SHIFT
Shared History Instruction Fetch for Lean-Core Server Processors

Cansu Kaynak, Boris Grot, Babak Falsafi

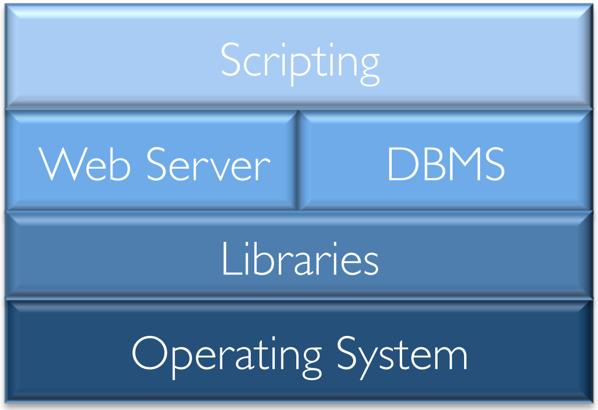
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

THE UNIVERSITY OF EDINBURGH

Instruction Fetch Stalls in Servers

Traditional and emerging server apps:

- Deep software stacks
- Multi-MB instruction working sets



Instruction fetch stalls:

- Account for up to 60% of execution time
- Cause severe core underutilization

Major performance and throughput bottleneck

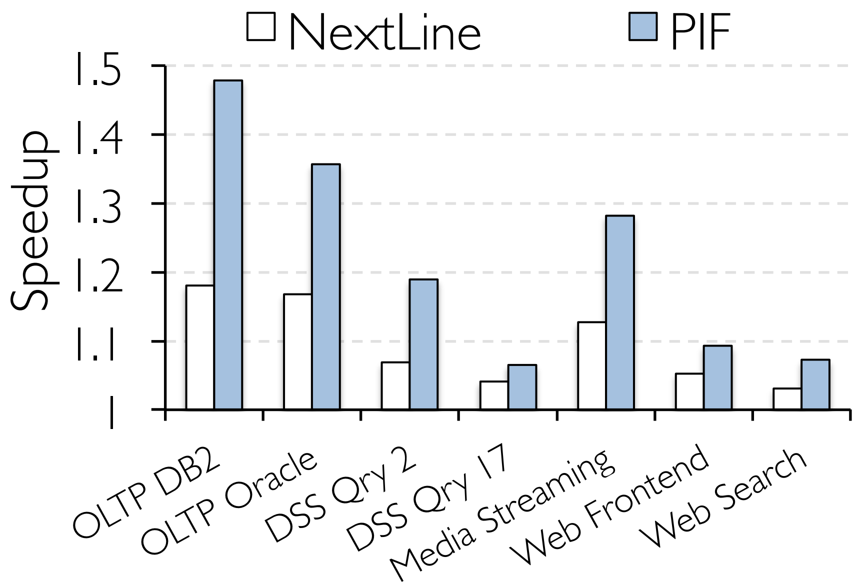
Mitigating Instruction Fetch Stalls

Next-Line Prefetcher:

- Cannot predict discontinuities

Temporal Streaming:

- Records & replays recurring
  I-Cache access sequences
- State-of-the-art: PIF

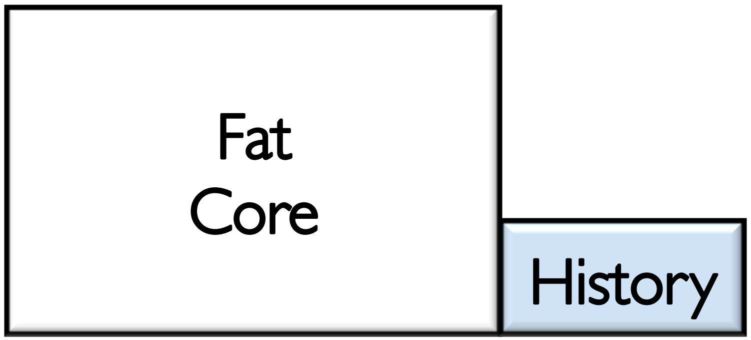


Proactive Instruction Fetch (PIF) [Ferdman'11]

- Can predict nearly all I-Cache misses
- … at cost of ~200KB per-core history

Temporal streaming is effective, but incurs prohibitive overhead

Overhead of Temporal Streaming



e.g., Intel Xeon, IBM Power

✓ High performance

✓ Negligible area overhead

e.g., ARM Cortex, Tilera

✓ High performance

✗ Significant area overhead

This Work: Practical & Effective Temporal Streaming

Shared History Instruction Fetch

Observation:

- Cores executing same app exhibit same control flow
- Significant overlap between temporal streams in history

Approach:

- History sharing across cores running common app
- Virtualized history for flexibility

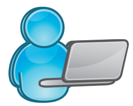
Compared to state-of-the-art:

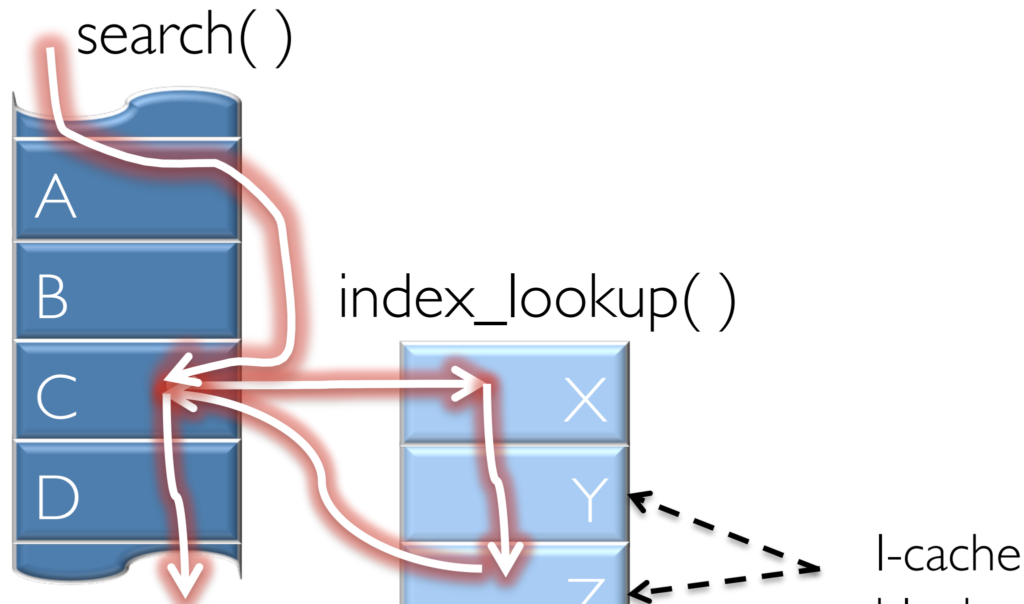
- 98% of performance with 14x less storage
- Relative area overhead: 70% ➔ 5% for leanest core

Outline

- Introduction
- Temporal Streaming
  - Background
  - Impracticality
- Instruction History Commonality
- Shared History Instruction Fetch
- Evaluation Highlights
- Conclusion

Temporal Streaming Basics





Time →

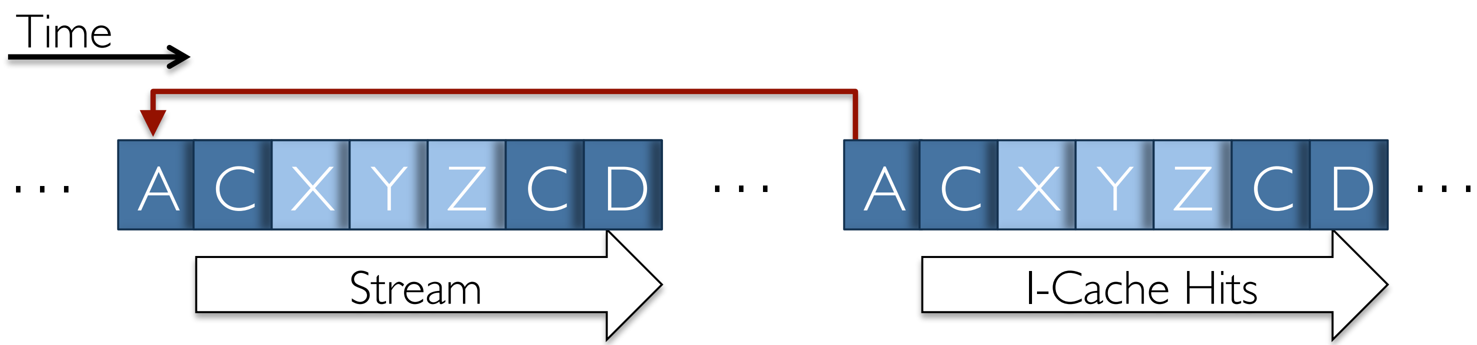
I-Cache Access Sequence: … A C X Y Z C D … A C X Y Z C D …

Recurring control flow → Recurring temporal streams

Exploiting Temporal Streams

State-of-the-art: Proactive Instruction Fetch [Ferdman'11]

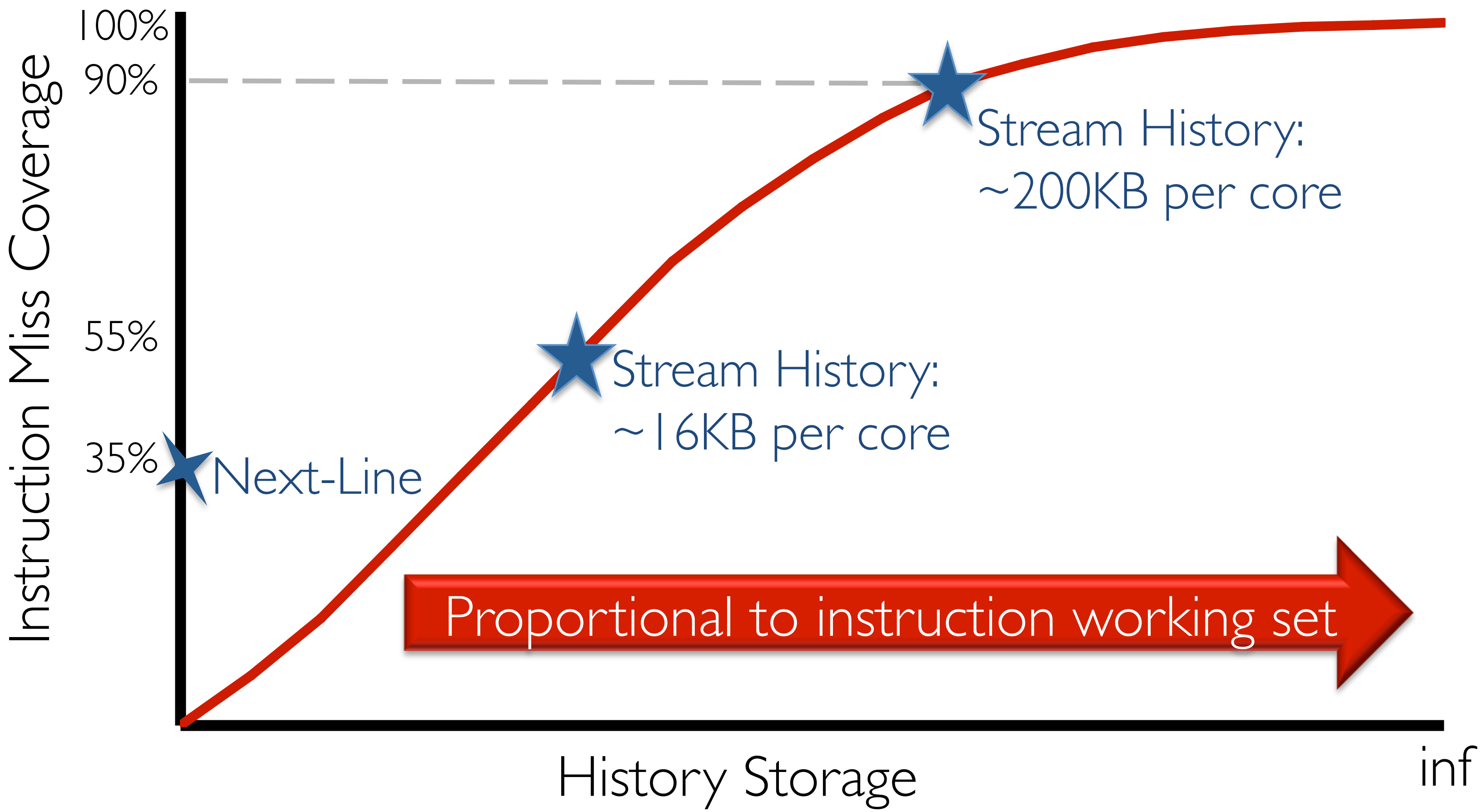
1. Record sequences of accesses
2. Locate latest occurrence of a sequence
3. Replay old sequence as prediction



Stable control flow & deep software stack
➔ Many long temporal streams

How much storage is required to capture these streams?

Streaming Effectiveness vs. Storage Overhead



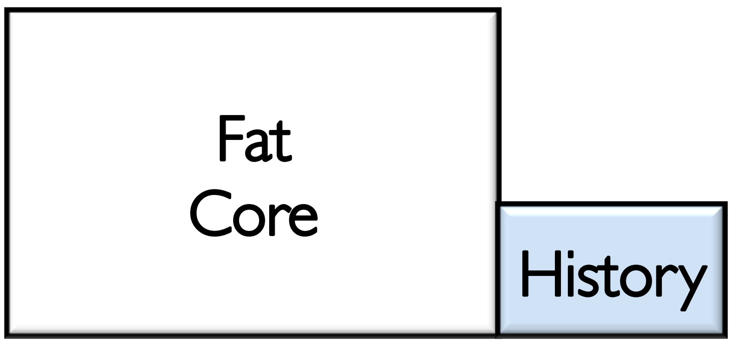
More history → More misses eliminated

Cost of Temporal Streaming

Instruction history size is function of instruction working set

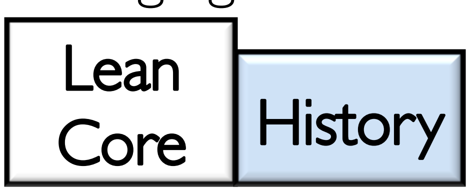
- Independent of core type!

Conventional Fat Cores



- e.g., Intel Xeon, IBM Power
- Core area much larger than history
- 4% of Xeon core

Emerging Lean Cores



- e.g., ARM Cortex, Tilera
- History storage approaches core area
- 70% of Cortex A8 core

Need for effective & low-overhead temporal streaming

Reducing Temporal Streaming Overhead

Less history storage per core:

- ✘ Much less coverage & performance

Predictor Virtualization [Burcea'08]:

Embed per-core instruction history into LLC

- ✘ Storage & write traffic scales with #cores

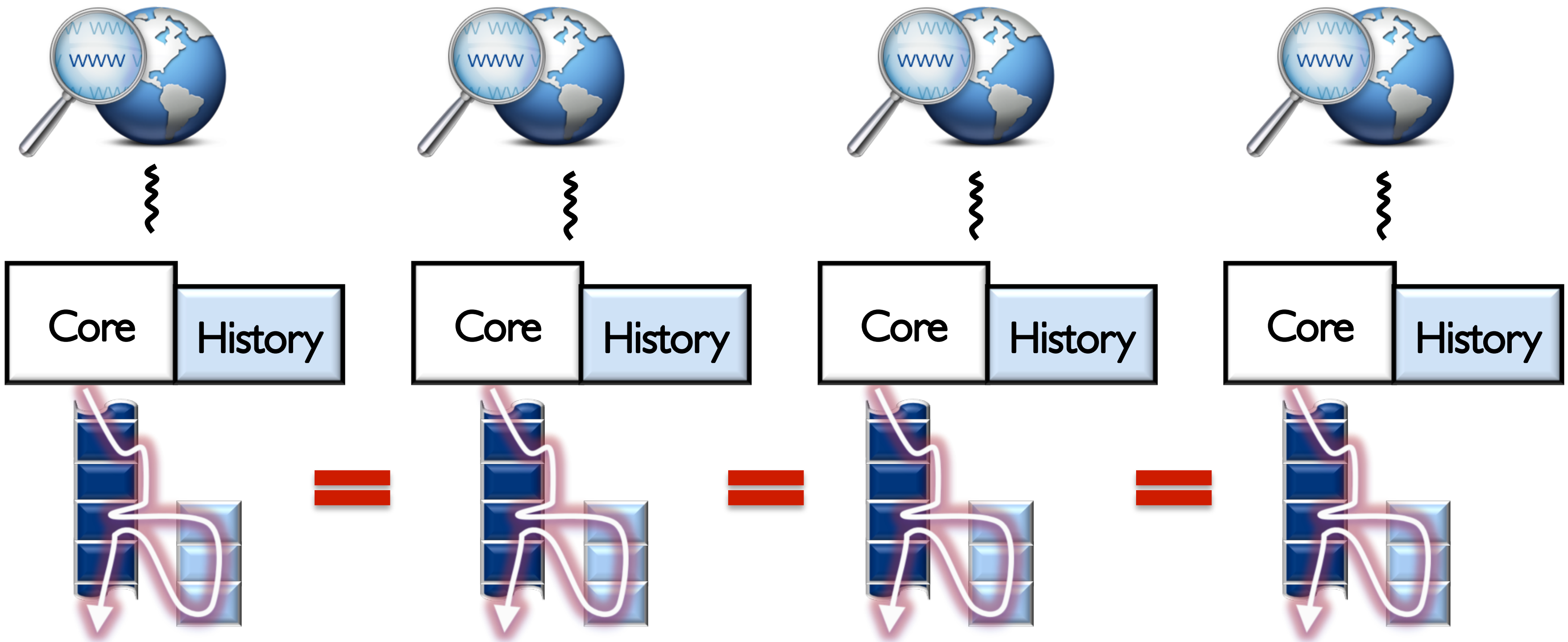
Need for a solution that preserves performance

Outline

- Introduction
- Temporal Streaming
- Instruction History Commonality
- Shared History Instruction Fetch
- Evaluation Highlights
- Conclusion

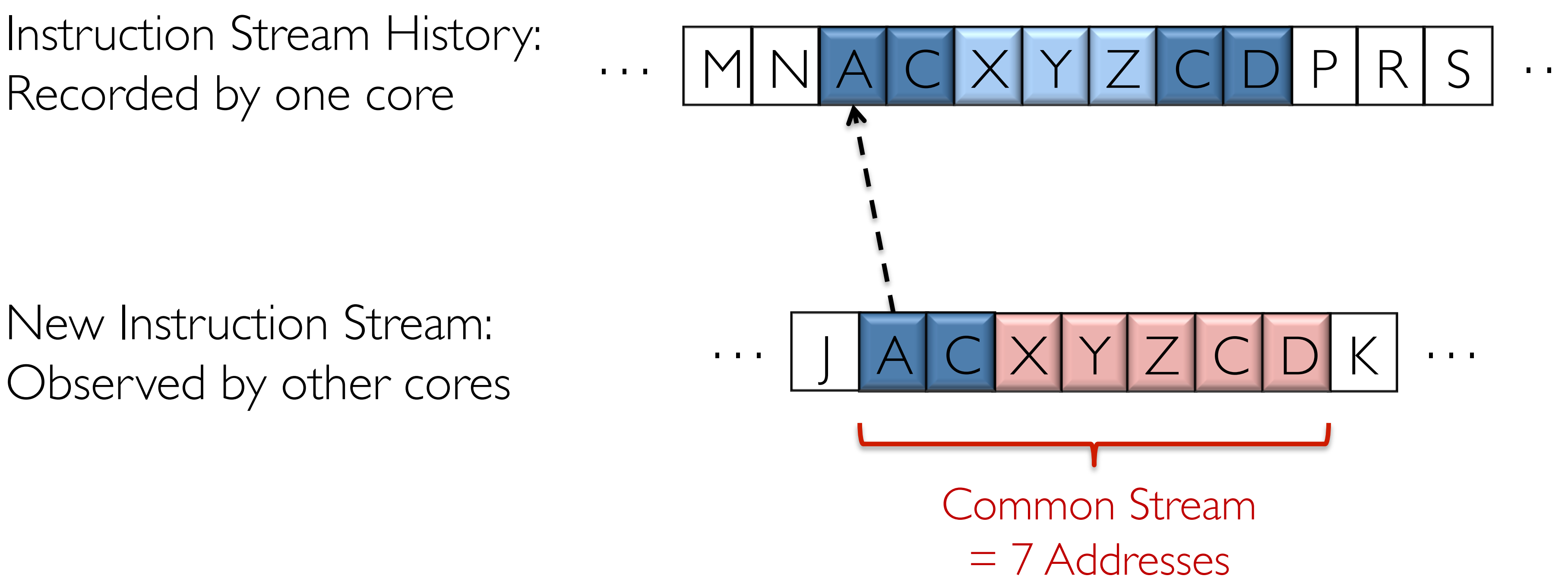
Instruction History Commonality Across Cores

- Worker threads execute same types of requests from clients
- Cores exhibit common control flow



Instruction histories overlap significantly

Quantifying Instruction History Commonality



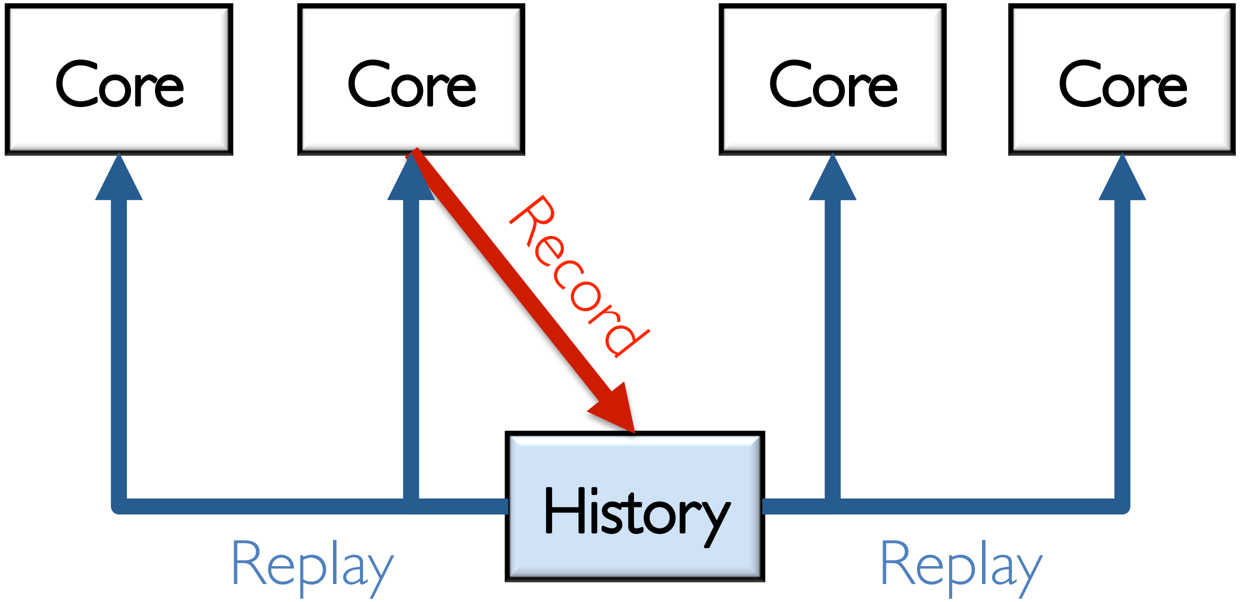
>90% of I-cache accesses in common streams

Outline

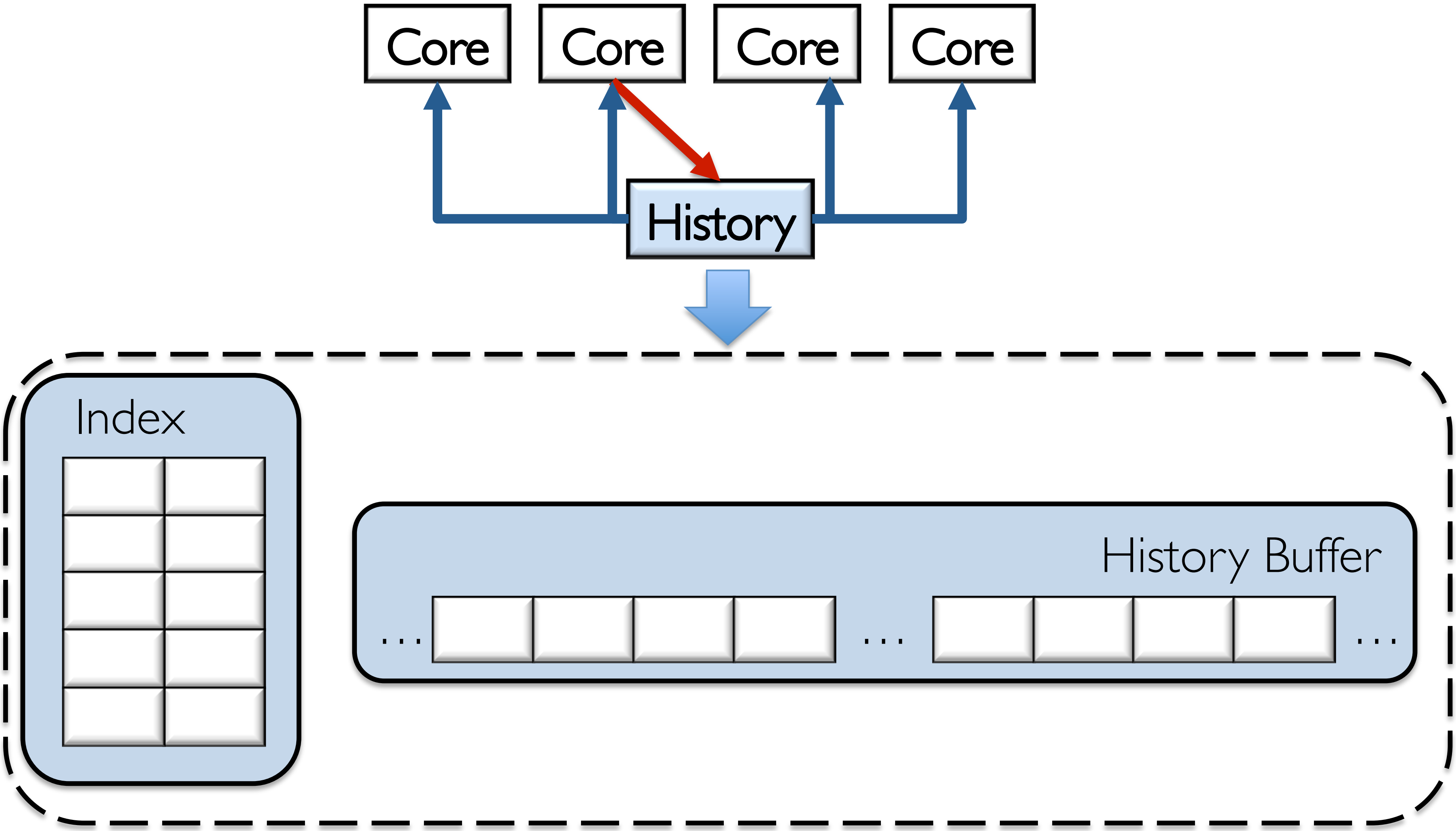
- Introduction
- Temporal Streaming
- Instruction History Commonality
- Shared History Instruction Fetch
- Evaluation Highlights
- Conclusion

Shared History Instruction Fetch (SHIFT)

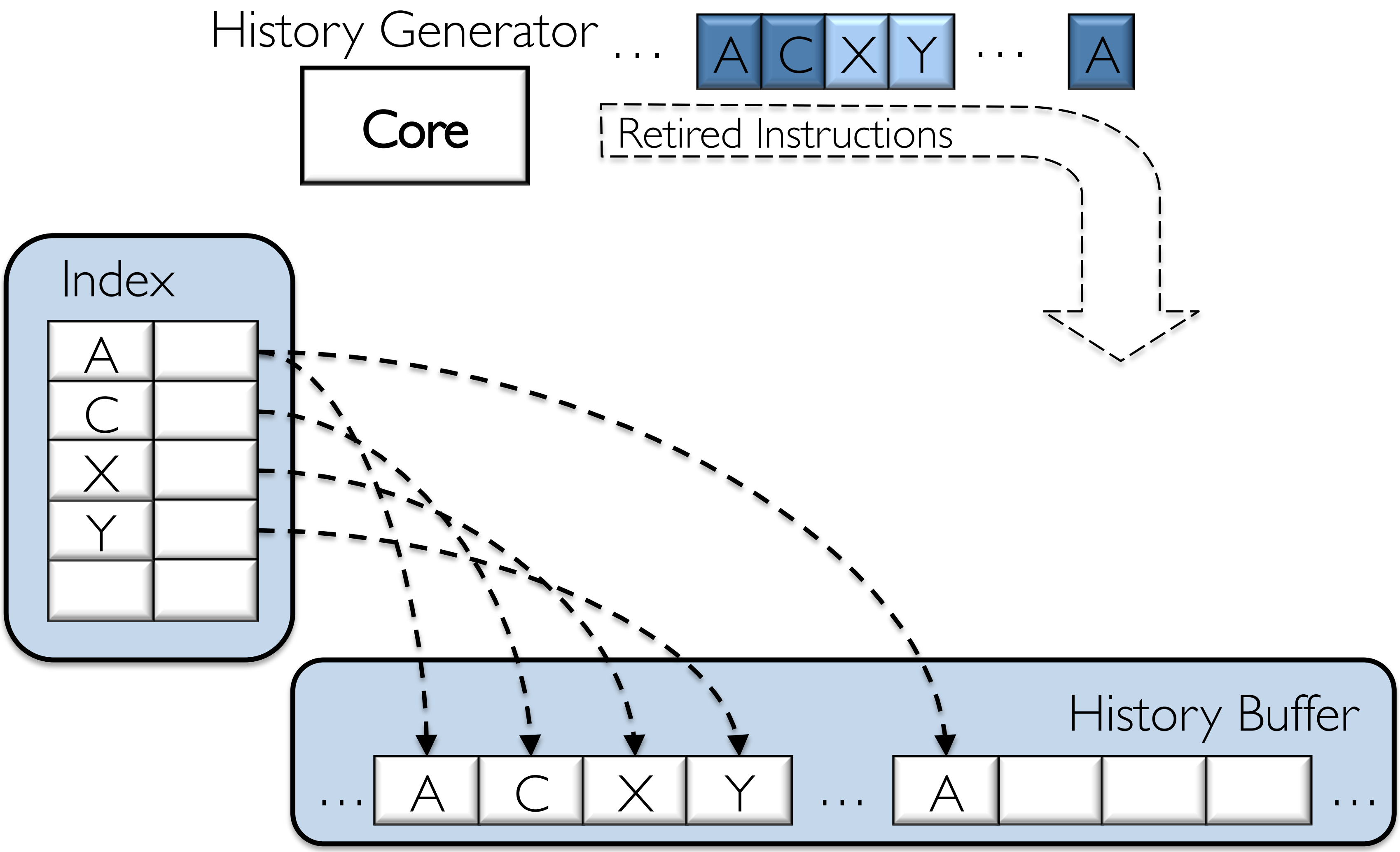
- Single shared history across cores running common app
- One core picked at random generates history
- All cores replay shared history for streaming



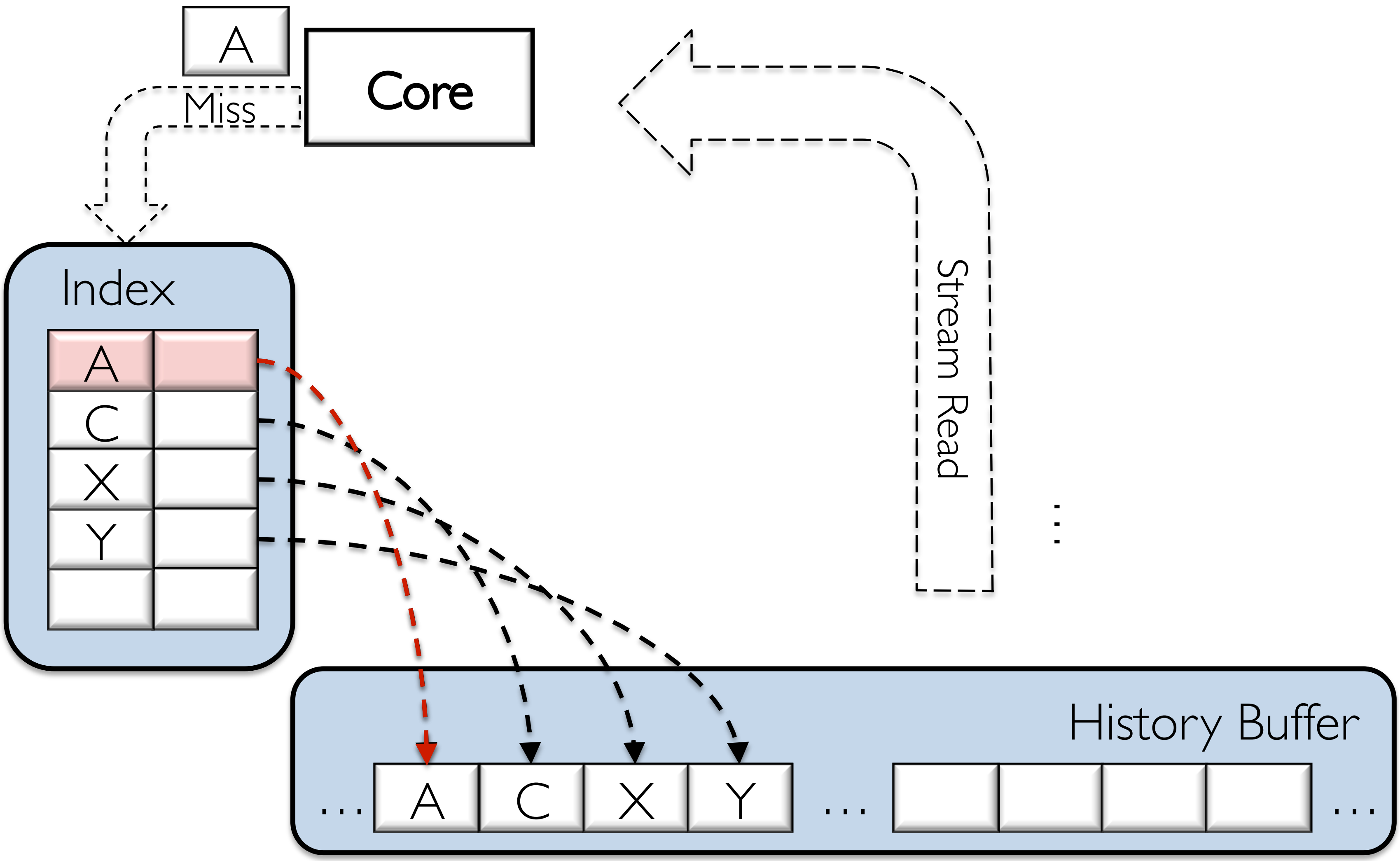
Shared Instruction History Storage



Recording Temporal Streams

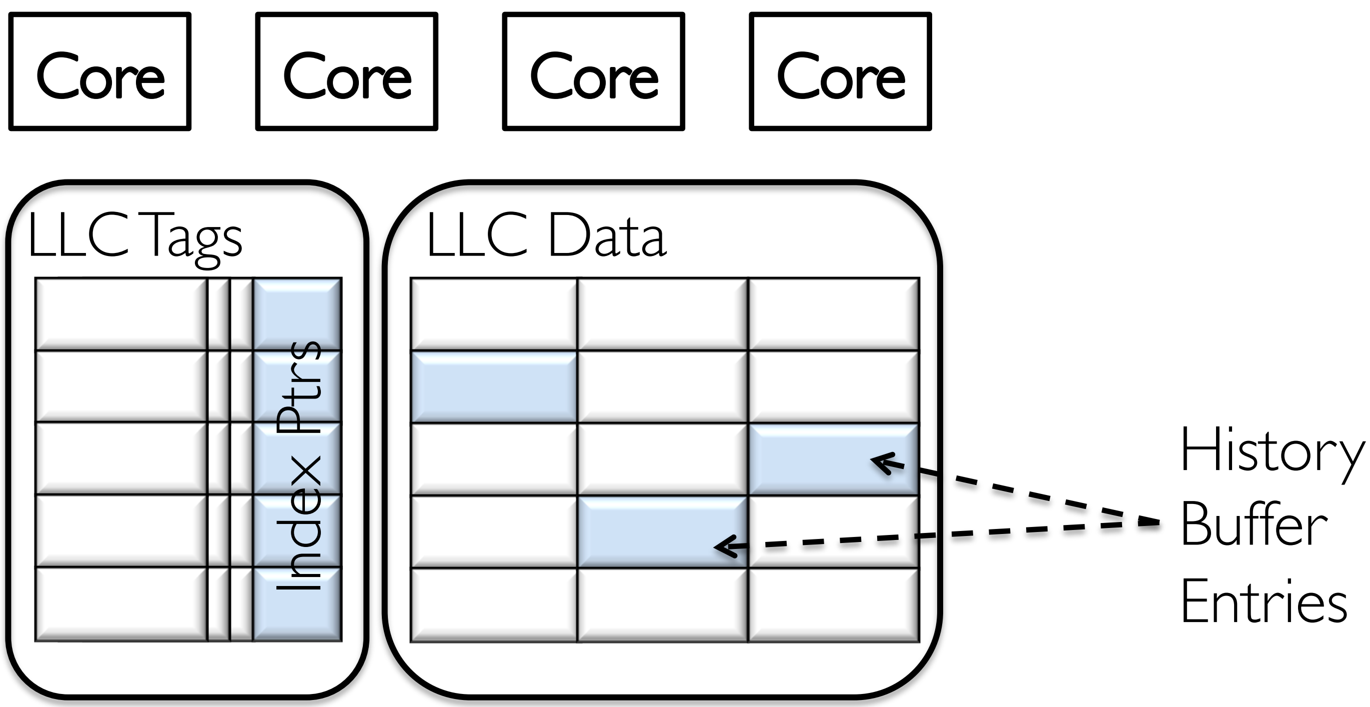


Replaying Temporal Streams



Virtualizing SHIFT

- Shared instruction history:
  - Minimizes aggregate storage requirements & history writes
  - Allows embedding shared history into LLC [Burcea'08]



Shared history allows for history virtualization

Outline

- Introduction
- Temporal Streaming
- Instruction History Commonality
- Shared History Instruction Fetch
- Evaluation Highlights
- Conclusion

Methodology

- FLEXUS full-system trace and OoO timing simulator [Wenisch'06]

Traditional Server Apps:

- OLTP
- DSS
- Web Frontend

Emerging Server Apps:

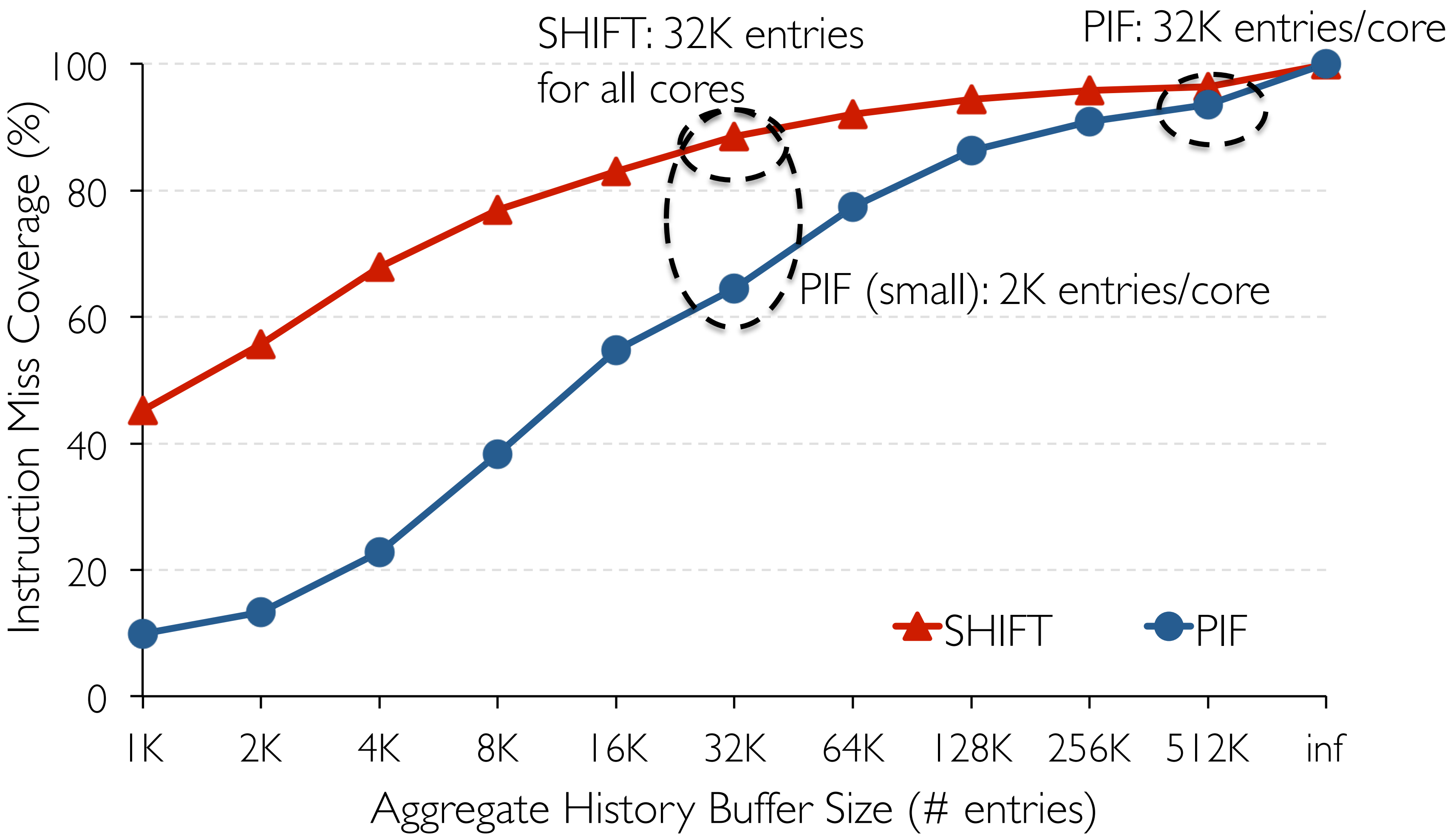
- Media Streaming
- Web Search

Simulated System:

- 16-core processor @ 2GHz
- Cortex A8- & A15-like core
- L1 (I and D): 32KB, 2-way
- NUCA L2: 8MB, 16-way
- On-Chip Network: Mesh

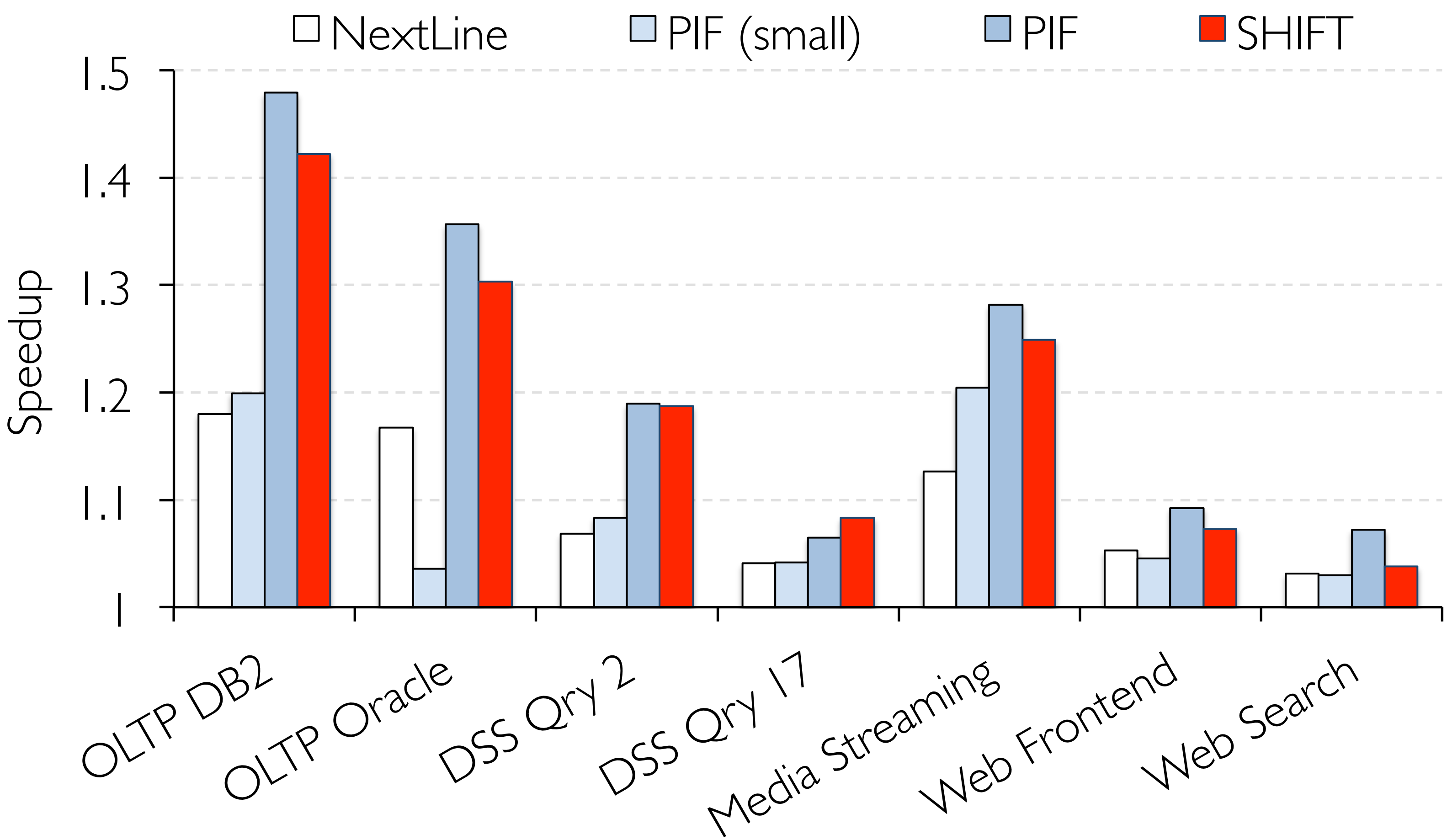
- Comparison w/ state-of-the-art instruction prefetcher:
  - Proactive Instruction Fetch (PIF) [Ferdman'11]

Miss Coverage Comparison



SHIFT outperforms PIF for any given aggregate history size

Performance Comparison



SHIFT preserves 98% of PIF's performance with 14X less storage

Conclusion

- I-misses are critical for server performance
- Temporal Streaming is effective
  - But incurs prohibitive storage overhead
- Shared History Instruction Fetch
  - Minimizes storage overhead by sharing history
  - Preserves benefits of temporal streaming

Thanks!

Questions?