PAM: An Efficient and Privacy-Aware Monitoring Framework for Continuously Moving Objects

Bhakte D.B.[1],Vharkate M. N.[2] & Gojre K. N.[3]

Abstract : **Efficiency and privacy are two fundamental issues in moving object monitoring. This paper proposes a privacy-aware monitoring (PAM) framework that addresses both issues. The framework distinguishes itself from the existing work by being the first to holistically address the issues of location updating in terms of monitoring accuracy, efficiency, and privacy, particularly, when and how mobile clients should send location updates to the server. Based on the notions of safe region and most probable result, PAM performs location updates only when they would likely alter the query results. Furthermore, by designing various client update strategies, the framework is flexible and able to optimize accuracy, privacy, or efficiency. We develop efficient query evaluation/reevaluation and safe region computation algorithms in the framework. The experimental results show that PAM substantially outperforms traditional schemes in terms of monitoring accuracy, CPU cost, and scalability while achieving close-to-optimal communication**

Keywords : **Spatial databases, location- dependent and sensitive, mobile applications.**

1. INTRODUCTION

In mobile and spatiotemporal databases, monitoring continuous spatial queries over moving objects is needed in numerous applications such as public transportation, logistics, and location-based services. Fig. 1 shows a typical monitoring system, which consists of a base station, a database server, application servers, and a large number of moving objects (i.e., mobile clients).

1. *College of Engineering Osmanabad*
deep_bhakte07@yahoo.co.in

2. *College of Engineering Osmanabad*
minakshi_ghodke1@yahoo.co.in

3. *College of Engineering Osmanabad*
gojre.komal@gmail.com

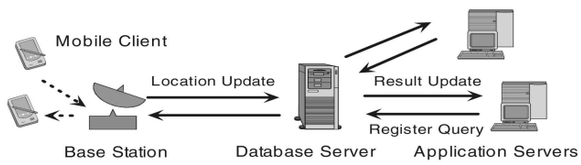

Fig.1 The system architecture.

The database server manages the location information of the objects. 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. The fundamental problem in a monitoring system is when and how a mobile client should send location updates to the server because it determines three principal performance measures of monitoring—accuracy, efficiency, and privacy. Accuracy means how often the monitored results are correct, and it heavily depends on the frequency and accuracy of location updates. As for efficiency, two dominant costs are: the wireless communication cost for location updates and the query evaluation cost at the database server, both of which depend on the frequency of location updates. As for privacy, the accuracy of location updates determines how much the client's privacy is exposed to the server. In the literature, very few studies on continuous query monitoring are focused on location updates. Two commonly used updating approaches are periodic update (every client reports its new location at a fixed interval) and deviation update (a client performs an update when its location or velocity changes significantly) . However, these approaches have several deficiencies. First, the monitoring accuracy is low: query results are correct only at the time instances 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. Third, the server workload using periodic update is not balanced over time: it reaches the peak when updates arrive (they must

arrive simultaneously for correct results) and trigger query reevaluation, but is idle for the rest of the time. Last, the privacy issue is simply ignored by assuming that the clients are always willing to provide their exact positions to the server. Some recent work attempted to remedy the privacy issue. Location cloaking was proposed to blur the exact client positions into bounding boxes. 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 as k-anonymity. However, the use of bounding boxes makes the query results no longer unique. As such, query evaluation in such uncertain space is more complicated. A common approach is to assume that the probability distribution of the exact client location in the bounding box is known and well formed. Therefore, the results are defined as the set of all possible results together with their probabilities. However, all these approaches focused on one-time cloaking or query evaluation; they cannot be applied to monitoring applications where continuous location update is required and efficiency is a critical concern.

2. FRAMEWORK OVERVIEW

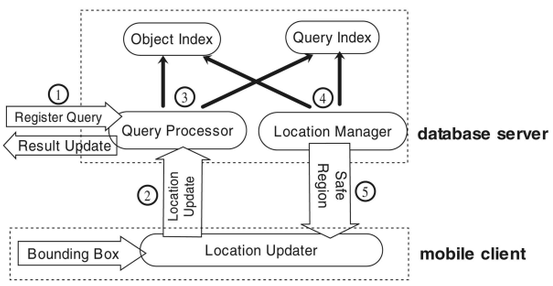

Fig.2 PAM framework overview

As shown in Fig. 2, 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. Without loss of generality, we make the following assumptions for simplicity:

The number of objects is some orders of magnitude larger than that of queries. As such, the query index can accommodate all registered queries in main memory, while the object index can only accommodate all moving objects in secondary memory. The database server handles location updates sequentially; in other words, updates are queued and handled on a first-come-first-serve basis. This is a reasonable assumption to relieve us from the issues of read/write consistency. The moving objects maintain good connection with the database server. Furthermore, the communication cost for any location update is a constant. With the latter assumption, minimizing the cost of location updates is equivalent to minimizing the total number of updates. PAM framework works as follows (see Fig 2): 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, reevaluates 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 reevaluated incrementally, and that the objects whose safe regions are changed due to this new query must be notified.

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: recompute its quarantine area and update the query index;
15: update the safe region of p;

3. PROPOSED SYSTEM

In our approach to maintain safe region we have object index, Query index, the query processor and location manager. As for efficiency, the framework significantly reduces location updates to only when an object is moving out of the safe region, and thus, is very likely to alter the query results. 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.

4. METHODOLOGY

1. Privacy-Aware Location Model

We assume that the clients are privacy conscious. That is, the clients do not want to expose their genuine point locations to the database server to avoid spatiotemporal correlation inference attack which an adversary may infer users' private information such as political affiliations, alternative lifestyles, or medical problems. For example, knowing that a user is inside a heart specialty clinic during business hours, the adversary can infer that the user might have a heart problem. This has been cited as a major privacy threat in location-based services and mobile computing. To protect against it, most existing work suggests replacing accurate point locations by bounding boxes to reduce location resolutions with a large enough location box covering the sensitive place (e.g., the clinic) as well as a good number of other insensitive places, the success rate or confidence of such spatiotemporal correlation inference can be reduced significantly. 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, the client-side location updater decides whether or not to update that box to the server. Without any other knowledge about the client locations or moving patterns, upon receiving such a box, the server can only presume that the genuine point location is distributed uniformly in this box.

2. 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 δ- square because its shape also depends on the client-side location updater. That is, it must be a function (denoted by $\odot$) of the last updated δ- square and the safe region. As such, this box is called a bbox as a mark of distinction. In particular, for the standard update strategy, the bbox is the safe region enlarged by $\delta/2$ on each side, or formally, the "Minkowski sum"2 of the safe region and a $\delta/2$- square.

3. 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. An object becomes a result object if it enters the inner region; likewise, it becomes a nonresult object once it enters the outer region. However, the ideal quarantine line is difficult to compute, especially in the context of the most probable result. In addition, as object locations have extensions rather than points, the quarantine line is not unique for a query. As such, we allow fuzziness by relaxing the line to an area called "quarantine area." That is, the entire space is split into three regions: the inner region, the quarantine area, and the outer region. The former two are separated by the inner bound of the quarantine area, whereas the latter two are separated by the outer bound of the quarantine area. To ease the computation of these two bounds, an object becomes a result object if its δ- square moves totally inside the inner bound; on the other hand, an object becomes a nonresult object once its δ- square crosses or is outside the outer bound. Therefore, a query Q is not affected only if "of the updated δ- square p and its last updated δ- square p_{lst}, both of them are totally inside the inner bound or both of them cross or are outside the outer bound of the quarantine area."

4. Query Processor and Location Manager

In the PAM framework, based on the object index, the query processor evaluates the most probable result when a new query is registered, or reevaluates the most probable result when a query is affected by location updates. Obviously, the reevaluation is more efficient as it can be based on previous results.

The location manager computes the safe region of an object p (denoted as $p.sr$). Recall that a safe region is a rectangle within which the change of the centroid of p's δ-square does not change the most probable result of any registered query. As queries are independent of each other, we can further define the safe region for a single query Q (denoted as $p.sr_Q$) as a rectangle in which the change of the centroid of p's δ-square does not change the most probable result of Q. By this

definition, $p.sr_Q$ is a rectangular approximation, or more accurately an inscribed rectangle, of $Q's$ inner (if p is a result object) or outer (if p is a nonresult object) regions, which are separated by the quarantine line. The reason why the safe region is based on the quarantine line rather than the quarantine area is that the latter is much coarser. Furthermore, the quarantine area is used only to filter out the queries that are not affected by a location update, so we trade accuracy for efficiency. The safe region, on the other hand, directly dictates the frequency, and hence, the cost of location updates, so we compute it based on the more accurate quarantine line.

After each individual $p.sr_Q$ is computed, $p.sr$ is simply the intersection of these $p.sr_Q$ from all registered queries. To eliminate those queries whose safe regions do not contribute to $p.sr$, the location manager further requires every $p.sr_Q$ (and thus, the $p.sr$) to be fully contained in the home cell(s). Recall that the home cell(s) are the grid cell(s) of the query index where the δ-square of p is contained or overlaps. By this means, the location manager only needs to compute the safe regions for those queries (subsequently called relevant queries) whose quarantine areas are contained or overlap with the home cell(s). These relevant queries are exactly those indexed by the home cell(s) of the query index.

The location manager recomputes the safe region of an object p in two cases: 1) after a new query Q is evaluated and 2) after p sends a location update. In the former case, since no existing queries change their quarantine lines, the new safe region $p.sr'$ is simply the intersection of the current safe region $p.sr$ and $p.sr_Q$, the safe region for this new query Q. If $p.sr'$ is different from $p.sr$, the new safe region should be updated to p. In the latter case, the quarantine areas of some existing queries might change; therefore, $p.sr'$ needs to be completely recomputed by computing the $p.sr_Q$ for each relevant query and then getting the intersection.

5. RESULTS

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 also devise three-client update strategies that optimize accuracy, privacy, and efficiency, respectively. The performance of our framework is evaluated through a series of experiments. The results show that it substantially outperforms periodic monitoring in terms of accuracy and CPU cost while achieving a close-to-optimal communication cost. Furthermore, the framework is robust and scales well with various parameter settings, such as privacy requirement, moving speed, and the number of queries and moving objects.

6. CONCLUSION

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 also devise three-client update strategies that optimize accuracy, privacy, and efficiency, respectively. The performance of our framework is evaluated through a series of experiments. The results show that it substantially outperforms periodic monitoring in terms of accuracy and CPU cost while achieving a close-to-optimal communication cost. Furthermore, the framework is robust and scales well with various parameter settings, such as privacy requirement, moving speed, and the number of queries and moving objects.

7. REFERENCES

[1] J. Broch, D.A. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva, “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols,” Proc. ACM/IEEE MobiCom, pp. 85-97, 1998.

[2] S. Prabhakar, Y. Xia, D.V. Kalashnikov, W.G. Aref, and S.E. Hambrusch, “Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects,” IEEE Trans. Computers, vol. 51, no. 10, pp. 1124-1140, Oct. 2002.

[3] N. Roussopoulos, S. Kelley, and F. Vincent, “Nearest Neighbor Queries,” Proc. ACM SIGMOD, 1995.

[4] M.L. Yiu, C.S. Jensen, X. Huang, and H. Lu, “Spacetwist: Managing the Trade Offs among Location Privacy, Query Performance, and Query Accuracy in Mobile Services,” Proc. IEEE Int’l Conf. Data Eng. (ICDE ’08), 2008.

[5] S. Babu and J. Widom, “Continuous Queries over Data Streams,” Proc. ACM SIGMOD, 2001.

[6] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. ACM SIGMOD, pp. 322-331, 1990.
[7] R. Benetis, C.S. Jensen, G. Karciauskas, and S. Saltenis, "Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects," Proc. Int'l Database Eng. and Applications Symp. (IDEAS), 2002.
[8] A. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," IEEE Pervasive Computing, vol. 2, no. 1, pp. 46-55, Jan.-Mar. 2003.
[9] J. Broch, D.A. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," Proc. ACM/IEEE MobiCom, pp. 85-97, 1998.
[10] Y. Cai, K.A. Hua, and G. Cao, "Processing Range-Monitoring Queries on Heterogeneous Mobile Objects," Proc. IEEE Int'l Conf. Mobile Data Management (MDM), 2004.
[11] J. Chen and R. Cheng, "Efficient Evaluation of Imprecise Location-Dependent Queries," Proc. IEEE Int'l Conf. Data Eng. (ICDE), pp. 586-595, 2007.
[12] J. Chen, D. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," Proc. ACM SIGMOD, 2000.