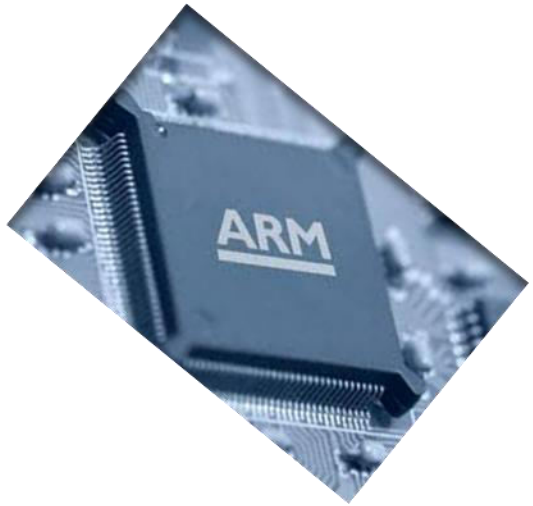
MICROPROCESSOR AND COMPUTER ARCHITECTURE

CACHE OPTIMIZATION



Credits:

MPCA Team

CACHE OPTIMIZATIONS - RECAP

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

- Six "easy" ways to improve above equation
- Reducing miss rate:

1. Larger block size (compulsory misses)
2. Larger cache size (capacity misses)
3. Higher associativity (conflict misses)

- Reducing miss penalty:

4. Multilevel caches
5. Prioritize reads over writes

- Reducing hit time:

6. Avoiding address translation

CACHE OPTIMIZATIONS - RECAP

Uniprocessor **misses** fall into **three categories**:

- **Compulsory:** First access (ever to location)

Capacity : Program accesses more than will fit in cache

- **Conflict:** Too many blocks map to given set
  - Sometimes called collision miss
  - Can’t happen in fully associative caches

CACHE OPTIMIZATIONS- REDUCE MISS RATE

FIRST OPTIMIZATION : Larger Block Size to Reduce Miss Rate

- **Brings in more data per miss**
  - Reduce compulsory misses
  - Larger block size take advantage of principle of locality
    - Temporal & **Spatial Locality**
  - Unfortunately can increase conflict and capacity misses.
  - Increase **Miss Penalty.** May compensate the decrease in miss rate.
- Selection of the block size:
  - Latency & Bandwidth of the lower level memory.
  - High Latency & High Bandwidth encourage large block size.
  - Low Latency & Low Bandwidth encourage small block size.

Ex: Twice miss penalty of a small block may be close to the penalty of a block twice the size

Example 1: Assume the memory system takes 80 clock cycles of overhead and then delivers 16 bytes every 2 clock cycles. That is, it can supply 16 bytes in 82 clock cycles, 32 bytes in 84 clock cycles, and so on… Which block size has the smallest average memory access time for each cache size?

Ans: Average access time = Hit time + Miss rate x Miss penalty

If we assume that the hit time is 1 clock cycle independent of the block size, then,

The access time for a 16- byte block in a **4KB cache is**

Average access time = 1 + (8.57% x 82)

= 1 + 7.0274

= **8.0274 clock cycles.**

For, 256 byte block in a 256 KB cache the

Average access time = 1 + (0.49% x (80+32))

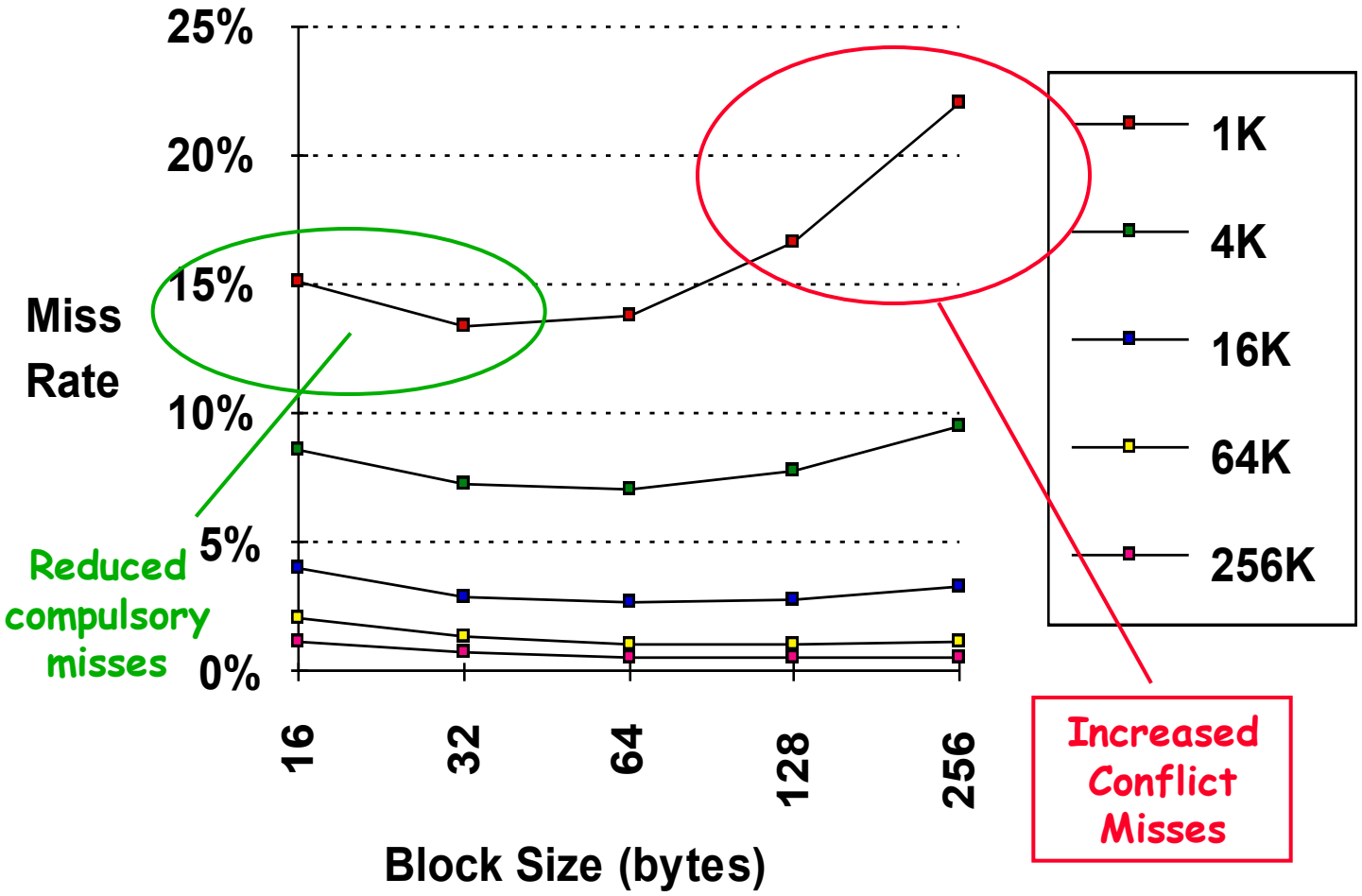
= 1 + (0.5488)

= **1.5488 clock cycles.**

	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Figure B.11 Actual miss rate versus block size for the five different-sized caches in Figure B.10. Note that for a 4 KB cache, 256-byte blocks have a higher miss rate than 32-byte blocks. In this example, the cache would have to be 256 KB in order for a 256-byte block to decrease misses.

Miss rate Vs Block size: Larger Block Size (fixed size & assoc)



What else drives up block size? - Associativity

SECOND OPTIMIZATION :

Larger Caches to Reduce Miss Rate

- Increase Cache Capacity to reduce capacity misses.
- Very popular in Off- chip caches.
- Drawbacks:
  - Longer hit time
  - Higher cost & power.

Block size	Cache size			
	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Figure B.11 Actual miss rate versus block size for the five different-sized caches in Figure B.10. Note that for a 4 KB cache, 256-byte blocks have a higher miss rate than 32-byte blocks. In this example, the cache would have to be 256 KB in order for a 256-byte block to decrease misses.

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%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

THIRD OPTIMIZATION :

Higher Associativity to Reduce Miss Rate

Comparision:

- Data suggests 8 way associative is often as good as fully associative.
- 2:1 cache rule (for cache size 128KB)

A direct-mapped cache of size N has the same miss rate of 2-way associative cache of size N /2.

Problem: Greater associativity comes at the cost of increased hit time

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%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

THIRD OPTIMIZATION :

Higher Associativity to Reduce Miss Rate

Example 2: Assume that higher the associativity would increase the clock cycle time as listed.

$$\text{Clock cycle time}_{2\text{-way}} = 1.36 \times \text{Clock cycle time}_{1\text{-way}}$$
$$\text{Clock cycle time}_{4\text{-way}} = 1.44 \times \text{Clock cycle time}_{1\text{-way}}$$
$$\text{Clock cycle time}_{8\text{-way}} = 1.52 \times \text{Clock cycle time}_{1\text{-way}}$$

The hit time is 1 clock cycle, that the miss penalty for the direct mapped case is 25 clock cycles to a level 2 cache that never misses, and that the miss penalty need not be rounded to an integral number of clock cycles. Using, the details in the table for miss rates, for which cache sizes are each of these three statements true?

$$\text{Average memory access time}_{8\text{-way}} < \text{Average memory access time}_{4\text{-way}}$$
$$\text{Average memory access time}_{4\text{-way}} < \text{Average memory access time}_{2\text{-way}}$$
$$\text{Average memory access time}_{2\text{-way}} < \text{Average memory access time}_{1\text{-way}}$$

CACHE OPTIMIZATIONS- THIRD OPTIMIZATION

Cache size (KB)	Associativity			
	1-way	2-way	4-way	8-way
4	3.44	3.25	3.22	3.28
8	2.69	2.58	2.55	2.62
16	2.23	2.40	2.46	2.53
32	2.06	2.30	2.37	2.45
64	1.92	2.14	2.18	2.25
128	1.52	1.84	1.92	2.00
256	1.32	1.66	1.74	1.82
512	1.20	1.55	1.59	1.66

Figure B.13 Average memory access time using miss rates in Figure B.8 for parameters in the example. Boldface type means that this time is higher than the number to the left, that is, higher associativity *increases* average memory access time.

Ans:

Average memory access time $_{8\text{-way}}$ = Hit Time + Miss Rate x Miss Penalty

= 1.52 + Miss Rate x 25

access time $4_{\text{-way}}$ = 1.44 + Miss Rate x 25

access time $2_{\text{-way}}$ = 1.36 + Miss Rate x 25

access time $_{1\text{way}}$ = 1.00 + Miss Rate x 25

The miss penalty is the same time in each case, i.e, 25 clock cycles.

Ex: The average memory access time for 4KB direct–mapped cache is

Average memory access time $_{1\text{-way}}$ = 1.00 + (0.098 x 25) = 3.45

And the time for a 512 KB, eight-way set associative cache is

Average memory access time $_{8\text{-way}}$ = 1.52 + (0.006 x 25) = 1.67

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

- Observing the cache performance formula,
  - **Avg. memory access time = Hit time + Miss rate x Miss penalty**
- Improvements in Miss penalty is as advantageous as improvements in miss rate.
- The performance gap between processor & memory raises a question:
  - Should I make the cache faster to keep the pace with the speed of the processor? **Or**
  - Make the cache larger to overcome the widening gap between the processor & the main memory?

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

- Answer for these questions is to do both.
  - **Adding another level of cache between memory & original cache simplifies the decision.**

- First level cache can be small enough to match the clock cycle time of the processor.

- Second level cache be can be large enough to capture many accesses that would go to main memory, thus reducing miss penalty.

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

- Multilevel cache will complicate performance analysis.

- Considering Memory access time for a two level cache using subscripts L1 & L2 to refer, respectively, to the first & second level

- The original formula is:

$$\textbf{AMAT} = \textbf{Hit Time}_{L1} + \textbf{Miss Rate}_{L1} \times \textbf{Miss Penalty}_{L1}$$

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

and

$$\textbf{Miss Penalty}_{L1} = \textbf{Hit Time}_{L2} + \textbf{Miss Rate}_{L2} \times \text{Miss Penalty}_{L2}$$

Substituting in original equation, we get,

$$\textbf{AMAT} = \textbf{Hit Time}_{L1} + \textbf{Miss Rate}_{L1} \times [\text{Hit Time}_{L2} + \text{Miss Rate}_{L2} \times \text{Miss Penalty}_{L2}]$$

Here, Second level miss rate is measured on the leftovers from the first level cache.

To avoid ambiguity, the following terms are used for a two level cache system.

- **Local miss rate**
- **Global miss rate**

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

Local miss rate:

•The number of misses in the cache divided by the total number of memory accesses to this cache.

Ex: For first level cache it is, **Miss Rate$_{L1}$**

For second level cache it is, **Miss Rate$_{L2}$**

Global miss rate:

•The number of misses in the cache divided by the total number of memory accesses generated by the processor.

Ex: Global miss rate for level1 cache is still **Miss Rate$_{L1}$**

but, for level2 cache it is : **Miss Rate$_{L1}$ x Miss Rate$_{L2}$**

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

- Local miss rate is large for second level caches because
  - The first level cache skims the cream of the memory accesses.
  - Global miss rate is more useful measure.
    - It indicates what fraction of the memory accesses that leave the processor go all the way to memory.
  - Here, the misses per instruction metric shines.

Expanding the memory stalls per instruction to add the impact of a second level,

Avg. Memory stalls per instruction = Misses per instruction$_{L1}$ x Hit Time$_{L2}$ + Misses per instruction$_{L2}$ x Miss Penalty$_{L2}$

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty - Example

Suppose that in 1000 memory references there are 40 misses in the first level cache and 20 misses in the second –level cache. What are the various miss rates?

Assume the miss penalty from the L2 cache to memory is 200 clock cycles, the hit time of the L2 cache is 10 clock cycles, hit time for L1 cache is 1 clock cycle and there are 1.5 memory references per instruction.

What is the average memory access time and average stall cycles per instruction?

Ignore impact of writes.

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty - Example

Answer:

The miss rate [either global or local] for the first level cache is 40/1000 = 4%.

The local miss rate for the second-level cache is 20/40 = 50%.

The global miss rate of the second level cache is 20/1000 = 2%.

Then,

AMAT = Hit Time$_{L1}$ +_Miss Rate$_{L1}$ x [Hit Time$_{L2}$ +_Miss Rate$_{L2}$ x Miss Penalty$_{L2}$]

= 1+4% x (10 + 50% x 200) = 1 + 4% x 110 = 5.4 clock cycles.

# of misses per instruction $= \frac{\#\ of\ memory\ references}{\#\ of\ memory\ references\ per\ instructions} = \frac{1000}{1.5}$ **= 667 ins.**

Thus, for L1 cache ,

 # misses for 40 memory accesses for 1000 instructions = 40 x 1.5 = 60 misses and for 20 misses for L2 cache it is 1.5 x 20 = 30 misses.

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty - Example

Average memory stalls per instruction = Misses per instruction $_{L1}$ x Hit time$_{L2}$ + Misses per instruction $_{L2}$ x Miss Penalty

= (60/1000) x 10 + (30/1000) x 200

= 0.06 x 10 + 0.03 x200 = 0.06 + 6

= 6.6 clock cycles.

Then, Average memory = (AMAT - Hit time$_{L1}$) x Average # of memory references per stalls per instruction

= (5.4 - 1.0) x 1.5 = 6.6 clock cycles.

Note: The computation of the memory stalls per instruction is same for either way.

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

- Note:
- Formulae are for combined reads & writes.
- A write back first level cache

A write-through first level cache will send

- Misses
- All writes to the second level
- May use a write buffer.

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

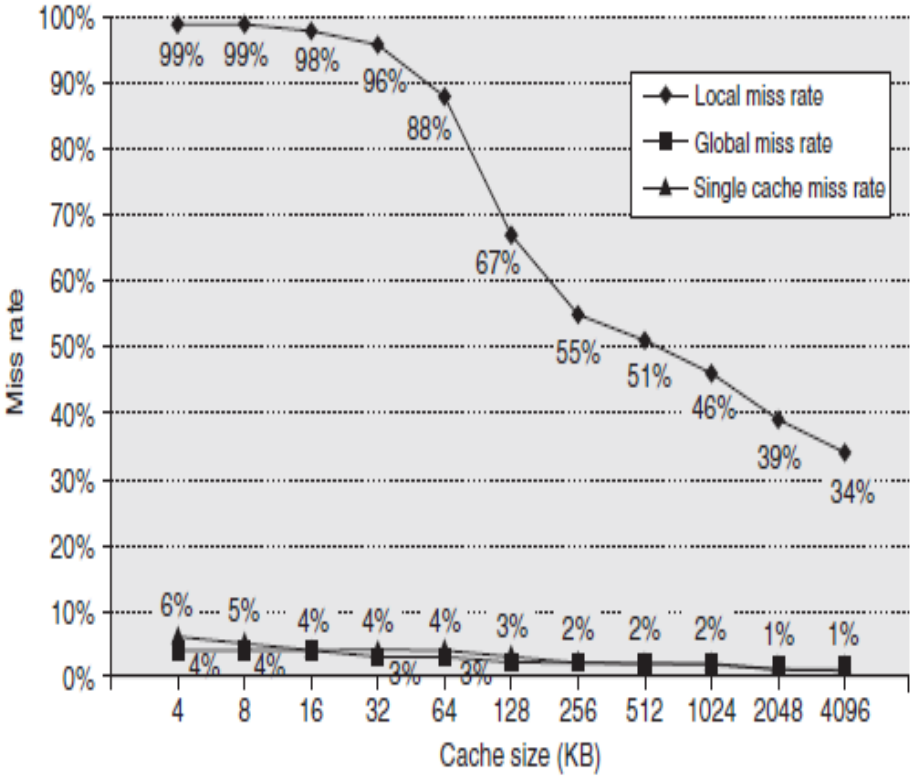


Figure B.14 Miss rates versus cache size for multilevel caches. Second-level caches *smaller* than the sum of the two 64 KB first-level caches make little sense, as reflected in the high miss rates. After 256 KB the single cache is within 10% of the global miss rates.

First perspective :

Global cache miss rate is very similar to the single cache miss rate of the second level cache.

- Provided that the second level cache is much larger than the first level cache.

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

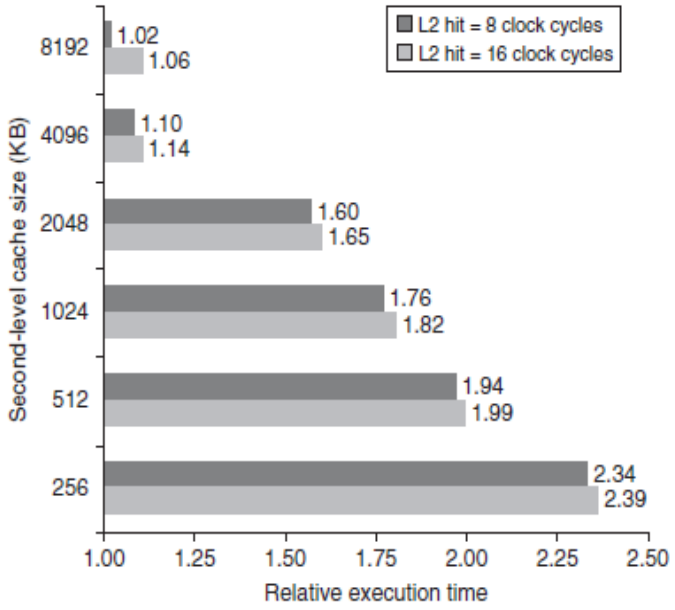


Figure B.15 Relative execution time by second-level cache size. The two bars are for different clock cycles for an L2 cache hit. The reference execution time of 1.00 is for an 8192 KB second-level cache with a 1-clock-cycle latency on a second-level hit. These data were collected the same way as in Figure B.14, using a simulator to imitate the Alpha 21264.

Second Perspective:

- Local cache miss rate is not a good measure of secondary caches.
- It is a function of the miss rate of the first level cache.
- Can vary by changing the first – level cache.

Note: Global cache miss rate should be used when evaluating second level caches.

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

Parameters for second level caches:

- Difference between two levels is the speed of the first level cache
  - That affects the clock rate of the processor.
  - While, Speed of the second level cache only affects the miss penalty of the first level cache.
  - Thus, many alternatives can be considered in second level cache that are ill chosen for the first level cache.

- **Two major questions for the design of the second level cache:**
  1. Will it lower the average memory access time portion of the CPI?
  2. How much does it cost ?

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

- Initial decision is the size of a second level cache.
- Everything in first level cache is likely in the second level cache.
- The size of the second level cache should be much higher than the first.
- If second level caches are just a little bigger, the local miss rate will be higher.
- Thus, inspires the design of huge second level caches.

NOTE: Multi level inclusion is the natural policy for memory hierarchies.

- L1 data is always present in L2.

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

Example Given the data below, what is the impact of second-level cache associativity on its miss penalty?

- Hit time$_{L2}$ for direct mapped = 10 clock cycles.
- Two-way set associativity increases hit time by 0.1 clock cycle to 10.1 clock cycles.
- Local miss rate$_{L2}$ for direct mapped = 25%.
- Local miss rate$_{L2}$ for two-way set associative = 20%.

Answer For a direct-mapped second-level cache, the first-level cache miss penalty is

$$\text{Miss penalty}_{\text{1-way L2}} = 10 + 25\% \times 200 = 60.0 \text{ clock cycles}$$

Adding the cost of associativity increases the hit cost only 0.1 clock cycle, making the new first-level cache miss penalty:

$$\text{Miss penalty}_{\text{2-way L2}} = 10.1 + 20\% \times 200 = 50.1 \text{ clock cycles}$$

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

In reality, second-level caches are almost always synchronized with the first-level cache and processor. Accordingly, the second-level hit time must be an integral number of clock cycles. If we are lucky, we shave the second-level hit time to 10 cycles; if not, we round up to 11 cycles. Either choice is an improvement over the direct-mapped second-level cache:

$$\text{Miss penalty}_{\text{2-way L2}} = 10 + 20\% \times 200 = 50.0 \text{ clock cycles}$$
$$\text{Miss penalty}_{\text{2-way L2}} = 11 + 20\% \times 200 = 51.0 \text{ clock cycles}$$

Fourth optimization:
Multilevel Caches to Reduce Miss Penalty

The essence of all cache designs is:

- Balancing fast hits and few misses.

- For second level caches,
  - Many fewer hits than in the first level cache.
  - Emphasis shifts to fewer misses.

- Insight leads to much larger caches and techniques to lower the miss rate:
  - Such as higher associativity & larger blocks.

Fifth optimization: Giving priority to Read misses over Write misses to reduce miss penalty

- This optimization serves reads before writes have been completed.
- Complexities of a write buffer with a **write-through cache:**
  - The most improvement is a write buffer of a proper size.
  - Memory accesses are complicated as it may hold the updated value of a location needed on a read miss.

Fifth optimization: Giving priority to Read misses over Write misses to reduce miss penalty

To resolve this ambiguity,

- Read miss to wait until the write buffer is empty.

  or

- To check the contents of the write buffer on a read miss and if there are no conflicts and the memory system is available.
- Let the read miss continue.
- Virtually, all processors use the later approach.
- Gives priority reads over writes.

Fifth optimization: Giving priority to Read misses over Write misses to reduce miss penalty

Write-back cache:

The cost of the writes by the processor in a write back can also be reduced.

Consider a read miss replacing a dirty block.

Instead of writing the dirty block to the memory, and then reading memory, we could copy the dirty block to a buffer, then read memory and then write memory will finish sooner.

Thus, if a read miss occurs, the processor can either stall until the buffer is empty or

Check the addresses of the words in the buffer for conflicts.

Fifth optimization: Giving priority to Read misses over Write misses to reduce miss penalty

Consider the following code sequence.

```
Ex:  STR  R3, 512 (R0)
     LDR  R1, 1024 ( R0)
     LDR  R2, 512 (R0)    Assume Direct mapped:
```

Write –through cache that maps 512 and 1024 to the same block. Four word write buffer that is not checked on a read miss. Will the value in R2 always be equal to the value in R3? o R2!

Fifth optimization: Giving priority to Read misses over Write misses to reduce miss penalty

Ans:

- This is a read-after-write data hazard in memory.
- The data in R3 are placed into the write buffer after the STR.
- The following LDR instruction uses the same cache index and is therefore a miss.
- The second LDR instruction, tries to put the value in location 512 into the register R2.
- This also results in a miss.
- If the write buffer hasn't completed writing to location 512 in memory,
- The read of location 512 will put the old, wrong value into the cache block and then into R2.
- Without proper precautions, R3 would not be equal to R2!

Q & A
Cache Optimizations