# 淘宝面向公网数据同步服务设计与实现

顾风胜 / 阿里巴巴高级技术专家

ArchSummit 全球架构师峰会 2016

[北京站]

主办方 Geekbang 极客邦科技 InfoQ

# 大纲

- 传统的数据同步技术
- 双十一数据同步的挑战
- 分布式数据一致性保证
- 通用数据存储模型
- 资源动态调配与隔离
- 如何降低RDS查询与写入开销
- 性能优化

# 传统的数据同步技术

## 基于SQL语句的同步

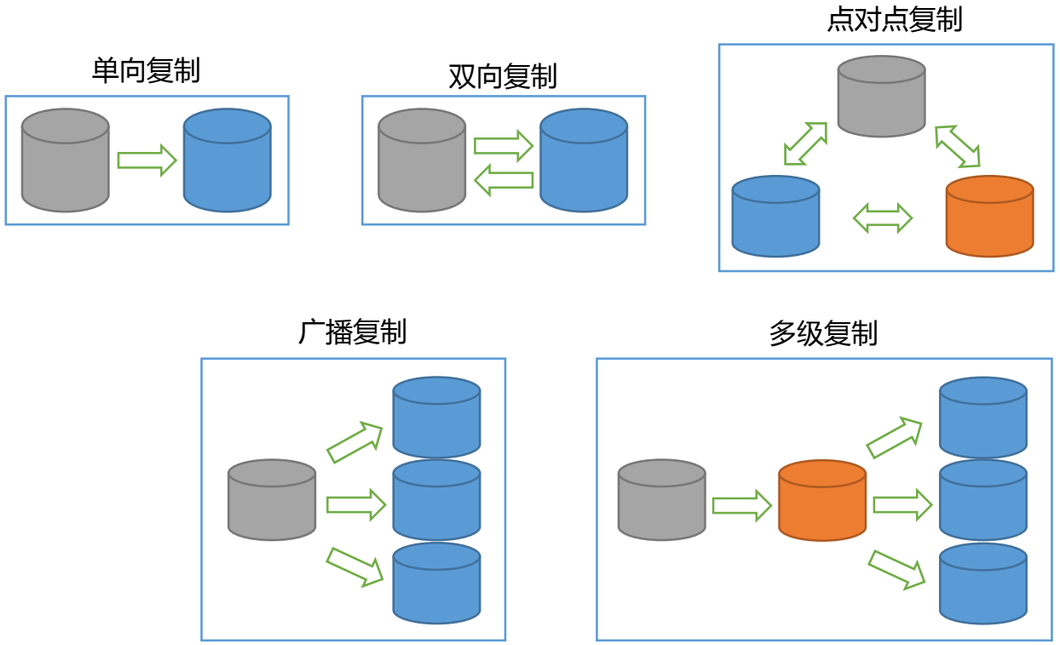
- insert into Table2(f1,f2,...) select v1,v2,... from Table1

## 数据库自带的同步技术

- MySQL基于binlog的语句复制、行复制、混合复制等
- Oracle的DataGuard、流复制、高级复制等

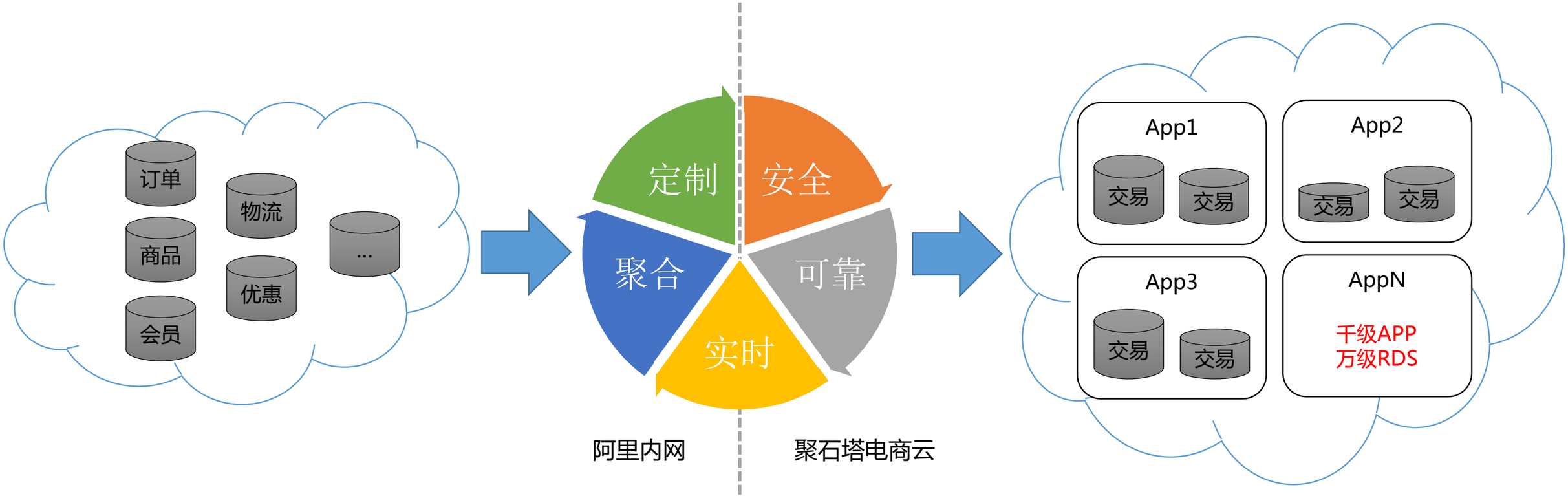
## 异构数据库之间的同步

- 基于MySQL的binlog或Oracle的redolog构建消息队列进行日志重放
- 如：阿里云的数据传输服务（DTS）、Oracle的GoldenGate




适用范围：企业内网、点对点、无业务语义同步

# 淘宝公网数据同步的诉求



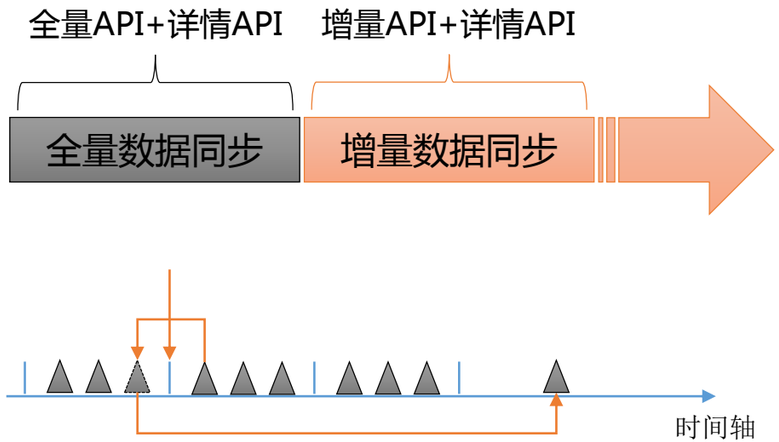
# 基于开放API的数据同步方案

## 方案

- 全量API，数据初始化
- 增量API，增量获取变更数据
- 详情API，获取业务详情数据

## 问题

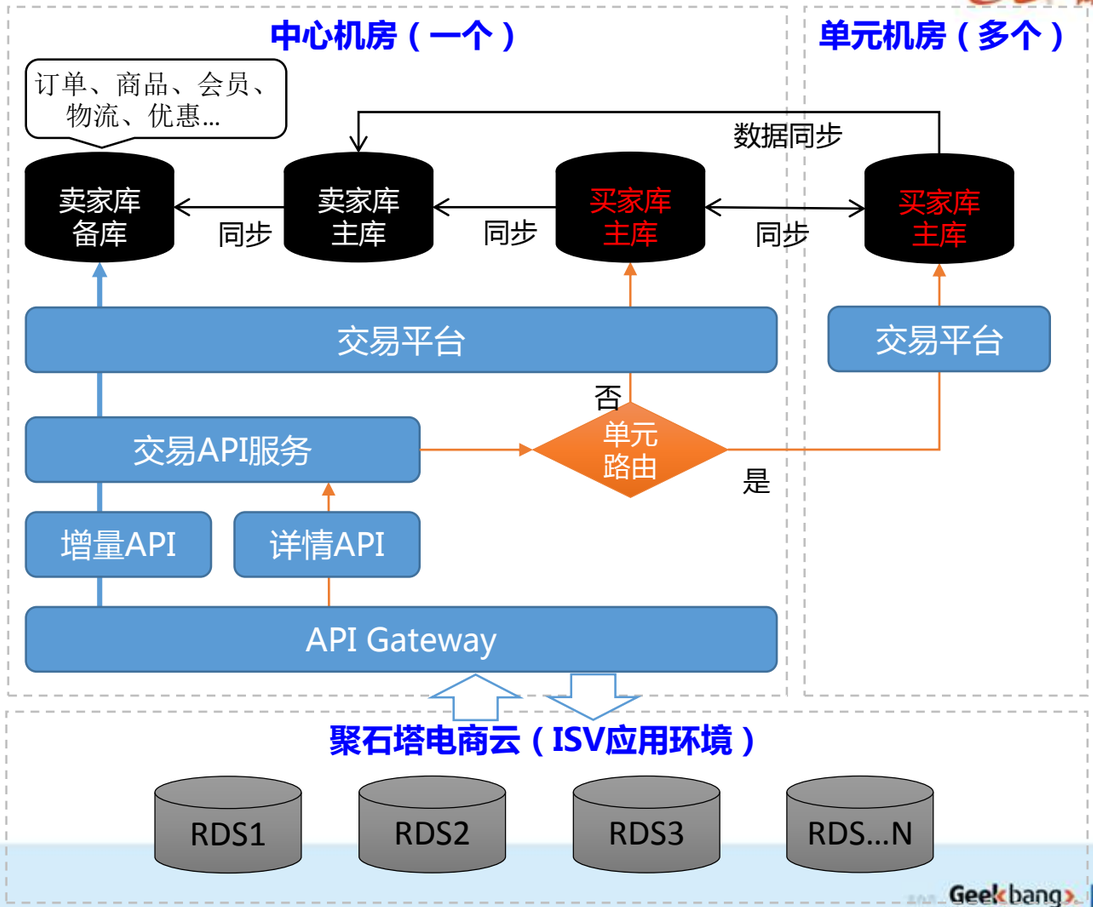
- 轮询实时性不高，性能差，空查询
- 分页获取方式不对导致数据不一致
- 数据热点导致查询DB超时
- ISV重复建设



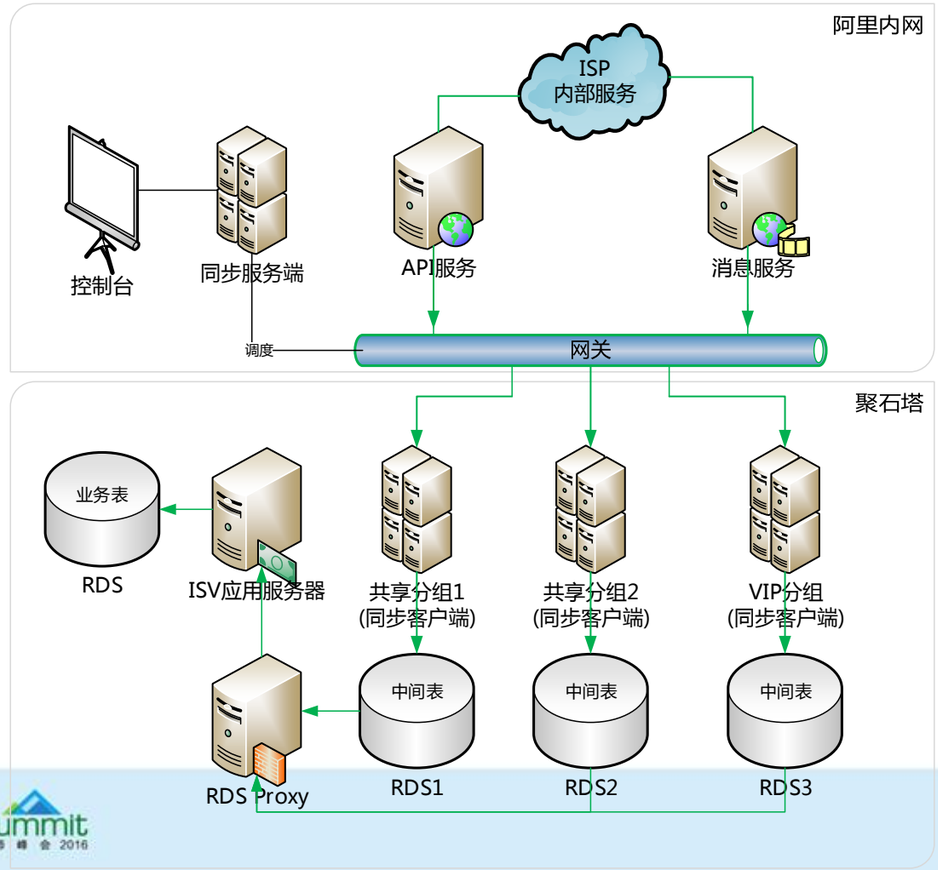

通过API同步，很多大卖家反馈双十一当天订单拉不下来

# 双十一数据同步的挑战

- 聚合数据同步
- 个性化定制同步
- 核心业务接口限流
- 大卖家数据热点问题
- 下游RDS稳定性问题
- 主备同步延迟问题



# 淘宝数据同步服务的架构




## 设计要点：

- 实时可靠消息服务
- 高性能API对账服务
- 分组隔离+双机房容灾
- 中间表设计+大字段设计
- 独享RDS+就近访问

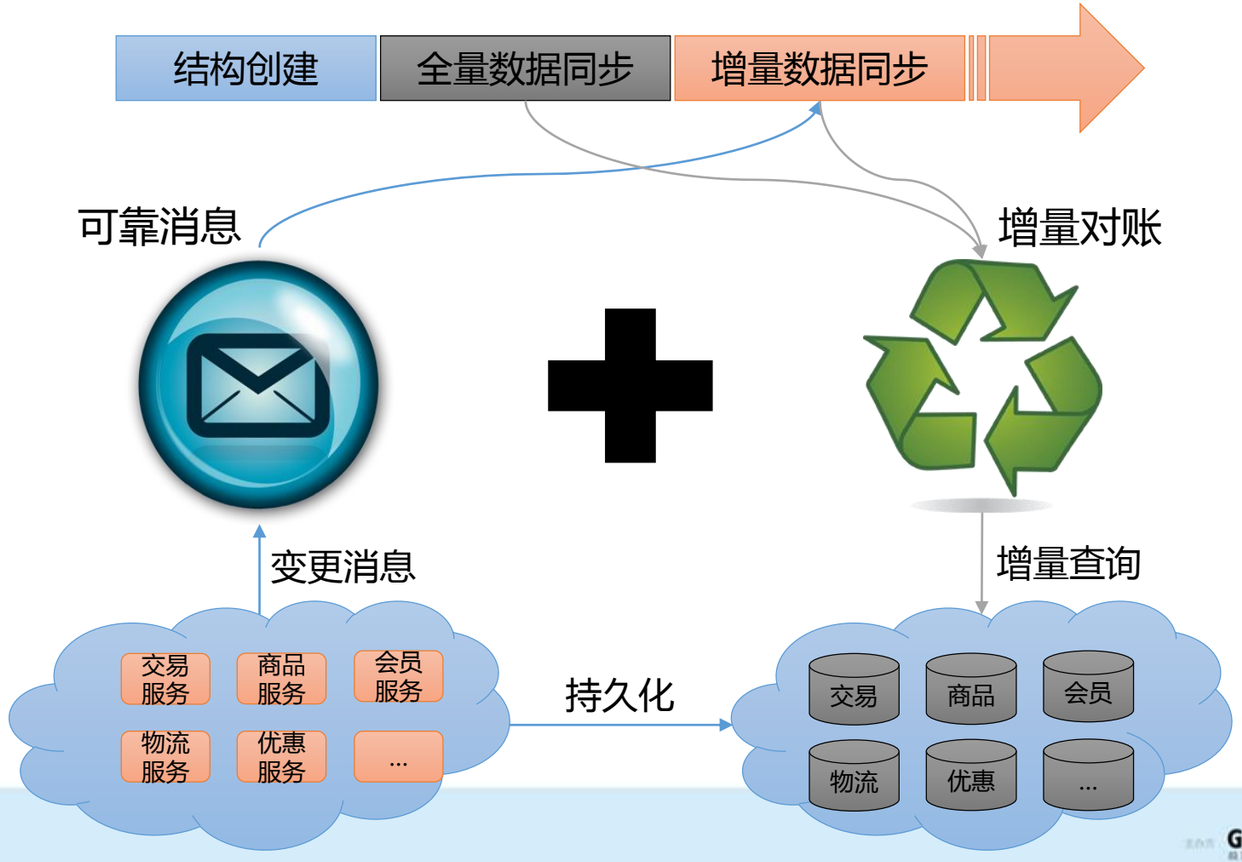
# 分布式数据同步一致性保证

## 消息的必要性

- 数据实时同步
- 降低扫DB的压力

## 对账的必要性

- 源头消息缺失
- 用户授权失效补数据
- RDS不可用导致消息过期



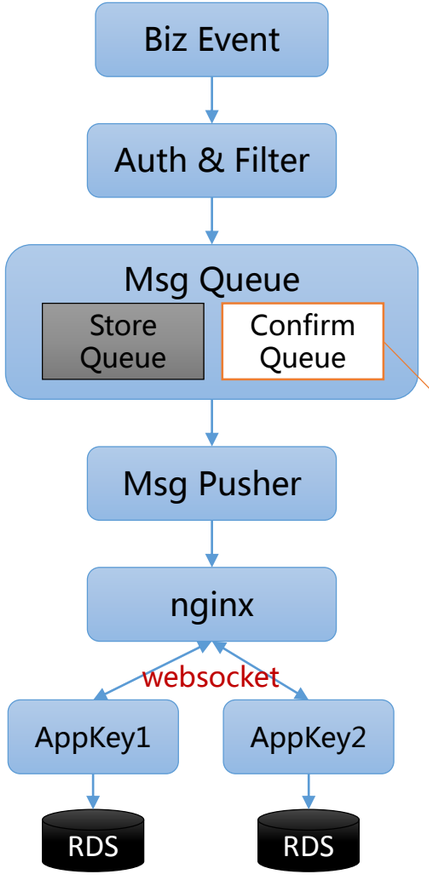
# 消息可靠投递

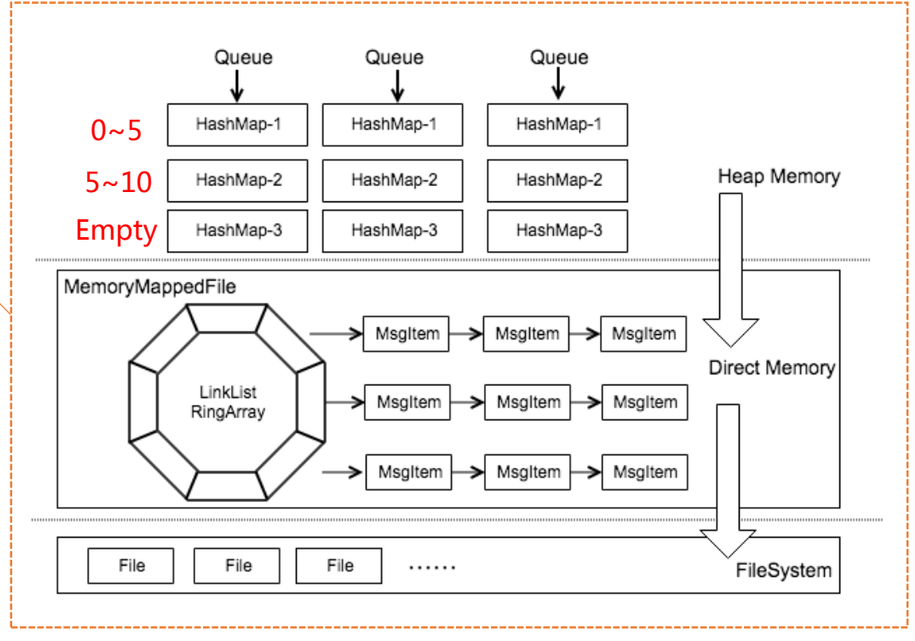
**产生**
- 确保写入成功
- 失败回滚或写日志进行重放

**存储**
- 顺序存储 - 内存映射文件
- 数据备份 - 核心数据主备同步
- 性能提升 - SATA盘升级SSD盘

**投递**
- WebSocket双工通道
- 至少投递一次，定时重试

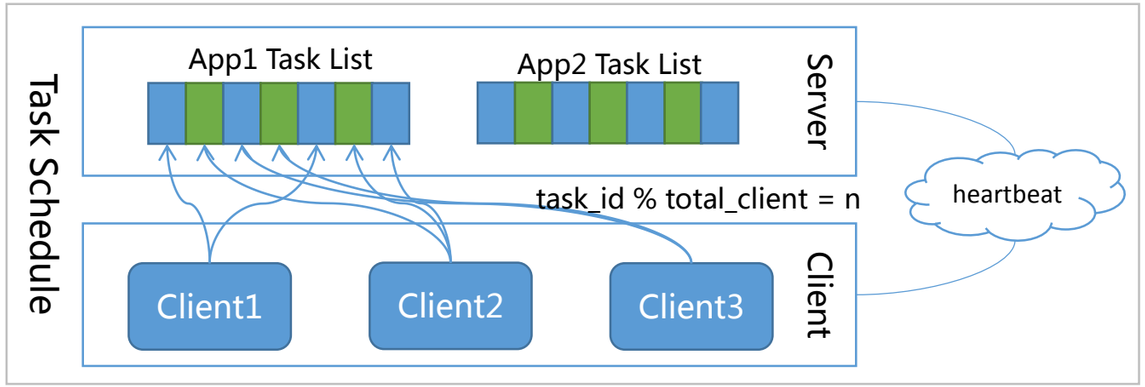
**确认**
- DB表确认升级为多级存储确认
- JVM堆内存中确认（95%）
- 操作系统直接内存确认（5%）
- 文件系统中确认（极少数）

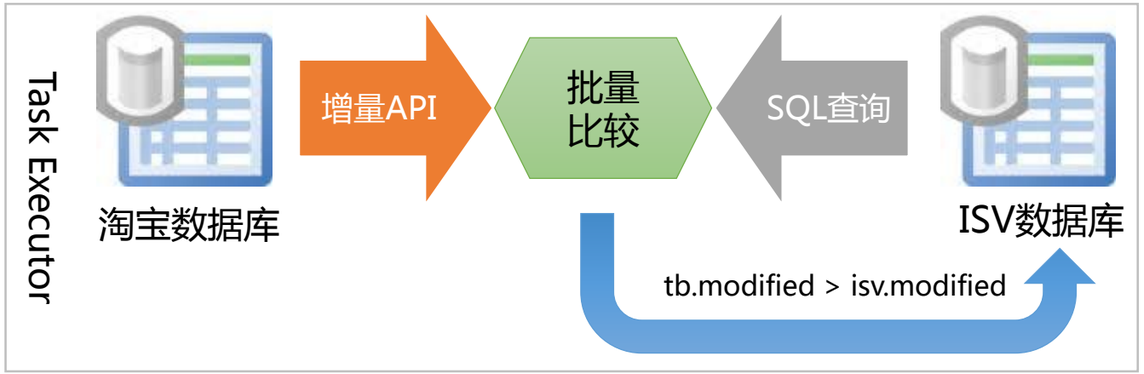


# API增量对账

- 简单可靠的分布式任务调度，多机器并行对账
- 每隔15分钟进行一次对账，平衡业务与DB的压力
- 断点继续，对账途中失败记录成功位点
- 大卖家自动切片查询，避免DB超时
- 数据批量比较，减少RDS查询次数
- 对单超时自动重试3次，避免重新调度



# 通用数据存储模型

## 关键字段

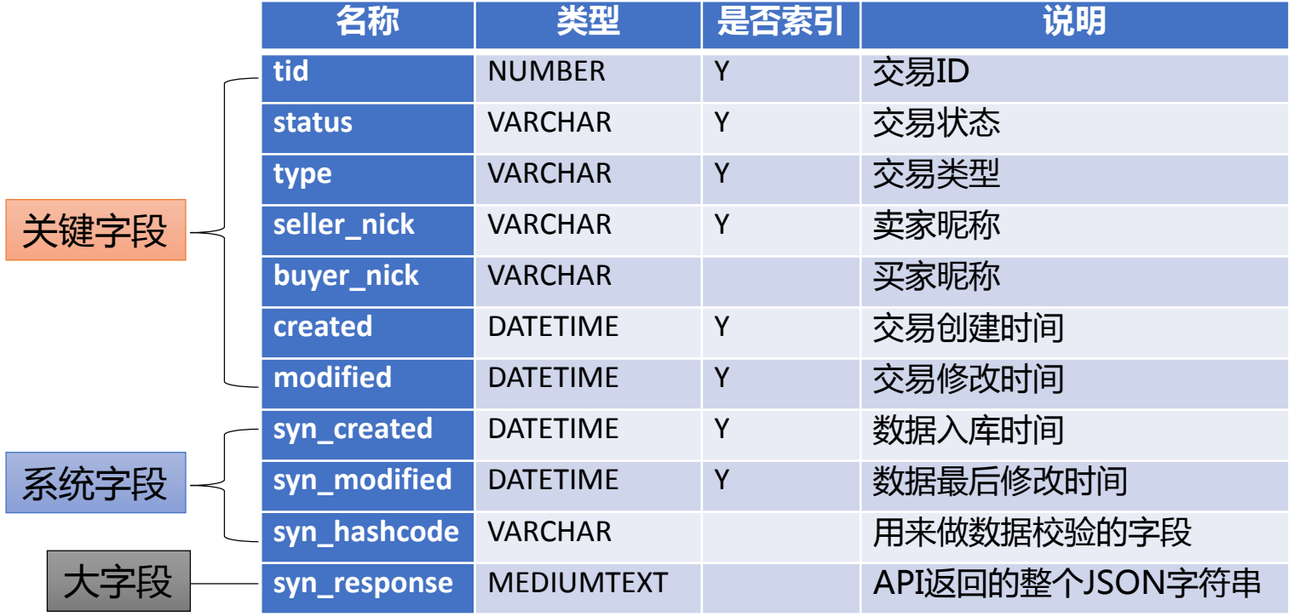
- 业务核心字段，用于查询过滤

## 系统字段

- syn_modified避免分页查询漏单
- syn_hashcode乐观锁、减少DB操作

## 大字段

- 业务详情数据（JSON）
- 可灵活配置返回的字段
- 只需要一份SqlMap
- 避免DDL锁表影响业务




| 名称 | 类型 | 是否索引 | 说明 |
|---|---|---|---|
| tid | NUMBER | Y | 交易ID |
| status | VARCHAR | Y | 交易状态 |
| type | VARCHAR | Y | 交易类型 |
| seller_nick | VARCHAR | Y | 卖家昵称 |
| buyer_nick | VARCHAR | | 买家昵称 |
| created | DATETIME | Y | 交易创建时间 |
| modified | DATETIME | Y | 交易修改时间 |
| syn_created | DATETIME | Y | 数据入库时间 |
| syn_modified | DATETIME | Y | 数据最后修改时间 |
| syn_hashcode | VARCHAR | | 用来做数据校验的字段 |
| syn_response | MEDIUMTEXT | | API返回的整个JSON字符串 |

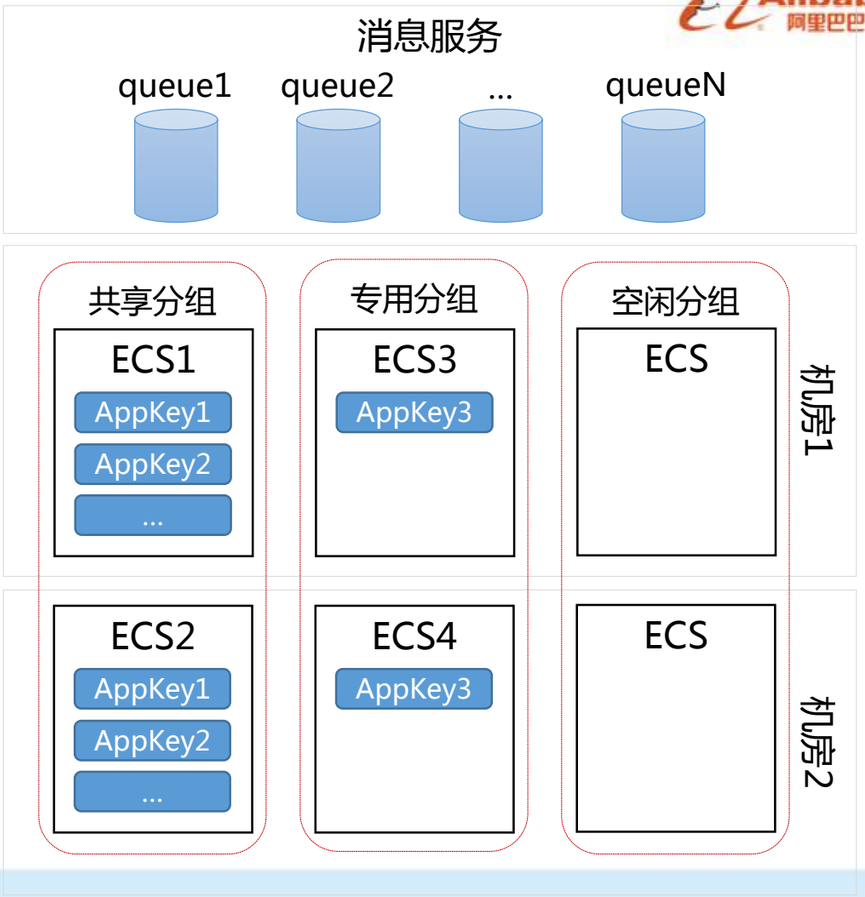
# 资源动态分配与隔离

## 分配

- 新加入的应用自动分配到负载低的分组
- 每个分组至少包含2台机器且分布在不同机房
- 小应用共享分组、大应用专用分组
- 每天定时检查分组QPS，超过一定数值从空闲分组租用机器，反之归还机器

## 隔离

- 消息隔离：消息逻辑队列隔离存储
- 机器资源隔离：每个应用只会分配到一个机器分组
- 系统资源隔离（消息处理线程、对账任务线程、HTTP处理线程、RDS连接池）
- 慢RDS隔离线程池、不可用RDS加黑名单



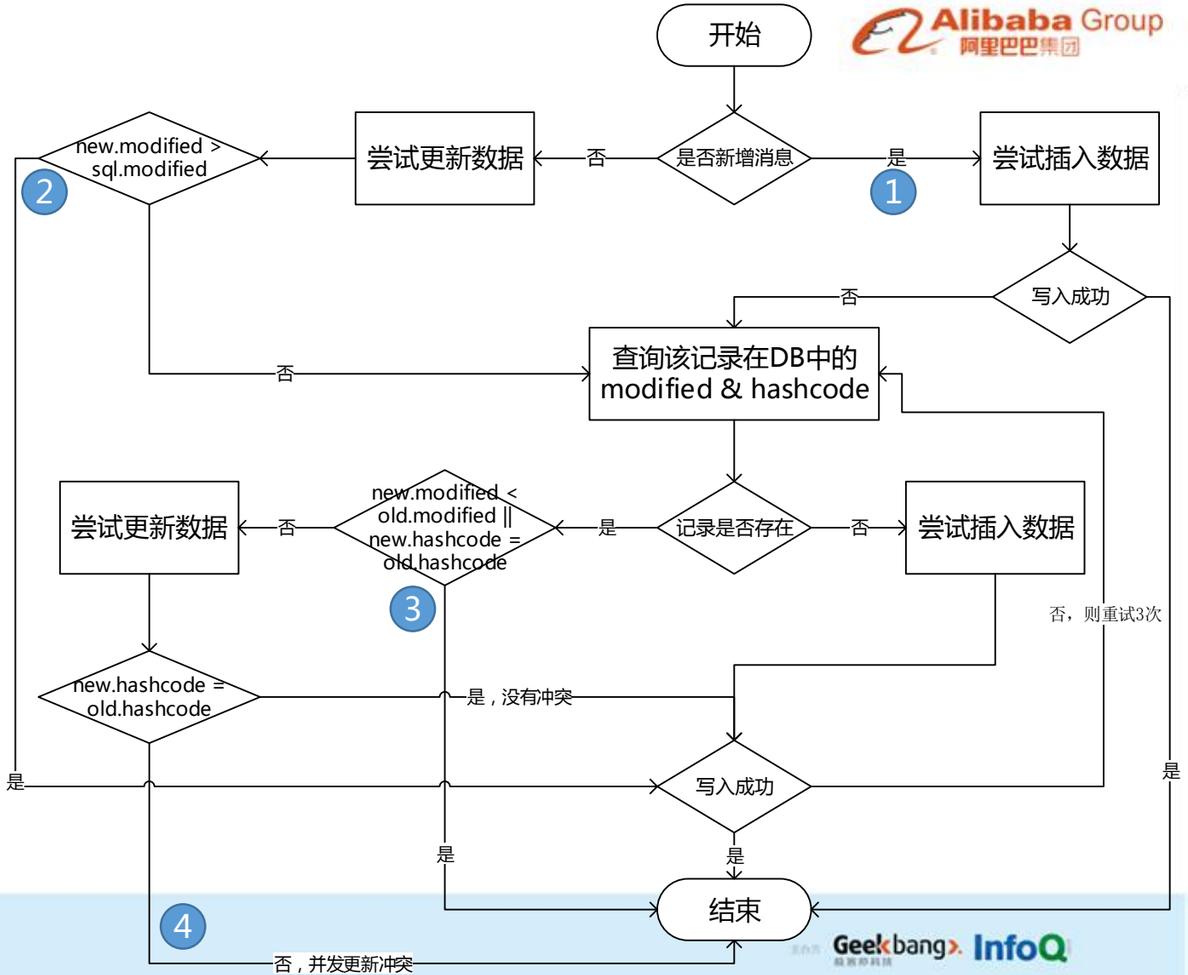
# 如何降低RDS查询与写入开销

## 必要性

- RDS规格普遍比较低，给到平台的连接数有限（只有120）
- 平台与客户共用RDS，需要尽量少占用RDS的资源

| 写入 | 查询 | 删除 |
| --- | --- | --- |
| 1 区分创建和更新事件<br>2 不查询老数据直接更新<br>3 数据内容相同不更新<br>4 乐观锁避免脏数据写入 | • 只查询需要用到的数据<br>• 批量查询DB进行对账 | • 分片删除以缩小区域锁<br>• 凌晨删除避免影响业务 |

优化后减少了90%的DB消耗

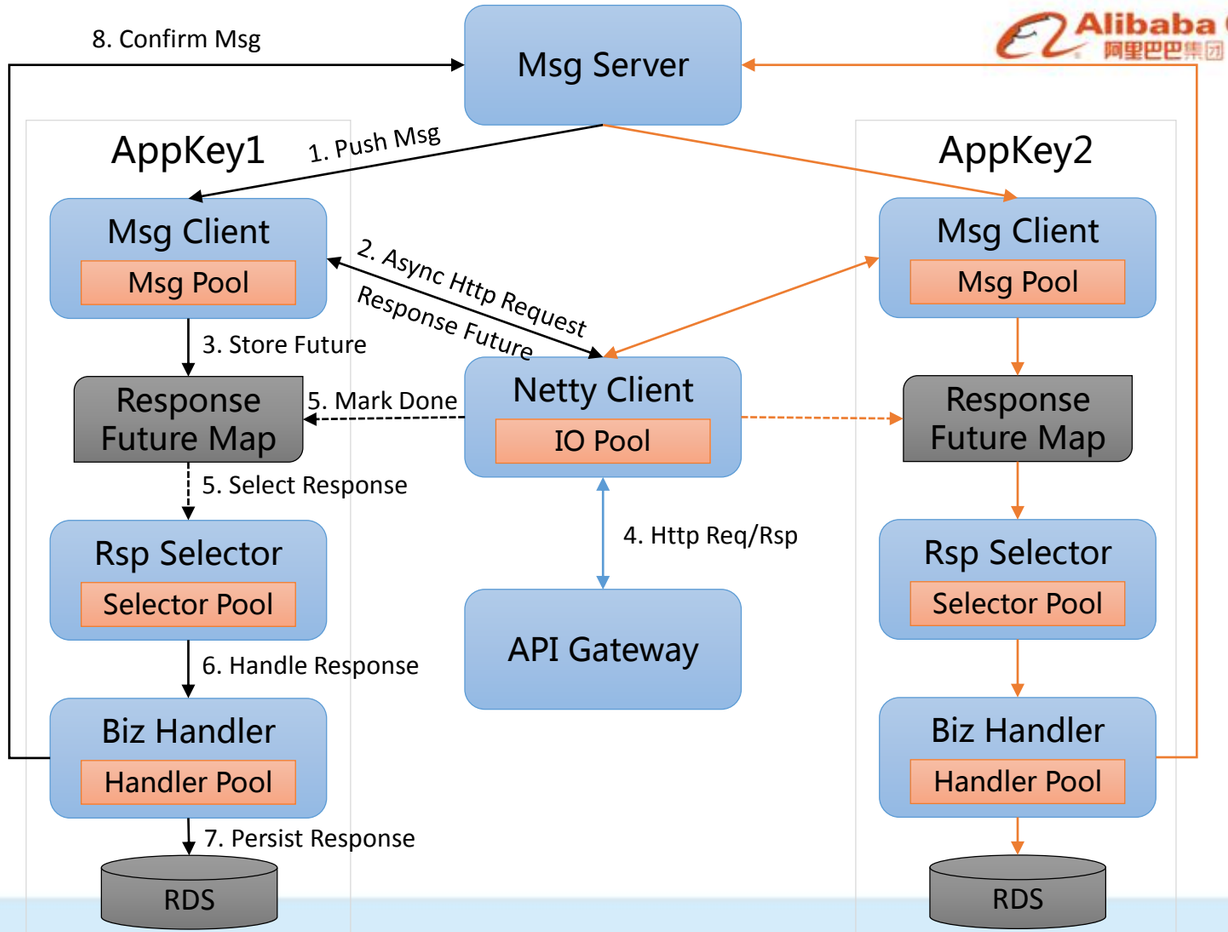


# 消息处理异步化

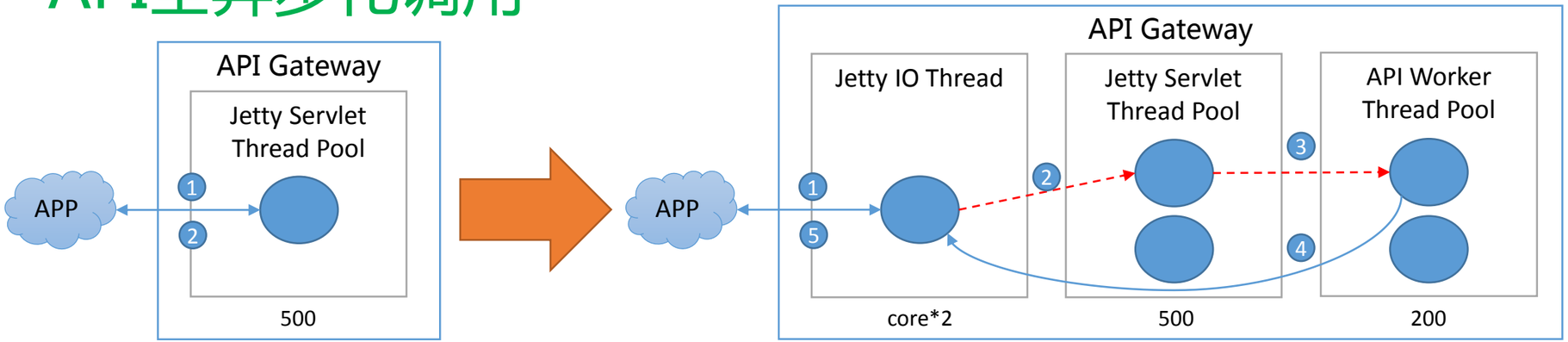
## 隔离一切由等待引起的线程阻塞

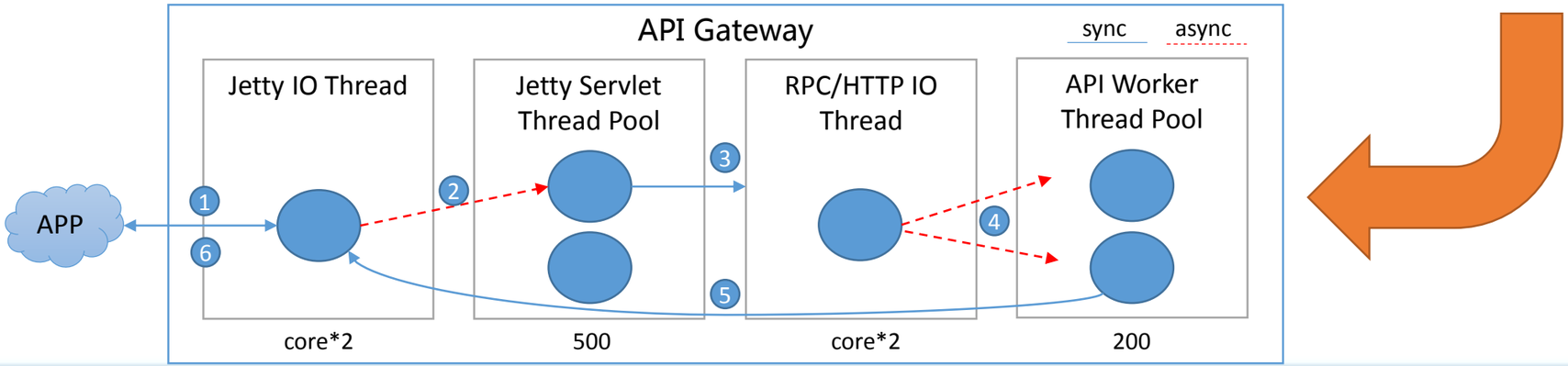
- 前提：消息异步化确认
- 不阻塞Msg Client线程
- HTTP API 请求异步化
- 不阻塞Netty的IO线程
- 隔离慢HTTP请求引起的线程阻塞
- 共享无阻塞线程池，独享业务处理线程池

优化后4核8G的机器QPS由500提升到2000



# API全异步化调用

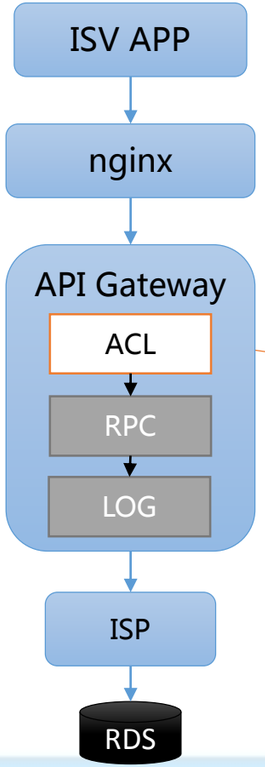


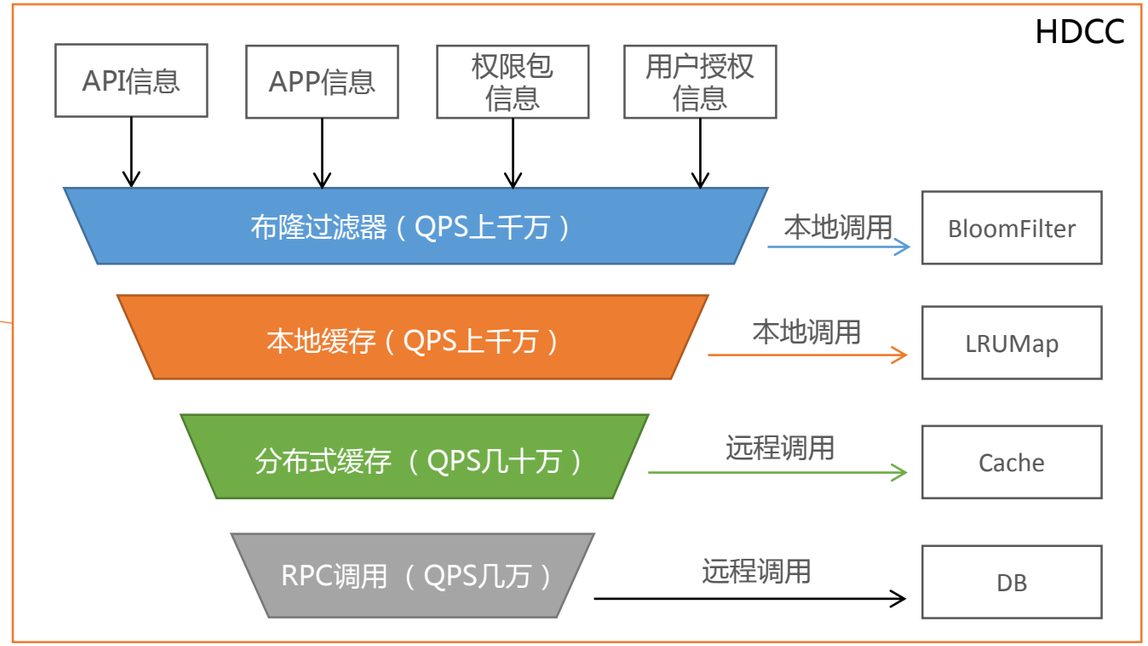




## 异步化的好处

- 释放网络等待引起的线程占用
- 彻底隔离API请求之间的影响
- 线程数不再成为网关的瓶颈
- 慢API不会引起网关的不稳定

# API无阻塞调用



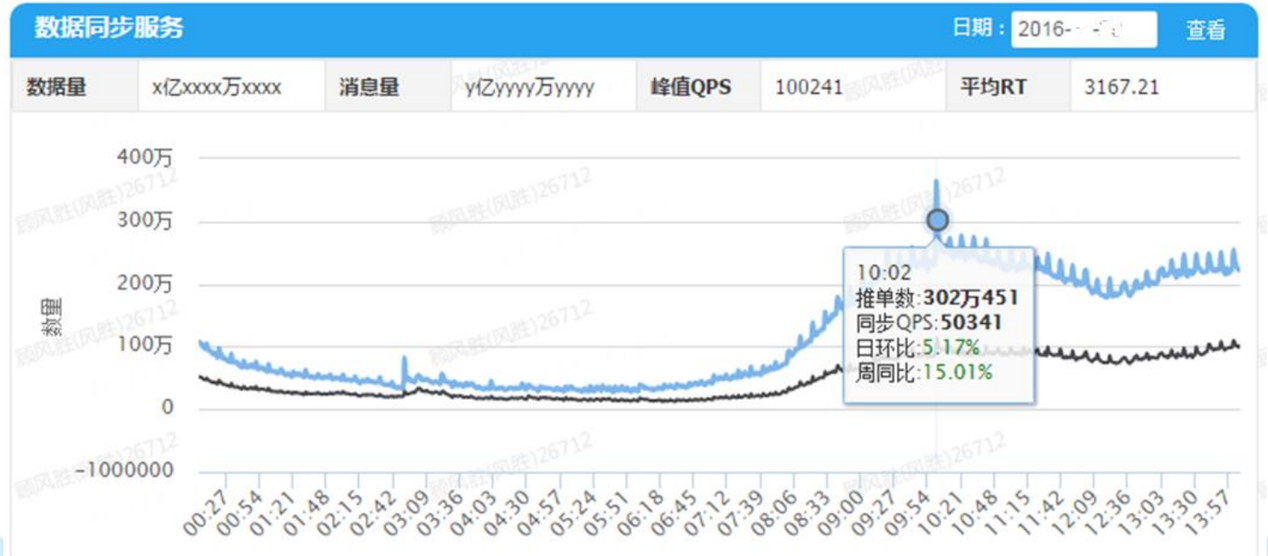




## 无阻塞的好处

- 消灭99.9%的IO等待
- CPU只用于计算资源
- 网关耗时降低到1~2毫秒
- 较少的DB资源支持高并发

# 运维与监控

- 运维
  - 阿里EWS，批量发布、弹性扩缩容
  - ISV同步问题自助排查工具
  - 数据同步轨迹查询
  - 对账任务调度控制台
- 监控
  - 系统监控-阿里云ECS控制台
  - 业务监控-实时同步QPS
  - ECS集群健康查询与告警
  - 对账大盘延迟告警

# 双十一保障过程

## 大促前

- 强弱依赖梳理
- 单机压测
- 性能优化（提升30%）
- 容量评估
- 机器扩容
- 全链路压测
- 服务商能力/业务分级
- 提前预案执行（减压）

## 大促中

- 系统监控（CPU/Load/RT...）
- 业务监控大屏
- 用户问题反馈群
- 集中办公与24小时值班
- 紧急预案执行
- 资源调配（机器/API/MSG）
- 应急处理流程与小组

## 大促后

- 预案恢复
- 大促复盘
- 用户回访
- 制定产品开发计划
- 制定技术优化方案


# THANKS

**更多阿里技术干货**
**关注“阿里技术”官方微信公众号**