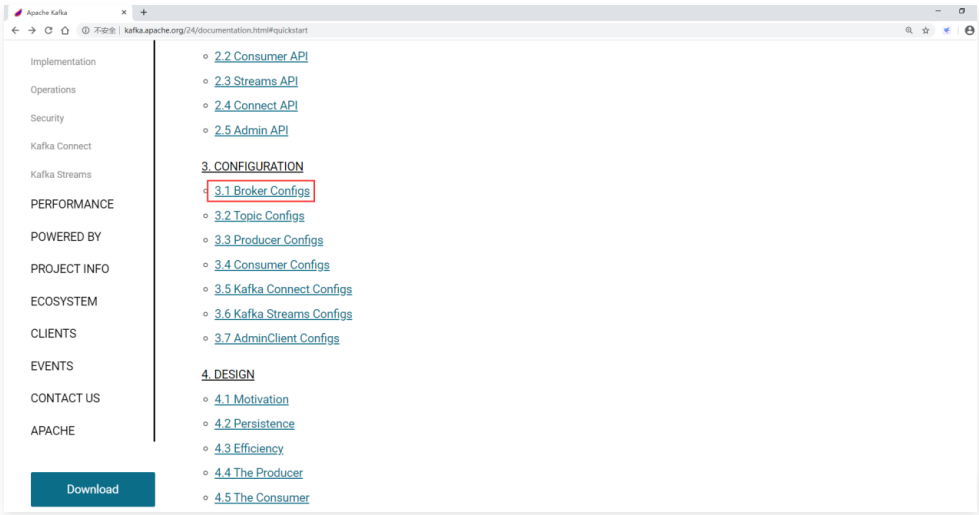
徐老师 • 更新于 2020-09-25

上一节 3 Kafka使用初体… 5 Kafka核心之存… 下一节

Broker扩展

Broker的参数可以配置在server.properties这个配置文件中，Broker中支持的完整参数在官方文档中有体现



具体链接为：
http://kafka.apache.org/24/documentation.html#brokerconfigs

针对Broker的参数，我们主要分析两块
1：Log Flush Policy：设置数据flush到磁盘的时机
为了减少磁盘写入的次数,broker会将消息暂时缓存起来,当消息的个数达到一定阀值或者过了一定的时间间隔后,再flush到磁盘,这样可以减少磁盘IO调用的次数。
这块主要通过两个参数控制

- `log.flush.interval.messages` 一个分区的消息数阀值，达到该阈值则将该分区的数据flush到磁盘，注意这里是针对分区，因为topic是一个逻辑概念，分区是真实存在的，每个分区会在磁盘上产生一个目录

代码块

```
[root@bigdata01 kafka-logs]# ll
total 20
drwxr-xr-x. 2 root root 141 Jun  8 17:12 88888888-0
drwxr-xr-x. 2 root root 141 Jun  8 17:12 88888888-3
-rw-r--r--. 1 root root   4 Jun  8 15:23 cleaner-offset-checkpoint
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-0
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-12
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-15
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-18
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-21
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-24
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-27
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-3
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-30
```

```
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-36
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-39
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-42
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-45
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-48
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-6
drwxr-xr-x. 2 root root 141 Jun  8 17:08 __consumer_offsets-9
drwxr-xr-x. 2 root root 141 Jun  8 17:04 hello-1
drwxr-xr-x. 2 root root 141 Jun  8 17:04 hello-4
```

这个参数的默认值为 `9223372036854775807` ，long的最大值
默认值太大了，所以建议修改，可以使用 `server.properties` 中针对这个参数指定的值10000，需要去掉注释之后这个参数才生效。

- `log.flush.interval.ms` 间隔指定时间
  默认间隔指定的时间将内存中缓存的数据flush到磁盘中，由文档可知，这个参数的默认值为 `null` ，此时会使用 `log.flush.scheduler.interval.ms` 参数的值， `log.flush.scheduler.interval.ms` 参数的值默认是 `9223372036854775807` ，long的最大值
  所以这个值也建议修改，可以使用 `server.properties` 中针对这个参数指定的值1000，单位是毫秒，表示每1秒写一次磁盘，这个参数也需要去掉注释之后才生效

2：Log Retention Policy：设置数据保存周期，默认7天
kafka中的数据默认会保存7天，如果kafka每天接收的数据量过大，这样是很占磁盘空间的，建议修改数据保存周期，我们之前在实际工作中是将数据保存周期改为了1天。

数据保存周期主要通过这几个参数控制

- `log.retention.hours` ，这个参数默认值为168，单位是小时，就是7天，可以在这调整数据保存的时间，超过这个时间数据会被自动删除
- `log.retention.bytes` ，这个参数表示当分区的文件达到一定大小的时候会删除它，
  如果设置了按照指定周期删除数据文件，这个参数不设置也可以，这个参数默认是没有开启的
- `log.retention.check.interval.ms` ，这个参数表示检测的间隔时间，单位是毫秒，默认值是300000，就是5分钟，表示每5分钟检测一次文件看是否满足删除的时机

Producer扩展

Producer默认是随机将数据发送到topic的不同分区中，也可以根据用户设置的算法来根据消息的key来计算输入到哪个partition里面
此时需要通过partitioner来控制，这个知道就行了，因为在实际工作中一般在向kafka中生产数据的都是不带key的，只有数据内容，所以一般都是使用随机的方式发送数据

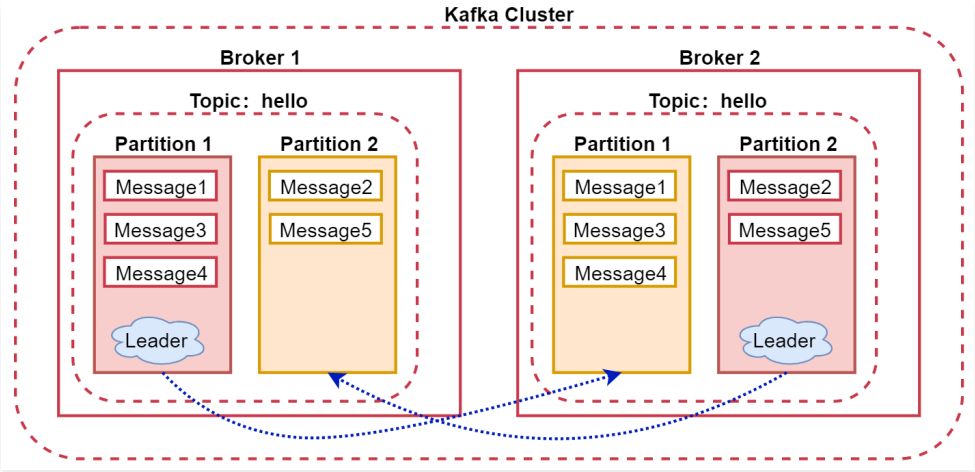
在这里有一个需要注意的内容就是

> 针对producer的数据通讯方式：同步发送和异步发送

同步是指：生产者发出数据后，等接收方发回响应以后再发送下个数据的通讯方式。
异步是指：生产者发出数据后，不等接收方发回响应，接着发送下个数据的通讯方式。

具体的数据通讯策略是由 `acks` 参数控制的
acks默认为 `1` ，表示需要Leader节点回复收到消息，这样生产者才会发送下一条数据
acks： `all` ，表示需要所有Leader+副本节点回复收到消息（acks=-1），这样生产者才会发送下一条数据
acks： `0` ，表示不需要任何节点回复，生产者会继续发送下一条数据

再来看一下这个图：



我们在向hello这个topic生产数据的时候，可以在生产者中设置acks参数，
acks设置为1，表示我们在向hello这个topic的partition1这个分区写数据的时候，只需要让leader所在的broker1这个节点回复确认收到的消息就可以了，这样生产者就可以发送下一条数据了

如果acks设置为all，则需要partition1的这两个副本所在的节点(包含Leader)都回复收到消息，生产者才会发送下一条数据

如果acks设置为0，表示生产者不会等待任何partition所在节点的回复，它只管发送数据，不管你有没有收到，所以这种情况丢失数据的概率比较高。

> 针对这块在面试的时候会有一个面试题：Kafka如何保证数据不丢？

其实就是通过acks机制保证的，如果设置acks为all，则可以保证数据不丢，因为此时把数据发送给kafka之后，会等待对应partition所在的所有leader和副本节点都确认收到消息之后才会认为数据发送成功了，所以在这种策略下，只要把数据发送给kafka之后就不会丢了。
如果acks设置为1，则当我们把数据发送给partition之后，partition的leader节点也确认收到了，但是leader回复完确认消息之后，leader对应的节点就宕机了，副本partition还没来得及将数据同步过去，所以会存在丢失的可能性。
不过如果宕机的是副本partition所在的节点，则数据是不会丢的

如果acks设置为0的话就表示是顺其自然了，只管发送，不管kafka有没有收到，这种情况表示对数据丢不丢都无所谓了。

Consumer扩展

在消费者中还有一个消费者组的概念
每个consumer属于一个消费者组，通过group.id指定消费者组

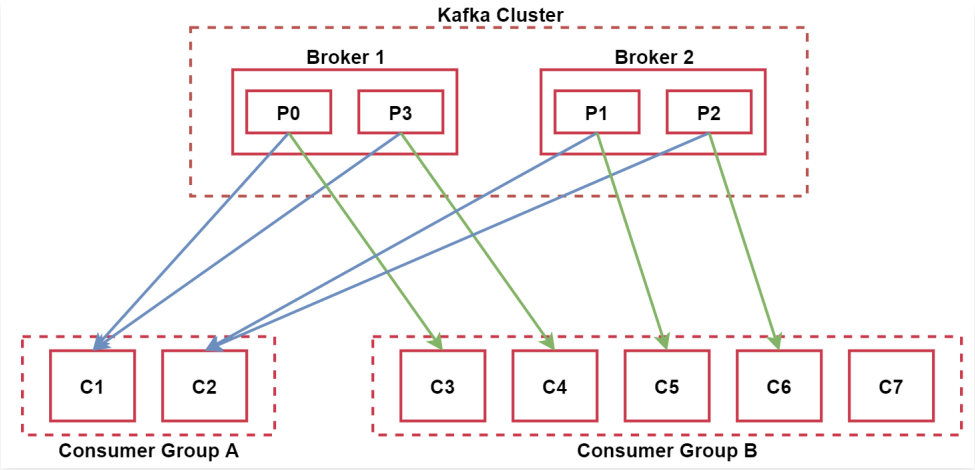
那组内消费和组间消费有什么区别吗？

- 组内：消费者组内的所有消费者消费同一份数据；

注意：在同一个消费者组中，一个partition同时只能有一个消费者消费数据
如果消费者的个数小于分区的个数，一个消费者会消费多个分区的数据。
如果消费者的个数大于分区的个数，则多余的消费者不消费数据
所以，对于一个topic,同一个消费者组中推荐不能有多于分区个数的消费者,否则将意味着某些消费者将无法获得消息。

- 组间：多个消费者组消费相同的数据，互不影响。

来看下面这个图，加深一下理解



Kafka集群有两个节点，Broker1和Broker2
集群内有一个topic，这个topic有4个分区，P0,P1,P2,P3
下面有两个消费者组
Consumer Group A和Consumer Group B
其中Consumer Group A中有两个消费者 C1和C2，由于这个topic有4个分区，所以，C1负责消费两个分区的数据，C2负责消费两个分区的数据，这个属于组内消费

Consumer Group B有5个消费者，C3~C7，其中C3,C4,C5,C6分别消费一个分区的数据，而C7就是多余出来的了，因为现在这个消费者组内的消费者的数量比对应的topic的分区数量还多，但是一个分区同时只能被一个消费者消费，所以就会有一个消费者处于空闲状态。
这个也属于组内消费

Consumer Group A和Consumer Group B这两个消费者组属于组间消费，互不影响。

3 Kafka使用初体验 上一节 下一节 5 Kafka核心之存储和容错机制

我要提出意见反馈

意见反馈 收藏教程 标记书签