软件复用第二次讨论课方案

1252874 陈薇伊

参考系统：

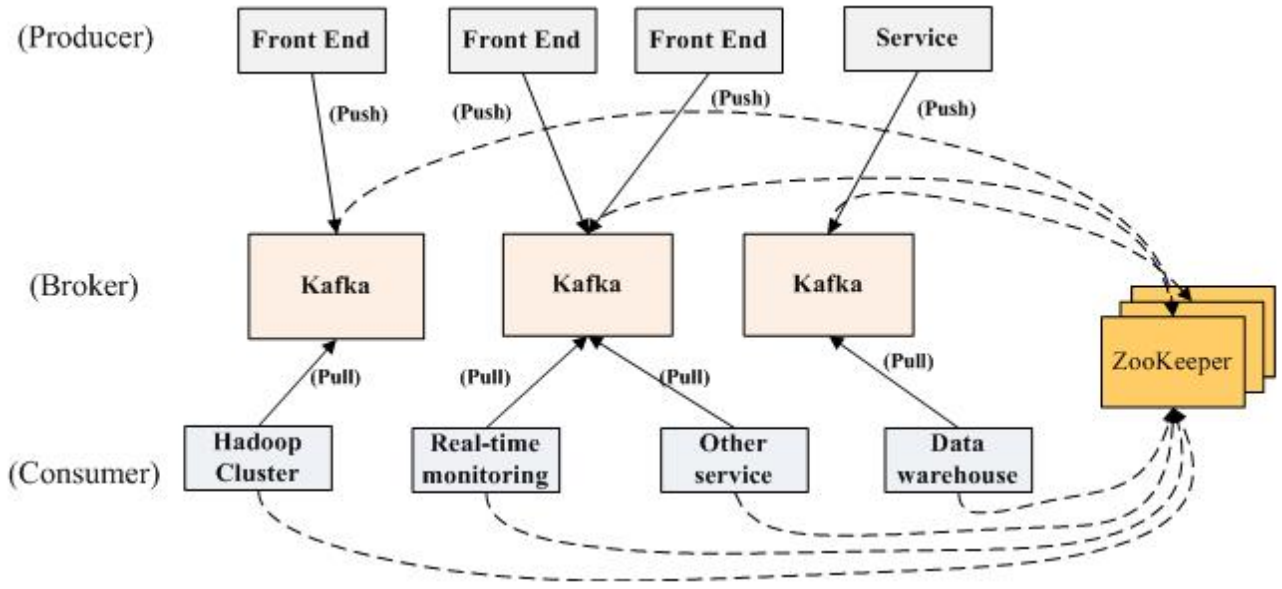
Apache Kafka是分布式发布-订阅消息系统。它最初由LinkedIn公司开发，之后成为Apache项目的一部分。Kafka是一种快速、可扩展的、设计内在就是分布式的，分区的和可复制的提交日志服务。

Apache Kafka与传统消息系统相比，有以下不同：

- 它被设计为一个分布式系统，易于向外扩展；
- 它同时为发布和订阅提供高吞吐量；
- 它支持多订阅者，当失败时能自动平衡消费者；
- 它将消息持久化到磁盘，因此可用于批量消费，例如ETL，以及实时应用程序。

框架：

一个典型的kafka集群中包含若干producer（可以是web前端产生的page view，或者是服务器日志，系统CPU、memory等），若干broker（Kafka支持水平扩展，一般broker数量越多，集群吞吐率越高），若干consumer group，以及一个Zookeeper集群。Kafka通过Zookeeper管理集群配置，选举leader，以及在consumer group发生变化时进行rebalance。producer使用push模式将消息发布到broker，consumer使用pull模式从broker订阅并消费消息。



消息发送的流程：

一、基本概念：

1、Topic：特指Kafka处理的消息源（feeds of messages）的不同分类。

2、Partition：Topic物理上的分组，一个topic可以分为多个partition，每个partition是一个有序的队列。partition中的每条消息都会被分配一个有序的id（offset）。

3、Message：消息，是通信的基本单位，每个producer可以向一个topic（主题）发布一些消息。

4、Producers：消息和数据生产者，向Kafka的一个topic发布消息的过程叫做producers。

5、Consumers：消息和数据消费者，订阅topics并处理其发布的消息的过程叫做consumers。

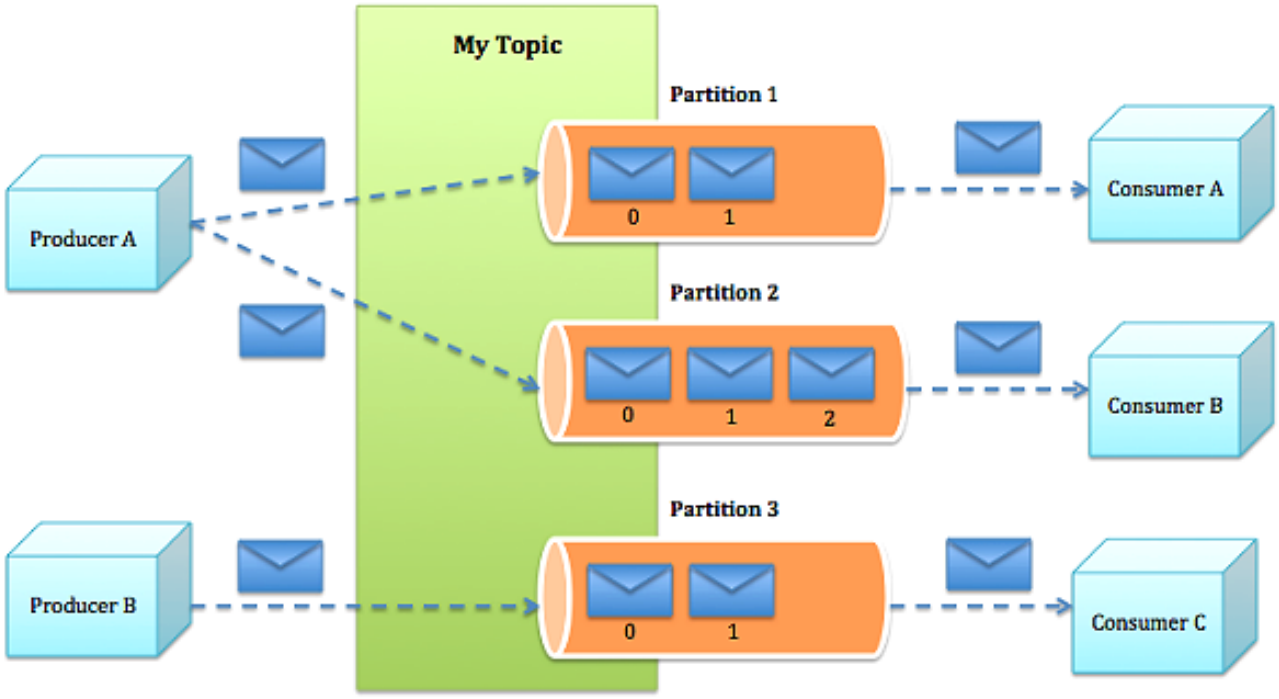
6、Broker：缓存代理，Kafa集群中的一台或多台服务器统称为broker。

二、流程

1、Producer根据指定的partition方法（round-robin、hash等），将消息发布到指定topic的partition里面

2、kafka集群接收到Producer发过来的消息后，将其持久化到硬盘，并保留消息指定时长（可配置），而不关注消息是否被消费。

3、Consumer从kafka集群pull数据，并控制获取消息的offset

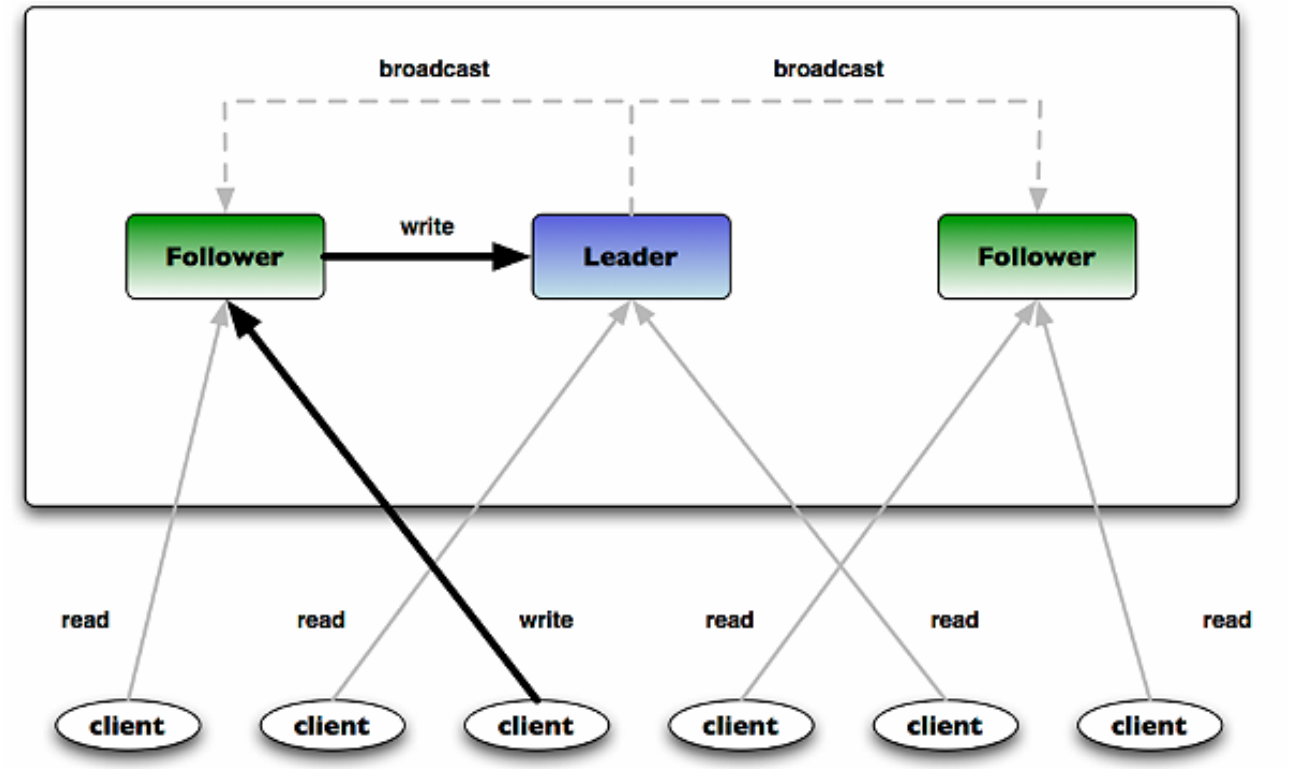


分布式：

Kayka的整体架构非常简单，是显式分布式架构 ，即所有的producer、broker和consumer都会有多个，均为分布式的。它具有以下特性：快速持久化，可以在O(1)的系统开销下进行消息持久化；高吞吐，在一台普通的服务器上既可以达到10W/s的吞吐速率

数据储存：

Producer，consumer实现Kafka注册的接口，数据从producer发送到broker，broker承担一个中间缓存和分发的作用。broker分发注册到系统中的consumer。broker的作用类似于缓存，即活跃的数据和离线处理系统之间的缓存。数据磁盘持久，即消息不在内存中cache，直接写入到磁盘，充分利用磁盘的顺序读写性能。所有broker和consumer都会在zookeeper中进行注册，ZooKeeper作为一个分布式的、分层级的文件系统会保存他们的一些元数据信息。ZooKeeper运行多个ZooKeeper服务器，称为Ensemble，以获得高可用性。每个服务器都持有分布式文件系统的内存复本，为客户端的读取请求提供服务。ZooKeeper Ensemble架构如下图所示：



负载均衡：

Producer发送消息到broker时，会根据Paritition机制选择将其存储到哪一个Partition。如果Partition机制设置合理，所有消息可以均匀分布到不同的Partition里，这样就实现了负载均衡。

1、producer根据用户指定的算法，将消息发送到指定的partition

2、存在多个partiiton，每个partition有自己的replica，每个replica分布在不同的Broker节点上

3、多个partition需要选取出lead partition，lead partition负责读写，并由zookeeper负责fail over

4、通过zookeeper管理broker与consumer的动态加入与离开

消息队列，ID分配：

kafka以topic来进行消息管理，每个topic包含多个part（ition），每个part对应一个逻辑log，有多个segment组成。每个segment中存储多条消息（见下图），消息id由其逻辑位置决定，即从消息id可直接定位到消息的存储位置，避免id到位置的额外映射。每个part在内存中对应一个index，记录每个segment中的第一条消息偏移。发布者发到某个topic的消息会被均匀的分布到多个part上（随机或根据用户指定的回调函数进行分布），broker收到发布消息往对应part的最后一个segment上添加该消息，当某个segment上的消息条数达到配置值或消息发布时间超过阈值时，segment上的消息会被flush到磁盘，只有flush到磁盘上的消息订阅者才能订阅到，segment达到一定的大小后将不会再往该segment写数据，broker会创建新的segment。

通信：

所有broker和consumer都会在zookeeper中进行注册，且zookeeper会保存他们的一些元数据信息。如果某个broker和consumer发生了变化，所有其他的broker和consumer都会得到通知，保证通信可靠性。

协议：

客户端和服务器端的通信，是基于简单，高性能，且与编程语言无关的TCP协议。

安全：

安全方面，主要有以下方面设计：

1、客户端连接borker使用SSL或SASL进行验证

2、borker连接ZooKeeper进行权限管理

3、数据传输进行加密（需要考虑性能方面的影响）

4、客户端读、写操作可以进行授权管理

5、可以对外部的可插拔模块的进行授权管理