58同城Webim实现与优化

沈剑
shenjian@58.com

关于我-58沈剑

- 前百度-高级工程师
- 58同城-高级架构师
- 58同城-技术委员会执行主席
- 58同城-技术学院优秀讲师
- 58同城-业界技术分享嘉宾
- 58同城-C2C技术部负责人
- 58同城-**程序员一枚**

关于58im及Webim-58帮帮

- **日请求量：十亿级别**（哪些请求量最大？）
- 数据库量：亿级别
- 日消息量：千万级别
- 同时在线：百万级别
- 机器数量：50左右

目录-58Webim实现与优化

- Webim实现难点
- Webim架构简介
- Webim如何实现消息推送
- 58Webim**消息实时性**优化
- 58Webim**消息可达性**优化

（1）个人消息

（2）离线消息

（3）群消息

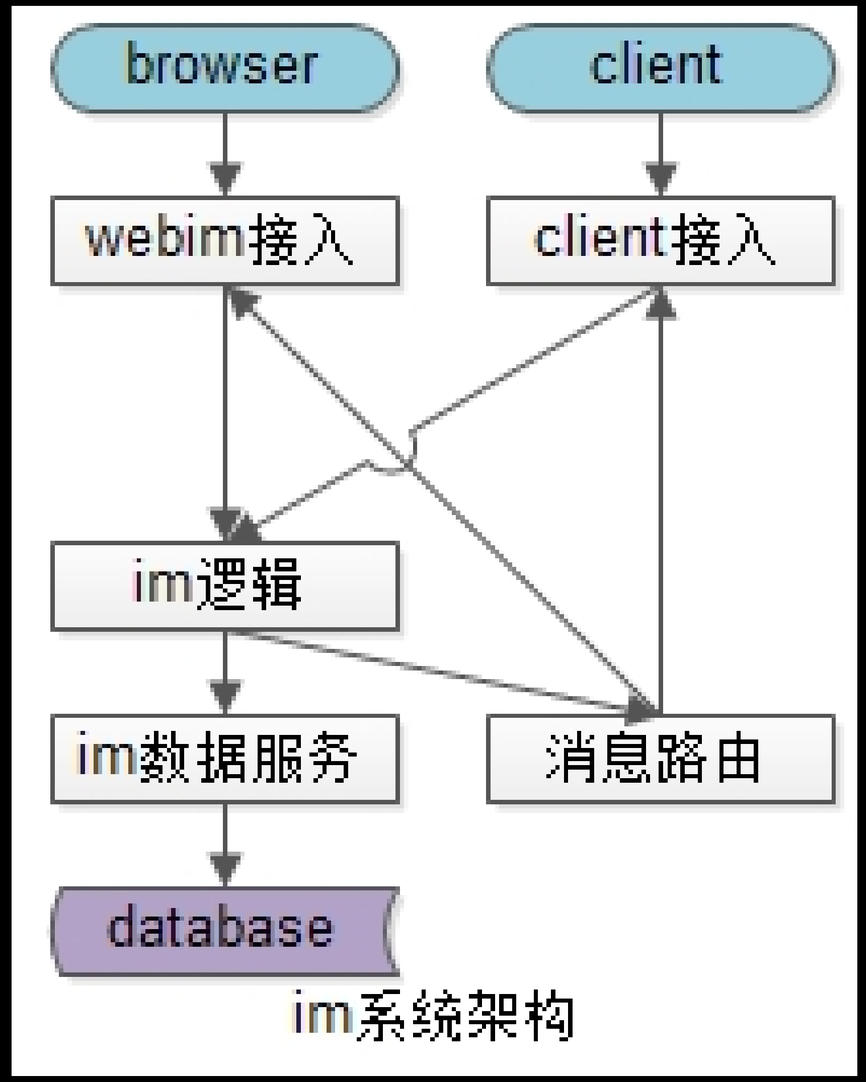
- 58Webim**状态一致性**优化
- 58Webim匿名聊天与熟客识别
- 58Webim多TAB同步优化

一、Webim难点

- 基于推送的系统
- 消息实时性
- 消息可达性
- 状态一致性
- 匿名登录，熟客识别
- 多TAB同步

二、Webim架构简述

- Web接入层
- 逻辑处理层
- **消息路由层**（哪里和传统系统架构不一样？）
- 数据存储层



im系统架构

三、Webim如何实现消息推送

- WebSocket
- FlashSocket
- **Http长轮询**（今天重点讲解）

四、58Webim消息实时性优化

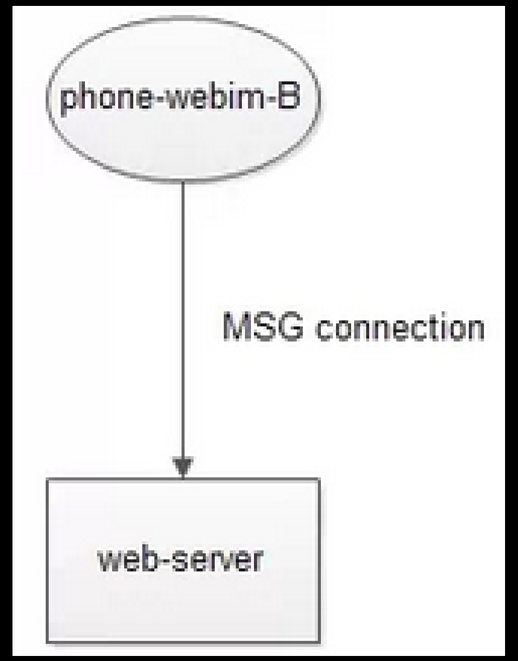
- WebSocket，FlashSocket实时
- 为什么人们会误解Http长轮询不实时？

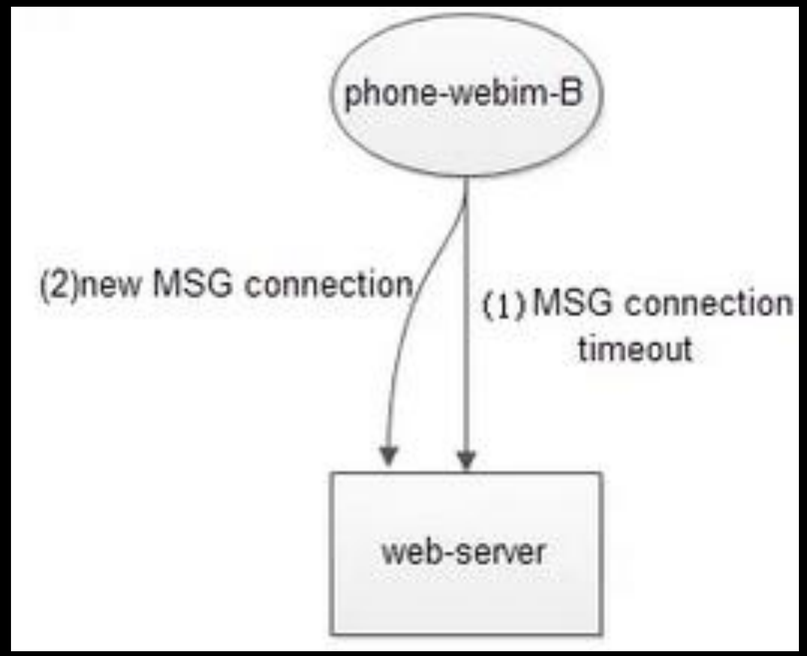
（1）什么是轮询？

（2）轮询存在什么问题？

（3）缩短轮询时间能否解决时延问题？

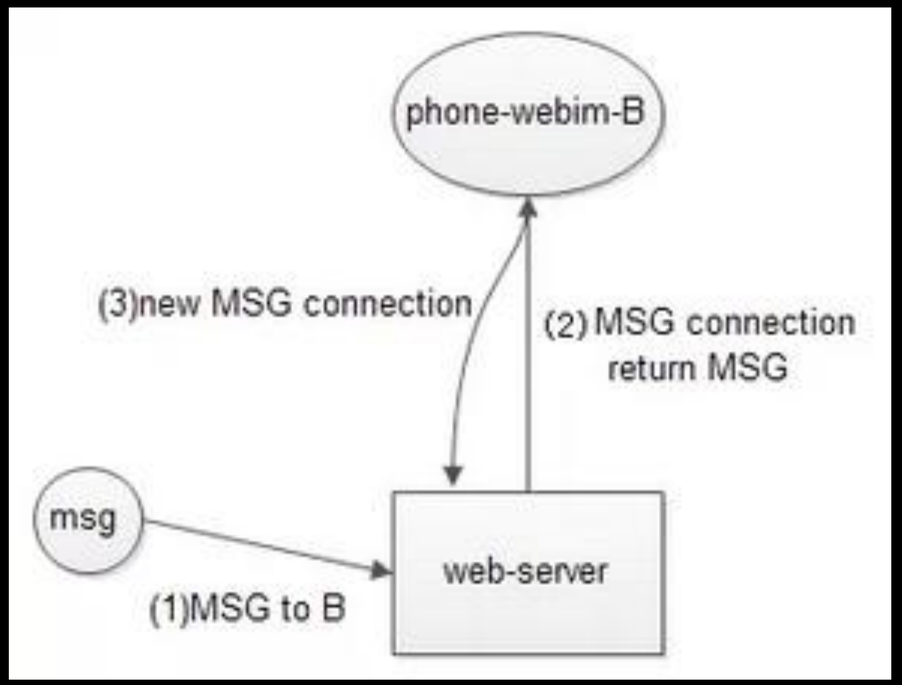
- 什么是真正的Http长轮询？
- 什么是消息连接？
- **特性一：夯住**
- **特性二：超时后再起发起**

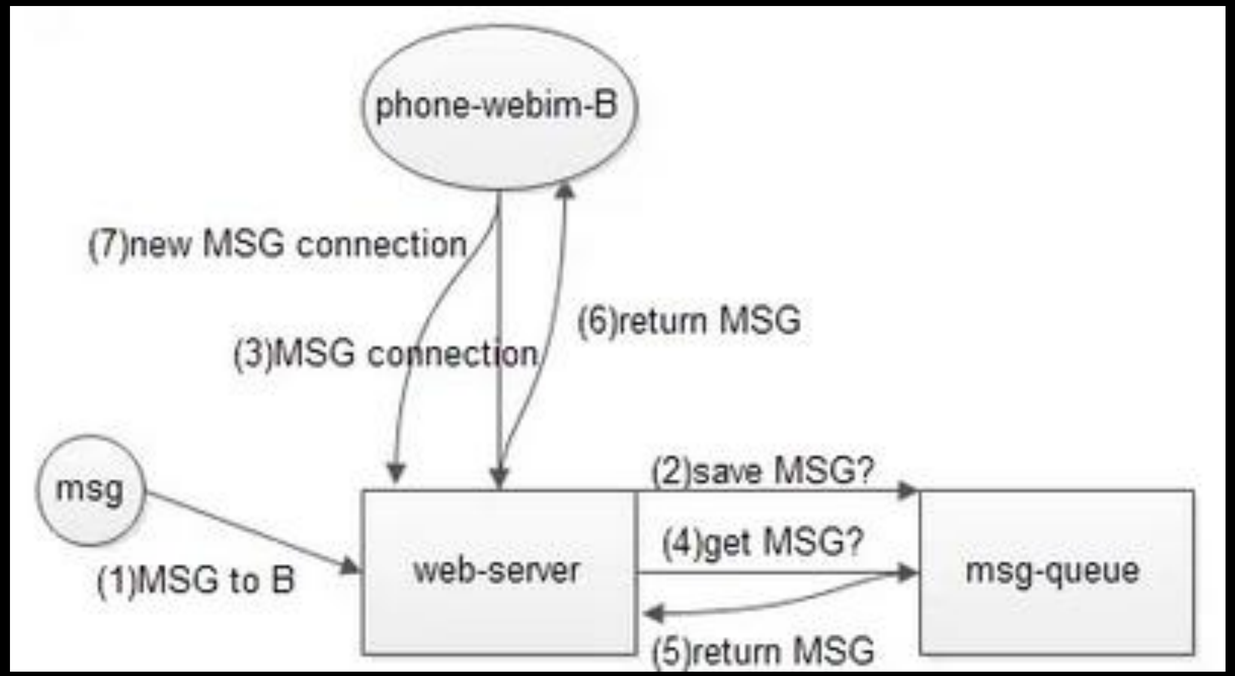




四、58Webim消息实时性优化

- 特性三：消息到达，实时返回，再次发起
- 特性四：无连接则入消息池，有连接后实时返回，再次发起





四、58Webim消息实时性优化

- 结论

（1）webim的实时性**不是通过缩短轮询时间来保证的**

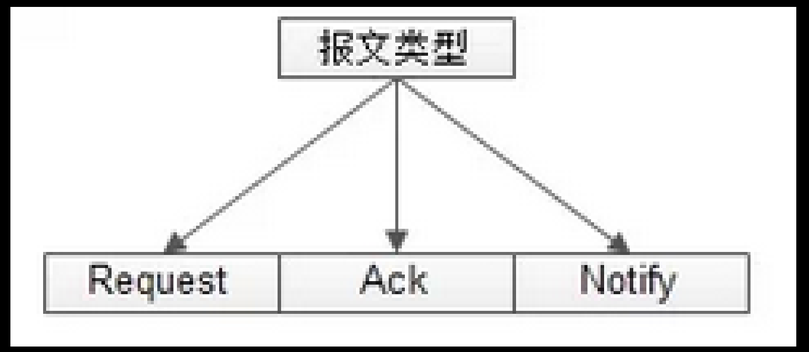
（2）webim的实时性**是通过夯住http消息连接来保证的**

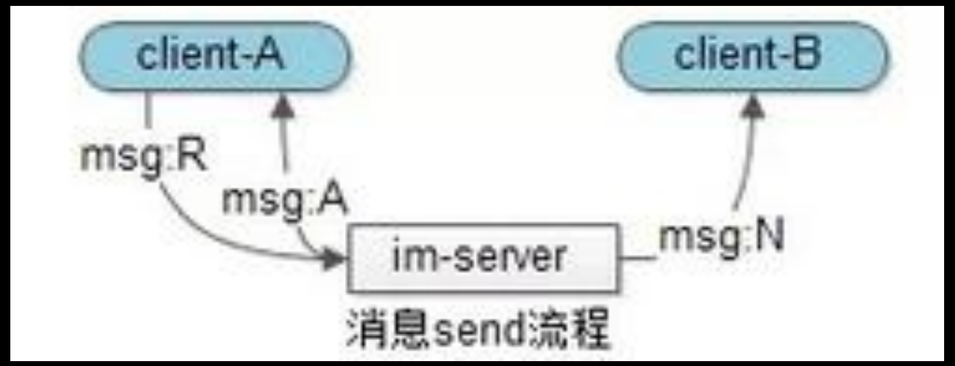
五、58Webim消息可达性优化（单人聊天）

- Webim是基于通知的系统，3种报文类型
- 普通消息投递流程
- 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



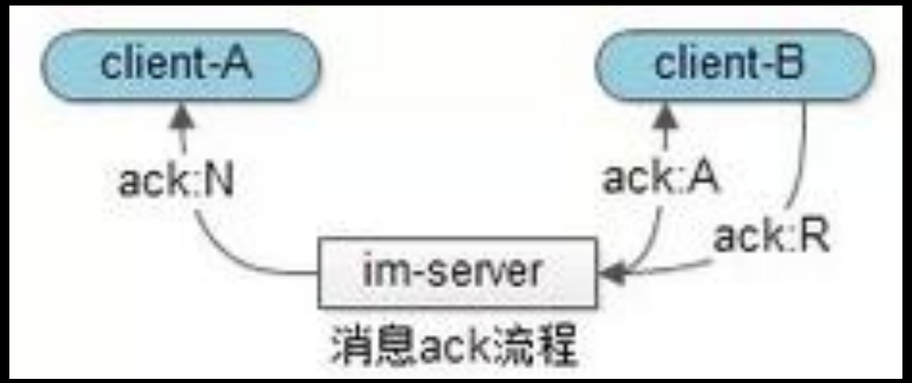


五、58Webim消息可达性优化（单人聊天）

- 58Webim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？



五、58Webim消息可达性优化（单人聊天）

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

（1）消息的重复

（2）目标用户不在线ack怎么办？

- **优化三：接收方去重**

五、58Webim消息可达性优化（单人聊天）

- 总结

（1）im系统是通过超时、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）im系统的消息不重复，只是业务层的不重复，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

五、58Webim消息可达性优化（离线消息）

- 离线消息拉取的过程：返回并删除离线消息
- 存在的问题？

（1）离线消息返回过程中丢包

- **优化方案：增加应用层拉取离线消息的ack**
- 新的问题？

（1）拉取离线消息的ack丢失，会拉取到重复的离线消息

- **优化方案：业务层离线消息去重**
- 结论：通过应用层ack和应用层去重保证离线消息拉取的可达性

五、58Webim消息可达性优化（群聊天）

- 什么是群聊？1 -> N
- 群聊的难点？

（1）能直接应用层ack么？

（2）能让发送方重发么？

（3）能让服务器重发么？服务器需要记录状态么？

- 群聊如何保证可达性，两种方案：

（1）基于登录时间

（2）基于ack，并降低ack频率

（2.1）基于时间降频

（2.2）基于消息条数降频

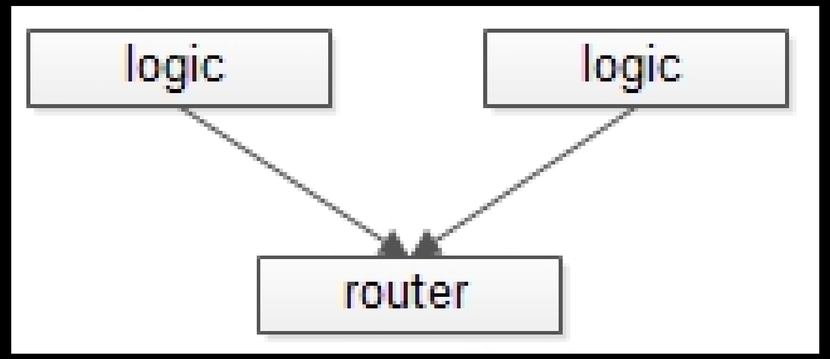
- 应用层去重必不可少

六、58Webim状态一致性优化（业务层）

- XMPP（Extensible Messaging and **Presence** Protocol）
- 什么是“出席”？
- 状态什么时候扩散？（推）=> 对一致性要求高的业务场景
- 状态什么时候拉取？（拉）=> 对一致性要求低的业务场景

六、58Webim状态一致性优化（系统层）

- 状态存储在哪里？
- 状态可用性如何保证？



六、58Webim状态一致性优化（系统层）

- **架构设计核心思路：保证可用性的方法是冗余**

（1）站点高可用，冗余站点

（2）服务高可用，冗余服务

（3）数据高可用，冗余数据

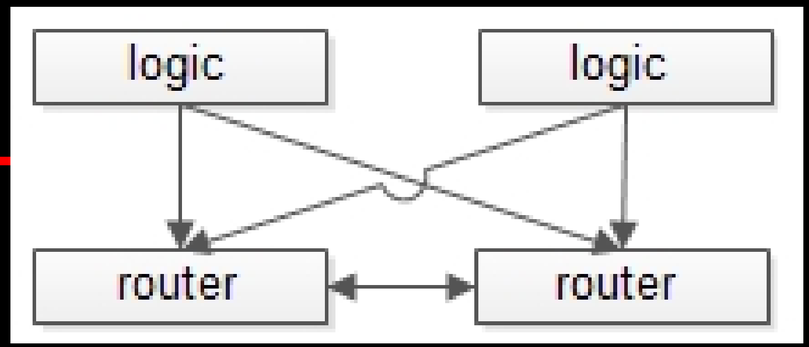
- 数据高可用有什么副作用？
- **架构设计注意点：数据冗余的副作用是一**

（1）主从数据库不一致

（2）数据库与缓存不一致

（3）内存冗余数据不一致

- 状态一致性如何保证？



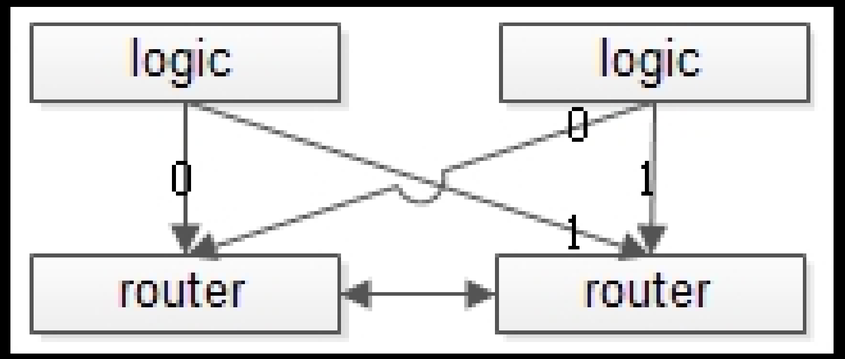
六、58Webim状态一致性优化（系统层）

- **优化方案：串行化**

（1）两个router保存全量数据

（2）两个router相互同步数据

（3）每个router只为一半的用户服务



- 存在的问题？

（1）router内存放不下怎么办？

- **优化方案：水平切分**

六、58Webim状态一致性优化

总结

（1）状态扩散推拉结合

（2）在线状态存储在内存

（3）用冗余解决可用性问题，但会引发一致性问题

（4）用串行化解决一致性问题

（5）用水平切分解决扩展性问题

七、58Webim匿名聊天与熟客识别

- 匿名登录

（1）如何区分匿名用户与登录用户？

（2）匿名用户的信息如何在客户端存储？

（3）匿名用户的信息如何在服务端存储？

- 熟客识别

（1）匿名用户的身份标识如何在客户端存储？

（2）匿名用户的身份标识如何防止伪造？

八、58Webim多TAB同步优化

- **Flash-Socket如何做多TAB同步**

（1）主机，焦点从机，非焦点从机

（2）主机和服务器连接，从机和主机连接

（3）通过ShareObject传递数据

（4）主从机心跳，重选主机

（5）主机将最新数据发给焦点从机，非焦点从机从ShareObject同步数据

- **Http长轮询如何做多TAB同步**

（1）焦点TAB和非焦点TAB

（2）焦点TAB和服务器建立消息连接

（3）通过LocalStorage（UserData）传递数据

（4）非焦点TAB从LocalStorage同步数据

（5）跨域问题

总结

- webim系统和其他系统，架构的不同之处，在于有个**路由层**
- webim实现推送主要有**WebSocket、FlashSocket、Http长轮询**三种方式
- webim通过**消息连接**来保证消息的绝对实时性
- webim通过应用层的**超时、重传、确认、去重**来保证消息的可靠投递（不丢不重）
- webim一个“你好”的**消息投递，涉及6个报文**
- webim状态一致性优化方法

（1）业务上，推拉结合

（2）在线状态存储在内存

（3）用冗余解决可用性问题，但会引发一致性问题

（4）用串行化解决一致性问题

（5）用水平切分解决扩展性问题

谢谢！

“架构师之路”公众号