A Performance Comparison of Document Oriented NoSQL Databases

Sundhara Kumar K.B*, Srividya*, Mohanavalli.S*
* Department of Information Technology,
SSN College of Engineering
Kalavakkam, Tamil Nadu, India

***Abstract*—Due to exponential growth of data over the years, there are lot of formats in which the data is available. Many schema-less databases are identified as the data these days do not pertain itself to a particular scheme. So, the effective storage and processing of such data were not possible with the existing RDBMS. The looming of NoSQL databases proved to be one of the best solutions for handling these kind of schema-less data. This work comprises about the various characteristics of NOSQL databases. The performance comparison of two widely used Document-oriented NOSQL databases viz MongoDB and CouchDB are analysed in this work.Both qualitative as well as quantitative features are taken and a comparison for streaming applications among those features are provided using the two databases under study.**

Keywords—RDBMS, NoSQL databases, Big Data, MongoDB, CouchDB.

I. INTRODUCTION

A document-oriented database is designed for storing, retrieving, and managing document-oriented, or semi structured data. Document-oriented databases are one of the main categories of NoSQL databases. The central concept of a document-oriented database is the notion of a Document. While each document-oriented database implementation differs on the details of this definition, in general, they all assume documents encapsulate and encode data (or information) in some standard format(s) (or encoding(s)). Encodings in use include XML, YAML, JSON and BSON, as well as binary forms like PDF and Microsoft Office documents (MS Word, Excel, and so on)

The hidden strength of DODBs is that they are a collection of key value collections. That is within a bucket similar to key value stores, there is an additional level of key value indexing that allows much more efficient queries. It is likely that if you have several big data projects and none call out for a specific specialty database type like graph, the DODB will be your go-to default.

When the factors such as scalability, fast look-ups fast prototyping, replication, easy maintenance, etc are considered, using these factors through Relational Database Systems - RDBMS[4] seems to get tougher. In order to overcome such factors along with the quickly emerging schema-less data, there was an emergent requirement of new database systems. Since there are many programmers already familiar with the SQL-type of queries, the new database that has to be formulated should not discard the SQL type. So, NOSQL databases[8]emerged that dealt with almost all the factors that went haywire in RDBMS at the same time incorporated the SQL-like queries. Thus, NOSQL abbreviated to Not Only SQL.

The RDBMS has been the most influential model in a database management system.In recent times, a new database model called NoSQL (Not only SQL) is proving to be a best alternative to the traditional database models. The goal of NoSQL is not to completely discard SQL but to work well for applications which do not apply the relational database model.

With the data growing exponentially in the modern era due to various data sources, the relational database model fail to apply itself to the applications that support concurrent users which spreads the load across a collection of application servers accompanied by load balancers. NoSQL database technologies have instead emerged to enable the cost-effective management of data behind new modern web and mobile applications. NoSQL databases can be used with applications that have: large transaction volumes, have the need for low-latency access to massive datasets, and have the need for nearly perfect service availability while operating in an unreliable environment. Companies using relational databases to achieve these goals, started building a large cluster of databases.

When adding more hardware failed companies tried to scale down existing relational solutions. Companies began simplifying the database schema, denormalizing the schema, relaxing durability and referential integrity, introducing query caching layers, separating read-only from write dedicated replicas, and data partitioning. These techniques helped with the existing relational databases, but none of the techniques addressed the core limitations that companies were trying to solve and added additional overhead and technical tradeoffs. The schema in the relational databases devolved from many interrelated fully expressed tables to simple key/value look-up.

Trying to improve scalability in relational databases meant adding bigger servers. Bigger servers are highly complex, proprietary, and disproportionately expensive. They are in contrast unlike the low-cost, commodity hardware in web and cloud-based architectures. At this point, companies had taken relational databases so far outside their intended use cases that they were unable to meet performance requirements. Companies began building in-house databases that were tailored to their particular workloads. These in-house custom solutions are the inspiration behind the NoSQL products that are currently seen on the market.

The most important question in evaluating the needs of an application and whether to use NoSQL databases or relational databases depends on the type of application being written, the type of queries that are expected, and the regularity vs. variability of the data's structure.

When the factors such as scalability, fast look-ups fast prototyping, replication, easy maintenance, etc are considered, using these factors through Relational Database Systems - RDBMS seems to get tougher. In order to overcome such factors along with the quickly emerging schema-less data, there was an emergent requirement of a new database systems. Since there are many programmers already familiar with the SQL-type of queries, the new database that has to be formulated should not discard the SQL type. So, NOSQL databases emerged that dealt with almost all the factors that went haywire in RDBMS at the same time incorporated the SQL-like queries. Thus, NOSQL abbreviated to Not Only SQL.

The standard ACID properties were a hindrance with scalability and hence by leaving these ACID properties, we could achieve better performance and scalability. Hence NoSQL uses a lighter set of properties called as BASE[7] - Basically Available: Nodes in a distributed environment can go down, but the whole system should not be affected; Soft State(scalable) : The state of the system and data change over time; Eventual Consistency: Given enough time, data will be consistent across the distributed system.

Considering the distributed transactions, the NoSQL databases adhere to CAP theorem [9]. CAP stands for Consistency - all the nodes across the distributed environment must read the same data, Availability - data should be available for read and write operations and Partition tolerant - system works well across physical network partitions . There are three levels of consistency - Strict consistency, tunable consistency and eventual consistency. At a given time, of these three properties, only two can be achieved[6],i.e, either consistency-availability, consistency-partition-tolerantor availability and partition tolerance.

There are numerous NoSQL databases that are available. They can be grouped to four major categories - Key-value store, Document-oriented store, Wide-column store and graph based database[1].

Key-Value store : The data are stored as sets of key-value pairs. All the keys are unique and each key is associated to one or a set of values. The values themselves can be a set of keys each of which referring to some other values. Example: Riak, Amazon S3 Dynamo,etc.

- Document-Oriented Store: The data in this database is stored as documents like XML or JSON. Each document however is associated with the key-value pairs. Data can be accessed using either the key or the value. Example: MongoDB, CouchDB, etc.
- Wide-Column Store: This database is very similar to the relational DBMS where the data are represented in rows and columns. New columns can be added as and when needed. A column family is required in order to access the data. Example: HBase, Cassandra, etc.
- Graph based Store: Whenever there is a possibility of representing the data as a graph, particularly visualizing the data as a graph in social network analysis, they can be stored using graph based store. Example: Neo4J, InfoGrid, etc.

The art of choosing right database for right application plays a very crucial role as it is the factor that constitutes the platform for analyzing the performance of the application that is under consideration. For instance an application may run only on a server or the application can have some mobile components or it can have some offline processing which can later be synced to the server. In such cases we need to carefully pick the databases. In other words, some applications might need consistency and availability. Some applications might need availability and partition tolerance and so on. This leads to a tricky situation to choose one database from the many options that needs a good domain knowledge.

There are various metrics that are to be considered for performing a comparative performance analysis. The metrics include both the qualitative metrics as well as quantitative metrics. Some of the commonly used qualitative metrics as described in [3] are Persistence, Replication , High Availability, Transactions, Rack-locality awareness, Implementation Language, Influences / sponsors , License type and the quantitative measures include size and performance measurements.

In this research work, we focus on streaming applications. For example, an application that reads a tweet from twitter instantaneously the moment it had been tweeted or re-tweeted, or an application that users sensors to log on the information every few seconds like recording a temperature reading or sensors used in healthcare monitoring systems. The use case chosen for this work is the twitter stream. We use NodeJS[13] a framework written in Javascript for handling the twitter data.

MongoDB and CouchDB are used as the two NoSQL databases which are taken for performance analysis. The metrics that we have taken into consideration are replication, MapReduce, CAP features, Storage type and sharding.

II. RELATED WORK

Tudorica, Bogdan George, and Cristian Bucur[3]compares various NoSQL systems. The NoSQL database focused to offer high performance and high availability. Although the SQL and the NoSQL databases are having some shared features, some of their behaviors are not similar in given instances. This suggests that these databases cannot be used interchangeable for solving any type of problem. Rather one shall choose between the two types of databases for a given instance.

Hecht, R., &Jablonski, S. [11]presents a survey on security issues in big data and NoSQL databases. Most of the NoSQL databases lack data encryption. To have a more secure database it is essential to encrypt sensitive fields in the database. Due to the high volume, variety and velocity of big data, traditional security models have difficulties in dealing with large scale of data.

Leavitt.N suggested that Big data is considered to be a data collection that has grown so large. Hence such a large scale of data cannot be effectively managed or exploited using conventional data management tools[10]. To handle this problem, specifically designed alternative database; such as - NoSQL and Search-based systems can be used. The author provided some advanced features of NoSQL databases and showcased how NoSQL databases can live upto their expectations when these new conditions are encountered.

Moniruzzaman, A. B. M., and Syed Akhter Hossain[2]presents - classification, characteristics and evaluation of NoSQL databases in Big Data Analytics. This paper provides an understanding of the pros and cons of various NoSQL database approaches; also provides a overview of the non-relational NoSQL databases.

Strauch, Christof, Ultra-Large Scale Sites, and Walter Kriha [5]provides an overview of the motives and rationales, common concepts, techniques and patterns as well as several classes of NoSQL databases (key-/value-stores, document databases, column-oriented databases).

There are lot of parameters taken into consideration that includes both the qualitative[19] and quantitative features. In this work we present the qualitative features alone and we propose a system for a streaming application that uses both these databases in order to compare the performances when they encounter various types of queries. This makes sense as we compare two document oriented NoSQL databases in the same environment.

III. DATABASE DESCRIPTION

In this section, we present an in-depth analysis of the two document-oriented NoSQL databases viz MongoDB and CouchDB.

A. MongoDB

MongoDB[12] is an open source document oriented nosql database which uses dynamic schemas for integration of certain types of application for making it easier and faster. MongoDB architecture is built upon collections and documents. Here documents are grouped into collections based on their structure and contains sets of key value pair as their basic unit. The database uses BSON, a binary representation of JSON like document. BSON is a document storage and data interchange format which supports the embedding of arrays and documents within other documents. BSON enables extension which allows representation of data types which are not part of JSON. BSON has three major characteristics namely lightweight, traversable and efficient.

The collections in MongoDB enables it to store a business subject in minimal number of document without having the necessity of breaking it up into multiple relational subjects. The ad hoc querying capability[15] of mongoDB enables it to support field, regular expression search and range queries. All mongo documents can be indexed and primary and secondary indices are also available.

MongoDB provides large collection of replica sets, and each replica set can contain two or more copies of data. The secondary replica set can be used in case of failure of the primary set. All read and write functions are performed on the primary set. In mongo db the scaling process is performed horizontally by using sharding. A shard key is chosen by the user based on which the data's in the collection will be distributed.

MongoDB can efficiently load balance by running multiple servers, balancing load and duplicating data for the purpose of keeping the system up even during hardware failure. MongoDB uses grid file system with expose function for file manipulation. Thegridfile system divides files into parts and stores each parts separately. The aggregation function is done by MapReduce which enables to obtain results similar to sql GROUP BY clause[16]. It also supports server side java script execution and fixed size collection called capped collection.

B. CouchDB

CouchDB[14]is an open source document oriented nosql database with a main focus on ease of use. It uses a concurrency oriented language called Erlang which is a general purpose garbage collected programming language whose sequential subset is almost a functional language with single assignment dynamic typing and eager evaluation. It uses java script as its query language and with mapreduce and HTTP for API[17]. In CouchDB data are not stored as

relationship in tables but as a collection of independent document. All document maintains their own data and schema. It is possible to merge any difference whichcouldhave occurred In database with the revision information contained in the meta data. CouchDB implements a multi-version concurrency control to avoid the need for locking the database during writes.

CouchDB stores json as documents with capacity to attach non-json files with those document. The data are written and queried using REST API[18] and allows to retrieve data based on document content by using system of views. The servers of couch db operates as nodes with each containing copy of data which needs to be stored. Applicationserver can read and write to any couchDB server. Couch db provides ACID semantics by implementation of MVCC. Also couch db has the ability to replicate devices to go offline and can sync it back when the device is back online. It has a distributed architecture with the support of bidirectional synchronization. The documents stored have field value pair with things like strings, numbers etc also ordered list and associative arrays can be used. There is no requirement of schema in couch db since each database has a unique id.

The problem that we focus in this research work is to compare the NoSQL databases specifically the two most widely used document oriented databases MongoDB and CouchDB. The objective of this work is not to write off any other databases but to provide a clear understanding of when to choose which databases whenever there is a conflict of interest in choosing between these two databases. With the type of application - a streaming application, there is a high possibility of dynamic and ad-hoc queries to arise often. As the needs is known only on the run, it is not efficient to use a set of pre-defined queries for these kind of applications.

IV. PROPOSED WORK

In this section, we work on comparing MongoDB and CouchDB - the two databases taken under experimentation. These two are widely used document oriented NoSQL databases. The reason for considering these two databases are to understand for which application, which database suits well. Streaming applications (twitter stream) is taken as a reference application which involves NodeJS.

NodeJS is an open source, cross-platform runtime environment for developing server-side web-based applications. An application is developed using NodeJS which constantly takes the tweet from twitter based on a key-word filter specified in the application. The resultant JSON file from the application are loaded into MongoDB and CouchDB. Applications such as sentimental analysis can be done using these techniques. In order to calculate score or classify positive and negative tweets, the tweets may have to be logged. Those are logged on to MongoDB and couchDB to populate the records.

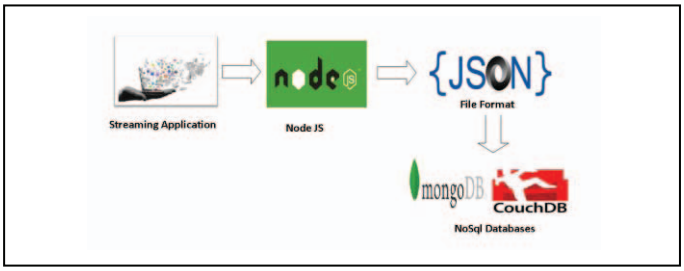

Fig 1: Proposed Framework

The System Architecture in Figure 1 shows that streaming data is accessed through the NodeJS platform. Once the tweets are captured, they appear in the form of JSON documents. These JSON documents are stored in MongoDB and CouchDB to carry out the performance comparison.

The framework above suggests that the twitter stream data is accessed through the NodeJS platform. Once the tweets are captured, they appear in the form of JSON documents. These Json documents are stored in MongoDB and CouchDB to carry out the performance comparison.

V. RESULTS AND DISCUSSIONS

We have selected few qualitative features for listing out the comparisons between the two databases. Few of the features taken are :

- Replication - a replica set is a group of instances that hosts the same dataset.
- Storage type - The format of document which the database can hold
- CAP features - as per CAP theorem, at a time a database can provide only two of the three features.
- MapReduce - a set of functions that make parallel processing easier.

These features are then analyzed based on both the databases and a comparative table is shown in Table1 as per the work carried out in [19].

TABLE 1: FEATURE COMPARISON

FEATURES	MONGODB	COUCHDB
Developmental Language	C++, JavaScript	Erlang
CAP Features.	Consistence and Partition Tolerance (CP)	Availability and Partition Tolerance (AP)
Replication	Master-Slave.	Master-Master (Multi Master)
Map Reduce	Supports Map Reduce on sharded collections, to be written in Java Script.	Supports Map Reduce for queries - supports HTTP and REST API.
Storage Type	BSON (Binary JavaScript Object Notation)	JSON (JavaScript Object Notation)

Sharding	Supports Sharding.	Supports Sharding.
License	GNU AGPL v3.0, MongoDB	Apache license 2.0

The features pertaining to both the database under study are taken and for any streaming applications how they behave is discussed. The qualitative attributes are taken into consideration and an analysis is performed as shown in Table1. When a user arrives at an ambiguous situation as to which database to use for a streaming application, these feature comparison would easily allow them to decide to select the database.

Further in this work, acomparison between the time for insertion, deletion, updation and retrieval are done. The following graphs show a comparative analysis between the two databases.

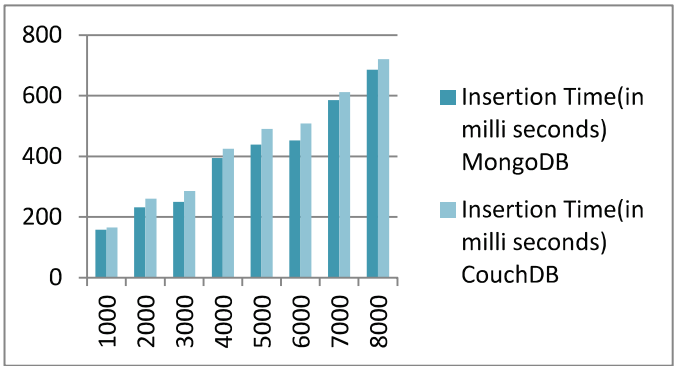

Fig 2: Insertion Time

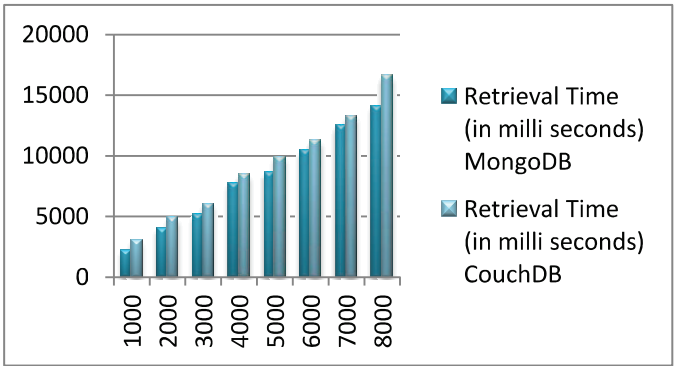

Fig 3: Retrieval Time

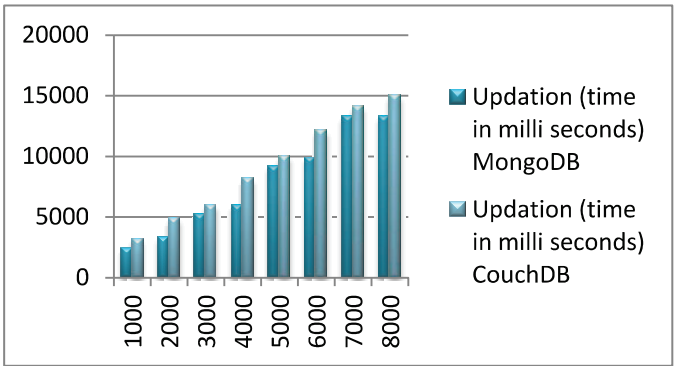

Fig 4: Updation Time

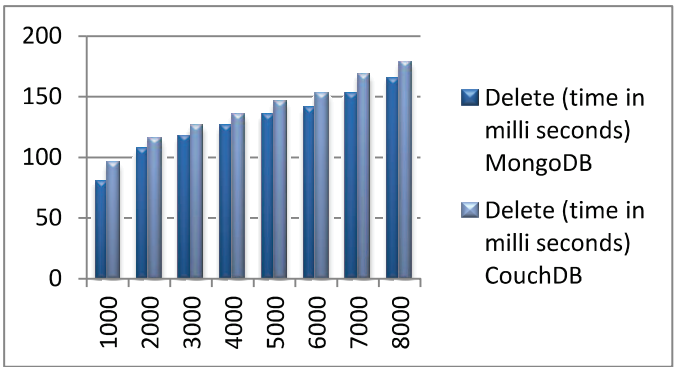

Fig 5: Deletion Time

The above graphs from Figure 2 to 5 shows a comparison between the two databases under study. As discussed, we have taken a streaming application and performed this comparison. Hence from the graphs, we can conclude that MongoDB performs well for these kind of applications.

VI. CONCLUSION AND FUTURE WORK

The qualitative features of both the document-store databases are analyzed in this work. Since SQL and NoSQL databases have very few common aspects within them, it was very tough to provide a comparative analysis of SQL and NoSQL. Hence comparisons within NoSQL databases are performed. Further, quantitative attributes like size of the data stored in both the databases and how the databases perform when various types of queries encountered are analyzed. The results suggests that for streaming applications, MongoDB clearly have an advantage over CouchDB. This is very much evident from the graphs shown in the above section. Since CouchDB doesn't have any options for bulk-importing JSON documents, it proves a failure model for these kind of streaming applications.

Whenever there is a scenario of using a streaming application like the case of using the sensors or streaming data from the social media apps, there may be an option of modeling the data in graph format rather than choosing JSON format. Hence, with the advent of OrientDB may be a boon as

the database is a blend of relational, non-relational and graph database. Further study on OrientDB and analysing its performance when it encounters various types of queries can be done in the future.

REFERENCES

[1] Indrawan-Santiago, M., "Database Research: Are We at a Crossroad? Reflection on NoSQL," Network-Based Information Systems (NBiS), 2012 15th International Conference on , vol., no., pp.45,51, 26-28 Sept. 2012. doi:10.1109/NBiS.2012.95

[2] Moniruzzaman, A. B. M., and Syed Akhter Hossain. "Nosql database: New era of databases for big data analytics-classification, characteristics and comparison." *arXiv preprint arXiv:1307.0191* (2013).

[3] Tudorica, Bogdan George, and Cristian Bucur. "A comparison between several NoSQL databases with comments and notes." *Roedunet International Conference (RoEduNet), 2011 10th*. IEEE, 2011.

[4] Date, C. J. *An introduction to database systems*. Addison-Wesley publ., 1975.

[5] Strauch, Christof, Ultra-Large Scale Sites, and Walter Kriha. "NoSQL databases." *Lecture Notes, Stuttgart Media University* (2011).

[6] Abramova, Veronika, and Jorge Bernardino. "NoSQL databases: MongoDB vs cassandra." *Proceedings of the International Conference on Computer Science and Software Engineering*. ACM, 2013.

[7] Cook, John D., "ACID versus BASE for database transactions", http://www.johndcook.com/blog/2009/07/06/brewer-cap-theorem-base/.

[8] http://www.slideshare.net/adorepump/voldemort-nosql

[9] Konstantinou, I., Angelou, E., Boumpouka, C., Tsoumakos, D., & Koziris,N.(2011,October). On the elasticity of nosql databases over cloud management platforms.In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 2385-2388). ACM.

[10] Leavitt, N. (2010). Will NoSQL databases live up to their promise?. *Computer*,*43*(2), 12-14.

[11] Hecht, R., & Jablonski, S. (2011, December). NoSQL evaluation: A use case oriented survey.In *Cloud and Service Computing (CSC), 2011 International Conference on* (pp. 336-341).IEEE.

[12] https://www.mongodb.org/

[13] https://nodejs.org/en/

[14] http://couchdb.apache.org/

[15] Kaur, Kanwalpreet, and Rinkle Rani. "Modeling and querying data in NoSQL databases." *Big Data, 2013 IEEE International Conference on*. IEEE, 2013.

[16] Nayak, Ameya, Anil Poriya, and Dikshay Poojary. "Type of NOSQL databases and its comparison with relational databases." *International Journal of Applied Information Systems* 5.4 (2013): 16-19.

[17] Anderson, J. Chris, Jan Lehnardt, and Noah Slater. *CouchDB: the definitive guide*. " O'Reilly Media, Inc.", 2010.

[18] Li, Li, and Wu Chou. "Design and describe REST API without violating REST: A Petri net based approach." *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011.

[19] Sundhara Kumar K B, Senthil Kumar V, Srividya, Mohanavalli.S, "Comparison of NoSQL Databases", Proceedings of National Conference on Communication and Informatics - 2016, Sri Venkateswara College of Engineering, Sriperumbudur.