

QCon beijing 2011

# 吃蟹

视觉中国的MongoDB应用实践



# About.me

- 潘凡 ( nightsailer) 视觉中国
- @nightsailer //twitter,sina,linkedin, github ...
- nightsailer # gmail.com
- <http://nightsailer.com/>

# 为啥用MongoDB

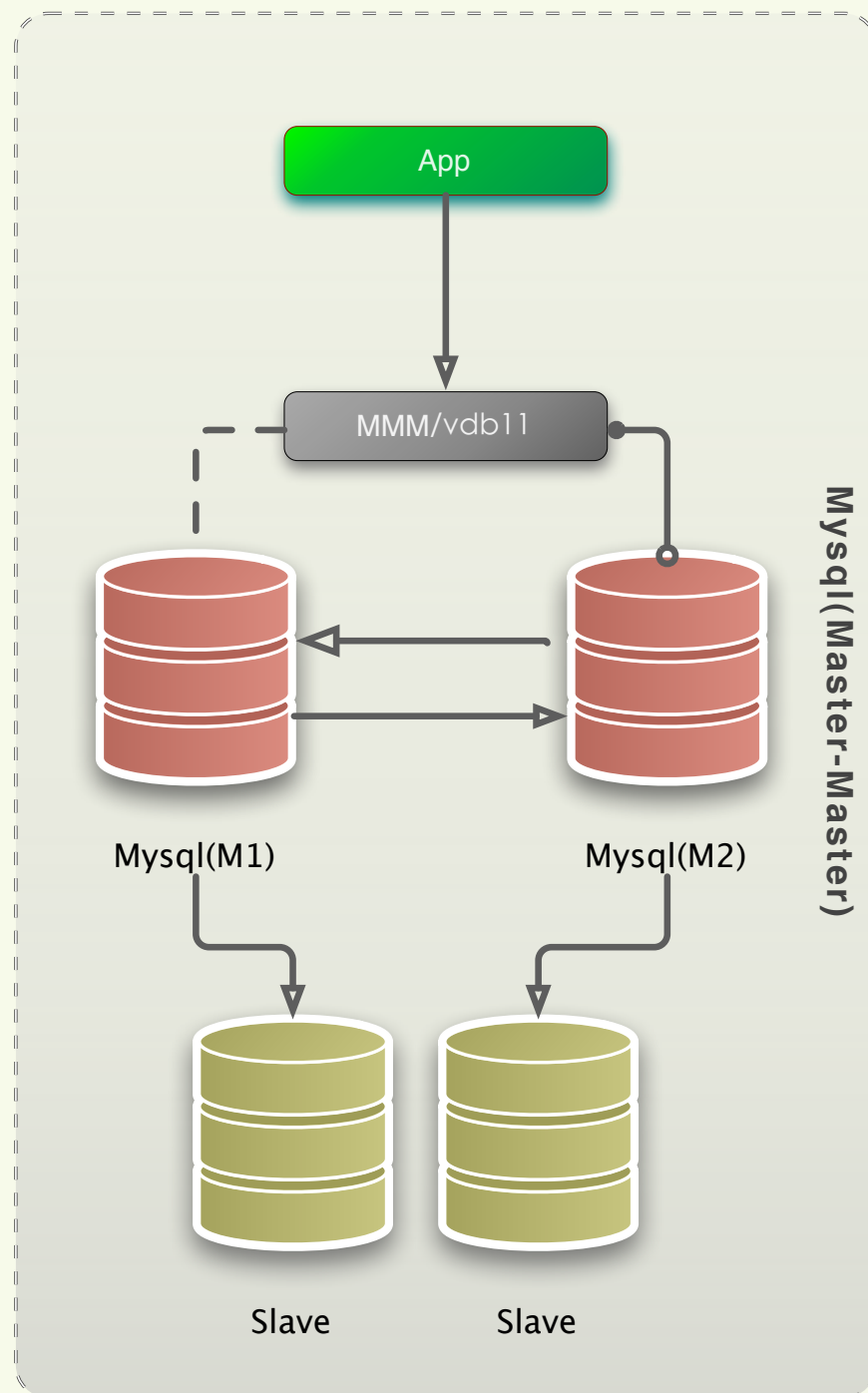
赶NoSQL时髦?

Auto-shard等激动人心的特性?

- No! 08年，还都是浮云

最初的想法是寻找一个可靠的分布式K/V  
解决MySQL的问题

# 原来的架构



- MySQL ( Percona) , Master-Master-Slaves
- HA:MMM

# 新要求

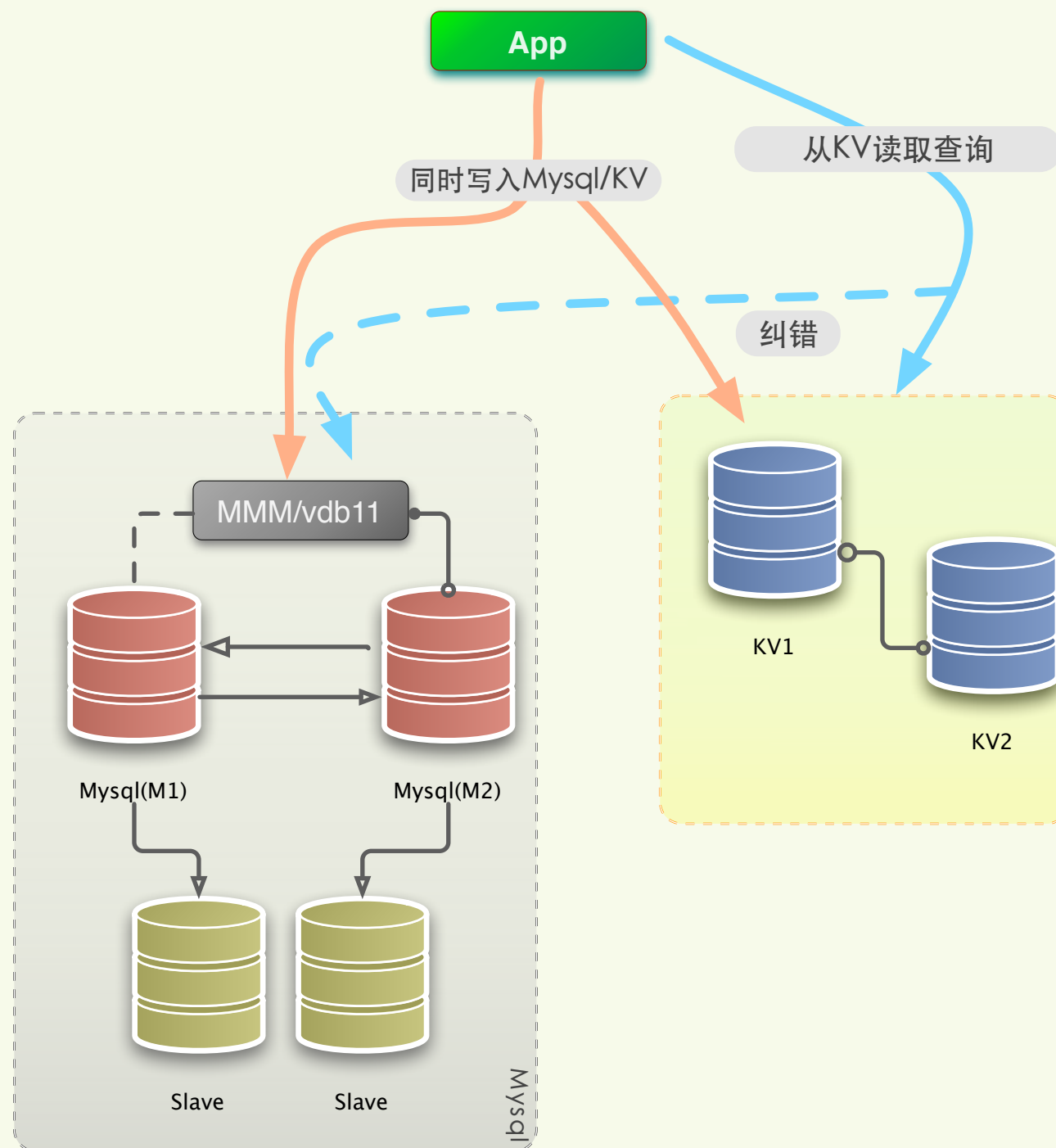
- 能够适应多数据源
- 需要灵活，不确定的schema
  - 不同模型的字段不定，属性不定
- 属性更新频繁
- 服务器等硬件资源有限

# 如何解决？

## 原有MySQL的方案

- \* 使用文本字段，JSON存储
  - schema自由度高
  - 更新容易，直接检索困难
- \* 使用关联表
  - schema自由度有限，类型控制
  - 更新频繁，query缓存失效

# 新思路



# 新思路

继续使用Memcached?

- 缓存失效，内存不足

决定：寻找Memcached替代品

- 能够持久化的分布式KV



# 选型条件

- 同时有PHP/Perl的良好客户端支持
- 性能和Memcached相差不要太多
- 支持分布式集群
- 低碳环保，节约资源
- 文档清晰，有成功案例

# 一些候选者

- Flare
- Repcached
- Redis
- TC/TT

# 最初的选择

## 选中Flare

- 内置cluster看起来很美好,可靠性有保证
- 使用Memcached协议，现有改动小

# 代价

项目开发1个月后

- 官方更新显示似乎并非如此可靠
- 水土不服：
  - 个人能力有限，无法解决一些灵异事件
  - 没有开发者社区
  - 不懂日文 ;-(

# 新的候选者

- Cassandra
  - 技术实力无法支撑，用不起
- CouchDB
  - 灵活，但性能口碑似乎不佳

# 重新选择

## MongoDB

- 读写性能中庸，比Redis逊色
- Document模型令人满意
- 内置操作没有Redis惊喜，基本够用
- MySQL类似的集群机制,上手方便
- 某些MySQL的优化部署经验可以复用

# 胆子再大一点

## MySQL + MongoDB?

- 多余：很多MySQL的操作MongoDB均可实现
- 同时维护 MySQL  $\Leftrightarrow$  MongoDB 的数据，代码逻辑有些复杂
- 人员流失，培训新人表示鸭梨大

# 胆子再大一点

能否彻底抛弃MySQL

- Transaction? 可以没有
- Joins? 原本少用，可以没有
- 数据一致性：不太高



# 胆子再大一点

好吧，试试

- 拿一个旧项目开刀
- 1人1个星期完成90%代码移植
- 原有35个table => 10 collection
- 开发过程很happy!

# MongoDB,就是它了

一举两得的加分项：GridFS

- 简单，符合我们的要求
- 无需再考虑分布文件系统
  - 放弃了原来的MogileFS，减轻运维压力

# MongoDB,就是它了

选对了

- SourceForge等一些大站应用增强信心
- 分享的信息逐渐丰富
- 10gen核心团队在mailing-list快速响应, 有问必答
- NoSQL开始受到追捧, MongoDB的口碑良好

# 应用案例 I

- 评论



```
comments:{  
  _id:ObjectId('xxx'),  
  art_id:2,  
  content: '呼呼 真是...',  
  replied_on: 12233  
  created_on: 12222  
  replies: [{  
    _id: ObjectId('xxx'),  
    content: '哈哈, 谢了...',  
    replies:[]  
  }]}  
}}
```

# 应用案例2

卷发女孩 80总分, 6652次浏览, 评论, 收藏



- ♥ 被LXtiramisu收藏了 1小时, 32分钟前
- ♥ 被ssciwei收藏了 10月21日 23点00分
- 💬 lulu,姚 发表了新评论 10月9日 09点40分  
哦, 好想Popeye
- ♥ 被xzz1016收藏了 8月13日 17点50分



trueliesfans

- 🖼️ 上传了作品 White Rabbit 3小时, 22分钟前
- ♥ 开始关注傅纯强 3月10日 11点23分
- ♥ 开始关注傅纯强 3月10日 11点23分
- 💬 评论了傅纯强的作品 春花秋月 3月10日 11点23分



大声旺旺

- ★ 对朱小强的作品 练习板评分(10) 6小时, 51分钟前
- 💬 评论了大声旺旺的作品 四眼加四个眼镜 3月7日 16点44分
- 🖼️ 上传了作品 画不好的树和房子 3月2日 14点50分
- 🖼️ 上传了作品 四眼加四个眼镜 3月2日 14点50分

## 用户动态/新鲜事:

- 关注的用户的动态
- 个人作品的新动态
- 关注的作品的新动态  
(评论过/收藏过/打分过...)
- 可取消对特定事件的订阅

# 应用案例2

## 推送模式的实现

- `db.activity_stream.feed` (订阅事件)
- `db.activity_stream.user` (用户动态汇总)
- `db.activity_stream`(动态详细信息)

# 应用案例2

## feed的优化

- 区分热门用户和普通用户
  - 热门用户的follower使用独立的collection
  - 普通用户使用embed list

# 应用案例3:全文检索

## 尝试1: Regex

- 无索引，慢！

## 尝试2: Sphinx

- 手动生成xml，刷新重建索引
- 无法增量索引



# 应用案例3:全文检索

尝试3: 分词 + Array/List 查询(\$all)

- 简单，够用
- 分词: PHP-SCWS (谢谢! )
- 增量索引:
  - 无侵入索引过程: 从oplog读取, 解放应用端

# 准备部署

## 手动编译MongoDB

- Scons, 要懂点Python
- 升级boost(CentOS)
- static link mongod
- 使用tcmalloc

# 准备部署

- 手动编译MongoDB( ICC编译选项)

```
COMMON_CXXFLAGS='-fp-model source -unroll2 -axSSE4.1,SSE4.2 -xSSE3 -static-intel -fpic -fno-strict-aliasing'
```

```
CXXFLAGS="-O3 -ipo -static-libgcc $COMMON_CXXFLAGS"
```

```
scons --release --static --extrapath=/opt/local --cxx=$CXX --icc --extralib='tcmalloc_minimal' --icc-cxxflags="$CXXFLAGS" --icc-cppflags="$CPPFLAGS" -c $BIN_SERVER
```

```
scons -j4 --release --static --extrapath=/opt/local --cxx=$CXX --icc --extralib='tcmalloc_minimal' --icc-cxxflags="$CXXFLAGS" --icc-cppflags="$CPPFLAGS" $BIN_SERVER
```

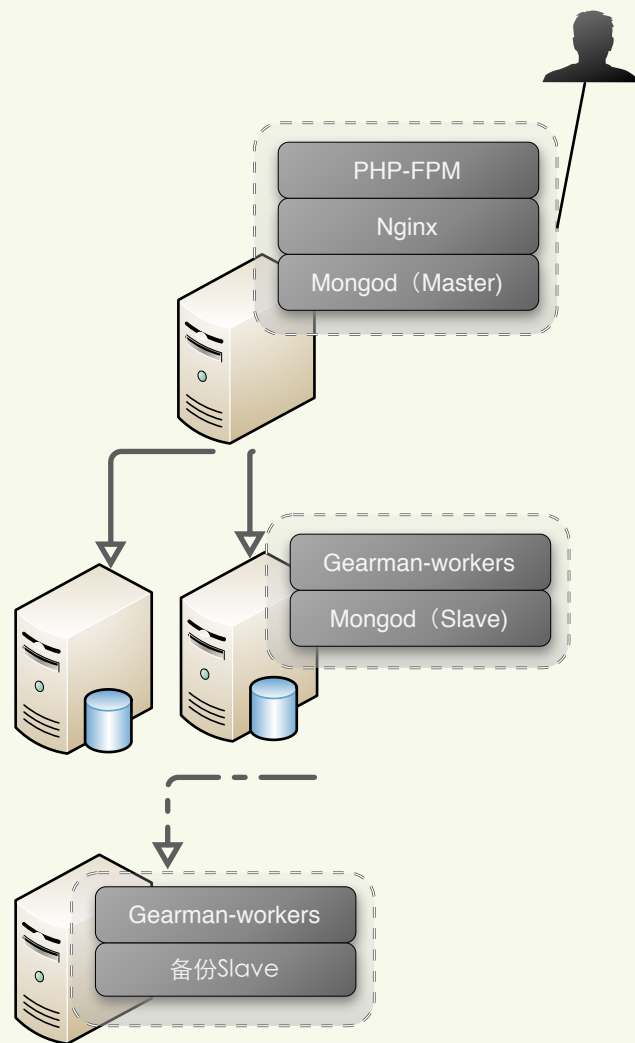
# 准备部署

- 定制编译麻烦，但是相对稳定，问题很少（不仅仅是心理安慰）
- 缺点：无法及时更新编译，只针对性的升级

# 准备部署

- 磁盘： 使用Raid10
- 文件系统：
  - XFS
  - Ext4 (?)

# 部署



开始, 2009/6, 版本: 0.9/1.0

- 1 Master + 2 slaves
- 1m 初始数据
- 20g存储
- Dell 2850/4g(Master) + 2\*Dell 2950/4g(公用)

# 备份

\* 使用独立Slave方式备份

- 慢，不影响

\* lvm snapshot

- 快，客户端瞬时无法写
- fsync & lock db

\* mongodump?

- 速度慢 + 空间浪费 + 恢复慢

# 监控

- mongostat / vmstat / iostat
- collectd（我们使用这个）
  - 自定义: json-rest+perl plugin
- 其他: Munion / Nagios ...



# 美好的时光

MongoDB不是重口味

- CPU友好，几乎无负担
- 内存需要足够放下索引，4G以上为佳
- 可以和磁盘IO不高的应用混搭



美好的时光都是短暂的，6个月后

开始有点问题了



# 空间使用率

磁盘空间不足，纪录没有显著增长，  
Why?

- 应用执行操作：Map/Reduce删除/重新插入
- 删除空间不能回收（早期版本）
- 可复用但频繁更新导致碎片（1.6+）
  - 检索时间加长

**\*\* 定期reIndex或repairDatabase,**

# 故障0:开发版的血案

## \* MongoDB 宕机

- Out of Memory!
- 新版本v1.3某些情况出现内存泄露

## \* 服务端cursor过多

- Perl driver的bug, 修复

# 故障0:开发版的血案

## 教训1:

- 永远别用开发版，即使天大的诱惑
- 谨慎使用稳定版的头2个小版本

## 教训2:

- 选用了—个语言，团队内就要有人熟悉对应的driver代码
- 自己实现—个driver是个不错的想法

# 故障 1:50x 的烦恼

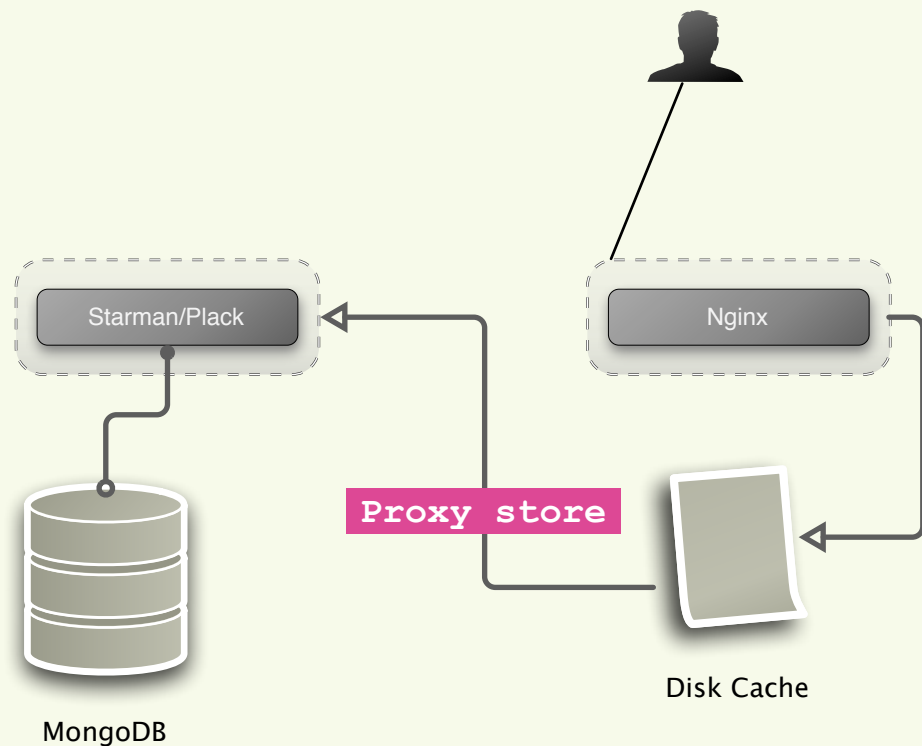
## 502 Bad Gateway

### \* GridFS 的问题?

- 使用 Perl Plack 替换 = 》好转

### \* 减少 GridFS 的读取

- Nginx proxy\_store
- 不是最优，但简单，够用



# 故障 I: 50x 的烦恼

## 504 Gateway timeout

- 灵异：开发环境无法重现
- 无规律，白天或晚上均会出现
- Perl救场：根据nginx\_errorlog的时间抽取前后5分钟内mongodb.log日志
- 找到元凶：空间分配延时导致client端超时

# 故障 1:50x 的烦恼

## 教训3

- 使用XFS等支持pre-allocation的FS
- 提前预分配数据文件：

```
for i in {1..50}
do
    echo $i
    head -c 2146435072 /dev/zero > $db.$i
done
```



# 故障2: 空间使用

- Mongod crash
- 磁盘空间不足, Why?
- 应用执行操作: Map/Reduce删除/重新插入
- 删除空间不能回收



# 第一次灾难

- 机房电源空开跳闸
- 10台服务器全部遇难
- MongoDB无法启动
  - 尝试Repair~5小时后失败
  - 从备份中恢复
  - 丢失了10小时数据！！



# 第一次灾难

- MongoDB的致命缺陷：
  - 单机可靠性低
  - cluster可靠性: 通过slave复制保证
  - 部署时没有考虑机房断电的特殊情况

# 亡羊补牢之I

- 缩短磁盘刷新间隔 `--syndelay`
  - 60s(default) => 15~30s
  - 磁盘IO受到影响，可接受
- 应用端批量写入后调用`fsync`
  - 写入速度降低

# 亡羊补牢之2

- 升级到1.6.3,转换Master-Slaves到RelicaSets
  - 1 Primary + 2 Secondary
- 服务器升级到4g-8g内存
- 应用端同时使用w=2参数
  - 写入速度降低



# 教训5

- 我们很业余
- 安全第一
- 要考虑极端条件下的容灾和数据安全
- 完整的离线备份总是你最后的救命稻草



然而，还没有结束。。。。





# 故障4: RS fail-over失效

- MongoDB 不可用
- Primary维护期间不小心kill了第2个 secondary
- 剩余的secondary未能正常接管

# 故障4: RS fail-over失效

解决:

- 增加2个 Arbitor
- `$ mongod --bind_ip 127.0.0.1,192.168.8.10 --replSet rs10 --oplogSize 1 ...`
- `> rs.addArb('192.168.8.10:27020')`
- ...

# 故障4: RS fail-over失效

教训6: 这才是完整的ReplicaSet

- 1 Primary + 2 Secondary + n Arbitor
- 文档中语焉不详的往往是你最被忽视的，把它搞清楚
- 避免手动维护，使用事先写好的脚本

# 又一次断电了！

- 6个月后，“又”一次出现断电
  - 这次是修空调拉错闸；-(
  - 临时工不好当
- 重启后，ReplicaSet集群fail-over到一个secondary，中间过程无干预
- Primary 无法启动,仍需要修复！
  - 还好！

# 1.8的改进

- 1.8提高了单机可靠性的一个措施
  - journaling file
  - mongod -dur
  - 避免crash后必须repairDatabase
- 初步测试，效果尚可，准备稳定后升级

# 有了新需求

- GridFS的新需求
  - 能够存储较大文件（高清图片，压缩包）10mb-500mb不等
  - 允许用户使用多线程下载

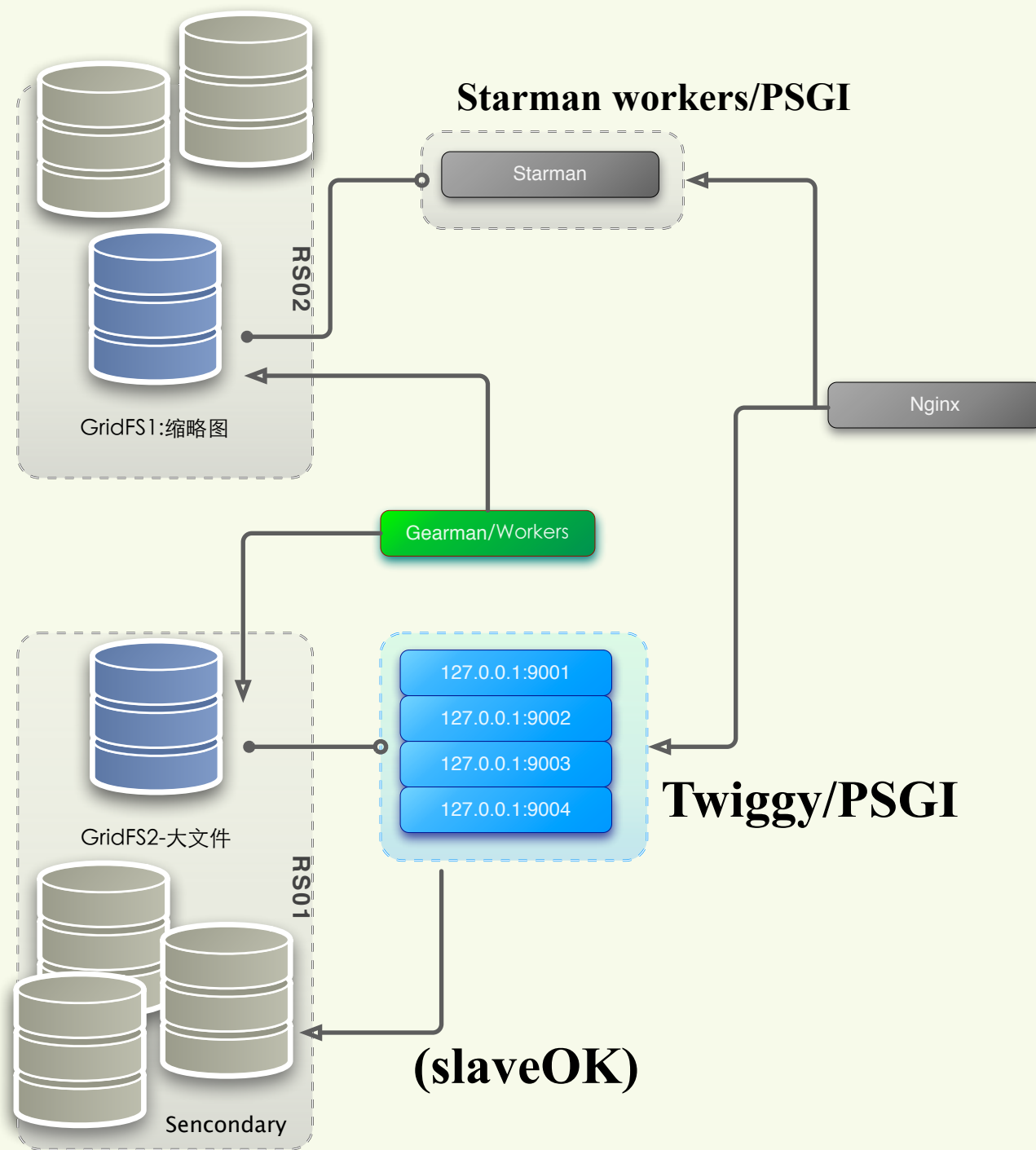
# 问题

- GridFS大文件的分发
  - 不能使用Nginx proxy\_store加速
- MongoDB读取负担加大
- 并发链接增加,Plack app响应速度下降
  - prefork模式有限制

# 措施

- GridFS文件分库
  - 缩略图， 高清图片 + 大文件
- 读写分离（slaveOK）
- 将Plack app部署到Twiggy（AnyEvent），解决高并发链接的问题

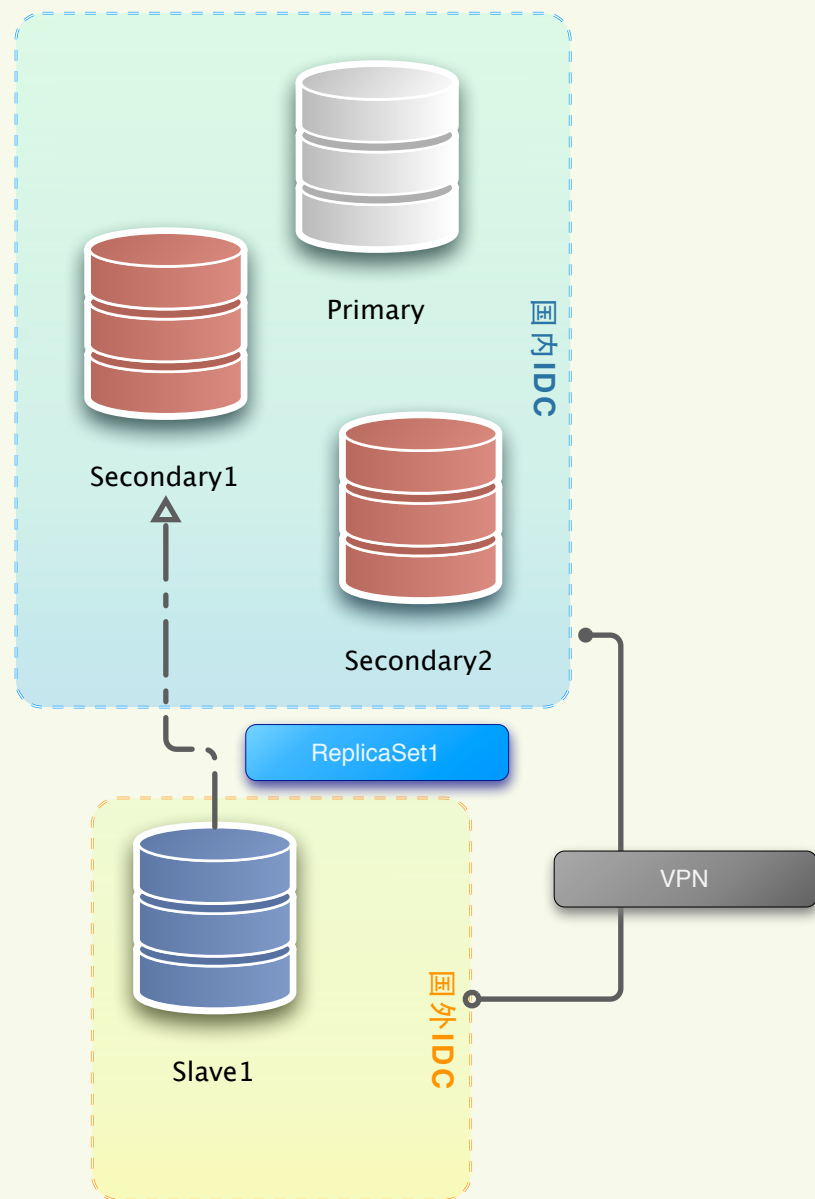




# 带宽不够用了

- 下载太消耗带宽
- 国外的服务器和带宽都便宜
- 怎么同步？

# 理想



- 国外使用一个Slave only的MongoD节点
- `priority = 0`
- VPN链接2个数据中心

# 现实

- 超高的网络延时 >300ms
- MongoDB 启动失败，无法完成初始化

# 折腾

- 解决MongoDB延迟启动
- 将snapshot手动复制到国外
- 手动作复制工作
  - 监控local.oplog.rs (使用tailable cursor)
  - 压缩后发送到国外节点
  - 接收端接收后replay oplog

# 失败!

- 运行了2个月，暴露的问题：
- GridFS数据传输较大，复制延迟时间较高
- 墙让VPN不时抽风

# 调整

- 放弃集群和VPN
- 将GridFS本地导出压缩后传送到国外
- 仅仅将需要更新的数据更新到国外
  - BSON压缩，使用HTTP协议 + 多点续传
- 国外使用普通的mongod

# 教训7



- 小心陷入思维定式
- 太华丽 == 不现实
- 不要忽视最土鳖的做法
- 灵活运用不同的unix工具



# Auto-sharding

- 1.6GA 内置了Auto-sharding
  - 有诱惑，尝试
- 2套方案并行1个月
  - 从原有生产系统导入数据
  - 并行写入测试集群

# 测试结果

- Shard\_key的烦恼
  - 单一shard key导致chunk分配不均衡
    - 类似4sq案例
  - 复合shard\_key需要修改现有的应用端
- counting 不准
  - 务必减少chunk的移动
- balancer的稳定性，也不够智能

# 测试结果

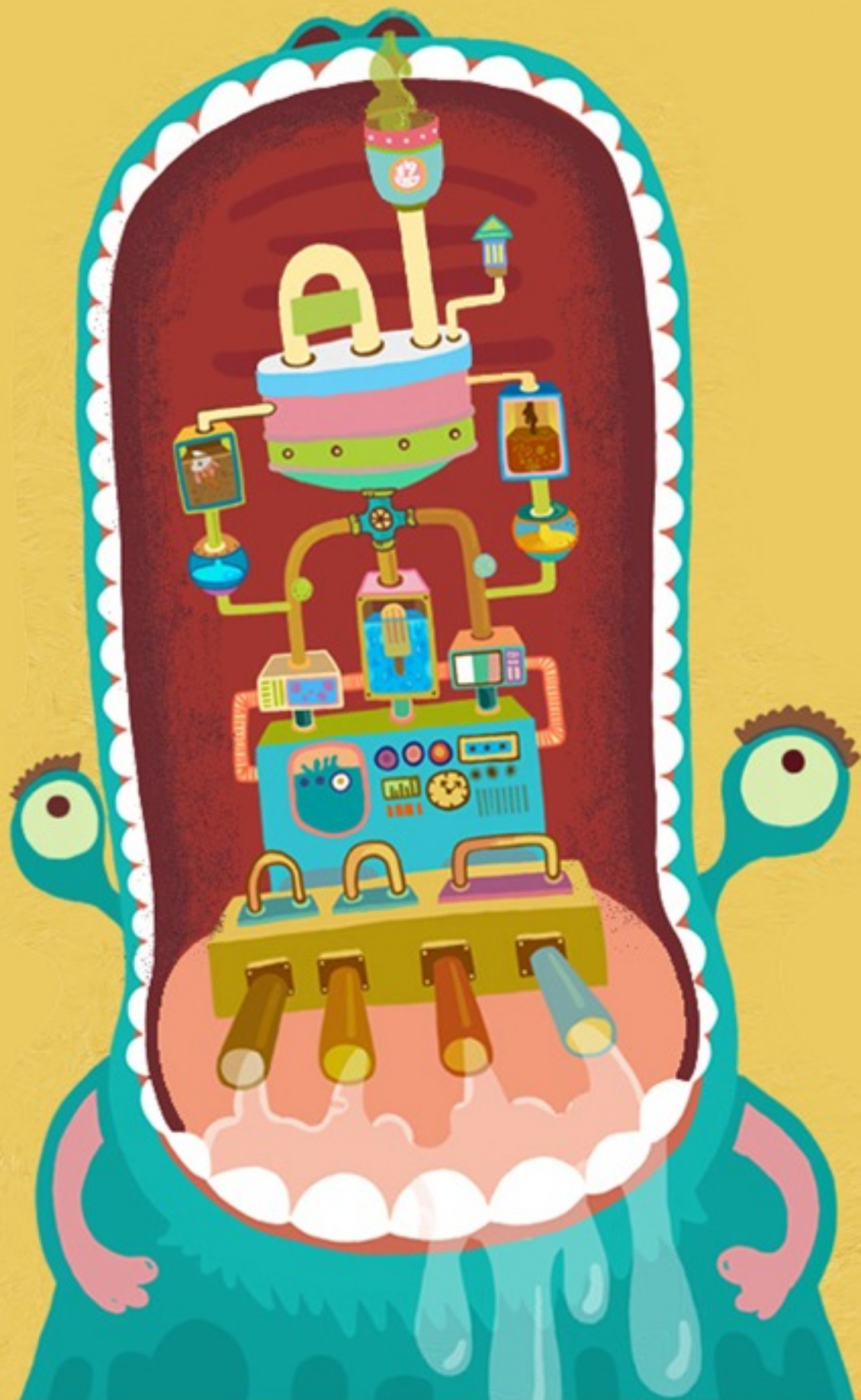
- 结论：
  - 从现有非shard环境升迁是很痛苦的过程
  - 如果最初就是shard，相对容易
  - 维持现有的手动分片机制
  - 等待1.8成熟，新项目可以考虑使用

# 无责任小结

- 市场宣传亮点和实际会有差距
- MongoDB的auto-shard目标感觉很好
- GA和完全成熟可靠多少还有些差距
  - 2.0+ ?

# MongoDB其他小结

- 多数MySQL的Web应用均能胜任
- Schema free，大大提升开发效率
- Geo搜索降低位置服务门槛
- MySQL优化的一些经验可以复用
- GridFS能够胜任中小规模的文件系统
- 千万级别的数据量完全无需做sharding
- Auto-sharding注意shared\_key和balancing的过程，建议1.8/2.0以后使用



Question?

谢谢大家！



The QCon logo features a large, stylized letter 'Q' in a vibrant green color, followed by the letters 'Con' in a bold, blue sans-serif font. The background of the entire slide is a photograph of a traditional Chinese stone pagoda with multiple tiers and circular openings, situated in a body of water with a hazy cityscape in the distance.

# QCon

杭州站 · 2011年10月20日~22日

[www.qconhangzhou.com](http://www.qconhangzhou.com) ( 6月启动 )

QCon北京站官方网站和资料下载

[www.qconbeijing.com](http://www.qconbeijing.com)

全球企业开发大会

THE ANNUAL  
INTERNATIONAL  
SOFTWARE DEVELOPMENT  
CONFERENCE