International Journal of Advancements in Technology

OPEN ACCESS Freely available online

Research Article

# Techniques to Improve Cache Utilization for a Better Computing Performance

**Laviza Falak Naz***

*Department of Software Engineering, NED University of Engineering and Technology, Karachi, Pakistan*

## ABSTRACT

Cache marks tremendous importance in the life of a processor as it contributes to enhancing the performance of a CPU. It is of no use if it's not efficiently and effectively utilized. This study focuses on finding approaches that are helpful for cache utilization in a much organized and systematic way. Multiple tests were implemented to remove the challenges faced during the practical implementation of such techniques. These approaches include Data Alignment, Eager Write back, Access Aware Cache Management, Linear Relaxation Method, and Data-Computation Reorganization.

**Keywords:** Cache utilization; Data computation recovery, Data alignment; Linear relaxation method; Eager writeback-a-technique; Access pattern aware cache management

## INTRODUCTION

The cache is one of the essential elements of the computing devices which have helped in the improvement of the computational capabilities of the computing devices. Modern digital devices have a cache or quick memory access, which contributes to faster computation and better functionality. The caches have contributed the right amount in generating computational disparity among the processor 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 utilization can be defined as ratio of utilizing the cache data lines before eviction, or in other words, the frequency of accessing the data lines in the cache before cache clearance. There are multiple levels of cache in the processing system. Different groups provide different access times and speeds, depending upon the processing requirements of the system. The on-chip cache architectures usually include level one caches on the same processing cheap, with second and third in the other chips. In this way, the dependencies of the cache levels are determined with their distance from the processing unit. However, the size of cache values the most as usually the level one cache has very little space as compared to other levels [2].

There are different techniques available that can be adopted, depending upon the system requirements, architecture, and organizations. In this scenario, the following paper discusses brief details about the methods that can be adopted to enhance cache utilization.

### Background

The reduction in capabilities of the cache also extends the program speed and compile-time, which damages the processors. This might also lead to massive data losses as the inability to provide data accessibility and quick navigation along the stack can skip some essential instructions on the way. The prioritized data, which are often generated with high-value instructions, all of a sudden, are ignored. This can not only be dangerous for the processor but also hampers the leniency of the user experiences [2]. Attributes like memory latency and memory bandwidth are lemmatized. It not only affects the computing power of the system but also affects the capabilities of the CPU by hampering its peak performance attributes [2].

In modern computation devices such as supercomputers and the advanced servers, the caches play a very significant role in determining the speed and efficiency of the computing circuits. Such devices rely on the cache management programs on basic and advanced levels to handle the high number of threads as a single time. Whist multiple threads are not only helpful in determining the computational skills of these computers [3]. Such computation devices are always highly dependent upon the speed and higher computational speeds. Their clock speeds are very high, and the access rates of the computing registers and the stakes are very high. Therefore, the cache plays a vital role in providing a quick access place for the frequent memory controls to the computing devices [4]. About 44% to 99% of the values in the cache are used within one repetitive loop of the clock cycle [4]. This illustrates the value of cache, which helps in the computation of processing systems and has

**Correspondence to:** Laviza Falak Naz, Department of Software Engineering, NED University of Engineering and Technology, Karachi, Pakistan; E-mail: lavizaniazi2001@gmail.com

**Received:** May 03, 2021, **Accepted:** May 10, 2021, **Published:** May 30, 2021

**Citation:** Naz LF, (2021) Techniques to Improve Cache Utilization for a Better Computing Performance. Int J Adv Technol 12:8.

even more functionality to perform in the plans. Cache utilization becomes a value at this place when the functionality of CPU. The multicore architecture in the modern computer systems has also promoted the use of multi-level caches which have a complicated make and access models. In this scenario, the management of cache utilization becomes a significant task while considering the facts of data importance, memory bandwidth, clock speeds, and cache dependencies, the cache utilization holds an important place in the light of having functional computing systems with the least data fluctuations and loses. Cache utilization has been a challenge for modern computer architects as the programs with multiple file handling capabilities often disrupt the data lines in the cache, turning the cache utilization down to a non-negligible extent [5].

Despite considering the value that such programs are highly accepted in customers' markets and have a perfect role in making modern computers. Yet, the damage to the architectural foundations of the systems in terms of reduced cache utilization can be dangerous. In this regard, there are ways and techniques approached to enhance the cache utilization on architecture, software, and network levels.

## MATERIALS AND METHODS

Data alignment strategy is a functional cache utilization improvement strategy that deals with the identification of parameters that are hampering the performance of the cache. It aligns the data attained from different parameters in a serial pattern, allowing the processor to select the appropriate methodology to work on (Figure 1).

```
Algorithm ComputePad
Input:
  Array Dimensions: N' × N; /* Row Size = N */
  Tile Dimensions: Rows × Cols
  Cache Size (in words): CacheSize
  Cache Line Size (in words): LineSize
Output:
  Pad Size for Array: PadSize

if (N mod LineSize == 0) then InitPad = 0
else InitPad = LineSize − N mod LineSize
for PadSize = InitPad to CacheSize step LineSize
  Status = OK
  Initialize LinesArray[i] = 0 for all i
    (0 ≤ i ≤ ⌈CacheSize/LineSize⌉)
  for i = 0 to Rows
    for j = 0 to Cols step Linesize
      k = (i × (N + PadSize) + j) mod CacheSize
      if (LinesArray[k/LineSize] == 1)
      then Status = CONFLICT
      else LinesArray[k/LineSize] = 1
    end for /* loop j */
  end for /* loop i *
  if (Status == OK) then return PadSize
end for /* loop PadSize */
```

**Figure 1:** Data Alignment algorithm for padding size [1].

It is a very flexible model that enables the system to choose data based on their tile sizes to identify the applications to be prioritized [1]. In the result of the enhanced padding, the data alignment technique also allows the data to be transformed into linear arrays.

These arrays are indexed and addressed using quantized indices which allow the system to manage the data with a reference. In this way, the data is not addressed directly, preventing data loss, traversal issues and maintains memory bandwidth [1]. There are various approaches that can be adopted to enhance the tiling process. One of such approaches is looping though reduced self-interference of cache. Although, it has a longer implementation time and requires a good experimental period for successful practice. Yet, there are remarkable memory access patterns received with huge leniency, which can help the systems to be trained for these algorithms, providing them a pathway to work on while handling the caches [6].

The increased management of data lines in the cache of the processor helps to optimize the system for having better control over the data stored in the plans. In the part of the various cache levels, the data alignment model must be implemented individually for each group, while considering all the requirements and resources for each cache chip. With differences in memory sizes and data lines, the implementation of data alignment is also changed [6]. Various expressions are used for the manipulation and calculation of the correct cache data line index while working under the data alignment model. The following algorithm is used for the identification of loop pad size.

### Pros

- Primary advantage of Data Alignment technique is adaptability - without any respect to self-interference clashes, tile measure can be optimized when decoupling from padding phase, comes about in selection of bigger tile estimate which maximize cache utilization
- The other pros are that in many tiled arrays, cross-interferences can be ignored so padding technique can also be applied to greater than two dimensions' arrays

### Cons

- The structure of the data arrays is transformed, and references to the array in the rest of the code has to reflect the new structure [1]
- If the array size is unknown, then the incorporation into the library routine is impossible

### Data and computation reorganization method

Data and computation reorganization at run time applications is one of the popular methods of improving cache performance with manageable implementations. This method includes various sub approaches that can be used collectively or individually to determine the cache accessibility. These methods vary from architectural and on-chip performances to programmed implementation, which has their own purposes and contributions towards the improvement of cache utilization [2].

One of these are methods is runtime computation and data transformation method. This helps to reorder data access for the determination. The process of the locality group is a subcategory that allows the reordering of data in different dynamic arrays. It helps to index the data concerning the neighboring items, which not only helps to enhance the system value but also contributes to boosting the system performance in terms of cache efficiency (Figure 2).

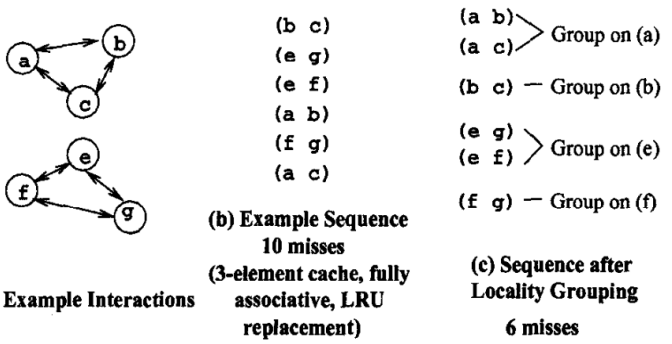


**Figure 2:** Example of logical grouping [2].

It also includes the process of data packaging with dynamic indexes. It helps the systems to create data packages while considering the cache data lines undervalue. There are algorithms applied for the conversion of data arrays into lenient data packages [2]. One such algorithm is mentioned in the figure below.

```
initializing each tag to be false (not packed)
for each object i in the access sequence
    if i has not been packed
       place i in the next available location
       mark its tag to be true (packed)
    end if
end iteration
place the remaining unpacked objects
```

**Figure 3:** Algorithm for dynamic data packaging [2].

The compiler implementation included in the determination of the data reorganization includes approaches such as a program with multiple computational capabilities or multi-thread processing capabilities. Such programs can handle and run the same algorithm simultaneously, enabling full access to the system over the data. Although there are a large number of parameters to consider while understanding and studying the computer implementation of the data reorganization module such as system specifications, clock speeds, simultaneous programs, and others, there are highly reliable and competitive resources to improve cache utilization in the systems [2].

### Pros

- The most important advantage of this technique is that it can be applied to dynamic applications where data access patterns are unknown and may change during the compilation of program
- The locality grouping transformation method results in eliminating 96.9% of cache misses in 2K cache [2]
- Dynamic Data packing methods enhance spatial locality and reduce the number of cache misses specifically in L2 cache by rearranging the data in order

### Cons

- It is not yet established that data reorganizing methods at runtime are cost-effective

## Linear relaxation method

While considering the high-speed computation and processing devices such as supercomputers and the relevant high computation services, which work at exceptionally higher speeds, the value of cache receives an increased flow of value when the memory bytes stored in the cache serves the most critical purpose. The high-speed computation of the supercomputers and servers can grow to a considerable extent if their caches are well managed, and the data assigned to the cache is valuable, in terms of ease of use and frequency. In this scenario, the prospect of improving cache utilization gains a much significant value. The linear relaxation methods are one of those high-speed computational methods which are ideally used in supercomputers, large mainframes, server farms, and similar computation devices that have high data and speed requirements [3].

The linear relaxation method includes sweeping and tiling of cache data lines with algorithms that can traverse across each data line in the same loop and select the data of value. Likewise, it also produces a boost in the data lines but re-indexing and verifying the data stored in each data line. Amazingly, the data stored is traversed at high speed. The liar relaxation method leads to improvements in the core-wise performance of the systems to win their cache. By this, we mean the version of each individual chip-set in each core chip. Since the modern infrastructure is based on multicore architectures, the data is handled in multiple lines at each phase. The figure below represents the cache leniency achieves in multiple cores after the application of linear relaxation methods.

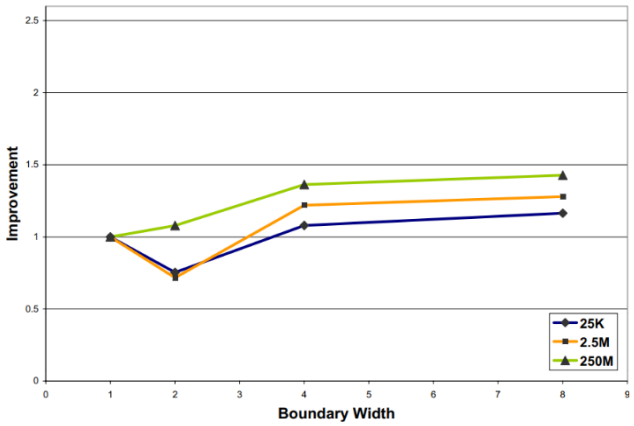


**Figure 4:** Improvement by widening boundaries on 8 processors [3].

The algorithm and programs for the linear relaxation methods are developed in C++ language of computation, providing a broader scope of applicability to the agenda [3]

### Pros

- The system is compatible for multiple platforms including EDG C++, SAGE and other interfaces making it competent and approachable for multiple platforms [3]
- The system handles operations in the preprocessors, avoiding the main stream memory consumption and working faster than other processes [3]

### Cons

- The system needs to be revised for every system. Therefore, it is not convenient and handy [3].
- A new set of code and testing is required before implementing for a new system which itself is a very hectic process [3].

### Eager writeback-a-technique

**Eager Write back technique is used to facilitate bus activities when dirty lines are being traced into the cache [7]. A method used to reduce the bandwidth limitations by providing dirty cache lines only when it is evicted. This approach is used in graphics applications where there is a massive threat of cache misses [8]. The methods that were tested to compare the difference between baseline and eager writeback cycles to remove dirty writeback are as under**

**With injecting memory traffic: Data traffic is injected in the form of blocks onto the memory bus and 3 different bandwidths with higher and lower frequencies (from 400 to 3200 clock cycles) to observe simulations of data. All injections are equally distributed among the simulations (of 3 loops). As the amount of injected bus traffic increases, the eager writeback decreases. Unfortunately, it does not provide a speedup at high frequencies. On the contrary, a rise of 11% in rate was shown with the same bandwidth at lower frequencies. [8]**

**Without injecting memory traffic: The result of these simulations with no dirty writeback shows that the spikes are higher in the baseline case and, ultimately, the execution cycles in all of the 3 loops. Here, the memory bus bandwidth drops to zero and affects the overall performance. Eager Writeback sets the bus idle states with first data cache lines due to which the memory bandwidth gets fully utilized in the third loop. The CPU, as a result, fetches cache requests quickly. As the LSQ is not fully used, instructions can leave the IFQ faster and cycle lost is less [9].**

The Eager Writeback technique solved the issues of load response time and finite memory bandwidth. It mainly turns down the switching time of context by cleaning dirty lines ahead of time from the switched context. It minimizes the miss cache latency and pushes the improved data available in the memory level near [11]. This helps in countering other processors' requests faster than ever and managing tasks quickly, such as write, update, and invalidate protocols to remove traffic coherency. As a result, a significant improvement was observed in overall system performance by its implementation.

#### Pros

- The system includes reduction in the bandwidth limitations. Therefore, the system can also be deployed on systems with small cache functionalities [8]
- The bandwidth is limited to a small finite number which reduces the technical system requirements such as clock speed [8]

#### Cons

- The systems including the writeback deployments includes excessive data loses complaints due to dirty datalinks [9]
- The limitations of the bandwidths also reduce the fetching speeds of the data which can be challenging for the working of supercomputers [8]

### Access pattern aware cache management

General-purpose GPU applications undergo substantial memory bottlenecks because the data cache is small and have to be used across dozens of warps.

To address this issue, the warp throttling technique had been proposed earlier, due to which the active warps competing for cache space are reduced in number. However, APCM suggests that instead of warp-wide cache management, we can use per-load locality type information to improve utilization of GPU L1 data cache since within a GPU kernel, the same type of locality is exhibited consistently across all warps. The kinds of data locality behavior that is displayed by single warp load instruction may be one of the four types:

- Streaming data: data fetched by a warp load is used on only one occasion [10]
- Intra-warp locality: data brought by a warp load is reused on multiple occasions within the same warp [10]
- Inter-warp locality: data fetched by a warp is reused on numerous occasions but across different warps [10]
- A combination of both inter and intra warp locality is displayed by some data [10]

The type of locality of each load instruction is detected by APCM dynamically by utilizing a Monitor Tag Array (MTA), which is a small tag array structure. It monitors the accesses from a single warp. Based on the locality type which is detected, either a cache line brought by a load instruction is protected by APCM, or for a load instruction, the cache is totally bypassed [11].

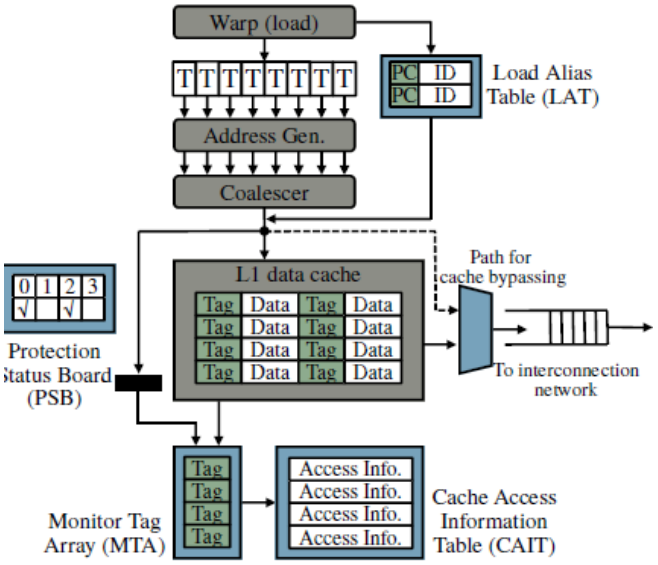


**Figure 5:** Hardware architecture of APCM [11].

The hardware architecture of APCM is shown below. Primarily the L1 data cache access pipelines are modified. For a warp under monitoring, the track of access count is kept by MTA. The type of locality detected per each load instruction is managed by CAIT. PSB keeps information on which warps are presently protecting cache lines. For cache sensitive applications, the performance of GPUs is improved 34% due to APCM. While average improvement is 22% for all types of applications.

#### Pros:

- The APCM provides a minimal solution to the mainstream cache utilization techniques which provide a hardware based system for the issues. It helps to avoid technical coding complexities [11]

- APCM provides an even distribution of the load which helps provide an easier cache handling [11]

**Cons:**

- Multiple threads working simultaneously might lead to a clash of data lines. It can lead to an array of similar indices [11]
- A change occurred in a consecutive data line while another traverse cycle is working can affect the loop and thus, results [11]

## CONCLUSION

We have discussed various techniques that help to solve multiple problems in the utilization of cache memory as it plays a significant part in overall system performance. Since the cache is expensive and its size is small, therefore, it is essential to be able to utilize it to its maximum capacity. Each technique discussed in this paper views the utilization of cache from a different perspective and describes various approaches to maximize cache utilization.

## REFERENCES

1. Panda PR, Nakamura H, Dutt ND, Nicolau A. A data alignment technique for improving cache performance. Proc Int Conf on Comput Des VLSI in Comput Process. 1997;587-592.
2. Ding C, Kennedy K. Improving cache performance in dynamic applications through data and computation reorganization at run time. ACM SIGPLAN Notices. 1999;34:229-41.
3. Bassetti F, Davis K, Marathe M, Quinlan D, Philip B. Improving cache utilization of linear relaxation methods: Theory and practice. ISCOPE. 1999;225-36.
4. Callahan D, Porterfield A. Data cache performance of supercomputer applications. Supercomputing '90: Proc of the 1990 ACM/IEEE Conf on Supercomputing.1990; 564-572.
5. Zahid Y, Khurshid H, Memon ZA. On improving efficiency and utilization of last level cache in multicore systems. Inf Technol Control. 2018;47:588-607.
6. Panda PR, Nakamura H, Dutt ND, Nicolau A. Augmenting loop tiling with data alignment for improved cache performance. IEEE transactions on computers. 1999;48:142-9.
7. Sohi GD, Vaipevam S. Instruction issue logic for high performance, interruptable pipelined processors. Proceedings of the 14th Annual International Symposium on Computer Architecture. 1987;27-34.
8. Lee HH, Tyson GS, Farrens MK. Eager writeback-a technique for improving bandwidth utilization. Proceedings of the 33rd annual ACM/IEEE International Symposium on Microarchitecture. 2000;11-21.
9. Ghosh M, Lee HH. Smart refresh: An enhanced memory controller design for reducing energy in conventional and 3D die-stacked DRAMs. 40th Annual IEEE/ACM International Symposium On Microarchitecture (MICRO 2007). 2007;134-145.
10. Lee CJ, Narasiman V, Ebrahimi E, Mutlu O, Patt YN. DRAM-aware last-level cache writeback: Reducing write-caused interference in memory systems. 2010.
11. Koo G, Oh Y, Ro WW, Annavaram M. Access pattern-aware cache management for improving data utilization in GPU. Proceedings of the 44th Annual International Symposium on Computer Architecture. 2017;307-319.