International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES)
Impact Factor: 5.22 (SJIF-2017),e-ISSN:2455-2585
International Conference on
Recent Explorations in Science, Engineering And Technology (ICRESET'19)
Volume-5, Special Issue-March, 2019.

RANGE AND LOCATION BASED PRIVACY AWARE MONITORING FOR DYNAMIC OBJECTS

Kamala V[1] , Shanthini M[2]

[1,2]*Assistant Professor, Department Of Computer Science and Engineering, KGiSL Institute of Technology*,

ABSTRACT

Privacy Accuracy Monitoring(PAM) is a framework that addresses the fundamental issues accuracy, efficiency and privacy of moving objects. Location update is the key for this framework which distinguishes it from the existing system. The scheme defines clearly, when and how the mobile clients should exchange their locations with the server. When the query is altered, the PAM updates the location based on relevant result and safe region. Moreover, various client update strategies have been used to optimize accuracy, privacy and efficiency. We develop efficient query evaluation/ revaluation and safe region computational algorithms in the framework. The experimental result show that PAM substantially outperforms traditional schemes in terms of monitoring accuracy, CPU cost and scalability while achieving close-to-optimal communication cost.

Keywords – spatial database, location-dependent and sensitive, mobile applications.

I. INTRODUCTION

In mobile and spatial temporal databases, numerous applications such as public transportation, logistics and location-based services. Require monitoring continuous spatial queries over moving objects. The location information of the objects is managed by database server.. The application servers gather monitoring requests and register spatial queries at the database server, which then continuously updates the query results until the queries are deregistered.

When and how a mobile client should send location updates to the server is the fundamental problem in monitoring system because all the three principal performance measures of monitoring- accuracy, efficiency, and privacy needs to be determined. Accuracy purely depends upon correctness of monitored results, and frequency of location updates. As for efficiency, two dominant costs are: query evaluation cost at the database server and the wireless communication cost for location updates, both of which depend on the frequency of location updates. As for privacy, the accuracy of location update determines how much the client's privacy is exposed to the server.

In the literature, continuous query monitoring is focused on location updates in very few studies. Periodic update and deviation update are two commonly used updating approaches. First problem is the monitoring accuracy is low: query results are correct only at the time instance of periodic updates, but not in between them or at any time of deviation updates. Second location updates are performed regardless of the existence of queries- a high update frequency may improve the monitoring accuracy, but is at the cost of unnecessary updates and query reevaluation. It reaches the peak when updates arrive and trigger query reevaluation, but is idle for the rest of the time. Last, the privacy issues is simply ignored by assuming that the clients are always willing to provide their exact position to the server.

Some recent work attempted to remedy the privacy issue. To blur the exact client positions into bounding boxes, location cloaking was proposed . By assuming a centralized and trustworthy third-party server that stores all exact client positions, various location cloaking algorithms were proposed to build the bounding boxes while achieving the privacy measure such ask-anonymity. The query results no longer unique, the use of bounding boxes makes. As such, query evaluation in such uncertain space is more complicated. The probability distribution of the exact client location in the bounding box is known and well formed is to assume a common approach .The set of all possible results together with their probabilities. All these approaches are focused on one-time.

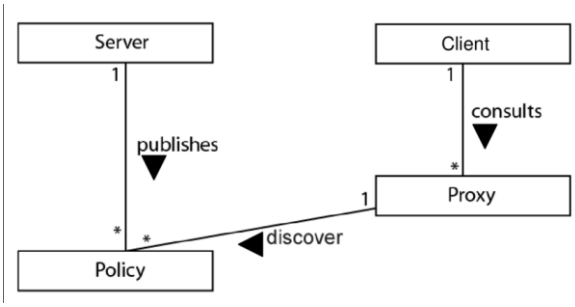

Fig.1.system architecture

We proposed a monitoring framework where the clients are aware of the spatial queries being monitored, so they send location updates only when the results for some queries might change. To maintain a rectangular area, called safe region, for each object this is our basic idea. The safe region is computed based on the queries in such a way that the current results of all queries remain valid as long as all objects reside inside their respective safe regions. We propose a privacy-aware monitoring (PAM) framework that incorporates the accuracy, efficiency, and privacy issues altogether. First, with the introduction of bounding boxes, the result of a query is no longer unique. one of the main contributions of this paper is to devise efficient query processing algorithms for common spatial query types. Second, the most probable result also adds complexity to the definition of safe region. In order to reduce the number of location updates, and thus, improve efficiency new algorithms must be designed to compute maximum safe regions .The strategy determines the accuracy, privacy, and efficiency of the framework. The standard strategy is to update when the centroid of the bounding box moves out of the safe region, which guarantees accuracy- no miss of any change of the most probable result.

PAM framework has the following advantages: First comprehensive framework that addresses the issue of location updating holistically with monitoring accuracy, efficiency, and privacy altogether. Accuracy, the framework offers correct Monitoring results at any time. Provides a common interface for monitoring various types of spatial queries the framework is flexible.

II. RELATED WORK

Early work assumed a static data set and focused on efficient access methods (e.g., R-tree [1]) and query evaluation algorithms (e.g., [2], [3]). Recently, where data objects or queries or both of them move with a lot of attention has been paid to moving-object databases.

Distributed approaches have been investigated to monitor continuous range queries and continuous kNN queries. To shift some load from the server to the mobile clients is the main idea. Monitoring queries have also been studied for distributed Internet databases [4], data streams [5], and sensor databases [6].Where a two-dimensional space is assumed, however, these studies are not applicable to monitoring of moving objects.

III . FUNDAMENTALS OF PAM FRAMEWORK

A. Privacy-Aware Location Model

In our monitoring framework, we take the same privacy-aware approach. Specifically, each time a client detects his/her genuine point location; it is encapsulated into a bounding box. Then, location updater of the client-side decides whether or not to update the box to the server without any other knowledge about the client locations or moving patterns, upon receiving.

The key idea to solving the problem is "safe region," which was defined in [21] as a rectangle within which the change of object location does not change the result of any registered spatial query. The reason why we exclude all other less probable results in this definition is threefold: 1) monitoring continuous queries usually trades accuracy for efficiency-although the most probable result does not always align with the genuine result.

B. *Framework Overview*

The PAM framework consists of Components located at both the database server and the moving objects. At the database server side, we have the moving object index, the query index, the query processor, and the location manager. At moving objects' side, we have location updaters.

Algorithm 1 summarizes the procedure at the database server to handle a query registration/deregistration or a location update.

Algorithm 1. Overview of Database Behavior

1: while receiving a request do

2: if the request is to register query q then

3: evaluate q;

4: compute its quarantine area and insert it into the Query index;

5: return the results to the application server;

6: update the changed safe regions of objects;

7: else if the request is to deregister query q then

8: remove q from the query index;

9: else if the request is a location update from object p then

10: determine the set of affected queries;

11: for each affected query q0 do

12: reevaluate q0;

13: update the results to the application server;

14: recomputed its quarantine area and update the Query index;

15: update the safe region of p;

At any time, application servers can register spatial queries to the database server (step _1). When an object sends a location update (step_2), the query processor identifies those queries that are affected by this update using the query index, and then, re evaluates them using the object index (step _3). The updated query results are then reported to the application servers who register these queries. Afterward, the location manager computes the new safe region for the updating object (step _4), also based on the indexes, and then, sends it back as a response to the object (step _5). The procedure for processing a new query is similar, except that in step _2 , the new query is evaluated from scratch instead of being re evaluated incrementally, and that the objects whose safe regions are changed due to this new query must be notified. Algorithm 1 summarizes the procedure at the database server to handle a query registration/deregistration or a location update

In this paper, the most probable result is used; this framework can also adapt to other query result definitions such as over a probability confidence (e.g., "returns objects that have 90 percent probability inside the query range"). The only changes needed to reflect the new result definition are the query evaluation algorithms in the query processor and safe region computation in the location manager.

C. The Object Index

The object index is the server-side view on all objects. More specifically, to evaluate queries, the server must store the spatial range, in the form of a bounding box, within which each object can possibly locate. Note that this bounding box is different from a _-square because its shape also depends on the client-side location updater. That is, it must be a function (denoted by _) of the last updated _-square and the safe region. As such, this box is called a bbox as a mark of distinction. Since the bbox changes each time the object updates, the index is optimized to handle frequent updates [29].

D. The Query Index

For each registered query, the database server stores: 1) the query parameters (e.g., the rectangle of a range query, the query point, and the k value of a kNN query); 2) the current query results; and 3) the quarantine area of the query. The quarantine area is used to identify the queries whose results might be affected by an incoming location update. It originates from the quarantine line, which is a line that splits the entire space into two regions: the inner region and the outer region. For a range query q, the query window can serve as an inner bound of the quarantine area, for a kNN query, since only the distance to the query point q matters, we set both the inner and the outer bounds as circles centered at q.

III. QUERY PROCESSING

In this space, spatial relations such as overlapping, containment, or even HU ET AL.: PAM: AN EFFICIENT AND PRIVACY-AWARE MONITORING FRAMEWORK FOR CONTINUOUSLY MOVING OBJECTS 409 distance are implemented differently from a conventional Euclidean space. By using the new implementations of spatial relations, existing spatial query processing algorithms can be applied directly to the new space.

A. Spatial Relations

In the new space, an object p is contained in a rectangle R if in the euclidean space, the majority of p is in R. The rectangle divides the enlarged safe region of any object p into two regions: the region inside rectangle q (where p is a result object of q) and the region outside q (where p is not a result). The region with the larger area decides the most probable result.In the new space, an object p1 is closer to a point q than object p2 if and only if in the euclidean space, for two randomly picked points a; b from p1 andp2,respectively, a is equally or more probably closer to q than b. The closer relation has a nice property**.**

The closer relation has a nice property that it is a total order relation, which is proved by the following preposition:

Proposition 4.1. The closer relation is a total order relation that satisfies

1. Reflexivity,

2. Anti symmetry,

3. Transitivity, and

4. Comparability.

Cases 1 and 2 are trivial.

3. Transitivity: if p1 is closer than p2 and p2 is closer than p3, then a (from p1) is more probably closer to q than b (from p2), which is, in turn, more probably closer to q than c (from p3). As such, p1 is closer than p2.

4. Comparability: 8p1; p2, either p1 is closer than p2, or p2 is closer than p1.

B. Query Evaluation and Re evaluation on Object Index

In conventional euclidean space, a new range query is evaluated as follows: We start from the index root and recursively traverse down the index entries that overlap with the query window until the leaf entries storing the objects are reached. Then, we test each object using the containment relation in the new space.

The best-known algorithm to evaluate a kNN query q in conventional euclidean space is the best-first search (BFS). It uses a priority queue H to store the to-be-explored index entries which may contain kNNs. The entries in H are sorted by their minimum distances to the query point q.BFS works by always popping up the top entry from H,pushing its child entries into H, and then, repeating the process all over. When a leaf entry, i.e., an entry of a leaf node, is popped, the corresponding object is returned as a nearest neighbour. The algorithm terminates if k objects have been returned.

Algorithm 2: Evaluating a new kNN Query

Input: root: root node of object index

q: the query point

Output: C: the set of kNN

Procedure:

1: initialize queue H and H;

2: enqueue (root, d(q, root))into H;

3: while |C| < k and H is not empty do

4: u = H.pop();

5: if u is a leaf entry then

6: while d(q, u)>D(q ,u)do

7: v = H.pop();

8: insert v to C;

9: en queue u into H;

10: else if u is an index entry then

11: for each child entry v of u do

12: en queue (v ,d(v, q))into H;

In the new space, the query is evaluated similarly, which is shown in Algorithm 2. However, the algorithm maintains an additional priority queue H besides H. It is a priority queue of objects sorted by the "closer" relation. The reason to introduce H is that when an object p is popped from H, it is not guaranteed a kNN in the new space. Therefore, H is used to hold p until it can be guaranteed a kNN. This occurs when another object p0 is popped from H, and its minimum distance to q (d(q ,p')) is larger than the maximum distance of p to q (D(q ,p)). In general, when an object u is popped from H, we need to do the following. If d(q, u) is larger than D(q, v), where v is the top object in H, then v is guaranteed a kNN and removed from H. Then, d(q, u) is compared with the next D(q, v) until it is no longer the larger one. Then, u itself is inserted to H and the algorithm continues to pop up the next entry from H. The algorithm continues until k objects are returned.

Algorithm 3: Reevaluating a KNN Query

Input: C: existing set of kNNs

p: the updating object

Output: C: the new set of kNNs

Procedure:

1: if p is closer to the k-th NN then

2: if p 2 C then

3: p_ = the rank of p in C;

4: else

5: p_ = k;

6: enqueue p into C;

7: else

8: if p 2 C then

9: evaluate 1NN query to find u;

10: p_ = k;

11: remove p and enqueue u into C;

12: relocate p or u in C, starting from p_;

To reevaluate an existing kNN query that is affected by the updating object p, the first step is to decide whether p is a result object by comparing p with the kth NN using the "closer" relation: if p is closer, then it is a result object; otherwise, it is a non result object. This then leads to three cases: 1) case 1: p was a result object but is no longer so; 2) case 2: p was not a result object but becomes one; and 3) case 3: p is and was a result object.4 For case 1, there are fewer than k result objects, so there should be an additional step of evaluating a 1NN query at the same

query point to find a new result object u. The evaluation of such a query is almost the same as Algorithm 2, except that all existing kNN result objects are not considered. The final step of re evaluation is to locate the order of new result object p in the kNN set. This is done by comparing it with other existing objects in the kNN set using the "closer" relation. For cases 1 and 2, since this object is a new result object, the comparison should start from the kth NN, then k-1th NN, and so on. However, for case 3, since p was in the set, the comparison can start from where p was. Algorithm 3 shows the pseudo code of KNN query re evaluation, where p_ denotes the starting position of the comparison.

IV. PROPOSED SYSTEM:

To overcome the problem while monitoring moving objects we propose spatial join queries. In order to find the nearest location of dynamic objects also optimize the performance of framework then the optimal safe region not only depend on the query. But also on the accumulated safe region.

A. Dynamic Client Update Strategy:

The standard update strategy, which updates when the centroid of _-square is out of the safe region, guarantees 100 percent monitoring accuracy in the context of the most probable result. This is a static strategy where the decision is made independent of previous decisions. In this section, we discuss two dynamic strategies that achieve objectives other than monitoring accuracy.

Performance Evaluation

To evaluate the monitoring performance, we implement a simulation test bed, where N moving objects move within a unit-square space [0..1, 0..1]. Each object detects its point location at frequency f, encapsulates it into a _-square, and forwards the square to the location updater. Each object has an individual _ and it follows a normal distribution with mean value .

We compare our PAM framework with two other frameworks, namely, the optimal monitoring (denoted as OPT) and the periodic monitoring (denoted as PRD). In optimal monitoring, every object has the perfect knowledge of the registered queries and the _-squares of other moving objects at any time. Therefore, it knows precisely when the most probable result of any query changes, and only then does it send a location update to the server. OPT serves as the lower bound for all monitoring frameworks. In periodic monitoring, all objects periodically send out location updates simultaneously and the server all registered queries based on these updates. Obviously, its monitoring accuracy and cost depend on the updating interval. In this paper, we test PRD with updating intervals 0.1 and 1, denoted as PRD(0.1) and PRD(1)

B. Simulation Setup

In the simulation test bed, each object moves according to the random waypoint mobility model: the client chooses a random point in the space as its destination and moves to it at a speed randomly selected from the range ½0; 2v;upon arrival or expiration of a constant movement period (randomly picked from the range ½0; 2tv), it chooses a new destination and repeats the same process.

The performance metrics for comparison include:

Monitoring accuracy: The monitoring accuracy at time ,is defined as whether the monitored results for all queries accord with the results from the OPT framework.

Wireless communication cost: It is the amortized number of location updates sent by a moving object over time.

CPU time : This is measured by the amortized server CPU time, which includes the time for query evaluation and safe region computation.

C. Validity Of Most Probable Result:

The first set of experiments is to validate the definition of most probable result. Under various the mean and the location detection frequency, we compare the most probable result from the OPT framework with the genuine result (the result as if all the point locations were known) for all W queries. The proportion of time when the two results are the same. As or f increases, the consistency rate drops. However, the Curve is not linear: the drop becomes slower when and f become larger. As such, even when or f is very large, the consistency rate is above 70 percent. This justifies our claim that the most probable result is a nice approximation of the genuine result for monitoring tasks.

D. Overall Performance

As is guaranteed, our PAM framework achieves 100 percent accuracy, while PRD gets only 80-90 percent. Obviously, PRD(0.1) is more accurate than PRD(1) but the performance gap is less than10 percent. Further, it is at the cost of 10 times higher communication overhead. On the other hand, the communication cost of PAM is much smaller than PRD and remains close to OPT

E. Effects Of Query Types

In this section, we study the performance of PAM on range and kNN queries separately. We vary the average query length qlen of range queries and kmax—themaximumk of kNN queries. For range queries, as qlen increases, the communication cost of OPT always increases at a steady pace. However, the communication cost of PAM increases more slowly when qlen is relatively small (at 0.001) or large (>0:01). Since the size of a cell is fixed, the cost tends to saturate. On the other hand, for kNN queries, as kmax increases, the costs of both OPT and PAM grows steadily. Even so, PAM manages to narrow the gap when kmax becomes larger. This suggests that for a heavy workload when results change frequently, the safe region achieves even better approximation to the ideal safe area.

VI. CONCLUSIONS

This paper proposes a framework for monitoring continuous spatial queries over moving objects. The framework is the first to holistically address the issue of location updating with regard to monitoring accuracy, efficiency, and privacy. We provide detailed algorithms for query evaluation/ reevaluation and safe region computation in this framework. We plan to incorporate other types of queries into the framework, such as spatial joins and aggregate queries. We also plan to further optimize the performance of the framework. A possible solution is to sequentially optimize the queries but maintain the safe region accumulated by the queries optimized so far. Then, the optimal safe region for each query should depend not only on the query, but also on the accumulated safe region.

REFERENCES

(1)A.Guttman,"R-Trees:A Dynamic Index Structure for Spatial Searching",Proc.ACM SIGMOD,1984.

(2)G.R.Hjaltason and H.Samet,"Distance Browsing in Spatial Databases",ACM Trans. Database systems, vol24,no.2,pp.265-318,1999.

(3)Roussopoulos,S.Kelly,andF.Vincent,"Nearest Neighbors Queries",Proc.ACM SIGMOD,1995.

(4)J.Chen,D.Dewitt,F.Tian,andY.Wang,"NiagaraCQ:A Scalable Continous Query System for Internet Databases",Proc.ACM SIGMOD,2000

(5)S.Babu and j.Widom,"Continuous Queries over Data streams",Proc.ACM SIGMOD,2001

(6)S.R.Madden,M.J..Franklin,J.M.Hellerstein,and W.Hong,"TAG: A T iny Aggregation Service for Ad-Hoc Sensor Networks",Proc.USENIX S ymp.Operating Systems Design and Implemenation(OSDI),2002.