CHAPTER 4

OPTIMIZATION OF WEB CACHING PERFORMANCE BY CLUSTERING-BASED PRE-FETCHING TECHNIQUE USING MODIFIED ART1 (MART1)

4.1 INTRODUCTION

One of the prime research objectives of this thesis is to optimize the Web caching performance so as to improve the Webpage access. In this chapter, section 4.2 mentions the need for optimization of Web caching while section 4.3 brings out Web caching with clustering-based approach for Web pre-fetching. Section 4.4 elaborates on clustering-based pre-fetching using the algorithm MART1. Section 4.5 presents the proposed system architecture of this research work involving Web caching and Web pre-fetching; while section 4.6 elaborates the processes involved in clustering-based pre-fetching technique using MART1 algorithm. The architecture of pre-fetching system with MART1 algorithm is also presented. Section 4.7 draws the conclusion based on the results and discussions on the Web caching system using clustering-based Web pre-fetching technique. Performance evaluation test was conducted through the integrated Web caching and pre-fetching system.

4.2 MOTIVATION

There are many factors that affect the performance of a Web-based system which includes diversity of network connectivity, real world distances,
and congestion in networks or servers due to unpredicted demand. As a result, many research works have focused on the problem of improving Web response times. Some of the researchers want to improve the performance either by increasing the bandwidth or improvement through alternate communication technologies. The Internet has evolved from a system that deliver a simple and static HTML page to one that provides higher end services like e-learning, video conferencing, e-commerce, etc. (Pallis et al 2008). The probable solutions to reduce network traffic are given below:

- Web caching
- Web pre-fetching

Web caching is a technique used to reduce the user perceived latency when user is accessing the Webpages. Web pre-fetching is another scheme where Webpages are pre-fetched into the intermediate server (proxy server) cache before the user accesses it. When combined, these techniques would complement each other.

Thus, a clustering-based pre-fetching technique has been introduced which will pre-fetch Web objects for group of users having similar access behavior instead of single user interest. In this chapter, MART1 algorithm has been implemented for clustering-based pre-fetching technique to pre-fetch Webpages into the proxy cache. Empirical analysis also performed to show the performance of proposed work (MART1) with traditional ART1. By using this approach the hit rate of the cache increases, which in turn reduces the user perceived latencies. The MART1 would provide better inter-cluster and intra-cluster distance and produce highly homogeneous clusters than the traditional ART1.
4.3 WEB CATCHING AND PRE-FETCHING

WWW is an evolving system and consists of interlinked objects like audio, images, videos and so on. Here Web caching and pre-fetching is found as an important strategy to enhance performance of Web-based system. Recently WWW is most widely used by many users to distribute information across the world. Therefore, it leads to increased network traffic over the internet and Web server. As a result, Web becomes one of the primary areas to hold up network performance.

Transferring of Web objects over the internet increases the network traffic; thus, reducing bandwidth for competing requests and increasing latencies for Web users while accessing the Webpages. In order to reduce such access delay, it is desirable to store popular Web objects closer to the user so that they can access them. Hence, proxy-based pre-fetching technique has been used. Thus, Web caching and Web pre-fetching has become an important research topic to reduce the noticeable response latency incurred by the end users.

4.3.1 Clustering-based Approach

It is likely to group all the requests made from the IP addresses and assign the most popular Webpages in a cluster for each IP address. Chen et al (2003) states that the popularity of each Web objects vary significantly. It is difficult to choose the popular objects early and also to predict the popularity threshold. Therefore, Pallis et al (2008) argues that a graph-based approach to cluster the Webpages would be more effective. It is based on association rule mining and it is controlled by the threshold value. Jyoti et al (2008) have adopted an approach that predicts the next page to be accessed based on higher-order Markov model.
Rangarajan et al (2004) has presented an approach to group users based on their Web access patterns using ART1 by grouping users and then pre-fetching. They have compared the quality of clustering produced by ART1 with K-Means clustering algorithm in terms of inter-cluster and intra-cluster distances. The result shows that ART1 performs better by providing high hit rate and better average inter- and intra-cluster distance than the clusters formed by K-Means algorithm. Hence concluded that ART1 provides high quality of clusters.

Ramya and Shreedhara (2012) have presented a complete preprocessing methodology and pre-fetching system using ART1 and compared the quality of clustering of ART1 based clustering technique with that of traditional K-Means and Self Organization Map clustering algorithms. The results show that the average inter-cluster distance of ART1 is high when compared to K-Means and SOM when there are fewer clusters. As the number of clusters increase, average inter-cluster distance of ART1 is low compared to K-Means and SOM which indicates the high quality of clusters formed using ART1.

Most of the research work discussed for pre-fetching concentrates on pre-fetching individual users’ requests according to their past 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, it presents a pre-fetching scheme that uses MART1 clustering technique to pre-fetch requests for a large community of users instead of pre-fetching individual users’ requests.

Here in this thesis Web Usage Mining is used to optimize the existing Web cache policy for better performance. As the live data cannot be captured and used at once, the only offline data source available for Web usage mining is the access log recorded in server since it serves as a
substantial source of information about users’ Web access patterns. So such logs can be well exploited to analyze and discover the useful information about the users’ patterns of access and interests.

4.4 CLUSTERING-BASED PRE-FETCHING TECHNIQUE USING MART1

The problem with Web pre-fetching is in determining which page has to be pre-fetched and cached which is aggravated by the fact that, there is a wide spectrum of users and each has their own preferences. Hence a clustering-based pre-fetching technique is introduced. But the problem is that the multiple objects are pre-fetched making traditional replacement algorithms to end with poor performance and increases the bandwidth consumption. Hence in this work, Web usage mining is used to optimize the existing Web cache policy for better performance.

This research work tries to solve the problem mentioned above by grouping the users based on their past access patterns. Due to the grouping of the users, it avoids the task of going into the individual preferences which will be a tedious task. This work also presents an approach that integrates Web caching and clustering-based Web pre-fetching scheme using MART1 algorithm to improve the performance of proxy cache as mentioned in next chapter (chapter 5). ART neural networks are capable of developing stable clusters, whereas ART1 will cluster binary input vector and ART2 will cluster real-valued input vectors. When compared, ART1 will be more effective than ART2 because clustering the binary values is easier than real values (Chakraborty 2010).

The problems with existing ART1 clustering-based pre-fetching technique is deficient in generating cluster prototype and the similarity measures used. The prototype of a cluster represents an access pattern of all the objects within the cluster. The MART1 will provide better inter-cluster
and intra-cluster distance and produces highly homogeneous clusters than existing ART1. The intra-cluster distance of MART1 algorithm is zero since the center of each cluster is formed on the basis of binary addition. From the results obtained, it is observed that the average inter-cluster distance still lies above 0.991 which shows that 99.1 percent of the pages pre-fetched by the clusters are different. The performance of proposed technique is improved in terms of hit rate than traditional ART1.

4.5 SYSTEM ARCHITECTURE

The architecture of the proposed system for clustering-based Web pre-fetching is depicted in Figure 4.1.

![Figure 4.1 Architecture of the Proposed System](image-url)
The components involved in the proposed system for clustering-based Web pre-fetching are:

1. A Web log file contains the entire client’s requested information that can be used to extract access pattern of different users on different Websites.

2. Feature extractor extracts the access patterns from the log file and it is binary encoded.

3. The access patterns extracted above is given as an input to the clustering component.

4. The clustering component groups the user based on their access pattern.

5. Pre-fetcher identifies the user cluster and pre-fetches for all the URL’s from specific cluster.

6. Proxy-server responds with the pre-fetched URL objects.

4.6 METHODOLOGY

The process involved in clustering-based pre-fetching technique involves the following tasks.

- Data Preprocessing
- Access Pattern Generation
- Clustering
- Pre-fetching
Data preprocessing, Feature extraction and Generation of binary access pattern vector have been already discussed in chapter 3. Binary access pattern generated from the Web log file explained in chapter 3 is fed as input to the clustering module. This section describes the basic concepts involved in clustering-based pre-fetching technique using MART1.

A matrix is formed when pre-processing is done. Each row represents the unique user and each column represents the count of the individual page referred by the user. Next, the total number of requests to each page by all the users is found. Similarly it also finds the total number of requests made by each user for all the pages. Here, the threshold value is set as 2 so as to check for pages and users having access count greater than the threshold limit in the matrix formed and then term them as frequent pages and frequent users respectively. Thus the initial matrix is now reduced to a matrix containing frequent users and pages.

From this matrix, a binary matrix, similar to access pattern in Figure 3.7 (chapter 3), called workload matrix is formed by considering the requests made by the frequent user to each frequent page as shown in Figure 4.2. Here, the threshold value used is 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 the threshold is assigned as 0(means that the page need not be pre-fetched when an actual request arrives from that particular user).

Sometimes there has to be no pages pre-fetched for some users, means after applying the threshold limit, the value for all the columns may become 0. So it must be added to the default cluster which specifies that no pages are to be pre-fetched for those users and leave them from consideration during performance measures.
Figure 4.2 depicts the workload matrix (binary access pattern) where each row represents each frequent users’ access pattern and each column represents frequently accessed pages. In this access pattern, the entry in $a_{ij}$ can be either 0 or 1. 1 represents the user ‘i’ accessing frequent page ‘j’.

$$
\begin{array}{cccc}
\text{P} & a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\
\text{U} & a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\
\text{} & \ldots & \ldots & \ldots & \ldots & \ldots \\
\text{} & \ldots & \ldots & \ldots & \ldots & \ldots \\
\text{} & a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mn} \\
\end{array}
$$

**Figure 4.2 Work Load Matrix**

4.6.1 Clustering

The basic idea behind this MART1 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 traditional ART1.

4.6.2 Architecture of MART1

The architecture of neural network using MART1 involved in clustering-based pre-fetching is given in Figure 4.3. It consists of three main components: F1 layer called as Input layer, F2 layer called Output layer and Vigilance test module. Two control parameters G1 and G2 are used to activate and deactivate the nodes present in F1 and F2 layers.
Each node present in the input layer is connected with nodes present in the output layer via bottom up weights called $b_{um}$. Similarly each node present in the output layer is connected with nodes present in the input layer via top down weights called $t_{dm}$. Vigilance layer checks dissimilarity between current input and the winning node present in the F2 layer using vigilance parameter $\rho$.

Each input vector activates a winner node in the layer F2 that has the highest value among the product of input vector and the bottom-up weight vector. The F2 layer then reads out the top-down expectation of the winning node to F1, where the expectation is normalized over the input pattern vector and compared with the vigilance parameter $\rho$.

Figure 4.3  Proposed MART1 System Architecture
If the winner node in F2 layer and current input vector in F1 layer match within the tolerance allowed by the \( \rho \) then the MART1 algorithm sets the control gain \( G_2 \) to 0 and updates the top-down weights corresponding to the winner. If mismatch occurs, the gain controls \( G_1 \) and \( G_2 \) are set to 1 and disables the current winning node in F2 and process the input vector on another uncommitted node. Once the network is stabilized, the top-down weights corresponding to each node in F2 layer represent the prototype vector for that node.

There are two major differences observed between MART1 and ART1:

- Using binary addition of the input pattern with winning column of the top down matrix instead of the usual multiplication in ART1.

- Using binary addition introduces a new problem that all the bits in the centroid of the cluster become ‘1’. To overcome this, the centroid (top down weight of the winning column) is chosen by performing the test 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 MART1 algorithm which is described below.

i. ‘n’ and ‘m’ are number of frequent users and pages respectively.

ii. Two matrices are used to hold the top down with dimension \( n \times m \) and the bottom up weights that of \( m \times n \).

iii. The bottom up matrix is initialized with the value 1/ \((m+1)\) and top down matrix is initialized 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 more than one winner, it always chooses the second as the winner.

It then performs the vigilance parameter test on the winner.

- 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 less than the vigilance parameter value then the test is passed.

- 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.

- For updating the top down weight, it performs binary addition of the values in winner’s row and the input pattern values.

This MART1 outperform ART1 algorithm because it provides higher hit rate by ensuring that all the frequent pages corresponding to all the frequent users are pre-fetched. The pseudo code of MART1 algorithm is given in Figure 4.4 while the detailed schematic representation of steps of MART1 for Clustering process is given in Figure 4.5.

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-cluster and intra-cluster distance. The distance between two binary vectors A and B is calculated using the formula given in Equation (4.1) below.
DAB = No. of zeroes in A when corresponding bit in B is 1/|B| (4.1)

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 \( \rho \), \( 0 \leq \rho \leq 1 \); (Note: Here we use \( \rho = 0.5 \))

2. Apply the new input pattern \( A_i \):
3. Compute the activation values \( y_i \) for the input
   \[ y_i = \sum_{j=1}^{m} bu(j)[i] \times A[i][j] \]
4. Select the winner node \( k \) (0 \( \leq k \leq n \)) which maximize the value \( y_i \)
5. Retrieve the top-down weight of winning node \( k \) and perform vigilance test
   - 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 \( \rho \) go to step 6; else go to step 7;
6. Winner \( k \) is disabled from further activity and search for another cluster. Precede step 3 to 5 until a new cluster is found.
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

Figure 4.4 MART1 Algorithm
where DAB is the distance between A and B which is a conditional probability that defines objects similarity. If A and B are highly similar then it should have minimum DAB value.

4.6.3 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 accelerating users' online experience by making many pages load faster.

Pre-fetching improves the performance of the Web caching techniques as the user pages are predicted in advance before the user’s next request. It is the process of accessing the Web objects before the user's request arrives. Whenever a user's request arrives, before accessing a page, prediction is made for accessing that Webpage. Figure 4.6 gives the pseudo code for MART1 Pre-fetching Algorithm.

The steps involved in pre-fetching are:

- After the training of the MART1 neural network, if a particular user is accessing a Webpage, the previous access pattern of the user is extracted from access matrix depicting the users past access.

- If the access pattern is found, it is normalized and binary encoded. It is then fed into MART1 network identifying the cluster to which the user belongs. After identification, it outputs the prototype of the cluster.

- Based on this prototype, the pages to be pre-fetched and cached are decided. Once cached, the user can access these pages at much higher speed.
Figure 4.5 Schematic Representation of Steps of MART1 for Clustering
In this proposed work, Web cache is used to store objects that include both actual requests and pre-fetching requests in the Web-based environment. To manage the cache overflow cache replacement policies are used such as FIFO, LRU and LFU. These policies are used to measure the performance of pre-fetching system through HIT and MISS.

//Algorithm: MART1 based Pre-fetching algorithm

//Cluster the user pattern using the MART1 Clustering algorithm. Each cluster is denoted by C_n, where n is the number of clusters formed. The clusters C_1, C_2, C_k... C_n are represented by prototype vector T_k

//Input: uid of the user that requests a page.

//Output: the array pre-fetched PURL [], which contains a list of pages that are to be Pre-fetched for the host ‘uid’.

Initialize count = 0
For n clusters formed using MART1 Clustering algorithm
    If (uid is a member of cluster C_k)
        For j = 1 to m do //where m is the size of the winning cluster prototype
            If (T_kj == 1) //where t_kj is the j^{th} element of T_k
                PURL [count] = page_i
                Count = count +1
            End
        End
    End
End
Return PURL []

Figure 4.6 MART1 Pre-fetching Algorithm
4.7 RESULTS AND DISCUSSION

In proxy-based Web caching system (proxy server), each request from the client is assumed to be forwarded through proxy server. When the proxy receives a user request, it intercepts the request and checks its local cache to see if it has a copy of the requested object. If requested object is present then it is called cache hit and the object is returned to the user, else the proxy results in cache miss. As a result of cache miss, the proxy server contact origin server for fetching the object, stores the copy in its local cache for future use and sends back to the requesting client. If the cache is full and new object needs to be stored, then a page replacement policy is invoked to decide which object is to be removed.

The proposed model also ignores the issues of cache consistency (make sure that the cached copy is up-to-date). This is because revalidation takes place automatically by the proxy server. It sends IMS query to server and validates it. If it returns status code 200 then it is out of date and fresh copy is attached in response header, else it sends 304 means that cached copy is good and served from the cache.

Finally, the proposed system is tested by considering static files like HTML and XML files only.

4.7.1 Experimental Design

This section describes the design and the performance study of cache replacement policies with clustering-based Web pre-fetching. The discussion begins with the factors and the levels used for the simulation and the following section discuss the performance metrics used to evaluate the performance of each of the replacement policy used in the proposed system. The design factors considered are briefed in the next subsection.
4.7.1.1 Design factors

There are two main factors used in the trace-driven simulation of proposed work: i) cache size and ii) cache replacement policy. The following two subsections describe each of these factors in detail.

Cache size

The first factor in this model is cache size. For the proxy logs here used five levels from 1 MB to 5 MB. The upper bounds represent the total unique bytes in the trace (Bahn et al 1999). An infinite cache is the one that is so large and once brought into the cache then it can never be evicted (Busari and Williamson 2001). In order to evaluate the performance, the sample cache size has been chosen smaller than testing datasets.

The minimum cache size of proxy server is 5 MB. But based on size of the actual request present in the testing dataset, the cache size considered here are 1MB, 2MB, 3MB, 4MB, 5MB respectively.

Cache replacement policy

The proposed model is tested based on three different cache replacement policies namely:

i) Least Frequently Used

ii) Least Recently Used

iii) First In First Out.
4.7.1.2 Datasets details

As mentioned in section 3.8 (chapter 3), the datasets are obtained from http://www.ircache.net/. IRCACHE is a National Laboratory of Applied Network Research (NLANR) project that encourages Web caching and provides data for researchers. These represent the traces for proxy server installation at Research Triangle Park, North Carolina.

The graphs depicted in this section use four different datasets for which the proposed algorithm has been verified. The four data sets are: sv [1].sanitized-access.20070109 (Dataset1), bo2[1].sanitized- access.20070109 (Dataset 2), ny[1].sanitized-access.20070109 (Dataset 3) and uc [1].sanitized-access.20070109 (Dataset 4). The Name, Size, number of users, pages, frequent users and frequent pages for all the three datasets, which is the outcome after preprocessing as shown in Table 3.1 (chapter 3).

The details of the datasets are described in Table 4.1 through Table 4.4. The table mentions the data source name and details of datasets. It is understood from these four tables that total number of items in each dataset is beyond 10000 while the number of items used for clustering is more than 95%. The number of items taken for testing is uniform as 200 from each of the datasets with varying MBs of bytes used for testing.

Table 4.1 Testing Dataset1 Details

<table>
<thead>
<tr>
<th>Data Source Name</th>
<th>sv[1].sanitized-access.20070109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of items</td>
<td>12891</td>
</tr>
<tr>
<td>Total Number of Items used for Clustering</td>
<td>12691</td>
</tr>
<tr>
<td>Total Number of Items used for Testing</td>
<td>200</td>
</tr>
<tr>
<td>Size of the Testing Dataset (in MB)</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 4.2 Testing Dataset2 Details

<table>
<thead>
<tr>
<th>Data Source Name</th>
<th>bo2[1].sanitized-access.20070109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of items</td>
<td>12788</td>
</tr>
<tr>
<td>Total Number of Items used for Clustering</td>
<td>12588</td>
</tr>
<tr>
<td>Total Number of Items used for Testing</td>
<td>200</td>
</tr>
<tr>
<td>Size of the Testing Dataset (in MB)</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.3 Testing Dataset3 Details

<table>
<thead>
<tr>
<th>Data Source Name</th>
<th>ny[1].sanitized-access.20070109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of items:</td>
<td>14907</td>
</tr>
<tr>
<td>Total Number of Items used for Clustering</td>
<td>14707</td>
</tr>
<tr>
<td>Total Number of Items used for Testing</td>
<td>200</td>
</tr>
<tr>
<td>Size of the Testing Dataset (in MB)</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.4 Testing Dataset4 Details

<table>
<thead>
<tr>
<th>Data Source Name</th>
<th>uc[1].sanitized-access.20070109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of items</td>
<td>10884</td>
</tr>
<tr>
<td>Total Number of Items used for Clustering</td>
<td>10684</td>
</tr>
<tr>
<td>Total Number of Items used for Testing</td>
<td>200</td>
</tr>
<tr>
<td>Size of the Testing Dataset (in MB)</td>
<td>1</td>
</tr>
</tbody>
</table>

4.7.1.3 Experimental setup

All the methods presented in this work have been developed in JAVA programming language using Java Development Kit (JDK) 1.6 with
the hardware specification mentioned in subsection 3.8.2 (chapter 3). Software specification includes the operating system Windows XP or above, with frontend tool JDK and backend tool Microsoft Access database management system and text files as well.

4.7.1.4 Performance evaluation

The proposed scheme of clustering-based pre-fetching technique has been tested by combining both Web caching and Web pre-fetching techniques and evaluated the caching performance. The performance metrics used here are as follows.

- Hit Rate
- Precision
- Recall
- Average Inter and Intra-clusters Distance

Hit rate is used to measure the Web cache performance and precision and recall are used to measure the accuracy and coverage of pre-fetching system whereas last one is used to measure cluster quality produced by ART1 and MART1.

i. **Least Recently Used (LRU):** Only the most recently accessed Webpages are retained in the cache. When the cache is full and a new page is arrived then LRU will evict the page which has been least recently accessed.

ii. **First in First out (FIFO):** The pages are retained based on age factor that is how long it is in the cache. When the cache
is full and a new page is arrived then FIFO will evict an oldest page which had entered first into the cache and create space for the new page.

iii. **Least Frequently Used (LFU):** In this scheme, only the most frequently accessed pages are stored in the cache. When the cache is full and a new page is arrived then LFU evict a page which has least access count in the cache and create space for the new page.

iv. **Pre-fetching based LRU and LFU:** Here, the client that made the request is identified. In the next step, it finds out the cluster which contains the current client and pre-fetches all the pages within that cluster. Once the pages are brought into the cache then the cache replacement algorithms of LRU or FIFO or LFU are used to manage these pages. If the client does not have any pre-computed cluster then the request is processed as usual using the routine LRU or LFU or FIFO scheme for caching.

**4.7.2 Experimental Results**

The performance of MART1 clustering algorithm is better than the existing traditional ART1 clustering algorithm in terms of various metrics defined earlier and the results are shown in Figure 4.7 through Figure 4.11. Graph in Figure 4.7 shows the difference in intra-cluster distance between ART1 and MART1 while 4.8 shows the comparison between average inter-cluster and intra-cluster distance of ART1 and MART1 under various datasets.
Best clustering algorithm is the one where the average inter-cluster distance must be maximum and the intra-cluster distance must be minimum. The intra-cluster distance for MART1 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. It is observed that MART1 outperforms ART1 in terms of intra-cluster distance.

Graph depicted in Figure 4.8 shows the difference in inter-cluster distance between ART1 and MART1 under various datasets. It is inferred from the graph that MART1 produces higher inter-cluster distance. The average inter-cluster distance still lies above 0.99.1 which shows that 99.1% of the pages pre-fetched by the clusters are different. Thus MART1 produces highly homogeneous clusters than ART1.

![Figure 4.7 ART1 Vs MART1 Intra-cluster Distance](image-url)
Figure 4.8 ART1 Vs MART1 Inter-cluster Distance

Figure 4.9 ART1 Vs MART1 based on Number of Clusters
Figure 4.9 gives the graph representing number of clusters formed by ART1 and MART1. It is seen from the graph that MART1 produces less number of clusters compared to ART1 with high inter-cluster distance value.

Further, to evaluate the accuracy and coverage of prediction system, metrics such as precision and recall are used. Under both the metrics, value ranges from 0 to 1. Higher value indicates more efficient and lower value indicates less efficient ones.

For instance, graph in Figure 4.10 depicts the coverage of ART1 and MART1 on uc [1].sanitized-access.20070109 dataset. Here 200 user requests are considered for evaluating the proposed system under varying cache size. This performance of MART1 based pre-fetching technique with ART1 based pre-fetching technique using precision as one of the measure infers that the precision value of MART1 algorithm is higher than ART1. Hence it establishes that MART1 outperforms the ART1.

Similarly by considering bo2 [1].sanitized-access.20070109 dataset, its precision value ranges from 0.973146 to 0.973146 in MART1 whereas under ART1 it ranges from 0.910051546 to 0.959278. While considering recall value it ranges from 0.64 to 0.64 in ART1 whereas MART1 it ranges from 0.65 to 0.67. This also points out that MART1 produces highly accurate system where user requests are satisfied in the cache.
Figure 4.10 Precision ART1 Vs MART1 on Dataset uc [1].sanitized-access.20070109

Figure 4.11 Recall ART1 Vs MART1 on Dataset uc [1].sanitized-access.20070109
The graph depicted in Figure 4.11 compares the coverage of MART1 based pre-fetching technique with ART1 using recall as another measure. It is also inferred that the recall value of MART1 algorithm is higher than ART1. Hence again MART1 outperforms ART1.

Table 4.5 through Table 4.8 gives the picture of number of hits under the sample four different datasets. These tables have recorded the hit count information under pre-fetching using ART1 algorithm and MART1 algorithm in addition to without pre-fetching for various cache replacement algorithms namely FIFO, LFU and LRU.

Figure 4.12 through Figure 4.15 portrays the cache performance in terms of hit rate on four different datasets for cache system under different environments such as Caching without pre-fetching scheme, Caching with ART1 based pre-fetching and Caching with MART1 based pre-fetching technique. All the schemes are tested with different cache sizes.
Table 4.5 Hit Count for Dataset sv [1].sanitized-access.20070109(out of 200 users request)

<table>
<thead>
<tr>
<th>Policies</th>
<th>Cache Size</th>
<th>Without Pre-Fetching</th>
<th>With pre-fetching using ART1</th>
<th>With pre-fetching using MART1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1MB</td>
<td>2MB</td>
<td>3MB</td>
<td>4MB</td>
</tr>
<tr>
<td>FIFO</td>
<td>113</td>
<td>113</td>
<td>113</td>
<td>113</td>
</tr>
<tr>
<td>LRU</td>
<td>113</td>
<td>113</td>
<td>113</td>
<td>113</td>
</tr>
<tr>
<td>LFU</td>
<td>113</td>
<td>113</td>
<td>113</td>
<td>113</td>
</tr>
</tbody>
</table>
Table 4.6 Hit Count for Dataset bo2[1].sanitized- access.20070109(out of 200 users request)

<table>
<thead>
<tr>
<th>Policies</th>
<th>Cache Size</th>
<th>Without Pre-Fetching</th>
<th>With pre-fetching using ART1</th>
<th>With pre-fetching using MART1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1MB</td>
<td>2MB</td>
<td>3MB</td>
<td>4MB</td>
</tr>
<tr>
<td>FIFO</td>
<td>71</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>LRU</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>LFU</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>Policies</td>
<td>1MB</td>
<td>2MB</td>
<td>3MB</td>
<td>4MB</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>FIFO</td>
<td>38</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>LRU</td>
<td>39</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>LFU</td>
<td>39</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
</tbody>
</table>

Table 4.7 Hit Count in ny[1].sanitized-access.20070109(out of 200 users request)
Table 4.8 Hit Count for Dataset uc [1].sanitized-access.20070109(out of 200 users request)

<table>
<thead>
<tr>
<th>Policies</th>
<th>Without Pre-Fetching</th>
<th>With Pre-Fetching Using ART1</th>
<th>With Pre-Fetching Using MART1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache Size</td>
<td>1MB</td>
<td>2MB</td>
<td>3MB</td>
</tr>
<tr>
<td>FIFO</td>
<td>121</td>
<td>121</td>
<td>121</td>
</tr>
<tr>
<td>LRU</td>
<td>121</td>
<td>121</td>
<td>121</td>
</tr>
<tr>
<td>LFU</td>
<td>121</td>
<td>121</td>
<td>121</td>
</tr>
</tbody>
</table>
Figure 4.12 ART1 Vs MART1 using Data Set (bo2[1].sanitized- access.20070109)
Figure 4.13 ART1 Vs MART1 using Data Set (sv [1].sanitized-access.20070109)
Figure 4.12 shows the performance of pre-fetching techniques on dataset-1. From the graph it is observed that hit rate of simple caching system is only about 36.5% while integrating Web caching system with pre-fetching technique hit rate is improved significantly. It is seen from the figure that ART1 based pre-fetching system gives hit rate further up to 39.5% whereas MART1 based pre-fetching produces the high hit rate 44.5%. Therefore, it is declared that MART1 scheme outperforms the ART1 scheme.

Similarly, the graph in Figure 4.13 gives the performance of pre-fetching techniques on dataset-2. Here, the hit rate of simple caching is only 56.5% while with pre-fetching technique hit rate is improved to 71% under both ART1 pre-fetching system as well as MART1. Yet another observation is evident that the performance increases based on the cache size increase.

Figure 4.14 shows the performance of pre-fetching techniques on dataset-3. It is observed that the hit rate of simple caching is only 21% whereas the hit rate has increased to 22.5% under ART1 pre-fetching system which is far behind the MART1 scheme with 62.5%.

Figure 4.15 shows the performance of pre-fetching techniques on dataset-4. In this case, hit rate of simple caching is seen as 60.5%. The integrated cache system with MART1 based pre-fetching scheme gives an increase to 67% while that of ART1 pre-fetching system improves hit rate to 64%.
Figure 4.14 ART1 Vs MART1 using Data Set (ny[1].sanitized-access.20070109)
Figure 4.15 ART1 Vs MART1 using Data Set (UC [1].sanitized-access.20070109)
Based on the above four graphs, it is observed that the average hit rate increases when integrating Web caching system with pre-fetching system. It is also observed that MART1 based pre-fetching technique improves hit rate significantly than traditional ART1 pre-fetching technique. For dataset-1, the number of hits increased when going from ART1 to MART1. In dataset-2 the number of hits remains the same in both ART1 and MART1. This is because most of the users just come and find new information via browsing. For dataset3 and dataset-4 also, the number of hits are in increasing trend. Therefore, it is concluded that the average hit rate increases with pre-fetching. However, this rise of hit rate is gradually increasing as the cache size increases.

4.8 SUMMARY

There has been considerable research work done in the area of clustering users based on useful information obtained from details extracted from the log files of the host (proxy server). This research work also brings out the similar contribution. Throughout this chapter, the performance of clustering- based pre-fetching technique using ART1 and MART1 is discussed and presented. The performances of MART1 algorithm has been empirically shown using real time datasets and compared with ART1. From these results, it is observed that MART1 outperforms ART1 in all cases. Efficient clusters lead to an effective pre-fetching policy and increases accuracy. Hence it is concluded that:

- The proposed pre-fetching technique, MART1, improves the network performance since it achieves higher hit rate than the other approaches.
- MART1 is a realistic scheme, which can be adapted easily to a Web cache environment.
• It bridges the performance gap between hit rate and the percentage of the requests satisfied by the proxy cache.

• It is also suitable for larger datasets.

Excellent tallying comparisons have been noticed in the case of MART1. The performances on ART1 and simple caching without pre-fetching have also been presented. The next chapter elaborates on the novel approaches to integrate the Web caching and Web pre-fetching with clustering-based MART1 approach.