Building a Sensor infrastructure for long term monitoring

Harris N[1], James P[1], Dawson D[1], Joncyzk J[1],Pearson D[1]

Newcastle University, School of Civil Engineering and Geosciences, Newcastle upon Tyne, NE1 7RU

December 1st, 2016

Summary

Long term monitoring off a city at many different scales and across many different sectors requires a flexible system to incorporate the heterogeneous nature of the data. He were outline the schema-less approach to data management taken by the Newcastle Urban Observatory.

KEYWORDS: Big-data, schema-less, database, open-source

1. Background

The Newcastle Urban Observatory aim is to create a platform for understanding how the city behaves as a whole. In order to achieve this a city as a whole must be monitored. This is challenging due to complex nature of a city, comprising of many sectors that are not only complex in isolation but also have complex inter-sector relationships (Muller *et al.*, 2013). Therefore a holistic monitoring platform must be developed that is able to deal with the diverse data representing as many sectors as possible.

In most previous platforms where data is assembled of many heterogeneous cross sectoral data sources the data is often sourced through various feeds and kept in isolation. This confinement of the data creates issues when trying to perform any kind of cross sectoral analysis. Therefore for a more successful holistic system a heterogeneous data scheme should be implemented.

As each sectoral data source is diverse storing this feeds together in one cohesive whole can be challenging and requires a flexible data structure that is able to not only suit current data feeds but is also malleable enough to accommodate future data feeds whose structure is unknown.

The traditional structured database approach to either involve creating a table for every data feed which is problematic when trying to query the data, or one large table with multiple columns which not only creates a large amount of redundancy but is impossible to future-proof for newer data feeds. Therefore many cohesive data projects adopt a schema-less approach. The approach adopted here is to use a key value pairs of semi-structured data

Key value stores allow a much more flexible data store as opposed to a database table with strict columns. (Seeger, 2009) outlines this key value approach to storing data, describing it as data usually consisting of a string which represents the key and the actual data which is considered to be the value in the "key - value" relationship. The data itself is usually some kind of primitive of the programming language (a string, an integer, and an array) or an object that is being marshalled by the programming languages bindings to the key value store. This replaces the need for fixed data model and makes the requirement for properly formatted data less strict. Nakamura *et al.* (2011) Designed and implemented a new database approach for storage and delivery of sensor data called uTupleSpace. uTupleSpace is a schema-less style data store consisting of key/value pairs. It was found by using this schema-less approach they were able to meet increases in variety and quantity of sensor data.

2. The current approach

The approach that Newcastle Urban Observatory took was to build a flexible data store in PostgreSQL using the hstore extension (Corti *et al.*, 2014). This approach is also used by openstreetmap for storing their data (Corti *et al.*, 2014). The hybrid approach of creating a schema-less store inside a structured database allows for the ACID nature of PostgreSQL to be harnessed but allowing for varied to be stored in a single store without the need for on-the-fly schema changes. Additionally PostGIS (Obe and Hsu, 2015) the spatial extension of PostgreSQL can also be utilised allow the data to be queried and analysed spatially in the database environment rather than having to carry out spatial operations in an additional middleware layer which can often be time consuming .

One drawback to using the flexible column type hstore is that all of the values of the key value pair are stored as text and whilst when querying the data simple on-the-fly conversions allow both numerical and geographical queries to be made, there is no data side system preventing the insertion of an incorrect data type. Therefore surrounding the database and managing all interactions with the data is a python-based middleware layer. This was is built on top of the pyscopg2 module and is used to perform all interactions and insertions with the database (Westra, 2015).

As the ideology behind the UO was to build a simple system that worked from the outset but was able to grow organically as the system grew, the database initially consisted of 4 tables their schemas outlined in figure 1. With information on the source of sensor feeds going into the sensor_source table, information on individual sensors, including geometry, going into the sensors table. Raw sensor readings going into the sensor_data table and information about each variable recorded being stored in the readings table. This table along with python middleware layer are used to retrospectively flag readings with values outside of a suitable range, i.e. negative rainfall values. On-the-fly unit conversions also take place in the middleware layer using default unit and conversion formula which has also stored in the readings table.

Table Name	Column Name	Column Type	Description
Sensors	Row_num	Int	Row number
	Info	hstore	Sensor information including location
Sensor_source	Row_num	Int	Row number
	Info	hstore	Sensor feed information including documentation
Sensor_data	Row_num	Int	Row number
	Info	hstore	Raw sensor readings
Readings	Row_num	Int	Row number
	Info	hstore	Variable information including default units acceptable value range and unit conversion formula

Figure 1Database structure

This system was able to perform adequately for the start of the project but as the number of readings and daily data traffic increased overtime the query performance on the sensor_data table dropped below at critical threshold. In order to increase efficiency the sensor_data table was partitioned over the year of the reading with a function added to postgresql to generating a set of subqueries to allow the corresponding partition table to be queried.

3. Current Status and Performance

As described earlier the Urban Observatory has grown in volume over time with the number of sensor reading records now approaching 250 million. The growth of the data and feeds are shown in figures 2,3 and 4. With the data not only increasing in amount and daily traffic but also the breadth of the readings that are recorded in the UO system.

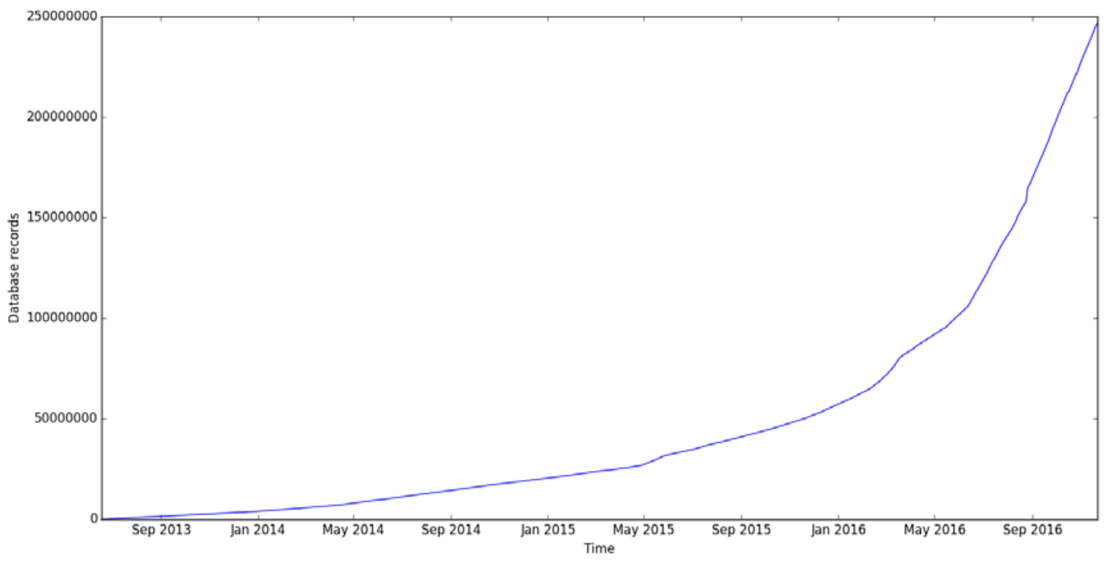

Figure 2 Database records over time

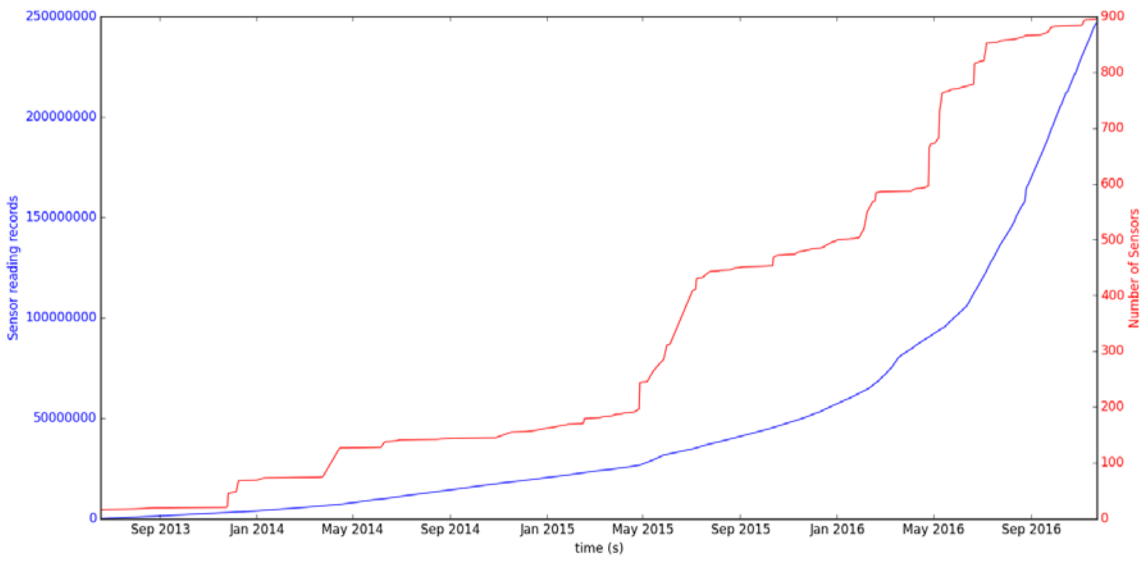

Figure 3 sensor reading and number of sensors growth over time

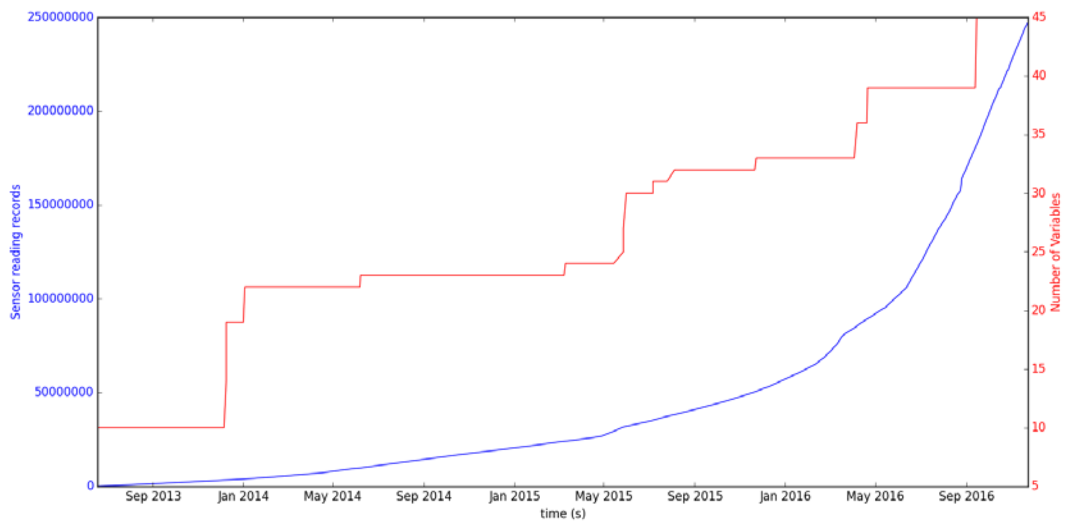

Figure 4 variable growth over time

As the system grows it is important that querying the database remains at a suitable level. The query performance was investigated by running some analysis that queried an increasing amount of data and measured both the response size and time for that query to return the results. Poorly tuned databases not only takes more time than it ought, it also tends to scale badly as data volumes increase. Therefore it is important that there is not an exponential relationship between query time and size of query (Moniruzzaman and Hossain, 2013). From the analysis (figure 5) performed it is invent that there is a linear relationship between both the query time and hours queried and query time and size of response.

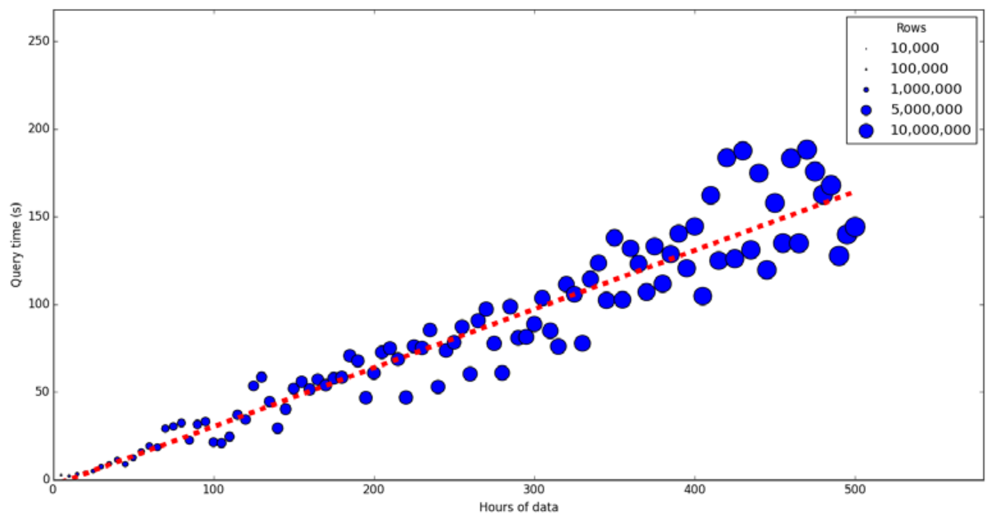

Figure 5 Query perform of hours of data

As well as testing the performance of query with increasing amount of hours of data further analysis was run with keeping the number of hours as 50 but querying on an increasing amount variables and then sensor ids. The performance again showed a linear and not exponential relationship between size of query and time taken shown in figures 6 and 7.

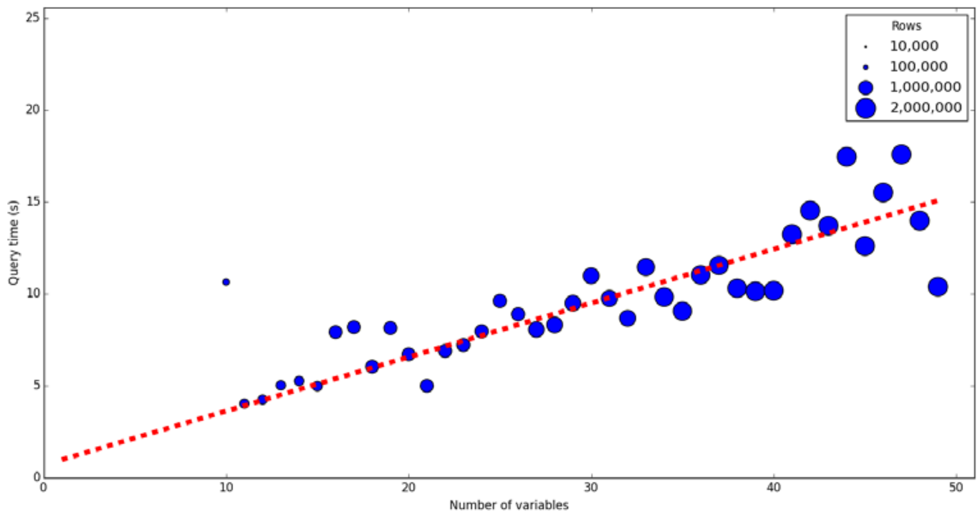

Figure 6 Variable performance

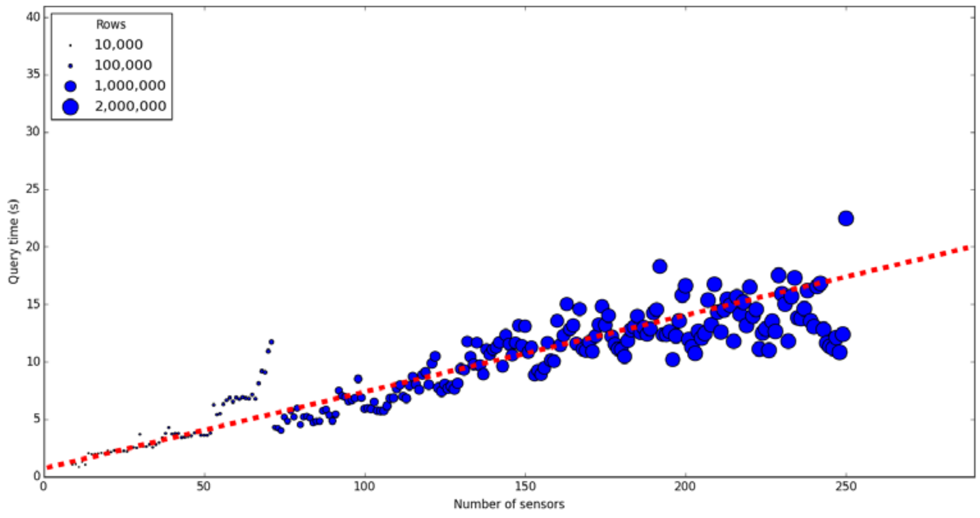

Figure 7 Sensor Performance

As the vision of the UO is to both record and disseminate data in real-time in addition to the query time it is also important that the inserts into the database remain rapid as the table grows. A quick demonstration of the insert performance was tested where an increasing dummy dataset was inserted into the system. The system’s insert rate remains relatively stable as the amount of rows inserted. Figures 8 and 9 show the results, with the insert rate remaining relatively stable as the amount of data inserted increased.

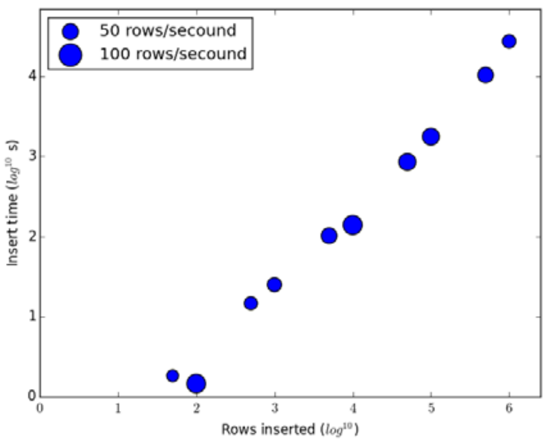

Figure 8 Insert rate

Figure 9 Insert rate

4. Future Work

Whilst the quick analysis shows that the database system performance is of a reasonable level, the current system is a compromise between time series, spatial and attribute queries. And as in the grand scheme of a holistic sensing platform disc space is relatively in expensive therefore rather than feeding data into one database harnessed for query comprise at faster more elegant system may be to duplicate all data inputs into several different databases tuned to perform best at certain query types. I.e. a PostGIS enabled PostgreSQL database for spatial queries, an Influxdb instance for time series queries and an in-memory database such as Redis database instance for live value queries. This will involve the development of a much more complex middleware layer as an elegant means of feeding data into several database instances will need to be developed and well as a query management system that passes a given set of query parameters to the appropriate database.

5. Biography

Mr Neil Harris is a researcher and software developer working on the UO programme. Mr Philip James is a Senior Lecturer in Geographic Information Science at Newcastle University. Prof. Richard Dawson is the Professor of Earth Systems Engineering and co-leads the UO programme of research at Newcastle. Dr Jennine Joncyk is a researcher working on the UO programme with a focus on urban water. Mr Dave Pearson is a technician working on UO deployment.

References

Corti, P., Kraft, T.J., Mather, S.V. and Park, B. (2014) *PostGIS Cookbook*. Packt Publishing Ltd.
Moniruzzaman, A.B.M. and Hossain, S.A. (2013) 'Nosql database: New era of databases for big data analytics-classification, characteristics and comparison', *arXiv preprint arXiv:1307.0191*.
Muller, C.L., Chapman, L., Grimmond, C.S.B., Young, D.T. and Cai, X. (2013) 'Sensors and the city: a review of urban meteorological networks', *International Journal of Climatology*, 33(7), pp. 1585-1600.
Nakamura, T., Kashiwagi, K., Arakawa, Y. and Nakamura, M. (2011) *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*. IEEE.
Obe, R.O. and Hsu, L.S. (2015) *PostGIS in action*. Manning Publications Co.
Seeger, M. (2009) 'Key-Value stores: a practical overview', *Computer Science and Media, Stuttgart*.
Westra, E. (2015) *Python Geospatial Analysis Essentials*. Packt Publishing Ltd.