Improvement of Cache Memory Performance by Reducing Cache Miss Rate and Miss Penalty

Nagmden Miled shtewi[1] **Abdurrezagh S. Elmezughi**[2]

Faculty of Engineering, Azzaytuna University[12]

najmaddindf@gmail.com

تاريخ الاستلام 2024-04-19

الملخص :

ذاكرة التخزين المؤقت هي جزء صغيرة من الذاكرة التي تعد جزءًا من وحدة المعالجة المركزية وهي أقرب فعليًا إلى وحدة المعالجة المركزية من ذاكرة الوصول العشوائيRAM . كلما زادت ذاكرة التخزين المؤقت، زاد عدد البيانات التي يمكن تخزينها بالقرب من وحدة المعالجة المركزية. تحتوي الذاكرة المؤقتة على التعليمات/البيانات المستخدمة بشكل متكرر والتي قد يحتاجها المعالج بعد ذلك، وهي ذاكرة وصول أسرع من ذاكرة الوصول العشوائي، كما أنها تقلل من وقت الوصول إلى المعلومات عن طريق تخزين البيانات والتعليمات المستخدمة بشكل متكرر. تقلل الذاكرة المؤقتة من مرات انتظار المعالج للبيانات والتعليمات من الذاكرة الرئيسية للنظام، مما يؤدي إلى تحسين سرعة وحدة المعالجة المركزية.

Abstract

Cache is a small amount of memory which is part of the CPU which is physically closer to the CPU than RAM is the more cache there is the more data can be stored closer to the CPU. The cache contains frequently used instructions/data that are used by the processor The following may be required, which is faster access memory than RAM, , It reduces the information access time by storing frequently used data and instructions. Cache memory reduces of times processor waits for data and instructions from the main memory of the system, resulting in the improved speed CPU.

Keywords: Cache memory, Miss Rate, Hit Rate, Miss Penalty.

Introduction

Cache memory is one of the basic elements of computing devices that has helped in improving the computational capabilities of computing devices. Modern digital devices have cache, or fast memory access, which contributes to faster calculations and better functionality. The cache contributed a fair amount to generating the computational discrepancy between the processor's performance and speed . This is due to the essential information skips that the caches usually make while handling data at the highest clock speeds. This often leads to a reduced capacity of the cache, as mentioned in the processor specifications [1].Cache memory is divided into two different parts; one is cached data memory and another is cache tag memory. Cache data memory contains various collections of memory words called cache block or line or page. Each cache block has a block address or tag. Collection of all block addresses or tags is called cache tag memory. Whenever central processor needs any knowledge or instruction it sends address to the cache memory, address is searched in cache memory firstly, if data is found in cache it is given to the CPU. Finding a data in cache memory is named as cache HIT. If data isn't found in cache then it's referred to as cache MISS, in this case address is searched in RAM for the data or instruction. After data is received from RAM, a block of data is transferred from RAM to cache memory so all any request is consummated from cache. Performance of cache memory is measured in HIT Ratio[2].

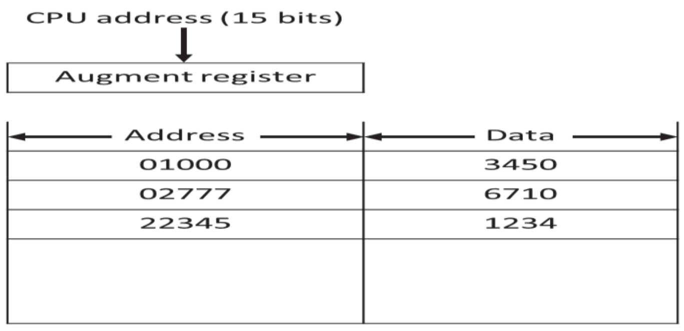
Background

The concept used in this research has been based on certain techniques as described below.

A. Existing Mapping Techniques

There are three main mapping techniques for organization of Cache.

*1) Associative Mapping:*Cache memory works on the basis of associative memory since it is the most flexible and fastest. An associative memory is used to store addresses and content of data for the word memory which is shown in Fig 1.



Address	Data
01000	3450
02777	6710
22345	1234

Figure1: Associative mapping Cache [3].

2) *Direct Mapping*: In direct mapping, the cache consists of normal high-speed random-access memory. Each location in the cache holds the data, at a specific address in the cache. This address is given by the lower prominent bits of the RAM address. This

enables the block to be selected directly from the lower significant bit of the memory address This address is given by the least significant bits of the main memory address. This allows the block to be determined directly from the least significant part of the memory address as shown in Fig.2.

3) Set Associative Mapping: It is another type of cache organization, which is built above the organization of direct mapping to improve its performance by giving the ability to each word in the cache to store two or more words using the same index address. In this organization, every word will be stored together with its tag and then a set will be formed by considering several tag items of data in just one word of cache. To illustrate in Fig 3.

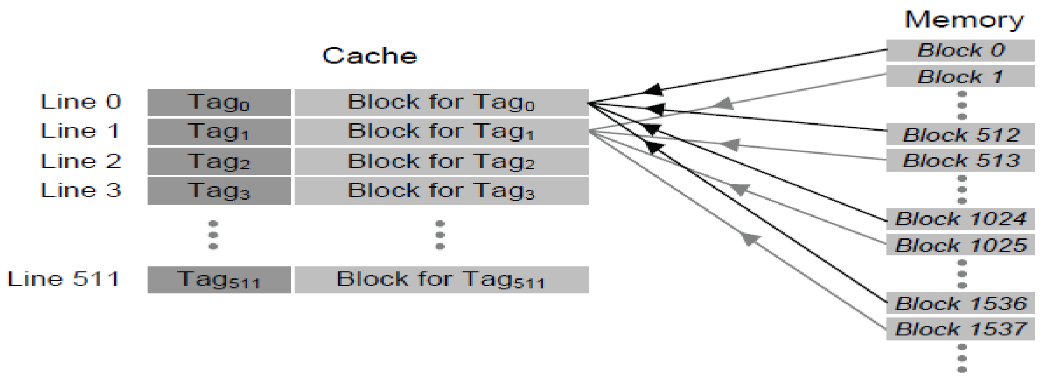


Figure 2: Direct Mapping of RAM to Cache[4].

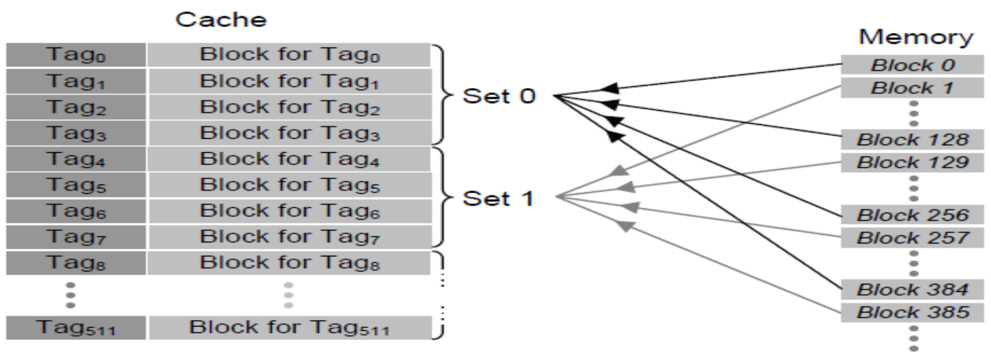


Figure 3:Set Associative Mapping of RAM to Cache.[4]

If a request from memory is generated by the processor, the cache will be accessed by the index value of the address. After that, the tag field of the processor address and combined two tags in cache memory compared with each other in order to fetch the required data and then sees whether the match occurs or not. The logical comparison will be handled by searching the tags associatively in the set in the same way we search in associative memory. Due to this it is named as 'set associative'. An improvement in the hit ratio will be shown due to the rise of set size since different tags can be assigned to the same index[3].

Literature review

Various researches have shown how to an enhancing the performance. Chaplot [5] He proposed an approach that could provide higher correlation to improve the cache miss rate and discussed two types of spatio-temporal locality , i.e. temporal locality and spatial locality respectively. Dimitrios Stiliadis[6] researchers have method for improving the performance of victim caching. The method selectively places data in the main cache or the victim cache based on a prediction algorithm. Hiroyuki Tomiyama[7] Is proposed a size-constrained code placement method which minimizes miss rates of instruction caches under constraint on code size given by system designers.

Cache controller

Cache Controller is designed to manage the operation of cache memory in a computer system. It plays a crucial role in improving the efficiency and performance of the computer by reducing the time required for the CPU to access data from the main memory. It handle the request by dividing the index, tag, valid and dirty bits, associated with a whole block of data The set index is used to select the corresponding line from the cache. If a valid bit represents that the line is active, the flag will be compared. If successful in both cases, the item is fetched and considered a Cache Hit, otherwise a Cache Miss. [9]

Cache Hit

In cache, the requested data is found in the cache, allowing the data to be retrieved quickly. In contrast, a cache miss means that the data is not in the cache at the time of the request, necessitating slower retrieval from main memory or another lower-level cache[8].

Cache Miss

Cache miss Data not found in cache, Processor loads data from memory and copies into cache. This results in extra delay, called miss penalty. In this case, the data request is delayed by a reasonable amount of time and the processor has to wait for the data to arrive from the memory which is a comparatively time taking process.[8]

$$Miss\ rate = \frac{Misses}{Memory\ Acesses} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (1)$$

$$Miss\ rate = 1 - Hit\ rate \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$$

Classifying Misses

Compulsory : Compulsory misses happen when a block is referenced for the first time. The computer can‛t get a block that doesn‛t exist yet.

Capacity: If the cache cannot hold all the required blocks during program execution, a capacity loss will occur because the blocks are discarded and retrieved later.

Conflict: These types of errors just happen In the directly mapped and specified cache. Multiple blocks Can be assigned to a group, forcing Evictions when the pool is full, that become misses in an n-way set-associative cache are due to more than n requests on some popular sets. [9]

$$n - way\ Associative. = (Block\ address) MOD\ (Number\ of\ sets\ in\ cache) \ldots (3)$$

Miss Penalty

It is the time to copy data from main memory to the cache. This often requires dozens of clock cycles (at least). The time taken to bring the data to cache from memory, in case of cache miss, is regarded as miss penalty.[9]

$$\text{Miss Penalty} = Memory\ Access\ Time + Send\ Address\ Time + \left(\frac{Cache\ Size}{Block\ \ Size}\right). (4)$$

Measuring Cache Performance

AMAT (Average Memory Access Time) is a measure that is used to test performance of a memory. It is calculated using following formula:

$$Average\ memory\ access\ time (AMAT) = Hit\ Time (HT) + Miss\ rate (MR) + Miss\ penalty (MP) \ldots.. \quad (5)$$

Hence, we organize six cache optimizations into three categories:

- **Reducing the miss rate**: and higher associativity ,larger block size and larger cache size.
- **Reducing the miss penalty**: multilevel caches and giving reads priority over writes
- **Reducing the time to hit in the cache**: avoiding address translation .

Simulation and Result

Higher Associativity to Reduce Miss Rate

Assume that higher associativity would increase the clock cycle time as listed as follows:

$$clock\ cycle\ time_{2-way} = 1.36 \times lock\ cycle\ time_{1-way}$$
$$clock\ cycle\ time_{4-way} = 1.44 \times lock\ cycle\ time_{1-way}$$
$$clock\ cycle\ time_{8-way} = 1.52 \times lock\ cycle\ time_{1-way}$$

Assume that the hit time is 1 clock cycle, that the miss penalty for the direct mapped case is 25 clock cycles to a L2 cache (see [9]) that never misses, and that the miss penalty need not be rounded to an integral number of clock cycles. Using Index I for miss rates, for which cache sizes are each of these three statements true?

$$Average\ memory\ accwss\ time_{8-way} < Average\ memory\ accwss\ time_{4-way}$$
$$Average\ memory\ accwss\ time_{4-way} < Average\ memory\ accwss\ time_{1-way}$$
$$Average\ memory\ accwss\ time_{2-way} < Average\ memory\ accwss\ time_{1-way}$$

Average memory access time for each associativity is

$$Average\ memory\ accwss\ time_{8-way} = Hite\ time_{8-way} + Miss\ rate_{8-way} \times Miss\ penalty_{8-way}$$
$$= 1.52 + Miss\ rate_{8-way} \times 25$$
$$Average\ memory\ access\ time_{4-way} = 1.44 + Miss\ rate_{4-way} \times 25$$
$$Average\ memory\ accwss\ time_{2-way} = 1.36 + Miss\ rate_{2-way} \times 25$$
$$Average\ memory\ accwss\ time_{1-way} = 1 + Miss\ rate_{1-way} \times 25$$

the average memory access time for a 4 KB direct-mapped cache is

$$Average\ memory\ accwss\ time_{1-way} = 1 + (0.098 \times 25) = 3.44$$

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

$$Average\ memory\ accwss\ time_{8-way} = 1.52 + (0.006 \times 25) = 1.66$$

Using these formulas and the miss rates [9], Table 1 shows the average memory access time for each cache and associativity.

Table 1: AMAT using miss rates

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

The formulas in this Table 1 contain caches less than or equal to 8 KB for up to a 4-way link. Starting at 16 KB, the larger hit time of the larger thread outweighs the time saved due to reduced errors. Note that we did not take into account the slower clock rate in the rest of the program, and thus understand the direct-mapped cache advantage.

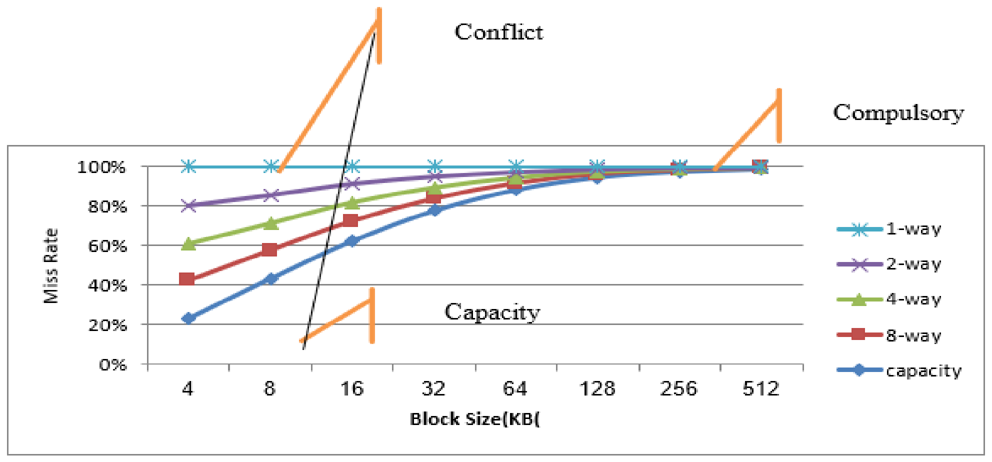


Figure 4: distribution of miss rate for each size cache according to the three C's

conflict misses. Full associativity is expensive in hardware, however, and may slow the processor clock rate, leading to lower overall performance. There is little to be done about capacity except to enlarge the cache. If the upper-level memory is much smaller than what is needed for a program, and a significant percentage of the time is spent moving data between two levels in the hierarchy, the memory hierarchy is said to thrash. Because so many replacements are required, thrashing means the computer runs close to the speed of the lower level memory, or maybe even slower because of the miss overhead. Another approach to improving the three C's is to make blocks larger to reduce the number of compulsory misses, but, as we will see shortly, large blocks can increase other kinds of misses.

Larger Block Size to Reduce Miss Rate

Assume the hit time is 1 clock cycle, hit rate of 95%, independent of block size, then the access time 1 cycle and 15 cycle to send address for a 16-byte block in a 4 KB cache is:

$$\text{AMAT} = \text{Hit time} + \text{Miss rate} \times (Memory\ Access\ Time + Send\ Address\ Time + \left(\frac{Cache\ Size}{Block\ \ Size}\right))$$

$$\text{AMAT} = 1 + 5\% \times \left(1 + 15 + \left(\frac{4}{8}\right)\right) = 1.9 \text{ clock cycles}$$

and for a 256-byte block in a 256 KB cache the AMAT is

$$\text{AMAT} = 1 + 5\% \times \left(1 + 15 + \left(\frac{256}{256}\right)\right) = 1.850 \text{ clock cycles}$$

Table 2: Actual miss rate versus block size for the eight different-sized caches

Block Size						Cache Size (KB)
256	128	64	32	16	8	
5.000	3.400	2.600	2.200	2.000	1.900	4
3.400	2.600	2.200	2.000	1.900	1.850	8
2.600	2.200	2.000	1.900	1.850	1.825	16
2.200	2.000	1.900	1.850	1.825	1.813	32
2.000	1.900	1.850	1.825	1.813	1.806	64
1.900	1.850	1.825	1.813	1.806	1.803	128
1.850	1.825	1.813	1.806	1.803	1.802	256
1.825	1.813	1.806	1.803	1.802	1.801	512

Table 2 shows the average memory access time for all block and cache sizes between these two extremes. The bolded entries show the fastest block size for the given cache size: 32 bytes for 4KB and 64 bytes for the larger cache. These sizes are, in fact, the common block sizes of processor caches today.

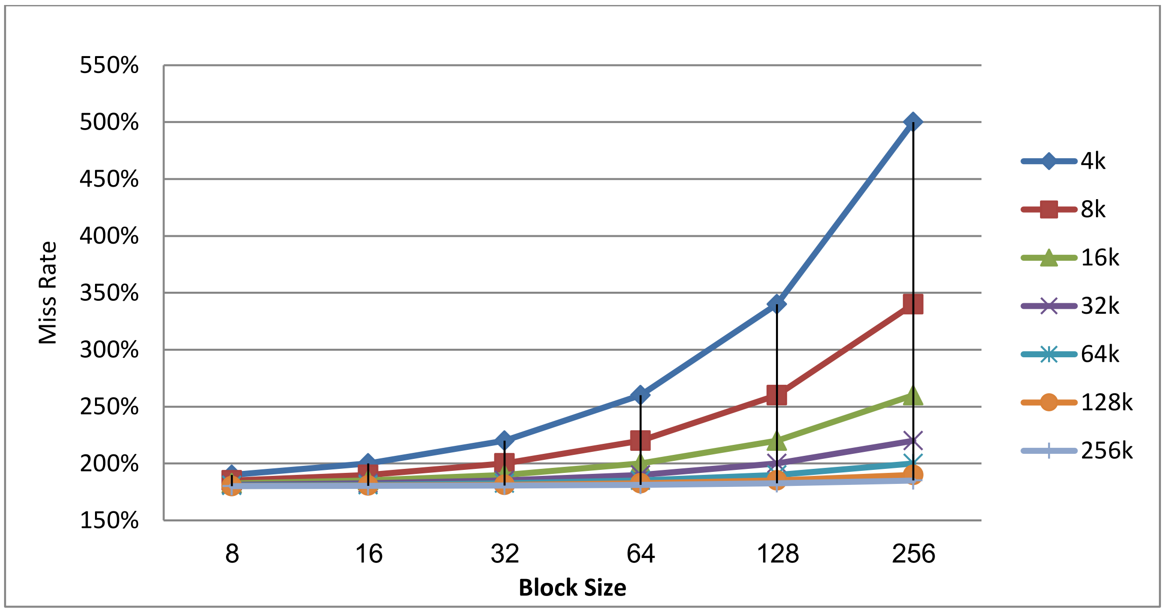


Figure5 : Miss rate versus block size for Eight different-sized caches.

From figure 5 that miss rate actually goes up if the block size is too large relative to the cache size. Each line represents a cache of different size.

Reducing Miss Penalty

A second level (L2) cache reduces the miss penalty by providing a large cache between the first level (L1) cache and RAM, form Eq.(5) then : If a direct mapped cache has a hit rate of 95%, a hit time of $4ns$, and a miss penalty of 100 ns, what is the AMAT?

$$AMAT = Hit\ time + miss\ Rate\ x\ Miss\ penalty = 4 + 0.05x100 = 9ns$$

If an L2 cache is added with a hit time of 20 ns and a hit rate of 50%, what is the new AMAT?

$$\text{AMAT} = \text{Hit time}_{L1} + \text{Miss rate}_{L1} \times \left((\text{Hit time}_{L2} + \text{Miss rate}_{L2}) \times Miss\ Penalty_{L2}\right)$$

$$= 4 + 0.05\text{x}(20 + 0.5\text{x}100) = 7.5\text{ns}$$

Form result then Reduces AMAT form *9 ns* to *7.5 ns* Than as time to replace a block from lower level, including time to replace in CPU

Discussion and analysis:

Table 1 and Figure 4 show how failure rates improve as interconnectivity increases. There are two general rules that can be drawn from these numbers. The first is that the eight-way threaded array is for practical purposes effective in reducing misses in caches of this size as fully associative.

Note that Table 2 and Figure 5 illustrate the difference between determining the block size based on minimizing the miss rate versus minimizing the AMAT . After seeing the positive and negative impact of larger block size on error forcing and capacity, the next two subsections investigate the possibilities of higher capacity and higher correlation.

As in all these techniques, the cache designer attempts to minimize both the miss rate and the miss penalty. Determining the block size depends on the access time and bandwidth of the lower-level memory. High latency and high bandwidth encourage a large block size because the cache gets more bytes per error resulting in a slight increase in the error penalty. Conversely, low latency and low bandwidth encourage smaller block sizes because there is little time saved from a larger block.

Conclusions

In this work, different methods are proposed to an enhancing the performance of cache memory, The proposed methods are discussed in detail to find out the advantages and limitations of each one. Therefore, The main takeaway from this study is that the conflict failure rate can be reduced after taking a larger block size. The techniques to improve hit time, bandwidth, miss penalty, and miss rate generally affect the other components of the AMAT(average memory access) equation as well as the complexity of the cache memory.

References

[1]. Laviza Falak Naz, Techniques to Improve Cache Utilization for a Better Computing Performance Int J Adv Technol, Vol.12 Iss.4 No:8

[2]. Y POORNA CHANDRA and S AFREED, A LITERATURE SURVEY ON CACHE MEMORY, IJARIIE-ISSN(O)-2395-4396, Vol-5 Issue-2 2019

[3]. Sonia, Ahmad Alsharef, and Gaurav Gupta, Cache Memory: An Analysis on Performance Issues, 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom 2021) .

[4]. David Tarnoff, Computer Organization and Design Fundamentals, Published with the assistance of Lulu.com, July 2007,p291

[5]. V. Chaplot, “Virtual Memory Benefits and Uses,” International Journal of Advance Research in Computer Science and Management Studies, vol. 4, no. 9, 2016.

[6] Dimitrios Stiliadis and Anujan Varma,” Selective Victim Caching: A Method to Improve the Performance of Direct-Mapped Caches”, IEEE 1060-3425194 $3.00 , 1994

[7] Hiroyuki Tomiyama and Hiroto Yasuura,” Size-Constrained Code Placement for Cache Miss Rate Reduction”, IEEE Xplore DOI: 10.1109/ISSS.1996.565887, · December 1996.

[8] Ameer Khan,” Brief Overview of Cache Memory”, Technical Report DOI: 10.13140/RG.2.2.22359.21921 ,April 2020.

[9] John L. Hennessy and David A. Patterson ,“Computer Architecture, A Quantitative Approach, Sixth Edition ,2019.