1. MongoDB 高可用复制集架构
2. MongoDB 集群分片机制原理
3. MongoDB 应用与开发实战

MongoDB 复制集

MongoDB复制集的主要意义在于实现服务高可用，类似于Redis中的哨兵模式

它主要提供两个方面的功能

1. 数据写入主节点（Primary）时将数据复制到另一个副本节（Secondary）点上
2. 主节点发生故障时自动选举出一个新的替代节点

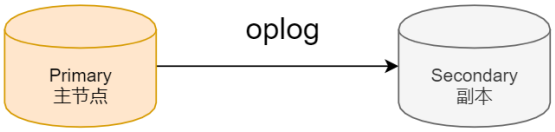
在实现高可用的同时，复制集实现了其他几个作用

数据分发：将数据从一个区域复制到另一个区域，减少另一个区域的读延迟
读写分离：不同类型的压力分别在不同的节点上执行
异地容灾：在数据中心故障时快速切换到异地

典型复制集结构
一个典型的复制集由三个或三个以上具有投票权的节点组成，其中一个主节点（Primary）：接收写入操作，读操作和选举时投票，两个或多个从节点(Secondary)：复制主节点上的新数据和选举时投票

数据是如何复制的?
当一个修改操作，无论是插入，更新或删除，到达主节点时，它对数据的操作将被记录下来（经过一些必要的转换）。这些记录称为oplog

从节点通过从主节点上不断获取新进入主节点的oplog，并在自己的数据上回放，以此保持跟主节点的数据一致。



通过选举完成故障恢复

具有投票权的节点之间两两互相发送心跳；
当5次心跳未收到时判断为节点失联
如果失联的是主机点，从节点会发起选举，选出新的主节点
如果失联的是从节点则不会产生新的选举
选举基于RAFT一致性算法实现，选举成功的必要条件是大多数投票节点存活
复制集中最多可以有50个节点，但具有投票权的节点最多7个

影响选举的因素
整个集群必须有大多数节点存活
被选举为主节点的节点必须
1.能够与多数节点建立连接
2.具有较新的oplog
3.具有较高的优先级（如果有配置）

复制集节点有以下的选配项

是否具有投票权（v 参数）： 有则参与投票
优先级（priority参数）：优先级越高的节点越优先成为主节点。优先级为0的节点无法成为主节点,默认值为1。

隐藏（hidden参数）：复制数据，但对应用不可见。隐藏节点可以具有投票权，但优先级必须为0
延迟（slaveDelay参数）：复制 n 秒之前的数据，保持与主节点的时间差
从节点不建立索引(buildIndexes)

复制集注意事项

硬件：
因为正常的复制集节点都有可能成为主节点，它们的地位是一样的，因此硬件配置上必须一致
为了保证节点不会同时宕机，各节点的硬件必须具有独立性。

软件：
复制集各节点软件版本必须一致，以避免出现不可预知的问题

增加节点不会增加系统写性能

复制集搭建

1. 创建数据目录文件

Linux系统
mkdir -p /data/db{1,2,3}

3. 准备每个数据库的配置文件
复制集的每个mongod进程应该位于不同的服务器。我们现在在一台服务器上运行三个实例，所以要为它们各自配置
1.不同的端口： 28017， 28018，28019.
2.不同的数据目录
data/db1,data/db2,data/db3
3. 不同的日志文件路径。实例中使用
/data/db1/mongod.log
/data/db2/mongod.log

/data/db3/mongod.log

data/db1/mongod.conf

```
systemLog:
 destination: file
 path: /data/db1/mongod.log
 logAppend: true
storage:
 dbPath: /data/db1
net:
 bindIp: 0.0.0.0
 port: 28017
 replication:
  replSetName: rs0
 processManagement:
  fork: true

```

/data/db2/mongod.conf

```
systemLog:
 destination: file
 path: /data/db2/mongod.log
 logAppend: true
storage:
 dbPath: /data/db2
net:
 bindIp: 0.0.0.0
 port: 28018
 replication:
  replSetName: rs0
 processManagement:
  fork: true

```

/data/db3/mongod.conf

```
systemLog:
 destination: file
 path: /data/db3/mongod.log
 logAppend: true
storage:
 dbPath: /data/db3
net:
 bindIp: 0.0.0.0
```

```
 port: 28019
replication:
 replSetName: rs0
processManagement:
 fork: true

```

分别启动

mongod -f /data/db1/mongod.conf

mongod -f /data/db2/mongod.conf

mongod -f /data/db3/mongod.conf

window没法 fork， 所以只能用前台进程开启， 需要 开3个命令窗口分别启动

配置复制集

mongo --port 28017

```
> rs.initiate({

_id:"rs0",
members:[{
_id:0,
host:"localhost:28017"
},{
_id:1,
host:"localhost:28018"
},{
_id:2,
host:"localhost:28019"
}]
})

默认情况下非主节点不允许读数据
可以通过执行 rs.secondaryOk() 开启读权限
```

mongo --port 28017 连接到主节点

```
rs0:PRIMARY> use gjtest
switched to db gjtest
rs0:PRIMARY> db.users.insertOne({"username":"guojia","age":"35"});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5fbbc6b2d7ca687867f9b4db")
}
```

mongo --port 28018 连接到从节点上面

执行

```
rs0:SECONDARY> rs.secondaryOk()
rs0:SECONDARY> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
rs0:SECONDARY> use gjtest
switched to db gjtest
rs0:SECONDARY> db.users.find();
{ "_id" : ObjectId("5fbbc6b2d7ca687867f9b4db"), "username" : "guojia", "age" : "35" }
```

分片集群

1. 什么是分片?

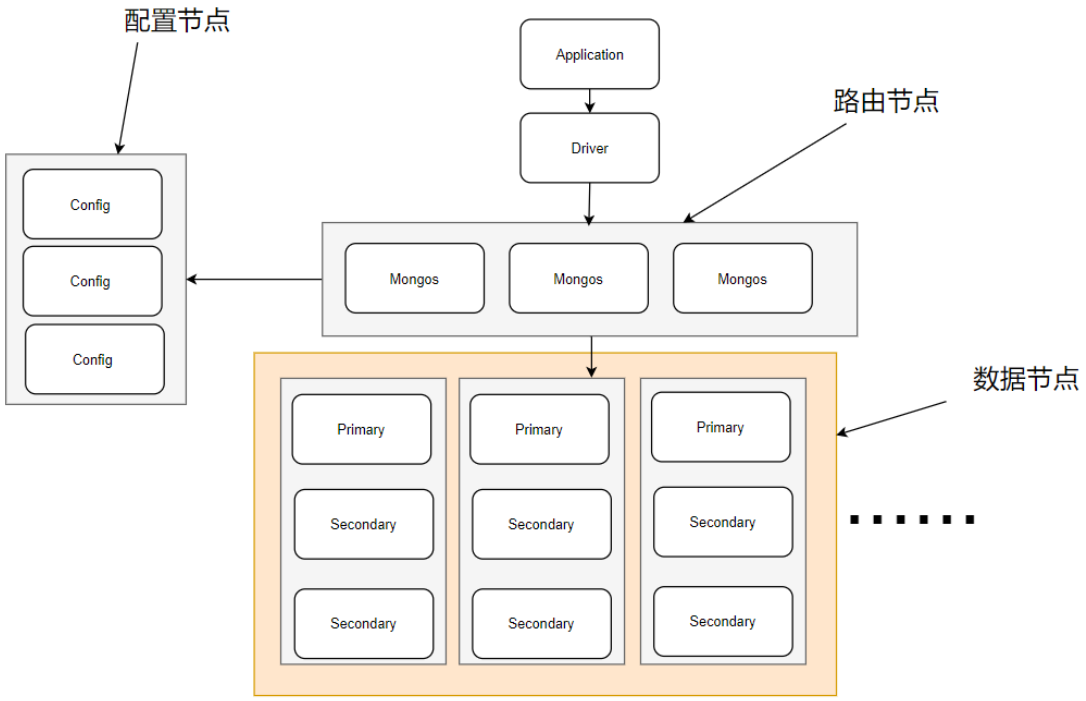
将数据水平拆分到不同的服务器上

2. 为什么要使用分片集群

数据量突破单机瓶颈，数据量大，恢复很慢，不利于数据管理

并发量突破单机性能瓶颈

MongoDB 分片集群由一下几部分组成



分片集群角色：

路由节点： mongos， 提供集群单一入口，转发应用端请求，选择合适的数据节点进行读写，合并多个数据节点的返回。无状态，建议 mongos节点集群部署以提供高可用性。客户请求应发给mongos，而不是 分片服务器，当查询包含分片片键时，mongos将查询发送到指定分片，否则，mongos将查询发送到所有分片，并汇总所有查询结果。

配置节点：就是普通的mongod进程， 建议以复制集部署，提供高可用
提供集群元数据存储分片数据分布的数据。主节点故障时，配置服务器进入只读模式，
只读模式下，数据段分裂和集群平衡都不可执行。整个复制集故障时，分片集群不可用

数据节点：
以复制集为单位，横向扩展最大1024分片，分片之间数据不重复，所有数据在一起才可以完整工作。

分片键

可以是单个字段， 也可以是复合字段

1. 范围分片
比如 key 的值 从 min - max
可以把数据进行范围分片

2. hash 分片
通过 hash(key) 进行数据分段

片键值用来将集合中的文档划分为数据段,片键必须对应一个索引或索引前缀（单键、复合键）,可以使用片键的值 或者片键值的哈希值进行分片

选择片键

1. 片键值的范围更广（可以使用复合片键扩大范围）
2. 片键值的分布更平衡（可使用复合片键平衡分布）
3. 片键值不要单向增大、减小（可使用哈希片键）

数据段的分裂
当数据段尺寸过大，或者包含过多文档时，触发数据段分裂,只有新增、更新文档时才可能自动触发数据段分裂,数据段分裂通过更新元数据来实现

集群的平衡
后台运行的平衡器负责监视和调整集群的平衡,当最大和最小分片之间的数据段数量相差过大时触发
集群中添加或移除分片时也会触发

MongoDB分片集群特点

1.应用全透明
2.数据自动均衡
3.动态扩容，无需下线

搭建集群环境：
搭建一个2个分片的集群

步骤：
配置第一个 分片集群：
1. 创建数据目录： 准备给两个复制集使用，每个复制集有三个实例 ,共 6 个数据节点

```
mkdir -p /data/shard1 /data/shard1second1 /data/shard1second2 /data/shard2 /data/shard2second1 /data/shard2second2
```

2. 创建日志文件，共6 个文件

```
touch /data/shard1/mongod.log /data/shard1second1/mongod.log /data/shard1second2/mongod.log /data/shard2/mongod.log /data/shard2second1/mongod.log /data/shard2second2/mongod.log
```

3. 启动第一个 mongod 分片实例(一共三个实例)

```
mongod --bind_ip 0.0.0.0 --replSet shard1 --dbpath /data/shard1 --logpath /data/shard1/mongod.log --port 27010 --fork --shardsvr
```

```
mongod --bind_ip 0.0.0.0 --replSet shard1 --dbpath /data/shard1second1 --logpath /data/shard1second1/mongod.log --port 27011 --fork --shardsvr
```

```
mongod --bind_ip 0.0.0.0 --replSet shard1 --dbpath /data/shard1second2 --logpath /data/shard1second2/mongod.log --port 27012 --fork --shardsvr
```

4. 第一个分片的mongod 实例都启动好了后，将其加入到复制集中

```
rs.initiate(
{_id:"shard1",
 "members":[
 {"_id":0,"host":"192.168.109.200:27010"},
 {"_id":1,"host":"192.168.109.200:27011"},
 {"_id":2,"host":"192.168.109.200:27012"}
 ]
});
```

等待集群选举

5. 查看状态

```
rs.status();
```

配置Config 复制集：一共三个实例

1. 创建数据目录：

```
mkdir -p /data/config /data/configsecond1 /data/configsecond2
```

2. 创建日志文件

```
touch /data/config/mongod.log /data/configsecond1/mongod.log /data/configsecond2/mongod.log
```

3. 启动配置复制集

```
mongod --bind_ip 0.0.0.0 --replSet config --dbpath /data/config --logpath /data/config/mongod.log --port 37010 --fork --configsvr
```

```
mongod --bind_ip 0.0.0.0 --replSet config --dbpath /data/configsecond1 --logpath /data/configsecond1/mongod.log --port 37011 --fork --configsvr
```

```
mongod --bind_ip 0.0.0.0 --replSet config --dbpath /data/configsecond2 --logpath /data/configsecond2/mongod.log --port 37012 --fork --configsvr
```

4. 配置复制集进行初始化

```
rs.initiate(
{_id:"config",
"members":[
{"_id":0,"host":"192.168.109.200:37010"},
{"_id":1,"host":"192.168.109.200:37011"},
{"_id":2,"host":"192.168.109.200:37012"}
]
});
```

配置mongs 路由节点

1. 启动mongos 实例，需要指定配置服务器的地址列表

```
mongos --bind_ip 0.0.0.0 --logpath /data/mongos/mongos.log --port 4000 --fork
--configdb config/111.229.189.98:37010,111.229.189.98:37011,111.229.189.98:37012
```

其中 configdb 为配置服务器的地址列表

2. 连接到mongos中，并添加分片

直接通过mongo shell 客户端进行连接

mongo --port 4000 本地直连

执行脚本

```
sh.addShard("shard1/111.229.189.98:27010,111.229.189.98:27011,111.229.189.98:2
7012");
```

查看分片状态：

```
sh.status();
```

mongos 可以用同样的方式，创建多个

创建分片表：

1. MongoDB的分片时基于集合的，就算有分片集群不等于数据会自动分片，需要显示分片表

首先需要 启用数据库分片

sh.enableSharding("库名");

如：

sh.enableSharding("order");

sh.shardCollection("库名.集合名",{_id: "hashed"});
sh.shardCollection("order.account",{_id: "hashed"});

添加一组数据

```
for( var i =0;i<100; i++){
db.productdesc.insert({i:i});
}
```

动态扩容

创建第二个复制集来实现分片

```
mongod --bind_ip 0.0.0.0 --replSet shard2 --dbpath /data/shard2 --logpath /dat
a/shard2/mongod.log --port 27013 --fork --shardsvr
```

```
mongod --bind_ip 0.0.0.0 --replSet shard2 --dbpath /data/shard2second1 --logpa
th /data/shard2second1/mongod.log --port 27014 --fork --shardsvr
```

```
mongod --bind_ip 0.0.0.0 --replSet shard2 --dbpath /data/shard2second2 --logpa
th /data/shard2second2/mongod.log --port 27015 --fork --shardsvr
```

初始化复制集

```
rs.initiate(
{_id:"shard2",
 "members":[
 {"_id":0,"host":"111.229.189.98:27013"},
 {"_id":1,"host":"111.229.189.98:27014"},
 {"_id":2,"host":"111.229.189.98:27015"}
 ]
});
```

加入到集群分片

```
mongo --port 4000
```

```
sh.addShard("shard2/111.229.189.98:27013,111.229.189.98:27014,111.229.189.98:27015");
```

课上异常说明:

https://www.cnblogs.com/lazyboy/archive/2012/11/26/2789401.html

```
mongos> db.account.drop();
true
mongos> sh.shardCollection("order.account",{_id: "hashed"});
{
        "collectionsharded" : "order.account",
        "collectionUUID" : UUID("d40bc0f1-aa5c-4ba2-91bb-e5a4eb2fcb20"),
        "ok" : 1,
        "operationTime" : Timestamp(1609079316, 13),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1609079316, 13),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

前面有数据了， 再分配分片键会报错

MongoDB 应用实战

最新驱动的地址
https://mongodb.github.io/mongo-java-driver/4.1/driver/

1. java 原生客户端
引入maven

```
<dependencies>
 <dependency>
 <groupId>org.mongodb</groupId>
 <artifactId>mongodb-driver-sync</artifactId>
 <version>4.1.1</version>
 </dependency>
</dependencies>
```

```java
public class QuickStart {

 public static void main(String[] args) {
  // 连接本地默认端口的Mongod
  // MongoClient mongoClient = MongoClients.create()
  // 连接远程服务的指定端口的Mongod
  // MongoClient mongoClient = MongoClients.create("mongodb://host1:27017");
  // 连接指定端口复制集
  // MongoClient mongoClient = MongoClients.create("mongodb://host1:27017,host2:27017,host3:27017/?replicaSet=myReplicaSet");
  // 连接 mongos路由: 连接一个
  // MongoClient mongoClient = MongoClients.create( "mongodb://localhost:27017");
  // 连接多个mongos路由
  MongoClient mongoClient = MongoClients.create("mongodb://111.229.189.98:4000");


  //获取数据库
  MongoDatabase database = mongoClient.getDatabase("productdb");

  // 获取集合
  MongoCollection<Document> productdesc=database.getCollection( "productdesc");

  Document doc = new Document("name", "MongoDB")
  .append("type", "database")
  .append("count", 1)
  .append("versions", Arrays.asList("v3.2", "v3.0", "v2.6"))
  .append("info", new Document("x", 203).append("y", 102));

  productdesc.insertOne(doc);


  Bson eq = eq("name", "MongoDB");
  FindIterable<Document> find = productdesc.find(eq);
  Document first=find.first();
  System.out.println(first);

  }
}
```

2. Spring Boot 整合:

https://docs.spring.io/spring-data/mongodb/docs/3.1.2/reference/html/#preface

1. 引入maven

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <parent>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-parent</artifactId>
 <version>2.4.0</version>
 <relativePath/> <!-- lookup parent from repository -->
 </parent>
 <groupId>com.example</groupId>
 <artifactId>demo</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <name>demo</name>
 <description>Demo project for Spring Boot</description>

 <properties>
 <java.version>1.8</java.version>
 </properties>

 <dependencies>
 <dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-data-mongodb</artifactId>
 </dependency>
 <dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-web</artifactId>
 </dependency>

 <dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-test</artifactId>
 <scope>test</scope>
 </dependency>
 <dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
```

```
<version>1.18.12</version>
</dependency>


</dependencies>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>

</project>
```

2. 添加配置类

比如连接上述的 mongos, 使用MongoTemplate 进行数据库操作

```
package com.example.demo.config;

import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.mongodb.core.MongoTemplate;

@Configuration
 public class AppConfig {

  public @Bean
  MongoClient mongoClient() {
  return MongoClients.create("mongodb://111.229.189.98:4000");
  }

  public @Bean
  MongoTemplate mongoTemplate() {
  return new MongoTemplate(mongoClient(), "productdb");
  }
 }
```

3. 测试类

```java
package com.example.demo;

public class Person {

 private String id;
 private String name;
 private int age;

 public Person(String name, int age) {
  this.name = name;
  this.age = age;
  }

  public String getId() {
  return id;
  }
  public String getName() {
  return name;
  }
  public int getAge() {
  return age;
  }

  @Override
  public String toString() {
  return "Person [id=" + id + ", name=" + name + ", age=" + age + "]";
  }

}

package com.example.demo;

import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.stereotype.Component;

import java.util.List;

import static org.springframework.data.mongodb.core.query.Criteria.where;
import static org.springframework.data.mongodb.core.query.Query.query;
```

```
import static org.springframework.data.mongodb.core.query.Update.update;

@Component
@Slf4j
public class ApplicationRunnerTest implements ApplicationRunner{

 @Autowired
 private MongoTemplate mongoOps;

 @Override
 public void run(ApplicationArguments applicationArguments) throws Exception {

 Person p = new Person("Joe", 34);

 // 插入文档
 mongoOps.insert(p);
 log.info("Insert: " + p);

 // 查询文档
 p = mongoOps.findById(p.getId(), Person.class);
 log.info("Found: " + p);

 // 更新文档
 mongoOps.updateFirst(query(where("name").is("Joe")), update("age", 35), Perso
n.class);
 p = mongoOps.findOne(query(where("name").is("Joe")), Person.class);
 log.info("Updated: " + p);

 // 删除文档
 mongoOps.remove(p);

 // Check that deletion worked
 List<Person> people = mongoOps.findAll(Person.class);
 log.info("Number of people = : " + people.size());
 mongoOps.dropCollection(Person.class);
 }
}
```

文档：VIP-03 MongoDB复制集，分片集群.note
链接：http://note.youdao.com/noteshare?id=f42ca06d983dd456e9d158bcbb025477&sub=3778E316E3724C72A356D0EDD23FEBB6