Asian Journal of Information Technology 12 (11-12): 369-375, 2013
ISSN: 1682-3915

Efficient URL Caching in a Web Caching System to Improve the Search Performance

[1]S. Kalarani and [2]G.V. Uma
[1]Department of Information Technology,
St. Joseph's Institute of Technology, Chennai, India
[2]Department of IST, Anna University, Chennai, India

Abstract: As the WWW has grown exponentially in last two decades, every day the search engines process billions of queries. The web users and web applications are growing in number and this will lead to increase in latency, network congestion and server overloading. Proxy server caches and personalized client caches are suggested to overcome these problems. They are used to store a subset of seen urls in the main memory. When correctly deployed, web caching reduces the access latencies and band width consumption. The heart of the caching system is its replacement policies which determine when and what to evict from the cache. In this study, researchers have implemented a combined LRU and LFU algorithm based on recency and frequency for a Client Cache System. Researchers propose a personalized Cache System with transparent proxy server which employs the above policy. The simulation results show that the proposed replacement policy performs better than other algorithms like LRU, LFU and FIFO.

Key words: Web caching, personalized cache, transparent proxy server, LRU, LFU

INTRODUCTION

The World Wide Web is a distributed global system of inter connected networks with the massive increase in the number of web users, the web browsers are supposed to process billions of information resources. As WWW is a network of networks that consisting of billions of interlinked hypertext documents, satisfying each user request in an efficient manner may not be possible. Web caching is one of the best ways to speed up the information retrieval process, to store the subset of already requested or seen urls in the cache memory. A cache stored roughly 50,000 entries can achieve a hit rate of almost 46%.

Web caching plays a vital role to reduce the access latency, server load, network congestion. Caching can be deployed at three levels: client level, proxy level and original server level. Therefore for achieving a better performance, researchers implemented the system with a personalized client cache and transparent proxy cache. Transparent proxy cache server can be deployed in two ways, one at the switch level and another at the router level. In the research, researchers deploy it at router level.

LITERATURE REVIEW

Buckley and Lewit (1985) is one of the earliest research, take a term at a time approach. Markatos (2001) uses the existence of temporal locality in queries and compares the performance of different caching policies. Lempel and Moran (2003) propose probabilistic driven caching, estimates the probabilistic distribution of all possible queries. Saraiva *et al*. (2001) propose a two level dynamic caching system. Their target is to improve response time for search engines. Baeza-Yates and Saint-Jean (2003) uses three level index organization.

Baeza-Yates *et al*. (2007) introduces an algorithm for static caching of posting list that improves the static caching and obtaining high hit ratio. Fagni *et al*. (2006) boosting the performance of web search engine by two level cache system. One as static cache to process historical data, other as dynamic cache to process remaining data. Serrano *et al*. (2010), presented e-Caching a consistent multi-tier Caching System. e-Caching avoids the inconsistencies that might appear when combining independent caching systems at different tiers. Their evaluation shows that although e-Caching maintains data consistency, it provides a noticeable performance improvement by combining caching at multiple tiers. Broder *et al*. (2003) uses a cache memory to store a subset of seen URLs. They achieve a hit rate of 80%. In their system, they use Mercator crawler architecture. Dhawaleswar (2012) discuss cache ejection policy in case of cache saturation. A respose time gain factor is included in the web object replacement algorithm with size heterogeneity of a web object for performance improvement of response speed.

Corresponding Author: S. Kalarani, Information Technology, St. Joseph's Institute of Technology, Chennai, India

Caching: Web caching technology can be classified into three categories according to the locations where the objects are stored. These are given.

Client's side caching: In the client side caching, web objects are cached in the client's local disk. If the user accesses the same object more than once in a short time, the browser can fetch the items directly from the local disk, eliminating the repeated network latency.

Client browser caching: Client's browser caching refers to caches that are built into most web browsers which cache internet objects for a single user but from a variety of servers. A cache is located in the browser and the proxy. Since most users visit the same web site often, it is beneficial for a browser to cache the most recent set of pages downloaded.

Client side proxy caching: In the client side proxy caching, objects are cached in the proxy near the clients to avoid repeated round-trip delays between the clients and the web servers. In addition, cooperative caching architectures enable the participating proxies to share their cache content with one another. Since, each participating proxy can seek for a remote cache hit from other participating proxy's cache, the overall hit ratio can be further improved. This study focuses on proxy caching and client side caching, one form of web caching which aims to reduce the overall bandwidth consumption of a network. Some pit falls of this method which includes: when the cache memory is full and a new request is to be cached, some item in cache ought to be removed to give opportunity to the arriving item and when an item in the application server is updated, how the same item in cache can be updated, so that the web proxy server does not return a stale copy of a result link to the requesting client. to obtain the updated copy. Researchers propose to update the web cache with a certain time frame. Cache updating by a server follows a clairvoyant algorithm to update its content. This algorithm is suitable for updating the stored results. It knows very well where its items are cached. When a result is modified, server notifies the caches and gives them an updated version of the same result. Caches always assume that they have the latest contents. Updating of results stored in the web cache can be executed automatically on a scheduled time.

Whenever the stored results are updated from the server, the proxy cache makes the notification to the server on the periodic basis for all its stored items. Then, server accept the request and if there is modification in the web page the web pages of the cache will automatically updated.

HYBRID FACTORS BASED INFORMATION RETRIEVAL

Now a days, researchers are witnessing the vast amount of information available over the world wide web. As the web grows, more and more new general-purpose and domain-specific information services assist the user in searching for the relevant data. The proposed web information retrieval system retrieves information from the CLRLFU based arranged cache as well as from the server.

Figure 1 illustrates the process of the proposed system. The query given by the user is preprocessed first and then all the vicinity of every term in the query is found, subsequently the related documents of query terms are retrieved from the cache memory which uses the recency and frequency. The relevant documents of the remaining query term which are not having the relevant documents in the cache are retrieved from the server.

Query preprocessing: From the literature survey, it is observed that the query frequencies follow a power law distribution. That implies the repetition of many queries that appear very often. A few of the queries are singleton queries, appearing just once. If the repeated query frequency is increased then the result cache will achieve high hit rates. Like normal search engine the system will accept the user query as a simple text. Preprocess the query using simple natural language processing and find the related terms that is sequence of search terms. The following simple natural language processing is applied over the user query:

- Stop word removal
- Stemming
- Indexing

Stop word removal: Using this method, researchers can find a proper sequence of words from the given query. Stop words can be removed by maintaining a list of stop words like studys, preposition, etc., count number of frequency of terms, ignore the words with high frequency words like he, she, that, this, etc.

Every query included in the information retrieval system requires preprocessing which is carried out to change the unstructured queries into structured format. The pre processing steps are as follows: at first, all the words which have low content discriminating power except logical words (and, or, not) are removed. Example of such words are a, an, the, that, it, he, she, etc. A query is tokenized into set of strings separated by some delimiters for example space. These tokens (terms) may

represent words, phrases or any key word patterns. Let Q be a given query then the tokenized keywords would be:

$$q = \{q_i | 1 < i \leq n\} \quad (1)$$

Stemming: Stemming is used to improve quality. During the stemming process morphological variations are reduced common root form removing all white spaces. After stemming, researchers need to assure word sense disambiguation for each term. Any disambiguation system might be able to receive an input as an unrestricted text and find related words for each term present in the query with most likely sense with reasonable accuracy and efficiency. Researchers define the disambiguation process based on WordNet.

It consists of nouns, verbs, adjectives and adverbs, organized in terms of heir meanings which include synonyms, antonym, hyponymy, metonymy and holonymy, lexicalized concepts are represented as a set of synsets called synonyms. Synsets are basic elements of WordNet. This method does not need any domain ontology, any training. It directly uses word sense tags from the WordNet, a lexicon database. Next to the tokenization, all the vicinity of every keywords are found which is represented as:

$$K = \{k_i | 1 < i \leq n\} \quad (2)$$

$$k_i = \{V_j \text{ Vicinity of } k_i | 1 < i \leq n; 1 < j \leq m\} \quad (3)$$

$$V_j = \{V_j \cup W \text{ if } W \in H_1 \text{ or } W \in H_2\} \quad (4)$$

Where:

m = A real number which varies for every token

H_1 and H_2 = Hyponyms

Finally all the query term which are used for information retrieval is as follows:

$$I = q \cup K \quad (5)$$

A query Q is represented as q. Since, q is related with all terms in its vicinity K, the documents representing q includes all the term occurring in K.

Indexing: After removing stop words and stemmed remaining terms, researchers have a list of related terms. To get a quick look up on terms, researchers use a database index data structure. This structure is used to store the related terms along with the associated URLs. Each time when a new term gets processed, it is stored along with an index. The index is a random number that is generated automatically. For this, researchers can use a dense index in the database. In this data structure, the database is a file with a pair of keys (random number) and points for every record (containing a list of URLs) in the data file. In this method, an index is used to quickly locate the data record from which the required data is read. To store occurrences an inverted list approach is adopted as shown in the Fig. 1. To find the terms, Google the ID 123 will be useful. However, an index (ID, Term) contains the data field and eliminates the need to look up the entire content.

Transparent cache proxy server: In order to improve the performance and reduce the server work load in world wide web, researchers deploy a transparent cache proxy server. The architecture of transparent cache proxy server is shown in the Fig. 2. Initially the client request is passed through a proxy cache server which makes the request to the original origin server.

The response of the server is retained in the proxy server and a copy is passed on to the client. If the same request is passed again to the proxy server from the same client, the response can be generated from the client cache. If the request is raised by another client, the response will be generated from the cache proxy server without further reference to the original source. Each client can be attached to the proxy server after proper authentication.

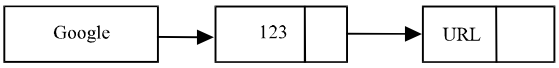

Fig. 1: List structure to store related terms

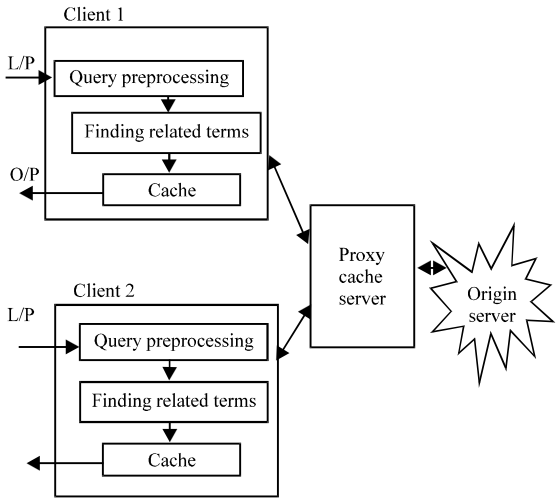

Fig. 2: Architecture of the system

The client cache has an upper bound of 1MB in size. If the client cache reaches its upper bound, least recently used information is evicted from the cache, based on aging and create a room for the new request. In the web crawler context, the number of visited URLs is too large to store in main memory. So, researchers designed the system which stores the visited urls in a small portion in the main memory in the cache. Both the client cache and transparent proxy cache are used to store equal sized atomic items. Here, the atomic items are url address which occupies 8-10 bytes.

This model has many advantages. Every cache at some time provides the user with stale data with respect to their master copy on the web server. The content (proxy server cache) server must update its content with original master copies periodically so that it can provide the result which is as fresh as possible. Standard accounts that send >200,000 visits per day to Google Analytics will result in the reports being refreshed only once a day. Daily processing begins at 12:00 UTC and continues for approximately 10 h. So, the content server must update its content automatically for every 24 h in order to keep the content fresh. In that way the stale data can be removed from its location.

Retrieval phase: Let $C = \{C_i | 1<i\leq l\}$ be the cache memory used by the proposed system where, 'l' is the size of the cache. When user requests for a query, the mapping of every query term to a web log data in cache is computed by measuring the distance between query vector and web log data vector. The result is the ranked list of documents. The document with the highest cosine coefficient similarity represents the user-intended meaning of term 'I'. The cosine coefficient similarity between two vectors $w_i = (w_{il}, w_{i2}, ..., w_{iL})$ and $v_i = (v_{il}, v_{i2}, ..., v_{iL})$ is calculated by using the following equation:

$$S(w_i, w_j) = \frac{\sum_{y=1}^{L} w_{iK} w_{jk}}{\sqrt{\sum_{k=1}^{L} (w_{ik})^2 \times \sum_{k=1}^{L} (w_{jk})^2}} \quad (6)$$

Each query term is mapped with the cache and the related documents are retrieved from the cache memory. If all the related documents are available in the cache, the documents are retrieved successfully; otherwise, the proposed system searches the remaining documents from the server ach client has its own cache. When the cache is saturated, a request for new item arrives which is not in the cache the miss occurs. A good replacement algorithm is needed to decide which item has to be evicted from the cache and allocate a space to store the new request. For this purpose, researchers have used combined LRU and LFU algorithm (CLRCFU). In LRU (Least Recently Used) algorithm the item that has not been requested for the longer time in the cache has a high priority to evict from the cache. This algorithm is based on the intuition that any item that has not been needed for a long time in the past will not be needed for long time in the future. Therefore, it minimizes the number of misses. LRU is basically very effective. However, it requires to maintaining a priority queue of request.

LFU the simplest method to employing an LFU algorithm is to assign a counter to every item that is loaded into the cache. Each time a reference is made to that item the counter is increased by one. When the cache reaches capacity and has a new item waiting to be inserted the system will search for the item with the lowest counter and remove it from the cache.

In the proposed system a combined LRU and LFU is used to avail the advantages of both the methods. CLRLFU is used to take an URL out of the cache and create a place for the new retrieved results when the cache reaches its full condition. It is based on combined recency and frequency as CRF. Let us define Combined Recency and Frequency (CRF) as:

$$CRF = \sum_{i=1}^{k} T(Tc - Ti) + Tc \quad (7)$$

Tc = Current time
Ti = Time at which the url has been referenced
K = No. of URL's in the cache

CLRLFU algorithm has to take an URL out of cache, it will choose CRF with minimum value. Because that is the one which is least recently used. If more than one URL has the same CRF value then it takes the frequency count (less frequently used) also to resolve the problem.

- Least recently used-based on date
- LRU-k no of times each URL referenced
- K-min-least frequently used

CLRLFU algorithm:

- Get the number of URLs to be inserted
- Get min CRF value for a stored related terms
- Remove their associated URLs from the cache
- Insert new set of URLs in to the cache

Old date and min k value URL's are removed first. If more than one set of URLs has the same CRF value then it takes the frequency count also to resolve this problem.

Cache updating and work flow: According to Fig. 3 the developed system works as follows. First activate the server; it automatically updates its content with the origin server once in every 24 h. Once the proxy server gets connected with the server, it updates its own content to avoid stale data. After it is gets ready to listen the clients, activate the client. The developed system can also be extended to multiple clients which forms a cluster of cache clients. Each client must be attached with transparent cache proxy server after proper authentication. Each client has its own cache apart from the content server cache. The data which is stored locally to individual client cache updates its content with the proxy server cache automatically to maintain cache coherence. Whenever the content server cache get updating, the same can be reflected in the individual cache also. The proxy server differentiates between various clients, using an Authenticated Security Model. Reduction of the delay between the client request and the content delivery will improve the performance effectively. For example, if various clients posting a same set of request for same copies of information content, the network carries duplicate of the data to each clients. That increases network congestion. But with transparent cache proxy server, importing the content just once and then passing the local copies to each client will improve the efficiency of the network. When each user gets connected with server, a client frame is opened to do authentication. By this way, researchers can create a secured information sharing. When a new query is given by any one the client named k1, it is fresh query, it does not have a matching URL list both in the client cache and also in proxy server cache. Response time will be reasonable. So, it directly gets processed by the origin server. If the already processed query is again given by the same client, the stored URLs list will be dispalyed from the client cache. This time the response time is smaller as comapred to previous search. The developed system supports multiple clients. So, next when the same query is raised bt another new client say k2, the result has been displayed from the proxy server's cache (Fig. 4).

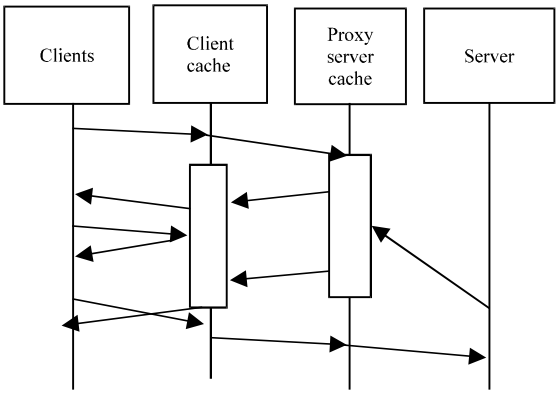

Fig. 3: Work flow

Batch query processing-cache update: Batch query processing can be used to keep the cache content up to date and fresh. In order to meet the challenges, researchers should keep the cache content up to date. This updation can be done by batch query processing. To keep the stored result up to date, periodically refresh the content by issuing the batch of stored query result to the search engine. As compared to interactive queries large batch of queries can be processed more efficiently. Researchers can reduce the cost of updation by submitting the batched queries during off-peak times. Batch query processing uses the following approach.

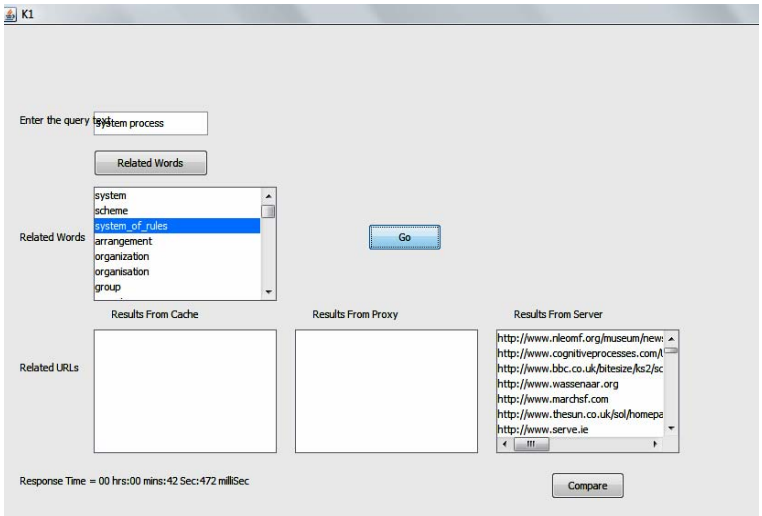

Fig. 4: Developed system's user interface screen

Clairvoyance: The aim is to refresh the content of proxy server cache with the original server. At the same time refresh the content of client's cache with proxy server content. In both the situations, researchers know the previously stored result in advance. So, researchers can use Clairvoyance algorithm. This algorithm is best when researchers know all future queries.

Performance analysis: The retrieval time drastically decreases over the each retrieval. It was reasonable during the first time search. As the result should obtained from the origin server. Whereas a subsequent request for the same query by the same client and different clients had only a minimum access time which shows effective performance. As shown in the graph of Fig. 5 gives the access latency for a single client. In case of multiple clients the access latency is still decreased because of the effectiveness of proxy cache. These performances are compared after running the system with necessary queries in Fig. 6-8. The number of terms in a query has a direct impact on the number of related words. Related words count will significantly improve the information search.

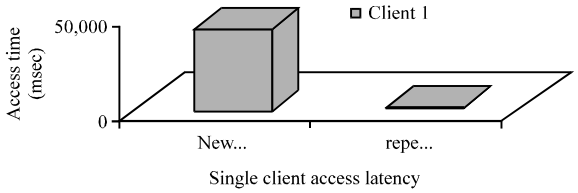

Fig. 5: Access latency for a single client

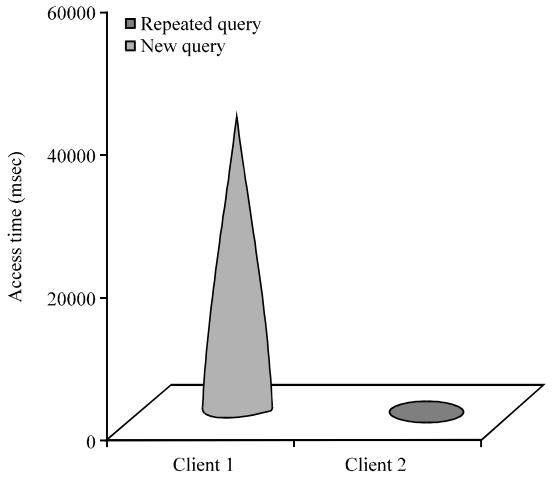

Fig. 6: Access latency for multiple clients

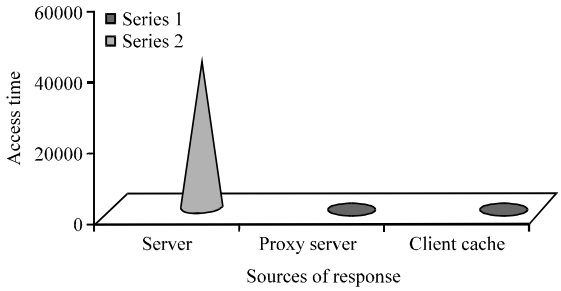

Fig. 7: Various sources of response

Fig. 8: Comparison frame work

CONCLUSION

After running 1000 queries, the conclusion is that URL caching is most effective and it can achieve an increased hit rate of almost 46%. It mainly reduces server researchload and network traffic.

The future research focused on how to create privileged information environment. Because of this it deliver privileged information to certain clients. Individual client identification can be used to create public and virtual private information services where a copy of authenticated clients can be directed to access privileged information while the others are passed to the public default content.

REFERENCES

Baeza-Yates, R., A. Gionis and F. Junqueira, V. Murdock, V. Plachouras and F. Silvestri, 2007. The impact of caching on search engines. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 23-27, 2007, New York, USA., pp: 183-190.

Baeza-Yates, R.A. and F. Saint-Jean, 2003. A three level search engine index based in query log distribution. String Proc. Inform. Retrieval, 2857: 56-65.

Broder, A.Z., M. Najork and J.L. Wiener, 2003. Efficient URL caching for world wide web crawling. Proceedings of the 12th international conference on World Wide Web, May 20-24, 2003, Budapest, Hungary, pp: 679-689.

Buckley, C. and A.F. Lewit, 1985. Optimization of inverted vector searches. Proceedings of the 8th Annual International SIGIR Conference on Research and Development in Information Retrieval, June 13-15, 1985, Montreal, Canada, pp: 97-110.

Dhawaleswar, R.C.H., 2012. Study of the web caching algorithms for performance improvement of the response speed. Indian J. Comput. Sci. Eng., 3: 374-379.

Fagni, T., R. Perego, F. Silvestri and S. Orlando, 2006. Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. ACM Trans. Inform. Syst., 24: 51-78.

Lempel, R. and S. Moran, 2003. Predictive caching and prefetching of query results in search engines. Proceedings of the 12th International Conference on World Wide Web, May 20-24, 2003, Budapest, Hungary.

Markatos, E.P., 2001. On caching search engine query results. Comput. Commun., 24: 137-143.

Saraiva, P.C., E.S. de Moura, N. Ziviani, W. Meira, R. Fonseca and B. Riberio-Neto, 2001. Rank-preserving two-level caching for scalable search engines. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, LA, USA.

Serrano, D., S. Bouchenak, R. Jimenez-Peris and M. Patino-Martinez, 2010. Multi cache coherence protocol for distributed internet services. http://lsd.ls.fi.upm.es/lsd/papers/2011/Marta_Multi-cache%20coherence%20protocol.pdf.