20 | 分布式通信之发布订阅：送货上门

2019-11-06 聂鹏程 来自北京

《分布式技术原理与算法解析》

你好，我是聂鹏程。今天，我来继续带你打卡分布式核心技术。

在上一篇文章中，我带你一起学习了分布式通信中的远程调用。远程调用的核心是在网络服务层封装了通信协议、序列化、传输等操作，让用户调用远程服务如同进行本地调用一样。

其实，这种方式就是通过网络服务层的封装实现了不同机器上不同进程之间的直接通信，因为是直接通信，所以通过线程阻塞的方式实现同步调用比较容易，因此通常被用于同步调用。比如，机器 1 上的进程 A 调用机器 2 上的进程 B，进程 A 被挂起，进程 B 开始执行，当进程 B 将值返回给 A 时，A 继续执行。

虽然这种方式也可以用于异步通信，但因为进程之间是直接交互的，所以当进程比较多时，会导致进程维护通信的复杂度非常高，且一个进程通信接口改变，与其通信的进程都会受到影响。

随着业务和分布式计算规模的逐渐增大和复杂化，远程调用模型有点心有余力而不足了，为此出现了专门的异步通信模式，也就是消息发布订阅模式和消息队列模式。在接下来的两篇文章中，我将与你详细讲述这两种通信模式。

话不多说，今天，我就带你一起打卡分布式通信中的发布订阅模式吧。

什么是发布订阅？

其实，发布订阅的思想在我们的生活中随处可见。

比如，学术届电子论文的订阅方式。通常，各个会议方或出版社会将学术论文发布到论文网站（或平台上，比如 ACM、知网等），然后学生或老师向论文网站订阅自己感兴趣的论文，比如分布式相关的、AI 相关的等。

当会议方或出版社将论文发布到论文网站后，论文网站会根据订阅信息，将相应的论文推送给订阅者（比如通过邮件的方式）。这里的会议方或出版社就相当于生产者，负责发布论文，学生或老师就相当于消费者，而论文网站就相当于一个消息中心。

由此可以看出，**发布订阅的三要素是生产者、消费者和消息中心，**生产者负责产生数据放到消息中心，消费者向消息中心订阅自己感兴趣的消息，当发布者推送数据到消息中心后，消息中心根据消费者订阅情况将相关数据推送给对应的订阅者。这种将数据送到消费者手里的行为，是不是和我们现在常说的“送货上门”一样呢？

发布订阅的原理及应用

这个论文订阅的例子，充分体现了发布订阅的思想。接下来，我就与你进一步分析下发布订阅的原理吧。

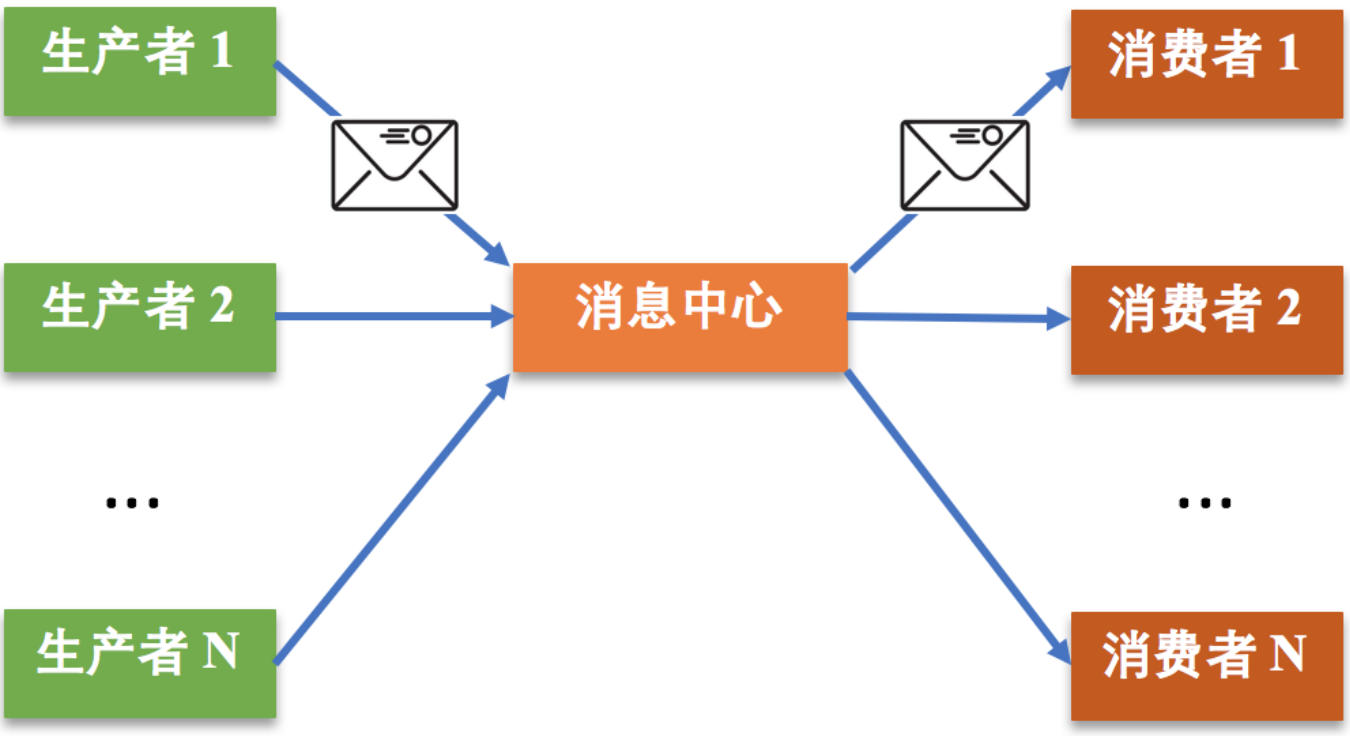
发布订阅的基本工作原理

在分布式通信领域中，消息系统一般有两种典型的模式。一种是点对点模式（P2P，Point to Point），另一种是发布订阅模式（Pub/Sub，Publish/Subscribe）。接下来，我们就一起看看这两种模式，以帮助你深入理解发布订阅模式的原理。

首先，我们一起看一下**什么是点对点模式**。

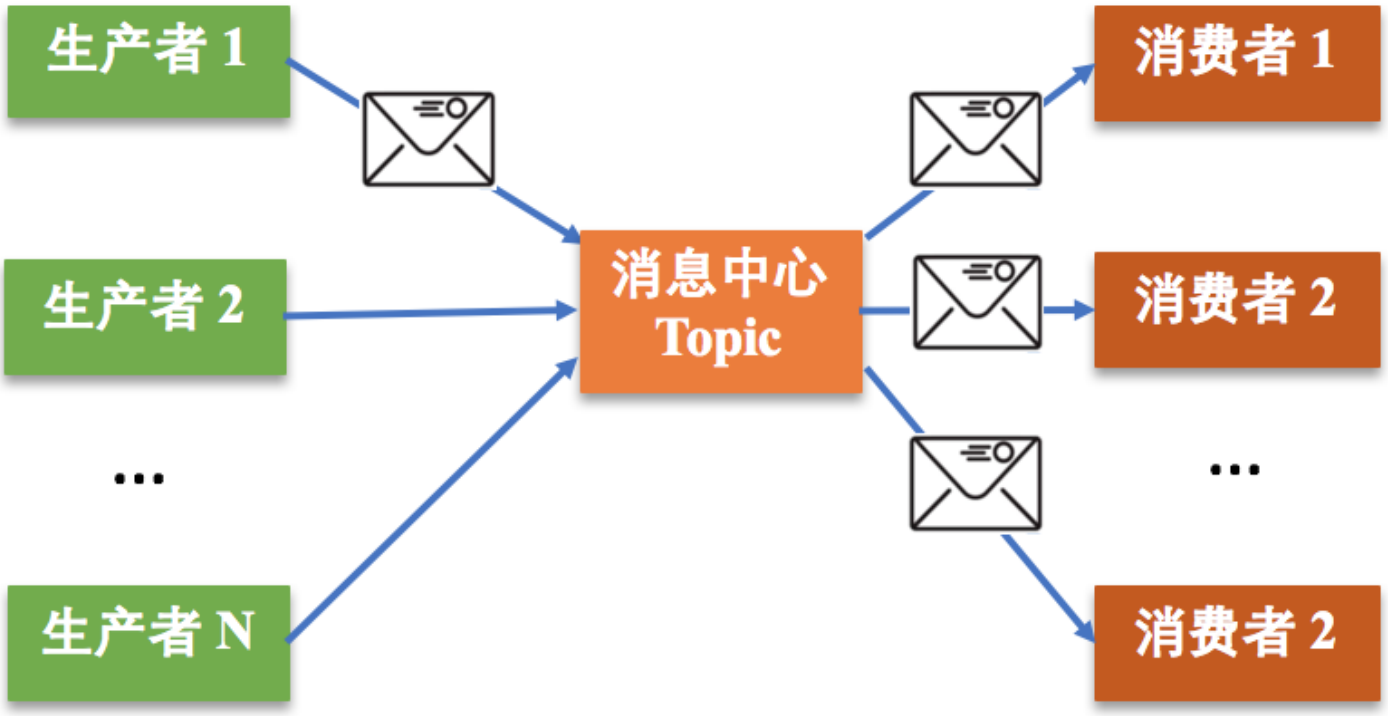
生产者将消息发送到消息中心，然后消费者从消息中心取出对应的消息进行消费。消息被消费后，消息中心不再存储该消息，因此其他消费者无法再消费该消息。也就是说，点对点模式虽然支持多个消费者，但一个消息只能被一个消费者消费，不允许重复消费。

这种模式就好比，限定了每篇论文只能被一个用户消费，比如现在有一篇分布式相关的论文，这篇论文推送给学生 A 之后，论文网站就必须将其删除或下架，也就是说其他用户无法再获取或阅读该论文了。（当然实际情况并不是这样的，这里只是为了方便你理解，我做了相应的假设。）



接下来，我们看一下**发布订阅模式**。

生产者可以发送消息到消息中心，而消息中心通常以主题（Topic）进行划分，每条消息都会有相应的主题，消息会被存储到自己所属的主题中，订阅该主题的所有消费者均可获得该消息进行消费。



比如图中假设生产者 1 发布一个 Topic 相关数据或消息，消费者 1～3 均订阅了该 Topic 消息，则该消息会推送消费者 1～3，也就是说同一个消息被 3 个消费者消费了。

这种模式就好比，不同的方向代表不同的主题，比如分布式领域代表一个主题，当会议方或出版社发布分布式相关的论文时，该论文会被存储到论文网站的分布式主题下，同时学生或老师也会根据自己感兴趣的主题进行订阅。如果学生 A 订阅了分布式主题，那么当会议方或出版社发布分布式相关的论文后，会议网站会将这些论文推送给学生 A。

与点对点模式相比，发布订阅模式中一个消息可以被多个消费者进行消费，这也是和点对点模式的本质区别。

以上就是发布订阅中的两种典型模式了。

在分布式系统中，通常会为多用户服务，而多个用户通常会关注相同类型的消息，因此发布订阅模式在分布式系统中非常常见。接下来，我再结合经典的分布式发布订阅消息系统 Kafka

的发布订阅原理及工作机制，来帮助你巩固对发布订阅的理解。

Kafka 发布订阅原理及工作机制

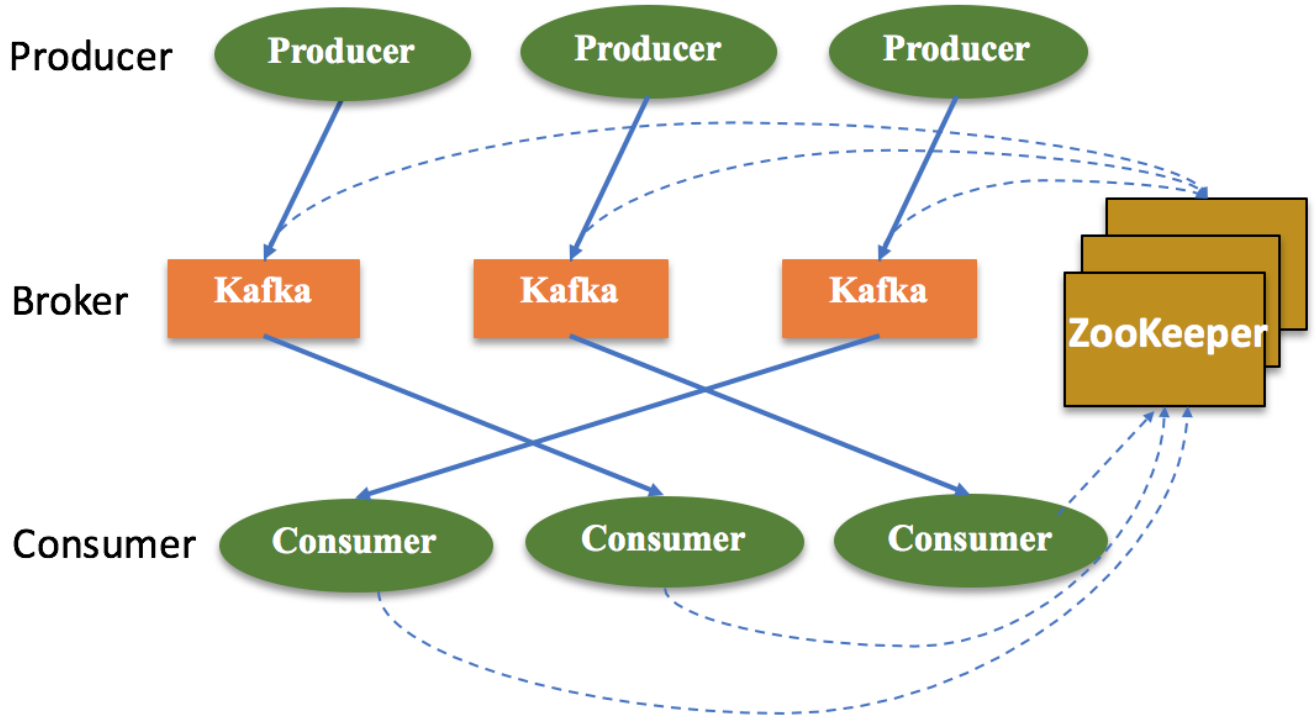
Kafka 是一种典型的发布订阅消息系统，其系统架构也是包括生产者、消费者和消息中心三部分。

生产者（Producer）负责发布消息到消息中心，比如电子论文的会议方或出版社；

消费者（Consumer）向消息中心订阅自己感兴趣的消息，获得数据后进行数据处理，比如订阅电子论文的老师或学生；

消息中心（Broker）负责存储生产者发布的消息和管理消费者订阅信息，根据消费者订阅信息，将消息推送给消费者，比如论文网站。在 Kafka 中，消息中心本质上就是一组服务器，也可以说是 Kafka 集群。

Kafka 的架构图，如下所示：



可以看到，Kafka 中除了 Producer、Broker、Consumer 之外，还有一个 ZooKeeper 集群。Zookeeper 集群用来协调和管理 Broker 和 Consumer，实现了 Broker 和 Consumer

的解耦，并为系统提供可靠性保证。

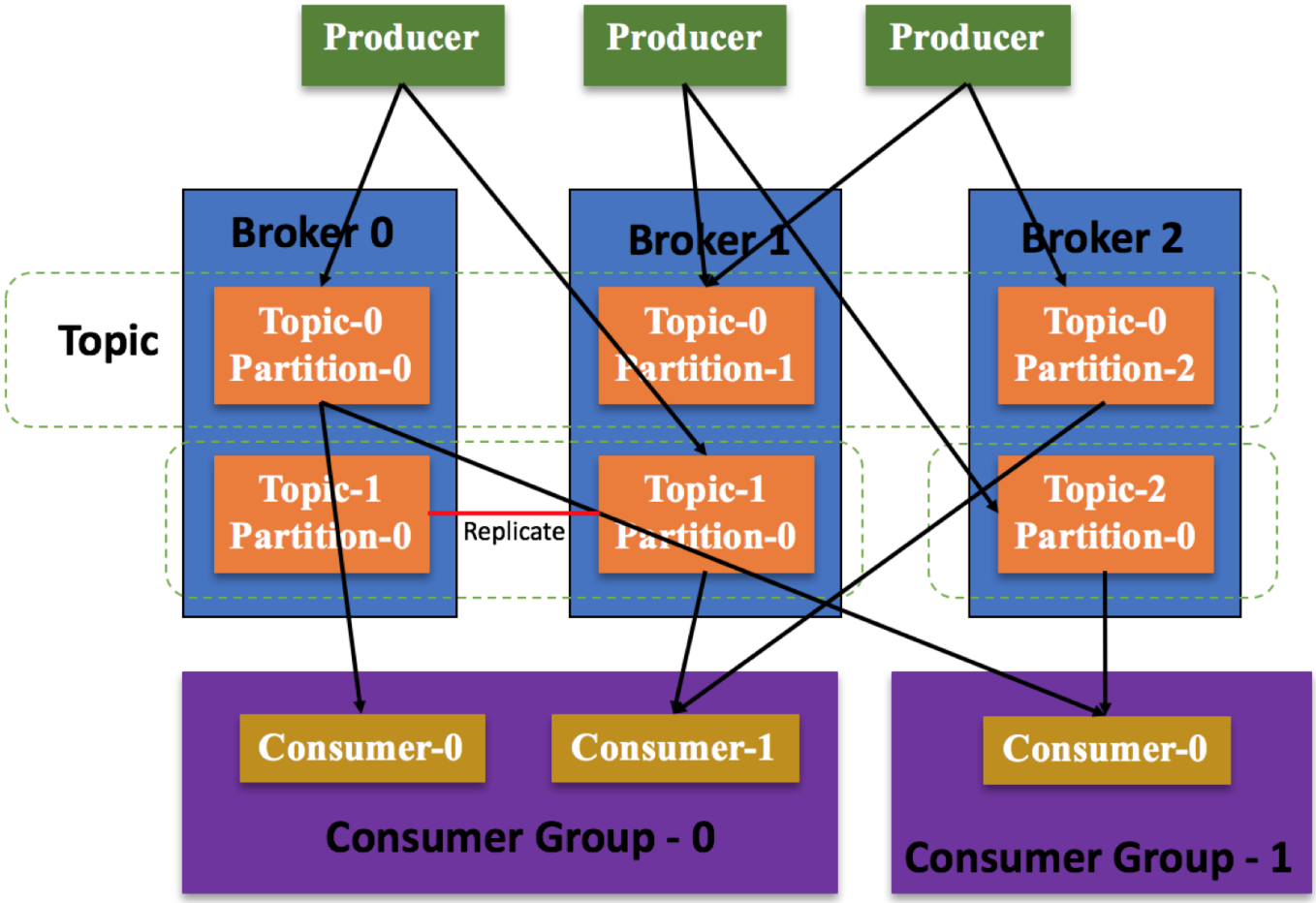
ZooKeeper 集群可以看作是一个提供了分布式服务协同能力的第三方组件，Consumer 和 Broker 启动时均会向 ZooKeeper 进行注册，由 ZooKeeper 进行统一管理和协调。

ZooKeeper 中会存储一些元数据信息，比如对于 Broker，会存储主题对应哪些分区（Partition），每个分区的存储位置等；对于 Consumer，会存储消费组（Consumer Group）中包含哪些 Consumer，每个 Consumer 会负责消费哪些分区等。

接下来，我们看看**分区和消费组的原理和作用**吧。

从上面的介绍可以看出，Broker 负责存储消息数据，Consumer 负责消费数据，Consumer 消费数据的能力会影响 Broker 数据存储是否溢出的问题。若 Consumer 消费太慢，会导致 Broker 存储溢出，Broker 就会丢弃一部分消息。

因此，Broker 和 Consumer 是 Kafka 的核心。接下来，我将带你进一步了解 Kafka 中 Broker 和 Consumer 的关键技术，如下图所示：



首先，我们看一下 Broker。

在 Kafka 中，为了解决消息存储的负载均衡和系统可靠性问题，所以引入了主题和分区的概念。其中，主题是一个逻辑概念，指的是消息类型或数据类型，就好比电子论文案例所讲的分布式是一个主题。

而分区是针对主题而言的，指的是一个主题的内容可以被划分成多个集合，分布在不同的 Broker 上，不同的 Broker 在不同的节点上。这里的集合就是分区，其中同一个分区只属于一个 Broker。

那么，**分区有什么好处呢?**

在我看来，分区的好处主要包括如下两点：

实现负载均衡，避免单个 Broker 上的负载过高。比如，Topic 0 被分为 Partiton-0、Partiton-1 和 Partiton-2 三个分区，分别分布在 Broker 0、Broker 1 和 Broker 2 上。这，就使得 Topic 0 的消息可以分布在这 3 个分区中，实现负载均衡。

实现消息的备份，从而保证系统的高可靠。比如，Topic 1 包含两个分区 Partiton-0、Partiton-1，每个分区内容一致，分别存储在 Broker 0 和 Broker 1 上，借此实现了数据备份。

接下来，我们再看看 Consumer 吧。

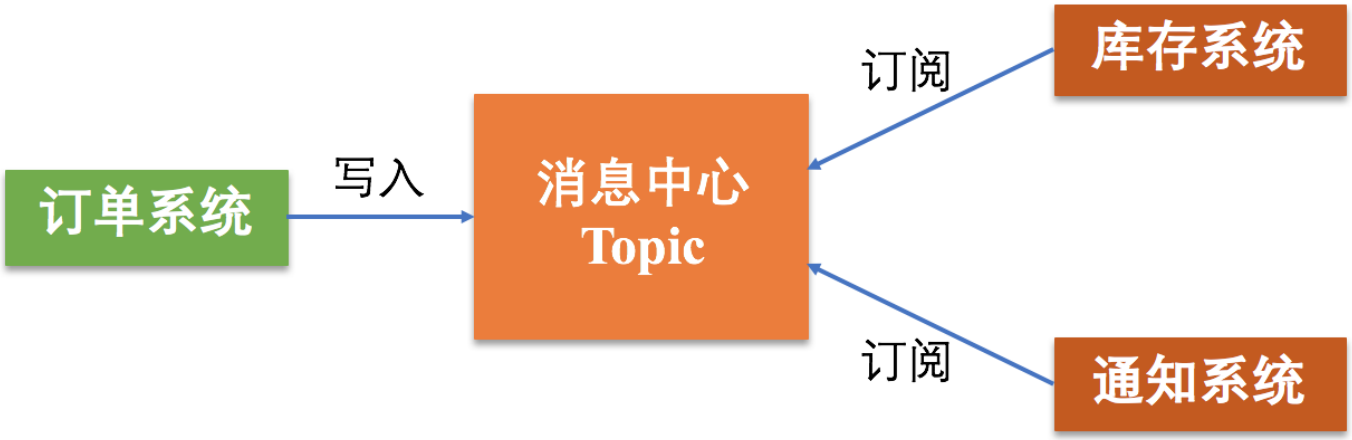
Kafka 中的消费组，指的是多个消费者的一个集合。一个消费组中的消费者共同消费主题消息，并且主题中每个消息只可以由消费组中的某一个消费者进行消费。

引入消费组的目的是什么呢？我们知道，在消息过多的情况下，单个消费者消费能力有限时，会导致消费效率过低，从而导致 Broker 存储溢出，丢弃一部分消息。Kafka 为了解决这个问题，所以引入了消费组。

这样一来，我们对发布订阅的基本工作机制就比较清楚了。接下来，我们再结合电商购物平台的例子，来看看发布订阅技术的具体应用吧。

发布订阅实践应用

假设在电商购物平台（为了方便理解，我对电商购物平台做了一定的简化）中，用户首先在订单系统下单，下单后库存系统会进行出货，通知系统则负责通知用户，整个流程可以用发布订阅的模式进行，如下图所示：



订单系统对应发布订阅模式中的生产者，消息中心有个主题专门存放下单信息，每次用户下单后，订单系统会向该主题写入数据；

库存系统和通知系统对应发布订阅模式中的消费者，它们会向消息中心订阅下单信息相关的主题；

订单系统向消息中心发布订单信息后，库存系统和通知系统都会获取到相应的下单信息，然后进行各自后续的操作，即库存系统进行出货，通知系统通过短信或邮件等方式通知用户。

接下来，我们总结下**发布订阅模式的关键特征**吧。

实现了系统解耦，易于维护。生产者 / 发布者只负责消息的发布，不需要知道订阅者 / 消费者的数量，也不需要知道订阅者 / 消费者获取消息用来做什么，而订阅者 / 消费者也不需要知道什么时候生产者 / 发布者会发布消息。

所以，生产者 / 发布者和订阅者 / 消费者互相独立，进而实现了系统解耦，每个部分可以单独维护，减少了因为生产者和消费者的耦合引入的一些相互影响。比如，如果两者耦合在一起，当生产者逻辑更改需要修改代码时，消费者部分的代码也受影响，因此每个部分单独维护降低了维护的复杂度。

实现了异步执行，避免高负载。生产者 / 发布者发布消息到消息中心，当消息超过消息中心可以存储的容量后，消息中心会丢弃掉超出的消息，这样系统就不会因为消息数量多而导致系统故障。

知识扩展：观察者模式和发布订阅模式的区别是什么？

我们还经常会听到一个概念，叫作观察者模式，也会在分布式系统中都会经常用到。那么，观察者模式和发布订阅模式的区别到底是什么呢？

首先，我们看一下观察者模式。顾名思义，观察者模式下有观察者，那么就有被观察者，两者之间的关系是什么呢？

观察者负责监控被观察者的状态变更，如果被观察者的状态发生了改变，那么观察者根据状态的变更执行相关操作。举个例子，现在进程 A 是被观察者，进程 B 和进程 C 是观察者，当进程 B 观察到进程 A 中变量 X 由 3 变为 4 时，执行 X+1 的操作；当进程 C 观察到进程 A 中变量 X 由 3 变为 4 时，执行 X-1 的操作。也就是说，观察者模式，定义了被观察者与观察者的直接交互或通信关系。

接下来，我们看一下发布订阅模式。发布订阅模式中存在发布者、订阅者和消息中心，订阅者需要向消息中心指定自己对哪些数据感兴趣，发布者推送的数据放入消息中心后，消息中心根据订阅者订阅信息推送数据。也就是说，发布者和订阅者之间引入了消息中心，实现的是间接通信。

总结来讲，观察者模式采用了直接通信，观察者和被观察者通信时延会低一些，但它们的依赖关系比较强，不管是被观察者还是观察者逻辑或接口有更改，另外一个均会受影响。而发布者和订阅者模式采用间接通信，引入了消息中心，相对比较厚重，且通信时延相对会高一点，但实现了订阅者与发布者的解耦。

总结

我首先通过论文订阅的案例，与你介绍了什么是发布订阅以及发布订阅的基本原理，然后介绍了一个经典的分布式发布订阅消息系统 Kafka，最后以一个电商购物平台的案例描述了发布订阅模式的应用场景。

我再和你总结下今天的核心知识点吧。

发布订阅就是生产者产生消息发布到消息中心，消费者订阅自己感兴趣的消息，消息中心根据消费者的订阅情况将相关消息发给对应的消费者。

Kafka 是一个经典的发布订阅消息系统，采用多分区实现了消息备份、负载均衡，并引入消费组提高了消费者的消费能力，防止 Broker 因为存储资源不够丢弃消息的情况，从而提高了 Kafka 系统的可靠性。

发布订阅模式可以使系统松耦合易于维护，也可异步执行解决高负载问题，适用于系统解耦、流量削峰等场景。

最后，我再通过一张思维导图梳理下今天的核心知识点，以帮助你理解与记忆。

发布订阅模式易于理解，与点对点模式很类似。不同的是，点对点模式中一个消息只能由一个消费者消费，而发布者订阅者模式中一个消息可以由多个消费者消费。

不同的通信模式适用于不同的分布式场景，其中发布订阅模式适合具备多个生产者、多个消费者且异步处理的场景，比如现在的视频 App，多个用户都可以通过同一款 App 看同一部电视

剧，当然这个电视剧可以是被不同的生产者发布。点对点模式由于其局限性，一般适用于需要进行点对点通信的场景，比如近场投屏等。

相信你通过本讲的学习后，可以针对不同的分布式场景选择合适的通信模式，加油！

思考题

发布订阅模式下，当发布者消息量很大时，单个订阅者的处理能力是有限的，那么能否实现订阅者负载均衡消费呢？又该如何实现呢？

我是聂鹏程，感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎你把这篇文章分享给更多的朋友一起阅读。我们下期再会！

精选留言 (25)

阿星
2019-11-13

Kafka的partition和replica搞混了吧？ 副本才是实现备份机制的吧，分区是实现了负载均衡和水平扩展

共 7 条评论 >　　27

Eternal
2019-11-09

发布订阅的时候，如果是消费者主动拉去消息，是拉模式，如果是消息中心推送消息给消费者就是推模式。

推模式：消息中心需要考虑消费者的消费能力，不能把消费者压垮了，站着消息中心的角度，这个样消息中心能控制消费的速度，也能主动调控消息消费的积压，对消息中心是有利的，对消费者是有风险的；

拉模式：由消费者自己控制自己的消费速度，不用担心自己压力；站在消费者的角度，自己控制消费速度，有多到能力干多大事，自己的风险自己掌控，这样消息中心的消息积压就会存在

风险，因为消息消费的速度自己不能控制，很容易造成消息积压，然后消息丢失，或消息中心不可用。

消息中心是两头都有风险，生产者的生产速度变化，消费者的消费速度变化都会造成消息积压风险，因为消息中心的消息存储能力，通信能力都是由限制的，消息中心是发布订阅模式中最复杂的一部分

 18

Devin
2019-11-16

文章中 “实现消息的备份，从而保证系统的高可靠。比如，Topic 1 包含两个分区 Partiton-0、Partiton-1，每个分区内容一致实现消息的备份，从而保证系统的高可靠。比如，Topic 1 包含两个分区 Partiton-0、Partiton-1，每个分区内容一致”，这个说法应当是有误的，“实现消息的备份”应该是“副本机制”，假如Topic 1 分区数是 2 ，那么总消息是 Partiton-0 和 Partiton-1 的合集，不是文中说的“每个分区内容一致”

作者回复: 这里介绍的是分区的功能或作用，不同分区存储不同数据子集，以及不同分区存储相同数据作为数据备份都可以看成是分区的作用

共 3 条评论 > 3

Eternal
2019-11-09

Kafka订阅的时候，客户端不需要考虑订阅哪个分区
当一个消费者组的消费者数量小于分区数量的时候是消费能力不足，该组中的一个消费者会超负载消费，存在挂掉风险
当一个消费者组的消费者数量大于分区数量的时候是分区的数量不足，该组中的一个消费者会存在空负载的情况，消费资源浪费

因此，一般一个消费者组的消费者数量和该组订阅的topic的分区数量一致，或者是成倍数。

成倍数是：
如果一个topic有3个分区，那么消费者组的消费者可以是，3个，6个，9个，这样一个组中的每个消费者就会均衡
如果一个消费者组的消费者数量是3个，那么他们订阅的topic的分区数量可以是是，3个，6个，9个，这样一个组中每个消费者消费的分区会均衡

如果一个消费者组中的消费者数量和组订阅的topic中的分区数量不成倍数，会存在问题：
当消费者挂掉，或者新的消费者加入组的时候，当分区数量新增或减少的时候，都会触发重平衡，即消费者和分区数量映射的重平衡
重平衡如果不均衡就会导致消费者负载过高，消费慢，也会造成topic消息积压，所有关键的问题说就是要使消费者和分区怎么均衡映射

4

chen
2020-05-08

其实之所以能实现负载均衡，消费组也是个重要的因素，老师没有说啊

> 作者回复: 这个问题非常好，确实消费组在一定程度上能实现负载均衡，本章留下的思考问题就是想大家在思考后在留言区能一起讨论起来。

2

钱
2020-02-18

阅过留痕
分布式系统三剑客：RPC、MQ、REDIS
上节是 是RPC这节是MQ，RPC核心是系统解藕，远程调度简易化，有返回值。MQ的核心也是解藕，而且更加的彻底，另外，就是削峰填谷。
关于RPC和MQ都需要一整个专栏来介绍，越来越感觉老师这里，有些科普的感受。不过比较集中和系统的介绍了一下，这方面的内容也挺好，定位问题吧！
为了加深理解，来个比喻：
点对点——类似一个母鸡在鸡窝里下了一个蛋，不论家里的谁，拿走了，就是拿走了就没有了
发布订阅——类似农村电线杆子上贴的一个广告，整村的人都可以看，都看到了，他的效果也就没有了
有关MQ的高频问题有如下几个：
消息少发？多发？怎么处理？顺序消息怎么实现？大量消息积压，且需要及时消费，怎么处理？

2

Jackey
2019-11-06

Kafka的消费者组有点像思考题提到的订阅者负载均衡，不过应该是分区数大于消费者数才会进行多个消费者消费吧。所以是否可以考虑订阅时不仅仅是指定主题，而是需要指定到具体的

分区?

2

趁早

2021-09-04

真的是太多的介绍性内容了，感觉干活真的好少哈

1

88591

2020-04-03

增加消费者数量，可以提高处理速度

作者回复: 一定程度上是可以的

1

修愿三秋

2019-11-20

Broker的数据存储是否溢出和Consumer 消费数据的能力没有什么关系的，俩组件是独立的，互不影响

共 2 条评论>

1

leslie

2019-11-06

不知道为何老师选择kafka：Coding的易读性和易操作以及排错性考虑还是？Kafka其特性是收之后打包，解包是在Cilent端。

老师今天的问题"单个订阅者的处理能力是有限的，那么能否实现订阅者负载均衡消费呢？又该如何实现呢？"其实老师今天的问题最合适的MQ应当是rocketMQ:阿里共享给开源社区的这款产品，Kafka在高并发的性能上其实还是相对偏弱。

可能不同的MQ在分布式环节中适用的场景应当是不同的，这就像DB这块-RMDB和NOSQL DB承担的是软件过程中不同的场景而已；kafka和rocket我都学过都简单研究过，自己后续准备用在不同场景下。

1

另一条路

2022-03-18

对消息中间件的认识更加深刻了，从老师通俗易懂的介绍中学到了很多

Zzz
2020-11-15

“Broker 负责存储消息数据，Consumer 负责消费数据，Consumer 消费数据的能力会影响 Broker 数据存储是否溢出的问题。若 Consumer 消费太慢，会导致 Broker 存储溢出，Broker 就会丢弃一部分消息”这里是不是写错了，对于kafka来说，消费者消费数据会影响broker存储数据？意思是我一个消费者消费过了，其他消费者没有消费，这些数据可以删除？kafka可以暂存数据，数据过期可以删除，但是没说消费太慢，也可以删除吧，还是说有这个参数配置，支持这种操作？


极客雷
2020-03-28

图做的挺漂亮


陈启明
2020-01-09

consumer在监听topic的时候，需要指定topic的partition么？


Jaime
2019-12-16

消息模式是没有返回值，如果业务是需要返回值的，是不是只能使用rpc方式来实现？多谢老师了


易儿易
2019-11-29

Topic模式消息中心存储的消息何时被销毁呢？是消费量=（通过zk确定保持连接的）订阅者数量后自动销毁吗？如果订阅者出现问题消息中心也会超期自动销毁吧？


new life
2019-11-23

思考题: 负载均衡 不能解决消息堆积的问题 多个节点部署相同的代码 都在一个消费者组下

只会有一个节点消费到消息
解决方法: 创建多个消费者组 通过消费方的业务代码 来控制


_____Harvey凝枫✿...
2019-11-20

消费组与Broker存储溢出的关系是什么?

一个消费组中的消费者共同消费主题消息，每个消息只由组内的某个消费者消费;
那么单个消费者消费能力有限时，难道强制推送给组内的其它消费者？这样不合理吧;
如果说时不强制推送给组内的其它消费者，那么这个和Broke溢出有什么关系?


旷野星空
2019-11-10

思考题，使用消费者组，分布在不同的节点上