**MongoDB分片集群的引入**

**一 简介**

本篇主要介绍MongoDB分片集群的引入及在云主机自建说明

**二 现状**

**后端服务使用MongoDB组件存储数据**

* 采用副本集模式。
* 由于MongoDB初期业务代码涉及事务，要使用事务必须要连接副本集或分片集群
* MongoDB 中的*副本集*是一组维护相同数据集的MongoDB进程。副本集提供冗余和高可用性
* 默认存储引擎为WiredTiger，会对数据进行压缩存储，当前统计值为压缩后的占用空间。实际使用MongoDB备份工具导出的数据文件大小更大。
* 以下为正式服数据，统计时间为2024年8月12日，统计数值由MongoDB Compass（GUI）给出。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| feed数据库：总共占用磁盘空间：47.84MB |  |  |  |  |
| 集合名 | 说明 | 占用磁盘空间 | 总共索引大小 | 文档数量 |
| like | 点赞 | 9.98MB | 10.25MB | 251K |
| comment | 评论 | 12.74MB | 7.02MB | 171K |
| activity | 动态 | 13.71MB | 2.87MB | 76K |
| favorites | 收藏 | 139.26kB | 106.50kB | 3.3K |
| votes | 投票 | 20.48kB | 20.48kB | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| gamping数据库：总共占用磁盘空间：458.72MB |  |  |  |  |
| 集合名 | 说明 | 占用磁盘空间 | 总共索引大小 | 文档数量 |
| action\_logs | 用户行为日志 | 369.24MB | 696.35MB | 14M |
| score\_log | 用户部分积分日志 | 18.78MB | 8.91MB | 470K |
| text\_identify | 文本审核记录 | 39.24MB | 7.11MB | 374K |
| image\_identify | 图片审核记录 | 12.58kB | 2.49MB | 121K |
| clockin\_history | 签到记录 | 2.23MB | 4.57MB | 60K |
| mp\_opr\_log | 后台道具下发记录 | 3.02MB | 774.14kB | 38k |
| video\_identify | 视频审核记录 | 401.41kB | 86.02kB | 2.2K |
| user\_whitelist | 用户审核白名单 | 20.48kB | 36.86kB | 49 |
| login\_white\_users | 用户登录白名单 | 20.48kB | 36.86kB | 11 |
| url\_whitelist | 跳转链接白名单 | 20.48kB | 36.86kB | 5 |
| content\_activity | 内容活动模板 | 20.48kB | 36.86kB | 3 |
| app\_config | 配置信息 | 20.48kB | 36.86kB | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| game数据库：总共占用磁盘空间：38.96MB |  |  |  |  |
| 集合名 | 说明 | 占用磁盘空间 | 总共索引大小 | 文档数量 |
| reward\_record |  | 10.15MB | 8.12MB | 427k |
| action\_log |  | 3.87MB | 2.40MB | 117k |
| player |  | 13.73MB | 888.83kB | 44k |
| risk\_control |  | 2.80MB | 921.60kB | 40k |
| team\_hall\_msg |  | 24.58kB | 36.86kB | 146 |
| id\_card\_bind |  | 20.48kB | 36.86kB | 44 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Intelligence：总共占用磁盘空间：77.82kB |  |  |  |  |
| 集合名 | 说明 | 占用磁盘空间 | 总共索引大小 | 文档数量 |
| question |  | 77.82kB | 45.06kB | 376 |

* 整体来看，虽然数据整体的磁盘占用空间比较小，但是每个集合的数量比较多。以目前查询比较频繁的动态列表来说，查询包含动态、点赞、评论，一次查询的数据量就很多了。

**三 历史背景/问题**

**2024年2月22日 A5放量**

* 前端获取动态列表超时、页面加载失败，服务被击垮
* MongoDB动态、点赞、评论集合针对查询条件建立索引
* 增加服务限流
* 定时任务拆分

**2024年8月7日 T5放量**

* 后端 动态服务 大量错误日志显示请求超时，服务不可用

**四 解决方案**

**方案一 MongoDB 副本集读写分离**

* 设置MongoDB客户端的读取偏好、设置为secondaryPreferred优先读取从节点数据



* 优点：
* 不改变MongoDB现有集群方案，只需要再增加多个从节点来提升数据读取性能
* 缺点/不支持：
* 数据冗余，集群中所有节点保存相同数据。
* 项目代码如果包含读请求的事务，会与优先读取从节点数据的设置冲突，两者不能共存。解决方法当然可以设置项目代码无关事务的代码单独请求，但是会降低代码可维护性、可读性，破坏代码结构；或者去除事务，但也需要评估功能是否需要事务。
* 对于主节点的写请求压力没有分摊。并且副本集异步同步主节点数据，和主节点写入压力基本一致。
* 从节点的读权限高于写权限，意味着如果从节点读太高会影响数据同步，数据不一致的可能性会增加；同时如果写滞后太多，超过主oplog上限，触发整体同步，会给主节点带来一定的性能压力。

**方案二 MongoDB 分片集群**

* 分片是一种跨多台机器分布数据的方法。MongoDB 使用分片来支持超大数据集和高吞吐量操作的部署。
* 优点：
* 数据被分散存储，每个节点只需要处理部分数据，降低了单节点的负载压力
* 由于数据被分散到多个节点上，因此可以并行处理查询请求，提高了系统的吞吐量。
* 支持横向扩展。将系统数据集和负载划分到多个服务器，以及按需增加服务器以提高容量。虽然单个机器的总体速度或容量可能不高，但每个机器均可处理总体工作负载的一部分，因此可能会比单个高速、高容量服务器提供更高的效率。
* 缺点：
* 自建MongoDB分片集群需要复杂的配置和管理
* 自建MongoDB分片集群需要多台云主机
* 数据冗余，集群中各分片副本集保存相同数据。
* 也正是因为加了一层mongos代理，所以性能会有部分损失

**方案三 项目代码优化**

* 将串行获取动态相关数据（用户信息、服务器信息）优化为并行调用（参考B站微服务早期对接口调用的优化）
* APP首页-对动态首页缓存，目前所有用户进入APP都会请求拉取的第一页的动态数据，并且所有用户拉取的大部分数据都是相同的，这样能抵挡大部分的mongodb请求，编辑动态可能会带来缓存不一致的问题，虽然数据不一致问题也只限制在首页动态中，但影响很大

**五 MongoDB分片集群搭建**

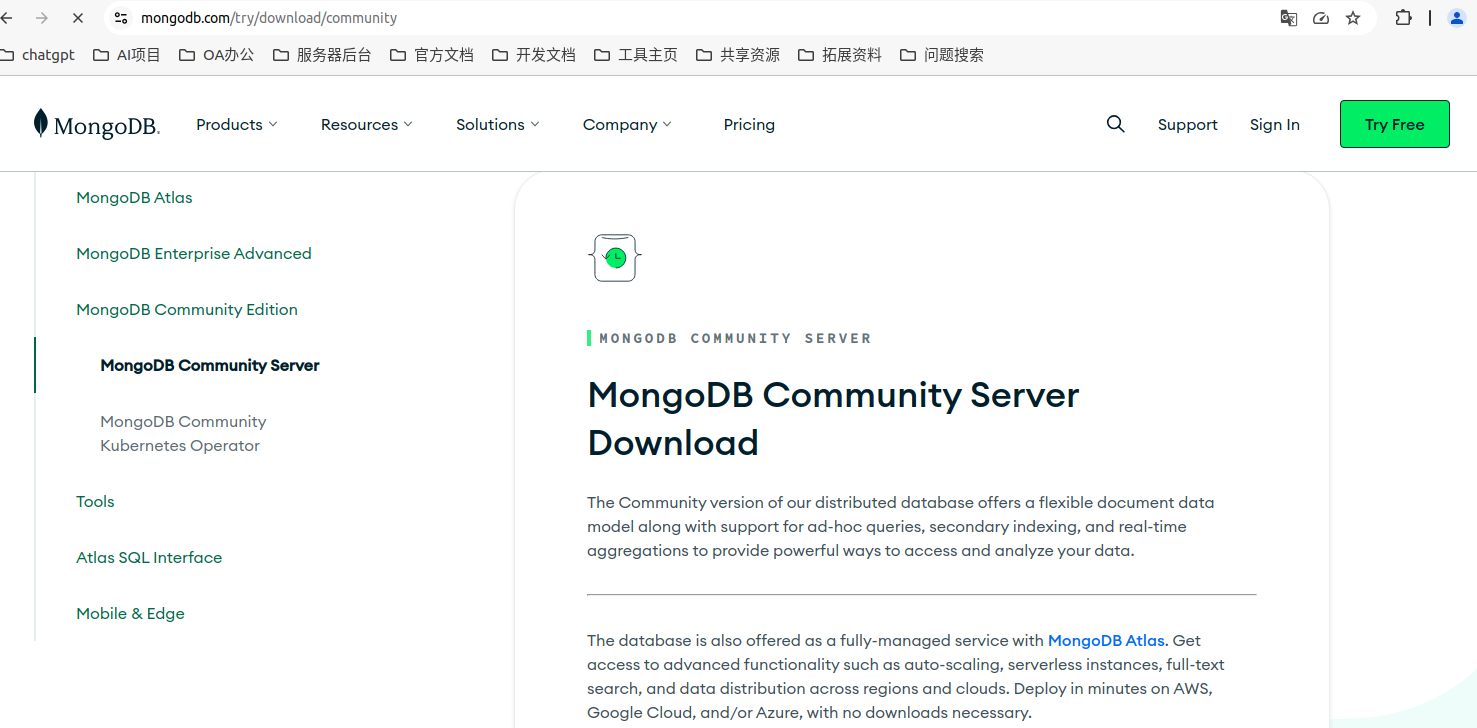
**（一）安装包下载：**

1. 查看腾讯云中被部署的服务器平台类型



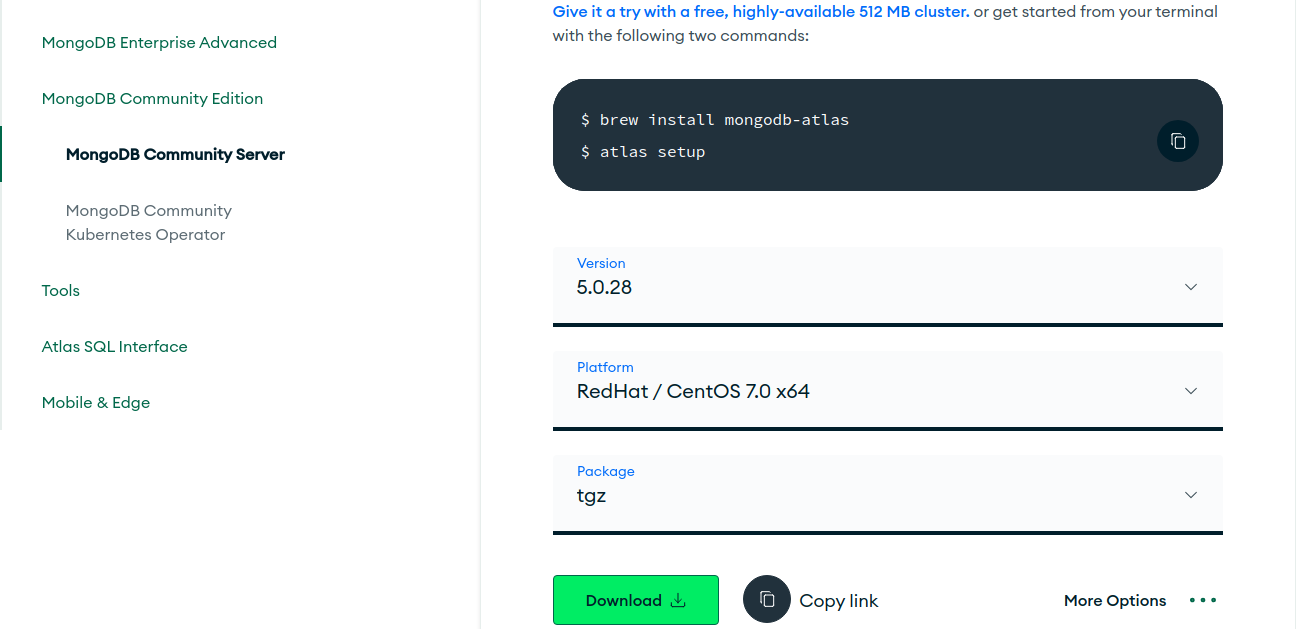
1. 下载MongoDB安装包

* 官网地址：https://www.mongodb.com/try/download/community
* 包含企业版和社区版，选择社区版



1. 选择合适安装包

由于服务之前用的就是5.0版本，所以版本信息不变



1. 下载完成



**（二）安装MongoDB**

1. 解压

* tar -zxvf mongodb-linux-x86\_64-ubuntu2004-5.0.28.tgz

1. 移至指定路径

* mv mongodb-linux-x86\_64-ubuntu2004-5.0.28 /usr/local/mongodb

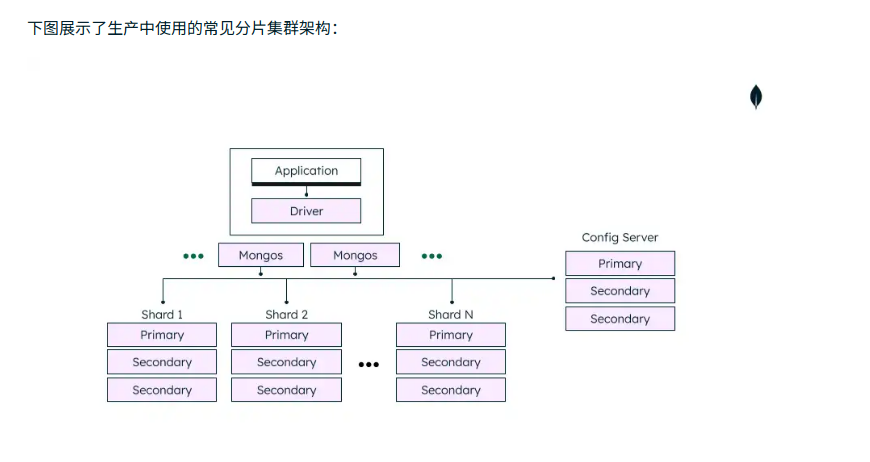
1. 配置环境变量

* 打开环境变量文件
* vim /etc/profile
* 文件末加入以下语句：
* export PATH="$PATH:/usr/local/mongodb/bin"
* 生效
* source /etc/profile

1. 查看安装结果：

|  |
| --- |
| Plain Text robo@robo-System-Product-Name:~/4mongodb$ mongod -version db version v5.0.28 Build Info: {  "version": "5.0.28",  "gitVersion": "a8f8b8e1e271f236e761d0138e2418d0a114c941",  "openSSLVersion": "OpenSSL 1.1.1f 31 Mar 2020",  "modules": [],  "allocator": "tcmalloc",  "environment": {  "distmod": "ubuntu2004",  "distarch": "x86\_64",  "target\_arch": "x86\_64" } |

**（三）架构/组件/配置文件介绍**



1. mongos

* 请求分发中心，数据库集群的请求入口，将对应的数据请求转发到对应的shard服务上
* 为了保证高可用，通常部署多个

1. config server

* 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，只是缓存在内存中，由配置服务器实际来存储这些数据。
* mongos启动会从配置服务器加载配置信息，配置信息发生改变会通知所有mongos同步更新
* 为了保证高可用，通常部署多个

1. shard

* 分片，保存了服务最终需要存储的数据
* 依据分片键将数据分散到不同机器上，每个分片负责总数据的一部分
* 为了保证高可用，通常部署多个

1. replica set

* 副本集，为了保证节点的高可用性，提供了数据的冗余备份

1. 集群负载均衡器

* 负载均衡器进程负责在每个sharded collection的分片之间均匀地重新分配数据块。默认，负载均衡器进程始终处于启用状态。
* 为了解决分片集合数据段分布不均的问题，负载均衡器会将数据段从数据段较多的分片迁移到数据段较少的分片。 负载均衡器会迁移数据段，直到集合的数据段均匀分布在各分片上。
* 数据段表示分片中连续的分[片键值](https://www.mongodb.com/zh-cn/docs/v5.0/reference/glossary/#std-term-shard-key)[范围](https://www.mongodb.com/zh-cn/docs/v5.0/reference/glossary/#std-term-shard)。 数据段范围包括下边界，但不包括上边界。 当数据段大小超过配置的数据段大小（默认为64 MB）时，MongoDB 会分割数据段。 当一个分片包含的集合数据段远多于其他分片时，MongoDB 会迁移数据段。

1. 配置文件

* processManagement.fork
* 启用守护进程模式
* processManagement.pidFilePath
* 指定存储 mongos 或 mongod 进程的进程 ID (PID) 的文件位置
* net.bindIp
* mongos 或 mongod 应监听客户端连接的主机名和/或 IP 地址和/或完整 Unix 域套接字路径,要绑定到多个地址，请输入逗号分隔值的列表
* net.port
* MongoDB 实例监听客户端连接的 TCP 端口
* net.maxIncomingConnections
* mongos 或 mongod 接受的最大并行连接数
* storage.dbPath
* mongod 实例存储数据的目录
* storage.journal
* 启用或禁用持久性日志，确保数据文件保持有效且可恢复。
* systemLog.file
* MongoDB 发送所有日志输出的目标。如果指定 file，则必须同时指定 systemLog.path
* systemLog.path
* mongod 或 mongos 应将所有诊断日志信息发送到的日志文件路径
* systemLog.logAppend
* 如果为 true，则当实例重新启动时，mongos 或 mongod 会在现有日志文件的末尾附加新条目
* sharding.clusterRole
* mongod 实例在分片集群中的角色。
* configsvr : 将此实例作为配置服务器启动
* shardsvr : 将该实例作为分片启动
* replication.replSetName
* mongod 所属副本集的名称
* setParameter.maxNumActiveUserIndexBuilds
* 设置主节点上允许并发索引的最大构建次数。
* 未设置字段采用默认值

1. 更多配置文件信息

* https://www.mongodb.com/zh-cn/docs/v5.0/reference/configuration-options

**（四）部署config server**

1. config server配置文件，如下修改成三份，参数不冲突

|  |
| --- |
| Plain Text cat > config\_server1/conf/config\_server.conf << EOF processManagement:  fork: true  pidFilePath: /home/robo/4mongodb/config\_server1/run/server3.pid #需要修改  net:  bindIp: 0.0.0.0 #看情况绑定内网地址还是需要被外网访问  port: 27020 #需要修改  maxIncomingConnections: 20000  storage:  dbPath: /home/robo/4mongodb/config\_server1/data #需要修改  journal:  enabled: true  systemLog:  destination: file  path: /home/robo/4mongodb/config\_server1/log/server.log #需要修改  logAppend: true  sharding:  clusterRole: configsvr  replication:  replSetName: config  # security:  # keyFile: /home/robo/4mongodb/keyfile/mongo.keyfile EOF |

1. 启动三个节点

* mongod -f config\_server1/conf/server.conf
* mongod -f config\_server2/conf/server.conf
* mongod -f config\_server3/conf/server.conf

1. 节点初始化

* mongo 192.168.30.172 --port 27020

|  |
| --- |
| Plain Text > rs.initiate(  {  \_id: "config",  configsvr: true,  members: [  { \_id : 0, host : "192.168.30.172:27020" }, #修改为节点1启动的地址和端口  { \_id : 1, host : "192.168.30.172:27021" }, #修改为节点2启动的地址和端口  { \_id : 2, host : "192.168.30.172:27022" } #修改为节点3启动的地址和端口  ]  } ) |

* 初始化后状态

|  |
| --- |
| Plain Text config:SECONDARY> rs.status() |

**（五）部署shard**

1. shard配置文件，如下修改成三份，参数不冲突

|  |
| --- |
| Plain Text cat > shard1/conf/shard.conf << EOF processManagement:  fork: true  pidFilePath: /home/robo/4mongodb/shard1/run/shard.pid #需要修改  net:  bindIp: 0.0.0.0 #看情况绑定内网地址还是需要被外网访问  port: 27030 # 需要修改  maxIncomingConnections: 20000  storage:  dbPath: /home/robo/4mongodb/shard1/data #需要修改  journal:  enabled: true  systemLog:  destination: file  path: /home/robo/4mongodb/shard1/log/shard.log #需要修改  logAppend: true  # security:  # keyFile: /home/robo/4mongodb/keyfile/mongo.keyfile  sharding:  clusterRole: shardsvr  replication:  replSetName: shard1  setParameter:  maxNumActiveUserIndexBuilds: 6 EOF |

1. 启动三个节点

* mongod -f shard1/conf/shard.conf
* mongod -f shard2/conf/shard.conf
* mongod -f shard3/conf/shard.conf

1. 进入任一节点，初始化配置

* mongo 192.168.30.172 --port 27030

|  |
| --- |
| Plain Text rs.initiate(  {  \_id: "shard1", #与配置文件中的replSetName对应  members: [  { \_id : 0, host : "192.168.30.172:27030" }, #修改为节点1启动的地址和端口  { \_id : 1, host : "192.168.30.172:27031" }, #修改为节点2启动的地址和端口  { \_id : 2, host : "192.168.30.172:27032" } #修改为节点3启动的地址和端口  ]  } ) |

* 配置后节点状态

|  |
| --- |
| Plain Text shard1:SECONDARY> rs.status() |

**（六）部署mongos**

1. mongos配置文件，修改成三份，参数不冲突

|  |
| --- |
| Plain Text cat > mongos1/conf/mongos.conf << EOF processManagement:  fork: true  pidFilePath: /home/robo/4mongodb/mongos1/run/mongos.pid #需要修改  net:  bindIp: 0.0.0.0 #看情况绑定内网地址还是需要被外网访问  port: 27010 #需要修改  maxIncomingConnections: 20000  systemLog:  destination: file  path: /home/robo/4mongodb/mongos1/log/mongos.log #需要修改  logAppend: true  # security:  # keyFile: /home/robo/4mongodb/keyfile/mongo.keyfile  sharding:  configDB: config/192.168.30.172:27020,192.168.30.172:27021,92.168.30.172:27022 #修改为config server的三个节点地址端口 EOF |

1. 启动三个节点

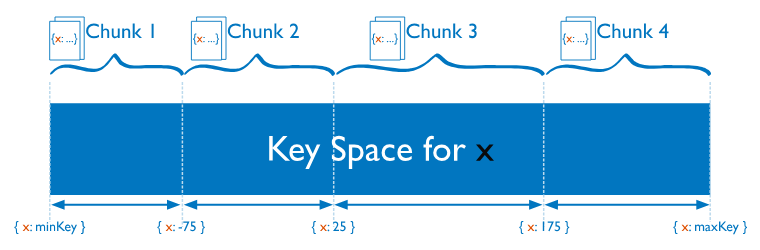
* mongos -f mongos1/conf/mongos.conf
* mongos -f mongos2/conf/mongos.conf
* mongos -f mongos3/conf/mongos.conf

1. 登录任一mongos节点进行分片设置

* mongo 192.168.30.172 --port 27010
* sh.addShard( "shard1/192.168.30.172:27030,192.168.30.172:27031,192.168.30.172:27032")

**（七）设置数据库及集合分片生效**

* **分片键**
* 一个好的片键对分片至关重要。分片键可以是单个索引字段，也可以是复合索引涵盖的多个字段
* MongoDB 将分片键值（或哈希分片键值）的取值区间划分为不重叠的分片键值（或哈希分片键值）范围。每个范围都与一个[数据块](https://www.mongodb.com/zh-cn/docs/v5.0/reference/glossary/#std-term-chunk)相关联，MongoDB 会尝试在集群中的各个分片之间均匀分配这些数据块。



* **分片键的选择考虑以下几种条件**
* 分片键的关联基数
* 分片键的关联基数决定了负载均衡器可以创建的最大数据段数。尽可能选择关联基数高的分片键。关联基数低的分片键会降低集群中水平扩展的有效性。比如性别只有男女，那么最多只会创建2个数据段来保存男女。因为一个分片值只会在一个数据段内
* 分片键值出现的频率
* 分片键的 frequency 表示给定分片键值在数据中出现的频率。如果多数文档仅包含可能的分片键值的一部分，则存储具有这些值的文档的数据段可能会成为集群内的瓶颈。
* 潜在的分片键是否单调
* 对于范围分片来说，针对单调增加或减少的值的分片键更有可能将插入操作分布到集群内的单个数据段中。
* 但是单调增加或减少的值的分片键可以使用哈希分片
* 分片查询模式
* 也就是要应对我们代码的常用查询
* 分布式查找
* 在分片集群中，如果查询包含分片键，则 [mongos](https://www.mongodb.com/zh-cn/docs/v5.0/reference/program/mongos/#mongodb-binary-bin.mongos) 仅会将这些查询路由到包含相关数据的分片，在传送到特定分片时，对分片集群的读取操作效率最高。如果查询不包含分片键，则这些查询会被广播到所有分片以进行评估。这些类型的查询称为“分散-聚合”查询。每个请求涉及多个分片的查询效率较低（当然需要考虑集群的规模）。在较大的集群上，分散聚集查询对于常规操作是不可行的。但是针对大量数据进行操作的聚合查询（如查询文档数量）。“分散-聚合”查询可能是一种有用的方法，因为它允许查询在所有分片上并行运行。
* 分片键限制
* 分片键的降序索引
* 只包含部分索引
* 多键（数组值的字段）索引
* 文本（全文搜索）索引
* 通配符索引
* 分片键通常使用一个大粒度的字段加上经常用来查找的字段
* **分片策略**
* 哈希分片

哈希分片在分片集群中提供更均匀的数据分布，但代价是相较于广播，减少了定向操作。经过哈希运算后，具有 "close" 分片键值的文档不太可能位于同一数据段或分片上 - [mongos](https://www.mongodb.com/zh-cn/docs/v5.0/reference/program/mongos/#mongodb-binary-bin.mongos) 更有可能执行广播来完成给定的范围查询。

* 范围分片

基于范围的分片涉及将数据划分为由分片键值确定的连续范围。在此模型中，具有 "close" 分片键值的文档可能位于同一数据块或分片中。如此，便可在读取连续范围内的目标文档时实现高效查询。但是，由于分片键的选择不当，读写性能可能会下降。

* 这两种分片策略，范围分片更容易导致热点数据，对比通常使用哈希分片
* **分片设置示例**
* mongo 192.168.30.172 --port 27010
* use admin
* sh.enableSharding("gamping") #为gamping数据库启用分片
* sh.shardCollection("gamping.activity", {"\_id": "hashed" }) #为activity集合设置依据ID进行hash的分片规则
* 数据写入后查看集合的数据分布
* use gamping
* db.activity.stats()
* **注意**
* **在新数据库或新集合创建前，需要提前设置启用分片**

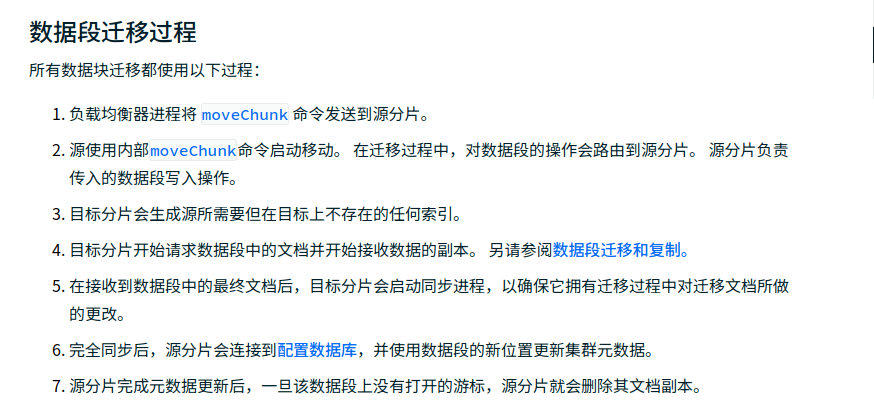
**（八）shard扩容**

1. 依据部署shard的流程在另外的机器中部署shard2
2. 登录任一mongos节点进行分片设置

* mongo 192.168.30.172 --port 27010
* sh.addShard( "shard1/192.168.30.172:27040,192.168.30.172:27041,192.168.30.172:27042")

1. mongos内部借助负载均衡器实现数据迁移

* 为了解决分片集合数据段分布不均的问题，负载均衡器会将数据段从数据段较多的分片迁移到数据段较少的分片。 负载均衡器会迁移数据段，直到集合的数据段均匀分布在各分片上。



1. 查看集合的数据分布来验证

* use gamping
* db.activity.stats()

**（九）shard缩容**

* 删除分片shard3
* mongo 192.168.30.172 --port 27010
* 删除命令
* db.runCommand({removeShard:"shard3"})
* 状态
* "state" : "started"

|  |
| --- |
| Plain Text mongos> db.runCommand({removeShard:"shard3"}) {  "msg" : "draining started successfully",  "state" : "started",  "shard" : "shard3",  "note" : "you need to drop or movePrimary these databases",  "dbsToMove" : [  "test"  ],  "ok" : 1,  "$clusterTime" : {  "clusterTime" : Timestamp(1723348601, 3),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  },  "operationTime" : Timestamp(1723348601, 3) } |

1. mongos内部借助负载均衡器实现数据迁移

* 如果删除的shard包含基片，如下
* 状态
* "state" : "ongoing"

|  |
| --- |
| Plain Text mongos> db.runCommand({removeShard:"shard3"}) {  "msg" : "draining ongoing",  "state" : "ongoing",  "remaining" : {  "chunks" : NumberLong(0),  "dbs" : NumberLong(1),  "jumboChunks" : NumberLong(0)  },  "note" : "you need to drop or movePrimary these databases",  "dbsToMove" : [  "test"  ],  "ok" : 1,  "$clusterTime" : {  "clusterTime" : Timestamp(1723358064, 1),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  },  "operationTime" : Timestamp(1723358064, 1) } |

* 则需要手动迁移至指定分片：
* use admin
* db.runCommand({"movePrimary": "Activity", "to": "shard1"})
* 等待数据迁移结束，可以多次执行以下命令来查看
* db.runCommand({removeShard:"shard3"})
* 状态
* "state" : "completed"
* 当状态为completed的时候，就可以退出shard3分片的所有进程了

|  |
| --- |
| Plain Text db.runCommand({removeShard:"shard3"}) mongos> db.runCommand({removeShard:"shard3"}) {  "msg" : "removeshard completed successfully",  "state" : "completed",  "shard" : "shard3",  "ok" : 1,  "$clusterTime" : {  "clusterTime" : Timestamp(1723358159, 5),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }, "operationTime" : Timestamp(1723358159, 5) } |

**（十）设置安全验证**

1. 随机密码

* 创建管理员账号
* db.createUser({user:"admin",pwd:"74dZB3G3gUHJw3NRqwT9",roles:["root"]})
* 创建普通用户账号
* db.createUser({user:"random",pwd:"aWWKyAdDoU2Yi5doL66c",roles:["readWrite"]})

1. 创建密钥文件

* openssl rand -base64 756 > ./mongo.keyfile
* chmod 600 mongo.keyfile

1. 重启mongo分片集群

* **参考（十二）扩展-重启分片集群 说明**

1. 认证

* mongo 192.168.30.172 --port 27010
* use admin
* db.auth("admin","74dZB3G3gUHJw3NRqwT9")

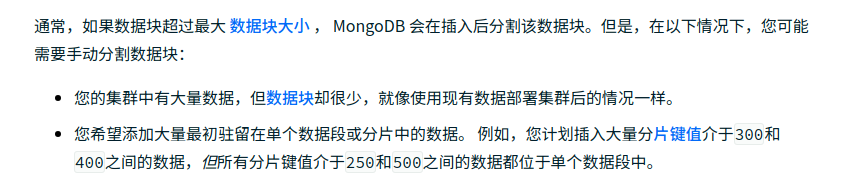
**（十一）副本集群数据迁移至分片集群中**

迁移方案 ：

* 使用mongodump工具将线上数据库导出来
* 使用mongorestore工具将导出来的数据恢复至分片集群中
* **注意：**
* **必须提前为数据库和数据库中的集合启用分片**
* sh.enableSharding("feed")
* sh.shardCollection("feed.activity", {"\_id": "hashed" })
* 现象：
* 数据平均分配在每个节点上

**（十二）扩展**

* **手动分隔数据块：**



也就是 涉及未分片的集合写入和已经写入过的数据 需要手动分片，原则上 不应该人为去改动这些数据块，所以我们必须对集合和数据库提前启用分片

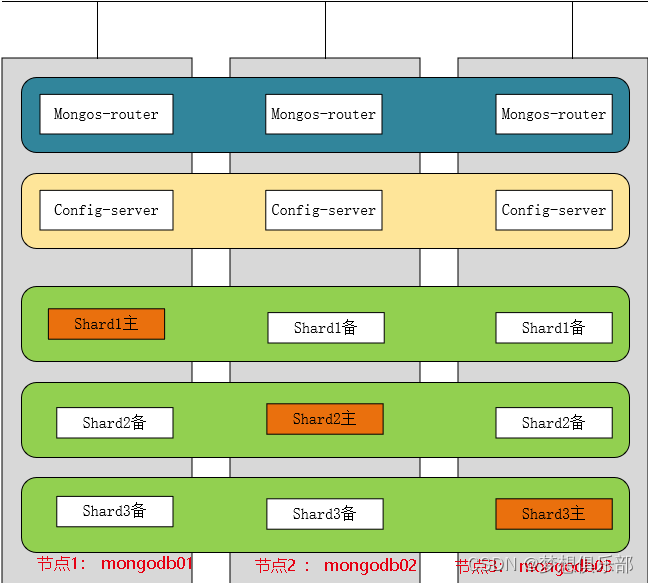
* **重启分片集群：**
* 以不同的顺序停止或启动分片集群的组件，可能导致成员之间出现通信错误。例如，如果没有可用的配置服务器，分片服务器可能会挂起。
* 停止分片集群
* 禁用负载均衡器
* sh.stopBalancer()



* 按顺序启动分片集群
* 启动配置服务器
* 启动分片副本集
* 启动mongos路由器
* 启动负载均衡器
* sh.startBalancer()

**（十三）项目实际部署**

三台机器



机器一（新mogno机器 [134.175.53.214](http://134.175.53.214/) ）：

部署 shard1、config\_server1、mongos1

机器二（新服务机器 106.52.193.187）：

部署 mongos2、config\_server2

机器三 （原服务机器 42.194.237.166）：

部署 mongos3、config\_server3