Super Stream Collider–Linked Stream Mashups for Everyone[⋆]

Hoan Nguyen Mau Quoc, Martin Serrano, Danh Le-Phuoc, and Manfred Hauswirth

Digital Enterprise Research Institute, NUI Galway

Abstract. This paper describes the Super Stream Collider (SSC) platform and tools, which provide a web-based interface and tools for building sophisticated mashups combining seamnitcally annotated Linked Stream and Linked Data sources into easy to use resources for applications. The system includes drag&drop construction tools along with a visual SPARQL/CQELS editor and visualization tools for novice users while supporting full access and control for expert users at the same time. Tied in with this development platform is a cloud deployment architecture which enables the user to deploy the generated mashups into a cloud, thus supporting both the design and deployment of stream-based web applications in a very simple and intuitive way.

1 Introduction

The use of near-realtime stream data is a key enabler and driver in such diverse application domains as smart cities, home automation, ambient assisted living, or recommender systems. As on the Web, access to and integration of information from large numbers of heterogeneous sources under diverse ownership and control is a resource-intensive and cumbersome task without proper support. Semantic Web and Linked Data technologies can answer to many of the requirements but need to be (1) extended to transparently cater for dynamic stream information, (2) tailored to the existing infrastructures – from Twitter streams down to resource-constrained sensing hardware, and (3) made easily accessible and usable to promote uptake.

For (1) and (2), approaches have been proposed, e.g., C-SPARQL, EP-SPARQL, CQELS, SPITFIRE, GSN, etc. Still, wide-spread access to real streams does not exist at the same level as for Web resources. A public resource to remedy this problem is LSM, the Linked Stream Middleware (lsm.deri.ie), which provides access to more than 100,000 stream sources via a RESTful interface and a SPARQL/CQELS endpoint. However, to the best of our knowledge, no general-purpose infrastructure to support (3) exists to lower access thresholds for users and developers.

In this paper we describe the Super Stream Collider (SSC) platform and tools, which build on our pre-existing work for (1) and (2), i.e., CQELS, LSM, DERI Pipes, SPITFIRE, and GSN, and provides a web-based interface and tools for building sophisticated mashups combining semantically annotated Linked Stream and Linked Data sources into easy to use resources for applications. The system includes drag&drop construction tools along with a visual CQELS editor and visualization tools for novice users while supporting full access and control for expert users at the same time. Tied in with

[⋆] This work has been supported by SFI under Grant No. SFI/08/CE/I1380 (Lion-2) and by the EU under Grant No. ICT-2011-7-287305 (OpenIoT).

this development platform is a cloud deployment architecture which enables the user to deploy the generated mashups into a cloud, thus supporting both the design and deployment of stream-based web applications in a very simple and intuitive way.

The structure of the paper is as follows: Section 2 starts with a high-profile end-user application that we have built using SSC and which was used by approx. 4300 users during the Volvo Ocean Race finale in Galway June/July 2012 ("eat your own dog food"). Section 3 then describes the conceptual architectural design of the SSC stream mashup platform followed by a presentation of the main functionalities and features available to the user via the GUI in Section 4. Section 5 then outlines the implementation with particular emphasis on the principles towards enabling stream processing in cloud environments. In Section 6, we conclude with a summary of our findings. The appendix of the paper discusses how SSC addresses the requirements of the semantic web challenge.

2 Example SSC Application

The Volvo Ocean Race (VOR) finale in Galway, June/July 2012 attracted more than 820,000 visitors in a 9-day event (http://www.letsdoitgalway.com/). DERI developed the official VOR mobile app for Android/iPhone which included a mobile sensing part based using SSC as the stream processing platform. In the deployment we had 3000 iPhone and 1300 Android users during the finale. The goal was to test the platform and test it in a real, large-scale deployment with hard requirements and constraints. Using the application – besides "standard" information like timetables, results, location infos, etc., all using Linked Data sets – people can explore and experience in a smart city through real-time sensor data, i.e., traffic lights, traffic cams, weather stations, parking spaces, etc. together with social streams (in real-time, e.g., tracking other people's locations, tracking of crowd behavior, etc.). The GUI provides all kinds of query possibilities and map visualizations (see Figure 1 below) for these stream data, so that it becomes readily usable and useful for the end-user. The users also can decide if they want to use other information as real-time monitoring service, e.g., parking spaces, weather stations, traffic cams, etc. This mobile sensing application is a typical SSC application combining user-defined semantically annotated Linked Stream sources and their combinations with static Linked Data deployed in our execution engine and data center backends.

3 Architectural Design

The SSC platform is designed as a classical dataflow/workflow execution environment connecting processing input/outputs through pipelines for creating data mashups. Conceptually, each operator has n input streams and one output stream as illustrated in Figure 2. The inputs can be in any format while the output is RDF. Only the final operator of a workflow can return a format other than RDF, if necessary. Operators can be of three classes: A *data acquisition operator* is used to collect or receive data from data sources or gateways and can be pull-based or push-based. In these operators the data transformation and alignment can be done to produce a normalized RDF output format. A *stream processing operator* defines stream processing functionalities in a declarative language, e.g., CQELS. A *streaming operator* streams the outputs of the final operator of a workflow to the consuming applications. An example of a consuming application is the real-time visualisation widget described in Section 5.

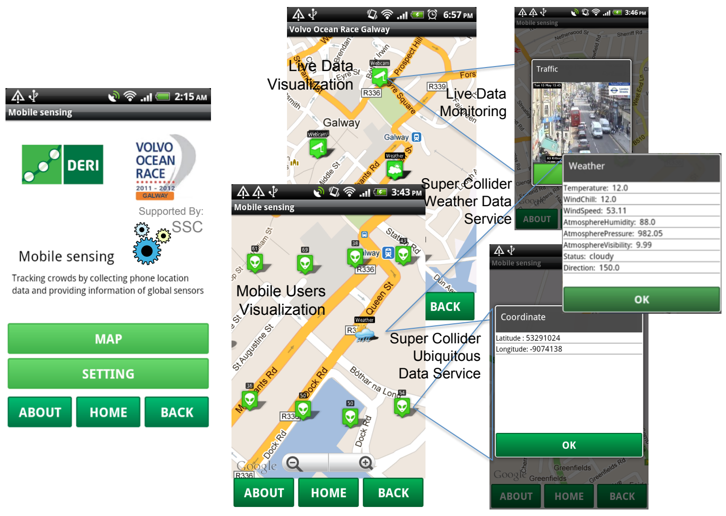


Fig. 1. Volvo Ocean Race Application

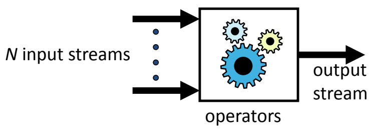


Fig. 2. Architecture of the SSC platform.

The operators of the same class are executed on an execution container. An example of execution container is a continuous query precessing engine that is used for stream processing operators. The execution containers are running on networked machines that can be dynamically allocated based on the processing load registered to SSC. Figure 3 shows an informal high-level view of this architecture.

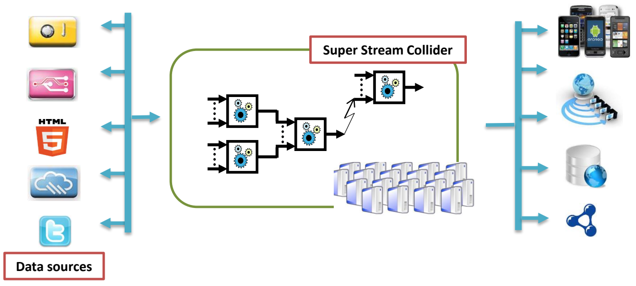


Fig. 3. Layered architecture of the SSC platform.

For instance, if the number of registered sources grows, SSC will request additional data fetching engines from its underlying cloud computing infrastructure and allocate them to this task. Thus SSC can flexibly answer to dynamic load-profiles which are common in stream-based applications. .In a concrete workflow, two connected operators can be executed in different execution containers. For instance, the data acquisition operator for collecting Tweets can stream data via the network to the stream processing engine. The external computing services such as SPARQL endpoints or web services are called external execution containers.

To support the easy and intuitive definition of data processing workflows in a "box-and-arrows" fashion, the SSC platform offers a visual programming environment. The interactive process of creating a mashup with SSC features context-aware discovery services for data sources. This process enables the user to incrementally build a workflow in a step-by-step fashion by dragging&dropping the required building blocks and connecting and parametrizing them. Also, this supports visually debugging the workflow of the mashup. When the user finishes a mashup, it can be deployed to the SSC cloud to be re-used as a data source or an operator.

4 System Demonstrations

A deployment of SSC is online at http://superstreamcollider.org, which provides a user-friendly interface called SSC visual editor. This is a light-weight Web-based workflow editor for composing mashup data through drag&drop. Using the SSC visual editor, we aim at providing a programmable Web environment suitable not only for expert users but also for non-expert programmers. Figure 4 provides a overview screenshot of SSC with the numbers feature explained in the text below.

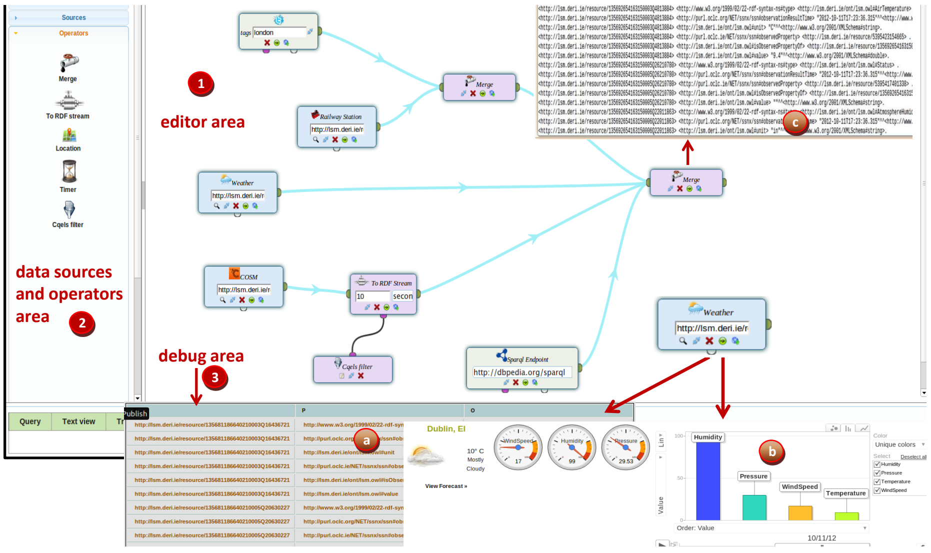
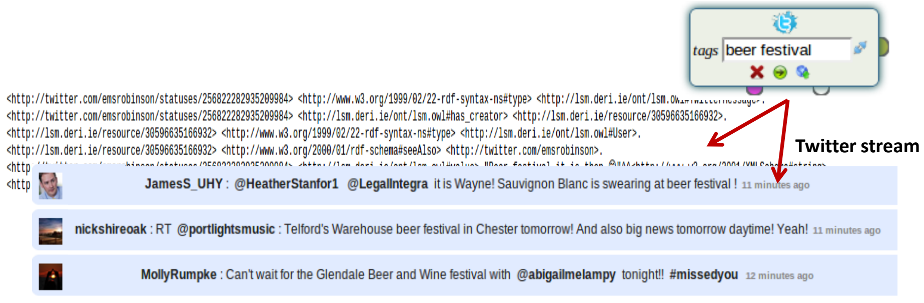


Fig. 4. SSC User interface.

The main components of the platform are the visual editor (1), data sources and operators (2) and the debugging component (3). Each data source and operator is visualized as a block in the editor area Visually, a mashup workflow is a combination of connected operators and data sources. It is incrementally designed by dragging the icons representing for corresponding operators in (2) and then dropping to the editor area (1).

The flows of data from the sources to the final output are defined by wiring the blocks with configured parameters. The live visualisations of operator outputs are shown in (3). The output of the workflow is a live mashup data stream which can be published, visualized and queried. Currently SSC supports several types of live data sources, such as LSM sensors (over 100,000 sensor around the world including weather, train schedule, traffic status, etc.), twitter streams, DBPedia and Sindice data sources, among others, which can be discovered by the SSC discovery component. This context-aware discovery service uses relevant text, location, sensor data sources that the user has typed and chosen as inputs to form the queries to such systems to find useful data items to recommend to the user. 3(a) in Figure 4 shows a temperature sensor as an example.

SSC's debugging component supports the user by showing the results of each of his/her actions. (b) in Figure 4 shows an example. The result data can be shown as raw data, RDF data or can be visualized in different types of charts, so that users can easily monitor their data processing workflows. In Figure 4, the output is a merge of multiple input streams. Another typical example of stream data is Twitter data as shown in Figure 5. In this example, the SSC collects all tweets mention about the user-specified topic and provides them as an RDF stream. For this the user only needs to drag an operator into the editor and enter the topic of interest.



```
<http://twitter.com/emsrobinson/statuses/256822282935209984> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://lsm.deri.ie/ont/lsm.owl#TwitterMessage>.
<http://twitter.com/emsrobinson/statuses/256822282935209984> <http://lsm.deri.ie/ont/lsm.owl#has_creator> <http://lsm.deri.ie/resource/30596635166932>.
<http://lsm.deri.ie/resource/30596635166932> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://lsm.deri.ie/ont/lsm.owl#User>.
<http://lsm.deri.ie/resource/30596635166932> <http://www.w3.org/2000/01/rdf-schema#seeAlso> <http://twitter.com/emsrobinson>.
<http
<http
```

JamesS_UHY : **@HeatherStanfor1** **@LegalIntegra** it is Wayne! Sauvignon Blanc is swearing at beer festival ! 11 minutes ago

nickshireoak : RT **@portlightsmusic** : Telford's Warehouse beer festival in Chester tomorrow! And also big news tomorrow daytime! Yeah! 11 minutes ago

MollyRumpke : Can't wait for the Glendale Beer and Wine festival with **@abigailmelampy** tonight!! **#missedyou** 12 minutes ago

Fig. 5. Twitter data to RDF stream.

An important aspect of the SSC platform is that the data produced by a stream mashup can again be published through a web socket URL and thus be re-used as an input by other web applications or mashups. When a mashup is published, it will be assigned to a unique websocket URL, e.g., ws://superstreamcollider.org/websocket/8a8291b73215232 as shown in Figure 4.

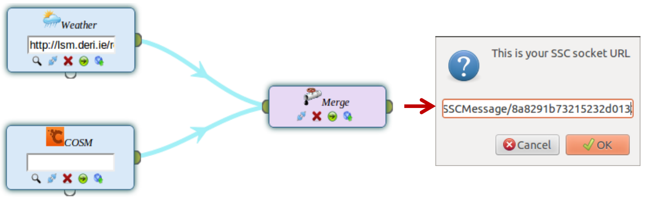
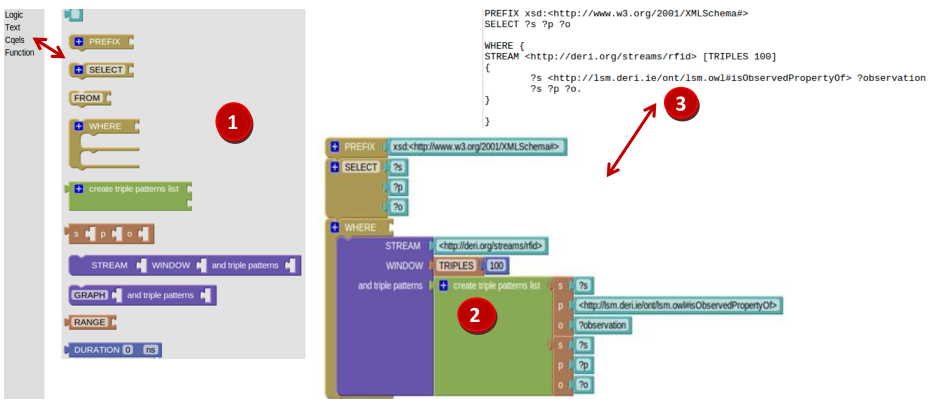


Fig. 6. Publishing a live mashup with an URI

In addition, a mashup can be serialized as an "UserView" and be stored in a custom JSON format in order to be later loaded into the editor or run by the server-side execution engine. For more advanced users, SSC also provides a CQELS/SPARQL query editor. The user can either enter the query directly into the query text area or use SSC's visual CQELS editor (Figure 4) which supports end-users in creating, modifying and reusing CQELS/SPARQL queries in a drag&drop fashion.



```
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
SELECT ?s ?p ?o

WHERE {
STREAM <http://deri.org/streams/rfid> [TRIPLES 100]
{
        ?s <http://lsm.deri.ie/ont/lsm.owl#isObservedPropertyOf> ?observation
        ?s ?p ?o.
}

}
```

Logic
Text
Cqels
Function
PREFIX
SELECT
FROM
WHERE
create triple patterns list
s p o
STREAM WINDOW and triple patterns
GRAPH and triple patterns
RANGE
DURATION 0 ns
1
2
3
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
SELECT ?s ?p ?o
WHERE
STREAM <http://deri.org/streams/rfid>
WINDOW TRIPLES 100
and triple patterns
create triple patterns list
s ?s
p <http://lsm.deri.ie/ont/lsm.owl#isObservedPropertyOf>
o ?observation
s ?s
p ?p
o ?o

Fig. 7. CQELS visual editor.

5 Operators and interesting features

This section overviews interesting SSC functionalities. Due to space constraints we cannot go into great detail, but extensive documentation is available at http://superstreamcollider.org/. SSC provides a wide range of data acquisition operators which enableaccess to huge amount of data sources. The sensor wrappers of LSM allow SSC users to collect data directly from physical sensors or via gateways. The RDF-izing operators extended from Any23[1] help to convert dynamic web data sources to RDF-based streams. We also implemented wrappers for transforming social stream data to RDF streams, as already mentioned for Twitter. For output streams of SSC mashups we support streaming protocols such as PubSubHubbub[2], XMPP[3] and WebSockets.[4]

SSC also provides developers with various data manipulation operators. For RDF-based data mashups and data consolidation, we extended and support the operators of DERI Pipes [2]. To filter data streams, we use our CQELS engine [1] for constructing window-based filters with the full expressive power of SPARQL 1.1 (CQELS is an extension of SPARQL 1.1).

To reduce the effort of learning SPARQL and CQELS, SSC also offers visual SPARQL and CQELS editor which enable the user to build SPARQL/CQELS queries interactively and a step-by-step way. Furthermore, this interactive workflow editing process is leveraged by the context-based discovery services which recommend potentially useful data sources and data items in every step of building a mashup in SSC. These services are powered by Sindice APIs, LSM's sensor database, and other online SPARQL endpoints such as Dbpedia, LinkedGeoData, etc. The user can add more knowledge by pointing SSC to further SPARQL endpoints.

[1] http://incubator.apache.org/projects/any23.html
[2] http://code.google.com/p/pubsubhubbub/
[3] http://xmpp.org/
[4] http://dev.w3.org/html5/websockets/

The mashups deployed in SSC will be run in our elastic cloud computing infrastructure. This infrastructure dynamically allocates execution containers to execute the constituent operators of the deployed mashups. For operators that query RDF streams and RDF datasets, a container with a CQELS engine is created on demand to host such operators. To enable low-latency continuous queries over big RDF datasets, each CQELS engine has its own cache manager to cache and index relevant subsets of such datasets. The (heavy) pull-based data acquisition operations are scheduled by our LSM's Hadoop cluster.

To support HTML and Android developers, we also built light-weight widgets for consuming data from our live mashup. As demonstrated on the SSC website, they can embed these widgets to their HTML pages or Android applications with few lines of code and without having to learn RDF or SPARQL/CQELS, etc. With only a few parameter customisations, web pages or applications can receive live updates from SCC via Websocket or Google Cloud Messaging for Android. [5]

6 Conclusions

In this paper we have introduced our Super Stream Collider platform (SSC) which enables the user to build complex mashups using semantically annotated Linked Streams and Linked Data. SSC offers a huge number of stream sources, including social streams such as Twitter, which can easily be extended by the user and provides sophisticated visual tools both for novice and for expert users that speed up the learning curve. SSC comes with a sophisticated cloud-based deployment environment which supports the execution of mashups and the use of mashup output streams by other web applications and mashups. We have extensively evaluated SSC in a large-scale real-world deployment with approx. 4300 users as part of the Volvo Ocean Race finals in Galway.

References

1. D. Le-Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *Proceedings of 10th International Semantic Web Conference*, pages 370–388, 2011.
2. D. Le-Phuoc, A. Polleres, M. Hauswirth, G. Tummarello, and C. Morbidoni. Rapid prototyping of semantic mash-ups through semantic web pipes. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 581–590, New York, NY, USA, 2009. ACM.

Appendix: Meeting the Semantic Web Challenge Requirements

Minimal requirements

$\surd$ **The application has to be an end-user application.** The SSC platform has a user-friendly web interface for different user levels as describe in Section 4. A comprehensive description is given in the SSC user manual at http://www.superstreamcollider.org.
$\surd$ **The information sources used should be under diverse ownership or control.** By design, SSC allows the user to collect data from different providers under diverse ownership and control policies. SSC also allows the user to annotate and integrate existing data sources to create new ones, which can have specific access/publishing policies defined by the user.

[5] http://developer.android.com/guide/google/gcm/index.html

$\surd$ **The information sources used should be heterogeneous.** As described in Section 3, SSC supports a broad variety of heterogeneous data sources with various data formats and access protocols.
$\surd$ **The information sources used should contain substantial quantities of real world data.** SSC's ultimate goal is to provide any real-world data captured by sensors or available as streams as linkable Web information sources. With the SPARQL/CQELS editors and processing engine, SSC enables developer to access to the full Linked Data Cloud with the support of the discovery service of Sindice. SSC has access to a large database of sensors provided by LSM with over 100,000 live sensor data sources all over the world with approximately 20 million updates per day. It also provides a wrapper to access to approximately 10000 sensors data sources from COSM.
$\surd$ **The meaning of the data has to play a central role.** Facilitate by data transformation and annotations of SSC, the sensor and web data sources are semantically annotated.

<u>**Additional Desirable Features**</u>

$\surd$ **The application provides an attractive and functional Web interface (for human users).** SSC provides an easy-to-use GUI which has several visualisations of stream data such as maps, charts, animated data updates, etc. The interactive faceted search functionality makes the data exploration more intuitive and efficient and server-push technology brings the experience of real-time web rendering to the users.
$\surd$ **The application should be scalable.** The scalability of the system depends on underlying triple storage technologies, e.g. Virtuoso and the Linked Stream Data processing engine (CQELS) [1]. Currently, the static dataset for approximately 100,000 sensor data sources contains approximately 20 million triples. We are constantly adding more data sources and enriching more metadata on the assumption that Virtuoso can handle billions of triples. For processing continuous queries, CQELS engine can handle up to 50,000 updates per seconds with thousands of query instances.
$\surd$ **Novelty, in applying semantic technology to a domain or task that have not been considered before.** Applying semantic technology to sensor data is a new trend in data integration, and this work has been pioneering towards this trend. It is one of the first systems that allows the integration of live sensor data with data in the Linked Data Cloud.
$\surd$ **The application has clear commercial potential and/or large existing user base.** This platform provides an easy and unified way to integrate useful data captured from sensors. Its Linked Stream Data can be used to rapidly build interesting applications in different domains like "smart cities", e-health and tourism. We apply the infrastructure in research projects and with industry partners of DERI.
$\surd$ **Multimedia documents are used in some way.** Some sensor data sources supported in SSC/LSM produce image, audio or videos video data, e.g., traffic cameras, satellite/radar images and noise sensors.
$\surd$ **There is an use of dynamic data, perhaps in combination with static information.** Dynamic Linked Stream Data combined with static data from Linked Data Cloud is the main feature and motivation of the system.
$\surd$ **There is support for accessibility on a range of devices.** SSC publishes data in several formats to simplify the data consumption on different platforms. For example, we support the Android and iPhone platforms in applications such as the described Volvo Ocean Race apps.