SACC 2015中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2015
互联网+ 重塑IT架构

58同城移动im架构优化实践

58沈剑
shenjian@58.com

关于-我

- 百度-高级工程师
- 58同城-高级架构师
- 58同城-技术委员会主席
- 58同城-技术学院优秀讲师
- 58到家-技术总监
- 58到家-技术委员会负责人
- 本质：程序员！

微博

微信

目录

零、移动im的业务价值

移动im的业务价值

- 移动APP时代，**即时沟通是基本需求**
- 2014年，FB花190亿美金收购WhatsApp
- 微信的本质是什么？
- im对腾讯意味着什么？
- im对淘宝意味着什么？
- im对百度意味着什么？
- im对滴滴意味着什么？
- 如果刀塔传奇加上im功能呢？





一切APP皆需im！

一、移动im难点

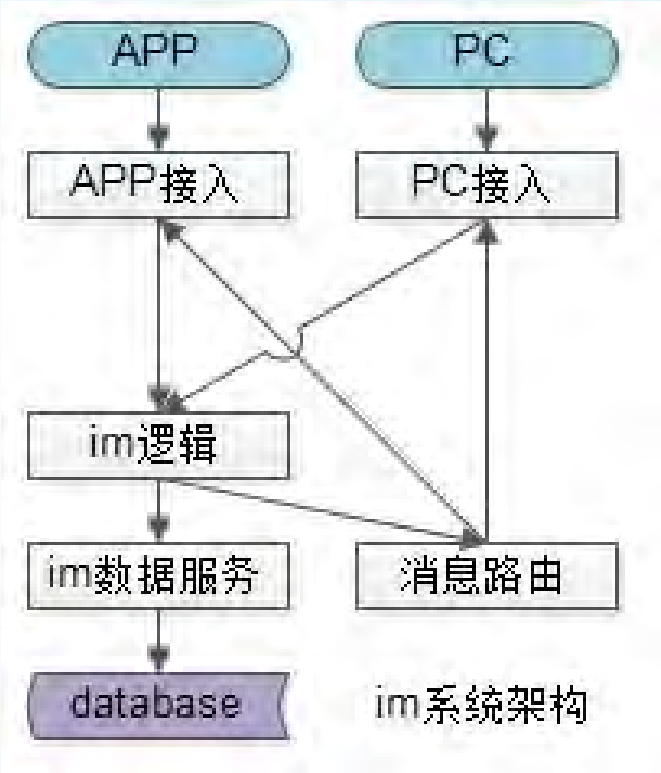
移动im难点

- 基于推送的系统 => TCP消息通道
- 连接稳定性 => 进电梯，出电梯，断线？
- 流量敏感性 => PC上你关注一个page是200k，还是300k么？
- 消息可达性 => 连接不稳，消息总丢？
- …

二、移动im架构简述

移动im架构简述

- APP接入层
- 逻辑处理层
- 消息路由层
- 数据存储层



与传统系统架构不一样的地方？

三、连接稳定性优化

稳定性-短连接优化

- 无线环境下，网络稳定性较差
- 所有的请求都通过TCP长连接走，经常断线
- **Request-Ack式的请求，可以优化为短连接**

（1）拉取离线消息

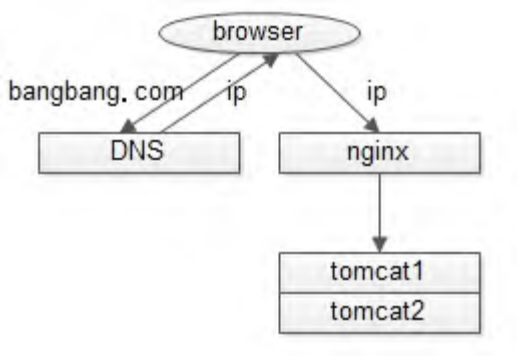
（2）拉取好友列表

（3）拉取好友信息

（4）...

稳定性-DNS优化（一）

- HTTP短连接，第一步是DNS解析
- 无线环境下，DNS解析的时间（以及nginx转发的时间）是不能忽略的
- APP再帅气，DNS失败，一切都白搭！
- PC时代的玩法



稳定性-DNS优化（二）

- **APP时代我们的玩法：直接使用ip连接**

（1）第一次需要拉取ip-list

（2）后续直接使用ip

（3）用时间戳机制来更新ip-list

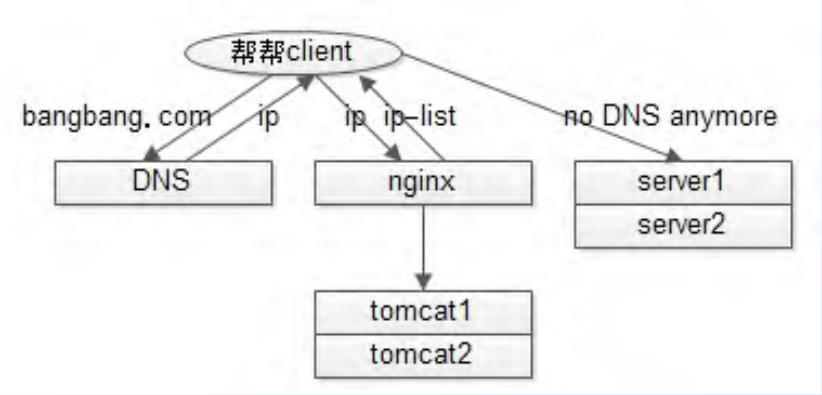
- 优点

（1）不再需要DNS解析

（2）不再需要nginx转发

（3）扩展性好？

（4）支持异构服务器负载均衡？



稳定性-session保持（一）

- 传统基于TCP的im系统的传统做法

（1）TCP与session有强绑定关系

（2）TCP断开，则清除session，走登出流程，向好友反向通知登出

（3）TCP重连，新建session，走登录流程，向好友反向通知登录

- 新建session要走密钥协商，建立安全通讯信道，成本高
- 无线时代网络不稳，TCP时断时连

稳定性-session保持（二）

- 优化：解除TCP与session的强绑定关系，TCP断开，session不清除
- 在线状态不变更，不反向通知状态变化
- 存在的问题？

（1）tcp断开，session保持的过程中，万一有消息发过来呢？

（2）session建立在接入服务器A，重连万一连到接入服务器B呢？

稳定性-快速重连

- 优化：快速重连，快速回复session

TCP接入是有状态的！

（1）优先接连上次连接的接入服务器

（2）不走登录过程，不验证用户名密码，直接验证加密密钥

（3）多拉取一次离线消息

- 存在的问题？

（1）如何验证加密密钥

（2）快速重连失败怎么办？

四、流量优化

ID信息和详细信息分开拉取

- 客户端连接上服务器后，为了界面展示，干了什么？
- 需要同步哪些数据？

（1）分组数据【id + 分组详情】

（2）好友数据【id + 好友详情】

（3）群组数据【id + 群组详情】

（4）...

- 优化：**id和info分开拉取**

table:user_friend	
id	friend_id
1	2
1	3
1	4

table:user_info			
id	nick_name	gender	birthday
1	酷	男	1979
2	帅	男	NULL
3	靓	男	NULL
4	拽	男	NULL

延迟拉取、按需拉取

- 什么是延迟拉取 => 按需拉取？
- 优点，缺点？
- 哪些数据可以延迟拉取？

（1）个人详细信息

（2）好友详细信息

（3）群组详细信息

（4）...

- 和业务相关，例如分组信息就不能延时拉取

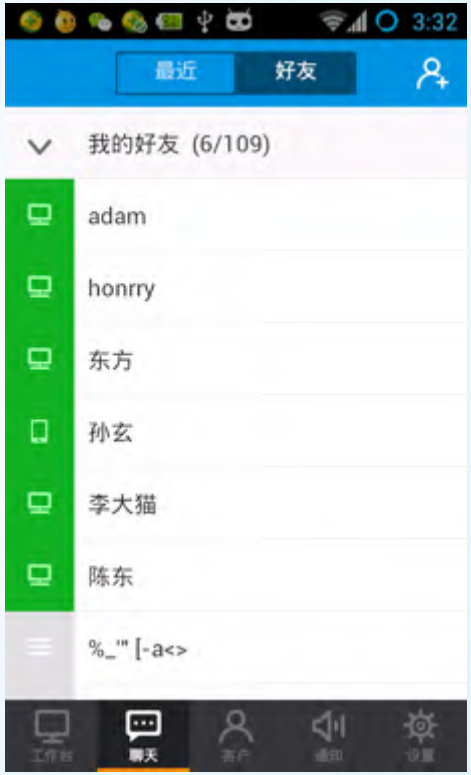


table:user_friend

id	friend_id
1	2
1	3
1	4

table:user_info

id	nick_name	gender	birthday
1	酷	男	1979
2	帅	男	NULL
3	靓	男	NULL
4	拽	男	NULL

本地数据+时间戳优化（一）

- 能否利用客户端本地数据，来优化需要同步的数据量？
- 时间戳优化
- 列表数据时间戳：**时间戳表**

=> 列表id数据发生增减，更新时间戳表

- 详细数据时间戳：**时间戳列**

=> 详情数据发生改变，更新时间戳列

table:list_timestamp

id	friend_list	group_list
1	10	100

table:user_info

id	nick_name	gender	birthday	timestamp
2	酷	男	1979	10
3	帅	男	NULL	100
4	靓	男	NULL	1000
5	拽	男	NULL	58

table:group_info

id	group_name	intro	timestamp
100	神奇群	NULL	20
200	灌水群	NULL	200
300	基友群	NULL	58

table:user_friend

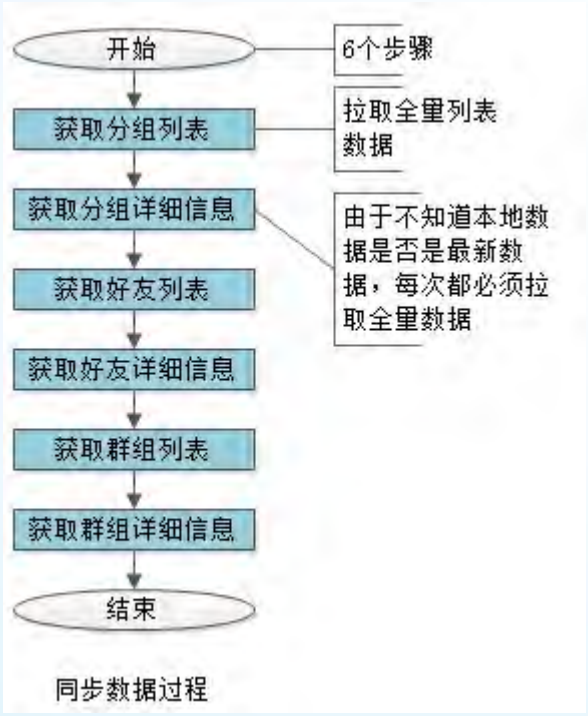
id	friend_id
1	2
1	3
1	4

table:user_group

id	group_id
1	100
1	200
1	300

本地数据+时间戳优化（二）

- 原有流程？缺点：每次拉取全部数据



同步数据过程

table:list_timestamp

id	friend_list	group_list
1	10	100

table:user_info

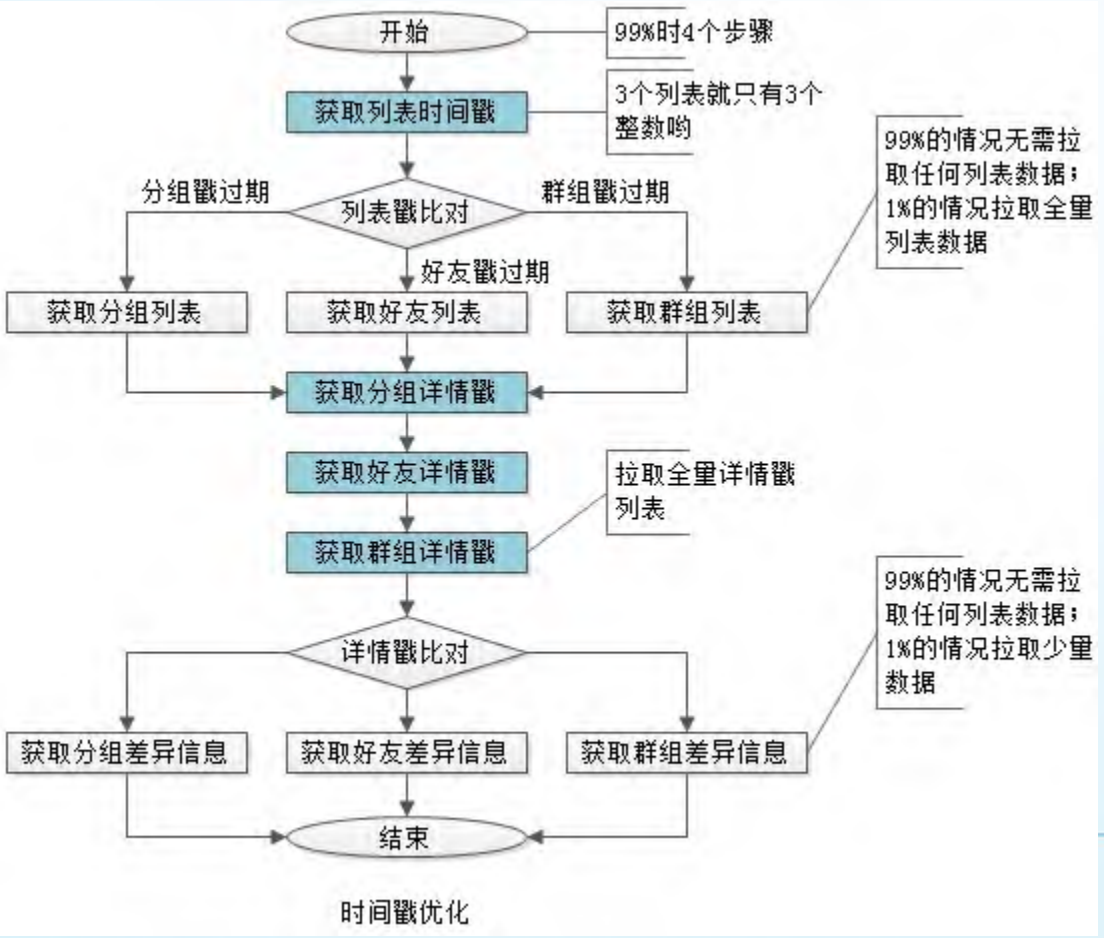
id	nick_name	gender	birthday	timestamp
2	酷	男	1979	10
3	帅	男	NULL	100
4	靓	男	NULL	1000
5	拽	男	NULL	58

table:group_info

id	group_name	intro	timestamp
100	神奇群	NULL	20
200	灌水群	NULL	200
300	基友群	NULL	58

本地数据+时间戳优化（三）

- **优化流程：使用时间戳**　　缺点：增加了交互次数



时间戳优化

table:list_timestamp

id	friend_list	group_list
1	10	100

table:user_info

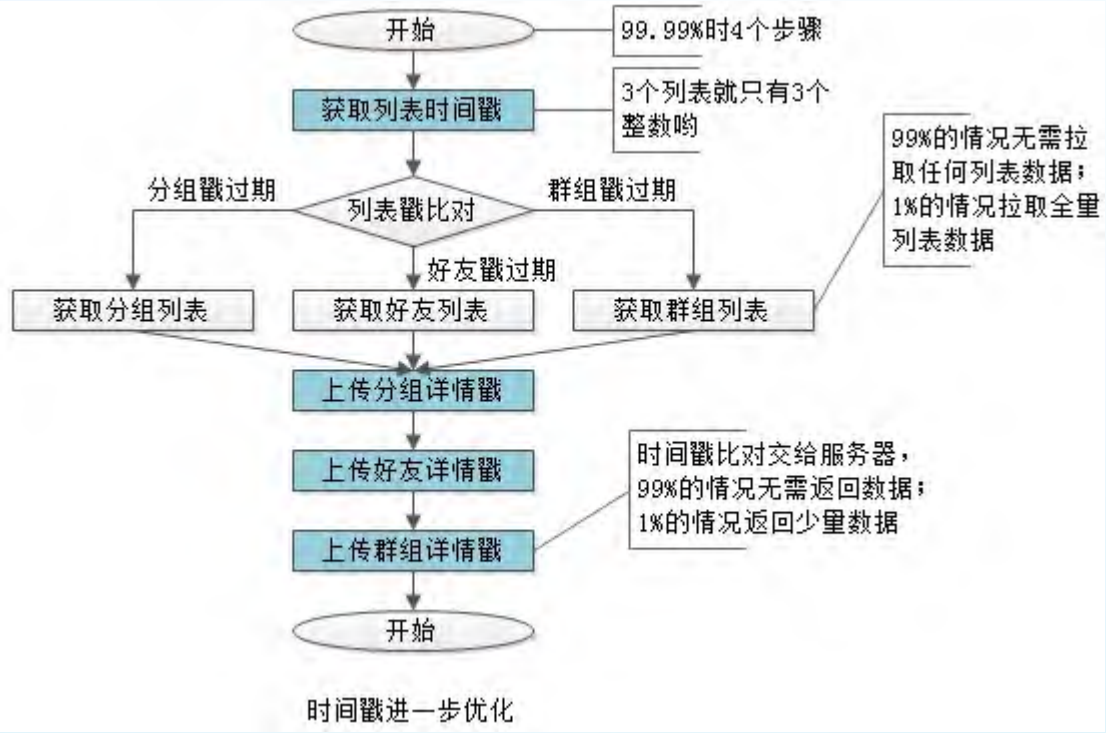
id	nick_name	gender	birthday	timestamp
2	酷	男	1979	10
3	帅	男	NULL	100
4	靓	男	NULL	1000
5	拽	男	NULL	58

table:group_info

id	group_name	intro	timestamp
100	神奇群	NULL	20
200	灌水群	NULL	200
300	基友群	NULL	58

本地数据+时间戳优化（四）

- **进一步优化流程：拉取列表数据戳，上传详情数据戳**



时间戳进一步优化

优化上报日志（一）

- 如何统计“按钮A”点击了多少次，有多少人点击？
- 简易方法：Google Analytics
- 粗暴方法：自己上报日志

curl
http://www.bangbang.58.com/report/up?
uid=1&action=click&key1=xxx&key2=xxx&key3=xxx

- 缺点？

（1）无效流量多（2）url冗余（3）key冗余（4）上报频繁

优化上报日志（二）

- 优化方向？

（1）协议优化

（2）url优化

（3）kv优化

（4）频度优化 => 借鉴map-reduce的思想 => 合并函数

延迟合并上报

- 延时上报策略

（1）打开、退出APP时上报

（2）每隔X分钟上报

（3）每积累Y条上报

五、消息可达性优化

消息可达性优化（一）

- im是基于通知的系统，3种报文类型
- 消息投递流程
- 存在的问题？

（1）发送方收到msg:R包不代表接收方收到消息

（2）msg:N包有可能丢失

消息可达性优化（二）

- 58im消息可达性优化
- **优化一：应用层ACK保证可达性**
- **核心技术：一条消息发送的6个请求**
- 存在的问题？

（1）msg:N包可能丢失

（2）ack:N包可能丢失

消息可达性优化（三）

- “因为网络原因，消息发送失败”意味着什么？（没有收到ack:N）
- **优化二：发送方等待ack队列，ack超时时消息重传**
- 超时与重传存在的问题？消息重复
- **优化三：接收方去重**

总结（一）

- 一、im与其他系统架构不同之处在于多了一个**路由层**
- **二、稳定性优化**

（1）Request-Ack式的请求，可以优化为短连接

（2）跳过DNS，直接使用ip连接（不是在客户端写死ip哈）

（3）session保持，TCP与session不再强绑定

（4）客户端记录上次接入层ip，快速重连

总结（二）

- **三、流量优化**

（1）id和info分开拉取

（2）延迟拉取，按需拉取

（3）时间戳优化：拉取列表时间戳、上传详情时间戳

（4）优化日志上报：协议优化、url优化、kv优化、合并函数

- **四、消息可达性优化**

（1）通过超时、重传、应用层确认、去重的机制来保证消息的可靠投递，不丢不重

（2）业务层的不重复，系统层的不可能不重复

（3）切记，一个“你好”的发送，包含上半场msg:R/A/N与下半场ack:R/A/N的6个报文

Q&A

谢谢！

“架构师之路”公众号

THANKS

SequeMedia 盛拓传媒 | ITPUB