International Journal of Computer Applications (0975 – 8887)
Volume 44– No.1, April 2012

Optimizing the Web Cache Performance by Clustering based Pre-Fetching Technique using Modified ART1

V. Sathiyamoorthi
Assistant Professor/CSE
Sona College of Technology
Salemi-5, Tamil Nadu, India

V. Murali Bhaskaran
Principal
Paavai College of Engineering
Paachal, 637018, Tamil Nadu, India.

ABSTRACT

Web caching is a technique which is used to reduce user perceived latency when user is accessing the Web pages. Web pre-fetching is a scheme where Web pages are pre-fetched into the intermediate server (proxy) cache before user accessing it. These two techniques can complement each other since the Web caching exploits the temporal locality, whereas Web pre-fetching utilizes the spatial locality of Web objects. In this paper, we developed modified ART1 neural network to pre-fetch Web pages into the proxy cache. We have also empirically shown the performance of proposed work with the existing ART1 based pre-fetching. By using this approach the hit rate of the cache increases, which in turn reduces the user perceived latencies.

General Terms

Data Mining, Web Mining, Web Usage Mining, Web Caching

Keywords

Web pre-fetching, Web caching, latencies, Web log mining, Work load matrix.

1. INTRODUCTION

World Wide Web is a evolving system of interlinked files like containing audio, images, videos, and other multimedia. Here Web caching and pre-fetching is found as an important technology. Now a day’s World Wide Web is widely used and this has led to a substantial increase in the amount of traffic over the internet. As a result, the Web has now become one of the primary hold up to network performance. Transferring of objects over the networks is leading to increase in the level of traffic in the network. This increase in traffic will reduce bandwidth for competing requests and increase latencies for Web- users. In order to reduce such access latencies, it is desirable to store copies of popular objects closer to the user. Consequently, Web-caching and Web pre-fetching has become an increasingly important topic across the network to reduce the noticeable response latency perceived by users. In Web-caching technology if a client is requesting a page from proxy-server it will fetch from the server and will give response to the client. In Web pre-fetching scheme the proxy server itself will give the response to the clients if the Web page requested is present in the proxy server itself.

The Web access logs serve as a substantial source of information about users’ Web access patterns. So such logs can be well exploited to analyze and discover useful information about users’ interests.

In this paper, we have presented a tailor made ART1 clustering algorithm to group users based on their Web access patterns. The advantage of using the modified ART1 algorithm to group users is that it acclimatizes to the change in users’ Web access patterns over time without losing information about their past Web access patterns. In our modified ART1 based clustering approach, each cluster of users is represented by a binary pattern vector that is representation of the URL’s frequently accessed by all the members of that cluster. One can control the degree of similarity between the members of each cluster by changing the value of vigilance parameter.

The cluster performance between ART1 and modified ART1 algorithm is compared using the average inter and intra cluster distance for all the data sets considered. The improvement in the cache performance is shown by comparing the number of hits produced by using pre-fetching based on ART1 clustering and the pre-fetching based on out modified ART1.

The minimum cache size of proxy server is 5 MB. As we consider only last 200 requests as our actual request for the page replacement algorithm we consider the cache size to be 1, 2,3,4,5 MB’s respectively.

2. RELATED WORK

Here we discuss work done in the area of Web user clustering and pre-fetching.

2.1 Research Progress in Clustering Web Users

Clustering the users using ART1 algorithm results in high accuracy. Based on the ART1 neural network we have grouped users according to their Web access patterns. Comparison is done with the K-Means algorithm in terms of inter-cluster and intra-cluster distances resulting in a minimum average. This clustering technique is applied to the pre-fetching scheme [1].

The performances of clustering algorithm is measured using modified grouping efficiency (MGE).The modified ART1 algorithm are compared with other genetic algorithm [2].

ART neural networks are capable of developing stable clusters, whereas ART-1 will cluster binary input vector and

ART-2 will cluster real-valued input vectors. In this ART1 will more effective than ART2 because clustering the binary value is easier than real value [3].

2.2 Research Progress in Pre-Fetching

Pre-fetching URLs based on users' profiles. Each user's profile is represented by a weighted directed graph in which the nodes represent URLs and the edges represent the access paths. The weight of a node represents the frequency of access of URLs and the weight of an edge represents the frequency of access of one URL after another. Ibrahim et al. [13] present a context specific pre-fetching technique, which uses an artificial neural network for predicting users' requests. Li Fan et al. [14] investigate an approach to reduce Web latency, by pre-fetching between caching proxies and browsers. Their technique uses the Prediction by Partial Match (PPM) algorithm for pre-fetching. Evangelos et al. [15] present a simple and effective Top-10 approach for pre-fetching. In their approach, the ten most popular Web pages are pre-fetched.Padmanabhan et al [16] present a pre-fetching scheme in which the server computes the likelihood that a particular Web page will be accessed and conveys this information to the client.

The client program then decides whether to pre-fetch the Web page. The prediction is based on a dependency graph similar to the one used in [17]. The authors conclude that their methodology results in substantial reduction in Web latency, but increases the traffic on the network. Tian, Choi, et al. [17] present an intelligent and adaptive neural network predictor, which uses the back propagation learning rule to learn the changing access patterns of pages in a Web site. Most of the research discussed in pre-fetching concentrates on pre-fetching individual users' requests according to their previous access patterns. Although these methods are efficient for pre-fetching, they may considerably overload the network with unnecessary traffic when pre-fetching for a large number of users. To reduce such an effect of pre-fetching, we present a pre-fetching scheme that uses modified ART1 clustering technique to pre-fetch requests for a large community of users instead of pre-fetching individual users' requests.

Caching and pre-fetching have often been studied as separate tools for reducing the latency observed by the users in accessing the Web. Less work has been done on integration of caching and pre-fetching techniques. Kroeger et al [18] study the combined effect of caching and pre-fetching on end user latency. Yang and Zhang have proposed an Integrated Pre-fetching and Caching Algorithm using a Correlation-Based Prediction Model [19,20]. Lan et al. [21] have proposed a Rule-Assisted Pre-fetching in Web-Server Caching. Yang et al. [22, 23] have proposed a method for Mining Web Logs to obtain a Prediction Model and using the model to extend the well known GDSF caching policy. Curcio et al[24] have proposed an Integrated Pre-fetching and Caching for the World Wide Web via User Cooperation.

There are number of pre-fetching models, in that Markov reference model and Markov-based pre-fetching model are used for analysis here and their performance have been compared to find out the optimal model Lei Shi [11]. Abdullah Balamash[10] et al. discussed various cache replacement algorithms. Barbara [4] used ART1 algorithm for clustering.

3. ARCHITECTURE

The Architecture of the proposed system is depicted in figure 1. It includes the following steps:

1. Recording a client's request in the proxy servers' Web log file.
2. Feature Extractor extracts the access patterns of the log file.
3. The Access patterns extracted above is given a input to the Clustering component.
4. The Clustering component groups based on the given input.
5. Pre-fetcher requests for all the URL's from specific cluster.
6. Proxy-server responds with the pre-fetched URL Objects.

4. METHODOLOGY

In this section we describes the data pre-processing of Web logs, extracting feature of binary pattern vector ,clustering user using modified ART1 algorithm ,pre-fetching process and integration scheme.

4.1 Data Pre-Processing

For testing our research and to measure the efficiency of working, we use Web log files provided by UC sanitized Web log as input. The Web logs contain HTTP request to the host server. The requests which have URL's of extension (.html) and (.xml) are used in this research paper, because the html and xml pages are mostly static whereas others are left out. The raw data from the Web log file will in the following form:

<Timestamp, elapsed time, remote host, code/status, bytes, method, URL, peer status /peer host type >

Timestamp

A Unix timestamp as UTC seconds with a millisecond resolution.

Duration

The elapsed time considers how many milliseconds the transaction busied the cache. It differs in interpretation between TCP and UDP:

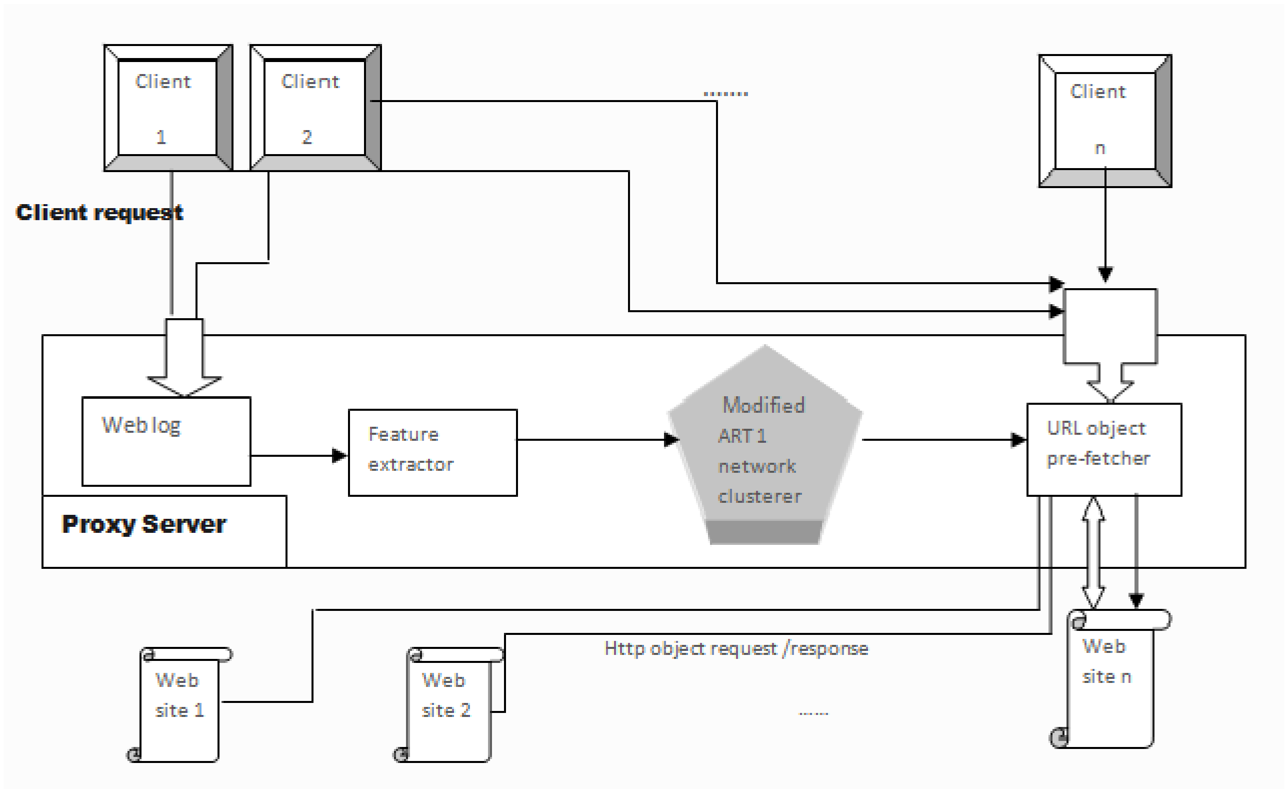


Fig 1: Proposed System Architecture

- For HTTP/1.0, this is basically the time between *accept ()* and *close ()*.
- For persistent connections, this ought to be the time between scheduling the reply and finishing sending it.
- For ICP, this is the time between scheduling a reply and actually sending it.

Client Address

The IP address of the requesting instance, the client IP address.

Result Codes

- This column is made up of two entries separated by a slash. This column encodes the transaction result:
- The cache result of the request contains information on the kind of request, how it was satisfied, or in what way it failed.
- The status part contains the HTTP result codes with some Squid specific extensions. Squid uses a subset of the RFC defined error codes for HTTP.

Bytes

The size is the amount of data delivered to the client. Also, failed requests may deliver an error page, the size of which is also logged here.

Request Method

The request method to obtain an object.

URL

This column contains the URL requested.

Rfc931

The eighth column may contain the ident lookups for the requesting client.

Hierarchy Code

The hierarchy information consists of three items:

- Any hierarchy tag may be prefixed with *TIMEOUT_*, if the timeout occurs waiting for all ICP replies to return from the neighbors.
- A code that explains how the request was handled, e.g. by forwarding it to a peer, or going straight to the source.
- The IP address or hostname where the request (if a miss) was forwarded.

Type

The content type of the object as seen in the HTTP reply header.

4.2 Feature Extraction

Once pre-processing is done a matrix is formed, each row represents the unique user and each column represents count of the individual page referred by the user. After the matrix is formed we proceed to find out the total number of requests to each page by all the users. Similarly we also find the total number of requests made by each user for all the pages. We then set the threshold value as 2 and check for pages and users greater than the threshold limit in the matrix formed and term them as frequent pages and users respectively. Thus the initial matrix is now reduced to a matrix containing frequent user and pages. From this matrix a binary matrix called workload matrix is formed by considering the requests made by the frequent user to each frequent page. Here we set the threshold as 3. Values above the threshold will be set as 1(means that the page must be pre-fetched when an actual request arrives from that particular user) and value below is assigned as 0(means that the page need not to be pre-fetched when an actual request arrives from that particular user).

Sometimes there has to be no pages pre-fetched for some user, means after applying the threshold limit the value for all the columns may become 0.So we must add a default cluster which specifies that no pages has to be pre-fetched for those users and leave it from consideration during performance measures.

In the below depicted workload matrix each row represents each frequent users' access pattern and each column represents frequently accessed pages.

The threshold values are chosen based upon the number of requests considered from the input log file. Here the number of requests considered is less (approximately 15000), so we have chosen 2 as the threshold.

$$\begin{array}{cc} & \mathbf{P} \\ \mathbf{U} & \begin{pmatrix} a_{11}\ a_{12}\ a_{13}\ \ldots\ldots a_{1m} \\ a_{21}\ a_{22}\ a_{23}\ \ldots\ldots a_{2m} \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ a_{n1}\ a_{n2}\ a_{n3}\ \ldots a_{nm} \end{pmatrix} \end{array}$$

Fig 2: Work Load Matrix

4.3Clustering

The basic idea behind this modified ART1 is to make sure that the centroid of the cluster must be in a way such that it pre-fetches all the frequent pages of all the members (frequent users). This can be achieved through changing the top down weight updation of the existing ART1.

There are two major differences between ART1 and modified ART1:

Using binary addition of the input pattern with winning column of the top down matrix instead of the usual multiplication

Using binary addition introduces a new problem which is that all the bits in the centroid of the cluster becomes '1'. To overcome this, the centroid(top down weight of the winning column) is chosen by performing the rtest between the current input pattern and all the input patterns of the users belonging to that cluster.

For clustering the binary feature vectors extracted above we use the modified ART1 algorithm which is described below.

The modified ART1 algorithm Description:

- n and m are number of frequent users and pages respectively
- Two matrices are used to hold the top down and bottom up weights. The dimensions of the top down matrix is n*m and that of the bottom up is m*n.
- The bottom up matrix is initialized with the value 1/(m+1) and top down matrix is initializes to zero.

For each row in the workload matrix

- The winners are identified by multiplying the row with all the columns of the bottom up matrix thus resulting in 'n' values.
- The columns which produces the maximum values are the winners
- If there is only one winner it is the default winner. In case of presence of more than one winner, we always choose the second as the winner

We then perform the vigilance parameter test on the winner

- If the test is passed, then the user is added to the winning cluster and top down weights are updated. If the test fails, we remove the current winner and find a new one.
- Vigilance parameter test is done to check whether the average degree of match between all the users in the cluster and the input, is more than the vigilance parameter value. If it is greater than the vigilance parameter value, test is passed.
- For updating the top down weight we perform binary addition of the values in winner's row and the input pattern values.

This modified algorithm works because it intelligently pre-fetches more number of pages than the usual ART 1 by ensuring that all the frequent pages corresponding to all the frequent users are pre-fetched. The pseudo code of modified ART1 given below:

Pseudo Modified ART1 Algorithm:

1. Initializations

Bottom-up weights (m*n)

$$bu[j][i] = \frac{1}{(m+1)}$$

Top-down weights(n *m) $td[i][j] = 0$ Where n and m are the number of frequent user and pages Also, choose the vigilance threshold ρ, $0 \le \rho \le 1$;(Note: Here we use ρ=0.5)

2. Apply the new input pattern Ai:

3. Compute the activation values Yi for the input

$$yi = \sum_{j=1}^{m} bu[j][i] * A[i][j]$$

4. Select the winner k ($0 \leq k < n$)

5. Vigilance test if

For each user in the winning cluster, the degree of deviation with their pattern and the input is calculated by comparing bit-by-bit.

If the average degree of deviation is greater than ρ go to step 6; else go to step 7;

6. Winner k is disabled from further activity. Go to step 3.

7. Update the weights

- Top-down weight updating:

$td[k][j] = td[k][j] + A[i][j]$ Where j=1 to m; '+' is binary addition

- Bottom-up weight updating:

$$bu[j][k] = \frac{td[k][j]}{(0.5+magnitude(td[k][j]))}$$

where j=1 to m and magnitude returns number of 1's in the specified.

8. Continue for the next input pattern

The performance and other metrics of the clustering algorithm are shown by comparing the metrics obtained by using ART1 on the same dataset. The performance metrics include the average inter and intra cluster distance. Here we define our own distance measure.

The distance between two binary vectors A and B is calculated using the formula below:

DAB = number of 0's in A when corresponding bit in B is 1/ (magnitude of B)

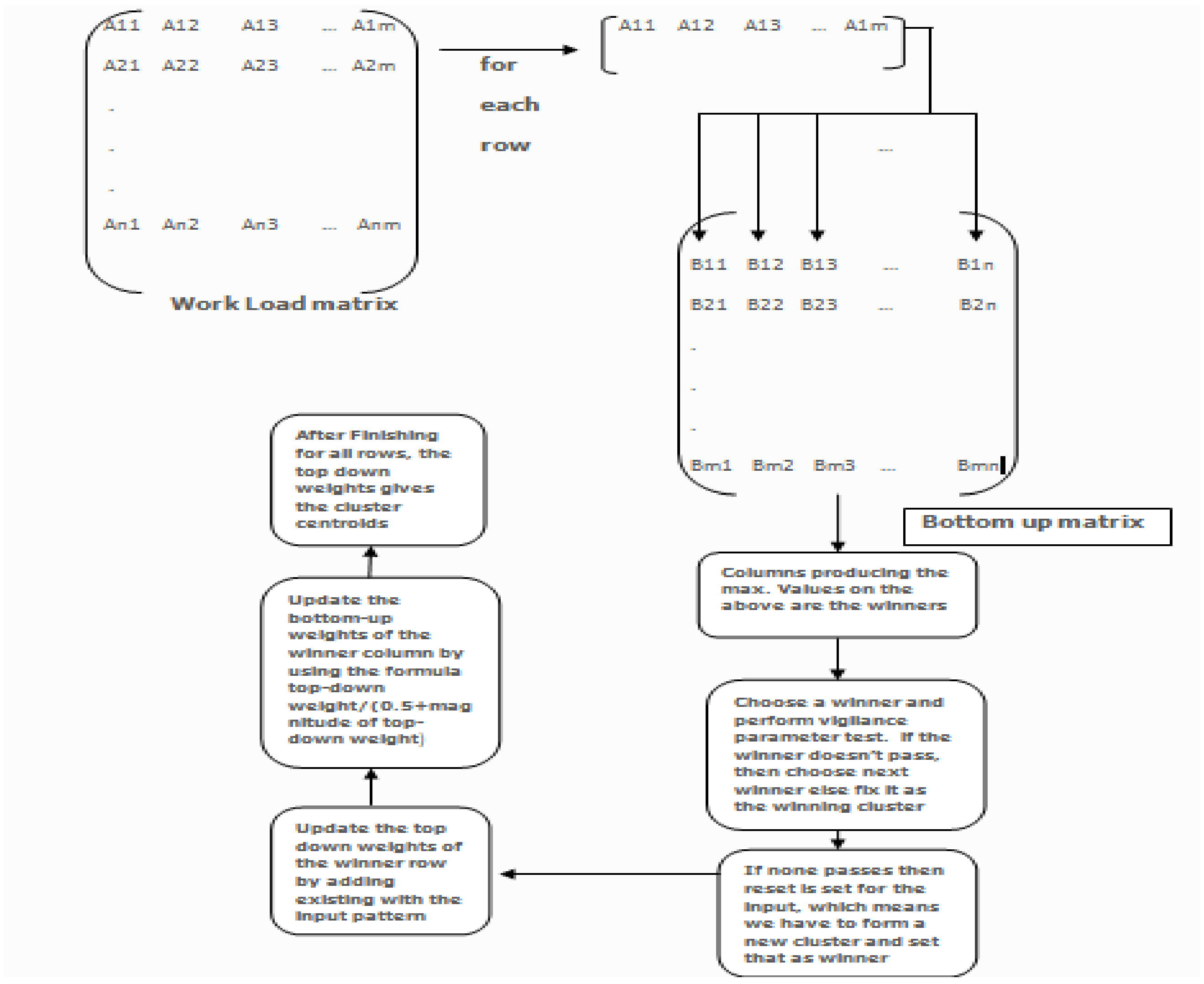

A11 A12 A13 ... A1m
A21 A22 A23 ... A2m
An1 An2 An3 ... Anm
Work Load matrix
for each row
A11 A12 A13 ... A1m
B11 B12 B13 ... B1n
B21 B22 B23 ... B2n
Bm1 Bm2 Bm3 ... Bmn
Bottom up matrix
Columns producing the max. Values on the above are the winners
Choose a winner and perform vigilance parameter test. If the winner doesn't pass, then choose next winner else fix it as the winning cluster
If none passes then reset is set for the input, which means we have to form a new cluster and set that as winner
Update the top down weights of the winner row by adding existing with the input pattern
Update the bottom-up weights of the winner column by using the formula top-down weight/(0.5+magnitude of top-down weight)
After Finishing for all rows, the top down weights gives the cluster centroids

Fig 3: Schematic representation of steps of our modified ART1 for clustering

4.4 Pre-fetching

Pre-fetching takes advantage of idle time when a user is viewing a page to speed up the links which he or she is likely to follow next by "pre-fetching" those pages. This accelerates users' online experience by making many pages load faster. Web Pre-fetching technique improves the performance of the Web Caching techniques due to prediction of the user pages in advance before the user requests. It is the process of accessing the Web objects before the user's request arrives. Whenever a user requests arrives, before that accessing the Web page a prediction is made for accessing that Web page.

4.5Integrating Web Caching and Pre-Fetching

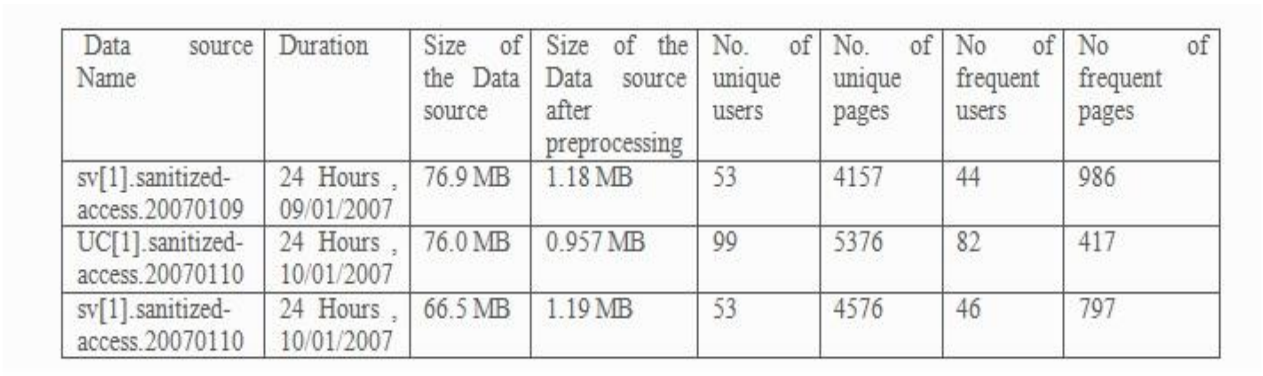
Web caching and Web pre-fetching are two important techniques which will reduce the noticeable response time perceived by users. It is the area of research in Web mining. Caching allows content to be accumulated closer to end users so that when they request it, it can be retrieved from the "closest" source without going back to the origin server. Caching both reduces the load on the origin server and accelerates users' online experience. The two main types of caches are browser caches (which reside on the user's computer) and proxy caches (which are on the network and can serve one or many users). Properly tuning a Website's caching rules is tricky, but also provides substantial benefits which make it worth a try. Web caching is limited due to its size. By integrating Web caching and Web pre-fetching, these two techniques can complement each other since the Web caching technique exploits the temporal situation, whereas Web pre-fetching technique utilizes the spatial locality of Web objects.

In this paper , we use cache replacement in Web cache which is used to avoid the over flow of cache memory ,which not only considers the caching effect in the Web based environment, but also evaluates the pre-fetching rules provided by various pre-fetching schemes. To manage the cache over flow problem, here we provide Web page replacement algorithm in cache memory, such as FIFO (First In First Out), LRU (Least Recently Used), LFU (Least Frequently Used). These are used to measure the pre-fetching process through HIT rate and MISS .If the user requested page is already present in cache then it said to be hit otherwise it comes under miss.

5. RESULTS AND DISCUSSIONS

The data sets for testing have been obtained from IRCACHE. IRCache is a NLANR (National Laboratory of Applied Network Research) project that encourages Web caching and provides data for researchers. The files that have been used for the testing of the scheme can be obtained under: ftp.ircache.net/Traces/bo2[1].sanitized-access.2007-01-09/. These represent the traces for proxy server installation at Research Triangle Park, North Carolina for the dates 01/09/2007 to 01/10/2007. Here, we present the result of our research using various data sets. The graphs below depict four different data sets for which we try out our algorithm. The four data sets are:sv[1].sanitized-access,uc[1].sanitized-ccess.20070110 and sv[1].sanitized-access.20070110.The Name, Size, number of users, pages, frequent users and frequent pages for all the three datasets ,which is the outcome after pre-processing is shown n the table below:

Table1: Details of Preprocessed datasets



Data source Name	Duration	Size of the Data source	Size of the Data source after preprocessing	No. of unique users	No. of unique pages	No of frequent users	No of frequent pages
sv[1].sanitized-access.20070109	24 Hours , 09/01/2007	76.9 MB	1.18 MB	53	4157	44	986
UC[1].sanitized-access.20070110	24 Hours , 10/01/2007	76.0 MB	0.957 MB	99	5376	82	417
sv[1].sanitized-access.20070110	24 Hours , 10/01/2007	66.5 MB	1.19 MB	53	4576	46	797

The performance of the modified ART1 clustering algorithm is better than the existing ART1 clustering algorithm in terms of the distance we had defined earlier. This is shown using the graph below. The graph shows the average inter and intra cluster distance. For all the clustering algorithms the average inter cluster distance must be maximum and the intra cluster distance must be minimum.

Graph for showing the difference in intra cluster distance between ART1 and modified ART1 is below:

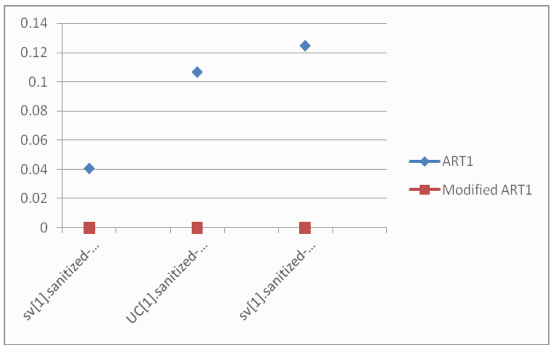


Fig 4: ART1 Vs Modified ART1 intra cluster distance

The intra cluster distance for our modified ART1 algorithm is zero since the center of each cluster is formed based on binary addition. So it pre-fetches all the frequent pages corresponding to all the frequent users belonging to the cluster.

Graph for showing the difference in inter cluster distance between ART1 and modified ART1 is below:

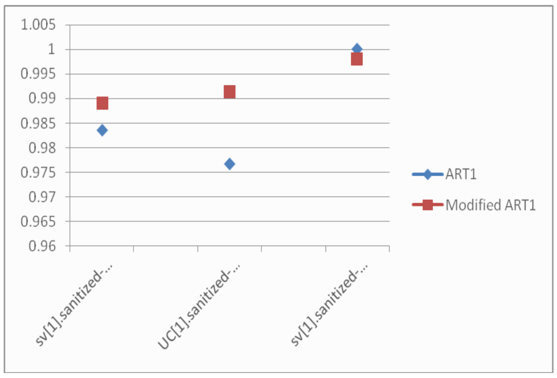


Fig 5: ART1 Vs Modified ART1 inter cluster distance

From the graph we can see that the use of modified ART1 algorithm doesn’t affect the inter cluster distance. The average inters cluster distance still lies above 0.975 which shows that 97.5 percent of the pages pre-fetched by the clusters are different.

The graph below compares the number of hits (total number of requests processed is 200) without pre-fetching, with pre-fetching using ART1 and with pre-fetching using modified ART1 for different cache sizes.

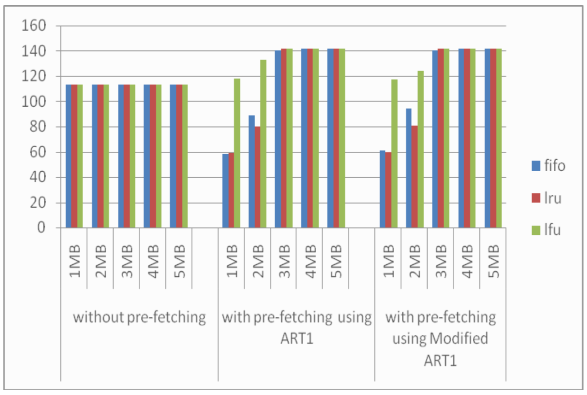


Fig 6: ART1 Vs Modified ART1 using Data Set1(sv[1].sanitized-access.20070109)

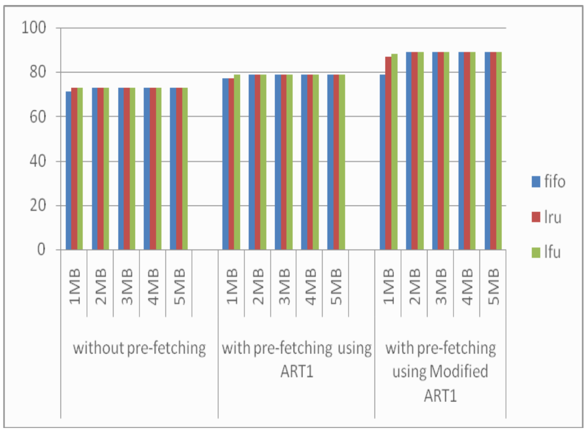


Fig 7: ART1 Vs Modified ART1 using Data Set2(UC[1].sanitized-access.20070110)

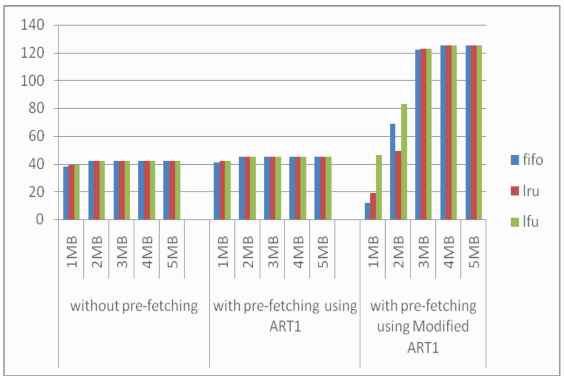


Fig 8: ART1 Vs Modified ART1 using Data Set3(sv[1].sanitized-access.20070110)

From the above graphs we can observe that the average hit rate increases as we move from schemes using ART1 algorithm to our modified ART1 algorithm based pre-fetching technique. Here we have used only 15,000 log requests for extracting the pattern vectors and clustering. For the first data set used the number of hits remains the same when using ART1 and modified ART1. For the second data set used the number of hits increases by an average of 9 hits in the modified ART1. . For the third data set used the number of hits increases by an average of 48 hits in the modified ART1. We can also observe that the average hit rate increase when we move from without pre-fetching to with pre-fetching but this rise increases gradually as the cache size increases. However, if more log requests are given as input the pre-fetching accuracy could increase further.

6. CONCLUSION
There has been considerable research done in the area of clustering users based on useful information obtained from details extracted from the log files of the host (proxy server). In this paper, we have presented a modified ART1 algorithm which has helped in considerably increasing the hit rate by making sure that almost all the frequent pages accessed by the user are pre-fetched.

7. ACKNOWLEDGMENT
Special acknowledgements to Dr.V.Murali Bhaskaran, Dr.A.M.Natarajan and Dr.Ilango Krishnamurthy for their insightful comments and suggestions.

8. REFERENCES
[1] Santosh K. Rangarajan.,”Web User Clustering and Its Application to Pre-fetching Using ART Neural Networks.”Louisiana Tech University.

[2] R.SudhakaraPandian,”Modified ART1 Neural networks for Cell Formation using Production Data.”Key Bridge Marriott, Washington DC, USA August 23-26, 2008.

[3] R C Chakraborty. “Adaptive Resonance Theory(ART).”www.myreaders.info/html/softcomputing.html.

[4] Barbara M., “ART1 and Pattern Clustering.” In Proceedings of the 1988 Connectionist Models Summer 1998, Published by M.Kaufmann, San Mateo.

[5] Fu Y., Sandhu K., and Shih M., “Clustering of Web Users Based on Access Patterns.” International Workshop on Web Usage Analysis and User Profiling (WEBKDD’99), San Diego, CA, 1999.

[6] Daniel Zeng.,”Efficient Web Content Delivery Using Proxy Caching Techniques.”IEEE transactions on systems, man, and cybernetics—part c: applications and reviews, vol. 34, no. 3, august 2004.

[7] AkshayShenoy.,”Improving the Performance of a Proxy Server using Web log mining.”San Jose State University,4-1-2011.

[8] Abdullah Balamash and Marwan Krunz.,”an overview of Web caching replacement algorithms. “University of Arizona.

[9] Anupam Bhattacharjee.,”A New Web Cache Replacement Algorithm1”Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh.

[10] Stefan Podlipnig., “A Survey of Web Cache Replacement Strategies “University Klagenfurt.

[11] Lei Shi.,” Optimal Model of Web Caching and Pre-fetching” ISCSCT ’09.

[12] Loon T. S., and Bharghavan V., “Alleviating the Latency and Bandwidth problems in WWW Browsing.” In Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97), December 1997.

[13] Ibrahim T. I., and Xu C. Z., “Neural Nets based predictive Pre-fetching to tolerate WWW Latency”. In Proceedings of the 20th International Conference on Distributed Computing Systems, IEEE, Taipei, Taiwan, Republic of China, April 2000.

[14] Fan L., Cao P., and Jacobson Q., “Web Prefetching between Low-Bandwidth Clients and Proxies: Potential and Performance.” In Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems(SIGMETRICS’99), Atlanta, GA, May 1999.

[15] Markatos E. P., and Chronaki C. E., “A Top-10 Approach to Prefetching on the Web.” In Proceedings of the Eighth Annual Conference of the Internet Society(INET'98), Geneva, Switzerland, July 1998.

[16] Padmanabhan V. N., and Mogul J. C., “Using Predictive Prefetching to Improve World Wide Web Latency.” ACM Computer Communication Review, Vol. 26, No.3,page 2336, July 1996.

[17] Tian W., Choi B., and Phoha V. V, “An Adaptive Web Cache Access Predictor Using Neural Network.” In Proceedings of the15th International Conference onIndustrial and Engineering.Applications of Artificial Intelligence and ExpertSystems, pages 450-459,IEA/AIE, Cairns, Australia, June 2002.

[18] T. M. Kroeger, D. D. E. Long, and J. C. Mogul, “Exploring the Bounds of Web Latency Reduction from Caching and Prefetching”, In Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems, Monterey, CA, December 1997.

[19] Z. Zhang, “An Integrated Prefetching and Caching Algorithm for Web Proxies using a Correlation-based Prediction Model”, M. S.Thesis, Department of Computing Science,Simon Fraser University, December 2000.

[20] Q. Yang, and Z. Zhang, “Model based Predictive Prefetching”, In Proceedings of the 12th International Workshop on Database and Expert Systems Applications, Pages 291-295, September 03-07, 2001.

[21] B. Lan, S. Bressan, B. C. Ooi, and K. L. Tan, “Rule-Assisted Prefetching in Web-Server Caching”, In Proceedings of the 9th International Conference on Information and Knowledge Management, Pages 504-511, Washington DC, USA, November, 2000.

[22] Q. Yang, H.H. Zhang, and I.T.Y. Li, “Mining Web Logs for Prediction Models in WWW Caching and Prefetching”, In Proceedings of Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pages 473-478, August 2001.

[23] O. Yang, H. H. Zhang, and H. Zhang, “Taylor Series Prediction: A Cache Replacement Policy Based on Second-Order Trend Analysis, In Proceedings of the 34th Hawaii International Conference on Systems Sciences, IEEE Computer Society, Piscataway, NJ,2001.

[24] M. Curcio, S. Leonardi, A. Vitaletti, “Integrated Prefetching and Caching for the World Wide Web”, Alcom-FT Technical Report Series, ALCOMFT-TR-01-41, 2001.