10gen mongoDB

使用 MongoDB 进行伸缩

Alvin Richards

alvin@10gen.com

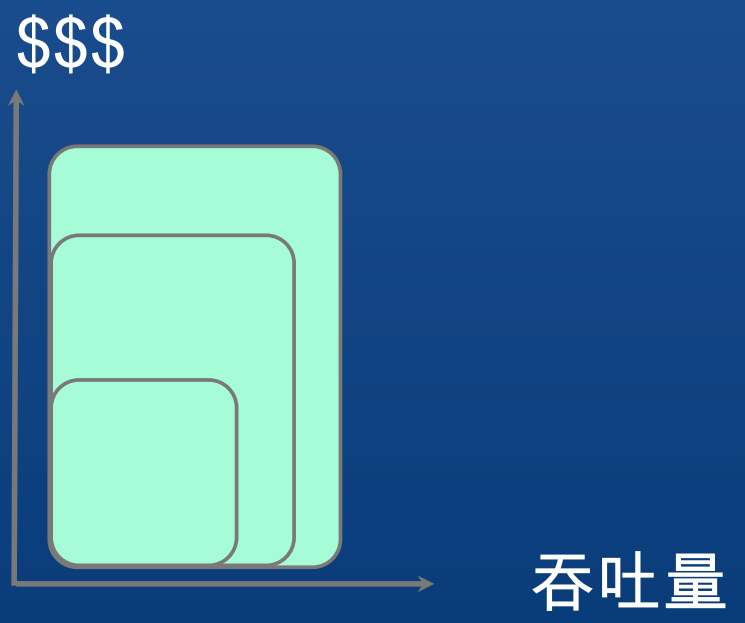
议程

- 垂直伸缩
- 使用 MongoDB 进行横向伸缩
  - 架构及索引设计
  - 自动水平切分 (Auto Sharding)
  - 复制

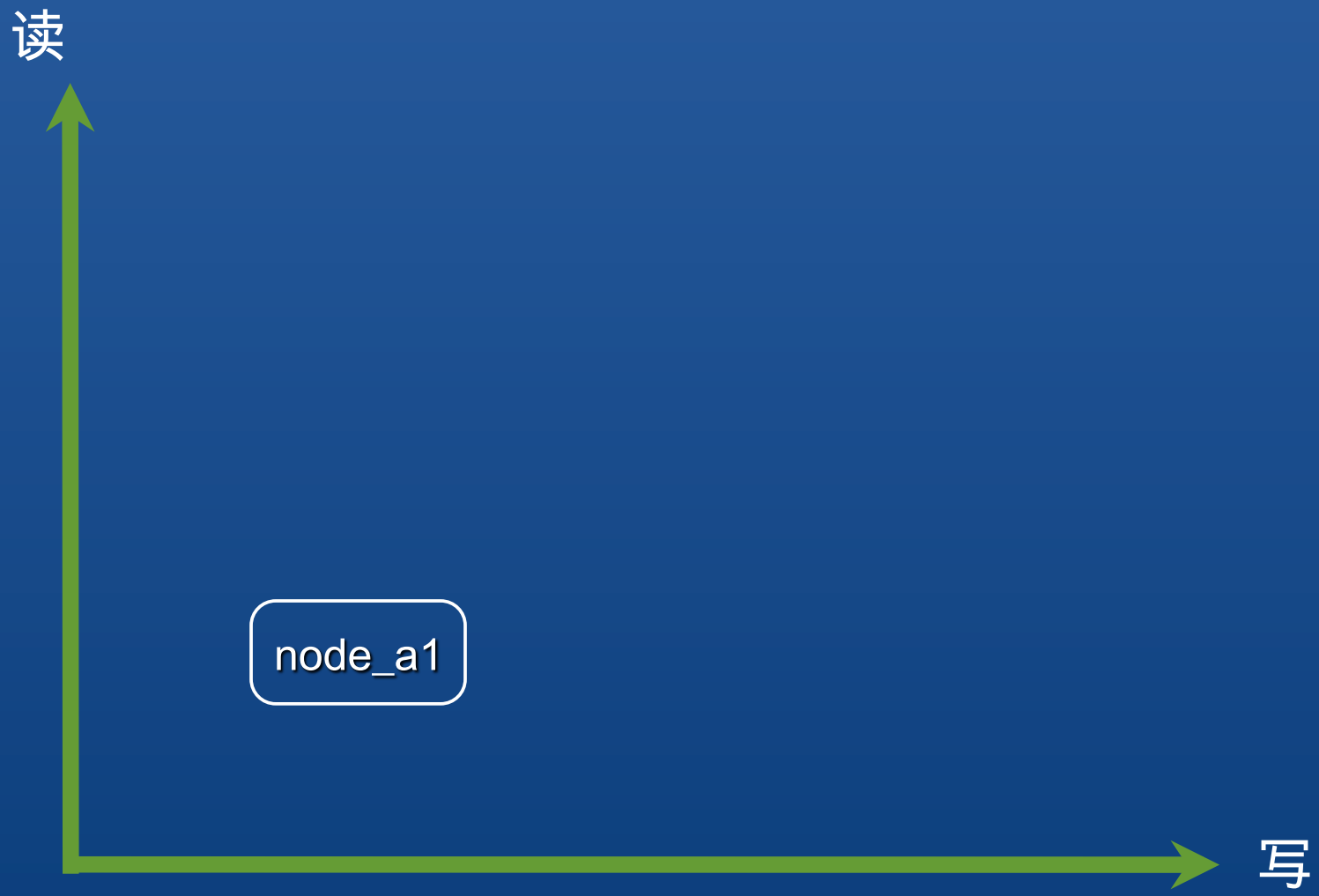
伸缩

- 每秒操作次数增加
- 存储需要增加
  - 容量
  - IOPs
- 复杂程度增加
  - 缓存

您现在如何进行伸缩？

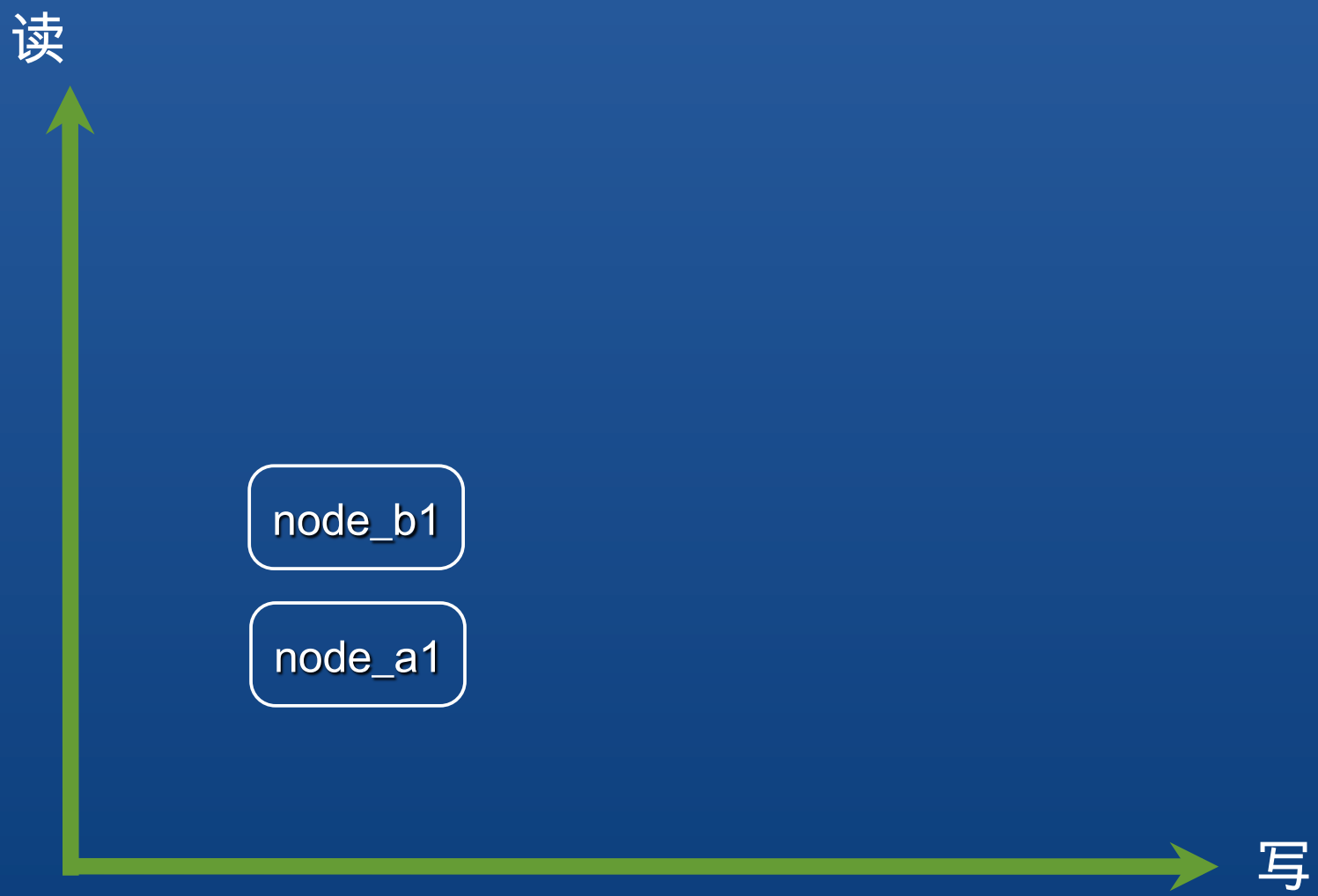


MongoDB 伸缩——单节点



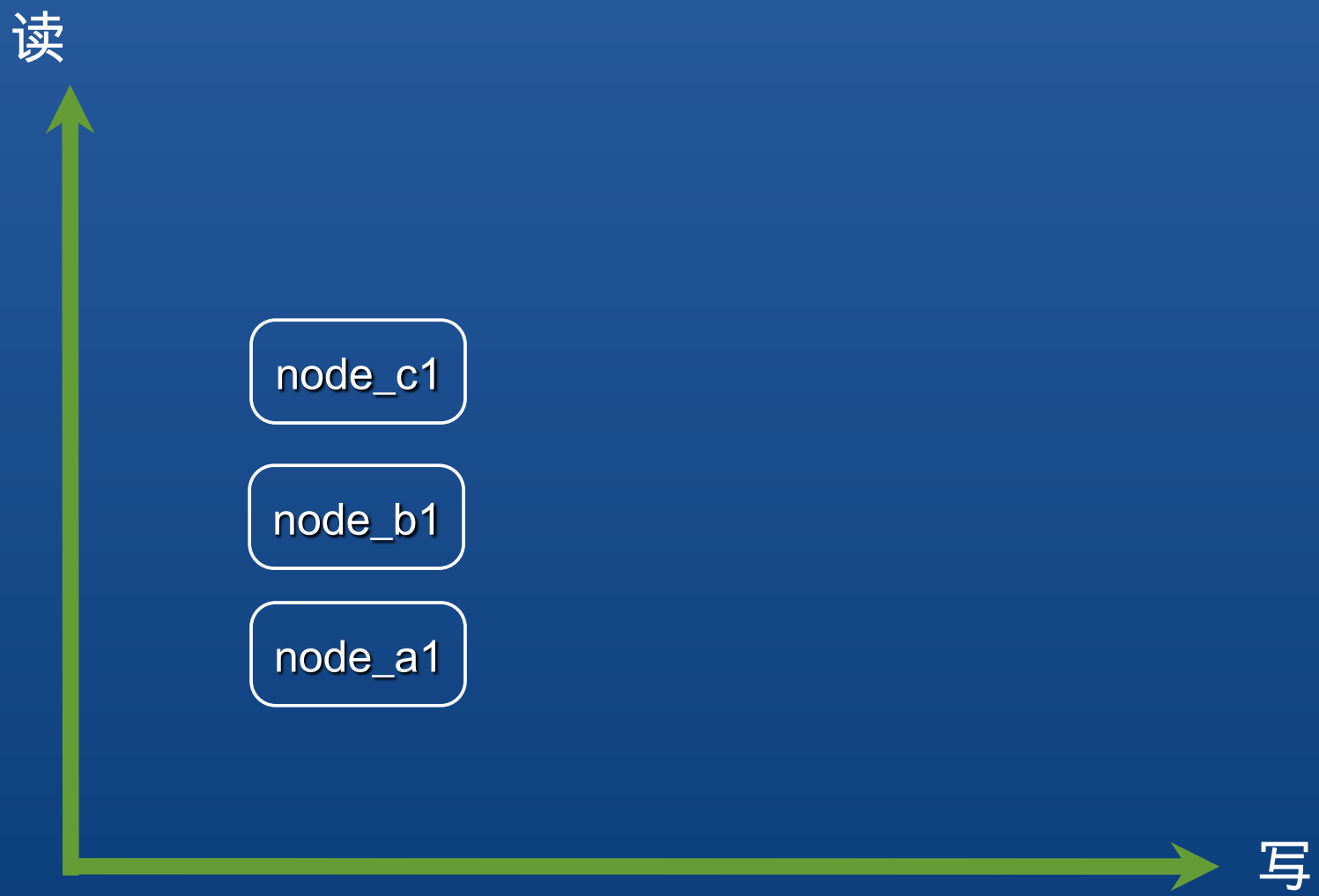
10gen mongoDB

读伸缩——新增副本



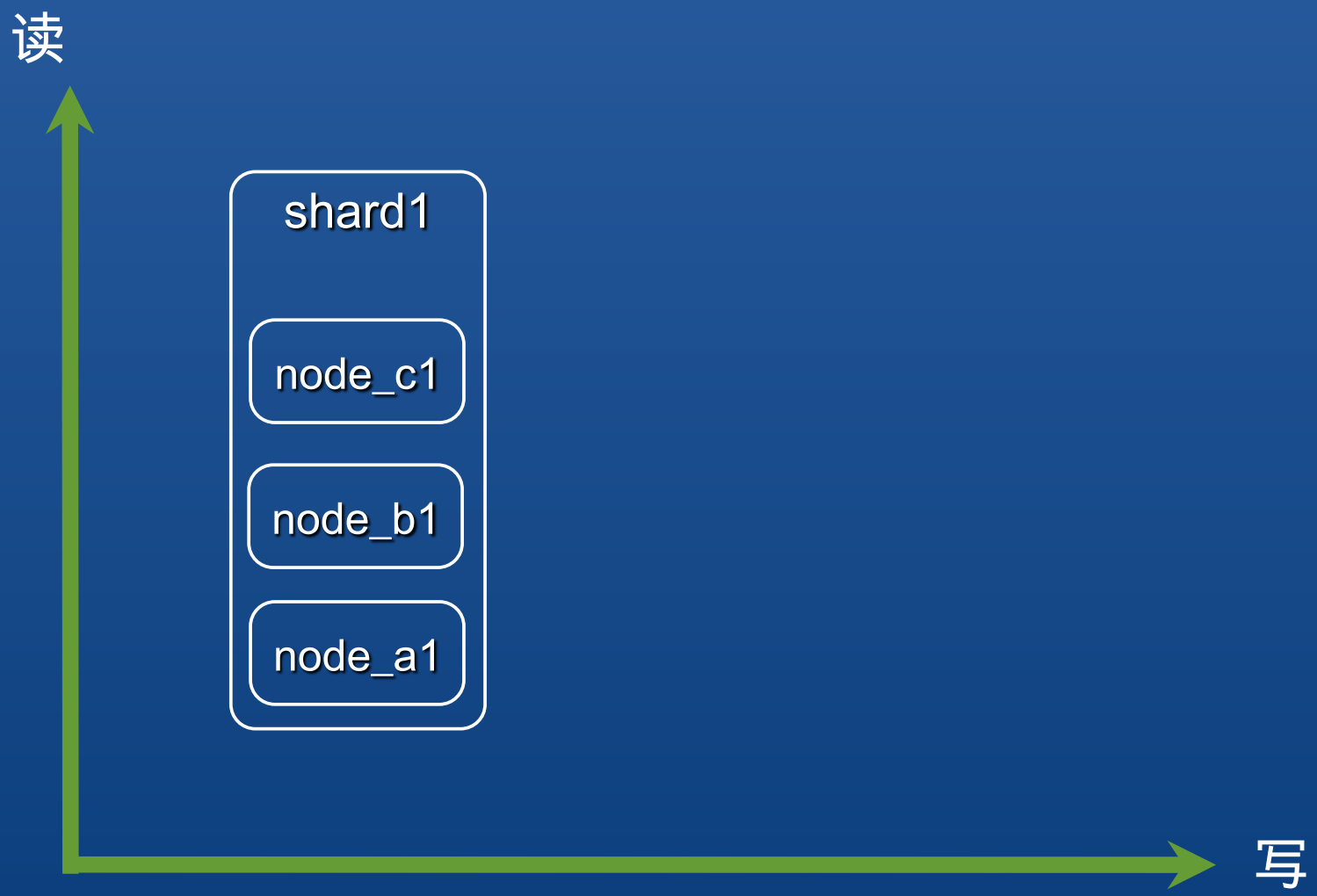
10gen mongoDB

读伸缩——新增副本



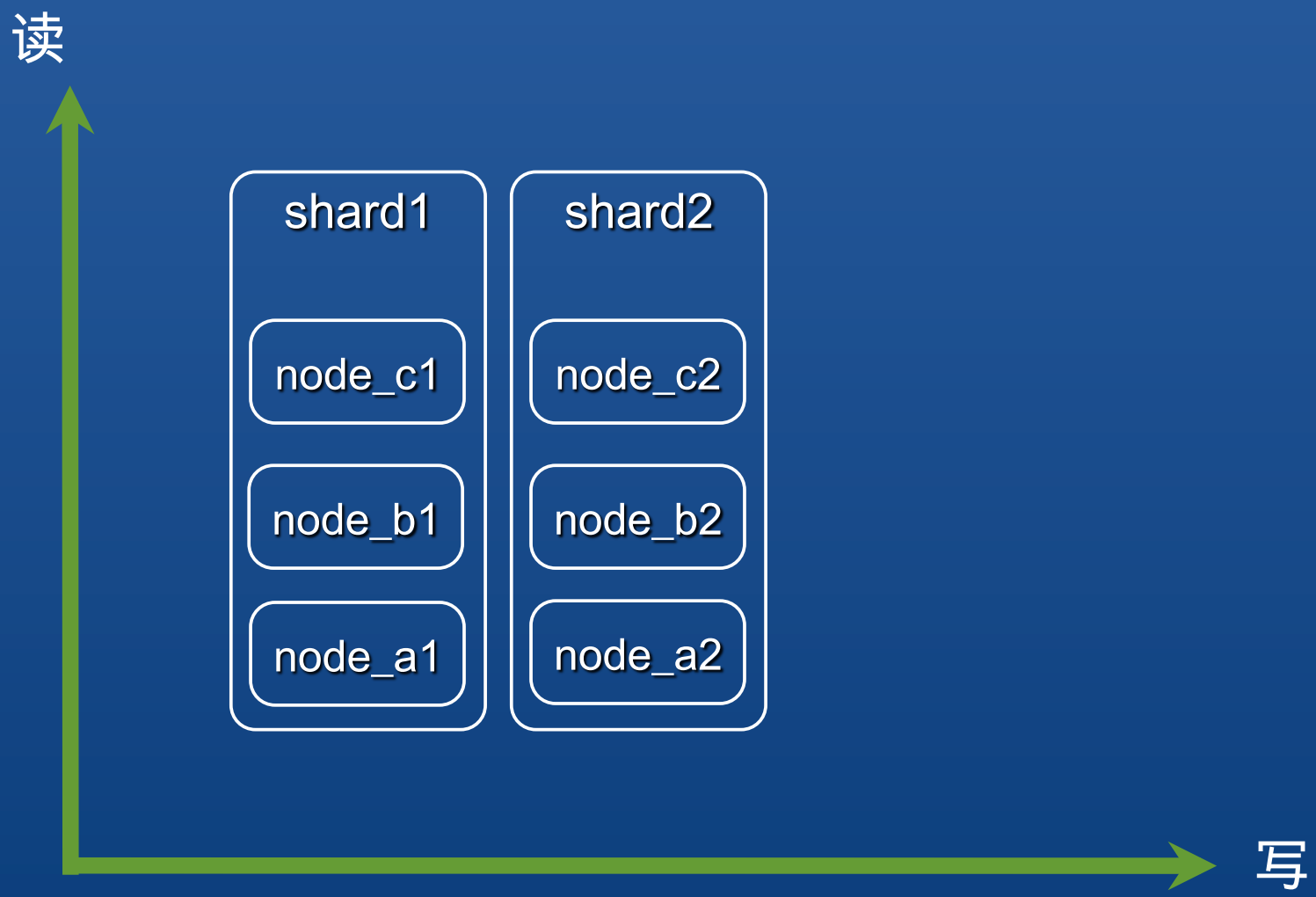
10gen mongoDB

写伸缩——水平切分



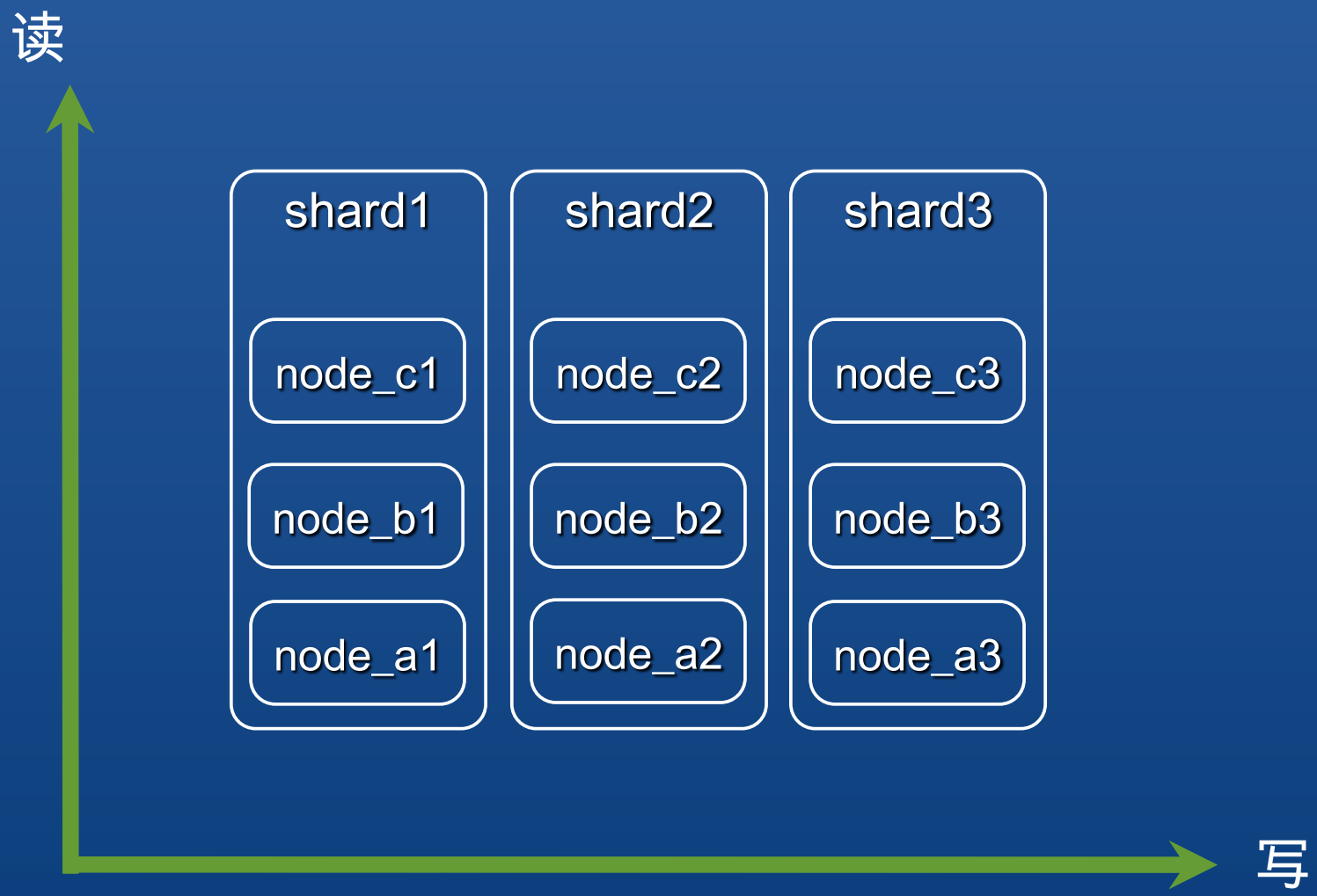
10gen mongoDB

写伸缩——添加水平切分



10gen mongoDB

写伸缩——添加水平切分



10gen mongoDB

使用 MongoDB 进行伸缩

- 架构及索引设计
- 水平切分
- 复制

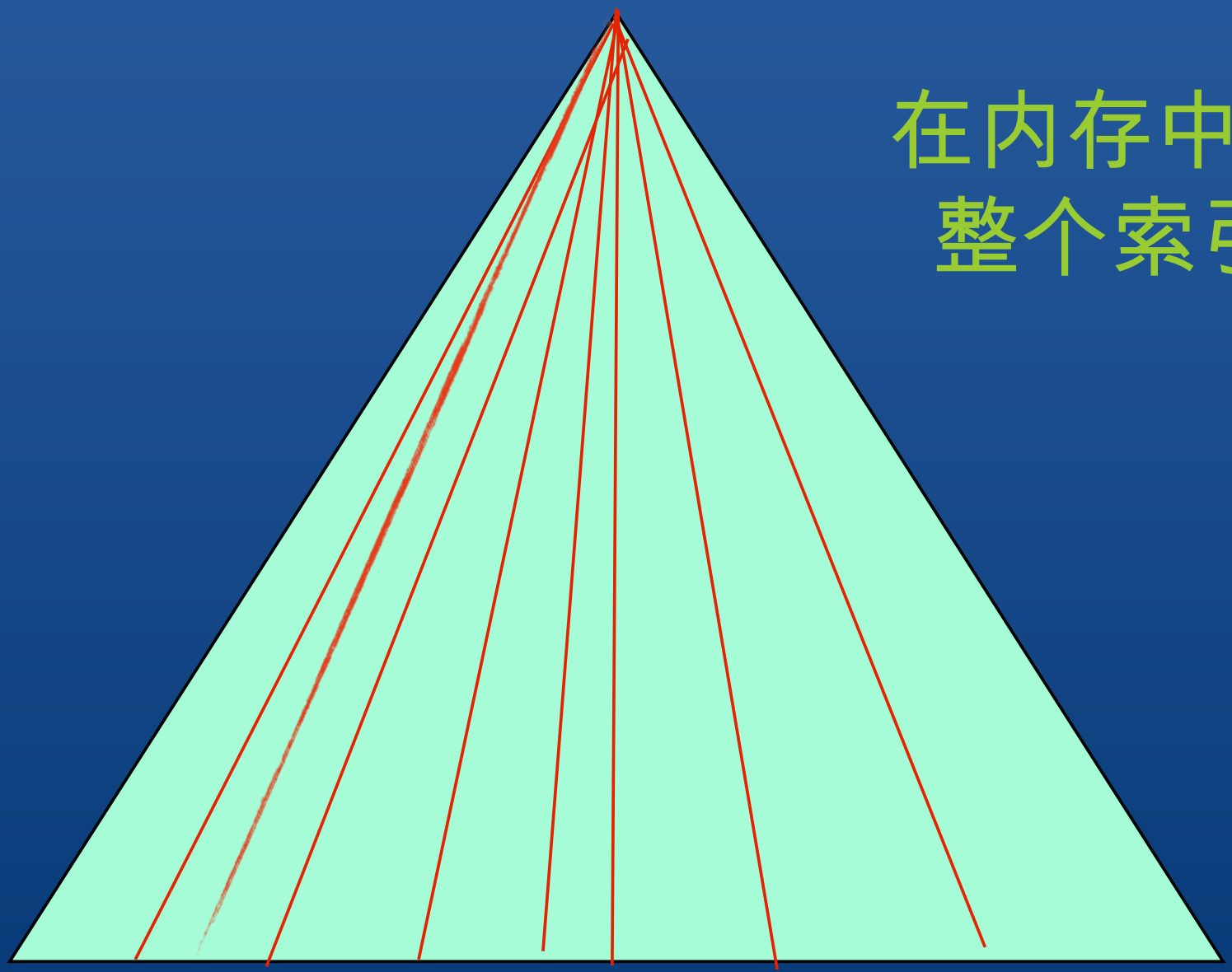
架构

- 数据模型对性能的影响
  - 嵌入与链接
    - 到数据库的环回时间
    - 磁盘寻道时间
    - 有读写数据时的大小
  - 部分文件写入与整个文件写入
- 性能问题可通过更改架构来解决

索引

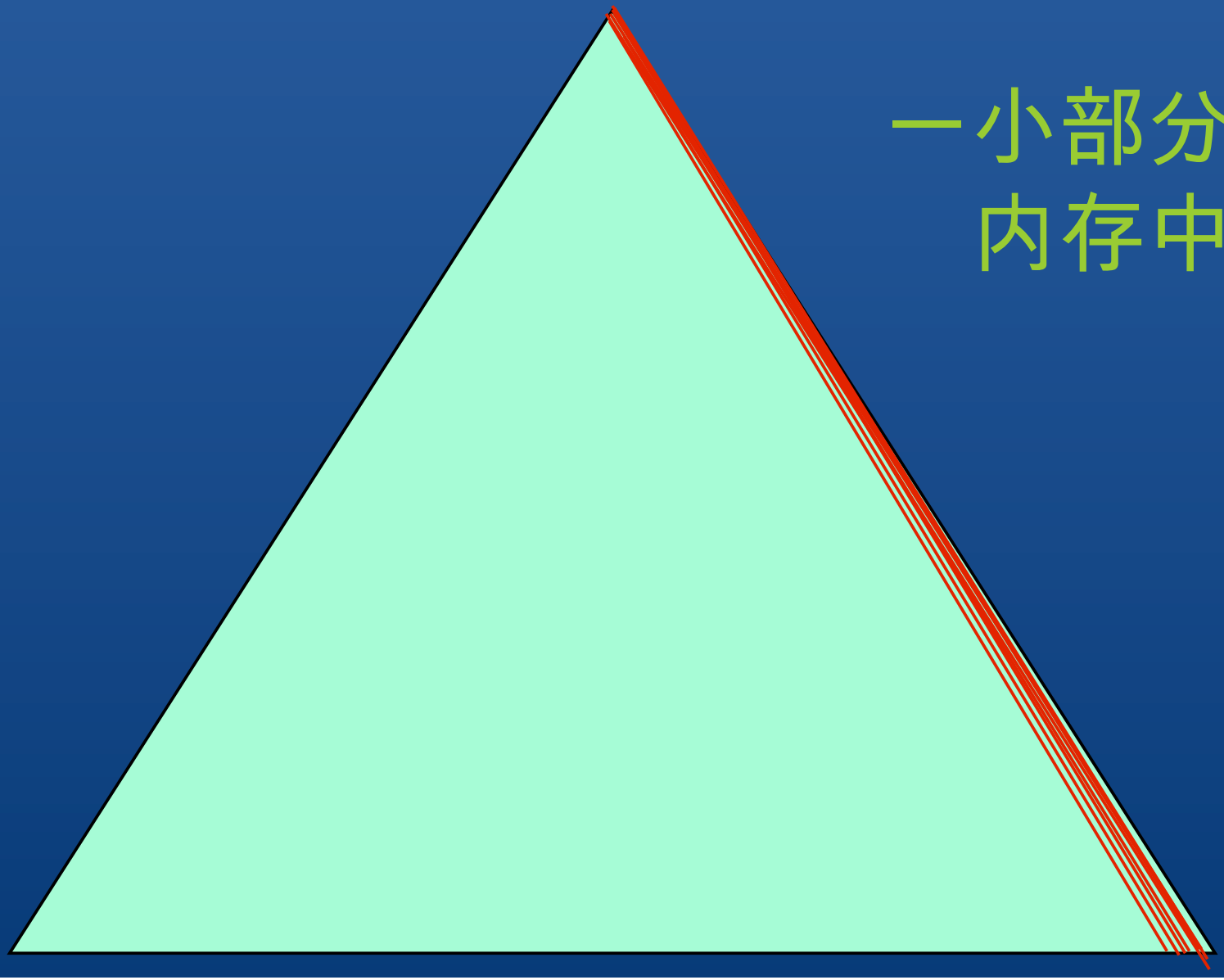
- 索引常见查询
- 不要过分索引
  - (A) 和 (A,B) 是相等的，选择一个
- 正确平衡的索引会使工作集较小

随机索引访问



在内存中放
整个索引

正确平衡的索引访问



一小部分在
内存中

什么是水平切分 (Sharding)

- 特设分区
- 一致性哈希散列
  - Amazon Dynamo
- 基于范围的分区
  - Google BigTable
  - Yahoo!PNUTS
  - MongoDB

MongoDB 水平切分

- 自动分区及管理
- 基于值域
- 不停机转换到水平切分系统
- 在单主上几乎不损失任何功能
- 完全一致

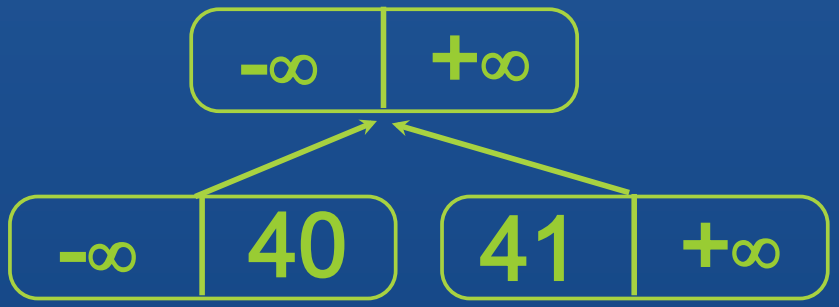
MongoDB 水平切分如何工作

```
> db.runCommand( { addshard :"shard1" } );
```

- 值域键从-∞ 到 +∞
- 值域以“块”的形式存储

MongoDB 水平切分如何工作

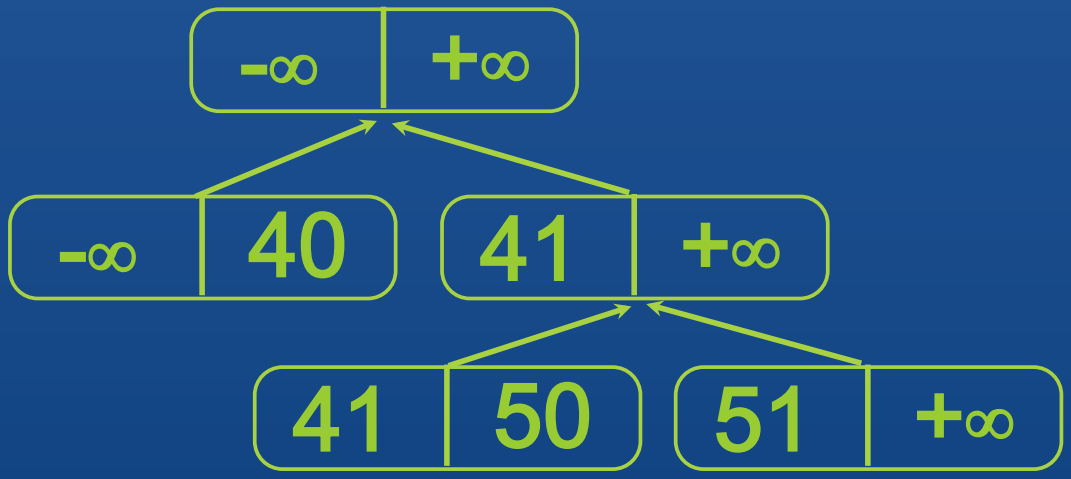
```
> db.posts.save( {age:40} )
```



- 插入数据
- 值域被分为更多个“块”

MongoDB 水平切分如何工作

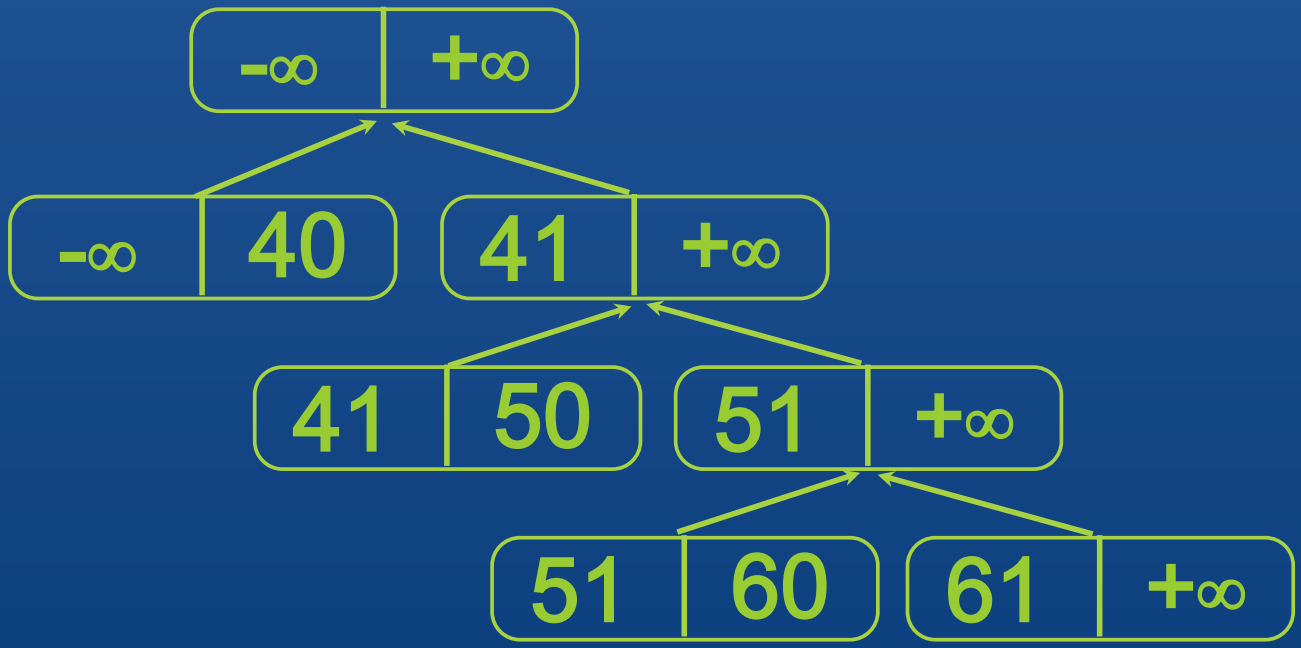
```
> db.posts.save( {age:40} )
> db.posts.save( {age:50} )
```



- 插入更多数据
- 值域被分为更多个“块”

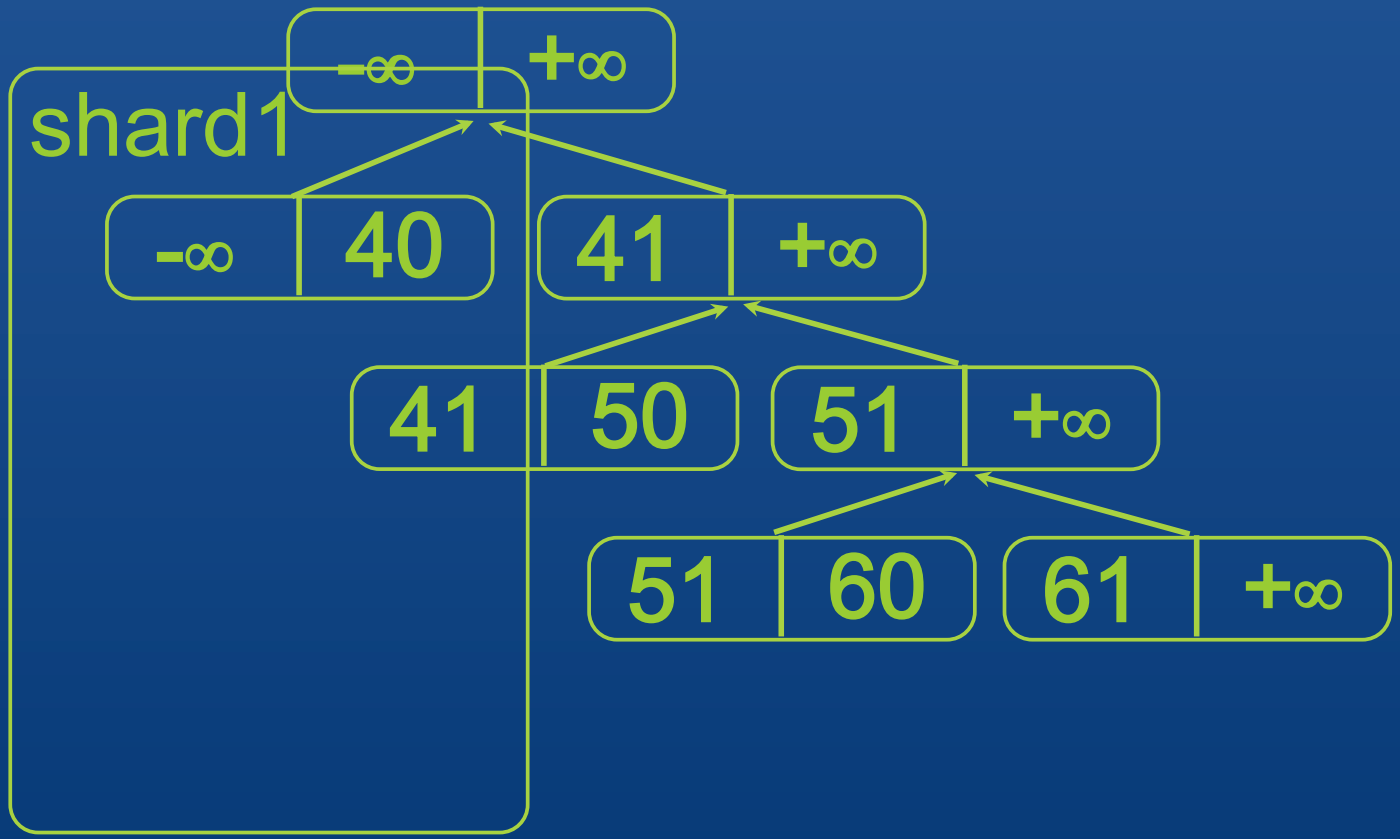
MongoDB 水平切分如何工作

```
> db.posts.save( {age:40} )
> db.posts.save( {age:50} )
> db.posts.save( {age:60} )
```



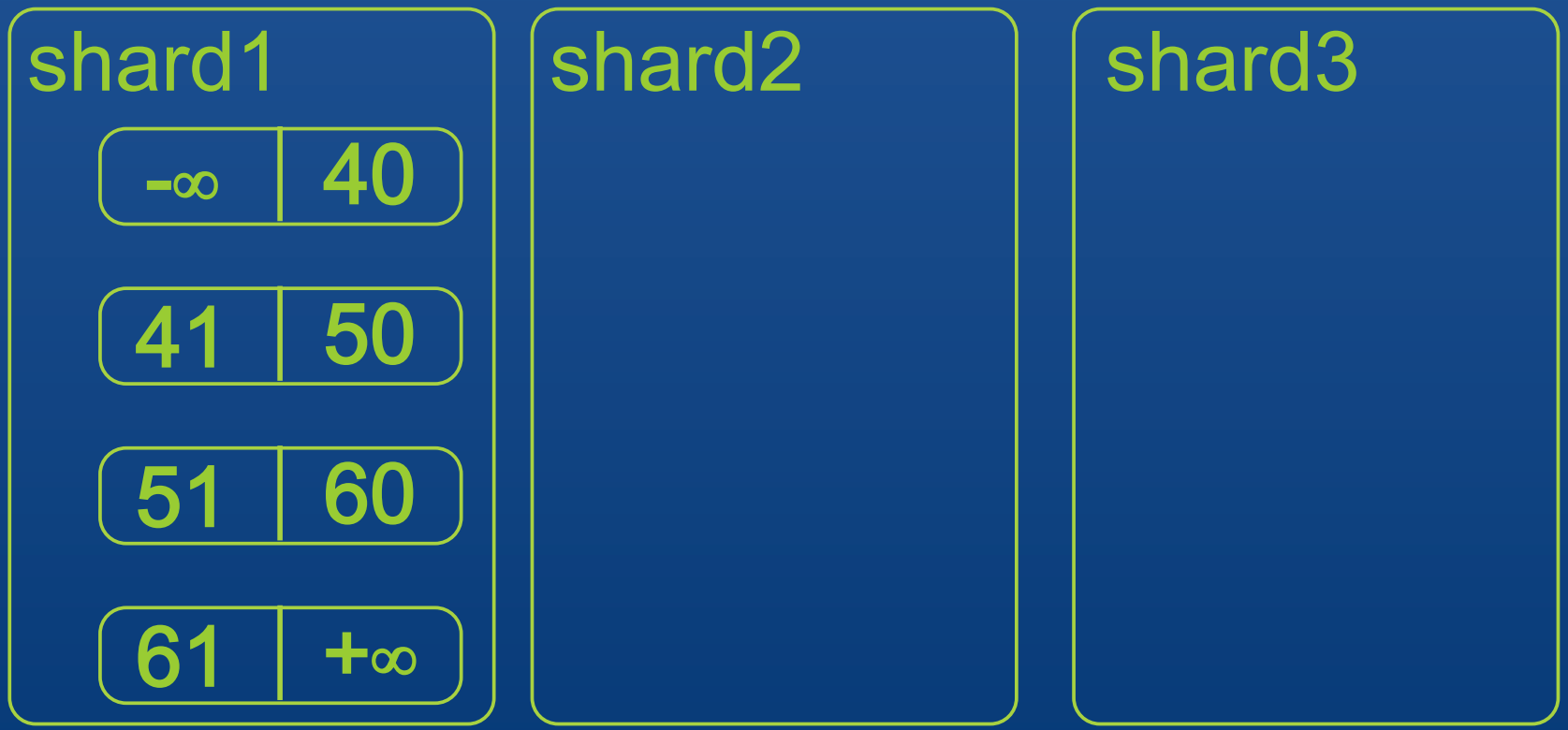
MongoDB 水平切分如何工作

```
> db.posts.save( {age:40} )
> db.posts.save( {age:50} )
> db.posts.save( {age:60} )
```



MongoDB 水平切分如何工作

```
> db.runCommand( { addshard :"shard2" } );
> db.runCommand( { addshard :"shard3" } );
```



水平切分键例子

```
{
server :"ny153.example.com" ,
application :"apache" ,
time :"2011-01-02T21:21:56.249Z" ,
level :"ERROR" ,
msg :"something is broken"
}
```

- 好：{server:1}
  - 一个服务器的数据都在一个块中
  - 块不能再细分
- 更好：{server:1,time:1}
  - 块可以毫秒分

水平切分键例子

```
{
server :"ny153.example.com" ,
application :"apache" ,
time :"2011-01-02T21:21:56.249Z" ,
level :"ERROR" ,
msg :"something is broken"
}
```

- 好：{time :1}
  - 时间是个越来越大的数字
  - 所有数据均将写至一个单一的水平切分
- 更好：{server:1,application:1,time:1}
  - 更多键值，使分开更容易

水平切分功能

- 不停机水平切分数据
- 写数据时自动平衡
- 命令被路由至（转至）正确的节点
  - 插入——必须有水平切分键
  - 更新——必须有水平切分键
  - 查询
    - 有水平切分键——被路由至节点
    - 无水平切分键——分散集中
  - 被索引的查询
    - 有水平切分键——按顺序被路由
    - 无水平切分键——分布式分类合并

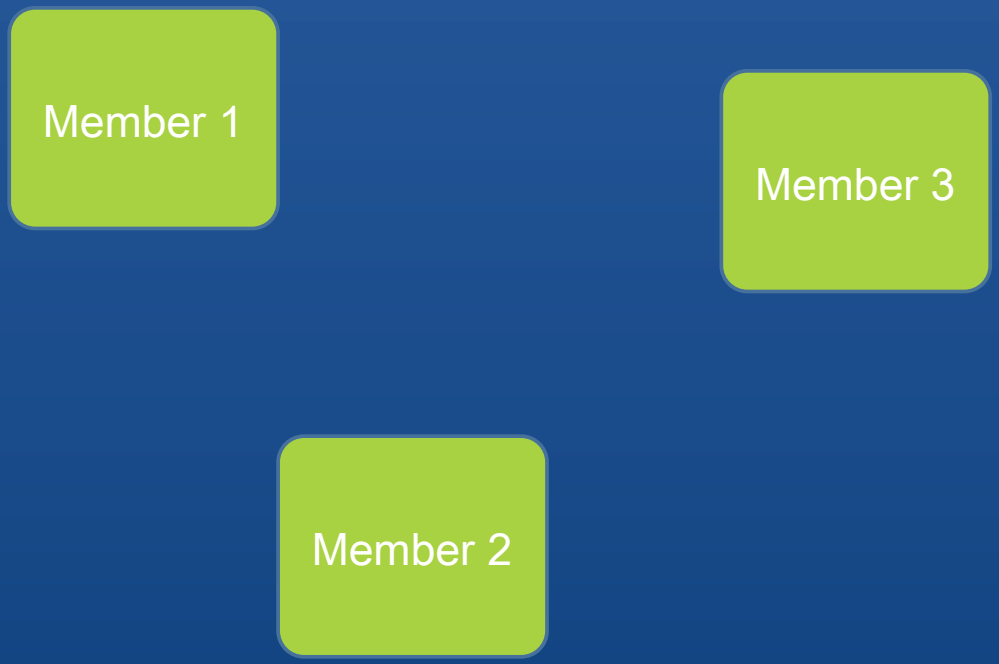
MongoDB 复制

- MongoDB 复制如 MySQL 复制
  - 异步主/从
- 变化：
  - 主/从
  - 副本集

副本集功能

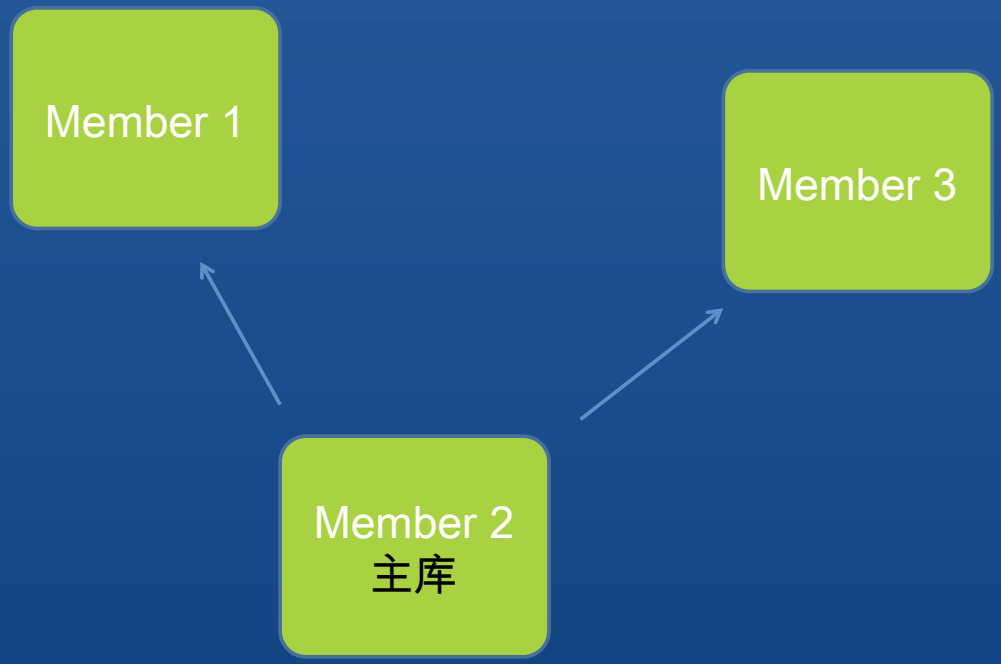
- N 个服务器簇
- 任何一个节点都可为主库
- 主库共识选举
- 自动容错移转
- 自动恢复
- 所有写至主库
- 读可以为主库（默认）或次库

MongoDB 复制如何工作



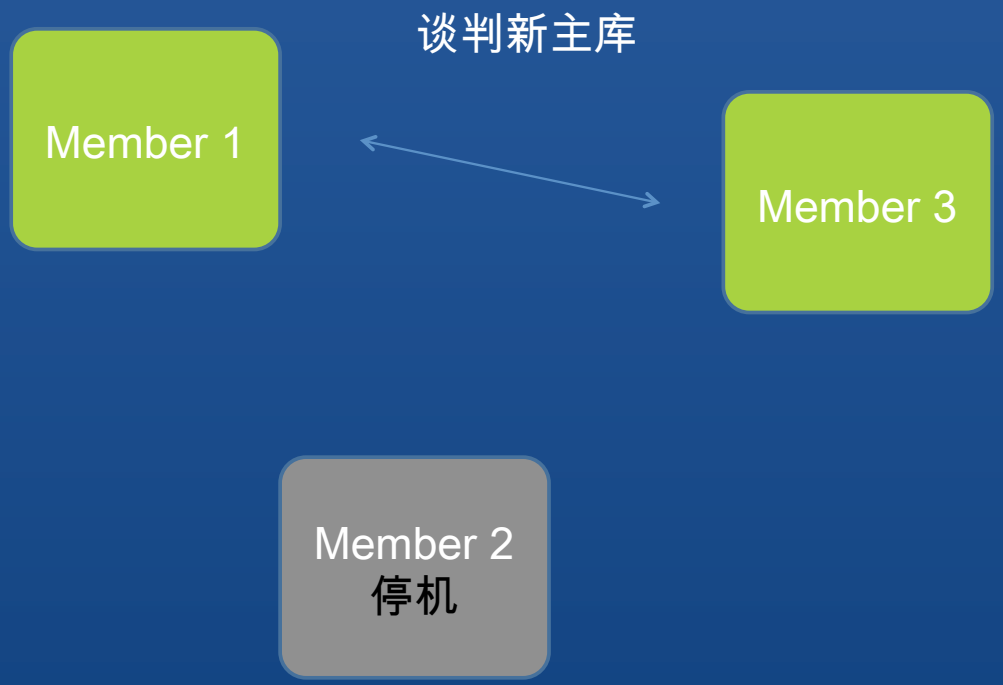
- 集由两个或更多个节点组成

MongoDB 复制如何工作



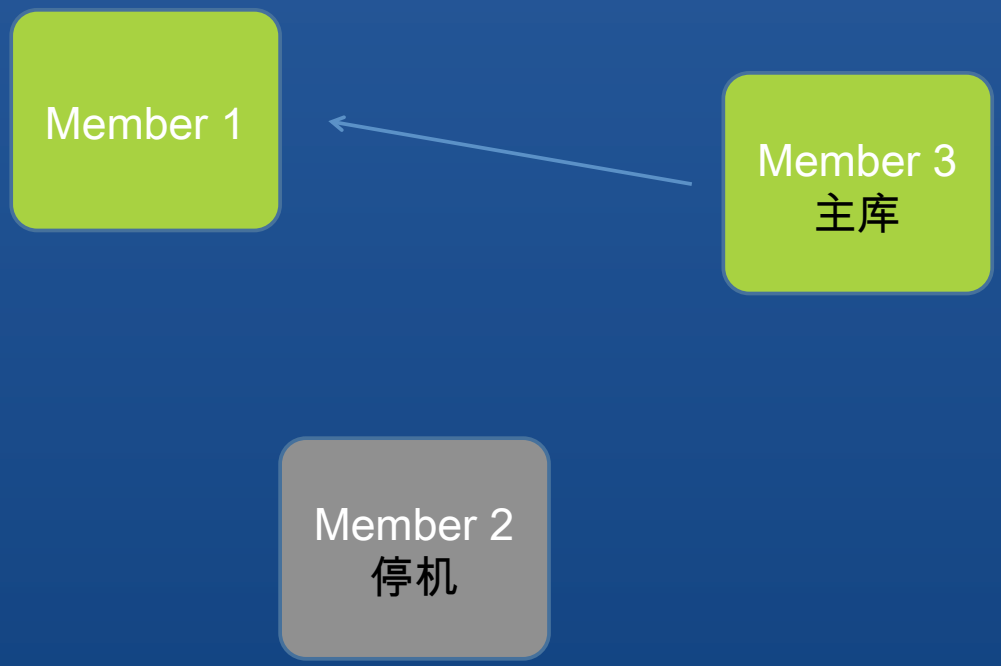
- 选举建立主库
- 数据复制从主库到次库进行

MongoDB 复制如何工作



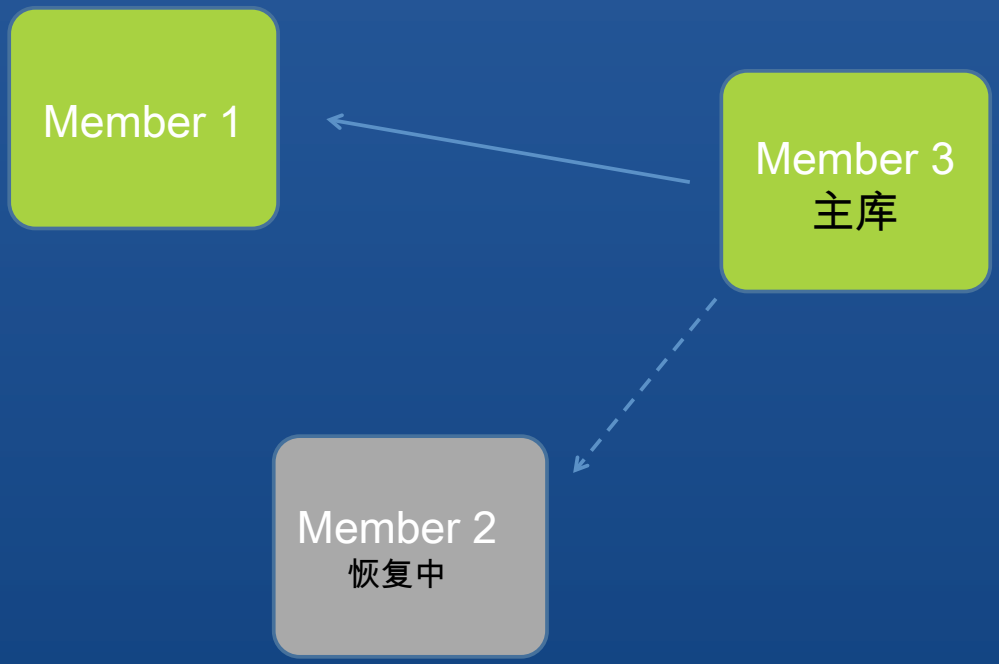
- 主库可能会出现故障
- 自动选举中新的主库

MongoDB 复制如何工作



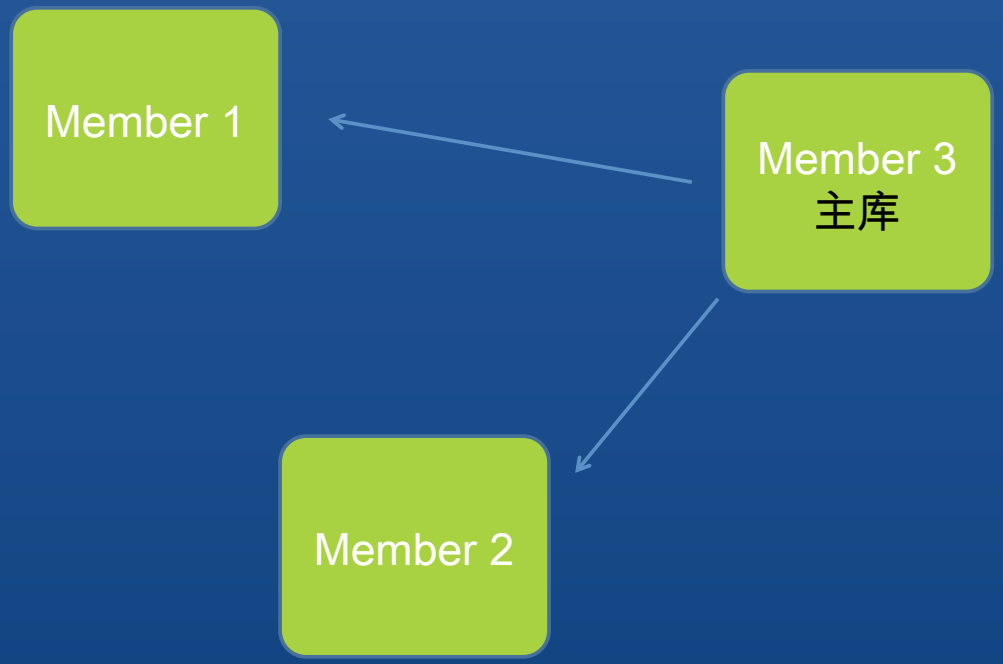
- 选出新主库
- 复制集重新建立

MongoDB 复制如何工作



- 自动恢复

MongoDB 复制如何工作



- 复制集重新建立

使用复本

- slaveOk()
- - 驱动将向次库发送请求
- - 驱动将总向主库发送写命令

- Java 示例
- - DB.slaveOk()
- - Collection.slaveOk()
- - find(q).addOption(Bytes.QUERYOPTION_SLAVEOK);

创建复本集

- > cfg = {
- ..._id :"acme_a",
- ... members :[
- ...{ _id :0, host :"sf1.acme.com" },
- ...{ _id :1, host :"sf2.acme.com" },
- ...{ _id :2, host :"sf3.acme.com" }] }
- > use admin
- > db.runCommand({replSetInitiate:cfg})

复本集成员类型

- 正常{priority:1}
- 被动{priority:0}
  - 不能被选举为主库
- 仲裁者
  - 可在选举中投票
  - 不持有任何数据

复制功能

- 从主库中读总是一致的
- 从从库中读最终是一致的
- 主库故障时自动容错移转
- 当节点加入集时自动恢复

概要

- 架构及索引设计
  - 最简单的伸缩方式
- 水平切分
  - 自动伸缩写
- 复制
  - 自动伸缩读

下载地址： mongodb.org

会议、出场和聚会
http://www.10gen.com/events

Facebook
http://bit.ly/mongofb

|

Twitter
@mongodb

|

LinkedIn
http://linkd.in/joinmongo

10gen mongoDB