IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Consistency Improvisation in MongoDB during Lag on secondary

Abuzar Kamal[1], Mr. Saravana Kumar K[2]

Student, Department of Computer Science, Christ University, Bangalore, India [1]

Assistant Professor, Department of Computer Science, Christ University, Bangalore, India[2]

Abstract: Replication lag in Mongo database is serious problem in replication environment if the reporting is done through secondary server, In replication environment the read only query can be routed to secondary server, however if there is huge lag then it can result into inconsistent data. In the current version this issue has not been addressed, In the proposed system, an attempt has been made to overcome with this problem. Before execution of query the session should verify the lag, if the lag is beyond the specified limit then the query should be re-routed back to primary. This is possible through continuous short messaging between primary and secondary. The metadata about the lag can send through the messaging, upon reading message the secondary server should re-direct the query to primary server.

Keywords: Mongo DB, Replication, Lag, Consistency, Replica set, NoSQL, CAP Theoram.

I. INTRODUCTION

In MongoDB, replica sets are group of servers which works together to provides the high availability by employing the replication. It does not uses the conventional master-slave replication. Replica Sets improves master-slave replication with failover capabilities [1]. If a primary node is down, one of the secondary nodes become new primary. The secondary node initiates an election among secondary nodes, if it cannot reach the primary node [2]. Replicates can be used for offloading reads and writes. In read offloading, secondary nodes will Perform the read operation. Since replication is asynchronous, and there is always a time lag between a write request executing on primary node and the read request executing on secondary server, data can be inconsistent.

MongoDB cannot ensure the consistency for reads operation from secondary servers, one can setup the client and driver to ensure that write guarantee strict consistency for read operations from secondary members. More over if the replication strategy includes the cascading replication then the lag issue will be compounded. In the proposed system we are making and effort to reduce the inconsistencies when the read operation is performed on secondary.

II. MOTIVATION

In real world we do need to offload the reporting queries to the standby side that reduces the load from the production but this comes with certain penalty i.e. stale data. The motivation behind this research is to reduces the staleness and provide more recent data. As per the business logic the lagging period can be defined and based on the lag limit the conditionally query can be routed to other available standby which has lag under the specified limit or to the primary.

This proposal certainly not going to make the MongoDB completely ACID compliant but it's certainly going to improve the consistency when replication is in place. Every business model has different requirement, therefore by specifying the lag limit the staleness in report can be reduced.

III. REPLICATION ARCHITECTURE

The opinions and complaints posted on social networking A replica set in MongoDB is a group of Mongod processes that maintain the same data set. Replica sets provide redundancy and high availability, and are basic building blocks for all production systems. Replication provides redundancy and high availability. With multiple copies of data on different database nodes, replication provides a level of fault tolerance against the loss of a single database server.

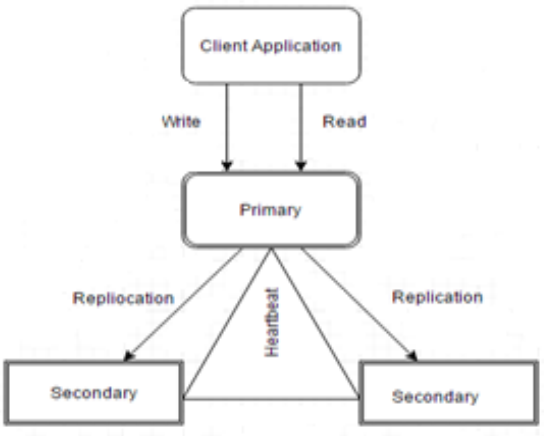

Fig 1. Replication Architecture

DOI10.17148/IJARCCE.2017.63164

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Figure 1 shows the replication architecture of MongoDB. There is one primary and two secondary, The changed made to the primary is replicated to secondary. Secondary server does supports the read only query. Hence the read only query can be routed to secondary. The replication depends on the network between the nodes, Its uses the TCP/IP protocol for the communication.

In some cases, replication can provide better read efficiency as clients can send read operations to different nodes. Maintaining multiple copies of datain different location can increase availability for distributed applications [5]. You can also keep additional copies for dedicated purposes, such as disaster recovery purpose, reporting, or backup [9]. The replica set surely gives the advantage when it comes to high availability and disaster recover however, it comes with certain penalty which is lag. Therefore, if the data is read form the system having lag will result in the stale data.

IV. REPLICATION LAG

Replication lag is the time delay between the last transaction committed on primary and that the last transaction applied on secondary [13]. In a smoothly running replica set, all secondary closely follow changes on the primary, fetching each group of operations from its oplog and replaying them approximately as fast as they occur. That is, replications lag remains as close to zero as possible. Reads from any node are then reasonably consistent; and, should the current primary become unavailable, the secondary that assumes the PRIMARY role will be able to serve to clients a dataset that is almost identical to the original.

The problem starts when the lag is huge, so if the application is configured to route the read query to secondary then it will not get the latest data because of lag. Although the MongoDB is not ACID compliant but with the proposed system the consistency can be improved.

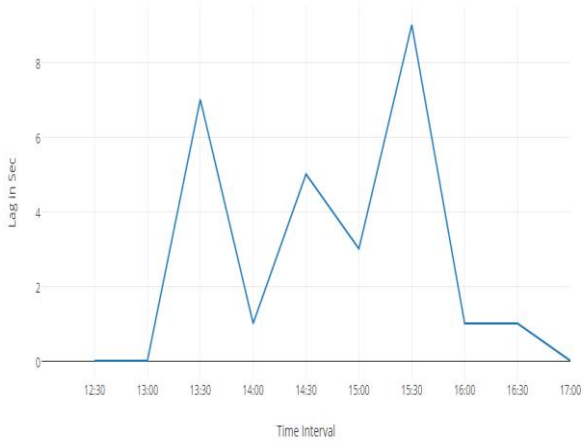

Fig 2 Replication lag

Figure 2 shows the spike in the lag because of network issue. The replication can never be removed completely as it depends on network. The lag plays major role in the deteriorating the consistency. Therefore, to improve the lag high speed interconnect network should be employed.

Secondary node must have sufficient network bandwidth so that it can retrieve oplog from primary at the reasonable rate and also enough storage throughput that it can apply the oplog i.e., read any affected JASON documents and their index entries into memory, and commit the altered JASON documents back to disk nearly at that same rate. CPU rarely becomes a bottleneck, but it may need to be considered if there are many index keys to compute and insert for the documents that are being added or changed.

V. PROPOSITION

In the proposed system the primary can update the secondary with short message about the current log sequence, since the message is very shot there is very little overhead of passing it to the secondary. In the proposed architectural diagram, Primary and secondary is in replication mode. Using the interconnection between the primary and secondary, short message can be sent about the current transaction log number, Upon receiving the secondary can verify the lag, Therefore If the lag is beyond the predefined time then the session will be routed back to primary.

The size of message should be very small as because during the time on network congestion. Sending message during the congestion can be problematic sometimes, therefore, separate network or subnet can be used to sending message. This technique is widely available in the clustered environment. The messaging passing can also deployed the UDP protocol for faster delivery as the UDP does not require the 3 way handshaking.

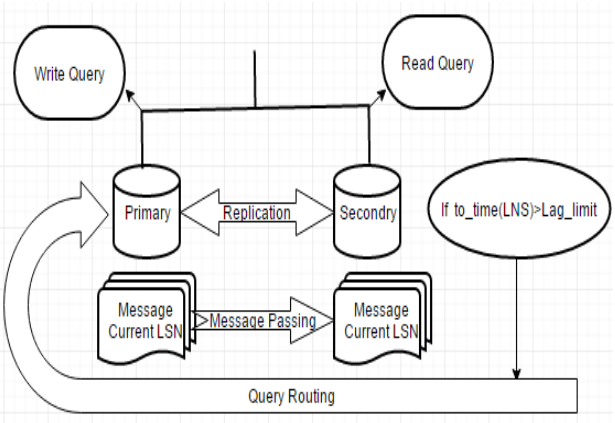

Fig 4. Replication and Message passing Architecture

VI. CONSISTENCY

A relational database is mainly used by application to storing and retrieving data. Managing a large amount of data like the internet was incompetent in RDBMS. To conquer this problem, NOSQL comes into reality. The name attempt totag the appearance of a mounting number

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

of non- relational, distributed data storage that frequently did not attempt to give ACID. There are many benefits of NoSQL as compared to RDBMs, but also there are many obstacles to overcome before they can appeal to conventional enterprises.

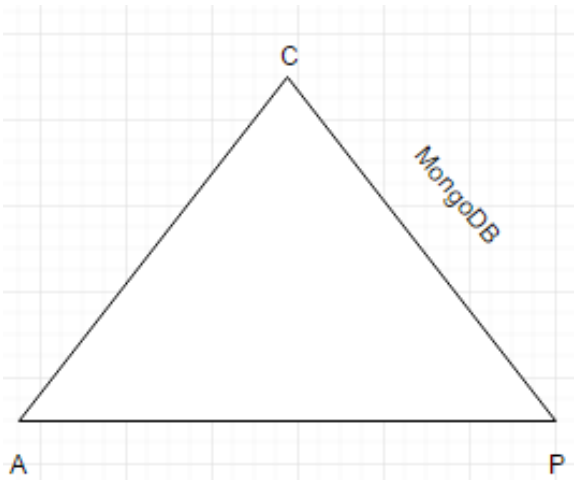

Fig 5. CAP and Mongo

CAP stands for consistency (C), availability (A), and partition tolerance (P). Therefore, the system only which supports Consistency and Availability but not the partition-tolerant, On the other hand the system which supports Consistency and Partition tolerance not support highly available, and the system which supports Availability and Partition tolerance does not supports consistent. There is fundamental tradeoff between consistency, availability, and latency. This tradeoff exists even when there are no network partitions, and thus is completely separate fromthe trade-offs CAP describes. The reason for the tradeoff is that a high availability requirement implies that the system must replicate data. If the system runs for long enough, at least one component in the system will eventually fail. When this failure occurs, all data that component controlled will become unavailable unless the system replicated another version of the data prior to the failure.
Consistency is strong in primary MongoDB server even in replica set configurations. However, secondary nodes maybe out of sync and MongoDB can only ensure eventual consistency with respect to the primary MongoDB server [6]. By default, MongoDB does not allow reads from secondary servers because of the chances of inconsistent data, however this can be changed knowing that changing the default could cause inconsistent data reads. As per CAP theorem Consistency and Availability both cannot be achieved together as in this case.

MongoDB allows two distinct types of distributed system: sharding for horizontal scaling and replica sets for failover (HA). Both can be employed together or in isolation. CAP theorem applies little differently to the two forms:
Sharding level - MongoDB stores data set on at most one authoritative shard.
Replica set level - MongoDB replicates primary data within a shard, guarantees consistency via a central, authoritative primary server.

VII. CONCLUSION

The rise of modern e-commerce and social networking gave rise to the NOSQL databases, which does not fit into the conventional relation database. The data gathered form these sources are humongous and the type of data it deals with is impossible to be handled by the conventional RDBMS. MongoDB uses the sharding methodology to handle the humongous data. Replication plays a vital role while employing the high availability. However, the lag during the replication major obstacle to offload the query. The offloading is normally used to reduce the load from the primary, as MongoDB supports the read only query from the secondary.

However, while reading form secondary can produce the stale report because of network lag. From the architectural prospective eradicating the lag completely is nearly impossible. Hence the algorithmic approach can be employed to reduce the lag, in turn improving the consistency. Improvement at the level one consistency will play a major role in terms of reliability of the MongoDB. In today's world we have number of databases which are fully ACID compliant, hence for MongoDB to compete with other relational architecture, this proposition is going to improve the MongoDB drastically.

REFERENCES

[1] 10gen Inc., "The MongoDB 2.2 Manual," Online, 2012. [Online]. Available: http://docs.mongodb.org/v2.2/. [Accessed: 03-Feb-2012].
[2] MongoDB replication DOC https://docs.mongodb.com/manual/replication/
[3] Alexandru Boicea, Florin Radulescu, Laura Ioana Agapin, "MongoDB vs Oracle - database comparison", IEEE 2012
[4] Kris Zyp http://www.sitepen.com/blog/2010/05/11/ nosql-architecture/, May 2010
[5] http://www.rackspace.com/blog/nosql-ecosystem/
[6] Ruxandra Burtica, Eleonora Maria Mocanu, Mugurel Ionuț Andreica, Nicolae Țăpuș, "Practical application and evaluation of no-SQL databases in Cloud Computing", IEEE 2012
[7] Jim Gray, "The Transaction Concept: Virtues and Limitations", Proceedings of Seventh International Conference on Very Large Databases, June 1981
[8] F. Chang et al, "BigTable: A Distributed Storage System for Structured Data", Seventh Symposium on Operating System Design and Implementation, November 2006.
[9] B. Cooper et al, "Benchmarking Cloud Serving Systems with YCSB", ACM Symposium on Cloud Computing (SoCC), Indianapolis, Indiana, June 2010.
[10] B. DeCandia et al, "Dynamo: Amazon's Highly Available Key-Value Store", Proceedings 21st ACM SIGOPS Symposium on Operating Systems Principles, 2007.
[11] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility