International Journal of Advance Engineering and Research Development (IJAERD)
Volume 1,Issue 5,May 2014, e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

Performance improvement of sharding in MongoDB using k-mean clustering algorithm

Mrugesh P Patel[1],Mosin I. Hasan[2],Hemant D. Vasava[3]

[1,2,3]*Department Of Computer Engineering,Birla VishvakarmaMahavidyalaya,VallabhVidyanagar, India*
[1]*mrugesh56739@gmail.com,* [2]*mihasan@bvmengineering.ac.in,* [3]*hdvasava@bvmengineering.ac.in*

Abstract-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.No-SQL (MongoDB) database which breaks theshackles of RDBMS is becoming the focus of attention.In thispaper, firstly represented the architecture and implementation process of Auto-Sharding process in MongoDB database, then animprovedalgorithm based on K-Mean clustering operation isproposed in order to solve the problem of uneven distribution ofdata in auto-sharding.so using this balancing strategy we caneffectively balance the data among shards and improve thecluster's concurrent reading and writing performance.

Keywords: MongoDB, issues, Auto-sharding, K-Mean clustering algorithm, balancing strategy

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. MongoDB [1] 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[2].]. 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 [3].

Fig. 1 MongoDB database

In MongoDB, there are some issues like [4]: (a) 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. (b) 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.(c) Data size in MongoDB is typically higher e.g. each document has field names stored it.(d)Less flexibility with more complex querying because there is no join operation. (No-join). (e) No support for transactions – certain atomic operations is supported, at a single document level. (f) 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. (g) 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 [5]. So it is necessary to improve the balancing strategy of Auto-Sharding in MongoDB.

1. AUTO-SHARDING IN MONGODB

Sharding [6] refers to the process of splitting data up andstoring different portions of the data on different machines. Bysplitting data up across machines, it becomes possible to storemore data and handle more loads without requiring large orpowerful machines. MongoDB supports Auto-Sharding,which eliminates some of the manual sharding; the cluster cansplit up data and rebalance automatically. MongoDB shardingprovides: (1) automatic balancing for changes in load and datadistribution, (2) easy addition of new machines without downtime, (3) no single points of failure and (4) Automatic failover.

1.1. Auto-sharding architecture

A MongoDB shard cluster consists of two or more shards, one or more config servers, and any number of routing process.

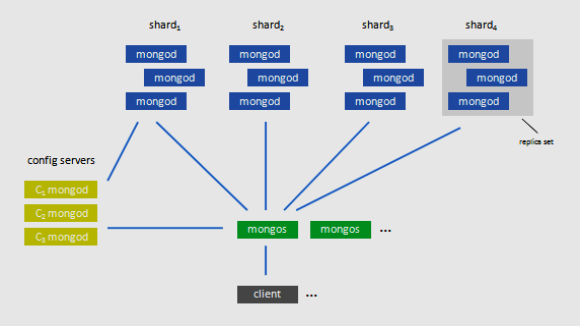


Figure 2.Architecture of Auto-Sharding [7]

Each of the components is described below [5]:

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.

1.2. Balancing strategies of auto-sharding [5]

MongoDB uses balancer to keep chunks evenly across allservers of the cluster. The unit of transfer is a chunk, andbalancer waits for a threshold of uneven chunks counts tooccur. In the field,

having a difference of 8 chunks between theleast and most loaded shards showed to be a good heuristic.Once the threshold is reached, balancer will redistributechunks, until that difference in chunks between any two shardsis down to 2 chunks. In order to reduce the amount of data transferred, eachshard contains multiple ranges. When a new shard added intothe cluster or some of the shards contain too much data to reachthe threshold, the balancer will skim chunks off of the top ofthe most-populous shard and move these chunks to the least populousshard, allowing the data evenly distributed in thecluster by moving the bare minimum.

The goal of the balancer is not only to keep the data evenlydistributed but also to minimize the amount of data transferred.The balancer's algorithm isn't terribly intelligent. It moveschunks based on the overall size of the shard [7]. The migrationchunks are just the ones located on the top of each shard, butthe operation of data is not taking into consideration. The datatransferred between shards may not often be used, it will makesthe system load cannot achieve an effective balance. It isnecessary to improve the balancing strategy of Auto-Shardingin MongoDB.

2. ALGORITHM BASED ON K-MEAN CLUSTREING OPERATION

In web application, there are huge amount of database. Normally 100 to 150 queries are fire every time in this entire database. All this queries are saving into pattern which is making cluster of table in database. This cluster gives information of the frequency of used tables.so using this cluster we can find most usable table in database and this table put in the particular shard of MongoDB's auto-sharding process. Using this algorithm we can solve uneven distribution of data problem in MongoDB.

2.1. K-Mean clustering algorithm[8]

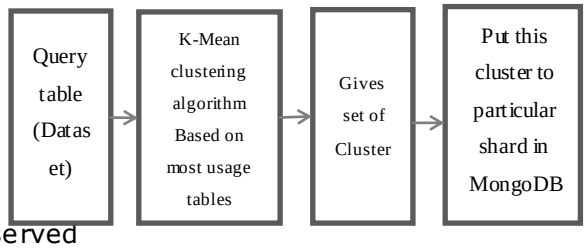
K-means clusteringis a data mining/machine learning algorithm used to cluster observations into groups of related observations without any prior knowledge of those relationships. The k-means algorithm is one of the simplest clustering techniques and it is commonly used in medical imaging, biometrics and related fields.

The k-means algorithm is an evolutionary algorithm that gains its name from its method of operation. The algorithm clusters observations into k groups, where k is provided as an input parameter. It then assigns each observation to clusters based upon the observation's proximity to the mean of the cluster. The cluster's mean is then recomputed and the process begins again. Here's how the algorithm works:

1. The algorithm arbitrarily selects k points as the initial cluster centers ("means").
2. Each point in the dataset is assigned to the closed cluster, based upon the Euclidean distance between each point and each cluster center.
3. Each cluster center is recomputed as the average of the points in that cluster.
4. Steps 2 and 3 repeat until the clusters converge. Convergence may be defined differently depending upon the implementation, but it normally means that either no observations change clusters when steps 2 and 3 are repeated or that the changes do not make a material difference in the definition of the clusters.

2.2. Proposed approach

Using K-Mean clustering algorithm we can make different cluster base on most usage tables in database.so using these algorithm we can find most usage table and make cluster of that database.



Taking one example, take large databases: disaster management .Now using the clustering algorithm making different cluster of that database like:
Q1:disaster_master, volunteer_team, volunteer
Q2:disaster_master, lost_person, rescue_person
Q3: volunteer, state, area, disaster_type
So using clustering algorithm, we can make different clusters like q1, q2 and q3.
Now using this cluster we can get most use table in database. Now put this most used table to particular shard in MongoDB's auto-sharding process using shardtag and addshardrange tag command.so finally check concurrent reading and writing performance of original algorithm and k-mean clustering algorithm.

3. PERFORMANCE EVALUATION

The test environment based on the MongoDB Auto-Sharding cluster, it contains 10 virtual machines, each of which has the same hardware configuration: 4GB of memory and CPU speed is 2.4G HZ. Each virtual machine has a Window 7 operating system, and be connected by the 100Mbps LAN. The MongoDB version is 2.6.1, and the Auto-Sharding cluster has three shards, each of which consists of a replica set. Each replica set contains three nodes: one primary node and two secondary nodes. We should run the rs.slaveOk () command on the secondary nodes in order to query data from them.

Now firstly create large dataset based on our requirement than apply k-mean clustering algorithm to make different cluster based on most usable table in database.now using this cluster we can find most usage table so put those all tables to particular shard in sharding process using shardtag and addshardrange command.We realize the clustering algorithm based on k-mean clustering in MongoDB, andcompare the two algorithms by testing the concurrent read andwrite performance of the cluster. In order to see the effect of the k-mean clustering algorithm, we onlyadd two shards, insert 12,000 records and use randomlygenerated id to do find and update operations. Then we add thethird shard to the cluster. The purpose of this action is to ensurethe cluster has data at the beginning of the test.

Firstly, we test the concurrent writing performance of the cluster. We insert 12,000 records into the cluster and remain the overall amount of records be same under different number of concurrent. The concurrent writing performance of this two algorithm is shown in the Fig. 3.Remain the concurrent number and records are unchanged and test the concurrent reading performance of the cluster. Theconcurrent reading performance of these two algorithms isshown in the Fig. 4.

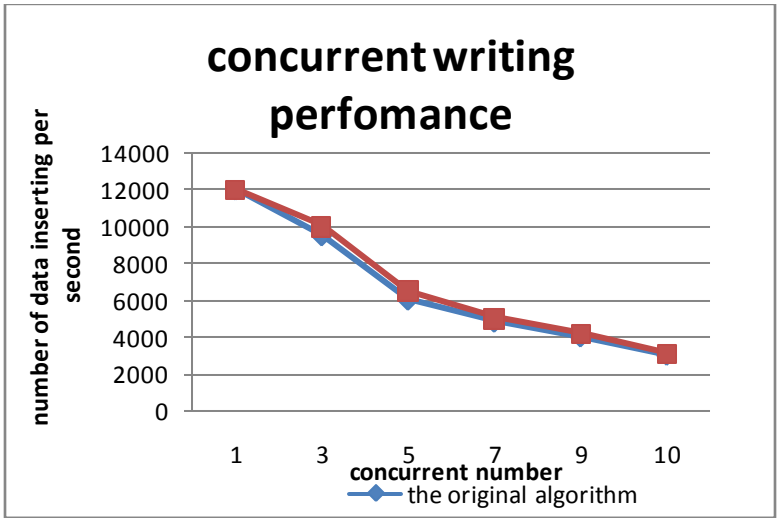


Figure 3. Concurrent writing performance

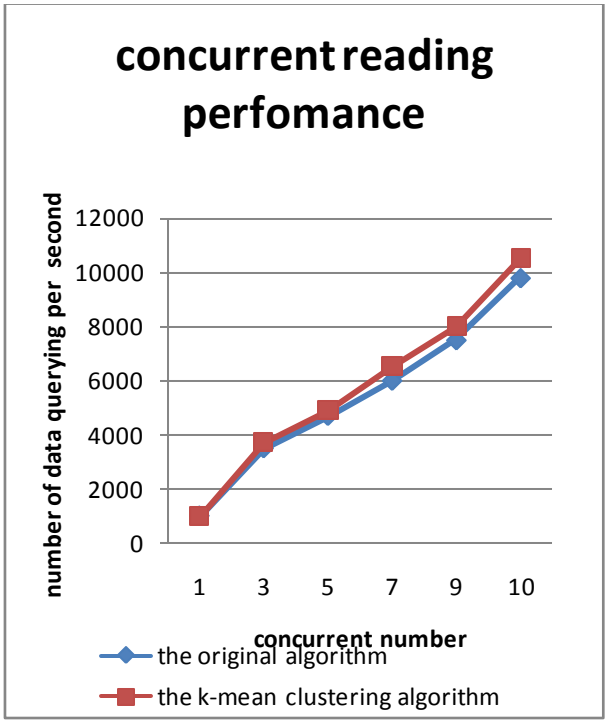


Figure 4. Concurrent reading performance

II. CONCLUSION

In this paper first explain architecture and principle of MongoDB's auto-sharding process. There is problem of uneven distribution of data among the shards; we introduce an improved balancing algorithm based on K-mean clustering algorithm. A data balancing strategy based on this k-mean clustering algorithm proposed and its gives concurrent writing and reading performance of auto-sharding significantly improved verified by experiment. For future work, we can also use k-mean clustering algorithm is based on column fragmentation on sharding process and improve data balancing strategy.

REFERENCES

[1]Tudorica,B.G. "A comparison between several NoSQL databases with comments and notes", In the 10lh Roedunet International conference(RoEduNet), pp. 1-3, Jun 2011.
[2]xiaopingDai;Qiyixue. "An approach to build highy_Performance OPC-XML DA server system based on MongoDB and Comet", In Proceedings of International Conference on Electrical and Control Engineering (ICECE), Sep 2011, pp. 2881-2884.
[3]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.
[4] 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.
[5] Kristina Chodorow, "Scaling MongoDB". O Reilly Media, January2011. p13.
[6]Kristina Chodorow; Michael Dirolf."MongoDB: The Definitive Guide".O'Reilly Media, September 2010. p135.
[7]http://technoturd.wordpress.com/2013/01/14/afternoon-with-ceo-of-10gen-mongodb-company last accessed on September 14, 2013.
[8]http://databases.about.com/od/datamining/a/kmeans.htm last accessed on January 2, 2014.

