Survey Study on Issues in MongoDB in Cloud Environment

Mrugesh P Patel[#1], Mosin I. Hasan[#2],Hemant D. Vasava[#3]
[#]*Department Of Computer Engineering,*
Birla Vishvakarma Mahavidyalaya,
VallabhVidyanagar, India
[1]mrugesh56739@gmail.com
[2]mihasan@bvmengineering.ac.in
[3]hdvasava@bvmengineering.ac.in

Abstract-Now a day, Web applications are growing at a staggering rate every day. As web applications keep getting more complex, their data storage requirements tend to grow exponentially. Traditional relational database use two-dimensional table to represent data, with strict consistency of database transactions. However, for a wide range of massive data queries, multi-table query is not effective. MongoDB is a document store database that does not have strict schemas that RDBMs require and can grow horizontally without performance degradation. MongoDB brings possibilities for different storage scenarios and allow the programmers to use the database as a storage that fits their needs, not the other way around. In this paper, we have explore the issues in MongoDB namely sharding, indexes, etc. in cloud environment.

Keywords: cloud computing, cloud data management, SQL, No-SQL, MongoDB, issues.

I. INTRODUCTION

Web applications are growing at a staggering rate every day. As web applications keep getting more complex, their data storage requirements tend to grow exponentially. Information Technology (IT) department of any organization is responsible for providing reliable computing, storage, backup and network facilities at the lowest feasible cost. Cloud computing [1] becomes a natural and ideal choice for such organizations and customers. Cloud computing takes benefit of many technologies such as server consolidation, huge and faster storage, grid computing virtualization, N-tier architecture and robust networks. It provides IT-related services such as Software-as-a-Service, Development Platforms-as-a-Service and Infrastructure-as a-Service over the network on-demand anytime from anywhere on the basis of "pay-as-you-go" model. Elasticity, scalability, high availability, price per-usage and multi-tenancy are the main features of Cloud computing. It reduces the cost of using expensive resources at the provider's end due to economies of scale. Quick provisioning and immediate deployment of latest applications at lesser cost are the benefits which force people to adopt Cloud computing.

Cloud computing has brought a paradigm shift not in the technology landscape, but also in the database landscape. With more usage of Cloud computing, demand for provisioning of database services has raised. Provisioning of Cloud databases is known as Database-as-a-Service in Cloud terminology. The main objective of the paper is to explore the trends in Cloud databases and analyze the potential challenges in MongoDB databases. The structure of paper has been divided into four sections. Second section describes Cloud databases. Third provides an overview and architecture of MongoDB databases. Section 4 discusses major issues in these databases.

II. CLOUD DATABASE

Massive growth in digital data, changing data storage requirements, better broadband facilities and Cloud computing led to the emergence of cloud databases [2]. Cloud Storage, Data as a service (DaaS) and Database as a service (DBaaS) are the different terms used for data management in the Cloud. They differ on the basis of how data is stored and managed. Dropbox, iCloud etc. are popular cloud storage services [3]. DaaS allows user to store data at a remote disk available through Internet. It is used mainly for backup purposes and basic data management. DBaaS is one step ahead. It offers complete database functionality and allows users to access and store their database at remote disks anytime from any place through Internet. Amazon's SimpleDB, Amazon RDS, Google's BigTable, Yahoo's Sherpa and Microsoft's SQL Azure Database are the commonly used databases in the Cloud [3].

Different ways of storing data have been a big subject of discussion since the early databases.one of

the most popular ways to store information is inside a flat file, containing delimiters that separate fields of data. An example that is still being in use today is the CSV (Comma Separated Values) file, where each field's value is stored inside the file with a comma (delimiter) separating the values. A disadvantage of such a database structure based on a flat file is that in order to find a record a system would have to iterate through the entire file in order to find the correct record. With any file that has thousands of records and multiple columns it may take a while to go through each row and column to find the information needed. Another big disadvantage of the flat file architecture is having records that require different number of columns.

A) Relational Databases

A solution for this flat file databases started with the relational databases where the schema is designed and managed through a RDBMS (Relational Database Management System). By having the schema designed in a language such as SQL (Structured Query Language) the underlying architecture can be designed and structured in different ways, therefore allowing the same language to be used to communicate with different databases. One of the best optimizations that all RDBMS provide are indexes, which greatly speed up a database performance so that the system no longer has to look at every column/row in order to figure out where the record exists, instead it would be able to look it up quickly. Relational Databases are able to go one step further and alleviate duplicate data by setting a relationship between two tables, which allows duplicate information to be stored separately in another table and then referenced inside the current table.

B) Document-Oriented(NoSQL) databases

NoSQL means 'Not Only SQL' or 'Not Relational'. A NoSQL database is defined as a non-relational, shared-nothing, horizontally scalable database without ACID guarantees. NoSQL implementations are classified further into key/value stores, document stores, object stores, tuple stores, column stores and graph stores. They can store and retrieve unstructured, semi-structured and structured data. They are item-oriented. A domain can be compared to a table and contains items having different schemas. The items are identified by keys. All data relevant to a particular item is stored within that item. It improves scalability of these databases as complex joins are not required to regroup data from multiple tables. They have the ability to replicate and distribute data over many servers. They are dynamically provisioned on demand. They have emerged to address the requirements of data management in the cloud as they follow BASE (Basically Available, Soft state, eventually consistent) in contrast to the ACID guarantees. So, they are not suitable for update intensive transaction applications. They provide high availability at the cost of consistency [5].

There are several advantages and disadvantages for both SQL and NoSQL databases. Developers often wonder which database is better and which they should use on their project. There is no easy or straightforward answer for this question and there is not one database that will work for every project. MongoDB has both strengths [6] and weaknesses [7], but overall it performs quite well and it doesn't have as many restrictions and limitations as other NoSQL databases [8][9].

III. MONGODB

MongoDB [10] is an open-source project supported by 10gen, Inc. The lOgen cloud platform can be used to create a private cloud. MongoDB is an open source, document-oriented database designed with both scalability and developer agility in mind. Instead of storing your data in tables and rows as you would with a relational database. MongoDB is easy to store data with object types, including documents-insert objects and number group, and suit for a large number of data to insert real-timely, update and query ,and have the ability to copy and high scalability the application program needed to do real-time data storage. In MongoDB, you store JSON-like documents with dynamic schemas. The goal of MongoDB is to bridge the gap between key-value stores (which are fast and scalable) and relational databases (which have rich functionality). MongoDB maintains many of the great features of a relational database - like indexes and dynamic queries. But by changing the data model from relational to document-oriented, you gain many advantages, including greater agility through flexible schemas and easier horizontal scalability. The Fig. 1 shows the model of MongoDB database [11].

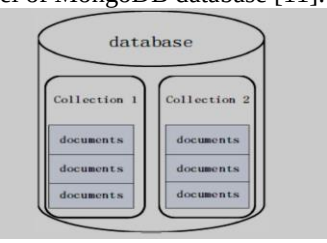

Figure 1. MongoDB document -oriented database

A) Document-Oriented Storage

Mongo uses BSON, which is the binary-encoded serialization of JSON format to query and store data [12]. The developers chose BSON instead of JSON because of its efficiency – binary strings can be smaller when encoded, and support of additional data types. JSON currently only supports the following data

types: string, number, Boolean, array and object. BSON supports: string, int, double, Boolean, date, bytearray, object, array and others. BSON's only restriction is that data must be serialized in little-endian format. Also, since BSON is the format that the data is sent/retrieved and stored, there is little need to decode it to text. The most important benefit of a document is that it may contain other documents embedded inside.
Below is an example of a document with two embedded documents:

```
{
        "name" : "John Smith",
        "phone" : {
                        "home" : "555-123-4567",
                        "cell" : "555-321-7654"
                  },
        "address" : {
                "street" :"123 Main Street",
                "city" : "San Francisco",
                "state" : "CA",
                "zip" : 94115
                }
}
```

B) Index Support

When tables grow in size, indexes will dramatically speed up queries. Indexes in MongoDB work almost identically with the relational databases. MongoDB indexes are implemented as B-Tree just like the default in MySQL. Indexes can be added to any key including documents and their keys as shown below:
Adding a simple index:
db.users.ensureIndex({ "name" : 1 })
Adding an index to embedded keys:
db.users.ensureIndex({ "address.city" : 1 })
Adding an index to an entire document:
db.users.ensureIndex({ "address" : 1 })

C) Replication

Replication is a concept that duplicates data over a group of servers known as replica sets. In a replica set there is a primary server and one or more secondary servers. With this setup the primary acts as a master, where all writes will go. Secondaries can be used for reading as long as the developer is aware that data will eventually get there. Secondaries are great places to run backups, since it will not increase the load on the primary node. MongoDB can be set up with automatic failover so if the primary node out of the two (or more) nodes goes down, one of the secondaries will step up and become primary in order for writes to continue [13].

D) Auto-sharding[14]

A MongoDB shard cluster consists of two or more shards, one or more config servers, and any number of routing process.
Each of the components is described below:

1. Shard: Each shard consists of one or more servers and stores data using mongod processes. In production situation, each shard will consist of a replica set to ensure availability and automated failover.
2. Config server: It stores the cluster's metadata which includes basic information on each shard server and the chunks contained therein.
3. Mongos (Routing Processes): It can be thought of as a routing and coordination process. When receiving requests from client, the mongos routes the request to the appropriate server and merges any results to be sent back to the client.

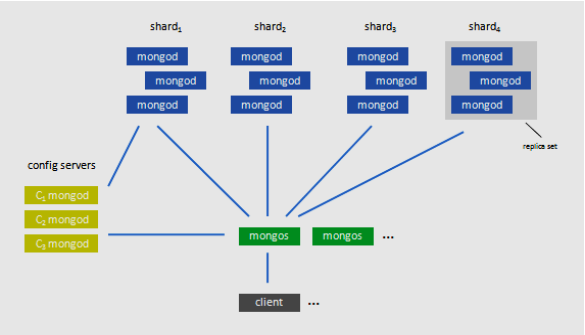

Figure 2.Architecture of Auto-Sharding [15]

Sharding refers to the process of splitting data up and storing different portions of the data on different machines. By splitting data up across machines, it becomes possible to store more data and handle more loads without requiring large or powerful machines. MongoDB supports Auto-Sharding, which eliminates some of the manual sharding; the cluster can split up data and rebalance automatically. MongoDB sharding provides: (1) automatic balancing for changes in load and data distribution, (2) easy addition of new machines without down time, (3) no single points of failure, and (4) Automatic failover. The basic concept behind MongoDB's sharding is to break up collections into smaller chunks. These chunks can be distributed across shards so that each shard is responsible for a subset of the total data set. To partition a collection, MongoDB specify a shard key pattern which names one or more fields to define the key upon which we distribute data. Because of shard key, chunks can be described as a triple of collection, minkey and maxkey. Chunks grow to a maximum size, usually 200MB, once a chunk has reached that approximate size, the chunk splits into two new chunks.

IV. ISSUES IN MONGODB

In MongoDB, there are some issues like [16]:

- If system or network crashes while it's updating 'table-contents' – you lose all you data. Repair takes a lot of time, but usually ends up in 50-90% data loss if you aren't lucky. So only way to be fully secure is to have 2 replicas in different data centers.
- Indexes take up a lot of RAM. They are B-tree indexes and if you have many, you can run out of system resources really fast.so require fast resource process.
- Data size in MongoDB is typically higher e.g. each document has field names stored it.
- Less flexibility with more complex querying because there is no join operation. (No-join).
- No support for transactions – certain atomic operations is supported, at a single document level.
- At the moment Map/Reduce (e.g. to do aggregations/data analysis) is OK, but not blisteringly fast. So if that's required, something like Hadoop may need to be added into the mix.
- Balancing strategies of auto-sharding process is not terribly intelligent. The goal of the balancer is not only to keep the data evenly distributed but also to minimize the amount of data transferred. It moves chunks based on the overall size of the shard [17]. So it is necessary to improve the balancing strategy of Auto-Sharding in MongoDB.

CONCLUSION

Massive data generated by web-based applications have changed the whole database scenario. Cloud databases appear to be a good solution for handling such data. MongoDB performance can vary significantly between cloud providers due to many different factors like ability to cache all the indexes in RAM and fast I/O access. The database does not require high computing performance, so it can handle multiple threads without being overloaded. But there are some issues related to its performance like balancing strategies of sharding process is not quite intelligent, map-reduce process is not fast, etc.

REFERENCES

[1] Indu Arora; Dr. Anu Gupta al," Cloud Databases: A Paradigm Shift in Databases", IJCSI- International Journal of Computer Science Issues, Vol. 9, Issue 4, No 3, July 2012.

[2]Rajkumar Buyya et al., "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, Vol. 25, Issue 6, June 2009, pp. 599-616.

[3] Jiyi Wu et al, "Recent Advances in Cloud Storage", in Third International Symposium on Computer Science and Computational Technology(ISCSCT '10), Jiaozuo, P. R. China, 14-15,August 2010, pp. 151-154.

[4] Daniel J. Abadi et al., "Column-oriented Database Systems", VLDB '09.

[5] Arpita Mathur et al., "Cloud Based Distributed Databases: The Future Ahead", International Journal on Computer Science and Engineering (IJCSE) Vol. 3, No. 6, 2011.

[6] B. Wong, "Choosing a non-relational database; why we migrated from MySQL to MongoDB," Online, 2010. [online]. Available:http://benjaminwss.posterous.com/choosing-a-non-relational-database-why-we-mig/.

[7] E. Gunderson, "Two reasons you shouldn't use MongoDB," Online, 2010. [Online]. Available: http://ethangunderson.com/blog/two-reasons-to-not-use-mongodb/.

[8] N. Leavitt, "Will NoSQL Databases Live Up to Their Promise?," Computer, vol. 43, no.2, pp. 12–14, Feb. 2010.

[9] D. Mytton, "Choosing a non-relational database; why we migrated from MySQL to MongoDB Folllow Up.,"Online, 2010.

[10] Tudorica,B.G. "A comparison between several NoSQL databases with comments and notes", In the 10th Roedunet International conference(RoEduNet), pp. 1-3, Jun 2011.

[11] Zhu Wei-ping;Li Ming-xin. "Using MongoDB to Implement Textbook Management System instead of MySQL", In Proceedings of the 3rd International Conference on Communication Software and Networks(ICCSN), pp. 303-305, May 2011.

[12] C. Creative, "Binary JSON", 2012. [Online]:http://www.bsonspec.org

[13] P. Eelco ; M. Peter ;T. Hawkins, "The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing", vol. 44, no. 5. Apress, 2010, p. 328.

[14] Yimeng Liu;Yizhi Wang;Yi Jin," Research on The Improvement of MongoDB Auto-Sharding in Cloud Environment", The 7th International Conference on Computer Science & Education (ICCSE),Thp6.11,2012.

[15]http://technoturd.wordpress.com/2013/01/14/afternoon-with-ceo-of-10gen-mongodb-company last accessed on September 14, 2013.

[16]http://blog.iprofs.nl/2011/11/25/is mongodb-a-good-alternative-to-rdbms-databases-like-oracle-and-mysql last accessed on December 25, 2013.

[17]Kristina Chodorow, "Scaling MongoDB". O Reilly Media, January2011. p13.

