Kafka 学习笔记

Kafka offset机制

概要

- offset 原理
- offset 提交方式
- offset 存储
- 总结

offset 原理

- 原理
  在kafka的消费者中，有一个非常关键的机制，那就是offset机制。它使得Kafka在消费的过程中即使挂了或者引发再均衡问题重新分配Partation，当下次重新恢复消费时仍然可以知道从哪里开始消费。它好比看一本书中的书签标记，每次通过书签标记(offset)就能快速找到该从哪里开始看(消费)。
- 如何消费
  consumer 往一个叫做 `_consumer_offset` 的特殊主题发送消息，消息里面包含每个分区的偏移量。如果消费者一直处于运行状态，那么偏移量就没有什么用处。不过，如果消费者发生崩溃或者有新的消费者加入群组，就会触发rebanlance(再均衡)，完成在均衡之后，每个消费者可能分配到新的分区，而不是之前处理的那个，为了能够继续之前的工作，消费者需要读取每个分区最后一次提交的偏移量，然后从偏移量指定的地方继续处理。
  - Q1 如果提交的偏移量小于客户端处理的最后一个消息的offset，则两者之间的数据就会被重复消费。
  - Q2 如果提交的偏移量大于客户端处理的最后一个消息的offset,则两者职期间的数据就会丢失。

  所以，偏移量的提交对客户端有很大的影响。

offset 提交方式

Kafka对于offset的处理有两种提交方式：(1) 自动提交(默认的提交方式) (2) 手动提交(可以灵活地控制offset)

- 自动提交偏移量
  最简单的方式就是consumer自动提交offset，如果 `enable.auto.commit =true` ，那么每过5s,consumer会自动把poll()方法接收到的最大offset提交上去。提交时间间隔由 `auto.commit.interval.ms` 控制，默认是 5s.与消费者里其他的东西一样，自动提交也是在轮询里进行的。consumer每次在进行查询的时候回检查是否该提交偏移量了，如果是，那么就会提交从上一次轮询返回的偏移量。

  ```
  group_id=test_group_1

  partation=hash(“test_group_1”)%50=28
  ```

  不过， 在使用这种渐变的方式之前，需要知道它将会带来怎样的后果。
  假设我们使用默认的5s提交时间间隔，在最近一次提交之后的3是，发生了在均衡，在均衡之后，消费者从最后一次提交的offset的位置开始读取消息，这个时候offset已经落后了3s,所以在这3s到达的消息会被重复处理。可以通过修改提交时间来频繁的提交offset，减少可能出现重复消息的时间窗，不过这种情况是无法完全避免的。
- 手动提交偏移量
  Kafka自动提交offset的不灵活性和不精确性(只能是按指定频率的提交)，Kafka提供了手动提交offset策略。手动提交能对偏移量更加灵活精准地控制，以保证消息不被重复消费以及消息不被丢失。
  对于手动提交offset主要有3种方式：1.同步提交 2.异步提交 3.异步+同步 组合的方式提交
  - 同步提交
    处理完当前批次的消息，在轮询更多的消息之前，调用 `commitSync` 方法提交当前批次最新的offset只要没有发生不可恢复的错误， `commitSync()` 会一直尝试直至提交成功，如果提交失败，我们也只能把异常记录到日志里。
  - 异步提交
    提交一个 `offset` ，然后继续做其他事情，如果提交失败，错误信息和偏移量会被记录下来。 `commitAsync` 和 `commitSync` 不同在于，

它不会一直重试，是因为有可能在它收到服务器响应之前，可能有一个更大的 offset 已经提交成功。另外 commitAsync 支持回调。

- 异步+同步 组合的方式提交偏移量
  通过对消费者进行异步批次提交并且在关闭时同步提交的方式，这样即使上一次的异步提交失败，通过同步提交还能够进行补救，同步会一直重试，直到提交成功。

offset 存储

- 首先来说说消费者如果是根据javaapi来消费，也就是 kafka.javaapi.consumer.ConsumerConnector ，我们会配置参数 zookeeper.connect 来消费。这种情况下，消费者的offset会更新到zookeeper的 consumers/{group}/offsets/{topic}/{partition} 目录下，例如：

```
[zk: localhost(CONNECTED) 0] get /kafka/consumers/zoo-consumer-group/offsets/my-topic/0
5662
cZxid = 0x20006d28a
ctime = Wed Apr 12 18:20:51 CST 2017
mZxid = 0x30132b0ed
mtime = Tue Aug 22 18:53:22 CST 2017
pZxid = 0x20006d28a
cversion = 0
dataVersion = 5758
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 4
numChildren = 0
```

- 如果是根据kafka默认的api来消费，即 org.apache.kafka.clients.consumer.KafkaConsumer ，我们会配置参数 bootstrap.servers 来消费。而其消费者的offset会更新到一个kafka自带的topic __consumer_offsets 下面，查看当前group的消费进度，则要依靠kafka自带的工具 kafka-consumer-offset-checker ，例如：

```
[root@localhost data]# kafka-consumer-offset-checker --zookeeper localhost :2181/kafka --group test-consumer-group  --topic stable-
[2017-08-22 19:24:24,222] WARN WARNING: ConsumerOffsetChecker is deprecated and will be dropped in releases following 0.9.0. Use Co
Group           Topic                          Pid Offset          logSize         Lag             Owner
test-consumer-group stable-test                0   601808          601808          0               none
test-consumer-group stable-test                1   602826          602828          2               none
test-consumer-group stable-test                2   602136          602136          0               none
```

- kafka消费者在会保存其消费的进度，也就是offset，存储的位置根据选用的kafka api不同而不同。

方式	方式来源	存储位置	优点	缺点
自动提交	kafka	kafka	Spark应用从kafka中读取数据之后就自动提交	不是数据处理之后提交，无法控制
异步提交	kafka	kafka	Spark应用从kafka中读取数据并处理好之后提交offset	如果kafka服务出错，存储offset的topic可能会受影响
checkpoint	spark streaming	hdfs	Spark Streaming的checkpoint是最简单的存储方式，并且在Spark框架中很容易实现;存储在HDFS上，能够从失败中恢复数据。	如果对Spark Streaming应用进行升级的话，不能checkpoint的数据没法使用
hbase存储	程序开发	hbase	持久化offsets;能将多个Spark Streaming应用和多个Kafka topic存放在一张表格中	开发程序;设计表结构;读取offset时可能会有延迟
zookeeper存储	程序开发	zookeeper：		

/consumers/[groupId]
/offsets/topic/[partitionId]|持久化存储offest;路径明确;|消费者需要频繁的去与 Zookeeper 进行交互;如果期间 Zookeeper 集群发生变化，那

Kafka 集群的吞吐量也跟着受影响|
|redis存储|程序开发|redis|持久化存储offest;易获取|对历史offset的存储不友好|