***Chapter 2***

**LITERATURE REVIEW**

Cache replacement policies also called cache algorithms plays an important role in web caching. Following are various algorithms

**2.1 Cache algorithms:**

Due to the limited size of the size, a cache replacement policy is required to handle the cache content. If the cache is full and an object needs to be accommodated, the cache replacement algorithm will determine which object is to be evicted in order to allocate space for the new arrival object. An optimal replacement algorithm aims to make the best use of available cache space, in order to improve he cache hit rate, and to reduce the work load at origin server.

Most of the web cache servers still using the traditional caching algorithms. In [3] P. Cao et. al shown that these conventional algorithms are suitable in traditional caching like CPU caches and virtual memory systems, but they are not efficient in web caching area. The simplest and most common cache algorithm is Least-Recently Used (LRU) algorithm, which evicts the least recently accessed object in order to create space of the newly arrived web object. LRU is easy to implement and proficient for uniform size objects, like in the memory cache. However, it does not perform well in Web caching since it does not consider the size or the download latency of objects.

In LRU, every time object with least frequent usage is selected for replacement. Because of this there is a chance that unpopular objects may also reside in the cache which creates cache pollution. If the cache pollution occurs because of unpopular objects then it is termed as cold-cache pollution. In order to avoid this Rassul Ayani proposed an extension to the LRU in [2]. They proposed two algorithms namely LRU-Distance and Segmented LRU (SLRU).

In LRU-Distance algorithm, instead of placing new object at the top of the stack it places the object at a distance D from the bottom of the cache stack. Distance D defines the total number of objects away from the bottom of the cache stack. For example if D is 10 then an object that is accessed exactly once would be the tenth object to be removed from the cache. If the object is accessed more than once it moves to the upper part of the stack. So it checks to the cold cache problem by dropping the object if it is not accessed again in a certain time limit.

In Segmented LRU, cache is partitioned into lower and upper segments. The least accessed objects are placed in lower segment and the highest accessed objects are placed in bottom segment. The object in lower segment cache will be promoted into upper cache segment based on the number of times it is accessed. Each segment uses LRU algorithm for replacement. When the upper cache size is full then the least recent object is migrated into the lower segment.

Least frequently used (LFU) is another common web caching algorithm which removes the object with least number of accesses. It maintains a count value for every web object, so that it keeps objects with more number of counts and removes the objects with least number of counts. If the cache size is full then the objects with least number of count values are selected for replacement. However, it is also suffers from the cache pollution.

The size based algorithms are the common web caching algorithms that replaces the largest objects from cache when space is needed for the newly arrived objects. If the cache size if full then the object with largest size is selected as victim. If two objects are of same size, then the object with lowest identifier is evicted. Due to the replacement with these large objects, the cache might be polluted with the small unused objects which are not accessed again in the near feature.

To resolve the cache pollution, [3] suggested Greedy-Dual-Size (GDS) algorithm as an extension to the size based algorithm. The algorithm integrates several factors and assigns a key value for each web object stored in the cache. When cache space becomes full and new object is arrived, the object with the lowest key value is evicted. When user requests an object i, GDS algorithm assigns key value K(i) of object i as shown in Eq.(1):

K(i)=L+C(i)/S(i)

where C(i) is the cost of fetching object i form server into the cache; S(i) is the size of object i; and L is an aging factor. L starts at 0 and is updated to the key value of the last replaced object. The key value K(i) of object i is updated using the new L value since the object i is accessed again. Thus, larger key values are assigned to objects that have been visited recently.

[3] proved that the GDS algorithm achieved better performance compared with some traditional caching algorithms. However, the GDS algorithm ignores the frequency of web object. L. Cherkasova, et. al in [25] enhanced GDS algorithm by integrating the frequency factor into of the key value K(p) as shown in Eq.(2). The policy is called Greedy- Dual-Size-Frequency (GDSF).

K(i)=L+F(i)\*C(i)/S(i)

Where F(i) is frequency of the visits of object i. Initially, when i is requested by user, F(i) is initialized to 1. If i is in the cache, its frequency is increased by one.

Table 2.1: Comparison of conventional web cache algorithms

|  |  |  |  |
| --- | --- | --- | --- |
| **Policy** | **Brief description** | **Advantage** | **Disadvantage** |
| **LRU** | The least recently used objects are removed first. | Simple and efficient with uniform size objects, such as the memory cache. | ignores download latency and the size of web objects |
| **LFU** | The least frequently used objects are removed first. | simplicity | Ignores download latency and size of objects and may store obsolete web objects indefinitely. |
| **SIZE** | Big objects are removed first | Prefers keeping small web objects in the cache, causing high cachet hit ratio. | Stores small web objects even if these object are never accessed again.   * Low byte hit ratio. |
| **GD-size** | It assigns a key value to each object in the cache as equation (2.1) . Consequently, the object with the lowest key value is replaced when cache space becomes occupied | Overcomes the weakness of SIZE policy by removing objects which are no longer requested by users. | not take into account the previous frequency of web objects |
| **GDSF** | It extends GDS algorithm by integrating the frequency factor into of the key value K(p) as shown in Eq.(2) | overcomes the drawback of GD-size | not take into account the predicted accesses in the future |

**2.2 Intelligent Web caching algorithms**

The main motivation to the intelligent web caching approaches are, web access log files that can be supplied as training data in order to get the most popular web objects. The second motivation, since Web environment changes and updates rapidly and continuously, an efficient and adaptive schemes are required in Web environment. The machine learning techniques can adapt to the important changes through a training phase. Although there are many studies in Web caching, enhancement of Web caching performance using intelligent techniques is still fresh. In [29] Q. Liu et. al shown that the intelligent approaches are more efficient and adaptive to the Web caching environment compared to the previous approaches

**2.2.1 Fuzzy Algorithm** In [4] they proposed a replacement algorithm based on the fuzzy logic. This algorithm includes identification of input variables, output variables, and definition of membership function and construction of rule set. Object size, access frequency and time elapsed since last access are treated as input variables. Probability of replacement for each object is the output variable. For each input variables, output variable the corresponding membership function (MF) is defined. Two fuzzy sets fuzzy20 and fuzzy12 are defined. Based on these, the object with highest probability will be selected as victim.

**2.2.2 Generic algorithm (GA)** In [5] based on generic algorithms they implemented replacement algorithm. The GA is used because of two main reasons: First, the basic idea of the GA is based on the evolution of population by the criterion “survival of the fittest” and the cache should contain the fittest (i.e. non-stale, frequently accessed) information objects. Second, the GAs are applied to problems demanding optimization out of spaces which are too large to be exhaustively searched and the cache content usually consists of a large amount of information objects (stored files).

**2.2.3 Artificial neural networks (ANN)** In [6] they proposed a solution based on ANN with the help of particle swam optimization. Particle swam optimization [7] helps in reducing network weight and network structure. In ANN design for web caching, pre-processing data procedure, number of input nodes procedure, number of hidden layers and hidden nodes procedure, and number of output nodes procedures were used. Based on the result obtained by the ANN the replacement object will be decide.

**2.3 Prefetching algorithms:**

Broadly we can categorize prefetching policies into two ways. Content based prefetching and history based prefetching.

**2.3.1 Content based prefetching:** In this approach next page will be predicted based on the current page. By extracting the hyperlinks from the current page we can predict the future one. In this approach we no need to worry about user’s experience (i.e., browsing session). In [6] they used artificial neural network (ANN) to predict the next user request by sending the html links in the current page as inputs to the ANN.

**2.3.2 History based prefetching:** In this, future request will be predicted based on past history. By considering the users past experience the next user requests will be predicted. We can classify this approach into four categories.

**2.3.2.1 Dependency graph approach (DDG):** In this approach we will construct a dependency graph, where nodes indicate web pages and edges indicate next node after the current one and each edge assigned by certain weight. In [10, 11] this approach is used for predicting the next user request.In [12] they implemented the double dependency graph approach by analysing the current web page in order to improve the performanceof web caching with the help of prefetching. DDG algorithm consists two classes. First class consist of objects of the same page and the second class contains objects of different pages. In each class they applied basic dependency graph approach.

**2.3.2.2Markov model approach**: In this, next user request is based on the past sequence of requests. Depending on the number of past requests used to predict the next possible request, Markov model based technique is called as first order, second order, ....., and kth order Markov model technique where last, last two, .........., and last k requests are used respectively. As the number of order increases the recent history objects will also increases which leads to greater accuracy. [13] used Kth order markov model, in which they used up to k orders. In this approach, the prediction is done by combining different order Markov models.

Firstly, prediction of next user requests are done by using highest order markov model. If those requests are not found then the order of markov model will be decreased until it founds. But this model suffers from greater time complexity.

Prediction-by-Partial-Matching (PPM) algorithm based on higher order markov model is implemented in [14]. In PPM, based on past user requests markov prediction tree will be constructed. Prefetching decisions are made based on this markov prediction tree.

If the past user requests are more, then the size complexity of prediction model increases. In order to avoid this, in [13] they proposed an approach Longest Repeating Subsequence (LRS). In this, they used two methods: a mining method which extracts the past user history with longest repeating sub sequences and pattern matching method which is based on the weightage of pages.

In [15] they proposed a variation to the PPM model called popularity based PPM model (PB-PPM), in which they used grades to rank web access patterns and builds a markov predictor tree to do prefetching. In the prediction tree, they assigned long branches to most popular web pages and short branches to less popular web pages. It is possible to limit the size of web history by deleting the short branches.

In LRS approach, the size of the past history structure is reduced by storing only longest repeated sub-sequences that is frequently requested pages. But in real web, most of the requests are infrequently accessed pages. So it affects the prefetching performance, but the cost of memory is reduced. In PB-PPM, it allows only most popular based requests as tree roots. By doing this, it further reduces the past history structure size. But it slightly effects on prefetching performance.

**2.3.2.3 Cost function based approach:** In this approach based on functions like popularity, life time, the next user requests are predicted. [16] Used popularity function, in which it prefetches the top ten popular web objects.

In [17] they presented an approach based on the lifetime of objects. In this, every time they will fetch the objects with longest lifetime. In [17] they also used prefetch by APL. In APL, for object i, the arrival rate of object (A), probability of accessing that object (P) and lifetime of object (L) values will calculate. Here a\*pi represents user request rate of object i and a\*pi\*li represents the number of requests for object i before its life time expires.

A prediction algorithm based on maximum weight matrix is presented in [].The basic idea is to train the machine (caching system) to learn the request pattern from the client side little by little. The learning process is by prediction on next request following the current one. If the prediction is proved to be correct, the corresponding probability is increased. Otherwise, the probability matrix is not changed. Every element present in the matrix is the probability from one web page to the other web page.

In [23]they proposed an algorithm based on dynamic web prefetching. In this technique, subsequent links are prefetched only if bandwidth usage of existing network is less than a predefined threshold. For each web page request, the retrieved page is parsed to identify the subsequent links and URL’s corresponding to these links are searched in the hash table to get its weight information. Intelligent agents monitor the bandwidth usage, user’s preferences and hash table weights to identify the number of URLs to be prefetched.

**2.3.2.4 Data mining based approach:** This is classified into two approaches. 1. Association rules based approach and 2. Clustered based prefetching approach.

**Association rules based approach:** Set of rules which finds the linkage between the most frequent accessed pages in a user session is called association rules.Number of pages requested by a single user to a particular web server during certain time period is termed as user session. Support and confidence are the two metrics used to test the association rules. Finding the most frequent pages is termed as support and finding the linkage rules among these frequent pages is termed as confidence.

**Clustering based prefetching approach:** A group of similar pages is called cluster. That is objects in one cluster are similar and the objects in other cluster are similar. The intra cluster distance is low and inter cluster distance is high.

[18] have introduced a clustering based prefetching approach, based on clustWeb algorithm that identifies a group of related cluster web pages based on users past history. In this algorithm, they divided the users past history into certain number of clusters based on user domain. The requests of each cluster are represented by a weighted graph. If there are more number of web pages present in a cluster, the accuracy of prefetching will also increases.

Clustered page accesses based on page rank algorithm is presented in [19]. In this every page is ranked based on the linkage with other pages. The pages with highest rank will be fetched**.** This algorithm will also helpful for cache replacement policy. Suppose if the cache needs replacement then the page with lower rank will be selected for replacement.

**2.4 Integrating Web caching and prefetching**

Web proxy caching and prefetching are the most widely used techniques which play an important role in improving the performance of Web. Web proxy caching exhibits temporal locality and the web prefetching exploits the spatial locality of the Web objects. Thus, web proxy caching and prefetching can complement each other. Combination of the web caching along with prefetching helps on improving hit ratio and reducing the user perceived latency. However, if the web caching and prefetching are integrated inefficiently, this might cause increasing the network traffic as well as the Web server’s load [24, 25, 26, 27]. Therefore, the prefetching approach should be designed carefully in order to overcome these limitations.

Basically, the web prefetching requires two steps: predicting the next user requests and prefetching them into the cache. This means the web prefetching also involves in caching.

It is important to take into consideration the impact of these two techniques combined together. [12] studied the effect of combination of web caching and prefetching on end user latency. They concluded that the combination of web caching along with prefetching can improve the performance up to 60%, whereas web caching alone improves the performance up to 26%. In [28] Q.Yang et. al suggested web log mining application in order to obtain the web object access patterns and used these patterns to extend the GDSF caching algorithm. In [29] W. Teng et. al proposed IWCP cache replacement algorithm for integrating the web caching along with web prefetching at client side proxies. They derived a normalized profit function to evaluate the profit from caching an object either a no implied object or an implied object according to some prefetching rule.

[13] used artificial neural network (ANN) approach as both prefetching approach and web caching approach. This approach depended on the keywords of URL anchor text in order to predict the next subsequent requests. The most important factors like frequency were ignored in web cache replacement algorithm. Moreover, since the keywords extracted from web documents were given as inputs to ANN, applying ANN in this way may cause extra overhead on the server.

In [30] B. jin et. al presented a compact set of algorithms for integrating web caching and prefetching for wireless local area network, including sequence mining based prediction algorithm, context-aware prefetching algorithm and profit driven web caching replacement algorithms.

In [31] S. Sulaiman et. al proposed a framework for combining web caching and prefetching on mobile environment. They proposed a hybrid model called Neuro-PSO based on the combination of artificial neural network (ANN) and particle swam optimization (PSO) for classification of web objects. Then, rules from log data are generated using Neuro-PSO technique on the proxy server. At prefetching side, prediction algorithm based on XML is suggested to be implemented on mobile device to handle communication between client and server.