# 选型

我与消息队列的八年情缘： <https://mp.weixin.qq.com/s/PphJjWrihspgDfRWTo-xOw>

哈啰在分布式消息治理和微服务治理中的实践：<https://mp.weixin.qq.com/s/N-vd6he4nsZp-G3Plc4m6A>

ActiveMQ

缺点：高吞吐下，堆积到一定消息量易Hang住；

技术团队发现在吞吐量特别高的场景下，假如消息堆积越大，ActiveMQ有较小几率会Hang住的。

方案：先持久化数据到db，再通过定时任务慢慢发送消息。

RabbitMQ

是管子不是池子。RabbitMQ对消息堆积的支持并不好，当大量消息积压的时候，会导致 RabbitMQ 的性能急剧下降

# 消息重复出现的场景

重复发送消息

发送消息时，消息已经在broker上已经接收到（甚至已经完成持久化），由于网络闪断、客户端宕机等问题，导致服务端对客户端应答失败，如果此时生产者意识到消息发送失败并尝试再次发送消息，消费者后续会收到两条内容相同并且Message ID也相同的消息。

重复消费消息

消息消费的场景下，消息已投递到消费者并完成业务处理，当客户端给服务端反馈应答的时候网络闪断。为了保证消息至少被消费一次，消息队列RocketMQ版的服务端将在网络恢复后再次尝试投递之前已被处理过的消息，消费者后续会收到两条内容相同并且Message ID也相同的消息。

负载均衡时消息重复投递（包括但不限于网络抖动、Broker重启以及消费者应用重启）

当消息队列RocketMQ版的Broker或客户端重启、扩容或缩容时，会触发Rebalance，此时消费者可能会收到重复消息。

比较大的问题是，消息没有成功发出或者发出后消息丢失，以及消费者重复消费

问题1：记录发送状态，补发异常消息。消费者可回调处理结果，甚至进行两侧数据的核算

简单的消息去重解决方案：消费者做好并发环境的幂等性

Exactly Once

RocketMQ Exactly-Once方案： <https://help.aliyun.com/document_detail/102777.html>

即消息肯定会被成功消费，并且只会被消费一次

但这在分布式的场景下想找一个通用的方案几乎是不可能的。

不过如果是针对基于数据库事务的消费逻辑，实际上是可行的。

基于数据库的事务实现的

基于这种方式，的确这是有能力拓展到不同的应用场景，因为它的实现方案与具体业务本身无关——而是依赖一个消息表。

但是这里有它的局限性：消息的消费逻辑必须是依赖于关系型数据库事务。

如果消费的消费过程中还涉及其他数据的修改，例如 Redis 这种不支持事务特性的数据源，则这些数据是不可回滚的。

还有，数据库的数据必须是在一个库，跨库无法解决。

另外，需要特别注意的是：在业务上，消息表的设计不应该以消息 ID 作为标识，而应该以业务的业务主键作为标识更为合理，以应对生产者的重发。

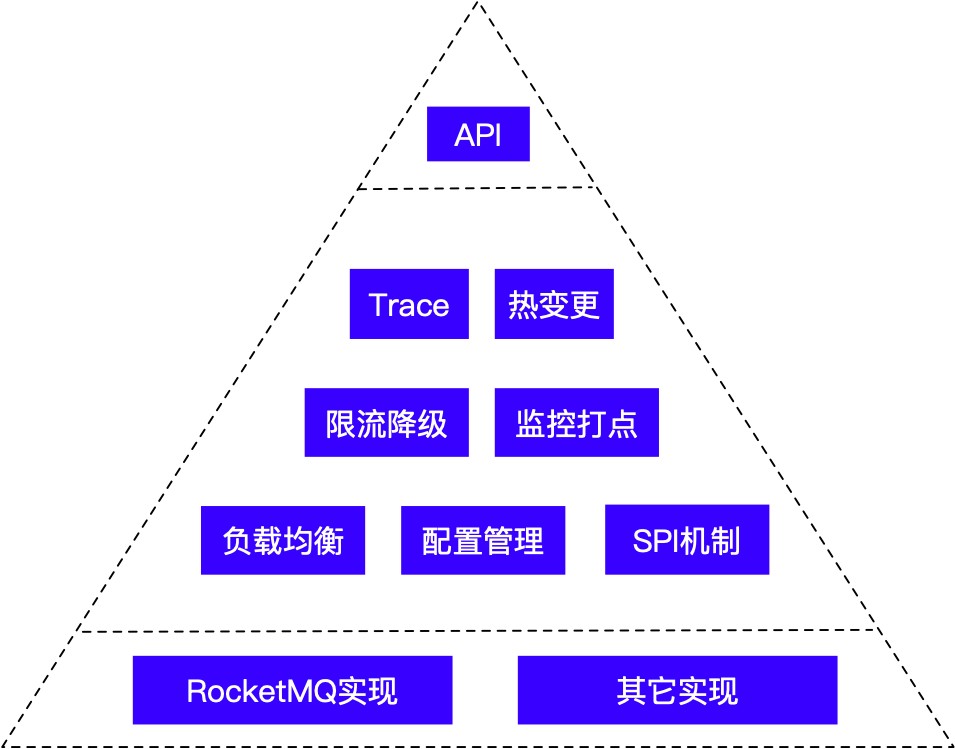
# 封装三方包

我们的思维认知是封装一层可以减少研发人员使用第三方组件的心智投入，统一技术栈，也就如此了。

但是一个糟糕的封装却是灾难的开始

快手消息的SDK封装策略：

1. 对外只提供最基本的 API，所有访问必须经过SDK提供的接口。简洁的 API 就像冰山的一个角，除了对外的简单接口，下面所有的东西都可以升级更换，而不会破坏兼容性;
2. 业务开发起来也很简单，只要需要提供 Topic（全局唯一）和 Group 就可以生产和消费，不用提供环境、NameServer 地址等。SDK 内部会根据 Topic 解析出集群 NameServer 的地址，然后连接相应的集群。生产环境和测试环境环境会解析出不同的地址，从而实现了隔离；
3. 上图分为 3 层，第二层是通用的，第三层才对应具体的 MQ 实现，因此，理论上可以更换为其它消息中间件，而客户端程序不需要修改；
4. SDK 内部集成了热变更机制，可以在不重启 Client 的情况下做动态配置，比如下发路由策略（更换集群 NameServer 的地址，或者连接到别的集群去），Client 的线程数、超时时间等。通过 Maven 强制更新机制，可以保证业务使用的 SDK 基本上是最新的。



短信服务

短信服务应用很广泛，比如用户注册登录验证码，营销短信，下单成功短信通知等等。最开始设计短信服务的时候，我想学习业界是怎么做的。于是把目标锁定在腾讯云的短信服务上。腾讯云的短信服务有如下特点：

* 统一的SDK，后端入口是http/https服务 , 分配appId/appSecret鉴权；
* 简洁的API设计：单发，群发，营销单发，营销群发，模板单发，模板群发。

于是，我参考了这种设计思路。

1. 模仿腾讯云的SDK设计，提供简单易用的短信接口；
2. 设计短信服务API端，接收发短信请求，发送短信信息到消息队列；
3. worker服务消费消息，按照负载均衡的算法，调用不同渠道商的短信接口；
4. Dashboard可以查看短信发送记录，配置渠道商信息。