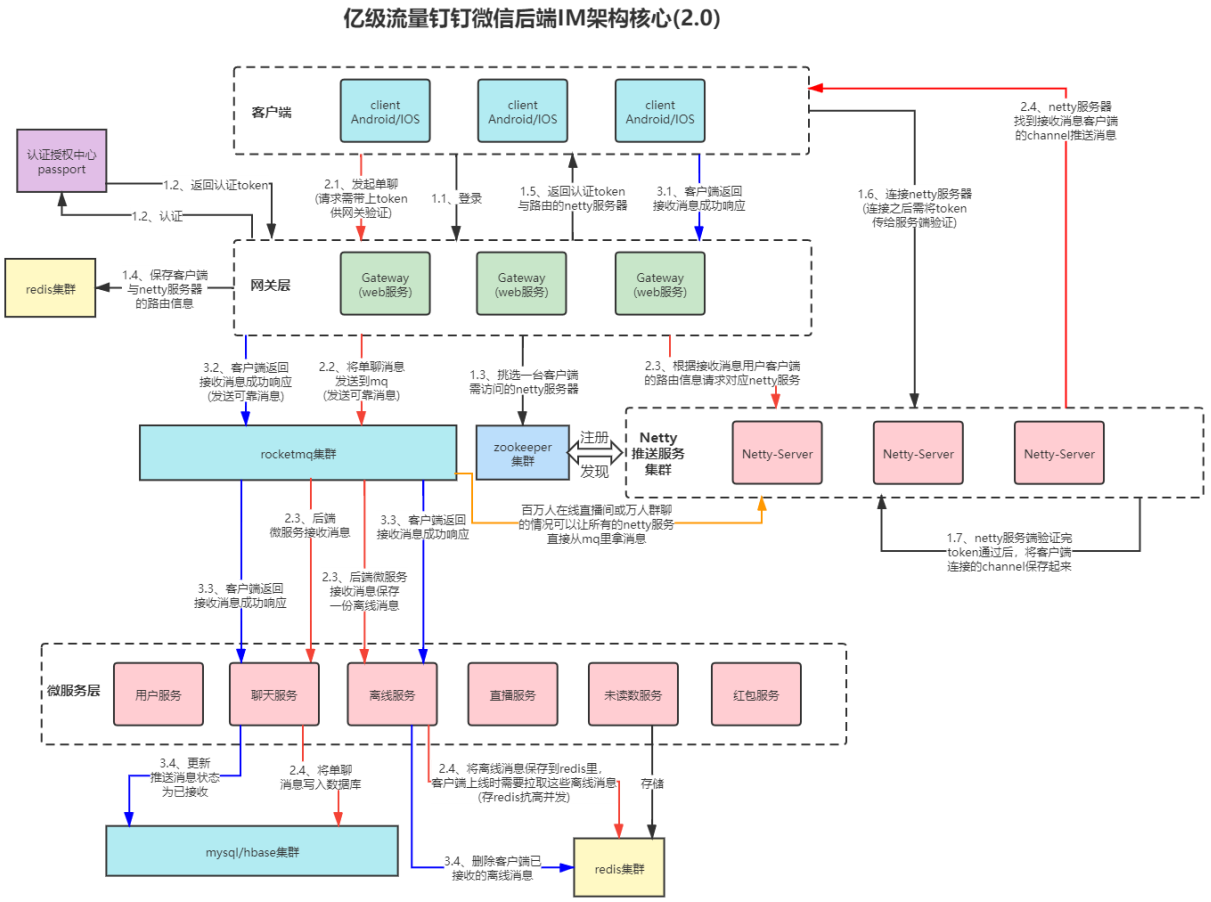
## 企业级IM系统核心架构图




## 如何保证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. 为保证服务端消息不丢失，我们可以使用Rocket MQ的可靠消息机制来保证
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. 离线消息就是用户不在线时别人发给他的消息，到用户上线时这些消息需要接收到，因为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，如果直接从数据库里拉，则会对数据库造成极大的压力，所以对于离线消息我们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. 当然Redis肯定是集群架构，而且会是很多节点，当然有同学也会担心这些离线消息肯定也是非常多的，Redis集群能存下吗，在大厂里Redis都是有很多节点的，可以存储很多T的数据，据说十年前新浪微博后端的Redis存储数据就已经达到几百T级别了，当然我们是可以设置一些存储策略的，比如，限制只存储最近一周或一个月的数

据，然后再加一个存储消息的条数限制，比如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. 因为本身用户上线后查看离线消息很少会把历史所有的离线消息全部看完的，我们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，这种小概率的操作让数据库抗下来是没问题的。

**海量历史聊天消息数据存储方案详解**

1. 消息存储结构参考数据库表结构设计
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，为什么要存储两条记录，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
    - 查看用户1跟用户2的聊天记录，首先可以先分页查询聊天消息索引的id，select mid,box_type from im_user_msg_box t where t.owner_uid = 1 and t.other_uid = 2 order by mid;(注意要分页查)，然后再for循环在im_msg_content表查每条消息内容展示。
    - 因为聊天消息数据巨大，我们肯定要考虑数据库的**分库分表**方案，im_user_msg_box表我们可以**按照 owner_uid 来分**，这样我们正常的聊天记录查询是不需要跨表查询的，im_msg_content表我们可以**按照 mid 来分**，因为查询消息基本都是按照消息id主键来查，性能非常高。
4. 关于表设计的解释
    - 这里解释下为什么将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，所以分成两张表来存储的好处在于，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，而不需要查询消息内容这种大数据表，对性能有一定提升。
    - 收发消息方可能存在各自删除消息的情况，所以要存储两份消息索引，因为我们将消息索引和消息内容是分开存储的，所以也不会导致消息内容这种大数据被存储多份。