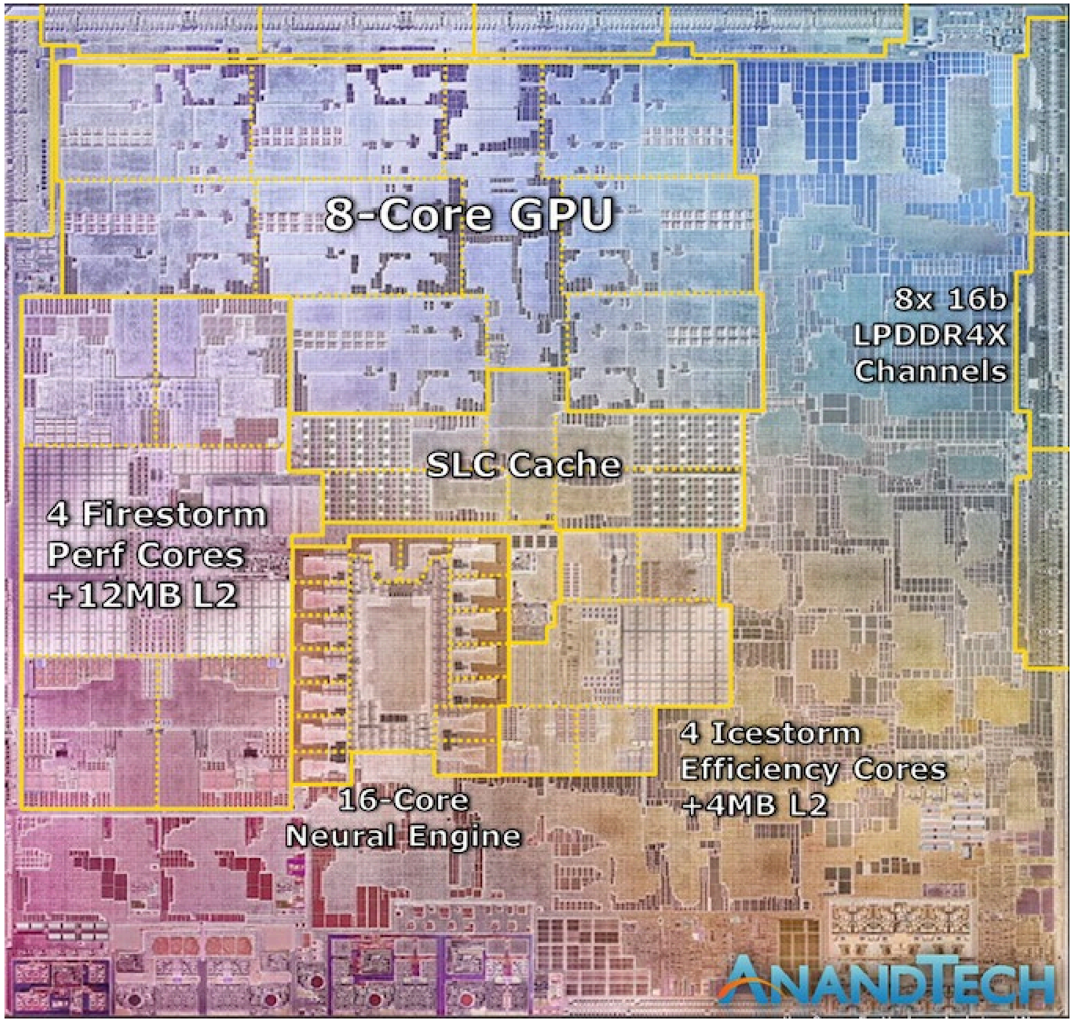
18-742:
Computer Architecture & Systems

Bingo Spatial Data Prefetcher

Prof. Phillip Gibbons

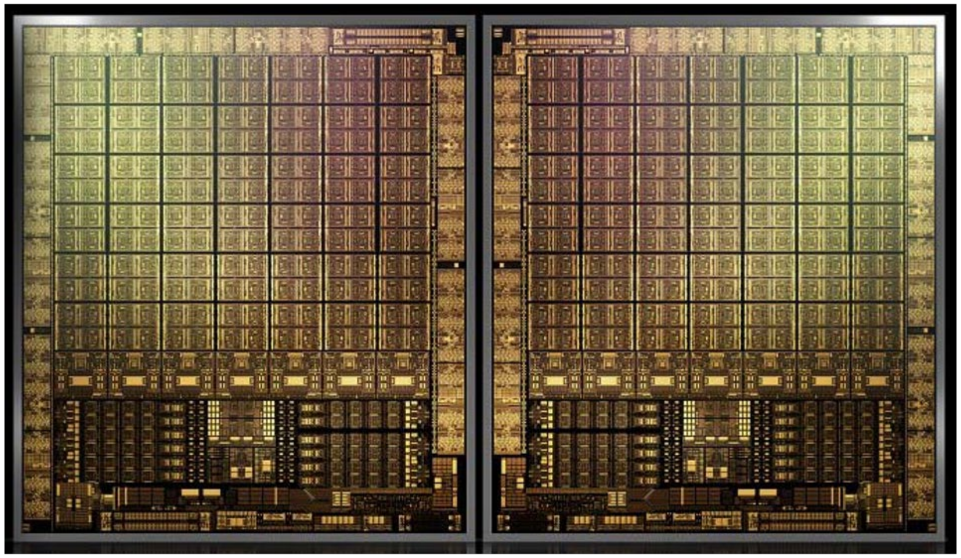
Spring 2025, Lecture 6

Memory Hierarchy Is Getting Deeper and Larger



Apple M1
2021

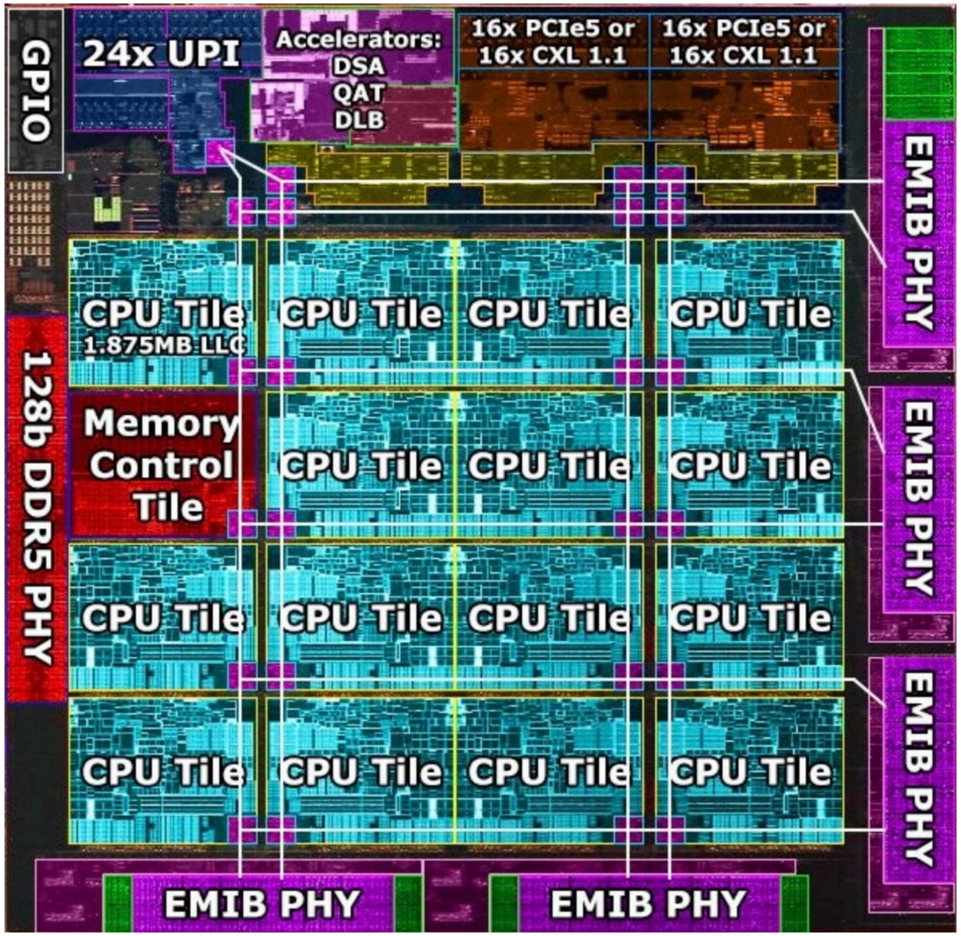
Memory Hierarchy Is Getting Deeper and Larger



Nvidia Hopper
2022

L1/Scratchpad:
256KB
L2: 60MB

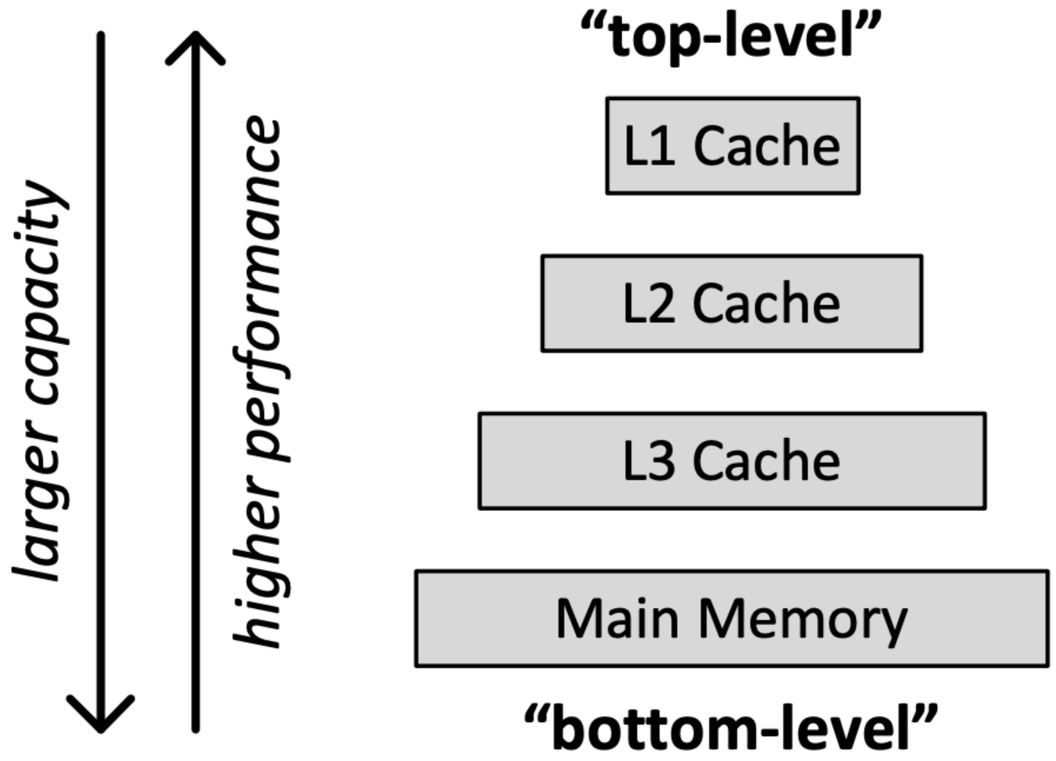
Memory Hierarchy Is Getting Deeper and Larger



Intel Sapphire Rapids
2023

L1: 80KB
L2: 2MB per core
L3: 112.5MB
L4: 64GB

Memory Hierarchy Aims to Achieve the Best of Both Worlds

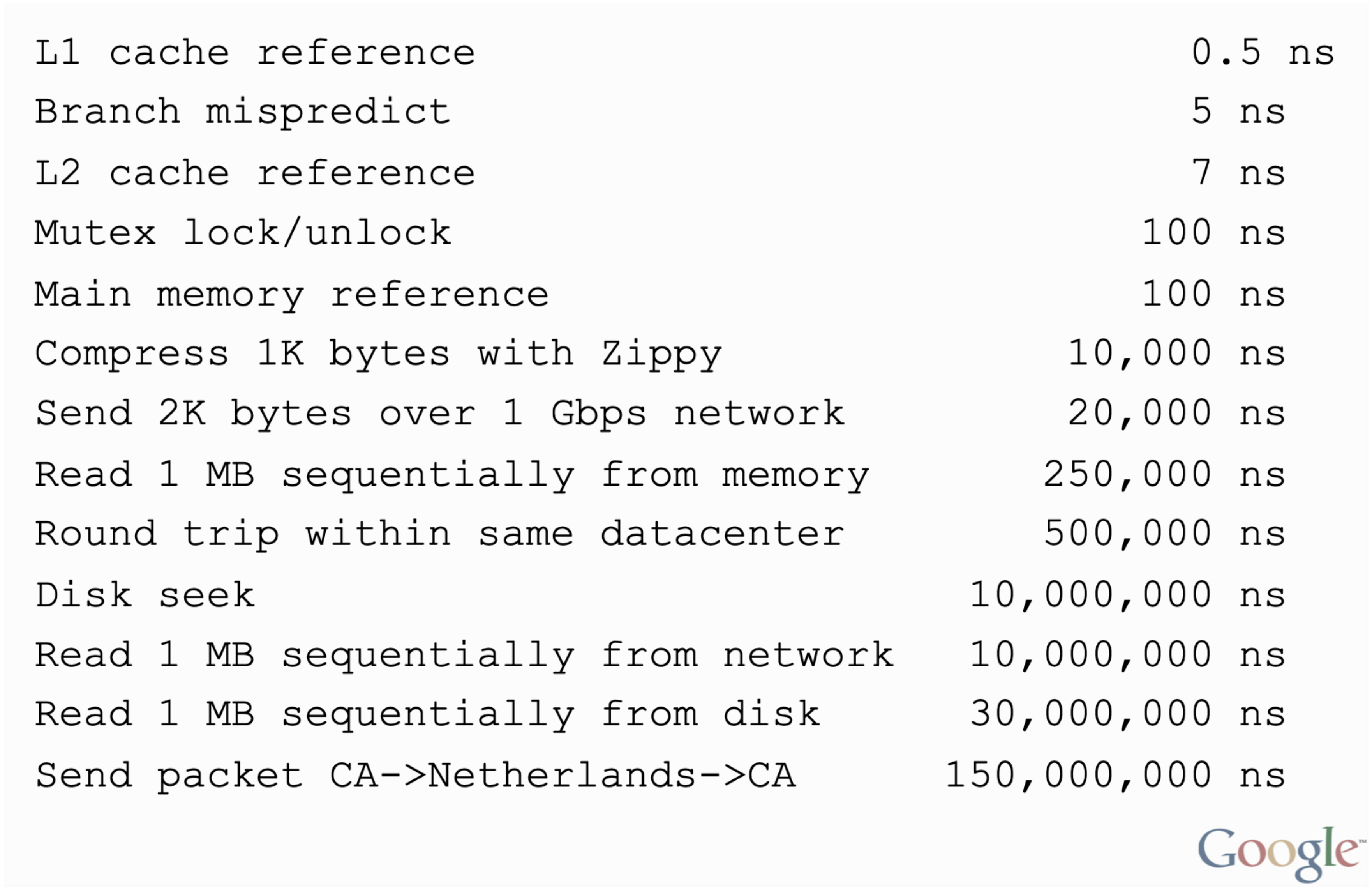


Numbers Everyone Should Know [2009]

```
L1 cache reference
Branch mispredict
L2 cache reference
Mutex lock/unlock
Main memory reference
Compress 1K bytes wit
Send 2K bytes over 1
Read 1 MB sequentiall
Round trip within sam
Disk seek
Read 1 MB sequentiall
Read 1 MB sequentiall
Send packet CA->Nethe
```

Google

Numbers Everyone Should Know [2009]



L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Numbers Everyone Should Know [2009]

L1 cache reference	0.5 ns
L2 cache reference	7 ns
Main memory reference	100 ns
Read 1 MB sequentially from memory	250,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns

Prefetching

- **Bring data from slow memory to fast memory ahead of the time**

- **Involves:**
  - Learning the access pattern
  - Predicting future accesses
  - Fetching data ahead of the time
  - Placing data into fast memory

Hardware Prefetching

- **Rely on <u>common</u> memory access patterns**
  - Stride
  - Temporal
  - Spatial

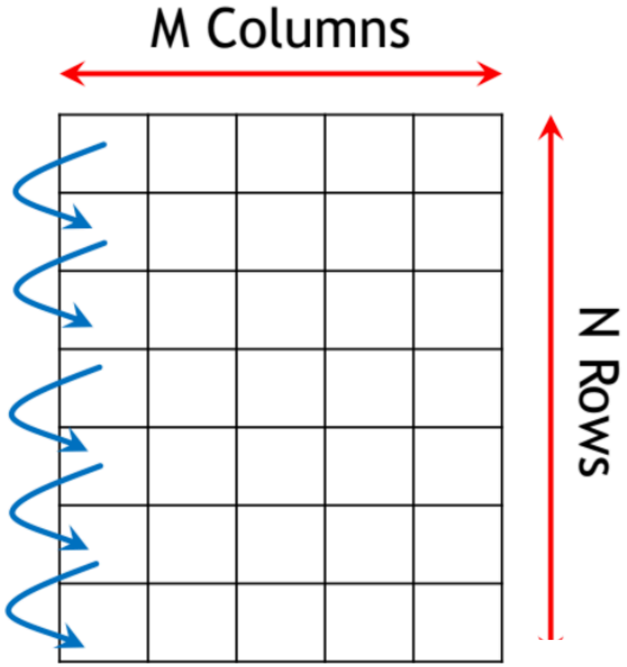
[Stride patterns are a special case of spatial patterns]

Strided Patterns

- Example: 2D matrix traversal

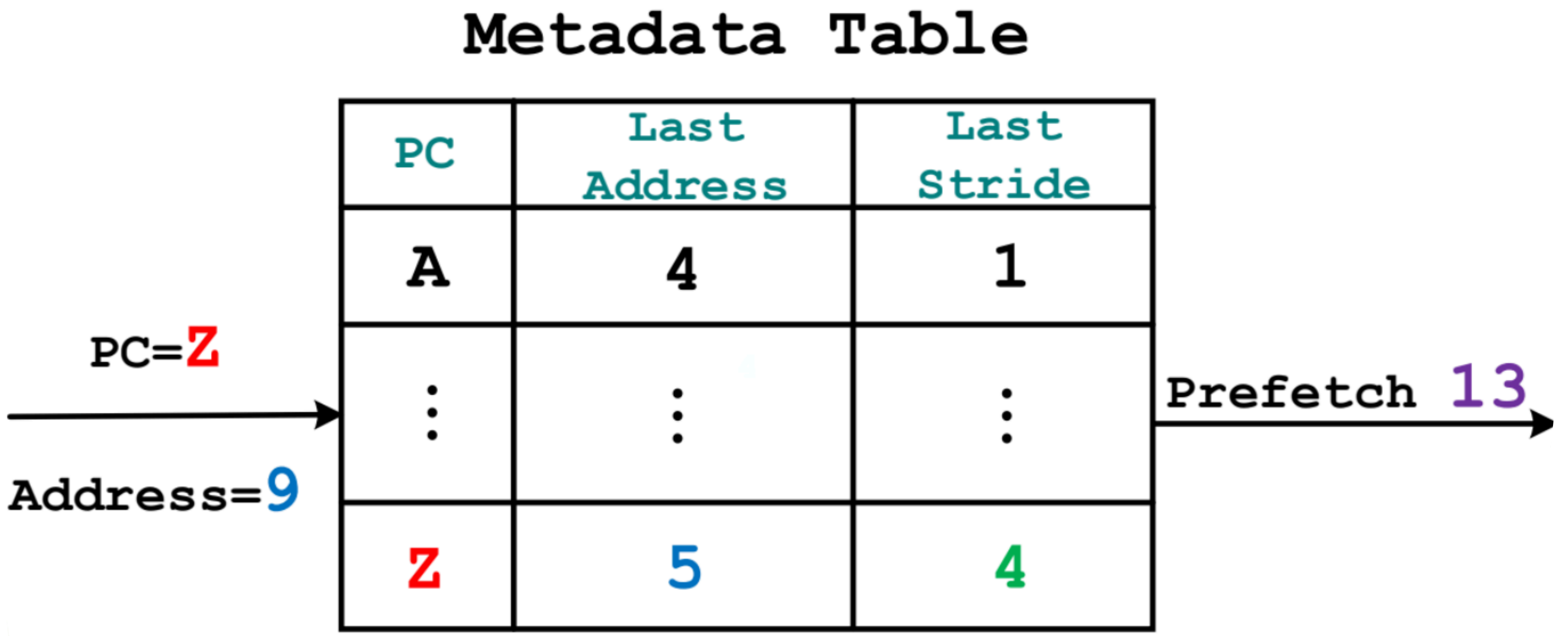
```
for i = 0 : N
   load A[i][0]
```

```
A, A+M, A+2M, A+3M, ...
```



- Idea: Learn the strides & prefetch next addresses

Instruction-Based Stride Prefetching



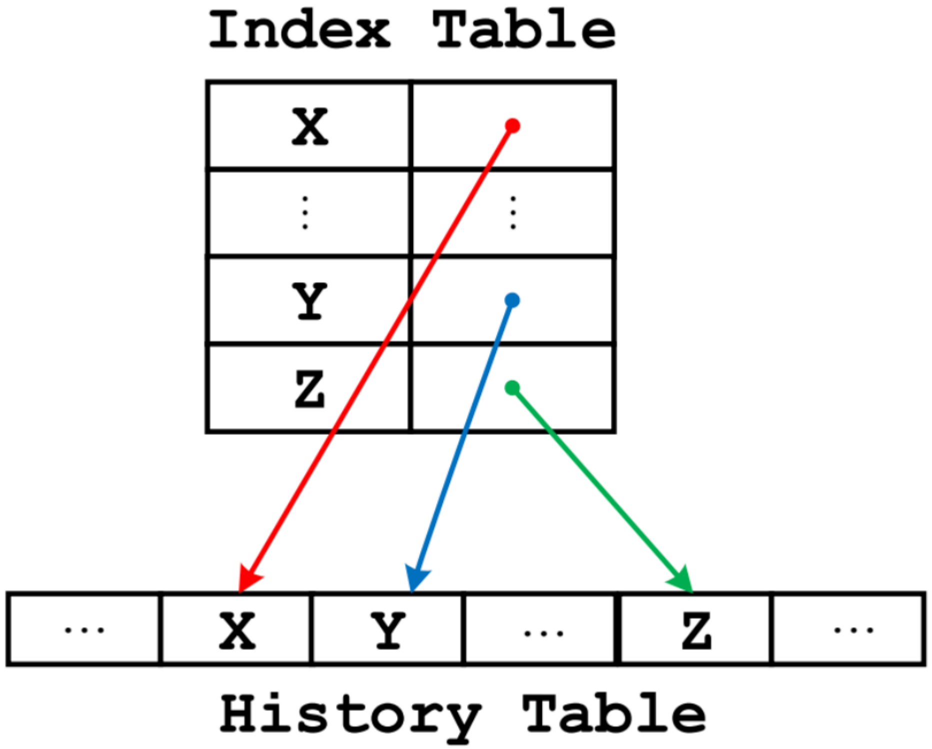
PC	Last Address	Last Stride
A	4	1
⋮	⋮	⋮
Z	5	4

Temporal Patterns / Prefetchers

- Example: Loops with accesses to fixed addresses

```
for i = 0 : N
    load A
    load B
    load C
    f(A, B, C)
```

A, B, C, ..., A, B, C, ...



Spatial Patterns / Prefetchers

- Example: Loops with accesses to offset fields

```
for obj in objects
   load obj.x
   load obj.y
   load obj.z
   f(obj.x, obj.y, obj.z)
```

```
A+X, A+Y, A+Z, ..., B+X, B+Y, B+Z, ...
```

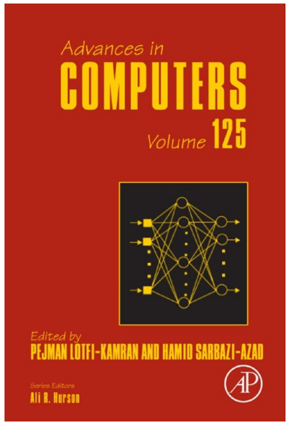
Metadata Table

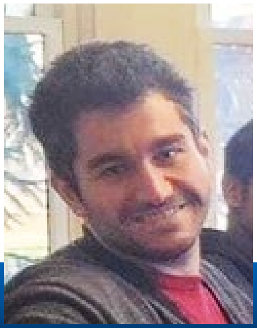
Event	Pattern
PC_0	01110…01010
PC_1	01010…00011
⋮	⋮

"Bingo Spatial Data Prefetcher"

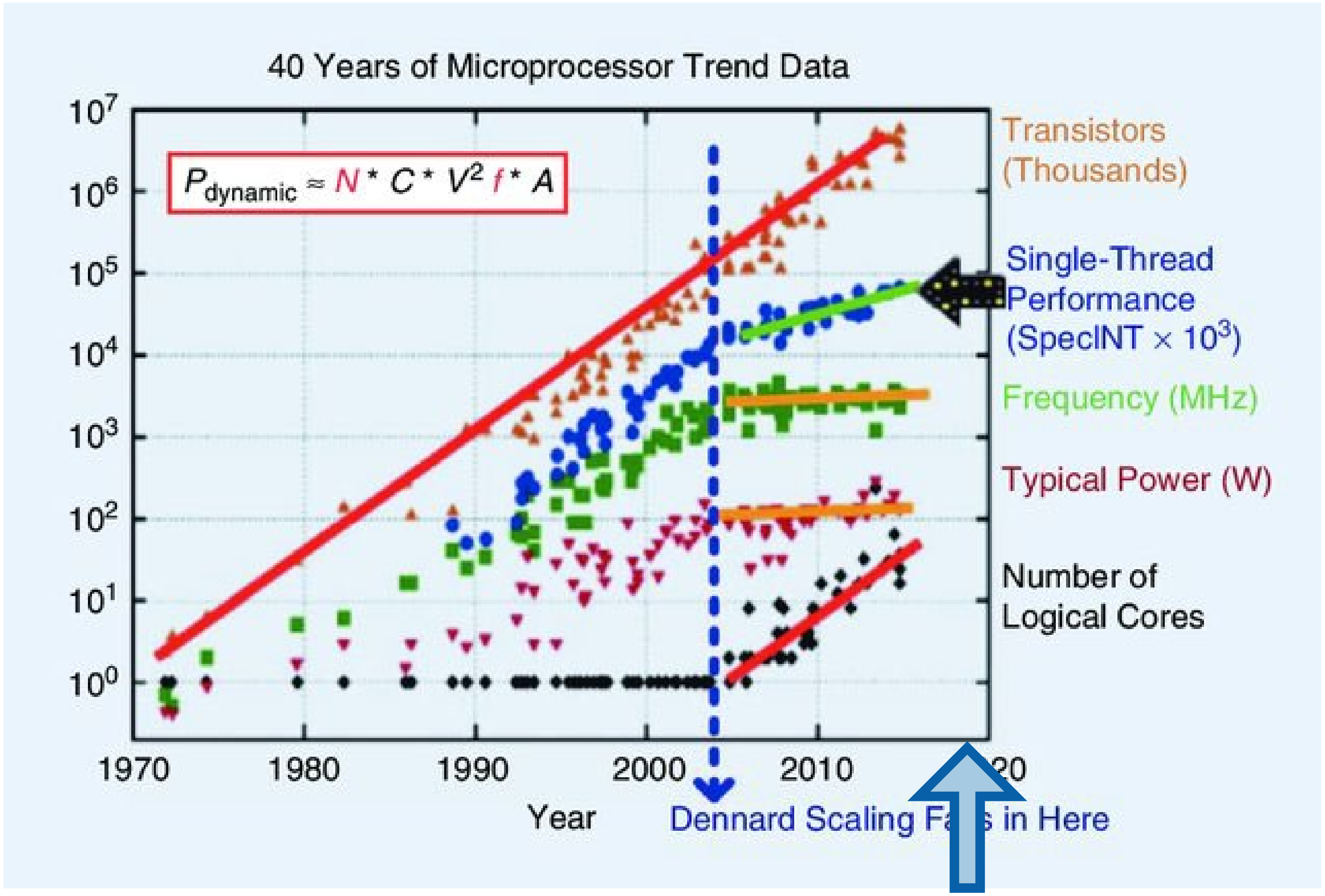
Mohammad Bakhshalipour, Mehran Shakerinava, Pejman Lotfi-Kamran, Hamid Sarbazi-Azad 2019

- **Mohammad: Sharif U of Tech MS, now Nvidia**
  - ➢ CMU PhD (won best ECE dissertation 2024 despite Advisor)
- **Mehran: Sharif U of Tech BS, now McGill PhD in ML**
  - ➢ 1st Place Iran's National Master's Entrance Exam
- **Pejman: IPM Iran prof**
  - ➢ Head of School of Computer Science at IPM
- **Hamid: Sharif U of Tech prof**
  - ➢ Book: Data Prefetching Techniques in Computer Systems





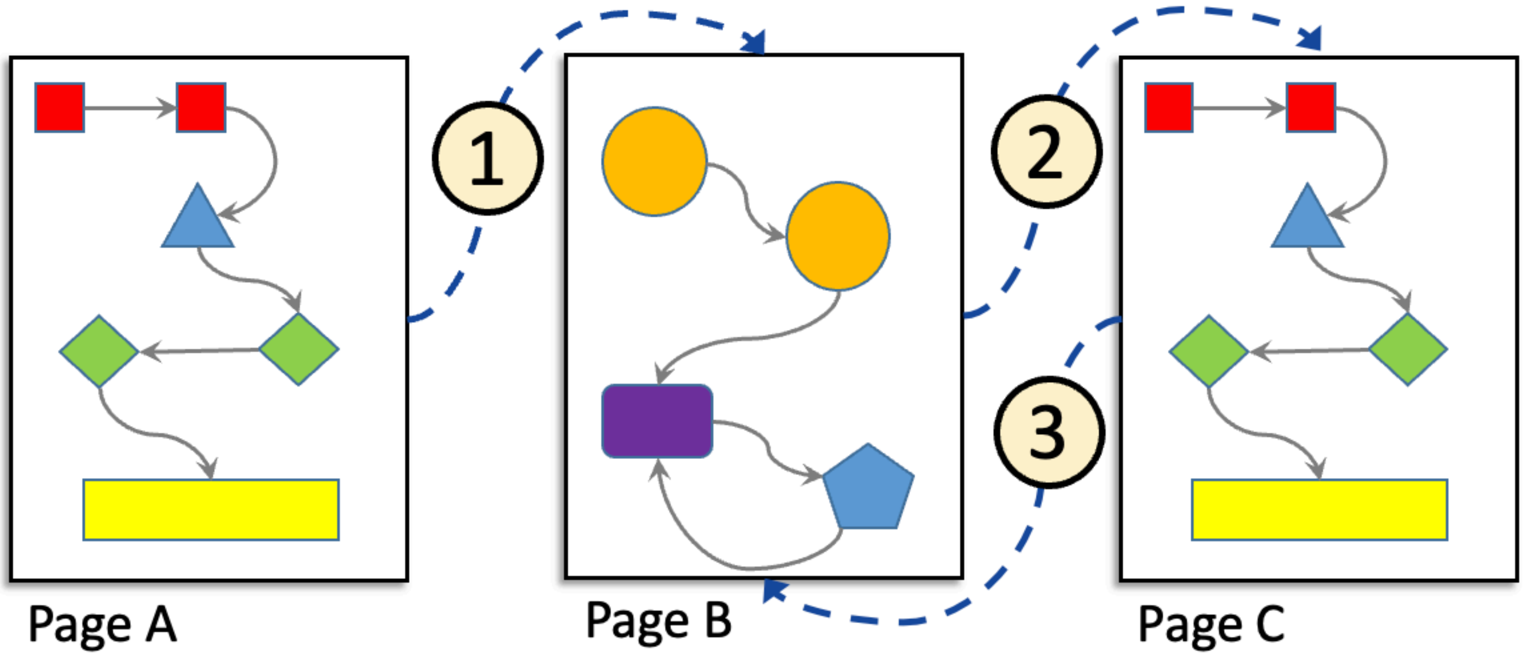
Moore’s Law w/o Dennard Scaling



We are here

Spatial Data Correlation: A Deeper Look

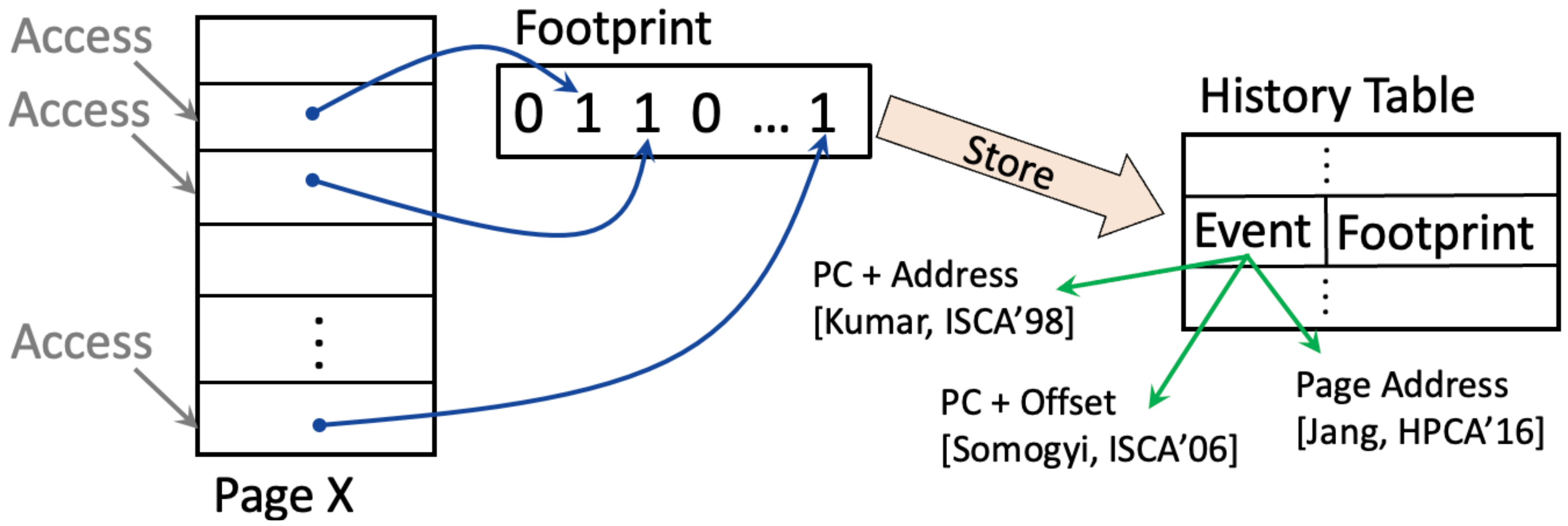
- **Access patterns repeat over memory pages**
  - Because data objects have fixed and regular layout



Spatial Data Prefetching

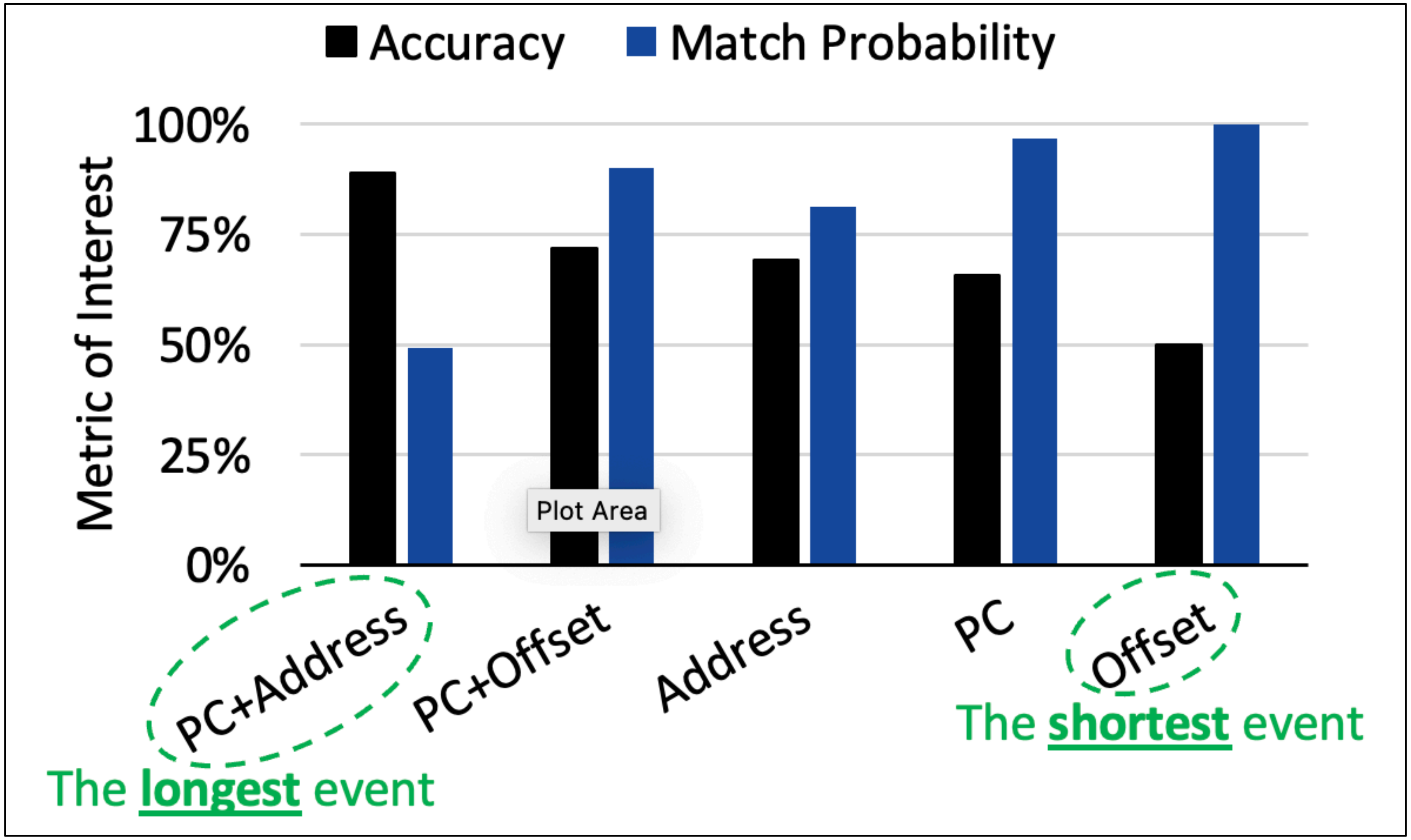
Before Bingo

- **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 **trigger access**


Access
Access
Access
Page X
Footprint
0 1 1 0 … 1
Store
History Table
Event
Footprint
PC + Address [Kumar, ISCA'98]
PC + Offset [Somogyi, ISCA'06]
Page Address [Jang, HPCA'16]

But: One event is not accurate enough

What Is The Best Event?



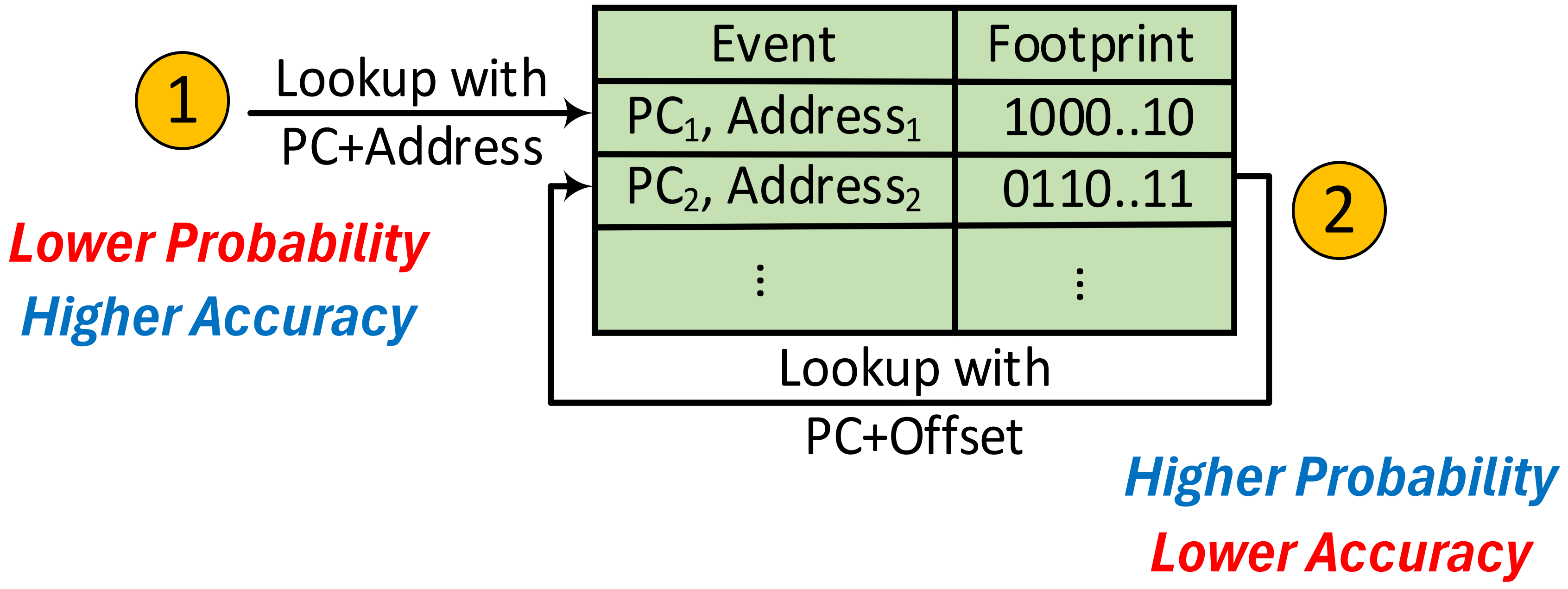
No single event has all good characteristics

→ Use multiple events

Bingo Overview

- Correlates each footprint with **multiple** events
  - Two events is best (PC+Address, PC+Offset)

- Consolidates metadata information to remove redundancies
  - Recursive prediction based on a single history table



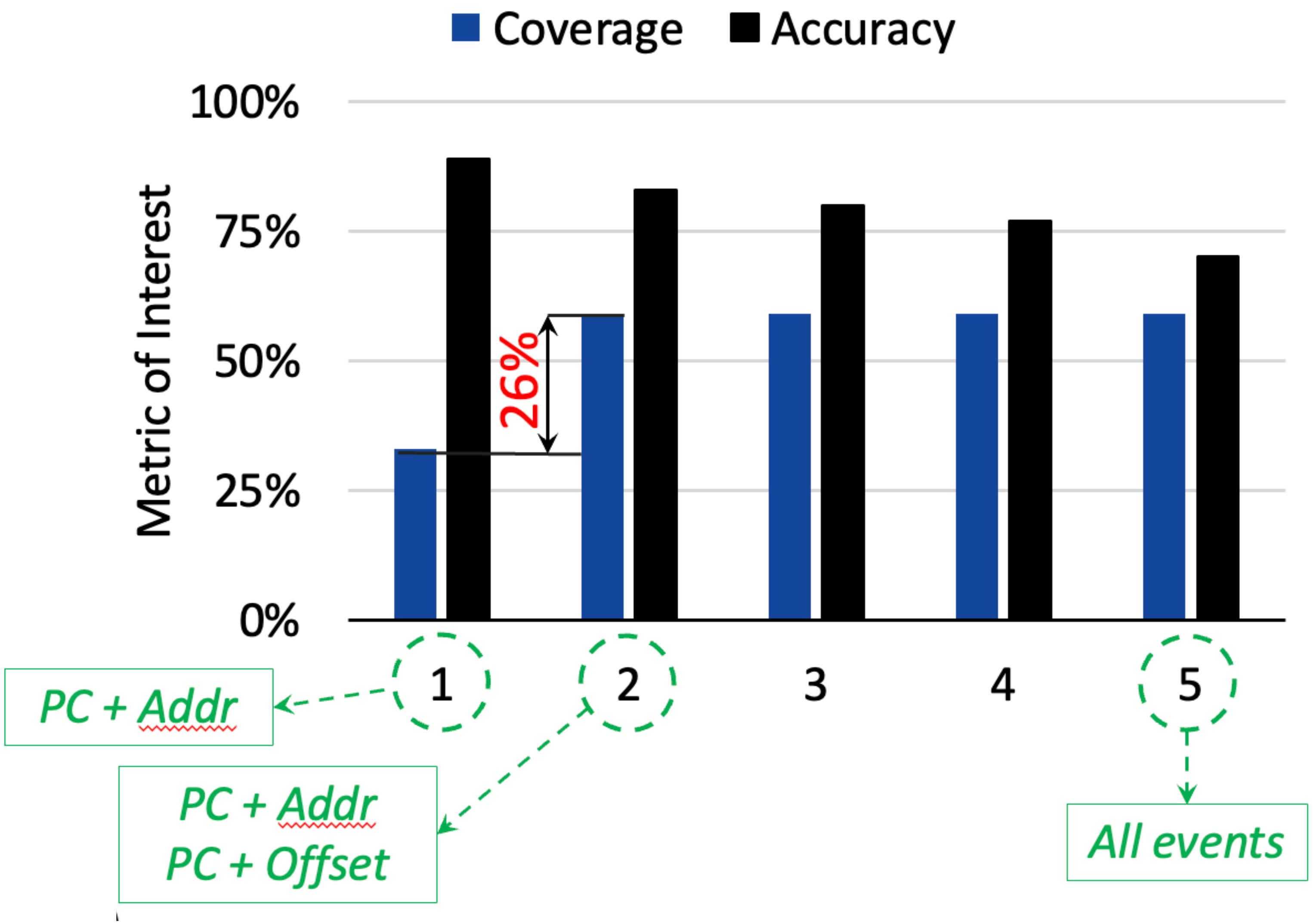
Discussion: Summary Question #1

What Did the Paper Get Right?

State the 3 most important things the paper says.

These could be some combination of the motivations, observations, interesting parts of the design, or clever parts of the implementation.

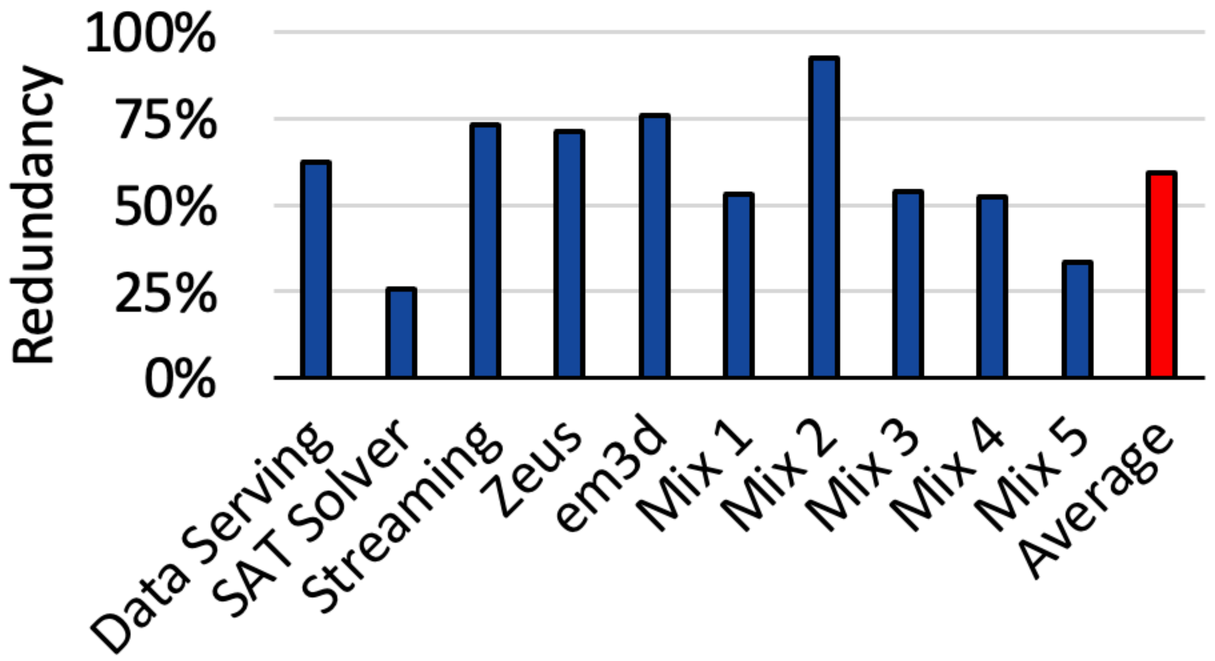
How Many Events?



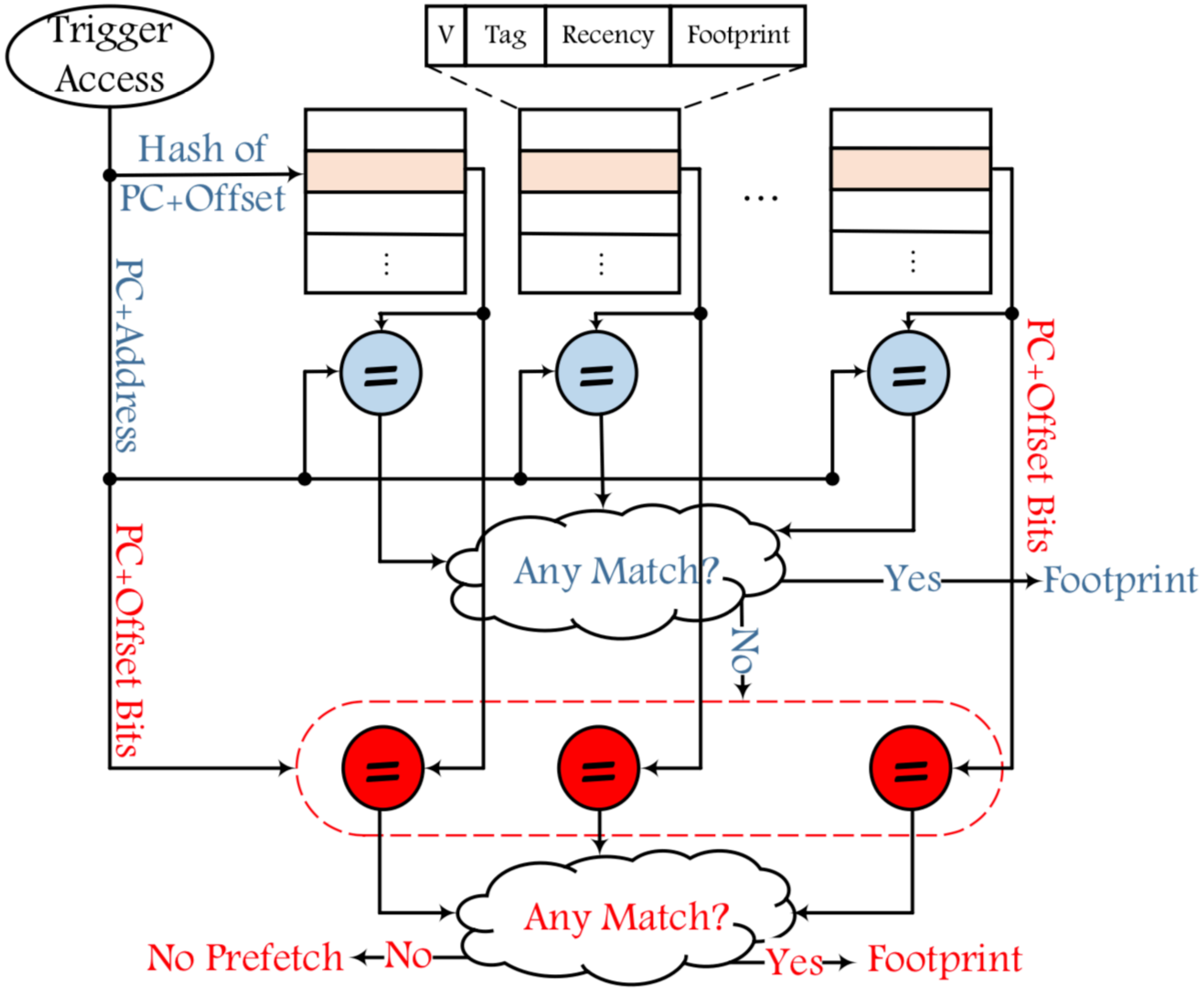
Two events suffice

Two Tables: Significant Redundancy

- Redundancy = Percentage of cases where both tables offer the same prediction (i.e., the same footprint)

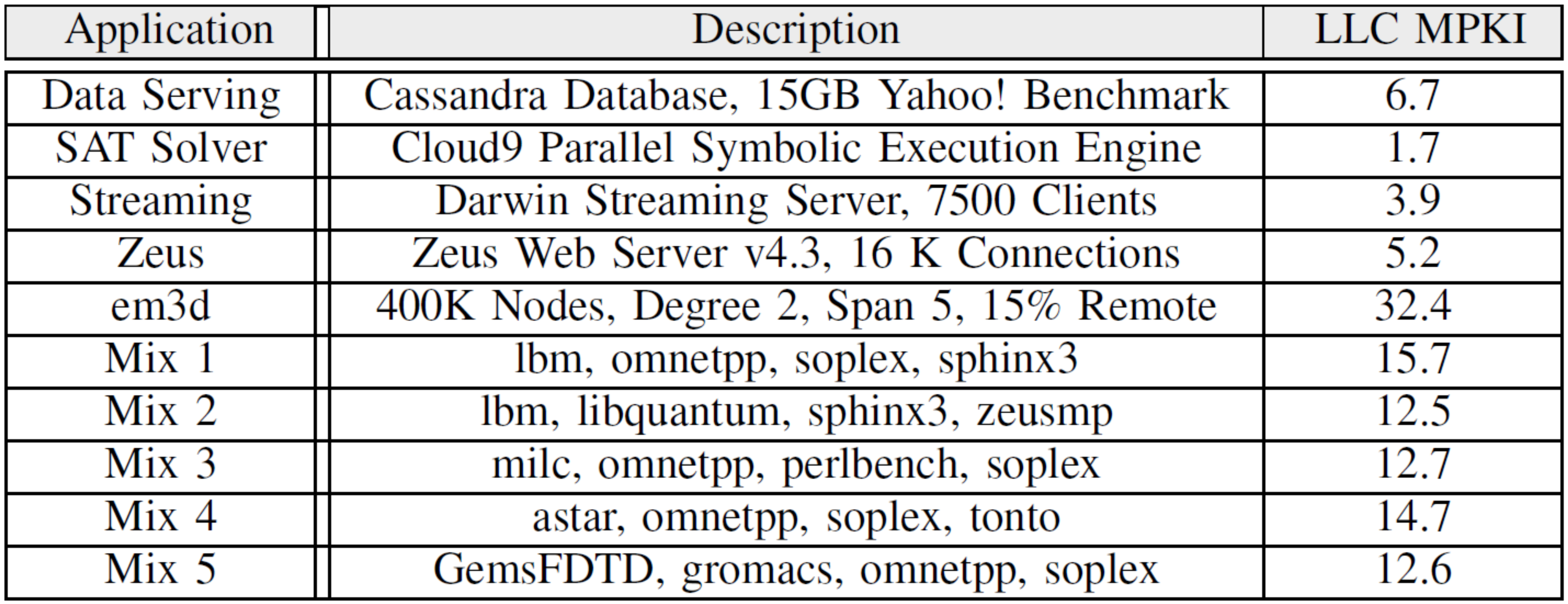


Bingo: Hardware Realization



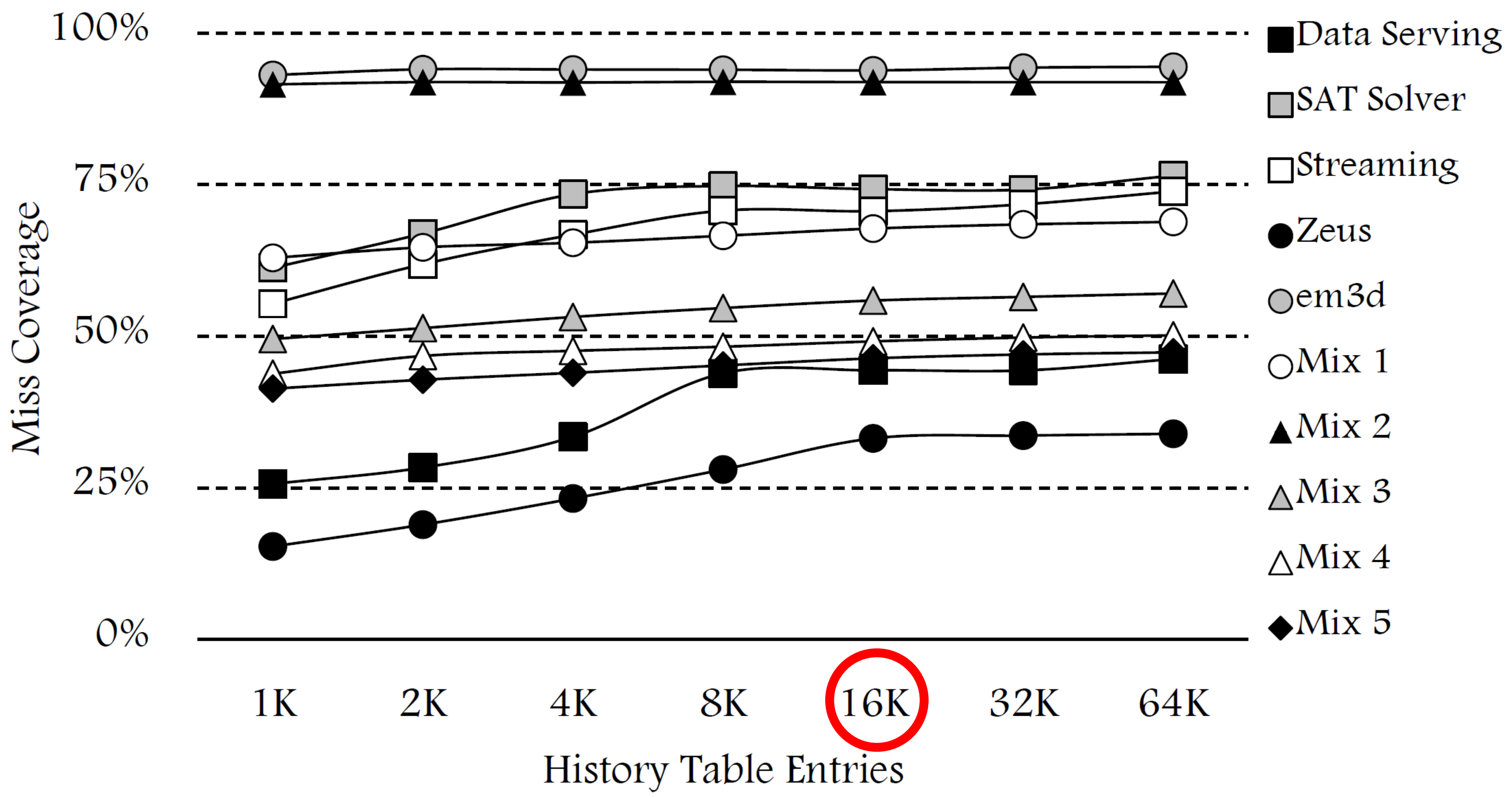
Multiple matches? Prefetch blocks in ≥20% of matching entries

Applications Studied



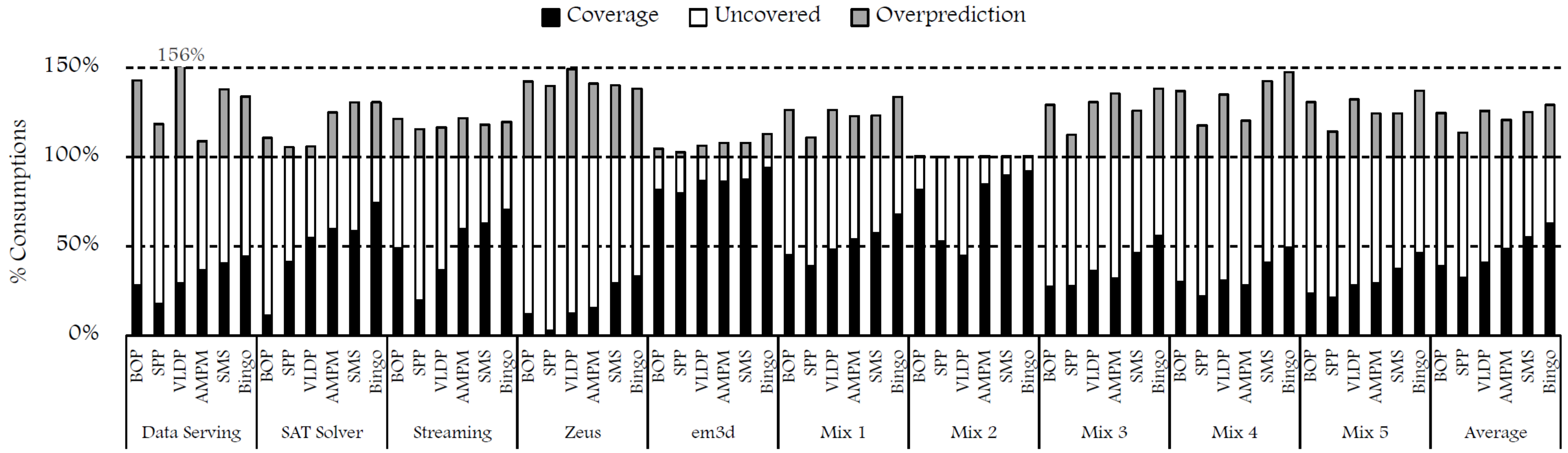
Application	Description	LLC MPKI
Data Serving	Cassandra Database, 15GB Yahoo! Benchmark	6.7
SAT Solver	Cloud9 Parallel Symbolic Execution Engine	1.7
Streaming	Darwin Streaming Server, 7500 Clients	3.9
Zeus	Zeus Web Server v4.3, 16 K Connections	5.2
em3d	400K Nodes, Degree 2, Span 5, 15% Remote	32.4
Mix 1	lbm, omnetpp, soplex, sphinx3	15.7
Mix 2	lbm, libquantum, sphinx3, zeusmp	12.5
Mix 3	milc, omnetpp, perlbench, soplex	12.7
Mix 4	astar, omnetpp, soplex, tonto	14.7
Mix 5	GemsFDTD, gromacs, omnetpp, soplex	12.6

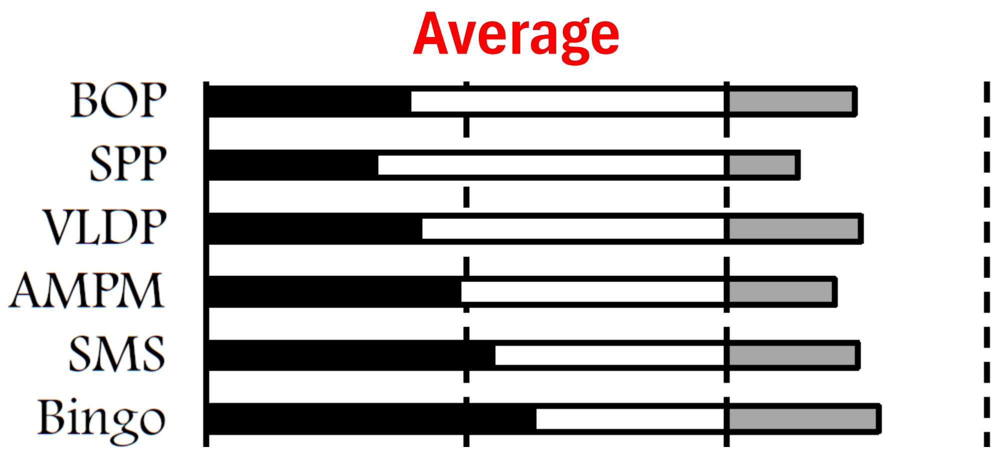
Bingo: Miss Coverage vs. Table Size



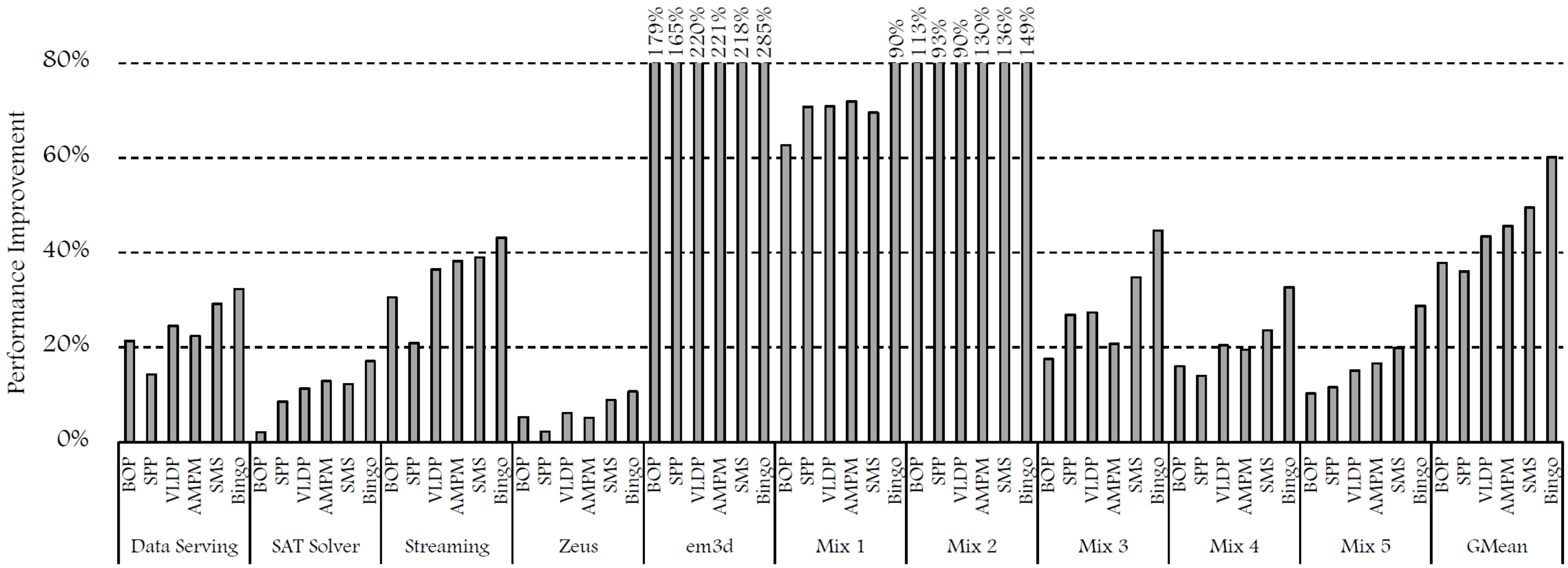
16K entries = 119 KB = 6% of LLC Area

Coverage & Overprediction Comparison

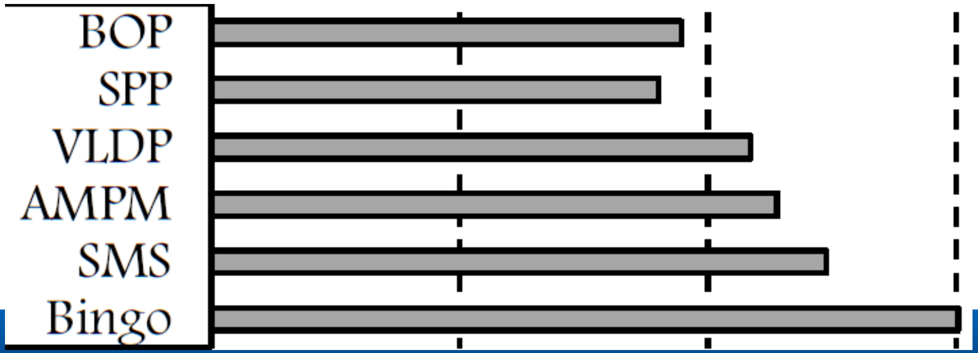




Application Improvements



GeoMean



Discussion: Summary Question #2

What Did the Paper Get Wrong?

Describe the paper's single most glaring deficiency.

Every paper has some fault. Perhaps an experiment was poorly designed or the main idea had a narrow scope or applicability.

Data Prefetching Today

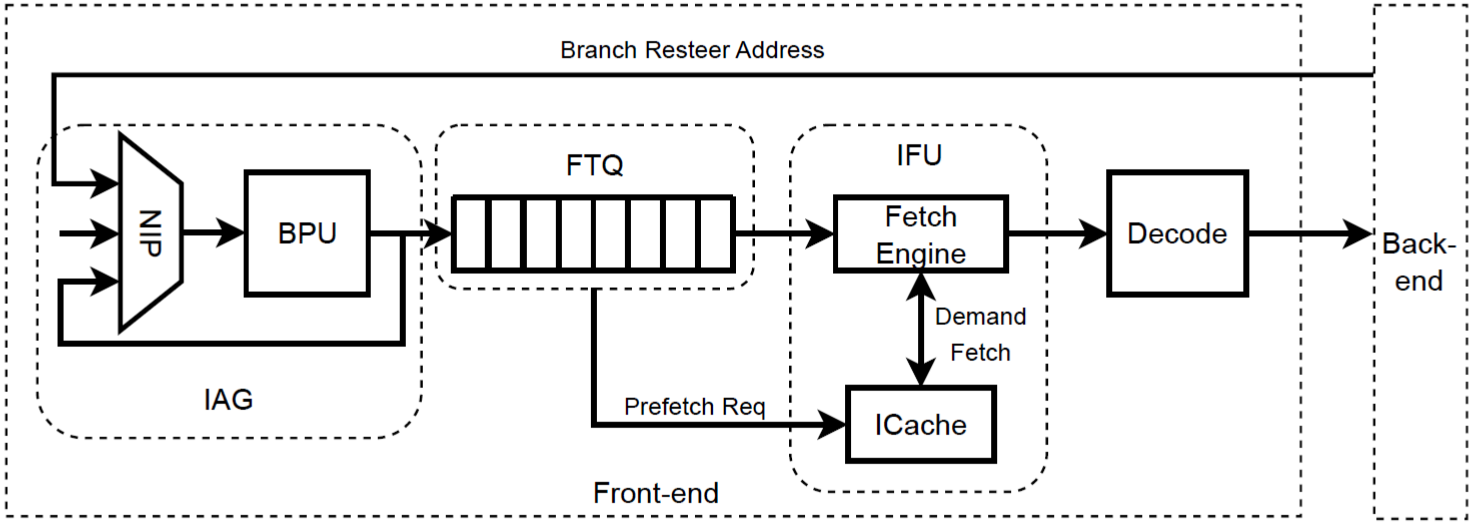
- Active research area for more than 40 years
- Software and hardware prefetching
- Every high-performance processor has multiple data prefetchers
- Newer memory technologies → New challenges & opportunities for prefetchers
- Machine learning for prefetching?

“PDIP: Priority Directed Instruction Prefetching”

Bhargav Reddy Godala, Sankara Prasad Ramesh, Gilles A. Pokam, Jared Stark, Andre Seznec, Dean Tullsen, David I. August 2024

- Modern server workloads $\Rightarrow$ High I-Cache capacity misses

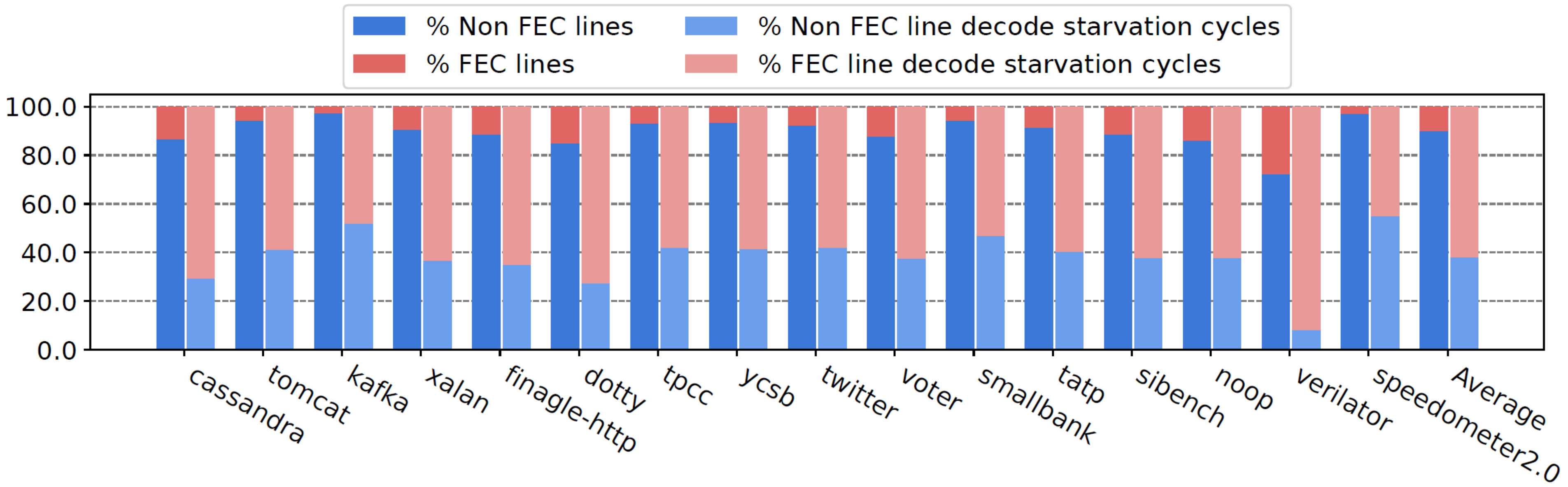
- FDIP* used to tolerate I-Cache misses, but significant stalls remain



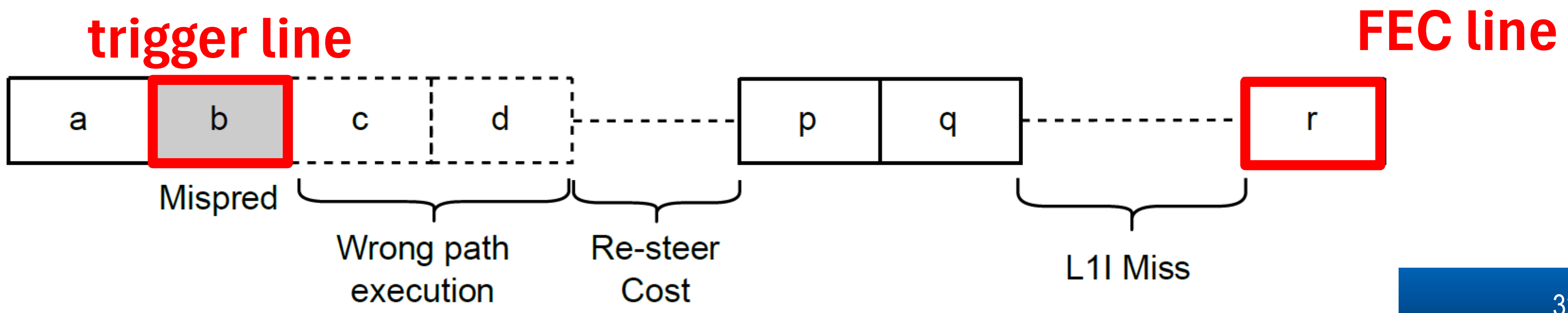
- PDIP goal: prefetch instructions iff FDIP can’t hide the miss

*Fetch Directed Instruction Prefetching

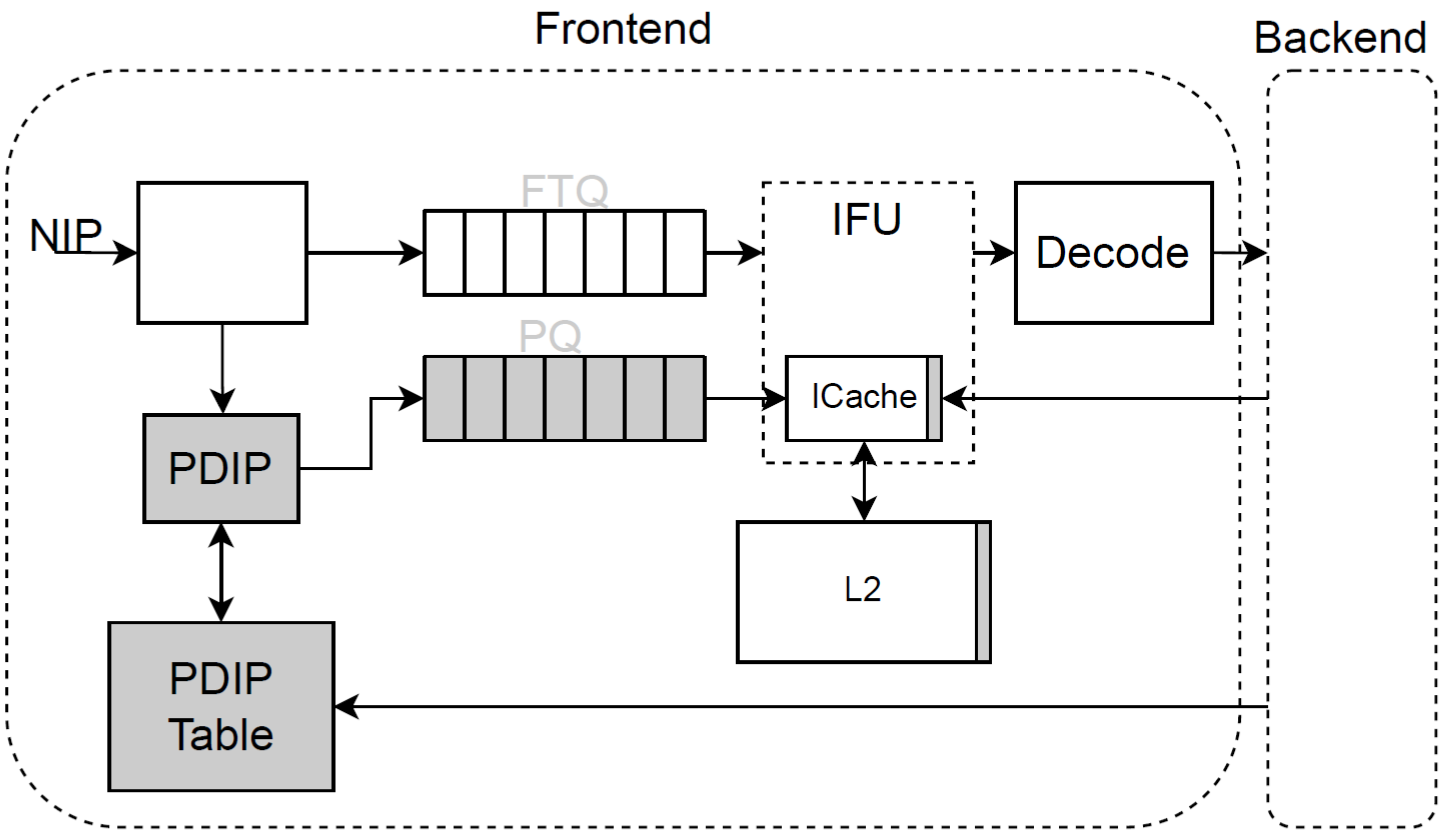
Front End Critical (FEC) Lines



- **10% of lines cause 62% of decode starvation cycles**
- **PDIP goal: prefetch these FEC lines**

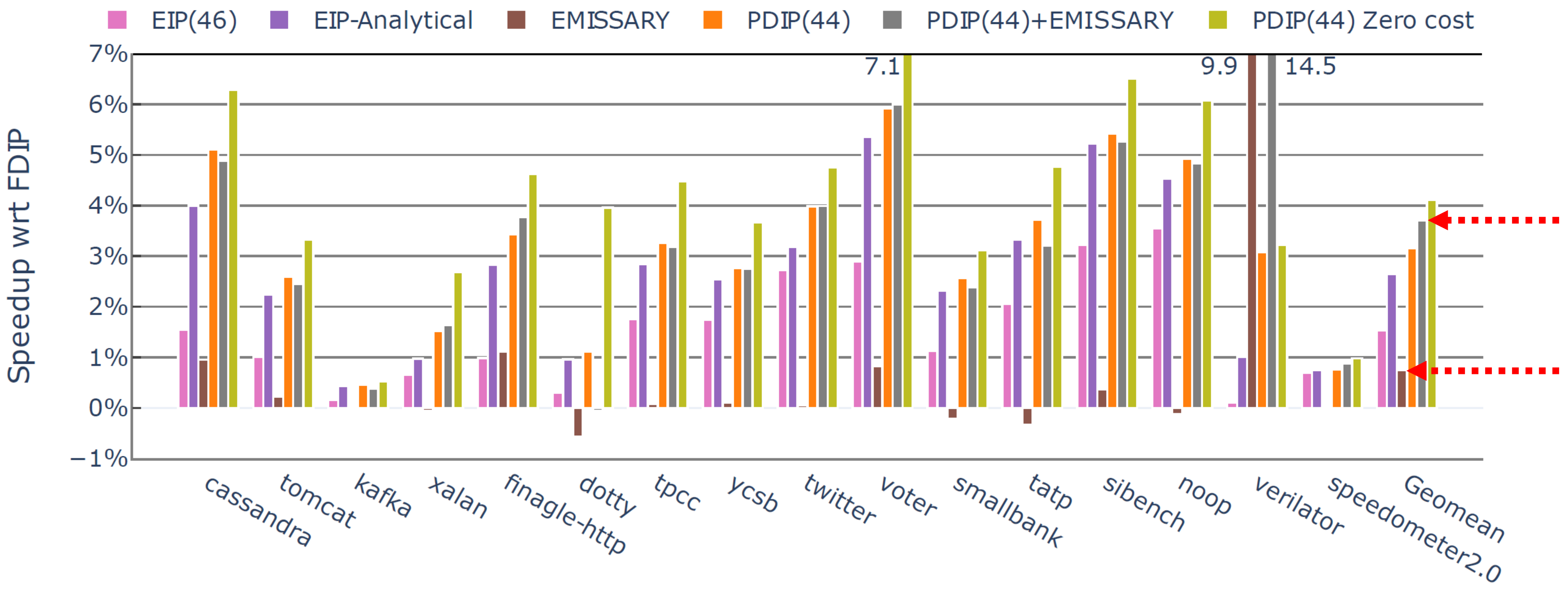


PIDP Pipeline & Optimizations



- Prioritize demand misses over prefetches
- Insert FECs into PDIP with prob ¼
- Fetch up to 4 cache blocks, based on mask in PDIP Table entry

PDIP+Emissary:
3.7% Speedup over Emissary



***Emissary: Prioritizes retaining FECs on I-Cache evictions**

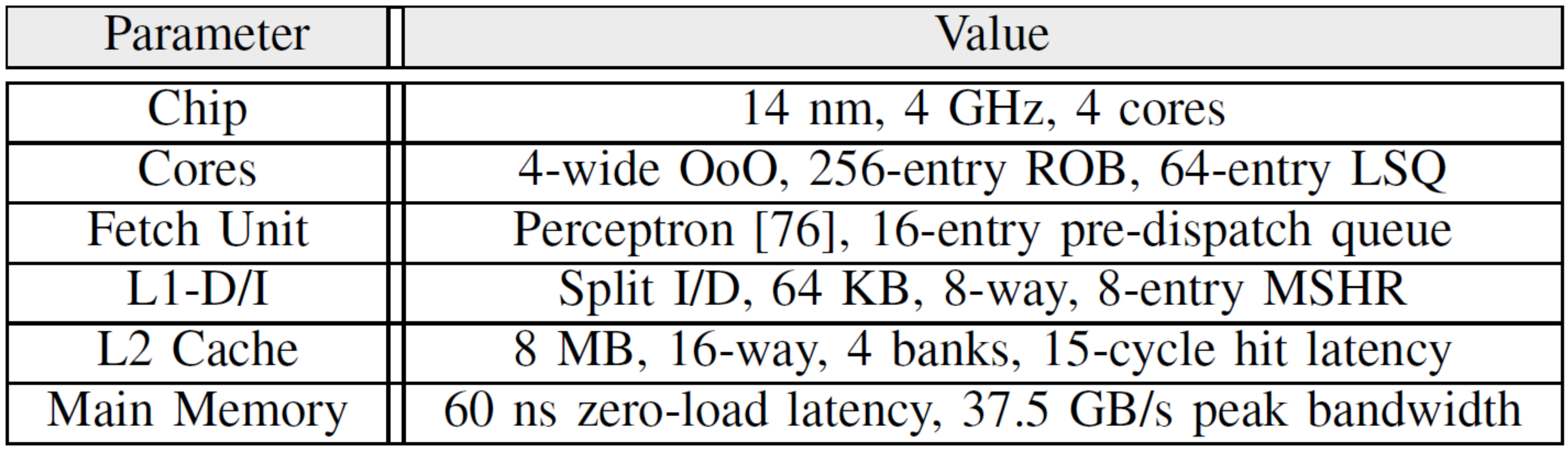
To Read for Wednesday

"Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors"

Onur Mutlu, Jared Stark, Chris Wilkerson, Yale N. Patt 2003

Modeled Architecture

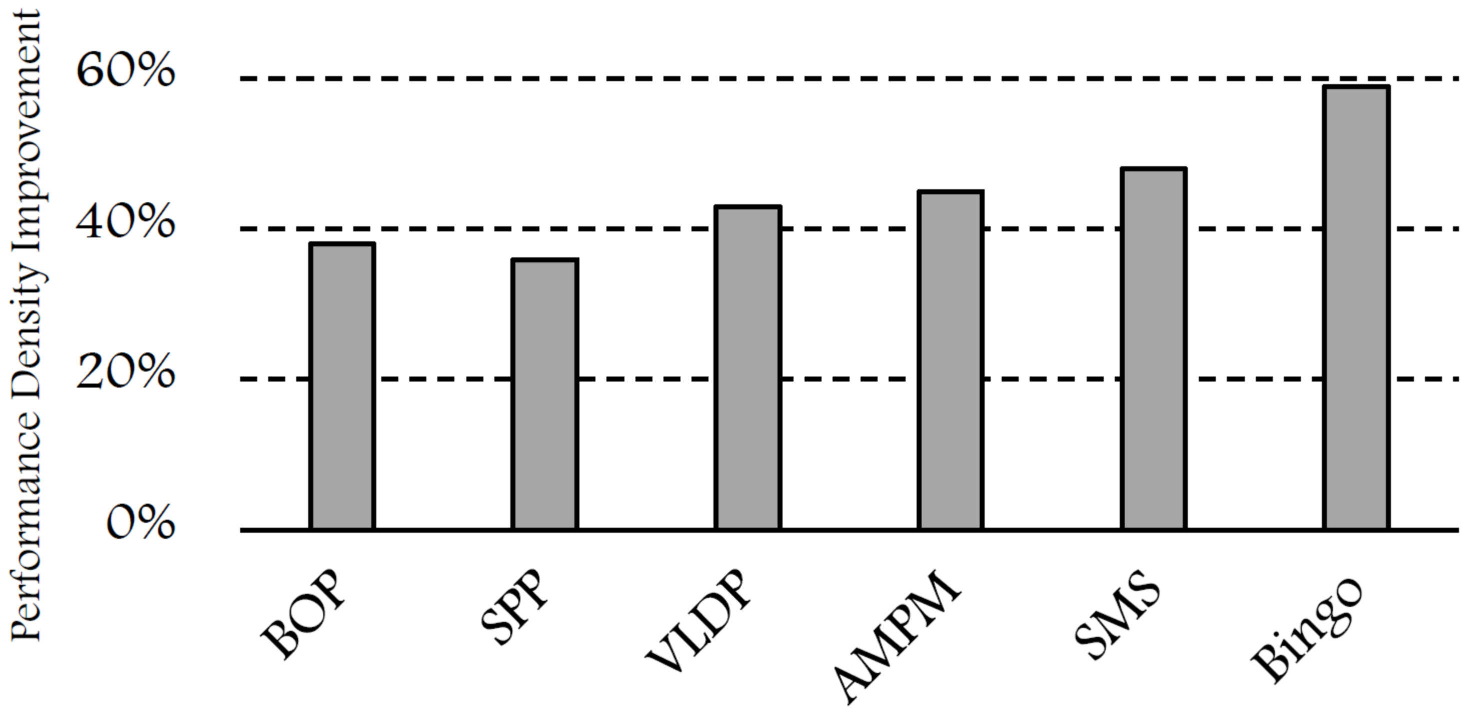
Table I
EVALUATION PARAMETERS.



Parameter	Value
Chip	14 nm, 4 GHz, 4 cores
Cores	4-wide OoO, 256-entry ROB, 64-entry LSQ
Fetch Unit	Perceptron [76], 16-entry pre-dispatch queue
L1-D/I	Split I/D, 64 KB, 8-way, 8-entry MSHR
L2 Cache	8 MB, 16-way, 4 banks, 15-cycle hit latency
Main Memory	60 ns zero-load latency, 37.5 GB/s peak bandwidth

Performance Density

Throughput pre unit area vs. No prefetcher



ISO-degree Comparison

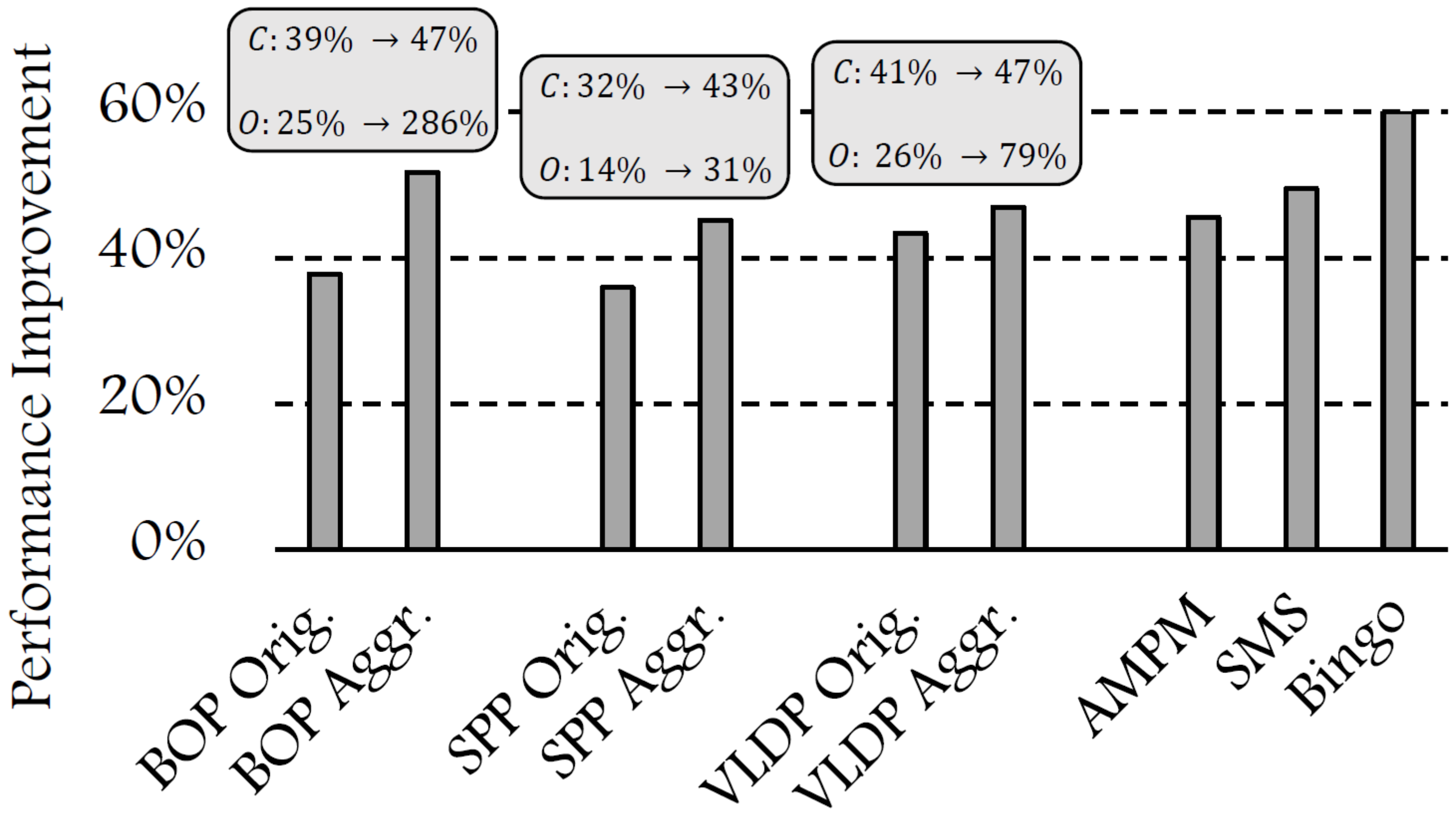


Figure 10. ISO-degree comparison of prefetching methods. 'Orig' indicates the original and so-far–evaluated version of an SHH-based prefetcher; however, 'Aggr' represents the aggressive and high-degree version. Callouts indicate how the coverage and overprediction of prefetchers vary from the original version to the aggressive version: 'C' and 'O' stand for 'Coverage' and 'Overprediction,' respectively.