复制限流

在上一节中我们了解了分区重分配本质在于数据复制，先增加新的副本，然后进行数据同步，最后删除旧的副本来达到最终的目的。数据复制会占用额外的资源，如果重分配的量太大必然会严重影响整体的性能，尤其是处于业务高峰期的时候。减小重分配的粒度，以小批次的方式来操作是一种可行的解决思路。如果集群中某个主题或某个分区的流量在某段时间内特别大，那么只靠减小粒度是不足以应对的，这时就需要有一个限流的机制，可以对副本间的复制流量加以限制来保证重分配期间整体服务不会受太大的影响。

副本间的复制限流有两种实现方式：kafka-config.sh 脚本和 kafka-reassign-partitions.sh 脚本。

首先，我们讲述如何通过 kafka-config.sh 脚本来实现限流，如果对这个脚本的使用有些遗忘，则可以再回顾一下19节的内容。不过19节里只演示了主题相关的配置变更，并没有涉及其他的类型，本节的内容会与broker类型的配置相关，不妨借助这个机会再来了解一下 broker 类型的配置用法。

kafka-config.sh 脚本主要以动态配置的方式来达到限流的目的，在 broker 级别有两个与复制限流相关的配置参数：follower.replication.throttled.rate 和 leader.replication.throttled.rate，前者用于设置 follower 副本复制的速度，后者用于设置 leader 副本传输的速度，它们的单位都是 B/s。通常情况下，两者的配置值是相同的。下面的示例中将 broker 1中的 leader 副本和 follower 副本的复制速度限制在1024B/s之内，即1KB/s：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
configs.sh --zookeeper localhost: 2181/kafka --
entity-type brokers --entity-name 1 --alter --
add-config follower.replication.
throttled.rate=1024,leader.replication.throttled.
rate=1024
Completed Updating config for entity: brokers
'1'.
```

我们再来查看一下 broker 1 中刚刚添加的配置，参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
configs.sh --zookeeper localhost: 2181/kafka --
entity-type brokers --entity-name 1 --describe
Configs for brokers '1' are
leader.replication.throttled.rate=1024,follower.
replication.throttled.rate=1024
```

在19节中我们了解到变更配置时会在 ZooKeeper 中创建一个命名形式为/config/ <entity-type>/<entity-name>的节点，对于这里的示例而言，其节点就是/config/brokers/1，节点中相应的信息如下：

```
[zk: localhost:2181/kafka(CONNECTED) 6] get
/config/brokers/1
{"version":1,"config":
{"leader.replication.throttled.rate":"1024","foll
ower.replication.throttled.rate":"1024"}}
```

删除刚刚添加的配置也很简单，与19节中主题类型的方式一样，参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
configs.sh --zookeeper localhost: 2181/kafka --
entity-type brokers --entity-name 1 --alter --
delete-config follower.
replication.throttled.rate,leader.replication.thr
ottled.rate
Completed Updating config for entity: brokers
'1'.
```

在主题级别也有两个相关的参数来限制复制的速度：leader.replication.throttled. replicas 和 follower.replication.throttled.replicas，它们分别用来配置被限制速度的主题所对应的 leader 副本列表和 follower 副本列表。为了演示具体的用法，我们先创建一个分区数为3、副本数为2的主题 topic-throttle，并查看它的详细信息：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
create --topic topic-throttle --replication-
factor 2 --partitions 3
Created topic "topic-throttle".

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-throttle
Topic:topic-throttle    PartitionCount:3
ReplicationFactor:2 Configs:
    Topic: topic-throttle   Partition: 0
Leader: 0   Replicas: 0,1   Isr: 0,1
    Topic: topic-throttle   Partition: 1
Leader: 1   Replicas: 1,2   Isr: 1,2
    Topic: topic-throttle   Partition: 2
Leader: 2   Replicas: 2,0   Isr: 2,0
```

在上面示例中，主题 topic-throttle 的三个分区所对应的 leader 节点分别为0、1、2，即分区与代理的映射关系为0:0、1:1、2:2，而对应的 follower 节点分别为1、2、0，相关的分区与代理的映射关系为0:1、1:2、2:0，那么此主题的限流副本列表及具体的操作细节如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
configs.sh --zookeeper localhost: 2181/kafka --
entity-type topics --entity-name topic-throttle -
-alter --add-config
leader.replication.throttled.replicas=
[0:0,1:1,2:2],follower.replication.throttled.repl
icas=[0:1,1:2,2:0]
Completed Updating config for entity: topic
'topic-throttle'.
```

对应的 ZooKeeper 中的/config/topics/topic-throttle 节点信息如下：

```
{"version":1,"config":
{"leader.replication.throttled.replicas":"0:0,1:1
,2:2","follower.replication.throttled.replicas":"
0:1,1:2,2:0"}}
```

在了解了与限流相关的4个配置参数之后，我们演示一下带有限流的分区重分配的用法。首先按照上一节的步骤创建一个包含可行性方案的 project.json 文件，内容如下：

```
{"version":1,"partitions":[{"topic":"topic-
throttle","partition":1,"replicas":
[2,0],"log_dirs":["any","any"]},{"topic":"topic-
throttle","partition":0,"replicas":
[0,2],"log_dirs":["any","any"]},{"topic":"topic-
throttle","partition":2,"replicas":
[0,2],"log_dirs":["any","any"]}]}
```

接下来设置被限流的副本列表，这里就很有讲究了，首先看一下重分配前和分配后的分区副本布局对比，详细如下：

partition	重分配前的AR	分配后的预期AR
0	0,1	0,2
1	1,2	2,0
2	2,0	0,2

如果分区重分配会引起某个分区AR集合的变更，那么这个分区中与 leader 有关的限制会应用于重分配前的所有副本，因为任何一个副本都可能是 leader，而与 follower 有关的限制会应用于所有移动的目的地。从概念上理解会比较抽象，这里不妨举个例子，对上面的布局对比而言，分区0重分配的AR为[0,1]，重分配后的AR为[0,2]，那么这里的目的地就是新增的2。也就是说，对分区0而言，leader.replication.throttled.replicas 配置为[0:0, 0:1]，follower.replication.throttled.replicas 配置为[0:2]。同理，对于分区1而言，leader.replication.throttled.replicas 配置为[1:1,1:2]，follower.replication. throttled.replicas 配置为[1:0]。分区3的AR集合没有发生任何变化，这里可以忽略。

获取限流副本列表之后，我们就可以执行具体的操作了，详细如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
configs.sh --zookeeper localhost: 2181/kafka --
entity-type topics --entity-name topic-throttle -
-alter --add-config
leader.replication.throttled.replicas=
[1:1,1:2,0:0,0:1],follower.replication.throttled.
replicas=[1:0,0:2]
Completed Updating config for entity: topic
'topic-throttle'.
```

接下来再设置 broker 2 的复制速度为10B/s，这样在下面的操作中可以很方便地观察限流与不限流的不同：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
configs.sh --zookeeper localhost: 2181/kafka --
entity-type brokers --entity-name 2 --alter --
add-config follower.
replication.throttled.rate=10,leader.replication.
throttled.rate=10
Completed Updating config for entity: brokers
'2'.
```

在执行具体的重分配操作之前，我们需要开启一个生产者并向主题 topic-throttle 中发送一批消息，这样可以方便地观察正在进行数据复制的过程。

之后我们再执行正常的分区重分配的操作，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
reassign-partitions.sh --zookeeper
localhost:2181/kafka --execute --reassignment-
json-file project.json
Current partition replica assignment

{"version":1,"partitions":[{"topic":"topic-
throttle","partition":2,"replicas":
[2,0],"log_dirs":["any","any"]},{"topic":"topic-
throttle","partition":1,"replicas":
[1,2],"log_dirs":["any","any"]},{"topic":"topic-
throttle","partition":0,"replicas":
[0,1],"log_dirs":["any","any"]}]}

Save this to use as the --reassignment-json-file
option during rollback
Successfully started reassignment of partitions.
```

执行之后，可以查看执行的进度，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
reassign-partitions.sh --zookeeper
localhost:2181/kafka --verify --reassignment-
json-file project.json
Status of partition reassignment:
Reassignment of partition topic-throttle-1
completed successfully
Reassignment of partition topic-throttle-0 is
still in progress
Reassignment of partition topic-throttle-2
completed successfully
```

可以看到分区 topic-throttle-0 还在同步过程中，因为我们之前设置了 broker 2 的复制速度为10B/s，这样使同步变得缓慢，分区 topic-throttle-0 需要同步数据到位于 broker 2 的新增副本中。随着时间的推移，分区 topic-throttle-0 最终会变成“completed successful”的状态。

为了不影响 Kafka 本身的性能，往往对临时设置的一些限制性的配置在使用完后要及时删除，而 kafka-reassign-partitions.sh 脚本配合指令参数 verify 就可以实现这个功能，在所有的分区都重分配完成之后执行查看进度的命令时会有如下的信息：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
reassign-partitions.sh --zookeeper
localhost:2181/kafka --verify --reassignment-
json-file project.json
Status of partition reassignment:
Reassignment of partition topic-throttle-1
completed successfully
Reassignment of partition topic-throttle-0
completed successfully
Reassignment of partition topic-throttle-2
completed successfully
Throttle was removed.
```

注意到最后一行信息“Throttle was removed.”，它提示了所有之前针对限流做的配置都已经被清除了，读者可以自行查看一下相应的 ZooKeeper 节点中是否还有相关的配置。

kafka-reassign-partitions.sh 脚本本身也提供了限流的功能，只需一个 throttle 参数即可，具体用法如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-reassign-partitions.sh --zookeeper localhost:2181/kafka --execute --reassignment-json-file project.json  --throttle 10
Current partition replica assignment

{"version":1,"partitions":[{"topic":"topic-throttle","partition":2,"replicas":[2,0],"log_dirs":["any","any"]},{"topic":"topic-throttle","partition":1,"replicas":[1,2],"log_dirs":["any","any"]},{"topic":"topic-throttle","partition":0,"replicas":[0,1],"log_dirs":["any","any"]}]}

Save this to use as the --reassignment-json-file option during rollback
Warning: You must run Verify periodically, until the reassignment completes, to ensure the throttle is removed. You can also alter the throttle by rerunning the Execute command passing a new value.
The inter-broker throttle limit was set to 10 B/s
Successfully started reassignment of partitions.
```

上面的信息中包含了明确的告警信息：需要周期性地执行查看进度的命令直到重分配完成，这样可以确保限流设置被移除。也就是说，使用这种方式的限流同样需要显式地执行某些操作以使在重分配完成之后可以删除限流的设置。上面的信息中还告知了目前限流的速度上限为10B/s。

如果想在重分配期间修改限制来增加吞吐量，以便完成得更快，则可以重新运行 kafka- reassign-partitions.sh 脚本的 execute 命令，使用相同的 reassignment-json-file，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
reassign-partitions.sh --zookeeper
localhost:2181/kafka --execute --reassignment-
json-file project.json  --throttle 1024
There is an existing assignment running.
```

这样限流的速度上限为1024B/s，可以查看对应的 ZooKeeper 节点内容：

```
[zk: localhost:2181/kafka(CONNECTED) 30] get
/config/topics/topic-throttle
{"version":1,"config":
{"follower.replication.throttled.replicas":"1:0,0
:2","leader.replication.throttled.replicas":"1:1,
1:2,0:0,0:1"}}
```

可以看到 ZooKeeper 节点内容中的限流副本列表和前面使用 kafka-config.sh 脚本时的一样。其实 kafka-reassign-partitions.sh 脚本提供的限流功能背后的实现原理就是配置与限流相关的那4个参数而已，没有什么太大的差别。不过使用 kafka-config.sh 脚本的方式来实现复制限流的功能比较烦琐，并且在手动配置限流副本列表时也比较容易出错，这里推荐大家使用 kafka-reassign-partitions.sh 脚本配合 throttle 参数的方式，方便快捷且不容易出错。