

构建可扩展微博架构

Tim Yang
新浪微博
技术架构师

从博客到微博

博客

- **功能**
 - 发表
 - 浏览
 - 留言
- **Content Manager System**

博客

- 技术，LAMP
 - MySQL master/slave
 - Memcached
 - PHP
 - CDN

- 微博，产品
- Real-time
- 关注关系
- 信息聚合

新周刊
NEW WEEKLY

国产微博盛宴史
99条最精彩围脖
“30亿村官”背后的夺权记
中国的三个新加坡
武广高铁的富贵病？
看得见的乌托邦

公共产品定价权属于国民
315
2010.1.15
社论

微革命

从推特到新浪微博



02 > www.newweek.com.cn 010-65001000
ISSN 1673-0690

信息聚合

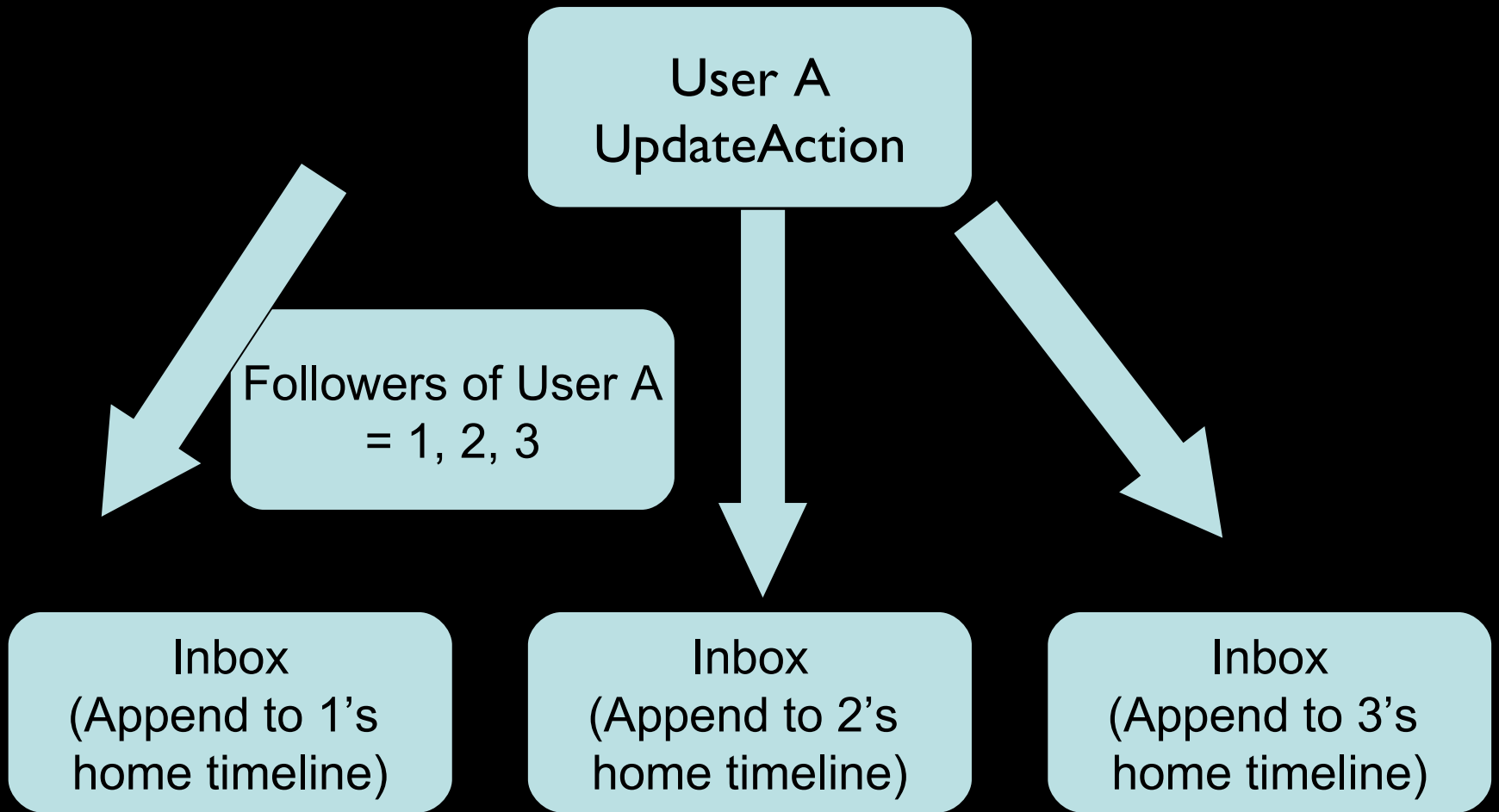
信息聚合

- 微博两种信息聚合设计模式
 - Push(推)
 - Pull(拉)

Push

- 把微博看做邮件
 - Inbox: 收到的微博
 - Outbox: 已发表微博
- 发表: 存到所有粉丝
inbox(重)
- 查看: 直接访问 **Inbox(轻)**

Push(Figure)



Push

- 优点：实现简单，首选
- 缺点：分发量



姚晨 **v**

<http://t.sina.com.cn/yaochen>

 北京,朝阳区

博客：<http://blog.sina.com.cn/yaochen>

一颗很逊的卤蛋。

+ 加关注

推荐给朋友

v 新浪认证

407
关注

1464587
粉丝

1443
微博



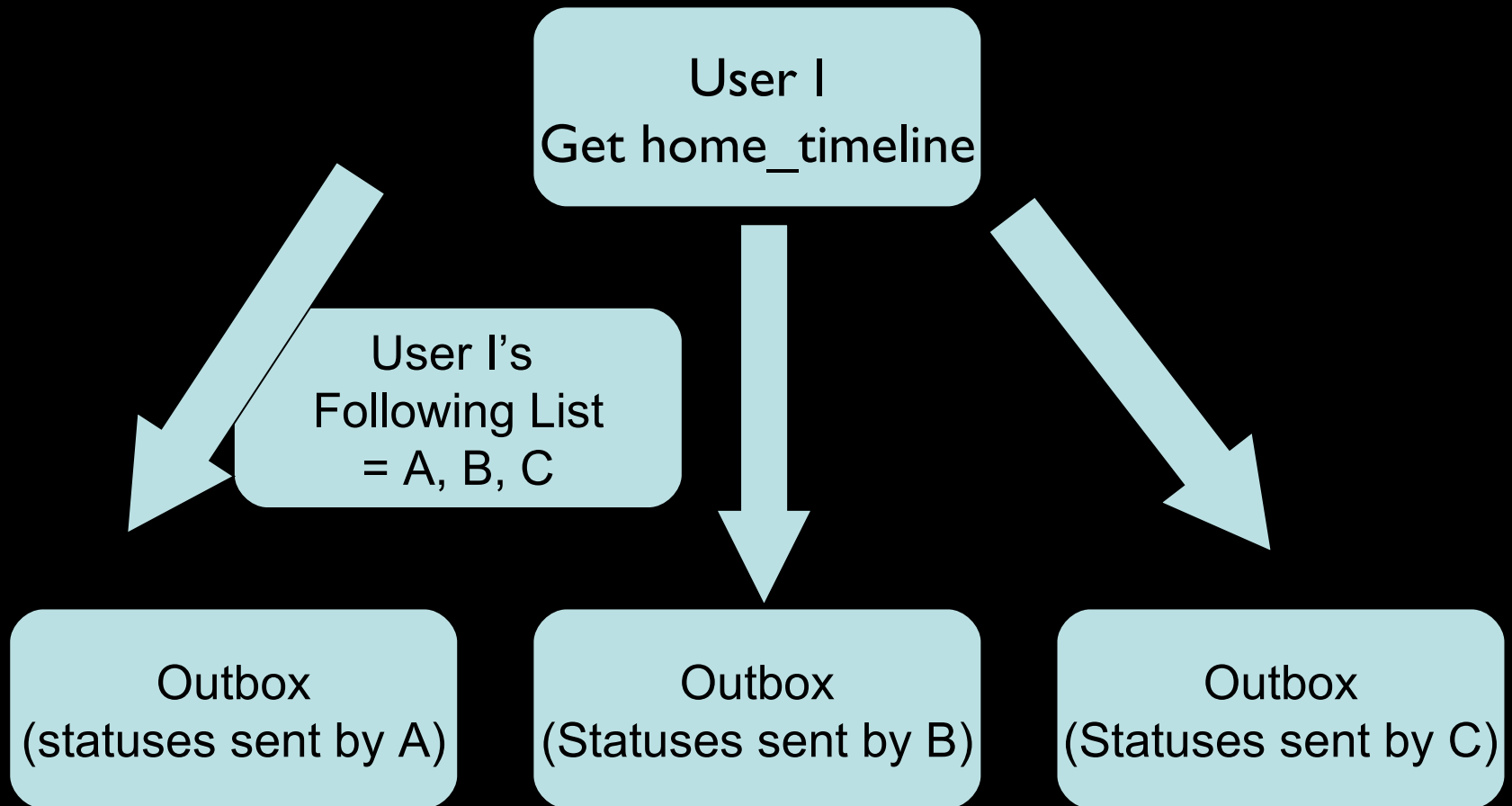
个人资料

还没有填写，请稍候。

Pull

- 发表：存到自己 **outbox(轻)**
- 查看：所有关注对象
Inbox(重)

Pull



Pull

- 优点：节约存储
- 缺点：计算量大

- 微博是一个消息分发系统
- 可采取推或拉的方式实现

架构挑战：峰值

- 如除夕、春节

请求量

- 如果发表量 5,000 万 / 天
- 平均： 578 条 / 秒
- 设计系统容量： 2,000?

IO 瓶颈

- 峰值: 5,000 – 10,000?
- 100,000?

后果

- **Latency**
- **DB read timeout**
- **前端 timeout (503 error)**
- **解决方案?**

异步设计

- 不同步等待
- 将消息存入消息队列
(Message Queue)
- 轻量级的发表

MQ products

- Kestrel by twitter
- RabbitMQ, an Erlang Queue Server
- **Memcached**
 - 在新浪微博项目大规模使用

Memcached

- 基于 Berkeley db, 稳定可靠
- Memcached protocol, 丰富的 client library
- 容易监控 (stats queue)
- 只有 2 个命令: get/set

避免单点故障

核心服务，需避免单独故障
方法

1. 使用多个 Memcacheq 池
2. Get 操作：轮询所有服务器
3. Set 操作：随机选择一个

无需其他复杂“架构”设计

MQ 方式通用的优点

- **Offline work**
- **应用请求量不均衡**
- **解耦**
- **异步通讯**
- **原则**

使用 MQ 原则

计算开销大于消息分发开
销

架构挑战：实时性

越重要的事件，越希望实时性

第十九届世界杯 6月11日-7月12日

2010 South Africa
FIFA World Cup
南非世界杯



足球



黄健翔



董路



李承鹏



贺炜



詹俊



朱骏



谢晖



孙继海

*The value of the tweet
decreases
exponentially with
time*

John Kalucki, Twitter

解決思路

Cache 中心化

*Ram is the new the
disk*

- **Local Cache**
- **Memcached**
- **Database buffer/cache**

- **LAMP 中， cache= 可选层**
- **Cache 中心化后新的问题**

容量问题

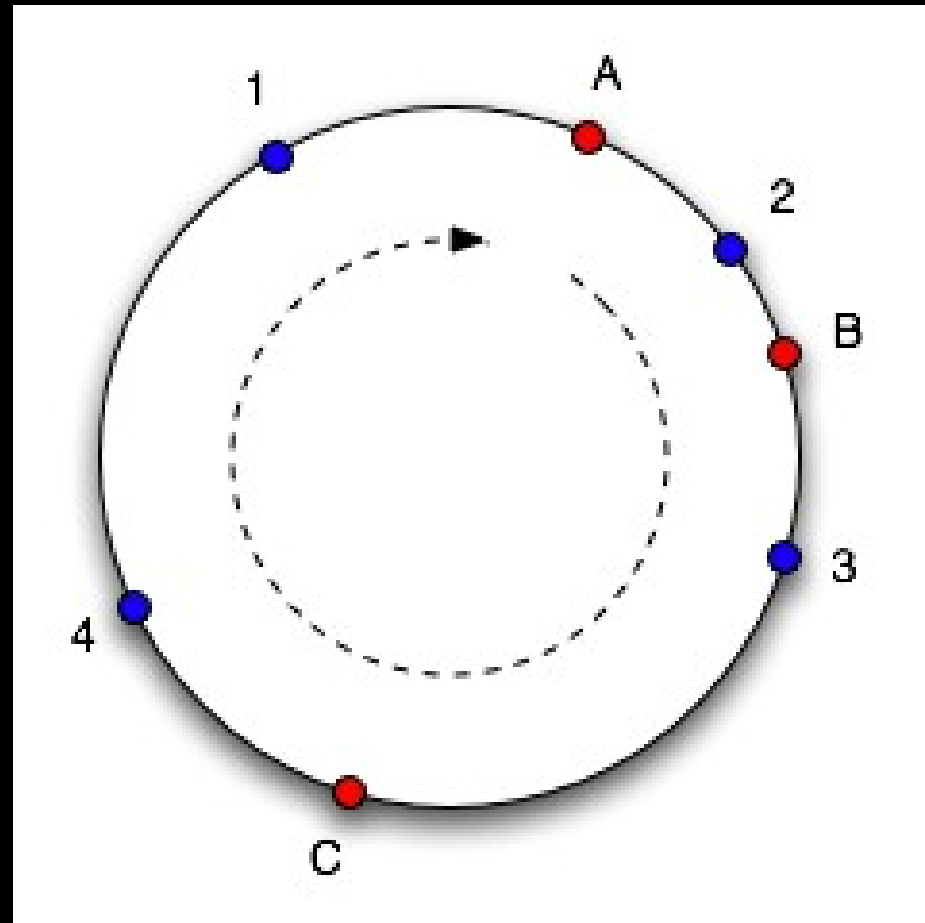
- TB 级
- 思路：压缩
 - QuickLZ
 - LZ0
 - 不用 gzip

单点问题

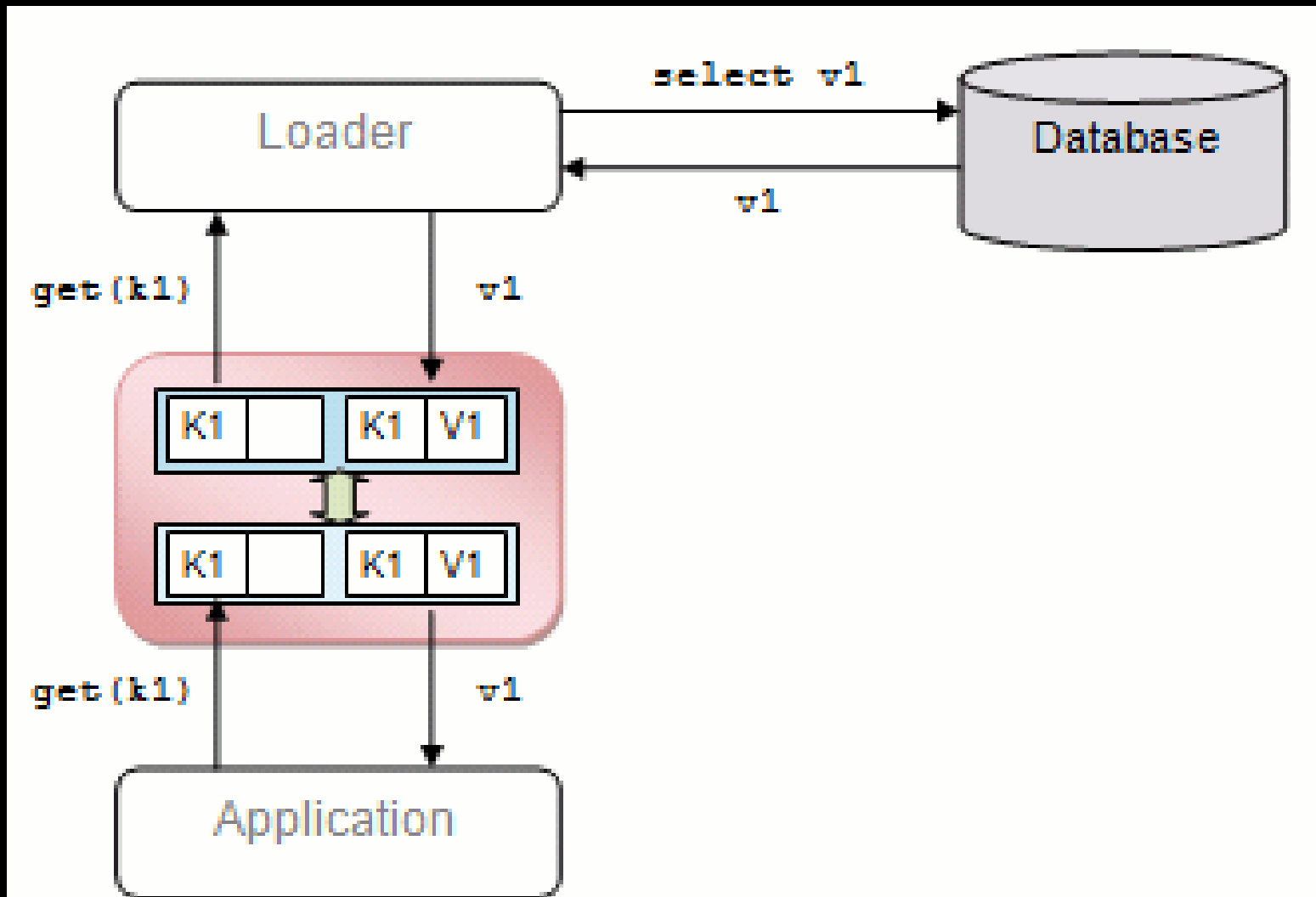
- 单点故障，**SIGSEGV**
- 如何应对
 - 1. Consistent hash
 - 2. Read-through cache

Consistent hash

- 原理
- 优点
 - 震荡最小



Read-through cache



Read-through and Write-through

- **Products or projects**
 - MySQL memcached UDF
 - Cache money for Ruby on Rails
- **Or wrap a proxy for the db driver, in any language**

Evictions 问题

- **Evictions: cache 数据被踢**
- **性能的噩梦**
- **Latency 产生的源头之一**

如何避免 **evictions**

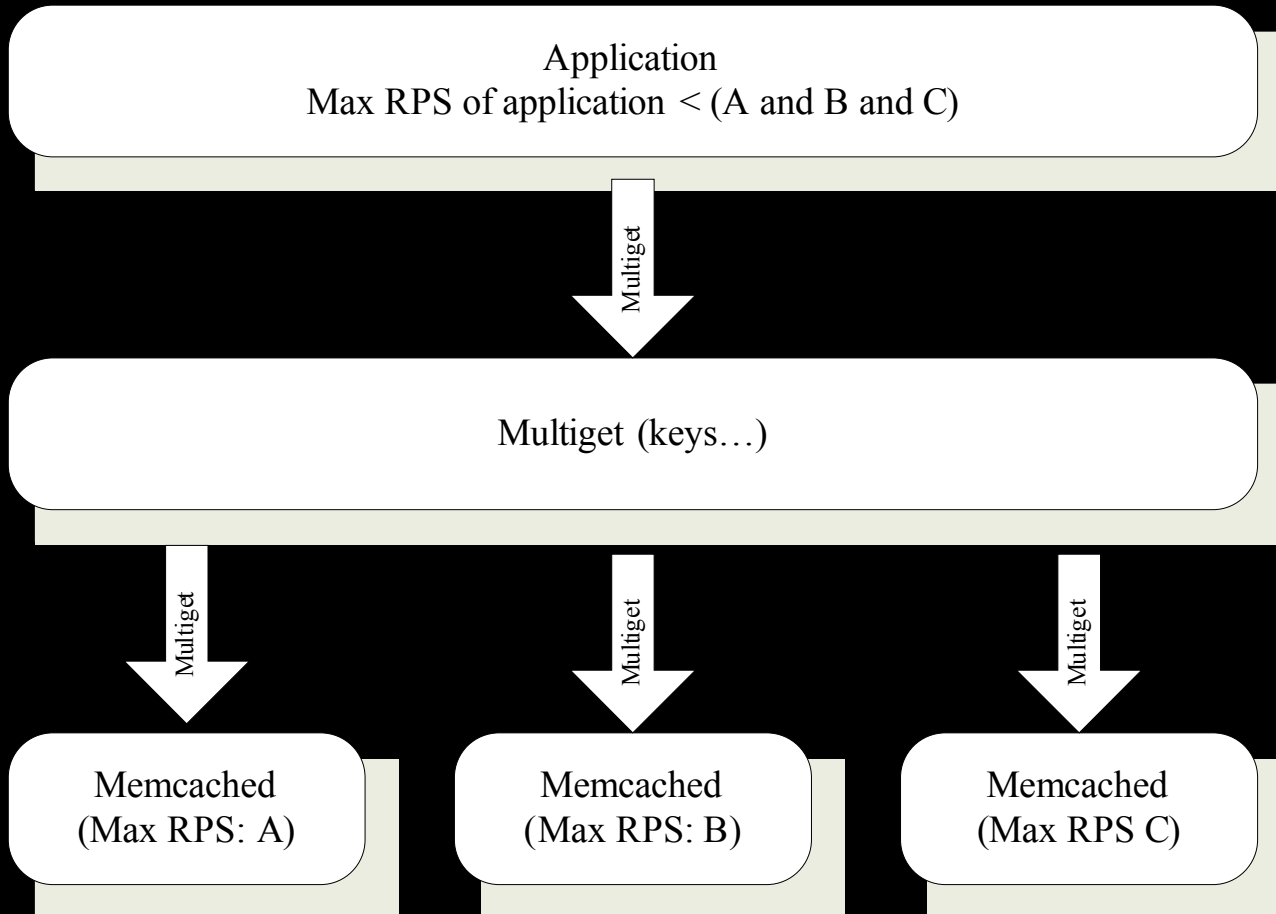
- 规划 **cache** 容量
- 将永久数据与临时数据分开
- 不使用随机字符作为 **key**

Multiget 问题

When memcached servers are CPU bound, adding more memcached servers doesn't help serve more requests.

- Jeff Rothschild, Vice President of Technology at Facebook

Cache 挑战: multiget hole



解决方法

- **Memcached replication**

架构挑战：海量存 储

架构挑战： 国内网络带宽问题

地理分布

- 考慮到以下原因，需要分布式部署
 - 访问速度
 - IDC 不可用
 - 故障
- 分布的核心是数据分布

数据地理分布原理

- **Master-slave**
- **Master-master**
- **2PC/3PC**
- **Paxos**
- **<http://timyang.net/data/mu/>**

地理分布的方案

- **MySQL master/slave**
- **Dynamo/Cassandra**
- **PNUTS**

架构挑战：API 访问 量

以新浪微博开放平台为例

- **REST API**

- 编程简单，library 丰富

- 可用 curl, javascript 实现一个 client

- 缺点单向询问方式

- **如何解决轮询压力**

解决方案： Sina App Engine

- Sina App Engine 应用云平台提供微博 API 底层支持
- 并可以 host 微博 app

Sina App Engine

<http://sae.sina.com.cn> ALPHA2

- 微博，Web 2.0 最核心技术之一
- 还有更多的架构挑战等待解决
- 欢迎加入新浪微博技术团队

Q & A

新浪微博: @ TimYang

Twitter: @ xmpp

Email: iso1600 @ gmail.com