30 | 怎么重设消费者组位移？

2019-08-10 胡夕

你好，我是胡夕。今天我要跟你分享的主题是：如何重设消费者组位移。

为什么要重设消费者组位移？

我们知道，Kafka和传统的消息引擎在设计上是有很大区别的，其中一个比较显著的区别就是，Kafka的消费者读取消息是可以重演的（replayable）。

像RabbitMQ或ActiveMQ这样的传统消息中间件，它们处理和响应消息的方式是破坏性的（destructive），即一旦消息被成功处理，就会被从Broker上删除。

反观Kafka，由于它是基于日志结构（log-based）的消息引擎，消费者在消费消息时，仅仅是从磁盘文件上读取数据而已，是只读的操作，因此消费者不会删除消息数据。同时，由于位移数据是由消费者控制的，因此它能够很容易地修改位移的值，实现重复消费历史数据的功能。

对了，之前有很多同学在专栏的留言区提问：在实际使用场景中，我该如何确定是使用传统的消息中间件，还是使用Kafka呢？我在这里统一回答一下。如果在你的场景中，消息处理逻辑非常复杂，处理代价很高，同时你又不关心消息之间的顺序，那么传统的消息中间件是比较合适的；反之，如果你的场景需要较高的吞吐量，但每条消息的处理时间很短，同时你又很在意消息的顺序，此时，Kafka就是你的首选。

重设位移策略

不论是哪种设置方式，重设位移大致可以从两个维度来进行。

1. 位移维度。这是指根据位移值来重设。也就是说，直接把消费者的位移值重设成我们给定的位移值。
2. 时间维度。我们可以给定一个时间，让消费者把位移调整成大于该时间的最小位移；也可以给出一段时间间隔，比如30分钟前，然后让消费者直接将位移调回30分钟之前的位移值。

下面的这张表格罗列了7种重设策略。接下来，我来详细解释下这些策略。

维度	策略	含义
位移维度	Earliest	把位移调整到当前最早位移处
	Latest	把位移调整到当前最新位移处
	Current	把位移调整到当前最新提交位移处
	Specified-Offset	把位移调整成指定位移
	Shift-By-N	把位移调整到当前位移+N处（N可以是负值）
时间维度	DateTime	把位移调整到大于给定时间的最小位移处
	Duration	把位移调整到距离当前时间指定间隔的位移处

Earliest策略表示将位移调整到主题当前最早位移处。这个最早位移不一定就是0，因为在生产环境中，很久远的消息会被Kafka自动删除，所以当前最早位移很可能是一个大于0的值。**如果你想要重新消费主题的所有消息，那么可以使用Earliest策略**。

Latest策略表示把位移重设成最新末端位移。如果你总共向某个主题发送了15条消息，那么最新末端位移就是15。**如果你想跳过所有历史消息，打算从最新的消息处开始消费的话，可以使用Latest策略。**

Current策略表示将位移调整成消费者当前提交的最新位移。有时候你可能会碰到这样的场景：你修改了消费者程序代码，并重启了消费者，结果发现代码有问题，你需要回滚之前的代码变更，同时也要把位移重设到消费者重启时的位置，那么，Current策略就可以帮你实现这个功能。

表中第4行的Specified-Offset策略则是比较通用的策略，表示消费者把位移值调整到你指定的位移处。**这个策略的典型使用场景是，消费者程序在处理某条错误消息时，你可以手动地“跳过”此消息的处理**。在实际使用过程中，可能会出现corrupted消息无法被消费的情形，此时消

费者程序会抛出异常，无法继续工作。一旦碰到这个问题，你就可以尝试使用Specified-Offset策略来规避。

如果说Specified-Offset策略要求你指定位移的**绝对数值**的话，那么Shift-By-N策略指定的就是位移的**相对数值**，即你给出要跳过的一段消息的距离即可。这里的“跳”是双向的，你既可以向前“跳”，也可以向后“跳”。比如，你想把位移重设成当前位移的前100条位移处，此时你需要指定N为-100。

刚刚讲到的这几种策略都是位移维度的，下面我们来聊聊从时间维度重设位移的DateTime和Duration策略。

DateTime允许你指定一个时间，然后将位移重置到该时间之后的最早位移处。常见的使用场景是，你想重新消费昨天的数据，那么你可以使用该策略重设位移到昨天0点。

Duration策略则是指给定相对的时间间隔，然后将位移调整到距离当前给定时间间隔的位移处，具体格式是PnDTnHnMnS。如果你熟悉Java 8引入的Duration类的话，你应该不会对这个格式感到陌生。它就是一个符合ISO-8601规范的Duration格式，以字母P开头，后面由4部分组成，即D、H、M和S，分别表示天、小时、分钟和秒。举个例子，如果你想将位移调回到15分钟前，那么你就可以指定PT0H15M0S。

我会在后面分别给出这7种重设策略的实现方式。不过在此之前，我先来说一下重设位移的方法。目前，重设消费者组位移的方式有两种。

- 通过消费者API来实现。
- 通过kafka-consumer-groups命令行脚本来实现。

消费者API方式设置

首先，我们来看看如何通过API的方式来重设位移。我主要以Java API为例进行演示。如果你使用的是其他语言，方法应该是类似的，不过你要参考具体的API文档。

通过Java API的方式来重设位移，你需要调用KafkaConsumer的seek方法，或者是它的变种方法seekToBeginning和seekToEnd。我们来看下它们的方法签名。

```
void seek(TopicPartition partition, long offset);
void seek(TopicPartition partition, OffsetAndMetadata offsetAndMetadata);
void seekToBeginning(Collection<TopicPartition> partitions);
void seekToEnd(Collection<TopicPartition> partitions);
```

根据方法的定义，我们可以知道，每次调用seek方法只能重设一个分区的位移。OffsetAndMetadata类是一个封装了Long型的位移和自定义元数据的复合类，只是一般情况下，

自定义元数据为空，因此你基本上可以认为这个类表征的主要是消息的位移值。seek的变种方法seekToBeginning和seekToEnd则拥有一次重设多个分区的能力。我们在调用它们时，可以一次性传入多个主题分区。

好了，有了这些方法，我们就可以逐一地实现上面提到的7种策略了。我们先来看Earliest策略的实现方式，代码如下：

```
Properties consumerProperties = new Properties();
consumerProperties.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, false);
consumerProperties.put(ConsumerConfig.GROUP_ID_CONFIG, groupID);
consumerProperties.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
consumerProperties.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class.getNa
consumerProperties.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class.get
consumerProperties.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, brokerList);

String topic = "test";  // 要重设位移的Kafka主题
try (final KafkaConsumer<String, String> consumer =
 new KafkaConsumer<>(consumerProperties)) {
        consumer.subscribe(Collections.singleton(topic));
        consumer.poll(0);
        consumer.seekToBeginning(
 consumer.partitionsFor(topic).stream().map(partitionInfo ->
 new TopicPartition(topic, partitionInfo.partition()))
 .collect(Collectors.toList()));
}
```

这段代码中有几个比较关键的部分，你需要注意一下。

1. 你要创建的消费者程序，要禁止自动提交位移。
2. 组ID要设置成你要重设的消费者组的组ID。
3. 调用seekToBeginning方法时，需要一次性构造主题的所有分区对象。
4. 最重要的是，一定要调用带长整型的poll方法，而不要调用consumer.poll(Duration.ofSecond(0))。

虽然社区已经不推荐使用poll(long)了，但短期内应该不会移除它，所以你可以放心使用。另外，为了避免重复，在后面的实例中，我只给出最关键的代码。

Latest策略和Earliest是类似的，我们只需要使用seekToEnd方法即可，如下面的代码所示：

```
consumer.seekToEnd(
 consumer.partitionsFor(topic).stream().map(partitionInfo ->
 new TopicPartition(topic, partitionInfo.partition()))
 .collect(Collectors.toList()));
```

实现Current策略的方法很简单，我们需要借助KafkaConsumer的committed方法来获取当前提交的最新位移，代码如下：

```
consumer.partitionsFor(topic).stream().map(info ->
 new TopicPartition(topic, info.partition()))
 .forEach(tp -> {
 long committedOffset = consumer.committed(tp).offset();
 consumer.seek(tp, committedOffset);
});
```

这段代码首先调用partitionsFor方法获取给定主题的所有分区，然后依次获取对应分区上的已提交位移，最后通过seek方法重设位移到已提交位移处。

如果要实现Specified-Offset策略，直接调用seek方法即可，如下所示：

```
long targetOffset = 1234L;
for (PartitionInfo info : consumer.partitionsFor(topic)) {
 TopicPartition tp = new TopicPartition(topic, info.partition());
 consumer.seek(tp, targetOffset);
}
```

这次我没有使用Java 8 Streams的写法，如果你不熟悉Lambda表达式以及Java 8的Streams，这种写法可能更加符合你的习惯。

接下来我们来实现Shift-By-N策略，主体代码逻辑如下：

```
for (PartitionInfo info : consumer.partitionsFor(topic)) {
        TopicPartition tp = new TopicPartition(topic, info.partition());
 // 假设向前跳123条消息
        long targetOffset = consumer.committed(tp).offset() + 123L;
        consumer.seek(tp, targetOffset);
}
```

如果要实现DateTime策略，我们需要借助另一个方法：**KafkaConsumer. offsetsForTimes方法**。假设我们要重设位移到2019年6月20日晚上8点，那么具体代码如下：

```
long ts = LocalDateTime.of(
 2019, 6, 20, 20, 0).toInstant(ZoneOffset.ofHours(8)).toEpochMilli();
Map<TopicPartition, Long> timeToSearch =
        consumer.partitionsFor(topic).stream().map(info ->
 new TopicPartition(topic, info.partition()))
 .collect(Collectors.toMap(Function.identity(), tp -> ts));

for (Map.Entry<TopicPartition, OffsetAndTimestamp> entry :
 consumer.offsetsForTimes(timeToSearch).entrySet()) {
consumer.seek(entry.getKey(), entry.getValue().offset());
}
```

这段代码构造了LocalDateTime实例，然后利用它去查找对应的位移值，最后调用seek，实现了重设位移。

最后，我来给出实现Duration策略的代码。假设我们要将位移调回30分钟前，那么代码如下：

```
Map<TopicPartition, Long> timeToSearch = consumer.partitionsFor(topic).stream()
        .map(info -> new TopicPartition(topic, info.partition()))
        .collect(Collectors.toMap(Function.identity(), tp -> System.currentTimeMillis() - 30 * 1000  * 60));

for (Map.Entry<TopicPartition, OffsetAndTimestamp> entry :
        consumer.offsetsForTimes(timeToSearch).entrySet()) {
        consumer.seek(entry.getKey(), entry.getValue().offset());
}
```

总之，使用Java API的方式来实现重设策略的主要入口方法，就是seek方法。

命令行方式设置

位移重设还有另一个重要的途径：**通过kafka-consumer-groups脚本**。需要注意的是，这个功能是在Kafka 0.11版本中新引入的。这就是说，如果你使用的Kafka是0.11版本之前的，那么你只能使用API的方式来重设位移。

比起API的方式，用命令行重设位移要简单得多。针对我们刚刚讲过的7种策略，有7个对应的参数。下面我来一一给出实例。

Earliest策略直接指定**-to-earliest**。

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-host:port --group test-group --reset-offsets --all-topics --to-e
```

Latest策略直接指定**-to-latest**。

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-host:port --group test-group --reset-offsets --all-topics --to-la
```

Current策略直接指定**-to-current**。

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-host:port --group test-group --reset-offsets --all-topics --to-c
```

Specified-Offset策略直接指定**-to-offset**。

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-host:port --group test-group --reset-offsets --all-topics --to-o
```

Shift-By-N策略直接指定**-shift-by N**。

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-host:port --group test-group --reset-offsets --shift-by <offset_
```

DateTime策略直接指定**-to-datetime**。

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-host:port --group test-group --reset-offsets --to-datetime 201
```

最后是实现Duration策略，我们直接指定**--by-duration**。

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-host:port --group test-group --reset-offsets --by-duration PT0
```

小结

至此，重设消费者组位移的2种方式我都讲完了。我们来小结一下。今天，我们主要讨论了在Kafka中，为什么要重设位移以及如何重设消费者组位移。重设位移主要是为了实现消息的重演。目前Kafka支持7种重设策略和2种重设方法。在实际使用过程中，我推荐你使用第2种方法，即用命令行的方式来重设位移。毕竟，执行命令要比写程序容易得多。但是需要注意的是，0.11及0.11版本之后的Kafka才提供了用命令行调整位移的方法。如果你使用的是之前的版本，那么就只能依靠API的方式了。

Kafka重设消费者位移的7种策略和2种方法

7种策略

- Earliest：把位移调整到当前最早位移处。
- Latest：把位移调整到当前最新位移处。
- Current：把位移调整到当前最新提交位移处。
- Specified-Offset：把位移调整成指定位移。
- Shift-By-N：把位移调整到当前位移+N处。
- DateTime：把位移调整到大于给定时间的最小位移处。
- Duration：把位移调整到距离当前时间指定间隔的位移处。

2种方法

- 通过Java API的方式来重设位移：调用KafkaConsumer的seek方法，或者是它的变种方法seekToBeginning和seekToEnd。
- 用命令行重设位移。

开放讨论

你在实际使用过程中，是否遇到过要重设位移的场景，你是怎么实现的？

欢迎写下你的思考和答案，我们一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

精选留言

QQ怪 6

比较暴力的重新开个消费组从头消费

2019-08-10

锋芒 0

请问，用命令行重设位移，应该在当前group 的leader 节点上？

2019-08-23

作者回复

不需要的

2019-08-24

此方彼方Francis 0

有遇到过，之前有一条Kafka消息的crc校验值出错了（不知道为什么会出错，非常奇怪），这种状况下就只能跳过这条消息了。

不过有个问题请教老师，重设消费者位移之前，是不是有必要让消费者停止消费？

2019-08-22

作者回复

如果使用程序API的方式不必停止消费

2019-08-22

godtrue 0

打卡，此节讲解了

1：为啥需要重设消费者组位移？

当需要实现重复消费历史数据的时候，就需要重设消费者组位移。

2：重设消费者组位移的实现？

有两种方式七种策略(两种维度：位移和时间)

两种方式：操作API、操作kafka命令

七种策略：

2-1：Earliest 策略——表示将位移调整到主题当前最早位移处。

2-2：Latest 策略——表示把位移重设成最新末端位移。

2-3：Current 策略——表示将位移调整成消费者当前提交的最新位移。

2-4：Specified-Offset 策略——则是比较通用的策略，表示消费者把位移值调整到你指定的位移处。这个策略的典型使用场景是，消费者程序在处理某条错误消息时，你可以手动地“跳过”此消息的处理。

2-5： Shift-By-N 策略——指定的是位移的相对数值，即你给出要跳过的一段消息的距离即可。这里的“跳”是双向的，你既可以向前“跳”，也可以向后“跳”。

2-6：DateTime策略—— 允许你指定一个时间，然后将位移重置到该时间之后的最早位移处。

2-7：Duration 策略——则是指给定相对的时间间隔，然后将位移调整到距离当前给定时间间隔的位移处，具体格式是 PnDTnHnMnS。

2019-08-19

常银玲

0

老师，留言想问一个问题，现在项目有一个需求是做一套仿真系统，仿真的数据来源于之前历史数据，查询准备用es像这种情况我们可以用kafka吗？

2019-08-12

作者回复

我觉得可以用啊

2019-08-12

Li Shunduo

0

试了下开着console consumer的时候去调整offset,遇到以下错误:
Error: Assignments can only be reset if the group 'test_group' is inactive, but the current state is Stable.
停掉console consumer之后，就可以调整offset了。
好像不能动态调整?

2019-08-12

作者回复

嗯，必须是非active的group才行

2019-08-12

cricket1981 0

"最重要的是，一定要调用带长整型的 poll 方法，而不要调用 consumer.poll(Duration.ofSecond(0))。"--- 能讲下为什么吗？如果不遵守会怎么样？

2019-08-12

作者回复

两个的实现方式不一样。详细设计原理差别可以看看：https://www.cnblogs.com/huxi2b/p/10773559.html

2019-08-12

边城 0

老师您好，请教您两个问题。

如果把位移调整成指定位移，是不是每次消费都需要持久化位移点？

这样的话，发布应用的时候，我应该怎么停止消费程序进程呢？

感谢！

2019-08-12

作者回复

你不需要停止程序也能动态调整位移

2019-08-12

陈华应 0

生产环境中遇到过这种情况，
现象是:本来正常消费的一个topic，突然因为业务调整，大数据侧向此topic推送了大量的另一个平台的历史数据，而这些数据对现在使用此topic的业务场景是无效数据，并且评估到按现有能力吧历史数据消化完需要几个小时时间，业务上是不能接受的
最终就是通过调整位移来“越过”历史数据，消费最新的数据，解决了可能是故障的一次

2019-08-11

我自成魔 0

老师，我当前使用的Kafka是0.10，重置offset使用seek方式，但是在实际使用过程中，发现使用subscribe加poll消费消息，无法消费到消息，程序也不报错，而使用seek方式重新指定各分

区的offset进行消费就可以，麻烦老师解惑，希望老师可以讲一下Kafka如何去消费消息的，谢谢！

2019-08-11

作者回复

是不是没有数据可以消费了呢？另外0.10版本的新consumer不建议使用，这个时候bug还比较多。建议至少到0.10.2或0.11之后再切换到新版本consumer

2019-08-12

willmeng 0

python读取过kafka，关闭自动提交，把每个分区的位移写文件。重启程序或者需要重新读，就靠文件了。

2019-08-11

Stalary 0

老师，我的理解latest是不是应该是当前消费者组消费到的最新位移，目前我是使用latest来保证不丢消息的是否有问题呢，生产环境中会出现消费失败的情况，这个时候就会手动关闭消费者不再提交位移，处理完毕后重启从上次挂掉的offset继续消费。

2019-08-11

mickle 0

请教老师一个问题，java连接kafka时报 "IllegalStateException: No entry found for connection 2147483647" 的错误，大概率是什么原因？

2019-08-10

作者回复

像是DNS配置的问题。目前Kafka直连主机，最好两端统一使用主机名交互—Broker端的listeners配置主机名，clients端连接使用主机名

2019-08-12

我来也 0

请问在发布订阅模式下，也可以用这些调整位移的方法么？

2019-08-10

作者回复

都可以的

2019-08-12

许童童 0

你在实际使用过程中，是否遇到过要重设位移的场景，你是怎么实现的？
还没有遇到过重设位移的场景，后续有遇到，再来留言。

2019-08-10

开水 0

我们用的0.9版本，当机器资源跟不上的时候，消费跟不上生产速度。所以只好把不怎么重要的消息offset重设了，用的是直接set znode的方式。

2019-08-10

明翼

0

以前遇到过无法消费的信息只能重新搞个新的topic原来还有这个操作666

2019-08-10

曹伟雄

0

specified offser，这个策略的典型使用场景是，消费者程序在处理某条错误消息时，你可以手动地“跳过”此消息的处理。

这里不是很理解，是指消费时发生业务失败或运行时异常时，继续往下一条消费吗？ 还是说重新seek到这条记录重新消费?

如果是第1种情况，消费失败的消息后续怎么处理? 如果是第2种情况，seek重新消费时，还是处理失败，这样就导致死循环了。

关于消费失败处理这一块，我还是不太清晰，希望老师能解答一下，谢谢。

2019-08-10

作者回复

seek跳到下一条消息，跳过这个corrupted消息

2019-08-12