# Cache Memory: An Analysis on Performance Issues

**Conference Paper** · June 2021

DOI: 10.1109/INDIACom51348.2021.00033



**6 authors**, including:

Sonia Garg
University of Delhi
**7** PUBLICATIONS **57** CITATIONS
SEE PROFILE

Syed Rameem Zahra
Indian Institute of Technology Delhi
**22** PUBLICATIONS **58** CITATIONS
SEE PROFILE

Monika Arora
Apeejay Education Society
**22** PUBLICATIONS **47** CITATIONS
SEE PROFILE

# Cache Memory: An Analysis on Performance Issues

**Sonia**
Yogananda School of AI, Computer and Data Sciences
Shoolini University
Solan, India
soniacsit@yahoo.com

**Ahmad Alsharef**
Yogananda School of AI, Computer and Data Sciences
Shoolini University
Solan, India
ahmadalsharef994@gmail.com

**Pankaj Jain**
Department of Management
JJT University
Rajasthan, India
pkjmailbox@gmail.com

**Monika Arora**
Department of Operations
Apeejay School of Management
New Delhi, India
marora.asm@gmail.com

**Syed Rameem Zahra**
National Institute of Technology (NIT)
Srinagar, India
rameemzahra@gmail.com

**Gaurav Gupta**
Yogananda School of AI, Computer and Data Sciences
Shoolini University
Solan, India
solan.gaurav@gmail.com

2021 8th International Conference on Computing for Sustainable Global Development (INDIACom) | 978-93-80544-43-4/21/$31.00 ©2021110.1109/INDIACom51348.2021.00033

***Abstract*—Cache memory is mainly inculcated in systems to overcome the gap created in-between the main memory and CPUs due to their performance issues. Since, the speed of the processors is ever-increasing, so a need arises for a faster speed cache memory that can definitely assist in bridging the gap between the speed of processor and memory. Therefore, this paper proposes architecture circumscribed with three improvement techniques namely victim cache, sub-blocks, and memory bank. These three techniques will be implemented one after other to improve and make the speed and performance of cache comparative to main memory. Moreover, the different variables like miss penalty ratio, access speed of cache and miss rate ratio, which were already in use, are used in this paper to estimate the cache memory performance after implementation of proposed approach. After performance estimation it can be determined that proposed approach at level 1, using Victim Cache technique decreases the rate of misses, at level 2, Sub-blocks Division technique further reduces the penalty ratio of miss rate and then at level 3 Memory Bank Technique is useful in further decreasing memory access time. Thus, using the suggested approach, performance of Cache Memory can be improved several times.**

***Keywords—RAM, Miss Ratio, Access Rate, Hit Ratio, Tags and Addresses, Associative Mapping***

## I. INTRODUCTION

Cache Memory is an important element in the computer that could affect the program execution because its access time is less than the access time of the other memories. It is the fastest component in the memory hierarchy and approaches the speed of CPU components. It works on the property of 'locality of reference' i.e. reference to memory at any given interval of time tends to be confined to a few localized areas in memory. The active instructions of the program in addition to the variables will be stored in the cache memory; this will reduce the total execution time of the program because of reducing the average access memory time.

The concept behind the Cache memory organization is to make the average access time of the memory approaches the average access time of the cache memory by keeping the instructions and data, which are most accessed, in the fast cache memory. Very frequent and large numbers of memory requests are seen in the cache memory although it is much smaller in size as compared to the main memory. Various levels of the cache memory are present in the computer, among them the largest and the slowest cache is being present at the last level. In contrast, the smallest and the fastest cache will be found at first level cache. The level one cache, that is L1, is normally reside inside the processor; however, the level two, L2, and level three, L3, caches are present on separate chips out of the processor. Each processor, in the multi-core processors, will have its personal L1 cache memory and all the cores will be shared by the last level of the cache.

Whenever a word is to be searched, at first the CPU searches primary address of that word in its cache memory. If it is found there a HIT occurs, else a MISS occurs. In that case the word is searched in their main memory and then data sub-part is fetched from its main memory and finally stored in the cache for future reference. Hit Ratio is defined as the ratio between the, number of HITS divided by the sum of number of HITs and the number of Misses. The HIT ratio closer to one can be taken as on:

- If the memory address is accessed first time then Misses will take place.
- If 2 blocks are simultaneously mapped on the same address, Misses could occur because of the inadequate size.
- Misses could occur because of the small size of cache.

The cache MISS rates in addition to the handling time needed for the cache are two main factors which have the major impact on the performance of cache. Victim cache (which is a short-term storage space of line of cache memory, removed from cache) and with it column associative cache can be used to reduce cache miss rate.

Overall, cache memory acts as an connecting point in-between the CPU and slower memory unit, since it can provide data at the fast pace for execution inside memory which has been shown in Fig. 1.

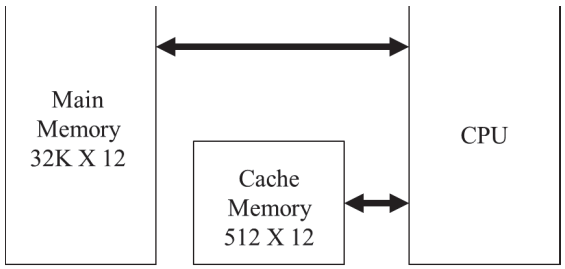


Fig. 1. Location of cache memory

Since cache systems are based on on-chip memory essentials, which can retain data, so it also works like a buffer to act as an intermediate for processors and the main memory of a system. Recent studies propose that for deciding the multi-core systems performance, on-board elements like cache memory performance plays a crucial role. Indeed, cache memory is quicker than main memory and RAM because it is very much nearer to the microprocessor. Thus, it becomes a basic need for transferring data synchronously in-between the processor and main memory. Data storage on cache has substantial advantages in comparison to RAM. One of them is the speed of cache memory for fetching-up the data. However, it consuming more energy as it is on-chip. As we know that speed of the processors are increasing day by day so a dire need arises to increase the speed of cache memory. Probably, the faster cache can assist in narrowing the gap between the processor and memory speed. Thus, in this paper three improvement techniques namely victim cache, sub-blocks, memory bank has been implemented simultaneously to increase the cache speed at each level. These techniques will be implemented one after other to improve and make the speed and performance of cache comparative to main memory. Moreover, the different variables like miss penalty ratio, access speed of cache and miss rate ratio, which were already in use, has been used in this paper to estimate the cache memory performance after implementation of proposed approach.

## II. Background

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

### A. Existing Mapping Techniques

There are 3 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 2. This allows any address in cache to store any word from the main memory. The fifteen-bits address value is represented as a 5 digit octal value and 12 bits word corresponding to it is represented as a 4 digit octal value. Processor address value of fifteen digits is sent to the argument register and then the associative memory will search for the memory address. In case, the address exists in the associative cache memory, the twelve-bit data will be sent from it to the CPU. Otherwise, the order will be sent to the main memory and then it will search for the matching address in the memory. If the address was found, the content pair is then fetched for the associative memory and stored in it.

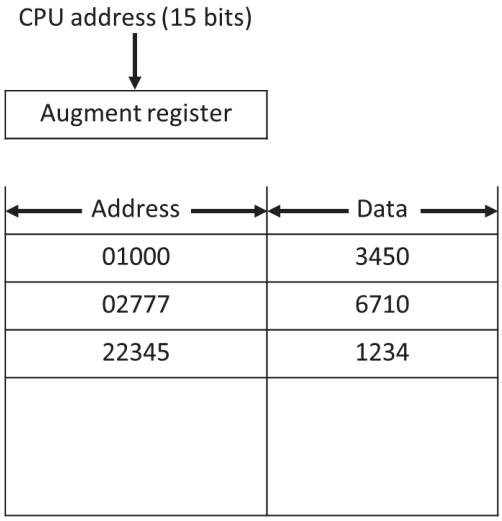


Fig. 2. Associative mapping Cache (all numbers in octal)

#### 2) Direct Mapping

Fig. 3 illustrates the possibility that whether RAM can be used for cache or not. The fifteen bits of processor address will be distributed among 2 parts. The index part will be framed by considering 9 LSB (least Significant Bits) while the tag field will be determined by the six bits. As shown in figure 3, both tags and index parts should be included in the address of the main memory. Here, the bit number of the index field should match the address bit needed to take up the cache memory. Commonly, 2k words are the existed size of cache memory where 2n words will be in main memory. The n bits address of the memory can be distributed among the index field of K-bits and tag field of n-k bits. The n-bits address can be utilized by the direct mapping of cache address is used by the cache direct mapping to access the main memory while the k-bits index address is used to access the cache. Words in cache memory will be organized as shown in Fig. 3, where both data and its associated tag will be combined to form one word. If a new word is first fetched to the cache, the cache bits will be stored together with the data-bits. If a memory request is generated by the CPU, then the cache will be accessed by its address using the index field. After that, tag part of processor address and the tag field of word going to be considered from cache part will be compared to each other in order to fetch the required data if match occurs.

#### 3) Set Associative Mapping

Set associative mapping 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, consider a main memory accommodating

1024 words where each and every word from the cache memory consist of 2 data words. Commonly, K-words of main memory for every word of cache memory can be considered by a set-associative cache of the given set size as demonstrated in Fig. 4.

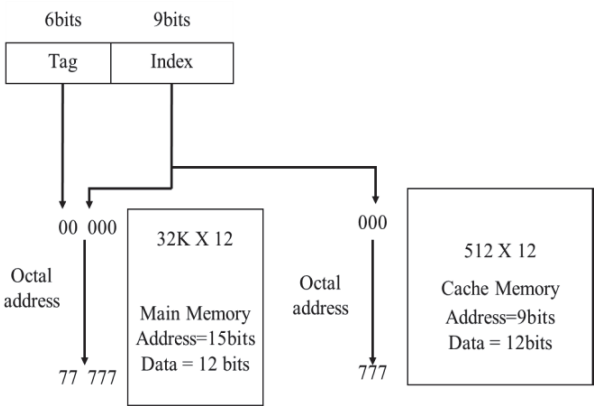


Fig. 3. Illustration of relation between the Main and Cache memory

| Index | Tag | Data | | Tag | Data |
|---|---|---|---|---|---|
| 000 | 01 | 3450 | | 02 | 5670 |
| | | | | | |
| 777 | 02 | 6710 | | 00 | 2340 |

Fig. 4. Two-way set–associative mapping Cache

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 the 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 processed by searching tags associatively in the set in the same manner as we are doing associative memory search. 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.

## III. Literature Review

Various researches have shown how to improve cache performance. Chaplot [1] has proposed an approach that can provide higher associativity to improve cache miss rate and discussed two locality types in time and in space, i.e. temporal locality and spatial locality respectively. In [2] and [3], researchers have proposed automatic self-adaptable Cache memory models for memory systems. The performance analysis of cache memory, in context of replacement policies, is presented in [4] and [5]. K. Westerholz [6] has proposed the implementation of cache designs and efficiency of cache driven memory management to bridge the communication between the CPU and main memory. Divya [7] suggested a very useful graceful code for an efficient virtual memory [7].

In [8] and [9], average access memory time reduction approach is proposed via adding victim cache. Kim and Song detailed out the effect of cache memory on the data storage [10]. Some researchers have discussed cache memory performance with other memories like scratchpad memory, split memory, etc. [11], [12]. Kolanski [13] proposed the usage of Cache memory for mobile and embedded systems. Smith et al. [14] suggested the performance for virtual memory using a page based approach which has been affected by page fault frequency. Moreover, Banday and Khan [15] presented an overview and review on present status of performance in Cache memory. They also discussed about the Cache performance and its adaptability, when system is working over the network.

A summary of related research works, done by the researchers to improve the cache performance, is presented in Table I.

TABLE I. Summary of Related Research Works

| Reference | Title of Paper | Advantages | Gaps |
|---|---|---|---|
| [1] | • Virtual Memory Benefits and Uses | Provides higher associativity in terms of time and space to improve cache miss rate | Not examined average memory access time reduction happened due to his approach. |
| [2], [3] | • A Self-Tuning Cache Architecture for Embedded Systems<br>• Column-associative Caches: a Technique for Reducing the Miss Rate of Direct-mapped Caches | Proposed automatic self-adaptable cache memory models for memory systems | Any measure to improve cache performance has not been suggested. |
| [4], [5] | • An overview of modern cache memory and performance analysis of replacement policies<br>• Implementation of LRU Replacement Policy for Reconfigurable Cache Memory Using FPGA | Analyzed performance of cache memory in context of replacement policies. | Not suggested any improvement approach in terms of replacement policies. |
| [8], [9] | • A Fine-Grained Configurable Cache Architecture for Soft Processors<br>• Cache Memory Simulators: A Comparative Study | Used victim cache and proposed average access memory time reduction technique. | Technique is beneficial however can't improve the cache performance up to a certain extent. |
| [14] | • Modelling and Enhancing Virtual Memory Performance in Logic Simulation | Analyzed and suggested technique to improve the performance of virtual memory that has been affected after using page fault frequency. | Technique alone is not sufficient to improve the performance of cache memory. |

Based upon the tabulated summary and reviewed literature, it is observed that there is need to have a comprehensive model which could provide the improvement in cache memory performance at a place. Therefore, we have suggested an architecture containing all together three improvement techniques namely victim cache, sub-blocks, memory bank. These techniques will be implemented one after other to improve and make the speed and performance of cache comparative to main memory.

## IV. Proposed Architecture for Improving Cache Performance

As discussed in previous sections, there arises a dire need of detailed research in order to bridge the speed gap in-between cache memory and main memory. Thus, in this section, a solution of this problem has been suggested through an architecture.

In this architecture, different techniques have been combined together that extracts out one by one, removing the limitations of previous one. This step by step architecture will increase the speed of cache memory considering all of techniques comprehensively where each one of them individually is helpful in improving the performance of the memory. Further, in the presented paper, different techniques and methodologies has been examined to assess the usefulness of proposed technology in improving performance. The architecture has been illustrated with the flow diagram in Fig. 5, and then elaborated in detail with its implementation.

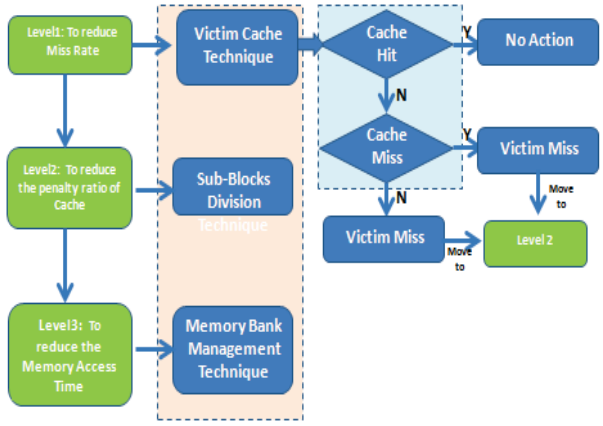


Fig. 5. Proposed architecture for improving Cache performance

*Step 1: Reducing The Miss Rate using Victim Cache*

A victim cache can also be called as hardware cache memory, is used to boost hit latency rate for Direct-mapped cache memories and to reduce the conflict miss rate. This cache can be placed at the refill path of the cache coming at Level 1, so that whenever any cache-line gets thrown out from the cache memory, it will again get cached in the Victim cache. Thus, any kind of data is evicted from the Level 1 cache, then the victim cache will automatically get populated. The victim cache will be considered only when there arises a miss in Level 1. The contents of the matching victim cache line and the Level 1 cache-line will be swapped if the access results are coming out as a hit.

*Implementation of Victim Cache:*

The victim cache behavior in the corresponding interaction with the associated level if cache is described below

Cache Hit: No action

Cache Miss means Victim Hit: In this case, the blocks inside the victim cache and the cache will be swapped. Therefore, the new block which has been stored in victim cache will be considered as the block that has been used most recently.

Cache Miss means Victim Miss: The part from next level will be fetched to cache and the part coming out of cache will get stored in victim cache.

This can be explained through an example: Assume a cache of direct-mapped category containing two blocks A, B referring towards the same set of values. Then it is considered to be linked to a second entry fully associative victim cache having C, D blocks. The path to be considered is as shown in the Figure 6, i.e. A, B, A, B…. and so on.

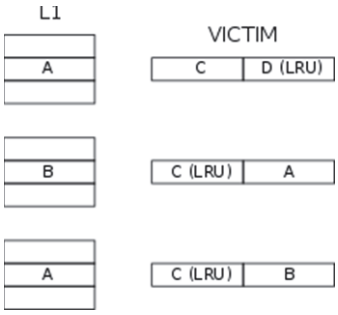


Fig. 6. Implementation example

As shown in the Fig. 6, it can be observed that when a victim cache hit occurs, part of blocks A and B will be swapped and the LRU block of victim cache does not change. Thus, we got an illusion of associatively towards the direct-mapped L1 cache which in contrast decreasing the number of conflict misses.

If there are 2 cache memories, Level1 and Level2 with a particular policy (L2 and L1 do not store similar memory locations twice), L2 behaves like victim cache for L1.

*Step2: Reducing the Penalty Ratio of Cache Miss*

The performance could be affected by tags at which much space is required, and cache speed is decreased. To decrease the required space of optimizing tags (in addition to making them shorter) on the chip, large blocks are used. Due to reducing the necessary misses, the miss rate might be decreased too. However, because the full block should be transferred between cache and other memories, the miss penalty will be high.

To overcome this problem, every block has to be divided into sub-blocks as shown in Fig. 7, each one of them has a valid bit.

Although the tag is valid for the whole block, just a single block has to be read on a miss. So that a smaller miss penalty will be achieved since a block cannot be identified as the minimum unit fetched between cache and memory anymore.

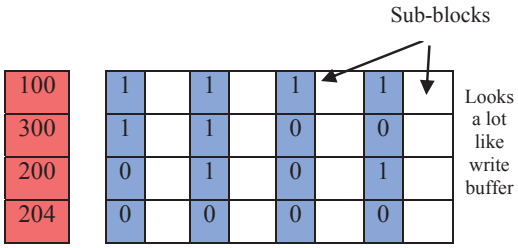


Fig. 7. Dividation of block into sub-blocks

*Step3: Decreasing Memory Access Time*

Memory bank is a hardware-dependent logical unit of storage in electronics. In a computer, the memory controller with physical organization of the slots of the hardware memory, define the memory bank. A memory bank is a partition of cache memory that is addressed sequentially in the entire set of memory banks. For example, assuming that a(n) is data item which is stored in bank (b), the following data item a(n+1) will be stored in bank (b+1). To avoid the bank cycle time impacts, cache memory will be separated into several banks. Therefore, if data is saved or recovered sequentially each bank will have sufficient time to recover before processing the next request of the same bank.

The required number of memory modules to have a similar number of data bits as the bus. A bank can contain many numbers of memory modules as illustrated in Fig. 8.

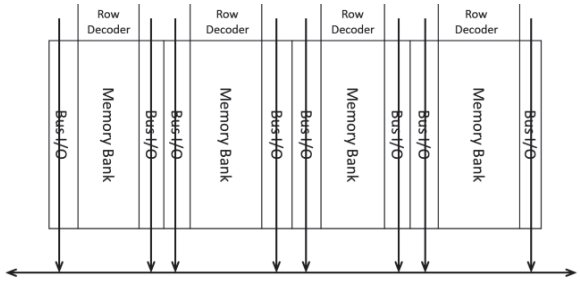


Fig. 8. Memory bank modules

## V. Conclusion and Future Scope

In this work, different methods are proposed to improve the performance of cache memory. The proposed methods are discussed in detail to find out the advantages and limitations of each one. Furthermore, several metrics were used to compare the behaviour of them. Therefore, the main redemptions that we draw out from this study is that the conflict miss rate can be decreased after taking bigger block size, however for that, more cache size is required. Usage of bigger block size can rise penalty ratio of misses, decrease the time of the hit in addition to decreasing the power dissipation. Larger cache results in slow access time and more cost. However, more associatively results in fast access time and consequently less cycle time or lesser number of cycles. Victim Cache always decreases the rate of misses but at a higher cost in contrast to look aside miss cache.

As a future work, implementation of suggested model will be done. Moreover, further improvement of cache performance will be suggested by considering several methods that can predict future access of data and instructions.

## References


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

[2] C. Zhang, F. Vahid, and R. Lysecky, "A Self-Tuning Cache Architecture for Embedded Systems," ACM Transactions on Embedded Computing Systems, vol. 3, no. 2, pp. 407-425, 2004.

[3] A. Agrawal and S. D. Pudar, "Column-associative Caches: a Technique for Reducing the Miss Rate of Direct-mapped Caches," in Proc. of the 20th Annual International Symposium on Computer Architecture, 1993, pp. 179-190.

[4] S. Kumar and P. K. Singh, "An overview of modern cache memory and performance analysis of replacement policies," in Proc. of the IEEE International Conference on Engineering and Technology (ICETECH), 2016, Coimbatore, India, pp. 210-214.

[5] S. S. Omran and I. A. Amory, "Implementation of LRU Replacement Policy for Reconfigurable Cache Memory Using FPGA," in Proc. of the International Conference on Advanced Science and Engineering, 2018, pp. 13-18.

[6] K. Westerholz, S. Honal, J. Plankl, and C. Hafer, "Improving performance by cache driven memory management," in Proc. of the 1st IEEE Symposium on High Performance Computer Architecture, 1995, Raleigh, USA, pp. 234-242.

[7] Y. A. Divya, "An Efficient Virtual Memory using Graceful Code," International Journal of Trend in Scientific Research and Development vol. 3 no. 4, pp. 623-626, 2019.

[8] M. Biglari, K. Barijough, M. Goudarzi, and B. Pourmohseni, "A Fine-Grained Configurable Cache Architecture for Soft Processors," in Proc. of the 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS), 2015.

[9] A. Q. Ahmad, M. Masoud, and S. S. Ismail, "Average Memory Access Time ReductionVia Adding Victim Cache," International Journal of Applied Engineering Research, vol. 11, no. 19, pp. 9767-9771, 2016.

[10] Y. Kim and Y. H. Song, "Impact of processor cache memory on storage performance," in Proc., of the International SoC Design Conference (ISOCC), 2017, Seoul, pp. 304-305.

[11] W. P. Dias and E. Colonese, "Performance Analysis of Cache and Scratchpad Memory in an Embedded High Performance Processor," in Proc. of the 5th International Conference on Information Technology: New Generations, 2008, Las Vegas, USA, pp. 657-661.

[12] K. Singh and S. Khanna, "Split Memory Based Memory Architecture with Single-ended High Speed Sensing Circuit to Improve Cache Memory Performance," in Proc. of the 6th International Conference on Signal Processing and Communication (ICSC), 2020, Noida, India, pp. 188-193.

[13] R. Kolanski, "A logic for virtual memory, "Electronic Notes in Theoretical Computer Science, vol. 217, pp. 61-77, 2008.

[14] S. P. Smith and J. Kuban, "Modelling and Enhancing Virtual Memory Performance in Logic Simulation," in Proc. of the IEEE International Conference on Computer-Aided Design, January 1988, pp. 264-265.

[15] M. T. Banday and M. Khan, "A study of recent advances in cache memories," in Proc. of the International Conference on Contemporary Computing and Informatics (IC3I), 2014, Mysore, India, pp. 398-403.