UDA: Update driven approach for Continuously Moving Objects

G.Sriram murthy,Ch.Jyosthna Devi,G.Bala venkata kishore

Abstract—In the present scenario location updating is main problem in mobile applications. Mobile clients will be spread based on location hierarchy. Updating all mobile clients' information at single database server faces overhead problem. To overcome this, in our approach we are providing server hierarchy for load balancing. Mobile clients generally send continuous information to servers. But if movement is local then the local server will have responsible to update data. If movement is global then central server will send data to all. Using this approach load is balanced. We develop efficient query evaluation/reevaluation and safe region computation algorithms in the approach. The experimental results show that UDA substantially outperforms traditional schemes in terms of monitoring overhead, CPU cost, and scalability while achieving close-to-optimal communication cost.

Index Terms :Spatial databases, Update Driven, Performance, Efficiency,Scalability,Computation,Object Index.

1 Introduction

In mobile and spatiotemporal databases, monitoring continuous spatial queries over moving objects has more overhead on server 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). 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—overhead, efficiency, and privacy. Overhead means how often the monitored results are correct, and it heavily depends on the frequency and overhead 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 overhead of location updates determines how much the client's privacy is exposed to the server. But server has more burdens to update all mobiles information. So considering existing approach may yield result into overhead. Server has to maintain all movements without fail. Otherwise database server will not be up-to-date. Location hierarchy for mobile clients will start from country to building based on user requests Fig 2 shows location hierarchy.

- *G.Sriram Murthy is currently pursuing masters degree program in computer science& engineering . E-mail:smurthy182@gmail.com*
- *G.Balavenkata Kishore is currently pursuing masters degree program in computer science& engineering. E-mail:g.kishore841@gmail.com*

2. Related Work

There is a large body of research work on spatial temporal query processing. Early work assumed a static data set and focused on efficient access methods (e.g., R-tree [9]) and query evaluation algorithms (e.g., [2], [7]). Recently, a lot of attention has been paid to moving-object databases, where data objects or queries or both of them move. Assuming that object movement trajectories are known a priori, Saltenis et al. [8] proposed the Time-Parameterized R-tree (TPR-tree) for indexing moving objects, where the location of a moving object is represented by a linear function of time. Benetis et al. [3] developed query evaluation algorithms for NN and reverse NN search based on the TPR-tree. Tao et al. [4] optimized the performance of the TPR-tree and extended it to the TPR*-tree. Chon et al. [1] studied range and kNN queries based on a grid model. Patel et al. [4] proposed a novel index structure called STRIPES using a dual transformation technique.

The work on monitoring continuous spatial queries can be classified into two categories. The first category assumes that the movement trajectories are known. Continuous kNN monitoring has been investigated for moving queries over stationary objects [4] and linearly moving objects [2], [6]. Iwerks et al. [22] extended to monitor distance semijoins for two linearly moving data sets [3]. However, as pointed out in [9], the known-trajectory assumption does not hold for many application scenarios (e.g., the velocity of a car changes frequently on road).

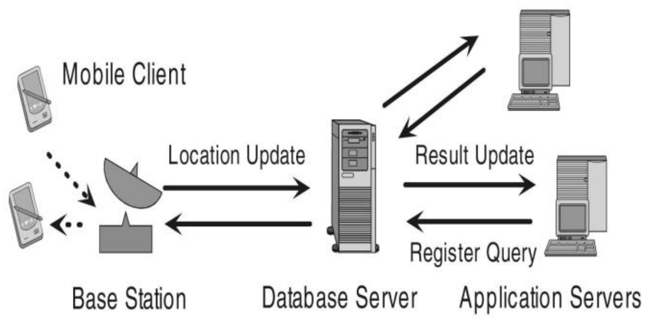

Fig 1: Basic Monitoring System

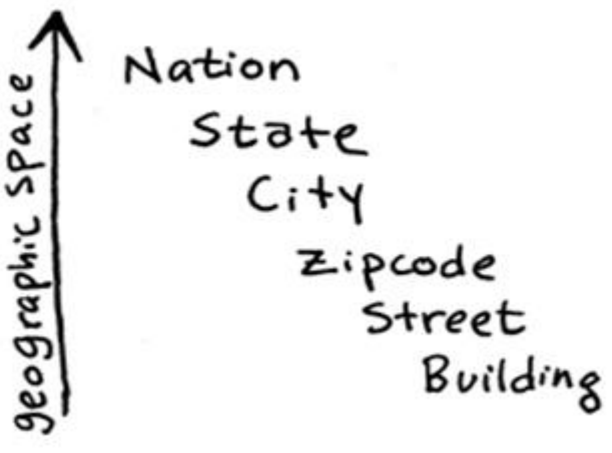

Fig 2: Geographic Space

3 UDA for moving objects

In our approach we are considering database servers into 2 types. Those are Central server and Local server. Central server will have responsible about global behavior but local server will have local behavior. If movement is local then it sends request to local server. Local server will update information. Whenever central server needed this information he requests update information from local server. Local server sends updated data. If behavior is global then central server will update data. It sends updated data to all its servers later.

The key idea to solving the problem is "safe region," which was defined in [2] as a rectangle within which the change of object location does not change the result of any registered spatial query. Now that locations are δ-squares instead of points, to clarify the definition of "within," we use the centroid point of the square as a representative, so the safe region is essentially a safe region for the centroid of the δ-square. However, the consequence of introducing δ-square is more than that—the result of a spatial query is no longer unique. For example, if the δ-square of an object partially overlaps with a range query, this object could be either a result object or a nonresult object of this query.

As such, a unique definition of query result under δ-squares is a prerequisite of safe region. Since the genuine point location of an object is distributed uniformly in its δ-square, we can define the (unique) query result as the one with the highest probability among all possible results. As in the previous range query example, if the majority of the δ-square falls inside the range query, that object is most probably a result object of this query; otherwise, that object is most probably a nonresult object. With the notion of most probable result, we thereby define the safe region as a rectangle within which the change of the centroid of the object's δ-square does not change the most probable result of any registered spatial query. The standard update strategy of the client is therefore "to update when the centroid of the δ-square is out of the safe region." The reason why we exclude all other less probable results in this definition is threefold: 1) monitoring continuous queries usually trades overhead for efficiency— although the most probable result does not always align with the genuine result (the result derived based on genuine point locations of all objects), that it is efficient to compute, and therefore, prevents the server from being computationally overloaded;
2) if the query result were defined as the set of all possible results, the safe region would have to be extremely small to report location updates if any of the possible results changes, which makes the update cost overwhelmingly high;
3) we do not want the choice of δ-square—which is made by the client—to affect query results heavily, and obviously the most probable results are less vulnerable than other result definitions.

3.1 Our approach Overview

As shown in Fig. 3, the UDA our approach 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. This assumption has been widely adopted in many existing proposals [5], [7], [1].

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.

UDA our approach works as follows. 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.

Pseudo Code:

```
 while(request)
  do
  **if(m<threshold)**
 then
send(req, applserver)
m=cal(quarantine area)
k= insert(m,query index)
return k ;
 update (safe region)
else **if(m>=threshold)** then
send(req, dbserver)
i=0;
n=no of local servers
for(i=0;i<n;i++)
reevaluate q0 ;
send(req, localserver)
update(result,applserver)
cal(area)
update(index)
update_safe region(p)
if central server (request)
o=send(req,central server)
returns o;
```

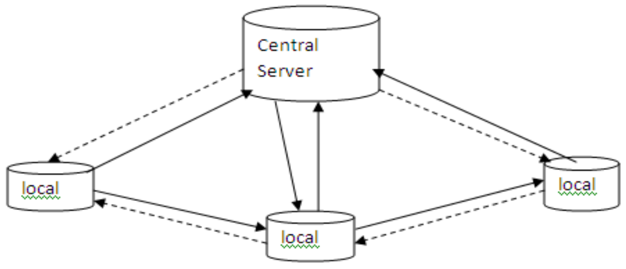

Fig 3: Communication between local and central servers.

3.2 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 δ) 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 δ /2 on each side, or formally, the "Minkowski sum"2 of the safe region and a δ /2-square.
With the same rationale for which we assume the genuine point location of an updating object to distribute uniformly in the δ -square, we assume that the genuine point locations are distributed uniformly in their respective bboxes when queries are evaluated or reevaluated. The object index is built on the bboxes to speed up the evaluation. While many spatial index structures can serve this purpose, this paper employs the R*-tree index [2], [1], which is most widely adopted in the literature. Since the bbox changes each time the object updates, the index is optimized to handle frequent updates [9].

3.3 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 no result 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 no result 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."

For a range query q, the query window can serve as an inner bound of the quarantine area, because any object whose δ -square is fully inside q is a trivial result of q. On the other hand, an outer bound can be the Minkowski sum of q and a 2-square, i.e., enlarging q by δ /2 on each side. The correctness of this bound can be verified by the observation that for any δ -square that crosses this bound, the majority of this square must be outside q, thus making the object a nonresult object. In case, there are different δs for different objects, the largest δ is used.

4 CONCLUSIONS

This paper proposes a UDA approach for monitoring continuous spatial queries over moving objects to decrease overhead also. Our approach is the first to holistically address the issue of location updating with regard to monitoring with respect to location hierarchy. We provide detailed algorithms for query evaluation/ reevaluation and safe region computation in this based on existing systems. To overcome overhead we changed algorithm to modified level. We also devise three-client update strategies that optimize overhead, privacy, and efficiency, respectively. The performance of our our approach is evaluated through a series of experiments. The results show that it substantially outperforms periodic monitoring in terms of overhead and CPU cost while achieving a close-to-optimal communication cost. Furthermore, our approach is robust and scales well with various parameter settings, such as privacy requirement, moving speed, and the number of queries and moving objects.

REFERENCES

[1] S. Babu and J. Widom, "Continuous Queries over Data Streams," Proc. ACM SIGMOD, 2001.

[2] 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.

[3]. Duri, S., Gruteser, M., Liu, X., Moskowitz, P., Perez, R., Singh, M., Tang, J.M.: Framework for security and privacy in automotive telematics. In: WMC '02: Pro- ceedings of the 2nd international workshop on Mobile commerce, New York, NY, USA, ACM (2002) 25–32

[4]. Ardagna, C.A., Cremonini, M., Damiani, E., di Vimercati, S.D.C., Samarati, P.:Supporting location-based conditions in access control policies. In: ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, New York, NY, USA, ACM (2006) 212–22224

[5]. Zibuschka, J., Scherner, T., Fritsch, L., Rannenberg, K., Goethe, J.W.: Towards a unified interface for privacy regulation-conformant location-based services. In: W3C Workshop on Languages for Privacy Policy Negotiation and Semantics- Driven Enforcement, Ispra/Italy (October 2006)

[6]. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5) (2002) 557–570 [7]. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5) (2002) 571–588

[8]. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services throughh spatial and temporal cloaking. In: MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services, New York, NY, USA, ACM (2003) 31–42

[9]. Gedik, B., Liu, L.: Location privacy in mobile systems: A personalized anonymization model. In: ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, Washington, DC, USA, IEEE Computer So- ciety (2005) 620–629

[10]. Kalnis, P., Ghinita, G., Mouratidis, K., Papadias, D.: Preserving anonymity in location based services. Technical report, National University of Singapore (2006)

[11]. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The new casper: query processing for location services without compromising privacy. In: VLDB '06: Proceedings of the 32nd international conference on Very large data bases, VLDB Endowment (2006) 763–774

[12]. Hoh, B., Gruteser, M.: Protecting location privacy through path confusion. In: SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SE- CURECOMM'05),Washington, DC, USA, IEEE Computer Society (2005) 194–205

[13]. Cheng, R., Zhang, Y., Bertino, E., Prabhakar., S.: Preserving user location privacy in mobile data management infrastructures. In: PET '06: 6thWorkshop on Privacy Enhancing Technologies. (2006)

[14]. Ghinita, G., Kalnis, P., Skiadopoulos, S.: Prive: anonymous location-based queries in distributed mobile systems. In:WWW'07: Proceedings of the 16th international conference on World Wide Web, New York, NY, USA, ACM (2007) 371–380

[15]. Ghinita, G., Kalnis, P., Skiadopoulos, S.: Mobihide: A mobile peer-to-peer system for anonymous location-based queries. In: SSTD '07: 10th International Sympo- sium on Advances in Spatial and Temporal Databases, Boston, MA, USA, Springer (2007) 221–238

[16]. Duckham,M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. In: Pervasive 05': Third International Conference on Pervasive Computing.(2005) 152–170

[17]. Schilit, B.N., LaMarca, A., Borriello, G., Griswold,W.G., McDonald, D., Lazowska, E., Balachandran, A., Hong, J., Iverson, V.: Challenge: ubiquitous location-aware computing and the "place lab" initiative. In:

WMASH ’03: Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots, New York, NY, USA, ACM Press (2003) 29–35

[18]. Chow, C.Y., Mokbel, M.F., Liu, X.: A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In: GIS ’06: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems, New York, NY, USA, ACM (2006) 171–178