網易 NETEASE
www·163·com

MongoDB 中文社区

IT大咖说 知识分享平台

网易MongoDB云服务设计实践

温正湖

大数据平台部 云数据库项目组

2017年7月16日

MongoDB目录

- MongoDB简介
- 云服务实践

MongoDB简介

- 概述
- 复制集 – replica set
- 分片 - sharding

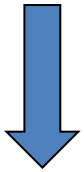
MongoDB概述

Database -- Database

Collection -- Table

Document -- Record

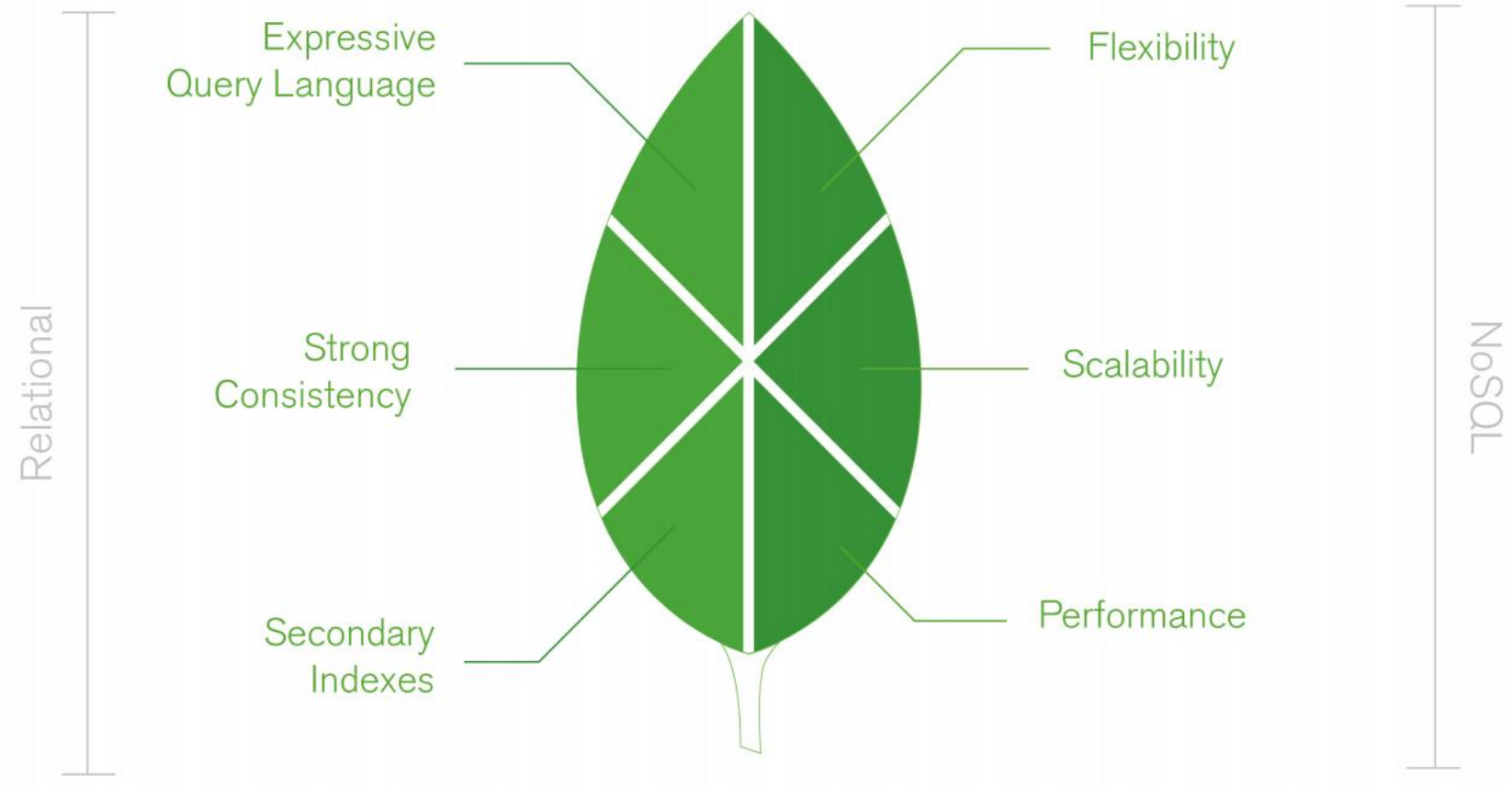
Schema free!
Not schema-less!



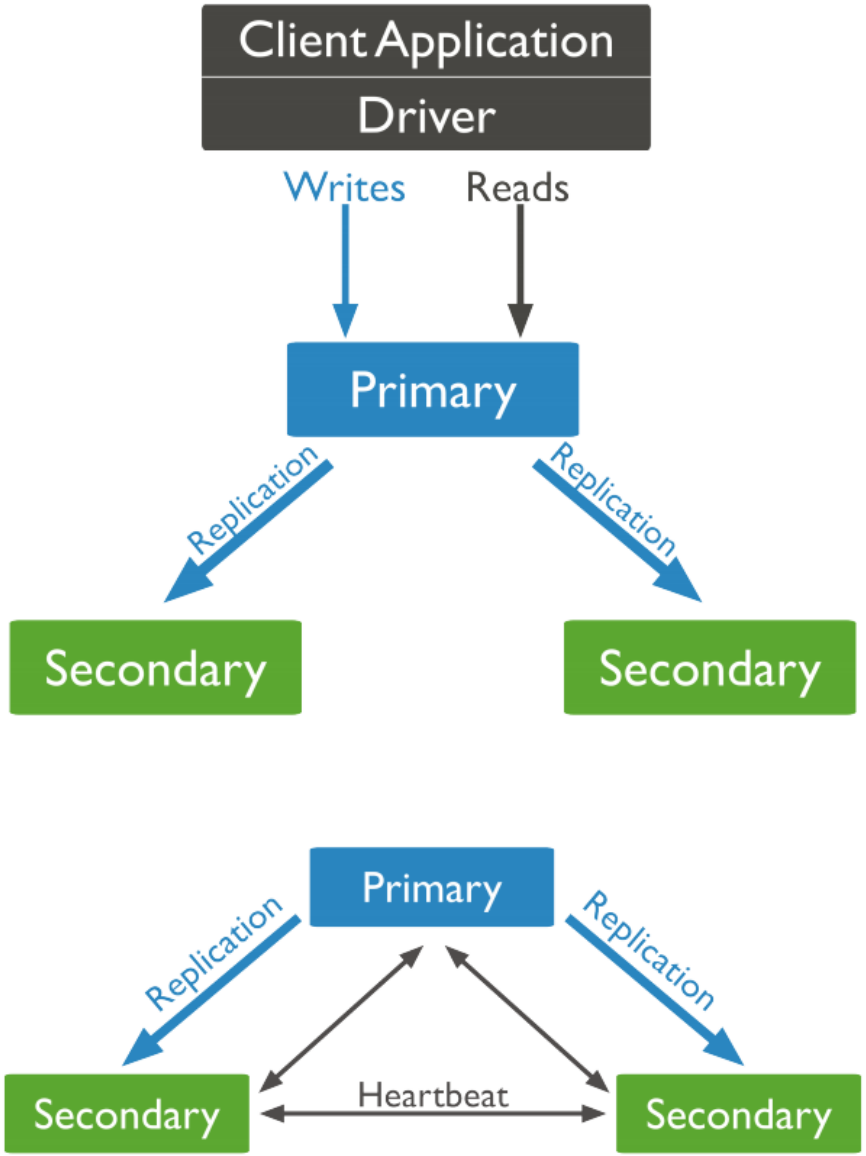
```
{
  name: "sue",          ← field: value
  age: 26,              ← field: value
  status: "A",          ← field: value
  groups: [ "news", "sports" ]   ← field: value
}
```

MongoDB概述

- 集大成者



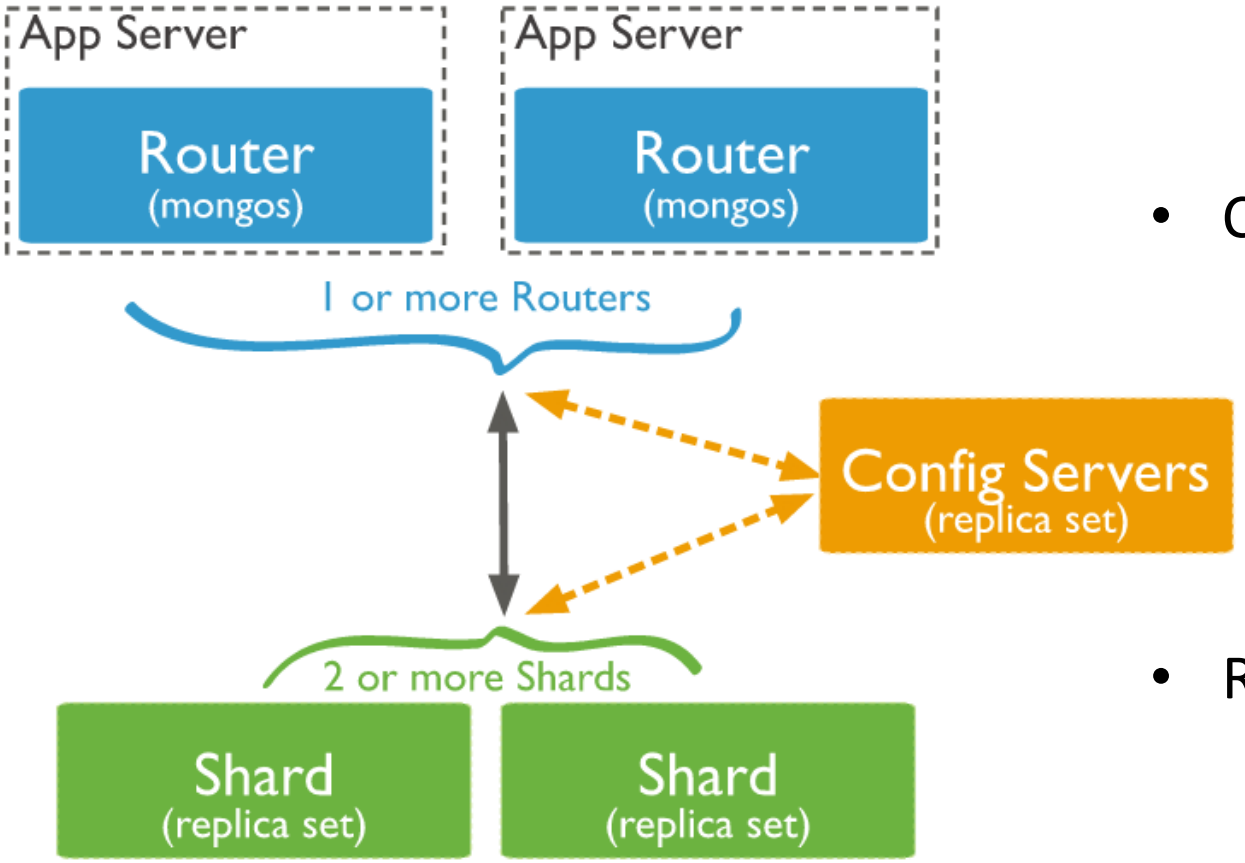
Replica set



复制集架构：

- One Primary
  - Write
  - Read (default)
- Multi Secondary
  - Replication（async）
  - Read(set by query)
- Heartbeat

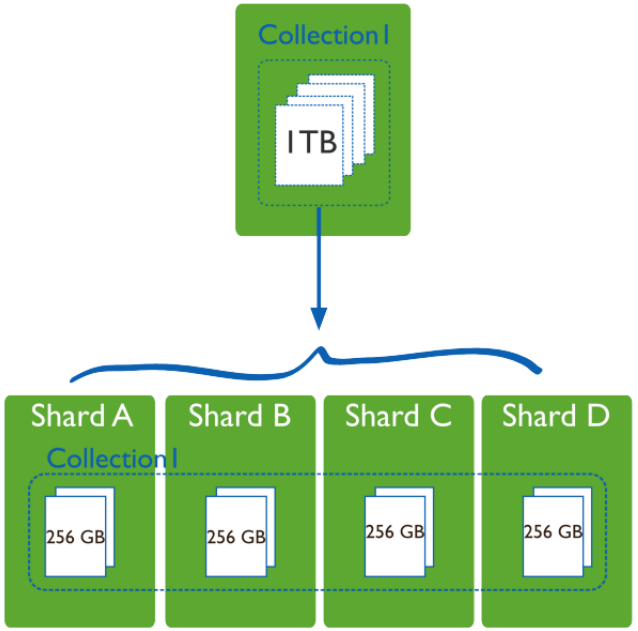
Sharding

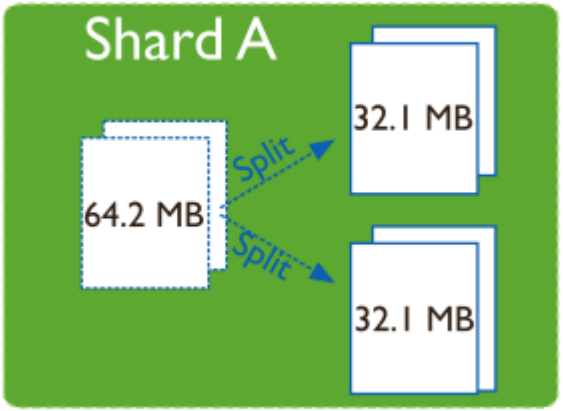


- Shards
  - Subset of data
  - Instance or Replica set
- Config server
  - Balancer： migrate
  - Metadata and settings
    - range⟺chunk⟺shard
    - secondaryThrottle
    - waitForDelete
- Router server
  - query router
  - cluster interface
  - chunk split

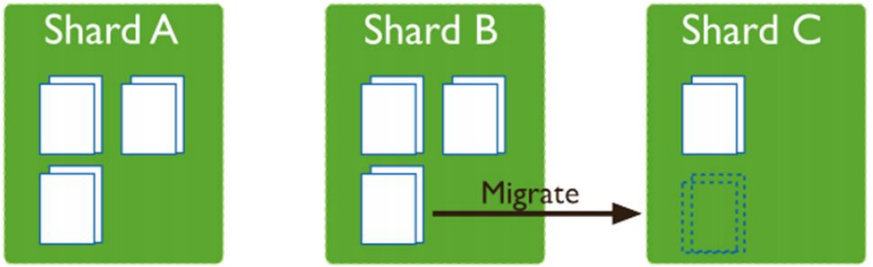
Sharding

- chunk
  - Collection level





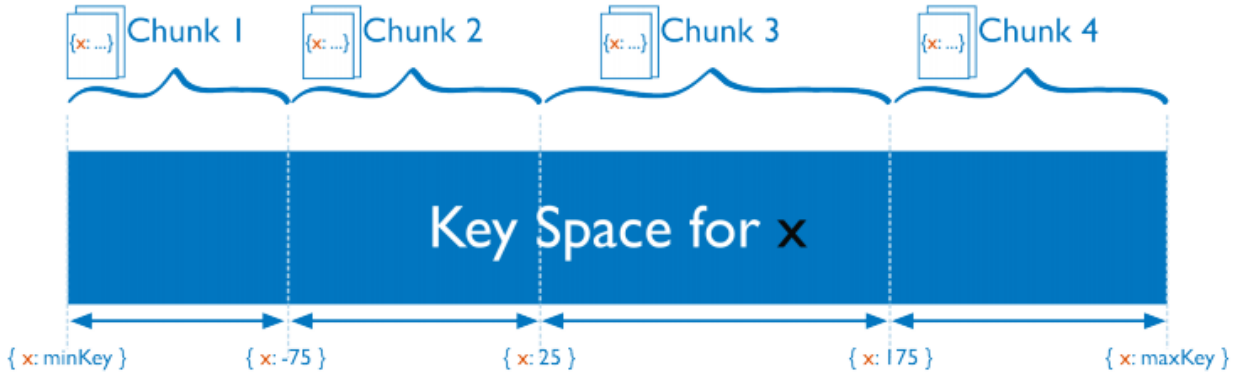
- Splitting
  - Only meta-data change



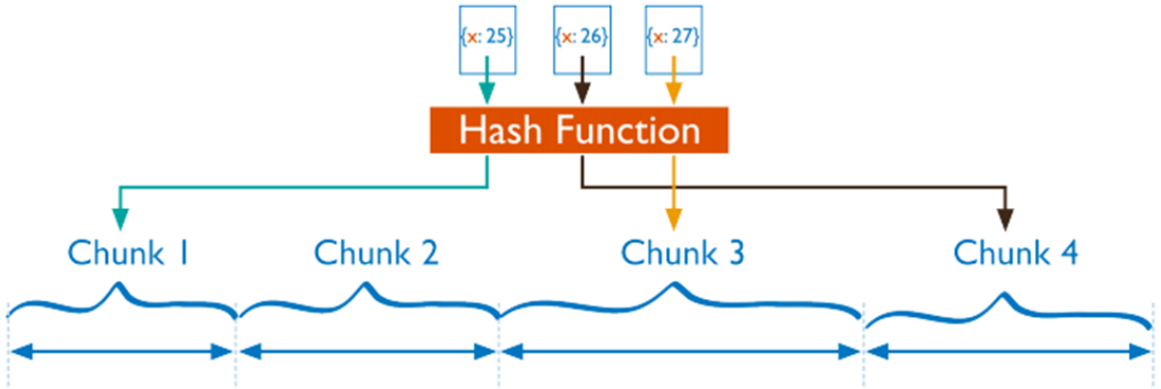
- Chunk migration

Sharding

- Range based



- Hash based

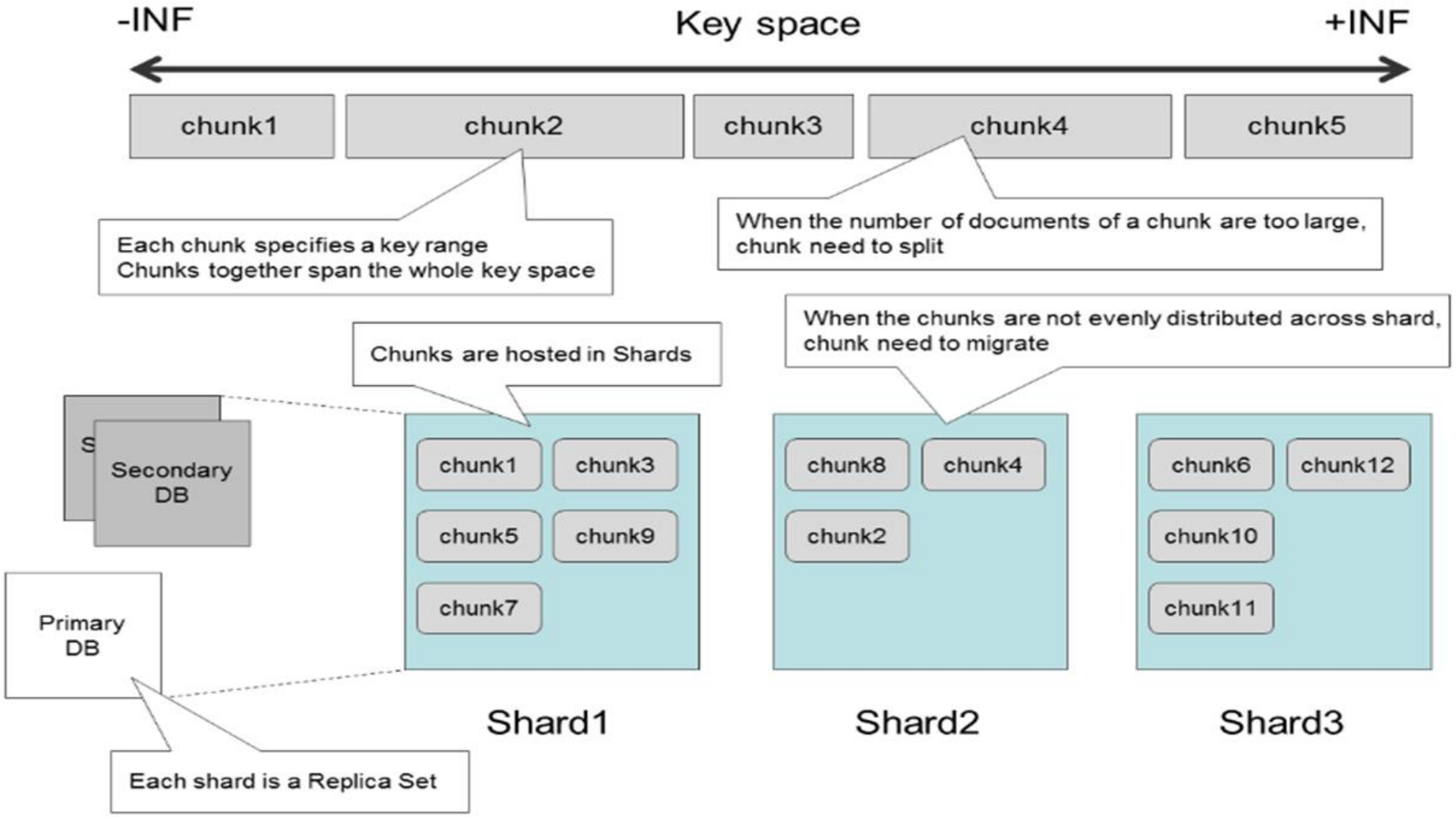


Shard key

- Large Cardinality
- Low Frequency
- Non-Monotonically changing

Sharding

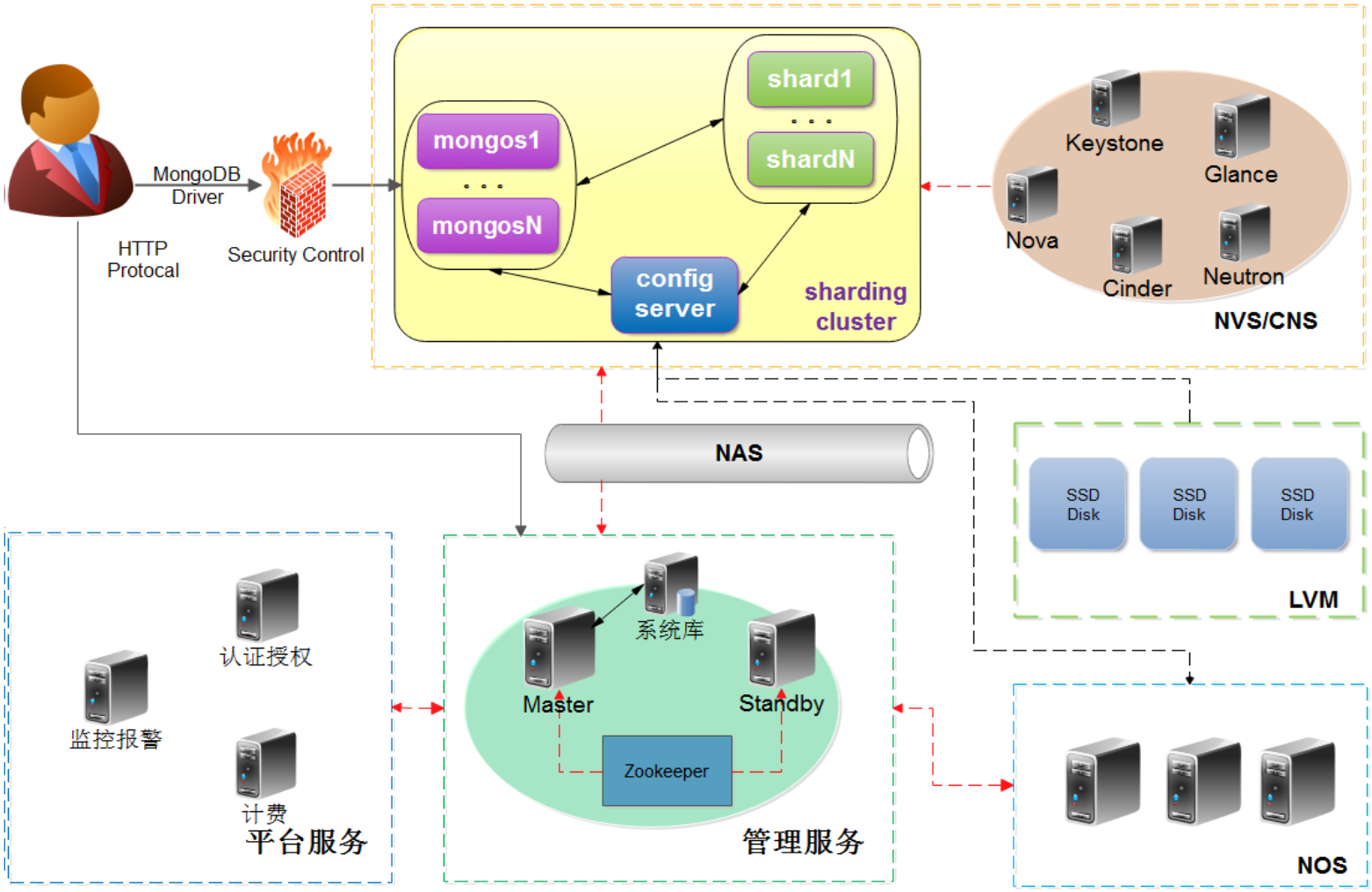
- summary



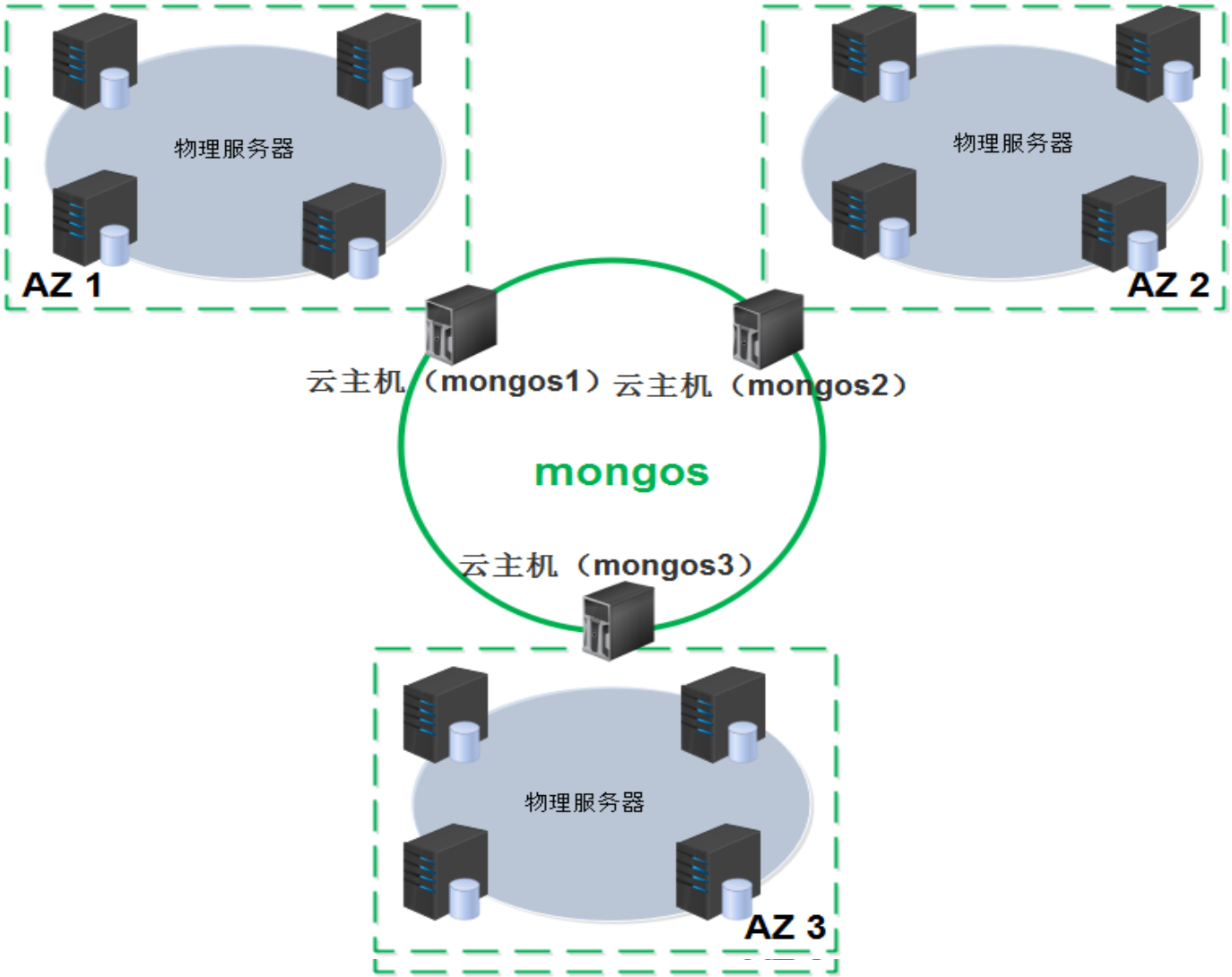
云服务实践

- 整体框架
- 高可用设计
- 数据备份
- 升级与扩容
- 故障修复
- 应用读写规范

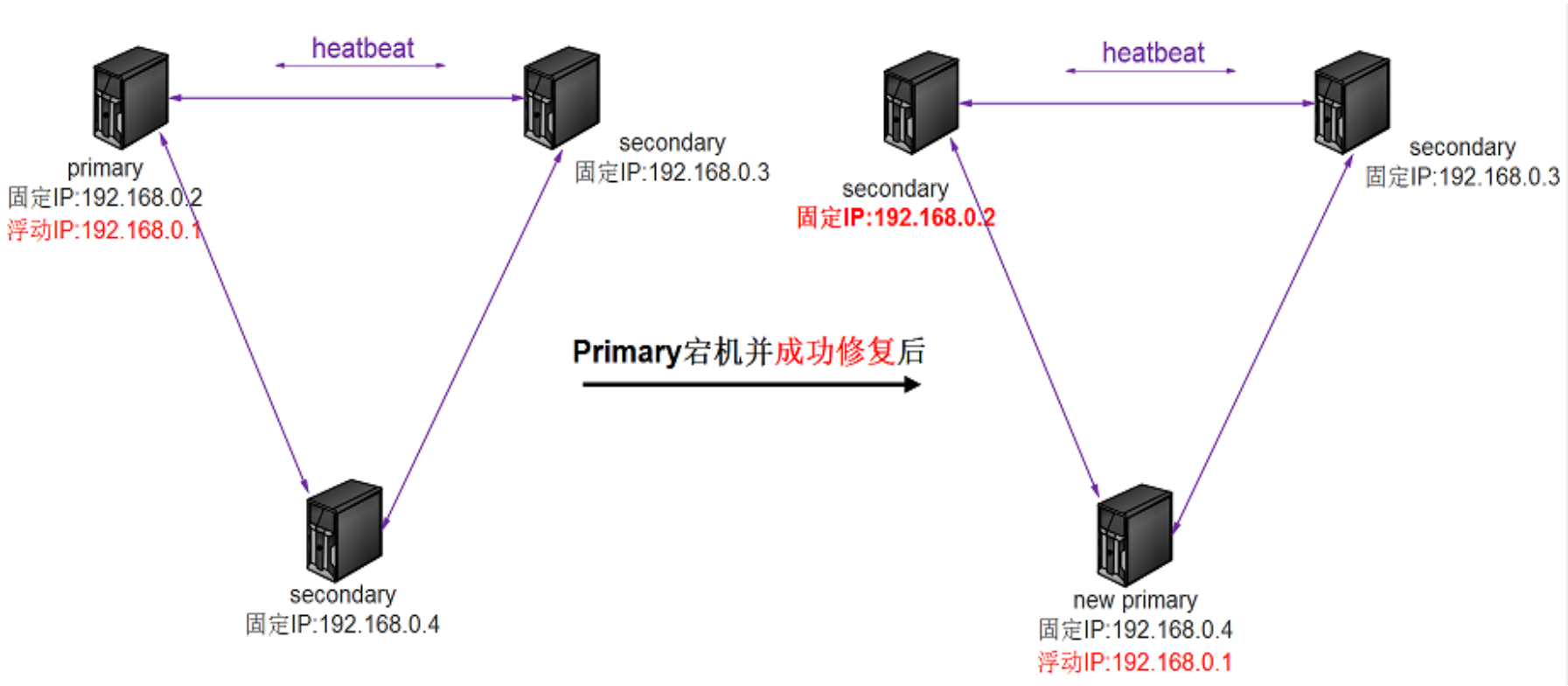
- 整体框架



- 高可用 – AZ设计

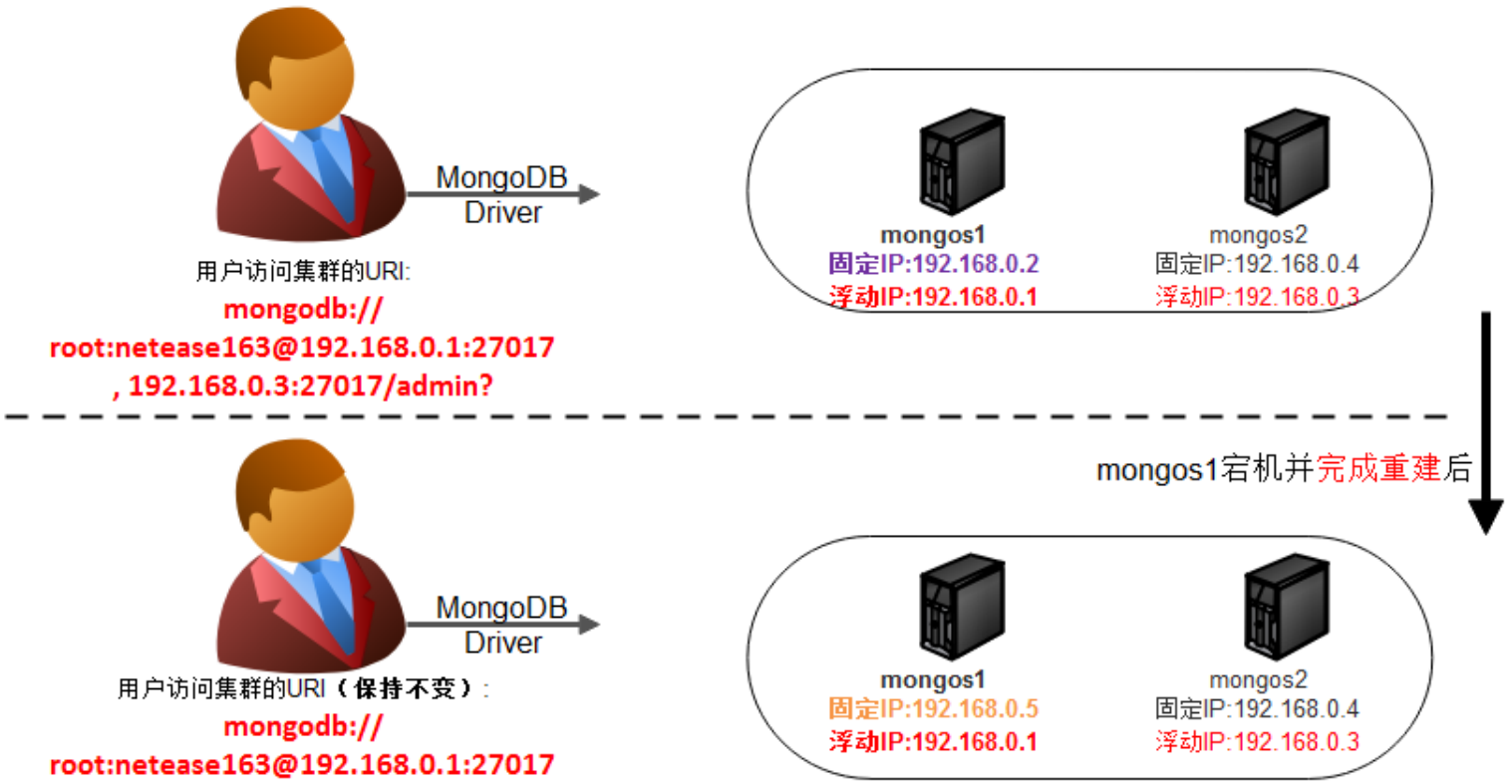


- 高可用 – 用户访问（网络）
  - 复制集

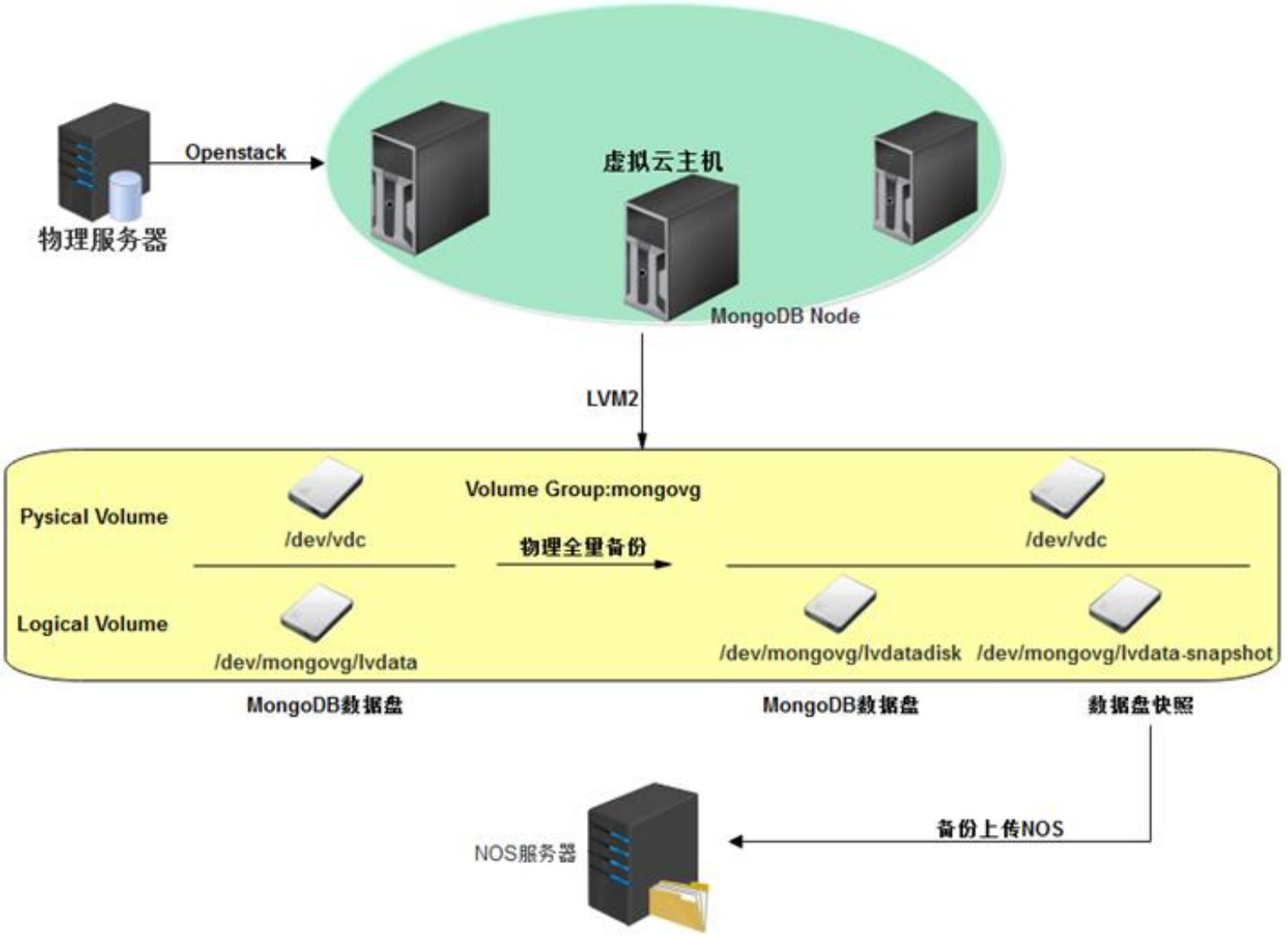


mongodb://root:****@192.168.0.1:27021/admin?replicaSet=mg163-582

- 高可用 – 用户访问（网络）
  - 分片集群



- 数据备份 – lvm snapshot



物理 vs 逻辑

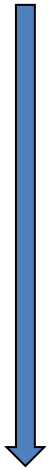
- **数据备份 – 优化**
  - 资源竞争
    - 在Secondary备份
    - （可选）限制QOS，防止写爆snapshot
  - 大小优化
    - 删除local库
    - 删除oplog.rs
    - 删除存储引擎预分配日志

```
data/journal# ls -lh

 100M May  7 16:41 WiredTigerLog.0000000004
 100M May  7 15:03 WiredTigerPreplog.0000000002
 100M May  7 15:46 WiredTigerPreplog.0000000005
data/journal#
```

- **数据备份 – 分片集群场景**
  - 分片集群数据一致性
    - 用户读写设置 Primary?
    - 各Shard主从延时

物理备份优势明显：能够事先知道备份的具体时间点



T1a：写了ShardA，Data1；

T1b：写了ShardB，Data2；

T1c：写了ShardC，Data3；Data2复制到ShardB的Secondary；

T2：用户/系统发起备份操作；Data3复制集到ShardC的Secondary；

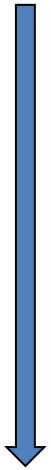
T3：分别对ShardA/B/C、Config Server的Secondary做LVM Snapshot，开始拷数据；

T4：Data1复制到ShardA的Secondary；

T2时间点的备份没有之前T1a的数据！！

- **数据备份 – 分片集群场景**
  - 分片集群数据一致性
    - 用户读写设置 Primary?
    - 各Shard主从延时

物理备份优势明显：能够事先知道备份的具体时间点



T1a：写了ShardA，Data1；

T1b：写了ShardB，Data2；

T1c：写了ShardC，Data3；Data2复制到ShardB的Secondary；

T2：用户/系统发起备份操作；Data3复制集到ShardC的Secondary；

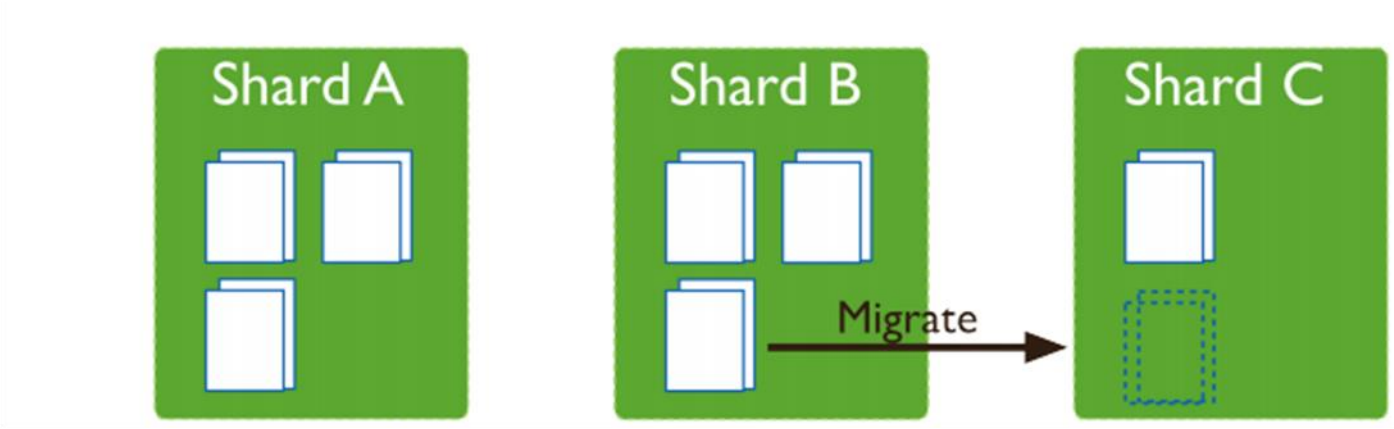
T3：分别确认ShardA/B/C、Config Server的Secondary oplog中已经有T2时间点的数据；之后再做LVM Snapshot，开始拷数据；所以ShardB/C、Config Server可以Snapshot；

T4：Data1复制到ShardA的Secondary；

T5：ShardA的Secondary做Snapshot，开始拷数据；

T6：完成一致性备份；

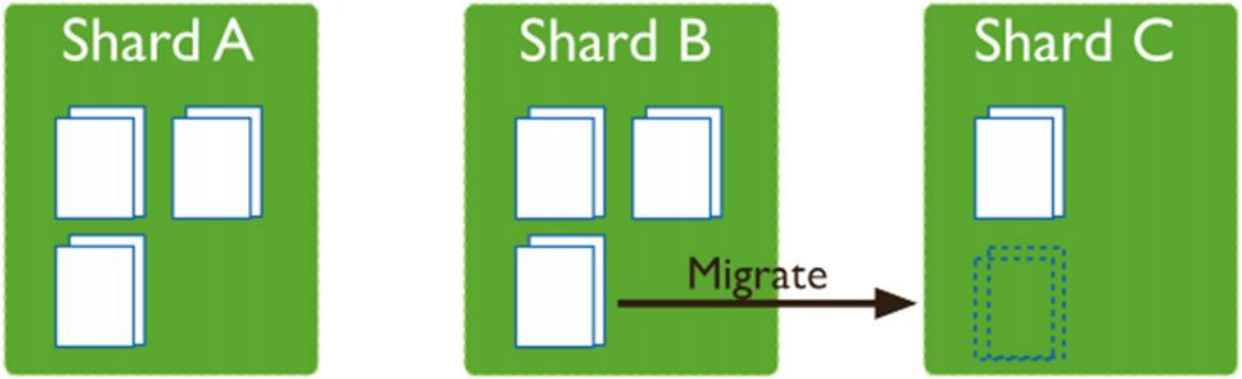
- **数据备份 – 分片集群场景**
  - 分片集群数据一致性



- _secondaryThrottle
  - 迁移时，chunk数据默认仅写到目标shard的Primary节点 （wt）
- _waitForDelete
  - chunk commit时，源shard的chunk数据未删除。删除进度不持久化

- **数据备份 – 分片集群场景**
  - 分片集群数据一致性
    - _secondaryThrottle 为false

延伸忧虑：
如果目标Shard Primary挂了，会不会丢数据



Independent of the `secondaryThrottle` setting, certain phases of the chunk migration have the following replication policy:

- MongoDB briefly pauses all application writes to the source shard before updating the config servers with the new location for the chunk, and resumes the application writes after the update. The chunk move requires all writes to be acknowledged by majority of the members of the replica set both before and after committing the chunk move to config servers.
- When an outgoing chunk migration finishes and cleanup occurs, all writes must be replicated to a majority of servers before further cleanup (from other outgoing migrations) or new incoming migrations can proceed.

实际上：

Take it easy！

云服务实践

- **备份数据恢复**
  - 基于备份数据初始化新的复制集
    - Primary节点
  - 加入2个空的Secondary节点
    - 通过initial sync完成数据同步
      - MongoDB 3.4版本提升明显：oplog buffer等
  - **分片集群备份恢复**
    - 数据变多了！！

- 备份数据恢复
  - 分片集群备份恢复
    - 数据变多了？
      - Chunk迁移时旧数据异步非持久化删除

- _waitForDelete

The RangeDeleter

The migration protocol does not necessarily cause the data on the donor shard to be synchronously deleted after the migration completes (unless it is opted-in via the `_waitForDelete` field in the moveChunk command). This is because it would require waiting for active queries using data in the donated range to complete.

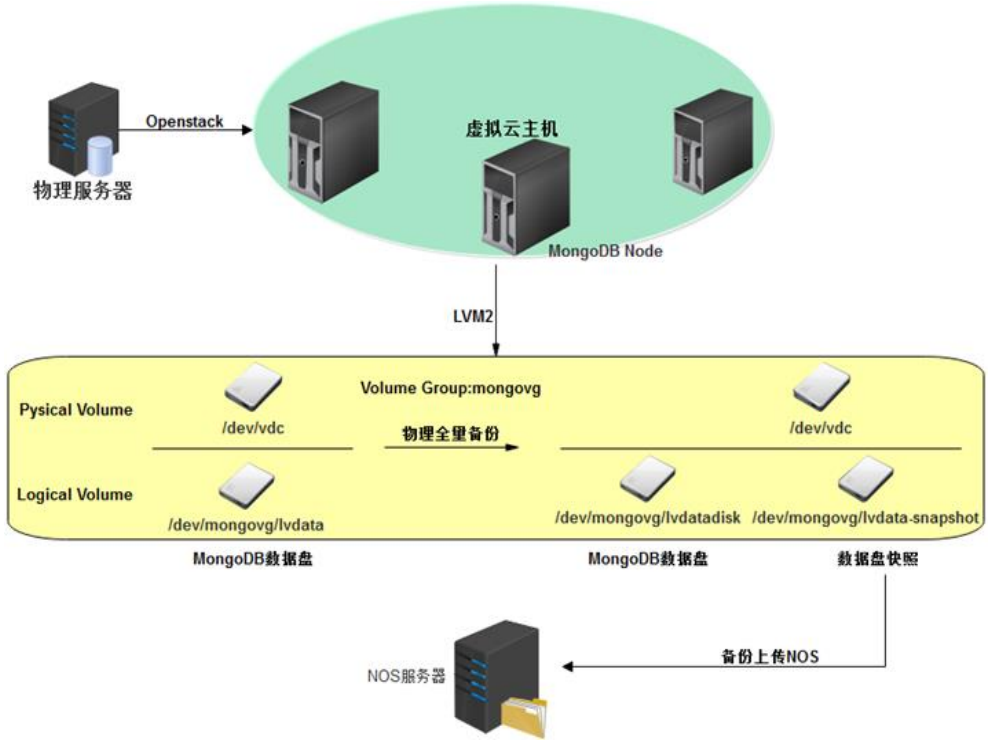
Instead, the `RangeDeleter` runs in a separate thread on the shard primary. The `RangeDeleter` deletes data in the donated chunk's range once all cursors that were active at the start of the migration have been exhausted.

If the shard primary crashes or fails over before the RangeDeleter has a chance to delete a range, the documents in that range remain "orphaned" on the donor shard. This is because the RangeDeleter does not persist or replicate the ranges it has yet to clean.

However, there is a `cleanupOrphaned` command that can be issued against the new primary to delete any such orphaned documents.

So What？

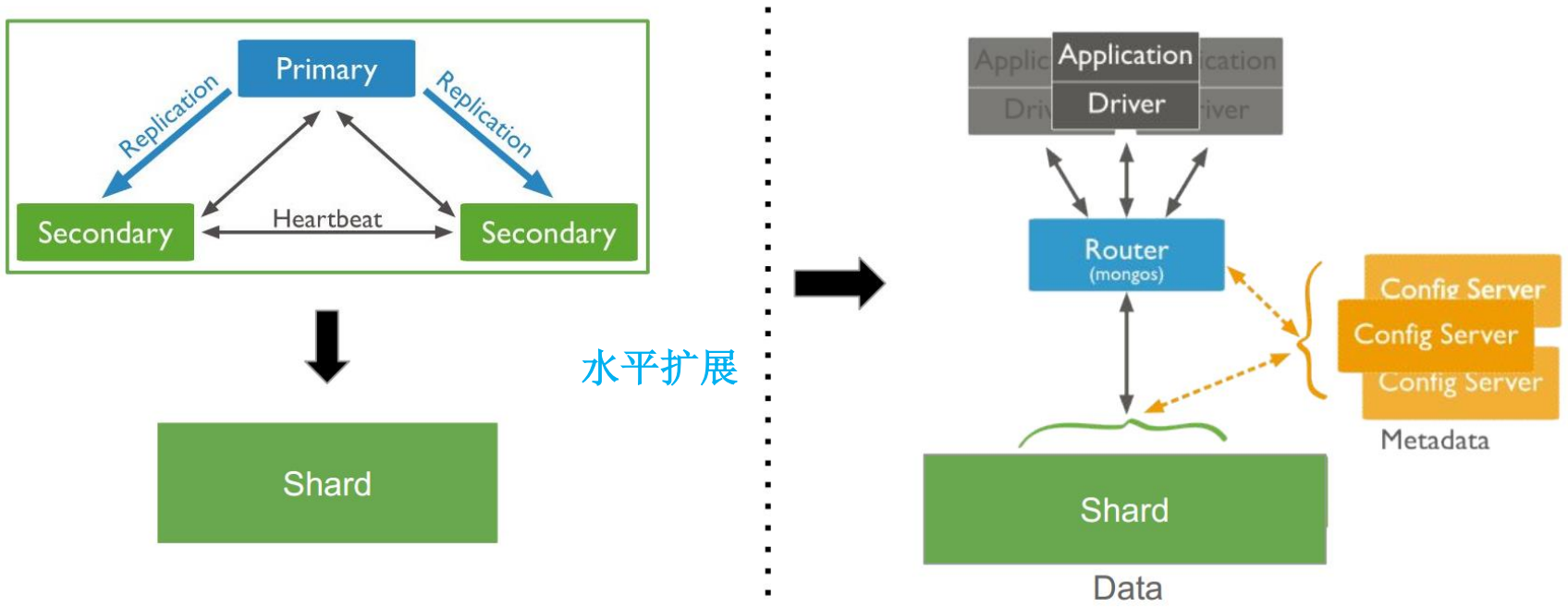
- 备份数据恢复
  - 分片集群备份恢复
    - 数据变多了？
      - Chunk迁移时旧数据异步非持久化删除
    - _waitForDelete



Snapshot就是一次内存版的mongod Crash。

只是对磁盘数据做Snapshot，无法保存内存中数据

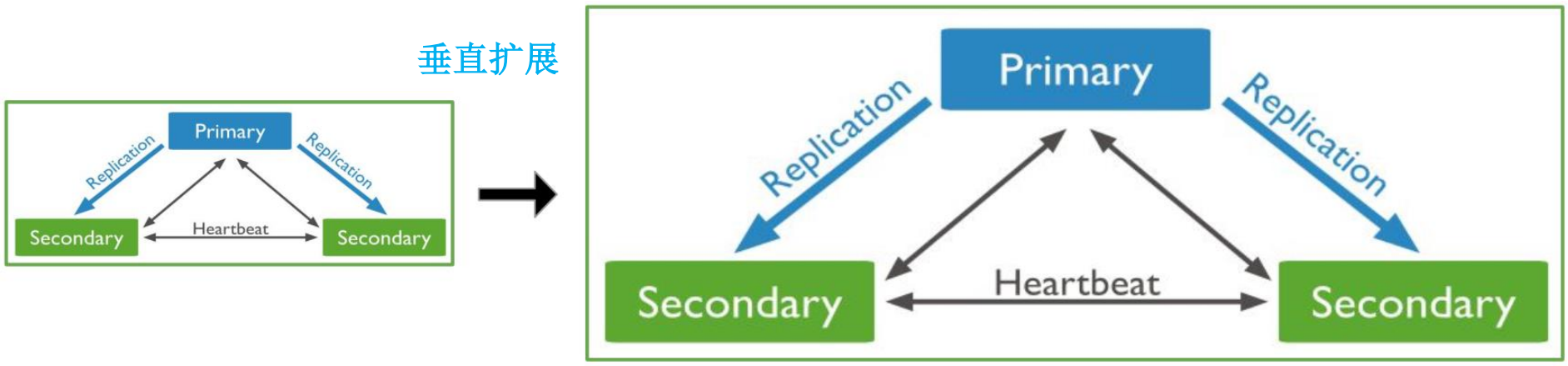
- **实例规格升级与扩容**
  - 遇到瓶颈：业务压力太大，数据盘空间不够
    - 分片集群



复杂度考虑！

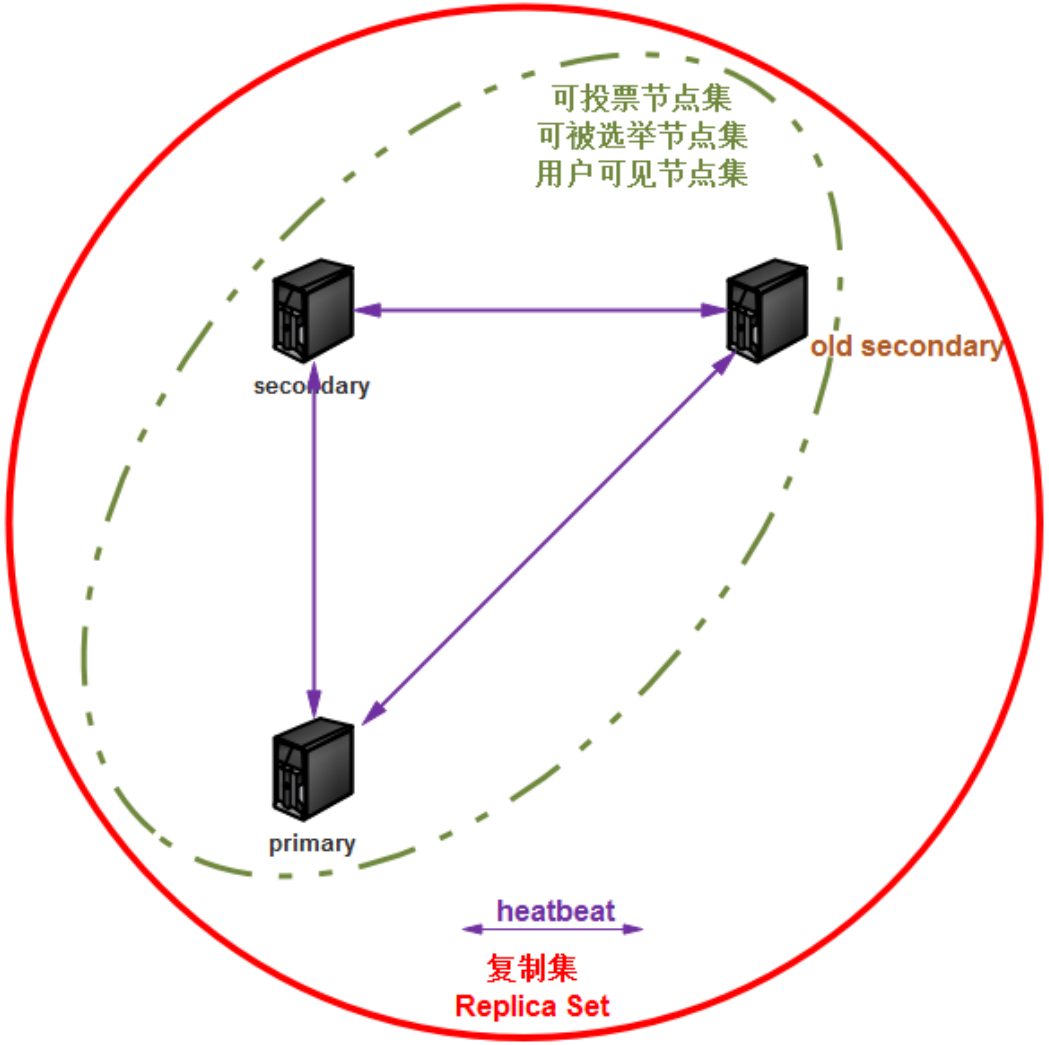
- **实例规格升级与扩容**
  - 遇到瓶颈：业务压力太大，数据盘空间不够
    - 更大的复制集

垂直扩展

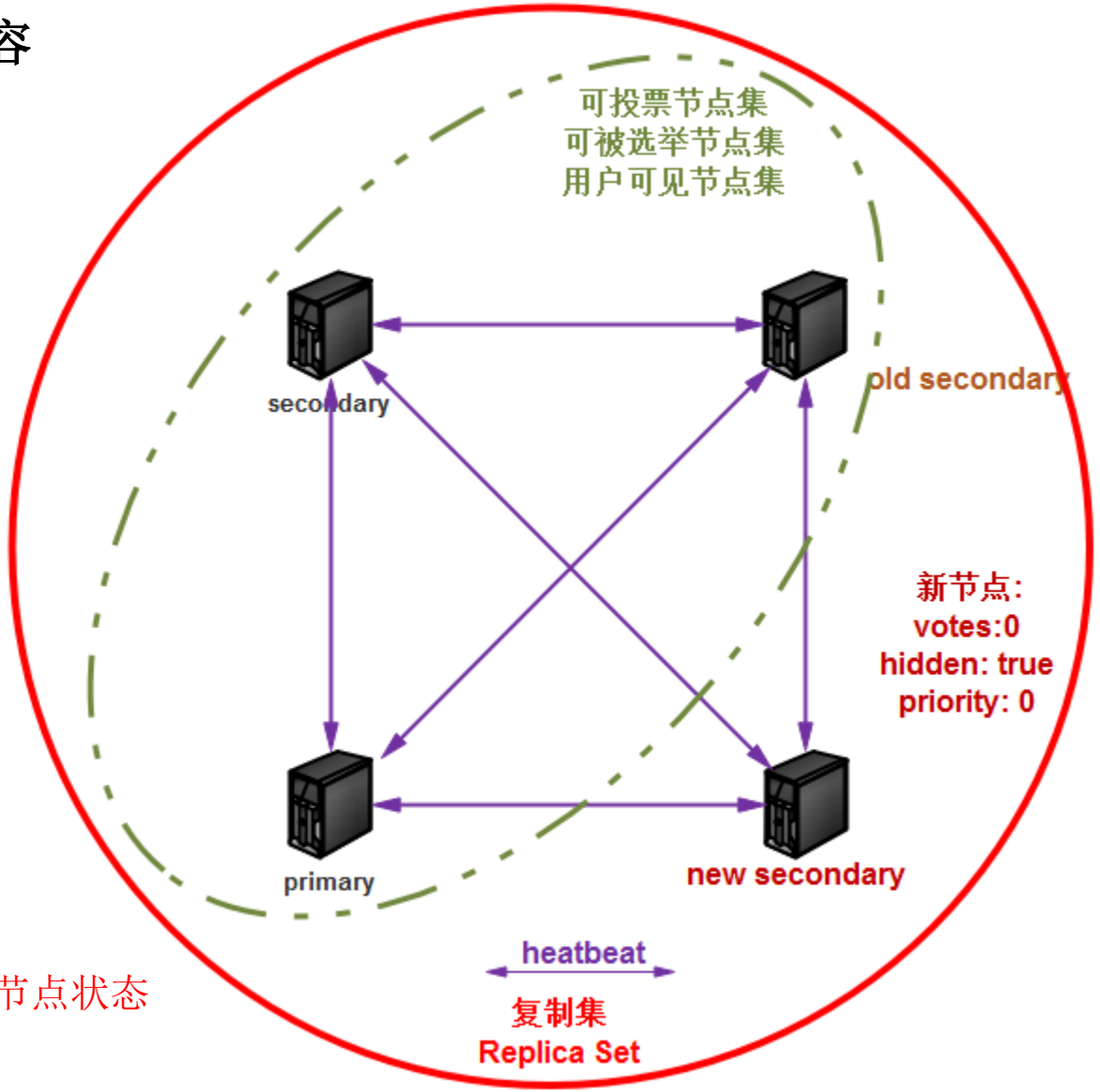


3个复制集节点逐次修改

- 实例规格升级与扩容
  - 修改前

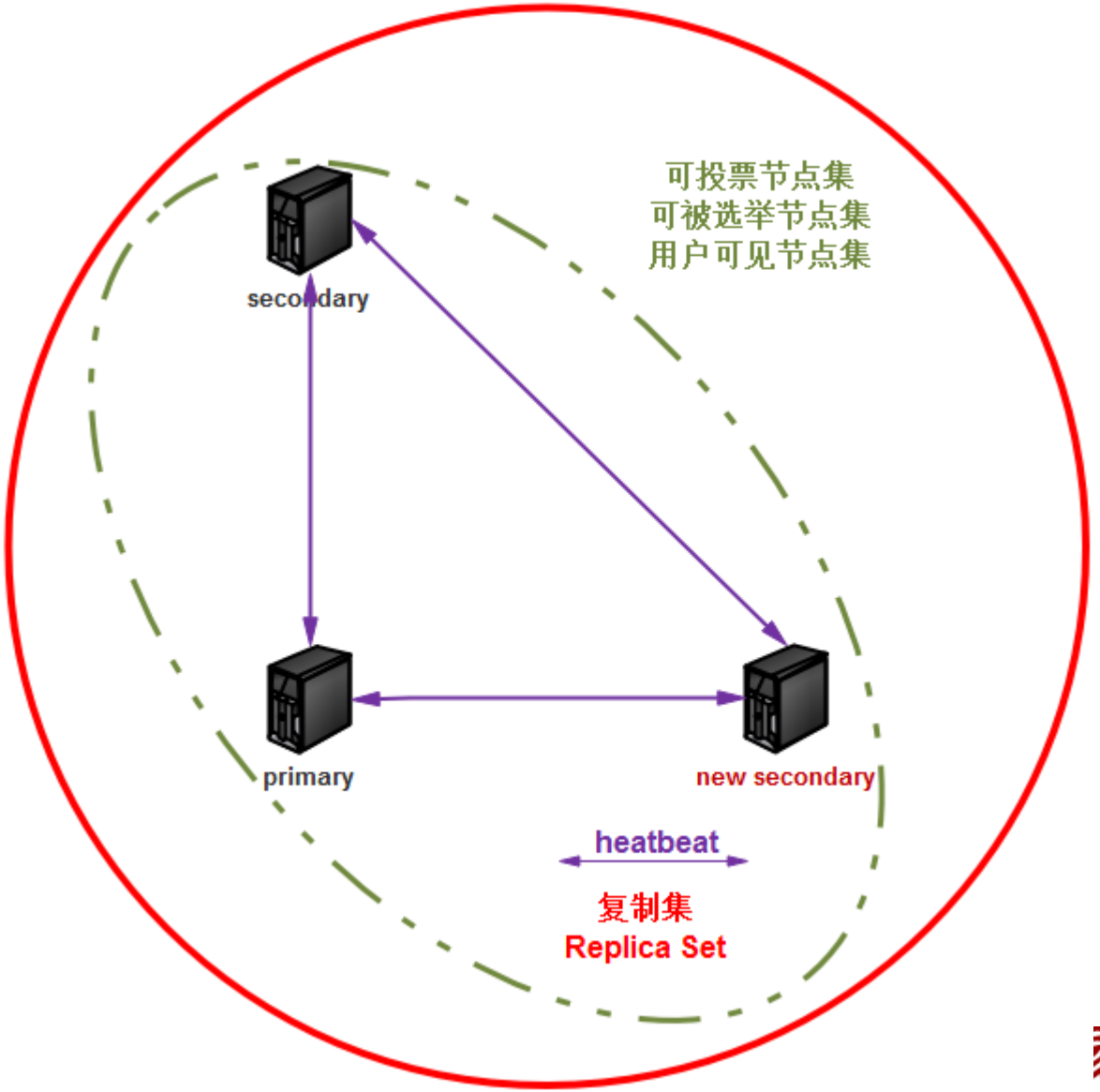


- **实例规格升级与扩容**
  - 修改中
  - 新节点属性
    - 隐藏节点
    - 非投票节点



修改中维持正常3节点状态

- 实例规格升级与扩容
  - 恢复正常



- **实例规格升级与扩容 – 优化**
  - 本地（同宿主机）升级
    - 升规格时，只需换云主机，逻辑卷重新挂载

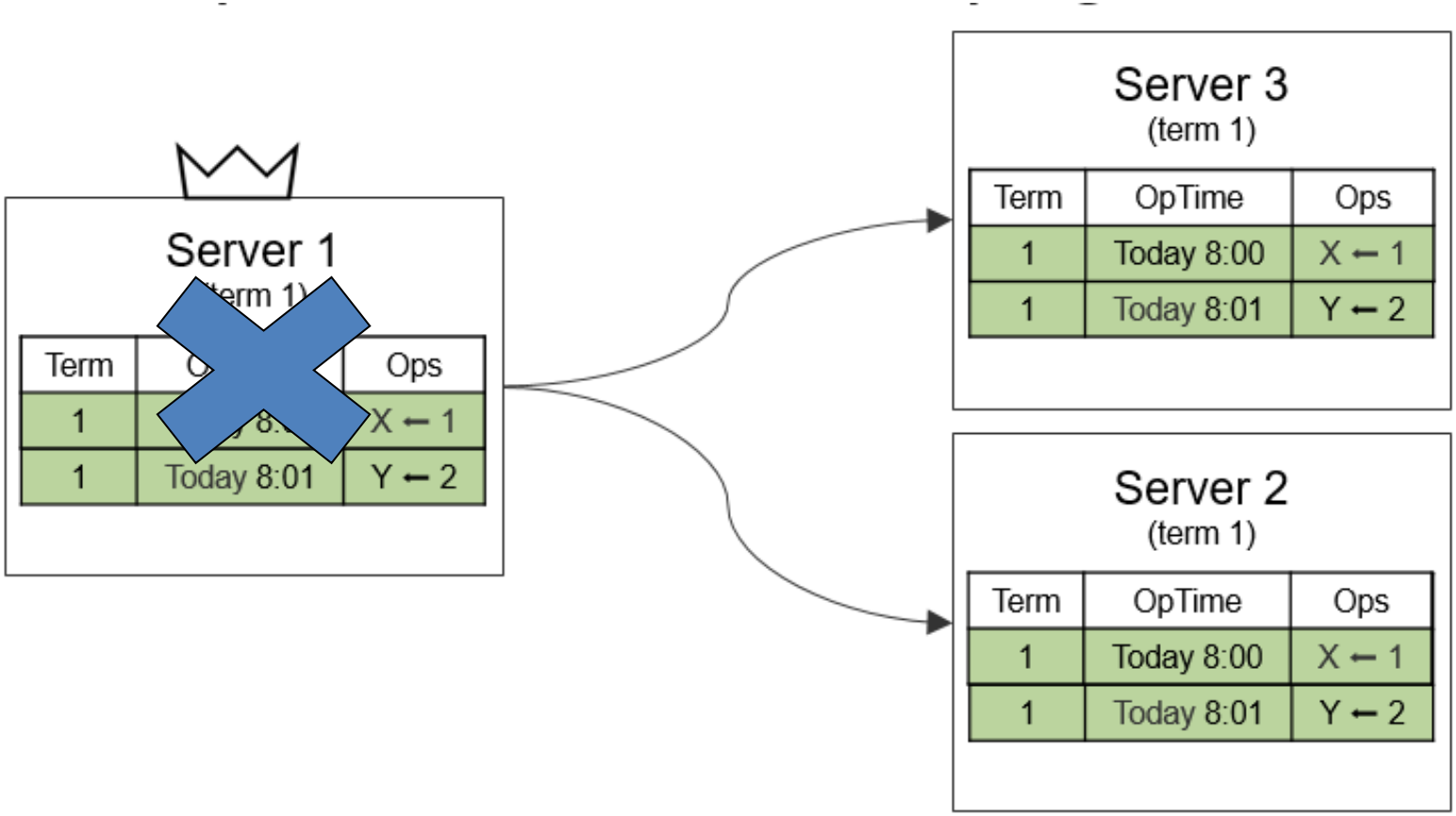
将数据盘挂载到新的云主机上并重启 mongod。步骤为：

1、 先将卷组 datavg 激活（vgchange -a y datavg）

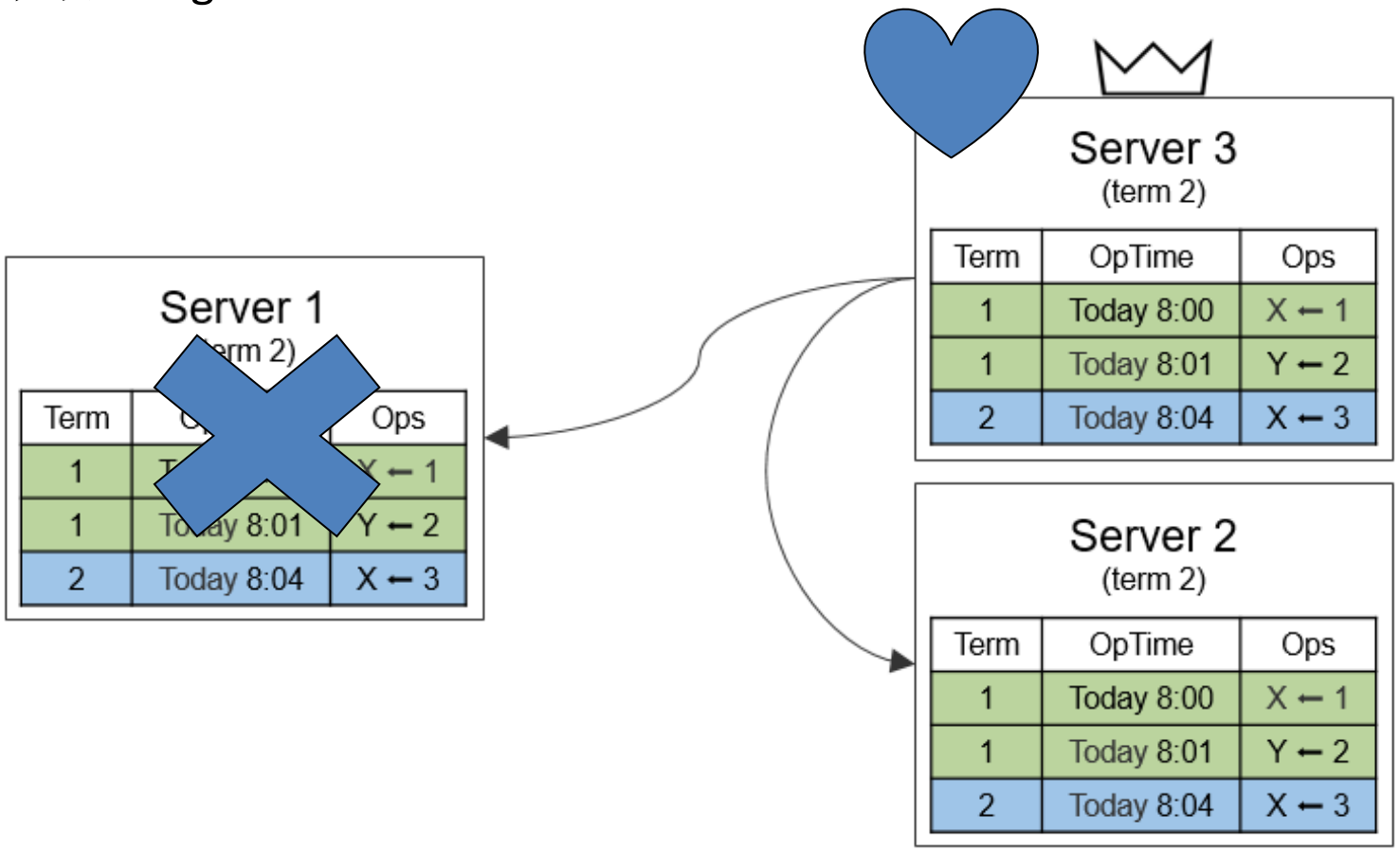
2、 再将 MongoDB 数据卷 mount 到/ebs 上

  - 异地升级
    - 扩容时/本地无资源时
      - 新节点数据initial sync

- **故障修复 – raft选主**
  - 不干预MongoDB选主



- **故障修复 – raft选主**
  - 不干预MongoDB选主

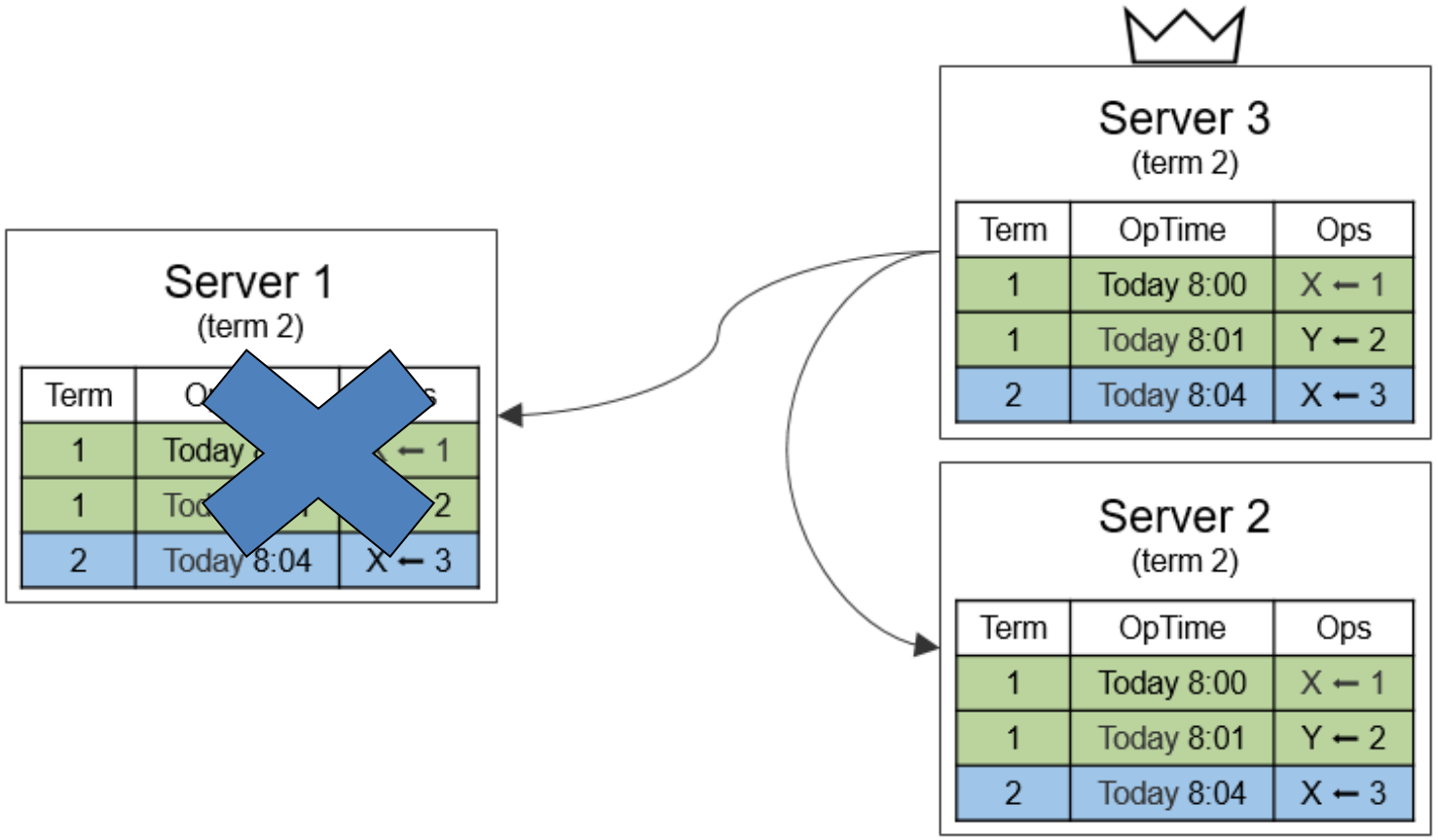


- **故障修复 – 故障检测**
  - 节点心跳
    - Secondary节点
      - 本节点mongod宕机
      - 本节点数据过旧
    - Primary节点
      - 处理主从切换时飘浮动IP
      - Secondary节点主机宕机
  - 实例心跳轮训线程
    - 多节点主机宕机
    - 复制集无Primary

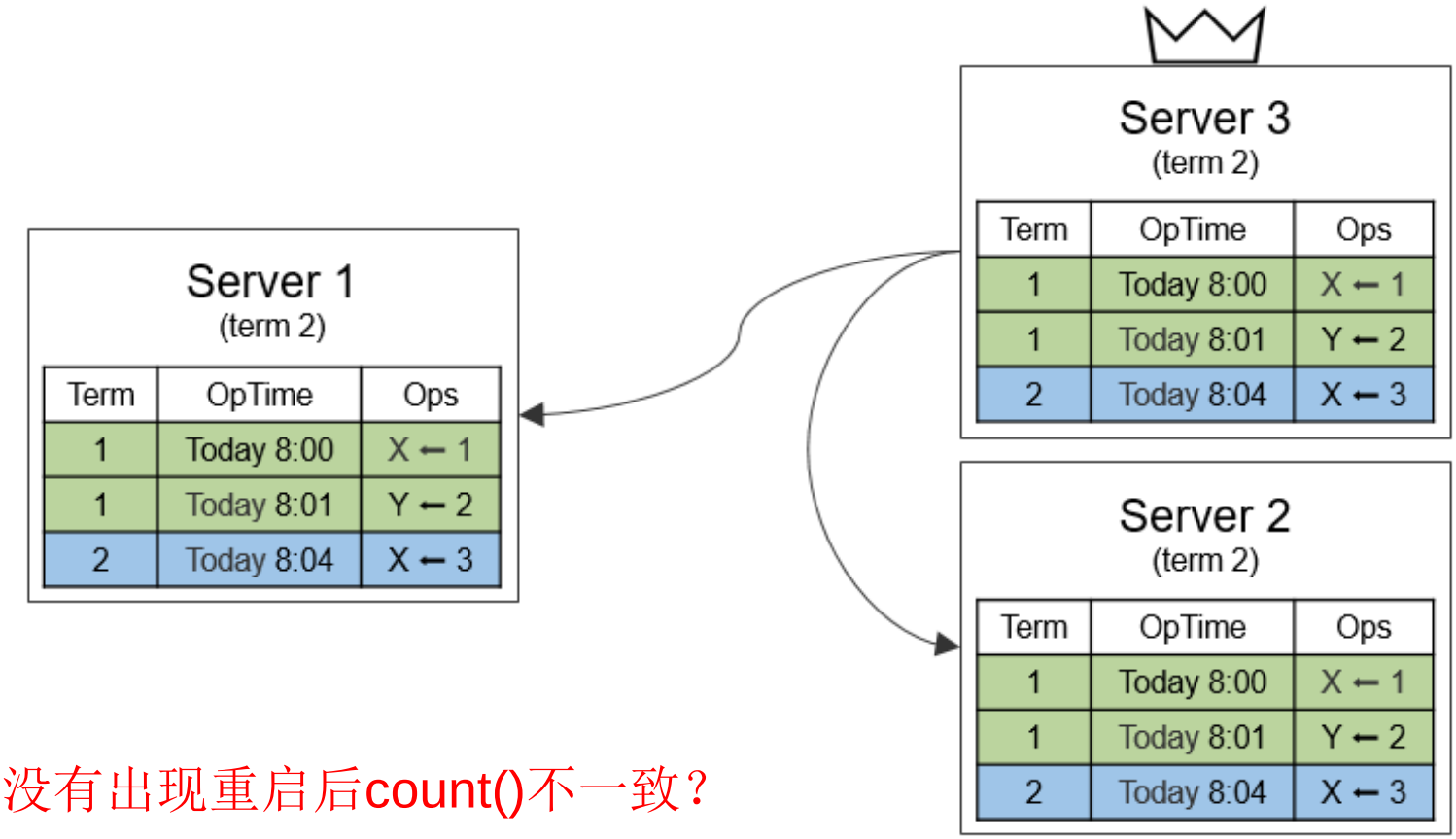
节点心跳内容：

- 本节点状态信息
- 复制集状态信息
- 异常节点信息

- **故障修复 – 实现**
  - 重启或重建旧主

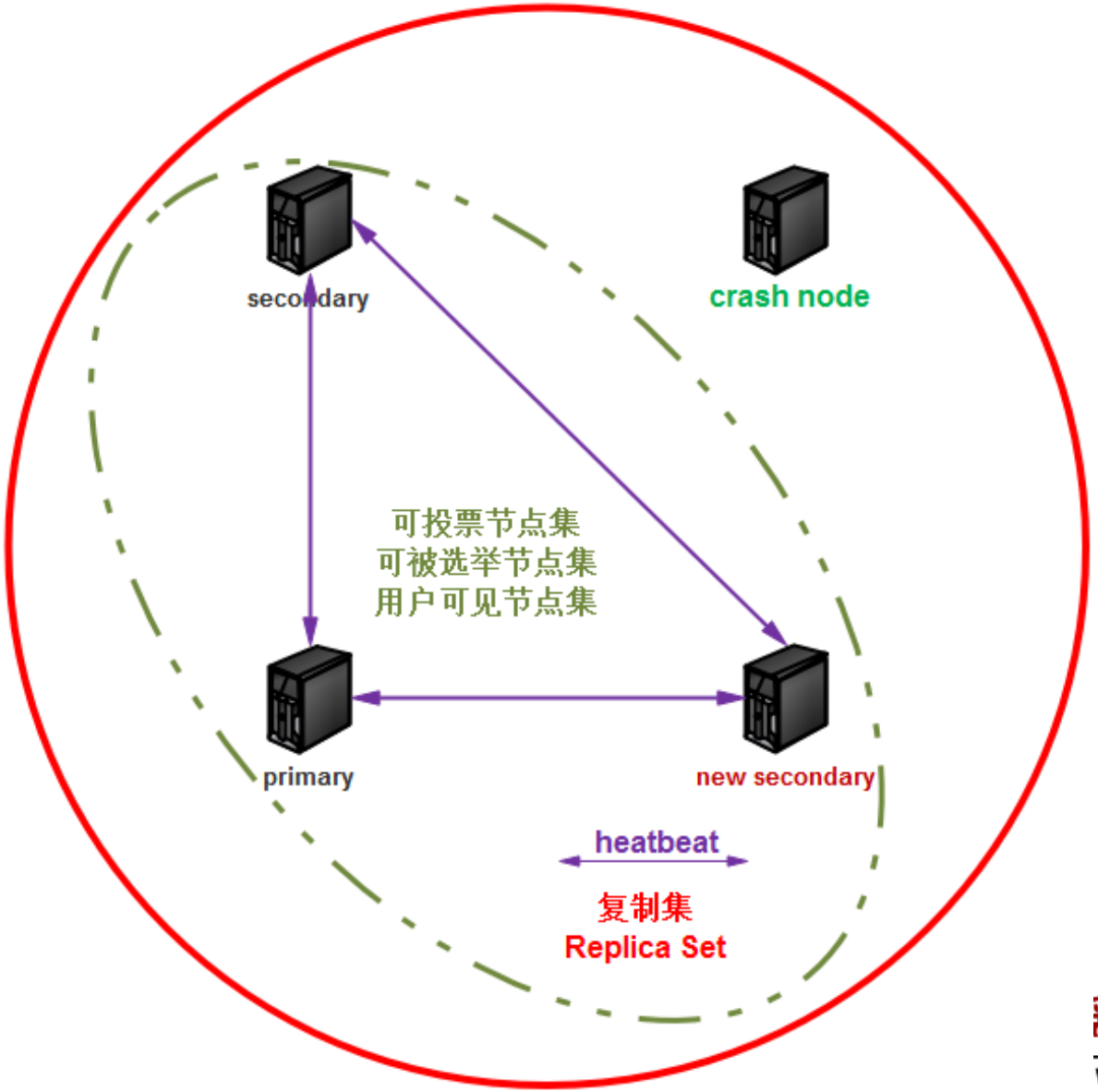


- **故障修复 – 实现**
  - 重启旧主成功



有没有出现重启后count()不一致？

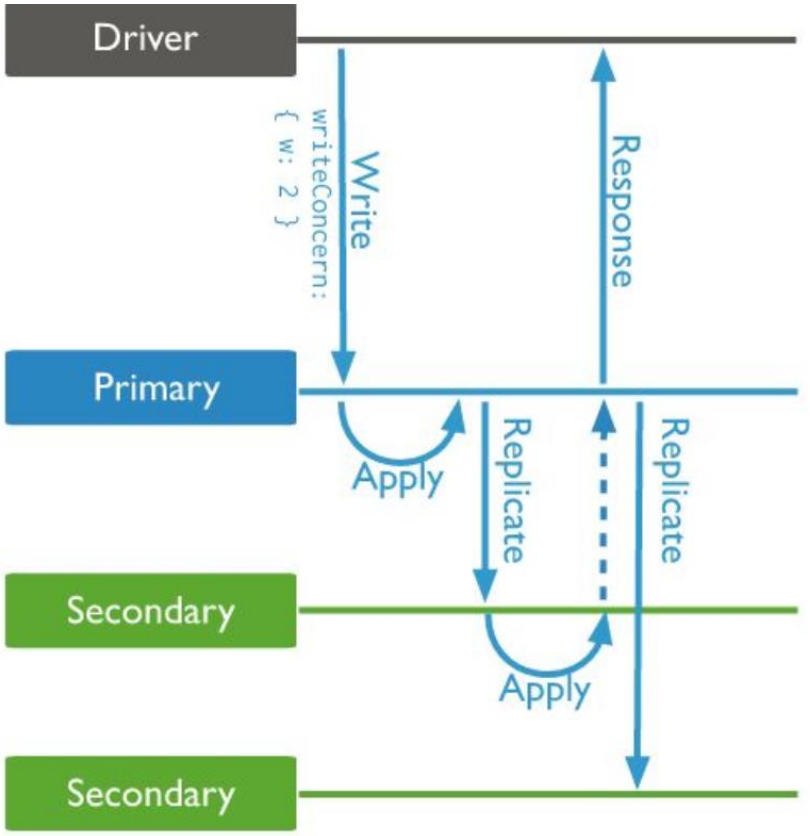
- **故障修复 – 设计**
  - 故障节点重建



- 读写行为
  - Write concern

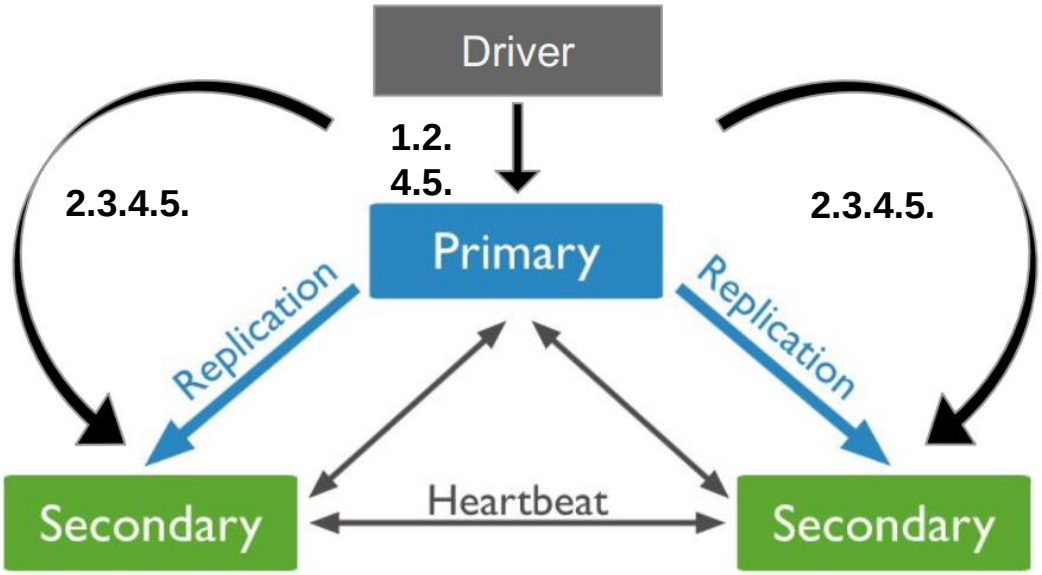
{ w: <value>, j: <boolean>, wtimeout: <number> }

- w - number of mongod instances that acknowledged the write operation
- j - acknowledgement that the write operation has been written to the journal
- wtimeout - time limit to prevent write operations from blocking indefinitely



- 读写行为
  - Read preference

1. primary (by default)
2. primaryPreferred
3. secondary
4. secondaryPreferred
5. nearest (least network latency)



MongoDB 3.4 maxStalenessSeconds (>= 90 seconds)

- 读写行为
  - Read Concern

The following read concern levels are available:

level	Description
"local"	Default. The query returns the instance's most recent data. Provides no guarantee that the data has been written to a majority of the replica set members (i.e. may be rolled back).
"majority"	The query returns the instance's most recent data acknowledged as having been written to a majority of members in the replica set.
"linearizable"	The query returns data that reflects all successful writes issued with a write concern of "majority" *and* acknowledged prior to the start of the read operation. For replica sets that run with writeConcernMajorityJournalDefault set to true, linearizable read concern returns data that will never be rolled back.

- 驱动设置
  - 复制集
    - 仅需提供种子，db.isMaster()获取更多信息

```
mg163-582:PRIMARY> db.isMaster()
{
        "hosts" : [
                "10.173.33.6:27021",
                "10.173.33.59:27021",
                "10.173.33.58:27021"
        ],
        "setName" : "mg163-582",
        "setVersion" : 103288,
        "ismaster" : true,
        "secondary" : false,
        "primary" : "10.173.33.6:27021",
        "me" : "10.173.33.6:27021",
        "electionId" : ObjectId("7fffffff0000000000000001"),
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1499739032, 1),
                        "t" : NumberLong(1)
                },
                "lastWriteDate" : ISODate("2017-07-11T02:10:32Z")
        },
        "maxBsonObjectSize" : 16777216,
        "maxMessageSizeBytes" : 48000000,
        "maxWriteBatchSize" : 1000,
        "localTime" : ISODate("2017-07-11T02:10:34.823Z"),
        "maxWireVersion" : 5,
        "minWireVersion" : 0,
        "readOnly" : false,
        "ok" : 1
}
```

```
mg163-582:SECONDARY> db.isMaster()
{
        "hosts" : [
                "10.173.33.6:27021",
                "10.173.33.59:27021",
                "10.173.33.58:27021"
        ],
        "setName" : "mg163-582",
        "setVersion" : 103288,
        "ismaster" : false,
        "secondary" : true,
        "primary" : "10.173.33.6:27021",
        "me" : "10.173.33.59:27021",
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1499739072, 1),
                        "t" : NumberLong(1)
                },
                "lastWriteDate" : ISODate("2017-07-11T02:11:12Z")
        },
        "maxBsonObjectSize" : 16777216,
        "maxMessageSizeBytes" : 48000000,
        "maxWriteBatchSize" : 1000,
        "localTime" : ISODate("2017-07-11T02:11:20.520Z"),
        "maxWireVersion" : 5,
        "minWireVersion" : 0,
        "readOnly" : false,
        "ok" : 1
}
```

mongodb://root:****@10.173.33.6:27021/admin?replicaSet=mg163-582

- 驱动设置
  - 分片集群
    - 需提供全集，驱动不会自动加上其他mongos
    - 配置不少于2个mongos，实现业务访问高可用

```
mongos> db.isMaster()
{
        "ismaster" : true,
        "msg" : "isdbgrid",
        "maxBsonObjectSize" : 16777216,
        "maxMessageSizeBytes" : 48000000,
        "maxWriteBatchSize" : 1000,
        "localTime" : ISODate("2017-07-11T02:17:42.473Z"),
        "maxWireVersion" : 5,
        "minWireVersion" : 0,
        "ok" : 1
}
```

云服务实践

- **驱动设置**
  - 分片集群
    - 多个mongos实现业务负载均衡
    - 多个mongos分组，服务不同的上层业务
    - 复制集相关配置透传
      - readPreference
      - writeConcern
      - 等等
  - 其他配置
    - maxPoolSize
    - 等等

云服务实践

- 使用场景
  - 游戏
  - 电商
  - 日志
  - 物联网
  - 。。。

Q&A