PIOTRe: Personal Internet of Things Repository

Eugene Siow, Thanassis Tiropanis, and Wendy Hall

Electronics & Computer Science, University of Southampton
{eugene.siow,t.tiropanis,wh}@soton.ac.uk

Abstract. Resource-constrained Internet of Things (IoT) devices like Raspberry Pis', with specific performance optimisation, can serve as interoperable personal Linked Data repositories for IoT applications. In this demo paper we describe PIOTRe, a personal datastore that utilises our *sparql2sql* query translation technology on Pis' to process, store and publish IoT time-series historical data and streams. We demonstrate with PIOTRe in a smart home scenario: a real-time dashboard that utilises RDF stream processing, a set of descriptive analytics visualisations on historical data, a framework for registering stream queries within a local network and a means of sharing metadata globally with HyperCat and Web Observatories.

Keywords: SPARQL, SQL, RSP, Query Translation, Internet of Things, Analytics, Web Observatory

1 Introduction

Internet of Things (IoT) time-series data that is flat and wide can be efficiently stored in relational databases and queried with SPARQL using mappings and query translation engines as shown in our previous work [1]. *sparql2sql* which translates SPARQL to SQL and *sparql2stream* which translates SPARQL to Event Processing Language (EPL) are two such engines for historical data and streams respectively. Both engines show performance improvements in query latency for IoT scenarios on Raspberry Pis' that range from 2 times to 3 orders of magnitude.

In this demonstration paper we will present PIOTRe[1], a personal repository that utilises *sparql2sql* and *sparql2stream* to provide efficient Linked Data access through SPARQL endpoints for IoT historical data and streams on Pis'. We will then demonstrate how PIOTRe supports applications like:

1. A real-time smart home dashboard that utilises *sparql2stream*'s RDF stream processing to update widgets with push results via web sockets.
2. A smart home visualisation application that uses a set of SPARQL queries with space-time aggregations, translated by *sparql2sql*, on historical data.
3. A lightweight client-broker-server architecture that supports registering stream queries and delivering results on an offline local network.
4. A means of publishing and sharing metadata and mappings online as HyperCat or with decentralised catalogues known as Web Observatories.

[1] https://github.com/eugenesiow/piotre

2 PIOTRe Design and Architecture

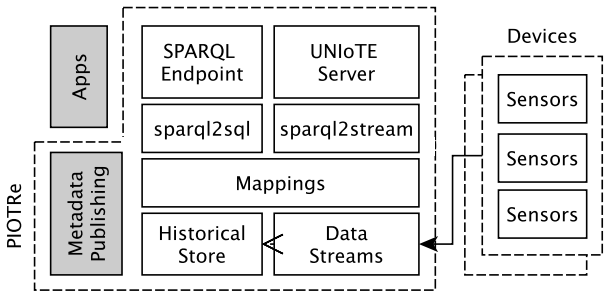

Fig. 1. Architecture of PIOTRe

PIOTRe is designed to run on a compact and mobile lightweight computer like a Raspberry Pi (it has also been tested on an x86 Gizmo2[2]). PIOTRe consists of a number of components as shown in Fig. 1. Sensors produce time-series data that forms a set of input data streams to the system.

Data streams, enter an event stream for processing and are stored in a historical relational store. Each stream forms a table in the store. PIOTRe uses H2[3] as a default relational store and Esper[4] as an event stream processing engine. H2 can be replaced by a relational database or column store that supports SQL.

sparql2sql and *sparql2stream* work on top of the historical store and event stream respectively. Mappings are shared across the engines and are a representation of how RDF maps to columns in relational tables or a field of an event in a stream. A SPARQL endpoint uses the *sparql2sql* engine to translate incoming SPARQL queries to SQL, execute the query on the relational store and return the result set in the appropriate format. The UNified IoT Environment (UNIoTE) Server is part of a client-broker-server architecture to allow streaming SPARQL queries to be registered with the *sparql2stream* engine. The engine in turn translates queries to EPL, registers them with the underlying event stream and sends results to requesting clients. UNIoTE is described in Section 4.

Apps in PIOTRe have the flexibility of being written in any language and framework and communicate with the SPARQL Endpoint and UNIoTE Server through HTTP, although underlying the UNIoTE server are ZeroMQ sockets[5].

The metadata publishing component publishes the metadata from mappings, like the sensor descriptions, locations, data fields and formats to support global discovery and interoperability. It is described in more detail in Section 5.

[2] http://www.gizmosphere.org/products/gizmo-2/
[3] http://www.h2database.com
[4] http://www.espertech.com/
[5] http://zeromq.org/

3 Example Applications using PIOTRe

IoT Freeboard[6] is a real-time dashboard that consists of a simulator that replays a stream of smart home data at a variable rate to PIOTRe and a web application dashboard that receives the output from the set of registered push-based streaming SPARQL queries as a web socket events stream. Each result in the stream consists of the query name it was generated by and fields of the result set e.g. `{"queryName":"averagePower","averagePower":"10.5","uom":"W"}`. A host of widgets can be added to the dashboard to visualise these events. A demo video is available[7].

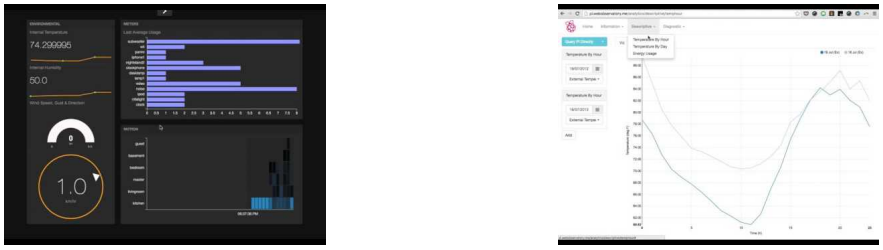

Fig. 2. Apps using streaming and historical data with PIOTRe

Pi-SmartHome[8] is an application that uses SPARQL queries on historical smart home data to provide visualisations across time and space. The queries are as follows: 1) hourly aggregation of temperature, 2) daily aggregation of temperature, 3) hourly and room-based aggregation of energy usage and 4) diagnosis of unattended devices through energy usage and motion, aggregating by hour and room. The application allows the user to tweak the days and months as parameters to the SPARQL queries and generate and compare graph visualisations. A demo video[9] and online demo are available.

4 Lightweight Local, Offline Client-Broker-Server

In some IoT scenarios like disaster management or environmental monitoring in remote locations, local, offline networks of devices are necessary. The UNified IoT Environment's[10] (UNIoTE) client-broker-server architecture enables stream queries to be registered and results to be delivered to devices. A publish-subscribe mechanism based on lightweight ZeroMQ sockets are used. Servers subscribe to URIs and clients publish queries (with URIs in the FROM clause) facilitated by brokers with known addresses. As shown in Fig. 1 a UNIoTE server uses a *sparql2stream* engine to translate and register queries. Results are then delivered directly to all requesting clients through push-pull sockets.

[6] https://github.com/eugenesiow/iotwo
[7] https://youtu.be/oH0iSWTmKUg
[8] https://github.com/eugenesiow/ldanalytics-PiSmartHome
[9] https://youtu.be/g8FLr974v9o
[10] https://github.com/eugenesiow/uniote-broker

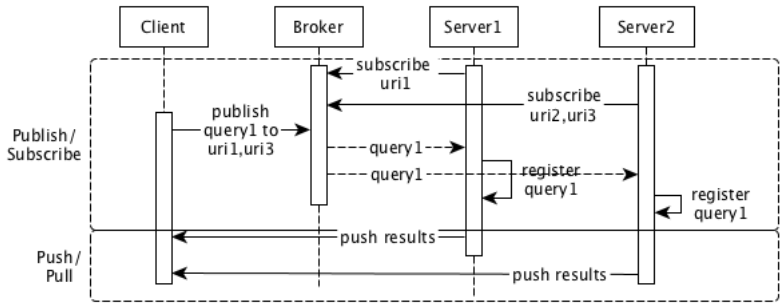

Fig. 3. Sequence Diagram of Issuing a Query with UNIOTE

Fig. 3 describes, using a sequence diagram, how the two publish-subscribe and push-pull mechanisms work in UNIOTE. A client publishes a streaming SPARQL query with *uri1* and *uri3* in the FROM clause and its address to each of the URI topics. Servers 1 and 2 have each subscribed to URI topics based on their streams and receive the query, register the query using *sparql2stream* and push results directly to the client's addresses.

5 Observatories and Online Metadata Sharing

The Web Observatory Project is developing a global decentralised, distributed infrastructure that allows the use and exchange of datasets, analytical apps and visualisations [2]. Web Observatory instances exist as collections of datasets and analytical tools protected by access controls. The principles espoused by the Web Observatory as a distributed infrastructure are a possible solution for managing datasets and apps in the IoT. By sharing metadata of IoT datasets and providing the Observatory access through the SPARQL endpoint, PIOTRe systems, when online, are able to support apps which use globally distributed data sources across Observatory instances securely.

HyperCat[11] is a complementary catalogue for exposing collections of uniform resource identifiers (URLs) that refer to IoT assets over the web. PIOTRe systems connected to various sensors, devices and things, publish metadata of these from mappings with a HyperCat server when online, increasing interoperability.

References

1. Siow, E., Tiropanis, T., Hall, W.: SPARQL-to-SQL on Internet of Things Databases and Streams. In: Proceedings of 15th International Semantic Web Conference (ISWC2016) (2016)
2. Tiropanis, T., Hall, W., Hendler, J., de Larrinaga, C.: The Web Observatory: A Middle Layer for Broad Data. Big Data 2(3), 129–133 (2014)

[11] http://www.hypercat.io/