分区重分配

当集群中的一个节点突然宕机下线时，如果节点上的分区是单副本的，那么这些分区就变得不可用了，在节点恢复前，相应的数据也就处于丢失状态；如果节点上的分区是多副本的，那么位于这个节点上的 leader 副本的角色会转交到集群的其他 follower 副本中。总而言之，这个节点上的分区副本都已经处于功能失效的状态，Kafka 并不会将这些失效的分区副本自动地迁移到集群中剩余的可用 broker 节点上，如果放任不管，则不仅会影响整个集群的均衡负载，还会影响整体服务的可用性和可靠性。

当要对集群中的一个节点进行有计划的下线操作时，为了保证分区及副本的合理分配，我们也希望通过某种方式能够将该节点上的分区副本迁移到其他的可用节点上。

当集群中新增 broker 节点时，只有新创建的主题分区才有可能被分配到这个节点上，而之前的主题分区并不会自动分配到新加入的节点中，因为在它们被创建时还没有这个新节点，这样新节点的负载和原先节点的负载之间严重不均衡。

为了解决上述问题，需要让分区副本再次进行合理的分配，也就是所谓的分区重分配。Kafka 提供了 kafka-reassign-partitions.sh 脚本来执行分区重分配的工作，它可以在集群扩容、broker 节点失效的场景下对分区进行迁移。
kafka-reassign-partitions.sh 脚本的使用分为3个步骤：首先创建需要一个包含主题清单的 JSON 文件，其次根据主题清单和 broker 节点清单生成一份重分配方案，最后根据这份方案执行具体的重分配动作。

下面我们通过一个具体的案例来演示 kafka-reassign-partitions.sh 脚本的用法。首先在一个由3个节点（broker 0、broker 1、broker 2）组成的集群中创建一个主题 topic-

reassign，主题中包含4个分区和2个副本：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
create --topic topic-reassign --replication-
factor 2 --partitions 4
Created topic "topic-reassign".

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-reassign
Topic:topic-reassign    PartitionCount:4
ReplicationFactor:2 Configs:
    Topic: topic-reassign   Partition: 0
Leader: 0   Replicas: 0,2   Isr: 0,2
    Topic: topic-reassign   Partition: 1
Leader: 1   Replicas: 1,0   Isr: 1,0
    Topic: topic-reassign   Partition: 2
Leader: 2   Replicas: 2,1   Isr: 2,1
    Topic: topic-reassign   Partition: 3
Leader: 0   Replicas: 0,1   Isr: 0,1
```

我们可以观察到主题 topic-reassign 在3个节点中都有相应的分区副本分布。由于某种原因，我们想要下线 brokerId 为1的 broker 节点，在此之前，我们要做的就是将其上的分区副本迁移出去。使用 kafka-reassign-partitions.sh 脚本的第一步就是要创建一个 JSON 文件（文件的名称假定为 reassign.json），文件内容为要进行分区重分配的主题清单。对主题 topic-reassign 而言，示例如下：

```
{
        "topics":[
                {
                        "topic":"topic-reassign"
                }
        ],
        "version":1
}
```

第二步就是根据这个 JSON 文件和指定所要分配的 broker 节点列表来生成一份候选的重分配方案，具体内容参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
reassign-partitions.sh --zookeeper
localhost:2181/kafka --generate --topics-to-move-
json-file reassign.json --broker-list 0,2
Current partition replica assignment
{"version":1,"partitions":[{"topic":"topic-
reassign","partition":2,"replicas":
[2,1],"log_dirs":["any","any"]},{"topic":"topic-
reassign","partition":1,"replicas":
[1,0],"log_dirs":["any","any"]},{"topic":"topic-
reassign","partition":3,"replicas":
[0,1],"log_dirs":["any","any"]},{"topic":"topic-
reassign","partition":0,"replicas":
[0,2],"log_dirs":["any","any"]}]}

Proposed partition reassignment configuration
{"version":1,"partitions":[{"topic":"topic-
reassign","partition":2,"replicas":
[2,0],"log_dirs":["any","any"]},{"topic":"topic-
reassign","partition":1,"replicas":
[0,2],"log_dirs":["any","any"]},{"topic":"topic-
reassign","partition":3,"replicas":
[0,2],"log_dirs":["any","any"]},{"topic":"topic-
reassign","partition":0,"replicas":
[2,0],"log_dirs":["any","any"]}]}
```

上面的示例中包含4个参数，其中 zookeeper 已经很常见了，用来指定 ZooKeeper 的地址。generate 是 kafka-reassign-partitions.sh 脚本中指令类型的参数，可以类比于 kafka-topics.sh 脚本中的 create、list 等，它用来生成一个重分配的候选方案。topic-to-move-json 用来指定分区重分配对应的主题清单

文件的路径，该清单文件的具体的格式可以归纳为{"topics": [{"topic": "foo"},{"topic": "foo1"}],"version": 1}。broker-list 用来指定所要分配的 broker 节点列表，比如示例中的“0,2”。

上面示例中打印出了两个 JSON 格式的内容。第一个“Current partition replica assignment”所对应的 JSON 内容为当前的分区副本分配情况，在执行分区重分配的时候最好将这个内容保存起来，以备后续的回滚操作。第二个“Proposed partition reassignment configuration”所对应的 JSON 内容为重分配的候选方案，注意这里只是生成一份可行性的方案，并没有真正执行重分配的动作。生成的可行性方案的具体算法和创建主题时的一样，这里也包含了机架信息，具体的细节可以参考17节的内容。

我们需要将第二个 JSON 内容保存在一个 JSON 文件中，假定这个文件的名称为 project.json。

第三步执行具体的重分配动作，详细参考如下：

```
[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-reassign","partition":2,"replicas":[2,1],"log_dirs":["any","any"]},{"topic":"topic-reassign","partition":1,"replicas":[1,0],"log_dirs":["any","any"]},{"topic":"topic-reassign","partition":3,"replicas":[0,1],"log_dirs":["any","any"]},{"topic":"topic-reassign","partition":0,"replicas":[0,2],"log_dirs":["any","any"]}]}

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

我们再次查看主题 topic-reassign 的具体信息：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-reassign
Topic:topic-reassign    PartitionCount:4
ReplicationFactor:2 Configs:
    Topic: topic-reassign   Partition: 0
Leader: 0   Replicas: 2,0   Isr: 0,2
    Topic: topic-reassign   Partition: 1
Leader: 0   Replicas: 0,2   Isr: 0,2
    Topic: topic-reassign   Partition: 2
Leader: 2   Replicas: 2,0   Isr: 2,0
    Topic: topic-reassign   Partition: 3
Leader: 0   Replicas: 0,2   Isr: 0,2
```

可以看到主题中的所有分区副本都只在0和2的 broker 节点上分布了。

在第三步的操作中又多了2个参数，execute 也是指令类型的参数，用来指定执行重分配的动作。reassignment-json-file 指定分区重分配方案的文件路径，对应于示例中的 project.json 文件。

除了让脚本自动生成候选方案，用户还可以自定义重分配方案，这样也就不需要执行第一步和第二步的操作了。

分区重分配的基本原理是先通过控制器为每个分区添加新副本（增加副本因子），新的副本将从分区的 leader 副本那里复制所有的数据。根据分区的大小不同，复制过程可能需要花一些时间，因为数据是通过网络复制到新副本上的。在复制完成之后，控制器将旧副本从副本清单里移除（恢复为原先的副本因子数）。注意在重分配的过程中要确保有足够的空间。

细心的读者可能观察到主题 topic-reassign 中有3个 leader 副本在 broker 0 上，而只有1个 leader 副本在 broker 2 上，这样负载就不均衡了。不过我们可以借助上一节中的 kafka-perferred-

replica-election.sh 脚本来执行一次优先副本的选举动作，之后可以看到主题 topic-reassign 的具体信息已经趋于完美：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-reassign
Topic:topic-reassign    PartitionCount:4
ReplicationFactor:2 Configs:
    Topic: topic-reassign   Partition: 0
Leader: 2   Replicas: 2,0   Isr: 0,2
    Topic: topic-reassign   Partition: 1
Leader: 0   Replicas: 0,2   Isr: 0,2
    Topic: topic-reassign   Partition: 2
Leader: 2   Replicas: 2,0   Isr: 2,0
    Topic: topic-reassign   Partition: 3
Leader: 0   Replicas: 0,2   Isr: 0,2
```

对于分区重分配而言，这里还有可选的第四步操作，即验证查看分区重分配的进度。只需将上面的 execute 替换为 verify 即可，具体示例如下：

```
[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-reassign-2
completed successfully
Reassignment of partition topic-reassign-1
completed successfully
Reassignment of partition topic-reassign-3
completed successfully
Reassignment of partition topic-reassign-0
completed successfully
```

分区重分配对集群的性能有很大的影响，需要占用额外的资源，比如网络和磁盘。在实际操作中，我们将降低重分配的粒度，分成多个小批次来执行，以此来将负面的影响降到最低，这一点和优先副本的选举有异曲同工之妙。

还需要注意的是，如果要将某个 broker 下线，那么在执行分区重分配动作之前最好先关闭或重启 broker。这样这个 broker 就不再是任何分区的 leader 节点了，它的分区就可以被分配给集群中的其他 broker。这样可以减少 broker 间的流量复制，以此提升重分配的性能，以及减少对集群的影响。