马哥教育

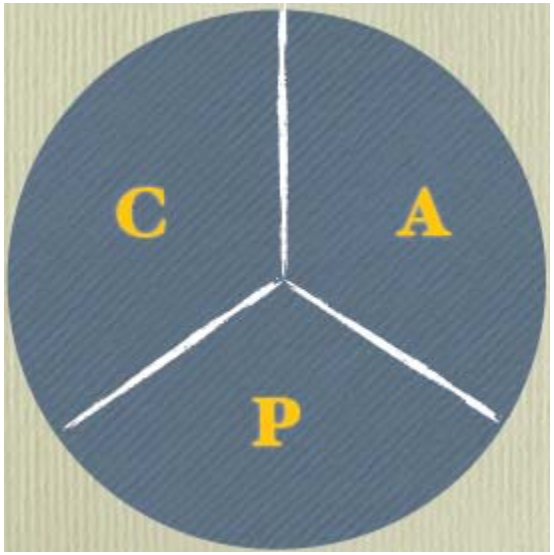
主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

Agenda

- 分布式系统理论
  - CAP
  - BASE
- NoSQL
  - 数据存储模型
- MongoDB
  - 安装
  - CRUD
  - 索引
  - 副本集(replica sets)
  - 分片(sharding)

- NoSQL
  - 1998，NoREL
  - 2009，NoSQL
    - 非关系型
    - 分布式
    - 不提供ACID
- NoSQL
  - 简单数据模型
  - 元数据和应用数据分离
  - 弱一致性
- 优势
  - 避免不必要的复杂性
  - 高吞吐量
  - 高 水平扩展能力和低端硬件集群
  - 不使用对象-关系映射
- 劣势
  - 不支持ACID
  - 功能简单
  - 没有统一的数据查询模型

CAP Theorem



- Pick two
  - Consistency
  - Availability
  - Tolerance to network Partitions

ACID & BASE

- Atomicity
- Consistency
- Isolation
- Durability

- Basically Available
- Soft state
- Eventually consistent

ACID vs. BASE

ACID

- Strong consistency
- Isolation
- Focus on "commit"
- Nested transactions
- Availability?
- Conservative
- Difficult evolution (schema)

BASE

- Weak consistency
- Availability first
- Best effort
- Approximate answers
- Agressive (optimistic)
- Simpler!
- Faster
- Easier evolution

Database Paradigms

- Relational (RDBMS)
- NoSQL
  - Key-value stores
  - Document databases
  - Wide column stores (BigTable and clones)
  - Graph databases

Relational Databases

- ACID (Atomicity Consistency Isolation and Durability)
- SQL
- MySQL, PostgreSQL, Oracle, etc.

What is No-SQL

- Non-Relational
- Distributed
- Open-Source
- Horizontally
- Schema-Free
- Replication Support
- Simple API
- Eventually Consistent

Key-value stores

- "One key, one value, no duplicates and very fast"
- It's a Hash!
- The value is a binary object aka "blob" – the DB doesn't understand it and doesn't want to understand it.
- Amazon Dynamo, MemcacheDB, etc.

Document databases

- Key-value stores, but the value is (usually) structured and "understood" by the DB.
- Querying data is possible (by means other than just a key).
- Amazon SimpleDB, CouchDB, MongoDB, Riak, etc.

Why NoSQL?

- Schema-free
- Massive data stores
- Scalability
- Some services simpler to implement than using RDBMS
- Great fit for many Web 2.0 applications

Why NOT NoSQL?

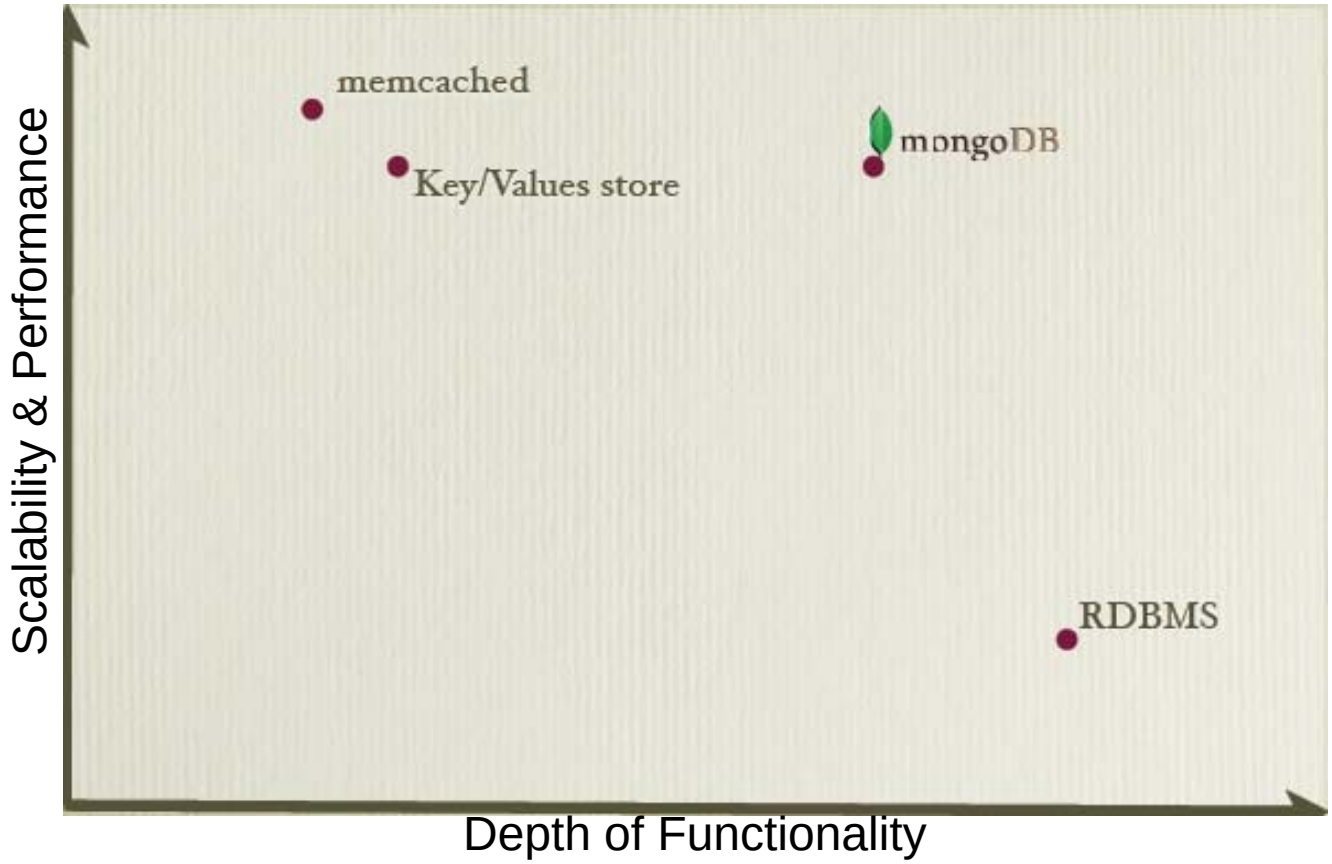
- RDBMSes and its tools are mature
- NoSQL implementations are often in their "alpha" state
- Data consistency, transactions
- "Don't scale until you need it"

RDBMS vs NoSQL

- Strong consistency vs Eventual consistency
- Big dataset vs HUGE dataset
- Scaling is possible vs Scaling is easy
- Good availability vs Very high availability

Introduction to MongoDB

- MongoDB (from "humongous") is a scalable, high-performance, open source, schema free, document no-sql oriented database



What is MongoDB?

- Humongous (huge + monstrous)
- Document Database
- Schema free
- C++
- Open Source
- GNU AGPL v3.0 Licence
- OSX, Linux, Windows, Solaris | 32 bit, 64 bit
- Development and Support by 10gen and was first released in February 2009
- NoSQL!

What is MongoDB?

- Document-oriented database
  - Uses JSON (BSON actually)
- Schema-free
- Performance
  - Written in C++
  - Full index support
  - No transactions (has atomic operations)
  - Memory-mapped files (delayed writes)
- Scalable
  - Replication
  - Auto-sharding
- Commercially Supported (10gen)
  - Lots of documentation

What is MongoDB?

- Document-based queries
  - Flexible document queries expressed in JSON/Javascript
- Map/Reduce
  - Flexible aggregation and data processing
  - Queries run in parallel on all shards
- GridFS
  - Store files of any size easily
- Geospatial Indexing
  - Find object based on location. (i.e. find closes n items to x)
- Many Production Deployments

Features

- Collection oriented storage: easy storage of object/JSON -style data
- Dynamic queries
- Full index support, including on inner objects and embedded arrays
- Query profiling
- Replication and fail-over support
- Efficient storage of binary data including large objects (e.g. photos and videos)
- Auto-sharding for cloud-level scalability (currently in alpha

Great for

- Websites
- Caching
- High volume, low value
- High scalability
- Storage of program objects and json

Not as great for

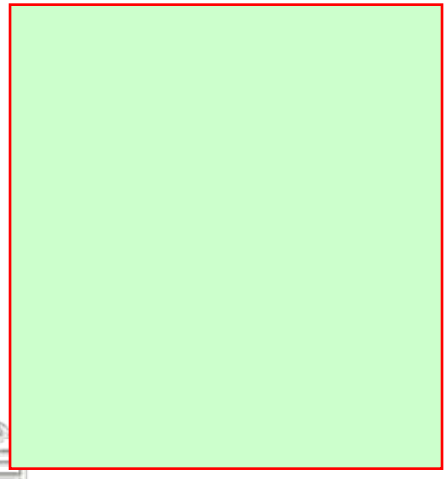
- Highly transactional
- Ad-hoc business intelligence
- Problems requiring SQL

马哥教育

Installation

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

- 面向collection的数据库
  - 数据库：但数据库无须创建
  - 表：行 ← → 集合：文档
  - 集合无须事先定义；



马哥教育

MongoDB CRUD 快速入门

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

- 客户端工具mongo
- MongoDB CRUD Operations

获取帮助

- db.help()
  - help on db methods
- db.mycoll.help()
  - help on collection methods
- sh.help()
  - sharding helpers
- rs.help()
  - replica set helpers
- help admin
  - administrative help
- help connect
  - connecting to a db help
- help keys
  - key shortcuts
- help misc
  - misc things to know
- help mr
  - mapreduce

显示类命令

- show dbs
  - show database names
- show collections
  - show collections in current database
- show users
  - show users in current database
- show profile
  - show most recent system.profile entries with time >= 1ms
- show logs
  - show the accessible logger names
- show log [name]
  - prints out the last segment of log in memory, 'global' is default

MongoDB CRUD Operations

- MongoDB provides rich semantics for reading and manipulating data
- CRUD stands for create, read, update, and delete

MongoDB Documents

- MongoDB stores data in the form of *documents*, which are JSON-like field and value pairs
- Documents are analogous to structures in programming languages that associate keys with values, where keys may hold other pairs of keys and values (e.g. dictionaries, hashes, maps, and associative arrays)
- Formally, MongoDB documents are BSON documents, which is a binary representation of JSON with additional type information

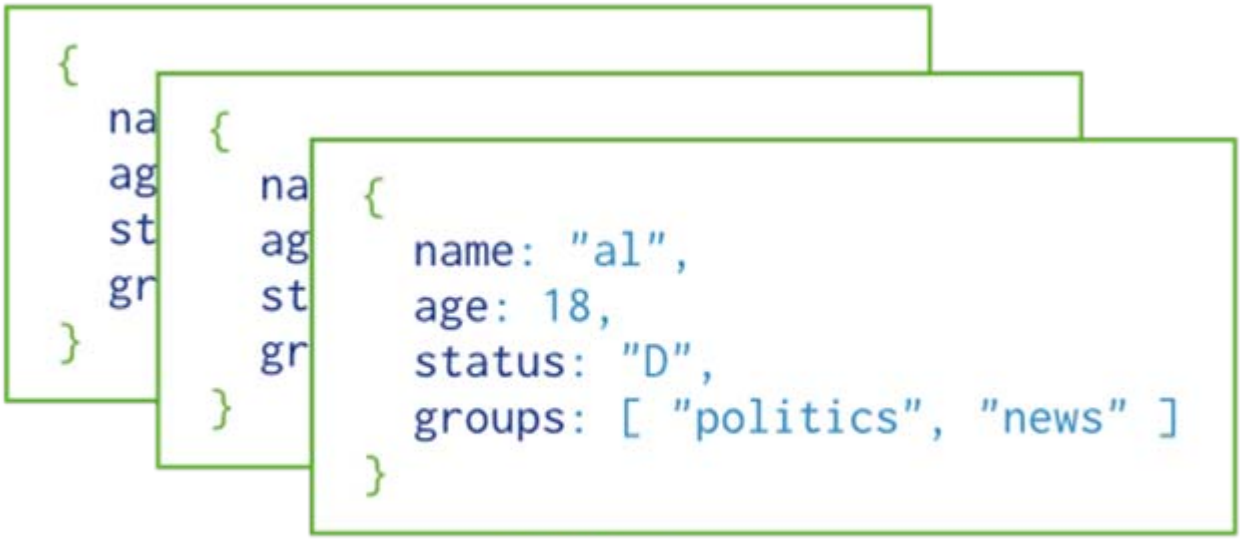

```
{
  name: "sue",          <-- field: value
  age: 26,              <-- field: value
  status: "A",          <-- field: value
  groups: [ "news", "sports" ]   <-- field: value
}
```

Document

- Stored in collection, think record or row
- Can have _id key that works like primary key in MySQL
- Two options for relationships: subdocument or db reference

MongoDB Collections

- MongoDB stores all documents in *collections*
- A collection is a group of related documents that have a set of shared common indexes
- Collections are analogous to a table in relational databases



Collection

- Think table, but with no schema
- For grouping into smaller query sets (speed)
- Each top entity in your app would have its own collection (users, articles, etc.)
- Full index support

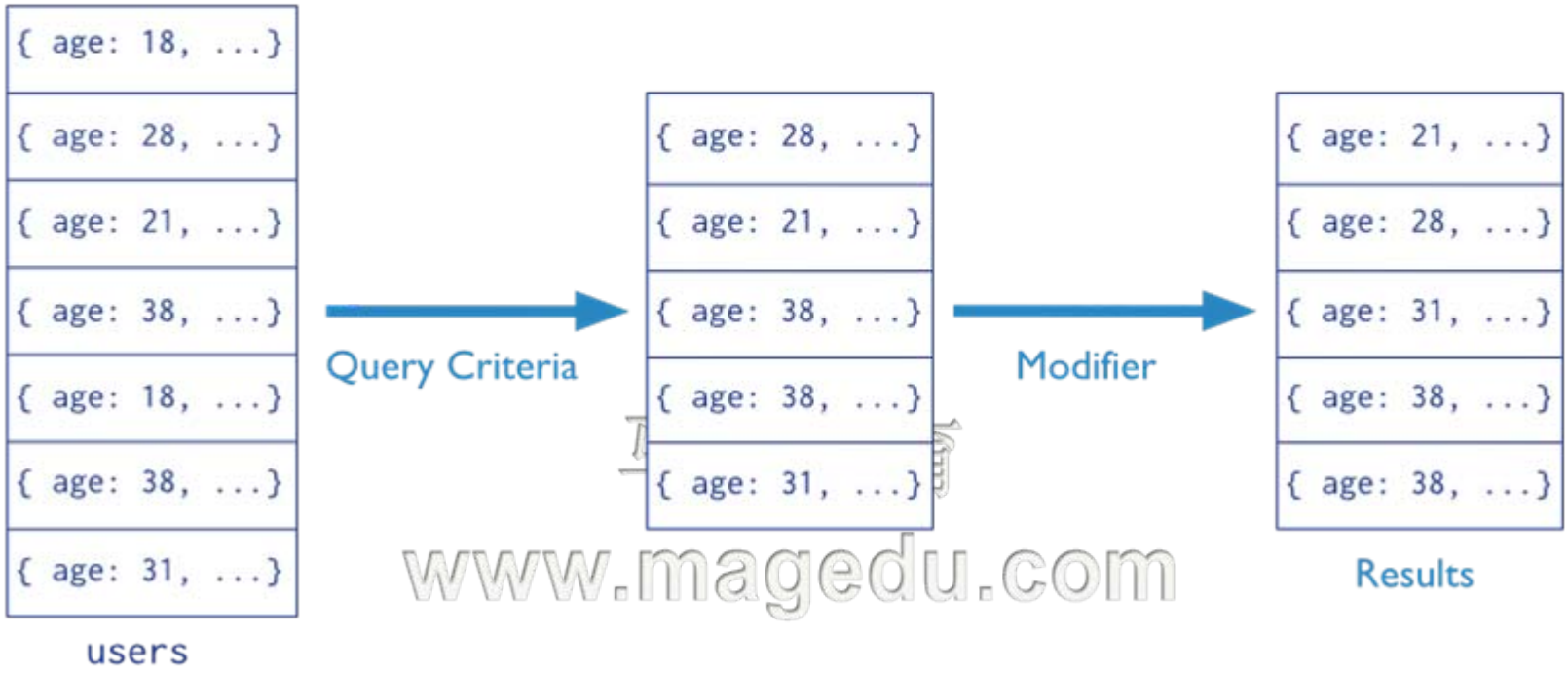
Database Operations: Query

- In MongoDB a query targets a specific collection of documents
- Queries specify criteria, or conditions, that identify the documents that MongoDB returns to the clients
- A query may include a projection that specifies the fields from the matching documents to return
- You can optionally modify queries to impose limits, skips, and sort orders

Database Operations: Query

Collection　　Query Criteria　　Modifier

```
db.users.find( { age: { $gt: 18 } } ).sort( {age: 1 } )
```



Query Interface

- For query operations, MongoDB provide a db.collection.find() method
- The method accepts both the query criteria and projections and returns a cursor to the matching documents

SELECT & db.coll.find()

```
SELECT _id, name, address   <-- projection
FROM   users                <-- table
WHERE  age > 18             <-- select criteria
LIMIT  5                    <-- cursor modifier
```

```
db.users.find(                          <-- collection
   { age: { $gt: 18 } },                <-- query criteria
   { name: 1, address: 1 }              <-- projection
).limit(5)                              <-- cursor modifier
```

Query Behavior

- All queries in MongoDB address a single collection
- You can modify the query to impose limits, skips, and sort orders
- The order of documents returned by a query is not defined and is not necessarily consistent unless you specify a sort()
- Operations that modify existing documents (i.e. updates) use the same query syntax as queries to select documents to update
- In aggregation pipeline, the $match pipeline stage provides access to MongoDB queries

Data Modification

- Data modification refers to operations that create, update, or delete data
- In MongoDB, these operations modify the data of a single collection
- All write operations in MongoDB are atomic on the level of a single document
- For the update and delete operations, you can specify the criteria to select the documents to update or remove

The stages of a MongoDB insert operation

Collection

Document

```
db.users.insert(
                {
                    name: "sue",
                     age: 26,
                  status: "A",
                  groups: [ "news", "sports" ]
                }
               )
```

Document

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

insert

Collection

```
{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }
```

users

INSERT vs. db.coll.insert()

SQL INSERT

```
INSERT INTO users                    <-- table
            ( name, age, status )    <-- columns
VALUES      ( "sue", 26, "A" )       <-- values/row
```

db.coll.insert()

```
db.users.insert (                    <-- collection
   {
     name: "sue",                    <-- field: value
     age: 26,                        <-- field: value
     status: "A"                     <-- field: value
   }                                 } document
)
```

UPDATE vs. db.coll.update()

SQL UPDATE

```
UPDATE users          <-- table
SET    status = 'A'   <-- update action
WHERE  age > 18       <-- update criteria
```

db.coll.update()

```
db.users.update(                      <-- collection
   { age: { $gt: 18 } },              <-- update criteria
   { $set: { status: "A" } },         <-- update action
   { multi: true }                    <-- update option
)
```

DELETE vs. db.coll.delete()

SQL DELETE

```
DELETE FROM users      <-- table
WHERE  status = 'D'    <-- delete criteria
```

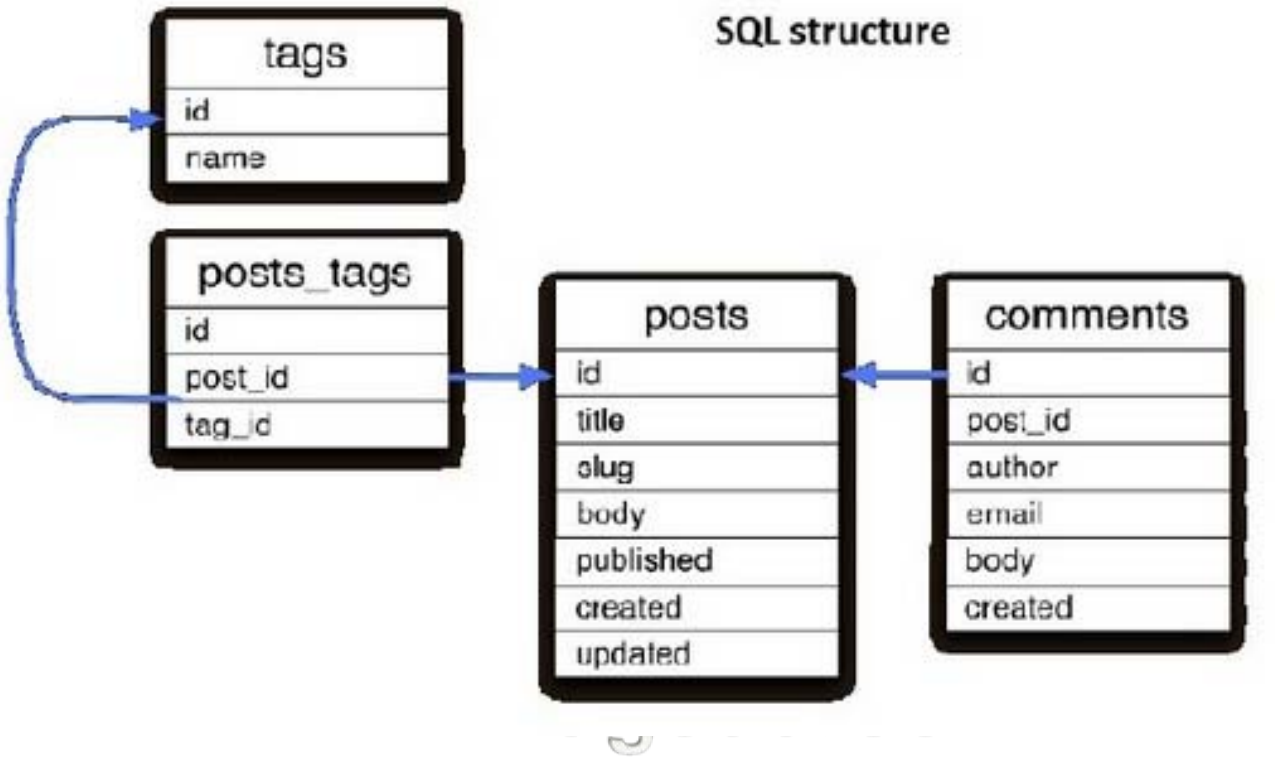
db.coll.delete()

```
db.users.remove(       <-- collection
   { status: "D" }     <-- remove criteria
)
```

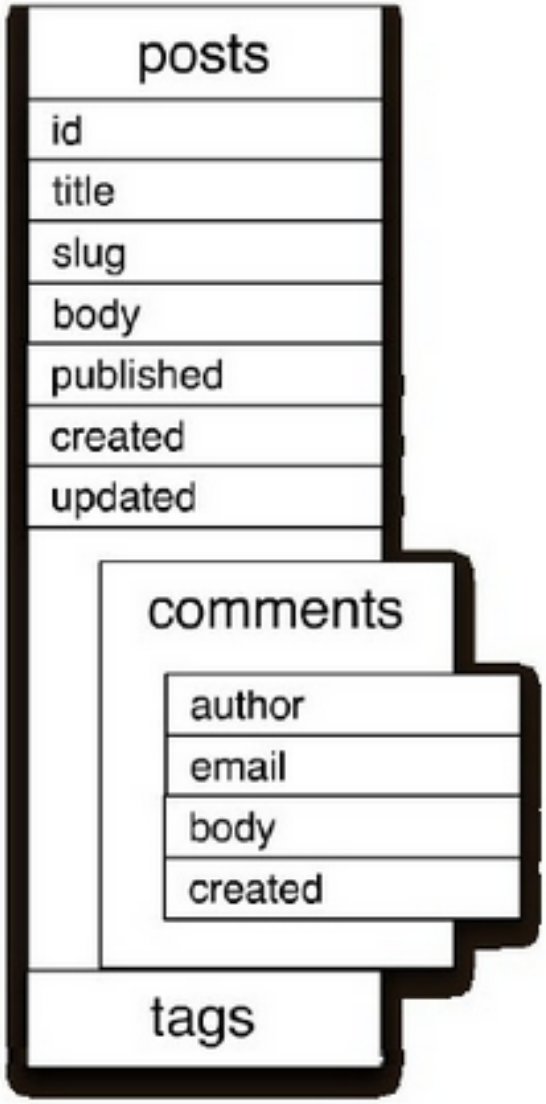
Terminology

RDBMS		Mongo
Table, View	➜	Collection
Row	➜	JSON Document
Index	➜	Index
Join	➜	Embedded
Partition	➜	Shard
Partition Key	➜	Shard Key

SQL Structure



mongoDB structure



使用示例

- db.mycoll.insert()
- db.mycoll.find()
  - find的高级操作
- db.mycoll.update()
- db.mycoll.delete()

db.collection.find()

- "db.collection.find(<query>, <projection>)"
  - 类似于SQL中的SELECT语句，其中<query>相当于WHERE子句，而<projection>相当于要选定的字段
  - 如果使用的find()方法中不包含<query>，则意味着要返回对应collection的所有文档
- "db.collection.count()"方法可以统计指定collection中文档的个数

db.collection.find() 高级操作

- MongoDB的查询操作支持挑选机制有"comparison"、"logical"、"element"和"JavaScript"等几类
- 比较运算(Comparison)
  - $gt：挑选指定字段值大于指定值的文档，语法格式" {field: {$gt: value} }";
  - $gte：挑选指定字段值大于等于指定值的文档，语法格式" {field: {$gte: value} }";
  - $in：挑选指定字段的值位于指定数组中的文档，语法格式"{ field: { $in: [<value1>, <value2>, ... <valueN>] } }";
  - $lt：挑选指定字段值小于指定值的文档，语法格式" {field: {$lt: value} }";
  - $lte：挑选指定字段值小于等于指定值的文档，语法格式" {field: {$lte: value} }";
  - $ne：挑选指定字段值不等于指定值的文档，语法格式" {field: {$lte: value} }";
  - $nin：挑选指定字段的值没有位于指定数组中或不存在的文档，语法格式"{ field: { $in: [<value1>, <value2>, ... <valueN>] } }";

使用示例

- 使用下面的命令生成一个含有100个文档的collction
  - > use testdb
  - > for (i=1;i<=100;i++)
    { db.testColl.insert({ hostname:"www"+i+".magedu.com"})
    }
  - > db.testColl.find({hostname: {$gt: "www95.magedu.com"}})
  - { "_id" : ObjectId("525d1e5bd249bfb8ae8c702e"), "hostname" : "www96.magedu.com" }
  - { "_id" : ObjectId("525d1e5bd249bfb8ae8c702f"), "hostname" : "www97.magedu.com" }
  - { "_id" : ObjectId("525d1e5bd249bfb8ae8c7030"), "hostname" : "www98.magedu.com" }
  - { "_id" : ObjectId("525d1e5bd249bfb8ae8c7031"), "hostname" : "www99.magedu.com" }

db.collection.find() 高级操作

- 逻辑运算
  - 逻辑运算一般用于连接多个选择条件，MongoDB支持的逻辑运算"Query Selector"有如下几种
    - $or：或运算，语法格式"{ $or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }"
    - $and：与运算，语法格式"{ $and: [{ <expression1> }, { <expression2> } , ... , { <expressionN> }] }"
    - $not：非运算，语法格式"{ field: { $not: { <operator-expression> } } }"
    - $nor：反运算，即返回不符合所有指定条件的文档，语法格式"{ $nor: [{ <expression1> }, { <expression2> }, ... { <expressionN> }] }"

使用示例

- 例如，查询testColl中hostname的值小于"www2.magedu.com"或大于"www96.magedu.com"的文档
  - > db.testColl.find({$or: [{hostname: {$lt: "www3.magedu.com"}}, {hostname: {$gt: "www96.magedu.com"}}]})

```
> db.testColl.find({$or: [ {hostname: {$lt: "www2.magedu.com"}}, {hostname: {$gt: "www96.magedu.com"}}]})
{ "_id" : ObjectId("525d20b1d249bfb8ae8c7097"), "hostname" : "www1.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a0"), "hostname" : "www10.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a1"), "hostname" : "www11.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a2"), "hostname" : "www12.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a3"), "hostname" : "www13.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a4"), "hostname" : "www14.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a5"), "hostname" : "www15.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a6"), "hostname" : "www16.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a7"), "hostname" : "www17.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a8"), "hostname" : "www18.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70a9"), "hostname" : "www19.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70f7"), "hostname" : "www97.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70f8"), "hostname" : "www98.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70f9"), "hostname" : "www99.magedu.com" }
{ "_id" : ObjectId("525d20b1d249bfb8ae8c70fa"), "hostname" : "www100.magedu.com" }
```

db.collection.find() 高级操作

- 元素查询(element)
  - 如果要根据文档中是否存在某字段等条件来挑选文档，则需要用到元素运算
    - **$exists**：根据指定字段的存在性挑选文档，语法格式"{ field: { $exists: <boolean> } }"，指定<boolean>的值为"true"则返回存在指定字段的文档，"false"则返回不存在指定字段的文档；
    - **$mod**：将指定字段的值进行取模运算，并返回其余数为指定值的文档；语法格式"{ field: { $mod: [divisor, remainder]} }"；
    - **$type**：返回指定字段的值类型为指定类型的文档，语法格式"{ field: { $type: <BSON type> } }"；
      - 可用类型的列表请参考官方文档

使用示例

- 例如，为testColl新增两个具有url字段的文档，而后根据其进行查询
  - > db.testColl.insert({"hostname": "www101.magedu.com", "url": "index.html"})
  - > db.testColl.insert({"hostname": "www102.magedu.com", "url": "index.html"})
  - > db. testColl.find({url: {$exists: true}})
  - { "_id" : ObjectId("521083cfb639c917f2cd86e8"), "hostname" : "www10.magedu.com", "url" : "index.html" }

```
> db. testColl.find({url: {$exists: true}})
{ "_id" : ObjectId("525d21b6d249bfb8ae8c70fb"), "hostname" : "www101.magedu.com", "url" : "index.html" }
{ "_id" : ObjectId("525d21c2d249bfb8ae8c70fc"), "hostname" : "www102.magedu.com", "url" : "index.html" }
```

更新操作

- update()方法可用于更改collection中的数据，默认情况下，update()只更新单个文档，若要一次更新所有符合指定条件的文档，则要使用multi选项
- update()方法的使用格式为“db.collection.update(<query>, <update>, <options>)”，其中<query>类似于SQL语句中的WHERE，而<update>相当于附带了“LIMIT 1”的SET，如果在<options>处提供“multi”选项，则update语句则类似于不带LIMIT语句的update()

更新操作

- <update>参数的使用格式比较独特，其仅能包含使用update专有操作符来构建的表示式，其专有操作符大致包含“Field”、“Array”、“Bitwise”和“Bitwise”几类，这里只介绍第一类的使用
- “Field”类常用的操作如下。
  - $inc：增大指定字段的值，其使用格式为“db.collection.update({ field: value }, { $inc: { field1: amount } })”，其中“{ field: value }”用于指定挑选标准，“{ $inc: { field1: amount }”用于指定要提升其值的字段及提升大小“amount”；
  - $rename：更改字段名，使用格式为“{$rename: { <old name1>: <new name1>, <old name2>: <new name2>, ... } }”；
  - $set：修改字段的值为新指定的值，使用格式“db.collection.update({ field: value1 }, { $set: { field1: value2 } })”；
  - $unset：删除指定的字段，使用格式“db.collection.update({ field: value1 }, { $unset: { field1: "" } })”；

使用示例

- 例如，更新其hostname为www15的url为index.php

```
> db.testColl.update({hostname: "www95.magedu.com"}, {$set: {url: "index.php"}})
> db.testColl.find({url: "index.php"})
{ "_id" : ObjectId("525d2674d249bfb8ae8c715b"), "hostname" : "www95.magedu.com", "url" : "index.php" }
```

马哥教育

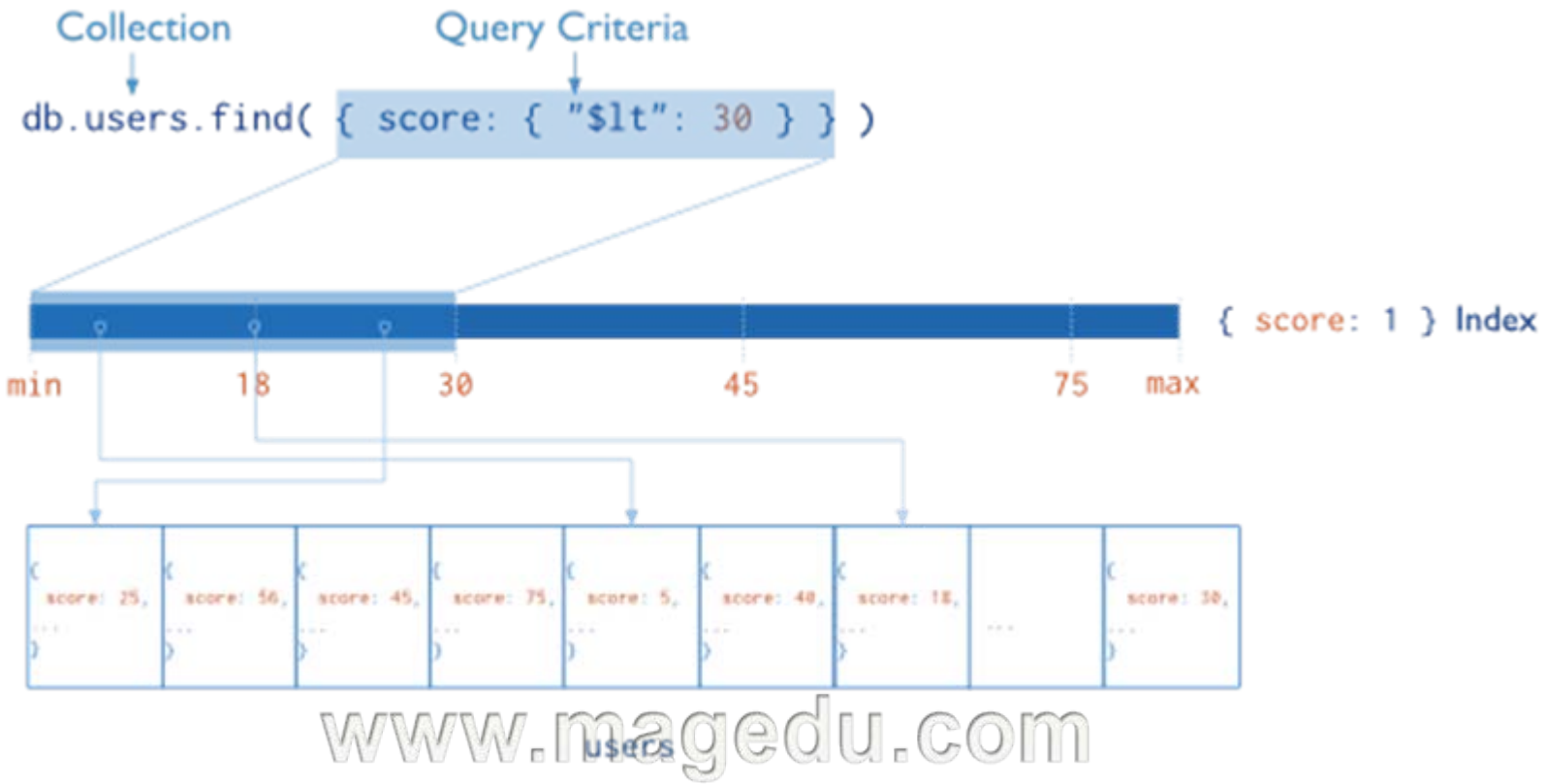
MongoDB Indexes

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

index introduction

- Indexes are special data structures that store a small portion of the collection's data set in an easy to traverse form
  - The index stores the value of a specific field or set of fields, ordered by the value of the field
- MongoDB defines indexes at the collection level and supports indexes on any field or sub-field of the documents in a MongoDB collection

query with index



Index Type

- Single Field Indexes
  - A single field index only includes data from a single field of the documents in a collection
  - MongoDB supports single field indexes on fields at the top level of a document *and* on fields in sub-documents
- Compound Indexes
  - A compound index includes more than one field of the documents in a collection
- Multikey Indexes
  - A multikey index references an array and records a match if a query includes any value in the array

- Geospatial Indexes and Queries
  - Geospatial indexes support location-based searches on data that is stored as either GeoJSON objects or legacy coordinate pairs
- Text Indexes
  - Text indexes supports search of string content in documents
- Hashed Index
  - Hashed indexes maintain entries with hashes of the values of the indexed field

Index Creation

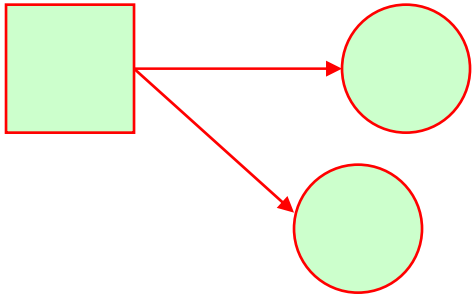
- MongoDB provides several options that only affect the creation of the index
- Specify these options in a document as the second argument to the db.collection.ensureIndex() method
  - unique index
    - db.addresses.ensureIndex({ "user_id": 1 }, { unique: true })
  - sparse index
    - db.addresses.ensureIndex({ "xmpp_id": 1 }, { sparse: true })

马哥教育

Replication Sets

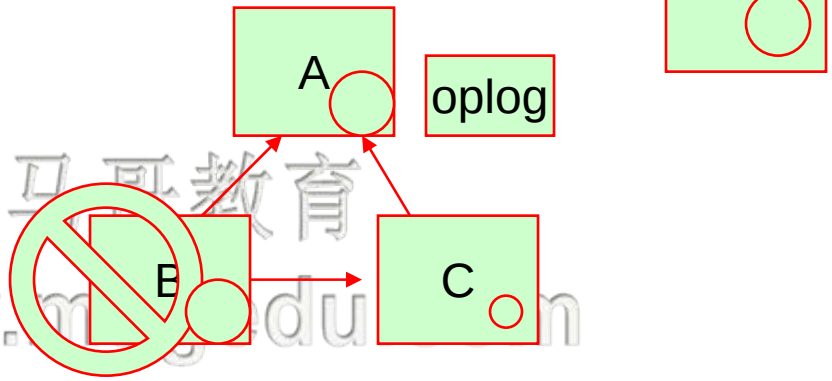
主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

mongodb的复制架构

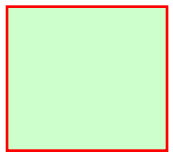


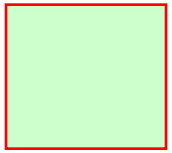
Replica Set:

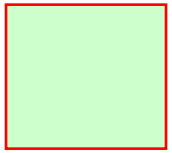
复制集，副本集名称(复制集群名称)



- replset=testrs0



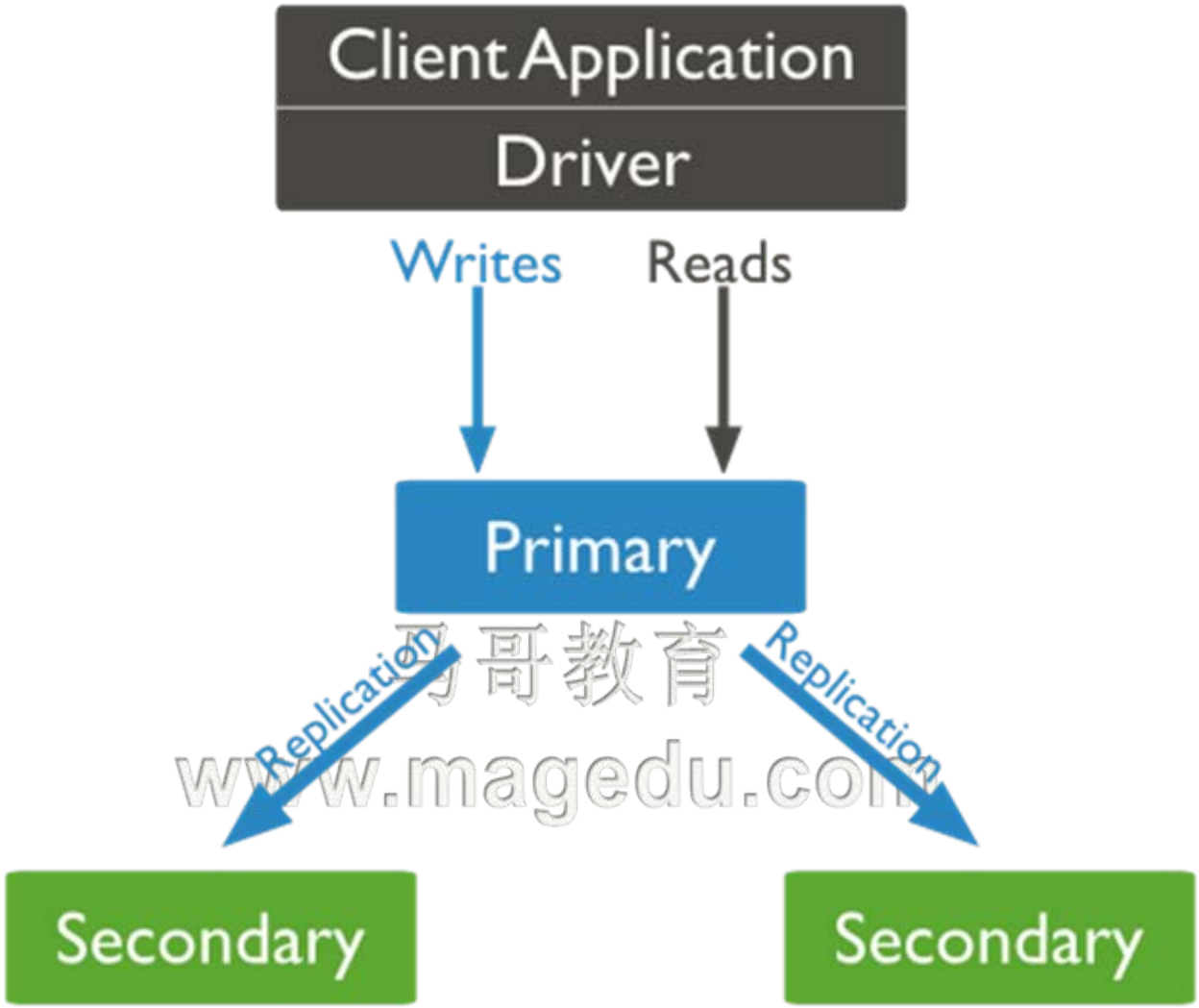




Purpose of Replication

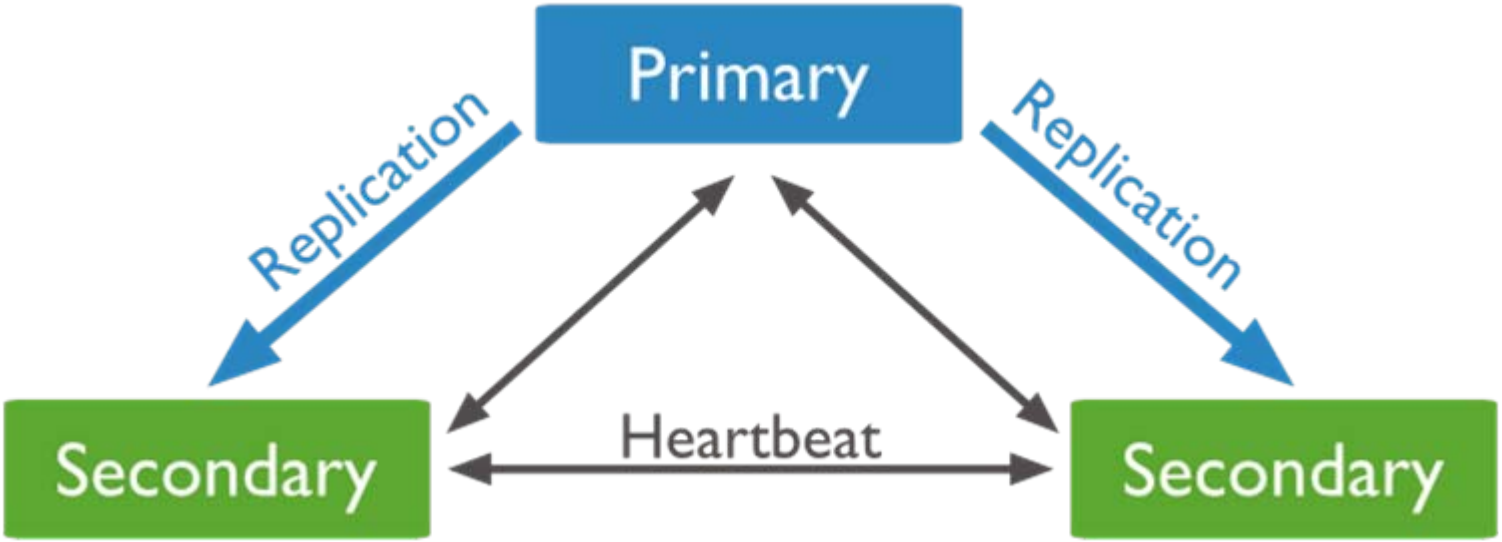
- Replication provides redundancy and increases data availability
- With multiple copies of data on different database servers, replication protects a database from the loss of a single server
- Replication also allows you to recover from hardware failure and service interruptions
- With additional copies of the data, you can dedicate one to disaster recovery, reporting, or backup

Replication in MongoDB



Replication in MongoDB

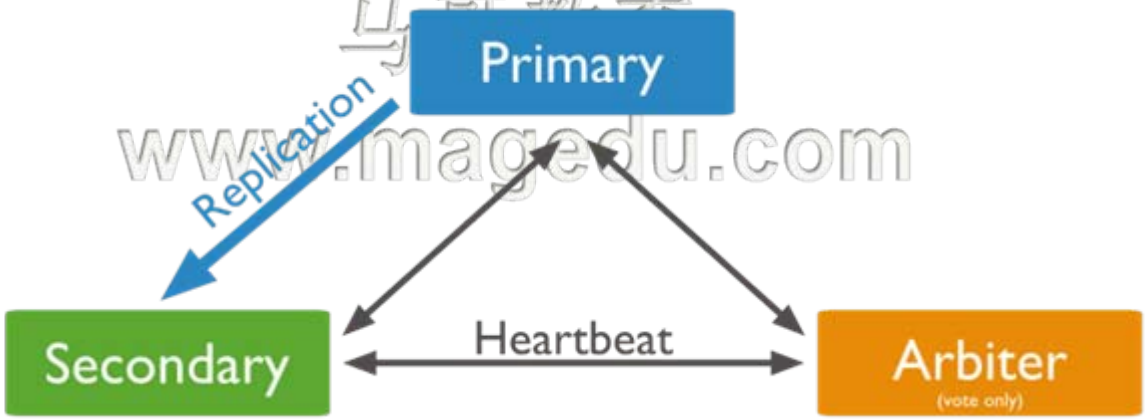
- A replica set is a group of mongod instances that host the same data set
  - One mongod, the primary, receives all write operations
  - All other instances, secondaries, apply operations from the primary so that they have the same data set
- The primary accepts all write operations from clients. Replica set can have only one primary
  - Because only one member can accept write operations, replica sets provide strict consistency
  - To support replication, the primary logs all changes to its data sets in its oplog



- The secondaries replicate the primary's oplog and apply the operations to their data sets
- Secondaries' data sets reflect the primary's data set
- If the primary is unavailable, the replica set will elect a secondary to be primary
- By default, clients read from the primary, however, clients can specify a read preferences to send read operations to secondaries

Arbiter

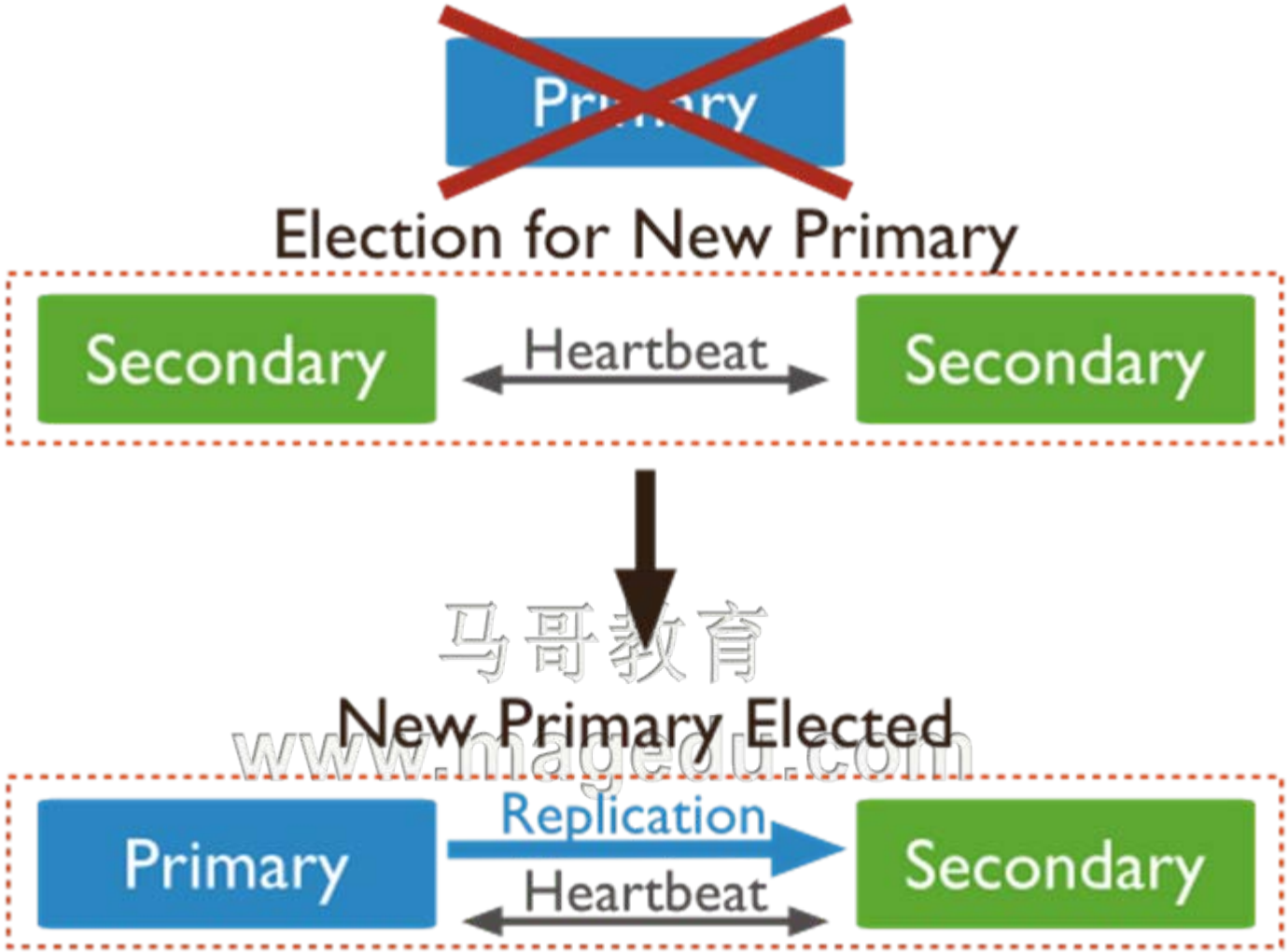
- You may add an extra mongod instance a replica set as an arbiter
- Arbiters do not maintain a data set. Arbiters only exist to vote in elections
- If your replica set has an even number of members, add an arbiter to obtain a majority of votes in an election for primary
- Arbiters do not require dedicated hardware



Automatic Failover

- When a primary does not communicate with the other members of the set for more than 10 seconds, the replica set will attempt to select another member to become the new primary
- The first secondary that receives a majority of the votes becomes primary

Automatic Failover

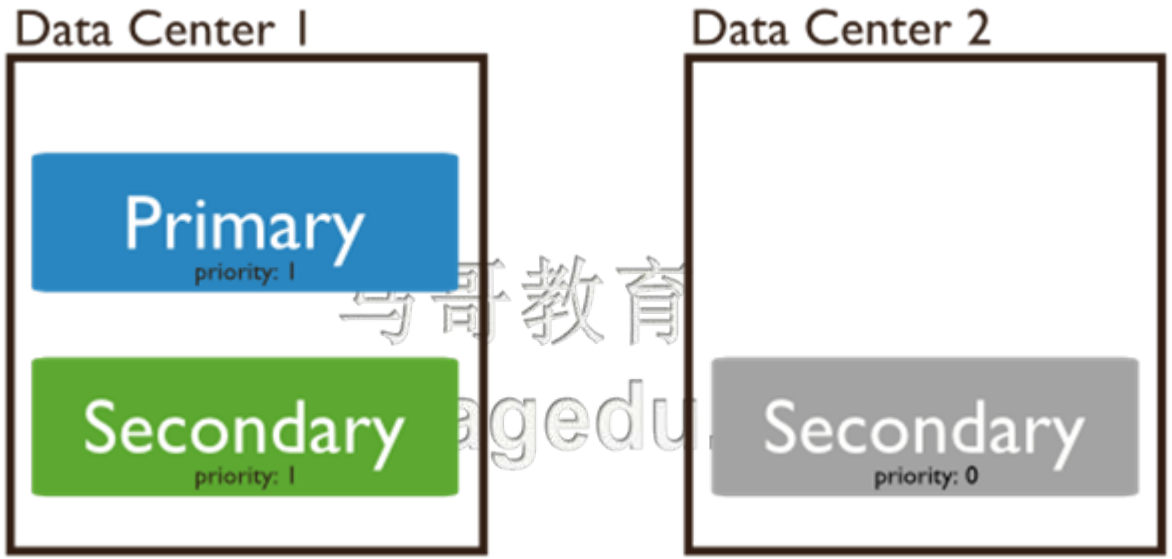


Priority 0 Replica Set Members

- A priority 0 member is a secondary that cannot become primary
- Priority 0 members cannot trigger elections
  - Otherwise these members function as normal secondaries
- A priority 0 member maintains a copy of the data set, accepts read operations, and votes in elections
- Configure a priority 0 member to prevent secondaries from becoming primary, which is particularly useful in multi-data center deployments

Priority 0 Replica Set Members

- In a three-member replica set, in one data center hosts the primary and a secondary
- A second data center hosts one priority 0 member that cannot become primary



马哥教育

MongoDB Sharding

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

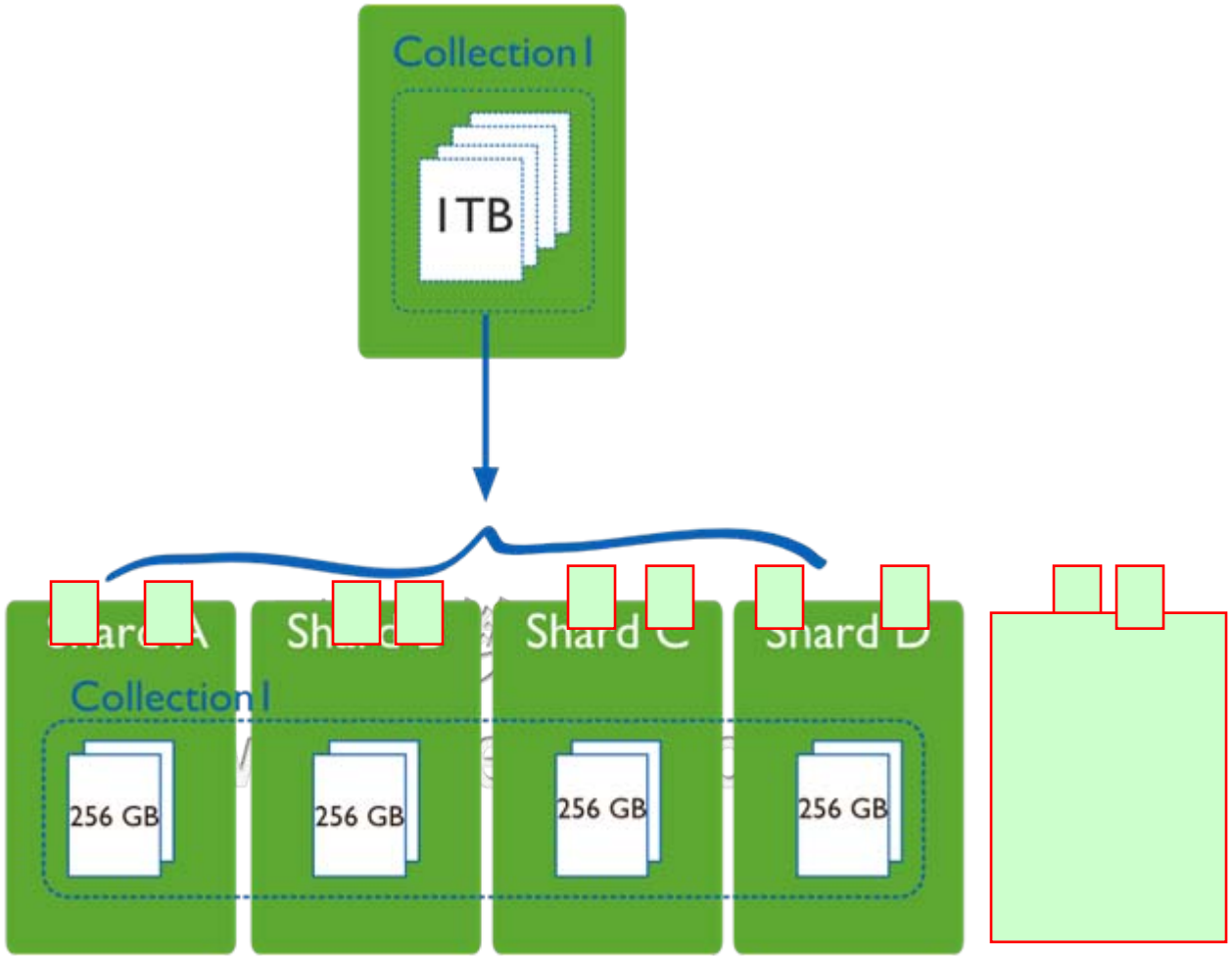
Sharding

- Sharding is the process of storing data records across multiple machines and is MongoDB's approach to meeting the demands of data growth
- As the size of the data increases, a single machine may not be sufficient to store the data nor provide an acceptable read and write throughput
- Sharding solves the problem with horizontal scaling
- With sharding, you add more machines to support data growth and the demands of read and write operations

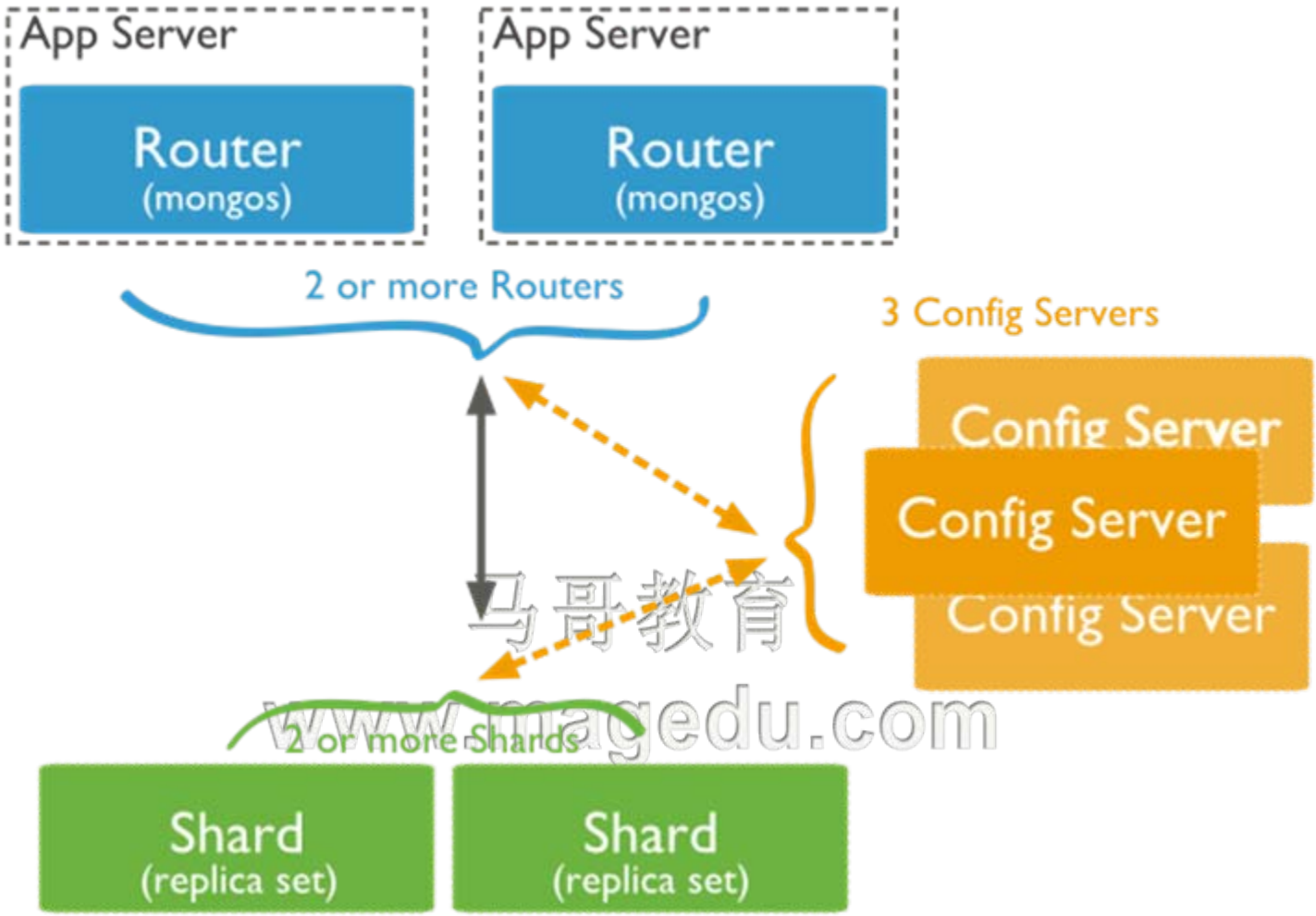
Purpose of Sharding

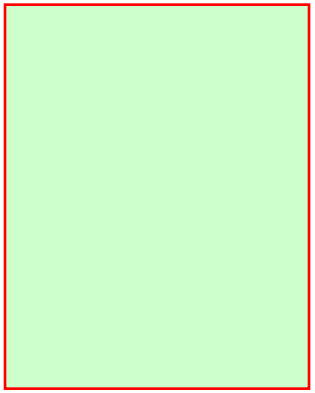
- Database systems with large data sets and high throughput applications can challenge the capacity of a single server
- High query rates can exhaust the CPU capacity of the server
- Larger data sets exceed the storage capacity of a single machine
- Finally, working set sizes larger than the system's RAM stress the I/O capacity of disk drives

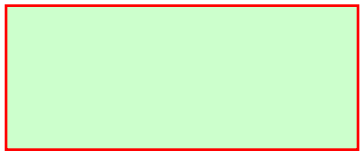
Sharding Diagram

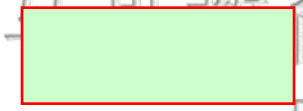


Sharding in MongoDB









Sharding in MongoDB

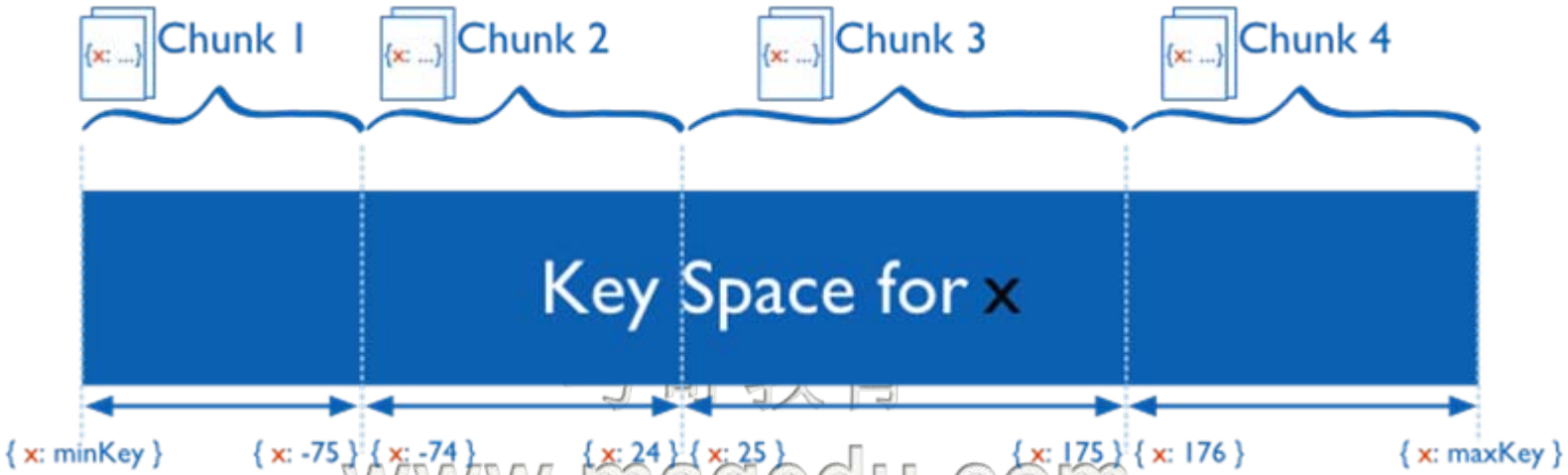
- Sharded cluster has the following components: shards, query routers and config servers
  - Shards store the data. To provide high availability and data consistency, in a production sharded cluster, each shard is a replica set
  - Query Routers, or mongos instances, interface with client applications and direct operations to the appropriate shard or shards
    - The query router processes and targets operations to shards and then returns results to the clients
    - A sharded cluster can contain more than one query router to divide the client request load
    - A client sends requests to one query router
    - Most sharded cluster have many query routers
- Config servers store the cluster's metadata
  - This data contains a mapping of the cluster's data set to the shards
  - The query router uses this metadata to target operations to specific shards
  - Production sharded clusters have exactly 3 config servers

Data Partitioning

- Shard Keys
  - To shard a collection, you need to select a shard key
  - A shard key is either an indexed field or an indexed compound field that exists in every document in the collection
  - MongoDB divides the shard key values into chunks and distributes the chunks evenly across the shards
  - To divide the shard key values into chunks, MongoDB uses either range based partitioning and hash based partitioning

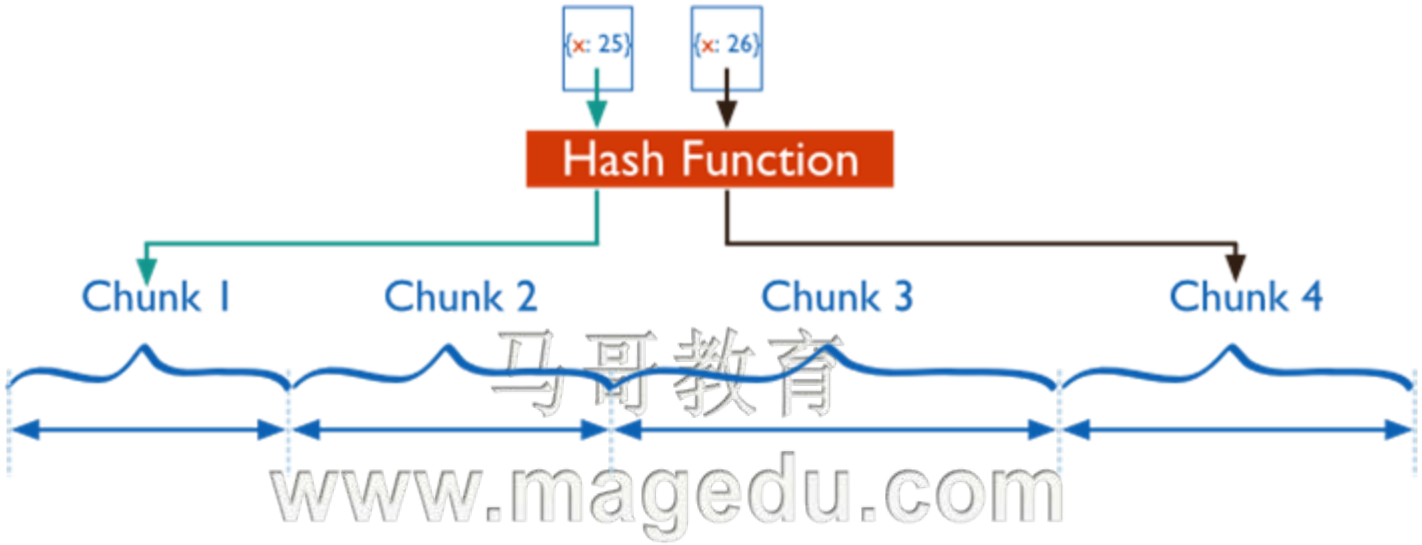
Range Based Sharding

- For range-based sharding, MongoDB divides the data set into ranges determined by the shard key values to provide range based partitioning



Hash Based Sharding

- For hash based partitioning, MongoDB computes a hash of a field's value, and then uses these hashes to create chunks



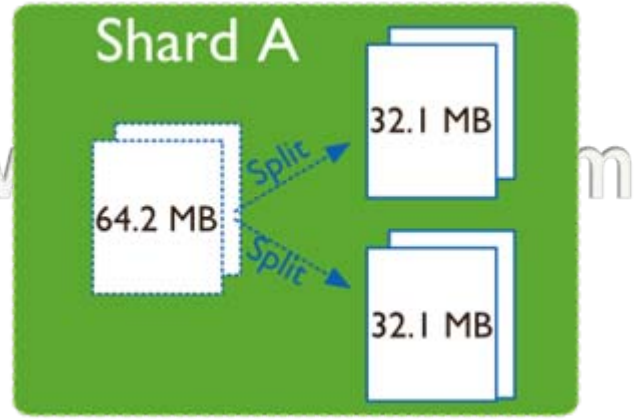
- Performance Distinctions between Range and Hash Based Partitioning
  - Range based partitioning supports more efficient range queries
  - However, range based partitioning can result in an uneven distribution of data, which may negate some of the benefits of sharding
  - Hash based partitioning, by contrast, ensures an even distribution of data at the expense of efficient range queries
  - But random distribution makes it more likely that a range query on the shard key will not be able to target a few shards but would more likely query every shard in order to return a result

Maintaining a Balanced Data Distribution

- MongoDB ensures a balanced cluster using two background process: splitting and the balancer

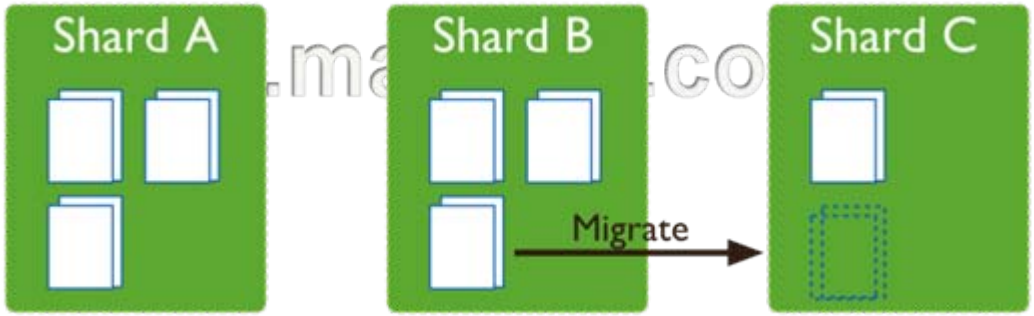
Splitting

- Splitting
  - A background process that keeps chunks from growing too large
  - When a chunk grows beyond a specified chunk size, MongoDB splits the chunk in half
  - Inserts and updates triggers splits
  - Splits are a efficient meta-data change
  - To create splits, MongoDB does not migrate any data or affect the shards



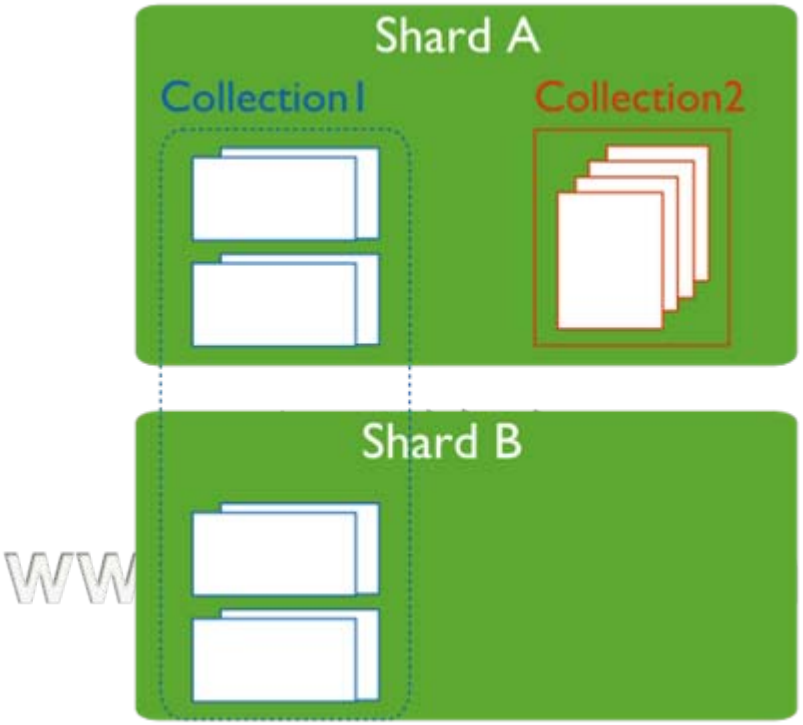
Balancing

- Balancing
  - The balancer is a background process that manages chunk migrations
  - The balancer runs in all of the query routers in a cluster
  - When the distribution of a sharded collection in a cluster is uneven, the balancer process migrates chunks from the shard that has the largest number of chunks to the shard with the least number of chunks until the collection balances
  - The shards manage chunk migrations as a background operation
  - During migration, all requests for a chunks data address the "origin" shard



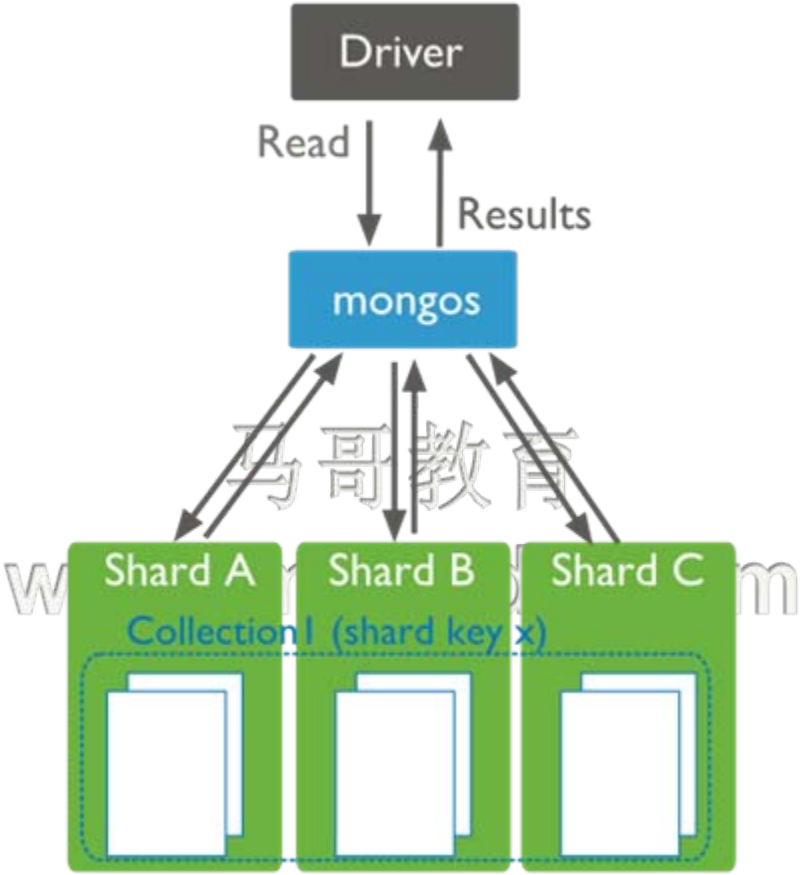
Primary Shard

- Every database has a "primary" shard that holds all the un-sharded collections in that database.



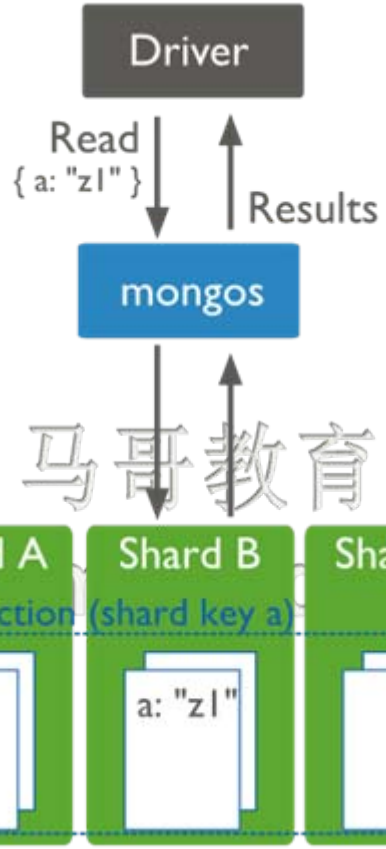
Broadcast Operations

- mongos instances broadcast queries to all shards for the collection unless the mongos can determine which shard or subset of shards stores this data

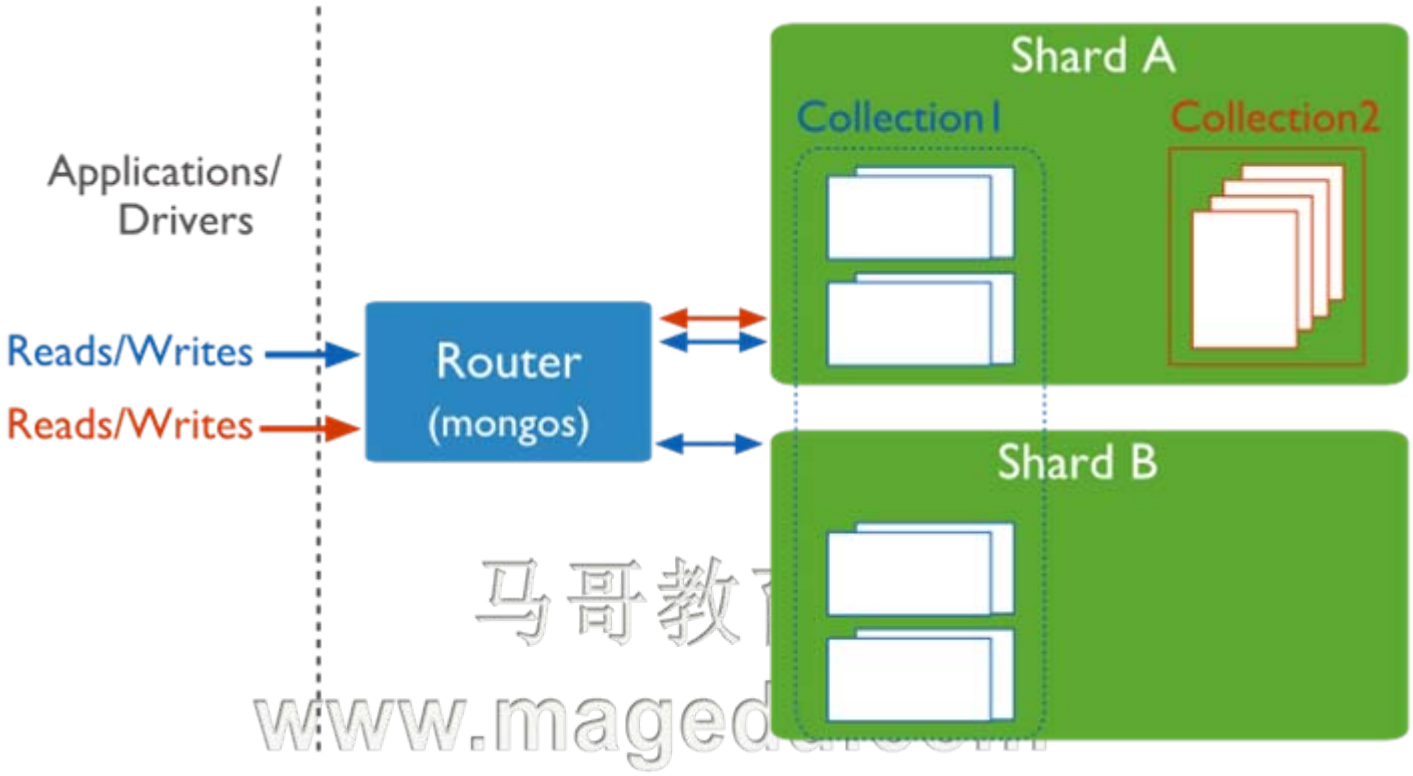


Targeted Operations

- All insert() operations target to one shard
- All single update() (including upsert operations) and remove() operations must target to one shard



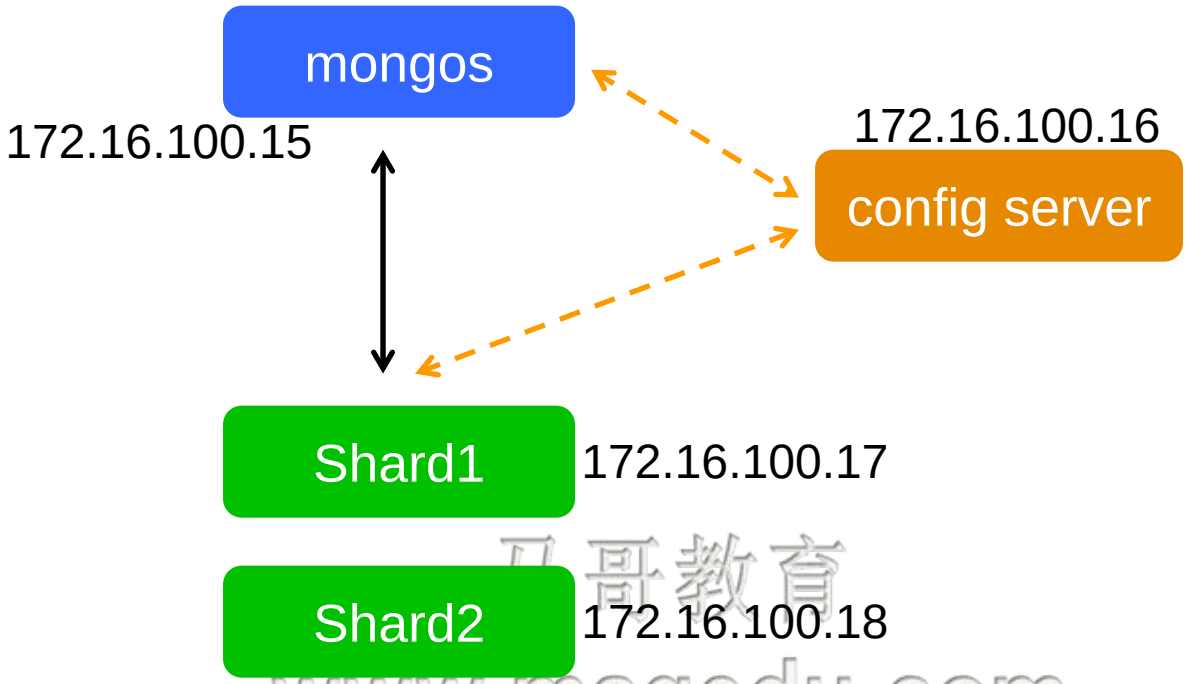
Sharded and Non-Sharded Data



Sharded Cluster Metadata

- Config servers store the metadata for a sharded cluster
  - The metadata reflects state and organization of the sharded data sets and system
  - The metadata includes the list of chunks on every shard and the ranges that define the chunks
  - The mongos instances cache this data and use it to route read and write operations to shards
- Config servers store the metadata in the Config Database

案例



马哥教育

管理 MongoDB

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

- 配置mongod
  - mongod在启动时通过命令行选项或配置文件(如/etc/mongod.conf)接口读取其配置属性
  - 这两种配置接口能够提供相同的功能，因此，管理员可以根据偏好进行选择
  - 如果要通过配置文件读取配置选项，则可以在启动mongod时使用--config或-f选项来指定配置文件的位置

mongod的基本配置参数

- fork={true|false}
  - 是否以daemon方式启动mongod，为true表示其启动后其自动转入后台工作；
- bind_ip=IP
  - 指定mongod监听的IP地址；
- port=PORT
  - 指定mongod监听的端口，默认为27017；
- quiet={true|false}
  - 是否工作于静默模式，以最小化的记录日志信息；正常条件下，应该设置为true，仅在调试时才将其设置为false；

mongod的基本配置参数

- dbpath=/PATH/TO/SOMEWHERE
  - 指定mongod存储数据的位置，通常为/data/mongodb、/var/lib/mongodb或/srv/mongodb等；
- logpath=/PATH/TO/SOMEFILE
  - 日志文件路径，例如/var/log/mongodb/mongod.log；如果没有指定文件路径，日志信息则会发往标准输出；
- logappend={true|false}
  - 设定mongod在启动时是否不覆盖日志文件中的原有日志信息，true表示以追加方式记录日志，即不覆盖；
- journal={true|false}
  - 是否启动日志功能；日志功能是保证工作于单实例模式的mongod写入数据持久性的惟一途径；

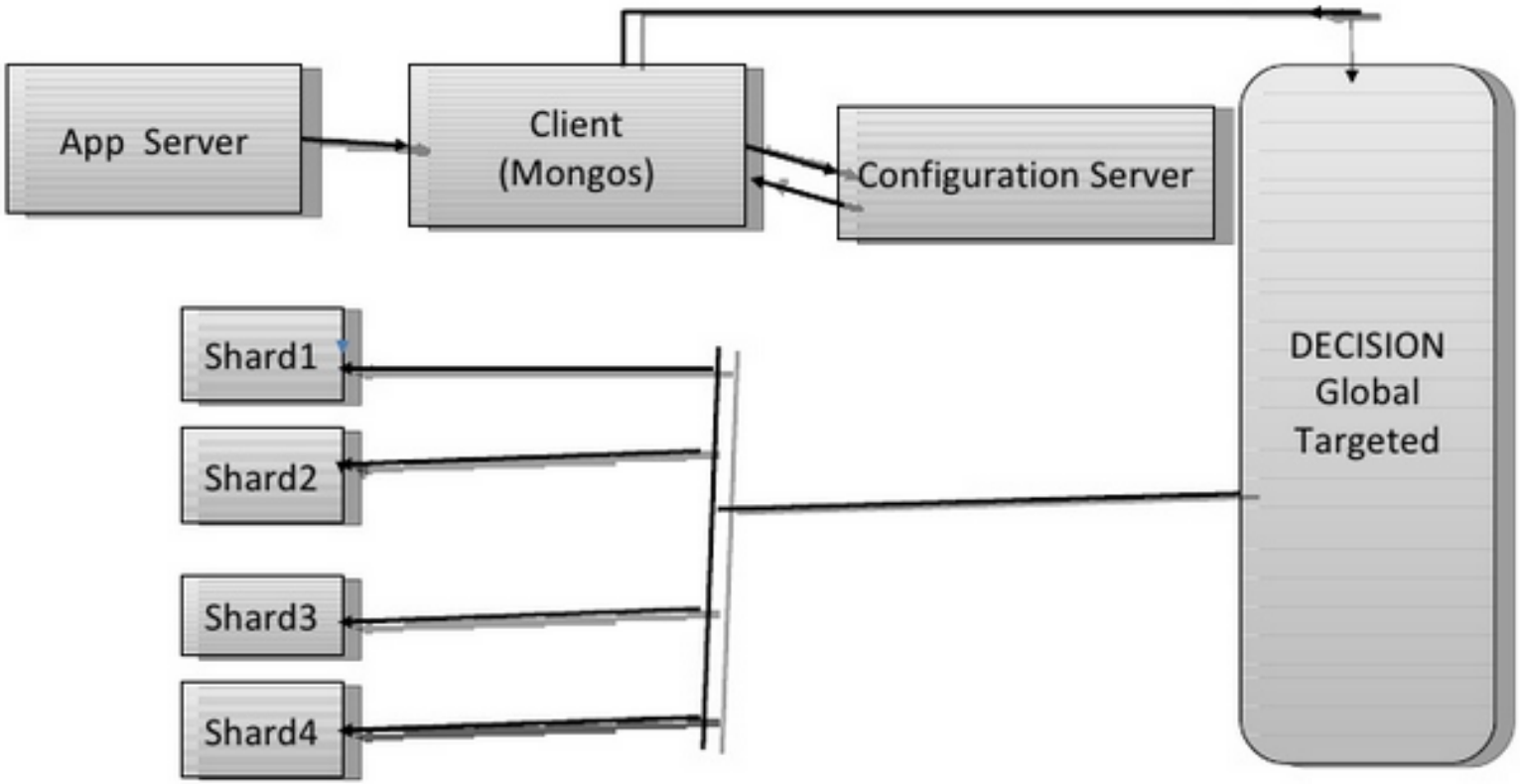
安全相关的配置参数

- bind_ip=IP
  - 指定mongod监听的IP地址；生产环境中，通常需要将其指定为一个可接受外部客户端请求的IP地址，但应该仅开放需要接受客户端请求的IP地址；
  - 如果需要指定多个地址，彼此间使用逗号分隔即可；
- nounixsocket={true|false}
  - 是否禁用mongodb的Unix套接字功能，默认为启用；主要用于本地通信；
- auth={true|false}
  - 是否启用认证功能；
  - 如果要启用，远程客户端需要被授权方能访问mongodb服务器；

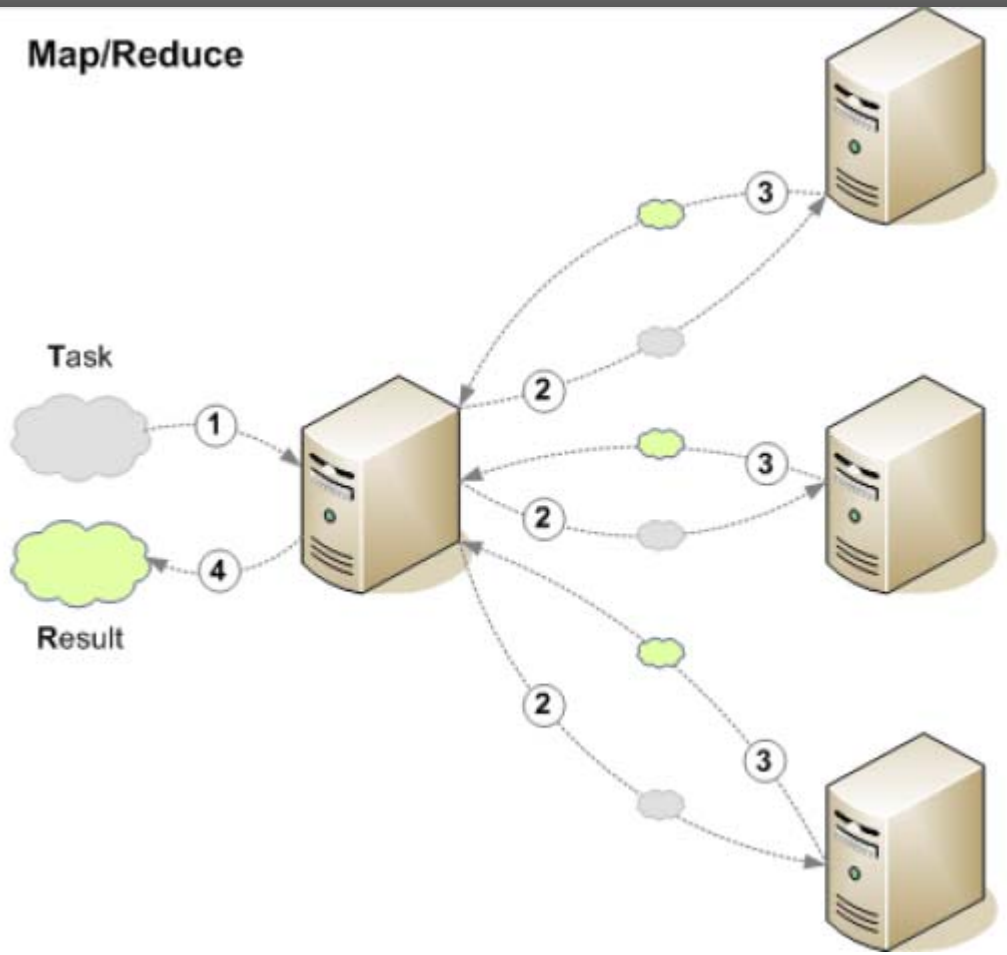
调试类参数

- 为了获释mongod的动作细节，可以通过调整其调试类参数进行
  - slowms=#：慢查询阈值，单位为毫秒，主是为profiler所用到；默认为100ms；
  - profile=#：指定profile的级别，默认没有启用；建议仅在需要评估mongod性能时启用；
  - verbose={true|false}：是否向日志中记录调试信息，生产环境不建议启用；其可以简写为v，并且能够通过增加v的个数来定义其详细信息级别；
  - diaglog=#：是否记录诊断信息至日志中；
  - objcheck={true|false}：是否强制mongod验证客户端发来的每一个请求；
  - cpu={true|false}：设定mongod报告写锁占用时间百分比的时间间隔，默认为4秒种；

Flow Chart



Map/Reduce



关于马哥教育

- 博客：http://mageedu.blog.51cto.com
- 主页：http://www.magedu.com
- QQ：2813150558，1661815153，113228115
- QQ群：203585050，279599283

马哥教育

Thank You!