Kafka 简介

Kafka 是最初由 Linkedin 公司开发，是一个分布式、支持分区的（partition）、多副本的（replica），基于 zookeeper 协调的分布式消息系统。

它的最大的特性就是可以实时的处理大量数据以满足各种需求场景：比如基于 hadoop 的批处理系统、低延迟的实时系统、storm/Spark 流式处理引擎，web/nginx 日志、访问日志，消息服务等等，用 scala 语言编写，Linkedin 于 2010 年贡献给了 Apache 基金会并成为顶级开源项目。

它是一种高吞吐量的分布式发布订阅消息系统，可以处理大规模网站中所有动作流数据：网页浏览、搜索和其他用户的行为，这些数据通常是由于吞吐量的要求，通过处理日志和日志聚合来解决。

Kafka 特点

1. 高吞吐量、低延迟：kafka 每秒可以处理几十万条消息，它的延迟最低只有几毫秒，每个 topic 可以分多个 partition, consumer group 对 partition 进行 consume 操作。
2. 可扩展性：kafka 集群支持热扩展
3. 持久性、可靠性：消息被持久化到本地磁盘，并且支持数据备份防止数据丢失
4. 容错性：允许集群中节点失败（若副本数量为 n,则允许 n-1 个节点失败）
5. 高并发：支持数千个客户端同时读写。

Kafka 原理

通常来讲，消息模型可以分为两种：队列和发布-订阅式。队列的处理方式是一组消费者从服

务器读取消息，一条消息由其中的一个消费者来处理。在发布-订阅模型中，消息被广播给所有的消费者，接收到消息的消费者都可以处理此消息。Kafka 为这两种模型提供了单一的消费者抽象模型： 消费者组(consumer group)，消费者用一个消费者组名标记自己。

一个发布在 Topic 上消息被分发给此消费者组中的一个消费者。假如所有的消费者都在一个组中，那么这就变成了 queue 模型。假如所有的消费者都在不同的组中，那么就完全变成了发布-订阅模型。更通用的， 我们可以创建一些消费者组作为逻辑上的订阅者。每个组包含数目不等的消费者，一个组内多个消费者可以用来扩展性能和容错。

并且，kafka 能够保证生产者发送到一个特定的 Topic 的分区上，消息将会按照它们发送的顺序依次加入，也就是说，如果一个消息 M1 和 M2 使用相同的 producer 发送，M1 先发送，那么 M1 将比 M2 的 offset 低，并且优先的出现在日志中。消费者收到的消息也是此顺序。如果一个 Topic 配置了复制因子（replication facto）为 N,那么可以允许 N-1 服务器宕机而不丢失任何已经提交（committed）的消息。此特性说明 kafka 有比传统的消息系统更强的顺序保证。

但是，相同的消费者组中不能有比分区更多的消费者，否则多出的消费者一直处于空等待，不会收到消息。

Kafka 设计思想

Kakfa Broker 集群受 Zookeeper 管理，所有的 Kafka Broker 节点一起去 Zookeeper 上注册一个临时节点，因为只有一个 Kafka Broker 会注册成功，其他的都会失败，所以这个成功在 Zookeeper 上注册临时节点的这个 Kafka Broker 会成为 Kafka Broker Controller，其他的 Kafka broker 叫 Kafka Broker follower，这个过程叫 Controller 在 ZooKeeper 注册 Watch。

这个 Controller 会监听其他的 Kafka Broker 的所有信息，如果这个 kafka broker controller 宕机了，在 zookeeper 上面的那个临时节点就会消失，此时所有的 kafka broker 又会一起去

Zookeeper 上注册一个临时节点，成功在 Zookeeper 上注册临时节点的这个 Kafka Broker 会成为一个新的 Kafka Broker Controller，其他的会成为 follower。

例如：一旦有一个 broker 宕机了，这个 kafka broker controller 会读取该宕机 broker 上所有的 partition 在 zookeeper 上的状态，并选取 ISR 列表中的一个 replica 作为 partition leader；如果 ISR 列表中的 replica 全挂，选一个幸存的 replica 作为 leader; 如果该 partition 的所有的 replica 都宕机了，则将新的 leader 设置为-1，等待恢复，等待 ISR 中的任一个 Replica“活”过来，并且选它作为 Leader；或选择第一个“活”过来的 Replica（不一定是 ISR 中的）作为 Leader。

这个 broker 宕机的事情，kafka controller 也会通知 zookeeper，zookeeper 会通知其他的 kafka broker。