Available online at www.sciencedirect.com
ScienceDirect
Computers and Electrical Engineering 34 (2008) 309–323
Computers and Electrical Engineering
www.elsevier.com/locate/compeleceng

A clustering-based prefetching scheme on a Web cache environment

George Pallis [a,*], Athena Vakali [a], Jaroslav Pokorny [b]

[a] Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
[b] Faculty of Mathematics and Physics, Charles University, Praha, Czech Republic

Received 19 April 2006; received in revised form 12 February 2007; accepted 27 April 2007
Available online 5 September 2007

Abstract

Web prefetching is an attractive solution to reduce the network resources consumed by Web services as well as the access latencies perceived by Web users. Unlike Web caching, which exploits the temporal locality, Web prefetching utilizes the spatial locality of Web objects. Specifically, Web prefetching fetches objects that are likely to be accessed in the near future and stores them in advance. In this context, a sophisticated combination of these two techniques may cause significant improvements on the performance of the Web infrastructure. Considering that there have been several caching policies proposed in the past, the challenge is to extend them by using data mining techniques. In this paper, we present a clustering-based prefetching scheme where a graph-based clustering algorithm identifies clusters of ''correlated'' Web pages based on the users' access patterns. This scheme can be integrated easily into a Web proxy server, improving its performance. Through a simulation environment, using a real data set, we show that the proposed integrated framework is robust and effective in improving the performance of the Web caching environment.

Keywords: Web prefetching; Web data clustering; Web caching; Users' access patterns; World Wide Web

1. Introduction

The Web has evolved rapidly from a simple information-sharing mechanism offering only static text and images to a rich assortment of dynamic and interactive services, such as video/audio conferencing, e-commerce, and distance learning. The explosive growth of the Web has imposed a heavy demand on networking resources and Web servers. Users often experience long and unpredictable delays when retrieving Web pages from remote sites [21]. Hence, an obvious solution in order to improve the quality of Web services would be the increase of bandwidth, but such a choice involves increasing economic cost. However, the higher bandwidth would solve temporarily the problems since it would ease the users to create more and more

* Corresponding author. Tel.: +30 2310 991927; fax: +30 2310 998419.
E-mail address: gpallis@ccf.auth.gr (G. Pallis).

0045-7906/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compeleceng.2007.04.002

resource-hungry applications, bunching again the network. Therefore, the network limitations will remain or worsen unless effective software solutions are also provided.

1.1. The Web caching approach

Caching proved itself as an important technique to optimize the way the Web is used [26]. In particular, many of the Web caching aspects are originated from the caching idea implemented in various computer and network systems whereas Web caching introduces new issues in Web objects management and retrieval across the network. Specifically, Web caching is implemented by proxy server applications developed to support many users. Proxy applications act as an intermediate between Web users and servers. Users make their connection to proxy applications running on their hosts. The proxy connects the server and relays data between the user and the server. At each request, the proxy server is contacted first to find whether it has a valid copy of the requested object. If the proxy has the requested object this is considered as a cache hit, otherwise a cache miss occurs and the proxy must forward the request on behalf of the user. Upon receiving a new object, the proxy services a copy to the end-user and keeps another copy to its local storage.

From the above discussion follows that Web caching reduces bandwidth consumption, network congestion, and network traffic because it stores the frequently requested content closer to users. Also, because it delivers cached objects from proxy servers, it reduces external latency (the time it takes to transfer objects from the origin server to proxy servers). Finally, caching improves reliability because users can obtain a cached copy even if the remote server is unavailable. As far as concerned the performance of a Web proxy caching scheme, it is mainly dependent on the cache replacement algorithm [25] (identify the objects to be replaced in a cache upon a request arrival) which has been enhanced by the underlying proxy server. However, cache hit rates have not improved much with these schemes. Particularly, a Web caching scheme has three significant drawbacks: If the proxy is not properly updated, a user might receive stale data, and, as the number of users grows, origin servers typically become bottlenecks. For instance, when numerous users access a Web site simultaneously – such as when ''flash crowds'' flooded popular news sites with requests in the wake of the September 2001 terrorist attack in the US – serious caching problems result and sites typically become unavailable [1]. Finally, several factors diminish the ideal effectiveness of Web caching. The obvious factors are the limited system resources of cache servers (i.e., memory space, disk storage, I/O bandwidth, processing power, and networking resources). However, even if the cache space is unlimited, there are significant problems that cannot be avoided by such an approach. Specifically, large caches are not a solution because, the problem of updating such a huge collection of Web objects is unmanageable. Therefore, we must resort to an approach, which will predict the future users' requests and retain in cache the most valuable objects.

1.2. The Web prefetching approach

Prefetching attempts to overcome these limitations by pro-actively fetching content before users actually request it [11]. Web prefetching is the process of deducing user's future requests for Web objects by locating popular requested objects into the cache prior to an explicit request for them. Unlike Web data caching, which exploits the temporal locality, the Web prefetching schemes are based on the spatial locality of Web objects. In particular, the temporal locality refers to repeated users' accesses to the same object within short time periods, whereas, the spatial one refers to users' requests where accesses to some objects frequently entail accesses to certain other objects. Typically, the main benefits of employing prefetching is that it prevents bandwidth underutilization and reduces the latency. Therefore, bottlenecks and traffic jams on the Web are bypassed and objects are transferred faster. Thus, the proxies may effectively serve more users' requests, reducing the workload from the origin servers. Consequently, the origin servers are protected from the flash crowd events as a significant part of the Web traffic is dispersed over the proxy servers. On the other hand, the main drawback of systems which have enhanced prefetching policies is that some prefetched objects may not be eventually requested by the users. In such a case, the prefetching scheme increases the network traffic as well as the Web servers' load. In order to overcome this limitation, high accuracy prediction models have been used [32].

From the above it occurs that caching and prefetching complement each other in order to reduce the noticeable response time perceived by users [16]. Table 1 presents the main difference between Web caching and Web

Table 1
Caching vs. prefetching

Approach	Locality	Architecture	Objects' placement
Caching	Temporal	Pull-based	Reactive
Prefetching	Spatial	Push-based	Proactive

prefetching. In this paper, we propose a scheme which integrates efficiently caching and prefetching approaches. Specifically, the potential main advantage of adopting prefetching policies over a proxy cache server is that we manage effectively the Web content by exploiting both the temporal and the spatial locality of objects. Another important feature in this paper is that we represent the Web users' requests using a Web navigational graph. The rest of this paper is organized as follows: Section 2 reviews the related work and outlines the motivation and contribution of this work. Section 3 describes the proposed framework, which is the clustering algorithm and how it is adapted in the prefetching scheme. Section 4 provides the experimental results and finally, Section 5 concludes the paper.

2. Related work and paper's contribution

Prefetching and caching are two well-known approaches for improving the performance of the Web and have become essential components of the Web infrastructure. The benefits of these technologies have given rise to new industries, including equipment and service vendors that supply cache servers that offer caching and prefetching services to consumers and providers of Web resources. Nowadays, a number of commercial systems implement some form of prefetching. For example, a number of browser extensions for FireFox [9], Netscape and Microsoft Internet Explorer as well as some personal proxies that perform prefetching of links of the current page. In this section, we provide a classification of the existing prefetching policies and we further present the motivation and contribution of this work.

2.1. Web prefetching

The Web prefetching approaches can be characterized according to its short-term and long-term benefits. In this context, we categorize the existing prefetching policies as follows:

- *Short-term prefetching policies*: Future requests are predicted to the cache's recent access history. Based on these predictions, clusters of Web objects are pre-fetched [4]. In this context, the short-term prefetching schemes use Dependency Graph (DG), where the patterns of accesses are held by a graph and Prediction by Partial Matching (PPM), where a scheme is used, adopted from the text compression domain [3,24]. In addition, several short-term prefetching policies [6,19] are based on Markov models, which are used for modeling and predicting user's browsing behavior over the Web.
- *Long-term prefetching policies*: Global object access pattern statistics (e.g., objects' popularity, objects' consistency) are used to identify valuable (clusters of) objects for prefetching. In this type of scheme, the objects with higher access frequencies and no longer update time intervals are more likely to be prefetched [30].

The existing Web caching schemes use the short-term prefetching policies, by prefetching objects which are likely to be referenced in the near future. On the other hand, the long-term prefetching scheme may be applied on replication schemes (such as a Content Distribution Network (CDN) platform) [21] as well as on mobile computing environments [7,17]. Here, we deal with short-term prefetching policies which are applied on a Web caching environment.

2.2. Motivation and paper's contribution

Short-term prefetching helps on reducing the user-perceived latency, however, it suffers from two drawbacks. First, without a carefully designed prefetching policy, a prefetching scheme may cause excessive

network traffic. For instance, an aggressive one has as result that several documents (which have been prefetched by the proxy) are not used by the user at all. On the contrary, if the prefetching control is strict, a proxy will tend to discard some beneficial hints provided by the Web servers. Second and foremost, the cache space is not used optimally. Considering that the cache space is limited, the challenge is to integrate Web caching and Web prefetching in order to improve the Web cache performance.

Motivating by the wealth of research in Web caching [13,25,28,32] as well as the benefits of Web prefetching [16,26,28], the idea of this paper is to integrate these approaches. We extend the most popular cache replacement policies which have been adopted by proxy servers (e.g., Squid[1]) by inserting a prefetching mechanism. More precisely, we present a clustering-based prefetching scheme where a number of clusters of ''correlated'' Web pages based on the users' access patterns is identified. These pages may belong to different Web sites. According to the user's request, one of the resulted clusters is selected to be fetched by the proxy cache, where each proxy manages its content by a cache replacement policy.

There is a wide range of Web data clustering schemes in the literature [29], where most of them cluster Web pages which belong in the same Web site (intra-site Web pages). However, due to the complex nature of the Web, most clustering schemes have low performance if they are applied to grouping inter-site Web pages (Web pages which belong in different Web sites). Furthermore, the content of the resulted clusters should be adapted to changes in the Web users' patterns, which are rather natural in the Web. Thus, the Web log file is represented by a Web navigational graph and we use graph partition techniques [27], which have been inspired by the graph theory, in order to construct efficient clusters of Web pages. The present paper makes the following contributions:

- We present an integrated approach which combines effectively caching and prefetching. Specifically, Web caching and prefetching can complement each other since the first one exploits the temporal locality whereas the second one utilizes the spatial locality of the Web objects.
- We represent the proxy traces (users' requests) using a Web navigational graph. Then, we effectively manage it by using graph mining techniques.
- We introduce an algorithm for clustering inter-site Web pages, called *clustWeb*. According to this algorithm, the clusters have been resulted by partitioning the Web navigational graph using association rule mining techniques. The efficiency of the algorithm is that the resulted clusters are based on the connectivity among Web pages in a Web navigational graph.
- We develop a simulation environment to test the efficiency of the proposed integrated scheme. Specifically, we provide a clustering based short-term prefetching scheme, called *clustPref*, which can be easily adapted on a Web cache environment, integrating in a sufficient way both caching and prefetching. According to this scheme, each time a user requests an object, the proxy fetches all the objects which are in the same cluster with the requested object. Using real data, we show the robustness and efficiency of the proposed method.

3. Clustering towards prefetching

In the next paragraphs, we present the process which we follow in order to determine which objects to select for prefetching in the Web caching environment as well as the proposed prefetching scheme.

3.1. Preprocessing proxy logs

A Web user may visit several Web sites from time to time and spend arbitrary amount of time between consecutive visits. To deal with the unpredictable nature of Web browsing, we should analyze the Web proxy log file. In particular, the Web proxy access log is a sequential file with one user access record per line. Considering that each Internet Service Provider (ISP) has a proxy server cache, the Web proxy log files provide

[1] Squid Web Proxy Cache: http://www.squid-cache.org/.

```
986074304.817 81019 155.207.148.16 TCP MISS/503 1180 GET
http://www.mymobile.com/ - DIRECT/www.mymobile.com -
986074304.828 51360 155.207.128.30 TCP MISS/000 214 GET
http://www.battle.net/includes/ads.js - DIRECT/www.battle.net -
986074312.188 3140 155.207.128.3 TCP MISS/000 123 GET
http://www.battle.net/includes/ads.js - DIRECT/www.battle.net-
```

Fig. 1. A sample access log file.

information about activities performed by a user from the moment the user logs in to the ISP to the moment the same user logs out from it. An access proxy log entry usually consists of the following fields:

- *Time:* A Unix timestamp of the date and time of the request with a millisecond resolution.
- *Duration:* The elapsed time considers the milliseconds that the transaction busied the cache.
- *Client address:* The user IP address of the request.
- *Result codes:* The cache result of the request contains information on the kind of request, how it was satisfied, or in what way it failed.
- *Bytes:* The size is the amount of data delivered to the user.
- *Request method:* The request method to obtain an object.
- *URL:* This column contains the URL requested.
- *rfc931:* It contains the ident lookups for the requesting user.
- *Hierarchy code:* The IP address or hostname where the request (if a miss) was forwarded.
- *type:* The content type of the object as seen in the HTTP reply header.

More details for the Squid log files can be found in the official Web site of the Squid proxy server (http://www.squid-cache.org/). An example of a Web proxy server log file is given in Fig. 1.

Then, these data are undergone a certain pre-processing [33], such as invalid data cleaning. Data cleaning removes those requests that are obviously generated by programs running on a user. In addition, we remove the uncacheable requests (i.e., queries with ? in the URLs and cgi-bin requests).

The next step is to group the IPs into different categories, according to their domains. The process of grouping the IPs into categories improves the data management and in addition it eliminates the complexity of the underlying problem (as the number of domains is smaller than the number of individual IPs) [22]. Furthermore, with such an approach, we avoid erroneous assumptions since in proxy logs, the real-world individual users cannot be uniquely identified (several records of proxy log files come from crawlers, Web accelerators, etc.) and thus multiple users may be mapped into one IP address.

3.2. The clustering approach

Since we have identified what objects have been requested by each client group (we group the users based on their domains), the next step is to assign, for each client group, these objects into clusters.[2] One simplistic solution to this problem is to cluster for each client group the most popular objects. However, authors in [4] report that the popularity of each object varies considerably. In addition, the use of administratively tuned parameters to select the most popular objects, or decide the number of clusters causes additional headaches, since there is no at priori knowledge about where to set the popularity threshold or how many clusters of objects exist. Refraining from the above limitations, we present a graph-based approach in order to cluster in an efficient way the Web pages.

[2] The clustering problem is about partitioning a given data set into clusters (groups) such that the data points in the same cluster are more similar to each other than points in different clusters.

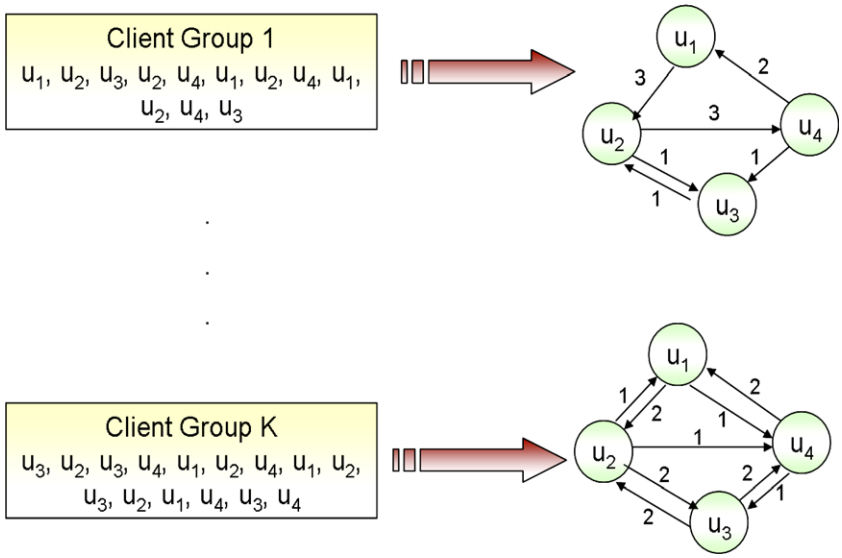

Fig. 2. The Web navigational graph.

3.2.1. Web navigational graphs

The requests of each client group are represented by a weighted directed *Web graph* $G(u,v)$, where each node u represents a Web page and each edge v represents a set of users' transitions from one Web page to another. The weight of each edge is proportional to the number of transitions in the set. An illustration example of constructing such graphs is given in Fig. 2.

However, due to the vast amount of Web pages, the resulting *Web graphs*, which have been occurred by the users' access patterns, become frequently incomprehensible and unmanageable. Therefore, association rule mining techniques should be used in order to create sub-graphs [12,18], since the nodes and the edges can be considered as binary association rules. In this work, the edges are filtered out by their weight. More specifically, we remove the edges where the connectivity between two pages is lower than a pre-specified threshold, whereas the connectivity between two pages is determined by two parameters: *support* and confidence.

Definition 1. Let $v{:}\langle u_i,u_j\rangle$ be an edge from node u_i to node u_j, the **support** of G, denoted by $freq(u_i,u_j)$, is defined as the frequency of navigation steps between u_i to u_j. The **confidence** of G is defined as $\frac{freq(u_i,u_j)}{pop(u_i)}$, where $pop(u_i)$ is the popularity of u_i.

According to this definition, it occurs that the support value of the edge $\langle u_2,u_3\rangle$ for the *client group 1* in Fig. 2 is $freq(u_2,u_3)=1$, whereas its confidence value is $\frac{freq(u_2,u_3)}{pop(u_2)}=0.25$, since $pop(u_2)=4$. In particular, if the support threshold is low, the resulted sub-graphs may include too many spurious Web pages involving transitions with substantially different support levels. If the support threshold is high, we may miss many interesting transitions occurring at low levels of support. This suggests that the confidence value is also important for capturing transitions containing Web pages which are strongly related with each other. Therefore, in order to measure the overall affinity among transitions within a Web graph, both support and confidence thresholds should be used [12].

3.2.2. The clustWeb algorithm

Here, we describe our algorithm in order to cluster inter-site Web pages. We use a weighted directed Web graph $G(u,v)$ which represents the access patterns of a client group. Then, we partition the graph into sub-graphs by filtering out some edges (edges with low support and confidence values). The nodes in each connected sub-graph in the remaining navigational graph identify a cluster.

The detailed algorithm is described in pseudocode in Fig. 4. More specifically, the input to clustWeb consists of the following information: the Web navigational graph, the number of client groups, a confidence

```
BFS Algorithm
Input: G(u,v);
unmark all nodes u;
choose some starting node x;
mark x;
list L = x;
subgraph T = x;
while L nonempty;
{ choose some node y from front of list;
visit y; }
for each unmarked neighbor w
{ mark w;
add it to end of list;
add edge yw to T; }
```

Fig. 3. The BFS algorithm.

threshold, and a support threshold. All edges with support or confidence value less than the corresponding threshold values will be removed from the graph (procedures *CutWithConfidence* and CutWithSupport). It should be noted that these values are critical in specifying the particular cluster size.[3] Then, we apply the BFS (Breadth First Search) algorithm on the above navigational graph (procedure *TraverseWithBFS*). Specifically, BFS takes a graph and a node in the graph known as the source, and visits each node that can be reached from the source by traversing the edges. In doing so, it outputs a sub-graph which consists of the nodes that can be reached from the source. Note, that all the nodes which BFS traverses are marked. This procedure iterates until BFS has been traversed all the nodes of the initial graph (at the last iteration no-one node of the initial graph remains unmarked). The pseudocode of BFS is depicted in Fig. 3. The nodes in each connected sub-graph in the remaining graph constitute a Web page cluster. Thus, each client group has a separate number of clusters.

In view of this, a benefit of the clustWeb (in comparison with the hierarchical and partitional clustering schemes) is that it exploits the users' access patterns (Web proxy log file), with no need to determine the number of clusters in advance either randomly (naive approach) or by heuristic techniques. In clustWeb, the number of clusters is dynamically estimated by the confidence and support measures. Specifically, the clustWeb is not guided by the number of clusters but it is rather guided by how strong the connectivity among Web pages is in the Web navigational graph.

Concerning the complexity of the proposed scheme, it should be noted that the graph partitioning problem is, generally, an NP complete problem [10]. The clustWeb is an heuristic approach where its complexity is affected by the BFS complexity. Specifically, as it is depicted in Fig. 3, the BFS has to consider all paths to all possible nodes of the graph. Thus, the time complexity of BFS is $O(|u| + |v|)$ where $|u|$ is the number of nodes and $|v|$ the number of edges in the graph [5]. In clustWeb, the BFS, in the worst case, would be repeated $K|u|$ times, where K is the number of client groups.

3.3. The clustPref scheme

Here, we describe how the proposed clustering scheme can be adapted in a Web prefetching scheme, the so-called *clustPref*. Fig. 5 depicts the proposed prefetching scheme. In particular, the following steps are taken place:

Step 1: A Web user requests an object in a particular time.
Step 2: The proxy identifies the Web user, according to its IP address, and assigns it to one of the client groups. In that framework, given that the clusters of Web objects are known, the proposed prefetching

[3] The total size of objects in a cluster should not exceed the total cache size.

```
clustWeb Algorithm
Input Parameters: Confidence, Support, G(i)=navigational graph, K=number of client groups;
Begin
for i=1 to K
{
CutWithConfidence(G(i),Confidence);
CutWithSupport(G(i),Support);
TraverseWithBFS(G(i));
}
End
Procedure CutWithConfidence(G(i),Confidence)
{
for j=1 to num-of-edges
if (edge[j]<Confidence) remove edge[j];
}
Procedure CutWithSupport(G(i),Support)
{
for j=1 to num-of-edges
if (edge[j]<Support) remove edge[j];
}
Procedure TraverseWithBFS(G(i))
{
int cl[i]=0;
Repeat
{
C[cl[i]]=BFS(G(i)); //C: a list array of traversed nodes
cl[i]++;
} until (all nodes of G(i) are traverersed);
}
```

Fig. 4. The clustWeb algorithm.

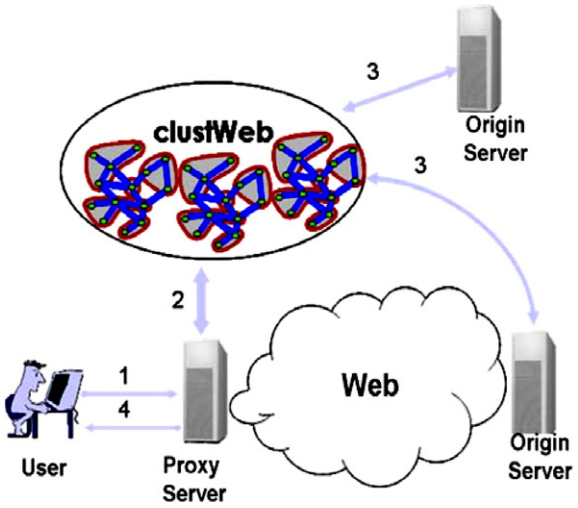

Fig. 5. The ClustPref scheme.

scheme works as follows: The proxy searches inside the existing clusters to find in which of them the requested Web object exists.

Step 3: The proxy prefetches from the origin servers all the objects that exist in the cluster which is selected in the previous step. Then, these objects are managed by the proxy's cache replacement policy.
Step 4: The proxy sends the requested object to user.

From the above, it is obviously, that the clustPref scheme is strongly dependent on the content of clusters. If each cluster contains a large amount of objects, it will result in an overaggressive prefetching policy. On the contrary, if each cluster contains only a few objects, then the clustPref will not have a significant effect on the performance of the Web caching environment. Regarding it, the most critical issue is the selection of support and confidence values, which determine the quality of clusters as well.

Another issue that should be addressed is the cache consistency, which tackles the problem of staleness in cached objects. In this context, considering a study [20], which showed that the probability of requesting a stale object is very small, we assume that we have strong consistency (accessed objects are always up to date).

4. Performance evaluation

In this section, we study the performance of the clustWeb in the proposed integrated environment.

4.1. Examined cache replacement policies

Whenever the cache is full and the proxy needs to cache a new object, it has to decide which object to evict from the cache to accommodate the new object. As we have seen above, the policy used for the eviction decision is referred to as the *replacement policy*. Here, we evaluate the gains of the proposed integration approach (clustPref) comparing with cache replacement policies. It should be noted that we do not compare the clustPref approach with other prefetching schemes (such as PPM) since it is beyond the scope of this paper. Our objective is to show how the performance of the proxy cache could be improved by enhancing a clustering-based short-term prefetching scheme. Furthermore, we do not compare the efficiency of the clustWeb algorithm with other clustering schemes because the existing graph partitioning tools (such as METIS [14,27]) require the number of clusters to be given in advance. However, such an approach cannot be applied on our scheme since there is not knowledge about how many clusters of Web pages exist. Therefore, clustWeb algorithm is used as an indicative one in order to test the proposed integrating scheme. In this framework, we examine the most indicative and popular policies that have been adopted by the majority of Web cache environments as well as some hybrid policies:

- *LRU:* It evicts from the cache the least recently referenced object. LRU has been applied in several proxy caching servers, such as Squid [25].
- *LRU-Prefetching:* It is the standard LRU, when it is adapted with the clustPref scheme.
- *LFU:* It removes from the cache the least frequency requested object [25].
- *LFU-Prefetching:* It is the standard LFU, when it is adapted with the clustPref scheme.
- *FRES-CAR:* It identifies the objects that should be evicted by considering together three important criteria: object's frequency, recency and size [23].
- *FRES-CAR-Prefetching:* It is the FRES-CAR, when it is adapted with the clustPref scheme.
- *CRF:* The CRF's decisions for replacement are based on a combination of the recency and frequency criteria [15].
- *CRF-Prefetching:* It is the CRF, when it is adapted with the clustPref scheme.
- *SIZE:* The large objects are evicted first from the cache [25].
- *SIZE-Prefetching:* It is the standard SIZE, when it is adapted with the clustPref scheme.

4.2. Experimental results

The algorithms are simulated and experimented under a real workload of Web cache traces provided by a major (Squid) proxy cache server installation. The Squid proxy server has been installed in many academic institutions such as the Aristotle University (AUTH) and it is one of the top proxy servers in the Greek

Universities. Traces collected by the proxy servers, referred to as proxy logs or proxy traces, contain information about Web document accesses by many users against many Web sites. In our experiments, the proxy traces refer to the period from February 2001, regarding a total of almost 2,000,000 requests, of more than 26 GB (dynamic and static) content. Table 2 summarizes the details of this data set.

In order to create the clusters, we select the first (ordering by time) of the 70% of the total requests as training data set and the rest as testing data set for testing the proposed approach. Regarding the size of the cache, it is expressed in terms of the percentage of the total number of bytes of all objects in a Web log file [31]. In our experiments, we consider that the default value of cache size is defined as the 1.5% of bytes of all objects in a Web proxy log.

4.2.1. Performance metrics

In order to evaluate the proposed scheme, we use two performance rates:

- *Hit Rate (HR):* is the percentage of the number of requests that are served by the cache over the total number of requests;
- *Byte Hit Rate (BHR):* is the percentage of the number of bytes that correspond to the requests served by the cache over the total number of bytes requested.

A high HR indicates the user's satisfaction and defines an increased user servicing. On the other hand, a high BHR improves the network performance and reduces the user-perceived latency (i.e., bandwidth savings, low congestion etc.).

4.2.2. Impact of confidence and support thresholds

We tested the competing algorithms with varying the confidence and support thresholds. In our experiments we tested a wide range of confidence and support values. The results are reported in Tables 3–10. Considering that all the prefetching algorithms are imperfect, since fetch some content that is not requested by the users, results in high bandwidth usage. Thus, we expect that the prefetching schemes would have lower BHR than the caching schemes. To our surprise, Tables 7–10 depict that the clustPref presents in the most cases better performance with respect to BHR. On the other hand, the integration of caching and prefetching does not improve significantly the HR. In view of Tables 3–6, we observe that the prefetching algorithms exhibit quite similar performance with respect to HR when they are compared with the caching schemes.

Table 2
Data set details

Data set	Time period	Number of requests
Access logs data from the proxy server of AUTH	2/4/2001–2/14/2001	2,000,000

Table 3
A comparison of hit rates for support = 2

Policies	Sup = 2	Sup = 2	Sup = 2	Sup = 2
	Conf = 0.1	Conf = 0.3	Conf = 0.6	Conf = 0.9
LRU (%)	41.5	41.5	41.5	41.5
LRU-Prefetching (%)	42.5	42.5	42.5	42.5
LFU (%)	40.3	40.3	40.3	40.3
LFU-Prefetching (%)	41.3	41.3	41.3	40.4
FRES-CAR (%)	43.7	43.7	43.7	43.7
FRES-CAR Prefetching (%)	43.8	42.8	42.5	42
CRF (%)	9.4	9.4	9.4	9.4
CRF-Prefetching (%)	14	14	14	14
SIZE (%)	49.7	49.7	49.7	49.7
SIZE-Prefetching (%)	49.8	49.8	49.2	49.2

Table 4
A comparison of hit rates for support = 4

Policies	Sup = 4	Sup = 4	Sup = 4	Sup = 4
	Conf = 0.1	Conf = 0.3	Conf = 0.6	Conf = 0.9
LRU (%)	41.5	41.5	41.5	41.5
LRU-Prefetching (%)	41.6	41.6	41.6	41.6
LFU (%)	40.3	40.3	40.3	40.3
LFU-Prefetching (%)	40.3	40.3	40.3	40.3
FRES-CAR (%)	43.7	43.7	43.7	43.7
FRES-CAR Prefetching (%)	42.4	42.4	42	42
CRF (%)	9.4	9.4	9.4	9.4
CRF-Prefetching (%)	13	11	13	9.4
SIZE (%)	49.7	49.7	49.7	49.7
SIZE-Prefetching (%)	49.3	49.3	49.3	49.3

Table 5
A comparison of hit rates for support = 6

Policies	Sup = 6	Sup = 6	Sup = 6	Sup = 6
	Conf = 0.1	Conf = 0.3	Conf = 0.6	Conf = 0.9
LRU (%)	41.5	41.5	41.5	41.5
LRU-Prefetching (%)	41.6	41.6	41.6	41.6
LFU (%)	40.3	40.3	40.3	40.3
LFU-Prefetching (%)	40.3	40.3	39.8	39.8
FRES-CAR (%)	43.7	43.7	43.7	43.7
FRES-CAR Prefetching (%)	42.4	42.4	42.4	42.4
CRF (%)	9.4	9.4	9.4	9.4
CRF-Prefetching (%)	11.3	10.4	10.4	9.5
SIZE (%)	49.7	49.7	49.7	49.7
SIZE-Prefetching (%)	49.3	49.3	48.8	48.8

Table 6
A comparison of hit rates for support = 8

Policies	Sup = 8	Sup = 8	Sup = 8	Sup = 8
	Conf = 0.1	Conf = 0.3	Conf = 0.6	Conf = 0.9
LRU (%)	41.5	41.5	41.5	41.5
LRU-Prefetching (%)	41.6	41.6	41.6	41.6
LFU (%)	40.3	40.3	40.3	40.3
LFU-Prefetching (%)	40.4	40.4	40.4	40.4
FRES-CAR (%)	43.7	43.7	43.7	43.7
FRES-CAR Prefetching (%)	42.4	42.4	42.4	42.4
CRF (%)	9.4	9.4	9.4	9.4
CRF-Prefetching (%)	10.4	9.8	9.8	9
SIZE (%)	49.7	49.7	49.7	49.7
SIZE-Prefetching (%)	48.8	48.8	48.8	48.8

Regarding the HR, as the confidence threshold is increasing, we observe a decrease of it for some cache replacement algorithms. This can be explained by the fact that as long as the confidence threshold increases, more clusters are being created for every single client group and thus, each cluster contains a very small number of Web pages. In the same context, FRES-CAR-prefetching shows better performance than the FRES-CAR (with respect to HR) for low values of confidence and support (Table 3). This is due to the fact that the FRES-CAR partitions the cache in several segments and thus avoids overflowing the cache with large-scale objects. Therefore, high values of confidence create large clusters which cannot be managed by

Table 7
A comparison of byte hit rates for support = 2

Policies	Sup = 2	Sup = 2	Sup = 2	Sup = 2
	Conf = 0.1	Conf = 0.3	Conf = 0.6	Conf = 0.9
LRU (%)	20.8	20.8	20.8	20.8
LRU-Prefetching (%)	21.6	22.8	22.8	22.8
LFU (%)	23.1	23.1	23.1	23.1
LFU-Prefetching (%)	23.9	23.9	23.9	23.9
FRES-CAR (%)	13.9	13.9	13.9	13.9
FRES-CAR Prefetching (%)	13.7	13.7	13.9	13.9
CRF (%)	5.8	5.8	5.8	5.8
CRF-Prefetching (%)	7	7.2	7.2	6.8
SIZE (%)	11.1	11.1	11.1	11.1
SIZE-Prefetching (%)	11.4	11.7	12.1	12.1

Table 8
A comparison of byte hit rates for support = 4

Policies	Sup = 4	Sup = 4	Sup = 4	Sup = 4
	Conf = 0.1	Conf = 0.3	Conf = 0.6	Conf = 0.9
LRU (%)	20.8	20.8	20.8	20.8
LRU-Prefetching (%)	21.6	21.6	21.6	21.6
LFU (%)	23.1	23.1	23.1	23.1
LFU-Prefetching (%)	23.9	24.4	24.6	24.6
FRES-CAR (%)	13.9	13.9	13.9	13.9
FRES-CAR Prefetching (%)	13.9	13.9	13.9	13.9
CRF (%)	5.8	5.8	5.8	5.8
CRF-Prefetching (%)	7.2	6.8	7.2	6.4
SIZE (%)	11.1	11.1	11.1	11.1
SIZE-Prefetching (%)	11.6	11.6	11.9	11.6

Table 9
A comparison of byte hit rates for support = 6

Policies	Sup = 6	Sup = 6	Sup = 6	Sup = 6
	Conf = 0.1	Conf = 0.3	Conf = 0.6	Conf = 0.9
LRU (%)	20.8	20.8	20.8	20.8
LRU-Prefetching (%)	21.6	22	22	22
LFU (%)	23.1	23.1	23.1	23.1
LFU-Prefetching (%)	23.9	23.9	23.9	23.9
FRES-CAR (%)	13.9	13.9	13.9	13.9
FRES-CAR Prefetching (%)	13.7	13.7	13.7	13.7
CRF (%)	5.8	5.8	5.8	5.8
CRF-Prefetching (%)	6.8	7.2	7	6.8
SIZE (%)	11.1	11.1	11.1	11.1
SIZE-Prefetching (%)	11.5	11.5	11.5	11.5

FRES-CAR. Concerning the BHR, we observe that LRU-prefetching, LFU-prefetching, CRF-prefetching and SIZE-prefetching outperform the LRU, LFU, CRF and SIZE, respectively, (consistently around 3–22% absolute improvement), regardless of the values of confidence and support (Tables 7–10).

As far as concerned the value of support threshold, as depicted from the Tables 3–10, we observe that the different values of it (while the confidence threshold remains stable) do not impact significantly the performance of the proxy cache. Although our experiments show that the behavior of clustPref is not very sensitive on the value of support, we include both the confidence and the support as indicators of the connectivity among two pages. The reason is that the distribution of page references on the Web has been proven that

Table 10
A comparison of byte hit rates for support = 8

Policies	Sup = 8	Sup = 8	Sup = 8	Sup = 8
	Conf = 0.1	Conf = 0.3	Conf = 0.6	Conf = 0.9
LRU (%)	20.8	20.8	20.8	20.8
LRU-Prefetching (%)	21.6	21.6	21.6	21.6
LFU (%)	23.1	23.1	23.1	23.1
LFU-Prefetching (%)	23.9	24.2	24.2	24.2
FRES-CAR (%)	13.9	13.9	13.9	13.9
FRES-CAR Prefetching (%)	13.7	13.7	13.7	13.7
CRF (%)	5.8	5.8	5.8	5.8
CRF-Prefetching (%)	6.8	6.8	6.6	6.2
SIZE (%)	11.1	11.1	11.1	11.1
SIZE-Prefetching (%)	11.3	11.3	11.5	11.5

it is highly skewed [8]. Thus, if we have used only the support will result in a large number of pages to be clustered together by some popular Web pages erroneously. For instance, it is assumed that the average number of accesses to one Web page for a not popular Web site is three times per day and a Web page from a popular Web site can actually have more than 10,000 accesses. In such a case, large values of support for most pages, which are linked to a popular Web page, will be obtained, affecting both the number and the content of clusters.

Summarizing the above results, we make the following remarks:

- Efficient clusters lead to an effective prefetching policy.
- The prefetching scheme, clustPref, improves significantly the network performance since it achieves higher BHR than the other approaches.
- The clustPref is a realistic scheme, which can be adapted easily to a Web cache environment, bridging the performance gap between BHR and the percentage of the requests satisfied by the proxy cache.
- The number of clusters does not affect in most cases the efficiency of the proposed prefetching scheme, since the prefetching policies present quite similar performance regardless of the values of confidence and support.
- Our approach is characterized by its adaptiveness to changes in the Web users' patterns, which are rather natural in the Web. This is due to the fact that the proposed scheme is parametric with respect to the Web data clusters, which can be recomputed periodically in order to keep track of the recent past.

5. Conclusions—future work

In this paper, we addressed the short-term prefetching problem on a Web cache environment using an algorithm (clustWeb) for clustering inter-site Web pages. The proposed scheme efficiently integrates Web caching and prefetching. According to this scheme, each time a user requests an object, the proxy fetches all the objects which are in the same cluster with the requested object. Specifically, the proxy traces are represented by a Web navigational graph. Then, the clusters have been resulted by partitioning this graph, where the number of clusters is not determined at priori but it is dynamically estimated by the confidence and support measures. Using real data, we show the robustness and efficiency of the proposed method. It should be noticed that the proposed scheme may also have a number of practical applications in information management and e-commerce (recommending new products to Web site visitors [2]).

For the future, we plan to further investigate the efficiency of the clustWeb to a wide range of applications, such as discovering usage patterns and profiles, detecting copyright violations, and reporting search results. Furthermore, another future work is to compare the proposed prefetching scheme with other clustering algorithms [14,27]. As we referred above, the clustWeb algorithm is used as an indicative one in order to test the proposed integrating scheme. Finally, research efforts are underway towards extending the clustPref scheme under a Content Distribution Network (CDN).

Acknowledgements

The authors appreciate and thank the anonymous reviewers for their valuable comments and suggestions, which have considerably contributed in improving the paper's content, organization and readability.

References

[1] Ari I, Hong B, Miller EL, Brandt SA, Long DDE. Managing flash crowds on the Internet. In: Proceedings of the MASCOTS 2003. Orlanto, USA: IEEE Press; 2003. p. 246–9.
[2] Berendt B, Spiliopoulou M. Analysing navigation behaviour in web sites integrating multiple information systems. VLDB J Special Issue on Databases and the Web 2000;9(1):56–75.
[3] Chen X, Zhang X. Popularity-based PPM: an effective web prefetching technique for high accuracy and low storage. In: Proceedings of the international conference on parallel processing. Canada, Vancouver; 2002.
[4] Chen Y, Qiu L, Chen W, Nguyen L, Katz RH. Efficient and adaptive Web replication using content clustering. IEEE J Selected Areas Commun 2003;21(6):979–94.
[5] Cormen TH, Leiserson E, Rivest R, Stein C. Introduction to algorithms. 2nd ed. MIT Press and McGraw-Hill; 2001, ISBN 0-262-03293-7.
[6] Deshpande M, Karypis G. Selective Markov models for predicting Web-page accesses. In: Proceedings of the 1st SIAM international conference on data mining. Chicago, USA; 2001.
[7] Drakatos S, Pissinou N, Makki K, Douligeris C. A context-aware prefetching strategy for mobile computing environments. In: Proceedings of the 2006 international conference on communications and mobile computing. Vancouver, British Columbia, Canada: ACM Press; 2006. p. 1109–16.
[8] Faloutsos C, Faloutsos P, Faloutsos M. On power-law relationships of the Internet topology. In: Proceedings of the ACM SIGCOMM conference on network architectures and protocols. Cambridge, USA; 1999.
[9] Fisher D, Saksena G. Link prefetching in Mozilla: a server-driven approach. In: Proceedings of the WCW; 2003.
[10] Garey M, Johnson D. Computers and intractability – a guide to the theory of NP-completeness. W.H. Freeman; 1979.
[11] Jiang Y, Wu M, Shu W. Web prefetching: costs, benefits and performance. In: Proceedings of the 7th international workshop on web content caching and distribution (WCW2002). Boulder, Colorado; 2002.
[12] Jiang N, Gruenwald L. Research issues in data stream association rule mining. SIGMOD Record 2006;35(1):14–9.
[13] Jung J, Lee D, Chon K. Proactive Web caching with cumulative prefetching for large multimedia data. Comput Networks 2000; 33(1–6):645–55.
[14] Karypis G. Cluto: software for clustering high dimensional data sets. <www.cs.umn.edu-karypis>, 2005.
[15] Katsaros D, Manolopoulos Y. Caching in Web memory hierarchies. In: Proceedings of the ACM symposium on applied computing (ACM SAC). Nicosia, Cyprus: ACM Press; 2004. p. 1109–13.
[16] Kroeger TM, Long DDE, Mogul JM. Exploring the bounds of web latency reduction from caching and prefetching. In Proceedings of the USENIX symposium on Internet technologies and systems. Monterey, California, USA; 1997.
[17] Kuenning GH, Popek GJ. Automated hoarding for mobile computers. In: Proceedings of the 16th ACM symposium on operating system principles. Malo, France; October 1997. p. 264–75.
[18] Lou W, Liu G, Lu H, Yang Q. Cut-and-pick transactions for proxy log mining. In: Proceedings of the 8th international conference on extending database technology (EDBT 2002). Prague, Czech Republic; 2002. p. 88–105.
[19] Padmanabhan V, Mogul JC. Using predictive prefetching to improve World Wide Web latency. ACM SIGCOMM Comput Commun Rev 1996;26(3):22–36.
[20] Padmanabhan V, Qiu L. The content and access dynamics of a busy Web site: findings and implications. In: Proceedings of the ACM SIGCOMM conference on applications, technologies, architectures, and protocols for computer communication. Stockholm, Sweden; 2000. p. 111–23.
[21] Pallis G, Vakali A. Insight and perspectives for content delivery networks. Commun ACM (CACM) 2006;49(1):101–6.
[22] Pallis G, Angelis L, Vakali A. Validation and interpretation of Web users' sessions clusters. Information Processing and Management 2007;43(5):1348–67.
[23] Pallis G, Vakali A, Sidiropoulos E. FRES-CAR: An adaptive cache replacement policy. In: Proceedings of the 1st IEEE international workshop on challenges in Web information retrieval and integration (WIRI'05) in cooperation with the 21st IEEE conference on data engineering ICDE 2005. Tokyo, Japan; 2005.
[24] Palpanas T, Mendelzon A. Web prefetching using partial match prediction. In: Proceedings of the 4th international web caching workshop; 1999.
[25] Podlipnig S, Boszormenyi L. A survey of Web cache replacement strategies. ACM Comput Surveys 2003;35(4):374–98.
[26] Rabinovich M, Spatsheck O. Web caching and replication. Addison Wesley; 2002.
[27] Schloegel K, Karypis G, Kumar V. Parallel multilevel algorithms for multi-constraint graph partitioning. In: Proceedings of 6th international Euro-Par conference. September 2000. p. 296–310.
[28] Teng WG, Chang CY, Chen MS. Integrating Web caching and Web prefetching in client-side proxies. IEEE Trans Parallel Distributed Syst 2005;16(5):444–55.
[29] Vakali A, Pokorny J, Dalamagas T. An overview of Web data clustering practices. In: Proceedings of the EDBT Workshops 2004. Heraklion, Crete; 2004. p. 597–606.

[30] Venkataramani A, Yalagandula P, Kokku R, Sharif S, Dahlin M. The potential costs and benefits of long term prefetching for content distribution. Comput Commun 2002;25(4):367–75.
[31] Yang Q, Zhang HH. Web-log mining for predictive web caching. IEEE Trans Knowledge Data Eng 2003;15(4):1050–3.
[32] Yang Q, Zhang H. Integrating Web prefetching and caching using prediction models. World Wide Web 2001;4(4):299–321.
[33] Xing D, Shen J. Efficient data mining for web navigation patterns. Inform Software Technol 2004;46(1):55–63.

George Pallis received his B.Sc. and Ph.D. degree in Department of Informatics of Aristotle University of Thessaloniki (Greece). His current research interests include Web data caching, content distribution networks, and Web data clustering. He is co-editor of the book ''Web Data Management Practices: Emerging Techniques and Technologies'' published by Idea Group Publishing.

Athena Vakali received a B.Sc. degree in Mathematics from the Aristotle University of Thessaloniki, Greece, a M.Sc. degree in Computer Science from Purdue University, USA (with a Fulbright scholarship) and a Ph.D. degree in Computer Science from the Department of Informatics at the Aristotle University of Thessaloniki. Since 2002, she is Assistant Professor of the Department of Informatics, Aristotle University of Thessaloniki, Greece. She is recently working on Web data management and she has focused on Web data caching, content delivery and Web data clustering. She has published over 90 papers in international journals and conferences. She is co-editor of the book ''Web Data Management Practices: Emerging Techniques and Technologies'' published by Idea Group Publishing.

Jaroslav Pokorny received the Ph.D. degree in theoretical cybernetics from the Charles University, Prague, Czechoslovakia, in 1984. Currently he is a professor of computer science at the Faculty of Mathematics and Physics at Charles University and the head of its Department of Software Engineering. J. Pokorny has published more than 200 papers and books on data modelling, relational databases, query languages, file organization, XML. His research interests include also database design, information retrieval, and Semantic Web. He is a member of ACM and IEEE.