Credits: Dr. Park (Virginia Tech), Dr. Patterson (UCB)

CpE 313: Microprocessor Systems Design

Handout 08
Measuring Cache Performance

October 07, 2004

Shoukat Ali

shoukat@umr.edu

UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

Where Are We?

finished discussing Q1 and Q2

will divert to cache performance for the moment

will deal with Q3 and Q4 later

- Q1: On a miss, when a new block is brought from memory, <u>where can the block be placed in the cache</u>? (Block placement)
- Q2: On a cache access, how does the HW know if the requested block is in the cache? (Block identification)
- Q3: On a miss, which block should be replaced to make room for the new block? (Block replacement)
- Q4: What happens on a write? (Write strategy)

More Terminology

- Average memory-access time (AMAT)
  = Hit time + Miss rate x Miss penalty (ns or clocks)
- Like miss rate, AMAT is an indirect measure of performance
- CPU time is a direct measure and is always the best

3

Revising the CPU Time Equation

- CPU Time = CPU Clock Cycles for a Program * Clock Cycle Time
- CPU clock cycles for a program can be divided into:
  - cycles CPU spends in executing
  - cycles CPU spends waiting for the memory system
- CPU time = (CPU execution cycles + Memory-stall cycles) x cycle time
- Formula for memory stall cycles:
  total # of memory stall cycles = total # of memory accesses x miss rate x miss penalty
- notes
  - memory stalls due to cache misses dominate other reasons for stalls (e.g., contention due to I/O devices using memory)
  - cycles for a cache hit are usually considered as part of the CPU execution cycles

4

Revising the CPU Time Equation – Cont'd

- another form
  - CPU Time = IC * CPI * Clock Cycle Time
- split CPI into CPI execution and CPI memory stall

$$\text{CPU time} = \text{IC} \times \underbrace{\left(\text{CPI}_{\text{exec}} + \frac{\text{memory accesses}}{\text{instruction}} \times \text{miss rate} \times \text{miss penalty} \right)}_{\text{CPI}} \times \text{cycle time}$$

5

Example: Impact of a Cache on the CPU Time

- Assume:
  - Miss penalty is 50 cycles
  - All instructions normally take 2 cycles (ignoring memory stalls)
  - Miss rate is 2%
  - Average of 1.33 memory accesses per instruction
- What is the CPU time w/o a cache?
  - No cache or 100% misses in a cache
  - CPI = 2 + 1.33 x 100% x 50 = 68.5
  - CPU time = IC x 68.5 x cycle time
- What is the CPU time w/ a "perfect cache" (no misses)?
  - CPI = 2 + 1.33 x 0% x 50 = 2
  - CPU time = IC x 2 x cycle time

6

Example: Impact of a Cache on the CPU Time

- What is the CPU time with a cache miss rate of 2%?
  - CPI = 2 + 1.33 x 2% x 50 = 3.33
  - CPU time = IC x 3.33 x cycle time
- w/o cache vs. w/ cache:

  68.5 >> 3.33
- Conclusion: Cache behavior can have enormous impact on performance!

7

Computing AMAT: DMC Versus 2-Way

- If a direct mapped cache has a miss rate of 5%, a hit time of 4 ns, and a miss penalty of 100 ns, what is the AMAT?

  AMAT = Hit time + Miss rate x Miss penalty
  = 4 + 0.05 x 100 = 9 ns
- If replacing the above cache with a 2-way set associative cache decreases the miss rate to 3%, but increases the hit time to 5 ns, what is the new AMAT? (Assume that the miss penalty stays the same.)

  AMAT = Hit time + Miss rate x Miss penalty
  = 5 + 0.03 x 100 = 8 ns
- so the 2-way cache has a lower AMAT

8

Effect of Associativity on AMAT

AMAT (in cycles) for a D-cache system on a DECstation 5000

Cache Size	Associativity			
(KB)	1-way	2-way	4-way	8-way
1	7.65	6.60	6.22	5.44
2	5.90	4.90	4.62	4.09
4	4.60	3.95	3.57	3.19
8	3.30	3.00	2.87	2.59
16	2.45	2.20	2.12	2.04
32	2.00	1.80	1.77	1.79
64	1.70	1.60	1.57	1.59
128	1.50	1.45	1.42	1.44

Numbers in red indicate that higher associativity resulted in increased AMAT.

AMAT = Hit time + Miss rate x Miss penalty

This is due to the fact that higher associativity results in increased hit time
→ speed of CPU is tied directly to the speed of a cache hit
→ increased clock cycle

what about PowerPC from IBM?

9

Computing CPU Time: DMC Versus 2-Way

- Does increasing associativity always result in better performance?
- Assume that:
  - One cache is direct mapped and the other is 2-way associative
  - CPI with a perfect cache is 2
  - Clock cycle time is 2 ns
  - There are 1.3 memory references per instruction
  - CPU clock cycle time must be stretched 1.1 times to accommodate the selection MUX of the set-associative cache
  - Miss penalty is 70 ns for either cache organization
  - Hit time is 1 clock cycle
  - Miss rates:
    - Direct mapped – 1.4%
    - Two-way set associative – 1.0%

10

Computing CPU Time: DMC Versus 2-Way

- Compute AMAT
  - $\text{AMAT}_{1\text{-way}} = 2 + (0.014 \times 70) = 2.98$ ns
  - $\text{AMAT}_{2\text{-way}} = 2 \times 1.1 + (0.010 \times 70) = 2.90$ ns
- CPU time formula

$$\text{CPU time} = \text{IC} \times \left(\text{CPI}_{\text{exec}} + \frac{\text{memory accesses}}{\text{instruction}} \times \text{miss rate} \times \text{miss penalty} \right) \times \text{cycle time}$$

$$= \text{IC} \times \left[\left(\text{CPI}_{\text{exec}} \times \text{cycle time} \right) + \left(\frac{\text{memory accesses}}{\text{instruction}} \times \text{miss rate} \times \text{miss penalty} \times \text{cycle time} \right) \right]$$

(70 ns = miss penalty x cycle time)

- $\text{CPU time}_{1\text{-way}} = \text{IC} \times (2 \times 2 + (1.3 \times 0.014 \times 70)) = 5.27 \times \text{IC}$
- $\text{CPU time}_{2\text{-way}} = \text{IC} \times (2 \times 2 \times 1.1 + (1.3 \times 0.010 \times 70))$
  $= 5.31 \times \text{IC}$

11

Computing CPU Time: DMC Versus 2-Way

- Conclusion from example
  - AMAT is better for the 2-way set associative cache
  - However, direct mapped cache leads to slightly better average performance
    - This is because of longer hit time → longer clock cycle
  - In this example, the preferred cache is direct mapped because
    - CPU time is the bottom-line evaluation, and
    - Direct mapped is simpler to build

12

Split Vs. Unified Cache

- Unified cache (mixed cache): Data and instructions are stored together (von Neuman architecture)
- Split cache: Data and instructions are stored separately (Harvard architecture)

Miss rates for instruction, data, and unified caches on the DECstation 5000

Size	Instruction Cache	Data Cache	Unified Cache
1 KB	3.06%	24.61%	13.34%
2 KB	2.26%	20.57%	9.78%
4 KB	1.78%	15.94%	7.24%
8 KB	1.10%	10.19%	4.57%
16 KB	0.64%	6.47%	2.87%
32 KB	0.39%	4.82%	1.99%
64 KB	0.15%	3.77%	1.35%
128 KB	0.02%	2.88%	0.95%

13

Split Vs. Unified Cache: Performance Comparison

- Assumptions
  - Split cache: 16 KB instructions + 16 KB data
  - Unified cache: 32 KB (instructions + data)
  - Use miss rates from previous chart
  - Miss penalty is 50 cycles
  - Hit time is 1 cycle
  - On the unified cache, a load or store hit takes an extra cycle for data access, since there is only one cache port for instructions and data
- Which one has the lower miss rate?
  - 32-KB unified cache has miss rate of 1.99% (from table)
  - Overall miss rate for split cache
    - assume 33% of instructions access memory
      - inst cache refs = 1/1.33 = 0.75 --- data cache refs = 0.33/1.33
    - 0.75 x 0.64% + 25% x 6.47% = 2.10%
  - Unified cache has lower miss rate

14

Split Vs. Unified Cache: Performance Comparison

- Which one has a lower AMAT?

  Average memory-access time (AMAT) = Hit time + Miss rate x Miss penalty

AMAT = %instr x (instr hit time + instr miss rate x instr miss penalty) +
%data x (data hit time + data miss rate x data miss penalty)

For the split cache:

AMAT = 75% x (1 + 0.64% x 50) + 25% x (1 + 6.47% x 50) = 2.05 cycles

For the unified cache

AMAT = 75% x (1 + 1.99% x 50) + 25% x (***2*** + 1.99% x 50) = 2.24 cycles

The unified cache has a longer AMAT, even though its miss rate is lower, due to conflicts for instruction and data (single cache port)

15

Lecture Conclusions

- go with
  - split caches
  - DMC or low associativity caches unless you can develop ways of not degrading hit time with high associativity caches

16