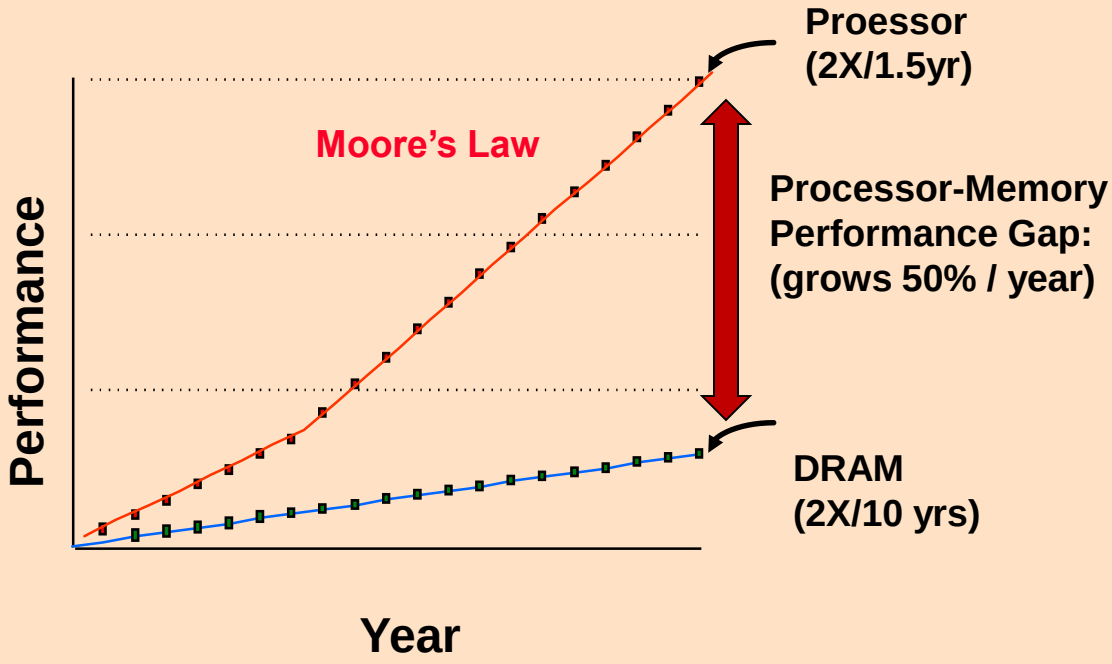
Memory Hierarchy Design for Multicore Architectures

R. Govindarajan
Computer Science & Automation
Indian Institute of Science
Bangalore , India
govind@iisc.ac.in

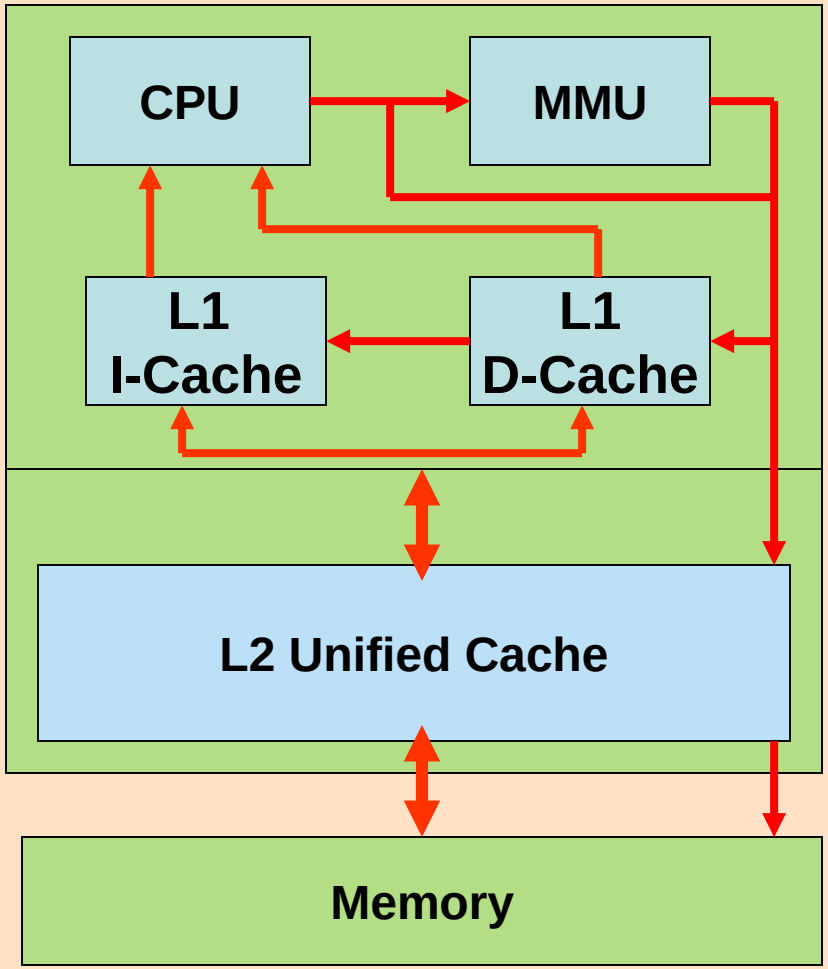
July 2019

Memory Performance

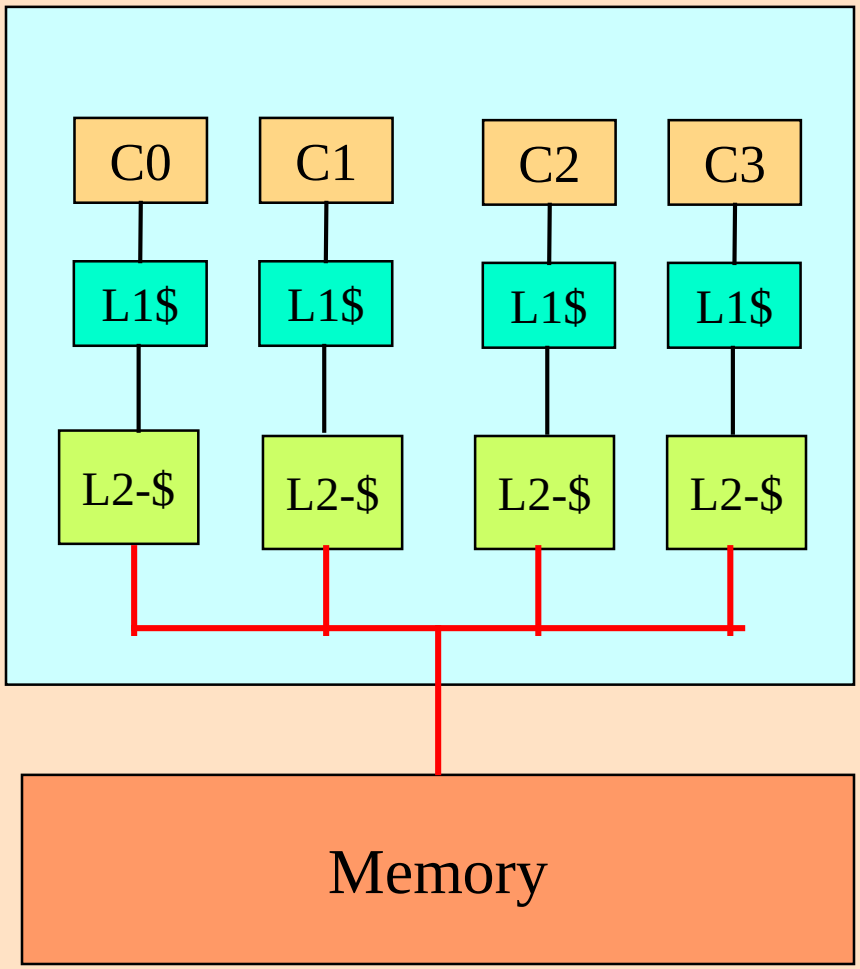
- Memory Wall [McKee'94]
  - CPU-Memory speed disparity
  - 100's of cycles for off-chip access

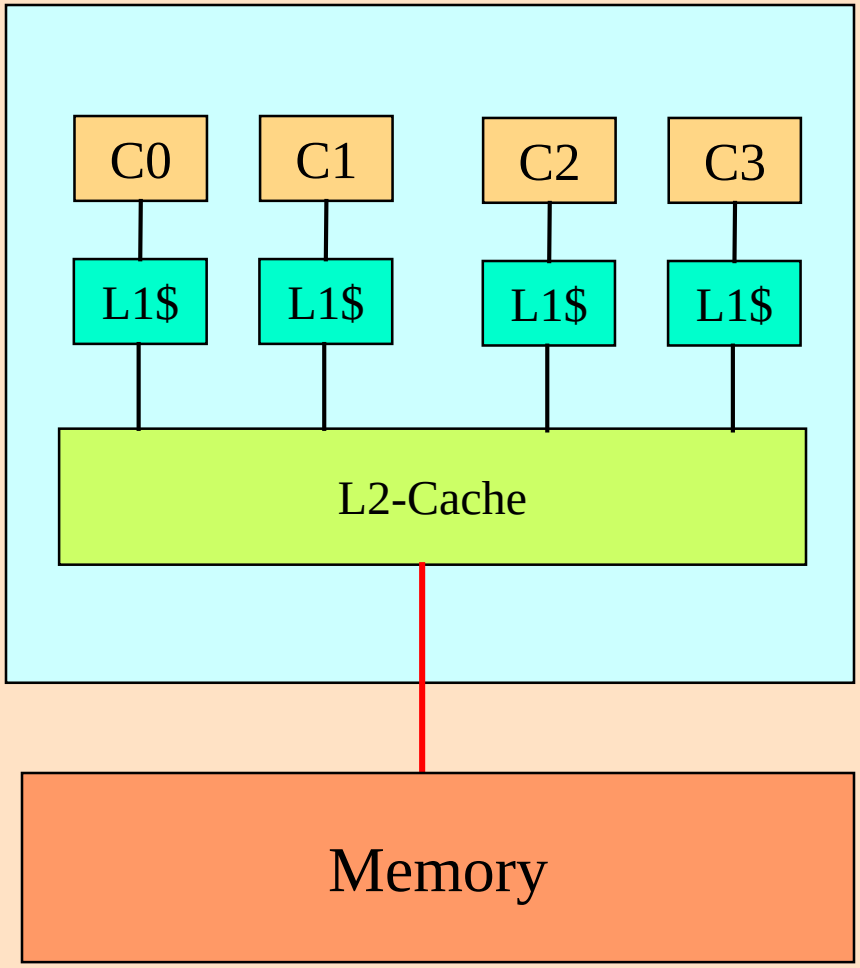


Memory Hierarchy

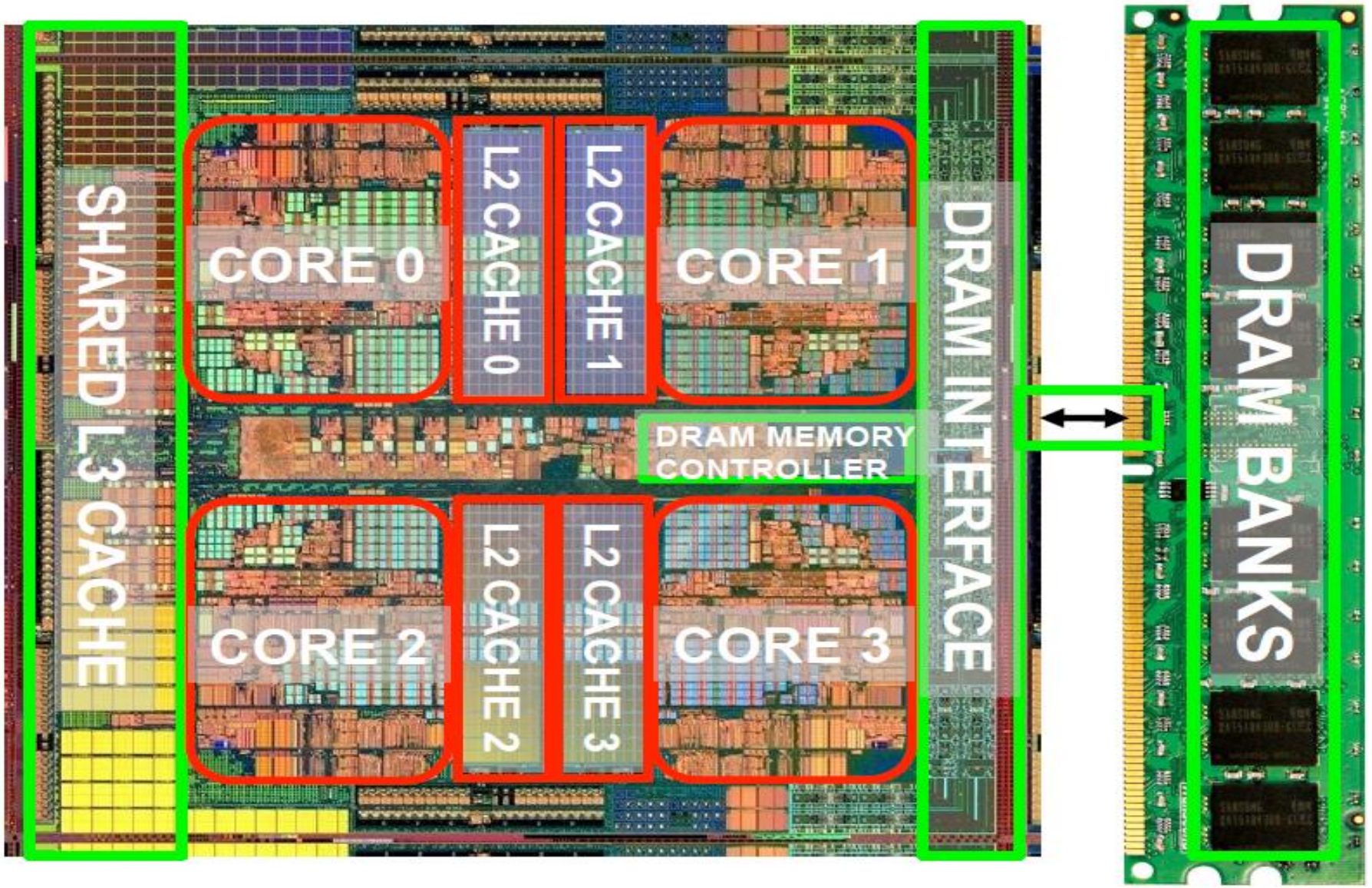


Memory Hierarchy in Multicore

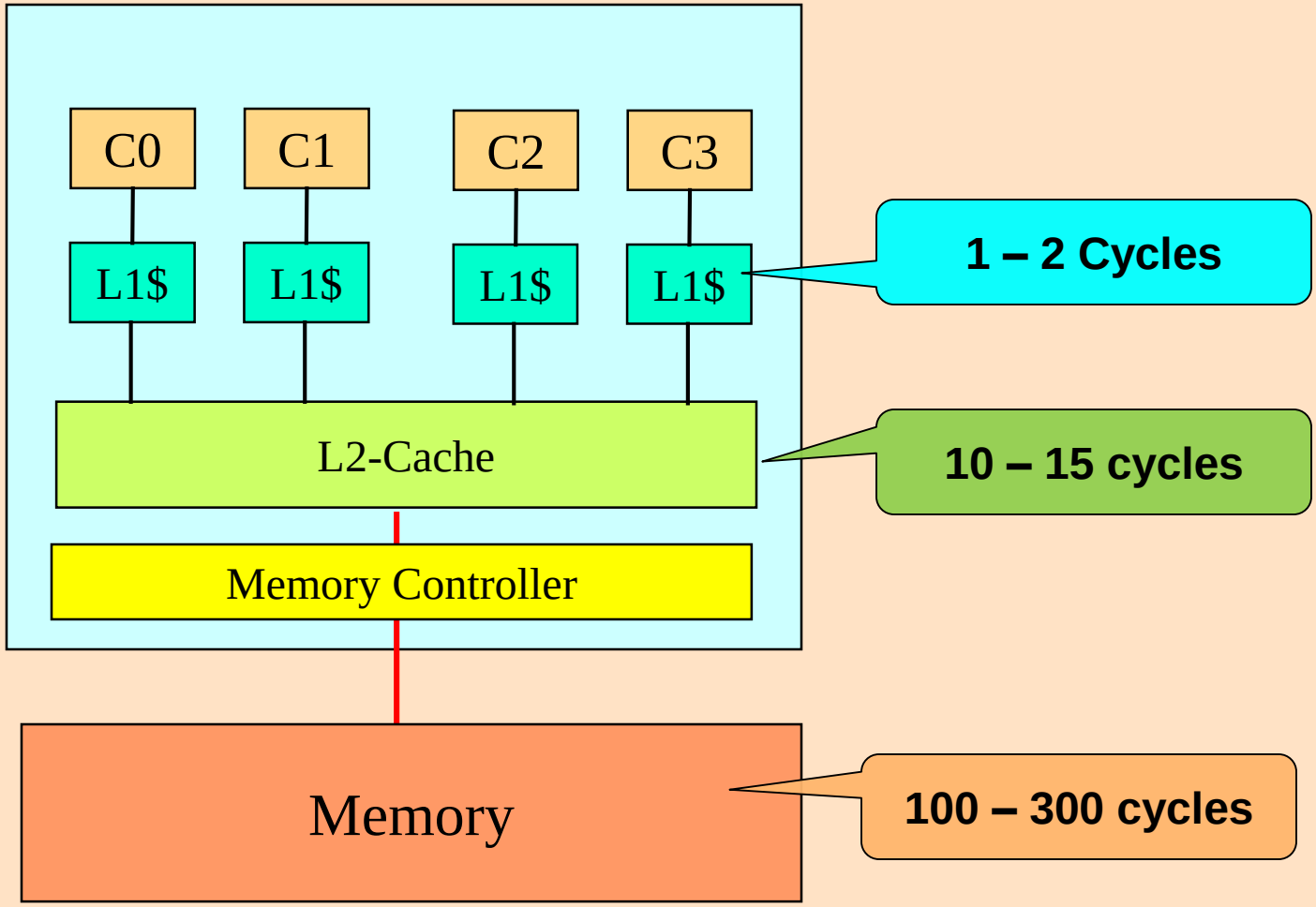




Memory Hierarchy in Multicore



Memory Hierarchy in Multicore



Memory Performance

- Memory Wall [McKee'94]
  - CPU-Memory speed disparity
  - 100's of cycles for off-chip access
- Bandwidth Wall [ISCA'09]
  - More cores and limited off-chip bandwidth
  - Cores double every 18 months
  - Pincount grows only by 10%

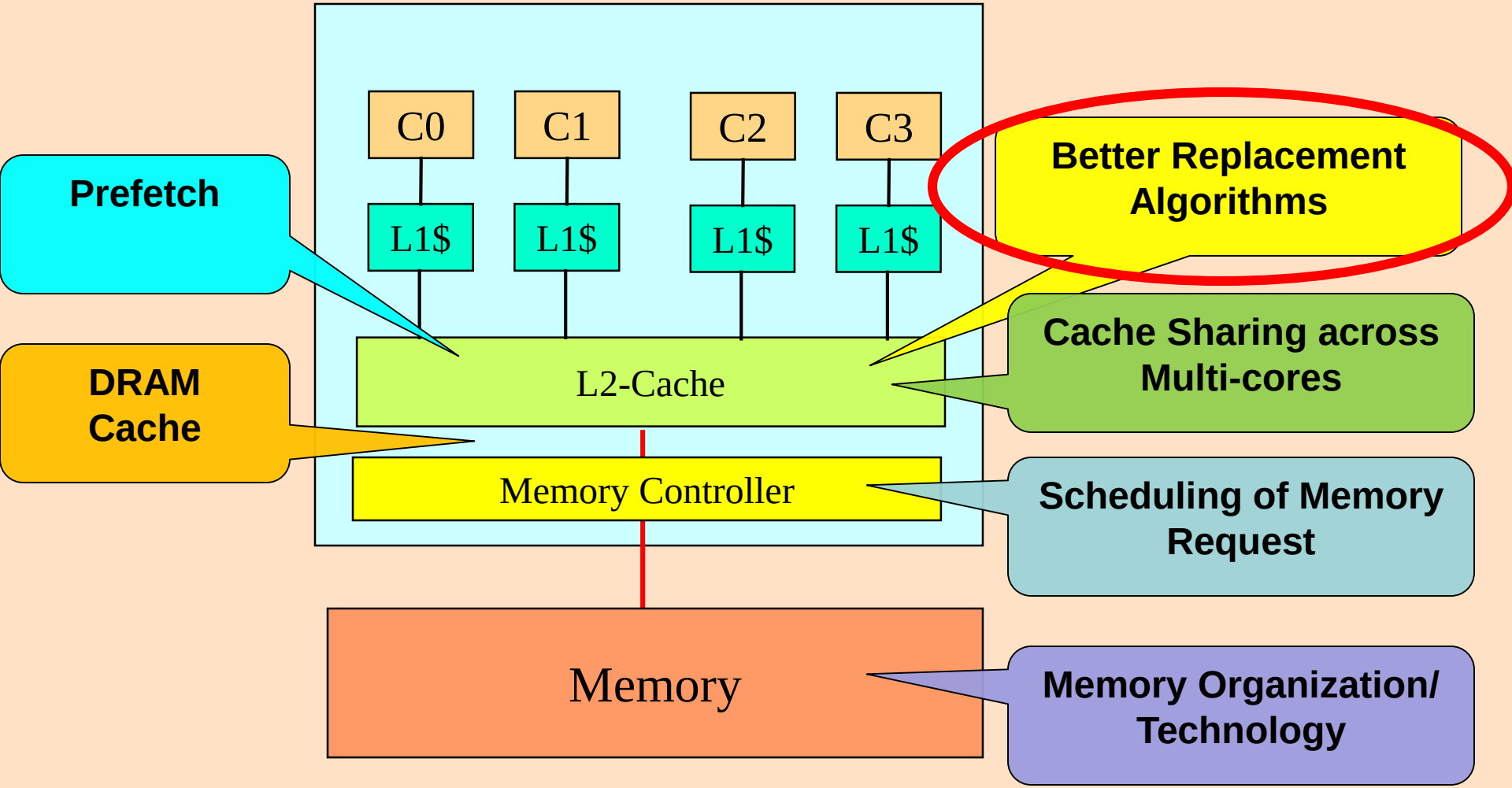
Off-chip accesses are expensive !

Memory System Performance is Critical

Experimental Evaluation Approach

- Cache Performance
  - Trace-driven methodology
  - Cache simulator (Dinero)
  - Hit Rate, MPKI *(are they same?)*
- MemoryPerformance
  - Trace-driven simulation
  - DRAMSim, Cactii
- System Performance
  - Execution-driven simulation
  - Trace-Driven simulation
  - SimpleScalar, Tejas, M5, GEM-5, Sniper, PIN, …

Research Issues in Multicore Memory Hierarchy



L2/L3 Cache Replacement

- L2 or L3 cache is larger (~MB) and has higher associativity
- High Associativity $\Rightarrow$ replacement policy crucial to performance
- L1 cache services temporal accesses $\Rightarrow$ Locality filtered by lower-level caches (L1 or L2) $\Rightarrow$ LRU replacement inefficient
- Miss penalty long-enough for sophisticated replacement policy

Replacement Algorithms for L2 / L3

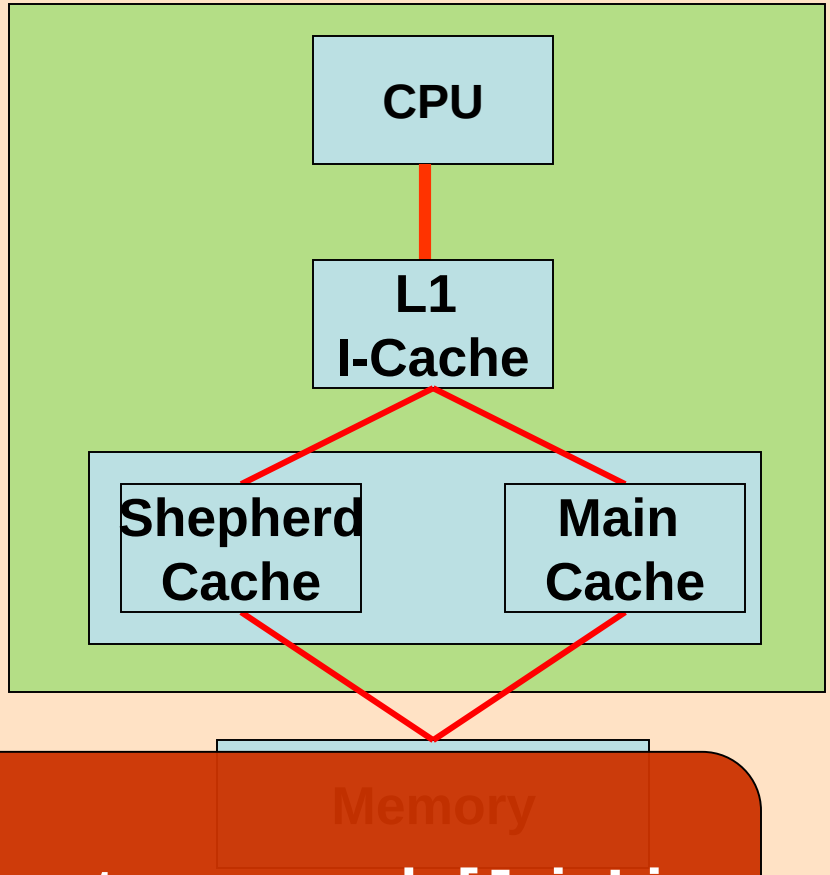
- Least Recently Used (LRU)
- V-way Associative Cache
- Indirect Index Cache (~ Fully associative)
- Dynamic Insertion Policy
  - choose between LRU and BIP (insertion at LRU or MRU)
- Pseudo Insertion and Promotion Policy (PIPP)
- etc. etc.

Can OPTIMAL replacement be done?

Optimal Replacement: On a miss replace the candidate to which an access is least imminent !

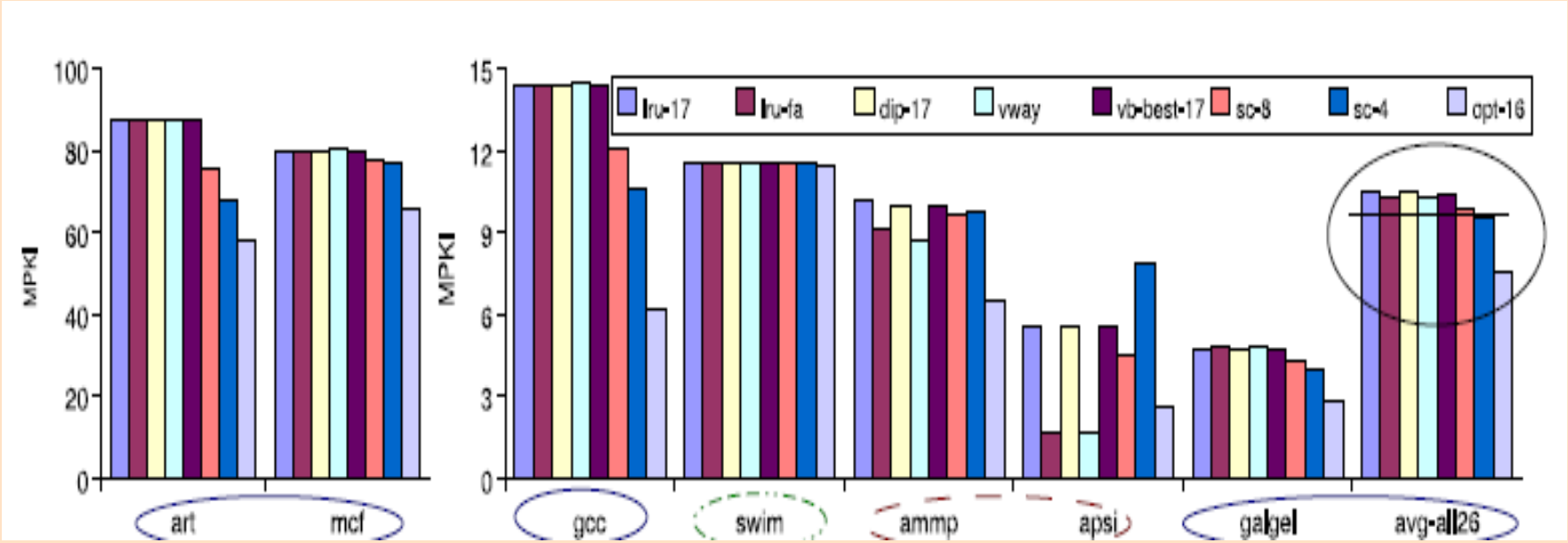
Shepherd Cache [Micro 2007]

- OPT requires lookahead for least imminent line ⇒ Use part of cache to emulate OPT for remaining cache
- Part of L2 Cache, used as FIFO buffer, to track imminence of new lines
- Lines flowing out of Shepherd Cache move to Main L2 with LRU replacement



More recent OPTIMAL replacement approach [JainLin, ISCA 2016]

Performance of Shepherd Cache



- On average both SC-4 and SC-8 out-performs LRU, DIP, v-way, Fully-Associative, and victim by 4-10%

NUcache [HPCA 2011]

- Can we improve LLC hits by making them Next-use aware?
  - Next-Use : Distance betn. Eviction and next Access

Miss Stream : A, A1, A2, … A10, X

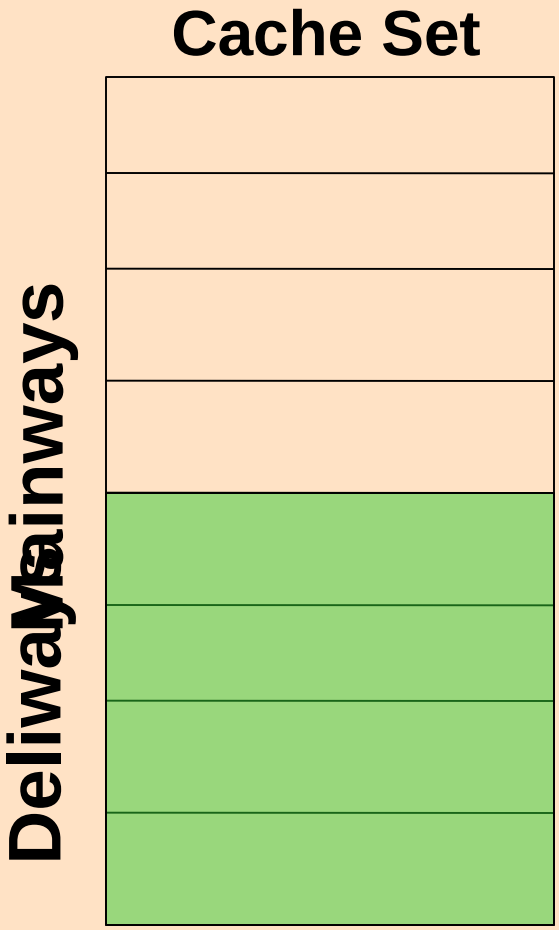
Evictions : X, Y1, Y2, … Y10, M

NextUse Dist = 10

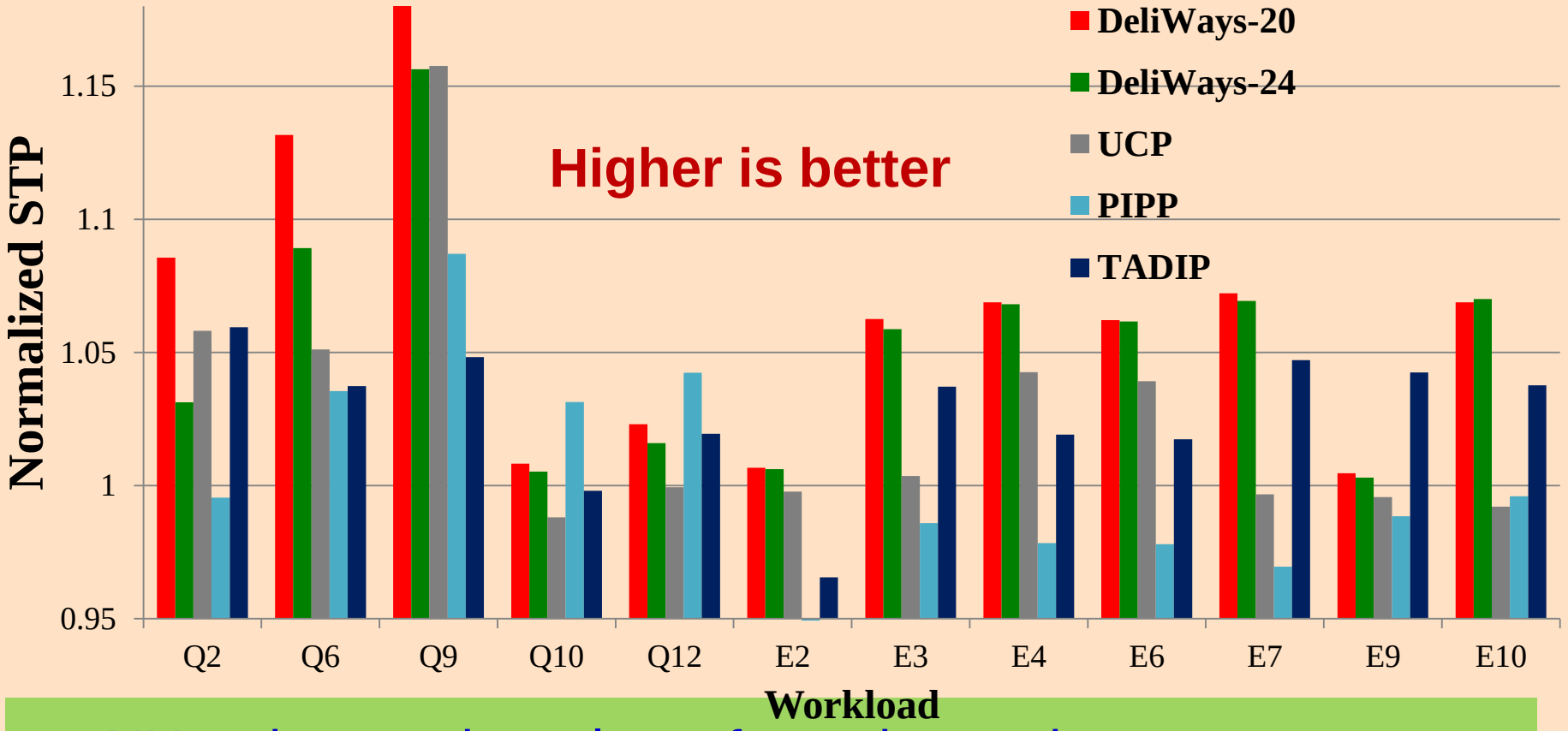
If X has to be retained longer for additional hits, it should be retained for at least next 10 misses

NUCache Organization

- Logical partitioning of the associativity of the cache.
- Each cache set split into MainWays and DeliWays.
- DeliWays are used on demand to retain selected lines longer
- Which lines should go to Deliways?
  - Lines of Delinquent PC, whose collective Next-Use will turn Deliways to be hits!

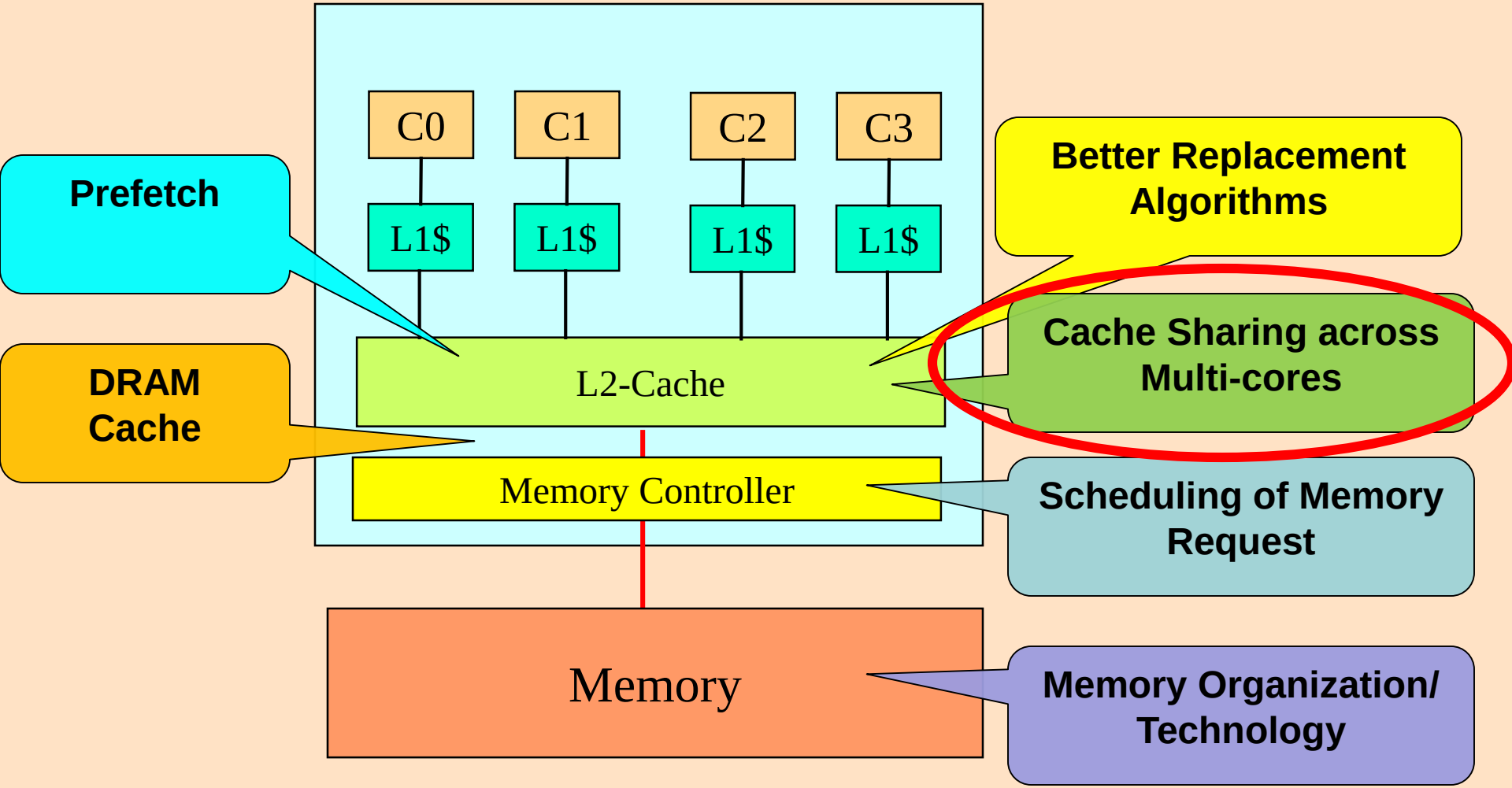


Comparison with Other Schemes



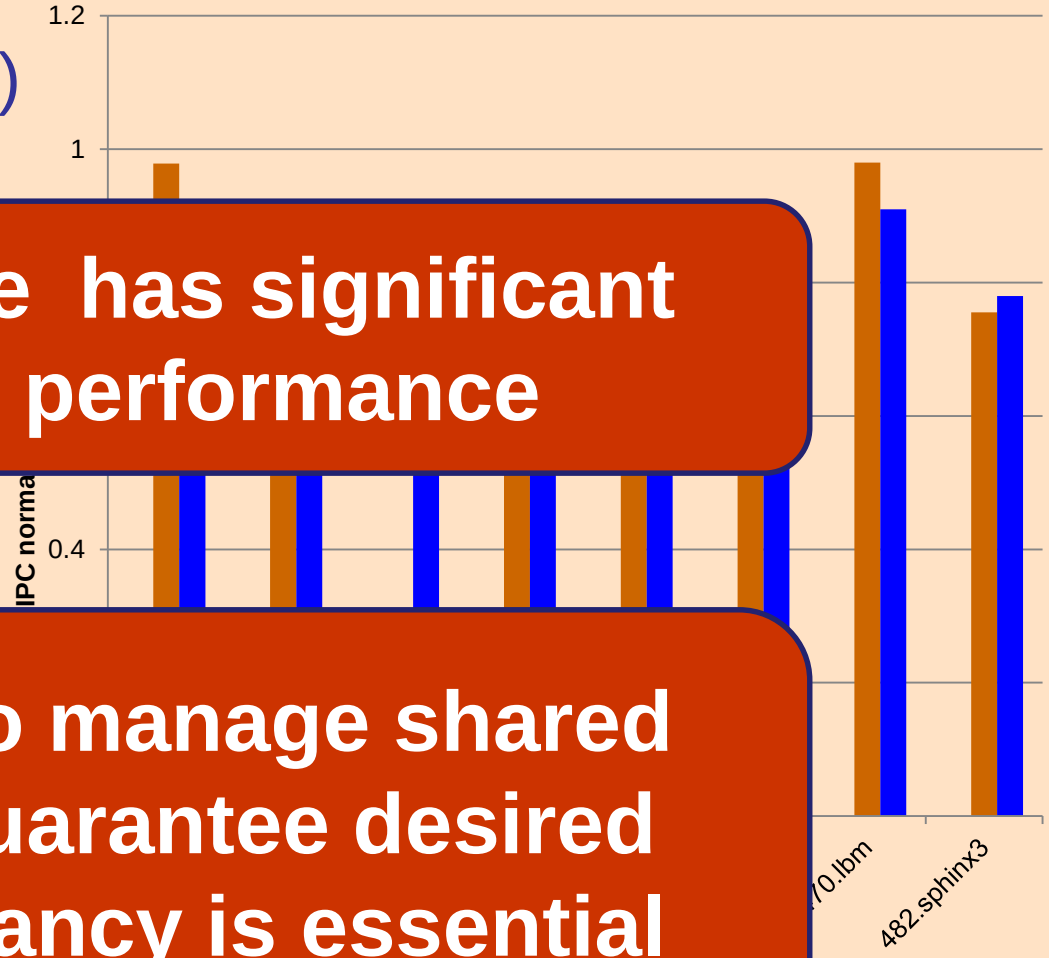
- NUcache consistently performs better than
  - Utility based Cache Partitioning
  - Promotion/Insertion Psuedo Partitioning
  - Thread Aware Dual Insertion Policy

Research Issues in Multicore Memory Hierarchy



Impact of Cache Sharing

- Last Level Cache (LLC) is shared across all cores
- A
  - s
  - s
    - Significant slowdown in some programs
- A
  - o
  - p
    -
    -



Shared Cache has significant impact on performance

Framework to manage shared cache and guarantee desired cache occupancy is essential

Shared Cache Management

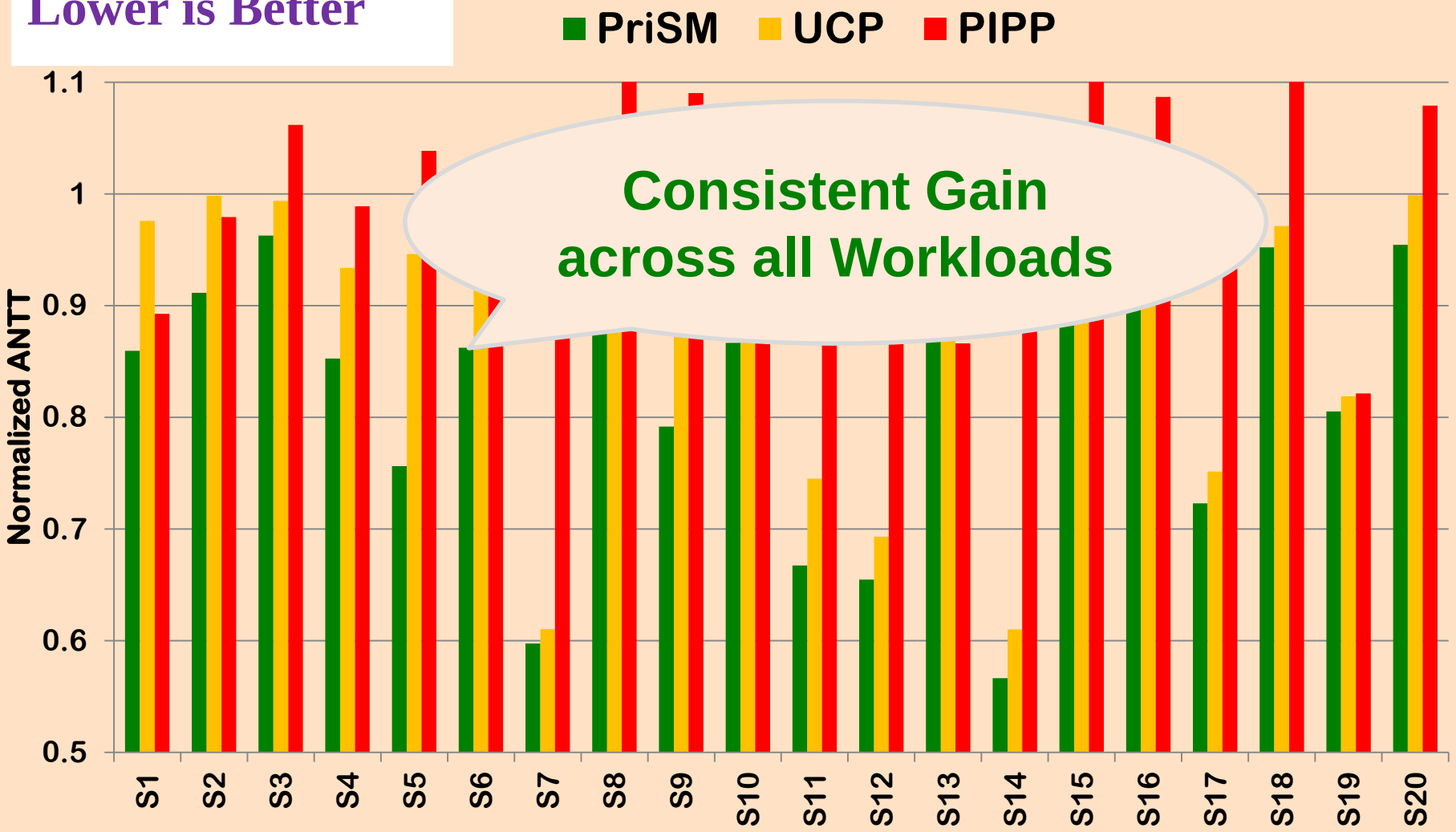
- Way Partitioning
  - Partition associativity of cache (based on some objective)
  - Modified Replacement
    - Step1 -- Identify Victim Core
    - Step2 – Identify Victim block belonging to Victim core
  - UCP, PIPP, …
- Way partitioning is at coarse granularity
  - Granularity = 1/K, for K-way associativity (e.g., 1/16)
  - Does finer granularity (block level) help (e.g., 1/16384) and be achieved ?

Probabilistic Shared Cache Management (PriSM) [ISCA-2012]

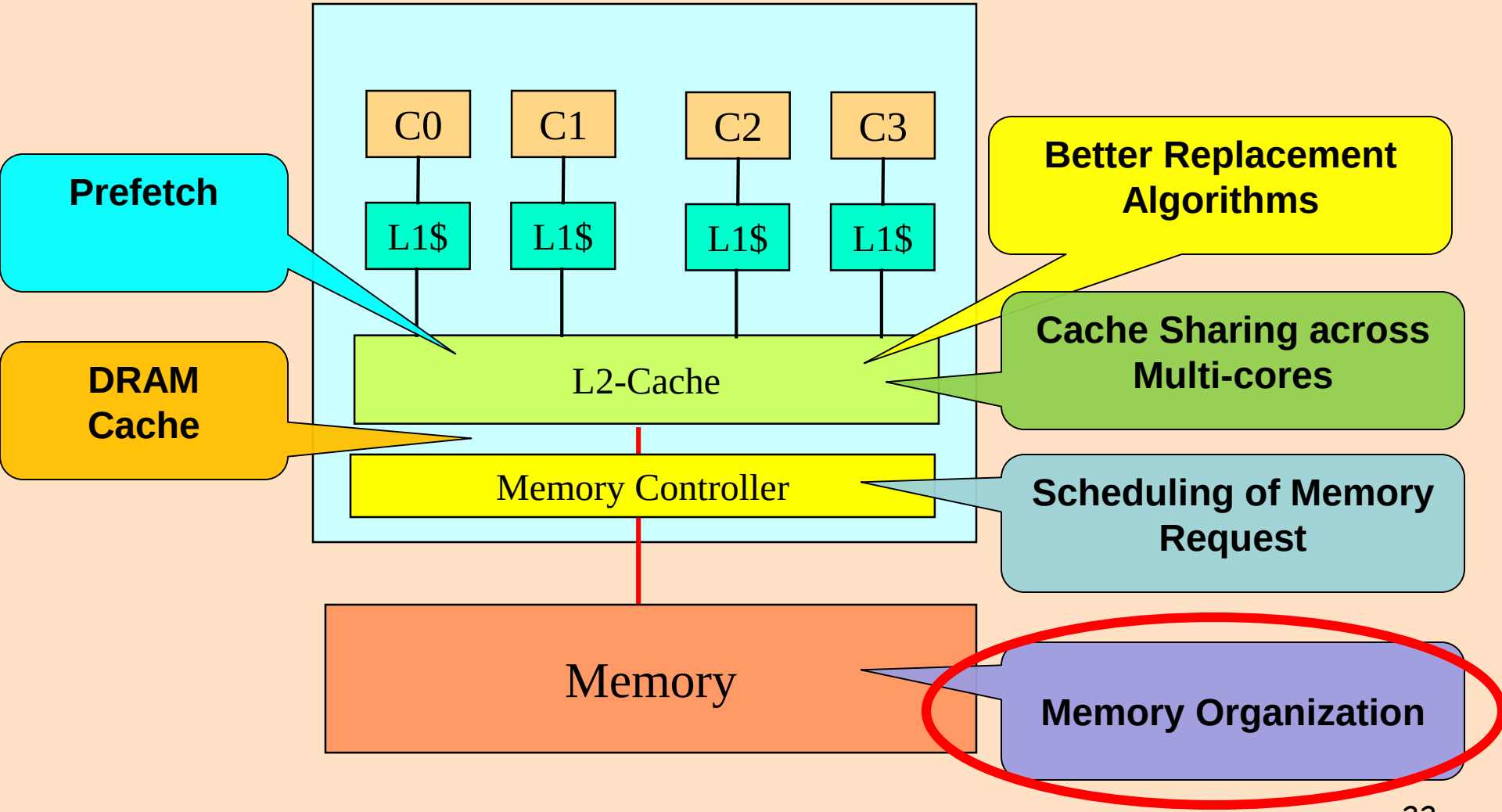
- Eviction probability associated with each program sharing the cache
  - Used during replacement
  - Eviction Probabilities determined based on fine-grain cache occupancy for the core
  - Cache occupancy determined based on target goals: maximize overall IPC, fairness, QoS, …
- Replacement
  - Step 1 – Generate Victim-Core ID based on Eviction Probability Distribution
  - Step 2 – Identify Victim block
    - Use baseline replacement
    - Victim belonging to victim-core identified above

Performance of PriSM: Hit-Maximization

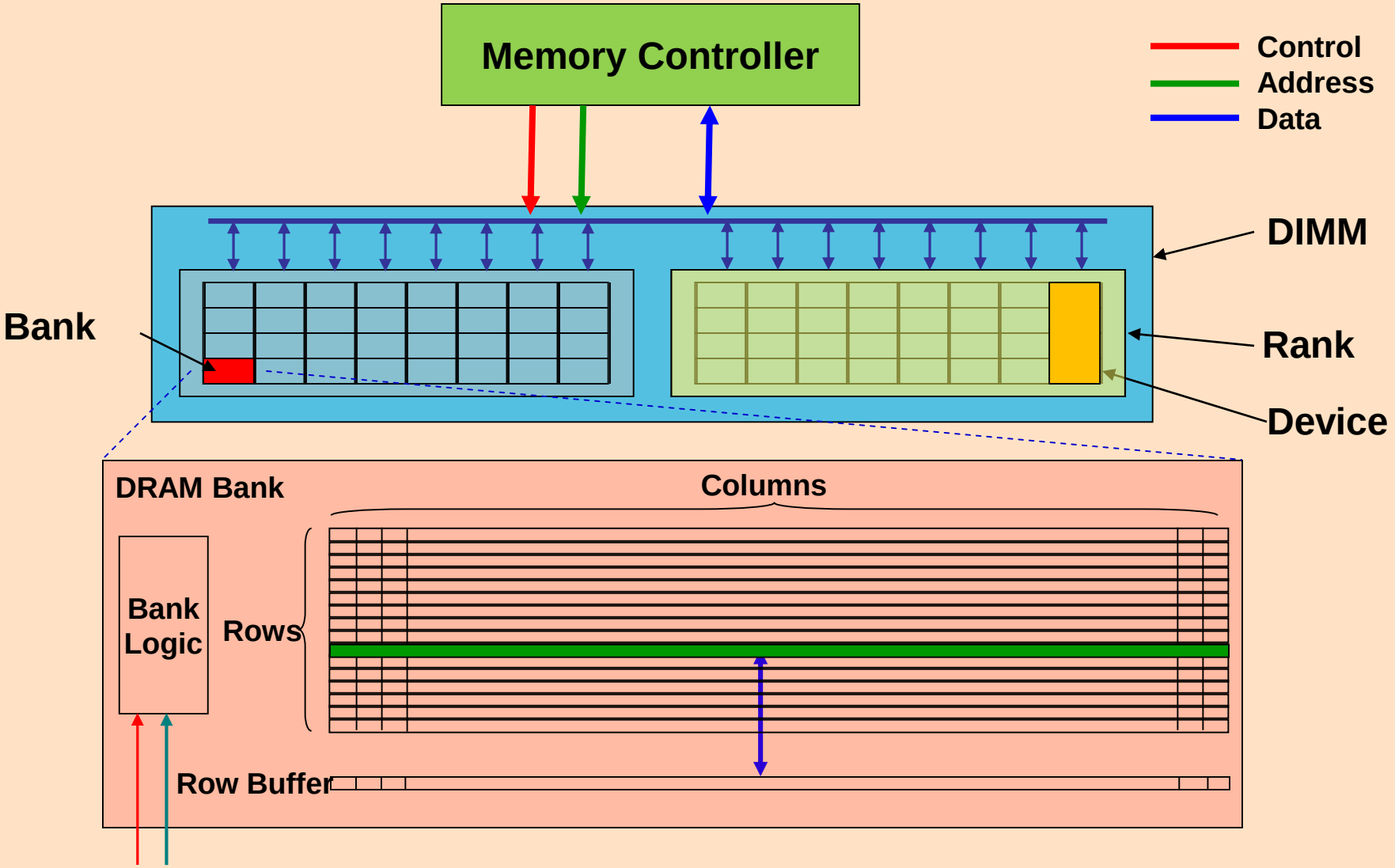
Lower is Better



Research Issues in Multicore Memory Hierarchy

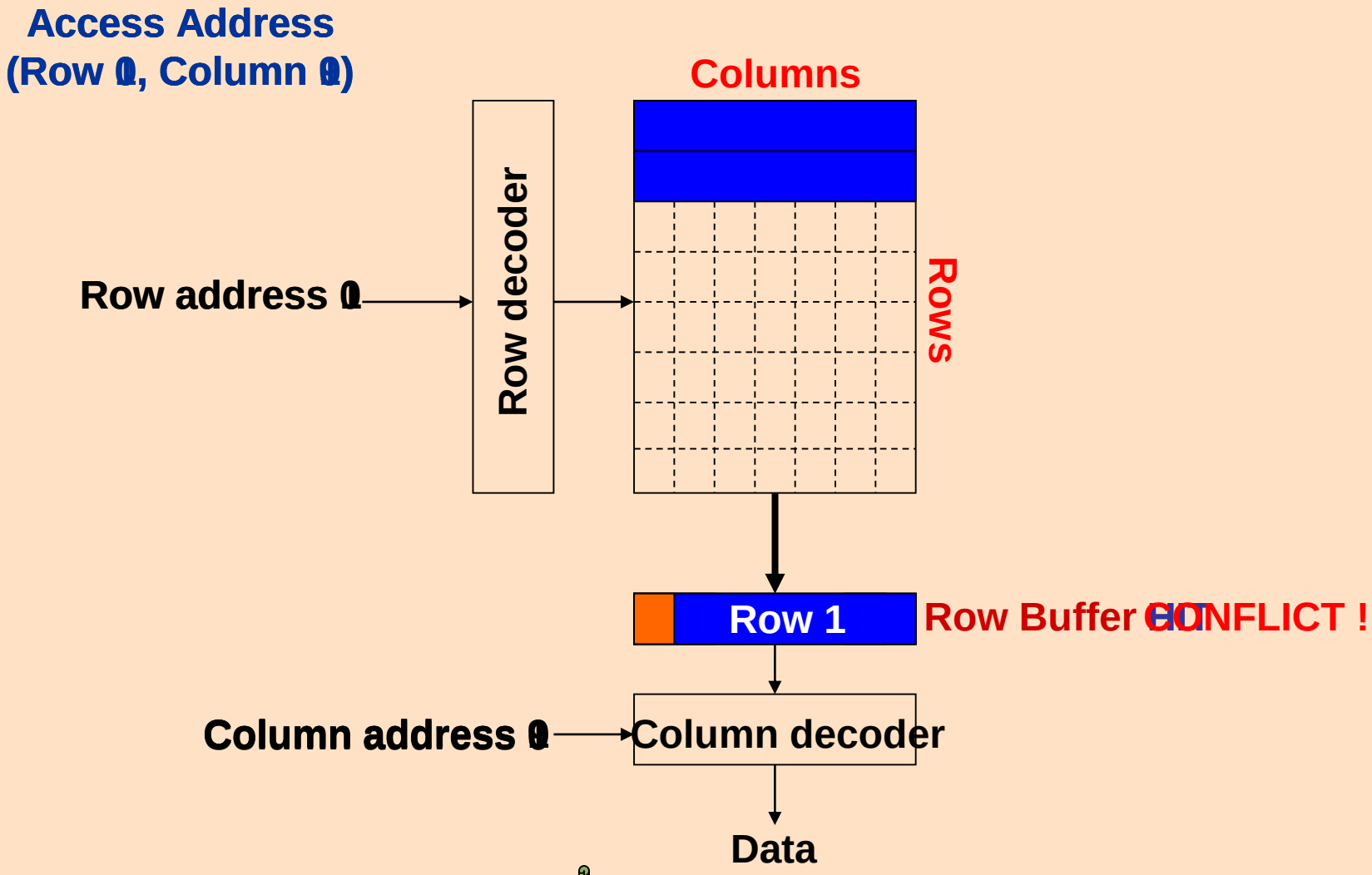


Overview of a DRAM based memory



Data Read & Write operations

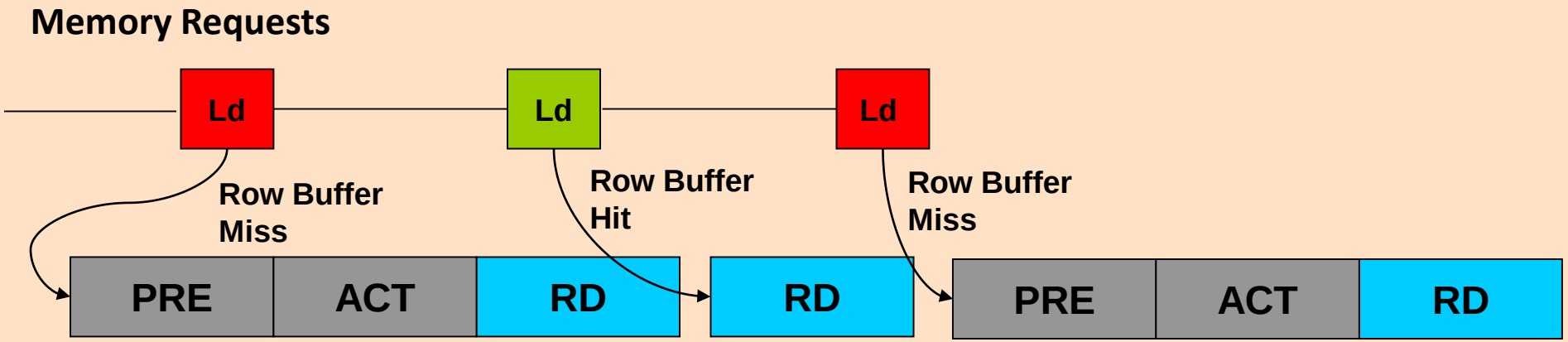
DRAM Bank Operation



Slide Source: Onur Mutlu, CMU

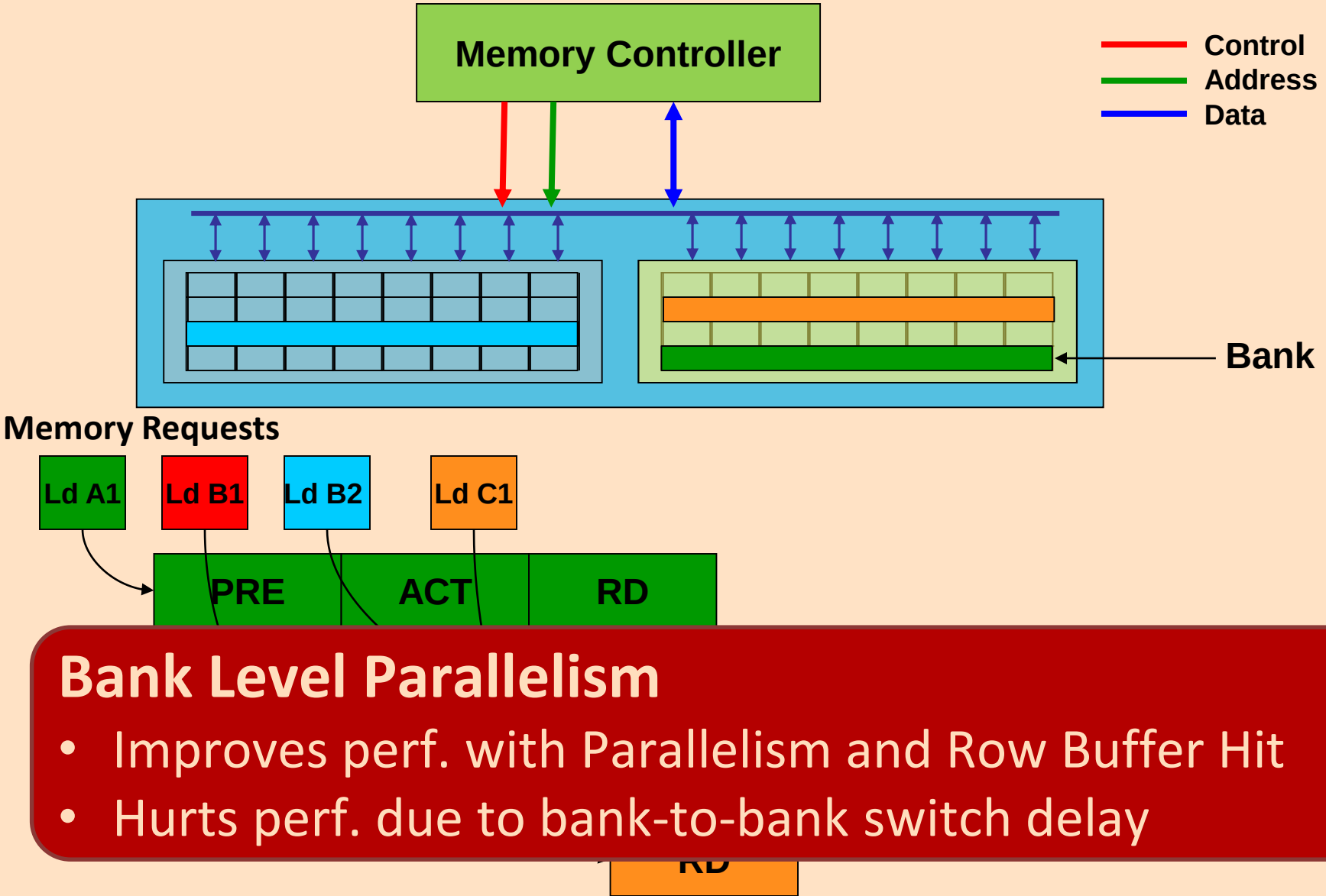
Basic DRAM Operations

- ACTIVATE → Bring data from DRAM core into the row-buffer
- READ/WRITE → Perform read/write operations on the contents in the row-buffer
- PRECHARGE → Store data back to DRAM core (ACTIVATE discharges capacitors), put cells back at neutral voltage



Row buffer hits are faster and consume less power

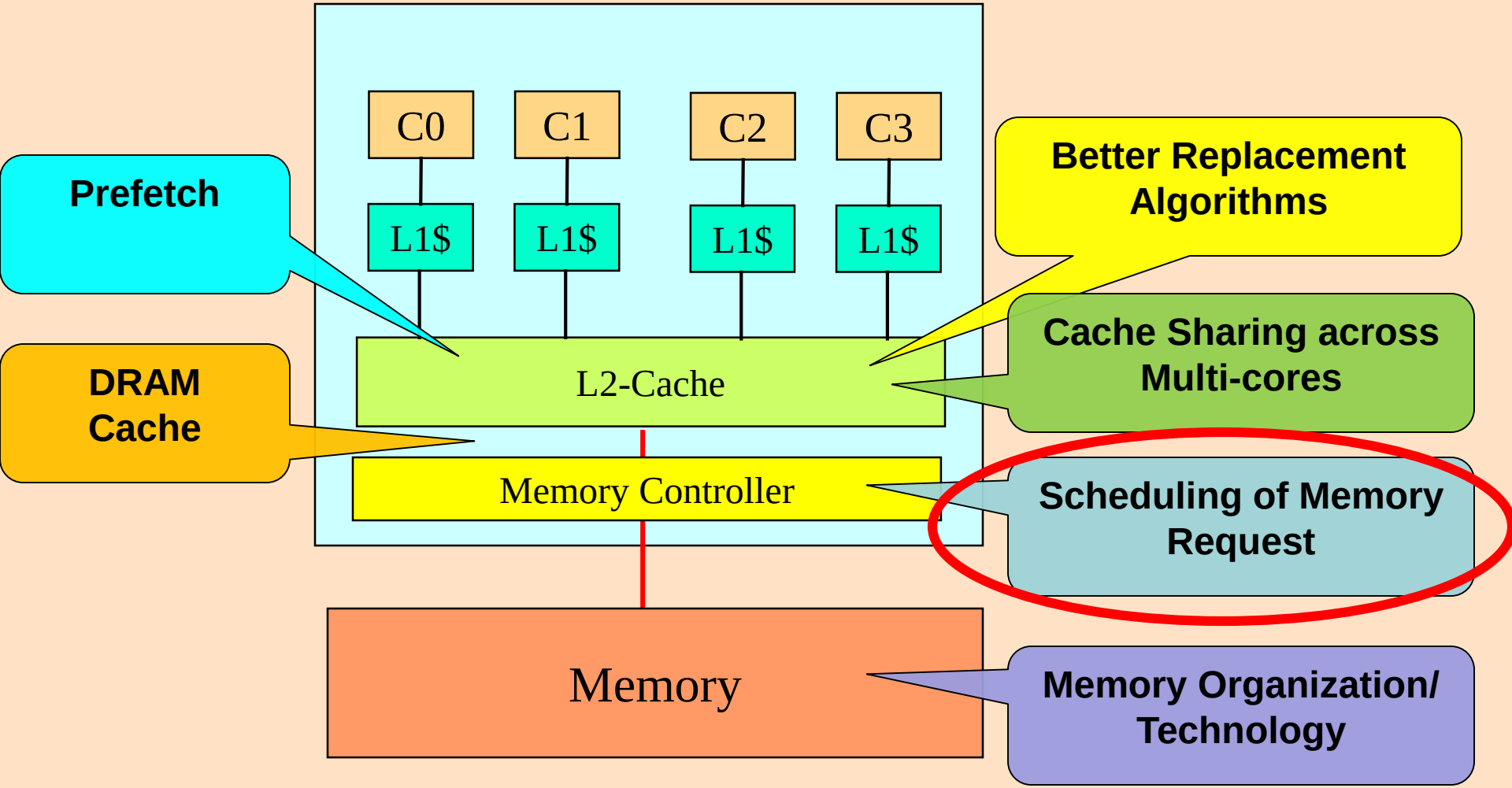
Bank Level Parallelism in DRAM



Bank Level Parallelism

- Improves perf. with Parallelism and Row Buffer Hit
- Hurts perf. due to bank-to-bank switch delay

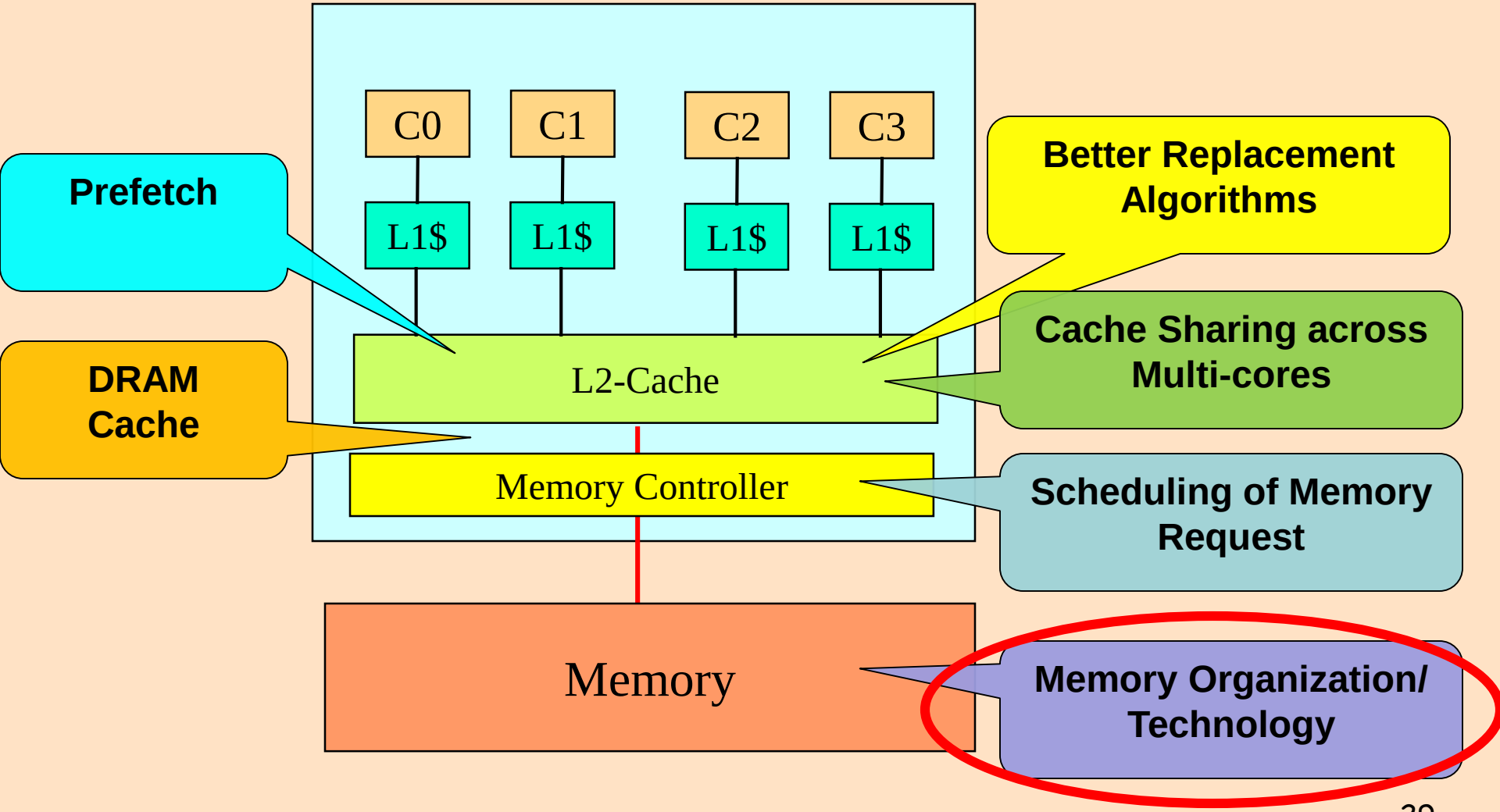
Research Issues in Multicore Memory Hierarchy



Memory Access Scheduling

- A row-conflict memory access takes significantly longer than a row-hit access
- Scheduling policy (FR-FCFS) [Rixner, ISCA'00] to improvem DRAM throughput
  - (1) Row-hit (column) first: Service row-hit memory accesses first
  - (2) Oldest-first: Then service older accesses first
- Multiple Small Row Buffers Organization [ICS2012]
  - A few (< 4) buffers per bank improve temporal locality
  - Small buffers (512-1024B) capture the Spatial locality

Research Issues in Multicore Memory Hierarchy



Memory System Design

Why is it complex?

- **Design space is huge! Simulation-based evaluation for the entire design-space is time consuming!**

- **Analytical Model for Memory System Performance**
  - Enables Rapid evaluation of alternatives
  - Non-trivial insights compared to simulation

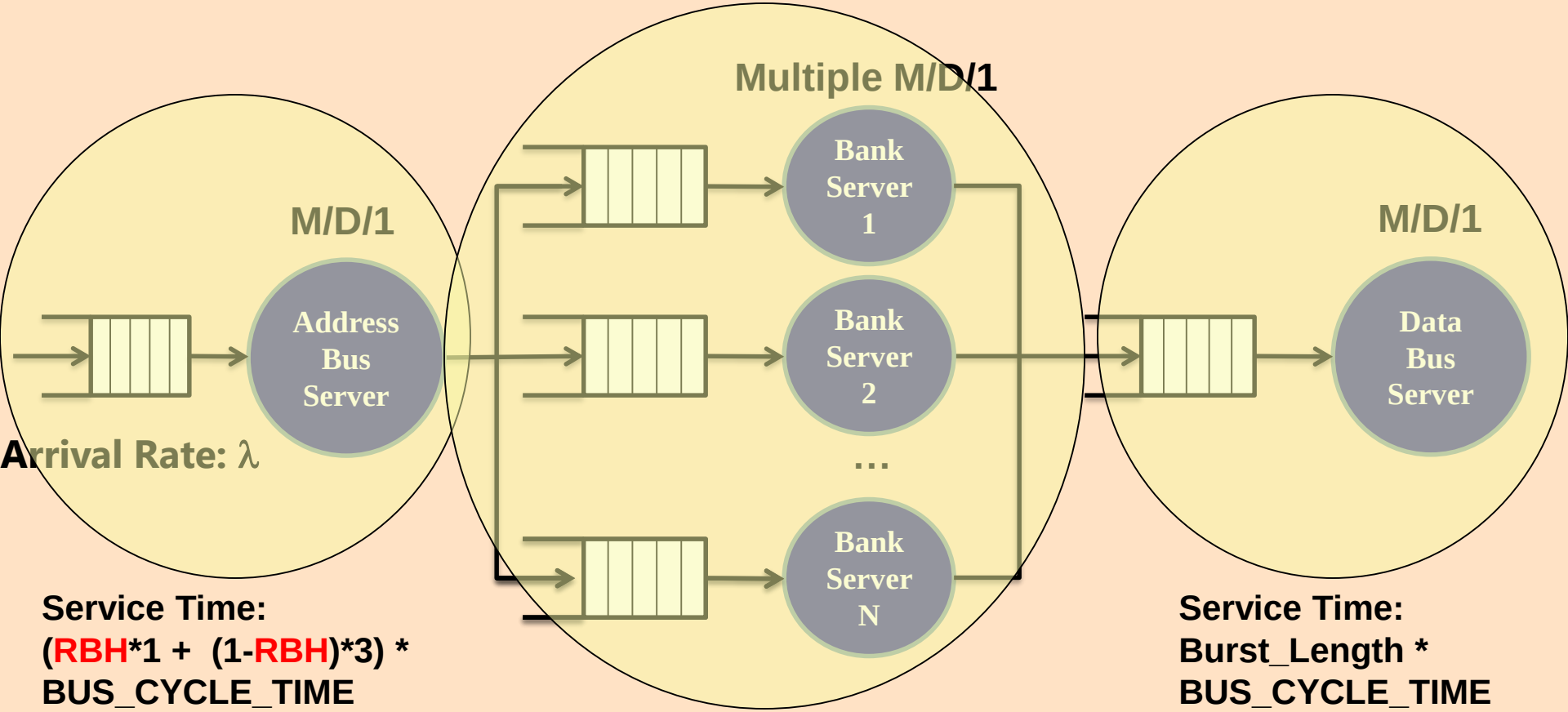
ANATOMY – Analytical Model of Memory (SIGMETRICS 2014)

Two components

- Queuing Model of Memory
  - Captures Organizational and Technological characteristics
  - Protocols like DDR3, DDR4/Wide-IO, PCM, …
  - Workload characteristics used as input
    - Computed by the other component
- Summarize Workload Characteristics
  - Captures Locality and Parallelism exhibited by workload's memory accesses

Analytical Model for Memory System Performance

A System with single Memory Controller



Service Time:
(RBH*1 + (1-RBH)*3) * BUS_CYCLE_TIME

Service Time:
t_{CL}* RBH + (t_{CL}+t_{PRE}+t_{RCD}) * (1-RBH)

Service Time:
Burst_Length * BUS_CYCLE_TIME

Summarizing Workload Characteristics

- All characteristics impact performance to varying degree
- How to estimate these characteristics **from a single trace** for each workload?

Estimating RBH

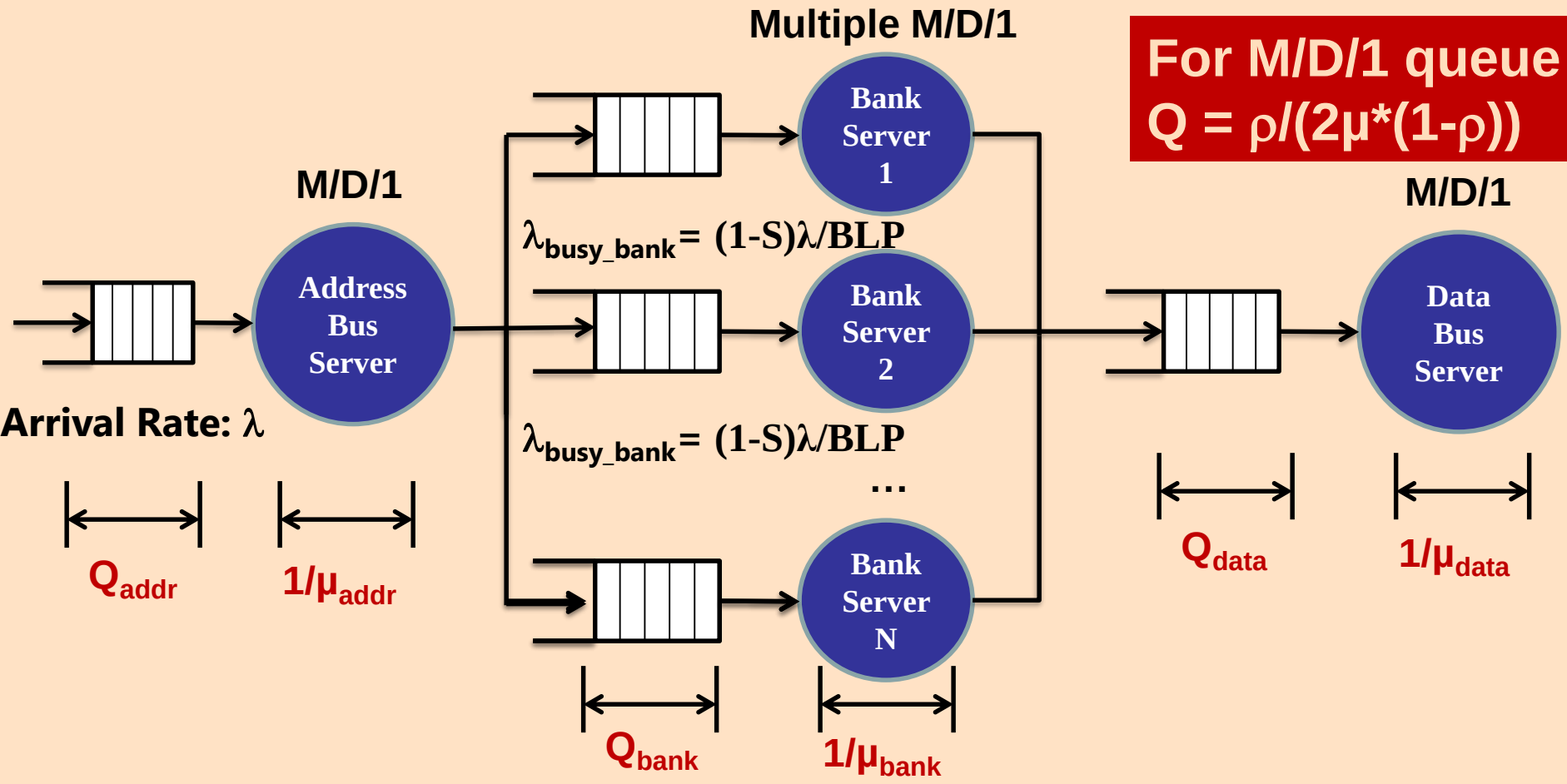
- Summarize locality in accesses
  - Reuse Distance Histogram (obtained from a single trace)
  - One Per Row-Buffer size
- Row Buffer Hit (RBH) rate estimation using combinatorial evaluation
  - When will Reuse Distance of ‘K’ translate into a Row-Buffer Hit?
  - Only if the intervening ‘K’ pages are to a subset of the remaining ‘N-1’ banks

$$RBH(N) = \sum_{K=0}^{K=\infty} RD_K \times \left(\frac{N-1}{N} \right)^K$$

Estimate BLP

- Parallelism (BLP) in accesses depends on
  - The time that ONE request occupies a bank
  - Number of new requests in that time
  - Their distribution
- BLP depends on how the requests spread S across idle vs. active banks
- Combinatorial answer leads to BLP estimation.

Putting It Together



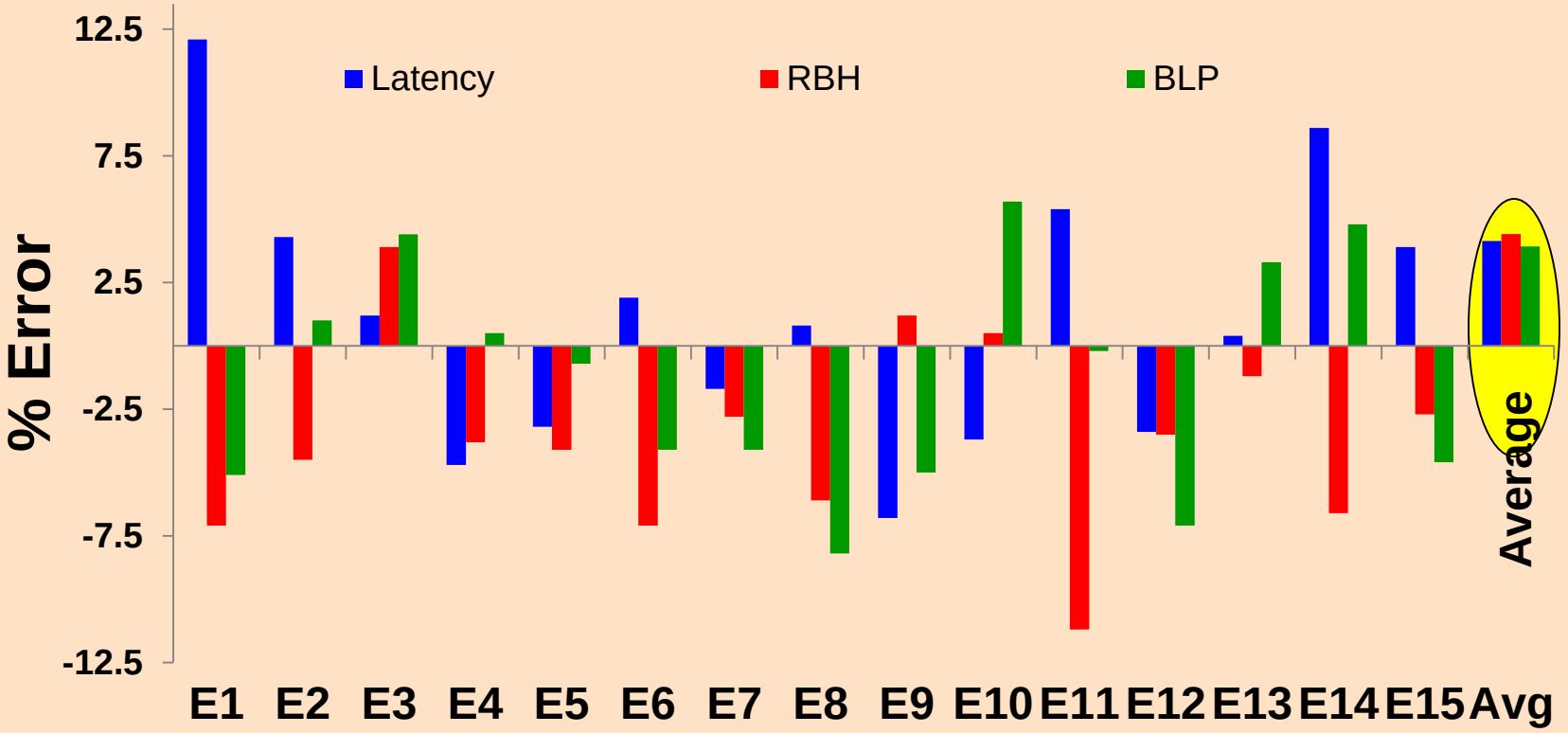
$$\text{Latency} = Q_{addr} + Q_{bank} + Q_{data} + 1/\mu_{addr} + 1/\mu_{bank} + 1/\mu_{data}$$

$$\text{Peak\_BW} = \text{Min}(\mu_{addr}, \mu_{bank} * N, \mu_{data})$$

Extensions to the Model

- Multiple memory controllers
- Different memory scheduling algorithms
- Refresh in DRAM
- Different memory technology (e.g., PCM)
- Closed network model

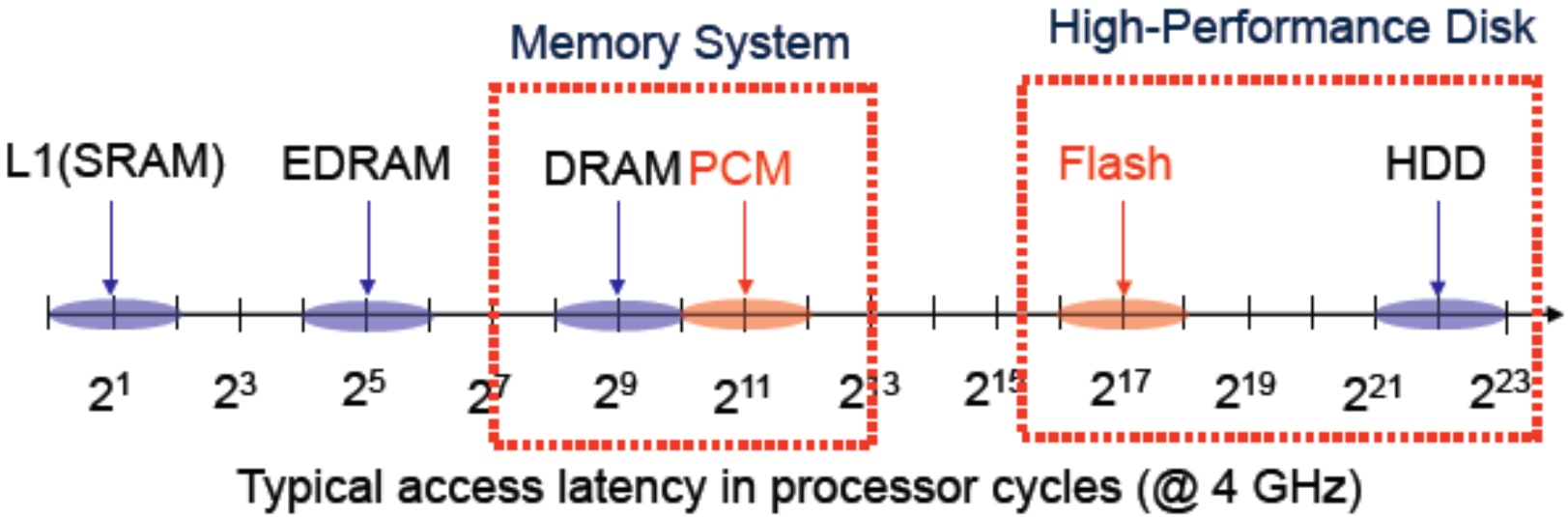
Validation - Model Accuracy



- Validation using GEM5 Simulation (with detailed Memory simulation) on Multiprogrammed workload
- Low Errors in RBH, BLP and Latency Estimation
- Average error of 3.9%, 4.2% and 4%

Emerging Memory Technology

- Non-Volatile Memory technology
  - Phase Change Memory (PCM), Magnetic RAM (MRAM), Resistive RAM (RRAM), Spin Torque Transfer RAM (STT-RAM), …



Slide Source: Moin Quereshi, Georgia Tech.

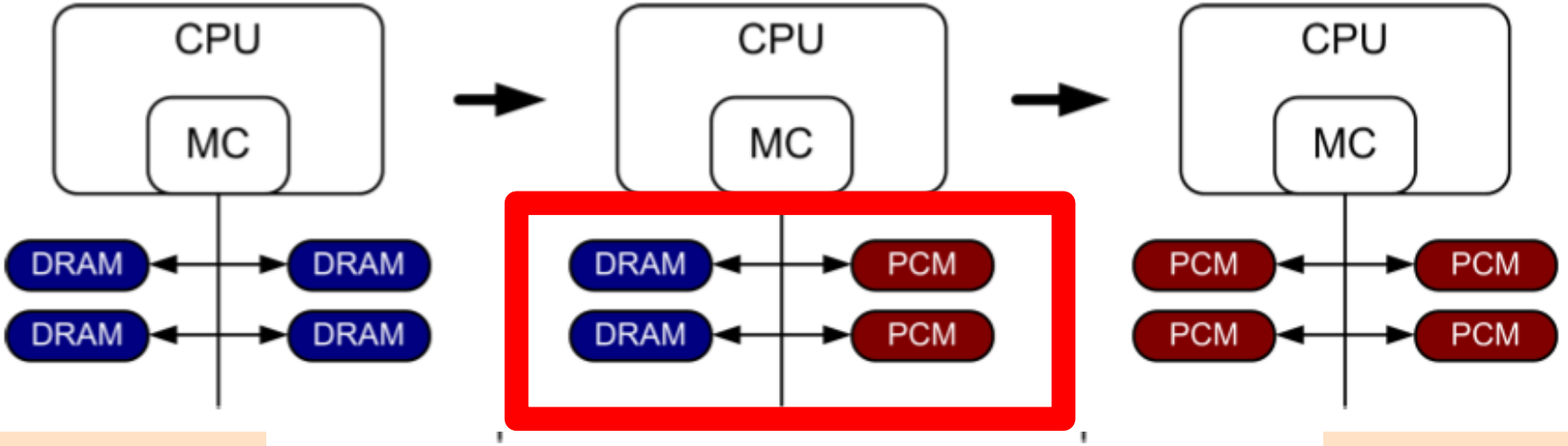
Emerging Memory Technology

- Phase Change Memory
  - Data stored by changing phase of special material
  - Data read by detecting material's resistance
  - Phase change material (chalcogenide glass) exists in two states:
    1. Amorphous: high resistivity – reset state or 0
    2. Crystalline: low resistivity – set state or 1
  - Non-volatality and low idle power (no refresh)
  - Expected to scale (to 9nm), denser than DRAM, and can store multiple bits/cell
  - Higher Write latency and write-energy
  - Endurance issues (cell dies after 10^8 writes)

Slide Source: Onur Mutlu, CMU

DRAM – PCM Hybrid Memory

- PCM-based (main) memory be organized?

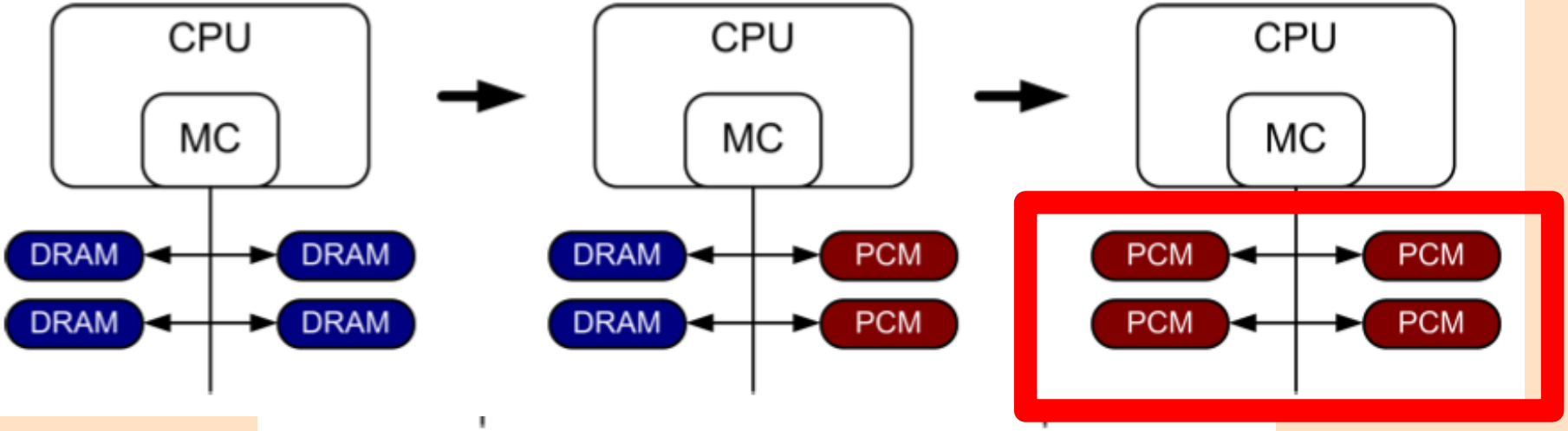


- Hybrid PCM+DRAM
  - How to partition/migrate data between PCM and DRAM
  - Is DRAM a cache for PCM or part of main memory?
  - How to design the hardware and software

Slide Source: Onur Mutlu, CMU

PCM-based Main Memory

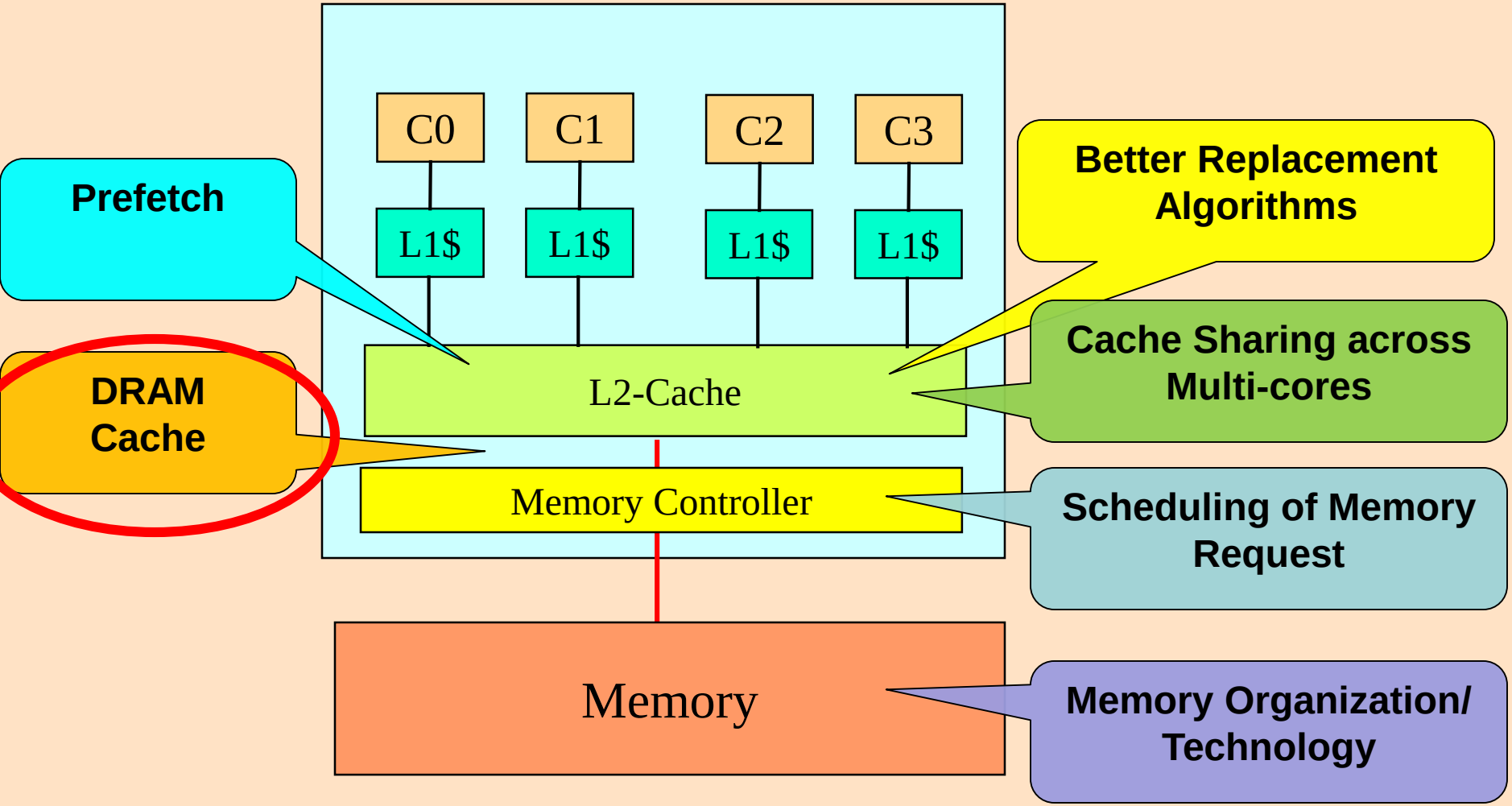
- How should PCM-based (main) memory be organized?



- Pure PCM main memory [Lee et al., ISCA'09, Top Picks'10]:
  - How to redesign entire hierarchy (and cores) to overcome PCM shortcomings

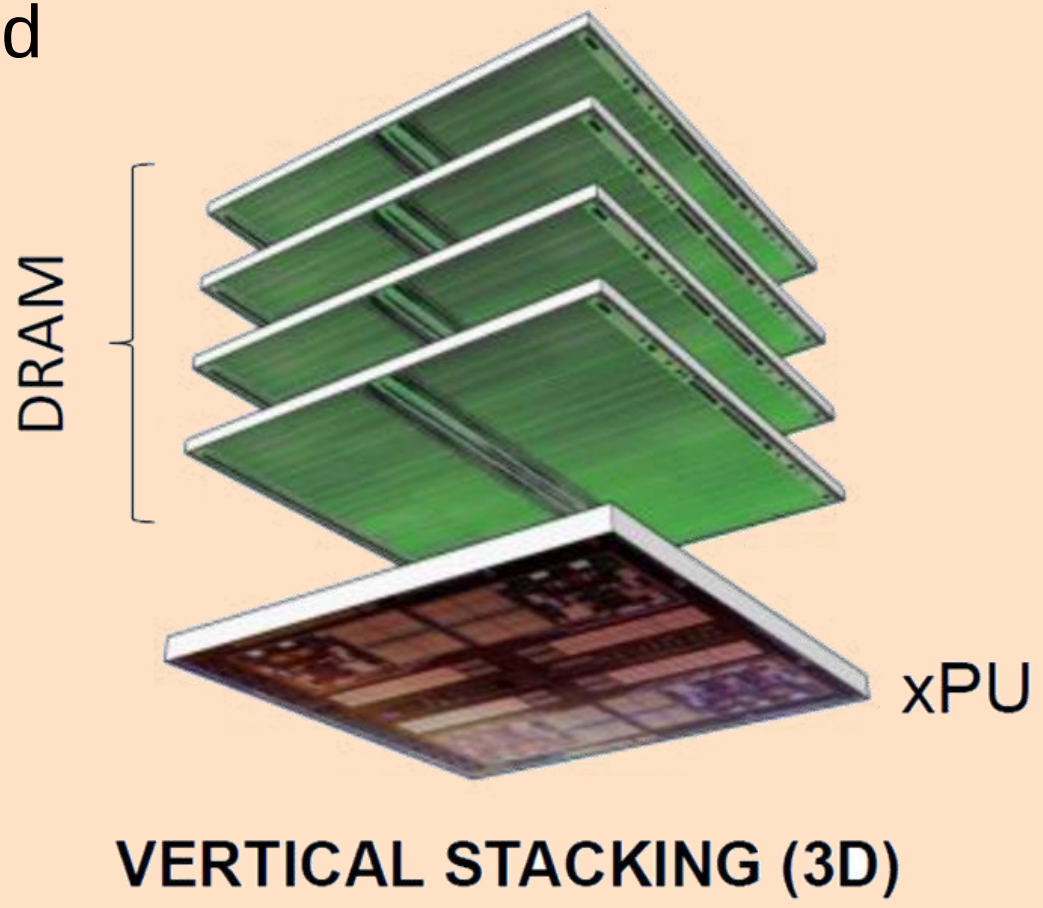
Slide Source: Onur Mutlu, CMU

Research Issues in Multicore Memory Hierarchy



Stacked DRAM

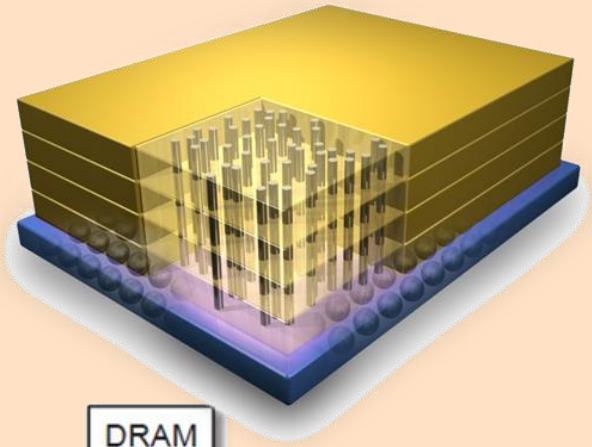
- DRAM vertically stacked over the processor die.
- Stacked DRAMs offer
  - High bandwidth
  - High capacity
  - Moderately low latency.
- Several proposals to organize this large DRAM as a last-level cache.

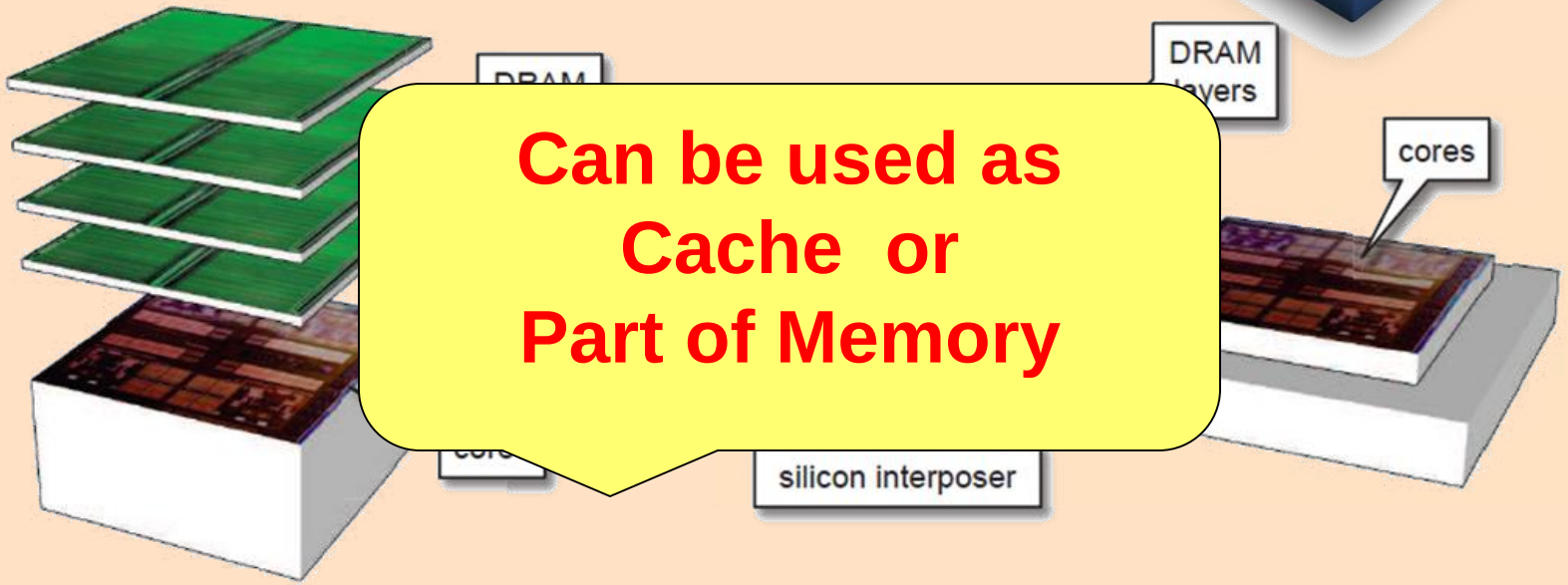


Picture courtesy Bryan Black (From MICRO 2013 Keynote)

Stacked DRAM

- DRAM vertically stacked on the processor die.
- Stacked DRAMs offer
  - High bandwidth
  - Large capacity
  - Same or slightly lower latency.



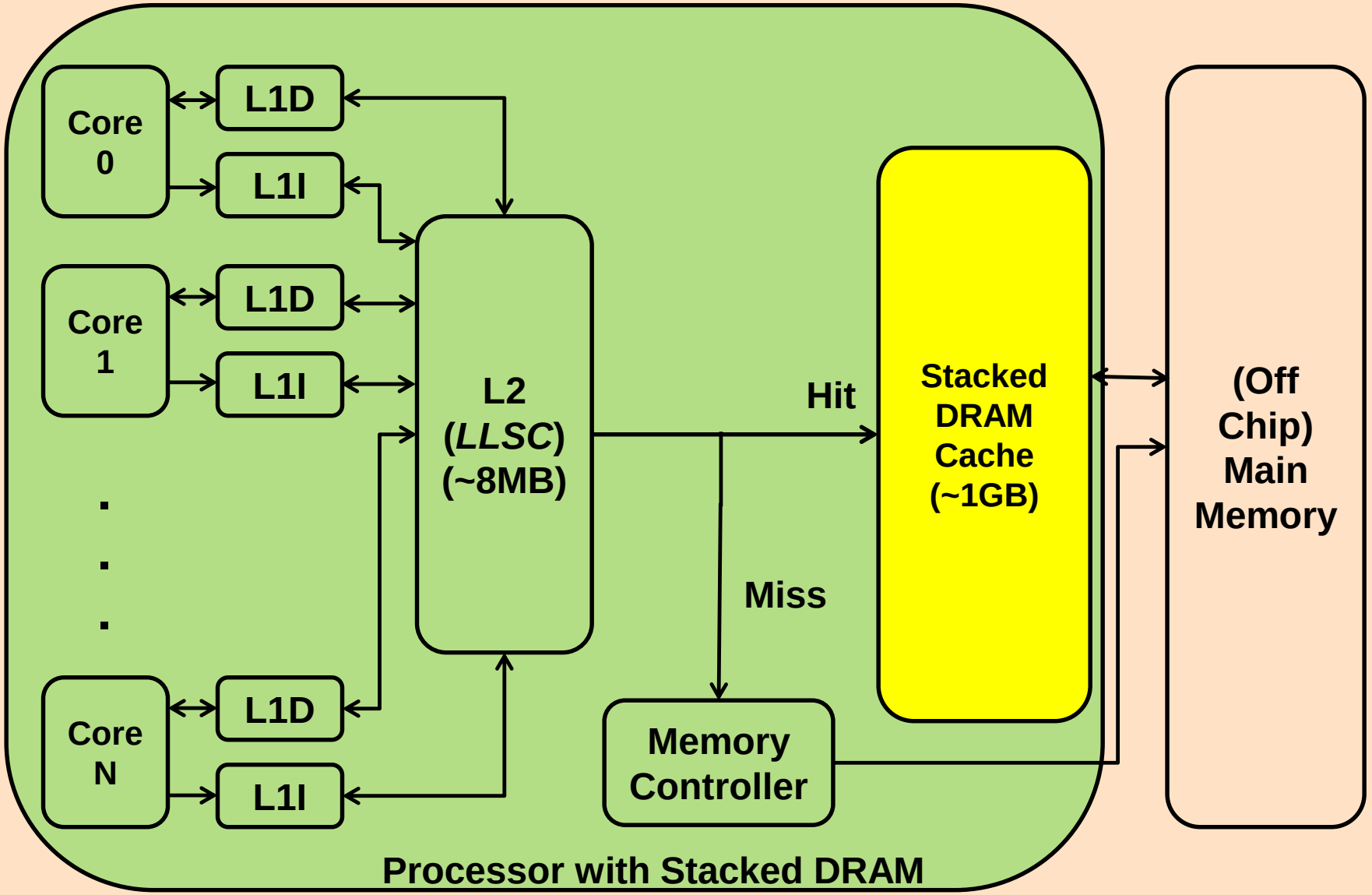


Can be used as Cache or Part of Memory

3-D Stacked DRAM

2.5-D Stacked DRAM

Processor Orgn. With DRAM Cache



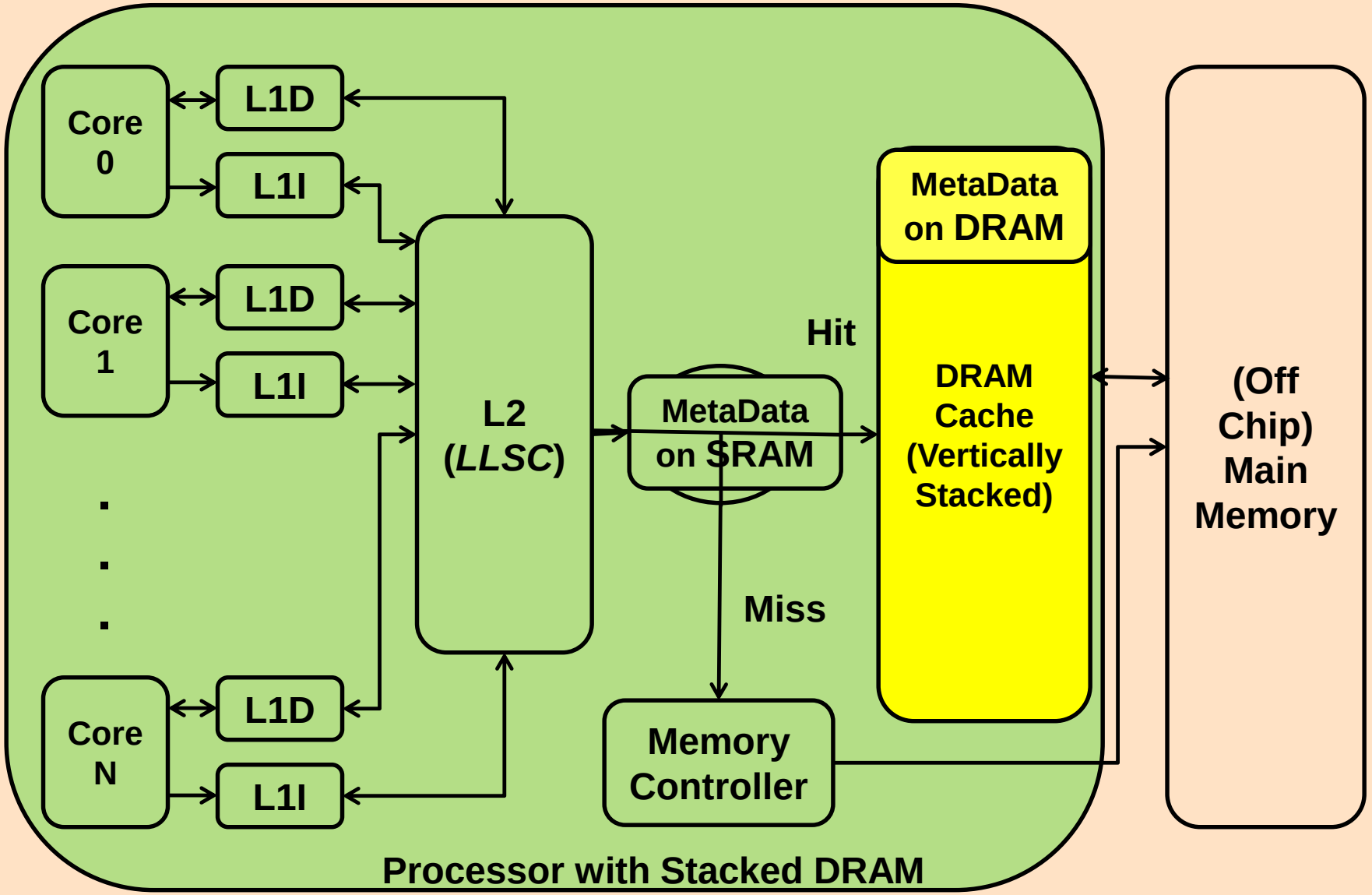
Problems in Architecting Large Caches

- Small cache line size (64 B): Lower spatial locality, but reduced wasted bandwidth and cache capacity
- **Problem:** Cache of hundreds of MB needs tag-store of tens of MB

Can Hit Time and Hit Rate
be improved simultaneously
while decreasing wasted
off-chip bandwidth and cache capacity ?

- **Problem:** wasted off-chip bandwidth and wasted cache capacity

Processor Orgn. With DRAM Cache



Bi-Modal Cache (Micro-2014)

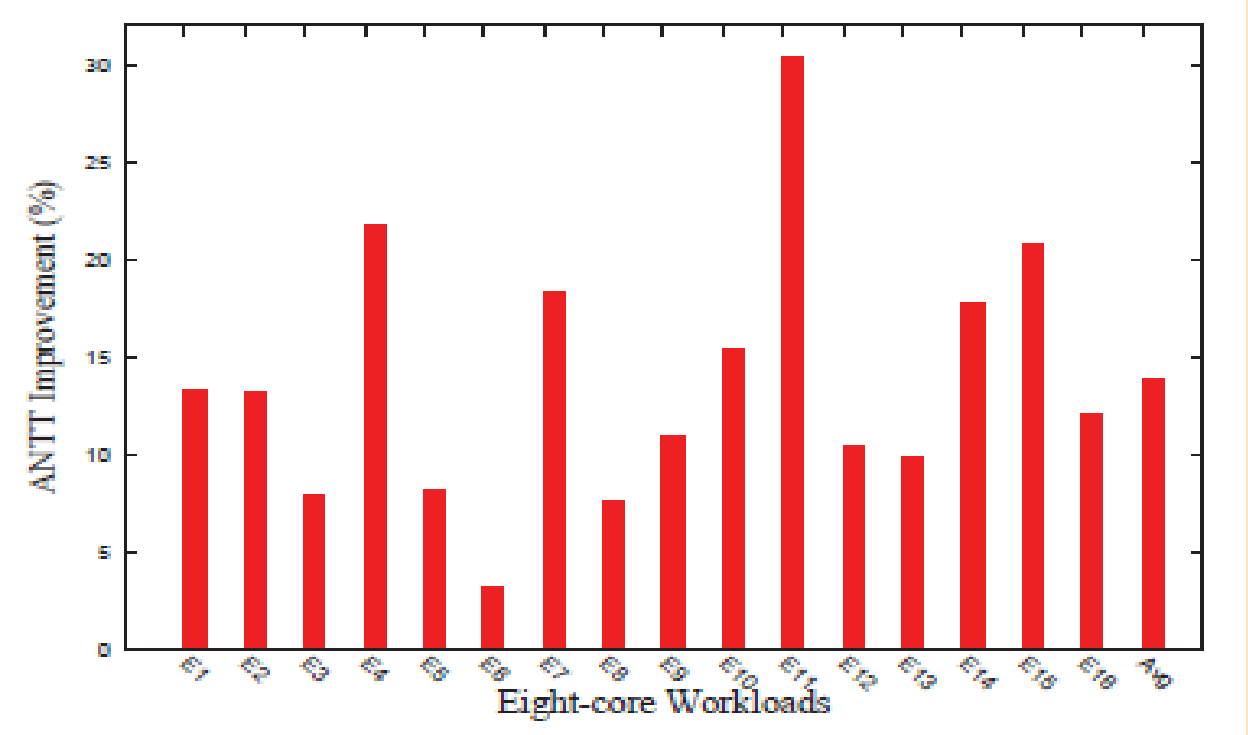
- Tags-In-DRAM organization

- With 3 new organizational features:
  1) Cache Sets are *Bi-Modal* – they can hold a combination of big (512B) and small (64B) blocks → **Improves Hit Rate And Reduces Off-Chip Bandwidth**
  2) Parallel Tag and Data Accesses
  3) Eliminating Most Tag Accesses via a small SRAM based *Way Locator*

  → **Reduce Hit Latency**

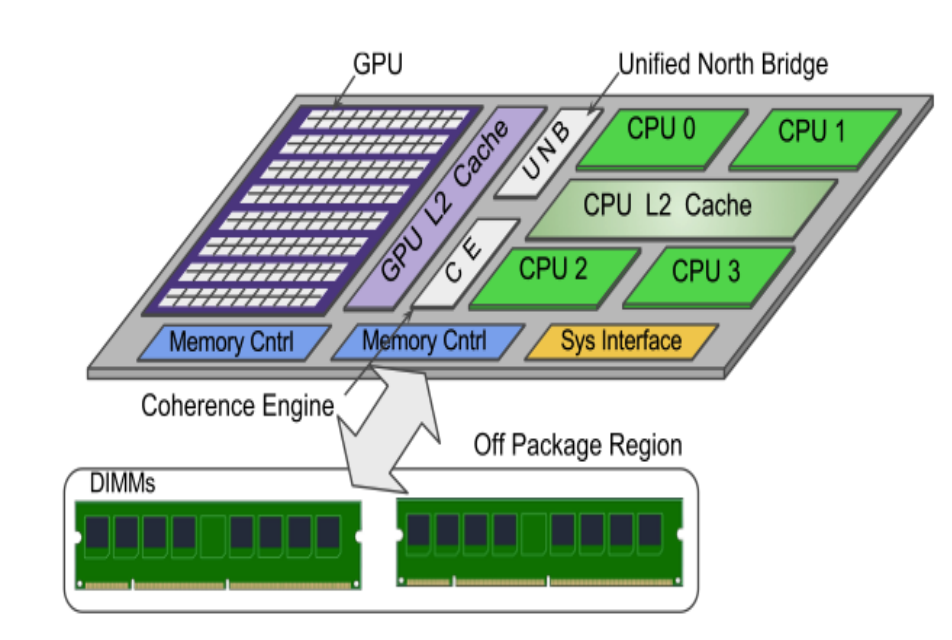
Results - Performance

- Performance improvement of 10.8%, 13.8% and 14% in 4, 8 and 16-core respectively over an aggressive baseline



Integrated Heterogeneous Systems (IHS) Architecture

- Latency-oriented CPU cores + Throughput-oriented GPGPU SMs on-chip
  - Simplifies Programming - Shared Virtual Memory, pointer sharing
  - Allows GPUs to operate on data sets larger than memory size
  - Share resources - NoC, caches, memory controllers, DRAMs
  - e.g. AMD APUs, Intel Iris, NVIDIA Denver



Integrated Heterogeneous Systems (IHS) Architecture

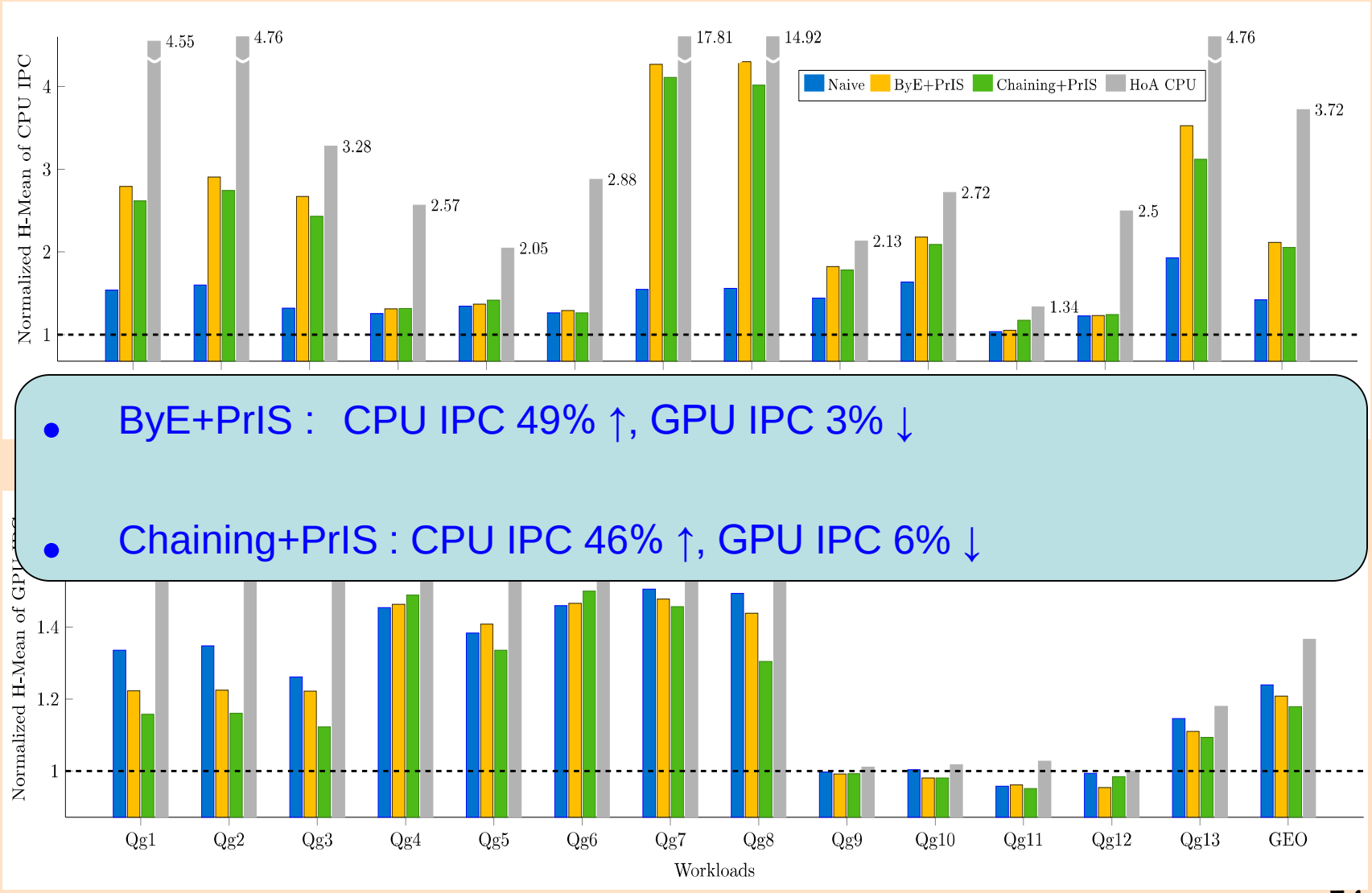
- Integrated Heterogeneous System (IHS) Architecture with CPU and GPU cores sharing certain level of memory hierarchy
- Have disparate memory access pattern and requirements!
  - GPU cores pump in large no. of requests, bandwidth hungry, but are latency-tolerant!
  - CPU cores require small foot-print, low demand rate, but latency sensitive!

➔ Shared resource management for effective use of CPU and GPU cores

HAShCache : Heterogeneity Aware Shared DRAM Cache

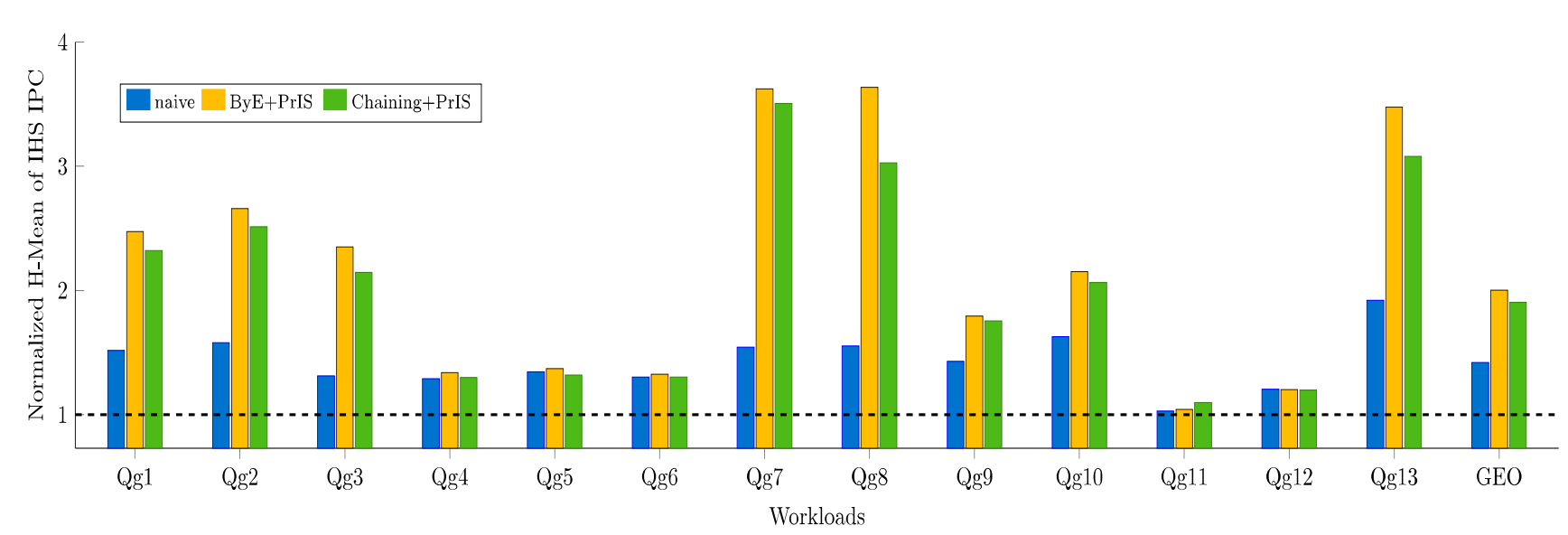
- An optimized DRAM cache for IHS processors
- Efficient DRAM cache design for heterogeneous architecture
  - Carefully architect the first order design constraints
  - Cache block size, metadata overheads, set associativity, miss penalty, addressing scheme
- Three Heterogeneity aware DRAM cache mechanisms
  - Heterogeneity aware DRAM cache scheduler - PrIS
  - Heterogeneity aware Temporal Bypass - ByE
  - Heterogeneity aware Spatial Occupancy Control – Chaining

HAShCache Performance



- ByE+PrIS : CPU IPC 49% ↑, GPU IPC 3% ↓
- Chaining+PrIS : CPU IPC 46% ↑, GPU IPC 6% ↓

HAShCache: IHS Performance



- ByE+PrIS : 107% improvement
- Chaining+PrIS : 101% improvement

Conclusions

- Memory hierarchy performance is important in multicore architectures
- Research issues/opportunities exist across the hierarchy
- Many (open source) simulators available for the experimentation

Acknowledgment

- All my Graduate Students, specifically
  - R. Manikantan
  - Kaushik Rajan
  - G.D. Nagendra
  - Adarsh Patil
- Funding Agencies
  - AMD Corporation
  - Intel Corporation
  - IBM Faculty Award
  - Nvidia Research Partnership

Thank You !!

July 2019