

ACCORD: Associativity for DRAM Caches by Coordinating Way-Install and Way-Prediction

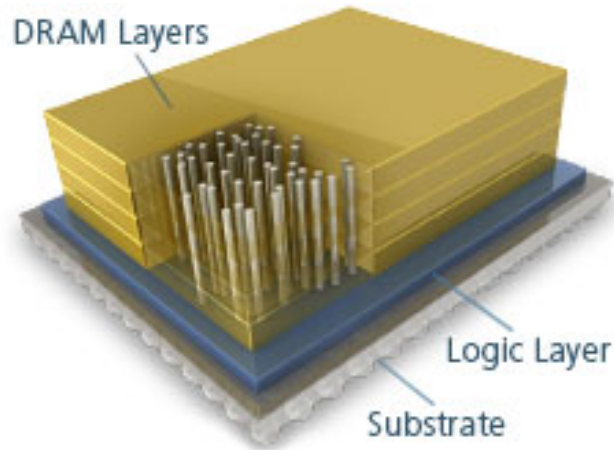
ISCA 2018

Authors: **Vinson Young** (GT)
Chiachen Chou (GT)
Aamer Jaleel (NVIDIA)
Moinuddin K. Qureshi (GT)



3D-DRAM MITIGATES BANDWIDTH WALL

Modern system packing many cores → Bandwidth Wall



3D-Stacked DRAM



4-8x Bandwidth
(of traditional memory)



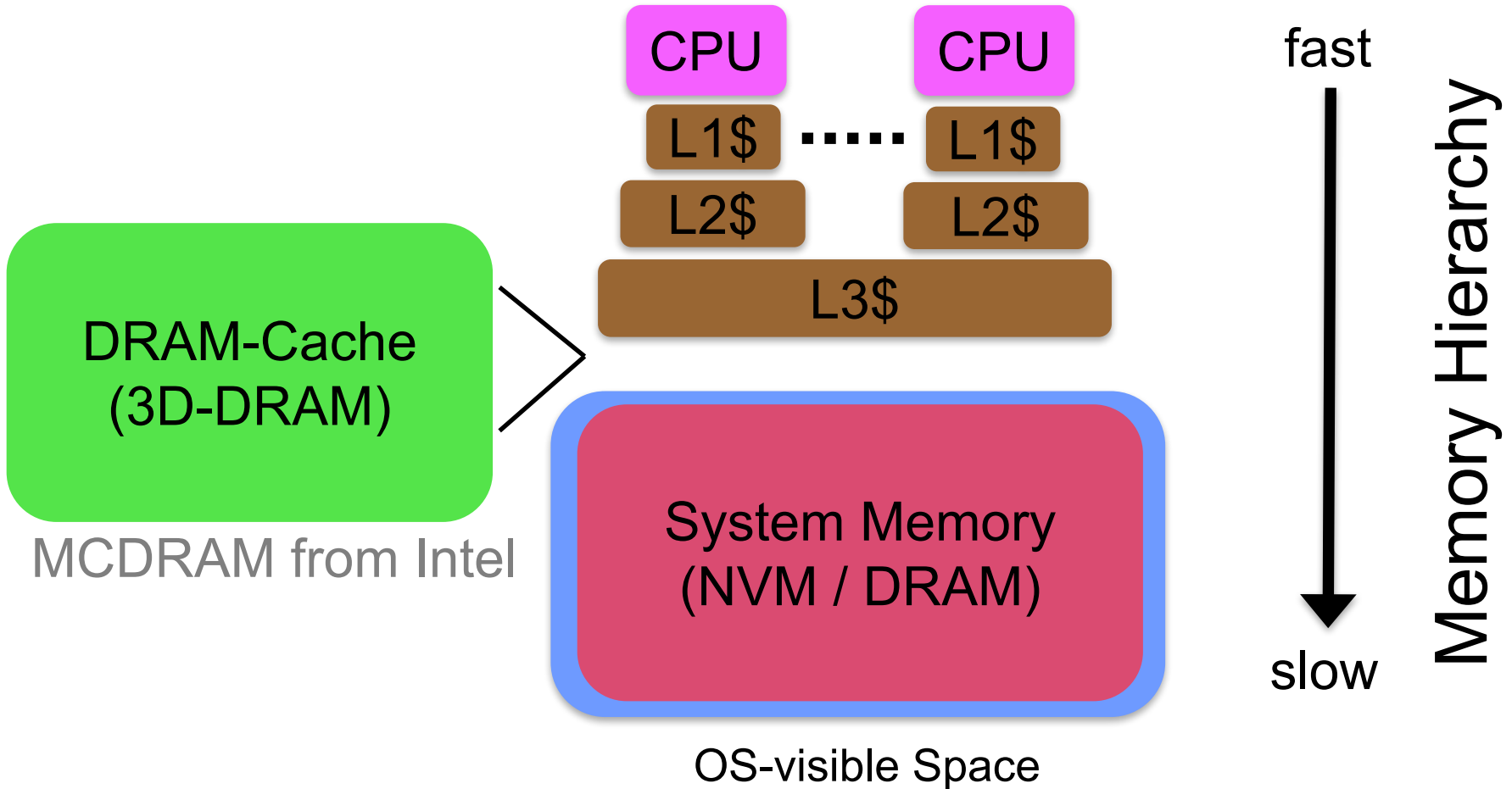
Limited Capacity



Memory

3D-DRAM + High-Capacity Memory = Hybrid Memory

USE 3D-DRAM AS A CACHE



Using 3D-DRAM as a DRAM cache, can improve memory bandwidth (and avoid OS/software change)

ARCHITECTING LARGE DRAM CACHES

Organize at *line granularity* (64B) for capacity/BW utilization

Gigascale cache needs *large tag-store* (tens of MBs)



Tags?

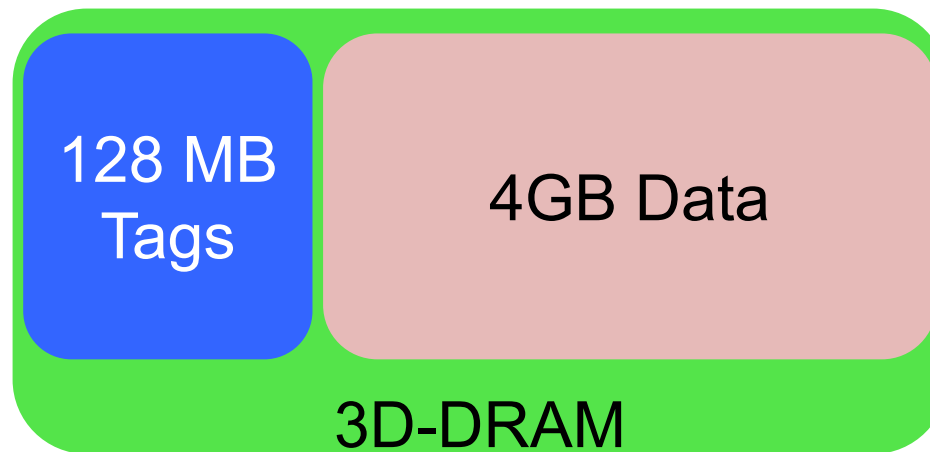


Too large for SRAM

ARCHITECTING LARGE DRAM CACHES

Organize at *line granularity* (64B) for high cache utilization

Gigascale cache needs *large tag-store* (tens of MBs)

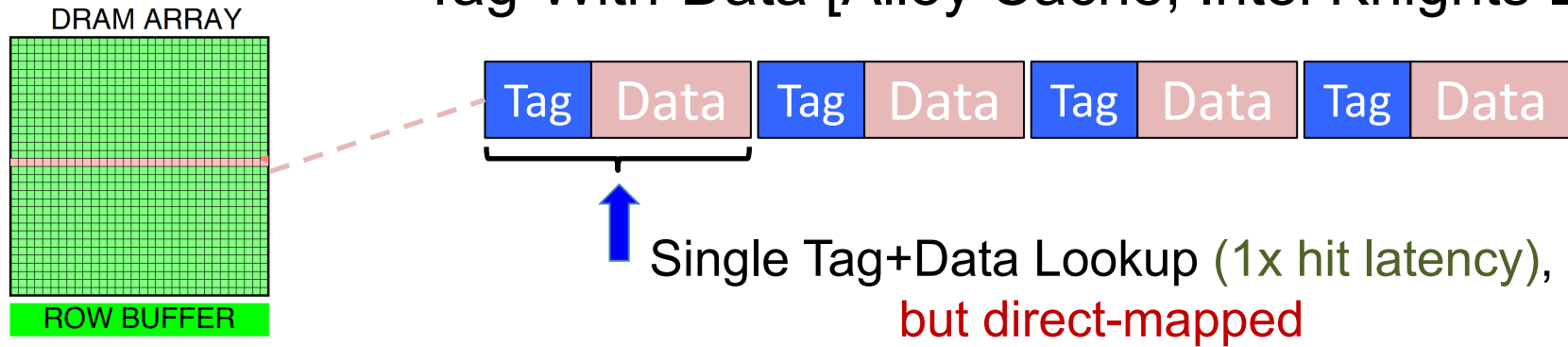


Practical designs must store **Tags in DRAM**

How to architect tag-store for low-latency tag access?

EFFICIENT TAG ORGANIZATION (KNL CACHE)

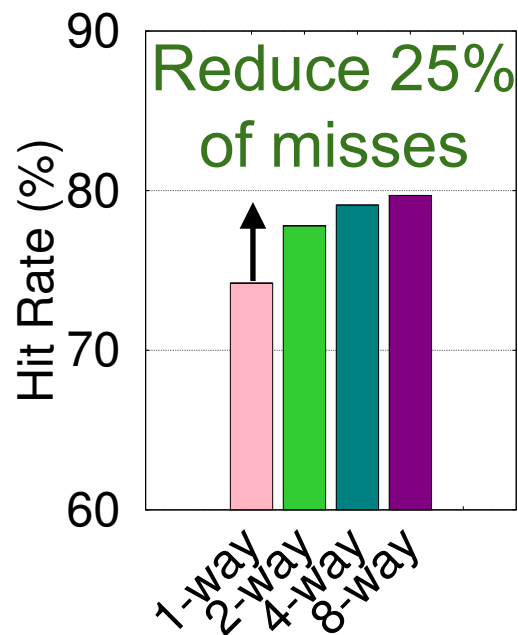
Tag-With-Data [Alloy Cache, Intel Knights Landing]



Practical designs are 64B line-size, store *Tag-With-Data*, and are *direct-mapped*, to optimize for hit-latency.

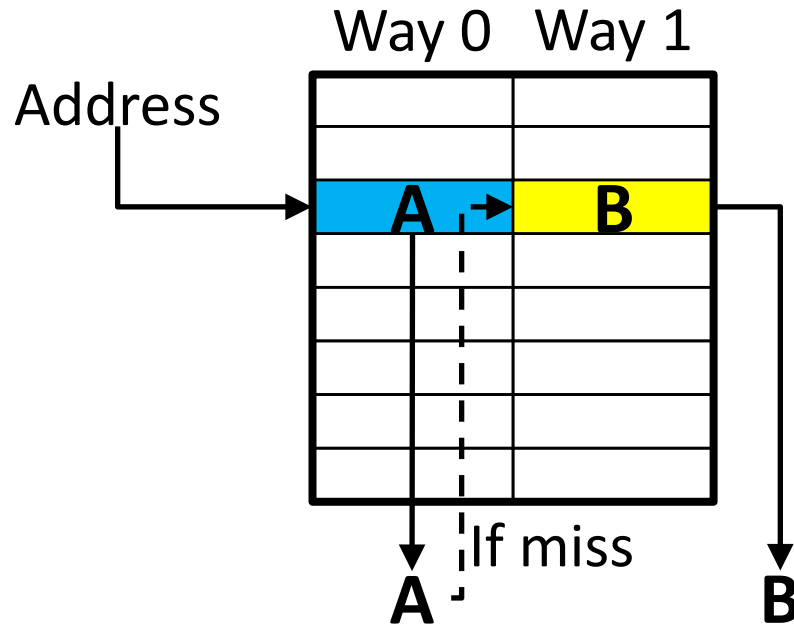
Intel Knights Landing Product (MCDRAM) uses this DRAM-cache organization.

POTENTIAL OF ASSOCIATIVITY



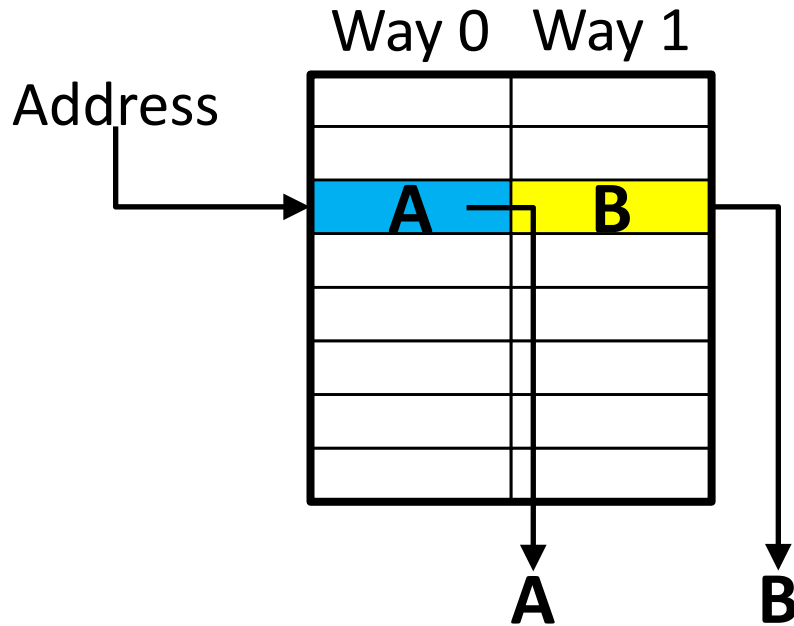
How can we make DRAM caches associative?

ASSOCIATIVITY OPTION 1: SERIAL TAG LOOKUP



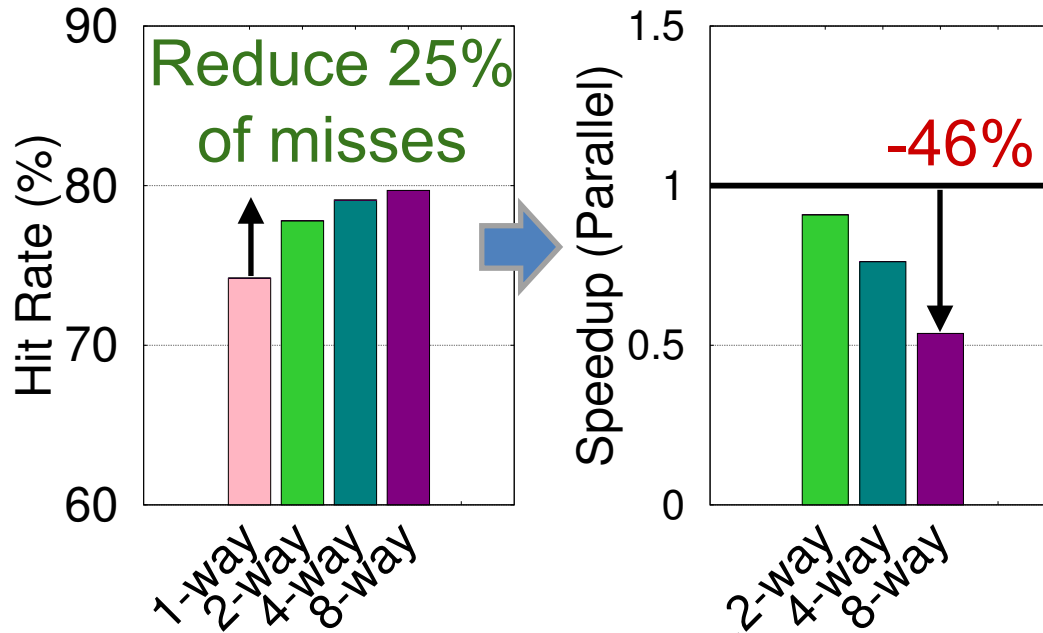
Serial Tag Lookup enables associativity, but, it has serialization delay.

ASSOCIATIVITY OPTION 2: PARALLEL TAG LOOKUP



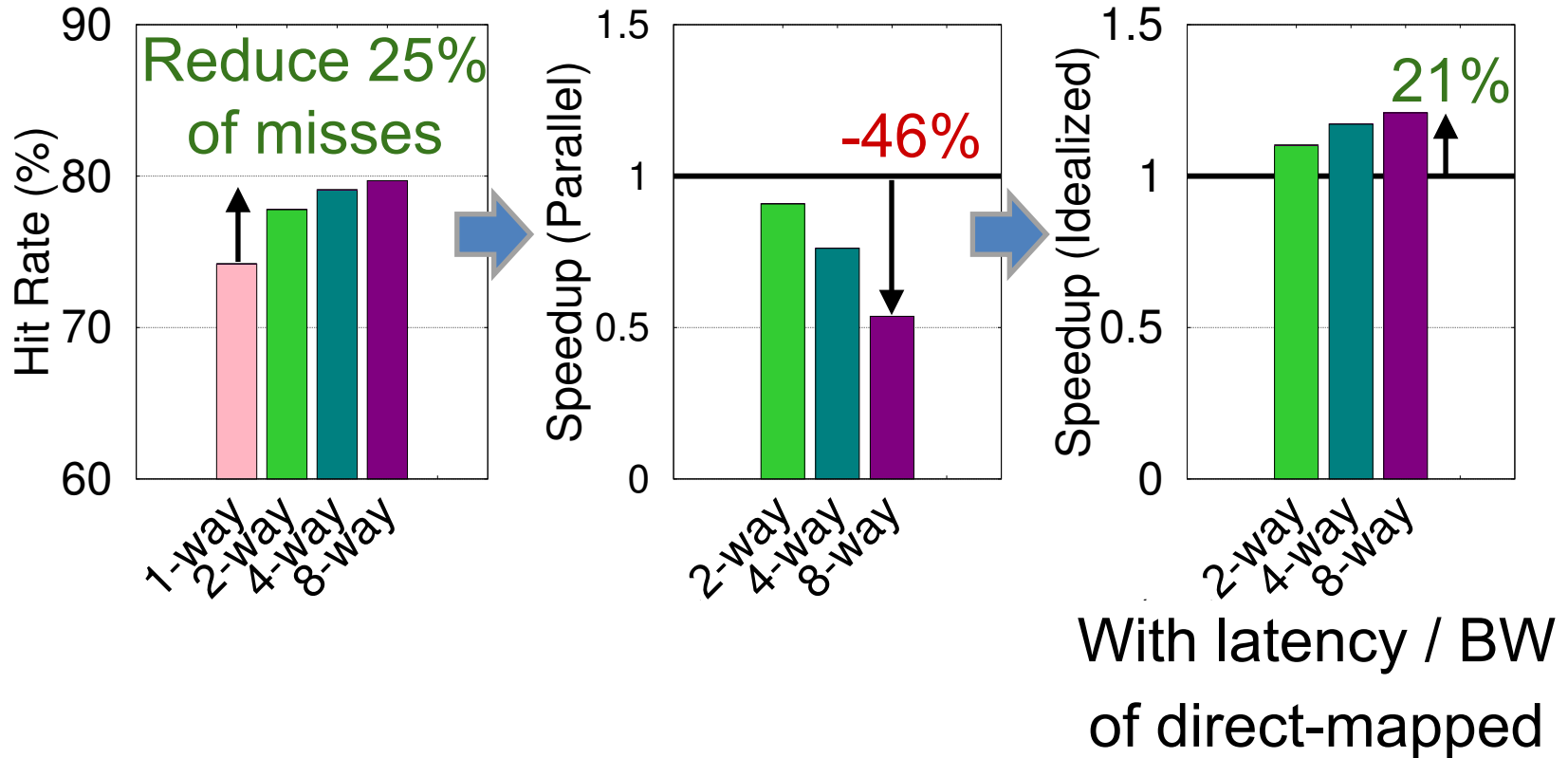
Parallel Lookup avoids serialization latency, but, it introduces 2x bandwidth cost.

ASSOCIATIVITY FOR DRAM CACHE (PARALLEL)



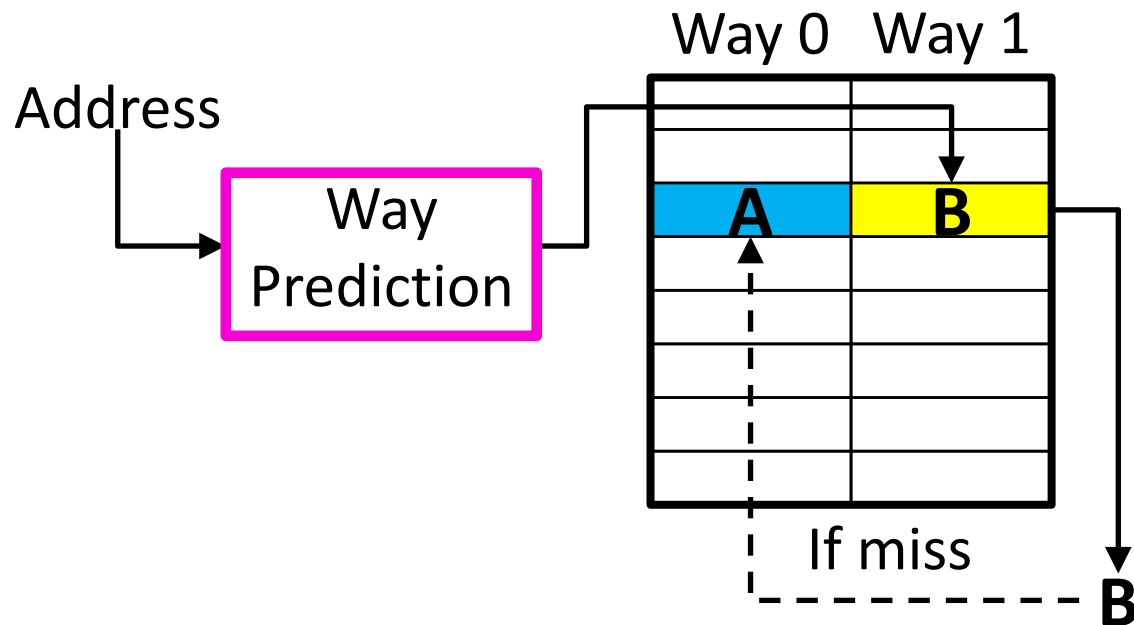
Increasing associativity naively actually degrades performance due to increased BW cost

ASSOCIATIVITY FOR DRAM CACHE (IDEAL)



Associativity must still maintain the latency/BW of direct-mapped caches. How?

OPTION 3: WAY-PREDICTED TAG LOOKUP



Way-Predicted Tag Lookup

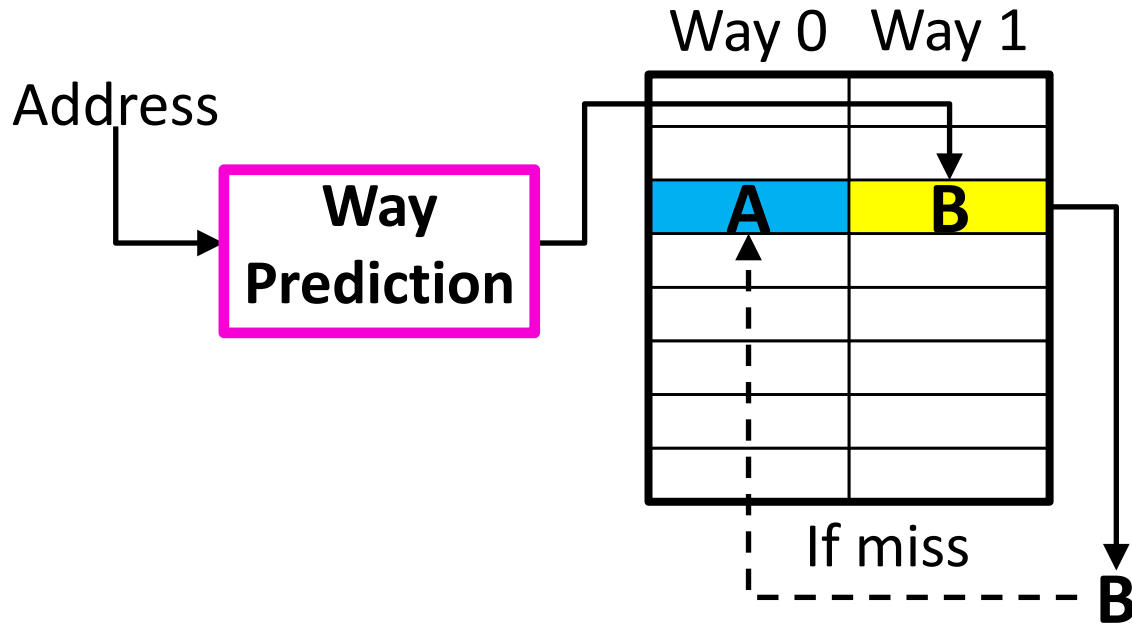
Way-Predicted Tag Lookup can obtain improved hit-rate, with BW / latency of direct-mapped cache.

WAY-PREDICTION ACCURACY & COST

	MRU Pred (1bit/set)	Partial-Tag (4bit/line)
SRAM Storage	4MB	32MB
Way-Pred Accuracy (2-way)	85.7%	97.3%
Accuracy (4-way)	74.3%	91.6%
Accuracy (8-way)	63.2%	81.2%

Prior methods for way-prediction have *low accuracy* and/or have *high storage overhead*.


TOWARDS ASSOCIATIVITY W/ WAY-PREDICTION



Way-Predicted Tag Lookup

Goal: Low storage-overhead and high accuracy way-prediction, to enable associative DRAM cache

ACCORD OVERVIEW

- Background
- **ACCORD** 
 - Probabilistic Way-Steering (PWS)
 - Ganged Way-Steering (GWS)
 - Skewed Way-Steering (SWS)
- Summary

INSIGHT: WAY-PREDICTABILITY AT LOW STORAGE?

Way 0 Way 1

EVEN	
ODD	EVEN
ODD	
EVEN	ODD

Base Install Policy (Rand)
Hard-to-predict (~50%)

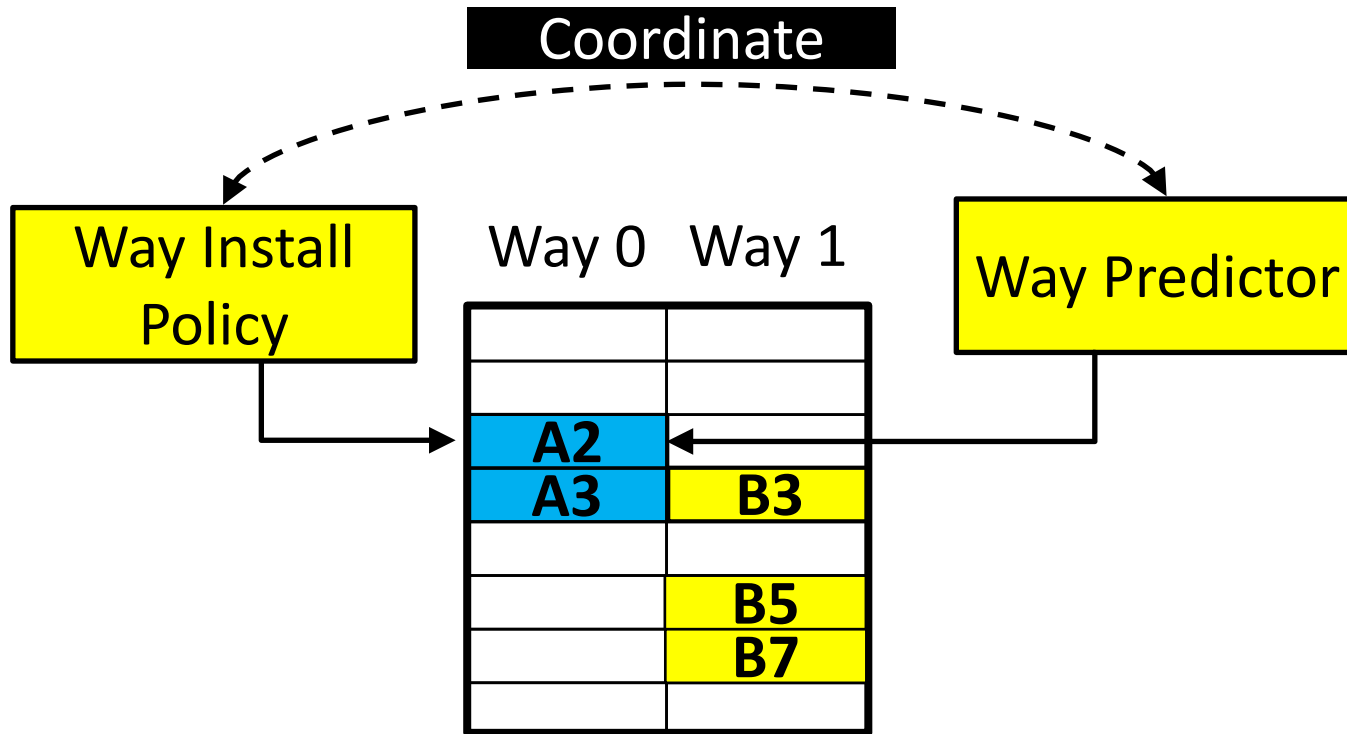
Way 0 Way 1

EVEN	
EVEN	ODD
	ODD
EVEN	ODD

Tag-based Install Policy
Predict 100%!
But, direct-mapped


Insight: Modifying install policy can make way-prediction much simpler!

PROPOSAL: ACCORD

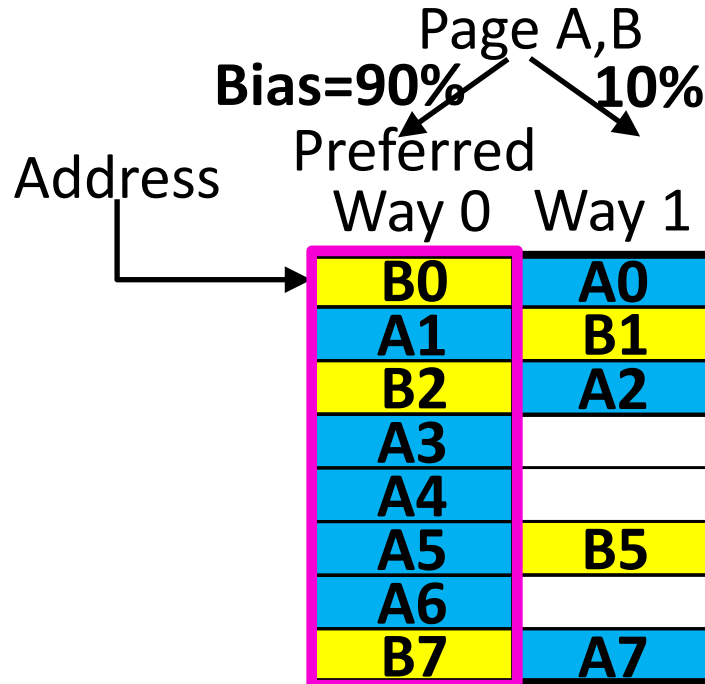


AssoCiativity by CoORDinating way-install and prediction. ACCORD achieves a way-predictable cache at low cost.

ACCORD OVERVIEW

- Background
- ACCORD
 - Probabilistic Way-Steering (PWS) 
 - Ganged Way-Steering (GWS)
 - Skewed Way-Steering (SWS)
- Summary

PROBABILISTIC WAY-STEERING



Static prediction: ~90%

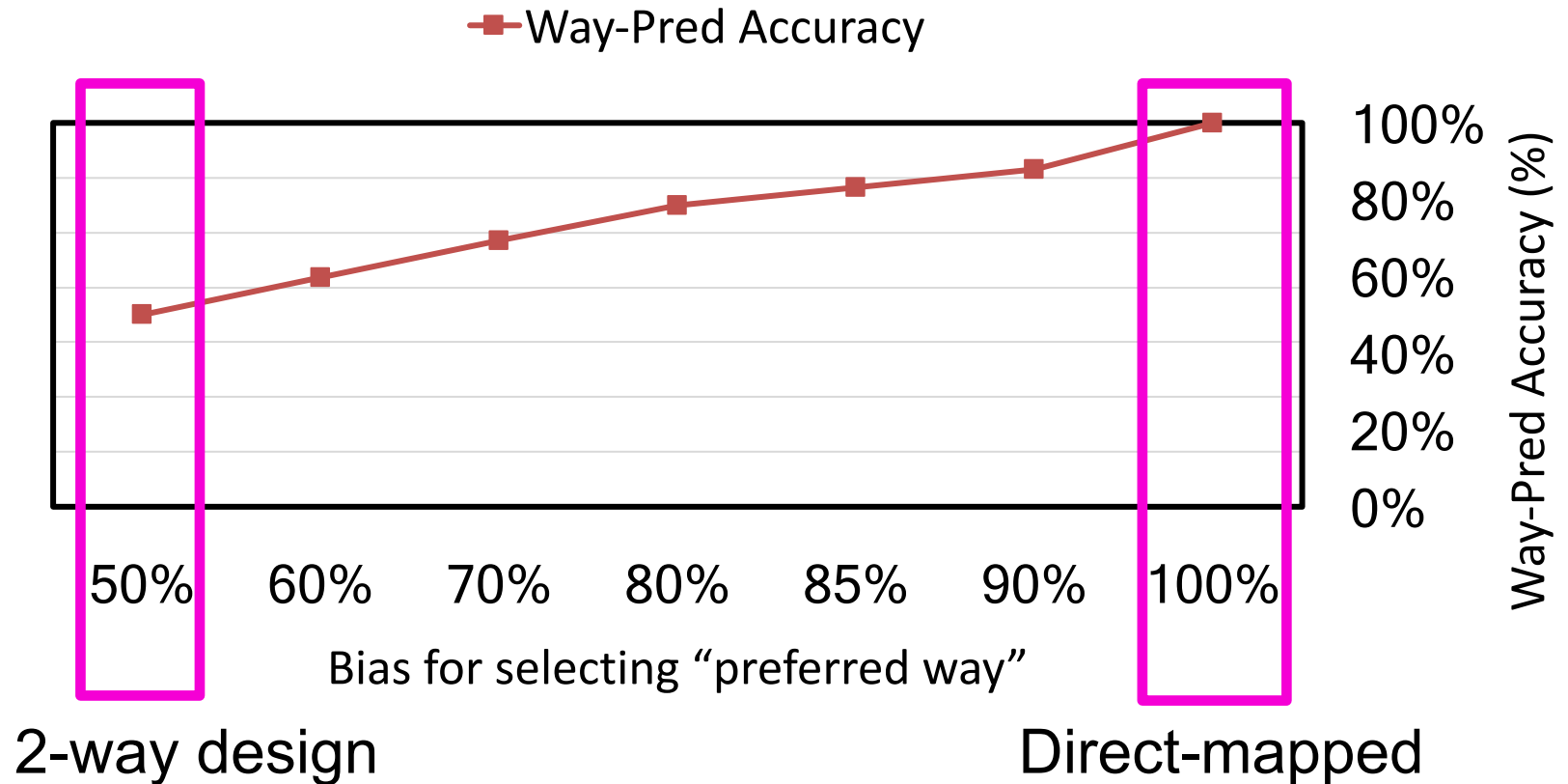
Install using PWS

Will use both ways, improve hit-rate

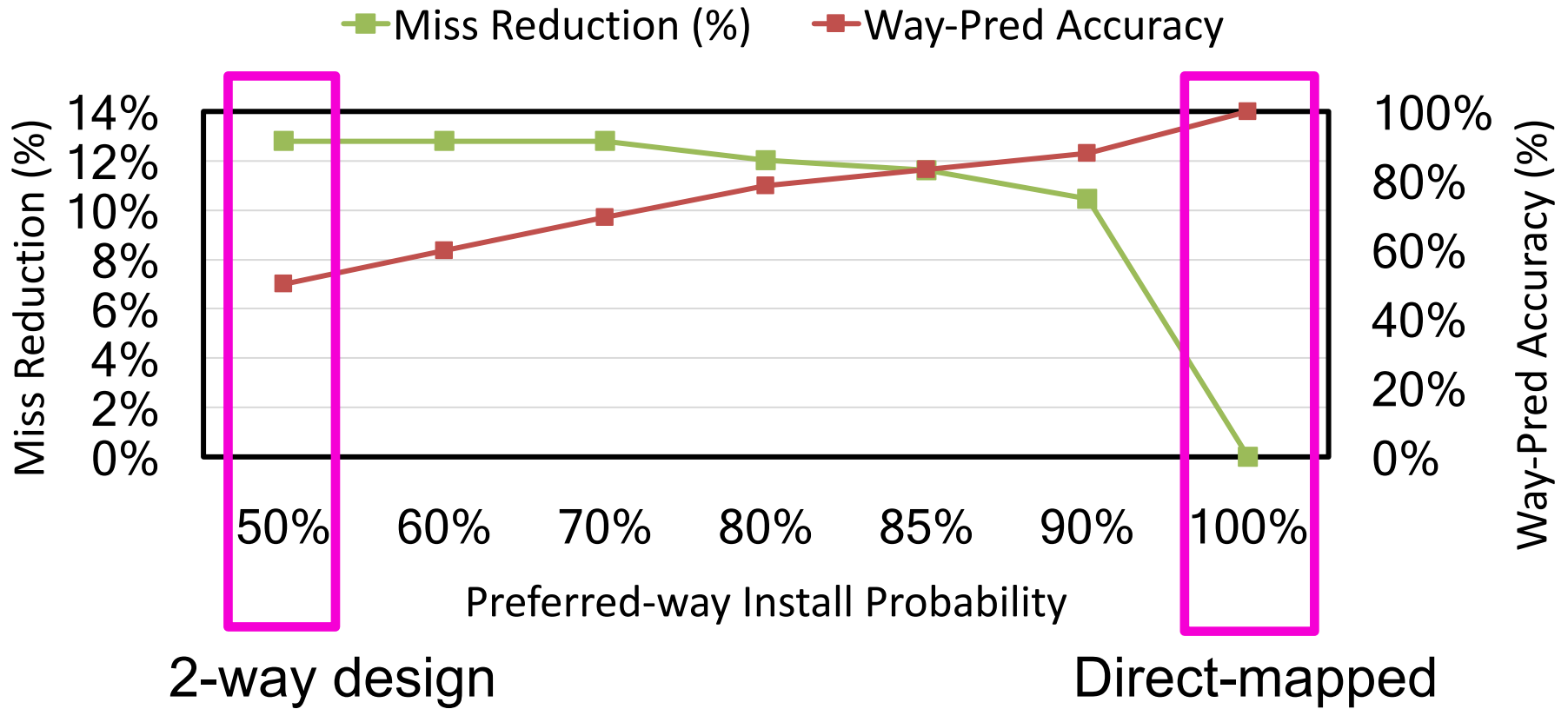
PWS enables way-predictability, by trading speed of learning to use both ways (hit-rate)

SENSITIVITY TO PWS PROBABILITY

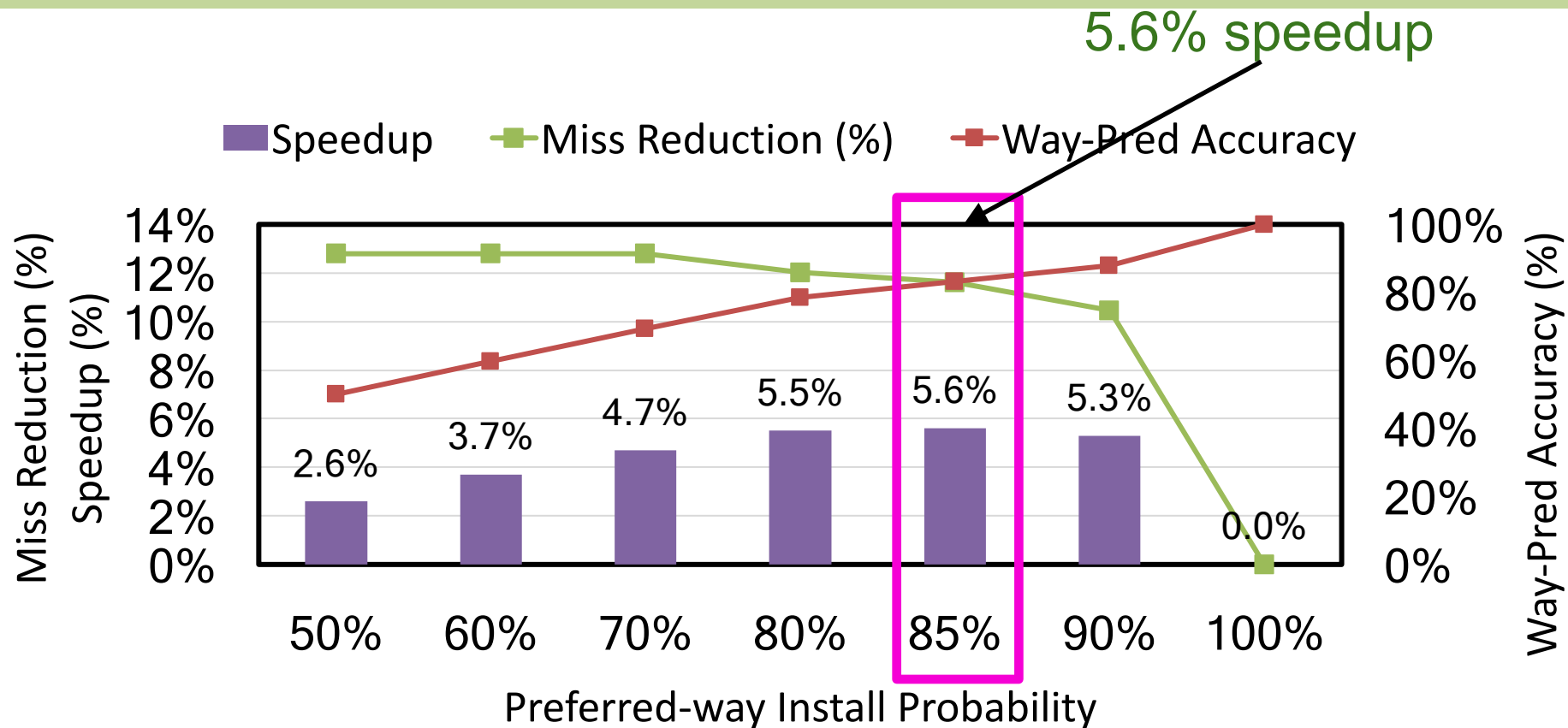
Preferred-way Install Probability = x% bias to install in preferred way



SENSITIVITY TO PWS PROBABILITY




SENSITIVITY TO PWS PROBABILITY

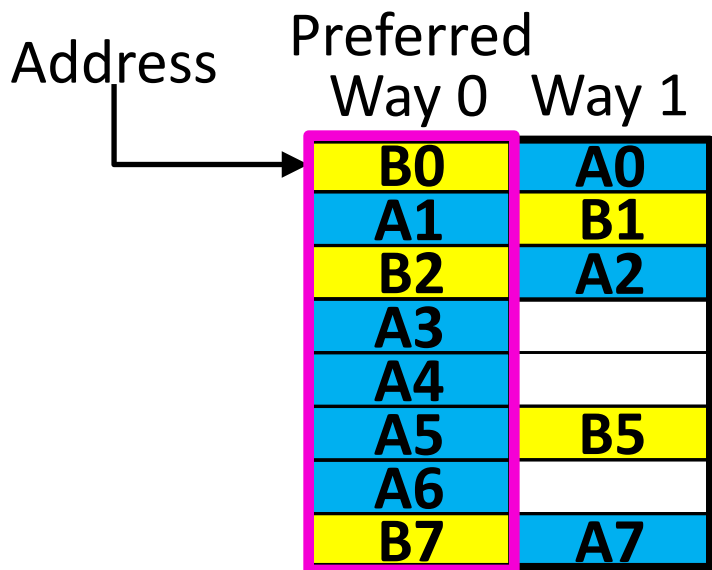


Preferred-way Install Probability (85%) provides best trade-off of hit-rate for WP accuracy, for 5.6% speedup.

ACCORD OVERVIEW

- Background
- ACCORD
 - Probabilistic Way-Steering (PWS)
 - **Ganged Way-Steering (GWS)** 
 - Skewed Way-Steering (SWS)
- Summary

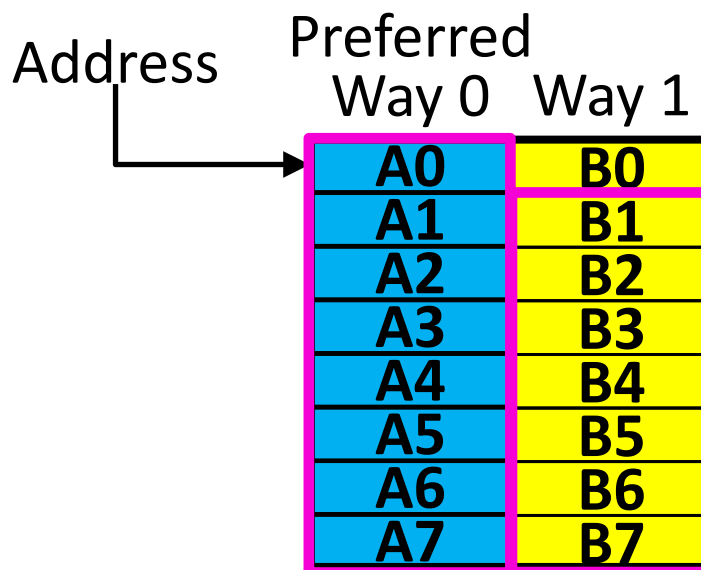
GANGED WAY-STEERING



Pred ~50%

Probabilistic Way-Steering

Per-line randomized decision



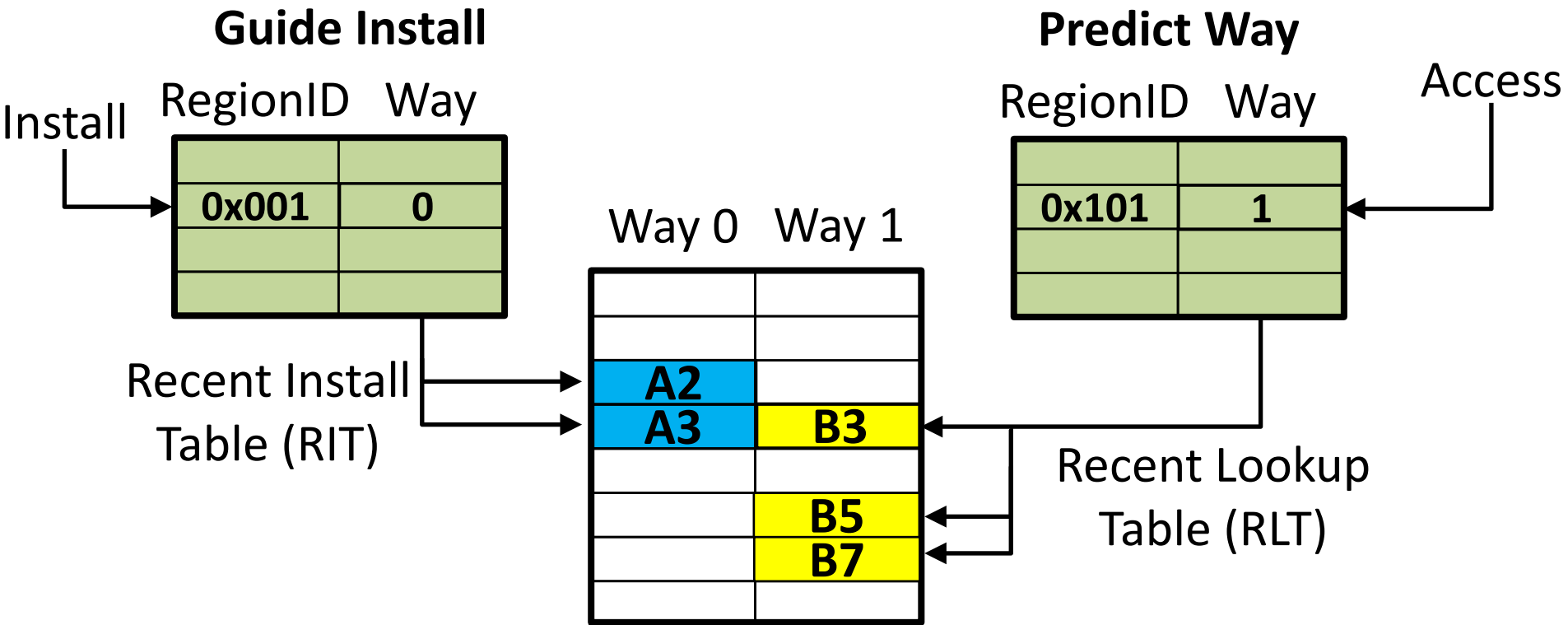
Pred >90%

Ganged Way-Steering

Per-page rand decision

Ganged Way-Steering makes install decision at large granularity, to improve predictability for workloads with high spatial locality.

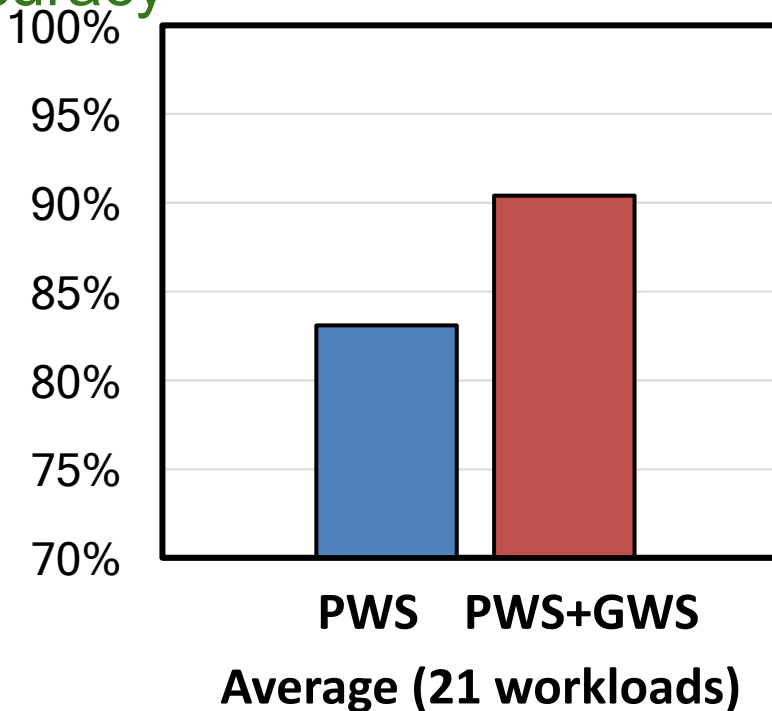
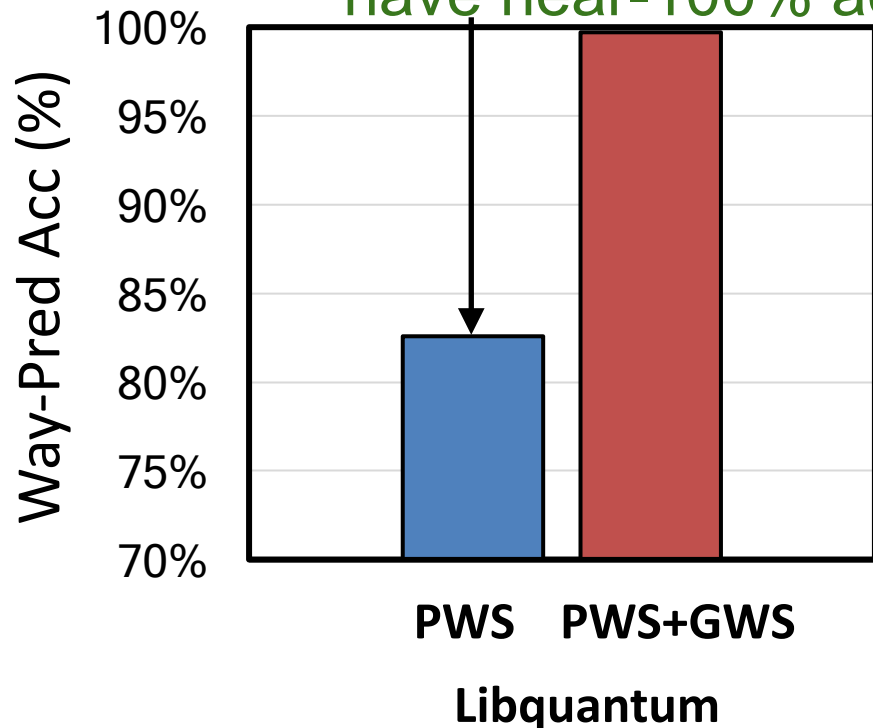
GANGED WAY-STEERING IMPLEMENTATION



GWS Per-Region Last-Way install + Last-Way prediction.
64-entry RIT and 64-entry RLT needs *only 320 Bytes*.

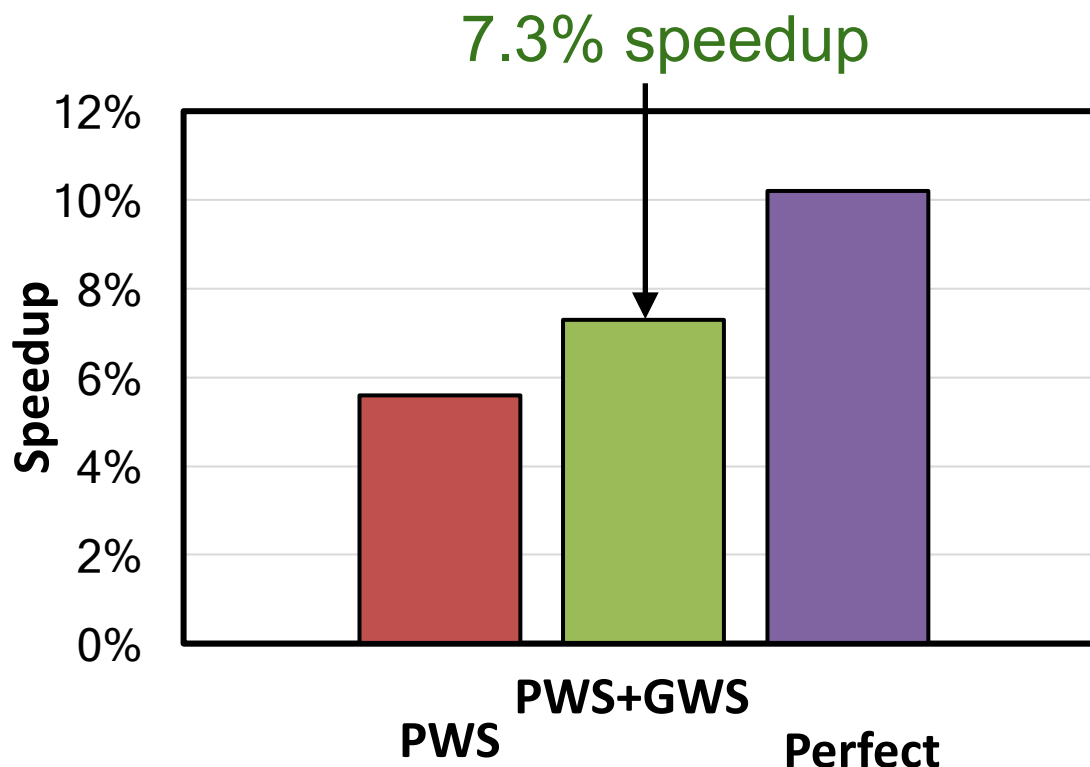
PWS+GWS WAY-PREDICTION ACCURACY

GWS enables spatial workloads to
PWS has ~85% base accuracy
have near-100% accuracy




Combination of PWS+GWS achieves 90% accuracy,
at the cost of 320B storage.

PWS+GWS (ACCORD 2-WAY) RESULTS



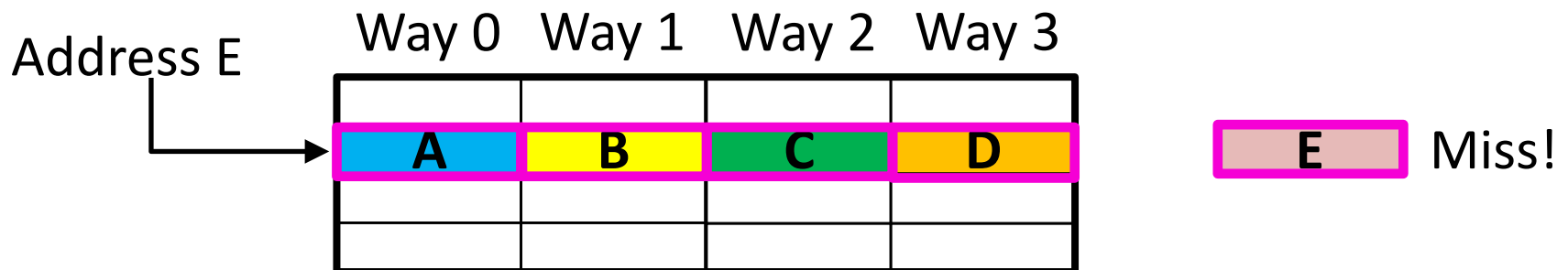
PWS + GWS gets 7.3% of 10% speedup of perfectly-predicted 2-way cache.

ACCORD OVERVIEW

- Background
- ACCORD
 - Probabilistic Way-Steering (PWS)
 - Ganged Way-Steering (GWS)
 - Skewed Way-Steering (SWS) 
- Summary

DIFFICULTY IN SCALING TO N-WAYS

- Scaling ACCORD to N-ways
 - ACCORD 4-way has 3% speedup
 - ACCORD 8-way has 6% slowdown...



- Miss confirmation: N-way cache needs N accesses to confirm line is not resident

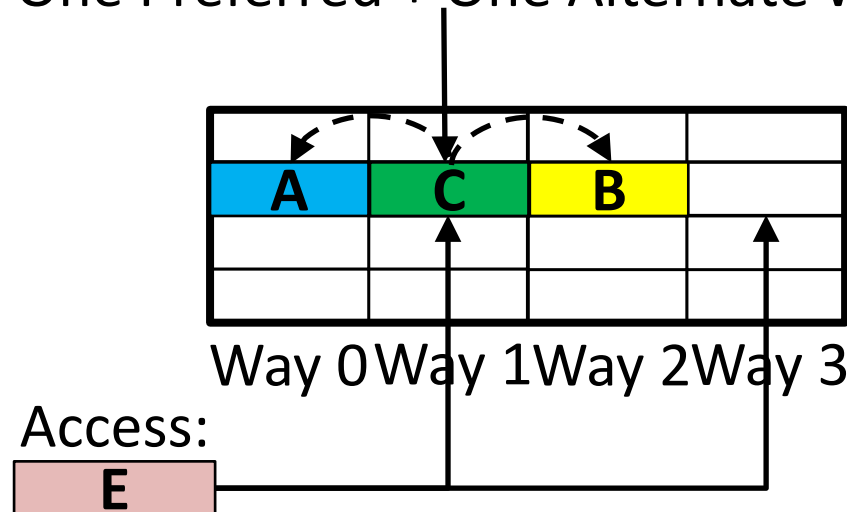
We need solutions to reduce miss-confirmation

SOLUTION: SKEWED WAY-STEERING

4-way with 2-skew:

Access: ABC

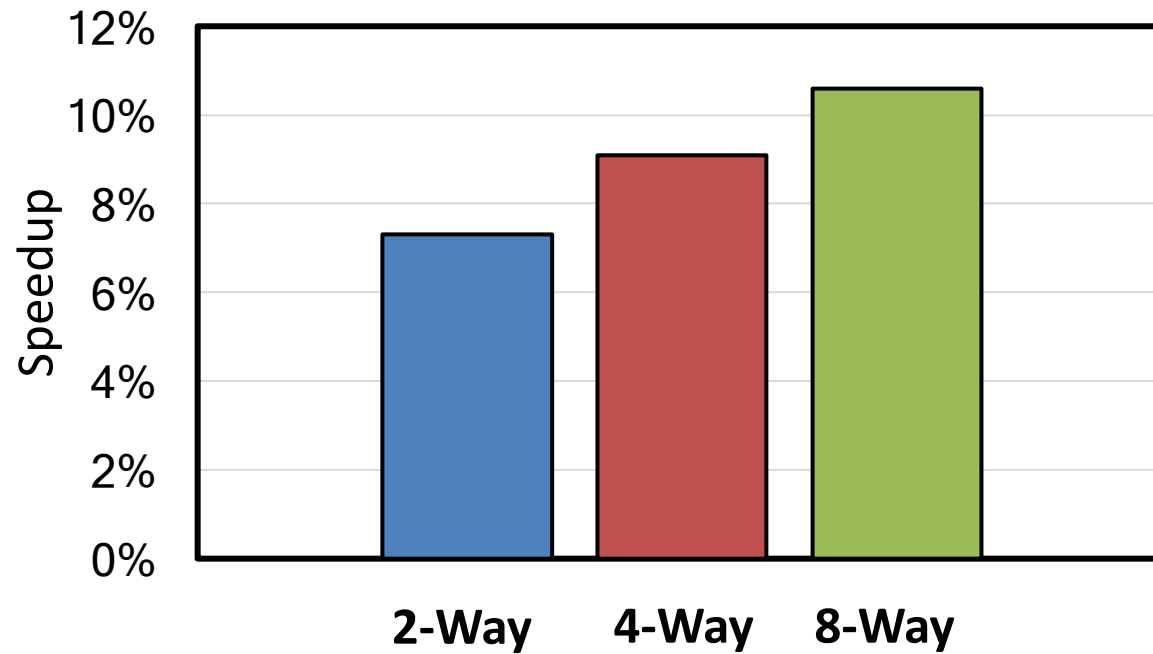
One Preferred + One Alternate way



Only 2 lookups to determine miss


Restricting placement, reduces miss-confirmation → hit-rate benefits without any storage overhead

SPEEDUP FROM ACCORD (WITH SWS)



SWS 8-way achieves 11% speedup

ACCORD OVERVIEW

- Background
- ACCORD
 - Probabilistic Way-Steering (PWS)
 - Ganged Way-Steering (GWS)
 - Skewed Way-Steering (SWS)
- Summary 

SUMMARY OF ACCORD

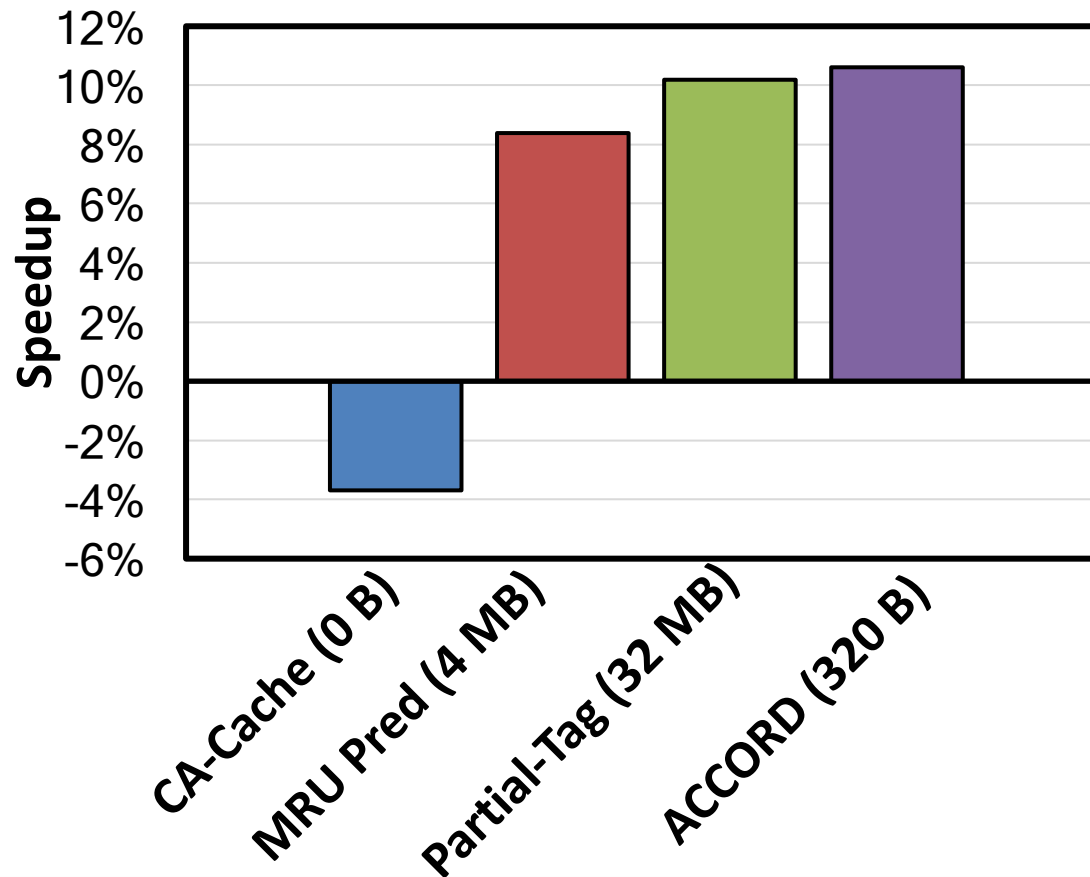
- ACCORD: associative DRAM caches by coordinating way-install and way-prediction.
- Probabilistic Way-Steering
 - Biased-install enables accurate static way-prediction
- Ganged Way-Steering
 - Region-based install enables accurate region-based way-prediction
- Skewed Way-Steering
 - Skew enables flexibility in line placement, while maintaining miss cost
- ACCORD enables *associativity* at negligible storage cost (320B), to achieve 11% speedup.

ACCORD backup slides

REPLACEMENT POLICY?

- LRU
 - State in SRAM
 - 1-bit per line needs 8MB. Size of Last-level cache
 - State in DRAM
 - 9% slowdown due to state-update cost (Hit to alternate way)

COMPARISON TO OTHER WAY PREDICTORS



ACCORD outperforms other predictors while needing negligible storage overhead (320 B)

COLUMN-ASSOCIATIVE CACHE

- Column-associative / Hash-Rehash cache
 - Install lines in preferred way (way-0)
 - On eviction, move line to alternate way (way-1)
 - On hit to alternate way, move to preferred way
- Effectiveness
 - In general, way-prediction accuracy similar to MRU
 - But, requires significant bandwidth to swap lines on hit to alternate way. CA-cache thus causes 4% slowdown.