BCT 2408 Computer Architecture-II: Lab 4 Solution

Name: Jeff Gicharu
Reg No: SCT212-0053/2021

This document provides the solutions to the problems presented in Lab 4, focusing on quantitative cache analysis and the tradeoffs between write-through and write-back policies.

E1: Cache Parameter Calculation and Performance Impact (individual 10 min)

Problem

Assume we have a computer where the CPI is 1.0 when all memory accesses (including data and instruction accesses) hit in the cache. The cache is a unified (data + instruction) cache of size 256 KB, 4-way set associative, with a block size of 64 bytes. The data accesses (loads and stores) constitute 50% of the instructions. The unified cache has a miss penalty of 25 clock cycles and a miss rate of 2%. Assume 32 bit instruction and data addresses.

(a) What is the tag size for the cache?

(b) How much faster would the computer be if all memory accesses were cache hits?

- **Answer (a): Tag Size Calculation**
  1. **Block Offset:** The block size is 64 bytes. The number of bits needed for the block offset is log2(64)=6 bits.
  2. Number of Sets: The cache size is 256 KB (218 bytes). The associativity is 4-way. The number of sets is calculated as:
     Number of Sets = Cache Size / (Block Size * Associativity)
     Number of Sets = 256 KB / (64 Bytes * 4) = 262144 / 256 = 1024 sets (210 sets).
  3. **Index Bits:** The number of bits needed to index these sets is log2(1024)=10 bits.
  4. Tag Bits: The total address size is 32 bits. The tag size is the remaining bits after accounting for index and offset:
     Tag Bits = Total Address Bits - Index Bits - Offset Bits
     Tag Bits = 32 - 10 - 6 = 16 bits.

  *Therefore, the tag size for the cache is **16 bits**.*
- **Answer (b): Speedup Calculation**

1. **Ideal CPI:** The CPI when all memory accesses hit is given as CPI_ideal = 1.0.
2. **Memory Accesses per Instruction:** Each instruction requires 1 instruction fetch access. Additionally, 50% of instructions are data accesses (loads/stores). So, the average number of memory accesses per instruction is 1 + 0.5 = 1.5.
3. Stall Cycles per Instruction due to Misses: The stall cycles are calculated as:
   Stall Cycles = Memory Accesses per Instruction * Miss Rate * Miss Penalty
   Stall Cycles = 1.5 * 0.02 * 25 = 0.75 cycles per instruction.
4. Actual CPI: The actual CPI, considering cache misses, is:
   CPI_actual = CPI_ideal + Stall Cycles per Instruction
   CPI_actual = 1.0 + 0.75 = 1.75.
5. Speedup: The speedup of the ideal computer (no misses) compared to the actual computer is the ratio of their CPIs (since IC and Clock Time are the same):
   Speedup = CPI_actual / CPI_ideal = 1.75 / 1.0 = 1.75.

*Therefore, the computer would be **1.75 times faster** if all memory accesses were cache hits.*

E2: Cache Bandwidth Usage (groups of 2 - 15 min)

Problem

You purchased an Acme computer with the following features:

- 95% of all memory accesses are found in the cache (Hit Rate = 0.95, Miss Rate = 0.05).
- Each cache block is two words, and the whole block is read on any miss.
- The processor sends references to its cache at the rate of 109 words per second.
- 25% of those references are writes.
- Assume that the memory system can support 109 words per second, reads or writes.
- The bus reads or writes a single word at a time.
- Assume at any one time, 30% of the blocks in the cache have been modified (dirty).
- The cache uses write allocate on a write miss.

You are considering adding a peripheral to the system, and you want to know how much of the memory system bandwidth is already used. Calculate the percentage of memory system bandwidth used on the average in the two cases below. Be sure to state your assumptions.

(a) The cache is write through.
(b) The cache is write back.

- **Calculations & Assumptions:**
  - Processor Reference Rate = 109 words/sec
  - Memory System Bandwidth = 109 words/sec
  - Fraction of Reads = 1−0.25=0.75
  - Fraction of Writes = 0.25
  - Hit Rate = 0.95, Miss Rate = 0.05
  - Block Size = 2 words
  - Bus transfers 1 word at a time.
  - Dirty Block Fraction (for Write Back) = 0.30
  - Clean Block Fraction (for Write Back) = 1−0.30=0.70
  - Cache uses Write Allocate.
- **Answer (a): Write-Through Cache Bandwidth Usage**
  We calculate the average number of words transferred between cache and memory per processor reference:
  - **Read Hit (0.75 * 0.95):** 0 words transferred.
  - **Read Miss (0.75 * 0.05):** 2 words transferred (fetch block from memory).
  - **Write Hit (0.25 * 0.95):** 1 word transferred (write data through to memory).
  - **Write Miss (0.25 * 0.05):** 3 words transferred (fetch block due to write allocate = 2 words, then write data through to memory = 1 word).

  Avg Words Transferred = (0.75 * 0.95 * 0) + (0.75 * 0.05 * 2) + (0.25 * 0.95 * 1) + (0.25 * 0.05 * 3)Avg Words Transferred = 0 + 0.075 + 0.2375 + 0.0375 = 0.35 words per processor reference.Bandwidth Used = Avg Words Transferred * Processor Reference RateBandwidth Used = 0.35 * 10^9 words/sec.Percentage Bandwidth Used = (Bandwidth Used / Memory System Bandwidth) * 100%Percentage Bandwidth Used = (0.35 * 10^9 / 10^9) * 100% = 35%.*With a write-through cache, **35%** of the memory system bandwidth is used.*
- **Answer (b): Write-Back Cache Bandwidth Usage**
  We calculate the average number of words transferred between cache and memory per processor reference:
  - **Read Hit (0.75 * 0.95):** 0 words transferred.
  - **Read Miss (0.75 * 0.05):**
    - If replaced block is clean (70%): 2 words transferred (fetch block).
    - If replaced block is dirty (30%): 4 words transferred (write back dirty block = 2 words, fetch new block = 2 words).
    - Avg words on read miss = (0.70 * 2) + (0.30 * 4) = 1.4 + 1.2 = 2.6 words.
  - **Write Hit (0.25 * 0.95):** 0 words transferred (write happens only in cache).

- **Write Miss (0.25 * 0.05):** (Uses Write Allocate)
    - If replaced block is clean (70%): 2 words transferred (fetch block).
    - If replaced block is dirty (30%): 4 words transferred (write back dirty block = 2 words, fetch new block = 2 words).
    - Avg words on write miss = (0.70 * 2) + (0.30 * 4) = 1.4 + 1.2 = 2.6 words.

Avg Words Transferred = (0.75 * 0.95 * 0) + (0.75 * 0.05 * 2.6) + (0.25 * 0.95 * 0) + (0.25 * 0.05 * 2.6)Avg Words Transferred = 0 + 0.0975 + 0 + 0.0325 = 0.13 words per processor reference.Bandwidth Used = Avg Words Transferred * Processor Reference RateBandwidth Used = 0.13 * 10^9 words/sec.Percentage Bandwidth Used = (Bandwidth Used / Memory System Bandwidth) * 100%Percentage Bandwidth Used = (0.13 * 10^9 / 10^9) * 100% = 13%.*With a write-back cache, **13%** of the memory system bandwidth is used*.

E3: Write Policy Performance Comparison (groups of 2 - 15 min)

Problem

One difference between a write-through cache and a write-back cache can be in the time it takes to write. During the first cycle, we detect whether a hit will occur, and during the second (assuming a hit) we actually write the data. Let's assume that 50% of the blocks are dirty for a write-back cache. For this question, assume that the write buffer for the write through will never stall the CPU (no penalty). Assume a cache read hit takes 1 clock cycle, the cache miss penalty is 50 clock cycles, and a block write from the cache to main memory takes 50 clock cycles. Finally, assume the instruction cache miss rate is 0.5% (MRI = 0.005) and the data cache miss rate is 1% (MRD = 0.01). Assuming that on average 26% and 9% of instructions in the workload are loads and stores, respectively, estimate the performance of a write-through cache with a two-cycle write versus a write-back cache with a two-cycle write.

- **Calculations & Assumptions:**
    - Read Hit Time = 1 cycle
    - Write Hit Time = 2 cycles
    - Miss Penalty (fetch) = 50 cycles
    - Write Back Penalty (dirty block) = 50 cycles
    - Instruction Fraction = 1.0 (100%)
    - Load Fraction = 0.26
    - Store Fraction = 0.09
    - Other Instruction Fraction = 1.0–0.26–0.09=0.65
    - MRI = 0.005
    - MRD = 0.01

    - Dirty Block Fraction (for WB) = 0.50
    - Clean Block Fraction (for WB) = 0.50
    - Write-Through buffer never stalls (no extra penalty on write hits).
- Base CPI (Execution CPI without memory stalls):
  CPI_execution = (Load Fraction * Read Hit Time) + (Store Fraction * Write Hit Time) + (Other Fraction * 1)
  CPI_execution = (0.26 * 1) + (0.09 * 2) + (0.65 * 1)
  CPI_execution = 0.26 + 0.18 + 0.65 = 1.09
- **Answer (a): Write-Through Cache Performance**
  Calculate memory stall cycles per instruction:
    - Instruction Miss Stalls: Instruction Fraction * MRI * Miss Penalty
      1.0 * 0.005 * 50 = 0.25
    - Data Miss Stalls (Loads): Load Fraction * MRD * Miss Penalty
      0.26 * 0.01 * 50 = 0.13
    - Data Miss Stalls (Stores): Store Fraction * MRD * Miss Penalty (Write-allocate fetches block, write goes to buffer with no penalty)
      0.09 * 0.01 * 50 = 0.045
    - **Total Stall Cycles:** 0.25 + 0.13 + 0.045 = 0.425

  CPI_WriteThrough = CPI_execution + Total Stall CyclesCPI_WriteThrough = 1.09 + 0.425 = 1.515
- **Answer (b): Write-Back Cache Performance**
  Calculate memory stall cycles per instruction:
    - Instruction Miss Stalls: Instruction Fraction * MRI * Miss Penalty
      1.0 * 0.005 * 50 = 0.25
    - **Data Miss Stalls (Loads):** Load Fraction * MRD * Effective Miss Penalty
        - Effective Miss Penalty = (Clean Replace * Fetch Penalty) + (Dirty Replace * (Write Back Penalty + Fetch Penalty))
        - Effective Miss Penalty = (0.50 * 50) + (0.50 * (50 + 50)) = 25 + 50 = 75 cycles
        - Load Stalls = 0.26 * 0.01 * 75 = 0.195
    - **Data Miss Stalls (Stores):** Store Fraction * MRD * Effective Miss Penalty (Write-allocate uses same penalty as load miss)
        - Store Stalls = 0.09 * 0.01 * 75 = 0.0675
    - **Total Stall Cycles:** 0.25 + 0.195 + 0.0675 = 0.5125

  CPI_WriteBack = CPI_execution + Total Stall CyclesCPI_WriteBack = 1.09 + 0.5125 = 1.6025
- Performance Comparison:

The CPI for the Write-Through cache (1.515) is lower than the CPI for the Write-Back cache (1.6025) under these specific assumptions (mainly, the write buffer never stalling for WT, and the WB needing 50 cycles to write back a block). Therefore, the Write-Through cache system is estimated to be faster in this scenario.
Speedup_WT_vs_WB = CPI_WriteBack / CPI_WriteThrough = 1.6025 / 1.515 ≈ 1.058 (WT is about 5.8% faster).