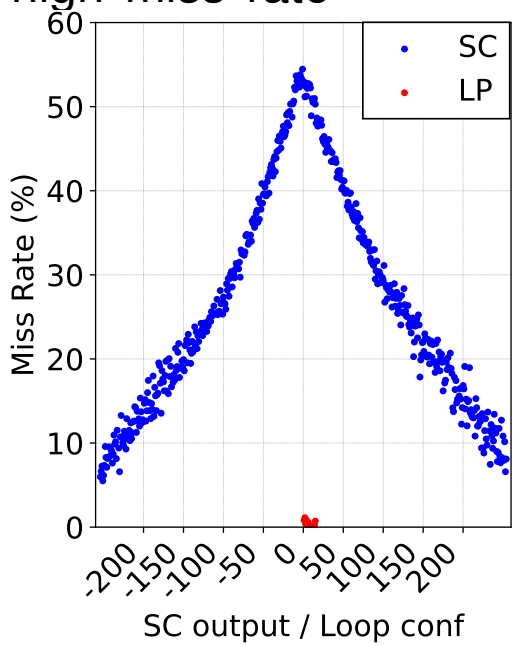
→ TAGE-Conf^2

- Not saturated predictions from AltBank, HitBank & BiModal
- Does not consider SC and LP

→ UCP-Conf

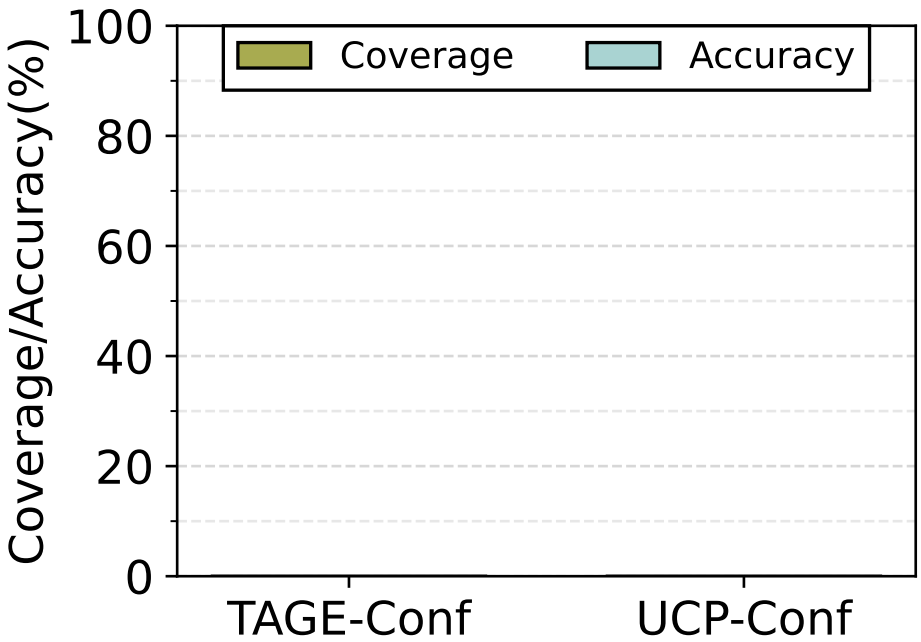
- All predictions from AltBanks shows high miss rate
- SC shows high miss rate





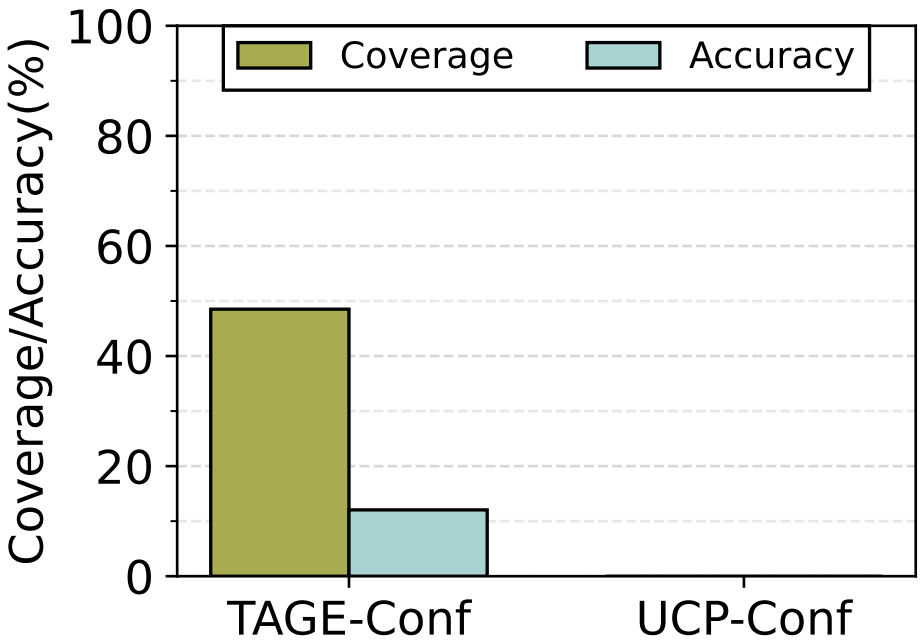
UCP

UCP: H2P Branch Detection



UCP

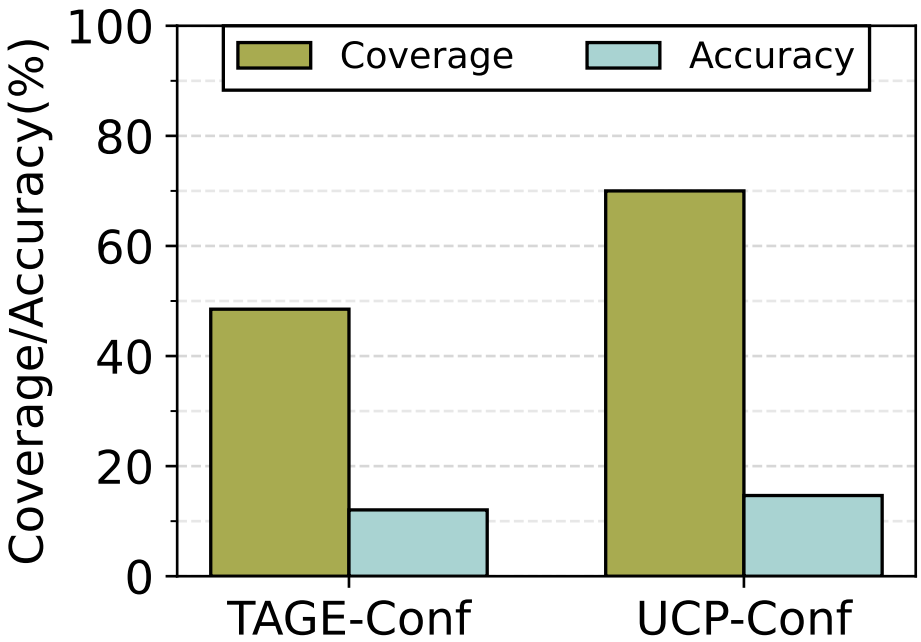
UCP: H2P Branch Detection



→ TAGE-Conf provide 48.5% coverage and 12% accuracy

UCP

UCP: H2P Branch Detection



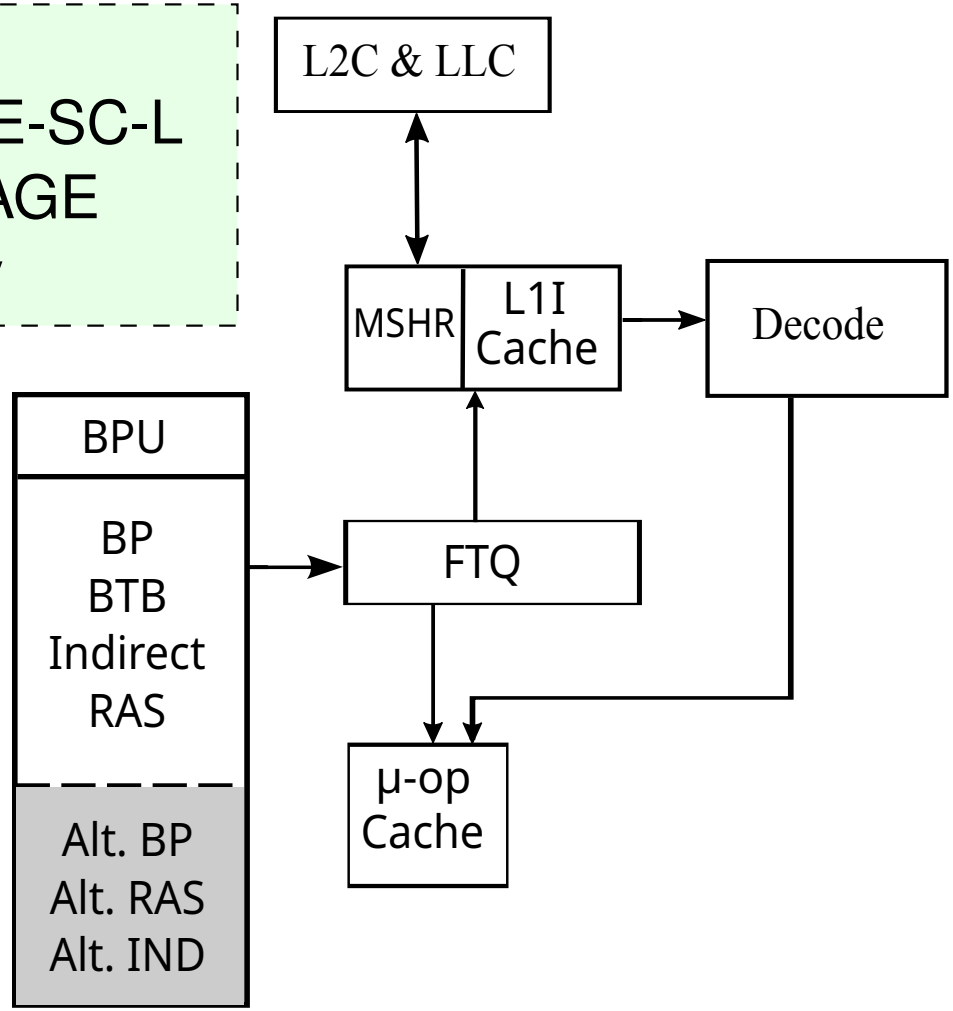
→ TAGE-Conf provide 48.5% coverage and 12% accuracy

→ UCP-Conf improve coverage to 70% and accuracy to 14.66%

UCP

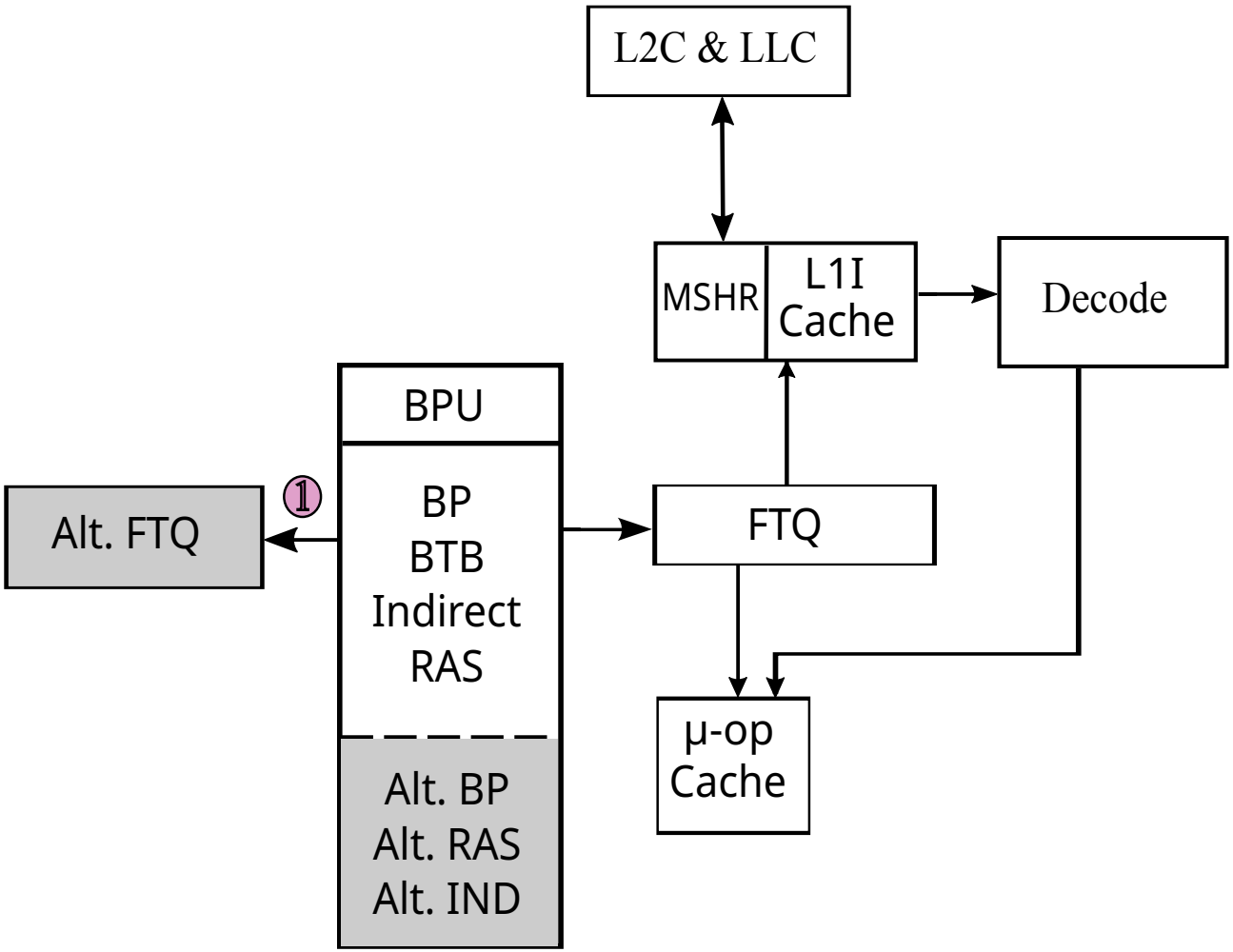
UCP: Design

BTB: Shared
Alt BP: 8KB TAGE-SC-L
Alt IND: 4KB ITTAGE
Alt RAS: 16-entry



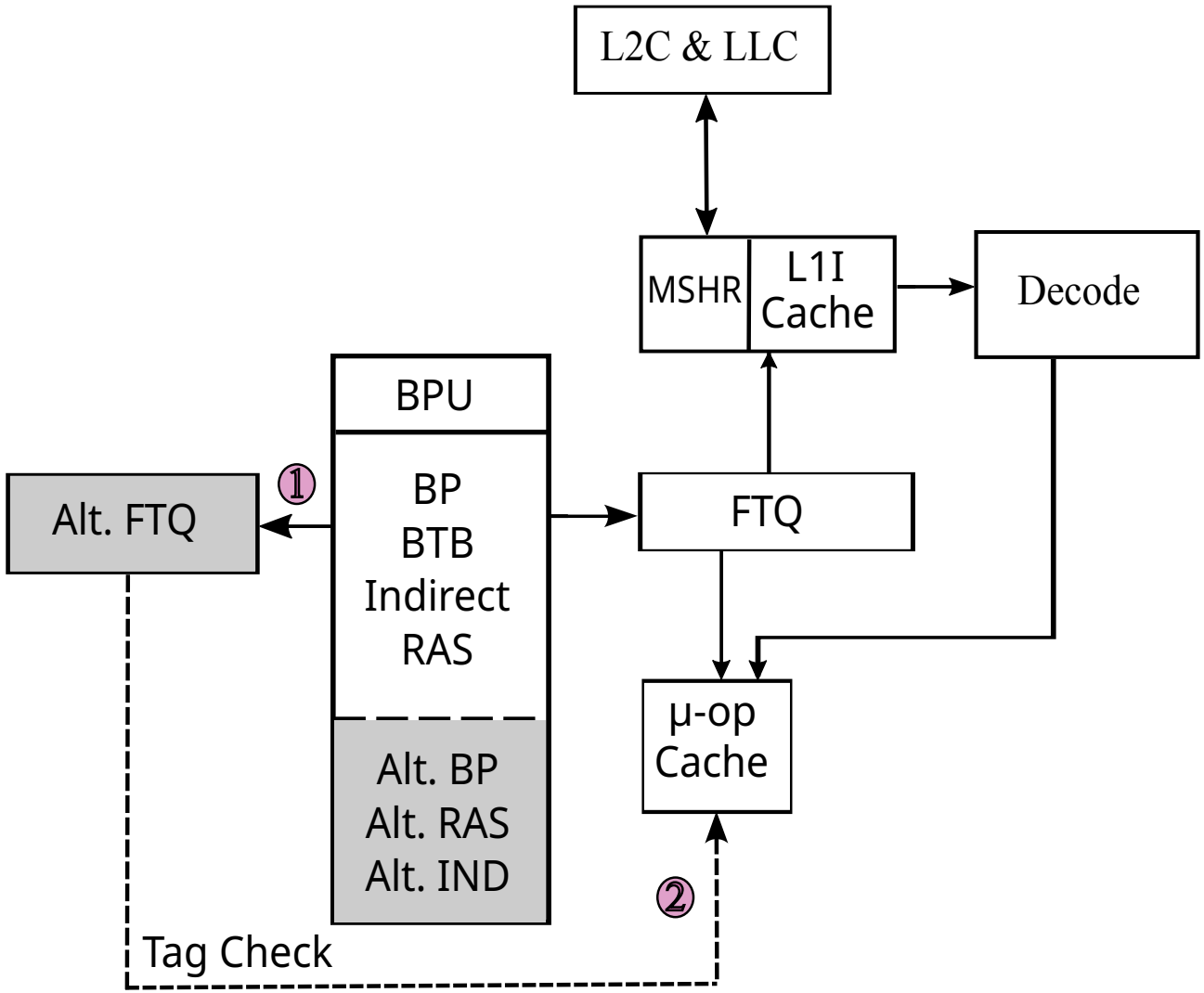
UCP

UCP: Design



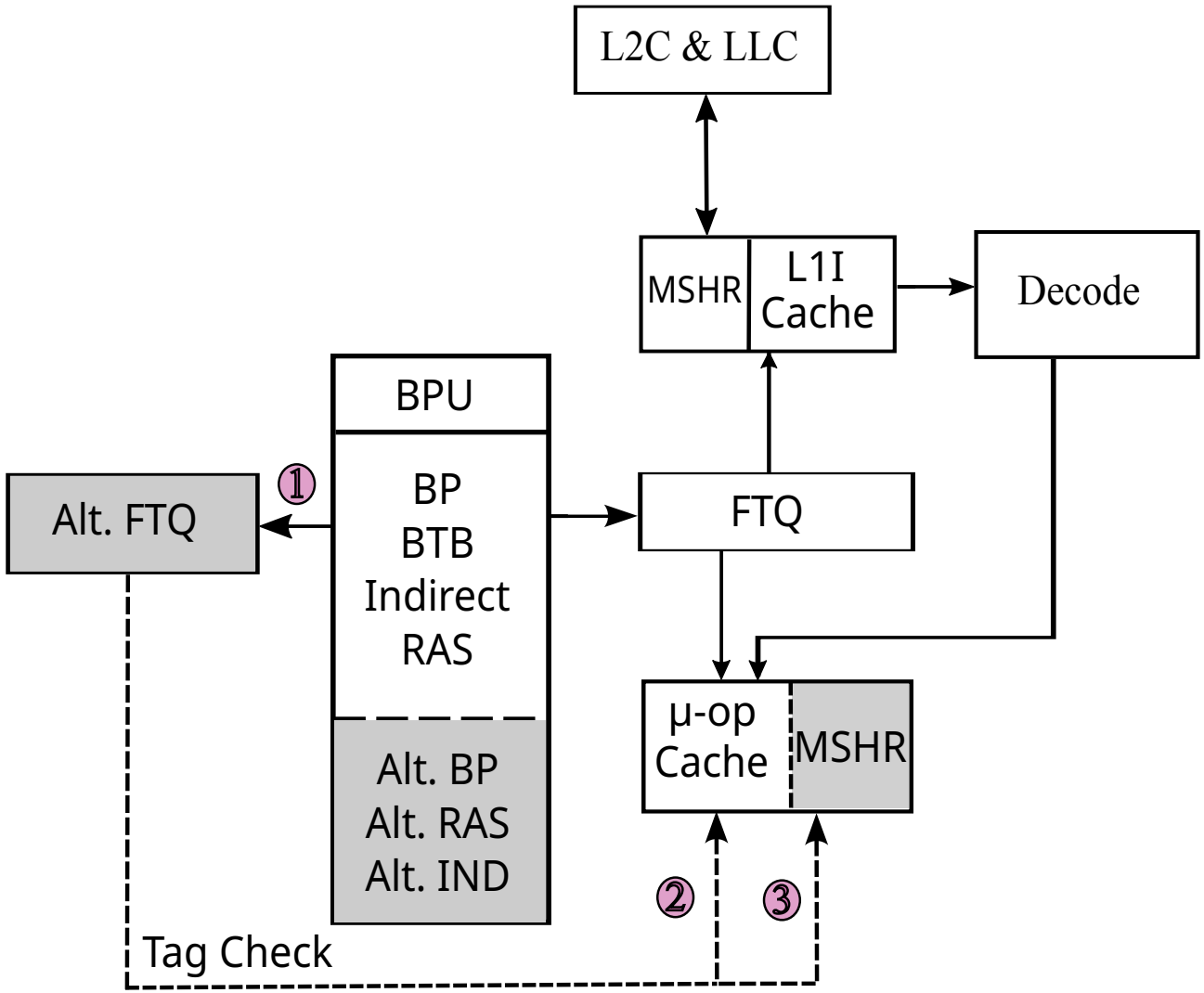
UCP

UCP: Design



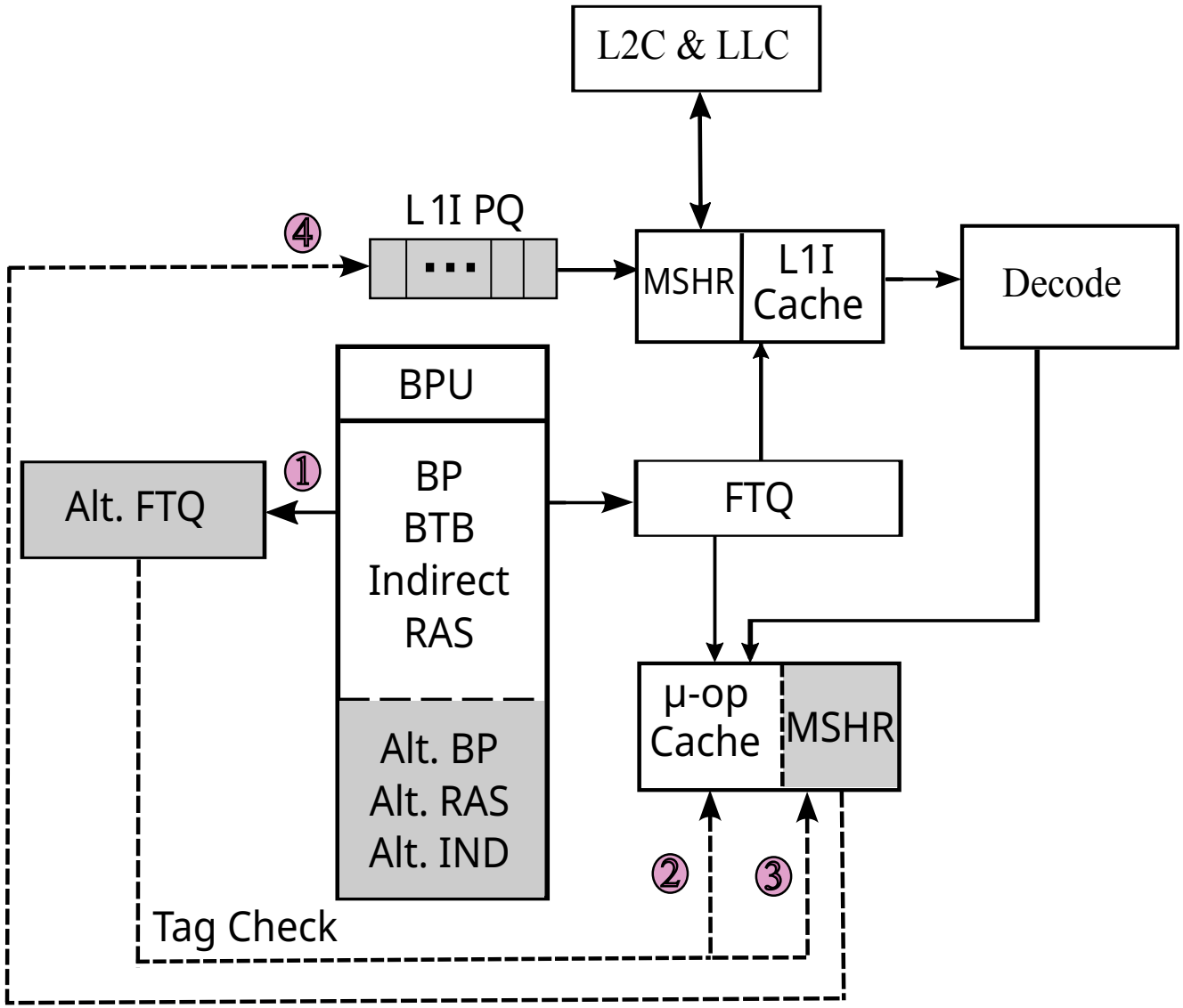
UCP

UCP: Design



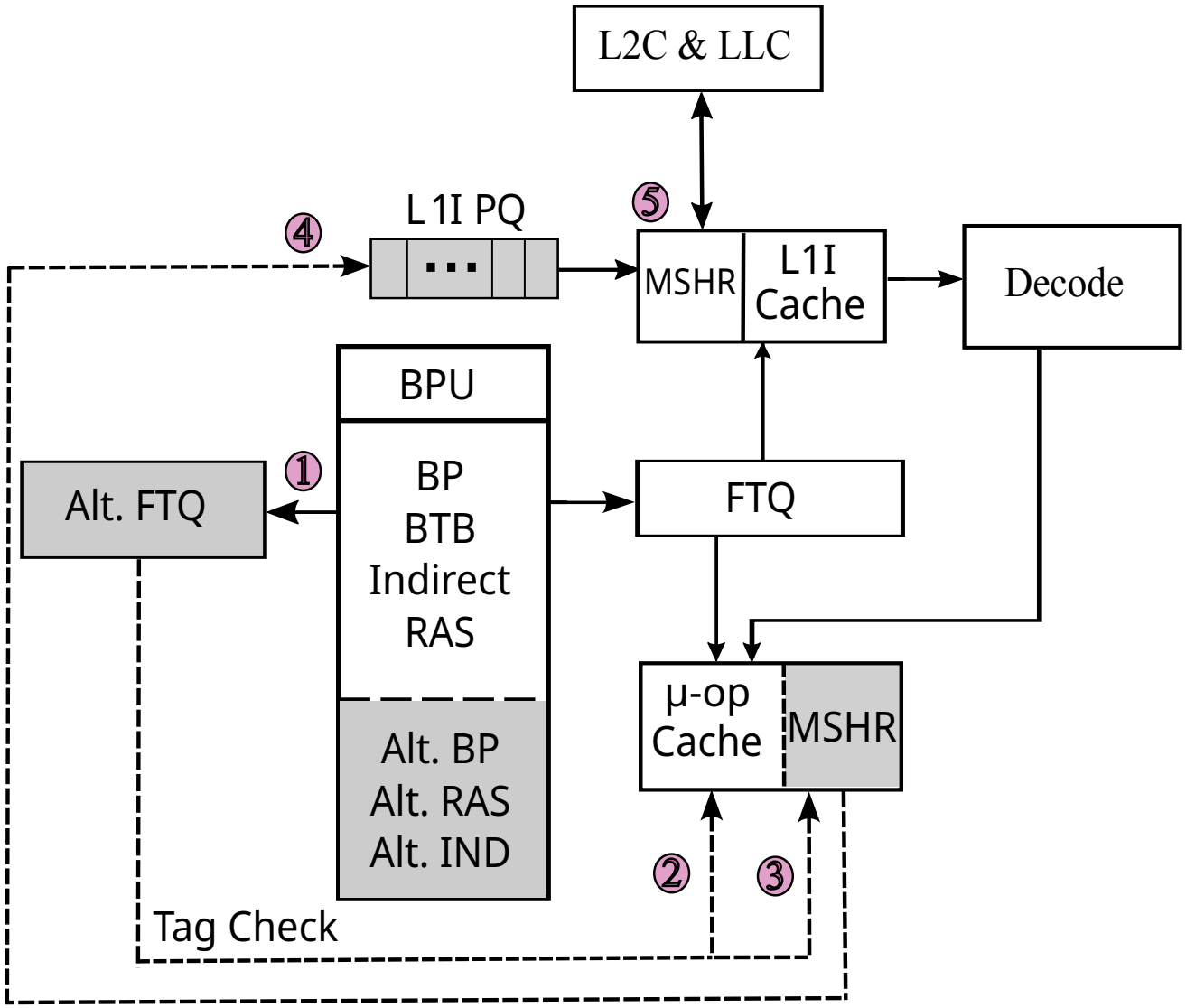
UCP

UCP: Design



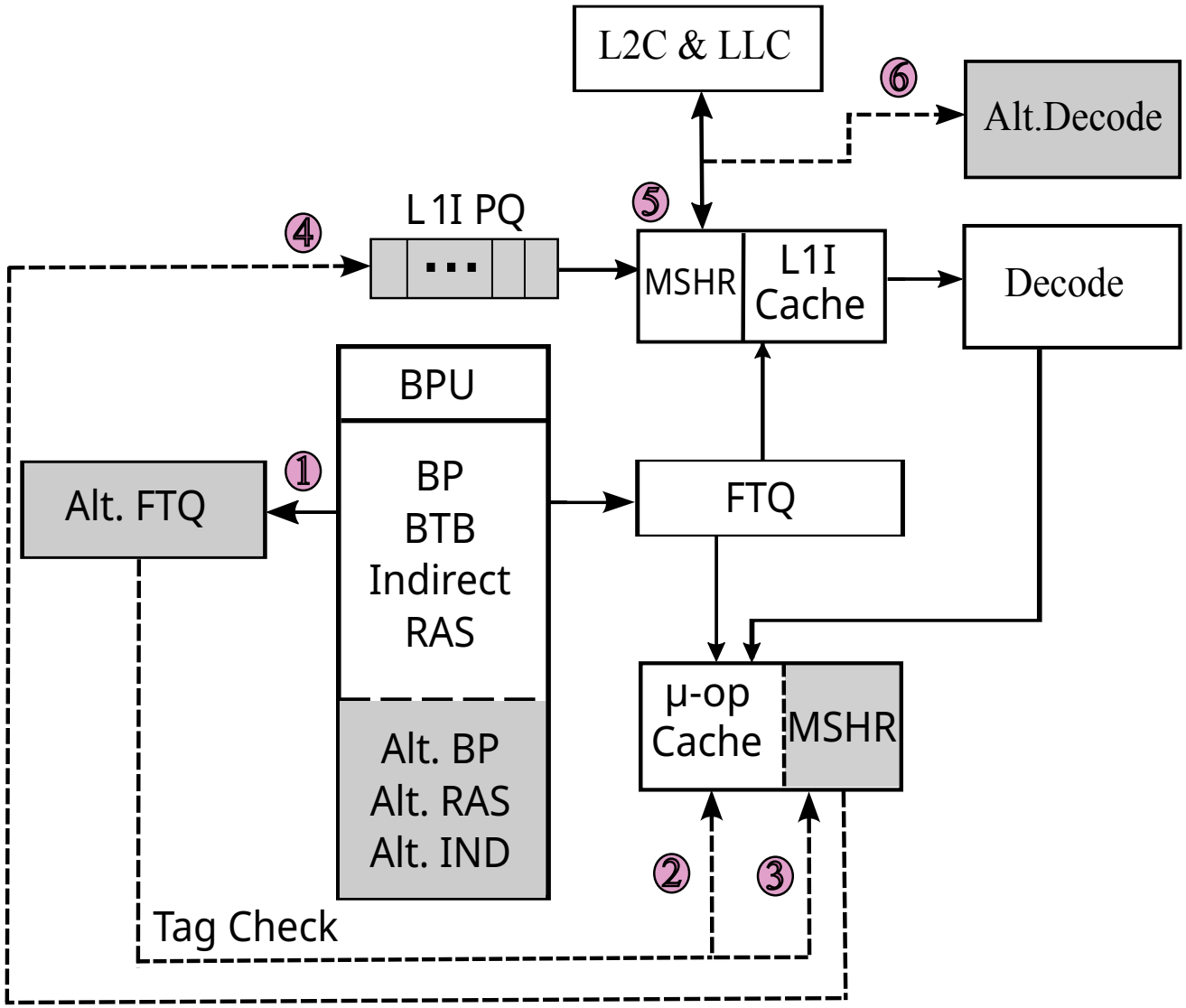
UCP

UCP: Design



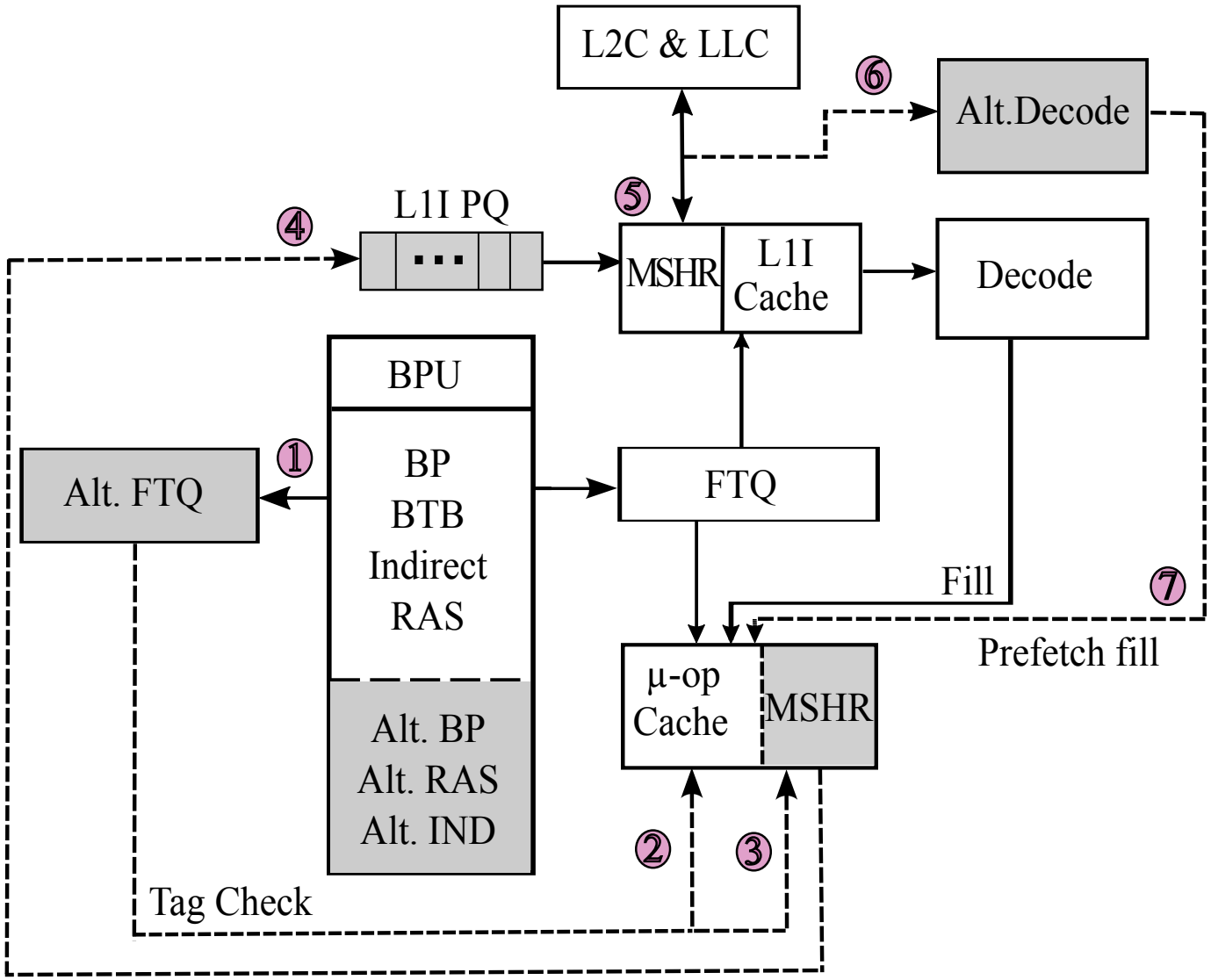
UCP

UCP: Design



UCP

UCP: Design



UCP: Alternate Path Stopping Conditions

→ Stopping alternate path is crucial to prevent μ-op cache pollution

UCP: Alternate Path Stopping Conditions

→ Stopping alternate path is crucial to prevent μ-op cache pollution

→ When a new H2P branch is detected

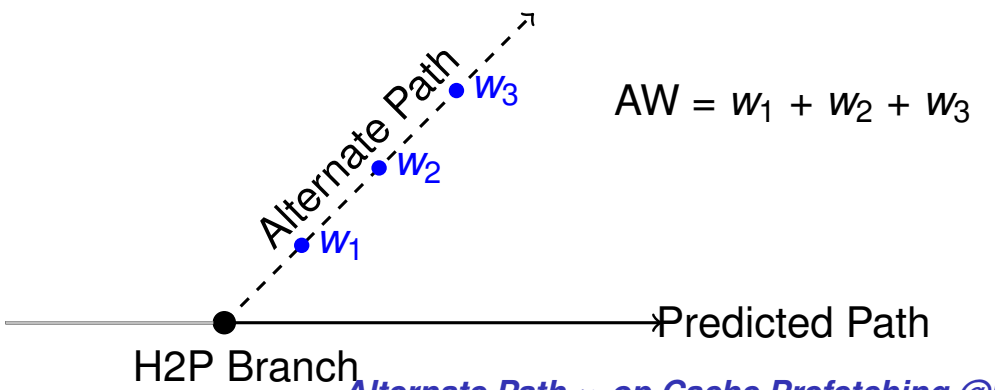
UCP: Alternate Path Stopping Conditions

- → Stopping alternate path is crucial to prevent μ-op cache pollution
- → When a new H2P branch is detected
- → When the alternate path is unlikely to become the correct path

UCP: Alternate Path Stopping Conditions

- → Stopping alternate path is crucial to prevent μ-op cache pollution
- → When a new H2P branch is detected
- → When the alternate path is unlikely to become the correct path
  - BTB miss on predicted taken branch on the alternate path

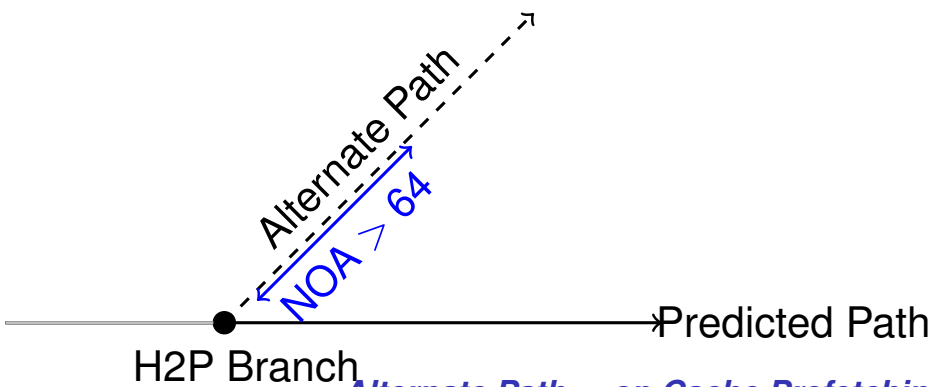
- → Stopping alternate path is crucial to prevent μ-op cache pollution
- → When a new H2P branch is detected
- → When the alternate path is unlikely to become the correct path
  - BTB miss on predicted taken branch on the alternate path
  - Weight of each branch on the alternate path is accumulated, high confidence branches have lower weight.



UCP

UCP: Alternate Path Stopping Conditions

- → Stopping alternate path is crucial to prevent μ-op cache pollution
- → When a new H2P branch is detected
- → When the alternate path is unlikely to become the correct path
  - BTB miss on predicted taken branch on the alternate path
  - Weight of each branch on the alternate path is accumulated, high confidence branches have lower weight.
  - Non-branch instructions after a branch are counted. Once the count reaches 64 alternate paths stops in our work



Outline

1. Overview
2. Background & Motivation
3. UCP
4. Methodology & Results
5. Conclusions

METHODOLOGY & RESULTS

SIMULATION SETUP

→ ChampSim + subset (traces showing $\geqq$ 5% improvement with ideal μ-op cache) of CVP traces (2 FP, 97 INT, 73 Crypto and 134 datacenter trace)

Methodology & Results

Simulation Setup

- → ChampSim + subset (traces showing $\geqq$ 5% improvement with ideal μ-op cache) of CVP traces (2 FP, 97 INT, 73 Crypto and 134 datacenter trace)
- → Intel Alder Lake like microarchitecture

Methodology & Results

Simulation Setup

- → ChampSim + subset (traces showing $\geqq$ 5% improvement with ideal μ-op cache) of CVP traces (2 FP, 97 INT, 73 Crypto and 134 datacenter trace)
- → Intel Alder Lake like microarchitecture
- → We execute 100M instructions, 50M warmup and 50M to collect stats

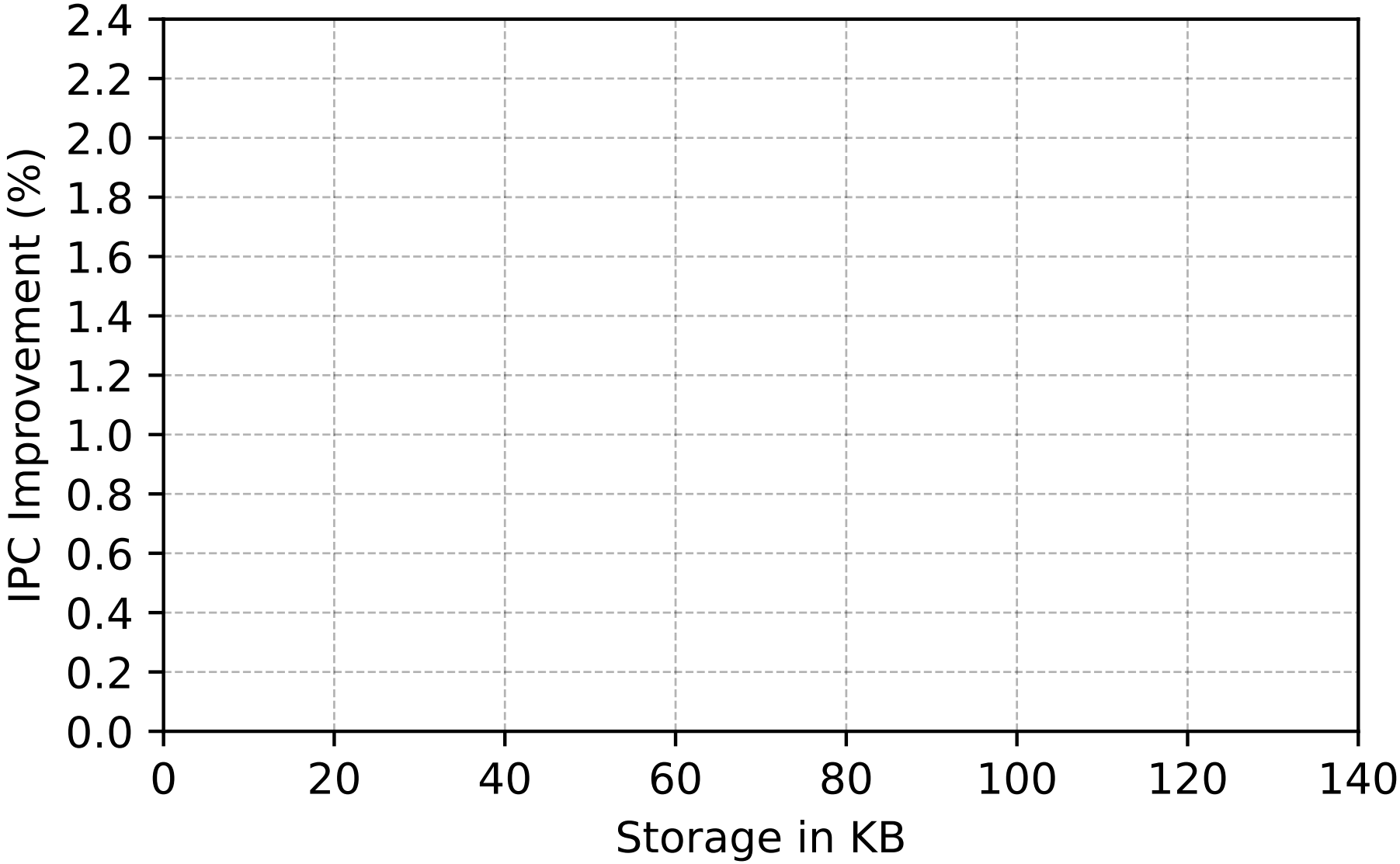
Methodology & Results

Simulation Setup

- → ChampSim + subset (traces showing $\geqq$ 5% improvement with ideal μ-op cache) of CVP traces (2 FP, 97 INT, 73 Crypto and 134 datacenter trace)
- → Intel Alder Lake like microarchitecture
- → We execute 100M instructions, 50M warmup and 50M to collect stats
- → 1 cycle penalty for switching from μ-op cache to L1I cache

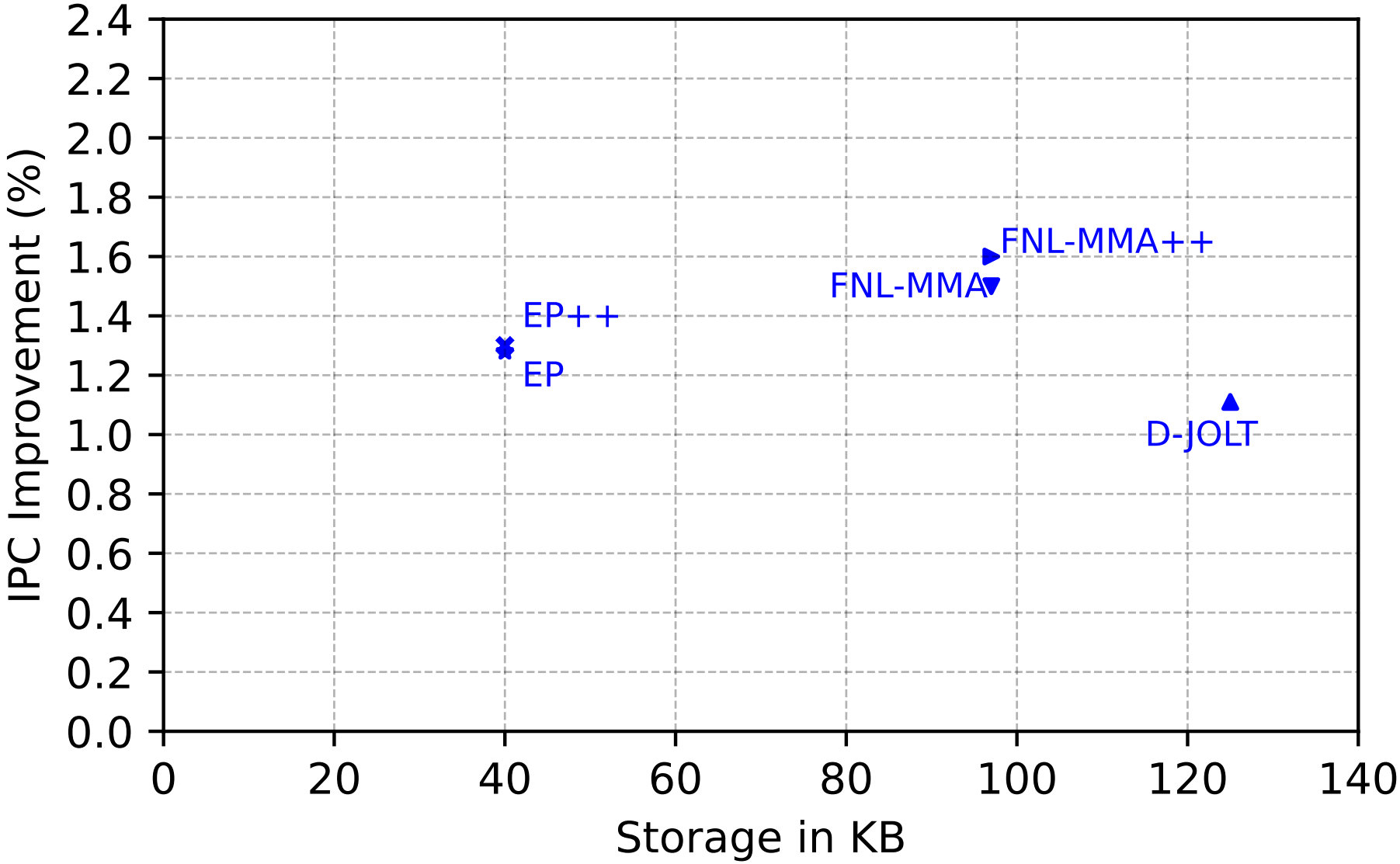
METHODOLOGY & RESULTS

IPC IMPROVEMENTS



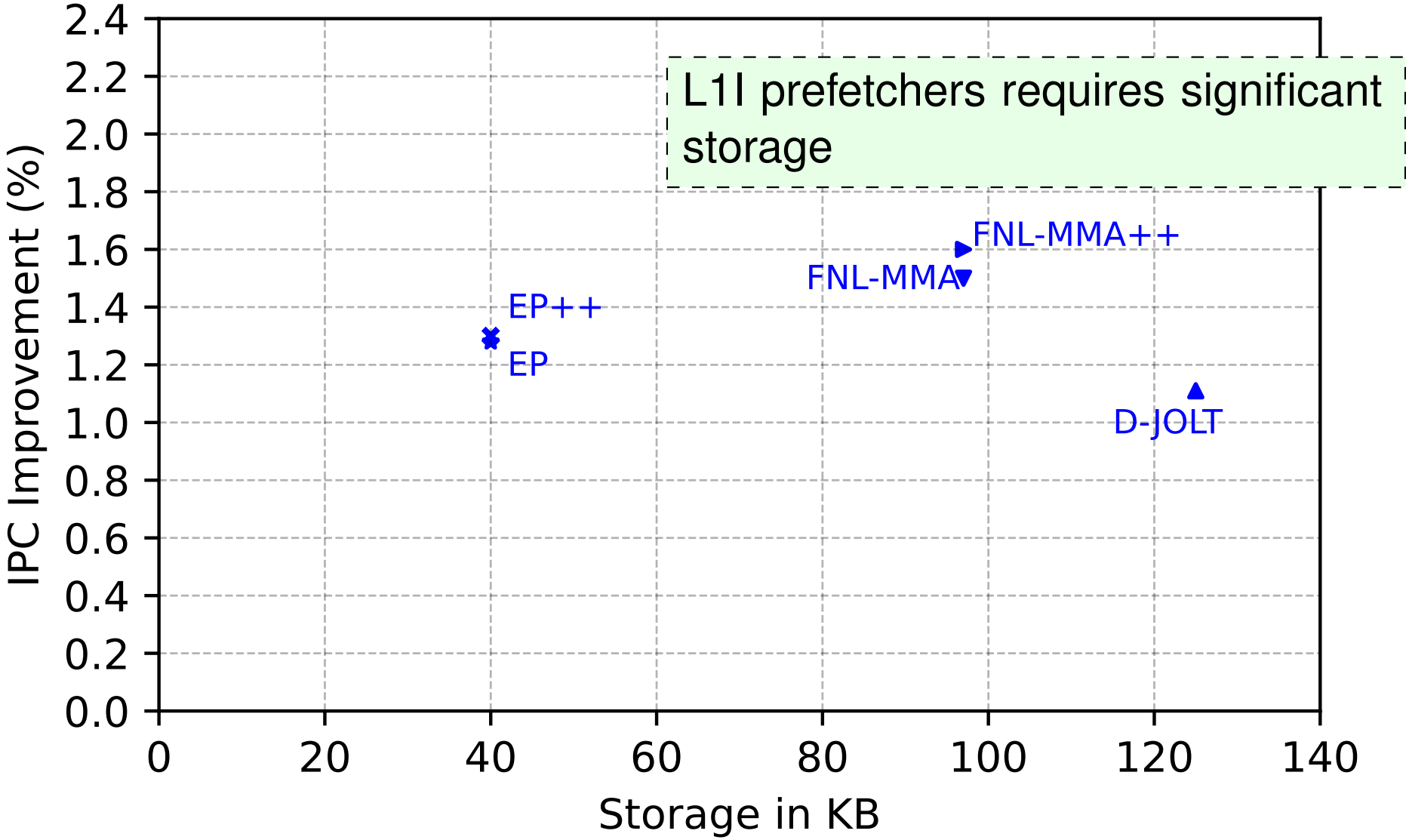
Methodology & Results

IPC Improvements



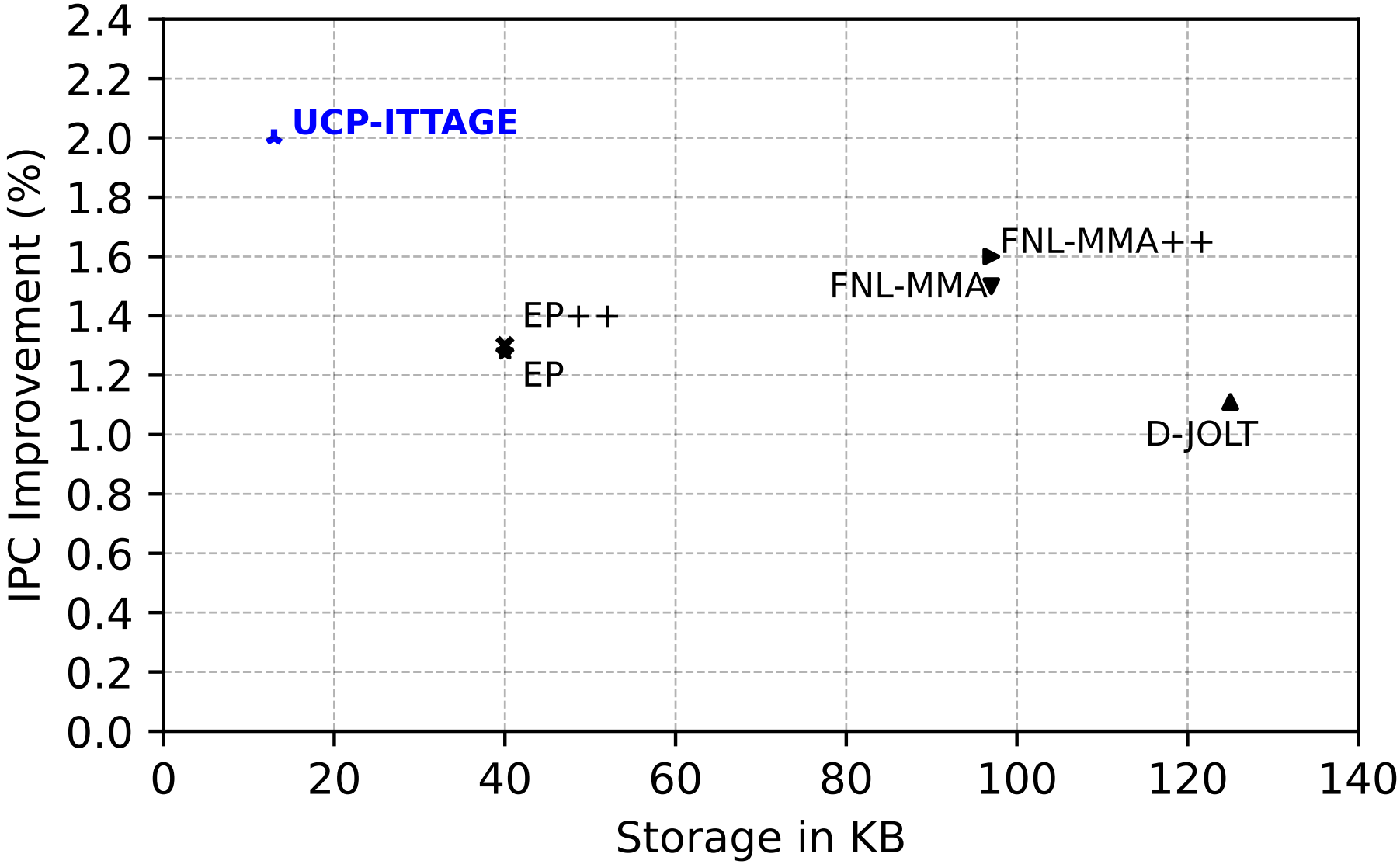
Methodology & Results

IPC Improvements



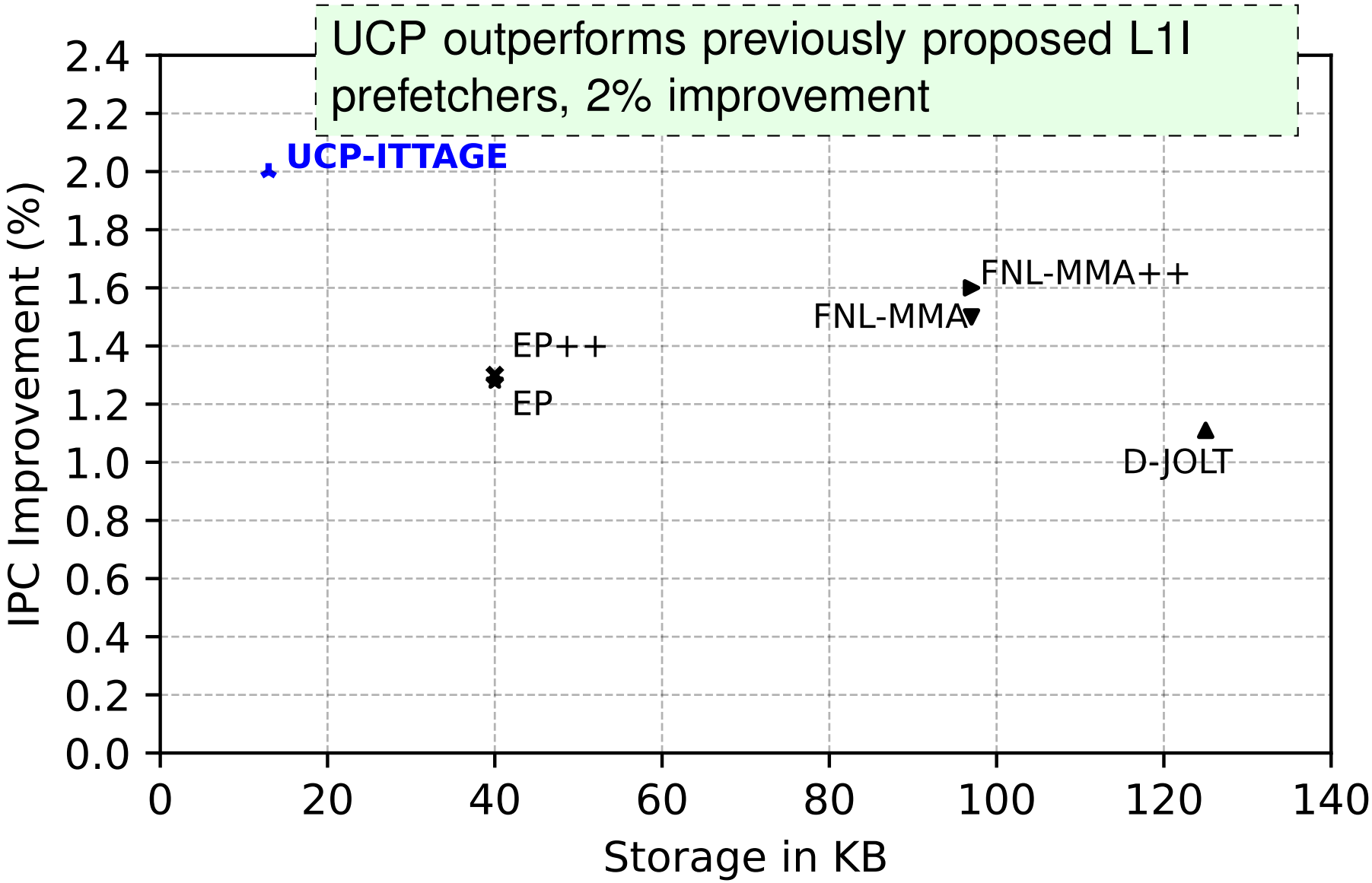
Methodology & Results

IPC Improvements



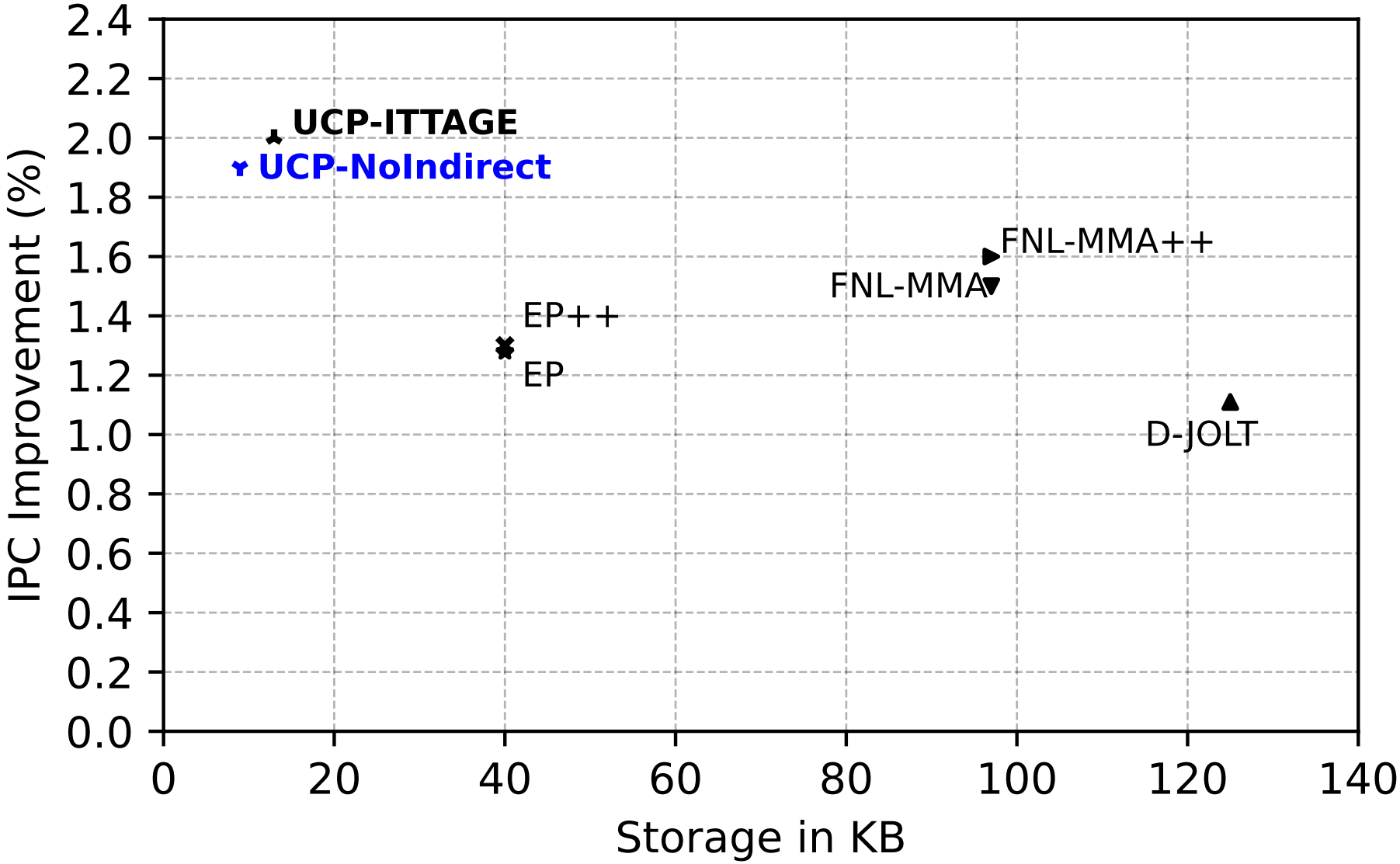
Methodology & Results

IPC Improvements



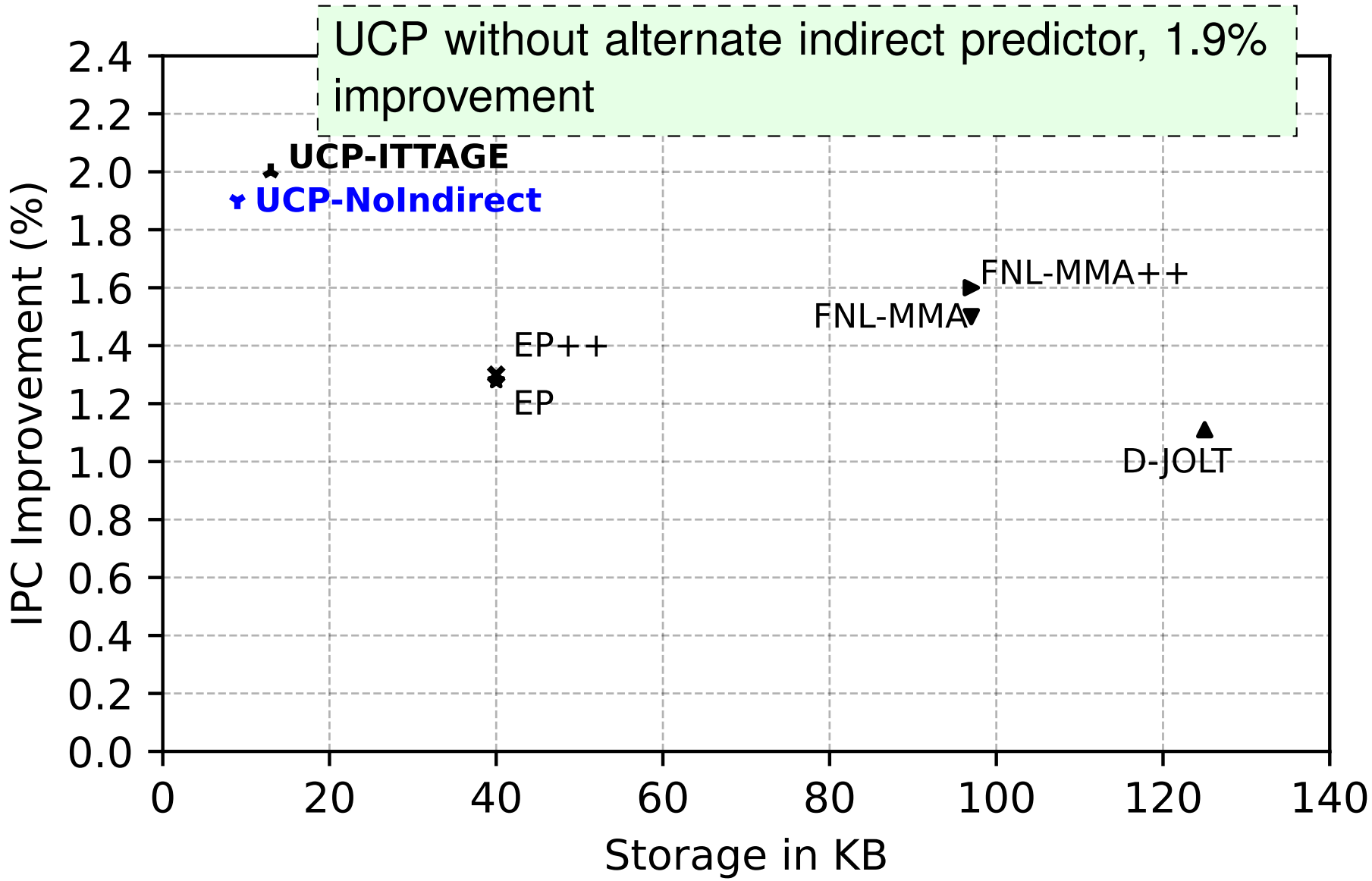
Methodology & Results

IPC Improvements



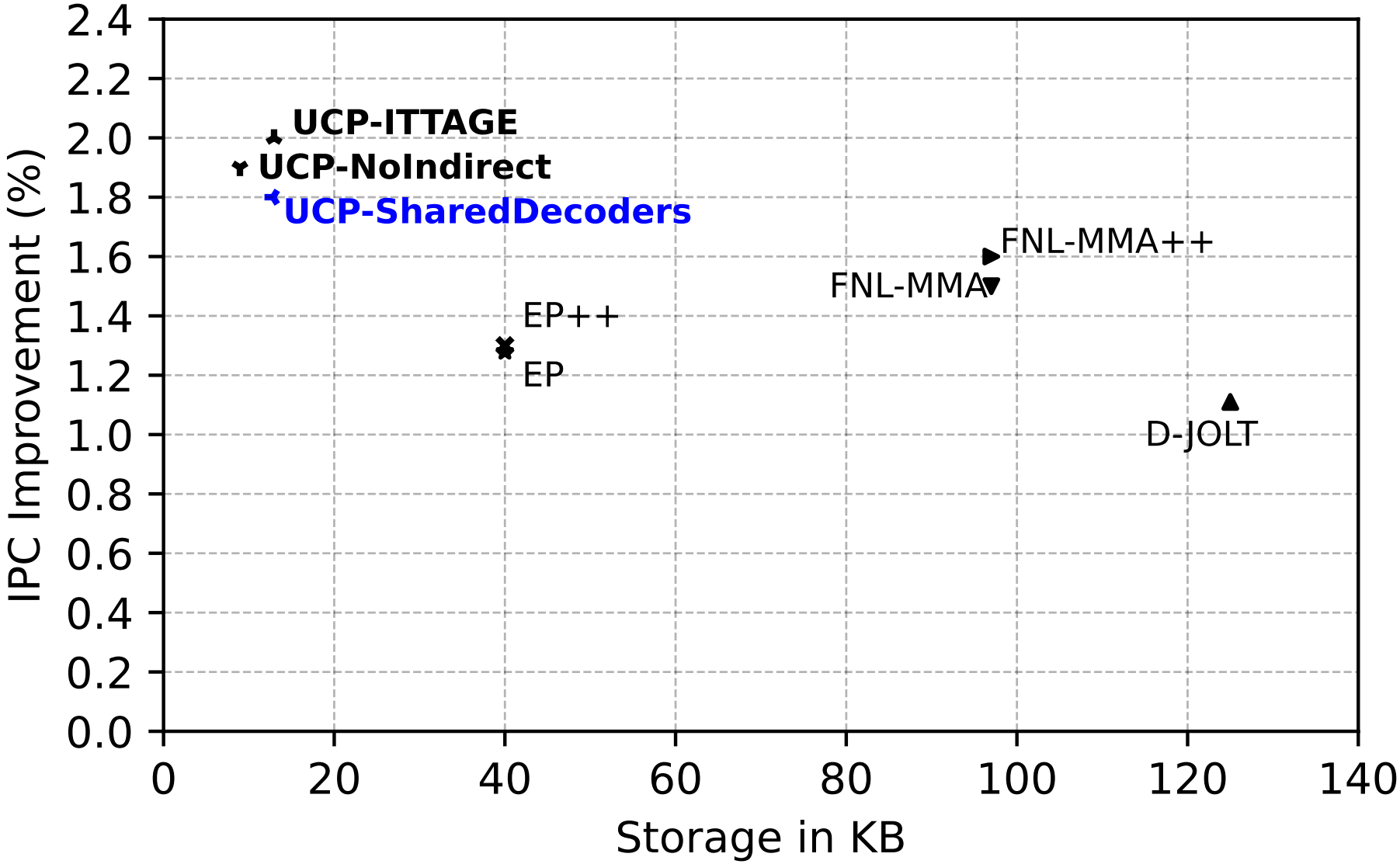
Methodology & Results

IPC Improvements



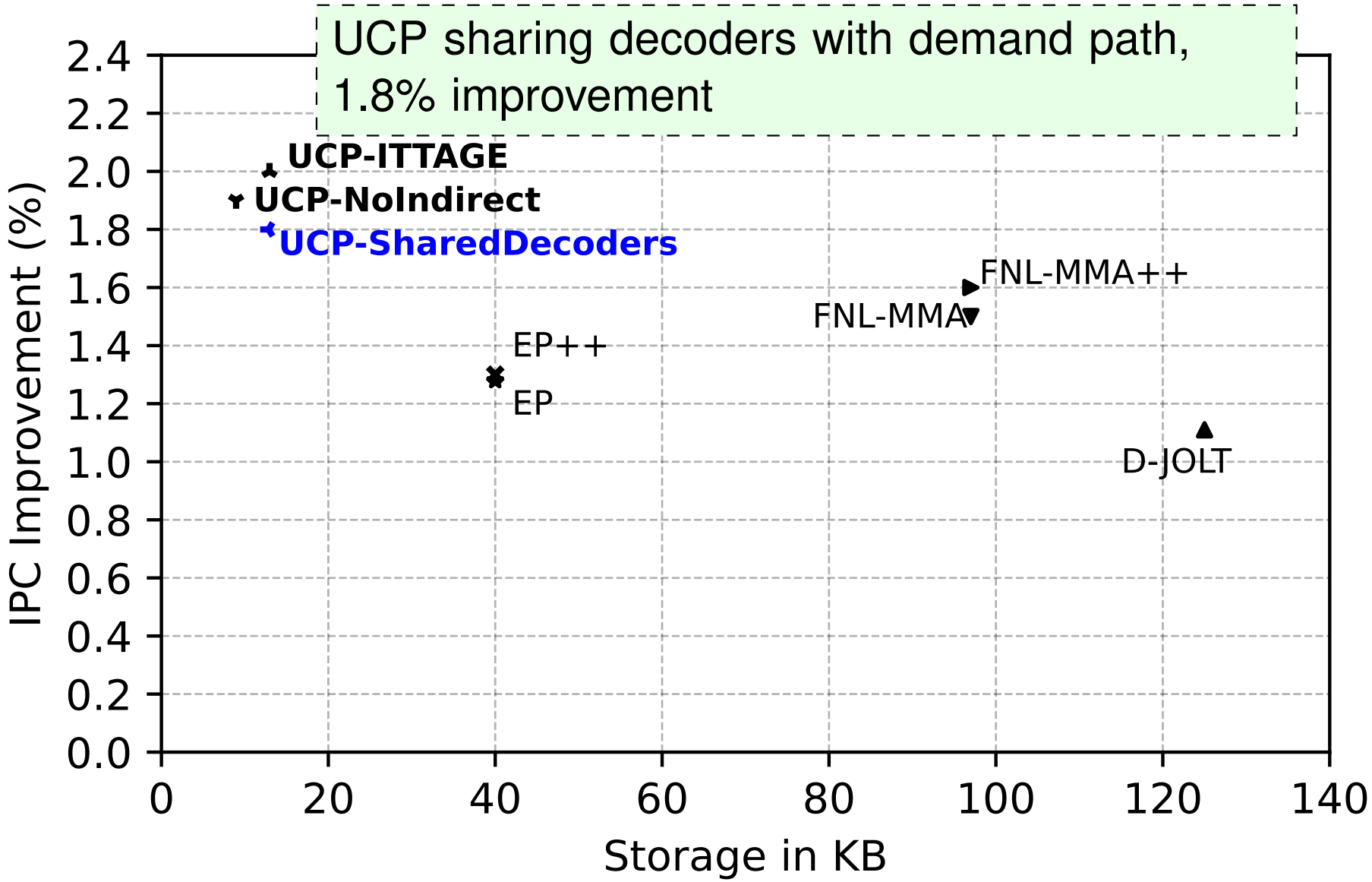
Methodology & Results

IPC Improvements



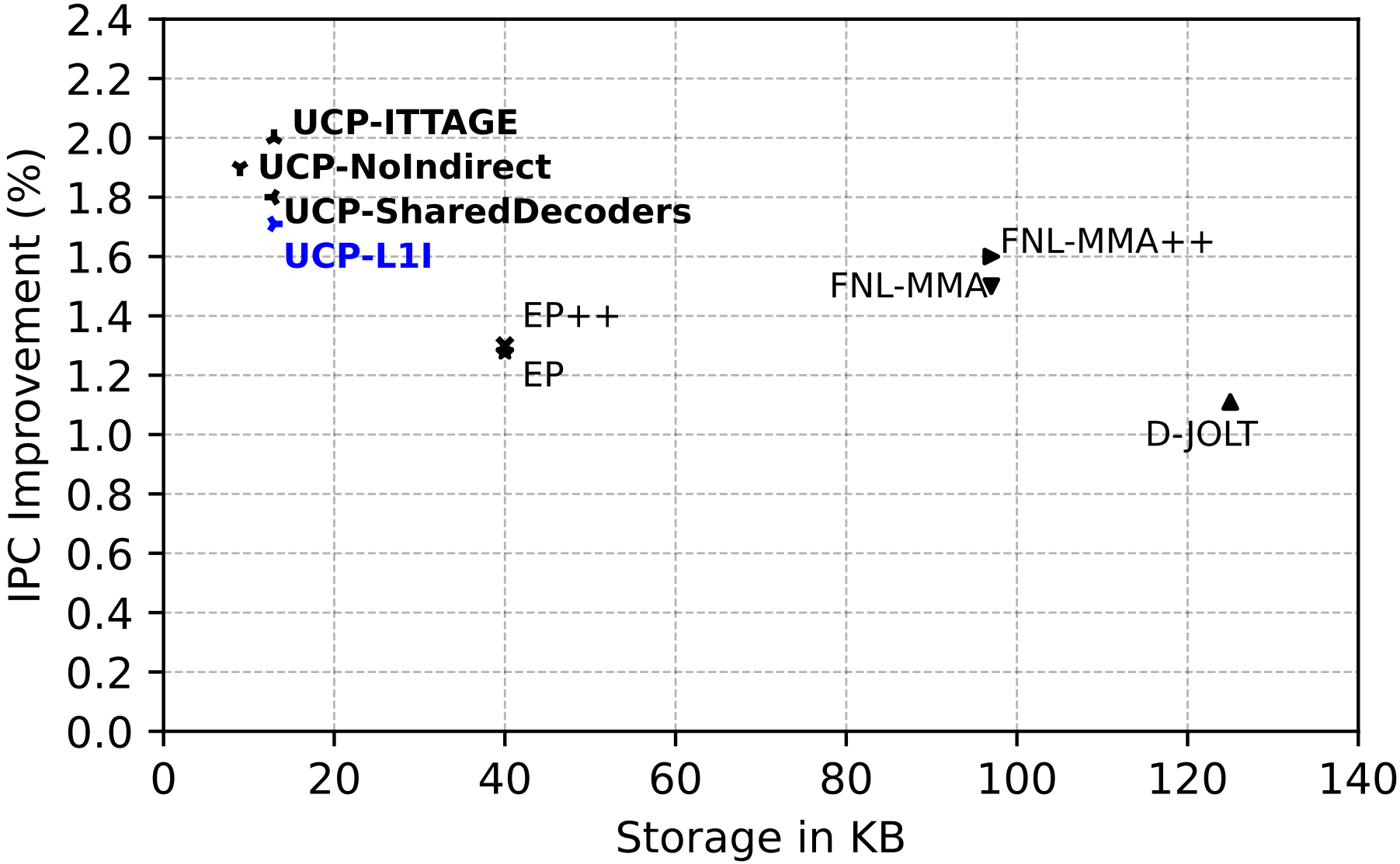
Methodology & Results

IPC Improvements



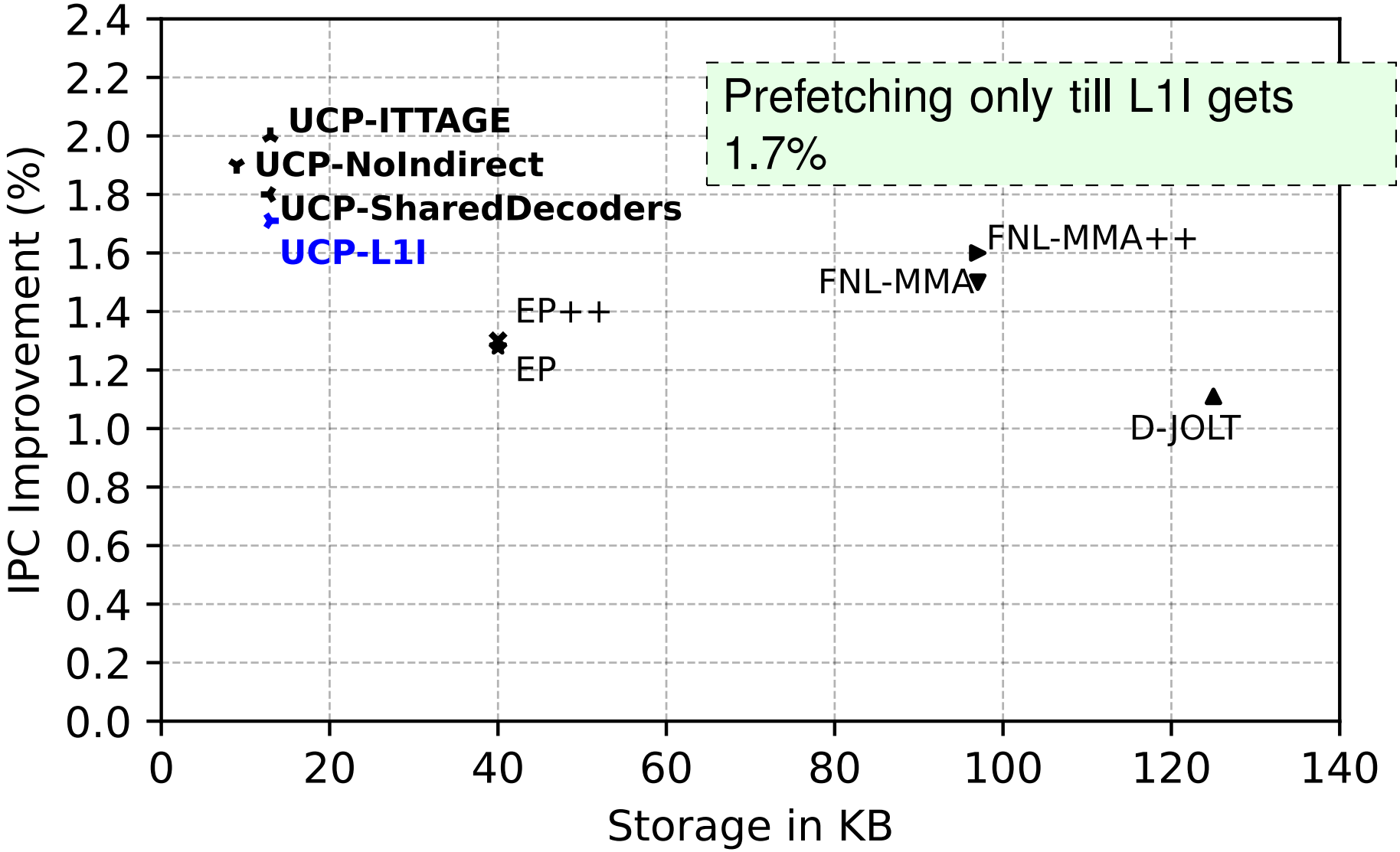
Methodology & Results

IPC Improvements



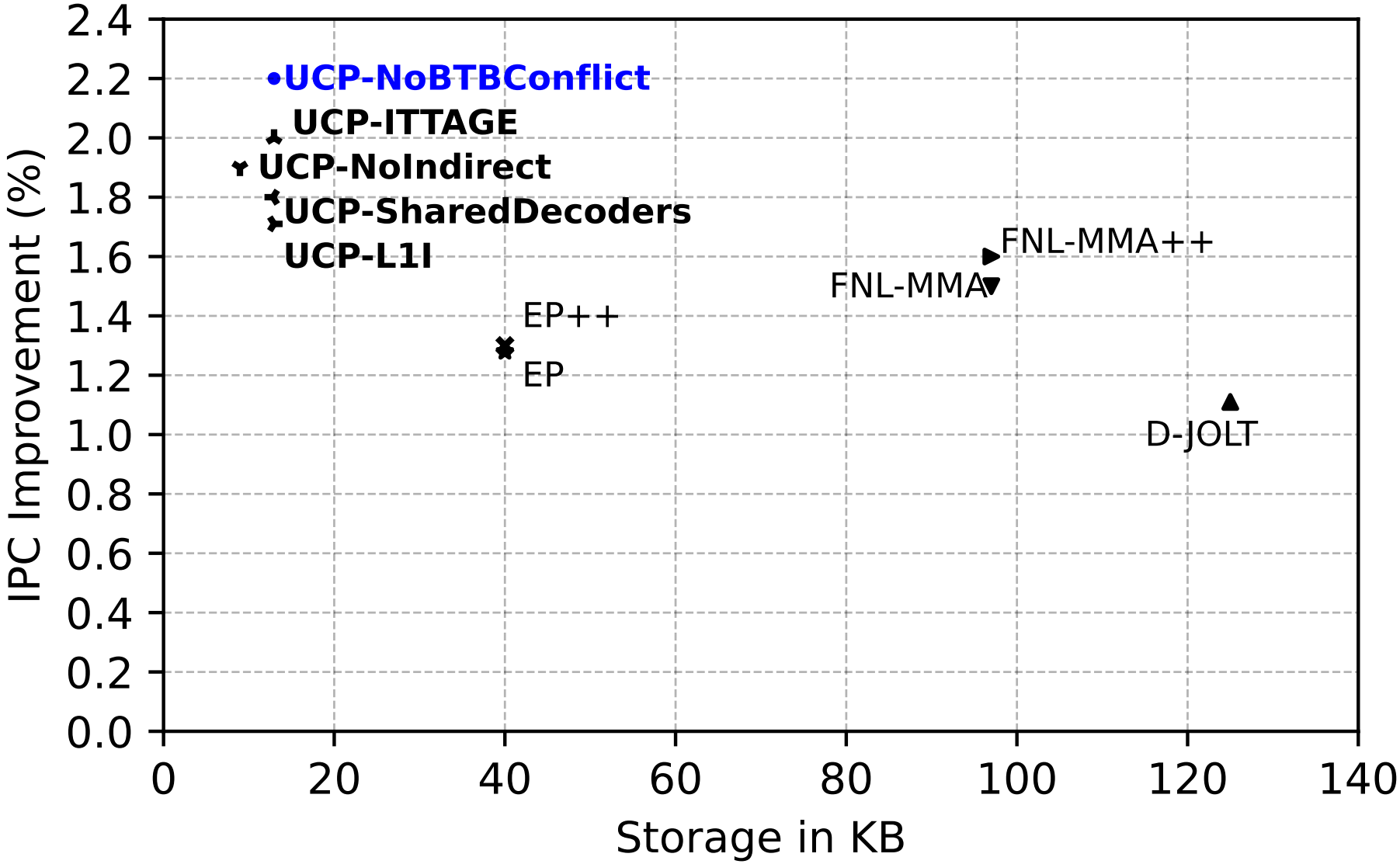
Methodology & Results

IPC Improvements



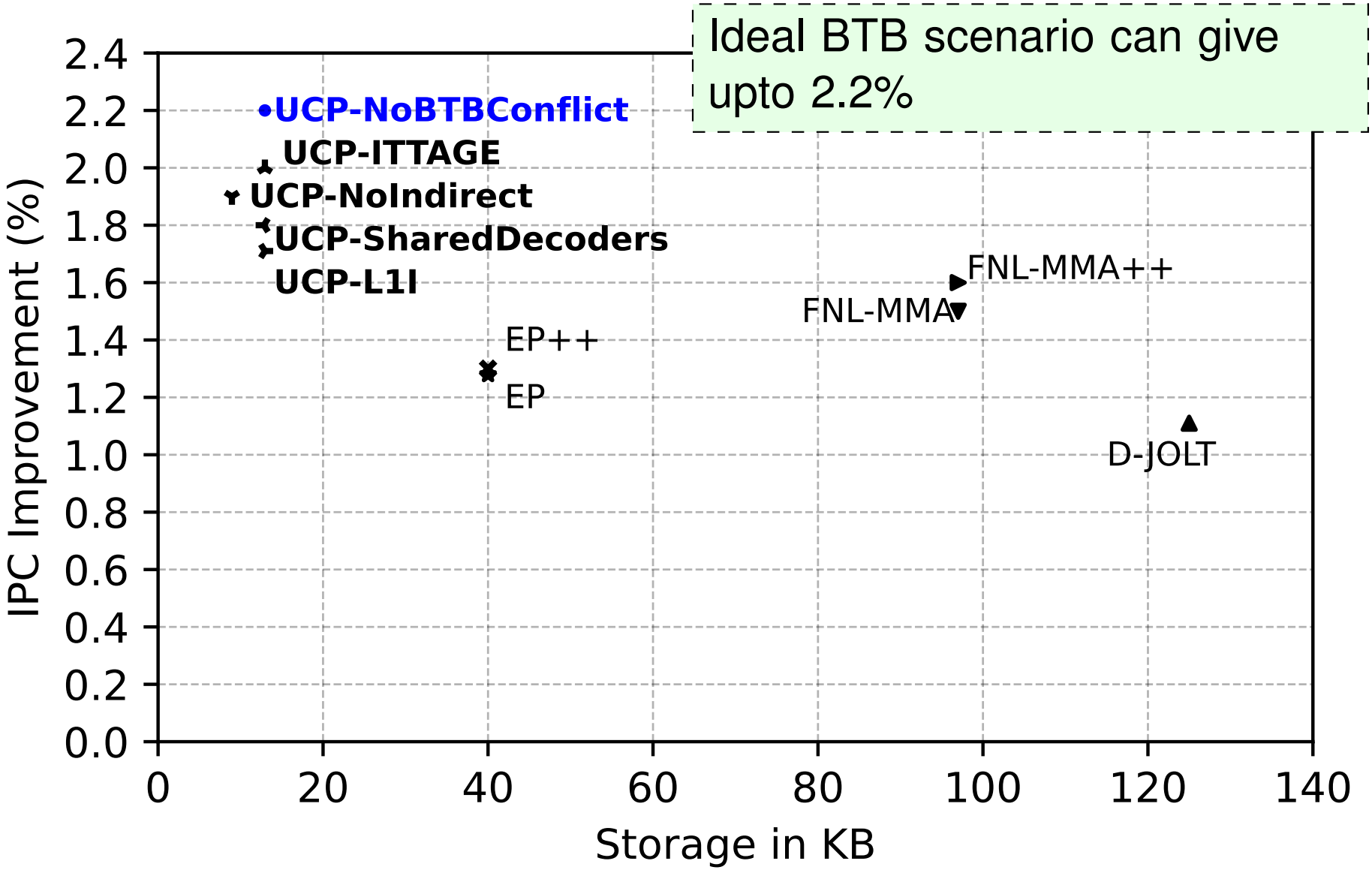
Methodology & Results

IPC Improvements



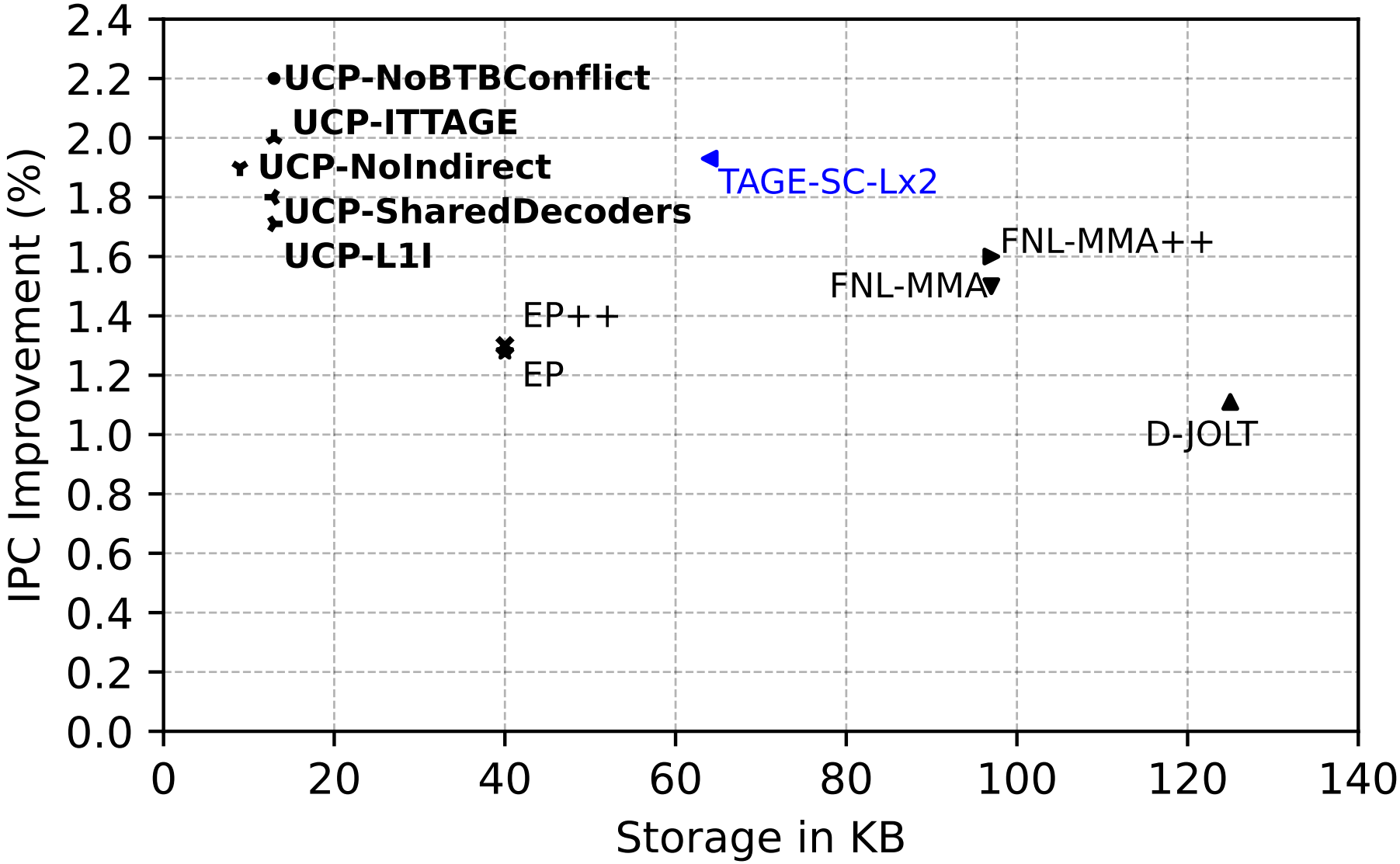
Methodology & Results

IPC Improvements



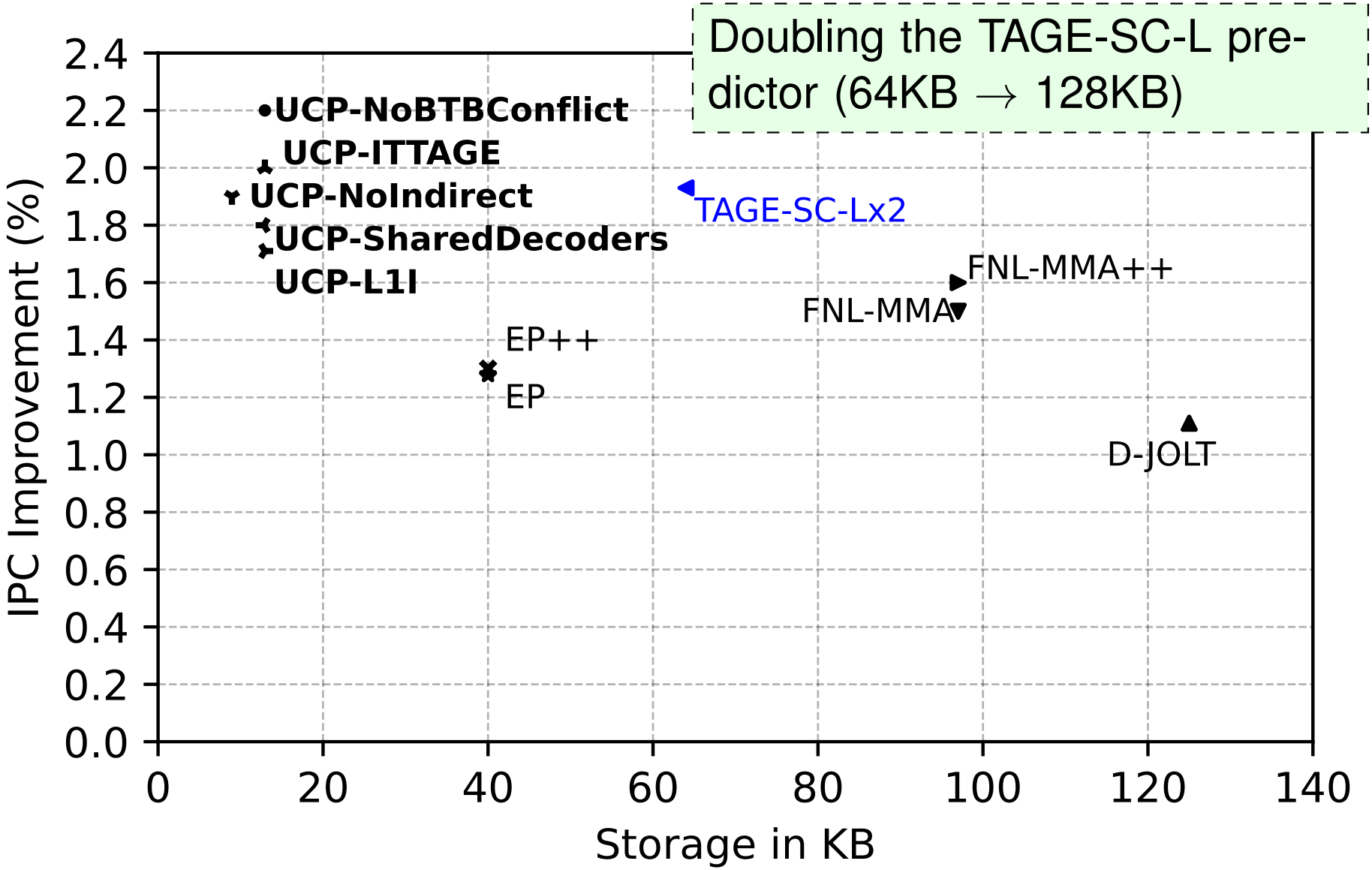
Methodology & Results

IPC Improvements



Methodology & Results

IPC Improvements



Outline

Conclusions

→ FTQ fails to hide L1I miss latency on branch miss

Conclusions

→ FTQ fails to hide L1I miss latency on branch miss

→ Focusing only a few but critical instructions can provide better performance

Conclusions

- → FTQ fails to hide L1I miss latency on branch miss
- → Focusing only a few but critical instructions can provide better performance
- → UCP focus on critical instructions after a H2P branch

Conclusions

- → FTQ fails to hide L1I miss latency on branch miss
- → Focusing only a few but critical instructions can provide better performance
- → UCP focus on critical instructions after a H2P branch
- → Still space for improvement in optimizing μ-op cache

ALTERNATE PATH μ-OP CACHE PREFETCHING

Sawan Singh[1] Arthur Perais[2] Alexandra Jimborean[1]
Alberto Ros[1]

singh.sawan@um.es

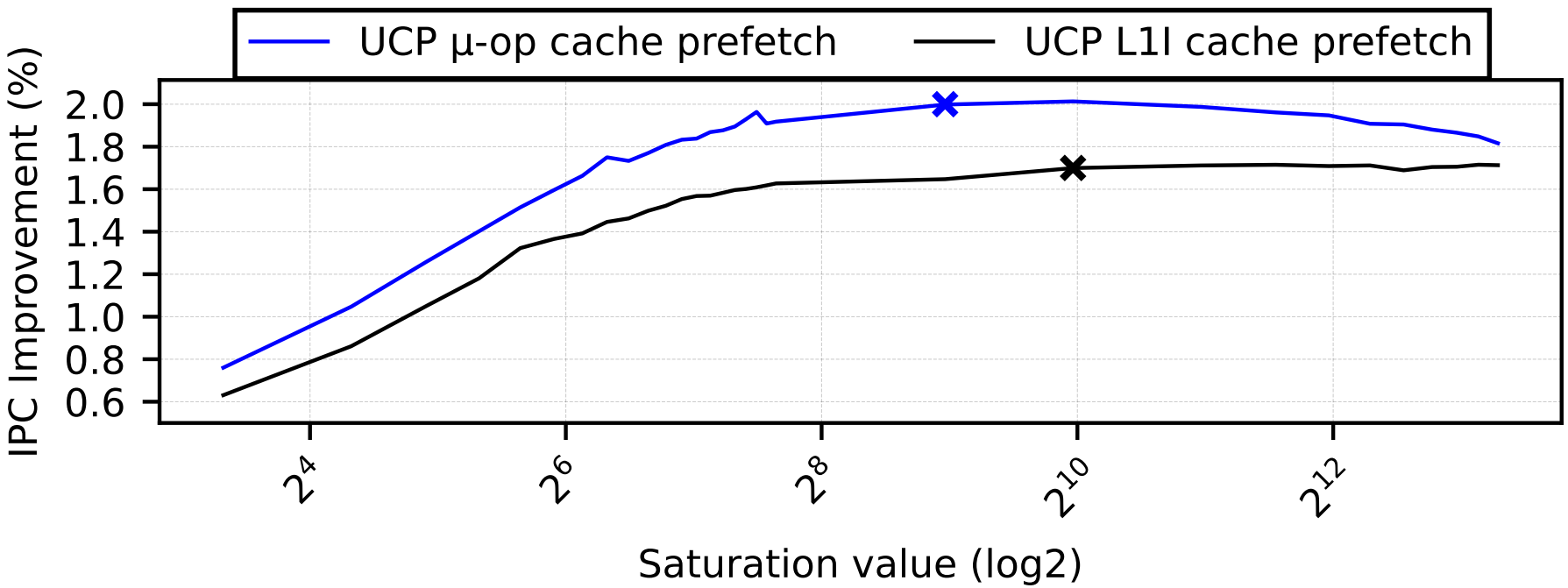
Thank you for your attention!

ECHO, ERC Consolidator Grant (No 819134)

BACKUP SLIDES[BTB BANKING]

- → UCP reuses the BTB by doubling the number of BTB banks (from 16 to 32)
- → Each cycle we determine the banks to be accessed
- → By default, demand requests are given priority to access the conflict banks
- → UCP keeps a 3-bit saturated counter which is incremented every time the alternate path is delayed
- → When the counter saturates, the alternate path is given priority for the conflict banks in that cycle
- → The counter resets next cycle

BACKUP SLIDES[SATURATION COUNTER]



BACKUP SLIDES[L1I PREFETCHERS]