# kafka 学习

1.分布式的,可分区的,可复制的消息系统,

消息系统基本功能:

消息以topic为单位进行归纳,发布消息producers,消费消息consumer,以集群方式运行,多个服务构成,每个服务称为broker

客户端和服务端通过TCP通信协议

topic是对一组消息的归纳,对于每个topic,kafka对他的日志进行了分区,每个分区都由一系列有序的,不可变的消息组成,这些消息被连续追加到分区中,分区中的每一个消息都有一个连续的序列号叫做offset,用来在分区中唯一的标识这个消息.

在一个可配置的时间段内,kafka集群保留所有的消息,不管这些消息有没有被消费,其中是有一个消息的保存策略 , a如果消息的爆粗策略设置为2天,那么在一个消息被发布的两天时间内,它是可以被消费的.之后他将被丢弃释放空间.kafka的性能是和数据量无关的常量级的,所以保留太多数据不会出现问题.

实际上每个消费者需要维护的数据是消息在日志中的位置,也就是offset.offset由消费者来维护,随着消费者不断读取消息,offset值不断增加,消费者可以以任意顺序读取消息,可以将offset设置成一个旧的值来重复读之前的消息.

总结:kafka的消费者非常轻量级,他们可以在不对集群和其他消费者造成影响的情况下读取消息,可以读取消息,二不会对其他正在消费消息的消费者造成影响.

日志分区的目的:每个日志的数量不会太大,可以在单个服务上保存.每个分区可以单独发布和消费,为并发操作topic提供了一种可能.

2.分布式

每个分区在集群的不同服务器中都有副本,这些持有副本的服务可以共同处理数据和请求.副本数量是可以配置的.副本是kafka拥有;额容错能力.

每个分区都有一个服务器作为leader,若干个服务器作为followers,

leader负责处理消息的读和写,f负责复制L,如果down掉,F中的一台就会自动成为L(机制?),集群中的每个服务都会都会同时作为leader和follower,作为自己分区的leader,其他分区的follower

3.producer

将消息发布到他指定的topic中,负责决定分发到那个分区.通常简单的使用负载均衡机制随机选择分区,但也可以通过特定的分区函数选择分区(常用).

4.Consumers

发布消息两种模式:队列模式(queuing),发布订阅模式(publish,subscribe).队列模式:同时从服务端读取消息,每个消息只被一个一个消费者读到 ;发布订阅模式消息被广播到所有的cosumer中.

所有的consumer可以加入到一个consumer组,共同竞争一个topic.topic中的消息将被分发到组中的一个成员,同一组中的消费者可以在不同的程序中,也可以在不同的机器上.如果所有的consumer都在同一个组中.就是传统的队列模式,在个consumer中实现负载均衡.如果都所有的消费者,都在不同的组中,成为了发布订阅模式.

常见的是,每个topic都有若干数量的consumer组,每个组都是逻辑上的订阅者,为了容错和稳定性.

相比传统的消息系统,kafka可以很好地保持有序性.

传统的消息系统,多个消费者同时消费消息,服务器会以消息存储的顺序向消费者发送消息.虽然服务器是按顺序发布消息,但消息是被异步分发到各个消费者上的,消息到达时已经失去了原有的顺序,意味着并发消费将导致顺序错乱,为了避免故障,这些消息系统通常使用专用消费者的概念,也就是只允许一个消费者消费消息,失去了并发性,kafka通过分区,在并发消费时,提供有序性和负载均衡.每个分区只分发给一个consumer组,分区只被这个组的一个consumer消费,就可以顺序消费这个分区的消息.从发送的源头保证了消息的顺序发送(理解为队列模式的一个集群,使用一个系统).

有多个分区,就可以在多个consumer组中负载均衡,组的数量不允许多于分区的数量,有多少分区允许多少并发.

kafka只能保证一个分区内消息的有序性,在不同分区间是不可以的.

# 环境搭建:

kafka含有内置的zookeeper

1.启动内置的zookeeper ./zookeeper-server-start.sh ../config/zookeeper.properties

2.启动kafka ./kafka-server-start.sh ../config/server.properties

3.创建topic

./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1(副本) --partitions 1(分区) --topic test(名称)

4.查看创建的topic

./kafka-topics.sh --list --zookeeper localhost:2181

出现问题创建和查询都很慢

5.发送消息

./kafka-console-producer.sh --broker-list localhost:9092 --topic test

发送的消息内容

6.消费消息

./kafka-console-consumer.sh --topic test --from-beginning -bootstrap-server localhost:9092

消费者消息时,使用的是发送消息的端口

提供者使用broker-list指定服务器 可以是数组,

消费者使用bootstrap-server指定消费的服务器

某个版本以后不能使用zookeeper指定zookeeper地址,需要使用bootstrap-server指定但是很慢

7.kafka在一段时间后会消失

8.端口

oracle :1521 sql server:1433 mysql:3306 pointbase:9092

db2 : 5000 mongodb:27017 redis :6379 memcached : 11211

=======================================================================

生产者:

配置属性:

metadata.broker.list : kafka暴露的地址,localhost:9092 元数据 broker的集合,可以是单机版,也可以是集群版.

serializer.class : 序列化类, 使用StringEncoder

partitioner.class : 可以是自定义的,也可以是kafka默认的

producer.type : async : 生产者的类型

batch.num.messages : 批量发送消息的数量

queue.buffer.max.ms : 队列中消息缓存的最长时间(仅仅是同步时使用)

queue.buffering.max.messages : 队列中能够缓存的最大消息数量(仅仅是同步时使用)

queue.enqueue.timeout.ms 0表示当queue满时丢,负值表示queue满时block,正值表示queue满时block相应的时间.

通过这些属性来创建生产者的配置对象,

ProducerConfig

通过生产者配置对象,创建生产者.

producer

生产者发送消息

消息的具体对象,KeyedMessage()参数主题topic,分区的相关信息, 具体消息,

Producer.send(keyedMessage)

最后需要将生产者关闭.

/\*\*/通过java代码发送消息,确定能够创建topic

==================================

消费者:

消费者配置文件的属性:

zookeeper.connect : java环境直接连接zookeeper ,命令行时,使用的bootstrap-server 时单个broker(kafka服务器)

group.id : groupid,一个字符串来指示一组consumer所在的组,是连接zookeeper必须的,

auto.offset.reset : 表示如果offset有返回,smallest:自动设置reset到最小的offset,largest,自动设置offset到最大的offset,其他值不允许,会抛出异常.

auto.commit.enable : 如果是true,consumer定期向zookeeper定期写入每个分区的offset

auto.commit.interval.ms 往zookeeper上写offset的

通过配置文件创建消费者连接对象consumerconfig

通过Consumer接口创建消费者连接

Consumer.createJavaConsumerConnector(consumerConfig);

连接相应的topic通过 hashmap传入参数

通过消费者连接创建消息流,

consumerConnector.createMessageStreams(topicCountMap);

通过这个对象通过topic获取第一条,最后得到kafkastream流