Martech 2021.
Marine Technology Workshop
16,18th June. Vigo, Spain

Use of TimescaleDB as a database for ocean-meteorological data storage

Ignacio G. Liaño[1], Marta V.Fernandez[2]

[1] *Unidad de Tecnologías Marinas (UTMAR), Fundación CETMAR (Centro Tecnológico del Mar), C/Eduardo Cabello s/n 36208 Vigo (Pontevedra)*
E-mail: igonzalez@cetmar.org
Tel:(+34) 986 247047

***Abstract* – Traditionally, time series databases (TSDB) are usually NoSQL, due to an improvement issue in the processing of large amounts of information and infrastructure management. TimeScale is a PostGreSQL extension that provides storage-level improvements that increase reading and writing processes to NoSQL levels and maintains the fundamental SQL tools for performing temporary space queries of ocean-meteorological data.**

***Keywords* –** *TSDB, SQL, TimeSeries, OceanDataSet, TimescaleDB*

I. INTRODUCTION

The monitoring of the environment is essential to know and study the different phenomena that occur in the natural environment. [1]. Atmospheric and oceanographic data sets share many characteristics. They can be very large; many cover limited periods of time and have a limited spatial extent; gaps (lack of data) and outliers are common; The spatial distribution of several observation networks is uneven; and, often, the time series of data are not homogeneous. The data sets contain variables that, in general, are not independent in time or space; therefore, most of the variables must be seen within a multivariate context [2].
A time series database [3] (TSDB) is a database optimized for timestamp data. These databases have the ability to provide queries with subsamples, gap filling or aggregations throughout the time series, must be stored efficiently to be inserted and retrieved quickly. TimescaleDB [4] is implemented as an extension on PostgreSQL, and exposes what look like singular tables, called hypertables, that are actually an abstraction or a virtual view of many individual tables holding the data, called chunks (created by partitioning the hypertable's data into one or multiple dimensions).
We will proceed to perform a series of tests to test the performance of TimeScale and compare it with other databases of the same style. Small comparisons will be made both in writing and reading data. Special emphasis will be placed on reading large datasets.

II. CREATE DATASET

We assume that we have 12 oceanographic stations (buoys), which have many sensors and one of which is the battery voltage of the station. The structure of the data is as follows: name (*String*), function (*String*), volt (*Float32*), volt_std (*Float32*). It used a sines and cosines function that simulate coherent temporal data. We use a random function with a probability of 0.01 that the data is not acquired (for simulate an error in the acquisition. For the standard deviation, we choose a normal probability. With all these considerations we generate the time series and add the values of the variables for a range of 10 years. We will get approximately 525600 data per year and season, total, for 10 years and 12 stations: approximately 63 million

III. PERFORMANCE TESTS

A) A write performance test was performed simulating data entry for 10 years from 12 different stations. This gives us approximately 63 million data. To match things, and avoid cache and buffer problems, we will proceed to write to the database in blocks of 10,000 items at a time. TimescaleDB being the fastest.
B) Temporary data reading
In this test, all the data contained in the table between two ranges of dates are requested. TimescaleDB is the winner, but keep in mind that MongoDB does not have Sharding activated.
D) Maximum and minimum reading
To test comparison features, we will perform a maximum and minimum search test in a time range.
The application has to perform comparative tasks on numbers, testing how the database works byte comparator. The conclusions are almost identical to the previous test: the penalty for having to consult all the data makes MongoDB take a long time compared to the other two.
D) Reading of arithmetic means
In this test we check the ability of the application to perform cumulative operations on the data. The arithmetic averages

of the chosen time range are calculated. We can test how the application behaves before a mathematical calculation at the database level. The same problem mentioned above reappears.

E) Reading hourly metrics

In this case, we make a real request for aggregation on the generation of hourly metrics. To do this, you must take representative measurements of each hour (60 measurements per hour) for each station.

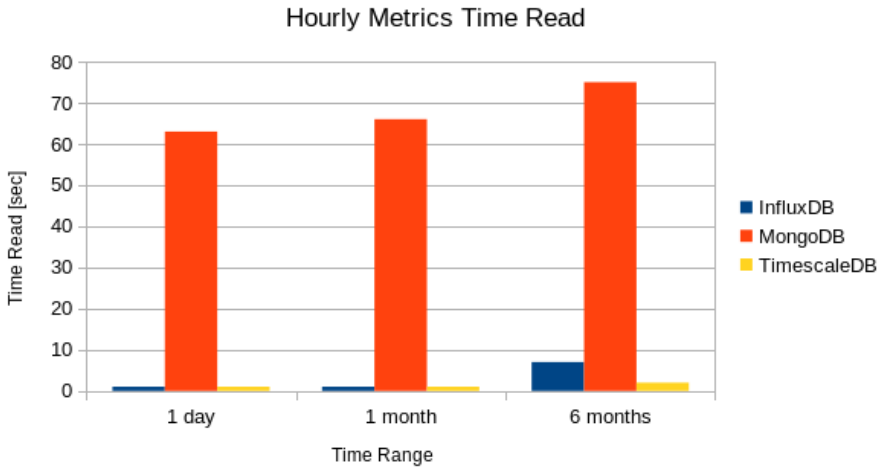

Fig 1. Time metric results

It can be seen that TimescaleDB provides better calculation algorithms than InfluxDB, since previous aggregation requests were on par, but now TimescaleDB has been better.

Possibly due to the fact that by leveraging the entire postgreSQL infrastructure, the algorithms are better debugged and better performing.

IV. CONCLUSIONS

What has pleasantly surprised us is the application of TimescaleDB, which although very young (about 2 years), has spectacular performance. And it is also compatible with the entire ecosystem of PostgreSQL, GIS, pure SQL queries and more. It is just one more extension and, perfectly compatible with the PostGIS extension and integrable in data visualization programs as Grafana.

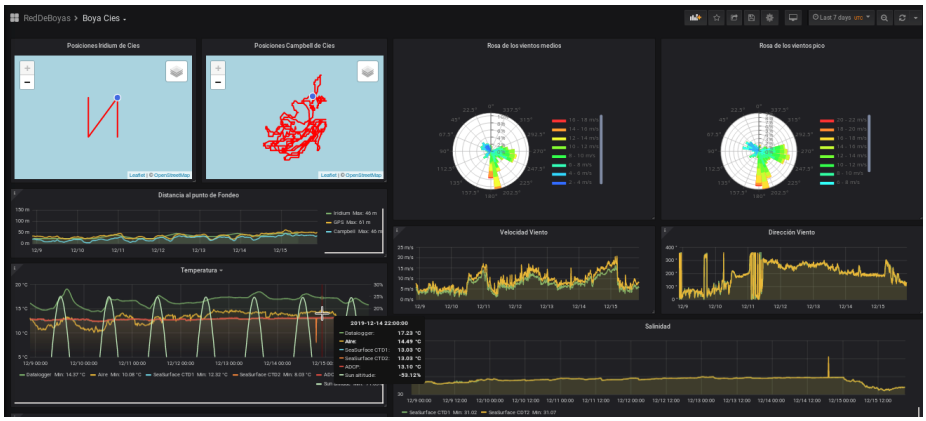

Fig 2. Grafana data monitor

[1] M.Estrada, E. Álvarez, A. Barragán, O. Bermúdez, M. García, A. Lavín, P. Masqué, F. Pérez, J. Piera Reflexiones sobre la gestión y custodia de datos oceanográficos en España. Recursos existentes y recomendaciones para el futuro, *Scientific Committee on Oceanic Research*, 2005

[2] Ifremer, Data Management Handbook: Handbook for Data Management activities regarding data flow and data integration, *Deliverable AtlantOS 633211*, 2016

[3] Namiot, Dmitry, Time Series Databases. 2015

[4] TimeScale Software, [online] https://www.timescale.com/ Jan 2020.