International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 8– No.4, February 2015 – www.ijais.org

Improvement in Load Balancing Technique for MongoDB Clusters

Harpinder Kaur
Computer Science and Engineering,
Lovely Professional University, India

Janpreet Singh
Computer Science and Engineering,
Lovely Professional University, India

ABSTRACT

Now is the era of cloud computing and related buzzwords are the virtualization, resource sharing, Big Data. With the advent of new technologies, gadgets or simply IOT has enabled the advanced connectivity of devices, systems, and services and with this the data is being produced at an enormous rates from these devices be it form sensors, GPS data, log files from different sources etc. which is mostly unstructured data. With the acquaintance with NoSQL technology MongoDB is extensively used to handle all types of data because of its various advantages as its auto-load balancing technique in which the primary node's read load is decreased by distributing load to the secondary nodes, other feature of MongoDB is its auto sharding technique which works by reducing the load over a node by splitting up data in chunks and migrating it over to other nodes. The present work endeavors to study the role of MongoDB's auto balancing technique. In present work MongoDB balancer is introduced to and the performance of balancer of MongoDB for MongoDB clusters in distributed environment is examined.

General Terms

SMAC, BSON, CLI (Command Line Interface) of MongoDB, document based database.

Keywords

Auto-Sharding, Cloud Computing, Load Balancing, MongoDB, NoSQL.

1. INTRODUCTION

Latest years have observed a rapid burst in web applications that are burgeoning at an astounding rate daily. As complexity of web applications increases, their necessity for storing data is subjected to grow exponentially. Relational DBs have prevailed in markets for years as a solution for data storage with many read actions and less number of write actions [1]. These let records to be kept and retrieved in tabular manner. However, the tabular structure has restrictions on how to span horizontally, columnar wise because of the need of huge quantity of space for storing each row. The issue can be solved by getting the data and dividing it into many relations or normalization of data helps us overcoming the problem, though, this means now the data is dispersed over the disk and needs several read operations at diverse sectors of the drive to regain the info [2]. As the data needed isn't confined to a single spot thus the query to retrieve records from relations with zillions or billions of rows can easily start to get crashing and take considerable amount of time to fetch back results. Hard disk drives are yet a big restriction for I/O access and maximum databases have need of enormous volume of space for storage of their structures.

SMAC [3], the acronym for Social, Mobility, Analytics and Cloud is becoming a new flavor for business reality. By 2020, as many as hundred billion computing gadgets will be associated with the Web and firms will be dealing fifty times the data than they do presently [3]. Due to this digitization of business models and processes data generation rates have grown sharply over the years. Humongous amount of users tend to request for same data at an instance and probably want to write some data at the very same time. Management of this amount of user requests is the main concern for organizations that have large scale distributed systems and they want to provide their clients with seamless and highly available services with least response time [4].

In Web2.0 applications, the performance and real time access of database is more vital than ACID properties. The resolution for treating such problems is to use NoSQL databases. The CAP theorem, also referred to as Brewer's theorem, tells that it is not possible for a distributed computer system to concurrently offer all three of the subsequent assurances [5]:

- Consistency: All nodes have the exact same data at the same instance of time.
- Availability: The node will answer queries at all times if possible.
- Partition tolerance: Works regardless of a network failure so nodes can communicate amongst them.

There are numerous pros and cons for both SQL and NoSQL databases. Developers repeatedly wonder as what database will go with their requirements and which amongst them is best so that they may use for their application development. There is no best or direct solution for this query and there is not a single database that would work well for each project. MongoDB has both strengths and weaknesses [6], but in general it does quite good and it doesn't have several constraints and restrictions as other NoSQL databases.

The performance of database is predominantly determined by the strategies for allocation of data and the condition of load over system for the reason that uniformly spread workload can largely help to optimize resource consumption, maximize throughput and reduce potent overload over system. Nevertheless, considering other aspect, most load-balancing algorithm and solutions for NoSQL are not that established or are having less adaptableness since it is still new to market as compared to solutions for load-balancing in traditional systems and also the prevailing load-balancing methods are not applicable for NoSQL databases[7] [8]. Thus need of the hour is to develop new strategy for balancing of load over clusters that is more effective, complaint with NoSQL solutions and is applicable easily.

This report reviews the background research for the topic. In order to explore the association between prevailing load-balancing algorithm and MongoDB cluster's load-balancing, it is supposed to primarily study the both fields individually. So the second and the third sections of this report correspond to the examination of load-balancing algorithm and MongoDB individually and finally improvement in MongoDB Cluster's balancer method using custom algorithm. Even though both fields have numerous diverse uses in themselves, it is tried to emphasize on the central ideas and the readings are thought would be valuable for current task. This report might seem to be quite broad-spectrum. Next part defines and reviews some readings about MongoDB, what are the current research completed with respect to improvement of MongoDB clusters, emerging better algorithms for load-balancing and MongoDB balancing technique.

2. LITERATURE REVIEW

Devoid of load-balancing, cloud computing management would become very hard. With the managed redirection, a central unreliable system can be made reliable through the gains offered by load-balancing in terms of fault tolerance together with failover mechanism. [9]. In spite of the amount of study that has been carried out with the expectation of better comprehension about the development of MongoDB's load-balancing concept or auto scaling mechanisms as autosharding, studies have shown various aspects to be considered for developing a better load-balancing technique [9].

Research process started with reading books, articles, and research papers dealing with the new and upcoming technology-NoSQL to get understandings of the technology and understand it. The relational databases have conquered the market for so long. Neither they were intended to face the scaling challenges which current applications have to deal with, nor were they developed to get advantage of the inexpensive storage and processing power. The following literature reviews aim to establish the same fact about load-balancing for MongoDB clusters.

In the first paper, Nyati S.S., Pawar S, Ingle R. [10] have very carefully elucidated few NoSQL unstructured databases and further performance examination of these considered databases was presented which was done on basis of several benchmarks as querying database, performance time, contrasting the time necessary for inserting data in different databases and also examining them with different number of entries. Diverse kinds of NoSQL DB types were discussed. Amongst all of them Nyati et al chose MongoDB as best option for their evaluation purpose because it is able to handle less data along with large data proficiently. The comparison is made between MongoDB and MySQL over various benchmark and the results showed MongoDB is very quicker in inserting the data as well as quicker in searching.

Brust A. [12]: In this article writers threw some light upon a verdict for NoSQL by a RDBMS prodigy of IBM and Oracle named Janan Dash. With the swarming of terms like Big Data, NoSQL, Database Appliance, NewSQL etc. offers various issues for traditional relational database management system (RDBMS) operators around the globe. Now is the time of dynamic schema in which updating of records must to be reflected daily, if not done on per hour basis, to meet the ever changing requirements of new data model. If an association is dealing with these types of concerns, then according to the writer they must make an intelligent choice switch over to NoSQL technologies, because most of them were specifically developed and designed to undertake these issues regarding scale-in (horizontal scaling) or scale-out (vertical scaling). NoSQL proposes ranges of solutions of firm to relaxed consistency which are essential to be considered as on a case-by-case basis. IBM has implemented the MongoDB API, data illustration, query semantic and wire protocol, thus forming a path for mobile and several other new applications to link with enterprise DBs such as IBM's DB2 relational DB and the grid WebSphere eXtreme Scale data grid. Performance and other features will continue to advance and develop over time for NoSQL databases. [12].

Kaur K, Rani R's [13] present work explained about modeling data in NoSQL DBs and way to query them. Authors elaborated different classes of NoSQL DBs briefly. Four classes of NoSQL databases were explained out of which two were taken under consideration i.e. a graph DB-Neo4j and a document DB- MongoDB. The contrast was presented between them, how they were able to work with variety and volume of data and how query semantics and syntaxes differ from one another. MongoDB, document database, is used to store, retrieve and handle semi-structured data, which is kept in the form of documents. It has no provision for joins but was designed specifically to work with growing data storage needs. On the contrary graph databases models the entire database as a network structure swarming with associations between nodes. Objects were stored in nodes and edges linking nodes work as entities and relationships, equivalent to relational database architecture. It is ACID compliant, has the capability of storing semi-structured information and also hierarchical data can be best denoted in graph databases. MongoDB has its own query language. Cypher is a declarative graph query language used to query Neo4j graph database. The further query formats are explained with the help of examples.

In their paper, Aggarwal R, Arora R [14] , have vividly described query execution and data modeling in MongoDB NoSQL database and demonstrate it with the help of class diagrams and also one more feature is illustrated i.e. no JOIN support. For illustrating modeling of schema of the database, class diagram and JSON format was used. Process of storing data in the denormalized form in MongoDB is known as embedding of document which means related data is stored in a single document that are JSON-style format made of key-and-value pairs. Rules for making collections and documents were explained further in the study. Using some sample collections, query format was elucidated carefully by giving corresponding queries in MySQL for better apprehension and understanding.

Next study was based on comparing two types of databases NoSQL document oriented database management system and relational DB [15]. The authors Alexandru B., Florin R., Laura I. A. compared Oracle Database and MongoDB. Different parameters were taken under consideration out of those were theoretical alterations, query and insertion times, structures, constraints, integrity, architecture, distribution and system requirements.

According to Alexandru et al., MongoDB focuses to four things: power, flexibility, ease of use and speed. It provisions features as indexing and replicated servers and it has support for multiple programming language as drivers are provided them. NoSQL's MongoDB is schema-less database model whereas Oracle is relational database model. The MongoDB can take humongous amount of data of extreme magnitude of 16 MB in contrast to Oracle Database which has extreme magnitude of merely only 4 KB. The main difficulty with the Oracle Database was the replication that wasn't in case of other database under consideration i.e. MongoDB. Also the Oracle DB is way sluggish in contrast to MongoDB. The conclusion that can be drawn out of this was that if you require a database to be fast and flexible, MongoDB is the answer. On the contrary, if you require relations amongst tables then without any concern or worrying about quickness you can trust the standard solution, relational database, Oracle Database.

Coe, B [16]: This article offered a basic comprehension of MongoDB. According to the author, MongoDB has schema less data-representation and is devoid of joins- these two features mark an imperative difference between it and other SQL-based technologies that were already ruling the field. For this non-relational document DB, collections and documents are respectively equivalent to tables and rows for relational DBs. The author has explained some details regarding concepts related to MongoDB. He elucidated that a in MongoDB collection, two documents can have different fields, and also the same field may have various data types. MongoDB's CLI is driven by JavaScript.

Queries can be done inside inner documents and inner arrays. Further MongoDB's built-in sharding support was explained as one of its most flaunted features. Much like indexing a column in SQL table, MongoDB allows us to index fields within documents. Replication means having a sole parent MongoDB server, with one or more child servers related with it. At conceptual level author discusses, sharding technique. Finally the article was concluded with some use cases for MongoDB primarily emphasizing upon Craigslist, Foursquare, and Bit.ly as great examples for MongoDB being used by them to carry out queries on a solo gigantic dataset and the faster results being produced by the NoSQL solution being discussed.

Arora R [17]: This paper proposed an algorithm for transforming relational data (here considered for MySQL data) to a document based non-relational data (here considered for MongoDB's document based structure). According to author, with the popularity of benefits of NoSQL technologies, many businesses and consumers want to migrate to NoSQL solutions. Henceforth, there is an ardent requirement for transformation of data from relational form to NoSQL databases' form. The proposed process for conversion of data was developed using NetBeans and Pentaho. The algorithm converted the datasets from relational form to document based NoSQL class. The algorithm first created link with MySQL server after that user chose the desired DB from listed relational DBs whose data was desired to be converted into MongoDB's document model. This resulted in documents within documents known as embedded documents. Text files were created in structure compatible with Pentaho DI tool. It received text files as input and produced the resultant MongoDB collection. As a future scope author suggested to extend the capability of converting data from relational form to other distributions of NoSQL solutions available in market.

Banker K [1]: Chapters 1-6 dealt with MongoDB's background, policies and scenarios where MongoDB is deployed, what makes MongoDB unique, compared it with other NoSQL DBs evolving. Primarily developed for scalability requirements of present internet applications, MongoDB supports dynamic queries and secondary indexes; quicker atomic updates and complex aggregations; and provisioning sharding for scaling horizontally along with replication with automatic failover. Rest of the chapters taught application development in MongoDB using its JS shell, basic CRUD operations and aggregation queries.

Chodorow K [2] in her book explained what is sharding and its principles. The author explains shard as one or more servers inside a cluster which are liable for some dataset. A MongoDB cluster fundamentally comprises of three kinds of processes: the mongos processes for directing requests towards the desired data, the shards for essentially storage of data, and the config servers for observing the cluster's condition. The author also demonstrates the setting up of cluster and how to use a shard key on basis of which data will be divided and distributed to different shards. And last chapter describes administration of MongoDB clusters.

Liu Y, Wang Y, and Jin Y [7]: In this paper, firstly authors presented the ideologies and deployment approaches of auto-sharding in MongoDB, and then put forward an enhanced form of their algorithm to resolute the difficulty of jagged distribution of data in auto sharding based on occurrence of data operations. The upgraded balancing scheme successfully equilibriums the data amongst shards, and increases the cluster's simultaneous writing and reading performance. Contrast between the suggested algorithm and auto sharding technique is done with the help of testing simultaneous read and write execution of the cluster. Initially, a test was executed to assess the simultaneous writing performance of the cluster. Ten million records were inserted and parallel reading behavior was also evaluated by keeping the amount of records unaltered. Enhanced results were shown and validated with the graph. Proposed data balancing strategy based on occurrence of data operation gave better results and its efficiency was proved with the help of conducting experiments. The synchronized writing and reading performance of the auto sharding cluster is considerably enhanced.

Zugic G [18]: This article provided insight of what was horizontal scaling and vertical scaling .Then the important thing that this article talked about was the sharding keys and how to select the best shard key for data. A comprehensive explanation for sharding key selection criteria is well explained. If the application is less write scalable but query isolation is important then range shard key is used, if it is highly write scalable but query isolation is not required then hashed shard key is used.

Huang, Chao-Wen; Hu, Wan-Hsun; Shih CC;Lin BT;Cheng CW [8]: In this study, the use of on-request characteristics of cloud computing and sharding characteristics of MongoDB were explained to offer a solution for virtualized auto-scaling database that would meet the SLA requirements. In the beginning, the author used auto-scaling mechanism of route server in MongoDB system. The experimental results exhibited that the average response time of auto-scaling DB solution was 4.3 seconds and non-scaling DB solution was 7.1 seconds. Secondly, they also prototyped a shard data transfer algorithm that resulted in reduced impact while migrating data to new virtual machine. The auto-scaling DB solution used the algorithm to determine how many VM to be further added and what data to be moved to those newly added VM.

Wang XL, Chen H, and Wang Z [19]: This paper highlighted the inclination towards MongoDB for its proficiency in automatic load-balancing system and the auto sharding method. Automatic load-balancing comprised of dispersion of read load from primary to secondary node for reduced of load of primary node while auto sharding was implemented for decrease load on a specific node by spontaneous division of data into chunks and later which is migrated to some of other nodes. The authors have concentrated on the mechanism of automatic load-balancing of MongoDB and suggested a dynamic load-balancing technique based on heat diffusion from cluster with much less cost. They arranged an experimental setup for their planned algorithm and presented their enhanced outcomes. Final conclusion that can be made is that traditional data amount-based load-balancing method was not adequate enough to efficiently equilibrate the data among shards so the suggested dynamic load-balancing algorithm based on heat diffusion is significant for improved results.

3. CONCLUSION

The load-balancing of servers is vital for storage applications that are mostly read intensive. Traditional balancing methods cannot be relied upon for distributed environment. So an efficient solution for balancing load over distributed MongoDB clusters will be developed and eventually increase their performance when huge amount of load arises. The results shall be verified by algorithm implementation. The algorithm will initially monitoring all the shards whether they are balanced or under loaded or overloaded. Then if a shard is monitored to be overloaded then according to improvised version of algorithm the load is redistributed till the shards are balanced. And few more aspects can be examined, such as migration threshold value of documents so that network traffic is not hampered, for future consideration.

4. ACKNOWLEDGEMENTS

I owe my gratitude to all those persons who have helped in making this study possible so far. Primarily I offer my genuine gratitude to my mentor, Mr. Janpreet Singh, who has supported me throughout my study process with his persistence and knowledge while permitting me the area to work in my own way. One simply could not wish for a better or approachable mentor.

I am thankful to Mr. Nitin Kumar for his inspiration and practical advice. I am also grateful to him for reading my reports, commenting on my views and helping me understand and enrich my ideas.

I would like to acknowledge Mr. Rohit Dhand for numerous discussions and talks on associated topics that aided me advance my knowledge in the area.

Many friends have helped me stay sound through these difficult months. Their support and care helped me overcome obstacles and stay concentrated and motivated towards my study. I prominently value their bond and I deeply appreciate their belief in me. Most notably, not a bit of this would have been imaginable deprived of the love and perseverance of my family. My family has been an endless source of affection, concern, care and strength all these months. I would like to express my genuine gratitude to my family.

5. REFERENCES

[1] Banker K (2012).MongoDB in Action. Shelter Island, NY: Manning Publications Co. pp. 3-126.

[2] Chodorow K. (2011) Scaling MongoDB. L. M, Ed., Sebastopol, CA: O'Reilly Media, Inc. pp. 1-45.

[3] http://www.tutorialspoint.com/mongodb/mongodb_advantages.htm. (n.d.). Retrieved from http://www.tutorialspoint.com.

[4] H. C., Z. W. XiaoLin Wang (2013). Research on Improvement of Dynamic Load Balancing in MongoDB. Dependable, Autonomic and Secure Computing (DASC). IEEE 11th International Conference.

[5] CAP Theorem, FoundationDB. https://foundationdb.com/key-value-store/white-papers/the-cap-theorem. Retrieved from https://foundationdb.com.

[6] http://www.databaseskill.com/3091951/. Retrieved from http://www.databaseskill.com

[7] Liu Y, Wang Y, Jin Y (2012). Research on the improvement of MongoDB Auto-Sharding in cloud environment. In The 7th International Conference on Computer Science & Education, Melbourne, VIC.

[8] Huang, Chao-Wen; Hu, Wan-Hsun; Shih CC; Lin BT; Cheng CW (2013). The improvement of auto-scaling mechanism for distributed database: A case study for MongoDB. Network Operations and Management Symposium (APNOMS), 15th Asia-Pacific, Hiroshima, Japan.

[9] Sosinsky B. (2011). Cloud Computing Bible. Indianapolis, Indiana: Wiley Publishing, Inc.

[10] Nyati S.S., Pawar S, Ingle R. (2013). Performance Evaluation of Unstructured NoSQL data. In International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore.

[11] NOSQL DATABASES EXPLAINED. http://www.mongodb.com/nosql-explained. Retrieved from http://www.mongodb.com

[12] Brust A., Dash J. RDBMS vs. NoSQL: How do you pick? http://www.zdnet.com/rdbms-vs-nosql-how-do-you-pick-7000020803/. Retrieved from http://www.zdnet.com/

[13] Kaur K, Rani R. (2013).Modeling and querying data in NoSQL databases. In IEEE International Conference on Big Data, Silicon Valley, CA.

[14] Aggarwal R, Arora R (July 2013). Modeling and Querying Data in MongoDB. In International Journal of Scientific & Engineering Research, vol. Volume 4, no. Issue 7, pp. 141-145.

[15] Alexandru B., Florin R., Laura I. A. (2012). MongoDB vs. Oracle - database comparison. In Third International Conference on Emerging Intelligent Data and Web Technologies, Bucharest.

[16] Coe, B. To MongoDB, or Not to MongoDB. http://www.codemag.com/Article/1309051. Retrieved form http://www.codemag.com

[17] A. R. Arora R (2013). An Algorithm for Transformation of Data from MySQL to NoSQL (MongoDB). International Journal of Advanced Studies in Computer

Science and Engineering (IJASCSE), vol. 2, no. 1, pp. 6-12.

[18] Zugic G, Selecting a MongoDB Shard Key (29 May 2014). https://goranzugic.wordpress.com/2014/05/29/selecting-a-mongodb-shard-key/. Retrieved from https://goranzugic.wordpress.com.

[19] Wang XL, Chen H, Wang Z (2013). Research on Improvement of Dynamic Load Balancing in MongoDB. Dependable, Autonomic and Secure Computing (DASC), 2013 IEEE 11th International Conference Chengdu.