Advanced Caching Techniques

Patterson

Review of Cache Basics

Assummed you already know answers to the following:-

- Why do we need caches and what's their goal?
- What's the basic idea of a cache and why does it work?
- How do we find something in a cache?
- What happens on a cache miss?
- What happens on a cache write?

(Revise from your Digital Electronics and Computer Organization Course material of last semester)

Review contd……Tuning Basic Cache Parameters: **S**ize, **A**ssociativity, **B**lock width

- Size:

– Must be large enough to fit working set (temporal locality)

– If too big, then hit time degrades

- Associativity

– Need large to avoid conflicts, but 4-8 way is as good as FA

– If too big, then hit time degrades

- Block

– Need large to exploit spatial locality & reduce tag overhead

– If too large, few blocks higher misses & miss penalty

Types of Misses (the 3 Cs)

- Compulsory or cold misses

–First access to an address within a program

(Misses even with an infinite sized cache)

- Capacity misses

– Misses because cache not large enough to fit working set

(Block replaced from cache and later accessed again)

- Conflict misses

– Misses due to associativity(in set-associative or direct E.g. two addresses map to same block in direct mapped cache)

Improving Cache Performance

- Goal: reduce the Average Memory Access Time
  - AvMemAccessTime = Hit Time + Miss Rate * Miss Penalty
- Approaches
  - Reduce Hit Time
  - Reduce Miss Penalty
  - Reduce Miss Rate

Understanding Cache Performance

–Average memory access time

AvMemAccessTime = Hit Time + Miss Rate * Miss Penalty

- What is MissRate?
  - Number of cache miss/Total number of memory reference
- What is Miss Penalty?
  - Memory access time /seconds per clock tick

  =memory access time given in number of clocks

Reducing Miss Rate

- Techniques we have seen so far

– **1. Larger caches**

- Reduces capacity misses

– **2.Higher associativity**

- Reduces conflict misses

– **3. Larger block sizes**

- Reduces cold misses

Reducing Miss Rate

- 4.**Using Victim Cache**
  - A small fully associative cache added between cache and its refill path. This victim cache contains only recent discarded lines from cache because of miss.This reduces conflict misses
- **5.Hardware Prefetching**
  - of next instruction along with the required instruction and keeping it in a buffer. **Overlaps execution and prefetching** of data as well.

Reducing Miss Rate contd..

- 6. **Compiler controlled Prefetch**
  - Compiler inserts prefetch instructions to request data before they are needed
    - **Register** prefetch loads value into register
    - **Cache** prefetch loads data into cache
      - Cache *continues to supply instruction and data to processor while waiting for prefetched data to come- called Non-blocking cache or lockup-free cache*

Reducing Miss Rate

- 7. Compiler Optimization
  - Rearrange code lines without affecting correctness but reducing miss rate

Reducing Miss Penalty

(Accessing memory in less number of clocks ticks)

- 1. **Give priority to read misses over writes**
  - Store writes in Write Buffers
  - After younger loads(reads) memory written back from buffers.(Extra hardware required)
- 2.**Sub-block replacement**
  - If large blocks are used in cache ,miss penalty increases(time to transfer block from memory).
  - By adding a *valid bit to units smaller than the block called sub-block*- only a sub-block need to be *transferred on a miss*.(Extra hardware needed)

Reducing Miss Penalty
(Accessing memory in less number of clocks ticks)

- 3. **Early start and critical word first**
  - CPU needs one word at a time, first word is sent to CPU starting it, then the rest of the block is loaded to cache
  - Two strategies:-
    - 1) ***Early start*** – as soon a ***requested word*** of block arrives, ***send it to CPU and let CPU continue execution***
    - 2) *Critical Word First* (also called wrapped fetch or requested word first)- *request the missed word first* from memory and *send it to CPU* and CPU continue execution while filling up the rest of the words of the block.(useful when block-size is very large

Reducing Miss Penalty
(Accessing memory in less number of clocks ticks)

- 4. **Non Blocking Caches:-**
  - In pipelined machines CPU continues to fetch instructions from instruction cache while waiting for data cache to return the missing data or *allowing to access next data* which is a hit *after <u>one miss</u>*(hit under one miss)

  (very complex cache circuit)

Reducing Miss Penalty
(Accessing memory in less number of clocks ticks)

- 5. **Second Level Caches:-**
  - AverageMemory access time =Hit timeL1+Miss rate L1 x Miss Panalty L1
  - Miss Panalty L1 =Hit time L2 + Miss rate L2 x Miss penalty L2
  - AverageMemory access time =Hit timeL1+Miss rate L1 x (Hit time L2 + Miss rate L2 x Miss penalty L2)
- **Note that only a fraction of memory access that leave CPU go all the way to memory**

Definitions:

– *Local miss rate*— misses in this cache divided by the total number of memory accesses *to this cache* (Miss rateL2)

– *Global miss rate*—misses in this cache divided by the total number of memory accesses *generated by the CPU* (Miss RateL1 x Miss RateL2)