08 | 发布/订阅模式：流处理架构中的瑞士军刀

2019-05-03 蔡元楠 来自北京

《大规模数据处理实战》

你好，我是蔡元楠。

今天我想要与你分享的是在处理大规模数据中十分流行的一种设计模式：发布 / 订阅模式（Publish/Subscribe Pattern），有些地方也称它为 Pub/Sub。

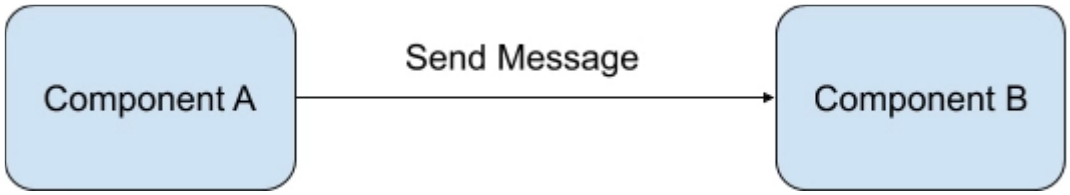
在了解发布 / 订阅模式之前，我想先简单介绍几个基础概念——消息（Message）和消息队列（Message Queue）。

消息

消息是什么呢?

在分布式架构里，架构中的各个组件（Component）需要相互联系沟通。组件可以是后台的数据库，可以是前端的浏览器，也可以是公司内部不同的服务终端（Service Endpoint）。

而各个组件间就是依靠通过发送消息互相通讯的。如下图所示。



消息可以是任意格式的。例如，我们可以利用 JSON 格式来传输一个消息，也能利用 XML 格式来传输一个消息，甚至可以使用一种自己定义的格式。

消息队列

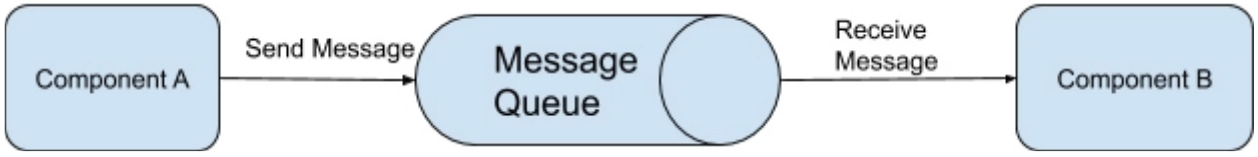
知道了消息的含义后，你知道消息队列有什么作用吗?

消息队列在发布 / 订阅模式中起的是一个持久化缓冲（Durable Buffer）的作用。

消息的发送方可以发送任意消息至这个消息队列中，消息队列在接收到消息之后会将消息保存好，直到消息的接收方确认已经从这个队列拿到了这个消息，才会将这条消息从消息队列中删除。

有的消息系统平台如 Apache Kafka，它能够让用户自己定义消息队列对消息的保留时间，我将会在介绍 Apache Kafka 的时候讲到。

有了消息队列后，整个发送消息的流程就变成下图所示。



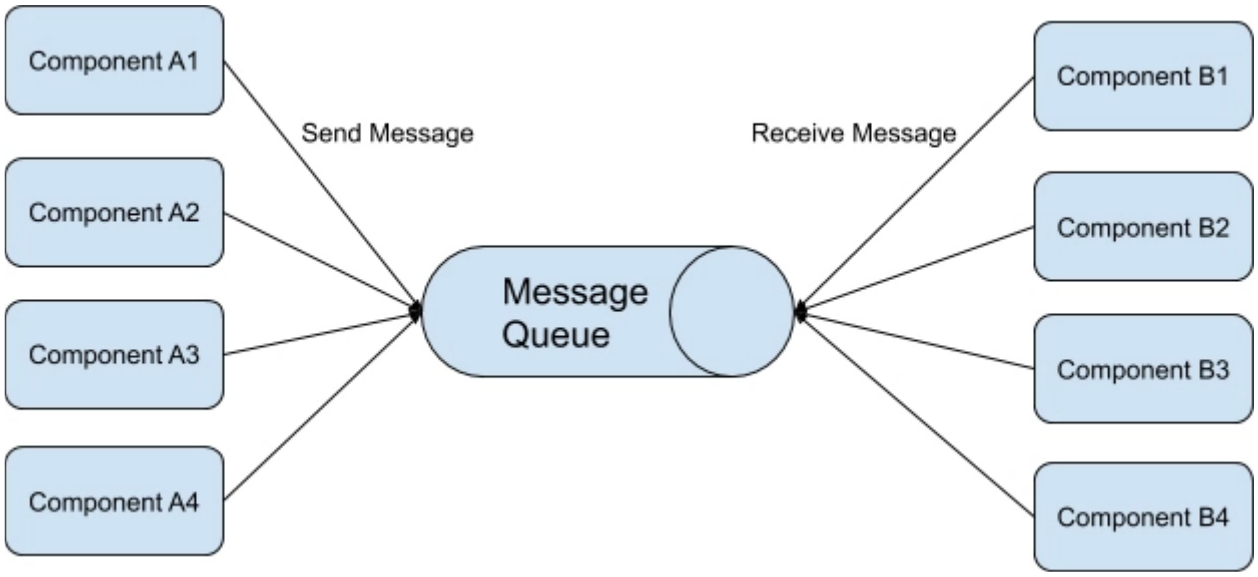
发布 / 订阅模式

在了解了消息和消息队列后，现在我想和你正式地介绍发布 / 订阅模式的概念。

发布 / 订阅模式指的是消息的发送方可以将消息异步地发送给一个系统中不同组件，而无需知道接收方是谁。在发布 / 订阅模式中，发送方被称为发布者（Publisher），接收方则被称作订阅者（Subscriber）。

发布者将消息发送到消息队列中，订阅者可以从消息队列里取出自己感兴趣的消息。

在发布 / 订阅模式里，可以有任意多个发布者发送消息，也可以有任意多个订阅者接收消息，如下图所示。

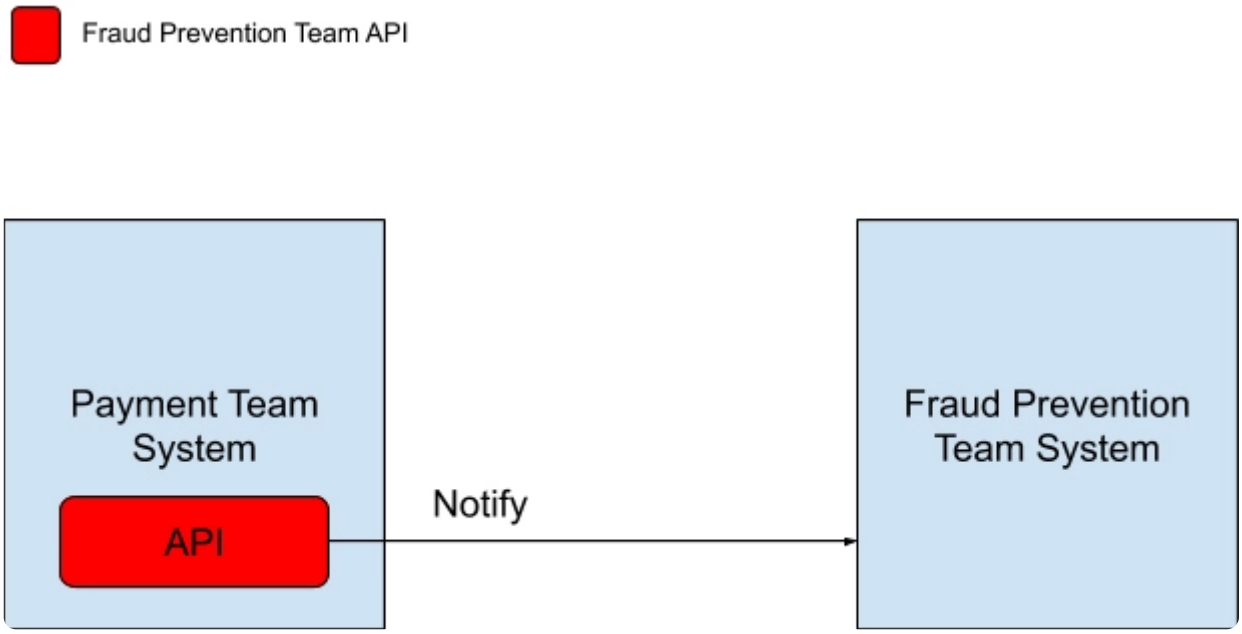


读到这里，你可能会有一个疑问，从概念上看，发布 / 订阅模式只是简单地在消息发送方和消息接收方中间多加了一个消息队列，为什么这种设计架构在实际应用会如此流行呢？我来给你举个例子说明一下。

现在假设，你是一个移动支付 App 公司里支付团队的开发人员，App 里所有的支付操作都是由你的团队来开发的。而公司最近新成立了一个欺诈预防团队，他们希望从你的支付团队里获取交易数据。

也就是说，每次有交易发生的时候，你都需要通知他们交易的金额、地点、时间这些数据，让他们能够实时分析这次的交易是否存在欺诈行为。

按照传统的做法，两个团队需要开会交流，以确定数据消息传输的 API 和传输数据的格式。确定后，两个团队系统的交流方式会如下图所示。



欺诈预防团队将自己需要的数据格式定义在 API 中告诉支付团队，每次有交易产生的时候，支付系统就会通过调用欺诈预防系统 API 的方式通知他们。

一段时间过后，公司希望和商家一起合作推动一项优惠活动，不同的商家会有不同的优惠。公司希望能够精准投放优惠活动的广告给感兴趣的用户，所以又成立了一个新部门，我们叫它广告推荐组吧。

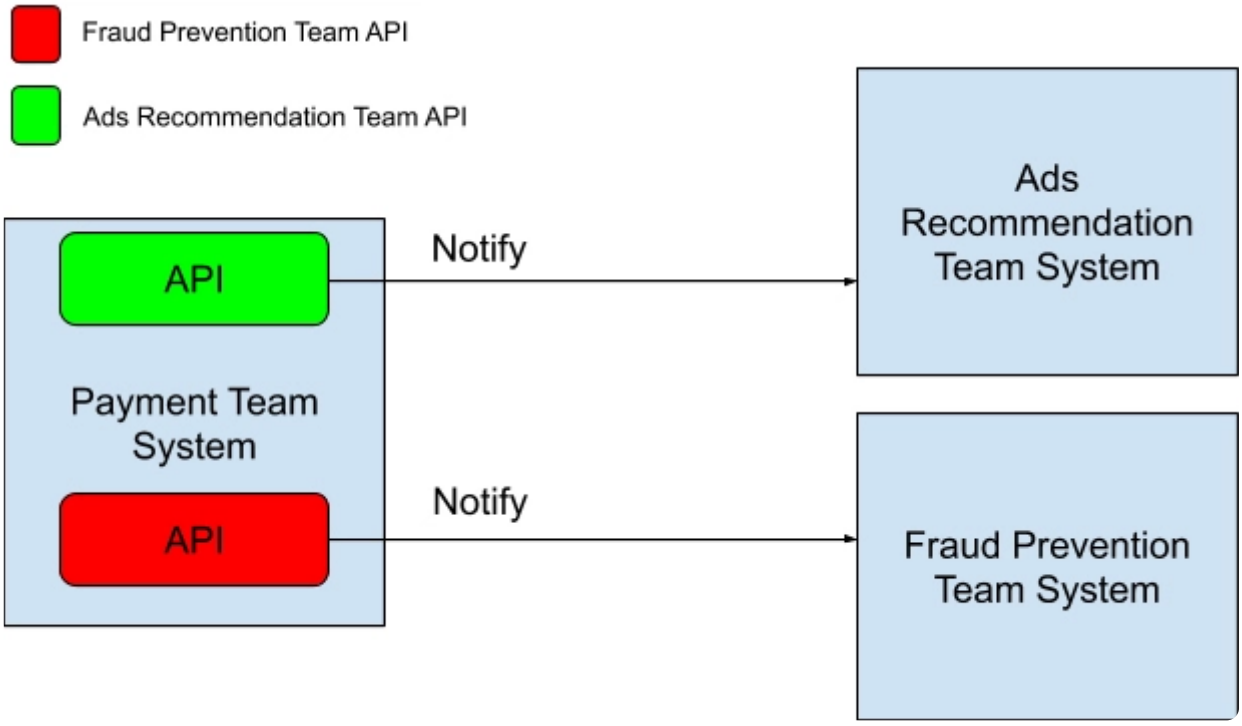
广告推荐组的同事也希望从你的支付团队里获取交易数据。这个时候，你有两种选择，一种是选择第六讲中说到的批处理方式，另一种就是今天讲的发布 / 订阅模式。

批处理方式会从数据库中一次性读取全部用户的交易数据来进行推荐分析。

这种做法有几个不好的地方。

它需要你开放自己数据库的权限给广告推荐组，推荐组每次大量读取数据时，可能也会造成你自己的数据库性能下降。同时，还要考虑广告推荐组也想维护一份自己的数据库的需求。

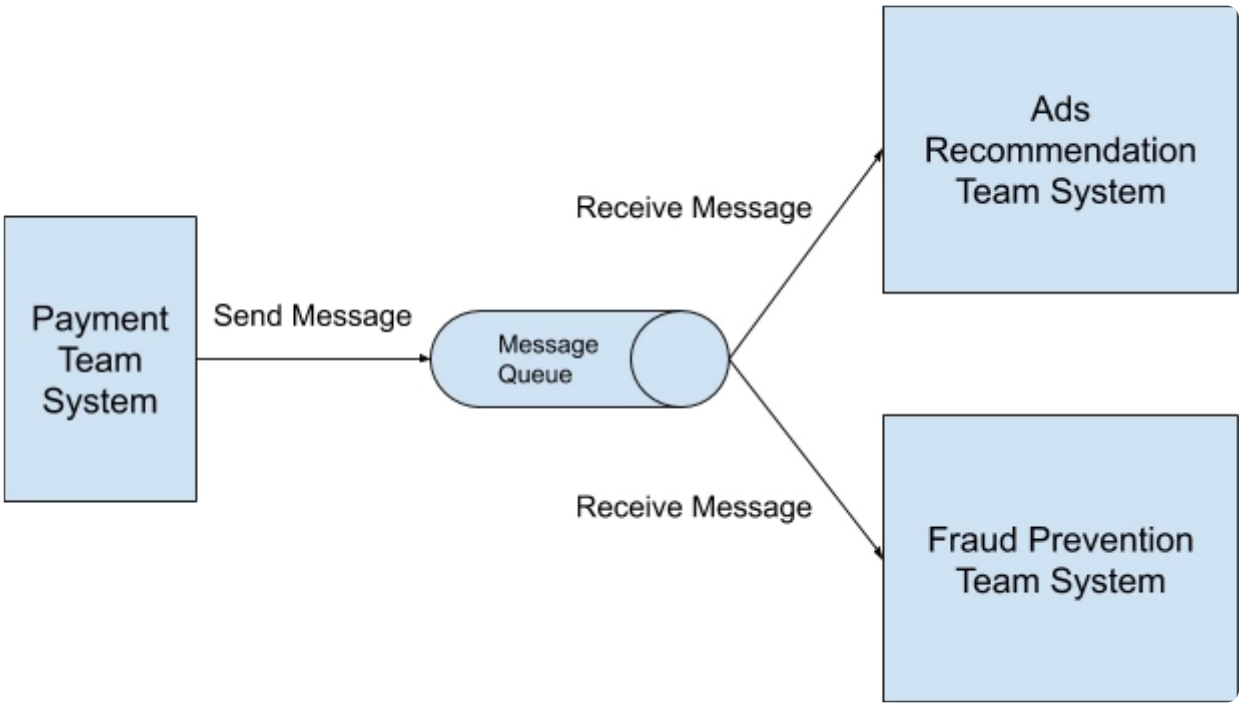
如果还是按照之前欺诈预防团队的做法，让广告推荐组分享 API 给你，每次有交易产生的时候再通知他们的话，系统的运行模式会如文中图片所示。



看到这里你应该明白了。每一次有一个新的系统想从支付团队里读取数据的话，都要双方开会讨论，定义一个新的 API，然后修改支付团队现有的系统，将 API 加入系统中。

而且这些 API 通常都是同步调用的，过多的 API 调用会让系统的延迟越来越大。这样的设计模式被称作观察者模式（Observer Pattern），系统中的各个组件紧耦合（Tightly Coupled）在一起。

如果是采用发布 / 订阅模式来重新设计呢？整个系统就如下图所示：



采用这样的数据处理模式，作为消息发布者的支付团队无需过多考虑以后有多少其它的团队需要读取交易数据，只需要设计好自己提供的数据内容与格式，在每次交易发生时发送消息进消息队列中就可以了。任何对这些数据感兴趣的团队只需要从消息队列中自行读取便可。

发布 / 订阅模式的优缺点

说到这里，我们可以看到发布 / 订阅模式会有以下几个优点：

松耦合（Loose Coupling）：消息的发布者和消息的订阅者在开发的时候完全不需要事先知道对方的存在，可以独立地进行开发。

高伸缩性（High Scalability）：发布 / 订阅模式中的消息队列可以独立的作为一个数据存储中心存在。在分布式环境中，更是消息队列更是可以扩展至上千个服务器中。我们从 Linkedin 公司的技术博客中可以得知，光在 2016 年，Linkedin 公司就维护开发了将近 1400 个消息队列。

系统组件间通信更加简洁：因为不需要为每一个消息的订阅者准备专门的消息格式，只要知道了消息队列中保存消息的格式，发布者就可以按照这个格式发送消息，订阅者也只需要按照这个格式接收消息。

虽然发布 / 订阅模式的数据处理模式优点多多，但是还是存在着自身的缺点的。

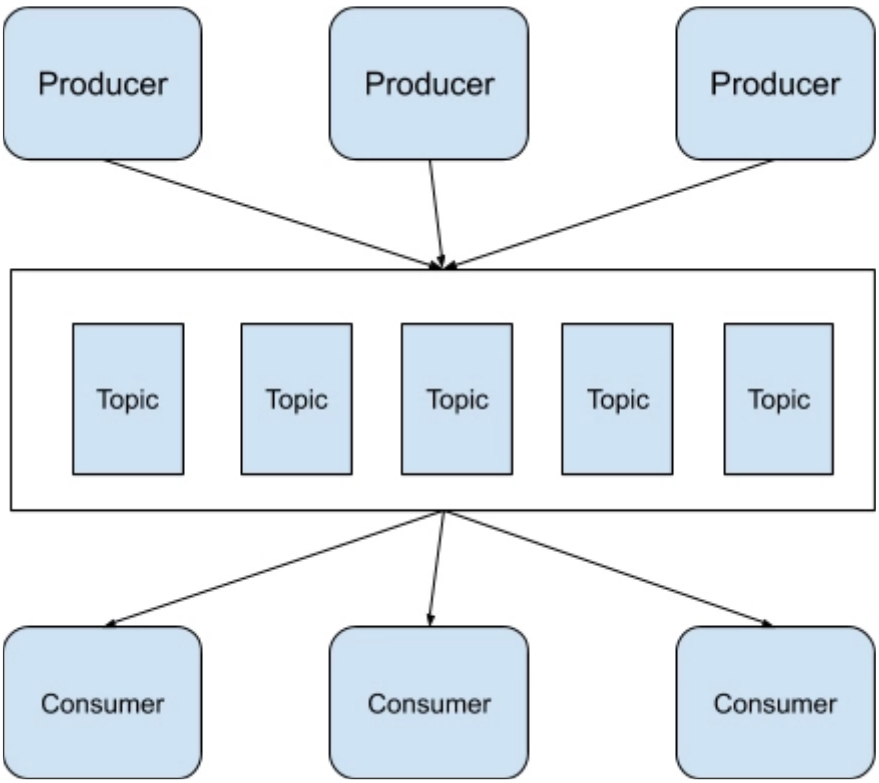
例如，在整个数据模式中，我们不能保证发布者发送的数据一定会送达订阅者。如果要保证数据一定送达的话，需要开发者自己实现响应机制。

在硅谷，很多大型云平台都是运用这个发布 / 订阅数据处理模式。

例如，Google 的 Cloud Pub/Sub 平台，AWS 的 Amazon Simple Notification Service（SNS）。被 Linkedin、Uber 等硅谷大厂所广泛使用的开源平台 Apache Kafka 也是搭建在发布 / 订阅数据处理模式之上的。甚至是连 Redis 也支持原生的发布 / 订阅模式。

Apache Kafka 作为一个被在硅谷大厂与独角兽广泛使用的开源平台，如果你是在创业阶段的话，说不定可以用得上，所以在这里我想简单介绍一下 Apache Kafka。

在 Apache Kafka 中，消息的发送方被称为 Producer，消息的接收方被称为 Consumer，而消息队列被称为 Topic。



Apache Kafka 在判断消息是否被接收方接收是利用了 Log offset 机制。

什么是 Log offset 机制呢？我举个例子给你解释一下。

假设发送方连续发送了 5 条数据到消息队列 Topics 中，这 5 条消息被编号为 10000、10001、10002、10003 和 10004。

如果接收方读取数据之后回应消息队列它接收的 Log offset 是 10000、10001 和 10003，那么消息队列就会认为接收方最多只接收了消息 10000 和 10001，剩下的消息 10002、10003 和 10004 则会继续发送给接收方，直到接收方回应接收了消息 10002、10003 和 10004。

发布 / 订阅模式的适用场景

我们说回到发布 / 订阅模式来，看看它能用在哪些场景。

如果你在处理数据的时候碰到以下场景，那么就可以考虑使用发布 / 订阅的数据处理模式。

- 系统的发送方需要向大量的接收方广播消息。
- 系统中某一个组件需要与多个独立开发的组件或服务进行通信，而这些独立开发的组件或服务可以使用不同的编程语言和通信协议。
- 系统的发送方在向接收方发送消息之后无需接收方进行实时响应。
- 系统中对数据一致性的要求只需要支持数据的最终一致性（Eventual Consistency）模型。

要提醒你注意的一点是，如果系统的发送方在向接收方发送消息之后，需要接收方进行实时响应的话，那么绝大多数情况下，都不要考虑使用发布 / 订阅的数据处理模式。

小结

今天我们一起学习了大规模数据处理中一种十分流行的设计模式——发布 / 订阅模式。它能够很好地解耦（Decouple）系统中不同的组件，许多实时的流处理架构就是利用这个数据处理的设计模式搭建起来的。因为发布 / 订阅模式同时具有很好的伸缩性。

如果你在开发的场景适合我所讲到的适应场景，可以优先考虑使用发布 / 订阅模式。

思考题

你认为微信的朋友圈功能适合使用发布 / 订阅模式吗？为什么？

欢迎你把答案写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

精选留言 (49)

挖矿的小戈
2019-05-03

盆友圈适合使用pub/sub模式

原因1：消息发送方需要向多个接收方（n个可以看自己盆友圈的好友）广播消息

原因2：多消费者组（微信朋友圈数据应该不仅仅只是用于社交，可能还有其他作用吧，所以可能会有多个模块需要用到这份数据）

原因3：发送方发送消息之后是不需要接收方立即进行响应的（异步），所以用消息队列可以有效起到解耦的作用

原因4：微信朋友圈对于数据而言，满足最终一致性的

作者回复: 谢谢你的答案，观点基本我都是赞同的！

共 4 条评论 >

30

paradox
2019-05-05

1. 微博粉丝订阅机制
由于存在流量明星，针对不同情况的用户，系统可以区别对待，在线用户采用“推”模式，尽量及时更新订阅者的timeline；不在线的用户采用“拉”模式，在该用户上线后更新timeline。

2. 微信朋友圈机制
由于不太可能存在微博这种超级发布者，因此可仅采用“推”模式。

3. 两款产品的共同点
a) 发布者必须实时同步处理，因为发布成功后需要直接看到自己生产的消息。
b) 订阅者可以异步处理，只要保证系统满足最终一致性即可。

作者回复: 谢谢你的留言！总结得很到位啊！

共 2 条评论 >

21

NEVER SETTLE

2019-05-03

老师，当今有许多很好的开源MQ，响Kafka，RabbitMQ等等，但是在看许多技术分享大会，为什么许多公司还会自己开发MQ呢?

作者回复: 谢谢你的留言提问！个人觉得即便公司会开发自己的MQ，很多都是建立在这些开源项目之上的。一方面可能因为开源项目的一些属性并不能满足到自身的应用场景，所以加以改进。另一方面如果是大厂的话，可以打造出自己的一套生态圈，开发接口更加适合内部使用。

17

LJK

2019-05-03

老师好，有一点模糊的地方，请问观察者模式和发布/订阅模式的区别从消息的传递上来看就是是否通过消息队列传递数据么？另外消息队列是对接收者push消息还是接收者主动从消息队列中pull消息出来呢？谢谢老师

作者回复: 你好呀，感谢你的提问!

其实在观察者模式下，你也可以通过消息队列去传递消息。但是更本质的区别是在观察者模式下，观察者必须知道被监听者的存在。就像例子所示，观察者必须调用被监听者的接口。而发布/订阅模式下，两者是解耦的，互相都不用知道对方的存在。

消息队列中既可以使用push模式也可以使用pull模式，关键在于应用场景的考虑。例如如果消息跟新发布频繁，而下游的接收者能够处理的qps不高，那可能会更加倾向于采取pull模式。

共 2 条评论 >

13

Chn.K

2019-05-06

log offset类似tcp的滑动窗口机制，有个问题：如例子讲的10003，在第一次由于某种原因未被消费者消费，生产者又发了一次10003，然后第一次发送的10003又到了消费者，那么10003会被消费者消费两次，这种情况kafka会有什么处理机制吗?

作者回复: 谢谢你的留言提问！一般来说，这种发送机制叫做at least once delivery，这种情况需要消费者自身具备Idempotency，也就是幂等性。消费者需要自己知道哪些消息是duplicate的，从而知道怎么处理这些重复消息。

共 2 条评论 >

11

J Zhang

2019-05-05

感觉没啥内容啊

作者回复: 哈哈哈，感觉你的基本功很扎实啊，那这一讲你可以当作是复习章节，当然也可以留言和我讨论更高级的内容。我的专栏也希望能对初学者友好一点，对于初学者来说，这一讲是全新的内容呢。

12

cl

2019-05-09

如果接收方读取数据之后回应消息队列它接收的 Log offset 是 10000、10001 和 10003，那么消息队列就会认为接收方最多只接收了消息 10000 和 10001，剩下的消息 10002、10003 和 10004 则会继续发送给接收方，直到接收方回应接收了消息 10002、10003 和 10004。第一句接收的log offset有10003吗?

作者回复: 谢谢你的提问！如果我没理解错你的问题的话，你是想问接收方是否回复了10000、10001和10003这三个Log offset对吧？是的，接收的log offset有10003。因为消息队列需要接收到连续的log offset才会判定接收方接收到消息，这里因为log offset从10002断开了，所以消息队列会认为接收方从10002开始往后的消息都没有接收到。

共 2 条评论 >

11

涵

2019-05-04

老师好，对于消息队列中的消息存储有一个问题。在一个发布订阅模式下的消息队列中消息会被保存多久呢?具体情景是如果有五个接收方订阅了一个队列的消息，其中四个成功接收了，第五个总也无法接收，这条消息会一直保存在队列中，并且不断尝试发给第五个订阅者吗？还是会设置最多重复次数?另一个情景，在发布者发送消息至队列后，新增加了一个订阅者，这时新的订阅者可以收到队列里已放入的消息吗？谢谢。

作者回复: 涵你好，谢谢你的留言与提问！

第一个问题，一般不同的系统有不同的机制去确定消息队列中消息会被保存多久。像Apache Kafka中可以通过设定Retention Period来确定消息被保存多久，甚至可以设置Retention Period为Forever来永久保存。

第二个问题，只要没有超过保留期，消息会一直保存在队列并且一直尝试发送给第五个订阅者。当然订阅者也可以自己设定retry times，如果第五个订阅者告诉消息队列只需要重试一次，那这条消息就不会再发送给第五个订阅者了。如果你要保证操作一定成功，就要考虑用RPC来调用了。

第三个问题还是要看系统设计了，设计上收到与不收到都可以做到，像Apache Kafka可以通过重设Log Offset的位置去实现新的订阅者收取队列里面旧的消息。如果你使用开源项目的话，文档里应该会有说明的。

6

每天晒白牙
2019-05-03

微信朋友圈适合用消息发布 - 订阅模式
1.系统的发送方需要向大量的接收方广播消息。一个人打朋友圈，需要向有好友关系的多方发消息

2.朋友圈的数据会被多个独立的业务方使用，比如做数据分析用于投放朋友圈广告，进行用户行为分析，构建用户画像

3.系统的发送方在向接收方发送消息之后无需接收方进行实时响应。因为朋友圈这种业务应该使用pull模式，当A发朋友圈时，A的好友B C D，只有当B C D进行刷新朋友圈时，会去pull数据

4.朋友圈对数据一致性只会要求最终一致性

所以朋友圈很适合消息发布 - 订阅模式

作者回复: 谢谢你的答案！分析得很详细，基本上我都是赞同的！

5

nurh

purn
2019-05-07

谢谢老师的答复，我的问题没表述完整，kafka后面接关系数据库可以重做这些日志以落地数据，并进一步用sql处理数据，但是如果Kafka后面接流处理引擎比如Flink等，Kafka里面存的redo日志，传给流引擎，在流引擎里面如何处理这些日志？比如分别针对一个表执行了insert update delete，流引擎会去重做类似关系数据库对这些DML日志的操作，再基于结果数据做分析么，还是直接操作这些增量日志。

我没有流计算的实战经验，问题可能比较菜，请老师见谅。

作者回复: 谢谢你的提问!

我的理解是两者都会有。作为底层实现的话，流处理的数据是无限的，流引擎也不可能无止境地等待数据到来，所以一般处理这种无限流数据的话都需要定义window和trigger。所谓window就是你希望处理数据的时间窗口多大，trigger定义了什么时候你想处理时间窗口内的数据。就如你所举的例子，如果Kafka后面接的是Flink的话，Flink会将这些数据抽象成DataStream，一个时间窗口内的数据你可以把它看作就是传统数据库中已有的数据，可以做类似关系数据库的操作。Flink会监听Kafka新传来的数据，等到下一个trigger开始了，Flink又会像之前一样计算这些新传来的在一定时间窗口内的数据。

希望这能帮助你理解到你所问的问题。


4

John
2019-07-05

請問老師 如果想把Oracle的數據實時同步到其他的地方比如MongoDB 要求幾秒鐘以內的latency 有沒有辦法使用pubsub的概念做到呢 又該如何做呢 謝謝

作者回复: 谢谢你的提问！这个问题挺常见的，PubSub肯定是可以做到的。具体的做法还是看你的架构吧。如果你是own整一套系统的话，一种做法是你的service直接对这两个数据库进行dual write。如果你只是own这两个database的话，我相信Oracle应该是有一些database change notification的机制，这个时候你可以将这个data change publish到一个service中，然后再做同步。


3

川杰
2019-05-03

老师您好，个人理解，观察者模式本质上是消息推出，订阅该消息的对象获取消息并进行处

理；所以我认为，发布/订阅模式其实是一种特殊的观察者模式，或者是观察者模式的实现方式。

作者回复: 谢谢你的留言！我觉得本质上还是有所不同的，毕竟在观察者模式中观察者需要知道被监听的对象，而发布/订阅模式通过消息队列解耦了这一层关系。

共 2 条评论 >

3

潘腾
2019-07-27

如果微信朋友圈使用发布订阅模式的话，那么是不是意味着每一个用户对应一个消息队列呢，那这样的话是不是就得有好几亿消息队列?

作者回复: 谢谢你的提问！这个看你的设计了，即便真的使用发布订阅模式的话，同一个队列的消费者可以通过设置filter来只接收自己感兴趣的内容。

共 2 条评论 >

2

lwenbin
2019-05-04

我觉得朋友圈可以用pub/sub模式，但是如果只是为了满足发朋友圈让好友能看到，应该在技术实现上不太会用消息队列实现。
对于其他应用可能会有一个大的消息队列或者按照某些规则分区，存放所有用户发布的消息，供下游应用消费，比如合规检查，统计等等。
单从朋友圈发布查看如果用消息队列，代价可能太大，消息队列会过多，而且某些消息可能不一定有订阅者。朋友圈这种查看基本以pull为主。

作者回复: 谢谢你的答案！是的，朋友圈还是以pull为主，而如何做到通知客户端朋友圈有更新这一步还是可以用pub/sub模式的。

2

JohnT3e
2019-05-03

适合，1. 场景对实时性要求不高；2. 用户自己可以刷新拉取；

作者回复: 谢谢你的答案！

  2

杨建

2019-05-14

老师，想问一下，举的例子中，那个支付部门和防诈骗部门之间如果采用发布/订阅模式是不是有问题，因为在我看来防诈骗这个功能应该是和支付之间存在时序上的先后关系的，只有防诈骗小组检验通过了，才能进一步支付，而发布/订阅模式是异步的，并不能保证这个时序，不知道我理解的有没有问题

作者回复: 谢谢你的留言！哈哈，那可能是中美差异了，在北美这边的信用卡支付和盗刷alert是平行进行的，如果持卡人发现真的被盗刷了再dispute掉这笔交易。

  1

...

2019-05-06

如果接收方读取数据之后回应消息队列它接收的 Log offset 是 10000、10001 和 10003，那么消息队列就会认为接收方最多只接收了消息 10000 和 10001，剩下的消息 10002、10003 和 10004 则会继续发送给接收方，直到接收方回应接收了消息 10002、10003 和 10004。

这样10003数据是不是被重复消费了

作者回复: 谢谢你的留言！是的，10003数据会被重复消费，这种消息发送机制也被称为at-least-once delivery。作为下游的客户端，一定要保证系统具有Idempotency幂等性，来保证重复的消息能够被正确处理。

  1

purh

2019-05-05

老师好，Kappa模式如果用Kafaka做持久化，Kafaka里面存的消息和关系数据库的表应该是不同的，关系数据库可以很轻易的用SQL访问表，比如做维度聚合，Kafaka里面存的是比如Redo日志，基于这些DML日志如何如何做数据分析？

作者回复: 谢谢你的提问！Kappa架构11讲会讲到，你这是预知未来了啊哈哈。

如果我没有理解错你的问题的话，维度聚合这一步一般不会放在Kafka里面完成。即便你的Kafka消息

队列里面存放的是Redo log，也是需要把这些Redo log message发送到下游客户端中，例如像你说的，下游客户端将这些message保存在关系数据库之后再进行聚合。

1

LJK

2019-05-04

我觉得朋友圈分成两部分吧
- 自己发送的时候需要同步返回结果并且在自己的朋友圈立即可见
- 自己发送完成之后在follower的朋友圈中显示可以是发布订阅模式

作者回复: 谢谢你的答案！把发朋友圈分成两步这个考虑非常好，自己发送朋友圈的时候一般都是需要确认发送成功了，而不是fire and forget。后面follower的朋友圈通知我也是十分赞同的。

1

apollo

2019-05-03

蔡老师对基于RabbitMQ,MQTT实现RPC怎么看?

作者回复: 谢谢你的提问！我是了解过RabbitMQ可以通过Direct reply-to实现RPC，不过我不太清楚使用场景，可能是因为公司已经在使用RabbitMQ而又需要有RPC的那种request-response机制，但是技术迁移成本太大。一般来说我们都希望Pub/Sub就使用专门的framework而RPC也使用专门的RPC framework。

1