A PROXY BASED APPROCH TO VALID REGION EXTENSION IN SPATIAL QUERIES

Siniya majeed[1] , Sathishkumar.s[2]

[1]PG scholar,Maharaja Engineering College, Avinashi, India

Email: sinim.sinim@gmail.com

[2]Assistent Professor, Maharaja Engineering College, Avinashi, India

Email:sathishmcame@gmail.com[2]

Abstract— a proxy-based approach to continuous NN and window queries in mobile environments.The proxy takes advantage of spatial and temporal locality of spatial queries to create EVRs of NN and window queries. Different from prior work, we devised new EVR creation and extension algorithms for NN queries, enabling the proxy to build effective EVRs efficiently. The devised algorithms make the proxy achieve high performance even when the cache size is small. On the other hand, we propose to represent EVRs of window queries in the form of vectors, called estimated window vectors, to achieve larger estimated valid regions. Moreover, due to distinct characteristics,we introduce an EVR-tree and a grid index to process NN and window queries, respectively. The algorithms for mutual support of the EVR-tree and the grid index are developed to further enhance the system performance. The experimental results show that the proposed approach significantly outperforms the existing proxy-based approaches since our proposed algorithms create much larger EVRs for mobile clients Compared with the representative server-based approach,the experimental results indicate that the proposed proxy-based approach achieves similar performance even though the proxy has only partial information of data objects. Besides, the results reveal that the proposed proxybased approach is suitable in a densely populated area,

whereas the server-based approach is suitable when mobile clients move at high speeds. Although offering the above benefits, the inherent problem of data object updates needs further investigation for the proposed proxy-based approach.In future work, we will investigate the impact of data object updates on the proposed approach and extend the proposed approach to efficiently handle frequent object updates

Index Terms—Nearest neighbor query, window query, spatial query processing, location-based service, mobile computing

INTRODUCTION

LOCATION-BASED services (LBSs), also known as location-dependent information services (LDISs), have been recognized as an important context-aware application in pervasive computing environments [1]. Spatial queries are one of the most important LBSs. According to spatial constraints, spatial queries can be divided into several categories including nearest neighbor (NN) queries and window queries. An NN query is to find the nearest data object with respect to the location at which the query is issued (referred to as the query location of the NN query). For example, a user may launch an NN query like "show the nearest coffee shop with respect to my current location." On the other hand, a window query is to find all the objects within a specific window frame. An example window query is "show all restaurants in my car navigation window."

In general, a mobile client continuously launches spatial queries until the client obtains a satisfactory answer. For example, a query "show me the rate of the nearest hotel with respect to my current location" is continuously submitted in a moving car so as to find a desired hotel. The naive method answering continuous spatial queries is to submit a new query whenever the query location changes. The naive method is able t~~o provide correct results,~~ but it poses the following problems:

. High power consumption. The power consumption of a mobile device is high since the mobile device keeps submitting queries to the LBS server.

. Heavy server load. A continuous query usually consists of a number of queries to the LBS server, thereby increasing the load on the LBS server.

Fortunately, in the real world, the queries of a continuous query usually exhibit spatial locality. Thus, caching the query result and the corresponding valid region (VR) in the client-side cache was proposed in [2], [3] to mitigate the above problems. The valid region, also known as the valid scope, of a query is the region where the answer of the query remains valid. Subsequent queries can be avoided as long as the client is in the valid region. Note that a VR is associated with a query by definition. However, as pointed out in [4], by associating a VR with the corresponding object, the VR of an object p can be used to resolve all the queries whose answer object is p.

Although VRs are useful, an LBS server may not provideVRs in practice since the server may simply provide query results only or would not compute VRs under heavy load. In these situations, mobile service providers (e.g., Verizon Wireless and AT&T) or smartphone makers (e.g., Apple and RIM) can utilize a proxy architecture where the proxy provides estimated valid regions (EVRs), which are the subregions of the corresponding VRs, for the clients. With the proxy architecture, the clients still can enjoy the benefits of EVRs even if the LBS server does not provide VRs and thus the mobile service providers and smartphone makers can attract more clients. In other words, the proxy-based approach is an alternative solution to serve the function ofVRs in case that the LBS server could not provide VRs. In [5], [6], the authors proposed to deploy a proxy between mobile clients and the LBS server to build EVRs by exploiting the queries interested in the same objects. EVR provisioning is inspired by that, in addition to spatial locality, spatial queries also exhibit temporal locality, resulting from that a number of queries with close query locations are likely to be launched during a short interval. For instance, a large number of NN queries about nearest hotels will be launched by passengers after a train arrives. Hence, a proxy may be able to answer subsequent queries interested in the same objects by caching EVRs created based on previous queries. Despite the success of proxy deployment and EVR provisioning in [5], [6], they have the following drawbacks:

. Slow growth of EVRs of NN queries. The algorithms proposed in [5] suffer from slow EVR growth, which causes a lower cache hit ratio. The effect of an EVR in [5] is also limited by the number and locations of the queries interested in the same data object.

. Slow growth of EVRs of window queries. The algorithm proposed in [6] has the same drawbacks as [5]. Besides, since VRs of window queries are likely to be concave polygons, the EVRs are built in a pessimistic manner, causing the EVRs to be extremely small.

. Lack of mutual support. The algorithms of [5] and [6] are executed independently. That is, the result of an NN query is useless for [6] to create the EVR of a window query, and vise versa.

In view of this, we propose in this paper a proxy architecture as well as several companion algorithms to provide EVRs of NN and window queries on static data objects for mobile clients. We aim to reduce the number of queries submitted by mobile clients, the time of obtaining query results and corresponding EVRs, and load on the LBS server. The contributions of this paper are summarized as follows:

. For NN queries, we devise new algorithms to efficiently create new and extend existing EVRs. The devised algorithms not only enable mobile clients to obtain effective EVRs immediately but also lead the proxy to build effective EVRs even when the proxy cache size is small.

. For window queries, different from [6], we propose to index the positions of data objects, instead of EVRs, by a grid index. Since VRs of window queries may not be convex polygons, creating effective polygonal EVRs by previous queries is inherently difficult. Thus, we propose to represent the EVRs of window queries in the form of vectors, called estimated window vectors (EWVs), to achieve larger estimated valid regions. Such novel representation and indexing lead the proxy to efficiently create more effective EVRs of window queries.

. We introduce two index structures, an EVR-tree for NN queries and a grid index for window queries, due to the distinct characteristics of NN and window queries. We also develop algorithms to make these two index structures mutually support each other. Specifically, the grid index can be used to answerNN queries and extend existing EVRs. The answer objects of NN queries are exploited to update the grid index, benefiting the creation of more effective EWVs of window queries.

. We conduct several experiments to compare the proposed approach with the existing proxy-based approaches [5], [6] and the representative server-based approach [7], [8]. The experimental results show that the proposed approach significantly out-performs the existing proxy-based approaches. Moreover, we compare the proposed approach with the representative server-based approach [7], [8] for understanding the benefits of the proxy architecture. The experimental results demonstrate that the performance of the proposed approach is close to that of [7], [8] even though the proposed approach has only partial information of data objects.

The rest of this paper is organized as follows: Section 2 reviews related work and presents the preliminaries. We elaborate NN and window query processing in Sections 3 and 4, respectively. The experimental results are shown in Section 5 to evaluate the performance of the proposed approach. In Section 6, we provide a comparison between proxy-based and server-based approaches. Finally, Section 7 concludes this paper and discusses future work.

2 PRELIMINARIES

2.1 Related Work

In recent years, a significant number of research studies have been proposed for spatial query processing. Most of these studies addressed representative spatial queries, such as NN queries, kNN queries, and window queries. For different scenarios, some studies coped with static data objects while some tackled mobile data objects. The latter studies [9], [10], [11], [12] coped with continuous window query monitoring while [12], [13], [14], [15], [16] addressed continuous kNN query monitoring. Since our work falls into the former category, we focus on reviewing previous studies on static data objects in the following.

R-tree and its variants, such as R–-tree [17], are one of the most popular methods of static spatial query processing. An LBS server is able to answer spatial queries quickly using R-tree-like index structures. However, in mobile environments, mobile clients usually launch a continuous query consisting of a number of queries with different query locations for obtaining a satisfactory answer. Continuous queries cause the conventional scheme to suffer from wireless medium con-tention and heavy query load. To address this problem,

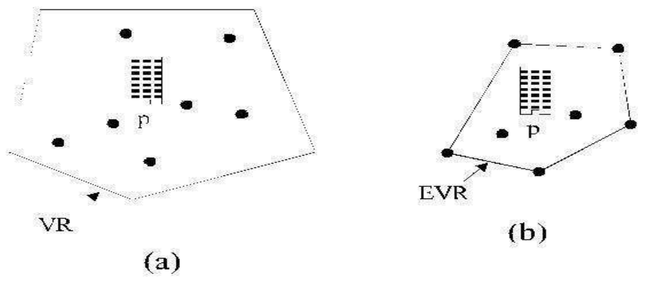

Fig. **1.Example VR and EVR. (a) An example VR. (b) An example EVR**

previous studies proposed that mobile clients could avoid launching unnecessary queries by caching VRs and EVRs. According to the architecture, these studies can be classified into two categories: server-based approaches and proxy-based approaches. Basically, server-based approaches have the complete information of data objects and can utilize the information to create VRs for mobile clients. On the other hand, proxy-based approaches have only partial information of objects and exploit spatial and temporal locality of queries of mobile clients to build EVRs. We first introduce the

.

of data objects in an offline manner and returns the answer object of an NN query as well as the corresponding Voronoi cell to the querying client. The client caches the Voronoi cell to reduce the number of subsequent queries since the Voronoi cell is the VR of the answer object. The main advantage is that the LBS server constructs the Voronoi diagram once and uses it repeatedly. However, searching the corresponding Voronoi cell is time consuming and object updates incur the overhead of partial reconstruction of the Voronoi diagram. Zheng et al. [19] introduced a grid-partition-index to improve search efficiency, but it is still impaired by object updates.

Zhang et al. [20] proposed to compute the VRs by executing a number of time parameterized queries [21] in an online manner that avoids object update cost. The drawback is that, when the number of queries increases, the average waiting time of clients becomes longer since VR computation requires extra I/O and computational cost. Thus, this approach is not suitable for large-scale LBSs. In [7], [8], Lee et al. proposed to retrieve the nearest object and to exploit the objects in the remaining queue to identify the introduced complementary objects. Complementary objects are those objects contributing the bisector as the Voronoi cell perimeter. The VR computation algorithm of [7], [8] only requires one single index lookup. Nutanong et al. [22] presented a technique called V _-Diagram, which computes VRs based on the data objects, the query location, and the current knowledge of the search space. V _-Diagram exploits the FRR (fixed-rank region proposed in [23]) of ðk þ xÞ maintained objects and the safe region of k nearest objects to create the VR of the kNN query. Both [7], [8] and [22] outperform [20] significantly. As to window queries, in [24], Dar et al. allowed a client to store the answer of a window query as a semantic region. Any window query covered by the semantic region could be directly resolved by the client's local cache. Zhang et al. [20] also introduced an online algorithm to compute the VRs of window queries via a number of time parameterized queries, and later, Lee et al. proposed to identify the complementary objects of a window query by enlarging the query window. If all answer objects are inside the query window and no complementary object is encountered, the mobile client can know that the cached answer objects remain valid. In [25], Ku et al. presented a peer-sharing paradigm, allowing mobile clients to obtain a complete or partial result of the query from other clients in the vicinity. A mobile client only waits for unresolved parts from the LBS server and thus the average waiting time could be reduced.

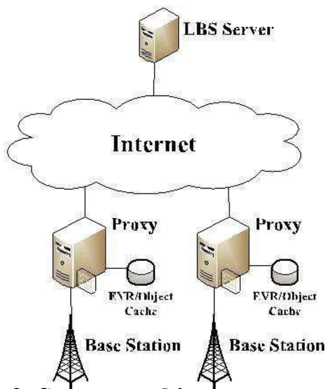

Fig. 2. System architecture

These server-based approaches suffer from heavy load on the LBS server and relatively longer waiting time formobile clients. In order to alleviate the problems, Gao and Hurson [5] presented a proxy-based approach which deploys a proxy between an LBS server and mobile clients. Based on spatial and temporal locality of spatial queries mentioned earlier, they proposed to make the proxy create EVRs according to query history. For example, in Fig. 1a, each point represents a previous NN query whose answer is object p. These query points are within the VR of p. Since VR computation cost is high, the proxy can build the EVR of p by connecting these query points. Because each EVR is a subregion of the corresponding VR, caching EVRs helps the proxy resolve some subsequent queries directly. Following [5], Gao et al. [6] employed the proxy to create the EVRs of window queries based on window query history. All query points with the same query result are used to create the corresponding EVR. Since the VR of a window query may not be a convex polygon, the proxy pessimistically assumes that these query points occur at the boundary of the VR, and creates a feasible but small EVR. Besides, EVR construction cannot be applied to the case where a window query has no answer objects. Moreover, the window queries with the same result set but of different sizes cannot be used to create an EVR.

2.2 System Architecture

Fig. 2 depicts the proposed system architecture for NN and window query processing. The system architecture consists of three parts: 1) an external LBS server, 2) deployed proxies, and 3) the mobile clients. The LBS server is responsible for managing static data objects and answering the queries submitted by the proxies. Note that the LBS server can use any index structure (e.g., R-tree or grid index) to process spatial queries. The LBS server is assumed not to provide VRs. Each of the deployed proxies supervises one service area and provides EVRs of NN queries and EWVs (vector form of EVRs) of window queries for mobile clients in the service area. Each base station serves as an intermediate relay for queries and query results between mobile clients and the associated proxy. Base stations, proxies, and the LBS server are connected by a wired network. A mobile client maintains a cache to store the query results and the corresponding EVRs.

3. MODULES

1. Nearest Neighbor Query Processing
2. Window Query Processing

3.1. Nearest Neighbor Query Processing

When receiving an NN query, a proxy first attempts to answer the query with the EVR-tree and the grid index. If the proxy cannot answer the query, it will submit one or two 2NN queries to the LBS server. The rationale of extending the received NN query into 2NN queries is that the distance between the nearest and second nearest objects is useful for building the EVR of the nearest object based on the theoretical result. By exploiting the objects returned by the LBS server, the proxy extends an existing or creates a new EVR. Besides, the returned objects are utilized to aid grid index growth, which will facilitate window query processing. For each NN query, the proxy provides the mobile client not only the answer object (the nearest object) but also an EVR for helping the client avoid subsequent queries. Note that these 2NN queries initially cause slightly heavier load on the LBS server, but they lead the proxy to provide effective EVRs for mobile clients efficiently. With the EVRs, the number of queries submitted by the clients and the load on the LBS server are reduced greatly. Note that given an object p, the EVR of p is denoted by EVRðpÞ.

3.2 Window Query Processing

On receiving a window query, the proxy first checks whether all the answer objects of the query are available in the object cache by looking up the grid index. Answer objects are the objects within the query window. According to the grid index, the proxy can know whether the grid cells overlapping the query window are fully cached. If a grid cell overlapping the query window is fully cached (e.g., Cell2, Cell4, Cell5, and Cell6 in Fig. 9a), the answer objects within the overlapping region can be directly retrieved from the object cache. Otherwise, the proxy uses the overlapping region of each uncached cell to generate a new window query (e.g., the overlapping regions of Cell1 and Cell3 in Fig. 9a). Then, the proxy submits the new window query to request the required answer objects from the LBS server. However, to accelerate fully cached cell growth, the proxy utilizes the corresponding entire cell as the new query window instead of the overlapping region if 1) the ratio of the region size to the cell size is above the predefined threshold, or 2) the number of interested queries in this cell exceeds the predefined threshold. For example, in Fig. 9a, the ratio of the overlapping region size to Cell3 size is assumed to exceed the predefined threshold. The proxy sends a new window query with Cell3 as the query window to the LBS server. Next, the proxy marks Cell3 as fully cached when it receives all data objects in Cell3 from the server, as shown in Fig. 9b. After obtaining the objects from the server, the proxy has all the answer objects.Finally, the proxy returns the answer objects together with the corresponding EWV to the query client. Note that these new queries used to accelerate grid index grow increase the load on the LBS server initially, but they lead to effectiveEWV creation, significantly reducing the subsequent client.

DESCRIPTION

NN Query

The processing steps executed by the proxy are as follows:

- Step 1. The proxy checks whether the NN query location ðxq; yqÞ is in any EVR of the EVR-tree by performing the general R-tree search operation since EVR-tree is an R-tree (or its variant). If so, go to Step 7.
- Step 2. If not, the proxy attempts to answer the query with the grid index. If the two nearest objects, say p1 and p2, are found in the grid index,2 the proxy looks up the EVR-tree to see whether p1 is located in any existing EVR. If so, go to step 6.
- Step 3. The proxy extends the received NN query into a 2NN query with the same query location ðxq; yqÞ and submits the 2NN query to the LBS server. Let p1 and p2 be the nearest and the second nearest objects to q, respectively. When receiving p1 and p2 from the LBS server, the proxy searches the EVR-tree for EVRðp1Þ. Meanwhile, the proxy runs algorithm GridIndexUpdate3 to update the grid index based on q, p1, and p2. If EV Rðp1Þ is found,go to Step 6.
- Step 4. The proxy generates another 2NN query with query location ðx1; y1Þ where ðx1; y1Þ is the location of p1. Obviously, the nearest object to ðx1; y1Þ is p1. Let the second nearest object to ðx1; y1Þ be p3. The proxy runs algorithm EVR-Creation4 to create the EVR of p1 based on p1, p2, and p3. At the same time,the proxy runs algorithm GridIndexUpdate to update the grid index based on p1 and p3.
- Step 5. The proxy inserts p1 and EVRðp1Þ into the object cache and the EVR-tree, respectively. Go to Step 7.
- Step 6. With q, p1, and p2, the existing EV Rðp1Þ is extended using algorithm EVR-Extension.5 The updated EV Rðp1Þ is reinserted into the EVR-tree.
- Step 7. The proxy returns the answer object p1 and the corresponding EVRðp1Þ to the mobile client.

Window Query

The processing steps executed by the proxy are as follows:

- Step 1. The proxy identifies the overlapping cells based on the width w, the length l, and the center ðxq; yqÞ of the query window.
- Step 2. The proxy examines which of the overlapping cells being fully cached, searches the grid index for the object pointers within these overlapping fully cached cells, and then retrieves the answer objects from the object cache.
- Step 3. For each unresolved overlapping region, the proxy sends a new query with either the region or the corresponding entire uncached cell as the query window to the LBS server.
- Step 4. After receiving the objects from the LBS server, the proxy creates the EWV by algorithm EWV-Creation.6 Meanwhile, if all the objects in one cell are received, the proxy marks the cell in the grid index as fully cached and stores the objects into the object cache.
- Step 5. The proxy returns the answer objects along with the corresponding EWV to the query client.

APPLICATION

- mobile service providers (e.g., Verizon Wireless and AT&T) or Smartphone makers (e.g., Apple and RIM) can utilize a proxy architecture where the proxy provides estimated valid regions (EVRs), which are the sub regions of the corresponding VRs, for the clients.
- With the proxy architecture, the clients still can enjoy the benefits of EVRs even if the LBS server does not provide VRs and thus the mobile service providers and Smartphone makers can attract more clients.

4. CONCLUSION

We have proposed a proxy-based approach to continuous NN and window queries in mobile environments.The proxy takes advantage of spatial and temporal locality of spatial queries to create EVRs of NN and window queries. Different from prior work, we devised new EVR creation and extension algorithms for NN queries, enabling the proxy to build effective EVRs efficiently. The devised algorithms make the proxy achieve high performance even when the cache size is small. On the other hand, we propose to represent EVRs of window queries in the form of vectors, called estimated window vectors, to achieve larger estimated valid regions. Moreover, due to distinct characteristics,we introduce an EVR-tree and a grid index to process NN and window queries, respectively. The algorithms for mutual support of the EVR-tree and the grid index are developed to further enhance the system performance. The experimental results show that the proposed approach significantly outperforms the existing proxy-based approaches since our proposed algorithms create much larger EVRs for mobile clients Compared with the representative server-based approach,the experimental results indicate that the proposed proxy-based approach achieves similar performance even though the proxy has only partial information of data objects. Besides, the results reveal that the proposed proxybased approach is suitable in a densely populated area,whereas the server-based approach is suitable when mobile clients move at high speeds. Although offering the above benefits, the inherent problem of data object updates needs further investigation for the proposed proxy-based approach.In future work, we will investigate the impact of data object updates on the proposed approach and extend the proposed approach to efficiently handle frequent object updates

REFERENCE

1. Y. Cai, K.A. Hua, and G. Cao, “Processing Range-Monitoring Queries on Heterogeneous Mobile Objects,” Proc. Fifth IEEE Int’l Conf. Mobile Data Management, pp. 27-38, 2004.
2. K.C. Lee, J. Schiffman, B. Zheng, and W.-C. Lee, “Valid Scope Computation for Location-Dependent Spatial Query in Mobile Broadcast Environments,” Proc. 17th ACM Conf. Information and Knowledge Management, pp. 1231-1240, 2008..
3. K.C.K. Lee, W.-C. Lee, H.V. Leong, B. Unger, and B. Zheng, “Efficient Valid Scope for Location-Dependent Spatial Queries in Mobile Environments,” J. Software, vol. 5, no. 2, pp. 133-145, Feb.2010.
4. D. Lee, B. Zheng, and W.-C. Lee, “Data Management in Location-Dependent Information Services,” IEEE Pervasive Computing,vol. 1, no. 3, pp. 65-72, July-Sept. 2002.
5. B. Zheng and D.L. Lee, “Processing Location-Dependent Queries in a Multi-Cell Wireless Environment,” Proc.Second ACM Int’lWorkshop Data Eng. for Wireless and Mobile Access, 2001.
6. B. Zheng, J. Xu, W.-C. Lee, and D.L. Lee, “On Semantic Caching and Query Scheduling for Mobile Nearest-Neighbor Search,” Wireless Networks, vol. 10, no. 6, pp. 653-664, Dec. 2004.