Comparison of MySQL and MongoDB with focus on performance

Petr Filip
Faculty of Electrical Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
petr.filip1@student.upce.cz

Lukáš Čegan
Faculty of Electrical Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
lukas.cegan@upce.cz

2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS) | 978-1-7281-9167-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICIMCIS51567.2020.9354307

***Abstract*—Almost every web application uses a database for storing data. Storing and retrieving data causes a delay between request to the system and its response. This has an impact on user-experience.**

The market offers many types of databases and each is more suitable for different types of applications. The main difference is in data-structures which are used for data storing. Data-structures directly influence the performance of the database system and all dependent systems.

In this paper, an overview and brief comparison of four types of NoSql databases has been made. Next, benchmark comparing performance of MongoDb and MySQL in different situation are discussed. Benchmarks are focused on operations such as inserting, updating, and deleting data with transactions and without transactions. The added value of the benchmarks is evaluated in terms of cost of indexed fields in the above-mentioned situations.

***Index Terms*—Database Benchmark, NoSQL, MySQL, MongoDB**

I. Introduction

Almost every web application is database powered and it has huge impact on web performance and user-experience. The time for getting required data from the database is crucial and every second decides how satisfied is the user with the website or web application. In many cases is not possible to cache the whole output for an individual user because results are unique for each user request. This can be mainly influenced by user permission or user specific requirements on data selection.

One of the most favorites technologies for web development are PHP and Spring Boot. The first release of PHP was in 1995 and the language is still developing. Because PHP was one the first programming languages for web development, popularity has been increasing until today. Other advantages of the PHP are very good learning curve, wide tutorial availability, license politics and many affordable web hostings. One of the most popular PHP based CMS (Content Management System) is WordPress which is used by 35.0% [1] of all the websites.

While PHP is a language, the Spring Boot is a Java-based framework for rapid web application development. The Spring Boot is mainly used by the business sector because it offers a great developer ecosystem with many packages for business needs.

As was mentioned before, those technologies are used for web development. Even in the case of simple or complex applications, getting data from the database is a basic requirement. According to stats [2], relational databases are among the most popular.

This article is focused on performance improvement of database web applications with MongoDB which can replace the very popular and often used MySQL database. The following sections describe basic differences between SQL and NoSQL databases and their basic divisions in detail, with reference to advantages and disadvantages. Next, experimental benchmarks with focus on database performance has been compared. This is followed by discussion about the result and its impacts on user-experience.

II. Technical background

A. SQL vs NoSQL database

Traditional relational databases are usually vertically scalable and offer strong data consistency because ACID (Atomicity, Consistency, Isolation, and Durability) properties are fulfilled. The consistency was considered for almost every system. As is indicated, the scalability is an issue because one powerful machine serves all requests.

On the other hand, NoSQL databases are designed with respect to scaling in mind, which leads to a weak consistency. This approach is known as BASE (Basically Available, Soft-state and Eventually consistent). Some of the current NoSQL database solutions offer strong consistency, too [3]. For the strong consistency, syncing of replicas is needed and it slows down the whole system.

Another main difference between SQL databases and NoSQL databases is in data-structures for storing data. This is shown in fig. 1. Data in relational store is organized into tables which are composed from a fixed number of columns with specified data type. The data is stored row by row consecutively. Connections between tables are defined as a relationship of foreign keys and primary keys. The move from SQL database to NoSQL database is also a research subject [4], [5] . For getting data from relational databases SQL (Structure Query Language) is used.

B. NoSQL types

The NoSQL databases use different data structure and approaches for storing data. According to data organization,

978-1-7281-9167-6/20/$31.00 ©2020 IEEE

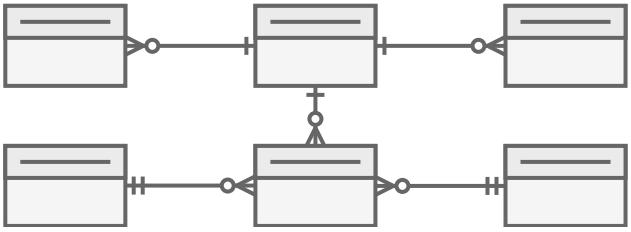

Fig. 1: Relational SQL database concept

it is possible to distinguish NoSQL databases into four basic groups [6]–[9]. The basic concepts are shown in fig 2. One of the biggest disadvantages of NoSQL databases can be disunited query languages which can complicate migration between different NoSQL systems. Also, the SQL allows to creation of complex queries such as combinations of aggregation. In the case of NoSQL databases it is very complicated to write more complex queries.

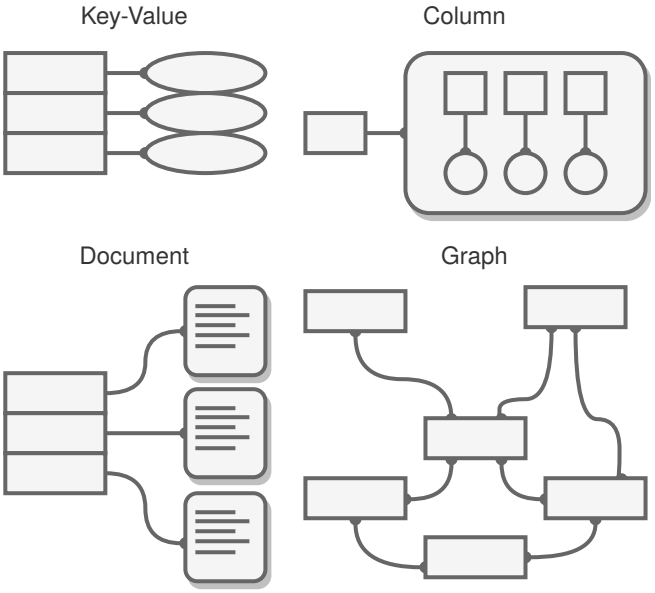

Fig. 2: NoSQL database types

1) Key-Value databases: The Key-Value store is a very simple data structure where a value corresponds to one key, and no key is stored multiple-times. It means that it is possible to get a value by a direct key hit with ($\mathcal{O}(1)$). This native feature puts key-value database into the role of high performance storage with high throughput. The value can contain any kind of data such as image, string, etc. In general, the values are stored as uninterpreted data. This data-structure is usually represented as HashMap or Dictionary. A disadvantage is its inability to search by value with good access time. Next, it is is possible to get an entire value even when only part is needed. This kind of database is very often used as cache. As a representative of this category, Redis [10] and Memcached [11] can be considered.

2) Document databases: The document databases extend the before-mentioned key-value databases. The main extension is that the value is represented as a document and its content can be analyzed. It allows operations to be done with stored documents such as filtering and indexing, which leads to faster search queries. Documents are usually stored as JSON which allows nested documents [12] to be done. Nesting of documents are common techniques to overcome joining tables which is a performance issue. On the other hand, joining data across documents is not possible and it causes another issue with missing relationships. The data models have to be well prepared for required querying.

These kind of databases are very flexible and its use-cases are practically unlimited. In context of websites and web-applications it reduces the number of database queries and joins, because documents can contain all related data. For example, the whole content of each web-page can be stored as one document. The main representative of document databases are MongoDB [13], CouchDB [14] and RavenDB [15].

3) Graph databases: For most of the aforementioned NoSQL databases, it is not possible to handle relational data. Graph databases are based on graph theory, where basic terms are vertices and edges [7]. In basics, nodes represent entities with their properties, and edges represent relationships between them. The graph databases are designed and optimized for traversing of relationships. In the case of the traditional relational database, many-to-many relationships are made via relationship tables and an explicit joining query is needed. So, graph databases found usage in applications where many relations between entities occurred – such as social networks and real-time recommendation systems [16]. The main representative in this category is Neo4j [17].

4) Column-oriented databases: As the name suggest, the data is stored in columnar form. From this approach arises the advantage of scaling, where each column can be stored in a different storage. This implicates fast querying for cases when only data from selected columns is needed. Getting of a value is by row-key and column-key. The resultant rows are composed as a joined columns output. As the most important representative, it is possible to consider Apache Cassandra [18] and Apache HBase [19].

III. Experiments

For the comparison two very popular databases have been chosen. MySQL represents the category of relational SQL databases while the MongoDB represents the NoSQL category of databases. At first, used terms have to specified, see table I.

TABLE I: Mysql vs. MongoDB terms

MySQL	***MongoDB***
Database	Database
Table	Collection
Row	Document
Column	Field

For context of experiments, basic information is needed to be provided. MySQL uses inverted index data-structure

for full-text search. Conversely, MongoDB use B-tree data-structure for full-text indexing.

Basically, Mysql supports transactions natively across the whole database. In the case of MongoDB, document transactions are natively transactional. MongoDB offers multi-document ACID transactions from [20] version 4.0.

With multi-document ACID transactions, it is possible to maintain the same data integrity as is guaranteed in traditional databases. A key difference is that MongoDB allows transactional guarantees to maintained even across highly distributed clusters spanning the globe.

The main disadvantage of MySQL is missing reference integrity between documents, so everything about the integrity needs to be supported at the application level.

A. Experiments

The goal of the research is to find out how much the indexes affect the speed of queries. The experiments compare the same type of queries with different conditions such as queries on dataset with full-text index (FTIDX), with index (IDX), and without index (NOIDX). These conditions are combined in transactional (TX) and non-transactional (NOTX) queries.

Types of queries which have been made are basic data modification operations such as insert, update, and delete. For the experiments, 10 items have been randomly picked from the collection and used for processing. According to testing, processing has been made in series. Each test has been repeated 100 times with different variants. See algorithm 1. Basically, each operation needs compensation which will keep the same items in the data-collection for the next benchmark round.

Algorithm 1 Algorithm for testing

```
for i = 0 to 100 do
    items ← getRandomlyTenItemsFromAll
    delete(items)          {// Primary operation}
    insert(items)          {// Compensation operation}
end for
```

As testing data, openweather data has been used, which contains cities and coordinates (see listing 1). The data set size is 29.69MB which is equal to 209579 records. Records also contain names with accents. As column/field index, *name* has been selected.

Listing 1: Example of input data

```
[ {
    "id": 519188,
    "name": "Novinki", // indexed column
    "country": "RU",
    "coord": {
      "lon": 37.666668,
      "lat": 55.683334
}, ... ]
```

B. Testing environment

The testing machine has been equipped with operation system Debian 10. As hardware, a laptop with Intel i5-6300 CPU and 24 GiB of RAM was used. MySQL version 8.0.20, MongoDB version 4.2.0, Docker version 18.09.1 and Java 1.8 was also used.

For the testing purpose, java based Spring Boot (2.2.1) application was developed for detailed measurement. For data queries, the test application uses MongoTemplate, JdbcTemplate and built-in transaction managers.

Databases were run in Docker with default settings without any other tuning.

IV. Results

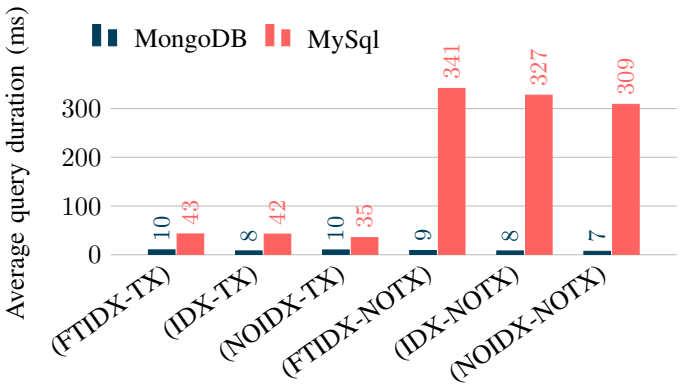

Fig. 3: Insert queries

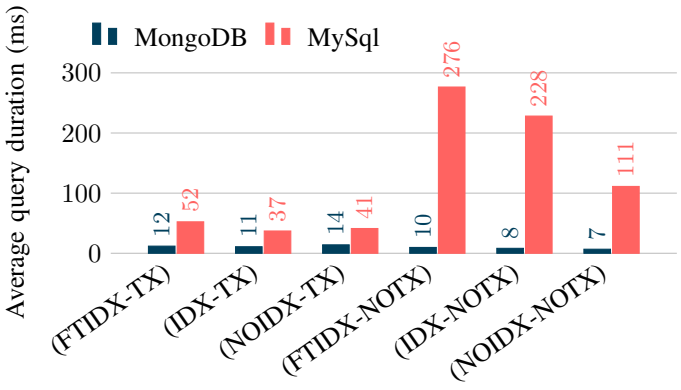

Fig. 4: Update queries

As is mentioned above, queries with same result have been made in experiments. Results for insert, update and delete are displayed on figures 3, 4 and 5. Displayed plots show side-by-side results. Results are represented as average duration of one query in milliseconds. All of those query types are tested in transactional and non-transactional modes for each database.

As can be seen at first glance, MongoDB shows several times better results. Also very interesting, is the difference between transactional and non-transactional queries for MySql

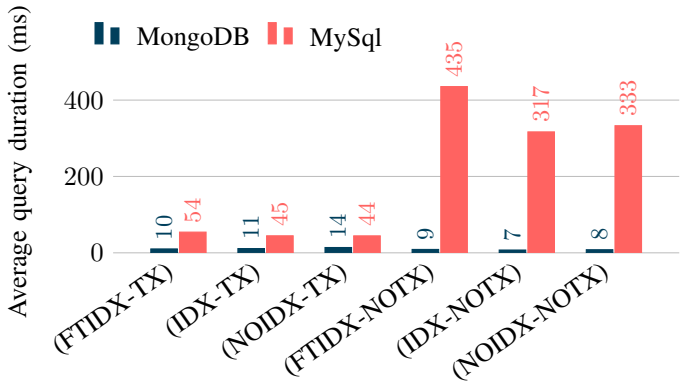

Fig. 5: Delete queries

case, where transactional queries are up to ten times faster then non-transactional queries.

Additional knowledge that comes from the results of updated scenarios in the MySql case shows that the data-structure usage with indexes outside of transactions can be a very expensive operation. The observed cost of index for a query of relational database is not negligible. For normal indexes it takes an additional 117 milliseconds and for full-text it adds 165 milliseconds compared with no-index queries.

V. Conclusion and Future Work

In the context of web applications, time for responding to a server from a request is usually composed from pieces such as business logic, database queries and web-server overheads. The delay between server request and response directly affects user-experience. The delay is a very often a used metric which is called Time-to-first-byte (TTFB).

For the web-user, there is nothing worse than long waiting for results of operations which makes the user stressed. In that case, the user can execute the operation multiple times, which can cause much longer waiting for results (in better case). In the worst case, multiple executions will cause unwanted changes in the system (e.g., two times executed bank request for sending money).

The paper focused on performance experiments which compare MySQL and MongoDB in many types of queries and situations such as inserting, updating and deleting data. Transactional and non-transactional variants brought interesting results, which show a performance drop of MySQL. The next knowledge gained is the cost of indexes for each situation.

The paper also focused on performance experiments which compare MySQL and MongoDB in many types of queries and situations such as inserting, updating and deleting data. Transactional and non-transactional variants brought interesting results. Every millisecond counts and it is possible to influence the outcome by correct indexing. The indexing helps in search queries, but decreases performance during the data modification phase. In the case of using MySQL, care needs to be taken with indexes usage. This needs to be kept in mind.

Future work will be focused on creating new types of benchmark which will include aspects such as geospatial searching or more complex indexing models. The benchmark gives a brief performance overview which will help to decide which database type is the most suitable for a given purpose. High performance and easy scalability of MongoDB implicates its usages for every business where performance matters. The JSON data-format is very popular because it is JavaScript native object representation. Usage of MongoDB and JavaScript creates the perfect couple without a data conversion barrier.

Future work will be focused on creating new types of benchmark which will include aspects such as geospatial searching or more complex indexing models.

Acknowledgment

The work has been supported by the Funds of University of Pardubice, Czech Republic. This support is very gratefully acknowledged.

References

[1] "Usage of content management systems," 2019. [Online]. Available: https://w3techs.com/technologies/overview/content_management
[2] solid IT gmbh, "Db-engines ranking - popularity ranking of database management systems," 2019. [Online]. Available: https://db-engines.com/en/ranking
[3] M. Diogo, B. Cabral, and J. Bernardino, "Consistency models of nosql databases," *Future Internet*, vol. 11, p. 43, 02 2019.
[4] Alotaibi and E. Pardede, "Transformation of schema from relational database (rdb) to nosql databases," *Data*, vol. 4, p. 148, 11 2019.
[5] L. Rocha, F. Vale, E. Cirilo, D. Barbosa, and F. Mourão, "A framework for migrating relational datasets to nosql1," *Procedia Computer Science*, vol. 51, pp. 2593–2602, 12 2015.
[6] A. Davoudian, L. Chen, and M. Liu, "A survey on nosql stores," *ACM Computing Surveys*, vol. 51, pp. 1–43, 04 2018.
[7] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*, 2nd ed. O'Reilly Media, Inc., 2015.
[8] R. Cattell, "Scalable sql and nosql data stores," *SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1978915.1978919
[9] A. Makris, K. Tserpes, V. Andronikou, and D. Anagnostopoulos, "A classification of nosql data stores based on key design characteristics," *Procedia Computer Science*, vol. 97, pp. 94–103, 12 2016.
[10] "Redis," 2019. [Online]. Available: https://redis.io/
[11] "memcached - a distributed memory object caching system," 2019. [Online]. Available: https://memcached.org/
[12] I. MongoDB, "Mongodb architecture guide: Overview," 08 2019. [Online]. Available: http://s3.amazonaws.com/info-mongodb-com/MongoDB_Architecture_Guide.pdf
[13] "The most popular database for modern apps," 2019. [Online]. Available: https://www.mongodb.com/
[14] "Apache couchdb," 2019. [Online]. Available: https://couchdb.apache.org/
[15] "Nosql database — ravendb acid nosql document database," 2019. [Online]. Available: https://ravendb.net/
[16] M. Lal, *Neo4j Graph Data Modeling*. Packt Publishing, 2015.
[17] "Neo4j graph database platform," 2019. [Online]. Available: https://neo4j.com/
[18] T. A. S. Foundation, "Apache cassandra," 2019. [Online]. Available: http://cassandra.apache.org/
[19] "Apache hbase," 2019. [Online]. Available: https://hbase.apache.org/
[20] solid IT gmbh, "Transactions – mongodb manual," 2019. [Online]. Available: https://docs.mongodb.com/manual/core/transactions/