

Discussion 11

Caches

Aditya Balasubramanian

`aditbala [at] berkeley [dot] edu`

Announcements 📢

Agenda

- Cache Basics
- TIO Breakdown
- Cache Workflow
- 3 C's of Cache Misses
- Multilevel Caches + AMAT

Caches

- The closer we get to the CPU, the faster we can load data
- Registers
 - On the CPU, blazingly fast, super tiny
- Memory
 - On the motherboard, still pretty fast, pretty big
- Between them, we have the Cache
 - On the CPU, faster than memory, small-ish
- We turn memory addresses into TIO bits to place it into the cache

Data that's close

- Spatial Locality
 - Close with respect to location in memory
 - Data that “lives” +/- a certain amount from the accessed address might be accessed soon
- Temporal Locality
 - Close with respect to time
 - Data that's accessed recently might be accessed soon again

Cache Terminology

- Cache line
 - a chunk of data that is loaded into the cache from main memory upon an access (also known as block)
- Set
 - groups of cache lines to form a larger group indexed by the index
- Associativity
 - number of cache lines needed to form a set (4-way associative cache means each set contains 4 blocks)

Types of Caches

- Direct Mapped
 - Each address can only go into one fixed cache line
 - Easiest to implement on hardware
- N-way Set Associative
 - Each address can go into one fixed set (N cache lines)
- Fully Associative
 - Each address can go into any cache line

TIO Breakdown

tag	index	offset
-----	-------	--------

- Tag
 - $\log_2 (\text{Memory size in bytes}) - I - O$
- Index
 - $\log_2 (\text{Number of blocks} / \text{associativity}) = \log_2 (\text{Number of indices})$
- Offset
 - $\log_2 (\text{Size of block in bytes})$

Workflow

- We know we're accessing address, say, 0xDEADBEEF, now what?
 1. **Breakdown address** based on TIO
 2. **Use index** to figure out which set our cache line goes into
 3. Within the set, **check the tag** of every existing cache line to see if the given tag matches.
 - a. If any tag matches, go to step 4, otherwise go to step 5
 4. Tag match! **Check the valid bit** to make sure we still have an updated copy of the cache line.
 - a. If the valid bit is 1, then data is already in the cache, so we can get the data using the offset
 - b. If valid bit is 0 (invalid), continue with step 5
 5. **Fetch the entire cache line** (block aligned!) into the cache
 - a. What if the set is full?
 - i. We need to evict some previous line...
 - ii. Which line to evict?

Eviction Policy

- Least Recently Used (LRU)
 - Probably most common
 - Least recently used block get evicted
- First In First Out (FIFO)
 - The oldest block in the set gets evicted
- Others
 - Most Recently Used (MRU)
 - First In Last Out (FILO)

Cache Misses

- Compulsory Miss
 - We haven't seen this block before
 - ALWAYS happens the first time we pull a block into the cache
 - Can be reduced by increasing block size
- Conflict Miss
 - We had to evict this block because some other block replaced it
 - Can be reduced by increasing associativity or having a better replacement policy
- Capacity Miss
 - We had to evict this block because the cache wasn't big enough
 - Can be reduced by increasing cache size

Multi-level Caches

- Usually has L1, L2 cache and occasionally L3
- Only cache misses from lower level will go to the next level!
 - If L1 hits, no need to look in L2
- Global Miss Rate
 - Miss rate of a cache level in the scope of the entire system
 - $MR = \# \text{ misses at this level} / \# \text{ total accesses of cache}$
- Local Miss Rate
 - Miss rate of a cache level in the scope of the level itself
 - $MR = \# \text{ misses at this level} / \# \text{ total access in this level}$

AMAT

$$\text{AMAT} = \text{Hit Time} + \text{Miss Rate} \times \text{Miss Penalty}$$

- Hit Time
 - Time it takes to get the data if it's in the cache already
- Miss Rate: local miss rate
 - # misses / # accesses in a level
- Miss Penalty
 - Time it takes to get the data if it's not in the cache
 - Single-level cache: MP = time it takes to access main memory
 - Multi-level cache: MP = AMAT of the next level
- e.g. 2 level cache

$$\text{AMAT} = \text{L1 HT} + \text{L1 MR} * (\text{L2 HT} + \text{L2 MR} * \text{Main Memory Access})$$

Thank you!