# Instruction-Aware Cooperative TLB and Cache Replacement Policies


*Dimitrios Chasapis[1], Georgios Vavouliotis,*

*Daniel A. Jiménez[1,2], Marc Casas[1,3]*

*[1]Barcelona Supercomputing Center*

*[2]Texas A&M University*

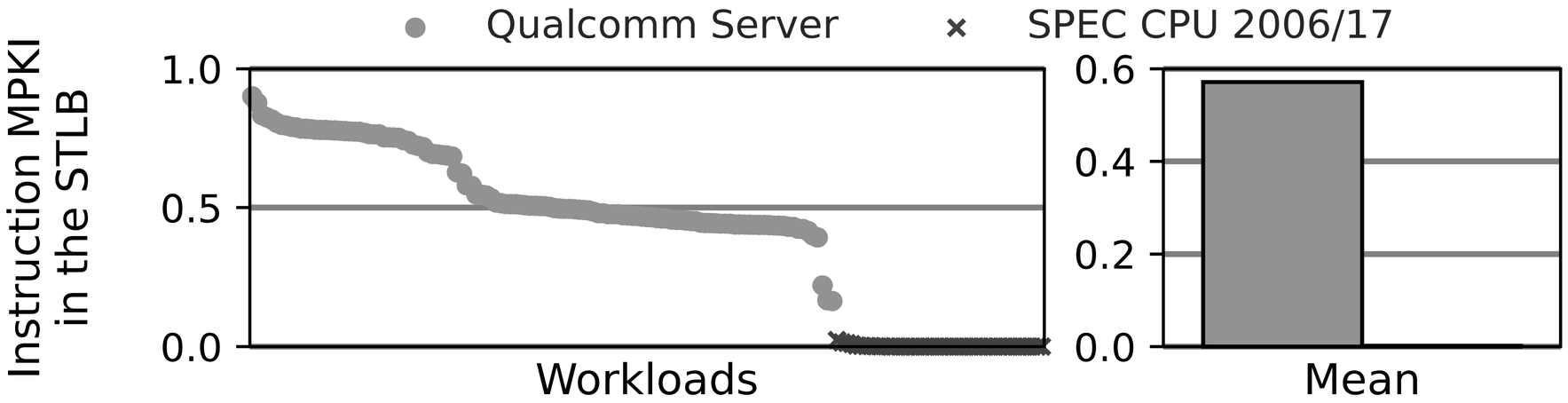
*[3]Universitat Politècnica de Catalunya*

dimitrios.chasapis@bsc.es

# Content

- Motivation
  - Instruction Address Translation
  - Prioritizing Instruction Address Translations
- Translation-Aware Replacement policy Design
  - Instruction Translation Prioritization (iTP)
  - Extended Page Table Prioritization (xPTP)
  - Combining iTP and xPTP
- Evaluation
  - Comparison with Instruction-aware Policies
  - Impact of LLC Policies
  - Page Size Impact
- Conclusions

# Instruction Address Translation




- TLB design led by HPC and Desktop applications
- Server Workloads exhibit different behavior and hardware requirements
  - Large instruction footprint → high front-end pressure (L1i, TLB)
    - Annual instruction growth rate → >20%

On the Impact of Instruction Address Translation Overhead

Yufeng Zhou[1], Xiaowan Dong[2], Alan L. Cox[1], and Sandhya Dwarkadas[2]

BOLT: A Practical Binary Optimizer for Data Centers and Beyond

Maksim Panchenko, Rafael Auler, Bill Nell, Guilherme Ottoni
Facebook, Inc.

**Profiling a warehouse-scale computer**

Svilen Kanev[†] Harvard University
Juan Pablo Darago[†] Universidad de Buenos Aires
Kim Hazelwood[†] Yahoo Labs
Parthasarathy Ranganathan Google
Tipp Moseley Google
Gu-Yeon Wei Harvard University
David Brooks Harvard University

**Morrigan: A Composite Instruction TLB Prefetcher**

Georgios Vavouliotis
georgios.vavouliotis@bsc.es
*Barcelona Supercomputing Center*
*Universitat Politècnica de Catalunya*

Lluc Alvarez
lluc.alvarez@bsc.es
*Barcelona Supercomputing Center*
*Universitat Politècnica de Catalunya*

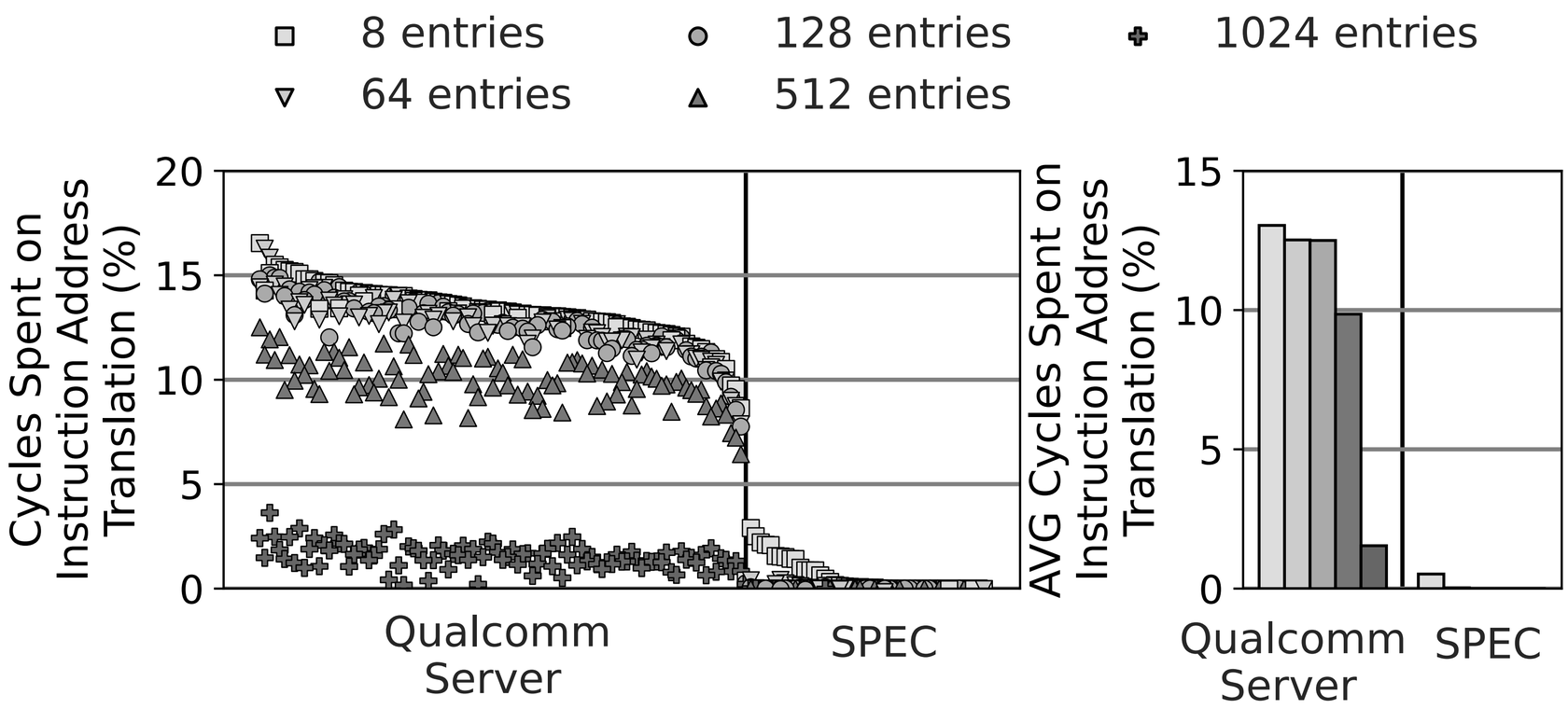
Boris Grot
boris.grot@ed.ac.uk
*University of Edinburgh*

Daniel A. Jiménez
djimenez@acm.org
*Texas A&M University*

Marc Casas
marc.casas@bsc.es
*Barcelona Supercomputing Center*
*Universitat Politècnica de Catalunya*

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

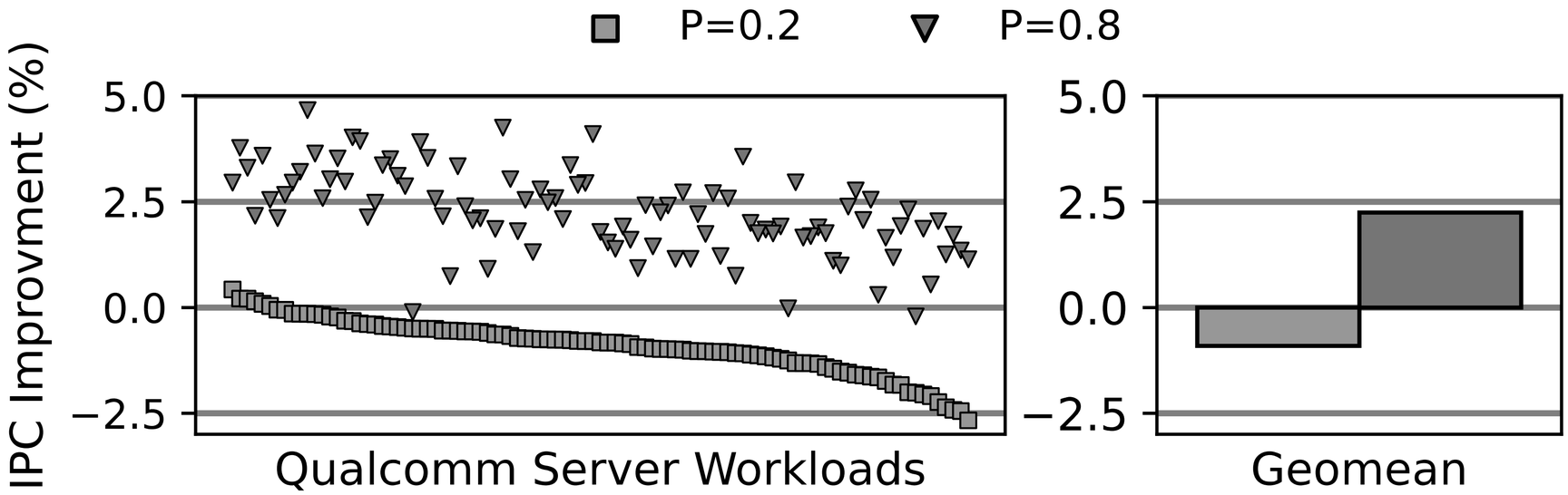
# Instruction Address Translation




- Applications with large code footprints amplify the address translation overheads!
- Larger ITLBs could mitigate the cost of larger instruction footprints
  - High latency makes larger ITLBs unrealistic

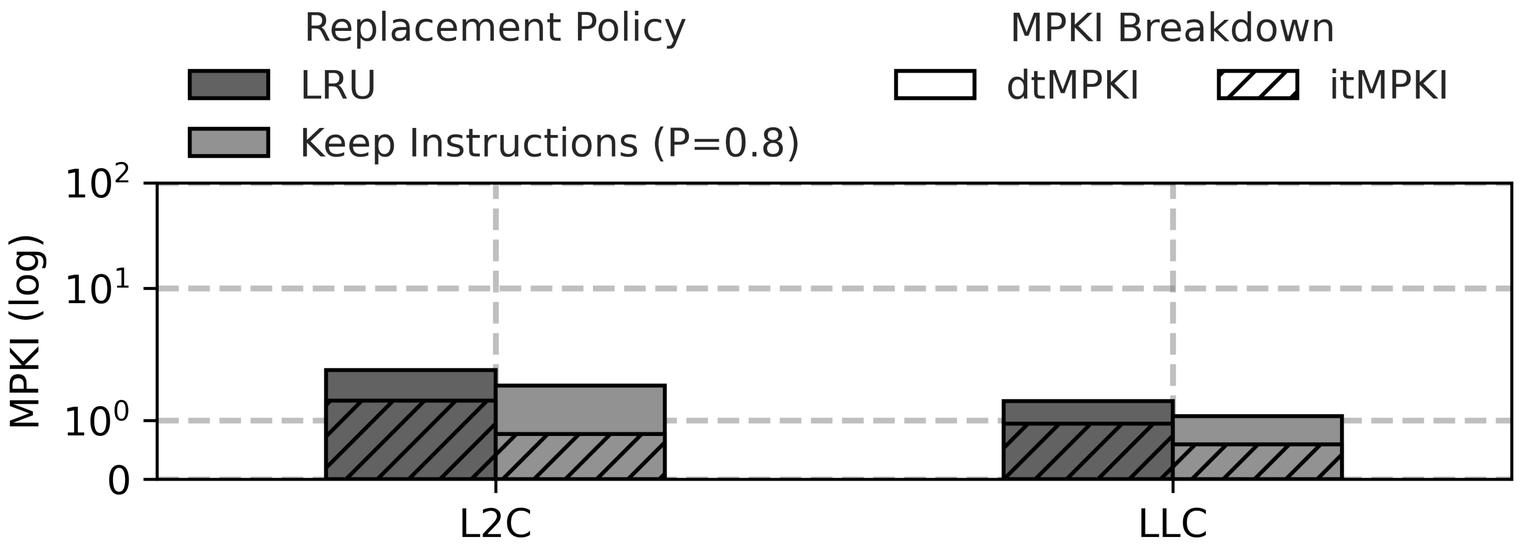
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Prioritizing Instruction Address Translations




- Naïve replacement policy that favors keeping instructions over data translations with probability P.
- Prioritizing instructions over data in the STLB can provide significant performance enhancements for workloads with large instruction footprints

# Prioritizing Instruction Address Translations



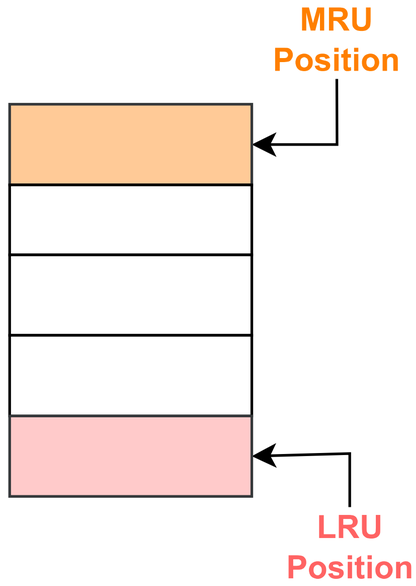

- Prioritizing instructions over data in the STLB increases the cache misses triggered by page walks serving data STLB misses
- This side effect can be observed as an increase in data PTE misses in the cache hierarchy

# Content

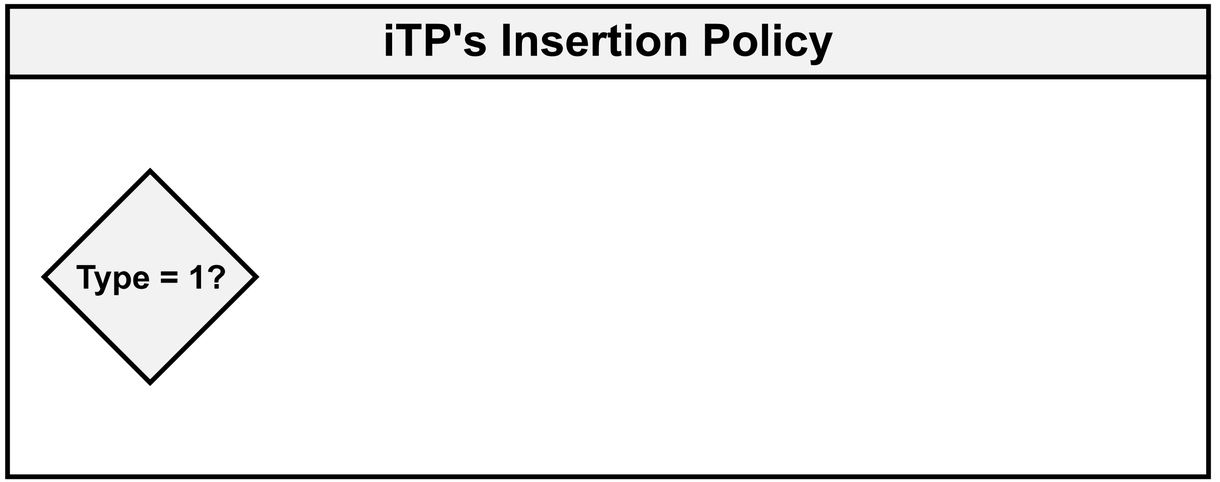
- Motivation
  - Instruction Address Translation
  - Prioritizing Instruction Address Translations
- Translation-Aware Replacement policy Design
  - Instruction Translation Prioritization (iTP)
  - Extended Page Table Prioritization (xPTP)
  - Combining iTP and xPTP
- Evaluation
  - Comparison with Instruction-aware Policies
  - Impact of LLC Policies
  - Page Size Impact
- Conclusions

# Instruction Translation Prioritization (iTP) - Overview

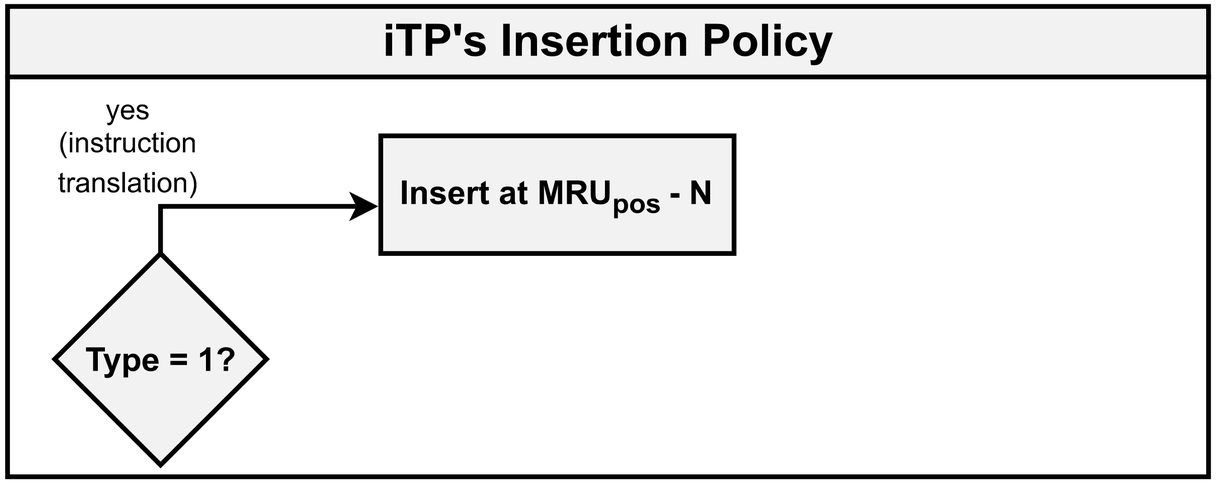
- Goal: Maximize instruction translation hits in the STLB
  - Mitigate the pressure of large instruction footprints in the STLB
  - Reduce frontend stalls attributed to instruction translation overheads
- Instruction translations get favorable treatment
  - Higher position in the recency stack
- Modified version of LRU
  - Eviction candidate is still the entry at the LRU position of the stack
  - Insertion (on miss) and promotion (on hit) of entry positions are modified



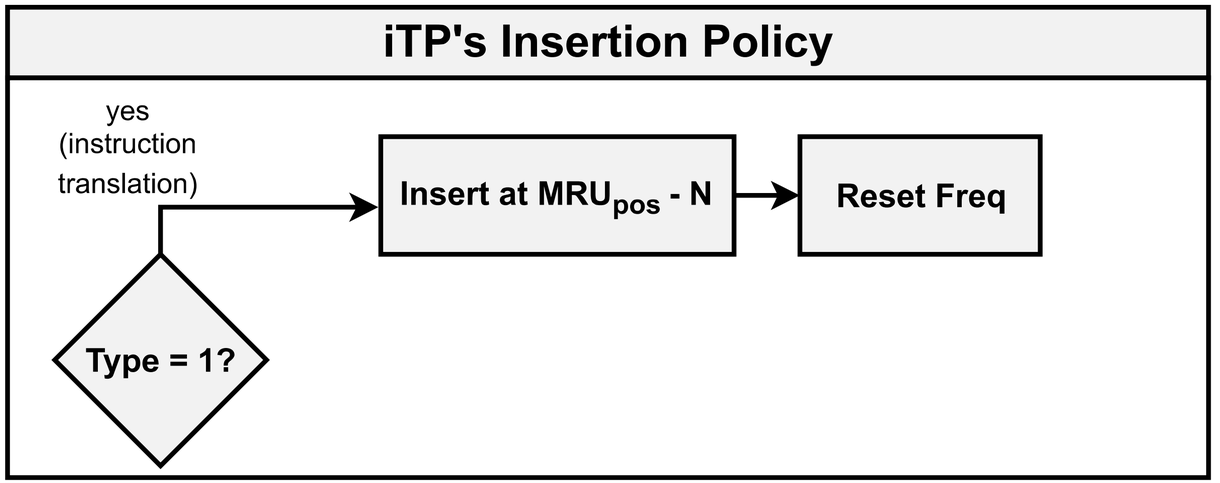
# Instruction Translation Prioritization (iTP) - Insertion



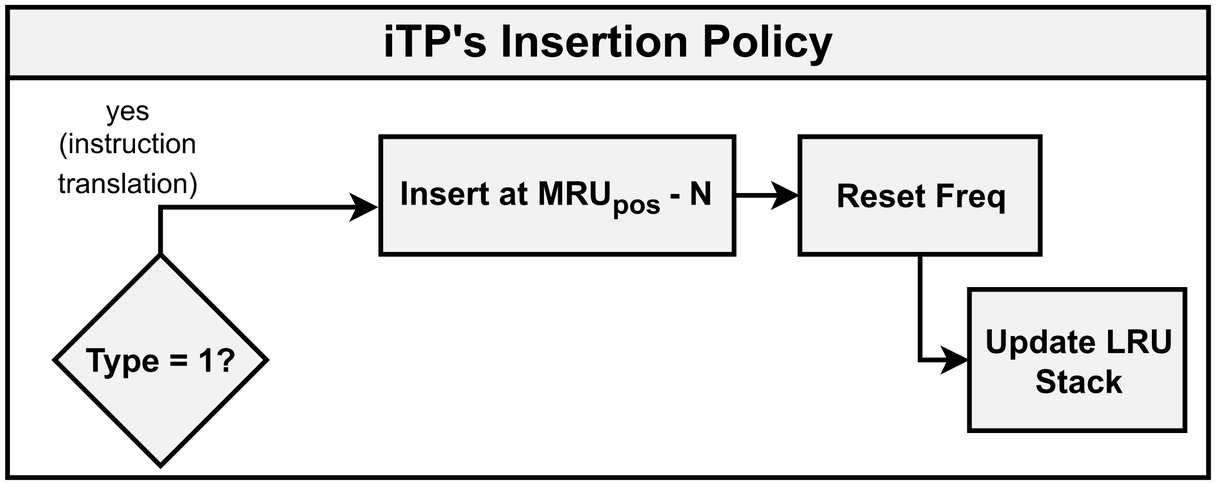
# Instruction Translation Prioritization (iTP) - Insertion



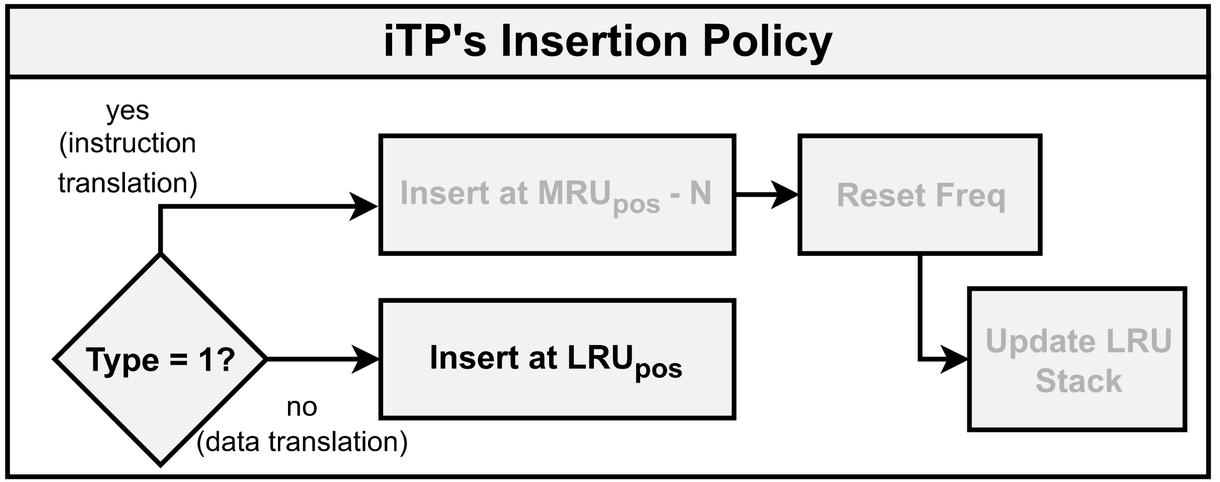
# Instruction Translation Prioritization (iTP) - Insertion



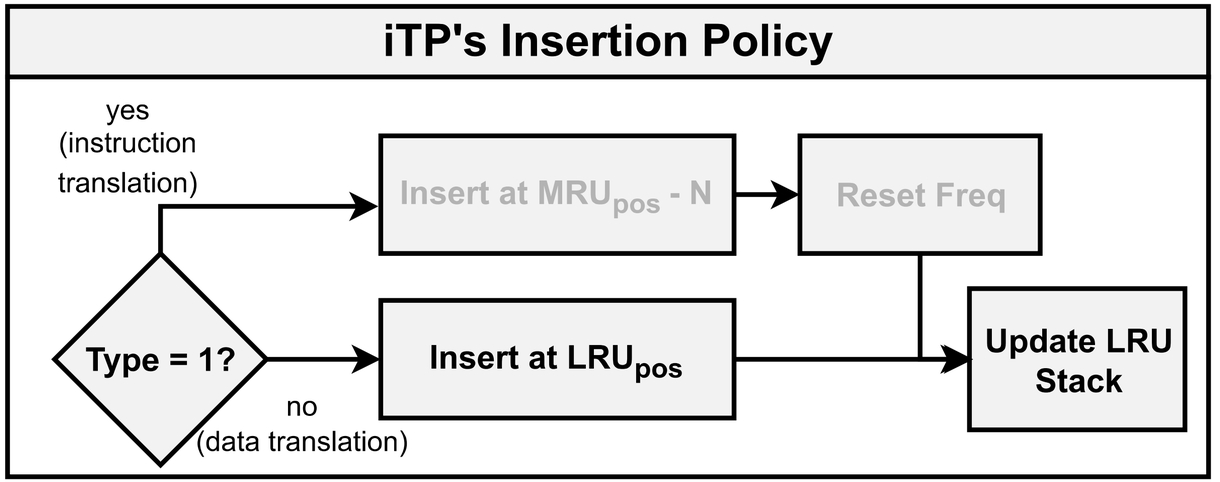
# Instruction Translation Prioritization (iTP) - Insertion



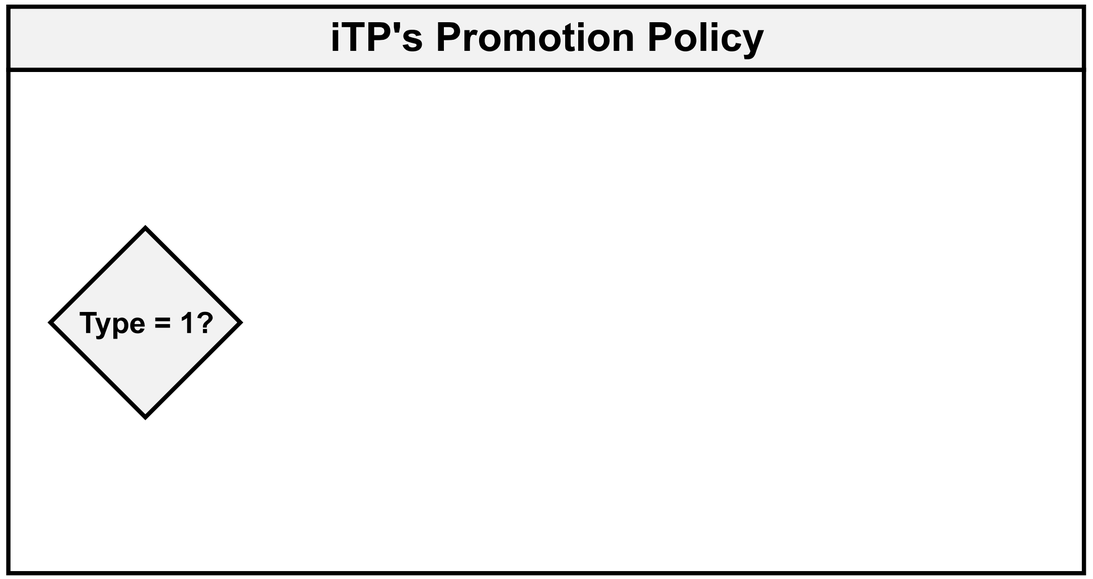
# Instruction Translation Prioritization (iTP) - Insertion



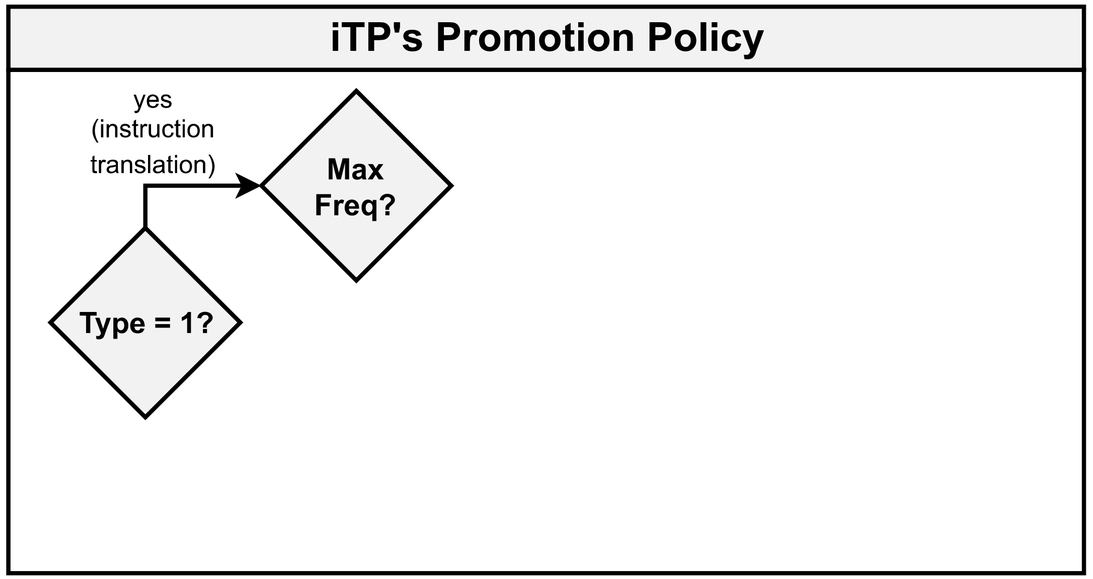
# Instruction Translation Prioritization (iTP) - Insertion



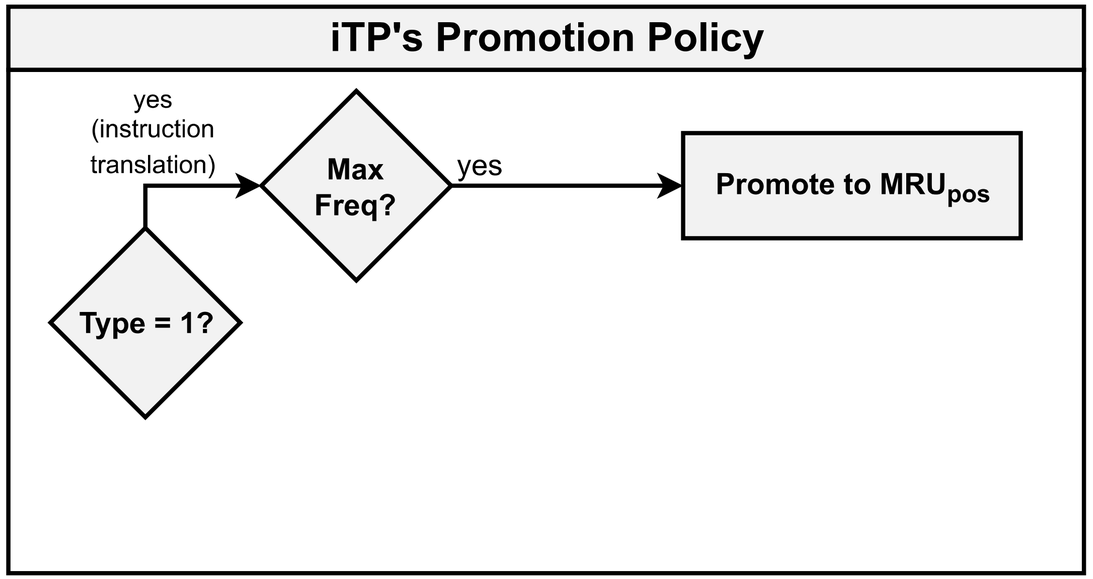
# Instruction Translation Prioritization (iTP) - Promotion



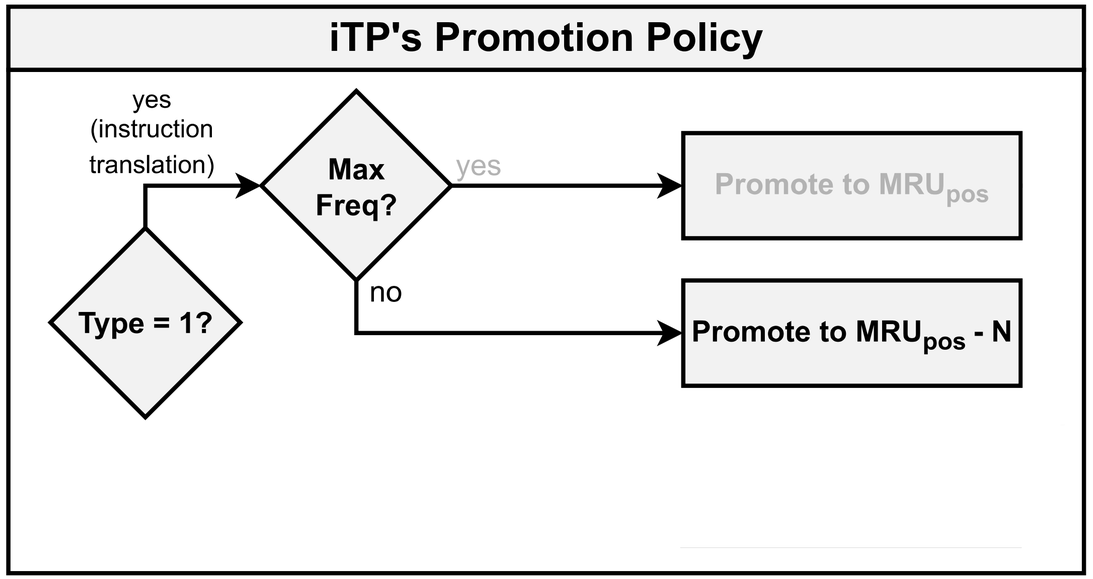
# Instruction Translation Prioritization (iTP) - Promotion



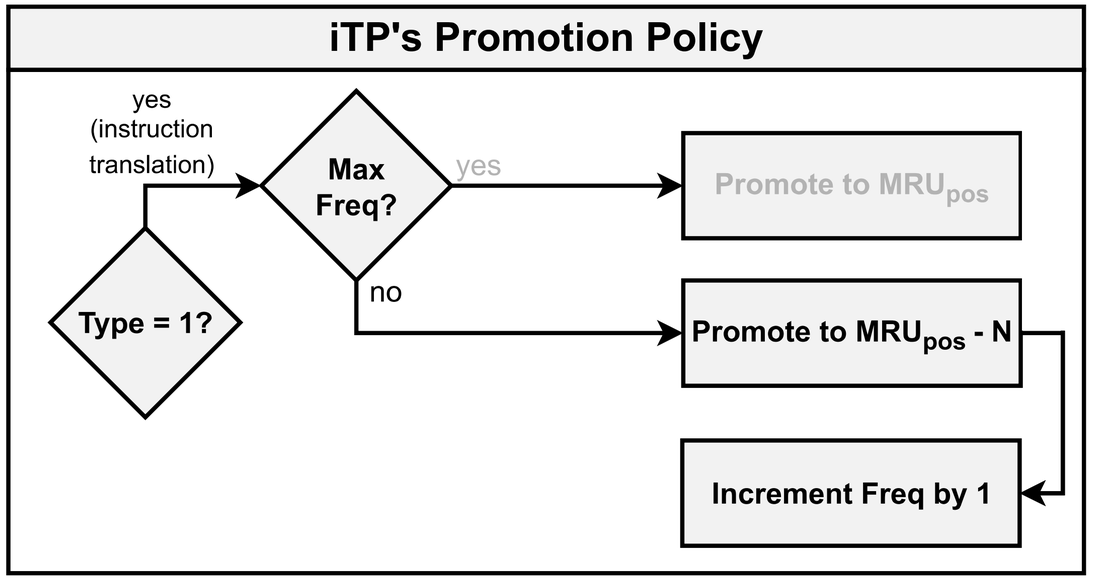
# Instruction Translation Prioritization (iTP) - Promotion



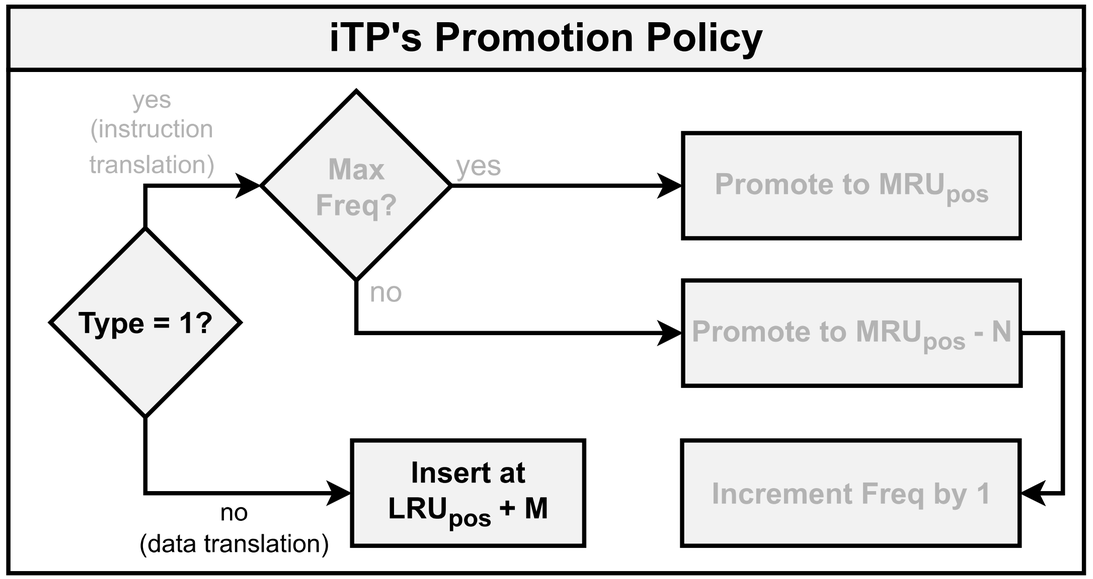
# Instruction Translation Prioritization (iTP) - Promotion



# Instruction Translation Prioritization (iTP) - Promotion

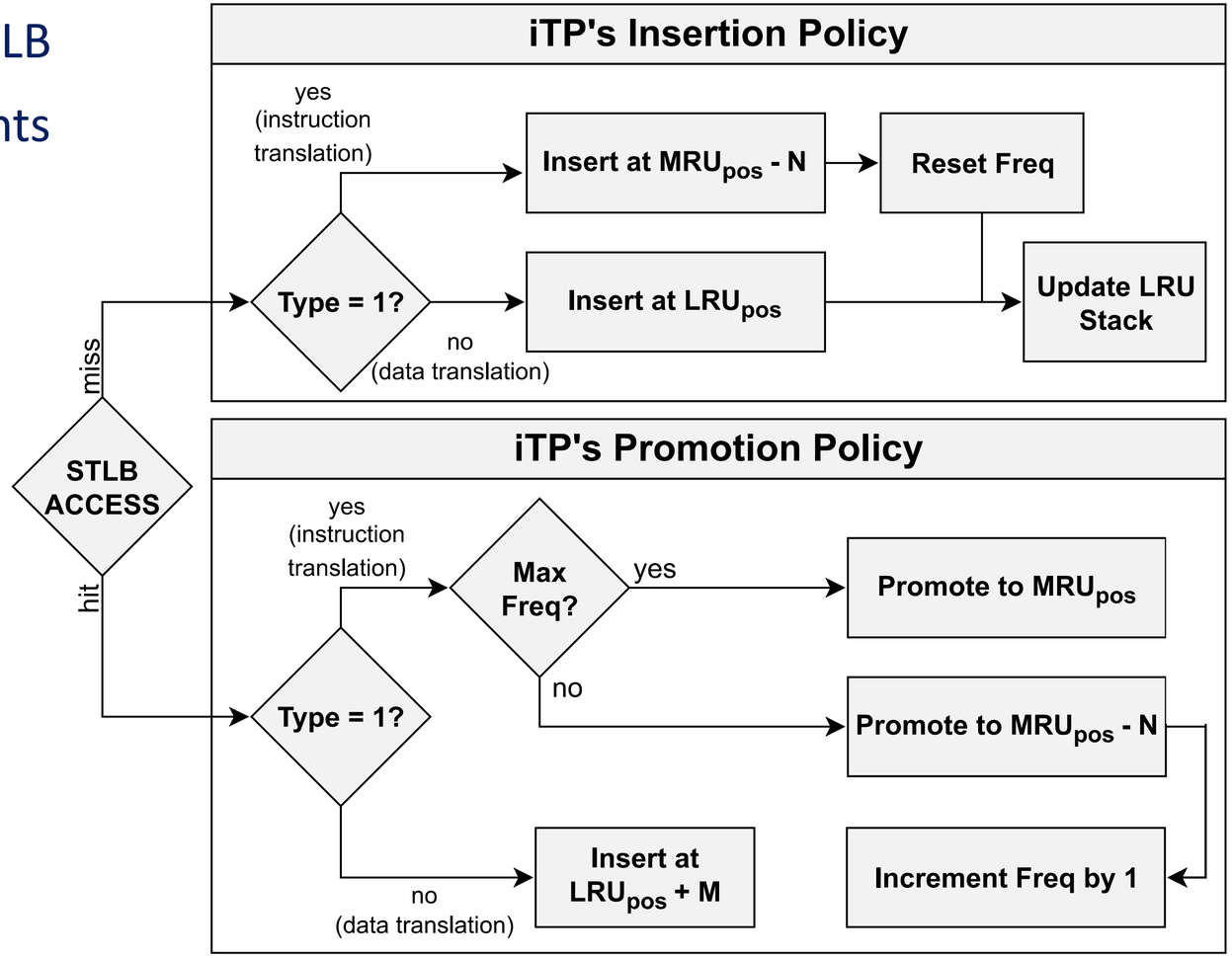


# Instruction Translation Prioritization (iTP) - Promotion



# Instruction Translation Prioritization (iTP) - Highlights

- Goal: Maximize instruction translation hits in the STLB
  - Mitigate the pressure of large instruction footprints in the STLB
  - Reduce frontend stalls attributed to instruction translation overheads
- Instruction translations get favorable treatment.
  - Higher position in the recency stack
- Side effect: Increases data PTE cache misses!
- Hardware budget?
  - Coming up!



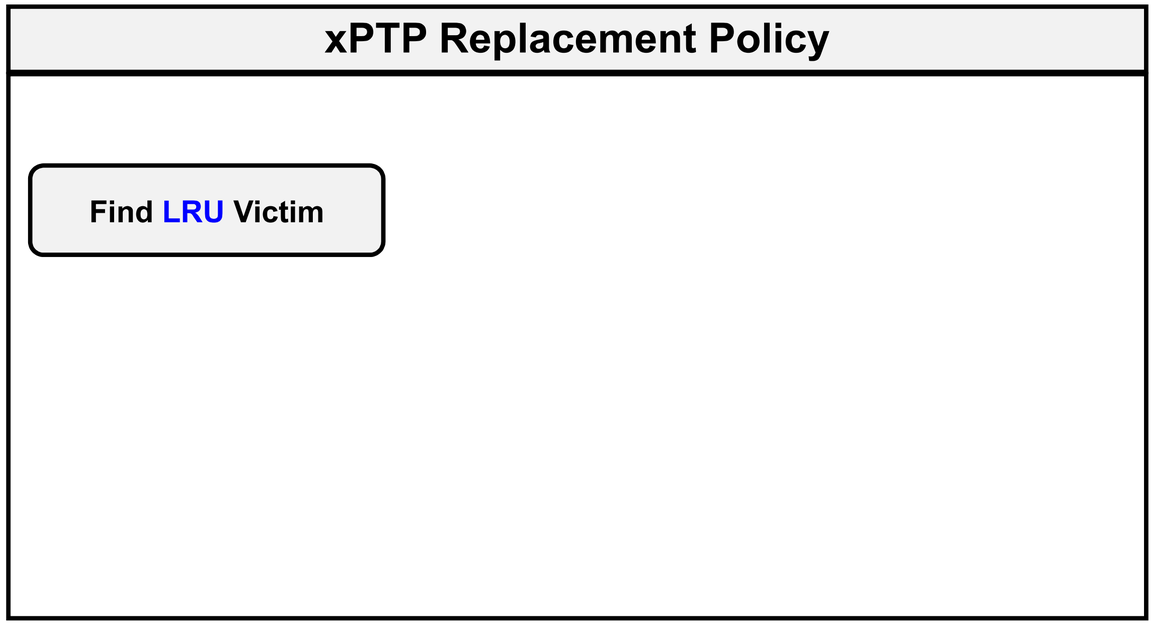
# Extended Page Table Prioritization (xPTP) - Overview

- Complementary to iTP
  - Mitigates the increase in data translation misses caused by iTP
  - Favors blocks that contain data PTEs.

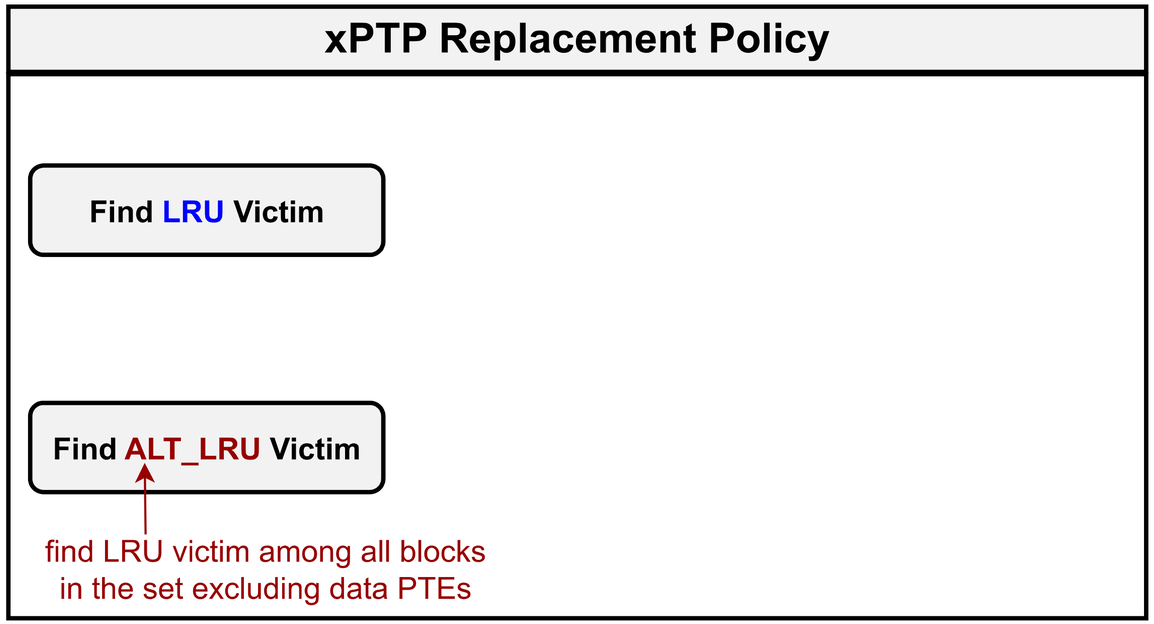
- Based on LRU (again!)
  - Maintaining two different recency stacks
    - Normal recency stack
    - Recency stack excluding PTEs that relate to data translations

- Applied to cache (L2C)

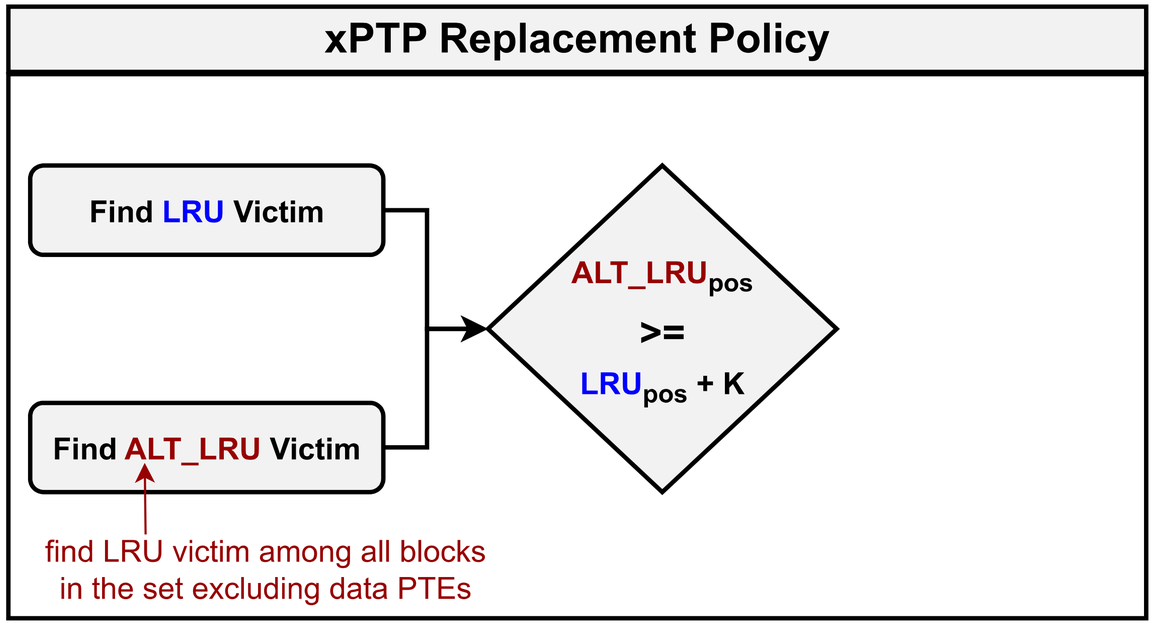
# Extended Page Table Prioritization (xPTP) - Eviction



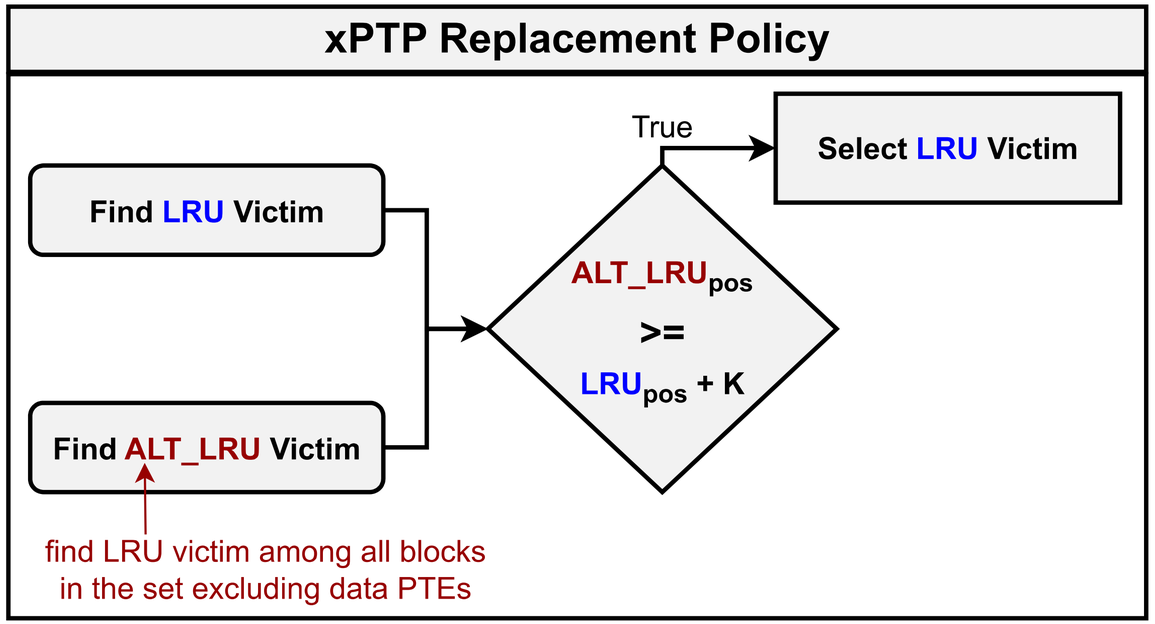
# Extended Page Table Prioritization (xPTP) - Eviction



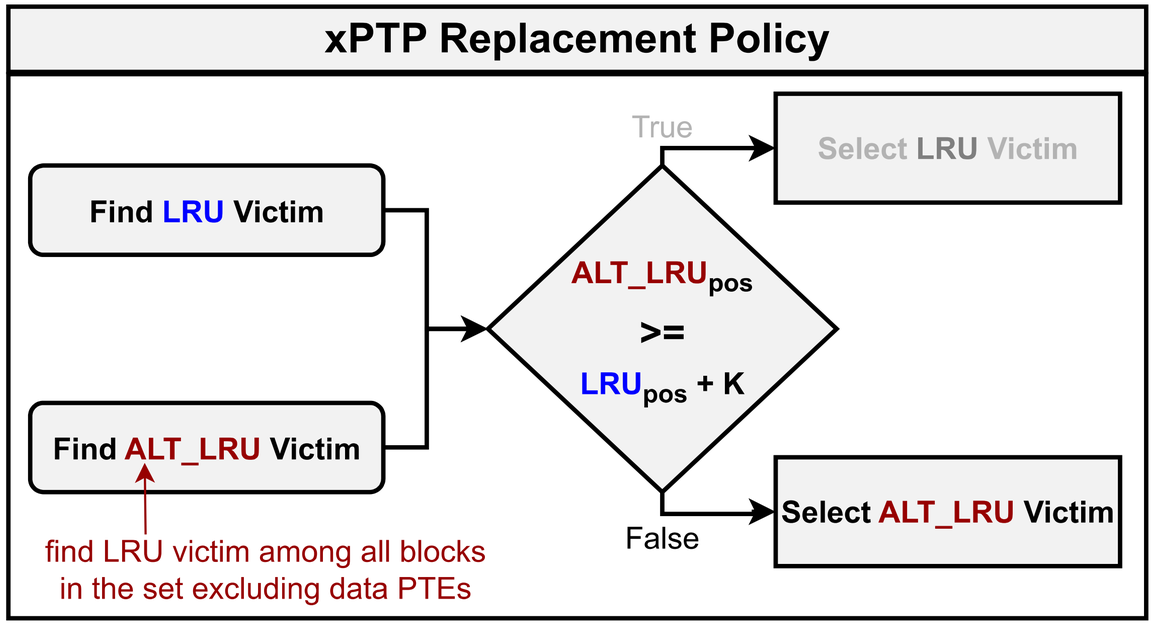
# Extended Page Table Prioritization (xPTP) - Eviction



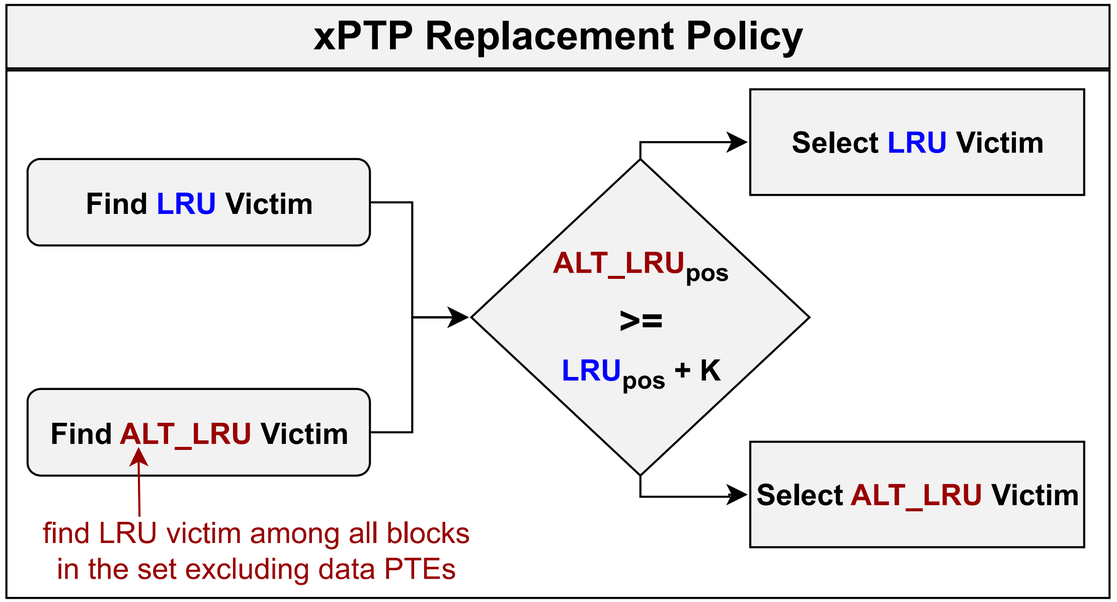
# Extended Page Table Prioritization (xPTP) - Eviction



# Extended Page Table Prioritization (xPTP) - Eviction



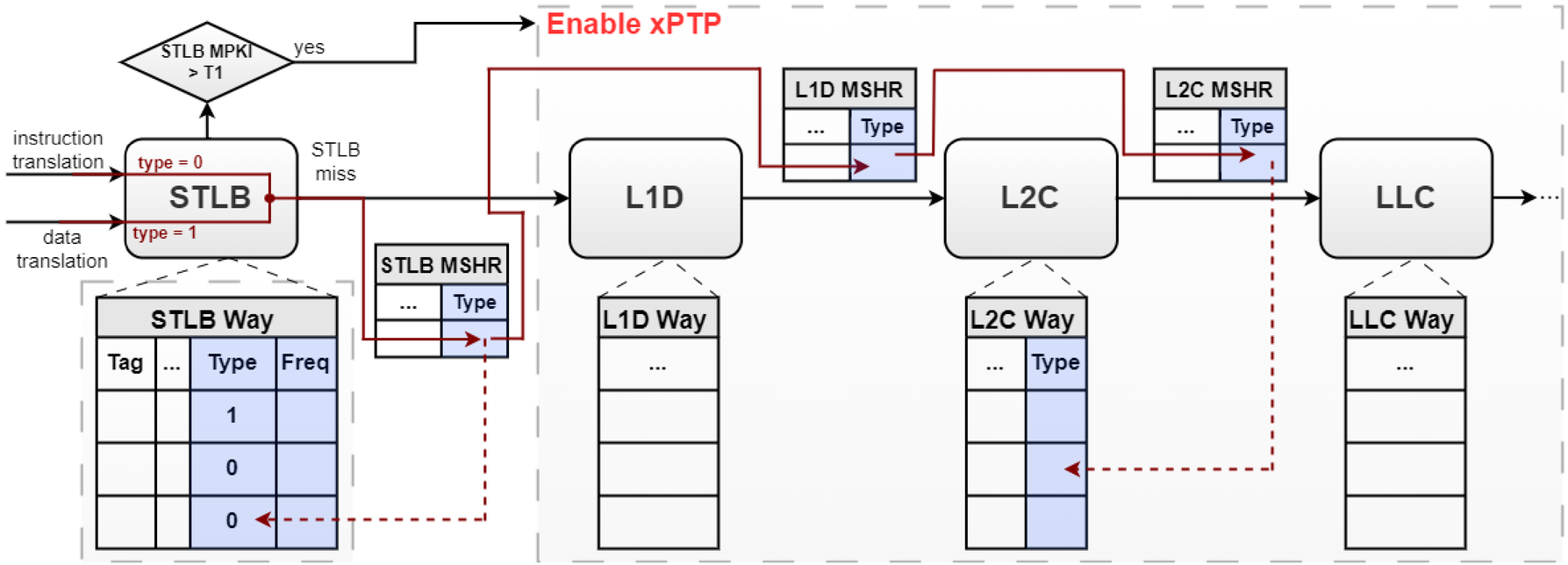
# Extended Page Table Prioritization (xPTP) - Highlights




- Designed to mitigate the increase in data translation misses caused by iTP
- Applied to L2C
- Favors blocks that contain data translations
- Value for K has been selected empirically based on a sensitivity analysis

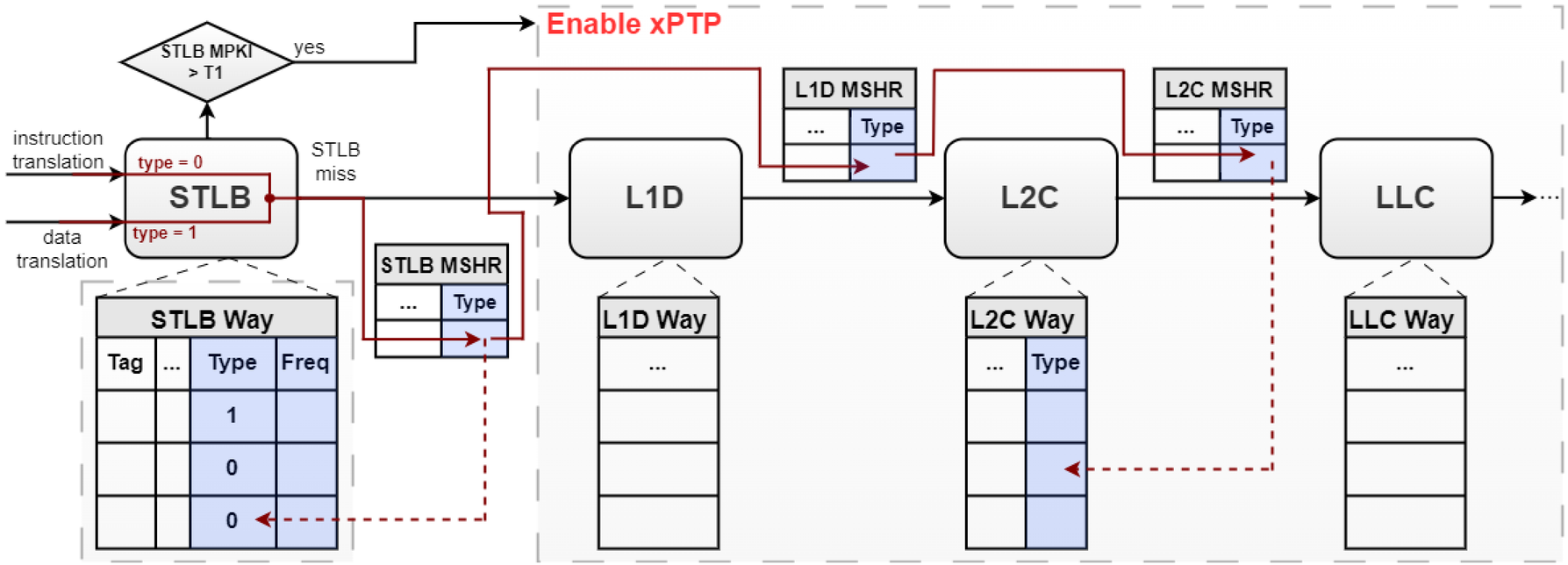
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Cooperative Replacement Policy Design (iTP+xPTP)




- Combined design
  - Improves instruction throughput while mitigating the negative effect of increased data translation misses
- Phase Adaptability: Results suggest that xPTP is only beneficial when there is high STLB pressure
  - Our policy monitors the miss rate of STLB and enable/disables xPTP accordingly

# Cooperative Replacement Policy Design (iTP+xPTP)




- Hardware Budget:
  - iTP: 4bits per entry – 1 for type and 3 for the saturating counter
  - xPTP: 1 bit per entry to track the type

# Content

- Motivation
  - Instruction Address Translation
  - Prioritizing Instruction Address Translations
- Translation-Aware Replacement policy Design
  - Instruction Translation Prioritization (iTP)
  - Extended Page Table Prioritization (xPTP)
  - Combining iTP and xPTP
- Evaluation
  - Comparison with Instruction-aware Policies
  - Impact of LLC Policies
  - Page Size Impact
- Conclusions

# Evaluation – Experimental Setup

More in the Paper!

- Simulation Infrastructure:
  - ChampSim trace-based simulator (https://github.com/ChampSim/ChampSim)
  - SMT-1 Traces:
    - SPEC CPU 2006/2007
    - Qualcomm Server Workloads (CVP-1, IPC-1) [1]
  - SMT-2 Traces:
    - Combination of SMT-1 traces
    - Three categories with randomly chosen benchmarks:
      - Two benchmarks with high STLB MPKI
      - One with high and one with medium STLB MPKI
      - One with high and one with low STLB MPKI

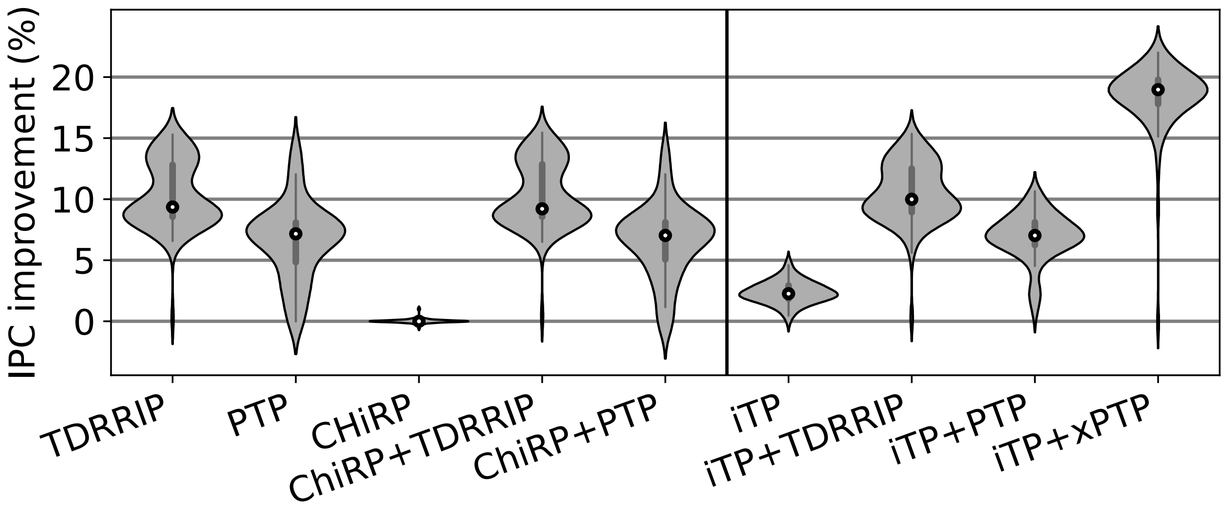
| Component | Configuration |
|---|---|
| CPU CORE | 4GHz, 352-etnry ROB |
| FETCH-TARGET-QUEUE (FTQ) | 128-entry |
| L1 ITLB | 64-entry, 4-way, 1-cycle, 8-entry MSHR |
| L1 DTLB | 64-entry, 4-way, 1-cycle, 8-entry MSHR |
| L2 TLB | 1536-entry, 12-way, 8-cycle,<br>16-entry MSHR, 1 page walk / cycle<br>iTP: 3-bit Freq counter, 1-bit Type, N=4, M=8 |
| PAGE STRUCTURE CACHES | 5-level Split PSC, 2-cycle.<br>PSCL5: 2-entry, fully; PSCL4: 4-entry, fully<br>PSCL3: 8-entry, 2-way; PSCL2: 32-entry, 4-way. |
| L1I CACHE | 32KB, 8-way, 4-cycle, 8-entry MSHR, VIPT, FDiP |
| L1D CACHE | 32KB, 12-way, 5-cycle, 8-entry MSHR,<br>VIPT, Next-Line prefetcher |
| L2 CACHE | 512KB, 8-way, 5-cycle, 32-entry MSHR,<br>PIPT, Stride prefetcher, xPTP: 1-bit Type, K=8 |
| LAST LEVEL CACHE | 2MB, 16-way, 10-cycle, 64-entry MSHR, PIPT |
| DRAM | tRP=tRCD=tCAS=12, 12.8 GB/s |
| BRANCH PREDICTOR | Hashed Perceptron |



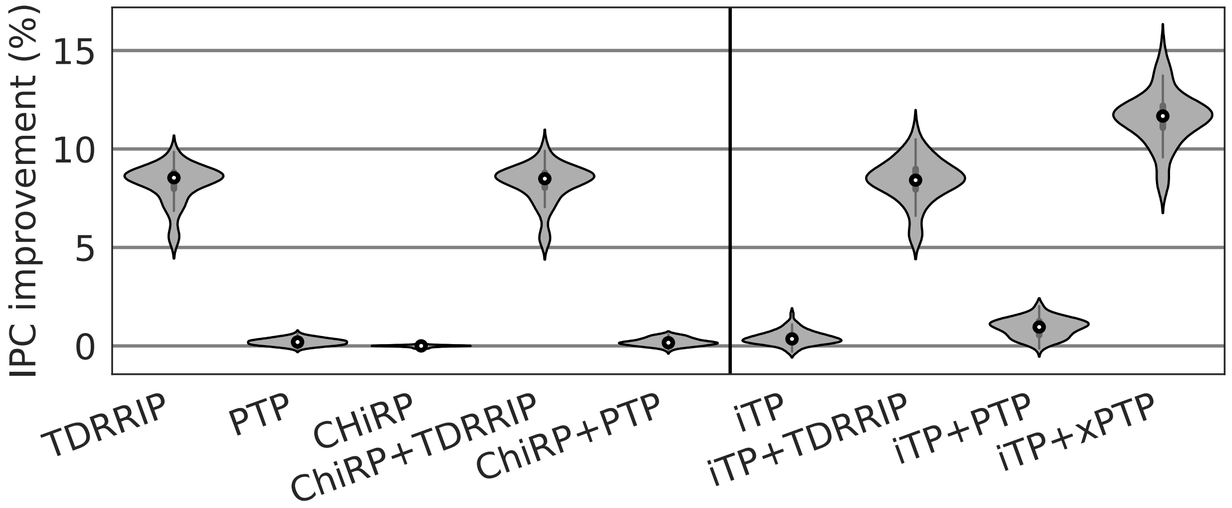
[1]J. Feliu, A. Perais, D. A. Jiménez and A. Ros, "Rebasing Microarchitectural Research with Industry Traces," *2023 IEEE International Symposium on Workload Characterization (IISWC)*, Ghent, Belgium, 2023, pp. 100-114, doi: 10.1109/IISWC59245.2023.00027.

# Evaluation – Comparison with Translation-aware Policies

Single Hardware Thread



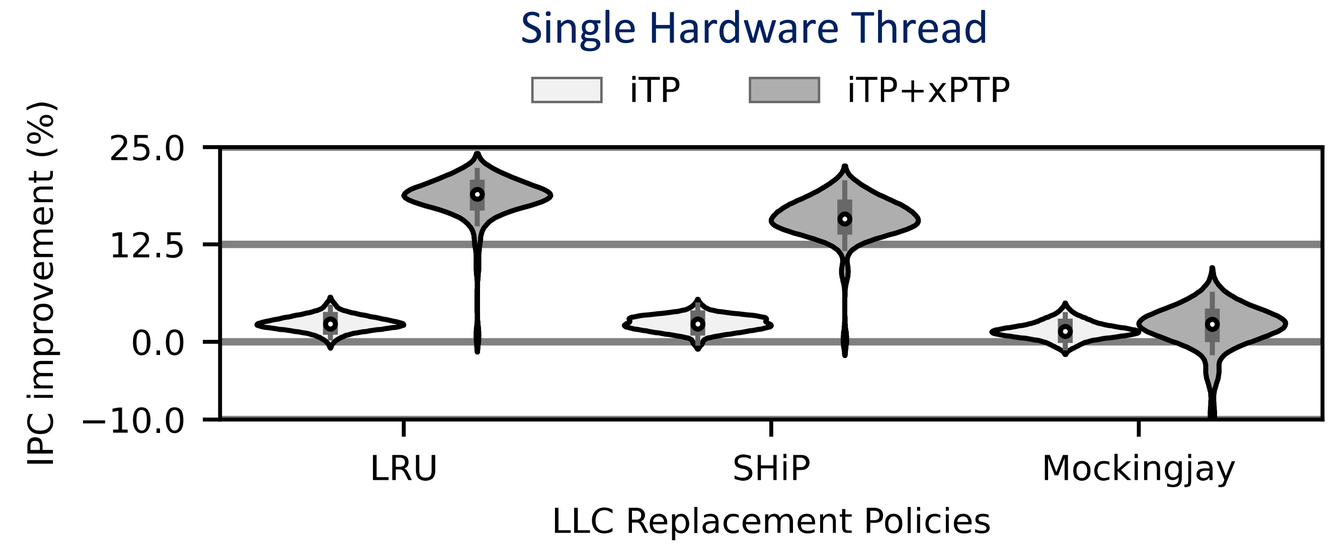

Two Hardware Threads

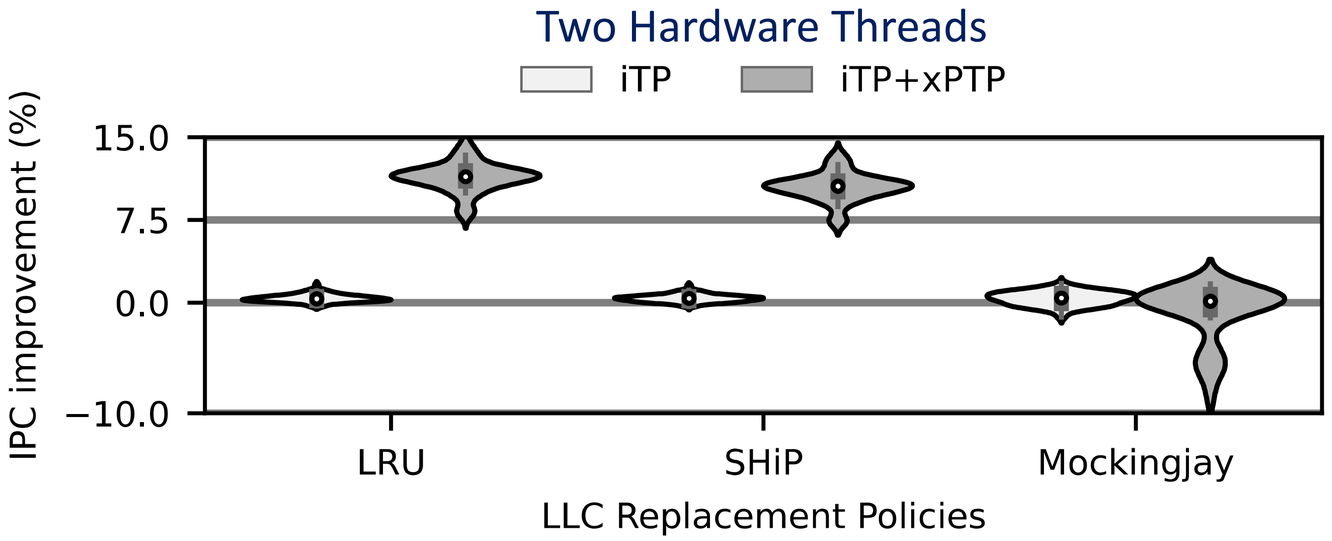



- iTP provides modest speedup in both singe and two hardware threads 2.2% and 0.3% respectively
  - Reduces instruction translation cost and STLB MPKI and miss latency
  - Increases page walks due to data translation misses
- Results are amplified when combining iTP+xPTP, 18.9% and 11.4% for single and two hardware threads
  - iTP+xPTP manage to reduce L2C average miss latency and LLC MPKI (reduced page walks)

# Evaluation – Impact of LLC Policies

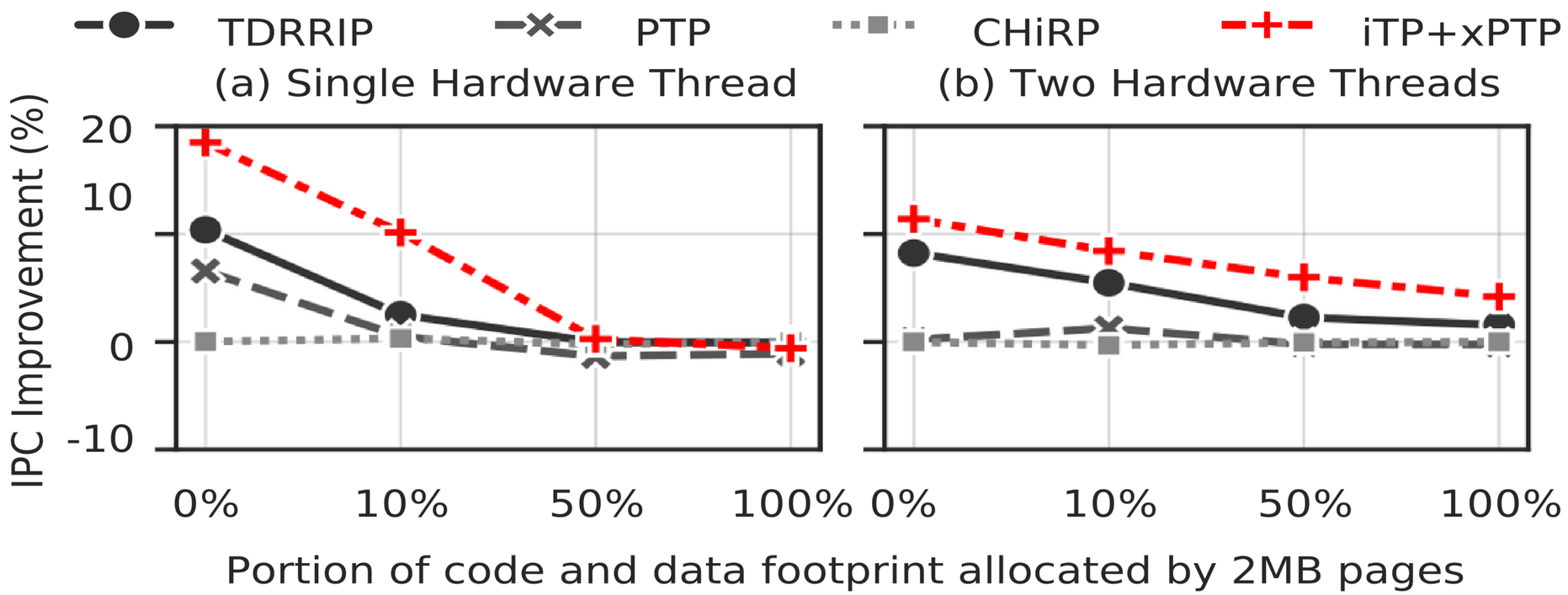
- iTP+xPTP reduces MPKI in the LLC, due to the reduction in page walks
- Improves performance under any LLC replacement policy
- We have observed that Mockingjay performed poorly on the Qualcomm Server Workloads, often resulting in slowdown over the LRU baseline
  - Our technique improves results, but the speedup remains low




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Evaluation – Page Size Impact




- Speedup reduced as more large pages are used
  - Less stress to the STLB
- iTP+xPTP maintain higher speedups over the other techniques, even as more large pages are used

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Conclusions

- Applications with large instruction footprints suffer significant performance degradation due to pipeline stalls caused from TLB misses
- Favoring instructions over data translations (STLB) improves performance for such applications
  - Side effect is an increase in page walks due to data translations misses
- Replacement policies that favor data PTEs in the cache hierarchy can alleviate this side effect
- The proposed instruction-aware replacement policies improve performance on server applications
  - Offer higher performance gains than other state-of-the-art translation-aware replacement policies
- iTP+xPTP outperform other techniques even when code and data are mapped to large pages

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Generalitat de Catalunya

# Thank you for your attention!

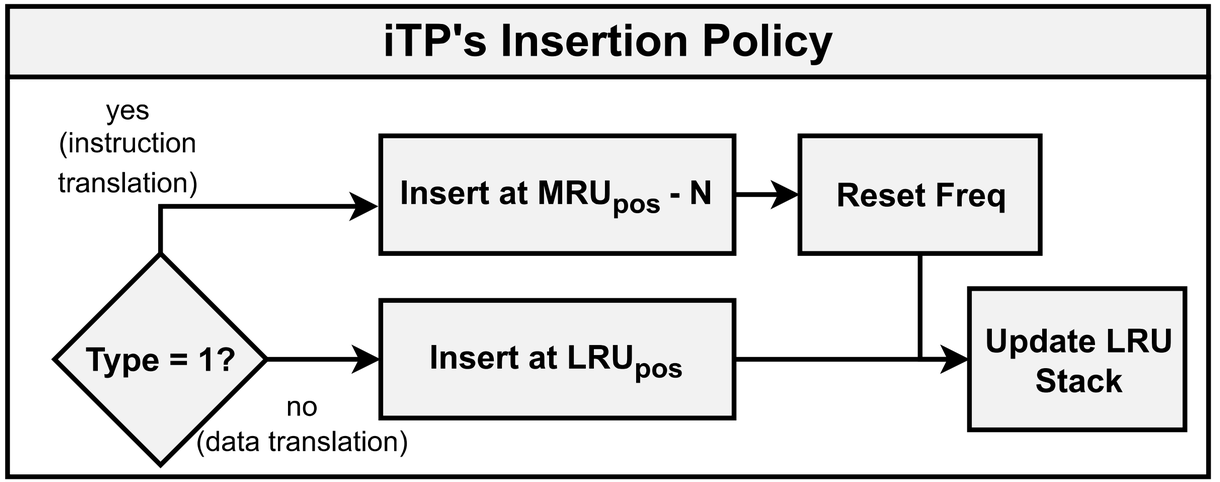
# Acknoledgments/ Fundings


- The authors are grateful to the anonymous reviewers for their valuable comments and constructive feedback that significantly improved the quality of the paper.
- This work has received funding from ‘Future of Computing, a Barcelona Supercomputing Center and IBM initiative’ (2023).
- This work has been partially supported by the Spanish Ministry of Science and Innovation MCINAEI/10.13039/501100011033 (contract PID2019-107255GB-C21) and ESF Investing in your future, the Generalitat of Catalunya (contract 2021-SGR-00763), the European HiPEAC Network of Excellence, and the European Processor Initiative (EPI), which is part of the European Union’s Horizon 2020 research and innovation program under grant agreement No. 826647 This work is supported by the National Science Foundation through grant CCF-1912617 and generous gifts from Intel.
- The authors thank the Departament de Recerca i Universitats de la Generalitat de Catalunya for supporting the Research Group "Performance understanding, analysis, and simulation/emulation of novel architectures" (Code: 2021 SGR 00865).

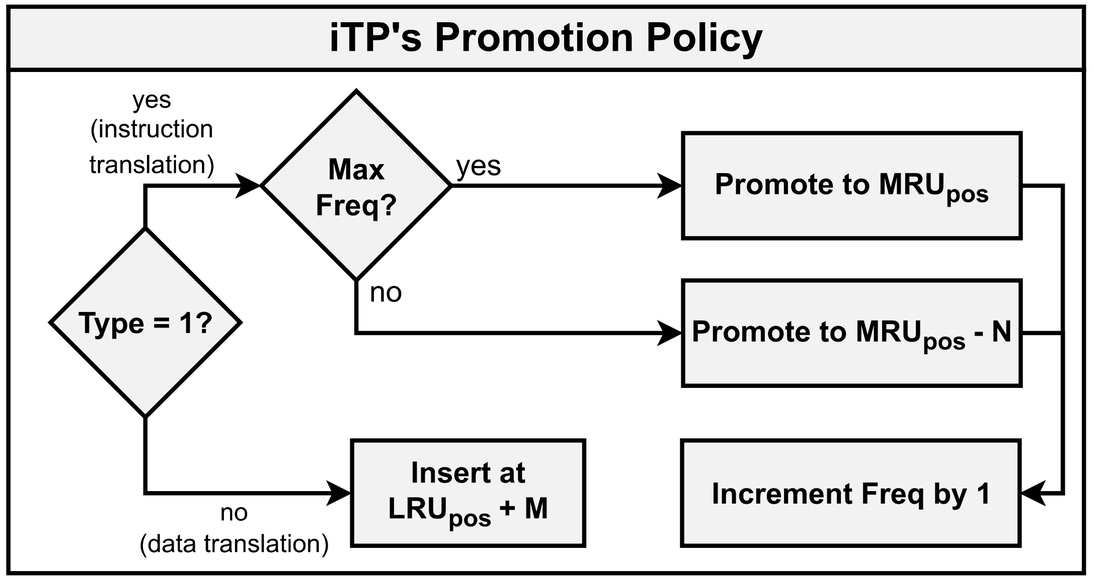
# BACKUP SLIDES


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

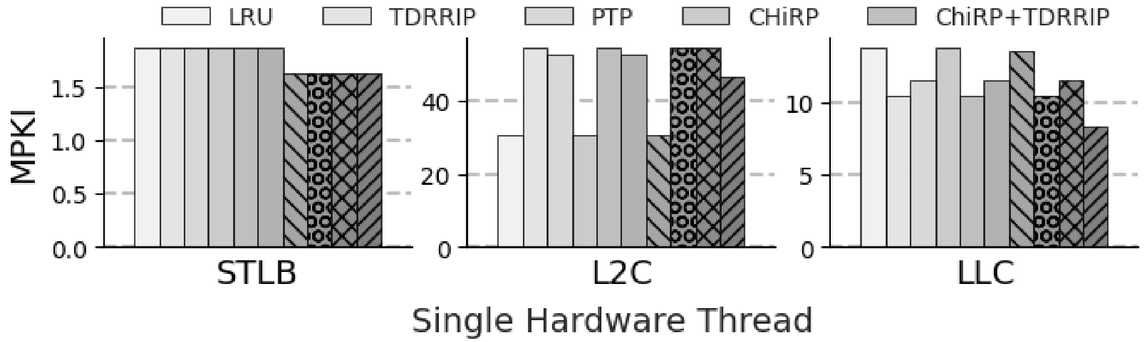
# Instruction Translation Prioritization (iTP) - Insertion

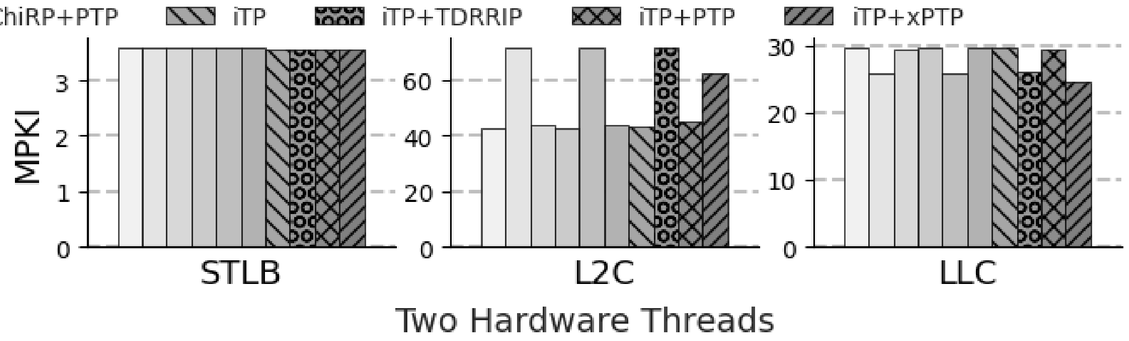


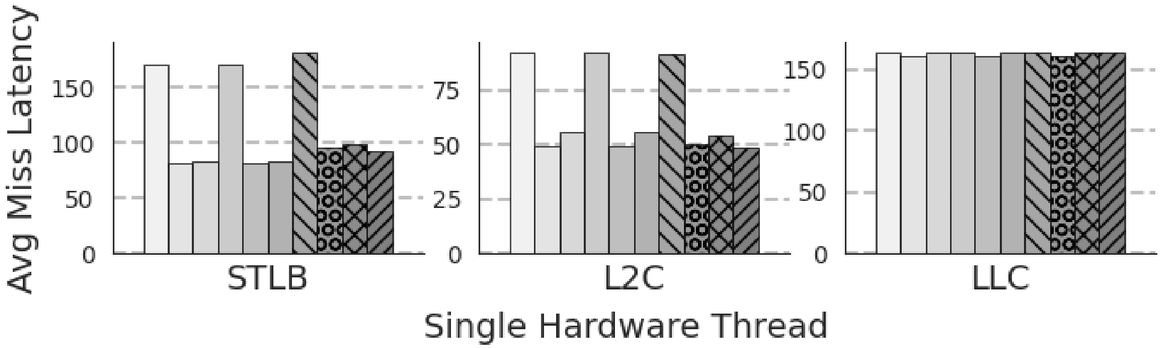
# Instruction Translation Prioritization (iTP) - Promotion

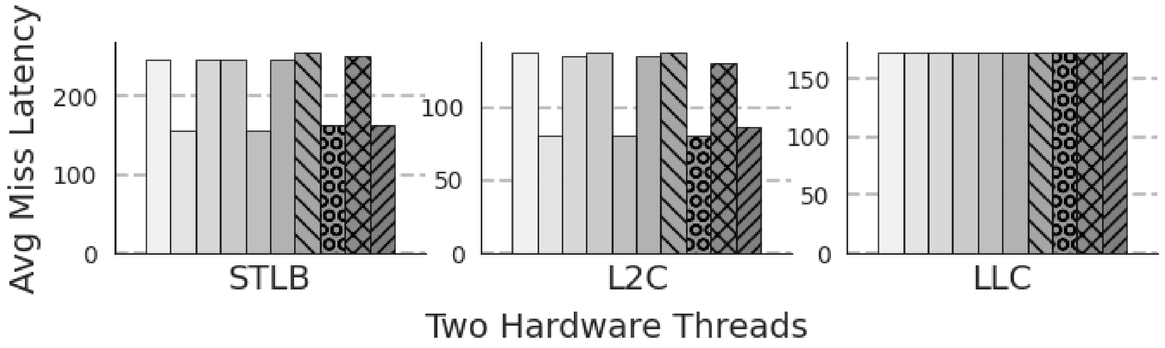


# Extra: MPKI and Latency Results



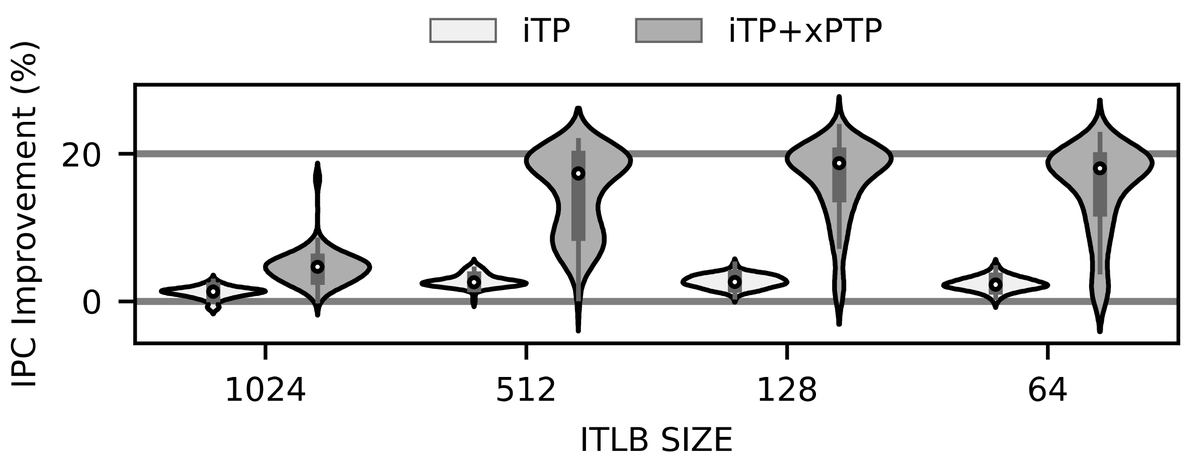



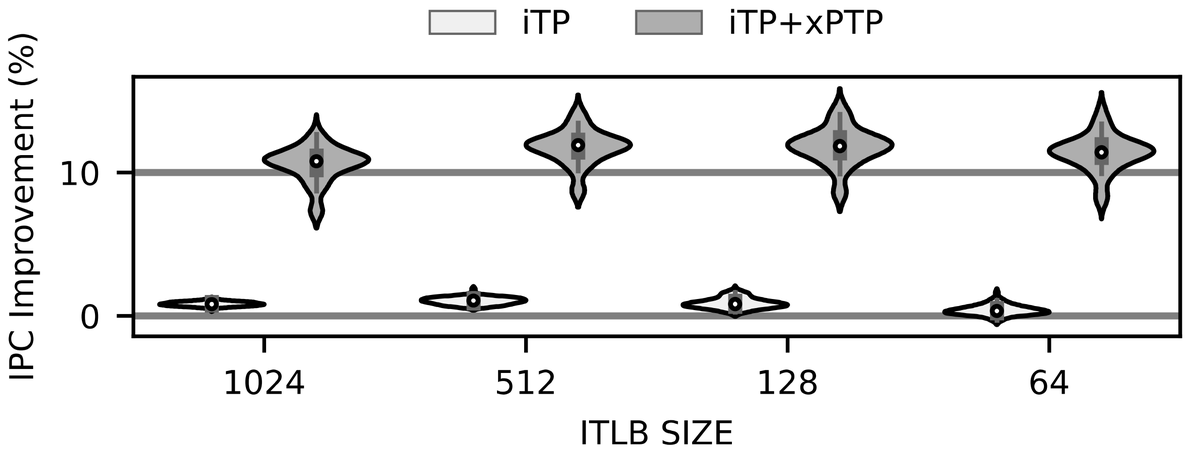

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Evaluation – ITLB Size Impact

Single Hardware Thread

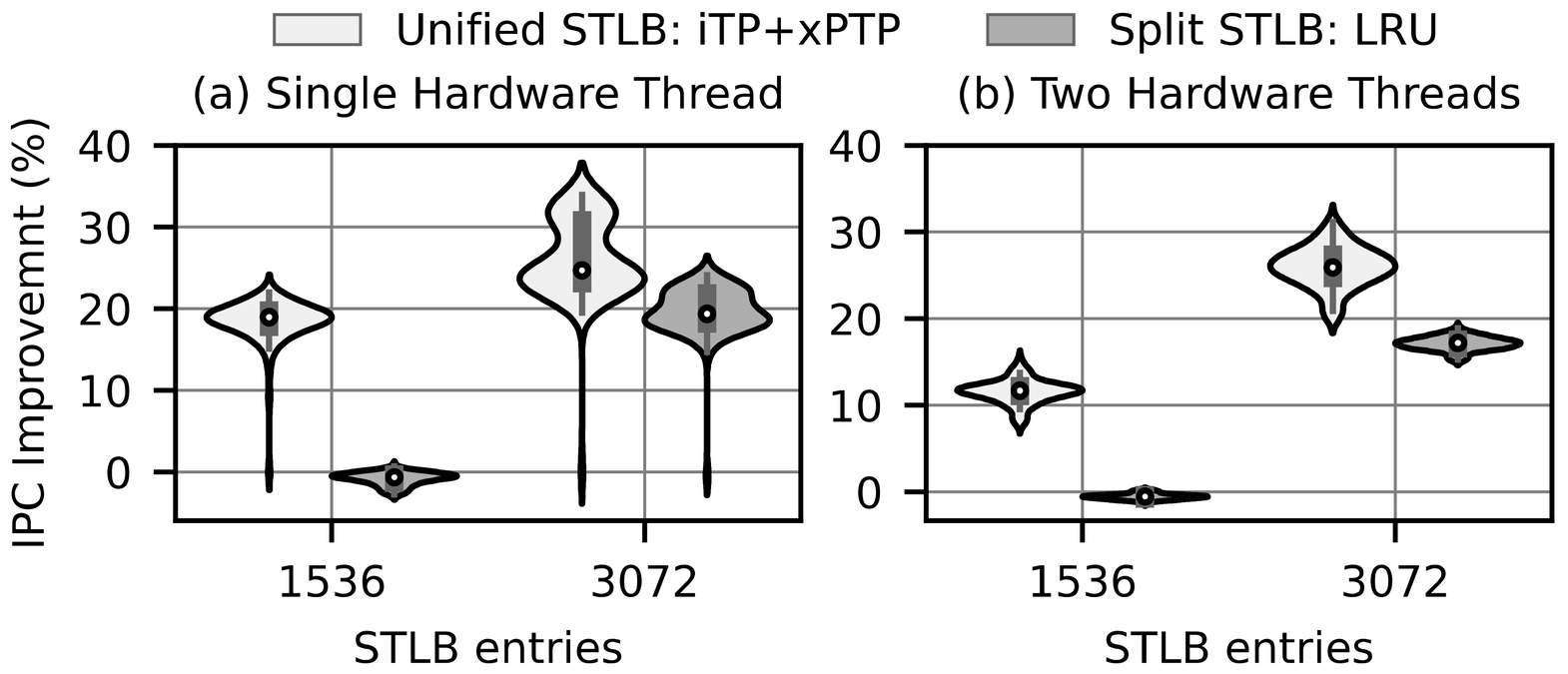



Two Hardware Threads




- Instruction TLB (ITLB) size can impact performance gains of iTP and iTP+xPTP
  - Larger ITLB sizes reduce the instruction translations misses in the second level TLB
  - iTP aims at reducing instruction translation misses
- Even for unrealistically large ITLBs (e.g. 512, 1024), there are still enough instructions translation misses for iTP and iTP+xPTP to deliver speedup

# Evaluation – Unified vs Split TLB




- Baseline is Unified TLB with 1536 entries and LRU as replacement policy
- Unified TLB with LRU is slightly better than Split with LRU
- Our proposal outperforms Split TLB even more!

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Content

- Motivation
  - Instruction Address Translation
  - Prioritizing Instruction Address Translations
- Translation-Aware Replacement policy Design
  - Instruction Translation Prioritization (iTP)
  - Extended Page Table Prioritization (xPTP)
  - Combining iTP and xPTP
- Evaluation
  - Comparison with Instruction-aware Policies
  - Impact of LLC Policies
  - Page Size Impact
- Conclusions