

# 淘宝数据魔方的系统架构

殷琳君（长林）

淘宝网

# LAMP人 主题分享交流会

[www.LAMPER.cn](http://www.LAMPER.cn)

QQ群：3330312

<http://weibo.com/lampercn>

淘宝网

## Agenda

- 数据产品总体架构
- 分布式MySQL集群
- NoSQL存储与计算
- 统一的数据中间层
- 通用数据报表框架

## 每天的数据

### □ 淘宝主站：

- 30亿店铺、宝贝浏览
- 千万量级交易笔数

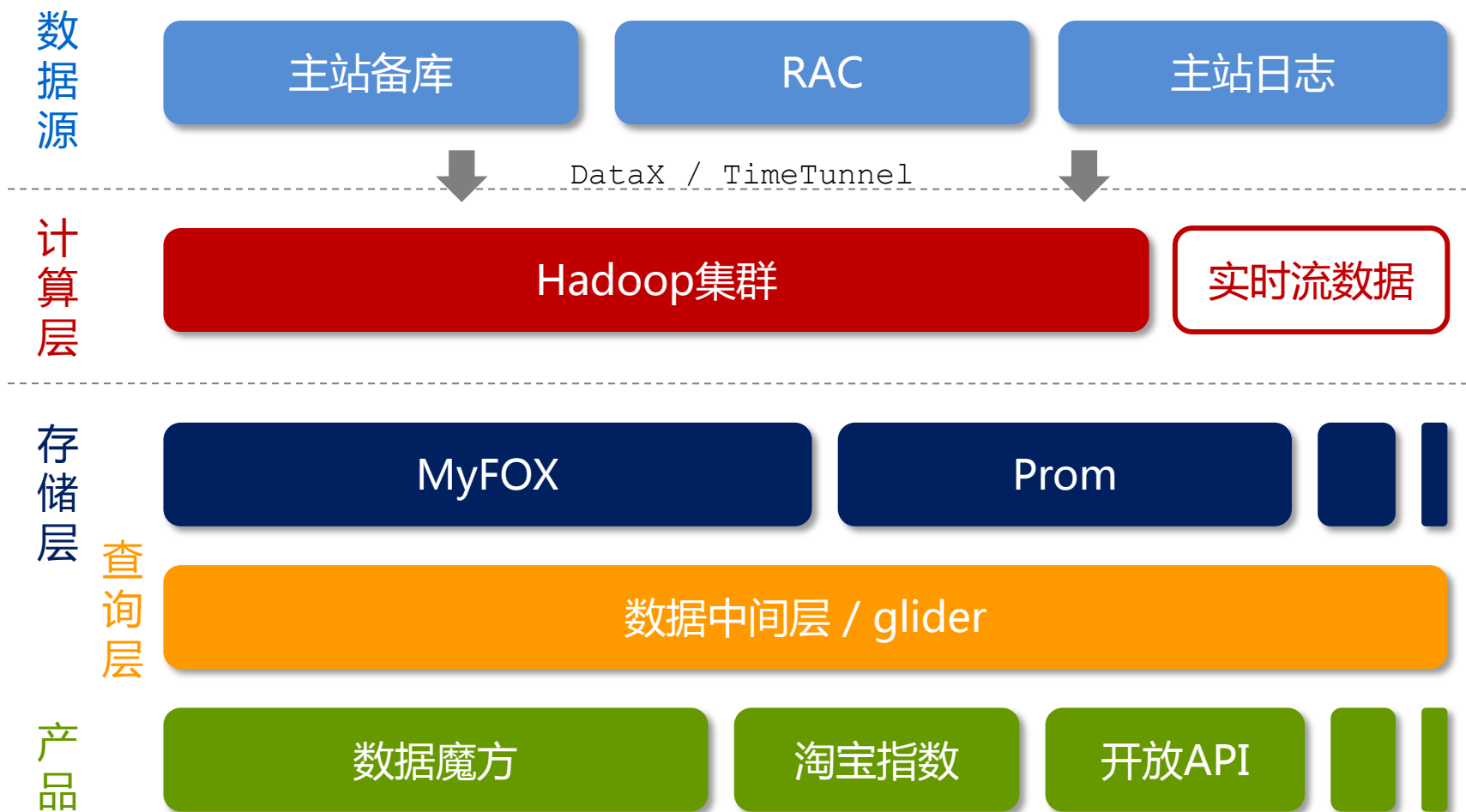
### □ 数据产品：

- 60G统计汇总结果
- 千万量级数据查询请求

# 海量数据带来的挑战

- 计算
- 存储
- 读写

# 架构总览



# 分布式MySQL集群—MyFOX

需求：

- SQL查询
- 海量存储
- 可横向扩展
- 对应用透明
- 兼顾性能

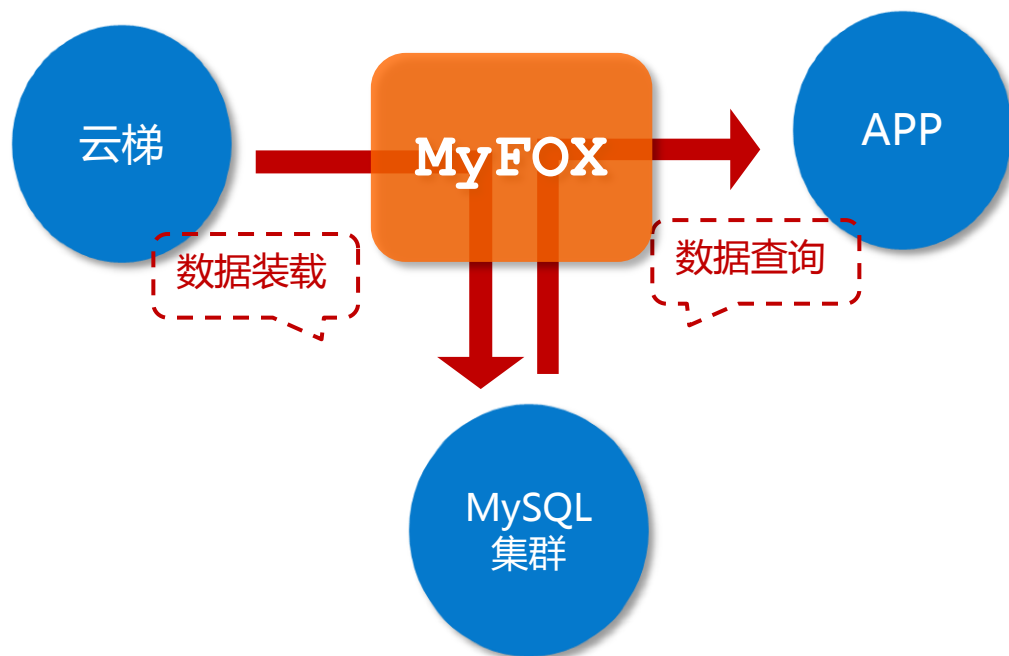
# 分布式MySQL集群—MyFOX

## □ 分库分表

- 基于业务特点

## □ 透明的中间层 ( MyFOX )

- 查询代理
- 数据装载
- 集群管理





# MyFOX一分片规则

## □ 冗余复制

- 小表
- 访问频繁
- JOIN

## □ 条目切割

- 按路由字段的值，每N行切片
- 路由字段是一级索引
- 分散压力、并行查询

# 示例：条目切割

## □ 切片

- 阈值 ( 200W )
- 上浮动 ( 5% )

```
thedata=20100816, tid=11 ^A2090000  
thedata=20100816, tid=12 ^A2120000  
thedata=20100816, tid=13 ^A760000  
thedata=20100816, tid=14 ^A289
```

```
thedata=20100816, tid=11 ^A2090000  
thedata=20100816, tid=12 ^A2000000  
thedata=20100816, tid=12 ^A120000  
thedata=20100816, tid=13 ^A760000  
thedata=20100816, tid=14 ^A289
```

## □ 装桶

- 一个桶装满再开新桶
- “桶” 即实际的物理表

```
thedata=20100816, tid=11 ^A2090000  
thedata=20100816, tid=14 ^A289
```

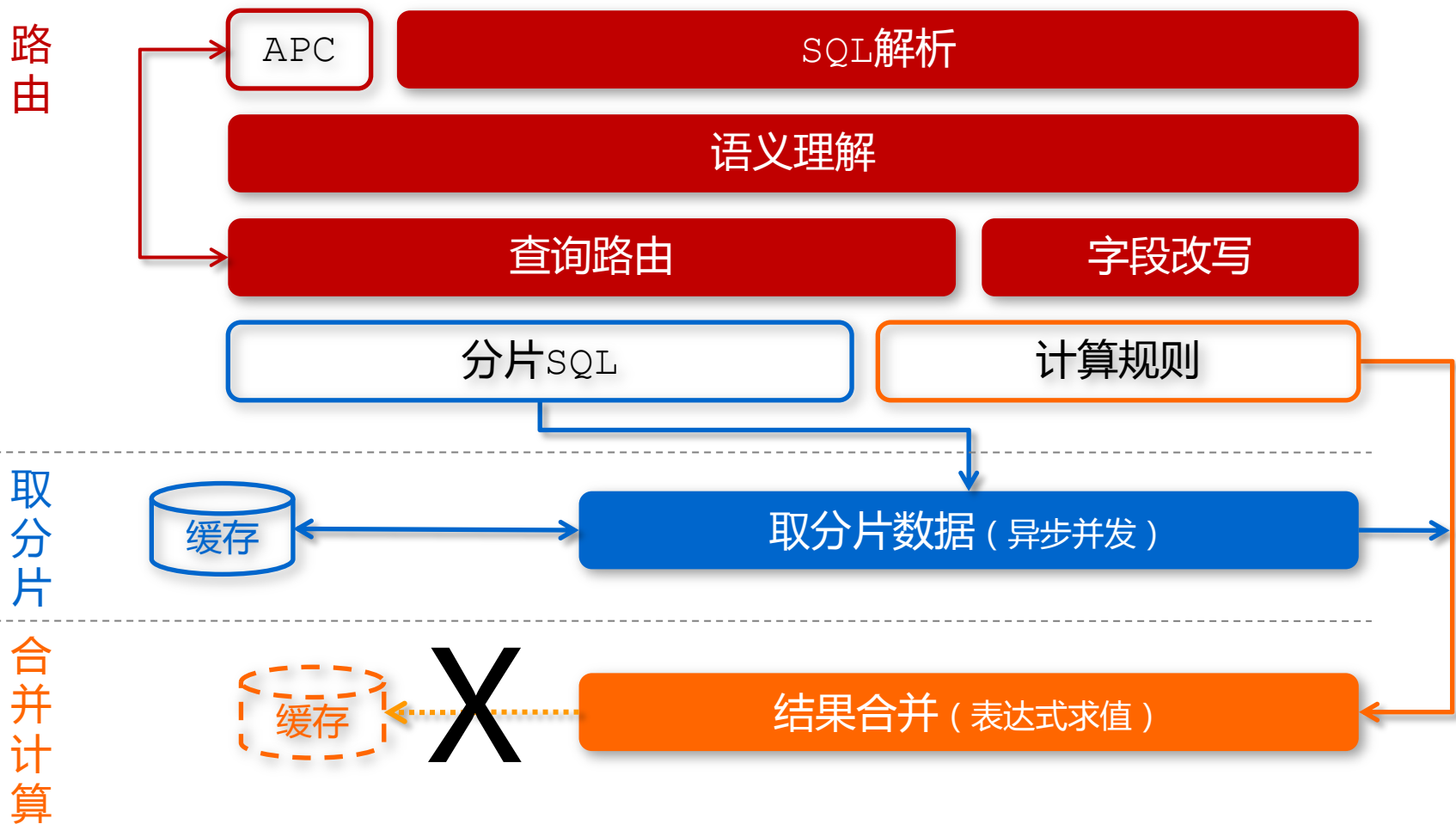
```
thedata=20100816, tid=12 ^A2000000
```

```
thedata=20100816, tid=13 ^A760000  
thedata=20100816, tid=12 ^A120000
```

# MyFOX—数据查询

```
SELECT IF(INSTR(f.keyword,' ') > 0, UPPER(TRIM(f.keyword)),  
          CONCAT(b.brand_name,' ',UPPER(TRIM(f.keyword)))) AS f0,  
          SUM(f.search_num) AS f1,  
          SUM(f.uv) AS f2,  
          ROUND(SUM(f.search_num) / SUM(f.uv), 2) AS f3,  
          AVG(f.uv) AS f4  
  
FROM f  
  
INNER JOIN dim_brand b ON f.keyword_brand_id = b.brand_id  
  
WHERE f.keyword_type_id = 1 AND f.keyword != ''  
        AND keyword_cat_id IN ('50002535')  
        AND thedate <= '2011-03-10'  
        AND thedate >= '2011-03-08'  
  
GROUP BY f0  
  
ORDER BY SUM(f.search_num) DESC LIMIT 0, 1500
```

# MyFOX—数据查询



# MyFOX路由层—语义理解

**WHERE** thedate <= '2011-03-10'

**AND** thedate > '2011-03-07'

**AND** toprank\_id IN (2, 3)

	2	3
2011-03-08	{"toprank_id":"2", "thedata":"2011-03-08"}	{"toprank_id":"3", "thedata":"2011-03-08"}
2011-03-09	{"toprank_id":"2", "thedata":"2011-03-09"}	{"toprank_id":"3", "thedata":"2011-03-09"}
2011-03-10	{"toprank_id":"2", "thedata":"2011-03-10"}	{"toprank_id":"3", "thedata":"2011-03-10"}

# MyFOX路由层一字段改写

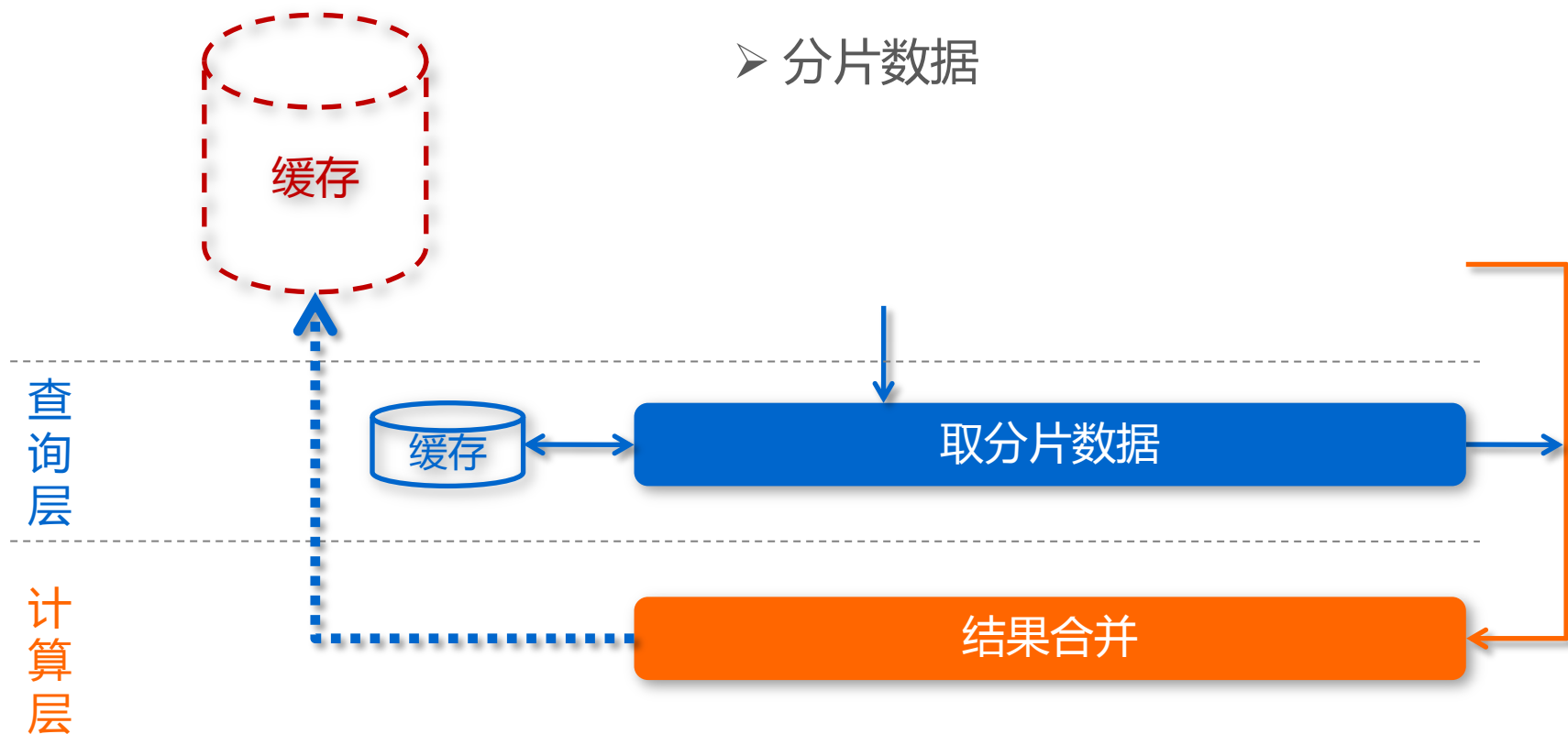
```
SELECT a AS f0,  
       SUM(f.search_num) AS f1,  
       SUM(f.uv) AS f2,  
       ROUND(SUM(f.search_num) / SUM(f.uv), 2) AS f3,  
       AVG(f.uv) AS f4
```

- AVG ( a )
- 1 + SUM(a)
- SELECT a FROM ... ORDER BY b
- 重复查询列

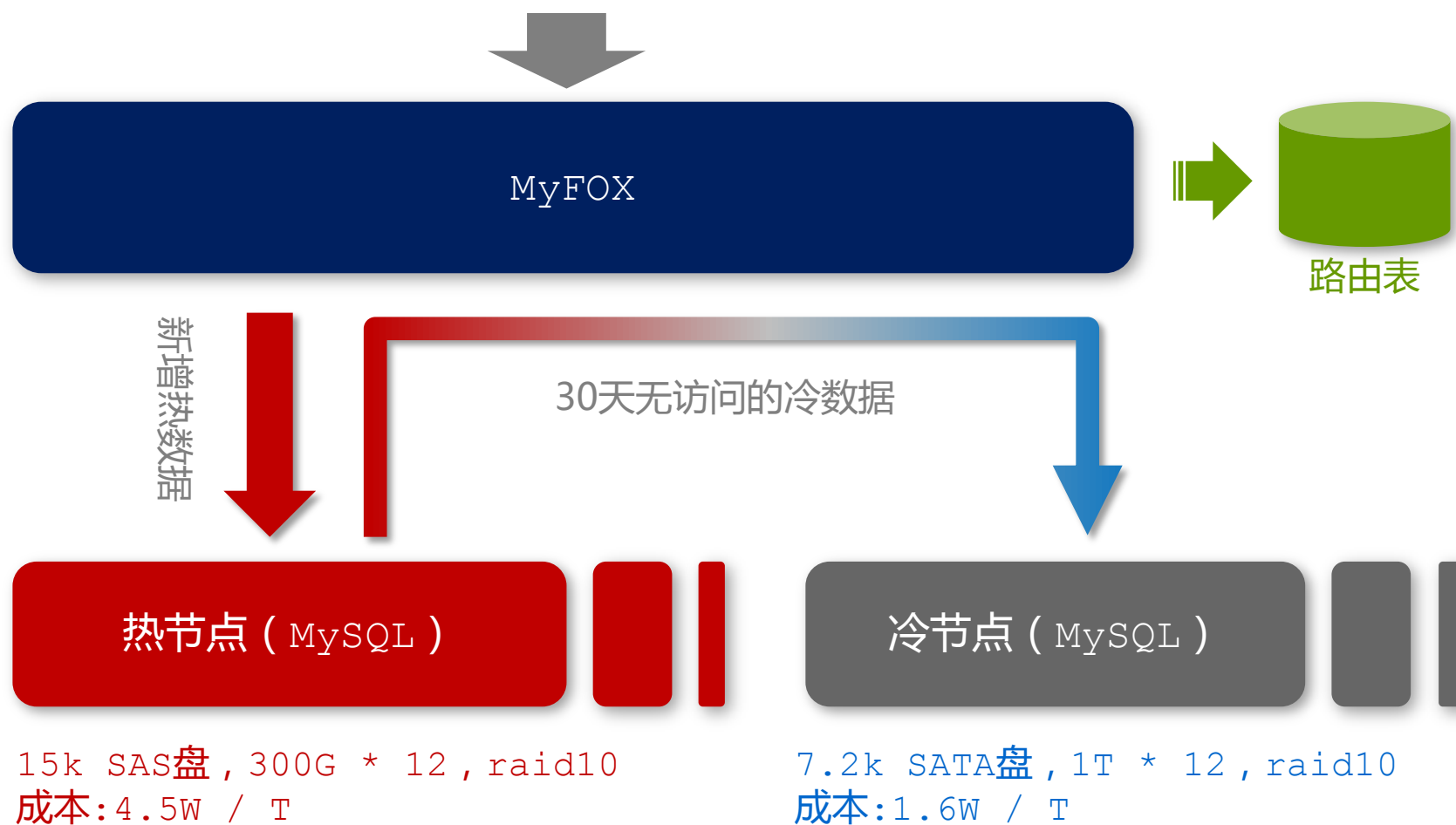
# MyFOX缓存

- 缓存哪部分数据？

- 最终结果
- 分片数据



# MyFOX—数据装载





# 这个问题能解决吗

查询条件

所有分类 > 笔记本电脑

查看

取消

品牌

Q

输入您要查询的品牌

\*\*热门品牌\*展示该类目下50个热销品牌，您可以在搜索框中查询全部品牌

热门品牌: ThinkPad Apple/苹果 Lenovo/联想 Asus/华硕 Dell/戴尔 Acer/宏基 HP/惠普 other/其它 Samsung/三星 Sony/索尼

\*属性选择目前可查看最近7天内的数据

笔记本尺寸: 11寸 5寸 7寸 8寸 9寸 10寸 12寸 13寸 14寸 15寸

笔记本定位: 商务定位 便携定位 家庭影音 女性定位 学生定位 游戏娱乐 迷你定位 入门定位

硬盘容量: 20G 30G 40G 60G 80G 40G以下 120G 160G 200G 10G

重量: 1公斤以下 1-1.5公斤 1.5-2公斤 2-2.5公斤 2.5公斤以上

颜色分类: 透明 花色 褐色 白色 黄色 红色 酒红色 紫色 浅灰色 绿色

蓝牙功能: 无 有

指纹功能: 无 有

迅盘技术: 无 有

显卡显存容量: 3G 950M 共享内存容量 256M 512M 128M 64M 1G 32M以下 32M

内存容量: 3G 32G 256M 512M 128M 64M 64M以下 1G 4G 2G

笔记本CPU: 酷睿四核 酷睿2至尊 奔腾M(Dothan) 奔腾双核 炫龙64 X2 移动奔腾3 奔腾4 苹果G4 苹果G3 速龙64

光驱类型: 无 CD-ROM DVD-ROM COMBO(康宝) CD-RW DVD刻录 其它光驱 蓝光机 DMR

## Prom—应用场景

- 数据量很大
- 无法穷举所有可能的查询条件组合
- 大量穷举出来的条件组合无意义

## Prom—算法

13寸、商务定位的笔记本昨天的交易金额是多少？

- 1、查询13寸笔记本昨天的交易ID
- 2、查询商务定位笔记本昨天的交易ID
- 3、求交易ID交集
- 4、根据3中的交易ID查询明细数据，按照金额进行汇总

# Prom—第一版Redis

## □ Redis

- Key-Value的内存数据库
- 支持List、Set等数据结构

## □ 瓶颈

- 大量随机读取明细
- 内存受限，数据持久化问题

# Prom—第二版HBase

## □ 定制化的存储

- HBase建立在hdfs之上，和Hadoop无缝集成
- 可水平扩展

## □ 实时计算

- 百万记录

# Prom—HBase表设计

Row-key	列族1 ( 交易ID , 索引 )	列族2 ( 交易明细 )
笔记本尺寸 : 13寸	1 , 2 , 4 , 9	{58, 600,35}, {72,999,14}, {92,103,77}, {36,702,101}
笔记本定位 : 商务定位	1 , 4 , 7	{58, 600,35}, {92,103,77}, {85, 442, 152}

# Prom—HBase实时计算

求SUM(alipay)

属性	属性值
笔记本尺寸	13寸
笔记本定位	商务定位

查索引



节点1	1, 2, 4, 9
节点2	1, 4, 7

求交集



节点2	1, 4
本地SUM运算 (Hbase扩展)	

汇总计算  
写入缓存



## Prom—小结

- 历史数据的实时计算
- 通用性强，支持sum, count, avg, group by, sort
- 空间换时间
  - 变大量随机读为顺序读
  - 避免明细数据网络传输



# SQL与NoSQL

- 互为补充，适用场景不同
- SQL
  - 应用开发更简单
  - 支持跨行跨表事务
  - 运维更成熟
- NoSQL
  - 水平扩展性更好
  - 更适合解决海量数据存储与计算

# 统一的数据中间层—Glider

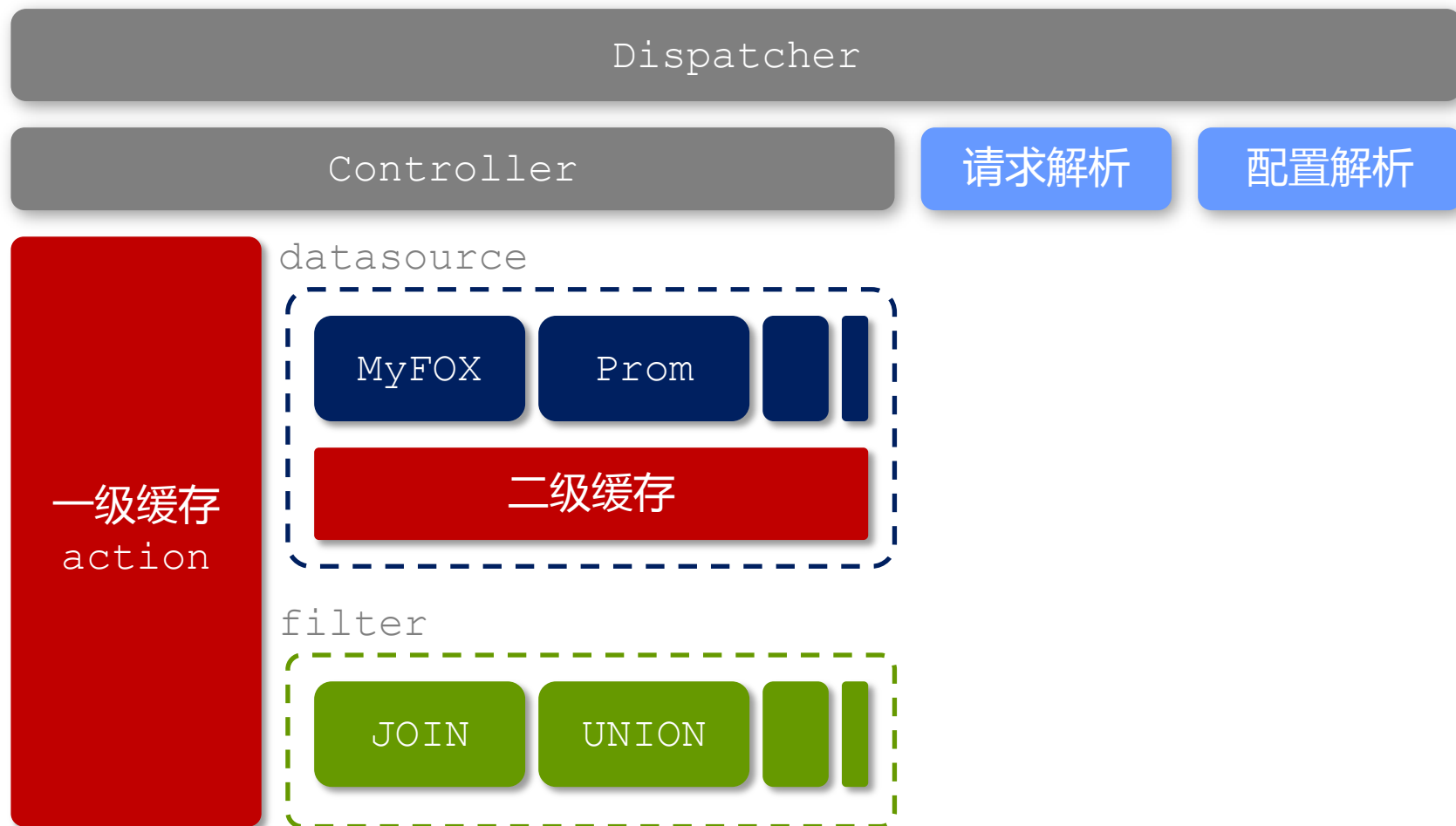
- 统一数据出口

- 类SQL
- HTTP REST
- JSON返回

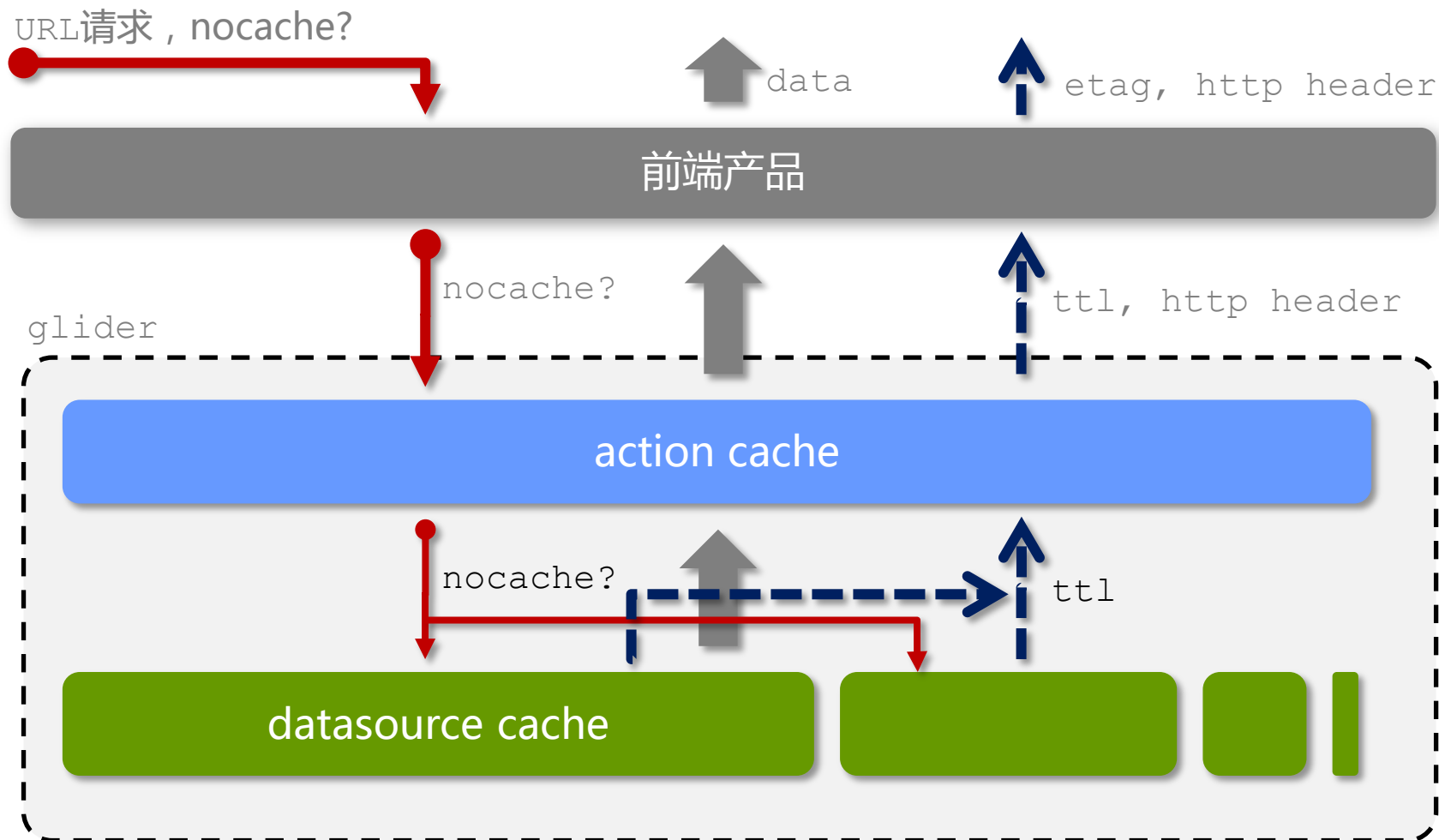
- 前后端解耦

- 数据缓存管理

# Glider架构



# Glider缓存管理



# Glider—小结

## □ 用中间层隔离前后端

- 底层架构对前端透明
- 水平可扩展性

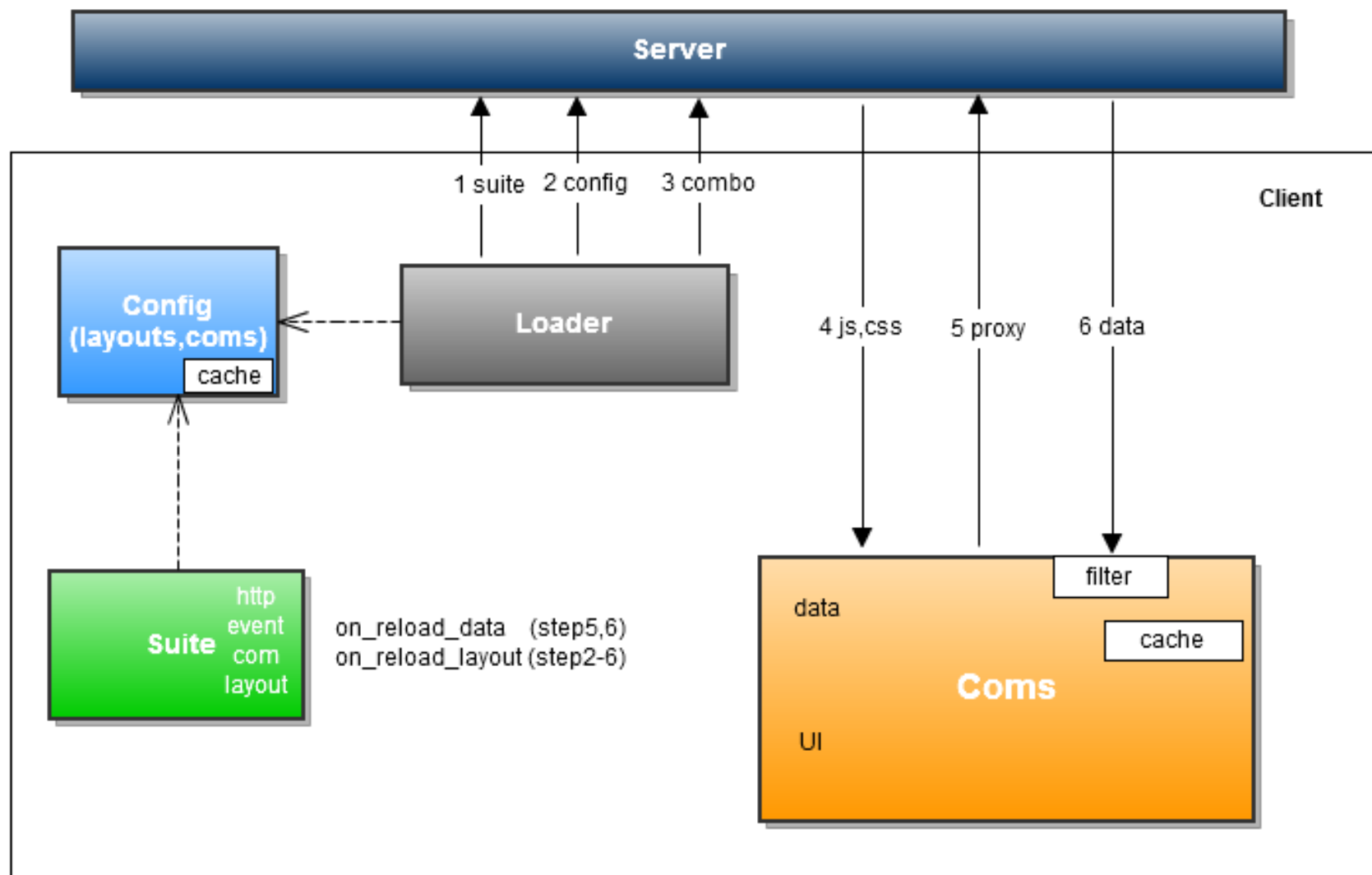
## □ 缓存是把双刃剑

- 降低后端存储压力
- 数据一致性问题

# 通用数据报表框架—Cubex

- 通过配置生成报表，快速开发
- 与组件化、事件驱动的JS框架集成
  - 基础服务封装：http, event
  - 可复用的组件库
  - 统一的组件生命周期管理，易于维护

# 前端JS框架



## 回顾

数据源

用户、店铺、商品库

收藏夹

交易表

...

日志

数据传输

计算层

Hadoop集群

实时流数据

存储层

MySQL

HBase

查询层

数据中间层

产品

数据魔方

淘宝指数

开放API





**谢谢!**