Web caching and prefetching with cyclic model analysis of web object sequences

K.C. Srikantaiah*, N. Krishna Kumar and K.R. Venugopal

Department of Computer Science and Engineering,
University Visvesvaraya College of Engineering,
Bangalore University, Bangalore, India
E-mail: srikantaiahkc@gmail.com
E-mail: krishnakumarn@live.com
E-mail: venugopalkr@gmail.com
*Corresponding author

L.M. Patnaik

Indian Institute of Science,
Bangalore, India
E-mail: patnaiklm@yahoo.com

Abstract: Web caching is the process in which web objects are temporarily stored to reduce bandwidth consumption, server load and latency. Web prefetching is the process of fetching web objects from the server before they are actually requested by the client. Integration of caching and prefetching can be very beneficial as the two techniques can support each other. By implementing this integrated scheme in a client-side proxy, the perceived latency can be reduced for not one but many users. In this paper, we propose a new integrated caching and prefetching policy called the WCP-CMA which makes use of a profit-driven caching policy that takes into account the periodicity and cyclic behaviour of the web access sequences for deriving prefetching rules. Our experimental results have shown a 10%–15% increase in the hit ratios of the cached objects and 5%–10% decrease in delay compared to the existing scheme.

Keywords: periodicity; sequential pattern analysis; web caching; web log; web prefetching.

Reference to this paper should be made as follows: Srikantaiah, K.C., Kumar, N.K., Venugopal, K.R. and Patnaik, L.M. (2014) 'Web caching and prefetching with cyclic model analysis of web object sequences', *Int. J. Knowledge and Web Intelligence*, Vol. 5, No. 1, pp.76–103.

Biographical notes: K.C. Srikantaiah is an Associate Professor in the Department of Computer Science and Engineering at S.J.B. Institute of Technology, Bangalore, India. He obtained his BE, ME and PhD in Computer Science and Engineering from Bangalore University, Bangalore. His research interests are in the areas of data mining, web mining and semantic web.

N. Krishna Kumar received his BE in Information Science and Engineering and MTech in Computer Engineering from Visvesvaraya Technological University, Belgaum, India, in the years 2010 and 2012, respectively. His research interests include web mining, web technologies and computer networks.

K.R. Venugopal is currently the Principal of the University Visvesvaraya College of Engineering, Bangalore University, Bangalore. He obtained his Bachelor of Engineering from University Visvesvaraya College of Engineering. He received his Masters in Computer Science and Automation from Indian Institute of Science Bangalore. He was awarded his PhD in Economics from Bangalore University and PhD in Computer Science from Indian Institute of Technology, Madras. He has authored and edited 40 books. During his three decades of service at UVCE he has over 400 research papers to his credit. His research interests include computer networks, wireless sensor networks, parallel and distributed systems, digital signal processing and data mining.

L.M. Patnaik is currently Honorary Professor, Indian Institute of Science, Bangalore. He was a Vice Chancellor, Defense Institute of Advanced Technology, Pune, India. He was a Professor since 1986 with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. During the past 35 years of his service at the institute, he has over 700 research publications in refereed international journals and refereed international conference proceedings. He is a fellow of all the four leading science and engineering academies in India. His areas of research interest have been parallel and distributed computing, mobile computing, CAD for VLSI circuits, soft computing and computational neuroscience.

This paper is a revised and expanded version of a paper entitled 'An integration of web caching and web prefetching using cyclic behaviour analysis of web sequential patterns' presented at the 2nd International Conference on Computational Intelligence and Information Technology (CIIT 2012), Chennai, India, 3–4 December 2012.

1 Introduction

The internet is an extremely large collection of network of networks, which in turn consist of web servers which contain huge quantities of data, clients or end-user system which request information or services from the servers and finally client-side and server-side proxies which are additional systems that help and provide for better communication among clients and servers.

A web object is any entity that is stored in the web server and requested by the clients (browsers). It can be a web page (such as HTML, XML, etc.), or an image (GIF, JPEG, PNG, etc.) or a multimedia file like audio or a video file. These web objects can contain links among them and are generally referred to as hyperlinks in HTML context.

Web caching is the process in which the web objects are temporarily stored to reduce bandwidth consumption, server load and latency. Caching can be performed at the client by browsers, or by proxies at client-side or proxies at server-side. Prefetching is the process of fetching web objects from the server before they are actually requested by the client which can reduce the user's perceived latency.

Applications of web caching and web prefetching are

1 a search engine may cache a website.
2 browser can prefetch and cache hyperlinked documents based on their hit rate.
3 an image-sharing website may prefetch popular images and cache them in the client
4 websites such as *YouTube* can prefetch and cache videos based on user's viewing history.

Prefetching can be achieved through three approaches such as path traversal techniques, probability techniques, display-based prediction prefetching. Web caching is achieved through the techniques

1 prediction-based buffer manager
2 proactive caching
3 profit-driven cache replacement
4 cache coherence
5 hierarchical caching.

This paper integrates web caching and web prefetching in a client-side proxy system with the application of periodicity and cyclic behaviour of 2-sequence web access patterns for finding out the prefetching rules.

1.1 Motivation

The internet faces many challenges regarding the quality of service (QoS) provided to the clients with respect to parameters such as efficient bandwidth usage, latency, hit rate etc. There are many solutions devised over the years to overcome these challenges and web caching and web prefetching are two such methods. The profit function employed by integration of web caching and prefetching (IWCP) (Teng et al., 2005) makes use of the confidence value of the prefetching rules. It completely disregards the possibility that objects with very high profit values may remain in the proxy's cache indefinitely.

1.2 Contribution

In this paper, we propose a new approach web caching and prefetching with cyclic model analysis of web object sequences (WCP-CMA) which uses the periodicity and cyclic behaviour of 2-sequence web access patterns instead of confidence for finding out the prefetching rules to alleviate the disadvantage of IWCP. To find out the periodicity and cyclic behaviour, first we take the web logs in the server into consideration. Preprocessing is performed on these web logs to identify users and sessions and generate appropriate access sequences for each session of every user. Next, we perform sequential pattern mining by employing the method of up down directed acyclic graph approach (Chen, 2010). After generating the sequential patterns we generate the prefetching rules using periodicity and cyclic behaviour (Chiang et al., 2009) of each web object pertaining to 2-sequences. We propose a new profit function which includes the periodicity, cyclic behaviour and the frequency of a web object.

1.3 Organisation

The remainder of the paper is organised as follows. In the next section the related works and existing techniques are discussed briefly. In Section 3, we explain the background of this paper. Section 4 contains assumptions and basic definitions. In Section 5, we derive the mathematical model and the WCP-CMA algorithm. Section 6 discusses the proposed architecture of the system. Section 7 provides the results the performance of WCP-CMA. Finally, Section 8 contains conclusions.

2 Related work

Several techniques have been proposed for web prefetching and caching. Eden et al. (2000) proposed a user-interactive prefetching scheme where the user is in control of the prefetching process and decides what and when to prefetch, thus eliminating extra traffic in the network by perfecting irrelevant data that are not required by the user. This mechanism can be implemented as a part of the browser as a plugin and does not need the involvement of the server or the proxy. This technique results in an excellent hit ratio as the user chooses what to prefetch thus reducing latency by more than 75%. But, some browsers may not support the plugin.

Kim and Kim (2003) implemented a novel mechanism of display-based prefetching. First, the users' request patterns are analysed by means of a link graph. The nodes in this graph are the documents and the links in this graph are the hyperlinks among them. The root node of the graph is the first page or the home page to be accessed by the user. Second, prediction and prefetching takes place. Prediction is done by examining the access rates of all the documents displayed in the browser. Prefetching is done by ordering the access rates of all the predicted objects in decreasing order and these objects are prefetched and cached only if the duration of reading of current page by the user exceeds a certain time limit. Here the prefetching effect is almost four times higher than conventional schemes. However, the high network overload should also be taken into consideration.

Feng and Vij (2008) designed a prefetching scheme which makes use of a multi-dimensional probability matrix to compare hit rates and analyse sequence patterns, so that it is possible to prefetch multiple web pages. Matrices are used to store probabilities of individual pages. With matrices, more than two-dimensional probabilities due to more number of multiple pages can also be recorded by constructing a multi-dimensional matrix of that order. If there are n pages stored at the server, a 3D page probability matrix of size ($n \times n \times n$) is maintained to store the probabilities. The cell [ijk] stores the probability of requesting page k after page i and page j have been requested in that order. Similarly, a 4D pre-fetching allows predicting based on four-page sequences which are based on the probability of requesting page l after page i, page j and page k have been requested. Multiple web pages can be analysed at the same time resulting in 10%–50% improvement in hit ratios. But the increase in matrix dimensionality may cause processing delay.

Foygel and Strelow (2000) developed a prefetching mechanism which makes use of the cache and its ancestors. This type of Hierarchical cache is observed by the prefetching algorithm and future requests are predicted such that the prefetched objects significantly mask the latency for previously unseen, but correctly predicted objects. The algorithm works as follows: First, a local prediction module executes separately within each cache and estimates, for every pair of documents whether one document is likely to be requested soon given that the other document is the most recently requested document. Next, the algorithm communicates predictions between caches when a child cache does not have enough access information to make informed predictions. The statistics of the different levels of hierarchy of the cache are combined to produce the final predictions for prefetching. Finally, it uses a formal cost-benefit model that will judge the value of any cached or prefetched document so that prefetching all the predicted requests can be avoided. Here, hit ratios increase with the increase in cache size but, scanning the cache hierarchy may increase overhead.

Klemm (1999) designed and implemented a client-side prefetching tool called *WebCompanion*. This component is installed in the client machine and directly interacts with the web browser to perform prefetching. *WebCompanion* utilises a novel round-trip time based prefetching mechanism. In this method, the hyperlinks which are embedded in a current document are examined by the tool. If the predicted documents are not stored in the cache, it checks its server statistics to see whether the user has already visited the server or not. If there is no such entry, then the tool accesses the server, fetches the documents and stores them in the cache and makes a new entry in its server statistics. This significantly reduces the round-trip delay required for accessing each of the individual hyperlinked documents in the current web page. Also the prefetched documents can directly be accessed from the local cache without accessing them from the web server. To reduce network congestion and delay, all the objects are not prefetched, but only those that have a HTTP method of request are prefetched. So the tool does not work non-HTTP and dynamic web documents such as scripts.

Jeon et al. (2003) proposed adaptive prefetching where the web documents of websites are not only modified or updated, but also their access rates as well. The next requested document is mostly based on the previously accessed documents. The *importance* (access rate) of the documents is measured using a hit counter. When a document is accessed by a user, all the related *popular* documents are prefetched and cached with respect to a certain threshold value, which is based on the hit ratios of the documents. The hit rates of the documents to be prefetched must be greater than the threshold. Adaptive prefetching overcomes the problems faced due to modification of web logs and decreases response time by 20% compared to conventional schemes. In this approach, calculation of the threshold value whether to cache or not must be more refined.

Pallis et al. (2008) considered a clustering-based prefetching scheme where a graph-based clustering algorithm identifies clusters of *correlated* web pages based on the users' access patterns and integrates web caching and prefetching. Each time a user requests a web object, the algorithm prefetches all the objects which are in the same cluster with the requested object. Specifically, the traces are represented by a web navigational graph. Then, the clusters are formed by partitioning this graph, where the number of clusters is not determined *apriori* but is dynamically estimated by the confidence and support measures. Here, it is possible that web objects in a cluster with no chance of being accessed might also be prefetched thus decreasing hit ratios of other useful objects.

Yang et al. (2001) proposed the extensive use of mining of the web logs for construction of association rules for integrated caching and prefetching. The links between documents are considered for the construction of the association rules. Association rules help in the forming of the prefetching rules. The documents in the web and the hyperlinks between them are in the form of a directed graph where the nodes are the documents and the directed arcs are the hyperlinks from one document to the other. Frequent sequences in the graph are observed to generate prediction rules based on sub graphs of order *n* from the graph. These rules are applied for prefetching and caching a web object when a document is accessed by the user. Although this method results in a high hit ratio, it does not work for dynamic web documents such as scripts.

Li et al. (2001) try to combine cache coherence and prefetching to decrease both the number of requests to the server and response latency, and save bandwidth. Cache coherence is the process where the stale and old documents stored in the cache are updated. The common aspect between cache coherence and prefetching is: both of these mechanisms need the cooperation of the proxy and the web servers. The difference between these mechanisms is: the documents stored in cache may become stale over time and need to be updated whereas the documents that are prefetched are always fresh and the client may request access to these prefetched documents. By combining these two mechanisms, both the number of requests to the server and response latency can be decreased. Also, bandwidth is saved by the quantity of bytes that had to be transmitted for original request. However, frequent updates may render prefetching useless.

Balamash and Krunz (2004) proposed a client-side caching policy called proactive caching where the dynamic nature of the web documents is countered by splitting the cache into two parts: the regular cache, which performs caching on-demand, i.e., documents that are cached are those that are directly requested by the user, and the prefetching cache, that is used to cache prefetched documents. The space allotted for prefetching is less than that of regular caching. If the user demands a prefetched document, then that document is simply moved from the prefetching cache to the regular cache, and new documents, that are related to this document, and prefetched. The prefetching mechanism sets a threshold value on the number of documents that can be prefetched to decrease network load and response time. However, the threshold value can limit the number of objects that can be prefetched which can prevent many useful objects from being cached.

Huang and Hsu (2005) developed two techniques to improve the performance of caching at the proxy servers. The access sequence miner is used for efficient mining of the web logs for predicting the user's behaviour for improving hit ratios of the documents. The prediction-based buffer manager is used to prefetch and cache and also dynamically adjust cache and prefetch buffer sizes for efficient utilisation of the available memory space. Here, two separate buffers are used for caching and prefetching. The access sequence miner scans the filtered records and if it finds a 2-sequence, then that sequence is used to formulate the prefetching rules. After all sequences are analysed, then the rule table is generated by the access sequence miner which consists of all the prediction and perfecting rules. Finally, the prediction-based buffer manager takes over and predicts the next document by using the rule table, prefetches and stores those documents in the proxy's buffer. These techniques result in effective document prefetching with efficient buffer size management, but, buffer management can cause additional overhead.

Shi et al. (2005) propose a new model for integration of web caching and web prefetching called the stochastic petri nets (SPN) model in a server-side proxy. Caching and prefetching are classified into two different categories: passive caching, where only caching takes place, i.e., the accessed documents are stored in the cache for future reference, and local prefetching, where objects are prefetched and cached based only upon the local information available in the client. The SPN model of integrated caching and prefetching is designed as follows: the system is modelled as a multi-user system consisting of a single web server. The users request objects from this server and share the same network link. These requests are handled by the proxy at the server which classifies them into two types: normal and prefetching requests. If the objects are not in the local cache of the client, the proxy issues normal requests to the remote web server. But, if the objects are present in the local cache, then proxy issues prefetching requests to the remote web server for obtaining the predictive objects from it. These prefetched objects are then passed on to the client to store its local cache. SPN model can support multiple users with only a single server proxy. The disadvantage is that the number of prefetching objects decreases as prefetching increases.

Jin et al. (2007) proposed three techniques: a prediction mechanism called sequence-mining based prediction, a prefetching mechanism called context-aware prefetching and a caching mechanism called Profit-driven cache replacement policy, for use in a wireless LAN environment. These techniques mainly deal with mobile web caching systems in a wireless network system. The prediction algorithm scans the web log within the range of the access sequence and extracts all items whose support is larger than the minimum specified support, i.e., hit rate ratio. Context-aware prefetching is mostly based on the quantity to be prefetched in the mobile client. Prefetching is integrated with caching by analysing the profit gained by caching an object. The profit is evaluated based on the ability of the cached item to introduce more prefetching candidates. This results in improved response time in a wireless network. The power of the device is saved by 5%, but, mobility of the node is not taken into consideration.

Songwattana (2008) proposed a novel web log mining technique called forward/backward scanning for the prediction of the next document to be prefetched and cached. The model consists of a log analyser for identifying and extracting usage patterns of client requests along with additional information such as number of hits and size of document. The model also consists of the mining module which utilises the forward/backward scanning technique to perform mining on the log after the analyser has completed its operation. In this technique, with reference to a given URL in the log, in a given time interval, the possibility of the future occurrence of the same URL in the web log is predicted by utilising the data of the previous occurrences of the same URL in the web log. The prefetching module utilises the information provided by the mining module for prefetching and caching the documents of the specified URL. Forward/backward scanning does not focus on characteristics of workload and web access patterns.

Marquez et al. (2009) developed a simulation framework for caching and prefetching which is mainly focused upon reducing the latency experienced by the user in acquiring a fully loaded after the user's request. Latency is divided into two broad categories: Internal latency, which is very small, is the latency experienced due to the internal processing in the client machine, and external latency, which is observed in the external network that connects the client to the web server. The prefetching mechanism has two components: the prediction engine and the prefetch Engine. The prediction engine predicts the user's next request and provides these predictions as hints to the prefetch engine. The prefetch engine handles the hints and decides whether to prefetch them or not depending on conditions like available bandwidth or idle time. Although, this framework is very flexible with different web architectures, the prefetching depends highly on available bandwidth.

Nigam and Jain (2010) analysed the use of the Markov model on web caching and prefetching and introduced a mechanism called the dynamically nested Markov model (DNMM). A Markov model is the model which is used for the modelling of the web navigation sessions to decrease latency and improve QoS, where each session is a state diagram and the web pages are states and the links are the transmissions among them. DNMM uses a tree-like structure for linking the nodes which are web pages and the sub-nodes are connected to their nodes by individual links (hyperlinks). Also in DNMM, there are two types of caches called regular cache, which is used for normal caching of the accessed web pages and prefetch cache, which is used for caching prefetched content. If the predicted and requested documents are same and found in the regular cache itself, it will be provided to the client from the cache, otherwise the client will request the document from the web server. DNMM results in improved QoS with minimum latency and retrieval time of web pages is reduced by 25%. The drawback is that Markov models are not known for accuracy.

3 Background

Teng et al. (2005) have proposed an algorithm IWCP which integrates web caching and prefetching in the client-side proxy by taking the confidence value of web access sequences into consideration. Only that sequence whose confidence exceeds a certain threshold value is selected as a prefetching rule. IWCP also makes use of a profit function which computes a profit value for each web object based on the mean reference rate, the time-window in which the object is accessed and the confidence of the rule that was triggered due to that object. The higher the profit, the more is the benefit of caching that object.

While such an approach is generally beneficial, it subsequently fails when an object with a very large profit value is placed in the cache. This object will occupy space and will not be removed from the cache until another object of a higher profit value occurs. Furthermore, when this object is prefetched and cached and is accessed by the clients a certain number of times, there is the great possibility that it will never be accessed again immediately after these accesses and also, the cache size in the proxy is very small. Thus, caching such objects is not only a waste of space during that time, but they also prevent other useful objects from being cached as they may have slightly lower profit values than that of this object.

In our approach, we propose WCP-CMA to utilise the periodicity and cyclic behaviour of the web object sequences as prefetching rules instead of using their confidence value. Hence, we remove the objects that are placed in cache immediately after they have reached the threshold of their cyclic behaviour, i.e., they have been accessed the maximum possible number of times and will most likely not be accessed again in the future. The hit ratio increases by 10%–15% and the delay decreases by 5%–10%. Hence, WCP-CMA is better than IWCP.

4 Problem statement

4.1 Problem definition

Given the web log W and the set of web objects $O = \{o_i: 1 <= i <= n\}$ in a web server, which consists of clients' information and fields such as IP address of the client, URL, date and time, method, protocol, referrer, etc., we first mine the prefetching rules from W. First of all, preprocessing must be performed on W. This is achieved by ordering the records in W according to users and identifying the sessions for each user and finally finding the access sequences for each user in different sessions.

Then, sequential patterns must be generated from the cleaned W to find out the web access patterns of the web objects. Using these patterns, the periodicity P and cyclic behaviour C for each pair of web objects $o_i, o_j \in O$ can be mined using an efficient sequential pattern analysis algorithm.

Our objective is to design an efficient algorithm WCP-CMA that integrates web caching and web prefetching in a client-side proxy system and utilises the periodicity and cyclic behaviour of the 2-sequence patterns to determine when the object should be placed or removed from cache to improve hit rate and increase spatial locality. The notations used in our model are as shown in Table 1.

Table 1 List of symbols used

Symbol	*Description*
o_i	The web object i
δ_i	The mean reference rate of o_i
w	Time window in which o_i is referenced
P_i	The periodicity of $o_h \rightarrow o_i$
C_i	The cyclic behaviour of $o_h \rightarrow o_i$
F_i	Frequency of access of o_i

4.2 Assumptions

The reference rate δ_i of object o_i is the rate of reference of a web object within a given time window w. In our simulation, the time window is assumed to be constant for all the web objects. The reference rate δ_i is exponentially distributed and can be determined using the mean function of exponential distribution. Furthermore, we assume that every request to the server causes a delay of 10 ms and every request to the proxy causes a delay of 5 ms.

4.3 Basic definitions

- *Implied object:* A web object o_j is said to be an implied object whenever it is referenced through a link from the web object o_i and the 2-sequence (o_i, o_j) should be frequent.
- *Periodicity*: The periodicity P between a pair of web objects (o_i, o_j) is defined as the time period t after which o_j shall be accessed periodically after o_i has been accessed. Example: The periodicity P between a pair of web objects (o_i, o_j) is 6 ms indicates

that the object o_j is accessed periodically with period 6 ms after object o_i has been accessed.

- *Tendency:* The tendency between a pair of web objects (o_i, o_j) is defined as the line in the trend graph that determines whether the cyclic behaviour of the rule $o_i \rightarrow o_j$ is increasing or decreasing.
- *Cyclic behaviour:* The cyclic behaviour C between a pair of web objects (o_i, o_j) is defined as the time that gives the stopping criteria for accessing object o_j after o_i has been accessed. It is calculated using periodicity and tendency. The periodicity is increased by adding the value to itself each time the object is accessed. The object will not be accessed after periodicity has reached the limit of cyclic behaviour, i.e., $P \leq C$. Example: The periodicity P between a pair of web objects (o_i, o_j) is 6 ms, the trend line is decreasing and the cyclic behaviour C = 100 ms indicates that accessing of object o_j repeats for every 6 ms after accessing the object o_i. This ends when the time *t reaches* 100 ms.

5 Mathematical model

5.1 Finding prefetching rules using periodicity

Consider the web log W and a database SP consisting of 2-sequence patterns which are mined from W. Consider (o_i, o_j) as a 2-sequence pattern in SP where object o_i is accessed at time T_1 and object o_j is accessed at time T_2 after o_i is accessed (through a link from o_i).The time interval between T_1 and T_2 is x, trend distribution function (TDF) (Chiang et al., 2009) $f(x)$ is used to find the number of occurrences of the 2-sequence at the time interval x.

Periodicity P is determined by a procedure generalised periodicity detection (GPD) function. GPD determines whether a certain sequence (2-sequence in this case of web access sequences) has periodicity and computes the value of the periods if periodicity exists based on regression. The regression model is in the form of $y = ax + b$ where a is the slope of the TDF curve and b is the intercept that are given by equations (1) and (2) respectively.

$$a = \frac{\left(N \times \Sigma xy - (\Sigma x)(\Sigma y)\right)}{\left(N \times \Sigma x^2\right) - (\Sigma x)^2} \tag{1}$$

$$b = \frac{(\Sigma y)\left(\Sigma x^2\right) - (\Sigma x)(\Sigma xy)}{\left(N \times \Sigma x^2\right) - (\Sigma x)^2} \tag{2}$$

where x is the random time variable within the range 1 to N, and y is the TDF at different values of x. GPD function $g(x)$ is given by

$$g(x) = \frac{f(x)}{y} \tag{3}$$

Now the mean error between two values of GPD functions, $g(x_i)$ and $g(x_{i+P}{}^1)$ is given by

$$error\left(P^1\right) = \frac{\Sigma_{i=1}^{n-P^1} \left|g\left(x_{i+P^1}\right) - g\left(x_i\right)\right|}{\Sigma_{i=1}^{n-P^1} \left|g\left(x_i\right)\right|} \tag{4}$$

Using equation (4), we calculate the set of errors for different values of P^1 in the range of 1 to (N/2). Among these errors, the value of P^1 that gives the minimum error is the periodicity P, i.e.,

$$\begin{aligned} &P = \arg\min\left\{Error\left(P^1\right)\right\} \\ &\quad P^1, \quad 1 \le P^1 \le \frac{n}{2} \end{aligned} \tag{5}$$

After determining the periodicity using equation (5), trend modelling (TM) (Chiang et al., 2009) is applied to find out the frequency of the trends using polynomial regression which is of the form:

$$(ax+b) = \sum_{i=0}^{m} z_i (x \bmod P)^i \tag{6}$$

where $Z = \{z_1, z_2, \ldots, z_m\}$ is the set of coefficients of polynomial regression which can be found out using

$$Z = \left(X^T X\right)^{-1} X^T Y \tag{7}$$

where X^T represents the transpose of the matrix X, and X^{-1} represents inverse of matrix X.

Finally, the cyclic behaviour which gives the stopping criteria C is determined using TM and GPD recursively to get periodicity and a polynomial. After analysing the polynomial, we get the stopping criteria. The rules for prefetching the web object o_j in the sequence ($o_i \rightarrow o_j$) can be found out using the periodicity of this sequence.

5.2 *Profit function*

Consider reference rate δ_i of object o_i is the rate of reference of a web object within a given time window w. Reference rate is an important factor in determining the profit of caching a web object and follows an exponential distribution. The number of references within a time window w follows Poisson distribution. So, the number of references to object o_i is given by:

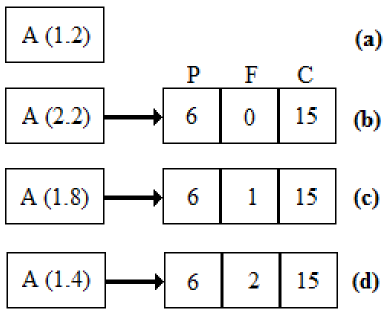
$$\delta_i w = \sum_{x=0}^{\infty} x \times \frac{\left(\delta_i w\right)^x e^{-(\delta_i w)}}{x!} \tag{8}$$

The profit of a web object o_i is dependent on the number of references ($\delta_i w$) to o_i and periodicity P_i and threshold of cyclic behaviour C_i. The parameters $\delta_i w$, P_i and C_i remain constant but the frequency number of access of o_i, i.e., F_i is incremented every time o_i is accessed from the proxy's cache. So, the profit value of o_i depends entirely on F_i. The duration in which the prefetched object o_i must remain in the cache is determined using the profit function given by:

$$Profit(o_i) = \delta_i w + \frac{C_i - (P_i \times F_i)}{C_i} \tag{9}$$

The profit value of o_i is inversely proportional to F_i, i.e., lower the number of access, higher the profit and vice-versa. Consider the example as shown in Figure 1(a). There is a link from a certain object X to object A with prefetching rules $P_A = 6$ and $C_A = 15$. Initially, when the implied object A is placed in a cache, the number of accesses is 0 and profit will be at its highest [Figure 1(b)]. As F_i increases, and as the periodicity value is multiplied to F_i, profit decreases [Figure 1(c)] and finally when P_i reaches C_i, o_i can be deleted from the proxy's cache after F_i reaches 3 as $6 \times 3 = 18$ which is greater than $C_A = 15$ [Figure 1(d)].

Figure 1 Example of profit computation



The profit obtained after placing an object o_j into the cache with sequence ($o_i \rightarrow o_j$) with the rules periodicity P_j and cyclic behaviour C_j can vary according to the values of P_j and C_j. These variations are reflected as shown in the following cases:

- *Case 1 (values of both P_j and C_j are large):* An object with a very large value of P_j is least frequently accessed and occurs only at P_j intervals. Since the value of P_j is large and C_j is also large, the object o_j remains in the cache for a large duration of time. So, it can significantly reduce the hit ratios of other objects. Example: if $P_j = 100$ ms and $C_j = 2{,}000$ ms, the object o_j should be accessed for every 100 ms (i.e., 100th ms after object *oi* has been accessed, 200th ms, …, and so on), and should be present in the cache up to 2,000th ms after object *oi* has been accessed. So, it will not be accessed from 1 to 99th ms and 101 to 199th ms and so on.

- *Case 2 (value of P_j is large and that of C_j is small):* An object with a very large value of P_j is least frequently accessed and occurs only at P_j intervals. Since the value of P_j is large and C_j is small, the object o_j remains in the cache for a very small duration of time. So, it can moderately reduce the hit ratios of other objects. Example: if $P_j = 100$ ms and $C_j = 300$ ms, the object o_j should be accessed for every 100 ms (i.e., 100th ms after object *oi* has been accessed, 200th ms, up to 300th ms). After that, the object o_j will be evicted from the cache.

- *Case 3 (value of P_j is small and that of C_j is large):* An object with a very small value of P_j is most frequently accessed and occurs at frequent P_j intervals. Since the value of P_j is small and C_j is large, the object o_j remains in the cache for a very large duration of time. While it significantly increases the hit ratio of o_j, it significantly reduces the hit ratios of other objects. Example: if P_j = 20 ms and C_j = 2,000 ms, the object o_j should be accessed for every 20 ms (i.e., 20th ms after object o_i has been accessed, 40th ms, and so on up to 2,000th ms). The number of hits for object o_j is very high (100 hits) and that of other objects is lesser.
- *Case 4 (values of both P_j and C_j are small):* An object with a very small value of P_j is most frequently accessed and occurs at frequent P_j intervals. Since the value of P_j is small and C_j is also small, the object o_j remains in the cache for a short period of time. After C_j has been reached, the object o_j is removed from the cache making room for other objects to be cached. So, this case is the most efficient of the four. Example: if P_j = 20 ms and C_j = 200 ms, the object o_j should be accessed for every 20 ms (i.e., 20th ms after object o_i has been accessed, 40th ms, …, and so on), and should be present in the cache only up to 200th ms after object o_i has been accessed.

5.3 WCP-CMA algorithm

In this paper we take only Case 4 into consideration as it is the most efficient prospect compared to the other three cases. WCP-CMA runs in the client-side proxy and is invoked every time a link is clicked in the browser (client). The cache linked list *L* is used to track the status of the cache. The web log *W*, the current web object, the prefetching rule table for all web object links are all inputs to WCP-CMA. This algorithm makes use of another list called as the Hinted objects list *H* which is used to store the details such as P_i and C_i of each link of o_i to be prefetched and cached. The size of the proxy's cache (i.e., the maximum number of web objects that the cache can store) is pre-specified and remains constant. The prefetching rule generating algorithm is shown in Table 2 whereas the WCP-CMA algorithm is shown in Table 3.

Table 2 Rule-generator algorithm

```
Algorithm Rule-Generator (W)
//Purpose:  To generate the Rule Table
//Input:    Web Log W
//Output:   Rule Table
Rule-Generator(W)
{
   CW = Preprocessing (W)
   SP = Find 2-sequence Patterns (CW)
   For each Pattern $sp_i \in SP$ do
      $P_i$ = Compute periodicity ($sp_i$)
      $C_i$ = Compute cyclic behaviour ($sp_i$)
   Endfor
   Store P and C in the prefetching rule depository in the server
}
```

Table 3 WCP-CMA algorithm

```
Algorithm WCP-CMA($o_i$)
//Purpose:    To perform integrated web prefetching and caching
              in client-side proxy by using cyclic model
              analysis
//Input:      Rule table, web object $o_i$, reference rate $\delta_i$, time
              window $w$, frequency $F_i$, periodicity $P_i$ and cyclic
              behaviour $C_i$ of $o_h \rightarrow o_i$, cache and linked list $L$
//Output:     Cache and linked list $L$
Profit($o_i$)
{
  Compute $Profit(o_i) = \delta_i w + \frac{C_i - (P_i \times F_i)}{C_i}$ // Equation (9)
  Return Profit($o_i$)
}
WCP-CMA-Proxy($o_i$)
{
  //Whenever object $o_i$ has been requested from client
  If ($o_i \in$ cache) Then
    Send $o_i$ from cache to the client
    Increment $F_i$
    Compute Profit($o_i$)
    If (($F_i \times P_i$)> $C_i$) Then
      Delete $o_i$ from $L_i$ and cache
    Endif
    Read prefetching rules corresponding
    to $o_i$ and store into $H$
  Else
    Read $o_i$ from server and send to client
    Store $o_i$ in cache
    Read prefetching rules corresponding
    to $o_i$ and store into $H$
    Compute Profit($P_i$, 0,$C_i$)
  Endif
  For Each object $o_j \in H$ do
    If ($o_j \notin$ Cache) Then
      Insert $o_j$ and its $P_j$ and $C_j$ into $L_j$
      Compute Profit($P_j$, 0, $C_j$)
```

```
        Else
          Update $o_j(P_j, C_j)$ = (min (Old_$P_j$, $P_j$), min(Old_$C_j$, $C_j$))
          Compute $Profit(P_j, F_j, C_j)$
        Endif
    Endfor
    Sort $L(o_n)$ according to decreasing order of $Profits(o_n)$
    While(Number of items in $L(o_n$ > Cachesize)do
      Delete Last item from $L(o_n)$
    Endwhile
    For Each item $L(o_n)$ do
      If ($o_i \notin Cache$) Then
        Read $o_i$ from Server and Store in Cache
      Endif
    Endfor
}
```

The *rule-generator*, an independent algorithm is a server-side operation which is first performed to generate the rule table and store it in the server. The input to this algorithm is the raw web logs that are available in the server. After preprocessing and mining 2-sequence patterns, it analyses the patterns to derive the prefetching rules (Srikantaiah et al., 2013) that consist of the links pertaining to each web object and each of the link's periodicity and cyclic behaviour (see Table 4). The *rule-generator* algorithm is called as and when the situation seems appropriate. If the server handles heavy traffic, *rule-generator* must be called frequently (every few hours) and if the traffic is sparse, then it can be called once in a few days interval (two to three days).

Table 4 Example rule table

Web object (o_i)	*Reference rate (δ_i)*	*Time window (w)*	*Prefetching rules*
K	2	5	-
L	1.8	5	$(L \rightarrow P, P_P = 5, C_P = 10)$
M	1.6	5	-
N	0.9	5	$(N \rightarrow O, P_O = 3, C_O = 9)$
			$(N \rightarrow P, P_P = 6, C_P = 16)$
O	1.4	5	-
P	1.5	5	$(P \rightarrow Q, P_Q = 5, C_Q = 15)$
			$(P \rightarrow R, P_R = 5, C_R = 20)$
Q	1.9	5	-
R	1	5	-

The procedure *profit* in WCP-CMA computes the profit value for a web object and is called by the main procedure *WCP-CMA-Proxy*, both of which are proxy-side operations. The *WCP-CMA-Proxy* procedure does the actual prefetching and caching in the client-side proxy system. It has many inputs such as implied web objects, prefetching rules, cache contents, linked list *L* and so on. This procedure is called each time a web object has been requested from the client. First, the procedure checks whether the requested object is present in the cache or not. If present, it records a 'hit' and sends the object from the proxy itself. Next the frequency of access of the object is incremented and the *profit* procedure is called to update the profit value of the object (profit decreases). Next, there is an important condition that needs to be checked, which is the main strategy employed by WCP-CMA to increase hit rates of other objects. We check whether the cyclic behaviour (as a product of *F and P*) has exceeded the threshold (*C*) or not. If so, then the object should be immediately removed from the cache and the linked list. Then, the prefetching rules corresponding to the requested object must be read from the server and stored into *H* and if the rules have already been read in an earlier access, then this step can be ignored.

Conversely, if the requested object is not present in the cache, then the procedure records a 'miss' and forwards the request to the server. The server sends the requested object and this is forwarded to the client by the algorithm. Next, the object is stored in the cache and the entry is made into the linked list after calculating the profit. Note that the profit will be at its highest as the frequency of access from cache will be 0. Then, the prefetching rules corresponding to the requested object must be read from the server and stored into *H*.

Next for each hinted object in *H*, we have to check whether the hinted object is present in the cache or not. If not present in the cache, then the object is inserted only into the linked list for the time being, along with the *P* and *C* of the sequence. The profit is also computed for the object by calling procedure *profit*. Conversely, if the hinted object is present in the cache, then the nodes pertaining to *P* and *C* of this object must be updated. If the requested object is the same, then the same rule is triggered and *P* and *C* remain the same. But, if the requested object is different, then it may have different prefetching rules for the same implied object present in *H*. So, the *P* and *C* values must be updated. For this, we take Case (3) as discussed in the previous section into consideration. On comparing the two prefetching rules, the *smaller* of the two periodicities and the two thresholds of cyclic behaviour is taken and inserted into the nodes, but the *frequency of access F should be retained*. Finally, the new profit is computed for the new value/s of *P* and/or *C*.

Next, the list *L* is sorted according to decreasing order of profit values to retain the objects with the highest profits. This is done by deleting from *L* the last item (and the corresponding file in the cache if it is present) and comparing the current number of items in *L* with the maximum size of the cache. Finally, by comparing the contents of the cache and *L*, those web objects that are not present in the cache but present in the list are prefetched and stored in the cache.

This way, we are left with those prefetched objects in the cache that have the highest profit value only. Also, since the cyclic behaviour of these prefetched objects is under surveillance, there is no concern about these objects staying indefinitely in the cache as they will be deleted once the threshold of their cyclic behaviour is reached (i.e., they are least likely to be accessed in the near future), thus paving the way for newer objects to be prefetched and thus significantly improving the hit ratio. In the following section, we discuss the working of the algorithm with an example.

5.4 Example

Here, we explain the working of WCP-CMA with an example. Consider the rule table as shown in Table 4. This table is only an extract from a large database of prefetching rules contained in the web server.

Assume that the objects *K*, *L*, *M* and *N* are already stored in the cache due to previous transactions. The cache size is four. The snapshot of the linked list is as shown in Figure 2. *K* has the highest profit value and *N* the least. *K* has been accessed once, *L* thrice, *M* twice and *N* has not been accessed from the cache at all. Let object *N* be requested by the client. As it is already stored in the cache, it is a hit. As shown in Figure 3, the F_N increases by 1 and *Profit*(*N*) decreases to 5.2. By requesting *N*, the client triggers two prefetching rules as shown in Table 4.

Figure 2 Initial linked list

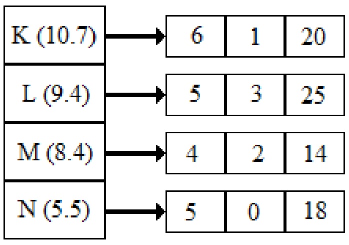


Figure 3 *O* and *P* are added to the list

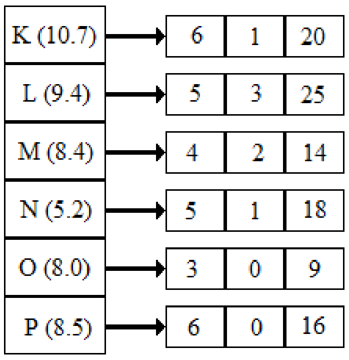


Figure 4 Last two items removed from the list

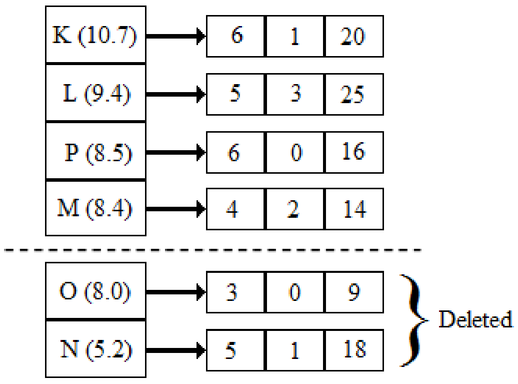
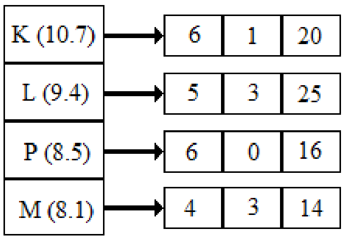


Figure 5 *M* reaches its cyclic behaviour



So, WCP-CMA prefetches the objects (*O* and *P*) and places them in the list only. The unsorted list is shown in Figure 3. The list is sorted according to the descending order of profits. As the maximum number of objects that the list (and cache) can store is four, and the current size of the list is six, the last two items, *O* and *N*, are deleted from the list, and object *N* is deleted from the cache (note that objects *O* and *P* are not yet prefetched) as shown in Figure 4. The list finally contains *K*, *L*, *P* and *M* and since object *P* is not present in the cache, it is prefetched and stored there by WCP-CMA.

Next, let *L* and *M* be accessed by two different users. They are both hits. First consider *M*. F_M increases by 1 to become 3 and *Profit*(*M*) becomes 8.1. Clearly, $C_M = 14$ has almost been reached as $4 \times 3 = 12$ as shown in Figure 5. When a 3rd user accesses *M*, then $4 \times 4 = 16$ which is greater than 14. Now let us consider *L*. By accessing *L*, the user triggers the rule ($L \rightarrow P$, $P_P = 5$, $C_P = 10$). Since *P* is already cached, WCP-CMA compares the old and new P_P and C_P values to bring about the efficient case of minimum *P* and *C*.

So, $P_P = 5$ since minimum of 5 and 6 is 5 and $C_P = 10$ since minimum of 10 and 16 is 10. The new profit value is computed and the list is updated as shown in Figure 6. Note that WCP-CMA retains the same *F* value. Next, let *P* be accessed by a user. It is a hit again. F_P becomes 1 and *Profit*(*P*) reduces to 8.0. By accessing *P*, the user triggers two prefetching rules ($P \rightarrow Q$, $P_Q = 5$, $C_Q = 15$) and ($P \rightarrow R$, $P_R = 5$, $C_R = 20$). Both of these implied objects *Q* and *R* are placed in the list as shown in Figure 7 and their profits computed. After sorting, *R* is removed from the list and finally *Q* is prefetched and cached into the proxy as shown in Figure 8. It should be noted that *Q* has a higher profit value than that of *M* which was evicted due to crossing of C_M threshold. IWCP also would have evicted *M* after sorting due to its lower profit value. Suppose *Q* had a lesser profit value than that of *M*, then IWCP would have failed as *M* would still be retained in the cache though it is no longer required to be present in the cache since it will not be accessed in the near future and *Q* would have been a miss. Therefore, in such a case, WCP-CMA would have prevented a miss by caching such objects by freeing the proxy of web objects that are no longer required to be cached.

Figure 6 *M* is removed and *P* has new values

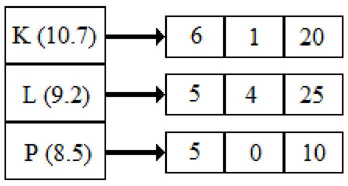


Figure 7 *Q* and *R* prefetched

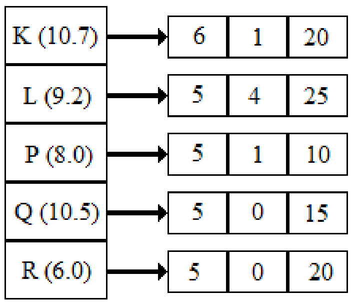
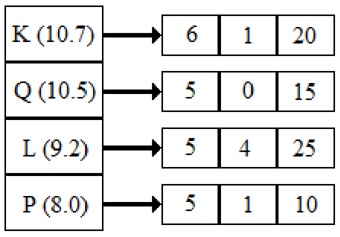


Figure 8 Final cache status



6 Proposed system architecture

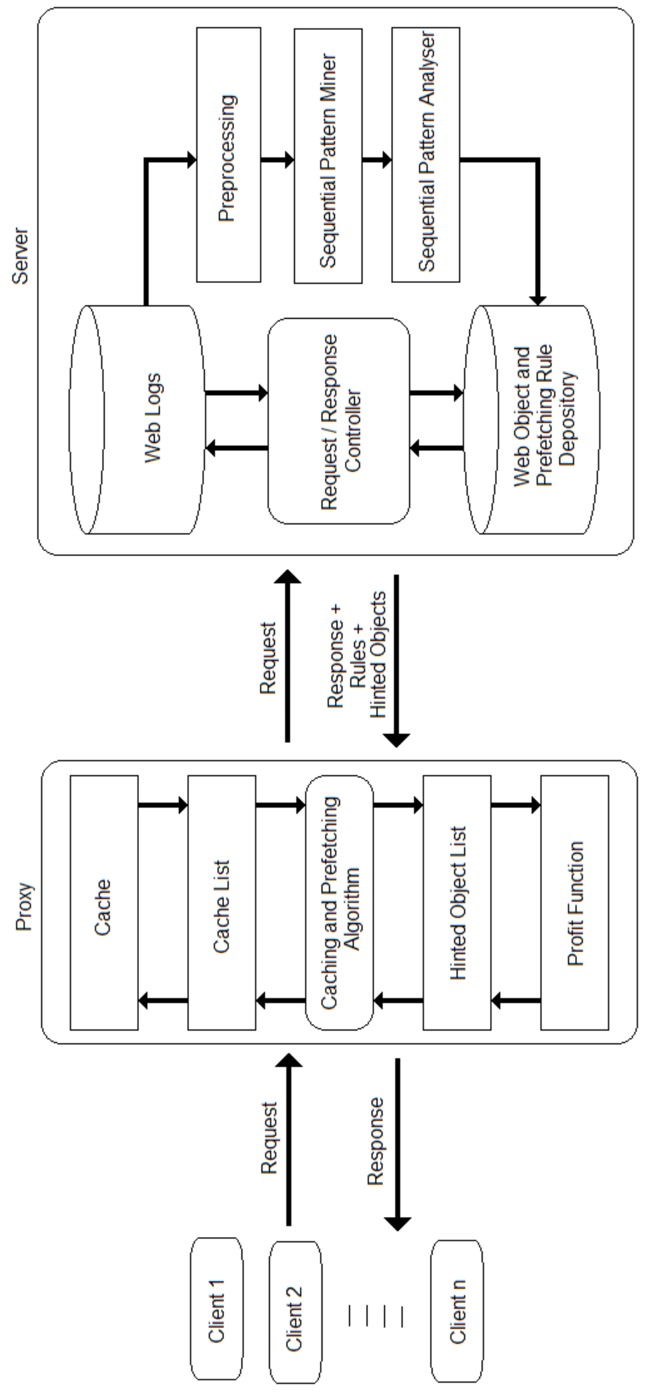
The system architecture consists of the following components,

1 the clients

2 the client-side proxy

3 the server is as shown in Figure 9.

Figure 9 The proposed system architecture


Server
Web Logs
Preprocessing
Sequential Pattern Miner
Sequential Pattern Analyser
Request / Response Controller
Web Object and Prefetching Rule Depository
Request
Response + Rules + Hinted Objects
Proxy
Cache
Cache List
Caching and Prefetching Algorithm
Hinted Object List
Profit Function
Request
Response
Client 1
Client 2
Client n

6.1 Clients

They are the different users that access the web objects through the help of web Browsers. The client systems send requests in the form of messages to the proxy/server and wait for the response from the proxy/server, which is usually a message or the desired web object.

6.2 Proxy

In a general environment, a client-side proxy accepts requests from various clients and forwards these requests to the server. The client-side proxy can serve *n* number of client systems at a time. The greatest advantage of implementing web caching and web prefetching at the client-side proxy is the fact that a cached web object may be accessed by another client which had not previously foreseen that it would need to access that object. That is, an object cached due to a prefetching rule triggered by one client can also be easily accessed by another client triggering another rule on the same implied object, without having to refer all the way up to the origin web server. The proxy consists of the following sub-components:

- *Caching and prefetching algorithm:* The caching and prefetching algorithm is the main function (WCP-CMA) that controls the other units of the proxy. The caching and prefetching operations are integrated into a single function. Besides interacting with the clients and the server, this module also decides which web objects to cache and maintain the cache and hinted object lists.
- *Cache:* The proxy consists of a fast, volatile memory called the cache which is used for the temporary storage of the web objects for prefetching and caching operations. The size (i.e., number of objects to be cached) of the cache is specified by the algorithm.
- *Cache list:* Cache list is a linked list that is dynamically maintained by the algorithm to track the status of the cache. Each item in the linked list pertains to one unique web object. Each node connected to the items is the update to that item (i.e., they specify the number of times the object was accessed). The nodes also contain two other fields which specify the periodicity and the cyclic behaviour of the item (web object) both of which remain constant at all times. Using these two fields as references, as the frequency of access of the web object increases and thus decreases its profit value.
- *Hinted object list:* This list is present in the proxy and is used to store the names of the hinted web objects that are sent along with the requested web object. Later, each hinted object pertaining to the requested object is examined whether it is present in the cache or not. If not, it is prefetched from the server and the corresponding entry is made in the cache linked list.
- *Profit function:* The profit function is the heart of the caching and prefetching algorithm. The profit value for a web object denotes the benefit gained by caching the said object. Higher the value, higher the benefit and vice-versa. The profit value is the criterion for determining whether an object has to be deleted from the cache or not. Unlike the profit function used by IWCP (Teng et al., 2005) which makes use of

the confidence of the prefetching rule, our profit function makes use of periodicity and cyclic behaviour of the prefetching rule, which yields better results.

6.3 Server

The web server is the module that accepts requests from clients/proxy and provides response for such requests. The server consists of the following sub-modules:

- *Request/response controller:* This module is responsible for accepting requests from the clients/proxy and returning the appropriate responses for those requests. Each request/response is stored in the web log database in the server. This module is responsible for the communication between the server and its clients and encapsulates the internal operations of the server from the clients. Also, this module interacts with the other sub-modules of the server system.
- *Web logs:* A web log is a large database consisting of the entries pertaining to each request of the client systems. There are many fields in the web log database, which conform to either common log format (CLF) or the extended common log format (ECLF). Some of the most important fields are the IP address of the client, URL, date and time, method, protocol, referrer, etc., web logs are very useful for predicting the behaviour of the clients and hence different mining techniques can be used to find and extract interesting browsing patterns of the clients.
- *Preprocessing component:* Preprocessing is the process in which the web logs are made uniform and consistent by removing unwanted details (such as scripts and image logs) and noise. There are five steps in preprocessing:
  1. *merging*: data collected from different sources is merged into a single component
  2. *data cleaning*: is the process of removing invalid and irrelevant entries
  3. *user identification*: users (clients) should identified by their logins and IPs
  4. *session identification*: users' sessions i.e., the list of requests by a user has to be identified along with the number of sessions of the user over different periods of time
  5. *data formatting and summarisation*: a proper format is applied to the web log data and is summarised by aggregation.
- *Sequential pattern miner:* After the preprocessing, the web logs have to be analysed for finding navigational patterns of browser requests. Each pattern consists of sequences of web objects that are visited by a client in different sessions. These patterns are mined using an efficient sequence pattern mining algorithm.
- *Sequential pattern analyser:* The sequence patterns generated by the sequence pattern miner should be further analysed to derive the prefetching rules. The analyser finds the periodicity and the cyclic behaviour of these sequential. Using these parameters, we can shorten the lives of implied objects in the cache that have a tendency to become obsolete over time and consequently prefetch and cache more web objects due to the freeing of the cache.

- *Web objects and prefetching rule depository:* It is a very large collection of the web objects such as web pages, images, videos, scripts, etc. These web objects are sent to the client through the proxy whenever they are requested. This module also contains a large database of prefetching rules pertaining to the requested web objects. These rules are stored after an exhaustive mining process and are sent along with the requested objects as hints to the caching and prefetching algorithm in the proxy. After analysing these rules, the implied objects that are requested by the proxy are sent from this depository.

7 Experimental results

The algorithm WCP-CMA has been implemented using Java Server Pages (JSP). The various operations pertaining to the proxy and the server have been implemented using Java in a Pentium Dual Core processor environment, with a 2 GB memory and 100 GB HDD. Apache Tomcat has been utilised as the server, while Google Chrome web browser has been utilised as the client.

A synthetic dataset comprising of 200 web objects with 559 links among them has been used in the simulation. The WCP-CMA algorithm has been implemented in the proxy in the client-side and will be triggered on every request from the client. The round-trip delay between the client and the web server and the client and the client-side proxy is assumed to be 10 ms and 5 ms respectively. The cache size is the maximum number of web objects that the proxy's cache can store at a time. The cache size should be as small as possible for the efficient running of the algorithm. The algorithm WCP-CMA has been simulated for different cache sizes and has been compared with IWCP. The following sections discuss the parameters used for evaluating the performance of WCP-CMA.

7.1 Cache hit ratio

Cache hit ratio is defined as the percentage of cache hits with respect to the total number of transactions that have occurred over a certain period of time. If the implied web object is found in the proxy's cache itself, it is a 'hit', otherwise it is a 'miss' (object has to be fetched from the web server). In Figure 10, it can be seen that the hit ratios are higher while using WCP-CMA instead of IWCP. Also, the hit ratios of the two schemes differ more in the region between 4 and 6 and tend to become equal as size of the cache increases.

Figure 10 clearly shows that WCP-CMA increases the hit ratios of the cached web objects by 10%–15% compared to IWCP. This is due to the fact that the web objects cached using IWCP tend to remain in the proxy's cache for duration longer than necessary. It would suffice to have the implied web objects cached only up to the threshold of their cyclic behaviour. Since IWCP utilises the confidence of web access sequences and not their periodicity and cyclic behaviour, it would naturally result in lower hit ratios than that of WCP-CMA. This is because the new implied objects which have lower profit values than the previously cached objects may not be allowed to be cached even though they may help in preventing at least one miss. Hence, some of the implied objects cached using IWCP not only remain in the cache for a large duration of time and never be accessed, but also cause other useful implied objects to be mercilessly evicted from the cache.

Figure 10 Hit ratio vs. cache size

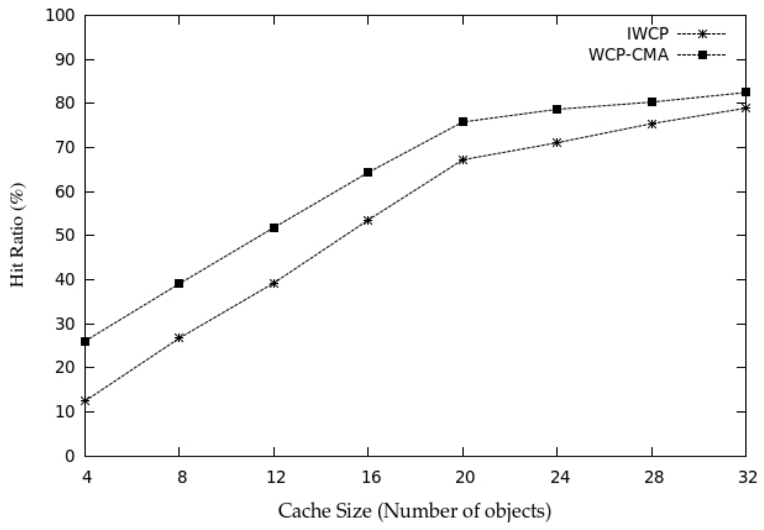
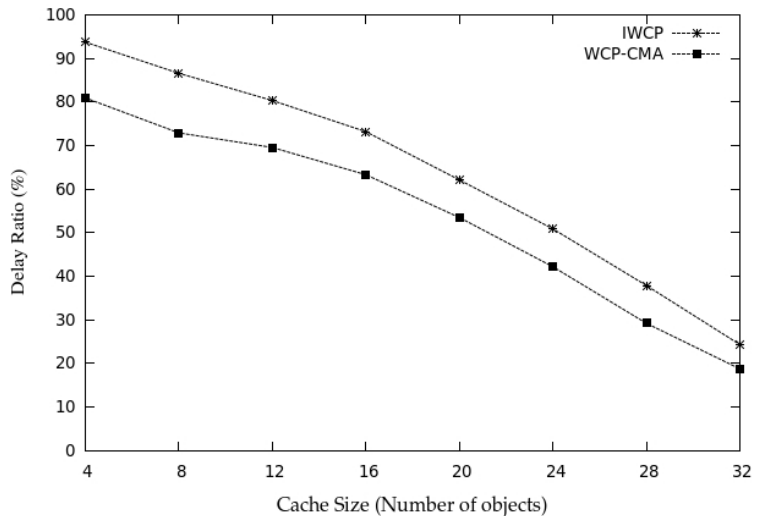


Figure 11 Delay vs. cache size



7.2 Delay

Delay is the round-trip latency perceived by the user when a web object is requested. Delay is lower when a 'hit' occurs, i.e., the object is available in the proxy itself, and more when a 'miss' occurs, i.e., the object has to be accessed from the server. In Figure 11, it can be seen that the delay experienced by the user is slightly lower when using WCP-CMA compared to IWCP. However, delay shall tend to be the same as the cache size increases. Figure 11 shows that WCP-CMA decreases the delay experienced by the user of the cached web objects by 5%–10% compared to IWCP. The delay is related to the availability of the web objects in the proxy's cache. As WCP-CMA increases availability of the web objects, it results in reduced delay as there is a higher hit ratio compared to IWCP.

As more web objects are available in the proxy's cache itself, the delay incurred by the client is reduced to some extent as the implied objects need not be accessed all the way from the origin web server. Since IWCP does not evict objects cached in the proxy until implied objects with higher profit values occur, it increases delay as new implied objects are not available at the proxy.

7.3 Effect of periodicity

It is observed from our experiments that as periodicity value increases, the implied web object tend to appear less frequently, irrespective of the cyclic behaviour. Caching such an implied object can cause moderate reduction in hit ratios of other implied objects whose rules have smaller periodicity values. For such implied objects whose rules have high periodicity values, a certain threshold of periodicity can be taken into consideration. One way to solve this problem is to introduce a secondary cache in the proxy to be used for storing only those objects whose periodicity exceeds the threshold value, thus releasing the primary cache. The secondary cache shall contain only those prefetched objects whose periodicities exceed the pre-specified threshold. This way, if the periodicity requires that the objects should be available at p^{th} second, then that object is simply copied from the secondary to the primary cache in the proxy at $(p - 1)^{\text{th}}$ second. After it has been accessed, it can be deleted from the primary cache thus paving the way for more objects (with much smaller periodicities) to be prefetched and cached increasing their hit ratio. Conversely, the implied web objects whose rules have smaller values of periodicity are more frequently accessed and it is beneficial for them to be stored directly in the primary cache itself.

7.4 Effect of cyclic behaviour

The rules which have a very high threshold value of cyclic behaviour tend to be in the cache for a long duration. Whilst caching implied objects with small values of periodicities and cyclic behaviour is extremely beneficial as seen in Case 4, it does not mean that those with high values of cyclic behaviour should not be considered. The reason for this is that these implied objects may have high reference rates and subsequently high profit values. Due to this, it may be highly beneficial to store such objects for a longer duration in the proxy's cache as they may be accessed by a large number of users. In such cases, the Case 4 has to be ignored and Case 3 (small P and large C) can be considered. This can be achieved by setting a threshold value for the reference rate and for those implied objects with a large reference rate and a large cyclic behaviour of the triggered rule, the larger of the two cyclic behaviours is to be considered if the object is already present in the cache. However, care should be taken that these additional conditions do not reduce the efficiency of WCP-CMA and so separate cases can be implemented in separate proxies based on the demand and, as and when they are applicable.

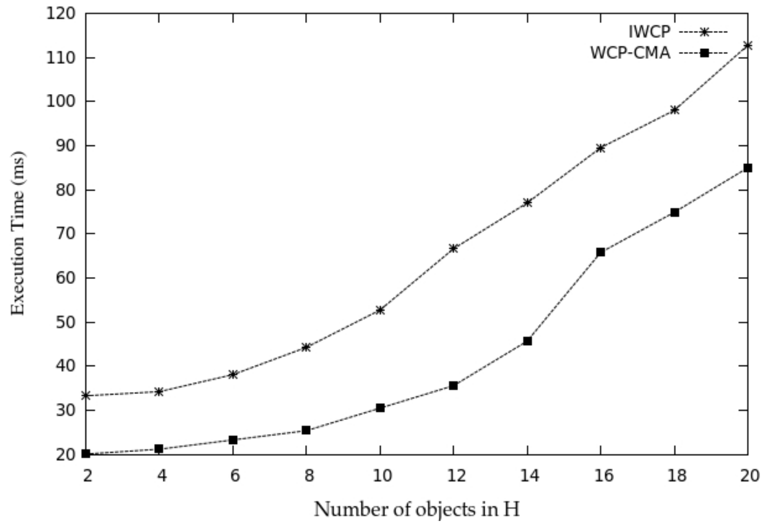
7.5 Execution time

For every user request, the algorithm WCP-CMA is called and executed in the proxy. The time taken for the execution is determined as follows.

Let *n* be the number of objects present in the hinted object list *H*. Let *t* be the time taken to compute the profit and *l* the time taken to insert/update the list. The time complexity function for computing profit and handling list operations for *n* objects is given by the expression $T(n) = n \times t \times l$. Therefore, the time complexity is of the order $O(n)$. Heapsort algorithm has been used to sort the *n* objects in the list in decreasing order. Hence the time complexity of sorting operation is $O(n \log n)$. Therefore, the overall time complexity of WCP-CMA is $O(n \log n)$.

WCP-CMA's execution time is much lower than that of IWCP as shown in Figure 12. This is due to the fact that IWCP prefetches and caches all those objects that are present in *H* and are not in the cache and then deletes some of the same prefetched objects due to cache size constraint after sorting. However, in our approach, we first insert those implied objects from *H* not present in the cache in the linked list only and do not prefetch them. After sorting the list and deleting the objects from the list, those objects that are present in the list but not present in the cache are the ones that are prefetched and cached. This way, the unnecessary objects i.e., the objects prefetched only to be deleted later (in the same cycle of execution of WCP-CMA) are *not* prefetched at all by WCP-CMA. This saves quite a significant amount of execution time, bandwidth, network load and proxy resources such as processor and memory.

Figure 12 Execution time vs. number of objects in H



8 Conclusions

In this paper, we have implemented a novel integrated web caching and prefetching policy called WCP-CMA that makes use of cyclic model analysis of the web access sequences for deriving prefetching rules. Our experimental results have shown a 10%–15% increase in the hit ratios of the cached objects and 5%–10% decrease in delay compared to IWCP. Also, the execution time of WCP-CMA is significantly lower than that of IWCP and WCP-CMA saves more network and the proxy's local resources. Hence, it is obvious that our approach is more efficient than IWCP. We have taken only Case 4 i.e., for small values of periodicity (*P*) and cyclic behaviour (*C*) for our implementation and the other three cases (especially Case 3) can be taken into consideration for further improvement.

References

Balamash, A. and Krunz, M. (2004) 'A client side WWW prefetching model', *Globecom Telecommunications Conference*, pp.948–952.

Chen, J. (2010) 'An up down directed acyclic graph approach for sequential pattern mining', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 7, pp.913–928.

Chiang, D-A., Wang, C-T., Chen, S-P. and Chen, C-C. (2009) 'The cyclic model analysis on sequential patterns', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, No. 11, pp.1617–1628.

Eden, A.N., Joh, B.W. and Mudge, T. (2000) 'Web latency reduction *via* client-side prefetching', *IEEE International Symposium on Performance Analysis of Systems and Software*, pp.193–200.

Feng, W. and Vij, K. (2008) 'Web cache prefetching by multi-dimensional matrix', *IEEE International Conference on Advanced Software Engineering and its Applications (ASEA 2008)*, pp.265–270.

Foygel, D. and Strelow, D. (2000) 'Reducing web latency with hierarchical cache-based prefetching', *International Workshop on Parallel Processing*, pp.103–108.

Huang, Y-F. and Hsu, J-M. (2005) 'Mining web logs to improve hit ratios of prefetching and caching', *IEEE International Conference on Web Intelligence*, pp.577–580.

Jeon, J., Lee, G., Cho, H. and Ahn, B. (2003) 'A prefetching web caching method using adaptive search patterns', *IEEE Pacific Rim Conference on Communications*, *Computers and Signal Processing*, pp.37–40.

Jin, B., Tian, S., Lin, C., Ren, X. and Huang, Y. (2007) 'An integrated prefetching and caching scheme for mobile web caching system', *8th ACIS International Conference on Software Engineering*, *Artificial Intelligence*, *Networking*, *and Parallel/Distributed Computing*, pp.522–527.

Kim, Y. and Kim, J. (2003) 'Web prefetching using display-based prediction', *IEEE International Conference on Web Intelligence*, pp.486–489.

Klemm, R.P. (1999) 'WebCompanion: a friendly client-side web prefetching agent', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 4, pp.577–593.

Li, J., Wang, Z.X., Zeng, D. and Wang, F-Y. (2001) 'Combined coherence and prefetching mechanisms for effective web caching', *IEEE International Conference on Systems*, *Man and Cybernetics*, pp.3034–3038.

Marquez, J., Domenech, J., Gil, J.A. and Pont, A. (2009) 'A web caching and prefetching simulator', *16th International Conference on Software*, *Telecom*, *and Computer Networks*, pp.346–350.

Nigam, B. and Jain, S. (2010) 'Analysis of Markov model on different web prefetching and caching schemes', *IEEE International Conference on Computational Intelligence and Computing Research*, pp.1–6.

Pallis, G., Vakali, A. and Pokorny, J. (2008) 'A clustering-based prefetching scheme on a web cache environment', *Elsevier Computers and Electrical Engineering*, Vol. 34, No. 4, pp.309–323.

Shi, L., Han, Y., Ding, X., Wei, L. and Gu, Z. (2005) 'SPN model for web prefetching and caching', *First International Conference on Semantics*, *Knowledge*, *and Grid*, pp.24–29.

Songwattana, A. (2008) 'Mining web logs for prediction in prefetching and caching', *Third International Conference on Convergence and Hybrid Information Technology*, pp.1006–1011.

Srikantaiah, K.C., Krishna, K.N., Venugopal, K.R. and Patnaik, L.M. (2013) 'Bidirectional growth based mining and cyclic behaviour analysis of web sequential patterns', *International Journal on Data mining and Knowledge Discovery Process (IJDKP)*, AIRCC, ISSN 2230-9608 (online), ISSN 2231-007X (print), March, Vol. 3, No. 2, pp.49–68.

Teng, W-G., Chang, C-Y. and Chen, M-S. (2005) 'Integrating web caching and web prefetching in client-side proxies', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 5, pp.444–455.

Yang, Q., Zhang, H.H. and Li, T. (2001) 'Mining web logs for prediction models in WWW caching and prefetching', *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.1–6.