



Build High Performance Weibo System

@TimYang

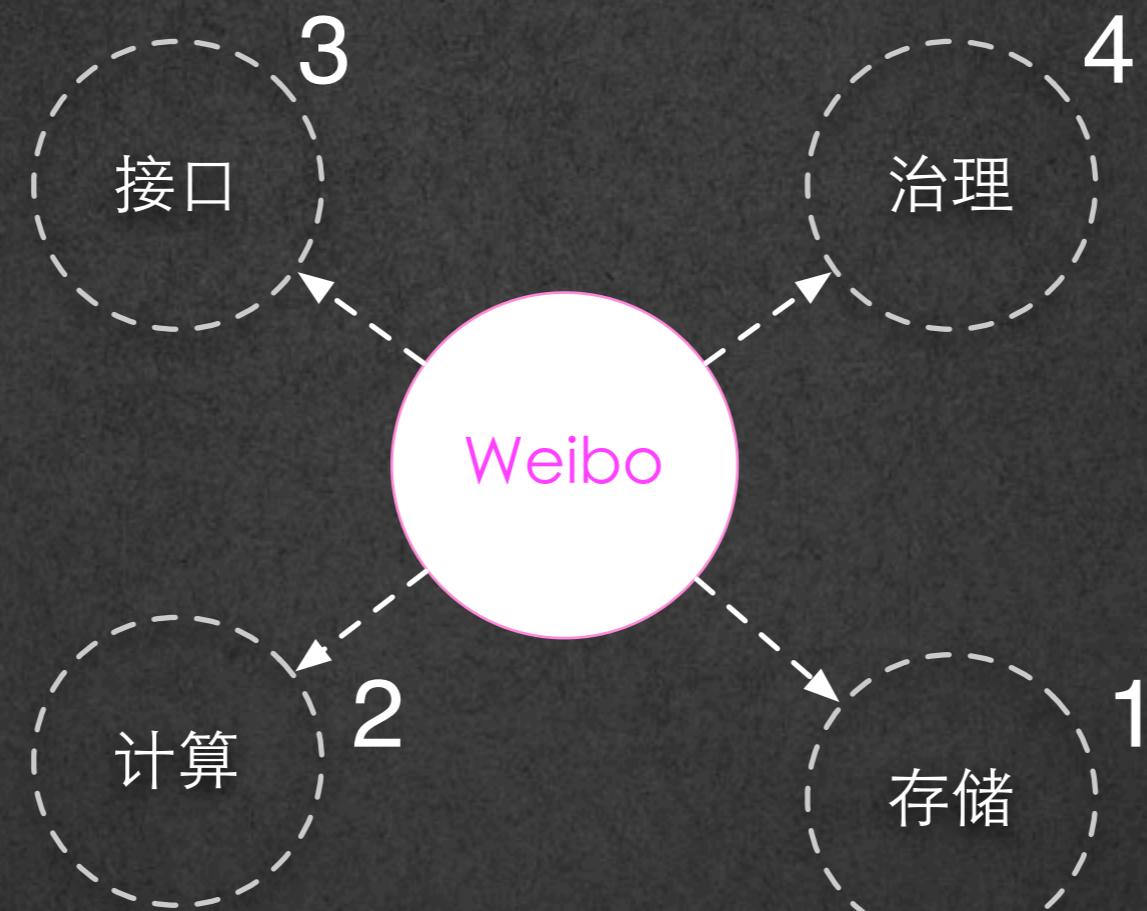


- 微博是什么？
 - 140字
 - 根据关注关系实时投递
 - 支持评论、转发及图片、视频等媒体

相关演讲

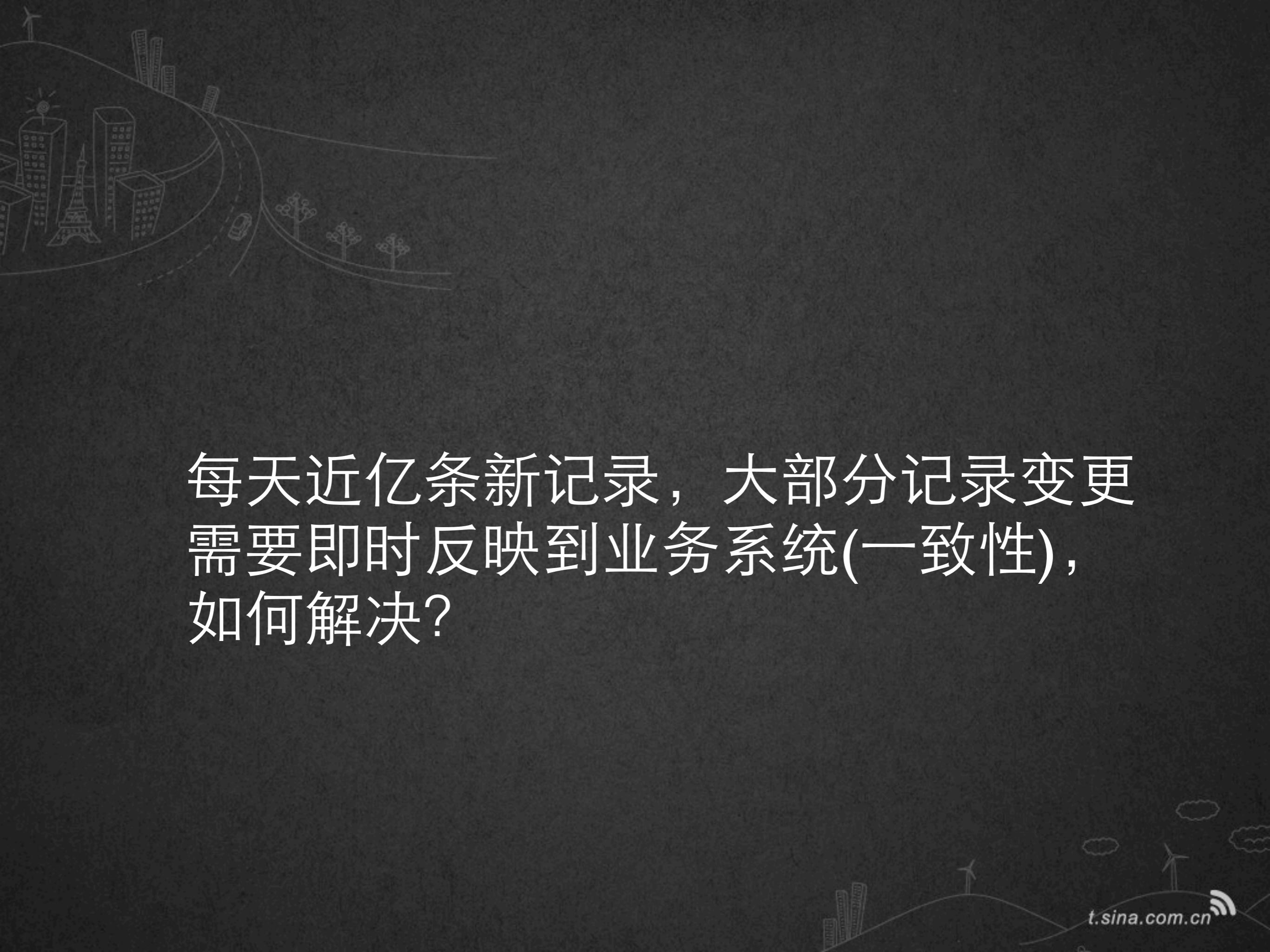
- 《构建可扩展的微博系统》 QCon 2010
- 《微博架构与平台安全》 WDC 2010

Agenda



Part I

海量存储



每天近亿条新记录，大部分记录变更
需要即时反映到业务系统(一致性)，
如何解决？

MySQL

- 适合海量存储，可不断拆分扩展
- 久经考验

拆分策略

- 按 hash 拆分
- 多维度，如关系
- 多级索引，如 user_timeline index

问题

- 单机性能限制、需要持续拆分
- 访问速度需求
 - 用户
 - 关系，每秒 5 万次
- 延迟

MySQL + cache 问题

- 雪崩
- 一致性
- cache数据类型不够，需要序列化
- 额外开销
- 添加及修改问题

NoSQL

“Databases are specializing – the “one size fits all” approach no longer applies.”

MongoDB设计哲学

NoSQL?

- 单机思路
 - Redis
 - MongoDB
- 分布式思路
 - Cassandra
 - HBase



用好一款开源产品的前提条件是深入了解它的定位。

特性比较

- MongoDB
- Redis
- HBase
- Cassandra

Redis - 持久方式

- 持久化存储模式理解
 - snapshot - 主流
 - vm - 作者放弃
 - diskstore - 作者新方向
 - aof - 重建慢

Redis - 数据类型

- string: key value redisObject 16 bytes/item
- list: 双向链表 40 bytes / item
- hash: zipmap(<64)
- set/sorted set

Redis - Replication

- 重读rdb文件问题
- 代码实现简单

Redis - 容量规划

Redis 容量评估表

需求	
业务名称	
用途	
数据类型 (List, String, Hash)	Hash
单位(user)	用户
当前单位容量	200 字节
预估最大单位容量	500 字节
当前数量	1 billion
预估最大数量	5 billion
当前峰值(rps 读)	20k
当前峰值(rps 写)	5k
预估最大峰值(rps 读)	100k
预估最大峰值(rps 写)	20k
方案	
需要端口数	
需要内存	32G
备注	
景深	
墨盒包装	35G
墨盒调用数	

Redis - 定位

- 需要高速读写访问
- 能容忍短期不可用，无成熟failover方案
- 有list/set数据结构需求(optional)

海量存储

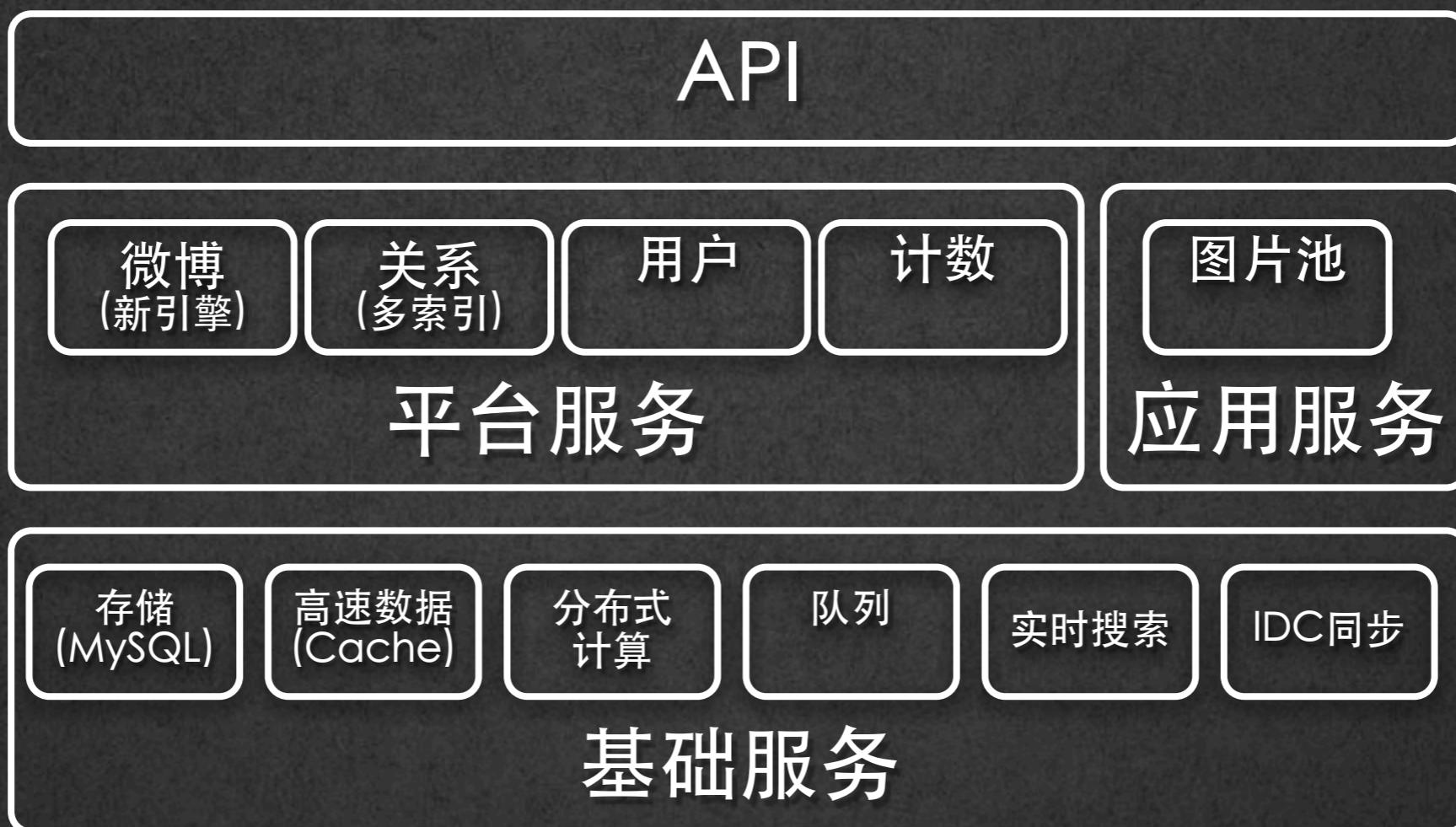
- MySQL，久经考验的海量存储
- NoSQL，填补MySQL与cache之间空隙，但是需要有合适的驾驭能力

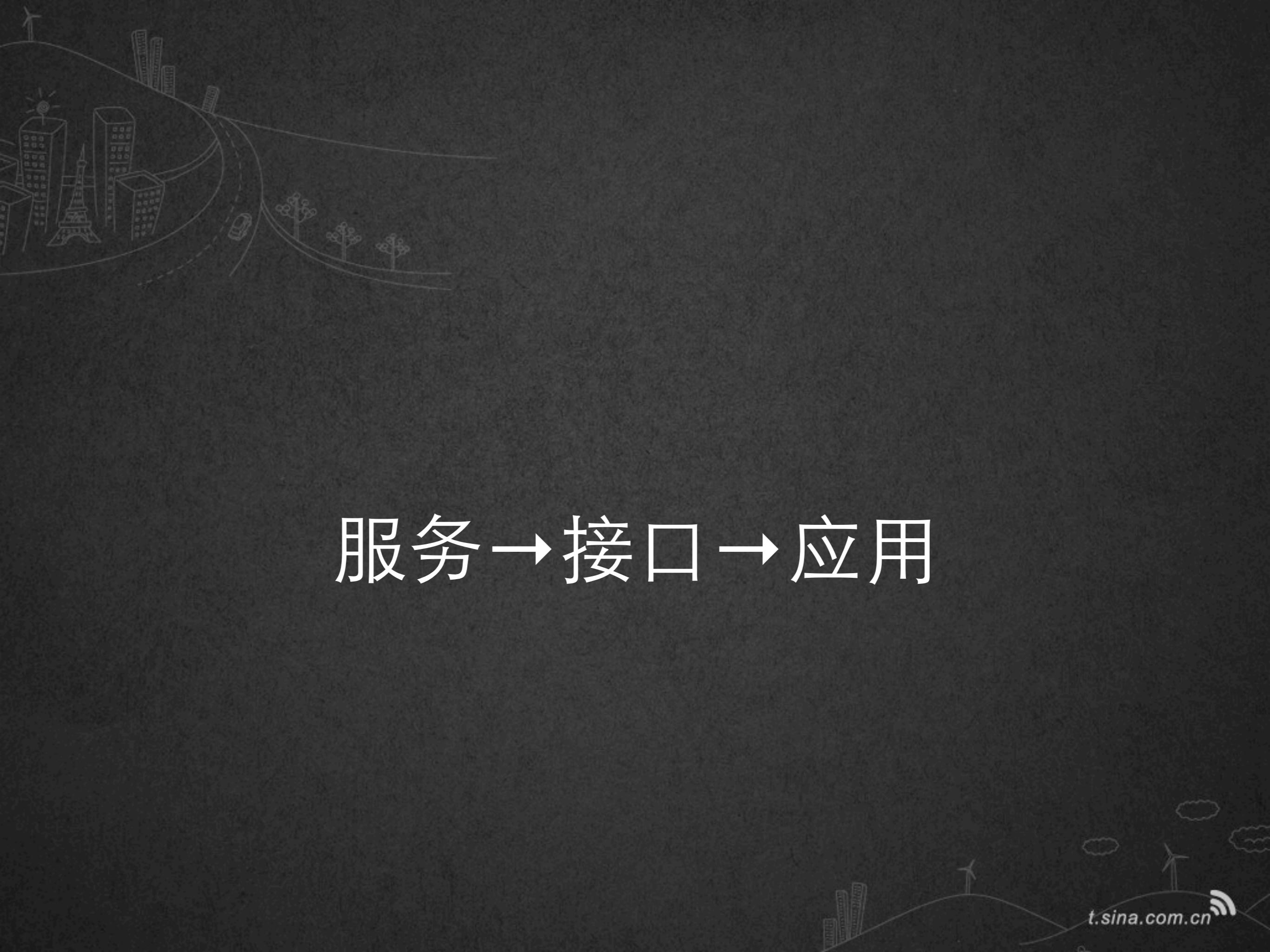
Part 2

实时计算

微博架构

(第三版)





服务 → 接口 → 应用

问题一

大部分Web系统瓶颈在于cache数据访问，仅用压缩是否足够？

可用的cache资源放不下所有热点数据，导致数据加载日趋缓慢

- 扩容？
- 减负？



“Web 系统用 *json* 来存储及 *cache* 非常浪费。一条微博用 *json* 数据结构来存所有字段(包括作者信息), 需要2~5K字节左右, 用 *xml* 需要10k左右, 用 *protobuf* 序列化后, 大约只有500字节。”

3月6日 00:47 转发(495) | 评论(134)

历程

RDBMS→

Key value (JSON)→

Protocol buffers(binary)

JSON 烦恼

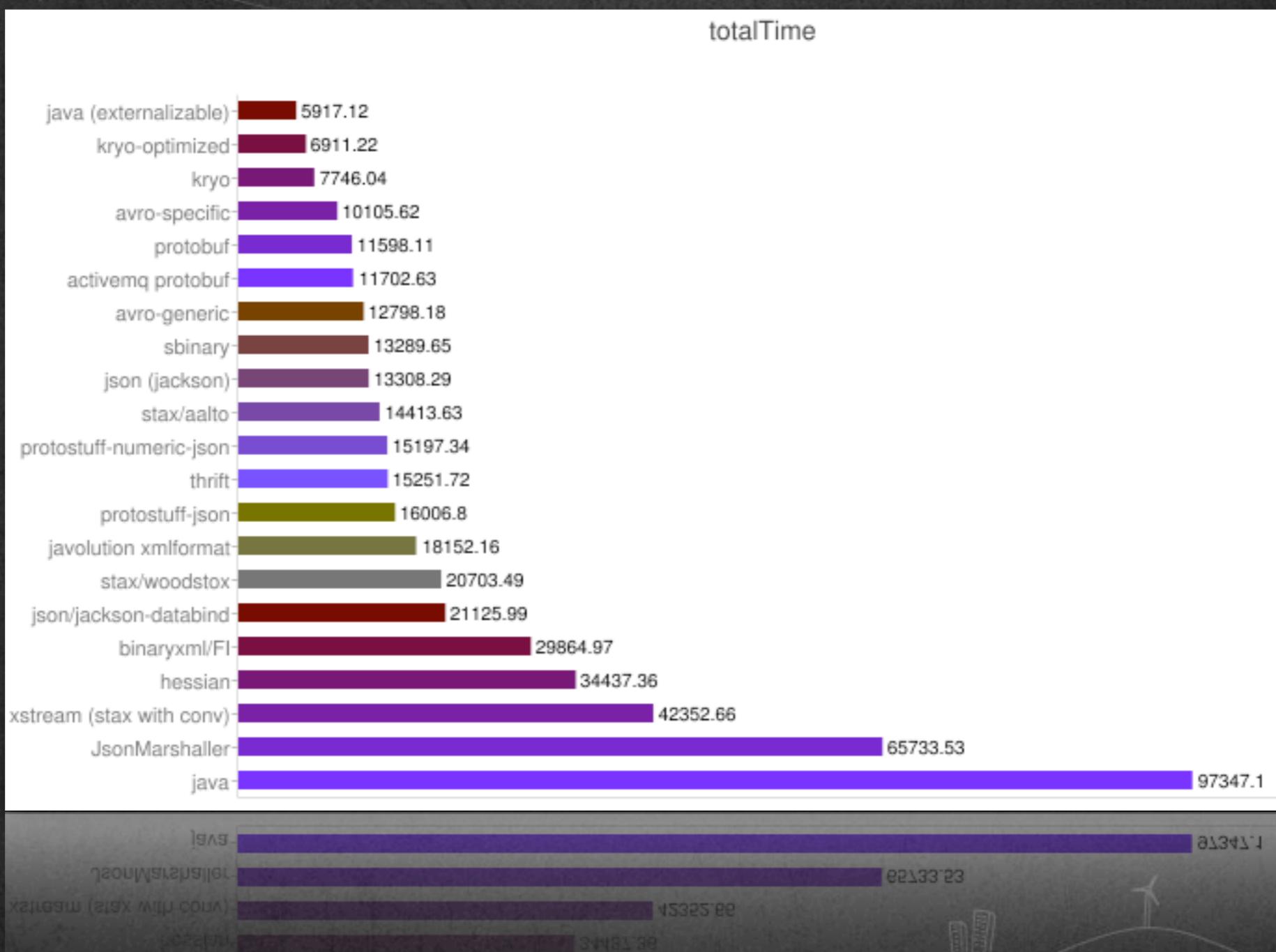
- DB
- Cache
- Message Queue
- API

PB 优势

- Numeric: varint, from 1 byte
- 仅传输编号，不传输字段名称
- 支持多语言：Java, C++, Python...

- 编解码高效
- 从 cache 最终结果到中间结果，进一步节约空间
- “192.168.0.1” → “0xc0a80000” → varint

Benchmark

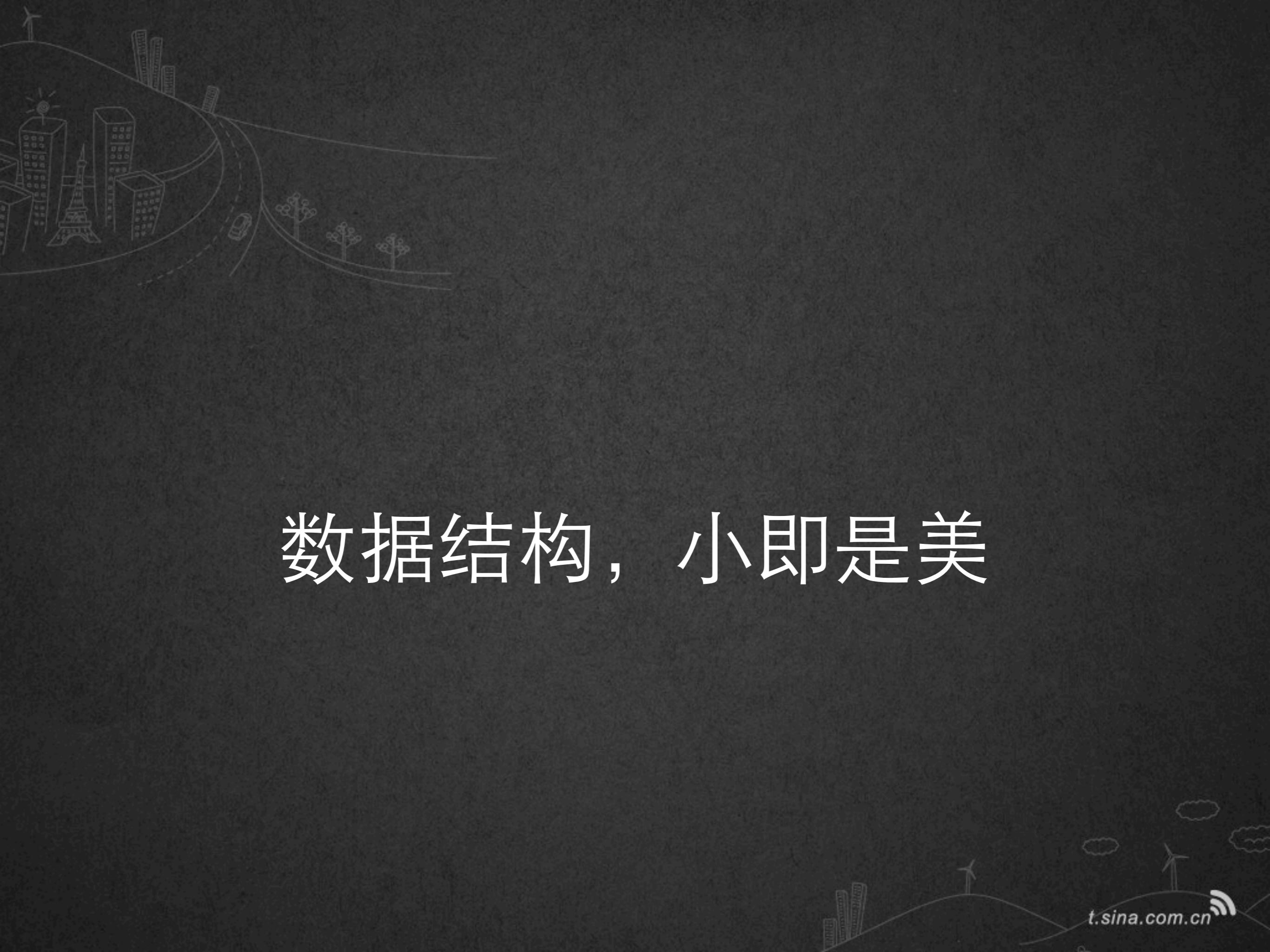


(数据来源：<http://code.google.com/p/thrift-protobuf-compare/wiki/Benchmarking>)



“We would like to provide public APIs that accept protocol buffers as well as XML, both because it is more efficient and because we're just going to convert that XML to protocol buffers on our end anyway.”

- Google



数据结构，小即是美

异步处理需求

- 微博的发表是异步处理
- 随着平台应用增多，与应用相关数据操作也逐渐转向异步方式
- 其他需要异步的操作？

问题二

随着系统增大，“一上线就出问题”情况增多，追究原因很大一部分是由于开发人员对异步处理缺乏深度理解。

- 传统 LAMP 应用中，单个操作步骤增多及处理时间增大对整体性能影响不大。
- 异步队列处理程序中 Ims 的处理时间延长可能会引起连锁反应及业务故障。
- 实时监控队列处理性能指标数据。

异步监控

- MQ stat
- MQ Processor stat

问题三

明星用户上百万粉丝，数据如何实时投递到所有用户？

为什么显示有“1条评论”，点开却没有数据？

实时性与一致性

- Timeline消息流，计数，关系变更等业务需要及时投递给所有用户
- 为了解除耦合，功能由不同服务完成
- 异步处理
 - 将写 cache 与数据库分开
- RAM化

老的发表流程

1. 提交发表
2. 入队列
3. 队列处理程序读出
4. 写cache及数据库master
5. 数据库replication到所有节点

問題



问题

- 第5步，破坏了实时性及一致性

问题

- 第5步，破坏了实时性及一致性
- 改进方法

问题

- 第5步，破坏了实时性及一致性
- 改进方法
 - 进一步异步处理

问题

- 第5步，破坏了实时性及一致性
- 改进方法
 - 进一步异步处理
 - 队列分级

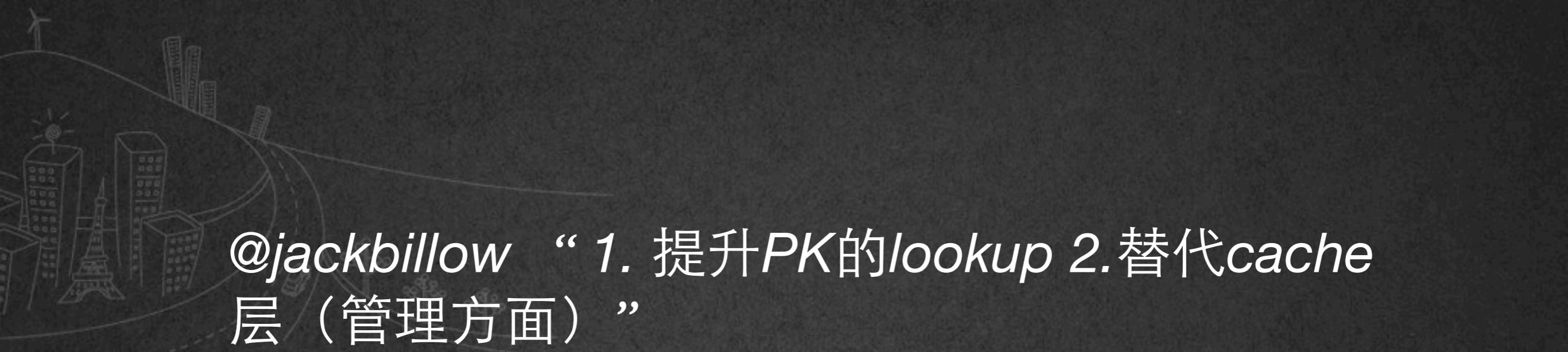
下一步

- RAM is the new disk
- 所有热点数据加载到内存

方法一

“*Percona Server now both SQL and NOSQL HandlerSocket*, 据称它的测试服务器可以跑到 100万 rps, 配置是 12cores/24threads and 380GB of RAM, mysql 如果这样用不就是个 Redis, 所有数据都加到内存, 使用NoSQL方式来访问”

2010-12-25 00:22 转发(9) | 评论(2)



@jackbillow “1. 提升PK的*lookup* 2. 替代cache层（管理方面）”

@kobe “还是运维成本低，挂`innodb`的话原有的工具全能用上”

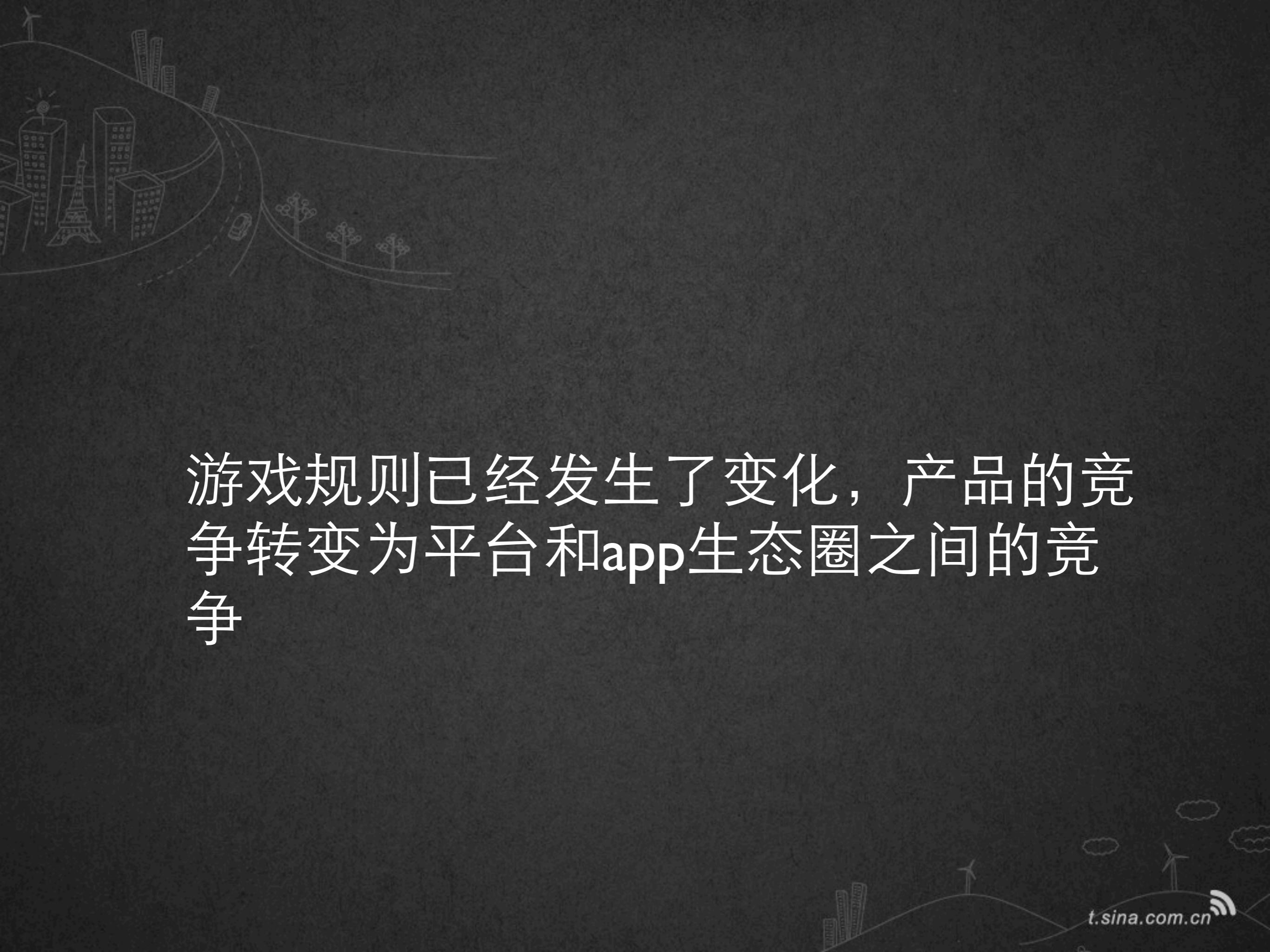
@TimYang “如果一张表可以全部放入内存，`InnoDB`会自动创建一个*Adaptive Hash Indexes*（基于hash的内存索引）以达到内存数据库的性能”

方法二

- 将NoSQL当作数据的读库
 - 通过中间 程序读取binlog放入Redis
 - jbinlog <https://github.com/tangfl/jbinlog>

Part 3

平 台 接 口



游戏规则已经发生了变化，产品的竞争转变为平台和app生态圈之间的竞争

接口设计要素

- 安全
- 隐私保护
- spam
- 易用
- 稳定

全国分布的 接口响应时间及可用性 监控

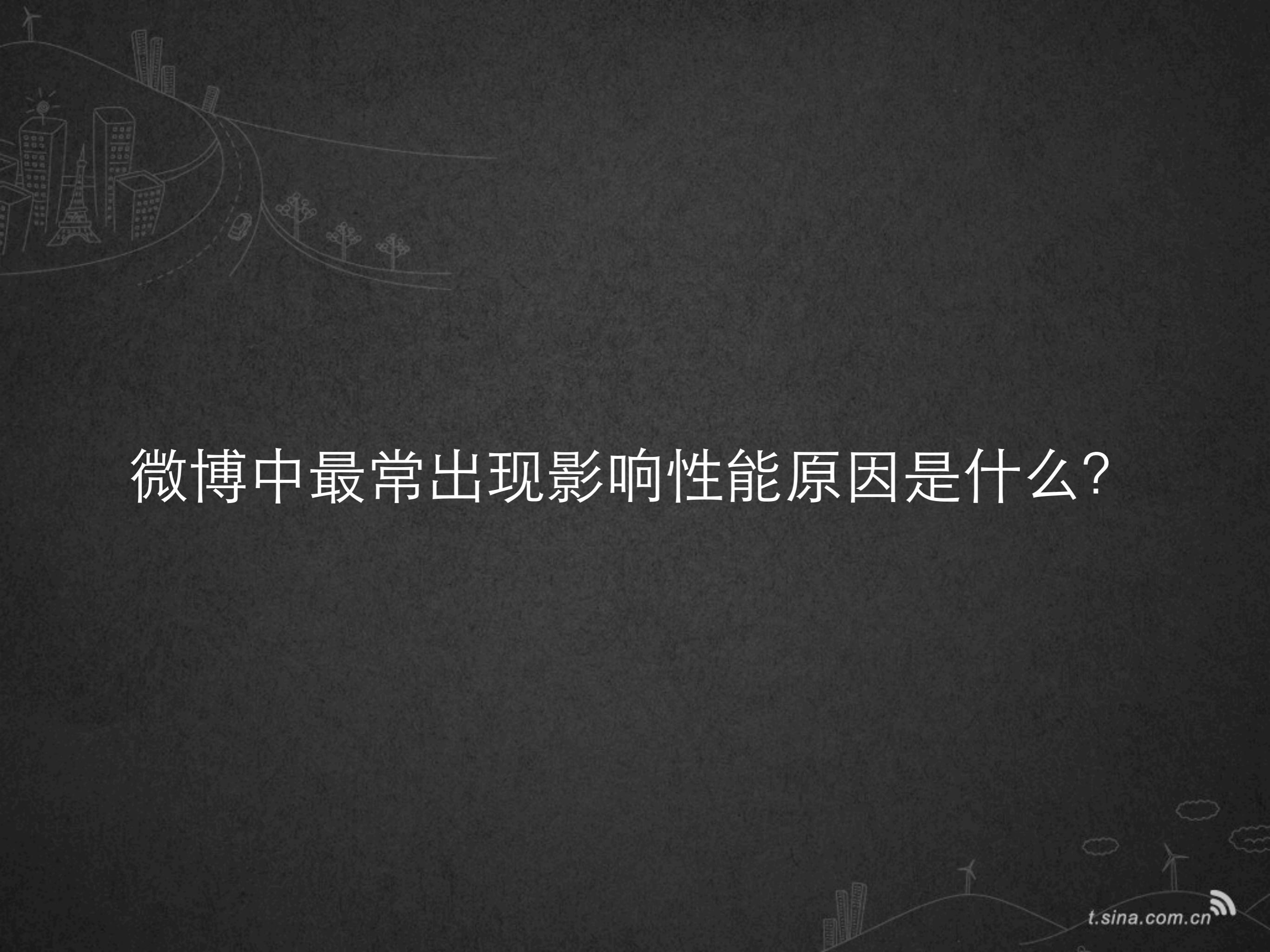
接 口	2011-04-03					2011-04-04					2011-04-05	
	网通	其他	移动	内网	电信	网通	其他	移动	内网	电信		
statuses/public_timeline	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/friends_timeline	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/user_timeline	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/mentions	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/comments_timeline	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/comments_by_me	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/counts	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/unread	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/show	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/update	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/comment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/reply	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
users/show	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
statuses/friends	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
sinafilterresults	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
wordslist	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
yieldfilterresults	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲

其他优化

- 批量访问接口
- GZIP压缩
- 推送接口

Part 4

服务治理



微博中最常出现影响性能原因是什 么？

“@郑环Zheng：“熵增定律：随着系统的持续运行，熵值将趋于无限增长。我们的运维职责就是持续对抗熵增，并尽可能降低熵值。” // @白鸦：架构师的功底就是控制熵。

@bian：晚上学习了"熵" <http://t.cn/h0k4r> 原文转发(49)|原文评论(15)”

2月14日 10:36 转发(22) | 评论(10)

治理一：重构

- 做减法，而不是做加法。
- 新业务也遵循这个原则。

治理二：容量

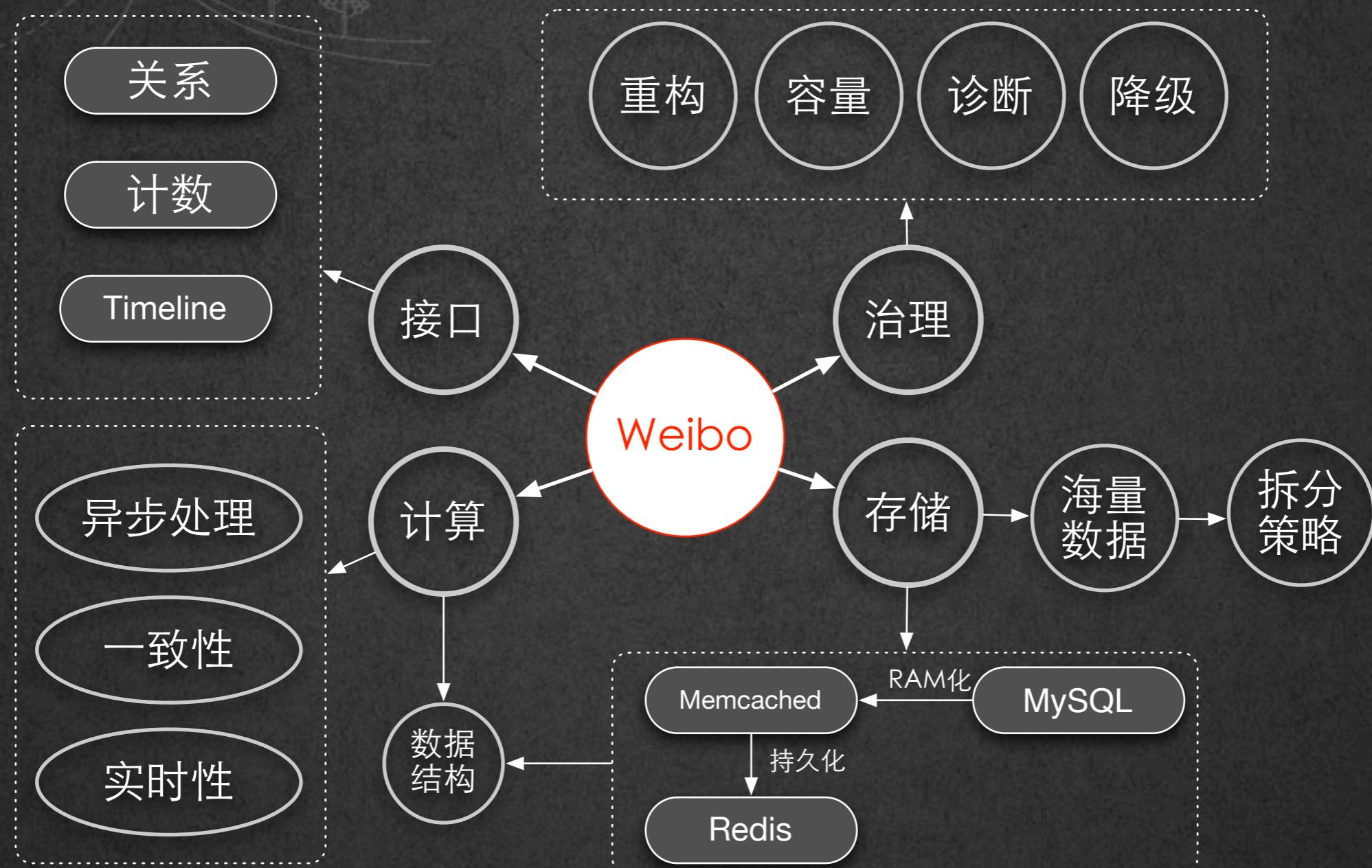
- 容量规划及监控
 - 容量未规划好，未有效监控现有系统
 - 系统突然变慢，不能快速定位
 - 业务迅速增长给容量规划带来挑战

治理三：诊断

- 通过日志和图形工具发现各种异常
 - 接口响应速度
 - 消息处理速度
 - cache命中率
- 迅速定位问题能力

治理四：降级

- 单个业务问题引发了连锁反映，造成整体处理性能下降。
- 降级，每个业务增加独立开关功能



下一步

- 做减法，可以走更远
- 更好的将数据RAM化
- 更好的分布式方式

Q&A

@TimYang

iso1600@Gmail.com

欢迎加入新浪微博技术团队

长期招聘



杭州站 · 2011年10月20日~22日

www.qconhangzhou.com (6月启动)

QCon北京站官方网站和资料下载

www.qconbeijing.com

全球企业开发大会

北京 2011

THE ANNUAL
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE