链滴

Kafka主要参数详解

作者：unhappydepig

原文链接：https://ld246.com/article/1460014965041

来源网站：链滴

############################## System ##############################
#唯一标识在集群中的ID，要求是正数。
broker.id=0
#服务端口，默认9092
ort=9092
#监听地址，不设为所有地址
host.name=debugo01
# 处理网络请求的最大线程数
num.network.threads=2
# 处理磁盘I/O的线程数<b />num.io.threads=8
# 一些后台线程数
background.threads = 4
# 等待IO线处理的请求队列最大数
queued.max.requests = 500
# socket的发送缓冲区（SO_SNDBUF）
socket.send.buffer.bytes=1048576
# scket的接收缓冲区 (SO_RCVBUF)
socket.receive.buffer.bytes=1048576
# socket请的最大字节数。为了防止内存溢出，message.max.bytes必然要小于
socket.request.max.byts = 104857600
############################## Topic ##############################
# 每个topic的分区个数，更多的partition会产生更多的segment file
num.partitions=2
# 是否允许自动创建topic ，若是false，就需要通过命令创建topic
auto.create.topics.enabl =true
# 一个topic ，默认分区的replication个数 ，不能大于集群中broker的个数。
default.replication.factor =1
# 消息体的最大大小，单位是字节
message.max.bytes = 100000
############################## ZooKeeper ##############################
# Zookeeper quorum设置。如果有多个使用逗号分割
zookeeper.connect=debugo1:2181,debugo02,debugo03
# 连接zk的超时时间
zookeeper.connection.timeout.m=1000000
# ZooKeeper集群中leader和follower之间的同步实际
zookeeper.sync.time ms = 2000
############################## Log ##############################
日志存放目录，多个目录使用逗号分割
log.dirs=/var/log/kafka
# 当达到下面的消息数量时，会将数据flush到日志文件中。默认10000
#log.flush.interval messages=10000
当达到下面的时间(ms)时，执行一次强制的flush操作。interval.ms和inteval.messages无论哪个达到，都会flush。默认3000ms
#log.flush.interval.ms=1000
# 检查是否需要将日志flush的时间间隔
log.flush.scheduler.interval.ms = 3000
# 日志清理策略（delete|compact）
log.cleanup.policy = delete
# 日志保存时间hours|minutes)，默认为7天（168小时）。超过这个时间会根据policy处理数据。bytes和minutes论哪个先达到都会触发。
log.retention.hours=168
# 日志数据存储的最大字节数。超这个时间会根据policy处理数据。
#log.retention.bytes=1073741824
# 控制日志segment文件的大小，超出该大小则追加到一个新的日志segment文件中（-1表示有限制）
log.segment.bytes=536870912
# 当达到下面时间，会强制新建一个segmen
log.roll.hours = 24*7
# 日志片段文件的检查周期，查看它们是否达到了删除策略的设（log.retention.hours或log.retention.bytes）
log.retention.check.interval.ms=60000

# 是否开启压缩
log.cleaner.enable=false
# 对于压缩的日志保留的最长时间
log.cleaner.delete.retention.ms = 1 day
# 对于segment日志的索引文件大小限制
log.index.size.max.bytes = 10 * 1024 * 1024
#y索引计算的一个缓冲区，一般不需要设置。
log.index.interval.bytes = 4096
############################## replica ##############################
# partition management controller 与replicas之间通讯的超时时间
controller.socket.timout.ms = 30000
# controller-to-broker-channels消息队列的尺寸大小
controller.message.queue.size=10
# replicas响应leader的最长等待时间，若是超过这个时间，就将replica排除在管理之外
replica.lag.time.max.ms = 10000
# 是否允许控制器关闭broker ,若设置为true,会关闭所有在这个broker上的leader，并转移到其他broker
controlled.shutdown.nable = false
# 控制器关闭的尝试次数
controlled.shutdown.max.retries = 3
# 每次关闭尝试的时间间隔
controlled.shutdown.retry.backoff.ms = 5000
# 如果relicas落后太多,将会认为此partition relicas已经失效。而一般情况下,因为网络延迟等原因总会导致replicas中消息同步滞后。如果消息严重滞后,leader将认为此relicas网络延迟较大或者消息吐能力有限。在broker数量较少,或者网络不足的环境中,建议提高此值.
replica.lag.max.messaes = 4000
#leader与relicas的socket超时时间
replica.socket.timeout.ms= 30 * 1000
# leader复制的socket缓存大小
replica.socket.receive.buffer.bytes=64 * 1024
# replicas每次获取数据的最大字节数
replica.fetch.max.bytes = 1024 * 1024
# replica

同leader之间通信的最大等待时间，失败了会重试
replica.fetch.wait.max.ms = 500
# 一个fetch操作的最小数据尺寸,如果leader中尚未同步的数据不足此值,将会等待直到数据达到这个大
replica.fetch.min.bytes =1
# leader中进行复制的线程数，增大这个数值会增加relipc的IO
num.replica.fetchers = 1
# 每个replica将最高水位进行flush的时间间隔
relica.high.watermark.checkpoint.interval.ms = 5000

# 是否自动平衡broker之间的配策略
auto.leader.rebalance.enable = false
# leader的不平衡比例，若是超过这个值，会对分区进行重新的平衡
leader.imbalance.per.broker.percentage = 10
# 检查leader是否不平衡的时间间隔
leader.imbalance.check.interval.seconds = 300
# 客户端留offset信息的最大空间大小
offset.metadata.max.bytes = 1024
#############################Consumer ############################## br /># Consumer端核心的配置是group.id、zookeeper.connect
决定该Consumer归属唯一组ID，By setting the same group id multiple processes indicate that they are all part of th same consumer group.
group.id
消费者的ID，若是没有设置的话，会自增
consumer.id
# 一个用于跟踪调查的ID ，最好同group.id相同
client.id = <group_id&t;

# 对于zookeeper集群的指定，必须和broker使用同样的zk配置
zookeeper.connect=debugo01:2182,debugo02:2182,debugo03:2182
# zookeeper的心跳超时时间，过这个时间就认为是无效的消费者
zookeeper.session.timeout.ms = 6000
# zookeepr的等待连接时间
zookeeper.connection.timeout.ms = 6000
# zookeeper的followe同leader的同步时间
zookeeper.sync.time.ms = 2000
# 当zookeeper中没有初始的offet时，或者超出offset上限时的处理方式 。
# smallest： 重置为最小值
# largest:重置最大值
# anything else：抛出异常给consumer
auto.offset.reset = largest
# socket的超时时间，实际的超时时间为max.fetch.wait + socket.timeout.ms.
socket.tmeout.ms= 30 * 1000
# socket的接收缓存空间大小
socket.receive.buffer.bytes=64 1024
#从每个分区fetch的消息大小限制
fetch.message.max.bytes = 1024 * 1024

# true时，Consumer会在消费消息后将offset同步到zookeeper，这样当Consumer失败，新的consumer就能从zookeeper获取最新的offset
auto.commit.enable = true
# 动提交的时间间隔
auto.commit.interval.ms = 60 * 1000

# 用于消费的最大数的消息块缓冲大小，每个块可以等同于fetch.message.max.bytes中数值
queued.max.messae.chunks = 10
# 当有新的consumer加入到group时,将尝试reblance,将partitions的消费端迁移到新的consumr中, 该设置是尝试的次数
rebalance.max.retries = 4
# 每次reblance的时间间隔
ebalance.backoff.ms = 2000
# 每次重新选举leader的时间
refresh.leader.backoff.m

# server发送到消费端的最小数据，若是不满足这个数值则会等待直到满足指定大小默认为1表示立即接收。
fetch.min.bytes = 1
# 若是不满足fetch.min.bytes时，等待费端请求的最长等待时间
fetch.wait.max.ms = 100
# 如果指定时间内没有新消息可用消费，就抛出异常，默认-1表示不受限
consumer.timeout.ms = -1
#############################Producer##############################<b /># 核心的配置包括：
metadata.broker.list
request.required.acks
producer.type
serializer.class
消费者获取消息元信息(topics, partitions and replicas)的地址,配置格式是：host1:port1,hos2:port2，也可以在外面设置一个vip
metadata.broker.list

#消息的确认模式
# 0：不保证消息的到达确认，只管发送，低延迟但是会出现消息的丢失，在某个server失败的情下，有点像TCP
# 1：发送消息，并会等待leader 收到确认后，一定的可靠性
# -1：发消息，等待leader收到确认，并进行复制操作后，才返回，最高的可靠性
request.required.acs = 0

# 消息发送的最长等待时间
request.timeout.ms = 10000
# socke的缓存大小
send.buffer.bytes=100*1024
# key的序列化方式，若是没有设置，同serializer.class
key.serializer.class
# 分区的策略，默认是取模
partitioner.class=kafka producer.DefaultPartitioner
# 消息的压缩模式，默认是none，可以有gzip和snappy
ompression.codec = none
# 可以针对默写特定的topic进行压缩
compressed.topics null
# 消息发送失败后的重试次数
message.send.max.retries = 3
# 每次失败的间隔时间
retry.backoff.ms = 100
# 生产者定时更新topic元信息的时间间隔 ，若是置为0，那么会在每个消息发送后都去更新数据
topic.metadata.refresh.interval.ms = 600 * 100
# 用户随意指定，但是不能重复，主要用于跟踪记录消息
client.id=""

 异步模式下缓冲数据的最大时间。例如设置为100则会集合100ms内的消息后发送，这样会提高吞吐

，但是会增加消息发送的延时
queue.buffering.max.ms = 5000
# 异步模式下缓冲的大消息数，同上
queue.buffering.max.messages = 10000
# 异步模式下，消息进入列的等待时间。若是设置为0，则消息不等待，如果进入不了队列，则直接被抛弃
queue.enqueue.timeout.ms = -1
# 异步模式下，每次发送的消息数，当queue.buffering.max.messages queue.buffering.max.ms满足条件之一时producer会触发发送。
batch.num.messages=200 /p>