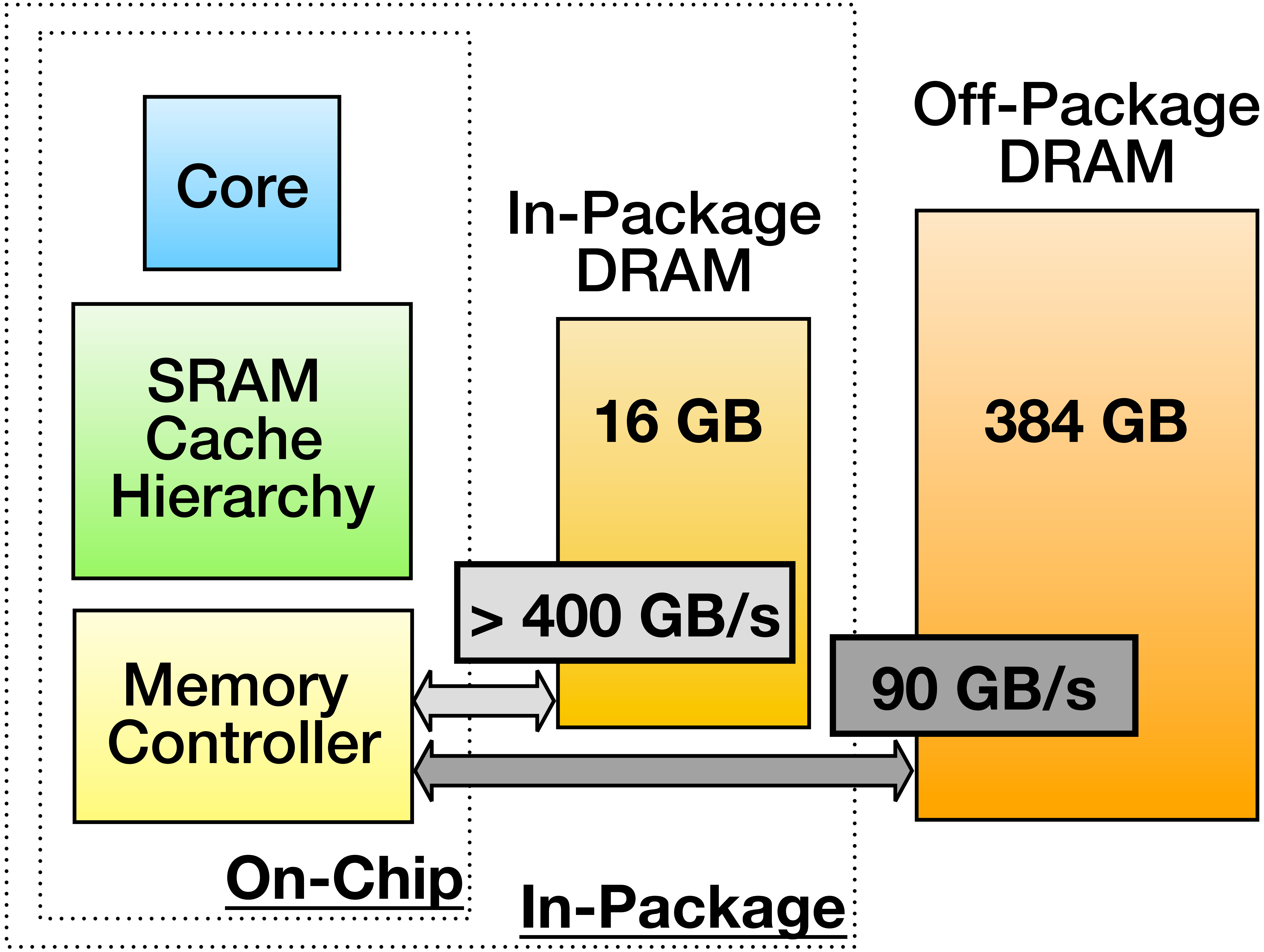
Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation

Xiangyao Yu[1], Christopher Hughes[2], Nadathur Satish[2],
Onur Mutlu[3], Srinivas Devadas[1]

[1]MIT [2]Intel Labs [3]ETH Zürich

High-Bandwidth In-Package DRAM

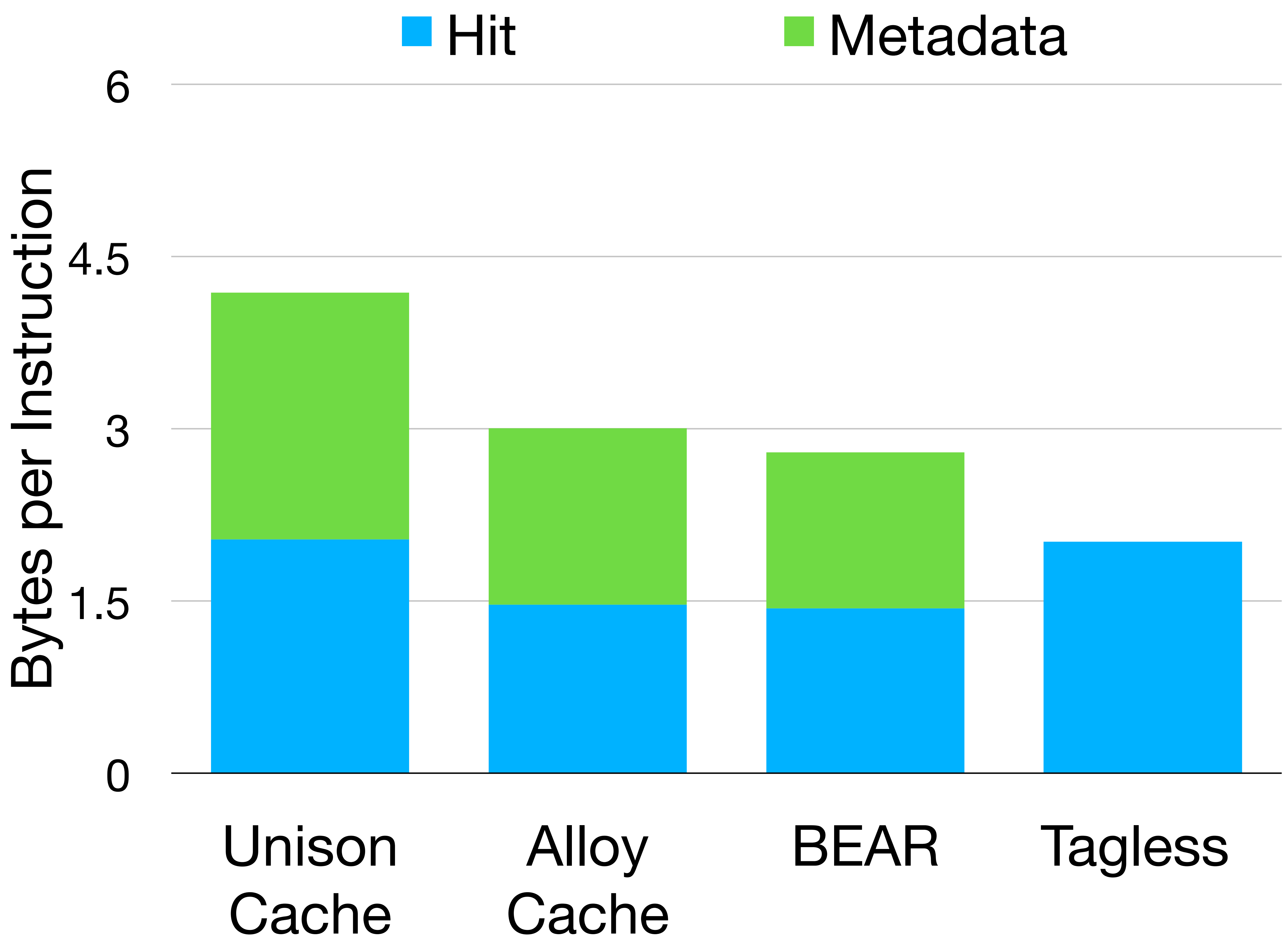
- In-package DRAM has
  - **5X higher bandwidth** than off-package DRAM
  - **Similar latency** as off-package DRAM
  - **Limited capacity** (up to 16 GB)
- In-package DRAM can be used as a cache



* Numbers from Intel Knights Landing

Bandwidth Inefficiency in Existing DRAM Cache Designs

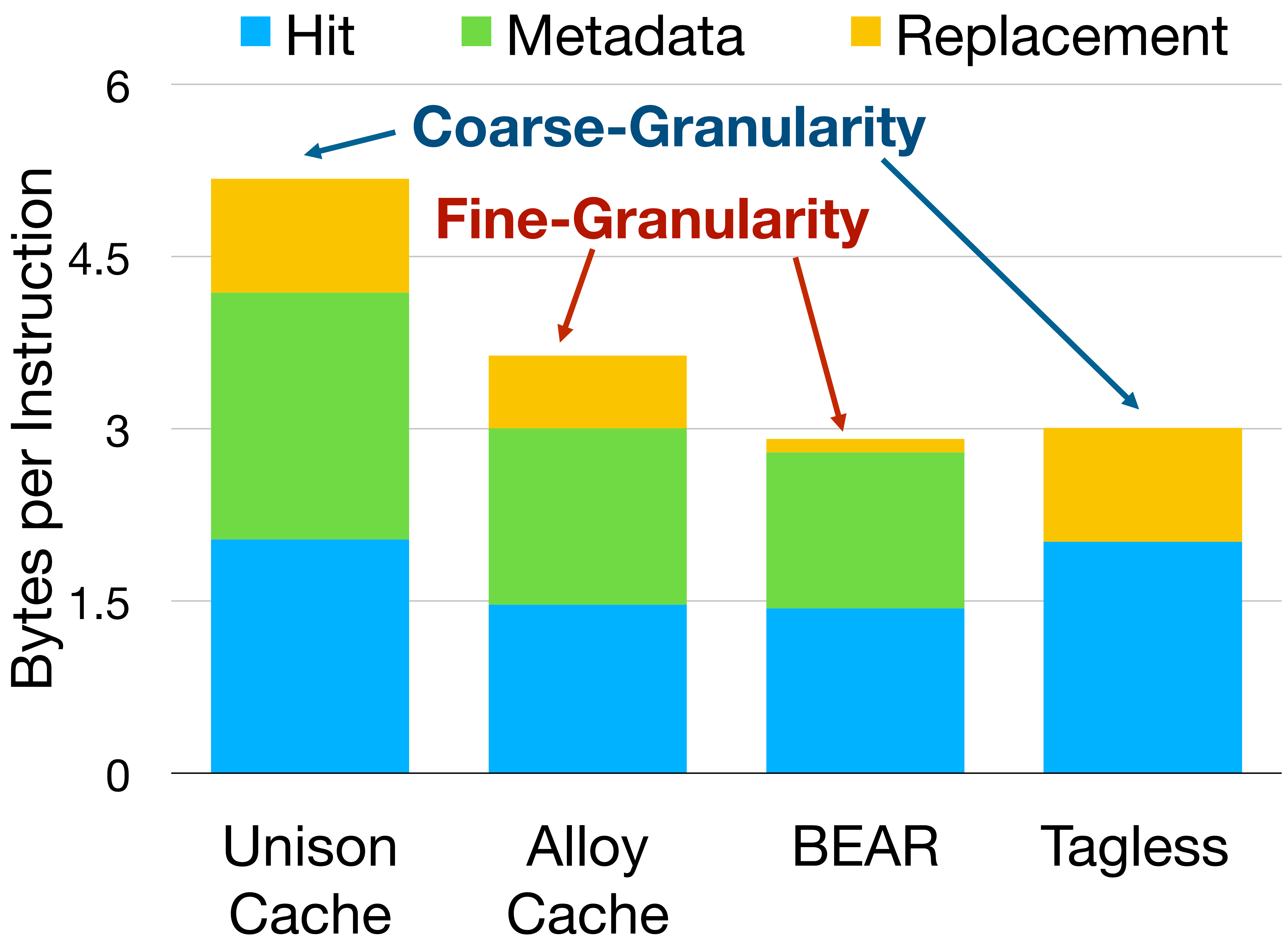
- **Drawback 1:** Metadata traffic (e.g., tags, LRU bits, frequency counters, etc.)



DRAM Cache Traffic Breakdown

Bandwidth Inefficiency in Existing DRAM Cache Designs

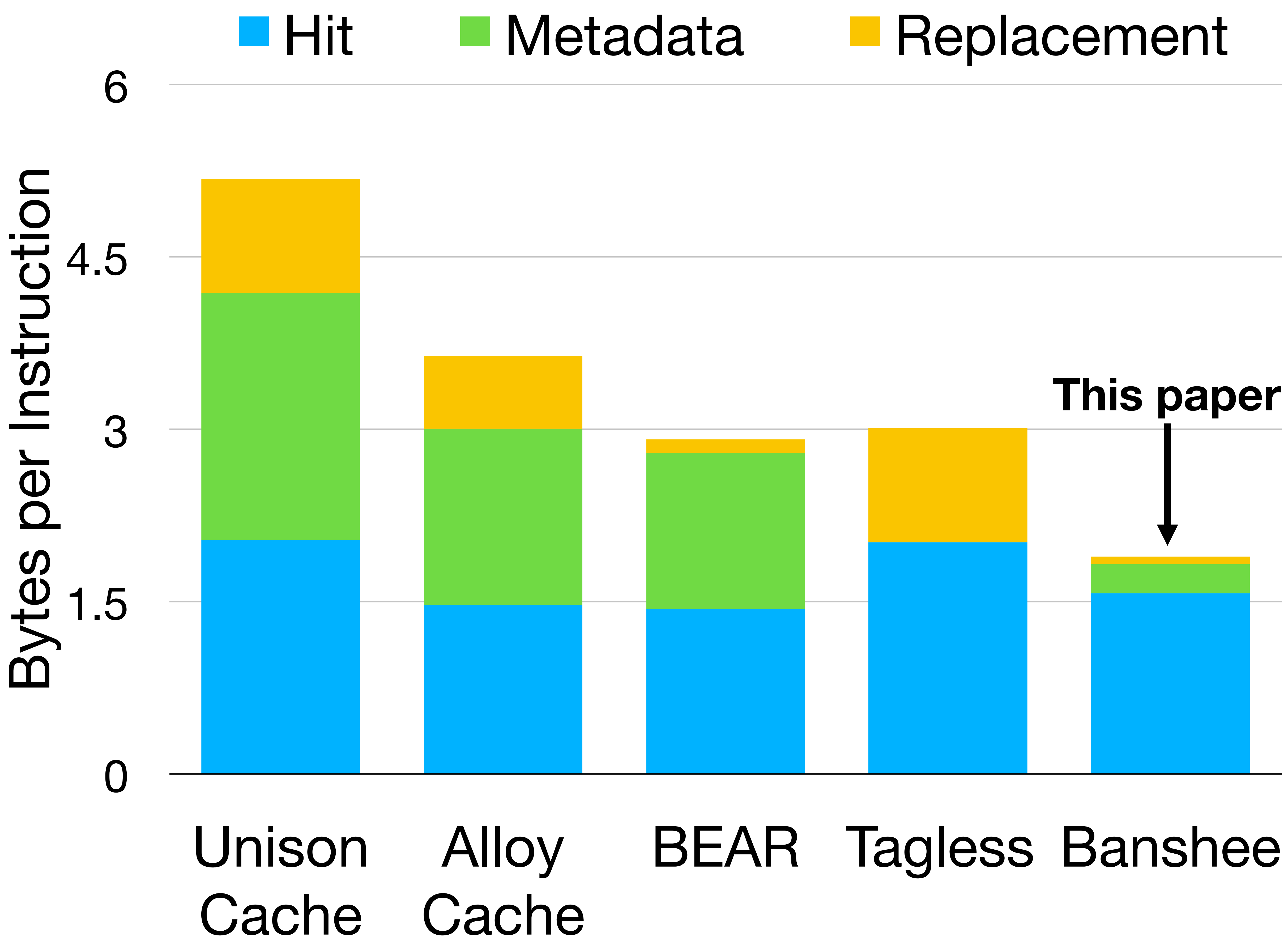
- **Drawback 1:** Metadata traffic (e.g., tags, LRU bits, frequency counters, etc.)
- **Drawback 2:** Cache replacement traffic
  - Especially for coarse-granularity (e.g., page-granularity) DRAM cache designs



DRAM Cache Traffic Breakdown

Banshee Improves DRAM Bandwidth Efficiency

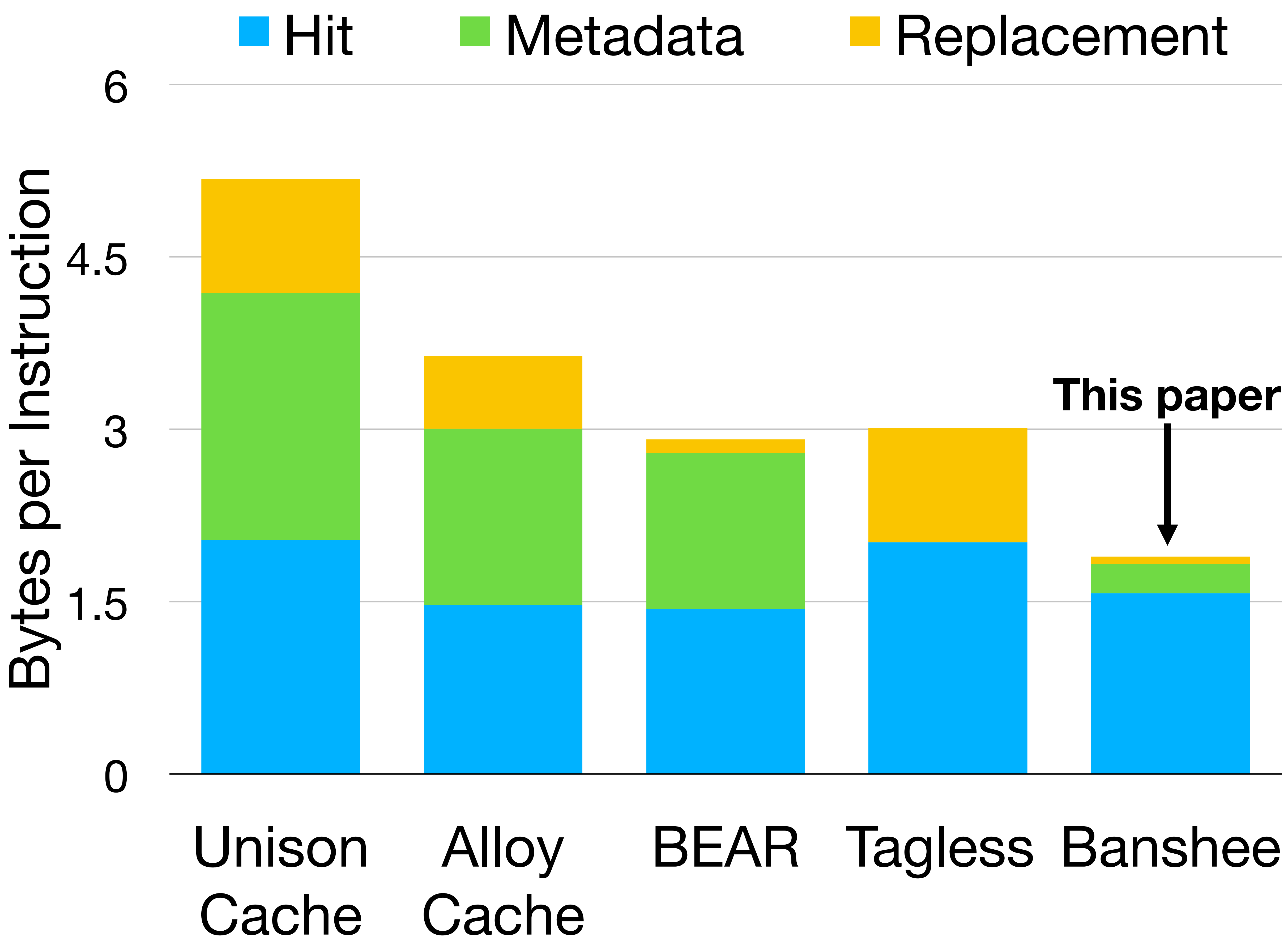
- **Idea 1**: Page-table-based contents tracking with efficient translation lookaside buffer (TLB) coherence
    - Track contents of DRAM cache using page tables and TLBs
    - Lightweight TLB coherence mechanism



DRAM Cache Traffic Breakdown

Banshee Improves DRAM Bandwidth Efficiency

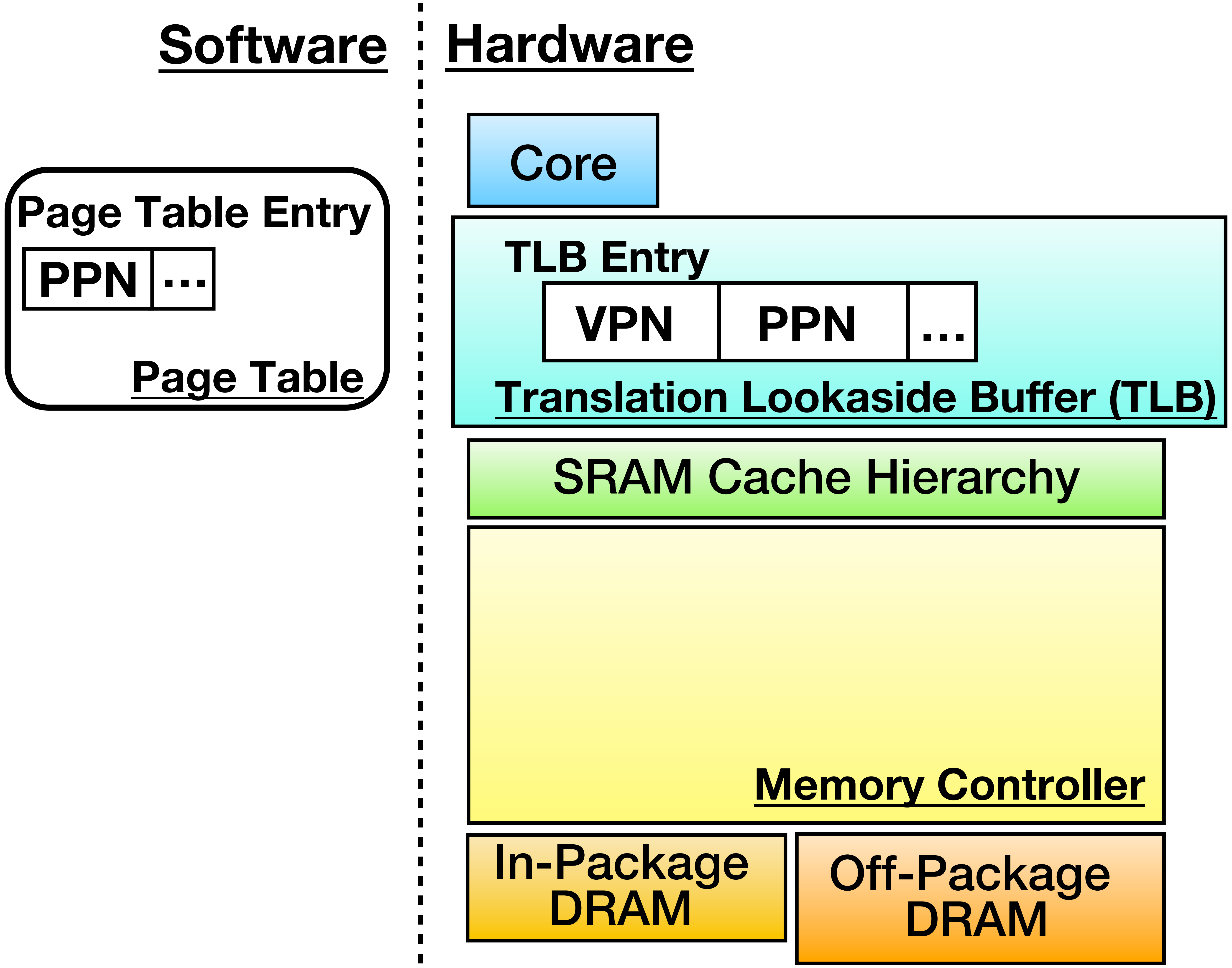
- **Idea 1**: Page-table-based contents tracking with efficient translation lookaside buffer (TLB) coherence
- **Idea 2**: Bandwidth-aware frequency-based replacement (FBR) policy
  - **Replacement traffic reduction:** Limit the rate of DRAM cache replacement
  - **Metadata traffic reduction:** Access metadata for a sampled fraction of memory accesses



DRAM Cache Traffic Breakdown

Page-Table-Based DRAM Cache Contents Tracking

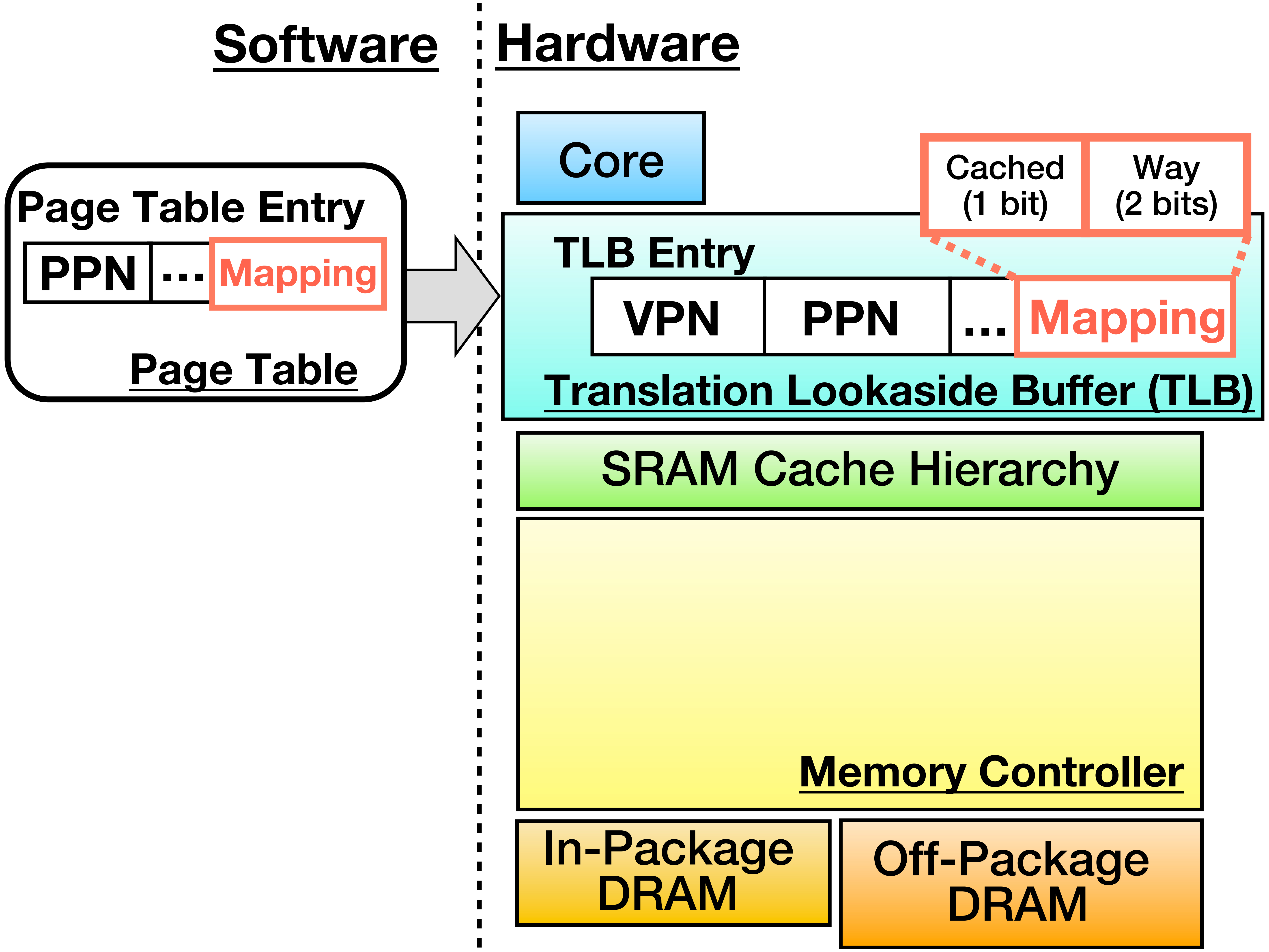
- Track DRAM cache contents using the the virtual memory mechanism
- **Advantage**
  - Zero overhead for tag storage and lookup
- **Disadvantage**
  - TLB coherence overhead
  - Cache replacement overhead



* assuming 4-way set associativity DRAM cache

Idea 1: Efficient TLB Coherence

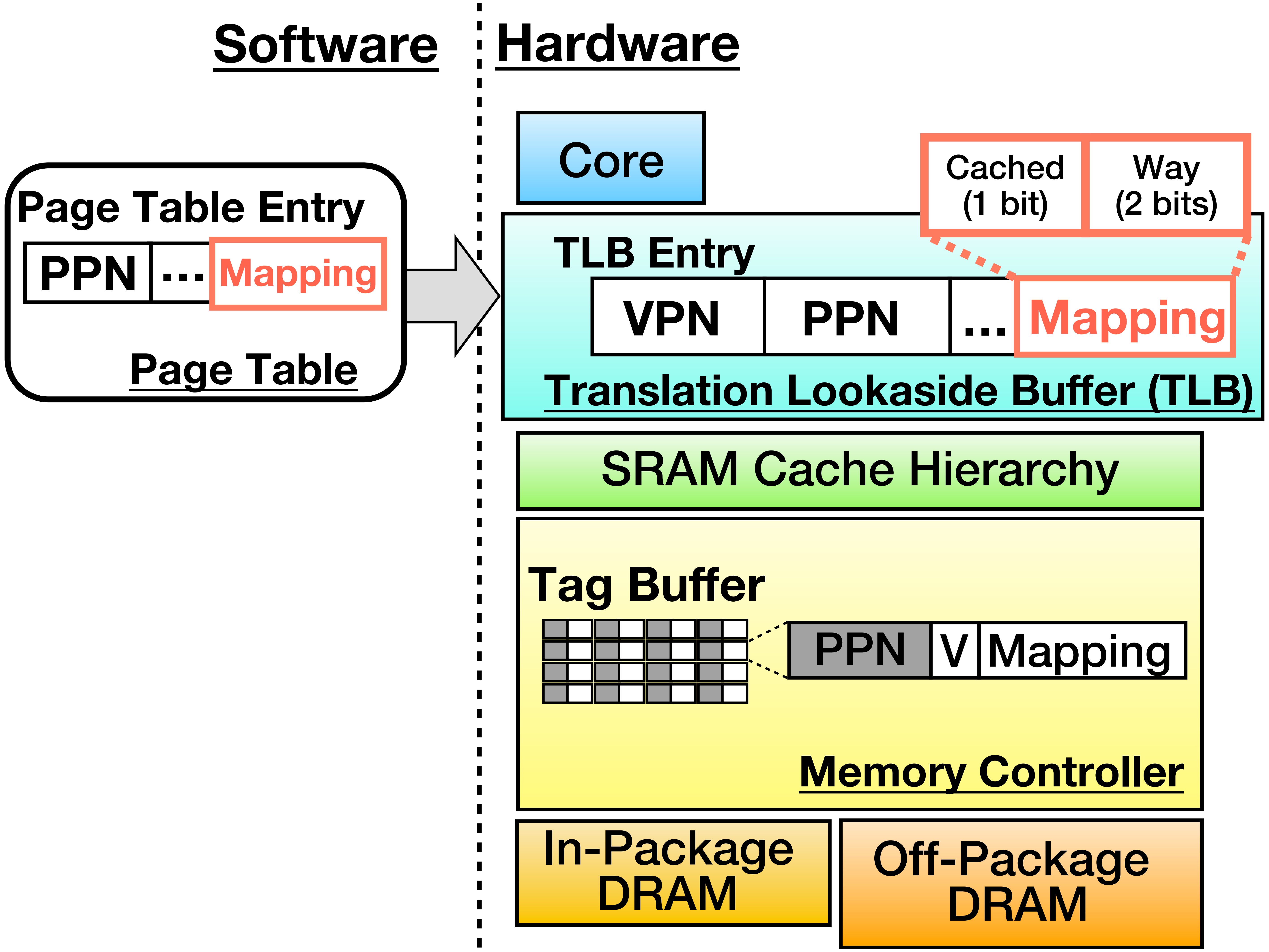
- Track DRAM cache contents using **page tables** and **TLBs**



* Assuming 4-way set-associative DRAM cache

Idea 1: Efficient TLB Coherence

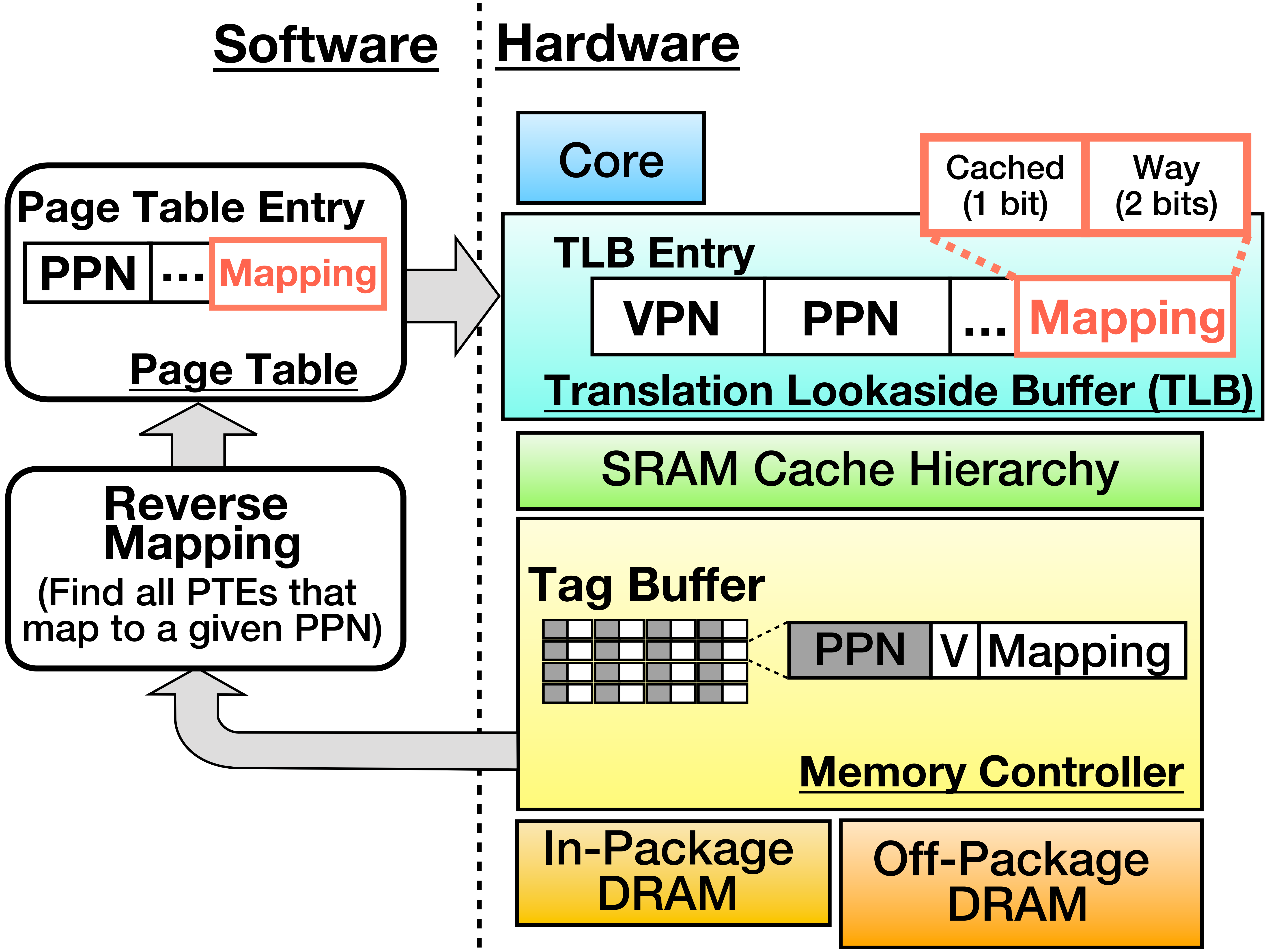
- Track DRAM cache contents using **page tables** and **TLBs**
- Maintain latest mapping for recently remapped pages in the **Tag Buffer**



* Assuming 4-way set-associative DRAM cache

Idea 1: Efficient TLB Coherence

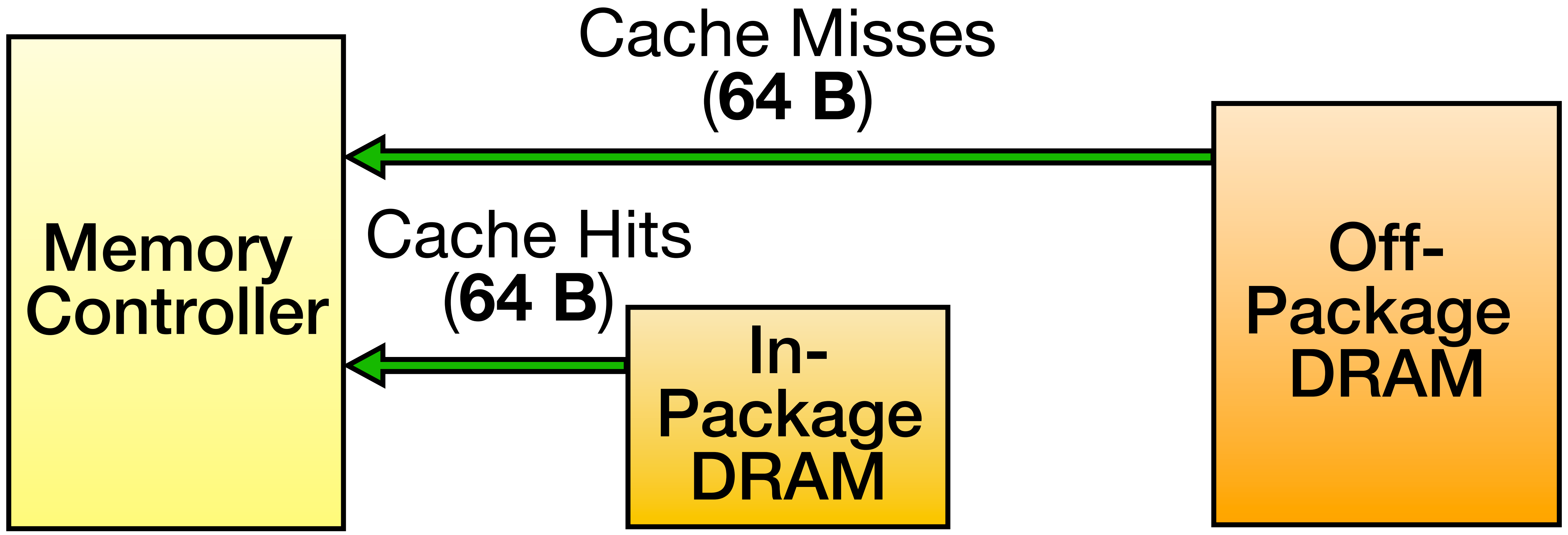
- Track DRAM cache contents using **page tables** and **TLBs**
- Maintain latest mapping for recently remapped pages in the **Tag Buffer**
- Enforce TLB coherence **lazily** when the Tag Buffer is full to amortize the cost



* Assuming 4-way set-associative DRAM cache

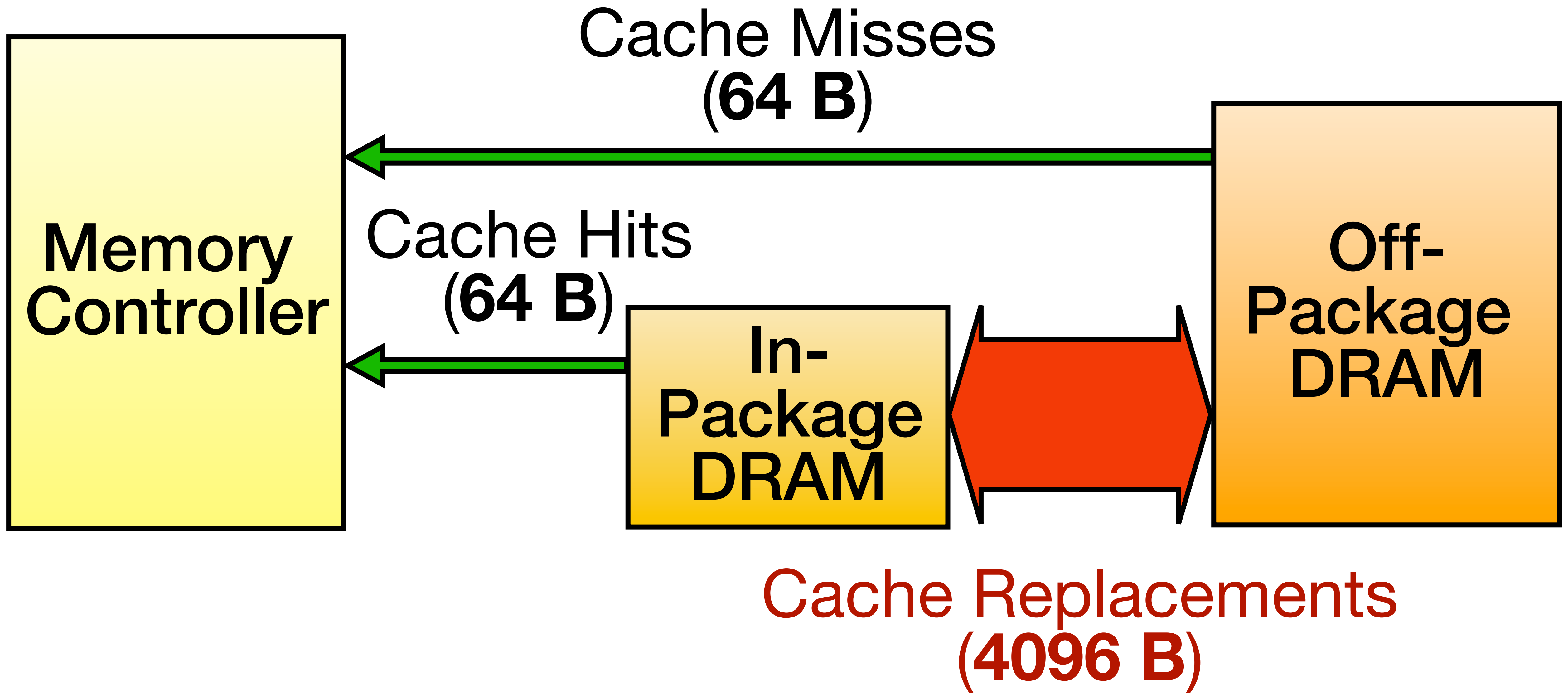
Idea 2: Bandwidth-Aware Cache Replacement

- **DRAM cache replacement incurs significant DRAM traffic**
  - Cache replacement traffic
  - Metadata traffic (e.g., frequency counter lookups/updates)



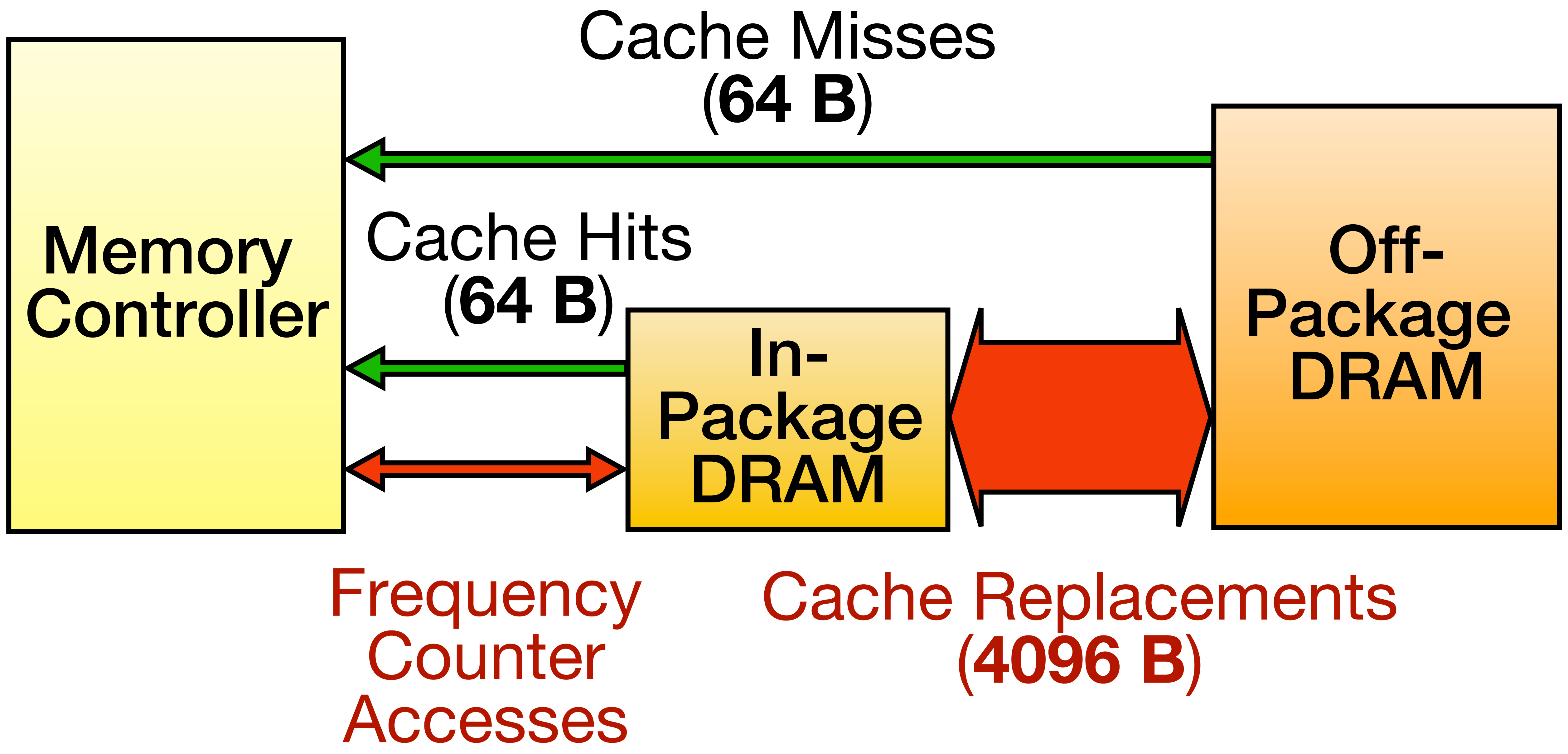
Idea 2: Bandwidth-Aware Cache Replacement

- **DRAM cache replacement incurs significant DRAM traffic**
  - Cache replacement traffic
  - Metadata traffic (e.g., frequency counter lookups/updates)



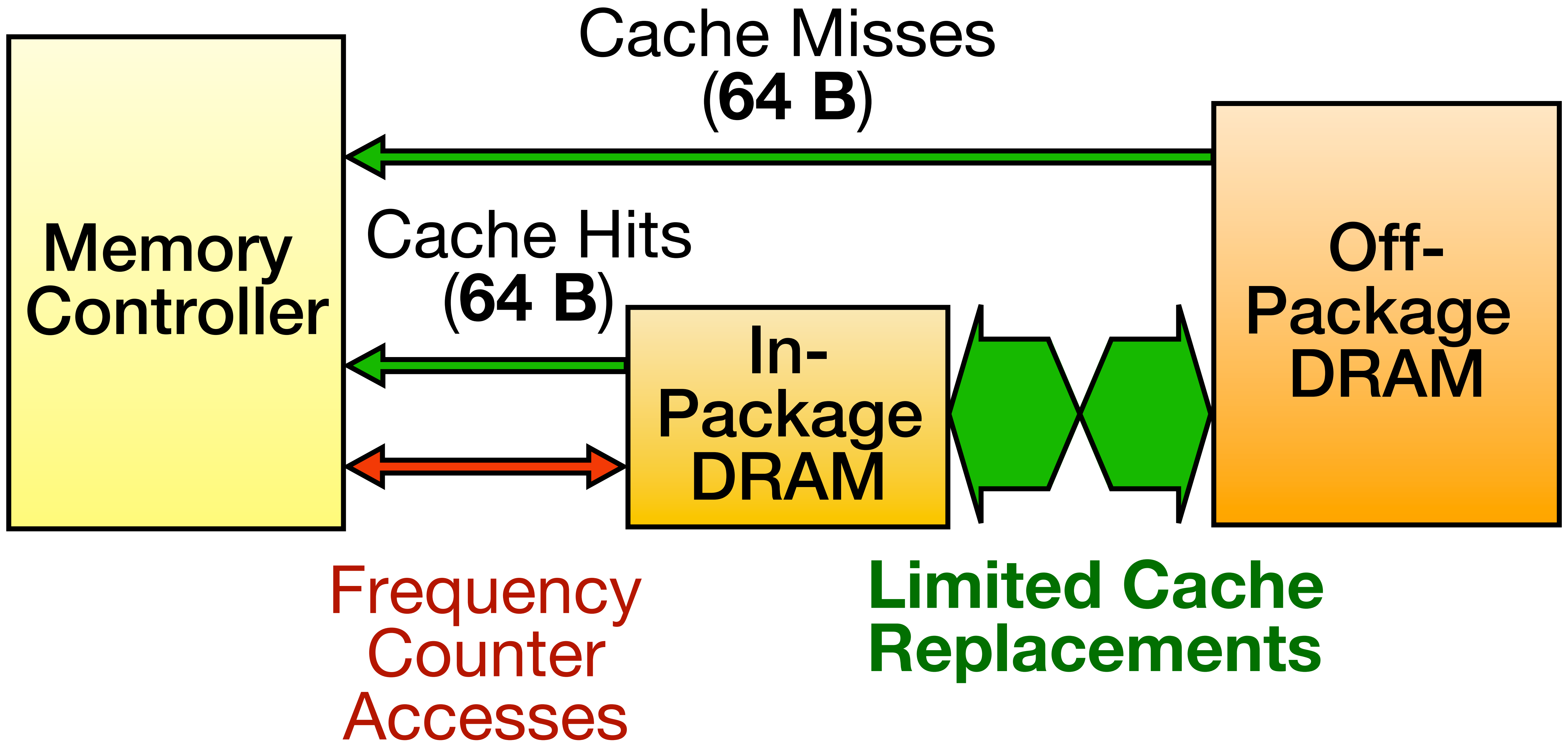
Idea 2: Bandwidth-Aware Cache Replacement

- **DRAM cache replacement incurs significant DRAM traffic**
  - Cache replacement traffic
  - Metadata traffic (e.g., frequency counter lookups/updates)



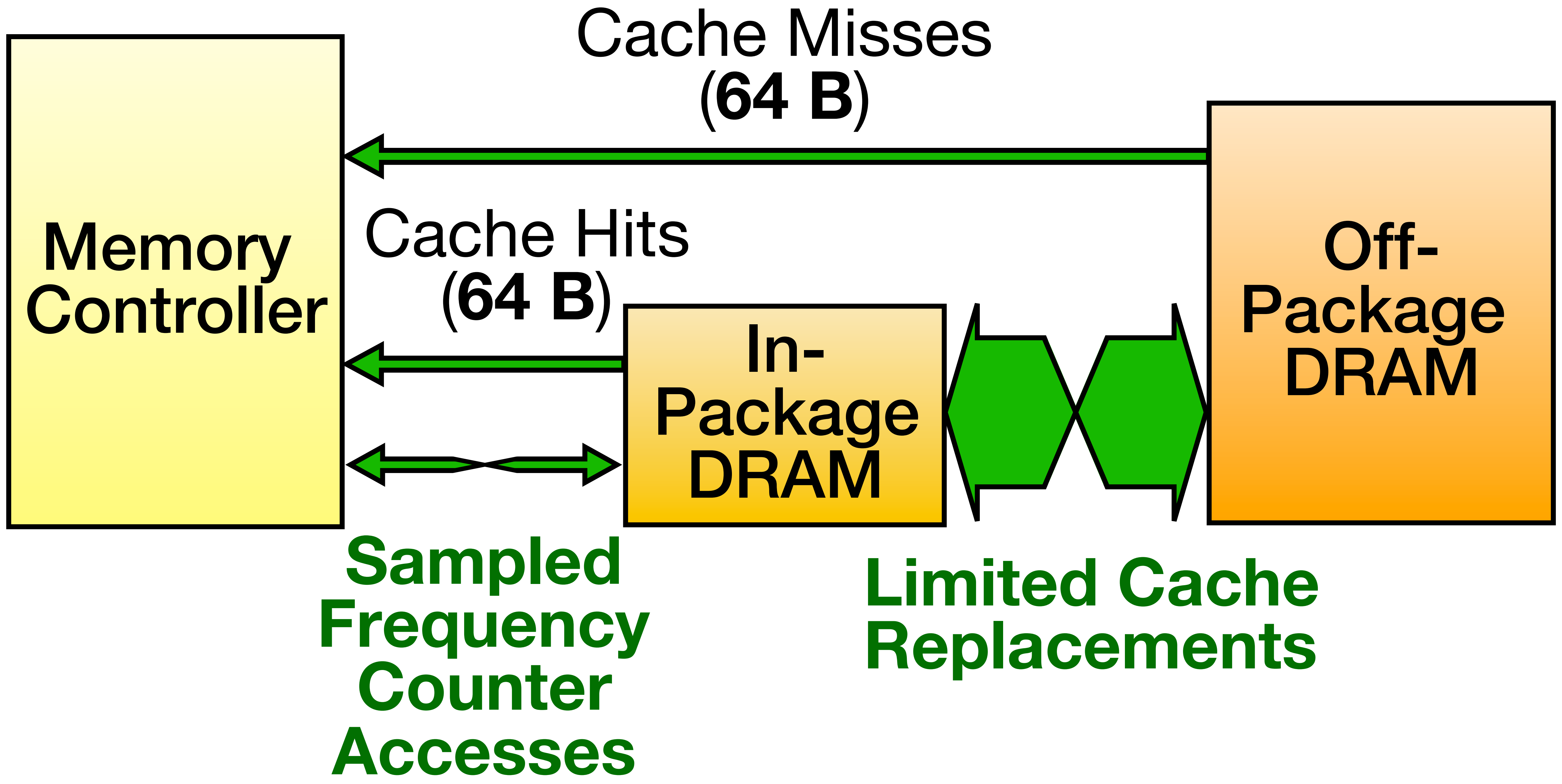
Idea 2: Bandwidth-Aware Cache Replacement

- **DRAM cache replacement incurs significant DRAM traffic**
- **Limit cache replacement rate**
  - Replace only when the incoming page's frequency counter is greater than the victim pages's counter by a threshold



Idea 2: Bandwidth-Aware Cache Replacement

- **DRAM cache replacement incurs significant DRAM traffic**
- **Limit cache replacement rate**
- **Reduce metadata traffic**
  - Access frequency counters for a **randomly sampled** fraction of memory accesses



Banshee Extensions

- **Supporting large pages (e.g., 2MB)**
  - A large page is cached either in its entirety or not at all

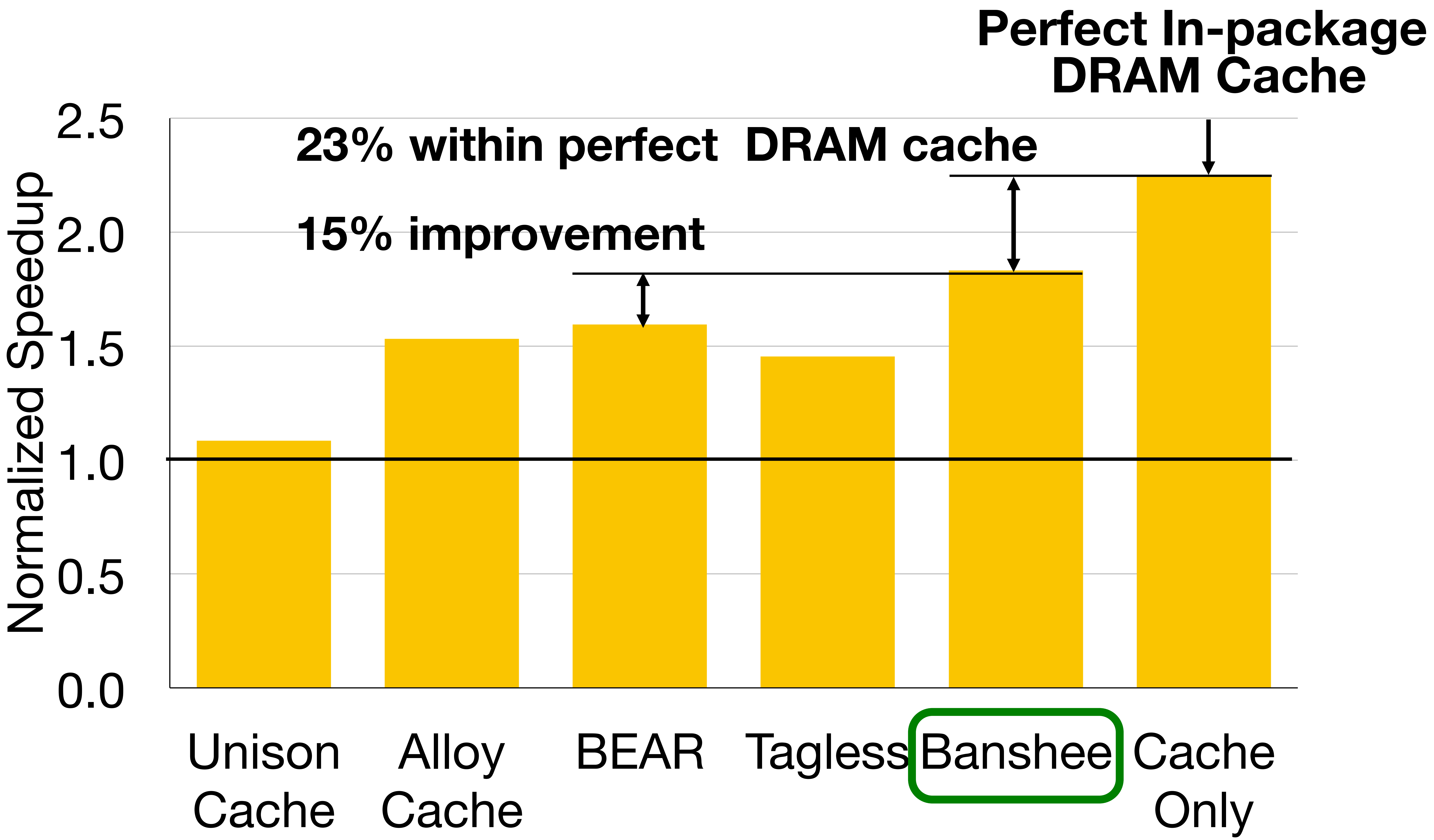
- **Supporting multi-socket processors**
  - Coherent DRAM caches
  - Partitioned DRAM caches

Performance Evaluation

- ZSim simulator [1]
- 16 cores (4-issue, out-of-order, 2.7 GHz)
- In-package DRAM (**1 GB, 84 GB/s**)
- Off-package DRAM (**21 GB/s**)
- Tag Buffer
  - One Tag Buffer per memory controller (MC)
  - 1024 entries, 5 KB in size

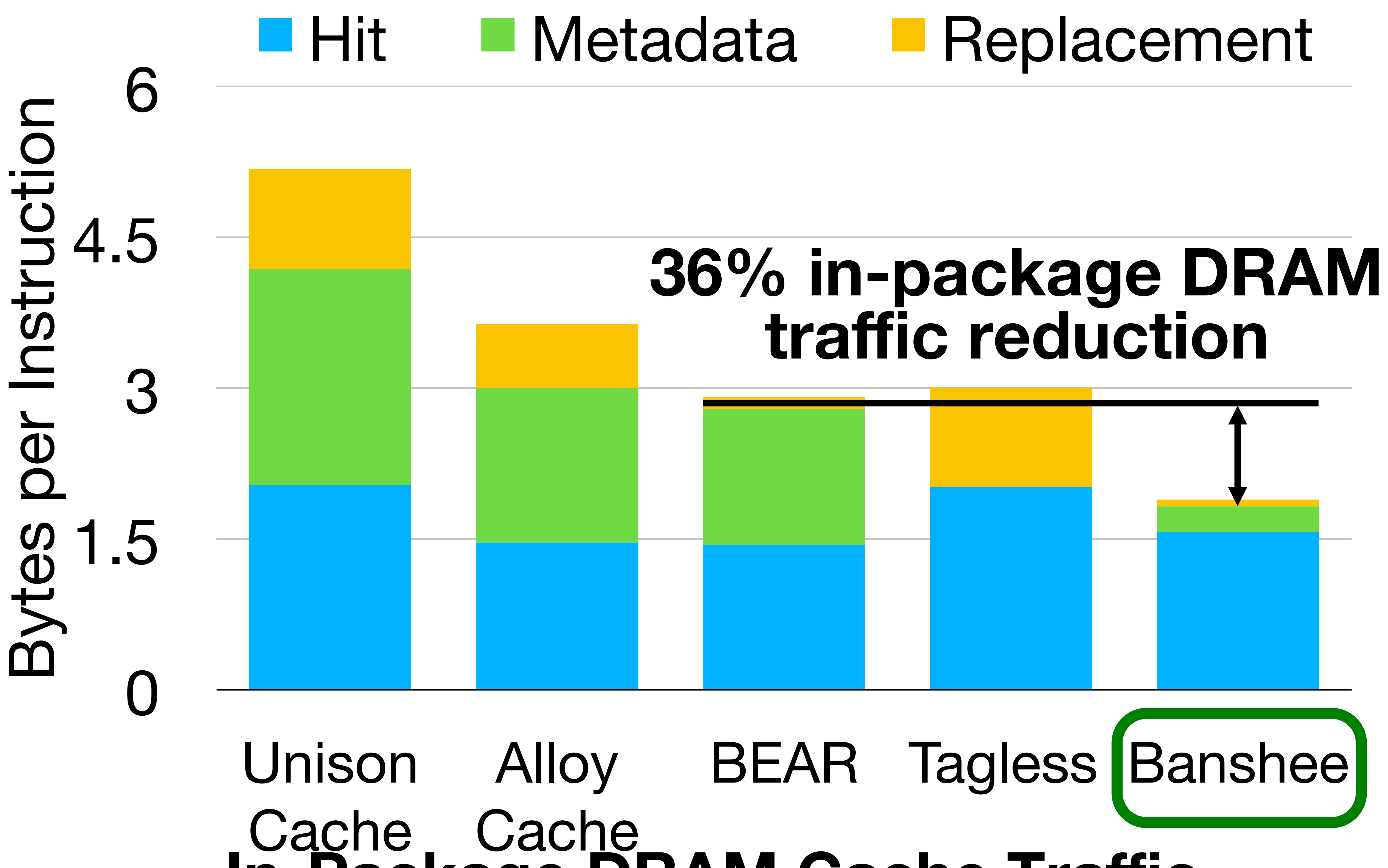
[1] Sanchez, Daniel, and Christos Kozyrakis. "ZSim: fast and accurate microarchitectural simulation of thousand-core systems." *ISCA*, 2013.

Speedup (Normalized to off-package DRAM only)



- Banshee improves performance by **15%** on average over the best-previous (i.e., BEAR) latency-optimized DRAM cache design

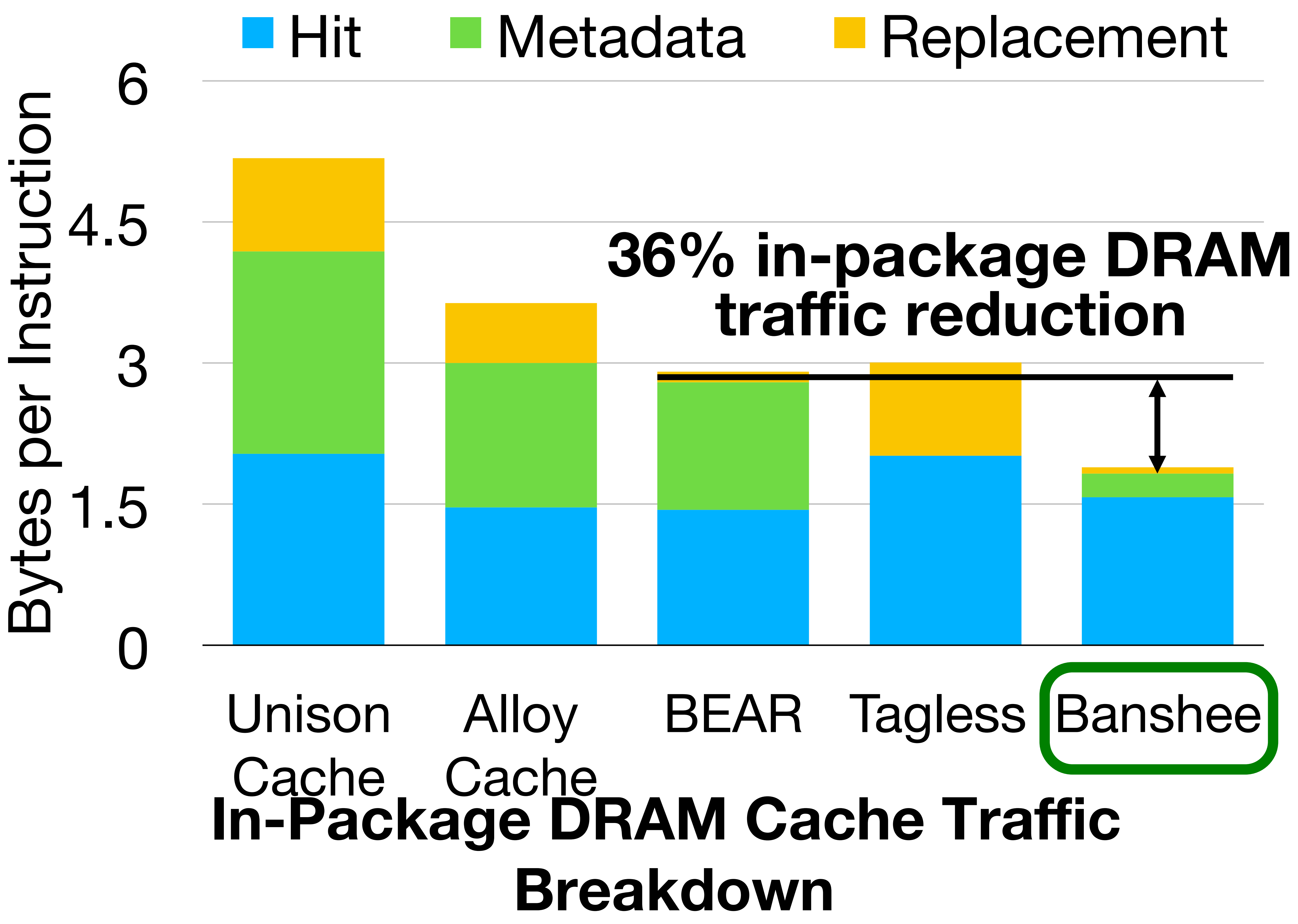
DRAM Bandwidth Efficiency



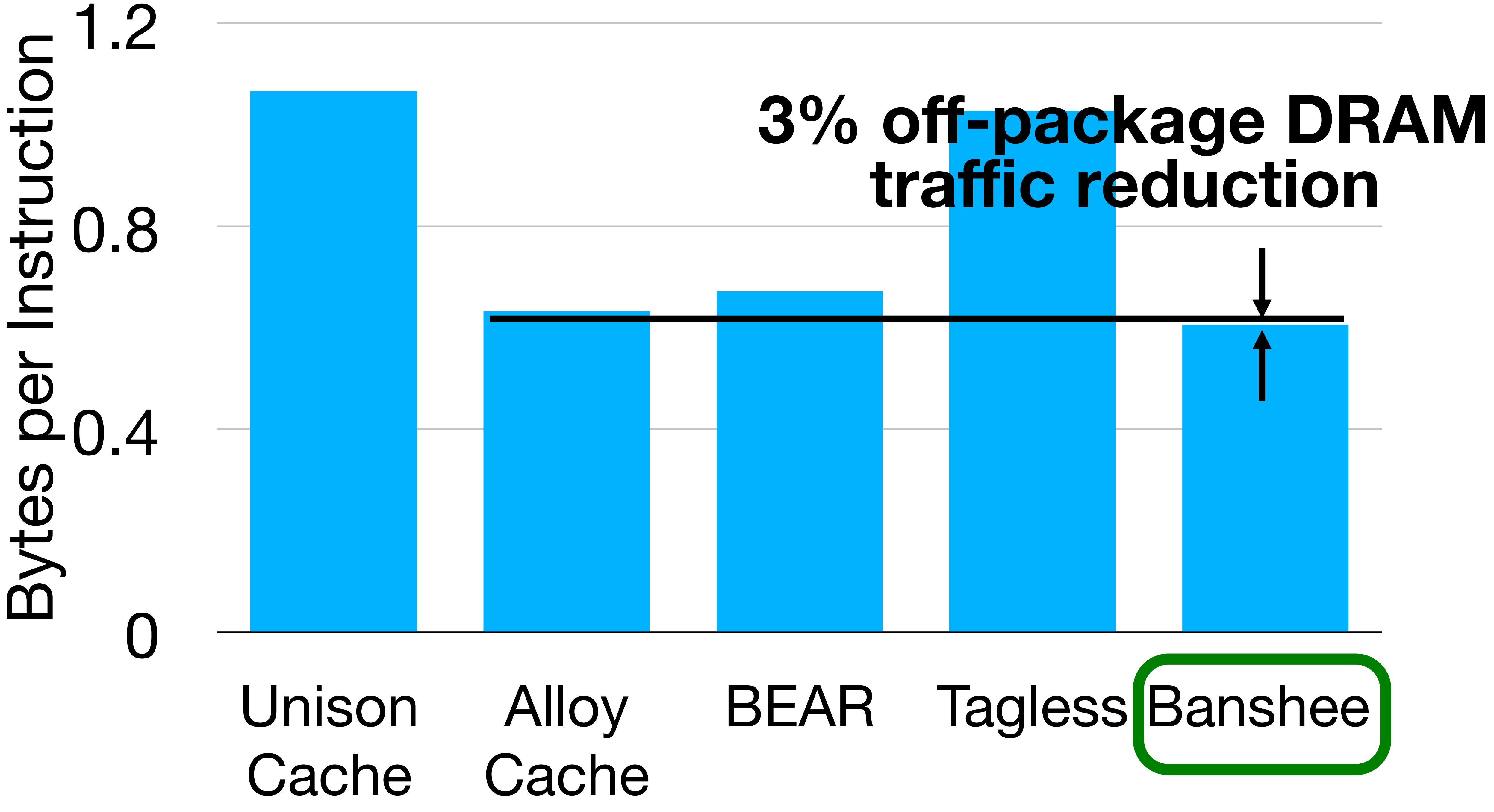
In-Package DRAM Cache Traffic Breakdown

- Banshee reduces **36%** in-package DRAM traffic over the best-previous design

DRAM Bandwidth Efficiency



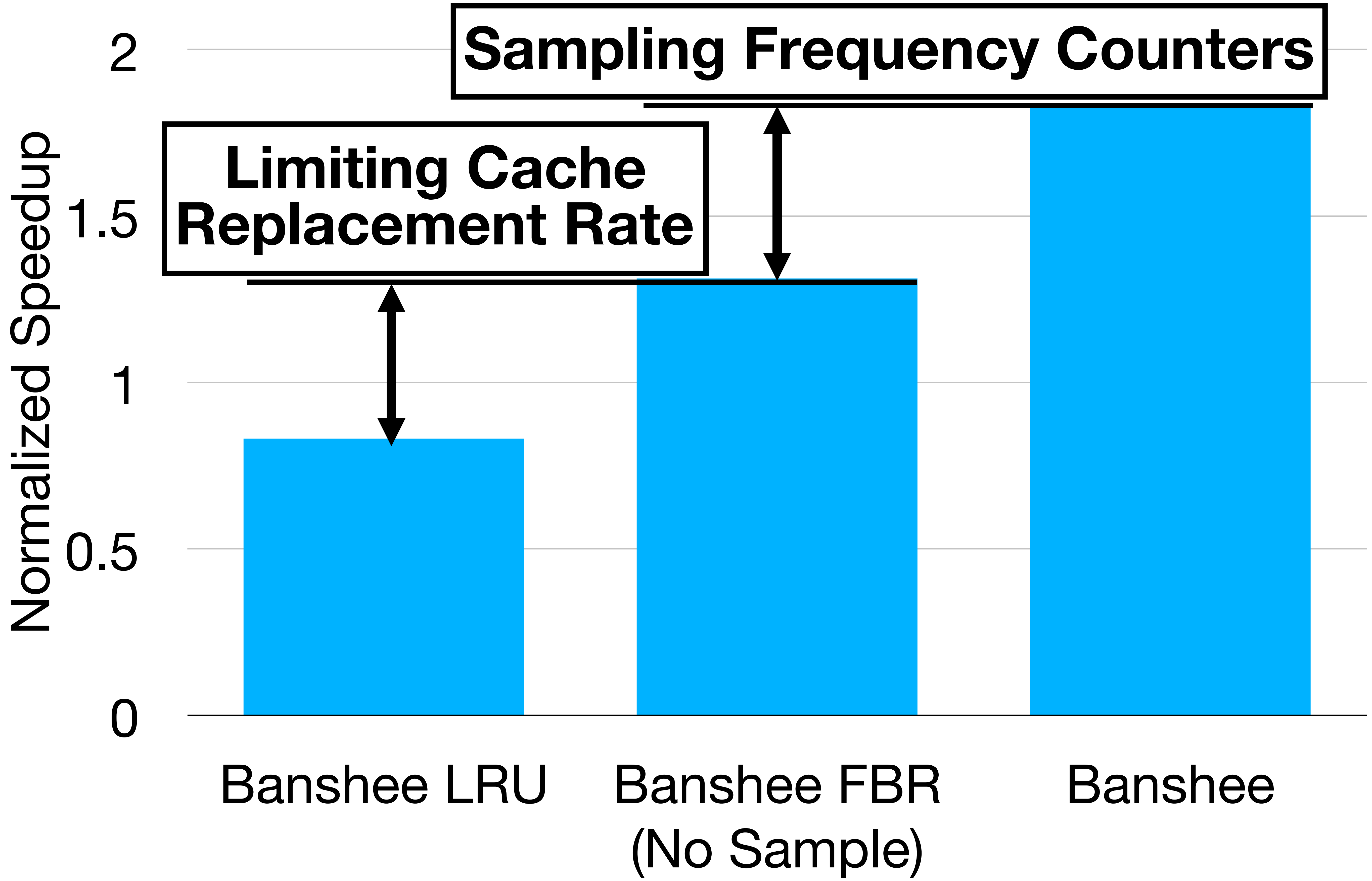
In-Package DRAM Cache Traffic Breakdown



Off-Package DRAM Traffic

- Banshee reduces **36%** in-package DRAM traffic over the best-previous design
- Banshee reduces **3%** off-package DRAM traffic over the best-previous design

Effect of Replacement Traffic Reduction



- **Limiting replacement rate** and **sampling frequency counters** are both important for bandwidth efficiency in Banshee

More Analysis in the Paper

- Performance with large (2 MB) pages
- Balancing in- and off-package DRAM bandwidth
- Overhead for page table update and TLB coherence
- Storing tags in SRAM
- Sweep DRAM cache latency and bandwidth
- Sampling coefficient
- DRAM cache associativity

Summary

- Need to optimize for bandwidth efficiency to fully exploit the performance of in-package DRAM
- **Idea 1: Improving page-table-based DRAM cache designs with efficient Translation Lookaside Buffer (TLB) coherence**
- **Idea 2: Bandwidth-aware frequency-based replacement (FBR) policy**
- Banshee improves performance by **15%** and reduces in-package DRAM traffic by **36%** over the best-previous latency-optimized DRAM cache design

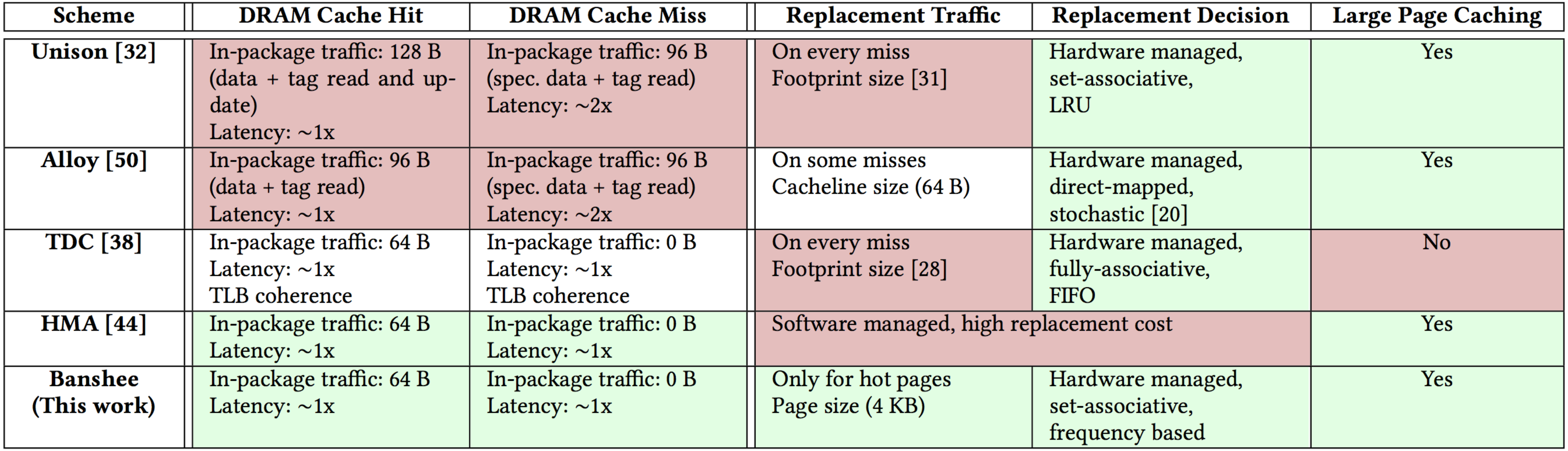
Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation

Xiangyao Yu[1], Christopher Hughes[2], Nadathur Satish[2], Onur Mutlu[3], Srinivas Devadas[1]

[1]MIT [2]Intel Labs [3]ETH Zürich

Backup Slides

Summary of Operational Characteristics of Different State-of-the-Art DRAM Cache Designs



Scheme	DRAM Cache Hit	DRAM Cache Miss	Replacement Traffic	Replacement Decision	Large Page Caching
Unison [32]	In-package traffic: 128 B (data + tag read and update) Latency: ~1x	In-package traffic: 96 B (spec. data + tag read) Latency: ~2x	On every miss Footprint size [31]	Hardware managed, set-associative, LRU	Yes
Alloy [50]	In-package traffic: 96 B (data + tag read) Latency: ~1x	In-package traffic: 96 B (spec. data + tag read) Latency: ~2x	On some misses Cacheline size (64 B)	Hardware managed, direct-mapped, stochastic [20]	Yes
TDC [38]	In-package traffic: 64 B Latency: ~1x TLB coherence	In-package traffic: 0 B Latency: ~1x TLB coherence	On every miss Footprint size [28]	Hardware managed, fully-associative, FIFO	No
HMA [44]	In-package traffic: 64 B Latency: ~1x	In-package traffic: 0 B Latency: ~1x	Software managed, high replacement cost		Yes
Banshee (This work)	In-package traffic: 64 B Latency: ~1x	In-package traffic: 0 B Latency: ~1x	Only for hot pages Page size (4 KB)	Hardware managed, set-associative, frequency based	Yes

Tag Buffer Organization

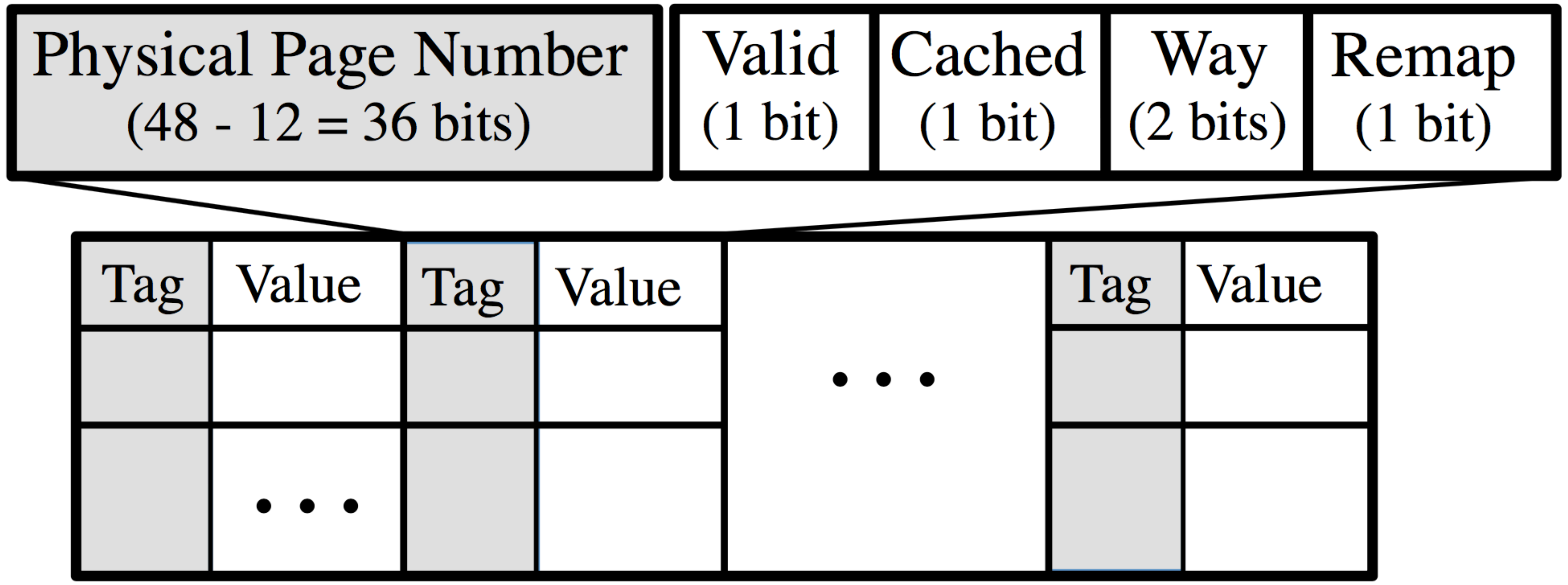
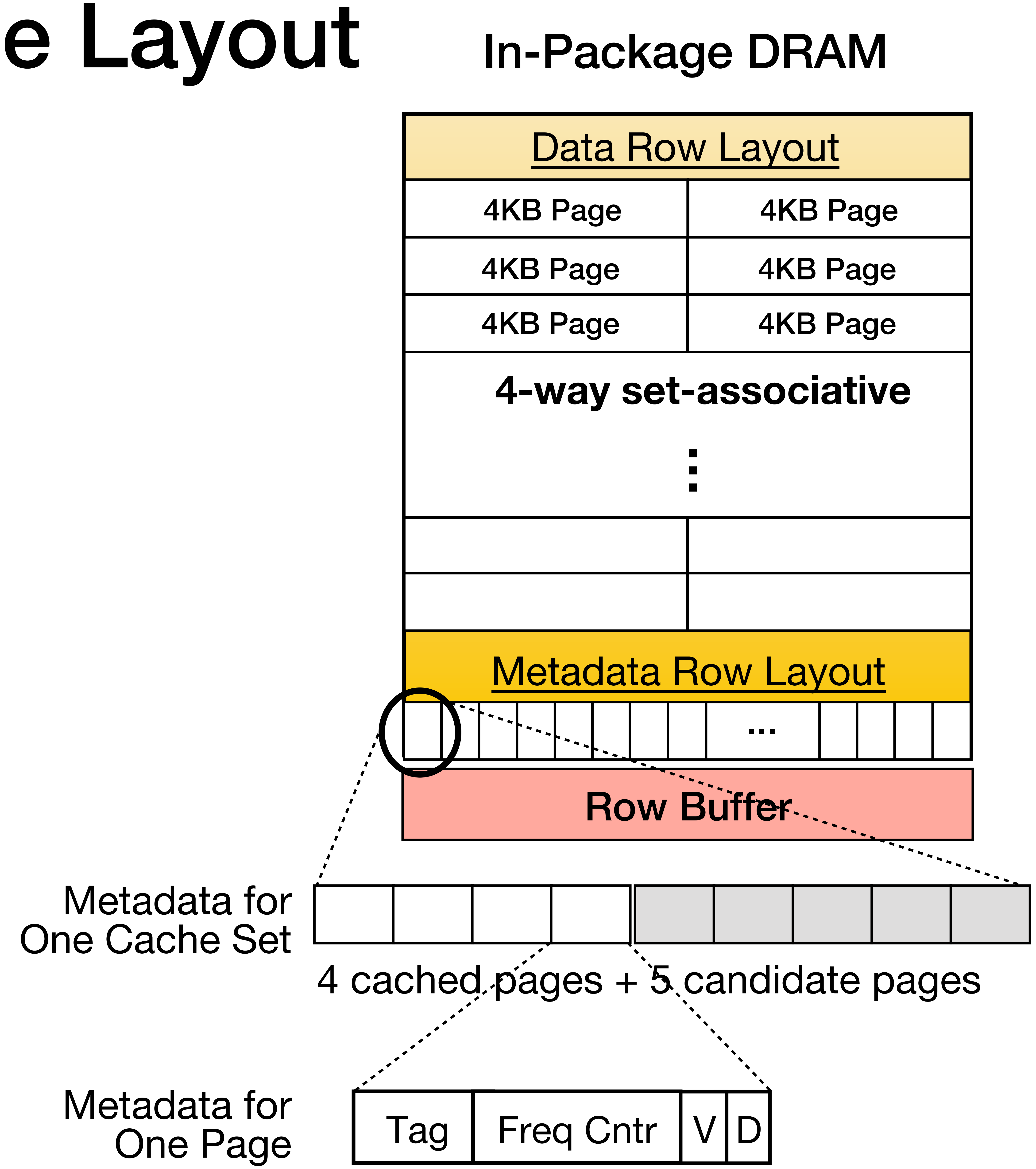


Figure 2: Tag Buffer Organization – The Tag Buffer is organized as a set-associative cache. The DRAM is 4-way set-associative in this example.

DRAM Cache Layout



Speedup Normalized to NoCache

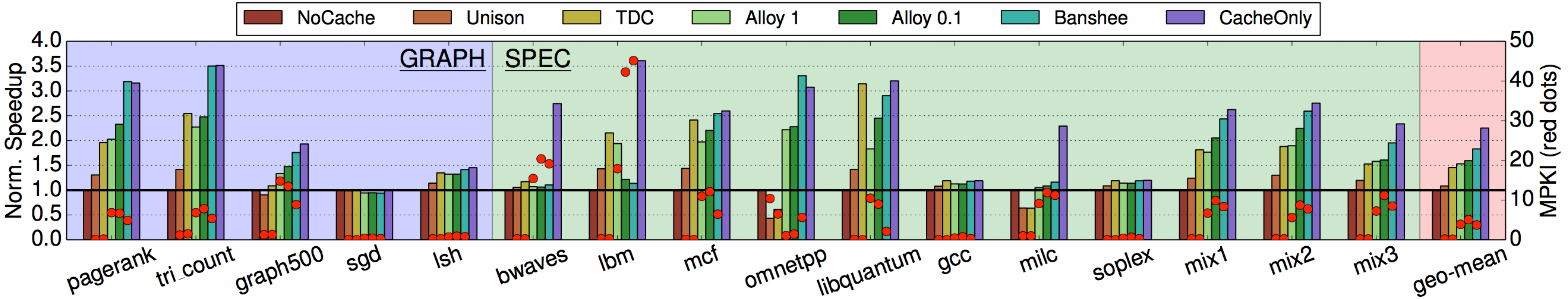


Figure 4: Speedup Normalized to NoCache – Speedup is shown in bars and misses per kilo instruction (MPKI) is shown in red dots.

In-Package DRAM Traffic Breakdown

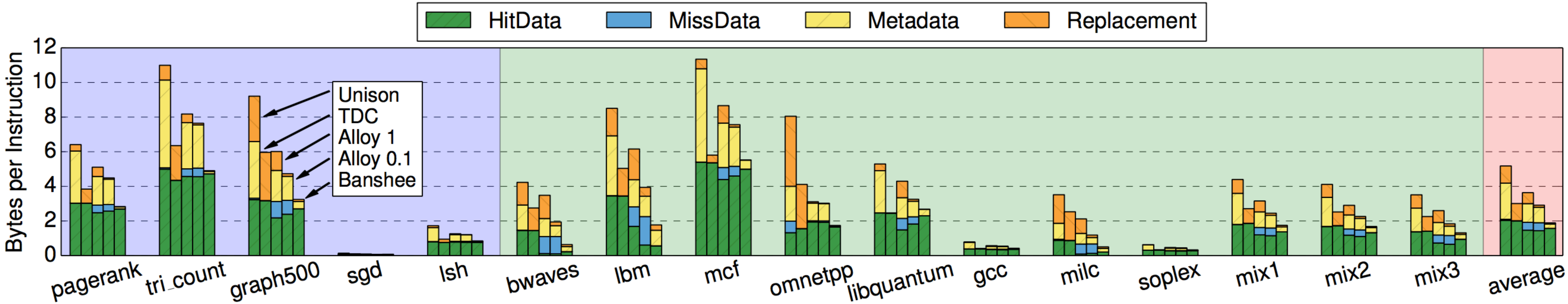


Figure 5: In-package DRAM Traffic Breakdown.

Off-Package DRAM Traffic

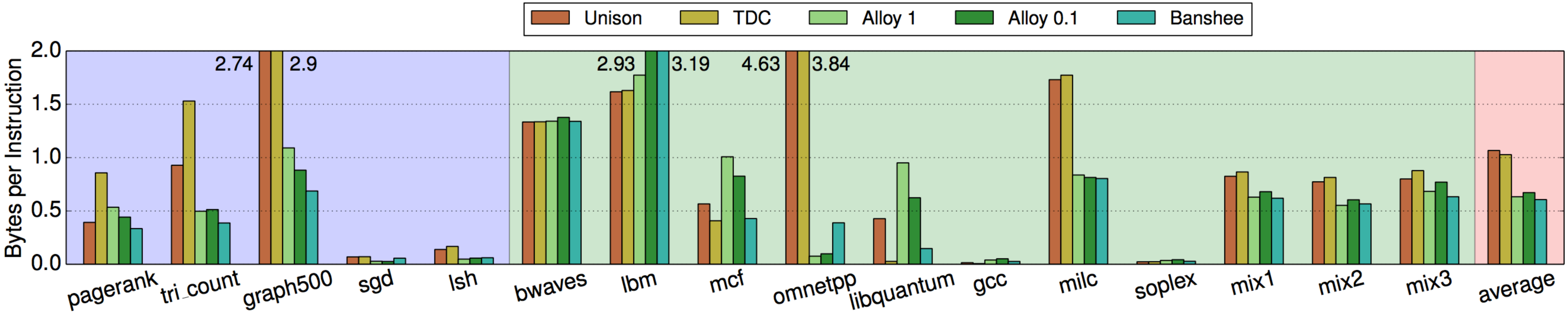


Figure 6: Off-package DRAM Traffic.

Sensitivity to Page Table Update Cost

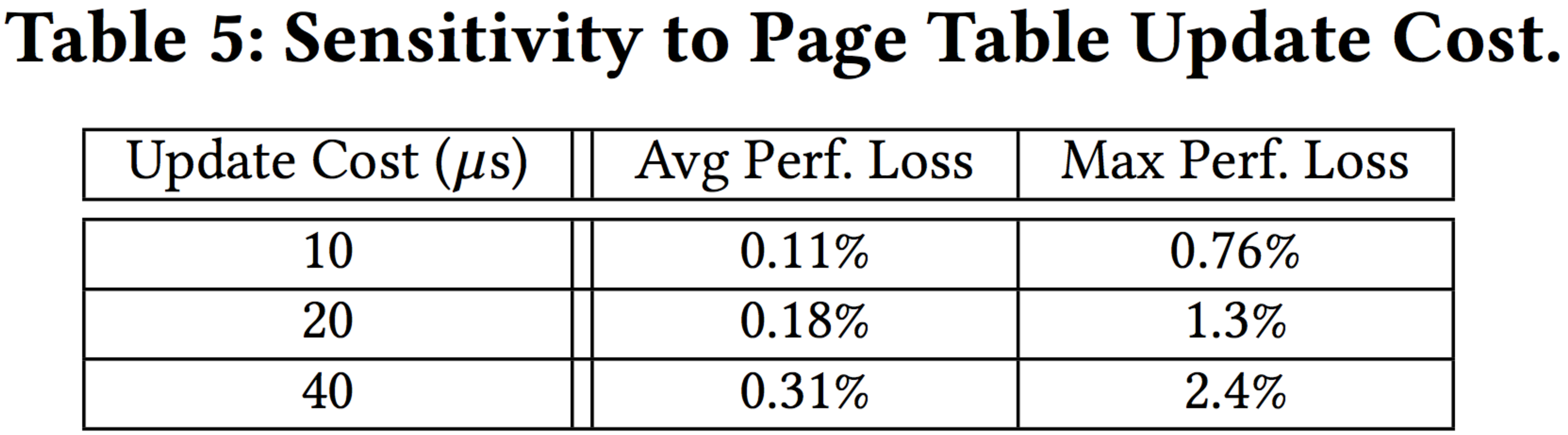


Table 5: Sensitivity to Page Table Update Cost.

Update Cost (μs)	Avg Perf. Loss	Max Perf. Loss
10	0.11%	0.76%
20	0.18%	1.3%
40	0.31%	2.4%

Sensitivity to DRAM Cache Latency and Bandwidth

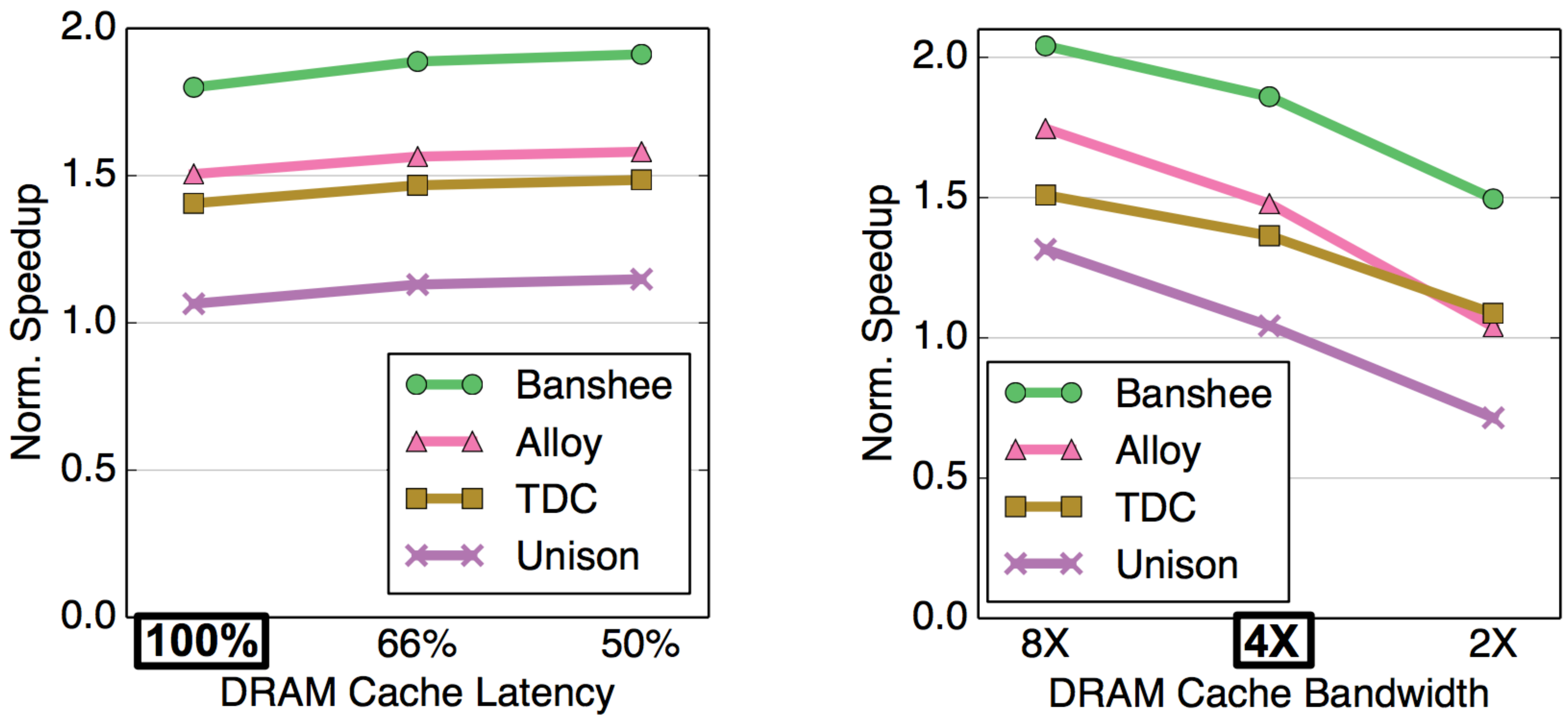


Figure 8: Sensitivity to DRAM Cache Latency and Bandwidth – Each data point is the geometric mean over all benchmarks. Default parameter setting is highlighted on x-axis. Latency and bandwidth values are relative to off-package DRAM.