A Best of Both Worlds Approach to Complex, Efficient, Time Series Data Delivery

Benjamin Leighton[1], Simon J.D. Cox[1], Nicholas J. Car[1], Matthew P. Stenson[1], Jamie Vleeshouwer[1], and Jonathan Hodge[2]

[1] Land & Water Flagship: CSIRO, Melbourne, VIC and Brisbane, QLD, Australia
Ben.Leighton@csiro.au
[2] Oceans and Atmosphere Flagship: CSIRO, Brisbane, QLD, Australia

Abstract. Point time series are a key data-type for the description of real or modelled environmental phenomena. Delivering this data in useful ways can be challenging when the data volume is large, when computational work (such as aggregation, subsetting, or re-sampling) needs to be performed, or when complex metadata is needed to place data in context for understanding. Some aspects of these problems are especially relevant to the environmental domain: large sensor networks measuring continuous environmental phenomena sampling frequently over long periods of time generate very large datasets, and rich metadata is often required to understand the context of observations. Nevertheless, timeseries data, and most of these challenges, are prevalent beyond the environmental domain, for example in financial and industrial domains.

A review of recent technologies illustrates an emerging trend toward high performance, lightweight, databases specialized for time series data. These databases tend to have non-existent or minimalistic formal metadata capacities. In contrast, the environmental domain boasts standards such as the Sensor Observation Service (SOS) that have mature and comprehensive metadata models but existing implementations have had problems with slow performance.

In this paper we describe our hybrid approach to achieve efficient delivery of large time series datasets with complex metadata. We use three subsystems within a single system-of-systems: a proxy (Python), an efficient time series database (InfluxDB) and a SOS implementation (52 North SOS). Together these present a regular SOS interface. The proxy processes standard SOS queries and issues them to the either 52 North SOS or to InfluxDB for processing. Responses are returned directly from 52 North SOS or indirectly from InfluxDB via Python proxy where they are processed into WaterML. This enables the scalability and performance advantages of the time series database to be married with the sophisticated metadata handling of SOS. Testing indicates that a recent version of 52 North SOS configured with a Postgres/PostGIS database performs well but an implementation incorporating InfluxDB and 52 North SOS in a hybrid architecture performs approximately 12 times faster.

Keywords: time, series, timeseries, SOS, OGC, sensor, database.

R. Denzer et al. (Eds.): ISESS 2015, IFIP AICT 448, pp. 371–379, 2015.

1 Time Series Data, Lots is Hard to Manage

Time series data is frequently used to represent environmental properties and processes and is fundamental in environmental science. Hey and Trefethen[1], describe and predict a scientific data deluge. Managing a deluge of time series data so that it can be discovered, understood, accessed and used effectively, is an ongoing challenge.

A simple time series of numerical values requires context to be useful. For example observational time series often include a unit of measure, and some description of an observed feature. Metadata provides a richer description and context and allows more complex queries to filter or group data. Furthermore standards for metadata mean data can be more readily exchanged because compatible systems and data sets can be identified and interfaced.

Very large amounts of data can be collected by sensors or generated computationally. Metadata can further increase the size and complexity of a dataset. Better analysis and use can be made of large time series data that are richly described. Such descriptions are more computationally expensive to process, require more storage, and consume more network bandwidth.

To address these challenges we review and compare some technologies for storing, delivering and managing time series data. We introduce a hybrid approach that merges a lightweight time series database with a standardized metadata-rich service for delivering sensor observations. This approach provides good performance for a both large dataset with rich metadata. In testing this approach performs faster than a standalone combined data and metadata service.

2 Sensor Observation Service for Management

Standards and models for interacting with and describing observations help address the problem of management, and effective use, of time series data.

The open geospatial consortium sensor observation service (SOS)[2] describes a standard way of managing and querying observation data. Services compliant to this standard are capable of returning stored observations and contextual information about related real world features, sensors, and observational procedures. SOS compliant services can provide observation records that conform to the Open Geospatial Consortium's (OGC) Observations and Measurements standard (O&M)[3]. O&M provides a flexible observations model including information about an observed feature, property, and the procedure associated with making an observation. SOS implementations also provide for complex descriptions of sensors, for example, using SensorML[4].

SensorCloud[5] is an architecture and associated application that performs functions similar to those of described by the SOS standard. SensorCloud specifies a service for delivering sensor data and metadata through REST like URL requests. SensorCloud encodes observations and related data using a JSON format. Sensor metadata is closely modelled on the StarFL[6] format. SensorCloud services can

provide information on networks of sensors, sensor platforms, sensors, phenomena reported by sensors, sensor calibrations and can deliver observations produced by a sensor. The SensorCloud services also provide create and insert operations for networks, sensor, platform, and observations data.

Both SensorCloud and OGC SOS standards describe ways of structuring and delivering time series data and related metadata in the scientific domain.

3 Other Databases

Beyond the scientific domain there are many databases and services for the storage and use of time series data. There is a continuum between specialised time series databases and more general purpose databases. Numerous specialized time series databases exist and many are in active development. Development of time series databases is driven by a broad need to access observations but not always in a scientific context. For example OpenTSDB[7], while somewhat general purpose, is typically used to monitor various I.T system metrics. Similarly Cube[8] is generic but was developed for use cases related to monitoring customer data, website, and system performance.

Time Series databases readily integrate with other systems and often provide web service based interfaces. REST interfaces and JSON provide rapid integration with JavaScript frameworks and thus ease integration with web based visualisation tools. KariosDB[9] is a rewrite of, and intended to be an improvement on, OpenTSDB. It provides a REST interface alongside other interfaces including a GUI and a telnet interface. Cube is built on MongoDB[10] and provides a REST like interface. InfluxDB[11] provides a REST interface.

More general purpose databases are also used for time series storage. MongoDB provides the back end database for SensorCloud as well as Cube. PostgreSQL[12] is one of the possible back-ends for 52 North SOS[13]. Cassandra[14] can be readily adapted to store time series data. Cois[15] describes a hybrid approach using Redis[16] and PostgreSQL for an environmental database prioritizing real time event detection. InfluxDB can use a variety of embedded backend databases.

Databases may be optimized for write versus read operations, can be highly normalised and efficient, or may enable rapid prototyping and querying of unstructured or loosely structured data.

4 Standards Support and Interoperability

SensorCloud and OGC SOS-compliant systems provide rich metadata models. In contrast most other time series databases provide minimal metadata and none provided any time series-specific standards conformance.

Non-relational databases, like MongoDB, provide storage of structured documents thus support arbitrary metadata schemas. Relational databases like PostgreSQL support table structures that may accommodate arbitrary metadata schemas. Neither MongoDB nor PostgreSQL provide a built in metadata model for time series data.

Specialised time series database typically provide limited and inflexible metadata capabilities that do not conform to recognised standards. OpenTSDB provides some very basic support through unique identifiers[17]. KariosDB annotates values with tags and a name[18]. Cube is much more flexible and supports arbitrary structured JSON data at each data point. In InfluxDB, a value is effectively a row consisting of multiple columns and thus simple metadata structures can be accommodated.

5 Performance Considerations for Rich Metadata Systems

Standards for interoperability and metadata are particularly important in the scientific domain. Standards provide a basis for a scientific infrastructure where data can be discovered, compared, reused, and through which experiments can be replicated [19][20]. Thus time series databases and services that conform to comprehensive metadata and encoding standard have an advantage over more general database systems and time series specific databases. Performance is always an important consideration: the ability to rapidly query, analyze and retrieve time series data is a requirement for a number of time series use cases. For example, exploratory interactive visualization of time series data is possible if data points can be quickly retrieved, processed and displayed. Similarly fast responses ensure that access to time series data is not a bottleneck in analysis, or when used as an input to a model.

An ongoing concern with the SOS standard is the performance of implementations. The well-known 52 North SOS implementation provides conformance to the SOS 2.0 standard and includes a number of extension methods and formats. Recent releases have addressed performance issues however older 52 North SOS implementations have suffered from numerous performance issues. Results[21] from performance tests for various older versions of 52 North SOS indicate that under a load of 10 concurrent requests every 5 seconds, SOS performs quite slowly. For example a GetObservation request, the average response time for a request for a yearly data set having approximately 500,000 records was 191 seconds. Similar performance issues with scaling SOS to millions of records for thousands of sensors have been discussed in the SOS community[22].

Performance has been greatly improved in 52 North SOS version 4.1[13] however with dataset sizes likely to increase more than linearly, revolutionary performance change will be needed in the longer term. Large performance gains may be possible through alternative approaches to data storage, that behind a SOS interface, leverage dedicated time series databases.

6 An Alternative Architecture

A good candidate for comparison with 52 North SOS's large data capacity is Influx DB. InfluxDB is a "Timeseries, events and metrics" database[23]. InfluxDB is easy to deploy and is packaged with an embedded backend database. Unlike 52 North SOS, InfluxDB lacks any native metadata model and provides no standards-compliant interface.

We developed a hybrid data/metadata architecture, combining InfluxDB with 52 North SOS. We tested an implementation of this architecture to determine whether it provide better performance than a standalone 52 North SOS instance.

Similar Hybrid architectures have been previously proposed. Cox[24][Fig 1.] describes scenarios in which SOS services consume data from other OGC standard Web Feature Services and Web Coverage Services. Bröring et. al. [25] describe patterns for bridging sensors and sensor delivery services in hybrid like architectures. In particular Bröring describes using Twitter as a middleware layer to store sensor metadata.

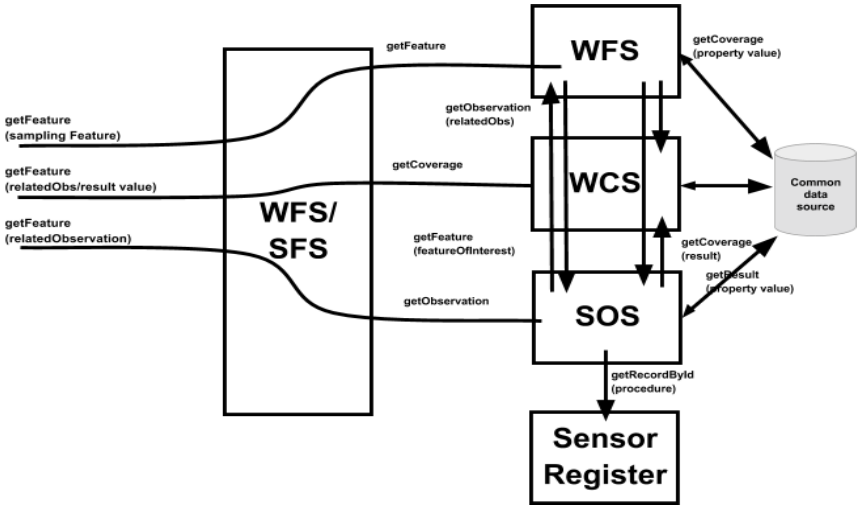

Fig. 1. Hybrid OGC architectures (adapted from Cox[24])

A prototype SOS service has also been developed that, through a heterogeneous systems architecture, provided a partially SOS-compliant service using a Web Feature Service (WFS) back-end[26].

Our hybrid architecture aims to meet multiple use cases. It provides a rich SOS implementation for standards compliant interoperability to enable sophisticated queries. For a subset of queries it provides SOS-compliant responses faster than 52 North SOS configured with a Postgres/PostGIS backend database. The system should enable exploration of data across the temporal domain with a minimum of delay to facilitate use cases such as real time interactive visualisation.

Figure 2 presents a high-level view of the hybrid architecture: SOS requests and responses are handled via a hybrid internet proxy. The hybrid proxy forwards certain requests to InfluxDB while others not able to be handled by InfluxDB are redirected to 52 North SOS.

A test implementation of the hybrid proxy is capable of processing and forwarding only a small subset of SOS queries to InfluxDB: SOS 2.0 requests provided as key value pairs (as URL arguments) where the response format is specified as WaterML2. This subset includes the main data delivery request GetObservation when WaterML2

format is used. The hybrid proxy can process queries that vary by real world feature or property of interest for a feature and can handle a temporal filter that specify results between varying start and end times.

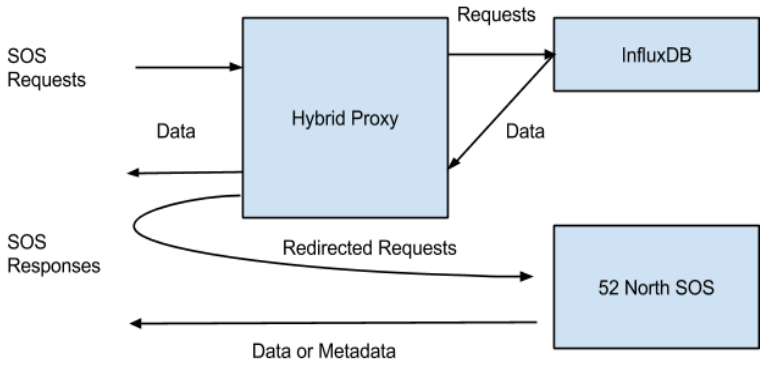

Fig. 2. High-level architecture of a hybrid SOS implementation

SOS queries identified for forwarding to InfluxDB are analysed and used to construct an equivalent InfluxDB query. InfluxDB provides simple responses back to the proxy in JavaScript Object Notation (JSON) form. The hybrid proxy translates the JSON response into WaterML (using lookup tables for variables like unit of measure) and returns it to the requester.

We compared the performance of a realistic 52 North SOS instance as a standalone service to a test implementation the hybrid proxy architecture incorporating the same 52 North SOS instance alongside InfluxDB.

The 52 North SOS version tested was 4.1 and the particular instance contained approximately 14 million weather station records stored in a backend a Postgres/PostGIS. Two configurations were tested: one in which a query was sent to a standalone 52 North SOS instance and another in which the query was sent to a hybrid proxy configuration combining 52 North SOS and InfluxDB. In the hybrid proxy configuration the 52 North SOS dataset was replicated into a parallel InfluxDB instance (version 0.7.3).

In both configurations the test used a SOS-compliant query specified as a URL with key value arguments. The test query varied between the standalone SOS instance and hybrid proxy implementation by the system specifying URL part only with key value pair arguments and format remaining the same. The test machines varied in specification and no attempt was made to compensate for hardware differences or network latencies. However our estimates are that the server 52 North SOS virtual machine was significantly more powerful than the Hybrid test virtual machine and the InfluxDB virtual machine. Therefore we expect that performance results are biased in favor of the standalone 52 North SOS instance.

The test query in both configurations retrieved approximately 52,230 hourly average air temperature records for a weather station between 19-11-2010 at midnight and 13-11-2013 at 13:00.

In both test configurations the request was sent using the cURL application. The cURL "w" parameter and a template file were used to generate response timing information. Server processing time was considered to be the time between time_pretransfer and time_starttransfer. This approach minimised noise related to network overheads and time taken to transport response data. Responses are listed in Table 1 below.

Table 1. Response times for 52 North standalone and 52 North Influx DB hybrid

Run	Hybrid	52 North SOS
First	0.45s	8.485s
Run 1	0.377s	4.891s
Run 2	0.31s	4.594s
Run 3	0.336s	4.282s
Run Average (excluding first result)	0.368s	4.589s

The first response was considered an outlier and was significantly slower than subsequent queries for both systems. It was assumed this was because parts of the databases are cached into memory after the first response. In our test the hybrid approach using InfluxDB generated responses 12 times faster on average than 52 North SOS. Our results are indicative of a significant performance benefit through the hybrid approach nevertheless further study could provide more certainty and help remove confounding factors such as differing machine and network performance.

In addition to high performance a hybrid approach offers other advantages. It maintains rich metadata capabilities. The hybrid proxy forwards SOS queries that can't be handled by InfluxDB to 52 North SOS. In our example InfluxDB replicates time series data stored in SOS but doesn't replicate any metadata. For example, InfluxDB cannot provide a response to a SOS DescribeSensor request because it doesn't store any information about the sensors used to produce a time series. The hybrid proxy can identify queries that can't be handled by InfluxDB and redirect these to the 52 North SOS instance. Thus the hybrid proxy handles a subset of queries quickly via InfluxDB but can also respond to the broader set of possible SOS requests by redirecting these to 52 North SOS. This approach improves performance while maintaining a rich metadata model. Additionally, a more general approach should be possible interfacing to other data stores. These kinds of systems could leverage existing data services and adapt those to return SOS responses. This improves the reuse of data and

reduces the need to duplicate data across multiple systems. In turn this "future proofs" SOS installations against reimplementation if and when data loads require new back-end database implementations.

7 Conclusion

Time series data is prolific. There are many systems to help manage, store and deliver time series data. In the scientific domain some systems also provide metadata capabilities that provide more context for understanding and analysing data. Beyond the scientific domain, many other time series databases and services are being developed. These typically have poor metadata capabilities but may be more scalable and provide faster performance. Near real time visualisation and other use cases requiring rapid retrieval and exploration of data would benefit from a system that is standards based, provides rich metadata and performs fast. The 52 North SOS implementation provides a standards based service for management and storage of time series data along with extensive metadata capabilities. Newer versions of 52 North SOS have improved performance. To investigate whether further performance gains were possible we developed a hybrid architecture that combined 52 North SOS with InfluxDB, a light-weight dedicated time series database. We compared the performance of a standalone 52 North SOS instance and a prototype implementation of the hybrid architecture that coupled the 52 North SOS instance with InfluxDB. Under our tests performance was approximately 12 times faster in the hybrid system. There are performance benefits using a hybrid architecture compared to a standalone 52 North SOS instance other advantages maybe that loose coupling between components readily allows integration of new technologies and reuse of data in existing deployments.

References

1. Hey, A.J.G., Trefethen, A.E.: The data deluge: An e-science perspective. Wiley Sons (2003)
2. Portal.opengeospatial.org: OGC Sensor Observation Service Interface Standard (2014), https://portal.opengeospatial.org/files/?artifact_id=47599
3. Cox, S.: Geographic information: observations and measurements. Doc. OGC (2010)
4. Portal.opengeospatial.org: OGCSensorML: Model and XML Encoding Standard (2014), https://portal.opengeospatial.org/files/?artifact_id=55939
5. Peters, C.: SensorCloudRESTful API, https://wiki.csiro.au/display/sensorcloud/SensorCloud+RESTful+API
6. Malewski, C., Simonis, I., Terhorst, A., Bröring, A.: StarFL–a modularised metadata language for sensor descriptions. Int. J. Digit. Earth 7, 450–469 (2014)
7. Opentsdb.net: OpenTSDB - A Distributed, Scalable Monitoring System (2014), http://opentsdb.net/
8. Square.github.io: Cube (2014), http://square.github.io/cube/
9. Code.google.com: kairosdb - Fast scalable time series database - Google Project Hosting (2014), https://code.google.com/p/kairosdb/

10. Mongodb.org: MongoDB (2014), http://www.mongodb.org/
11. Influxdb.com: InfluxDB - Open Source Time Series, Metrics, and Analytics Database (2014), http://influxdb.com/
12. Postgresql.org: PostgreSQL: The world's most advanced open source database (2014), http://www.postgresql.org/
13. Hollmann, C.: 52 North SOS 4.1 (2014), http://blog.52north.org/2014/09/02/52north-sos-4-1/
14. The Apache Cassandra Project, http://cassandra.apache.org/
15. Aaron Cois, C.: Large-Scale Data Collection and Real-Time Analytics Using Redis - O'Reilly Radar (2014), http://radar.oreilly.com/2013/03/large-scale-data-collection-and-real-time-analytics-using-redis.html
16. Redis, http://redis.io/
17. Metadata — OpenTSDB 2.0 documentation, http://opentsdb.net/docs/build/html/user_guide/metadata.html
18. PushingData - kairosdb - Pushing data into KairosDB - Fast scalable time series database - Google Project Hosting, https://code.google.com/p/kairosdb/wiki/PushingData
19. Haak, L.L., Baker, D., Ginther, D.K., Gordon, G.J., Probus, M.A., Kannankutty, N., Weinberg, B.A.: Standards and infrastructure for innovation data exchange. Sci. 338, 196 (2012)
20. Hendler, J.: Science and the semantic web. Science 299(80), 520 (2003)
21. Tan, F.: SOS 2.0 Performance Test (2013), https://www.seegrid.csiro.au/wiki/SISS4BoM/SOS2PerformanceTest
22. Fwd: ODIP-3 Prototype SOS - Google Groups, https://groups.google.com/forum/#!searchin/ioostech_dev/geoff/ioostech_dev/ThkMPTsrEdA/Sv9_iGib1DAJ
23. InfluxDB Documentation, http://influxdb.com/docs/v0.8/introduction/overview.html
24. Cox, S.: No Title (2007), https://www.seegrid.csiro.au/wiki/pub/AppSchemas/RecentPresentations/IN43C-07_Cox_Info_Viewpoints_Service_Architectures.ppt
25. Broring, A., Foerster, T., Jirka, S.: Interaction patterns for bridging the gap between sensor networks and the Sensor Web. In: 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 732–737. IEEE (2010)
26. Golodoniuc, P.: ThinSOS<SISS4BoM<SEEGrid (2013), https://www.seegrid.csiro.au/wiki/SISS4BoM/ThinSOS