**REPLACEMENT ALGORITHMS**

Substitution/policies are used to achieve optimized usage of the cache. When the cache is full, then substitution policies decide which piece of data is replaced to make space for new data that is currently being used. An efficient algorithm is that which can take less time and the number of cache misses are low and also balancing cost. Following are some of the algorithms.

**LRU (Least Recently Used) Algorithm):**

This algorithm reject the least recently used item from the cache to make space for the new data item. To gain this, a history of all data items that is which data item is used when is kept. A variable known as Aging Bit is used to load this information, Although this algorithm provides a better show the cost of implementation is much more [6]. Variants of LRU are the most beloved of all among all other algorithms. The key advantage of this policy is its simple execution, time, and space overhead is constant. “Recency” is the main factor in this algorithm while LRU takes into account the characteristics of “recency” of the workload; it ignores and exploits the capabilities of “frequency” of a workload [7]

**HLFU (History Least Recently Used) Algorithm):**

Cache replacing in a typical LFU algorithm is performed by replacing the least frequently requested objects. But in HLFU which is an extension of LFU considers the History function in the cache replacement process. Respective LFU threshold is a linear function on the amount of currently used cache. The HLFU algorithm will replace the cached objects based on the Hist value as compared to the defined threshold in LFU.

**LFU (Least Frequently Used) Algorithm:**

It counts how often data items have been used. The data items that are used less are deleted from the cache first. If all objects have the same frequency then this algorithm randomly refuse any data item [8].

**GDS (Greedy Dual Size):**

The index is calculated according to the size of a file. The larger the file smaller is the index. A file with the smallest index is replaced in this algorithm. Inflation value is used to keep track of frequently accessed files in the cache [9].

CAR (Clock with Adaptive Replacement) Algorithm:

CAR is simple to implement and it has a very low overhead on cache hits. It shows high performance and it also provides service of self-tuning. It is scan preventive which results in low space overheads that are less than 1% [10].CAR does not care for certain workloads [11].

ARC (Adaptive Replacement Cache):

It is easy to implement running time that is not dependent on cache size. ARC has a low position overhead of approximately 0.75% of the size of the cache. ARC is scan resistant also leads to self-tuning. This algorithm always balanced recency and frequency features by responding to changing access patterns [12]. This algorithm cache is divide into two queues, each is handle by using CLOCK or LRU that contains pages accessed only once, while the other contains the page which is accessed more than one time [13]. Like other algorithms, ARC also has a constant complexity per request. “Ghost cache” is a particular term used in this algorithm to handle the data element which will be used in near future [14-15].

**RR (Random Replacement) Algorithm:**

This algorithm randomly selects any of the data items from the cache and replace them with the desired one [4]. This algorithm doesn't need to keep track of the history of the data contents and it does not need any data structure. Due to which it consumes fewer resources, therefore its cost is less as compared to other algorithms [16].

**SLRU (Segmented LRU) Algorithm:**

This algorithm partitions the cache into two portions, one is unprotected and the other is protected. The protected portion is reserved for mostly used objects. When the first request for an object has been done then this object is added into the unprotected portion. On a cache hit the object is moved into the protected portion [17]. Both portions are managed by the LRU technique. But content from the unprotected part has been removed and content from the protected part has been moved back to the unprotected part of the recently used content. This method requires a variable that calculates what percentage of the cache space is reserved for the protected part [18].

**LR+5LF Algorithm:**

LR+5FU replacement policy is a combination of two popular replacement policies i.e. LRU and LFU. The problems that arrived in LRU and LFU policies are solved by a new policy called LR+5FU [6]. The weighing problem of LRU AND LFU is solved by this algorithm.LR+5LF policy reduces cache miss with a greater amount than LRU, FIFO, and LFU at L1 and L2 cache [19].

**FIFO Algorithm:**

The first in first out algorithm removes the page that has not been used for a long time. It treats the pages as a circular buffer, and pages are dismissed in around robin fashion. It causes early page fault [20].

**LLF (Lowest Latency First):**

It keeps the average latency to a minimum by first expelling the object with the lowest download latency [7]. This algorithm gives the best result in cases where the data is retrieved by executing a query against a relational database.

Algorithms that are discussed above are classified into several classes. These classes are made in terms of different parameters discussed in [9-2]. In all parametersrecency and frequency are the most important factors. This calcification is also used by other authors. Classification of classes is:

(i) Recency Based Algorithms

(ii) frequency Based Algorithms

(iii) Recency/Frequency Based Algorithms

(iv) Function based algorithms

(v) Randomized algorithms

This classification described in Fig. 2 and comparisons

between different algorithms are shown in Table 1.