Requirement analysis of time series in an Open Data ecosystem in Flanders

Dwight Van Lancker

Supervisor(s): Pieter Colpaert, Raf Buyle, Julián Andrés Rojas Meléndez, Brecht Van de Vyvere

Abstract— **Building a Smart City involves building a network of IoT-sensors. These sensors generate data periodically, leading to time series data. Smart Cities are building data brokers to make time series data available to their citizens. However, it is still unclear how the data should be published and archived in a sustainable and cost-effective way. Therefore, we interviewed 3 different stakeholders in the City of Things program in Flanders and came up with a prototype applied to air quality sensor data. We transformed the air quality data to Linked Data, raising the semantic interoperability with the Semantic Sensor Network (SSN) initiatives. Next, we extended the Linked Time Series API with geospatial tiles, in order to make the amount of sensors added to the Web API scalable. Finally, in order to estimate the costs for both data publisher and consumer, we benchmarked the Linked Time Series server and compared it with a database as a back-end. The contributions of this study, build upon the principles of Linked Open Data are valuable for governments that aim to publish air quality sensor data on a Web-scale at a cost-effective way. We found that in most cases the Linked Time Series consumes less CPU than the database, except in the case of a small number of clients. By increasing the number of clients, the advantage of the caching fragments can be used. Furthermore, it is a Linked Time Series client that consumes more bandwidth and more CPU than a database client. Finally, the Linked Data Fragments require more disk space than the data in the database tables. Based on these findings and taking into account that CPU is the cost determining factor, we can conclude that the Linked Time Series server is a good approach to publish and archive sensor data in a sustainable and cost-effective way.**

Keywords—**Smart Cities, Semantic Web, Linked Open Data, Air quality, Archiving, Time Series, Linked Data Fragments**

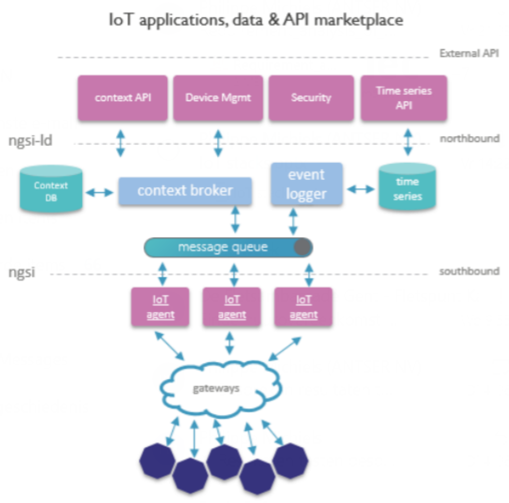

Fig. 1
FIWARE ARCHITECTURE THAT IS USED BY THE CITY OF THINGS PROGRAM IN FLANDERS TO DEVELOP ANTWERP AS A SMART CITY.

I. Introduction

TODAY , the greater part of the global population lives in urbanized areas and it is expected that this will increase to nearly 70% by 2050 [1]. Bear in mind that in 1950, 70% of the world population was living outside the city. The phenomenon of urbanization causes new problems, including air pollution. To avoid that this accelerated urbanization leads to a crisis, cities must become 'Smart' . To be able to make smarter decisions and raise awareness regarding to air pollution, cities need to combine various data sources including air pollution, weather and traffic data [2]. Also, real-time air quality data can support citizens to decide which route they take in order to avoid air pollution [3]. Various efforts have already been made to inform citizens about air quality. For example, the BelAir app from the Flanders Environment Agency (VMM), uses of the location of the smartphone to provide information about air quality to the user [4]. Another inspiring idea from the VMM that came up during the interview, is an application that measures the exposure of recreational joggers to air pollutants. The VMM would like to suggest a healthier route with a lower exposure for citizens. To realise this use case, multiple sensor data sources need to be queried. The sensor data is dependent on both location and time. However, it is unclear how this data should be published and archived in a sustainable and cost-effective way. Thus, we introduce the Linked Time Series server, using Linked Data Fragments to publish and archive time series of air quality sensor data.

The remainder of this article is structured as follows. We first describe the state of the art. We then describe the used data model to transform air quality sensor data to Linked Data followed by the implementation of the Linked Time Series server. Subsequently, we describe the evaluation setup. Then we display the obtained results followed by conclusions drawn from this work.

II. State of the art

A. *FIWARE*

FIWARE is an open source project with the purpose of creating an open, sustainable ecosystem around public, royalty-free and implementation-drive software platform standards [5]. It is the result of the European Future Internet Public-Private Partnership (FI-PPP) and created to simplify the development of new Smart Applications. With FIWARE comes an extensive library with components, called *Generic Enablers* (GEs). The most important and mandatory component in all applications that use FIWARE, is the *Orion Context Broker*. This component has a key function in every smart solution, particularly managing context information, enabling updates and bring access to the context [6].

The City of Things program (CoT) in Flanders that aims to develop Antwerp as a Smart City, uses an implementation of FI-

WARE. The used architecture is shown in figure 1. CoT used two components from the FIWARE library and are located between northbound and southbound on the figure. The first component is obviously the Orion Context Broker, using a MongoDB as a context database. The second component is Quantumleap with Crate-DB as time series database. Quantumleap supports the storage of time series data and provides the ability to query it through a time series API. Both components are reference implementations of the FIWARE standards.

Linked Data Fragments should not be considered as a counterpart, but rather as a useful addition. When working with real time data, FIWARE could be a good approach. However, in the case of historic data, complex queries can trigger a high CPU cost for the database. Therefore, Linked Data Fragments in combination with a better caching strategy (e.g. HTTP-cache) could lower these costs.

B. *Linked Time Series server*

The Linked Time Series (LTS) server is an ongoing implementation that aims on providing a cost-effective interface for data publishing. Through the extensible architecture, shown in figure 2, data publishers can define multidimensional interfaces on top of their data to provide query answering functionalities [7]. Multidimensional interfaces were introduced for generically fragmenting data with a specific order and publishing these fragments in an interface-level index. It allows clients to navigate to specific parts of a dataset for a certain *n*-dimensional range [7, 8]. To describe multidimensional interfaces, new terminology[1] was introduced. It describes the concepts of *Range Fragment* and *Range Gate*. A *Range Fragment* is a Linked Data Fragment that has an interval as selector. A *Range Gate* is a Linked Data interface that exposes a set of Range Fragments. This interface selects all Range Fragments whose interval overlap with the interval of the Range Gate [8]. The Linked Time Series server is very much in line with this research, which is why it will be used as base architecture during this study.

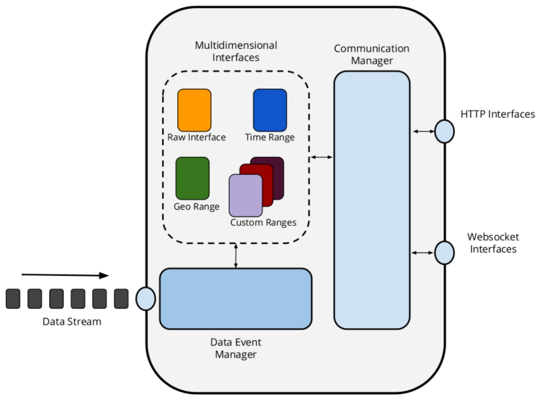

Fig. 2
ARCHITECTURE OF THE LINKED TIME SERIES SERVER.

III. DATA MAPPING

The Linked Time Series server expects Linked Data as input. In order to be compatible with the Linked Time Series server, the air quality sensor data was transformed to Linked Data. We developed a data model based on the *Semantic Sensor Network* (SSN) ontology, consisting of three parts. First, the model describes a part of the air quality (e.g. PM_{10}).

```
<PM10> a sosa:FeatureOfInterest .
```

As shown in the example above, an air quality part is identified by the property *sosa:FeatureOfInterest*. Next, the model describes a sensor that measures air quality parts

```
<Sensor/343233>
  a sosa:Sensor ;
  sosa:observes <Sensor/343233/PM10> .
```

As we can see in the example above, a sensor has two properties. The first property, *sosa:Sensor*, identifies the sensor. The second property states which part of the air quality is measured by the sensor. As it is possible for one sensor to measure multiple parts of the air quality, *sosa:observes* can have several values. Last, the data model describes an observation.

```
<Observation/343233/155639/PM10>
  a sosa:Observation ;
  sosa:hasFeatureOfInterest <PM10>;
  sosa:hasSimpleResult ”40”;
  sosa:madeBySensor <Sensor/343233>;
  sosa:observedProperty <Sensor/343233/PM10>;
  sosa:resultTime ”2019−04−27T22:02:15.698Z”;
  schema:geo [
      a schema:GeoCoordinates ;
      schema:latitude ”” ;
      schema:longitude ””
    ];
```

To identify an observation, the property *sosa:Observation* was used. Further, *sosa:hasFeatureOfInterest* specificies which part of air quality was measured, where *sosa:hasSimpleResult* contains the actual value of the measurement. The properties *sosa:madeBySensor* and *sosa:observedProperty* are related to the previous example. The first property specifies which sensor made the observation and the second property is linked to the *sosa:observes*-property of a sensor. Last, *schema:geo* was used to store the latitude and longitude of the observation.

IV. LINKED TIME SERIES SERVER IMPLEMENTATION

An effort to define a Time Range multidimensional interface was already made. Time constrained intervals can be used to create Range Fragments exposing raw data or to create summaries that compute statistical averages [7]. In that way it is possible to expose regular sensor data or average values at hour, day, month and year level. We re-implemented the above interface to the needs of the air quality sensor data. With every data update that comes in, the data is passed on to the interfaces that apply their logic to it. In this way, every interface creates its own fragments. Clients are now able to request regular air quality data or average values regarding the air quality parts within a time range. Furthermore, based on the interviews with the stakeholders, we extended the LTS API with geospatial tiles. We divided the city of Antwerp in 4 static tiles, each tile storing time series data in fragments. So, along the possibility the request time range data, the client also has the possibility to request data within a geospatial tile. Onward the regular sensor data in the tile, the client also receives an average value of the requested data, that applies to the tile. The metadata of a geospatial tile also contains links to the other tiles, so a client can discover all geospatial tiles. Furthermore, to reduce the workload on the Linked Time Series server, an NGINX reverse proxy cache was introduced.

V. Evaluation design

In order to estimate the costs for data publishing, archiving and consuming, we benchmarked the LTS server and a database. We used a database because the FIWARE architecture also makes use of a time series database. To do this benchmark, a PostGIS database was set up, along with two client consumers. PostGIS is a spatial database and an extension of a PostgreSQL database. One client consumes data from the LTS server, while the other client consumes data from the database. The following questions were asked by both clients:

1. Get most recent observation
2. Requesting data in a time range that has **not** yet ended
3. Requesting data in a time range that has ended
4. Requesting average values in a time range that has **not** yet ended
5. Requesting average values in a time range that has ended
6. Retrieve a geospatial tile

A total of 6 tests were conducted. As shown above, two tests were executed for time range and average values. When clients request a historic time range, it is possible to rely on the cache once the fragment has been cached, expecting a reduction in workload for the Linked Time Series server. For each test, the effort for the client and its back-end service was measured. Populating the database, creating fragments and transforming the air quality sensor data to Linked Data also requires a cost. Therefore, the effort to fulfill these tasks was also measured.

VI. Results

In section IV we mentioned that fragments were created or extended upon receiving data updates. This also results in a cost that must be included for the LTS server. To determine the cost, the CPU consumption was measured over a time period. Afterwards, an average value of the CPU consumption was calculated and was 2.18%. The CPU consumption for creating fragments is shown in figure 3. This average cost is already added to the graph of the LTS server in this figure.

A. CPU usage results

From the tests that were conducted, the general trend was that in most cases the LTS server consumed less CPU than the database. This was not the case when one client asked for sensor values in a time range. In that case, the LTS server consumed more CPU than the database. The reason for this is that the time range consisted of multiple fragments, so the client had to request all the fragments, resulting in a higher CPU usage. For the other cases, the result mostly consisted of one or two fragments. This means, that for one client the CPU usage of the LTS server most likely will be higher than the database if the answer consists of multiple fragments and no caching is involved. Whenever it is possible to use a cache, the database is outperformed, in all test cases. Figure 3 shows the CPU usage when requesting sensor values and a mean value in a geographical area. For all numbers of clients the LTS server consumes less CPU than the database. The figure shows that a cache has been used, but the LTS server still consumed an amount of CPU? When requesting values in a historic time range or when the results consists of multiple fragments, the last (most recent) fragment can not be cached. The last fragment can still be extended with new sensor data while this is not the case for previous fragments and can therefore be cached. So in the case of a non-historical time range, a client can rely on the LTS cache for all fragments, except the last one. To retrieve the last fragment, the LTS client always has to send its request to the LTS server.

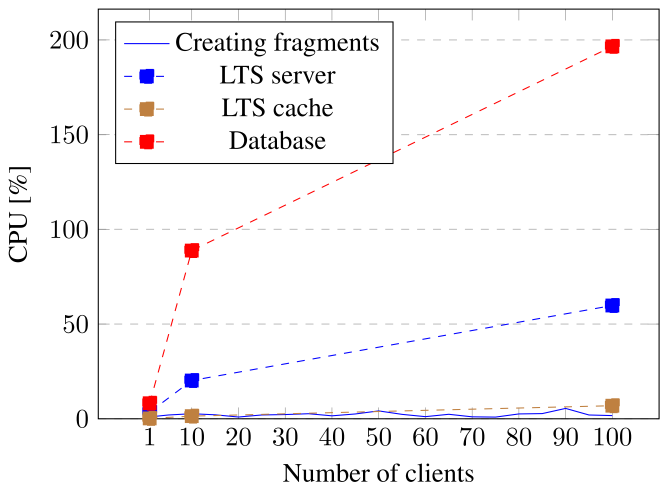

Fig. 3
CPU percentage as a function of the number of clients for requesting sensor values and an average value in a geographical area.

In case of a historic time range, all fragments can be cached. Then the LTS server only consumes CPU for applying the logic of its interfaces (time indexing, calculating mean and geospatial indexing) when receiving data. This outcome is shown in figure 4. It is important to note that this figure shows the course after all fragments have been cached. At first, the cache is empty and requests are sent to the LTS server. This results in a higher CPU percentage of 18.1% for 10 clients and 97.4% for 100 clients. As said above, from the moment that all fragments have been cached, the CPU usage is reduced to the effort of applying the logic of the interfaces to the received data.

B. Bandwidth results

Looking at the results displayed in figure 5, *input* refers to the amount of received kilobytes from the back-end systems. We see that using Linked Data Fragments consumes the most bandwidth. This trend applies to all the tests that were conducted. There are two reasons to explain this result. First, a Linked Data Fragments server does not provide a specific answer to the client's question. Instead, the server provides the fragments needed to answer the question and the client has to obtain the answer himself (e.g. by parsing the fragment). For the database it is more straightforward, because when a client asks a question, the database itself creates the answer and returns it to the client. Second, fragments consist of more than only the sensor values. Each fragment contains information of all sensors that have measured an air quality part in that fragment. Also, each fragment contains metadata, describing the fragment. A time range or geospatial tile can consist of multiple fragments. To be able to have the same answer as a client of the database, a client of the LTS server must request all fragments, resulting in multiple HTTP requests.

C. *Client results*

As said above, clients of the LTS server have much more work to do to, to come to the same answer as clients of the database. This results in a much higher CPU usage, which is also visible in figure 6. This figure displays the CPU usage of both clients when requesting *average values in a time range that has not yet ended*. Concluding from the test results, this figure is also the outcome for the other tests that were conducted.

D. *Storage results*

For both the LTS server and database the amount of storage they took up, was measured. The results are displayed in table I and show that the database consumes almost 10 times less space than the LTS server. This is mainly the result of time indexing the data. The incoming data is stored on hour level, but also on day, month and year level. This results in duplicate data, more fragments and above all, more occupied storage space.

VII. Conclusion

By transforming the air quality sensor data we raised the semantic interoperability with the SSN initiatives. We learned that applying the method of Linked Data Fragments for air quality sensor data lowers the CPU cost for publishing and raises the availability due to a better Web caching strategy. For other costs such as storage and bandwidth, Linked Data Fragments perform worse than a database. Considering that CPU is the deciding factor, because it is the biggest cost, we can conclude that Linked Data Fragments is a suitable method to publish and archive air quality sensor data in a sustainable and cost-effective way.

Looking at both clients, it is the client of the LTS that consumes the most CPU. A database client sends its query to the database and receives an answer to its question. This is different for the client of the LTS server. First, it is possible that the client has to request multiple fragments to collect all data needed to come to an answer, resulting in more bandwidth and higher CPU usage. Secondly, all fragments must be parsed to make the

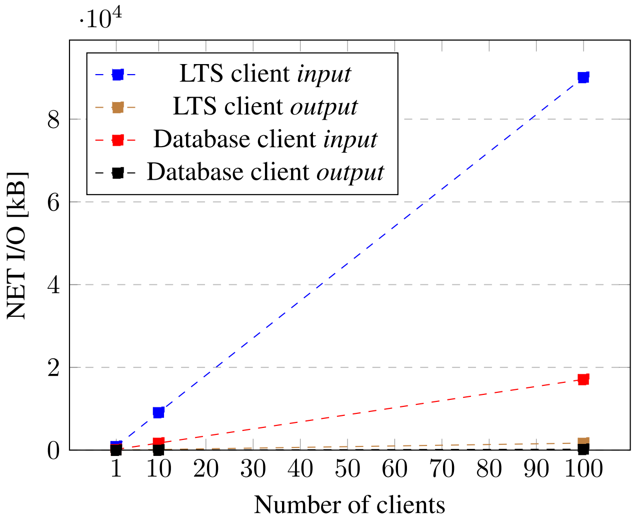

Fig. 5

Bandwidth as a function of the number of clients. This graph shows the result of clients requesting sensor values for a historic and non-historical time range.

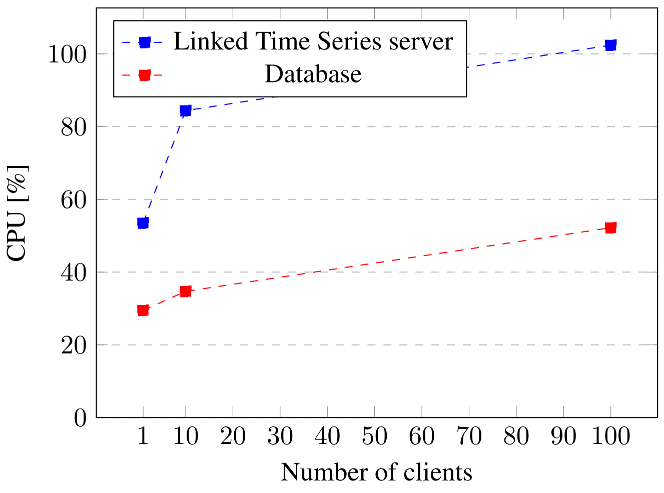

Fig. 6

CPU-percentage as a function of the number of clients. This graph shows the numbers when clients request average values in a time range that has not been ended yet.

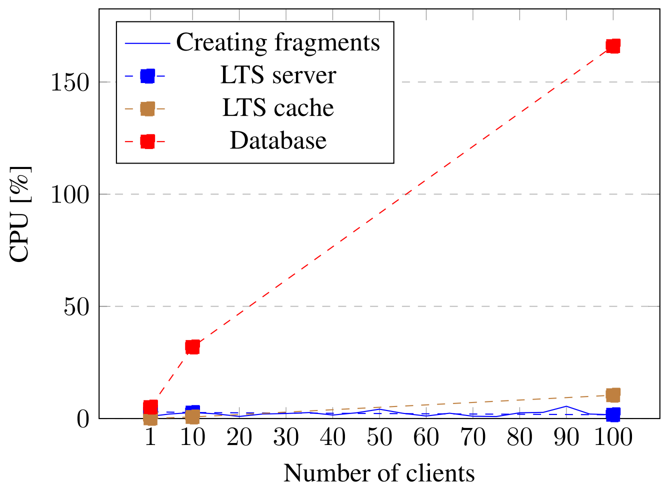

Fig. 4

CPU percentage as a function of the number of clients for requesting average values in a historic time range.

Naam	Opslag
LTS server	4.92 MB
Database	504kB

TABLE I
VALUES FOR USED STORAGE CAPACITY FOR LINKED TIME SERIES SERVER AND DATABASE. THE LINKED TIME SERIES SERVER USES ALMOST 10 TIMES MORE STORAGE THAN THE DATABASE.

required data pieces workable, also resulting in more CPU usage. With the LTS server and Linked Data Fragments in general, a part of the workload is shifted towards the client, resulting in less CPU usage for the LTS server and more CPU usage for the client. From this, we can conclude that Linked Data Fragments can be used to balance the cost for air quality data between data publisher and consumer.

REFERENCES

[1] United Nations, *2018 revision of world urbanization prospects* https://population.un.org/wup/Publications/Files/WUP2018-KeyFacts.pdf

[2] Ahuja, Tanuj and Jain, Vanita and Gupta, Shriya, *Smart Pollution Monitoring for Instituting Aware Traveling* International Journal of Computer Applications, p. 4-11, 2016

[3] Castell, Núria and Viana, Mar and Minguillón, María Cruz and Guerreiro, Cristina and Querol, Xavier, *Real-world application of new sensor technologies for air quality monitoring* ETC/ACM Technical Paper, p. 16, 2016

[4] Vlaamse Milieumaatschappij, *FAQ BelAir* https://www.vmm.be/data/luchtkwaliteit-op-zak-met-belair-app/faq-belair

[5] FIWARE Community, *What is FIWARE?* https://www.fiware.org/2011/05/17/what-is-fiware/

[6] FIWARE Community, *FIWARE Catalogue*, https://www.fiware.org/developers/catalogue/

[7] Julián Andrés Rojas Meléndez, Gayane Sedrakyan, Pieter Colpaert, Miel Vander Sande, Ruben Verborgh, *Supporting sustainable publishing and consuming of live Linked Time Series streams* European Semantic Web Conference, pp. 148-152, 2018.

[8] Ruben Taelman, Pieter Colpaert, Ruben Verborgh, Erik Mannens, *Multidimensional interfaces for selecting data within ordinal ranges* Proceedings of the 7th International Workshop on Consuming Linked Data co-located with 15th International Semantic Web Conference (ISWC 2015), 2016.