

MongoDB教育訓練-20211223-03

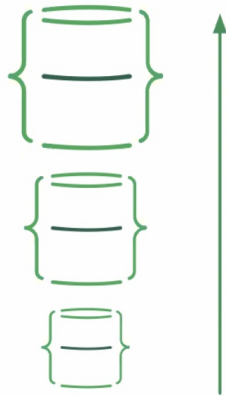
vertical scaling 垂直擴展(升級到某個程度，會很貴並有極限)

horizontal scaling 水平擴展 (比較省成本)

Vertical versus Horizontal Scaling

Vertical Scaling

Increase size of instances
(RAM, CPU etc)



Horizontal Scaling

Add more instances



沒有一定要shard(如果資料不多不需要，管理複雜度高)

Case Study

- In 2014 a fitness app company had a MySQL server with 750GB of RAM
- Expanding to 2TB of RAM as they grew was disproportionately expensive
- They changed to 4 MongoDB shards, each with 512 GB
- They have subsequently expanded beyond this significantly

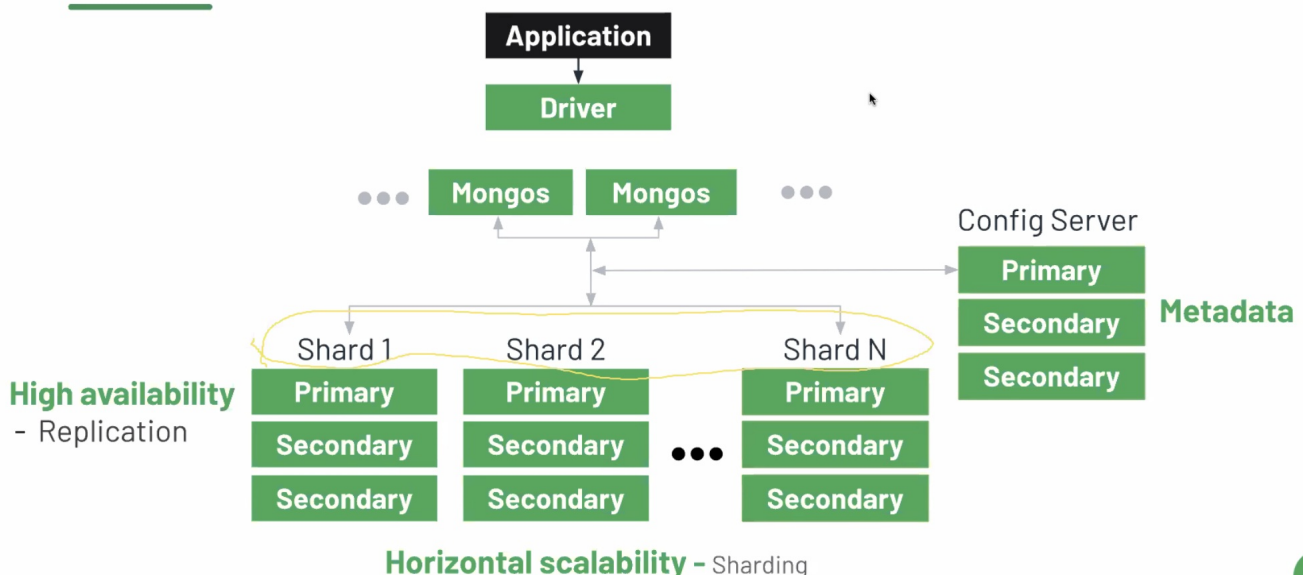
When do you need to shard

- You don't **need** to shard until you have a sizeable problem.
 - It's a solution to Big Data problems
 - Big data can be simply an aspect of many users.
 - The exact size differs based on the use case.
 - Vertical scaling is much easier up to a point.
 - How to tell if you need to Shard.
 - You have a resource maxed out (and you know why)
 - You have already optimized your schema and code
 - No affordable update to the current server will resolve it
 - Or you need to meet a backup restore time (RTO) target and need parallelism to do so.
-
- why learn about sharding now?
 - you might not need to shard for years
 - but decisions you make about schema design before that matter
 - start planning for sharding on day one

Why learn about sharding now?

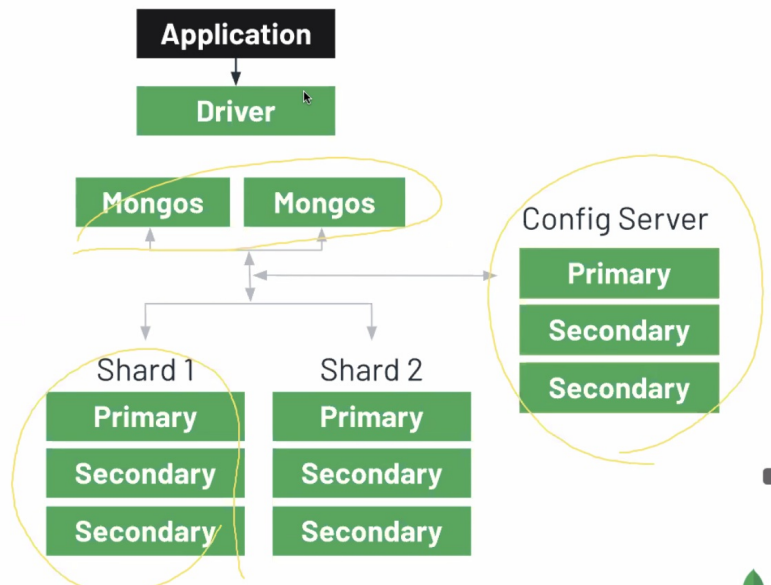
- You might not need to shard for years.
- But decisions you make about schema design before that matter.
 - Start planning for sharding on day one.

Sharding Architecture



Sharding uses additional hardware

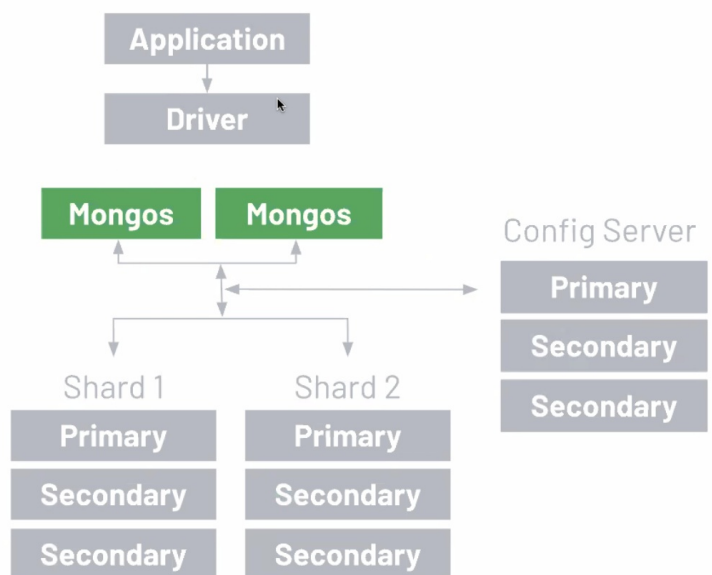
- Sharding adds config (metadata) replica set (3 nodes)
- You need ideally at least 2 query routers.
- Then the replica sets holding data.



mongos 屏蔽所有細節

mongos

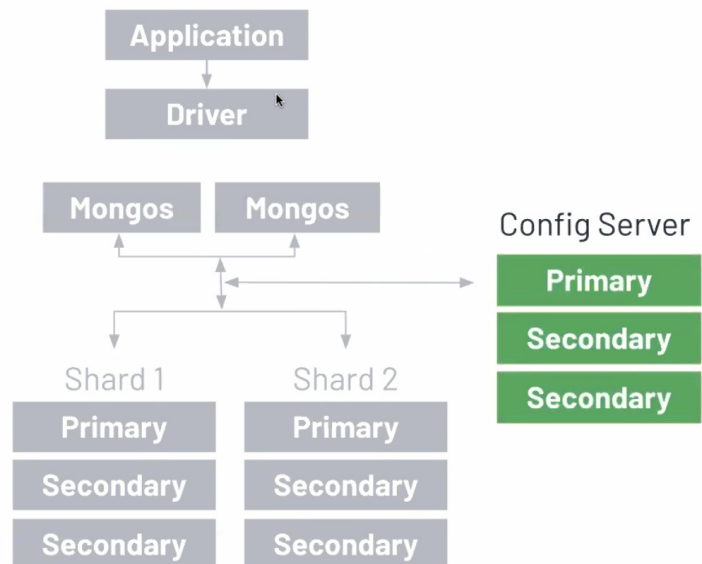
- Proxy server - behaves like a MongoDB database but routes requests instead
- Only sends work to places it needs to
- It needs no local storage
- It needs a good network connection.
- What you always connect to in sharding.



configuration server 決定資料分布

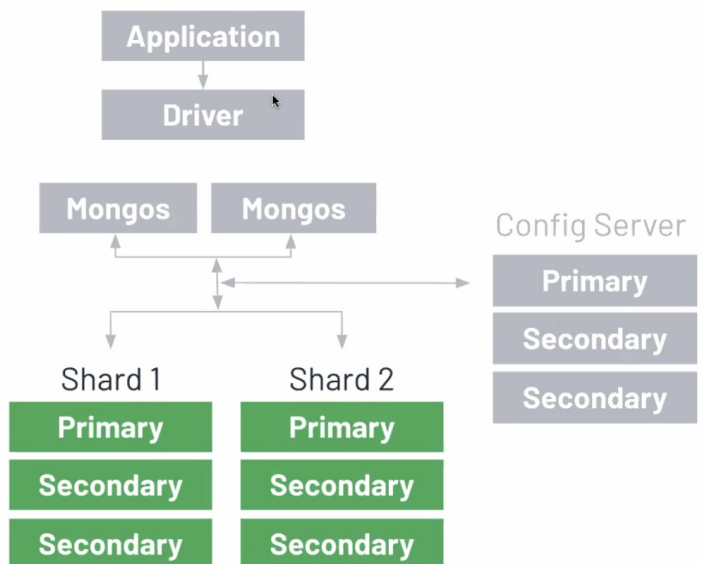
Configuration Servers

- Config Replica Set
- Holds metadata about users, partitioning, and data locations.
- Metadata is visible as a database called 'config' via mongos.
- Always a replica set - Why?

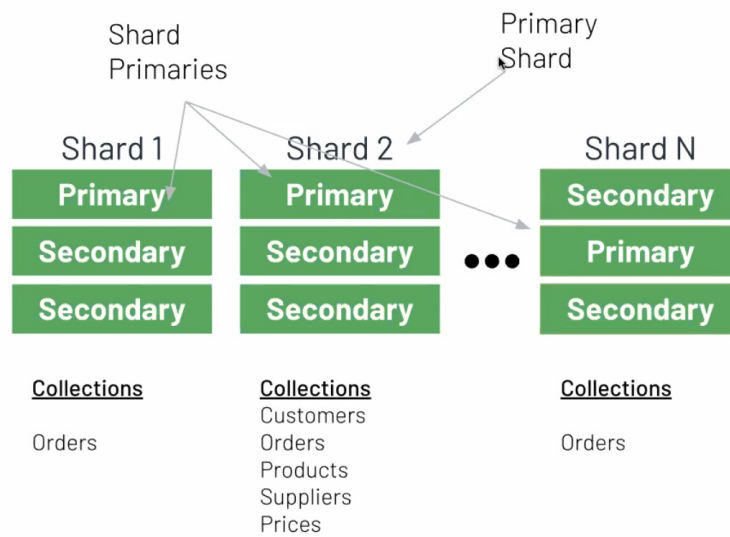


Shard Servers

- **Each shard must be a replica set.**
- One shard isn't useful, from two to thousands of shards are.
- Each will hold a subset of your largest collections.



Primary Shards and Shard Primaries



Shard Keys

- Shard Key means two things.
 - What fields are chosen to partition data.
 - What the value of those fields is in a given document.

修改分片鍵

Choosing Shard Keys

- You need to select what fields to use to locate the data
 - You cannot change this afterward in MongoDB before 4.4
 - You can add another field to refine it from version 4.4
 - Version 5.0 lets you change your shard key and will reorganize the data.
- Rules for choosing a good shard key
 - Choose something that is included in most of your queries
 - Choose something with a reasonably high cardinality
 - Ideally you want no more than 64MB of data to share a shard key
 - Choose something that will co-locate data you wish to retrieve together

選擇分片鍵原則

shard keys 不一定要一樣的，重複的不可超過64MB

Choosing Shard Keys

- You need to select what fields to use to locate the data
 - You cannot change this afterward in MongoDB before 4.4
 - You can add another field to refine it from version 4.4
 - Version 5.0 lets you change your shard key and will reorganize the data.
- Rules for choosing a good shard key
 - Choose something that is included in most of your queries
 - Choose something with a reasonably high cardinality
 - Ideally you want no more than 64MB of data to share a shard key
 - Choose something that will co-locate data you wish to retrieve together

Choosing shard keys in practice

Typically, it isn't challenging to choose a shard key; however, many who do so give it no thought and then have problems later.

- You only shard large collections.
- In large collections
 - Users work with obvious subsets of the data
 - If it's 'their' data - shard by a user, for example, bank accounts or games
 - If it's departmental - shard by department or branch
- Where there are no clear subsets
 - Shard for parallelism
 - For instance, in an analytic data store, add a random value for the shard key.

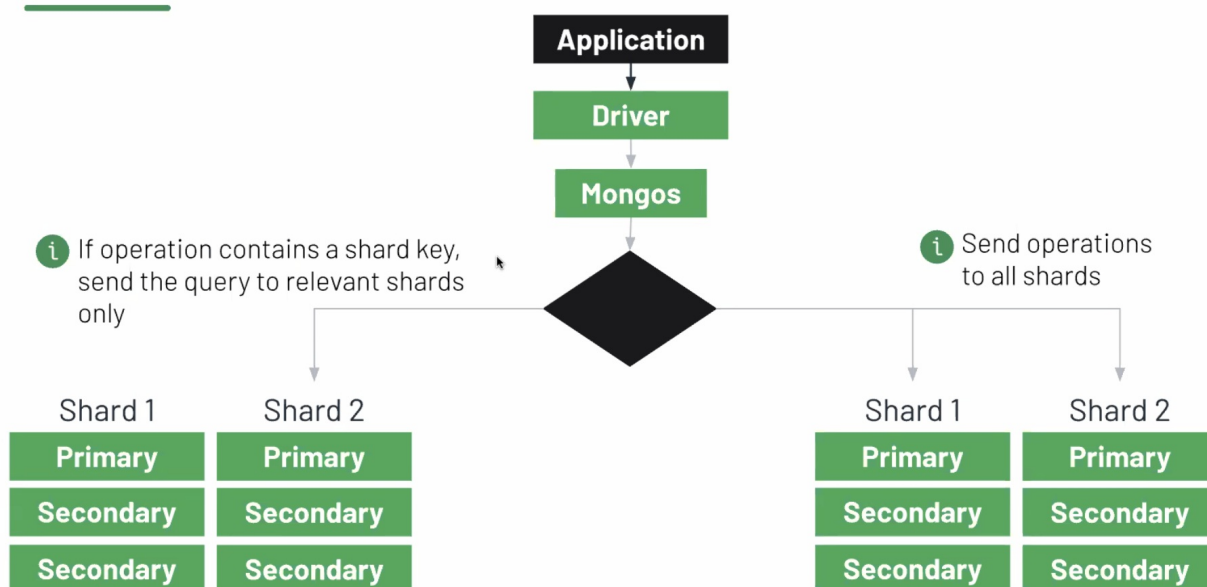


How to Shard a Collection in 4 steps.

1. Ensure you have a sharded cluster - use `sh.status()` to check.
2. Decide on your shard key - this is the hardest part.
3. Configure the **database** for sharding - `sh.enableSharding("MyGameDB")`
4. Specify the shard key for the **collection** -
`sh.shardCollection("MyGameDB.players", {playerid:1, gametime:1 })`

Ops and Cloud Manager tooling give you a GUI way to do this.

How Sharding works - distributing operations



How config metadata caching works

- Driver sends operation to mongos
- mongos has a cached copy of config data which is versioned
 - mongos uses this to target relevant shards with the operation
- The operation is sent along with the version of the config known by mongos
- If mongos cache of config is an older version than the replica set's knows
 - Replica Set ignores operation and tells mongos to refresh it's cache
 - mongos does so, then re-sends operation to the correct servers.

Sharded Operations

- When you have a good shard key
 - Most operations will target a single shard or a few shards.
 - An individual user will mostly hit a single shard.
- This is more efficient than every query going to all shards
 - Less work done
 - Lower average latency
- An operation that cannot target specific shards is called "broadcast"
- If a shard is down - all operations that touch it will fail completely.
 - So all scatter gather queries fail.
 - A good shard key means if a shard is down, only some operations fail
 - Ideally, only a subset of users are affected by a shard being down.

Group Exercise - Choosing Shard Keys

What fields would you suggest to be shard keys in the following and why?

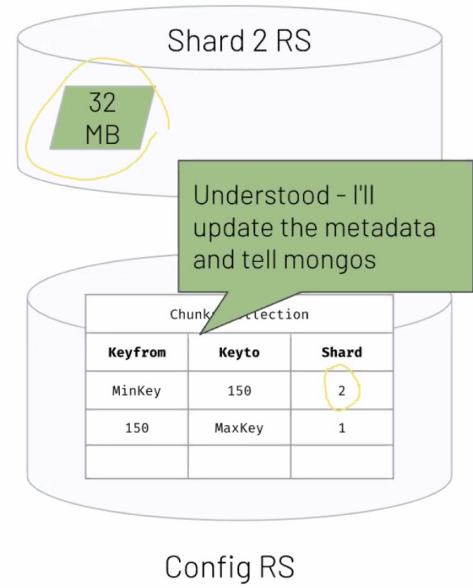
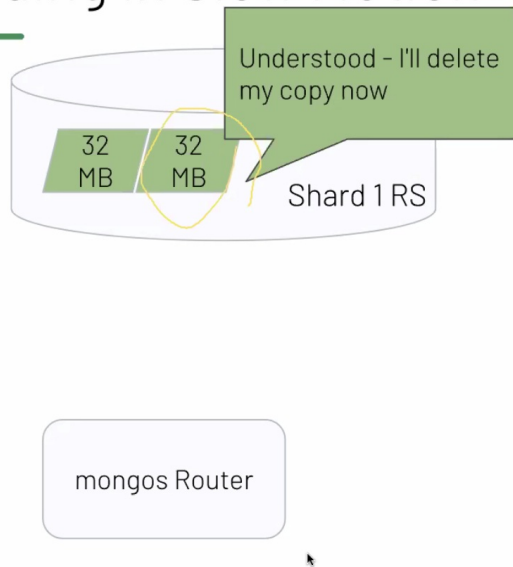
- A web-based email service like Gmail.
- A government database of businesses and their directors.
- A stock database for a national chain of electronic stores.
- A customer accessible product catalog like Amazon.

Chunks

- A Chunk is a term used to refer to all documents where the shard key is in a given range of values.
- Each Chunk exists on a single shard.
- If a shard key falls into the range, then it is set to be in the Chunk.
- Data in each range is always inclusive of its lower boundary and exclusive of its upper boundary.
- The reads/writes are routed to a shard hosting the specific Chunk.



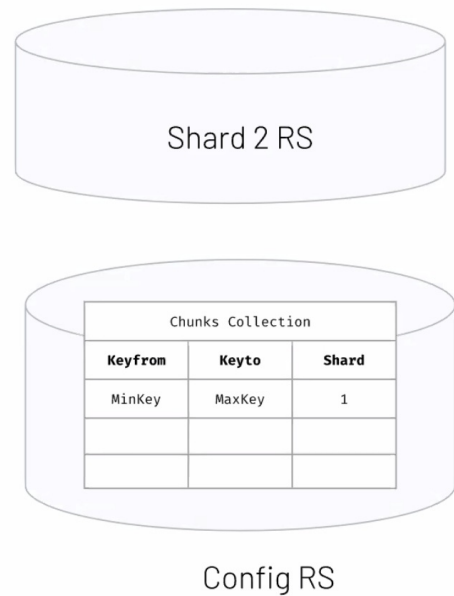
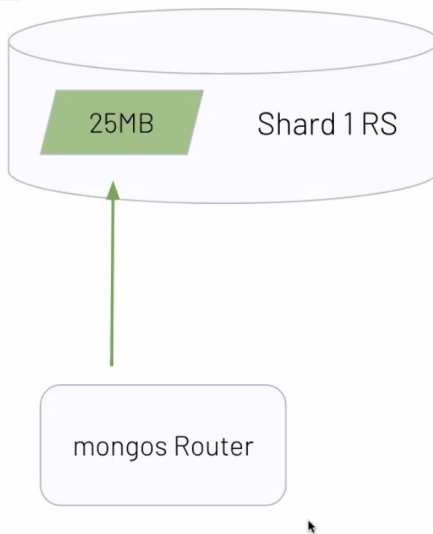
Sharding in Slow Motion



pyright 2020-2021 MongoDB, Inc. All rights reserved.

Slide 35

Sharding in Slow Motion



pyright 2020-2021 MongoDB, Inc. All rights reserved.

Sharding Pitfalls

- Having to balance this is not very efficient
 - Each chunk moved is being written three times (write on 1, write on 2, delete on 1)
 - In a busy system getting the deltas can take time (more reads and writes)
 - Once chunks are distributed, as long as the writes hit random chunks, it's good.
 - But it can take a while to balance first.
- What if we are inserting and shard key is increasing?
 - All writes go to same shard { sk: x } -> { sk: maxKey }
 - 50% of chunks are then moved as every other chunk gets copied to shard 2
 - There needs to be 3X number of writes!
 - You could shard by the md5 hash of the shard key instead for random distribution.

預分片

Presplitting before loading

- To avoid the initial shuffling of data via the first shard we can presplit
- We explicitly split the chunk before it has any data in.
- We can then either let the balancer move the empty chunks...
- Or we can explicitly move them which may be faster.
- Doing this correctly can greatly speed up a bulk load of data.

範例 1000億筆資料

Presplitting before loading

- To avoid the initial shuffling of data via the first shard we can presplit
- We explicitly split the chunk before it has any data in.
- We can then either let the balancer move the empty chunks...
- Or we can explicitly move them which may be faster.
- Doing this correctly can greatly speed up a bulk load of data.

YCSB 壓測工具

Alex tries YCSB

- Someone explains to Alex that they need:
 - A multi-threaded load generator
 - Something faster than Javascript
 - To load data in batches not one document at a time.
- So Alex tries again with YCSB a well known NoSQL Benchmark tool.
- This time a single Replica set manages 40,000 per second
 - Alex is *amazed*! 3,000 seemed good.
- However, the three shards manage just 20,000 between them!
 - Alex goes away and checks stack overflow.
 - Alex sees this happens if all writes go to one chunk due to an increasing shard key. Stack overflow recommends using a hashed shard key.

© 2020 MongoDB Inc. All rights reserved.

Downsides of hashed indexes

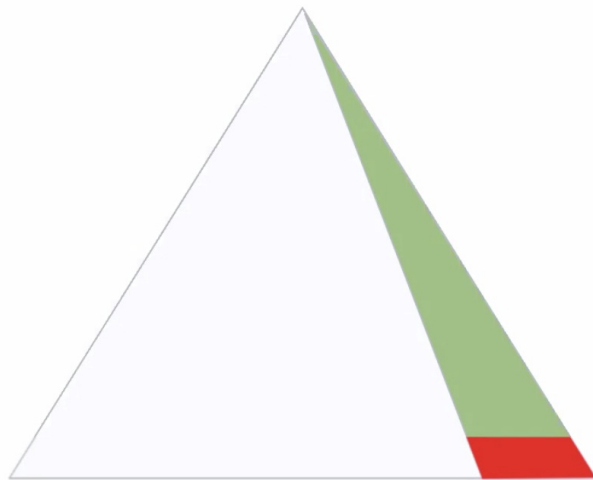
The index is a BTree with internal and leaf nodes.

White - data can be on disk.

Green - frequently read, some writes, should be in RAM.

Red - frequently written, needs Disk IO.

With an unhashed index an increasing key goes to the right of the tree.



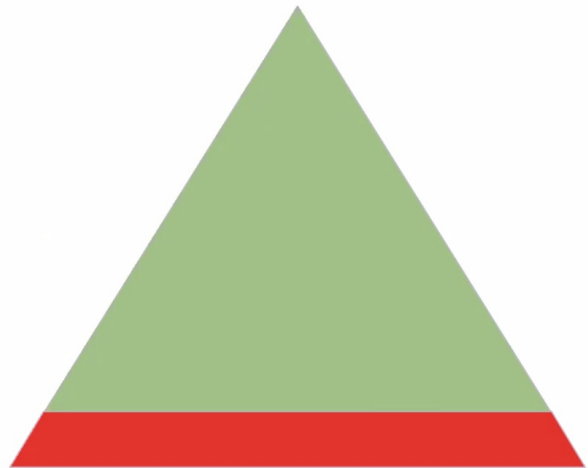
Downsides of hashed indexes

A random key (hashed value) could go anywhere in the index.

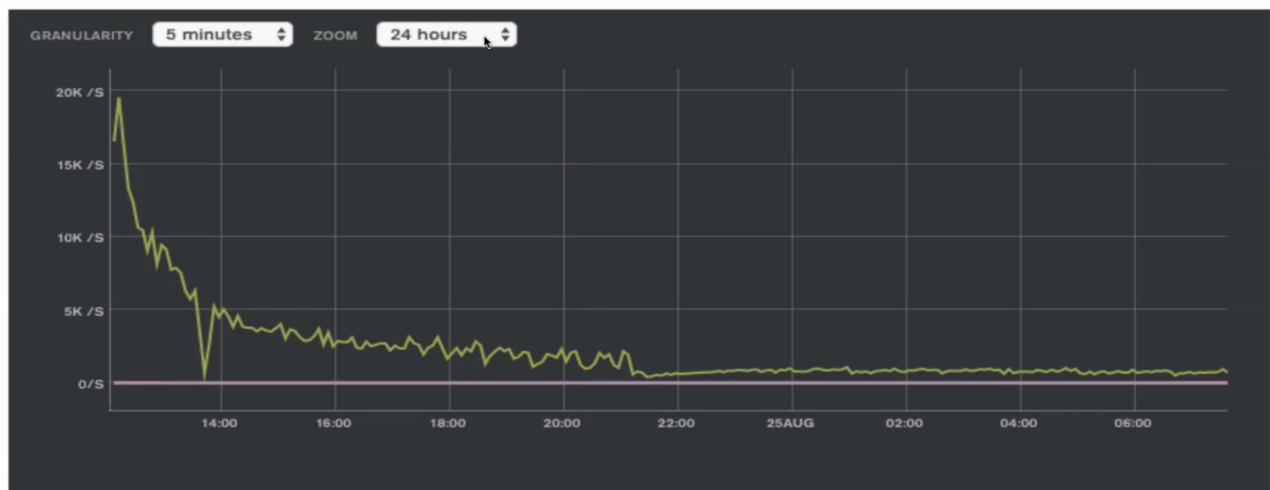
The whole index is being accessed continuously, so it needs RAM

The maximum number of disk block are being dirtied.

So simple hashed shard keys are not a good idea at scale.



The impact of hashed sharding



primary shared 最忙

Common sharding challenges

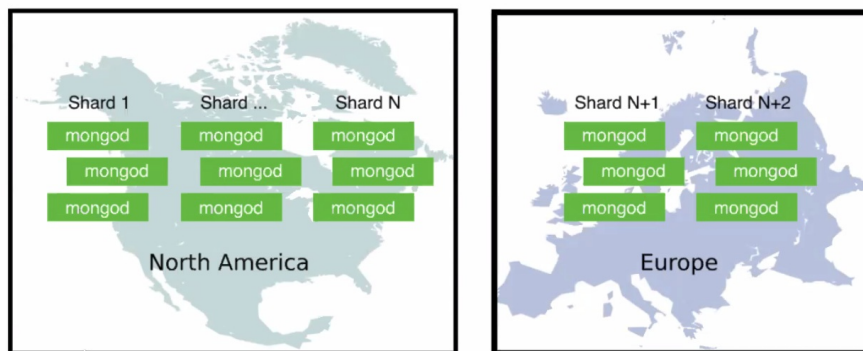
- Our data may be badly distributed
 - The cluster will eventually balance volume as long as there is a reasonable cardinality
 - However we can get 'Hot' shards doing a lot work relative to others.
 - Hashed sharding avoids these hot shards but takes more resources overall.
 - Our obvious shard key may be a random value anyway (e.g. An MS GUID) and therefore take too many resources.
- Scaling out a cluster can temporarily take a lot of resources
 - Just when we are running out of resources and need to scale out!
 - It's simple to add another shard or two, but those are initially empty.
 - The balancer is responsible for copying data over to them.
 - It is far from instant - it can take days or weeks as lots network and I/O !

完全停止

Managed Sharding: How to scale painlessly

- Managed sharding is an expert level technique to avoid sharding pitfalls.
 - It requires planning and effort, but sharding means we are already in big data territory.
 - It requires code changes in our application
 - A little effort can have enormous benefits.
- Managed sharding means.
 - We never need to enable or use the balancer, data is always balanced.
 - Our indexes do not slow down as they grow.
 - We can provision new servers instantly.
 - We can age out older data cheaply onto smaller hardware.

Zone based Sharding



分散多筆sharding，利用換取CPU計算

Sharding for Parallelism

Unusual use case but powerful when used.

- Running Aggregations and some queries - reads sent to all shards
- Bad for OLTP / Normal queries
- But for a small number of analytics queries - the more CPUs the better
- Sometimes we run many shards on one server! (microsharding). This works where:
 - (a) All data can fit in RAM and therefore disks not a bottleneck
 - (b) You have a small number of power users - more CPUs than users
 - (c) You know how to write aggregations that can use parallelism.

🕒 修訂版本 #3

★ 由 treeman 建立於 23 🕒🕒🕒🕒 2021 22:29:59

🔧 由 treeman 更新於 5 🕒🕒🕒🕒 2023 10:14:59