新浪微博
t.sina.com.cn

微博架构与平台安全

演讲者 @TimYang

China Weibo Developer conference 2010

互联网公司技术架构系列资料

由

为您悉心整理

快简历

KuaiJianLi.com

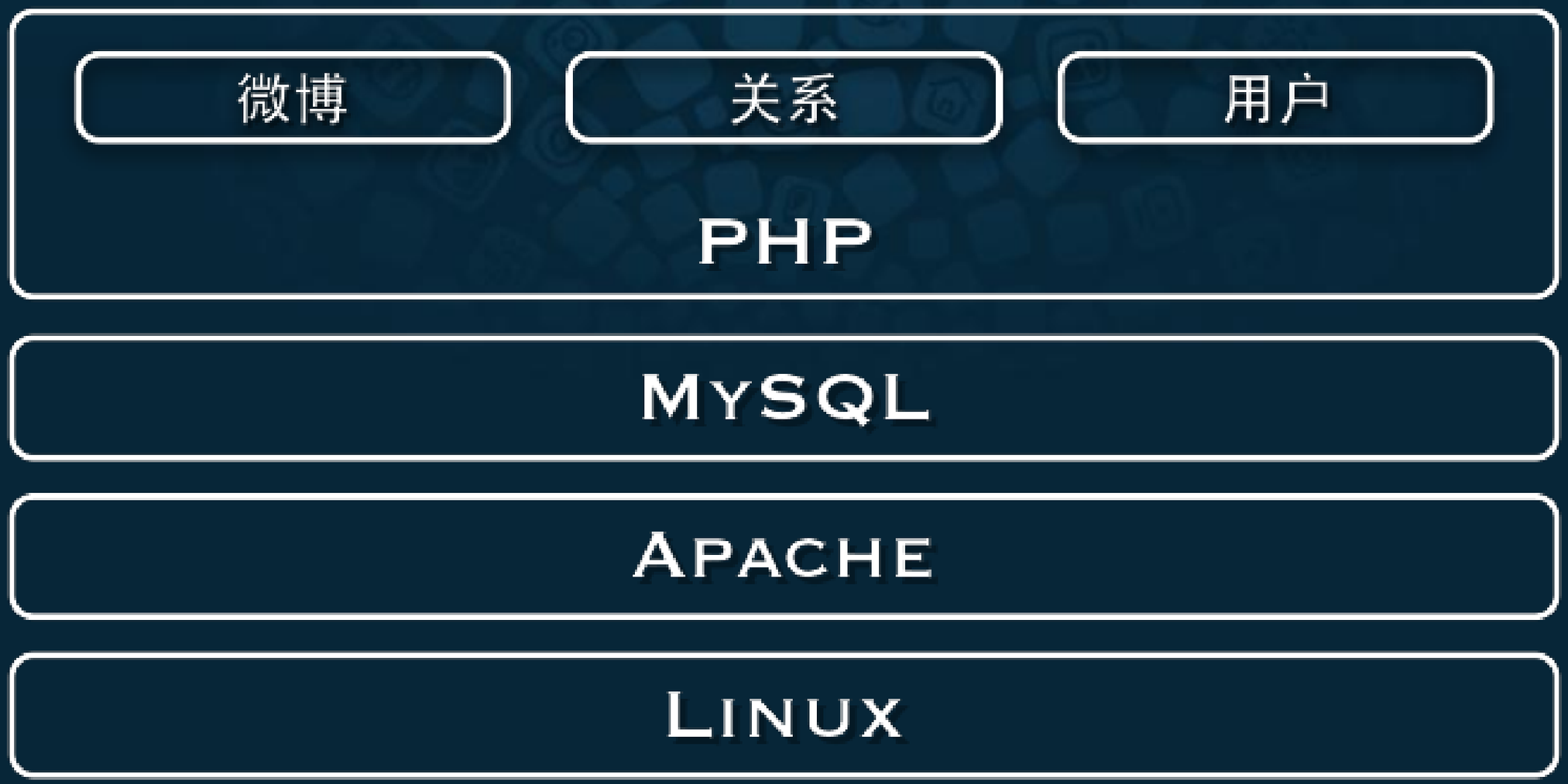
/* 让工作重新关于成长和成就、关于快乐和分享、关于梦想和荣光 */

微博架构发展

China Weibo Developer conference 2010

- 新浪微博从 0 ~ 50,000,000 用户
- 技术架构经历了 3 个阶段

第 1 版



技术特点

- 微博本质是解决发表/订阅问题
- 第 1 版采用推消息模式，将发表/订阅简化成 insert / select 问题

技术细节

- 典型 LAMP 架构
- MySQL：单库单表, MyISAM
  - MPSS (Multi-Port Single Server)


Service1
Service2
Service3
Service1
Service2
Service3
Service1
Service2
Service3
Service1
Service2
Service3

快速成长

- 用户快速增长
- 出现发表延迟现象，尤其是明星用户

架构演变

- 分发推送是造成发表延迟首因
  - 模式改进
- 数据规模增大也带来一定延迟
  - 规模增大：数据拆分
  - 锁表问题：更改引擎
  - 发表过慢：异步方式

第 2 版

投递模式优化

- 推模式改进，不需要推送到所有用户
- 存储及发表峰值压力减轻
- 投递延迟减小

数据拆分

- 优先按时间维度拆分
- 内容和索引分开存放
- 内容使用 key-value 方式存储 (NoSQL)
- 索引由于分页访问，拆分有挑战

异步处理

- 发表异步化
- 发表速度及可靠性得到提高
- 使用 MemcacheQ
  - 增加 stats queue，适合大规模运维

技术细节

- InnoDB 引进，避免锁表烦恼
- PHP 中 libmemcached 代替 memcache
  - 在高并发下稳定性极大提高

高速发展

- 系统问题
  - 单点故障、“雪崩”
  - 访问速度，国内复杂网络环境
- 数据压力及峰值
  - MySQL 复制延迟、慢查询
  - 热门事件微博发表量，明星评论及粉丝

如何改进

- 系统方面
  - 允许任意模块失败
  - 静态内容 CDN 加速
- 数据压力及峰值
  - 将数据、功能、部署尽可能拆分
  - 提前容量规划

平台化需求

- Web 系统
  - 有用户行为才有请求
- API 系统
  - 轮询请求
  - 峰值不明显
  - 用户行为很难预测

- 系统规模持续增大
- 平台化需求

- 新的架构如何设计？

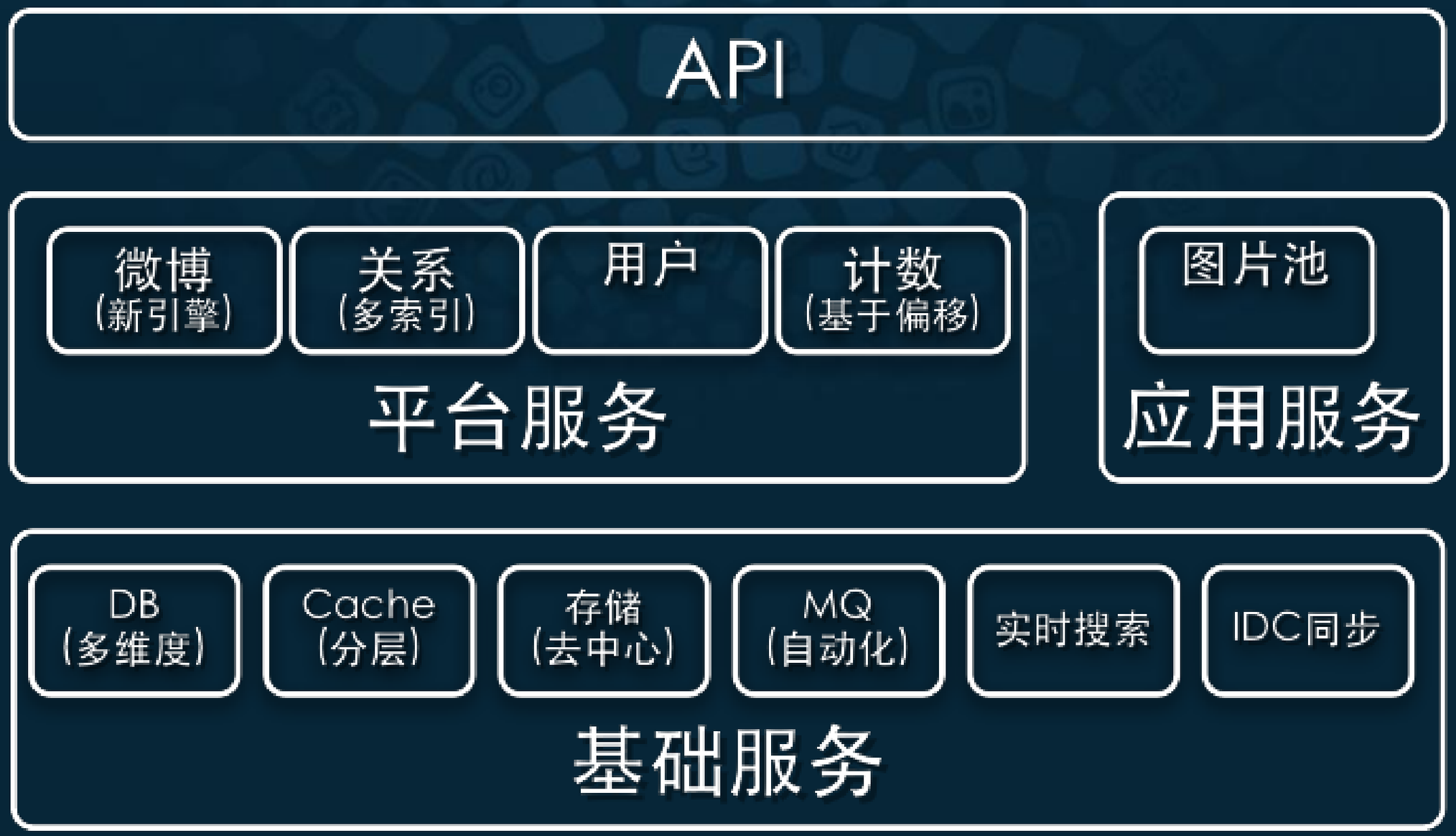
- *“Break large complex systems down into many services... google.com search touches 100s of services (ads, web search, books, news, spelling correction...)”*

- - Jeff Dean, Google Fellow

服务化

- 服务→接口→应用

第 3 版


API
微博
(新引擎)
关系
(多索引)
用户
计数
(基于偏移)
平台服务
图片池
应用服务
DB
(多维度)
Cache
(分层)
存储
(去中心)
MQ
(自动化)
实时搜索
IDC同步
基础服务

平台服务

- 平台服务和应用服务分开，模块隔离
- 新微博引擎，实现 feed cache 分层
- 关系多维度索引结构，性能极大提高
- 计数服务改成基于偏移，更高的一致性、低延迟

基础服务

- DB 冷热分离等多维度拆分
- 图片等存储去中心化
- 动态内容支持多 IDC 同时更新

高性能架构

- 50,000,000 用户使用新浪微博
- 最高发表 3,000 条微博 / 秒
- 姚晨发表一条微博，会被 3,689,713 粉丝读到（11 月 10 日 数据）

问题本质

- 解决高访问量、海量数据规模下
- 易于扩展、低延迟
- 高可用

异地分布能力

- 每天数十亿次Web及接口请求
- 请求内容随时变化，结果无法cache

- 如何扩展？

思路

- 去状态，可请求服务单元中任意节点
- 去中心化，避免单点及瓶颈
- 可线性扩展，如
  - 100 万用户，10 台服务器
  - 1000 万用户，100 台服务器
- 减少模块耦合

实时性

高性能系统具备低延迟、高实时性

实时性核心是让数据离 CPU 最近，避免磁盘 IO

“CPU 访问 L1 就像从书桌拿一本书，L2 是从书架拿一本书，L3 是从客厅桌子上拿一本书，访问主存就像骑车去社区图书馆拿一本书。”

- 余锋 @ ecug 2010

淘宝网核心系统专家，Erlang技术专家

微博 cache 设计


Inbox
(home)
L0 cache
Outbox
(Myfeed)
L1 cache
(最近发表)
L2 cache
(中期)
Content
L0 cache
(热门内容)
L1 cache
(最近发表)
L2 cache
(中期)

高可用

- 好的架构具有高可用性
- 业界
  - Amazon S3: 99.9%
  - Amazon EC2: 99.95%
  - Facebook: n/a
- 微博平台 ~ 99.95% (5 小时 / 年)

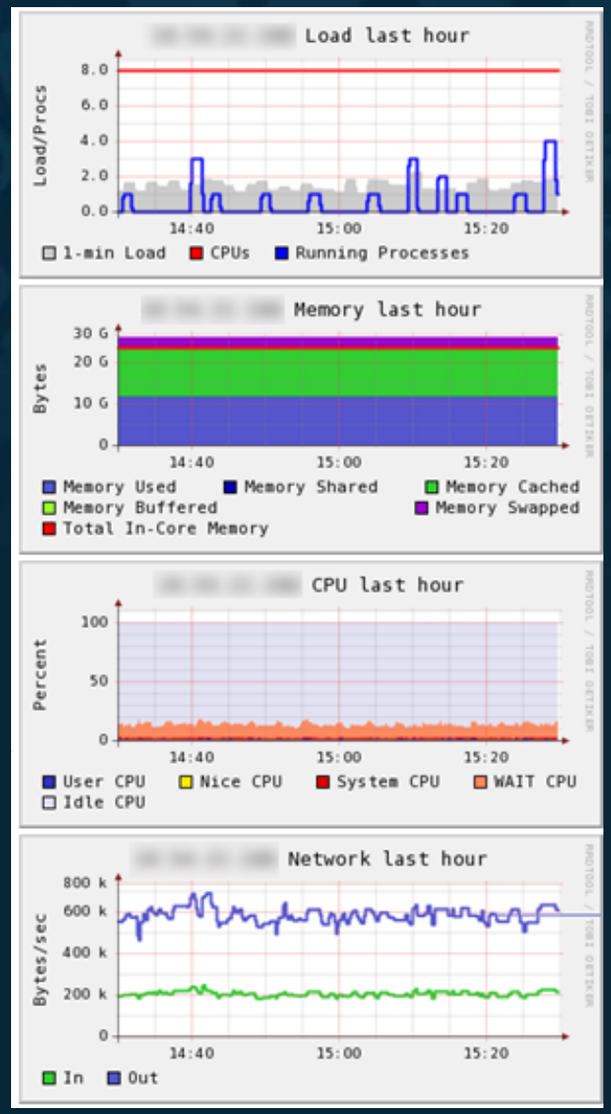
如何达到

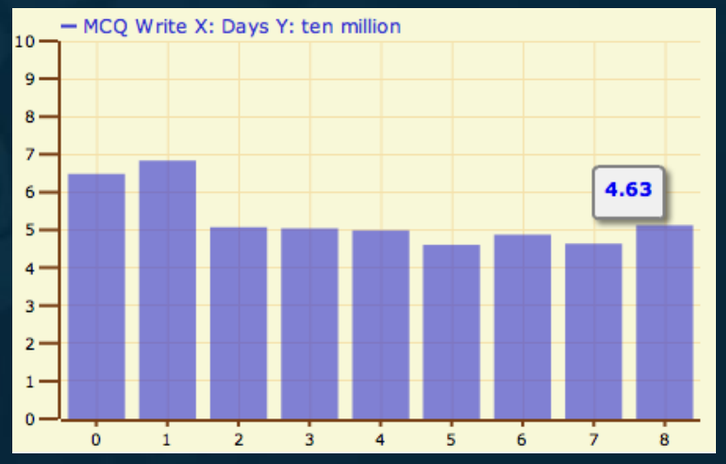
- 容量规划
  - 图表
- 监控及 admission control...
  - 接口及资源监控, 7x24
  - 业务回环测试, 监测业务逻辑有效性
  - 集成测试

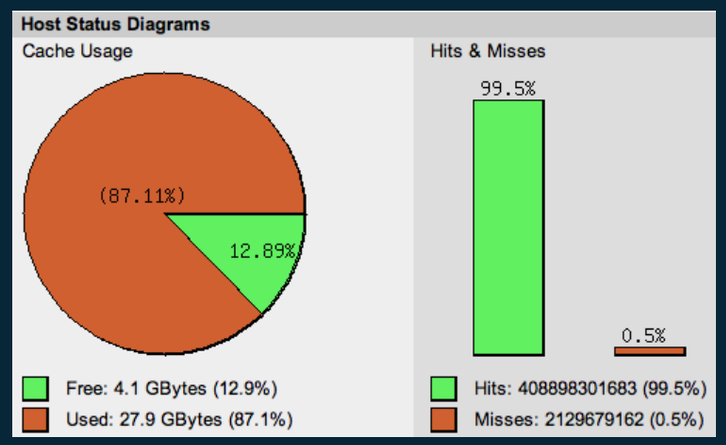
图表

通过图表

了解系统容量







接口监控

- curl / 各地请求情况及响应时间
- 流量异常 / access log
- non-200 结果 / 失败率 / exceptions
- 将监控指标量化
  - 类似 mysql seconds behind master

- *“Many services are written to alert operations on failure and to depend upon human intervention for recovery, about 20% of the time they will make mistakes.*
- *Designing for automation.”*
  - - James Hamilton, VP of Amazon

自动化

- 大规模互联网系统运作需要尽可能自动化
- 发布及安装
- 服务启用、停止
- 故障处理
- 前提，去状态化，允许单点故障及重启

- *“System administration at Google usually have 1 week of "on call" duty, and the other 5 weeks are spent making improvements to make the on call portion more optimized, automated, and trouble-free”*

- - Tom Limoncelli @ Everything Sysadmin
- Lumeta Corporation总监，贝尔实验室专家

微博系统运转依赖大量自动化工具

工具在持续改进并增加中……

```
Master_Auto_Failover(...)
    If MasterActive() = Failed
        Get_All_Slave_Servers()
    For (;;)
        Change_Master_To()
        Modify_DNS_Resolve()

Rebuild_Slave(...)
    If This_Slave_Backup_exist = True
        rsync data from backup
    Start_Slave();
    If Slave_OK = True
        add_DNS_Resolve();
```

- 高可用性还有异地分布的需求
- 在国内网络环境下，IDC 灾难、机房检修维护会导致服务中断
- 用户就近访问可提高速度

- 静态内容分布采用 CDN 技术，成熟
- 动态内容分布是业界难点
- 核心是数据的分布式存储

- 理想的分布式存储产品
  - 支持海量规模、可扩展、高性能、低延迟、高可用性
  - 多机房分布，异地容灾
  - 调用简单，具备丰富数据库特性

分布式存储

需要解决多对多的数据复制同步

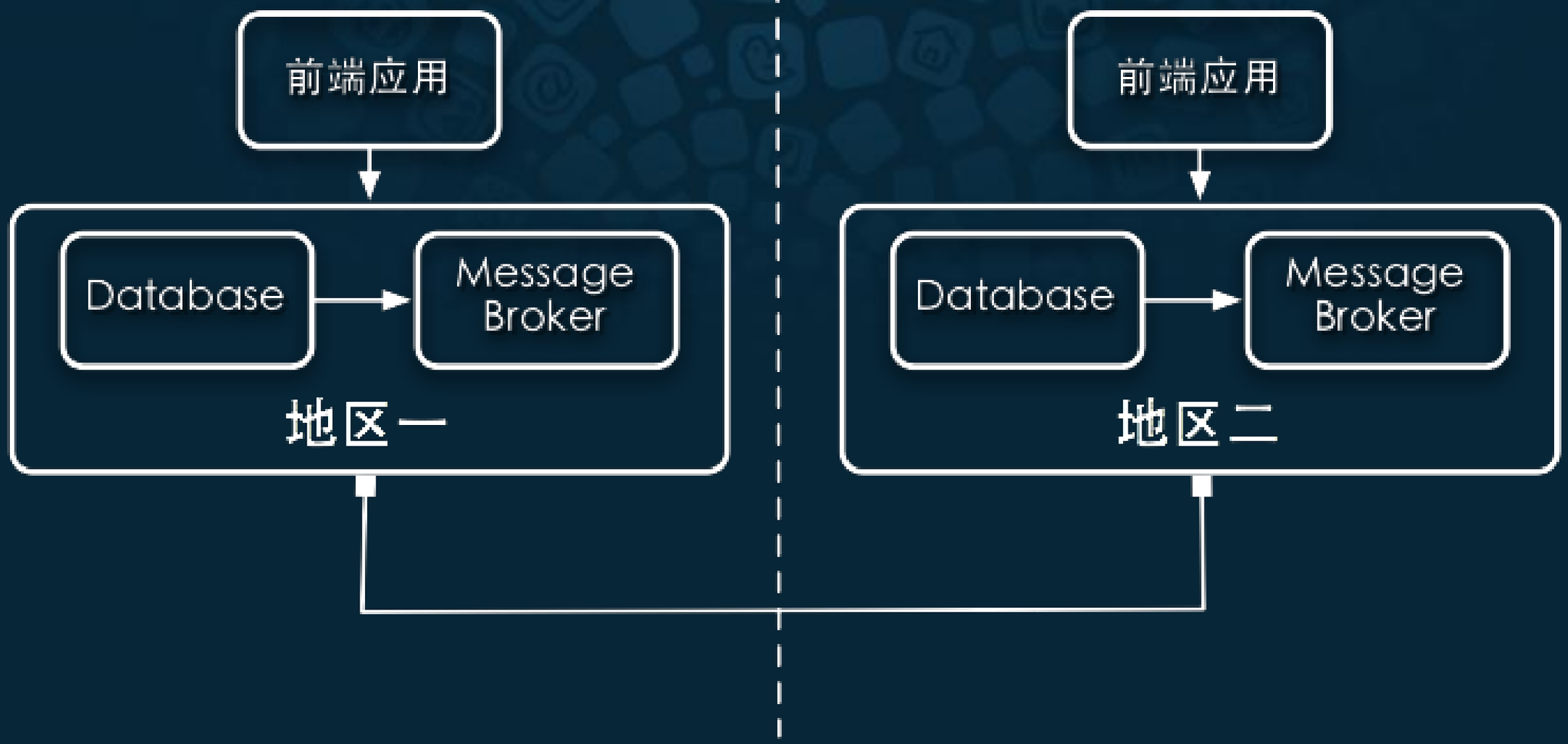
及数据一致性

复制策略

- Master / Slave
  - 实现简单，master 有单点风险
- Multi-Master
  - 合并多处写，异步，最终一致性
  - 需要应用避免冲突
- Paxos：强一致性，延迟大

- Multi-Master
- Web 应用多地区同步的最佳策略
- 没有现成成熟的产品

微博方案


前端应用
前端应用
Database
Message Broker
Database
Message Broker
地区一
地区二

- 通过消息广播方式将数据多地分布
- 类似 Yahoo! Message Broker

- *“We use YMB for replication for 2 reasons.*
- *1. YMB ensure msgs are not lost before they are applied to the db.*
- *2. YMB is designed for wide-area replication. This isolates individual PNUTS clusters from dealing with update between regions”*
  - *PNUTS: Yahoo!’s Hosted Data Serving Platform*

新推送架构

China Weibo Developer conference 2010

现状

- API 大部分请求都是为了获取最新数据
- 重新思考 Rest API
  - 大部分调用都是空返回
  - 大部分时间在处理不必要的询问
  - 无法实时投递
  - 存在请求数限制（rate limit）

如何解决

- 新一代推送接口(Stream API)
- 采用推送的方式
  - 有新数据服务器立即推送给调用方
  - 无数据则不消耗流量
  - 客户端实现更简单

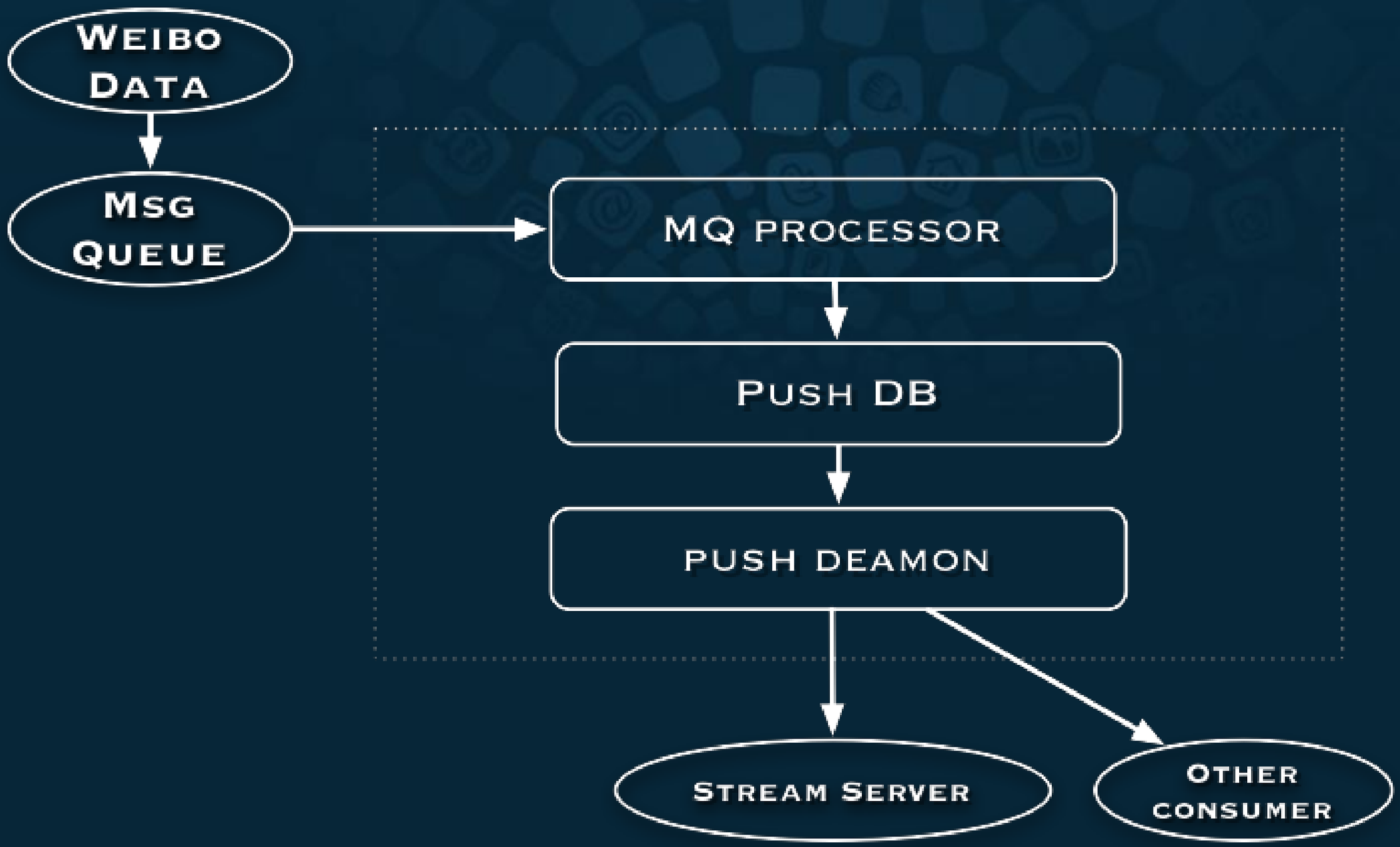
```
$ curl -u 10503:xxx http://api.t.sina.com.cn/\
  stream?uid=10503&since_id=22110111266526577 2

{"created_at": "Fri Nov 12 18:29:53 +0800 2010",
"id": 22110111266526577 2,
"text": "test msg",
"source": "null",
"favorited": false,
"truncated": false,
"in_reply_to_status_id": "",
"in_reply_to_user_id": "",
"in_reply_to_screen_name": "",
"geo": null,
"user": {"id": 1432494550,"screen_name": "Thomas"}
},
{"created_at": "Fri Nov 12 18:29:55 +0800 2010",
"id": 20110111116501651 00,
"text": "...",
...
```

技术特点

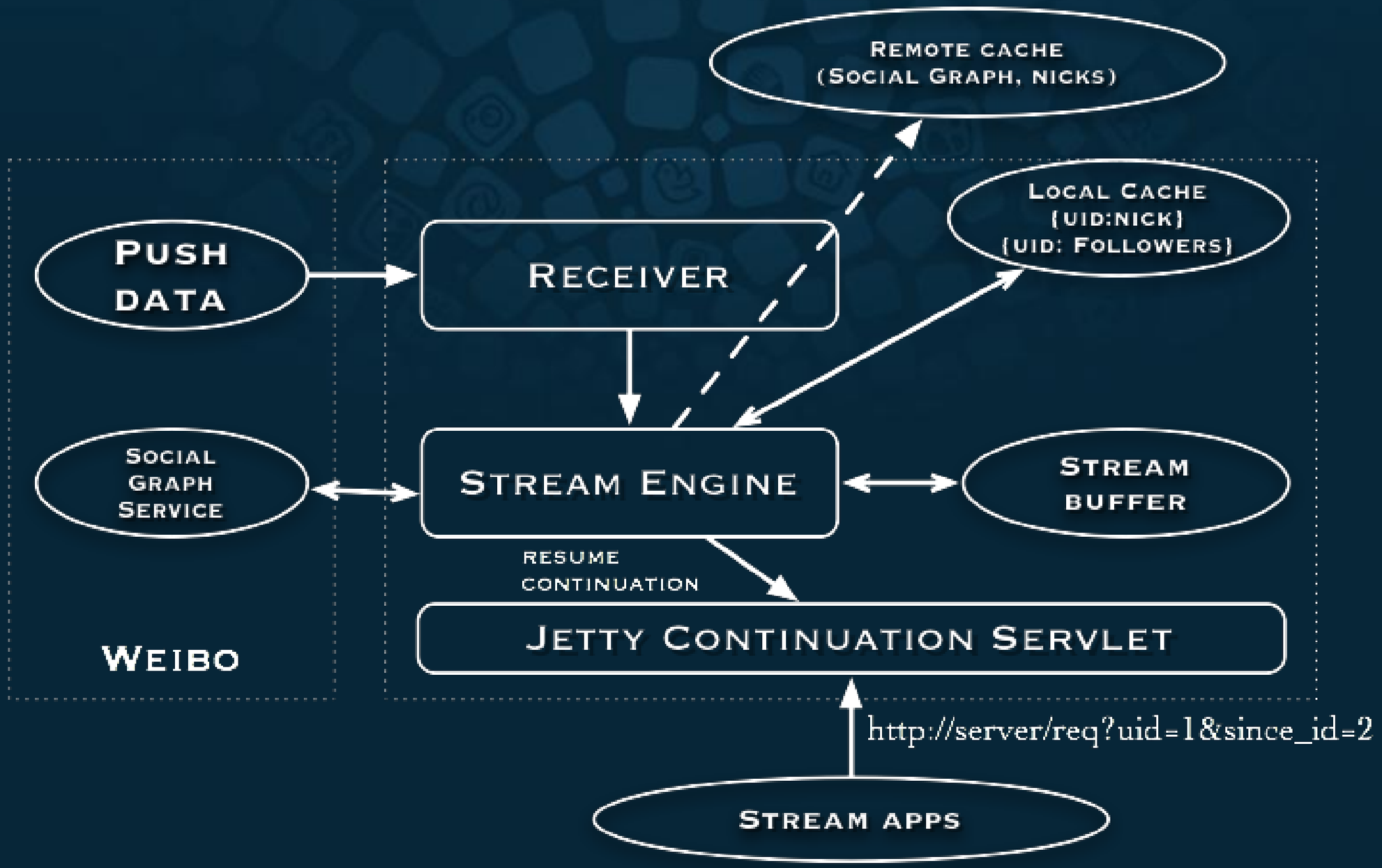
- 低延迟，从发表到客户端接收1秒内完成
- 高并发长连接服务

推送架构


Weibo Data
Msg Queue
MQ processor
Push DB
Push deamon
Stream Server
Other consumer

- 为什么先持久化
- KISS，Keep It Simple and Stupid
- 测试表明持久几乎不增加延迟开销
  - batch insert
  - cursor read

内部细节


Remote cache
(Social Graph, nicks)
Local Cache
{uid:nick}
{uid: Followers}
Push
data
Receiver
Social
Graph
Service
Stream Engine
Stream
buffer
resume
continuation
Jetty Continuation Servlet
Weibo
http://server/req?uid=1&since_id=2
Stream apps

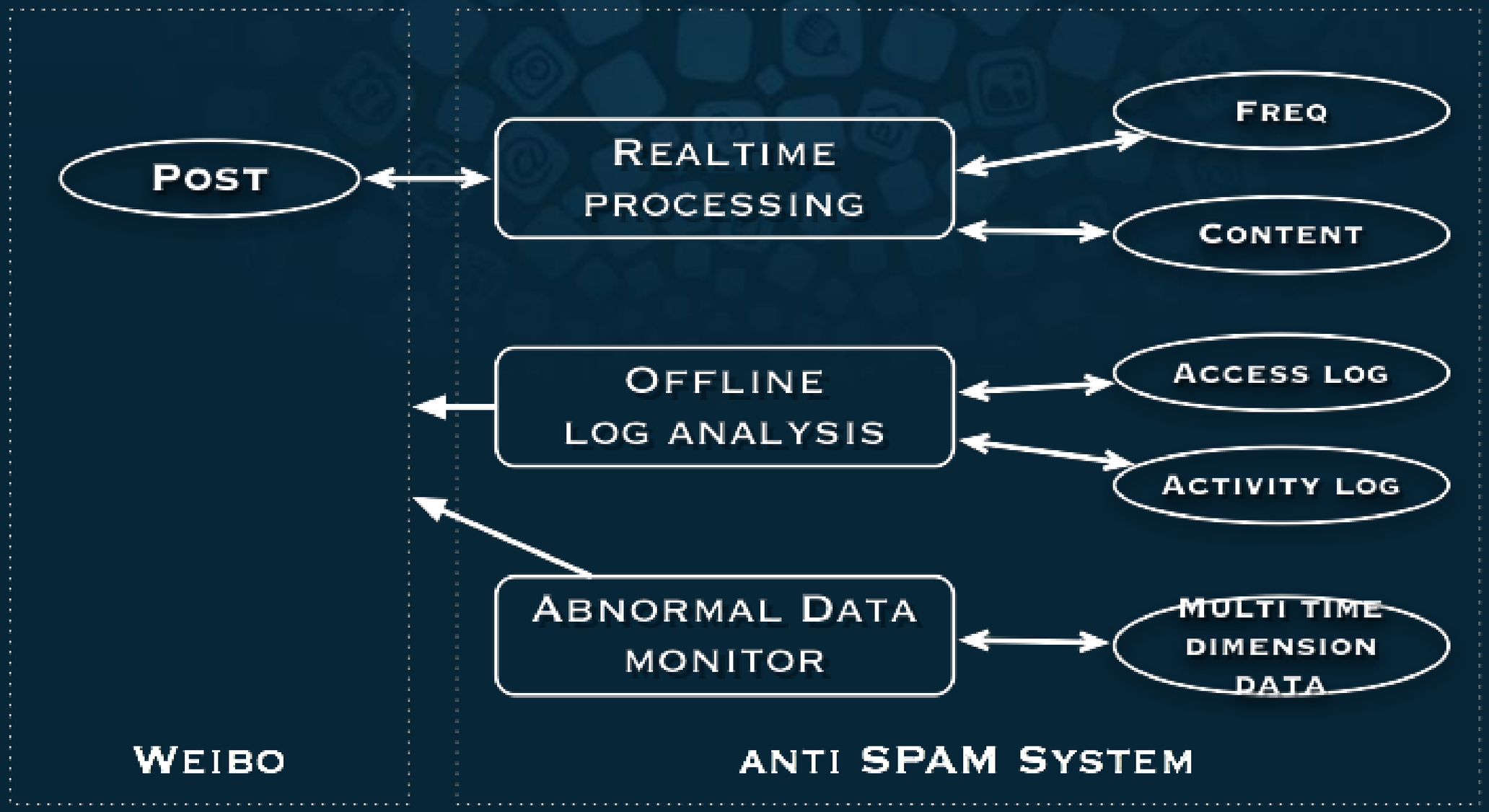
- Stream Buffer
- 保存用户最近数据
- 保存客户端断线重连之间下行数据

平台安全

China Weibo Developer conference 2010

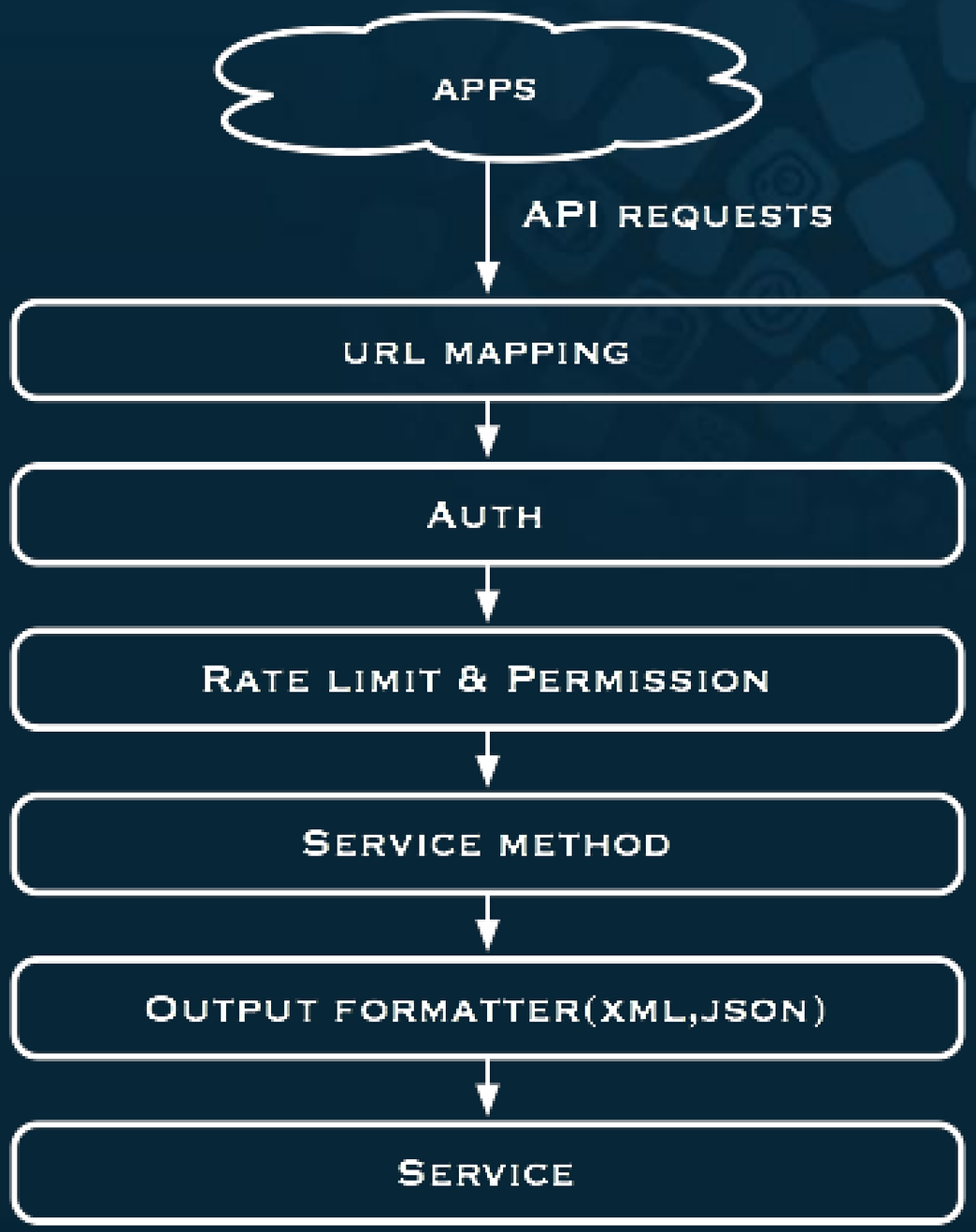
- 由于接口开放，需要防范各种恶意行为
- 垃圾内容
- 垃圾粉丝
- 恶意行为

内容安全


POST
REALTIME PROCESSING
FREQ
CONTENT
OFFLINE LOG ANALYSIS
ACCESS LOG
ACTIVITY LOG
ABNORMAL DATA MONITOR
MULTI TIME DIMENSION DATA
WEIBO
ANTI SPAM SYSTEM

- 微博平台需要
- 为用户提供安全及良好体验的应用
- 为开发者营造公平的环境
- 接口需要清晰的权限控制及安全规则

接口安全



- Auth层
  - 访问需要 AppKey
    - 需要 OAuth 授权
- 权限层
  - 流量控制、权限

- 架构就是将复杂问题抽象简单并解决

- 下一代微博架构，期待您的参与
- Join us! @TimYang

THANKS