目录

01.IM系统重要功能

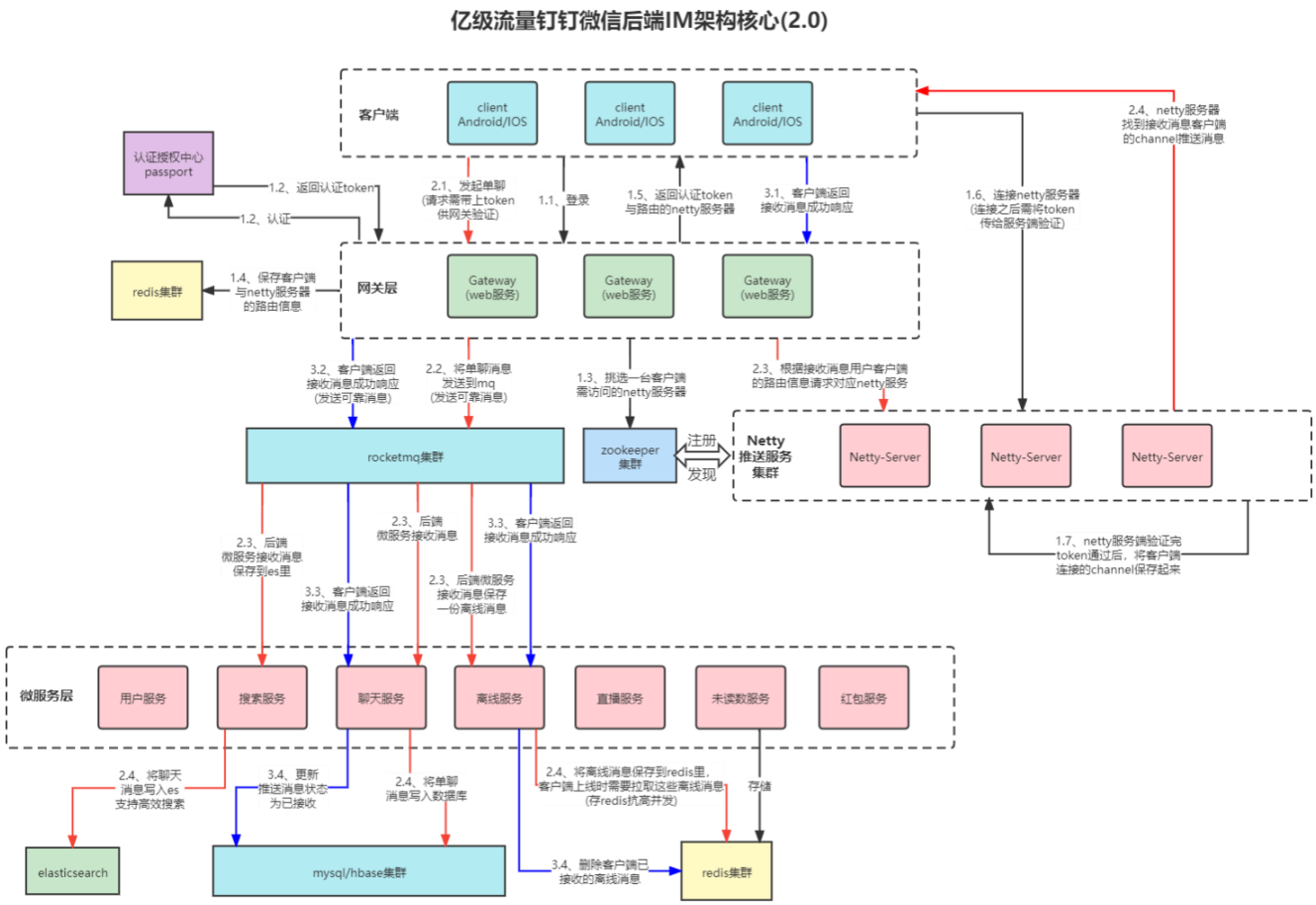
IM系统重要功能：

- **未读数**：指用户还没读的消息数量。
- **单聊**：一对一聊天。
- **群聊**：多人聊天。

IM系统基础架构：

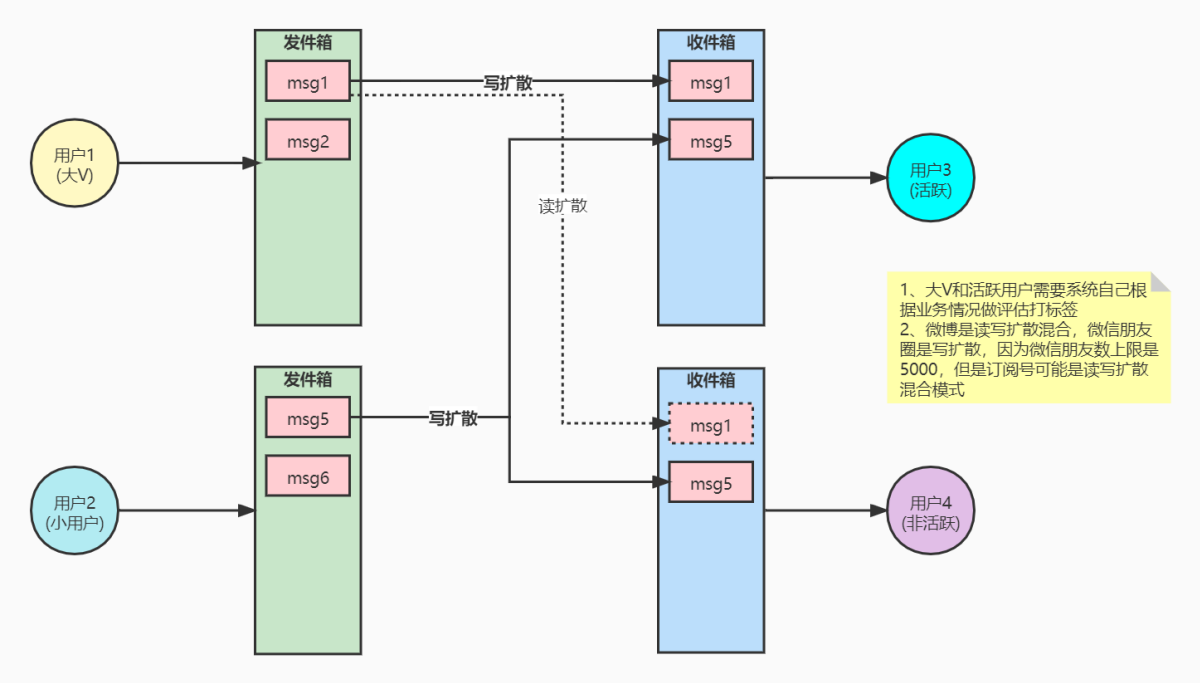
- 客户端：作为与服务端进行消息收发通信的终端
- 网关层：也叫接入层，为客户端收发消息提供入口
- 服务层：负责IM系统各功能的核心业务逻辑实现，比如聊天服务、离线消息服务、红包服务、直播服务等
- 存储层：负责IM系统相关数据的持久化存储，包括消息内容、账号信息、社交关系链等

02.IM系统高并发架构



- 聊天系统消息的可靠投递(不丢消息)
    1. IM客户端发送消息如果超时或失败需要重发，客户端在发送消息时需要给**每条消息生成一个id，IM服务端根据此id做好去重机制**
    2. **使用Rocket MQ的可靠消息机制**
    3. 通过**客户端的ACK确认接收消息的机制来保证不丢消息**
- 离线消息服务保证IM系统的高性能
    1. 用户上下线可能是非常频繁的操作，一般是在用户上线时会主动拉取服务端的离线消息。一般会选择一些高性能的**缓存来存储**，比如Redis，这样能抗住高并发的访问压力。
    2. 限制只存储最近一周或一个月的数据；**一个用户的离线消息最多就存储最近的1000条。**或者都按照存储条数的限制。
    3. **就展示最近的一些离线消息**，如果用户一直往上翻离线消息，后面的消息可以从数据库查询
- 海量历史聊天消息数据存储方案
    1. 存储结构使用数据库表结构
    2. 发送消息处理：用户1给用户2发送一条消息，需要在消息内容表里存储一条记录，同时需要在用户信箱消息索引表存储两条记录，**一条是用户1的发件箱，一条是用户2的收件箱，因为会存在消息的收发方各自删除记录的情况。**
    3. 查询聊天消息处理：首先可以**先分页查询聊天消息索引的id，**然后再for循环在im_msg_content表查每条消息内容展示。
    4. 数据库的**分库分表**：im_user_msg_box表我们可以按照 owner_uid 来分，im_msg_content表我们可以按照 mid 来分
    5. **收发消息分为用户信箱消息索引表和消息内容表两张表来存储**，因为很多时候消息内容会比较大
        - 如果我们有时候**只是需要读取一些消息收发的关系**，而不关注消息内容的时候我们只需要查询用户信箱消息索引表即可
        - **收发消息方可能存在各自删除消息的情况，**不会导致消息内容这种大数据被存储多份
- Redis数据结构存储离线消息
    1. 添加消息：`zadd offline_msg_#{receiverId}  #{mid}  #{msg}` (score就存储消息的id)
    2. 查询消息：`zrevrange offline_msg_#{receiverId} 0 9` (按消息id从大到小排序取最新的十条消息，上拉刷新继续查)
    3. 删除消息：`zremrangebyscore offline_msg_#{receiverId}  min_mid  max_mid` (删除客户端已读取过的介于最小的消息id和最大的消息id之间的所有消息)
    4. 如果单个key消息存储过大，可以考虑**按周或者按月针对同一个receiverId多搞几个key分段来存储**
- 群聊数据收发机制读扩散与写扩散
    - **读扩散**：用户在群里发**一条消息只存一份数据，群里所有人都读同一份消息数据**
    - **写扩散：用户在群里发一条消息会针对群里每个用户都存一条消息索引，然后再单独存储一份消息内容**，这样可以针对用户是否已读做一些处理；群的人数不能太多，否则性能会有问题

- **微博读写扩散架构**



- Lua脚本保证消息未读数的一致性
  - 张三给王五发一条消息，需要维护两个redis key，一个是王五总的未读数key加1，一个是张三_王五的未读数key加1。这两个key的加1操作，我们是需要尽量保证它的原子性的
    - 为什么要单独维护，而不是对所有消息会话的未读数求和：如果用户的消息会话比较多的话，那可能就会有性能瓶颈了，在**求和的过程中，可能之前读取的消息会话未读数又发生变化了，所以我们一般会单独维护总未读数。**
  - 当客户端读了消息了，就给服务端发消息，服务端收到后更新未读数。未读数一般来说是在客户端自己去维护的，**服务端的未读数更多是为了给客户端多端数据同步用的。**
  - 对于群聊的未读数，一般可以**针对群里每个人维护一个未读数key，比如用hash结构来存储**，一个群里的所有用户的未读数可以用一个：`hincrby msg:noreadcount:gid uid 1` (gid为群id，uid为用户id)
- 万人群聊系统和百万在线直播互动
  - 未读数更新高并发问题：**因为未读数其实主要是在客户端自己维护的，服务端维护主要是为了多端同步，所以我们可以不实时更新服务端的未读数，而改为定时批量更新，到了5秒了，再统一往redis里刷一次，在其他端要同步的时候可以触发一次内存未读数刷新redis的操作之后再同步。**
  - 活跃群和百万直播的消息高并发发送问题：
    - **我们其实可以不经过redis，直接把消息投递给所有的netty服务器，我们可以让每台netty服务直接将消息推送给自己服务器上对应连接的直播间用户。也可以让所有的netty服务器监听mq的消息。**
    - **在netty服务器上可以维护好直播间或万人群聊里连接到本台服务器的用户列表缓存，这样只要收到直播间或群聊消息，直接根据直播间号或者群聊id找到对应用户连接channel推送消息。**
- 熔断限流机制：大V出镜直播，会导致消息量暴增，是需要做一些**限流措施的，可以丢掉部分消息，以确保整个系统的稳定，一般客户端在直播间，每秒接收几十条消息已经快到极限了。**