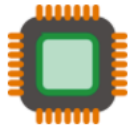


CS305: Computer Architecture

Caches-III

https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

Cache misses

Cold Miss: cache starts empty and this is the first reference

Conflict Miss: Many mapped to the same index bits

Capacity Miss: Cache size is not sufficient

Coherence Miss: in Multi-core systems, only [not I/O coherence]

On a Miss, Replace a block, which block?

Think of each block in a set having a “priority”
Indicating how important it is to keep the block in the cache

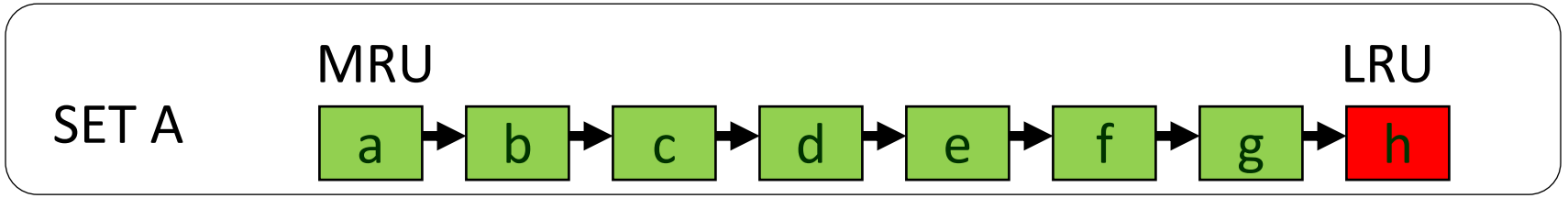
Key issue: How do you determine/adjust block priorities?

Ideally: Belady’s OPT policy, replace the block that will be used furthest in the future. No one knows the future though ☺

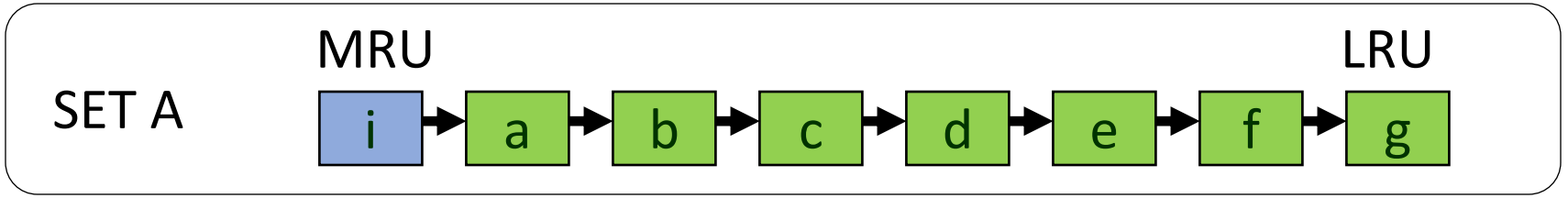
There are three key decisions in a set:
Insertion, promotion, eviction (replacement)

A simple LRU (Least-Recently-Used) Policy

Cache Eviction Policy: On a miss (block *i*), which block to evict (replace) ?



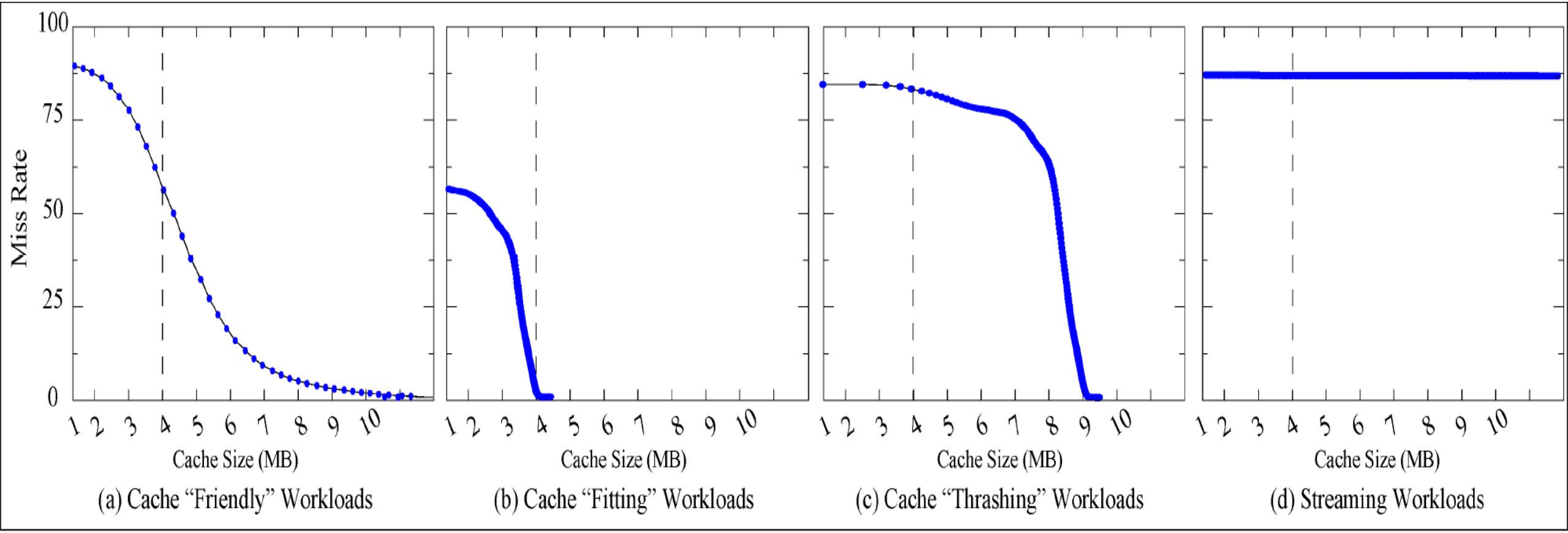
Cache Insertion Policy: New block *i* inserted into MRU.



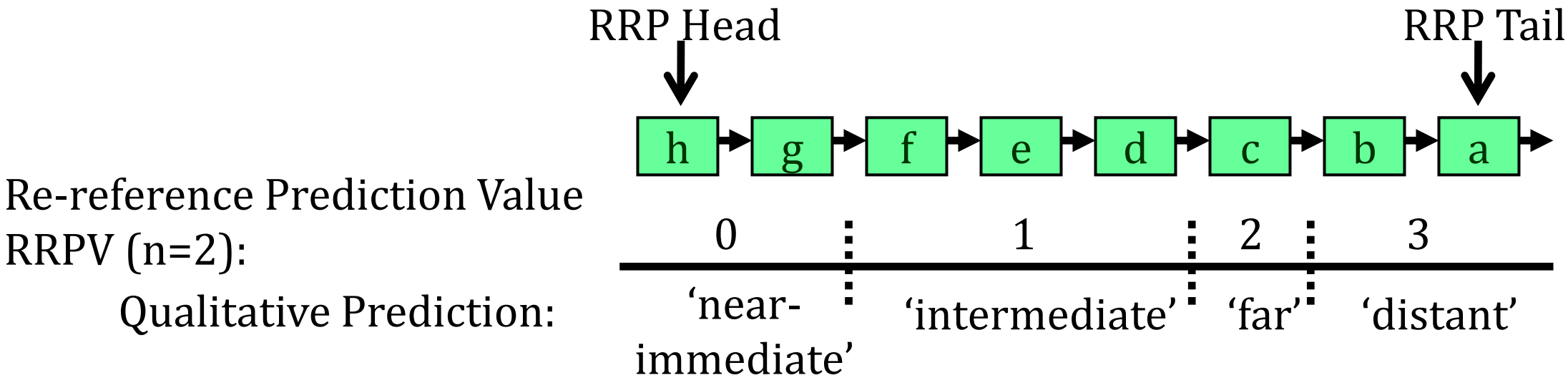
Cache Promotion Policy: On a future hit (block *i*), promote to MRU

We need priority bits per block. For example, a 16-way cache will need four bit/block LRU causes thrashing when working set > cache size

Types of Applications



LRU is not effective for Shared Caches



Intuition: New cache block will not be re-referenced soon. Replaces block with distant RRPV. Only two bits per block.

Insert with RRPV=2, Evict with RRPV=3, increment RRPVs till we get a block with RRPV=3, promote blocks with RRPV=0.

Let's redefine cache misses

Compulsory: first reference to a line (a.k.a. cold start misses)

- *misses that would occur even with infinite cache*

Capacity: cache is too small to hold all data

- *misses that would occur even under perfect (Belady's) replacement policy*

Conflict: misses that occur because of collisions due to line-placement strategy

- *misses that would not occur with ideal full associativity*

Cache knobs and Misses

- Larger cache size
  +reduces capacity and conflict misses
  - hit time will increase
- Higher associativity
  +reduces conflict misses
  - may increase hit time
- Larger line size
  +reduces compulsory misses
  - increases conflict misses and miss penalty

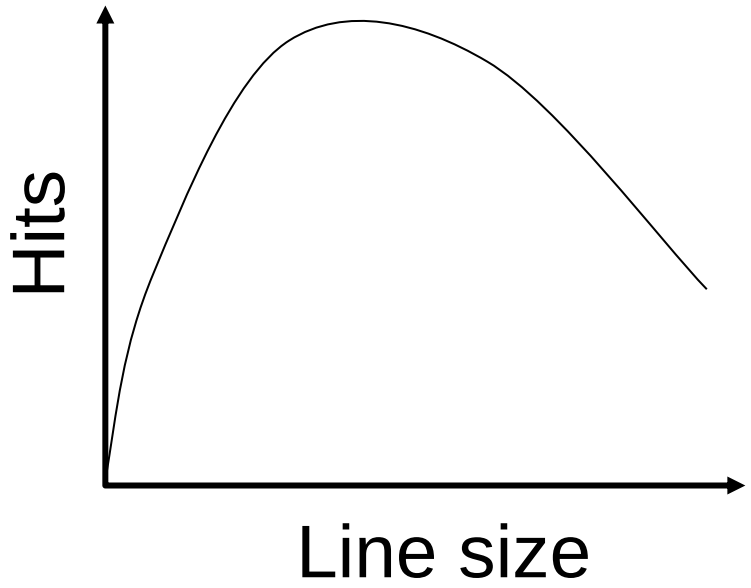
Line size

Too small blocks:

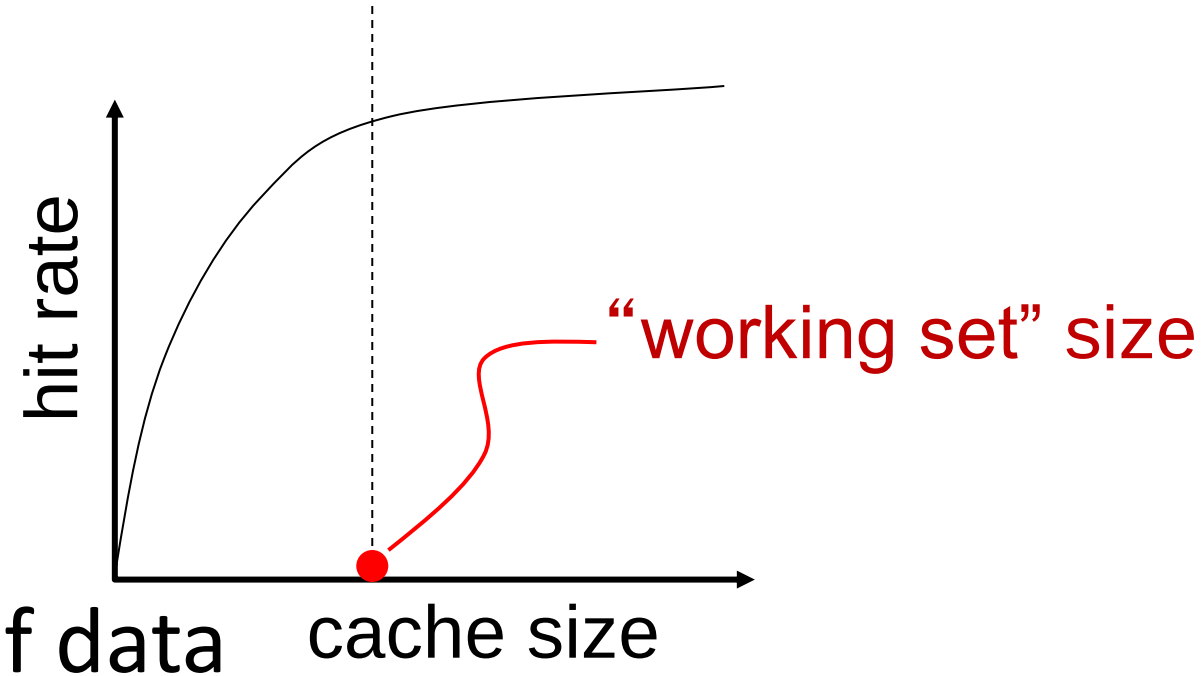
- don’t exploit spatial locality well
- have larger tag overhead

Too large blocks:

- too few total # of blocks
  - likely-useless data transferred
  - Extra bandwidth/energy consumed



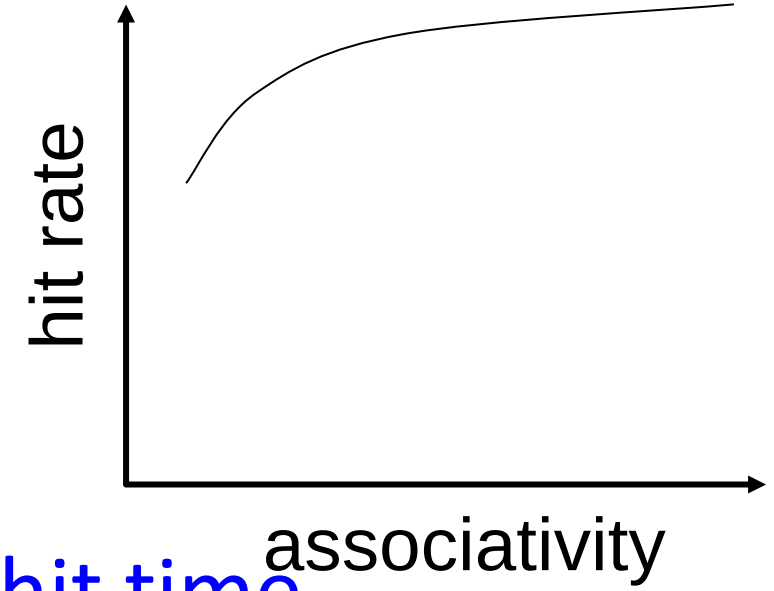
Cache size



Working set: the whole set of data the executing application references within a time interval

Associativity

Myth: It should be power of two. NO!!



L1 cache: lower associativity, hit time

L3 cache: higher associativity

Cache Performance

How good is the cache for a given application?

Hit rate

Miss rate

Misses per kilo instructions (MPKI)

But Why?

Average Access Time

On average, how much time it takes for a LOAD to complete

Average memory access time (AMAT) =

Hit time + Miss rate x Miss penalty

Hit time-L1 + Miss rate-L1 x Miss penalty-L1

Miss penalty-L1 = Hit time-L2 + Miss rate-L2 x Miss penalty-L2

Hit time: Low, Miss rate: Low , Miss penalty: Low

Ideally, miss rate = 0.00% and hit time should be one cycle, so all LOADs will take just one cycle ☺

Au Kun