1.前言:

对于整个选举算法的详情需要先了解Raft选举算法，kafka是基于该算法来实现leader选举的。有兴趣的读者可以参考之前的文章【分布式一致性协议：Raft算法详解】。

kafka 的选举有三类:

1. 控制器（Broker）选主
2. 分区多副本选主
3. 消费组选主

2.选举类型:

2.1 控制器（Broker）选举

所谓控制器就是一个Borker，在一个kafka集群中，有多个broker节点，但是它们之间需要选举出一个leader，其他的broker充当follower角色。集群中第一个启动的broker会通过在zookeeper中创建临时节点/controller来让自己成为控制器(其实大家竞争成为控制器,一般先启动的,先注册成功嘛)，其他broker启动时也会在zookeeper中创建临时节点，但是发现节点已经存在，所以它们会收到一个异常，意识到控制器已经存在，那么就会在zookeeper中创建watch对象，便于它们收到控制器变更的通知。

那么如果控制器由于网络原因与zookeeper断开连接或者异常退出(此时其他borker也是竞争成为控制器)，那么其他broker通过watch收到控制器变更的通知，就会

去尝试创建临时节点/controller，如果有一个broker创建成功，那么其他broker就会收到创建异常通知，也就意味着集群中已经有了控制器，其他broker只需创建watch对象即可。

如果集群中有一个broker发生异常退出了，那么控制器就会检查这个broker是否有分区的副本leader，如果有那么这个分区就需要一个新的leader，此时控制器就会去遍历其他副本，决定哪一个成为新的leader，同时更新分区的ISR集合。

如果有一个broker加入集群中，那么控制器就会通过Broker ID去判断新加入的broker中是否含有现有分区的副本，如果有，就会从分区副本中去同步数据。

集群中每选举一次控制器，就会通过zookeeper创建一个`controller epoch`，每一个选举都会创建一个更大，包含最新信息的epoch，如果有broker收到比这个epoch旧的数据，就会忽略它们，kafka也通过这个epoch来防止集群产生“脑裂”。

> 原文链接：https://blog.csdn.net/qq_37142346/article/details/91349100
> controller作用: 维护ISR集合,选举leader,增加分区时的重新分配工作

2.2 分区副本选举机制

在kafka的集群中，会存在着多个主题topic，在每一个topic中，又被划分为多个partition，为了防止数据不丢失，每一个partition又有多个副本，在整个集群中，总共有三种副本角色：

- 首领副本（leader）：也就是leader主副本，每个分区都有一个首领副本，所有的生产者与消费者的请求都会经过该副本来处理。
- 跟随者副本（follower）：除了首领副本外的其他所有副本都是跟随者副本，跟随者副本不处理来自客户端的任何请求，只负责从首领副本同步数据，保证与首领保持一致。如果首领副本发生崩溃，就会从这其中选举出一个leader。
- 首选首领副本：创建分区时指定的首选首领。如果不指定，则为分区的第一个副本。

我们希望每个分区的leader可以分布到不同的broker中，尽可能的达到负载均衡，所以会有一个首选首领，如果我们设置参数`auto.leader.rebalance.enable`为true，那么它会检查首选首领是否是真正的首领，如果不是，则会触发选举，让首选首领成为首领(如果设置了首选首领,则一定会让它成为leader)。

> 原文链接：https://blog.csdn.net/qq_37142346/article/details/91349100

Kafka在ZooKeeper中动态维护了一个ISR（in-sync replicas），这个ISR里的所有Replica都复制了leader，只有ISR里的成员才有被选为Leader的可能。 默认的，如果follower与leader之间超过10s内没有发送请求，或者说两者数据差太多(指的是条数,估计是用offset来判断,差值可配置,默认为4000)，此时该follower就会被认为“不同步副本”(Out-Sync Relipcas)。而持续请求的副本就是“同步副本”，当leader发生故障时，会从“同步副本”(In-Sync Replicas)中选举为leader。其中的请求超时时间可以通过参数`replica.lag.time.max.ms`参数来配置。

由一个控制器认定谁是leader,

在ISR中至少有一个follower时，Kafka可以确保已经commit的数据不丢失，如果由于服务宕机,导致某个分区的所有副本都失效,就无法保证数据不丢失了。这种情况下有两种可行的方案：

1. 等待ISR中的任一个Replica“活”过来，并且选它作为Leader
2. 选择第一个“活”过来的Replica（不一定是ISR中的）作为Leader

如果一定要等待ISR中的Replica“活”过来，那不可用的时间就可能会相对较长。而且如果ISR中的所有Replica都无法“活”过来了，或者数据都丢失了，这个Partition将永远不可用。

选择第一个“活”过来的Replica作为Leader，而这个Replica不是ISR中的Replica，那即使它并不保证已经包含了所有已commit的消息，它也会成为Leader而作为consumer的数据源（前文有说明，所有读写都由Leader完成）。(此时称为脏leader选举)

Kafka0.8.*使用了第二种方式。根据Kafka的文档，在以后的版本中，Kafka支持用户通过配置选择这两种方式中的一种，从而根据不同的使用场景选择高可用性还是强一致性。

`unclean.leader.election.enable` 参数决定使用哪种方案，默认是true，采用第二种方案

> 链接：https://www.jianshu.com/p/1f02328a4f2e
> 链接: https://www.cnblogs.com/qingyunzong/p/9004703.html

> 这个leader的作用是接收读写操作,follwer只是个副本,如果leader挂了,则其中之一成为leader,继续接收读写操作

2.3 消费组选主

在kafka的消费端，会有一个消费者协调器以及消费组，组协调器GroupCoordinator需要为消费组内的消费者选举出一个消费组的leader，那么如何选举的呢？

如果消费组内还没有leader，那么第一个加入消费组的消费者即为消费组的leader，如果某一个时刻leader消费者由于某些原因退出了消费组，那么就会重新选举leader，如何选举？

> 原文链接：https://blog.csdn.net/qq_37142346/article/details/91349100

消费组里有**Rebalance** 过程,做的是consumer如何达成一致来分配订阅topic的每个分区,其中就得先选leader,具体看 原理介绍