Reuse Detector: Improving the management of STT-RAM SLLCs

R. Rodríguez, J. Díaz, F. Castro, P. Ibáñez, D. Chaver, V. Viñals, J.C. Sáez, M. Prieto-Matías, L. Piñuel, T. Monreal, J.M. Llabería

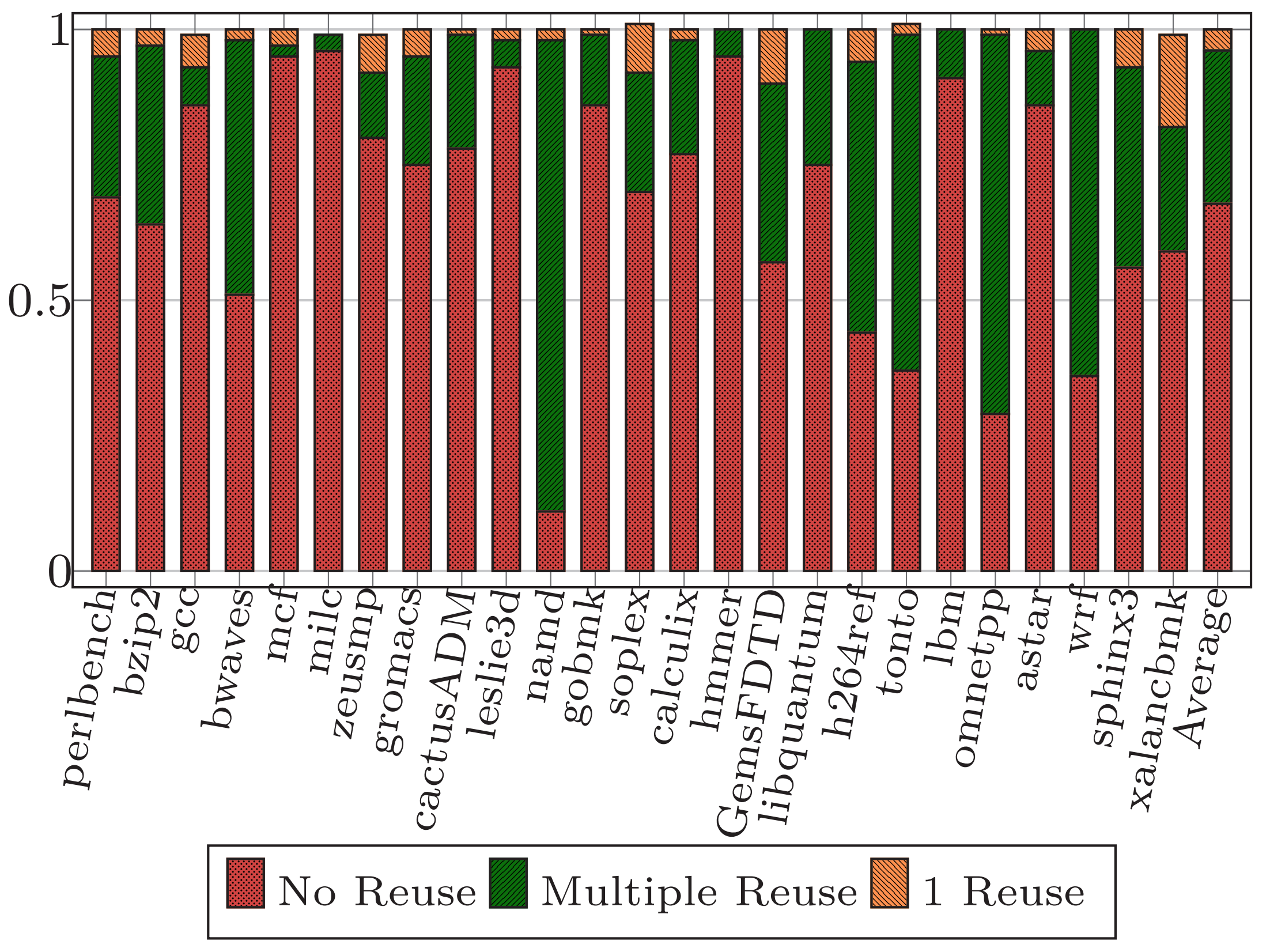
Contribution

In this work we deal with the slow and energy-hungry write operations problem of Spin-Transfer Torque RAM (STT-RAM) by leveraging the reuse locality property of stream references to the Last-Level Cache (LLC). We propose a new management policy for STT-RAM SLLCs that employs a *Reuse Detector* between private cache levels and the SLLC that avoids the insertion in the SLLC of blocks not exhibiting reuse, reducing the amount of writes to the SLLC and also the energy consumption.

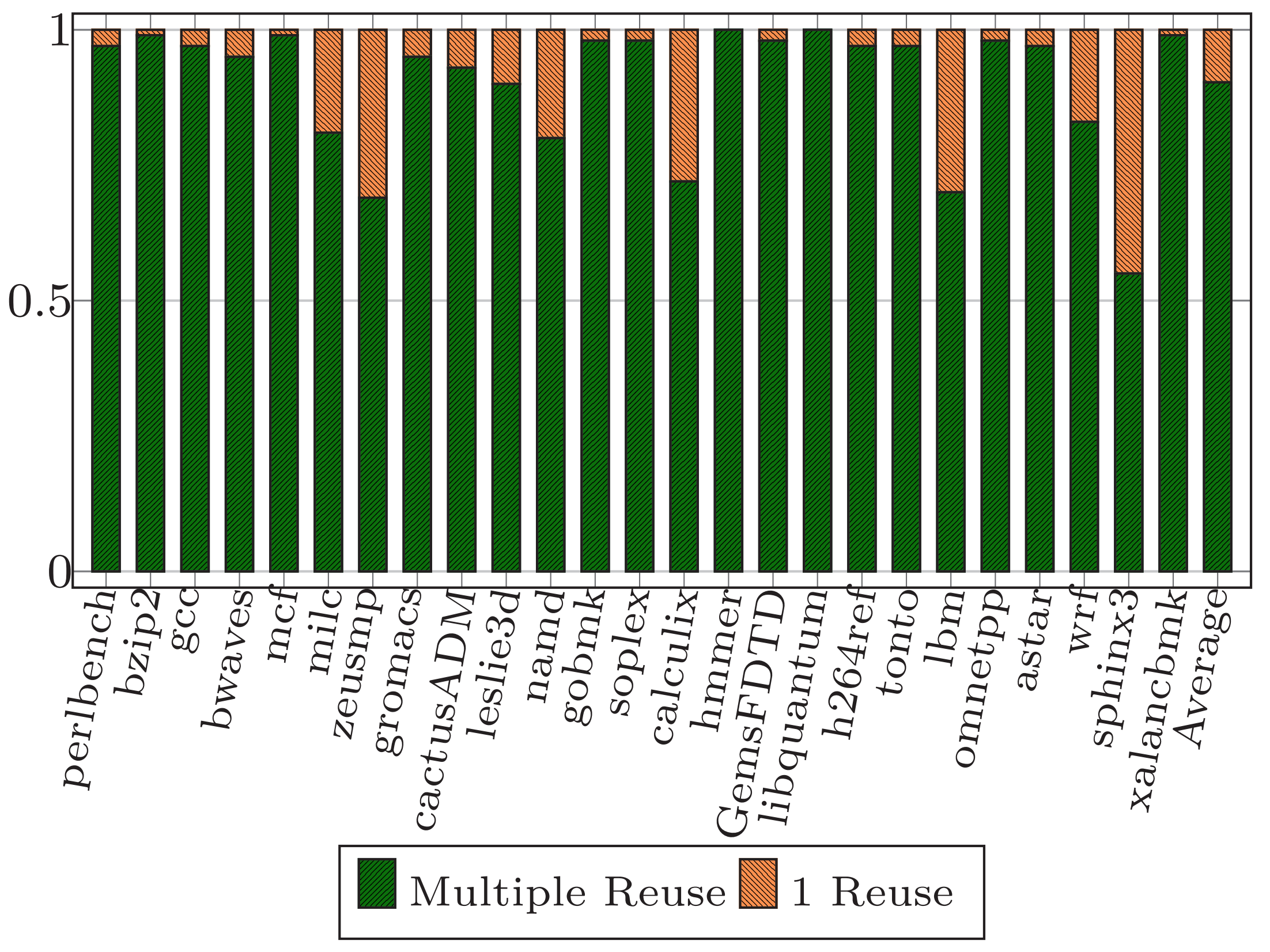
Our experimental results reveal that our scheme outperforms DASCA, the state-of-the-art STT-RAM SLLC management, reporting on average, energy reductions of 40% (quad-core) and 35% (eight-core) in the SLLC, an additional 6.5% (in both quad and eight-core) energy reduction in the main memory, and improving performance by 3% (quad-core) and 7% (eight-core) compared with a conventional STT-RAM SLLC replacement policy.

Motivation

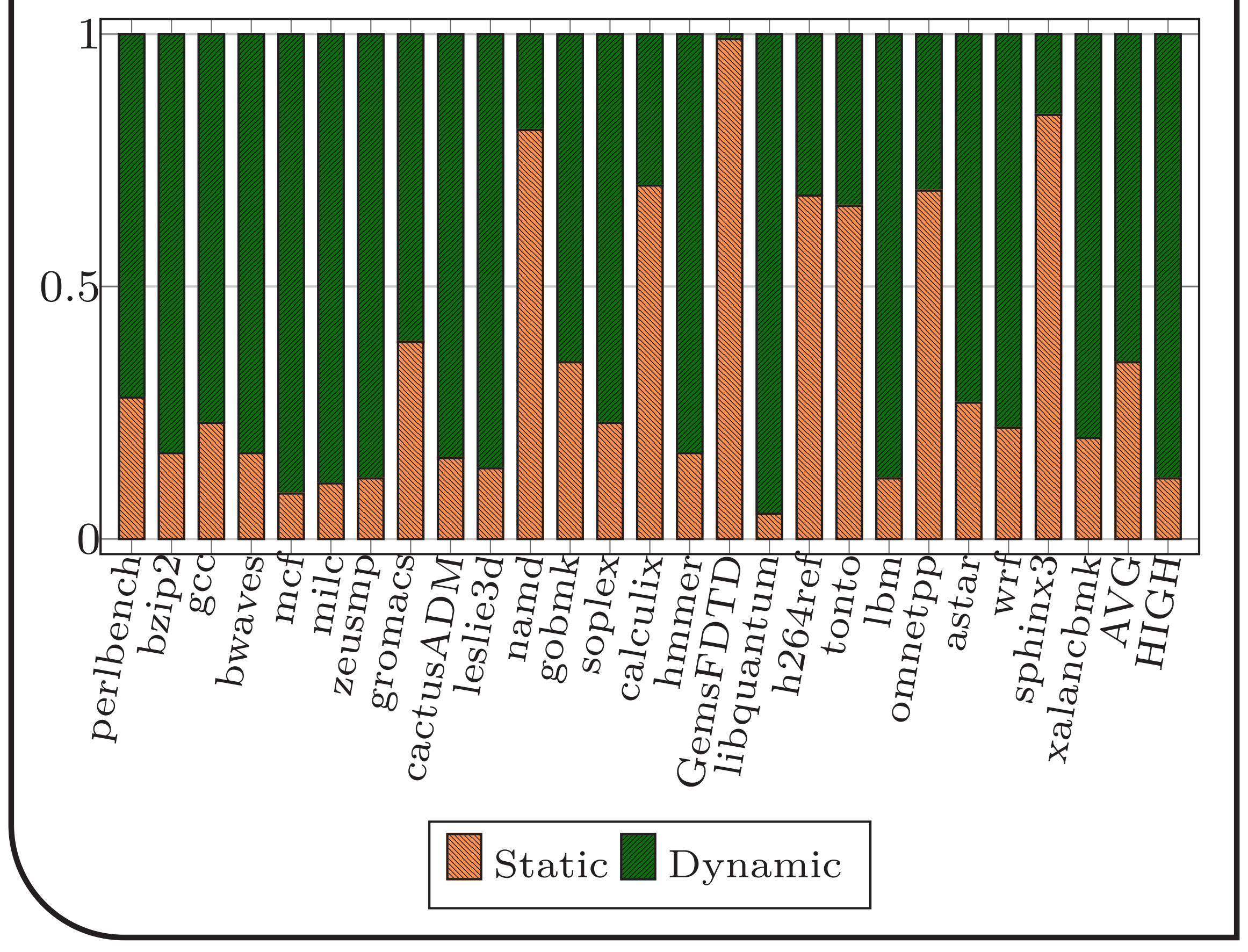
1) Breakdown of blocks replaced from the LLC: **most of blocks exhibit no reuse.**



2) Breakdown of blocks hits at the LLC: **most of hits correspond to blocks with multiple reuse.**

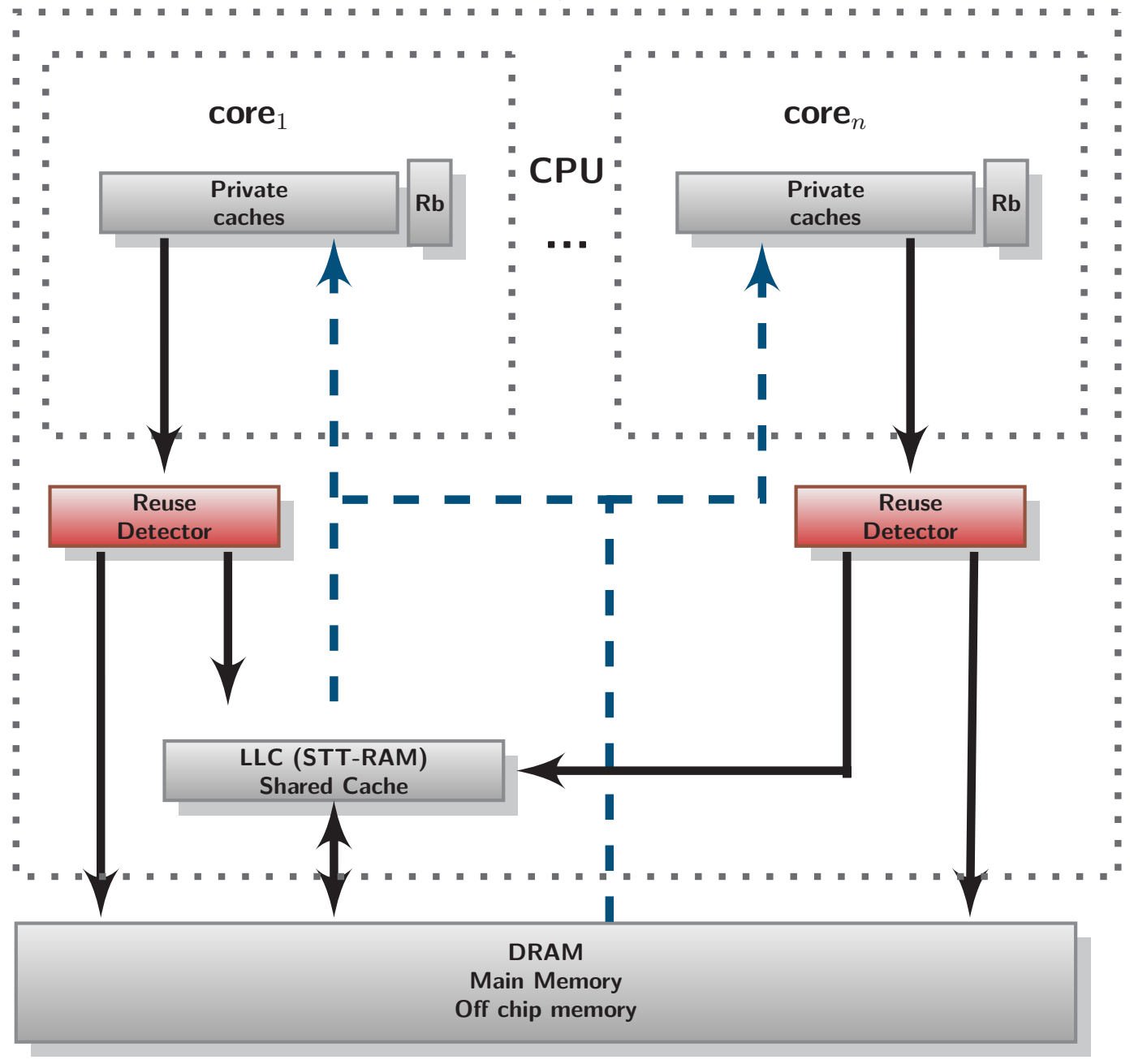


3) Breakdown of energy consumption in the LLC: **most energy consumption is dynamic.**



Proposal

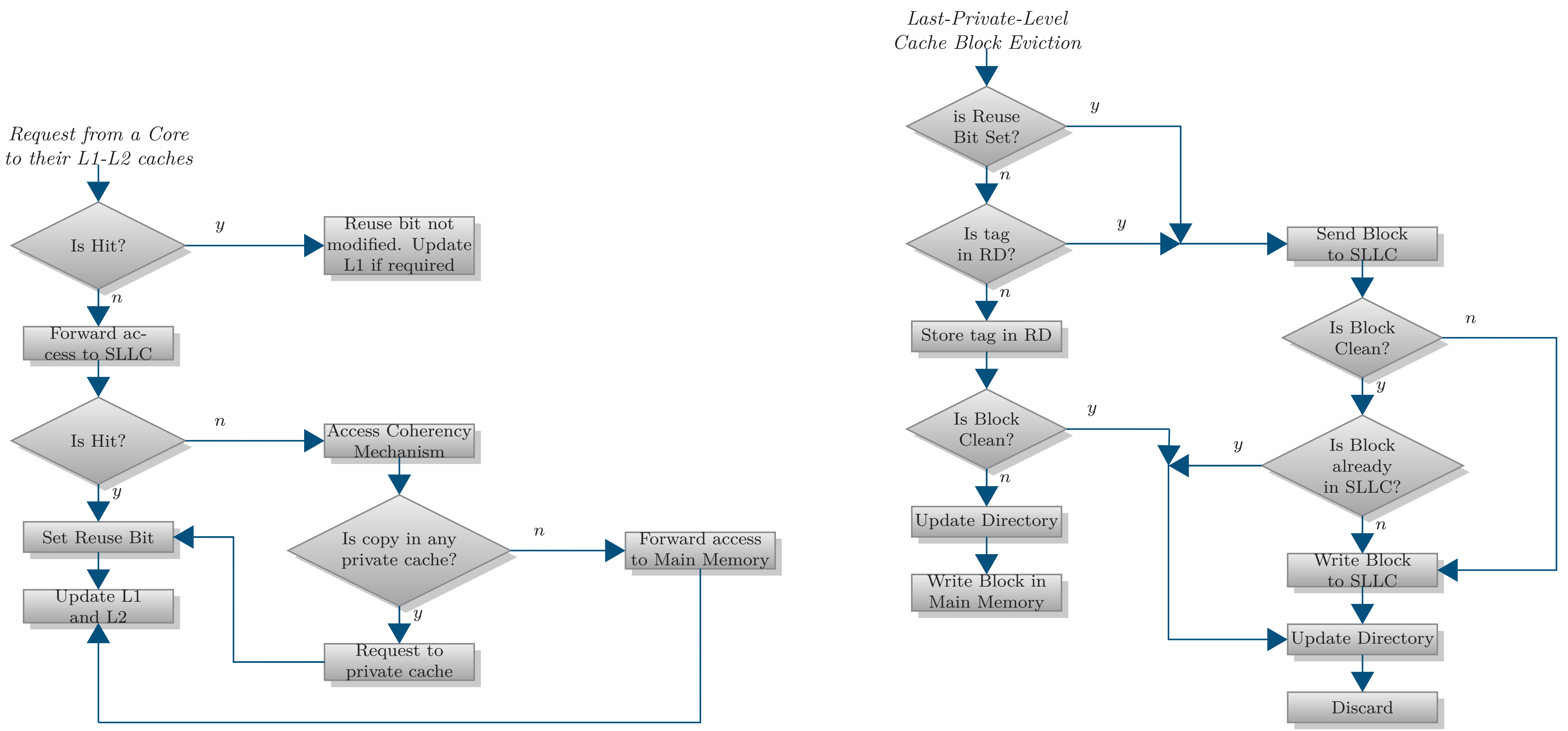
- **Main objective**: Tackle the high energy and high latency of STT-RAM writes by reducing the number of non-reused blocks allocated to the LLC.
- **State-of-the-art STT-RAM filter**: DASCA scheme [AHN14], in HPCA 2014. Our content selector is based on previous work [ALB13][DIA15].
- **Implementation**: Use a Reuse Detector (RD) as content selector for the LLC. The RD is a FIFO buffer that stores *supertags* (addresses of blocks evicted by private levels).



CompressedTag	S$_1$	S$_0$	RPL	V
Bits 13-4	Bit 3	Bit 2	Bit 1	Bit 0

The RD has 1024 sets and 16 ways and a sector size of 2 blocks. Each RD entry requires 14 bits (10 for the compressed tag, 2 for the block presence, 1 for the replacement policy and 1 validity bit)as Figure 6 illustrates. Given that the amount of entries in the RD is 8K, the total **extra storage required per core is 14 KB**, which represents a negligible 1.3% of an 1MB last level cache.

- **Operation**:

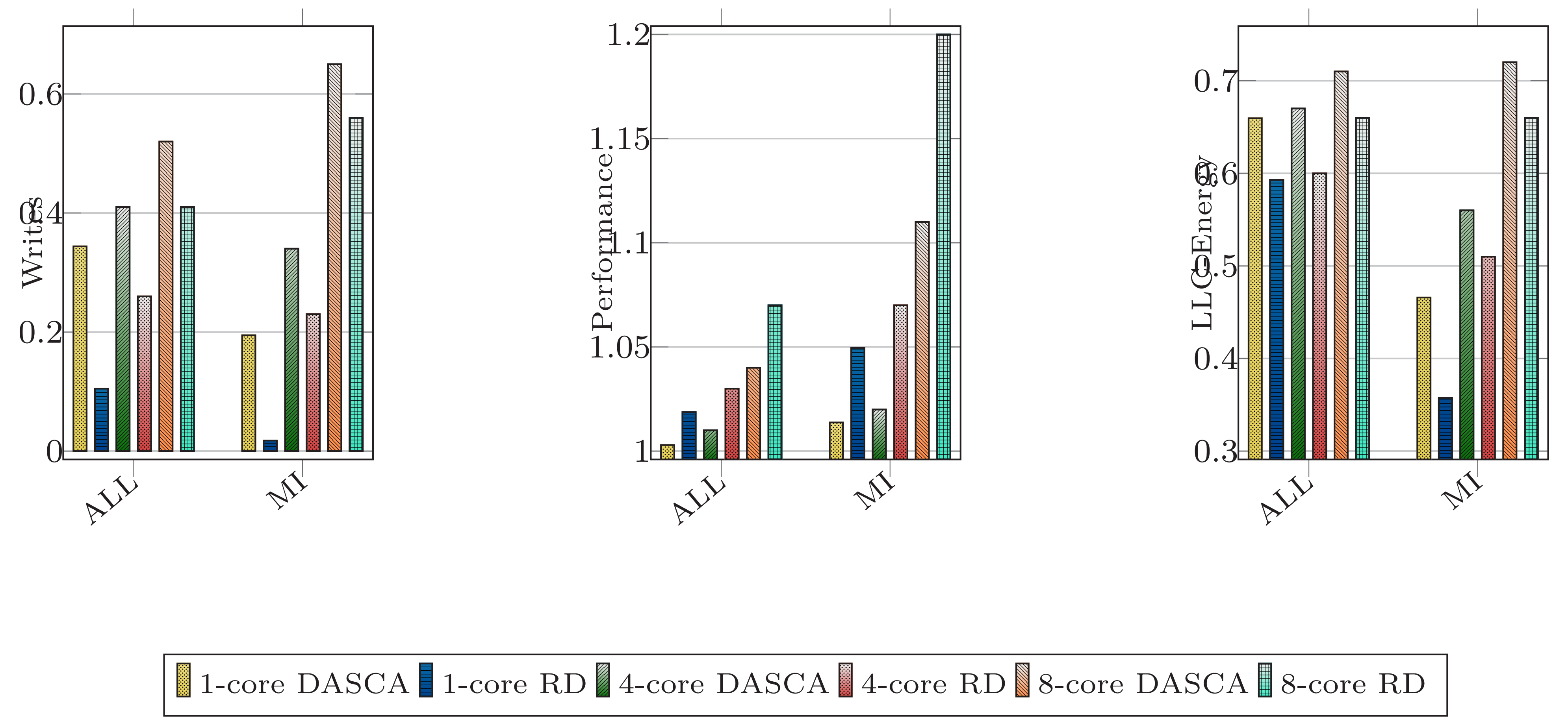

Request from a Core to their L1-L2 caches
Is Hit?
y
Reuse bit not modified. Update L1 if required
n
Forward access to SLLC
Is Hit?
n
Access Coherency Mechanism
y
Set Reuse Bit
Update L1 and L2
Is copy in any private cache?
n
Forward access to Main Memory
y
Request to private cache
Last-Private-Level Cache Block Eviction
is Reuse Bit Set?
y
n
Is tag in RD?
y
n
Store tag in RD
Is Block Clean?
y
n
Update Directory
Write Block in Main Memory
Send Block to SLLC
Is Block Clean?
n
y
Is Block already in SLLC?
y
n
Write Block to SLLC
Update Directory
Discard

Results normalized to the baseline

Simulator: gem5 **Benchmarks:** SPEC CPU2006, classified as All and memory-intensive (MI)

Private L1: 32KB, 8-ways, 64B line, 2 cycle-latency, SRAM; **Private L2:** 256KB, 16-ways, 64B line, 5 cycle-latency, SRAM

Shared LLC (L3): 1MB per core, 16-ways, 64B line, R/W latency: 6/17 cycles, STT-RAM



References

[AHN14] J. Ahn, S. Yoo and K. Choi DASCA: Dead write prediction assisted STT-RAM cache architecture. In *International Symposium on High Performance Architecture* (HPCA), pp. 25-36, 2014.

[ALB13] J.Albericio, P. Ibañez, V. Viñals and J.M. Llabería The Reuse Cache: Downsizing the shared last-level cache. In *International Symposium on Microarchitecture* (MICRO), pp. 310-321, 2013.

[DIA15] J. Díaz, T. Monreal, V. Viñals, P. Ibañez, and J.M. Llabería Selección de contenidos basada en reuso para caches compartidas en exclusión. In *Jornadas de Paralelismo* (JP), pp. 433-442, 2015.