MongoDB : distributed storage

LINAGORA

Content

MongoDB Presentation

Document Oriented DB

Replication Model

MongoDB distributed architecture overview

Architecture details : Replica Sets

Architecture details : Sharding

MongoDB : Presentation

MongoDB : Presentation

- First version : 2009
- Last stable version : 3.2 (december 2015)
- Written in C++
- Developped by : MongoDB Inc.
- License : AGPL v3
- Numerous and active community
- Exhaustive documentation

- Widely used (Adobe, eBay, LinkedIn)

MongoDB Data Model

- Data model :
  - Document : BSON object
  - Collection : a collection of documents
  - Database : a collection of collections

- Each MongoDB instance can have multiple data bases

Document oriented DB

Document oriented DB

- Type of noSQL data base
  - Scalable using distributed architecture
  - Used for high read and write performances in cloud context

- Stores a list of mapping : id → object

- Structure of object is known by the DB
  - Enables complex queries

Document oriented DB

- MongoDB documents format is BSON : binary JSON
  - JSON + extensions to support binary data into JSON objects

- BSON documents may contain :
  - Nested arrays
  - Nested JSON objects
  - Binary data

Document oriented DB

- To gain the benefits of the noSQL DBs :
  - Documents should be denormalized (duplicated data) when needed
  - Documents should be business oriented : stored data should be useable directly by the application

MongoDB : Replication model

MongoDB Replication

- A single instance of MongoDB might not be enough for an application if :
  - Too many documents to store
  - Lots of concurrent accesses
  - High performances are needed
  - ...
- To solve this problem, several MongoDB instances can be combined to build a storage cluster.

MongoDB Replication

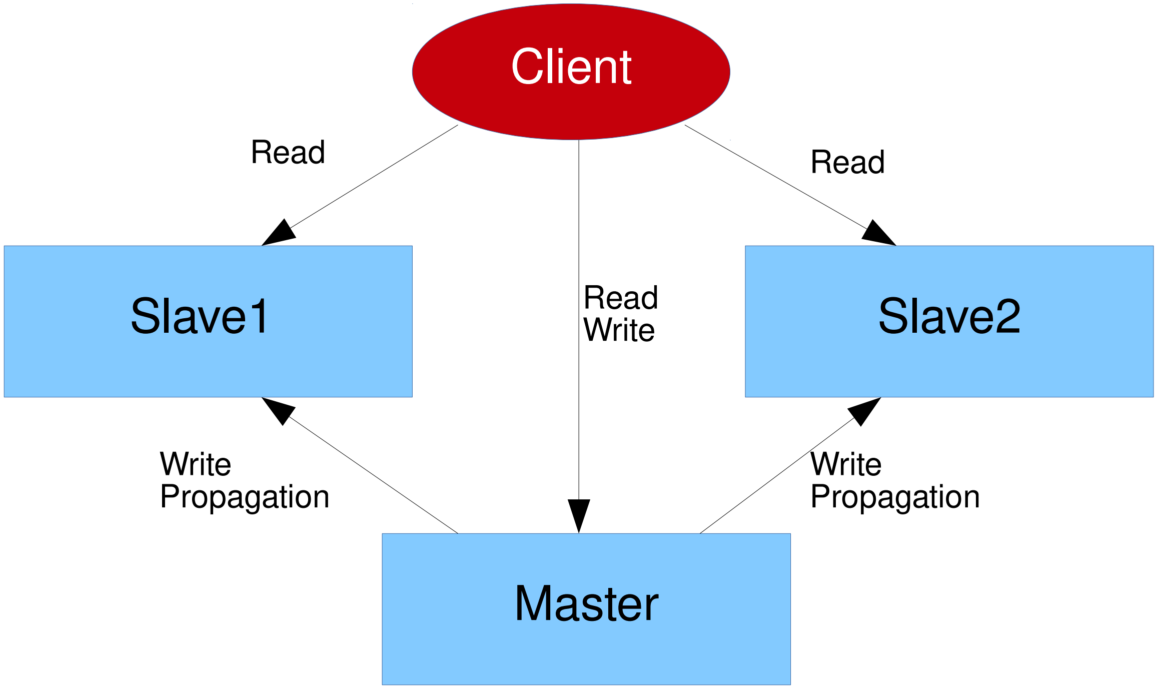
- MongoDB has 2 types of replication :
  - Replica sets
  - Sharding

- These 2 types can be used at the same time to maximize the gain of both choices

MongoDB Replica Sets

- Replica sets are a master slave architecture.
- A replica set has one master instance and may have some slave instances.
- All instances have all the data.
- The slaves are read-only.
- The master accepts writes of new documents and propagates the written info to the slaves.
- To prevent inconsistencies, writing on a document can be blocked if a propagation is in progress.

MongoDB Replica Sets



- Advantages :
  - Fast reads
- Drawbacks :
  - Slow write if write propagations are blocking
  - Read inconsistencies if write propagations are not blocking

MongoDB Sharding

- Sharding is separating the data among different servers or group of servers which are called « shards ».

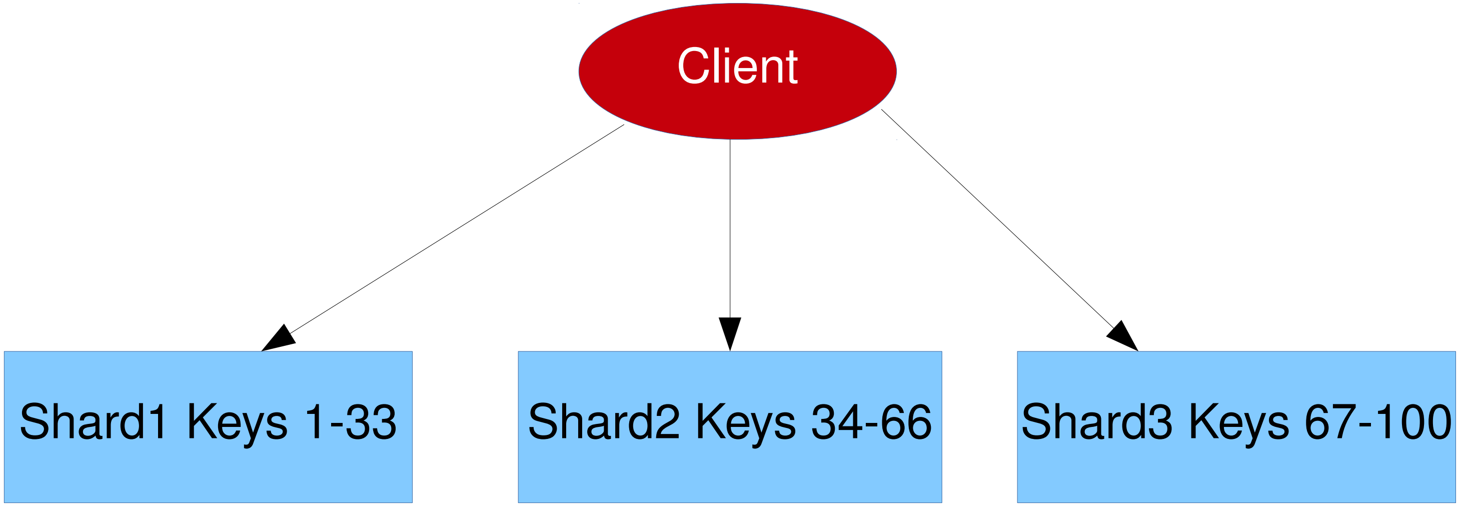
- Each shard only has a part of the data.

- The data can be split according to different criteria :
  - Range-based : based on the key of the documents
  - Hash-Based : based on hash of the keys
  - Tag-aware : based on tags defind by the developper

MongoDB Sharding

- Sharding is efficient if queries implying mutiple documents can be done by clients can be executed inside a single shard.

  Sample of ranged-base sharding with 100 documents :

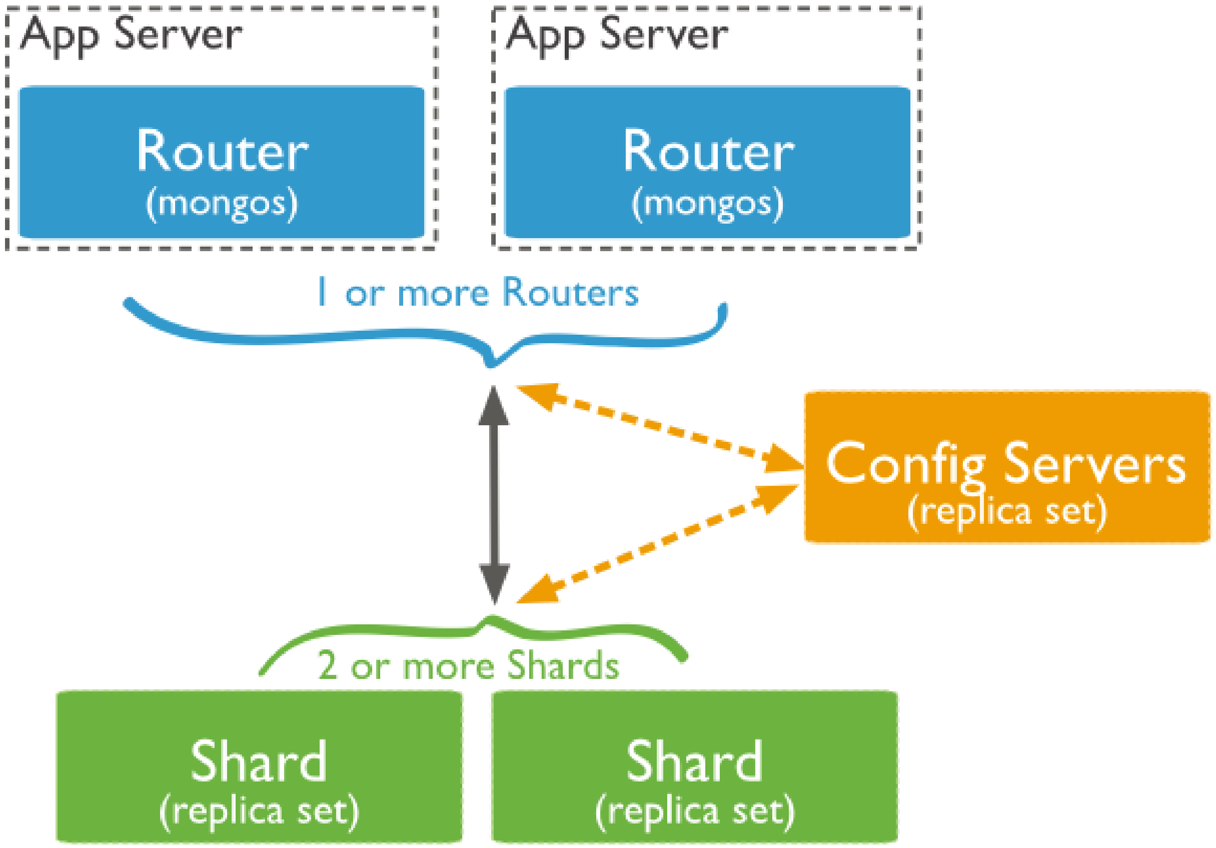


MongoDB : Distributed Achitecture

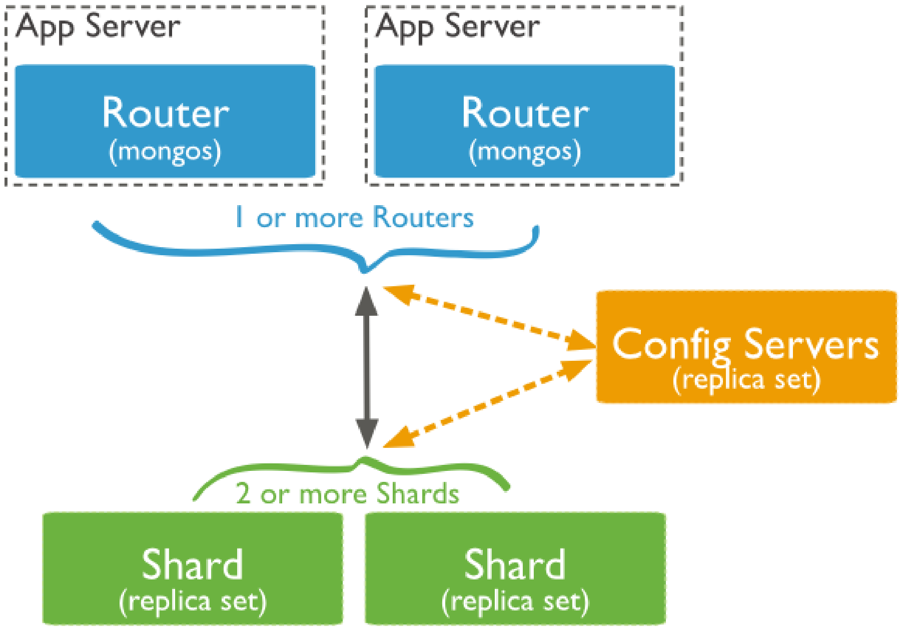
MongoDB distributed architecture

- MongoDB allows to combine Replica Sets and Sharding.
- This brings advantages of both replication architectures.
- The idea is to have multiple shards with each shard being a replica set.

MongoDB : Replica Set + Sharding

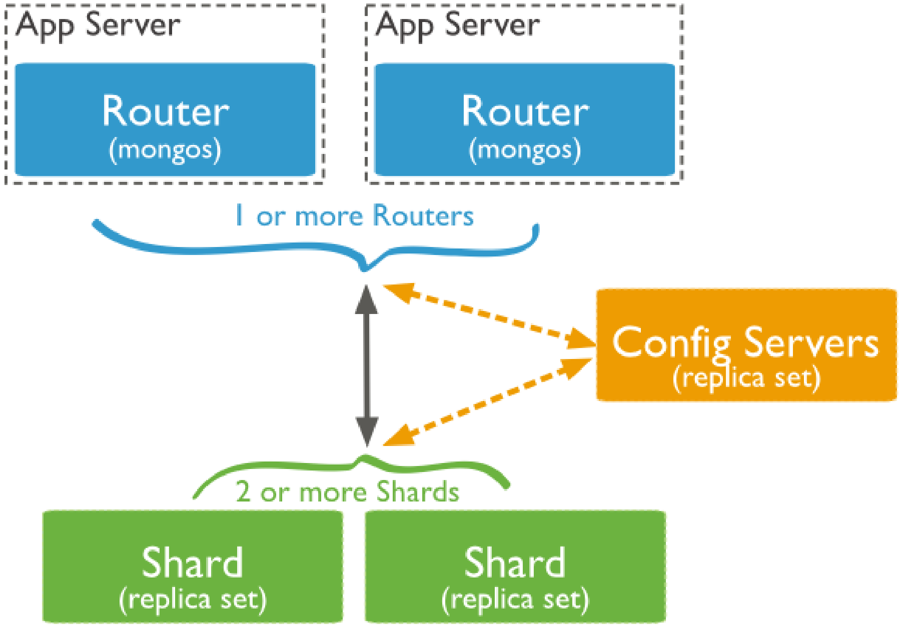


Routers



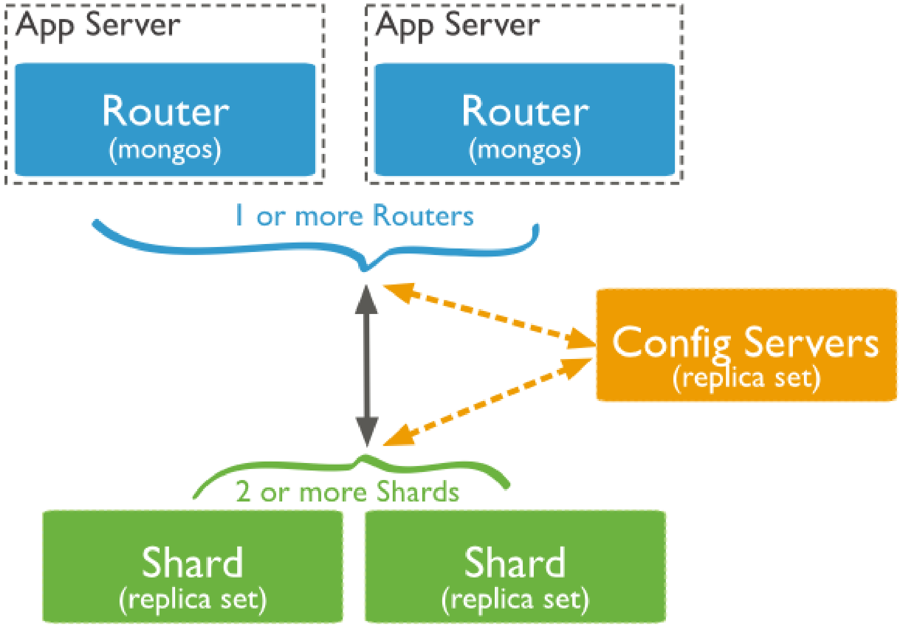
- Routers receive client requests
- Routers forward requests to the shard having the data
- Routers are « mongos » processes

Config servers



- Config servers hold the knowledge of data repartition
- Config servers responds to router : which shard has the wanted document
- Config servers store information received from the shards after writes

Shards



- Each shard is a replica set
- Shards hold the application data
- Shards can be added to increase system capabilities

Achitecture details : Replica Sets

Architecture details : Replica Sets

- A replica set is a group of « mongod » processes
- One of these processes is launched as master of the set
- Other nodes are launched as slaves of this master

Architecture details : Replica Sets

- A replica set must always have a master node

- If the master node crashes :
  - The other nodes should detect it
  - A new master node shoudl be chosen

Architecture details : Replica Sets

- Node crash detection :
  - Nodes send each other « heartbeat » messages every 2 seconds
  - A node not responding « heartbeats » for 10 seconds is marked as inacessible

Architecture details : Replica Sets

- Master re-election :
  - When master crashes, slaves choose a new master among them
  - It may take a few minutes :

    Time for slaves to detect that master node is down

    PLUS

    Time for new master designation

Architecture details : Replica Sets

- Master re-election strategy :
  - Each node has a priority
  - Some nodes can vote
  - All voters try to make the higher priority node the new master

- There can be nodes with priority 0 (non eligible)

- Some nodes cannot vote

Architecture details : Replica Sets

- Priority 0 nodes : nodes which cannot be elected as master
    - Useful for multi data-center architectures
    - Potential system failures if only priority 0 nodes remain active

Architecture details : Replica Sets

- Master election :
  - A replica set has up to 7 voting members
  - Some nodes might not vote during the election
  - The highest priority nodes asks to become master
  - Other nodes vote yes/no according to the last operation available on the candidate
  - Highest priority node is elected OR next highest priority tries to get elected

Architecture details : Replica Sets

- Arbiters nodes :
  - Specific nodes to avoid ties in new master election
  - Arbiters do not store data
  - Arbiters have priority 0 : cannot be master
  - Arbiters should make the number of voters uneven

Architecture details : Replica Sets

- Fault tolerance : Number of node that may crash without breaking the system

There should remain enough nodes to elect a new master

- Fault tolerance is a metric of the system robustness

N° of nodes	N° of nodes for elections	Fault tolerance
3	2	1
4	3	1
5	3	2
6	4	2

Architecture details : Replica Sets

- Hidden replica set member :
  - Hidden members have a copy of the data
  - Hidden members cannot be reached by the client
  - Used for specific purpose : reporting, backup...
  - Hidden members have priority 0 and may vote

Architecture details : Replica Sets

- Delayed replica set member :
  - Delayed members have an outdated copy of the data
  - Delayed members are a running history of the data
  - Used for specific purpose : rollback after human error when performing massive data operations or application upgrades
  - Delayed members should be hidden members

Architecture details : Replica Sets

- Rollbacks :
  - Rollbacks might be necessary on master failure :

    If write operations have sucessed on master

    AND

    These operations have not been replicated to slaves

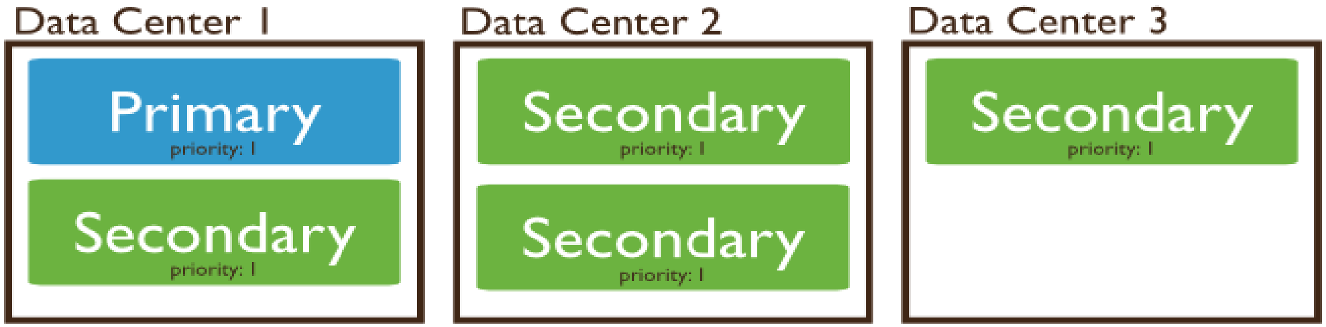
  - Rollback reverts the non propagated write operations on the failed master

Architecture details : Replica Sets

- Multi Data Center architecture :

  - Splitting replica set nodes across multiple data center allows to resist data center failures

  - Usual setups are :
    - 3 nodes across 2 data centers
    - 5 nodes across 3 data centers



Achitecture details : Sharding

Architecture details : Sharding

- Shard key :
  - Objects are distributed among the shards according to their shard key
  - Shard key cannot be changed once the shard has been created
  - Choice of the shard key determines the efficiency of the sharding

Architecture details : Sharding

- Goal of the shard key:
  - Splitting documents evenly between shards
  - Distribute documents so that read and write are set evenly to the shards

Architecture details : Sharding

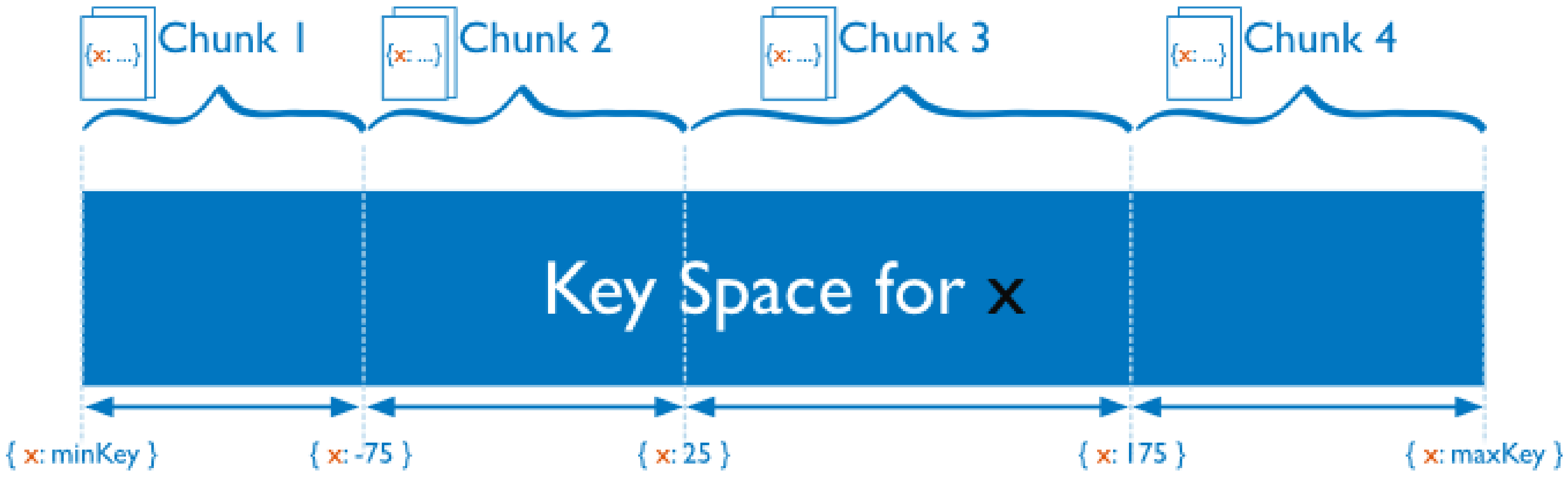
- Shard key important properties :
  - Cardinality : number of possible values for the shard key
  - Frequency : how often a shard key value appears in documents

These properties and the evolution of the shard key value will result in an efficient or inefficient sharding.

Architecture details : Sharding

- Sharding strategy : range based

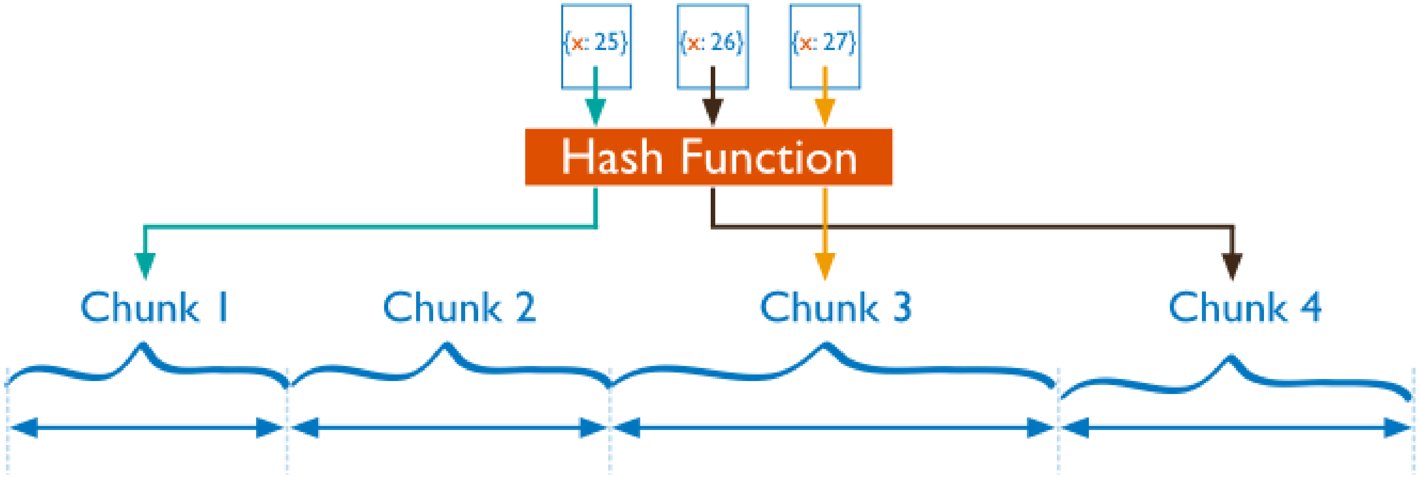
Shard keys are divided into ranges, and each range is associated to a shard



Architecture details : Sharding

- Sharding strategy : hash based

A hash is computed for each shard key. Each range of hashes is associated to a shard

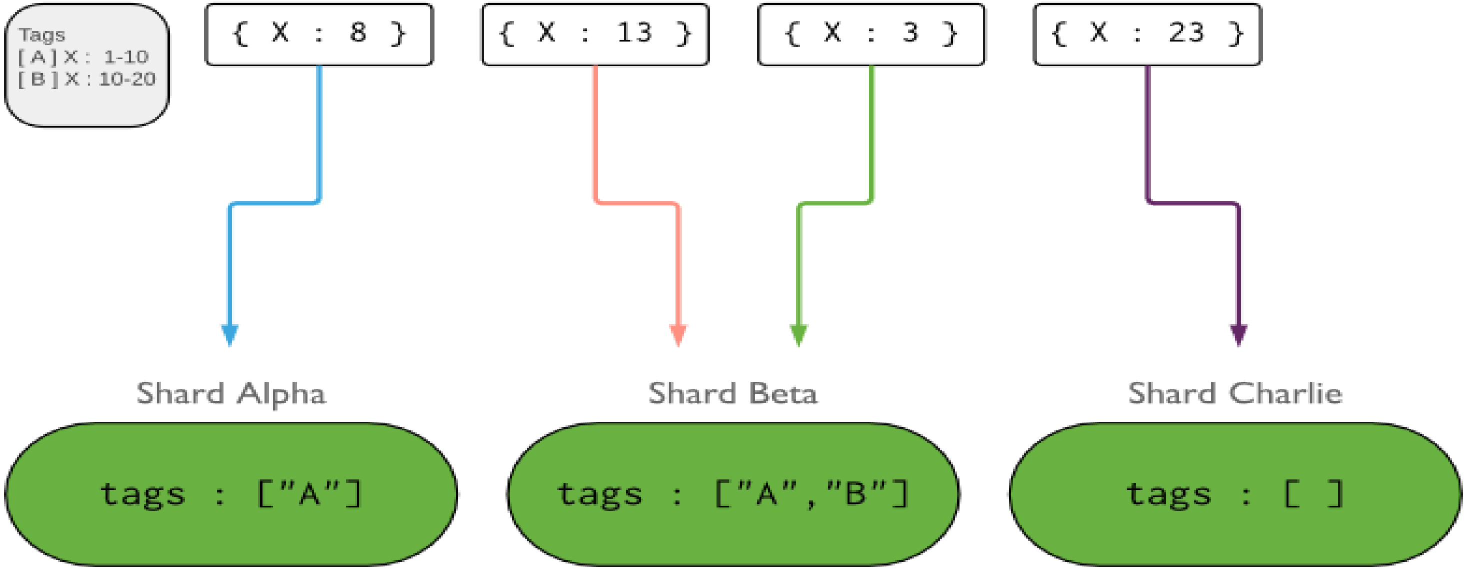


Architecture details : Sharding

- Sharding strategy : tag aware

Tags can be defined on shard keys ranges. These tags can be associated to one or more shards

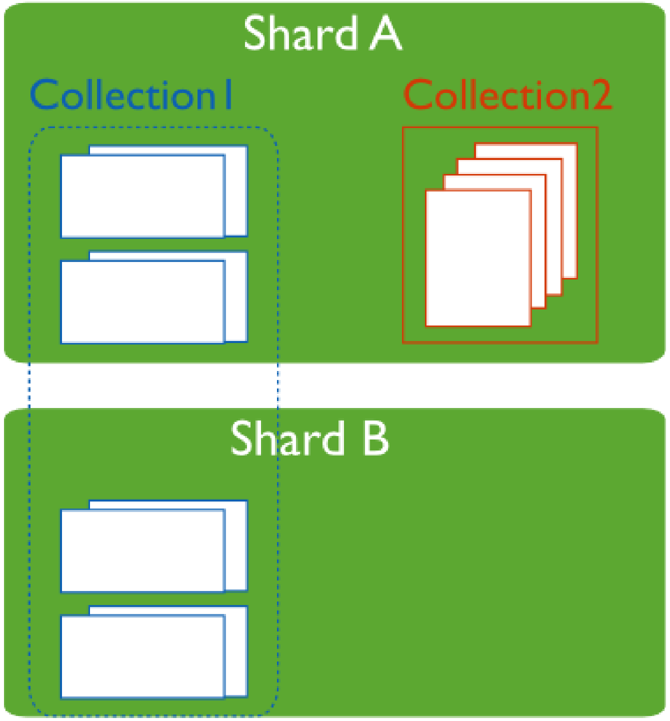
This allows to have a custom strategy for the shard repartition



Architecture details : Sharding

- Sharding is at collection level :

Within the same mongo cluster, there might be sharded and not sharded collection



Xin cảm ơn !

LINAGORA – Việt Nam

Số 59A Lý Thái Tổ,
phường Tràng Tiền,
quận Hoàn Kiếm, thành phố
Hà Nội

Điện thoại : +84(0) 906 220 340

Info : vietnam@linagora.com

Web : vn.linagora.com

@LinViet

facebook.com/linagoravietnam

LINAGORA VIETNAM