构建可扩展微博架构

Tim Yang
新浪微博
技术架构师

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 */

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

信息聚合

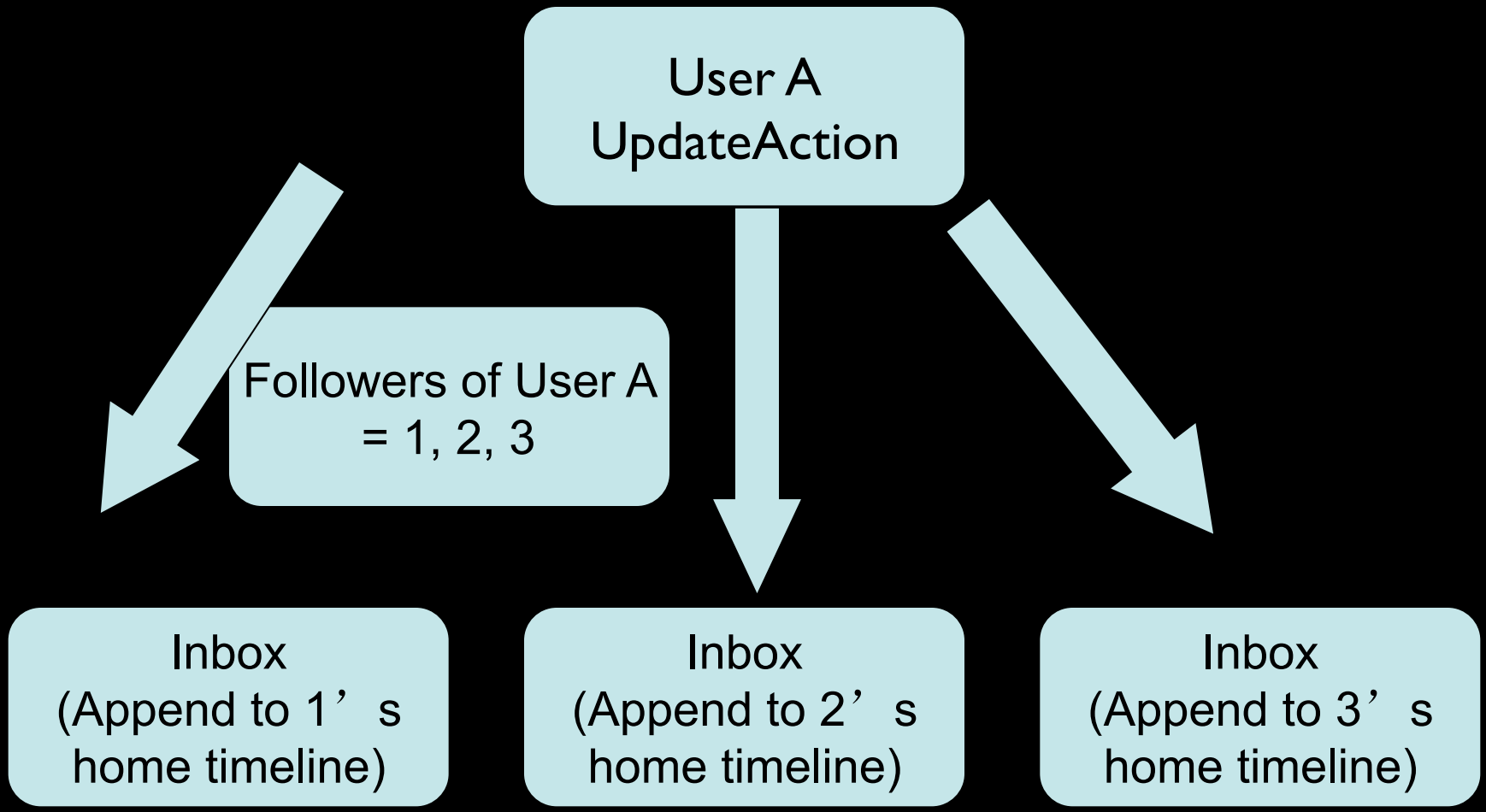
信息聚合

- 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)


User A
UpdateAction
Followers of User A
= 1, 2, 3
Inbox
(Append to 1’s
home timeline)
Inbox
(Append to 2’s
home timeline)
Inbox
(Append to 3’s
home timeline)

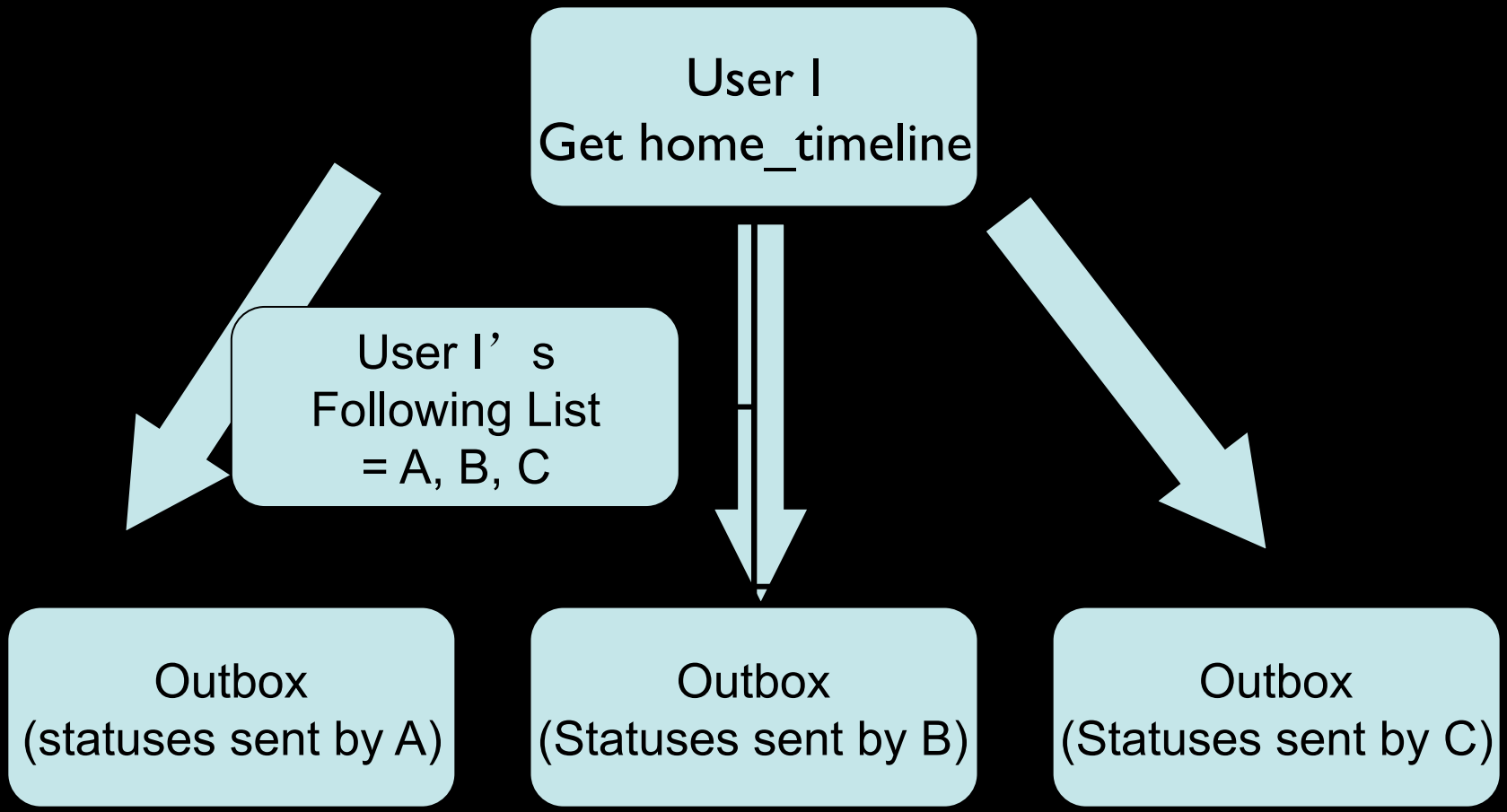
Push

- 优点：实现简单，首选
- 缺点：分发量

Pull

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

Pull


User I
Get home_timeline
User I’s
Following List
= A, B, C
Outbox
(statuses sent by A)
Outbox
(Statuses sent by B)
Outbox
(Statuses sent by C)

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
- **Memcacheq**
  - 在新浪微博项目大规模使用

Memcacheq

- 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

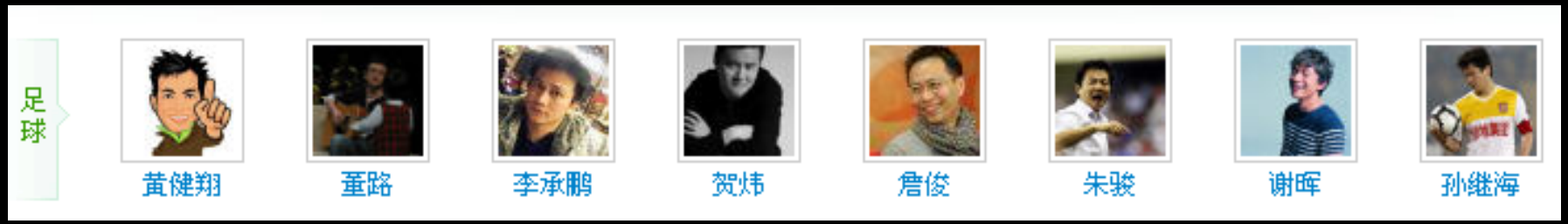
- **Offline work**
- 应用请求量不均衡
- 解耦
- 异步通讯
- 原则

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



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

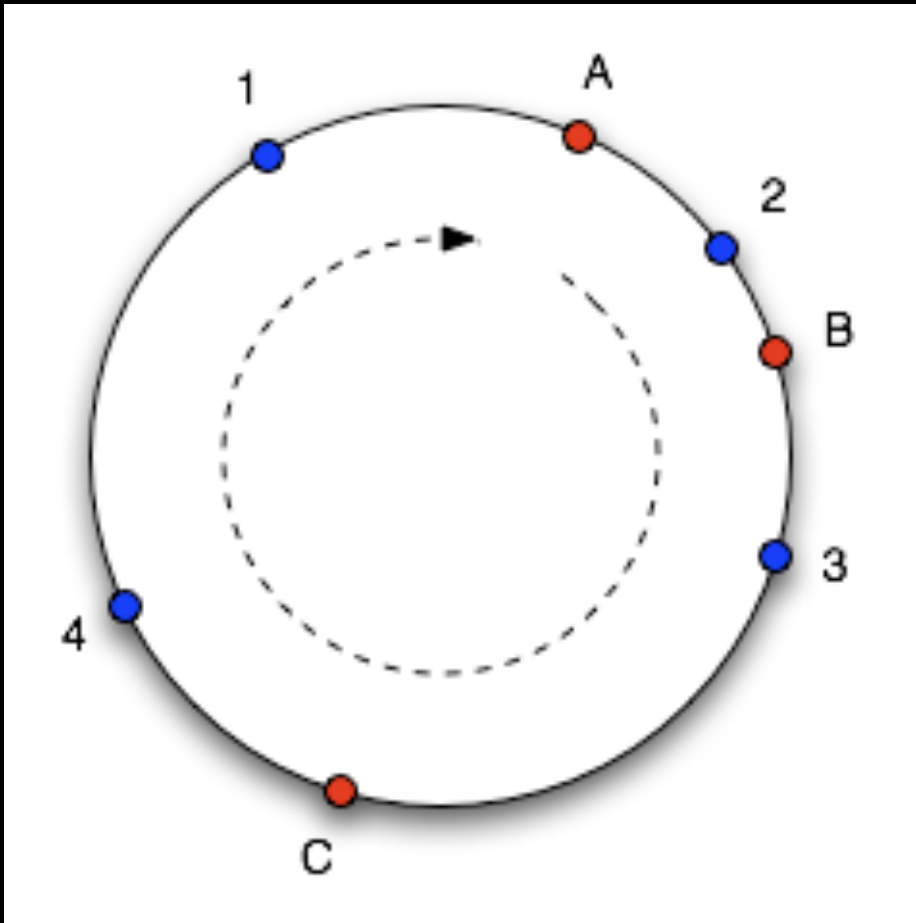
- **TB级**
- 思路：压缩
  - –QuickLZ
  - –LZO
  - –不用gzip

单点问题

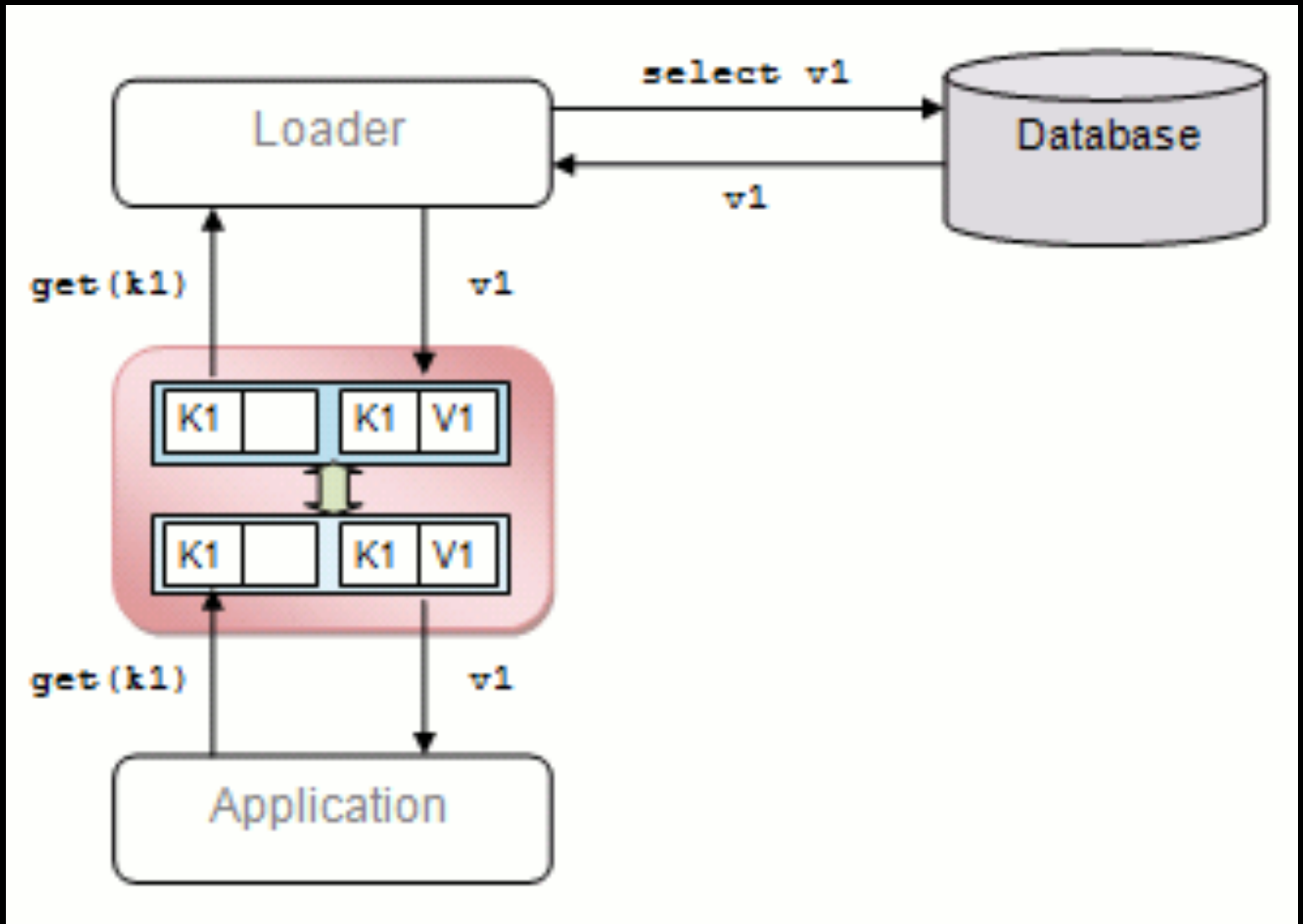
- 单点故障, **SIGSEGV**
- 如何应对
  - 1. Consistent hash
  - 2. Read-through cache

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

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

如何避免evictions

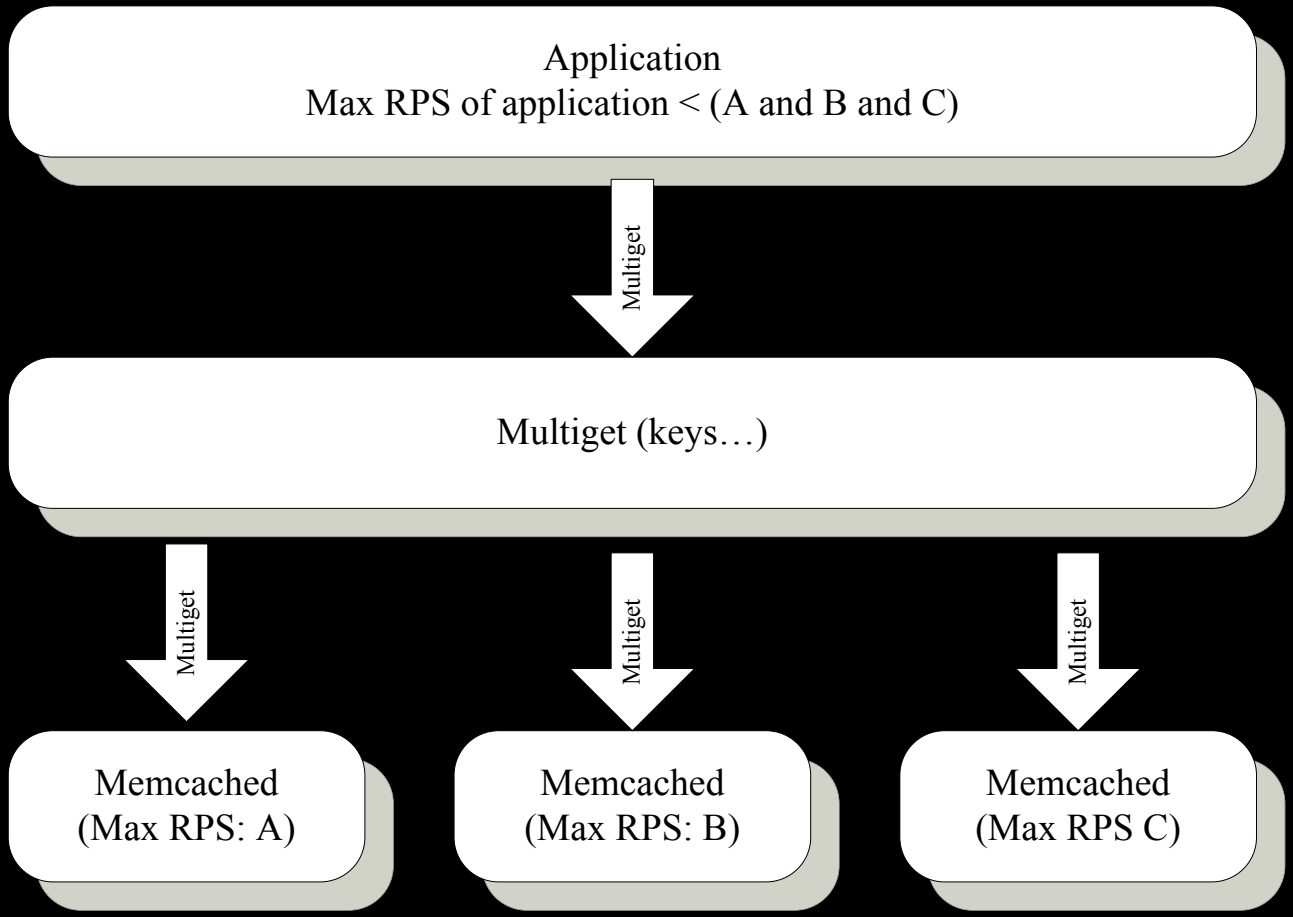
- 规划cache容量
- 将永久数据与临时数据分开
- 不使用随机字符作为key

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



解決方法

- **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

- 考虑到以下原因，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/multi-idc-design/**

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

- 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