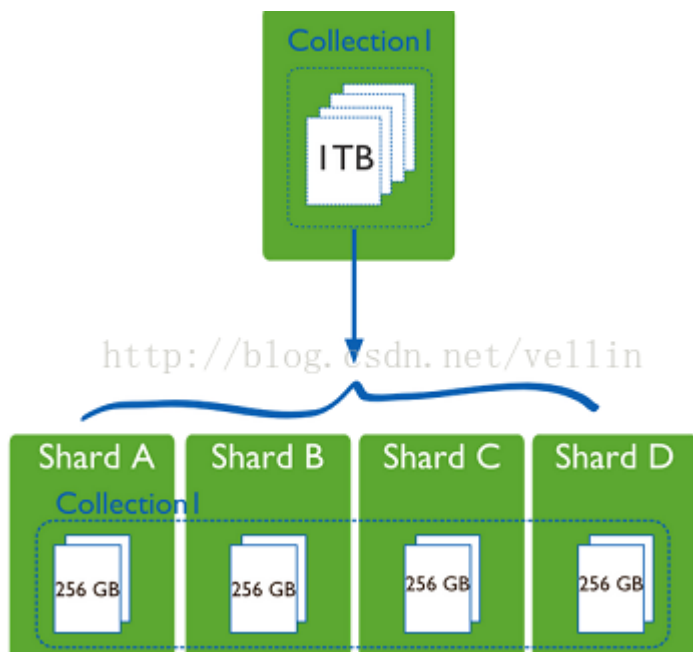


mongodb集群搭建 (sharding集群)

mongos：数据库集群请求的入口，所有的请求都通过mongos进行协调，不需要在应用程序添加一个路由选择器，mongos自己就是一个请求分发中心，它负责把对应的数据请求请求转发到对应的shard服务器上。在生产环境通常有多mongos作为请求的入口，防止其中一个挂掉所有的mongodb请求都没有办法操作。

config server：顾名思义为配置服务器，存储所有数据库元信息（路由、分片）的配置。mongos本身没有物理存储分片服务器和数据路由信息，只是缓存在内存里，配置服务器则实际存储这些数据。mongos第一次启动或者关掉重启就会从 config server 加载配置信息，以后如果配置服务器信息变化会通知到所有的 mongos 更新自己的状态，这样 mongos 就能继续准确路由。在生产环境通常有多个 config server 配置服务器，因为它存储了分片路由的元数据，这个可不能丢失！就算挂掉其中一台，只要还有存货，mongodb集群就不会挂掉。

shard：这就是传说中的分片了。上面提到一个机器就算能力再大也有天花板，就像军队打仗一样，一个人再厉害喝血瓶也拼不过对方的一个师。俗话说三个臭皮匠顶个诸葛亮，这个时候团队的力量就凸显出来了。在互联网也是这样，一台普通的机器做不了的多台机器来做，如下图：



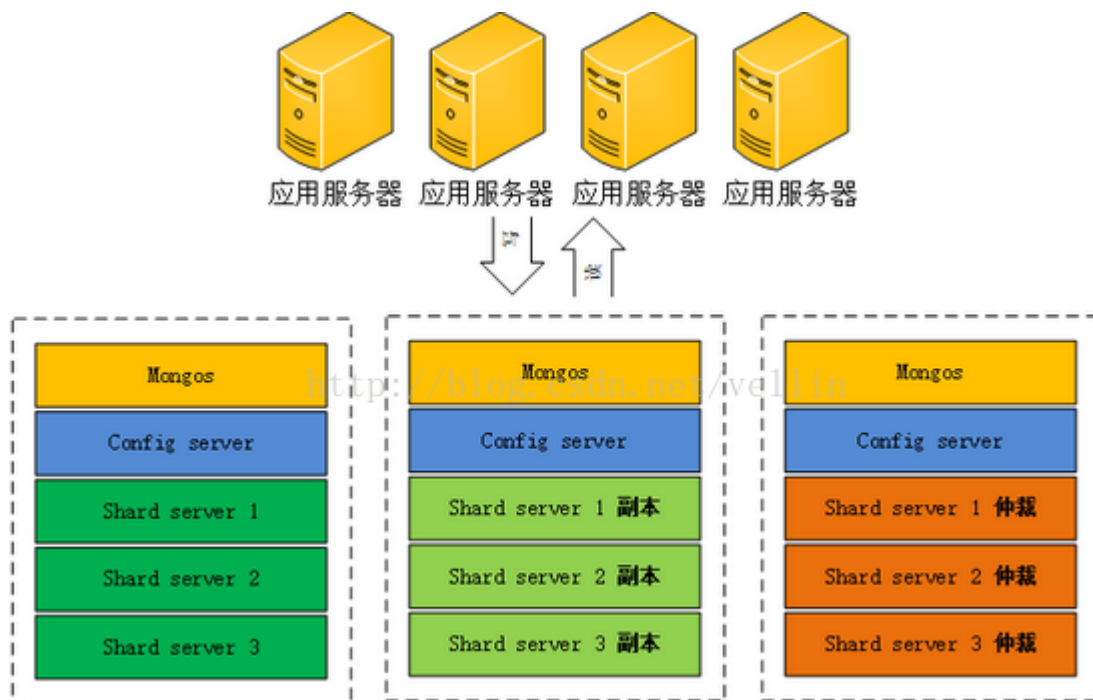
一台机器的一个数据表 Collection1 存储了 1T

数据，压力太大了！在分给4个机器后，每个机器都是256G，则分摊了集中在一台机器的压力。也许有人问一台机器硬盘加大一点不就可以了，为什么要分给四台机器呢？不要光想到存储空间，实际运行的数据库还有硬盘的读写、网络的IO、CPU和内存的瓶颈。在mongodb集群只要设置好了分片规则，通过mongos操作数据库就能自动把对应的数据操作请求转发到对应的分片机器上。在生产环境中分片的片键可要好好设置，这个影响到了怎么把数据均匀分到多个分片机器上，不要出现其中一台机器分了1T，其他机器没有分到的情况，这样还不如不分片！

replica set：前面已经详细讲过了这个东东，怎么这里又来凑热闹！其实上图4个分片如果没有 replica set 是个不完整架构，假设其中的一个分片挂掉那四分之一的数据就丢失了，所以在高可用性的分片架构还需要对于每一个分片构建 replica set 副本集保证分片的可靠性。生产环境通常是 2个副本 + 1个仲裁。

说了这么多，还是来实战一下如何搭建高可用的mongodb集群：

首先确定各个组件的数量，mongos 3个， config server 3个，数据分3片 shard server 3个，每个shard 有一个副本一个仲裁也就是 $3 * 2 = 6$ 个，总共需要部署15个实例。这些实例可以部署在独立机器也可以部署在一台机器，我们这里测试资源有限，只准备了3台机器，在同一台机器只要端口不同就可以，看一下物理部署图：



1、准备机器，IP分别设置为：172.17.0.2、172.17.0.3、172.17.0.4，并分别创建mongos、config、shard1、shard2、shard3对应目录

```
#建立mongos及日志目录
mkdir -p /data/mongodbttest/mongos/log
#建立config server 数据文件存放目录
mkdir -p /data/mongodbttest/config/data
#建立config server 日志文件存放目录
mkdir -p /data/mongodbttest/config/log
#建立shard1 数据文件存放目录
mkdir -p /data/mongodbttest/shard1/data
#建立shard1 日志文件存放目录
mkdir -p /data/mongodbttest/shard1/log
#建立shard2 数据文件存放目录
mkdir -p /data/mongodbttest/shard2/data
#建立shard2 日志文件存放目录
mkdir -p /data/mongodbttest/shard2/log
#建立shard3 数据文件存放目录
mkdir -p /data/mongodbttest/shard3/data
#建立shard3 日志文件存放目录
mkdir -p /data/mongodbttest/shard3/log
```

172.17.0.2、172.17.0.3、172.17.0.4，并分别创建mongos、config、shard1、shard2、shard3对应目录

mongos为20000，config server为21000，shard1为22001，shard2为22002，shard3为22003。

```
docker run -d -p 2220:22 -p 20000 -p 21000 -p 22001 -p 22002 -p 22003 lamp
```

```
docker run -d -p 2221:22 -p 20000 -p 21000 -p 22001 -p 22002 -p 22003 lamp
```

```
docker run -d -p 2222:22 -p 20000 -p 21000 -p 22001 -p 22002 -p 22003 lamp
```

3、在每一台服务器分别启动配置服务器（后台运行）

```
config
```

```
mongod --configsvr --replSet cfgReplSet --dbpath /data/mongodbttest/config/data --port 21000 --logpath /data/mongodbttest/config/log/config.log --fork
```

连接到任意一台配置服务器上

```
mongo --host 172.17.0.2 --port 21000
```

#创建配置服务器副本集

```
rs.initiate({_id:"cfgReplSet",configsvr:true,members:[{_id:0,host:"172.17.0.2:21000"},{_id:1,host:"172.17.0.3:21000"},{_id:2,host:"172.17.0.4:21000"}]})
```

4、在每一台服务器分别启动分片及副本集。（后台运行）

#在每一台服务器分别以副本集方式启动分片1

```
mongod --shardsvr --replSet shard1ReplSet --port 22001 --dbpath /data/mongodbttest/shard1/data --logpath /data/mongodbttest/shard1/log/shard1.log --fork --nojournal
```

```
mongod --shardsvr --replSet shard2ReplSet --port 22002 --dbpath /data/mongodbttest/shard2/data --logpath /data/mongodbttest/shard2/log/shard2.log --fork --nojournal
```

```
mongod --shardsvr --replSet shard3ReplSet --port 22003 --dbpath /data/mongodbttest/shard3/data --logpath /data/mongodbttest/shard3/log/shard3.log --fork --nojournal
```

```

#连接任意一台分片服务器
mongo --host 172.17.0.2 --port 22001
use admin
# 创建副本集并初始化
rs.initiate({_id:"shard1ReplSet",members:[{_id:0,host:"172.17.0.2:22001"},{_id:1,host:"172.17.0.3:22001"},{_id:2,host:"172.17.0.4:22001"}]})
mongo --host 172.17.0.3 --port 22002
use admin
rs.initiate({_id:"shard2ReplSet",members:[{_id:0,host:"172.17.0.2:22002"},{_id:1,host:"172.17.0.3:22002"},{_id:2,host:"172.17.0.4:22002"}]})
mongo --host 172.17.0.4 --port 22003
use admin
rs.initiate({_id:"shard3ReplSet",members:[{_id:0,host:"172.17.0.2:22003"},{_id:1,host:"172.17.0.3:22003"},{_id:2,host:"172.17.0.4:22003"}]})

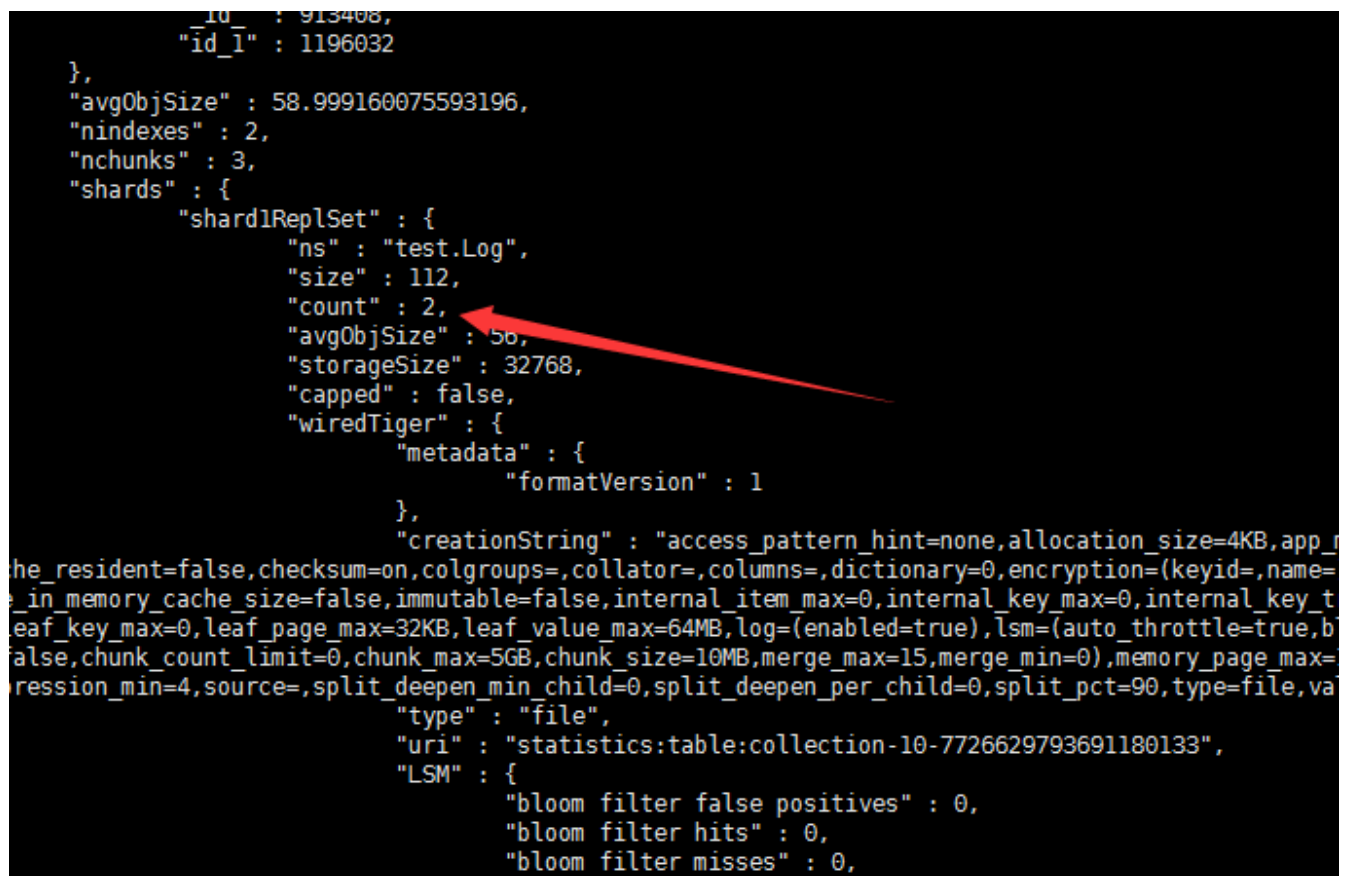
# 5、选择一台服务器启动mongos路由服务。
mongos --configdb cfgReplSet/172.17.0.2:21000,172.17.0.3:21000,172.17.0.4:21000 --port 20000 --logpath
/data/mongodbtest/mongos/log/mongos.log --fork
#登录路由服务客户端
mongo --host 172.17.0.2 --port 20000
#添加分片到集群
sh.addShard("shard1ReplSet/172.17.0.2:22001,172.17.0.3:22001,172.17.0.4:22001")
sh.addShard("shard2ReplSet/172.17.0.2:22002,172.17.0.3:22002,172.17.0.4:22002")
sh.addShard("shard3ReplSet/172.17.0.2:22003,172.17.0.3:22003,172.17.0.4:22003")

#6、 Enable Sharding for a Database
sh.enableSharding("test")

#7、 Shard a Collection
sh.shardCollection("test.Log", { id: 1})
#8、 测试
use test
for(var i = 1; i <= 100000; i++){
  db.Log.save({_id:i,"message":"message"+i});
}

db.Log.stats()

```



```

    "_id" : 913408,
    "_id_1" : 1196032
  },
  "avgObjSize" : 58.999160075593196,
  "nindexes" : 2,
  "nchunks" : 3,
  "shards" : {
    "shard1ReplSet" : {
      "ns" : "test.Log",
      "size" : 112,
      "count" : 2,
      "avgObjSize" : 56,
      "storageSize" : 32768,
      "capped" : false,
      "wiredTiger" : {
        "metadata" : {
          "formatVersion" : 1
        },
        "creationString" : "access_pattern_hint=none,allocation_size=4KB,app_
he_resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0,encryption=(keyid=,name=
e_in_memory_cache_size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_t
leaf_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,b
false,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_max=15,merge_min=0),memory_page_max=
ression_min=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,type=file,va
      "type" : "file",
      "uri" : "statistics:table:collection-10-7726629793691180133",
      "LSM" : {
        "bloom filter false positives" : 0,
        "bloom filter hits" : 0,
        "bloom filter misses" : 0,

```

```

},
"shard2ReplSet" : {
  "ns" : "test.Log",
  "size" : 1466,
  "count" : 26,
  "avgObjSize" : 56,
  "storageSize" : 32768,
  "capped" : false,
  "wiredTiger" : {
    "metadata" : {
      "formatVersion" : 1
    },
    "creationString" : "access_pattern_hint=none,allocation_size=4KB,app_metadata=(f
resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0,encryption=(keyid=,name=),exclu
_memory_cache_size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_truncate
_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom=tr
e,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_max=15,merge_min=0),memory_page_max=10m,os_
sion_min=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,type=file,value_for
"type" : "file",
"uri" : "statistics:table:collection-10--3225313794850134786",
"LSM" : {
  "bloom filter false positives" : 0,
  "bloom filter hits" : 0,
  "bloom filter misses" : 0,
  "bloom filter pages evicted from cache" : 0,
  "bloom filter pages read into cache" : 0,
  "bloom filters in the LSM tree" : 0,
  "chunks in the LSM tree" : 0,
  "highest merge generation in the LSM tree" : 0,
  "queries that could have benefited from a Bloom filter that did not exist in the cache" : 0,
  "sleep for LSM checkpoint throttle" : 0,
  "sleep for LSM merge throttle" : 0,
}
}
}

```

```

},
"shard3ReplSet" : {
  "ns" : "test.Log",
  "size" : 5987821,
  "count" : 99981,
  "avgObjSize" : 59,
  "storageSize" : 1921024,
  "capped" : false,
  "wiredTiger" : {
    "metadata" : {
      "formatVersion" : 1
    },
    "creationString" : "access_pattern_hint=none,allocation_size=4KB,app_metadata=(f
ident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0,encryption=(keyid=,name=),exclusive
nory_cache_size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_truncate=tru
y_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom=true,b
chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_max=15,merge_min=0),memory_page_max=10m,os_cach
n_min=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,type=file,value_format=
"type" : "file",
"uri" : "statistics:table:collection-10-8812553479728450453",
"LSM" : {
  "bloom filter false positives" : 0,
  "bloom filter hits" : 0,
  "bloom filter misses" : 0,
  "bloom filter pages evicted from cache" : 0,
  "bloom filter pages read into cache" : 0,
  "bloom filters in the LSM tree" : 0,
  "chunks in the LSM tree" : 0,
  "highest merge generation in the LSM tree" : 0,
  "queries that could have benefited from a Bloom filter that did not exist in the cache" : 0,
  "sleep for LSM checkpoint throttle" : 0,
  "sleep for LSM merge throttle" : 0,
}
}
}

```

分片不均匀，这是shard key的策略造成的

```
db.Log.drop()
```

```
sh.shardCollection("test.Log",{id:"hashed"})
```

```
db.Log.stats()
```

```

    },
    "shard3ReplSet" : {
      "ns" : "test.Log",
      "size" : 1982453,
      "count" : 33102,
      "avgObjSize" : 59,
      "storageSize" : 651264,
      "capped" : false,
      "wiredTiger" : {
        "metadata" : {
          "formatVersion" : 1
        }
      },
      "creationString" : "access_pattern_hint=none,allocation_size=4KB,app_metadata=none,checksum=on,collator=,columns=,dictionary=0,encryption=(keyid=,name=),exclusive_memory_cache_size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_truncate=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom_filter=false,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_max=15,merge_min=0),memory_page_max=10MB,min_olap=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,type=file,value_format=(key=sstring,value=sstring)",
      "type" : "file",
      "uri" : "statistics:table:collection-13-8812553479728450453",
      "LSM" : {
        "bloom filter false positives" : 0,
        "bloom filter hits" : 0,
        "bloom filter misses" : 0,
        "bloom filter pages evicted from cache" : 0,
        "bloom filter pages read into cache" : 0,
        "bloom filters in the LSM tree" : 0,
        "chunks in the LSM tree" : 0,
        "highest merge generation in the LSM tree" : 0,
        "queries that could have benefited from a Bloom filter that did not fit in memory" : 0,
        "sleep for LSM checkpoint throttle" : 0,
        "unmerged chunks in the LSM tree" : 0,
        "worker time" : 0
      }
    }
  }
}

```

```

    "ok" : 1
  },
  "shard2ReplSet" : {
    "ns" : "test.Log",
    "size" : 1984826,
    "count" : 33143,
    "avgObjSize" : 59,
    "storageSize" : 647168,
    "capped" : false,
    "wiredTiger" : {
      "metadata" : {
        "formatVersion" : 1
      }
    },
    "creationString" : "access_pattern_hint=none,allocation_size=4KB,app_metadata=none,checksum=on,collator=,columns=,dictionary=0,encryption=(keyid=,name=),exclusive_memory_cache_size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_truncate=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom_filter=true,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_max=15,merge_min=0),memory_page_max=10MB,min_olap=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,type=file,value_format=(key=sstring,value=sstring)",
    "type" : "file",
    "uri" : "statistics:table:collection-13--3225313794850134786",
    "LSM" : {
      "bloom filter false positives" : 0,
      "bloom filter hits" : 0,
      "bloom filter misses" : 0,
      "bloom filter pages evicted from cache" : 0,
      "bloom filter pages read into cache" : 0,
      "bloom filters in the LSM tree" : 0,
      "chunks in the LSM tree" : 0,
      "highest merge generation in the LSM tree" : 0,
      "queries that could have benefited from a Bloom filter that did not fit in memory" : 0,
      "sleep for LSM checkpoint throttle" : 0,
      "unmerged chunks in the LSM tree" : 0,
      "worker time" : 0
    }
  }
}

```

```

},
"avgObjSize" : 59,
"nindexes" : 2,
"nchunks" : 6,
"shards" : {
  "shard1Rep1Set" : {
    "ns" : "test.Log",
    "size" : 2021616,
    "count" : 33755,
    "avgObjSize" : 59,
    "storageSize" : 659456,
    "capped" : false,
    "wiredTiger" : {
      "metadata" : {
        "formatVersion" : 1
      },
      "creationString" : "access_pattern_hint=none,allocation_size=4KB,app_metadata_resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0,encryption=(keyid=,name=),exclude_in_memory_cache_size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_truncation=0,leaf_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom=false,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_max=15,merge_min=0),memory_page_max=10m,os_compression_min=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,type=file,value_factor=1",
      "type" : "file",
      "uri" : "statistics:table:collection-13-7726629793691180133",
      "LSM" : {
        "bloom filter false positives" : 0,
        "bloom filter hits" : 0,
        "bloom filter misses" : 0
      }
    }
  }
}

```

这里基本保持了数据的均衡。