mongo分片

1.分片简介

分片（sharding)是指将数据拆分，将其分散存放在不同的机器上的过程。有时也用分区（partitioning)来表示这个概念。

为了对应用程序隐藏数据库架构的细节，在分片之前要先执行mongos进行一次路由过程。这个路由服务器维护着一个“内容列表”，指明了每个分片包含什么数据内容。应用程序只需要连接到路由服务器，就可以像使用单机服务器一样进行正常的请求了，如图13-1所示。路由服务器知道哪些数据位于哪个分片，可以将请求转发给相应的分片。每个分片对请求的响应都会发送给路由服务器，路由服务器将所有响应合并在一起，返回给应用程序。对应用程序来说，它只知道自己是连接到了一台单机mongod服务器

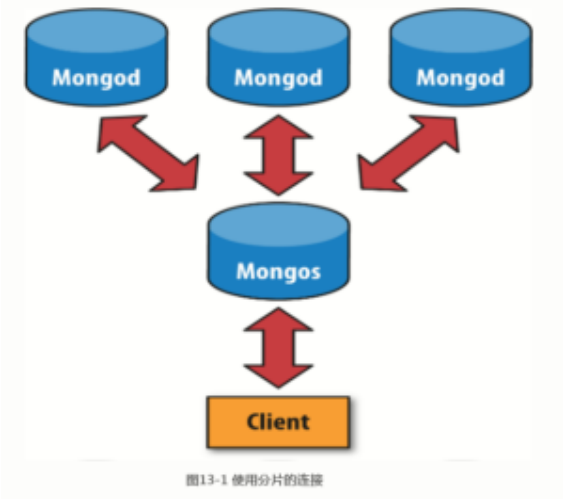


图13-1 使用分片的连接

1.分片如何瓜分数据

片键（shard key）：片键是集合的一个键，MongoDB根据这个键拆分数据。选择片键可以认为是选择集合中数据的顺序 ，例如 "al-steak-sauce"到"defcon"位于第一片，"defcon1"到"howiel998"位于第二片。就是简单粗暴的把首字母a--d为一个片，d—h 为一个片。。。。只有被索引过的键才能够作为片键。

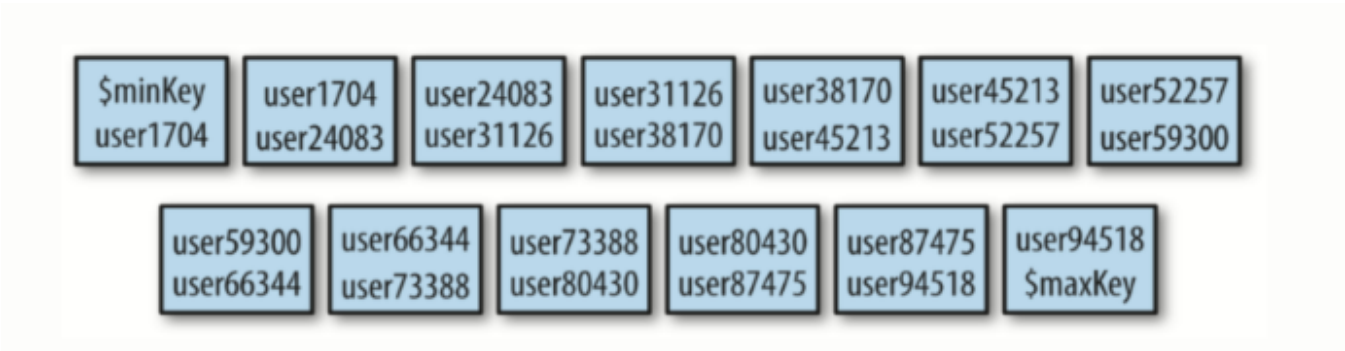
先将某个字段设置为索引，

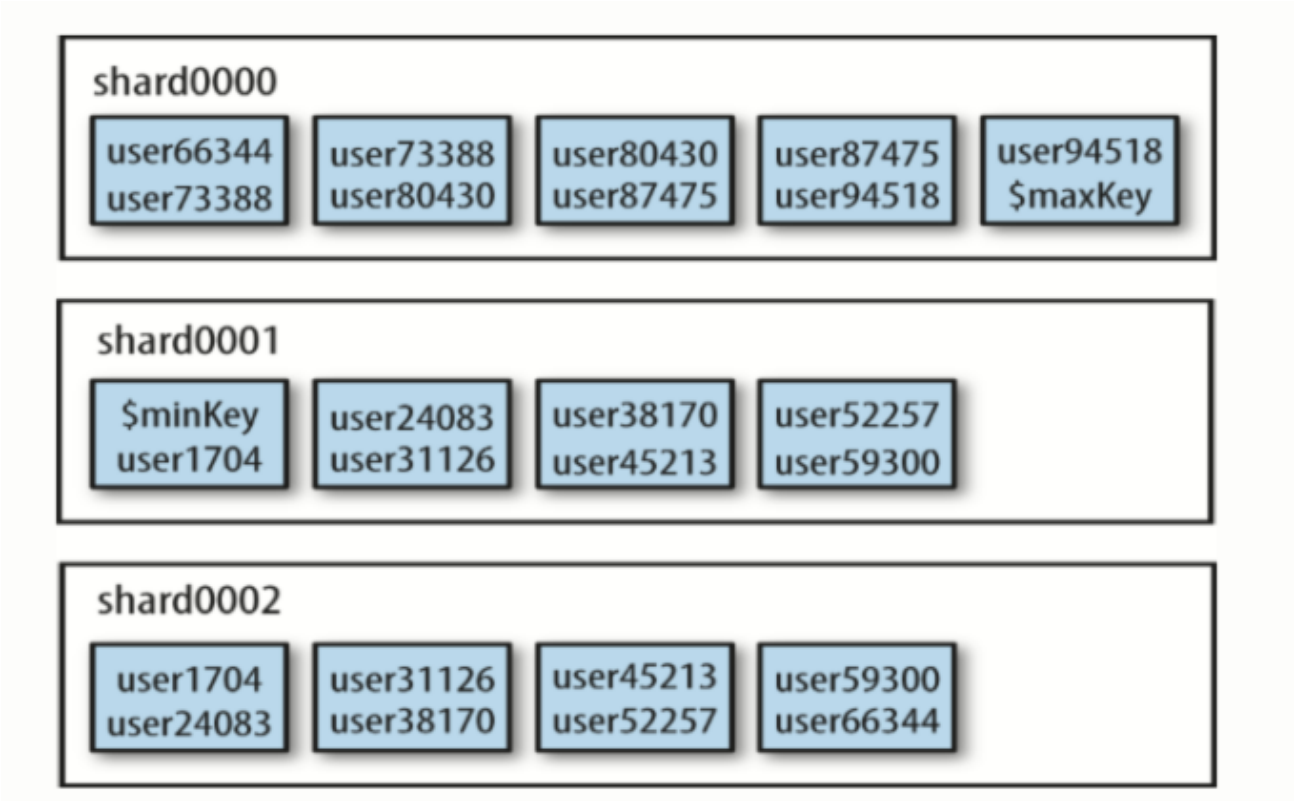
```
db.users.ensureIndex({"username" : 1});
```

然后在用这个字段的value进行划分分片的范围

mongo 中 sh.status 可以查询分片的情况

```
--- Sharding Status ---
  sharding version: { "_id" : 1, "version" : 3 }
  shards:
    {  "_id" : "shard0000",  "host" : "localhost:30000" }
    {  "_id" : "shard0001",  "host" : "localhost:30001" }
    {  "_id" : "shard0002",  "host" : "localhost:30002" }
  databases:
    {  "_id" : "admin",  "partitioned" : false,  "primary" : "config" }
    {  "_id" : "test",  "partitioned" : true,  "primary" : "shard0000" }
        test.users chunks:
            shard0001    4
            shard0002    4
            shard0000    5
        { "username" : { $minKey : 1 } } -->> { "username" : "user1704" }
            on : shard0001
        { "username" : "user1704" } -->> { "username" : "user24083" }
            on : shard0002
        { "username" : "user24083" } -->> { "username" : "user31126" }
            on : shard0001
        { "username" : "user31126" } -->> { "username" : "user38170" }
            on : shard0002
        { "username" : "user38170" } -->> { "username" : "user45213" }
            on : shard0001
        { "username" : "user45213" } -->> { "username" : "user52257" }
            on : shard0002
        { "username" : "user52257" } -->> { "username" : "user59300" }
            on : shard0001
        { "username" : "user59300" } -->> { "username" : "user66344" }
            on : shard0002
        { "username" : "user66344" } -->> { "username" : "user73388" }
            on : shard0000
        { "username" : "user73388" } -->> { "username" : "user80430" }
            on : shard0000
        { "username" : "user80430" } -->> { "username" : "user87475" }
            on : shard0000
        { "username" : "user87475" } -->> { "username" : "user94518" }
            on : shard0000
        { "username" : "user94518" } -->> { "username" : { $maxKey : 1 } }
            on : shard0000
```






```
                } on : shard3 Timestamp(1, 5)
        wisteria_assets.docker_web_app
                shard key: { "agentId" : 1 }
                unique: false
                balancing: true
                chunks:
                        shard2  1
                { "agentId" : { "$minKey" : 1 } } -->> { "agentId" : { "$maxKey"
                Timestamp(1, 0)
        wisteria_assets.linux_account
                shard key: { "agentId" : 1 }
                unique: false
                balancing: true
                chunks:
                        shard2  1
                { "agentId" : { "$minKey" : 1 } } -->> { "agentId" : { "$maxKey"
                Timestamp(1, 0)
        wisteria_assets.linux_account_group
                shard key: { "agentId" : 1 }
                unique: false
                balancing: true
                chunks:
                        shard2  1
```

以上的方式是范围瓜分 递增片键 的情况，

这种瓜分方式有好处也有坏处，好处就是查询连体数据方便，请求发给mongos 后，mongos路由往往只需要发送一次消息给分片，就可以回去所有的数据；但是 插入数据效率不高，如果数据书连体的，那么它大概是处于同一个shard,甚至于处于同一个chunk 默认64MB，那么mongos 就只会和一个shard通信，其他的shard 就站在干岸上，写入不具有分散性，这会导致单台服务器压力较大，无法提高效率；

还要有一种是哈希片键 ，也称之为散列索引，使用一个哈希索引字段作为片键，优点是使数据在各节点分布比较均匀，数据写入可随机分发到每个分片服务器上，把写入的压力分散到了各个服务器上。但是读也是随机的，可能会命中更多的分片，但是缺点是无法实现范围区分

组合片键： 片键基数太小（即变化少如星期只有7天可变化），可以选另一个字段使用合片组键

标签片键： 可以为分片添加tag标签，然后指定相应的tag，比如让10.*.*.*(T)出现在shard0000上

1.怎么选择片键

无非从两个方面考虑，数据的查询和写入，最好的效果就是数据查询时能命中更少的分片，数据写入时能够随机的写入每个分片，关键在于如何权衡性能和负载。