每月约150亿条消息接收处理，该如何设计?

> 问题简述：一个应用系统A 需要通过HTTP接口接收某外部系统B的消息，系统A 提供3个接口，接收来自系统B的三类消息，每月数据总量约150亿条，经处理后推送前端。

已收录于，我的技术网站 [](http://cxyroad.com/ "https://www./")，有大厂完整面经，工作技术，架构师成长之路，等经验分享

> 请问如何设计:
> -------
>
>
> 这3个接口每次被调用一般就通过请求入参最多送10条数据，大多数情况一次调用大概送两三条数据，总共数据量一个月大概150亿条，其中1个接口T1推送过来的消息实时性要求较高，一般要求15秒内要处理完推送到前端渠道，并且消息量大概占了总量的70%，另外两个接口T2和T3推送过来的消息实时性要求较低，当天能推送给用户即可，这两个接口的消息总量占总量的30%。每次接口调用包含的数据数组大小不超过10，数组里的数据对象为JSON对象，不超过10个字段，每个字段值不超过30个字符。接口是加密的，需要验签解密后得到明文。
>
>
> 系统A 接收数据进行处理的需求如下:
> ------------------
>
>
> （1）拿到接口推送的数据，对于每一条数据进行解密后，查询本地客户信息表（单表，大概3000万条数据），对于属于本地客户的消息数据，调用系统C的接口推送消息给前端渠道。
>
>
> （2）对于上面三类消息的所有数据，每一类消息都需要推送给数据仓库，目前推送的方式是每天生成文件的方式推过去。
>
>
> (3) 所有消息不能丢失，接口T1收到的数据要尽量按接口被调用的顺序来推送，接口T2、T2的顺序性要求不高。
> 请问，大致需要怎么设计来满足需求？限于资源情况，主要只能考虑用kafka、redis、mysql。

大规模消息接收和处理策略设计

在设计一个能够接收和处理每月约150亿条消息的系统时，我们需要综合考虑系统的实时性、可靠性和可扩展性。以下是针对该问题的一种详细设计方案。

一、系统架构设计

消息接收层

HTTP 接口：系统A提供三个HTTP接口（T1, T2, T3）接收来自系统B的消息。考虑到高并发情况，可以使用Nginx作为反向代理，并通过负载均衡将请求分发到多个后端服务器。

加密和验签：每条消息需要经过验签和解密，可以考虑使用独立的微服务来处理这部分逻辑，以减少主业务逻辑的负担。

消息队列层

Kafka：使用Kafka作为消息队列系统，将接收到的消息推送到不同的Topic中。T1的消息推送到`Topic_T1`，T2的消息推送到`Topic_T2`，T3的消息推送到`Topic_T3`。Kafka能够保证消息的顺序性和高吞吐量，非常适合这种大规模消息处理的场景。

数据处理层

实时处理（T1）：对于实时性要求高的T1消息，可以使用Kafka Streams或Flink来处理。消费`Topic_T1`中的消息，解密后查询本地客户信息表（MySQL），并调用系统C的接口推送消息到前端。

批量处理（T2, T3）：对于T2和T3消息，可以采用批量处理的方式。消费`Topic_T2`和`Topic_T3`中的消息，定时批量查询本地客户信息表，并推送消息到前端。

数据存储层

MySQL：本地客户信息表存储在MySQL中。可以使用分表分库策略提高查询性能。对于高并发读写需求，可以考虑读写分离。

Redis：作为缓存层，加速客户信息的查询，减少数据库压力。对于T1的实时性要求，可以将常用客户信息缓存到Redis中。

数据仓库推送

每天将消息数据导出为文件，通过定时任务推送到数据仓库。可以使用Spark等大数据处理工具来生成文件。

消息可靠性

消息存储和重试机制：Kafka保证消息不丢失。同时可以设计消息处理的重试机制，对于处理失败的消息进行重试，确保所有消息都能成功处理。

二、具体流程设计

消息接收和验签解密

* 接收HTTP请求后，调用验签解密微服务进行处理。
* 解密后的消息根据类型推送到对应的Kafka Topic。

T1消息处理流程

* 消费`Topic_T1`中的消息。
* 解密并查询本地客户信息表（先查询Redis缓存，未命中再查MySQL）。
* 调用系统C的接口，推送消息到前端。
* 处理完毕后，将消息标记为已处理。

T2和T3消息处理流程

* 定时批量消费`Topic_T2`和`Topic_T3`中的消息。
* 解密并批量查询本地客户信息表。
* 批量推送消息到前端。
* 处理完毕后，将消息标记为已处理。

数据仓库推送流程

* 每天定时从Kafka中消费所有类型的消息，生成文件。
* 将文件推送到数据仓库。

三、性能优化

1. 负载均衡和集群化：通过Nginx和后端服务器集群处理高并发请求。
2. 缓存优化：使用Redis缓存客户信息，减少数据库查询压力。
3. 数据库分库分表：对MySQL进行分库分表，提升查询性能。
4. 异步处理：Kafka异步处理消息，解耦接收和处理流程，提高系统吞吐量。
5. 批量处理：T2和T3消息采用批量处理，减少频繁IO操作，提高处理效率。

四、容错和监控

容错机制

* Kafka具备消息重试和持久化能力，确保消息不丢失。
* 系统C接口调用失败时，进行重试或持久化存储，待后续处理。

监控和报警

* 使用Prometheus和Grafana进行系统监控，监控指标包括请求量、处理时延、错误率等。
* 设置报警策略，当处理时延或错误率超出阈值时，及时报警处理。

五、总结一下

通过上述设计，我们可以构建一个高效、可靠的消息接收和处理系统，满足每

月150亿条消息的处理需求。系统采用Kafka作为消息队列，结合MySQL和Redis进行数据存储和查询，通过异步和批量处理机制，确保消息的实时性和可靠性。

同时，通过负载均衡、缓存优化和数据库分库分表，提升系统的性能和扩展性。

最后，通过完善的监控和容错机制，保障系统的稳定运行。

已收录于，我的技术网站 [](http://cxyroad.com/ "https://www./")，有大厂完整面经，工作技术，架构师成长之路，等经验分享

原文链接: https://juejin.cn/post/7392244532615233573