A Least Grade Page Replacement Algorithm for Web Cache Optimization

Naizheng Bian[1)], Hao Chen[1,2)*]
1) Software School, Hunan University, Changsha, China
2) School of Information Science and Engineering, Central South University, Changsha, China
chenhao@hnu.cn

Abstract

Caching is a technique first used by memory management to reduce bus traffic and latency of data access. Web traffic has increased tremendously since the beginning of the 1990s. With the significant increase of web traffic, caching techniques are applied to web caching to reduce network traffic, user-perceived latency, and server load by caching the documents in local proxies. In this paper, we analyzed both advantages and disadvantages of some current web cache replacement algorithms including lowest relative value algorithm, least weighted usage algorithm and Least Unified-Value (LUV) algorithm. Based on our analysis, we proposed a new algorithm, called least Grade Replacement (LGR), which takes recency, frequency, perfect-history, and document size into account for web cache optimization. The optimal recency coefficients were determined by using 2- and 4-way set associative caches. The cache size was varied from 32k to 256k in the simulation. The simulation results showed that the new algorithm (LGR) is better than LRU and LFU in terms of Hit Ratio (HR) and Byte Hit Ratio (BHR).

1. Introduction

The keys of web caching have three aspects: 1. Algorithm of routing requests; 2. Algorithm of replacing documents; 3. Algorithm of updating documents (consistence). This paper focused on the second aspect, algorithm of replacing documents.

With the study of web coaching's characteristics going further [2], algorithms of replacing documents based on the statistics of collected web data were proposed. Each of them considers one or more of the following factors into its scheme: 1. Document reference frequency; 2. Document reference recency; 3. Document sizes; 4. Strong or loose consistence of documents; 5. Replica document in different proxies; 6. Non-replica document in different proxies. Most of efficient schemes combine more than one of factors in their implement of web cache. Some algorithms consider different cache architecture to improve caching performance, such as a fully-connected network of N cache [3].

2. LGR Related Algorithm of Web Caching

2.1. Lowest Relative Value Algorithm

Luigi [5] and Vicisano proposed a replace algorithm for proxy cache called Lowest Relative Value (LRV). It is based on maximizing an objective function for the whole cache. The objective function uses a cost/benefit model to calculate the relative value of each document in the cache. Two performance parameters of cache are used: the HR and BHR. The value of the document is calculated according to this formula: V= (C*P)/B, where C=size of document + overhead of getting document from original sites, B is the bandwidth toward the server, and P is the possibility that a document is accessed again. When a replacement is needed, the cached document with lowest value is replaced. Compared with other algorithms, such as LFO, FIFO, random, and replacement based on document's size, the performance of the LRV outperforms all of them.

The LRV is particularly useful for small caches. With the cache capacity becoming larger, the overhead of maintaining the list of relative values of all cached documents increases, the performance of LRV drops.

2.2. Least Weighted Usage Algorithm

Ying, Edward, and Ye-sho [7] argued that model-driven simulation was more objective than trace-driven. A web cache algorithm called Least Weighted Usage (LWU) was proposed using model-driven simulation. The LWU was regarded as dynamic and self-adaptive. The performance of LWU was above popular web cache algorithms' up-bound of

0-7695-3090-7/08 $25.00 © 2008 IEEE
DOI 10.1109/WKDD.2008.124

HR, which is 55% [7]. The access history was divided into same size intervals. In each interval a cached document was assigned by either 1 or 0 corresponding to whether it was accessed or not during this interval. The probability of accessing a given document at current time was the sum of the product of the previously assigned value with a weight (denoted as γ). The weight was the parameter that determined how rapidly the influence of past access on the new selection died out. Another parameter that determined the probability of a new entry in cache was defined as α=R/T, where R is the total number of different documents in cache and T is the total number of usages. The parameter α is used to calculate the value of γ with the access statistics. The optimal combination of α and γ was determined by simulation.

Model-driven simulation allows a much wider range of access patterns while trace-driven simulation only assumes a specific user access pattern. The LWU performs best due to its adaptation to both frequency and recency in different environments. The drawback of LWU is that it ignores the size of web documents.

2.3. LUV Algorithm

Bahn et al. proposed a web cache replacement algorithm called LUV that uses complete reference history of documents, in terms of reference frequency and recency [6]. It uses a parameter H(i) to represent the possibility of re-reference. $H(i) = \sum_{k=1}^{n} F(t_c - t_k)$, where F is the weighting function, t_c is the current time, n is the number of references made to document I since it has been brought into the cache, and tk is the time of the kth reference. An efficient algorithm is used to update the H(i). When a replacement needs to be made, the document with the lowest value of H(i) is replaced.

Its advantage is that it uses the full history of documents, so the parameter H(i) contains more information compared with that used in other algorithms. Its weakness is that the optimization of weighting function is not objective.

3. LGR Algorithm

3.1. Motivation

Based on the above analysis of the web cache replacement policies, we obtained the following salient conclusions that perfect-history LFU outperforms in-cache-history LFU in terms of hit rate (HR) and byte BHR [10], LRU outperforms LFU in term of HR [9], and the priority of replacement is increased as the increment of the document size [1]. However, the research literature is remarkably silent on this particular problem to integrate these factors into a cache replacement policy.

3.2. LGR Page Replacement Policy

According to the above conclusions, we proposed a new replacement algorithm. The algorithm grades each in-cache document basing on its passed history, which are recency, frequency, perfect-history and size. When the set is full, the least grade document will be replaced, but its grade will be stored in a perfect-history grade depository (PHGD) for future references. Due to the survey, we can draw a conclusion that the relatively most important factor is its recency, and then frequency, perfect-history and size. We only consider these four factors for grading because they are relatively most important factors for real network traffic.

The n-way set associative caches is used here. The least grade page replacement algorithm is given below.

```
ALGORITHM :: ReplaceLeastGradePage (
    /* In-cache document based on its passed history */
    ICDR: In-Cache Document's Records
    /* Discarded document in cache */
    PDDG: Previously Discarded Document's Grade
    /* The de facto hit ratio beginning with 0 */
    F : Frequency of ICDR ;
    R : Recency Set of ICDR; /* R=< r1;r2;...;rn>*/
    L : Length of document;
    BG : Bonus Grade of document)
{
    IF (document k in cache) THEN
        FOREACH doc in ICDR DO
            WHILE(doc.F in F
                  & doc.R in R
                  & Size(doc) in L)
              doc ← newValue;
              PDDG ←weight^α , α ∈ (0, 1);
    ELSE
        FetchDocFromOrigina();
        DiscardInCachedDoc();
}
/*Fetch document k from original site*/
PROCEDURE: FetchDocFromOrigina()
{
     IF(L is NOT NULL) THEN
         INSERT k into Cache;
         UPDATE each ICDR;
         k.BG = 0;
     ELSE
         FOREACH g in grade DO
```

$$g = \delta_1 / R + \delta_2 \times F + \delta_3 \times C + \delta_4 \times BG;$$

```
}
/*discard in-cached doc with the least grade*/
PROCEDURE: DiscardInCachedDoc()
{
    PDDG ← PHGD.grade;
    INSERT k into Cache;
    IF ( PDDG of k in PHGD) THEN
        k.BG←PDDG;
        Delete its PDDG in PHGD;
    ELSE
        k.BG=0;
        Update each ICDR and PDDG;
}
```

Figure 1. **LGR Algorithm**

3.3. Simulation Implementation

Step 1: Initialize 2- and 4-way set associative caches;
Step 2: Determine the cache entry and the output link:
Step 3: Tokenize the whole source IP of a document into 4 integers by dots, and let them be I[0], I[1], I[2], I[3], and its length be L.
Step 4: Employ the following algorithm (in order to distribute index as broadly as possible, we use full component of IP address and its length) to obtain cache set index for the packet.
Step 5: For simplifying the simulation, let link=I[3] +L/100+1;
Step 6: Compare results of LFU and LRU.

3.4. Performance Measures

Performance measures of interest in the Web caching realm can be defined according to the goal of caching. The three popular performance measures are used in Web caching, that is, the hit rate, the byte hit rate, and the delay-savings ratio, denoted as HR, BHR, and DSR, respectively. We will employ the first two measures due to our trace library.

$$\text{HR} = \sum h_i / \sum r_i$$

$$\text{BHR} = \sum (s_i h_i) / \sum (s_i r_i)$$

Where

h_i: number of hit references to document i,

r_i: total number of references to document i (number of hits + number of misses),

s_i: size of document i,

3.5. Coefficient Selection

Since the grade is determined by the addition of multiplications of four factors and their coefficients. The grade equation is

$$\text{grade} = \delta_1 / R + \delta_2 \times F + \delta_3 \times C + \delta_4 \times BG$$

Four factors are recency (R), frequency (F), document size (C) and bonus grade (BG), the last section takes the perfect history into consideration. The coefficients (δ_1, δ_2, δ_3 and δ_4) are weights for these factors. Therefore it is significant to select the effective coefficients to implement this algorithm. We employed experiments to select the effective coefficients for 2 and 4-way set associative caches. Since recency is the most important factor and the coefficients' values are relative, we would like to select optimal δ_1 by fixing other three coefficients' values for simplification. The HR of 2-way set associative caches is shown in figure 2, and the HR of 4-way set associative caches is shown in figure 3.

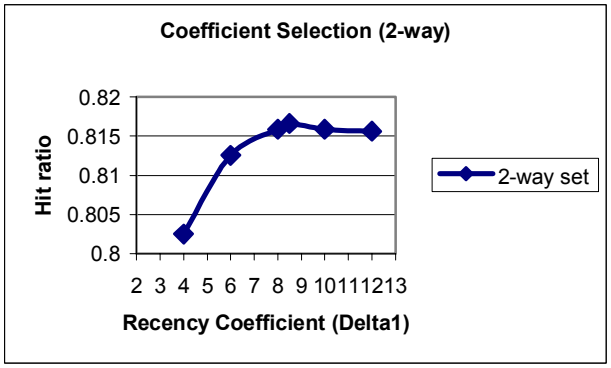

Figure 2. **Coefficient Selection (2-way)**

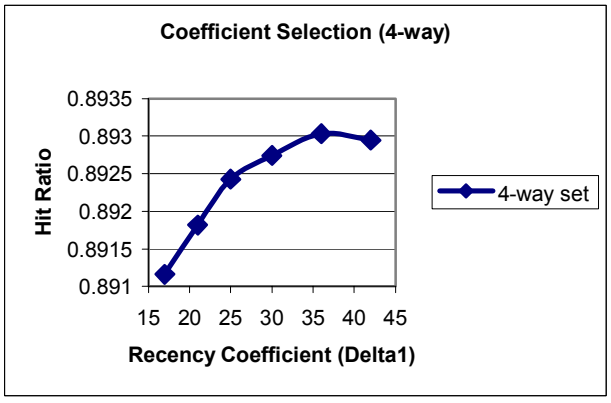

Figure 3. **Coefficient Selection (4-way)**

Based on the above results, we can determine the optimal recency coefficients: 8.5 for 2-way set associative caches and 36 for 4-way set associative caches.

3.6. Simulation Result

LRU is the most widely-used web cache replacement algorithm [9], and LFU is often used to be compared with other algorithm as a baseline. So we adopted these two algorithms to compare with LGR.

We used data collected at proxy of our university. First we employed 32k as cache size and compared both HR and BHR using 2-way and 4-way set cache. The 32k 2-way result is shown in figure 4 and 5. The 4-way result is similar to that of 2-way.

Later, we also employed 64k and 256k as cache size and compared both HR BHR using 2-way and 4-way set cache. All the results show that LGR outperform LRU and LFU.

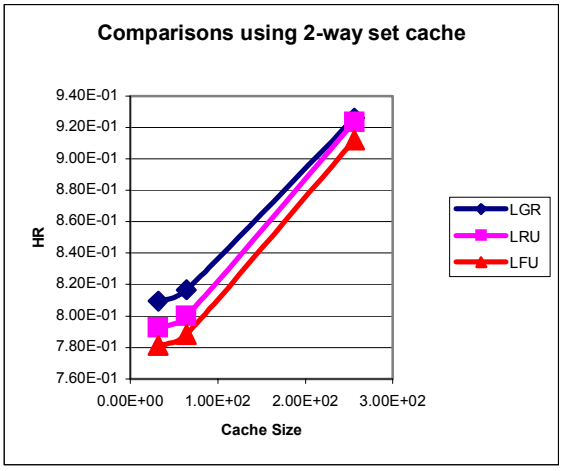

Figure 4. **HR of 32k 2-way simulation**

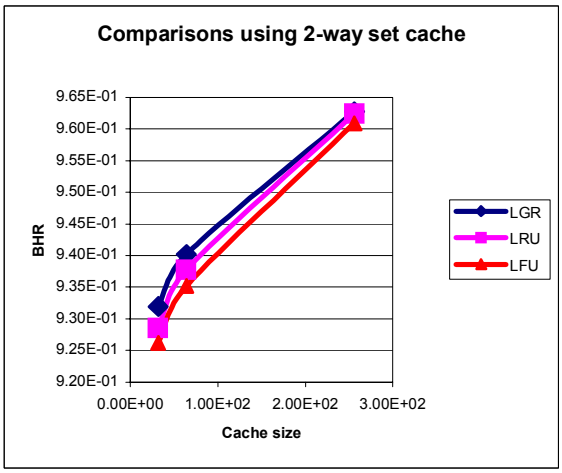

Figure 5**. BHR of 32k 2-way simulation**

4. Discussion & Conclusion

Based on perfect-history LFU and LRU, we proposed a new algorithm (LGR) by considering recency, frequency, perfect-history and size in replacing policy. The 2- and 4-way set associative caches were used to determine the optimal recency coefficients.

The simulation with cache size varying from 32k to 256k showed that the new algorithm (LGR) is better than LRU and LFU in terms of Hit Ratio (HR) and Byte Hit Ratio (BHR). Experimental results show that the proposed algorithm can reduce network traffic and latency of data access efficiently.

Acknowledgments

This work was partially supported by the National Natural Science Funds of China Grant #60673093.

References

[1] Pei Cao, Snady Irani, "Cost-Aware WWW Proxy Caching Algorithms", in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December, 1997.

[2] K.Thompson, G. Miller, and R. Wilder, "Wilde-Area Internet Traffic Patterns and Characteristics," in *Proceedings of 3rd Intelnational Conference Web caching,* May, 1998.

[3] Seda Cakiroglu, Erdal Arikan, "Replace Problem in Web Caching", in *Proceedings of IEEE Symposium on Computers and Communications*, June, 2003.

[4] Chrlos Maltzahn, Kathy J. Richardson, Dirk Grunwald, "Reducing the Disk I/O of Web Proxy Server Caches", *in Proceeding of the 1999 USENIX Annual Technical Conference, Monterey, California,* June, 1999.

[5] Luigi Rizzo and Lorenzo Vicisano, "Replacement Policies for a Proxy Cache". *IEEE/ACM Trans. Networking*. Apr. 2000, pp: 158-170.

[6] H. Bahn, S. Noh, S. L. Min, and K. Koh, "Using Full Reference History for Efficient Document Replacement in Web Caches", in *Proceedings of the 2nd USENIX Symposium on Internet Technologies & Systems*, October, 1999.

[7] Ying Shi, Edward Watson, and Ye-sho Chen, "Model-Driven Simulation of World-Wide-Web Cache Policies". in *Proceeding of the 1997 Winter Simulation Conference,* June,1997, pp: 1045-1052.

[8] Ganesh, Santhanakrishnan, Ahmed, Amer, Panos K. Chrysanthis and Dan Li, "GDGhOST: A Goal Oriented Self Tuning Caching Algorithm", in *Proceeding of The 19th ACM Symposium on Applied Computing.* March,2004.

[9] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," *In Proceedings of Infocom '99,* May,1999.

[10] Ben Smith, Anurag Acharya, Tao Yang, and Huican Zhu, "Exploring Result Equivalence in Caching Dynamic Web Content ", *2nd USENIX Symposium on Internet Technologies and Systems, October*, 1999.