Lecture 10: Large Cache Design III

• Topics: Replacement policies, prefetch, dead blocks, associativity

• Sign up for class mailing list

• Pseudo-LRU has a 9% higher miss rate than true LRU
Overview

MIP: MRU insertion policy (traditional approach)

DIP: Selects the best

LIP: LRU insertion policy

TADIP: Selects the best for each thread

BIP: Bimodal insertion policy
Few insertions at head, most at tail

RRIP: Probabilistic insertion near tail

PIPP: Inserts each thread at different positions + probabilistic promotion

AGGRESSOR-VT:
Victimizes the aggressor thread with a high probability

UCP: Partitions ways across threads based on marginal utility

Highest priority ← Priority stack of blocks in a set → Lowest priority
Set Partitioning

• Can also partition sets among cores by assigning page colors to each core

• Needs little hardware support, but must adapt to dynamic arrival/exit of tasks
• Memory level parallelism (MLP): number of misses that simultaneously access memory; high MLP $\rightarrow$ miss is less expensive

• Replacement decision is a linear combination of recency and MLP experienced when fetching that block

• MLP is estimated by tracking the number of outstanding requests in the MSHR while waiting in the MSHR

• Can also use set dueling to decide between LRU and LIN
A fill stack is a FIFO that tracks the order in which blocks entered the set.

Most hits are serviced while a block is near the top of the stack; there is usually a knee-point beyond which blocks stop yielding hits.

Evict the highest block above the knee that has not yet serviced a hit in its current fill stack position.

Allows some blocks to probabilistically escape past the knee and get retained for distant reuse.
Scavenger

- Half the cache is used as a victim cache to retain blocks that will likely be used in the distant future

- Counting bloom filters to track a block’s potential for reuse and make replacement decisions in the victim cache

- Complex indexing and search in the victim cache

- Another recent paper (NuCache, HPCA’11) places blocks in a large FIFO victim file if they were fetched by delinquent PCs and the block has a short re-use distance
V-Way Cache
Qureshi et al., ISCA’05

• Meant to reduce load imbalance among sets and compute a better global replacement decision

• Tag store: every set has twice as many ways

• Data store: no correspondence with tag store; need forward and reverse pointers

• In most cases, can replace any block; every block has a 2b saturating counter that is incremented on every access; scan blocks (and decrement) until a zero counter is found; continue scan on next replacement
ZCache  
Sanchez and Kozyrakis, MICRO’10

- Skewed associative cache: each way has a different indexing function (in essence, W direct-mapped caches)

- When block A is brought in, it could replace one of four (say) blocks B, C, D, E; but B could be made to reside in one of three other locations (currently occupied by F, G, H); and F could be moved to one of three other locations

- We thus get a tree of replacement options and we can pick LRU among these options

- Every replacement requires multiple tag look-ups and data block copies; worthwhile if you’re reducing off-chip accesses
• When a thread incurs a series of misses to blocks (PQRS, not necessarily contiguous), other threads are likely to incur a similar series of misses

• Each thread maintains its miss log in a circular buffer in memory; the directory for P also keeps track of a pointer to multiple log entries of P

• When a thread has a miss on P, it contacts the directory and the directory provides the log pointers; the thread receives multiple streams and starts prefetching

• Log access and prefetches are off the critical path
• Threads often enter a new region (page) and touch a few arbitrary blocks in that region

• A predictor is indexed with the PC of the first access to that region and the offset of the first access; the predictor returns a bit vector indicating the blocks accessed within that region

• Can even prefetch for regions that have not been touched before!
Feedback Directed Prefetching  

A stream prefetcher has two parameters:

- P: prefetch distance: how far ahead of the start do we prefetch
- N: prefetch degree: how much do we advance the start when there is a hit in the stream

- Can vary these two parameters based on pref effectiveness

- Accuracy: a bit tracks if a prefetched block was touched

- Timeliness: was the block touched while in the MSHR?

- Pollution: track recent evictions (Bloom filter) and see if they are re-touched; also guides insertion policy
Dead Block Prediction

• Can keep track of the number of accesses to a line during its previous residence; the block is deemed to be dead after that many accesses. Kharbutili, Solihin, IEEE TOC’08

• To reduce noise, an access can be considered as a block’s move to the MRU position. Liu et al., MICRO 2008

• Earlier DBPs used a trace of PCs to capture when a block has completed its use

• DBP is used for energy savings, replacement policies, and cache bypassing
Distill Cache

- Half the ways are traditional (LOC); when a block is evicted from the LOC, only the touched words are stored in a word-organized cache that has many narrow ways.
- Incurs a fair bit of complexity (more tags for the WOC, collection of word touches in L1s, blocks with holes, etc.).
- Does not need a predictor; actions are based on the block’s behavior during current residence.
- Useless word identification is orthogonal to cache compression.
Title

• Bullet