A CLUSTERING BASED WEB PREFETCHING IN HIGH TRAFFIC ENVIRONMENT

BY

ATTA, FATIMA OHEZA

DEPARTMENT OF MATHEMATICS

FACULTY OF SCIENCE

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

AUGUST, 2016

A CLUSTERING BASED WEB PREFETCHING IN HIGH TRAFFIC ENVIRONMENT

By

ATTA FATIMA OHEZA (ABU, 2012)

P13SCMT8012

A THESIS SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF MATHEMATICS,

FACULTY OF SCIENCE

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

AUGUST, 2016

DECLARATION

I declare that the work in this thesis entitled "A clustering based web prefetching in high traffic environment" has been performed by me in the Department of Mathematics under the supervision of Dr. A. F. Donfack Kana and Dr. A. A. Obiniyi. The information derived from the literature has been duly acknowledged in the text and list of references provided. No part of this project report was previously presented for another degree or diploma at any university.

ATTA FATIMA OHEZA		
Name of student	Signature	Date

CERTIFICATION

This thesis entitled A CLUSTERING BASED WEB PREFETCHING IN HIGH TRAFFIC ENVIRONMENT by ATTA FATIMA OHEZA meets the regulation governing the award of the degree of M.Sc Computer Science of Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

(Signature) _____

_____ Date _____

Chairman, Supervisory Committee

(Signature) _____

_____ Date _____

Member, Supervisory Committee

_____

(Signature) _____

_____ Date _____

Member, Supervisory Committee

(Signature) _____

_____ Date _____

Head of Department

(Signature) _____

_____ Date _____

Dean, School of Postgraduate Studies

DEDICATION

I dedicate this work to Almighty Allah, for seeing me through this program.

ACKNOWLEDGEMENT

Thanks are probably intricate than one may think, for the simple reason that it is impossible to write in a few lines all the people who have contributed to achieving this important goal of my life (but they know it).

My deepest gratitude goes to ALMIGHTY ALLAH for His infinite blessings, whose mercy has been keeping me alive till this day and in sound health. Alhamdulillah, all praises belong to Allah for His blessings and protection, indeed I am deeply indebted to Him, without which my work would not have been possible.

My sincere gratitude goes to Dr. A. F. Donfack Kana (Major) and Dr. A. A. Obiniyi (Minor), for their guidance, ideas, valuable direction and support throughout this research work. They are the brain behind this research work; this work would never have been possible without these supervisors. Their insights and suggestions on the research work and for valuable supervision at various stages of my work has helped me in many ways and has developed my academic thinking, problem solving and technical writing which will be very helpful in my future work and research. I also acknowledge the effort of the entire lecturers of the Department of Mathematics, Ahmadu Bello University Zaria, for their academic and intellectual support towards the success of this thesis.

Most importantly, I thank the ones I hold dearly, my family. My sisters Kaka, Bilkis and Aisha for your support. Without your constant encouragement, unconditional love, psychologically and economically support, I would not have been able to undertake this endeavor.

I am indeed grateful to a great deal of people, Khalifa, Mubarak, Abdulmalik,Aminu, Alex, Ben, Musa, Precious, Jovi. A few of the long list of people whom has encouraged me during the program. All I can say is May Almighty Allah continue to reward and bless their efforts.

I am sure that the outcome would not have been nearly as good as it turned out without their help and supports. I would like to express my heart-felt gratitude.

ABSTRACT

The continued increase in demand for objects on the Internet causes high web traffic and consequently low user response time which is one of the major bottleneck in the network world. Increase in bandwidth is a possible solution to the problem but it involves increasing economic cost. An alternative solution is web prefetching. Web prefetching is the process of predicting and fetching web pages in advance by proxy server before a request is sent by a user. Prefetching is performed during the server idle time. Most literature based on the classical prefetch algorithm assumes that the server idle time is large enough to prefetch all user's predicted requests which is not true in a real life situation. This research aims at improving the web prefetching technique by developing a prefetching technique that can be effective in a high traffic environment when the server idle time is very low.Log files were collected and preprocessed for several client group within a domain. The preprocessed log files were used to create web navigation graph, which shows the transition from one web page to another web page.Support and confidence threshold were used to remove web pages with values less than the threshold values. Several clusters were formed in a particular client group. When the prefetch time is predicted to be too small to prefetch, the entire clusters formed from various domains will be used to create a prioritized cluster based on several user request. The model was evaluated based on hit rate, byte rate, precision, accuracy of prediction and usefulness of prediction. The result shows that the proposed WebClustering algorithm performs better than the classical prefetch technique when the server idle time is small and behaves same as the classical algorithm as the server time becomes large enough to prefetch all users predictions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACCRONYMS

WNG:Web Navigation Graph

BFS:Breadth First Search

LRU:Least Recently Used

LFU:Least Frequently Used

ISP:Internet Service Provider

HTTP: Hypertext Transfer Protocol

HTML:Hypertext Mail Language

DG:Dependency Graph

ART:Adaptive Resonance Theory

NASA:National Aeronautics and Space Administration

IPGDSF#:Intelligent Predictive Greedy Dual Size Frequency

ANN:Artificial Neural Network

PSO:Particle Swam Optimization

XML:Extendible Markup Language

SVM:Support Vector Machine

PPM:Prediction by Partial Matching

URL:Uniform Resource Locator

FIFO:First Come First Serve

GIF:Graphics Interchange Format

JPEG: Joint Photographic Experts Group

CPU:Central Processing Unit

CLR:Common Language Runtime

HR:Hit Ratio

CHAPTER ONE

INTRODUCTION

1.1 Background of Study

The web is a collection of text documents and other resources, linked by hyperlinks and Uniform Resource Locator (URLs), usually accessed by web browsers, from web servers. The web started from a simple information sharing system, and has now grown to a rich collection of dynamic and interactive services. The tremendous growth of web has resulted into high demand for high bandwidth and delay in fetching user request (Neha, 2013). Users sometimes experience unpredictable delay while retrieving web pages from the server. Increase in bandwidth is a possible solution to the problem but it involves high economic cost. Web caching reduces the latency perceived by the user, reduces bandwidth utilization and reduces the loads on the origin servers (Pallis, 2007). Latency refers to the time elapsed from the time a request is sent to the time sender receives the requested information.

Many latency tolerant techniques have been developed over the years to solve this problem without necessarily increasing the bandwidth. Most notably are caching and prefetching. Web prefetching helps to fetch and cache users request during server idle time, which will reduce the load on the origin server. To reduce the access delay experienced by users, it is advisable to predict and prefetch web object based on user access patterns and cache them. Studies on web pre-fetching are mostly based on the history of user access patterns. If the history information shows an access pattern of URL address A followed B with a high probability, then B will be prefetched once A is accessed (Cheng-Zhong, 2000).

Web prefetching is the process of obtaining web pages in advance by proxy server before a request is sent by a user. When a client makes a request for web object, rather than sending request to the web server, it may be fetched from the cache. The main factor for selecting a web pre-fetching algorithm is its ability to predict the web object to be prefetched in order to reduce latency. Web prefetching exploits the spatial locality of web pages, i.e. pages that are linked with current page will be accessed with higher probability than other pages. Web prefetching can be applied in a web environment as between clients and web server, between proxy servers and web server and between clients and proxy server (Greeshma, 2012).

Web prefetching techniques are categorized into probability based and clustering based using weight-functions. In the probability based pre-fetching, probabilities are calculated using the history of data access. This method assumes that the request sequence follows a pattern and calculates the probabilities of following this pattern. Clustering based pre-fetching methods make decisions using the information of the web pages that have been fetched previously, assumes that pages that are close to the previously fetched pages are more likely to be requested in the near future (Greeshma, 2012).

Moreover, web prefetching is a research topic that has gained increasing attention in recent years. The web pre-fetching fetches some web objects before users actually request it. Thus, the cache pre-fetching helps on reducing the user perceived latency. Many studies have shown that the combination of caching and pre-fetching doubles the performance compared to single caching (Waleed, 2012).

Web caching is a well-known strategy for improving performance of Web based system by keeping Web objects that are likely to be used in the near future in location closer to user. The Web caching mechanisms are implemented at three levels client level, proxy level and original server level. Significantly, proxy servers play the key roles between users and web sites to reduce of the response time of user requests and saving of network bandwidth. Therefore, for achieving better response time, an efficient caching approach should be built in a proxy server (Waleed, 2011).

Due to the limitation of cache space, an intelligent mechanism is required to manage the Web cache content efficiently. The classical caching policies are not efficient in the Web caching since it considers either recency, frequency, sizeand ignore a combination of two factors that have impact on the efficiency of the Web caching. Unfortunately, the cache hit ratio is not improved much with classical caching schemes. Even though with a cache of infinite size, the hit ratio is still limited regardless of the caching scheme. This is because most people browse and explore the new web pages trying to find new information. In order to improve the hit ratio of cache, Web pre-fetching technique is integrated with web caching to overcome these limitations.

Knowing the user's browsing history provides extra information like the type of the user or his/her preferences. This information about the user can help to improve prediction accuracy in pre-fetching process (Lenka, 2010).

1.2 Problem Statement

As the Internet continues to grow in size and popularity, web traffic and network bottlenecks are major issues in the network world. The continued increase in demand for objects on the Internet

causes severe traffic and low idle time to prefetch all clusters generated from users' request. Clustering based prefetching has been explored in several ways all assumed the server idle time for prefetching is large enough to accommodate the prefetching. In a real scenario, this is not always the case since in high traffic, the idle time may not be so high to accommodate prefetching of large size data. This work therefore seeks to address this lack of consideration of volume of high traffic during the prefetching.

1.3 Motivation

Internet users expect the web to be more friendly and meaningful with reduced network traffic. Every user needs the channel with high bandwidth and low traffic. In order to reduce the web server load, the access latency and to improve the network bandwidth from heavy network traffic, a web prefetching scheme taking low bandwidth during high traffic is considered.

1.4 Aim and Objectives

The aim of this research is to improve the web prefetching technique, by developing a prefetching technique that can be effective in a high traffic environment when the server idle time is very low.

The specific objectives are to:

a) predict user request based on history of user
b) determine which pages will be requested by majority of users in the nearest future.
c) prioritize the prefetching based on the frequency of the server idle time
d) evaluate the algorithm in respect to existing prefetching algorithm

1.5 Research Method

In order to meet the objectives of this work, the following steps will be taken in the proposed inter clustering scheme:

a) Review of existing literature in the field of study.
b) Log files of users request will be collected using squid proxy server. The log files will pass through stages of cleaning processes for the removal of irrelevant information, user identification will be created for the size of pages made by users during a visit to a particular site.
c) The preprocessed log file will be used to construct a weighted Web Navigation Graph (WNG). The node of the graph represent the web pages while its edges represent the movement from one web page to another. The edges are assigned weights based on the frequency of visiting a page. Support and confidence threshold will be applied on the WNG to eliminate pages with low support and confidence value.
d) The graph will be transversed using Breadth First Search (BFS) algorithm to form several clusters within a domain.
e) In high traffic environment, clusters will be formed in favour of the requested web object by setting the support and confidence values to accommodate the requested web object from several domain. An inter domain cluster will be reconstructed from the several clusters.
f) C# will be used to implement the algorithm.
g) The proposed technique will be compared with that of Thulase*et al*. (2014) based on hit ratio, byte ratio, usefulness of prediction, accuracy of prediction and precision.

1.6 Organization of Dissertation

The rest of the work is organized asfollows:Chapter 2 is the literature review, the proposed web prefetching scheme is discussed in chapter 3, chapter 4 entails the result and analysis and chapter 5 concludes, summarizes and recommend the future works.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter discusses web prefetching and caching in details. It also elaborate on the related work.

2.2 Web Caching

Web caching is one of the most successful solutions for improving the performance of Web-based system. In Web caching, the popular web objects that are likely to be visited in the near future are stored in positions closer to the user like client machine or proxy server. Thus, the web caching helps in reducing Web service bottleneck, alleviating of traffic over the Internet and improving scalability of the Web system (Waleed, 2011).

2.3 Types of Web Cache

Web caching keeps a local copy of Web pages in places close to the end user. Caches are found in browsers and in any object between the user agent and the origin server. Figure 2.1 depicts the various locations of the web cache. Typically, a cache is located in client (browser cache), proxy server (proxy cache) and origin server (cache server)

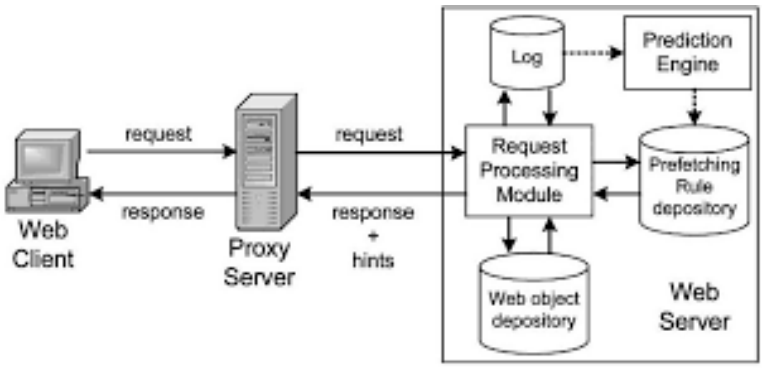

Figure 2.1: Web Prefetching and Caching Environment (Padmapriya*et al*., 2013)

2.3.1 Client Side Cache

The client side cache is located in the user browser. The user can notice the cache setting of any modern Web browser such as Internet Explorer, Safari, Mozilla Firefox, Netscape and Google chrome. This cache is useful, especially when users hit the "back" button or click a link to see a page they have just looked at. In addition, if the user uses the same navigation images throughout the browser, they will be served from browsers' caches almost immediately.

2.3.2 Proxy Server Cache

The proxy server cache is found in the proxy server which is located between client machines and origin servers. It works on the same principle of browser cache, but on a much larger scale. The proxies serve hundreds or thousands of users in the same way. When a request is received, the proxy server checks its cache. If the object is available, it sends the object to the client. If the object is not available, or has expired, the proxy server will request the object from the origin server and send it to the client. The object will be stored in the proxy's local cache for future requests. This work seeks to use a proxy server which will be elaborated in section 2.5.

2.3.3 Origin Server Cache

Even at the origin server, web pages can be stored in a server-side cache for reducing the need for redundant computations or database retrievals. Thus, the server load can be reduced if the origin server cache is employed.

2.4 Cache Replacement Policy

Cache replacement policy is the rule required to remove a web object from the cache whenever the cache memory is full. When a cache miss occurs, the cache controller will select a block to be replaced with the desired data. A replacement policy determines which block should be replaced. The selection of the block to be replaced can be determined in several ways. The conventional cache replacement policies are Least Recently Used (LRU) and Least Frequently Used (LFU). LRU cache replacement policy replaces the least recently requested web page while the LFU cache replacement policy replaces the least frequently requested web page. The following are the factors (features) of Web objects that influence Web proxy caching:

a) Recency: object's last reference time.
b) Frequency: number of requests made to an object.
c) Size: size of the requested Web object. Access latency of web object

2.5 Proxy Caching

A Proxy server processes requests from within a firewall by forwarding them to the remote servers, intercepting the responses, and sending the replies back to the clients. Since the same proxy servers are typically shared by all clients inside, this leads to the question of the effectiveness of using these proxies to cache documents. Clients within the same firewall usually belong to the same organization and likely share common interests. They would probably access

the same set of documents. Therefore on the proxy server a previously requested and cached document would likely result in future hits. Web caching at proxy server can not only save network bandwidth but also lower access latency for the clients. Caching can be done at the client browser, sever side and the proxy server. There are three types of proxy caching which are forward proxy caching, reverse proxy caching and transparent proxy caching (Reddy, 2007).

2.5.1 Forward Proxy Caching

In forward proxy caching, caches are generally positioned at the edge of a network. This is done so that large number of internal consumers may be serviced. Placement of proxy caches can lead to bandwidth savings, quicker response times, and enhanced accessibility to static Web objects. However, proxy cache placement could become a single point of failure in the network which can cause problem (Reddy, 2007).

2.5.2 Reverse Proxy Caching

In reverse proxy caching, caches are placed near the source of the content rather than the destination of the content. This is an attractive solution for servers that expect a high number of requests and want to ensure a high level of quality of service. Reverse proxy caching is also a useful mechanism when supporting web hosting farms (virtual domains mapped to a single physical site), an increasingly common service for many Internet service providers (ISPs). Reverse Proxy is a service between a client and a server where the request is sent by the client to the proxy. It acts as a forwarding service which connects the server and exchanges data between the client and the server (Reddy, 2007).

2.5.3 Transparent Caching

In transparent caching, the caches intercept HTTP requests and anticipate them to Web cache servers. Transparent proxy caching overcomes the problem of configuration of Web browsers encountered in proxy caching. Transparent caching avoids the need for configuring browsers of users (Sachan, 2014). Transparent proxy caching eliminates one of the big drawbacks of the proxy server approach; the requirement to configure Web browsers. This style of caching establishes a point at which different kinds of administrative control are possible; for example, deciding how to load balance requests across multiple caches. There are two ways to deploy transparent proxy caching: at the switch level and at the router level. Router-based transparent proxy caching uses policy-based routing to direct requests to the appropriate cache. For example, requests from certain clients can be associated with a particular cache. In switch-based transparent proxy caching, the switch acts as a dedicated load balancer. This approach is attractive because it reduces the overhead normally incurred by policy-based routing. Although it adds extra cost to the deployment, switches are generally less expensive than routers(Reddy, 2007).

2.6 Web Prefetching

Web pre-fetching is a mechanism which is used for prefetching web pages into the cache before the actual request arrives. When combining web pre-fetching and web caching they can complement each other since the web caching technique make use of temporal locality, whereas web prefetching technique exploit the spatial locality of web objects. The problem is to determine which page has to be pre-fetched and cached. This problem is aggravated by the fact that there are wide spectrums of users and each has their own preferences.

The web documents that are prefetched from the server are stored in the prefetching cache to satisfy the user's future requests. To eliminate the caching impact due to temporal locality exhibited in the user access patterns, prefetching cache is managed separately from the browser's in-built cache. It is the speculative retrieval of a resource into a cache in the anticipation that it can be served from the cache in the near future, thereby decreases the load time of the object (Ramu, 2012).

Prefetching techniques can be implemented on the client side, server side or on the proxy side. The client side prefetching helps in keeping track of the patterns of a single user across the various web servers. The server side prefetching helps in keeping track of the patterns of all the users accessing a particular web site. The proxy side prefetching helps in keeping track of the patterns of a group of users accessing many Web servers. Prefetching algorithms are classified into content based prefetching and history based prefetching. The former method analyses the web page contents and predicts the HTML links to be followed by the clients. History based prefetching makes prediction based on observed page access behavior of the user in the past. The algorithms that come under this category are classified into four types namely approach based on dependency graph, approach based on Markov model, approach based on cost function and approach based on data mining.

a) In the dependency graph (DG) approach, graph is constructed for prediction of next page. The dependency graph consists of nodes that represent Web pages, and arcs that indicate that the target node is accessed after the original node within a time window.The dependency graph based prefetching approach predicts and prefetches the nodes whose arcs connect to the current accessed node and have weights higher than a threshold.

b) Markov based prefetching approach is a kind of effective model for predicting the next Web page by matching the user's current access sequence with the user's historical Web access sequences.Markov approach prefetches some web objects into the cache depending on some factors such as popularity and lifetime of web pages.

c) The data mining based prefetching approach in turn is classified into association based prefetching approach and clustering-based prefetching approach.Association rule discovers groups of homogeneous pages that are commonly accessed together in same user session. The user session is defined as the sequence of pages made by a single user during a visit to a particular site. In association rules, support and confidence are two important measures for benchmarking strength of any association rule. Support is defined as the discovery of frequent pages, while confidence is defined as the discovery of association rules from these frequent pages. Clustering is utilized for finding similarity groups in data, called clusters, such that the data instances in the same cluster are similar to each other while data instances in different clusters are different from each other. If caching and prefetching methods are combined together, the hit ratio gets improved and the user perceived latency gets reduced.

Web prefetching scheme can be classified into two types: short term pre-fetching and long term prefetching schemes.

2.6.1 Short Term Prefetching

In short-term prefetching, cache use recent access history to predict and prefetch objects likely to be referenced in the near future. Objects that are likely to be referenced in the near future are prefetched based on the client's recent access history. Future requests are predicted based on the cache's recent access history. From this predictions, clusters of Web objects are prefetched.

In addition, several short-term prefetching policies exist namely Predictive Web prefetching, semantic web prefetching and proxy web prefetching (Ramu, 2012).

2.6.2 Long Term Prefetching

The long-term prefetching uses long-term steady-state object access rates and update frequencies to identify objects to replicate to content distribution locations. Compared to demand caching, long-term prefetching increases network bandwidth and disk space costs but may benefit a system by improving hit rates. In the long term policy, objects are prefetched and updated based on long-term global access and update patterns. Global object access pattern statistics such as object popularity and object consistency are used to identify valuable objects for prefetching (Ramu, 2012).

2.7 Related Works

Several works have been carried out in the recent years in order to improve the performance of web prefetching and caching.

Cheng-Zhong (2000) predicts future requests based on semantic preferences of past retrieved document. This technique was applied to news reading activities and prototyped a News Agent prefetching system. The system extracts document semantics by identifying keywords in their URL anchor texts and relies on neural networks over the keyword set to predict future requests. It features a self-learning capability and good adaptively to the change of user surfing interest. They cross-examined the system in daily browsing of British Broadcasting Corporation (BBC), Cable News Network (CNN), and National Broadcasting Commission (NBC) news sites for three months.

Rangarajan *et al* (2002) presented an approach to group users based on Adaptive Resonance Theory (ART1) neural networks to group users according to their Web access patterns. The quality of clustering of ART1 based clustering technique with that of the K-Means clustering algorithm in terms of inter-cluster and intra-cluster distances were compared. The prefetching scheme was achieved using clustering technique to group users and then prefetch their requests according to the prototype vector of each group. The Web log files were provided by National Aeronautics and Space Administration (NASA). ART1 neural network based clustering technique was used to prefetch requests for a community of users.

Yang (2003) presented a data-cube model to represent Web access sessions for data mining for supporting the prediction model construction. The author proposed an integrated web-caching and web-prefetching model, where the issues of prefetching aggressiveness, replacement policy and increased network traffic were addressed together in an integrated framework. The cube model organizes session data into three dimensions. With the data cube in place, an efficient data mining algorithm was applied for clustering and correlation analysis. In their work, prefetching those web documents that are close to a user requested document is used to form a cluster model. These pages are managed by an integrated caching and prefetching system.

Pallis *et al* (2007) presented a clustering based prefetching scheme called clustpref algorithm where a graph based clustering algorithm identifies clusters of Web pages based on the users' access patterns. The author dealt with short-term prefetching policy which was applied on a Web caching environment. The Web log file was collected and preprocessed. The preprocessed web log file was used to create Web navigational graph. Support and confidence value were applied on the web navigation graph and BFS was applied to from several clusters within a domain. A

user request for a web object, the proxy server is checked for the object, if it is present in the proxy server it will be retrieved and sent to the user .If the web object isn't present in the cache the object will be retrieve from the origin server and used to create several clusters for each domain.

Mehrdad *et al* (2008) proposed an approach based on the graph partitioning for modeling user navigation patterns. They proposed a novel formula for assigning weights to edges of the graph and an approach for clustering of user navigation patterns based on the graph partitioning. For the clustering of user navigation patterns, they created an undirected graph based on connectivity between each pair of web pages and assign weights to edges of the graph. Several pre-treatment tasks were done before performing web mining algorithms on the Web server logs. Data pre-treatment in a web usage mining model (Web-Log preprocessing) aims to reformat the original web logs to identify all web access sessions. The degree of connectivity in each pair of pages depends on two main factors: the time position of two pages in a session and the occurrence of two pages in a session. They proposed an algorithm for modeling the pages accessed as an undirected graph M= (V, E). They proposed a weight measure for approximating the connectivity degree of each two web pages in sessions.

Venketesh *et al* (2010) proposed a semantic prefetching scheme that uses anchor texts present in the web page to make effective predictions. The proposed scheme is responsible for making efficient predictions of web objects to be prefetched for satisfying the user's future requests with low latency. It is based on the concept of client-side prefetching, where the client directly prefetches web documents from the server and stores it in local cache to service user requests. It

significantly reduces the latency when servicing user requests, since there is no network latency for retrieving locally cached documents.

Lenka (2010), proposed a prefetching algorithm to increase the efficiency of Web prefetching and to embody the new demands for Web personalization and Web search assistance. The search engine offers some result links based on the entered keyword and the user starts to evaluate the result links. The user selects a page from the result list and opens it. Current user's browsing history and Markov models predicts next page that will be requested by the user. It also tries to find the next page based on semantic similarity of user's current page and pages linked to it.

Pre-IPGDSF# is an integrated Web caching and Web prefetching in Web servers (Patil, 2011). Intelligent Predictive Greedy Dual Size Frequency# (IPGDSF#) is an innovative algorithm which is an enhanced version of algorithm GDSF. The model also assumes file-level caching. Only complete documents are cached; when a file is added to the cache, the whole file is added, and when a file is removed from the cache, the entire file is removed. The simulation model completely ignores the issues of cache consistency. For prefetching, a static prefetching method was implemented. The prefetching scheme is capable of adapting its behavior based on access statistics.

Rathy and Silky (2011) proposed a bracing approach for increasing web server performance by analyzing user behavior. In their approach, prefetching and prediction is done by preprocessing of logs as it is the main requirement to provide user with best recommendations and also overcomes the limitation of path completion and pattern discovery. Clustering using Markov model and association rules techniques were integrated using frequency, support, and pruning. It achieves complete logs, better accuracy, less state space complexity and less number of rules.

The predicted pages are prefetched and kept in the server cache which reduces the accessing time of that page and increases the web server performance.

Renuka and Amit (2012) studied a hybrid technique based on combination of Artificial Neural Network (ANN) and Particle Swarm Optimization (PSO) for classification of Web object both to cache and generate rules from log data by using Rough Set technique on proxy server (Rough Neuro-PSO). Web caching is done by grouping of Rough Set, ANN and PSO. After applying bandwidth load, Prefetching is placed based on XML on mobile device in order to handle communication between client and server. The smart client will then maintain the timing to access to server. The features are classified on social network into certain priorities based on the popularity and file size.

Bhaskaran *et al* (2012) presented a tailor made ART1 clustering algorithm to group users based on their Web access patterns. A prefetching scheme that uses modified ART1 clustering technique was used to prefetch requests for a large community of users instead of prefetching individual user requests. In the approach, the web log files are being preprocessed, extracting feature of binary pattern vector, clustering user using modified ART1 algorithm, prefetching process and integration scheme.

Bhaskaran *et al*., (2013) proposed prefetching using clustering technique combined with Support Vector Machine (SVM), a machine learning technique for Web proxy caching. Web pages that are referred by various clients are identified from the log file. The Web log file contents are preprocessed and trained using the features namely recency, frequency, retrieval time and size of web object. A Web Navigation Graph (WNG) is constructed for each user independently using a user's session time interval of thirty minutes. A clustering algorithm gets the contents of WNG

as input. Support and Confidence are the parameters used to keep track of frequently visited pages by the user. LFU technique is used for removal of pages from the short term cache if sufficient space is not available for caching a new Web object. SVM classifier is used to classify the Web objects as class 0 or class1 while moving them from short term cache to long term cache. 70% of the requests ordered by time were used for the user's access pattern analysis, creating training dataset and testing. The remaining 30% of the requests were used for testing the scheme. Hit ratio and byte hit ratio were the performance metric used.

Neha *et al* (2013) proposed a framework for reducing web traffic. The data is extracted from the proxy server and then preprocessing is performed. The preprocessed data is then classified and the patterns to be prefetched are obtained. The preprocessed data is classified using the SVM algorithm. SVM algorithm takes a set of input data and predicts for each given input, which of two possible classes forms the output. They show how prefetching combined with classification increases the speed of data retrieval and thus reduces web congestion.

Temgire *et al* (2013) designed an innovative cache replacement algorithm by calculating the prefetching techniques provided by prefetching structure. Web prefetching has been extensively studied, Major approaches of prefetching fall into three categories: probability based, clustering based and using weight-functions. The authors reviewed and categorize different Web prefetching models and parameter.

Neha *et al* (2013) proposed a framework for web traffic reduction.The steps to predict and prefetch the user’s requests are Data extraction from proxy web log, data preprocessing, data clustering, Prediction by Partial Matching (PPM) and prefetching according to the PPM results obtained. In this approach, proxy web log data were collected, preprocessed and then clustered

according to the user access behavior. PPM was applied on the clustered set of data to generate the prefetching rules. Clusters are made on the dataset using fuzzy c-means based clustering algorithm.

Thulase and Meenatchi(2014) proposed a methodology that effectively reduces the web access latency through prefetching and caching. In this approach the cache is divided into two which are object cluster cache and cluster cache. If the object cluster cache is full then LRU replacement policy is applied to purge the cluster and store that into cluster cache if the cluster is classified as needed by the classifier. The classifier uses the recency and frequency of access as the criteria for classifying. If the cluster cache is full then LRU replacement policy is used to purge the cluster for replacement. As LRU Cache replacement algorithm is employed, and clustering is made based on frequency. Figure 2.2 shows the model for the clustering, prefetching and caching mechanism. User's log files are preprocessed and clusters are constructed based on the frequency of access. Prefetching the clusters, is decided based on the frequency of access of the objects in the cluster. The result was analyzed using hit ratio based on a period of one week, one month, four months and six months. The result shows that as the cache size increases the hit ratio also increases.

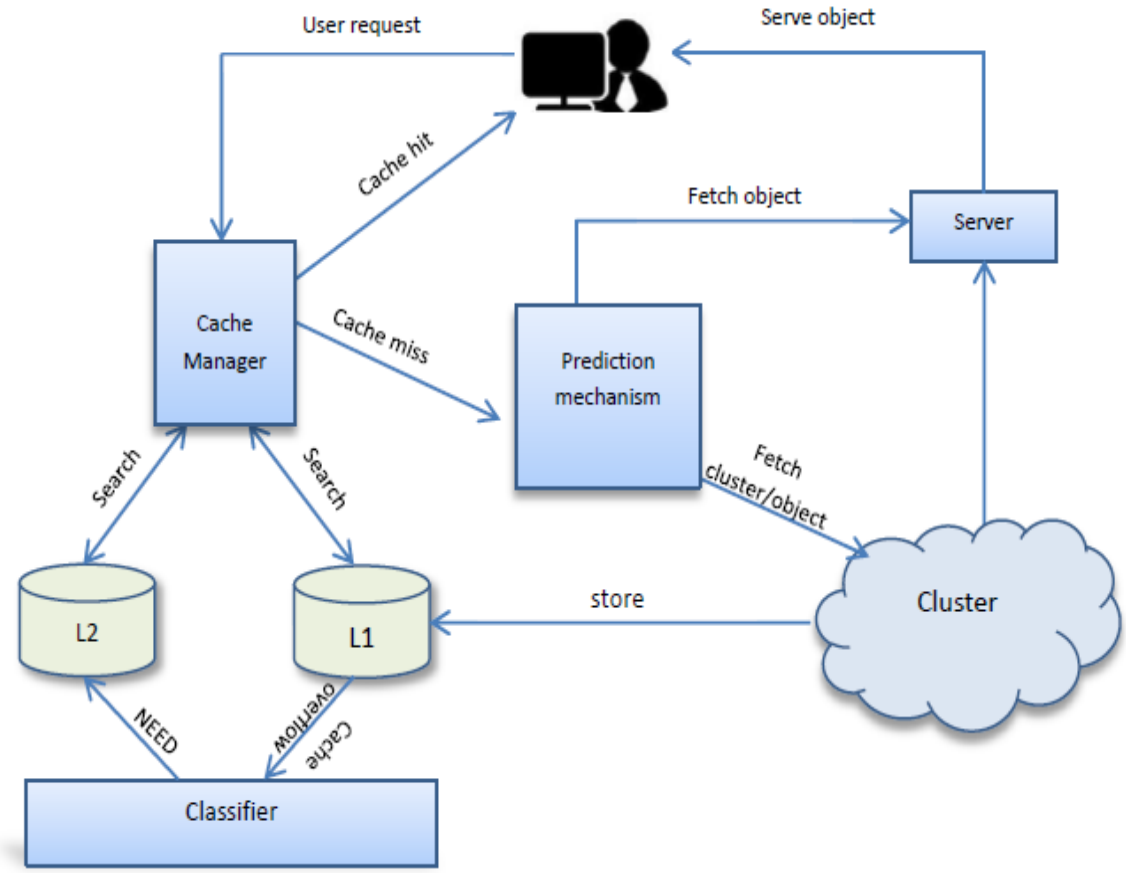

Figure 2.2: Block Diagram for Clustering, prefetching and caching mechanism (Thulase*et al*., 2014)

DoPis a domain based prefetching that presents the user with several generic domains with the top visited web requests in each Domain(Thangaraj*et al*., 2014). The web log files were collected and preprocessed. The preprocessed files were categorized based on its content from the corresponding html file through the meta tag. The classified domain is mapped into web ontology file. The ranking of the web request is carried out by taking URL and the hit rate as the sort keys. The Hit rate, Byte Hit Rate, Waste Ratio and Byte Waste Ratio were the performance metric used to measure the result.

Sathiyamoorthi and Ramya(2014) proposed a prefetching scheme using clustering technique combined with SVM LFU algorithm, a machine learning technique for web proxy caching. Grouping the users was based on their access patterns. The clustering algorithm gets the content of WNG as input. Support and confidence were the parameters used to keep track of frequently visited pages by user. Breadth First Search (BFS) algorithm was applied to the navigational graph. The proxy cache was divided in to short-term cache and long-term cache. The Web objects requested for the first time are loaded in to short-term cache. LFU algorithm was used for managing short-term cache. Those objects visited more than once from the short- term cache are moved to long-term cache.

Sachan and Rao (2014) worked on solving the problem of tie between different objects in a cache. The author used secondary keys as a parameter. LRU algorithm was used to solve the problem. Dataset of 100 different websites were used to check the performance in the basis of hit ratio. The time of access and size of webpage were calculated simultaneously. Using the time of access and size of webpage calculated, eviction of object from cache was possible. The hit ratio was calculated and a graph was generated for hit ratio values.

Bhaskaran and Kalaiarasan (2015) combined prefetching based on clustering technique and caching using Support Vector Machine(SVM) to keep track of the tourist spots that are likely to be visited by the tourists in the near future based on the previous history of visits. In this approach, users are requested to fill a questionnaire that ask for preferences of tourist places to be visited in a particular state in south India. The questionnaire are preprocessed and classified as Class 0 or 1 based on how high or low the frequency, rank and size of object. The frequency, rank and size are incorporated into the SVM classifier and it is trained. Web

Navigation Graph (WNG) is constructed from user's preference pattern of the various tourist destinations. A clustering algorithm gets the contents of WNGs as inputs and two parameters namely Support and Confidence are used to keep track of frequently visited places by the user. Questionnaires were administered which is the best way of data collection and the sample size is limited to a particular state in Indian.

Varunand Nidhi(2015) integrate web caching and web pre-fetchingapproach to improve the performance of proxy server's cache. Access web log files were collected, extracted and preprocessed. Aprori was used for getting unique url and the frequency of the number of times the url was used by user. KMeans clustering approach was used to gather different users into clusters on the basis of their usage behavior and searching pattern. LRU, LFU and FIFO replacement policy were used for the cache replacement policy. Three replacement technique were compared (LRU,FIFO,LFU) on basis of hit rate on cache. The author also explained enhancement of the performance of caching and prefetching using user based approach and analyze the prefetching hit ratio between priority and user based approach.

2.8 Literature Gap and Contribution of this Work

Lots of research has been done on web caching by using the classical techniques of web replacement policies which evolved to combining several replacement policies to get an accurate replacement policy that take into consideration the factors of web objects. Subsequently, researches evolved to combining web prefetching and caching in which both cases has to do with temporal and spatial locality. Several approaches has been used for web prefetching. Clustering based prefetching has been explored in several ways all assume the idle time for prefetching is large enough to accommodate the prefetching. In a real scenario, this is not always the case since

in a high traffic, the idle time may not be so high to accommodate prefetching of large size data. This work therefore seeks to address this lack of consideration of volume of high traffic during the prefetching.

CHAPTER THREE

HIGH TRAFFIC WEB PREFETCHING MODEL

3.1 Introduction

This chapter describes the proposed high traffic web prefetching model. The architecture is presented as well as the corresponding algorithm of its sub models.

3.2 The Prefetching Model Architecture

The internet is a network environment where several users request for web pages. The connection to the web pages is done using the proxy server which is an intermediary between the web browser and the origin server. Several users request for a web page via the internet. The proxy cache is searched for the web page. The web page is fetched from the cache if it is present. The presence of the web page in the cache signifies a cache hit. Otherwise, a cache miss is recorded. The web page is fetched from the origin server. The web page is served to the user within a client group and also saved as a log file. The log files are preprocessed to remove noise. The preprocessed log files is used to update the web navigation graph. A support and confidence threshold value is applied on the navigation graph to remove web pages and edges that have low values. Several clusters are form in a client group. When the predicted prefetch time is greater than the server idle time inter domain cluster is formed. Figure 3.1 shows the web prefetching and caching environment.

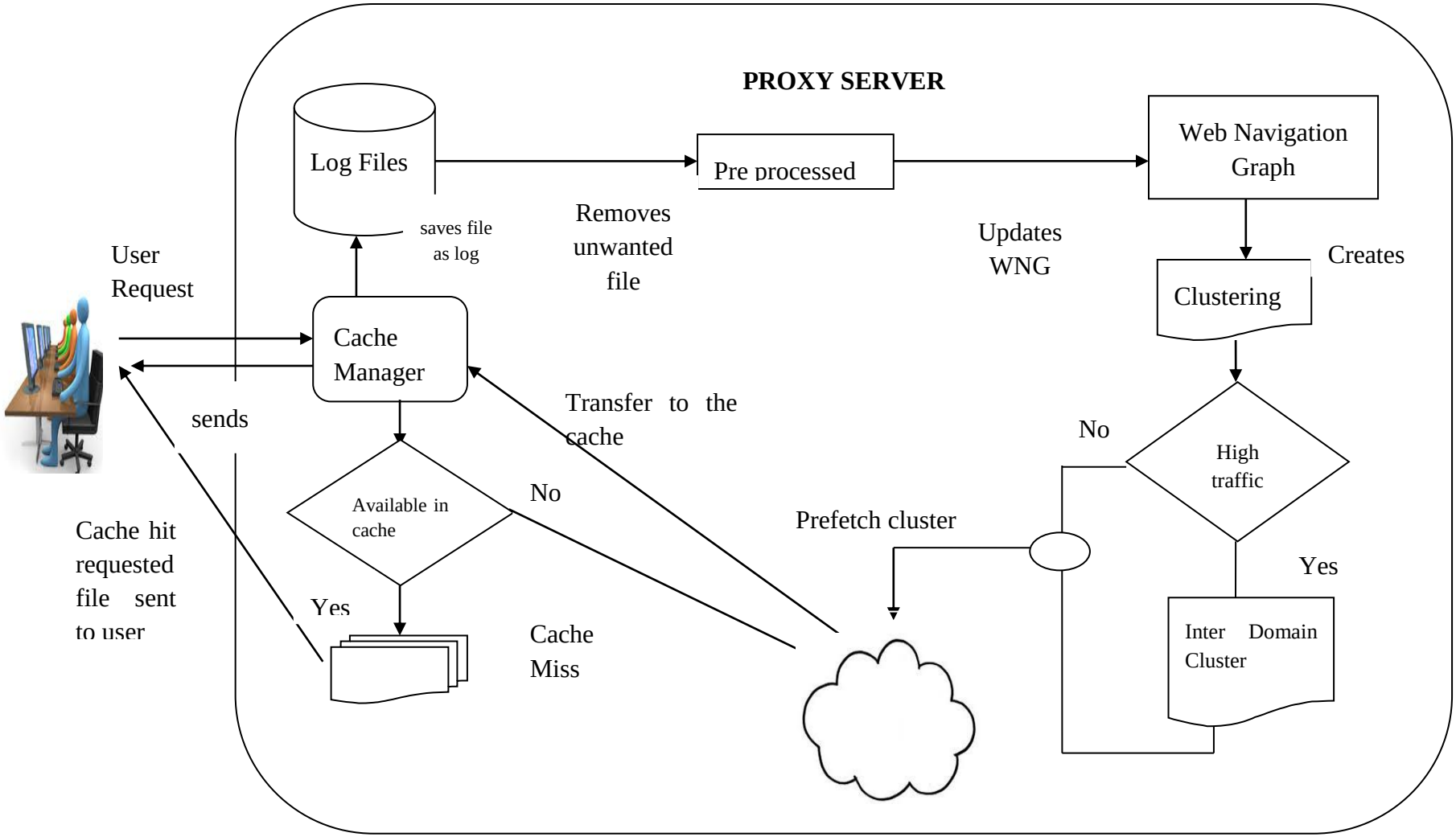

Figure 3.1: A proposed web prefetching and caching environment

Algorithm 1 describes the process of retrieving web pages by several users in a queue from the server and the replacement policy for replacing the web page in the cache.

```
Algorithm WebClustering (P (U0P0, …, Un-1Pn-1))
 // (P (U0P0, …, Un-1Pn-1))Queue of user requested web page
C[0, …, p-1] List of pages stored in the cache.
Q List of clusters to be prefetched
MS Cache memory where the prefetched pages are stored
Begin
        While Not IsEmpty(P)
        Begin
                Pi ←P.dequeue()
                If Is InArray [Pi,C]
                then
                        Hit ←true
                Ui Assign MS[C[Index of (Pi)]]
                Else
                        Hit ←false
                        X ←fetchPage(Pi)
                        Ui. Assign(X)

                        If Not IsFull(MS)
                        then

                                MS[ ]← X
                        Else
                                MS.Remove(LRU Page)
                                MS[ ]← X
                        Endif
                Endif
                Q.enqueue (Clustering(user i))
        Endwhile
        Prefetch (Q)
end
```

Algorithm 1: WebClustering

3.3 Preprocessing of Proxy Access Log Files

Access log files are Data of all incoming request and information about clients of server and it is used to record all requests that are processed by server. Web proxy access log is a sequential file with one user access record per line. Each time a visitor requests any file from the Internet information on his request is appended to a current log file. Most log files have text format and each log entry (hit) is saved as a line of text. Log file range 1KB to 100MB (Shaily, 2012). The Web proxy log files provide information about activities performed by a user from the moment the user logs in to the Internet Service Provider (ISP) to the moment the same user logs out from it. Squid proxy server was installed on the web server to enable the extraction of proxy log files from the server. The web access log file is saved to keep a record of every request made by the users. Web proxy server takes HTTP request from user, gives the web server, then the result is passed to web server and return to user. Clients send request to web server via proxy server.

3.3.1 Data Collection

Data collection is performed in order to retrieve data from data sources for preprocessing. Proxy access log files were collected from the squid server. The log files contain the user's IP address, date and time, duration and accessed site as shown in figure 3.2.

```
893252015.307 14 <client-ip> TCP_HIT/200 227 GET
  http://images.go2net.com/metacrawler/images/transparent.gif - NONE/- image/gif
893252015.312 23 <client-ip> TCP_HIT/200 4170 GET
  http://images.go2net.com/metacrawler/images/head.gif - NONE/- image/gif
893252015.318 38 <client-ip> TCP_HIT/200 406 GET
  http://images.go2net.com/metacrawler/images/bg2.gif - NONE/- image/gif
893252015.636 800 <client-ip> TCP_REFRESH_MISS/200 8872 GET
  http://www.metacrawler.com/ - DIRECT/www.metacrawler.com text/html
893252015.728 355 <client-ip> TCP_HIT/200 5691 GET
  http://images.go2net.com/metacrawler/images/market2.gif - NONE/- image/gif
893252016.138 465 <client-ip> TCP_HIT/200 219 GET
  http://images.go2net.com/metacrawler/templates/tips/../../images/pixel.gif -
  NONE/- image/gif
893252016.430 757 <client-ip> TCP_REFRESH_HIT/200 2106 GET
  http://images.go2net.com/metacrawler/templates/tips/../../images/ultimate.jpg -
  DIRECT/images.go2net.com image/jpeg
```

Figure 3.2: Sample Access log file

3.3.2 Data Cleaning

The purpose of data cleaning is to remove irrelevant information stored in the log files that may not be useful for analysis purposes. When a user accesses an HTML document, the embedded images, if any, are also automatically downloaded and stored in the server log. For example, log entries with file name suffixes such as GIF, JPEG, jpg and JPG are removed. Since the main objective of data preprocessing is to obtain only the usage data. This can be done by checking the suffix of the URL name and erroneous files can be removed by checking the status of the request. The cleaned log represents the user's accesses to the Web site. Figure 3.3 depicts the cleaned access log file.

Figure 3.3: Cleaned access log file

3.3.3 User & Session Identification

The sequence of pages visited by a single user during a visit to a particular site determines the user session. For identification of each user and session measures like IP address and duration on a web page were considered. Once the data cleaning phase is performed, all useful data records are available in the database and irrelevant entries are considered to be removed. Based on the log files collected, Web pages requested by several users at different time interval were compared to know the frequency of request for an object. The higher the frequency, the likely that the object will be requested in the next instance at same time interval.

3.4 Clustering of Preprocessed Log Files

Clustering involves grouping of similar objects based on a common characteristics.Each cluster group consist of objects that are similar between themselves and dissimilar to objects of other groups. A group of users that share a particular IP address are a domain of users. In graph based approach,users are clustered based on their domain. The web object the user request for is

grouped based on the domain using graph based approach (Web Navigation Graph). Analyzing web log files to extract useful patterns is called Web Usage Mining. In order to analyze the interaction between users and web objects, the log files were broken up into different user sessions. From the moment the user logs into the network, the user session begins, all web pages requested by the users are saved as log file. Once the user logs out of the network the user session ends.

3.4.1 Web Navigation Graph (WNG)

Preprocessed log files are used to create navigation graph. This is achieved by grouping the IPs into different categories, according to their domains. Grouping the IPs into categories improves the data management (as the number of domains is smaller than the number of individual IPs).

In web navigation graph,visits to new sites are represented by nodes and movements between web pages are represented by edges. The requests of each client group are represented by a weighted directed Web graph $G(V, E)$, where each node $v_i \in V$ represents a Web page and each edge $e_i \in E$ represents a set of users' transitions from one Web page to another. The weight of each edge is proportional to the number of transitions from one page to another.

For example, assuming a user within a particular client group request for web pages v_1, v_2, v_3 and $v_4$in the following order: v_1, v_2, v_3, v_4, v_1, v_2,v_3, v_4, v_3.The combination of web pages and the frequency form the web navigation graph as shown in figure 3.4.

Client Group Request: $v_1, v_2, v_3, v_4, v_1, v_2, v_3, v_4, v_3$

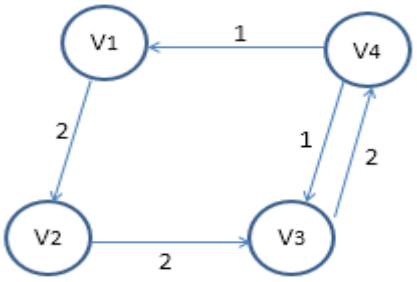

Figure 3.4: Client navigation and the corresponding web navigation graph

To make the size of the WNG manageable, the edges whose connectivity between two Web objects is lower than a specified threshold are removed.A threshold is the minimum acceptable entry point. The edges where the connectivity between two pages is lower than a prespecified threshold is removed, whereas the connectivity between two pages is determined by two parameters: support and confidence.

Support is defined as the discovery of frequent pages, while confidence is defined as the discovery of association rules from these frequent pages.

Given two nodes $v_i$and v_j, Support(v_i,v_j) is the frequency of navigation steps between two web objects ($v_i to\ v_j$). By providing a minimum support, we can limit predictions to those that have had sufficient experience to warrant a good prediction

The Confidence (v_i,v_j) is defined as the ratio of the support of $v_i$and $v_j$and thepopularity of $v_i\ support\ (\mathrm{vi}, \mathrm{vj})\ /pop(V_i)$, where $pop(V_i)$ is the popularity ofv_i.The $pop(V_i)$ denotes the number of request made for a particular page or the number of edges pointing towards a node.

By enforcing a minimum threshold,we can restrict the predictions to those that have high expected probability ofbeing correct.

For example, the support value of the edge $(v_2, v_3)$for the user in the client group in Figure 3.2 is (v_2, v_3) = 2 and confidence value is$\frac{freq\ \ (v_2,\ \ v_3)}{pop(v_2)}$ =1.

If the support threshold value chosen is low, too many unimportant user's transitions for clustering may be included and if the chosen threshold value is high, many interesting transitions that occur at low level of support may be missed. Algorithm 2 shows the process of creating web navigation from the log files collected. Support and confidence threshold value is applied to form several clusters in a domain.

```
Algorithm Clustering (W [0,…,n-1])
//Input W [0,…,n-1] array of pages
For i = 0 to n-1
{
        Vi = W[i]
        IfIsaNode of (G, vi) Then
                Weight(e(vi-1, vi)) = Weight(e(vi-1, vi)+1)
        Else
                Create node (vi)
                If (vi is notroot node)then
                Weight(e(i-1, vi))= Weight(e(vi-1, vi) + 1)
}
For each vi in V,
{
        Calculate support=(vi to vj)
        Calculate confidence =freq  (v2,  v3) / pop(v2)
        If confidence < threshold Then
                Remove(vi)
        Endif
        If support < threshold Then
                Remove(ei)
        Endif
}
```

Algorithm 2: Clustering Algorithm

For example, the support and confidence threshold for figure 3.2 are 1 and 0.6 respectively. The resulting clusters for the client group will be in figure 3.5.

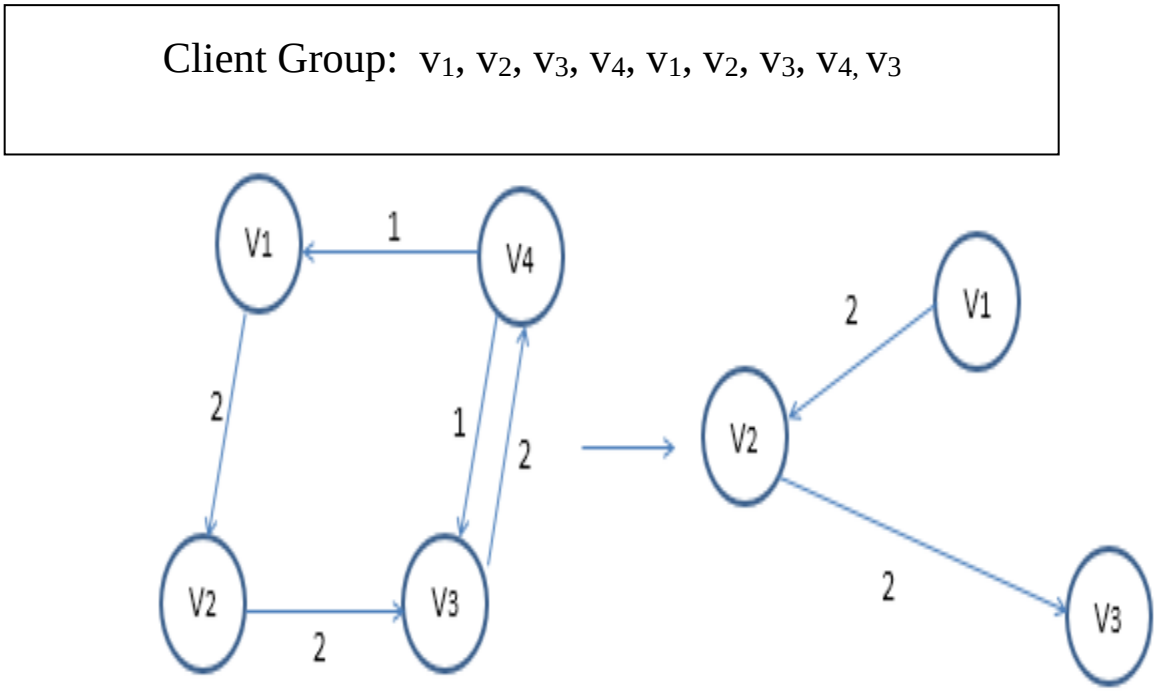

Figure 3.5: Clustering of web objects in a client group

3.7 Inter Clustering

Prefetching can only be done at CPU idle time and clustering based prefetching predict users request based on domain. In a situation where there is high traffic, prefetch web objects that will be requested by several users in different domain. High traffic is determined if the frequency of the server idle time is not enough to prefetch all clusters arriving at the server. This happens if the ratio of the estimated prefetch time and the server idle time is greater than zero. If the estimated prefetch time is greater than the server idle time prefetch inter domain else prefetch

clusters. The server will be able to prefetch the cluster if the server idle time is less than the cluster prefetch time.

Let P_t denotes the estimated prefetch time of the clusters to be prefetched from the server at time t, n denotes the number of clusters in a domain, k denote the number of pages in a cluster, $S_{i,j}$ denotesthe size of pages(j) in a cluster(i) and B denote bandwidth. Then,

$$P_t = \frac{\sum_{i=1}^{n} \sum_{j=1}^{k} S_{ij}}{B} \qquad \text{Equation 3.1}$$

Equation 3.1 assume that the prefetch time depend only on the page size and bandwidth only. For example, assuming a cluster containing three web pages of sizes 5 byte, 10 byte and 15 byte respectively and the bandwidth is 256 byte per seconds. The estimated prefetch time will be $P_t = \frac{5+10+15}{256} = 0.1172$ seconds.

Let S denote the average server idle time, if Pt <= S, then the prefetch of all clusters is possible otherwise, the prefetch is not possible. In a high traffic and low bandwidth situation, since all the clusters cannot be prefetched due to limited server idle time, clusters created for each domain will be used to form an inter domain cluster. This inter domain clustering will help to increase accuracy in predicting only the mostly requested web pages.

Assuming we have several request from different domain, the cluster of each request in a domain will be prefetched and stored in the cache. If the estimated prefetch time is greater than the idle time. The cluster from each domain will be used to form a prioritized inter domain cluster. The priority will be set by using a support and confident threshold value. The inter domain cluster with the highest priority will be prefetched first subsequently others will follow. Algorithm 4

elaborates on the inter domain clustering.Algorithm prefetch depict the algorithm for forming prioritized inter domain clusters.

Algorithm prefetchCluster(Q[p_1,p_2,…,p_n], serveridletime, bandwidth)
Input: Q[q1,…,qn] of web cluster to be prefetched where each cluster consist of set of pages p_i
Output: prefetched web object
priorityQueue[$P_{i,k}$] array of priority page to be prefetched where I is the page index and k is the number of occurrence of the page

```
While(IsEmpty(Q) ==false) do
        Prefetchtime = Σ_{i=1}^{n} Σ_{j=1}^{k} p_{i,j} / Bandwidth
        If(prefetchtime<=severidletime)        Then
                q=Q.dequeue()
                While(IsEmpty(q) ==false) do
                        While !(serverIsBusy)
                                For each pi in q
                                Prefetch pi
                                q.remove(Pi)
                        end
end
        else
                PriorityQueue== null
                While(IsEmpty(Q) ==false) do
                        q=Q.dequeue()
                        For each Pi in q
                                If Pi IsInArray(PriorityQueue) Then
                                        Pi,k=Pi,k+1
                                Else
                                        PriorityQueue [] = Pi,1
                end
        end
        priorityQueue.sort()
while (IsEmpty(PriorityQueue)==false)do
        i=0
        while!(ServerIsBusy) do
                P=PriorityQueue[i]
                Prefetch(P)
                PriorityQueue.remove(P)
                i++
end
end
endif
end
```

Algorithm 3: Algorithm Prefetch

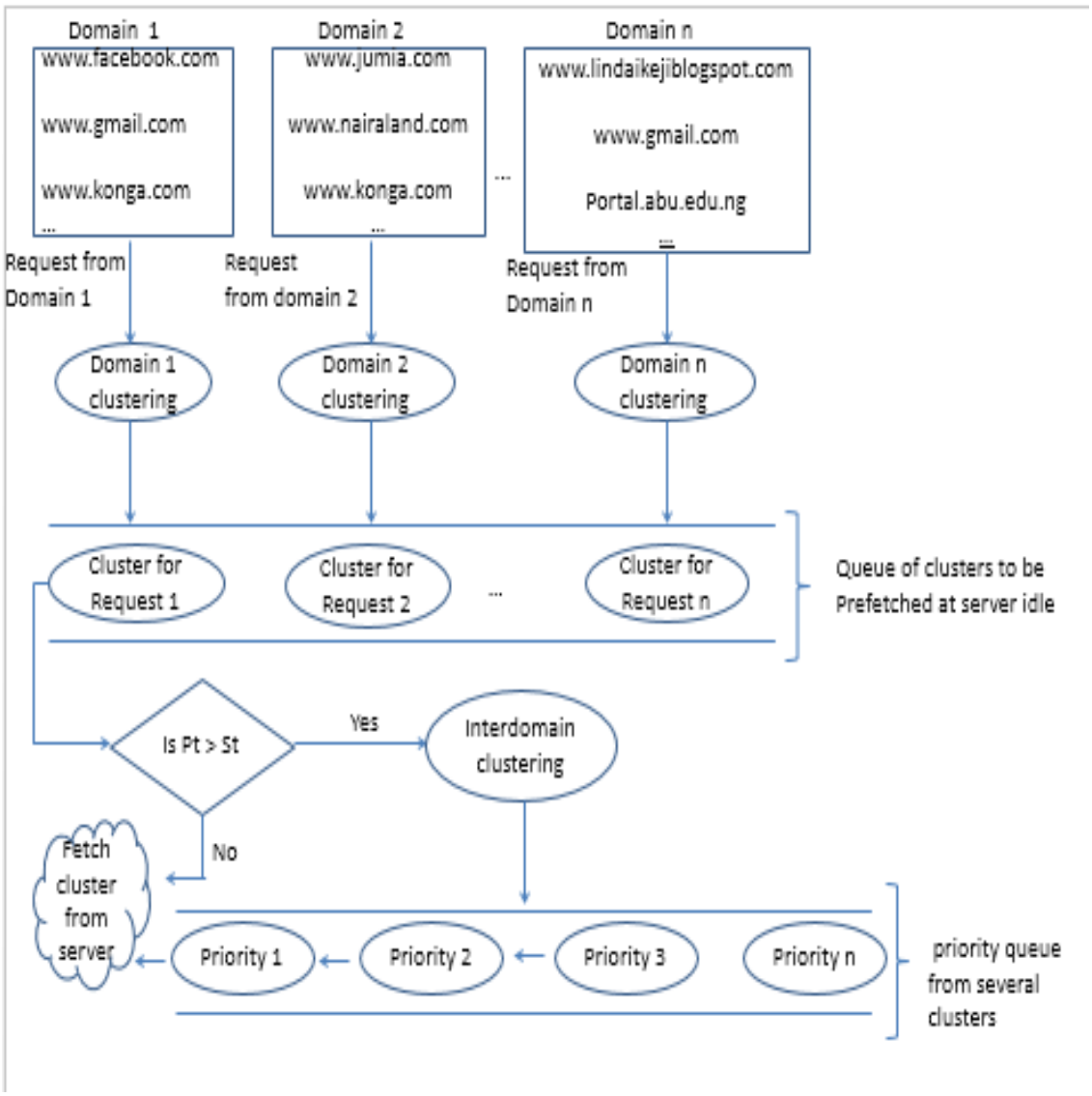

Figure 3.6: Prefetching in high traffic environment

Assuming there are several clusters in several domain 1,2,…,n. The frequency of each webpage in the cluster will be used to form the prioritized interdomain cluster. Figure 3.7 depicts the image of clusters in several domain. V_1 has a frequency of three, v_2 has a frequency of two, v_3

has a frequency of three and v_4 has a frequency of one respectively. Web page v_1 will be removed from the queue followed by v_3, v_2 and v_4 depending on the server idle time.

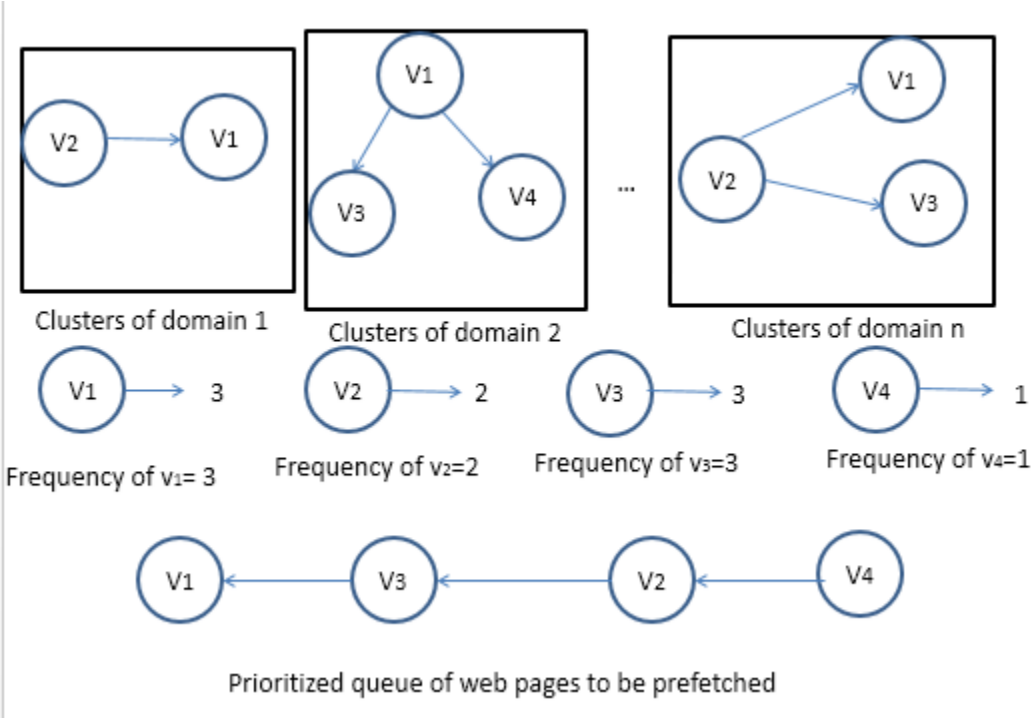

Figure 3.7: Prioritized prefetching

CHAPTER FOUR

IMPLEMENTATION AND RESULT

4.1 Introduction

This chapter discusses implementation details of the proposed model as well as the result of the experimental evaluation of the model. Its performance is also compared with the simulated algorithm when using InterClustering and Thulase*et al.*,(2014).

4.2 Implementation Details

4.2.1 Programming Language

The proposed model was implemented using C#. C# (pronounced "C sharp") is the native language for the .NET Common Language Runtime. It has been designed to fit seamlessly into the .NET Common Language Runtime (CLR). In cooperation with the .NET Common Language Runtime, it provides a language to use for component oriented software, without forcing programmers to abandon their investment in C, C++, or COM code. C# is designed for building robust and durable components to handle real world situations. The .NET Common Language Runtime is a component based environment. It's a component centric language, in that all objects are written as components, and the component is the centre of the action.

Component concepts, such as properties, methods, and events, are first-class citizens of the language and of the underlying runtime environment.

4.2.2 Squid Proxy Server

Squid was used as the proxy server to capture the log files. Squid is a caching proxy server, which improves the bandwidth and the response time by caching the recently requested web pages. All the requests originating from client and going to internet on port 80 are automatically

redirected by proxy. Depending on the requirement configuring the squid as transparent or non-transparent proxy is needed. A web proxy makes the HTTP request on behalf of the client and caches the result for later. That way, if multiple users request the same document, the cache can serve its local copy, rather than wasting bandwidth. The benefits are users get their content faster (when it's cached) and the internet connection isn't bogged down with duplicate requests. Squid proxy server enables you to cache your web content and return it quickly on subsequent requests. By deploying Squid in accelerator mode, requests are handled faster than on normal web servers, thus making the site perform quicker.

4.2.3 Dataset

The dataset used for the experiment was obtained from Ahmadu Bello University squid proxy server. Log files for four weeks were captured. The raw file for the dataset is shown in table 4.1.

Table 4.1 Dataset Table

Total Number of web pages	155,785
Total Size of Byte of web pages	245MB
Total Number of web pages used for Training	86,325
Total Size of Bytes Used for Training	85.4MB

Seventy percent of all the preprocessed log files have been used for the user's access pattern analysis, creating training dataset and testing. The remaining 30% of the requests were used for testing the scheme.

4.3 System Specification

The hardware and software specification used for the simulation of the algorithms are as follows

Hardware

a Hewlett Packard (HP) laptop with a (Tm)i3-311om processor running at 2.40GHz

b 8.00GB of RAM and

c 500GB of hard disk

Software

a Windows 7 operating system

b Microsoft Visual Studio

4.4 Implementation Result

4.4.1 System Model

The proposed model was compared with the prefetching technique of Thulase *et al*., (2014).For the experimental purpose the cache size is assumed to be unlimited, the bandwidth size is assumed to be 512byte per seconds, the support and confidence threshold value are assumed to be 1 and 0.5 respectively and the server idle time range to a time frame where WebClustering proposed algorithm and that of Thulase *et al* (2014) prefetch same content.

4.5 Performance Evaluation Criteria

The performance metrics used to evaluate the proposed web prefetching techniques are the following:

Hit Rate: The hit rate (HR) is the ratio between the number of requests that hit in the cache and the total number of requests. Hit rate refers to the percentage of user access requests that are

found in the prefetching cache. We tested the cache hit ratio for different cache sizes over a period of time.

Byte Rate: Byte Rate (BR) is the ratio between the number of bytes that hit in the proxy cache and the total number of bytes requested sent to its clients. A high HR indicates the user's satisfaction and defines an increased user servicing. On the other hand, a high BHR improves the network performance and reduces the user-perceived latency.

Precision: It is the ratio of the number of correct predictions to the number of total predictions. If users in the subsequent requests access the predicted page that is in the prefetching cache, the prediction is considered to be correct, otherwise it is incorrect. The metric represents the fraction of predicted pages that are actually used.

Accuracy of prediction: Accuracy of prediction is the percentage of predicted pages that were later actually requested by the user.

Usefulness of prediction: It defines the percentage of predicted pages that had already been predicted is being prefetched by the user.

4.6 Discussion of Results

In order to evaluate the performance of the proposed algorithm, a set of experiments were conducted to measure hit ratio, byte ratio, accuracy of prediction, usefulness of prediction and precision against the server idle time in seconds.

4.7.1 Accuracy of Prediction

The accuracy of the prediction was measured based on different server idle time at interval of 10 seconds until the proposed algorithm and that of Thulase *et al* (2014) converged. Table 4.2 show

the accuracy of both algorithm at different server time and figure 4.1 show the corresponding accuracy graph.

Table 4.2 Accuracy of Prediction

Server Idle time(Seconds)	WebClustering(%)	Thusule*et al.*,(2014)(%)
0	0	0
10	32	7
20	44.4	12.3
30	54.3	28.4
40	57.3	35
50	59.2	49
60	61	58
70	62	62

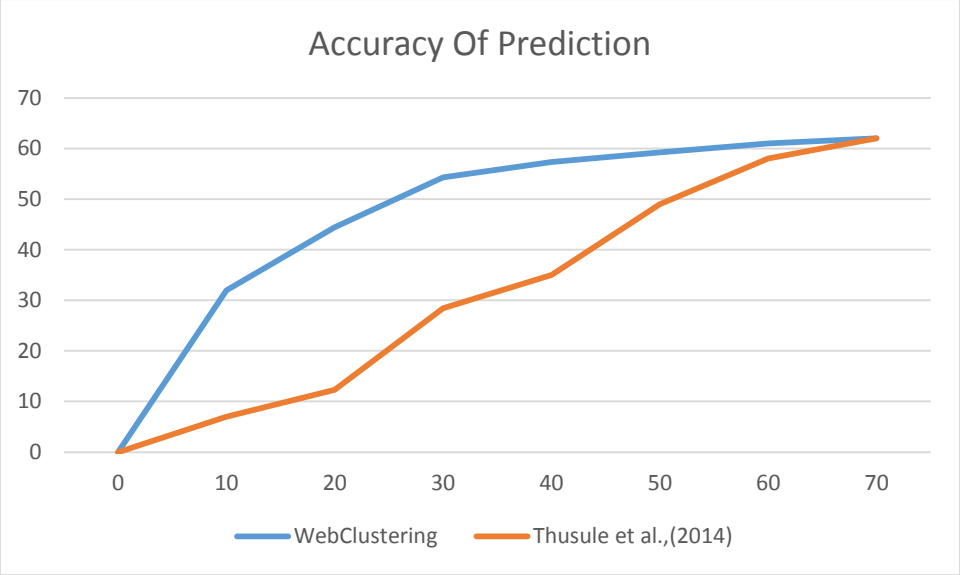

Figure 4.1 Graph of Accuracy of Prediction at different time interval

The result in figure 4.1 shows that as the server idle time increases the accuracy of prediction for webclustering and Thulase *et al* (2014) converge to a point at which they become similar.

4.7.2 Usefulness of Prediction

The usefulness of prediction was measured based on different server idle time at interval of ten seconds until the proposed algorithm and that of Thulase*et al* (2014) converge. Table 4.3 shows the usefulness of prediction at different server idle time and figure 4.2 shows the corresponding usefulness graph.

Table 4.2 Usefulness of Prediction

Server Idle time (Seconds)	WebClustering (%)	Thusule*et al*.,(2014)(%)
0	0	0
10	9	2
20	12.3	9
30	21	13
40	28	24
50	34	34
60	40	40

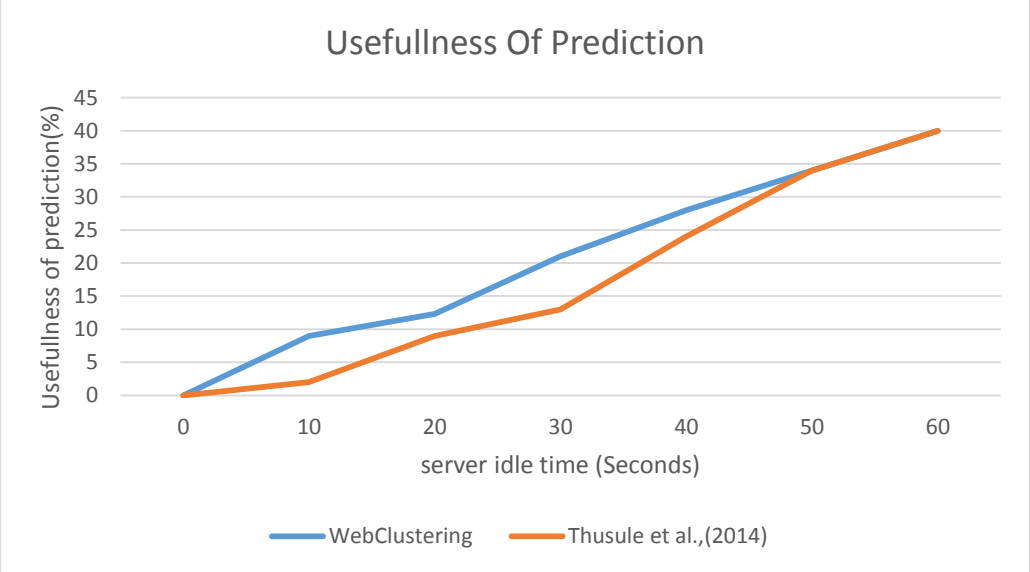

Figure 4.2: Graph for usefulness of prediction at different time interval

The result in figure 4.2 shows that as the server idle time increases the usefulness of prediction for webclustering and Thulase*et al*(2014) converge to a point at which they behave the same.

4.7.3 Precision

The precision was measured based on different server idle time at interval of ten seconds until the proposed algorithm and that of Thulase*et al* (2014) converge. Table 4.4 show the precision of both algorithm at different time in interval and figure 4.3 shows its corresponding precision graph.

Table 4.3: Precision

Server Idle time (seconds)	WebClustering(%)	Thusule*et al*.,(2014)(%)
10	80	12
20	77.4	23
30	70	35
40	63	47
50	60	58
60	62	62

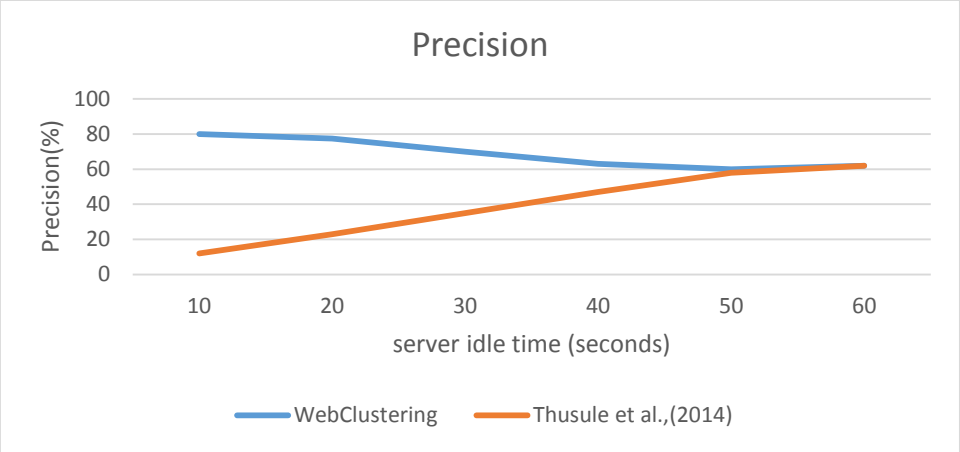

Figure 4.3: Graph of Precision at different time interval

The result in figure 4.3 shows that as the server idle time increases the precision reduces for webclustering and for Thulase*et al* (2014) the precision increases and at a point the precision behave the same.

4.7.4 Hit Ratio

The hit ratio was measured based on different server idle time at interval of ten seconds until the proposed algorithm and that of Thulase*et al* (2014) converged at a point. Table 4.5 shows the hit ratio of both algorithm at different time in interval and figure 4.4 shows its corresponding hit ratio graph.

Table 4.4: Hit Ratio

Server Idle time	WebClustering	Thusule*et al*.,(2014)
0	0	0
10	30	18
20	43	22
30	58.5	45
40	62	54
50	64	60
60	68	65
70	70	70

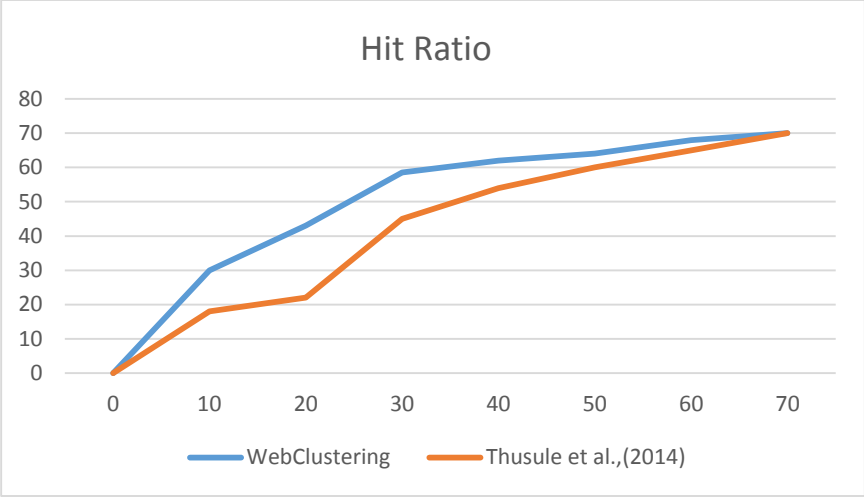

Figure 4.4: Graph of Precision at Different time interval

The result in figure 4.4 shows that as the server idle time increases the hit ratio for webclustering and Thulase*et al*(2014) behaves similar.

4.7.5 Byte Ratio

The byte ratio was measured based on different server idle time at interval of ten seconds until the proposed algorithm and that of Thulase*et al* (2014) converge. Table 4.6 shows the byte ratio of both algorithm at different time in interval and figure 4.5 shows its corresponding byte ratio graph.

Table 4.5: Byte Ratio

Server Idle time(seconds)	WebClustering(%)	Thusule et al.,(2014)(%)
0	0	0
10	30	18
20	43	22
30	58.5	45
40	62	54
50	64	60
60	68	65
70	70	70

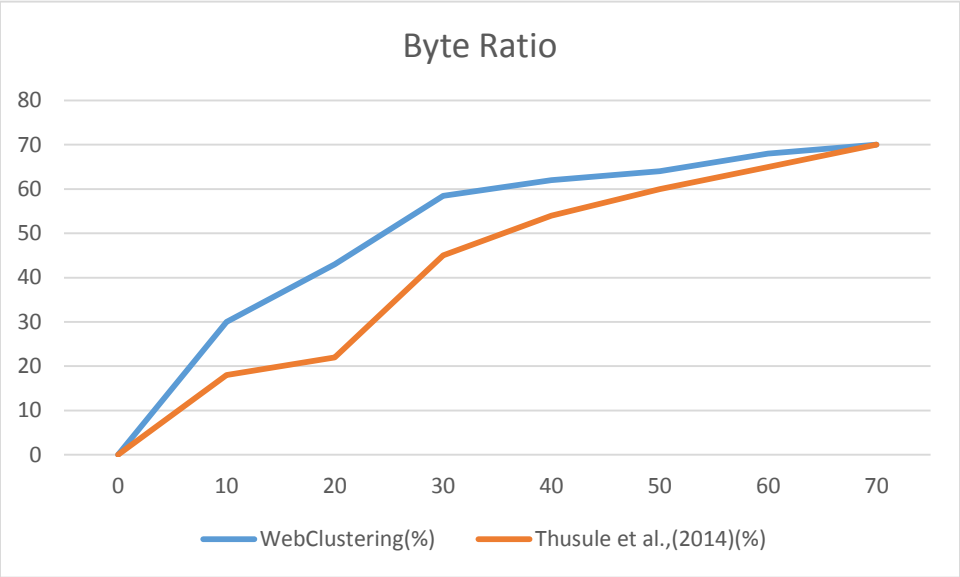

Figure 4.5: Graph of Byte Ratio at different time interval

The result in figure 4.5 show that as the server idle time increases the byte ratio for webclustering and Thulase *et al* (2014) behaves similar.

4.7 Summary of the Result

From the result of all the comparism it shows that the proposed algorithms perform better when the server idle time is small and behaves almost same as that of Thulase *et al* (2014) as the server idle time increases. This is because as the server idle time increases, the two algorithm behave the same. Since, inter domain clustering is no longer performed.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Summary

Web prefetching is predicting user web object prior to the request of the web page. Prefetching is only done at server idle time. The prefetching problem addressed on a Web cache environment using an algorithm for clustering web pages.In this work, a technique to improve and prefetch at the server idle time is proposed. Users' log files was collected and preprocessed. Clustering of the Web objects in the WNG is done. Availability of requested Web object in the cache leads to prefetching of all other Web objects in the cluster during server idle time. In a situation where the predicted server idle time is greater than the actual server idle time, inter domain cluster is performed. Several clusters will be formed from different domain based on users' request. The proposed algorithm is evaluated based on hit ratio, byte ratio, precision, accuracy of prediction and usefulness of prediction were the performance metrics used.

5.2 Conclusion

Web prefetching is only done at server idle time. This research improves the prefetching at the smallest server idle time. From results of the experiments carried out, the proposed algorithm webclustering, improved on addressing the high traffic problem thereby enabling prefetching at the minimum server idle time.

5.3 Recommendations

Although the result obtained from the work shows some intensity result as compared to the standard prefetch technique, the measure of the estimated prefetch time depends only on the page

size and bandwidth. Future work should look into other parameters that affect the speed of fetching a web page including the processor speed, the file location.

REFERENCES

Baskaran, k., and Kalaiarasan, C. (2015). Combining Pre-fetching and Intelligent Caching Technique (SVM) to Predict Attractive Tourist Places. *Research Journal of Applied Sciences, Engineering and Technology* , 9(1), 40-46.

Baskaran, k., Kalaiarasan C., and Sasi, A. (2013). Study of combined web prefetching with web caching based on machine learning technique. *Journal of Theoretical and Applied Information Technology*, 55(2) , 280-291.

Bhaskaran, V., and Murali, V. (2012). Optimizing the Web Cache Performance by Clustering based Pre-Fetching Technique using Modified ART1. *International Journal of Computer Applications*, 4(1) , 50-57.

Cheng-Zhong, X., and Tamer I. (2000). Semantics-Based Personalized Prefetching to Improve Web PerformanceI. *Institute of electrical and electronic engineering* , 20(2), 636-643.

González-Cañete, F., (2007). A Content-type Based Evaluation of Web Cache Replacement Policies. *International Conference Applied Computing* , 2(3), 90-96.

Greeshma, G., and Jayasudha, J. (2012). A Survey on Web Prefetching and Web Caching in a Mobile Environment. *International Journal of Computer Science and Information Technology*, 2(1), 119-136.

Jeeva, J., and Sojan, P., (2012). An Indiscernibility Approach For Pre Processing Of Web Log File. *International Journal Of Internet Computing*,1(3) , 58-61.

Lenka, H. (2010). Semantic Web Access Prediction Using WordNet. *Proceedings of the Doctoral Consortium of the International Conference on Web Engineering*, 404, 21-35.

Mehrdad, J., Norwati, M., Ali, M., and Nasir, B. (2008). Web User Navigation Pattern Mining Approach Based on Graph Partitioning Algorithm. *Journal of Theoretical and Applied Information Technology* , 2(5), 1125-1130.

Neha, K., Esha, D., and Neha, S. (2013). Proposed Framework for the Reduction of Web Congestion using Classification. *International Journal of Engineering and Advanced Technology,* 3(2) , 123-128.

Neha, S., and Sanjay K. (2013). Fuzzy C-means Clustering Based Prefetching to Reduce Web Traffic. *International Journal of Advances in Engineering & Technology,* 6(1) , 426-435.

Padmapriyya, V., and Thenmozhi, k. (2013). Web caching and Response time Optimization Based on Eviction Method. *International Journal of Innovative Research in Science, Engineering and Technology,* 2(4), 1171-1177.

Pallis, G., Athena, V., and Jaroslav, P., (2007). A Clustering-Based Prefetching Scheme on a Web Cache Environment. *Computers and Electrical Engineering* , 34, 309-323.

Patil, J., and Pawar, B. (2011). Integrating Intelligent Predictive Caching and Static Prefetching in Web Proxy Servers. *International Journal on Computer Science and Engineering,* 3(2) , 697-704.

Ramu, k., Sugumar, R., and Shanmugasundaram, B. (2012). A Study on Web Prefetching Technique. *Journal of Advances in Computational Research,*1(2) , 39-46.

Rangarajan, S., Virv, P., Kiran, B., Iyengar, S., and Rastko, S. (2002). Web User Clustering and Its Application to Prefetching Using ART Neural Networks . 1-17.

Rathy, K., and Silky, M. (2011). Web Server Performance Optimization using Prediction Prefetching Engine. *International Journal of Computer Applications,*23(9) , 19-24.

Sachan, D., and Rao, D. (2014). Performance Improvement of Web Caching Page Replacement Algorithms . *International Journal of Computer Science and Information Technologies*, 5(3) , 3112-3115.

Sathiyamoorthi, V., and Ramya, P. (2014). Enhancing Proxy Based Web Caching System Using Clustering Based Pre-fetching with Machine Learning Technique. *International Journal of Research in Engineering and Technology*, 3(7) , 463-469.

Shaily, L., Mehul, B., and Darshak, M. (2012). Pre-Processing: Procedure on Web Log File for Web Usage Mining. *International Journal of Emerging Technology and Advanced Engineering*, 2(12) , 419-423.

Renuka, S., and Amit, S. (2012). Web Caching and XML Prefetching for Accessing Social Sites from Mobile Environment . *International Journal of P2P Network Trends and Technology*, 2(1) , 38-41.

Temgire, S., and Poonam, G. (2013). Review on Web Prefetching Techniques. *International Journal of Technology enhancements and emerging engineering research*, 1(4) , 100-105.

Thangaraj, M., and Meenatchi, V. (2014). Domain Based Prefetching in Web Usage Mining. *International Journal of Advanced Computer Science and Applications*,5(6), 53-59.

Thulase, M., and Raju, G. (2014). Effective Web Access Latency Reduction Through Clustering Prefetching and Caching. *International Journal of Electrical & Computer Sciences* ,1(2), 7-12.

Varun, K., and Nidhi, S. (2015). A Novel Approach For Web Prefetching and Caching. *International Journal of Engineering Research and General Science*, 3(3) , 372-379.

Venketesh, P., Venkatesan, R. and Arunprakash, L. (2010). Semantic web prefetching scheme using NAÏVE Bayes Classifier. *International Journal of Computer Science and Applications*, 7(1), , 66-78.

Waleed, A., Siti, M., and Abdul, S. (2011). A Survey of Web Caching and Prefetching. *International Journal of Advance Soft Computing*, 3(1), 12-17.

Yang, Q., Joshua, Z., and Michael, N. (2003). A Data Cube Model for Prediction-Based Web Prefetching. *Journal of Intelligent Information System*s ,1(2), 11-30.