Accurately and Maximally Prefetching Spatial Data Access Patterns with Bingo

Mohammad Bakhshalipour
Mehran Shakerinava
Pejman Lotfi-Kamran
Hamid Sarbazi-Azad

Presenter: Farid Samandi (Stony Brook University)

Sharif University of Technology

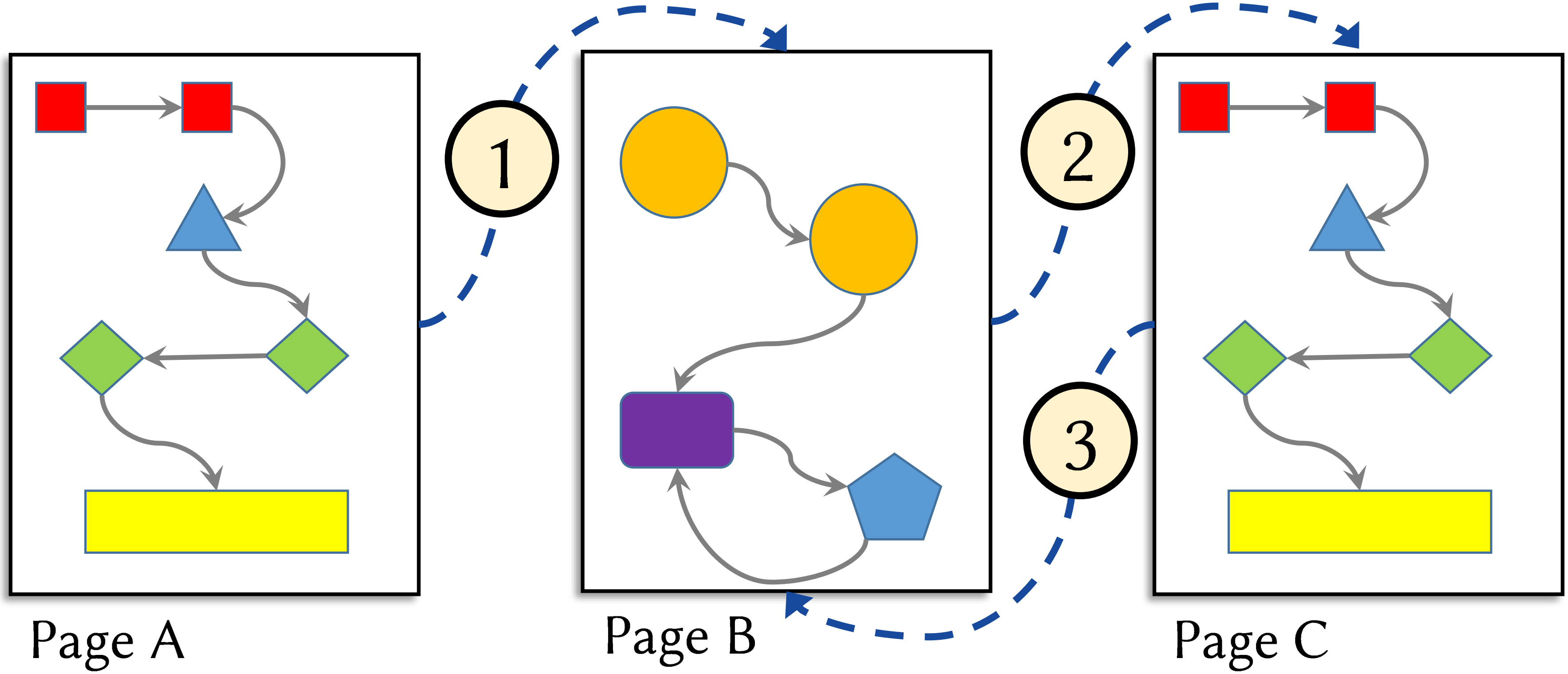
Institute for Research in Fundamental Sciences

The Third Data Prefetching Championship (DPC3), in conjunction with ISCA 2019

Spatial Data Correlation

Access patterns repeat over memory pages

  ❑ Because data objects have fixed and regular layout

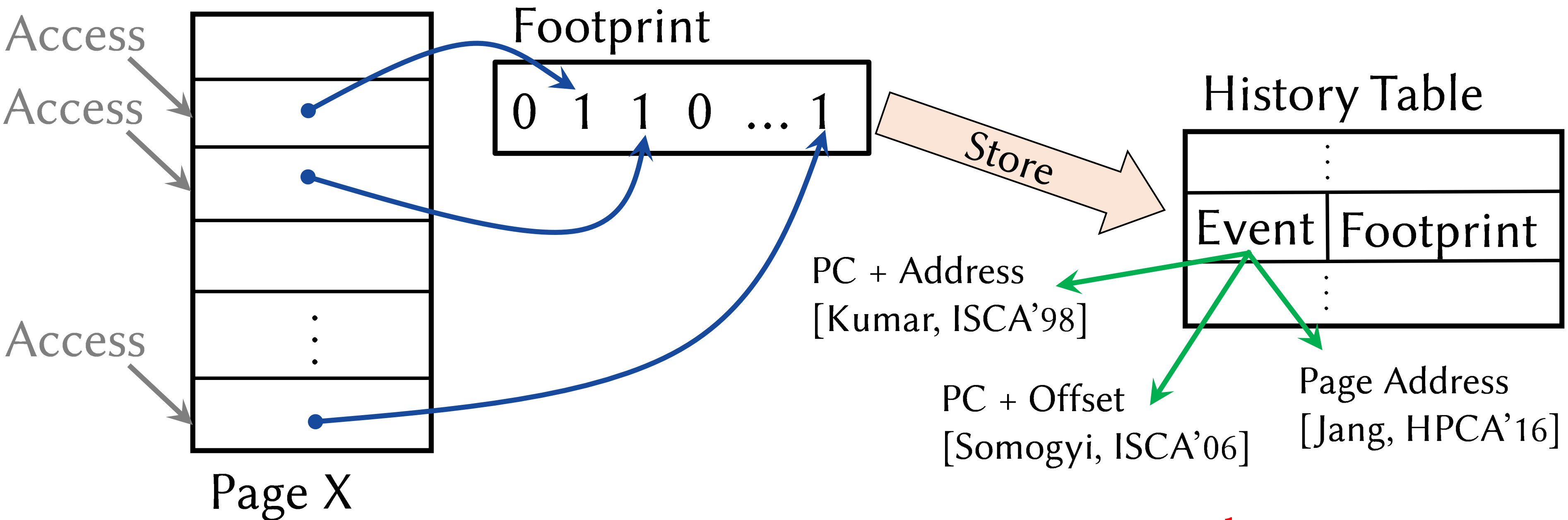


Spatial Data Prefetching

State-of-the-art

Per-Page History Prefetchers

- ❑ Record a footprint for each page
- ❑ Correlate the recorded footprint with **one** event
  - ➢ The event is usually extracted from the trigger access

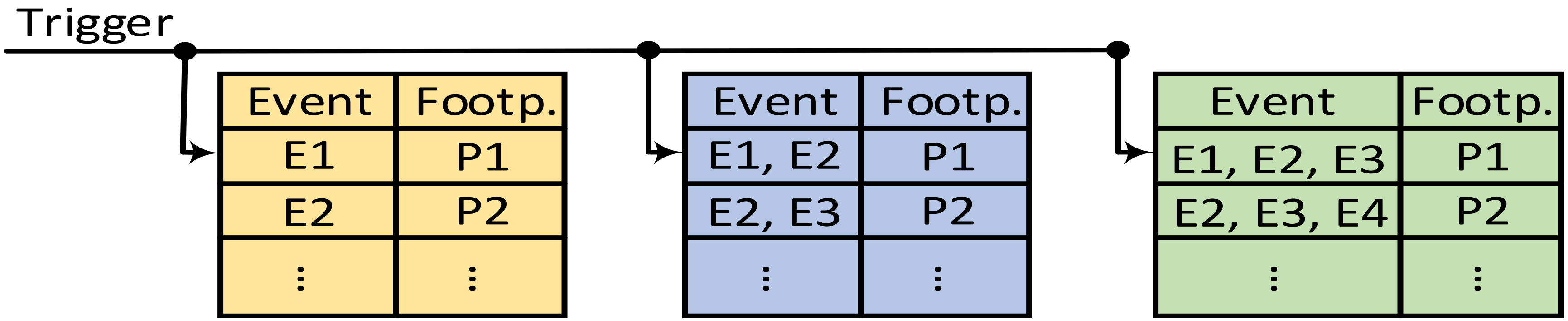


One event is not accurate enough

Our proposal: Bingo

Correlates each footprint with **multiple** events

- ❑ Employ **TAGE-like** history organization



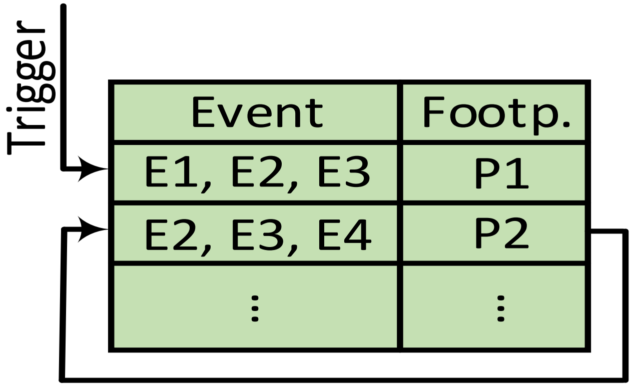
Event	Footp.
E1	P1
E2	P2
⋮	⋮

Event	Footp.
E1, E2	P1
E2, E3	P2
⋮	⋮

Event	Footp.
E1, E2, E3	P1
E2, E3, E4	P2
⋮	⋮

Significant storage overhead because of **redundancies**

- ❑ **Consolidate** metadata information



Event	Footp.
E1, E2, E3	P1
E2, E3, E4	P2
⋮	⋮

Look up with a
different event

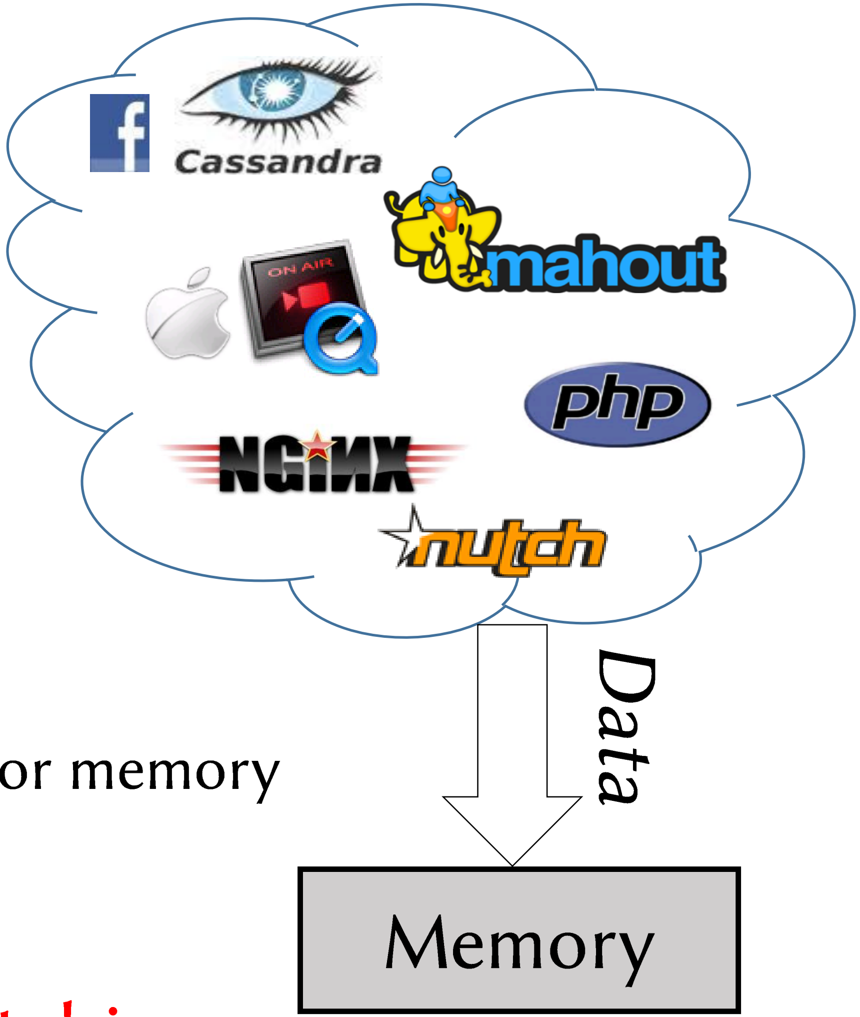
Outline

- ❑ Introduction
- ❑ Motivations
- ❑ Bingo
- ❑ Results
- ❑ Conclusion

Big-Data Applications

Large datasets

- □ Dwarf on-chip caches
- □ Long-latency memory accesses



Data

Memory

Performance implication

- □ > 50% of execution time waiting for memory [Ferdman, ASPLOS 2012]

Data Prefetching

Hardware Data Prefetching

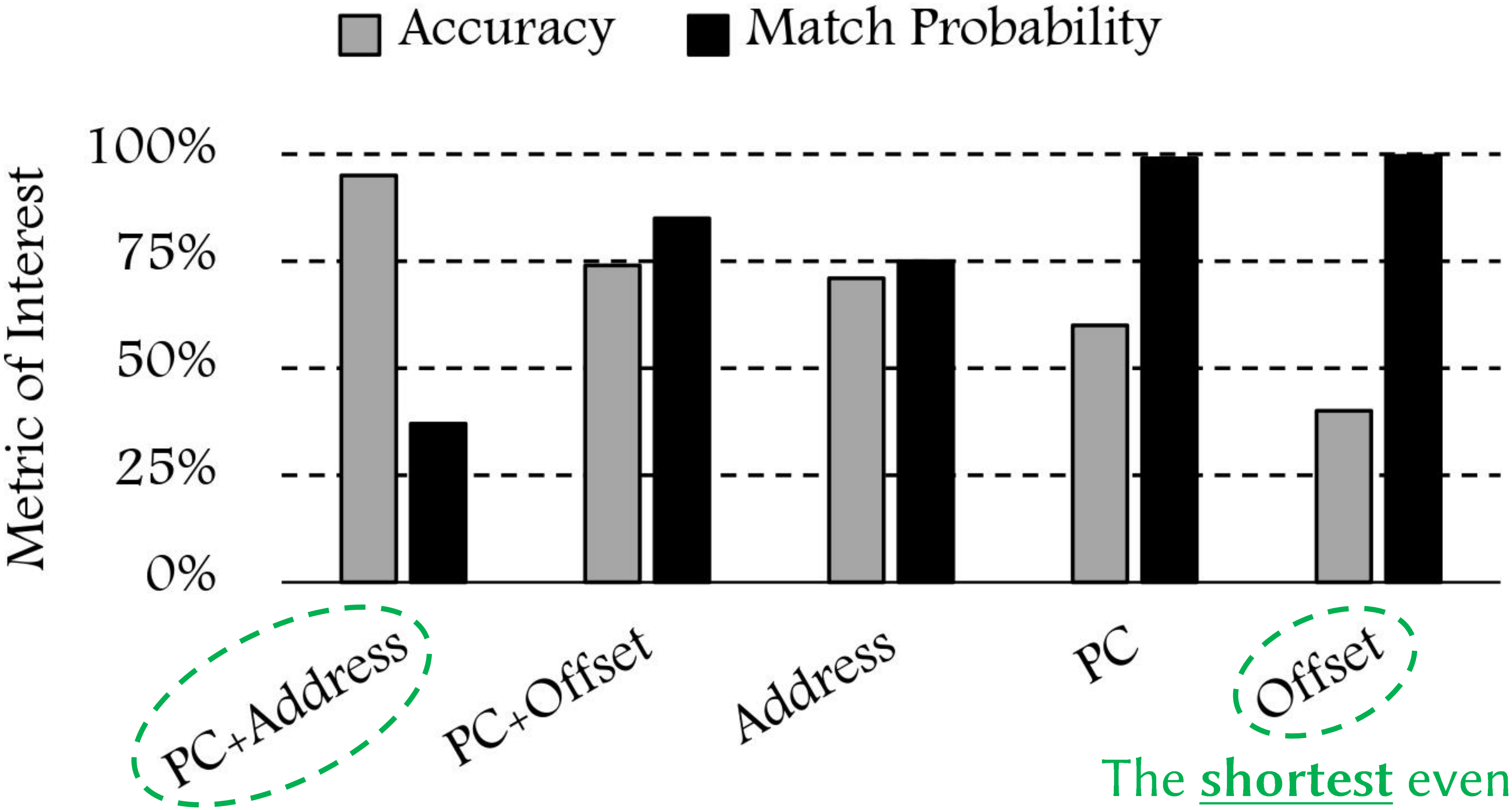
Predict future memory accesses and fetch them proactively

- ❑ Temporal prefetching
- ❑ Spatial prefetching
  - ✓ Low storage overhead
  - ✓ Overcome unseen cache misses

Spatial prefetching

- ❑ Records and replays data accesses in a per-page basis manner
- ❑ Works since applications use data objects with a regular and fixed layout
  - ✓ Accesses reappear while traversing data structures

What is the Best Event?



The **longest** event

The **shortest** event

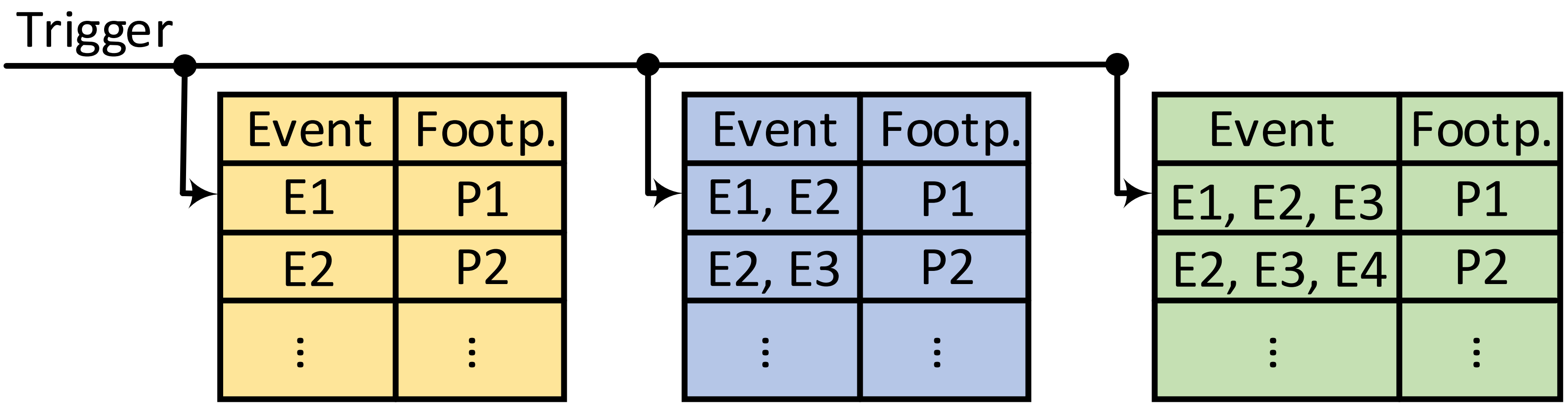
No single event has all good characteristics

→ Use **multiple** events

TAGE-Like Predictor

A TAGE-like predictor can bring the benefits of both worlds

- ❑ Correlate footprints to both long and short events
- ❑ Upon prediction: start from the longest event
  - ✓ In case of match → Use event for prediction
  - × Otherwise → Check the next-longest event



Event	Footp.
E1	P1
E2	P2
⋮	⋮

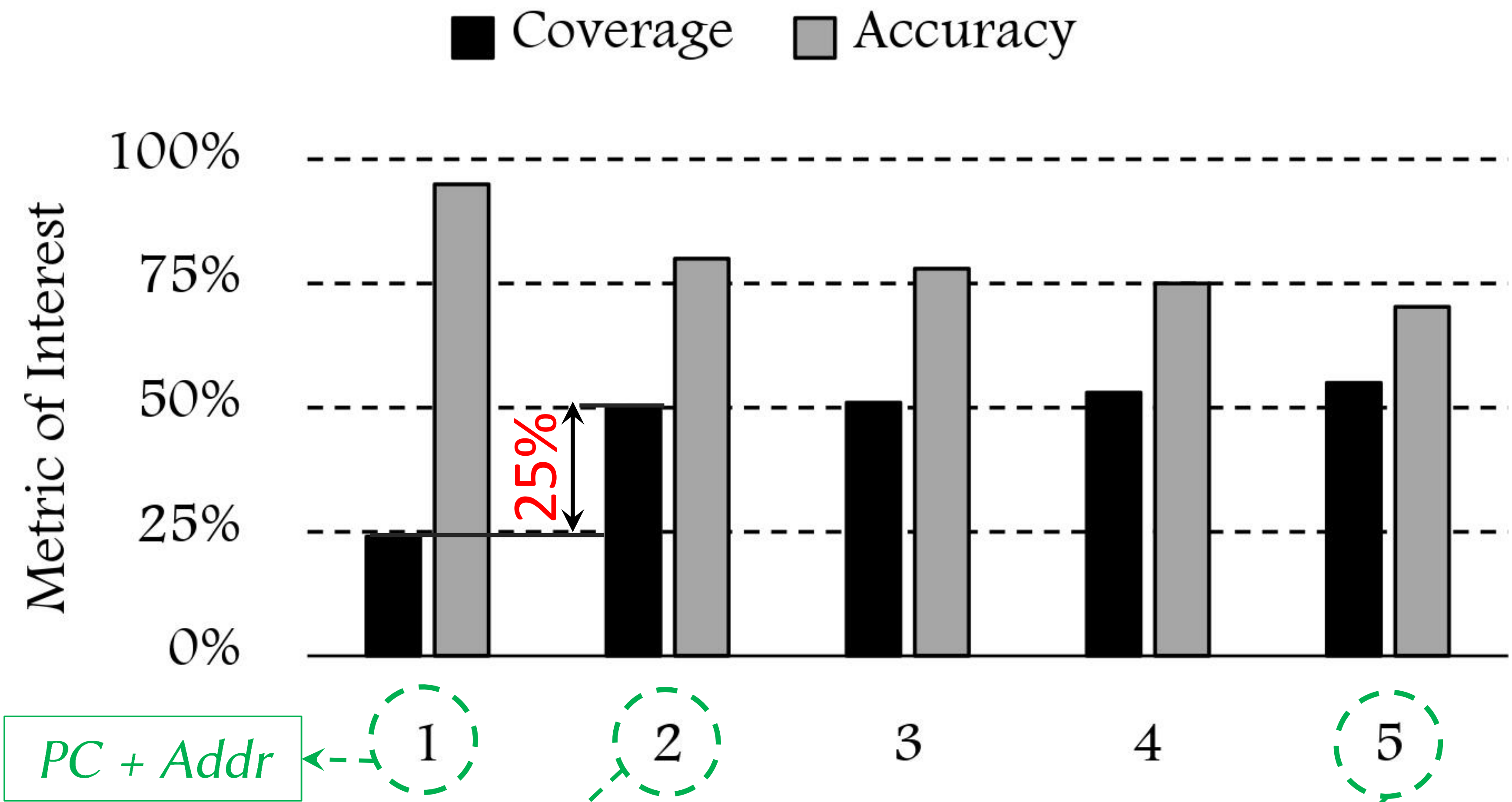
Event	Footp.
E1, E2	P1
E2, E3	P2
⋮	⋮

Event	Footp.
E1, E2, E3	P1
E2, E3, E4	P2
⋮	⋮

The Shortest History Table
Low Accuracy
High Probability

The Longest History Table
High Accuracy
Low Probability

How Many Events?



PC + Addr

PC + Addr
PC + Offset

All events

Two events suffice

A Naïve Implementation

Use multiple history tables

- ❑ Like all prior TAGE-like approaches

Long History Table

Event	Footprint
PC_1, $Address_1$	1000..10
PC_2, $Address_2$	0110..11
⋮	⋮

Short History Table

Event	Footprint
PC_1, $Offset_1$	1110..01
PC_2, $Offset_2$	0110..11
⋮	⋮

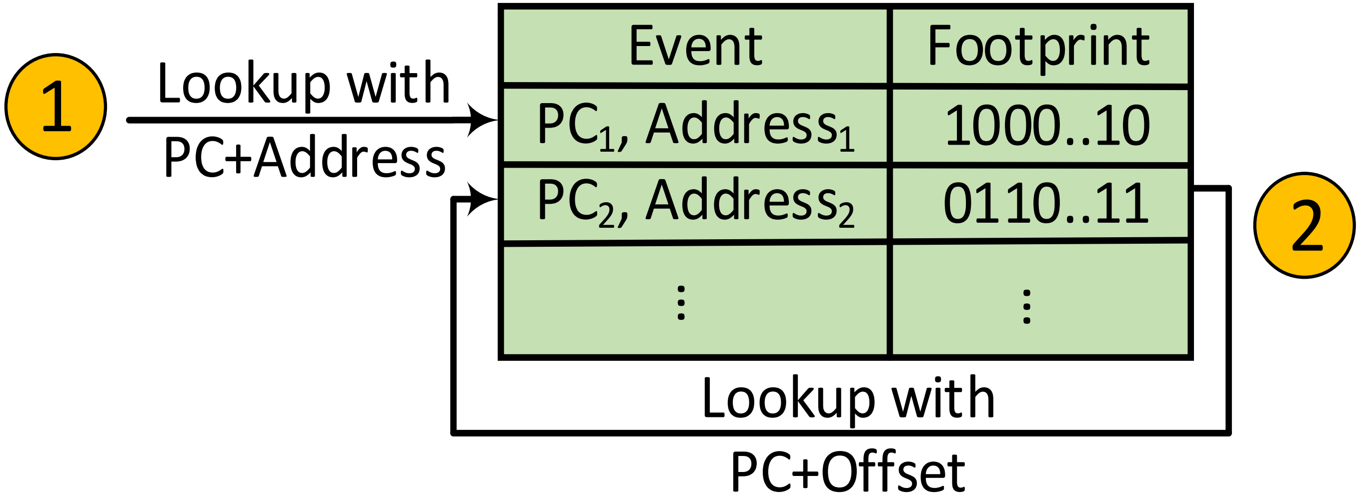
Significant storage overhead due to **redundancies**

Bingo: Consolidate History Tables

Instead of having multiple history tables, employ only one history table but **look it up multiple times** each time with a different event

- Store footprint information paired with **only the longest event**
- But look up the history table **with both long and short events**

Insight: *Short events are carried in long events*



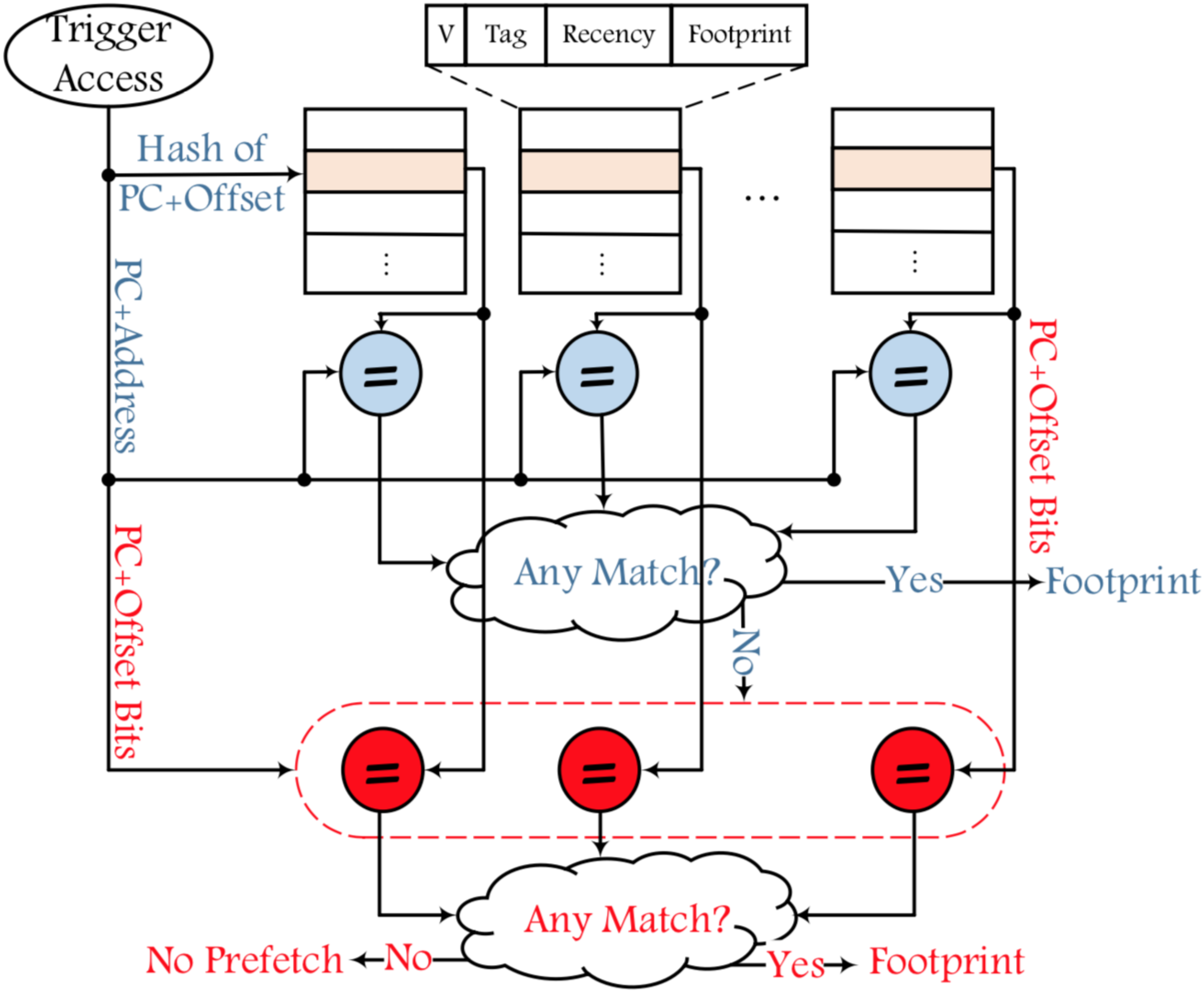
1 Lookup with PC+Address

Event	Footprint
PC_1, $Address_1$	1000..10
PC_2, $Address_2$	0110..11
⋮	⋮

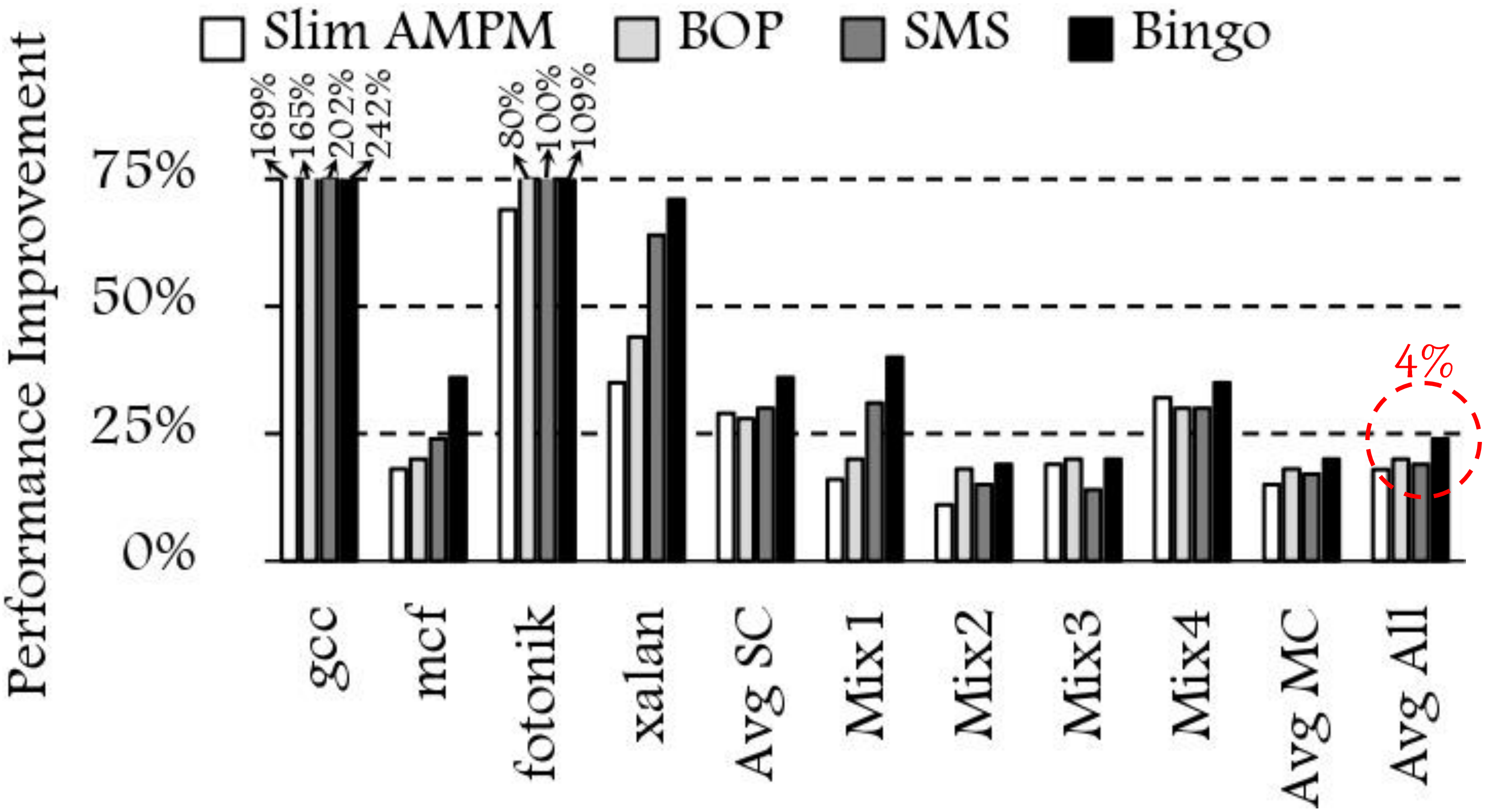
2 Lookup with PC+Offset

Consolidation → Automatically eliminating redundancies

Bingo: Details



Performance



Bingo outperforms state-of-the-art data prefetcher by 4% on average across 146 workloads

Conclusions

Big-data applications

- ❑ Huge memory-resident datasets
- ❑ Frequent data stalls
- ❑ Data prefetching for improving both throughput and latency

Bingo

- ❑ Uses both long and short events for correlation prefetching
- ❑ Consolidates history tables for storage efficiency
- ❑ Improves system performance by 23% over the baseline and 4% over prior best-performing data prefetcher

Thanks for your attention!