Desmond Oketch- SCT212-0083/2021

COMPUTER TECHNOLOGY

Lab 5

Computer Architecture – tutorial 5

Problem Summary

Analyse the data cache behaviour for the given code:

```
for (i = 0; i < 4096; i++)
   X[i] = X[i] * Y[i] + C;
```

Arrays:

- X[4096] and Y[4096] → 4096 elements of 4 bytes = 16 KB each.

Cache:

- Size: 16 KB
- Line size: 64 Bytes
- Associativity: Direct-mapped
- Addressing: Physical

Compute total cache lines:

Cache lines=16 KB64 B=1638464=256 lines\text {Cache lines} = \frac{16 \text{ KB}}{64 \text{ B}} = \frac{16384}{64} = 256 \text{ lines}Cache lines=64 B16 KB=6416384 =256 lines

Part a: Cache Miss Analysis

Memory Access Pattern

Each iteration:

1. Load X[i] → 4 bytes
2. Load Y[i] → 4 bytes
3. Store X[i]

Total: 3 memory operations per iteration.

Each cache line holds:

644=16 elements\frac{64}{4} = 16 \text{ elements}464=16 elements

So, accessing X[i] will bring X[i] to X[i+15] into cache.

Compulsory Misses (cold misses)

Each unique cache line accessed for the first time causes a compulsory miss.

X accesses:

- 4096 elements / 16 per line = 256 unique cache lines
- So, 256 compulsory misses.

Y accesses:

- 4096 elements / 16 per line = 256 unique lines

Conflict Misses

The cache is direct-mapped and X & Y are stored consecutively, they may map to the same cache lines.

- Let’s assume X starts at address 0.
- Since Y is allocated right after X, it will map to the same cache lines as X.

For every Y access, it evicts X from the cache.

Thus:

- When you load X[i], it’s a miss (first time: compulsory, later: conflict).
- Then you load Y[i] → also a miss, replaces X[i].
- Then you store to X[i] → miss again, because X[i] was evicted by Y[i].

So each iteration causes 3 cache misses:

- 1 for X[i] (load)
- 1 for Y[i]
- 1 for X[i] (store)

Total Misses

4096×3=12288 data cache misses4096 \times 3 = 12288 \text {data cache misses}4096×3=12288 data cache misses

Breakdown

- Compulsory Misses: 256 (X) + 256 (Y) = 512
- Conflict Misses: 12288 - 512 = 11776
- Capacity Misses: 0, because total working set = 32KB, but we're not accessing more than 2 elements at once.

Cache Miss Rate

Total memory accesses = 4096 × 3 = 12288

Miss rate = 1228812288=100%\frac{12288}{12288} = \boxed{100\%}1228812288=100%

Part b: Software Optimization

Reduce conflict misses.

Solution: Loop Interchange or Blocking, but simpler here:

Interleave accesses to avoid eviction:

Instead of accessing X then Y:

```
for (i = 0; i < 4096; i++) {
   tempX = X[i]; // load X
   tempY = Y[i]; // load Y
   tempX = tempX * tempY + C;
   X[i] = tempX; // store X
}
```

Better Software Fix: Loop Blocking

Use blocking with a block size that fits within the cache without causing conflicts.

```
int B = 64; // block size (or 128)
for (ii = 0; ii < 4096; ii += B) {
   for (i = ii; i < ii + B; i++) {
      X[i] = X[i] * Y[i] + C;
   }
}
```

Best fix: Padding arrays

Add padding between X and Y so they don’t map to same cache lines.

For example:

```
float X[4096];
float pad[256];  // adds 1KB of space
float Y[4096];
```

This makes sure that X and Y don’t conflict in cache.

Cache Misses Now:

- X accesses: 256 loads + 256 stores = 512 misses (compulsory only)
- Y accesses: 256 loads = 256 misses (compulsory)

Total = 768 cache misses

Breakdown:

- Compulsory Misses: 768
- Conflict Misses: 0 (due to padding)
- Capacity Misses: 0

Miss Rate

Total memory accesses = 12288

Miss rate = 76812288=6.25%\frac{768}{12288} = \boxed{6.25\%}12288768=6.25%

Part c: Hardware Optimization

Modify the hardware

Solution: Change Cache Organization

Change from direct-mapped → 2-way set associative or fully-associative.

2-way set associative cache:

- 16KB / 64B = 256 lines → 128 sets of 2 lines
- X and Y can now both map to the same set but occupy different ways → no conflict.

Cache Misses:

- First access of each block: compulsory
- No conflict misses (because associativity allows both to reside)

Same result as software padding.

Total:

- X: 256 load + 256 store = 512
- Y: 256 load = 256

→ Total = 768

Breakdown:

- Compulsory: 768
- Conflict: 0
- Capacity: 0

Miss Rate:

Miss rate = 76812288=6.25%\frac{768}{12288} = \boxed{6.25\%}12288768=6.25%