International Conference on Recent Innovations in Science and Engineering
Buddha Institute of Technology, Gida, Gorakhpur (India) (ICRISE-18)
1st-2nd April 2018 www.conferenceworld.in ISBN : 978-93-87793-15-6

# A Comparative Analysis of Cache Replacement Policy

**Komal Agrahari[1], Muzammil Hassan (Asst. Prof CSED)[2], Manish Kumar Verma[3]**

[1,2,3] *Dept. of Computer Science and Engg , MMM University of Technology , (India)*

**ABSTRACT**

*In the memory hierarchy CPU register is closest to the processor followed by cache, main memory and secondary memory. The main goal of this survey paper is to learn the performance analysis of various cache replacement techniques using simulator over the several benchmark on the basis of miss rate, hit rate, bus traffic and block transfer.*

***Keywords: Access Time, Cache Mapping Technique, Cost, Hit Time, Miss Rate;***

## I.INTRODUCTION

Memory progressive system in current time computer is shaped by keeping registers then cache on the processor and virtual memory on hard disk. In present day years different methodologies have been made to enhance the cache memory execution based on hit rate, idleness, replacement approaches, speed and utilization of energy. Replacement strategy of the cache is one of the critical plan parameter which influences the general processor execution and further turns out to be more vital with late mechanical moves towards exceptionally affiliated cache. Memory is principally of two sorts:

**1.1 Internal Memory** − cache memory and primary/main memory

**1.2 External Memory** − magnetic disk / optical disk etc.

Cache memory is a mediator between CPU and fundamental memory, to adjust the execution hole between them. Cache is a speediest semiconductor which goes about as a support and caches some data. Cache memory is utilized to give memory speed and in the meantime it gives a huge memory measure at the cost of lower cost sorts of semiconductor memories. At the point when the processor endeavors to peruse an expression of memory, a check is made to determine if the word is in the cache. Assuming this is the case, the word is sends to the processor. If not, a piece of principle memory is perused into the cache and after that the word is sending to the processor.

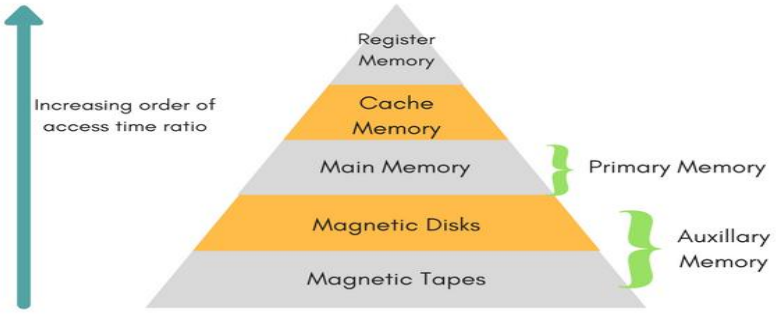


**Fig.1.Memory Hierarchy [6]**

### 1.3 Types of cache

**1.3.1 Splitted caches:** This refers to have a special or separate cache for data and machine instruction. Split cache structure has instruction and data caches separately. When the size of unified cache is divided into two parts, it is known as size of split cache [2].

**1.3.2 Unified caches:** This refers to a single cache for both data and machine instruction. Unified cache has higher hit rate than split cache because it adjusted the load between instruction and data fetches automatically [2]. At the point when a piece that is occupant in the cache is to be changed, there are two cases:-

**1.3.2.1 Write through-** All the compose tasks are made to primary memory and additionally to the cache, protecting that fundamental memory is constantly legitimate. Principle impediments it produces generous memory movement and it might make a littleneck.

**1.3.2.2 Write back-** It is utilized to low memory composes, with compose back, refreshes are made just in the cache. At the point when a refresh happens, a grimy piece related with line is set. At that point, when a piece is transformed, it is composed back to principle memory if and just if the grimy piece is set.

## II.MAPPING TECHNIQUES

### 2.1 Direct Mapping

The direct mapping is the very simplest technique in which we maps each block of main memory into only one possible cache line. This mapping can be expressed as given formula:

$$L = K \text{ modulo } N$$

Where

L cache line number

N number of lines in the cache

K main memory block number

### 2.2 Set-Associative Mapping

Set-associative mapping is a way to cut down the drawback of both direct mapping and associative mapping. In this mapping, the cache consists of a number sets, each of which consists of a number of lines. Thus the relationships are defined here as

$$A = B \text{ divides } C$$

$$I = J \text{ modulo } A$$

Where

B number of way

I cache set number

C number of lines in each set

A number of set

J main memory block number

## III.RELATED WORK

### 3.1 Mechanism to Cut Down Miss Rate of Cache [2]

Hardware based optimization techniques to cut down cache miss rate are:

#### 2.1.1 Use Large Size of Cache

Extensive cache estimate lessens the limit misses [4]. In bigger cache estimate there is constrained possibility that there will be struggle yet the impediment is bigger hit time and higher cost.

#### 2.1.2 Impact of Higher Associativity

Higher associativity diminishes strife misses and comes at the expanded cost of hit time. In for all intents and purposes, comes about demonstrate that 8-way set acquainted cache has, for the most part same miss rate as completely cache. Coordinate mapped cache of size N has around same miss rate as 2-way set affiliated cache of size N/2. This perception called 2:1 cache general guideline and held for cache sizes under 128 KB.

#### 2.1.3 Use Larger Blocks of Cache

Using large blocks is a simplified method to cut down the compulsory misses because larger blocks take advantage of spatial locality [1].

#### 2.1.4 Compiler Optimization

Compiler enhancement is a strategy to low miss rates with no equipment change. Enormous execution hole amongst processor and principle memory has enlivened compiler originators to examination the memory pecking order to check assemble time advancement can enhance execution. So the exploration is separated between developments in direction misses and in information misses. Converging of Arrays, Interchanging of circles, Loop Fusion and Blocking are enhancements found in different current compilers

#### 2.1.5 Impact of Way Prediction and Pseudo Associative Cache

Way Prediction is that where additional bits are put into the cache for expectation of the arrangement of next cache get to. Here, the multiplexer is set to choose the required piece and just a single label examination while getting to cache. Pseudo Associative caches are otherwise called segment affiliated cache. In this it is legitimately isolated into two areas. For each visit Pseudo Associative cache will go about as immediate mapped in first district so each piece has just a single place to discover in cache. If there should be an occurrence of hit this cache is like the direct mapped cache. If there should a rise an occurrence of miss, CPU will visit a predefined area in another zone and if cache hits this time a pseudo hit happens and afterward the piece is swapped for the square of the primary passage.

### *2.2* Mechanism to Cut Down Hit Time of Cache [2]

#### 2.2.1 High Bandwidth Using Pipeline Cache Access

This strategy for improvement is simply to pipeline cache get to with the goal that successful dormancy of cache level one for hit can be various clock cycles, giving quick clock process duration and high data transmission yet less hits. For instance, the pipeline for Pentium Processor took one clock cycle to get to guideline cache, for Pentium Pro through Pentium III it took clock cycles of two and for Pentium IV it took clock cycles of four.

This division raises the phases of pipeline, prompting high punishment on miss anticipated branches and more tickers between stack issue and the utilized information.

**3.2.2 Small Cache**

The file segment of deliver to peruse the label memory and afterward correlation with address is tedious region of cache hit. So little cache size can be quicker and enable the hit to rate. In any case, for second level cache, it is hard to keep little size of cache is sufficient to fit on an indistinguishable chip from the processor to remove the time punishment of off-chip. Thus, for bring down level caches some outline strike a trade off by keeping labels on chip and information off-chip, which gives a quick label check and give bigger ability to isolate memory chips.

3.2.3 **Trace Cache**

To find bunches of guideline level parallelism, it is additionally a test to discover enough direction each cycle without utilizing conditions. The Trace cache is a guideline cache in processor that keeps dynamic direction arrangements after they have been gotten and executed. With a specific end goal to take after directions at consequent circumstances, there is no compelling reason to go customary cache or memory for a similar guideline succession. The principle preferred standpoint of follow cache is that it diminishes the required bring data transfer capacity on handling pipeline.

**3.3 Replacement Algorithms [3]**

The Once the cache has been filled, when another piece is brought into the cache, one of the current squares must be supplanted. For coordinate mapping, there is just a single conceivable line for a specific square, and no decision is conceivable. For the affiliated and set cooperative systems, a replacement calculation is required. To accomplish rapid, such a calculation must be actualized in equipment. Various calculations have been attempted.

**3.3.1 First-In-First-Out (FIFO)**

The easiest page-replacement calculation is a FIFO calculation (first in first out). The first-in, first-out (FIFO) [5] page replacement calculation is a less-overhead calculation that involves little accounting with respect to the working framework. The thought is evident from the name - the working framework monitors each page in memory in a line, with the most recent landing in the back, and the soonest entry in front. The working framework supports a rundown of all pages directly in memory, with that page which is at the leader of the rundown the most established one and the page at the tail the most topical entry.

At the point when a page should be swapped, the page at the front of the line (the most established page) is considered. While FIFO is shabby and natural, it comes about inadequately in down to earth application.

**3.3.2 Least-Recently-used (LRU)**

This calculation replaces the page that has not been utilized for the longest timeframe. We can think about this technique as the ideal page-replacement calculation looking in cache in time, instead of forward [1]. FIFO calculation utilizes the time when a page was brought into memory, though the OPT calculation utilizes the time when a page is to be utilized. In the event that we utilize the current past as an estimation of the not so distant future, at that point we can supplant the page that has not been utilized for the longest timeframe. The LRU approach is frequently utilized as a page-replacement calculation and is thought to be great. The real issue is the

manner by which to execute LRU replacement. A LRU page-replacement calculation may need considerable equipment help. The issue is to decide a request for the edges characterized when of last utilize.

### 3.3.3 Optimal Page Replacement

One consequence of the disclosure of Belady's oddity was the look for an ideal page-replacement calculation which has the most minimal page-blame rate of all calculations and will never experience the ill effects of Belady's inconsistency [1]. Such a calculation does exist and has been called OPT or MIN. It is basically: Replace the page that won't be utilized for the longest timeframe. Utilization of this page-replacement calculation ensures the least conceivable page blame rate for a settled number of edges.

| | **Technique** | **Miss Rate** | **Access Time** | **Hit Time** | **Cost** |
|---|---|---|---|---|---|
| **Miss Rate** | Larger Size of block | low | low | Less | NOT DEFINED |
| | Higher Associativity | low | high | High | High |
| | Larger size of caches | low | low | High | High |
| | Way prediction and pseudo associative | low | low | High | NOT DEFINED |
| | Compiler optimizations | low | high | Less | NOT DEFINED |
| **Hit Rate** | Small cache | high | high | High | Less |
| | Pipelined cache access | NOT DEFINED | high | High | High |
| | Trace cache | NOT DEFINED | high | Less | NOT DEFINED |

Table I. **Comparison Table for Cache Optimization Techniques on the Basis of Miss Rate (MR), Hit Time (HT), Access Time (AT), Cost(C)**

## IV.SMP CACHE SIMULATOR

The SMP cache has clearly defined, full graphic and user friendly interface. A conventional way to enhance the performance of the system is to use many processors that can run in parallel to support a given assignment. The

least difficult multiprocessor association is SMPs (symmetric multiprocessors). A SMP consists of different indistinguishable processors inside a similar computer, which is interconnected by a transport. At that point when in excess of one processor is possessing on a solitary chip, the design is alluded to as chip multiprocessing. To manage cache consistency on a SMP, the information cache underpins a convention is called as MESI. It consists of two status bit for each tag. With the goal that every line can be in one of the four states:

A. Modified
B. Exclusive
C. Shared
D. Invalid

### *4.1 PLATFORM USED*

We have utilized SMP3.0, a follow driven test system for the execution examination of real cache replacement approaches [3]. Follow driven test system is a savvy strategy of execution assessment of computer framework plan, exceptionally for cache outline, paging framework and TLB. In this paper, we have utilized some SPEC92 Benchmarks such as: SWM, NASA7, UPCOMP and MDLJD for the performance analysis of replacement policies.

Table II. **4.2 *SIMULATION SETUP* Experimental Results**

| **BENCHMARKS** | **REPLACEMENT POLICY** | | | |
|---|---|---|---|---|
| | **RANDOM** | **FIFO** | **LFU** | **LRU** |
| **MDLJD** | 92.61 | 92.915 | 93.23 | 93.27 |
| **UPCOMP** | 90.377 | 90.926 | 90.962 | 90.962 |
| **SWM** | 90.415 | 89.55 | 90.65 | 90.34 |
| **NASA 7** | 90.296 | 90.674 | 90.674 | 90.674 |

Bus Arbitration = LFU
Cache Coherence Protocol = MESI
Blocks in Main Memory = 8192

Word Wide (bits) = 16

Block size = 32 bytes

Blocks in Cache = 128

Main Memory size = 256 K Bytes

Cache size = 16KB

Replacement Policies = RANDOM, FIFO, LFU, LRU

Writing Strategy = Write Back

Mapping = 8 way- set associative

Number of Processors = 1

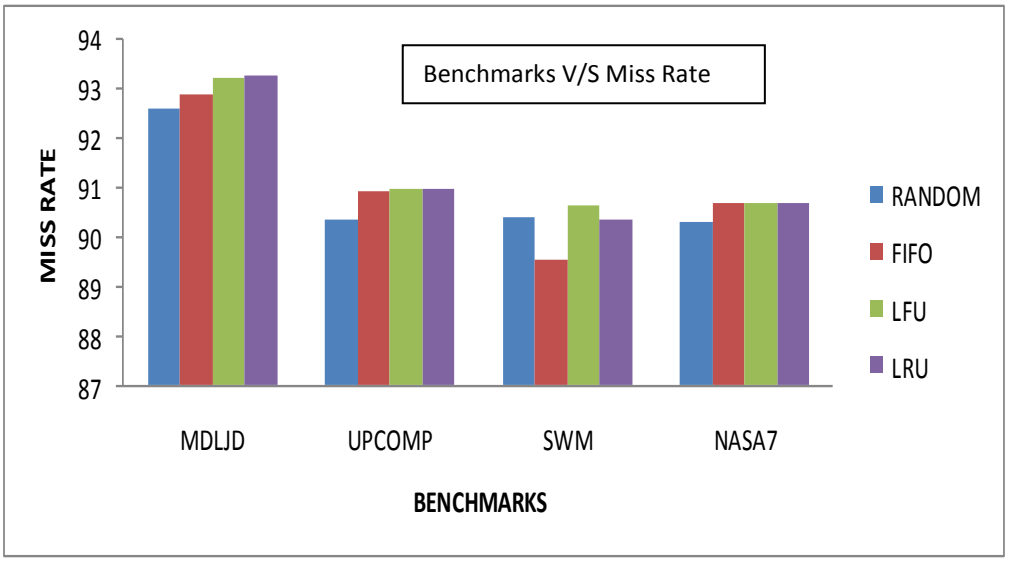


**Fig. 2 Resultant Graph of Benchmarks V/S Miss Rate**

## V.CONCLUSION

The main goal of this paper is to conclude, how the above mentioned replacement policy are comparative to each other on the particular benchmarks. Now the above table and the graph of experimental results show that the LRU replacement policy has better cache performance among all the cache replacement policy which is discussed above. Here we discussed the benchmarks MDLID, UPCOMP, SWM, NASA7 which is shown in the SMP cache simulator and it is used to explore our result under the given conditions.

## REFERENCES

**Journal Papers:**

[1] Komal Agrahari, Dr.P.K.Singh, "Realisation of Cache Optimisation using New Technique", *International journal of Advanced Research in Computer Science 2017, Volume.8 Issue. II*

[2] Fu, John WC, and Janak H. Patel. "Data prefetching in multiprocessor vector cache memories." *ACM SIGARCH Computer Architecture News, Vol. 19. No. 3. ACM, 1991.*

**Books:**

[3] W. Stallings, "Computer Organisation and Architecture," Eighth Edition, 2011.

**Conferences:**

[4] Komal Agrahari, Dr.P.K.Singh, "PERFORMANCE ANALYSIS OF CACHE MEMORY", (ICIIECS) IEEE Sponsored, 2017.

[5] M. K Verma, Komal Agrahari, " Analytical Study of Performance Trade-Off between Processor and Memory ", (ICIIECS) IEEE 2017

**Blogs:**

[6] https://www.google.co.in/search?q=memory+hierarchy&dcr=0&tbm=isch&source=iu&ictx=1&fir=Lg0HkWK0Z58ttM%253A%252CjUOHhveciDF0aM%252C_&usg=__y612VX7Xf_lfHwxdNV6-EOQHLKc%3D&sa=X&ved=0ahUKEwjJh7r00PLZAhWEuY8KHfS0CtsQ9QEIeDAR#imgrc=Lg0HkW