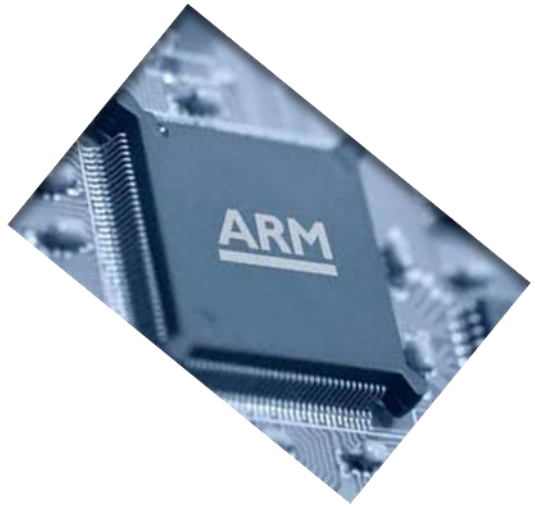
MICROPROCESSOR AND COMPUTER ARCHITECTURE

CACHE PERFORMANCE



Credits:

MPCA Team

CACHE PERFORMANCE

THE PROCESSOR PERFORMANCE EQUATION

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

CACHE PERFORMANCE

Different instruction types having different CPIs

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i$$

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

- One method to evaluate cache performance is to expand processor execution time, i.e. Instruction count - IC.
- Add memory stall cycles - the number of cycles during which the processor is waiting for a memory access.

- **CPU Execution = (CPU clock cycles + Memory stall cycles) x Clock cycle execution time**

CACHE PERFORMANCE

- Equation assumes that CPU clock cycles include
  - The time to handle a cache hit.
  - The processor stalled during the cache miss.
- The number of memory stall cycles depends on
  - The number of misses
  - The cost per miss
  - This is called Miss penalty.

Memory stall cycles = Number of misses x Miss penalty.

$$= IC * \frac{Misses}{Instruction} * Miss\ penalty$$

where, Miss rate = Cache access that result in a miss.

$$x = \frac{\#\ of\ access\ that\ miss}{\#\ of\ access}$$

Note: miss rates and miss penalties are often different for read & writes.

CACHE PERFORMANCE

- Memory stall cycles can be defined in terms of
  - Number of memory accesses per instruction.
  - Miss penalty (in clock cycles for reads & writes)
  - Miss rate (for reads & writes)
- That is,

$Memory\ stall\ clock\ cycles$= IC * Reads per instruction * Read miss rate + IC * writes per instruction * Write miss rate

Normally, simplifying the complete formula,

- Combining reads & writes.
- Finding the average miss rates and miss penalty for reads & writes.

$$= IC * \frac{Memory\ Access}{Instruction} * Miss\ rate * Miss\ Penalty$$

Note: Miss rate is not the only most important measures of the cache design.

CACHE PERFORMANCE – EXAMPLE 1

Assume we have a computer where the cycles per instruction (CPI) is 1.0 when all memory accesses hit in the cache. The only data accesses are loads and stores, and these total 50% of the instructions. If the miss penalty is 25 clock cycles and the miss rate is 2%, how much faster would the computer be if all instructions were cache hits?

SOLUTION:

First compute the performance for the computer that always hits:

CPU execution time = (CPU clock cycles + Memory stall cycles) × Clock cycle

= (IC × CPI + 0) × Clock cycle = IC × 1.0 × Clock cycle

Now for the computer with the real cache, first we compute memory stall cycles:

$$\text{Memory stall cycles} = IC * \frac{Memory\ Access}{Instruction} * Miss\ rate * Miss\ Penalty$$

CACHE PERFORMANCE – EXAMPLE 1

$$\text{Memory stall cycles} = IC \times (1 + 0.5) \times 0.02 \times 25$$
$$= IC \times 0.75$$

where the middle term (1 + 0.5) represents one instruction access and 0.5 data accesses per instruction. The total performance is thus

$$\text{CPU execution time cache} = (IC \times 1.0 + IC \times 0.75) \times \text{Clock cycle}$$
$$= 1.75 \times IC \times \text{Clock cycle}$$

The performance ratio is the inverse of the execution times:

$$\frac{CPU\ execution\ time\ (with\ cache)}{CPU\ execution\ time\ (without\ cache)} = \frac{1.75\ *IC\ *Clock\ cycle}{1.0\ *IC\ *Clock\ cycle} = 1.75$$

Hence, The computer with no cache misses is 1.75 times faster.

CACHE PERFORMANCE

- CPU performance is measured by
  - **Instruction Count**
  - **Miss Rate**
  - Both independent of hardware
- A better measure of memory hierarchy performance is - **Average memory access time.**

Average memory access time = Hit time + Miss rate x Miss penalty.
where Hit time – Time to hit in the cache.

The components of a average access time can be measured either as

- Absolute time on a hit, say, 0.25 to 1.0 ns.
- Number of clock cycles that the processor waits for the memory, such as 150 to 200 clock cycles.

Note:

- Average memory access time is still an indirect measure of performance., although, better measure than miss rate.
- This formula can help to decide between spilt cache and a unified cache.

CACHE PERFORMANCE

- CPU time = [CPU execution clock cycles + Memory stall clock cycles] x Clock cycle time.

The equation has a question:

- Clock cycles for a cache hit should be considered part of
  - CPU execution clock cycles? or
  - Memory stall clock cycles?

- **Most widely accepted, is to include in CPU execution clock cycles.**

CACHE PERFORMANCE – EXAMPLE 2

Assume that the cache miss penalty is 200 clock cycles, and all instructions normally take 1.0 clock cycles (ignoring memory stalls). Assume that the average miss rate is 2%, there is an average of 1.5 memory references per instruction, and the average number of cache misses per 1000 instructions is 30. What is the impact on performance when behavior of the cache is included? Calculate the impact using both misses per instruction and miss rate.

SOLUTION:

CPU_{time} = IC x [$CPI_{execution}$ + Memory stall clock cycles per instructon] x Clock cycle time

The performance, including cache misses, is

$CPU\ time_{with\ Cache}$ = IC x [1.0 + (30/1000 x 200) x Clock cycle time

= IC x 7.00 x Clock cycle time.

The clock cycle time and instruction count are the same, with or without a cache. Thus CPU time increases sevenfold, with CPI from 1.00 for a perfect cache to 7.00 with cache that can miss.

Cache Optimizations

The equation,

Average memory access time = Hit time + Miss rate x Miss penalty.

Gives a frame work to present cache optimizations for improving cache performance. Hence, six basic cache optimizations can be organized into three categories.

- **Reducing the miss rate**
  - Larger block size, Larger cache size, and higher associativity.
- **Reducing the miss penalty**
  - Multilevel caches and giving reads priority over the writes.
- **Reducing the time to hit in the cache**
  - Avoiding address translation when indexing the cache.

CACHE OPTIMIZATIONS

The classical approach to improve cache behaviour is to reduce miss rates.

Three categories misses are:

- **Compulsory :**
  - The very first access to block cannot be in the cache. So, the block must be brought into the cache.
  - These are called cold-start misses or first reference misses.

- **Capacity:**
  - If the cache cannot contain all the blocks needed during execution of a program, capacity misses [in addition to compulsory misses] will occur because of blocks being discarded and later retrieved.

Cache Optimizations

Three categories misses are:

- **Conflict:**
  - If the block placement strategy is set associative or direct mapped, conflict misses [in addition to compulsory and capacity misses] will occur because a block may be discarded and later retrieved if too many blocks map to its set. These misses are also called collision misses.
  - The idea is that hits in a fully associative cache that become misses in an n-way set associative cache, due to more than n requests on some popular sets.

NOTE: Further adding a 4th C , for coherence misses due to flushes to keep multiple caches coherent in a multiprocessor environment.

CACHE OPTIMIZATIONS

- **Compulsory Misses :**
  - Those that occur in an infinite cache.
- **Capacity Misses:**
  - Those that occur in a fully associative cache.
- **Conflict Misses:**
  - Those that occur going from fully associative to eight-way associative, four–way associative , and so on…

To show the benefit of associativity, conflict misses are divided into misses caused by each decrease in associativity.

Here are the four divisions of conflict misses and how they are calculated:

- *Eight-way*—Conflict misses due to going from fully associative to eight-way associative
- *Four-way*—Conflict misses due to going from eight-way associative to four way associative
- *Two-way*—Conflict misses due to going from four-way associative to two way associative
- *One-way*—Conflict misses due to going from two-way associative to one way associative

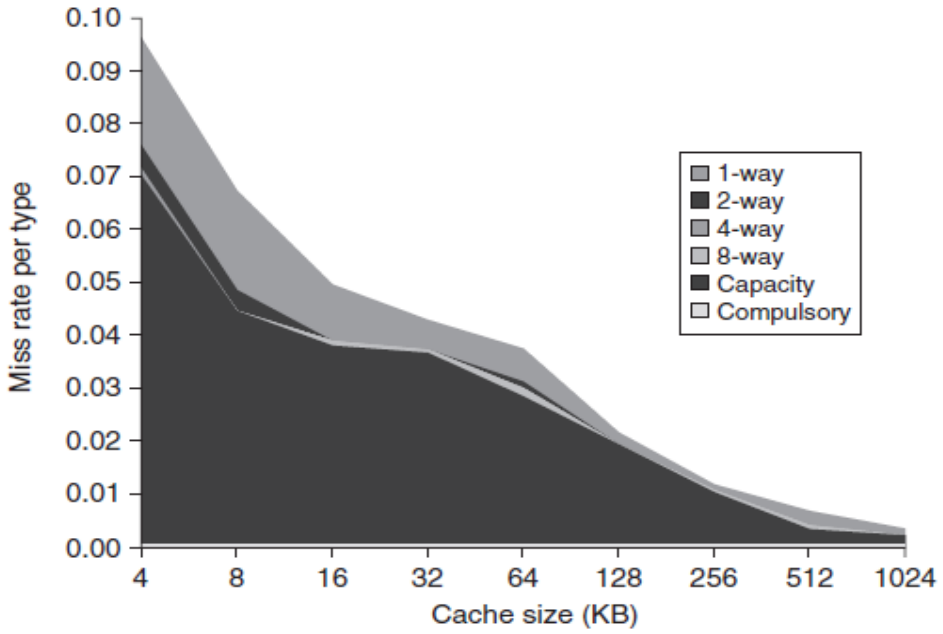
CACHE OPTIMIZATIONS

Conceptually, conflicts are the easiest:

- Fully associative placement avoids all conflict misses,
- Expensive in hardware and
- May slow down the processor clock rate leads to lower overall performance.

Cache size (KB)	Degree associative	Total miss rate	Miss rate components (relative percent) (sum = 100% of total miss rate)					
			Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%

CACHE OPTIMIZATIONS



Total miss rate for each size cache according to the three C's for the data diagram shows the actual data cache miss rates,

6 Basic Cache Optimizations

- **First Optimization:** Larger Block Size to Reduce Miss Rate.
- **Second Optimization:** Larger Caches to Reduce Miss Rate.
- **Third Optimization:** Higher Associativity to Reduce Miss Rate.
- **Fourth Optimization :** Multilevel Caches to Reduce Miss Penalty.
- **Fifth Optimization:** Giving Priority to Read Misses over Writes to Reduce Miss Penalty.
- **Sixth Optimization:** Avoiding Address Translation during Indexing of the Cache to Reduce Hit Time.

Q & A
Cache Optimizations