使用MongoDB助力DevOps

张耀星，首席咨询顾问，MongoDB

什么是DevOps？

DevOps背景

- DevOps = development + operation
- 始于2009年O'Reilly性能与运维大会："10+ Deploys per Day: Dev and Ops Cooperation at Flickr"
- 随后由Patrick Debois组织了第一次DevOpsDays
- DevOpsDays大受欢迎，发展为定期举行的Event
- Twitter和各种论坛上次之简称为DevOps

DevOps植根于敏捷

1. **个体和互动** 高于 流程和工具
2. **工作的软件** 高于 详尽的文档
3. **客户合作** 高于 合同谈判
4. **响应变化** 高于 遵循计划

尽管右侧项目有其价值，我们更重视右侧项目的价值

——敏捷宣言，2001

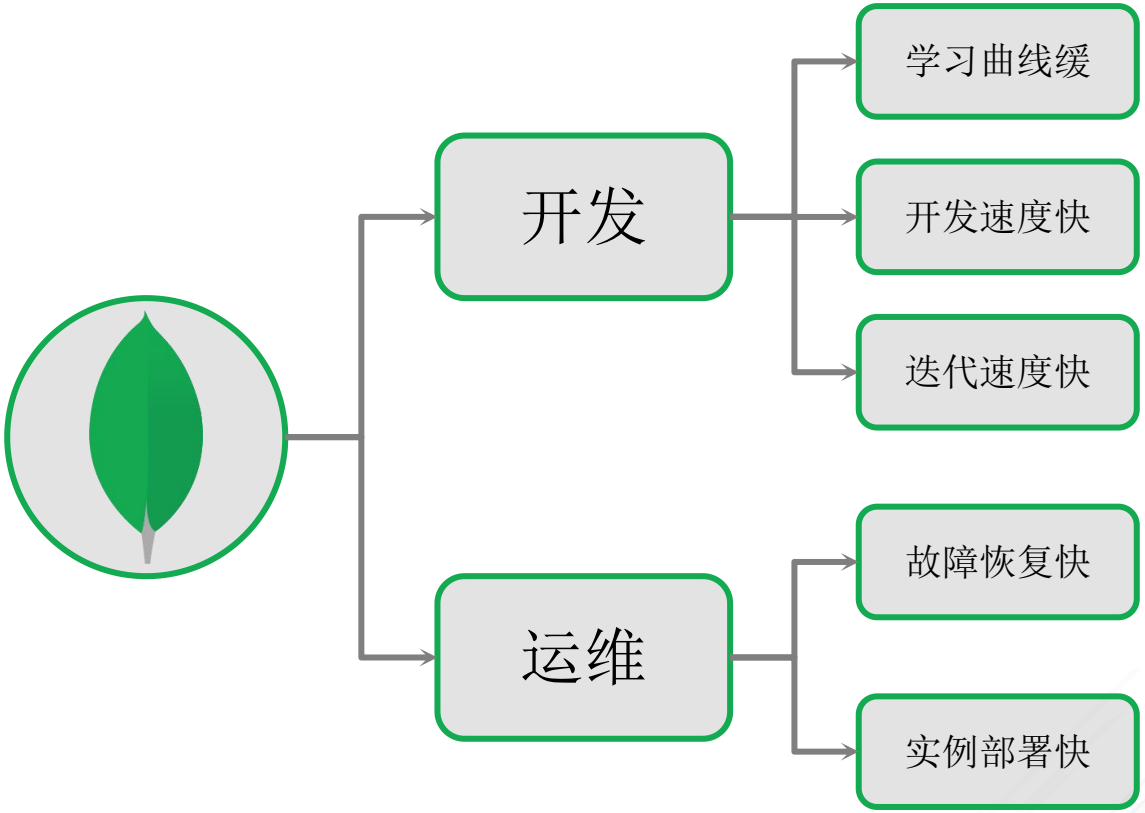
http://agilemanifesto.org/

DevOps的终极目标

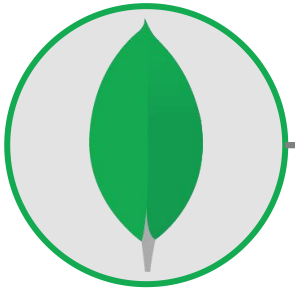
- 在不牺牲稳定性和质量的前提下，加速开发过程；
- 快速部署，快速试错，快速恢复；

MongoDB如何助力DevOps？

快



自动化



- 开发
  - 自动错误重试
  - 自动选择节点
  - 自动分发请求
  - 自动加解密*
- 运维
  - 自动数据均衡
  - 自动负载均衡
  - 自动故障恢复

案例一：世界500强保险企业

需求

该公司需要一个电话中心，客服需要能够快速查询接入客户的全部信息，减少客户等待时间。为了达到这个目的，需要整个70+历史系统中的客户信息，通过唯一入口查询

难点：

- 来自60多个国家的9000多万用户存在于70+已有系统中，这些数据需要汇集到一起；
- 已有系统在不断迭代，导致最终数据模型不断变化，关系数据库处理这种情况时异常艰难；

尝试

使用关系数据库的尝试：

- 使用DB2作为中心数据库汇集数据
- 历时2年
- 花费$2500万

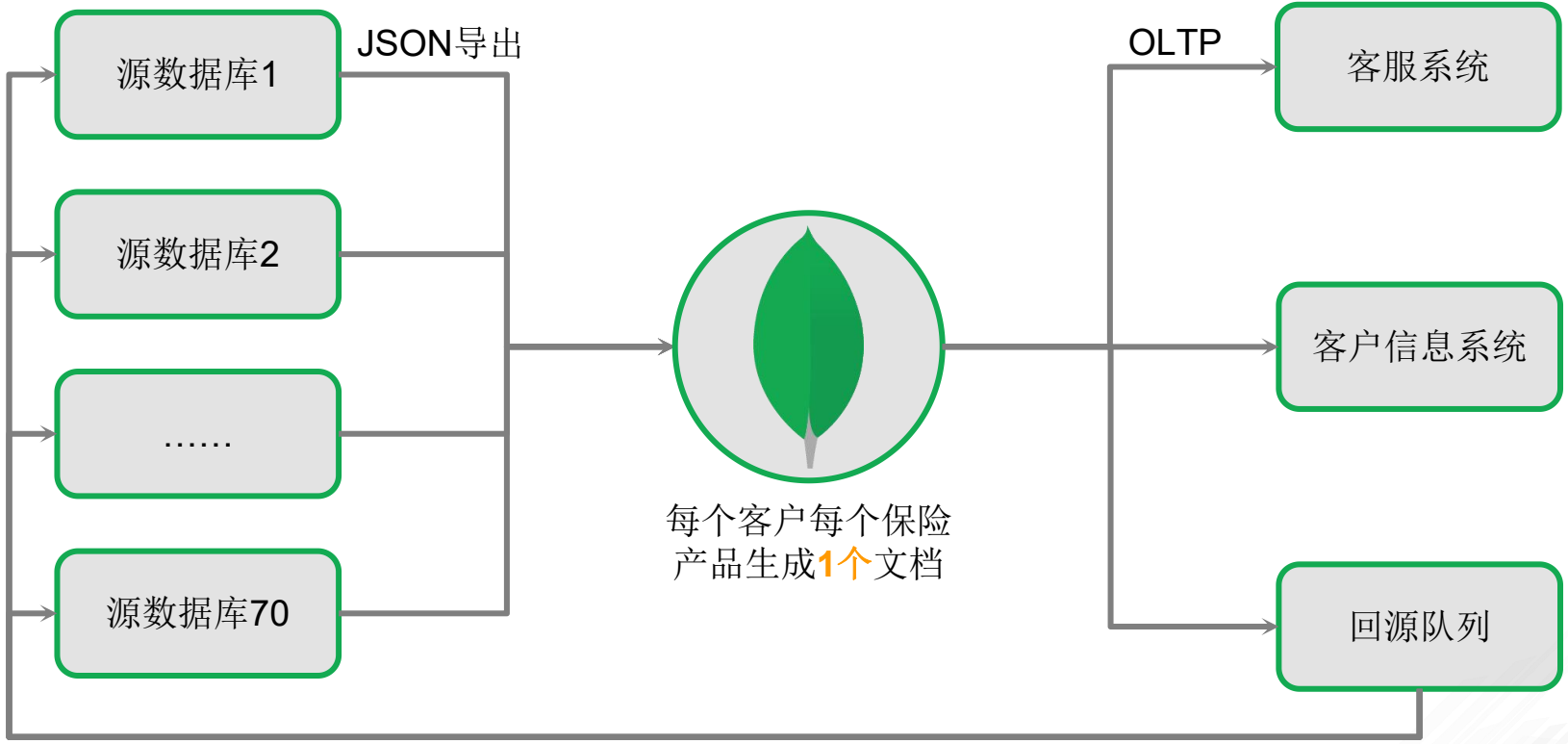
结果：失败

使用MongoDB的尝试：

- 动态数据模型轻松接收不同数据
- 7x24小时高可用
- 与Hadoop完美结合完成分析需求

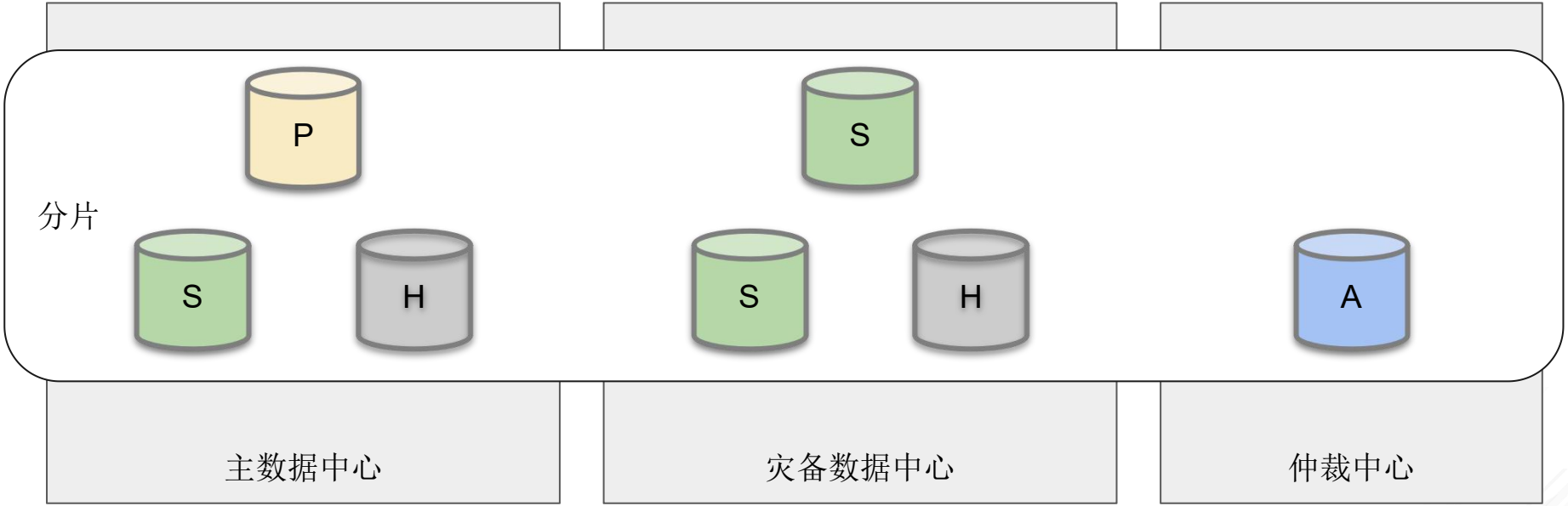
结果：成功

系统架构一



mongoDB.

系统架构二



案例总结

为达成目标，本案例中利用了MongoDB的以下特性：

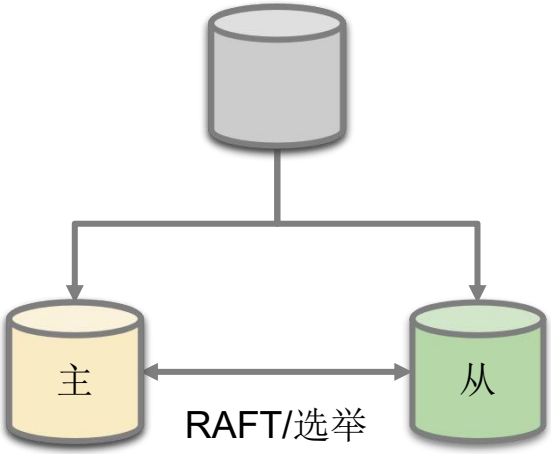
- 高可用（本地和跨机房）
  - 故障发生时应用可以自动切换到正常的节点上
  - 可以在秒级时间内完成故障转移，使得用户体验得到保证
- 反范式数据模型使得数据整合成为可能

因为这些特性的存在，使得实现需求非常容易。该项目2周完成原型，90天部署到生产环境，完美诠释了DevOps快速开发，快速迭代的要求。

相关特性——高可用

高可用

- 自动完成分布式节点故障时的自动恢复
- 在秒级时间内完成恢复流程



相关特性——反范式模型

灵活数据模式

- 无须上线前繁琐的模式变更
- 令开发更加自由发挥，加速开发过程

```
{
  name: "Yaoxing Zhang”,
  title: "Principle Consulting Engineer",
  location: "Shenzhen, China",
  email: "yaoxing.zhang@mongodb.com"
}
```

```
{
  name: "Yaoxing Zhang”,
  title: "Principle Consulting Engineer",
  location: "Shenzhen, China",
  email: "yaoxing.zhang@mongodb.com",
  startDate: ISODate("2016-02-05")
}
```

案例二：国内四大行

需求

为提升用户体验，该银行要在手机银行APP中支持实时账户交易历史查询。涉及的数据包括：

- 借记卡交易历史
- 信用卡交易历史
- 后续还将支持股票、基金账户等

对这些交易历史进行整合，使用户可以看到自己账户的交易历史全貌。涉及的交易数据量：

- 约6000万交易数据/天，结息日达到4.8亿/天
- 历史存量数据3年，共657亿

尝试

使用Oracle的尝试：

- 超大量数据需要巨大的Oracle实例，如果再考虑高可用，成本极高；
- 分库分表造成开发难度大幅上升；
- 整合不同账户数据时表结构差异大，合并困难；

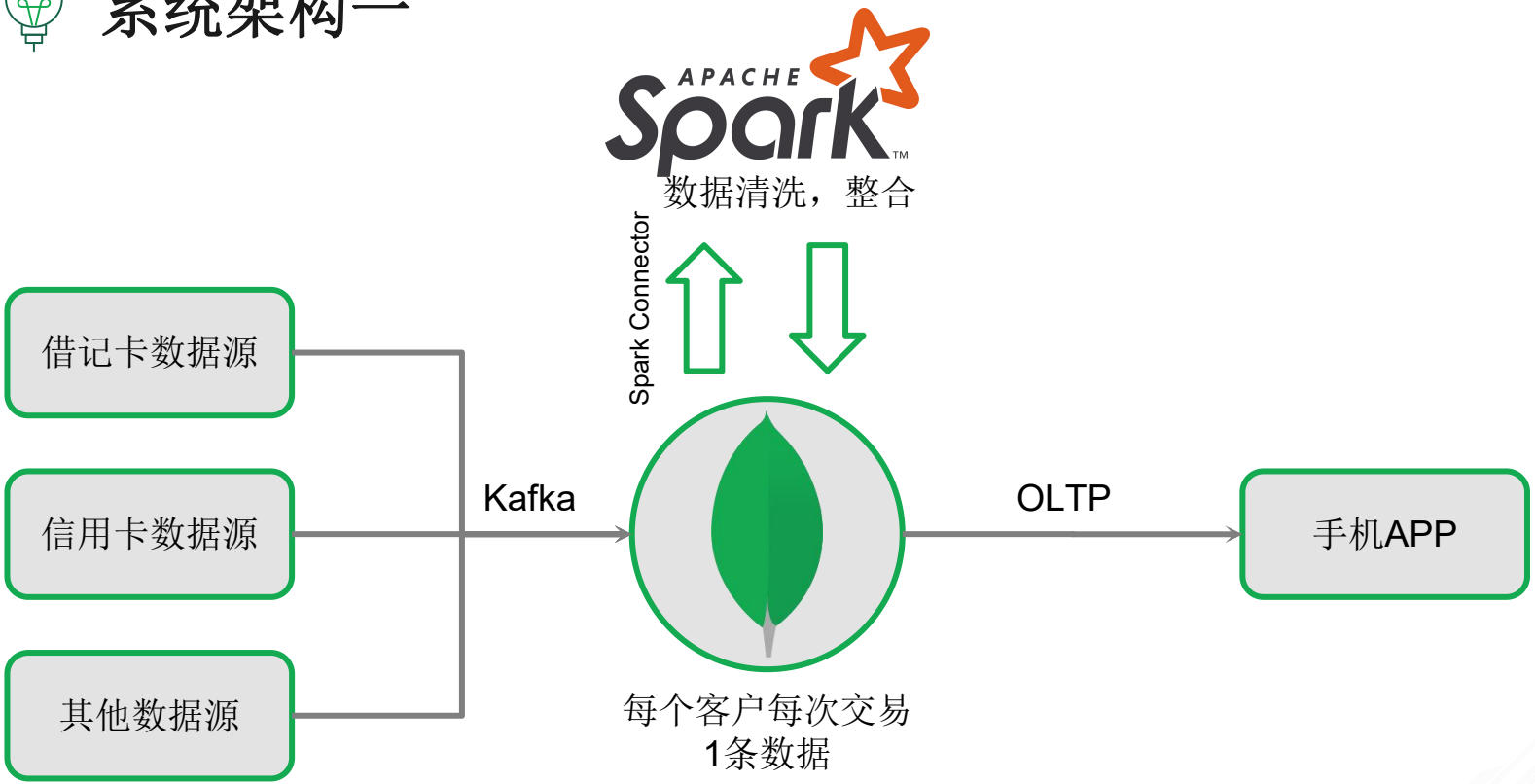
结果：评审期被否定

使用MongoDB的尝试：

- 动态数据模型轻松接收不同数据
- 水平扩展解决大数据量问题
- 7x24小时可用

结果：成功上线

系统架构一


APACHE Spark
数据清洗，整合
Spark Connector
借记卡数据源
信用卡数据源
其他数据源
Kafka
OLTP
手机APP
每个客户每次交易
1条数据

案例总结

为达成目标，本案例中利用了MongoDB的以下特性：

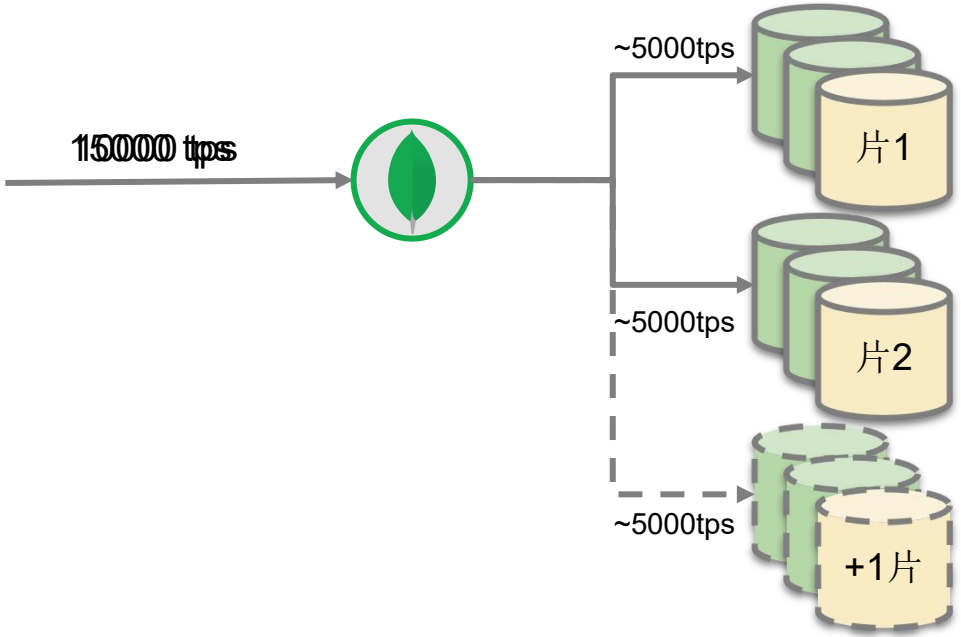
- 高可用：满足银行应用的SLA要求
- 反范式数据模型：使数据整合更为容易
- 弹性扩展：使得海量数据存储成为可能

以上特性使得开发人员可以将更多精力放在业务逻辑上，而不是如何分库分表等业务无关的问题上，从而加速开发过程。

相关特性——水平扩展

水平扩展

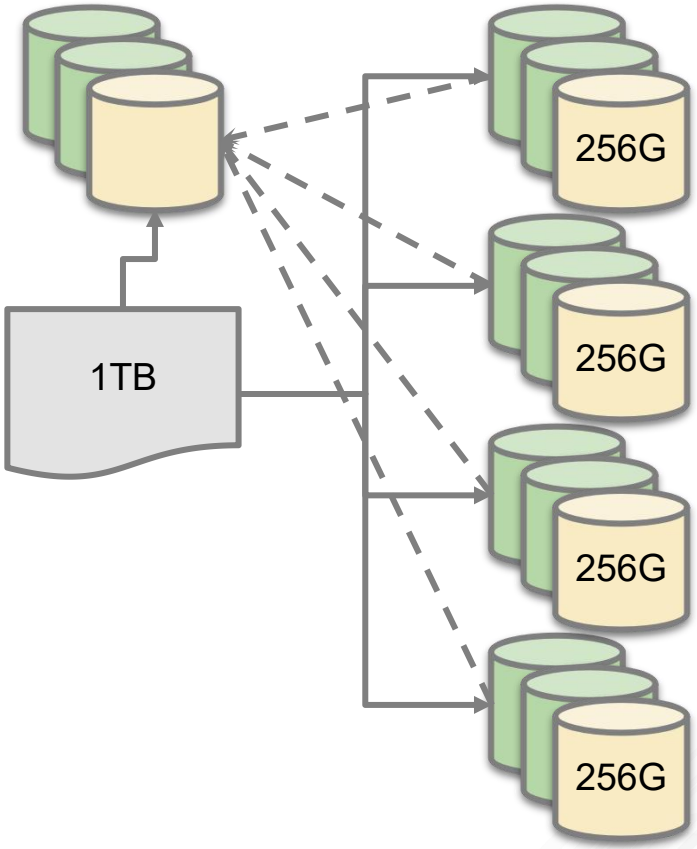
- 通过增加分片即可带来容量，性能上的扩展



相关特性——自动均衡

自动均衡

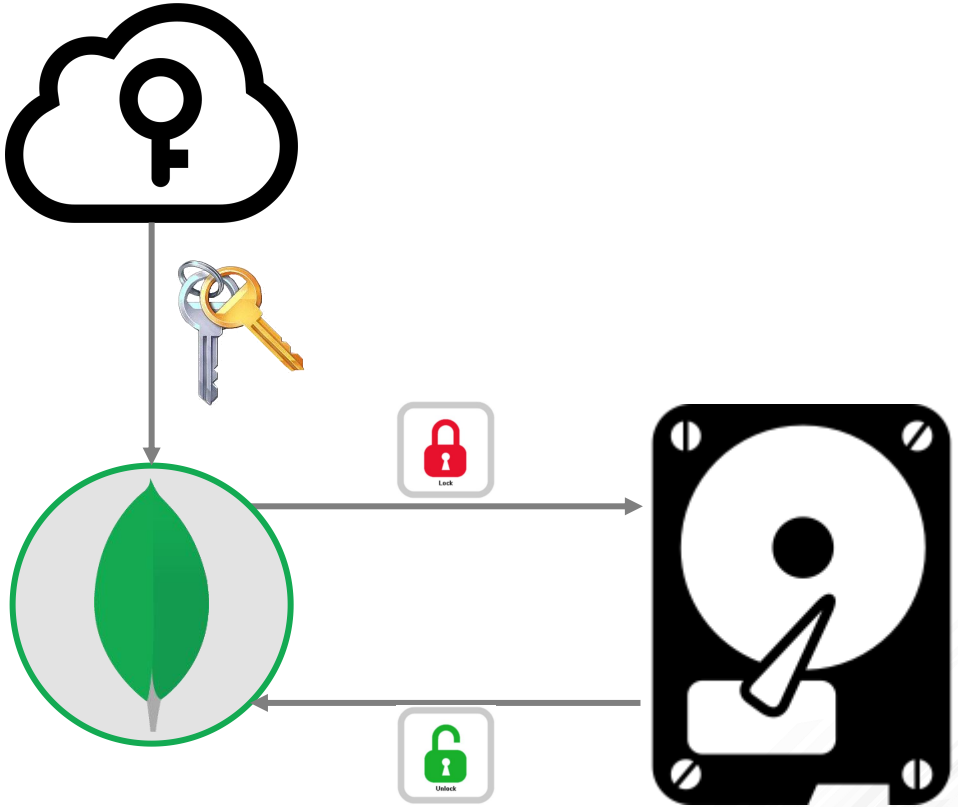
- 选择合适的片键，数据将在不同分片上自动完成均衡



相关特性——自动加解密

自动加解密

- 数据落盘前自动加密，读取时自动解密
- 密钥来自KMS服务器



案例三：国内制造企业

需求

该公司IT部门为了监控位于世界各地的机房服务器信息，定期从服务器上采集各项参数指标并存储在本地。
这些数据需要在中国进行汇总，提供统一的查询和监控门户。

尝试

使用Oracle的尝试：

- GoldenGate跨国家地区复制不稳定，时常失效。
- 机房数量众多，需要的Oracle实例非常多。

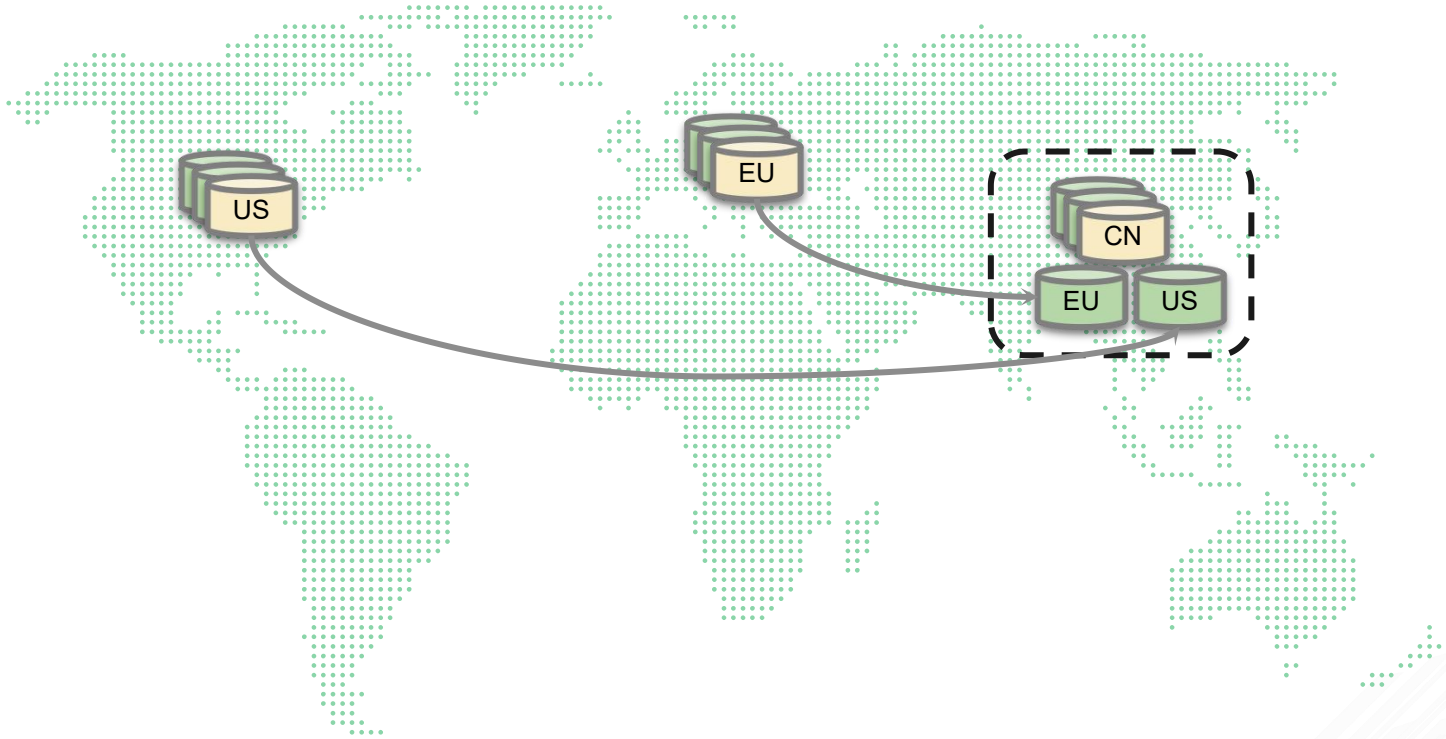
结果：评审期被否定

使用MongoDB的尝试：

- 复制集提供了稳定的数据复制能力
- 7x24小时高可用
- 区域分片使地区问题变得透明

结果：成功

系统架构



案例总结

本案例最大的挑战在于涉及的区域众多，分布在世界各地，造成数据集中困难。为了解决这些问题，MongoDB的以下特性提供了便利：

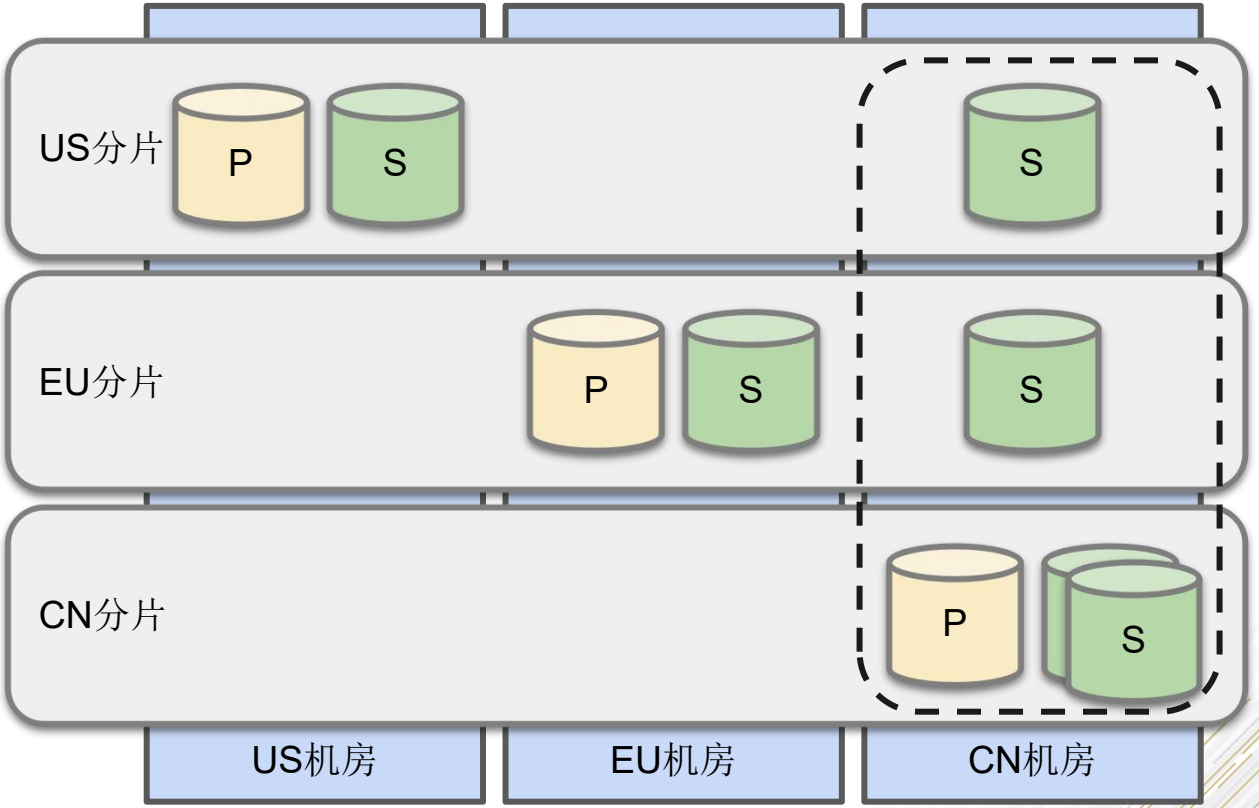
- 跨区域分片集群：解决各个区域分散写入问题；
- 复制集：解决数据从各个区域向中国集中的问题；

以上特性的使用使得区域问题透明化，不需要额外的第三方方案解决分散数据的集中问题。

相关特性——跨地区分片

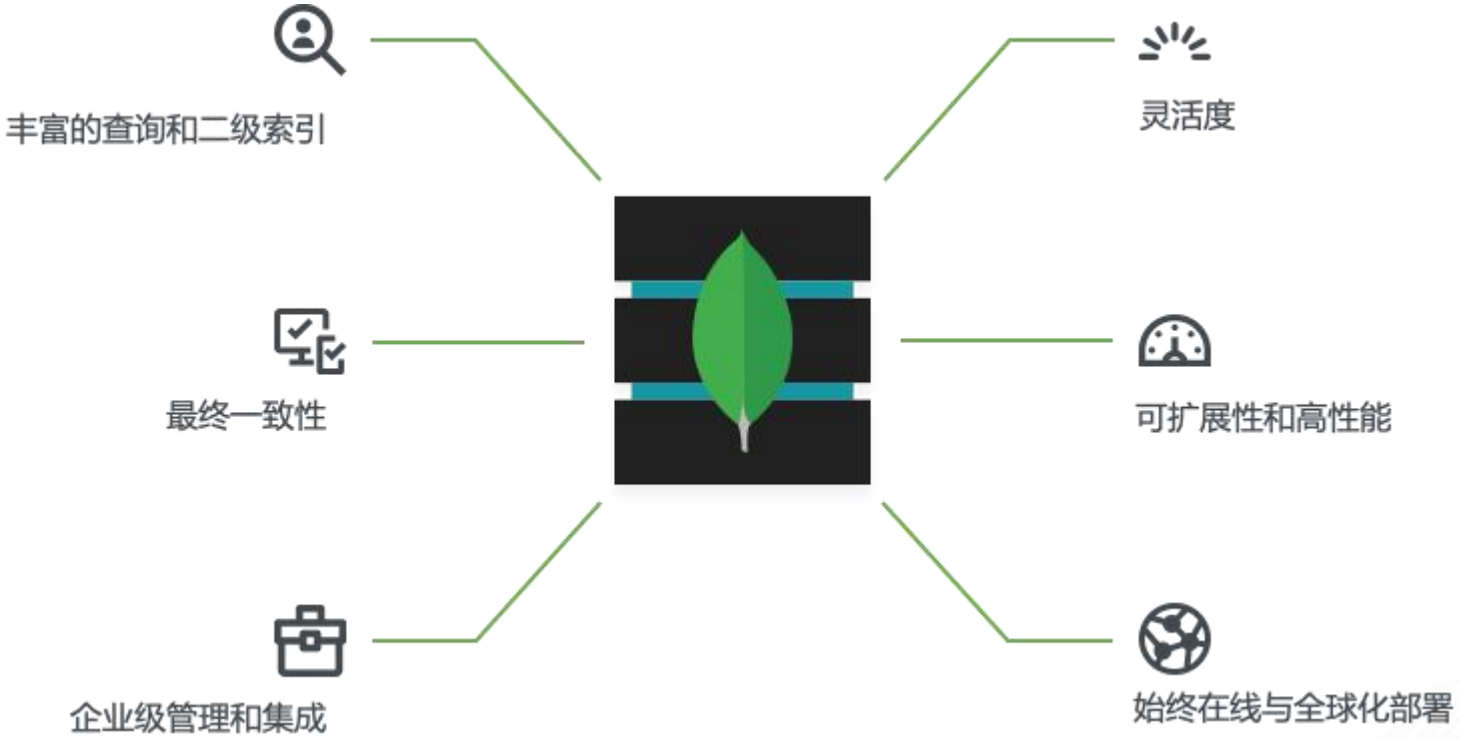
跨地区分片

- 利用**Zone**控制数据写入到哪个分片
- 利用复制集进行跨洋远程复制



总结

了解MongoDB



了解你的需求

需求	MongoDB	RDBMS
地址位置查询	Yes	Yes*
水平扩展	Yes	No
高可用	Yes	Yes*
动态数据模型	Yes	No
……	……	……

做出决策

使用最合适的技术解决最合适的问题，加速产品上线的速度！

Q&A