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 decoders

[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 decoders
- Potential to provide an IPC improvement of 10.82%

[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 decoders
- Potential to provide an IPC improvement of 10.82%

→ 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*

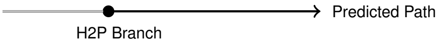
Overview

→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoders
- Potential to provide an IPC improvement of 10.82%

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

① Identify hard-to-predict branches



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

OVERVIEW

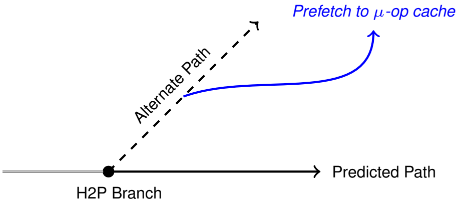
→ μ-op Cache

- Holds recently decoded μ-ops
- First introduced for energy savings[1] in x86 which requires complex decoders
- Potential to provide an IPC improvement of 10.82%

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

① Identify hard-to-predict branches

② Prefetch μ-ops from alternate path

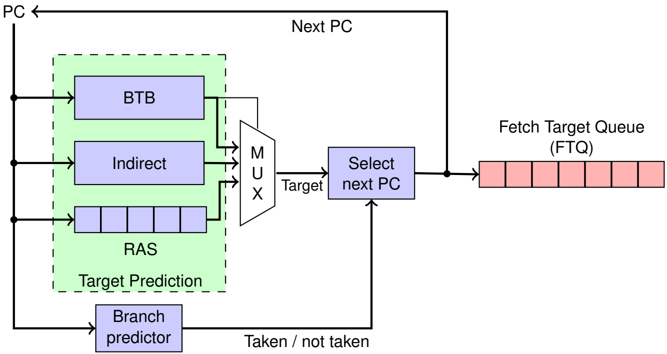


[1] *Solomon et al. Micro-operation cache: a power aware frontend for variable instruction length ISA*

OUTLINE

1. Overview
2. Background & Motivation
3. UCP (μ*-op Cache Prefetching*)
4. Methodology & Results
5. Conclusions

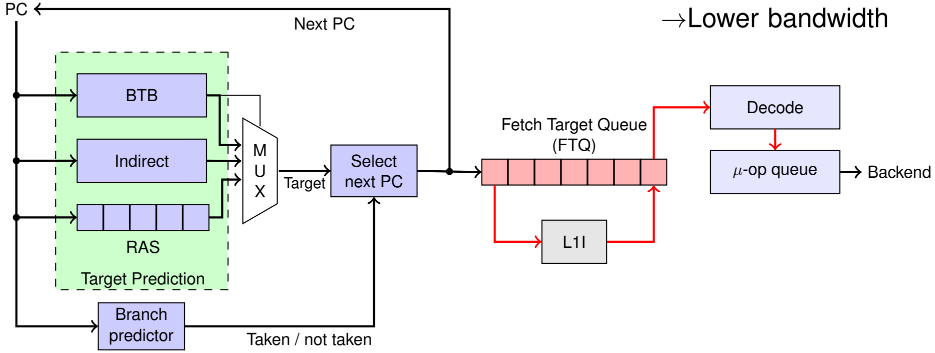
Background & Motivation
Processor Front-End



BACKGROUND & MOTIVATION

PROCESSOR FRONT-END

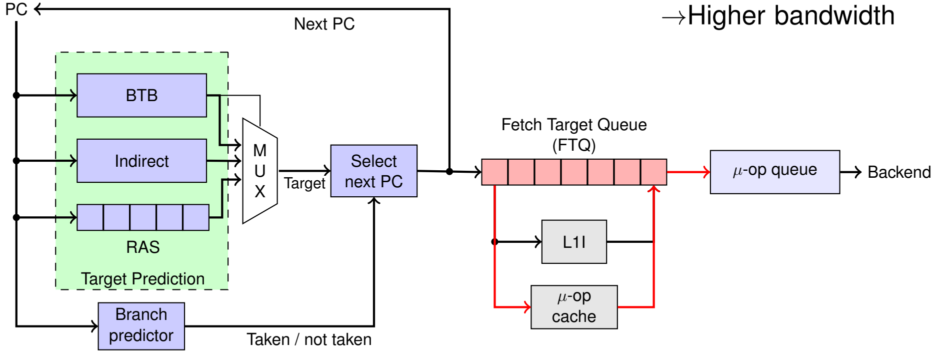
From L1I:
→Decode latency
→Decode energy
→Lower bandwidth



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

→ Server workloads overwhelm current μ-op caches

  → 0.87% IPC improvement over no μ-op cache

Background & Motivation

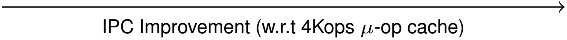
Performance of μ-ops cache with server workloads

- → Server workloads overwhelm current μ-op caches
  - → 0.87% IPC improvement over no μ-op cache
- → Increasing size of μ-op cache does not help

BACKGROUND & MOTIVATION

PERFORMANCE OF μ-OPS CACHE WITH SERVER WORKLOADS

- → Server workloads overwhelm current μ-op caches
  - → 0.87% IPC improvement over no μ-op cache
- → Increasing size of μ-op cache does not help



Background & Motivation

Performance of μ-ops cache with server workloads

- → Server workloads overwhelm current μ-op caches
  - → 0.87% IPC improvement over no μ-op cache
- → Increasing size of μ-op cache does not help

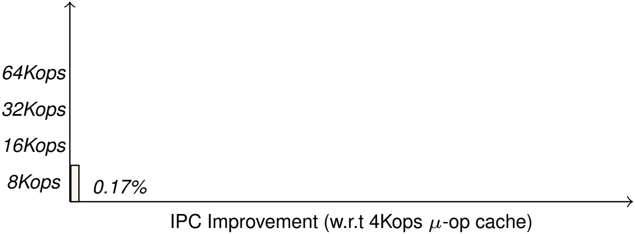
Background & Motivation

Performance of μ-ops cache with server workloads

→ Server workloads overwhelm current μ-op caches

  → 0.87% IPC improvement over no μ-op cache

→ Increasing size of μ-op cache does not help



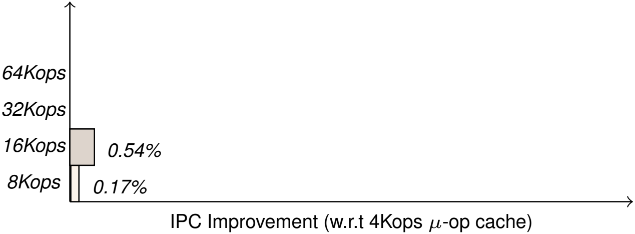
Background & Motivation

Performance of μ-ops cache with server workloads

→ Server workloads overwhelm current μ-op caches

→ 0.87% IPC improvement over no μ-op cache

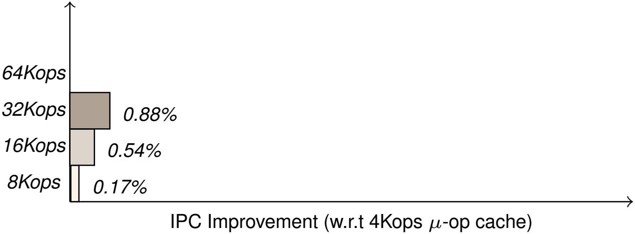
→ Increasing size of μ-op cache does not help



Background & Motivation

Performance of μ-ops cache with server workloads

- → Server workloads overwhelm current μ-op caches
  - → 0.87% IPC improvement over no μ-op cache
- → Increasing size of μ-op cache does not help



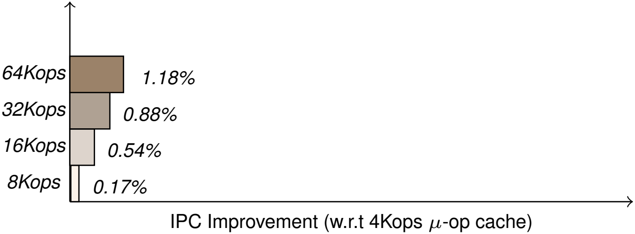
Background & Motivation

Performance of μ-ops cache with server workloads

→ Server workloads overwhelm current μ-op caches

→ 0.87% IPC improvement over no μ-op cache

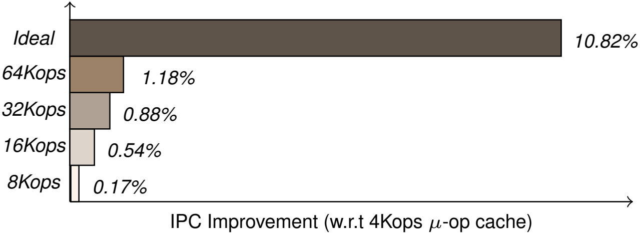
→ Increasing size of μ-op cache does not help



Background & Motivation

Performance of μ-ops cache with server workloads

- → Server workloads overwhelm current μ-op caches
  - → 0.87% IPC improvement over no μ-op cache
- → Increasing size of μ-op cache does not help
- → Ideal μ-op cache can provide 10.82% improvement



Background & Motivation

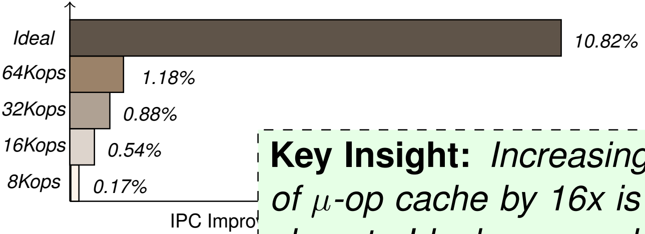
Performance of μ-ops cache with server workloads

→ Server workloads overwhelm current μ-op caches

  → 0.87% IPC improvement over no μ-op cache

→ 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

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

Background & Motivation

FTQ behavior on branch miss

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

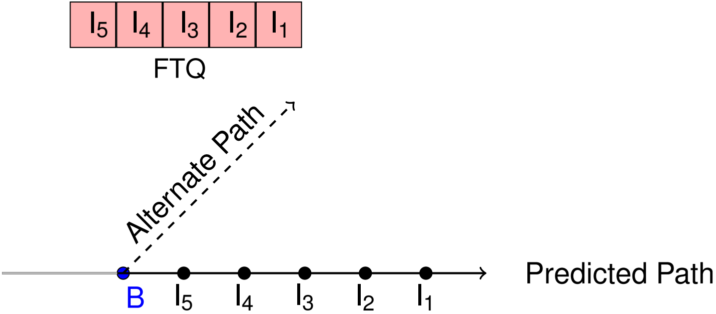
- The FTQ is cleared on a branch misprediction

Background & Motivation

FTQ behavior on branch miss

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

- The FTQ is cleared on a branch misprediction

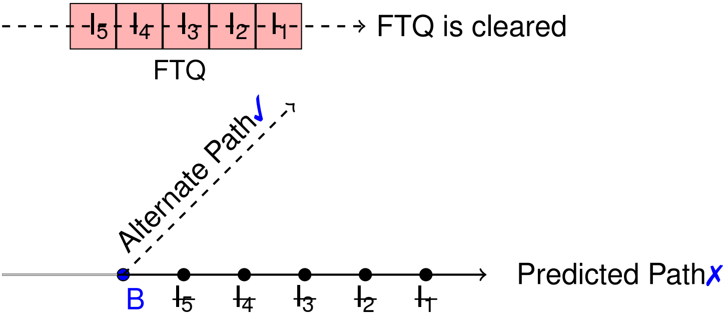


Background & Motivation

FTQ behavior on branch miss

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

- The FTQ is cleared on a branch misprediction



~~I_5~~ ~~I_4~~ ~~I_3~~ ~~I_2~~ ~~I_1~~
FTQ is cleared
FTQ
Alternate Path✓
Predicted Path✗
B
~~I_5~~ ~~I_4~~ ~~I_3~~ ~~I_2~~ ~~I_1~~

BACKGROUND & MOTIVATION

FTQ BEHAVIOR ON BRANCH MISS

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

- The 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?

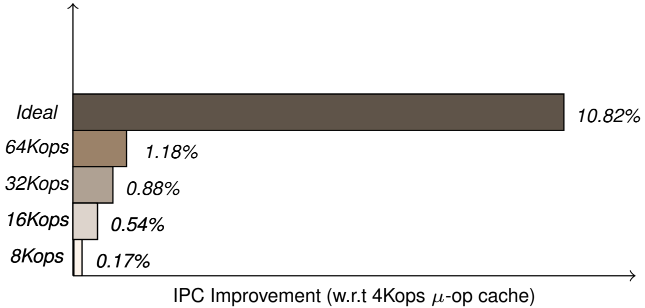
Background & Motivation

FTQ behavior on branch miss

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

- The 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?



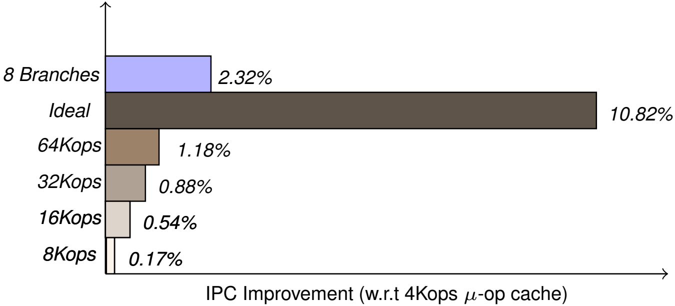
BACKGROUND & MOTIVATION

FTQ BEHAVIOR ON BRANCH MISS

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

- The 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?



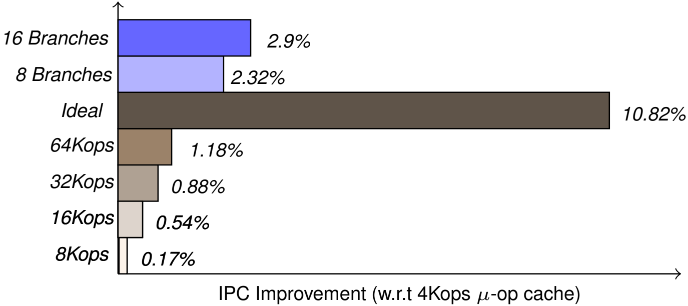
BACKGROUND & MOTIVATION

FTQ BEHAVIOR ON BRANCH MISS

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

- The 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?



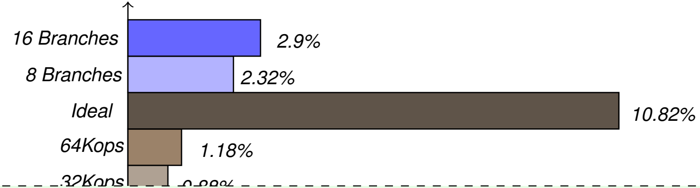
Background & Motivation

FTQ behavior on branch miss

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

- The 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:

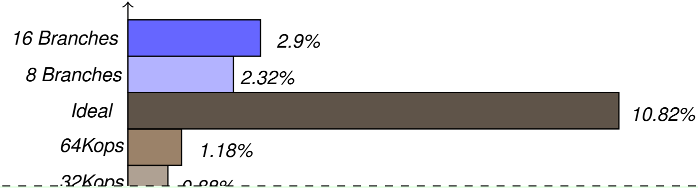
Background & Motivation

FTQ behavior on branch miss

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

- The 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*

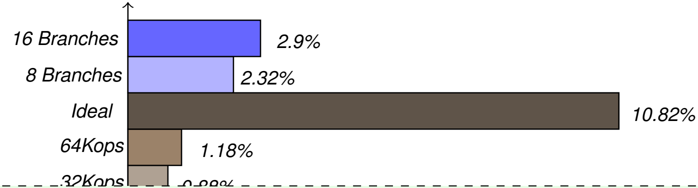
Background & Motivation

FTQ behavior on branch miss

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

- The 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*

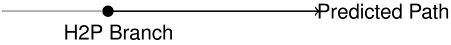
OUTLINE

1. Overview
2. Background & Motivation
3. *UCP (μ-op Cache Prefetching)*
4. Methodology & Results
5. Conclusions

UCP

UCP: Overview

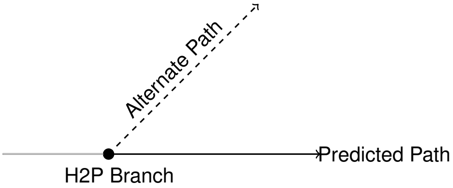
① Identifies a hard-to-predict conditional branch (H2P)



UCP: Overview

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

② Generate addresses on alternate path



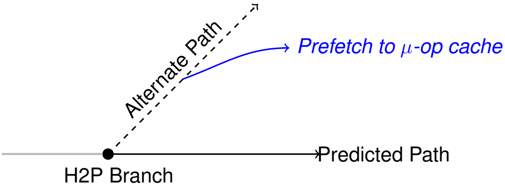
UCP

UCP: Overview

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

② Generate addresses on alternate path

③ Prefetch the alternate path to the μ-op cache



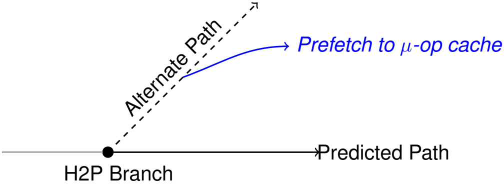
UCP

UCP: Overview

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

② Generate addresses on alternate path

③ Prefetch the alternate path to the μ-op cache



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

UCP

① H2P Branch Detection

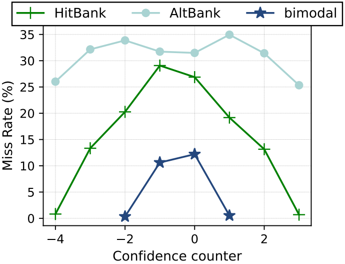
→ H2P Branch: a branch which has high chance of being mispredicted

UCP

① H2P Branch Detection

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

→ TAGE-Conf[2]



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

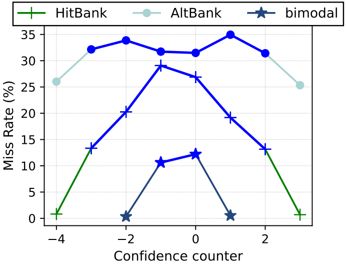
UCP

① H2P Branch Detection

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

→ TAGE-Conf[2]

- Not saturated predictions from AltBank, HitBank & BiModal



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

UCP

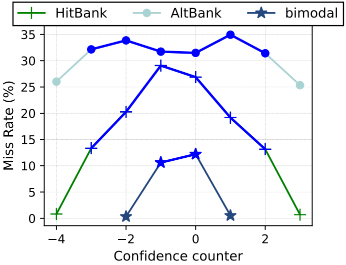
① H2P Branch Detection

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

  → TAGE-Conf[2]

    - Not saturated predictions from AltBank, HitBank & BiModal

  → UCP-Conf



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

UCP

① H2P Branch Detection

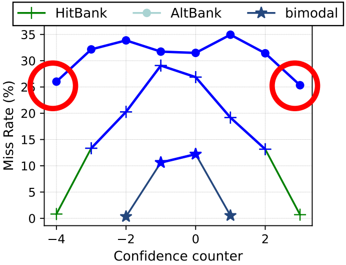
→ H2P Branch: a branch which has high chance of being mispredicted

→ TAGE-Conf[2]

- Not saturated predictions from AltBank, HitBank & BiModal

→ UCP-Conf

- All predictions from AltBanks shows high miss rate



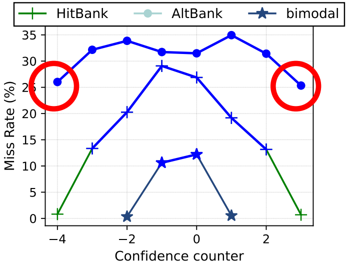
[2] Seznec et. al. *Storage free confidence estimation for the TAGE branch predictor*

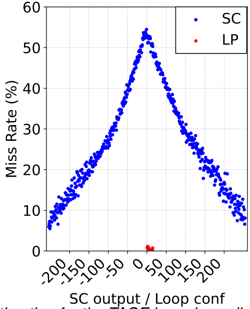
UCP

① H2P Branch Detection

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

- → TAGE-Conf[2]
  - Not saturated predictions from AltBank, HitBank & BiModal
- → UCP-Conf
  - All predictions from AltBanks shows high miss rate
  - SC shows high miss rate





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

UCP

① H2P Branch Detection

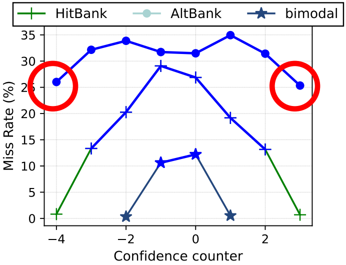
→ H2P Branch: a branch which has high chance of being mispredicted

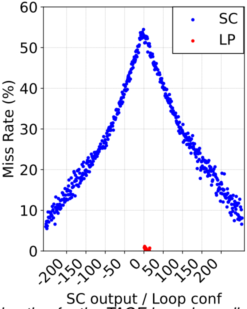
→ TAGE-Conf[2]

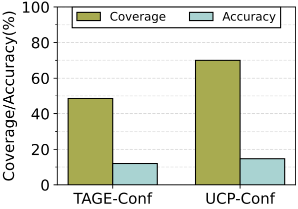
- Not saturated predictions from AltBank, HitBank & BiModal

→ UCP-Conf

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







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

UCP

② Generating Addresses

→ Needs BPU

UCP

② Generating Addresses

→ Needs BPU

- Banking
- Add new predictors

UCP

② Generating Addresses

→ Needs BPU

- Banking
  - For server workloads BTB size is critical
  - Increase the number of banks from 16 to 32
- Add new predictors

UCP

② Generating Addresses

→ Needs BPU

- Banking
    - For server workloads BTB size is critical
    - Increase the number of banks from 16 to 32
- Add new predictors
    - Alt BP: 8KB TAGE-SC-L
    - Alt Indirect: 4KB ITTAGE
    - Alt RAS: 16-entry

UCP

② Generating Addresses

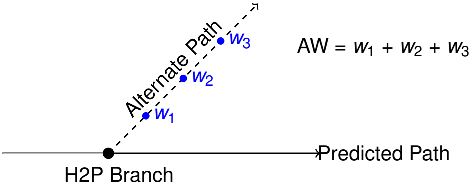
→ When to stop?

UCP

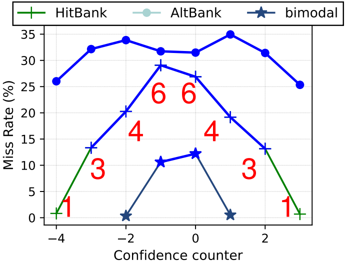
② Generating Addresses

→ When to stop?

- Weight of each branch on the alternate path is accumulated
  - High confident → lower weight
  - Lower confident → higher weight



AW = w_1 + w_2 + w_3

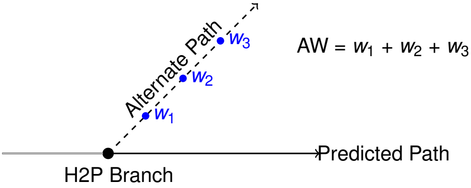


UCP

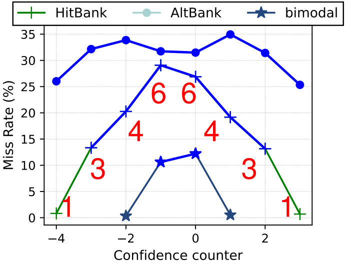
② Generating Addresses

→ 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



AW = $w_1 + w_2 + w_3$

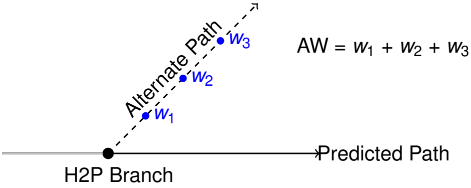


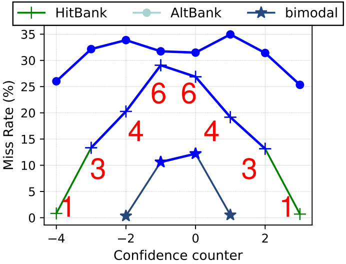
UCP

② Generating Addresses

→ When to stop?

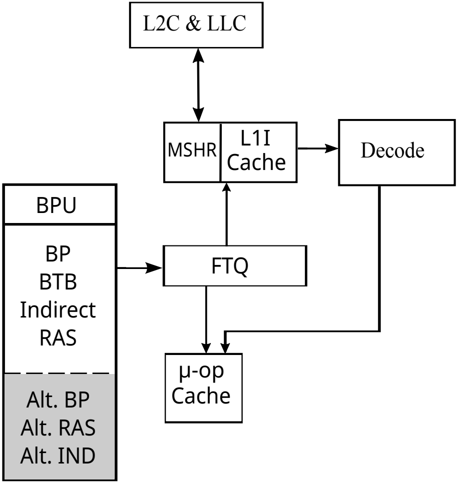
- Weight of each branch on the alternate path is accumulated
  - High confident → lower weight
  - Lower confident → higher weight
- BTB miss on the alternate path
- New H2P branch is detected





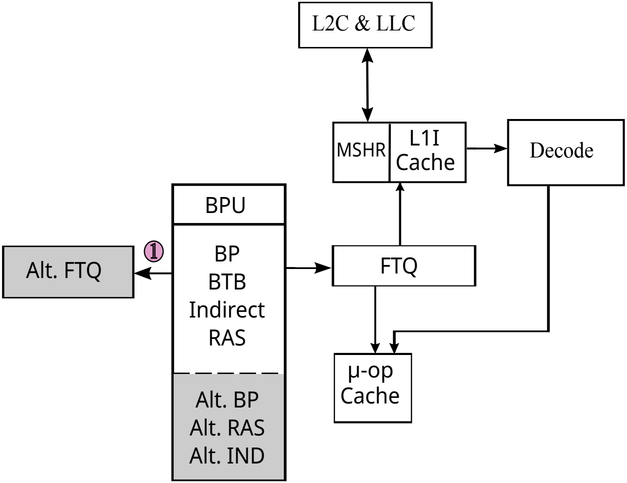
UCP

③ Prefetching



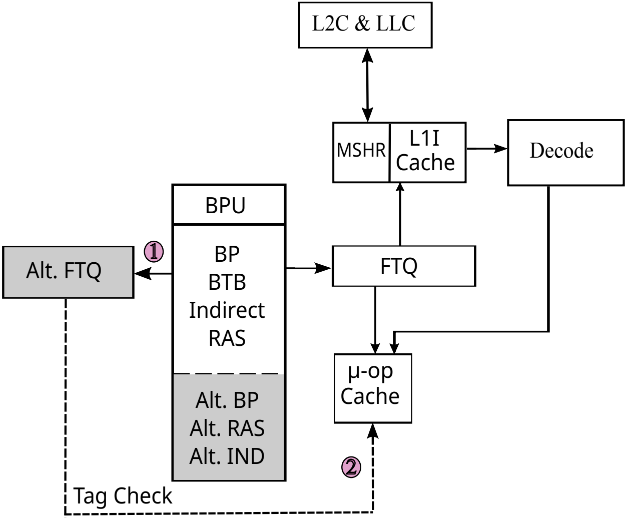
UCP

③ Prefetching



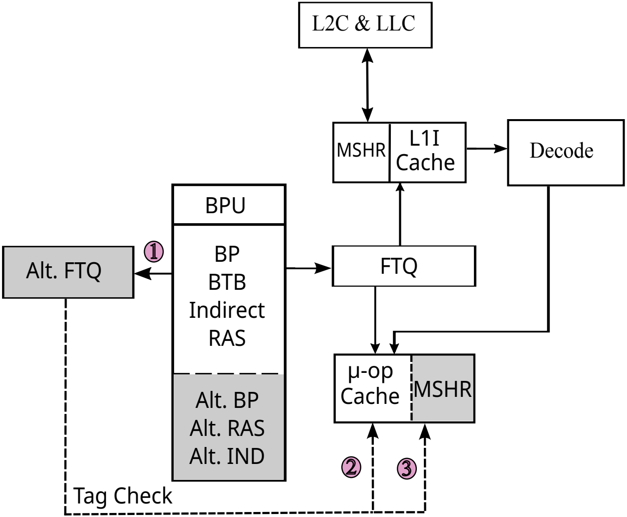
UCP

③ Prefetching



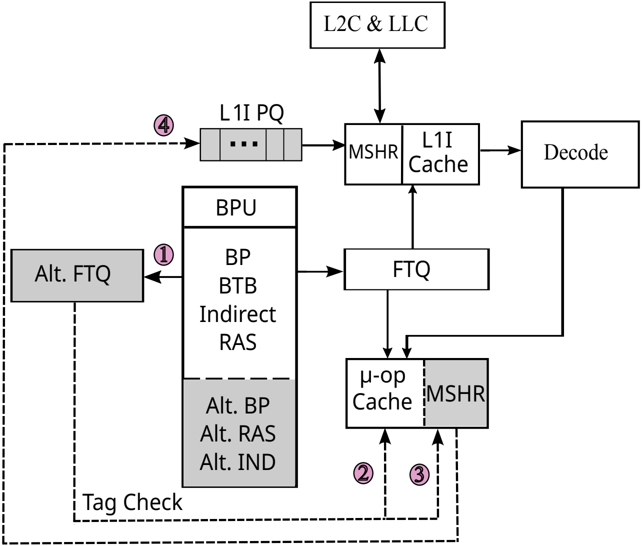
UCP

③ Prefetching



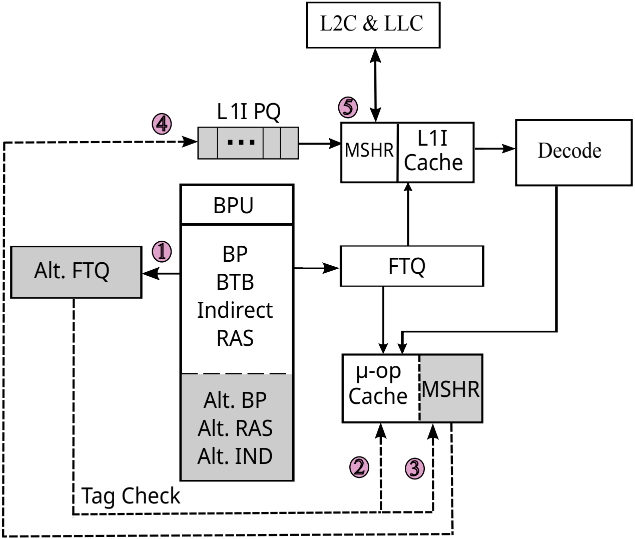
UCP

③ Prefetching



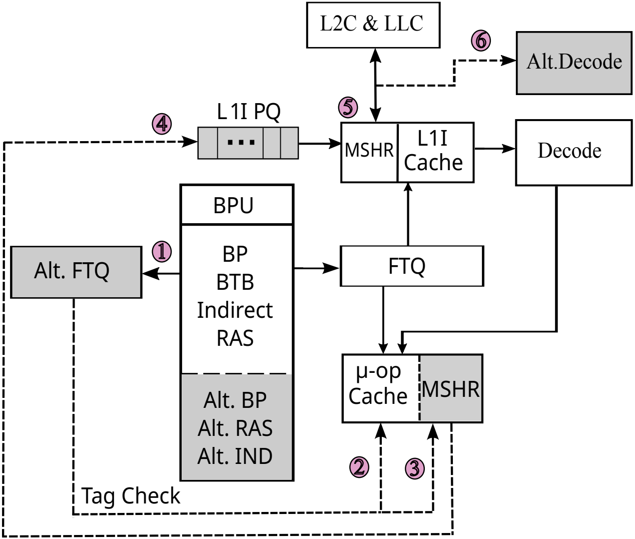
UCP

③ Prefetching



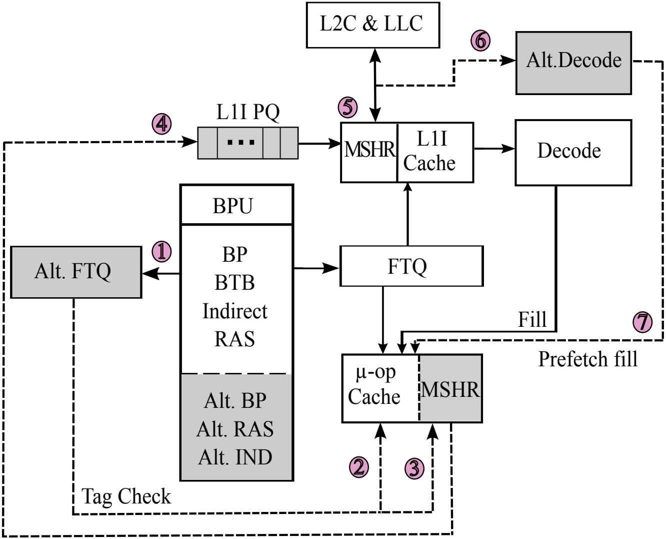
UCP

③ Prefetching



UCP

③ Prefetching



Outline

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

METHODOLOGY & RESULTS

SIMULATION SETUP

→ ChampSim

Methodology & Results

Simulation Setup

→ ChampSim

→ Intel Alder Lake like microarchitecture

Methodology & Results

Simulation Setup

→ ChampSim

→ Intel Alder Lake like microarchitecture

→ Subset of CVP1[3] (traces showing $\geqq$ 5% improvement with ideal μ-op cache) [2 FP, 97 INT, 73 Crypto and 134 datacenter trace]

[3] Feliu et. al. *Rebasing Microarchitectural Research with Industry Traces*

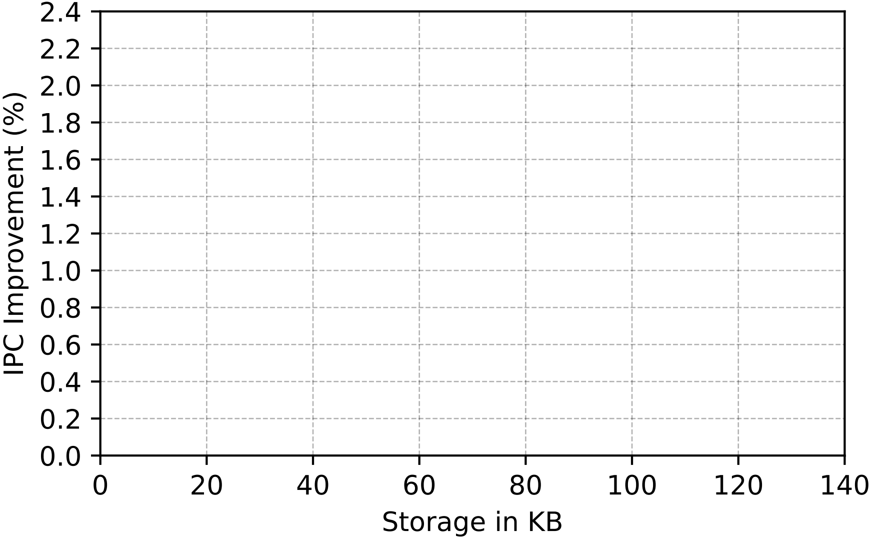
SIMULATION SETUP

- → ChampSim
- → Intel Alder Lake like microarchitecture
- → Subset of CVP1[3] (traces showing $\geqq$ 5% improvement with ideal μ-op cache) [2 FP, 97 INT, 73 Crypto and 134 datacenter trace]
- → We execute 100M instructions, 50M warmup and 50M to collect stats

[3] Feliu et. al. *Rebasing Microarchitectural Research with Industry Traces*

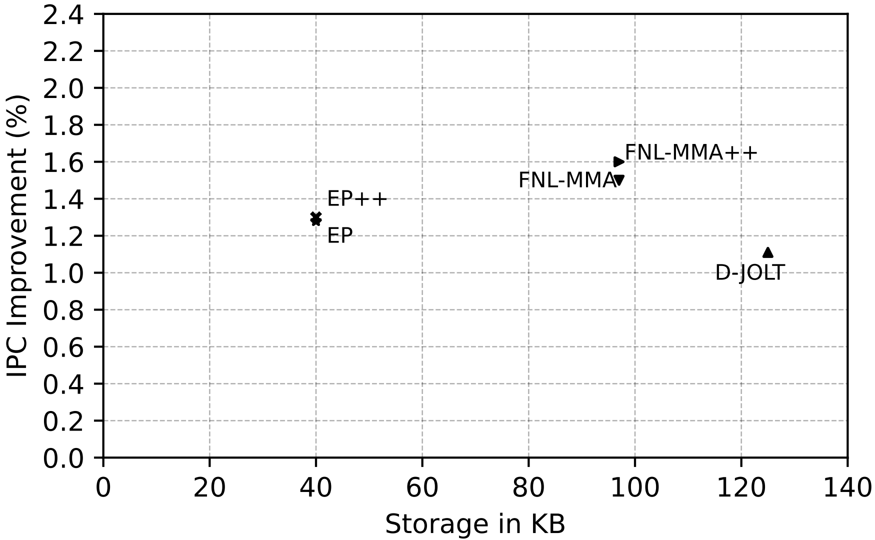
Methodology & Results

IPC Improvements



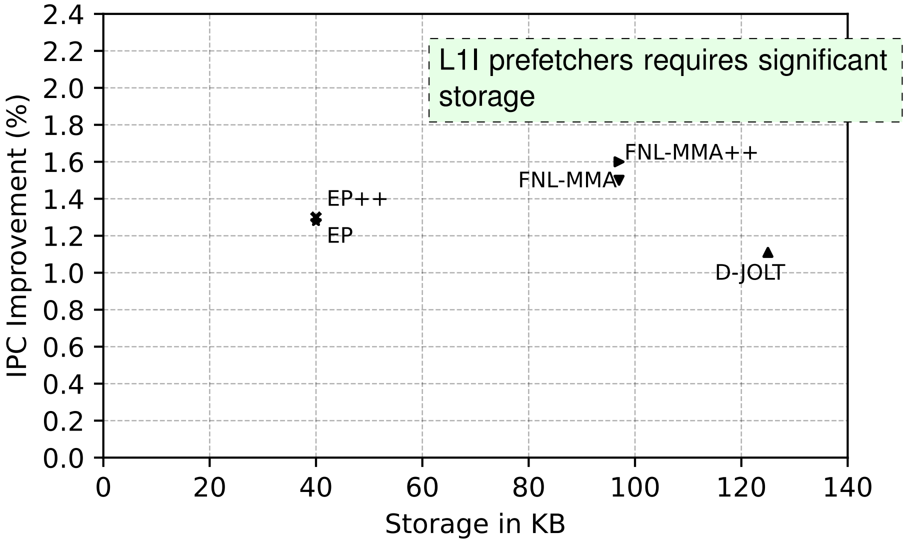
METHODOLOGY & RESULTS

IPC IMPROVEMENTS



Methodology & Results

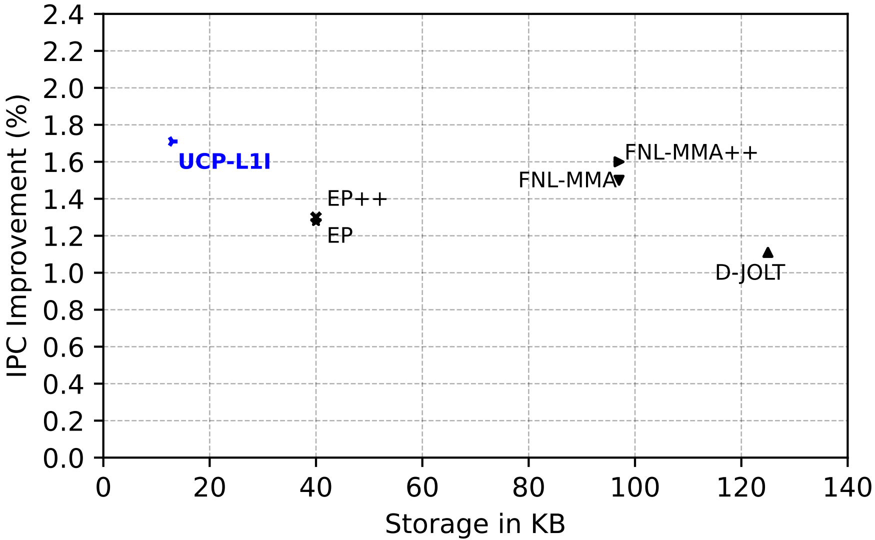
IPC Improvements



L1I prefetchers requires significant storage

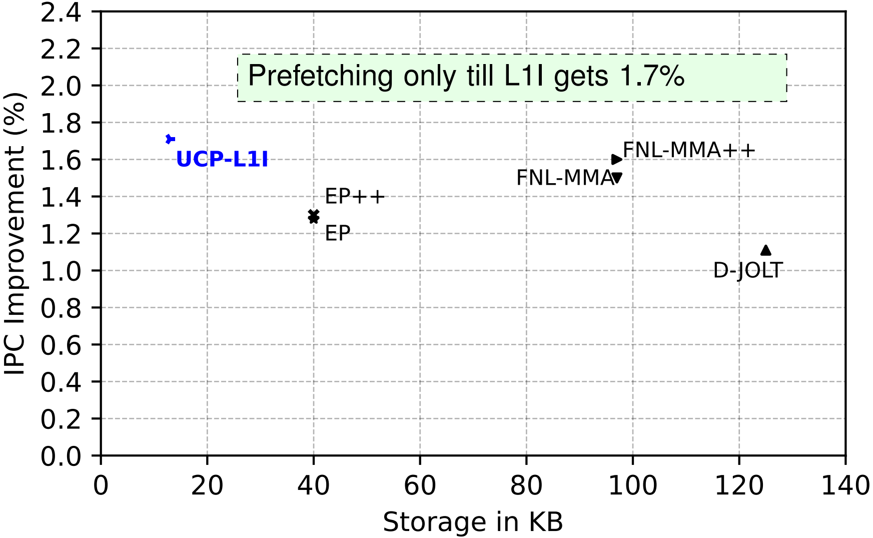
Methodology & Results

IPC Improvements



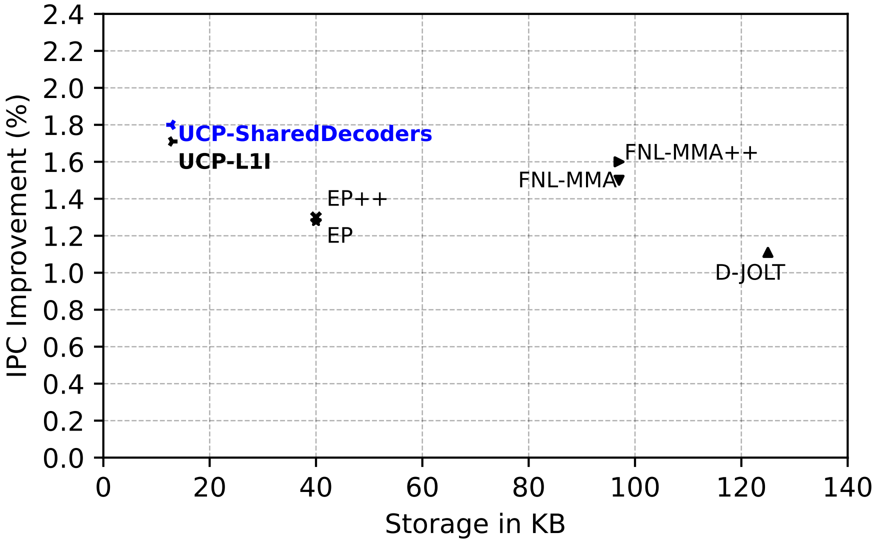
METHODOLOGY & RESULTS

IPC IMPROVEMENTS


Prefetching only till L1I gets 1.7%

METHODOLOGY & RESULTS

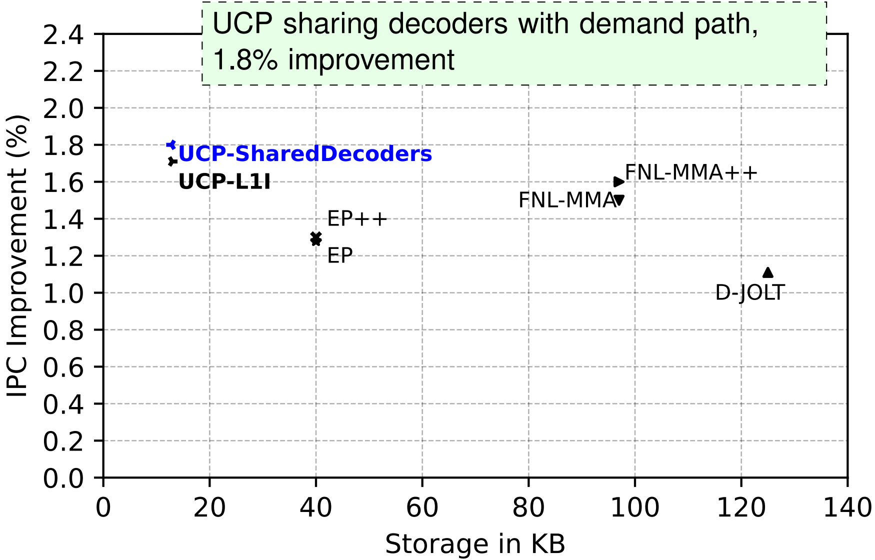
IPC IMPROVEMENTS



METHODOLOGY & RESULTS

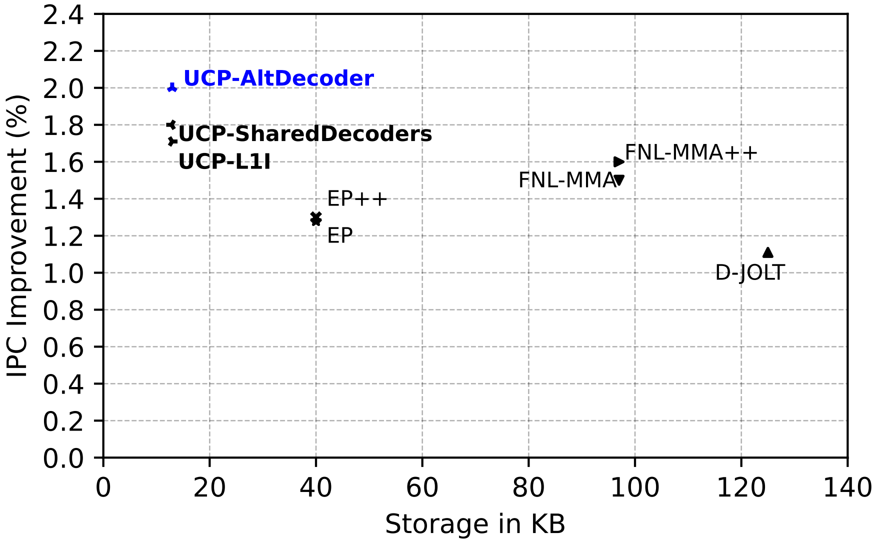
IPC IMPROVEMENTS

UCP sharing decoders with demand path, 1.8% improvement



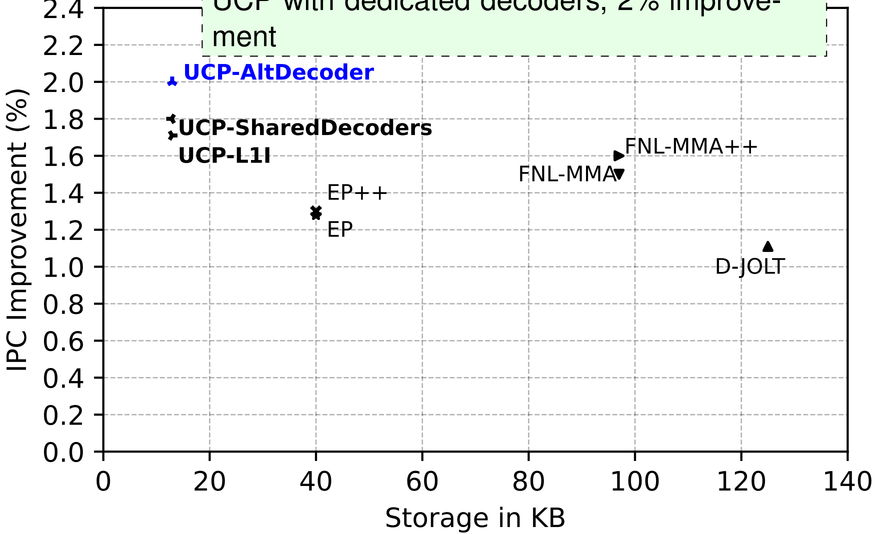
Methodology & Results

IPC Improvements



METHODOLOGY & RESULTS

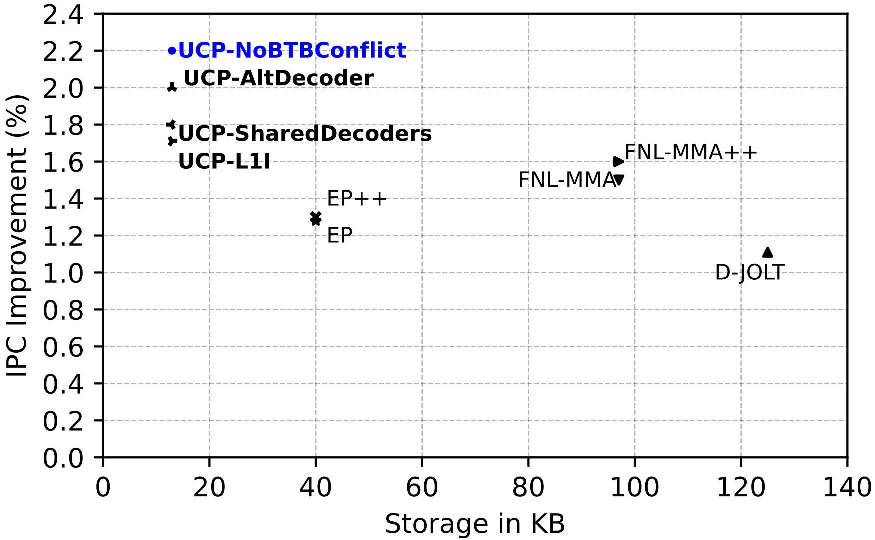
IPC IMPROVEMENTS



UCP with dedicated decoders, 2% improvement

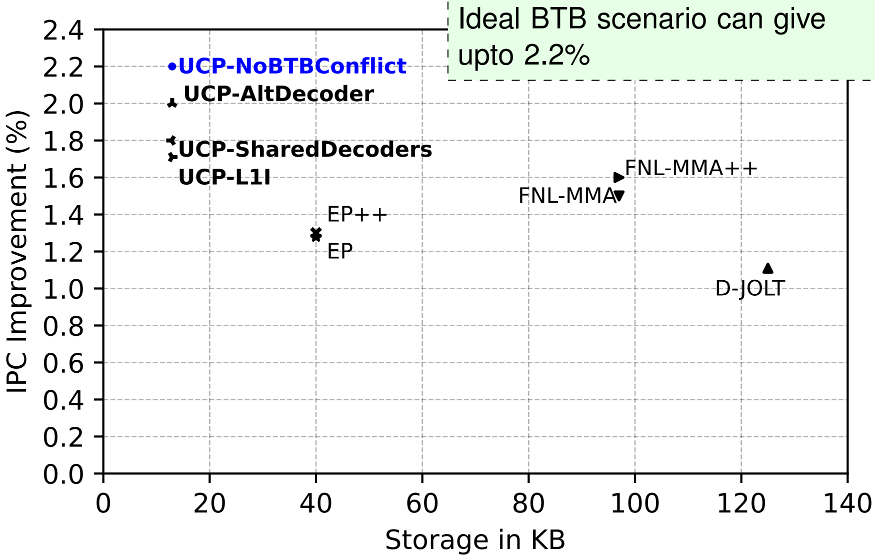
Methodology & Results

IPC Improvements



METHODOLOGY & RESULTS

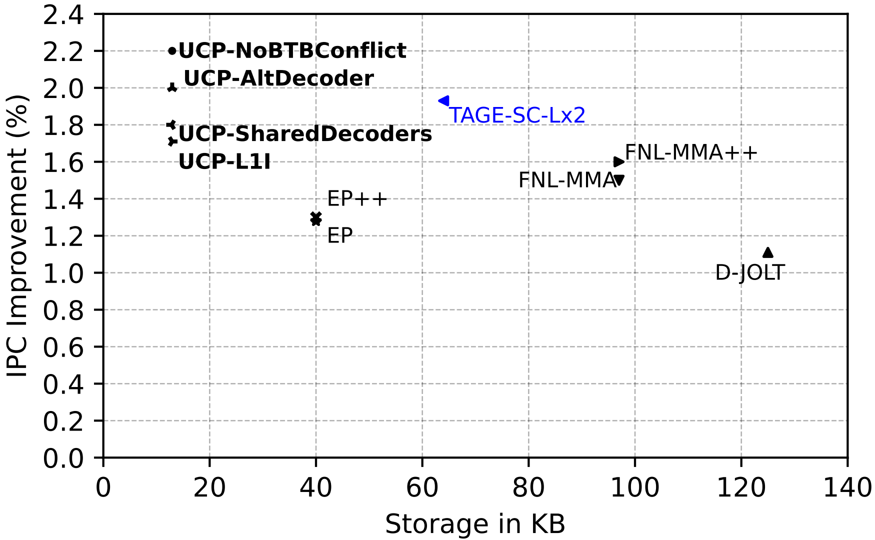
IPC IMPROVEMENTS



Ideal BTB scenario can give upto 2.2%

METHODOLOGY & RESULTS

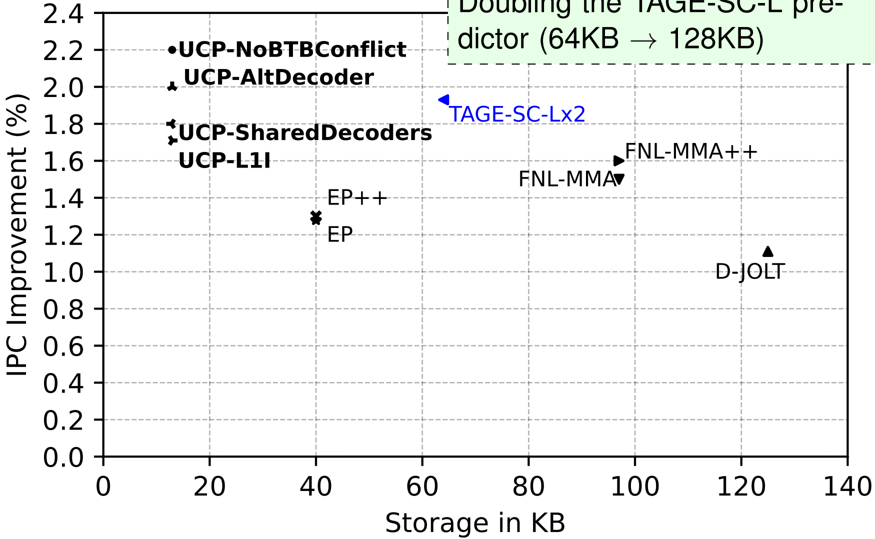
IPC IMPROVEMENTS



Methodology & Results

IPC Improvements

Doubling the TAGE-SC-L predictor (64KB → 128KB)



OUTLINE

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

CONCLUSIONS

→ μ-op cache can be used for performance improvements

CONCLUSIONS

→ μ-op cache can be used for performance improvements

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

Conclusions

→ μ-op cache can be used for performance improvements

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

  - Identifies a hard-to-predict branch

Conclusions

→ μ-op cache can be used for performance improvements

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

- Identifies a hard-to-predict branch
- Prefetch critical instructions in the 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

Artifacts Available acm V1.1

Results Reproduced acm V1.1

Artifacts Evaluated Functional acm V1.1

Thank you for your attention!

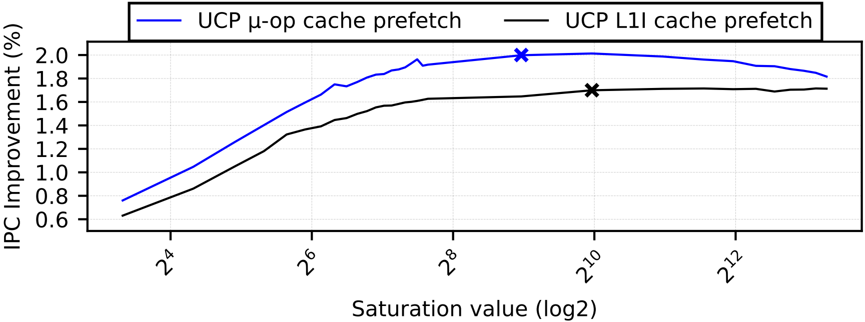
erc
European Research Council
Established by the European Commission

ECHO, ERC Consolidator Grant (No 819134)

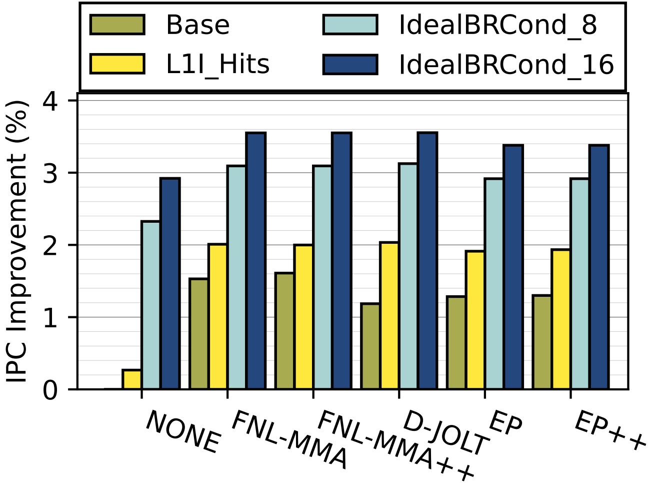
BACKUP SLIDES[BTB BANKING]

- → 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]


Base
L1I_Hits
IdealBRCond_8
IdealBRCond_16
IPC Improvement (%)
NONE
FNL-MMA
FNL-MMA++
D-JOLT
EP
EP++