***I.J. Education and Management Engineering*, 2018, 3, 21-32**
Published Online May 2018 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2018.03.03

Available online at http://www.mecs-press.net/ijeme

Web Clustering based Prefetching in High Traffic Environment

F. O. Atta[a*], A. F. Donfack Kana[a]

[a]*Department of Mathematics, Ahmadu Bello University, Zaria Kaduna, Nigeria*

Received: 20 September 2017; Accepted: 20 March 2018; Published: 08 May 2018

Abstract

The need of minimizing the latency perceived by the user in fetching web objects without necessarily increasing the bandwidth has attracted several researchers in the recent years. Although web prefetching and caching is seen as a solution, proposed techniques do not consider the frequency of server idle time in their models. This paper therefore proposes a short time web prefetching framework based on clustering technique that can be effective in high traffic with low bandwidth environment where the server idle time is too minimal to fetch all users anticipated requests. Clusters from different user requests are used to perform an inter domain clustering that prioritizes the prefetching of web pages based on speed at which requests are received from each domain and the popularity of each page. Experimental results show an improvement in hit rate and precision over the classical clustering based prefetching technique when the server idle time is not enough to prefetch all clusters.

Index Terms: Web Prefetching, clustering, low bandwidth, high traffic.

1. Introduction

The web is a collection of text documents and other resources, linked by hyperlinks and Uniform Resource Locators (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 [7]. 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 increasing economic cost. An alternative solution is web prefetching and caching.

Web pre-fetching is a mechanism for prefetching web pages into the cache before the actual request

* Corresponding author.
E-mail address: attafatimah@gmail.com

arrives[11][13]. Web prefetching helps to fetch and cache users request during server idle time, which will reduce the load on the origin server. Web caching reduces the latency perceived by the user, reduces bandwidth utilization and reduces the loads on the origin servers [9]. To reduce the access delay experienced by users, it is wise to predict and prefetch web object based on user access patterns and cache them. Figure 1 shows a classical web prefetching and caching architecture.

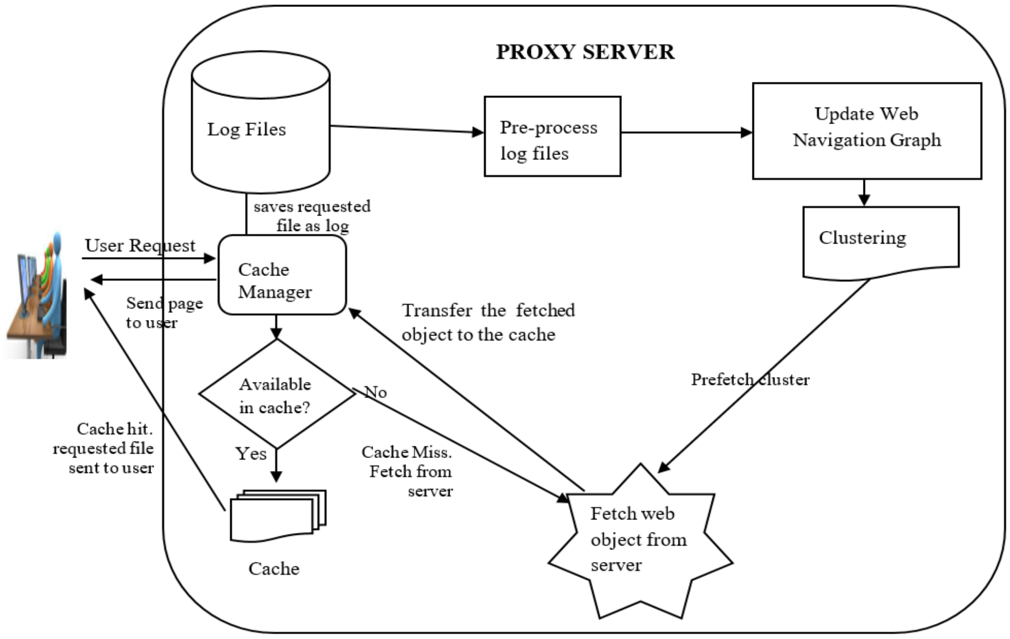

PROXY SERVER
Log Files
Pre-process log files
Update Web Navigation Graph
saves requested file as log
User Request
Cache Manager
Clustering
Send page to user
Transfer the fetched object to the cache
Prefetch cluster
Available in cache?
No
Cache hit. requested file sent to user
Yes
Cache Miss. Fetch from server
Fetch web object from server
Cache

Fig.1. Classical Prefetching Architecture (Adopted from [9])

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* is followed by *B* with a high probability, then *B* will be prefetched once *A* is accessed [5]. 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 [6].

Prefetching techniques can be implemented on the client side, server side or on the proxy server 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. The latter method makes prediction based on observed page access behaviour of the user in the past [2] [14]. In a situation where several users request are processed by the server and the server idle time is not enough to prefetch all the request, it therefore calls to determine which among the several web pages to be prefetched should be of high priority in order to optimize the overall performance. This paper therefore proposes an approach that uses inter domain clustering to prioritize the prefetching in order to optimize the overall prefetching performance.

The rest of the paper is organized as follows: Section 2 reviews some related works, the proposed model is discussed in section 3, section 4 discusses the performance evaluation and finally, section 5 concludes the work.

2. Related Works

The web is rich in information which are in the hidden form. Many researchers have applied the data mining techniques to extract the interesting information from the web[4]. Several works have been carried out in the recent years in order to improve the performance of web prefetching and caching.

A clustering based prefetching scheme called clustpref was proposed in [9]. Clustpref is a graph based clustering algorithm that identifies clusters of Web pages based on users' access patterns. It uses the preprocessed web log file to create Web navigational graph. Support and confidence value are applied on the web navigation graph to form several clusters within a domain. Whenever a user request for a web object, the proxy server is checked for the object, if it is present in the proxy server, it is retrieved and send to the user. If the web object isn't present in the cache, it is fetched from the origin server and the entire cluster in which the object belong is prefetched with the hope that the next request will be a page in that cluster.

A prefetching technique that combine the clustering technique with Support Vector Machine (SVM) was proposed in [2]. SMV is a supervised learning technique with associated learning algorithm primarily used for classification or regression. In their approach, web log file contents are preprocessed and trained using the features namely recency, frequency, retrieval time and size of web object. The preprocessed log file is used to form WNG. Support and confidence threshold values are applied on the WNG to form several clusters within a domain. The cache is divided into two which are, short term and long term cache. Whenever the cache is full, LFU technique is used for page replacement. SVM classifier is used to classify the Web objects as class 0 or class 1 while moving them from short term cache to long term cache. Hit ratio and byte hit ratio were the performance metric used. The result showed that increase in confidence and support value leads to increase in the number of clusters and decrease in the number of web objects in the cluster created.

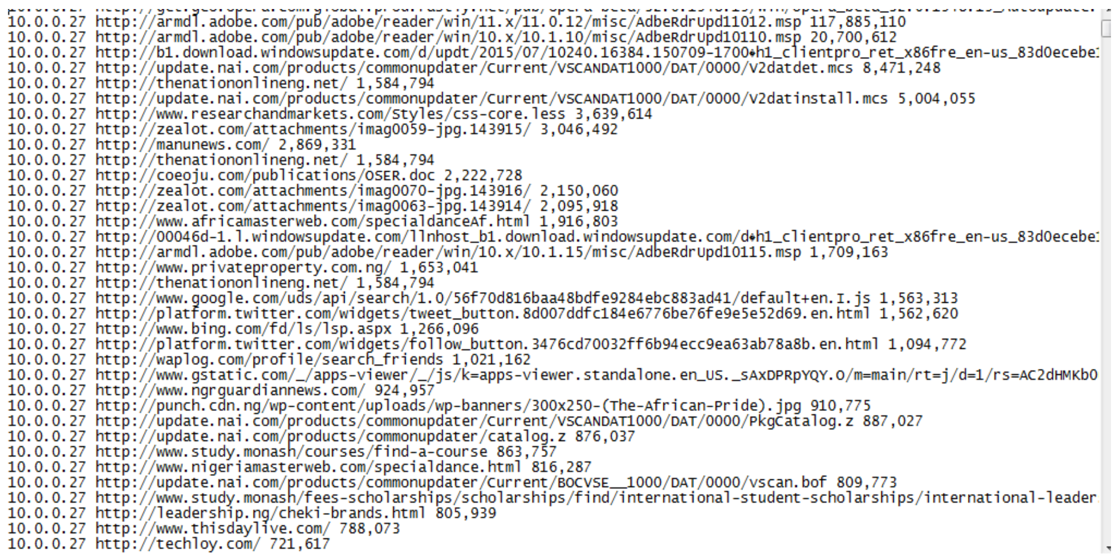
A framework for reducing web traffic was proposed in [7]. In their approach, the data extracted from the proxy server is pre-processed and is used to determine the patterns to be prefetched. The pre-processed data is classified using the SVM algorithm. SVM algorithm takes a set of input data and predicts for each given input, which of two possible classes (class 0 or class 1) forms the output. They show how prefetching combined with classification increases the speed of data retrieval and thus reduces web congestion.

[15] proposed an approach to reduce the web access latency through prefetching and caching. In their 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 used to purge it and store the content into cluster cache if cluster is classified as needed by the classifier. The classifier uses the recency and frequency of access as the criteria for the classification. If the cluster cache is full then LRU replacement policy is used to purge the cluster for replacement. User's log file are pre-processed and clusters are constructed based on the frequency of access of web pages which form cluster within a domain. The result was analysed based on hit ratio for a period of one week, one month, four months and six months. The result showed that as the cache size increases, the hit ratio also increases.

In [13], a prefetching scheme using clustering technique combined with Support Vector Machine Least Frequently Used (SVMLFU) algorithm is proposed. Their technique group users based on their access patterns. The clustering algorithm gets the content of WNG as input. Support and confidence are used to keep track of frequently visited pages by user. The technique divides the proxy cache into short-term cache and long-term cache. The Web objects requested for the first time are loaded into short-term cache. LFU algorithm is used in the short-term cache replacement.

3. Web Prefetching and Caching Technique

The model adopted in this paper is based on clustering technique. Whenever a page is requested by the user, the proxy cache is searched for the web page. Upon a cache hit, the web page is fetched from the cache. Upon a cache miss, the web page is fetched from the origin server and is served to the user within a client group and the page reference is recorded in the log file. The log files are pre-processed to remove noise. Figure 2 depicts a sample log file.



```
10.0.0.27 http://armdl.adobe.com/pub/adobe/reader/win/11.x/11.0.12/misc/AdbeRdrUpd11012.msp 117,885,110
10.0.0.27 http://armdl.adobe.com/pub/adobe/reader/win/10.x/10.1.10/misc/AdbeRdrUpd10110.msp 20,700,612
10.0.0.27 http://b1.download.windowsupdate.com/d/updt/2015/07/10240.16384.150709-1700•h1_clientpro_ret_x86fre_en-us_83d0ecebe
10.0.0.27 http://update.nai.com/products/commonupdater/Current/VSCANDAT1000/DAT/0000/V2datdet.mcs 8,471,248
10.0.0.27 http://thenationonlineng.net/ 1,584,794
10.0.0.27 http://update.nai.com/products/commonupdater/Current/VSCANDAT1000/DAT/0000/V2datinstall.mcs 5,004,055
10.0.0.27 http://www.researchandmarkets.com/Styles/css-core.less 3,639,614
10.0.0.27 http://zealot.com/attachments/imag0059-jpg.143915/ 3,046,492
10.0.0.27 http://manunews.com/ 2,869,331
10.0.0.27 http://thenationonlineng.net/ 1,584,794
10.0.0.27 http://coeoju.com/publications/OSER.doc 2,222,728
10.0.0.27 http://zealot.com/attachments/imag0070-jpg.143916/ 2,150,060
10.0.0.27 http://zealot.com/attachments/imag0063-jpg.143914/ 2,095,918
10.0.0.27 http://www.africamasterweb.com/specialdanceAf.html 1,916,803
10.0.0.27 http://00046d-1.l.windowsupdate.com/llnhost_b1.download.windowsupdate.com/d•h1_clientpro_ret_x86fre_en-us_83d0ecebe
10.0.0.27 http://armdl.adobe.com/pub/adobe/reader/win/10.x/10.1.15/misc/AdbeRdrUpd10115.msp 1,709,163
10.0.0.27 http://www.privateproperty.com.ng/ 1,653,041
10.0.0.27 http://thenationonlineng.net/ 1,584,794
10.0.0.27 http://www.google.com/uds/api/search/1.0/56f70d816baa48bdfe9284ebc883ad41/default+en.I.js 1,563,313
10.0.0.27 http://platform.twitter.com/widgets/tweet_button.8d007ddfc184e6776be76fe9e5e52d69.en.html 1,562,620
10.0.0.27 http://www.bing.com/fd/ls/lsp.aspx 1,266,096
10.0.0.27 http://platform.twitter.com/widgets/follow_button.3476cd70032ff6b94ecc9ea63ab78a8b.en.html 1,094,772
10.0.0.27 http://waplog.com/profile/search_friends 1,021,162
10.0.0.27 http://www.gstatic.com/_/apps-viewer/_/js/k=apps-viewer.standalone.en_US._sAxDPRpYQY.O/m=main/rt=j/d=1/rs=AC2dHMKb0
10.0.0.27 http://www.ngrguardiannews.com/ 924,957
10.0.0.27 http://punch.cdn.ng/wp-content/uploads/wp-banners/300x250-(The-African-Pride).jpg 910,775
10.0.0.27 http://update.nai.com/products/commonupdater/Current/VSCANDAT1000/DAT/0000/PkgCatalog.z 887,027
10.0.0.27 http://update.nai.com/products/commonupdater/catalog.z 876,037
10.0.0.27 http://www.study.monash/courses/find-a-course 863,757
10.0.0.27 http://www.nigeriamasterweb.com/specialdance.html 816,287
10.0.0.27 http://update.nai.com/products/commonupdater/Current/BOCVSE__1000/DAT/0000/vscan.bof 809,773
10.0.0.27 http://www.study.monash/fees-scholarships/scholarships/find/international-student-scholarships/international-leader
10.0.0.27 http://leadership.ng/cheki-brands.html 805,939
10.0.0.27 http://www.thisdaylive.com/ 788,073
10.0.0.27 http://techloy.com/ 721,617
```

Fig.2. Web access Log File

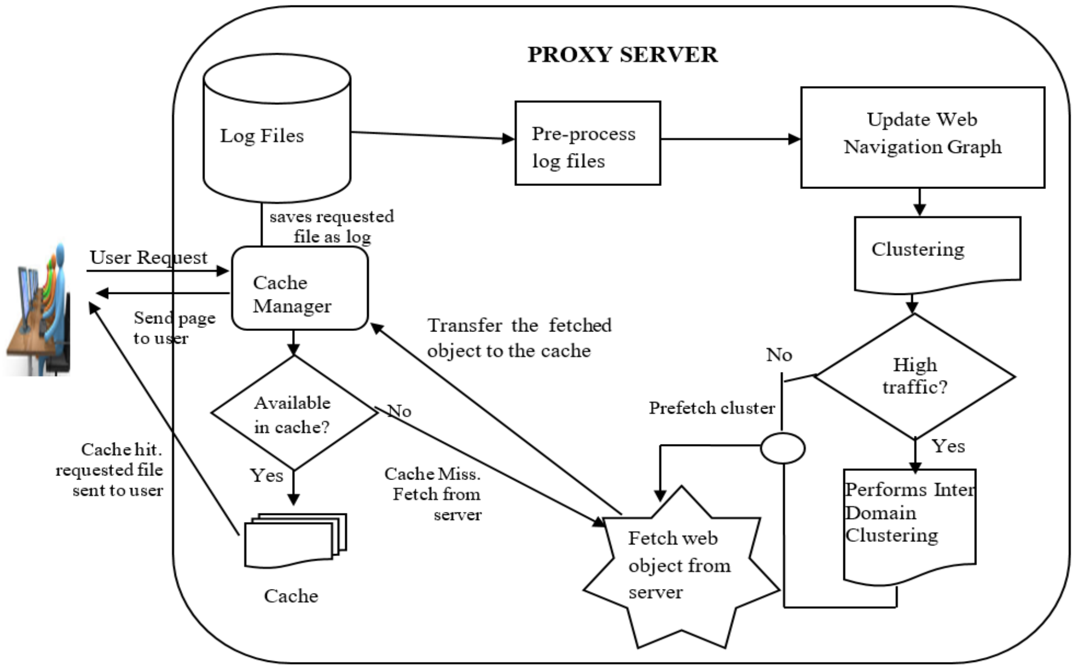

PROXY SERVER
Log Files
Pre-process log files
Update Web Navigation Graph
saves requested file as log
Clustering
User Request
Cache Manager
Send page to user
Transfer the fetched object to the cache
No
High traffic?
Available in cache?
No
Prefetch cluster
Yes
Cache hit. requested file sent to user
Yes
Cache Miss. Fetch from server
Performs Inter Domain Clustering
Fetch web object from server
Cache

Fig.3. Web Prefetching Architecture

The preprocessed log files is used to update the web navigation graph. Support and confidence threshold values are applied on the graph to form several clusters are form in a client group. When the predicted prefetch time is greater than the server idle time, inter domain clustering performed. Figure 3 shows the architecture of the proposed model.

3.1. Web Clustering

The technique used in this paper follows the technique defined in [8]. Requests of each client group are used to define a web navigation graph which is a weighted directed 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.

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 corresponding web navigation graph is as shown in figure 4.

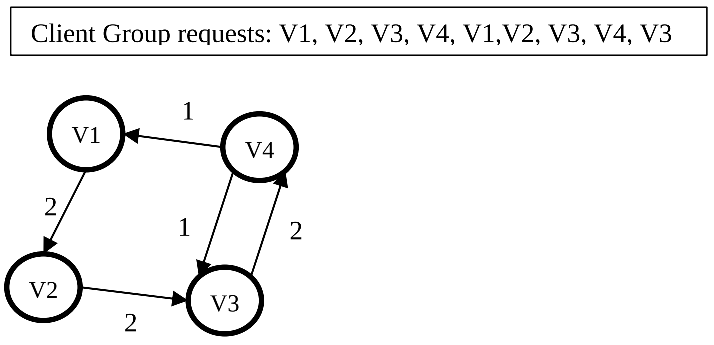


Fig.4. Client Navigation and the Corresponding Web Navigation Graph

Support and confidence threshold values are applied on the navigation graph to remove web pages and edges that have low values. Given two nodes v_i and v_j, Support (v_i, v_j) is the frequency of navigation steps between v_i and v_j. The Confidence (v_i, v_j) is defined as the ratio of the support of v_i and v_j and the popularity of v_i, that is

$$\text{Confidence } (v_i, v_j) = \frac{support(vi,vj)}{pop(vi)} \tag{1}$$

where $pop(V_i)$ is the popularity of v_i. The $pop(V_i)$ denotes the number of request made for a particular page or the number of edges pointing towards a node.

This operation therefore partitions the graph into sub graphs by filtering edges with low support and confidence values. Each sub graph forms a cluster. Thus, each client group has a set of clusters.

For example, if the support and confidence threshold for figure 4 are 1 and 0.6 respectively, the resulting clusters for the client group will be as shown in figure 5.

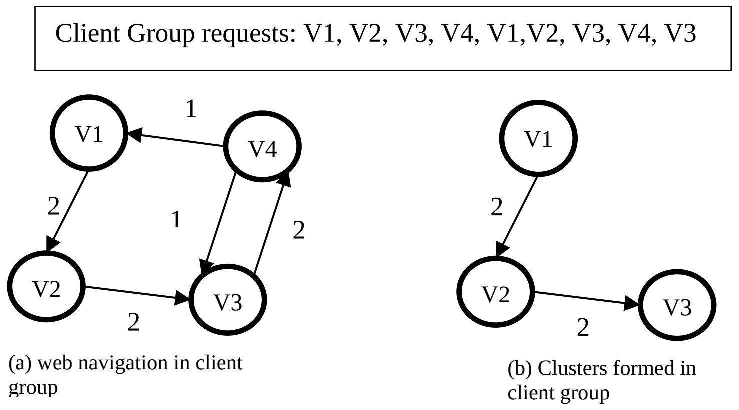

Client Group requests: V1, V2, V3, V4, V1,V2, V3, V4, V3
V1
V2
V3
V4
1
2
1
2
2
(a) web navigation in client group
V1
V2
V3
2
2
(b) Clusters formed in client group

Fig.5. Clustering of Web Object in a Client Group

3.2. Inter Domain Clustering

Prefetching can only be done at CPU idle time and clustering based prefetching predicts users request based on domains. In a situation where there is high traffic, since the server idle time will not be enough to prefetch all user clusters, it is wise to prioritize the prefetching based on the number of users that may likely request the page. This can be achieved by performing an inter domain clustering based on clusters of each domain. High traffic is determined if the frequency of the estimated server idle time is not enough to prefetch all clusters arriving at the server thus necessitate the inter domain cluster. This happens if the ratio of the estimated prefetch time and the server idle time is greater than zero. It should be noted that, if the estimated prefetch time is less than the server idle time, the clusters are prefetched in the classical way.

Let P_t denotes the estimated prefetch time of the clusters to be prefetched from servers at time t, n denotes the number of clusters in a domain, k denotes the number of pages in a cluster, S_{ij} denotes the size of pages(j) in a cluster(i) and B denotes bandwidth. Then, p_t can be estimated as follow.

$$P_t = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k} S_{ij}}{B} \tag{2}$$

Equation 2 assumes that the prefetch time depend on the page size and bandwidth only. In a real scenario, the exact prefetch time may be difficult to evaluate since there are lots of factors to consider including the location of the requested pages, the server speed, the network strength etc.

For example, assuming a cluster containing three web pages of sizes 5 kilobyte, 10 kilobyte and 15 kilobyte to be prefetched by high speed system over a network on a bandwidth of 256 kb/s. The estimated prefetch time will be

$$P_t = \frac{5+10+15}{256} = 0.1172 \text{ seconds.}$$

When the estimated prefetch time is greater than the idle time, the cluster from each domain will be used to form prioritized inter domain clusters. The inter domain cluster with the highest priority will be prefetched first subsequently others will follow based on the frequency of the server idle time. Figure 6 shows prefetching in high traffic environment architecture where s_t denote the predicted server idle time and p_t the estimated prefetching time.

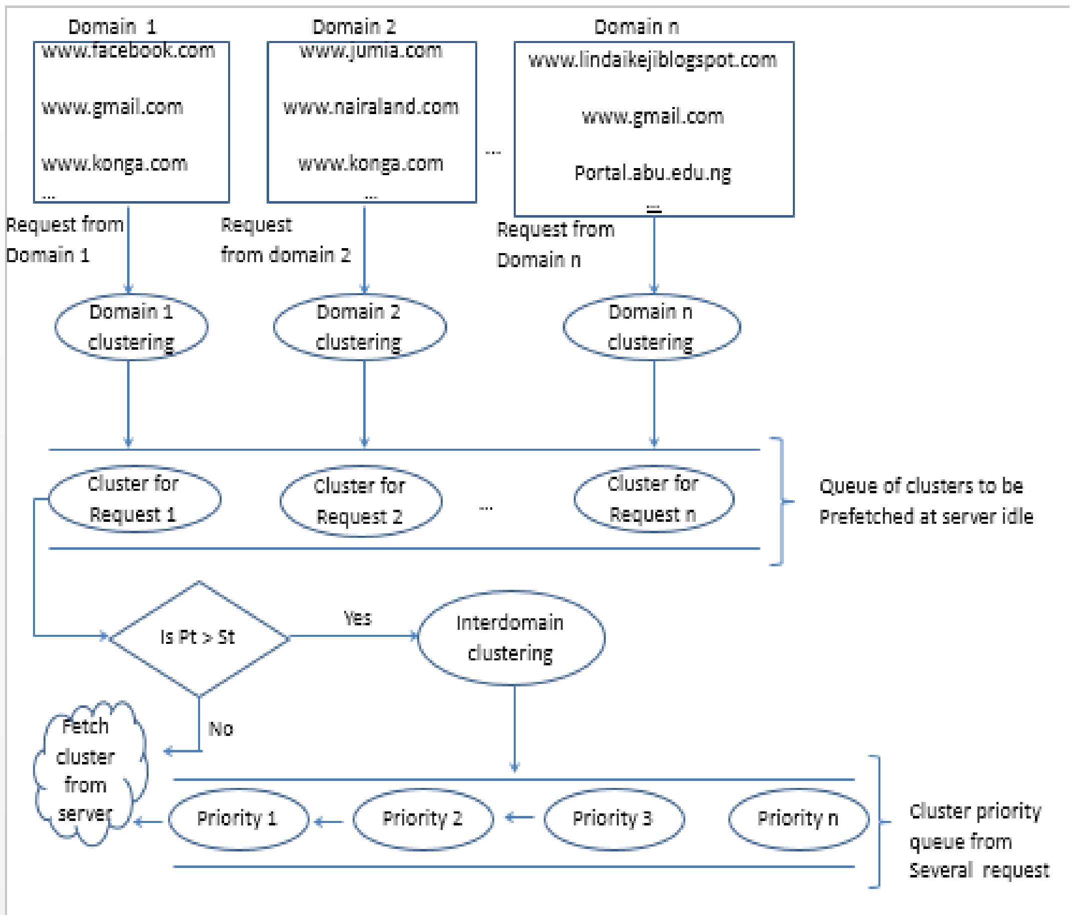


Fig.6. Prefetching in High Traffic Environment Architecture

When $P_t > S_t$, the server performs inter domain cluster. It can be observed that as the server idle time increases, the classical prefetching architecture shown in figure 3 is recovered since inter-domain clustering will no longer be performed.

Our approach for forming the prioritized clusters is based on the popularity of each page as well as the speed at which request are received by the proxy server. Recall that the popularity denotes the number of requests made for a particular page, which can be determined by the number of edges pointing towards a node in a cluster.

Let Q_i denotes the cluster formed from domain *i* and α_i the speed in term of average number of requests per second recently being received from domain *i* by the proxy server. It is obvious that the rate at which requests are received from a domain can influence how quick the page may be requested from that domain. The rate of requests is therefore used as a weight of the cluster to define a weighted cluster $Q_i' = \alpha_i \times Q_i$ such that, the weight of each edge in cluster Q_i' is obtained by multiplying its value on cluster Q_i by α_i. The obtained popularity can now be used to form the new priority clusters in such a way that pages from all weighted clusters with same popularity belong to same priority cluster and if a page belongs to more than one cluster then its priority is the sum of all its popularities from all the weighted clusters in which it appears. The following algorithm states our proposed technique.

Algorithm 1: Inter-domain web Clustering Algorithm

```
Algorithm prefetchCluster(Q[$q_1$,$q_2$,…,$q_n$], serveridletime, bandwidth)
Input: Q[$q_1$,$q_2$…,$q_n$]  queue of web clusters to be prefetched. Each cluster $q_i$ consists of set of pages $p_i$, the speed $\alpha_i$ at which requests are
          received. Each page $p_i$ consist of  its size and popularity
          Serveridletime: estimated time interval for which the server is idle
          Bandwidth: average rate at with pages are prefetched by the server
Output: prefetched web object
priorityQueue[$q_1$,…,$q_m$]: queue  of priority clusters to be prefetched

While ! (IsEmpty(Q))
          Prefetchtime ← $\frac{\sum_{i=1}^{n}\sum_{j=1}^{k} Q_i p_j.zize}{Bandwidth}$
          If(prefetchtime<=severidletime)
                    q ← Q.dequeue()
                    While ! (IsEmpty(q))
                              While !(serverIsBusy)
                                        foreach p in q
                                                  Prefetch (p)
                                                  q.remove(p)
          else
             PriorityQueue = null
             Temp= null
             While !(IsEmpty(Q))
                    q←Q.dequeue()
                    $\alpha$ ← q.speed
                    for each p in q
                       p.priority ← $\alpha$ × p.priority
                       If p IsInArray(temp) Then
                              temp[indexOf(p)].priority←temp[indexOf(p)].priority  +p.priority
                       else
                              temp [] ← p
           temp.sort(DESC)
           curentpriority  ← temp[i].prirority
           cluster[] = null
          curentpriority← temp[i].prirority
             for i=0 to temp.length-1 do
                     if temp[i].prirority==curentpriority
                              cluster []←temp[i]
                    else
                              PriorityQueue[]  ← cluster
                              cluster []= null
                              cluster [] ←temp[i]

            while ! (IsEmpty(PriorityQueue))do
                 while!(ServerIsBusy) do
                        q←PriorityQueue.deque()
                   foreach p in q
                              Prefetch( p)
                              q.remove(p)
```

4. Performance Evaluation

The dataset used for the experiment was obtained from an educational institution squid proxy server. Log files for four weeks were captured. 70% of all the preprocessed log files were used for the user's access pattern analysis, creating training dataset. The remaining 30% of the requests were used for testing the proposed scheme. The proposed model was compared with the prefetching technique of [15]. For the experimental purpose, the cache size is assumed to be unlimited, the bandwidth size is assumed to be 512kb per second, the support and confidence threshold value are assumed to be 1 and 0.5 respectively and the server idle time range from 0 second to a time frame where the proposed algorithm and that of [15] prefetch same content.

4.1. Experimental Results

We evaluate the performance of the proposed technique based on the following parameters:

Hit Rate: The hit rate (HR) is the ratio of the number of requests that hit in the cache over the total number of requests.

Byte Rate: Byte Rate (BR) is the ratio of the number of bytes that hit in the proxy cache over the total number of bytes request sent to its clients.

Precision: It is the ratio of the number of correct predictions over the total number predictions. The metric represents the fraction of predicted pages that are actually used.

Usefulness of prediction: It defines the percentage of requested pages that were previously prefetched.

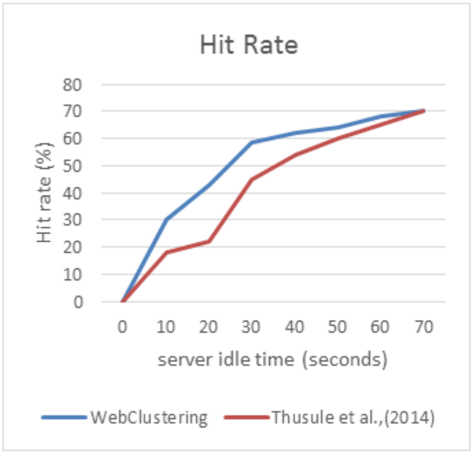


Fig.7. Hit Rate Comparison

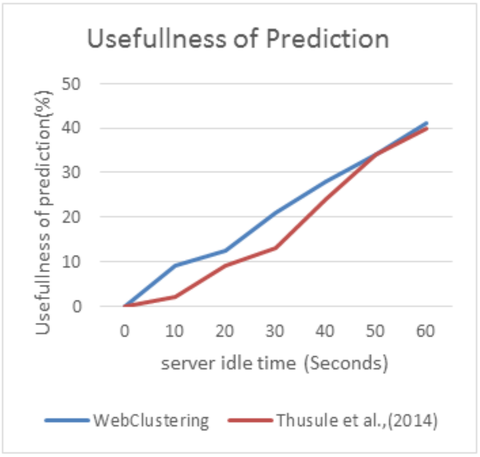


Fig.8. Usefulness of Prediction Comparison

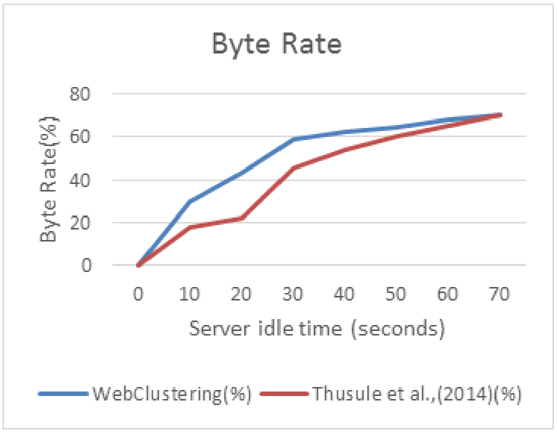


Fig.9. Byte Rate Comparison

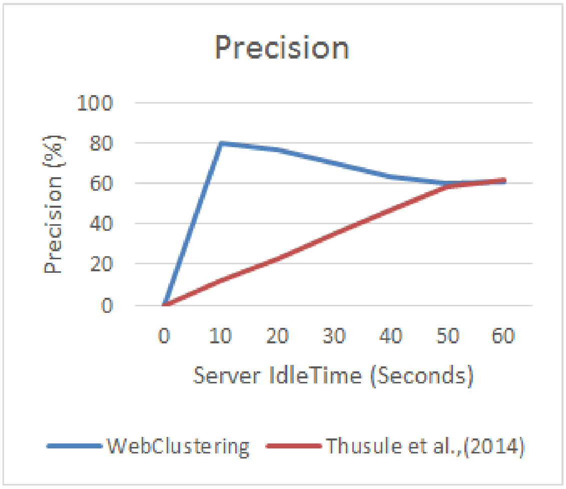


Fig.11. Precision Comparison

From figure 7 to figure 10, it can be observed that, whenever the server idle time is 0, all parameters used for evaluation are also evaluated to 0 since prefetching is only performed if the server is idle. Thus no page is prefetched when the server idle time is 0.

From cache hit rate comparison in figure 7, it can be observed that as the server idle time is increasing, the proposed technique outperforms the approach defined in [15] at a constant rate up to about 30s when the two schemes converge to a maximum value of about 70% when the server idle time is 70s. This value represents the maximum hit ratio for the clustering based prefetching as defined in these two schemes. Similar behaviour is observed for byte rate, precision and usefulness of prediction. It should be noted that as the server idle time becomes large, the two algorithms have the same performance for all the parameters used for the evaluation. This is expected because the two algorithms behave the same since inter domain clustering is no longer performed.

It is important to note the extra complexity added in performing the inter-domain clustering of O(|K| .|U|) in the worst case where |K| is the number of client groups and |V| the number of edges of the cluster having the highest number of nodes. It was shown in [9] that the time complexity of web based clustering is affected by the Breath First Search (BFS) complexity which is O(|U| + |V|) where |U| is the number of nodes and |V| the number of edges in the graph. They showed that in the worst case BFS would be repeated K|U| times, where K is the number of client groups. This shows that the cost of performing the web clustering is higher over the cost of performing inter-domain clustering. Thus, our proposed enhancement does not affect the overall efficiency of the clustering based prefetching algorithm.

5. Conclusion

This paper has presented an approach of short time prefetching in high traffic thereby enabling prefetching at the minimum server idle time. Clusters from several domains are used to form prioritized inter domain clusters and pages are prefetched based on their priority level. The results obtained showed that the proposed algorithm performs better than the classical clustering based prefetching described in [14] whenever the traffic is high and the bandwidth is limited. Future work will look into how to incorporate other parameters that affect the speed of fetching a web page including the processor speed, the file location etc. in our model and compare its performance with several prefetching techniques.

References

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

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

[3] Bhaskaran, V., and Murali, V, "Optimizing the Web Cache Performance by Clustering based Pre-Fetching Technique using Modified ART1", International Journal of Computer Applications, Vol. 4 Issue 1 , 50-57, 2012.

[4] Bina, B., Goudar, R. and Kaushal, K. "Quine-McCluskey: A Novel Concept for Mining the Frequency Patterns from Web data", International Journalfor Education and Management engineering, Vol. 2018 No.1, 40-47, 2018.

[5] Cheng-Zhong, X., and Tamer I., "Semantics-Based Personalized Prefetching to Improve Web Performance", Proc. of the 20th IEEE Conf. on Distributed Computing Systems, 636-643, 2000.

[6] Greeshma, G., and Jayasudha, J., "A Survey on Web Prefetching and Web Caching in a Mobile Environment", International Journal of Computer Science and Information Technology, Vol. 2, Issue 1 , pp. 119-136, 2012.

[7] Neha, K., Esha, D., and Neha, S., "Proposed Framework for the Reduction of Web Congestion using Classification", International Journal of Engineering and Advanced Technology, Vol. 3, Issue 2 , pp. 123-128, 2013.

[8] Neha, S., and Sanjay K., "Fuzzy C-means Clustering Based Prefetching to Reduce Web Traffic", International Journal of Advances in Engineering & Technology, Vol. 6, Issue 1 , pp. 426-435, 2013.

[9] Pallis, G., Athena, V., and Jaroslav, P., "A Clustering-Based Prefetching Scheme on a Web Cache Environment", Computers and Electrical Engineering, Vol. 34, Issue 2008, pp. 309-323, 2007.

[10] Patil, J., and Pawar, B., "Integrating Intelligent Predictive Caching and Static Prefetching in Web Proxy Servers", International Journal on Computer Science and Engineering, Vol.3, Issue 2,pp. 697-704, 2011.

[11] Ramu, k., Sugumar, R., and Shanmugasundaram, B, "A Study on Web Prefetching Technique", Journal of Advances in Computational Research, Vol. 1,Issue 1,pp. 39-46, 2012.

[12] Sachan, D., and Rao, D., "Performance Improvement of Web Caching Page Replacement Algorithms",

International Journal of Computer Science and Information Technologies, Vol. 5,Issue 3,pp. 3112-3115, 2014.

[13] Sathiyamoorthi, V., and Ramya, P., “Enhancing Proxy Based Web Caching System Using Clustering Based Pre-fetching with Machine Learning Technique”, International Journal of Research in Engineering and Technology, Vol. 3,Issue 7, pp. 463-469, 2014.

[14] Temgire, S., and Poonam, G., “Review on Web Prefetching Techniques”, International Journal of Technology enhancements and emerging engineering research,Vol. 1,No. 4 ,pp. 100-105, 2013.

[15] Thulase, M., and Raju, G., “Effective Web Access Latency Reduction Through Clustering Prefetching and Caching”, International Journal of Electrical & Computer Sciences, Vol. 14, No. 5, pp. 7-12, 2014.

Authors’ Profiles

Atta Fatima Oheza received B.Sc. and M.Sc degree in Computer Science from Ahmadu Bello University Zaria, Nigeria. She is currently a Ph.D student in Computer Science at the Department of Computer Science, Ahmadu Bello University, Zaria, Nigeria. Her current research interests include machine learning, data mining and knowledge representation and reasoning.

Armand F. Donfack Kana received the B.Sc. degree in Computer Science from University of Ilorin, Nigeria, M.Sc and Ph.D. degrees in Computer Science from University of Ibadan, Nigeria. He is currently a faculty member at the Department of Mathematics, Ahmadu Bello University, Zaria, Nigeria. His current research interests include Knowledge representation and reasoning, Formal Ontologies and Soft computing.

How to cite this paper: F. O. Atta, A. F. Donfack Kana,"Web Clustering based Prefetching in High Traffic Environment", International Journal of Education and Management Engineering(IJEME), Vol.8, No.3, pp.21-32, 2018.DOI: 10.5815/ijeme.2018.03.03