Low-Cost Epoch-Based Correlation Prefetching for Commercial Applications

Yuan Chou
Architecture Technology Group
Microelectronics Division

Motivation

- Performance of many commercial applications limited by processor stalls due to off-chip cache misses
- Applications characterized by irregular control-flow and complex data access patterns
- Software prefetching and simple stride-based hardware prefetching ineffective
- Hardware correlation prefetching more promising
  - can remember complex recurring data access patterns
- Current correlation prefetchers have severe drawbacks but we think we can overcome them

Talk Outline

- Traditional Correlation Prefetching
- Epoch-Based Correlation Prefetching
- Experimental Results
- Summary

Traditional Correlation Prefetching

- Basic idea: use current miss address M to predict *N* future miss addresses $F_1 \ldots F_N$ *(where N = prefetch depth)*

Miss address sequence:

assume N=2

Use A to prefetch B C

Use D to prefetch E F

Use G to prefetch H I

- Correlations recorded in correlation table
- Correlation table size proportional to application working set

Correlation Prefetching Drawbacks

- Very large correlation tables needed for commercial apps
  - impractical to store on-chip
- No attempt to eliminate all naturally overlapped misses

Miss address sequence:

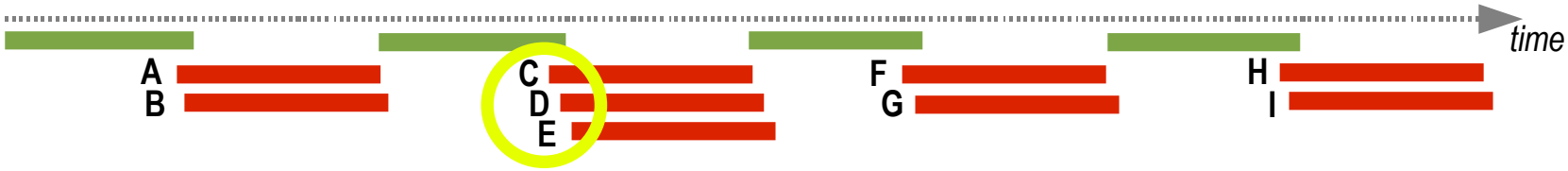
A B C D E F G H I

Correlation Prefetching Drawbacks

- Very large correlation tables needed for commercial apps
  - impractical to store on-chip
- No attempt to eliminate all naturally overlapped misses

Miss address sequence:

A B C D E F G H I



Since C, D and E naturally overlapped, prefetching only C may not improve performance

Correlation Prefetching Drawbacks

- Very large correlation tables needed for commercial apps
  - impractical to store on-chip
- No attempt to eliminate all naturally overlapped misses
- Prefetches misses naturally overlapped with current miss

Miss address sequence:

Since A and B naturally overlapped, prefetching B does not improve performance but wastes table storage

Epoch-Based Correlation Prefetching (EBCP)

Epoch MLP Model

- At high off-chip latencies, overlappable off-chip accesses appear to issue and complete together
- Program execution separates into recurring periods of on-chip computation followed by off-chip accesses

Epoch MLP Model

- At high off-chip latencies, overlappable off-chip accesses appear to issue and complete together
- Program execution separates into recurring periods of on-chip computation followed by off-chip accesses

- Call each period an epoch

Epoch MLP Model

- At high off-chip latencies, overlappable off-chip accesses appear to issue and complete together
- Program execution separates into recurring periods of on-chip computation followed by off-chip accesses

- Call each period an epoch
- Group off-chip accesses based on which epoch they issue

Epoch	i	i+1	i+2	i+3
Miss addresses	A B	C D E	F G	H I

Epoch Model Insights

Insight #1:

Target removal of entire epochs instead of individual misses

Miss address sequence:

A B C D E F G H I

Epoch	i	i+1	i+2	i+3
Miss addresses	A B	C D E	F G	H I

- Use first miss in epoch to prefetch all misses in next 2 epochs
- Results in removal of 2 epochs

Epoch-Based Correlation Prefetcher

- No prefetching

Epoch	i	i+1	i+2	i+3
Miss addresses	A B	C D E	F G	H I

- Epoch-based correlation prefetching (EBCP)

Epoch	i	i+1
Miss addresses	A B	H I
Prefetches	C D E F G	

- Traditional correlation prefetching (depth=2)

Epoch	i	i+1	i+2
Miss addresses	A B	E	H I
Prefetches	B C D	F G	

EBCP achieves better epoch reduction

Epoch Model Insights

Insight #2:

Hide latency of correlation table access under previous epoch

Epoch	i	i+1	i+2	i+3
Miss addresses	A B	C D E	F G	H I
Prefetches	Read correlation table	F G H I		

- Use miss in epoch i to prefetch all misses in epochs i+2 and i+3
- Use epoch i to read correlation table
- Use epoch i+1 to issue prefetches

Epoch Model Insights

Insight #2:

Hide latency of correlation table access under previous epoch

Epoch	i	i+1	~~i+2~~	~~i+3~~
Miss addresses	A B	C D E	~~F G~~	~~H I~~
Prefetches	Read correlation table	F G H I		

- Results in removal of 2 epochs

Correlation table can be stored in main memory!

EBCP Advantages

Trad: store correlation table on-chip

EBCP: store correlation table in main memory (hide table access latency under previous epoch)

Trad: no attempt to eliminate all naturally overlapped misses

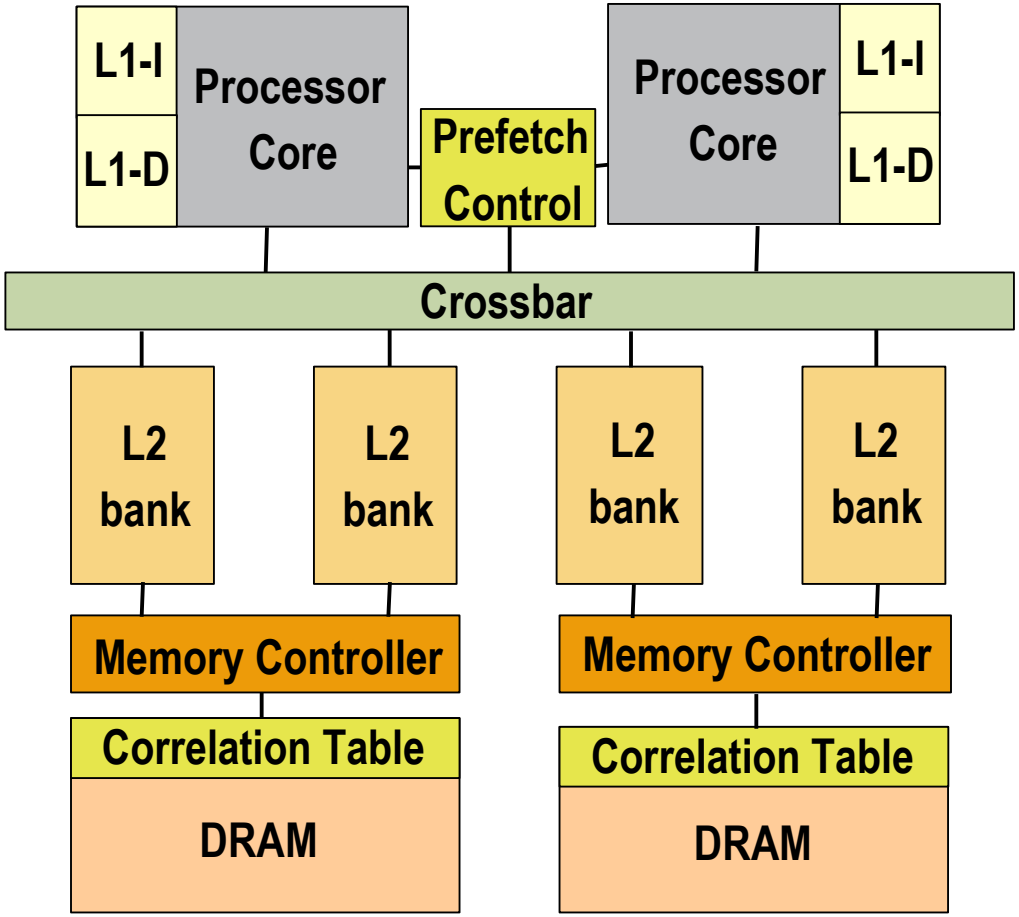
EBCP: target removal of entire epochs

Trad: prefetch misses naturally overlapped with current miss

EBCP: avoid prefetching these misses

EBCP overcomes drawbacks of traditional correlation prefetchers

EBCP Components



- Prefetcher control observes all L2 cache requests
- L2 banks notify prefetcher control which requests are misses

EBCP Prefetcher Control

- Request OS for memory to store correlation table
- Detect epochs
  - observe when number of off-chip misses transition 0 to 1
- Learn correlations
  - record correlations in main memory correlation table
- Issue prefetches
  - use first miss address in epoch to look up correlation table
  - select miss addresses from correlation table entry
  - issue prefetches (lower priority than demand accesses)
- Return memory to OS if needed

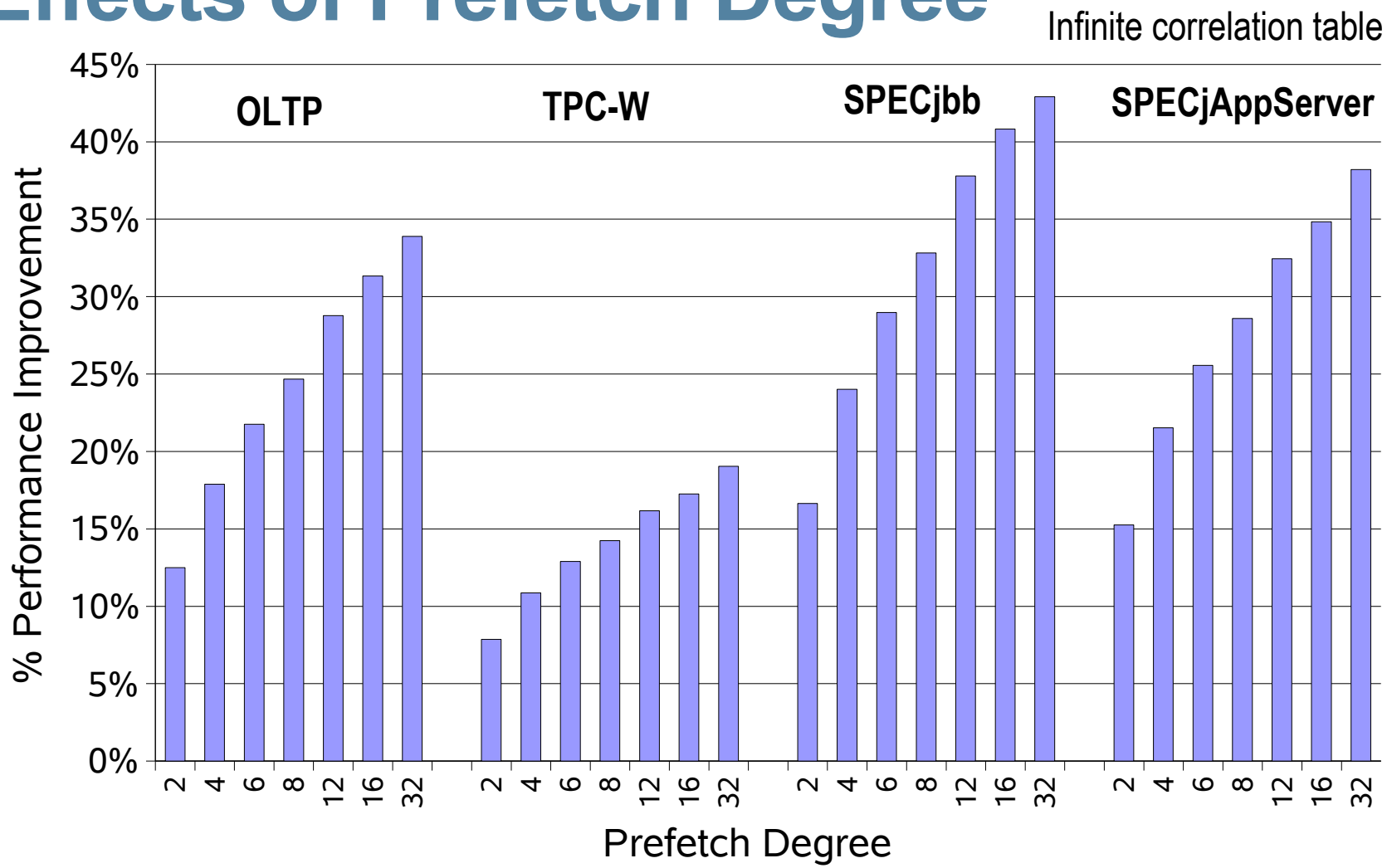
EBCP very simple and requires almost zero on-chip storage!

Experimental Results

Baseline Processor Model

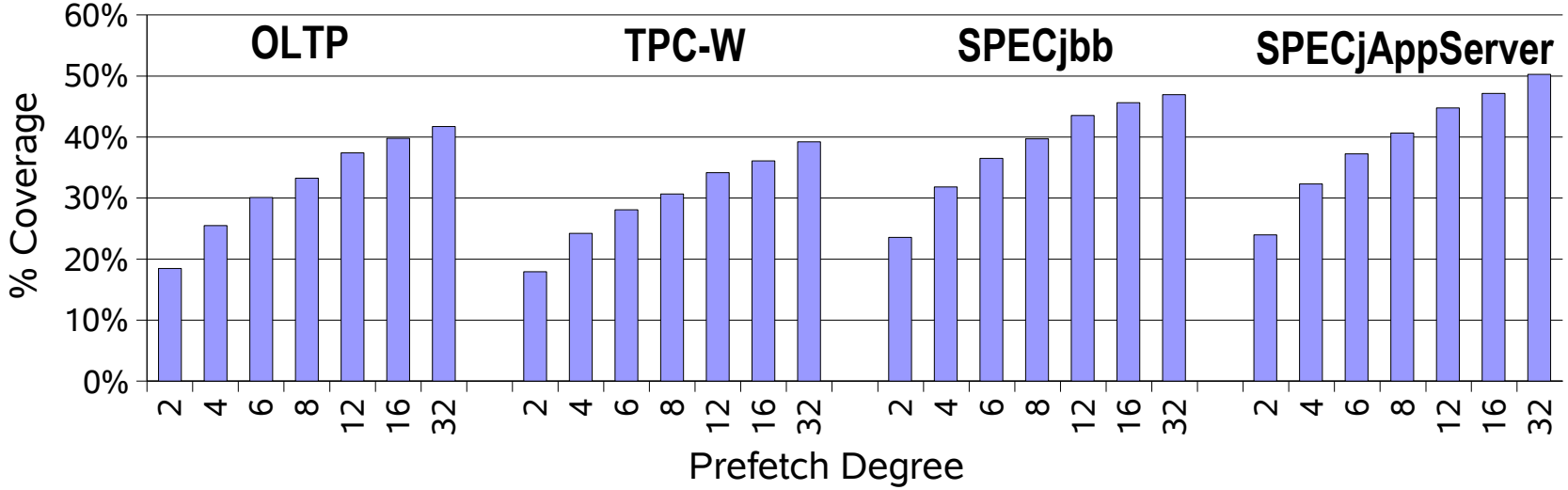
- Moderate out-of-order issue core
  - single thread
  - 4-wide issue
  - 64 entry issue queue, 128 entry reorder buffer
  - 32KB 4-way L1 instruction and data caches
  - 2MB 4-way L2 cache
  - prefetches installed into prefetch buffer
- Memory bandwidth model
  - 9.6 GB/s read bandwidth 4.8 GB/s write bandwidth
  - 500 cycle unloaded memory latency
- Commercial applications benchmarks
  - OLTP, TPC-W, SPECjbb2005, SPECjAppServer2004

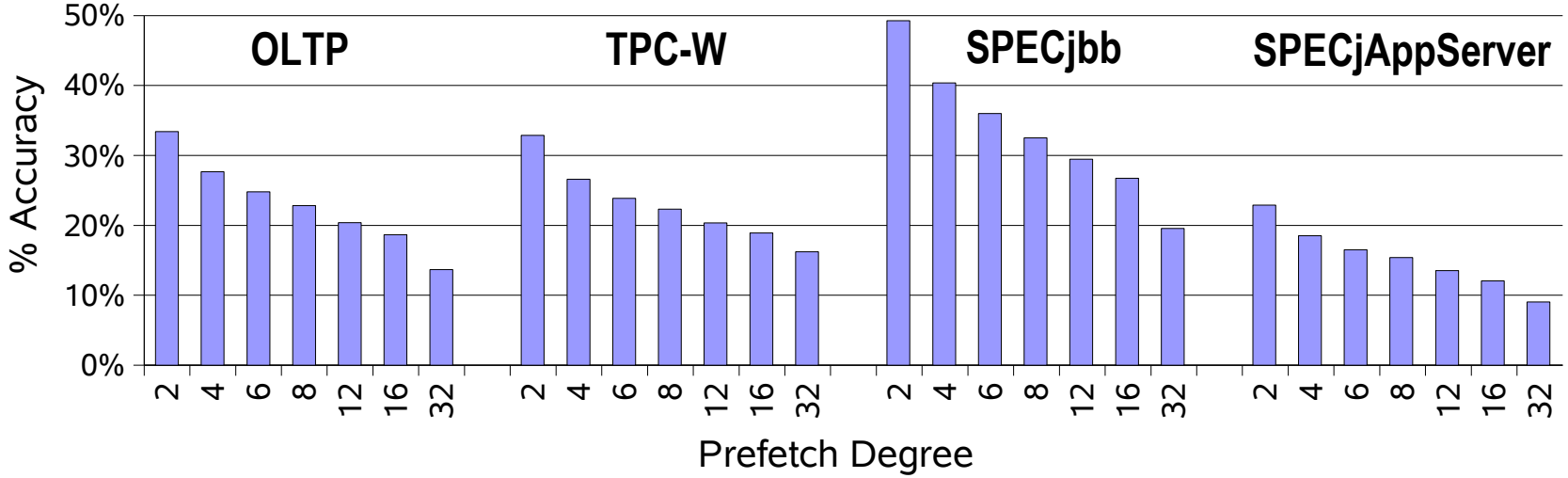
Effects of Prefetch Degree



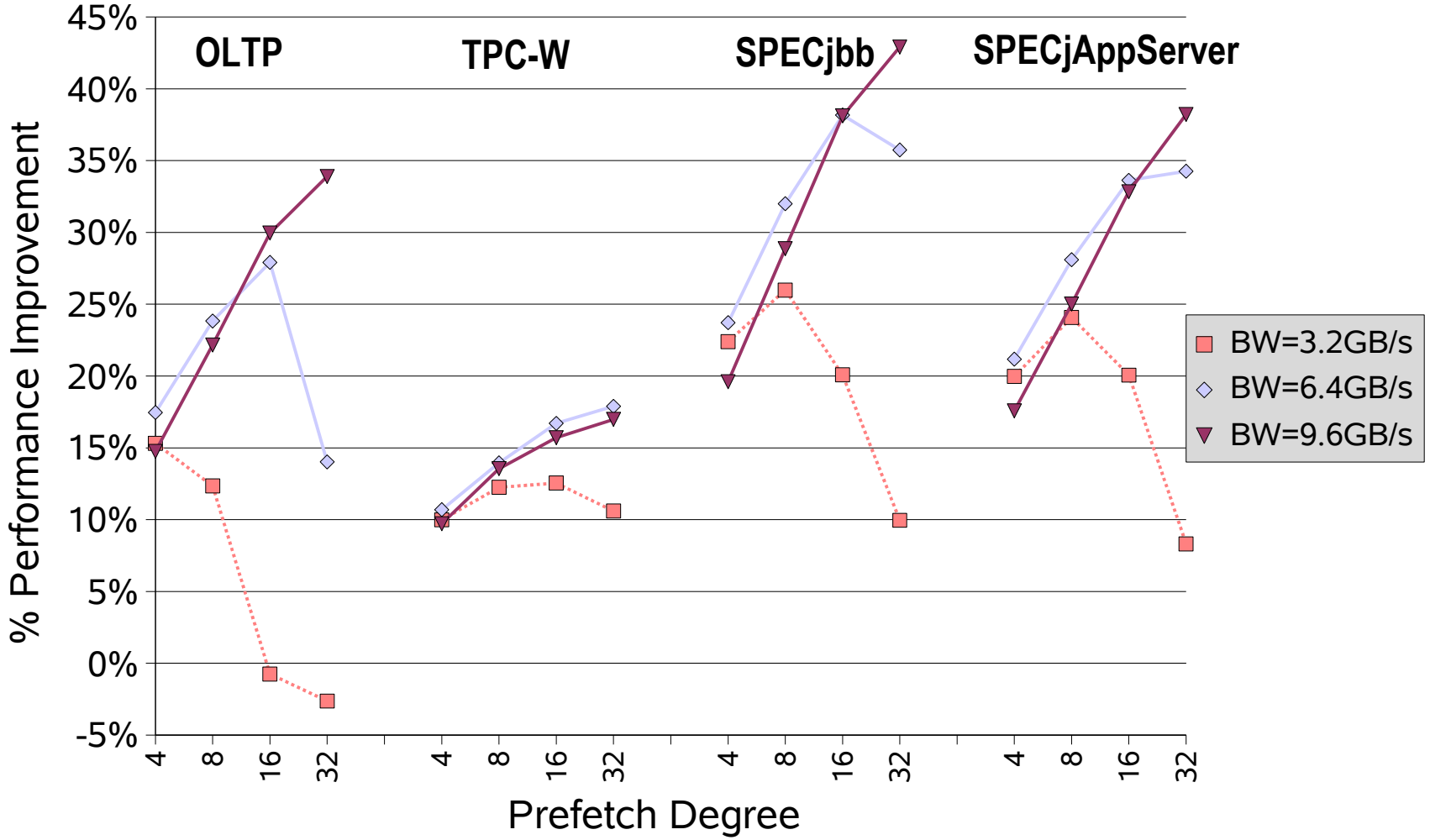
- Performance improvement increases with prefetch degree

Coverage vs Accuracy



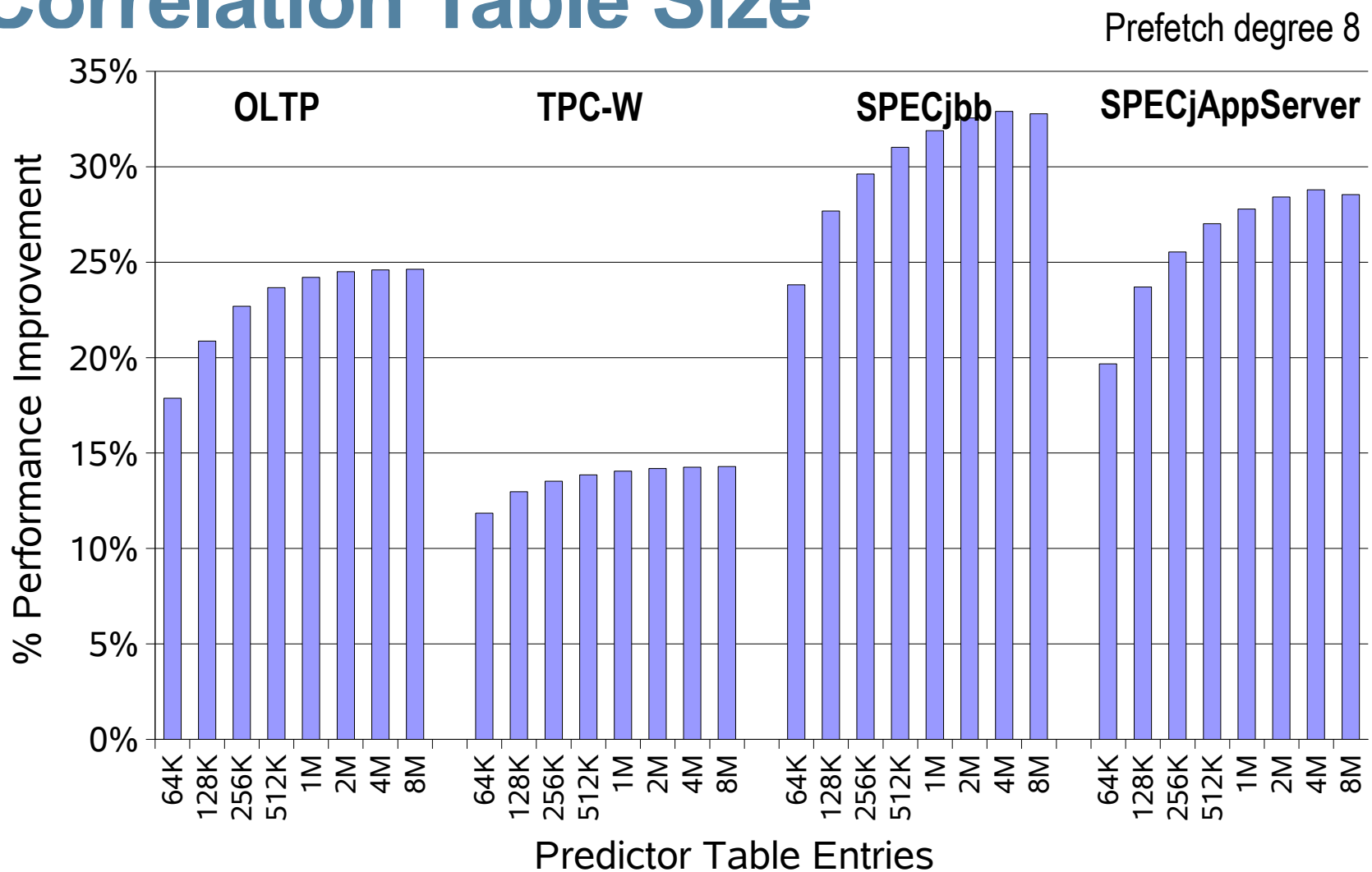


Memory Bandwidth Sensitivity



- Optimal prefetch degree depends on available memory BW

Correlation Table Size



- Storing table in main memory makes such large sizes practical

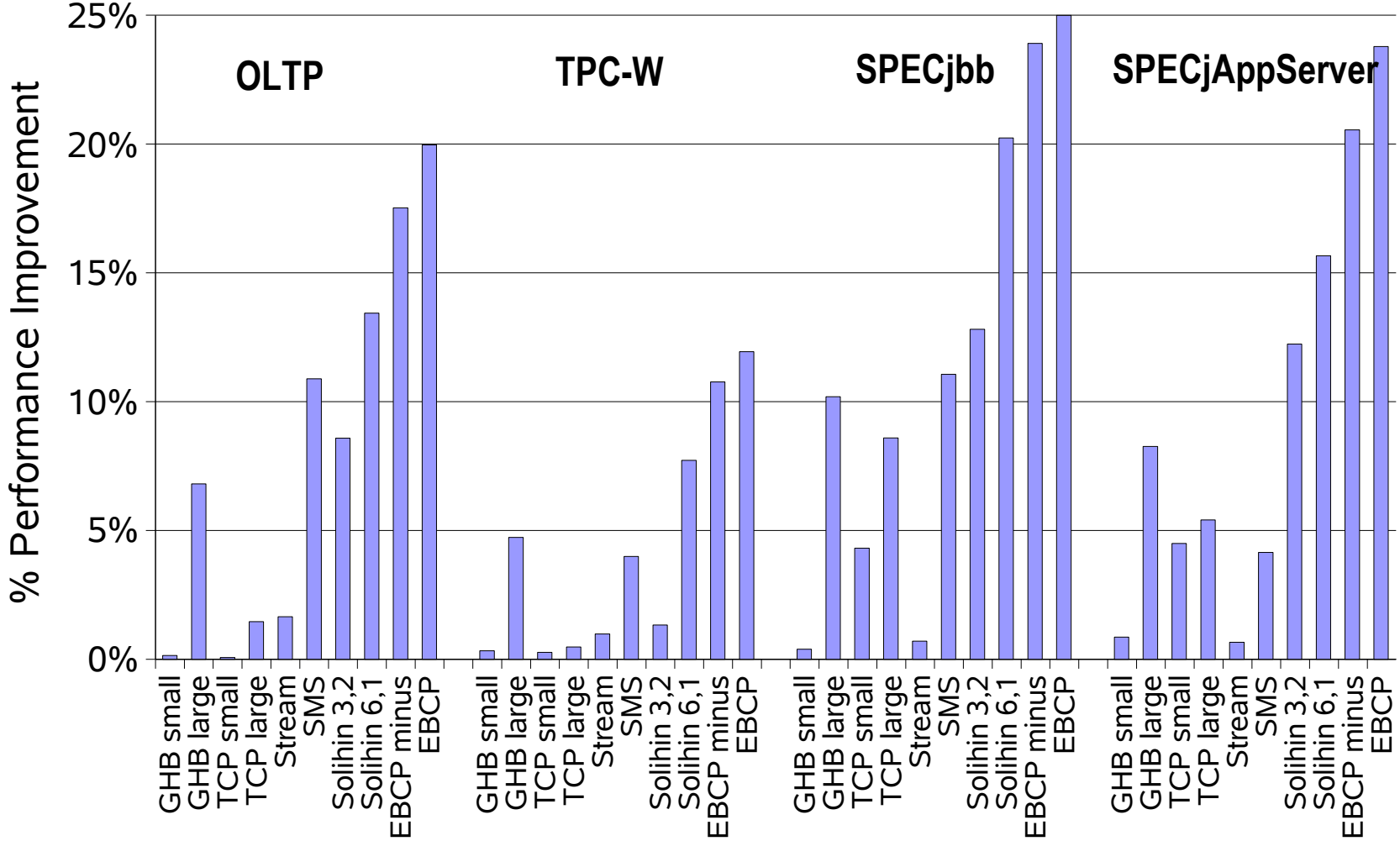
Comparison with Other Prefetchers

- Global History Buffer G/AC (GHB)
  - address correlation, unique table storage (small: 256KB large: 4MB)
- Tag Correlating Prefetcher (TCP)
  - tag correlation (small: 256KB large: 4MB)
- Stream
  - traditional stride-based stream prefetcher
- Spatial Memory Streaming (SMS)
  - spatial locality within region (128KB)
- Solihin
  - memory-side address correlation prefetcher (64MB)

Prefetch degree = 6 for all prefetchers (except SMS)

Prefetches brought into 64 entry prefetch buffer

Comparison with Other Prefetchers



- EBCP outperforms all prefetchers for all four benchmarks

Summary

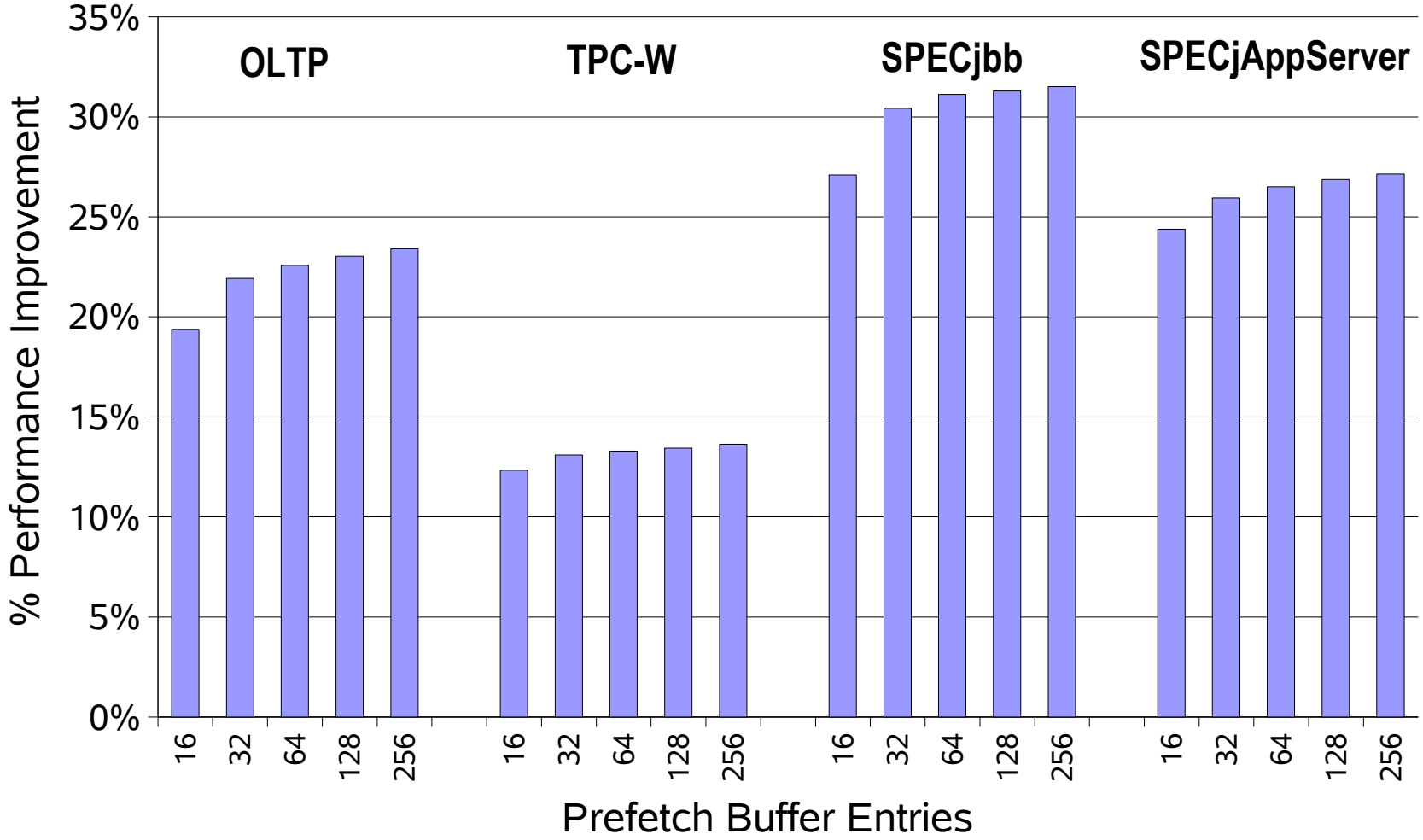
- EBCP successfully overcomes drawbacks of traditional correlation prefetchers
  - stores large correlation table in main memory
  - exploits unused memory capacity and bandwidth
  - targets removal of entire epochs
  - very simple prefetcher control
  - almost zero on-chip storage
- EBCP performs very well on all four commercial benchmarks
- Future work:
  - efficient implementation for chip multi-processors
  - improved accuracy

Epoch-based concept can be applied to other uarch techniques!

Yuan Chou
yuan.chou@sun.com

Prefetch Buffer Size

Prefetch degree 8
1 million table entries



- 64 entries sufficient for all four benchmarks