Geospatial Stream Query Processing using Microsoft SQL Server StreamInsight

Seyed Jalal Kazemitabar[1] Ugur Demiryurek[1] Mohamed Ali[2] Afsin Akdogan[1] Cyrus Shahabi[1]

[1]InfoLab, Computer Science Department
University of Southern California, Los Angeles, CA 90089
{kazemita,demiryur,aakdogan,shahabi}@usc.edu

[2]Microsoft SQL Server, Microsoft Corporation
One Microsoft Way, Redmond WA 98052
mali@microsoft.com

ABSTRACT

Microsoft SQL Server spatial libraries contain several components that handle geometrical and geographical data types. With advances in geo-sensing technologies, there has been an increasing demand for geospatial streaming applications. Microsoft SQL Server StreamInsight (*StreamInsight*, for brevity) is a platform for developing and deploying streaming applications that run continuous queries over high-rate streaming events. With its extensibility infrastructure, StreamInsight enables developers to integrate their domain expertise within the query pipeline in the form of user defined modules.

This demo utilizes the extensibility infrastructure in Microsoft StreamInsight to leverage its continuous query processing capabilities in two directions. The first direction integrates SQL spatial libraries into the continuous query pipeline of StreamInsight. StreamInsight provides a well-defined temporal model over incoming events while SQL spatial libraries cover the spatial properties of events to deliver a solution for spatiotemporal stream query processing. The second direction extends the system with an *analytical refinement and prediction* layer. This layer analyzes historical data that has been accumulated and summarized over the years to refine, smooth and adjust the current query output as well as predict the output in the near future. The demo scenario is based on transportation data in Los Angeles County.

1. INTRODUCTION

Real-time stream data acquisition through sensors and probes has been widely used in numerous applications such as network monitoring, telecommunications data management, security, manufacturing, and sensor networks. Besides the temporal nature of stream data, various sources provide stream data that has geographical locations and/or spatial extents such as point coordinates, lines, or polygons. In addition to stream data generated by stationary sensors, spatiotemporal stream data is also generated by moving objects thanks to advances in GPS and wireless communication technologies. Hence, the *geo-streaming* term refers to the ongoing effort in academia and industry to process and analyze stream data with geographic and spatial information.

On one hand, Microsoft SQL Server StreamInsight (*StreamInsight*, for brevity) [1, 2, 3, 4] is a platform for stream query processing. *StreamInsight* monitors stream data to extract meaningful patterns and trends. StreamInsight analyzes and correlates data from multiple sources incrementally to yield low latency response times. The key features of *StreamInsight* include: a declarative query language that adheres to relational algebra, the ability to provide consistency and correctness guarantees on the output, run-time query composability, query fusing, and operator sharing. Moreover, *StreamInsight* is an extensible system that seamlessly integrates user defined modules in the execution query pipeline.

On the other hand, SQL Server Spatial libraries [6] (*SQL Spatial*, for brevity) provide an easy to use, robust and high performance environment for persisting and analyzing spatial data. *SQL Spatial* provides data type support for point, line and polygon objects. It also provides various methods to handle spatial data types as well as spatial index support. *SQL Spatial* Adheres to the Open Geospatial Consortium Simple Features for SQL standard [7].

This paper presents the *GeoInsight* system, an ongoing effort to extend Microsoft SQL Server StreamInsight with spatial support to provide a platform for geo-streaming applications.

1.1 Challenges in Geospatial Stream Query Processing

There have been several libraries that provide methods to perform spatial operations on spatial data types. These libraries provided flexibility to a wide range of applications in the context of traditional spatial databases. Intersection, containment, nearest neighbor and shortest route queries are example queries that invoke methods in these libraries. However, these libraries have not been designed with the continuous stream processing paradigm in mind. The non-incremental nature of the implemented algorithms prevents stream query processors from delivering real time results. The first challenge that faces geo-streaming is to port geospatial libraries to the streaming domain with the incremental single-pass processing model in mind.

The second challenge that faces streaming application is the ability to make a decision based on the continuous query result. The real time nature of the continuous query output has a two-sided effect. The first side provides the instantaneous response to an incoming event. For example, an accident on one of the roads

Articles from this volume were presented at The 36th International Conference on Very Large Data Bases, September 13-17, 2010, Singapore.
Proceedings of the VLDB Endowment, Vol. 3, No. 2
© 2010 VLDB Endowment 21508097/10/09... $10.00

is given an immediate attention by the system to reroute traffic accordingly. On the other side, continuous query results are susceptible to noise in the incoming stream values. Using the same example, the accident is "*usually*" expected to be cleared in *T* time units and the traffic is expected to revert to its daily or weekly pattern. Over-reacting to a single incoming event may lead to sub-optimal results. In general, the ability to make decisions based on stream query processing necessitates the need for a system that weighs the decision according to real time behavior as well as historical data patterns.

1.2 Contributions

We present a real-world data-driven framework, called *GeoInsight*, built on *StreamInsight*. The proposed framework extends *StreamInsight* in two directions. First, we integrate Microsoft SQL Server Spatial Libraries into Microsoft StreamInsight to support the online processing of geo-stream data. Special attention is given to incrementally evaluate spatial operations. Second, we implement an online *analytical refinement and prediction* layer that enables querying of historical (archived) stream data. The former extension allows users to issue various real-time spatiotemporal queries about continuous events with user-defined temporal and spatial window. The latter extension enables the analysis of historical data (together with the real-time data) to refine the answer of real-time queries and predict the answer in the near future.

This paper demonstrates the basic features of the *GeoInsight* system in the context of traffic management in Los Angeles County. *GeoInsight* uses spatiotemporal real transportation data set obtained from RIITS (Regional Integration of Intelligent Transportation Systems) [5]. The RIITS dataset is collected by various organizations located in Los Angeles County including Caltrans D7, MTA-Metro, LADOT, and CHP. This dataset includes inventory and real-time traffic data for freeways. It includes arterial traffic sensor data and closed circuit television (CCTV) snapshots. Traffic sensor data is acquired from a collection of 6300 sensors located on the highways and main streets at the boundaries of Los Angeles County. Each sensor acquires readings at the rate of one reading per minute.

The remainder of this paper is organized as follows: Section 2 overviews the architecture of the *GeoInsight* system. Section 3 describes the demo scenario. The paper is concluded in Section 4.

2. ARCHITECTURE

With GeoInsight, we implemented and integrated three main components, namely StreamInsight, Spatial Cartridge, and Online Analytical Refinement and Prediction (OARP) module. Figure 1 shows the architecture of GeoInsight. Below, we explain these components in detail.

2.1 StreamInsight

GeoInsight employs StreamInsight to enable complex event processing. The run-time component of this platform includes Input Adapter, Streaming Engine, and Output Adapter. The Input Adapter provides interfaces to event sources (e.g., devices or sensors) and directs source data into Streaming Engine. The Streaming Engine runs pre-defined queries on input events. Special optimization techniques such as query fusing, operator sharing, and query and stream partitioning are employed in the Streaming Engine [2]. The Output Adapter connects streaming engine to events sinks such as user interfaces, monitoring devices and databases.

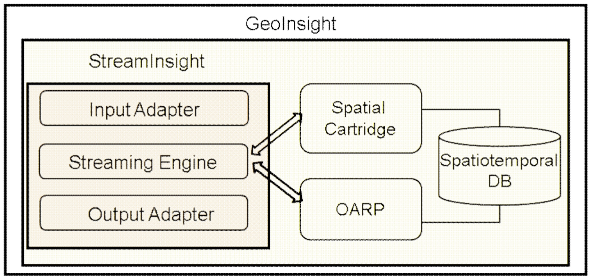


Figure 1 GeoInsight Architecture

2.1.1 Input Adapter

With our system, we implemented an input adapter that continuously listens on a TCP port for the streaming data. Once a packet of data is retrieved, the corresponding events are created and enqueued to the streaming engine. After inserting each event to the streaming engine, the input adapter verifies that the engine has not rejected any event. Finally, the input adapter declares the completeness of the existing events in the stream by inserting a current time interest (CTI) event type.

2.1.2 Streaming Engine

The streaming engine continuously executes the queries on the events received from the input adapter and subsequently forwards the results to the output adapter.

The query language supported by streaming engine is Language Integrated Query (LINQ) which supports various conventional SQL operators such as "group by" and "top-k". In order to make LINQ more expressive, the extensibility infrastructure of StreamInsight provides user-defined aggregate (UDA), user-defined operator (UDO), and user-defined function (UDF) facilities. While UDA enables to aggregate events over a user specified window by producing a single value, UDO is used to generate new events based on the existing events in that window. UDF allows users to use any complex expression in the streaming engine. With our implementation, we use UDOs and UDAs to seamlessly integrate our Spatial Cartridge and OARP modules into query pipeline of StreamInsight.

2.1.3 Output Adapter

The output adapter repeatedly dequeues the processed events form streaming engine and transfers them to defined user interfaces. With our implementation, we buffer the output events in XML format which enables flexible usage in variety of user interfaces.

2.2 Online Analytical Refinement and Prediction Layer

In general, the focus of stream data processing systems is to consume the incoming events immediately by providing instantaneous responses. However, decision making solely based on these instant responses is not ideal as the output may possibly be based on noisy or missing inputs. Therefore, it is essential to analyze (and refine) the real-time results together with the historical data patterns accumulated over the years. In addition to analysis and refinement, the historical data can be utilized to predict the future outputs.

The main challenge with the online refinement and prediction of streaming data is real-time access to large archived datasets. Due to high retrieval costs, conventional DBMS solutions cannot be used to accomplish the real-time access. Therefore, we address this challenge by considering a *sketch* based online analytical refinement and prediction module. Clearly, a sketching technique which can keep an abstract version of the dataset in memory, while keeping the maximum data precision is the ultimate add-on that can keep up with the real-time needs of the underlying stream management system.

We have adopted Principal Component Analysis (PCA) [4] to implement our main-memory sketches. Towards that end, we store a sketch of archived historical data (i.e., small number of principal components and a transformed dataset) in the main memory. The main idea behind this method is to represent a multivariate dataset using the smallest possible number of new variables (coordinates) that are selected based on the statistical characteristics of that specific dataset. The method returns a sorted list of new coordinates with their corresponding importance in representing data. Compressing the data by dropping the last coordinate has a negligible effect on data accuracy.

We have observed that the more correlated is the streaming data, the less number of components are needed to create accurate and concise sketches. For example, applying PCA to our huge traffic dataset resulted in sketches that save 98% of the initial data size while keeping the error rate as low as 10^{-4} miles per hour. Considering that geo-streaming data is usually correlated (in both space and time) our proposed PCA based sketches will yield a high compression rate with such datasets.

With our implementation, we use PCA based sketches in UDAs to enable streaming engine in answering all hybrid queries that define spatio-temporal windows over historic[1] and online streaming data simultaneously. Given efficient access to historical data through UDAs, one can accomplish the following refinement and prediction functions:

Refinement functions:

- Substituting missing streaming data with corresponding historical values for better visualization and decision making
- Smoothing noisy input data according to previously observed patterns
- Detection of anomalies characterized by sensor readings that are highly deviated from historical mean values

Prediction functions:

- Predicting near future trends based on previously observed patterns
- Responding to anomalies and deliberately attempting to change future conditions

2.3 Spatial Cartridge

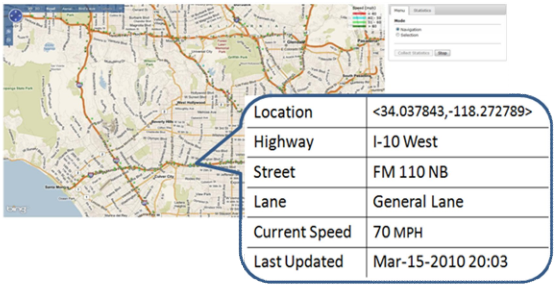
As mentioned, geo-streaming applications often need to join stream data with materialized relational tables that store spatial information about static objects, e.g., bus stops, gas stations, etc. These applications reference off-the-shelf spatial modules (e.g., Microsoft Sql Server Spatial Library) to perform a specific form of spatial joins. To realize such functionality we implemented a Spatial Cartridge that allows users to register desired spatial relations and packages together with continuous streams of data. Specifically, with the Spatial Cartridge, we have developed integrated set of functions and procedures that enable retrieval and processing of spatial data easier and more efficiently.

Spatial cartridge extends the capabilities of StreamInsight (i) by determining the way StreamInsight retrieves and interprets the underlying spatial information (e.g., road networks), and (ii) by customizing the spatial operators and idexes for efficient access to large spatial data. This way users can easily implement the functions or interfaces that have the specialized behavior required in the geostreaming applications.

3. DEMO SCENARIO

In this section, we present various types of queries that are posed against our traffic sensor data. As we mentioned, we receive events from 6300 sensors located in Los Angeles County road network. In our demo, the user queries real-time spatiotemporal statistics about traffic conditions of freeways in Los Angeles transportation network. The query result is updated continuously as new data is streamed into the system's input buffers. The query is issued and the query result is retrieved and visualized using an interactive web based map interface that is built on top of *Microsoft Bing Maps*.

Figure 2 shows an example query flow deployed over *GeoInsight*. The first query (Q_1) filters out noisy data (e.g. sensors providing zero speed events) from the sensor readings. The output stream produced by this query contains the current speed, location, and other relevant information (e.g., last-update time). The output of Q_1 is fanned out to two streams. The first stream is directly forwarded to the map interface (i.e., monitoring interface) where sensors are color-coded based on their speed reading values (e.g., red if speed is less than 10 km/h). Users can also see detailed information (e.g., street name, last-update time) by scrolling the mouse over the sensor icons (see Figure 3). The second stream is passed through a spatial filter (Query Q_2).



Location	<34.037843,-118.272789>
Highway	I-10 West
Street	FM 110 NB
Lane	General Lane
Current Speed	70 MPH
Last Updated	Mar-15-2010 20:03

Figure 3. Color-coded sensors in LA freeways

In Q_2, the user selects an area on the map (e.g. a segment in a highway) to collect statistics and analyze the data in the selected area. As shown in Figure 4, *GeoInsight* shows a refined current average speed value for all the sensors in that area. What happens in the underlying system is that (i) query Q_2 is started in the

[1] With our system, 'historic' no more means 'archived' as archived brings to mind a less frequently used medium with less retrieval speed.

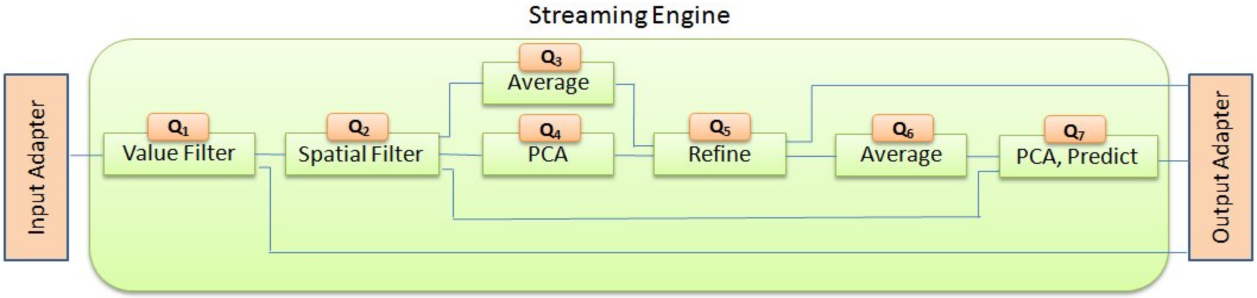
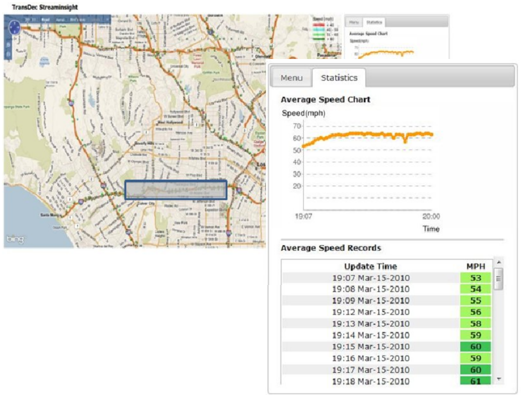


Figure 2 Stream flow and query sequence in GeoInsight

streaming engine; (ii) the output of Q_1 is streamed to Q_2 as input; (iii) Q_2 uses a UDA that uses Spatial Library functions to perform a spatial filter on sensor readings considering the selected rectangle chosen by the user.

For various technical reasons sensors may not provide correct values. Fortunately, we can use our OARP module to provide more realistic average values to the end-user by refining the raw average value with a historical average (obtained from archived sensor readings). One possible method is to find a weighted average based on former historic observations and current data and report the calculated value as a more reliable average speed for the selected area. The system gives the user the flexibility to configure the impact of raw average speed with respect to the historically observed mean value to formulate the final output.

Continuing with our existing scenario, Q_3 calculates the raw average speed based on sensor readings selected in Q_2 and at the same time Q_4 uses PCA as a UDA to aggregate the corresponding speed readings observed in historic data. Finally, Q_5 merges the (possibly noisy) result obtained from Q_3 with that of Q_4 based on user-specified weighting parameters. As shown in Figure 4, the calculated value in Q_5 will be monitored on the client side as a smoothed average speed for the selected area. The diagram on the right window shows the calculated average value per minute.



Average Speed Records

Update Time	MPH
19:07 Mar-15-2010	53
19:08 Mar-15-2010	54
19:09 Mar-15-2010	55
19:12 Mar-15-2010	56
19:13 Mar-15-2010	58
19:14 Mar-15-2010	59
19:15 Mar-15-2010	60
19:16 Mar-15-2010	59
19:17 Mar-15-2010	60
19:18 Mar-15-2010	61

Figure 4. Online refined average speeds for a user-selected area

As we mentioned, *GeoInsight* enables the prediction of future trends. We demonstrate this feature on our traffic dataset by predicting the average speed for the rest of the day for a user-selected area. Q_6 calculates the average refined values for a defined duration (say k min.). Subsequently, Q_7 predicts the average speed for the rest of the day by finding the date in history in which the selected sensors had the closest average speed in the past k minutes.

4. CONCLUSION

In this paper, we introduced the *GeoInsight* system, which enables interactive and extensive spatiotemporal querying and refinement of geo-streaming data. GeoInsight extends *Microsoft StreamInsight* by integrating Microsoft SQL Server Spatial Libraries and an Online Analytical Refinement and Prediction module. We presented an overview of the system design and proposed a demo scenario that is based on the traffic data in Los Angeles County. The demo is based on spatiotemporal continuous queries that join incoming stream data from traffic sensors spatially with rectangular region as well temporally with historical data.

We intend to pursue this work in two directions. First, we plan to extend the capabilities of *GeoInsight* to support more complex ad-hoc analytical queries. Second, we intend to port continuous spatial queries (e.g., continuous kNN and range query) to GeoInsight

5. ACKNOWLEDGMENTS

This research has been funded in part by NSF grants CNS-0831505 (CyberTrust), the NSF Center for Embedded Networked Sensing (CCR-0120778) and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

6. REFERENCES

[1] R.S. Barga, J. Goldstein, M. Ali, and M. Hong. *Consistent Streaming Through Time: A Vision for Event Stream Processing*. In CIDR, 2007.

[2] M. Ali et al.: *Microsoft CEP Server and Online Behavioral Targeting*. In VLDB 2009.

[3] B. Chandramouli, J. Goldstein, and D. Maier. *On-the-fly Progress Detection in Iterative Stream Queries*. In VLDB, 2009.

[4] I. T. Jolliffe: *Principal Component Analysis*. Springer, second edition, October 2002.

[5] RIITS: *http://www.riits.net*. Last accessed in March, 2010

[6] SQL Server Spatial Libraries. *http://www.microsoft.com/sqlserver/2008/en/us/spatial-data.aspx.* Last accessed in March, 2010

[7] Open Geospatial Consortium. *http://www.opengeospatial.org.* Last accessed in March, 2010