Middle-East Journal of Scientific Research 20 (8): 1000-1005, 2014
ISSN 1990-9233
© IDOSI Publications, 2014
DOI: 10.5829/idosi.mejsr.2014.20.08.114032

Privacy Conscious Screening Framework for Frequently Moving Objects

S.P. Vijayaragavan, B. Karthik and T.V.U. Kiran Kumar

Department of EEE,
Bharath University, Chennai, India

Abstract: The mainstay of this project is to monitor the continuously moving objects location and minimizes the number of location updates using PAM. 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. It is also proposed to optimize the performance of the framework. In particular the minimum cost update strategy shows that the safe region is a crude approximation of the ideal safe area, mainly because to separately optimize the safe region for each query, but not globally. 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 by developing efficient query evaluation/reevaluation and safe region computation algorithms in the PAM framework.

Key words: Spatial databases · Location-dependent and sensitive · Mobile applications

INTRODUCTION

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, in particular 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 [1-3]. 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 cost.

Related Work

Framework Overview: PAM framework has the following advantages:

- To our knowledge, this is the first comprehensive framework that addresses the issue of location updating holistically with monitoring accuracy, efficiency and privacy altogether.
- 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 [4-8].
- As for accuracy, the framework offers correct monitoring results at any time, as opposed to only at the time instances of updates in systems that are based on periodic or deviation location update.
- The framework is generic in the sense that it is not designed for a specific query type. Rather, it provides a common interface for monitoring various types of spatial queries such as range queries and kNN queries.
- The framework is flexible in that by designing appropriate location update strategies, accuracy, privacy, or efficiency can be optimized.

As shown in Fig. 1, 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'

Corresponding Author: S.P. Vijayaragavan, Department of EEE, Bharath University, Chennai, India.

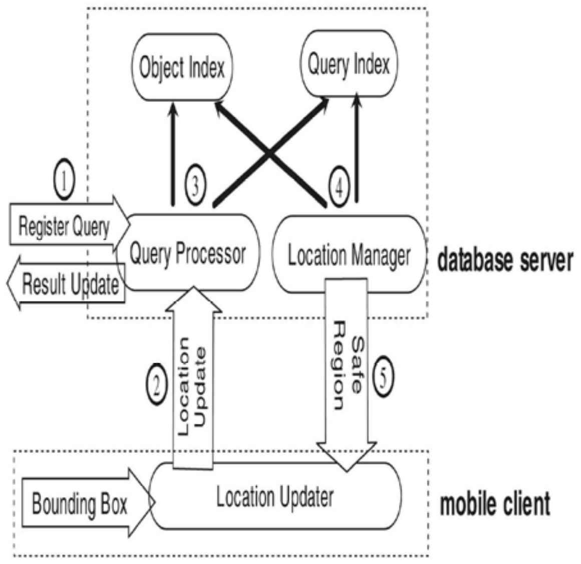

Fig. 1: PAM framework

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 [10]. With the latter assumption, minimizing the cost of location updates is equivalent to minimizing the total number of updates.

PAM framework works as follows (Fig. 1): 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 [11]. 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 summarizes the procedure at the database server to handle a query registration/deregistration or a location update.

Query Evaluation and Reevaluation: 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 : Recompute its quarantine area and update the
     query index;
15 : Update the safe region of p;
```

Handling a query registration/unregistration or a source- initiated location update. When a new query is registered, the query processor evaluates its initial result based on the object index. The evaluation is different from that for traditional spatial queries as the objects are represented by safe regions rather than their exact locations. In case of ambiguity, location probes are necessary. In order to reduce the number of probes, we apply a lazy probe technique so that probes occur only when the evaluation cannot continue. Similarly, upon receiving a source-initiated location update, the query processor finds out the affected queries using the query index. It then incrementally reevaluates these queries and updates their results if they change. Finally, the safe regions of the updated object and the probed objects are recomputed by the location manager. We discuss query evaluation and reevaluation algorithms in this section and the computation of safe regions in the next section [12]. Due to space limitations, this paper presents the solutions for two most common spatial queries, namely the range and kNN queries and the extension to other types of queries follows the same rationale.

Evaluating New Range Query: Processing a new range query on safe regions is very similar to that on exact object locations. We start from the index root and recursively traverse down the index entries that overlap the query rectangle until reaching the leaf entries where the safe regions are stored. If the safe region of an object is fully covered by the query rectangle, the object is a result. Otherwise, they overlap and the object must be probed to resolve the ambiguity.

Evaluating N ew kNN Query: Evaluating an order-sensitive kNN query on safe regions is more complicated than that on exact object locations. We adopt the best-first search (BFS) as the paradigm for this algorithm. Similar to the original BFS, we maintain a priority queue which stores intermediate or leaf index entries (i.e. object locations). The object location is in the form of either a safe region or the exact point (after the server probes it). The key to sort the elements in the queue is the (minimum) distance to the query point q. The searching algorithm (Algorithm 2) is the same as the original BFS except when a leaf entry, object p (Figure 2), is popped from the queue. If p is already represented by a point, it is returned as a result immediately. Otherwise p, represented by a safe region, is held until the next object u is popped. Then, the maximum distance between p and q (_(q, p)) is compared with the minimum distance between u and q (_(q, u)). If the former is shorter, p is guaranteed closer to q than any other objects in the queue. As such, p is returned as a result [13]. However, if the former is longer, the query result is undecided based on the safe regions. Therefore, p is probed and p together with its exact location is inserted back to the priority queue. And u is also inserted back to the queue. The algorithm continues until k objects are returned as results. In addition, in order to find the radius r of the quarantine area for this query, the algorithm pops one more element from the queue and the value of r is the midpoint between the keys of the k-th NN and the last popped element.

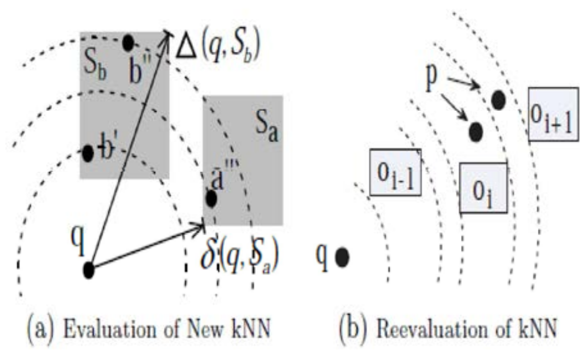

Fig. 2: Processing a kNN Query

Algorithm 2: Evaluating a new kNN Query

- Input: root: root node of object indexq: the query point
- Output: C: the set of kNNs

Procedure:

```
1  : Initialize queue H and H;
2  : Enqueue hroot; dðq; rootÞi into H;
3  : While jCj < k and H is not empty do
4  : U ¼ H.pop();
5  : If u is a leaf entry then
6  : While dðq; uÞ > Dðq; vÞ do
7  : v = H.pop();
8  : Insert v to C;
9  : Enqueue u into H;
10 : Else if u is an index entry then
11 : For each child entry v of u do
12 : Enqueue hv; dðv; qÞi into H;
```

Reevaluating Range and kNN Queries: The incremental reevaluation of an affected range query is straightforward. If the updated object p (figure 2) is now inside the query rectangle Q, p must have moved into this rectangle from outside, so it becomes a new result of Q. Similarly, if p is now outside the rectangle, p is removed from the result set of Q. In either case, the database server reports the update to the application server.

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_ ¼ the rank of p in C;
4  : Else
5  : P_ ¼ k;
6  : Enqueue p into C;
7  : else
8  : If p 2 C then
9  : Evaluate 1NN query to find u;
10 : P_ ¼ 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 nonresult 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 reevaluation 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 pseudocode of kNN query reevaluation, where p_ denotes the starting position of the comparison.

Safe Region Computation: The safe region of a moving object p (denoted as p.sr) designates how far p can reach without affecting the results of any registered query. As queries are independent of each other, we define the safe region for a query Q (denoted as p.srQ) as the rectangular region in which p does not affect Q's result. p.srQ is essentially a rectangular approximation of Q's quarantine area or its complement. Obviously, p.sr is the intersection of individual p.srQ for all registered queries. To efficiently

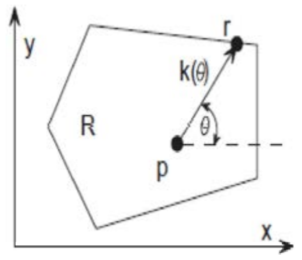

Fig. 3: Safe region computation

eliminate those queries whose p.srQ do not contribute to p.sr, we require p.sr (and p.srQ) to be fully contained in the grid cell in which p currently resides. By this means, we only need to compute p.srQ for those queries whose quarantine areas overlap this cell as the p.srQ for any rest query is the cell itself.

Given a convex safe region R (Figure 3) and the updated location p, the amortized location update cost for p over time, Costp, is

$$Cost_p = C_l \cdot \left(\int_0^{2\pi} \frac{k(\theta)\, d\theta}{2\pi\phi} \right)^{-1} = \frac{C_l . 2\pi\phi}{Perimeter(R)},$$

System Study
Problem Description:

- No safe region provided for moving objects.
- Compute safe region globally.
- Unnecessary updates of objects.

Existing System:

- Contains location resolutions.
- Assumed a static dataset and focused on efficient access methods.
- Privacy, efficiency and accuracy these parameters are low.
- Compute safe region globally.
- No safe region provide for moving objects.
- Cost of unnecessary updates and query evaluation.

Proposed System:

- PAM framework addresses these issues.
- To replace accurate point locations by bounding boxes to reduce location resolutions.
- To monitor the continuously moving objects location and minimizes the number of location updates.

- The PAM framework is to minimize the number of location updates.
- Separately optimize the safe region for each query, but not globally.
- To provide detailed algorithms for query evaluation/re-evaluation and safe region computation in this framework.
- To provide new safe region when the object come out the safe region.

CONCLUSION

This paper proposes a generic framework for monitoring continuous spatial queries over moving objects. The frame work distinguishes itself from existing work by being the first to address the location update issue and to provide a common interface for monitoring mixed types of queries. Based on the notion of safe region, the location updates are query aware and thus the wireless communication and query reevaluation costs are significantly reduced. We provide detailed algorithms for query evaluation/reevaluation and safe region computation in this framework. Enhancements are also proposed to take advantage of two practical mobility assumptions: maximum speed and steady movement. To evaluate the performance, we thoroughly conduct a series of experiments and compare the proposed framework with the optimal monitoring and the traditional periodic monitoring schemes. This paper demonstrates the feasibility and performance advantages of the framework. As for future work, we plan to incorporate other types of queries into the framework, such as spatial joins and aggregate queries. We also plan to optimize the performance of the framework. For example, so far the safe regions for kNN queries are computed separately; so the final safe region of the object p.sr may not be op- timal. To achieve a larger p.sr, we are going to develop algorithms which incrementally update the so-far computed p.sr for each relevant query.

ACKNOLDGEMENT

I would like to thank my parents who offered me an enormous support and enthusiasm till the completion of my project. I am extremely glad to express my sincere thanks to Mr.J.Selvaraj M.E. Lecturer Department of computer science and engineering st.Joseph's engineering college Chennai for his courteous help and valuable suggestions to complete my work.

REFERENCES

1. Hu, H., J. Xu and D.L. Lee, 2005. A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," Proc. ACM SIGMOD, pp: 479-490.
2. Yu, X., K.Q. Pu and N. Koudas, 2005. Monitoring k-Nearest Neighbor Queries over Moving Objects, Proc. IEEE Int'l Conf. Data Eng.(ICDE).
3. Xiong, X., M.F. Mokbel and W.G. Aref, 2005. SEA-CNN: Scalable Processing of Continuous k-Nearest Neighbor Queries in Spatio-Temporal Databases," Proc. IEEE Int'l Conf. Data Eng. (ICDE).
4. Ghinita, G., P. Kalnis and S. Skiadopoulos, 2007. Prive: Anonymous Location-Based Queries in Distributed Mobile Systems, Proc. Int'l World Wide Web Conf. (WWW '07), pp: 371-380.
5. Mokbel, M.F., C.Y. Chow and W.G. Aref, 2006. The New Casper: Query Processing for Location Services without Compromising Privacy, Proc. Int'l Conf. Very Large Data Bases (VLDB), pp: 763-774.
6. Gruteser, M. and D. Grunwald, 2003. Anonymous Usage of Location-Based Services through Spatial and Temporal Cloaking, Proc.MobiSys.
7. Cai, Y., K.A. Hua and G. Cao, 2004. Processing Range-Monitoring Queries On Heterogeneous Mobile Objects," Proc. IEEE Int'l Conf. Mobile Data Management (MDM).
8. Efficient Evaluation of Imprecise Location-Dependent Queries(Technical Report)Jinchuan Chen Reynold Cheng, Department of Computing The Hong Kong Polytechnic University Hung Hom, Kowloon, Hong Kong Email. fcsjcchen.
9. Jayalakshmi, V. and N.O. Gunasekar, 2013. Implementation of discrete PWM control scheme on Dynamic Voltage Restorer for the mitigation of voltage sag /swell, 2013 International Conference on Energy Efficient Technologies for Sustainability, ICEETS, pp: 1036-1040.
10. Uma Mageswaran, S. and N.O. Guna Sekhar, 2013. Reactive power contribution of multiple STATCOM using particle swarm optimization, International Journal of Engineering and Technology, 5(1): 122-126.
11. Arumugam, S. and S. Ramareddy, 2012. Simulation comparison of class D/ Class E inverter fed induction heating", Journal of Electrical Engineering, 12(2): 71-76.

12. Nagarajan, C. and M. Madheswaran, 2012. Experimental study and steady state stability analysis of CLL-T series parallel resonant converter with fuzzy controller using state space analysis, Iranian Journal of Electrical and Electronic Engineering, 8 (3): 259-267.
13. Ramkumar Prabhu, M., V. Reji and A. Sivabalan, 2012. Improved radiation and bandwidth of triangular and star patch antenna, Research Journal of Applied Sciences, Engineering and Technology, 4(12): 1740- 1748.