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

ISCA 2018

Authors:
Vincent 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

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

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

Too large for SRAM
Organize at \textit{line granularity} (64B) for high cache utilization.

Gigascale cache needs \textit{large tag-store} (tens of MBs).

Practical designs must store Tags in DRAM.

How to architect tag-store for low-latency tag access?
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.
How can we make DRAM caches associative?

Assumes 16-core system, with 4GB DRAM-Cache, in front of PCM memory.
Serial Tag Lookup enables associativity, but, it has serialization delay.
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)**

**Reduce 25% of misses**

**Speedup (Parallel)**

**Speedup (Idealized)**

With latency / BW of direct-mapped caches.

**Associatitivity must still maintain the latency/BW of direct-mapped caches. How?**
OPTION 3: WAY-PREDICTED TAG LOOKUP

Way-Predicted Tag Lookup can obtain improved hit-rate, with BW / latency of direct-mapped cache.
Prior methods for way-prediction have low accuracy and/or have high storage overhead.
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?

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

![Way 0 Way 1](image)

<table>
<thead>
<tr>
<th></th>
<th>Way 0</th>
<th>Way 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVEN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODD</td>
<td>EVEN</td>
<td></td>
</tr>
<tr>
<td>ODD</td>
<td></td>
<td>ODD</td>
</tr>
<tr>
<td>EVEN</td>
<td>ODD</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

**Static prediction:** ~90%

Install using PWS

Will use both ways, improve hit-rate

<table>
<thead>
<tr>
<th>Address</th>
<th>Way 0</th>
<th>Way 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>A0</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>B1</td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>A2</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td></td>
<td>B5</td>
</tr>
<tr>
<td>A6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B7</td>
<td>A7</td>
<td></td>
</tr>
</tbody>
</table>

- **Bias=90%**
- **Preferred**
- **Page A, B**

10%
SENSITIVITY TO PWS PROBABILITY

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

Way-Pred Accuracy

Bias for selecting “preferred way”

2-way design

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

• ACCORD
  – Probabilistic Way-Steering (PWS)
  – Ganged Way-Steering (GWS)
  – Skewed Way-Steering (SWS)

• Summary
GANGED WAY-STEERING

Probabilistic Way-Steering

Per-line randomized decision

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

Ganged Way-Steering

Per-page rand decision
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 have near 100% accuracy. PWS has ~85% base accuracy. Combination of PWS+GWS achieves 90% accuracy, at the cost of 320B storage.
PWS + GWS gets 7.3% of 10% speedup of perfectly-predicted 2-way cache.

System assumes 4GB DRAM Cache, and PCM-based main memory.
• 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

Way 0 Way 1 Way 2 Way 3

Access:

Only 2 lookups to determine miss

Restricting placement, reduces miss-confirmation ➔ hit-rate benefits without any storage overhead
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)
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.