Mongo DB

Table of Contents

Introduction

MongoDB has a flexible storage system, which means stored objects are not necessarily required to have the same structure or fields. MongoDB also has some optimization features, which distributes the data collections across, being overall a more balanced and performance focused system.

MongoDB is a schema-free, document-oriented database written in C++.

As a document store based, it stores values (referred to as documents) in the form of encoded data.

The choice of encoded format in MongoDB is JSON. This means that even if the data is nested inside JSON documents, it will still be queryable and indexable.

Architecture

Shards (mongod)

Sharding is the partitioning and distributing of data across multiple nodes. A **shard** is a collection of MongoDB nodes. Using shards also means the ability to make an horizontally scaling across multiple nodes. In the case that there is an application using a single database server, it can be converted to sharded cluster with very few changes to the original application.

Software is almost completely decoupled from the public APIs exposed to the client side.

Configuration servers (**mongod**)

Each one holds a copy of the metadata indicating which shard contains what data.

Routers (**mongos**)

A group of servers acting as a main interface for one or more clients.

A router dipatch the client requests to the appropriate shards using the configuration servers to know who owns the required information.

Global cluster

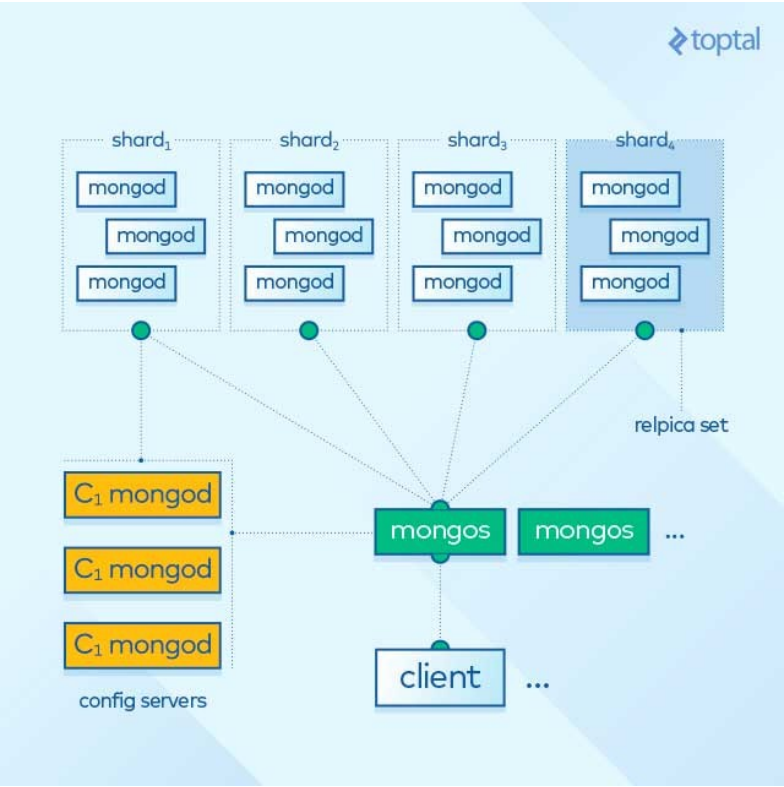
Read and write actions are sent from the clients to one of the router servers in the cluster, and are automatically routed by that server to the appropriate shards that contain the data with the help of the configuration servers.

A shard in MongoDB has a data replication scheme, which creates a copy set of each shard that holds exactly the same data. There are two types of replication schemes in MongoDB: Master-Slave replication and Replica-Set replication.

Replica-Set provides more automation and better failure handling, while **Master-Slave** requires the administrator intervention more often. Regardless of the replication scheme, at any point in time in a replica set, only one shard acts as the primary shard, all other replica shards are secondary shards.

All write and read operations go to the primary shard, and are then distributed evenly (if needed) to the other secondary shards in the set.

In the graphic below, we see the MongoDB architecture explained, showing the router servers in green, the configuration servers in yellow, and the shards that contain the blue MongoDB nodes.



It should be noted that sharding (or sharing the data between shards) in MongoDB is completely automatic, which reduces the failure rate and makes it a highly scalable database management system.

Supplementary information

Programming Languages Libraries

Official: C, C++, C#, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala

Community provided: https://docs.mongodb.com/ecosystem/drivers/community-supported-drivers/

Systems

Linux, Unix, BSD, Windows, Mac OSX, Android

Licensing

- Community Server: AGPL v3.0
- Enterprise server: commercial license
- Client: Apache License v2.0

Storage

Document based: BSON (Binary JSON)

Storage engines

https://docs.mongodb.com/manual/core/storage-engines/

- *WiredTiger* (default from v3.2): document-level concurrency model, checkpointing, and compression, among other features. In MongoDB Enterprise, WiredTiger also supports Encryption at Rest.
- *MMAPv1* (the original): performs well on workloads with high volumes of reads and writes, as well as in-place updates.
- *In-Memory*: available in MongoDB Enterprise. Rather than storing documents on-disk, it retains them in-memory for more predictable data latencies.

Query method and Interface

- *Wire Protocol*: primary method used by language libraries, faster and more secure.
- *RESTful API* (optional): CRUD database operations mapped to HTTP verbs.

HTTP verbs mapping:

- POST : insert or create an object.

- GET : read an object.
- PUT : update an object.
- DELETE : delete an object.

Some of the common HTTP result codes are often used inside REST APIs:

- 200 : “OK”.
- 201 : “Created” (Used with POST).
- 400 : “Bad Request” (Perhaps missing required parameters).
- 401 : “Unauthorized” (Missing authentication parameters).
- 403 : “Forbidden” (You were authenticated but lacking required privileges).
- 404 : “Not Found”.

Security features

Authentication

- SCRAM-SHA-1
- MongoDB Challenge and Response (MONGODB-CR)
- x509 certificate
- LDAP proxy (Enterprise only)
- Kerberos (Enterprise only)

Authorization

- Multiuser systems
- Role-Based access control

Encryption

- Transport Encryption TLS/SSL with OpenSSL
- Encryption at REST (Enterprise with WiredTiger only) with AES-256-CBC or AES-256-GCM (auth encryption) using OpenSSL libs.
- FIPS (Federal Information Processing Standard) mode (Enterprise only)

Other

Auditing: allows administrators and users to track mongod and mongos instances activities.

High availability

Partitioning and distributing data across multiple nodes (sharding).

Data can be duplicated across shards (replica sets). Data is written in a single shard instance and is successivelly distributed in the replica set.

Scalability

Very good horizontal scalability. Shards can be added easily on running systems.

Safer, easier and less expensive than vertical scaling (cpu,storage,ram).

https://docs.mongodb.com/manual/sharding/

Transaction model

When a single write operation modifies multiple documents, the modification of each document is atomic, but the operation as a whole is not atomic and other operations may interleave. For cases where a sequence of write operations must operate as if in a single transaction, you can implement a *two-phase commit* in your application. Using two-phase commit ensures data consistency.

Tools

- Compass: tool to visualize DB structs.
- Munin and Wireshark official plugins.
- Plenty FOSS software and huge community (mongodb-tools.com).
- Robo3T

Cloud

https://www.mongodb.com/cloud/atlas

- Amazon Web Services (AWS)
- Google Cloud
- Azure
- mLab

References

- Official documentation: https://docs.mongodb.com
- Wikipedia: https://en.wikipedia.org/wiki/MongoDB

Sharding

Components

A MongoDB sharded cluster consists of the following components:

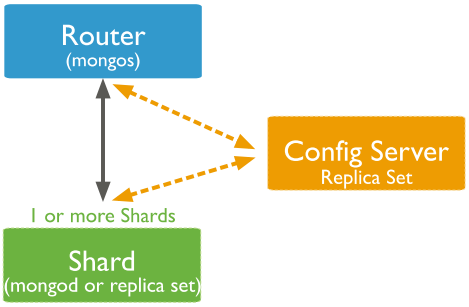
- **shards**: each shard contains a subset of the sharded data. Should be deployed as a replica set.
- **config servers**: store metatdata and configuration for the cluster. Deployed as a replica set.
- **mongos**: acts as a router. Provides an interface between the client application and the cluster.

To reduce network latency between the application and the router a common pattern is to place mongos in the same machine as the application. Nothing prevents to place the mongos in dedicated machines.

Another pattern is to place mongos on the machines of primary shards. Be aware of potential memory contention.

For development environments a common approach is to have:

- a config server with a single element replica set.
- at least one shard with a single single-member replica set.
- one mongos router



Shards

A shard contains a subset of sharded data for a sharded cluster.

Connections directly to a single shard should be avoided and done only to perform local administrative and maintenance operations (a direct modification of the shard does not update the config server).

Primary Shard

Each database in a sharded cluster has a primary shard that holds all the un-sharded collections for that database. Has no relation to the primary server in a replica set.

Note that the primary shard for two different databases could be different.

The mongos selects the primary shard when creating a new database by picking the shard in the cluster that has the least amount of data.

`sh.status()` method in the mongo shell to see an overview of the cluster.
Includes which shard is primary and the chunk distribution.

Chunk size

The default chunk size is 64 MB (allowed 1 to 1024 MB)

Config-servers

Store metadata for a sharded cluster. The metadata includes the list of chunks on every shard and the ranges that define the chunks. config-servers are deployed as replica set (max 50 members).

The `mongos` instances cache this data and use it to route read and write operations to the correct shards.

The `admin` database is related to user authentication and authorization.
The `config` database contains the sharded cluster metadata.

The `admin` and the `config` database exist on the config-servers.

MongoDB also uses these servers to manage distributed locks.

Mongos

Route read and write operations to shards. They are the only interface to the sharded cluster from the perspective of applications.

Are capable to cache metadata from config-servers and, since they consume minimal resources, is common to run `mongos` in the same machine where the application run.

The `mongos` merge the data fetched from the different shards before returning the documents to the application.

A broadcast operation is performed for queries that do not include the shard key, routing the query to all shards in the cluster. A targeted operation is performed only when the query includes the shard key, or a prefix of the shard key.

Practical commands

Check sharding status

```
sh.status()
```

Check if we are talking through `mongos`

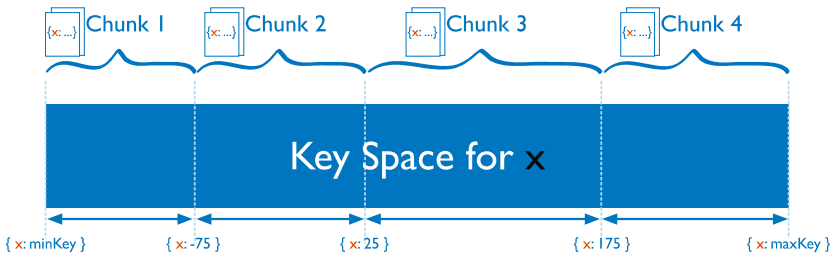
```
db.isMaster()

If through mongos then returns a document with a `msg` field holding the `isdbgrid`
string.
```

Enable sharding for a given database

```
sh.enableSharding("DBName")
```

Shard Keys

Determines the distribution of the collection's documents among the cluster's shards. The key is an indexed field or compound indexed fields that exists in every document in the collection.



Once the shard has been done, the shard key and shard key values are immutable.

To shard a collection the target collection and the shard key should be specified

```
sh.shardCollection(namespace, key)
```

The namespace consists of a <database>.<collection> string (e.g. `"DBName.CollectionName"`)

The key parameter consists of a document containing a field and the index traversal direction for that field (e.g. `{ fieldname: "hashed" } )`

The ideal shard key allows MongoDB to distribute documents evenly throughout the cluster.

Cardinality : maximum number of chunks the balancer can create. If the cardinality is 4 then no more than 4 chunks can exist.

Frequency: how often a given value occurs in the data

The field chosen as the hash key must have a good cardinality (or large number of different values). Hashed keys are ideal for shard keys with fields that cange monotonically.

Hashed indexes

Hashed sharding uses hashed indexes.

Hashed indexes maintain entries with hashes of the values of the indexed field