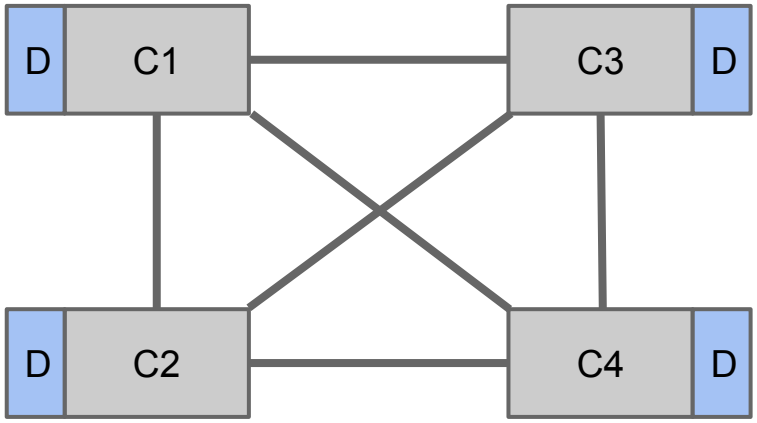
Increasing the Effectiveness of Directory Caches by Deactivating Coherence for Private Memory Blocks

Blas Cuesta, Alberto Ros, María E. Gómez,
Antonio Robles, and José Duato

Universitat Politècnica de València

Presented by: Anuj Kalia and Conglong Li
Oct 10, 2014

Directory protocol for cache coherence



A logically distributed directory (D) contains coherence information for cached blocks.

Problem: Directories should be Small

The directory *could* store the state for all blocks in the system.

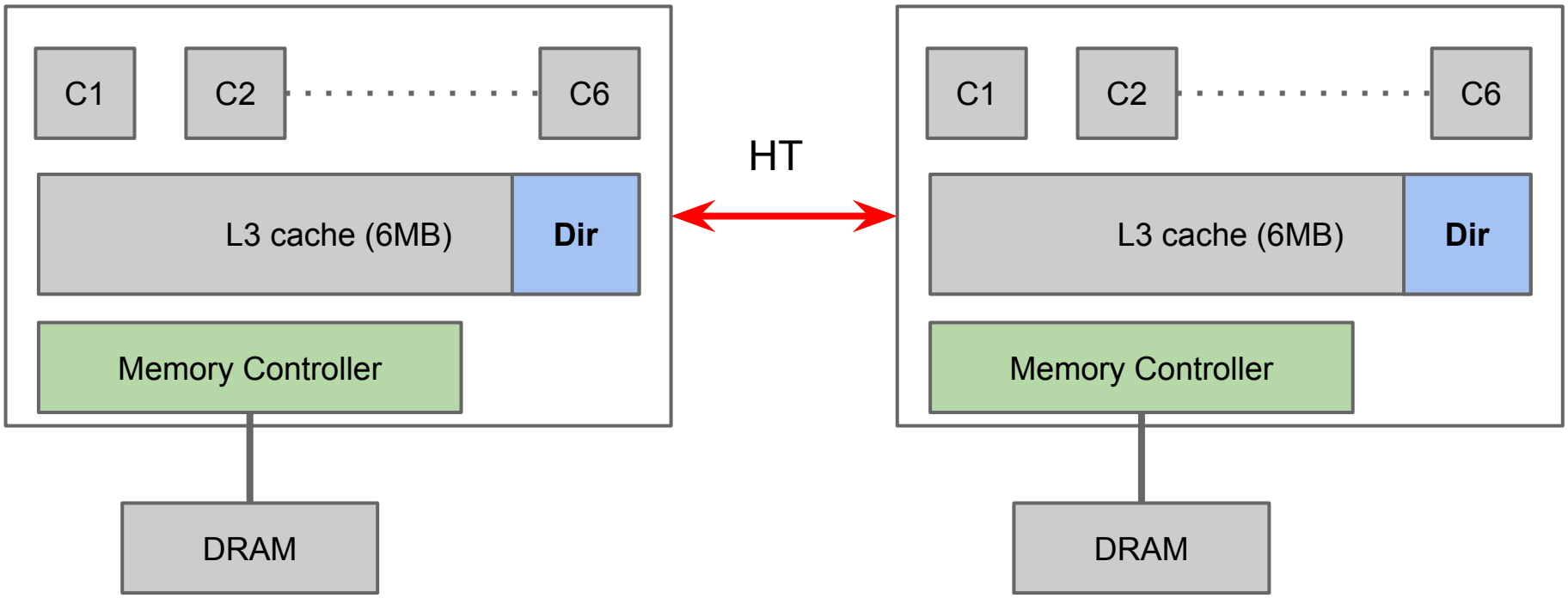
- But then, the directory needs to be in DRAM.

Solution: Use a “directory cache”.

- Track only cached blocks in the system.
- An eviction from the directory cache leads to eviction from all local caches.

Directory caches can have high miss rates: up to 70%.

Directories in AMD Magny-Cours



Conditions for a directory partition to track a block:

1. It is the home node for that cache block
2. The block needs to be cached somewhere in the system

Problem

Is it possible to make better use of directory cache capacity?

Observation: Many cache blocks are private

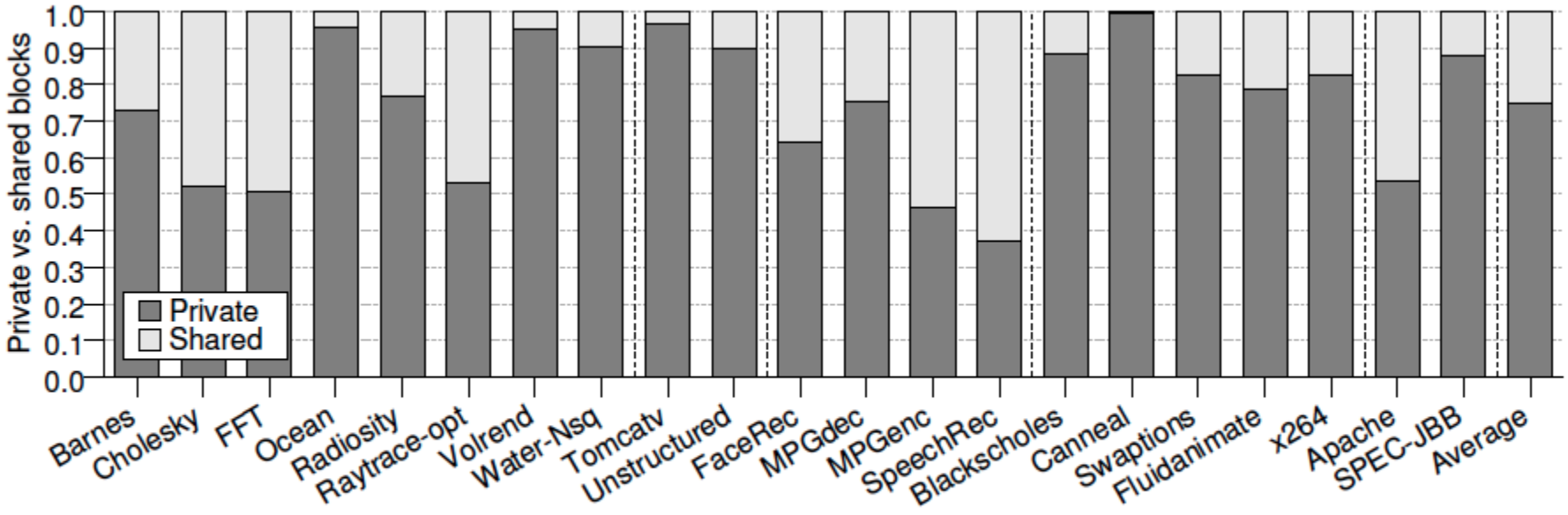


Figure 1: Fraction of private versus shared blocks.

Solution: De-activate coherence for private blocks

Need mechanisms to:

- Deactivate coherence on private blocks
- Detect when private blocks become shared
- Recover coherence on shared blocks

Exploit existing hardware (TLB) and software (page fault) support.

As a result

- Less tracking (57%) and evictions (70%) in directory cache
- Less misses (35%) in processor caches
- Better performance (15%) and less energy (40%)

De-activating coherence for private pages

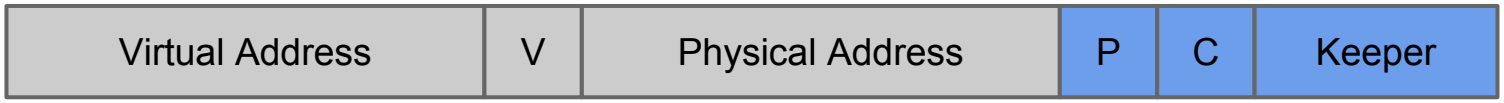
+ All memory operations have to go through the TLB.
+ The TLB contains some reserved bits.

V: the TLB entry is valid
P: the page is private
L: the page is locked (discussed later)

If a memory access uses a TLB entry marked “P”, a non-coherent request to the memory controller is issued.

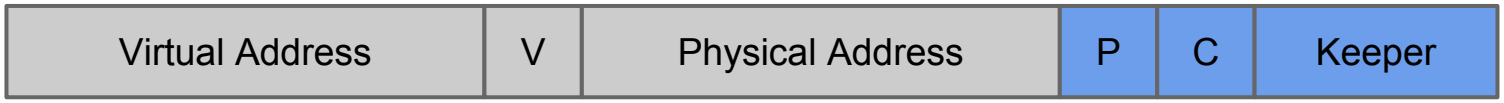
Detecting sharing of pages

+ All possible sharers of a page also share a page table entry.
+ Augment the page table entry with additional fields



V: the page-table entry is valid
P: this page is private
C: this page is cached by a processor
Keeper: a bitmask representing the processor that has this page

Detecting sharing of pages



When a Page Table Entry is created, set P = 1, C = 0.

When it is first accessed from processor #N, set C = 1, Keeper = N.

When it is accessed again from processor #M, compare Keeper to M.

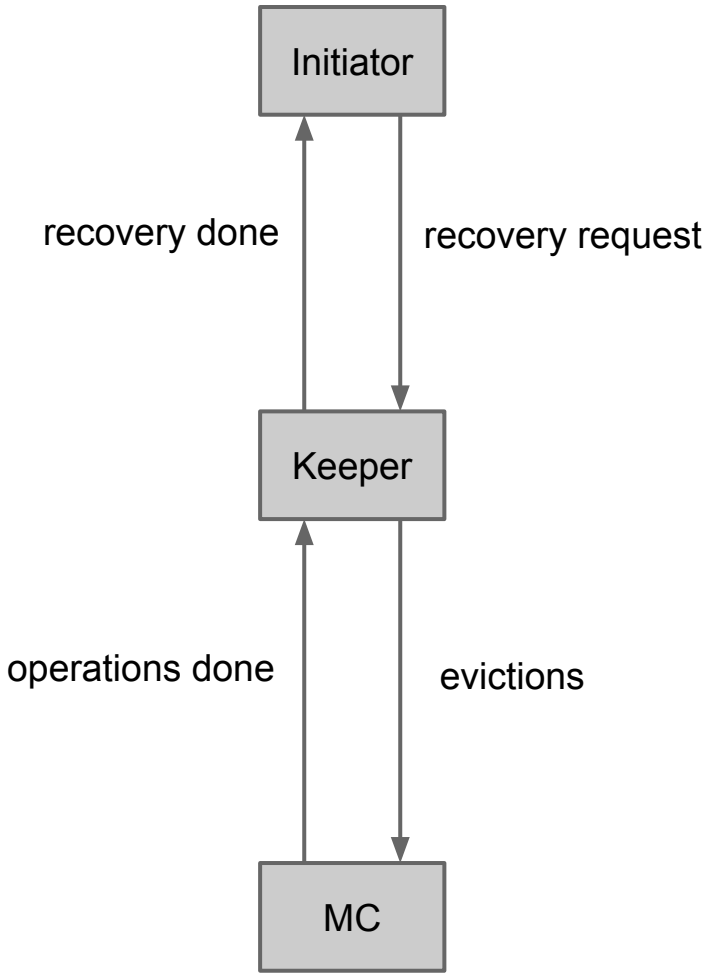
- If Keeper = M, do nothing
- If Keeper != M, trigger a coherence recovery mechanism

Coherence Recovery Mechanism (Flushing-Based)

- Initiator
- Recovery request

- Keeper
- Locking
- Flushing & writebacks
- Set to shared & unlock

- Updating-Based
- Bit-vector
- Avoid flushing



Evaluation: Methodology

- Simulation
  - Full-system: Virtutech Simics
  - Processor: GEMS toolset (8 dies x 2 cores)
  - Network: GARNET
  - Energy: McPAT

- Benchmarks
  - Parallel (SPLASH-2, ALPBenchs, PARSEC)
  - Scientific
  - Commercial

Evaluation: Private Blocks

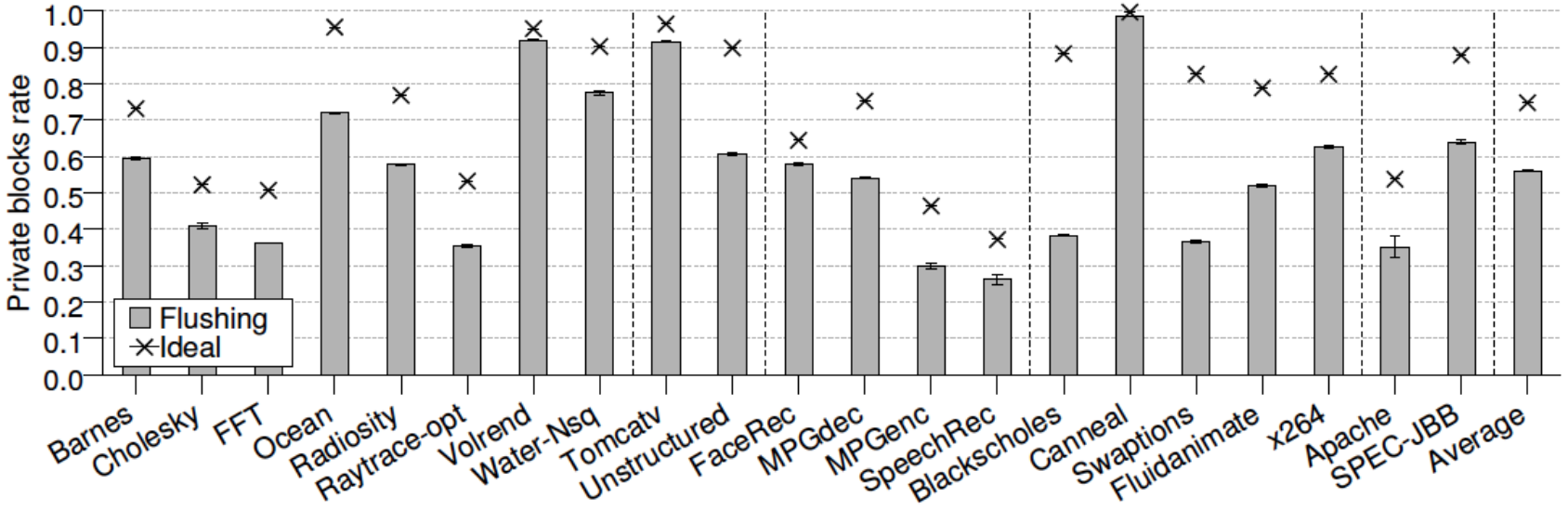


Figure 7: Fraction of actual private blocks versus detected private blocks.

- Actual 75%, detected 57%
- Granularity

Evaluation: Processor Cache Misses

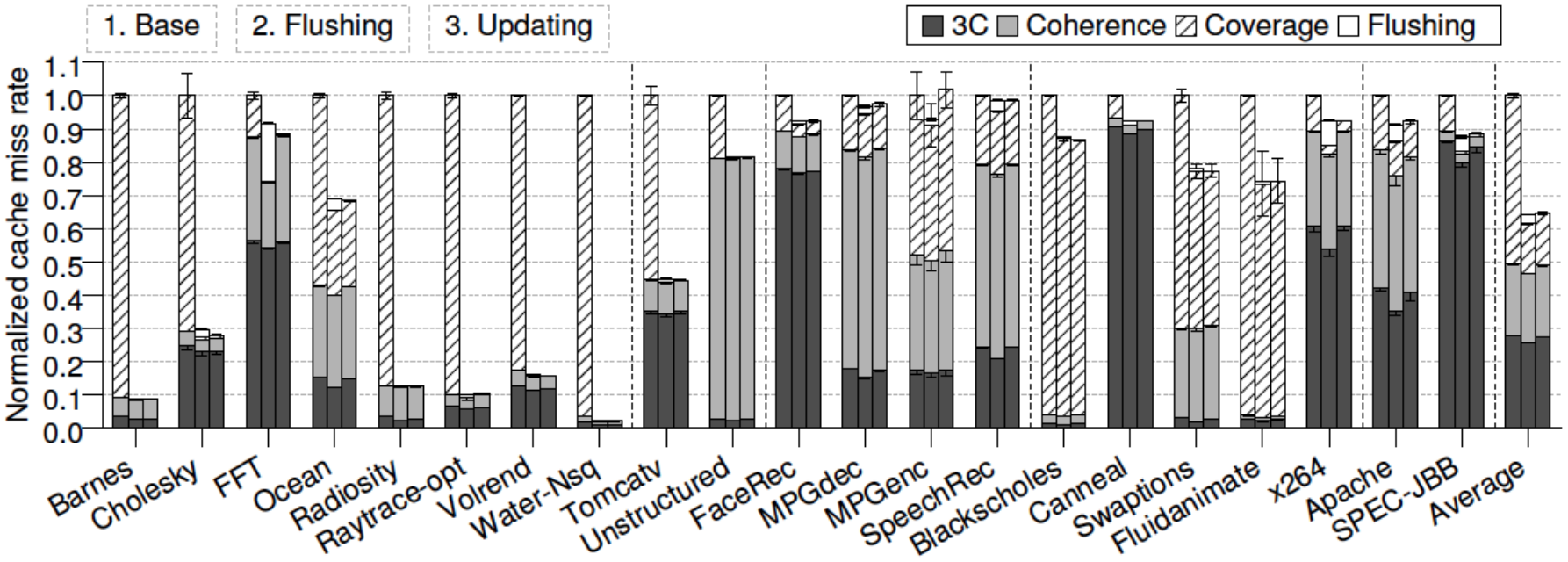


Figure 8: Normalized cache miss rate classification.

- Cache miss rate reduced by 35%
- Also reduce the network traffic and cache miss latency

Evaluation: Coherence Recovery Mechanism

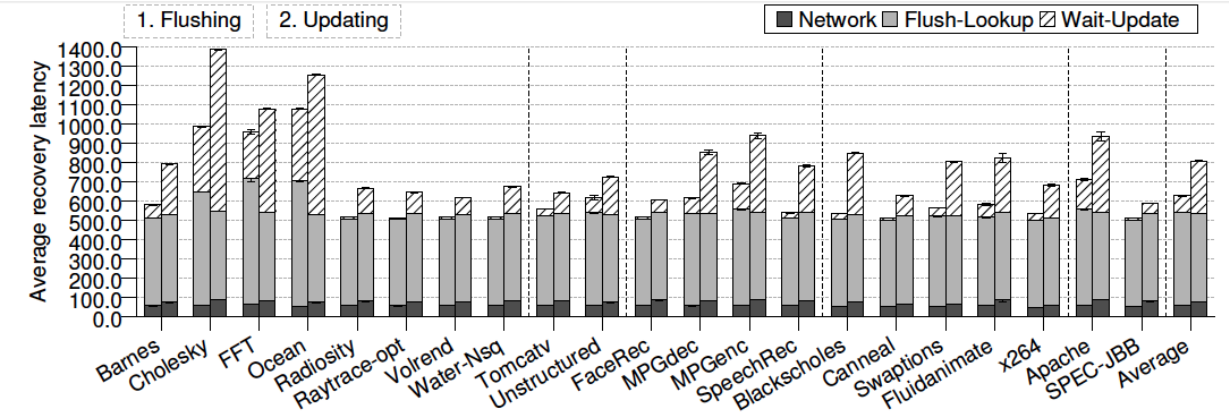


Figure 11: Average latency of the coherence recovery mechanisms (in clock cycles).

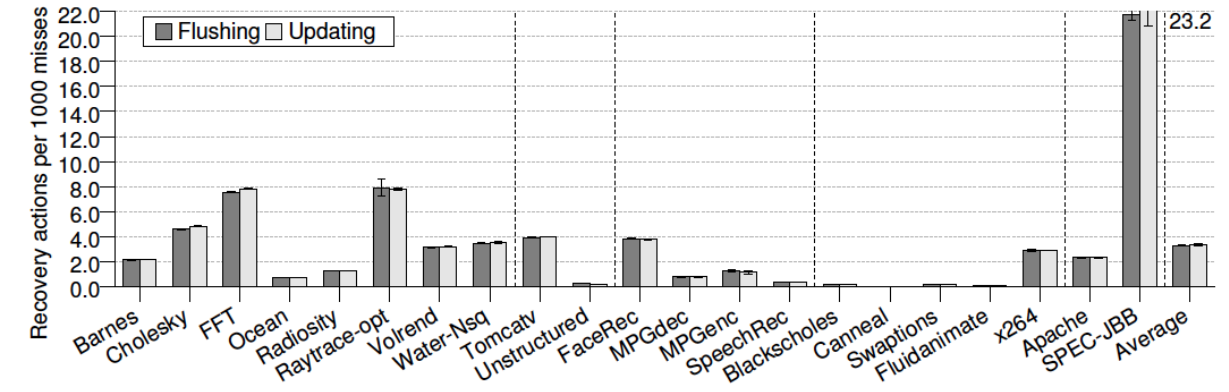


Figure 12: Times that the coherence recovery mechanism is triggered per 1000 cache misses.

Evaluation: Execution Time

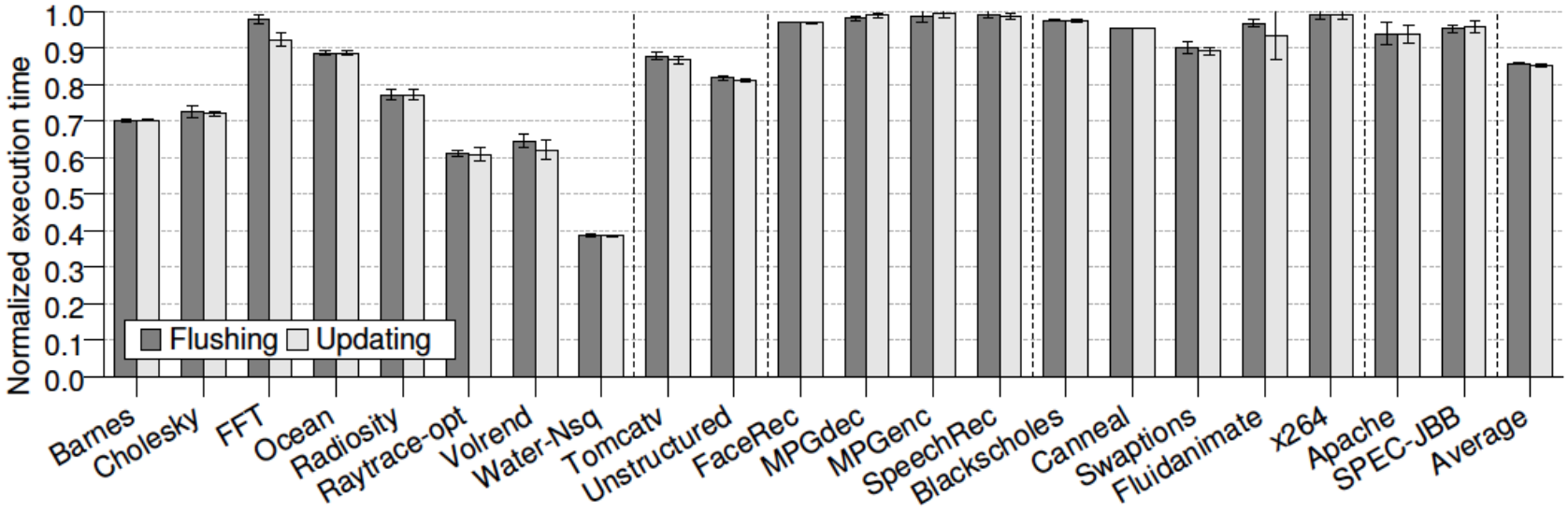
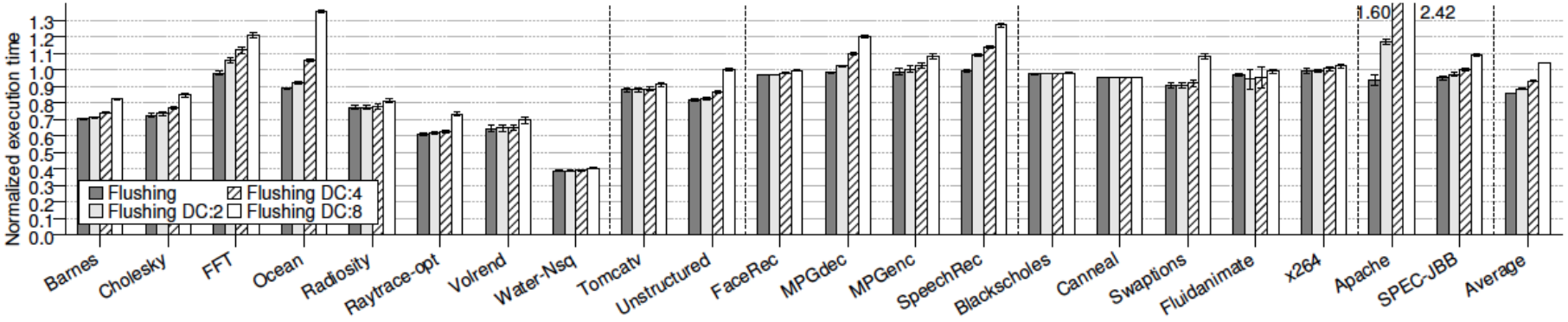


Figure 13: Normalized runtime maintaining the directory cache size.

- Runtime reduced by 15%

Evaluation: Execution Time



: Execution time normalized to the base system. *DC:2*, *DC:4*, and *DC:8* stand for directory caches ed by 2, 4, and 8, respectively.

- Smaller directory cache can do the job

Conclusion

- Avoid tracking of private blocks
  - OS-based detection

- Result
  - 57% out of 75% private blocks detected
  - 15% performance improvement
  - Similar performance with smaller directory
  - Coherence recovery overhead is small

- Discussion
  - Use software support (partition, etc) to avoid hotspotting

Other sources

A Primer on Memory Consistency and Cache Coherence - by David A. Wood