International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
Available at http://inpressco.com/category/ijcet

Research Article

MongoDB Performance Tuning using Sharded Cluster and Redis Cach

Doppa Srinivas

Computer Science, Padmabhushan Vasantdada Patil Institute of Technology, PVPIT, Pune, India.

Received 10 Nov 2020, Accepted 10 Dec 2020, Available online 01 Feb 2021, **Special Issue-8 (Feb 2021)**

Abstract

With the rapid development of the Database technology, the demands of large-scale distributed service and storage have brought great challenges to traditional relational database. NoSQL database which breaks the shackles of RDBMS is becoming the focus of attention. In this paper, the principles and implementation mechanisms of Auto-Sharding in MongoDB database are firstly presented, then Redis Cache is used to handle the frequency of data operation is proposed in order to solve the problem of uneven distribution of data in auto-sharding. The improved balancing strategy can effectively balance the data among shards, and improve the cluster's concurrent reading and writing performance.

Keywords: *Sharding, NoSQL, Redis, Database*

1. Introduction

The term "NoSQL" was first introduced in 1988 for the relational databases not having SQL interfaces [1]. However, the term was re-introduced in 2009 for the kind of modern web-scale databases which trade transactional consistency over large-scale data distributions and incremental scalability. NoSQL databases were not originally meant to replace traditional databases, rather they are more suitable to adopt when relational databases does not seem appropriate. The main reasons behind adopting NoSQL databases are their simple yet flexible architecture and the capability of handling large amount of multimedia, word processing, social media, emails and other unstructured data files [6]. While, the conventional relational databases are hard to scale-out mainly due to their pre-defined schemas and I/O performance bottlenecks [2, 4]. These issues have made relational databases difficult to fit in the new computing paradigms such as Grid and Cloud applications, data warehousing, web2.0, social networking [7] etc. Contrary to it, NoSQL databases are becoming a primary choice for cloud applications because of their highly available, reliable and scalable nature [8].

The BASE (Basically Available, Soft State and Eventually Consistent) properties of NoSQL databases allow them to be scalable and thus, NoSQL systems inherently support autosharding phenomenon [2]. Auto-sharding is the automatic and native horizontal distribution of data among different severs in NoSQL databases which, in turn, increase the performance and throughput of database operations [9]. Another significant advantage of auto-sharding is load balancing across the cluster such that a single server does not get overloaded with all the queries. This makes NoSQL databases a good candidate for high transaction and write-intensive database applications [4, 5].Redis cache is to further improve the performance of the database.

2. Literature Survey

A. Sharded Cluster

Sharding is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

Database systems with large data sets or high throughput applications can challenge the capacity of a single server. For example, high query rates can exhaust the CPU capacity of the server. Working set sizes larger than the system's RAM stress the I/O capacity of disk drives.

There are two methods for addressing system growth: vertical and horizontal scaling.

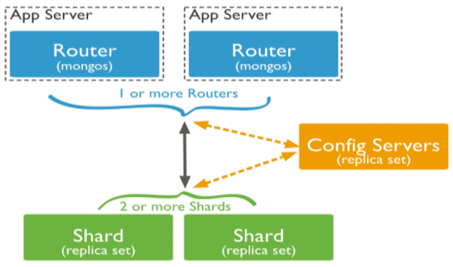
Vertical Scaling involves increasing the capacity of a single server, such as using a more powerful CPU, adding more RAM, or increasing the amount of storage space. Limitations in available technology may restrict a single machine from being sufficiently powerful for a given workload. Additionally, Cloud-based providers have hard ceilings based on available hardware configurations. As a result, there is a practical maximum for vertical scaling.

Horizontal Scaling involves dividing the system dataset and load over multiple servers, adding additional servers to increase capacity as required. While the overall speed or capacity of a single machine may not be high, each machine handles a subset of the overall workload, potentially providing better efficiency than a single high-speed high-capacity server [3]. Expanding the capacity of the deployment only requires adding additional servers as needed, which can be a lower overall cost than highend hardware for a single machine. The tradeoff is increased complexity in infrastructure and maintenance for the deployment.
MongoDB supports horizontal scaling through sharding.

A MongoDB sharded cluster consists of the following components:

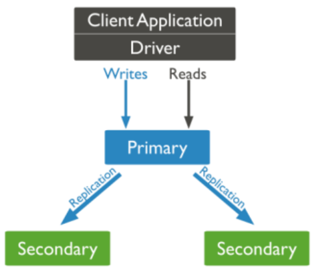
- **Shard**: Each shard contains a subset of the sharded data. Each shard can be deployed as a replica set.
- **Mongos**: The mongos acts as a query router, providing an interface between client applications and the sharded cluster.
- **Config Servers**: Config servers store metadata and configuration settings for the cluster. As of MongoDB 3.4, config servers must be deployed as a replica set (CSRS).

The following graphic describes the interaction of components within a sharded cluster:



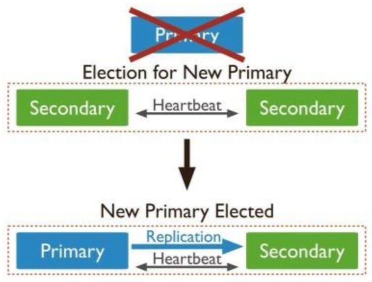
B. Replica Set

Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication provides a level of fault tolerance against the loss of a single database server.
A replica set is a group of mongod instances that maintain the same data set. A replica set contains several data bearing nodes and optionally one arbiter node. Of the data bearing nodes, one and only one member is deemed the primary node, while the other nodes are deemed secondary nodes.



The primary node receives all write operations. A replica set can have only one primary capable of confirming writes with { w: "majority" } write concern; although in some circumstances, another mongod instance may transiently believe itself to also be primary. [1] The primary records all changes to its data sets in its operation log, i.e. oplog. The secondaries replicate the primary's oplog and apply the operations to their data sets such that the secondaries' data sets reflect the primary's data set. If the primary is unavailable, an eligible secondary will hold an election to elect itself the new primary.

You may add an extra mongod instance to a replica set as an arbiter. Arbiters do not maintain a data set. The purpose of an arbiter is to maintain a quorum in a replica set by responding to heartbeat and election requests by other replica set members. Because they do not store a data set, arbiters can be a good way to provide replica set quorum functionality with a cheaper resource cost than a fully functional replica set member with a data set. If your replica set has an even number of members, add an arbiter to obtain a majority of votes in an election for primary. Arbiters do not require dedicated hardware.

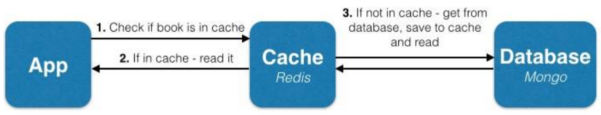


C. Redis Cache

Caching is a technique used to accelerate application response times and help applications scale by placing frequently needed data very close to the application. Redis [10], an open source, in-memory, data structure server is frequently used as a distributed shared cache (in addition to being used as a message broker or database) because it enables true statelessness for an applications' processes, while reducing duplication of data or requests to external data sources. In the redis cache we store the mongodb data so that for the next request we will not hit the database again.

D. What is an application cache

Once an application process uses an external data source, its performance can be bottlenecked by the throughput and latency of said data source. When a slower external data source is used, frequently accessed data is often moved temporarily to a faster storage that is closer to the application to boost the application's performance. This faster intermediate data storage is called the application's cache, a term originating from the French verb "cacher" which means "to hide." The following diagram shows the high-level architecture of an application cache:


App
1. Check if book is in cache
2. If in cache - read it
Cache
Redis
3. If not in cache - get from database, save to cache and read
Database
Mongo

3. Proposed Methodology

I have created three shards with five replica set each among those three are data nodes and two arbitrary nodes on three virtual machines and five config servers are created in other virtual machines. Hashed Sharding is used on the shard key so that all the records on the database randomly distributed and records has same shard key are stored in same shard because of they have same hash value.

Redis cache is used in between application and the mongodb database to improve the performance.

A. Architecture

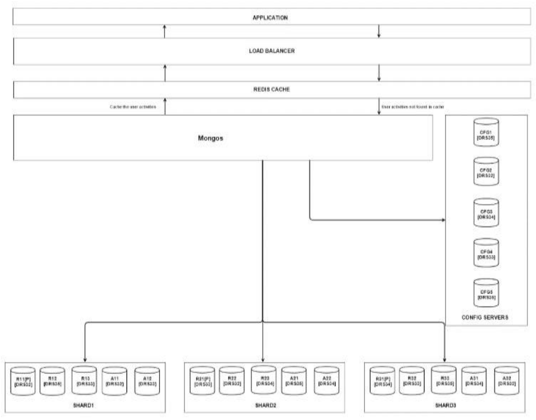


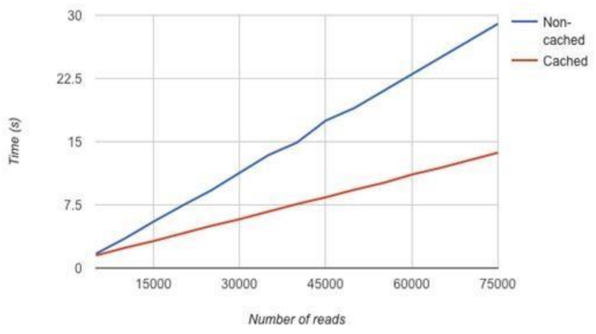
Table I.

<table>
<tr><th rowspan="2">S.No</th><th colspan="3">Hardware configuration details</th></tr>
<tr><th><i>Virtual Machine Name</i></th><th><i>RAM(in GB)</i></th><th><i>Number of Cores</i></th></tr>
<tr><td>1</td><td>DRS32</td><td>16</td><td>8</td></tr>
<tr><td>2</td><td>DRS33</td><td>16</td><td>8</td></tr>
<tr><td>3</td><td>DRS34</td><td>16</td><td>8</td></tr>
<tr><td>4</td><td>DRS35</td><td>16</td><td>8</td></tr>
</table>

Rij means ith shard **jth** Replica set and same arbitrary nodes **Aij**.

R12 means 1st shard 2nd replica set.**P** indicates the primary node.

S.No	*Number of Requests*	*Time Taken(sec)*
1	15000	1.5
2	30000	6
3	450000	8.2
4	60000	11.3
5	75000	14.2



The above graph shows the improvement of the performance of database.

Conclusions

The proposed system improved the read and write performance of the mongodb using sharded cluster and redis cache. And tested on setup of three shards with five replica sets each. We can increase the shards according to application requirements.

Choosing a right shard key is very important aspect of the system otherwise we may end up with wrong results. Cardinality and frequency of the shard key is also plays important role in the system.

The above mentioned performance is achieved to specific hardware only if hardware changed then whole results will be changed.

Further improve the performance by clustering redis cache and increasing the number of shards in the future.

Acknowledgment

cPGCON 2020 (Post Graduate Conference for Computer Engineering) Assistant Professor and Prof. Dr. B. K. Sarkar, Head, Department of Computer Engineering for sharing their pearls of wisdom with me during the course of this research, their comments on an earlier version of the manuscript, although any errors are my own and should not tarnish the reputations of these esteemed persons.

References

[1]. R. Masood. "Fine-Grained Access Control for Database Management Systems." MS (CCS) thesis, National University of Science and Technology, Pakistan, 2013.
[2]. 10Gen Corporation. "NoSQL Explained."Internet: http://www.mongodb.com/nosql-explained.
[3]. IBM Corporation. "Data Security and Privacy – A holistic Approach." Internet www.ibm.com/software/data/optim/protect-data-privacy.
[4]. CodeFutures Corporation. "Cost-effective Database Scalability using database Sharding." Internet: www.codefutures.com/database- sharding.
[5]. A. Viswanathan and C.J. Kothari. "Hibernate Framework-based
[6]. database sharding for SaaS Applications." Internet: http://www.ibm.com/developerworks/library/os-hibernatesaas.

[7]. C. Roe. "The Growth of Unstructured Data: What To Do with All Those Zettabytes?" Internet: http://www.dataversity.net/the-growth- of-unstructured-data-what-are-we-going-to-do-with-all-those- zettabytes, Mar.

[8]. N. Hardiman. "Cloud computing and the rise of big data." Internet: http://www.techrepublic.com/blog/the-enterprise-cloud/cloud-computingand-the-rise-of-big-data .

[9]. A. Schram and K.M. Anderson. "MySQL to NoSQL: data modeling challenges in supporting scalability," in Proc. of the 3rd annual conference on Systems, programming, and applications: software for humanity, 2012, pp. 191-202.

[10]. MongoDB Inc. "MongoDB sharding guide - Sharding and MongoDB Release 2.4.6." Internet: http://docs.mongodb.org/manual/sharding/, Mar. 2014.

[11]. Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker : https://redis.io/