Mongo @ 百度云

杨林

火龙果•整理
uml.org.cn

提纲

- Mongo的特点
- 百度云与Mongo
- 遇到的问题
- 一些经验

火龙果•整理
uml.org.cn

Mongo特点

- 丰富的数据类型
  - 嵌套结构
  - 数组
- 丰富的索引/查询模式支持
  - 地理索引/数组索引/全文索引/MapReduce
- Schemaless 灵活，易上手
- auto sharding
- 较高的性能

火龙果•整理
uml.org.cn

百度云与Mongo

火龙果•整理
uml.org.cn

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

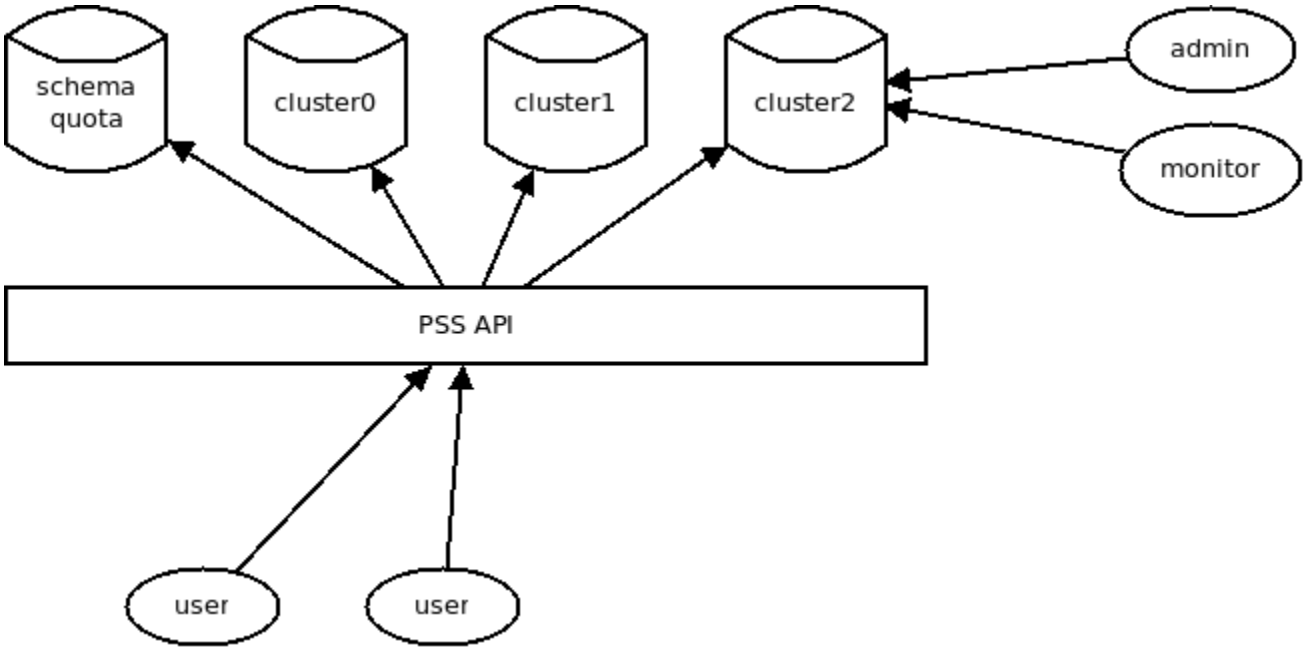
火龙果•整理
uml.org.cn

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

火龙果•整理
uml.org.cn

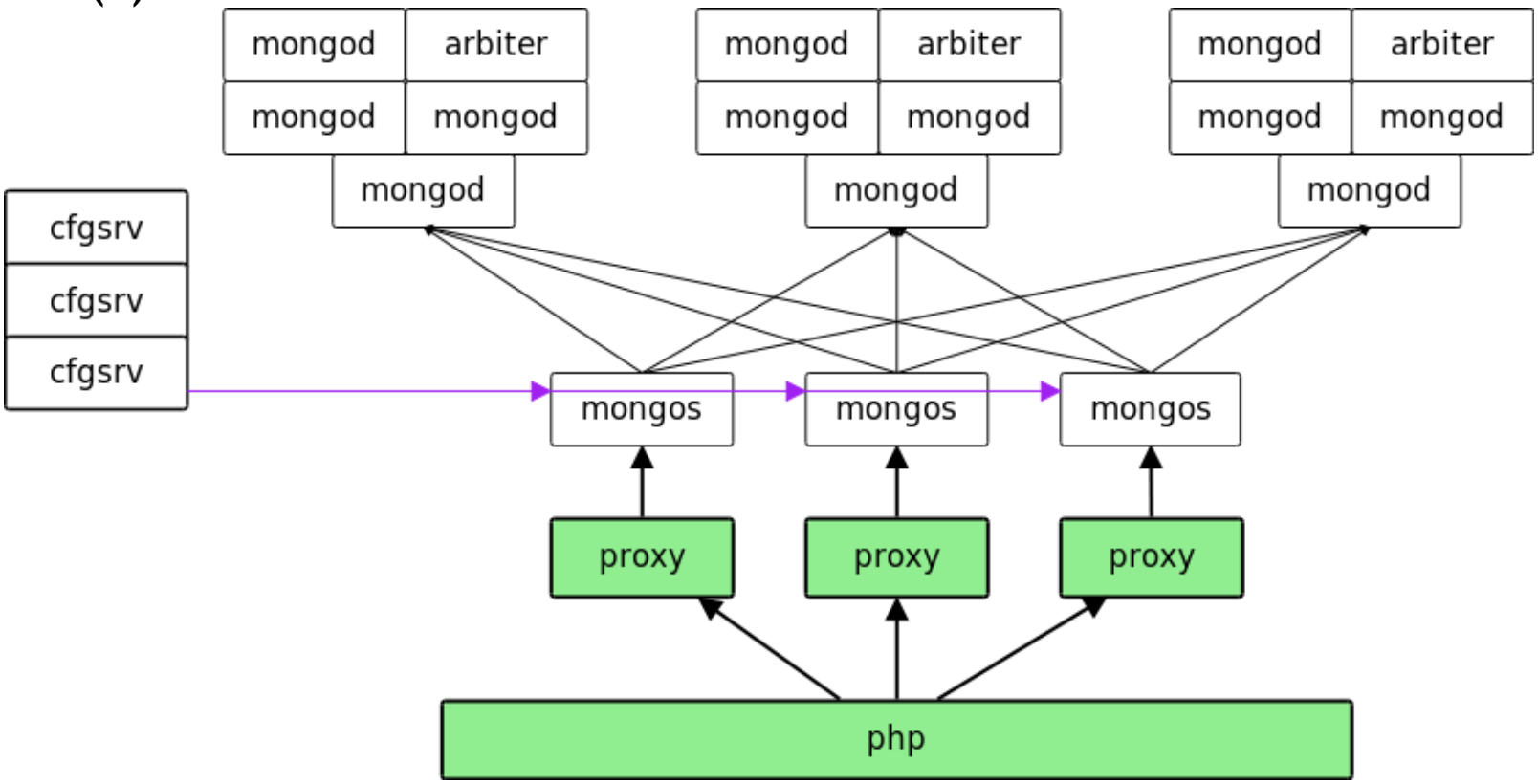
架构



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，分析和封禁
3. 系统管理员直接操作Mongo

火龙果•整理
uml.org.cn

架构(2)



火龙果•整理
uml.org.cn

遇到的问题

火龙果•整理
uml.org.cn

连接数爆炸

```
$ mongostat -h 10.50.18.33 --port 7111 -u __system -p xxxx -n 100
connected to: 10.50.18.33:7111
... faults locked db miss %     qr|qw   ar|aw  netIn netOut  conn ...
...     28 xxx:53.7%      0       3|0     0|2     1m   999k  3031 ...
...      8 xxx:48.0%      0      19|0     0|1     1m   886k  3038 ...
...     16 xxx:52.2%      0       0|0     0|1     1m     1m  3055 ...
...      6 xxx:45.1%      0       0|0     0|0     1m   812k  3061 ...
...     15 xxx:52.6%      0       0|0     0|0     1m   973k  3061 ...
...     13 xxx:44.3%      0       6|0     0|1   970k     1m  3062 ...
```

conn一直上涨，达到配置的极限值，开始拒绝服务.

火龙果•整理
uml.org.cn

原因

- 慢查询
- 客户端太多
  - `php-driver` 连接池在线程间不共享
  - 50 machines * 500 threads = 25000.

火龙果•整理
uml.org.cn

解决

- 通过监控及早发现
- 针对慢查询，用 `killOp()` 杀掉
- **针对客户端过多的情况，增加一层proxy**
  - 对后端使用长连接，前端使用短连接
  - 相当于跨进程连接池

火龙果•整理
uml.org.cn

慢查询

- 导致conn上升
- 导致qr上升
- io打满/lock高

火龙果•整理
uml.org.cn

避免这些查询模式

- 少用 `count()`

```
db.testColl.find({xxx: testValue}).count()
```

- 避免 `large-skip` (特别是sharding)

```
db.testColl.find({xxx: testValue}).skip(100000).limit(10)
```

- 不用 `$nin`

火龙果•整理
uml.org.cn

杀掉慢查询

命令:

```
db.currentOp()
db.killOp()
```

杀慢查询要注意:

```
op ['op'] == 'query' &&
!op.startswith('conn') &&
!splitVector &&
!moveChunk
```

火龙果•整理
uml.org.cn

建索引/repair

- 耗时非常长，单mongod上3亿条数据，对三个字段建索引：

```
db.testColl.ensureIndex( {"_uid": 1, "xxxx": 1, "md5": -1}, { background: true } )
```

  - 类型分别为 int, int, string(md5)，需要时间18 小时
  - 如果类型为int, int, int，需要时间会比较少
- io/cpu压力影响正常业务
- 阻塞主从同步
  - primary上索引建完写oplog 之后
  - 索引通过oplog同步到Secondary
  - Secondary 执行该oplog时，会阻塞住后续的Oplog (background 选项在Secondary会忽略)
- repair 很慢，还需要2倍磁盘空间

火龙果•整理
uml.org.cn

解决

轮转方式建索引

- 下线一台从库，做好索引后 挂上追oplog，然后做下一台机器
- 也可以做完一个机器后，拷贝数据文件到其它机器.
- 先做从库，再做主库.

利用迁移做索引

- 在数据迁移到另一个集群的过程中，写数据之前，建索引

火龙果•整理
uml.org.cn

auto balancer

- 小数据量时没问题
- 一旦开始balancer, 线上服务就开始不稳定(读写变慢, 超时, conn, qr上涨)
- 调整balancer窗口

```
db.settings.update ( { "_id" : "balancer" }, { $set : { "activeWindow" : { start : "2:00
```

不能从根本上解决问题, 要不了几天, 窗口被缩小到 0 .

火龙果•整理
uml.org.cn

原因

- auto ballance 是使用 moveChunk 实现的
- **moveChunk 删除阶段的问题**
  - chunk迁移所操作的数据，很多不是热数据，迁移和删除过程中导致需要频繁读取磁盘，造成IO负载高
  - 删除阶段维持时间长，比如move 10w条数据后，用1000/s的速度删除，需要维持100s
- moveChunk 不能并行.

火龙果•整理
uml.org.cn

解决

- moveChunk后的删除阶段限速(能缓解问题)
- pre-sharding:

```
db.runCommand( { split : "testDb.testColl" , middle : { uid: 100000000} } );
db.runCommand( { split : "testDb.testColl" , middle : { uid: 200000000} } );
...

db.runCommand( { moveChunk : "testDb.testColl" , find : {uid: 100000001} , to : 's0-set0
db.runCommand( { moveChunk : "testDb.testColl" , find : {uid: 200000001} , to : 's0-set1
```

然后可以关闭auto ballance (除非需要扩容)

火龙果•整理
uml.org.cn

扩容

- 磁盘用满70%
- 单mongod读写压力到达极限

火龙果•整理
uml.org.cn

方案(1)

1. **直接增加replset, 交给auto balancer**
    - 遇到auto balancer的问题, 每次moveChunk就影响服务
    - 非常慢.
    - 2-4个shard时可试一试.
2. **moveChunk中去掉删除数据阶段(noCleanup), 通过外部方式来删除**
    - 外部删除可以很好的控制压力, 保证不影响线上服务(一般200/s)
    - 可以并行删
    - 依然很慢

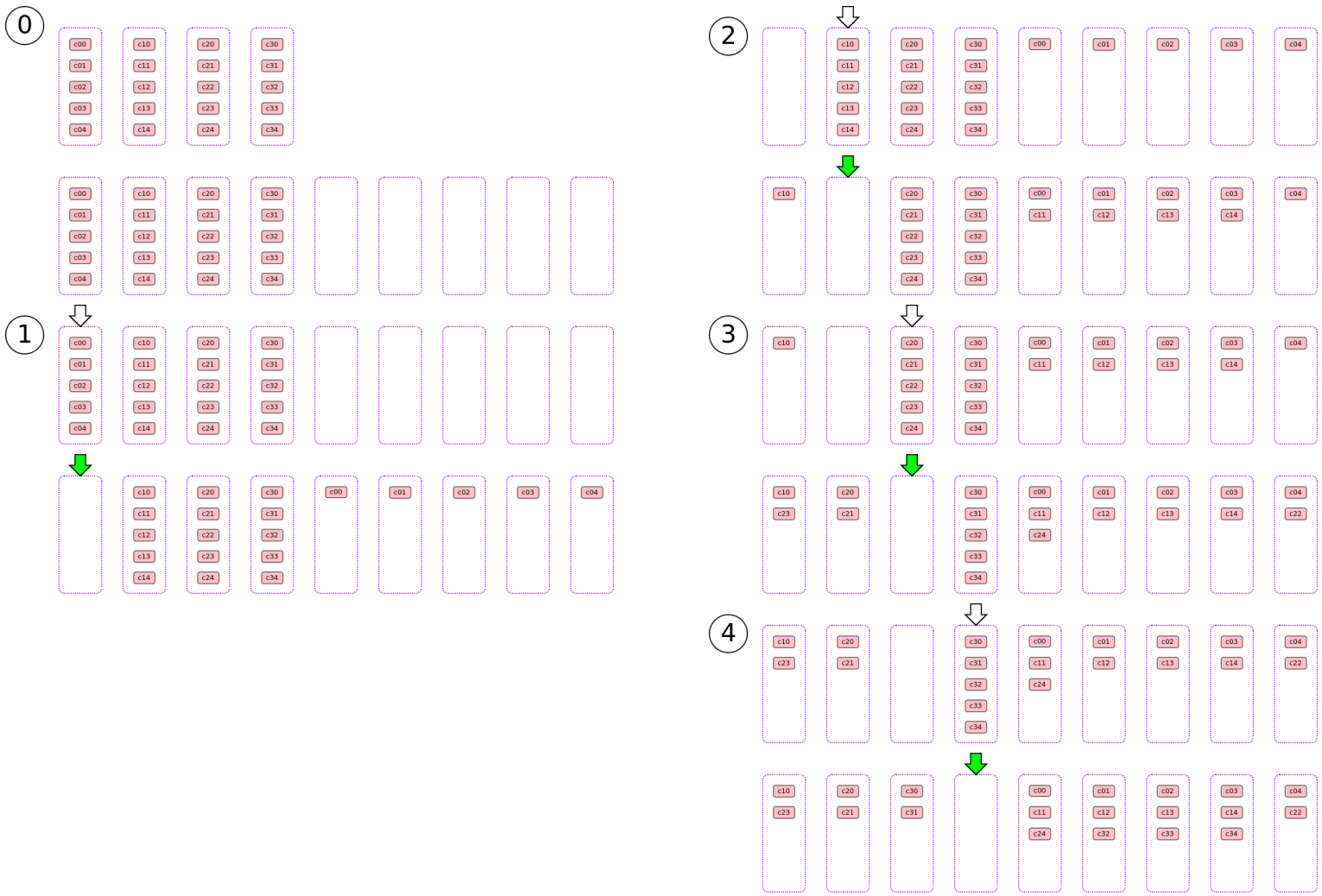
火龙果•整理
uml.org.cn

方案(2)

- 扩容慢主要是删数据阶段慢，能不能避免删数据?
- 把一个replset上所有chunk都move走之后，直接把这个replset上的db drop掉.
- 优点
  - 较快(没有删除阶段)
  - 全自动化
  - 可以取代repair操作.
- 问题:
  - moveChunk 可能因为chunk过大而失败

火龙果•整理
uml.org.cn

方案(2)



火龙果•整理
uml.org.cn

方案(3)

- 原生的moveChunk一次只能move一个chunk，不能并发，想办法提高moveChunk的速度
- 不使用原生的**moveChunk**
  1. 通过外部工具导出数据的方法迁移chunk到新shard(可并发)
  2. 追oplog
  3. 最后一次性修改configsrv中的路由表

火龙果•整理
uml.org.cn

迁移

- 扩容来不及
- 业务隔离需求
- 升级mongodb到新的版本，不支持平滑升级
- 新旧架构不一致

火龙果•整理
uml.org.cn

方案

步骤:

1. 迁移基准数据
2. 追oplog
3. api上切换读写到新集群.

mongomigrate(部分开源):

- 迁移时实现pre-sharding
- 并发导出导入基准数据/并发追oplog
- 并发追oplog需要在mongod 的delete/update 的oplog中增加shardkey字段, 据此分桶实现并发.
- 每天可以迁移5-10亿条数据

火龙果•整理
uml.org.cn

经验&工具

火龙果•整理
uml.org.cn

建议

- **HDD => SSD**
- 保证节点间时钟同步 **<30s**
- 一切地方用IP, 不要用hostname (rs.add/addshard等)
- **pre-sharding** (和hbase类似)
- 合理选择**shardkey**
  - 用户数据: 按照uid维度 sharding
  - 日志数据: 每天一个库.
  - _id/md5不是好选择(不均匀, 不便于pre-sharding)
- 如果来不及扩容, 可以考虑删一些索引

火龙果•整理
uml.org.cn

索引选择

- 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

火龙果•整理
uml.org.cn

mongostat监控

```
$ mongostat -h 10.50.18.33 --port 7111 -u __system -p xxxx -n 100
connected to: 10.50.18.33:7111
... faults locked db miss %     qr|qw   ar|aw  netIn netOut  conn ...
...     28 xxx:53.7%      0       3|0     0|2     1m   999k  3061 ...
...      8 xxx:48.0%      0      19|0     0|1     1m   886k  3061 ...
...     16 xxx:52.2%      0       0|0     0|1     1m     1m  3061 ...
...      6 xxx:45.1%      0       0|0     0|0     1m   812k  3061 ...
...     15 xxx:52.6%      0       0|0     0|0     1m   973k  3061 ...
...     13 xxx:44.3%      0       6|0     0|1   970k     1m  3062 ...
```

- 重点关注:
  - **conn** : 有多少连接数, 最直观的表现
  - **qr** : 有多少读请求在排队 (连续几秒大于 **10** => 必然有问题)
  - *locked* : 锁争用情况, ssd上, 该项关系不大

火龙果•整理
uml.org.cn

慢查询分析

mongod日志

```
Sun Nov 10 09:28:58 [conn3156482] command admin.$cmd command: { splitVector: "testDb.testCol

Sun Nov 10 11:50:37 [conn3141108] update testDb.testColl query: { $and: [ { $and: [ { $and:
```

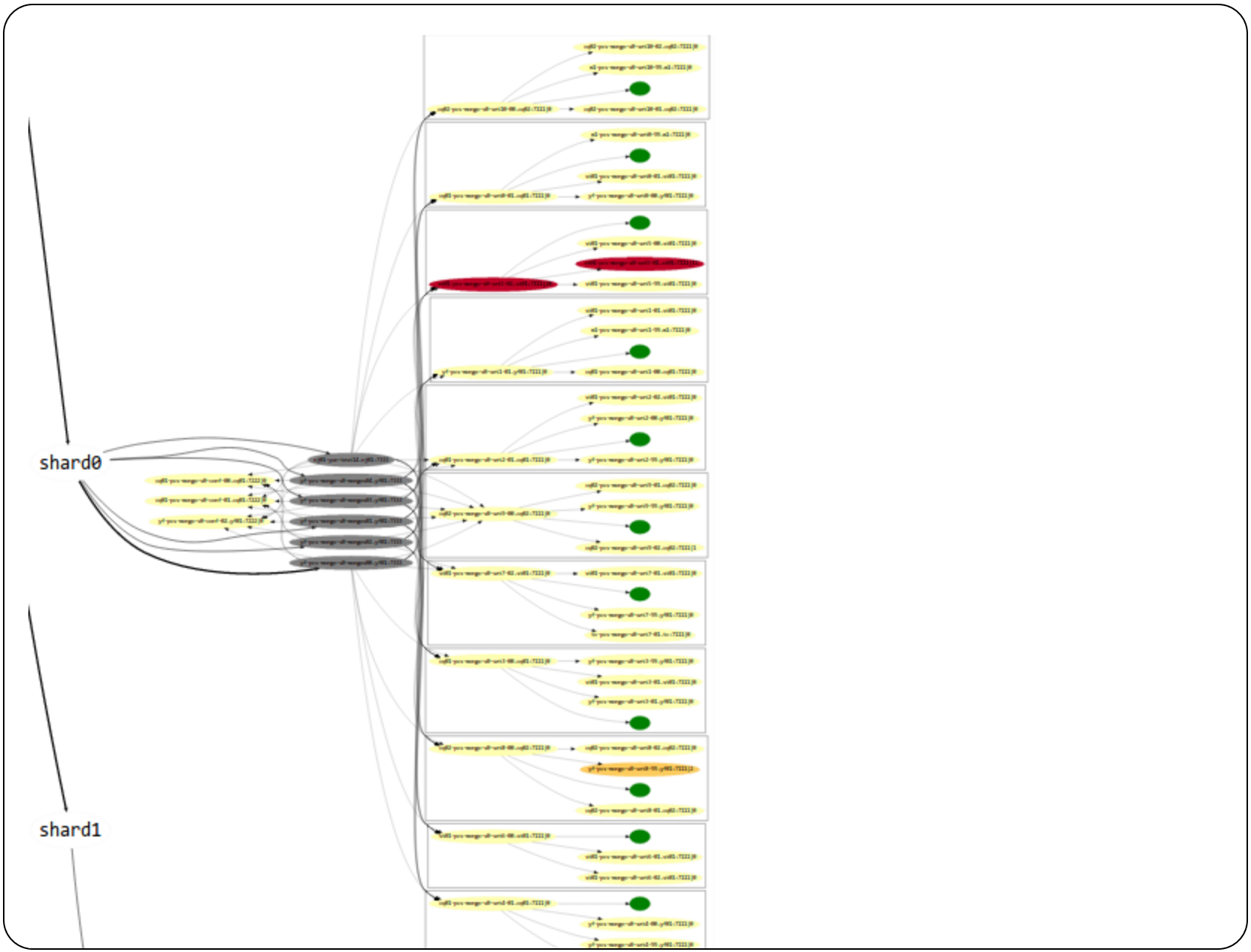
工具:

- **dex** (https://github.com/mongolab/dex)
  - 分析慢查询的模式

火龙果•整理
uml.org.cn

监控工具 mongoviz

火龙果•整理
uml.org.cn



火龙果•整理
uml.org.cn

其它工具

- 部署工具
- Replset 轮转加索引
- Replset 增加/重做secondary工具
- 杀慢查询脚本
- chunk数/磁盘空间分析工具.
- Web化管理平台
- mongoproxy
- mongomigrate
- draining扩容工具

火龙果•整理
uml.org.cn

thanks

火龙果•整理
uml.org.cn