Kafka数据可靠性与一致性解析

1.Partition Recovery机制

每个Partition会在磁盘记录一个RecoveryPoint， 记录已经flush到磁盘的最大offset。当broker fail 重启时，会进行loadLogs。 首先会读取该Partition的RecoveryPoint，找到包含RecoveryPoint的segment及以后的segment， 这些segment就是可能没有完全flush到磁盘segments。然后调用segment的recover，重新读取各个segment的msg，并重建索引。

优点

- 以segment为单位管理Partition数据，方便数据生命周期的管理，删除过期数据简单
- 在程序崩溃重启时，加快recovery速度，只需恢复未完全flush到磁盘的segment
- 通过index中offset与物理偏移映射，用二分查找能快速定位msg，并且通过分多个Segment，每个index文件很小，查找速度更快。

2.Partition Replica同步机制

- Partition的多个replica中一个为Leader，其余为follower
- Producer只与Leader交互，把数据写入到Leader中
- Followers从Leader中拉取数据进行数据同步
- Consumer只从Leader拉取数据

ISR：所有不落后的replica集合， 不落后有两层含义：距离上次FetchRequest的时间不大于某一个值或落后的消息数不大于某一个值， Leader失败后会从ISR中选取一个Follower做Leader

关于replica复制详解请移步：Kafka副本同步机制理解

3.数据可靠性保证

当Producer向Leader发送数据时，可以通过acks参数设置数据可靠性的级别

- 0：不论写入是否成功，server不需要给Producer发送Response，如果发生异常，server会终止连接，触发Producer更新meta数据；
- 1：Leader写入成功后即发送Response，此种情况如果Leader fail，会丢失数据
- -1：等待所有ISR接收到消息后再给Producer发送Response，这是最强保证
  仅设置acks=-1也不能保证数据不丢失，当Isr列表中只有Leader时，同样有可能造成数据丢失。

要保证数据不丢除了设置acks=-1， 还要保 证ISR的大小大于等于2，具体参数设置：

1. request.required.acks：设置为-1 等待所有ISR列表中的Replica接收到消息后采算写成功；
2. min.insync.replicas： 设置为大于等于2，保证ISR中至少有两个Replica Producer要在吞吐率和数据可靠性之间做一个权衡

4.数据一致性保证

一致性定义：若某条消息对Consumer可见，那么即使Leader宕机了，在新Leader上数据依然可以被读到

- HighWaterMark简称HW： Partition的高水位，取一个partition对应的ISR中最小的LEO作为HW，消费者最多只能消费到HW所在的位置，另外每个replica都有highWatermark，leader和follower各自负责更新自己的highWatermark状态，highWatermark <= leader. LogEndOffset
- 对于Leader新写入的msg，Consumer不能立刻消费，Leader会等待该消息被所有ISR中的replica同步后，更新HW，此时该消息才能被Consumer消费，即Consumer最多只能消费到HW位置

这样就保证了如果Leader Broker失效，该消息仍然可以从新选举的Leader中获取。对于来自内部Broker的读取请求，没有HW的限制。同时，Follower也会维护一份自己的HW，Folloer.HW = min(Leader.HW， Follower.offset)