An Effective Clustering-based Prefetching Scheme for Spatial Databases System

Huichao Mi and Yongqiang Yang
College of Computer and Information Engineering
Henan University of Economic and laws
Zhengzhou, Henan, China
mmhhcc@126.com; yyQndsc@sina.com

Abstract—**Because of the characteristics of web spatial databases, restrictions on the applications of web spatial databases focus on accessing speed. The main solution for improving accessing speed is caching and prefetching. The existing prefetching schemes prefetch future spatial objects based on the distances of spatial locations rather than accessing contents. For the improvement of the accuracy of prefetching, we propose a clustering algorithm-clustVoronoi, which based on the web user logs. And then, we apply clustVoronoi to the prefetching scheme, named clustPref. This prefetching scheme can automatically determine the number of clusters and give the real case of spatial databases accessing. Through a simulation environment, using a real data set, we draw a conclusion that this prefetch scheme can effectively improve the spatial databases accessing performance.**

Keywords-spatial databases; clustVoronoi; clustPref; prefetching

I. INTRODUCTION

Advances in internet technology promote the development of the spatial information services. Web is the main way to get spatial information. On the contrary, because of the network bandwidth limitations, users spend long time to access spatial databases. We always use two methods to reduce the waiting time mainly: caching and prefetching.

A. The Web caching approach

Caching proved itself as an important technique to optimize the way the Web is used [1]. 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, 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 [2] (identify the objects to be replaced in a cache upon a request arrival) which has been enhanced by the underlying proxy server.

B. The Web prefetching approach

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 benefit 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 [4]. From the above it occurs that caching and prefetching complement each other in order to reduce the notice-able response time perceived by users [5]. 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 spatial navigational graph (NG).

978-0-7695-4705-3/12 $26.00 © 2012 IEEE
DOI 10.1109/ICICSE.2012.48

C. Voronoi Diagram

After spatial databases are applied widely, many researchers have focused on spatial analysis. They analyze spatial data to explain the space distribution laws and forecast the spatial development trend.

Because Voronoi diagram has integrated characteristics of vector model and grid model, it has many features: influence region, lateral adjacency, local dynamization, and so on [10]. Voronoi diagram transforms grid computing into Graph Theory problems, which closes to the humans' habits of spatial cognition. So we can apply Voronoi diagram to establish the spatial databases model.

The main idea of Voronoi diagram is to first partition a large spatial databases into smaller manageable regions. We achieve these by generating a first-order network Voronoi diagram over the points of interest [6]. A Voronoi diagram divides a space into disjoint polygons where the nearest neighbor of any point inside a polygon is the generator of the polygon. Considering a set of limited number of points, called generator points, in the Euclidean plane (in general, generators can be any type of spatial object). We associate all locations in the plane to their closest generator(s). The set of locations assigned to each generator forms a region called Voronoi polygon or Voronoi cell (Figure 1), of that generator. The set of Voronoi polygons associated with all the generators is called the Voronoi diagram with respect to the generators set. A thorough discussion on regular and network Voronoi diagrams is presented in [7].

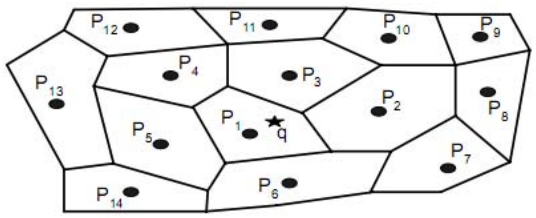

Figure 1. Voronoi polygons

II. Related work and papers

In this paper, we proposed a new prefetching scheme for Spatial Databases to improve the accessing rate. The present paper makes following contributions:

- We present an integrated approach which combines effectively caching and prefetching. Specifically, caching and prefetching can complement each other since the accessing for the spatial Databases in a short-time has the spatial locality.
- We divide the graphs into disjoint polygons based on Voronoi diagram. Most applications of the spatial databases are queries, and Voronoi can divide the space into pieces very well, which can response the query requests quickly. Given a series of special nodes, we divide the map into disjoint polygons (Voronoi polygons).
- We introduce an algorithm for clustering the map based on NG (navigational graph). The Web proxy access log is a sequential file with one user access record per line. Considering that each Internet Service Provider has a proxy server cache, the Web proxy log files provide information about activities performed by a user from the moment the user logs in to the Internet Service Provider to the moment the same user logs out from it. According to Web proxy access log， we divide the Voronoi polygons into different clusters based on the sequence and the frequency of accessing Voronoi polygons. This approach clusters the map based on the real accessing histories rather than the distances of points in the spatial databases.
- Finally, we develop a simulation environment to test the efficiency of the proposed integrated scheme. Specifically, we provide a clustering prefetching scheme, 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 Voronoi polygons which are in the same cluster with the requested object. Using real data, we show the robustness and efficiency of the proposed method.

III. Clustering algorithm for 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.

A. Spatial Navigational graphs

Firstly, we divide the graph into disjoint polygons, called Voronoi polygons where the nearest neighbor of any point inside a polygon is the generator of the polygons [6]. And then, the clusters have been resulted by partitioning the spatial navigational graph.

We get user logs from each client group (we group the users based on their domains). The spatial data requests of each client group are represented by a weighted directed Web graph G (p, v), where each node p represents a Voronoi polygon and each edge v represents a set of users' transitions from one Voronoi polygon 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 Figure 2.

However, due to the vast amount of Voronoi polygons, the resulting Web graphs, which have been occurred by the users' access patterns, become frequently incomprehensible and unmanageable. Therefore, we apply some data mining technologies to create sub-graphs. In this work, the edges are filtered out by their weight. More specifically, we remove the edges where the connectivity between two polygons is lower than a pre-specified threshold, whereas the connectivity between two polygons is determined by two parameters: support and confidence.

Definition: Let v: <p_i, p_j> be an edge from node p_i to node

p_j, the support of G, denoted by freq (p_i, p_j), is defined as the frequency of navigation steps between p_i to p_j. The confidence of G is defined as $\frac{freq(p_i, p_j)}{pop(p_i)}$, where pop ($p_i$) is the popularity of p_i.

According to this definition, the support of the edge <p2, p3> in the client group 1 in Figure 2 is $freq(p2, p3) = 2$ and the confidence value is $\frac{freq(p2, p_3)}{pop(p2)} = 0.4$, since $pop(p_2) = 5$.

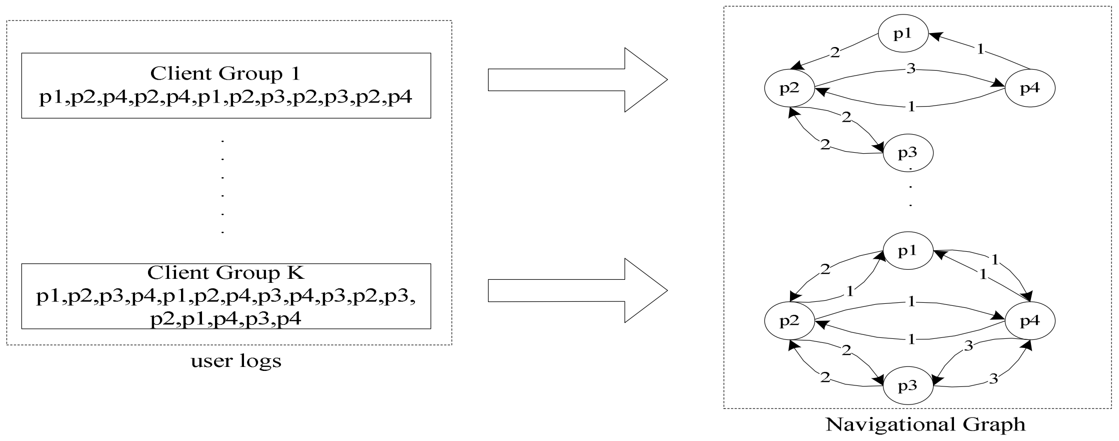

Figure 2. The Spatial navigational graph

We must select suitable support and confidence thresholds for better NG. In particular, if the support threshold is low, the resulted sub-graphs may include too many insignificant transitions between Voronoi polygons. If the support threshold is high, we may miss many interesting transitions. The confidence value is also important for capturing transitions that are strongly related with each other. Higher is the confidence value, stronger the relations of Voronoi polygons are.

B. *The clustering algorithm*

Here, we describe an algorithm in order to cluster the Voronoi polygons. We use a weighted directed Web graphs G(p, v) which represents the access patterns of a client group. We first partition the graph into sub-graph 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 algorithm is described in pseudo code in Figure 3. The input to this algorithm of the following information: the Web navigational graph, the number of client groups, a confidence threshold, and a support threshold. All edges with support of confidence value less than 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. We apply the BFS 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 pseudo code of BFS is depicted in Figure 4.The nodes in each connected sub-graph in the remaining graph constitute a Voronoi polygons cluster. Thus, each client group has a separate number of clusters.

The algorithm, called clustVoronoi, exploits the users' access patterns (Web proxy log file), with no need to determine the number of clusters in advance either randomly or by heuristic techniques. The number of clusters is dynamically estimated by the confidence and support measures.

```
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] <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 traversed);
}
```

Figure 3. The clustVoronoi algorithm

```
BFS  Algorithm
Input: G(P, v);
Unmark all nodes $p_i$;
Choose some starting node q;
Mark q;
List L=q;
Sub-graph T=q;
While L not empty {
        Choose some node x from front of list;
        Visit x;
        For each unmarked neighbor y{
            Mark y;
            Add it to end of L;
            Add edge xy to T;
        }
}
```

Figure 4. The BFS algorithm

C. *The clustering-based prefetching scheme*

In this paragraph, we describe how to adapt the clustering algorithm in a Web prefetching scheme, called clustPref. The following steps are taken place (they are described in Figure 5):

- Step 1: A user requests an object in a 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 scheme works as follows: The proxy searches inside the existing clustering 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.

Obviously, the clustPref scheme is strongly dependent on the content of clusters in the Voronoi diagram. 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, 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.

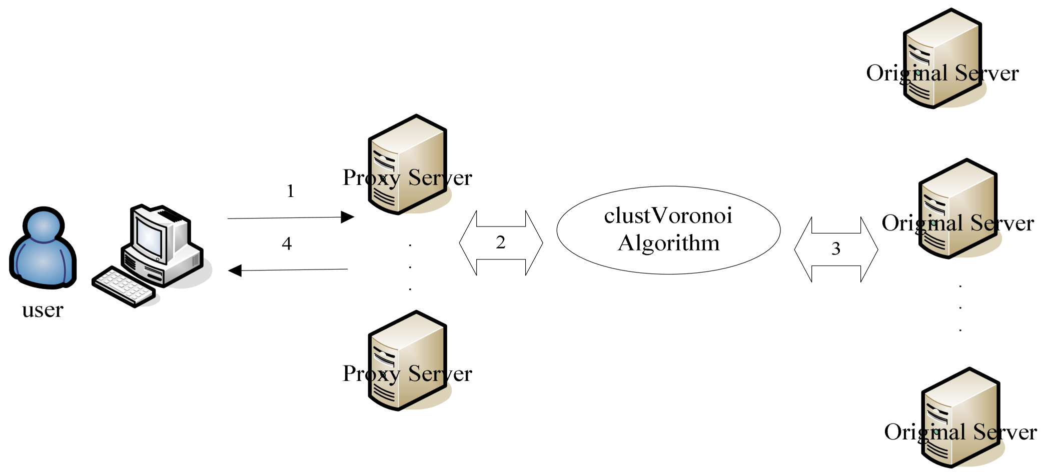

Figure 5. The clustPref scheme

IV. PERFORMANCE EVALUATION

We use the real road network, which has 21048 intersections and 21693 roads. We randomly select 0.4 percent of intersections as the generating set of Voronoi diagram. Considering the cache is always 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. Our objective is to show how the performance of the proxy cache could be improved by applying a clustering-based prefetching scheme. In this paper, 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 [8].
- 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 [8].
- LFU-Prefetching: It is the standard LFU, 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 [9].

- CRF-Prefetching: It is the CRF, when it is adapted with the clustPref scheme.
- SIZE: The large objects are evicted first from the cache [8].
- SIZE-Prefetching: It is the standard SIZE, when it is adapted with the clustPref scheme.

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. 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.

A. *Performance metrics*

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

- Hit Rate (HR): the percentage of the number of requests that are served by the cache over the total number of request.
- Byte Hit Rate (BHR): 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 users' satisfaction and defines an increased user servicing. On the contrary, a high BHR improves the network performance and reduces the user-perceived latency (i.e., bandwidth saving, low congestion etc).

B. *Experimental results*

We tested the competing algorithm with different confidence and support thresholds. The results are reported in Figure 6-11.

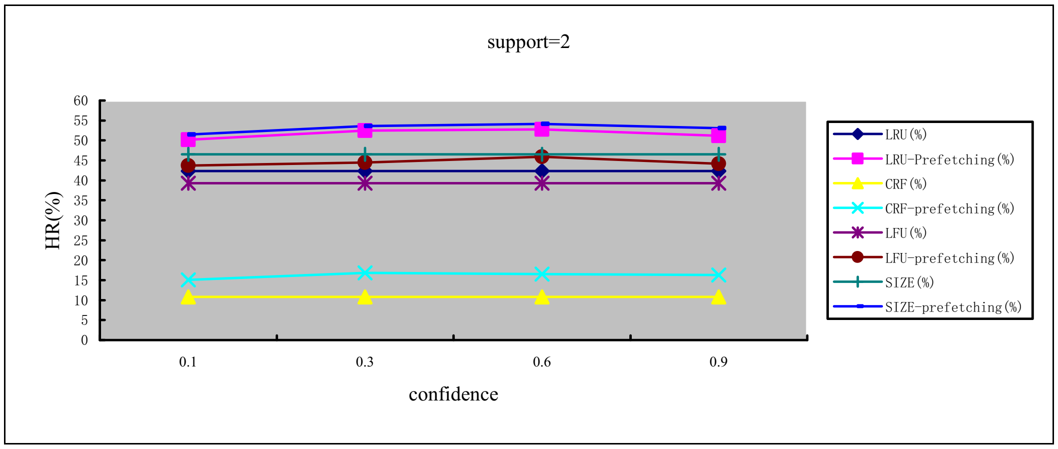

Figure 6. A comparison of hit rates for support=2

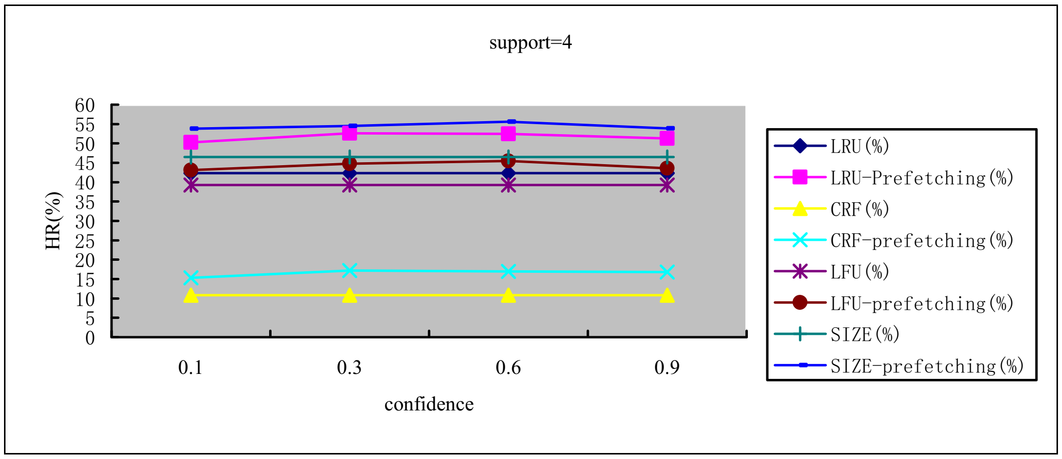

Figure 7. A comparison of hit rates for support=4

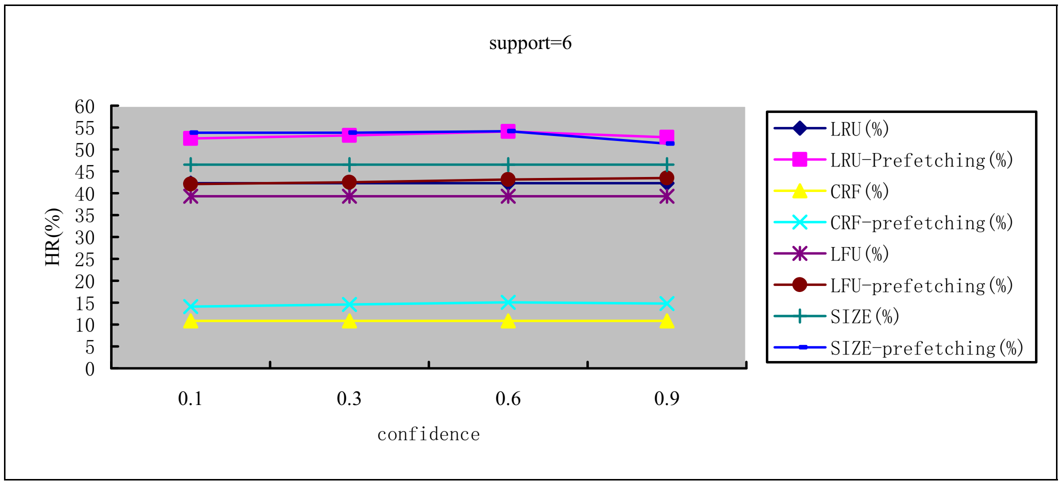

Figure 8. A comparison of hit rates for support=6

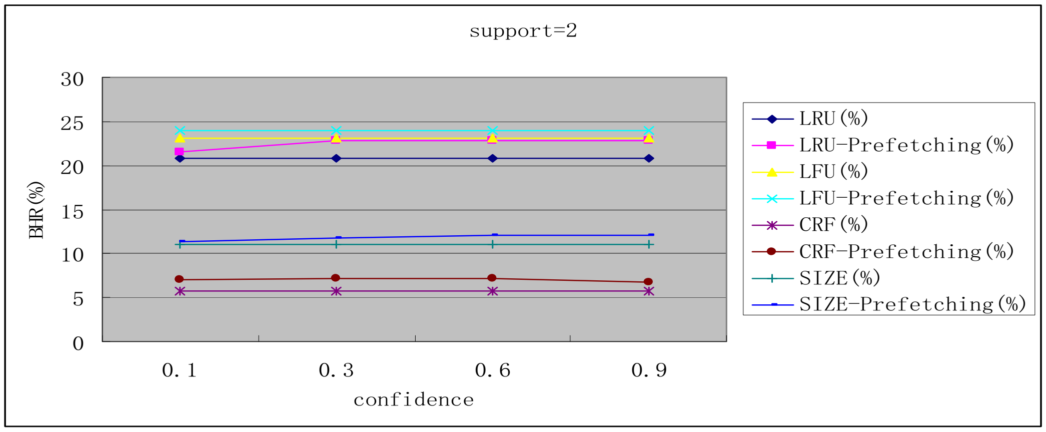

Figure 9. A comparison of byte hit rates for support=2

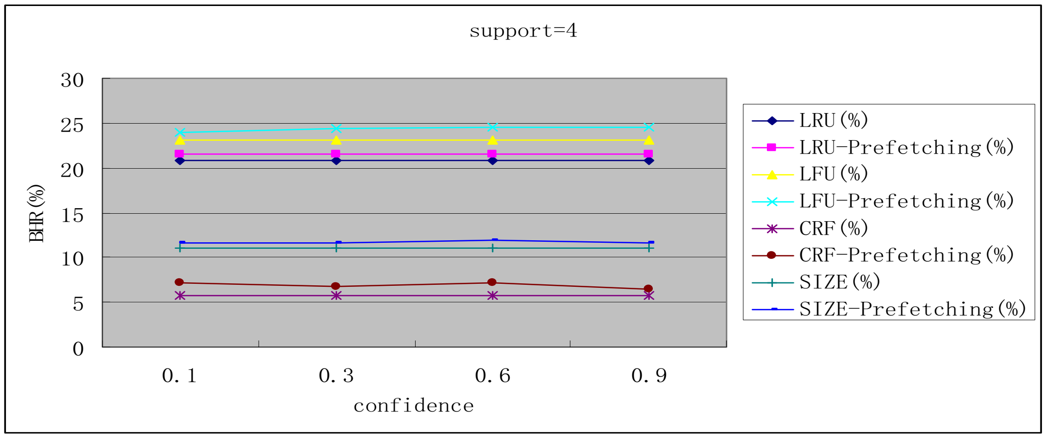

Figure 10. A comparison of byte hit rates for support=4

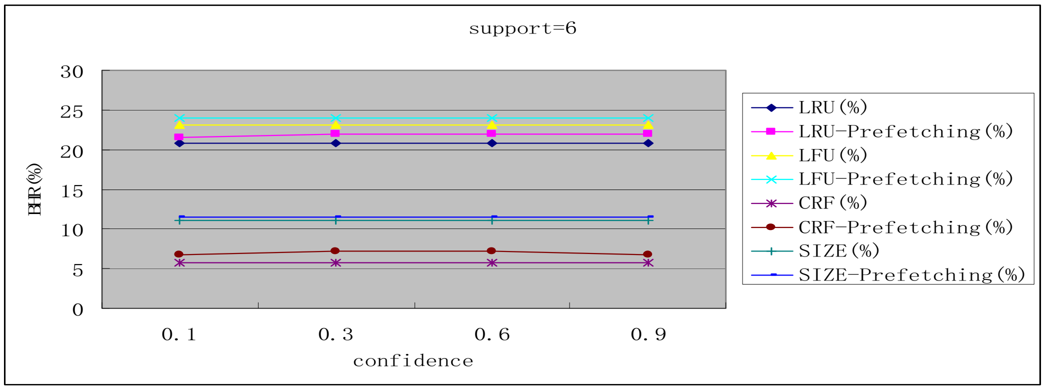

Figure 11. A comparison of byte hit rates for support=6

Regarding the HR, we observe that prefetching scheme can improve the hit rate respectively. At the same time, we observe a decrease of it for some cache replacement algorithm as the confidence threshold is increasing. The reason is that as long as the confidence threshold increases, many clusters that contain a very small number of Voronoi polygons are created in the same client group.

Considering the BHR, we observe that prefetching scheme outperformed others respectively (consistently around 3-22% absolute improvement), regardless of the values of confidence and support (Figure 9-11).

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 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.
- We can improve the performance of the proxy cache by applying prefetching scheme and improving caching scheme.

Our approach is characterized by its adaptiveness to changes in the 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 spatial data clusters, which can be recomputed periodically in order to keep track of the recent past.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a clustering algorithm (called clustVoronoi) for prefetching spatial data divided into Voronoi polygons, and then a prefetching scheme (clustPref) is adapted in the proxy cache. The proposed scheme efficiently integrates caching and prefetching. According to this scheme, the proxy fetches all of the objects in the same clusters when a user requests some object. These clusters are resulted by NG, where the number of clusters is dynamically determined by the confidence and support threshold. By the experiments, this scheme can improve significantly the network performance. But we found the network performance is impacted by the caching scheme too, so the improvement of caching scheme is important.

For the future, we plan to investigate the suitable caching scheme integrated with the clustPref scheme. Furthermore, another future work is to compare the proposed prefetching scheme based on clustVoronoi with others based on other clustering algorithms.

REFERENCES

[1] Rabinovich M and Spatsheck O, “Web caching and replication,” Addison Wesley, 2002.

[2] Podlipnig S and Boszormenyi L, “A survey of Web cache replacement strategies,” ACM Compute Surveys, 2003, 35(4): 374-98

[3] Jiang Y, Wu M and 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.

[4] Yang Q and Zhang H, “Integrating Web prefetching and caching using prediction models,” World Wide Web 2001,4(4): 299–321.

[5] Kroeger TM, Long DDE and 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.

[6] Kolahdouzan M and Shahabi C, “Voronoi-based K nearest neighbor search for spatial network databases,” Proc of the 30th VLDB Conf Toronto. San Francisco: Morgan Kaufmann,2004: 840-851

[7] A.Okabe, B.Boots, K.Sugihara, and S.N.Chiu, “Spatial Tessellations, Concepts and Applications of Voronoi Diagrams,” John Wiley and Sons Ltd., 2nd edition, 2000.

[8] Podlipnig S and Boszormenyi L, “A survey of Web cache replacement strategies,” ACM Comput Surveys 2003,35(4): 374–98.

[9] Katsaros D and Manolopoulos Y, “Caching in Web memory hierarchies,” In: Proceedings of the ACM symposium on applied computing (ACM SAC), Nicosia, Cyprus: ACM Press, 2004, pp. 1109–13.

[10] Gold C M, “The Meaning of Neighbour,” Lecture Notes in Coputing Science, 1992(39): 220-235