Cache Performance

Caiwen Ding
Department of Computer Science and Engineering
University of Connecticut

Adapted from *Computer Organization and Design*
by Patterson & Hennessy

Cache Performance

- Cache performance
- Improving cache performance
  - Set associative cache
  - Multi level cache

Reading: Sections 5.4, excluding the software approach.

Terms

Hit: data found in cache

Hit Rate: number of hits divided by number of memory accesses

Hit Time: Time to get data from cache on a hit

Miss: data NOT in cache

Miss Rate: number of Misses divided by number of memory accesses.

Miss Penalty: Additional time to get data from cache on a miss

Miss Rate = 1 – Hit Rate

Hit Time << Miss Penalty

Cache access time

- Hit Time: Time to access cache. Some steps can be performed in parallel
  - Select status bits/tag/data
  - Check hit or miss
  - Select the item within the block

- Miss Penalty: Time to replace a block in that level with the corresponding block from a lower level

access time on miss = hit time + miss penalty

Impact of Cache Performance

- Assuming cache hit costs are included as part of the normal CPU execution cycle, then

CPI overhead from memory stalls

CPU time

$= IC \times CPI \times CC$

$= IC \times \underbrace{(CPI_{perfect} + MemoryStallCycles/Instruction)}_{CPI_{stall}} \times CC$

$= (ExecutionClockCycles + MemoryStallCycles) \times CC$

IC: Instruction Count (number of instructions executed)

CC: Clock cycle time

$CPI_{prefect}$: CPI with perfect memory

Memory stalls cycles

- Memory-stall cycles come from both read and write

Read stalls

Read stall cycles

= reads/program × read miss rate × read miss penalty

Write stalls

- Write-through caches

Write stall cycles

= (writes/program × write miss rate × write miss penalty) + write buffer stalls

Write buffer stalls should be small in good designs

- Write-back caches
  - Potentially, additional stalls when replacing a dirty block

Memory stalls

We combine read and write into a single miss rate and miss penalty

MemoryStallCycles = accesses/program × miss rate × miss penalty

Often, we find out the average number of misses per instruction

MemoryStallCycles/instruction = misses/instruction × miss penalty

Then find the total memory stall cycles for a program

MemoryStallCycles = IC × misses/instruction × miss penalty

Or compute the CPU time

CPU time = IC × (CPI$_{\text{perfect}}$ + MemoryStallCycles/Instruction) × CC

Do not just memorize the equations.

Memory stalls - 2

- We have two caches: instruction cache and data cache
  - Every instruction has instruction cache access
  - Only load/store instructions have data cache access

misses/instruction

= I-Cache-Misses/instruction + D-Cache-Misses/instruction

Example: memory stall cycles per instruction

Assumptions: a 100 cycles miss penalty, 36% load/store instructions, and 2% I cache and 4% D cache miss rates.

What is MemoryStallCycles/Instruction?

Example: memory stall cycles per instruction

Assumptions: a 100 cycles miss penalty, 36% load/store instructions, and 2% I cache and 4% D cache miss rates.

What is MemoryStallCycles/Instruction?

MemoryStallCycles/Instruction

$= 0.02 \times 100 + 0.36 \times 0.04 \times 100 = 3.44$

Misses/instruction $= 0.02 + 0.36 \times 0.04 = 0.0344$

MemoryStallCycles/Instruction $= 0.0344 \times 100 = 3.44$

Explain each term.

Example: having a perfect memory

Assumptions: a 100 cycles miss penalty, 36% load/store instructions, and 2% I cache and 4% D cache miss rates.

If $CPI_{perfect}$ is 2, what is the speedup of having a perfect memory?

What is the percentage of CPU time on memory stalls?

MemoryStallCycles/Instruction = 3.44

$CPI_{stalls} = 2 + 3.44 = \mathbf{5.44}$

Example: having a perfect memory

Assumptions: a 100 cycles miss penalty, 36% load/store instructions, and 2% I cache and 4% D cache miss rates.

If $CPI_{perfect}$ is 2, what is the speedup of having a perfect memory?

What is the percentage of CPU time on memory stalls?

MemoryStallCycles/Instruction = 3.44

$CPI_{stalls} = 2 + 3.44 = \mathbf{5.44}$

The speedup of a perfect memory: 5.44 / 2 = 2.72

Percentage of time on memory stalls: 3.44 / 5.44 = 63%

Note that IC and CC in the calculation of CPU time are the same.

Question

Assumptions: a 100 cycles miss penalty, 36% load/store instructions, and 2% I cache and 4% D cache miss rates.

If $CPI_{perfect}$ is 1.5, what is the percentage of CPU time on memory stalls?

MemoryStallCycles/Instruction = 3.44

Round to the nearest tenth.

Question

Assumptions: a 100 cycles miss penalty, 36% load/store instructions, and 2% I cache and 5% D cache miss rates.

If $\text{CPI}_{\text{perfect}}$ is 2, what is the speedup of having a perfect memory?

What is the percentage of CPU time on memory stalls?

Example

Assumptions: a 100 cycles miss penalty, 36% load/store instructions, and 2% I cache and 4% D cache miss rates.

If $\text{CPI}_{\text{perfect}}$ is 2, what is the percentage of CPU time on memory stalls?

MemoryStallCycles/Instruction = $(0.02 + 0.36 \times 0.04) \times 100 = 3.44$

$\text{CPI}_{\text{stalls}} = 2 + 3.44 = \mathbf{5.44}$

Percentage of time on memory stalls: $3.44 / 5.44 = 63\%$

What if the memory remains the same but the processor's clock rate is doubled?

What numbers are changed?

Answer

The miss penalty is doubled.

MemoryStallCycles/Instruction

$= 0.02 \times 200 + 0.36 \times 0.04 \times 200 = 6.88$

The time spent on waiting for data is $6.88/(2 + 6.88) = 77.5\%$

Impacts of Cache Performance

- Relative cache penalty increases as processor performance improves (faster clock rate and/or lower CPI)
  - The memory speed is unlikely to improve as fast as processor cycle time
  - The lower the $\mathrm{CPI}_{\mathrm{perfect}}$, the more pronounced the impact of stalls
  - When calculating $\mathrm{CPI}_{\mathrm{stall}}$, the cache miss penalty is measured in processor clock cycles needed to handle a miss

Average Memory Access Time (AMAT)

- Hit time was not considered in pervious examples
  - Should be taken into account when designing processor pipeline
  - Normally, the processor pipeline should not stall on hit

- Designers use Average Memory Access Time (AMAT), which includes hit time, to compare cache performance

$$\text{AMAT} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

Example

What is the AMAT for a processor with a 1 ns clock cycle time, a miss penalty of 60 clock cycles, a miss rate of 3% misses per instruction and a cache access time of 1 clock cycle?

$$\text{AMAT} = 1 + 0.03 \times 60 = 2.8 \text{ cycles}$$

Example

Consider only data cache.

The clock rate is determined by the cache hit time.

Time to access cache is 0.5 ns. Time to access the main memory is 30 ns.

25% of the instructions of an application are load/store instructions.

Cache miss rate is 10%.

What is AMAT?

Question

Consider only data cache.

The clock rate is determined by the cache hit time.

Time to access cache is 0.5 ns. Time to access the main memory is 30 ns.

25% of the instructions of an application are load/store instructions.

Cache miss rate is 10%.

What is AMAT?

Cycle time is 0.5 ns.

Hit time = 1 cycle.

Miss penalty is 30 / 0.5 = 60 cycles.

AMAT = 1 + 0.1 × 60 = 7 cycles.

What is the CPI overhead due to data memory accesses?

Solutions

25% of the instructions of an application are load/store instructions.

Cache miss rate is 10%.

What is AMAT?

Hit time = 1 cycle.

Miss penalty is 30 / 0.5 = 60 cycles.

AMAT = 1 + 0.1 × 60 = 7 cycles.

What is the CPI overhead due to data memory accesses?

The CPI overhead is the number of stall cycles per instruction:

$$0.25 \times 0.1 \times 60 = 1.5 \text{ cycles.}$$

Or from AMAT:

Each data access needs 7 cycles, and one cycle is budgeted in the pipeline.

$$0.25 \times (7 - 1) = 1.5 \text{ cycles.}$$

How to improve cache performance?

$$\text{AMAT} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

Let us focus on miss rate first.

Parameters we can tune:

- Cache size
- Block size
- Set associativity

Sources of Cache Misses (3C's)

- Compulsory (cold start or process migration, first reference):
  - First access to a block, "cold" fact of life, not a whole lot you can do about it. If you are going to run "millions" of instruction, compulsory misses are insignificant
- Capacity
  - Cache cannot contain all blocks accessed by the program
- Conflict (collision):
  - Multiple memory locations mapped to the same cache location

Cache size

- Larger cache → More data in cache → Lower miss rates

- However, a larger cache will have a longer access time
  - An increase in hit time may add another stage to the pipeline

- At some point the increase in hit time for a larger cache will overcome the improvement in hit rate leading to a decrease in performance

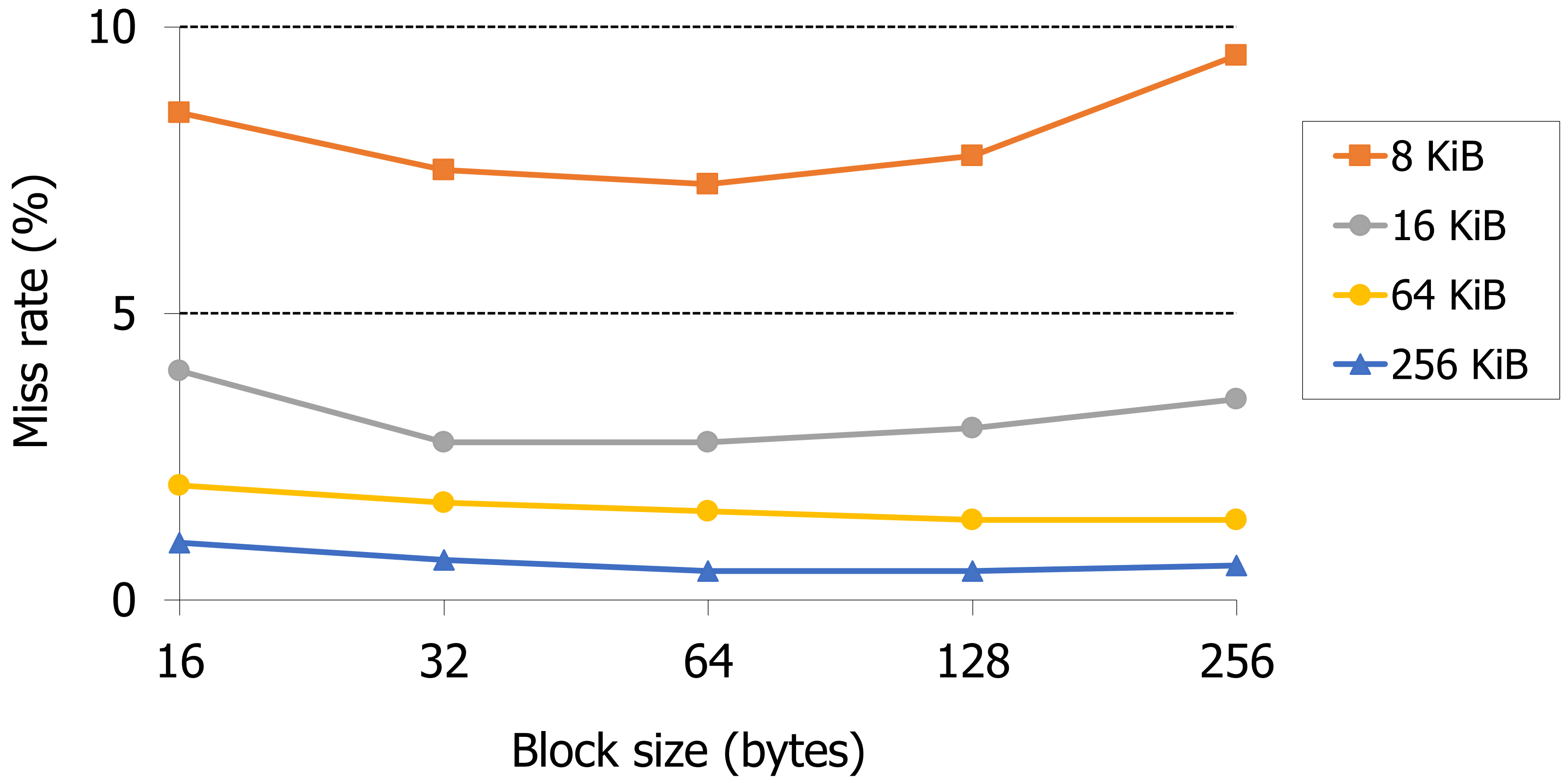
Block size

- Larger block → More data are loaded in cache on a miss

However,

- It takes a longer time to load a block (larger miss penalty)
- A cache of the same size has smaller numbers of blocks
  - Leading to more conflict misses
  - Miss rate goes up if the block size becomes a significant fraction of the cache size
- Data not needed may be loaded into cache
  - Taking away space that can be used for more useful data
  - Increasing capacity misses

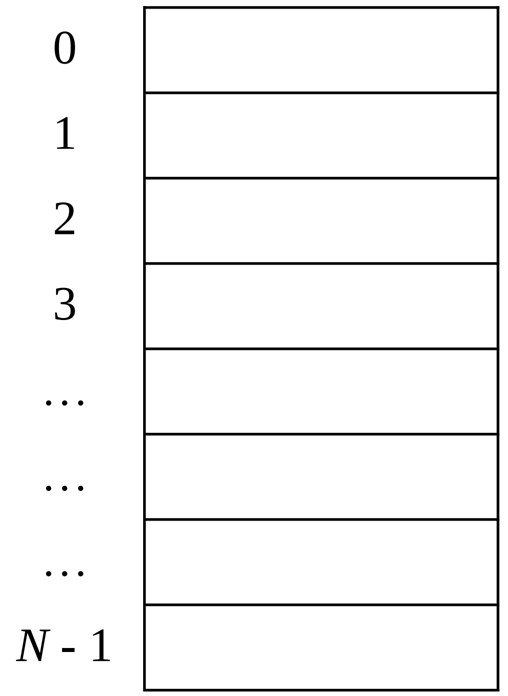
Miss Rate vs Block Size vs Cache Size



Conflict misses

- Consider a direct-mapped cache of 64 KiB.
- Array A is a large word array. A[0] is already in cache.
- What is the smallest index i such that the requests in the following access pattern are always cache misses?

```
A[i], A[0], A[i], A[0], …
```



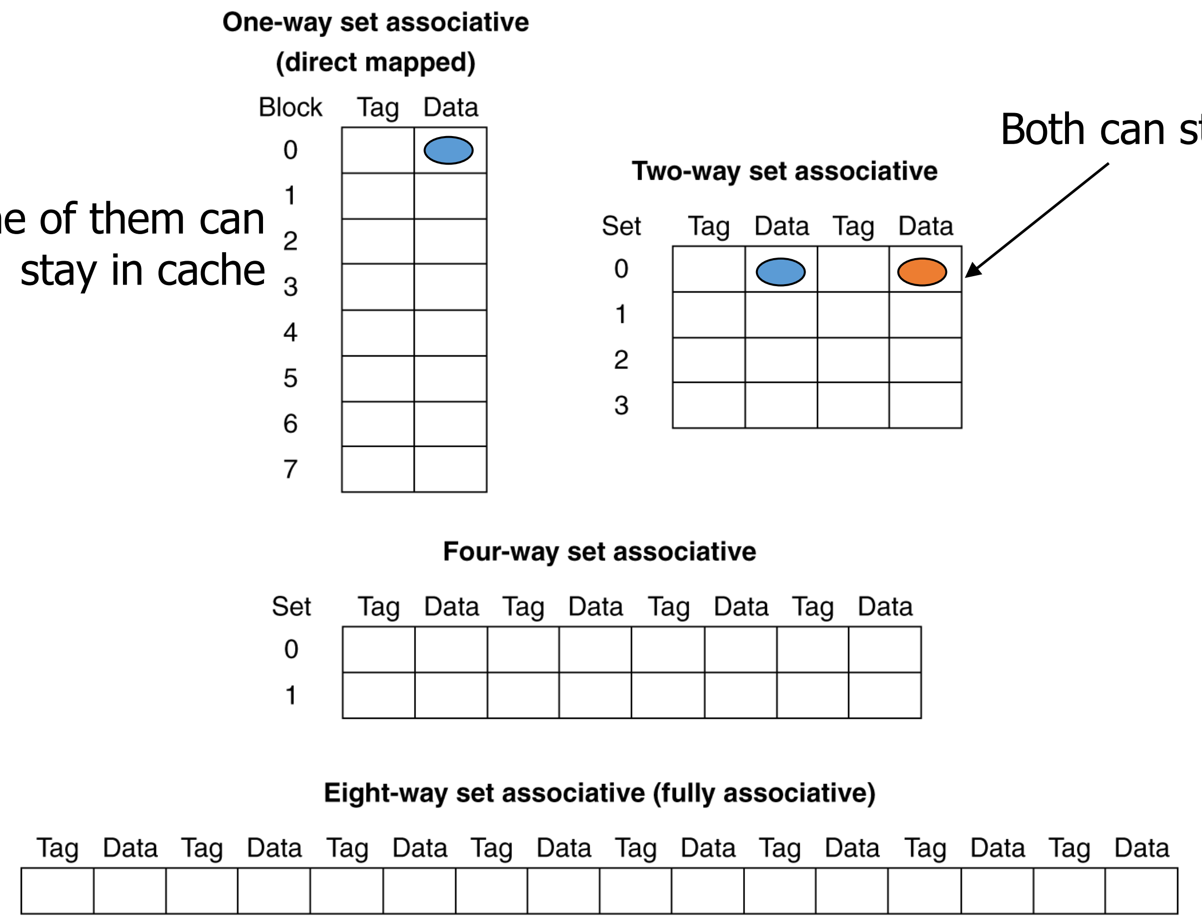
Set Associativity Example: A cache of 8 blocks

A cache index selects a set, which has multiple blocks
A block can be placed anywhere in the set
Downside?

Two blocks having the same cache index

Only one of them can stay in cache

Both can stay in cache

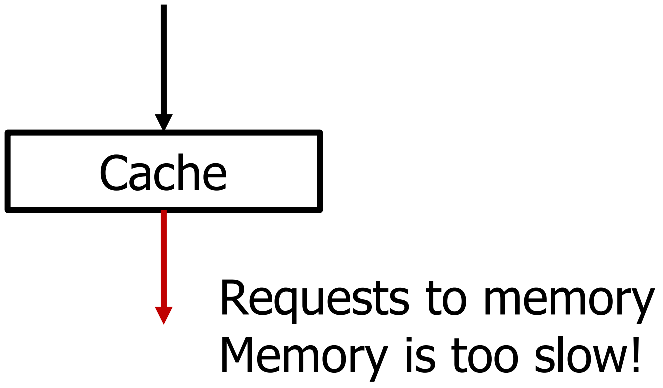


Sources of Cache Misses (3C’s)

- Compulsory (cold start):
  - First access to a block, “cold” fact of life, not a whole lot you can do about it. If you are going to run “millions” of instruction, compulsory misses are insignificant
  - Solution: increase block size (increases miss penalty; very large blocks could increase miss rate)
- Capacity:
  - Cache cannot contain all blocks accessed by the program
  - Solution: increase cache size (may increase access time)
- Conflict (collision):
  - Multiple memory locations mapped to the same cache location
  - Solution 1: increase cache size (may increase access time)
  - Solution 2: increase associativity (may increase access time)

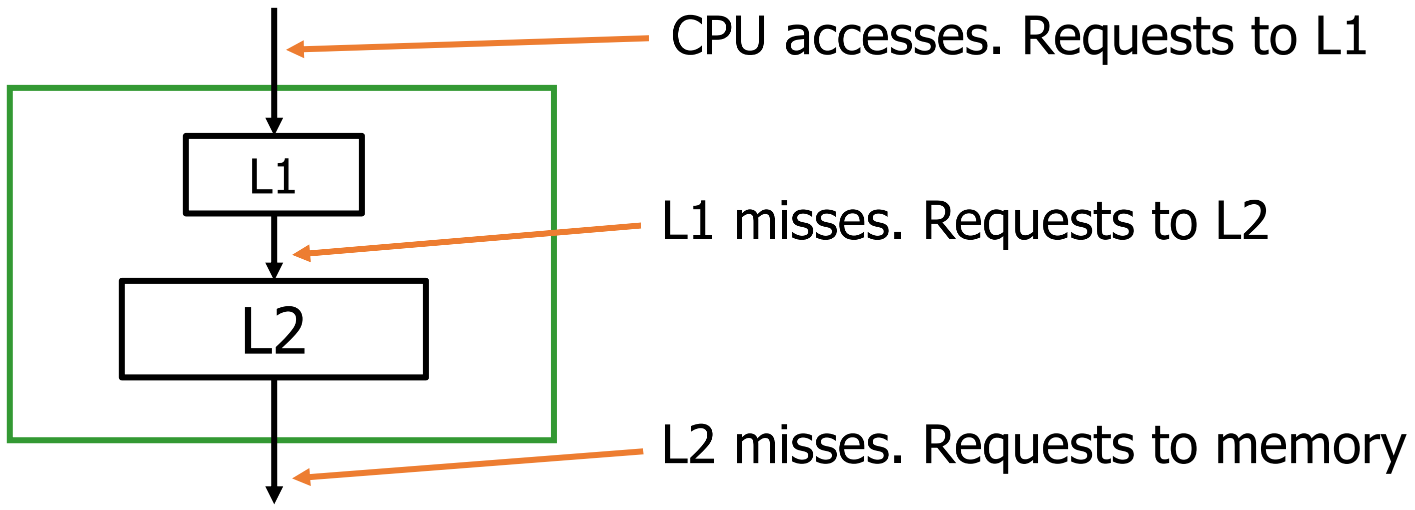
How can we reduce miss penalty?

AMAT = Hit time + Miss rate × Miss penalty



Multilevel Cache

- Use another level (or more levels) of caches
  - With advancing technology have more than enough room on the die for bigger L1 caches *or* for a second level of caches
  - Normally L2 cache is a unified cache (i.e., it holds both instructions and data)
  - Nowadays, many systems have a unified L3 cache



CPU accesses. Requests to L1
L1
L1 misses. Requests to L2
L2
L2 misses. Requests to memory