Lab 4

SCT212-0051/2021 MAKUMI BEDAN NGUGI

EXAMPLE 1

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.

- **Base CPI (all hits) = 1.0**
- **Cache Size = 256 KB = 2^18 Bytes**
- **Associativity = 4-way**
- **Block Size = 64 Bytes = 2^6 Bytes**
- **Address Size = 32 bits**
- **Data Accesses (Loads + Stores) = 50% of instructions**
- **Miss Penalty = 25 clock cycles**
- **Unified Cache Miss Rate = 2% (0.02)**

a. What is the tag size for the cache?

Num Blocks = C / B = 256 KB / 64 B = (2^18 B) / (2^6 B) = 2^12 blocks

Num Sets (S) = Num Blocks / Associativity = 2^12 / 4 = 2^12 / 2^2 = 2^10 sets

Index Bits = log2(Num Sets) = log2(2^10) = 10 bits

Offset Bits = log2(Block Size) = log2(64) = log2(2^6) = 6 bits

Tag Bits = Address Size - Index Bits - Offset Bits

Tag Bits = 32 - 10 - 6 = 16 bits

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

1. Calculate Effective CPI with Cache Misses:

CPI_effective = CPI_base + (Memory Stall Cycles per Instruction)

Memory Accesses per Instruction:

- Every instruction involves an instruction fetch (1 access).
- 50% of instructions are data accesses (0.5 accesses).
- Total accesses per instruction = 1 + 0.5 = 1.5

Memory Stall Cycles per Instruction = (Accesses per Instruction) * (Miss Rate) * (Miss Penalty)

Memory Stall Cycles = 1.5 * 0.02 * 25 cycles = 0.75 cycles per instruction

CPI_effective = 1.0 + 0.75 = 1.75

2. Calculate Execution Time Ratio (Speedup):

Speedup = Execution Time (All Hits) / Execution Time (With Misses)

Execution Time = Instruction Count * CPI * Clock Cycle Time

Since IC and Clock Cycle Time are the same in both scenarios, the ratio simplifies:

Speedup = (IC * CPI_base * CCT) / (IC * CPI_effective * CCT)

Speedup = CPI_base / CPI_effective

Speedup = 1.0 / 1.75 ≈ 0.5714

3. Interpret Speedup:

- Faster = Execution Time (With Misses) / Execution Time (All Hits)
- Faster = CPI_effective / CPI_base
- Faster = 1.75 / 1.0 = 1.75

EXAMPLE 2

Problem

You purchased an Acme computer with the following features:

- 95% of all memory accesses are found in the cache.
- 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 (the memory system cannot read or
- write two words at once).
- Assume at any one time, 30% of the blocks in the cache have been modified.
- 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.

- **Cache Hit Rate = 95% (=> Miss Rate = 5% or 0.05)**
- **Cache Block Size = 2 words**

- **Processor Reference Rate = 10^9 words/sec**
- **Write References = 25% of total references (=> Read References = 75%)**
- **Memory System Bandwidth = 10^9 words/sec (peak)**
- **Bus Transfer Size = 1 word at a time**
- **Modified (Dirty) Blocks = 30%**
- **Write Policy: Write Allocate on write miss**

Assumptions:

- **Word size matches bus transfer size.**
- **Address transfers don't consume significant bandwidth compared to data transfers.**
- **Write-Through cache has a write buffer that perfectly hides stall times (as often assumed in simple analyses), but still consumes bandwidth.**

Calculations

Total References: 10^9 words/sec

Write References: 0.25 * 10^9 words/sec

Read References: 0.75 * 10^9 words/sec

Total Misses: Miss Rate * Total References = 0.05 * 10^9 words/sec

Read Misses: Miss Rate * Read References = 0.05 * (0.75 * 10^9) = 0.0375 * 10^9 words/sec

Write Misses: Miss Rate * Write References = 0.05 * (0.25 * 10^9) = 0.0125 * 10^9 words/sec

a. The cache is write through.

- **Memory Traffic Sources:**
  1. **Read Misses:** Fetch the entire block (2 words). Since the bus transfers 1 word at a time, each read miss causes 2 word transfers from memory.
  2. **All Writes (Hits and Misses):** Every write goes through to memory. Each write reference causes 1 word transfer to memory.
- **Bandwidth Calculation:**
  - Read Miss Traffic = Read Misses * Words per Block = (0.0375 * 10^9 /sec) * 2 words = 0.075 * 10^9 words/sec
  - Write Traffic = Total Write References * 1 word = (0.25 * 10^9 /sec) * 1 word = 0.25 * 10^9 words/sec
  - Total Traffic (WT) = Read Miss Traffic + Write Traffic = (0.075 + 0.25) * 10^9 = 0.325 * 10^9 words/sec
- **Percentage Bandwidth Used:**
  - % BW = (Total Traffic / Memory Bandwidth) * 100

- % BW = (0.325 * 10^9 / 10^9) * 100 = **32.5%** of the memory system bandwidth is used with a write-through cache

b. The cache is write back.

- **Memory Traffic Sources:**
    1. **Read Misses:** Fetch the entire block (2 words). Causes 2 word transfers from memory.
    2. **Write Misses (Write Allocate):** Must fetch the block first (2 words from memory), then the write modifies it in the cache.
    3. **Write Backs on Replacement:** When a miss occurs and the block being replaced is dirty (modified), the dirty block must be written back to memory (2 words to memory). This happens on *some* fraction of misses (both read and write).
- **Bandwidth Calculation:**
    - Read Miss Fetch Traffic = Read Misses * Words per Block = (0.0375 * 10^9 /sec) * 2 words = 0.075 * 10^9 words/sec
    - Write Miss Fetch Traffic = Write Misses * Words per Block = (0.0125 * 10^9 /sec) * 2 words = 0.025 * 10^9 words/sec
    - Write Back Traffic: This happens only when a miss occurs *and* the block being replaced is dirty. The probability of replacing a dirty block is approximated by the fraction of dirty blocks (30%).
        1. Write Backs = Total Misses * Fraction Dirty * Words per Block
        2. Write Backs = (0.05 * 10^9 /sec) * 0.30 * 2 words = 0.03 * 10^9 words/sec
    - Total Traffic (WB) = Read Miss Fetch + Write Miss Fetch + Write Backs
    - Total Traffic (WB) = (0.075 + 0.025 + 0.03) * 10^9 = 0.13 * 10^9 words/sec
- **Percentage Bandwidth Used:**

    % BW = (Total Traffic / Memory Bandwidth) * 100

    % BW = (0.13 * 10^9 / 10^9) * 100 = **13.0%** of the memory system bandwidth is used with a write-back cache.

EXAMPLE 3

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% and the data cache miss rate is 1%. 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.

- **Write Hit Time = 2 cycles (for both policies)**
- **Read Hit Time = 1 cycle**
- **Cache Miss Penalty (fetch block from memory) = 50 cycles**
- **Block Write Back Time (WB only) = 50 cycles**
- **Instruction Cache Miss Rate = 0.5% (0.005)**
- **Data Cache Miss Rate = 1% (0.01)**
- **Loads = 26% (0.26) of instructions**
- **Stores = 9% (0.09) of instructions**
- **Write Buffer for WT: Assume never stalls (no penalty for write-through writes beyond the 2 cycles).**
- **Write Back Cache: 50% (0.50) of blocks are dirty.**

Formula: CPI_effective = CPI_base + CPI_ReadStalls + CPI_WriteStalls

1. **Calculate Base CPI considering 2-cycle writes:**

   - CPI = (Fraction Read Hits * 1 cycle) + (Fraction Write Hits * 2 cycles) + CPI_MissStalls
   - We need the fraction of accesses that are reads vs writes.
   - Total Accesses per Instruction = 1 (Inst Fetch) + 0.26 (Loads) + 0.09 (Stores) = 1.35
   - Fraction Reads = (1 + 0.26) / 1.35 = 1.26 / 1.35 ≈ 0.933
   - Fraction Writes = 0.09 / 1.35 ≈ 0.067
   - CPI_base_adjusted (assuming all hits) = (0.933 * 1) + (0.067 * 2) = 0.933 + 0.134 = 1.067
   - Let's refine this: Consider the instruction mix directly.
   - CPI = (Fraction non-mem * 1) + (Fraction loads * CPI_load) + (Fraction stores * CPI_store)
   - Fraction non-mem = 1 - 0.26 - 0.09 = 0.65
   - Assume ideal CPI=1. The extra cycle for write *hits* adds to the total cycles.
   - Cycles = IC * [(CPI_base=1) + (Extra Write Hit Cycles per Inst) + (Read Miss Stall Cycles per Inst) + (Write Miss Stall Cycles per Inst)]
   - CPI_eff = 1 + ExtraWriteHitCyclePenalty + ReadMissPenalty + WriteMissPenalty

2. **Calculate Access Frequencies per Instruction:**

- Instruction Fetches = 1 per instruction
- Data Reads (Loads) = 0.26 per instruction
- Data Writes (Stores) = 0.09 per instruction

3. **Calculate Miss Stalls per Instruction:**

    - Read Miss Stalls = (Inst Fetch Misses * Penalty) + (Load Misses * Penalty)
    - Read Miss Stalls = (1 * Inst Miss Rate * Penalty) + (0.26 * Data Miss Rate * Penalty)
    - Read Miss Stalls = (1 * 0.005 * 50) + (0.26 * 0.01 * 50)
    - Read Miss Stalls = 0.25 + 0.13 = 0.38 cycles/instruction

Write-Through (WT) Performance:

- **Extra Write Hit Cycle Penalty:** Write hits take 2 cycles instead of 1. Penalty = 1 cycle.
    - Penalty = (Data Writes per Inst) * (Hit Rate) * (1 extra cycle)
    - Penalty = 0.09 * (1 - 0.01) * 1 = 0.09 * 0.99 * 1 ≈ 0.089 cycles/instruction
- **Write Miss Stall Penalty:** A write miss requires fetching the block. Write buffer handles the write itself without stalls.
    - Penalty = (Data Writes per Inst) * (Miss Rate) * (Miss Penalty for fetch)
    - Penalty = 0.09 * 0.01 * 50 = 0.045 cycles/instruction
- **Total CPI (WT):**
    - CPI_WT = (Base CPI = 1) + ExtraWriteHitCyclePenalty + ReadMissStalls + WriteMissStalls
    - CPI_WT = 1 + 0.089 + 0.38 + 0.045 = **1.514**

Write-Back (WB) Performance:

- **Extra Write Hit Cycle Penalty:** Same as WT for hits.
    - Penalty = 0.09 * (1 - 0.01) * 1 ≈ 0.089 cycles/instruction
- **Write Miss Stall Penalty:** A write miss involves fetching the block (Write Allocate). Sometimes, it also involves writing back the dirty block being replaced.
    - Fetch Penalty = (Data Writes per Inst) * (Miss Rate) * (Miss Penalty for fetch)
    - Fetch Penalty = 0.09 * 0.01 * 50 = 0.045 cycles/instruction
    - Write Back Penalty = (Total Misses causing replacement) * (Fraction Dirty) * (Write Back Time)
    - Misses causing replacement = (Inst Misses) + (Load Misses) + (Write Misses)

    - Misses per Inst = (1 * 0.005) + (0.26 * 0.01) + (0.09 * 0.01) = 0.005 + 0.0026 + 0.0009 = 0.0085
    - Write Back Penalty per Inst = 0.0085 * 0.50 * 50 = 0.2125 cycles/instruction
    - Total Write Miss Penalty = Fetch Penalty + Write Back Penalty = 0.045 + 0.2125 = 0.2575 cycles/instruction

- **Total CPI (WB):**
    - CPI_WB = (Base CPI = 1) + ExtraWriteHitCyclePenalty + ReadMissStalls + TotalWriteMissPenalty
    - CPI_WB = 1 + 0.089 + 0.38 + 0.2575 = **1.7265**

The **Write-Through cache has better performance (lower CPI)** in this specific scenario (CPI ≈ 1.51) compared to the Write-Back cache (CPI ≈ 1.73). The cost of writing back dirty blocks on misses in the Write-Back cache outweighs the bandwidth savings on writes.