SGST: An Open Source Semantic GeoStreaming Toolkit

Jong Lee, Yong Liu, Liang Yu
National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign
1205 W Clark St.
Urbana, IL 61801, USA
{jonglee, yongliu, liangyu}@ncsa.illinois.edu

ABSTRACT
Geo-referenced data streams (geostreams) have created major challenges on streaming data management, query, and integration. Semantically managing geostream data has the potential to provide better data integration and reasoning. We develop an open source Semantic GeoStreaming Toolkit (SGST) that aims to provide an integrated sensor data management solution. In particular, this paper uses the Open Geospatial Consortium (OGC) GeoSPARQL recommendation and a time-annotated RDF streaming data management service. The toolkit offers geostreaming data management, fetching, and RESTful web services. We demonstrate SGST with two real-world use cases including USGS earthquake GeoRSS feeds and geo-referenced Twitter feeds for citizen sensing. Issues related to interoperability, performance, and full support of GeoSPARQL are discussed.

Categories and Subject Descriptors
H.2.8 **[Database Management]**: Systems – *query processing;* H.3.5 **[Information Storage and Retrieval]***: On-line Information Services* | *data sharing, web-based service*

General Terms
Management, Design, Standardization

Keywords
GeoStreaming, RDF, spatial index, temporal index, query, RESTful Web Service, geostreaming data fetcher

1. INTRODUCTION
Geo-referenced data streams are abundant from many sources. For example, sensor networks deployed in the natural and engineered environment provide continuous observation data streams about our physical environment such as temperature, humidity, rainfall, traffic and earthquakes [1]. In addition, a recent study by the Pew Internet Project finds that over 35% of American adults own smartphones as of May 2011 [2]. Since smartphones usually are equipped with GPS sensors, which allow users to produce geo-referenced data streams such as geo-tagged microblogs in Twitter.com, a massive "citizen sensing" data stream [3]. A "Virtual Sensor" that is derived through spatial, temporal and thematic transformations of raw sensor data streams and then republished as a new data stream is yet another example of geostreams [4]. Collectively, we refer all these geo-referenced sensor data streams as geostream data (or geostreams).

Because of the heterogeneity of streaming data and the need for sensor data integration, sharing, provenance tracking and streaming reasoning [5], there is a growing trend in building streaming data management systems by leveraging and extending W3C Semantic Web technologies such as the Resource Description Framework (RDF) and SPARQL (query languages for RDF data). This is in contrast to the approaches taken by previous non-RDF-based Data Stream Management Systems (DSMS) such as STREAM [6] and GeoInsight [7].

However, there are two main obstacles in using RDF for geostreaming data management. First, there is no direct temporal streaming support in the W3C RDF specification. Various efforts have been made such as C-SPARQL [8], StreamingSPARQL [9], and SensorMasher [10] to support RDF streams. Our previous work also developed a Time-Annotated RDF (TA-RDF) model which manages time-series data [11]. Second, there is no direct query support of geospatial information in the W3C RDF specification. The Open Geospatial Consortium (OGC) recently announced a candidate GeoSPARQL standard that defines an RDFS/OWL vocabulary and a set of spatial extension functions for SPARQL. Offering rich spatial queries of RDF, GeoSPARQL will increasingly blur the line between the Semantic Web and the Geo Web. This paper takes it one step further by combining our TA-RDF approach and GeoSPARQL recommendations to build an open source Semantic GeoStreaming Toolkit (SGST), which is urgently needed for the broad geosensor data community and the Linked Open Data community in general. Furthermore, in order to lower the barrier for the community to adopt SGST, a set of Representational State Transfer (REST)-ful Web Services for querying geostreaming was also developed, in the same spirit as "Linked Data API" [12].

In this paper, we present SGST by describing the architectural design and implementation (Section 2), followed by two case studies (Section 3). We discuss issues in current implementation and future works (Section 4), as well as related work (Section 5). Finally, we conclude in Section 6.

2. ARCHITECTURE AND IMPLEMENTATION
The SGST consists of three main components: 1) geostream data fetchers, 2) a geostream data manager, and 3) geostream RESTful services as shown in Figure 1. The toolkit provides the following capabilities: managing semantic geostream (spatiotemporal) data; fetching real-time geostream data from various sources; and querying geostream data with spatiotemporal filters via RESTful services. Below we elaborate on some of the technical details of these components.

IWGS'11, November 1, 2011, Chicago, IL, USA.

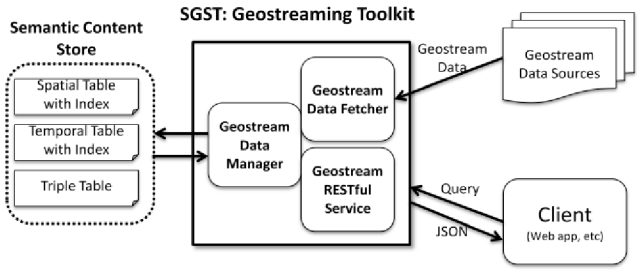

Figure 1. Architecture Diagram of SGST

2.1 Geostream Data Fetchers

The geostream data fetchers continuously retrieve real-time geostream data from various sensor data sources and translate them into RDF triples. The generated spatiotemporal RDF triples are stored into a semantic repository via the geostream data manager. A fetcher extracts out three parts: 1) temporal information, 2) spatial information (e.g., latitude and longitude, or address), and 3) thematic data value (e.g., earthquake magnitude, tweets etc.). Various fetchers have been implemented based on different transporting protocols (e.g., SFTP, telnet, HTTP, GeoRSS, and Twitter Streaming API etc.) with corresponding parsers to perform the extraction services.

2.2 Geostream Data Manager

The geostream data manager (GDM) utilizes spatial and temporal indexes using indexed tables in relational databases. Spatiotemporal and other information extracted from the fetchers are stored in three sets of tables: 1) a set of tables that support temporal index, 2) a set of tables that support spatial index, and 3) a RDF triple repository. The GDM processes any incoming spatiotemporal query using these indexed tables and returns aggregated final results by combining the query results from these different tables. Currently GDM uses Tupelo [15], a semantic middleware similar to Jena or Sesame, to interface with the storage layer. A single open source database (PostgreSQL 8.4 with PostGIS 1.5 (spatial extension)) is used for the entire geostreaming data store, as shown in Figure 1.

In order to implement geostream triples, TA-RDF [11] is used to represent temporal information and the ontology of GeoSPARQL to represent spatial information. TA-RDF is an extension of the RDF model where resources are optionally annotated with a time value (e.g., `usgs:1hour-M1["2011-07-18Z-06:00"^^xsd:date]`). The Well-Known Text (WKT) format [13] is used to represent the spatial geometry, and spatial functions from GeoSPARQL (e.g. `within`) is implemented [14]. Since spatial functions in PostGIS are defined OGC specifications, it's trivial to map OGC predicates to PostGIS functions [14][16].

2.3 Geostream RESTful Services

The geostream RESTful services provide users, or client applications easy-to-use query capabilities over geostream data without using any SPARQL query statements. Currently we support window query-based RESTful services.

A "window query" allows users to define a time window and a geospatial boundary. The API has the following format: `http://[hostname]/window/start/{start}/end/{end}/op/{op}/geom/{geom}/format/{format}/uri/{uri}`. Table 1 lists the details of the parameters of the geostream RESTful services. A service request searches geostream named {uri} in a time window between {start} and {end} with a spatial operation {within} of the geometry {geom}. If users do not specify a URI, the system will search all geostreams within the spatiotemporal constraints.

Table 1. Geostream RESTful Services Parameters

Parameter	Description
start	Start time in UTC (Coordinated Universal Time) format
end	End time in UTC format
op	Spatial operation (e.g., within, overlap, etc)
geom	Geometry for the spatial operation in WKT format
format	Format of the results such as JSON, KML, etc.
uri	URI of geostream (Optional)

3. CASE STUDIES

To demonstrate the toolkit capabilities, we present the following case studies.

3.1 USGS Earthquake GeoRSS

USGS publishes latest real-time earthquake information including location and magnitude via GeoRSS [17]. We developed a fetcher that retrieves the GeoRSS feed and extracts the spatial location, timestamp, and the magnitude value. Extracted information is translated into triples by the fetcher and stored into the database.

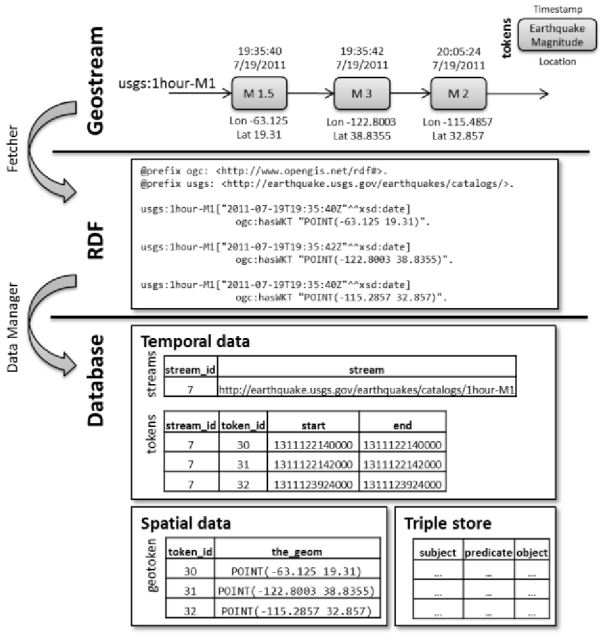

Figure 2. An Example of Simplified Schematic View of How SGST Stores Geostream Data into the Database

An example shown in Figure 2 depicts how the geostream data is fetched, transformed to RDF triples, and stored into the database. For example, the first triple in Figure 2, which means an earthquake at 19:35:40 on 2011-07-21 at location (40.119697, -88.243217),which is also shown in Figure 3, is processed in the following steps: 1) separating RDF resources into Time-Annotated (TA) resource (`usgs:1hour-M1["2011-07-19T19:35:40Z"^^xsd:date]`) and non-TA resources (`ogc:hasWKT "POINT(-88.243217 40.119697)"`); 2) extracting time information from the TA resource (`"2011-07-19T19:35:40Z"`) and storing the timestamp to *tokens* table along with a stream URI (`usgs:1hour-M1`) to the *streams* table. If a predicate among non-TA resources is

<ogc:hasWKT>, the literal "POINT(-88.243217 40.119697)" will be stored in the spatial table, *geotoken*.

A spatiotemporal query can be performed either using a SPARQL query statement or the geostream RESTful services. For example, a query that wants to find earthquakes that occurred between 2011-07-18Z-06:00 and 2011-07-23Z-06:00, and within a geographical area, is processed in a sequential two-step procedure (see Figure 4 for the complete SPARQL query statement): 1) the non-spatial query is processed using the temporal table (*tokens*) and regular triple tables; 2) the spatial query, <ogc:within>, with the given polygon in WKT format will be processed over the result from the previous step by using a PostGIS function, ST_WITHIN, and a spatial table (*geotoken*).

An example window query using the geostream RESTful services is shown in Figure 5. This query finds earthquakes occurred between 23:26:40 GMT-05:00, 7/29/2011 and 16:36:40 GMT-0500, 7/31/2011 within a given geographic bounding box. Figure 6 shows a portion of the query results in JSON format.

```
@prefix ogc: <http://www.opengis.net/rdf#>.
@prefix usgs:
   <http://earthquake.usgs.gov/earthquakes/catalogs/>.
usgs:1hour-M1["2011-07-19T19:35:40Z"^^xsd:date]
   ogc:hasWKT "POINT(-88.243217 40.119697)".
```

Figure 3. An Example Geostream RDF Triple in N3 Notation

```
SELECT ?s, ?wkt
WHERE {
?s["2011-07-18Z-06:00"^^xsd:date   ..   "2011-07-23Z-
06:00"^^xsd:date], <ogc:hasWKT>, ?wkt .
?s["2011-07-18Z-06:00"^^xsd:date   ..   "2011-07-23Z-
06:00"^^xsd:date], <ogc:within>, "POLYGON((-124.453125
53.4375,  -73.125  52.03125,  -73.828125  8.4375,  -
121.640625 9.140625, -124.453125 53.4375))" }
```

Figure 4. A Geostream Spatiotemporal Query in SAPRQL

```
http://[hostname]/window/start/1312000000000/end/13121
47981274/op/within/geom/POLYGON((-140.625
73.125,156.09375 74.53125,151.875 -46.40625,-146.25 -
47.109375,-140.625 73.125))
```

Figure 5: A Window Query Using the RESTful Services

```
{
  "streamURL":
"http://earthquake.usgs.gov/earthquakes/catalogs/1hour-M1",
  "frames": [{
    "UTCtime": 1311122140000,
    "dateTime": "Tue Jul 19 2011 19:35:40 CDT",
    "lon": -63.125,
    "lat": 19.31,
    "data": 1.5
  },
  ……..
```

Figure 6. A Portion of the Query Results in JSON

3.2 Twitter Stream

Twitter provides a set of APIs allowing users to query the tweets for citizen sensing purpose. We use the Twitter Stream API (TSA) [20] to implement a real-time fetcher and a filter so that every newly posted tweet that matches this filter will be pushed to our repository. Because the current TSA supports both spatial and keyword filters individually but not jointly, we choose to apply the spatial filter on the Twitter side, which can filter out a large portion of non-geospatial data, and then apply keyword filters locally. We use Yahoo Geocoding service to get the coordinates for those tweets that do not have explicit coordinate information. The tweets are then re-published into different geostreams, and indexed by their spatial-temporal attributes, while the contents of the tweets are treated as blobs at this moment. In addition, a set of RESTful APIs are deployed to serve the fetched data as new streams.

For example, Figure 7 shows a query over a single stream within the boundary of Champaign-Urbana, Illinois, and with a time interval of 90 days prior to April 4th, 2011. The URI "urn:streams/searchingCars/twitter" is the streaming ID of a stream which has tweets containing the keyword "car". Figure 8 shows the visualization of the query results in Google Earth by exporting the results in a time-annotated KML format.

```
http://[hostname]/window/start/1294117200000/end/13018
93200000/op/within/geom/POLYGON((40.421879 -88.529867,
39.818319 -88.529867, 39.818319 -87.618940, 40.421879
-87.618940, 40.421879 -
88.529867))/format/kml/uri/urn:streams/searchingCars/t
witter
```

Figure 7. An Example Twitter Stream Window Query

Figure 8: Query Results in KML over a Single Twitter Stream

4. DISCUSSION

4.1 Interoperability and Integration

The advantage of using semantics in geostreaming data management is particularly evident when multiple heterogeneous sensor data streams are needed for a scientific application. For example, an emergency management system might need to use both the earthquake's information as well as citizen reports in tweets to provide timely situational awareness and decision support. Since the geostream data are stored using a common model (TA-RDF and GeoSPARL) in SGST, it is easy to provide integrated queries over multiple sensor streams with better interoperability. For example, the example query in Figure 5 can retrieve both earthquakes and twitter feeds that satisfy the specified spatiotemporal constraints.

4.2 Query Performance

As explained in Section 3.1, the spatiotemporal query is performed in a sequential step-wise approach (i.e., the results from a temporal query are evaluated using a spatial SQL function (e.g., within) one by one). This means that if there are 15 results from a temporal query, there will be 15 spatial SQL queries executed. Such a constraint is imposed by the Tupelo middleware we use, where federated query processing among different databases can be supported. If the spatiotemporal index is stored in the same database, which is the case of our current implementation, a spatiotemporal query could be done in one step with one combined SQL query. We expect it will potentially enhance the query performance if implemented.

4.3 Full GeoSPARQL Support

Our current implementation supports a minimal set of the ontology in the GeoSPARQL draft such as <ogc:hasWKT> and <ogc:within>. A full GeoSPARQL support could bring a better capability. For example, if a user wants to search temperature data from all sensors within a particular watershed, the query could be performed in GeoSPARQL as shown in Figure 9, where <ncsa:Watershed_A> can be defined as an instance of a GeoSPARQL feature which contains the boundary polygon of a watershed named "Watershed_A". We will gradually add more GeoSPARQL support when the standards become mature and stable.

```
PREFIX geo:  <http://www.opengis.net/ont/OGC-
GeoSPARQL/1.0/>
PREFIX ncsa: <http://www.ncsa.illinois.edu/IACAT/>

SELECT ?sensor ?temperature
WHERE {
  ?sensor geo:within ncsa:Watershed_A .
  ?sensor ncsa:hasTemperature ?temperature
}
```

Figure 9. A Possible GeoSPARQL Query Example

5. RELATED WORK

The most relevant work to our SGST is SPARQL-ST [19] and stSPARQL [18]. Both can handle spatiotemporal queries over RDF data. However, we use TA-RDF to handle streaming data explicitly and GeoSPARQL recommendations for geospatial meta-data, which are different from prior works. In addition, we implement specific fetchers and RESFful APIs, which were not discussed before in the literature. It is expected that SGST will contribute to the geostreaming data management community with such new features and semantically-enhanced capabilities.

6. CONCLUSION

In this paper, we presented our SGST toolkit's architecture with three components: a Geostream Data Manager, Fetchers, and RESTful services. Currently this toolkit and its documentation are available at [21]. We presented two case studies to illustrate how the toolkit can be used. We expect SGST will continue to play a major role in our ongoing work in building a real-time virtual environmental observatory where heterogeneous geostreams from different sensors will be managed and integrated [22].

7. ACKNOWLEDGMENTS

The authors thank the IACAT Virtual Environmental Observatory project at the University of Illinois at Urbana-Champaign and Microsoft Research for partially funding this work.

8. REFERENCES

[1] Butler, D. 2006. 2020 Computing: Everything, everywhere. Nature, 440, 7083, 402-405. DOI=10.1038/440402a.

[2] Smith, A. 2011. "Smartphone Adoption and Usage". Available at: http://pewinternet.org/Reports/2011/Smartphones.aspx Accessed on July 23, 2011.

[3] Sheth, A. 2009. Citizen sensing, social signals, and enriching human experience. IEEE Internet Comput., 13 (4), 87-92. DOI=10.1109/MIC.2009.77.

[4] Liu ,Y.; Futrelle, J.; Myers, J.; Rodriguez, A.; Kooper, R. 2010. A provenance-aware virtual sensor system using the Open Provenance Model, Collaborative Technologies and Systems (CTS), 2010 International Symposium on , vol., no., pp.330-339, 17-21 May 2010. doi: 10.1109/CTS.2010.5478496

[5] Barbieri, D., Braga, D., Ceri, S., Della Valle, E., & Grossniklaus, M. 2010. Stream reasoning: Where we got so far. In S. Ceri, E. Della Valle, J. Hendler, & Z. Huang (Eds.), Proceedings of the 4th workshop on new forms of reasoning for the Semantic Web: Scalable & dynamic (pp. 1–7).

[6] Arasu, A., Babu, S. and Widom, J.. 2006. The CQL continuous query language: semantic foundations and query execution. The VLDB Journal 15, 2 (June 2006), 121-142. DOI=10.1007/s00778-004-0147-z

[7] Kazemitabar, S. J., U. Demiryurek, M. Ali, A. Akdogan, and Shahabi, C. 2010. Geospatial stream query processing using Microsoft SQL Server StreamInsight. Proc. VLDB Endow. 3, 1-2 (September 2010), 1537-1540.

[8] Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M. 2009. C-sparql: Sparql for continuous querying. In: WWW. 1061-1062

[9] Bolles, A., Grawunder, M., Jacobi, J. 2008. Streaming SPARQL: Extending SPARQL to Process Data Streams. In: Proc. Europ. Semantic Web Conf. (ESWC). 448-462

[10] Le-Phuoc, D., Parreira, J. X.,Hausenblas, M. and Hauswirth, M. 2010. Unifying Stream Data and Linked Open Data. Technical Report, DERI,Galway, July 2010.

[11] Rodriguez, A., McGrath, R., Liu, Y., Myers, J. 2009. Semantic Management of Streaming Data. In: Proc. Intl. Workshop on Semantic Sensor Networks (SSN).

[12] Linked Data API, http://code.google.comp/linked-data-api/

[13] Well-Known Text, http://en.wikipedia.org/wiki/Well-known_text

[14] OGC Simple Feature Specification, http://www.opengeospatial.org/standards/sfs

[15] Futrelle, J., Gaynor, J., Plutchak, J., Myers, J. D., McGrath, R. E., Bajcsy, P., Kastner, J., Kotwani, K., Lee, J. S., Marini, L., Kooper, R., McLaren, T. and Liu, Y. (2011), Semantic middleware for e-Science knowledge spaces. Concurrency and Computation: Practice and Experience, 23. doi: 10.1002/cpe.1705

[16] PostGIS Documentation, http://postgis.refractions.net/documentation/manual-1.5/reference.html

[17] GeoRSS, http://georss.org

[18] Koubarakis, M., Kyzirakos, K. 2010. Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL. ESWC (1) 2010: 425-439. Doi: 10.1007/978-3-642-13486-9_29

[19] Perry, M. 2008. A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data. PhD thesis, Wright State University

[20] Twitter Streaming API, https://dev.twitter.com/docs/streaming-api

[21] Semantic Geostreaming Toolkit, https://opensource.ncsa.illinois.edu/confluence/display/SGST/Semantic+Geostreaming+Toolkit

[22] IACAT Virtual Observatory Project, http://iacat.ncsa.uiuc.edu/themes/ais/vo.html