Effective Information Retrieval using Co-Operative Clients and Transparent Proxy Cache Management System with Cluster based Prediction and Pre-Fetching

S.Kalarani and Dr.G.V.Uma
1 St Joseph’s Institute of Technology/Information Technology, Chennai, India
Email: kala_subramanium@yahoo.co.in
2 Anna University/Information science and Technology, Chennai, India
Email: gvuma@annauniv.edu

Abstract—In this work we attempted to develop the techniques for co operative client cache and transparent proxy cache managers for effective information retrieval. As the World Wide Web 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. When caching is used, the request for content need not make multiple trips back and forth to the clients and the where the content is stored. Proxy server caches and personalized client caches are suggested to overcome these problems. The simulation results show that our proposed hybrid replacement algorithm performs better than other algorithms like LRU, LFU and FIFO.

Index Terms—web search, proxy server cache, clients log profile, clustering, pre fetching.

I. INTRODUCTION
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 way to speed up the information retrieval process, to store the subset of already requested or seen urls in the cache memory and corresponding web object in the multilevel cache system. Web caching plays a vital role to reduce the access latency, server load, and network congestion. Web pre-fetching is an additional technique to improve the web caching mechanism. Predicted objects are fetched from the origin server and stored in client cache in advance, which increases the cache hits and reduces latency. Caching can be deployed at three levels: client level, proxy level and original server level. Therefore, for achieving a better performance, this system is implemented with a personalized client cache in addition with transparent proxy cache along with cluster based web pre fetching technique.

© Elsevier, 2014
II. Related Works

In the earliest works, researchers use term-at-a-time approach. Markatos[1], use the existence of temporal locality in queries, and compare the performance of different caching policies. Lampel and Moran [2] have proposed probabilistic driven caching, estimates the probabilistic distribution of all possible queries. Saratva, Moura and maria [3], propose a two level dynamic caching system. Their target is to improve response time for search engines. Baeza-Yates and Saint-Jean [4], uses three level index organization. Richardo Baeza-Yates, Aristides Gionis and Flavio Junqueira [5], introduce an algorithm for static caching of posting list that improves the static caching and obtaining a high hit ratio. Tiziano Fagni, Raffaele Perego and Fabrigo siverstri [6], explain the 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. Damian Serrano, Sara Bouchenak, Ricara ana Marta [7], 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. Andrei Z Broder, Marc Najork and Janet L.Wiener [8] use a cache memory to store a subset of seen URLs. They achieve a hit rate of 80%. In their system they use the Mercator crawler architecture. Dhawaleswar Rao. CH[9], discusses cache ejection policy in case of cache saturation. A response time gain factor is included in the web object replacement algorithm with size heterogeneity of a web object for performance improvement of response speed. Harshal N Datir, Yogesh H Gulhane and P R Deshmukh[10], have surveyed various cache implementation techniques like client side caching, client’s browser caching and client side proxy caching. They also analyze various replacement policies such as LRU, LFU and MFU to improve the hit rate and for maintaining personalized cache. B Barla Cambazoglu, Flavio P. Junquera and Vassilis Plachouras [11], design a result cache used in yahoo search engine and present a practical algorithm for prioritizing entries to refresh. The refresh mechanism uses access frequency and age of cached entries to effectively select entries for refreshing. Qingqing Gan and Torsten Suel [12], they proposed an improved techniques for result caching in web search engines, by using Zipf-based query distribution and feature based approach to caching that achieves an improvements in hit ratio.

Xiaohui Long and Torsten Suel[13], propose a new three level caching architecture for web search engines that can improve query throughput. The three different caching includes result caching, intersection caching and list caching. Performance gets improved due to an extra level of caching. Anupam Bhattacharjee and Biolob kumar Debath[14], they combine Random replacement and Least recently used schemes, to replace a web page from a web cache. This algorithm improves the hit ratio reasonably. Abdullah Balamash and Marwan Kunz[15], accommodates both passive client and proxy caching along with pre fetching.

The main objective of this work is to improve the efficiency of retrieved information by using co-operative clients and transparent proxy caching system. Nowadays we are witnessing the vast amount of information available over the World Wide Web.

This approach includes

- Information Retrieval using single client cache manager.
- Information Retrieval using co-operative clients and transparent proxy server cache manager.
- Improving the web caching by integrating cluster based web prediction and web pre-fetching.

The proposed web information retrieval system retrieves information from the client’s cache as well as from the transparent proxy cache and the origin server. Fig. I illustrates the system architecture of the developed system. The query given by the user is pre-processed 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 proxy server cache. Still the required information is not present in the proxy cache also, and then it searches with the origin server. The retrieved results are updated subsequently.

The web pre-fetching is an effective way of utilizing the caching mechanism and improves the information retrieval. Pre-fetching technique has two modules. Prediction module and pre-fetching module. The web caching and web pre-fetching uses the temporal and spatial locality principles. The prediction module predicts the web objects based on spatial locality such that fetch the most likely web objects before the user request the object. The predicted objects are pre-fetched from the server and stored in the proxy server cache as well as in the client’s cache in advance. Web pre-fetching will help to improve the cache hits and reducing the user perceived latency by keeping the expected objects very close to the client side.
III. INFORMATION RETRIEVAL USING SINGLE CLIENT CACHE MANAGER

In this present work, Web cache management in an information retrieval is an important problem. After receiving the query from the client, the client cache manager pre-processes it and find the appropriate result in its own cache. If the result of the submitted query is already in the cache, then there is no need for further processing and it can be sent to the user. If the query result cannot found in the result cache then the system contacts to the origin servers for query processing. The important issue in caching system is how to decide what to cache. The traditional cache method LRU (Least Recently Used) which uses the time of last reference to each page is not sufficient for effective web information retrieval. The LRU based caching system uses the recency of retrieval whereas the MRU (Most frequently used) based caching system uses the frequency of the retrieval. Though researches used various factors which determine the likelihood of document to be accessed near future, the recency, occurrence frequency, size and access latency are important factors for ranking process in caching system. The developed caching system uses the hybrid factors such as frequency of occurrence, size, and cost of fetching the object in to the caching system.

A. Query Preprocessing

Each user query in the information retrieval system requires pre-processing which is carried out to change the unstructured queries into structured format. The following simple natural language processing is applied over the user query, tokenization, and stop word removal, stemming, and indexing.

Let \( Q \) be a given query then the tokenized keywords would be

\[
q = \{ q_i \mid 1 < i \leq n \}
\]  

Next to the tokenization, all the vicinity of each keywords are found with the help of wordNet which is represented as

\[
K = \{ k_i \mid 1 < i \leq n \}
\]

\[
k_i = \{ V_j \text{ Vicinity of } k_i \mid 1 < i \leq n; 1 < j \leq m \}
\]

\[
V_j = \{ V_j \cup W \text{ if } W \in H_1 \text{ or } W \in H_2 \}
\]

Where ‘m’ is a real number which varies for every token \( H_1 \) is hyponyms and \( H_2 \) is hyponyms. Finally all the query term which are used for information retrieval is as follows

\[
l = q \cup K
\]

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 \( K \).

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.

B. Hybrid Cache Algorithm

This hybrid algorithm consider the factors such as frequency of occurrence, size and cost which are the main factors for caching. For each web object ‘g’ requested by user,

1. if g is in cache (cache hit occurs)
   update \( k(g) \)
   \[
   k(g) = f(g) \times \frac{c(g)}{s(g)}
   \]  

2. if cache miss occurs
   while no enough space in cache for g
   calculate \( \min(k(q)) \) for each q in cache
   Evict q

3. Fetch g into cache from origin server
C. Context Adding Algorithm

The proposed system is simulated in phases. The first phase takes the training data set and retrieve the result using search engine. The second phase utilize the results from the first phase and find the context of each result. To add appropriate context the following algorithm is used. This phase will update the user profile with context.

Input: Query terms & set of previous search result from the cache
Output: urls
IV. INFORMATION RETRIEVAL USING CO-OPERATIVE CLIENTS AND TRANSPARENT PROXY SERVER CACHE MANAGER

In order to improve the performance and reduce the server work load in World Wide Web, in the present system uses client cache and a transparent proxy server cache. Web client which is going to participate in co-operative cache system are initially get connected with the proxy server after proper authentication by user names and IP address. Initially the developed system is run under training data in various fields such as travel, education and sports. A proxy cache manager will maintain a dynamic client log profile for all the co-operative clients. LRU algorithm was used to manage one part of the cache. Web objects which are used more than once are moved from lower part of the cache into upper part of the cache. LFU access frequency based algorithm is used to manage this part of the cache. Each web client issues their object request to the client cache manager. Client cache manager check the requested object availability. If the cache hit occurs it send the result immediately to the client with minimum latency. The response time is very fast, it returns the result without affecting the server workload. If the requested object not in client side cache that is a cache miss occurs then the request is given to proxy cache manager, which maintains the web objects based on co-operative clients log profile. If the proxy cache manager finds the requested web object then it returns the objects directly to the particular client cache manager, which returns the result to the client and maintains the retrieved result in its own cache. If the proxy server does not find the matching object, then the request will be given to origin server like a normal search. The result returned from the origin server should be updated in to the proxy server cache. The proxy server cache manager will maintain a list of accessed web uhrs and the corresponding web objects for the entire co-operative client caches request.

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. In the developed system client cache has an upper bound of 1MB in size. If the client cache reaches its upper bound, least recently based hybrid algorithm is to evict an object from the cache, and creates a room for the new request.

In the web crawler context, the number of visited urls is too large to store in main memory. In the designed system, visited urls are stored 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 to 10 bytes and the retrieved web objects are stored in the next level cache. So searching the availability of query content is very less, because it is performed only over the visited url list not with the web contents. The original web objects are stored in lower level cache.

V. WEB CACHING BY INTEGRATING CLUSTER BASED WEB PREDICTION AND PRE-FETCHING

The proxy server cache manager will run the pre-fetching module. It creates clusters according to the proxy log file. In order to create clusters first 60% of total request is used as training data and the rest 25% used as testing data. The proxy log file must include the following unique details as per the client past access. It contains client address, time stamp, duration, result codes, bytes, and URL. According to web mining the
web usage mining will be taken into account for prediction and pre-fetching. The proxy log is treated as web usage information. These data are pre-processed, and patterns are discovered and analyzed through clusters. Pre-processing removes many entries which are not necessary for further processing. It removes the log entries with extension like gif, jpeg, jpgc. It removes log entry with code 404, which means the requested resource is not existence. The entire proxy access is represented as a web navigational graph. The graph $G(u,v)$ is constructed as follows. Each node $u$ represent a web page(URL address), each edge $v$ represent a set of users navigation from one page to another. The weight of node represent the popularity of the URL, and weight of edge represent number of transition into the set. The initial navigational graph is vast due to the amount of web pages. An association rule mining should be used to create sub graphs. Each sub graph represents a cluster. The nodes attached to that cluster are URLs of user similar interest. The sub graphs are formed based on clustcreate algorithm. This algorithm follows breadth first search traversal of a graph. It takes a node from the graph and marks it as source node, and find each node that can be visited from the source node. In doing so, it outputs a sub graph, which includes all the nodes reached from the source. This procedure is iterated till all the nodes are marked in the initial graph. Now each cluster will represent a preference list. After the training phase, the following steps are taken to satisfy the user request.

Step 1: The proxy identifies the clients according to IP address.
Step 2: Once it receives the search request then it searches inside the existing clusters to find in which of them are exists.
Step 3: The proxy pre-fetches all the objects that exists in the cluster from the origin server.
Step 4: These objects are managed by the proxy cache manager. The proxy sends the requested objects to the client.
Step 5: if the client request does not have a matched object in any of the existing cluster, the it automatically send it to origin server. It this as a new entry in proxy log file.

\[ \text{Clustcreate algorithm} \]
\[ \text{Input } G(i) = \text{web navigational graph}, n = \text{number of clients} \]
\[ \text{Popularity of each node.} \]
\[ \text{Begin} \]
\[ \text{For all clients } C, C \in C \text{ do create a subgraph } S(i) \]
\[ S(i) = 0; \]
\[ \text{Repeat} \]
\[ \{ \]
\[ S(i) = \text{BFS}(G(i)); \]
\[ S(i)++; \]
\[ \} \text{ until all nodes of } G(i) \text{ are traversed.} \]

In this co-operative cache system, all participating clients share their cache content with one another. Once all the pages are pre-fetched into the cache, a combined LRU and LFU hybrid algorithm manages these web pages.

VI. Evaluation Results

The performance metrics used to evaluate the system are hit rate, byte hit rate and precision. These measures are evaluated for simple one word query, simple multi word query and complex multiword query. The proposed hybrid algorithm byte hit ratio is compared against the conventional algorithms and is shown in the table 1 as follows.

After simulation the client cache replacement based on the hybrid algorithm performs well than the traditional algorithm like LRU,LFU,FIFO. The comparison of byte hit ratio of the proposed hybrid algorithm with tradidtional techniques are shown in the fig 2. It improves the byte hit ratio approximately 10.6%. Training data sets were executed initially, and the desired features of web objects are extracted from the origin. Based on this client log profile and proxy log profile are created.

The below table 2 shows the example pre-processed data extracted from the proxy server. The retrieval time is drastically decreases over each retrieval. It was reasonable during the first. Because the results should be obtained from the origin server. Whereas subsequent request for the same query by the same client and different clients had only minimum access time, which shows effective performance and is given as a comparison graph in fig 3 as follows.
### Table I: The analysis of byte hit ratio

<table>
<thead>
<tr>
<th>No of request</th>
<th>Replacement algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LRU</td>
</tr>
<tr>
<td>100</td>
<td>0.4</td>
</tr>
<tr>
<td>200</td>
<td>0.486</td>
</tr>
<tr>
<td>300</td>
<td>0.495</td>
</tr>
<tr>
<td>400</td>
<td>0.51</td>
</tr>
<tr>
<td>500</td>
<td>0.563</td>
</tr>
<tr>
<td>600</td>
<td>0.591</td>
</tr>
<tr>
<td>700</td>
<td>0.652</td>
</tr>
<tr>
<td>800</td>
<td>0.689</td>
</tr>
<tr>
<td>900</td>
<td>0.671</td>
</tr>
<tr>
<td>1000</td>
<td>0.622</td>
</tr>
</tbody>
</table>

![Chart showing byte hit ratio analysis](image)

Figure 2 Hit ratio analysis

### Table II: Example pre-processed data extracted from proxy server cache

<table>
<thead>
<tr>
<th>URL ID</th>
<th>No of access</th>
<th>Response time (msec)</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.exploreworldwide.com">http://www.exploreworldwide.com</a></td>
<td>1</td>
<td>17</td>
<td>33070</td>
</tr>
<tr>
<td><a href="http://www.explore.co.uk">http://www.explore.co.uk</a></td>
<td>1</td>
<td>23</td>
<td>14589</td>
</tr>
<tr>
<td><a href="http://www.myexplore.co/en">http://www.myexplore.co/en</a></td>
<td>1</td>
<td>22</td>
<td>1456</td>
</tr>
<tr>
<td><a href="http://www.oradelights.com">http://www.oradelights.com</a></td>
<td>1</td>
<td>34</td>
<td>26894</td>
</tr>
<tr>
<td><a href="http://www.theorade.com">http://www.theorade.com</a></td>
<td>2</td>
<td>12</td>
<td>33070</td>
</tr>
<tr>
<td><a href="http://www.orade.com">http://www.orade.com</a></td>
<td>1</td>
<td>35</td>
<td>6892</td>
</tr>
<tr>
<td><a href="http://www.orade.com/technetwork">http://www.orade.com/technetwork</a></td>
<td>1</td>
<td>23</td>
<td>78651</td>
</tr>
<tr>
<td><a href="http://www.orade.com">http://www.orade.com</a></td>
<td>2</td>
<td>45</td>
<td>26894</td>
</tr>
<tr>
<td><a href="http://www.data.org">http://www.data.org</a></td>
<td>3</td>
<td>10</td>
<td>26894</td>
</tr>
<tr>
<td><a href="http://www.information.dk">http://www.information.dk</a></td>
<td>3</td>
<td>9</td>
<td>33070</td>
</tr>
<tr>
<td><a href="http://www.computerhistory.org">http://www.computerhistory.org</a></td>
<td>4</td>
<td>9</td>
<td>26894</td>
</tr>
</tbody>
</table>

Number of terms in a query has a direct impact on number of related words. Related words count will significantly improve the information search. Based on the simulated results, the origin server returns 10 most recently visited pages for each term in the given query. Client cache of size 1MB can store approximately 3879 related URLs. WorldNet can generate minimum 3 to maximum 7 related terms for individual terms. Instead of wordnet we can connect extended wordnet or existing domain ontologies to retrieve more context related information.
VII. CONCLUSION AND FUTURE WORK

During the training phase the system does not perform well. After running enough queries, our conclusion is that URL caching is most effective and it can achieve an increased hit rate of almost 46%. It mainly reduces server workload and network traffic. The performance is analyzed by comparing the result over LRU, LFU, FIFO and our approach, which reveals that precision, recall, F-score are improved. Web caching use temporal locality principle to cache the most frequently used web objects near the clients, and reduce access latency. Pre fetching will use spatial locality principle to fetch the most likely web pages before the user take the action. The future work focused on how to create privileged information environment. The entire developed system increases overall byte hit ratio and decreases bandwidth consumption of a network. Domain ontologies are used to improve the context specific information retrieval.

REFERENCES