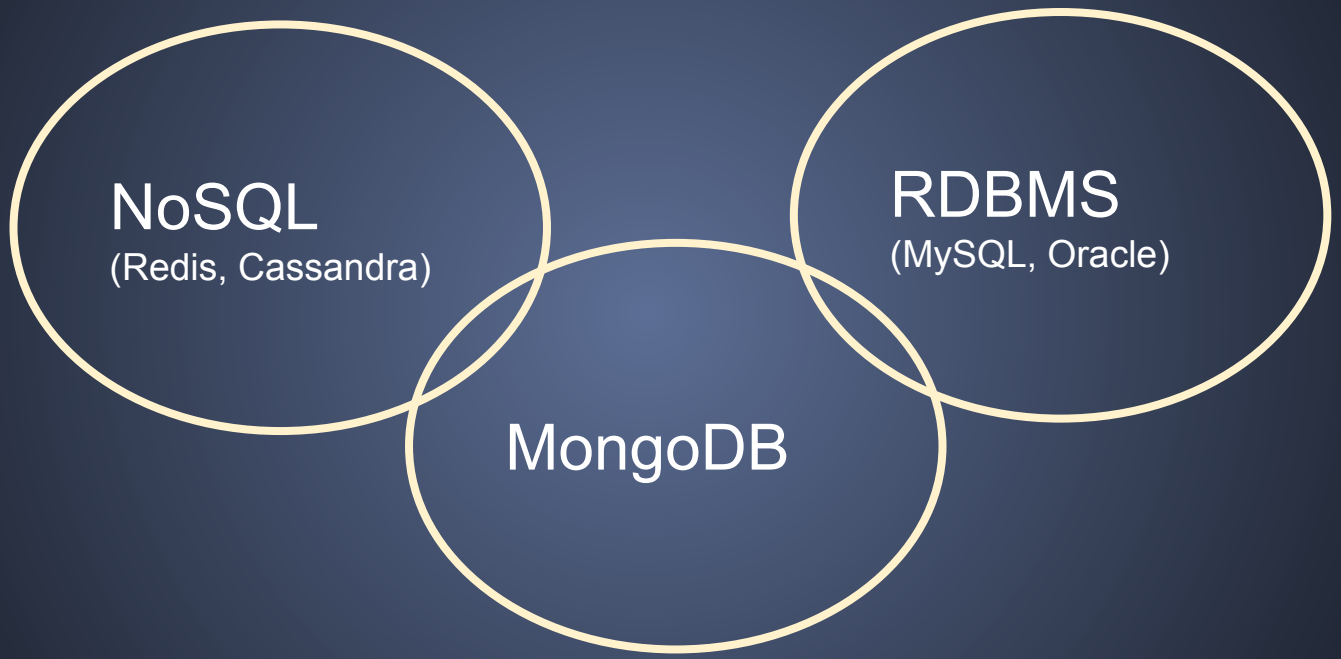
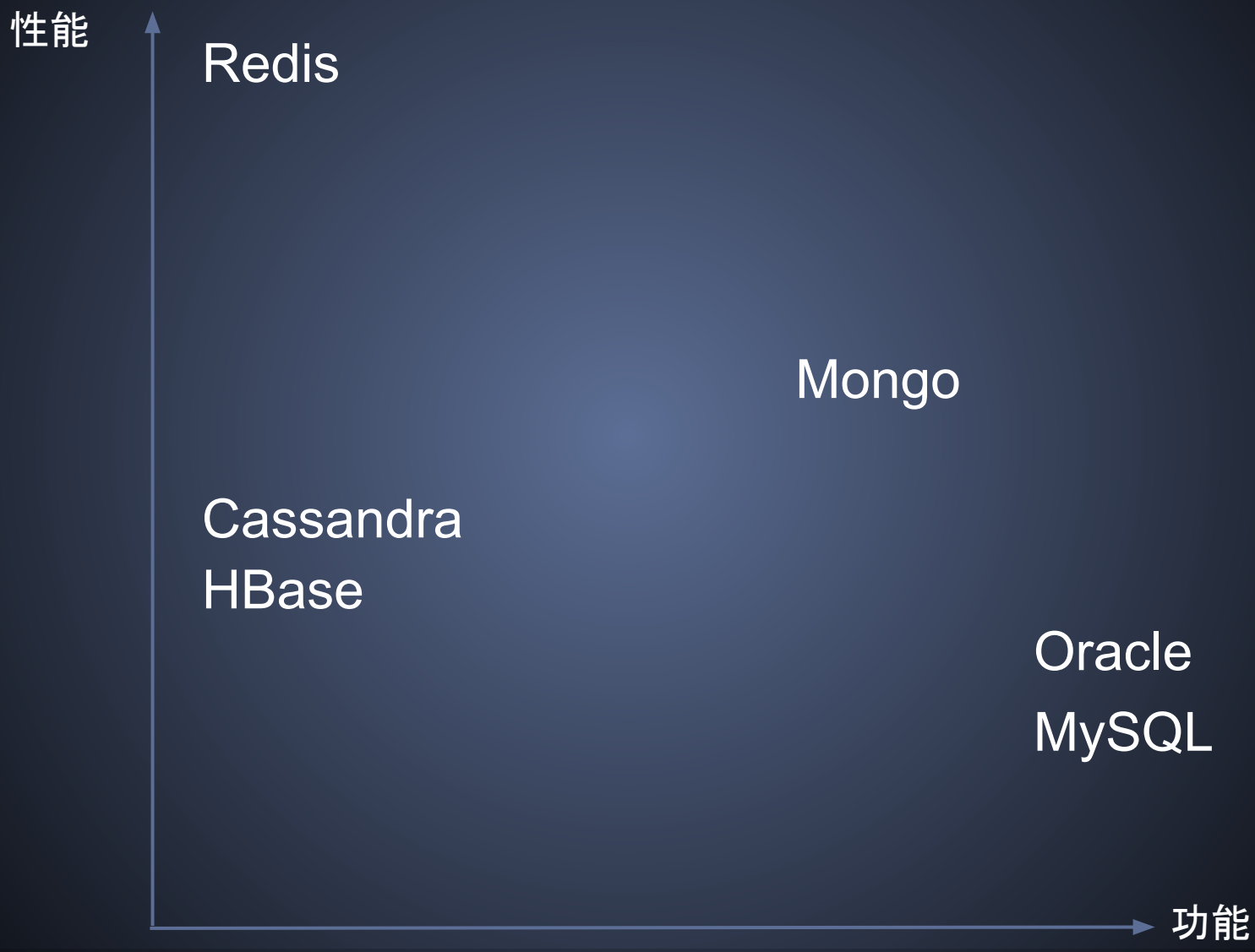
MongoDB @百度云

提纲

- MongoDB简介
- MongoDB @ 百度云
- 经验

MongoDB简介


NoSQL
(Redis, Cassandra)
RDBMS
(MySQL, Oracle)
MongoDB


性能
Redis
Mongo
Cassandra
HBase
Oracle
MySQL
功能

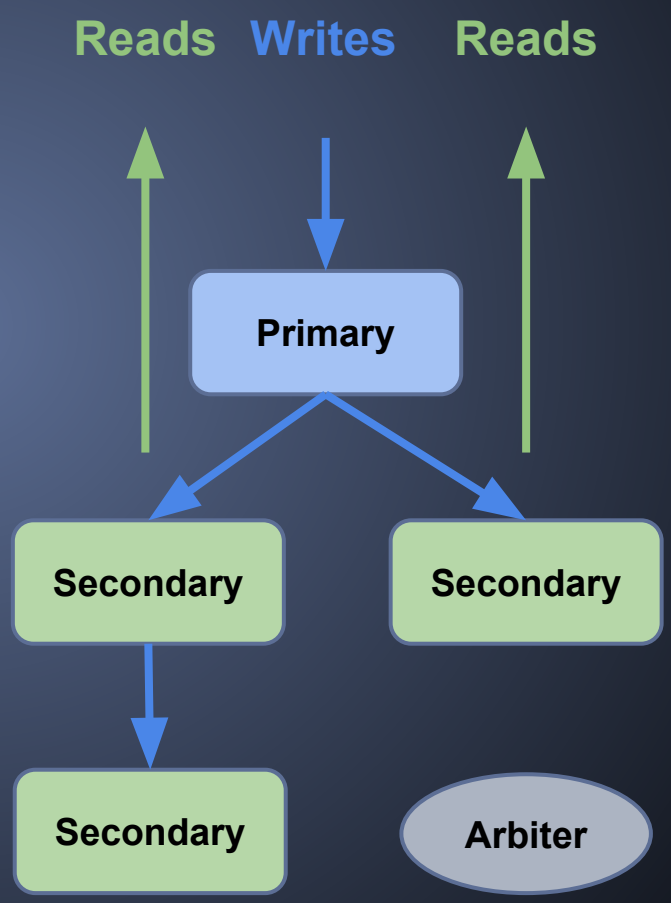
特点

- schema-free => 更容易使用
- 丰富的用法(数据&查询) => 满足各种需求
  - 嵌套json/地理查询/数组/MapReduce
- 没有事务 => 更高的性能
- 支持索引 => 相对于KV有明显优势
- 嵌入 javascript 引擎
- Replication
- Sharding

Replication

Replset

- Primary
- Secondary
- Arbiter



Replication

原理

- oplog
  - 类似MySQL Row模式
  - 同步: 拉
  - 可以设置w
- 选举
  - 投票
  - 选取optime 最新的secondary

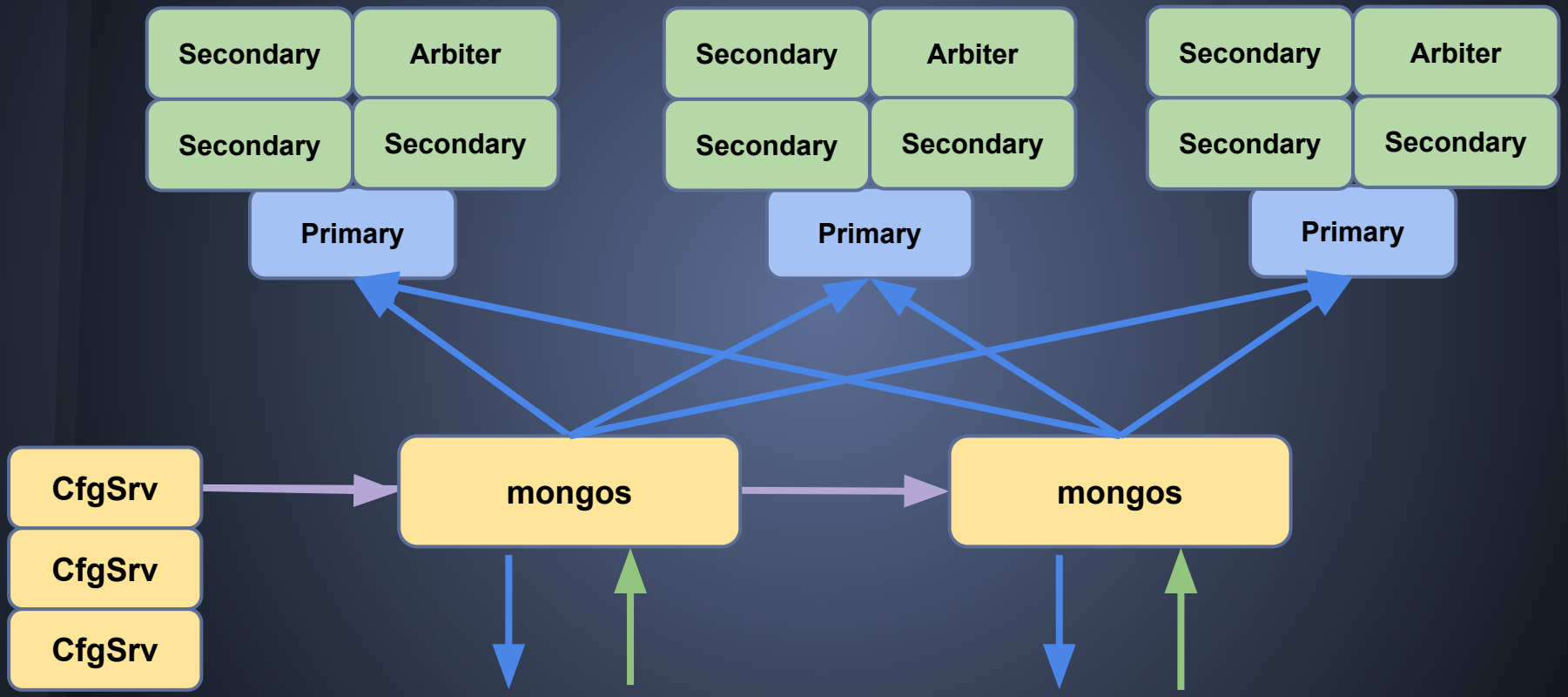
推荐结构:

  - 1主2从1备1 arbiter

Replication

- 作用
  - High availability
  - Read scaling
  - maintenance
- 问题
  - 主从延迟
  - 一致性
  - 主从切换时丢写

Sharding



- Shard(mongod) + mongos + cfgsrv

Sharding

- cfgsrv
  - 3 cfgsrv
  - 2段提交 确保强一致
  - down掉一个 => 只读
  - 路由表: 每个shardkey范围对应的shard
- Shard
  - Mongod / Replset
- mongos
  - router
  - cache路由信息

Sharding

- shardkey
  - 1个或多个字段
  - 每个document都必须有shardkey, 不能修改
- Chunk
  - Split
  - Move Chunk
    - 每次迁移一个chunk
    - 基准+增量
    - 源mongod主动更新路由, mongos被动更新路由

Sharding

- 操作
  - insert: 必须带shardkey
  - remove: routed or scattered
  - update: routed or scattered
  - query
    - with shardkey: routed
    - sort by shardkey: routed in order
    - no shardkey: scatter
    - sort by no shardkey: merge sort

Sharding

- 好处
  - auto-balance 全自动扩容/缩容
  - 通过mongos对客户端透明

- 问题
  - auto-balance 影响服务
  - shard过多时 扩容太慢
  - 路由表大时加载慢(10w条)
  - cfgsrv 不能迁移

MongoDB @ 百度云

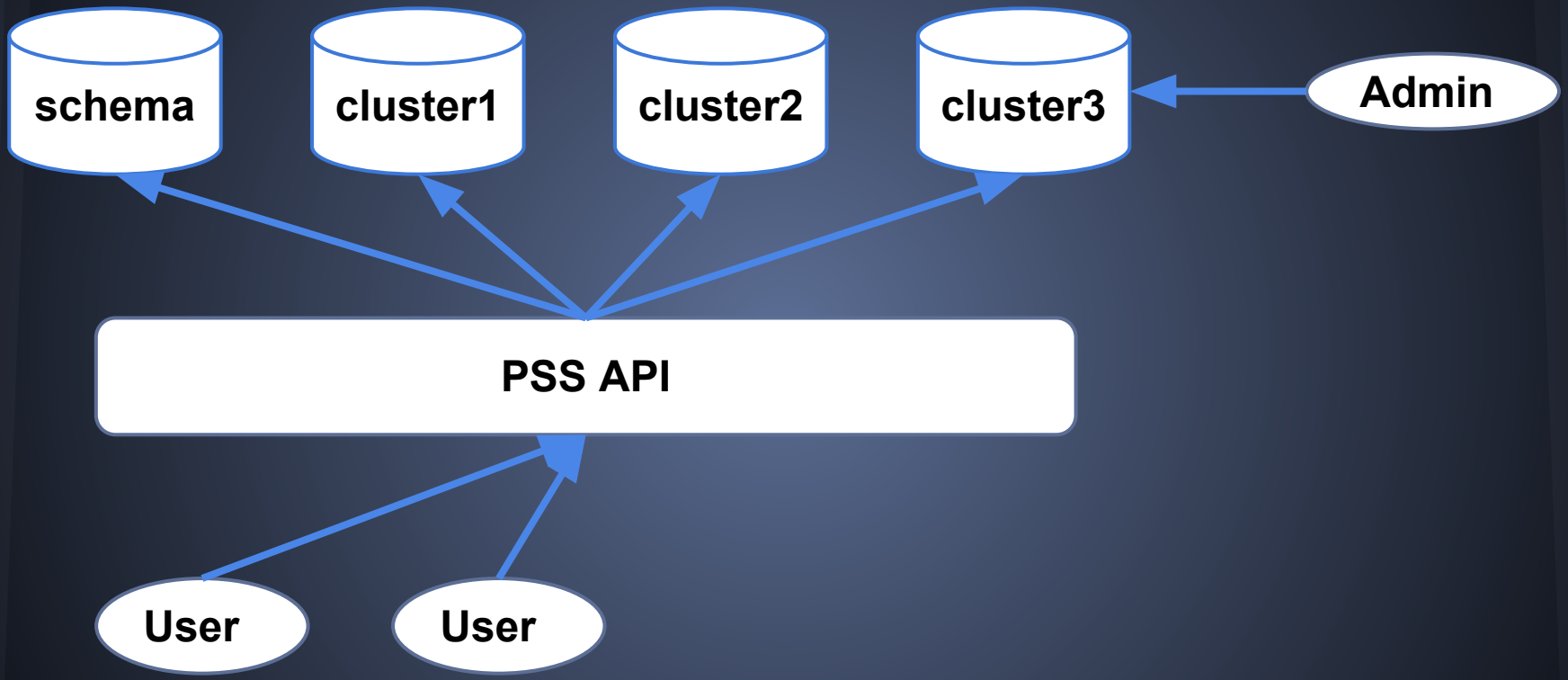
应用场景

- 结构化存储平台PSS(Personal Structural data Storage)
- 典型应用
  - 图片exif信息
  - 通讯录
  - 设备同步
- 数据量：百亿 - 千亿
- 性能：在线业务，平均响应时间要求10ms内
- 多索引需求
  - 包含数组索引

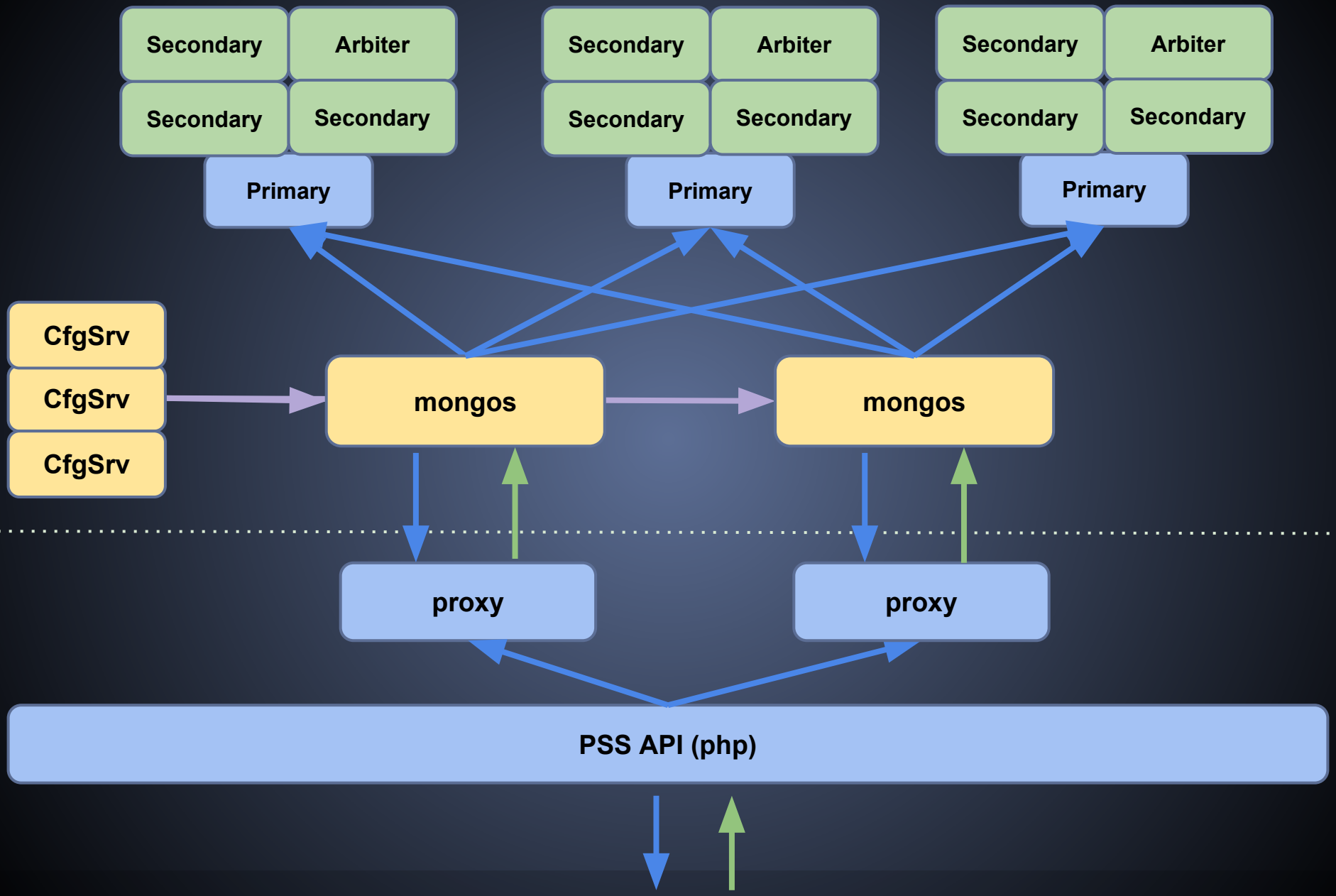
特点

- 规模大
  - 单表数据量数百亿(存储百T级别)
  - 容易触发mongo的一些性能隐患/隐藏BUG
- 数据冷
  - 需要大量机器，期望压缩功能.
- 用法固定
  - 都是基于个人的数据，可以很好按照uid sharding
  - 查询模式相对简单

架构



1. 提供API，隐藏后端多个集群的细节，防止对mongo的滥用
2. 便于在API层统计,封禁


Secondary
Arbiter
Secondary
Secondary
Primary
Secondary
Arbiter
Secondary
Secondary
Primary
Secondary
Arbiter
Secondary
Secondary
Primary
CfgSrv
CfgSrv
CfgSrv
mongos
mongos
proxy
proxy
PSS API (php)

Mongoproxy

- 解决问题
  - php-driver 连接池在线程间不共享
  - 50 machines * 500 threads = 25000.
- 功能
  - 对后端使用长连接, 前端使用短连接
  - 相当于跨进程连接池
  - 读写分离(连replset时)

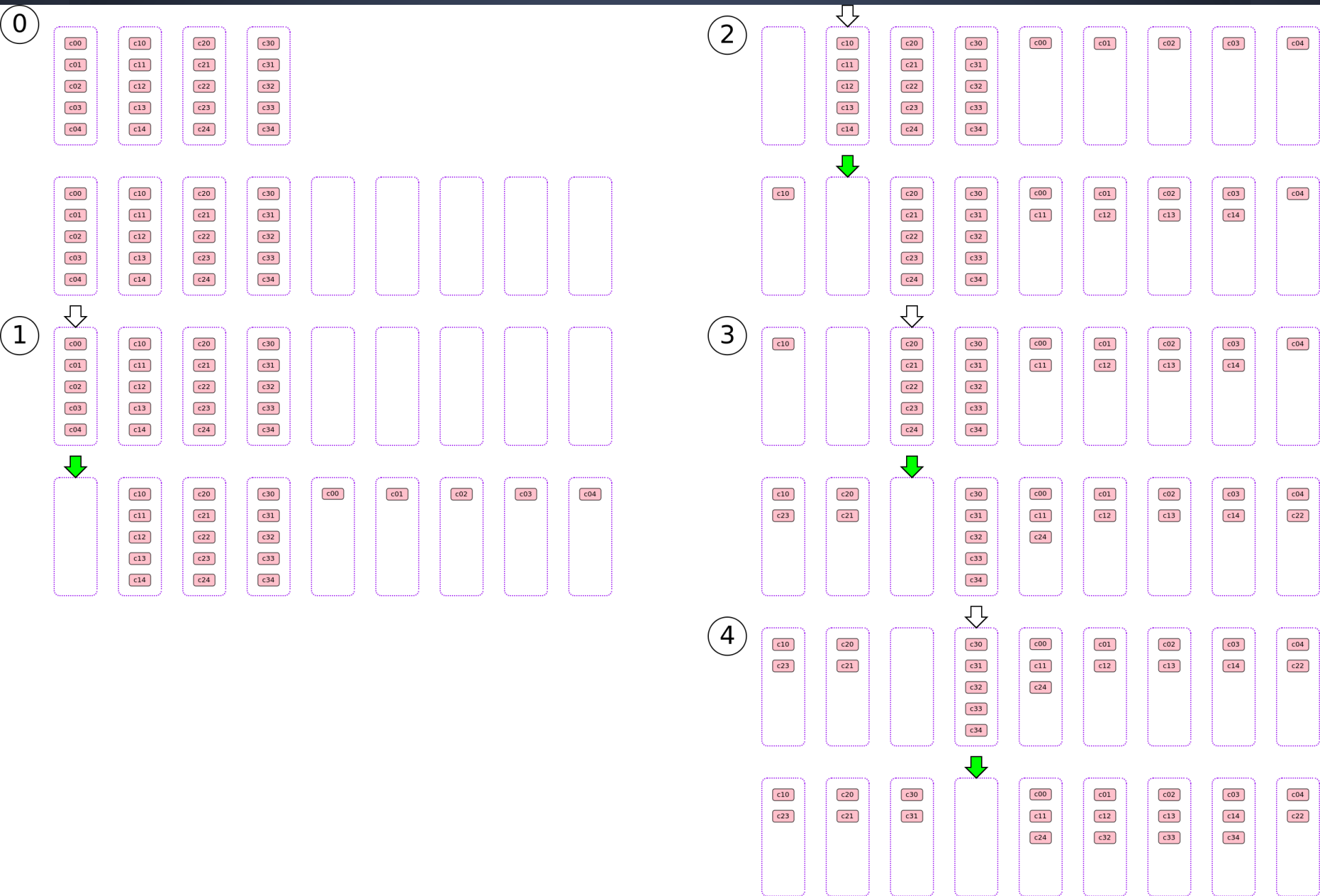
Balance

- 关闭auto-balance
- 使用pre-sharding

```
db.runCommand( { split : "testDb.testColl" , middle : { uid: 100000000} } );
db.runCommand( { moveChunk : "testDb.testColl" , find : {uid: 100000001} , to : 's0-set0'} )
```

扩容

- v0: 依靠auto-balance
- v1: noCleanup+外部工具删除
- v2: noCleanup+自有的balance调度
- v3: 彻底替换moveChunk, 实现并发


0
c00 c01 c02 c03 c04
c10 c11 c12 c13 c14
c20 c21 c22 c23 c24
c30 c31 c32 c33 c34
1
c00 c01 c02 c03 c04
c10 c11 c12 c13 c14
c20 c21 c22 c23 c24
c30 c31 c32 c33 c34
c00 c01 c02 c03 c04
c10 c11 c12 c13 c14
c20 c21 c22 c23 c24
c30 c31 c32 c33 c34
2
c10 c11 c12 c13 c14
c20 c21 c22 c23 c24
c30 c31 c32 c33 c34
c00 c01 c02 c03 c04
c10
c20 c21 c22 c23 c24
c30 c31 c32 c33 c34
c00 c11
c01 c12
c02 c13
c03 c14
c04
3
c10
c20 c21 c22 c23 c24
c30 c31 c32 c33 c34
c00 c11
c01 c12
c02 c13
c03 c14
c04
c10 c23
c20 c21
c30 c31 c32 c33 c34
c00 c11 c24
c01 c12
c02 c13
c03 c14
c04 c22
4
c10 c23
c20 c21
c30 c31 c32 c33 c34
c00 c11 c24
c01 c12
c02 c13
c03 c14
c04 c22
c10 c23
c20 c21
c30 c31
c00 c11 c24
c01 c12 c32
c02 c13 c33
c03 c14 c34
c04 c22

经验

建议

- HDD => Mem/SSD
- pre-sharding (和hbase类似)
- 监控主要关注 qr
- 合理选择shardkey
- 不能直接在线加索引，需要通过轮转方式做

- 性能：不要盲目相信别人的测试，适合自己的才是最好的

索引

- 参考MySQL建索引的方法
  - 前缀规则
  - 区分度大的字段放前面
- 注意索引的空间占用

慢查询

避免这些查询模式:

- count (特别是需要扫描大量条目时)

  db.testColl.find({xxx: testValue}).count()
- large-skip (特别是sharding下)

  db.testColl.find({xxx: testValue}).skip(100000).limit(10)
- 尽量减少 $nin/$regex
- Map Reduce
- 不带shardkey的查询

一句话: 避免扫描大量记录

发现慢查询: 杀

一些工具

- mongomigrate 集群间迁移
- 监控工具 mongoviz
- Replset 轮转加索引
- 加Replset/重做secondary工具
- 杀慢查询脚本

Thanks

Q&A