Locality-Aware Data Placement in DRAM-PCM Hybrid Memories

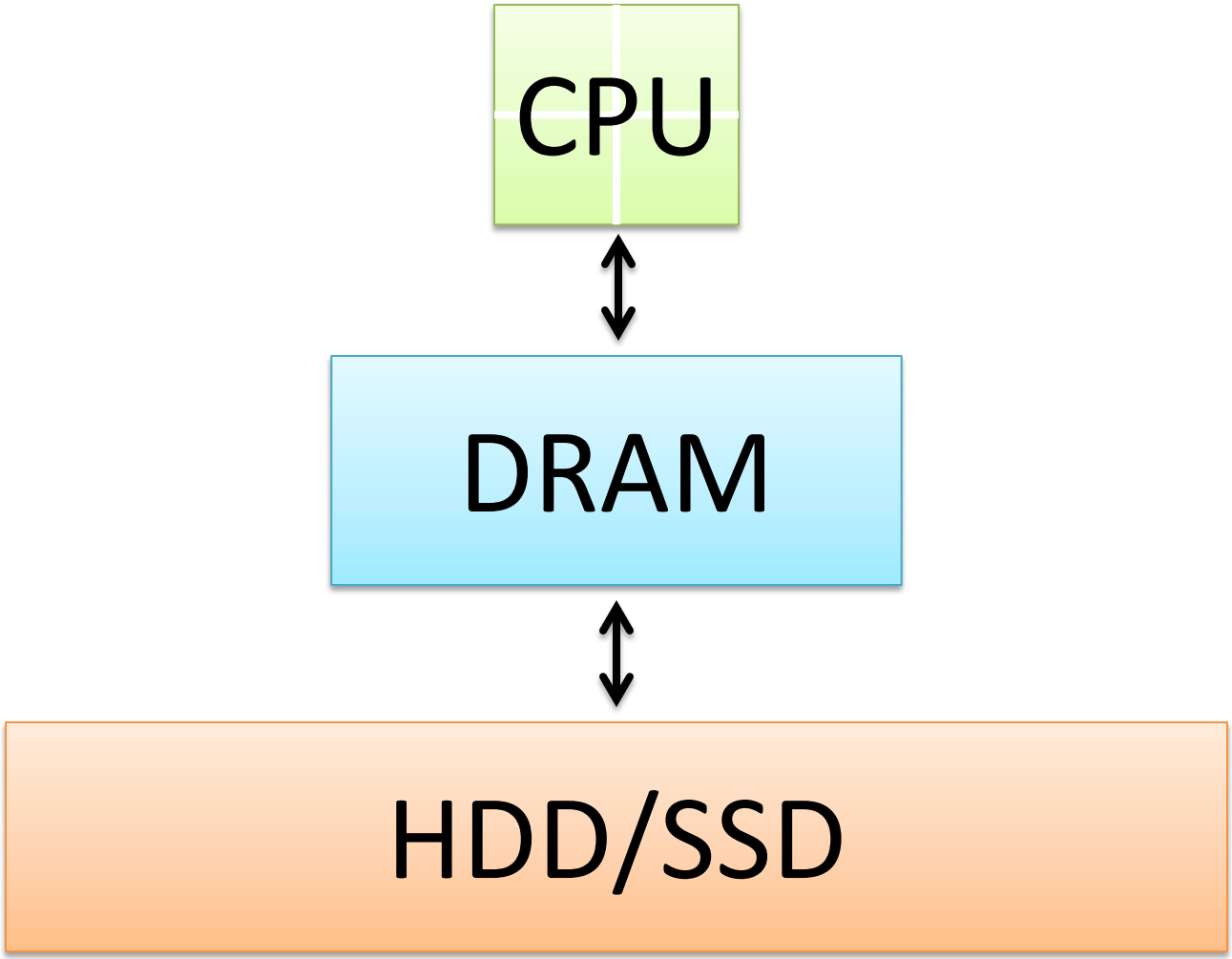
Justin Meza
HanBin Yoon
Rachata Ausavarungnirun
Rachael Harding
Onur Mutlu

Carnegie Mellon

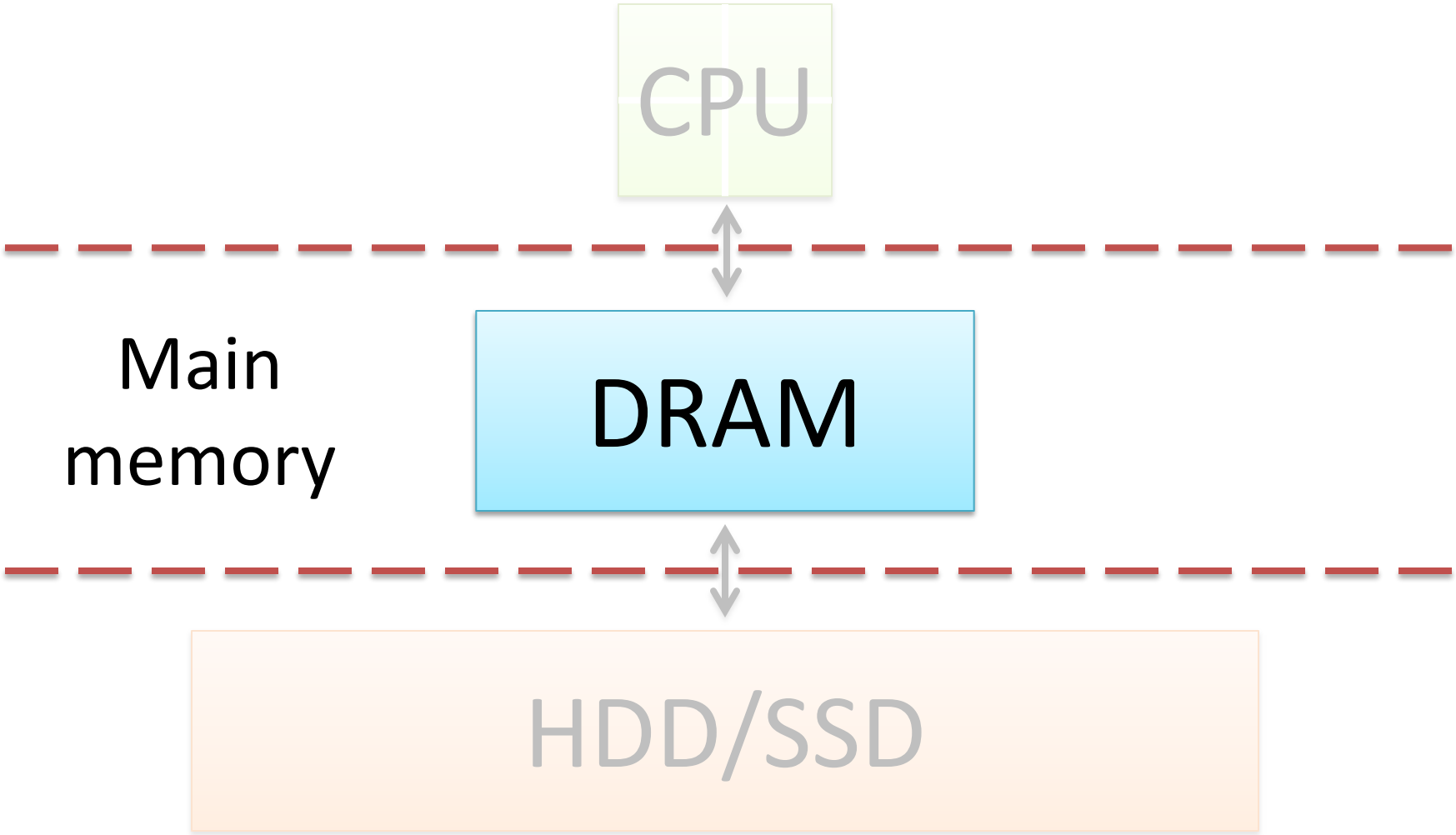
Overview

- New, data-intensive applications have spurred a demand for huge main memory capacity
  - New memories like PCM provide more density than DRAM, but have drawbacks of their own
  - Hybrid memories can achieve the best of both
- We identify row buffer locality (RBL) as a key metric for caching and design an adaptive policy that caches rows with low RBL and high reuse in DRAM
- 17% perf. improvement over all-PCM memory
- Within 21% performance of all-DRAM memory

Modern Systems



Modern Systems



DRAM

+ Low latency
+ Low cost
– Limited density

- Some new and important applications benefit from HUGE memory capacity

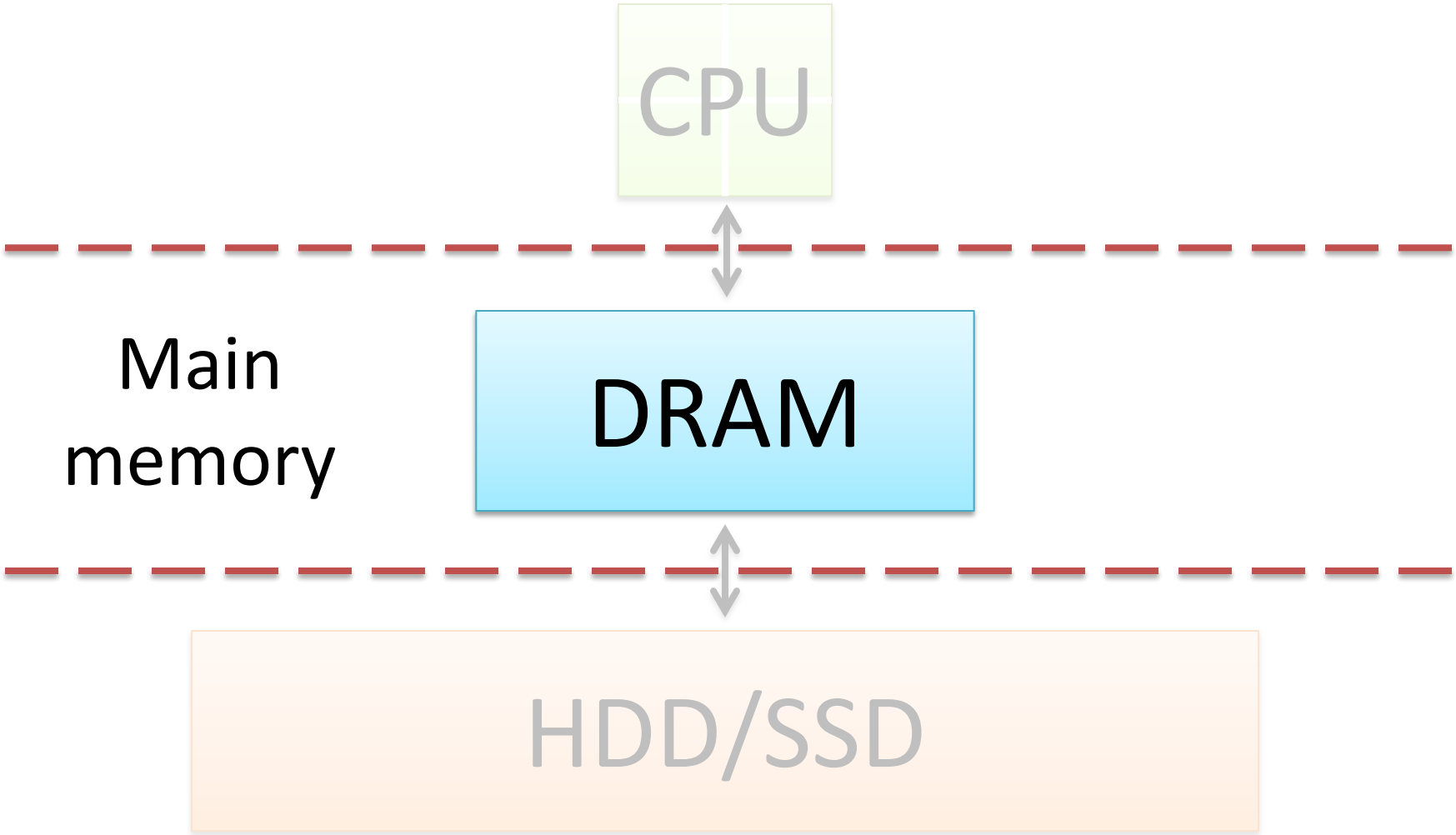
New, Higher Density Memories

- Phase change memory (PCM)
  - + Projected 3–12 × denser than DRAM[1]
- However, cannot simply replace DRAM
  - – Longer access latency (4–12 × DRAM[2])
  - – Higher access energy (2–40 × DRAM[2])
  - – Limited write endurance (~10^8 writes)

→Use DRAM as a cache to PCM memory

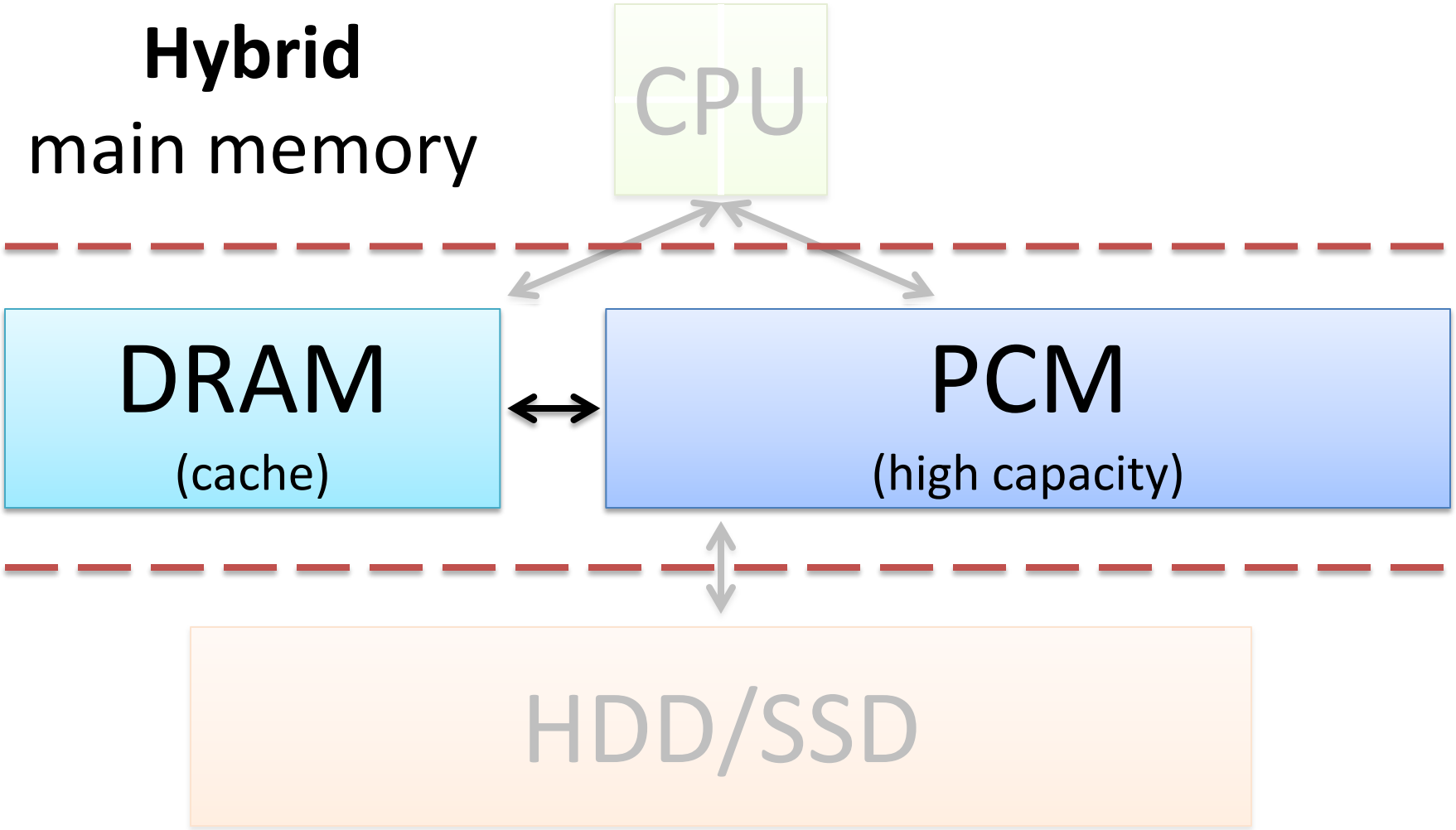
[1Mohan HPTS'09; 2Lee+ ISCA'09]

Modern Systems



Future Systems

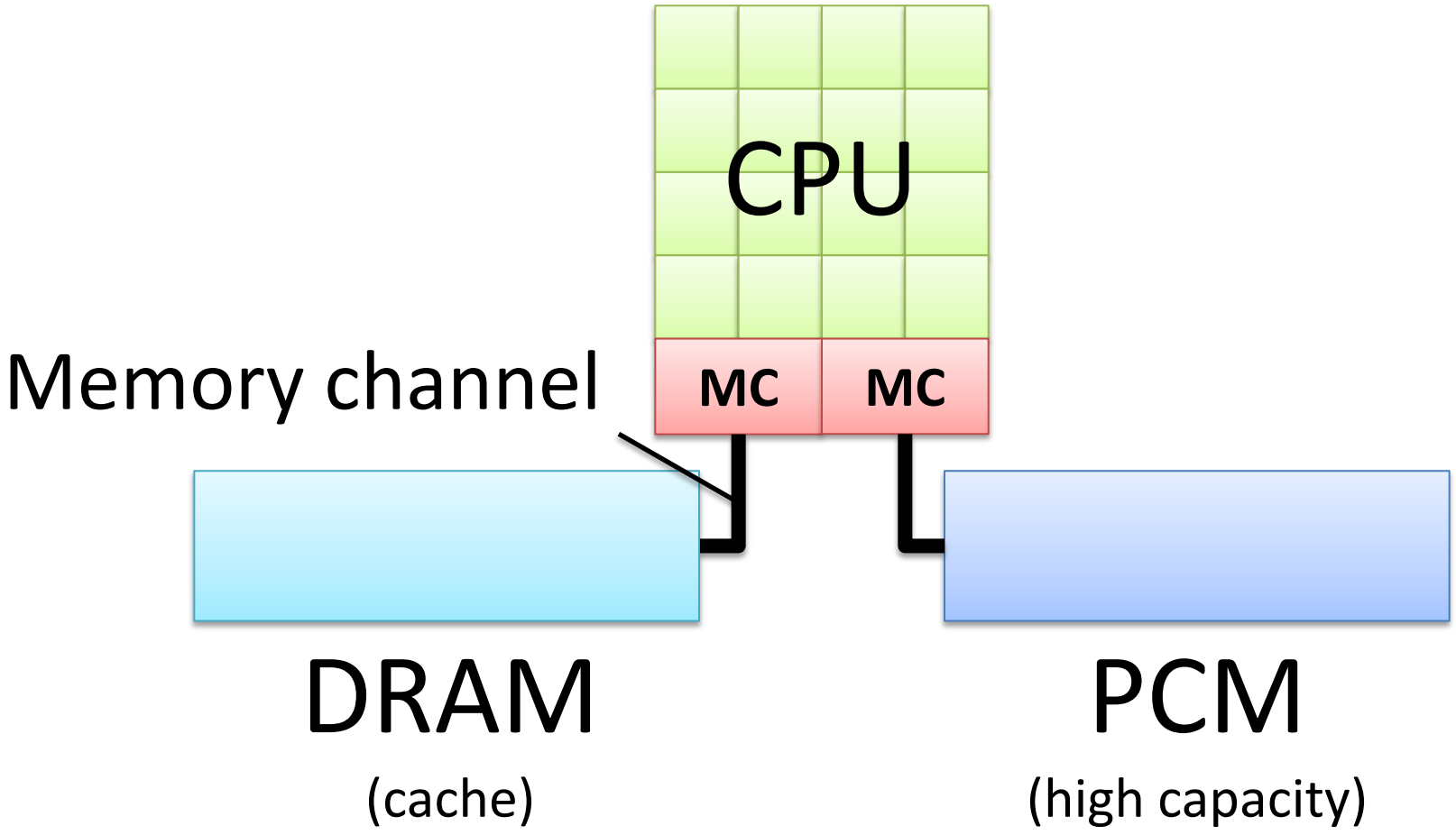
Hybrid
main memory



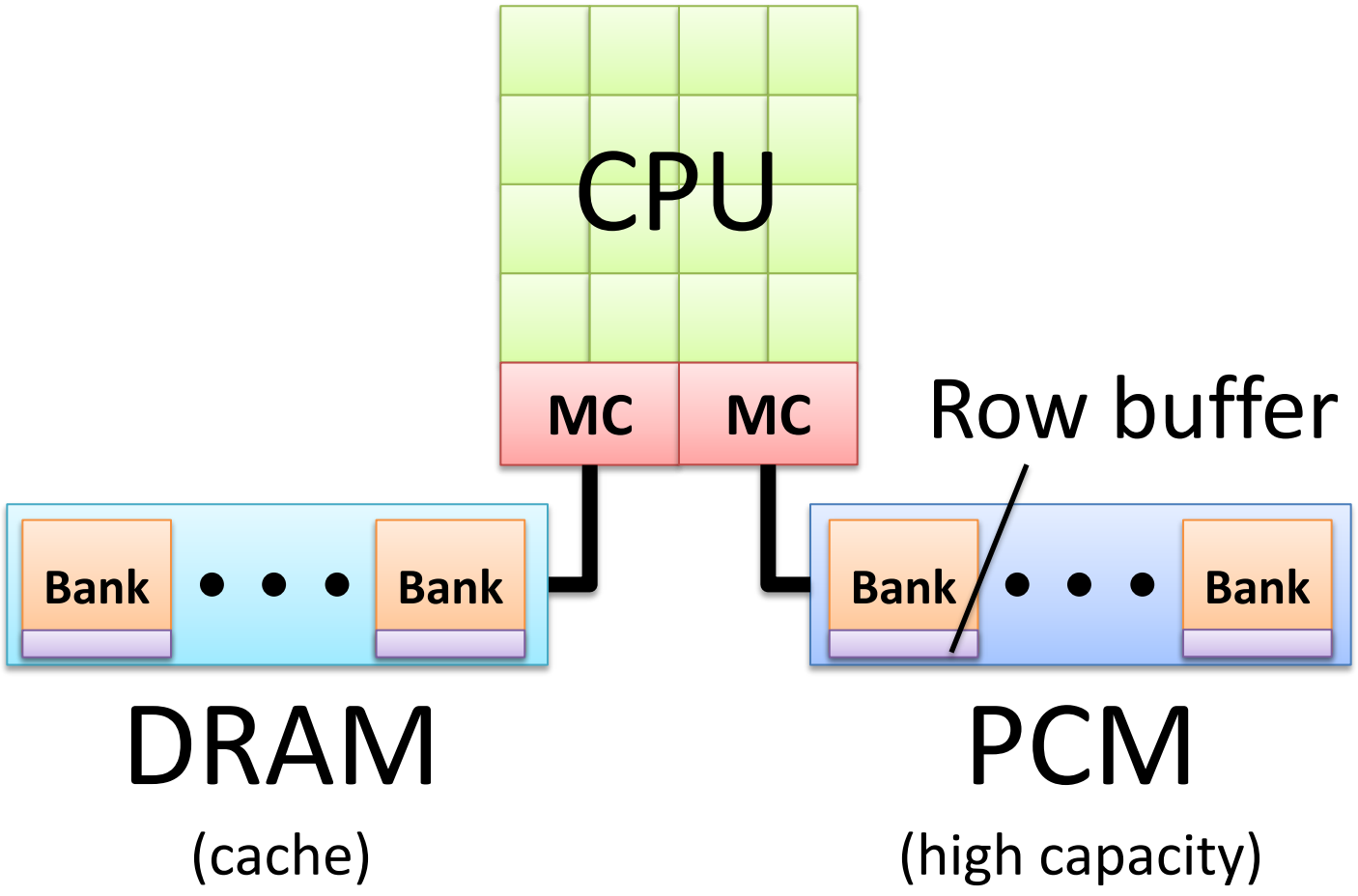
Hybrid Memory

- Benefits from both DRAM and PCM
  - DRAM: Low latencies, high endurance
  - PCM: High capacity
- Key question: Where to place data between these heterogeneous devices?
- To help answer this question, let's take a closer look at these technologies

Hybrid Memory: A Closer Look



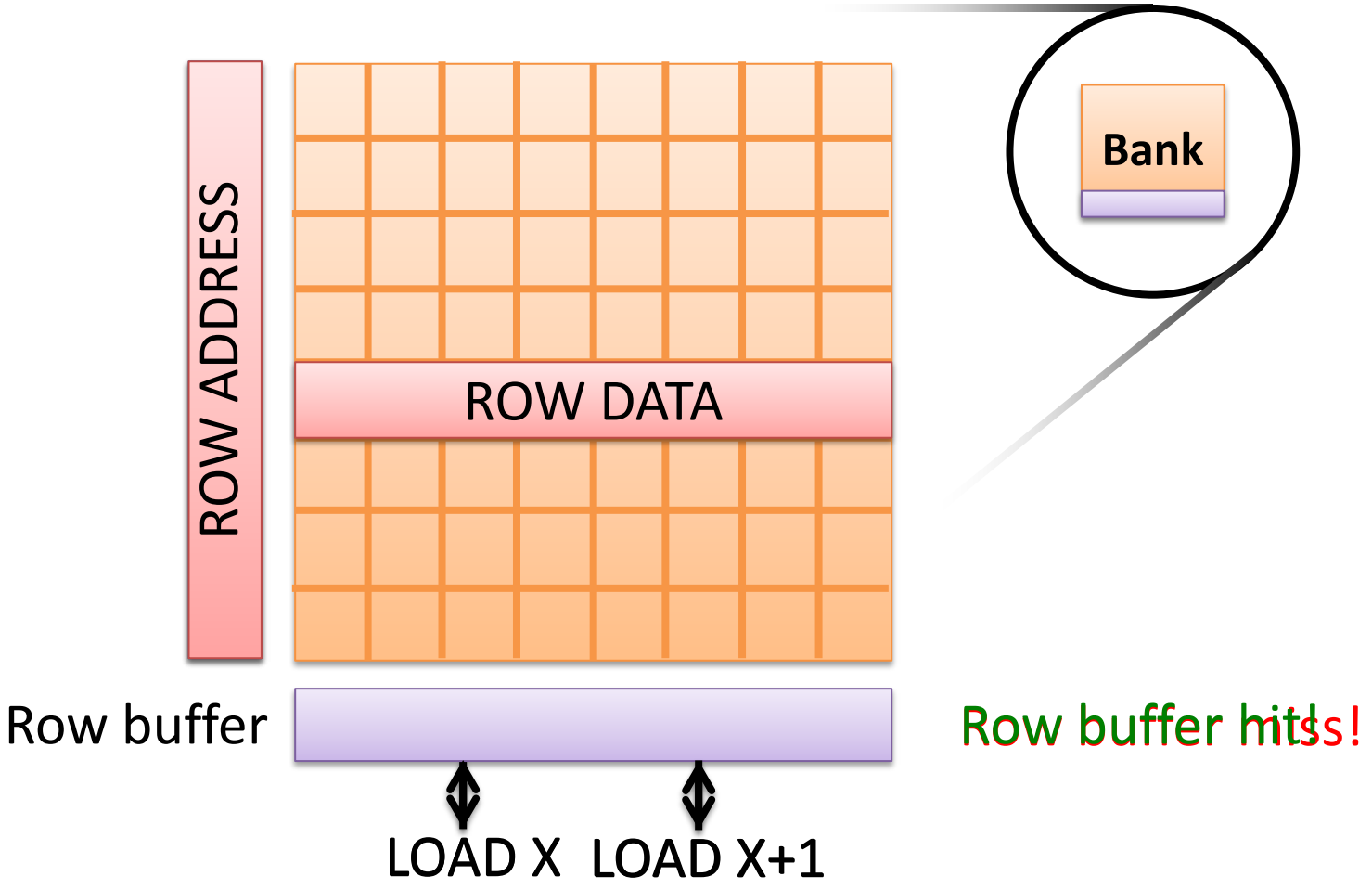
Hybrid Memory: A Closer Look



Row Buffers and Locality

- Memory organized in columns and rows
- Row buffers store last accessed row
- Accessing data from row buffer → fast
- Accessing data from device array → slow

Row Buffers and Locality

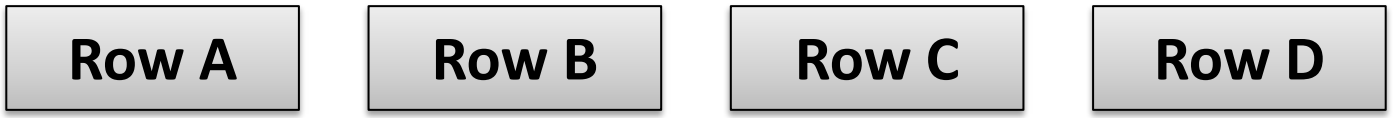


Key Observation

- DRAM and PCM both use row buffers
  - Row buffer hit latency **same** in both
  - Row buffer miss latency **small** in DRAM
  - Row buffer miss latency **large** in PCM
- Place data in DRAM which
  - Frequently miss in row buffer → miss penalty is smaller in DRAM
  - Are reused many times → caching data occupies channels and causes contention

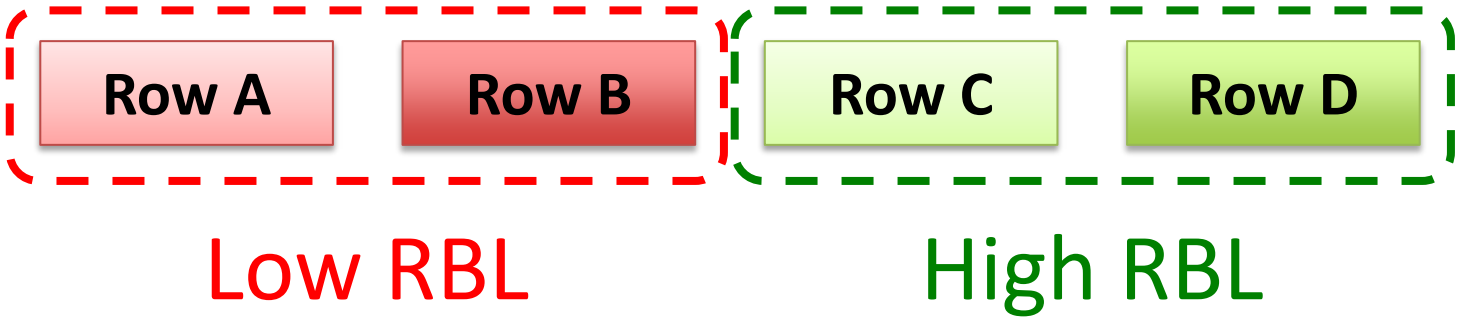
Data Placement Implications

Let’s say a processor accesses four rows



Data Placement Implications

Let's say a processor accesses four rows with different row buffer localities (RBL)



RBL-Unaware Policy

A **row buffer locality-unaware** policy could place these rows in the following manner

DRAM (High RBL)

PCM (Low RBL)

RBL-Unaware Policy

Accesses pattern to main memory:

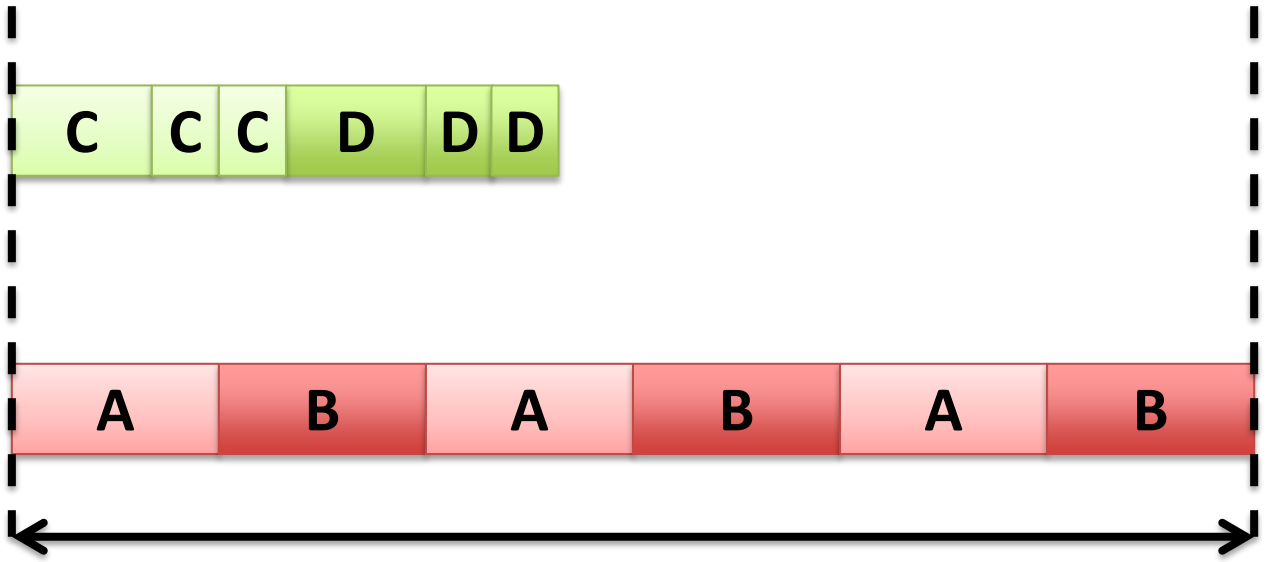
A (oldest), B, C, C, C, A, B, D, D, D, A, B (youngest)

DRAM

(High RBL)

PCM

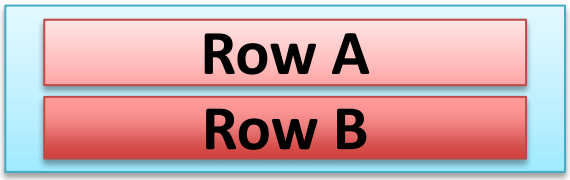
(Low RBL)



Stall time: 6 PCM device accesses

RBL-Aware Policy

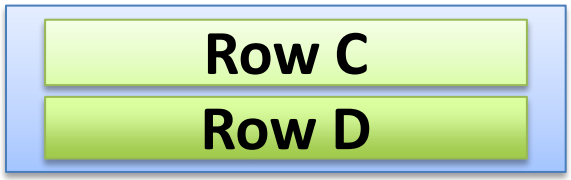
A **row buffer locality-aware** policy might place these rows in the following manner



DRAM

(Low RBL)

→ Benefit from reduced row buffer miss latency



PCM

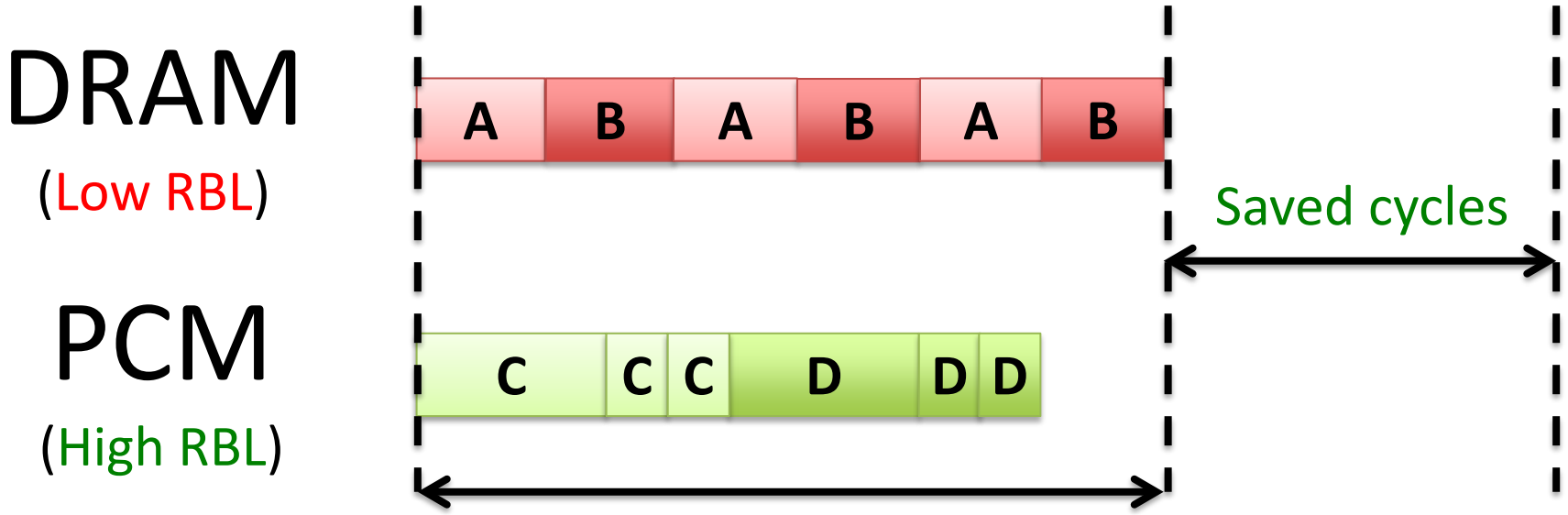
(High RBL)

→ Can access data from row buffer at same latency

RBL-Aware Policy

Accesses pattern to main memory:

A (oldest), B, C, C, C, A, B, D, D, D, A, B (youngest)



Stall time: 6 **DRAM** device accesses

Our Mechanism: DynRBLA

- For a subset of recently used rows in PCM:
  - Track **misses** as indicator of locality
  - Track **accesses** as indicator of reuse
- Cache rows with misses and accesses greater than a certain threshold
- Dynamically tune threshold to adjust to workload/system characteristics
  - Interval-based cost/benefit analysis

Related Work

- Cache rows based on frequency of access
- Similar to CHOP [Jiang+ HPCA'10]
  - + Does reduce bandwidth over caching on first access (conventional caching)
  - – But, also caches data which hit in row buffer → could have been serviced at same latency

Evaluation Methodology

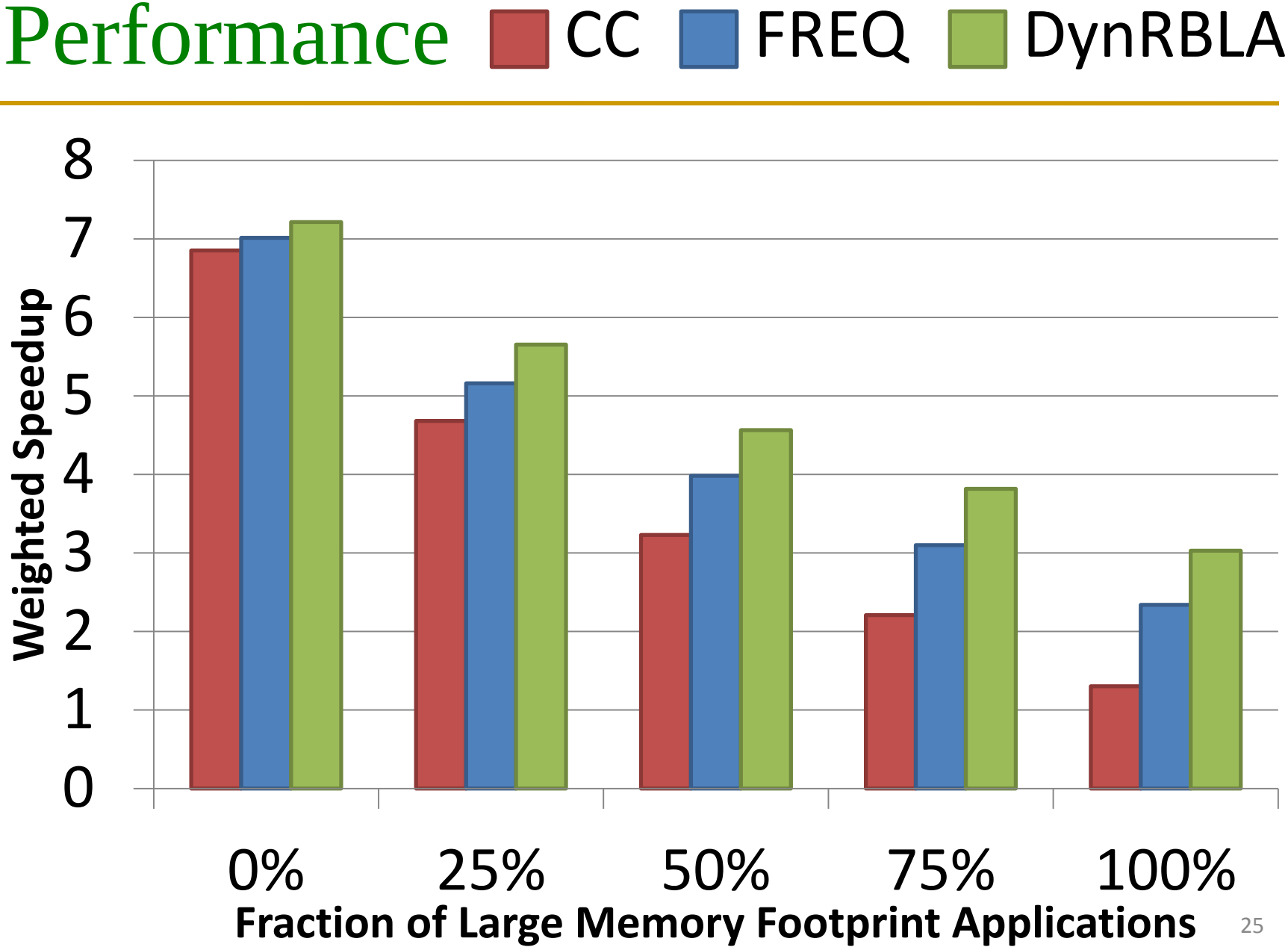
- Cycle-level x86 CPU/memory simulator
  - **CPU**: 16 out-of-order cores, 32KB private L1, 512KB shared L2
  - **Memory**: DDR3 1066 MT/s, 256MB DRAM, 8GB PCM, 2 KB row size
- SPEC CPU2006 benchmark suite
  - Categorized apps based on working set fitting in DRAM cache or not
  - 100 workload mixes per category

Policy Comparisons and Metrics

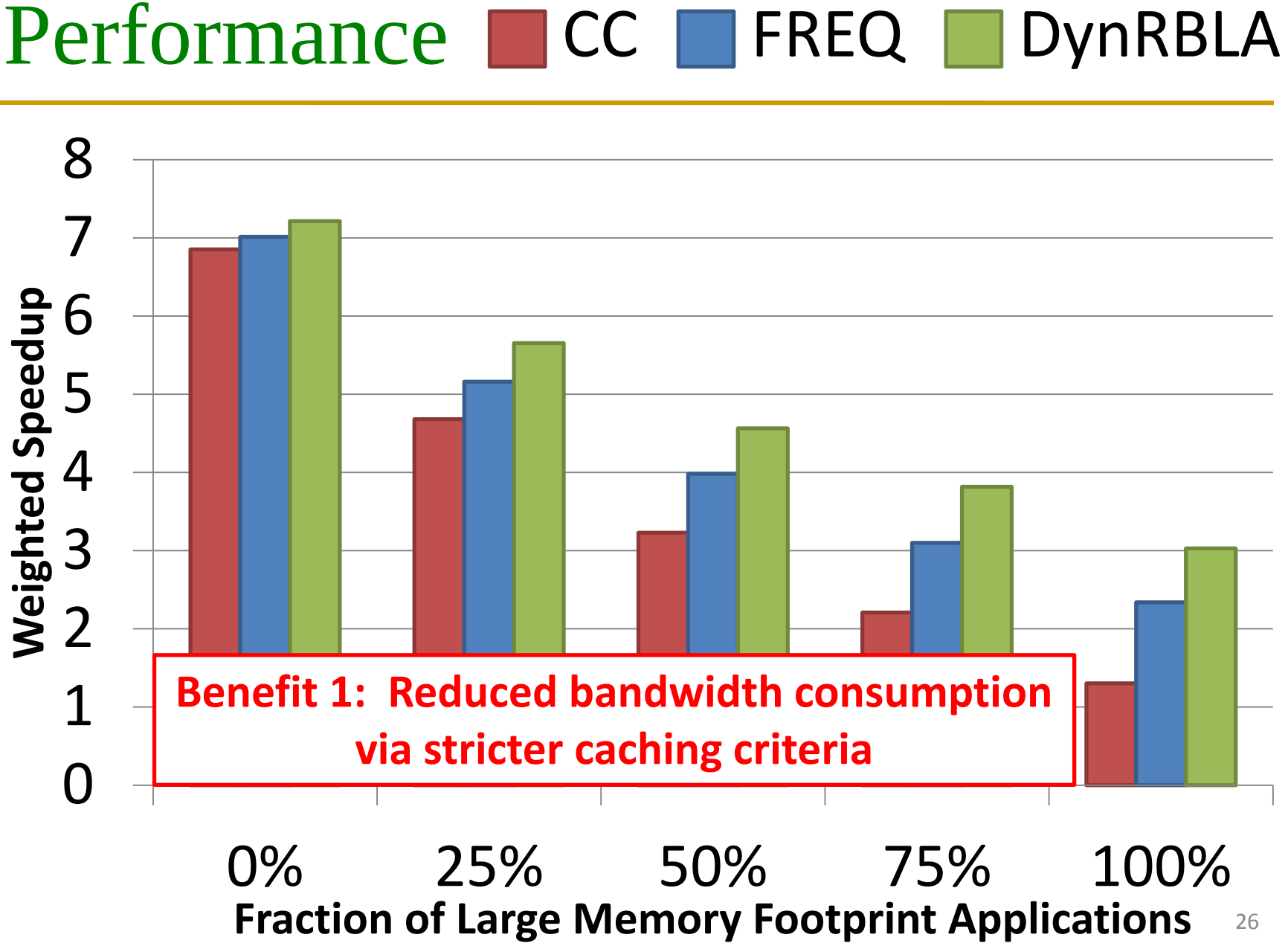
- **CC**: Conventional caching
- **FREQ**: Frequency-based caching
- **DynRBLA**: Adaptive, row buffer locality-aware caching

- **Weighted speedup (performance)** = sum of speedups versus when run alone
- **Max slowdown (fairness)** = largest slowdown experienced by any thread

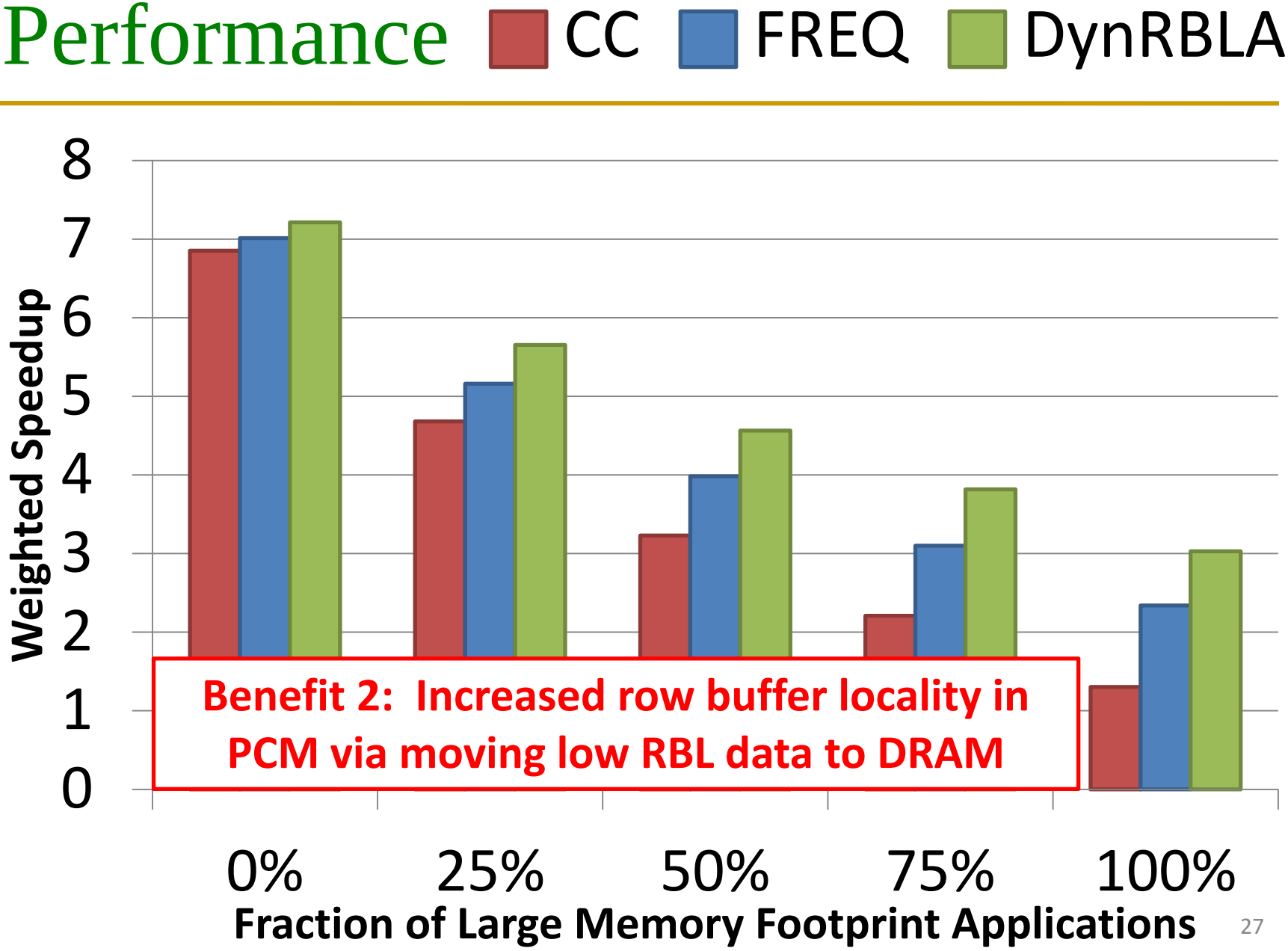
Performance



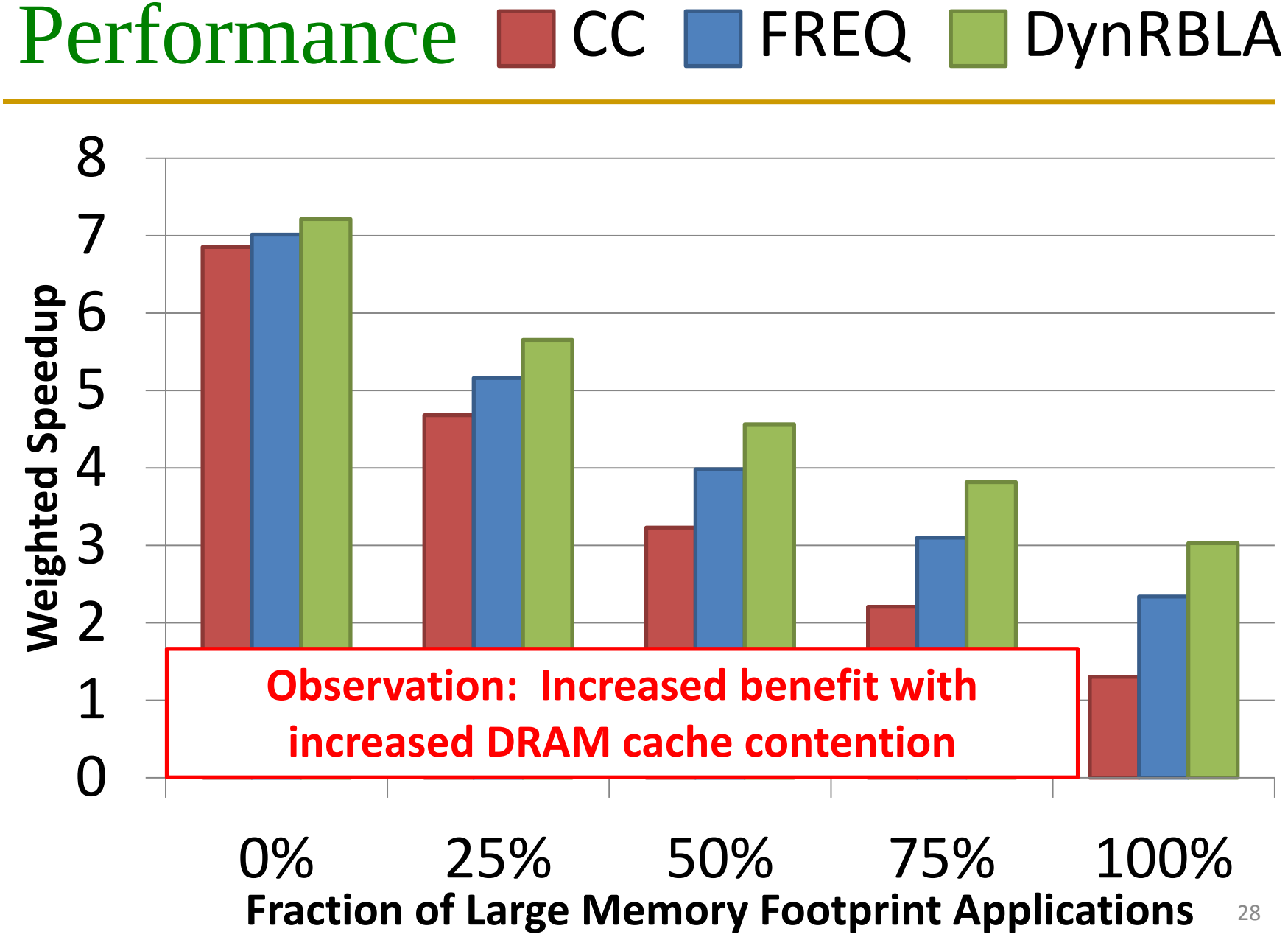
Performance



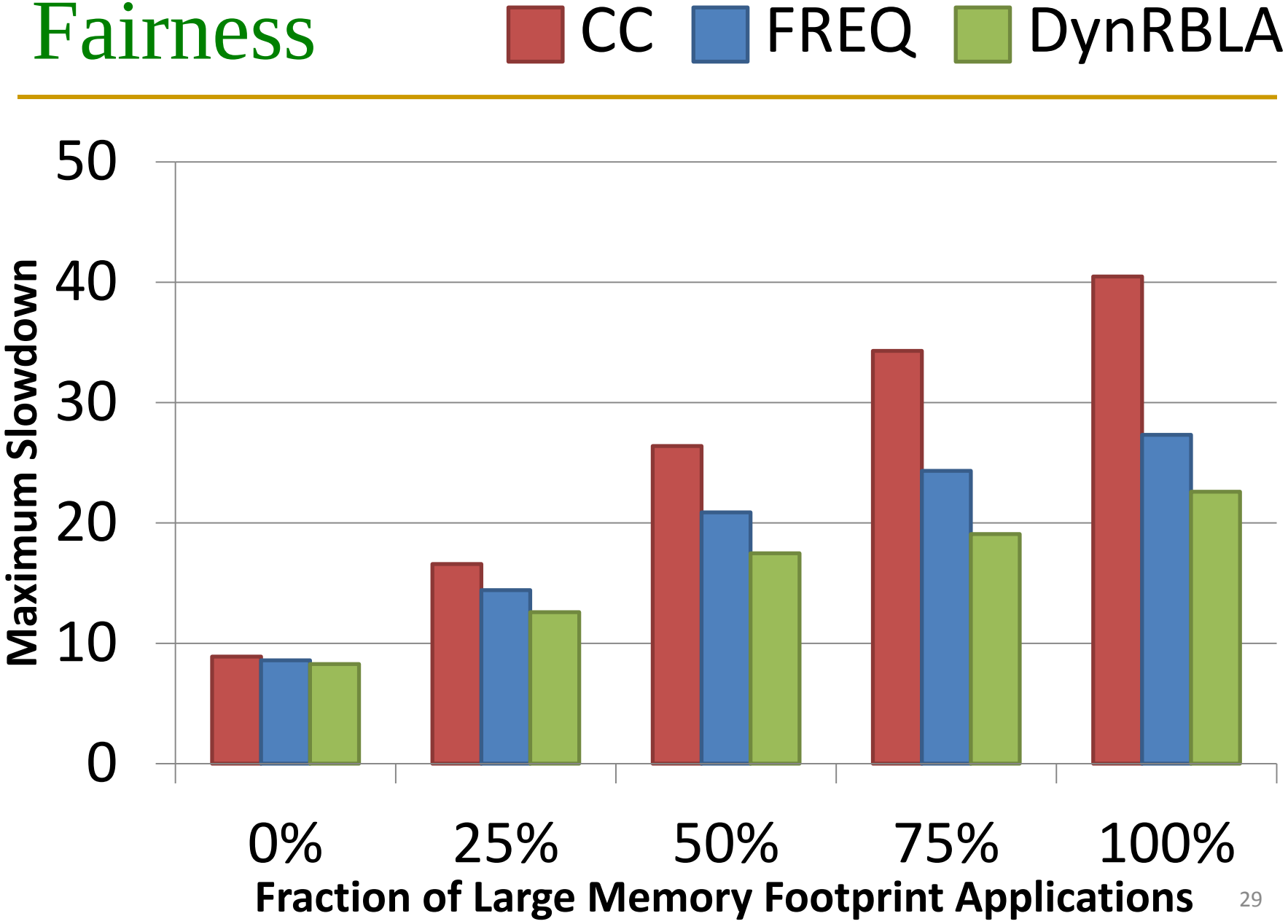
Performance



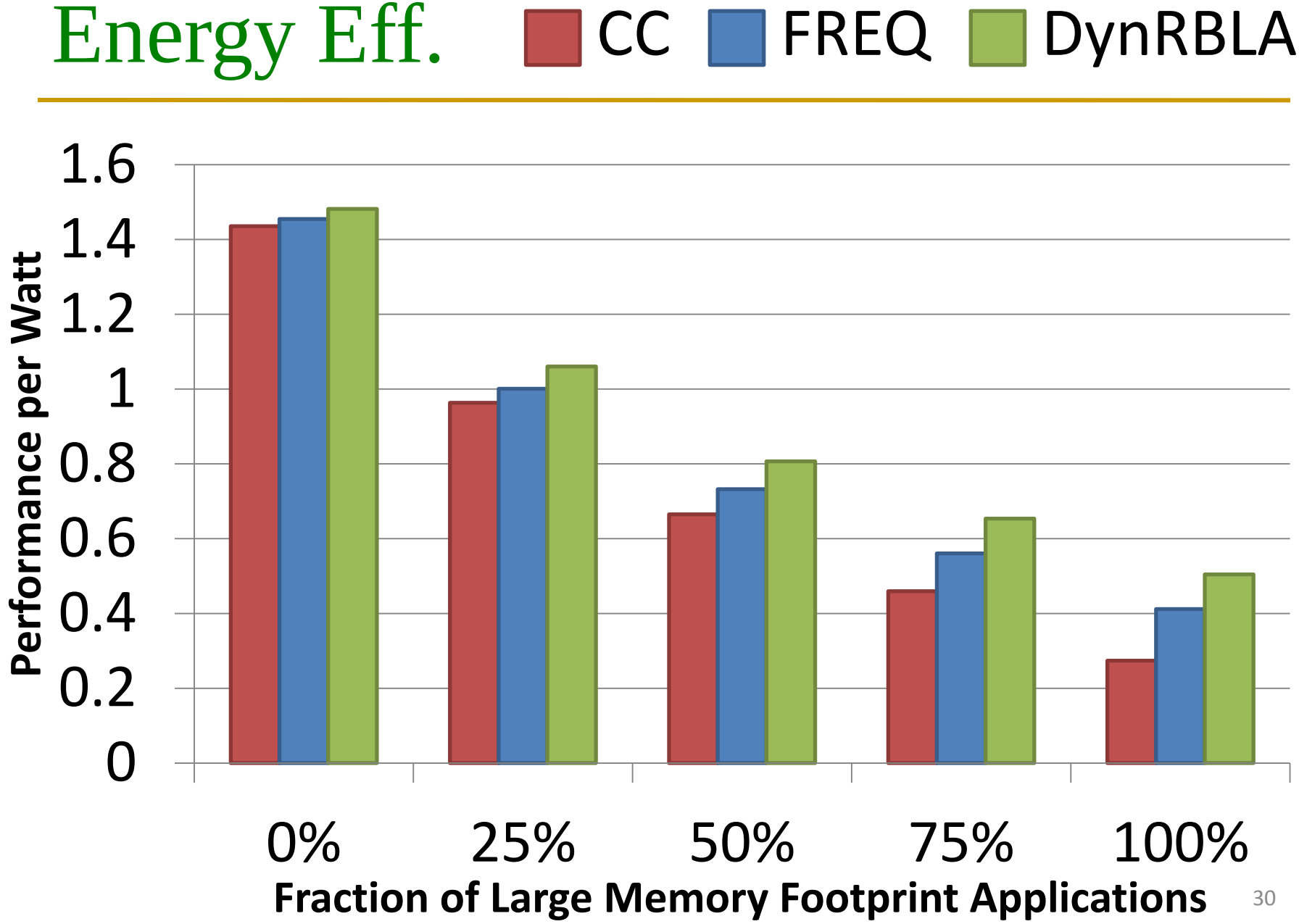
Performance



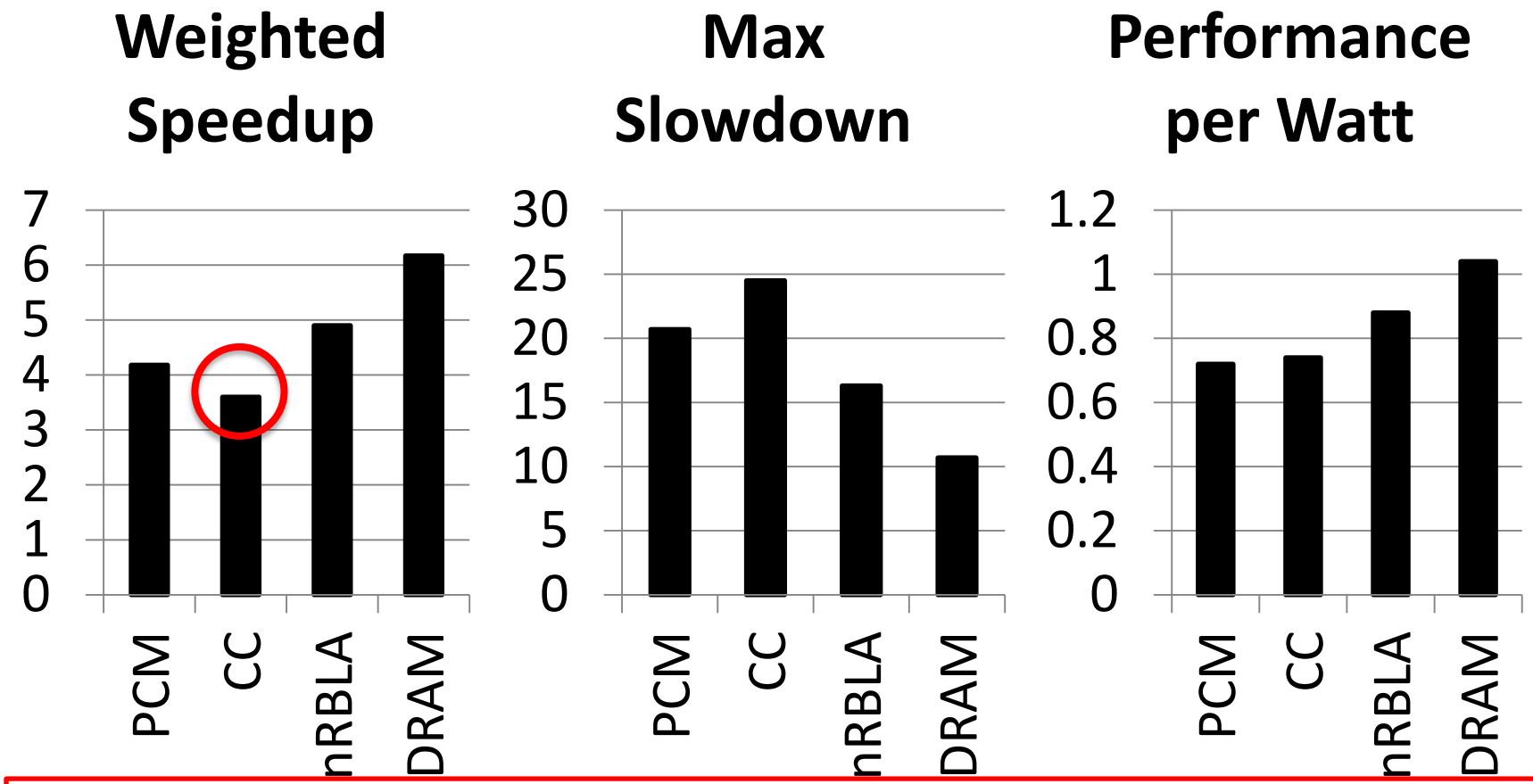
Fairness


CC
FREQ
DynRBLA
Maximum Slowdown
50
40
30
20
10
0
0%
25%
50%
75%
100%
Fraction of Large Memory Footprint Applications

Energy Eff.

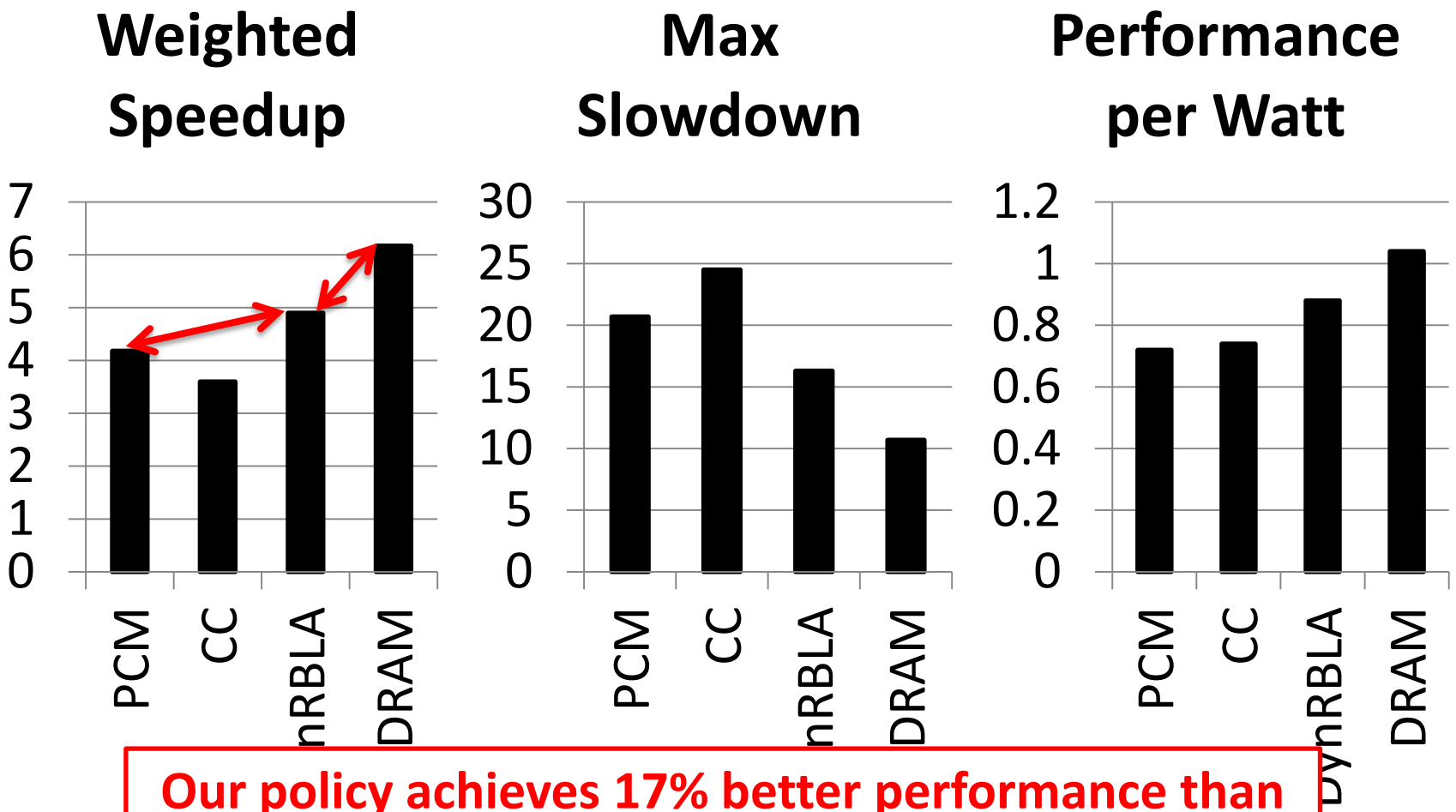

CC
FREQ
DynRBLA
Performance per Watt
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0%
25%
50%
75%
100%
Fraction of Large Memory Footprint Applications

Compared to All PCM/DRAM



Observation: Inefficient caching policies may achieve worse performance than PCM due to increased bandwidth contention

Compared to All PCM/DRAM



Our policy achieves 17% better performance than all PCM, within 21% of all DRAM performance

Ongoing Work

- How to make the most of multi-level bits per PCM cell: New data mapping schemes
- Further reducing the bandwidth problem: Adaptive data migration granularity
- Achieving the best of performance and fairness: Quality of service as a first-class data placement metric

Summary

- Demand for huge main memory capacity
  - PCM offers greater density than DRAM
  - Hybrid memories achieve the best of both
- We identify row buffer locality as a key metric for caching decisions and design a dynamic policy that caches rows with low RBL and high reuse in DRAM
- Helps enable efficient hybrid main memories

Thank you! Questions?

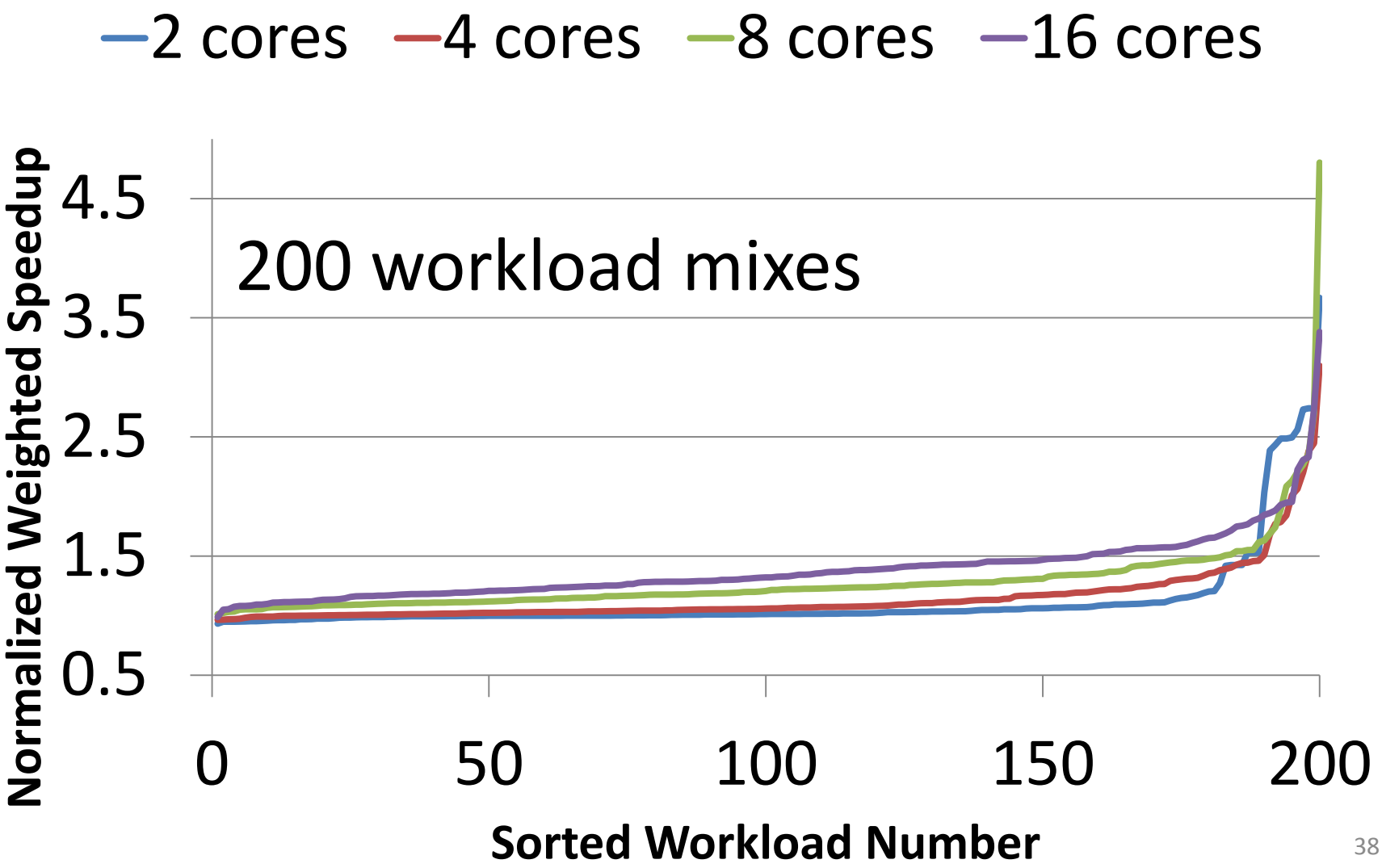
Backup Slides

Projected PCM Characteristics (~2013)

32 nm	DRAM	PCM	Relative to DRAM
Cell size	6 F^2	0.5–2 F^2	**3–12× denser**
Read latency	60 ns	300–800 ns	**6–13× slower**
Write latency	60 ns	1400 ns	**24× slower**
Read energy	1.2 pJ/bit	2.5 pJ/bit	**2× more energy**
Write energy	0.39 pJ/bit	16.8 pJ/bit	**40× more energy**
Durability	N/A	10^6–10^8 writes	**Limited lifetime**

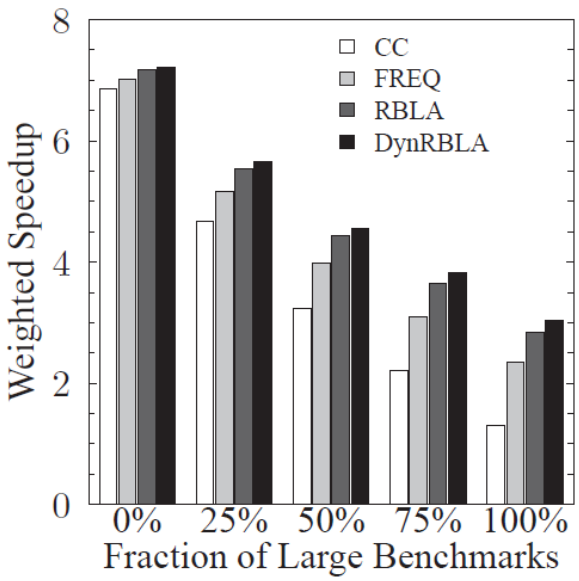
[Mohan, HPTS ’09; Lee+, ISCA ’09]

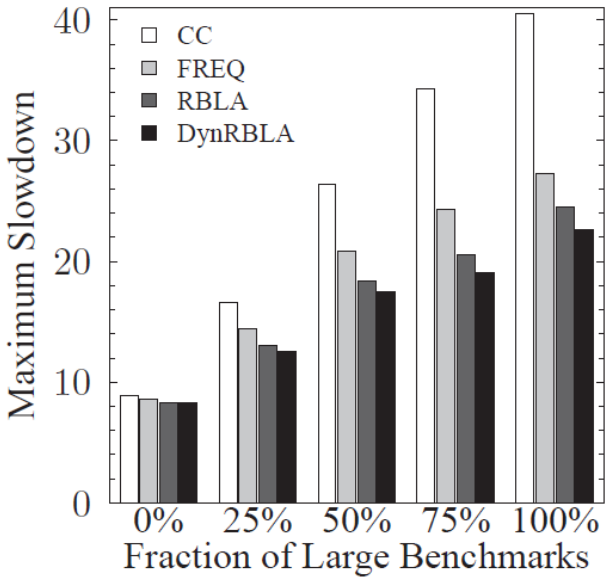
Scalability

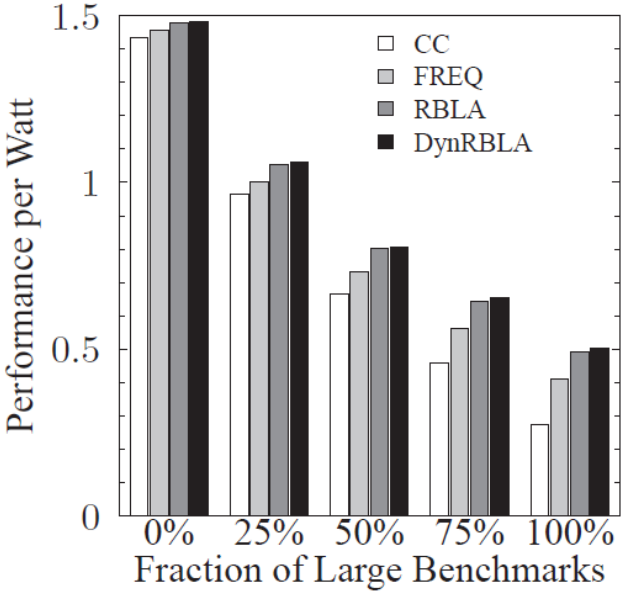


Results: Multi-core

- Performance, fairness, energy-efficiency

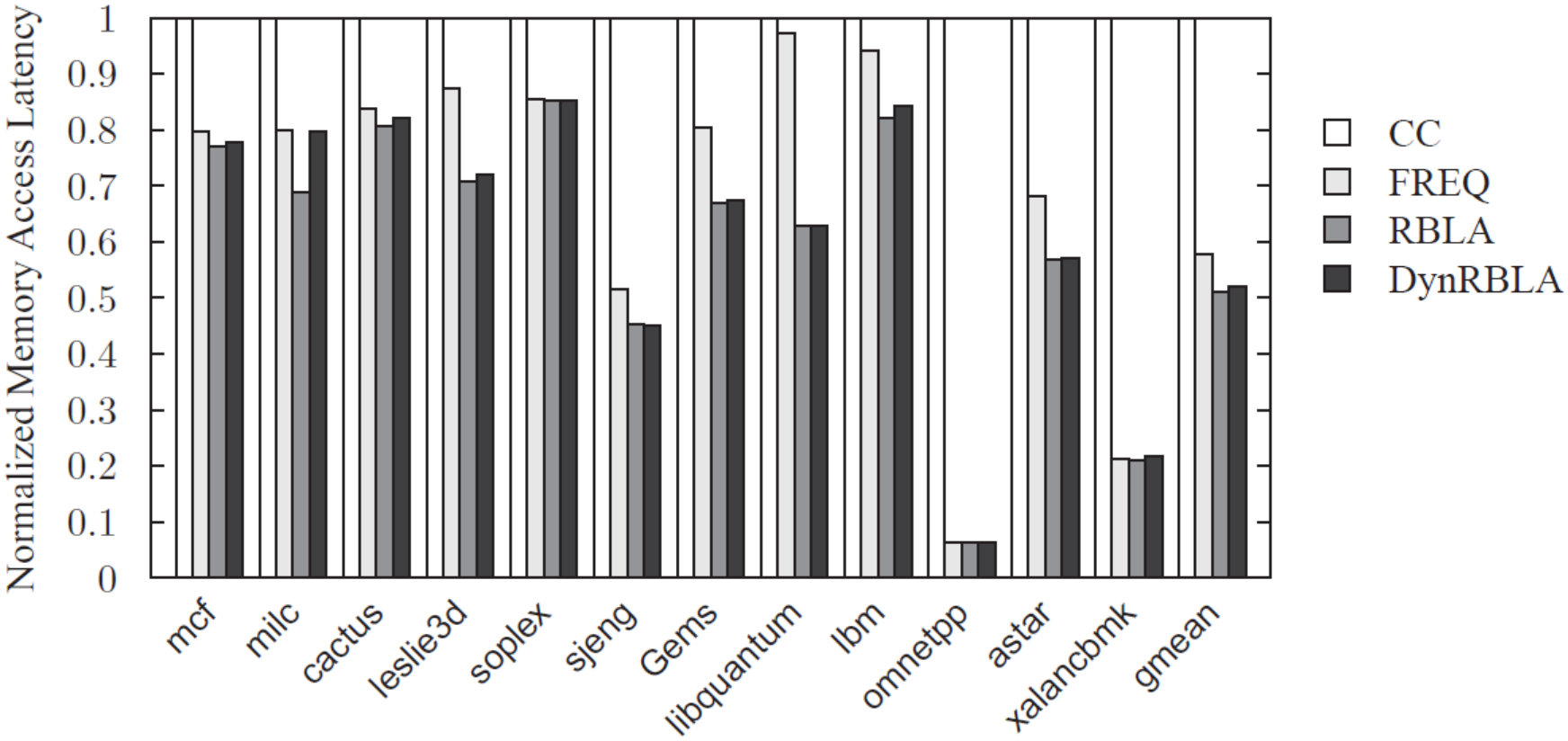






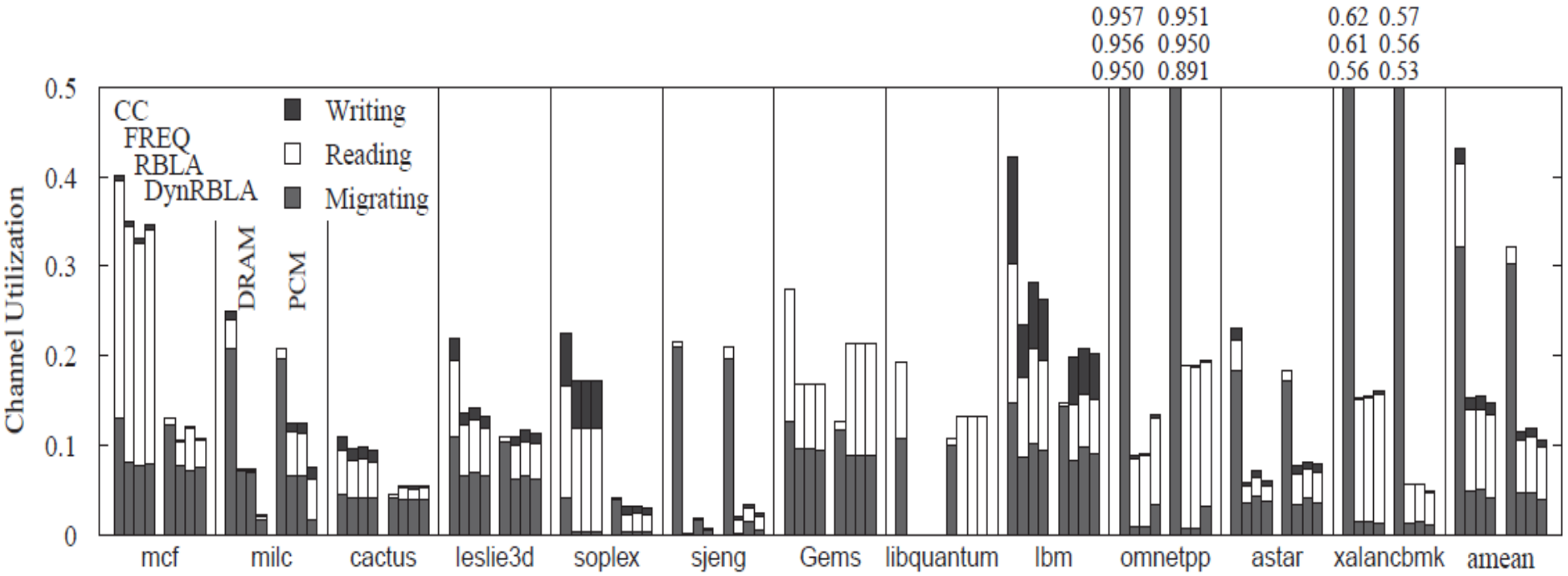
Results: Single-core

- Memory access latency



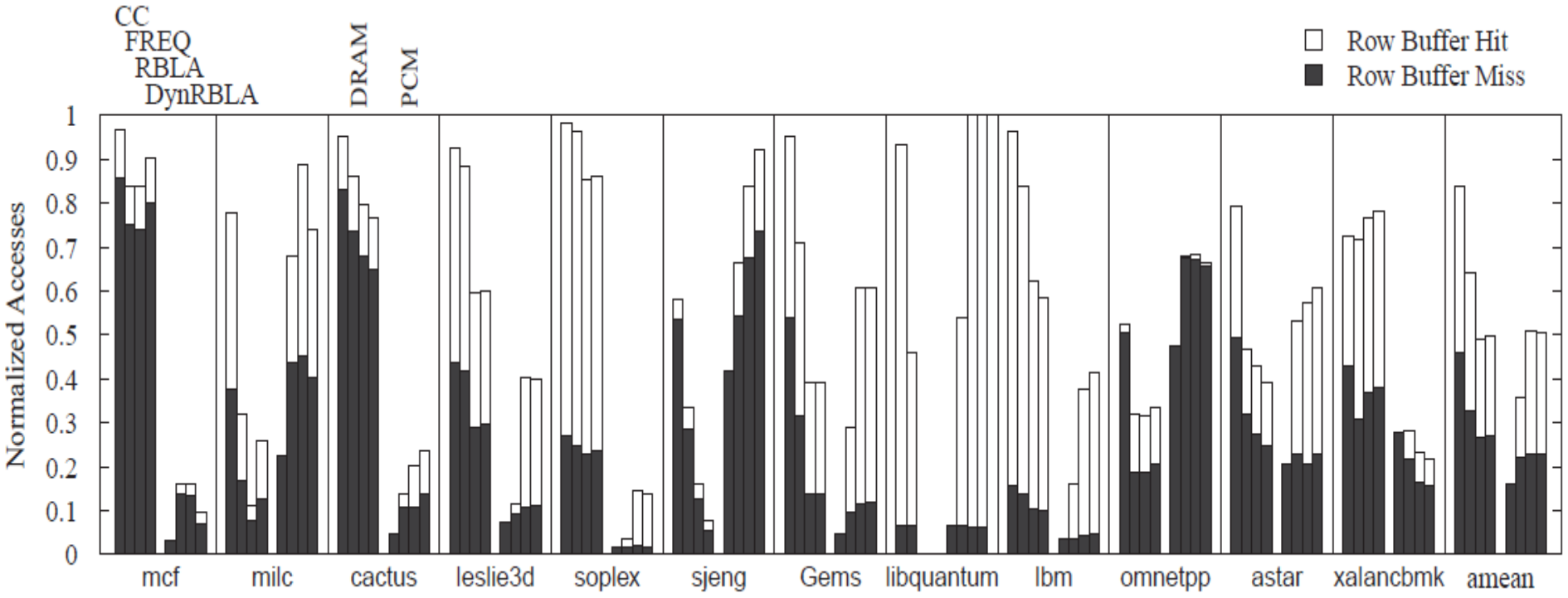
Results: Single-core

- Memory channel utilization



Results: Single-core

- Memory load balancing



Related Techniques

PCM Latency

DRAM Cache Size

Cost-Benefit Analysis

- Each quantum, we measure the *first-order* costs and benefits of the current *A* threshold
  - Cost = cycles of bus contention due to migrations
  - Benefit = cycles saved at the banks by servicing a request in DRAM versus PCM
- Cost = Migrations $\times$ $t_{migration}$
- Benefit = $Reads_{DRAM} \times (t_{read,PCM} - t_{read,DRAM})$
  $+ Writes_{DRAM} \times (t_{write,PCM} - t_{write,DRAM})$

Cost-Benefit Maximization Algorithm

Each quantum (10 million cycles):

```
1  Net = Benefit − Cost                    // net benefit
2  if Net < 0 then                         // too many migrations?
3      A++                                 //   increase threshold
4  else
5      if Net > PreviousNet then           // last A beneficial?
6          A++                             //   increasing benefit?
7      else                                //   try next A
8          A--                             // decreasing benefit
9      end                                 //  too strict, reduce
10 end
11 PreviousNet = Net
```

Methodology

- Core model
  - 3-wide issue with 128-entry instruction window
  - 32 KB L1 D-cache per core
  - 512 KB shared L2 cache per core
- Memory model
  - 16 MB DRAM / 512 MB PCM per core
    - Scaled based on workload trace size and access patterns to be smaller than working set
  - DDR3 800 MHz, single channel, 8 banks per device
  - Row buffer hit: 40 ns
  - Row buffer miss: 80 ns (DRAM); 128, 368 ns (PCM)
  - Migrate data at 2 KB row granularity

Implementation/Hardware Cost

- Requires a tag store in memory controller
  - We currently assume 36 KB of storage per 16 MB of DRAM
  - We are investigating ways to mitigate this overhead
- Requires a *statistics store*
  - To keep track of *accesses* and *misses*