

Effective Instruction Prefetching in Chip Multiprocessors for Modern Commercial Applications

Lawrence Spracklen, Yuan Chou & Santosh G. Abraham

Lawrence Spracklen

**Advanced Processor Architecture
Sun Microsystems**

International Symposium
on High-Performance
Computer Architecture

Feb 15th 2005



Outline

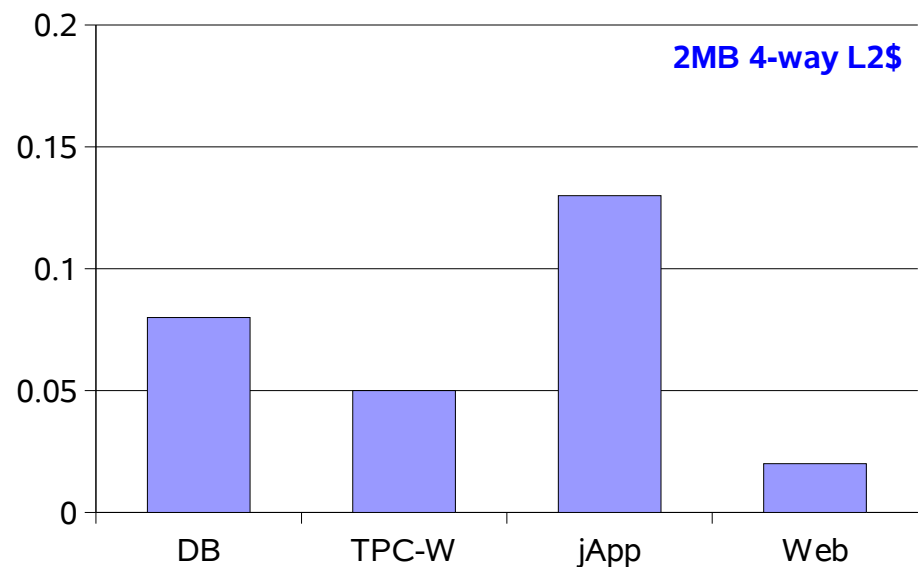
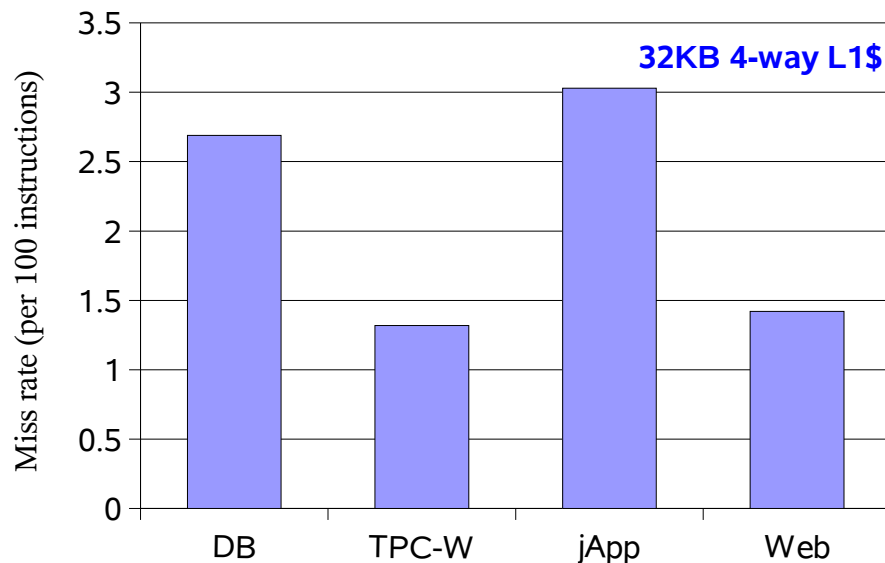
- Motivation
- Limit study
- The Discontinuity prefetcher
- Results
- Conclusions

Motivation

- Cache missing memory accesses frequently dictate application performance
- Typically think in terms of data misses
 - Dominate for SPEC CPU2000 benchmarks
- Commercial applications have large instruction working sets
 - Exemplified by databases, application servers, and web servers
- We investigate the performance implications for:
 - A database workload
 - TPC-W
 - SPECjAppServer2002
 - SPECweb99

Instruction Miss Rates

- Commercial applications observe significant stalls due to both L1 cache and L2 cache instruction misses
- Instruction misses can be more problematic than either load or store misses

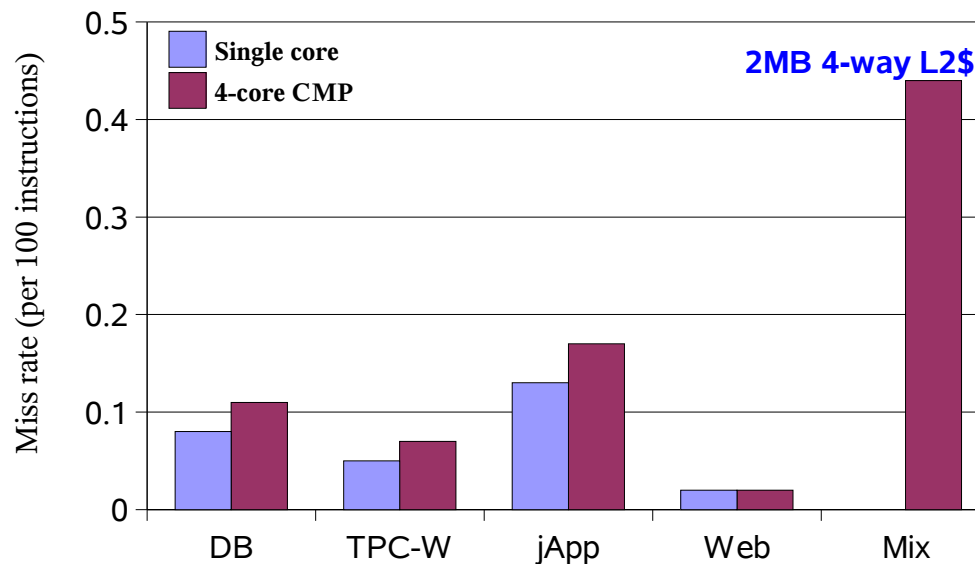


Chip Multiprocessors

- Recent years witnessed a significant paradigm shift and the emergence of Chip Multiprocessors (CMPs)
- CMPs are exemplified by multiple cores on a single chip
- Advanced CMP cores typically:
 - Have private Level-1 caches (L1\$)
 - Share a modest Level-2 cache (L2\$)
- We investigate how CMPs impact commercial workloads
 - Use a next-generation 4-core CMP, with a 2MB shared L2\$

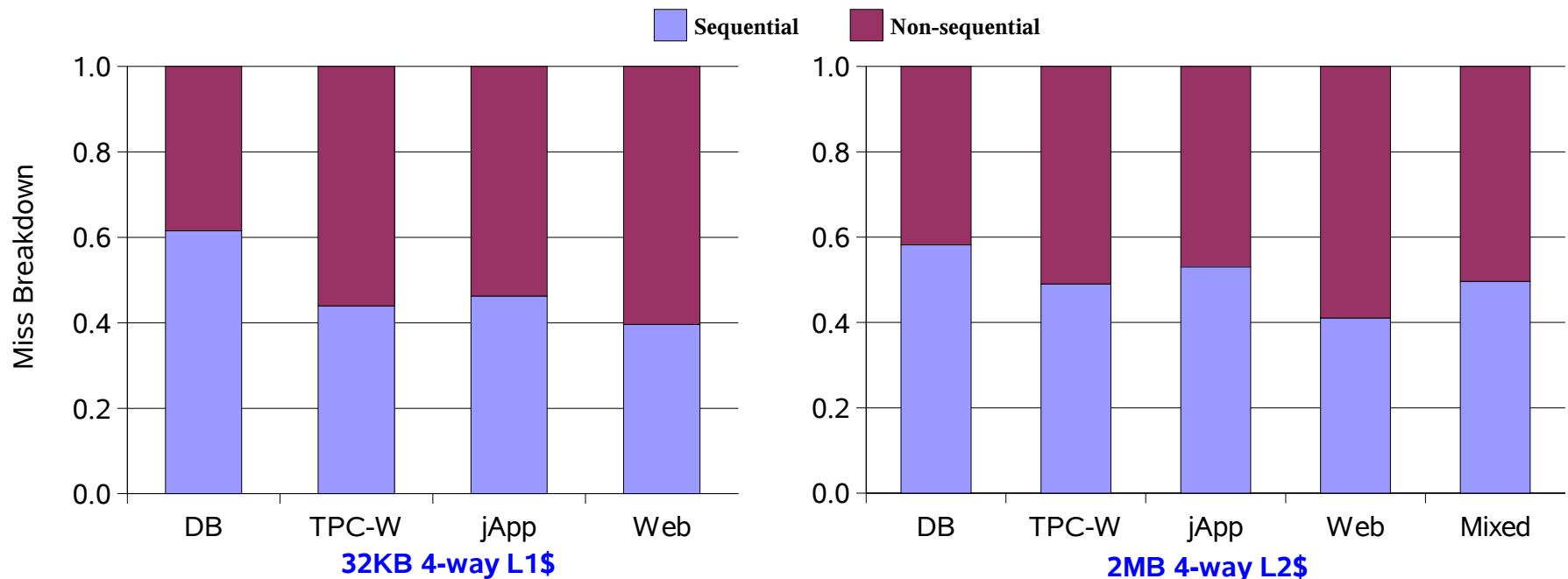
CMP Instruction Miss Rates

- L1\$ miss rates are identical as cores have private I\$
- L2\$ is shared between 4 cores
 - Less cache resources per core
- Applications experience more frequent cache misses
 - HW prefetchers are even more important!



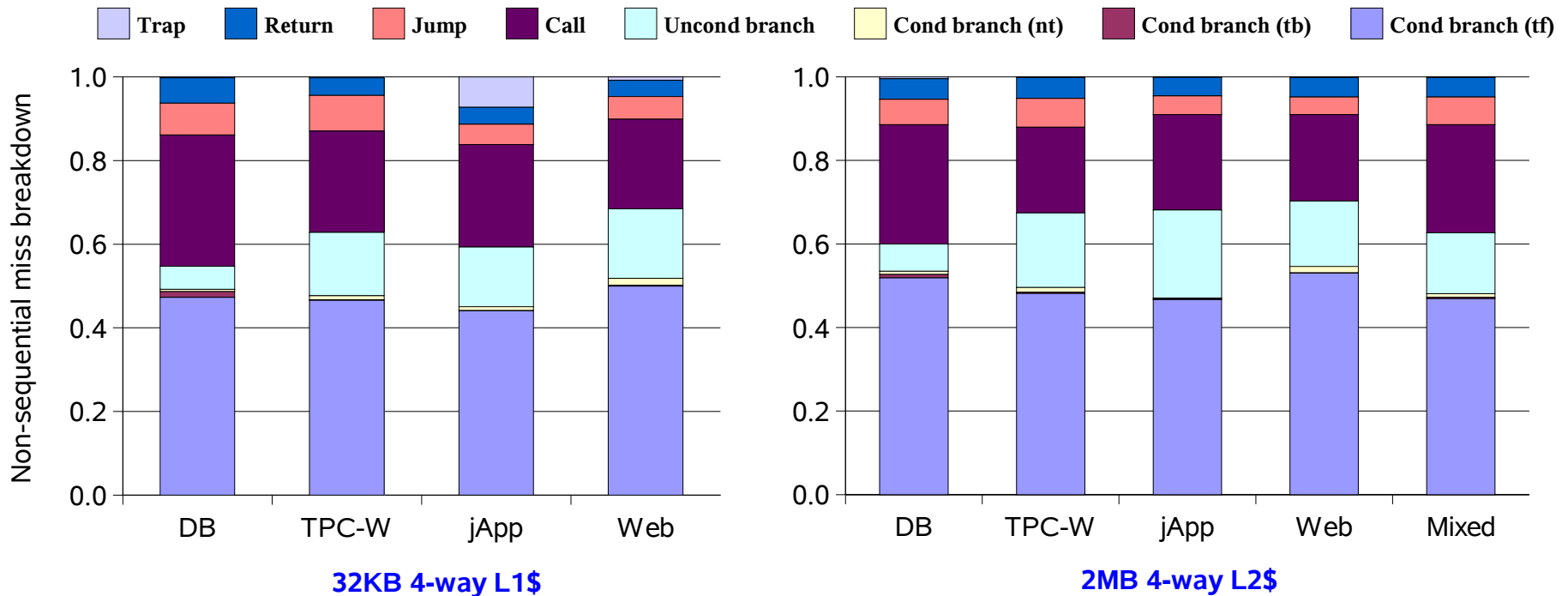
Miss Classification

- Few processors provide support for instruction prefetching
- Typically focus on sequential prefetchers
 - Given an access/miss for line L, prefetch line L+1
- Irregular control flow in commercial apps means many misses are non-sequential



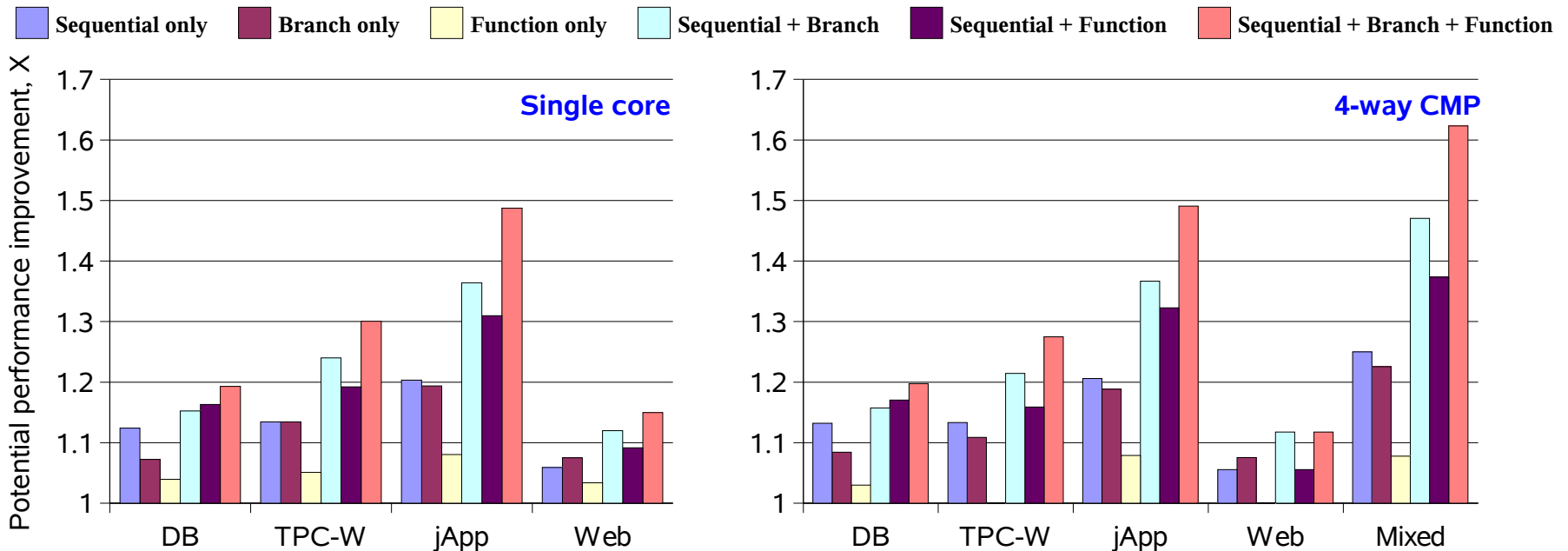
Non-Sequential Misses

- Sequential prefetchers fail to capture up to almost 60% of instruction misses in commercial applications
- Non-sequential misses are not attributable to a single cause
 - Caused by a variety of control transfer instructions (CTI)



Potential Performance Improvements

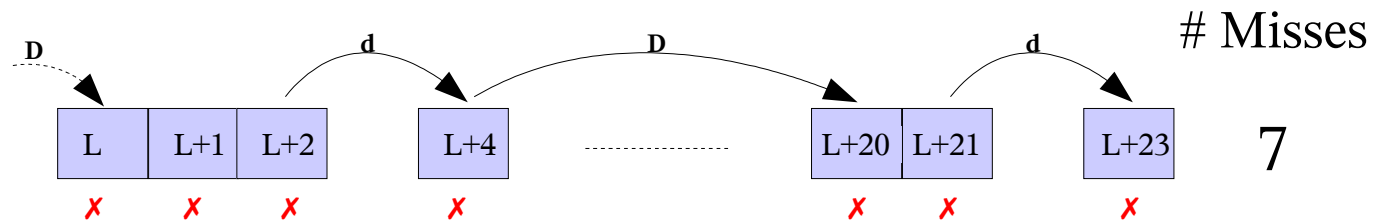
- Failure to address non-sequential instruction misses sacrifices significant performance
 - Can double performance gain by also targeting non-sequential misses
- Non-sequential prefetchers must target all main CTI groups
 - Many prior instruction prefetchers only captured a subset of misses



Introducing Discontinuities

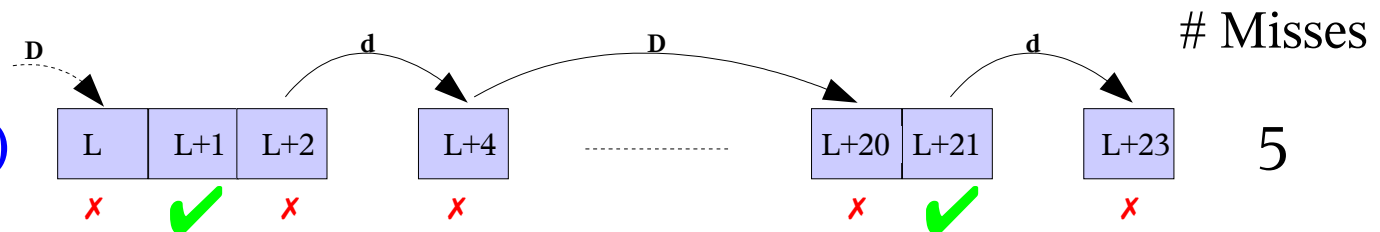
- When a CTI instruction causes a transition to a non-sequential cache line it causes a 'discontinuity' in the instruction fetch stream

No Prefetch

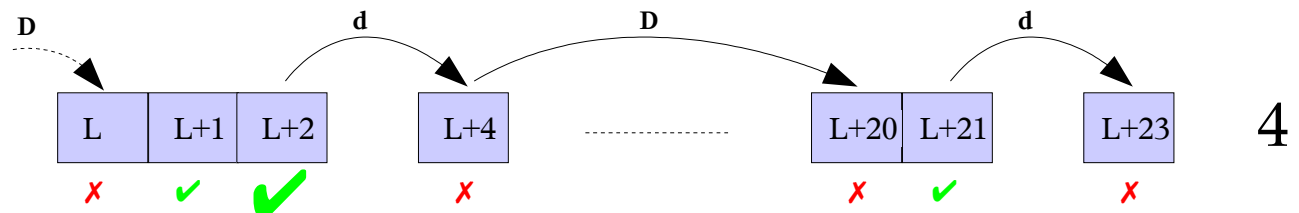


- Next-line prefetchers can't capture these transitions

Next-line (on miss)

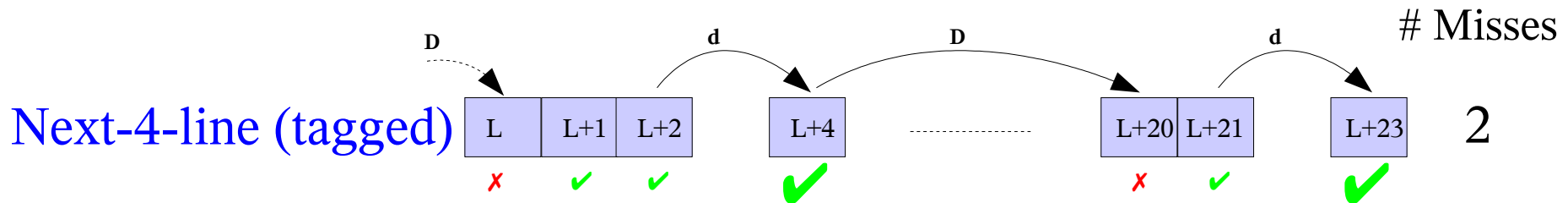


Next-line (tagged)



Capturing Discontinuities

- Next-N-line sequential prefetchers capture short forward discontinuities
 - Where the target lies within the prefetch-ahead distance



- Next-N-line sequential prefetchers represent a simple, low-cost mechanism to prefetch for small discontinuities
- An elegant, CTI independent method for capturing the remaining discontinuities is required

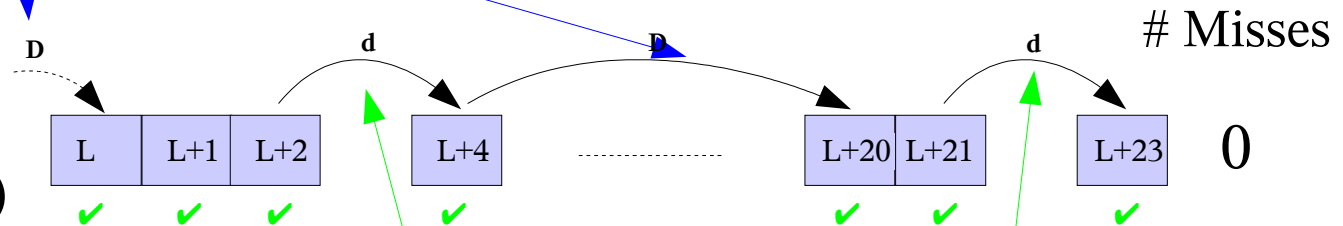
We propose the [Discontinuity prefetcher](#)

The Discontinuity Prefetcher

- The discontinuity prefetcher utilizes a history-based predictor to track discontinuities that incur an L1\$ miss
- Only need to track large discontinuities that aren't covered by the next-N-line sequential prefetcher
 - Significantly reduces the size of the required predictor

Predictor only needs to cover large discontinuity
=> **Small Predictor**

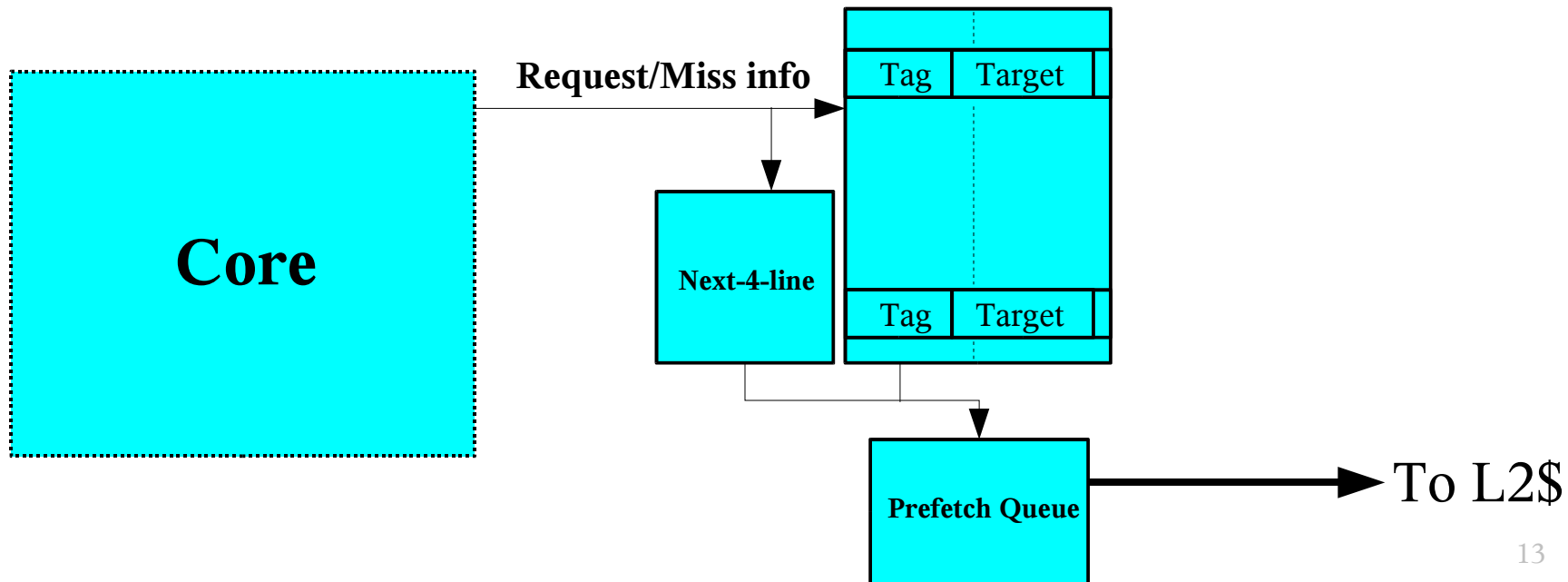
Discontinuity +
Next-4-line (tagged)



Small forward discontinuities covered by next-4-line sequential prefetcher

Prefetcher Implementation

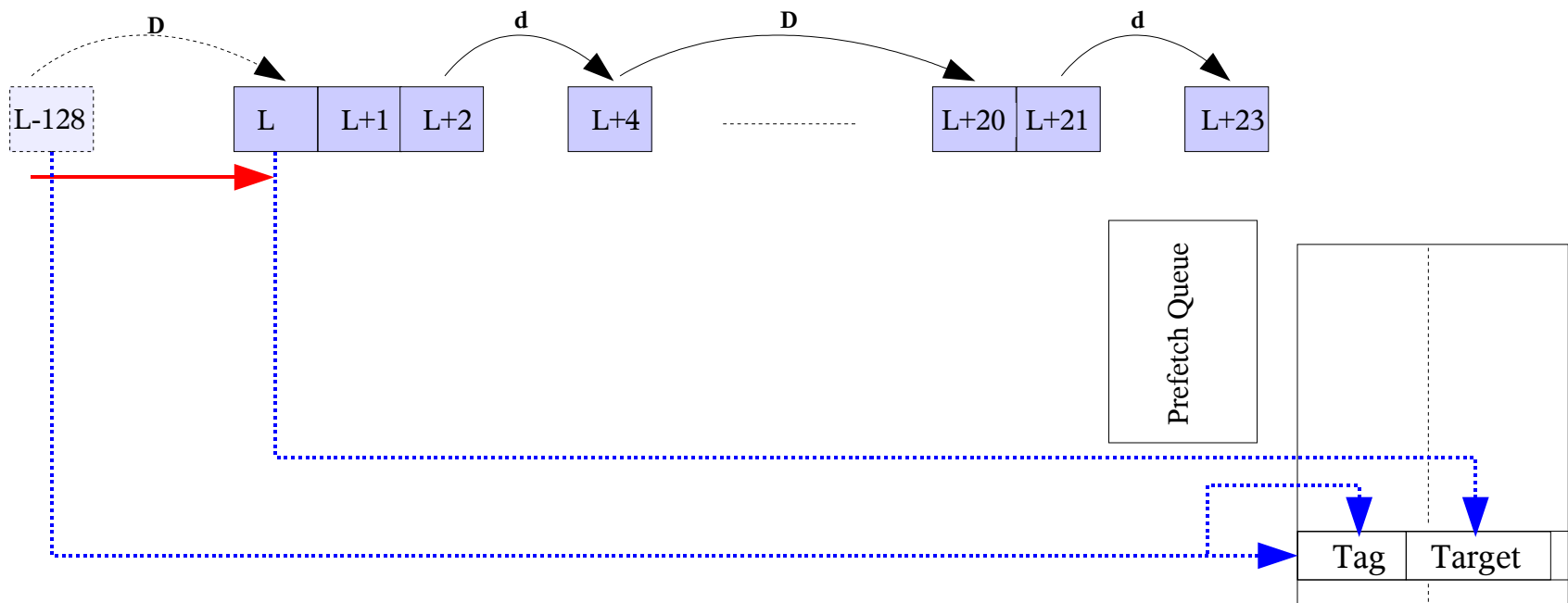
- Predictor is implemented as a direct-mapped table
- Indexed by a portion of the address of the trigger
- Entry is tagged with a portion of the address of the trigger
- Only require one target per entry



Prefetcher Operation

Allocation:

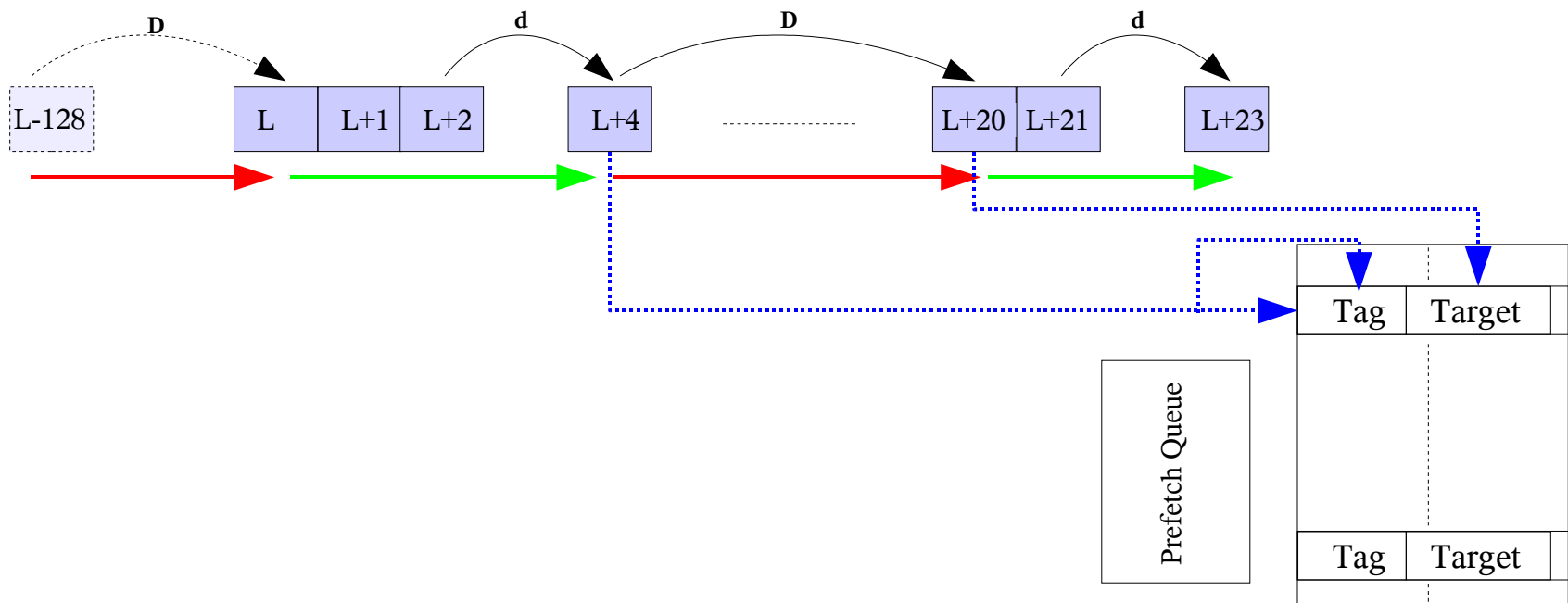
- When a discontinuity causes a miss, it is inserted into the table



Prefetcher Operation

Allocation:

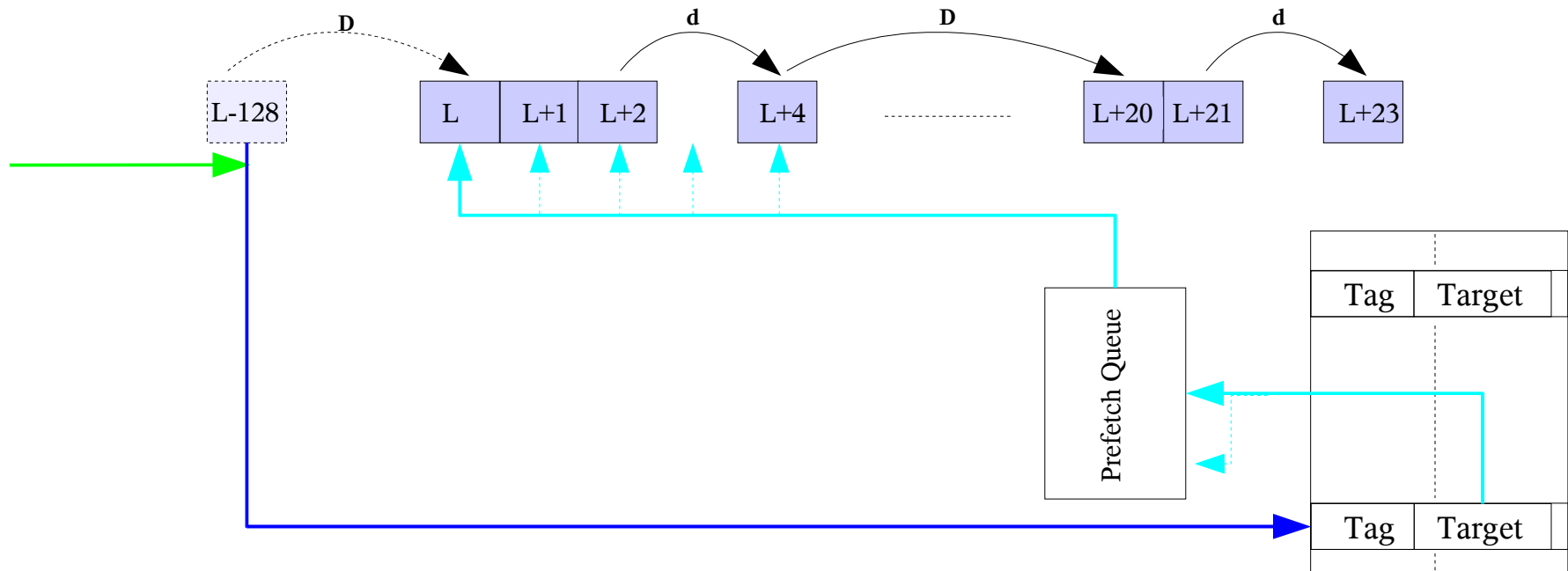
- When a discontinuity causes a miss, it is inserted into the table



Prefetcher Operation

Prediction:

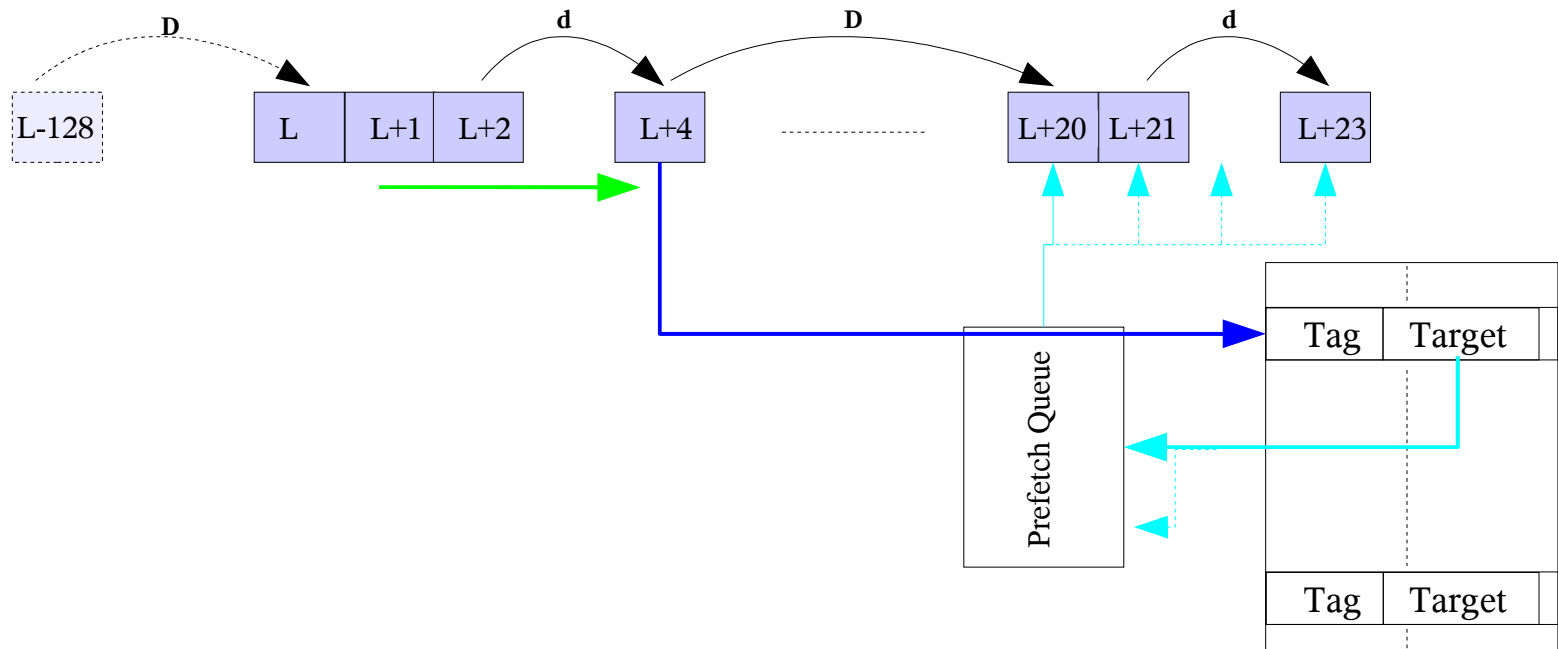
- Predictor is probed by the sequential prefetcher moving ahead of the demand fetch stream
- If a valid entry is located a prefetch is issued for the potential target
- Prefetches are also issued for sequential lines following the target (up to N)



Prefetcher Operation

Prediction:

- Predictor is probed by the sequential prefetcher moving ahead of the demand fetch stream
- If a valid entry is located a prefetch is issued for the potential target
- Prefetches are also issued for sequential lines following the target (up to N)



Methodology

- Processor overview

Processor

- 4-core CMP
- 64-entry issue window
- 3-wide issue
- 64K gshare predictor

Memory Hierarchy

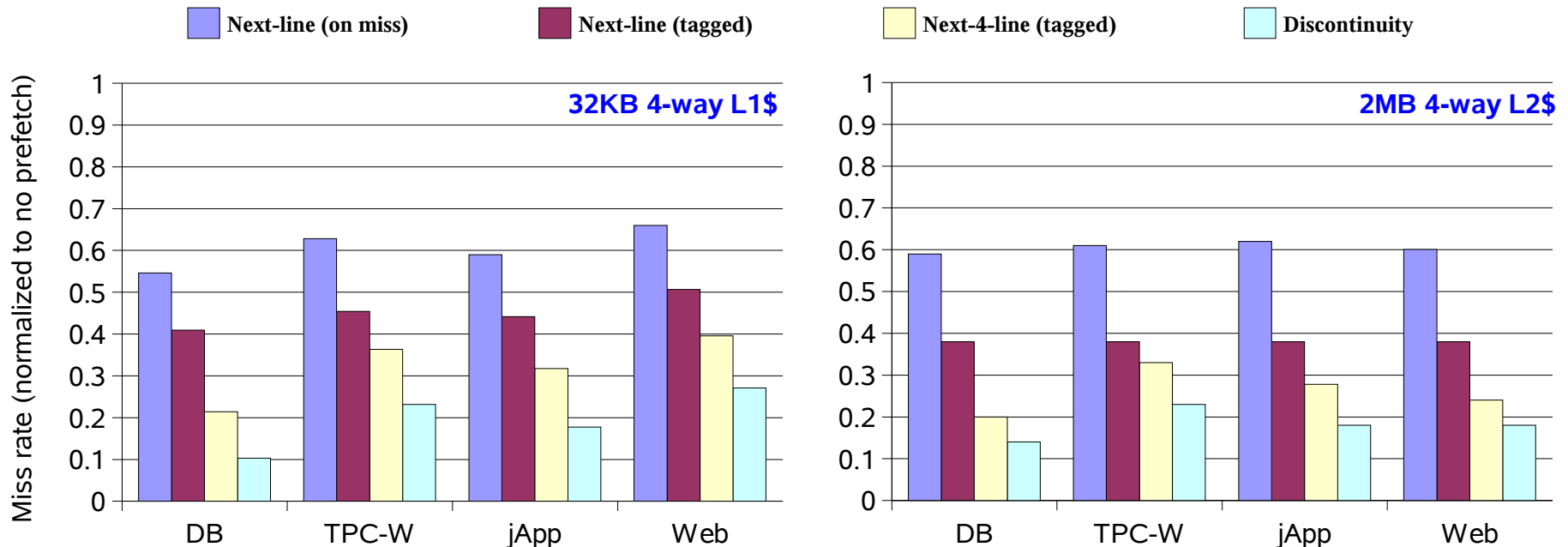
- 32KB 4-way 64B I\$ and D\$ (per core)
- 2MB 4-way 64B L2\$ (shared between cores)
- 400-cycle memory latency
- 20GB/s offchip BW

Discontinuity Prefetcher

- 8192-entry direct-mapped table (per core)
- Next-4-line sequential prefetcher
- Compared discontinuity prefetcher to:
 - **Next line (on miss)**: if line L is a miss, prefetch line L+1
 - **Next line (tagged)**: if line L is a miss or a previously prefetched line, prefetch line L+1
 - **Next-4-line (tagged)**: if line L is a miss or a previously prefetched line, prefetch lines L+1, L+2, L+3 and L+4

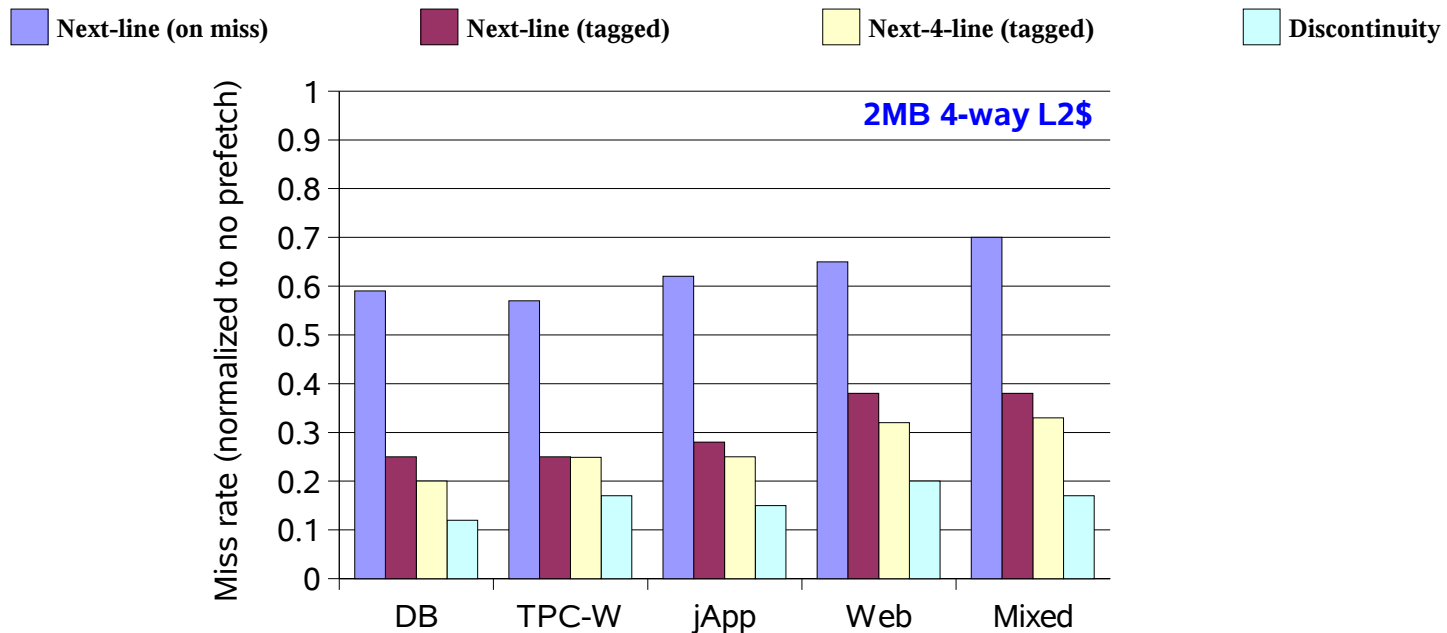
Miss Coverage – single core

- Achieve a significant reduction in both the I\$ miss rate and the L2\$ instruction miss rate
 - 90% of L1\$ misses and 85% of L2\$ misses eliminated for database workload
- The discontinuity prefetcher outperforms the sequential prefetchers



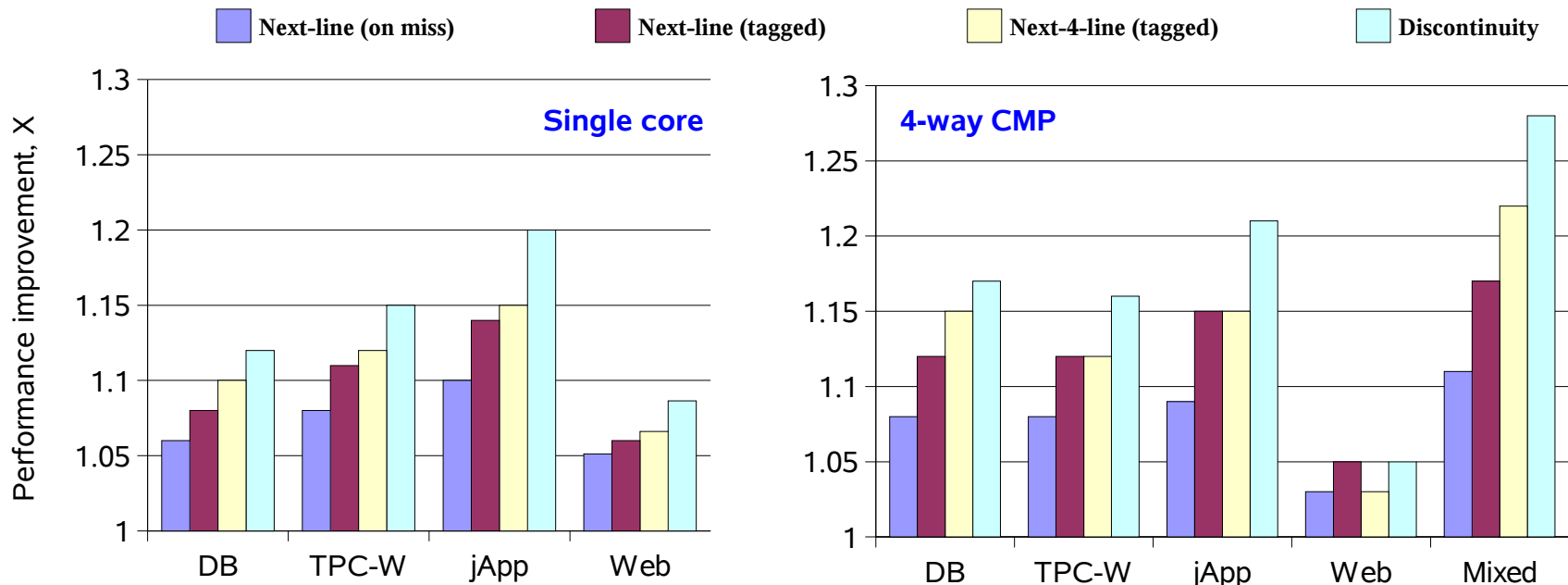
Miss Coverage – CMP

- CMP L1\$ miss reduction is identical to the reductions achieved for a single core
 - Cores have private L1\$s
- L2\$ miss rate reductions similar to single-core reductions
 - Also manage to eliminate 82% of L2\$ instruction misses for the mixed workload



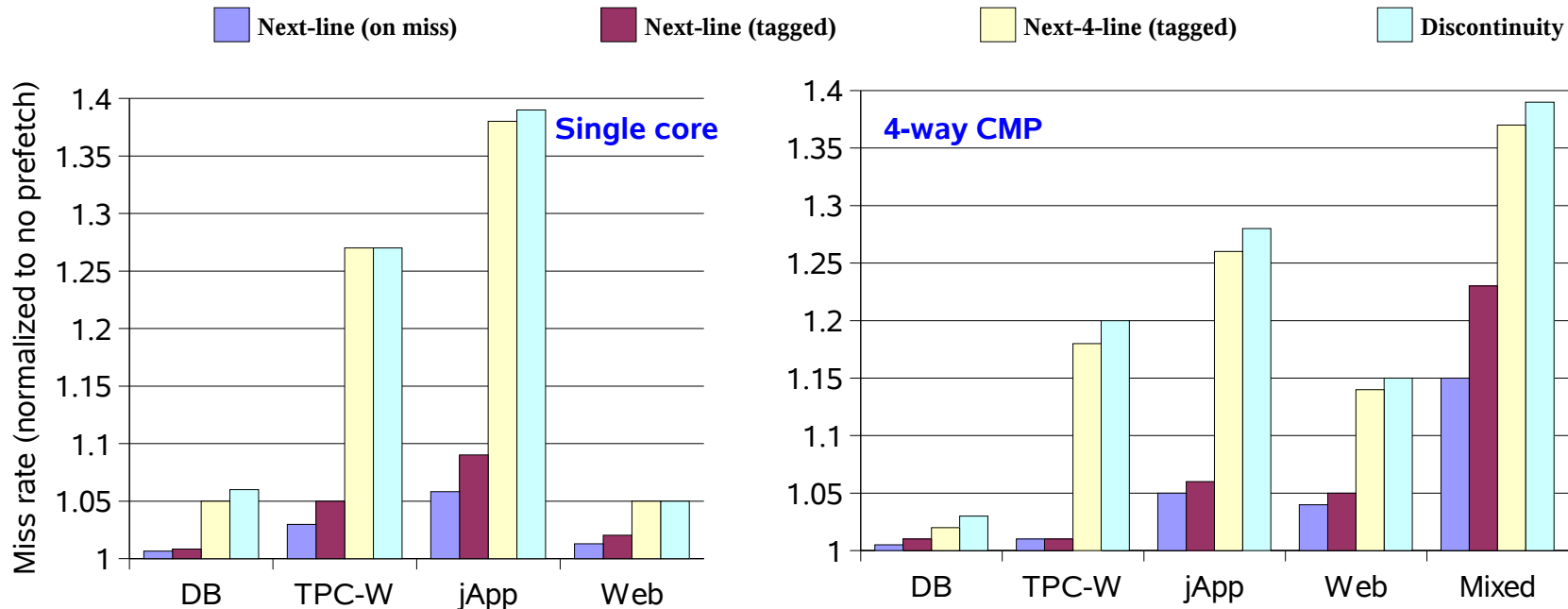
Performance Improvements

- The instruction prefetchers provide significant performance benefits
 - Higher performance benefits observed for the CMP
- Given the significant reduction in miss rates, greater performance improvements seemed likely



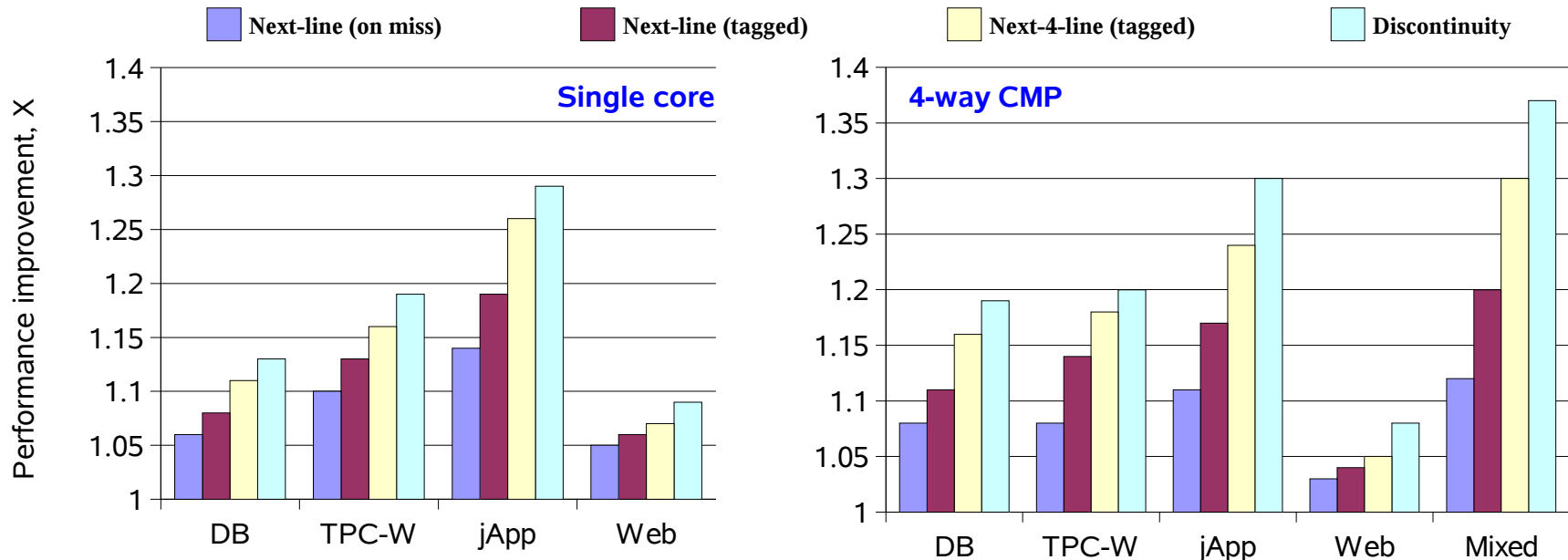
Data Miss Rates

- L2\$ data miss rates increased significantly when aggressive instruction prefetching was enabled
- Increase in data misses offsets the benefits from the reduction in instruction misses



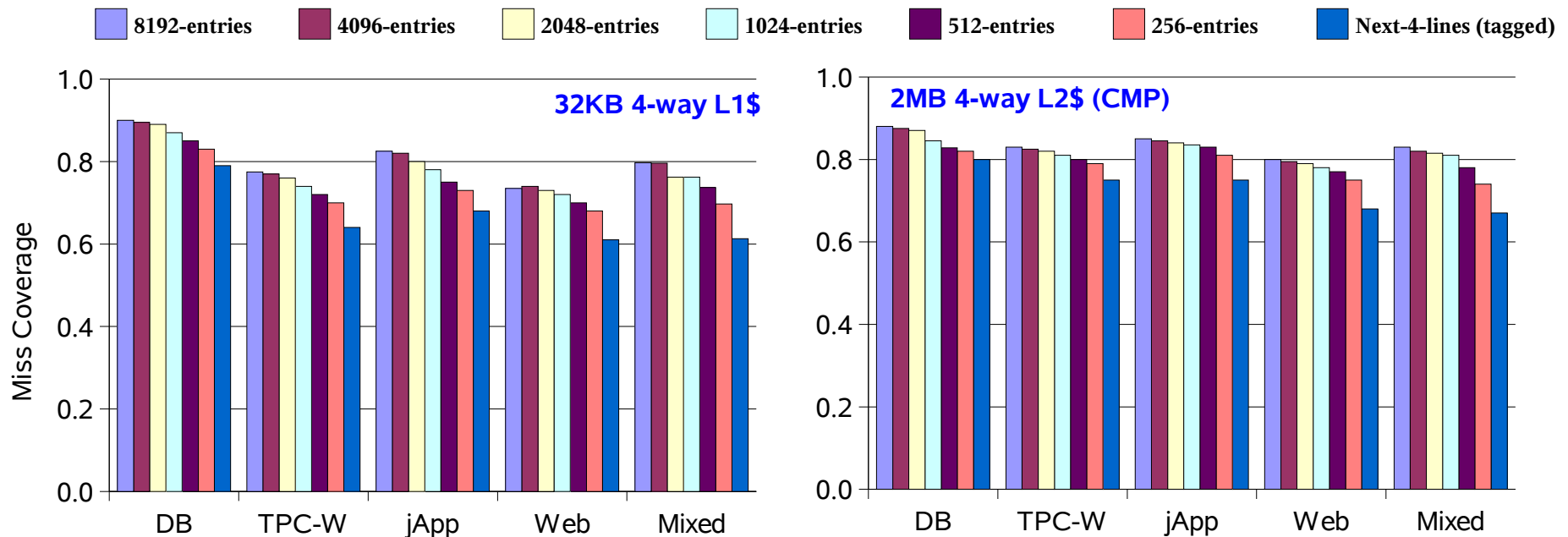
L2-Bypass Prefetching

- Introduced L2-Bypass prefetching
 - Prefetches are initially only installed in the L1\$
 - If the line is utilized during its residence in the L1\$, on eviction, the line is installed in the L2\$
- Eliminates L2\$ pollution by instruction prefetchers
 - Observe full performance benefits of the instruction prefetchers (up to 1.38X)



Low Cost?

- A small discontinuity prefetcher still provides appreciable performance increases



- Very little additional HW cost for the smaller predictors, yet they achieve significant performance gains over sequential prefetchers

Related Art

- Significant prior work on instruction prefetching in addition to the next-line and next-N-line sequential prefetchers:
 - Target prefetching [Hsu, Smith]
 - Markov prefetching [Joseph, Grunwald]
 - Branch-history guided prefetching [Tyson, Charney, Srinivasan, Davidson]
 - Call-graph prefetching [Davidson, Annavaram, Patel]
 - Fetch directed prefetching [Calder, Reinman, Austin]
 - Wrong-path prefetching [Pierce, Mudge]

- Benefits and drawbacks of these alternative schemes are discussed in more detail in the paper

Concluding Remarks

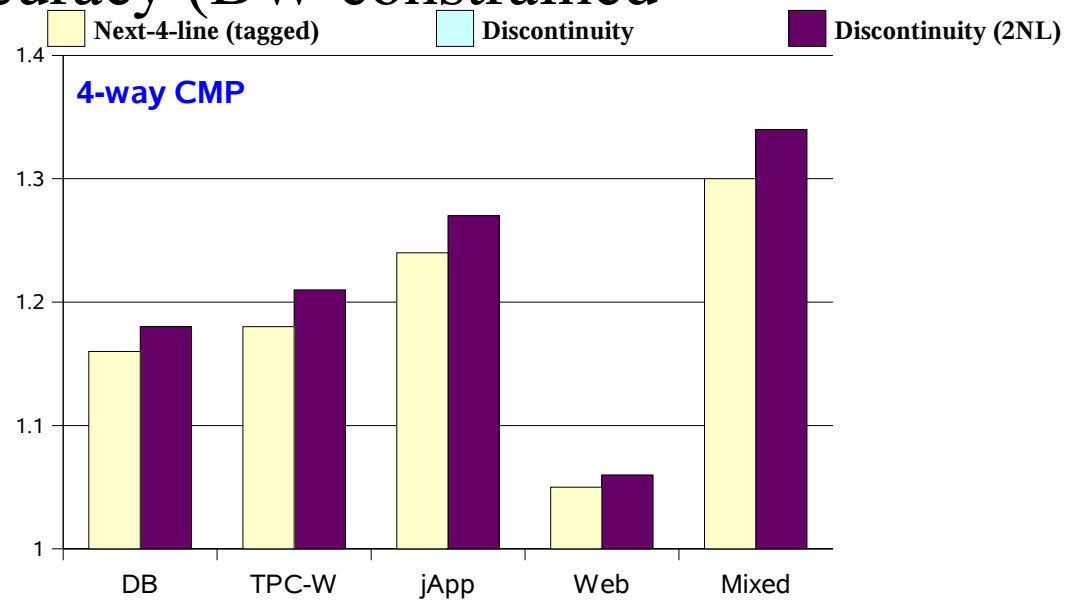
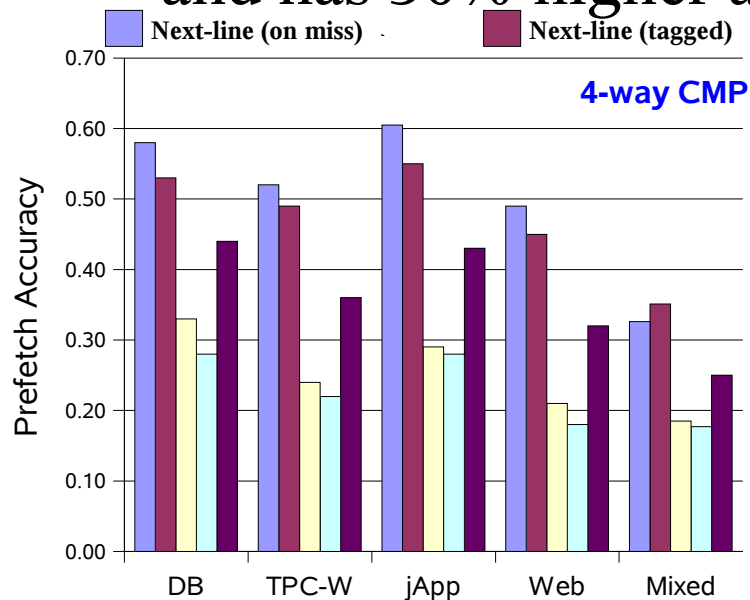
- Modern commercial applications have high instruction miss rates at both the L1 and L2 levels
- Effective instruction prefetching is imperative to mitigate the performance losses due to these misses
- Necessary to target all types of instruction misses
 - Sequential misses AND non-sequential misses caused by control transfer inst.
- Proposed the Discontinuity prefetcher which reduces the miss rate by ~90%
- Need to consider the pollution effects of aggressive prefetchers (especially in CMPs)
- Accelerated commercial apps by up to 38% using the Discontinuity prefetcher and selective L2\$ installation

Questions?

lawrence.spracklen@sun.com

Prefetch Accuracy

- Lower for the more aggressive instruction prefetchers
- Accuracy of the discontinuity prefetcher is comparable with the next-4-lines sequential prefetcher
 - Yet the Discontinuity prefetcher achieves superior performance
- 2-line discontinuity prefetcher outperforms next-4-lines and has 50% higher accuracy (BW constrained)



Prefetching for CMPs

- Implications for prefetching?
 - Resources per core decreased
 - Potential for inter-strand pollution increased
 - Chip real-estate available to support HW prefetchers decreased
- May require multiple L1\$ prefetchers per chip
- HW prefetchers need to be effective, accurate and low-cost