
WELCOME
TO
15-213
14-513
18-213
18-613
15-513

Future of Computing

15-213/18-213/14-513/15-513/18-613: Introduction to Computer Systems
28th Lecture, December 5, 2019

Today

- **Writeback-Aware Caching (guest lecture by Charles McGuffey)**
- Systems for Machine Learning (guest lecture by Angela Jiang)
- Prescriptive Memory

Writeback Aware Caching

Charles McGuffey

Nathan Beckmann, Phillip Gibbons

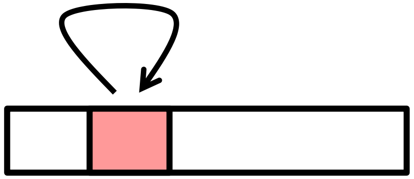
PARALLEL DATA LABORATORY

Carnegie Mellon University

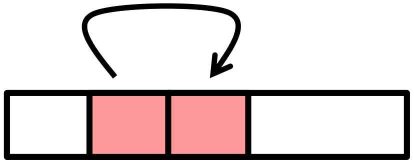
Recall: Locality

- **Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently**

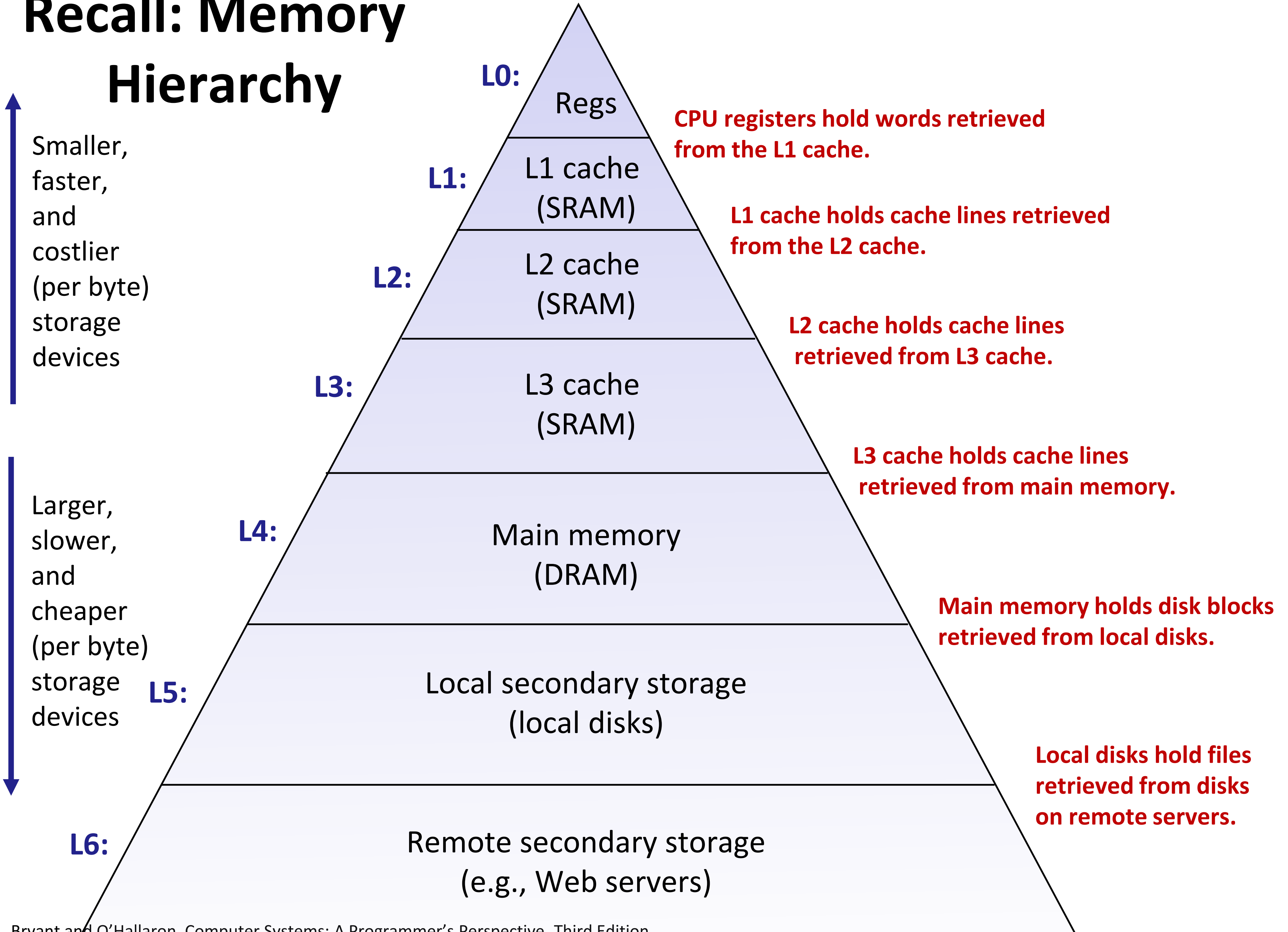
- **Temporal locality:**
  - Recently referenced items are likely to be referenced again in the near future



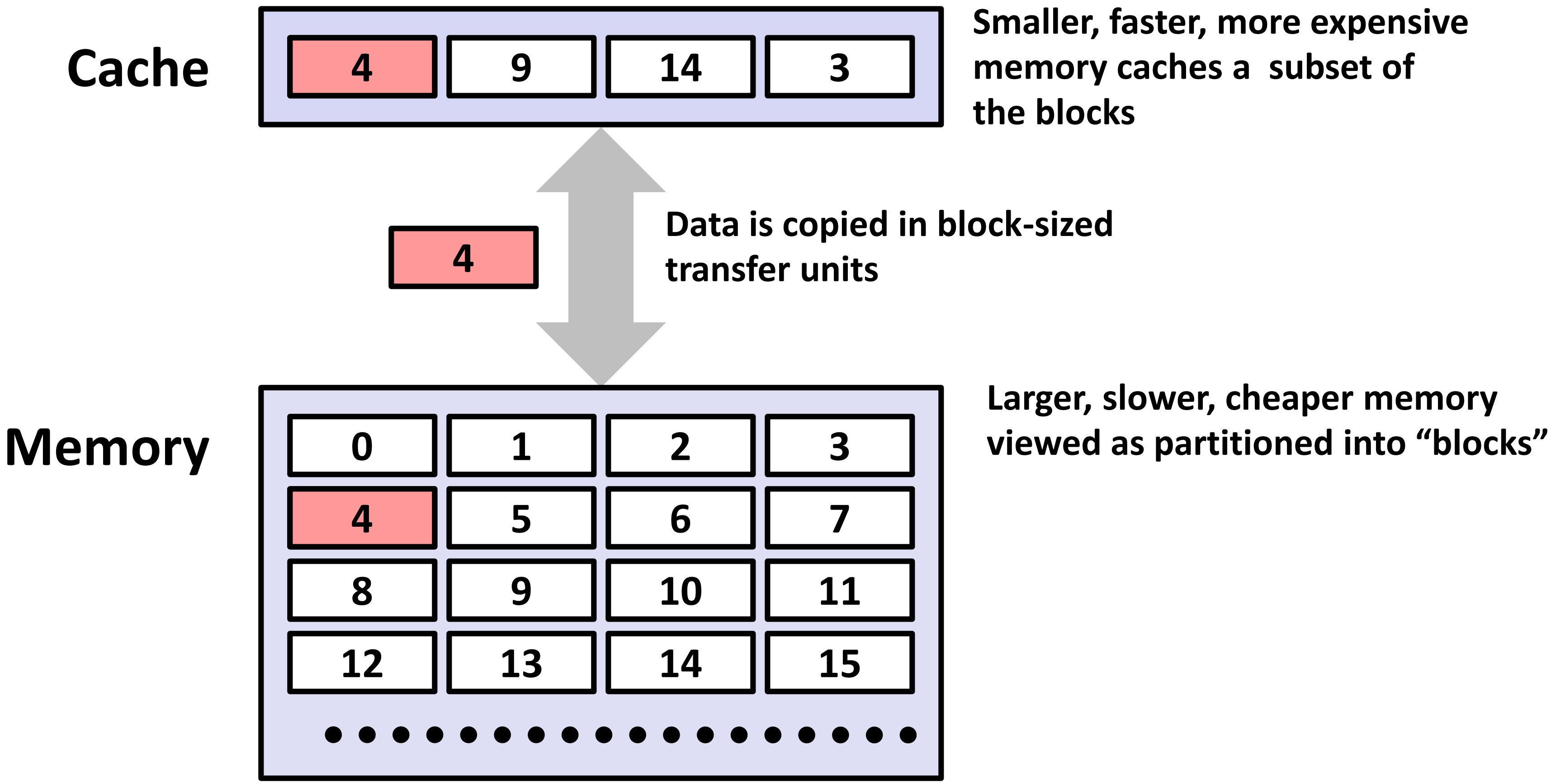
- **Spatial locality:**
  - Items with nearby addresses tend to be referenced close together in time



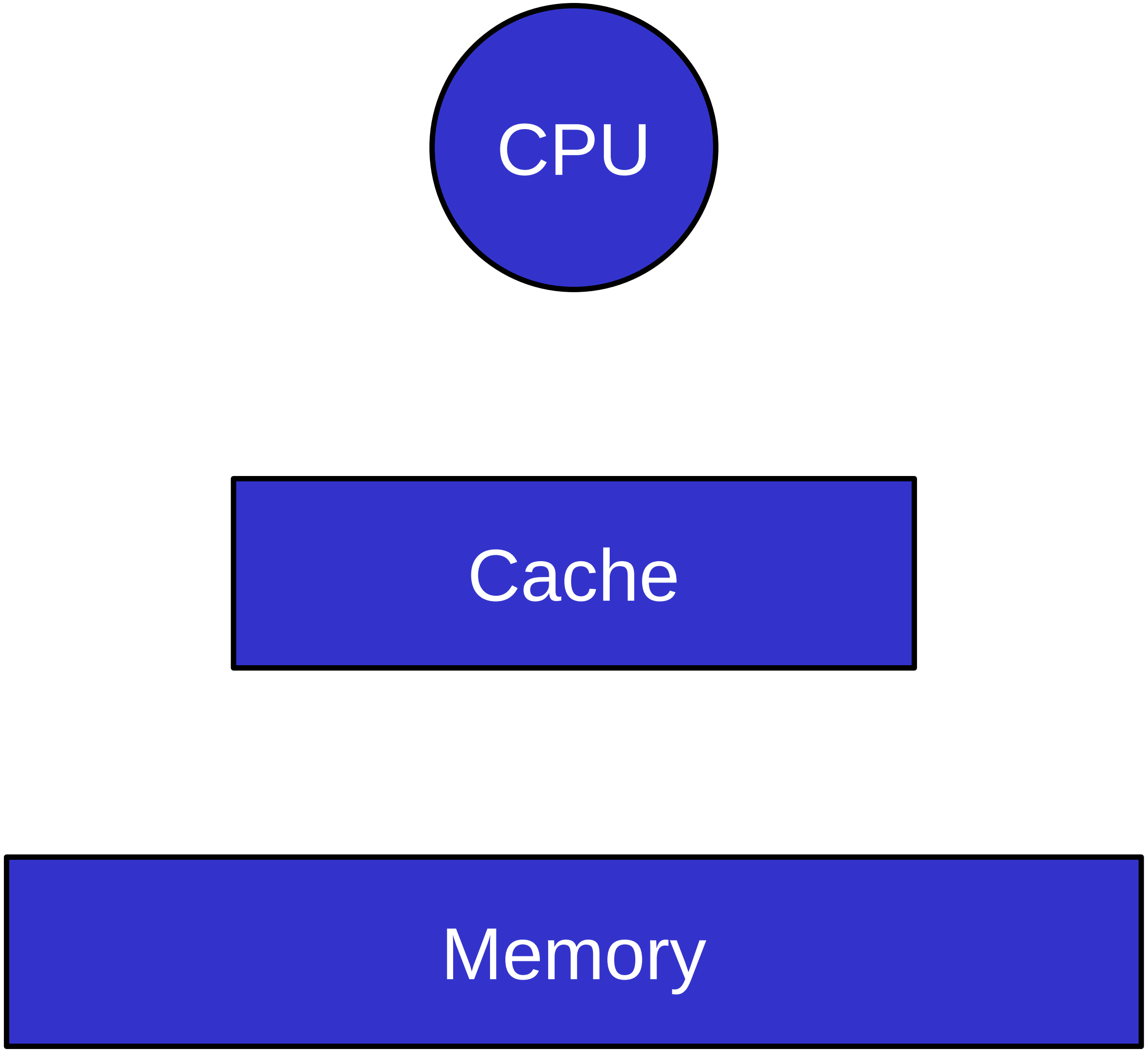
Recall: Memory Hierarchy



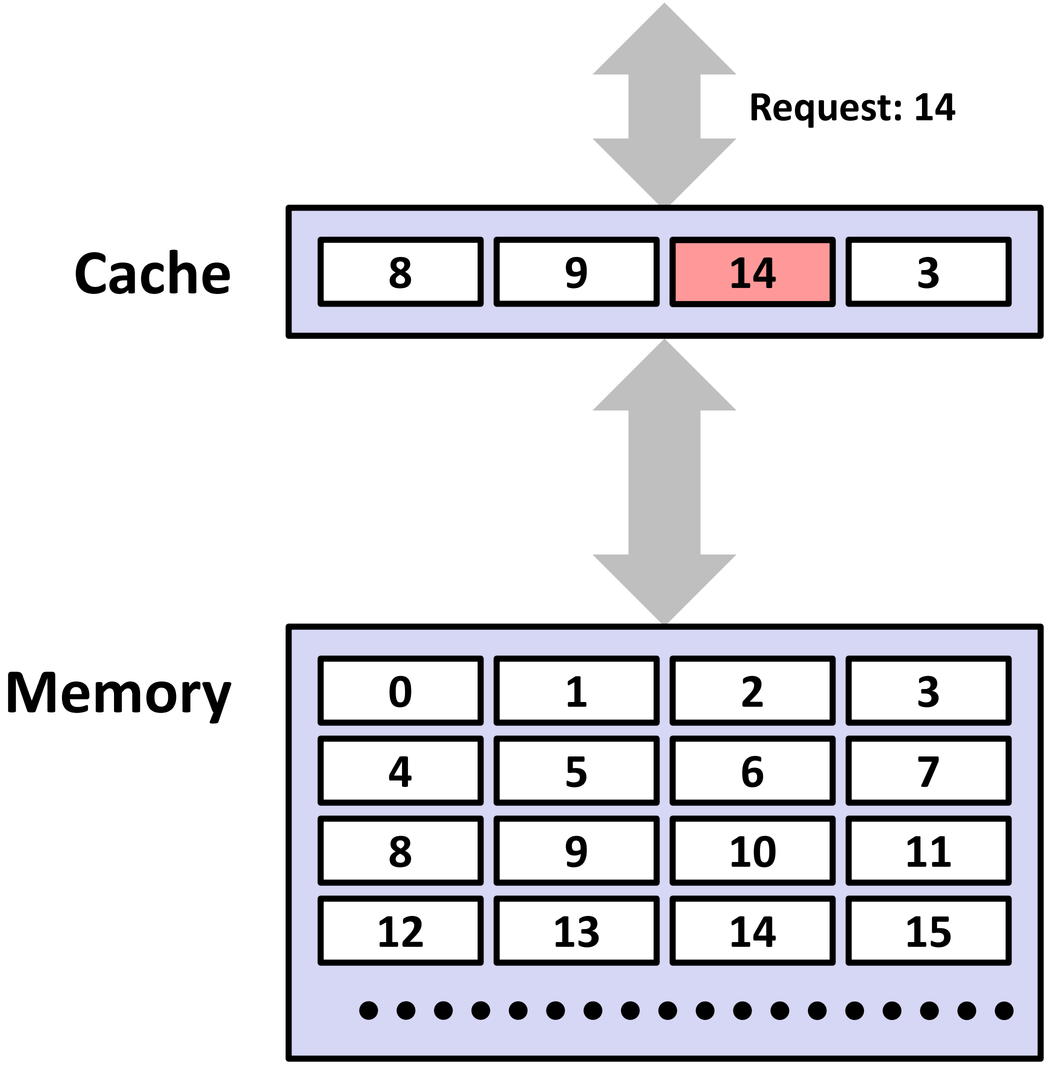
Recall: General Cache Concepts



Caching Model



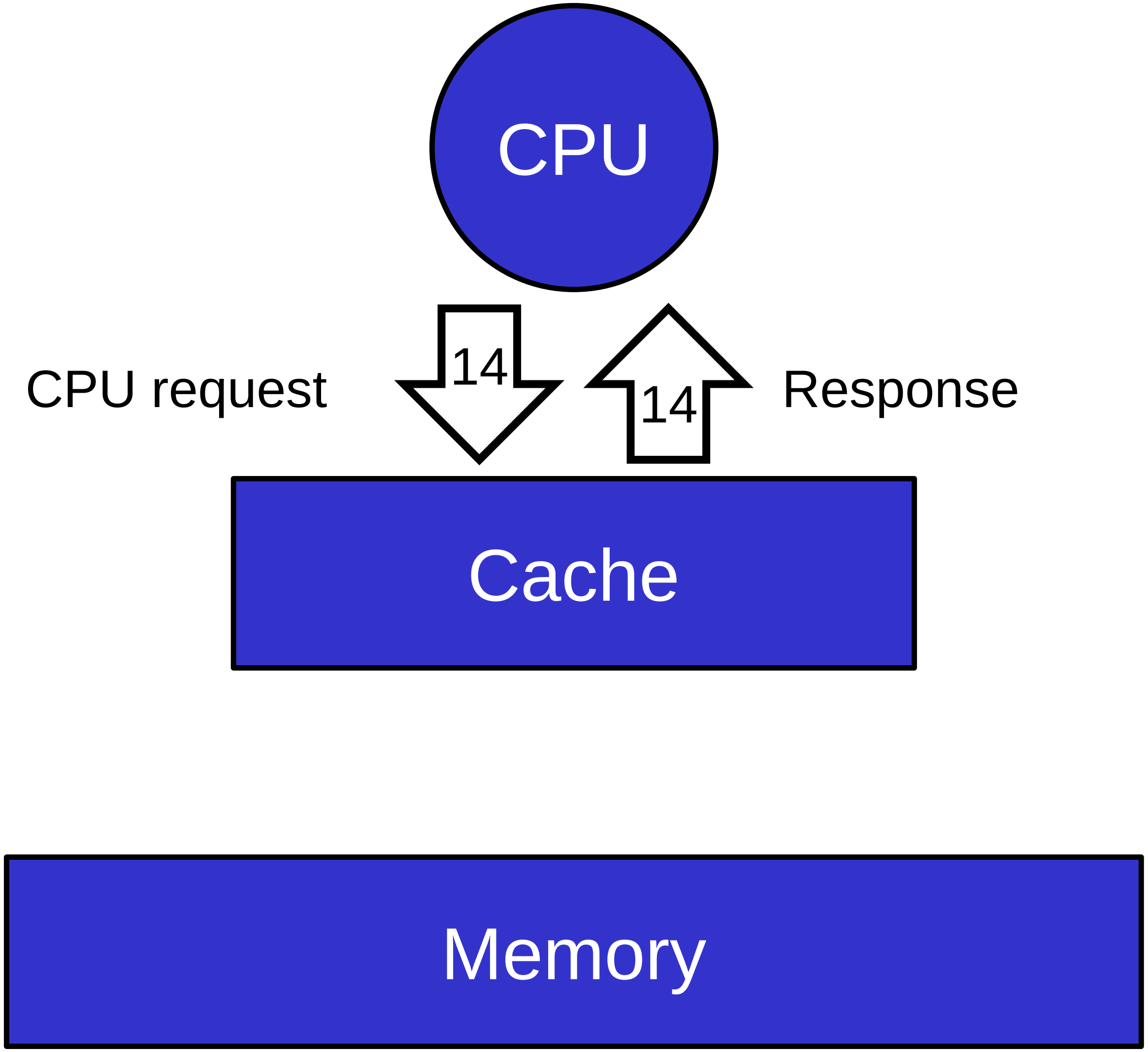
General Cache Concepts: Hit



Data in block b is needed

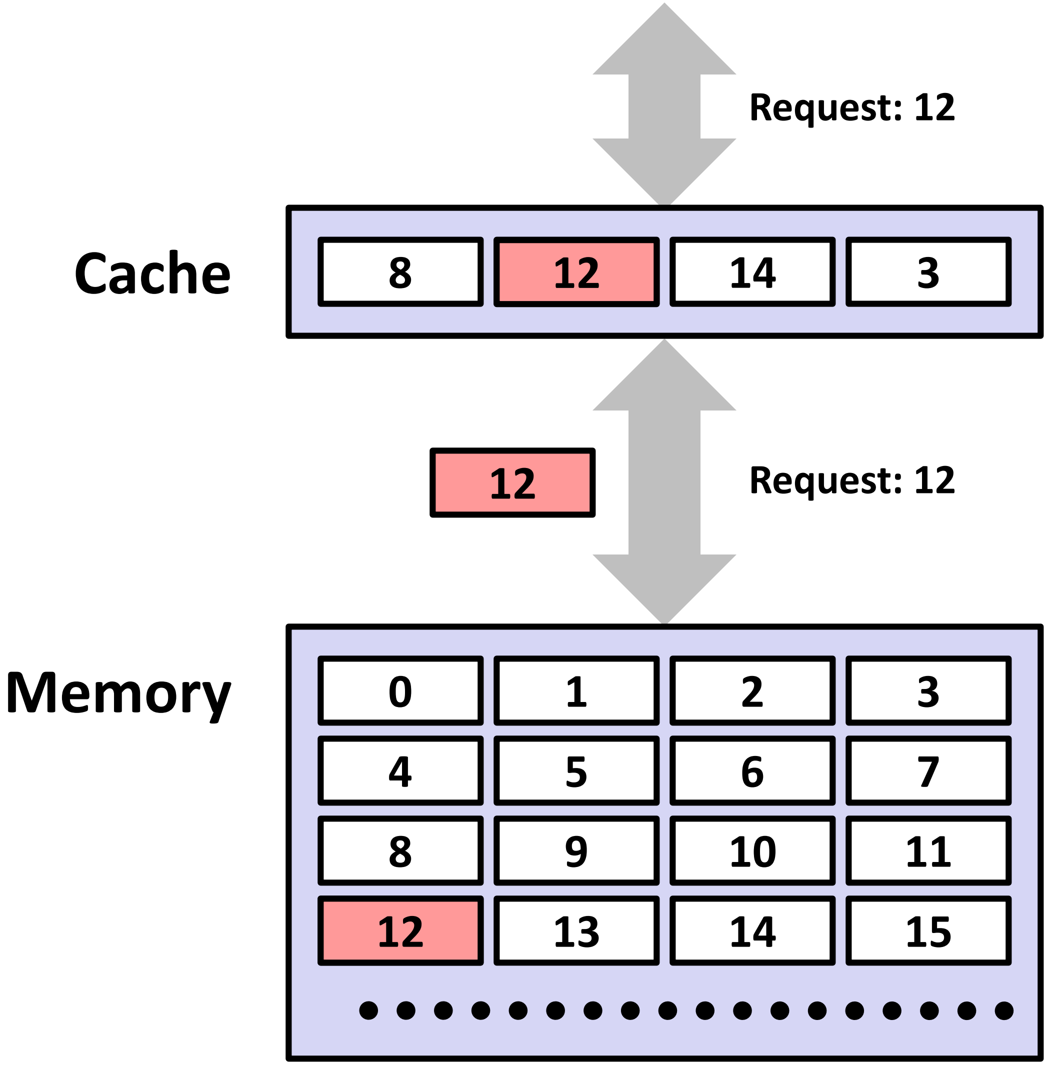
Block b is in cache:
Hit!

Modeling Hits



Carnegie Mellon
Parallel Data Laboratory

General Cache Concepts: Miss



Data in block b is needed

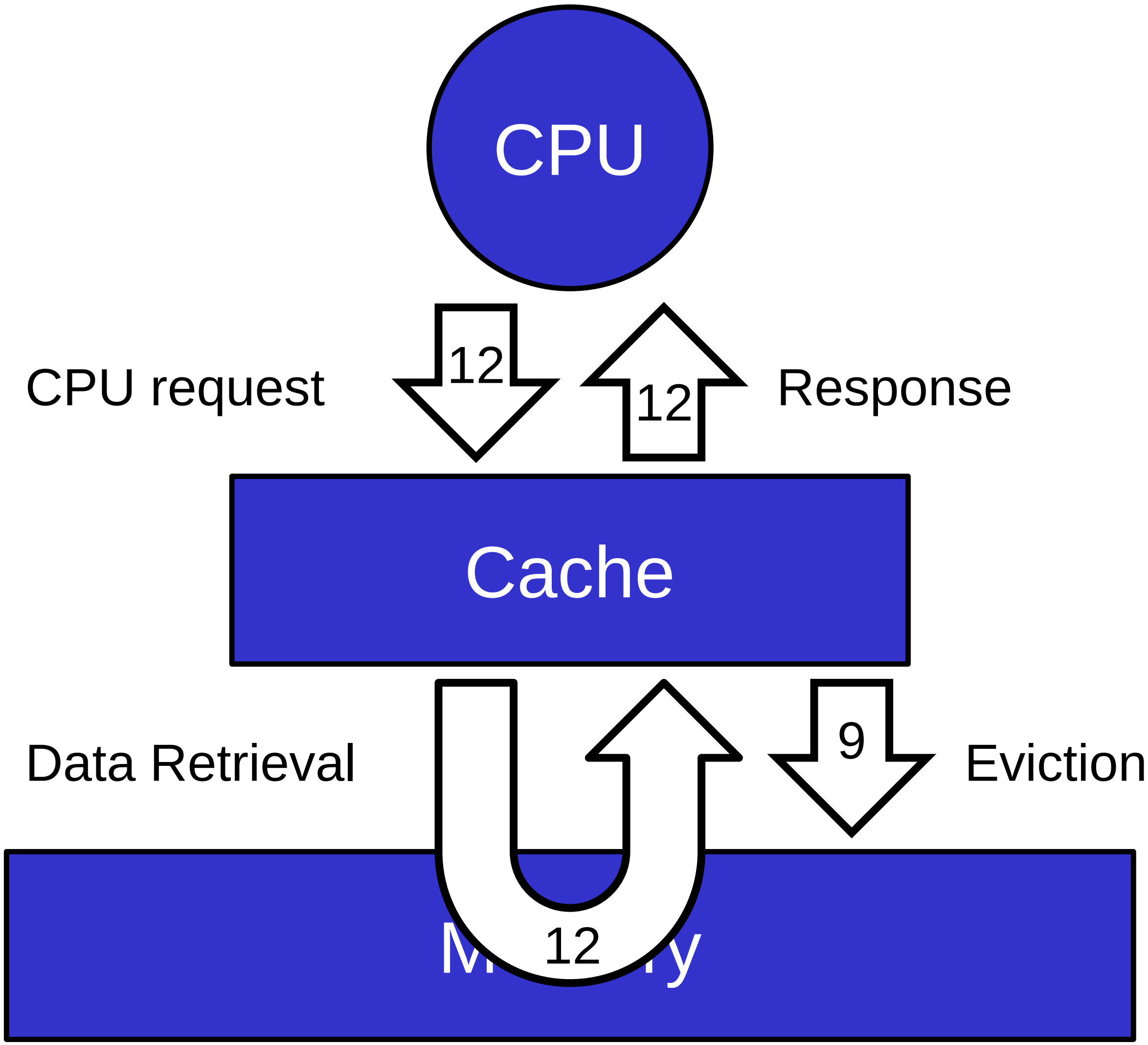
Block b is not in cache:
Miss!

Block b is fetched from memory

Block b is stored in cache

- Placement policy: determines where b goes
- Replacement policy: determines which block gets evicted (victim)

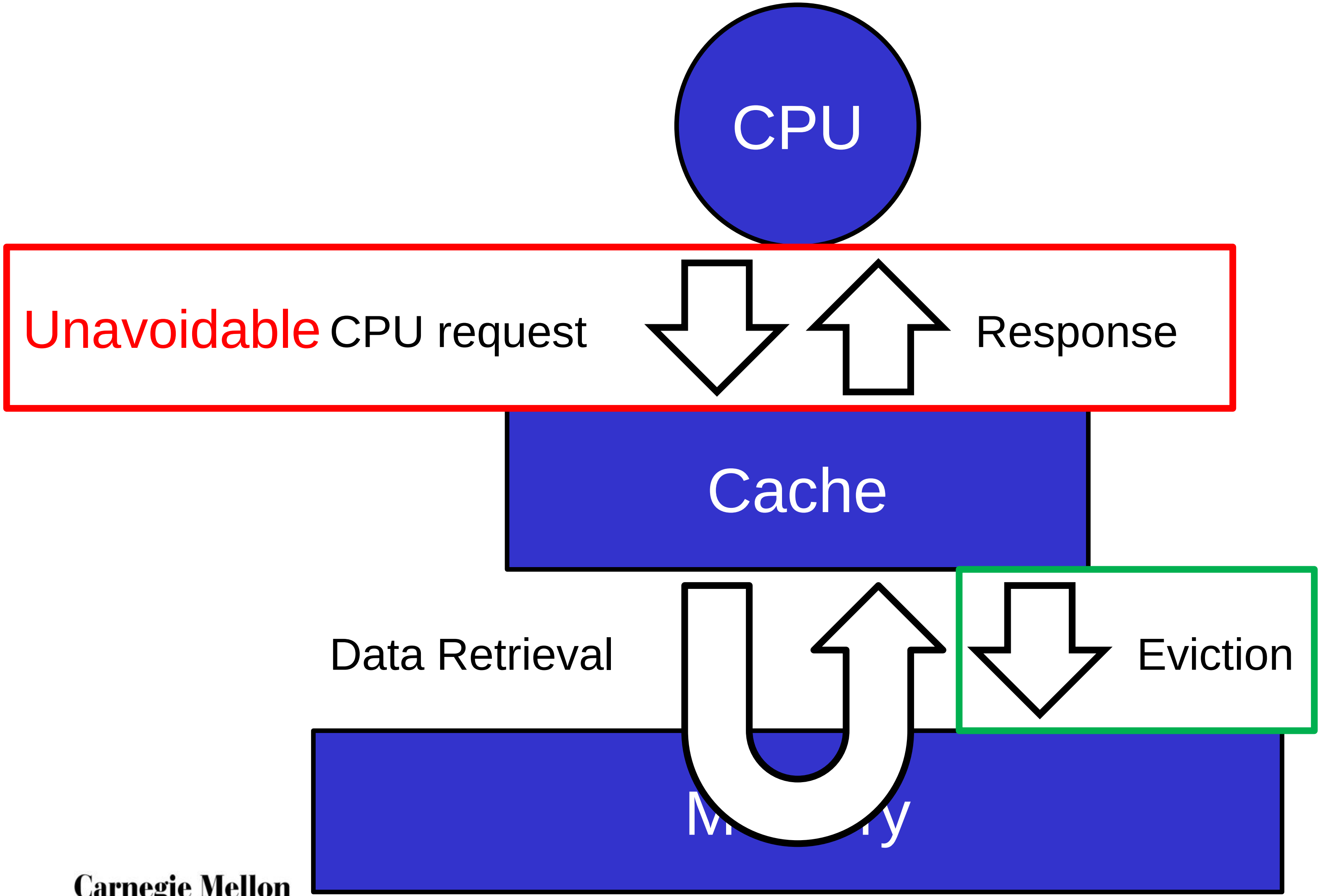
Modeling Misses



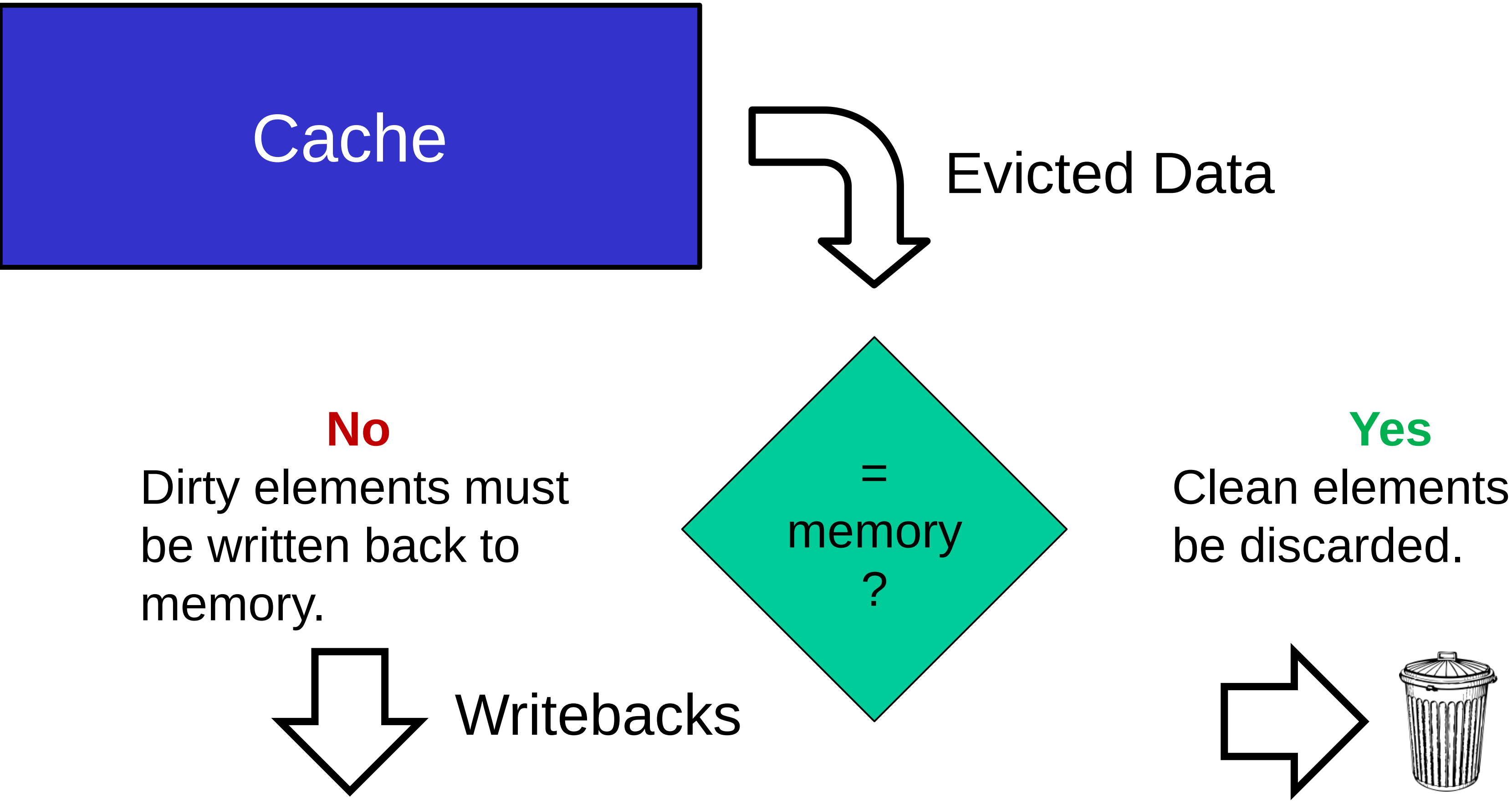
Cache Performance Metrics

- **Miss Rate**
  - Fraction of memory references not found in cache (misses / accesses) = 1 – hit rate
  - Typical numbers (in percentages):
    - 3-10% for L1
    - can be quite small (e.g., < 1%) for L2, depending on size, etc.
- **Hit Time**
  - Time to deliver a line in the cache to the processor
    - includes time to determine whether the line is in the cache
  - Typical numbers:
    - 4 clock cycle for L1
    - 10 clock cycles for L2
- **Miss Penalty**
  - Additional time required because of a miss
    - typically 50-200 cycles for main memory (Trend: increasing!)

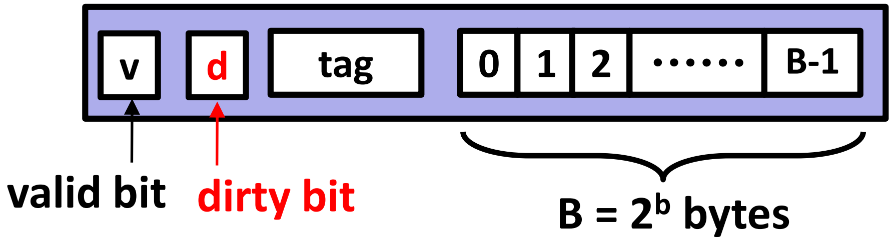
Modeling Miss Rate



Evictions

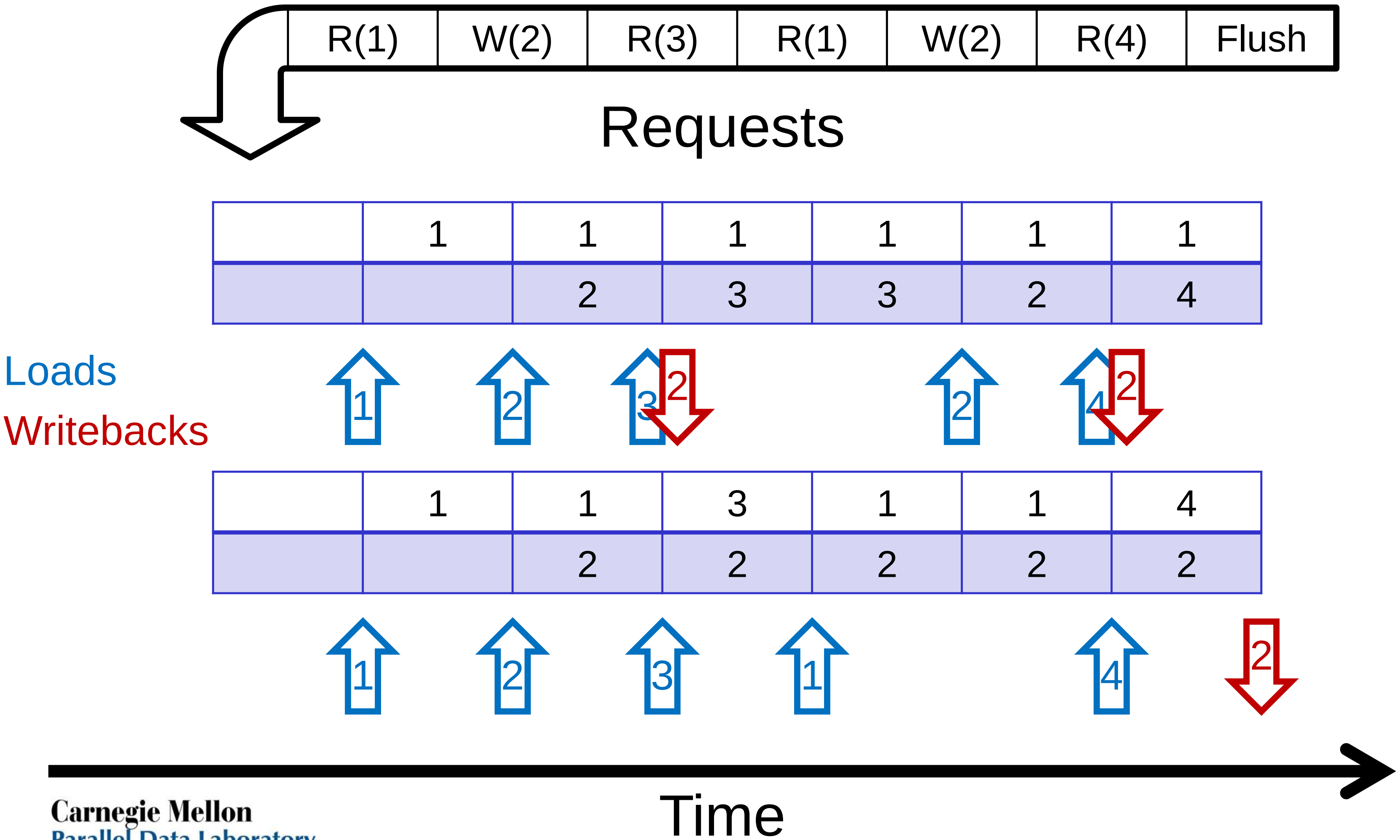

Cache
Evicted Data
No
Dirty elements must be written back to memory.
= memory ?
Yes
Clean elements can be discarded.
Writebacks

What about writes?



- **Multiple copies of data exist:**
  - L1, L2, L3, Main Memory, Disk
- **What to do on a write-hit?**
  - Write-through (write immediately to memory)
  - Write-back (defer write to memory until replacement of line)
    - Each cache line needs a dirty bit (set if data differs from memory)
- **What to do on a write-miss?**
  - Write-allocate (load into cache, update line in cache)
    - Good if more writes to the location will follow
  - No-write-allocate (writes straight to memory, does not load into cache)
- **Typical**
  - Write-through + No-write-allocate **1 memory write / CPU write**
  - **Write-back + Write-allocate**

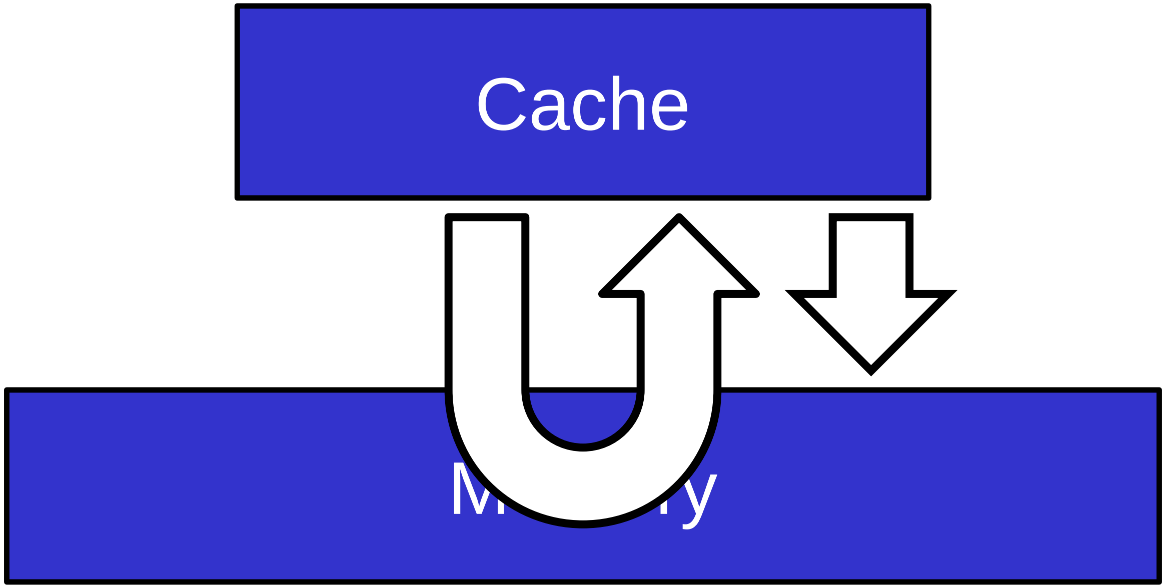
Combining Writebacks



Why Writebacks Matter

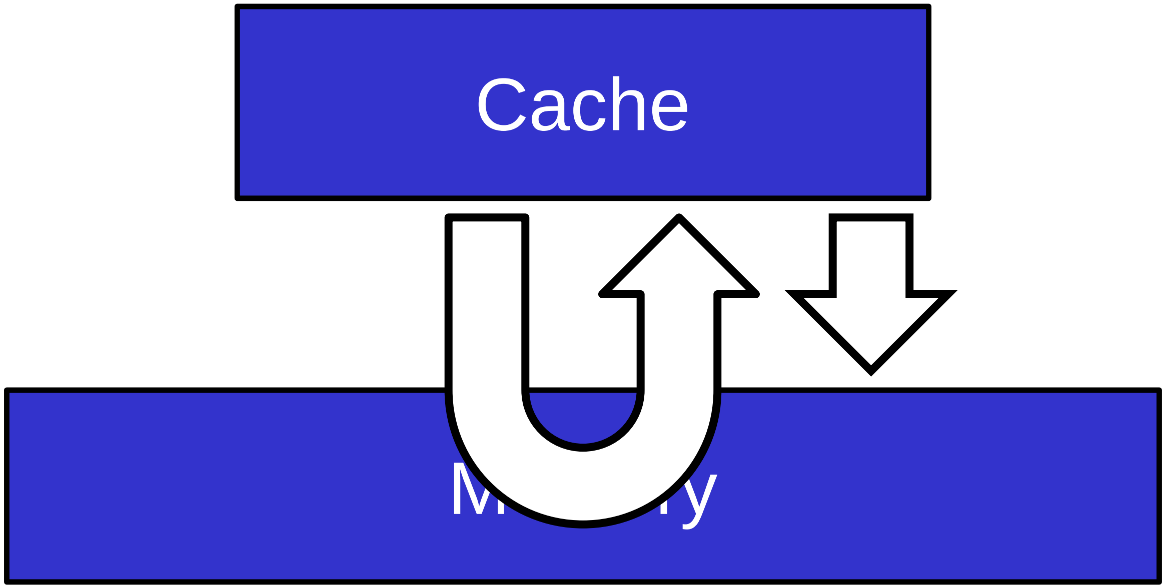
- Bandwidth
- Energy
- Wearout

These metrics are important to non-volatile memory and storage.



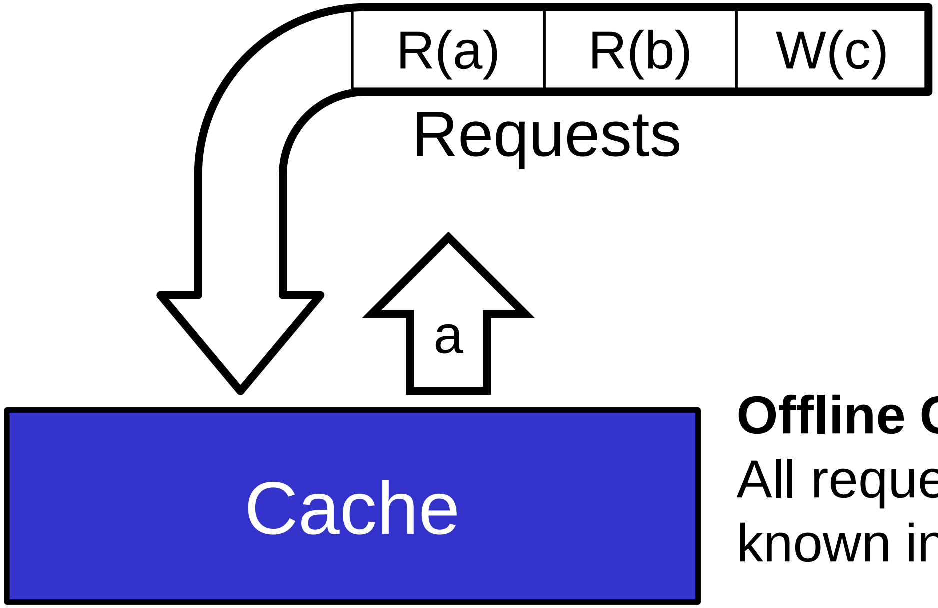
A New Metric

A weighted sum of the cost due to loads and the cost due to writebacks.



$$\sum_{\forall\, misses\ p} Cost_{Load}(p) + weight \times \sum_{\forall\, WBs\ q} Cost_{WB}(q)$$

Versions of the Problem



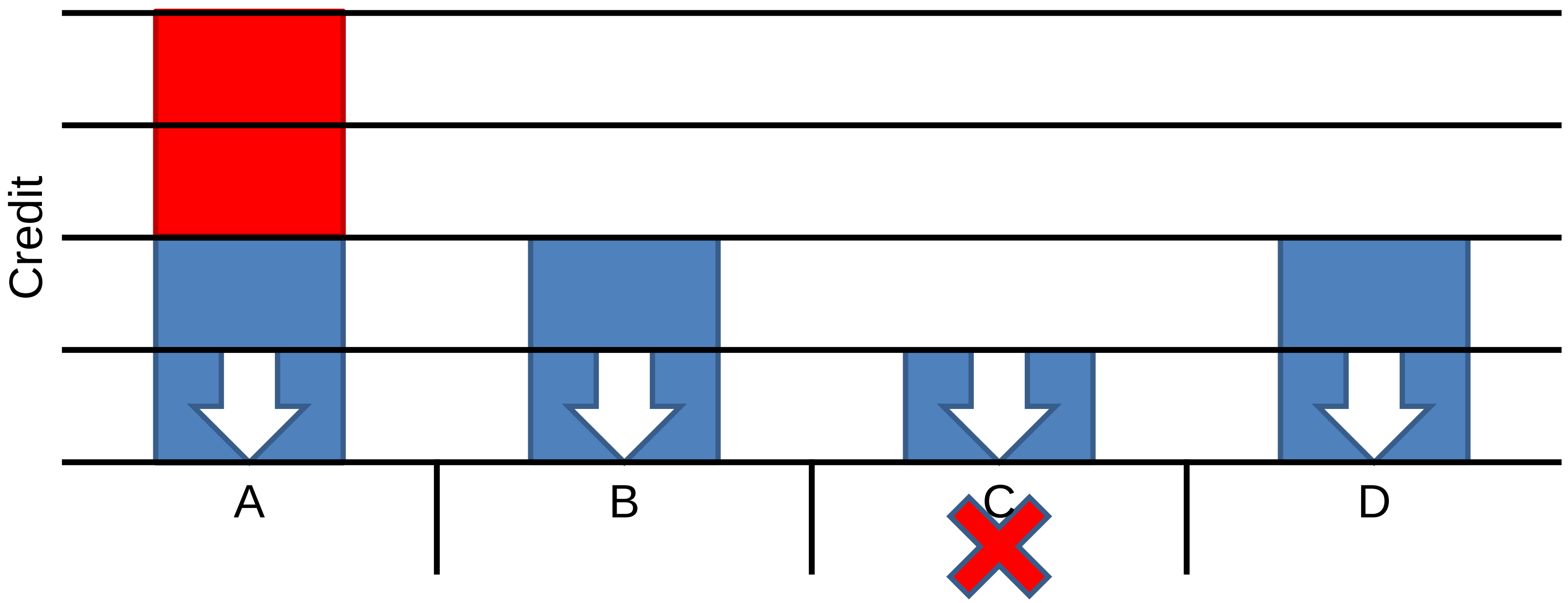
Online Caching:
A request must be served before the next appears.

Offline Caching:
All requests are known in advance.

Summary of Results

- Modeling Writebacks
  - Theoretical model that generalizes caching
- Offline Problem:
  - Analysis of writeback-oblivious policies
  - Complexity results
  - Approximation algorithms
- Online Policy: Writeback-Aware Landlord
  - Optimal worst-case analysis
  - Good empirical performance

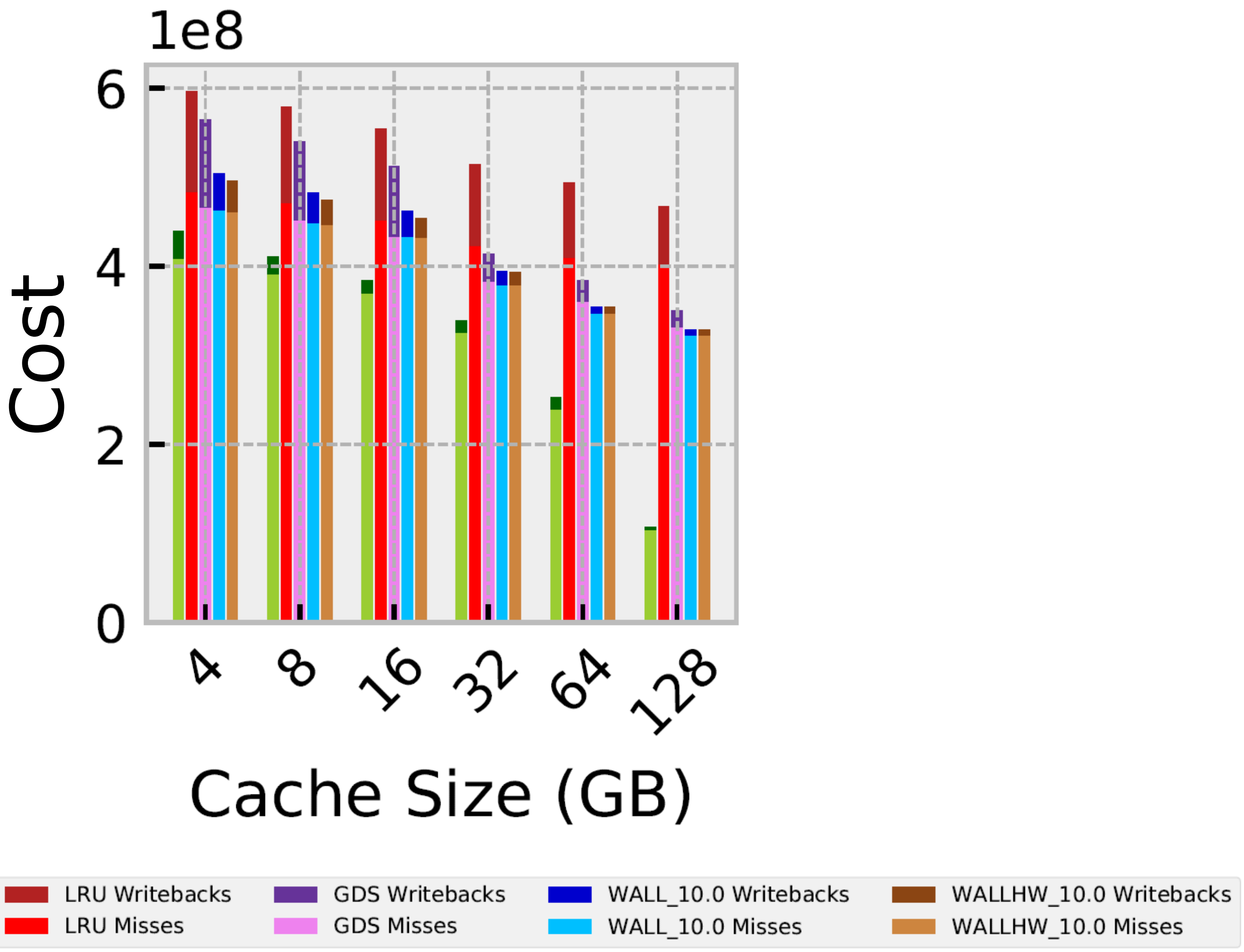
Writeback-Aware Landlord



Cache Simulation

- MSR storage traces
  - Disk requests for real MSR servers
- Competing policies:
  - Offline approximation (WAPFOO-L)
  - Least Recently Used (LRU)
  - Greedy-Dual Size (GDS)
- Metric:
  - Loads have unit cost
  - Writes have 10x the cost

Simulation Results on src1_1



More Results

- Other MSR traces
  - Shows broader impact
- Applying the frequency metric
  - Traditionally useful heuristic helps here
- Sensitivity to write/read cost ratio
  - Generally performs well

Summary of Results

- Modeling Writebacks
  - Theoretical model that generalizes caching
- Offline Problem:
  - Analysis of writeback-oblivious policies
  - Complexity results
  - Approximation algorithms
- Online Policy: Writeback-Aware Landlord
  - Optimal worst-case analysis
  - Good empirical performance

More Caching Research

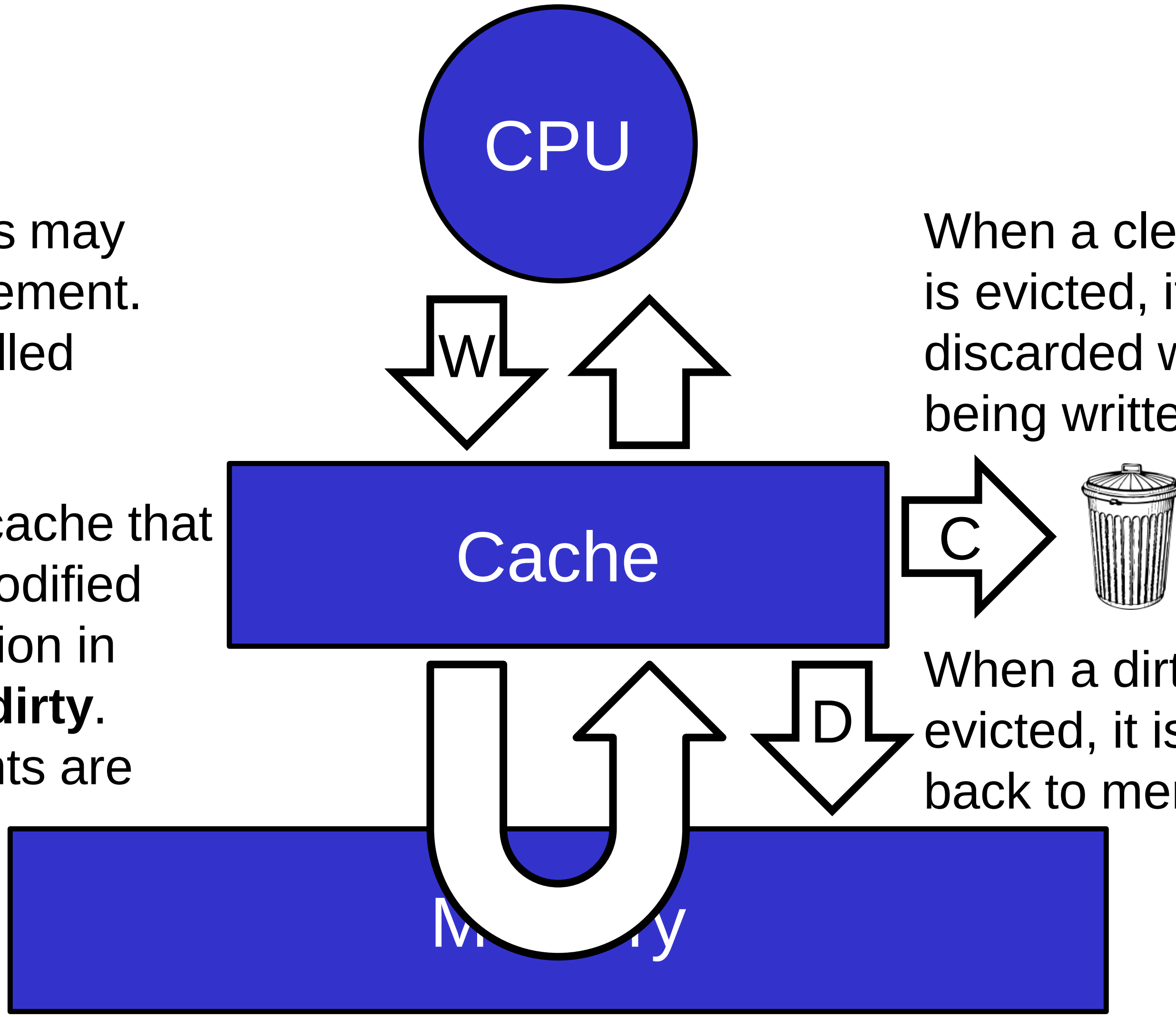
- This work
- Cache coherence
  - Ensuring data consistency
- Parallel Caching
  - Multiple CPUs sharing a cache
- Distributed Caching
  - Data can be in different locations
- Web Caching
  - Objects have different size and cost

Backup Slides

Writebacks

CPU requests may modify the element. These are called **writes**.

Elements in cache that have been modified from the version in memory are **dirty**. Other elements are **clean**.

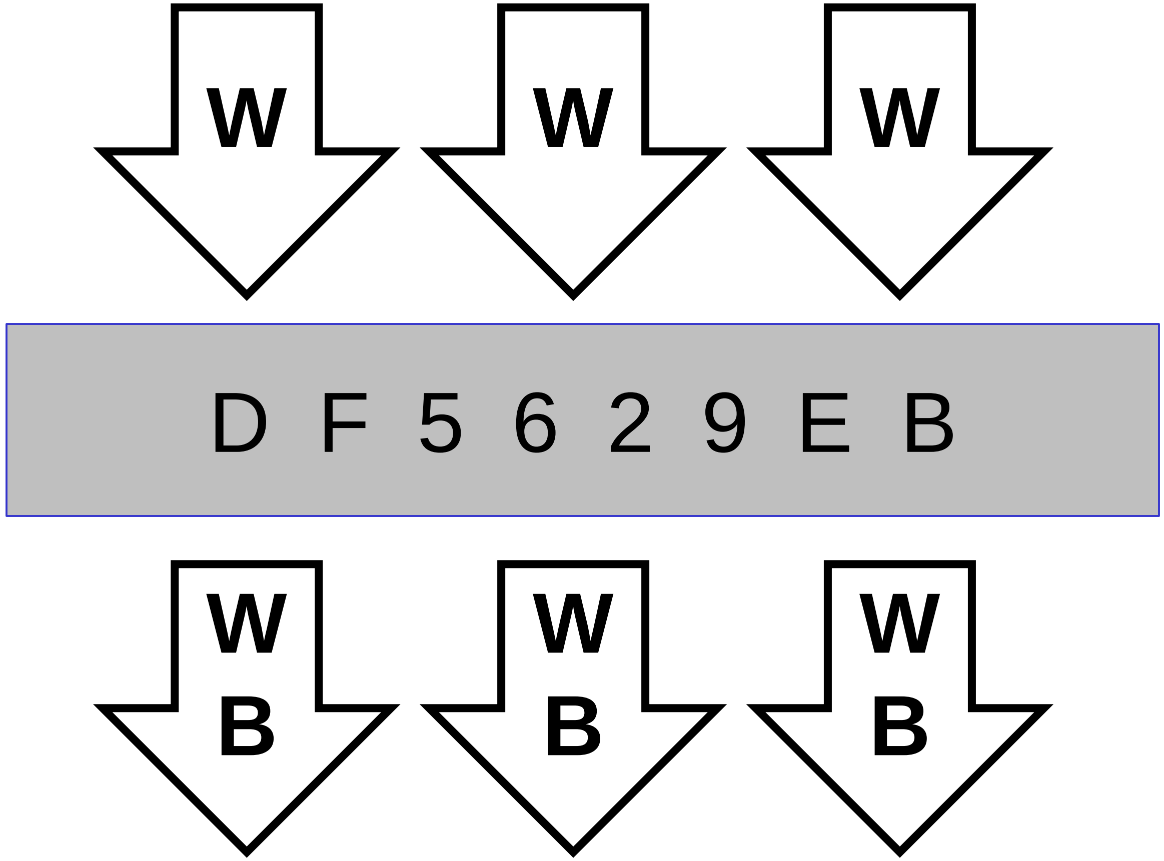


When a clean element is evicted, it can be discarded without being written back.

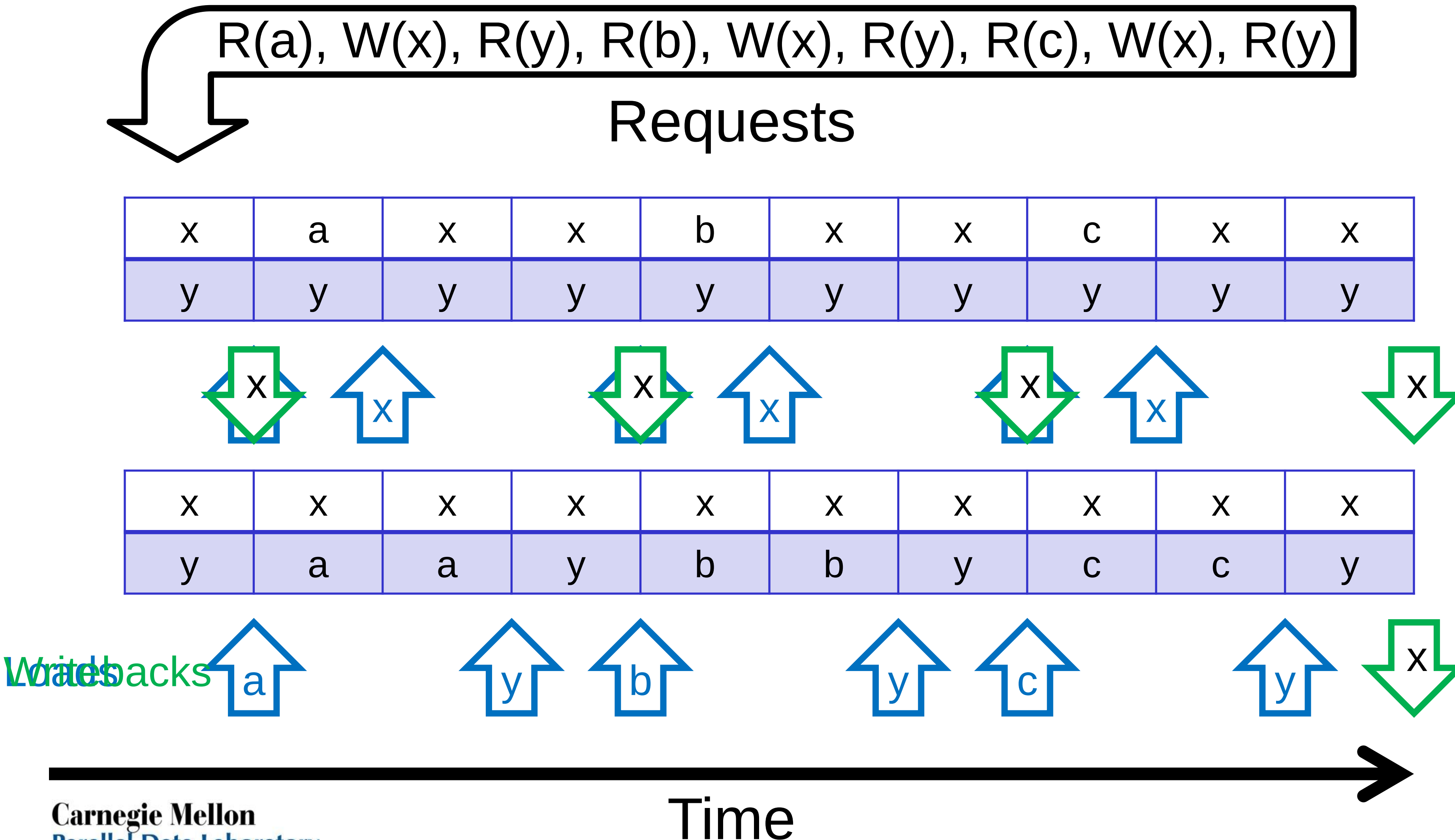
When a dirty element is evicted, it is written back to memory.

Carnegie Mellon
Parallel Data Laboratory

Combining Writebacks



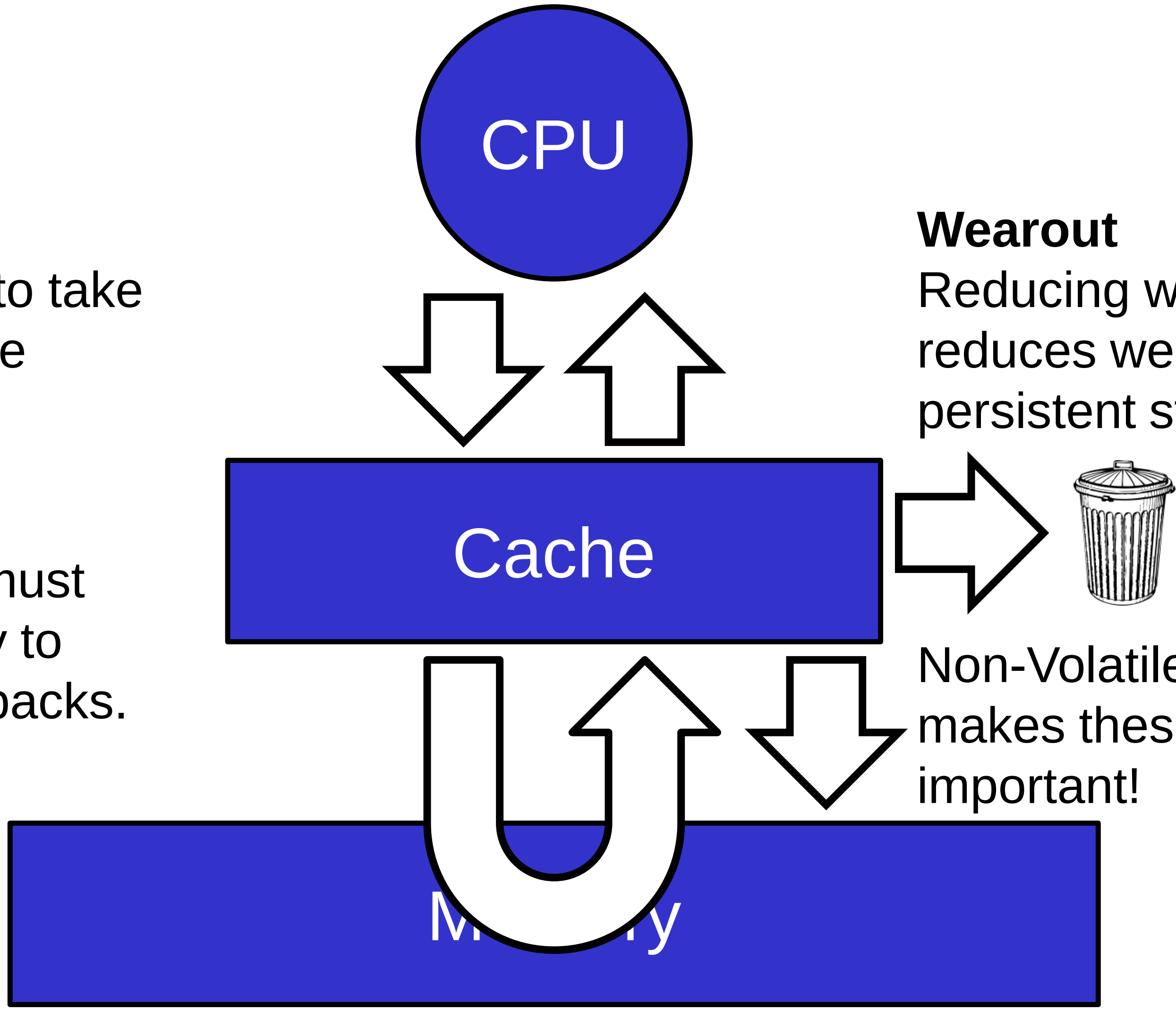
Combining Writebacks



Why Writebacks Matter

Bandwidth
Memory has to take time to service writebacks.

Energy
The system must spend energy to service writebacks.



Wearout
Reducing writebacks reduces wear on persistent storage.

Non-Volatile Memory makes these more important!

Considering Loads AND Writebacks

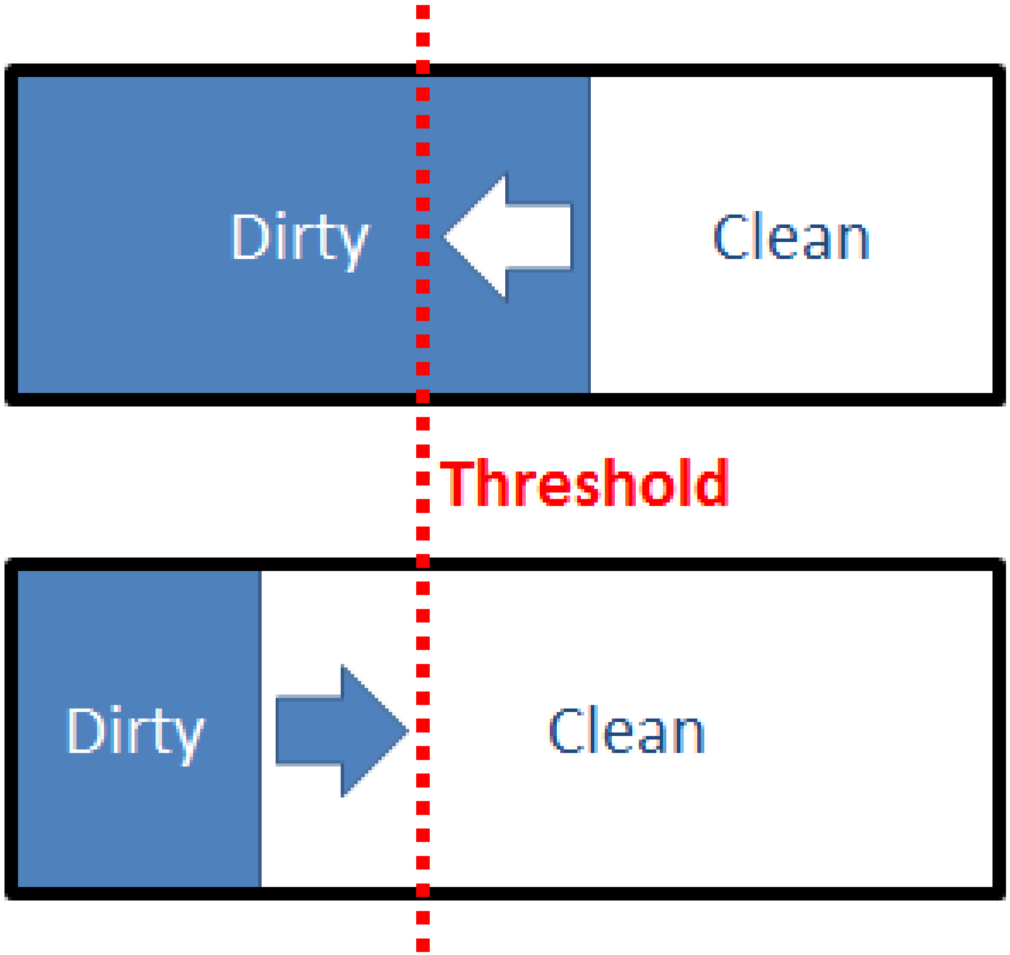
Traditional caches minimize:

$$\sum_{\forall\, misses\ p} Cost_{Load}(p)$$

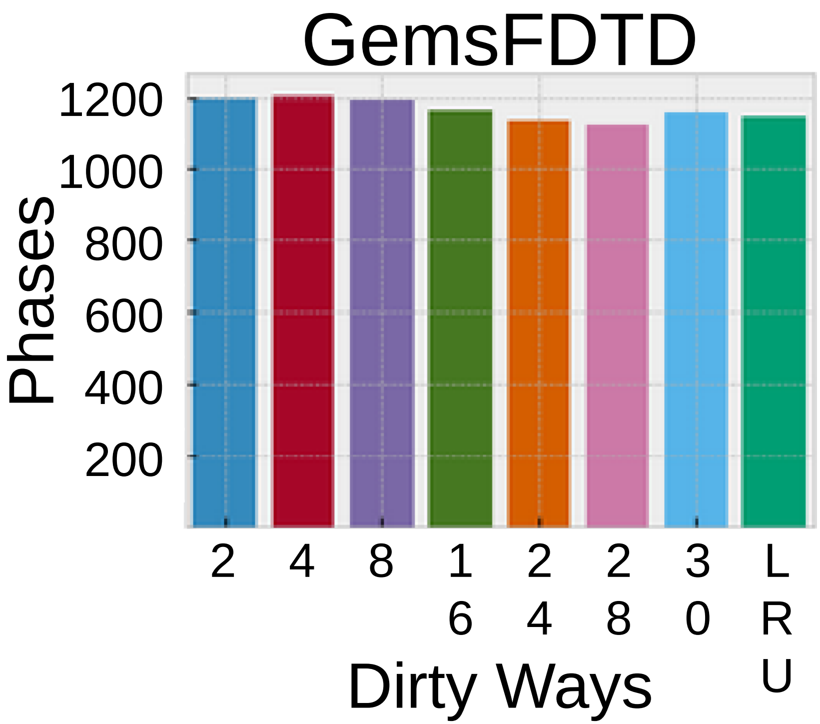
Our work seeks to minimize:

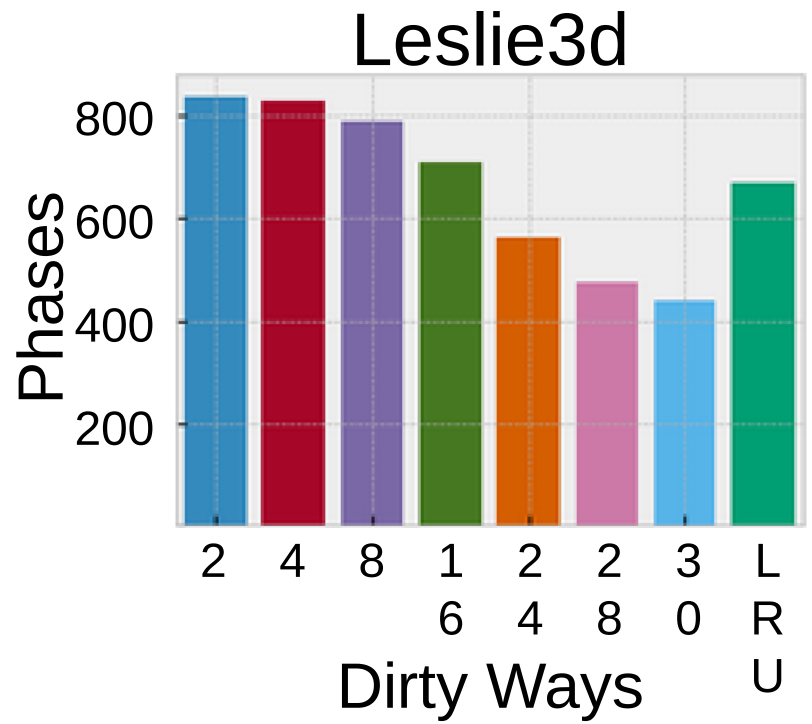
$$\sum_{\forall\, misses\ p} Cost_{Load}(p) + weight \times \sum_{\forall\, WBs\ q} Cost_{WB}(q)$$

Cache Partitioning

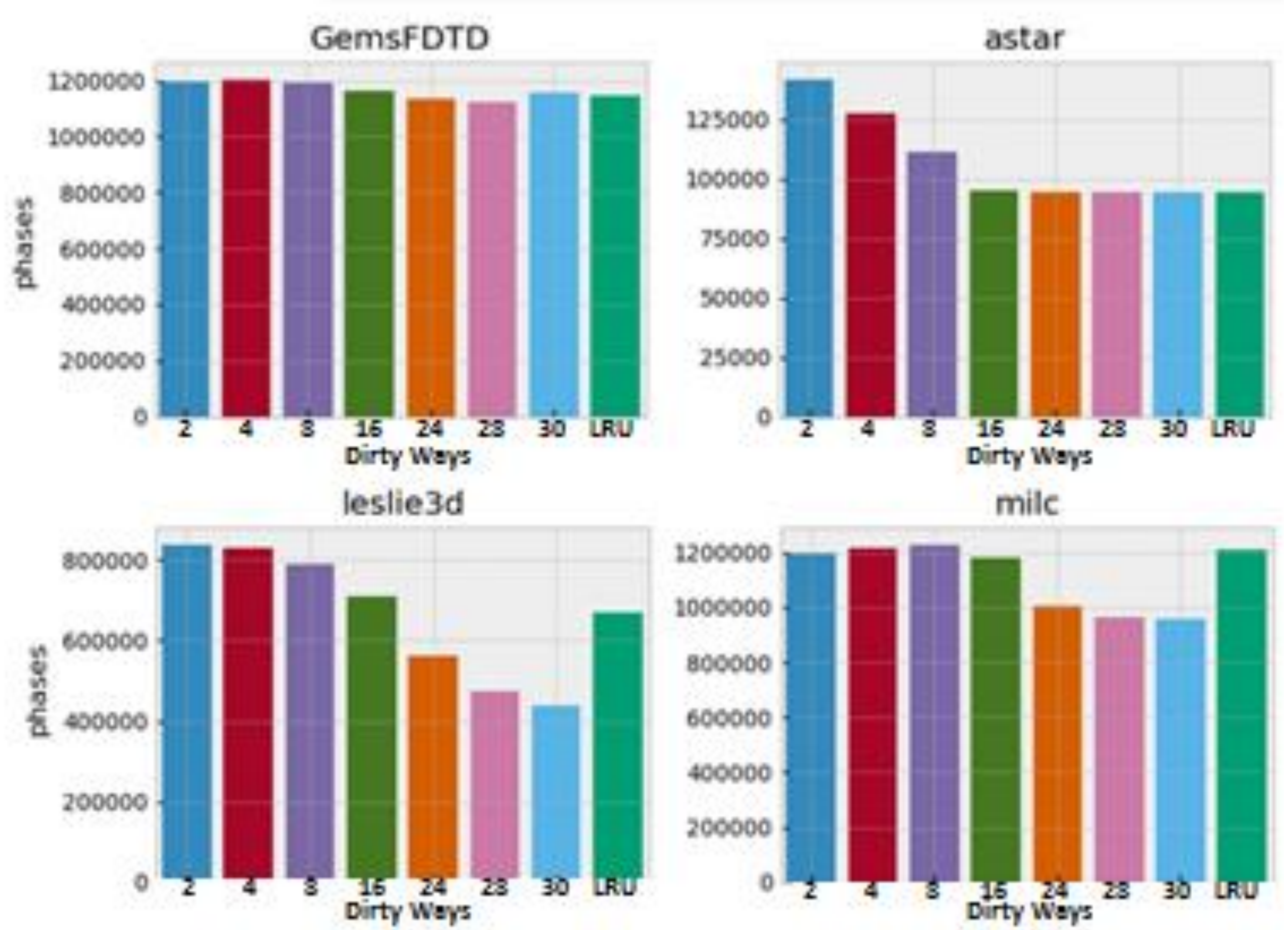


Initial Results





Initial Results



Summary of Results

- Offline Problem:
  - Non-optimality of traditional policies
  - Proof of NP-Completeness
  - Proof of APX-Completeness
  - Approximation Algorithms
- Online Problem:
  - Dirty/Clean Cache Partitioning
  - WA-RRIP
  - WA Landlord

Come see our poster for more details!