Semi-structured Data Management in PostgreSQL: Competing with MongoDB's Performance

Yudisney Vazquez Ortíz[1,*], Lisleydi Mier Pierre[1], Anthony R. Sotolongo León[2]

[1] University of Informatics Sciences, Havana, Cuba
{yvazquezo, lmpierre}@uci.cu
[2] ARKADIOS K&T, Santiago de Chile, Chile
asotolongo@gmail.com

Abstract. The use of semi-structured data has been a trend in recent years, new tools have been emerging for its handling among which, the NoSQL managers are the most important ones, being documents store, and within them MongoDB, the most widely used. However, these managers do not overlap with relational systems, since each type ensures the functionalities for which they were developed. This is why many companies implement solutions that involve both. PostgreSQL has gradually incorporated non-relational features, which can be an advantage because it could avoid the need of a third tool. One step in this direction was the inclusion in its version 9.4 of the JSONB data type. The purpose of this paper is to evaluate the performance of PostgreSQL against MongoDB in regards to their response times, evaluating on loading data and querying, concluding that PostgreSQL is making progress in the implementation of non-relational features, making it an option to be considered for semi-structured data management.

Keywords: json · jsonb · MongoDB · non-relational PostgreSQL features · semi-structured data

1 Introduction

The gradual use growth rate of databases NoSQL has been a trend in recent years; mainly due to the scalability and speed in their response times as well as variability of their schemes, higher than those that could offer relational systems [1]. These characteristics are achieved, mainly, by not implementing the relational model, not allowing joins and having a distributed architecture, which impacts positively on their objectives of offering flexible ways of manipulating data to gain in speed against the high volumes of data generated nowadays.

Such growth has led to a diversity of tools that are classified according to the form of storage, among others, in data oriented per column, key-value, graphs and documents store databases. The documents store saves the data as documents; defines a unique key for each record; allows searches by key-value, indexed by its properties and the execution of more complex queries, reasons why they are widely used [2].

According to DB-Engines scale [4], the database NoSQL type with higher popularity in recent times has been MongoDB [3]. MongoDB, implemented by 10gen [5] in C++, is scheme free, uses its own document system known as BSON and provides high performance, high availability and automatic scaling, offering fast response times that have become a highly valued and used option.

PostgreSQL, meanwhile, has incorporated non-relational features to offer more options, getting closer from version 9.3 to the times offered by MongoDB, according to an experiment [6]. These features are extended in version 9.4, says EnterpriseDB [7], improving considerably the performance by incorporating the JSONB data type.

The acceptance that has had MongoDB with its excellent response times is a good gauge for comparing PostgreSQL and to determine how close the system is coming, on non-relational features and semi-structured data management, to the top. Hence, the objective of this research is to evaluate the performance regarding this indicator against MongoDB.

2 Methods for assessing the behavior of PostgreSQL

PostgreSQL is an object-relational database management system that, thanks to its extensibility, has allowed the incorporation of new features designed to streamline and add flexibility to data management.

HSTORE and JSON data types provide management options of data without scheme, with the advantage of meeting the ACID properties [7]. HSTORE was added in 8.2 version as a Contrib module that implements a data type, with the same name, to store key-value pairs within a single value of PostgreSQL; useful in situations such as rows with many attributes rarely examined or semi-structured [8].

The ephemeral storage was added in version 9.1 by implementing UNLOGGED tables, option specified when creating tables that details data stored in them are not written to the WAL, which makes them considerably faster than ordinary tables, but it does not guarantee the permanence of the data in case of failure [8].

JSON was added in version 9.2 to support data storage in its format and ensure validation. Since then, it has been improved, adding to version 9.3 functions to work with it and in version 9.4 the JSONB data type was improved in efficiency [8].

With these features, PostgreSQL allows the following elements, making PostgreSQL a valid option for supporting applications requiring non-relational features:

- Storing data in a fast way and data schema free.
- Reading relational data from a table, returning it as JSON and vice versa.
- The integration of SQL statements with JSON and HSTORE, executed in the same ACID transactional environment and based on the same planner and optimizer.

2.1 Design of performance tests between PostgreSQL and MongoDB

Being JSON one of the most popular data exchange formats in the web, supported by several NoSQL databases, we used it to determine the progress in non-relational capabilities offered by PostgreSQL against MongoDB. In the following table were defined the indicators and metrics evaluated during the execution of comparison tests, they are essential elements in determining the performance of the database server.

Table 1. Indicators and metrics defined for testing between PostgreSQL and MongoDB

Indicator	Metrics	Unit of measurement
Database size	Database size	Mb
Database server response times	Loading data	Milliseconds
	Selection query	Milliseconds

To perform the tests:

- Two instances of a PostgreSQL 9.4 database server and a MongoDB 3.0.2 database server, both with their initial configuration, were used.
- One workstation was used with Intel Core i5 CPU to 2.8GHz, 4Gb RAM, 500Gb 7200rpm hard disk and Windows 8 Pro operating system.
- Five queries were defined which return 30 thousand to just over 80 thousand documents that comply with (1) one condition, (2) more than one condition, (3) more than one condition and ordered by any property, (4) at least one of the conditions and ordered by some property and (5) an aggregate function.

The tests were performed 3 times, the times were averaged to obtain an approximate value (except the databases size), including: (1) load of 1, 2 and 4 million documents in both JSON database servers; (2) determining the databases size once loaded JSON documents and; (3) select query execution of random records in both databases.

3 Results of the execution of performance tests

Loading JSON files was performed in 6 collections in MongoDB (with and without data compression) and in 12 tables in PostgreSQL (with and without the UNLOGGED option and using JSON and JSONB data types). As shown in Figure 1, the PostgreSQL response times loading records are lower in all cases to MongoDB using compressed and uncompressed data, only the loading of the 1 and 2 million records JSONB outnumber MongoDB 5 and 11 seconds respectively.

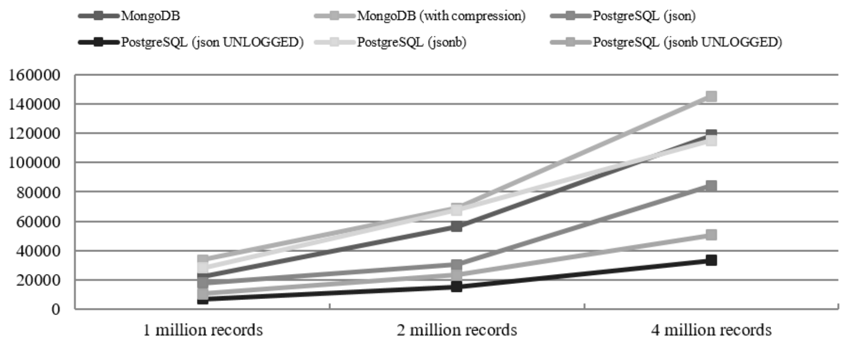

Fig. 1. Response times in load of 1, 2 and 4 million records JSON in both databases systems

As shown in Figure 2, PostgreSQL only occupies 36.02% storing JSON data type and 42.85% storing JSONB data space of that used by MongoDB without additional configuration; while MongoDB using the snappy method can reduce to as much as 10% of the original size.

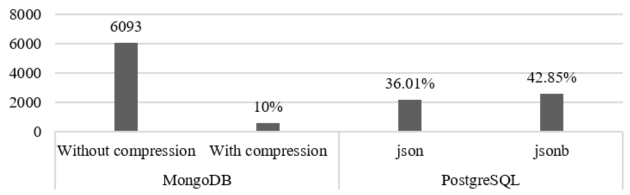

Fig. 2. Database size with 7 million records JSON in both databases systems

Select queries were executed in tables/collections of 4 million records. As shown in Figure 3, PostgreSQL response times on tables with the same options for recording data in the WAL and the same data type are similar, with no appreciable differences. However, there is a considerable improvement in the times obtained between JSON and JSONB, proving that JSONB is, indeed, more efficient to consult all the data 4.3 times faster than using JSON. Meanwhile, even though MongoDB compression saves disk space, it does not achieve the same efficiency in return, since the not compressed option is 1.8 times faster.

Analyzing each query and the times obtained in MongoDB without data compression and PostgreSQL using JSONB, it is evident that even if the new data type increased considerably the efficiency of PostgreSQL, MongoDB is still faster (2.2 times) in the return of data that meet certain conditions and ordered by some property. Not being the same when making groupings, where PostgreSQL is 5.9 times faster.

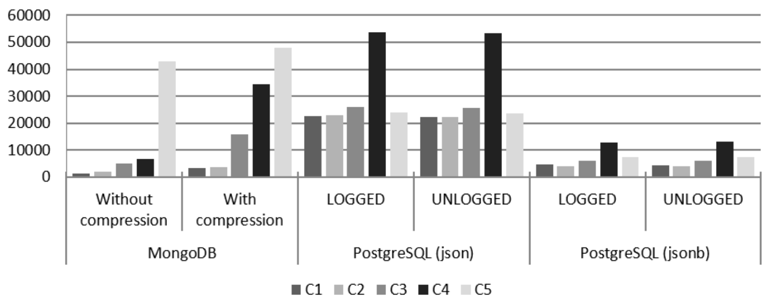

Fig. 3. Response times for execution of random records selection query in databases systems

4 Conclusions

PostgreSQL extensibility has allowed the incorporation of non-relational features to a database system initially object-relational. Of these new features the most important ones include JSON data types and ephemeral storage, making the addition of JSONB a significant improvement in performance. These characteristics were evaluated against MongoDB, showing that version 9.4 of PostgreSQL has greatly improved its response times.

It is needed to recognize that MongoDB is faster when selecting records that meet certain conditions and sorting options, but, PostgreSQL has come very close to the excellent response times of MongoDB.

5 References

1. Leavitt, N.: Will NoSQL Databases Live Up to Their Promise? In Computer 43 (2), pp. 12-14 (2010).
2. Tiwary, S.: *Professional NoSQL*. John Wiley, pp. 10-20. (2011).
3. DB-engines. DB-engines ranking. *DB-engines*. http://db-engines.com/en/ranking. Accedido el 24 de noviembre de 2015.
4. DB-engines. DB-engines ranking - Trend of MongoDB Popularity. *DB-engines*. http://db-engines.com/en/ranking_trend/system/MongoDB.html. Accedido el 24 de noviembre de 2015.
5. MongoDB. Introduction to MongoDB. MongoDB. http://docs.mongodb.org/manual/core/introduction/?_ga=1.20983161.39009857.1428546691#introduction-to-mongodb.html (2015).
6. Sotolongo León A. R. ; Vazquez Ortíz Y.: Evaluación de características NoSQL en PostgreSQL. *Semana Tecnológica de FORDES*. http://semanatecnologica.fordes.co.cu/?q=node/856 (2013).
7. EnterpriseDB. Using the NoSQL Capabilities in Postgres. *EnterpriseDB*. http://info.enterprisedb.com/rs/enterprisedb/images/EDB_White_Paper_Using_the_NoSQL_Features_in_Postgres.pdf (2014). Accedido el 26 de noviembre de 2015.
8. PGDG. PostgreSQL 9.4.0 Documentation. *PostgreSQL*. http://www.postgresql.org/docs/9.4/static/index.html (2015). Accedido el 26 de noviembre de 2015.