3. 消费组订阅的主题每个分区只能分配给消费组一个消费者。

bootstrap.servers

Kafka

```
key.deserializer
                                  value.deserializer
                       参数配置
                                  client.id
                                  group.id
             消费者
                                  auto.offset.reset
                                  enable.auto.commit
                                             有可能会造成重复消费
                                  自动提交
                                                         调用 commitSync 时,Consumer 处于阻塞状态,直到 Broker 返回结果
                       位移提交
                                                        会影响 TPS
                                             同步提交
                                                                                     会导致 Consumer 的提交频率下降
                                                         可以选择拉长提交间隔,但有以下问题
                                  手动提交
                                                                                    Consumer 重启后,会有更多的消息被消费
                                                        不会出现自动重试问题
                                             异步提交
                                      是一个规定了如何让消费者组下的所有消费者来分配 topic 中的每一个分区
                                                 1. 消费者组内成员发生变更,这个变更包括了增加和减少消费者,比如消费
                                                 者宕机退出消费组。
                                                 2. 主题的分区数发生变更,kafka目前只支持增加分区,当增加的时候就会触
                                      触发条件
                                                 3. 订阅的主题发生变化,当消费者组使用正则表达式订阅主题,而恰好又新
                       重平衡 / 再平衡
                                                 建了对应的主题,就会触发重平衡
                                               重平衡过程中,消费者无法从kafka消费消息,这对kafka的
                                               TPS影响极大,而如果kafka集内节点较多,比如数百个,那重平衡可能会耗
                                              时极多。数分钟到数小时
                                              都有可能,而这段时间kafka基本处于不可用状态。
                                                     session.timout.ms 控制心跳超时时间,
                                                                                  建议 6s
                                      避免重平衡办法
                                                     heartbeat.interval.ms 控制心跳发送频率,
                                                                                    建议 2s
                                                                                  建议消费者处理消息最长耗时再加 1 分钟
                                                     max.poll.interval.ms 控制poll的间隔。
                                                                    cleanup.policy
                                                                   compression.type
                                                                   delete.retention.ms
                                                                   file.delete.delay.ms
                                                                   flush.messages
                                                                   follower.replication.throttled.replicas
                                                                    index.interval.bytes
                                                                    leader.replication.throttled.replicas
                                                                    max.message.bytes
                                                                    message.format.version
                                                                    message.timestamp.difference.max.ms
                                  --config:为创建的或修改的主题指定配置信息。
                                                                    message.timestamp.type
                                                                    min.compaction.lag.ms
                                                                    min.insync.replicas
高级特性分析
                                                                    preallocate
                                                                    retention.bytes
                                                                    retention.ms
                                                                    segment.bytes
                                                                   segment.index.bytes
                                                                   segment.jitter.ms
                                                                    segment.ms
                                                                    unclean.leader.election.enable
                                  --create
                                  --delete
                                  --delete-config
                                           通过该命令只能增加分区,不能减少,否则会报错
                     参数配置
                                  --disable-rack-aware
                                  --if-exists
                                  --if-not-exists
                                 --replica-assignment <String:broker_id_for_part1_replica1
                                 :broker_id_for_part1_replica2 ,broker_id_for_part2_replica1
                                 :broker_id_for_part2_replica2 , ...>
                                                                                   当创建或修改主题的时候手动指定partition-to-broker的分配关系。
                                                                     1. 均衡地将副本分散于各个broker上
                                                                    2. 对于某个broker上分配的分区,它的其他副本在其他broker上
                                                   副本分配的三个目标
                                  --replication-factor
                                                                     3. 如果所有的broker都有机架信息,尽量将分区的各个副本分配到不同机架
                                  --topics-with-overrides
                                  --unavailable-partitions
                                  --under-replicated-partitions
                                     需要先知道 Kafka 是会在 ZooKeeper 上针对每一个 topic 维护一个称为 ISR
                                      (In-Sync Replicas)的集合,该集合中是一些分区的副本,只有当这些副本
                                      都跟 Leader 中的副本同步后,Kafka 才会认为消息成功提交,并反馈消息给
                                     生产者
                                      所以当某个分区所在的服务区出问题后,Kafka 就会从 ISR 集合中选择一个
                                      副本选举为新的 Leader
                                     这样做的好处就是 Kafka 的冗余度较低,可以容忍的失败数很高,如果某个
             比较浪费资源,比如某个 topic 有 N 个副本,如果是半数选举,它只允许 (N
                                      为什么不用 ZooKeeper 的半数选举?
                                                                 - 1)/2 台机器出问题,容忍度低
                                                              1. 等待ISR集合中的副本复活,
                                                                                     该方法可能耗时较长
                                     如果 ISR 副本都失败了怎么办?
                                                              2. 选择任何一个立即可用的副本,而这个副本不一定是在ISR集合中。需要设
                                                             置 unclean.leader.election.enable=true。
                                                                                                                该方法可能会造成数据丢失
                                                          1. 分区日志文件中包含很多的 LogSegment
                                                          2. Kafka 日志追加是顺序写入的
                                                          3. LogSegment 可以减小日志文件的大小
                                                  概述
                                                          4. 进行日志删除的时候和数据查找的时候可以快速定位。
                                                          5. ActiveLogSegment 是活跃的日志分段,拥有文件拥有写入权限,其余的
                                                         LogSegment 只有只读的权限。
                                                 每个 LogSegment 都有一个基准偏移量,表示当前 LogSegment 中第一条
                                                 消息的 offset。
                                    LogSegment
                                                                 1. 当前日志分段文件的大小超过了 broker 端参数 log.segment.bytes 配置
                                                                 的值。log.segment.bytes 参数的默认值为 1073741824,即 1GB。
                                                                 2. 当前日志分段中消息的最大时间戳与当前系统的时间戳的差值大于
                                                                 log.roll.ms 或 log.roll.hours 参数配置的值。如果同时配置了 log.roll.ms 和
                                                                 log.roll.hours 参数,那么 log.roll.ms 的优先级高。默认情况下,只配置了
                                                                 log.roll.hours 参数,其值为168,即 7 天。
                                                                 3. 偏移量索引文件或时间戳索引文件的大小达到 broker 端参数
                                                 切分文件满足条件
                                                                  log.index.size.max.bytes 配置的值。 log.index.size.max.bytes 的默认值为
                         主要角色
                                                                  10485760,即10MB。
                                                                                                                                              1024 * 1024 * 1024=1073741824
                                                                                                                                                                                           相对偏移量:表示消息相对与基准偏移量的偏移量,占4个字节
                                                                 4. 追加的消息的偏移量与当前日志分段的偏移量之间的差值大于
                                                                                                                                              在偏移量索引文件中,每个索引项共占用8个字节,并分为两部分。
                                                                 INTEGER.MAX_VALUE ,即要追加的消息的偏移量不能转变为相对偏移量。
                                                                                                                   为什么是 INTEGER.MAX_VALUE?
                                                                                                                                                                                           物理地址:消息在日志分段文件中对应的物理位置,也占4个字节
                                                                                                                                             4 个字节刚好对应 Integer.MAX_VALUE ,如果大于 Integer.MAX_VALUE
                                                                                                                                               ,则不能用 4 个字节 进行表示了。
                                                                 偏移量索引文件用于记录消息偏移量与物理地址之间的映射关系。
                                    index (索引)
                                                  timestamp / 时间戳
                                                                   时间戳索引文件则根据时间戳查找对应的偏移量。
                                             kafka高性能,是多方面协同的结果,包括宏观架构、分布式partition存储、
                                             ISR数据同步、以及"无所不用其极"的高效利用磁盘/操作系统特性。
             物理存储
                                             零拷贝并不是不需要拷贝,而是减少不必要的拷贝次数。通常是说在IO读写
                                             过程中。
                                    零拷贝
                                                                数据落盘通常都是非实时的,Kafka的数据并不是实时的写入硬盘,它充分利
                                                                用了现代操作系统分页存储来利用内存提高I/O效率。
                                                                1、网络数据持久化到磁盘 (Producer 到 Broker)
                                             kafka 的数据传输过程
                                                                                                    磁盘数据通过DMA(Direct Memory Access,直接存储器访问)拷贝到内核态
                                                                                                    Buffer
                                                                                                    直接通过 DMA 拷贝到 NIC Buffer(socket buffer),无需 CPU 拷贝。
                                                                2、磁盘文件通过网络发送 (Broker 到 Consumer)
                                                                                                    除了减少数据拷贝外,整个读文件 ==> 网络发送由一个 sendfile 调用完
                                                                                                    成,整个过程只有两次上下文切换,因此大大提高了性能。
                         磁盘存储
                                             页缓存是操作系统实现的一种主要的磁盘缓存,以此用来减少对磁盘 I/O 的
                                             操作。具体来说,就是把磁盘中的数据缓存到内存中,把对磁盘的访问变为对
                                             内存的访问。
                                             Memory Mapped Files
                                    页缓存 简称mmap,简单描述其作用就是:将磁盘文件映射到内存,用户通过修改内
                                                                                            它的工作原理是直接利用操作系统的Page来实现磁盘文件到物理内存的直接
                                             存就能修改磁盘文
                                                                                               映射。完成映射之后你对物理内存的操作会被同步到硬盘上(操作系统在适当
                                                                                               的时候)。
```