Tutorial 9

Caches (2)

What about cache misses?

Cache misses can occur for various reasons.

- Compulsory: first access to a block. It cannot already be in the cache
- Capacity: more blocks are needed during the execution of the program than will fit in cache
- Conflict: For set-associative and direct caches, many blocks needed by a program map to the same location in the cache.

What about cache misses?

When you have a n-way associative cache, you can replace a block in the cache with a newer one,

- At random: replace a random block in the set
- Least-recently used (LRU): replace the block which was the least recently used
- First in, first out (FIFO): replace the block which was put first into the cache

What do you do when you write?

When you do a write, you have to ask yourself if you want the written data to live only in the cache, or to be written all the way to main memory.

- Write through: write your change in the cache and in main memory (ensures consistency)
  - +: simpler to implement, better data coherency, read misses don't lead to writes to slower memory
- Write back: write your change only to the cache (and it will be reflected in main memory when the block is replaced in the cache)
  - +: writes at speed of cache. Multiple writes only require a single write to slower memory, less bandwidth, less power

What do you do when you write?

When you write to the cache, a <u>write miss</u> can occur. Two options here:

- <u>Write allocate</u>: when there is a write miss, the block is loaded in the cache and the write is re-attempted
- <u>No-write allocate</u>: when there is a write miss, only main memory is changed (block not loaded in the cache)

Cache Performance

- Average access time is usually used to assess performance of a cache strategy (but total execution time of the program is the only completely reliable indicator)
- Avg access time = Hit time + miss rate * miss penalty
- Miss rate = ((misses/1000instructions) / 1000) / (memory accesses/instruction)

- Hit time: time to hit the cache
- Miss rate (per memory access): how many of the memory accesses miss the cache
- Miss penalty: how long to recover from a miss

Cache Performance Example

- 16KB instruction + 16KB data cache; 36% of instructions are data transfer instructions. Hit=1CC, miss=200CC; 3.82 misses/1000 instructions for instructions; 40.9 misses/1000 instructions for data;74% memory accesses are for instructions and 26% for data.
- Miss rate_i = (3.82/1000) / 1.00 = 0.004
- Miss rate_d = (40.9/1000) / 0.36 = 0.114
- Avg access time = 74%*(1+0.004*200) + 26%*(1+0.114*200) = 7.52CC

Cache Performance Example

- 32KB unified cache; 36% of instructions are data transfer instructions; Hit=1CC, miss=200CC; 43.3 misses/1000 instructions. No structural hazards

- Miss rate = (43.3/1000) / (1 + 0.36) = 0.0318
- Avg. access time = 1 + 0.0318*200 = 7.36
- Miss rate and access time are <u>per memory access</u>

Multi-Level Cache Performance

- Miss penalty_{L1} = hit time_{L2} + miss rate_{L2} * miss penalty_{L2}
- Avg access time = hit time_{L1} + miss rate_{L1} * miss penalty_{L1}

  = hit time_{L1} + miss rate_{L1} * (hit time_{L2} + miss rate_{L2} * miss penalty_{L2})
- Local miss rate: misses in a cache / total accesses to this cache
- Global miss rate: misses in cache / total memory accesses by processor

Multi-Level Cache Example

- Suppose 40 misses in L1 / 1000 mem ref; 20 L2 misses / 1000 mem ref; Hit_{L2}=10CC, Miss_{L2}=200CC, Hit_{L1}=1CC.
- MissRate_{L1}=40/1000; MissRate_{L2}=20/40
- Avg access time = hit time_{L1} + miss rate_{L1} * (hit time_{L2} + miss rate_{L2} * miss penalty_{L2})

  = 1 + 4% * (10 + 50% * 200) = 5.4CC
- What if you had a third level of cache?

Midterm Review

- Three questions (15% of your term grade)
  - 1 short answers
  - 2 longer problems
- Covers lectures 14-23
  - Dynamic scheduling
  - Multiple-issue processors
  - Speculation
  - Memory hierarchy and caches

Midterm Review

- Dynamic scheduling and speculation
  - Tomasulo algorithm
    - Implicit register renaming
    - In-order issue, out-of-order exec, out-of-order completion
  - Speculation
    - Guess branch outcome to find more ILP
    - Incorrect guesses must be cancelled
  - Speculative Tomasulo algorithm
    - In-order issue, out-of-order exec, <u>in-order</u> completion
    - Uses ROB

Midterm Review

- Multiple-issue processors
  - Multiple pipelines, issue more than 1 instr/CC, want CPI < 1
  - Superscalar: issue more than one instruction per CC whenever possible
    - Static (in-order) or dynamic (ooo) scheduling
  - VLIW: static issue; compiler packs independent instructions into very large CPU instructions

Midterm Review

- Memory hierarchy
  - Average memory access times
  - Single level cache
    - Types of cache misses: 3 C's
    - Unified versus split
    - Associativity: direct, set, full
    - Policies: write-through/back, (no-)write-allocate, FIFO/LRU/rand
  - Multi-level caches
    - Local miss rate, global miss rate
  - How to split the memory addresses into fields to address the various levels of cache
  - Virtual memory