PLACE: A Query Processor for Handling Real-time Spatio-temporal Data Streams*

Mohamed F. Mokbel Xiaopeng Xiong Walid G. Aref Susanne E. Hambrusch
Sunil Prabhakar Moustafa A. Hammad

Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-1398
{mokbel,xxiong,aref,seh,sunil,mhammad}@cs.purdue.edu

Abstract

The emergence of location-aware services calls for new real-time spatio-temporal query processing algorithms that deal with large numbers of mobile objects and queries. In this demo, we present PLACE (*Pervasive Location-Aware Computing Environments*); a scalable location-aware database server developed at Purdue University. The PLACE server addresses scalability by adopting an *incremental* evaluation mechanism for answering concurrently executing continuous spatio-temporal queries. The PLACE server supports a wide variety of stationery and moving continuous spatio-temporal queries through a set of pipelined spatio-temporal operators. The large numbers of moving objects generate real-time spatio-temporal data streams.

1 Introduction

Combining the functionalities of personal locator technologies, global positioning systems (GPSs), wireless and cellular telephone technologies, and information technologies enables new environments where virtually all objects of interest can determine their locations. These technologies are the foundation for pervasive location-aware environments and services. Such services have the potential to improve the quality of life by adding location-awareness to virtually all objects of interest such as humans, cars, laptops, eyeglasses, canes, desktops, pets, wild animals, bicycles, and buildings.

By enabling an upward link, the data sent from the mobile objects to the servers enables an environment in which objects are aware of the locations of surrounding objects as well as other related information. Applications can range from locating lost or stolen objects, to tracking little children and alerting parents when their children step out of the backyard, and completely automating traffic and vehicle navigation systems.

In this demo, we present the PLACE server, developed at Purdue University [1]. The PLACE server employs special query processing techniques to support scalable and incremental evaluation of a wide variety of continuous spatio-temporal queries. In particular, the PLACE server has the following distinguishing characteristics:

1. **Scalability.** The PLACE server uses a *shared execution* paradigm as a means for achieving scalability in terms of the number of outstanding continuous spatio-temporal queries.

2. **Incremental evaluation.** The PLACE server employs an *incremental* evaluation paradigm by continuously updating the user with any change to the query answer by using the notion of *positive* and *negative* tuples.

3. **New operators.** The PLACE server employs a new set of spatio-temporal incremental operators (e.g., INSIDE and kNN operators) that support scalable and incremental evaluation of continuous queries through SQL.

4. **Wide variety of continuous queries.** By encapsulating the scalable and incremental algorithms into physical pipelined query operators, the PLACE server has the ability to express and evaluate a wide variety of continuous spatio-temporal queries.

*This work was supported in part by the National Science Foundation under Grants IIS-0093116, EIA-9972883, IIS-9974255, IIS-0209120, 0010044-CCR, and EIA-9983249.

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

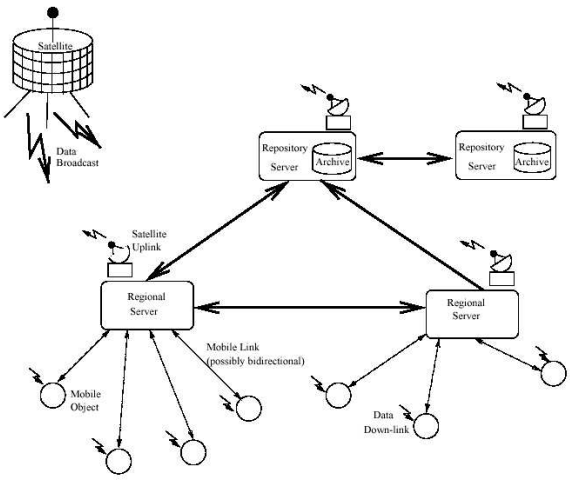

Figure 1: The PLACE Architecture.

5. **Predicate-based Sliding Windows**: We extend the notion of sliding windows beyond time-based and tuple-count windows to accommodate for predicate-based windows (e.g., an object expires from the window when it appears again in the stream).

2 The PLACE Architecture

Figure 1 sketches a hierarchical architecture of the PLACE server. Location detection devices (e.g., GPS devices) provide the objects with their geographic locations. Objects connect directly to regional servers that form the lowest level in this hierarchy. Regional servers handle the incoming data and process time-critical spatio-temporal queries. Regional servers communicate with each other, as well as with the high level servers i.e., the repository servers. Repository servers archive the past locations of moving objects.

The servers are interconnected by high bandwidth links. However, the mobile links between the regional servers and the objects have low bandwidth and a high cost per connection. For data that is being sent from the moving objects to the regional servers (i.e., in the uplink direction), we regulate the amount of data collected and the rate at which data is sent (the *upload frequency*). For query results and other data being sent from the location-aware regional servers to the objects (i.e., the downlink direction), an alternative to the point-to-point mobile links is allowed. Servers can transmit data to a satellite that broadcasts the information over the air to all objects. Broadcasting allows a server to send data to a large number of *listening* objects [4].

3 Shared Execution of Continuous Spatio-temporal Queries

The PLACE server [8, 9, 10, 12] exploits a *shared execution* paradigm as a means for achieving scalability when concurrently executing continuous spatio-temporal queries. The main idea is to group similar queries in a query table. Then, the evaluation of a set of continuous spatio-temporal queries is abstracted as a spatial join between the moving objects and the moving queries. Similar ideas of shared execution have been exploited in the NiagaraCQ [3] for web queries, PSoup [2], and [6] for streaming queries.

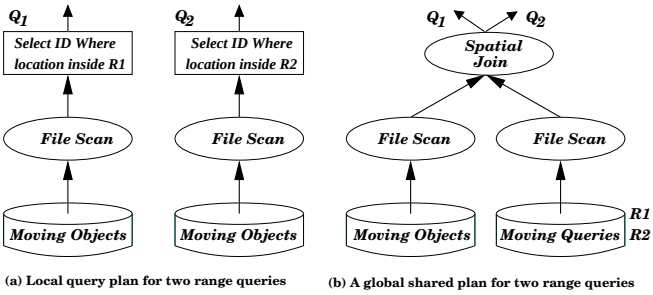

Figure 2: Shared execution of continuous queries.

Figure 2a gives the execution plans of two simple continuous spatio-temporal queries, Q_1: *"Find the objects inside region R_1"*, and Q_2: *"Find the objects inside region R_2"*. Each query performs a file scan on the moving object table followed by a selection filter. With *shared execution*, we have the execution plan of Figure 2b. The table for moving queries contains the regions of the range queries. Then, a spatial join is performed between the table of objects (points) and the table of queries (regions). The output of the spatial join is split and is sent to the individual queries.

Shared execution for a collection of spatio-temporal range queries can be expressed in the PLACE server by issuing the following continuous query:

```
SELECT Q.ID, O.ID
FROM QueryTable Q, ObjectTable O
WHERE O.location inside Q.region
```

The query processor recognizes that both the object and query tables are being updated continuously by the newly incoming objects and queries. Thus, the query is executed continuously.

4 Incremental Execution

The PLACE server employs an *incremental* execution paradigm [8, 10], where only the updates of the previously reported answer are sent to the user. We distinguish between two types of updates, namely *positive* and *negative* updates. Positive/Negative updates indicate that a certain object needs to be added to/removed from the previously reported answer, respectively. By employing *incremental evaluation*, the PLACE server achieves the following goals: (1) Fast query evaluation, since we compute only the update (change) of the answer not the whole answer. (2) In a typical location-aware server query results are sent

to the users via satellite servers [4]. Thus, limiting the amount of transmitted data to the positive and negative updates only rather than the whole query answer saves in network bandwidth. (3) When encapsulating incremental algorithms into physical pipelined query operators, limiting the tuples that go through the whole query pipeline to only the *positive* and *negative* updates reduces the flow in the pipeline. Thus, efficient query processing is achieved.

In the demo, we will show how the users and query operators interpret the *positive* and *negative* updates to continuously maintain the query results.

5 Spatio-temporal Pipelined Operators

The PLACE server encapsulates scalable and incremental algorithms into pipelined query operators. Examples of such operators are the $INSIDE$ operator for range queries and the kNN operator for k-nearest-neighbor queries. Such operators can represent both stationary and moving queries. In addition spatio-temporal queries may have a time window that allows querying recent history. A typical continuous query for the PLACE server may have the following form:

```
SELECT select_clause
FROM from_clause
WHERE where_clause
INSIDE inside_clause
kNN knn_clause
WINDOW window_clause
```

The *inside_clause* can represent stationary rectangular range or circular queries by specifying the two corners or the center and radius of the query region, respectively. If the first parameter to the *inside_clause* is M, then the query is moving and the second parameter represents the ID of the *focal* object of the query. Similarly, the *knn_clause* can represent stationary as well as moving k-nearest-neighbor queries. The *window_clause* allows having sliding window queries [5].

Incremental pipelined query operators may output *positive* or *negative* tuples. Thus, we furnish the rest of the query operators (e.g., aggregate, distinct, and joins) with special mechanisms to interpret the *negative* tuples.

6 Spatio-temporal Queries

By having the basic spatio-temporal operators, the PLACE server has the ability to evaluate a wide variety of continuous spatio-temporal queries. The following are some examples that are typical in the PLACE server:

Example I: Spatio-temporal aggregates. *Continuously, report the number of distinct cars that are inside a certain area*

```
SELECT COUNT (DISTINCT (M.ID)
FROM MovingVehicles M
WHERE M.type = "Car"
INSIDE 20,20,30,30
```

Example II: Catching speedy moving objects. *Continuously, report any car that passes through area R_2 then area R_1 in less than five minutes.* Such query can catch speedy cars that go very fast from area R_2 to area R_1.

```
SELECT DISTINCT M1.ID
FROM (SELECT ID
.       FROM MovingObjects
.       INSIDE R1) M1,
.       (SELECT ID
.       FROM MovingObjects
.       INSIDE R2
.       WINDOW 5) M2
WHERE M1.ID = M2.ID
```

7 Demo Description

The PLACE server is implemented on top of the NILE query processor [7]; an extended version of the PREDATOR database management system [11] to handle stream data. Figures 3 and 4 give snapshots of the client and server graphical user interface (GUI) of PLACE[1] .

The PLACE server GUI (Figure 3) is for the purpose of administration at the server side. The main idea is to keep track of the concurrently executing continuous queries from each type. All the processed queries along with their parameters are displayed in the bottom of the screen. In addition, the server GUI contains a regional map showing the movement of objects, and the parameters of the selected queries.

Client GUI (Figure 4) simulates a client end device used by the users. Users can choose the type of query from a list of available query types. The spatial region of the query can be determined using the map of the area of interest (the bold plotted rectangle on the map). Once the query is submitted to the server, the result appears to the query as a drop-down list at the bottom of Figure 4. A client can send multiple queries of different types to the PLACE server.

References

[1] W. G. Aref, S. E. Hambrusch, and S. Prabhakar. Pervasive Location Aware Computing Environments (PLACE). http://www.cs.purdue.edu/place/, 2003.

[2] S. Chandrasekaran and M. J. Franklin. Streaming Queries over Streaming Data. In *VLDB*, 2002.

[3] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In *SIGMOD*, 2000.

[1]The map in Figures 3 and 4 is for the Greater Lafayette area, IN, USA.

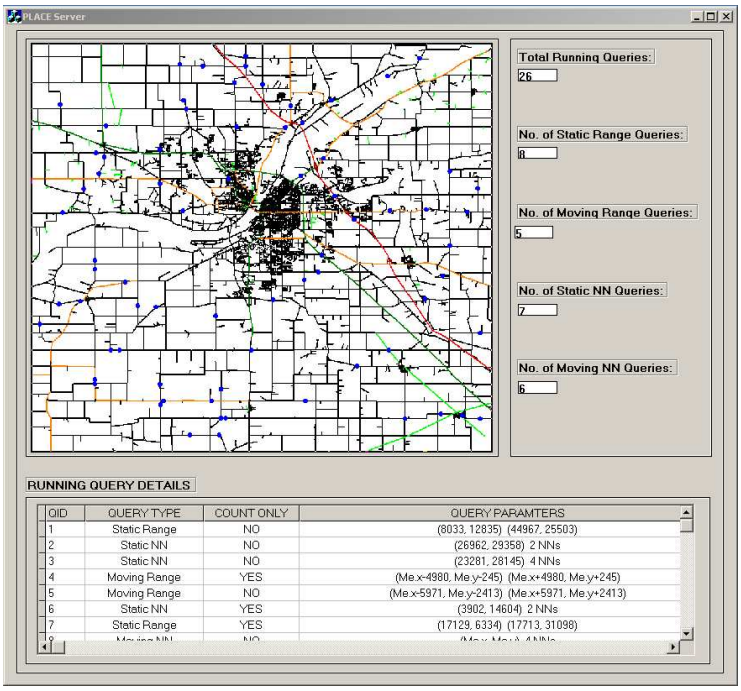

Figure 3: Snapshot of the PLACE server.

[4] S. E. Hambrusch, C.-M. Liu, W. G. Aref, and S. Prabhakar. Query Processing in Broadcasted Spatial Index Trees. In *SSTD*, 2001.

[5] M. A. Hammad, W. G. Aref, M. J. Franklin, M. F. Mokbel, and A. K. Elmagarmid. Efficient execution of sliding-window queries over data streams. Technical Report TR CSD-03-035, Purdue University Department of Computer Sciences, Dec. 2003.

[6] M. A. Hammad, M. J. Franklin, W. G. Aref, and A. K. Elmagarmid. Scheduling for shared window joins over data streams. In *VLDB*, 2003.

[7] M. A. Hammad, M. F. Mokbel, M. H. Ali, W. G. Aref, A. C. Catlin, A. K. Elmagarmid, M. Eltabakh, M. G. Elfeky, T. M. Ghanem, R. Gwadera, I. F. Ilyas, M. Marzouk, and X. Xiong. Nile: A Query Processing Engine for Data Streams. In *ICDE*, 2004.

[8] M. F. Mokbel. Continuous Query Processing in Spatio-temporal Databases. In *Proceedings of the ICDE/EDBT PhD Workshop*, 2004.

[9] M. F. Mokbel, W. G. Aref, S. E. Hambrusch, and S. Prabhakar. Towards Scalable Location-aware Services: Requirements and Research Issues. In *GIS*, 2003.

[10] M. F. Mokbel, X. Xiong, and W. G. Aref. SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases. In *SIGMOD*, 2004.

[11] P. Seshadri. Predator: A resource for database research. *SIGMOD Record*, 27(1):16–20, 1998.

[12] X. Xiong, M. F. Mokbel, W. G. Aref, S. Hambrusch, and S. Prabhakar. Scalable Spatio-temporal Continuous Query Processing for Location-aware Services. In *SSDBM*, June 2004.

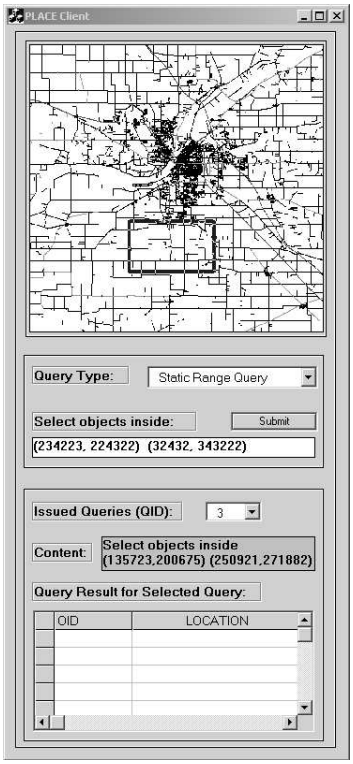

Figure 4: Snapshot of a client of the PLACE server.