A compromise Archive Platform for Monitoring Infrastructures

Carlos Garcia Calatrava*†, Fernando Cucchietti*, Yolanda Becerra*†
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
E-mail: {carlos.garcia, fernando.cucchietti, yolanda.becerra}@bsc.es

Keywords—Big Data, Stream processing, Polyglot Persistence, Monitoring, Archival, IoT.

I. Extended Abstract

The great advancement in the technological field has led to an explosion in the amount of generated data. Many different sectors have understood the opportunity that acquiring, storing, and analyzing further information means, which has led to a broad proliferation of measurement devices. Those *sensors'* typical job is to monitor the state of the enterprise *ecosystem*, which can range from a traditional factory, to a commercial mall, or even to the largest experiment on Earth[1].

Big enterprises (BEs) are building their own big data architectures, usually made out of a combination of several state-of-the-art technologies. Finding new interesting data to measure, store and analyze, has become a daily process in the industrial field.

However, small and medium-sized enterprises (SMEs) usually lack the resources needed to build those data handling architectures, not just in terms of hardware resources, but also in terms of contracting personnel who can master all those rapidly evolving technologies.

Our research tries to adapt two world-wide-used technologies into a single but elastic and moldable one, by tuning them, to offer an alternative and efficient solution for this very specific, but common, scenario.

A. Introduction: A historical approach

From one-size-fits-all to one-size-for-each

Traditionally, databases have been considered a passive asset: OLTP systems ingested structured data, in order to facilitate daily operations, and the relational model was considered, *de facto*, the standard model.

As soon as hardware evolved, organizations realized the real potential of data and several different technologies emerged, improving the handling and storage of the data in a wide range of scenarios.

In not many years, databases moved from one-size-fits-all[2] to one-size-for-each, where each scenario had a very specific and efficient data model, and each data model had a plethora of different databases to choose from. For example, Graph databases enabled the full potential of social networks and key-value databases became crucial in huge online marketplaces[3].

Polyglot Persistence

Alongside the increasing amount of new Data Analysis and Machine Learning algorithms, BEs encountered a new problem: Systems were evolving and, sometimes, choosing one single data model was not enough. Thus, BEs started to put in practice polyglot persistence[4]: Multiple data storage technologies were incorporated, chosen based on the way data was used by individual applications. Hence, applications started to benefit from different data models at the same time.

B. Monitoring Infrastructures and archive databases

In this research, we understand as Monitoring Infrastructure a set of devices, usually called sensors, where each supervises the state of a specific asset alongside time. The global reporting of those sensors is able to describe the state of the whole system, at a given point in time. Archive databases are meant to store the data that a Monitoring Infrastructure produces. The data obtained during this process is crucial for performing tasks such as predictive maintenance or forecasting.

Unlike BEs, SMEs are hardly ever able to implement Polyglot Persistence: Having an expert in each database technology is difficult. Moreover, some companies cannot afford to keep the historical data forever, not just in a single data model, as it becomes huge. Thus, they usually implement a *sliding window* technique, where only a fixed amount of data is kept: As new data is stored, the oldest data is removed. This generates two main problems: Firstly, data is stored just in a single data model, and analyzing it becomes really slow. Secondly, data is being discarded, so important information is lost.

Finally, for preventing data loss, databases are commonly replicated in different machines, which ends ups in an impressive consumption of hardware resources.

C. Soft Goals

The resulting platform should meet the following soft requirements:

- Allow the ingestion, in real time, of monitoring data
- Use as few different technologies as possible, while enabling fast analysis
- Minimize the amount of storage needed

D. Proposed solution

The platform was built with just two different well-known big data technologies. Also, a Java/Python API is provided in order to interact with the platform.

Apache Kafka

Apache Kafka is a distributed streaming platform, used for building real-time data pipelines and streaming apps[5].

In this scenario, Kafka helps in two different actions: It facilitates the stream data ingestion and, also, thanks to a specifically designed Java consumer, compacts and re-structures the data before sending it to the database.

MongoDB

MongoDB is the most popular NoSQL database[6]. It follows a schema-less design: Data Architects define a basic schema, but altering it or adding new fields does not enforce any global-schema modification, which allows a great flexibility. Hence, each record is, theoretically, schema-independent from the other ones.

In this platform, MongoDB was tuned for handling a columnar schema-full data model, apart from the standard document schema-less data model. Thanks to this modification, MongoDB behaves like a Polyglot Persistence system itself, but just using a single database technology.

This not only reduces the number of used technologies and needed experts, but also allows Polyglot Persistence Intra Communication: In traditional polyglot persistence systems, each data-model is held in a different technology, making the communication between them fairly difficult. By placing different data models under the same logical database, it is even possible to perform queries that benefit from different data models at the same time.

Furthermore, it also tackles the problem of database replication. Although both data models are in the very same logical database, they can be placed in different physical databases, by sharding. Thus, a data model can serve as a replica for the other one, and vice-versa: If a machine fails, its data can be recovered from its *sibling* database.

E. Experiment design

In the scope of this extended abstract, the experiment set up consisted in feeding the archiving platform with a 4-month pseudo-streaming factory simulation, that incorporates 3000 different sensors, producing data every 30 seconds. More than 1000 Million records were generated. The system was limited to 4 GB of RAM, and data was stored in a traditional HDD using the ZSTD compression technique.

The performance metrics were obtained in independent executions, by clearing RAM and cache. Each execution was performed 3 times, keeping the AVG.

The executions consisted in running three common queries. Each query asked for the data of 10 random sensors, either for 500 random timestamps or for a 3-month time range. Also, this 3-month time-sequential query was divided into two different queries: The first one obtained the RAW values, and the other one *resampled* them, obtaining the mean value per sensor and hour.

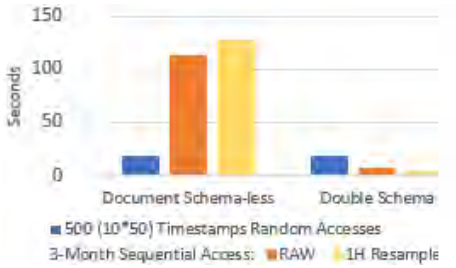

Fig. 1. Query execution time comparison for single vs multi-schema database.

F. Evaluation

As seen in Figure 1, by adding a columnar schema-full storage to the database, time-sequential queries reduce their execution time drastically, as this model captures the relationship between same-sensor measurements.

However, for Random Access queries, the document schema-less design continues being preferred, as it captures the relationship between same-time measurements.

Finally, regarding the disk consumption, the document schema-less design consumed 15.8GB (2-replica + arbiter), while the double schema design needed 12.8GB.

G. Conclusion and future work

The built Archiving Platform keeps a compromise between needed resources and efficiency, while providing an easy-to-use API. It has shown to speed up querying while reducing disk usage, in comparison to traditional schema-less designs.

As future work, it will be important to implement queries that benefit from both data models at the same time, and to add an automatic data recovery mechanism between data models, thus allowing a fully functional replica set.

REFERENCES

[1] P. Golonka *et al.*, "Future archiver for cern scada systems," in *ICALEPCS '17*.

[2] M. Stonebraker, "Technical perspective: One size fits all: An idea whose time has come and gone," 2008.

[3] (2020) Amazon key-value usage website. [Online]. Available: https://aws.amazon.com/en/nosql/key-value/

[4] M. Fowler and P. Safalage. (2012) The future is polyglot persistence. [Online]. Available: https://martinfowler.com/articles/nosql-intro-original.pdf

[5] (2020) Apache kafka website. [Online]. Available: https://kafka.apache.org/

[6] (2020) Database engines website. [Online]. Available: https://db-engines.com/en/ranking

Carlos Garcia Calatrava received his BSc degree in Informatics Engineering from the Facultat d'Informàtica de Barcelona (FIB), Universitat Politècnica de Catalunya (UPC), Spain, in 2016. Two years later, he completed his MSc degree in Innovation and Research in Informatics from the very same university. Regarding his professional background, Carlos has worked in the European Organization for Nuclear Research (CERN, Switzerland) and in the National Institute of Informatics (NII, Japan), among others. Currently, he is working at CASE-BSC, while pursuing a PhD in Computer Architecture.