



## 深入理解MongoDB分片的管理

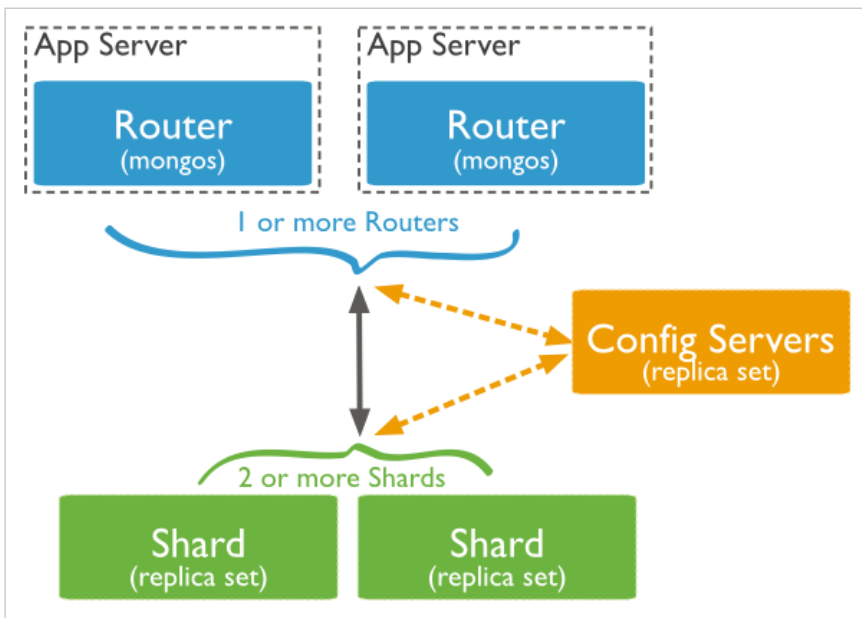
投稿: daisy 字体: [增加 减小] 类型: 转载 时间: 2016-09-18 我要评论

这篇文章带大家深入理解MongoDB分片的管理, 将通过主分片、分片的元数据、删除分片、增加分片、特大大块以及均衡器这几方面来详细介绍, 对大家的学习理解很有帮助, 有需要的可以参考借鉴。

### 前言

在MongoDB (版本 3.2.9) 中, 分片集群 (sharded cluster) 是一种水平扩展数据库系统性能的方法, 能够将数据集分布式存储在不同的分片 (shard) 上, 每个分片只保存数据集的一部分, MongoDB保证各个分片之间不会有重复的数据, 所有分片保存的数据之和就是完整的数据集。分片集群将数据集分布式存储, 能够将负载分摊到多个分片上, 每个分片只负责读写一部分数据, 充分利用了各个shard的系统资源, 提高数据库系统的吞吐量。

数据集被拆分成数据块 (chunk), 每个数据块包含多个doc, 数据块分布式存储在分片集群中。MongoDB负责追踪数据块在shard上的分布信息, 每个分片存储哪些数据块, 叫做分片的元数据, 保存在config server上的数据库 config中, 一般使用3台config server, 所有config server中的config数据库必须完全相同。通过mongo能够直接访问数据库config, 查看分片的元数据; mongo shell 提供 sh 辅助函数, 能够安全地查看分片集群的元数据信息。



对任何一个shard进行查询, 只会获取collection在当前分片上的数据子集, 不是整个数据集。Application 只需要连接到mongos, 对其进行的读写操作, mongos自动将读写请求路由到相应的shard。MongoDB通过mongos将分片的底层实现对Application透明, 在Application看来, 访问的是整个数据集。

### 一、主分片

在分片集群中, 不是每个集合都会分布式存储, 只有使用sh.shardCollection()显式将collection分片后, 该集合才会分布式存储在不同的shard中。对于非分片集合 (un-sharded collection), 其数据只会存储在主分片 (Primary shard) 中, 默认情况下, 主分片是指数据库最初创建的shard, 用于存储该数据库中非分片集合的数据。每个数据库都有一个主分片。

## 大家感兴趣的内容

- 1 MongoDB常用操作命令大全
- 2 MongoDB数据库插入、更新和删除操作
- 3 mongodb 数据库操作--备份 还原
- 4 MongoDB各种查询操作详解
- 5 MongoDB插入数据的3种方法
- 6 MongoDB整库备份与还原以及单个c
- 7 mongodb 实现远程连接
- 8 MongoDB查询操作限制返回字段的方
- 9 PHP中MongoDB数据库的连接、添加
- 10 mongodb中使用distinct去重的简单

## 最近更新的内容

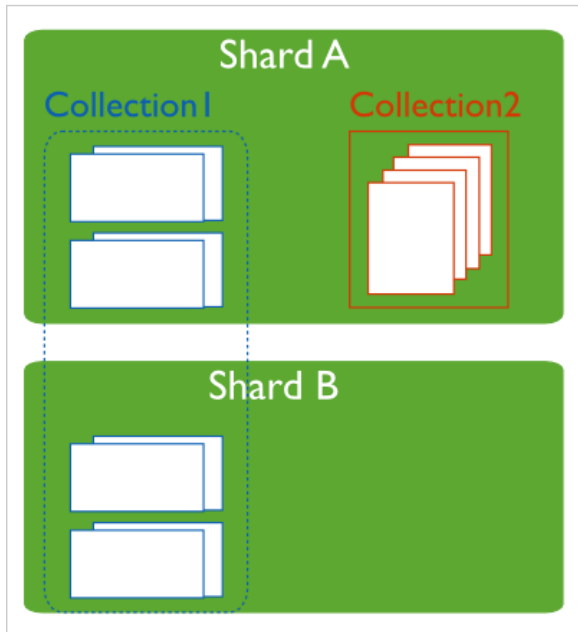
- centos yum 安装 mongodb 以及php扩展
- MongoDB查询技巧总结
- Mongo管理用户相关操作总结
- mongodb 数据类型(null/字符串/数字/日
- mongodb 添加用户及权限设置详解
- CentOS 6.4创建Mongodb副本集
- Java操作MongoDB数据库示例分享
- mongodb eval 执行服务器端脚本
- MongoDB查询字段没有创建索引导致的追
- Mongodb 删除添加分片与非分片表维护

## 常用在线小工具

Each database in a sharded cluster has a primary shard that holds all the un-sharded collections for that database. Each database has its own primary shard.

例如，一个分片集群有三个分片：shard1, shard2, shard3，在分片shard1创建一个数据库blog。如果将数据库blog分片，那么MongoDB会自动在shard2, shard3上创建一个结构相同的数据库blog，数据库blog的Primary Shard是Shard1。

图示，Collection2的主分片是ShardA。



使用 `movePrimary` 命令变更数据库默认的Primary shard，非分片集合将会从当前shard移动到新的主分片。

```
1 | db.runCommand( { movePrimary : "test", to : "shard0001" } )
```

在使用movePrimary命令变更数据库的主分片之后，config server中的配置信息是最新的，mongos缓存的配置信息变得过时了。MongoDB提供命令：`flushRouterConfig` 强制mongos从config server获取最新的配置信息，刷新mongos的缓存。

```
1 | db.adminCommand({ "flushRouterConfig":1})
```

## 二，分片的元数据

不要直接到config server上查看分片集群的元数据信息，这些数据非常重要，安全的方式是通过mongos连接到config数据查看，或者使用sh辅助函数查看。

使用sh辅助函数查看

```
1 | sh.status()
```

连接到mongos查看config数据库中的集合

```
1 | mongos> use config
```

### 1，shards 集合保存分片信息

```
1 | db.shards.find()
```

shard的数据存储在host指定的 replica set 或 standalone mongod中。

```
1 | {
2 |   "id" : "shard_name",
3 |   "host" : "replica_set_name/host:port",
4 |   "tag": [shard_tag1,shard_tag2]
5 | }
```

### 2，databases集合保存分片集群中所有数据库的信息，不管数据库是否分片

```
1 | db.databases.find()
```

如果在数据库上执行`sh.enableSharding("db_name")`，那么字段partitioned字段值就是true；primary 字段指定数据库的主分片（primary shard）。

```

1 {
2   "_id" : "test",
3   "primary" : "rs0",
4   "partitioned" : true
5 }

```

### 3, collections集合保存所有已分片集合的信息，不包括非分片集合（un-sharded collections）

key是：分片的片键

```

1 db.collections.find()
2
3 {
4   "_id" : "test.foo",
5   "lastmodEpoch" : ObjectId("57dcd4899bd7f7111ec15f16"),
6   "lastmod" : ISODate("1970-02-19T17:02:47.296Z"),
7   "dropped" : false,
8   "key" : {
9     "_id" : 1
10  },
11   "unique" : true
12 }

```

### 4, chunks 集合保存数据块信息，

ns: 分片的集合，结构是：db\_name.collection\_name

min 和 max: 片键的最小值和最大值

shard: 块所在的分片

```

1 db.chunks.find()
2
3 {
4   "_id" : "test.foo-_id_MinKey",
5   "lastmod" : Timestamp(1, 1),
6   "lastmodEpoch" : ObjectId("57dcd4899bd7f7111ec15f16"),
7   "ns" : "test.foo",
8   "min" : {
9     "_id" : 1
10  },
11   "max" : {
12     "_id" : 3087
13  },
14   "shard" : "rs0"
15 }

```

### 5, changelog集合记录分片集群的操作，包括chunk的拆分和迁移操作，Shard的增加或删除操作

what 字段：表示操作的类型，例如：multi-split表示chunk的拆分，

```

1 "what" : "addShard",
2 "what" : "shardCollection.start",
3 "what" : "shardCollection.end",
4 "what" : "multi-split",

```

### 6, tags 记录shard的tag和对应的片键范围

```

1 {
2   "_id" : { "ns" : "records.users", "min" : { "zipcode" : "10001" } },
3   "ns" : "records.users",
4   "min" : { "zipcode" : "10001" },
5   "max" : { "zipcode" : "10281" },
6   "tag" : "NYC"
7 }

```

### 7, settings 集合记录均衡器状态和chunk的大小，默认的chunk size是64MB。

```

1 { "_id" : "chunksize", "value" : 64 }
2 { "_id" : "balancer", "stopped" : false }

```

### 8, locks 集合记录分布式锁（distributed lock），保证只有一个mongos实例能够在分片集群中执行管理任务。

mongos在担任balancer时，会获取一个分布式锁，并向config.locks中插入一条doc。

The locks collection stores a distributed lock. This ensures that only one mongos instance can perform administrative task s on the cluster at once. The mongos acting as balancer takes a lock by inserting a document resembling the following into the locks collection.

```

1 {
2   "_id" : "balancer",
3   "process" : "example.net:40000:1350402818:16807",
4   "state" : 2,
5   "ts" : ObjectId("507daedf40e1879df62e5f3"),

```

```

6 | "when" : ISODate("2012-10-16T19:01:01.593Z"),
7 | "who" : "example.net:40000:1350402818:16807:Balancer:282475249",
8 | "why" : "doing balance round"
9 | }

```

### 三，删除分片

删除分片时，必须确保该分片上的数据被移动到其他分片中，对于以分片的集合，使用均衡器来迁移数据块，对于非分片的集合，必须修改集合的主分片。

#### 1，删除已分片的集合数据

step1，保证均衡器是开启的

```
1 | sh.setBalancerState(true);
```

step2，将已分片的集合全部迁移到其他分片

```

1 | use admin
2 | db.adminCommand({"removeShard":"shard_name"})

```

removeShard命令会将数据块从当前分片上迁移到其他分片上去，如果分片上的数据块比较多，迁移过程可能耗时很长。

step3，检查数据块迁移的状态

```

1 | use admin
2 | db.runCommand( { removeShard: "shard_name" } )

```

使用 `removeShard` 命令能够查看数据块迁移的状态，`remaining` 字段表示剩余数据块的数量

```

1 | {
2 |   "msg" : "draining ongoing",
3 |   "state" : "ongoing",
4 |   "remaining" : {
5 |     "chunks" : 42,
6 |     "dbs" : 1
7 |   },
8 |   "ok" : 1
9 | }

```

step4，数据块完成迁移

```

1 | use admin
2 | db.runCommand( { removeShard: "shard_name" } )
3 |
4 | {
5 |   "msg" : "removeshard completed successfully",
6 |   "state" : "completed",
7 |   "shard" : "shard_name",
8 |   "ok" : 1
9 | }

```

#### 2，删除未分片的数据库

step1，查看未分片的数据库

未分片的数据库，包括两部分：

- 1、数据库未被分片，该数据没有使用 `sh.enableSharding("db_name")`，在数据库config中，该数据库的 `partitioned` 字段是 `false`
- 2、数据库中存在collection未被分片，即当前的分片是该集合的主分片

```

1 | use config
2 | db.databases.find({$or:[{"partitioned":false},{ "primary":"shard_name"}]})

```

对于 `partitioned=false` 的数据库，其数据全部保存在当前shard中；对于 `partitioned=true`，`primary="shard_name"` 的数据库，表示存在未分片（un-sharded collection）存储在该数据库中，必须变更这些集合的主分片。

step2，修改数据库的主分片

```
1 | db.runCommand( { movePrimary: "db_name", to: "new_shard" })
```

### 四，增加分片

由于分片存储的是数据集的一部分，为了保证数据的高可用性，推荐使用Replica Set作为shard，即使Replica Set中只包含一个成员。连接到mongos，使用sh辅助函数增加分片。

```
1 | sh.addShard("replica_set_name/host:port")
```

不推荐将standalone mongod作为shard

```
1 | sh.addShard("host:port")
```

## 五, 特大块

在有些情况下, chunk会持续增长, 超出chunk size的限制, 成为特大块(jumbo chunk), 出现特大块的原因是chunk中的所有doc使用同一个片键(shard key), 导致MongoDB无法拆分该chunk, 如果该chunk持续增长, 将会导致chunk的分布不均匀, 成为性能瓶颈。

在chunk迁移时, 存在限制: 每个chunk的大小不能超过2.5万条doc, 或者1.3倍于配置值。chunk size默认的配置值是64MB, 超过限制的chunk会被MongoDB标记为特大块(jumbo chunk), MongoDB不能将特大块迁移到其他shard上。

MongoDB cannot move a chunk if the number of documents in the chunk exceeds either 250000 documents or 1.3 times the result of dividing the configured chunk size by the average document size.

### 1, 查看特大块

使用sh.status(), 能够发现特大块, 特大块的后面存在 jumbo 标志

```
1 | { "x" : 2 } --> { "x" : 3 } on : shard-a Timestamp(2, 2) jumbo
```

### 2, 分发特大块

特大块不能拆分, 不能通过均衡器自动分发, 必须手动分发。

#### step1, 关闭均衡器

```
1 | sh.setBalancerState(false)
```

#### step2, 增大Chunk Size的配置值

由于MongoDB不允许移动大小超出限制的特大块, 因此, 必须临时增加chunk size的配置值, 再将特大块均衡地分发到分片集群中。

```
1 | use config
2 | db.settings.save({"_id":"chunksize","value":"1024"})
```

#### step3, 移动特大块

```
1 | sh.moveChunk("db_name.collection_name",{sharded_file:"value_in_chunk"},"new"
```

#### step4, 启用均衡器

```
1 | sh.setBalancerState(true)
```

#### step5, 刷新mongos的配置缓存

强制mongos从config server同步配置信息, 并刷新缓存。

```
1 | use admin
2 | db.adminCommand({ flushRouterConfig: 1 })
```

## 六, 均衡器

均衡器是由mongos转变的, 就是说, mongos不仅负责将查询路由到相应的shard上, 还要负责数据块的均衡。一般情况下, MongoDB会自动处理数据均衡, 通过 config.settings 能够查看balancer的状态, 或通过sh辅助函数查看

```
1 | sh.getBalancerState()
```

返回true, 表示均衡器在正运行, 系统自动处理数据均衡, 使用sh辅助函数能够关闭balancer

```
1 | sh.setBalancerState(false)
```

balancer不能立即终止正在运行的块迁移操作, 在mongos转变为balancer时, 会申请一个balancer lock, 查看config.locks 集合,

```
1 | use config
2 | db.locks.find({"_id":"balancer"})
3 |
```

```
4  --or
5  sh.isBalancerRunning()
```

如果state=2，表示balancer正处于活跃状态，如果state=0，表示balancer已被关闭。

均衡过程实际上是将数据块从一个shard迁移到其他shard，或者先将一个大的chunk拆分小的chunk，再将小块迁移到其他shard上，块的迁移和拆分都会增加系统的IO负载，最好将均衡器的活跃时间限制在系统空闲时进行，可以设置balancer的活跃时间窗口，限制balancer在指定的时间区间内进行数据块的拆分和迁移操作。

```
1  use config
2
3  db.settings.update(
4    {"_id":"balancer"},
5    {"$set":{"activeWindow":{"start":"23:00","stop":"04:00"}}},
6    true
7  )
```

均衡器拆分和移动的对象是chunk，均衡器只保证chunk数量在各个shard上是均衡的，至于每个chunk包含的doc数量，并不一定是均衡的。可能存在一些chunk包含的doc数量很多，而有些chunk包含的doc数量很少，甚至不包含任何doc。因此，应该慎重选择分片的索引键，即片键，如果一个字段既能满足绝大多数查询的需求，又能使doc数量均匀分布，那么该字段是片键的最佳选择。

总结

以上就是这篇文章的全部内容，希望对大家的学习或者工作带来一定的帮助，如果有疑问的大家可以留言交流。

您可能感兴趣的文章：

- [mongodb中随机获取1条记录的实现方法](#)
- [1亿条记录的MongoDB数据库随机查询性能测试](#)
- [MongoDB 导出导入备份恢复数据详解及实例](#)
- [mongodb 3.2.5安装详细过程](#)
- [MongoDB 主从复制实例讲解](#)
- [python实现爬虫数据存到 MongoDB](#)
- [MongoDB windows解压缩版安装教程详解](#)
- [Yii框架连接mongodb数据库的代码](#)
- [MongoDB安装图文教程](#)
- [ASP.NET MVC4使用MongoDB制作相册管理](#)
- [mongodb 随机获取一条记录的方法](#)


Tags: mongodb 分片

相关文章



- |  |            |
|--|------------|
| ▪ <a href="#">MongoDB的索引</a>                                       | 2017-05-05 |
| ▪ <a href="#">Ubuntu下安装mongodb 3.4的详细过程</a>                        | 2017-01-01 |
| ▪ <a href="#">Ubuntu 14.04 安装 MongoDB 及 PHP MongoDB Driver详细介绍</a> | 2016-10-10 |
| ▪ <a href="#">Java操作MongoDB数据库示例分享</a>                             | 2014-08-08 |
| ▪ <a href="#">MongoDB 语法使用小结</a>                                   | 2011-10-10 |
| ▪ <a href="#">高效mongodb的php分页类（不使用skip）</a>                        | 2014-05-05 |
| ▪ <a href="#">MongoDB 学习笔记(一)-MongoDB配置</a>                        | 2016-05-05 |
| ▪ <a href="#">MongoDB入门教程之索引操作浅析</a>                               | 2014-08-08 |
| ▪ <a href="#">mongodb 数据库操作--备份 还原 导出 导入</a>                       | 2014-07-07 |
| ▪ <a href="#">Mongodb安装与配置笔记</a>                                   | 2014-09-09 |

最新评论


评论(0人参与 , 0条评论)





来说两句吧...



发布

 微博登录

 QQ登录

 手机登录

还没有评论，快来抢沙发吧！

Powered by 畅言