Department of Applied Mathematics

Computer Organization and Architecture
MC206

Project Review Report

Delhi Technological University

Topic:

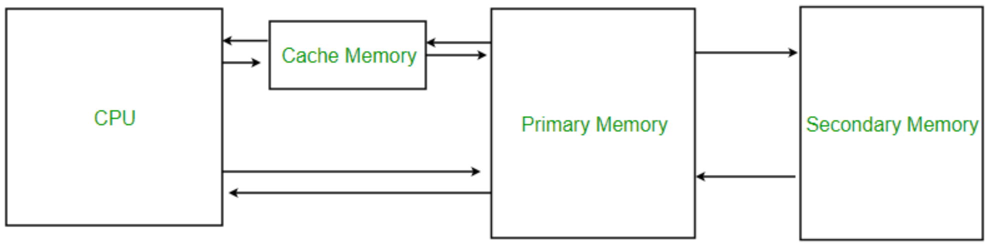
Cache Memory: Performance Issues

Project Supervisor –
Ms. Sumedha Seniaray

By –
Aneesh Panchal – 2K20/MC/21
Ayushi Sagar – 2K20/MC/35

Project Introduction

Till the date we have analyzed these research paper.
First one is **"CacheMemory: An Analysis on Performance Issue"** and,
Second is **"Cache Memory: An Analysis on Optimization Techniques"**.
Through these research papers we are trying to figure out what can be the best techniques and algorithms so that we can improve the performance of cache memory.



Cache Memory: An Analysis on Performance Issue

Cache Memory is a very high speed memory that acts as a buffer between RAM and the CPU. It is used to speed up and synchronizing with high speed CPU. It is used to reduce the average time to access data from the main memory. Cache Memory is designed to combine the memory access time of expensive, high speed memory combined with large memory size of less expensive, lower speed memory. It 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. There is relatively large and slow main memory together with a smaller, fastest cache memory. The cache contains a copy of portions of main memory. When a processor attempts to read a word of memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the processor. If not, a block main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor.

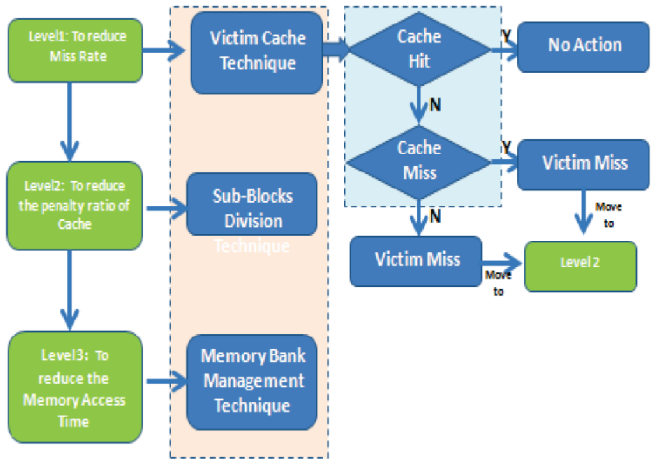
It works on the property of 'locality of reference' i.e. when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that there will be future references to the same memory location or to the other words in the block.

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. 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. Overall, cache memory acts as a connecting point in between the CPU and slower memory unit, since it can provide data at the fast pace for execution inside memory.

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.

In this paper three improvement techniques had been implemented simultaneously victim cache, sub-blocks, memory bank 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, had been used in this paper to estimate the cache memory performance after implementation of proposed approach.


Level1: To reduce Miss Rate
Victim Cache Technique
Cache Hit
Y
No Action
N
Cache Miss
Y
Victim Miss
Move to
N
Level2: To reduce the penalty ratio of Cache
Sub-Blocks Division Technique
Victim Miss
Move to
Level 2
Level3: To reduce the Memory Access Time
Memory Bank Management Technique

Step 1: Reducing The Miss Rate using Victim Cache
A victim cache also known as hardware cache memory, is used to boost hit latency rate for Direct-mapped cache memories and to reduce the conflict miss rate.

Implementation of Victim Cache:
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.

Step 2: Reducing the Penalty Ratio of Cache Miss
For the tags much space is required as a result the performance could be affected and hence cache speed is decreased. So, we need to decrease the required space for optimizing tags 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, 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.

Step 3: 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.

Conclusion:

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. 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.

Cache Memory: An Analysis on Optimization Techniques

Processor speed is increasing at a very fast rate comparing to the access latency of the main memory. By using cache memory in effective manner, the effect of this gap can be reduced. This paper will discuss how to improve the performance of cache based on miss rate, hit rates, latency, efficiency, and cost.

For improving overall performance of the computer, the data or instructions that get used very frequently are kept in cache so that it can be accessed at a very fast rate. The multiple level of cache having last level being largest and slowest, to the first level being fastest and smallest.

In most of the processor first level cache (L1) reside in the processor and second level cache (L2) and third level cache (L3) are on separate chip. In multi-core processors, each processor has its own L1 cache. While last level cache is being shared by all the cores.

The clock of the processor is several hundred times faster than the access latency of main memory. Cache provides the service to reduce this gap and make the performance of system better.

Cache miss rate and miss penalty are the two major factors which effect cache performance. Time required to handle a cache miss is called cache penalty.

There are numerous methods to reduce cache miss rates which include victim cache which is a location for temporary storage of cache line which is abolished from cache.

One way of improving cache performance is to predict future access of data or instructions that need to be replaced from cache. When cache miss occurs data will be fetched from main memory and then it will be decided which data will be removed from the cache in order to place newly fetched data. For this purpose, we discuss performance of different algorithms for replacement methods.

We will discuss the performance of Least Recently Plus Five Least Frequently Replacement Policy and Blocked algorithm which reduces cache misses. The paper discusses cache structure, cache mapping techniques, strategies to reduce cache misses, techniques for avoiding cache penalties and attaining higher cache hit rates.

Replacement Algorithms

There are different replacement polices which are used to reduce miss rates and make cache performance better. Few of them are Least Recent Used (LRU), Least Frequently used (LFU) and few others.

One of the proposed policy is Least Recently Plus Five Least Frequently (LR + 5LF) combines LRU and LFU which reduces cache miss rates more than LRU and LFU. In this policy, specific values are given to each block which coincides with LRU and LFU policies. Then these values are combined by an algorithm to give an overall value to the block.

Cache Algorithms

Cache-oblivious algorithm is used in CPU caches. It has information neither of cache size nor of length of cache line. Because of the fact that it does not have any information about parameters of cache so it will perform better on any size of caches. Cache-aware iterative algorithm is also used to improve performance. The design of replacement algorithms such as LRU is modified which picks number of cache lines that are not used recently for replacement. Before removing these cache lines one more check is placed on them which checks cache line for least frequently used out of those selected cache lines and then that with lowest frequency will be evicted or replaced. This algorithm has better performance than LRU.

Design Based Optimization

One way of reducing the gap between CPU cycle and memory latency is to use a multi-level cache.

We can reduce Misses of first level by introducing second level cache. Also, a user level technique had been implemented which is User Level Cache Control (ULCC) through which user can control the allocation of space in cache in their programs and hence reduce cache pollution. A remapping module is used to allocate cache space by remapping between virtual and physical pages. This technique is very complex to implement and using this method produces lesser hit rates. Different methods are used to make cache's performance better. One of them is to make different banks of cache that can be accessed at different latencies.

Compiler Based Optimization

One of the cache optimization techniques which was used in the past is to optimize the loops through compiler. Loops are reduced to smaller size so that accessed data can be set in the cache. Then compiler will analyse all the task which will share same data, those tasks will be assigned to one processor. In this way all the tasks will be executed consecutively which will use same data from cache.

Prediction Based Optimization

A set of compiler algorithms have been developed which does the prediction about the data to be reused in near future. These predications are used to improve the hit rates. The algorithm used for the purpose is **evict-me** which uses cache line tag of one bit. So, when ever evict-me tag is set for any cache line, the cache line will be replaced. This Technique is complex and consume more energy.

Web Based Optimization

The World Wide Web provides access to large shared data objects. Server overloading is a major problem faced by web users. As Internet usage is increasing at a very rapid rate its bandwidth is likely to suffer in near future. Mostly used objects are cached to locations closed to the clients. Perfecting, cache placement and replacement, cache coherency, cache contents, user access patterns, load balancing, proxy placement and dynamic data caching are few of the methods used for web cache optimization.

Conclusion:

In this paper, we discussed and evaluated the performance of different techniques and algorithms used for cache optimization. Through paper we can believe that it is hard to identify a certain cache optimization technique as the best choice in all the above cases.

Each technique is associated with its design constraints, advantages and limitations. For example, the rate of conflict misses is reduced by using larger block size, larger cache and way prediction methods. However, using larger block size may increase miss penalty, reduced hit time and power consumption. On the other side, larger cache produces slow access time and high cost. It is associated with cache coherence problem. Higher associativity produces fast access time but they have low cycle time. Victim cache reduces miss rate at a high cost comparing to Cache miss look aside. LR + 5LF is a very good technique for reducing cache misses at a high rate comparing to some of the other techniques like LRU, LFU. However, it is more complex than all of these methods. Cache pollution is eliminated by ULCC and also, they have fast access time but complexity is high. Miss penalty is reduced by using pipelined cache which is very complex method.

Date:

March 6, 2022