MONGODB应用与优化实践

分享人：李丹

目录

特性简介

高性能

文档化

高可用

强压缩

分布式

一致性

特性简介

关于“写一致性”

- 被忽视的连接参数 *fsync* 及 *journal*

- *fsync*

  每次写入强制**数据**刷盘

- *journal*

  每次写入强制**日志**刷盘

```php
<?php

$username    = "mongo";
$password    = "xxxxx";

$mongo_servers = "mongodb://10.206.86.1:37447,10.206.86.2:37447" ;

$options = array(
//    'replicaSet' => $replicaSet ,
      'username'   => $username   ,
      'password'   => $password   ,
      'db'         => 'admin'     ,
      'journal'    => true        ,
      'fsync'      => true        ,
      'connectTimeoutMS' => 30000 ,
      'readPreference'   => MongoClient::RP_SECONDARY_PREFERRED,
);

$conn = new MongoClient($mongo_servers, $options);
```

特性简介

关于“写一致性”

关于写策略 *Write Concern*

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

- ***w***

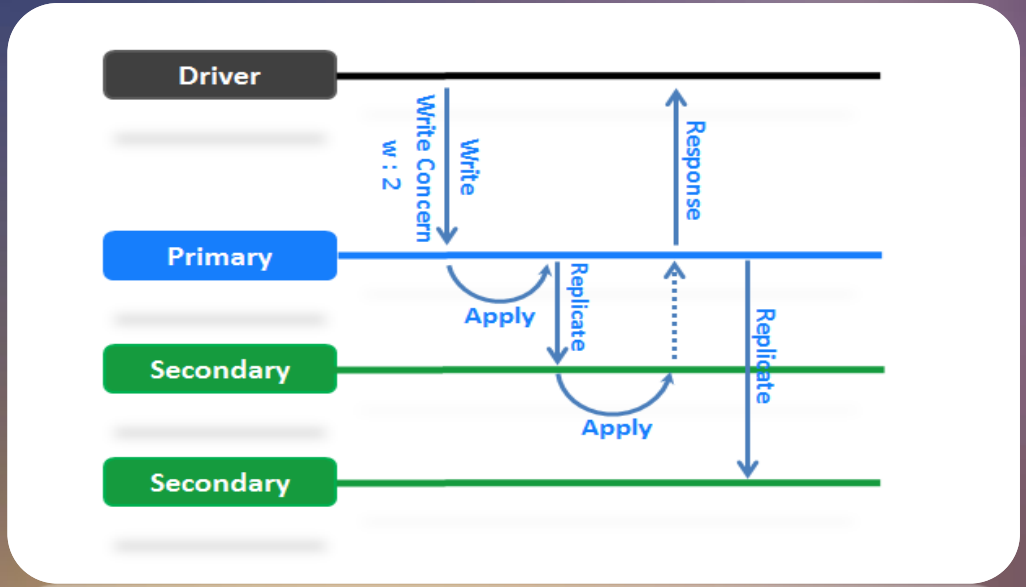
  写入到指定个数或特定的实例
  （0、n、***majority*** 或 *Tag*）

- ***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



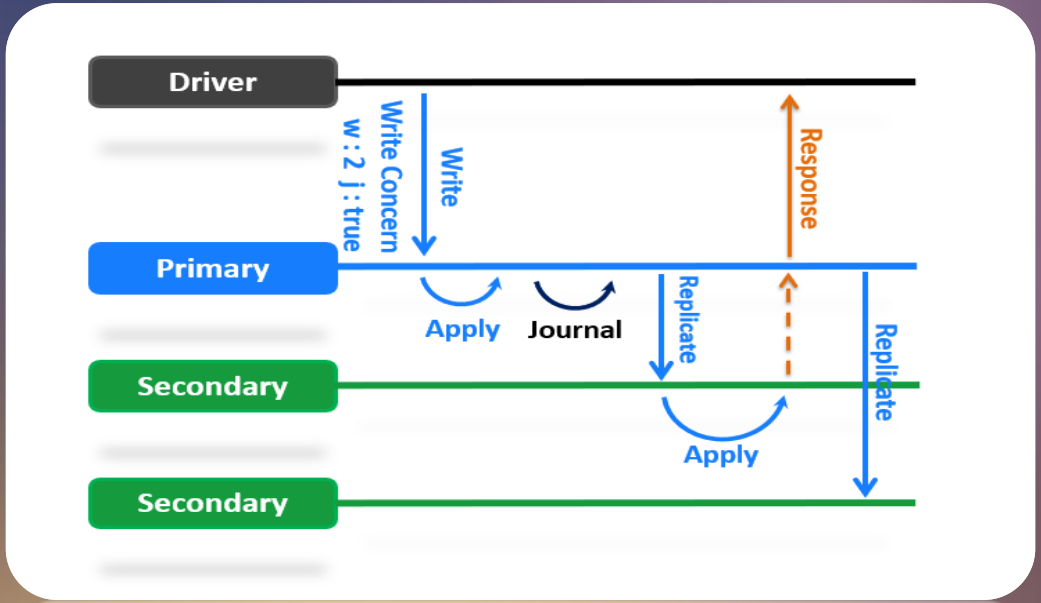
特性简介

关于“写一致性”

关于写策略 *Write Concern*

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

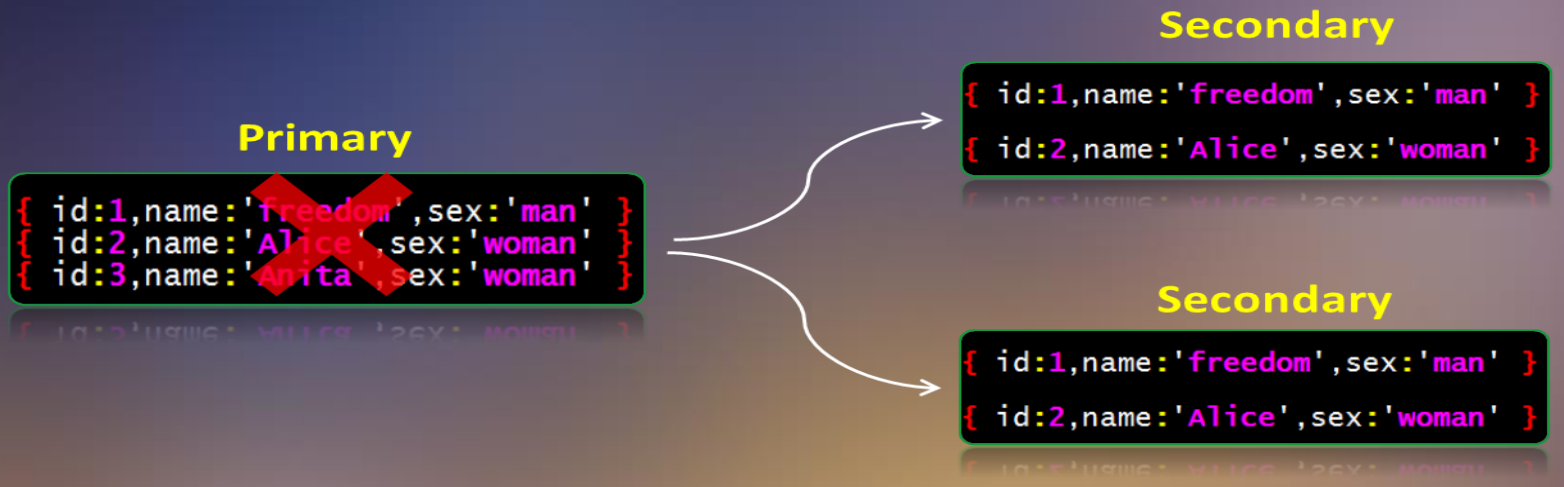
- 区别主从差异



特性简介

关于“写一致性”

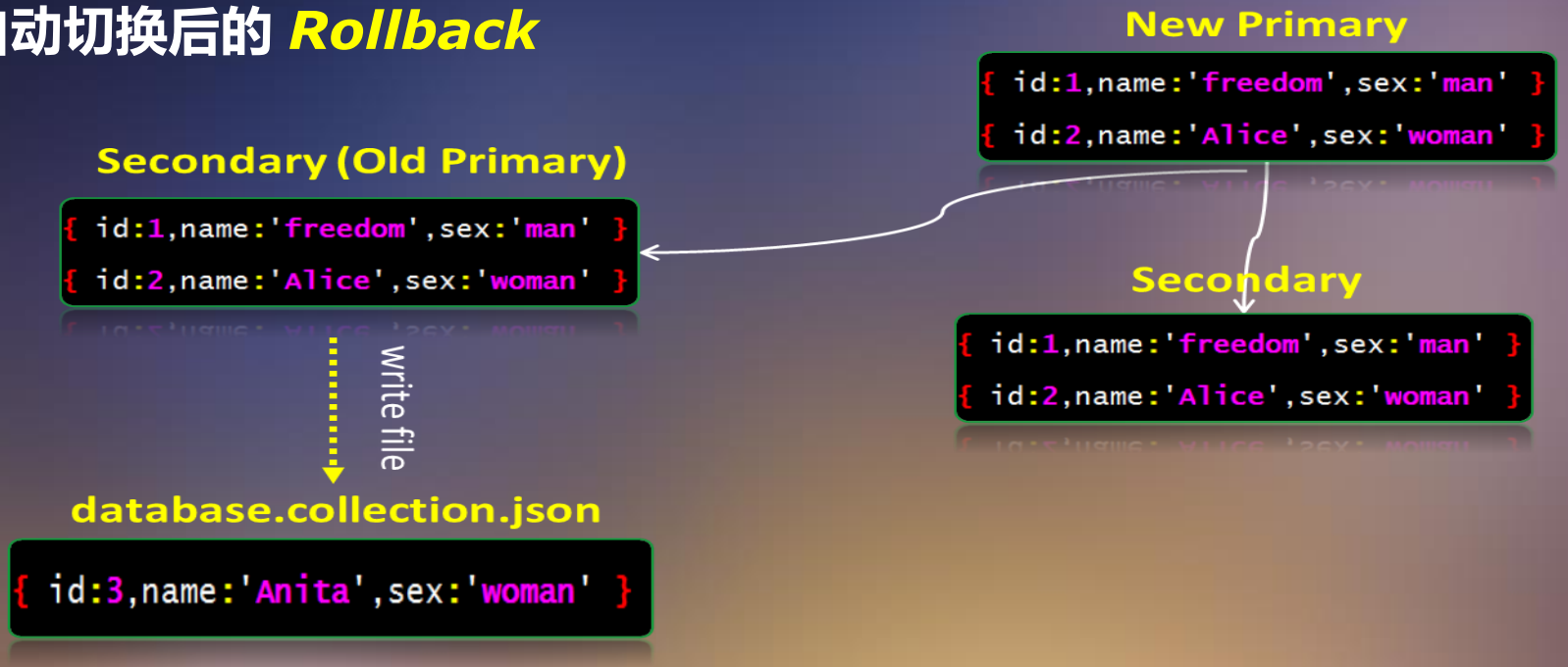
- 自动切换后的 ***Rollback***



特性简介

关于“写一致性”

- 自动切换后的 ***Rollback***



目录

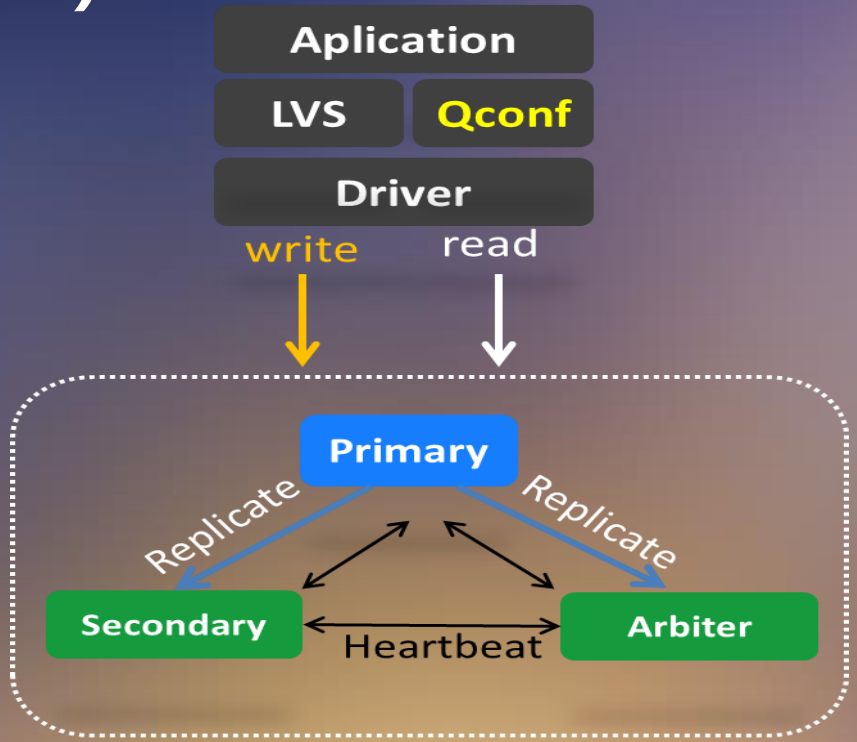
部署架构

副本集（*replica Set*）

- 最小资源

- 维护简单

- 使用方便



部署架构

副本集连接 *is not master* 问题

- 问题原因

驱动以*replicaSet*参数

自动识别副本集主从库，

否则请求不区分主从直接

发送至节点

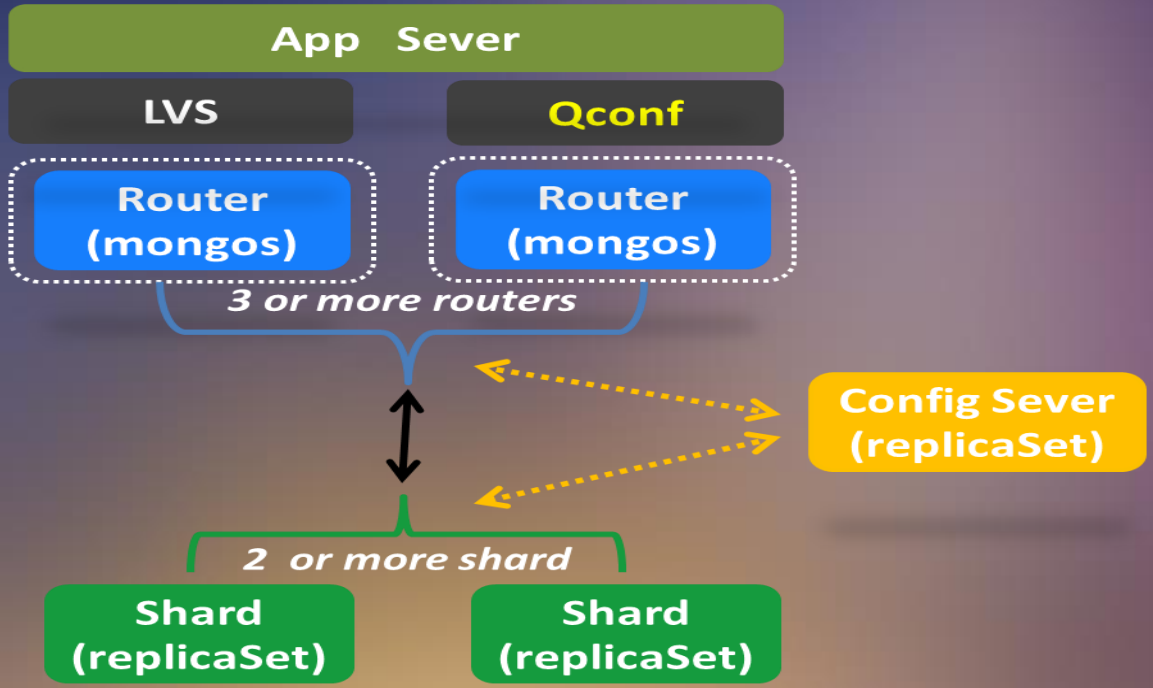
```php
<?php

$username    = "mongo";
$password    = "xxxxx";
$replicaSet = "7001" ;

$mongo_servers = "mongodb://10.128.1.1:7001,10.128.1.2:7001" ;

$options = array(
    'replicaSet' => $replicaSet ,
    'username'   => $username   ,
    'password'   => $password   ,
    'db'         =>'admin'      ,
    'connectTimeoutMS' => 30000 ,
    'readPreference'   => MongoClient::RP_SECONDARY_PREFERRED,
);

$conn = new MongoClient($mongo_servers, $options);

);
```

字符串类型

部署架构

分布式集群（*mongos*）

- 主要特性
  - 结构复杂
  - 读 写扩展
  - 弹性容量
- 注意事项
  - ✓ 数据切分



部署架构

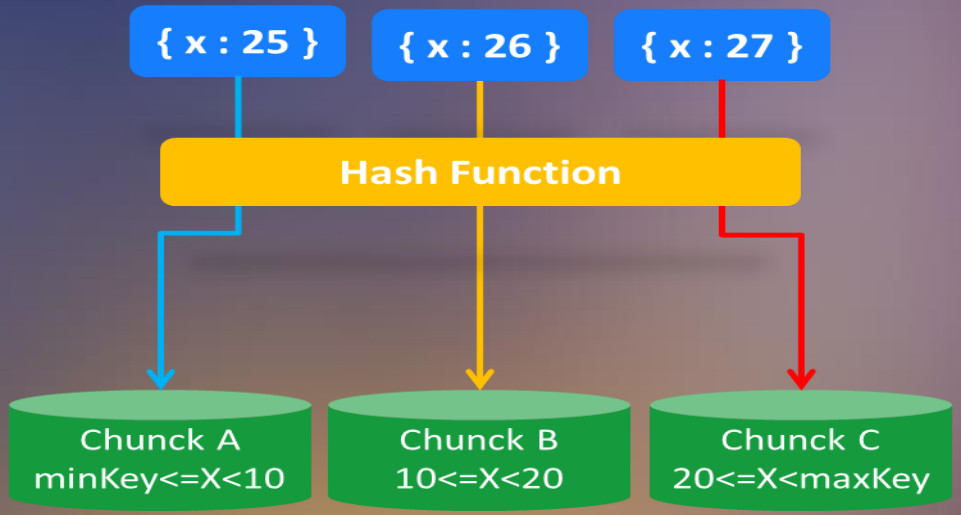
关于*mongos*的*shard key*

- *HASH* 分片

  数据分布均匀

- 适合场景

  高效等值查询



部署架构

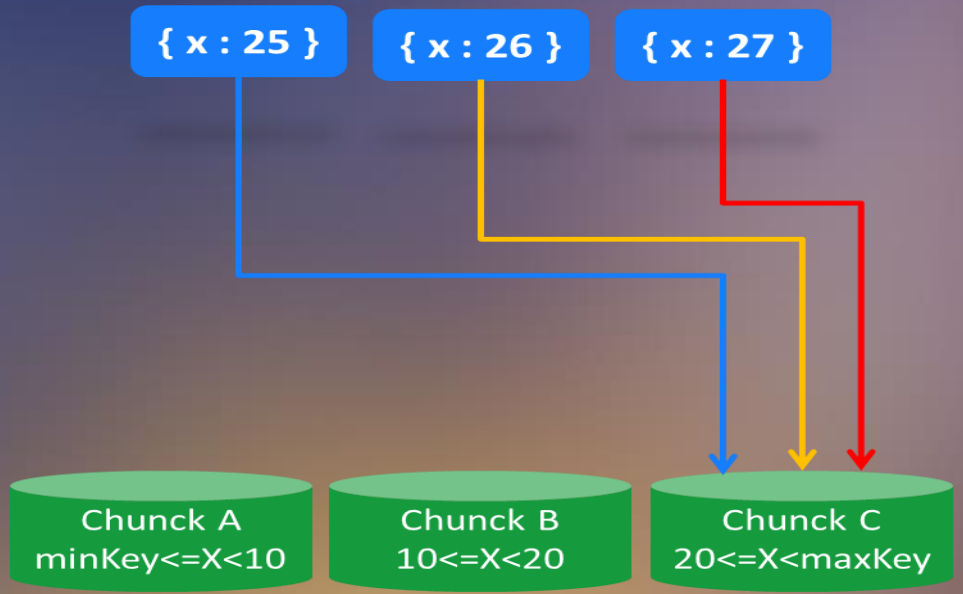
关于*mongos*的*shard key*

- ***RANGE***分片

  适合范围查询

- 片键选择

  - 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

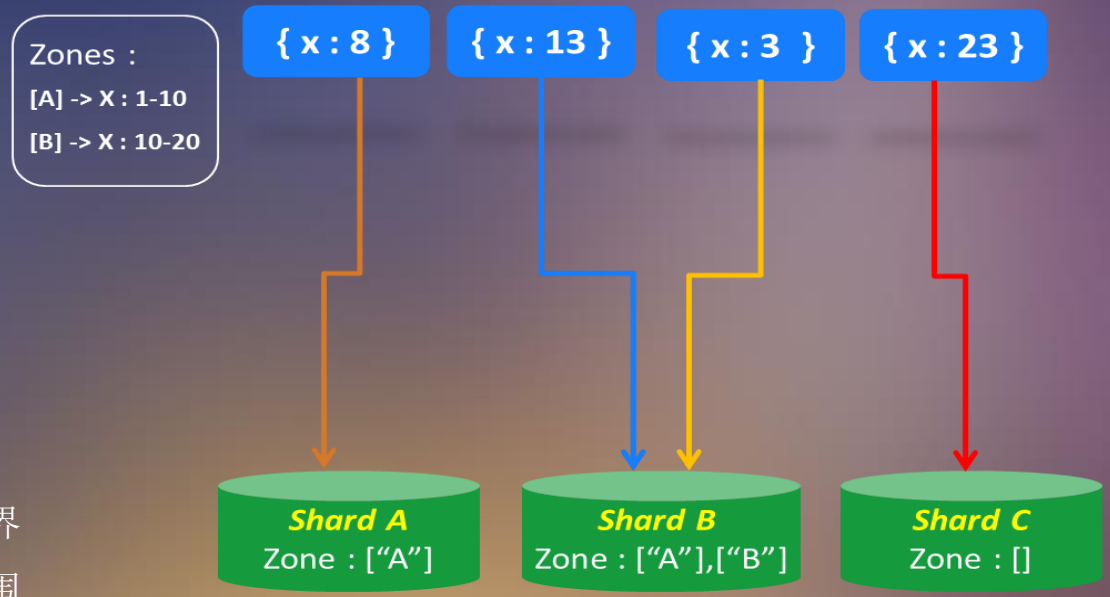
- 适合场景

  - 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

连接*mongos*的正确方式

- 连接说明

  - 程序直连*mongos*节点
  - 无需指定*replicaSet*参数
  - *mongos*节点建议配置3个
  - 考虑配置服务中心/LVS

```php
<?php

$username    = "mongo";
$password    = "xxxxx";

$mongo_servers = "mongodb://10.128.1.1:37777,10.128.1.2:37777,10.128.1.3:37777"

$options = array(
//   'replicaSet' => $replicaSet ,
     'username'   => $username   ,
     'password'   => $password   ,
     'db'         =>'admin'      ,
     'connectTimeoutMS' => 30000 ,
     'readPreference'   => MongoClient::RP_SECONDARY_PREFERRED,
);

$conn = new MongoClient($mongo_servers, $options);

);
```

也可以配置VIP

部署架构

mongos 使用限制及注意事项

- ✓ *shard key* 值不允许更新亦不可在线变更*shard key*
- ✓ 所有*update* 、*delete* 、*remove* 条件必须带*_id* /*shard key*
- ✓ *count*求分片集合总记录数不准可用*aggreagte* 替换
- ✓ 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

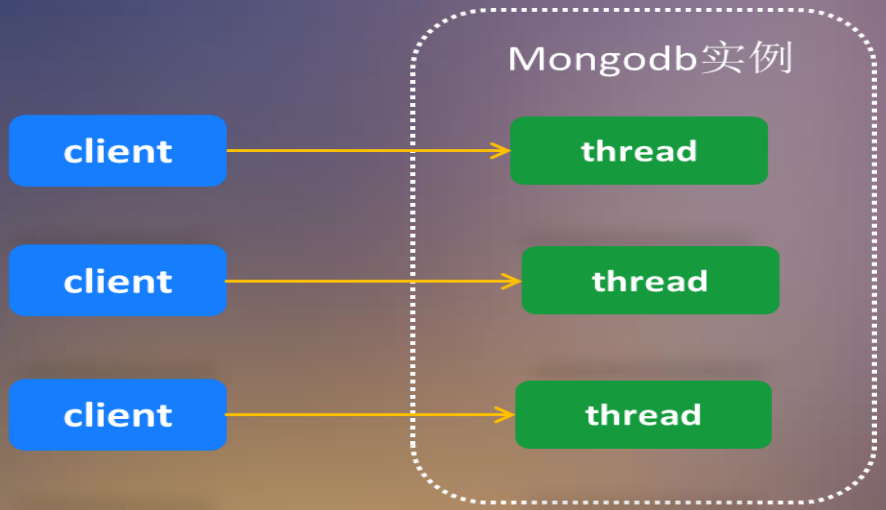
最佳实践

- ✓ 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 - 认证库与游标超时

- ✓ 正常账号密码登陆失败

  mongodb**34**/bin/mongo -uxyz -pxyz

  10.142.1.1:7003/abc (abc为业务库)

  authenticationDatabase=admin

- ✓ 为何我的查询自动超时退出

  调整*socketTimeoutMS*大小

```php
<?php
$username   = "mongo";
$password   = "xxxxx";

$mongo_servers = "mongodb://10.206.86.1:7007,10.206.86.2:7007";

$options = array(
    'replicaSet' => $replicaSet ,
    'username'   => $username   ,
    'password'   => $password   ,
    'db'         =>'admin'      ,
    'w'          => 'majority'  ,
    'connectTimeoutMS' => 30000 ,
    'socketTimeoutMS'  => 60000 ,
    'readPreference'   => MongoClient::RP_SECONDARY_PREFERRED,
);

$conn = new MongoClient($mongo_servers, $options);
```

最佳实践

认证问题 – 最新认证模式的坑

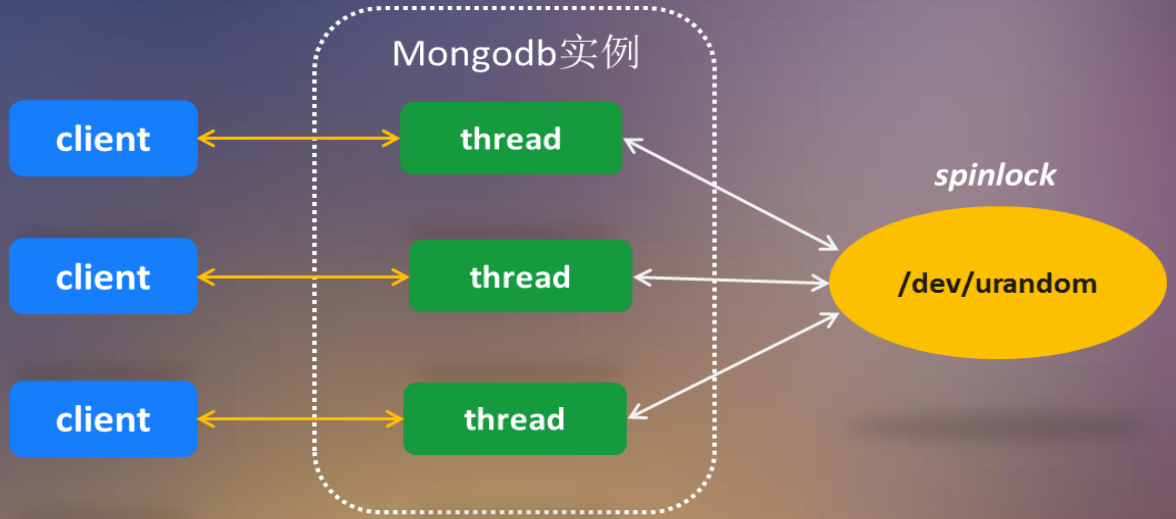
- ✓ *SCRAM-SHA-1*的优势
  - 更强的加密散列函数*SHA-1*
  - *Client*与*Server*端双向认证
- ✓ 引发的问题
  - 高并发短连接负载飙高
- ✓ 解决方法

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*）

- ✓ ***readPreference***
  - primary
  - ***primary Preferred***
  - secondary
  - ***secondary Preferred***
  - nearest

```php
<?php
$username   = "mongo";
$password   = "xxxxx";

$mongo_servers = "mongodb://10.206.86.1:7007,10.206.86.2:7007";

$options = array(
    'replicaSet' => $replicaSet ,
    'username'   => $username   ,
    'password'   => $password   ,
    'db'         =>'admin'      ,
    'w'          => 'majority'       ,
    'connectTimeoutMS' => 30000 ,
    'socketTimeoutMS'  => 60000 ,
    'readPreference'   => MongoClient::RP_SECONDARY_PREFERRED,
);

$conn = new MongoClient($mongo_servers, $options);
```

最佳实践

查询优化 – 读不回滚（ *readConcern* ）

- ***local***

  读取节点最新数据

- ***majority***

  写入大多数节点

  且commit的数据

- ***linearizable***

  读到已写入大多数节点且确

  认时间在读请求之前的数据

Read Concern	WiredTiger	MMAPv1
local	✓	✓
majority	✓	
linearizable	✓	✓

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

索引:{a:1,b:1,c:1}

- ✓ 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
- ✓ 最少索引覆盖最多查询

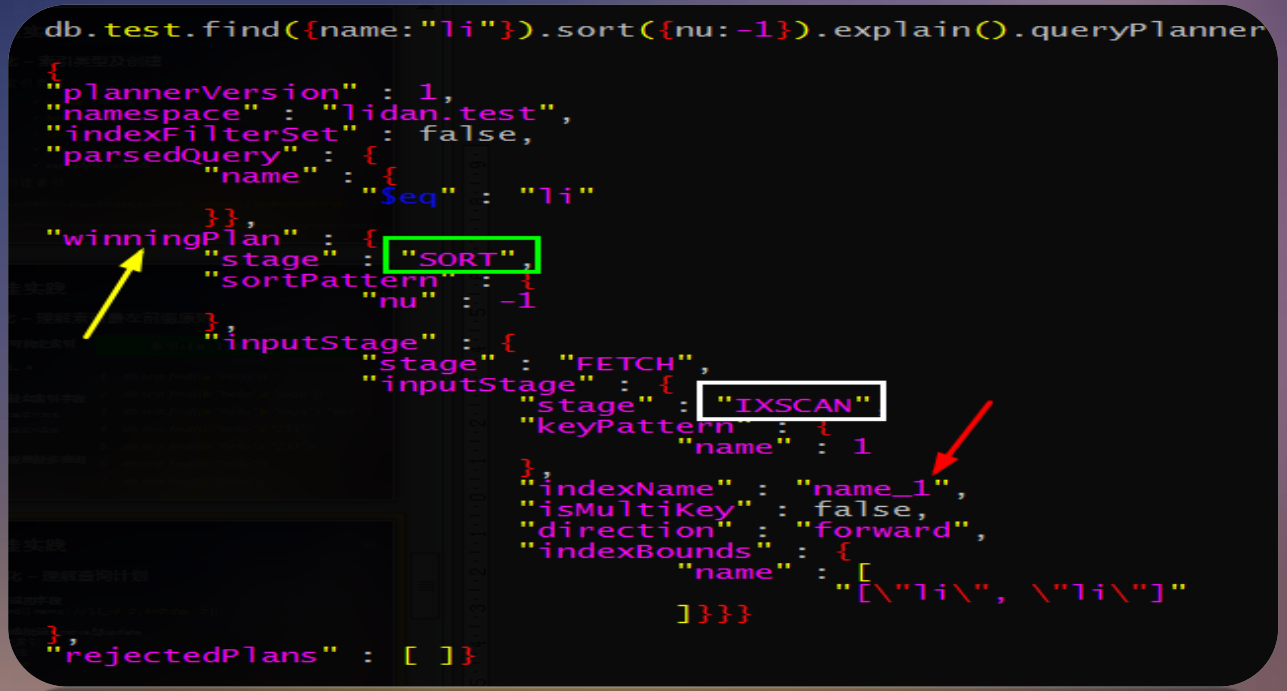
1. db.test.find({a:"hello"})
2. db.test.find({b:"hello",a:"soga"})
3. db.test.find({a:"hello",b:"soga",c:"666"})
4. db.test.find({c:"hello",a:"233"})
5. db.test.find({b:"hello",c:"233"})
6. db.test.find({b:"hello"})
7. db.test.find({c:"hello"})

最佳实践

查询优化 – 读懂查询计划

✓ 关键字

- *winningPlan*
- *rejectedPlans*
- *IXSCAN*
- *COLLSCAN*
- *SORT*



最佳实践

查询优化 – 干掉慢查询

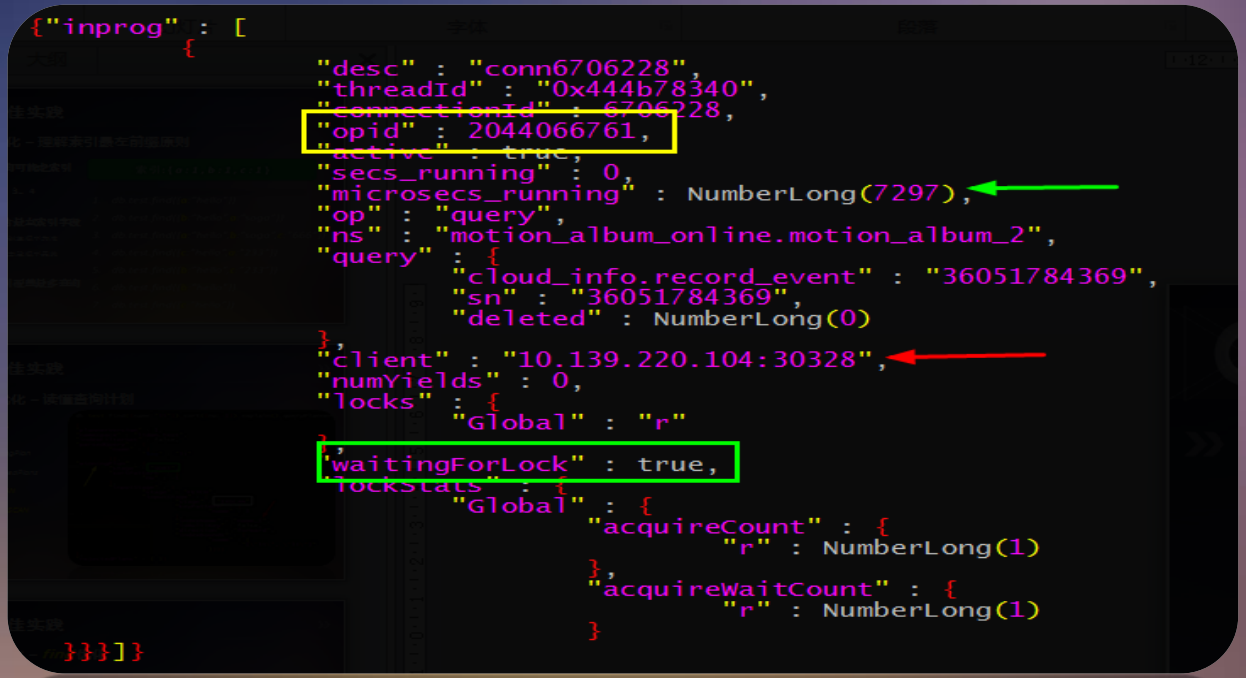
- 避免*Ctrl+c*

- 考虑*maxTimeMS(N)*

- 善用

  db.currentOp()

  \+ *db.killOp(opid)*



最佳实践

查询优化 – 普通查询优化

- ✓ 按需取字段

```
db.test.find({ name : /i/ },{ _id : 0 , birthday : 0 })
```

- ✓ 控制数据量

```
db.test.find({id:{$in:[1,2,3,4,5,6,7,8,9.....]}})
```

- ✓ 业务端排序

```
db.test.find({ name : /i/ },{ _id : 0 }).sort({age:1})
```

最佳实践

查询优化-插入优化

- ✓ 数据插入预计算以空间换时间
- ✓ 超长文本优先压缩或*HASH*处理
- ✓ 不常检索大字段可独立拆分存储
- ✓ 大批量高效插入考虑*bulkWrite*()

最佳实践

查询优化 – 更新、删除优化

- ✓ 批量更新与删除
  - 基于_id索引切分
- ✓ 原子查询与更新
  - *db.test.update({name:'freedom'},{$set:{a:1}},*
    *{**upsert:true**,**multi:true**})*
  - *db.findAndModify({query:{p:"3"},update:{*
    *$set:{a:"test"}},**upsert:true**})*

最佳实践

主要内容

- ✓ 常见连接问题
- ✓ 相关查询优化
- ✓ 其他使用事项

最佳实践

Q&A