Apache Kafka

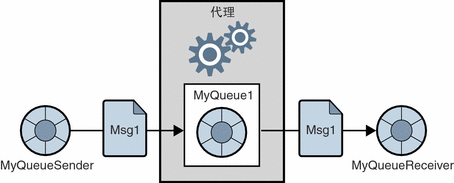
在大数据系统中，整个大数据是由各个子系统组成，数据需要在各个子系统中高性能、低延迟的不停流转，这就需要消息系统将数据从一个应用程序传输到另一个应用程序，从而使应用程序可以专注于数据，而不担心如何共享它。

# 1.消息系统

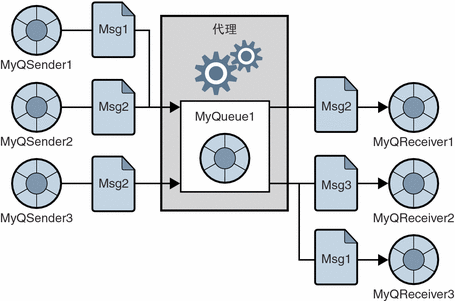
消息系统使各组件和应用可以通过生成并使用消息进行通信，根据通信的模式可分为：点对点消息传送和发布/订阅消息传送。

## 1.1 点对点消息传送

在点对点系统中，消息的生成方被称为发送者，而使用方被称为接受者，两者通过队列来交换消息：发送者生成队列中的消息；接受者使用队列中的消息



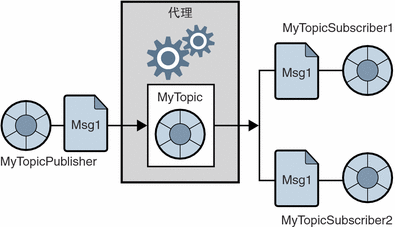
消息被保留在队列中，一个或者多个消息者可以消耗队列中的消息，但是特定消息只能由最多一个消费者消费，一旦消息者读取队列中的消息，它就从队列中消息。点对点的消息系统可以稍微复杂些，如下图所示：



在上图所示的消息系统中，多个消息发送方可向同一个队列发送消息，其共享连接或者使用不同连接。多个接受者可以使用该队列中的多条消息，但是每条消息只能由一个接受者使用，接受者可以共享连接或者不同连接。发送者和接受者之间不存在时间上的相关性，客户端发送一条消息后，无论是接受者是否在运行，都能获取该消息。

## 1.2 订阅-发布消息系统

在订阅-发布消息系统中，消息被保留在主题中，与点对点不同的是消费者可以订阅一个或者多个主题并使用该主题中所有消息。在发布-订阅系统中，消息生产者成为发布者，消息使用者称为订阅者。发布/订阅的消息传送操作如下：

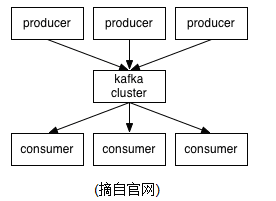


MyTopicPublisher向目标Topic发布消息，订阅者MyTopicSubscriber从Topic中接收Msg副本。

在上图中虽然发布、订阅模型不要求多个订阅用户，但是图中显示两个订阅用户来强调通过此模型可以广播消息，一个主题的所有订阅用户均可获得发布到该主题的任何消息的副本。

## 1.3 Kafka 分布式发布-订阅消息系统

Kafka是一个分布式发布-订阅消息系统，在系统中具有一个强大的队列来处理大量数据，并能够将消息从一个端点传递到另一个端点。Kafka提供了类似JMS的特性，但是在设计实现上完全不同，不符合JMS规范，系统架构图：

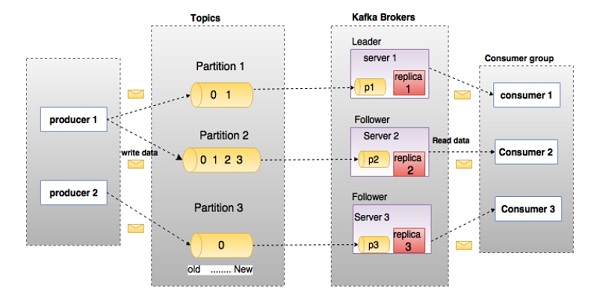


Kafka对消息保存是根据Topic进行归类，发送消息者称为Produer，消息接受者称为Consumer，此外Kafka集群有多个Kafka实例组成，每个实例（Server）都是Broker，Kafka中的组件都依赖Zookeeper保存Meta信息来保证系统可用性。

Kafka和JMS(ActiveMQ)不同的是即使消息被消费，不会立即删除，日志文件将根据配置要求，保留一定时间之后删除。

# 2. Kafka系统架构及概念

Kafka的拓扑结构如下所示：



1. Producers

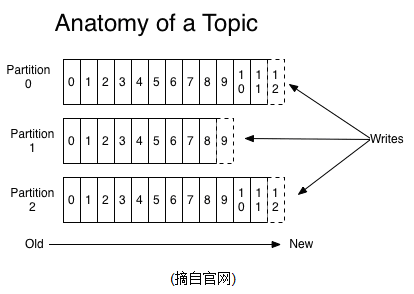
Producer将消息发布到指定的Topic钟，同时Producer也能决定将此消息属于哪个Partition，比如基于”round-robin”方式或者一些其他算法

1. Consumers

在Kafka中，每个Consumer属于一个Consumer Group，订阅的Topic消息发给Group中每个Consumer消费。

1. Topics

topic是一类消息，每个Topic被分成多个Partition，每个Partition在存储层面是append log文件，任何发布到该partition的消息都会追加到log文件的尾部，每条消息在文件中的位置称为offset(偏移量)，offset为一个long型数字，唯一标记一条消息。Kafka并没有提供其他额外的索引机制来存储offset，因此不允许对消息进行随机读写



1. Partition

在Kafka中，每个Topic可以分成多个Partition，在创建时指定，每个Partition存储一部分Message。Partition以文件的形式存储在文件系统中，比如创建一个名为page\_visit的topic，有5个partition，那么在kafka的数据目录中有5个目录：page\_visit-0,page\_visit-1…，其命名规则为<topic-name>-<partition\_id>，里面存储的分别是5个partition的数据。

1. Distribution&&Replicated

Topic的Partitions被分布在Kafka集群中的多个Servr上，每个Server(Kafka实例)负责partitions中消息的读写操作。Partitions需要备份Replicas，将其放置在多台集群中以提高可用性。

基于Replicated的方案，需要将多个备份进行调度，每个Partition中都有一个Server为Leader，负责所有的读写操作，如果Leader是喜爱哦，那么其他follower成为新的leader。Follower只是简单的和Leader进行同步消息。Leader的Server承载了全部的请求压力，因此从集群的整体性能考虑，Kafka会通过Distribution机制将leader均衡的分散到每个实例上，来确保集群的性能。

1. Guarantees

生产者发送到一个特定Topic分区上的消息按照发送顺序依次写入，消费者收到的消息也是该属性，如果一个Topic配置了复制因子为N，那么允许N-1服务器宕机而不丢失任何已经增加的消息。

https://blog.csdn.net/boonya/article/details/54783294

https://blog.csdn.net/ronmy/article/details/60467147

https://www.w3cschool.cn/apache\_kafka/apache\_kafka\_quick\_guide.html

https://docs.oracle.com/cd/E19148-01/820-0533/aerbk/index.html