Cost-Effective Instruction TLB Prefetching

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

[1]Barcelona Supercomputing Center [2]Texas A&M University [3]Universitat Politècnica de Catalunya

Virtual Memory

Paging

- Virtual and physical address spaces are split into pages
- Each memory access requires a virtual-to-physical address translation.

Software Support

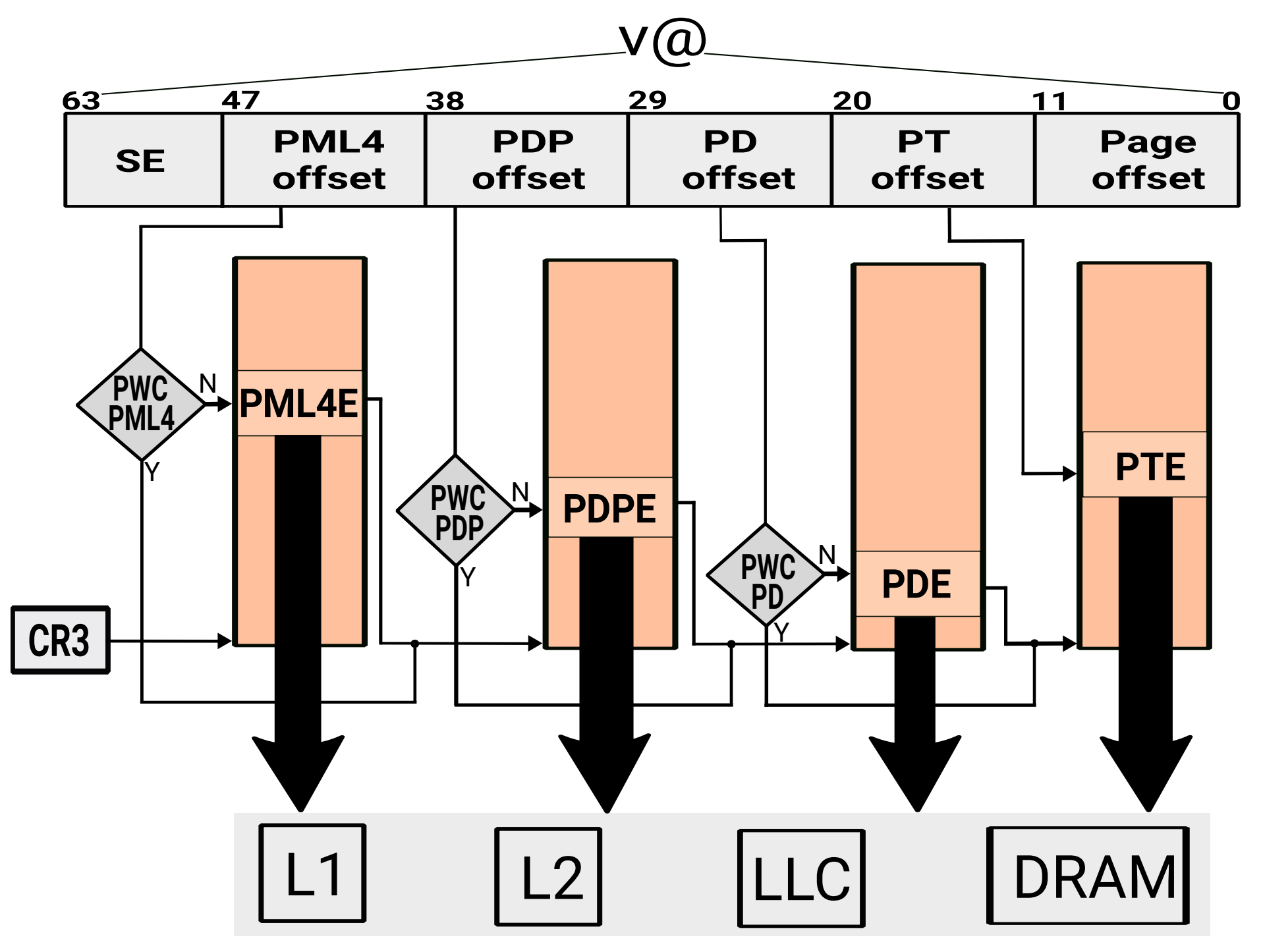
- Page Table stores the virtual-to-physical mappings of all pages loaded to memory
- X86 architectures implement 4 (or 5) level radix tree Page Table (PML4, PDP, PD, PT)

Hardware Support

- The MMU consists of (i) Translation Lookaside Buffer (TLB) and (ii) Page Walk Cache (PWC)
- TLBs cache the most recently used address translations
- Modern TLB organizations are multilevel (L1-ITLB, L1-DTLB, L2-TLB)
- PWCs store intermediate levels of the Page Table (PWC_{PML4}, PWC_{PDP}, PWC_{PD})

Address Translation Bottleneck

- Page walk hardware is activated on TLB misses
- Page walks incur great latency and energy costs
- Frequent TLB misses burden performance



Data TLB (D-TLB) Misses

- Previous work focuses on D-TLB misses due to the growing importance of Big Data workloads

Instruction TLB (I-TLB) Misses

- I-TLB performance used to be high
- Increasing code footprint
- Poor code locality

Server, database and cloud applications tend to have massive instruction footprints

Interaction between Instruction and Data TLB Entries

The increasing frequency of instruction TLB misses deteriorates TLB performance since instruction PTEs thrash from the TLB useful data PTEs (and vice versa)

Related Work on Instruction Address Translation

SW Approaches [4]

- Compile-time techniques for code layout optimization
- OS-schemes that leverage 2MB pages

SW/HW Approaches

Incremental Approaches [2]

- Rely on explicit OS and hardware support to increase TLB reach
- Susceptible to performance issues when coalescing opportunities are not guaranteed (HW support, memory fragmentation)

Disruptive Approaches [3]

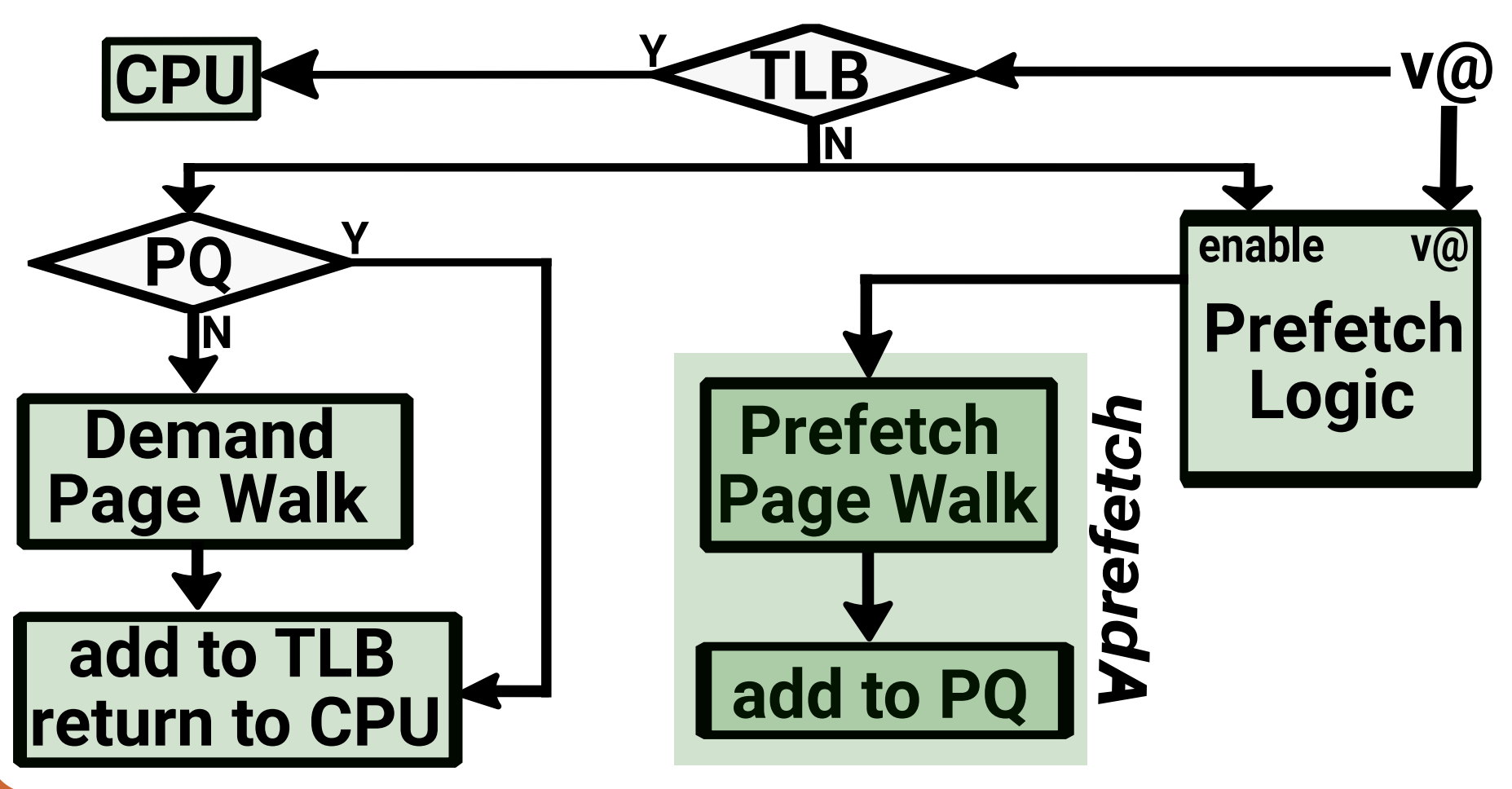
- Re-engineering of whole virtual memory subsystem
- New security vulnerabilities might be introduced

HW Approaches

- To the best of our knowledge, there is no proposed hardware scheme aimed at attenuating the instruction TLB performance bottleneck

Our Objective – TLB Prefetching

- Prefetching PTEs ahead of demand accesses
- Independent of the system state (OS, load, fragmentation)
- Does not disrupt virtual memory subsystem
- A Prefetch Queue (PQ) is used to store the prefetched PTEs



Methodology

Experimental Setup

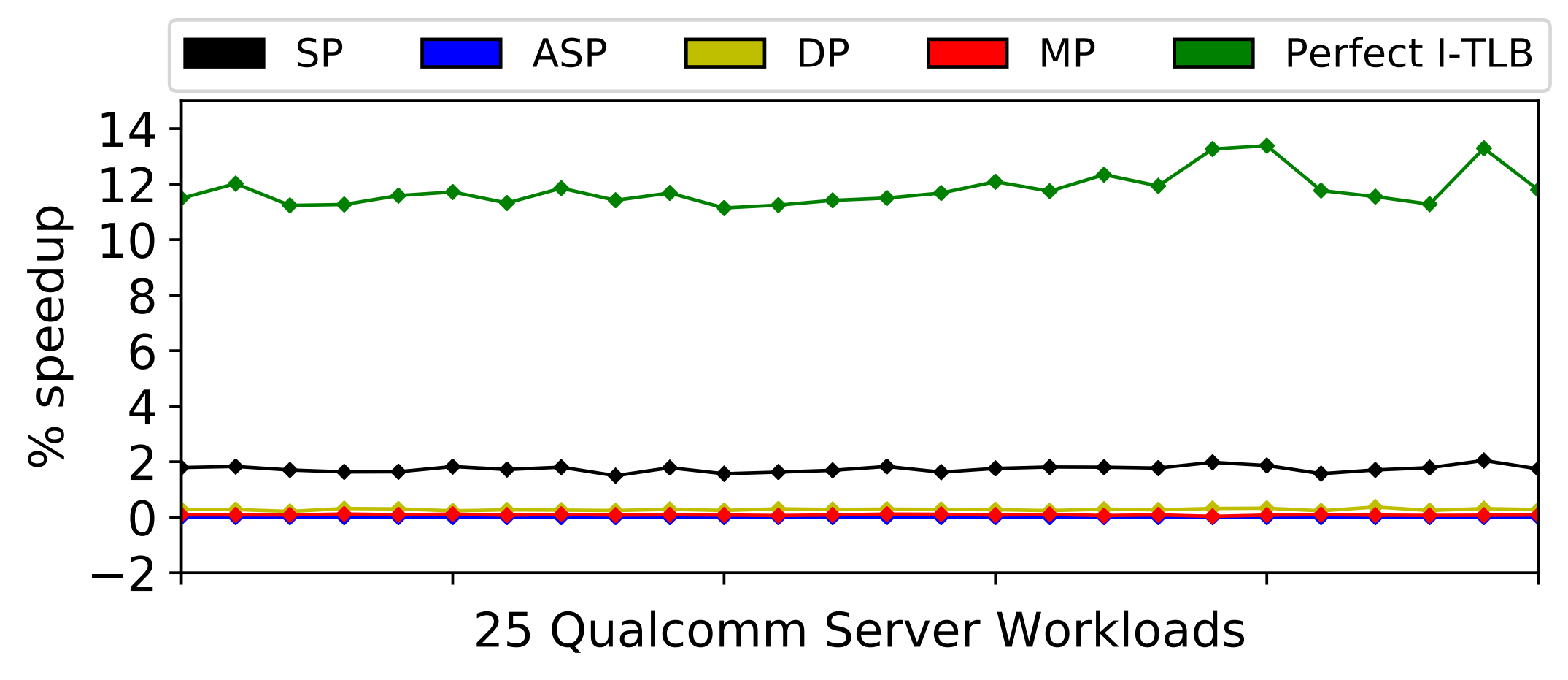
- ChampSim multicore simulator
- Realistic page table walk

Workloads

- Twenty five (25) server applications provided by Qualcomm

- We simulate a 3-level split PWCs
- The evaluation considers a 64-entry PQ

Efficacy of Data TLB Prefetchers on I-TLB Miss Stream



Data TLB Prefetchers [1]

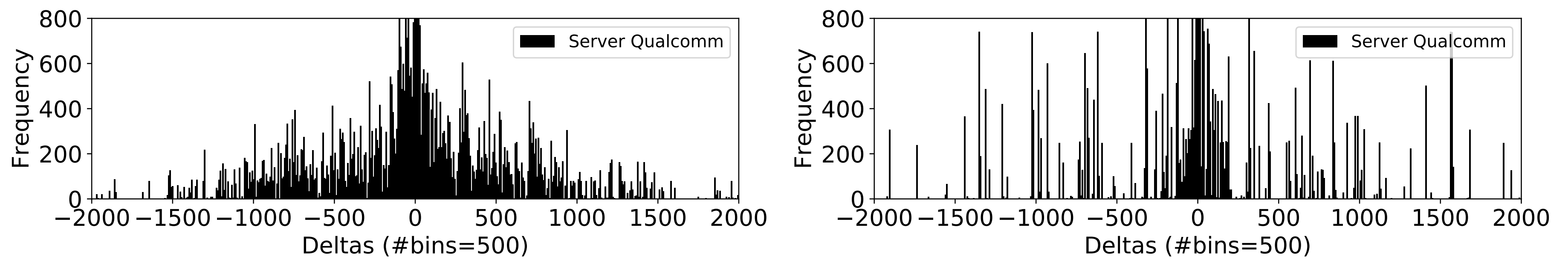
- Sequential Prefetcher (SP)
- Arbitrary Stride Prefetcher (ASP)
- Distance Prefetcher (DP)
- Markov Prefetcher (MP)

ASP, DP and MP use a 64-entry 4-way prediction table

Long Story Short

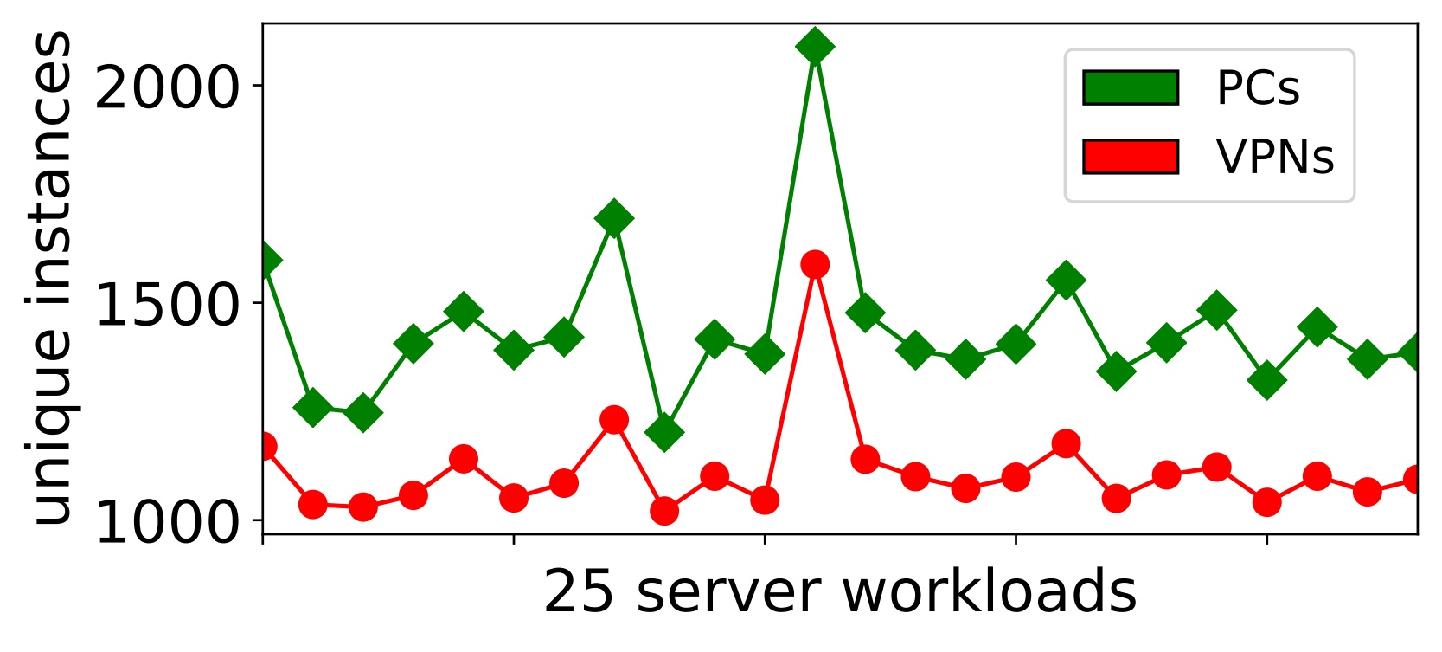
- Data TLB prefetchers are able to save a small fraction of the reported I-TLB misses
- Perfect I-TLB performance reveals significant room for improvement

Analysing the I-TLB Miss Stream



Analysis Findings

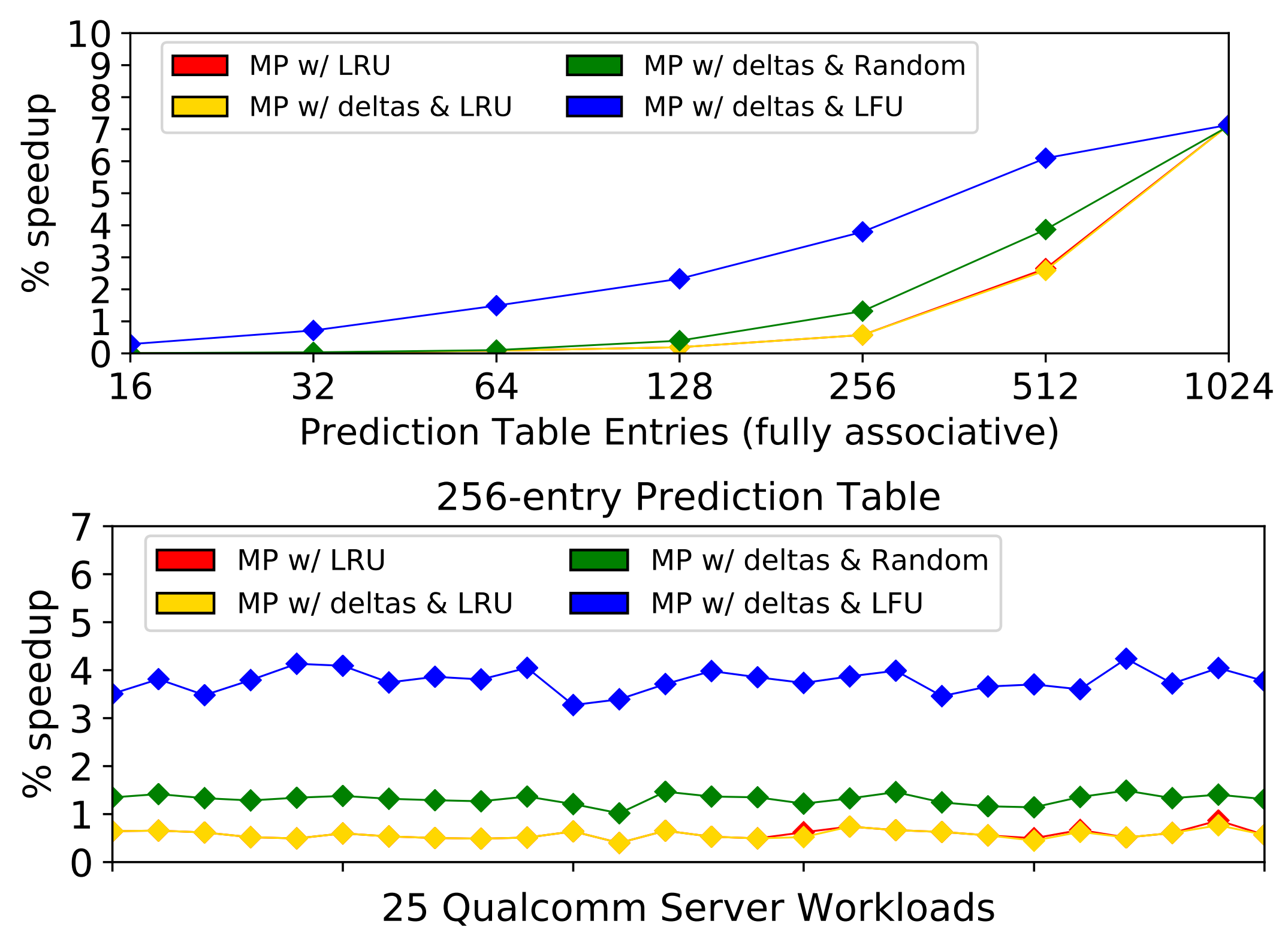
- Conventional delta-based prefetching is not promising for the I-TLB miss stream
- Small deltas (e.g. -7,..,+7) are constantly useful
- The evaluated server workloads exhibit different memory access patterns



Event Correlation

A table-based I-TLB prefetcher would require fewer prediction table entries to correlate the I-TLB misses with the virtual page number event compared to the pc event

Towards a Markov-based I-TLB Prefetcher



Evaluated Scenarios

- Markov Prefetcher (MP) w/ LRU [1]
- MP w/ deltas & LRU policy
- MP w/ deltas & Random policy
- MP w/ deltas & LFU policy

LFU policy

- On a prediction table miss randomly replace one of the least frequently accessed entries
- Hardware complexity similar to LRU policy

Conclusions

- MP with deltas and state-of-the-art MP achieve similar performance with LRU policy
- MP with deltas requires 47% less hardware complexity compared to state-of-the-art MP
- MP with deltas combined with LFU policy significantly improves performance over all evaluated scenarios and workloads

- Our preliminary results reveal that a Markov-based prefetcher has the potential to attenuate the I-TLB performance bottleneck
- Metrics like prediction accuracy, number of memory references introduced due to prefetch page walks and energy consumption has to be taken into account
- The hardware complexity of a novel I-TLB prefetcher has to be minimal (<4KB)

References

[1] G. B. Kandiraju and A. Sivasubramaniam, "Going the Distance for TLB Prefetching: An Application-driven Study", ISCA'02
[2] B. Pham, V. Vaidyanathan, A. Jaleel, and A. Bhattacharjee, "CoLT:Coalesced Large-Reach TLBs", HPCA'12
[3] H. Alam, T. Zhang, M. Erez, and Y. Etsion, "Do-it-yourself virtual memory translation", ISCA'17
[4] Y. Zhou, X. Dong, A. L. Cox, and S. Dwarkadas, "On the impact of instruction address translation overhead", ISPASS'19