开源数据库 MongoDB 专场
MongoDB疑难杂症分析及优化

阿里云数据库技术
张友东（林青）2016.10.15

阿里云

云数据库 MongoDB 版

主要内容

- Driver 使用问题
  - 连接池大小如何配置?
  - 如何干掉长（慢）请求?
- 复制集问题
  - 如何连接复制集?
  - 备同步为什么跟不上?
  - 备节点阻塞很长时间?
- Shared Cluster 问题
  - 什么时候该分片?
  - 为什么分片负载不均衡?

案例均源自 MongoDB 云数据库

扫码了解更多…

慎用 local、admin

- local：存储节点自身配置信息，数据不会被同步，重要的数据不要存储在 local 数据库，避免数据丢失
- admin：存储用户、角色等管理信息，写入时会加 DB 级别互斥写锁，业务数据不要存储在 admin 数据库，影响性能

database	lock	insert / s
admin	DB 锁	13500
other	文档锁	42600

wiredtiger 引擎
sysbench 16线程 insert

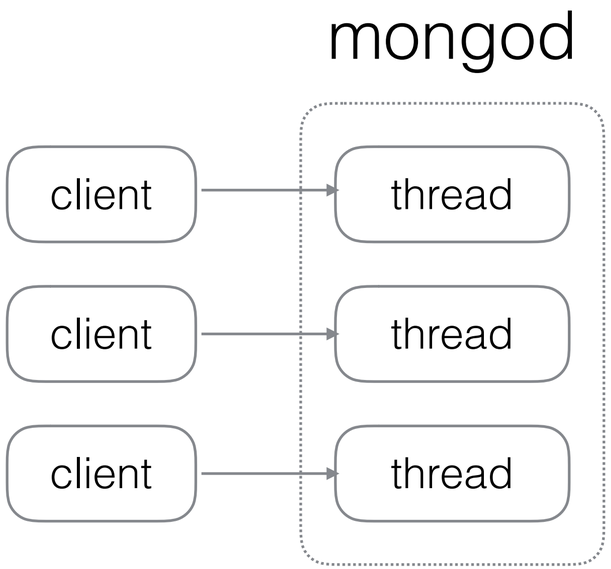
https://yq.aliyun.com/articles/7424

阿里云

云数据库 MongoDB 版

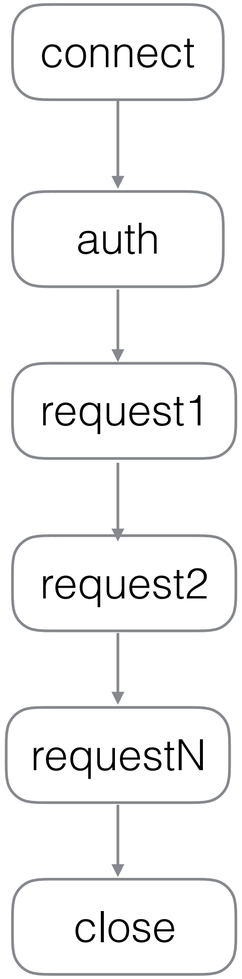
合理配置连接数

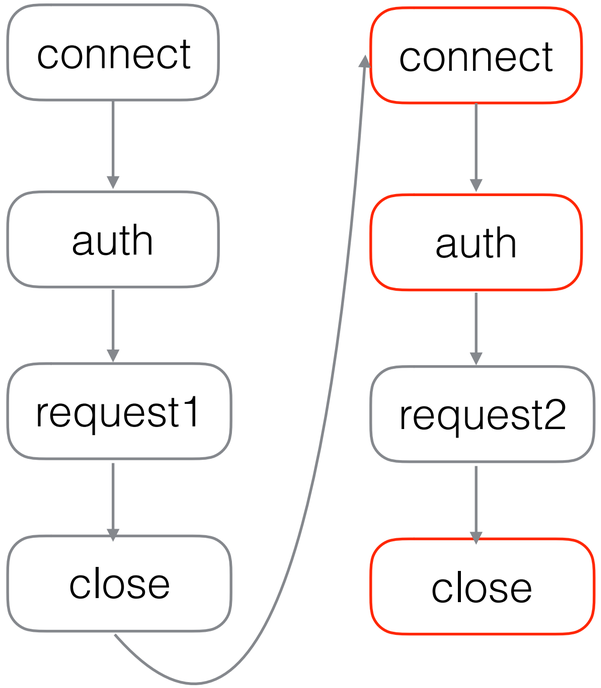
- thread per connection 网络服务模型
  - 每个线程需要1MB 的栈空间
  - 大量连接时，线程切换开销大
- 限制连接数资源
  - mongod 配置 net.maxIncomingConnections 参数
  - Driver通过 Connection String URI 的 maxPoolSize 参数来配置连接池大小



https://yq.aliyun.com/articles/59283

避免使用短连接





- 短连接增加额外的 connect、auth、close、线程创建及销毁开销
- MongoDB 鉴权性能问题，优化后性能提示10+倍

https://yq.aliyun.com/articles/35148

干掉长(慢)请求

- 客户端发起耗时请求，如遍历集合、建索引、mapreduce、aggregation 等，主动断开连接后，后端的请求仍然在执行
- currentOp + killOp 中止后端正在运行的请求

killOp原理：长时间执行的请求会设置检查点，发现有 kill 操作就会主动退出

killOp 后，请求执行到checkForInterrupt()就会退出

```
while (!createIndexFinished) {

    createIndexForOneElement();
    checkForInterrupt();
}
```

https://yq.aliyun.com/articles/6647

控制集合数量

- wiredtiger引擎特性
  - 每个集合对应一个物理文件，每个索引对应一个物理文件
  - listDatabases 时，需要遍历所有的集合及索引，逐个获取物理文件大小信息
- 问题及优化
  - 物理文件太多，数据库管理开销增加，影响性能，建议启用 storage.directoryPerDB选项，尽量让物理文件分散到多个目录。
  - listDatabases 开销太大，导致监控系统无法正常工作，还可能影响到主备同步（全量同步时，会先 listDatabases 拉取 DB 列表，设置的超时时间为30s）
  - 如果一定需要大量的集合，可考虑使用 mmapv1或 rocksdb 引擎

https://yq.aliyun.com/articles/58531

阿里云

云数据库 MongoDB 版

位置查询优化

- $near位置查询，cursor 会缓存结果数据，可能占用大量内存，默认10分钟后 cursor 超时释放
- 优化方法
  - 设置更小的 cursor timeout
  - 如无需遍历，find 时设置 singleBatch 选项

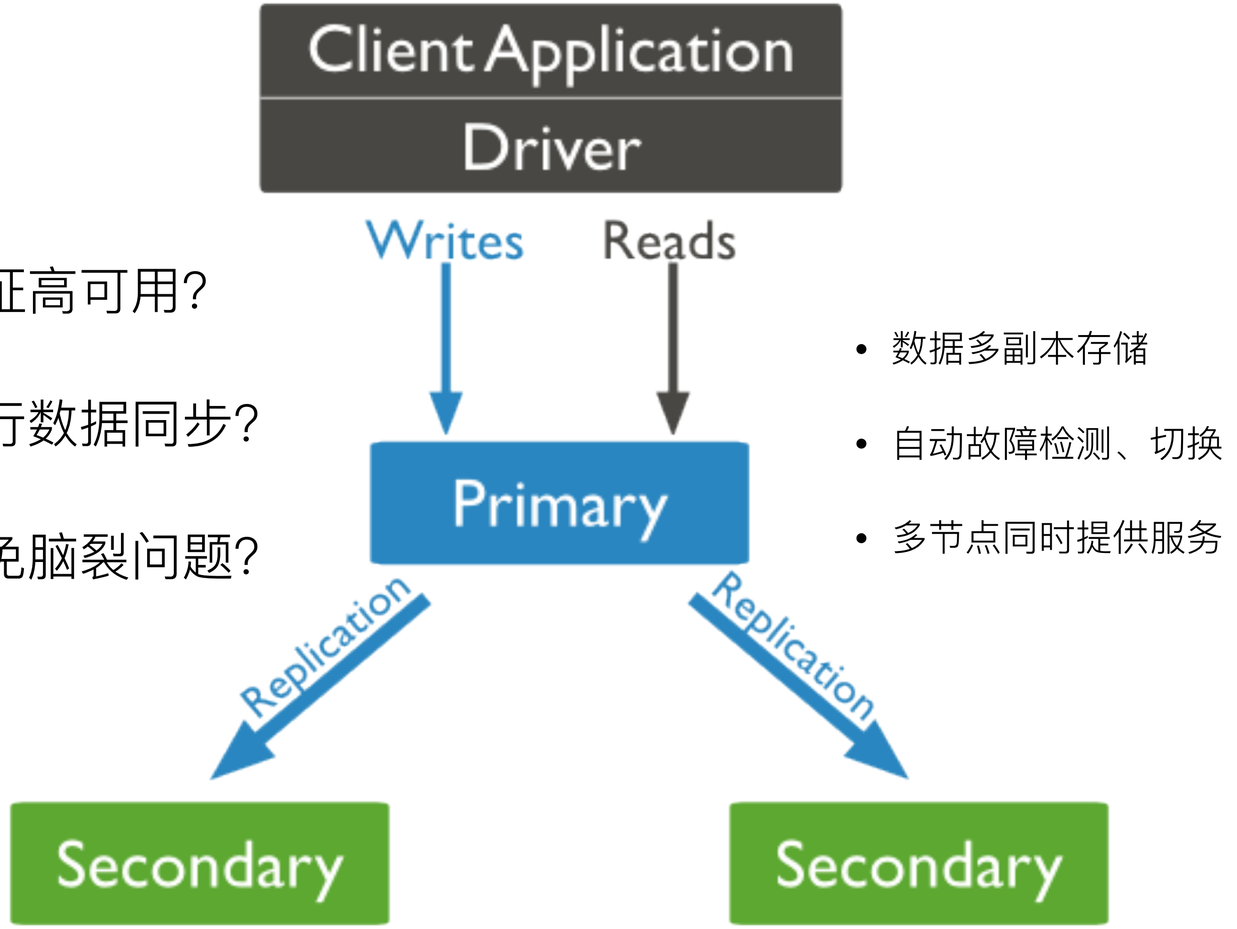
$near位置查询示例

```
{
    "find" : "userData",
    "filter" : {
        "latlng" : {
            "$near" : [
                116.34642045073839,
                39.87082232130999
            ],
            "$maxDistance" : 0.90        }
    },
    "ntoreturn" : 1000
}
```

https://yq.aliyun.com/articles/58532

复制集问题

- 如何保证高可用?
- 如何进行数据同步?
- 如何避免脑裂问题?



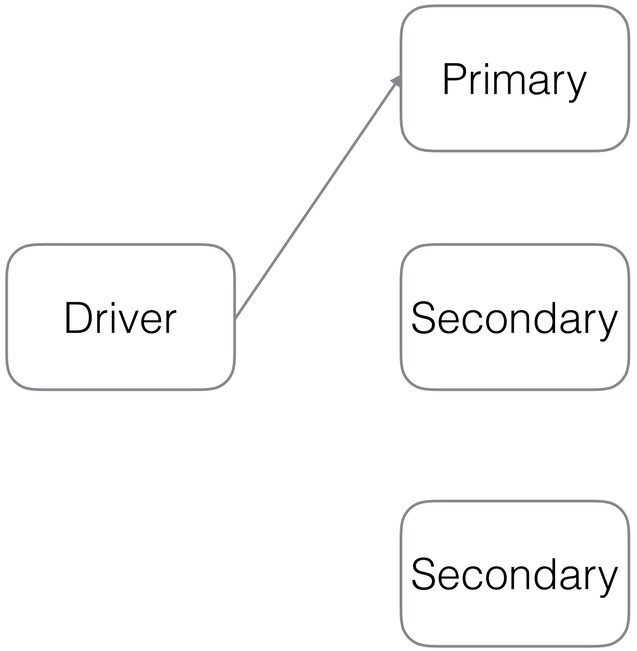
- 数据多副本存储
- 自动故障检测、切换
- 多节点同时提供服务

阿里云

云数据库 MongoDB 版

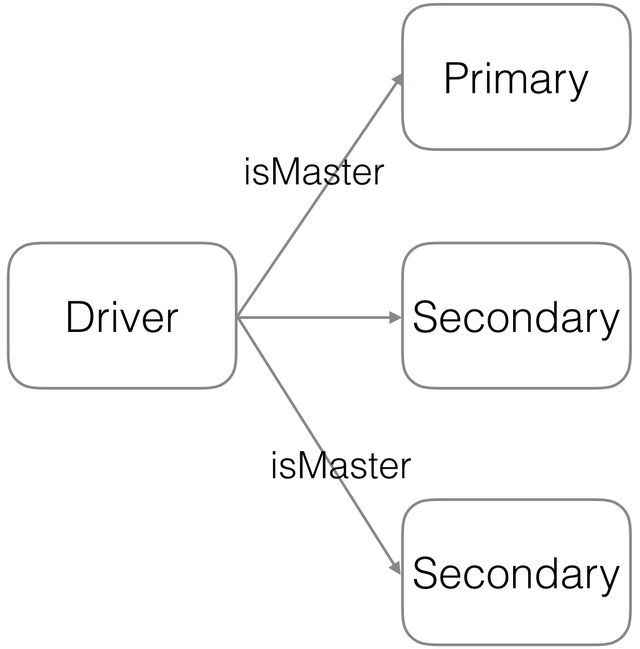
正确连接复制集

直连 Primary



Primary 故障时，不可读写

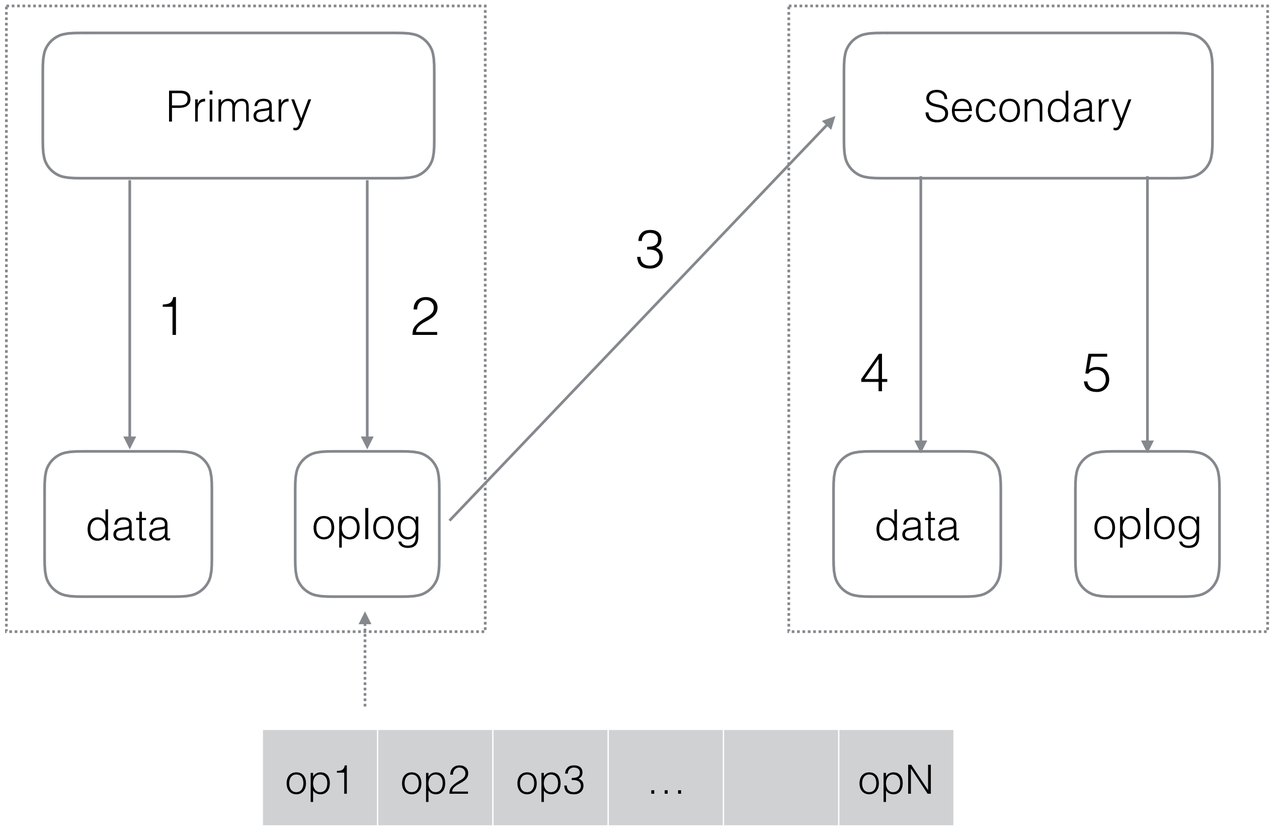
Driver指定多个节点连接复制集，强烈建议



跟所有节点保持心跳，Primary 故障时，
Client 联系新的 Primary 读写

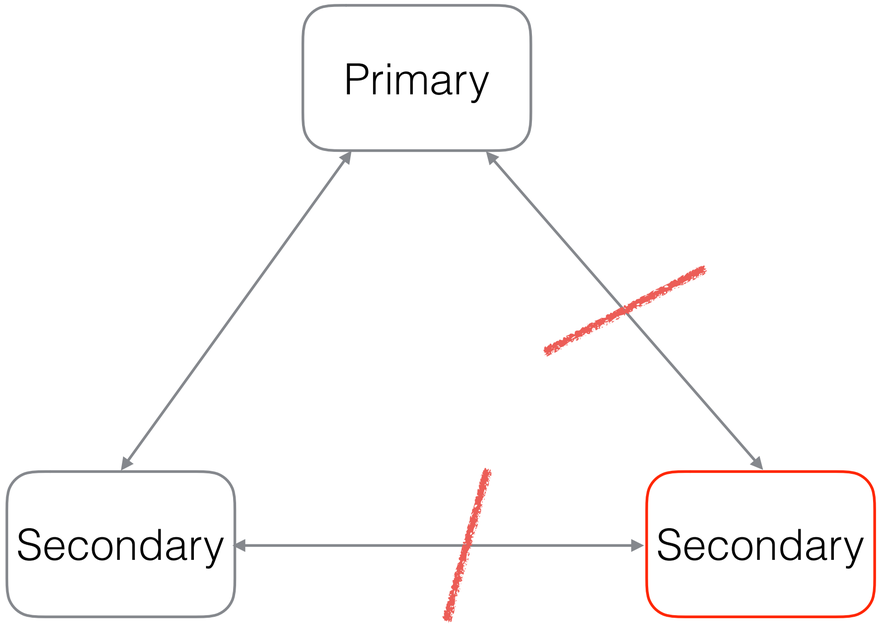
https://yq.aliyun.com/articles/8461

关于 oplog



oplog是固定大小集合，按时间戳排序，满了时自动删除最老的数据

oplog 问题 (1)

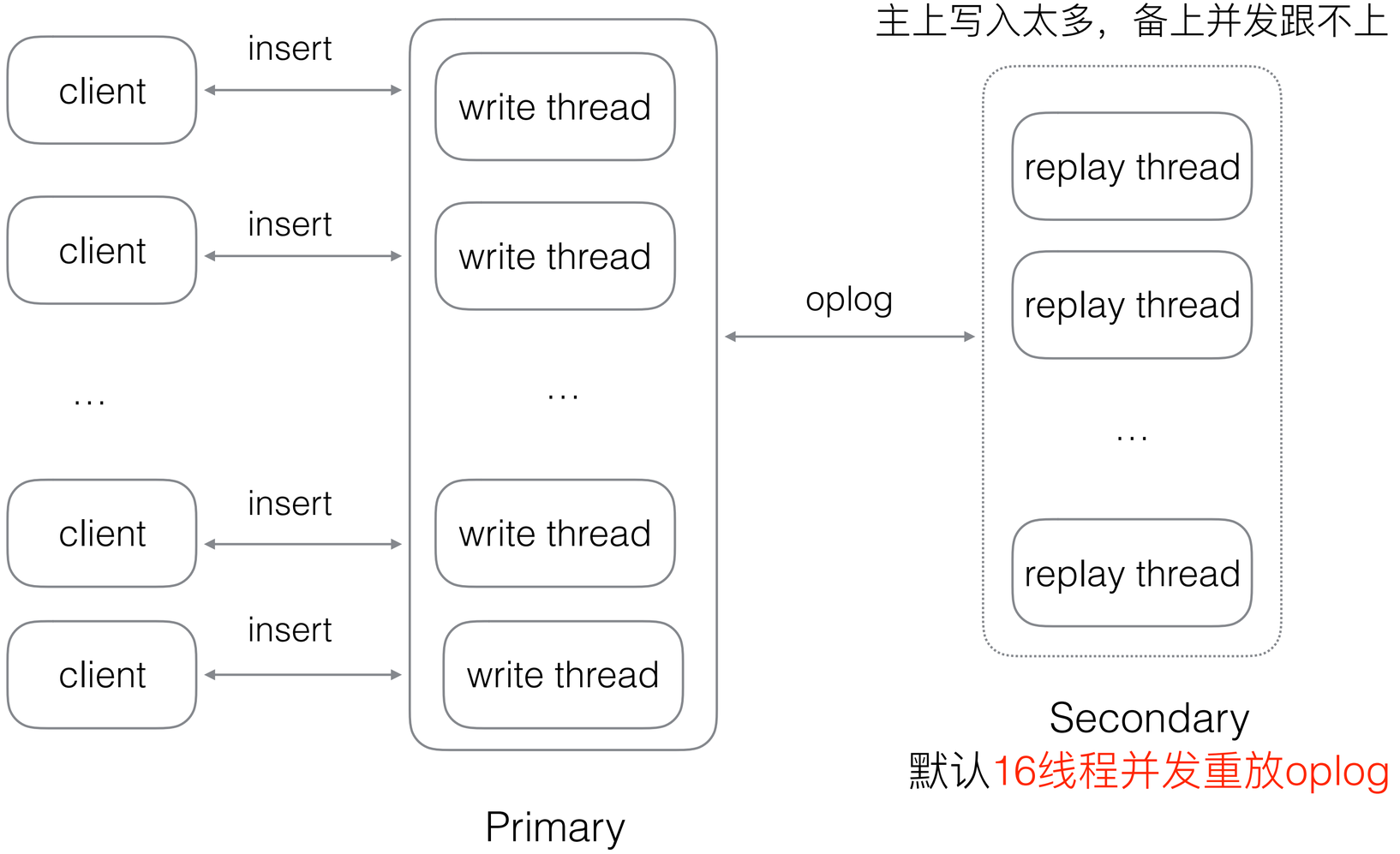


- Secondary 网络断开，或者停机维护，恢复时可能因 oplog 不足无法继续同步
- 停机维护尽量选择在业务低峰期来做

阿里云
云数据库 MongoDB 版

oplog 问题 (2)

主上写入太多，备上并发跟不上



默认16线程并发重放oplog

oplog 问题 (3)

```
mongo:PRIMARY> db.coll.find()
{ "_id" : 1, "x" : [ 1, 2, 3, 4, 5 ] }
mongo:PRIMARY> db.coll.update({_id: 1}, {$push: {x: { $each: [6, 7], $position: 0 }}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
mongo:PRIMARY> db.coll.find()
{ "_id" : 1, "x" : [ 6, 7, 1, 2, 3, 4, 5 ] }
mongo:PRIMARY> use local
switched to db local
mongo:PRIMARY> db.oplog.rs.find().sort({$natural: -1}).limit(1)
{ "ts" : Timestamp(1464082056, 1), "h" : NumberLong("6563273714951530720"),
  "v" : 2, "op" : "u", "ns" : "test.coll", "o2" : { "_id" : 1 },
  "o" : { "$set" : { "x" : [ 6, 7, 1, 2, 3, 4, 5 ] } }
}
```

- 为保证oplog 重放幂等性，往数组开始位置$push操作被转换为$set操作，并且带上整个数组的内容，数组较大时，同步网络流量被严重放大，影响主备同步
- 使用数组时，数组元素尽量不要太多，尽量使用$set 或 在末尾$push

https://yq.aliyun.com/articles/52404

oplog 问题（4）

```
mongo:PRIMARY> db.coll.find()
{ "_id" : ObjectId("57fcd82a66e32bd8f589c3f4"), "x" : 1 }
{ "_id" : ObjectId("57fcd82b66e32bd8f589c3f5"), "x" : 1 }
mongo:PRIMARY> db.coll.update({x: 1}, {$set: {x: 2}}, {multi: true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
mongo:PRIMARY> use local
switched to db local
mongo:PRIMARY> db.oplog.rs.find().sort({$natural: -1}).limit(2)
{ "ts" : Timestamp(1476188286, 2), "t" : NumberLong(23),
"h" : NumberLong("-86899975728035047l9"), "v" : 2, "op" : "u", "ns" : "test.coll",
"o2" : { "_id" : ObjectId("57fcd82b66e32bd8f589c3f5") }, "o" : { "$set" : { "x" : 2 } }
{ "ts" : Timestamp(1476188286, 1), "t" : NumberLong(23),
"h" : NumberLong("-3016675555126288909"), "v" : 2, "op" : "u", "ns" : "test.coll",
"o2" : { "_id" : ObjectId("57fcd82a66e32bd8f589c3f4") }, "o" : { "$set" : { "x" : 2 } }
```

- Primary 上一次性 update 或 remove 多个文档时，每个文档会对应一条 oplog，Secondary 上重放每条 oplog 来保持跟 Primary 数据一致，但开销却比 Primary 大，这样的操作很多时，可能导致备同步无法跟上。

oplog 问题（5）

- mongodump —oplog 因数据集太大，备份时间过长，结束时最初的 oplog 已经被删除，导致全量备份失败

- 使用 tailable cursor不断抓取 oplog（增量备份、或者跨机房同步），如果抓取速度跟不上写入速度，可能导致需要的 oplog 已经被删除，抓取的增量数据不全

阿里云

云数据库 MongoDB 版

如何管理 oplog?

- 支持在线动态修改 oplog 大小
  - db.runCommand({collMod: “oplog.rs”, maxSize: 1024000000})
- 支持复制集成员根据同步进度自适应管理 oplog
- 支持设置一个 oplog 保护时间戳，所有超过该时间戳的 oplog都会自动保留，支持全量、增量备份
  - db.runCommand({collMod: “oplog.rs”, oplogDeleteGuard: 1400000000})

https://yq.aliyun.com/articles/54392

阿里云

云数据库 MongoDB 版

避免无 id 索引的集合

- MongoDB 默认对所有的集合建立 _id 字段索引
- createCollection 时，可通过将 autoIndexId 选项设置为false，不对集合建立 _id 索引
- Primary 上的 insert 操作，在 Secondary 上重放时，先要根据_id查找文档是否已经存在，存在则执行 update、不存在则执行 insert，如果没有_id索引，备上需要进行全表扫描，同步开销被放大数倍。
- autoIndexId 选项会在新版本中废弃，不再支持，阿里云 MongoDB 上也已经禁用了该选项。

https://yq.aliyun.com/articles/7906

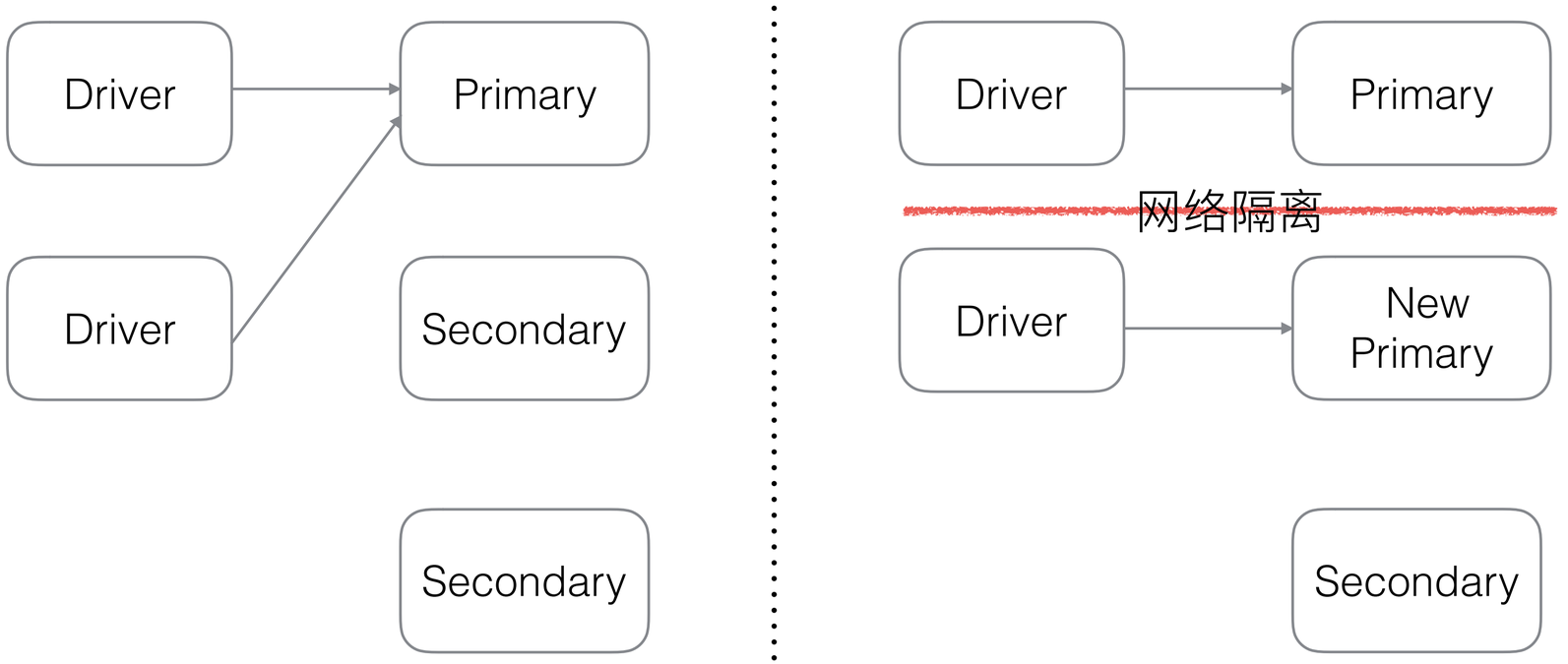
阿里云

云数据库 MongoDB 版

尽量后台建索引

- MongoDB 默认前台建索引，效率更高、索引体积更小
- Primary
  - 前台：DB 写锁，阻塞 DB 上的所有读写
  - 后台：DB 写意向锁，对读写无影响
- Secondary
  - 前台：整个建索引过程中，阻塞所有请求，包括鉴权
  - 后台：读写请求不阻塞
- 建议
  - 尽量在创建集合时，规划好索引，在集合为空的时候就创建索引
  - 针对已有大量数据的集合，尽量后台建索引

脑裂问题



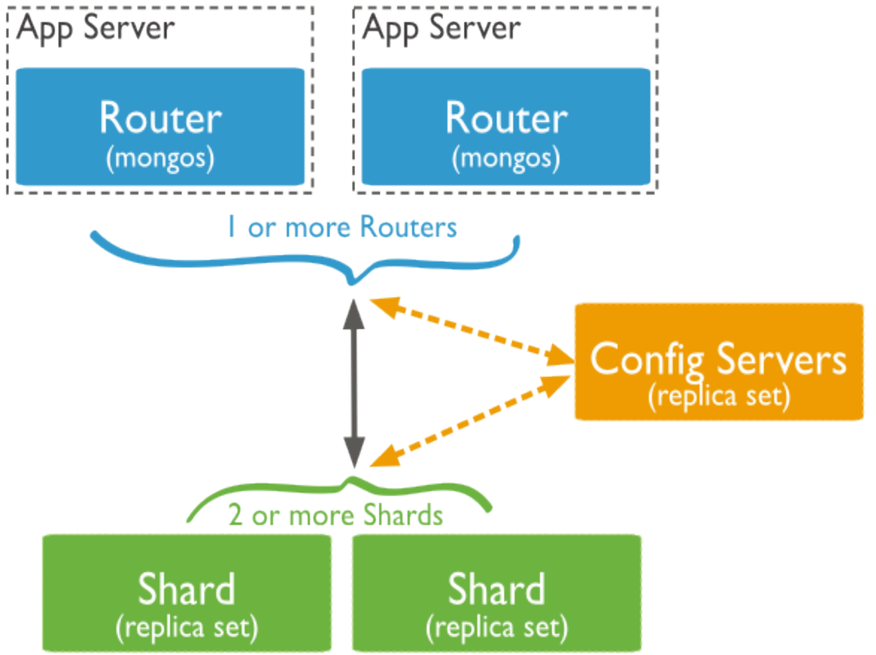
- 发生脑裂后，2个 Primary 同时写入，最终数据不确定
- MongoDB 3.2版本引入raft 协议，New Primary 会开启新的 term，最终数据以 New Primary为准
- 建议用户尽快升级到3.2版本，减少不必要的问题

https://yq.aliyun.com/articles/59504

阿里云
云数据库 MongoDB 版

Sharded Cluster 问题

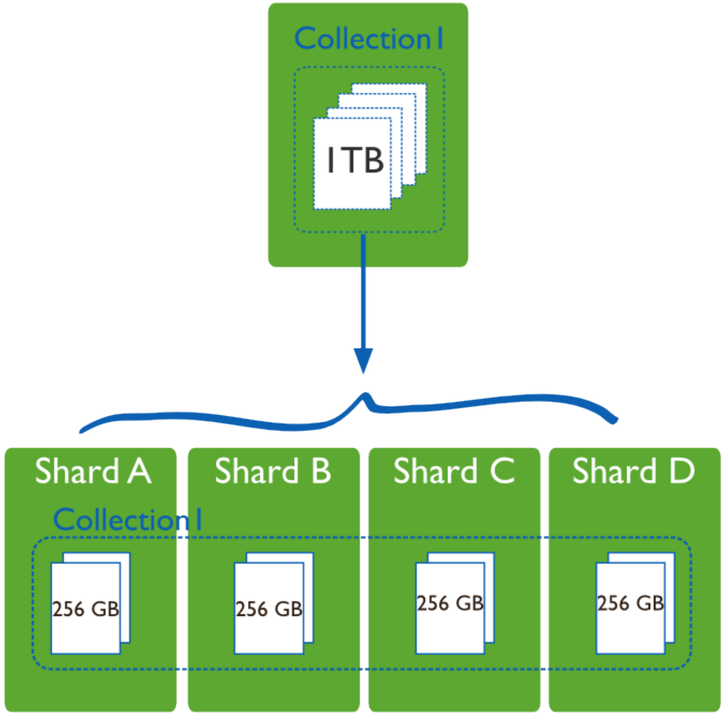
- 什么时候用Sharded cluster?
- 应该部署多少个 shard?
- 数据/请求在多个 shard 间分布不均衡?
- chunk 太大或文档太多无法迁移?



https://yq.aliyun.com/articles/60096

什么时候该分片

- 什么时候使用?
  - 尽量用复制集，运维更简单
  - 扩展存储容量
  - 扩展写（读）服务能力
- 部署多大规模?
  - 先定一个小目标，逐步扩展



shard key 选择

- 分片方式
  - 范围分片，能很好的支持范围查询
  - hash 分片，读写更好的均分到各个 shard
- shard key选择应结合实际业务需求，需要避免的问题
  - shard key 取值范围太小(low cardinality)
  - shard key 某个值的文档特别多，这样导致单个 chunk 特别大（ jumbo chunk），会影响chunk 迁移及负载均衡。
  - 根据非 shard key 进行查询、更新操作都会变成 scatter-gather 查询，影响效率。

https://yq.aliyun.com/articles/60096

分片配置问题

- sharding.chunkSize 默认64M，当jumbo chunk 太多，可考虑将 chunkSize 改大 (治标不治本，本质还是 shard key 选择不合理导致)

- sharding.archiveMovedChunks 决定是否对 chunk 迁移的数据进行归档，3.0默认为 true，3.2默认为 false

- sharding.recoverShardingState 默认为 true，shard 启动后会连接 config server 进行初始化工作，成功初始化后才能正常提供服务

总结

- MongoDB 3.2 版本整体稳定性很高
- 官方对 jira issue 响应很快、社区活跃
- 了解 MongoDB
  - 官网文档 https://docs.mongodb.com/
  - MongoDB中文社区 www.mongoing.com
  - 阿里云栖社区 https://yq.aliyun.com/groups/11

2016 The Computing Conference

THANKS

2016 杭州·云栖大会
THE COMPUTING CONFERENCE