ArchSummit全球架构师峰会北京站2015

基于数据同步的云服务架构实践

谢乔@野狗科技

Geekbang.
极客邦科技

整合全球最优质学习资源，帮助技术人和企业成长

Growing Technicians, Growing Companies

InfoQ
专注中高端技术人员的
技术媒体

EGO EXTRA GEEKS' ORGANIZATION NETWORKS
高端技术人员
学习型社交网络

StuQ
实践驱动的
IT职业学习和服务平台

GiT GEEKBANG INTERNATIONAL TRAINING 极客邦培训
一线专家驱动的
企业培训服务

地址：北京市朝阳区洛娃大厦C座6层1607室　　网址：www.geekbang.org

大纲与范畴

- 野狗的数据同步理念
- 数据同步的架构演进
- 数据同步的细节问题

Wilddog 是什么狗?

一个云端树形数据库

（一个App的所有的数据存到一个大JSON中）

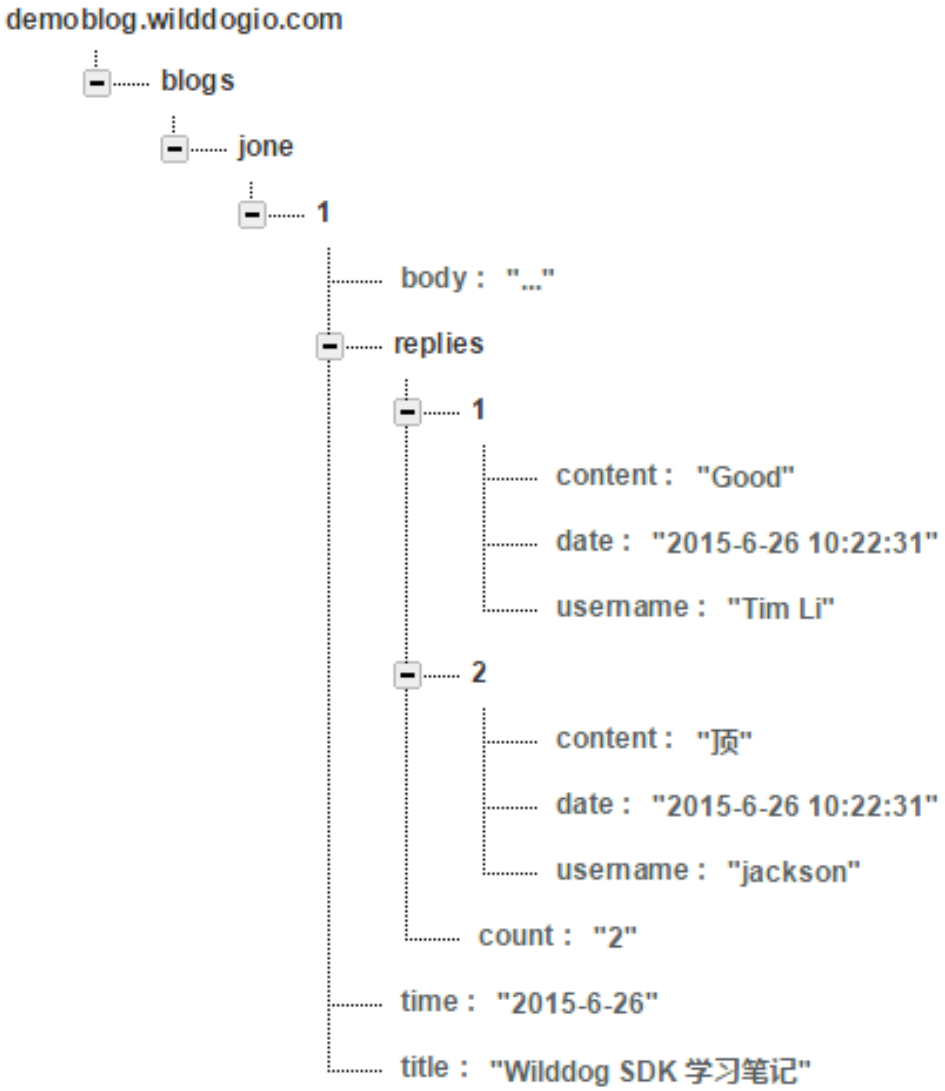
+

事件驱动客户端

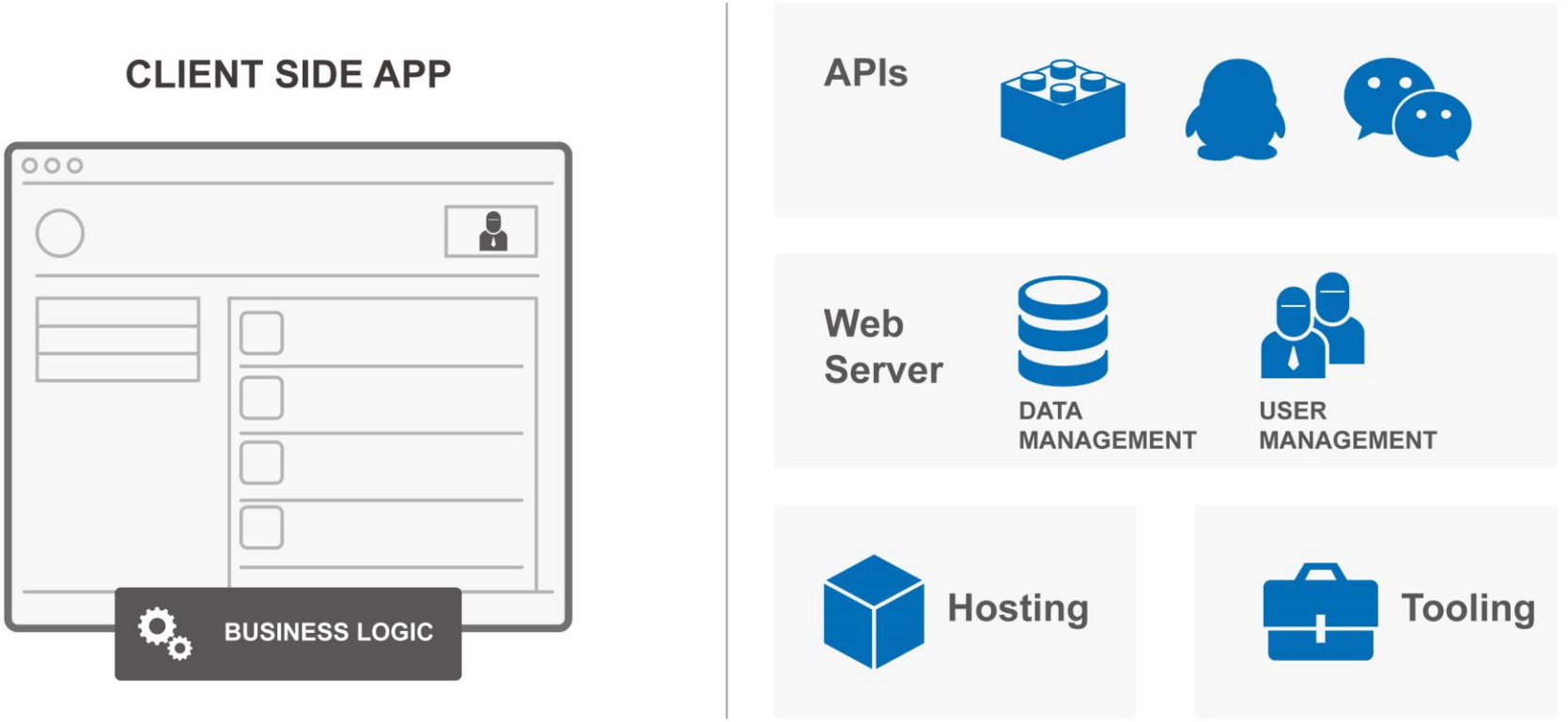
BaaS

Schema-free 数据存储

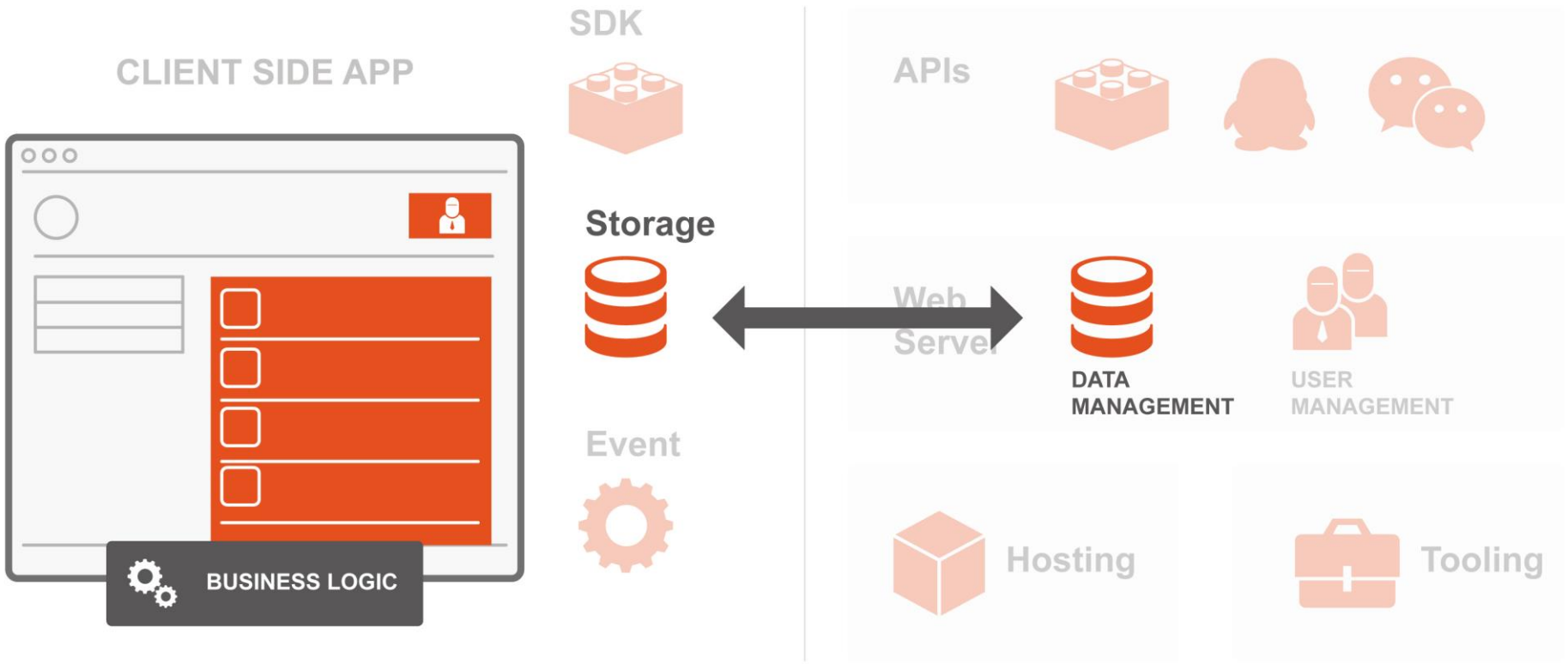
- 树形数据库
  - 像一颗Json树
  - 面向聚合
  - 数据之间的关系更直观
- 完美的与Url结合
  - 每条数据都能通过url来唯一定位
  - Path为key，key - value



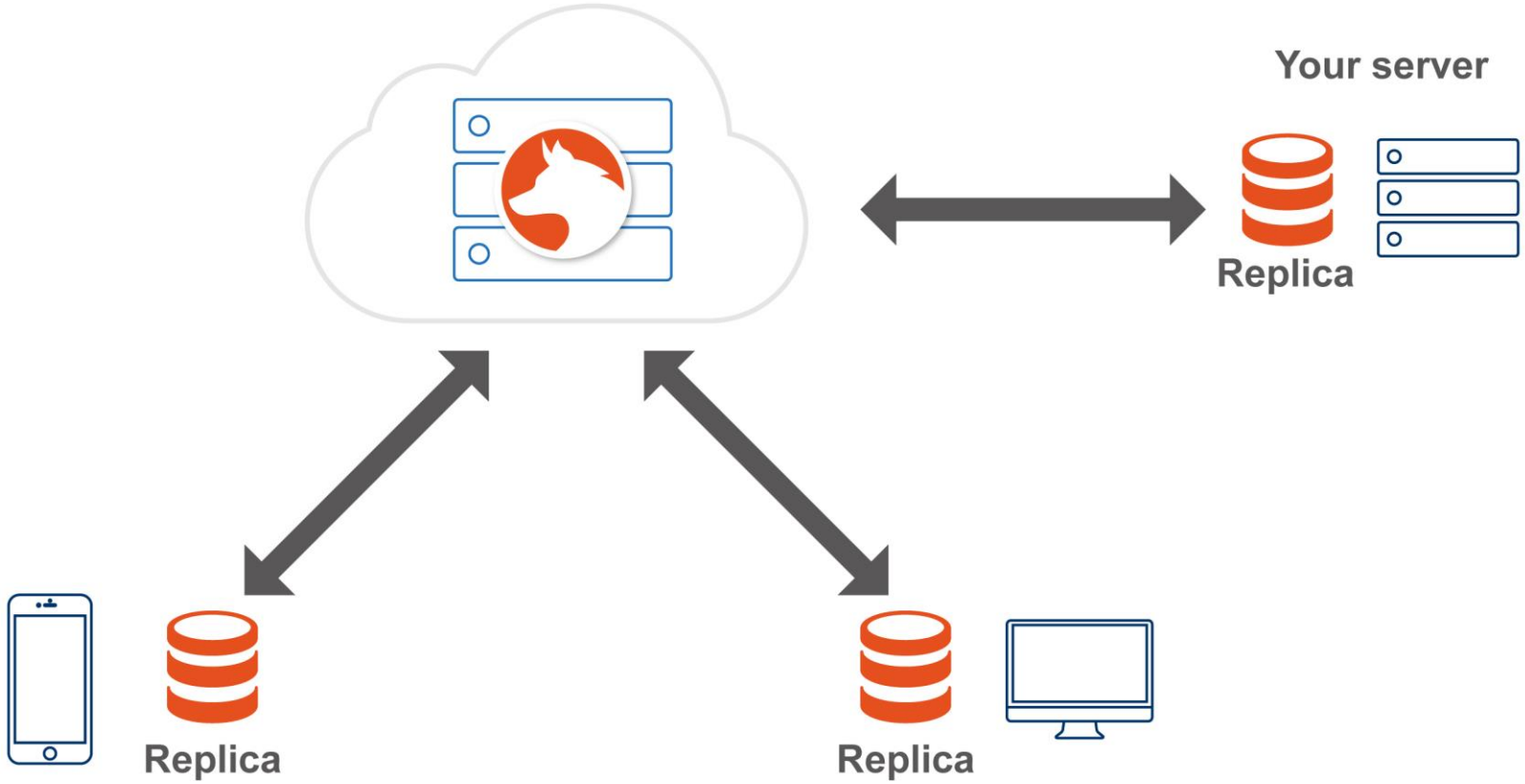
经典云服务


CLIENT SIDE APP
BUSINESS LOGIC
APIs
Web Server
DATA MANAGEMENT
USER MANAGEMENT
Hosting
Tooling

野狗是这样的

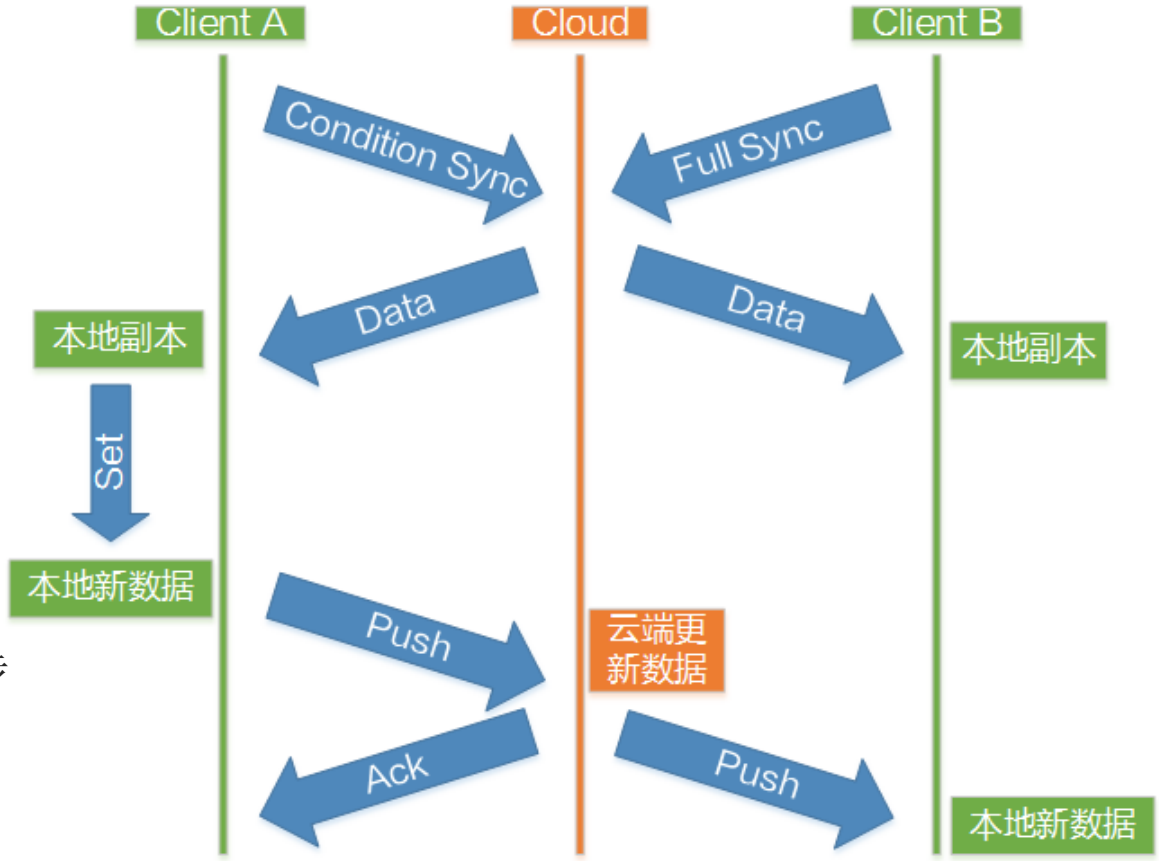

CLIENT SIDE APP
BUSINESS LOGIC
SDK
Storage
Event
APIs
Web Server
DATA MANAGEMENT
USER MANAGEMENT
Hosting
Tooling

副本的同步



数据同步的基本模型

- 初始化慢同步
  - full sync
  - condition sync
- 增量同步
  - 本地 best-effort
  - push op log
- 基于长连接
  - 保证有序性
  - 重连需要初始化慢同步



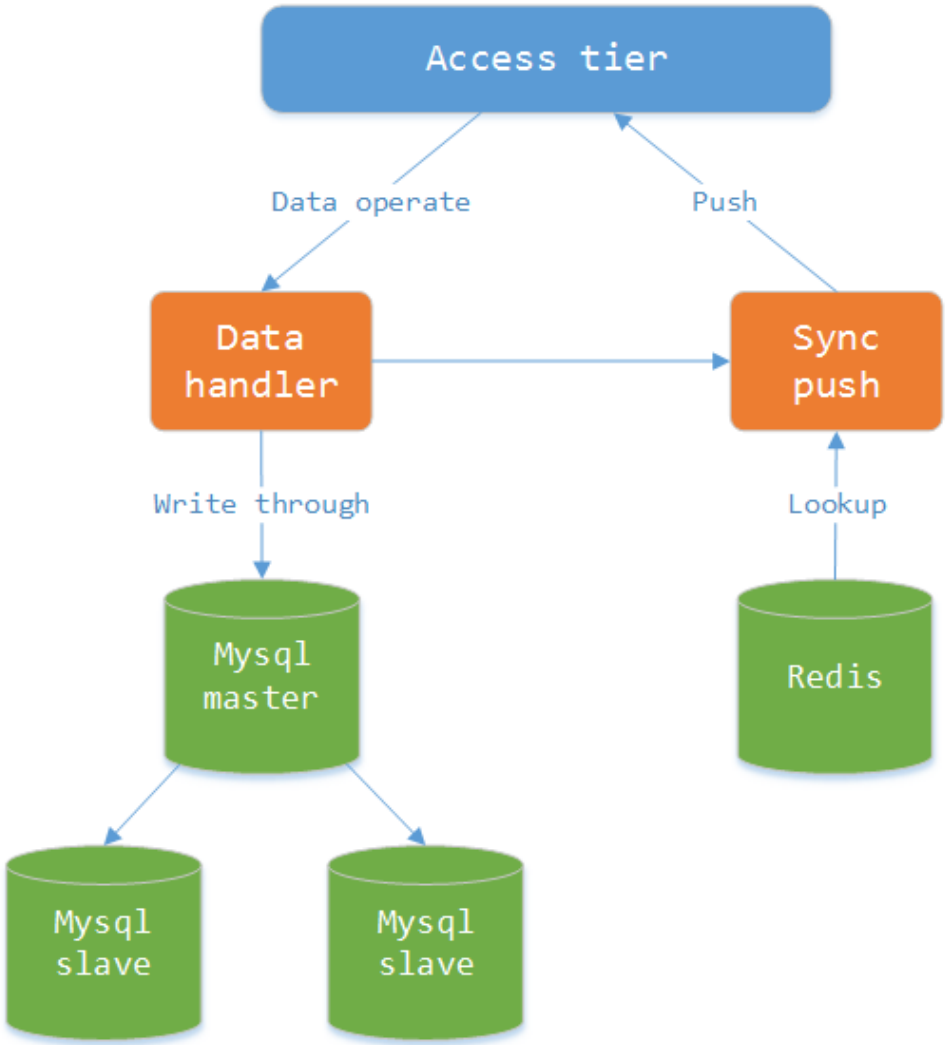
架构演进

架构特点

- 写多读少
  - 写同步越实时越好
  - 读容忍一定延迟
- 最终一致性
- 并行写冲突
- 实时
- 操作幂等

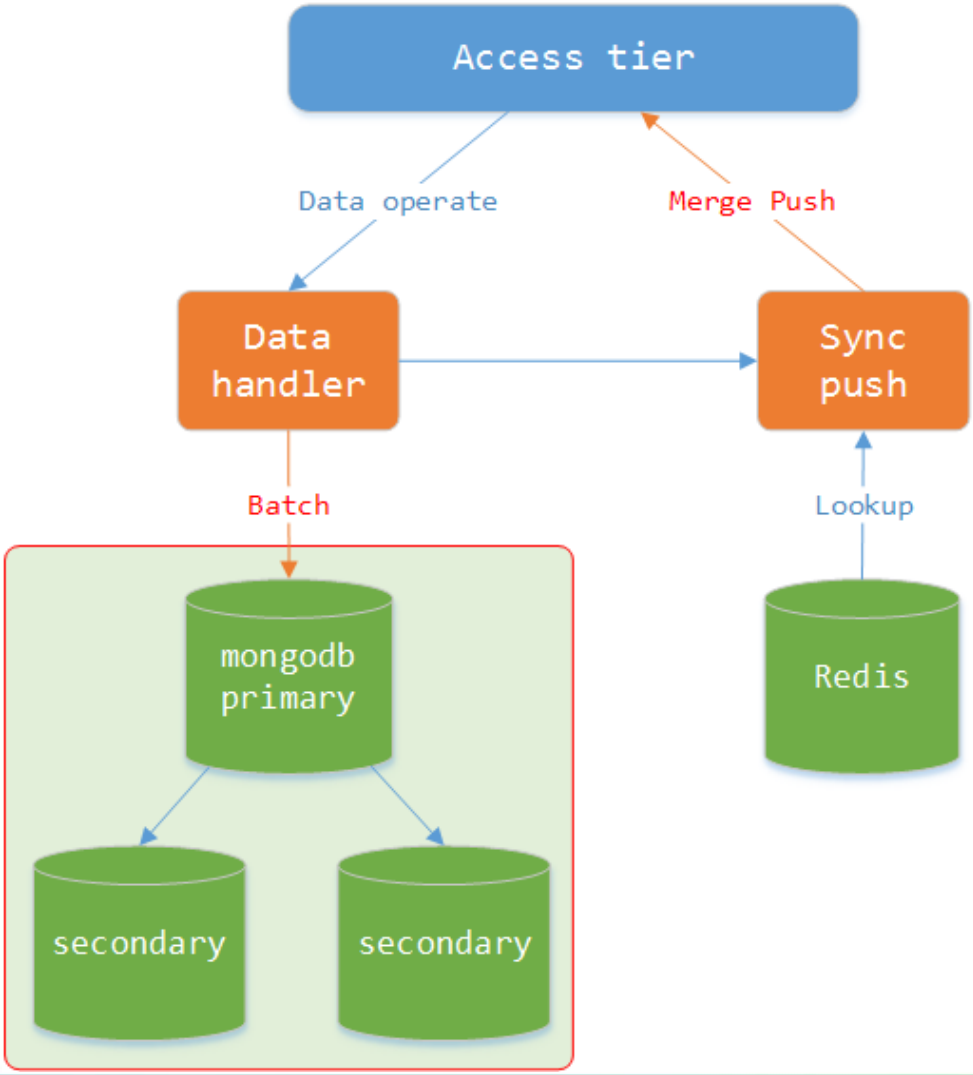
v 0.1 架构框图

- 面向早期用户
- 物化路径存储



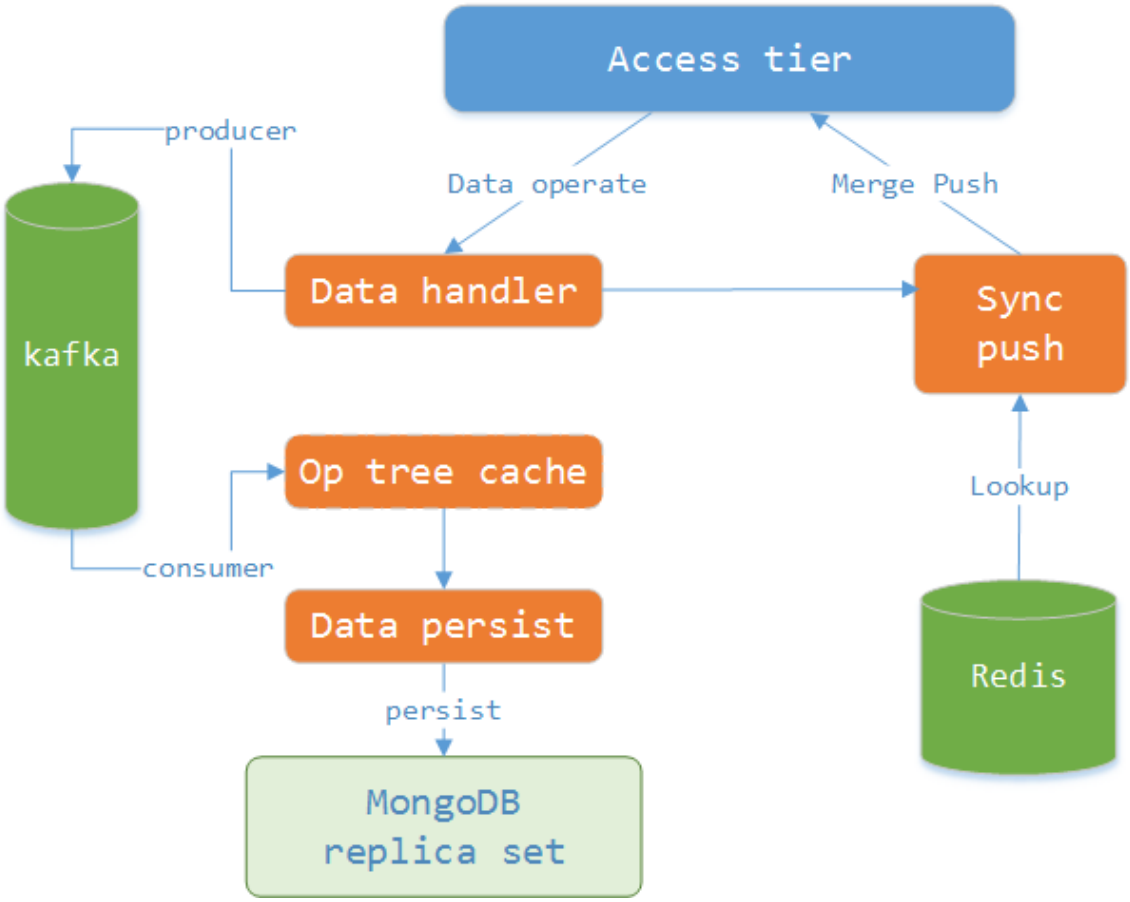
v0.2 架构框图

- 动态建库，app数据隔离
- Mongo 提升读写性能
- 副本集多活
- 机枪换导弹



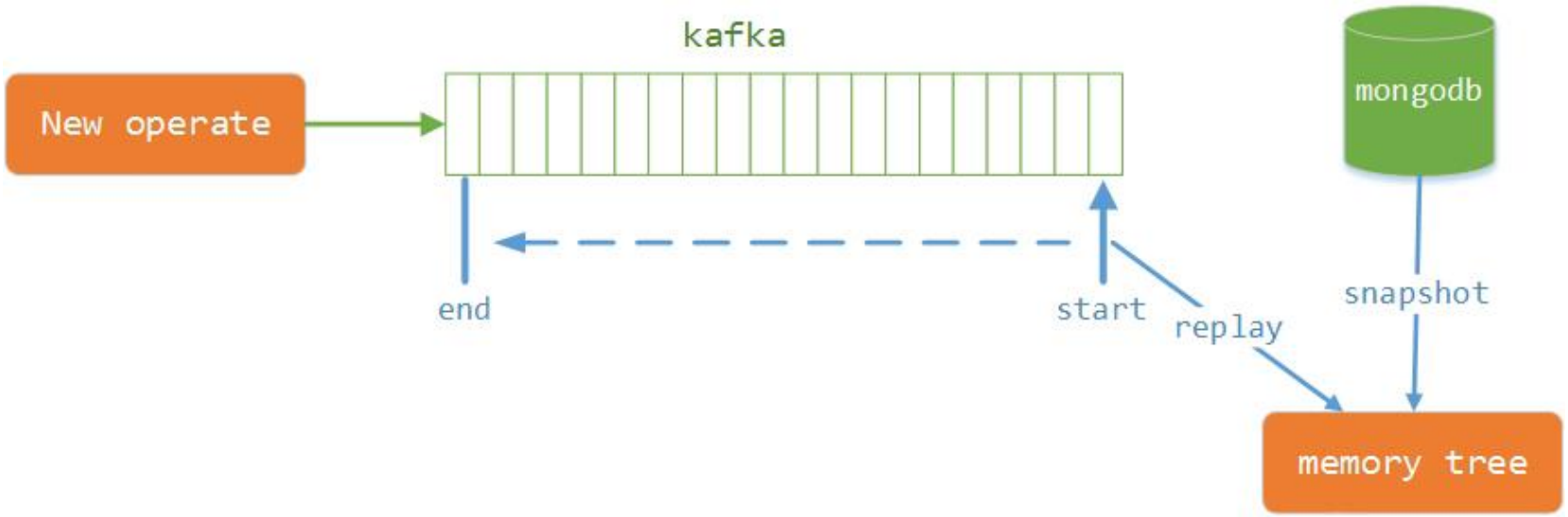
v0.3 架构框图

- appId - topic
- 大大提升写性能
- 类似Nagle减低写压力
- 读性能下降



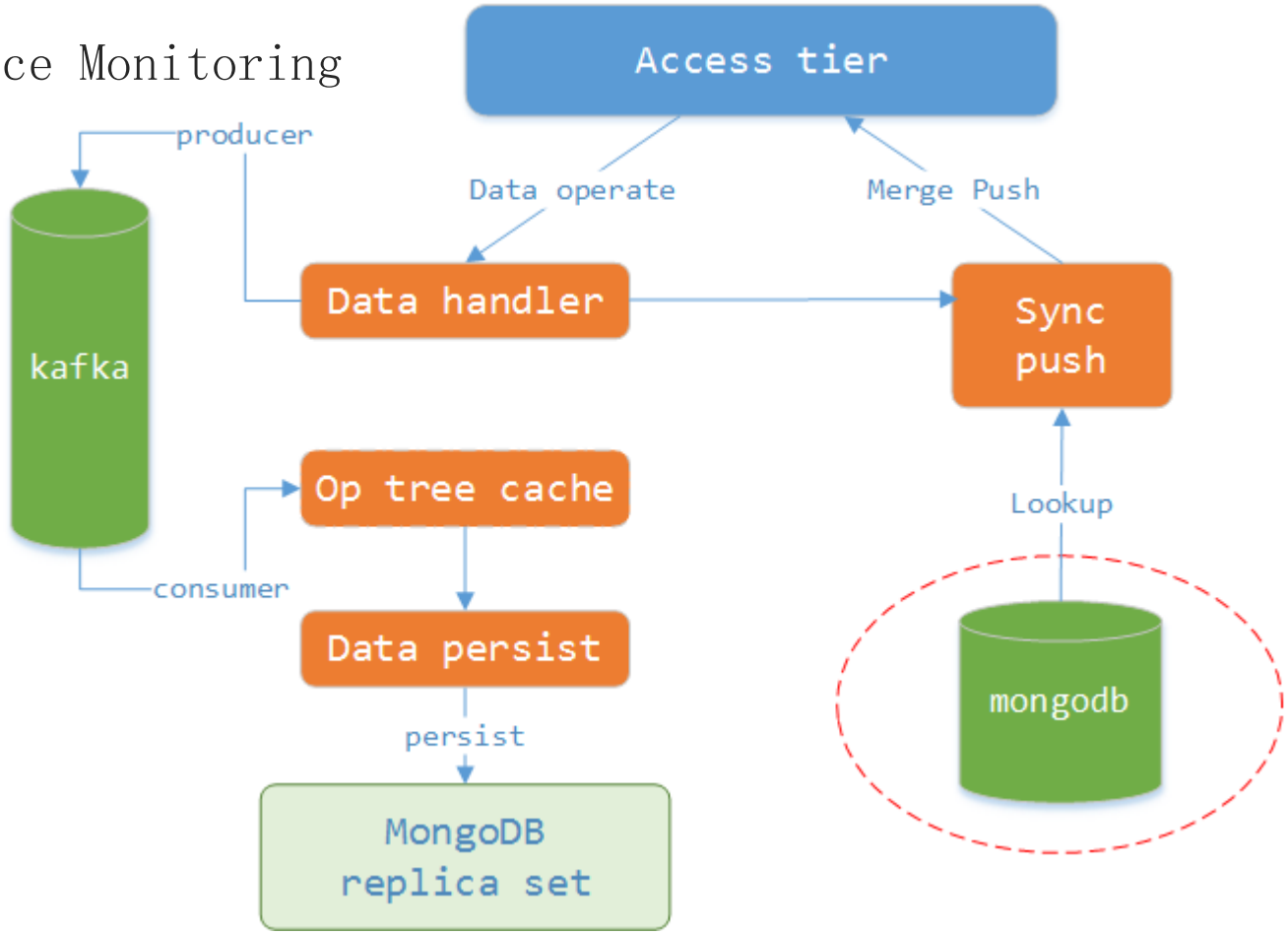
解决读的不一致问题

- 幂等的覆盖模式操作



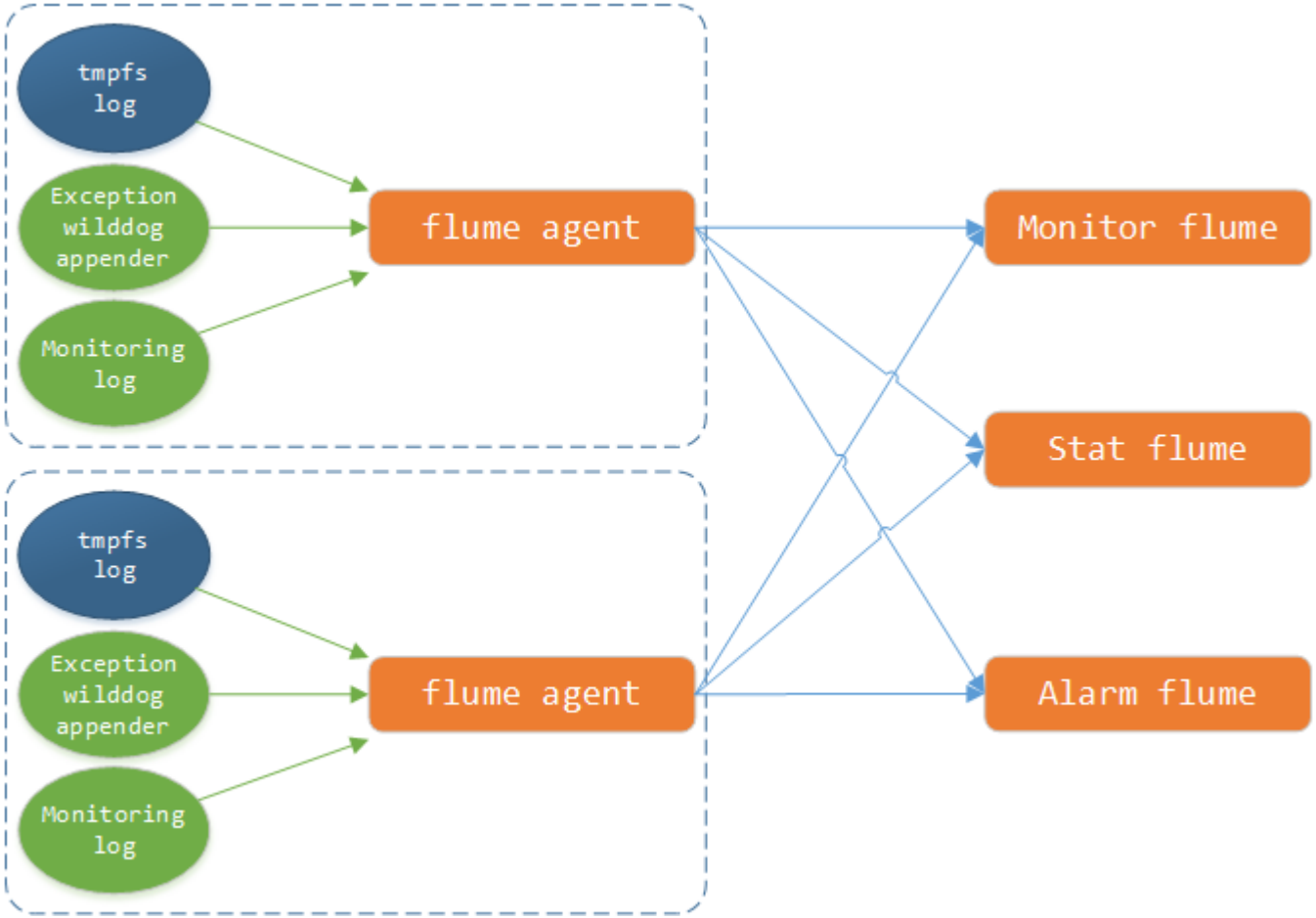
v 0.3.1 架构框图

- keys xxx* 引发的血案
- 需要Performance Monitoring



Performance Monitoring

- 流量统计
- 调用延迟
- 异常报警

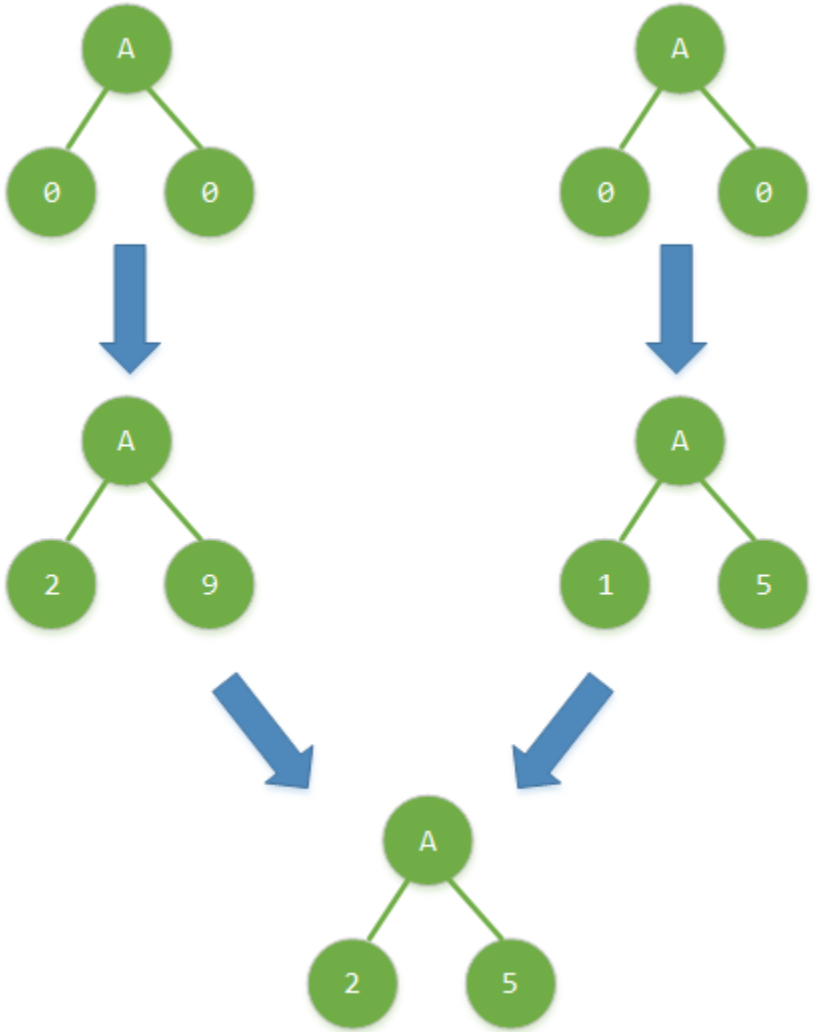


细节问题

大巧不工　重剑无锋

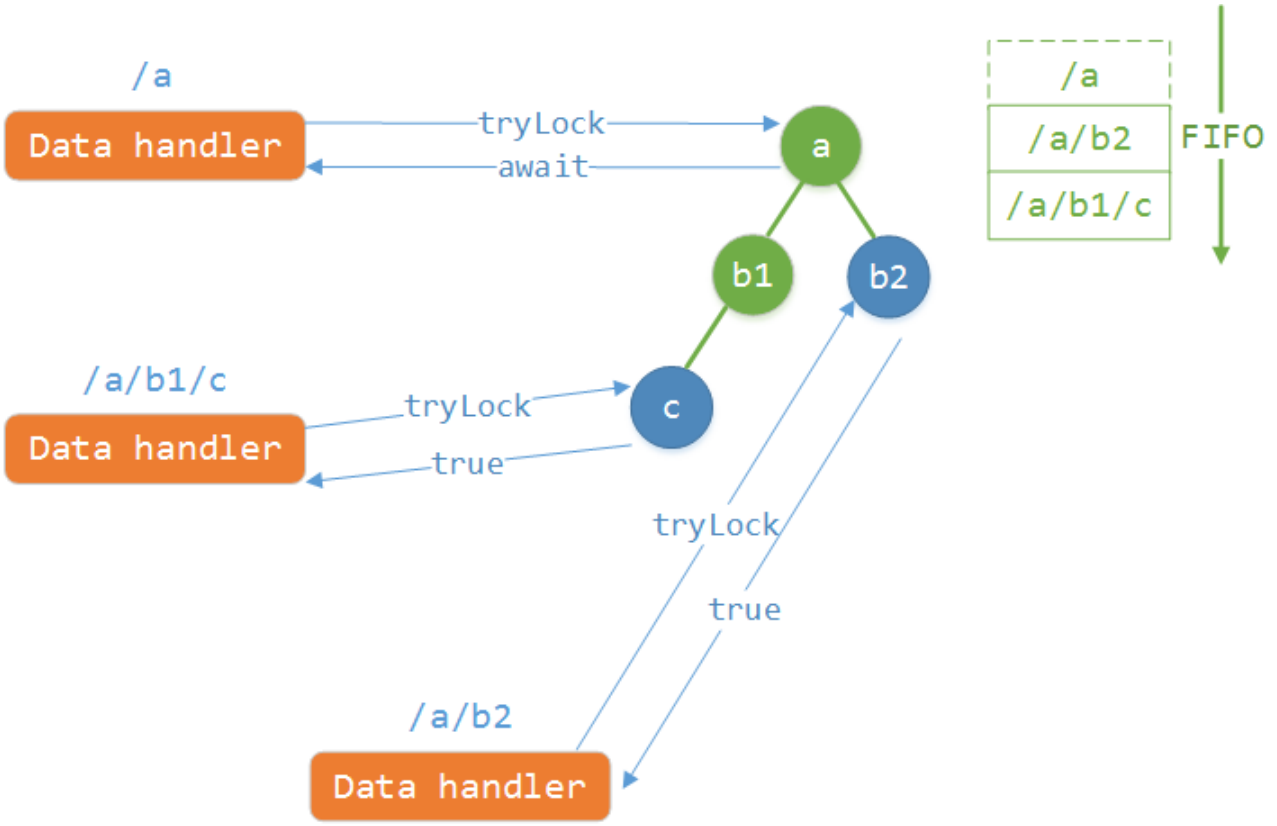
问题一　并发写

- 静态一致性
- 写隔离
- 解决方案
  - 中心化锁机制
  - 进程间协商机制



分布式树形锁

- 注意问题
  - tryLock/release 需要2次交互
  - 注册Lock的有效期
  - 等待Lock超时
  - 动态hash
  - 连接异常时退化

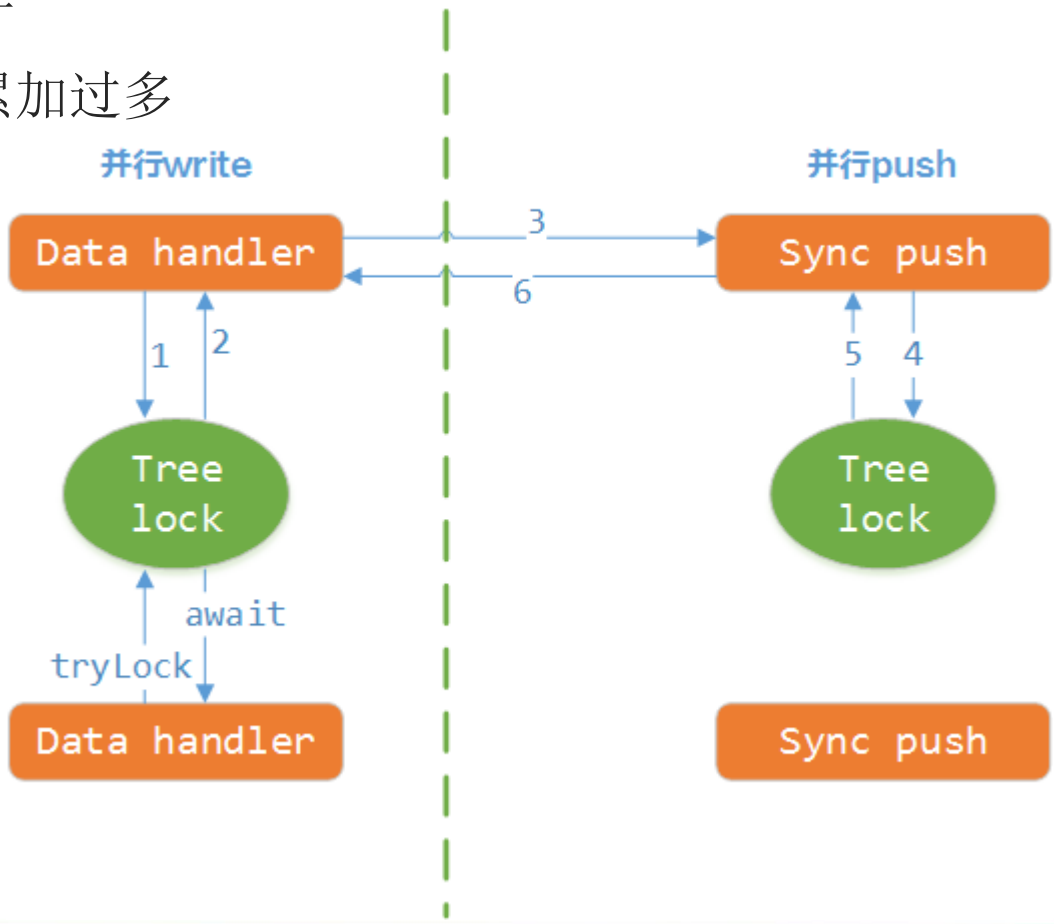


性能问题

- 吞吐量下降
  - 每个app一个树形锁，单进程，终究有吞吐上限
  - 任何操作，包括没有冲突操作，都需要先获得锁
- 主要性能的点
  - 单次push sync量大，可以导致阻塞
  - 异步push sync

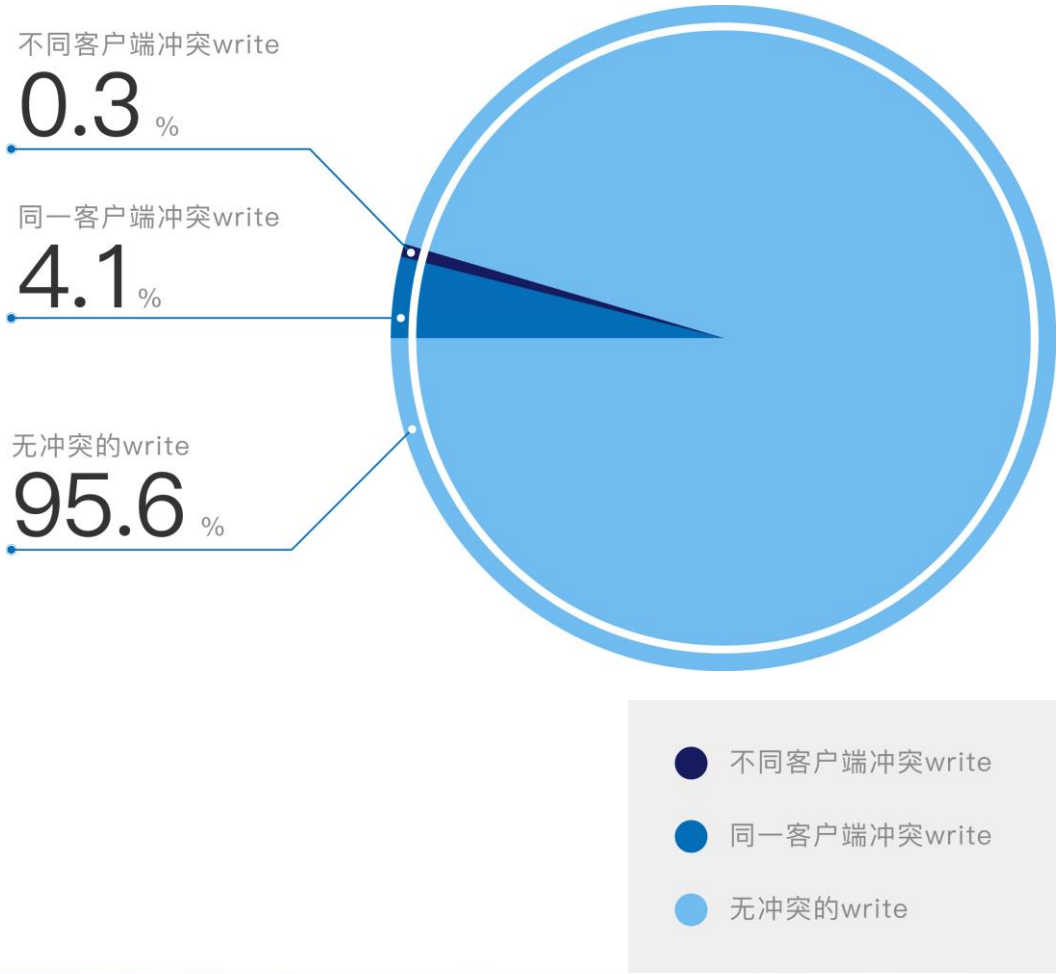
令人恶心的架构诞生了

- 缩减了write操作的过程
- 保证云端与客户端一致性
- 太过复杂，不确定因素累加过多



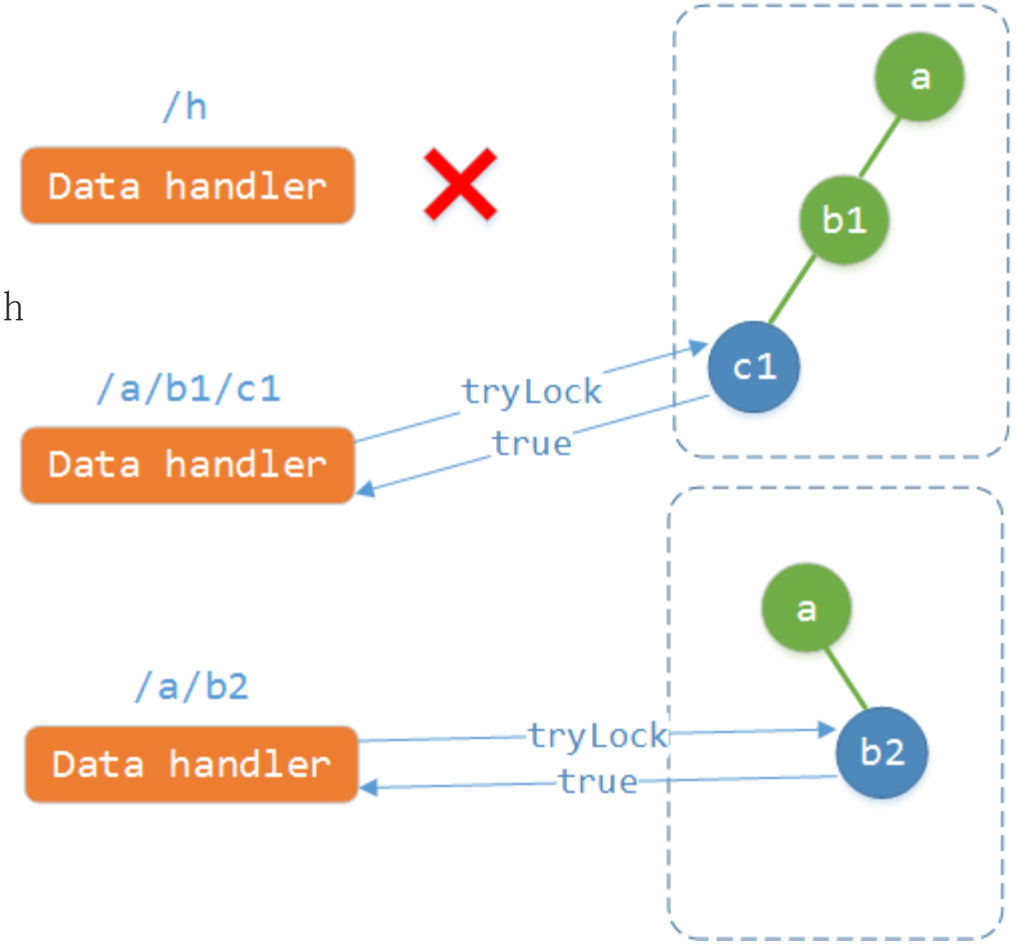
做技术不能意淫

- 同一客户端场景
  - GPS实时定位
  - 模拟游戏手柄的按键
  - 游戏中人物坐标的变化
  - 烟雾报警器实时采集
- 不同客户端场景
  - 投票、点赞等计数器
  - 拔河游戏
  - 协作文档
  - 智能彩灯



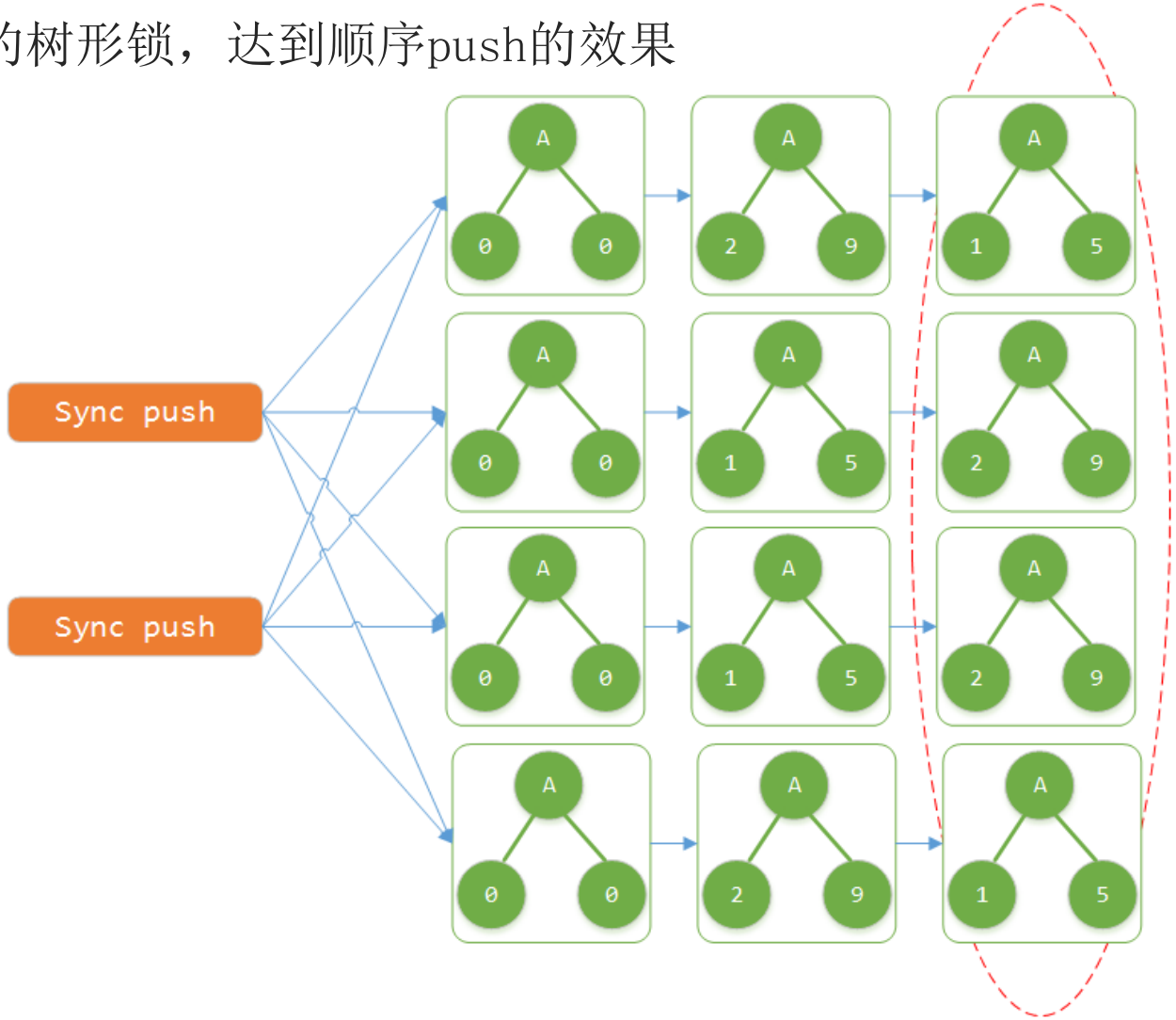
上帝的归上帝，野狗的归野狗

- 用户可配置化
  - 默认不启用
  - 大大减少不必要的开销
  - 降低锁粒度
  - 由appId hash 改进为 path hash



问题二　push并发冲突

- 依靠写时的树形锁，达到顺序push的效果

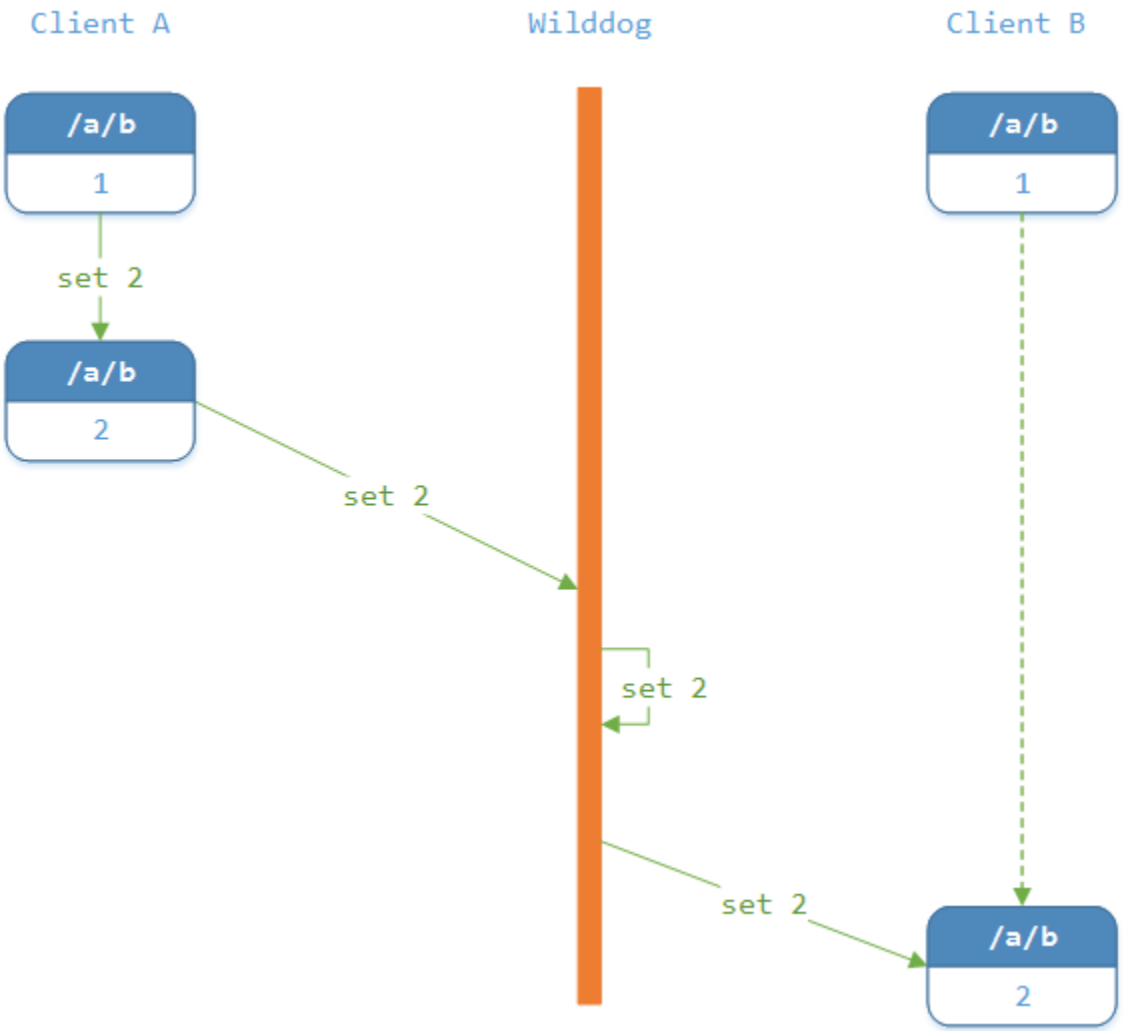


问题三　顺序一致性

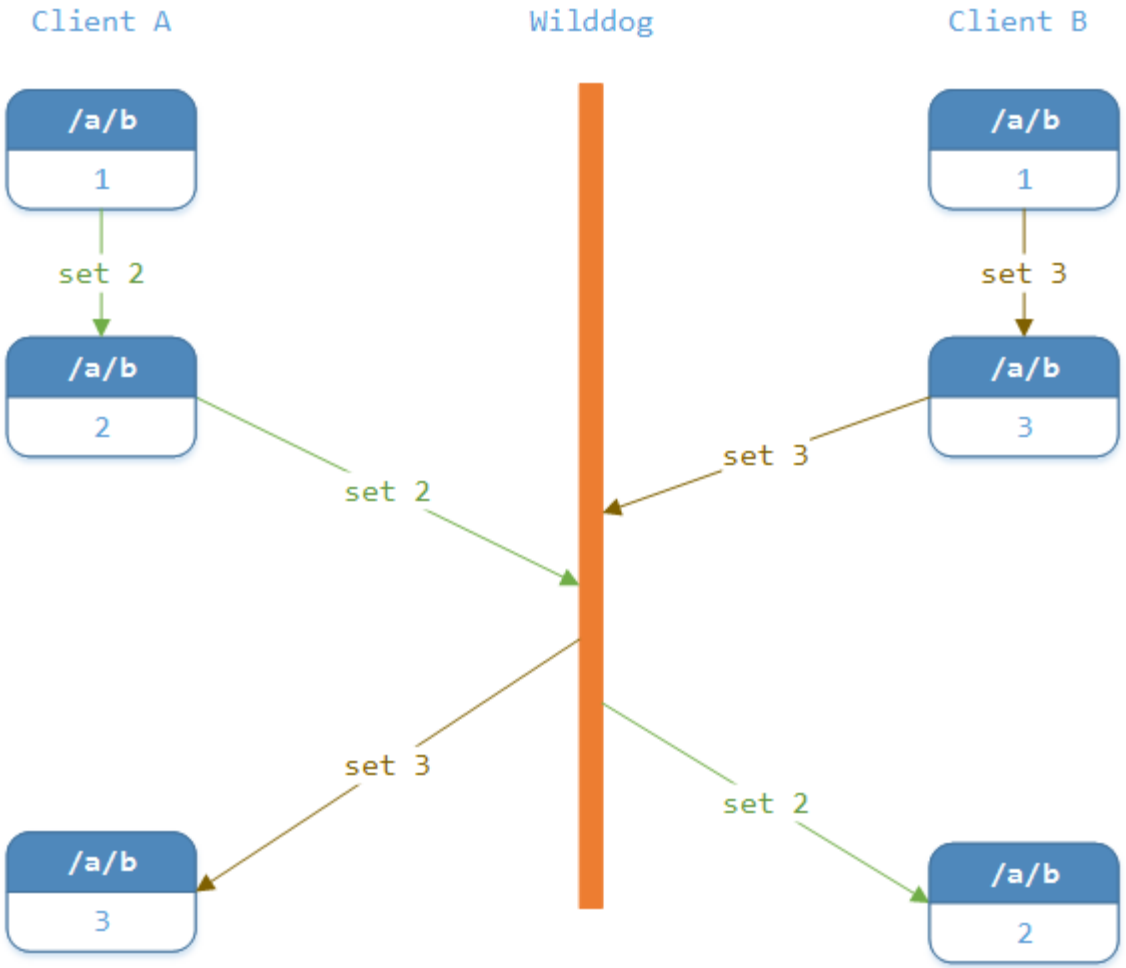
- 需要保证同一客户端的顺序性
  - 按照clientId 散列到data handler
  - 进程内锁实现顺序性
  - 目标又变成解决write的性能

问题四　最终一致性

- 实现最终一致性

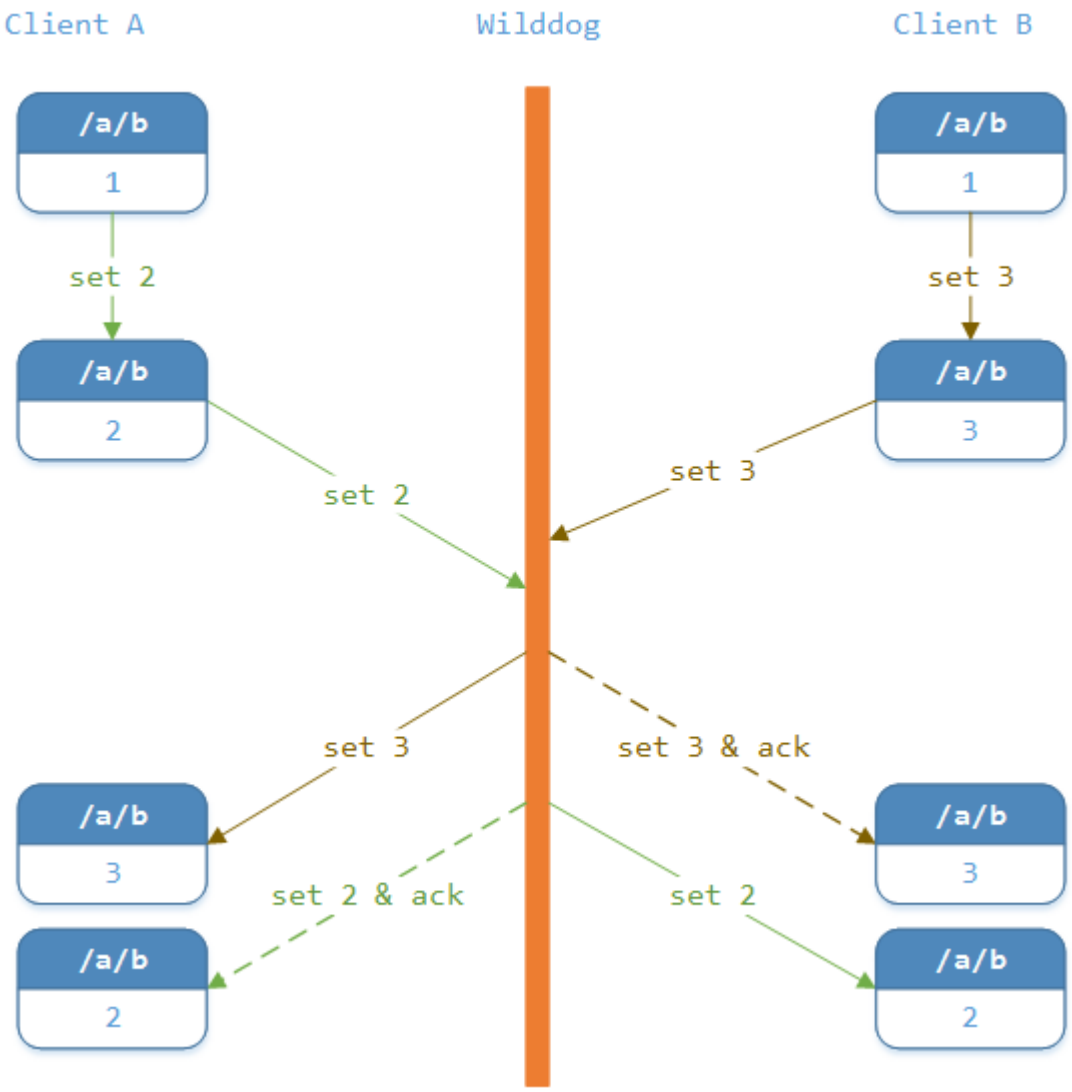


多client并发写冲突



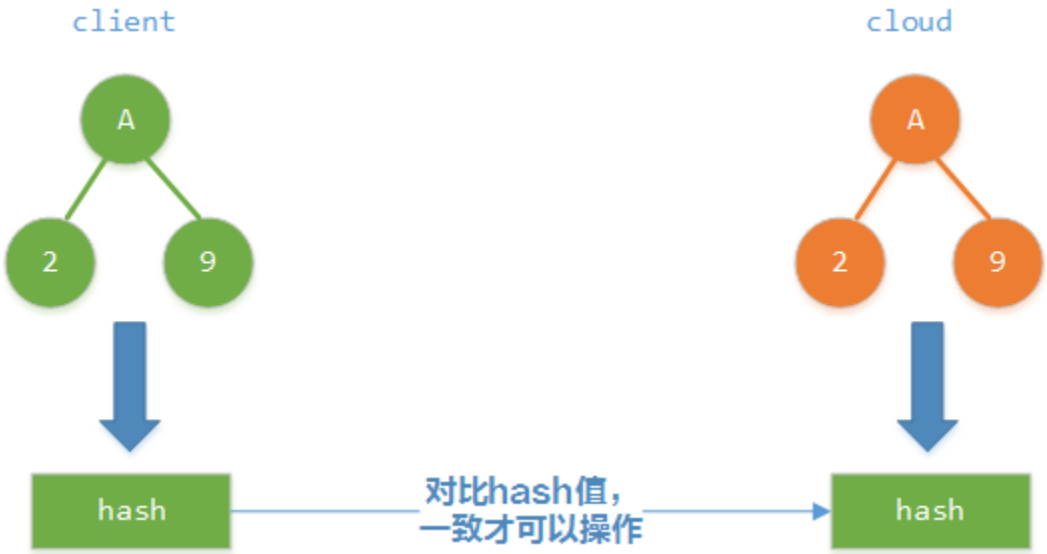
Push给自己

- 返回数据做diff操作



问题五　原子性操作

- 场景
  - if　then
  - i ++



- 自旋锁 CAS
  - 本地副本计算hash，与云端hash对比，一致才可以操作。
  - 与云端hash值不一致，则重新load数据到本地，重新上步操作。

加入我们

hr@wilddog.com

Wilddog 野狗™

Thanks!