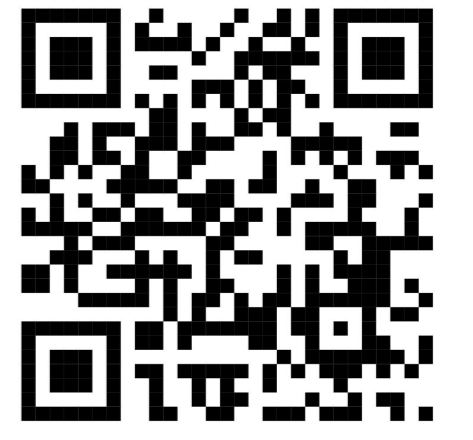# Leveraging Cache Coherence to Detect and Repair False Sharing On-the-fly

Vipin Patel, Swarnendu Biswas, Mainak Chaudhuri (Indian Institute of Technology Kanpur)

Scan for artifacts

Poster Flow: Top to Bottom, Left to Right

False Sharing in Real World Applications

Bug #87350 CACHE_LINE
Seeing through hardware counters: a journey to threefold performance increase

Netflix Technology Blog · Follow

secondary_super_cache does not scale well

Dates:
Created: 2017-05-16
Updated: 2024-09-16
Resolved: 2024-04-16

Cache Contention due to False Sharing
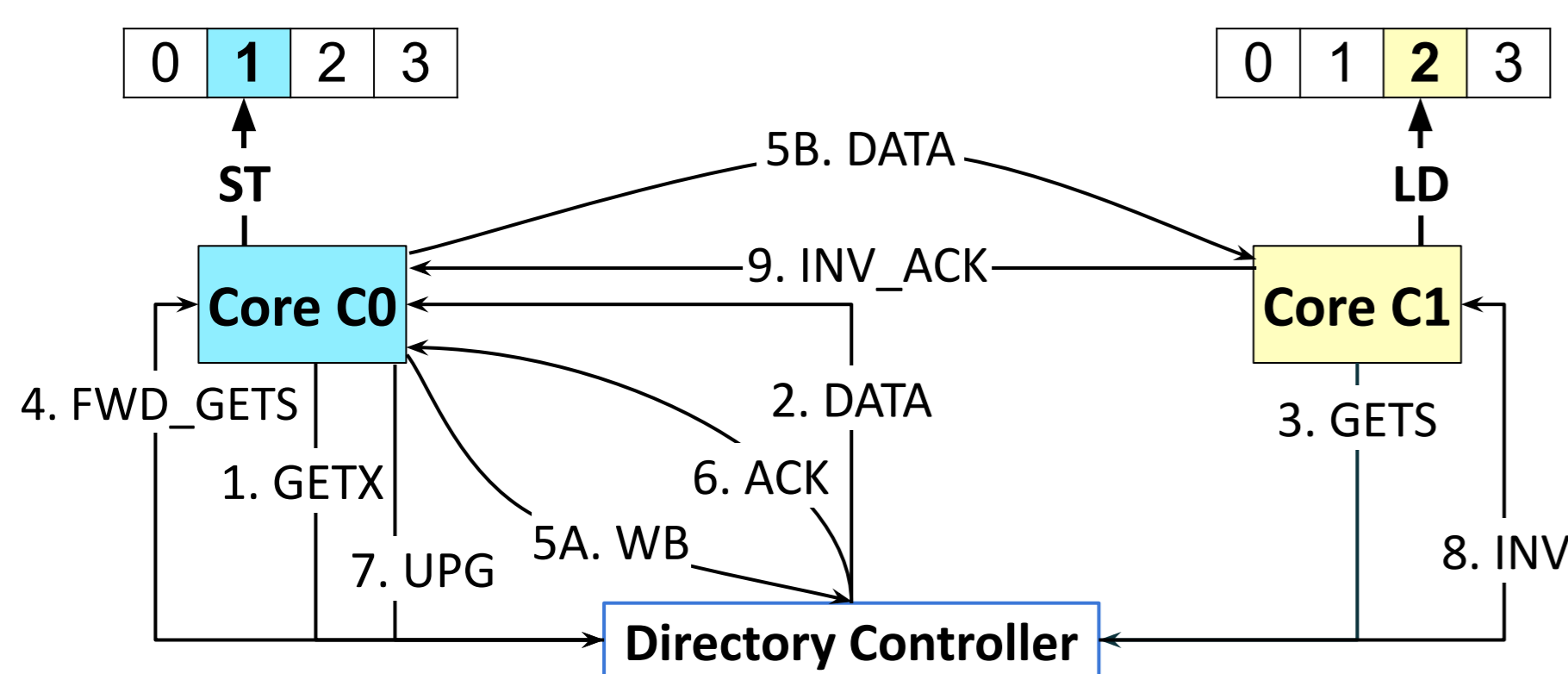
- **Degrades** performance
- **Inflates** interconnect traffic
- **Increases** energy consumption

Speedup by Manually Fixing False Sharing
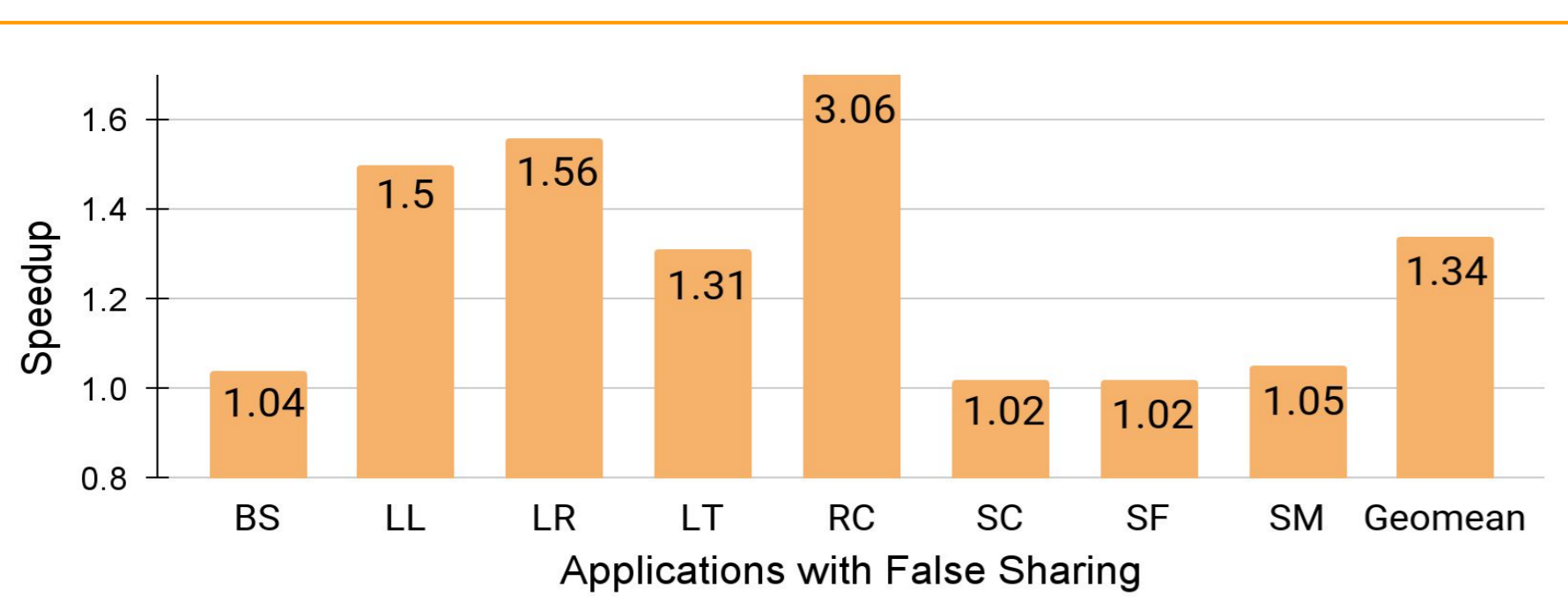
Avg Speedup **1.34x** and Max speedup of **3.06x**

Savings: **84%** interconnect traffic and **25%** in energy consumption

Challenges in False Sharing Elimination

- ❖ Heap Organization[1]
- ❖ Cache Organization
- ❖ Compiler Optimization[2]
- ❖ Application Language
- ❖ Runtime Environment

```
struct {
  pthread_t tid; POINT_T*
  points;
  int num_elems; long long SX;
  long long SY; long long SXX;
  long long SXY; long long SYY;
} lreg_args;[2]
```
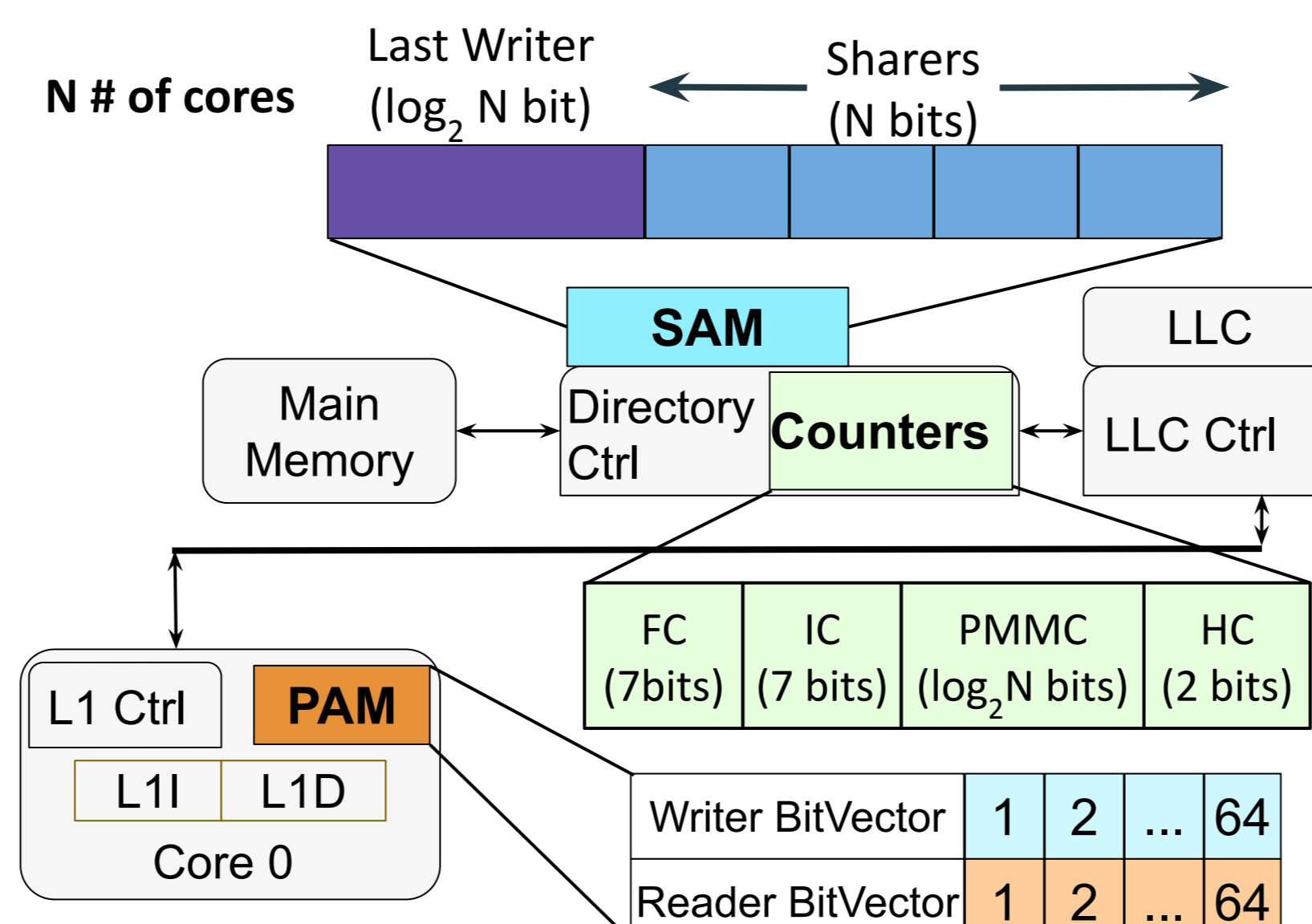
Shortcomings of the Existing Approaches

- ❖ Extending support to every language
- ❖ Inflation of memory footprint
- ❖ OS and Kernel modification[3]
- ❖ Cache organization modification
- ❖ Access to application source code
- ❖ Introduction of additional instructions

PORTABLE | EFFICIENT
FSDetect and FSLite
ACCURATE | EFFECTIVE
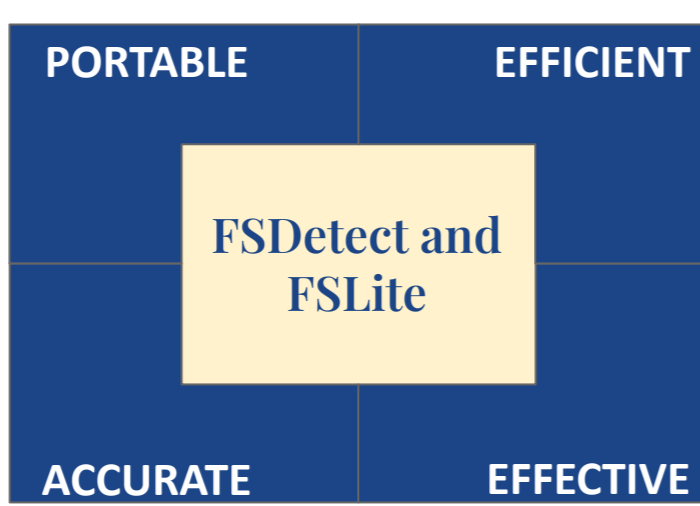
## Key Insight of Our Protocol Extensions

FSDetect
- ● Tracks the sharing patterns of block
- ● Identifies impactful instance of false sharing

FSLite
- ● Eliminates false sharing by **privatizing** the blocks
- ● Allows multiple writers for the disjoint bytes

Overview of Architectural Modification

N # of cores

Last Writer ($\log_2$ N bit) | Sharers (N bits)

SAM | LLC
Main Memory | Directory Ctrl | Counters | LLC Ctrl

L1 Ctrl | PAM
L1I | L1D
Core 0

FC (7bits)	IC (7 bits)	PMMC ($\log_2$N bits)	HC (2 bits)

	1	2	...	64
Writer BitVector	1	2	...	64
Reader BitVector	1	2	...	64

- ❖ Shared Access Metadata (SAM) to track the access history
- ❖ Private Access Metadata (PAM) to track the local access
- ❖ Fetch (FC) and Invalidation (IC) counters to filter the impactful instance
- ❖ Pending Metadata Message (PMMC) to track the inflight metadata
- ❖ Hysteresis Counter(HC) to prevent frequent trigger and termination of privatization
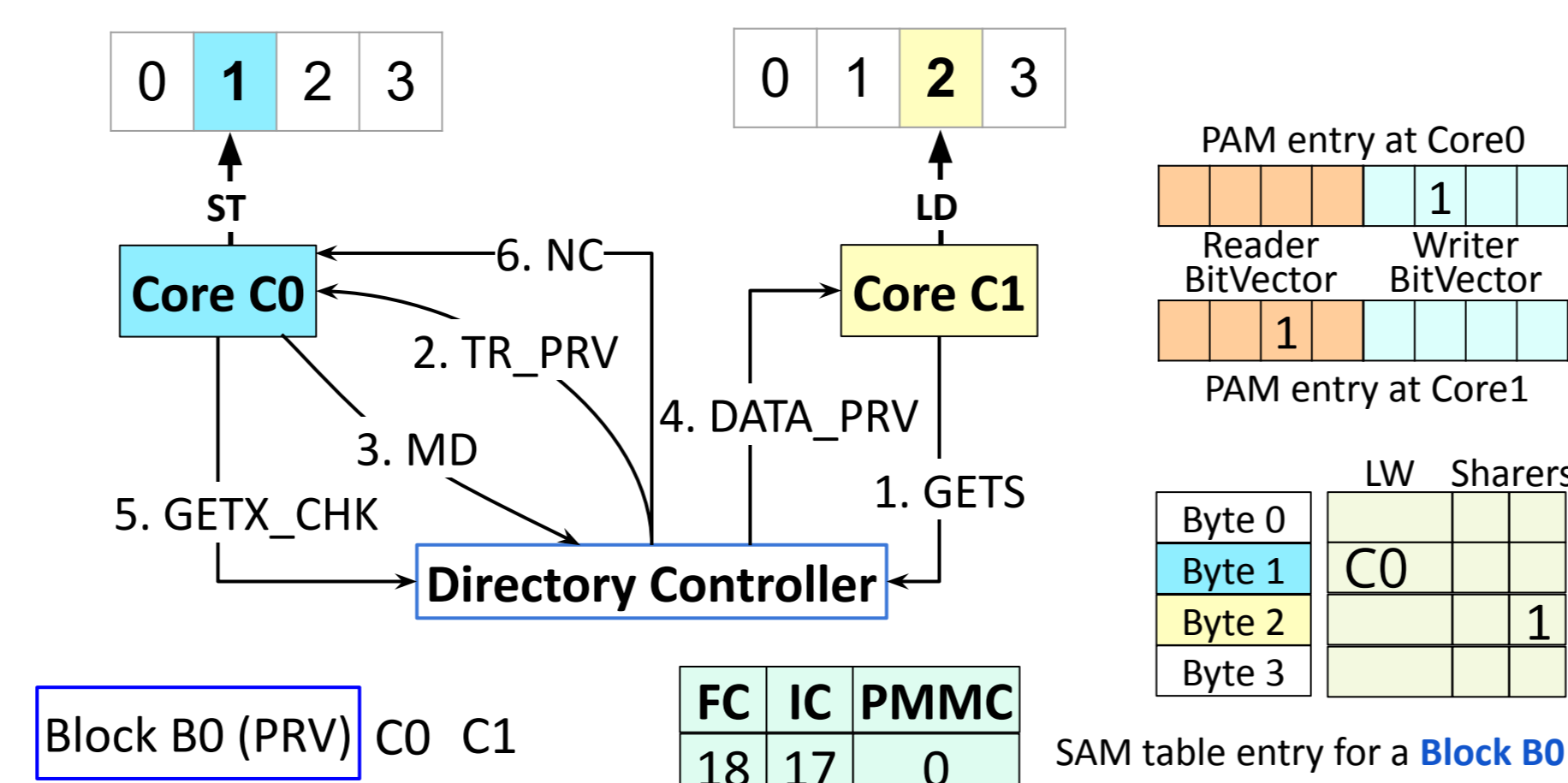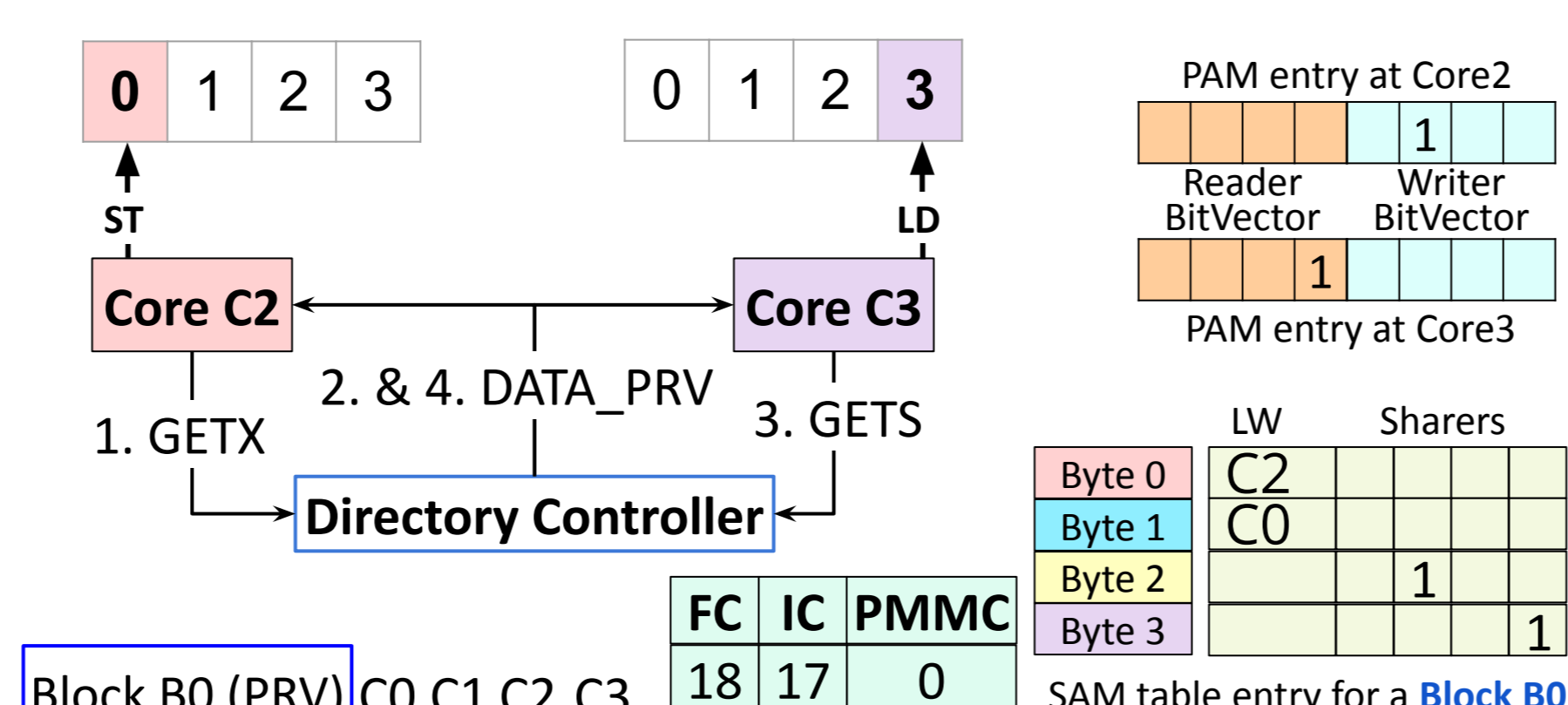
FSDetect: Detection of False Sharing

FC	IC	PMMC
18	17	0

Block B0 (M) C0 C1 — SAM table entry for a **Block B0**
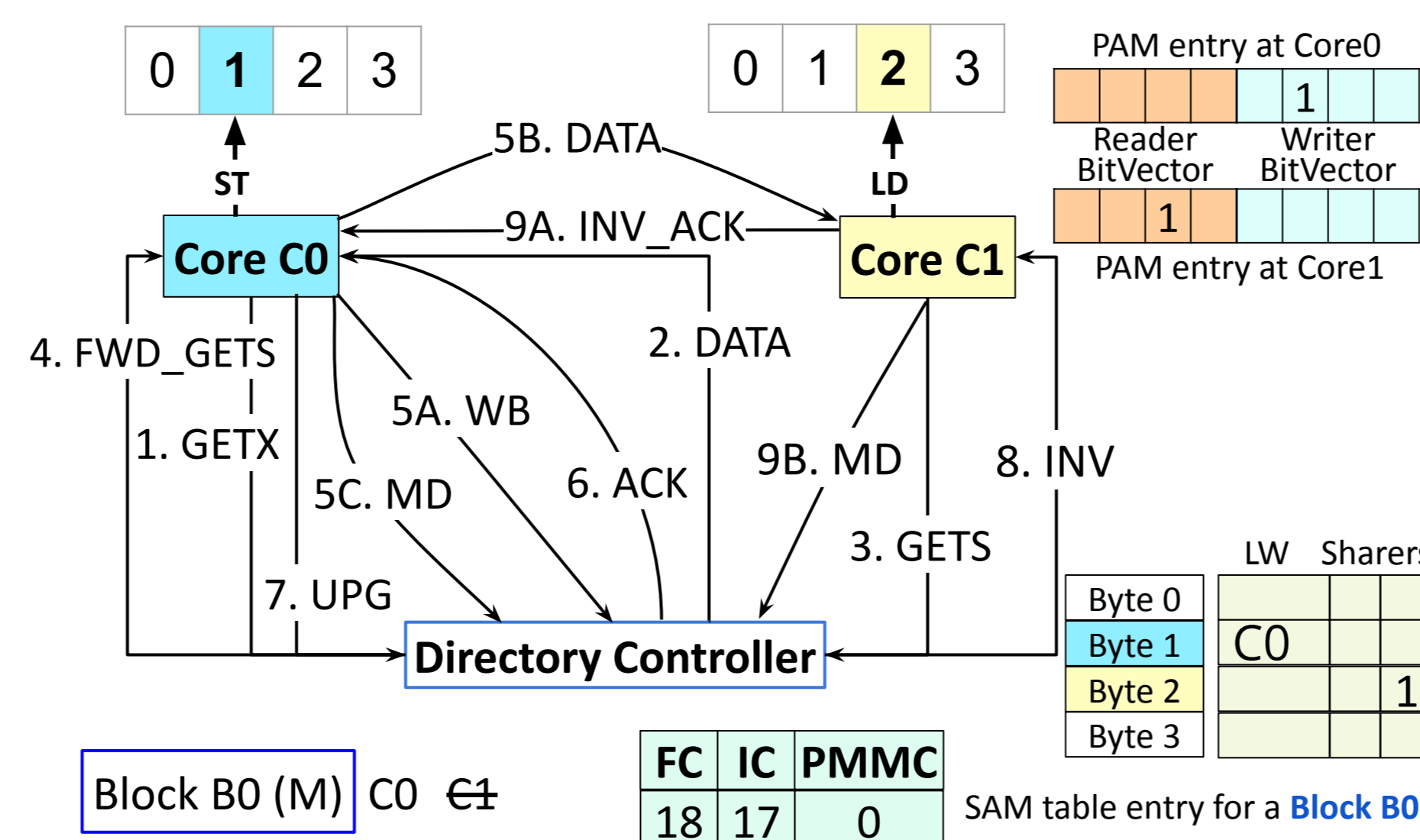
Impactful instance of false sharing

Necessary conditions to trigger Privatization

- ❖ Fetch Counter > τ_p
- ❖ Invalidation Counter > τ_p τ_p: Privatization Threshold
- ❖ Pending Metadata Messages == 0
- ❖ Directory performs a conflict detection check
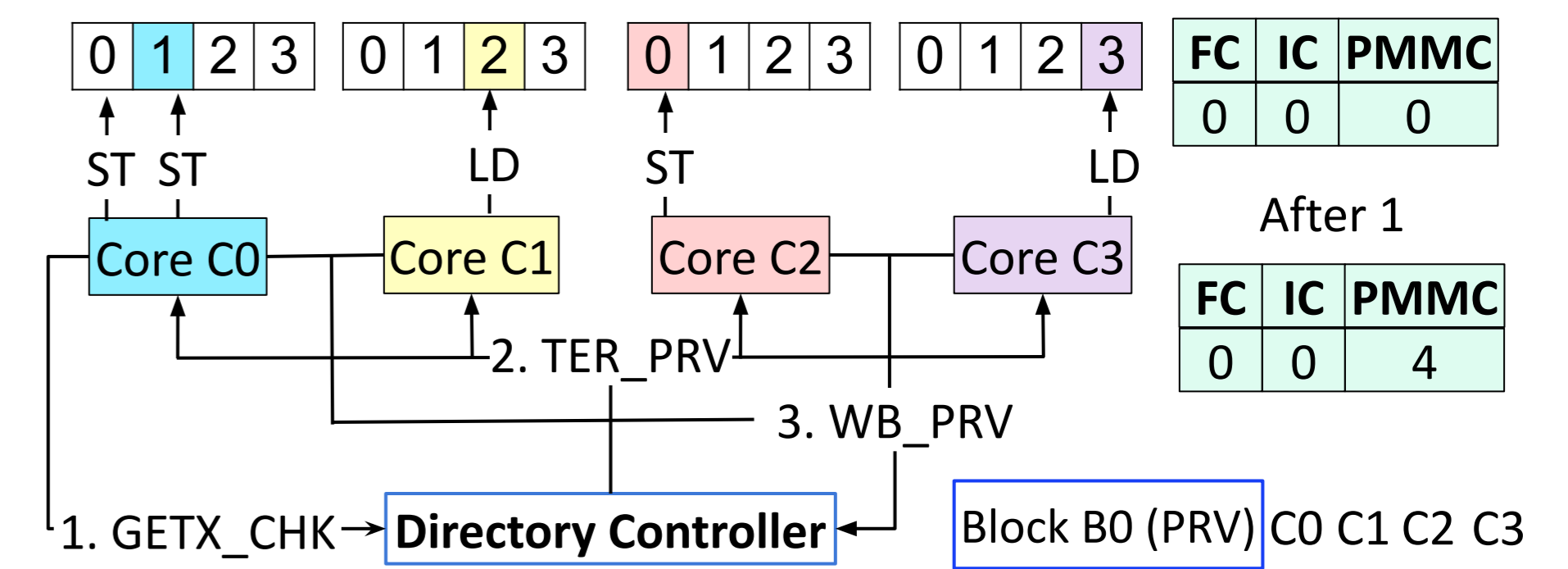- ❖ Access history in SAM has no overlapping accesses

FSLite: Privatization of falsely shared blocks

FC	IC	PMMC
18	17	0

Block B0 (PRV) C0 C1 — SAM table entry for a **Block B0**

Handling Request during Privatization

FC	IC	PMMC
18	17	0

Block B0 (PRV) C0 C1 C2 C3 — SAM table entry for a **Block B0**

Termination of Privatization

ST ST | LD | ST | LD
Core C0 | Core C1 | Core C2 | Core C3

FC	IC	PMMC
0	0	0

After 1

FC	IC	PMMC
0	0	4

1. GETX_CHK → Directory Controller
2. TER_PRV
3. WB_PRV

Block B0 (PRV) C0 C1 C2 C3

- ❖ Core C1 access (store) Byte 1 of the block B0
- ❖ C1 sends an access conflict detection request (GETX_CHK) to the directory
- ❖ Directory identifies the conflict with C2 and sends termination requests (TER_PRV) to all the sharers
- ❖ Each core responds with write back message(WB_PRV) to the directory
- ❖ Directory updates the LLC copy of the block by merging the bytes modified by each core using Write-Mask
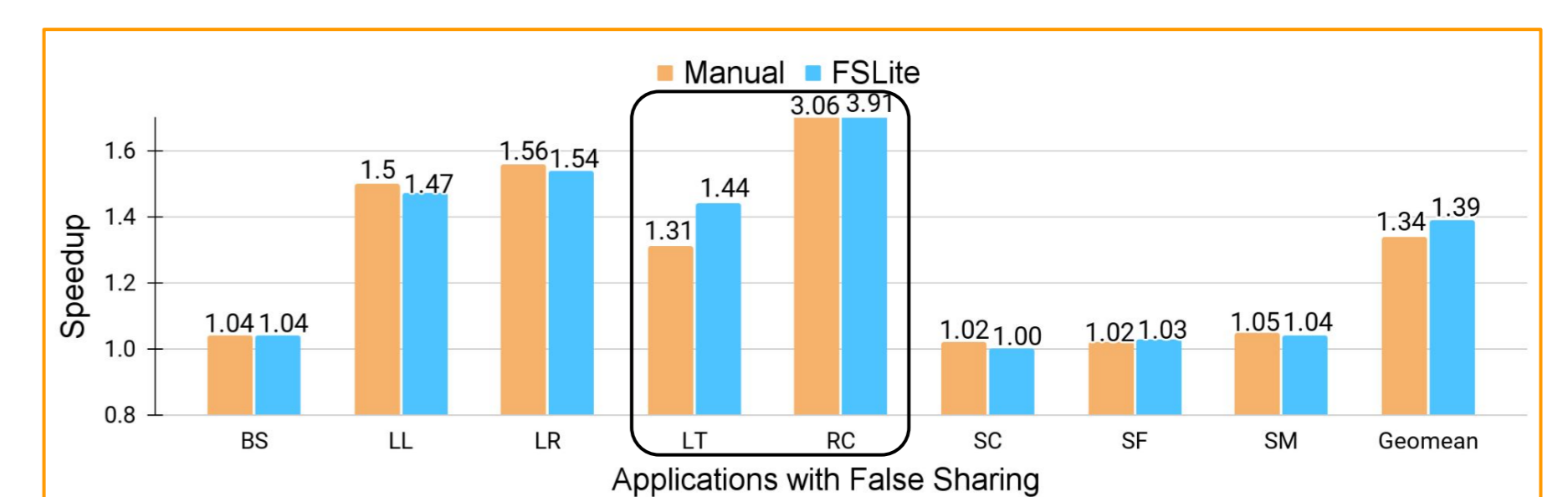
SAM table entry for a **Block B0**

Performance Improvement with FSLite
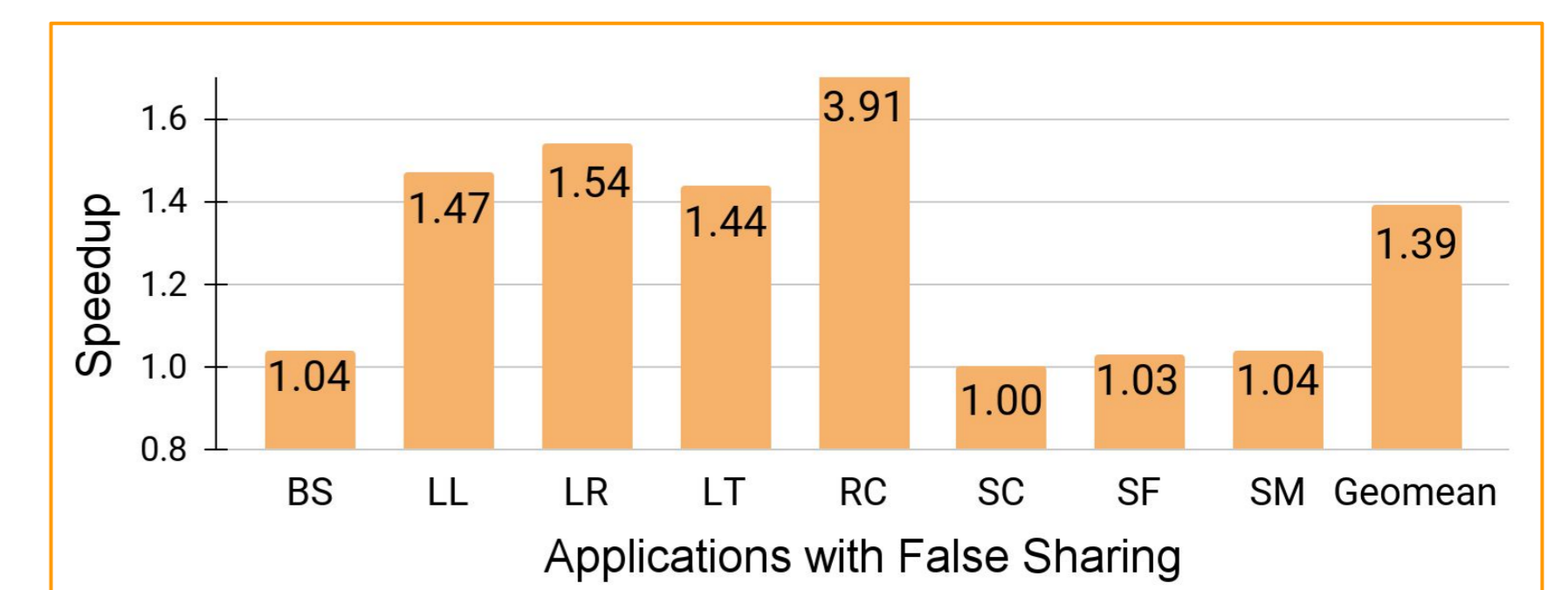
FSLite achieves on avg **1.39x** speedup and a max speedup of **3.91x**

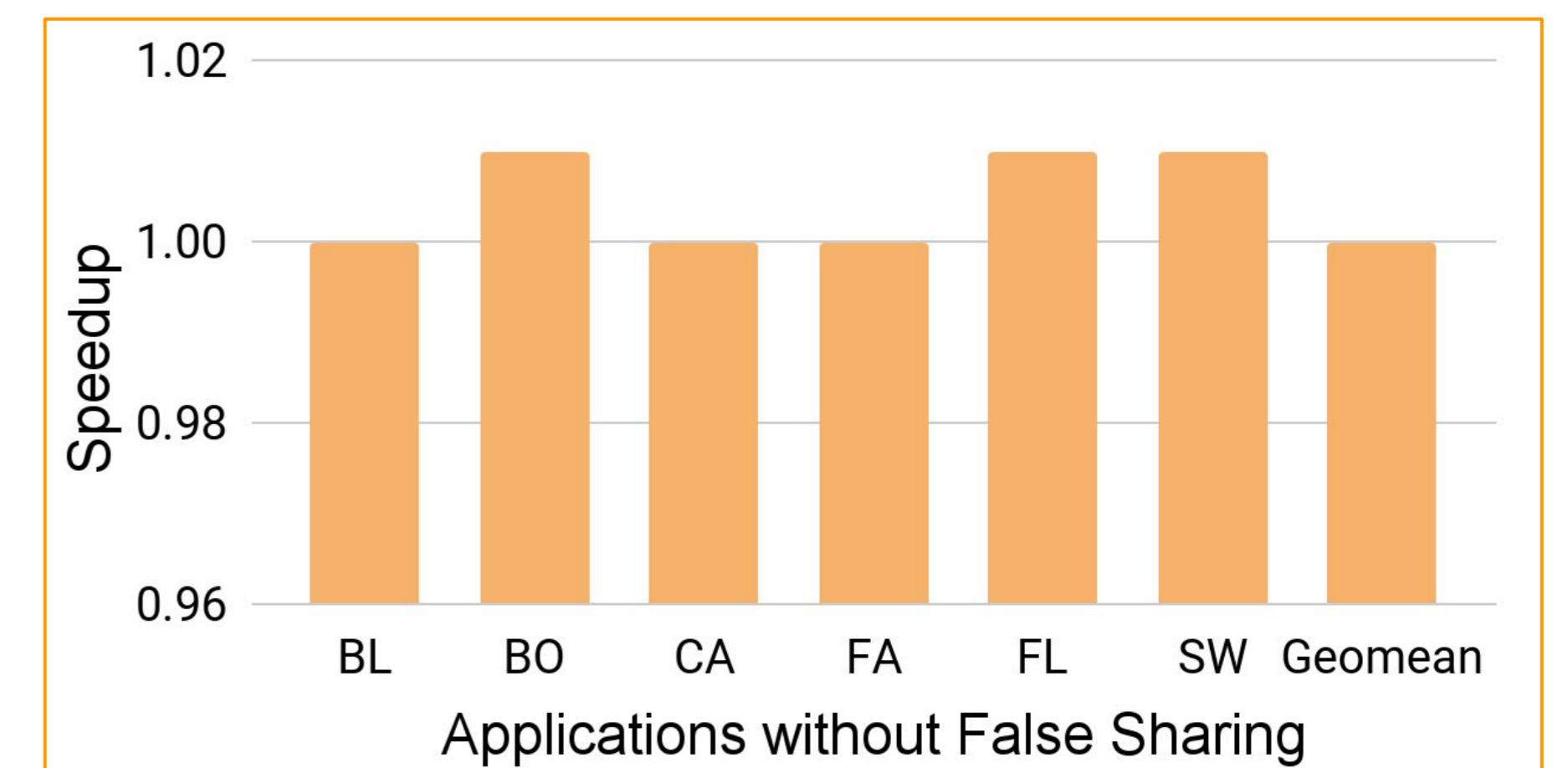**27%** energy savings and **89%** less interconnect messages

Comparison with Manual Fix

Padding **inflates the memory footprint**

Padding **introduces additional instructions**

Impact of FSLite on Applications w/o False Sharing

No slowdown for applications without false sharing

Additional Results

- ❖ FSLite achieves an average speedup of 1.63x for O-o-O cores
- ❖ FSLite achieves an average speedup of 1.21x over a baseline with 128KB L1D cache across all applications

References

1. Emery D. Berger et al. "Hoard: A Scalable Memory Allocator for Multithreaded Applications." ASPLOS'00.
2. Mihir Nanavati et al. "Whose Cache Line Is It Anyway? Operating System Support for Live Detection and Repair of False Sharing." EuroSys'13.
3. Liang Luo et al. "LASER: Light, Accurate Sharing dEtection and Repair" HPCA'16.