

# MongoDB应用运维与架构实践

李丹

## 移动应用



## 电商



## 内容管理



The New York Times



Forbes



sitecore  
Own the experience

## 360度信息面板，单一视图

MetLife



Telefonica

citi



stripe



## 物联网



Telefonica

BOSCH



阿里云  
aliyun.com

ADP

## 云计算



## 大数据



# 数据库选型之争

## 业务应用规模

实例数量 : 2570+

日访问量 : 263亿+

业务覆盖率: 90%+

DBA人数 : 1.5人

实例增长趋势图



# 数据库选型之争

- 无多文档**事务**及多表**关联查询**需求
- 业务快速迭代，需求频繁变动行业
- 单集群并发过大无法支撑业务增长
- 数据量增长预期TB及以上存储需求
- 期望要求99.999%数据库高可用场景

## 数据库版本选择

版本	关键特性	建议
2.X	index、writeConcern、readPreference	不稳定
3.0	Pluggable Storage Engine、Wiredtiger、improved mmapv1	不稳定
3.2	Raft 协议、文档校验、部分索引、inMemory、\$lookup	不维护
3.4	并行复制、sharding迁移改进、collation、\$facet、\$graphLookup	强烈建议使用
3.6	安全、并行性能、\$lookup增强、在线oplog维护、在线添加认证	非常建议使用
4.0	多文档事务Transaction	已经发布



# 目录

- » 特性介绍
- » 部署架构
- » 最佳实践

## 特性介绍





## 特性介绍

### 传说中的“数据丢失”

» 切换后的 **RollBack**

**Primary**

```
{ id:1,name:'freedom',sex:'man' }  
{ id:2,name:'Alice',sex:'woman' }  
{ id:3,name:'Anita',sex:'woman' }
```

**Secondary**

```
{ id:1,name:'freedom',sex:'man' }  
{ id:2,name:'Alice',sex:'woman' }
```

**Secondary**

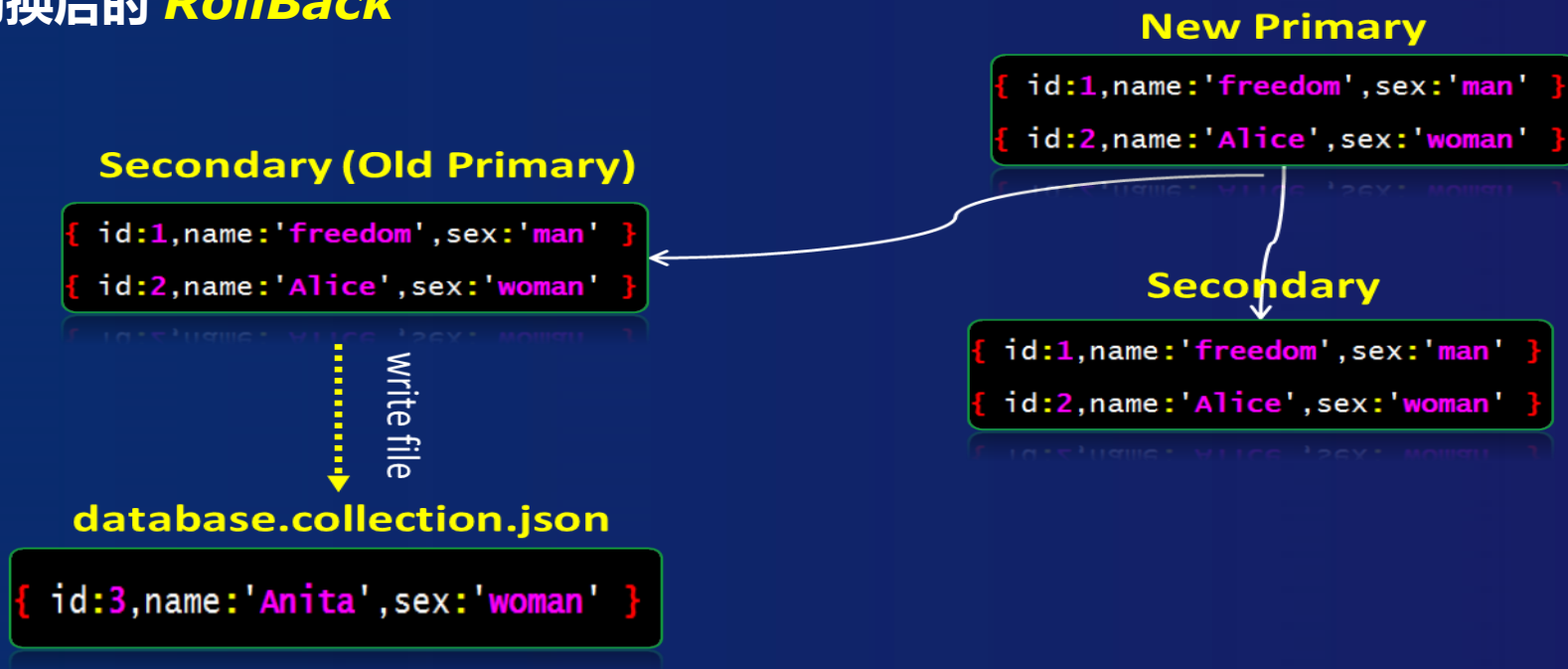
```
{ id:1,name:'freedom',sex:'man' }  
{ id:2,name:'Alice',sex:'woman' }
```



## 特性介绍

### 传说中的“数据丢失”

#### » 切换后的 **RollBack**



# 特性介绍

## 关于“写一致性”

### » 关于写策略 **Write Concern**

{ *w*: value, *j*: boole, *wtimeout*: n }

- **w**

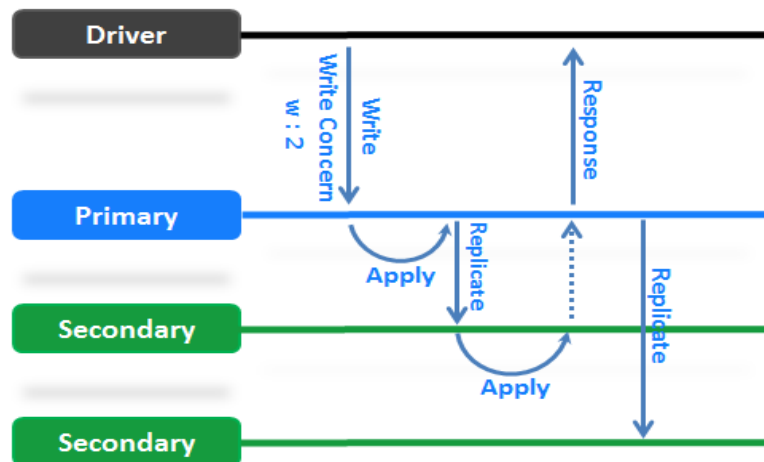
写入到指定个数或特定的实例  
(0、n、**majority** 或 *Tag*)

- **j**

写操作已确认写入 *journal* 日志

- **wtimeout**

写入请求等待确认的超时时间

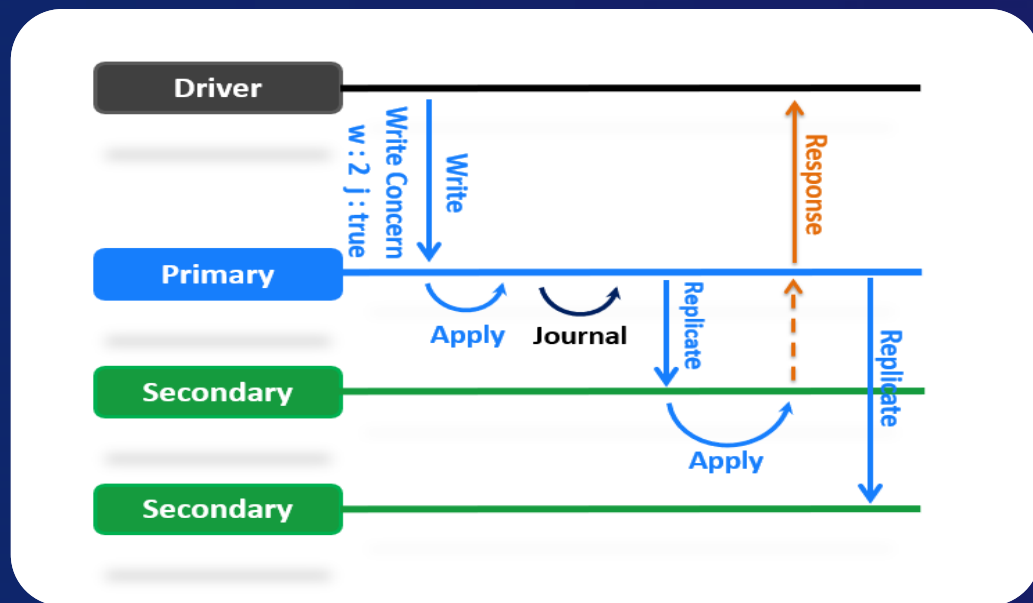


# 特性介绍

## » 关于写策略 *Write Concern*

`{ w: 2, j:true, wtimeout: 5000 }`

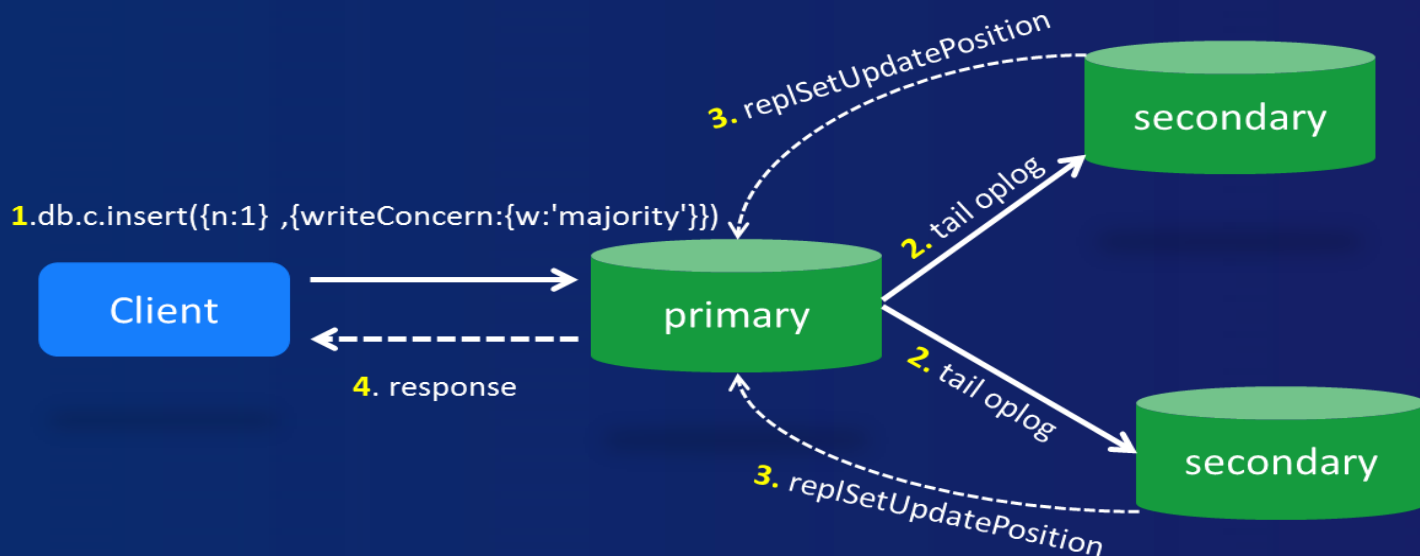
- 区别不同版本间主从差异性
- 服务端可通过复制配置控制



# 特性介绍

## 关于“写一致性”

关于写策略 **Write Concern** 原理





# 目录

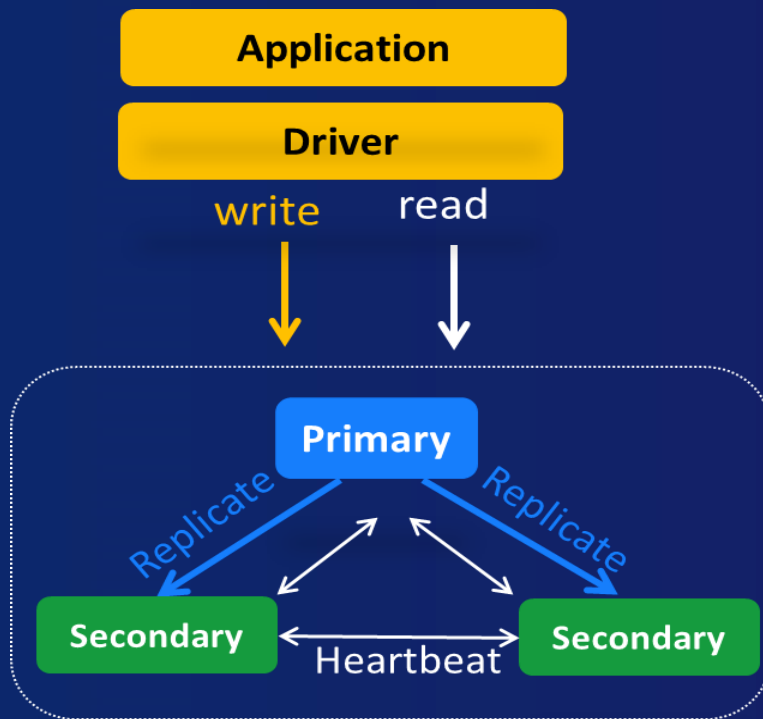
- » 特性简介
- » 部署架构
- » 最佳实践



# 部署架构

## » 副本集 (*Replica Set*)

- 最小资源
- 自动选主
- 维护简单
- 连接原理



投票节点

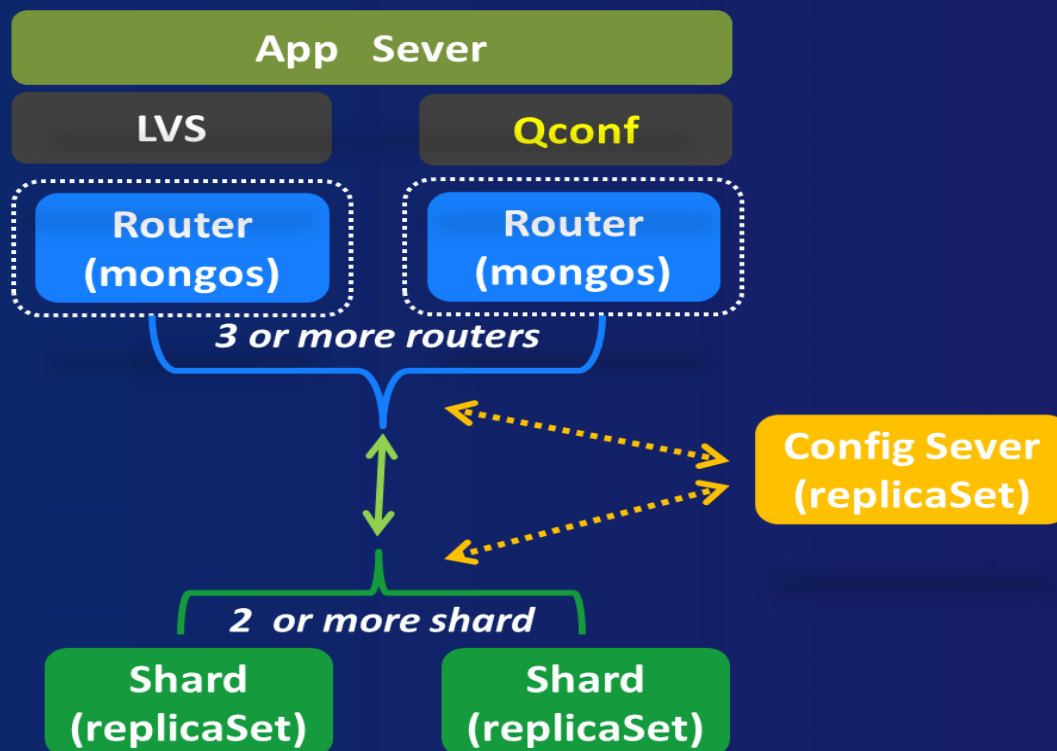
隐藏节点

延迟节点

# 部署架构

## 分布式集群 (*mongos*)

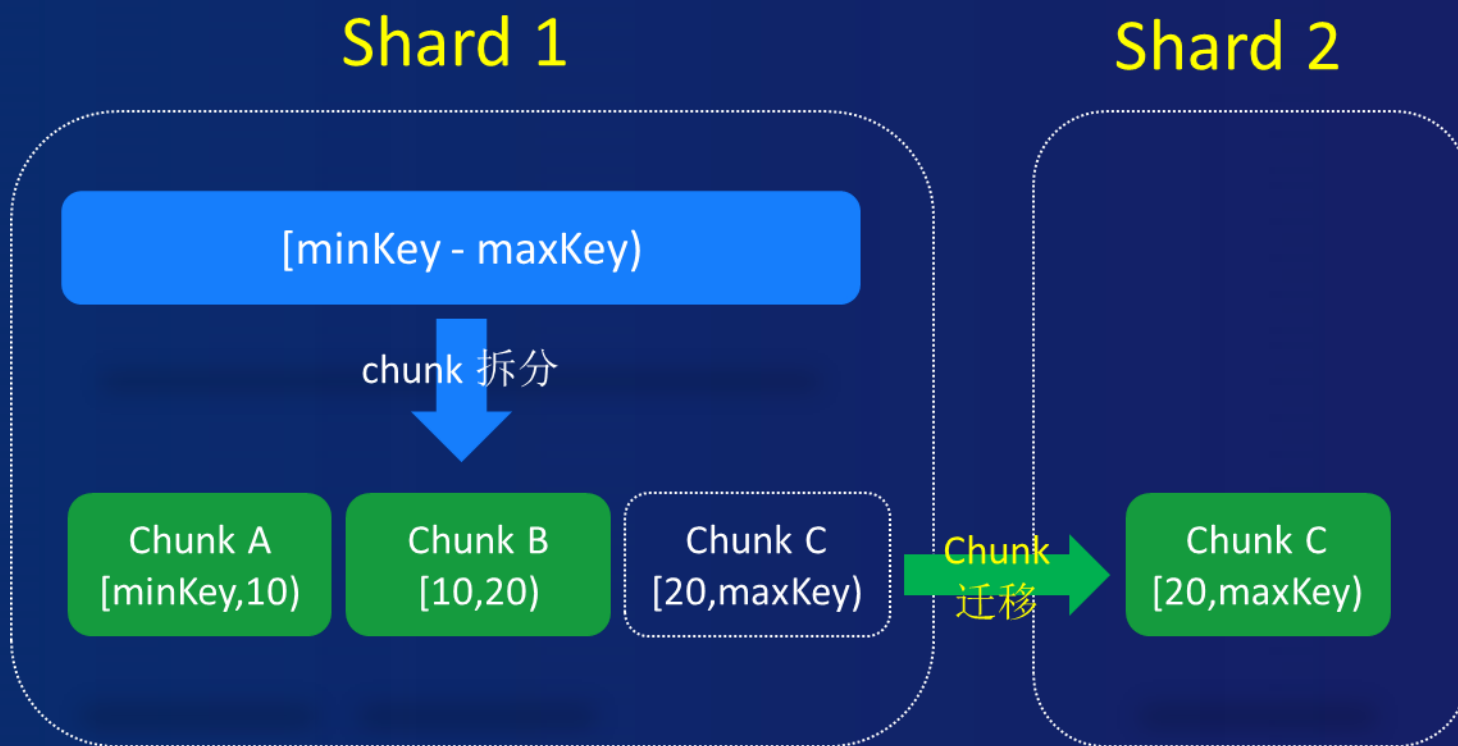
- 运维复杂
- 读写扩展
- 弹性容量
- 自动均衡
- 数据切分





# 部署架构

## » 关于 *mongos* 的 *chunk*



# 部署架构

## » 关于mongos的shard key

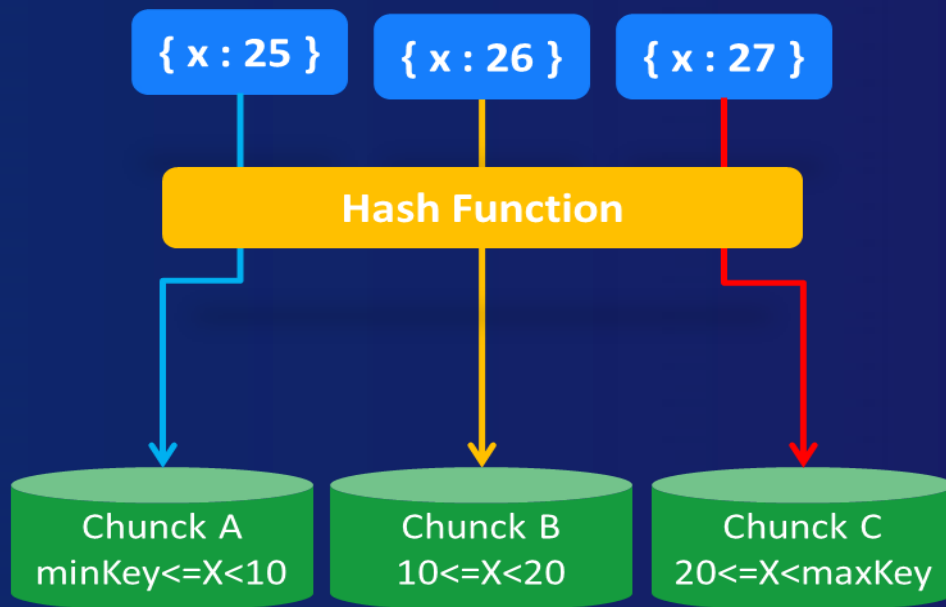
### - HASH 分片

数据分布相对均匀(写分散)

高效等值查询场景

### - 片键考量

- 基数 (*Cardinality*)
- 频率 (*Frequency*)



# 部署架构

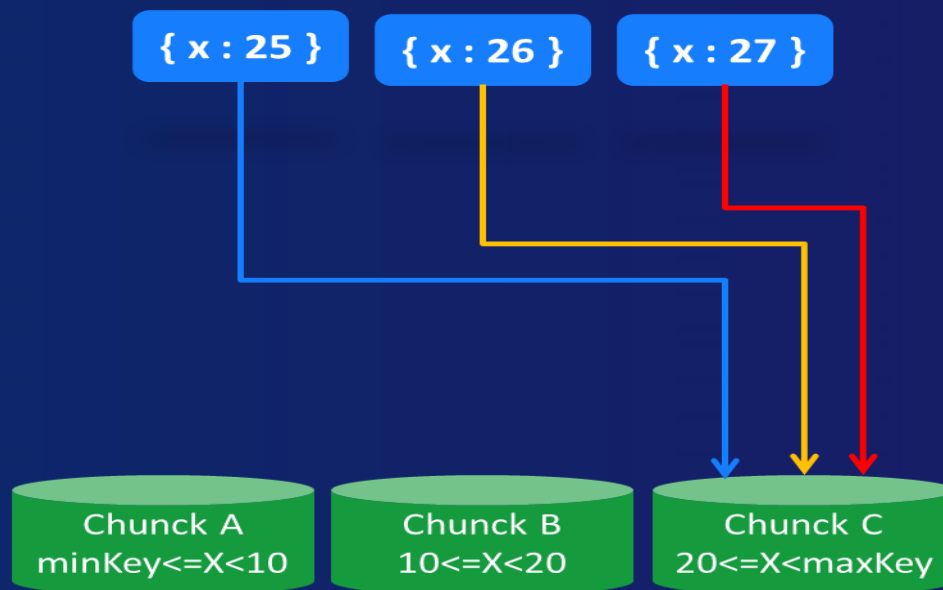
## » 关于 *mongos* 的 *shard key*

### - *RANGE* 分片

适合于范围查询（读局部）

### - 片键考量

- 基数（*Cardinality*）
- 频率（*Frequency*）
- 单调（*Monotonically*）



# 部署架构

## » 关于 *mongos* 的 *shard key*

### - *ZONES* 分片

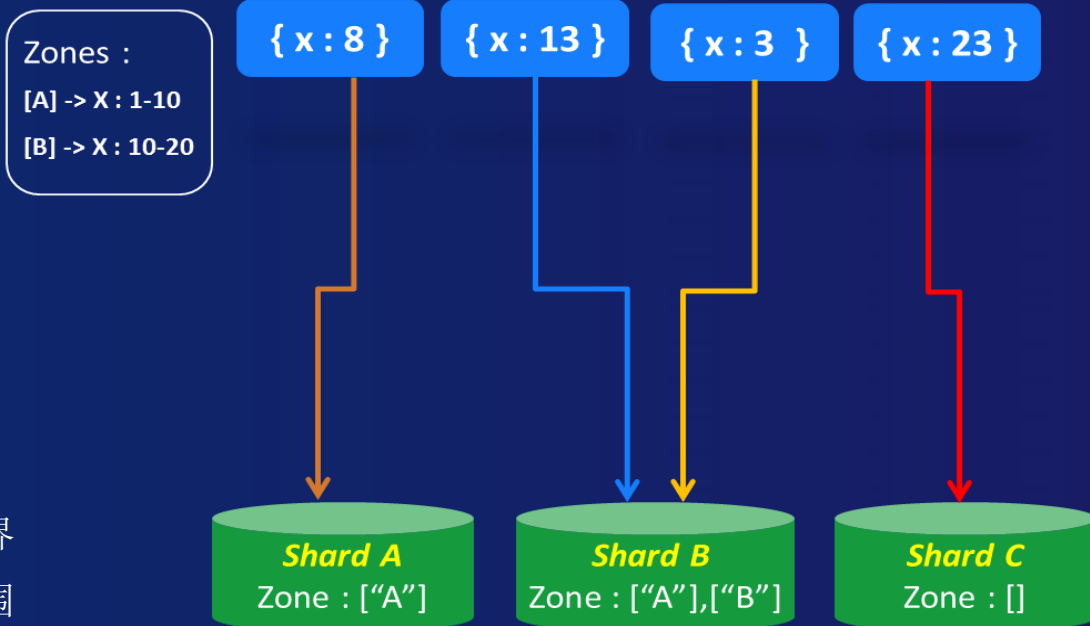
自定义区域分片

### - 适合场景

- 基于硬件性能差异的数据切分存储
- 特定地理位置数据与应用服务最近

### - 注意事项

- ✓ 区域覆盖的范围包含下界不含其上界
- ✓ 区域不可共享范围亦不能有交叉范围





# 部署架构

## » *mongos* 使用限制

- ✓ *shard key* 值不允许更新亦不可在线变更*shard key*
- ✓ 单文档*update*、*delete*、*remove* 条件必须带*shard key*
- ✓ *count*求分片集合总记录数不准可用*aggreagte*替代
- ✓ 不支持*group*操作可用*aggregate* 替代
- ✓ 不支持使用 *geoSearch* 命令的地理位置查询操作



# 目录

» 特性简介

» 部署架构

» 最佳实践



# 最佳实践

- ✓ 常见认证问题
- ✓ 相关查询优化
- ✓ 其他使用事项



# 最佳实践

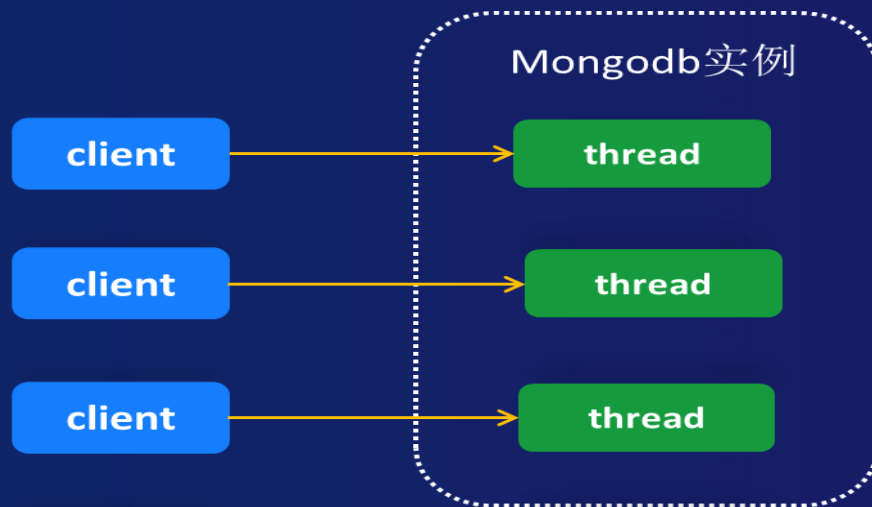
## » 认证问题 - 合理配置连接

### ✓ 线程-连接 网络模型

- 每个连接需分配1M的堆内存
- 大量连接创建与销毁开销大

### ✓ 连接资源控制

- 限制最大连接`maxConns`
- 配置连接池`maxPoolSize`
- 调整`cursorTimeoutMillis`



# 最佳实践

## 认证问题 – 最新认证模式的坑

### ✓ SCRAM-SHA-1的优势

- 更强的加密散列函数SHA-1
- Client与Server端双向认证

### ✓ 引发问题

- 高并发短连接负载飙升

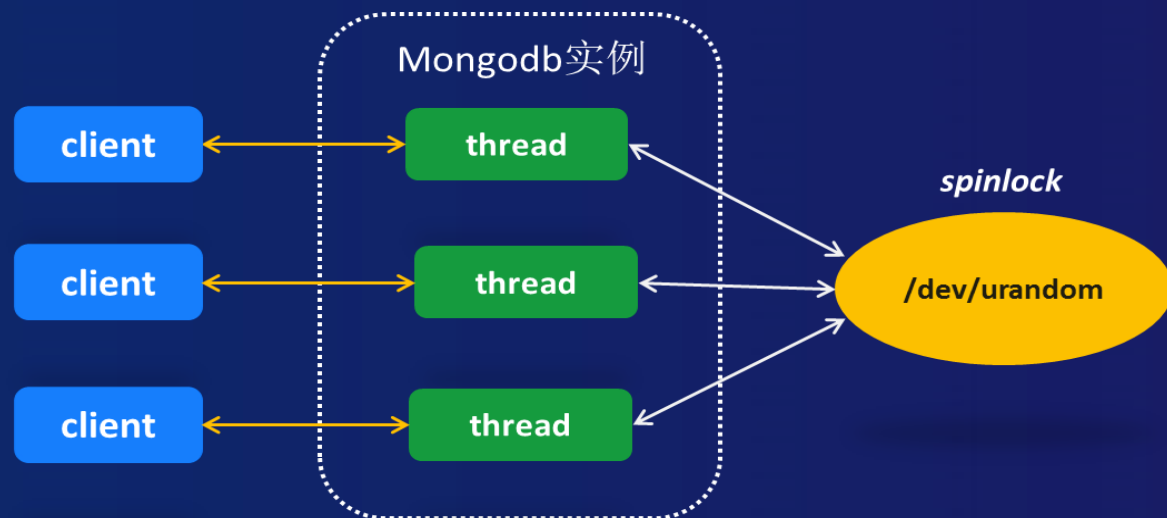
### ✓ 临时解决

client:

`authMechanism='MONGODB-CR'`

server:

`system.version -> currentVersion:3`





# ▶ 最佳实践

- ✓ 常见认证问题
- ✓ 相关查询优化
- ✓ 其他使用事项



# 最佳实践

## » 查询优化 - 读优先级控制 (*readPreference*)

- primary
- *primary Preferred*
- secondary
- *secondary Preferred*
- nearest

# 最佳实践

## » 查询优化 – 索引类型及创建

常见索引类型：

- 单列、多列索引
- 多key索引（Multikey Index）
- 哈希索引（Hashed Index）
- 地理索引（Geospatial Index）
- 文本索引（Text Index）

常见索引属性：

- ✓ 唯一索引（unique index）
- ✓ TTL索引（expired index）
- ✓ 部分索引（partial index）
- ✓ 稀疏索引（sparse index）

正确创建索引：

- 后台创建索引  
`db.test.createIndex({name:"hashed"},{background:true})`
- 批量创建索引  
`db.runCommand({createIndexes:"test",indexes:[{....}]})`

# 最佳实践

## » 查询优化 – 理解索引最左前缀原则

### ✓ 哪些查询可能走索引

1、2、3、4

### ✓ 查询包含最左索引字段

- 以索引创建顺序为准
- 与查询字段顺序无关

### ✓ 单个索引覆盖最多查询

索引: { **a** : 1 , **b** : 1 , **c** : 1 }

1. `db.test.find({a:"hello"})`
2. `db.test.find({b:"hello",a:"soga"})`
3. `db.test.find({a:"hello",b:"soga",c:"666"})`
4. `db.test.find({c:"hello",a:"233"})`
5. `db.test.find({b:"hello",c:"233"})`
6. `db.test.find({b:"hello"})`
7. `db.test.find({c:"hello"})`

# 最佳实践

## » 查询优化 – 分页查询

### ✓ 普通分页查询

```
db.test.find({ name : "li" }).skip(10000).limit(3)
```

- 优点: 使用简单粗暴效率低

### ✓ 一种新的思路

```
db.test.find({ _id : { $gt : 3 }, name : "li" }).limit(3)
```

- 优点: 顺序分页总体效率较高
- 缺点: 跳跃分页第一次效率低

{ \_id : 1 , name : "li" }

{ \_id : 2 , name : "li" }

→ { \_id : 3 , name : "li" }

{ \_id : 4 , name : "li" }

{ \_id : 5 , name : "li" }

→ { \_id : 6 , name : "li" }

{ \_id : 7 , name : "li" }

{ \_id : 8 , name : "li" }

→ { \_id : 9 , name : "li" }





# 最佳实践

## » 查询优化 – 批处理与更新操作

- `bulkWrite([...], {order>false})`
- `insertMany([...], {order>false})`
- `update({}, {upsert>true})`
- `findAndModify({...}, upsert>true)`



# ▶ 最佳实践

- ✓ 常见连接问题
- ✓ 相关查询优化
- ✓ 其他使用事项

# 最佳实践

## » 其他事项-数组更新

{name:["张三","李四"]}

```
db.array.update({},{$push:{name:{$each:["王二"]}}});
```

```
db.array.update({},{$push:{name:{$each:["麻子"],$position:0}}});
```

{"\_id":ObjectId("xxxxxxxx "), "o":{"\$set":{"name.2":"王二"}}

{"\_id":ObjectId("xxxxxxxx"), "o":{"\$set":{"name":["麻子","张三","李四","王二"]}}

## 关于「3306π」社区

围绕MySQL核心技术，将互联网行业中最重要  
的数据化解决方案带到传统行业中  
囊括**其他开源技术**，redis、MongoDB、Hbase、  
Hadoop、ElasticSearch、Storm、Spark等  
在全面互联网化的大趋势下，将互联网新鲜的核心技  
术理念带到传统行业里，构建良好交流互动环境  
分享干货知识，即便是赞助商，也要求如此，拒绝  
放水


「3306π」社区，欢迎您的加入



社区公众号



社区QQ群



谢谢