ISSN No. 0976-5697

Volume 8, No. 3, March – April 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Realisation of Cache Optimisation using New Technique

Komal Agrahari
Dept. of Computer Science and Engg
MMMUT Gorakhpur, Uttar
Pradesh komal.edu16@gmail.com

Dr.P.K.Singh
Dept. of Computer Science and Engg
MMMUT Gorakhpur, Uttar Pradesh
topksingh@gmail.com

Abstract: An impressive design of cache memory is an important view in computer architecture to boost or increase the system performance. In this task we study the response of reducing the cache performance to the cache address on the performance experimentally. Since cache miss penalty has direct contact on the system's performance. To overtake from this, we proposed a new model in which we use revisited second chance FIFO (RS FIFO), which uses partial comparator and use multiple search to increase cache hit rate, by reducing cache miss rate. The Above proposed technique gives maximum hit rate when using revisited SFIFO rather than FIFO.

Keywords: cache mapping technique; hit rate; miss rate; cache optimisation; cache replacement policies.

I. INTRODUCTION

In this paper we focused on new cache replacement policy which reduces the number of misses in the cache memory. After evaluating the cache replacement policy we found that how the speed of processor is growing and so we can say that processor's performance is very much dependent on the number of miss or miss rate.

Modern computers are very quickly grows due to technological growth in CPU memory access method and capacity of processing. While the processor to synchronise the inconsistency between the fastest processor to the slowest memory components at a reasonable or acceptable amount we introduce the cache memory. Cache memory is an intermediate between CPU and main memory that retains copies of recently used information from main memory. The processor runs at its high clock rate only when the piece of information that requires are present in the cache. So the overall average access time can be minimised [2]. The main focus of this dissertation is to realize the cache performance using the new technique of cache replacement policy. In the previous work FIFO is used in the sub blocks of cache memory to improve the performance of the cache [1]. Now after analyzing lots of result we conclude that the proposed policy has lower number of misses in comparison to existing policy. And we know that miss rate has very important role in the performance of the cache.This paper addresses the enhancement of cache performance by applying optimisation techniques [4].

The structure of the paper is as follows:

In Section1 we introduce the theory of cache memory and existing policy. In Section2 we present our proposed work with the steps which is followed by us to achieve the objective. In section 3 we present the case study and analyse the performance of calculated method with respect to the conventional method. In section4 we conclude the paper. And at last we go for various references which I gone through.

A. Replacement Algorithms

When the cache has been filled and then a new block is brought into the cache, one of the existing blocks must be replaced. For direct mapping, there is only one line for each particular block, and no choice is possible. For the set associative and an associative technique [3], a replacement algorithm is needed. To gain high speed, such an algorithm must be implemented in hardware.

- First-In-First-Out (FIFO)

The very simple page replacement algorithm is a FIFO algorithm (first in first out). The first-in, first- out (FIFO) [1] page replacement algorithm has less overhead that entails little book-keeping on the operating system part [4]. The idea is taken from the operating system that keeps track of each page in memory in a queue, with the latest arrival at the back, and the earliest arrival in front. The operating system sustains a list of all pages which is presently in memory, with that page which is at the head of the list the oldest one and the page at the tail the most topical arrival. When a page required to be swapped, the page at the front of the queue (the oldest page) is considered. Since FIFO is cheap and instinctive, it results poorly in practical application.

- Optimal Page Replacement

One result of the discovery of Belady's anomaly was the search for an optimal page-replacement algorithm which has the lowest page-fault rate of all algorithms and will never suffers from Belady's anomaly [1]. This is an algorithm which is exist and it is called as OPT or MIN. It is simply: Replace the page that will not be used for the longest period of time.

- Least-Recently-used (LRU)

This algorithm replaces the page that has not been used for the longest period of time. We can think of this strategy as the optimal page-replacement algorithm looking backward in time, rather than forward [4]. FIFO algorithm takes time when a block was brought into memory, whereas the OPT algorithm takes the time when a block is to be used. The LRU policy is often used as a block-replacement algorithm and is considered to be good. Now the question arises how the implement of LRU replacement is done. An LRU cache block-replacement algorithm may require substantial hardware assistance. The problem is to determine an order for the cache lines which is defined by the time of last use [5].

- Clock Algorithm

In CLOCK, all cache line is to be arranged in form of a circular list that seems like a clock. A pointer i.e. a hand on the clock indicates which cache blocks is to be replaced next. When a block is needed, the pointer advances until it finds a cache block with a 0 reference bit. As the approach, it clears the reference bits. Once a victim block is found, the block is replaced, and the new block is inserted in the circular queue in that position [2]. The worst case shows that if all reference bits are set, the hand cycles through the whole queue has given a second chance to each block. And before selecting the next block for replacement it clears all the reference bits. If all bits are set then Second-chance replacement degenerates to FIFO replacement [2].

- Second-Chance Algorithm

The extended algorithm of FIFO replacement is a second-chance replacement algorithm. When a block has been selected, we inspect its reference bit. We go to replace this block if the value is 0, but if the reference bit is set to 1then we give the block as second chance and move on to select the next FIFO block. When a block gets a second chance, its reference bit is cleared, and its arrival time is reset to the current time. Thus, that second chance block will not be replaced until all other block have been replaced or given second chances. In addition, if a block is used often enough to keep its reference bit set, it will never be replaced [1].

II. PROPOSED WORK

In this work we make some changes in the method of SFIFO and we get better performance in terms of reduce miss rate as compared to existing replacement policy. The main focus of this dissertation is to realize the cache performance using the new technique of cache replacement policy. In the previous work FIFO is used in the sub blocks of cache memory to improve the performance of the cache. Now after analyzing lots of result we conclude that the proposed policy has lower number of misses in comparison to existing policy. And we know that miss rate has very important role in the performance of the cache.

A. Steps Follows In Our New Policy (Revisited Sfifo)

- We generate the memory reference using randomize function.
- Maintain a table and associate counter with each calling memory reference. Parallel we fit it into the corresponding 'set'
  According to $\boxed{\text{A mod S = i}}$
- If the new reference is called then its counter value will be zero (0).
- If the duplicate reference is called then its counter value will increment by one (1).
- If the corresponding 'set' is full and we have to replace the element then which element goes into replacement are decided with the following:-
- If the counter value is zero then go in FIFO manner.
- If the counter value is greater than zero (and it is same for all) then also go in FIFO manner and the corresponding counter value will become zero.
- If the counter value is different for all the element in the set that mean it is a case of second chance , then firstly we replace the new reference with that reference whose counter value is less in the particular set.
- Now we decrement the counter value of the second chance element by 1.

B. Experimental result analysis:

For the implementation of the proposed cache replacement policy, here we used a high level language which is JAVA. Now firstly we have to generate a various random number as memory reference which is used for the evaluation of the cache performance.
Here we are showing our results in two categories which are based on size of random number sets.
Firstly we go through size of 100 random number after that we go for size of 200 random number which is compared on the number of misses occur in proposed replacement policy and the existing policy.

- Case 1 Firstly we generate random number of size 100 using Random function. On the basis of these random numbers we analyze these results and maintain in the form of table.

Table I. Comparisons among FIFO, SFIFO and Revisited SFIFO

Sr. No.	*Number of Set*	*Number of misses in FIFO*	*Number of misses in SFIFO*	*Number of misses in Revisited SFIFO*
1	Set 1	97	97	96
2	Set 2	98	97	97
3	Set 3	97	97	96
4	Set 4	96	96	95
5	Set 5	94	93	93
6	Set 6	96	95	95
7	Set 7	97	97	94
8	Set 8	94	93	90
9	Set 9	95	94	92
10	Set 10	97	97	96

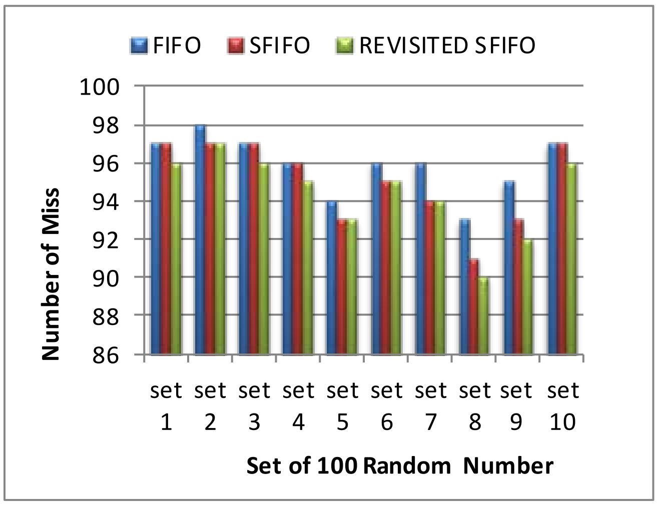

Figure 1. Comparisions among FIFO, SFIFO and Revisited SFIFO

The above algorithm which is used in our proposed replacement policy uses the combination on set associativity as well as the advance version of second chance FIFO and the basic concept of FIFO (first-in-first-out).

- Case 2 we generate random number of size 200 using Random function. On the basis of these random numbers we analyze these results and maintain in the form of table.

Table II. Comparisons among FIFO, SFIFO and Revisited SFIFO

Sr. No.	*Number of Set*	*Number of misses in FIFO*	*Number of misses in SFIFO*	*Number of misses in Revisited SFIFO*
1	Set 1	167	166	156
2	Set 2	165	164	158
3	Set 3	146	145	142
4	Set 4	156	155	149
5	Set 5	165	164	152
6	Set 6	169	165	161
7	Set 7	162	161	153
8	Set 8	166	165	155
9	Set 9	159	155	151
10	Set 10	156	155	150

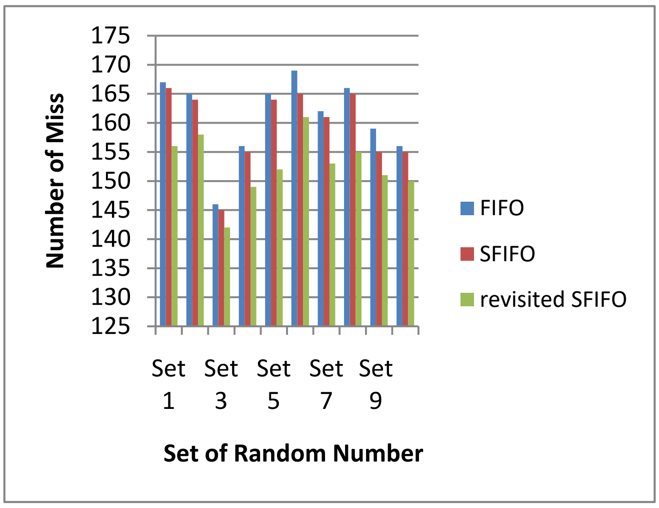

Figure 1. Comparisions among FIFO, SFIFO and Revisited SFIFO

III. Conclusion

From the above discussion of both the table I and table III and based on the experimental results we can conclude that our proposed method (revisited SFIFO) has equivalent or better performance than the existing policies. Now we can say that the revisited SFIFO will be used for the future reference and give contribution in the field of cache performance.

IV. Acknowledgment

Concentration, dedication, hard work and applications are essential but not the only factors to achieve the desired goals. It must be supplemented with guidance and cooperation of people to make it a success. Many people have given their precious ideas and valuable time to enable us to carry on the research paper. We are deeply indebted to all of them for their ideas and suggestions. We offer our sincere thanks to our dissertation guide **Dr.P.K. Singh**.

V. REFERENCES

[1]. Oluleye Olorode , Mehrdad Nourani 2015 "Improving Performance in Sub-Block Caches with Optimized Replacement Policies" ACM J. Emerg. Technol. Comput. Syst. 11, 4, Article 41 (April 2015).
[2]. W. Stallings, "Computer Organisation and Architecture," Eighth Edition, 2011.
[3]. Michail-Antisthenis I. Tsompanas, Christoforos Kachris, and Georgios Ch. Sirakoulis. 2016. "Modeling cache memory utilization on multicore using common pool resource game on cellular automata" ACM Trans. Model. Comput. Simul. 26, 3, Article 21 (January 2016).
[4]. Nathan Beckmann, Daniel Sanchez, "Modeling Cache Performance beyond LRU" vol.no. 978-1-4673-9211-2/16 ©2016 IEEE.
[5]. Mainak Chaudhuri, "Pseudo-LIFO: The Foundation of a New Family of Replacement Policies for Last-level Cache" ACM 978-1-60558-798-1/09/12.