ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

WEB CACHING AND RESPONSE TIME OPTIMIZATION BASED ON EVICTION METHOD

V.Padmapriya[1] K.Thenmozhi[2]

Assistant Professor, Dept. of Computer Science, Dr N.G.P Arts & Science College, Coimbatore, Tamilnadu, India [1]

Assistant Professor, Dept. of Computer Science, Dr N.G.P Arts & Science College, Coimbatore, Tamilnadu, India [2]

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, analization of 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 is done. 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 32 k to 256 k in the simulation. The simulation results showed that the new algorithm (LGR) is better than LRU and LFU in terms of hit ratio (BR) and byte hit ratio (BHR).

Keywords: web mining; cache mining; web cache; proxy cache

I. INTRODUCTION

Adjacency Cache Design

We discuss three general purpose cache distributions and lookup enhancements that improve both the locality and latency of cache advertisements. The system uses a form of hierarchical aggregation to summarize the contents of cached files available in a particular local area network. In this way the amount of indexing information that has to be exported to other systems in a WAN can be reduced. A common criticism of distributed hash tables is that they lack locality. This is a side effect of the hash function used to identify both nodes and content in the DHT network. The hash function provides a key-to node mapping that distributes keys uniformly at random across the address space of the DHT. As such, semantically related nodes and data items when processed by a systems hash function will be mapped to random locations in the network with high probability. This presents a number of problems for cache index and lookup systems. First, lookup requests for file content such as images and linked web pages require a separate lookup request for each URL. This will result in a worst-case time complexity of O (M log N) where M is the number of embedded file references in a webpage and N is the number of nodes in the system. Second, due to the random nature of the hash functions used to identify files, lookup requests for linked files are likely to be routed to nodes that are far away in the network.

This significantly adds to the latency of locating a cached file in the network. However, many of these lookup requests are unnecessary and can be reduced by exploiting the link structure of web pages. In a typical web browsing scenario, client software will make a connection to a web server and download the HTML specification of a web document. Once this has occurred, the client process will parse the document tree and generate a series of HTTP get requests to download embedded file content from the web server. As such, this content should also be available at the same remote cache system as the main webpage unless it has been evicted from the remote cache. To reduce these extraneous lookup requests, cache misses and extra round-trip delays, we have developed a combined indexing structure that client systems can use to identify the set of related cache items also available at a remote site. This combined index has been implemented using a bitmap vector the contents of which are used to determine the presence or absence of linked web content. This effectively allows a client system to select a remote cache based upon the number of related documents that it stores. As a result, lookup requests for related files such as embedded images can be downloaded from the same remote cache without having to specifically locate the file using the DHT index. The idea here is to extend the reach of the cache index by one link's worth, to enable a client system to determine ahead of time whether linked content is available at a remote proxy. As a consequence, communication between a client and remote cache system can be reduced because of these cache hints. This allows a client system to maintain a persistent connection with a remote cache, so that file requests for linked web content can be pipelined across

the same socket. To create this combined index, the link structure of a cached file has to be extracted using regular expressions. This process creates an ordered set of links that can be used to create a bitmap vector of the linked files available at a remote site. As such, the length of a bitmap vector corresponds to the number of out links in a given web page. To encode the availability of linked content at a remote site, the corresponding bit locations of these out links are set in the bitmap. Therefore, the ith link is represented by the ith bit in the bitmap vector. To illustrate this idea, consider a web page that has five links to other files. If each of these linked files were available at a remote cache, then each bit location in the bitmap vector of this cached item would be set to one. However, if only the second and third links were available at a remote cache, then only bit locations one and two would be set in the bitmap. The intuition here is that users will browse to a new page through an existing hyperlink directly, instead of jumping to a new page at random . Therefore, if we know which links are available ahead of time, the number of cache lookup messages routed across the network can be reduced. Once a browser has downloaded a list of IP addresses and adjacency cache bitmaps from the DHT, these are added to a fixed size in memory cache which has a least recently used eviction strategy.

Distributed Hash Table

Distributed hash tables (DHTs) are a class of decentralized distributed systems that provide a lookup service similar to a hash table: (*key*, *value*) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows DHTs to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures. DHTs characteristically emphasize the following properties:

Decentralization: The nodes collectively form the system without any central coordination.

Scalability: The system should function efficiently even with thousands or millions of nodes.

Fault tolerance: The system should be reliable (in some sense) even with nodes continuously joining, leaving, and failing.

II. ALGORITHMS

A. *Lowest Relative Value Algorithm (LRW)*

Luigi 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.

Disadvantage of LRV

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.

B. Least Weighted Usage Algorithm (LWU)

Ying, Edward, and Ye-sho 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.

Disadvantage of LWU

The drawback of LWU is that it ignores the size of web documents.

C. Least Unified Value Algorithm (LUV)

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.

Disadvantage of LUV

Its weakness is that the optimization of weighting function is not objective.

D. Least Grade Page Replacement Algorithm (LGR)

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, requency, 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.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

LGR Algorithm

```
ReplaceLeastGradePage (
ICDR: In-Cache Document's Records
PDDG: Previously Discarded Document's Grade
F : Frequency of ICDR ;
R : Recency Set of ICDR;
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 ←weightα , α ∈ (0, 1);
ELSE
FetchDocFromOrigina();
DiscardInCachedDoc();
}
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=§1 / R + §2 x F +§3 x C + §4 x BG
}
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;
}
```

III. IMPLEMENTATION

This work **web cache (proxy server)** is to develop a utility to share internet from single connection to a large network around 200 machines with different operating systems. The software is developed using the Java Language. Java applet applications are mostly used in the web pages, but we use JFC (swing) for developing the software.

- This work provides an intelligent environment containing a number of ready-made options like cache, log file, error checking, connection pooling, etc. These ready-made tools may be any of the GUI components that are available in the Java AWT package. By using this utility, Administrator can control and maintain the whole network.
- This thesis aim is to use the Least Recently Used document in web caches which replaces Randomized web cache replacement algorithm. A web cache sits between web server and a client and watches request for web pages. It caches web documents for serving previously retrieved pages when it receives a request for them.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

Advantage

- Saves memory
- Saves processing power
- Reduces network traffic
- Reduces Latency time
- To reduce load on web servers
- Avoid the need for data structures.
- The utility function assigns to each page a value based on -recentness of use -frequency of use -size of page -cost of fetching.
- Least grade page replacement algorithm is proposed to support the deletion decision. N-M Method.

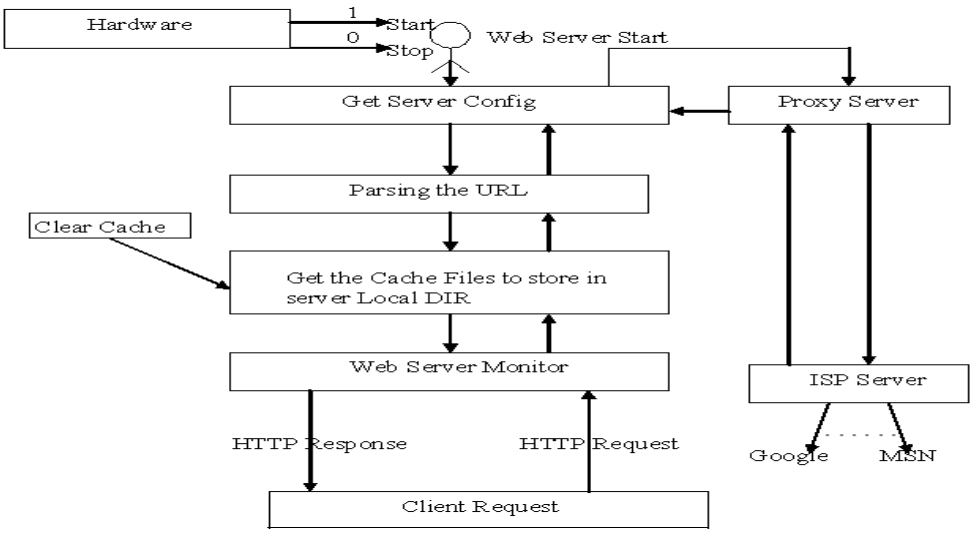

Fig.1. Data Flow Digram

IV. PROXY IMPLEMENTATION

A proxy server is developed which runs with mentioned features, which inherently helps speeder browsing of web pages with use of least grade page replacement algorithms. This server is successfully implemented with a few numbers of clients but it could be implemented for more of them. As mentioned before it is more reliable, more advantageous than the existing one which uses the old Data structures concept. It can work in a larger network and also maintains load balancing so I conclude that this system application is executable under any platform and with any number of clients.

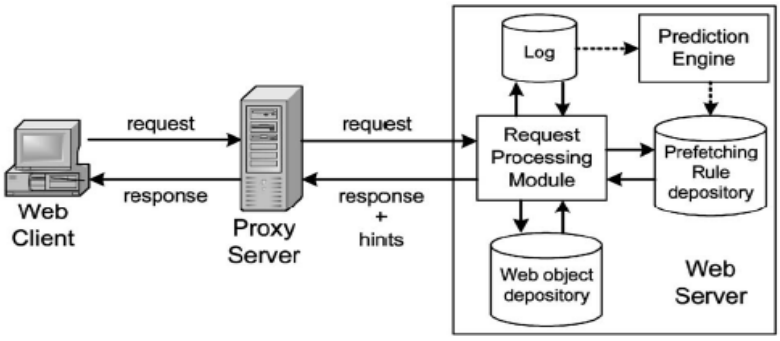

Fig.1. System Flow Digram

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

V. FETCH UNITS AND RESULT PRE-FETCHING

In many search engine architectures, the computations required during query execution are not greatly affected by the number of results that are to be prepared, as long as that number is relatively small. In particular, it may be that for typical queries, the work required to fetch several dozen results is just marginally larger than the work required for fetching 10 results. Since fetching more results than requested may be relatively cheap, the dilemma is whether storing the extra results in the cache (at the expense of evicting previously stored results) is worthwhile. Roughly speaking, result prefetching is profitable if, with high enough probability, those results will be requested shortly while they are still cached and before the evicted results are requested again. One aspect of result prefetching was analyzed in, where the computations required for query executions (and not cache hit ratios) were optimized.

Table 1.Upper bounds on hit ratios for different values of the fetch unit

Fetch Unit	Number of Fetches	Hit Ratio
1	4496503	0.372
2	3473175	0.515
3	3099268	0.567
4	2964553	0.586
5	2861390	0.600
10	2723683	0.620
20	2658159	0.629
32	3657410	0.629

VI. WEB CACHE REPLACEMENT ALGORITHM

The keys of web caching have three aspects:
1. Algorithm of routing requests
2. Algorithm of replacing documents
3. Algorithm of updating documents

This paper focused on the second aspect, algorithm of replacing documents .With the study of web coaching's characteristics going further, 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.

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.

Algorithm

```
If (eviction)
{
If (first iteration)
{
Sample (N); evict_least_useful;
keep_least_useful (M);
} else {
Sample (N-M);
evict_least_useful;
keep_least_useful (M);
}
}
```

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

VII.HTTP CONTENT AND PARSING ANALYSIS

Parsing is the process of analyzing an input sequence in order to determine its grammatical structure with respect to a given formal grammar.

Process Steps

A.Lexical analysis:

The input character stream is split into meaningful symbols (tokens) defined by a grammar of regular expressions – Example: the lexical analyzer takes "12*(3+4) ^2" and splits it into the tokens 12, *, (, 3, +, 4,), ^ and

B. Syntax analysis:

Checking if the tokens form an legal expression, w.r.t. a CF grammar. Limitations - cannot check (in a programming language): types or proper declaration of identifiers

C. Semantic parsing:

Works out the implications of the expression validated and takes the appropriate actions. Examples: an interpreter will evaluate the expression, a compiler, will generate code.

VIII. 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 Real time with cache size varying from 32k to 256k showed that the new algorithm (LGR using N-M Method) 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.

IX.FUTURE SCOPE

The traditional browsing cache systems can not address both nonstationary and stationary browsing behaviors at the same time. The response time for an interactive browsing system can be greatly increased.

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 International 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 199USENIX 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.

[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.

[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.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 4, April 2013

BIOGRAPHY

V.Padmapriya is working as Assistant Professor Department of computer science, Dr N.G.P Arts and Science College, Coimbatore and doing M.phil in area of Data mining in Bharathiar University, Coimbatore. She has done her Post graduate degree MCA in Anna university. She has Presented & Published a Number of paper in reputed Journals. She has two years of teaching experience and research interests include Datamining, WebMining, Textmining, Semantic webmining, Webcache mining.

K.Thenmozhi is working as Assistant Professor Department of computer science, Dr N.G.P Arts and Science College, Coimbatore and doing Ph.D in Bharathiar University, Coimbatore. She has done her M.phil in area of Data mining in Manomanian University, Tirunelvelli. She has done her Post graduate degree M.sc Bharathiar University, Coimbatore. She has Presented & Published a Number of paper in reputed Journals. She has six years of teaching experience and research interests include Datamining, Textmining, Semantic webmining.